Filling holes in complex surfaces using volumetric diffusion

Steve Marschner James Davis Matt Garr Marc Levoy

Stanford University

Abstract

We address the problem of building watertight 3D models from surfaces that contain holes—for example, sets of range scans that observe most but not all of a surface. We specifically address situations in which the holes are too geometrically and topologically complex to fill using triangulation algorithms. Our solution begins by constructing a signed distance function, the zero set of which defines the surface. Initially, this function is defined only in the vicinity of observed surfaces. We then apply a diffusion process to extend this function through the volume until its zero set bridges whatever holes may be present. If additional information is available, such as known-empty regions of space inferred from the lines of sight to a 3D scanner, it can be incorporated into the diffusion process. Our algorithm is simple to implement, is guaranteed to produce manifold non-interpenetrating surfaces, and is efficient to run on large datasets because computation is limited to areas near holes. By showing results for complex range scans, we demonstrate that our algorithm produces hole-free surfaces that are plausible, visually acceptable, and usually close to the intended geometry.

1 Introduction

Modern rangefinding systems can measure the shape of an object's surface with high accuracy and resolution. However, these systems often cannot observe the entire surface, so the resulting 3D models may be incomplete. The most fundamental cause of holes is occlusion—recesses may be too deep to be observed using a particular triangulation angle. However, holes can also be caused by low reflectance, constraints on scanner placement, or simply missing views.

In some applications, an incomplete surface model is appropriate—it represents the surface exactly as measured, without adding fabricated geometry. However, other applications require a watertight surface that bounds a volume of space. Examples include computations of physical properties, fabrication of physical replicas, or presentation in contexts like schools and museums where holes would be confusing and unattractive. Algorithms that are used in these applications often require that surfaces are valid 2-manifolds and/or that the geometry does not intersect itself.

To allow such uses while maintaining the data's accuracy and integrity, we need a surface reconstruction method that strictly preserves the geometry where it exists and smoothly transitions to valid, plausible geometry in unobserved areas. For scientific applications, it is also important to know which parts of the surface were observed and which parts are hypothetical.

One difficulty of hole filling is choosing appropriate topology. Many holes are simple and can be filled with disc topology; in these cases, triangulation algorithms can easily construct suitable patches. However, some holes have convoluted geometry, for which a naive triangulation algorithm may produce a self-intersecting surface. Other holes have multiple boundary components that should be filled, not with discs, but with patches that connect two or more loops of the boundary. This case occurs frequently when the data is intermittent (such as occurs with low object reflectance or grazing observation angles; see Figure 4c). A topologically inflexible approach may fail to find a valid manifold surface that passes through all available data in these areas. Moreover, decisions about hole topology are inherently global and must be made consistently.

To summarize, the ideal hole filling algorithm should:

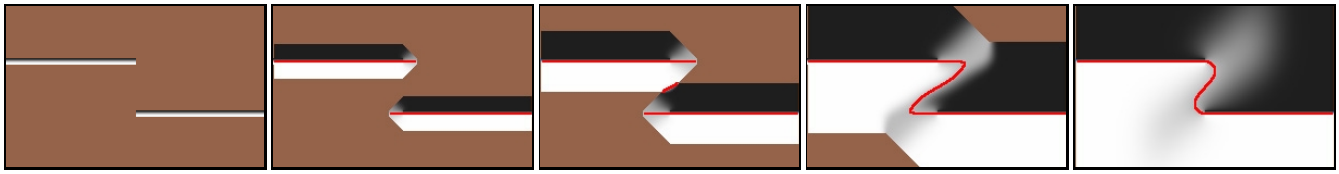
- produce manifold, non-self-intersecting surfaces for any input, no matter how noisy or convoluted,
- choose appropriate (possibly non-disc) topology for unusually shaped holes,
- construct plausible, visually pleasing geometry,
- distinguish in the output model between observed and fabricated surfaces,
- use all available information, including knowledge of the scanner's lines of sight and noise characteristics, and
- be efficient and scalable to large datasets, since scanned models can contain upwards of a billion samples [13].

We describe a new technique for filling holes in scanned models by processing a volumetric representation—the zero set of a signed distance function defined only near the observed surface. After converting the input data into a volume (we use VRIP, the volumetric surface reconstruction algorithm introduced by Curless and Levoy [7]), we apply diffusion in 3D to extend this incomplete surface description until it forms a watertight (hole-free) model. The result is always topologically consistent (i. e., manifold), cannot self-intersect, and maintains fidelity to the original data wherever it exists.

2 Prior work

Hole filling can be performed as a post-processing operation, applied after surface reconstruction, or it can be integrated into a surface reconstruction algorithm. One widely used approach for filling holes in an already reconstructed surface is to fill each connected component of the surface's boundary with a patch that has the topology of a disc. This technique works well for simple holes in nearly flat surfaces, but the 3D boundary of a hole can be convoluted, like a tangled loop of string. In general it is NP-complete to triangulate a closed boundary without self-intersections [3].

Scan-based methods of surface reconstruction [7, 17], which consider each scan, or range image, as a connected measurement of the surface, can use either separate or integrated hole filling. In either case, they begin by making the distinction between observed



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

Figure 1: Illustration of 2D diffusion in progress. (a) We begin with the source term; (b) the two surfaces are extended; (c) the surfaces begin to interact; (d) the hole closes; (e) the shape is converged. Brown denotes unknown areas ($v = 0$); grayscale values encode signed distance, with black and white corresponding to outside ($d = -1$) and inside ($d = 1$) respectively. The red curve marks the zero set.

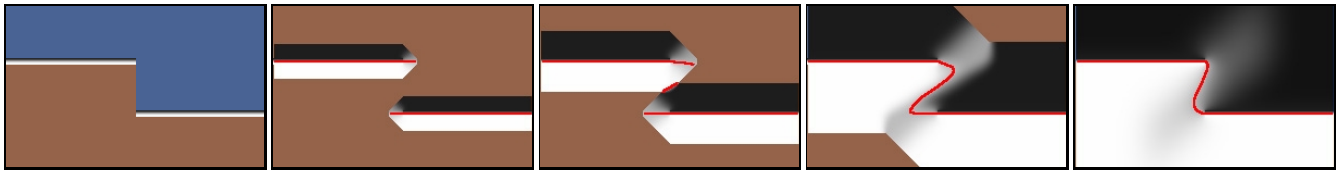


Figure 2: The same example as Figure 1, but with space carving. Incorporating the additional information into the diffusion process causes the converged surface to stay out of the region known to be empty. In (a) the blue marks where the space-carving source term indicates that space is empty ($w_c > 0$, visible here only when $w_s = 0$).

and unobserved surface. Specifically, they maintain the notion of connectivity between samples within each range image, and they explicitly avoid connecting samples that, due to missing data or a sudden change in depth, are probably not adjacent on the surface.

The only scan-based method we know of that integrates hole filling into surface reconstruction is Curless and Levoy's VRIP [7]. Their method marks the region of 3D space that lies between the scanner and observed surfaces and is therefore known to be empty, then extracts a surface that includes the boundary of this region in addition to the observed surface. This so-called space carving method creates a surface that bounds the maximum region of space consistent with the scans, an approach that is guaranteed to produce a watertight surface. However, this method requires knowledge of scanner lines of sight, and it performs poorly if these lines of sight do not adequately cover the volume outside the object. Our algorithm, by contrast, will use space carving information if it is available, but it can operate without it. Also, VRIP's implicit assumption that all unseen space is inside the object may lead to surfaces that are less plausible than would result from methods that try to smoothly extend the observed surfaces.

Point-cloud methods [1, 2, 4, 8, 10, 18], on the other hand, which treat the union of all the scans as an unorganized set of 3D points, generally fill holes during reconstruction. With no notion of connectivity between range samples, the large gap across a hole is conceptually equivalent to the space between adjacent samples.

Some point-cloud methods that interpolate the original samples, such as those based on alpha shapes [8, 2] or crusts [1], can bridge holes. However, these algorithms are slow, and they may fail if sample noise approaches sample density—which it often does. In algorithms based on alpha shapes [8, 2], it may be difficult to find a single alpha (or probe sphere radius) that bridges holes without also bridging fine surface details.

Another class of point-cloud methods works by evolving a closed "shrink-wrap" or "inflating balloon" surface until it interpolates or approximates the data. Some of these methods are *level-set* methods [6, 18], which resemble our method in their use of a 3D signed distance function to represent the surface. However, there are important differences. In order to use these methods to fill holes they must be applied to the entire surface, even though the holes typically involve only a small fraction of the total area—perhaps only a few percent. As a result, these level-set methods are slow to run on large models. Our diffusion process, on the other hand, operates only near holes; we do not attempt to maintain a distance function throughout the volume. Also, since our algorithm operates after surface reconstruction is complete, it is compatible with any reconstruction method. Finally, our algorithm provide a mechanism for including additional constraints, such as regions of space known to be empty.

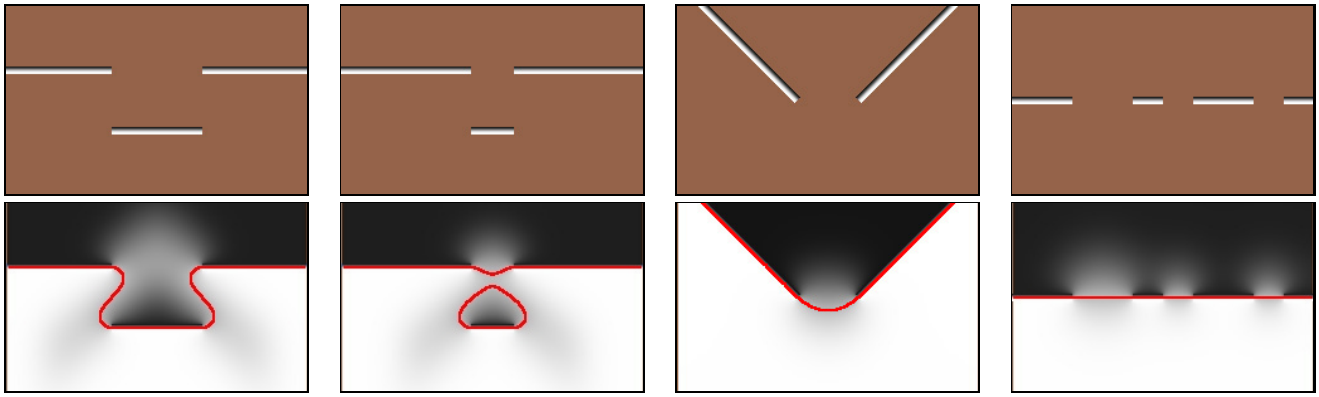
A different application of diffusion to filling gaps in sampled data is the image reconstruction work of Bertalmio et al. [5]. Like our method, they iterate a combination of simple operators to propagate information from known regions of the image into the unknown (e. g., scratched) regions. However, their method propagates sharp linear features, whereas ours propagates the zero set of a smooth function. Also, their method is designed to keep adjacent parallel propagating fronts from colliding, whereas ours is designed to find and connect such fronts.

3 Volumetric diffusion

Our hole filling algorithm can be used on any 3D surface model. The surface is first converted to our volumetric representation, which is a regularly spaced 3D grid of values of a clamped signed distance function $d_s(\mathbf{x})$. This function is defined only near the observed surface, and it is positive inside the surface and negative outside, with its values limited to $[-1, 1]$. The observed surface is the zero set of d_s.[1] One can construct d_s in many ways; our implementation uses the VRIP algorithm [7] to build this function directly from a collection of range scans. One could alternatively build d_s using volumetric scan conversion from an already reconstructed surface [9]. At the same time we define an associated weight function w_s, which ranges from 0 to 1 and measures our confidence in the value of d_s. In most areas $w_s = 1$, but it typically decreases near boundaries of the observed surface, where noise increases.

The goal of our algorithm is to extend d_s to a function d that is defined over the entire volume, though in practice we only compute d near the surface—in fact only near holes in the surface. We achieve this by diffusing the values of d_s outward from the observed surface into adjoining undefined areas. As the function spreads, so

[1] An alternative definition, equivalent in practice, is a filtered sidedness function: -1 outside the object's surface and $+1$ inside.



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

Figure 3: Examples of diffusion in 2D. Above: source term; below: diffusion result with zero set marked in red. (a) A double step discontinuity that is filled as two separate steps; (b) a narrower double step that results in the topological decision to build a bridge across the pit. (c) The common case in which the bottom of a depression is missed; (d) a flat surface with several holes.

does its zero set. In particular, the diffused function propagates inward across the holes, eventually spanning the holes. Once diffusion is complete, the zero set of this function is the desired hole-free surface.

The diffusion we propose is similar, although not identical, to classic solutions for the heat equation [12]. In heat diffusion, a scalar field representing temperature is propagated from each node in a computational domain to its neighbors according to the material's thermal conductivity. Optionally, after each diffusion step, a source term is added into each node, representing the addition of heat to the system. In the terminology of image processing, these propagation and adding steps have been shown to be equivalent to spatial convolution and image compositing, respectively [11].

The diffusion process consists of alternating steps of blurring and compositing. We begin with $d = d_s$, and each iteration first convolves d with a lowpass filter h, then composites d_s back into the volume using the *over* operator [15]. The algorithm uses two volumes, the diffusion volume and the source volume. The diffusion volume, which is where the computation takes place, has two values at each point, $d_i(\mathbf{x}) \in [-1, 1]$, the value of d after i iterations, and $v_i(\mathbf{x}) \in \{0, 1\}$, which indicates where the value of d_i is known. The source volume represents the observed surface, and contains two values, $d_s(\mathbf{x}) \in [-1, 1]$ and $w_s(\mathbf{x}) \in [0, 1]$. The initialization is:

$$(d_0, v_0) \quad = \quad (d_s, [w_s > 0]),$$

where $[p] = 1$ if p is true and 0 otherwise. A single iteration is:

$$\begin{aligned} (\hat{d}_i, v_i) &= h \star (d_{i-1}, v_{i-1}) \\ d_i &= w_s d_s + (1 - w_s)\hat{d}_i. \end{aligned}$$

During the convolution only valid voxels are used; h is renormalized to include only these voxels. The convolution also extends v to include all voxels where a value was computed. This process is continued for as many iterations as necessary to achieve a suitable surface. Due to the repeated blurring, the choice of the filter h is not critical; our implementation uses a $3 \times 3 \times 3$ box or a 7-point "plus" filter. The output surface is extracted by running Marching Cubes [14] once, after diffusion is complete, to extract the $d = 0$ isosurface. Figure 1 shows the stages of diffusion for a simple test case.

This volumetric diffusion algorithm, implemented naively, would consume time and memory proportional to n^3, the number of voxels in the volume, per iteration. Because most of the volume is empty, and only a small fraction of the surface contains holes, this is inefficient for large models. We take two measures to accelerate the computation: we use a sparse representation of the volume that avoids using memory for undefined areas, and we limit the computation to voxels that are not more than a certain predetermined distance from the holes in the original surface.

To implement the first optimization, we represent both the source volume and the working volume using a simple block structure. The volume is divided into fixed-sized cubical blocks, and storage is only allocated for those blocks where the values are changing; otherwise only a single value is stored. Since the time and space spent on the block occupancy table is negligible, this representation in practice requires memory proportional to the volume of space that is being used to store the surface and the diffusion result.

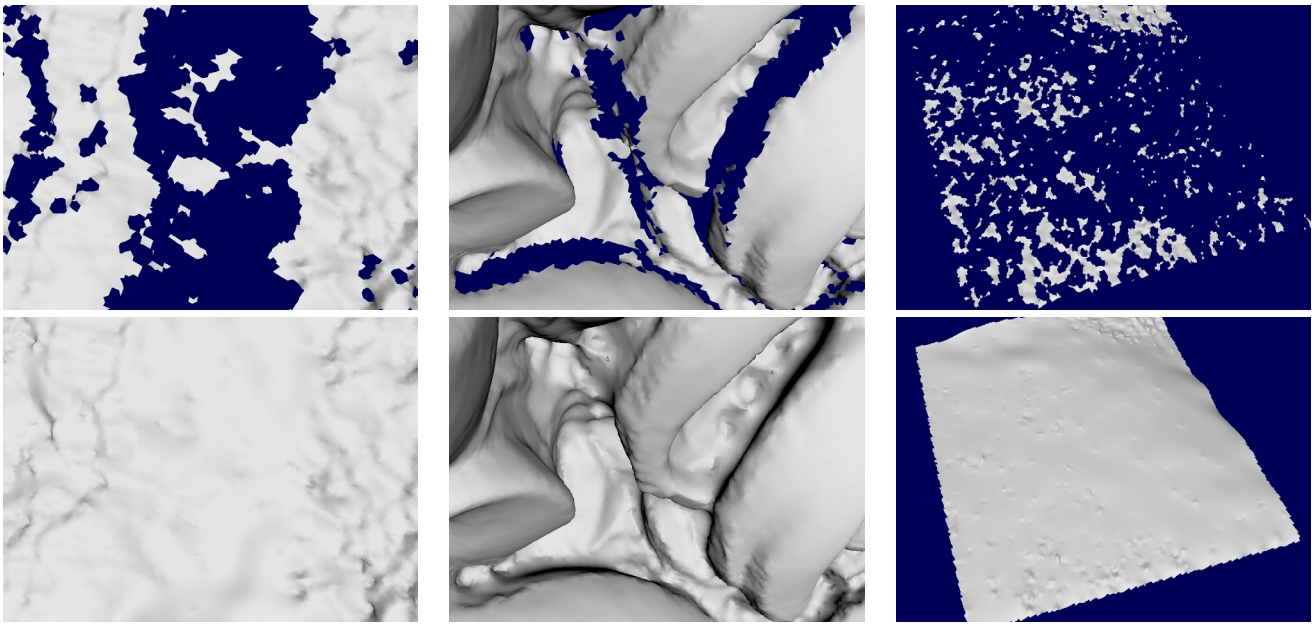
To implement the second optimization, we simply flag the voxels that are within m voxels from an edge in the source term[2] and process only those voxels during diffusion. The choice of m depends on the size of the largest hole that must be filled; m must be greater than half the width of that hole.[3] Typical values of m for the examples shown in this paper are 15 to 30 voxels.

With these two optimizations, the algorithm requires space proportional to the surface area and time that depends on m and the area and size of holes. If we let k be the fraction of the surface area that is within distance m of a boundary, the processing time for a diffusion iteration is proportional to kn^2m. The value of k reflects both the size and the shape of the holes, but is typically small (a few percent).

Range scanners provide information about where surfaces are not as well as where they are [7], based on the known emptiness of space between the range scanner and the observed surface. When such *space carving* information is available, it may be incorporated into the diffusion process as a second source term, (d_c, w_c). For volumes of space that are known to be empty, we set $d_c = -1$, indicating that those areas are outside the surface, and we set $w_c = \alpha$. The parameter α can be thought of as our confidence that these voxels are indeed empty. In practice, it controls how far into the empty region surfaces will be built; for high α the surface will turn sharply to avoid empty regions, and for lower values it will remain smoother

[2] A voxel is an edge voxel if it is valid, has at least one invalid neighbor (in v_0), and has at least one valid neighbor with the opposite sign (in d_0).

[3] The required value of m can be made lower by extending the computation region dynamically to include voxels within distance m of the frontier of the evolving $d = 0$ isosurface. However, we do not currently do this.



(a) (b) (c)

Figure 4: Results of our algorithm on holes that occur in practice. (a) A closeup (30 $\times$ 40 mm) of a fragment of the *Forma Urbis Romae*, a marble map of ancient Rome, shows a hole that has complex topology, though simple geometry. The surface is relatively flat, but the hole has many islands, which would be thrown away by algorithms based on triangulating the boundary. (b) A detail (75 $\times$ 50 mm) from the head of Michelangelo's *David* shows holes in a surface with complex geometry. Our diffusion process creates plausible surfaces to fill the holes. (c) A piece (40 $\times$ 50 mm) of Michelangelo's *Night*, a highly polished marble statue. The statue's high specularity and the grazing scanning angle in this example created an area that is mostly hole, rather than mostly surface, but the same computation fills it in as well.

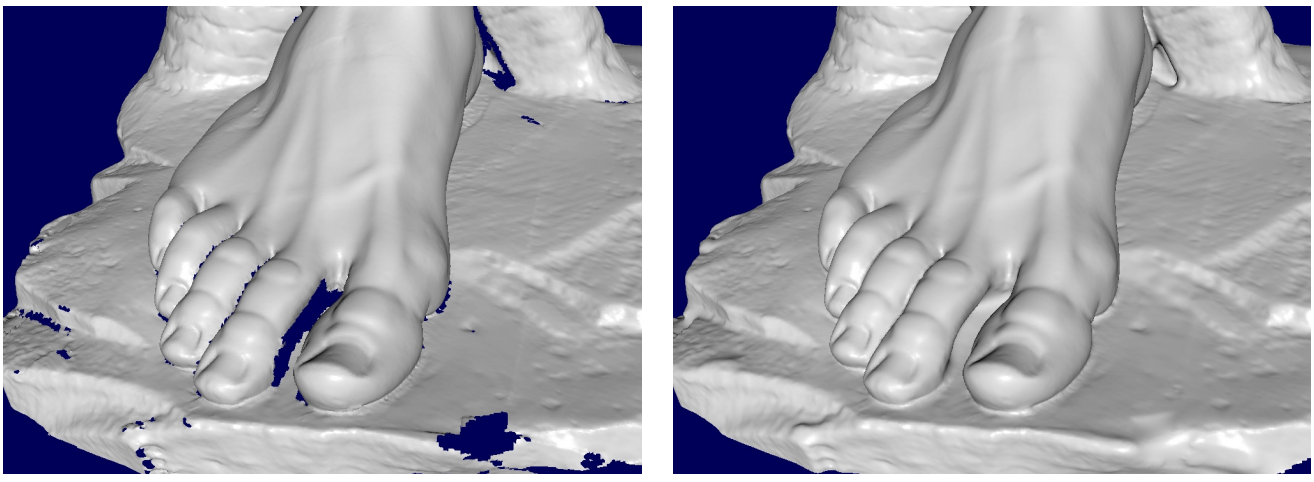


Figure 5: Volumetric diffusion applied to a larger dataset. The volume contained 440 million voxels, and the output triangle mesh contained 4.5 million triangles. Processing time for diffusion on a 1 GHz Pentium III PC was 20 minutes; the maximum memory allocated was 550 MB. The number of voxels touched during the diffusion was 4.5% of the total; the number of blocks allocated was 11.5% of the total.

at the expense of protruding slightly into the empty region. Typical values for α range from 0.001 to 0.01. At the boundaries between empty and unseen volumes of space, w_c ramps to zero over a length dependent on the precision of the scanner. When using space carving in the diffusion, we still initialize d_0 to d_s, but during each iteration we composite with (d_c, w_c), then (d_s, w_s). This has the effect of applying a constant, gentle pressure toward -1 (outside) as the function diffuses into the known-empty region of space. An example of space carving in 2D is shown in Figure 2.

4 Results

We have applied our algorithm using the clamped distance function generated by VRIP for d_s, with w_s derived from VRIP's confidence values. These confidences account for several sources of uncertainty, most notably surface orientation relative to the scanner's line of sight and distance to the nearest edge of the observed surface.

To illustrate the behavior of our algorithm, we show results in Figure 3 for several synthetic 2D test cases that resemble different types of holes. These images show that the algorithm generates plausible, smooth surfaces for a variety of configurations, and that it can generate different topologies.

Figure 4 shows results for three scanned models from the Digital Michelangelo Project [13]. These examples illustrate the algorithm's ability to fill a variety of very different holes that arise in practice using the same computation.

To demonstrate the scalability of the system, Figure 5 shows results for the entire foot of the *David* with 1 mm voxel spacing.

The value of space-carving information is demonstrated in Figure 6. The example shows the first two toes of the *David*'s right foot. In this case, the diffusion process made a topological choice to build a bridge between the toes. While the bridge is consistent with the shape of this large hole, it is not the correct topology. (Note that Figure 5 did not use space carving; the bridge occurs only for very specific parameter settings, and it did not happen to form in that particular run.) The space carving information, represented by blue in the slices of the volumetric representation, provides additional knowledge that prevents this bridge from ever forming.

5 Conclusions

We have presented a new technique for filling holes in range scans by using diffusion to complete a volumetric representation of the surface. The method is simple and effective, and it always produces a closed, manifold triangle mesh without self-intersections. We have demonstrated its feasibility on real data of significant size.

One limitation of our present implementation is that, since our source term comes from VRIP, the scanner's lines of sight define the boundary between $w_s = 0$ and $w_s > 0$. Generally this boundary is approximately perpendicular to the scanned surface, but when all available scans were taken obliquely it may be angled. Since the zero set of our diffusion process tends to propagate perpendicularly to the boundary of w_s, this can cause undesired ripples in the constructed surface. The correct solution would be to create a clamped signed distance function (or a filtered sidedness function) directly from the reconstructed surface [9]. We are currently addressing this problem.

Diffusion processes, and the closely related level set methods, fit into a very general computational framework [16], so our algorithm can be extended in many ways. In particular, although we have incorporated into our algorithm several important forms of ancillary information about our input data, there are other constraints we might wish to add:

- Our algorithm leads to smooth surfaces that generally blend well with the observed surface. However, greater control over the

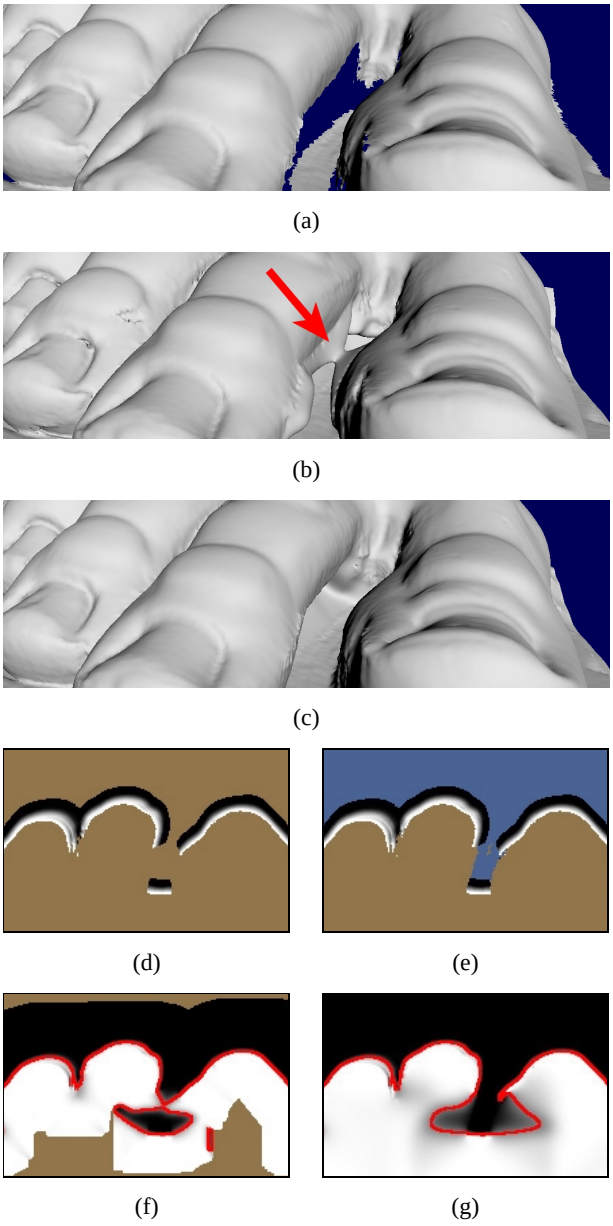


Figure 6: An example of using space carving information in the diffusion process. (a) Between the toes of Michelangelo's *David*, occlusion causes a large hole with ambiguous topology. (b) Filling the hole without space carving can result in an inappropriate bridge; (c) space carving constrains the choice to the correct topology. (d-e) Slices of the source volume without space carving (d) and with space carving (e). (f-g) The same slices of the diffused volume, showing the bridge that forms in (f) without space carving but is avoided in (g). Further diffusion will reduce the protrusion in (g) near where the bridge formed in (f).

properties of the generated surface—for example, its curvature or surface area—might be desirable. This can be achieved by incorporating terms related to the surface properties into the diffusion process [16, 18].

- Although space carving can often resolve ambiguous topology, there will always be cases that require high-level knowledge to disambiguate. We suggest that this knowledge is best provided through user intervention. The user could manipulate d_s directly [9], or indirectly by marking points in the volume as outside or inside.

References

[1] Nina Amenta, Marshall Bern, and Manolis Kamvysselis. A new voronoi-based surface reconstruction algorithm. *Proceedings of SIGGRAPH 1998*, July 1998.

[2] Chandrajit L. Bajaj, Fausto Bernardini, and Guoliang Xu. Automatic reconstruction of surfaces and scalar fields from 3D scans. *Proceedings of SIGGRAPH 95*, August 1995.

[3] G. Barequet, M. Dickerson, and D. Eppstein. On triangulating three-dimensional polygons. *Computational Geometry: Theory and Applications*, 10(3), June 1998.

[4] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Claudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, October-December 1999.

[5] Marcelo Bertalmio, Guillermo Sapiro, Vicent Caselles, and Coloma Ballester. Image inpainting. *Proceedings of SIGGRAPH 2000*, July 2000.

[6] Y. Chen and G. Medioni. Description of complex objects from multiple range images using an inflating ballon model. *Computer Vision and Image Understanding*, 61(3), May 1995.

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

[8] Herbert Edelsbrunner and Ernst P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1), 1994.

[9] Sarah Frisken, Ronald Perry, Alyn Rockwood, and Thouis Jones. Adaptively sampled distance fields: A general representation of shape for computer graphics. *Proceedings of SIGGRAPH 2000*, July 2000.

[10] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2), 1992.

[11] J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

[12] L.D. Landau and E.M. Lipshitz. *Fluid Mechanics, 2nd edition*. Butterworth Heinemann Oxford, 1987.

[13] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The Digital Michelangelo Project: 3D scanning of large statues. *Proceedings of SIGGRAPH 2000*, July 2000.

[14] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics (Proceedings of SIGGRAPH 87)*, 21(4), July 1987.

[15] Thomas Porter and Tom Duff. Compositing digital images. *Computer Graphics (Proceedings of SIGGRAPH 84)*, 18(3), July 1984.

[16] James Albert Sethian. *Level set methods: evolving interfaces in geometry, fluid mechanics, computer vision, and materials science*. Cambridge University Press, 1996.

[17] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. *Proceedings of SIGGRAPH 94*, July 1994.

[18] Ross Whitaker. A level-set approach to 3D reconstruction from range data. *International Journal of Computer Vision*, 29(3), October 1998.