Online Reconstruction of 3D Objects from Arbitrary Cross-Sections

AMIT BERMANO, AMIR VAXMAN, and CRAIG GOTSMAN
Technion – Israel Institute of Technology

We describe a simple algorithm to reconstruct the surface of smooth three-dimensional multilabeled objects from sampled planar cross-sections of arbitrary orientation. The algorithm has the unique ability to handle cross-sections in which regions are classified as being inside the object, outside the object, or unknown. This is achieved by constructing a scalar function on R^3, whose zero set is the desired surface. The function is constructed independently inside every cell of the arrangement of the cross-section planes using transfinite interpolation techniques based on barycentric coordinates. These guarantee that the function is smooth, and its zero set interpolates the cross-sections. The algorithm is highly parallelizable and may be implemented as an incremental update as each new cross-section is introduced. This leads to an efficient online version, performed on a GPU, which is suitable for interactive medical applications.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism

General Terms: Algorithms, Design, Measurement

ACM Reference Format:

Bermano, A., Vaxman, A., and Gotsman, C. 2011. Online reconstruction of 3D objects from arbitrary cross-sections ACM Trans. Graph. 30, 5, Article 113 (October 2011), 14 pages.
DOI = 10.1145/2019627.2019632
http://doi.acm.org/10.1145/2019627.2019632

1. INTRODUCTION

The reconstruction of a surface from a set of planar cross-sections S (also called *slices*), such that the surface interpolates, or approximates, S in the planes, has been thoroughly studied in the past decades. This problem arises mainly in the fields of medical imaging (MRI, CT, ultrasound, etc.) and geographical information systems (for terrain reconstruction). The input is assumed to have been segmented in a preprocessing step, to create a set of closed two-dimensional contours, separating the "inside" and "outside" of the object on each slice. A reconstruction algorithm typically creates a surface, enclosing a solid volume, which interpolates these contours and is consistent with the given inside/outside information. See Figure 1 for an illustration of the scenario.

The prior art related to this problem typically assumes that entire cross-sections are available and that the contour information in them is complete (i.e., a set of closed contours) and segmented correctly. However, in most practical cases, the input images are noisy, so there might be regions of uncertainty (see Figure 1) which cannot be reliably segmented. Thus, instead of a complete inside/outside classification per cross-section, there will also be regions which are unclassified. We deal with this general scenario.

The algorithm we describe constructs a signed "distance function," whose zero set is the surface. It consists of two steps: completing the regions of unknown information on the cross-section planes, and then using these to define a function on R^3, from which it extracts a zero set. Our construction is based on the definition of near-binary functions on the individual planes and their interpolation onto R^3 using transfinite interpolation methods based on barycentric coordinates.

Our algorithm solves the most general version of the problem, with every setting of cross-sections and contours, as well as the multilabeled contour problem (i.e., the contours within each plane may belong to separate components of the object, and are labeled accordingly). The resulting surface, obtained as the zero set of a smooth function, is naturally smooth. To the best of our knowledge, the only method that attempts to solve this problem at this level of generality is that of Barequet and Vaxman [2009], which, as we will see later, takes an entirely different, more complicated approach. Our algorithm handles all cases of noise and completion of cross-sections in a simple and uniform manner, without extra complexity due to irregular topologies. The resulting surfaces are smooth and well-behaved, requiring minimal postprocessing.

In addition, we deal with the online variant of the problem, in which the slices are not all available at reconstruction time, rather given individually over time, and the reconstruction is to be updated after receiving the new slice. This online reconstruction problem is typical of freehand ultrasound scanning applications, where the physician scans a patient and sees the anatomy build up as s/he scans more and more. Our algorithm easily adapts to this scenario, as we explain in Section 7.

2. PREVIOUS WORK

The first generation of work on the topic of reconstruction from contours (e.g., Batnitzky et al. [1981], Fix and Ladner [1998], Fuchs et al. [1977], Ganapathy and Dennehy [1982], Kehtarnavaz and Figueiredo [1988], Kehtarnavaz et al. [1988], Keppel [1975], Sloan and Painter [1988], Wang and Aggarwal [1986], and Welzl and

Authors' addresses: A. Bermano, A. Vaxman (corresponding author), and C. Gotsman, Computer Science Department, Technion – Israel Institute of Technology, Israel; email: avaxman@cs.technion.ac.il.

DOI 10.1145/2019627.2019632
http://doi.acm.org/10.1145/2019627.2019632

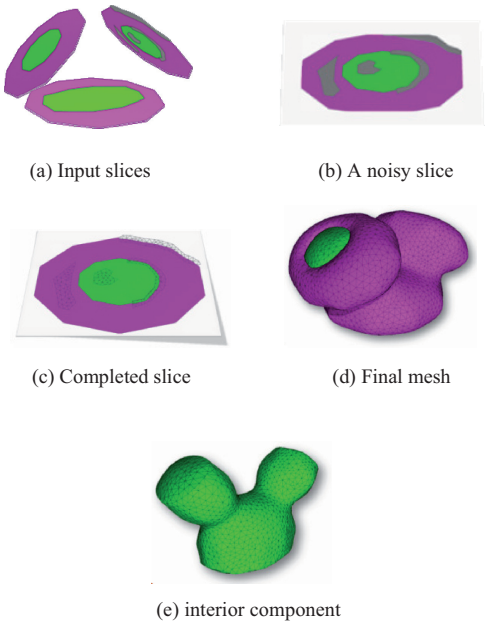


Fig. 1. Steps of our algorithm. The input (a) consists of three slices of an object having two connected components (materials), colored in purple and green. The top right slice (b) contains unknown regions (colored in gray). (c) The unknown regions in this slice are triangulated and completed. (d) The values of the indicator function in space are computed using barycentric interpolation, and the zero sets extracted. (e) The zero set corresponding to the green component.

Wolfers [1994]) studied the problem of interpolation between *parallel* cross-sections of a single object. Some (e.g., Choi and Park [1994], Christiansen and Sederberg [1978], Ekoule et al. [1991], Meyers et al. [1991], Shantz [1981], and Zyda et al. [1987]) even restrict the problem to scenarios where only a single contour is present in every input slice. Most early algorithms fail on complex inputs, such as multiple branching, or frequently create unacceptable self-intersecting solutions.

More recent works [Bajaj et al. 1996; Barequet and Sharir 1996; Boissonnat 1988; Barequet et al. 2004; Boissonnat and Geiger 1992; Cheng and Dey 1999; Felkel and Obdržálek 1999; Ju et al. 2005b; Oliva et al. 1996] address the problem of reconstruction of a single object from parallel slices without imposing any constraints on the contours, their geometries, or their containment hierarchies. They all focus on creating a portion of the surface between two adjacent slices, and concatenating the boundary surface from the set of created layers. Only a few (e.g., Bogush et al. [2004], Boissonnat and Memari [2007], Dance and Prager [1997], Liu et al. [2008], and Payne and Toga [1994]) have attempted to solve the problem for non-parallel, possibly intersecting slices.

The approach of most cross-section reconstruction algorithms is geometric, in which a correspondence is constructed between the vertices and edges of the polygonal input contours, forming triangulated surfaces, usually by projecting contours onto each other and solving the correspondence problem in two dimensions. The two most recent methods [Barequet and Vaxman 2009; Liu et al. 2008] form the arrangement of cross-section planes, and project the contours from the faces of each cell onto the faces of the straight skeleton or medial axis of the cell. They then erect "walls" between the original contours and the projections to create portions of the boundary surface. Albeit solving the problem in full generality, these methods create jagged surfaces with bad triangulations, which must then be improved using smoothing and/or remeshing in a postprocess. These geometric methods have the advantage of being fast and efficient for most reasonable inputs. However, both the medial axis and the straight skeleton have nonrational coordinates even for arrangements with rational coordinates, thus these methods are not numerically robust. This is further aggravated by the fact that the algorithms have zero error tolerance, as small errors in the computation of the skeleton might lead to self-intersecting surfaces, or surfaces with holes.

A completely different family of algorithms, to which our algorithm belongs, involves the use of distance functions (e.g., Turk and O'Brien [1999], and Csebfalvi et al. [2002]), commonly used in point cloud reconstruction algorithms. These scalar functions on R^3 are typically defined so that their zero set contains the given planar contours, and their absolute value increases with distance from the contour. However, these methods have been applied usually to the case of parallel, or at least ordered, slices (in which the ordering of the slices affects the solution). One might apply a traditional Radial-Basis Function (RBF) solution to our problem, interpolating the vertices (or some sampling) of the contours, but the sparsity of the data will typically result in the zero set having many spurious components.

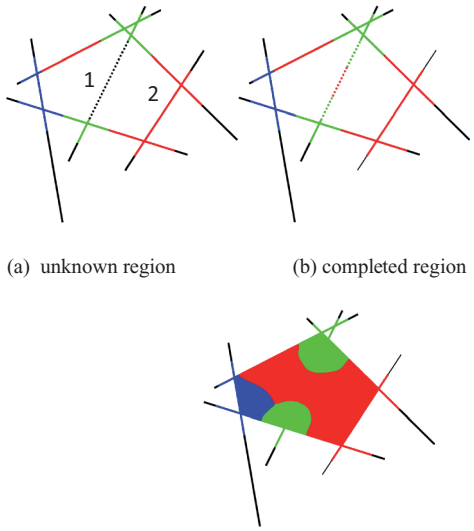
In order to deal with partial information, Barequet and Vaxman [2009] remove the parts of the plane which are unknown, compute the arrangement of partial planes and its three-dimensional straight-skeleton, and provide an extension to the algorithm of Liu et al. [2008], dealing with nonconvex cells in order to reconstruct the surface. This technique also allows the user to insert patches of planes inside cells, to better control the result. The complexity of the straight skeleton and the arrangement then grow at least cubically with the complexity of the unknown regions (see [Barequet et al. 2008]), which makes this algorithm prohibitive for complex noise patterns.

Finally, a few recent works [Ju et al. 2005b, Liu et al. 2008, Barequet and Vaxman 2009, Bajaj and Edwards 2010] addressed the more general problem of reconstruction from multilabeled contours (i.e., the contours within each plane may belong to separate components of the object). The main challenge here is to simultaneously reconstruct several nonintersecting surfaces. This is important in medical applications, where acquired images have been segmented to multiple organs. Our algorithm deals with multilabeled contours in a natural way.

3. CONTRIBUTIONS

We describe an intuitive and robust algorithm that applies the concept of barycentric transfinite interpolation to reconstruction from slices in a novel way. We solve the problem at the highest level of generality with a simple extension of our basic algorithm (see Section 4). Unlike state-of-the-art works, this extension enables treating inputs containing unknown regions with minimal effect on runtime and complexity. Furthermore, we provide a solution to the online reconstruction problem, which has not been treated before. The algorithm lends itself easily to a GPGPU implementation, which leads to interactive speeds.

From a theoretical point of view, we present a new closed-form solution to an important special case of mean value coordinates and an algorithm to construct a three-dimensional arrangement of planes that is both easy to implement and incremental.



(a) unknown region (b) completed region

(c) interpolated indicator function

Fig. 2. A two-dimensional illustration of our algorithm. (a) Two closed arrangement cells sharing a common face containing an unknown region (dashed gray). (b) The unknown region is completed using our completion algorithm (see Section 4). (c) The multilabeled indicator function is extracted and the output surface is defined to be the boundaries between the colored regions.

4. THE BASIC ALGORITHM

4.1 Overview

The input to our algorithm is a set of planar cross-sections (along with the orientation parameters of the planes). Each such plane is partitioned into labeled polygonal regions, with exactly one region having infinite area. The regions are labeled as follows.

(1) *Outside* (indicating the null material).
(2) The *ID* of a material (indicating that this region is in the interior of a specific three-dimensional object).
(3) *Unknown* (indicating that there is no reliable information as to whether the region is inside or outside an object).

In a nutshell, the algorithm proceeds as follows (see Figure 1).

(1) Assign a binary *indicator function* value to the vertices and half-edges of contours within each slice, indicating the type of regions they bound (inside or outside).
(2) Compute the arrangement of the slices within a global bounding box. The result is a set of convex cells. If there are regions labeled "unknown" within a slice, extrapolate the known indicator functions to these regions using barycentric coordinates.
(3) Apply three-dimensional barycentric transfinite interpolation to the indicator functions of the faces of an arrangement cell to obtain an indicator function value for every point inside the cell.
(4) Extract the zero set of the indicator function in the bounding box as a triangulated surface.

Figure 2 illustrates a two-dimensional version of our algorithm.

We begin by describing our method for the simplest special case of reconstruction of a single-material object, when there are no unknown regions on any of the planes. In Section 5 we describe the extension of this algorithm to cross-sections containing unknown regions, and in Section 6 its generalization to multilabeled contours. The output is a triangulated manifold mesh which represents the boundary of the object whose planar cross-sections agree with the input. Following Barequet and Vaxman [2009], we call the cross-section planes "slices," and the regions bounded by the contours on each slice "images."

4.2 Computing the Planar Arrangement

The main data structure supporting our algorithm is a three-dimensional arrangement of the planar slices. These are frequently used in algorithms performing reconstructions from nonparallel slices [Boissonnat and Memari 2007; Barequet and Vaxman 2009; Liu et al. 2008]. Constructing and maintaining an arrangement of planes, consisting of convex faces and cells, has been treated extensively in the literature. We create the three-dimensional arrangement in a manner which is slightly different from the conventional one, by first computing the intersection of all pairs of planes, and then creating a two-dimensional arrangement on every plane, consisting of the intersection lines of all other planes with it. It costs $O(k^2)$ time to create each two-dimensional arrangement, where k is the number of planes. The three-dimensional arrangement cells are then easily formed by matching corresponding edges from every two (or more) intersecting planes at an intersection line, thus the algorithm consumes $O(k^3)$ time in total, which is also optimal. This particular construction has the advantage that it allows for a natural extension to an online version, that is, where the three-dimensional arrangement, thus also the indicator function, is updated incrementally as each new slice is introduced. The complexity of introducing each new slice is the zone complexity of the new plane, enabling speedy runtimes of two seconds per slice or less, as seen in Table II. We discuss online reconstruction in more detail in Section 7.

4.3 Computing the Indicator Function

For the moment assume the simplest scenario, that there are no regions labeled "unknown." Suppose an arrangement face P contains a set of images of a material region Ω bounded by the contours forming $\partial\Omega$. We define an indicator function f, in the spirit of Kazhdan et al. [2006], as follows.

$$f(x) = \begin{cases} 1, & x \in \Omega \backslash \partial\Omega \\ 0, & x \in \partial\Omega \\ -1, & x \notin \Omega \end{cases} \tag{1}$$

This discrete-valued function indicates the regions of P which are outside an image, on the bounding contours, or inside an image. We next seek to interpolate this function into the cell using mean-value transfinite interpolation, which will generate a continuum of values. Notice that the function will not be continuous on the boundaries of the image.

In preparation for the interpolation, we slightly modify the theoretical definition (1) of f. We treat every contour as two back-to-back contours, one facing the material, and the other facing outside. Thus the edges of each contour are duplicated (as two half-edges), and the vertices of each contour are decoupled (as two *corners*, see Figure 4). We assign the value 1 to the corners of the inward-facing contour, and -1 to those of the outward-facing contour. By definition, f is not monovalent; however, we will see that the transfinite interpolation converges to the value 0 at the vertices and edges incident on two opposite regions. Throughout this article, we use the terms half-edges and corners, since we use two different kinds

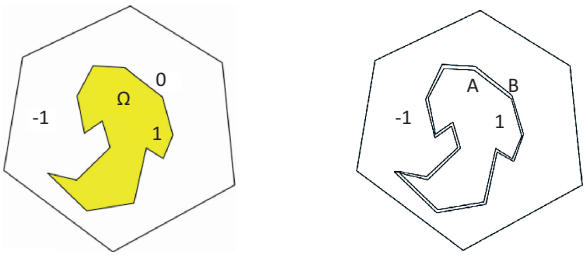


Fig. 3. (left) The theoretical indicator function on a cell face is implemented (right) using two back-to-back contours: contour A encloses the "1" region, and contours B and C enclose the "−1" region.

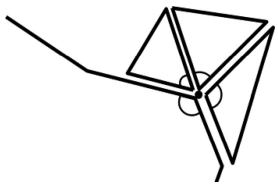


Fig. 4. Four corners of the same vertex. The vertex consists of corners of three unknown-region triangles and one corner of a general contour.

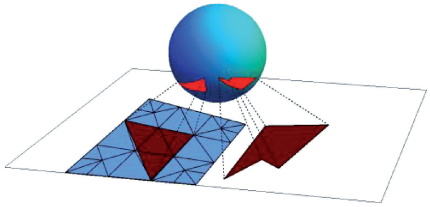


Fig. 5. Projection of planar regions unto the unit sphere.

of mean-value interpolation weights, and because we might assign different function values to different corners of the same vertex. Figure 3 illustrates the theoretical versus implemented indicator function.

4.4 The Function inside the Cells

Having defined the indicator function on all faces of the cells of the arrangement, we proceed to interpolate the indicator function values inside each cell using barycentric transfinite interpolation. Namely, given a cell C with function f defined on ∂C, the value of f_C at an interior point $x \in C$ is defined as the so-called "mean-value" boundary integral

$$f_C(x) = \frac{\iint_{\partial C} \frac{f(y)}{\|x-y\|} d\sigma(y)}{\iint_{\partial C} \frac{1}{\|x-y\|} d\sigma(y)}, \tag{2}$$

where $d\sigma(y)$ is the area of the boundary element at y projected onto the unit sphere centered at x (see Figure 5).

For general C and f, this integral might not have a closed-form solution. There are, however, a few special cases where a closed-form solution is available. Floater et al. [2005] treated the case of C being a convex polyhedron with f given at the vertices of the polyhedron, and assuming boundary values given by planar mean-value interpolation on the faces. This is not the same scenario as ours, where the function is already given on the faces of C. Ju et al. [2005a] treated the case that C is a triangulated non-convex polyhedron with f given at the vertices of the polyhedron, and piecewise-linear on its edges and faces. In this case, (2) is equivalent to assigning a *mean-value* weight function w_v:$\mathrm{R}^3 \rightarrow \mathrm{R}$ to every vertex v of the triangulation T of ∂C. The function f_C at any three-dimensional point x in ∂C is then just the weighted combination of the values of f at the vertices of the triangulation T.

$$f_C(x) = \sum_{v \in T} w_v(x) f(v) \tag{3}$$

The weight functions w_v (x) are described in Appendix A.

However, to apply (3) as is, each face of ∂C must be triangulated, which is wasteful in our scenario, since f is piecewise-constant on the boundary of a face, as all triangles not incident on the boundaries of the images have constant values. Towards this end, we have developed a solution to the mean-value integral (2) for boundaries having a piecewise-constant value of f. The contribution of a material image (possibly with holes) L on ∂C to the integral (2) is

$$f_{C,L}(x) = \frac{\iint_{L} \frac{1}{\|x-y\|} d\sigma(y)}{\iint_{\partial C} \frac{1}{\|x-y\|} d\sigma(y)}. \tag{4}$$

Because f is piecewise-constant, we are able to apply Stokes' theorem and compute a simple closed-form solution to this integral. This can be translated into weights for the contours ∂L, and, furthermore, weights for the corners of the contours ∂L, which we describe in Appendix A. We apply this to the regions with known inside/outside labels, and use the method of Ju et al. [2005a] for treating triangulations of the regions labeled "unknown," as we describe in Section 4. This results in

$$f(x) = \sum_{l \in L} w_l(x) - \sum_{l \in C \setminus L} w_l(x), \tag{5}$$

where $w_l(x)$ is the weight of contour l at point x.

We note that mean-value coordinates have constant precision, hence, when a cell is completely included in the object, the function is a constant "1", and vice versa for exclusion. Thus, no unnatural gaps are created in the surface.

4.5 Extracting the Zero Set

Once we have defined the three-dimensional indicator function f in each cell, and consequently in the entire global bounding box, we use a surface mesh generator algorithm, based on the three-dimensional Delaunay refinement algorithm of Rineau and Yvinec [2007], to create the triangulated zero set of f. This algorithm is based on the existence of an oracle that is able to tell if a linear object (segment, line, or ray) intersects the surface of the generated object. Since we are interested only in the zero set of f, we establish this oracle in two steps: first, we partition the bounding box using an octree, guided by the values of f, subdividing a cube essentially only if it contains zeros of f. Next, we perform the intersection queries on the octree, using a standard three-dimensional-DDA [Amanatides and Woo 1987], traversing the nodes of the octree until the leaves, and then locating the zero by assuming trilinear interpolation of the values at the cube corners. Note that it is easy to adapt the three-dimensional-DDA algorithm to nonuniform sized cells, due to the simple hierarchical tree structure: the intersection ray is always aware of the coarsest level for which it moves from a cell to its neighbor (on even coarser levels it stays in the same cell), and thus can easily deduce the finest cell in which it resides after every such move. See Figure 6 for an example of the octree refinement process.

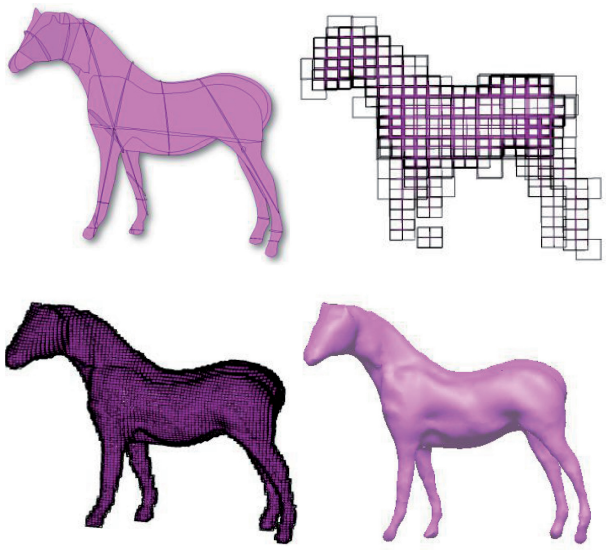


Fig. 6. Extracting the zero set. (Top left) Input slices of a horse. The indicator function was computed on an octree (coarsest level: top right, finest level: bottom left), which closely follows the final shape (bottom right).

The octree construction refines a node as long as one of the following conditions is satisfied, and a maximum depth (empirically set to 9 in all of our examples) has not been exceeded:

(1) The node contains more than one input contour vertex.
(2) The function in the node (as measured on the corners of the node) changes sign, having a maximal difference larger than a user-defined tolerance (which is typically the desired resolution of the output mesh).

We try to avoid situations in which the octree might undersample high curvature regions of the zero set. We do this by terminating subdivision of a cell one step later than the termination criteria would dictate. In practice, however, undersampling rarely if ever occurs; the barycentric blends smooth the indicator function considerably, and thus, the highest curvature values remain on the input slices, where they can be sampled as precisely as needed. In the case of a multilabeled input, the second refinement criterion of the octree is modified to having a node which contains more than one dominant label (see Section 6).

4.6 Postprocess Smoothing

The output surface is well-meshed and smooth; however, since the indicator function is a discontinuous step function on the slice near the zero set, the gradient of the function near the step tends strongly in the direction of the step, causing the zero set to become orthogonal to the slice at the contours. This side-effect sometimes leads to ripples in the surface. To alleviate this problem, we apply a few iterations of simple bi-Laplacian smoothing [Kobbelt et al. 1998] to the implicit function, on the octree, before extraction of the zero set. In order to maintain the interpolation property, we preserve the original function values in the nodes surrounding the original contours.

When treating multilabeled inputs, we simply apply smoothing to each function independently. Smoothing the implicit function before meshing avoids both the intricate treatment of nonmanifold smoothing, and the need to handle the tangential drift of the mesh in order to avoid the loss of mesh uniformity. See Figure 7 for an example of this smoothing.

Fig. 7. (left) Slices of a figure eight; (middle) reconstructed surface with some ripple effects; (right) smoothed version.

5. TREATING UNKNOWN REGIONS

We now proceed to the more general case where some regions within the slices are labeled "unknown." This means that we cannot reliably classify this region as belonging to any specific object. We seek to complete the information in these regions as a smooth extension of the information in the neighboring informative regions. We achieve that by extending our mean-value transfinite interpolation to groups of cells sharing faces having "unknown" images.

The indicator function of the known regions is defined as before. We triangulate the interior of each unknown region in order to support different function values at the corners of the triangulation. Given a single three-dimensional cell C, an interior point $x \in C$, a set of known regions $L \in \partial C$, and a triangulation T of the unknown regions, Eq. (5) generalizes to.

$$f(x) = \sum_{l \in L} w_l(x) - \sum_{l \in C \setminus (L \cup T)} w_l(x) + \sum_{t \in T} \sum_{i=1}^{3} f(v_{t,i}) w_{v_{t,i}}(x), \quad (6)$$

where $v_{t,i}$ is a *corner*, the i'th vertex of triangle t. The weights of the corners are those defined by Ju et al. [2005a], and we seek the function values $f(v_{t,i})$.

We proceed by defining cell groups. Consider a graph whose vertices are the arrangement cells with an edge connecting two cells sharing a face containing "unknown" regions. A cell group is a connected component of this graph. Note that these cell groups are equivalent to the (nonconvex) cells of the partial arrangement constructed in Barequet and Vaxman [2009], but we do not construct them explicitly. For every corner of the triangulation on a face s, we compute a blend of two quantities: the three-dimensional mean-value interpolant of all other faces of the cells in the cell group besides s, and the two-dimensional mean-value interpolant of the unknown region in the input slice that s originated from, where the boundary values of the unknown region are the function values of the contours bounding this region. This results in the following

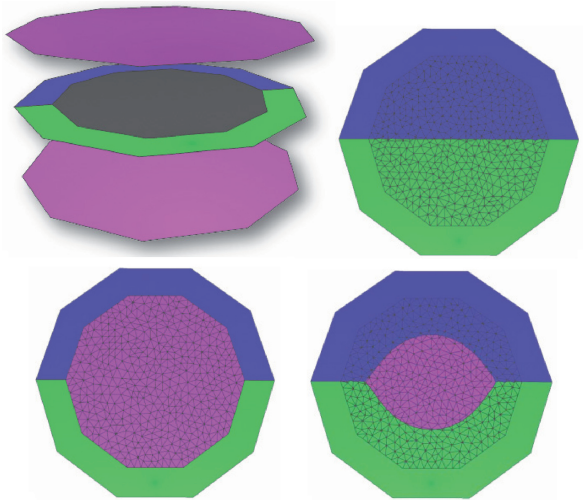


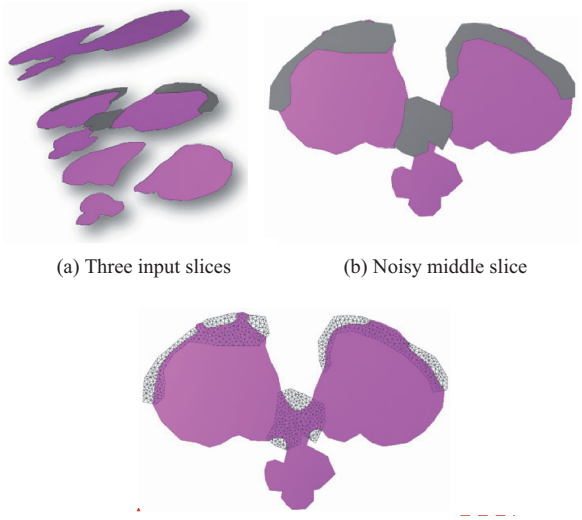
Fig. 8. Completion of values on a slice (middle slice in top left image). Green and blue values may be obtained from the neighboring two-dimensional data on the slice itself (top right) while purple values may be obtained from the surrounding three-dimensional data in neighboring slices (bottom left). The final result is a blend of both (bottom right).

completion formula:

$$f(v_{t,i}) = \left(\overbrace{\sum_{l \in G} w_l(v_{t,i}) - \sum_{l \in G/(L \cup T)} w_l(v_{t,i})}^{3D} + \overbrace{\sum_{e \in U} f(e) w_e(v_{t,i})}^{2D} \right), \qquad (7)$$

where G is the cell group, U is the unknown region containing $v_{t,i}$, and e is any edge of U whose value $f(e)$ is as the value of the contour bounding U sharing this edge. Normalization preserves the constant precision. Thus, when all the faces in a cell group are unicolored, we obtain the same color in the interior of the group, and as a completion value for the slices. Figure 7 illustrates this. Figure 9 depicts another case of completion in a slice of a lung.

There is some similarity between our interpolation of all faces of a cell group, and the use of the straight skeleton of nonconvex cells in Barequet and Vaxman [2009]. In fact, as Amini et al. [2010] point out, the algorithm of Liu et al. [2008] can actually be thought of as an interpolation algorithm as well, where an interior point assumes the value of the closest point on its cell boundary. It is not quite true that Barequet and Vaxman [2009] is also a degenerate-feature closest-point interpolation algorithm in the same spirit of Liu et al. [2008], since this would imply that the algorithm would use the medial axis instead of the straight skeleton, but it is a close approximation. Using the closest point as an indicator produces a reasonable zero-set topology in many cases (albeit the surface is jagged, having only C^0 continuity), but our method of blending values from all regions in the vicinity is much more robust to small changes in the input. Also, the discontinuous functions Liu et al. [2008] and Barequet and Vaxman [2009] induce are unsuitable for the mesh generator, as they contain low-frequency noise that is not easily suppressed (see rightmost artery in Figure 13). Figure 10 shows an example where the closest point interpolation used in the method of Barequet and Vaxman [2009] is sensitive to noise, and Figure 11 depicts a similar problematic case, where the completion



(a) Three input slices (b) Noisy middle slice

(c) completed middle slice

Fig. 9. Completing noisy slices: (a) Three input slices. (b) The middle slice contains noisy "unknown" regions. (c) The unknown region of the noisy slice is triangulated and completed. Completion values in the triangles are color-coded according to the dominant value (inside or outside).

of the unknown region is dominated by a relatively minor label present in the original slices.

Having completed the function values at the corners of the triangulations of the unknown regions, we may proceed with the algorithm of Section 4, combining our mean-value interpolation method with the one described by Ju et al. [2005a] for the completed triangles.

5.1 Function Values at Vertices

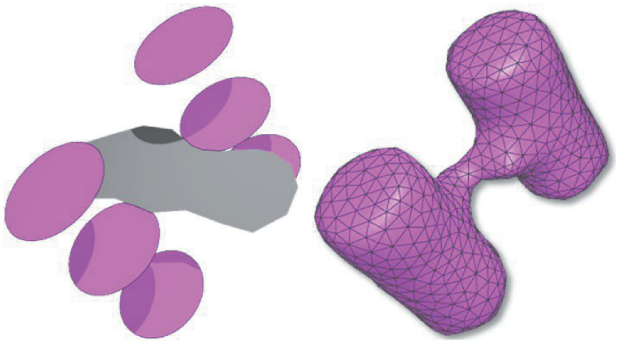
Throughout the algorithm, we establish function values at the corners of the images. The corners of contours bounding known regions have values which are either 1 or -1, but corners of the triangulation of unknown regions that coincide with known corners may have different function values. To reconcile them to a single function value per vertex, the function value at a vertex is implicitly averaged by the limit of the transfinite interpolation at the vertex, as a weighted sum of the corner values. Suppose that the weight of a corner y of vertex v is w_y, and that the function value at that corner is f_y, then the limit function value at the vertex becomes (see Figure 4)

$$f(v) = \frac{\sum_y w_y f_y}{\sum_y w_y}.$$

For example, two opposite corners at a contour separating the inside and outside regions have identical weights and function values of 1 and -1, respectively. This will result in a function value of 0, as expected from the indicator function.

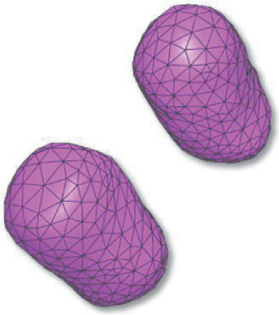
6. TREATING MULTILABELED REGIONS

Recent reconstruction algorithms [Barequet and Vaxman 2009; Liu et al. 2008] deal with inputs which consist of several segmented regions bounded by contours. The output is a so-called "surface network," which segments the three-dimensional space into closed



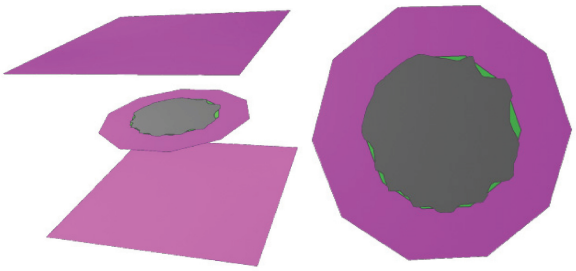
(a) input slices: three pairs of circles with a noisy region in the middle

(b) Barequet and Vaxman [2009] completion.



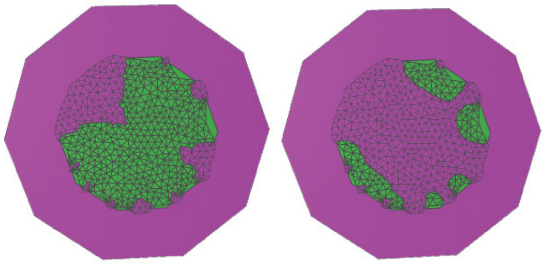
(c) our method

Fig. 10. The noisy region in the middle slice (a) leads to a large topological error (b), when applying a closest-point completion method on the sampled slices of two cylinders, even though the noise barely touches the material part of the slice. Our method (c) is almost unaffected by this noise.



(a) Input slices

(b) Middle slice



(c) [Barequet and Vaxman 2009]

(d) Our method

Fig. 11. Completion method of Barequet and Vaxman [2009] vs. our method. Notice that a few green spots near the boundary of an unknown region dominate the completed region, whereas our method blends the entire environment, including the two surrounding planes, better. This might lead to topological noise in the reconstruction of Barequet and Vaxman [2009].

volumes (plus the unique exterior region). The surfaces bounding these volumes are expected to be smooth everywhere except perhaps at the nonmanifold curves which are the junctions between three different volumes.

Our algorithm generalizes naturally to such inputs, ultimately assigning a unique label to any point in space. To achieve this, we compute a separate indicator function for each label, constructed as described in Sections 4 and 5. As usual, an indicator function for a single label is positive inside the labeled region, negative outside, and limits to zero on the image boundary. If there are unknown regions, they are completed independently for each label as described in Section 5. During the interpolation step, given a three-dimensional point x, as before, we compute a single spherical projection of all regions relative to x, compute the weights of the corners, and compute the function value for each label. The point x is then labeled according to the label having the largest function value. Thus, the algorithm requires just one spherical projection per point inside a cell, which is the most computationally expensive step, independent of the total number of labels. The multilabeled three-dimensional indicator function is a maximal-value function of smooth unilabeled mean-value interpolants. As such, it is smooth everywhere except at the medial regions, where two or more labels achieve the same maximal value. However, the output surface between exactly two materials is smooth as well, since it is the intersection of two smooth three-dimensional functions. Discontinuities in the surface may therefore appear only on the junction curves that are the intersection of at least three materials, which is as expected.

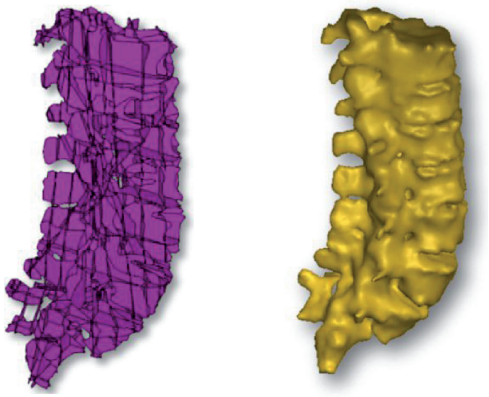
7. ONLINE RECONSTRUCTION

In a realistic medical application, such as freehand ultrasound, a patient may be scanned by a physician who observes the reconstruction of the scanned organ improving as the scanning proceeds. The application might even highlight regions which have not been scanned sufficiently, and the physician could focus effort in that region, improving the result there. Thus it is important that our reconstruction algorithm support this scenario: an incremental reconstruction scheme following the introduction of individual slices.

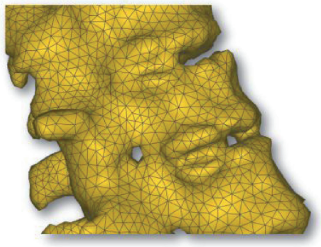
Our indicator function at each cell is independent of the other cells. Thus, a new slice in an existing arrangement affects only the cells contained in its zone, which has complexity $O(k^2)$. The octree nodes which intersect those cells become obsolete, and are updated accordingly. Having an updated octree, we can generate a new mesh. An example of online reconstruction can be seen in Figures 17 and 18, and in the accompanying video accessible through the ACM Digital Library.

8. IMPLEMENTATION DETAILS

Our implementation relies on the CGAL library [cgal] for most stages of our algorithm. Planar two-dimensional arrangements are constructed using the two-dimensional arrangement package in CGAL. This two-dimensional arrangement produces all the vertices, edges, and faces that each slice induces in the three-dimensional arrangement.



(a) input slices (b) our method



(c) closeup on a single vertebra

Fig. 12. Reconstructed vertebrae.

The zero set is extracted using the surface mesh generation package (for unicolored inputs), and the segmented mesh generation package (for multilabeled inputs), both instantiated with our octree oracle.

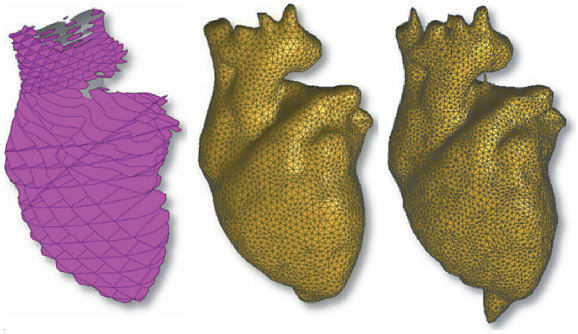
8.1 GPU Optimizations

The dominant component of our algorithm is the evaluations of the indicator function required to construct the octree. Usually, even for reasonable inputs, at least 100,000 evaluations are required to construct an octree deep enough to capture all features of the indicator function. A conventional CPU implementation would therefore be rather slow.

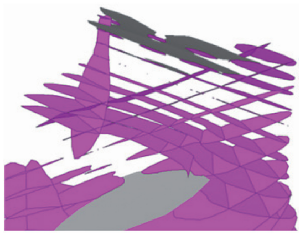
We have devised a GPGPU implementation, using CUDA, to speed up the evaluation process. As the octree is totally parallelizable in any single level, we build it in a breadth-first manner. The evaluations needed to construct each level of the octree are done in a single GPU pass. This results in octree update time of less than 3 seconds per slice (see Table II).

The mesh generator and the surface mesh generator are serial algorithms; however, our octree simplifies and optimizes the queries dramatically, therefore reducing the mesh generation time to a minimum.

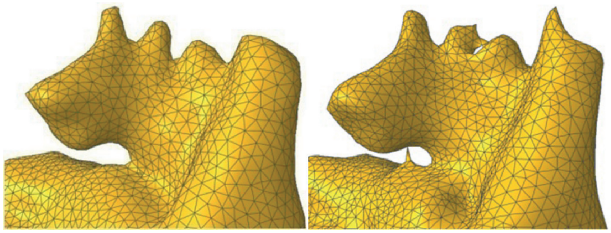
When a new slice is introduced into an existing arrangement, we update the two-dimensional arrangements of all other slices by adding the line of intersection the new slice shares with them, and create the two-dimensional arrangement of the new slice. As the complexity of the zone of this slice is $O(k^2)$, so is the time complexity of this step. The octree is updated using the GPU in a manner similar to its construction: We traverse it level by level, and update cells which are obsolete. Then, the mesh generation process is redone using the new octree.



(a) input slices (b) our method (c) Barequet and Vaxman [2009]



(d) closeup on the noisy part



(e) our method (f) Barequet and Vaxman [2009]

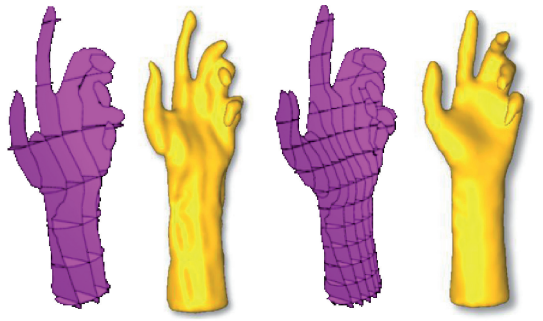
Fig. 13. Reconstructed heart.

9. EXPERIMENTAL RESULTS

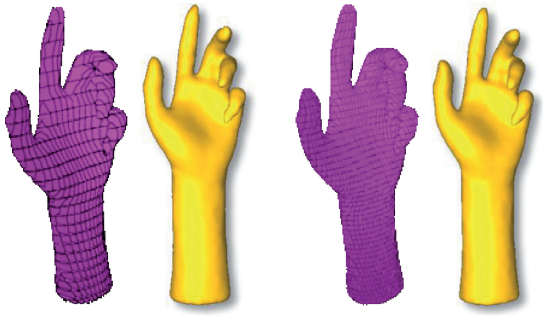
We show some examples of running our algorithm in several settings. First, we provide the reconstruction from cross-sections of some human vertebrae (Figure 12). This proves the ability of the algorithm to smoothly reconstruct a complex anatomical shape.

We next demonstrate on the heart model the ability of our algorithm to cope with inputs in which some regions are unknown. Notice, in Figure 13, how the missing information in the vicinity of the arteries is robustly completed with the help of the nearby slices. We compare this result to that of Barequet and Vaxman [2009] (which, had no unknown regions been present in the input, would have been identical to that of Liu et al. [2008]). Their work creates some topological noise and other low-frequency noise (the bottom of the heart) which is hard to diffuse, and which results from the (approximate) closest-point method.

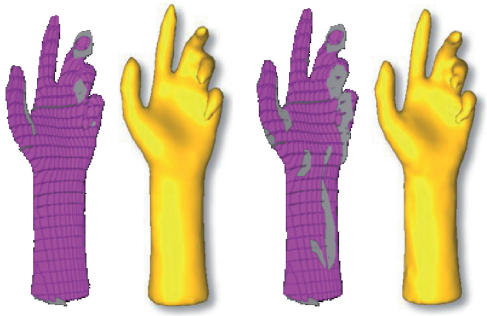
Another example is the reconstruction of the hand model (Figure 14), where we vary two parameters. One parameter is in the number of slices, and the other is the size of the unknown



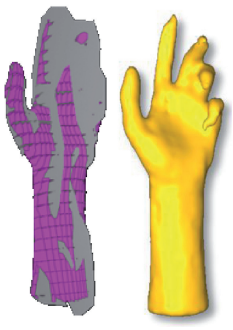
13 slices 25 slices



50 slices 100 slices

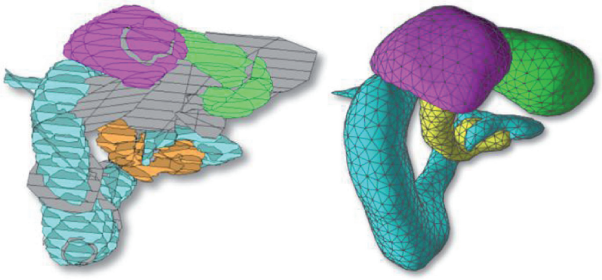


slightly noisy moderately noisy

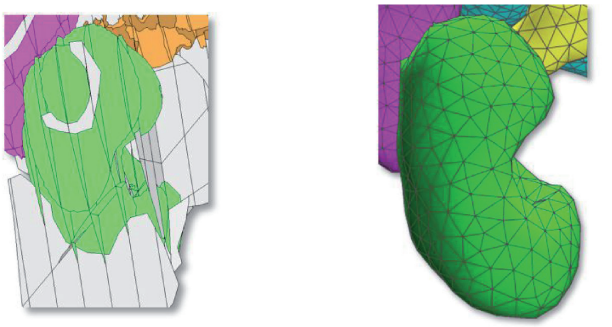


very noisy

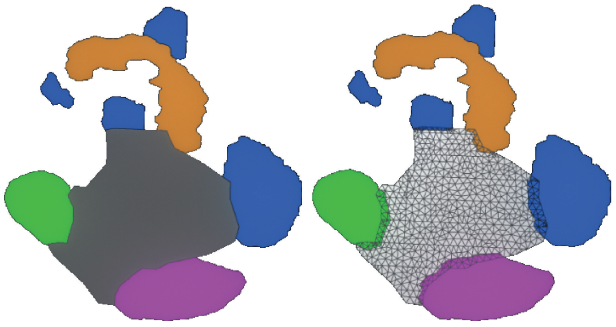
Fig. 14. The hand reconstructed from an increasing number of slices, and with an increasing amount of noise.



input slices reconstruction



closeup on noisy region closeup on completed region



noisy slice completed slice

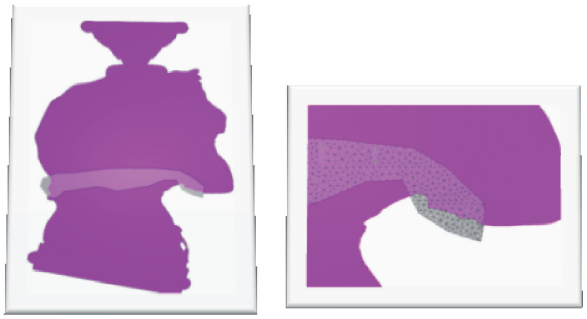
Fig. 15. Internal organs with noisy regions.

regions. Note that even with a small number of slices, or with a fairly large amount of unknown regions, the algorithm still performs well. We show two more general examples of our algorithm: reconstruction of a unilabeled lion vase in Figure 16, and an example of multilabeled reconstruction of a group of internal organs (stomach, spleen, pancreas, and another kidney) in Figure 15, which depicts the generality of the algorithm, as it is both multilabeled and has unknown regions.

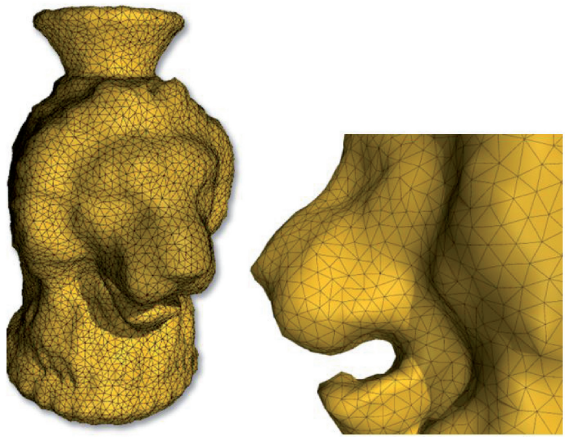
Finally, we show examples of online reconstruction in Figure 17 (unilabeled) and Figure 18 (multilabeled), where we see how the armadillo, liver and kidney shapes improve during the process.

Table I shows the vital statistics of the inputs, and Table II shows the algorithm's performance on them, for both CPU and GPU implementations (completion time is negligible), running on an Intel i7 2.7 GHz machine with 8GB RAM incorporating a Quadro FX 5800

(a) input (b) noisy slice

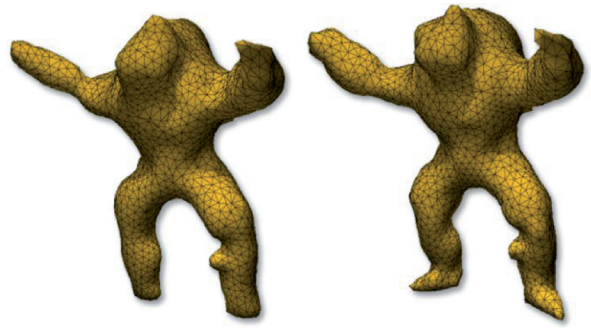


(c) completion of slice (d) closeup on completion



(e) mesh (f) closeup

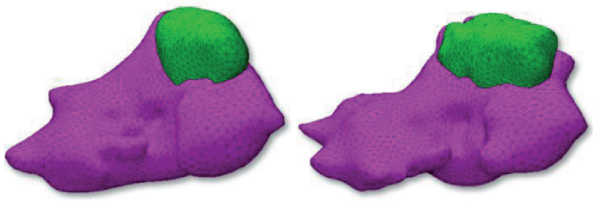
Fig. 16. Reconstruction of lion vase.



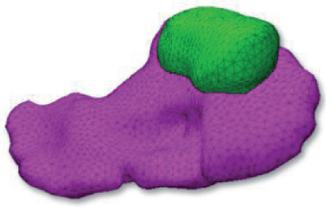
(a) 5 slices (2.7 sec) (b) 10 slices (1.7 sec)

(c) 42 slices (6.0 sec)

Fig. 17. Online reconstruction of armadillo. Timings are cost of last slice added.



(a) 4 slices (4.6 sec) (b) 10 slices (5.1 sec)



(c) 35 slices (7.7 sec)

Fig. 18. Online reconstruction of a liver and a kidney. Timings are cost of last slice added.

graphics card. Many important characteristics of the algorithm may be gleaned from the tables. First, we see that the GPU significantly reduces runtimes in most cases, sometimes up to a factor of five (see the online liver and kidney example). Overall, the runtimes are very fast, approaching interactive-time in the online cases, where the only remaining bottleneck is the serial meshgeneration algorithm. However, we see that the number of octree evaluations required

Table I. Reconstruction Statistics

Object name	#Slices	#Input vertices	#Arr. Faces	#Out. Tris (K)	#Octr. Evals. (K)		#CGAL Evals. (K)
					CPU	GPU	
Vertebrae	36	14815	25727	23	226	403	210
Heart	37	5942	8254	10	156	291	94
Internal Organs	34	19764	8737	5	137	226	102
Hand	100	6373	15606	12	122	213	198
(very) Noisy Hand	50	3560	4056	11	100	196	126
Hand – 100th slice addition	99	6360	15300	12	1.4	2.1	198
Armadillo 5th slice addition	4	1744	84	7	105	188	46
Armadillo 10th slice addition	19	7570	5181	6	38	56	39
Armadillo 42nd slice addition	41	16654	49380	11	13	18	122
Offline Armadillo	42	16775	52542	11	162	255	122
Liver & Kidney–4th slice addition	3	753	96	4	152	315	142
Liver & Kidney–35th slice addition	34	4511	24617	16	51	75	281
Offline Liver & Kidney	35	4723	26675	16	153	290	279
Vase Lion	63	33981	65629	25	276	438	100

Online statistics measured before addition.

Table II. Runtimes (in seconds)

Object name	Arr.	Octree		Smoothing	Mesh Gen.	Total	
		CPU	GPU			CPU	GPU
Vertebrae	4.7	9.3	6.1	2.8	3.3	20.4	16.9
Heart	1.9	6.8	2.8	0.6	2.2	11.8	7.8
Internal Organs	4.4	11	6.1	4	2.9	20	15.1
Offline Hand	6.8	2.7	3.7	1	2.5	13.5	14.5
(very) Noisy Hand	1.9	3.6	2.8	1.5	1.7	9.2	8.4
Hand – 100th slice addition	0.2	0.7	0.7	1	2.5	4.5	4.5
Online Armadillo 5th slice addition	0.1	10	1.2	0.4	1	11.5	2.7
Online Armadillo 10th slice addition	0.5	2.2	0.2	0.5	0.7	3.7	1.7
Online Armadillo 42nd slice addition	2	1.4	1.2	0.8	2	6.3	6.0
Offline Armadillo	7.2	6.1	6.5	0.8	2.3	16.4	17.0
Online Liver & Kidney–4th slice addition	0.1	12.7	1.5	0.5	2.5	17.6	4.6
Online Liver & Kidney–35th slice addition	1.2	2.7	1.6	1.3	3.5	8.8	7.7
Offline Liver & Kidney	3.2	5.2	4.3	1.3	3.5	13.2	12.3
Vase Lion	12.8	12.5	10.4	1.5	1.6	28.9	26.8

on the GPU is sometimes double that required on the CPU. This is due to the fact that some optimizations that are done on the CPU harm parallelism, therefore are not applicable on the GPU. This, along with data transition overheads, causes GPU runtimes to sometimes trail those of the CPU; evaluations spread among many cells are less parallelizable, as can be seen in the "offline Hand" and "offline Armadillo" entries of Table II. In addition, we can see that, paradoxically, an arrangement consisting of fewer cells increases the octree construction times. This can be explained by the fact that fewer cells mean more complex images on every face, in turn making the evaluations more complex. Another issue is that mesh generation takes roughly the same amount of time for both online and offline cases. This is due to the fact that this part of the algorithm is not incremental. On the same note, we observe that the other two stages of the algorithm are just as fast and efficient for the 40th or the 100th slice as they are for the 5th slice. When a slice is inserted into a low-complexity arrangement, the insertion takes little time; however, most of the octree is affected and needs to be rebuilt. When inserting the last slices, the arrangement insertion is more costly, but the zone of the slice is small, and so, the octree update costs less in comparison. All models were obtained from the AIM@SHAPE shape repository [AaS 2011].

10. CONCLUSIONS

We have described an algorithm which introduces barycentric interpolation to the problem of reconstruction from slices, and produces a smooth interpolating surface corresponding to a set of either unicolored or segmented cross-sections, even when the labeling of some regions is unknown. The algorithm also allows for an efficient online version. This solves the problem of reconstruction from arbitrarily oriented cross-sections at a new level of generality. The algorithm is intuitive and quite easy to implement. Our unknown region completion is a simple extension of the same mean-value transfinite interpolation. Our algorithm is also numerically robust in comparison to previous geometric tiling approaches, since small errors in the mean-value function evaluation only perturb the indicator function slightly, which does not significantly affect the results.

The main drawback of our algorithm is the need to evaluate the interpolated indicator function at a large number of points in space during iso-surface extraction, which can be quite expensive in a regular CPU computation. This bottleneck is alleviated using a natural GPU implementation, which is possible due to the parallel nature of the mean-value function evaluation, and which allows for

a fast offline and online zero-set extraction. Better results can be achieved by parallelizing the mesh generator, and allowing it to only update the surface in the affected zone.

Another drawback that our algorithm shares with the other state-of-the-art algorithms solving this problem is the use of arrangements, which makes this stage of the algorithm cumbersome for large numbers of slices (i.e., more than 150). Moreover, we are not able to control the topology of the resulting surface, nor provide any theoretical guarantee for the outcome, even though empirically we obtain quite nice results.

In addition, even though the results are smooth and consistent, there is an apparent loss of detail when slicing an object with cross-sections and then reconstructing (as seen in Figure 16) from them. This is common to all cross-section reconstruction methods, as far as we know. An interesting next step would be to create detail-preserving reconstructions, which might reproduce certain surfaces (e.g., sweep surfaces sampled by cross-sections parallel to the base), and incorporate the reconstruction of sharp features.

Finally, our choice of mean-value interpolation was motivated by the closed-form equations that fit our problem well. An interesting direction for future research is generalization of the barycentric method on an arrangement to any type of transfinite barycentric interpolation, for example, harmonic functions, and theoretical investigation of their properties.

APPENDIX

A. TRANSFINITE MEAN-VALUE COORDINATES

We provide here the formulae for the weighting functions for the corners using our constant integration method, and using the piecewise-linear triangulation method. Figure 19 illustrates the setup.

A Polyhedron with Piecewise-Constant Boundary Values

We want to assign weights to the corners of a planar region on the boundary of a volume, where this region has a constant function value. Towards that end, we wish to compute the integral

$$I_P(x) = \iint_P \frac{1}{\|y-x\|} d\sigma(y),$$

where P is a planar polygon (possibly with holes), x is a point in three-dimensional space (not on P's plane), y is a point on P, and $d\sigma(y)$ is the area of an area element of P at y projected onto the unit sphere centered at x.

Let u and v be two orthogonal unit vectors parallel to P's plane and n the plane's normal vector. Thus, x and y may be expressed in the coordinate system (u, v, n) such that $y\text{-}x = \alpha u + \beta v + \gamma n$, $\|y\text{-}x\|^2 = \alpha^2+\beta^2+\gamma^2, \sigma(y) = x + [\alpha u + \beta v + \gamma n]/[\alpha^2+\beta^2+\gamma^2]^{1/2}$, and

$$d\sigma(y) = \left\| \frac{d\sigma}{d\alpha} \times \frac{d\sigma}{d\beta} \right\| d\alpha d\beta = \frac{|\gamma|}{\|y-x\|^3} d\alpha d\beta.$$

resulting in

$$I_P(x) = \iint_P \frac{|\gamma|}{\|y-x\|^4} d\alpha d\beta.$$

Regarding this integral as a flux integral

$$I_P(x) = \iint_P (\nabla \times F(y-x)) \cdot n d\sigma(y)$$

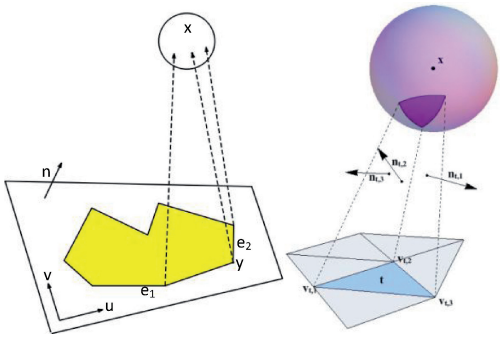


Fig. 19. (left) Mean-value projection of a constant-valued polygon on the unit sphere centered at x. (right) Mean-value projection of a linear-valued triangle on the unit sphere centered at x.

a possible F is

$$\begin{aligned} F &\equiv F_u u + F_v v + F_n n \\ F_u(y-x) &= \frac{\gamma}{2(\alpha^2+\gamma^2)} \\ &\quad \times \left[\frac{\beta}{(\alpha^2+\beta^2+\gamma^2)^{1/2}} + \frac{\tan^{-1}(\beta(\alpha^2+\gamma^2)^{-1/2})}{(\alpha^2+\gamma^2)^{1/2}} \right] \\ F_v &= 0. \end{aligned}$$

The F_n component is rather complicated, but is irrelevant. By Stokes theorem

$$I_P(x) = \iint_P (\nabla \times F(y-x)) \cdot n d\sigma(y) = \int_{\partial P} F(y-x) \cdot dy.$$

The integral may be computed as a sum of integrals over the edges of ∂P. Assume that P consists of properly oriented edges (exterior edges oriented CCW, and holes oriented CW), and let $y = x + au + (ca+d)v, a \in [a_1, a_2]$ be a parameterization of an edge e with endpoints $y_1 = x + a_1 u + b_1 v$ and $y_2 = x + a_2 u + b_2 v$. We then get the weight for edge e.

$$I_e(x) = \int_e F \cdot dy = \int_{a_1}^{a_2} (F_u, 0, F_n) \cdot (1, c, 0) da = \int_{a_1}^{a_2} F_u da$$

Denoting

$$L_e(y-x) = \frac{\alpha}{\gamma} \frac{\tan^{-1}\left(\frac{\beta}{(\alpha^2+\gamma^2)^{1/2}}\right)}{(\alpha^2+\gamma^2)^{1/2}} + \frac{d}{\gamma} \frac{\tan^{-1}\left(\frac{(c\beta+\gamma)}{(d^2+(1+c^2)\gamma^2)^{1/2}}\right)}{(d^2+(1+c^2)\gamma^2)^{1/2}}$$

the integral reduces to

$$I_e(x) = L_e(y_2 - x) - L_e(y_1 - x).$$

Note that when $u_1 = u_2$ (which renders the parameterization impossible) the integral simply vanishes.

$$I_e(x) = 0 - 0 = 0$$

If we reorganize the integral over ∂P, expressed until now as a sum over edges of ∂P, to a sum over the corners of ∂P, we find that the weight function of a corner at point $y \in \partial P$ is given through the weights of its incident edges e_1 and e_2.

$$w_y(x) = L_{e_2}(y-x) - L_{e_1}(y-x)$$

A Polyhedron with Piecewise-Linear Boundary Value on Triangular Faces

Following Ju et al. [2005a], given a point x inside a cell, all triangles are projected to a unit sphere centered at x (see Figure 18 (right)). The *mean vector* m_t is computed for each triangle t, and then the weights associated with each of the three corners $\{v_{t,1}, v_{t,2}, v_{t,3}\}$ of t are defined as

$$w_{t,i}(x) = \frac{m_t \cdot n_{t,i}}{n_{t,i} \cdot (v_{t,i} - x)},$$

where $n_{t,i}$ is the unit normal to the plane containing x, $v_{t,i-1}$ and $v_{t,i+1}$, and t's angle at $v_{t,i}$

$$m_t = -\frac{1}{2}\sum_{i=1}^{3} \theta_{t,i} n_{t,i}.$$

REFERENCES

AMANATIDES, J. AND WOO, A. 1987. A fast voxel traversal algorithm for ray tracing. In *Proceedings of the Eurographics Conference*. 3–10.

AMINI, O., BOISSONNAT, J.-D. AND MEMARI, P. 2010. Geometric tomography with topological guarantees, In *Proceedings of the Symposium on Computational Geometry*. To appear.

BAJAJ, C., COYLE, E., AND LIN, K. 1996. Arbitrary topology shape reconstruction from planar cross sections. *Graph. Models Image Process. 58*, 524–543.

BAJAJ. C. AND EDWARDS, J. 2010. Topologically correct reconstruction of tortuous contour forests. In *Proceedings of the ACM Symposium on Solid and Physical Modeling*. 51–60.

BAREQUET, G., EPPSTEIN, D., GOODRICH, M. T., AND VAXMAN, A. 2008. Straight skeletons of three-dimensional polyhedra. In *Proceedings of the 16th Annual European Symposium on Algorithms*. Lecture Notes in Computer Science, vol. 5193, 148–160.

BAREQUET, G., GOODRICH, M., LEVI-STEINER, A., AND STEINER, D. 2004. Contour interpolation by straight skeletons. *Graph. Models 66*, 3, 245–260.

BAREQUET, G. AND SHARIR, M. 1996. Piecewise-linear interpolation between polygonal slices. *Comput. Vis. Image Understand. 63*, 251–272.

BAREQUET, G. AND VAXMAN, A. 2009. Reconstruction of multi-label domains from partial planar cross-sections. *Comput. Graph. Forum 28*, 5, 1327–1337.

BATNITZKY, S., PRICE, H., COOK, P., COOK, L., AND DWYER, S. J. 1981. Three-dimensional computer reconstruction from surface contours for head ct examinations. *J. Comput. Assist. Tomograph. 5*, 60–67.

BOGUSH, A., TUZIKOV, A., AND SHEYNIN, S. 2004. 3D object reconstruction from non-parallel cross-sections. In *Proceedings of the International Conference on Pattern Recognition*, Vol. 3, 542–545.

BOISSONNAT, J.-D. AND GEIGER, B. 1992. Three dimensional reconstruction of complex shapes based on the Delaunay triangulation. Tech. rep. 1697, INRIA-Sophia Antiplis.

BOISSONNAT, J.-D. AND MEMARI, P. 2007. Shape reconstruction from unorganized cross-sections. In *Proceedings of the Symposium on Geometry Processing*. 89–98.

BOISSONNAT, J.-D. 1988. Shape reconstruction from planar cross sections. *Comput. Vis. Graph. Image Process. 44*, 1–29.

CGAL. *Computational geometry algorithms library*. http://www.cgal.org.

CHENG, S. AND DEY, T. 1999. Improved construction of Delaunay based contour surfaces. In *Proceedings of the 5th ACM Symposium on Solid Modeling and Applications*. 322–323.

CHOI, Y. AND PARK, K. 1994. A heuristic triangulation algorithm for multiple planar contours using anextended double branching procedure. *Vis. Comput. 10*, 372–287.

CHRISTIANSEN, H. AND SEDERBERG, T. 1978. Conversion of complex contour line definitions into polygonal element mosaics. *Comput. Graph. 13*, 187–192.

CSEBFALVI, B., NEUMAN, L., KANITSAR, A., AND GRØLLER, E. 2002. Smooth shape-based interpolation using the conjugate gradient method. In *Proceedings of the Vision, Modeling, and Visualization Conference*. 123–130.

DANCE, C. AND PRAGER, R. 1997. Delaunay reconstruction from multiaxial planar cross-sections. Tech. rep. CUED/F-INFENG/TR273, Cambridge University, UK.

DYKEN, C. AND FLOATER, M. 2009. Transfinite mean value interpolation. *Comp. Aid Geom. Des. 26*, 117–134.

EKOULE, A., PEYRIN, F., AND ODET, C. 1991. A triangulation algorithm from arbitrary shaped multiple planar contours. *ACM Trans. Graph. 10*, 182–199.

FELKEL, P. AND OBDRŽÁLEK, Š. 1999. Improvement of Oliva's algorithm for surface reconstruction from contours. In *Proceedings of the Spring Conference on Computer Graphics*. 254–263.

FIX, J. AND LADNER, R. 1998. Multiresolution banded refinement to accelerate surface reconstruction from polygons. In *Proceedings of the Symposium on Computational Geometry*. 240–248.

FLOATER, M. S., KOS, G., AND REIMERS, M. 2005. Mean value coordinates in 3D. *Comp. Aid. Geom. Des. 22*, 7, 623–631.

FUCHS, H., KEDEM, Z., AND USELTON, S. 1977. Optimal surface reconstruction from planar contours. *Comm. ACM* 20.

GANAPATHY, S. AND DENNEHY, T. 1982. A new general triangulation method for planar contours. *ACM Trans. Graph. 16*, 69–75.

GRANADOS, M., HACHENBERGER, P., HERT, S., KETTNER, L., MEHLHORN, K., AND SEEL, M. 2003. Boolean operations on 3D selective NEF complexes: Data structure, algorithms, and implementation. In *Proceedings of the European Symposium on Algorithms*. 654–666.

JU, T., SCHAEFER, S., AND WARREN, J. 2005. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph. 24*, 561–566.

JU, T., WARREN, J., CARSON, J., EICHELE, G., THALLER, C., CHIU, W., BELLO, M., AND KAKADIARIS, I. 2005. Building 3D surface networks from 2D curve networks with application to anatomical modeling. *Vis. Comput. 21*, 764–773.

KAZHDAN, M., BOLITHO, M., AND HOPPE, H. 2006. Poisson surface reconstruction. In *Proceedings of the Symposium on Geometry Processing*. 61–70.

KEHTARNAVAZ, N. AND FIGUEIREDO, R. D. 1988. A framework for surface reconstruction from 3D contours. *Comput. Vis. Graph. Image Process. 42*, 32–47.

KEHTARNAVAZ, N., SIMAR, L., AND FIGUEIREDO, R. D. 1988. Asyntactic/semantic technique for surface reconstruction from cross-sectional contours. *Computer Vis. Graph. Image Process. 42*, 399–409.

KEPPEL, E. 1975. Approximating complex surfaces by triangulation of contour lines. *IBM J. Res. Devel. 19*, 2–11.

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H. P. 1998, Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of the SIGGRAPH*. 105–114.

LIU, L., BAJAJ, C., DEASY, J., LOW, D., AND JU, T. 2008. Surface reconstruction from non-parallel curve networks. *Comput. Graph. Forum 27*, 155–163.

MEYERS, D., SKINNER, S., AND SLOAN, K. 1992. Surfaces from contours: The correspondence and branching problems. *ACM Trans. Graph. 11*, 228–258.

OLIVA, J.-M., PERRIN, M., AND COQUILLART, S. 1996. 3D reconstruction of complex polyhedral shapes from contours using a simplified generalized Voronoi diagram. *Comput. Graph. Forum 15*, 397–408.

PAYNE, B. AND TOGA, A. 1994. Discrete Sibson interpolation. *IEEE Comput. Graph. Appl. 14*, 28–35.

RINEAU, L. AND YVINEC, M. 2007. A generic software design for Delaunay refinement meshing. *Comp. Geom. Theory Appl. 38*, 100–110.

SHANTZ, M. 1981. Surface definition for branching contour defined objects. *Comput. Graph. 15*, 242–270.

SLOAN, K. AND PAINTER, J. 1988. Pessimal guesses may be optimal: A counterintuitive search result. *IEEE Trans. Patt. Anal. Mach. Intell. 10*, 945–955.

TAUBIN, G. 1995. A signal processing approach to fair surface design. *Proceedings of the of the 22nd Annual Conference on Composite Graphics and Interactive Techniques*. 351–358.

TURK, G. AND O'BRIEN, J. F. 1999. Shape transformation using variational implicit functions. In *Proceedings of the ACM SIGGRAPH*. 335–342.

WANG, Y. AND AGGARWAL, J. 1986. Surface reconstruction and representation of 3D scenes. *Patt. Recogn. 19*, 197–207.

WELZL, E. AND WOLFERS, B. 1994. Surface reconstruction between simple polygons via angle criteria. *J. Symbol. Comput. 17*, 351–369.

ZYDA, M., JONES, A., AND HOGAN, P. 1987. Surface construction from planar contours. *Comput. Graph. 11*, 393–408.

Received July 2010; revised February 2011; accepted June 2011