Consensus Surfaces for Modeling 3D Objects from Multiple Range Images

Mark D. Wheeler
Apple Computer, Inc.
1 Infinite Loop, MS: 301-3M
Cupertino, CA 95014
email: mdwheel@apple.com

Yoichi Sato
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
email: ysato@cs.cmu.edu

Katsushi Ikeuchi *
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
email: ki@cs.cmu.edu

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 resolution (and, hence, accuracy) of the volume grid. Our results demonstrate that our consensus-surface algorithm can construct accurate geometric models from rather noisy input range data and somewhat imperfect alignment.

*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.

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. 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 in the views
- Complete: models the surface as much as is observable from the sample views

We begin by reviewing three methods which are most closely related to our work and follow that by a brief discussion of other related work. The first three works are similar to our algorithm in that they all make use of volumetric, implicit-surface representation and the marching-cubes algorithm [Lorensen and Cline, 1987] to merge the range-image data from several views into a surface model. The main differences between these algorithms are their methods for computing the signed distance from each voxel (volume element) to the closest surface.

Hoppe et al. [Hoppe *et al.*, 1992] were the first to propose constructing 3D surface models by applying the marching-cubes algorithm [Lorensen and Cline, 1987] 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 the cloud of points. Nearest-neighbor search of the inferred surfaces is used to compute the signed distance from each voxel to the surface of the point set.

Curless and Levoy [Curless and Levoy, 1996] followed Hoppe's general scheme with a few significant departures. First, their method was geared towards using 3D data acquired from range images. Rather than performing a simple search for the closest point from a voxel's center to determine the signed distance, They take a weighted average of the signed dis-

tances from the voxel center to range-image points whose image rays intersect the voxel—integrating the signed distance along these rays through the volume.

The method most similar to our work is that of Hilton et al. [Hilton *et al.*, 1996]. As in our work and Curless and Levoy's work, Hilton et al. generate a volumetric implicit-surface representation from a number of triangle sets obtained from range images. Similarly to our algorithm, Hilton's method uses a nearest-neighbor search to find the surface points from each view which are closest to a given voxel's center. Heuristics are used to determine which closest points to use in computing the signed distance.

The main limitation of the above algorithms is that they do not compensate for noise or extraneous point data—the data is assumed to be part of the object and noise is assumed to be negligible. In addition, each of these methods can suffer from inaccuracy due to their integration strategy (for more details see [Wheeler, 1996]).

There have been several other approaches to this modeling problem; most notably, Soucy and Laurendre [Soucy and Laurendeau, 1992] and Turk and Levoy [Turk and Levoy, 1994] presented methods for piecing together sets of triangulated surfaces. Both methods marked significant advancements in the state of the art, but often perform poorly with respect to noise and alignment errors in the data.

2 Approach

The problem we are tackling in this paper is to build a 3D model, a unified surface representation, from a number of range images of an object. To build a 3D surface model from multiple range images, we must address the following problems:

- View alignment: 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.

Our solution makes use of a volumetric representation to avoid difficulties associated with topology. We will show how the volumetric representation simplifies our data-merging problem—virtually eliminating the topology issue. The volumetric representation can be conveniently converted into a triangulated mesh representation with little loss of geometric accuracy. The merging problem is then a matter of converting our input surface data to the volumetric representation.

The conversion problem is exacerbated by the fact that input surface data from real sensors (e.g., range sensors or stereo) is noisy and, in fact, will contain surfaces that are not part of the object we are interested in modeling. Our method for merging the surfaces into a volumetric representation takes full consideration of these facts to best take advantage of the multiple observations to smooth out the noise and eliminate undesired surfaces from the final model.

One important issue which we do not address in this work is how to select views in order to best cover the surface. The human operator determines the number of views and the object orientation for each view.

The rest of this paper provides the details of our solutions to these problems which combine to form a practical method for building 3D surface models from range images of an object.

3 View Alignment

After taking several range images of an object and converting them to surfaces (described in [Wheeler, 1996]), we need to eventually 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 views are aligned. Next, we consider the problem of merging all the data from these views into a single model of the object's surface.

4 Data Merging

Given a number of triangle sets 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 more or less perfectly aligned, the merging problem is difficult. The problem is that it is difficult to determine how to connect triangles from different surfaces without knowing the actual surface beforehand. There are an exponential num-

ber of ways to connect two triangulated surfaces together, some acceptable and some not acceptable. This 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. Recently, however, several researchers [Hoppe *et al.*, 1992, Curless and Levoy, 1996, Hilton *et al.*, 1996] have moved from trying to connect together surface patches from different views to using volumetric methods which hide the topological problems—making the surface-merging problem more tractable. In the next section we discuss the volumetric method which we use to solve the surface-merging problem.

4.1 Volumetric Modeling and Marching Cubes

Recently, the marching-cubes algorithm [Lorensen and Cline, 1987], an algorithm developed for graphics modeling applications, has made volumetric modeling more useful by virtually eliminating the blocky nature of occupancy grids. The representation used by the marching-cubes algorithm is slightly more complicated than the occupancy-grid representation. Instead of storing a binary value in each voxel to indicate if the cube 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.

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 location of the surface can be interpolated by examining the signed distances of neighboring voxels. Thus, the resulting surface will be relatively smooth and the accuracy of the surface will be greater than the resolution of the volume grid.

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 use 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.

4.2 Consensus-Surface Algorithm

In this section, we will answer 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—one set for each range image of our object—which are aligned in the same coordinate system. The triangle sets are denoted by T_i, where $i = 1, ..., N$, The union of all triangle sets is denoted by $T = \bigcup_i T_i$. 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 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 T.

The naive algorithm for $f(\mathbf{x})$ finds the nearest tri-

angle from all views and uses the distance to that triangle as the magnitude of $f(\mathbf{x})$. 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 idea 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 4 in the results section for an example of the type of noise that may be present). We are now faced with choices on how to proceed. 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.

Inconsistent values for the implicit distances will appear when a voxel center is on or near a surface, since the samples will be randomly scattered about the real surface location. It may become difficult to differentiate between several observations of a single surface or a thin wall. This is especially a problem if the noise is of similar scale to the voxel size.

A more sinister problem for the naive algorithm applied to real images is the existence of noise 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 and other possibly systematic problems of range imaging. This makes it very easy to infer the incorrect distance and more critically the incorrect sign, which will result in very undesirable artifacts in the final surface. One badly oriented triangle can result in a voxel which is assigned $f(\mathbf{x})$ with the incorrect sign. 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. This search for nearby observations can be done efficiently using k-d trees [Friedman *et al.*, 1977].

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 quorum 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 is simply computed by

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

where $\hat{\mathbf{v}}$ and $\hat{\mathbf{n}}$ are the viewing direction (of the given point in the range image) and normal, respectively, 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:

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

14. **return** $\langle \mathbf{p}, \hat{\mathbf{n}}, \omega \rangle$

Algorithm *ClosestConsensusSurface* examines the closest point in each view and searches for its consensus surface if one exists. After computing the closest consensus surfaces for each view, 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 $\mathbf{x}$. If none of the consensus surfaces exist, 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 $T = \bigcup_i T_i$
Output: the point, normal vector, and the sum of the observations confidences $\langle \mathbf{p}, \hat{\mathbf{n}}, \omega \rangle$

1. $\mathbf{p} \leftarrow \mathbf{n} \leftarrow \omega \leftarrow 0$
2. **for** $T_i \subset T$
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 this 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.

4.3 Accuracy and Efficiency

Volumetric modeling involves a tradeoff between accuracy and efficiency. To achieve desired accuracy we must use a dense sampling of the volume. Since the memory requirements of a volume grid is cubic with respect to the density of the sampling for volumetric modeling, the first thing that gets sacrificed is accuracy. With the straightforward use of volume grids, resources are wasted by computing signed distances in parts of the volume that are distant from the surface. For our purposes, the only voxels that we need to examine are those near the surface, a small fraction of the entire volume grid. Curless and Levoy [Curless and Levoy, 1996] alleviate this problem by run length encoding each 2D slice of the volume.

The octree representation [Meagher, 1980] solves both the accuracy and the efficiency 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 more finely in octants only when necessary—near the surface of the object. The octree in practice reduces the $O(n^3)$ storage and computation requirement to $O(n^2)$ since the surfaces of 3D objects are, in general, 2D manifolds in a 3D space.

4.4 Cost of the Consensus-Surface Algorithm

To get a rough estimate of the cost of our model-building algorithm, let us assume that there are N views being merged and that for each view the triangle set T_i has n triangles. Algorithm *ConsensusSurface* computes the closest surface for each view which on average will be an $O(N \log n)$ operation assuming k-d trees [Friedman *et al.*, 1977] are used. Algorithm *ClosestConsensusSurface* computes the closest surface and then the respective consensus surface for each view, which adds up to a cost of $O(N^2 \log n)$. Assuming that an $M \times M \times M$ voxel grid is used, the modeling algorithm will cost

$O(M^3N^2 \log n)$. 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 average cost to $O(M^2N^2 \log n)$.

5 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. In this paper, we show the results obtained for modeling a toy duck. See [Wheeler, 1996] for a complete description of our experiments.

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.

Figures 1-2 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), and three views of the resulting triangulated model. For this example, the 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 5 workstation (a 124 MIPS/49.0 MFLOPS machine).

As an example of what the naive algorithm, Algorithm *ClosestSignedDistance* of Section 4.2, would produce we show the example of the result of the naive algorithm on the duck data set in Figure 3. 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 4 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 surface in very noisy data.

6 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 data merging takes advantage of the volumetric implicit-surface representation and the marching-cubes algorithm to eliminate topological problems.

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

> *What is the closest 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 implicitly address this question, their algorithms do not capture the essence of the problem and produce answers by taking averages of possibly unrelated observations. 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 the troublesome effects of noise and extraneous surface observations in our data. 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.

We have presented the results of our modeling algorithm on a number of example problems. These results demonstrate that our consensus-surface algorithm can construct accurate geometric models from rather noisy input range data and somewhat imperfect alignment.

Acknowledgments

Harry Shum is gratefully acknowledged for helpful discussions we had during the progress of this work.

References

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

[Friedman *et al.*, 1977] 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.

[Hilton *et al.*, 1996] A. Hilton, A.J. Stoddart, J. Illingworth, and T. Windeatt. Reliable surface reconstruction from multiple range images. In *Proceedings of the European Conference*

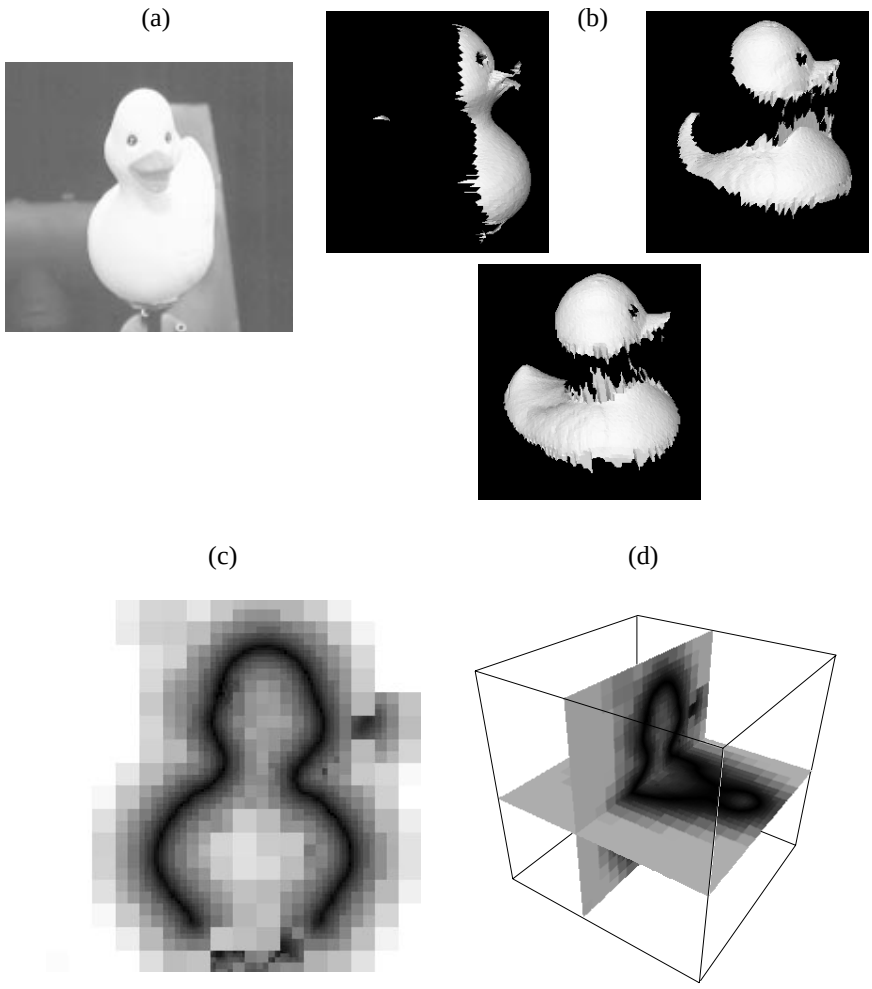

Figure 1: Results from modeling the rubber duck. (a) An intensity image of the duck, (b) a close-up of some of the triangulated range images used as input to the consensus-surface algorithm, (c) a slice of the implicit-surface octree volume where the grey-scale indicates the proximity to a surface point (black closest, white furthest), and (d) a 3D view of two cross sections of the implicit-surface octree volume.

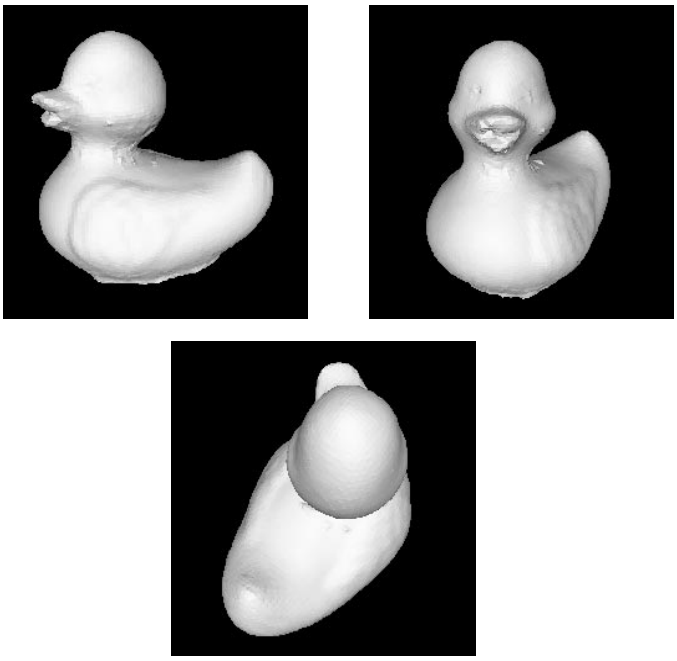

Figure 2: Three views of the resulting triangulated model of the duck.

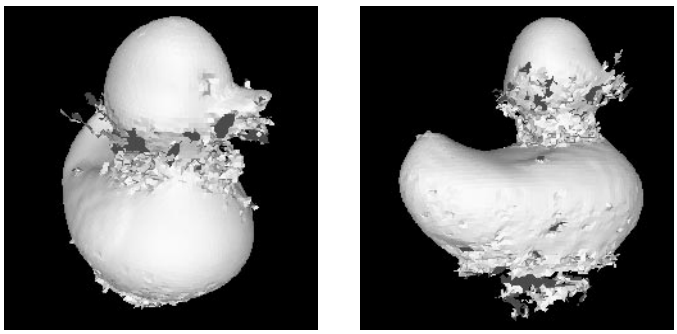

Figure 3: The result of the naive algorithm, Algorithm *ClosestSignedDistance*, on the duck image.

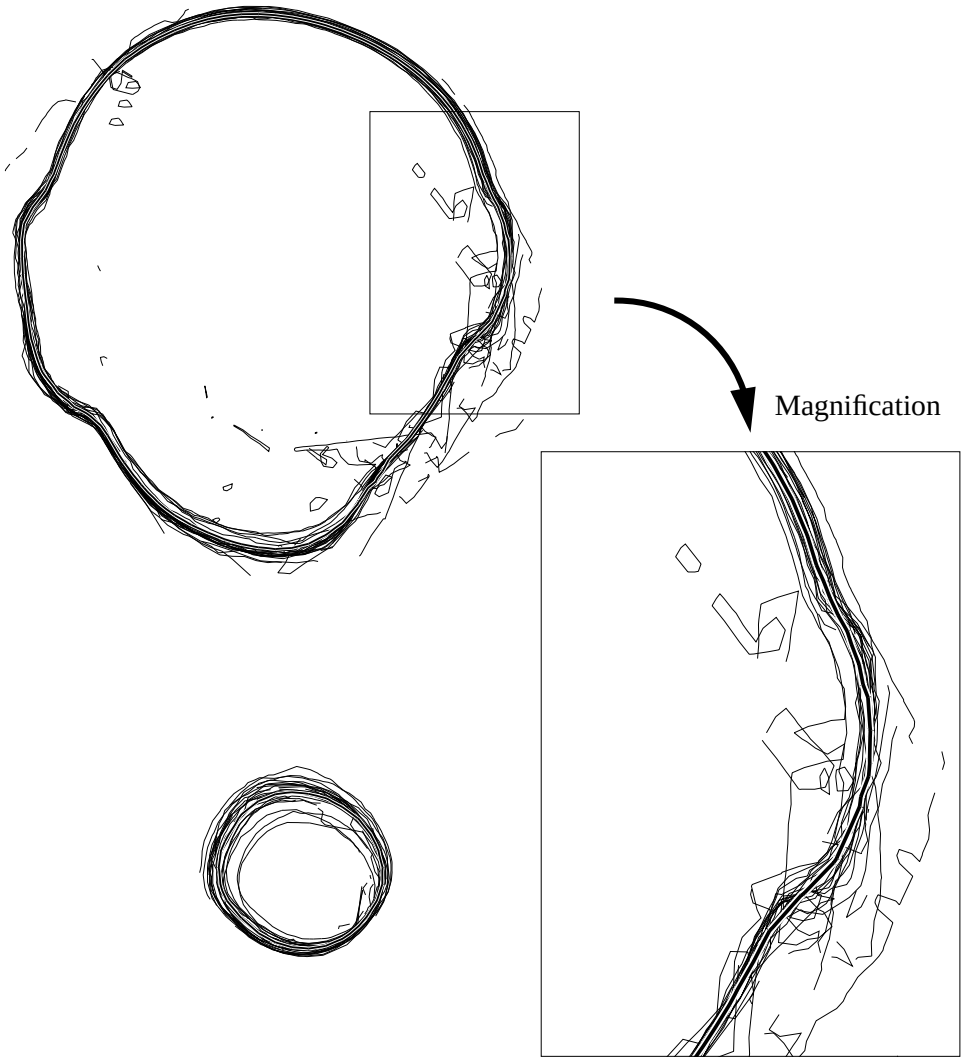

Figure 4: A cross section of the final model of the rubber duck (thick black line) and the original range-image data (thin black lines) used to construct it.

on Computer Vision, pages 117–126. Springer-Verlag, 1996.

[Hoppe *et al.*, 1992] H. Hoppe, T. DeRose, T. Duchamp, J.A. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphcics*, 26(2):71–78, 1992.

[Lorensen and Cline, 1987] 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.

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

[Soucy and Laurendeau, 1992] Marc Soucy and Denis Laurendeau. Multi-resolution surface modeling from multiple range views. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 348–353, 1992.

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

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