Eurographics Symposium on Geometry Processing (2004)
R. Scopigno, D. Zorin, (Editors)

Topology Preserving Surface Extraction Using Adaptive Subdivision

Gokul Varadhan[1], Shankar Krishnan[2], TVN Sriram[1], and Dinesh Manocha[1]

[1]Department of Computer Science, University of North Carolina, Chapel Hill, U.S.A
[2]AT&T Labs - Research, Florham Park, New Jersey, U.S.A
http://gamma.cs.unc.edu/recons

Abstract

We address the problem of computing a topology preserving isosurface from a volumetric grid using Marching Cubes for geometry processing applications. We present a novel topology preserving subdivision algorithm to generate an adaptive volumetric grid. Our algorithm ensures that every grid cell satisfies two local geometric criteria: a complex cell criterion and a star-shaped criterion. We show that these two criteria are sufficient to ensure that the surface extracted from the grid using Marching Cubes has the same genus and connectedness as that of the exact isosurface. We use our subdivision algorithm for accurate boundary evaluation of CSG combinations of polyhedra and low degree algebraic primitives, translational motion planning, model simplification and remeshing. The running time of our algorithm varies between a few seconds for simple models composed of a few thousand triangles to tens of seconds for complex polyhedral models represented using hundreds of thousands of triangles.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

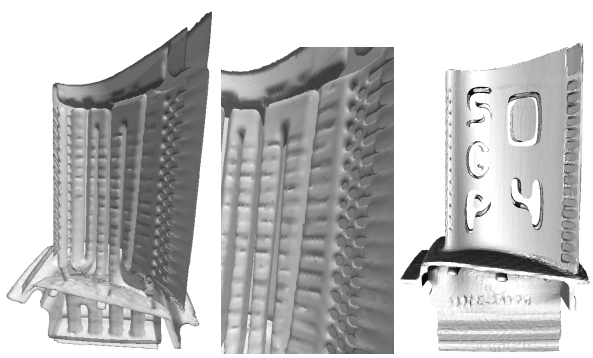
1. Introduction

Implicit surface representations have become increasingly common in computer graphics and geometric modeling. An implicit surface is typically defined as an isosurface of a scalar field. Such representations are useful for different applications including geometric modeling, volume rendering, morphing, path planning, swept volume computation, and sculpting digital characters. Compared to other surface representations (e.g. parametric surfaces), implicit surfaces offer many advantages when performing geometric operations like union, intersection, difference, blending and warping.

Our goal is to use implicit surface representations for geometric computations including Boolean operations (i.e. union, intersection and difference), Minkowski sum computation, simplification, and remeshing. In each case, we wish to obtain an accurate polygonal approximation of the boundary of the final solid. At a broad level, our algorithm performs three main steps:

1. **Sampling:** Generate a volumetric grid and compute a scalar field (e.g, a signed distance field) at its corner grid points
2. **Operation:** For each geometric operation, perform an analogous operation (e.g., min/max) on the scalar fields of the primitives.
3. **Isosurface Extraction:** Perform isosurface extraction using Marching Cubes [LC87] or its variant to obtain an approximation of the final surface.

We refer to the boundary of the final solid as the *exact isosurface* and the output of Marching Cubes as the *extracted isosurface*. Our goal is to ensure that the extracted isosurface has the same topology as the exact isosurface. In particular, it should have the same number of connected components and genus as the exact isosurface. Preserving the topology is important in many applications. For example, a geometric model designed using Boolean or offsetting operations can have a different topology as compared to the original primitives. In



Turbine(interior) Zoomed View Boolean Operations

Figure 1: *This* 1.7*M triangle model of a turbine has a high genus and many holes in the interior. We highlight the application of our novel topology preserving voxelization, simplification and boundary evaluation algorithms on this complex model. The simplified model of the turbine has* 511*K triangles and we show a zoomed view in the center image. We perform five difference (Boolean) operations on the turbine model and reconstruct the boundary of the final solid. Our boundary evaluation algorithm preserves the topology of the final solid and accurately computes all the components. Overall, our algorithm can perform such geometric computations on complex models in tens of seconds and give rigorous guarantees in terms of preserving the topology of the final surface.*

CAD, accurately reconstructing all the features and holes in the final solid is important. Many simplification algorithms used for collision detection and finite element analysis need to preserve the genus of the original model and compute a surface approximation with tight bounds on the geometric error. The geometric models used to represent the organs in medical datasets consist of holes and handles. It is important to preserve these topological features during visualization and analysis. The geometric model of a molecule consists of tunnels and cavities. The tunnels often act as atomic sieves that can aid the biochemical processes and preserving these features can be important for rational drug design.

The topology of an isosurface reconstructed using Marching Cubes can differ from that of the exact isosurface. The extracted surface may have different genus, missing components or unwanted handles. The main reason for these inaccuracies is an inadequate sampling of the exact isosurface, i.e., the resolution of the volumetric grid. To overcome these topological inaccuracy problems, many applications generate samples on a fine grid. However, the use of a fine grid can result in other problems. First, there may still be no guarantees on the topology of the extracted isosurface. Second, a fine grid increases the storage overhead and the extracted surface can have a high number of polygonal primitives. Finally, it is computationally expensive to use a fine grid. Recent work on adaptive grid generation and subdivision algorithms overcomes many of these problems. However, none of the subdivision algorithms can give rigorous guarantees on the topology of the reconstructed isosurface.

Main Contributions: We present a novel approach to compute a topology preserving isosurface using Marching Cubes for geometry processing applications. We present a conservative sampling criteria such that if every cell in the volumetric grid satisfies the criteria, then the extracted isosurface will have the same topology as the exact isosurface. Our approach is based on two geometric criteria: *complex cell* criterion and *star-shaped* criterion. The complex cell criterion guarantees that every intersection of an isosurface with that cell results in some sign changes of the scalar field along the vertices of that cell. The star-shaped criterion, as the name suggests, ensures that the isosurface restricted to a cell is *star-shaped*. We show that if every cell in the grid satisfies these two criteria, then the topology of the isosurface is preserved during isosurface extraction.

We present an adaptive subdivision algorithm to generate a volumetric grid such that every grid cell satisfies the above two geometric criteria. We present efficient computational techniques to check whether these two criteria are satisfied during grid generation. We use *max-norm* distance computation to verify the complex cell criterion and *kernel* computation to verify the star-shaped criterion. Our algorithm can easily perform these computations on polyhedra, algebraic or parametric primitives and their Boolean combinations. We have used our algorithm to perform accurate boundary evaluation of CSG combinations of polyhedral and low degree algebraic primitives, model simplification and remeshing of complex models. In each case, we compute a topology preserving polygonal approximation of the final surface. We also have applied our algorithm to compute the Minkowski sum of polyhedral models and used it for exact motion planning with translational degrees of freedom. The running time of our algorithm varies between a few seconds for simple models consisting of thousands of triangles and tens of seconds on complex primitives represented using hundreds of thousands of triangles on a Pentium IV PC.

Some *novel aspects* of our work include:

- Conservative sampling criteria — *complex cell* and *star-shaped* criteria — for the volumetric grid such that the topology of the isosurface is preserved.
- An efficient topology preserving subdivision algorithm to generate an octree satisfying the above criteria.
- A fast algorithm to compute topology preserving simplifications of a complex polygonal model.
- Efficient and accurate algorithms for boundary evaluation, remeshing, and translational motion planning.

As compared to prior work, the main benefits of our algorithm are its speed, simplicity, accuracy and reliability. It not only offers the efficiency and simplicity of Marching Cubes based reconstruction, but also the ability to guarantee the topology of the isosurface with limited additional overhead.

2. Previous Work

We give a brief overview of prior work on isosurface extraction for geometric applications. Marching Cubes, proposed by Lorensen and Cline [LC87], is a standard approach to extract an isosurface from a volume raster of scalar values. It is well known that there are certain grid cell configurations that result in topologically ambiguous output. Many extensions to the original Marching Cubes reconstruction algorithm have been proposed that resolve these ambiguities and generate topologically consistent isosurfaces [WG90, CGMS00]. Most of these algorithms deal with the problem of extracting a surface from a fixed volumetric dataset, or a given sampled scalar field. These approaches modify Marching Cubes to produce a topologically consistent surface. However, to the best of our knowledge, none of the previous algorithms on topological considerations for Marching Cubes provide guarantees on preserving topology of the exact isosurface.

[SH97, BCSV04] provide guarantees on the topology of implicit surface polygonization based on critical point analysis. These approaches assume that the implicit surface is smooth and require a priori knowledge of all the critical points. It is not clear how to apply these approaches to perform Boolean combinations where the final surface is not smooth and whose topology can be very different from those of the input primitives.

The original Marching Cubes algorithm is unable to extract high quality triangle meshes with sharp features from the volumetric data. Recently, few extensions have been proposed that can reconstruct sharp features and reduce aliasing artifacts in the reconstructed model [KBSS01, JLSW02]. The Marching Cubes algorithm can produce meshes with too many small and skinny triangles, which typically require im-

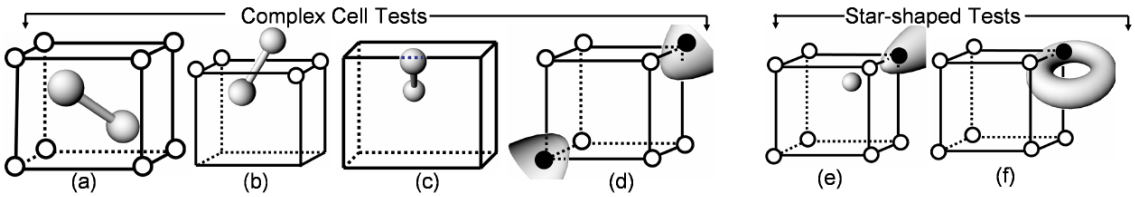


Figure 2: *This figure shows the different cases corresponding to the complex cell and star-shaped test. Figs (a), (b), (c) and (d) show cases of complex voxel, complex face, complex edge, and topological ambiguity. The white and black circles denote positive and negative grid points respectively. Fig. (e) shows the case where the surface is not star-shaped w.r.t a voxel. In Fig (f), the restriction of the surface to the right face of the cell is not star-shaped.*

provement using decimation, smoothing, or remeshing. Algorithms based on adaptive hierarchies can overcome some of these problems and result in isosurfaces with fewer triangles [FPRJ00, SFYC96]. However, these algorithms do not provide guarantees on the topology of the reconstructed surface.

Many volumetric approaches have used the topological properties for model repair and simplification of polygonal models [NT03] or generating an isosurface without additional handles or cavities for scanned datasets [GW01, WHDS02, BK02]. Often their input data contains noise due to the scanning process. They use a priori knowledge about the topology of the input data to eliminate the topological artifacts that arise from the noise. These approaches are complementary to our work. Moreover, in case of Boolean operations, we may not have a priori information about the topology of the final solid.

Since we are building upon our previous work [VKK*03], we would like to make our contribution clear. We had proposed an adaptive grid generation algorithm in [VKK*03] based on a *complex cell* test. However, the complex cell test by itself is insufficient; as a result, the previous algorithm did not provide any topological guarantees on the output. In our current work, we show that by using an additional *star-shaped* test, it is possible to ensure the correct topology.

3. Notation and Preliminaries

We assume that the isosurface is a closed manifold and is free from artifacts such as self intersections. The isosurface is defined as the zero set of a scalar field, usually a distance field or a function of distance fields defined over a set of primitives. All our primitives are assumed to be closed and compact solids, whose boundary is a manifold.

We present the notation used in the rest of this paper. The exact isosurface and our approximation are denoted as $\mathscr{E}$ and $\mathscr{A}$ respectively. The letter C denotes a grid cell used for sampling. We assume a cell consists of a cube-shaped voxel, six faces, twelve edges, and eight vertices. Given a cell C, we denote $\mathscr{E}_C = \mathscr{E} \cap C$ ($\mathscr{E}$ restricted to the cell) and $\mathscr{A}_C = \mathscr{A} \cap C$. We use upper case letters such as P, Q, P_1, and S to refer to the geometric primitives. We use lower case bold letters such as $\mathbf{p}, \mathbf{q}$ to refer to points in $\mathbb{R}^3$.

Let S be a nonempty subset of $\mathbb{R}^n$. The set Kernel(S) consists of all $\mathbf{s} \in S$ such that for any $\mathbf{x} \in S$, we have $\mathbf{s} + \lambda(\mathbf{x} - \mathbf{s}) \in S, \forall \lambda \in [0,1]$. S is *star-shaped* if Kernel$(S) \neq \emptyset$. We call a point belonging to Kernel(S) as Origin(S).

A surface is a topological disk if it is *connected* as well as *simply connected*. A manifold is *connected* if for any two points on the manifold, there exists a path between them in the set. A manifold is said to be *simply connected* if any simple closed curve on the manifold can be shrunk to a point continuously in the set. A homeomorphism is a continuous bijective mapping with a continuous inverse. Two objects are topologically equivalent if there exists a homeomorphism between them.

We define a curve I embedded in a planar face bounded by a set of edges $\mathscr{G}$ to be a *boundary curve* if $I \cap \mathscr{G} \neq \phi$. In other words, I cannot lie completely inside the face.

4. Topology Preserving Surface Extraction

Our goal is to design a criterion to sample a scalar field so that reconstruction methods like Marching Cubes and its variants preserve the topology of the isosurface. In the rest of the paper, we refer to Marching Cubes and its variants as *MC-like algorithms* and their output as *MC-like reconstruction* to simplify the exposition. In this section, we present a conservative criterion for sampling and prove the topological guarantees they provide.

4.1. Overview of Marching Cubes Algorithm

The Marching Cubes algorithm [LC87] generates a polygonal reconstruction of an isosurface of a sampled scalar field in $\mathbb{R}^3$. It operates on each grid cell of the volume dataset independently. It tests if the isosurface intersects the cell by comparing the scalar field values at the vertices of the grid cell with the isovalue. We assume the isovalue to be zero in the rest of the paper. If the grid vertices of a cell have different signs, Marching Cubes estimates intersection points along the cell edges and connects them to obtain a polygonal reconstruction within the cell. If all the grid vertices of a cell have the same sign, then it assumes that the isosurface does not intersect the cell and does not perform a reconstruction within the cell.

The accuracy of the extracted surface is governed by the rate of sampling, i.e., the resolution of the underlying volumetric grid. Due to insufficient resolution, the isosurface may have features smaller than the grid cell size or may have a complicated topology within a grid cell. This results in a reconstructed surface that can miss small surface features, have unwanted handles or multiple components, and hence incorrect topology. In addition, there are certain grid cell sign configurations that result in topologically ambiguous output (see Fig. 2(d)).

4.2. Topological Reliability of MC-Like Reconstruction

Our goal is to ensure that MC-like reconstruction algorithms generate an approximate surface $\mathscr{A}$ that is topologically equivalent to the exact surface $\mathscr{E}$. This can be guaranteed by making sure that the assumptions made by these algorithms are valid — that the exact surface should intersect each grid cell in a simple manner, and should have a simple topology within each cell. In particular, we ensure that the exact surface within each cell is a topological disk. We first present sufficient criteria to ensure this property. Then we show that this property can be used to guarantee the correct topology of $\mathscr{A}$. We start by defining two properties of the exact surface, $\mathscr{E}$, inside each cell.

Complex cell: A cell is *complex* if it has a *complex voxel*, *face*, *edge*, or an *ambiguous sign configuration*. We define a voxel (face) of a grid cell to be *complex* if it intersects $\mathscr{E}$ and the grid vertices belonging to the voxel (face) do not exhibit a sign change (see Figs. 2(a) & 2(b)). An edge of the grid cell is said to be *complex* if $\mathscr{E}$ intersects the edge more than once (see Fig. 2(c)).

As mentioned earlier, MC-like algorithms cannot handle certain sign combinations in a topologically reliable manner. There are two types of ambiguity — *face ambiguity* and *voxel ambiguity* [WG90]. When the signs at the vertices of a single face alternate during counterclockwise (or clockwise) traversal, the resulting configuration is a face ambiguity. A voxel ambiguity results when any pair of diagonally opposite vertices have one sign while the other vertices have a different sign (see Fig. 2(d)). Either of these cases is defined as an ambiguous sign configuration.

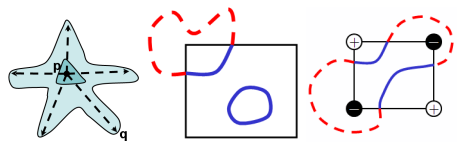
Intuitively, the complex cell criterion ensures that the surface intersects the grid cell in a simple manner in most cases. However, this criterion alone is not sufficient. There are cases where the cell may not be complex, but the isosurface may have a complicated topology within the cell (see Figs. 2(e) & 2(f)). We circumvent such situations by enforcing a star-shaped criterion within all the grid cells.

Star-shaped w.r.t. cell: Let S be a nonempty subset of $\mathbb{R}^n$. S is *star-shaped* if Kernel$(S) \neq \emptyset$. Intuitively, a star-shaped primitive has a point (called the origin) such that all the points in the primitive are visible from the origin (see Fig. 3(a)). Let $P \subseteq \mathbb{R}^3$ denote the closed solid defined by the isosurface $\mathscr{E}$. P is the set of points that lie inside $\mathscr{E}$. We define $\mathscr{E}$ to be *star-shaped with respect to* (w.r.t) *a voxel* V if $P_V = P \cap V$ is star-shaped. Similarly, we define $\mathscr{E}$ to be *star-shaped w.r.t a face* F if the two-dimensional set, $P_F = P \cap F$, is star-shaped. $\mathscr{E}$ is said to be *star-shaped w.r.t a cell* if it is star-shaped w.r.t the cell's voxel, and each of its six faces. Figs. 2(e) & 2(f) show two cases where the surface is not star-shaped w.r.t a voxel or a face of the cell.

DEFINITION 1 *A cell C is* MC-compatible *if it is* not complex *and $\mathscr{E}$ is* star-shaped *w.r.t. C.*

We start by listing few known properties of MC-like reconstruction algorithms and MC-compatible cells.

- **Property 1.** The signs of the scalar field at all the grid vertices are preserved in the surface reconstructed by a MC-like algorithm.
- **Property 2.** The reconstructed surface, $\mathscr{A}_C$, inside any MC-compatible cell is a topological disk. After eliminating the topologically ambiguous sign configurations, the remaining sign configurations always result in reconstruction homeomorphic to a disk.



(a) Star-shaped object (b) Intersection Curves

Figure 3: *Fig. (a) shows a star-shaped primitive and its kernel (shaded region in the middle). P is an origin of the kernel. Fig (b) supports proof of Lemma 1. It shows a grid cell face F and the intersection of the exact surface $\mathscr{E}$ with F. The figure on the left shows the intersection curve (in blue) consisting of two curve components, one of which is closed. As a result, it is not star-shaped. The figure on the right shows the case where the curve has two boundary curves. This results in a face ambiguity.*

We now present our main result related to topology-preserving reconstruction. If all the cells bounding $\mathscr{E}$ are MC-compatible, then the reconstructed surface, $\mathscr{A}$, has the same topology as $\mathscr{E}$. In order to prove this result, we first show that the exact surface within each cell is a topological disk. This property is then used to establish topological equivalence. We begin by presenting few lemmas.

LEMMA 1 *Let C be a MC-compatible cell with faces $F_i, i = 1,\ldots 6$. Then $\mathscr{E}_C \cap F_i$ is either empty or a single boundary curve.*

Proof: Consider a face F on C such that $I = \mathscr{E}_C \cap F$ is not empty. I can have no closed component. We prove this by contradiction. Suppose I has a closed component. Then I cannot have any other curve component because it will contradict the fact that I is star-shaped w.r.t F (see Fig. 3(b)-left). But if I is a single closed component, then F is a complex face. Therefore I cannot have any closed components.

Let us assume that I has multiple boundary curves. The complex-edge criterion ensures that a boundary curve intersects exactly two edges of the face F. It also ensures that two boundary curves cannot intersect the same edge. Therefore, I can have at most two boundary curves, each intersecting two edges of F. However, this results in a face ambiguity (see Fig. 3(b)-right). Hence, I can have only a single boundary curve.

LEMMA 2 *Let C be a MC-compatible cell. If $\mathscr{E}_C$ is not empty, then it is a connected surface with boundary.*

Proof: Let C enclose a voxel V. We prove that $\mathscr{E}_C$ cannot contain any closed component. Similar to the previous proof, the presence of a closed component would imply that either the primitive is not star-shaped w.r.t. V, or V is a complex voxel.

Let $\mathscr{E}_C$ have multiple surface components with boundary. The boundary of each surface component corresponds to a boundary curve on the faces of the cell. From the result of

Lemma 1, each face contains only one boundary curve. Furthermore, the complex face and complex edge criteria preclude the boundary curve from intersecting one or two faces. Therefore, each boundary curve intersects at least three faces. Since each face can have at most one boundary curve, $\mathscr{E}_C$ cannot have more than two surface components. The only way there can be two surface components is if two diagonally opposite cell vertices are inside the primitive while the others are outside (see Fig. 2(d)). This is the case of a voxel ambiguity. Therefore, $\mathscr{E}_C$ has at most one surface component with a boundary and is connected.

LEMMA 3 *Let C be a MC-compatible cell. If $\mathscr{E}_C$ is not empty, then it is a topological disk.*

Proof: The result from Lemma 2 shows that $\mathscr{E}_C$ has a single connected component with boundary. To show that it is a topological disk, we show that $\mathscr{E}_C$ is a simply connected surface. Let us assume that $\mathscr{E}_C$ is not simply connected. This means that $\mathscr{E}_C$ has two or more boundaries. The remainder of the proof is very similar to the argument in the proof of Lemma 2. Existence of two boundaries results in a voxel ambiguity. Therefore, $\mathscr{E}_C$ is simply connected. This proves that it is a topological disk. This concludes the proof.

We now show that both $\mathscr{E}$ and $\mathscr{A}$ intersect the grid cells in an identical fashion. Consider a MC-compatible cell C. Given any surface, suppose we associate a bit v with a voxel whose value is 1 or 0 depending on whether the voxel is intersected by the surface or not. Similarly, we can associate such a bit with each face and edge of the cell. Let $f_i,\ i = 1,\ldots,6$ and $e_j,\ j = 1,\ldots,12$ denote the bits associated with the ith face and jth edge respectively. These bits reflect the *intersection status* of the voxel, faces, and edges of the cell. We define the *intersection pattern* of cell C w.r.t the surface as the 19-bit tuple $(v, f_1,\ldots,f_6,e_1,\ldots,e_{12})$.

LEMMA 4 *A MC-compatible cell has an identical intersection pattern w.r.t both $\mathscr{E}$ and $\mathscr{A}$.*

Proof: An edge/face/voxel of a MC-compatible cell C is intersected by $\mathscr{E}$ if and only if it is intersected by $\mathscr{A}$. This is because the intersection status of an edge/face/voxel in a MC-compatible cell is solely determined by the signs at the grid vertices of the cell. For example, if an edge is intersected by $\mathscr{E}$, then the edge endpoints must exhibit a sign change because otherwise it is complex. Thus the signs at the endpoints of the edge capture its intersection status. This is also true in case of the faces and the voxel of cell C. Moreover, signs are preserved during surface reconstruction. This is due to Property 1. It follows that the intersection pattern is also preserved.

THEOREM 1 *If all the cells bounding $\mathscr{E}$ are MC-compatible, then the reconstructed surface, $\mathscr{A}$, has the same topology as $\mathscr{E}$.*

Proof: Lemma 3 and Property 2 show that $\mathscr{E}_C$ and $\mathscr{A}_C$ are topological disks inside cell C. Furthermore, Lemma 4 shows that the intersection pattern of $\mathscr{E}$ and $\mathscr{A}$ within each grid cell is identical. Using these two facts, we can define a homeomorphism between $\mathscr{E}$ and $\mathscr{A}$. This mapping can be extended to define a homeomorphism between the solids corresponding to $\mathscr{E}$ and $\mathscr{A}$. Consequently, $\mathscr{A}$ is topologically equivalent to $\mathscr{E}$.

4.3. Isosurface Extraction on Adaptive Grids

Applying Marching Cubes to an adaptive grid can result in cracks in the regions where grid cells at different resolutions meet. This becomes an issue when defining a homeomorphism between $\mathscr{E}$ and $\mathscr{A}$. We employ crack patching to overcome this problem. We perform crack patching by modifying the extracted isosurface within the larger cell to match the extracted surface within the smaller cell. In this way, we ensure that our approximation $\mathscr{A}$ is continuous. While this step may not preserve the star-shaped property within the grid cell, it maintains the property that $\mathscr{A}$ is a topological disk within the cell and preserves the intersection pattern. As a result, we can define a homeomorphism between $\mathscr{E}$ and $\mathscr{A}$.

We also considered dual methods such as Dual Contouring [JLSW02] for isosurface extraction. The advantage of this method is that it obviates the need for crack patching. However, we cannot apply Dual Contouring directly. This is because we require the contouring method to satisfy Property 1 in Section 4.2. Dual Contouring when applied to an adaptive grid may not satisfy this property. It is possible to extend the dual contouring method to overcome this problem. We plan to use this modified method for isosurface extraction in future.

In the rest of the paper, we use MC-compatibility as the conservative criterion for adaptive grid generation and computing a topology preserving isosurface.

5. Topology Preserving Subdivision

In this section, we provide adaptive subdivision criteria to generate a grid such that each grid cell is MC-compatible. Our sampling algorithm performs two tests on each grid cell. We first test check if a cell is complex using max-norm distance computation. Our second test checks if the surface is star-shaped using kernel computation. We first describe computational techniques for polyhedral models and their Boolean combinations and later extend them to higher order primitives. We also present the adaptive subdivision algorithm.

5.1. Max-Norm Distance Computation

We use max-norm distance computation to test if a primitive intersects the voxel, face, or edge of a cell. Under this norm, the distance between two points $\mathbf{x}$ and $\mathbf{y}$ (in 3 dimensions) is represented as $D_\infty(\mathbf{p},\mathbf{q})$ and is defined as

$$D_\infty(\mathbf{p},\mathbf{q}) = \max_i |p_i - q_i|, \qquad i = 1,2,\ldots,3 \qquad (1)$$

We can extend this definition in case of distance between a point $\mathbf{p}$ and a set $\mathscr{Q} \subseteq \mathbb{R}^d$. The iso-distance ball, i.e., the set of points at a constant distance from the origin, under max-norm is a cube; so it is a natural metric for cubical cells. We use the algorithm presented by Varadhan et al. [VKK*03] for max-norm distance computation. This algorithm is applicable to a wide variety of primitives such as polyhedra, algebraic primitives, etc.

When the isosurface $\mathscr{E}$ is defined by a Boolean expression involving a number of primitives, we estimate the max-norm distance to the isosurface by evaluating an equivalent expression involving min/max operators on the signed distances to the primitives. It is possible that the min/max operators can

produce incorrect distances at some points. However, this distance value will always be less than the exact distance to the surface boundary and is used for Boolean operations. We use max-norm distances to verify the complex cell criterion, which is presented in Section 5.3.1. For this test, these values are conservative and preserve correctness.

5.2. Kernel Computation

The exact kernel computation of a closed orientable surface primitive involves the intersection of halfspaces. These halfspaces are determined by the tangent plane at each point on the surface. Using the point-hyperplane duality, this reduces to a convex hull computation. In $\mathbb{R}^3$, for a polyhedral surface with n facets, this can be performed in $O(n \log n)$. However, for the star-shaped test, it suffices to check if the kernel is empty or not.

For the sake of simplicity, we first consider kernel computation for polyhedral primitives. We discuss extension to higher order primitives in Section 5.4. For a polyhedral primitive, testing for a non-empty kernel reduces to linear programming (LP). If $\mathbf{p}$ is a point belonging to the kernel, then each face of the polyhedron with centroid $\mathbf{c}$ and outward normal $\mathbf{n}$ defines the linear constraint $\mathbf{n} \cdot (\mathbf{c} - \mathbf{p}) > 0$ on $\mathbf{p}$. As a result, the kernel is non-empty if the set of constraints admits a feasible solution for $\mathbf{p}$. In fixed dimensions, LPs can be solved in linear time, and a number of efficient public domain implementations are available.

When the isosurface $\mathscr{E}$ is defined by a Boolean expression involving a number of primitives, a sufficient condition for the star-shapedness of the isosurface is that the intersection of all the primitive kernels is non-empty. If S_1 and S_2 are two star-shaped primitives with a common origin, then $S_1 \odot S_2$ is also star-shaped where $\odot$ denotes a Boolean operation such as union and intersection. This is because

$$Kernel(S_1) \cap Kernel(S_2) \subseteq Kernel(S_1 \odot S_2)$$

For polyhedral primitives, we check for the above condition by combining the linear constraints defined by the individual primitives and testing for feasibility by solving the resulting LP. The difference operation can be rewritten as an intersection by inverting the linear constraints of the negated primitive. We note here that the above condition is sufficient, but not necessary.

5.3. Adaptive Subdivision Algorithm

We start with a single grid cell that is guaranteed to bound $\mathscr{E}$. We perform two tests, *complex cell test* and *star-shaped test*, to decide whether to subdivide a grid cell. We now describe each test.

5.3.1. Complex Cell Test

To check whether a cell is complex, we perform the following tests:

- **Complex Voxel/Face**: We use max-norm distance computation to check whether the surface intersects a voxel or face of the cell. We use the fact that a voxel (face) is intersected by the surface if and only if its (unsigned) three- (two-) dimensional max-norm (l_∞) distance from the center of the voxel (face) is less than half the size of the cell. We determine if the voxel (face) is complex by checking for the sign change at its vertices and the result of the max-norm test. A voxel (face) is complex if it is intersected by the surface and does not exhibit a sign change.
- **Complex Edge**: We use directed distances [KBSS01] to test if an edge is complex. An edge is complex if the sum of the directed distances (along the edge) from the two endpoints of the edge is less than the edge length.
- **Ambiguity**: We use the signs at the grid vertices to resolve cases corresponding to face and voxel ambiguity (see Fig. 3(b) and Fig. 2(d)).

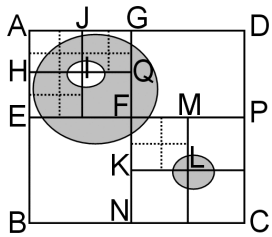


Figure 4: ***Adaptive Subdivision:*** *This figure is an illustration of our adaptive subdivision algorithm, The algorithm performs adaptive subdivision until the surface within each cell is a topological disk (in 3D). It ensures this by testing whether a cell is complex and if the surface is star-shaped with respect to the cell. In this figure, cell ABCD was subdivided because it corresponds to a complex voxel, cells AEFG and FNCP were subdivided because the curve within the cell was not star-shaped and FKLM was subdivided because of topological ambiguity. Edge IJ is complex; as a result, cells AHIJ and JIQG are subdivided.*

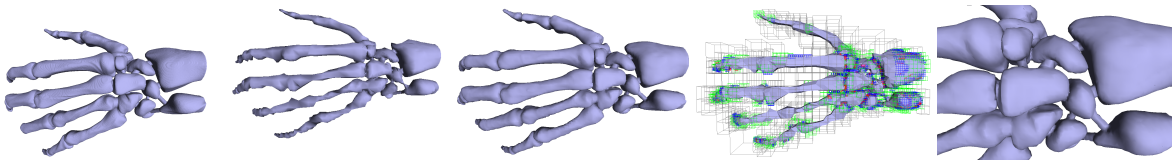
If any of these tests results in the affirmative, the cell is complex and we subdivide it and apply the algorithm recursively to the new cells.

5.3.2. Star-shaped Test

Linear programming (LP) is used to test for the star-shapedness of a polyhedral primitive. As described earlier in this section, the isosurface described by a single primitive or implicitly by a collection of primitives can be conservatively checked for the star-shaped property. We need to perform two tests at each cell on the isosurface – (a) star-shaped w.r.t voxel, and (b) star-shaped w.r.t. each face.

For each polyhedral primitive, we consider only those faces of the polyhedron that intersect the voxel. This set of faces defines the constraints for an LP in $\mathbb{R}^3$, whose solution answers the test (a). For each face of the cell, we consider those faces of the polyhedron that intersect it. These faces result in a collection of piecewise linear segments on the face of the cell. Solving a similar LP defined by these linear curves in $\mathbb{R}^2$ answers test (b). When there are multiple primitives intersecting the cell, we combine their linear constraints and then solve the resulting LP. Since we are only dealing with linear programs in two and three dimensions, a dual formulation of constraints and objective function is much more efficient in practice. We use such a formulation to perform the star-shaped test.

If either of these tests turns out to be negative, we subdivide the cell. Fig. 4 illustrates the working of the algorithm on an example.



(a) Hand Model (b) Coarse Approximation (c) Fine Approximation (d) Adaptive Subdivision (e) Closeup

Figure 5: ***Topology-Preserving Simplification****: Figs (a),(b),(c) show the original model along with a coarse and fine approximation generated using our topology preserving simplification algorithm. The original model has* 654*K triangles, while the two approximations consist of* 27*K and* 58*K triangles respectively at Hausdorff error bounds of* 1 *and* $1/128$*. Fig (d) shows the adaptive grid generated for the coarse approximation. The colors, green, blue, and red in that order, indicate the increasing level of subdivision. Fig (e) shows a closeup view of a part of the fine approximation. It highlights the features in the original model and our algorithm is able to reconstruct all these features accurately. It took* 36 *secs to generate the approximation.*

5.3.3. Termination

The subdivision algorithm will terminate if the isosurface is a closed manifold, free from artifacts such as self intersections and if there is no tangential contact between the primitives. The algorithm does not terminate when there are two primitives in tangential contact within a voxel, face or edge. It may be possible to use different spatial subdivision schemes (instead of octree) or perturbation techniques to overcome them. One possible approach is to subdivide using supporting planes that arise from the individual primitives themselves. This is akin to binary space partitioning (BSP) technique. Cells generated by this approach are no longer cubical, but general convex polyhedra. It is not difficult to prove that such a technique will always terminate. However, issues regarding isosurface extraction from such a grid and topological guarantees need to be explored further.

5.3.4. Hausdorff Error

Our subdivision algorithm can be used to generate bounded-error approximation to the exact isosurface $\mathscr{E}$. Given an error tolerance $\varepsilon > 0$, we can augment the subdivision algorithm with an additional criterion – that the diameter of all the grid cells be less than ε. Let $\mathscr{A}_\varepsilon$ denote the resulting approximation. This gives us an upper bound on the two-sided Hausdorff error between $\mathscr{E}$ and $\mathscr{A}_\varepsilon$.

5.4. Extension to Higher Order Primitives

The results presented in section 4 are general and are applicable to a wide variety of primitives. While techniques for max-norm computation presented in [VKK*03] are applicable to a wide variety of primitives, our method for kernel computation using linear programming was restricted to polygonal meshes. In this section, we will describe methods to extend this computation to nonlinear primitives. Exact methods for kernel computation of higher order primitives usually involves solving a system of nonlinear equations and curve tracing which can be computationally expensive. We present an approach that avoids exact kernel computation. As previously observed, the star-shaped test is reduced to testing whether the kernel is empty or not. Therefore, all we need is a point that is witness to the actual kernel. We provide a simple way to conservatively performs this test.

Our approach starts by performing a dense tessellation of the original primitive. This is done as a pre-processing step. We use the surface triangulation to determine an approximate kernel using linear programming. If such a kernel exists, we determine a point in it that is the best candidate to belong to the kernel of the exact curved primitive. As a candidate, we choose a point that is roughly in the center of the approximate kernel. To find this point, we augment the linear program by adding a slack variable δ to each of the constraints and set the objective function to maximize δ. A detailed explanation of this technique is provided in [VKSM04b].

This point is expected to lie inside the exact kernel (if it exists) with high probability. The probability depends on the approximation incurred in tessellation - the tighter the approximation, the higher the probability. However, we need to check if this point is actually a witness to the exact kernel. Such a witness $\mathbf{p}$ would satisfy $\mathbf{n}(\mathbf{x})^T(\mathbf{x}-\mathbf{p}) > 0$, for all points $\mathbf{x}$ on the primitive where $\mathbf{n}(\mathbf{x})$ is the normal to the surface at $\mathbf{x}$. For an algebraic surface $f(x,y,z) = 0$, the expression becomes $\bigtriangledown f^T(\mathbf{x}-\mathbf{p})$. We can derive a similar expression for parametric surfaces. Given such an expression and an axis-aligned cell, we need to verify if it is positive inside the cell. We use interval arithmetic to perform this test reliably on the interval determined by the cell. If the expression turns out to be positive inside the cell, $\mathbf{p}$ is a witness to the exact kernel and we stop subdivision. Otherwise, the cell is subdivided and the tests are repeated on each child cell. The performance of the interval arithmetic depends on the degree of the expression and the tightness of the interval used. Details of our technique to optimize the interval arithmetic step are presented in [VKSM04b].

The above approach is conservative and can result in some unnecessary subdivision. This is because interval arithmetic tests the above expression for $\mathbf{x}$ belonging to the entire interval rather than considering only points belonging to the algebraic surface. The main benefit of our method is that it is very similar in flavor to the test for polyhedral primitives and this makes it efficient. One requirement of this approach is the need for an explicit expression for the normal field of the surface. This is a reasonable assumption because such expressions are available for the class of surfaces representable in algebraic or parametric form.

6. Implementation & Applications

In this section, we describe the implementation of our algorithm and highlight four different applications of our adaptive subdivision algorithm. These are boundary evaluation

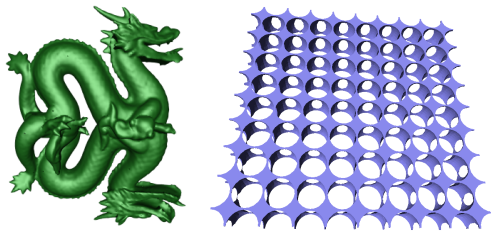


Figure 6: ***Boolean operations on complex models and curved primitives:*** *The left figure shows the result of the union of two dragons. Each dragon is represented using* 850*K triangles. Our algorithm computes a topology preserving approximation of the final boundary. It took* 95 *secs to compute an approximation with* 118*K triangles at a Hausdorff error bound of* $1/128$*. The right figure shows the result of* 100 *difference operations between a polyhedron and* 100 *ellipsoids. The resulting surface has a complex topology; it has a genus of* 208*. Our algorithm took* 16 *secs to generate an approximation with the correct topology at a Hausdorff error bound of* $1/64$*.*

on geometric primitives, motion planning, simplification, and remeshing.

We used C++ programming language with the GNU g++ compiler under Linux operating system. We used a freely available linear programming package, *QSOPT* (`http://www.isye.gatech.edu/~wcook/qsopt/index.html`), to implement the star-shaped test. We use a dual formulation of the linear program for computational efficiency. We demonstrate the performance of our algorithm on many complex models. Table 1 highlights the performance of our algorithm on these models. All timings are on a 2 GHz Pentium IV PC with a GeForce 4 graphics card and 1 GB RAM.

In all our applications, we first generate an adaptive grid on the primitives using our subdivision algorithm. We compute a polyhedral approximation to the boundary of the final solid using Extended Marching Cubes (EMC) algorithm [KBSS01] along with crack patching (as discussed in Section 4.3). The output surface in all our reconstructions is a manifold.

6.1. Boundary Evaluation

Boolean operations are frequently used to design complex solid models out of simple shapes. The solid objects are specified as constructive solid geometry (CSG) expressions and represented hierarchically using boolean operations on primitive solids. Our goal is to compute its boundary representation (B-rep). The topology of the resulting surface can be quite different from that of the individual primitives. As a result, it is important to accurately compute the boundary of the final solid with the correct topology. Analytic algorithms for exact boundary evaluation of CSG primitives are susceptible to robustness and accuracy problems. We assume that the B-rep is a manifold and limit ourselves to *regularized* boolean operations.

Figure 6 shows the union of two Dragon models. The second example is obtained by performing 5 difference operations on the Turbine Blade model (see rightmost image in Fig. 1). The model has more than 1.7 million triangles. The final surface has multiple components and a higher genus. Figure 6 highlights application of our algorithm to perform Boolean operations on curved primitives. It shows a challenging scenario where we perform 100 difference (Boolean) operations between a polyhedron and 100 ellipsoids. The resulting surface has a very complex topology with numerous small holes; it has a genus of 208.

Figure 7: *This figure shows the remeshing of a* Brake Hub *model. The center and right images show the triangulation of the original model and the remeshed model. The original brake hub model has* 14*K triangles. Our algorithm took* 1.85 *secs to perform remeshing and generate a mesh with* 7*K triangles at a Hausdorff error of* $1/32$*.*

Our algorithm generates an approximate surface that has the same topology as the exact boundary. Although volumetric approaches have been widely used for boolean operations [BMW00], earlier algorithms do not give any guarantees on the topology.

We have applied our algorithm to compute the Minkowski sum of polyhedral models and used it for exact motion planning with translational degrees of freedom. We consider the case of a 3D polyhedral robot undergoing translation motion among 3D polyhedral obstacles. This problem can be expressed in terms of the Minkowski sum of the robot and the obstacles. Minkowski sum of polyhedral models can be reduced to a union operation. However, exact computation of this union is not practical as it can have a complexity as high as $O(n^6)$. In [VKSM04a], we show how to use our topology preserving subdivision algorithm to compute an approximate Minkowski sum representation and use it for designing an exact path planner. Our planner is guaranteed to find a path, if one exists, even through narrow passages.

6.2. Topology Preserving Simplification

Given a polygonal object, model simplification algorithms produce a lower polygon count approximation that preserves its shape or appearance. Simplification techniques have been used for fast display and simulation. In order to compute a drastic simplification for interactive visualization, many algorithms do not preserve the topology. On the other hand, preserving the topology of the original object during simplification is important in some applications like CAD, collision detection, medical visualization and molecular modeling. Volumetric approaches have been proposed for model simplification [NT03]. These algorithms are quite fast in practice and are applicable to all kind of models. However, none of these approaches preserve the surface topology.

We use our algorithm to perform topology preserving volumetric simplification. We compute a discrete sampling of the distance field by applying our subdivision algorithm presented in Section 5. Different levels of detail are generated by changing the threshold value for 2-sided Hausdorff approximation. The reconstruction algorithm generates an isosurface that has the same topology as the original model. Our metric of Haus-

	Combinatorial Complexity		Performance (secs)			
Model	Input Tri Count	Output Tri Count	Complex Cell Test	Star-shaped Test	Subdivision	Total
Hand Simplification (Fig. 5)	654,666	58,966	4.3	8.1	23	36
Turbine Simplification (Fig. 1)	1,765,388	511,182	14.1	31.3	65.8	110
Turbine Blade Boolean (Fig. 1)	1,765,388	319,074	16.8	29.1	70.4	116
Union of Dragons (Fig. 6)	1,714,920	118,214	10.2	21.1	63.6	95
Curved Boolean (Fig. 6)	-	57,286	5.4	5.1	5.5	16
Brake Hub Remeshing (Fig. 7)	14,208	7,056	0.38	0.55	0.90	1.85

Table 1: *This table highlights the complexity of our input models and performance of our algorithm. The columns on the left shows the the triangle count of the input and triangle count in our reconstruction. The columns on the right show the cumulative time taken by the complex cell test, star-shaped test and adaptive subdivision over all the grid cells. The rightmost column shows the total execution time.*

dorff error can be easily combined with other metrics such as curvature, quadric error, etc, to guide the simplification. Figure 5 shows our simplification algorithm applied to a medical dataset, a 650*K* triangle *Hand* model. This model has a number of topological features that need to be preserved in order to maintain the anatomical structure of the hand. Figure 1 shows our simplification algorithm applied to the Turbine Blade model. Note that our method has preserved the complex topological features in the simplified model.

Some prior surface simplification methods can be adapted to perform topology preserving simplification [CVM*96, ZG02]. However, one limitation of these approaches is that they can produce self intersections. In order to avoid self-intersections, surface decimation algorithms need to perform global tests and this can result in considerable overhead. For example, simplification envelopes [CVM*96] can take $3-5$ hours on models composed of few hundred thousand triangles on a SGI R10000 processor.

Our subdivision criterion ensures that the isosurface is a topological disk within each grid cell by satisfying the complex cell and star-shaped criteria. In case of simplification, a simpler test exists. We can ensure the topological disk property by computing the Euler characteristic of the original polygonized surface within the grid cell and testing if it is equal to 1. However, this test is not applicable in case of Boolean operations where we do not have a polygonization of the final surface (in fact, our goal is to compute a polygonization). The test is also not applicable to motion planning.

6.3. Topology Preserving Remeshing

Volumetric approaches have been used for remeshing and repair of polygonal models [KBSS01, NT03]. In many applications, the polygonal models can have degenerate triangles, flipped normals, or triangles with bad-aspect ratios. The goal is to compute a valid manifold representation of the underlying closed solid and ensure that the resulting triangles have a good aspect ratio. The manifold representation generated after resampling can be used for multiresolution analysis or simplification. Earlier algorithms generate a volumetric representation by sampling the distance field on a uniform grid and extracting a surface with sharp features [KBSS01] or with a simplified topology [NT03]. However, these methods provide no guarantees on the genus or the number of connected components. We have applied our subdivision algorithm to compute a topology preserving remeshing of CAD models (see Fig. 7). The Euler characteristic test (see Section 6.2) could also be used for remeshing.

6.4. Performance

Table 1 highlights the performance of our algorithm on these models. It also provides a breakup of the total time spent in performing the complex cell test, star-shaped test, and adaptive subdivision. The complex cell and star-shaped tests are very efficient. On an average, the complex cell and star-shaped tests account for $10-15\%$ and $20-25\%$ of the total time respectively. The remainder of time is spent on adaptive subdivision. This correspond to the time taken to push the input triangles down the octree data structure. For each octree cell, we maintain a list of triangles that intersect it. As we subdivide the octree cell, we partition it into 8 children and compute their triangle lists. For large input models, this takes substantial fraction of the total time.

7. Limitations

In this section, we discuss some of the limitations of our algorithm. Our adaptive subdivision algorithm may not be able to handle all degenerate configurations in the input model. These include cases when the model has artifacts such as self intersections or when two primitives are touching tangentially. Our algorithm can only generate manifold boundaries and is not applicable to the cases where the exact boundary is non-manifold. Our adaptive subdivision criteria is conservative. Within a cell, we require all the primitives should be star-shaped with respect to a common origin. The isosurface defined by the Boolean expression over the primitives can be star-shaped within the cell even though this condition may not be satisfied. This can result in additional subdivision and lead to higher polygon counts in the approximation. Our topology preserving simplification algorithm cannot perform drastic simplifications. This is due to our conservative subdivision and also the fact that volumetric approaches can not produce drastic simplifications [ESV97]. Moreover, for a fixed polygon budget, approaches based on surface decimation opera-

tions like edge collapses or vertex removal [CVM*96] will generate a higher quality simplification.

8. Conclusions and Future Work

In this paper, we describe a novel approach to compute topology preserving isosurfaces that arise in a variety of geometric processing applications. We present a sufficient condition based on the *complex cell* and *star-shaped* criteria for sampling a distance field so that the reconstruction maintains the topology of the original isosurface. We also provide an adaptive subdivision algorithm which is very efficient and easy to implement. We have applied our subdivision criterion to boundary evaluation, motion planning, topology preserving simplification and remeshing on a number of complex examples successfully.

There are many avenues for future work. In applications such as laser scanning, the input data often contains topological noise due to inaccuracies in the scanning and merging process. We would like to investigate whether our results can be combined with the algorithms presented in [GW01, BK02] and used to perform topological reasoning for noise removal. Our current implementation supports polyhedral and low order algebraic primitives. We would like to apply our algorithm to the class of NURBS and subdivision surfaces. We would also like to use our subdivision algorithm to improve the accuracy of swept volume computation algorithms. Finally, we would like to improve our current subdivision criteria so that it is less conservative, and yet preserves topology.

9. Acknowledgments

We would like to thank David Applegate for providing us the QSOPT software and a modified dual LP formulation. This research is supported in part by ARO Contracts DAAD19-02-1-0390 and W911NF-04-1-0088, NSF awards ACI 9876914 and ACR-0118743, ONR Contracts N00014-01-1-0067 and N00014-01-1-0496, DARPA Contract N61339-04-C-0043 and Intel.

References

[BCSV04] BOISSONNAT J.-D., COHEN-STEINER D., VEGTER G.: Isotopic implicit surface meshing. *STOC* (2004). 2

[BK02] BISCHOFF S., KOBBELT L.: Isosurface reconstruction with topology control. *Proc. of Pacific Graphics* (2002), 246–255. 3, 10

[BMW00] BREEN D., MAUCH S., WHITAKER R.: 3d scan conversion of csg models into distance, closest-point and color volumes. *Proc. of Volume Graphics* (2000), 135–158. 8

[CGMS00] CIGNONI P., GANOVELLI F., MONTANI C., SCOPIGNO R.: Reconstruction of topologically correct and adaptive trilinear isosurfaces. *Computers & Graphics 24*, 3 (2000), 399–418. 2

[CVM*96] COHEN J., VARSHNEY A., MANOCHA D., TURK G., WEBER H., AGARWAL P., BROOKS F., WRIGHT W.: Simplification envelopes. In *Siggraph* (1996), pp. 119–128. 9, 10

[ESV97] EL-SANA J., VARSHNEY A.: Controlled simplification of genus for polygonal models. *Proc. of IEEE Visualization* (1997), 403–410. 9

[FPRJ00] FRISKEN S., PERRY R., ROCKWOOD A., JONES R.: Adaptively sampled distance fields: A general representation of shapes for computer graphics. In *SIGGRAPH* (2000). 3

[GW01] GUSKOV I., WOOD Z.: Topological noise removal. *Graphics Interface* (2001). 3, 10

[JLSW02] JU T., LOSASSO F., SCHAEFER S., WARREN J.: Dual contouring of hermite data. *SIGGRAPH*, (2002). 2, 5

[KBSS01] KOBBELT L., BOTSCH M., SCHWANECKE U., SEIDEL H. P.: Feature-sensitive surface extraction from volume data. In *Proc. of ACM SIGGRAPH* (2001), pp. 57–66. 2, 6, 8, 9

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH)* (1987). 1, 2, 3

[NT03] NOORUDDIN F. S., TURK G.: Simplification and repair of polygonal models using volumetric techniques. *IEEE Trans. on Visualization and Computer Graphics 9*, 2 (2003). 3, 8, 9

[SFYC96] SHEKHAR R., FAYYAD E., YAGEL R., CORNHILL F.: Octree-based decimation of marching cubes surfaces. *IEEE Visualization* (1996). 3

[SH97] STANDER B. T., HART J. C.: Guaranteeing the topology of an implicit surface polygonization for interactive modeling. In *Proc. of ACM SIGGRAPH* (1997), pp. 279–286. 2

[VKK*03] VARADHAN G., KRISHNAN S., KIM Y., DIGGAVI S., MANOCHA D.: Efficient max-norm computation and reliable voxelization. *Proc. of ACM SIGGRAPH/Eurographics Symposium on Geometry Processing* (2003), 116–126. 3, 5, 7

[VKSM04a] VARADHAN G., KRISHNAN S., SRIRAM T., MANOCHA D.: A simple algorithm for complete motion planning of translating polyhedral robots. *Workshop on the Algorithmic Foundations of Robotics* (2004). 8

[VKSM04b] VARADHAN G., KRISHNAN S., SRIRAM T., MANOCHA D.: *Topology Preserving Isosurface Extraction Using Adaptive Subdivision*. UNC Technical Report, 2004. 7

[WG90] WILHELMS J., GELDER A. V.: Topological considerations in isosurface generation extended abstract. *Computer Graphics 24*, 5 (1990), 79–86. 2, 4

[WHDS02] WOOD Z., HOPPE H., DESBRUN M., SCHRODER P.: *Iso-surface Topology Simplification*. Tech. rep., Microsoft Research, MSR-TR-2002-28, 2002. 3

[ZG02] ZELINKA S., GARLAND M.: Permission grids: Practical, error-bounded simplification. *ACM Trans. on Graphics* (2002). 9