A Volumetric Method for Building Complex Models from Range Images

Brian Curless and Marc Levoy
Stanford University

Abstract

A number of techniques have been developed for reconstructing surfaces by integrating groups of aligned range images. A desirable set of properties for such algorithms includes: incremental updating, representation of directional uncertainty, the ability to fill gaps in the reconstruction, and robustness in the presence of outliers. Prior algorithms possess subsets of these properties. In this paper, we present a volumetric method for integrating range images that possesses all of these properties.

Our volumetric representation consists of a cumulative weighted signed distance function. Working with one range image at a time, we first scan-convert it to a distance function, then combine this with the data already acquired using a simple additive scheme. To achieve space efficiency, we employ a run-length encoding of the volume. To achieve time efficiency, we resample the range image to align with the voxel grid and traverse the range and voxel scanlines synchronously. We generate the final manifold by extracting an isosurface from the volumetric grid. We show that under certain assumptions, this isosurface is optimal in the least squares sense. To fill gaps in the model, we tessellate over the boundaries between regions seen to be empty and regions never observed.

Using this method, we are able to integrate a large number of range images (as many as 70) yielding seamless, high-detail models of up to 2.6 million triangles.

CR Categories: I.3.5 [Computer Graphics] Computational Geometry and Object Modeling
Additional keywords: Surface fitting, three-dimensional shape recovery, range image integration, isosurface extraction

1 Introduction

Recent years have witnessed a rise in the availability of fast, accurate range scanners. These range scanners have provided data for applications such as medicine, reverse engineering, and digital film-making. Many of these devices generate *range images*; i.e., they produce depth values on a regular sampling lattice. Figure 1 illustrates how an optical triangulation scanner can be used to acquire a range image. By connecting nearest neighbors with triangular elements, one can construct a *range surface* as shown in Figure 1d. Range images are typically formed by sweeping a 1D or 2D sensor linearly across an object or circularly around it, and generally do not contain enough information to reconstruct the entire object being scanned. Accordingly, we require algorithms that can merge multiple range images into a single description of the surface. A set of desirable properties for such a surface reconstruction algorithm includes:

- *Representation of range uncertainty*. The data in range images typically have asymmetric error distributions with primary directions along sensor lines of sight, as illustrated for optical triangulation in Figure 1a. The method of range integration should reflect this fact.
- *Utilization of all range data*, including redundant observations of each object surface. If properly used, this redundancy can reduce sensor noise.
- *Incremental and order independent updating*. Incremental updates allow us to obtain a reconstruction after each scan or small set of scans and allow us to choose the next best orientation for scanning. Order independence is desirable to ensure that results are not biased by earlier scans. Together, they allow for straightforward parallelization.
- *Time and space efficiency*. Complex objects may require many range images in order to build a detailed model. The range images and the model must be represented efficiently and processed quickly to make the algorithm practical.
- *Robustness*. Outliers and systematic range distortions can create challenging situations for reconstruction algorithms. A robust algorithm needs to handle these situations without catastrophic failures such as holes in surfaces and self-intersecting surfaces.
- *No restrictions on topological type*. The algorithm should not assume that the object is of a particular genus. Simplifying assumptions such as "the object is homeomorphic to a sphere" yield useful results in only a restricted class of problems.
- *Ability to fill holes in the reconstruction*. Given a set of range images that do not completely cover the object, the surface reconstruction will necessarily be incomplete. For some objects, no amount of scanning would completely cover the object, because some surfaces may be inaccessible to the sensor. In these cases, we desire an algorithm that can automatically fill these holes with plausible surfaces, yielding a model that is both "watertight" and esthetically pleasing.

In this paper, we present a volumetric method for integrating range images that possesses all of these properties. In the next section, we review some previous work in the area of surface reconstruction. In section 3, we describe the core of our volumetric algorithm. In section 4, we show how this algorithm can be used to fill gaps in the reconstruction using knowledge about the emptiness of space. Next, in section 5, we describe how we implemented our volumetric approach so as to keep time and space costs reasonable. In section 6, we show the results of surface reconstruction from many range images of complex objects. Finally, in section 7 we conclude and discuss limitations and future directions.

2 Previous work

Surface reconstruction from dense range data has been an active area of research for several decades. The strategies have proceeded along two basic directions: reconstruction from unorganized points, and

Authors' Address: Computer Science Department, Stanford University, Stanford, CA 94305
E-mail: {curless,levoy}@cs.stanford.edu
World Wide Web: http://www-graphics.stanford.edu

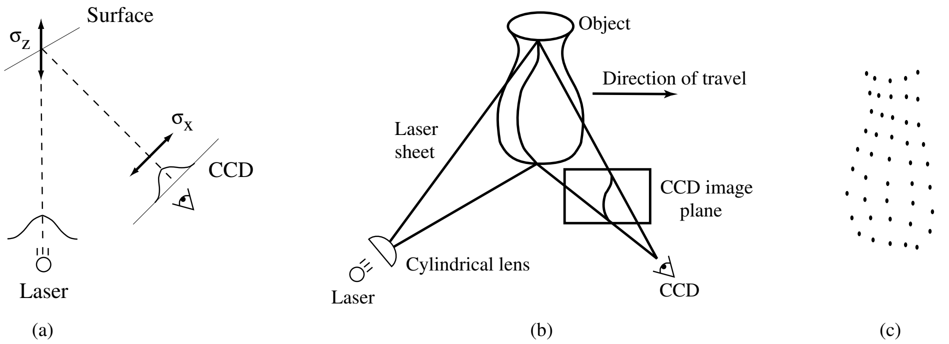


Figure 1. From optical triangulation to a range surface. (a) In 2D, a narrow laser beam illuminates a surface, and a linear sensor images the reflection from an object. The center of the image pulse maps to the center of the laser, yielding a range value. The uncertainty, σ_x, in determining the center of the pulse results in range uncertainty, σ_z along the laser's line of sight. When using the spacetime analysis for optical triangulation [6], the uncertainties run along the lines of sight of the CCD. (b) In 3D, a laser stripe triangulation scanner first spreads the laser beam into a sheet of light with a cylindrical lens. The CCD observes the reflected stripe from which a depth profile is computed. The object sweeps through the field of view, yielding a range image. Other scanner configurations rotate the object to obtain a cylindrical scan or sweep a laser beam or stripe over a stationary object. (c) A range image obtained from the scanner in (b) is a collection of points with regular spacing. (d) By connecting nearest neighbors with triangles, we create a piecewise linear range surface.

reconstruction that exploits the underlying structure of the acquired data. These two strategies can be further subdivided according to whether they operate by reconstructing parametric surfaces or by reconstructing an implicit function.

A major advantage of the unorganized points algorithms is the fact that they do not make any prior assumptions about connectivity of points. In the absence of range images or contours to provide connectivity cues, these algorithms are the only recourse. Among the parametric surface approaches, Boissanat [2] describes a method for Delaunay triangulation of a set of points in 3-space. Edelsbrunner and Mücke [9] generalize the notion of a convex hull to create surfaces called alpha-shapes. Examples of implicit surface reconstruction include the method of Hoppe, et al [16] for generating a signed distance function followed by an isosurface extraction. More recently, Bajaj, et al [1] used alpha-shapes to construct a signed distance function to which they fit implicit polynomials. Although unorganized points algorithms are widely applicable, they discard useful information such as surface normal and reliability estimates. As a result, these algorithms are well-behaved in smooth regions of surfaces, but they are not always robust in regions of high curvature and in the presence of systematic range distortions and outliers.

Among the structured data algorithms, several parametric approaches have been proposed, most of them operating on range images in a polygonal domain. Soucy and Laurendeau [25] describe a method using Venn diagrams to identify overlapping data regions, followed by re-parameterization and merging of regions. Turk and Levoy [30] devised an incremental algorithm that updates a reconstruction by eroding redundant geometry, followed by zippering along the remaining boundaries, and finally a consensus step that reintroduces the original geometry to establish final vertex positions. Rutishauser, et al [24] use errors along the sensor's lines of sight to establish consensus surface positions followed by a re-tessellation that incorporates redundant data. These algorithms typically perform better than unorganized point algorithms, but they can still fail catastrophically in areas of high curvature, as exemplified in Figure 9.

Several algorithms have been proposed for integrating structured data to generate implicit functions. These algorithms can be classified as to whether voxels are assigned one of two (or three) states or are samples of a continuous function. Among the discrete-state volumetric algorithms, Connolly [4] casts rays from a range image accessed as a quad-tree into a voxel grid stored as an octree, and generates results for synthetic data. Chien, et al [3] efficiently generate octree models under the severe assumption that all views are taken from the directions corresponding to the 6 faces of a cube. Li and Crebbin [19] and Tarbox and Gottschlich [28] also describe methods for generating binary voxel grids from range images. None of these methods has been used to generate surfaces. Further, without an underlying continuous function, there are no mechanism for representing range uncertainty or for combining overlapping, noisy range surfaces.

The last category of our taxonomy consists of implicit function methods that use samples of a continuous function to combine structured data. Our method falls into this category. Previous efforts in this area include the work of Grosso, et al [12], who generate depth maps from stereo and average them into a volume with occupancy ramps of varying slopes corresponding to uncertainty measures; they do not, however, perform a final surface extraction. Succi, et al [26] create depth maps from stereo and optical flow and integrate them volumetrically using a straight average. The details of his method are unclear, but they appear to extract an isosurface at an arbitrary threshold. In both the Grosso and Succi papers, the range maps are sparse, the directions of range uncertainty are not characterized, they use no time or space optimizations, and the final models are of low resolution. Recently, Hilton, et al [14] have developed a method similar to ours in that it uses weighted signed distance functions for merging range images, but it does not address directions of sensor uncertainty, incremental updating, space efficiency, and characterization of the whole space for potential hole filling, all of which we believe are crucial for the success of this approach.

Other relevant work includes the method of probabilistic occupancy grids developed by Elfes and Matthies [10]. Their volumetric space is a scalar probability field which they update using a Bayesian formulation. The results have been used for robot navigation, but not for surface extraction. A difficulty with this technique is the fact that the best description of the surface lies at the peak or ridge of the probability function, and the problem of ridge-finding is not one with robust solutions [8]. This is one of our primary motivations for taking an isosurface approach in the next section: it leverages off of well-behaved surface extraction algorithms.

The discrete-state implicit function algorithms described above also have much in common with the methods of extracting volumes from silhouettes [15] [21] [23] [27]. The idea of using backdrops to help carve out the emptiness of space is one we demonstrate in section 4.

3 Volumetric integration

Our algorithm employs a continuous implicit function, $D(\mathbf{x})$, represented by samples. The function we represent is the weighted signed distance of each point $\mathbf{x}$ to the nearest range surface along the line of

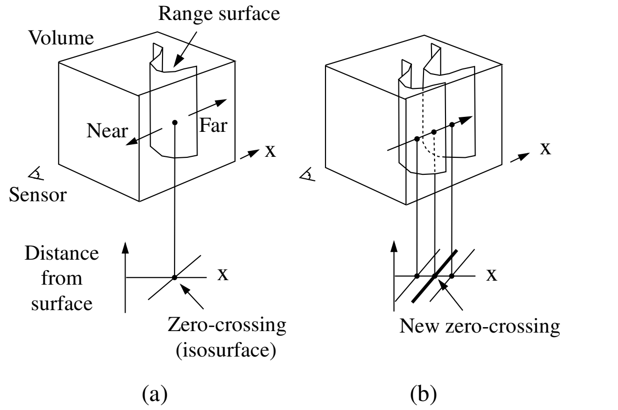


Figure 2. Unweighted signed distance functions in 3D. (a) A range sensor looking down the x-axis observes a range image, shown here as a reconstructed range surface. Following one line of sight down the x-axis, we can generate a signed distance function as shown. The zero crossing of this function is a point on the range surface. (b) The range sensor repeats the measurement, but noise in the range sensing process results in a slightly different range surface. In general, the second surface would interpenetrate the first, but we have shown it as an offset from the first surface for purposes of illustration. Following the same line of sight as before, we obtain another signed distance function. By summing these functions, we arrive at a cumulative function with a new zero crossing positioned midway between the original range measurements.

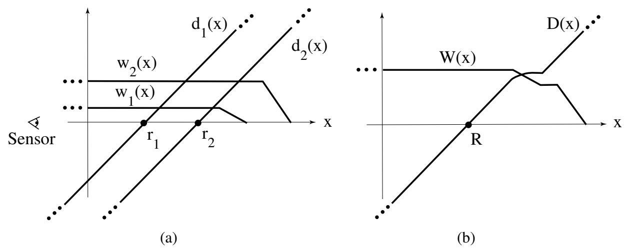


Figure 3. Signed distance and weight functions in one dimension. (a) The sensor looks down the x-axis and takes two measurements, r_1 and r_2. $d_1(x)$ and $d_2(x)$ are the signed distance profiles, and $w_1(x)$ and $w_2(x)$ are the weight functions. In 1D, we might expect two sensor measurements to have the same weight magnitudes, but we have shown them to be of different magnitude here to illustrate how the profiles combine in the general case. (b) $D(x)$ is a weighted combination of $d_1(x)$ and $d_2(x)$, and $W(x)$ is the sum of the weight functions. Given this formulation, the zero-crossing, R, becomes the weighted combination of r_1 and r_2 and represents our best guess of the location of the surface. In practice, we truncate the distance ramps and weights to the vicinity of the range points.

sight to the sensor. We construct this function by combining signed distance functions $d_1(\mathbf{x})$, $d_2(\mathbf{x})$, ... $d_n(\mathbf{x})$ and weight functions $w_1(\mathbf{x})$, $w_2(\mathbf{x})$, ... $w_n(\mathbf{x})$ obtained from range images 1 ... n. Our combining rules give us for each voxel a cumulative signed distance function, $D(\mathbf{x})$, and a cumulative weight $W(\mathbf{x})$. We represent these functions on a discrete voxel grid and extract an isosurface corresponding to $D(\mathbf{x}) = 0$. Under a certain set of assumptions, this isosurface is optimal in the least squares sense. A full proof of this optimality is beyond the scope of this paper, but a sketch appears in appendix A.

Figure 2 illustrates the principle of combining unweighted signed distances for the simple case of two range surfaces sampled from the same direction. Note that the resulting isosurface would be the surface created by averaging the two range surfaces along the sensor's lines of sight. In general, however, weights are necessary to represent variations in certainty across the range surfaces. The choice of weights should be specific to the range scanning technology. For optical triangulation scanners, for example, Soucy [25] and Turk [30] make the weight depend on the dot product between each vertex normal and the viewing direction, reflecting greater uncertainty when the illumination is at grazing angles to the surface. Turk also argues that the range data at the boundaries of the mesh typically have greater uncertainty, requiring more down-weighting. We adopt these same weighting schemes for our optical triangulation range data.

Figure 3 illustrates the construction and usage of the signed distance and weight functions in 1D. In Figure 3a, the sensor is positioned at the origin looking down the +x axis and has taken two measurements, r_1 and r_2. The signed distance profiles, $d_1(x)$ and $d_2(x)$ may extend indefinitely in either direction, but the weight functions, $w_1(x)$ and $w_2(x)$, taper off behind the range points for reasons discussed below.

Figure 3b is the weighted combination of the two profiles. The combination rules are straightforward:

$$D(\mathbf{x}) = \frac{\Sigma w_i(\mathbf{x}) d_i(\mathbf{x})}{\Sigma w_i(\mathbf{x})} \tag{1}$$

$$W(\mathbf{x}) = \Sigma w_i(\mathbf{x}) \tag{2}$$

where, $d_i(\mathbf{x})$ and $w_i(\mathbf{x})$ are the signed distance and weight functions from the ith range image.

Expressed as an incremental calculation, the rules are:

$$D_{i+1}(\mathbf{x}) = \frac{W_i(\mathbf{x}) D_i(\mathbf{x}) + w_{i+1}(\mathbf{x}) d_{i+1}(\mathbf{x})}{W_i(\mathbf{x}) + w_{i+1}(\mathbf{x})} \tag{3}$$

$$W_{i+1}(\mathbf{x}) = W_i(\mathbf{x}) + w_{i+1}(\mathbf{x}) \tag{4}$$

where $D_i(\mathbf{x})$ and $W_i(\mathbf{x})$ are the cumulative signed distance and weight functions after integrating the ith range image.

In the special case of one dimension, the zero-crossing of the cumulative function is at a range, R given by:

$$R = \frac{\Sigma w_i r_i}{\Sigma w_i} \tag{5}$$

i.e., a weighted combination of the acquired range values, which is what one would expect for a least squares minimization.

In principle, the distance and weighting functions should extend indefinitely in either direction. However, to prevent surfaces on opposite sides of the object from interfering with each other, we force the weighting function to taper off behind the surface. There is a trade-off involved in choosing where the weight function tapers off. It should persist far enough behind the surface to ensure that all distance ramps will contribute in the vicinity of the final zero crossing, but, it should also be as narrow as possible to avoid influencing surfaces on the other side. To meet these requirements, we force the weights to fall off at a distance equal to half the maximum uncertainty interval of the range measurements. Similarly, the signed distance and weight functions need not extend far in front of the surface. Restricting the functions to the vicinity of the surface yields a more compact representation and reduces the computational expense of updating the volume.

In two and three dimensions, the range measurements correspond to curves or surfaces with weight functions, and the signed distance ramps have directions that are consistent with the primary directions of sensor uncertainty. The uncertainties that apply to range image integration include errors in alignment between meshes as well as errors inherent in the scanning technology. A number of algorithms for aligning sets of range images have been explored and shown to yield excellent results [11][30]. The remaining error lies in the scanner itself. For optical triangulation scanners, for example, this error has been shown to be ellipsoidal about the range points, with the major axis of the ellipse aligned with the lines of sight of the laser [13][24].

Figure 4 illustrates the two-dimensional case for a range curve derived from a single scan containing a row of range samples. In practice, we use a fixed point representation for the signed distance func-

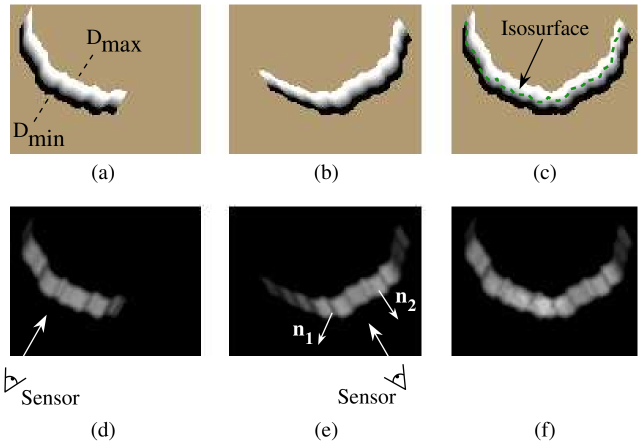


Figure 4. Combination of signed distance and weight functions in two dimensions. (a) and (d) are the signed distance and weight functions, respectively, generated for a range image viewed from the sensor line of sight shown in (d). The signed distance functions are chosen to vary between D_{min} and D_{max}, as shown in (a). The weighting falls off with increasing obliquity to the sensor and at the edges of the meshes as indicated by the darker regions in (e). The normals, $\mathbf{n_1}$ and $\mathbf{n_2}$ shown in (e), are oriented at a grazing angle and facing the sensor, respectively. Note how the weighting is lower (darker) for the grazing normal. (b) and (e) are the signed distance and weight functions for a range image of the same object taken at a 60 degree rotation. (c) is the signed distance function $D(\mathbf{x})$ corresponding to the per voxel weighted combination of (a) and (b) constructed using equations 3 and 4. (f) is the sum of the weights at each voxel, $W(\mathbf{x})$. The dotted green curve in (c) is the isosurface that represents our current estimate of the shape of the object.

tion, which bounds the values to lie between D_{min} and D_{max} as shown in the figure. The values of D_{min} and D_{max} must be negative and positive, respectively, as they are on opposite sides of a signed distance zero-crossing.

For three dimensions, we can summarize the whole algorithm as follows. First, we set all voxel weights to zero, so that new data will overwrite the initial grid values. Next, we tessellate each range image by constructing triangles from nearest neighbors on the sampled lattice. We avoid tessellating over step discontinuities (cliffs in the range map) by discarding triangles with edge lengths that exceed a threshold. We must also compute a weight at each vertex as described above.

Once a range image has been converted to a triangle mesh with a weight at each vertex, we can update the voxel grid. The signed distance contribution is computed by casting a ray from the sensor through each voxel near the range surface and then intersecting it with the triangle mesh, as shown in figure 5. The weight is computed by linearly interpolating the weights stored at the intersection triangle's vertices. Having determined the signed distance and weight we can apply the update formulae described in equations 3 and 4.

At any point during the merging of the range images, we can extract the zero-crossing isosurface from the volumetric grid. We restrict this extraction procedure to skip samples with zero weight, generating triangles only in the regions of observed data. We will relax this restriction in the next section.

4 Hole filling

The algorithm described in the previous section is designed to reconstruct the observed portions of the surface. Unseen portions of the surface will appear as holes in the reconstruction. While this result is an accurate representation of the known surface, the holes are esthetically unsatisfying and can present a stumbling block to follow-on algorithms that expect continuous meshes. In [17], for example, the authors describe a method for parameterizing patches that entails generating evenly spaced grid lines by walking across the edges of a mesh. Gaps in the mesh prevent the algorithm from creating a fair parameterization. As another example, rapid prototyping technologies such as stereolithography typically require a "watertight" model in order to construct a solid replica [7].

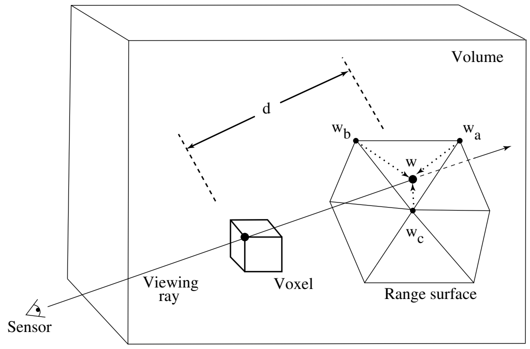


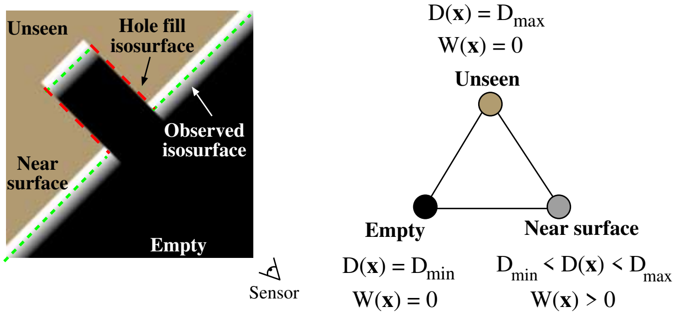
Figure 5. Sampling the range surface to update the volume. We compute the weight, w, and signed distance, d, needed to update the voxel by casting a ray from the sensor, through the voxel onto the range surface. We obtain the weight, w, by linearly interpolating the weights (w_a, w_b, and w_c) stored at neighboring range vertices. Note that for a translating sensor (like our Cyberware scanner), the sensor point is different for each column of range points.

One option for filling holes is to operate on the reconstructed mesh. If the regions of the mesh near each hole are very nearly planar, then this approach works well. However, holes in the meshes can be (and frequently are) highly non-planar and may even require connections between unconnected components. Instead, we offer a hole filling approach that operates on our volume, which contains more information than the reconstructed mesh.

The key to our algorithm lies in classifying all points in the volume as being in one of three states: unseen, empty, or near the surface. Holes in the surface are indicated by frontiers between unseen regions and empty regions (see Figure 6). Surfaces placed at these frontiers offer a plausible way to plug these holes (dotted in Figure 6). Obtaining this classification and generating these hole fillers leads to a straightforward extension of the algorithm described in the previous section:

1. Initialize the voxel space to the "unseen" state.
2. Update the voxels near the surface as described in the previous section. As before, these voxels take on continuous signed distance and weight values.
3. Follow the lines of sight back from the observed surface and mark the corresponding voxels as "empty". We refer to this step as *space carving*.
4. Perform an isosurface extraction at the zero-crossing of the signed distance function. Additionally, extract a surface between regions seen to be empty and regions that remain unseen.

In practice, we represent the unseen and empty states using the function and weight fields stored on the voxel lattice. We represent the unseen state with the function values $D(\mathbf{x}) = D_{max}$, $W(\mathbf{x}) = 0$ and the empty state with the function values $D(\mathbf{x}) = D_{min}$, $W(\mathbf{x}) = 0$, as shown in Figure 6b. The key advantage of this representation is that we can use the same isosurface extraction algorithm we used in the previous section without the restriction on interpolating voxels of zero weight. This extraction finds both the signed distance and hole fill isosurfaces and connects them naturally where they meet, i.e., at the corners in Figure 6a where the dotted red line meets the dashed green line. Note that the triangles that arise from interpolations across voxels of zero weight are distinct from the others: they are hole fillers.



(a) (b)

Figure 6. Volumetric grid with space carving and hole filling. (a) The regions in front of the surface are seen as empty, regions in the vicinity of the surface ramp through the zero-crossing, while regions behind remain unseen. The green (dashed) segments are the isosurfaces generated near the observed surface, while the red (dotted) segments are hole fillers, generated by tessellating over the transition from empty to unseen. In (b), we identify the three extremal voxel states with their corresponding function values.

We take advantage of this distinction when smoothing surfaces as described below.

Figure 6 illustrates the method for a single range image, and provides a diagram for the three-state classification scheme. The hole filler isosurfaces are "false" in that they are not representative of the observed surface, but they do derive from observed data. In particular, they correspond to a boundary that confines where the surface could plausibly exist. In practice, we find that many of these hole filler surfaces are generated in crevices that are hard for the sensor to reach.

Because the transition between unseen and empty is discontinuous and hole fill triangles are generated as an isosurface between these binary states, with no smooth transition, we generally observe aliasing artifacts in these areas. These artifacts can be eliminated by prefiltering the transition region before sampling on the voxel lattice using straightforward methods such as analytic filtering or super-sampling and averaging down. In practice, we have obtained satisfactory results by applying another technique: post-filtering the mesh after reconstruction using weighted averages of nearest vertex neighbors as described in [29]. The effect of this filtering step is to blur the hole fill surface. Since we know which triangles correspond to hole fillers, we need only concentrate the surface filtering on the these portions of the mesh. This localized filtering preserves the detail in the observed surface reconstruction. To achieve a smooth blend between filtered hole fill vertices and the neighboring "real" surface, we allow the filter weights to extend beyond and taper off into the vicinity of the hole fill boundaries.

We have just seen how "space carving" is a useful operation: it tells us much about the structure of free space, allowing us to fill holes in an intelligent way. However, our algorithm only carves back from observed surfaces. There are numerous situations where more carving would be useful. For example, the interior walls of a hollow cylinder may elude digitization, but by seeing through the hollow portion of the cylinder to a surface placed behind it, we can better approximate its geometry. We can extend the carving paradigm to cover these situations by placing such a backdrop behind the surfaces being scanned. By placing the backdrop outside of the voxel grid, we utilize it purely for carving space without introducing its geometry into the model.

5 Implementation

5.1 Hardware

The examples in this paper were acquired using a Cyberware 3030 MS laser stripe optical triangulation scanner. Figure 1b illustrates the scanning geometry: an object translates through a plane of laser light while the reflections are triangulated into depth profiles through a CCD camera positioned off axis. To improve the quality of the data, we apply the method of spacetime analysis as described in [6]. The benefits of this analysis include reduced range noise, greater immunity to reflectance changes, and less artifacts near range discontinuities.

When using traditional triangulation analysis implemented in hardware in our Cyberware scanner, the uncertainty in triangulation for our system follows the lines of sight of the expanding laser beam. When using the spacetime analysis, however, the uncertainty follows the lines of sight of the camera. The results described in section 6 of this paper were obtained with one or the other triangulation method. In each case, we adhere to the appropriate lines of sight when laying down signed distance and weight functions.

5.2 Software

The creation of detailed, complex models requires a large amount of input data to be merged into high resolution voxel grids. The examples in the next section include models generated from as many as 70 scans containing up to 12 million input vertices with volumetric grids ranging in size up to 160 million voxels. Clearly, time and space optimizations are critical for merging this data and managing these grids.

5.2.1 Run-length encoding

The core data structure is a run-length encoded (RLE) volume with three run types: empty, unseen, and varying. The varying fields are stored as a stream of varying data, rather than runs of constant value. Typical memory savings vary from 10:1 to 20:1. In fact, the space required to represent one of these voxel grids is usually *less* than the memory required to represent the final mesh as a list of vertices and triangle indices.

5.2.2 Fast volume traversal

Updating the volume from a range image may be likened to inverse volume rendering: instead of reading from a volume and writing to an image, we read from a range image and write to a volume. As a result, we leverage off of a successful idea from the volume rendering community: for best memory system performance, stream through the volume and the image simultaneously in scanline order [18]. In general, however, the scanlines of a range image are not aligned with the scanlines of the voxel grid, as shown in Figure 7a. By suitably resampling the range image, we obtain the desired alignment (Figure 7b). The resampling process consists of a depth rendering of the range surface using the viewing transformation specific to the lines of sight of the range sensor and using an image plane oriented to align with the voxel grid. We assign the weights as vertex "colors" to be linearly interpolated during the rendering step, an approach equivalent to Gouraud shading of triangle colors.

To merge the range data into the voxel grid, we stream through the voxel scanlines in order while stepping through the corresponding scanlines in the resampled range image. We map each voxel scanline to the correct portion of the range scanline as depicted in Figure 7d, and we resample the range data to yield a distance from the range surface. Using the combination rules given by equations 3 and 4, we update the run-length encoded structure. To preserve the linear memory structure of the RLE volume (and thus avoid using linked lists of runs scattered through the memory space), we read the voxel scanlines from the current volume and write the updated scanlines to a second RLE volume; i.e., we double-buffer the voxel grid. Note that depending on the scanner geometry, the mapping from voxels to range image pixels may not be linear, in which case care must be taken to resample appropriately [5].

For the case of merging range data only in the vicinity of the surface, we try to avoid processing voxels distant from the surface. To that end, we construct a binary tree of minimum and maximum depths for every adjacent pair of resampled range image scanlines. Before processing each voxel scanline, we query the binary tree to decide

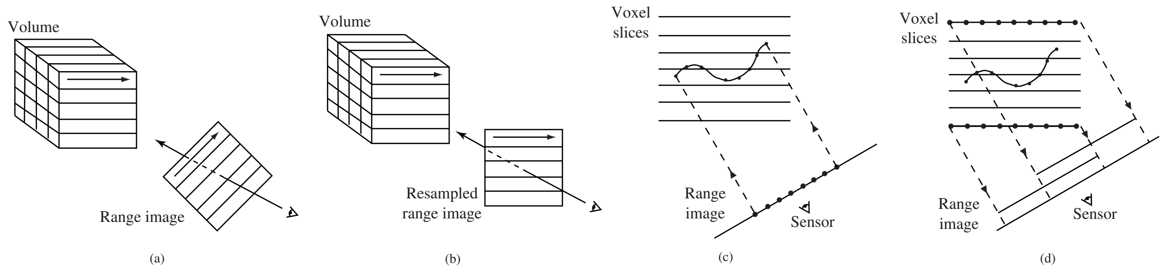


Figure 7. Range image resampling and scanline order voxel updates. (a) Range image scanlines are not in general oriented to allow for coherently streaming through voxel and range scanlines. (b) By resampling the range image, we can obtain the desired range scanline orientation. (c) Casting rays from the pixels on the range image means cutting across scanlines of the voxel grid, resulting in poor memory performance. (d) Instead, we run along scanlines of voxels, mapping them to the correct positions on the resampled range image.

which voxels, if any, are near the range surface. In this way, only relevant pieces of the scanline are processed. In a similar fashion, the space carving steps can be designed to avoid processing voxels that are not seen to be empty for a given range image. The resulting speedups from the binary tree are typically a factor of 15 without carving, and a factor of 5 with carving. We did not implement a brute-force volume update method, however we would expect the overall algorithm described here would be much faster by comparison.

5.2.3 Fast surface extraction

To generate our final surfaces, we employ a Marching Cubes algorithm [20] with a lookup table that resolves ambiguous cases [22]. To reduce computational costs, we only process voxels that have varying data or are at the boundary between empty and unseen.

6 Results

We show results for a number of objects designed to explore the robustness of our algorithm, its ability to fill gaps in the reconstruction, and its attainable level of detail. To explore robustness, we scanned a thin drill bit using the traditional method of optical triangulation. Due to the false edge extensions inherent in data from triangulation scanners [6], this particular object poses a formidable challenge, yet the volumetric method behaves robustly where the zippering method [30] fails catastrophically. The dragon sequence in Figure 11 demonstrates the effectiveness of carving space for hole filling. The use of a backdrop here is particularly effective in filling the gaps in the model. Note that we do not use the backdrop at all times, in part because the range images are much denser and more expensive to process, and also because the backdrop tends to obstruct the path of the object when automatically repositioning it with our motion control platform. Finally, the “Happy Buddha” sequence in Figure 12 shows that our method can be used to generate very detailed, hole-free models suitable for rendering and rapid manufacturing.

Statistics for the reconstruction of the dragon and Buddha models appear in Figure 8. With the optimizations described in the previous section, we were able to reconstruct the observed portions of the surfaces in under an hour on a 250 MHz MIPS R4400 processor. The space carving and hole filling algorithm is not completely optimized, but the execution times are still in the range of 3-5 hours, less than the time spent acquiring and registering the range images. For both models, the RMS distance between points in the original range images and points on the reconstructed surfaces is approximately 0.1 mm. This figure is roughly the same as the accuracy of the scanning technology, indicating a nearly optimal surface reconstruction.

7 Discussion and future work

We have described a new algorithm for volumetric integration of range images, leading to a surface reconstruction without holes. The algorithm has a number of desirable properties, including the representation of directional sensor uncertainty, incremental and order independent updating, robustness in the presence of sensor errors, and the ability to fill gaps in the reconstruction by carving space. Our use of a run-length encoded representation of the voxel grid and synchronized processing of voxel and resampled range image scanlines make the algorithm efficient. This in turn allows us to acquire and integrate a large number of range images. In particular, we demonstrate the ability to integrate up to 70 scans into a high resolution voxel grid to generate million polygon models in a few hours. These models are free of holes, making them suitable for surface fitting, rapid prototyping, and rendering.

There are a number of limitations that prevent us from generating models from an arbitrary object. Some of these limitations arise from the algorithm while others arise from the limitations of the scanning technology. Among the algorithmic limitations, our method has difficulty bridging sharp corners if no scan spans both surfaces meeting at the corner. This is less of a problem when applying our hole-filling algorithm, but we are also exploring methods that will work without hole filling. Thin surfaces are also problematic. As described in section 3, the influences of observed surfaces extend behind their estimated positions for each range image and can interfere with distance functions originating from scans of the opposite side of a thin surface. In this respect, the apexes of sharp corners also behave like thin surfaces. While we have limited this influence as much as possible, it still places a lower limit on the thickness of surface that we can reliably reconstruct without causing artifacts such as thickening of surfaces or rounding of sharp corners. We are currently working to lift this restriction by considering the estimated normals of surfaces.

Other limitations arise from the scanning technologies themselves. Optical methods such as the one we use in this paper can only provide data for external surfaces; internal cavities are not seen. Further, very complicated objects may require an enormous amount of scanning to cover the surface. Optical triangulation scanning has the additional problem that both the laser and the sensor must observe each point on the surface, further restricting the class of objects that can be scanned completely. The reflectance properties of objects are also a factor. Optical methods generally operate by casting light onto an object, but shiny surfaces can deflect this illumination, dark objects can absorb it, and bright surfaces can lead to interreflections. To minimize these effects, we often paint our objects with a flat, gray paint.

Straightforward extensions to our algorithm include improving the execution time of the space carving portion of the algorithm and demonstrating parallelization of the whole algorithm. In addition, more aggressive space carving may be possible by making inferences about sensor lines of sight that return no range data. In the future, we hope to apply our methods to other scanning technologies and to large scale objects such as terrain and architectural scenes.

Model	Scans	Input triangles	Voxel size (mm)	Volume dimensions	Exec. time (min)	Output triangles	Holes
Dragon	61	15 M	0.35	712x501x322	56	1.7 M	324
Dragon + fill	71	24 M	0.35	712x501x322	257	1.8 M	0
Buddha	48	5 M	0.25	407x957x407	47	2.4 M	670
Buddha + fill	58	9 M	0.25	407x957x407	197	2.6 M	0

Figure 8. Statistics for the reconstruction of the dragon and Buddha models, with and without space carving.

Acknowledgments

We would like to thank Phil Lacroute for his many helpful suggestions in designing the volumetric algorithms. Afra Zomorodian wrote the scripting interface for scanning automation. Homan Igehy wrote the fast scan conversion code, which we used for range image resampling. Thanks to Bill Lorensen for his marching cubes tables and mesh decimation software, and for getting the 3D hardcopy made. Matt Pharr did the accessibility shading used to render the color Buddha, and Pat Hanrahan and Julie Dorsey made helpful suggestions for RenderMan tricks and lighting models. Thanks also to David Addleman and George Dabrowski of Cyberware for their help and for the use of their scanner. This work was supported by the National Science Foundation under contract CCR-9157767 and Interval Research Corporation.

References

[1] C.L. Bajaj, F. Bernardini, and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3D scans. In *Proceedings of SIGGRAPH '95 (Los Angeles, CA, Aug. 6-11, 1995)*, pages 109–118. ACM Press, August 1995.

[2] J.-D. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics*, 3(4):266–286, October 1984.

[3] C.H. Chien, Y.B. Sim, and J.K. Aggarwal. Generation of volume/surface octree from range data. In *The Computer Society Conference on Computer Vision and Pattern Recognition*, pages 254–60, June 1988.

[4] C. I. Connolly. Cumulative generation of octree models from range data. In *Proceedings, Intl. Conf. Robotics*, pages 25–32, March 1984.

[5] B. Curless. *Better optical triangulation and volumetric reconstruction of complex models from range images*. PhD thesis, Stanford University, 1996.

[6] B. Curless and M. Levoy. Better optical triangulation through spacetime analysis. In *Proceedings of IEEE International Conference on Computer Vision*, pages 987–994, June 1995.

[7] A. Dolenc. Software tools for rapid prototyping technologies in manufacturing. *Acta Polytechnica Scandinavica: Mathematics and Computer Science Series*, Ma62:1–111, 1993.

[8] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach. Ridges for image analysis. *Journal of Mathematical Imaging and Vision*, 4(4):353–373, Dec 1994.

[9] H. Edelsbrunner and E.P. Mücke. Three-dimensional alpha shapes. In *Workshop on Volume Visualization*, pages 75–105, October 1992.

[10] A. Elfes and L. Matthies. Sensor integration for robot navigation: combining sonar and range data in a grid-based representation. In *Proceedings of the 26th IEEE Conference on Decision and Control*, pages 1802–1807, December 1987.

[11] H. Gagnon, M. Soucy, R. Bergevin, and D. Laurendeau. Registration of multiple range views for automatic 3-D model building. In *Proceedings 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 581–586, June 1994.

[12] E. Grosso, G. Sandini, and C. Frigato. Extraction of 3D information and volumetric uncertainty from multiple stereo images. In *Proceedings of the 8th European Conference on Artificial Intelligence*, pages 683–688, August 1988.

[13] P. Hebert, D. Laurendeau, and D. Poussart. Scene reconstruction and description: geometric primitive extraction from multiple viewed scattered data. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 286–292, June 1993.

[14] A. Hilton, A.J. Toddart, J. Illingworth, and T. Windeatt. Reliable surface reconstruction from multiple range images. In *Fourth European Conference on Com-*

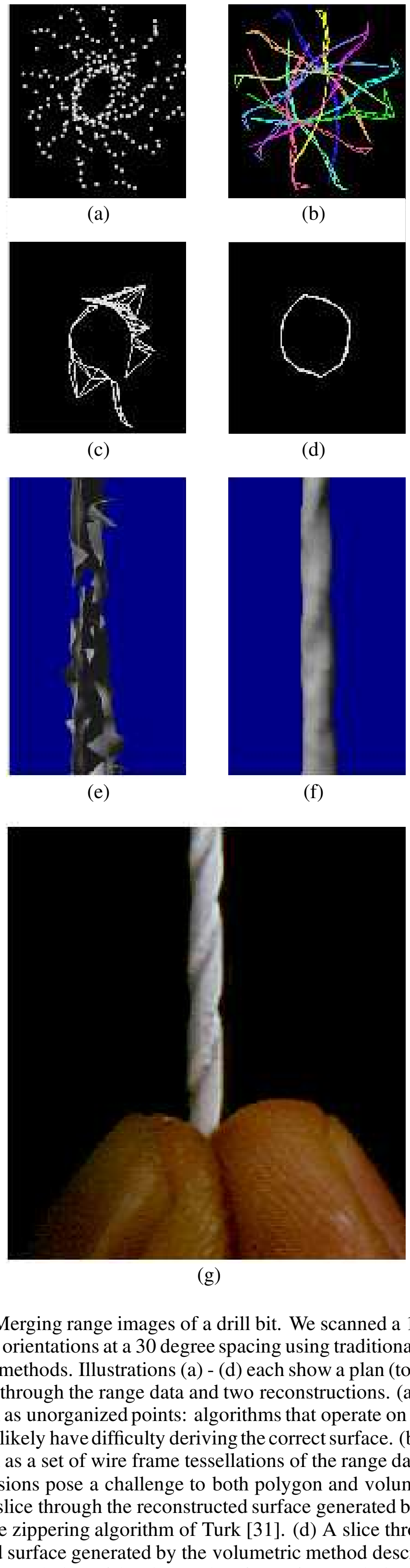


Figure 9. Merging range images of a drill bit. We scanned a 1.6 mm drill bit from 12 orientations at a 30 degree spacing using traditional optical triangulation methods. Illustrations (a) - (d) each show a plan (top) view of a slice taken through the range data and two reconstructions. (a) The range data shown as unorganized points: algorithms that operate on this form of data would likely have difficulty deriving the correct surface. (b) The range data shown as a set of wire frame tessellations of the range data: the false edge extensions pose a challenge to both polygon and volumetric methods. (c) A slice through the reconstructed surface generated by a polygon method: the zippering algorithm of Turk [31]. (d) A slice through the reconstructed surface generated by the volumetric method described in this paper. (e) A rendering of the zippered surface. (f) A rendering of the volumetrically generated surface. Note the catastrophic failure of the zippering algorithm. The volumetric method, however, produces a watertight model. (g) A photograph of the original drill bit. The drill bit was painted white for scanning.

puter Vision*, volume I, pages 117–126, April 1996.

[15] Tsai-Hong Hong and M. O. Shneier. Describing a robot's workspace using a sequence of views from a moving camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(6):721–726, November 1985.

[16] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 71–78, July 1992.

[17] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In these proceedings.

[18] P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proceedings of SIGGRAPH '94 (Orlando, FL, July 24-29, 1994)*, pages 451–458. ACM Press, July 1994.

[19] A. Li and G. Crebbin. Octree encoding of objects from range images. *Pattern Recognition*, 27(5):727–739, May 1994.

[20] W.E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 163–169, July 1987.

[21] W.N. Martin and J.K. Aggarwal. Volumetric descriptions of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):150–158, March 1983.

[22] C. Montani, R. Scateni, and R. Scopigno. A modified look-up table for implicit disambiguation of marching cubes. *Visual Computer*, 10(6):353–355, 1994.

[23] M. Potmesil. Generating octree models of 3D objects from their silhouettes in a sequence of images. *Computer Vision, Graphics, and Image Processing*, 40(1):1–29, October 1987.

[24] M. Rutishauser, M. Stricker, and M. Trobina. Merging range images of arbitrarily shaped objects. In *Proceedings 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 573–580, June 1994.

[25] M. Soucy and D. Laurendeau. A general surface approach to the integration of a set of range views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(4):344–358, April 1995.

[26] G. Succi, G. Sandini, E Grosso, and M. Tistarelli. 3D feature extraction from sequences of range data. In *Robotics Research. Fifth International Symposium*, pages 117–127, August 1990.

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

[28] G.H Tarbox and S.N. Gottschlich. IVIS: An integrated volumetric inspection system. In *Proceedings of the 1994 Second CAD-Based Vision Workshop*, pages 220–227, February 1994.

[29] G. Taubin. A signal processing approach to fair surface design. In *Proceedings of SIGGRAPH '95 (Los Angeles, CA, Aug. 6-11, 1995)*, pages 351–358. ACM Press, August 1995.

[30] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Proceedings of SIGGRAPH '94 (Orlando, FL, July 24-29, 1994)*, pages 311–318. ACM Press, July 1994.

[31] Robert Weinstock. *The Calculus of Variations, with Applications to Physics and Engineering*. Dover Publications, 1974.

A Isosurface as least squares minimizer

It is possible to show that the isosurface of the weighted signed distance function is equivalent to a least squares minimization of squared distances between points on the range surfaces and points on the desired reconstruction. The key assumptions are that the range sensor is orthographic and that the range errors are independently distributed along sensor lines of sight. A full proof is beyond the scope of this paper, but we provide a sketch here. See [5] for details.

Consider a region, R, on the desired surface, f, which is observed by n range images. We define the error between an observed range surface and a possible reconstructed surface as the integral of the weighted squared distances between points on the range surface and the reconstructed surface. These distances are taken along the lines of sight of the sensor, commensurate with the predominant directions of uncertainty (see Figure 10). The total error is the sum of the integrals for the n range images:

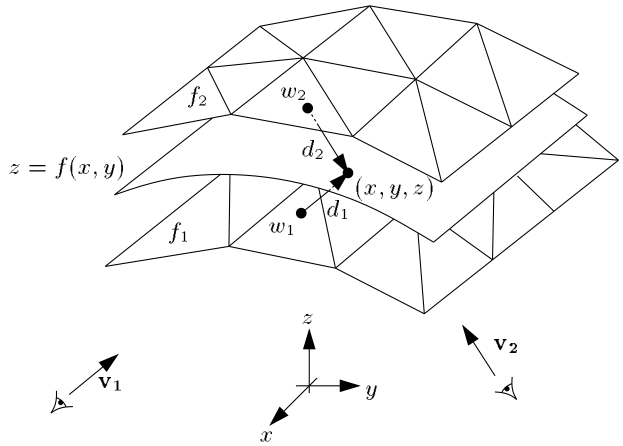


Figure 10. Two range surfaces, f_1 and f_2, are tessellated range images acquired from directions v_1 and v_2. The possible range surface, $z = f(x, y)$, is evaluated in terms of the weighted squared distances to points on the range surfaces taken along the lines of sight to the sensor. A point, (x, y, z), is shown here being evaluated to find its corresponding signed distances, d_1 and d_2, and weights, w_1 and w_2.

$$E(f) = \sum_{i=1}^{n} \iint_{A_i} w_i(s, t, f) d_i(s, t, f)^2 \, ds dt \tag{6}$$

where each (s, t) corresponds to a particular sensor line of sight for each range image, A_i is the domain of integration for the i'th range image, and $w_i(s, t, f)$ and $d_i(s, t, f)$ are the weights and signed distances taken along the i'th range image's lines of sight.

Now, consider a canonical domain, A, on a parameter plane, (x, y), over which R is a function $z = f(x, y)$. The total error can be rewritten as an integration over the canonical domain:

$$E(z) = \iint_A \sum_{i=1}^{n} \left[w_i(x, y, z) d_i(x, y, z)^2 \right] \left[\mathbf{v_i} \cdot \left(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}, -1 \right) \right] dx dy \tag{7}$$

where $\mathbf{v_i}$ is the sensing direction of the i'th range image, and the weights and distances are evaluated at each point, (x, y, z), by first mapping them to the lines of sight of the corresponding range image. The dot product represents a correction term that relates differential areas in A to differential areas in A_i. Applying the calculus of variations [31], we can construct a partial differential equation for the z that minimizes this integral. Solving this equation we arrive at the following relation:

$$\sum_{i=1}^{n} \partial_{\mathbf{v_i}} \left[w_i(x, y, z) d_i(x, y, z)^2 \right] = 0 \tag{8}$$

where $\partial_{\mathbf{v_i}}$ is the directional derivative along $\mathbf{v_i}$. Since the weight associated with a line of sight does not vary along that line of sight, and the signed distance has a derivative of unity along the line of sight, we can simplify this equation to:

$$\sum_{i=1}^{n} w_i(x, y, z) d_i(x, y, z) = 0 \tag{9}$$

This weighted sum of signed distances is the same as what we compute in equations 1 and 2, without the division by the sum of the weights. Since the this divisor is always positive, the isosurface we extract in section 3 is exactly the least squares minimizing surface described here.

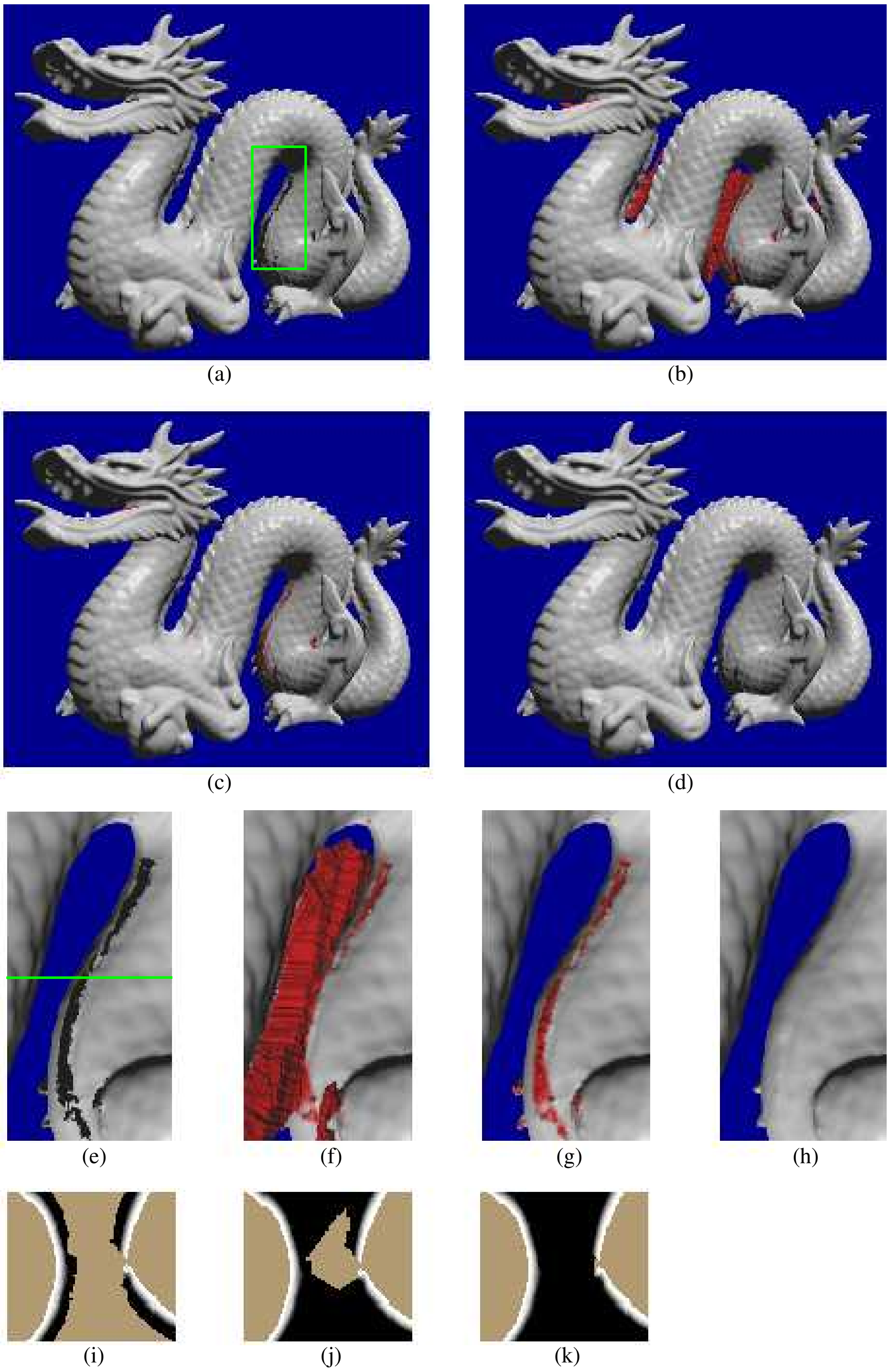


Figure 11. Reconstruction of a dragon. Illustrations (a) - (d) are full views of the dragon. Illustrations (e) - (h) are magnified views of the section highlighted by the green box in (a). Regions shown in red correspond to hole fill triangles. Illustrations (i) - (k) are slices through the corresponding volumetric grids at the level indicated by the green line in (e). (a)(e)(i) Reconstruction from 61 range images without space carving and hole filling. The magnified rendering highlights the holes in the belly. The slice through the volumetric grid shows how the signed distance ramps are maintained close to the surface. The gap in the ramps leads to a hole in the reconstruction. (b)(f)(j) Reconstruction with space carving and hole filling using the same data as in (a). While some holes are filled in a reasonable manner, some large regions of space are left untouched and create extraneous tessellations. The slice through the volumetric grid reveals that the isosurface between the unseen (brown) and empty (black) regions will be connected to the isosurface extracted from the distance ramps, making it part of the connected component of the dragon body and leaving us with a substantial number of false surfaces. (c)(g)(k) Reconstruction with 10 additional range images using "backdrop" surfaces to effect more carving. Notice how the extraneous hole fill triangles nearly vanish. The volumetric slice shows how we have managed to empty out the space near the belly. The bumpiness along the hole fill regions of the belly in (g) corresponds to aliasing artifacts from tessellating over the discontinuous transition between unseen and empty regions. (d)(h) Reconstruction as in (c)(g) with filtering of the hole fill portions of the mesh. The filtering operation blurs out the aliasing artifacts in the hole fill regions while preserving the detail in the rest of the model. Careful examination of (h) reveals a faint ridge in the vicinity of the smoothed hole fill. This ridge is actual geometry present in all of the renderings, (e)-(h). The final model contains 1.8 million polygons and is watertight.

(a) (b) (c) (d) (e)

Figure 12. Reconstruction and 3D hardcopy of the “Happy Buddha”. The original is a plastic and rosewood statuette that stands 20 cm tall. Note that the camera parameters for each of these images is different, creating a slightly different perspective in each case. (a) Photograph of the original after spray painting it matte gray to simplify scanning. (b) Gouraud-shaded rendering of one range image of the statuette. Scans were acquired using a Cyberware scanner, modified to permit spacetime triangulation [6]. This figure illustrates the limited and fragmentary nature of the information available from a single range image. (c) Gouraud-shaded rendering of the 2.4 million polygon mesh after merging 48 scans, but before hole-filling. Notice that the reconstructed mesh has at least as much detail as the single range image, but is less noisy; this is most apparent around the belly. The hole in the base of the model corresponds to regions that were not observed directly by the range sensor. (d) RenderMan rendering of an 800,000 polygon decimated version of the hole-filled and filtered mesh built from 58 scans. By placing a backdrop behind the model and taking 10 additional scans, we were able to see through the space between the base and the Buddha’s garments, allowing us to carve space and fill the holes in the base. (e) Photograph of a hardcopy of the 3D model, manufactured by 3D Systems, Inc., using stereolithography. The computer model was sliced into 500 layers, 150 microns apart, and the hardcopy was built up layer by layer by selectively hardening a liquid resin. The process took about 10 hours. Afterwards, the model was sanded and bead-blasted to remove the stair-step artifacts that arise during layered manufacturing.