Feature-Sensitive Subdivision and Isosurface Reconstruction

Gokul Varadhan*
University of North Carolina
at Chapel Hill

Shankar Krishnan†
AT&T Research Labs

Young J. Kim* Dinesh Manocha*
University of North Carolina
at Chapel Hill

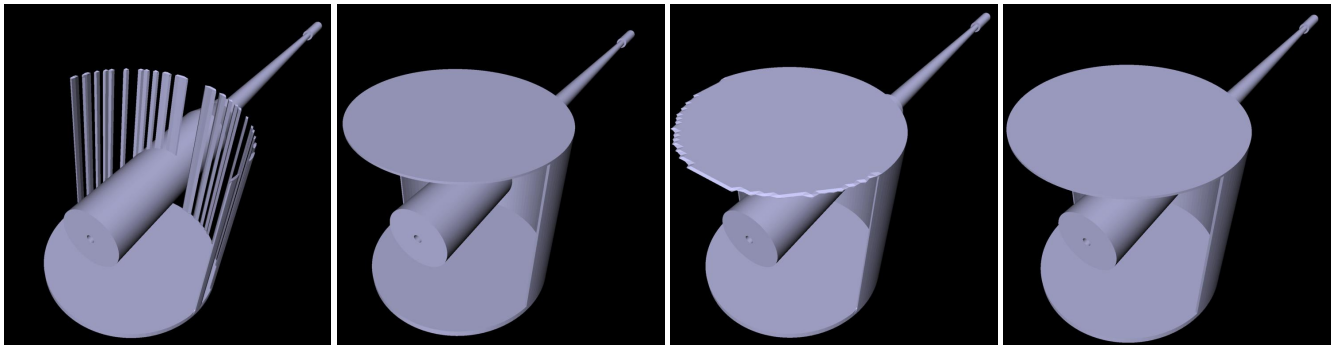


Figure 1: We present improved subdivision and isosurface reconstruction algorithms for polygonizing implicit surfaces and performing accurate geometric operations. We highlight its performance on the "gun model" of the Bradley Fighting Vehicle which is generated using 8 Boolean operations. The leftmost and center left images show an iso-surface reconstruction using the dual contouring algorithm [Ju et al. 2002] on a distance field sampled on a uniform $64 \times 64 \times 64 (\approx 262K)$ and $256 \times 256 \times 256 (\approx 16.7M)$ grid, respectively. The center right image shows the improved iso-surface generated from our new reconstruction on the uniform $64 \times 64 \times 64$ grid. It can reconstruct thin features without creating additional handles, but cannot reconstruct all the sharp features. We also present an algorithm to detect multiple sharp features in a cell and use it to generate an adaptive grid. The rightmost image shows our reconstruction applied to a grid generated by our subdivision algorithm. It can reconstruct all the sharp features, does not create any additional handles and uses only $313,168$ voxels.

Abstract

We present improved subdivision and isosurface reconstruction algorithms for polygonizing implicit surfaces and performing accurate geometric operations. Our improved reconstruction algorithm uses directed distance fields [Kobbelt et al. 2001] to detect multiple intersections along an edge, separates them into components and reconstructs an isosurface locally within each component using the dual contouring algorithm [Ju et al. 2002]. It can reconstruct thin features without creating handles and results in improved surface extraction from volumetric data.

Our subdivision algorithm takes into account sharp features that arise from intersecting surfaces or Boolean operations and generates an adaptive grid such that each voxel has at most one sharp feature. The subdivision algorithm is combined with our improved reconstruction algorithm to compute accurate polygonization of Boolean combinations or offsets of complex primitives that faithfully reconstruct the sharp features. We have applied these algorithms to polygonize complex CAD models designed using thousands of Boolean operations on curved primitives.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

Keywords: Implicit modeling, Boolean operations, Marching Cubes, Distance fields, Subdivision

∗ {varadhan,youngkim,dm}@cs.unc.edu
† krishnas@research.att.com
URL: http://gamma.cs.unc.edu/recons

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 3-dimensional scalar field. It can be mathematically expressed by a function $f(\mathbf{p}) = 0$ where $\mathbf{p}$ is a point. Moreover, $f(\mathbf{p})$ is often defined as the distance between $\mathbf{p}$ and the surface of the object being represented.

In this paper, we mainly deal with geometric models that are implicitly defined using a volumetric data set. Our main goal is to compute an accurate polygonization of the implicit surface and use these representations for geometric processing applications such as Boolean operations (i.e. union, intersection and difference) or offset computations. The resulting algorithms use the following two steps:

1. Generate a voxel grid and compute the signed distance field at its corner grid points.

2. Reconstruct the isosurface using some variant of the Marching Cubes algorithm.

Many issues arise when applying this approach to complex shapes and reliably generating the boundary of the final surface. The accuracy of the algorithm is mainly governed by the resolution of the underlying grid and the choice of the reconstruction algorithm. If the final surface has thin features, insufficient grid resolution can create handles in the reconstructed surface (see Fig. 1). Moreover, many geometric operations (e.g., Booleans) create new sharp features or edges on the boundary of the final surface. Our goal is to reconstruct them as faithfully as possible.

Recent work on implicit modeling techniques has addressed some of these problems. These techniques include generating adaptive grids based on octrees or using *adaptively sampled distance fields* (ADFs) [Frisken et al. 2000]. However, a key challenge is designing criteria for generating adaptive subdivision. Recently, two improved isosurface extraction algorithms have been proposed: *Extended Marching Cubes* [Kobbelt et al. 2001] and *dual contouring* [Ju et al. 2002]. Both algorithms use Hermite data and generate isosurfaces that contain sharp features. In practice, they work well when each cell contains no more than one sharp feature or *complex* edges (i.e. edges with more than one intersection with a surface). Our approach builds on the adaptive grid generation methods and the improved isosurface extraction algorithms.

Main Contributions: We present two new algorithms for accurate polygonization of implicit surfaces from volumetric data: *feature-sensitive adaptive subdivision* and *isosurface reconstruction*. Our reconstruction uses directed distances, i.e., distance along a direction [Kobbelt et al. 2001], to perform an exact edge-intersection test. This edge intersection test can reliably detect intersections of the edge with a surface. This test is combined with the dual contouring algorithm [Ju et al. 2002] to obtain an improved reconstruction algorithm, *Extended Dual Contouring*. It can reconstruct thin features and avoids creation of additional handles. The algorithm takes into account the characteristics of the grid and considers complex edges. It enumerates all the intersections along the edges, separates them into components and reconstructs the isosurface locally within each cell.

We present a novel subdivision algorithm that takes into account sharp features in the original primitives as well as new sharp features that are introduced by intersecting surfaces and Boolean operations. We analyze the problem of accurately reconstructing the sharp features and present a conservative test to check for multiple sharp features in a cell. This test is used as a subdivision criterion for ADF generation and geometry processing.

Our overall approach for polygonization uses two kinds of tests to generate a more accurate approximation of the final surface: an edge-intersection test for improved reconstruction of thin features without creation of additional handles, and a multiple sharp feature detection test for improved reconstruction of sharp features. We have applied our algorithms to compute the boundary of implicit surfaces and complex CAD models designed using Boolean operations and offsets. The underlying primitives consist of polyhedra, quadrics and tori; our benchmark model (Bradley Fighting Vehicle) is designed using thousands of Boolean operations. We also use the graphics rasterization hardware to accelerate the computation of distance fields. In practice, our algorithm is able to compute a good approximation of the final surface (as shown in Fig. 1).

New Results: Novel aspects of our work include:

1. An improved reconstruction algorithm based on dual contouring that can reconstruct thin features and takes complex edges into account.
2. An exact edge-intersection test based on directed distance.
3. A conservative technique for detecting multiple sharp features per cell that arise from intersecting surfaces or Boolean operations.

Organization: The rest of this paper is organized as follows. We give a brief survey of prior work in Section 2. We present our improved reconstruction algorithm to compute the isosurface in Section 3. We present a technique for detecting multiple sharp features in a cell and use it to perform adaptive subdivision in Section 4. We describe the implementation of our algorithms and highlight their performance on different benchmarks in Section 5. Finally, we analyze the performance of our algorithms and discuss some of their strengths and limitations in Section 6.

2 Prior Work and Preliminaries

In this section, we give a brief overview of prior work on generating discrete samples and reconstruction algorithms.

2.1 Sample Generation

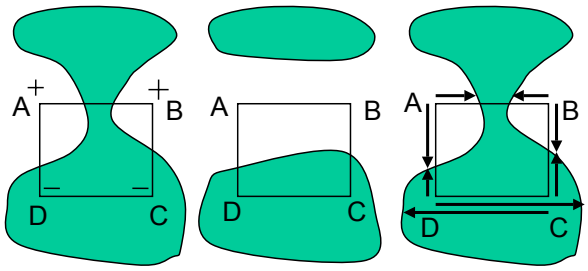
We mainly deal with generating distance samples of analytic functions or geometric models. Many efficient algorithms are known to compute the distance fields and their gradients at any point in space. A good overview of these algorithms has been given in [Cuisenaire 1999].

A key issue in generating discrete samples is the underlying sampling rate. Some of the common algorithms use an adaptive refinement strategy based on an octree, and only split those cells that contain a piece of the final surface in a top-down manner. Other techniques have used curvature information in generating the distance samples [Shekhar et al. 1996; Gibson 1998]. Moreover, [Frisken et al. 2000; Perry and Frisken 2001] have presented bottom-up and top-down methods for generating ADFs based on piecewise trilinear interpolation. Although, these algorithms optimize the sparsity of the octree representation, the approximation using a tri-linear interpolation may not work well for curved primitives or when the final surface has a lot of sharp features. [Huang et al. 2001] have proposed a complete distance field representation (CDFR) to capture sharp features of an object.

2.2 Isosurface Extraction and Reconstruction

Given a volume grid, some of the most common techniques for isosurface reconstruction are based on Marching Cubes and its variants [Lorensen and Cline 1987]. These algorithms assume that the surface samples are computed by an approximate intersection of the edges of a cell with the underlying surface. The Marching Cubes algorithm can produce too many small and badly-shaped triangles which typically require improving the mesh with decimation. Many authors have proposed using adaptive hierarchies to extract isosurfaces that will have fewer triangles [Perry and Frisken 2001; Shekhar et al. 1996].

The Marching Cubes algorithm is unable to extract high quality triangle meshes with sharp features from the volumetric data. Two kinds of algorithms have been proposed for accurate polygonization of implicit surfaces with sharp features. The first set of algorithms are based on an improved Marching Cubes based reconstruction. Kobbelt et al. [2001] proposed an enhanced distance field representation and an Extended Marching Cubes algorithm to perform feature sensitive sampling and reduce the aliasing artifacts in the reconstructed model. This algorithm explicitly identifies and processes sharp features. Ju et al. [2002] presented a Dual Contouring method for Hermite data which avoids the explicit identification and processing. Both of these algorithms work well as long as the grid contains at most one sharp feature. However, no good algorithms are known for generating a grid with atmost one sharp feature per cell. The second class of algorithms for sharp features



(a) (b) (c)

Figure 2: Fig (a) shows a surface (shown in green) intersecting a cell. The symbols, + & − represent positive and negative signs of distance at the grid points where the grid point outside the surface is defined as having positive distance and vice versa. Fig (b) shows reconstruction using dual contouring algorithm. Note that the reconstructed surface has disjoint components. Fig (c) shows how the directed distance can be used as a reliable edge intersection test. The surface intersects edge AB if and only if $|D_{\vec{AB}}(A)| < D(A,B)$

improve the output of Marching Cubes based on optimization techniques and smoothing operations, as a post-processing step [Ohtake et al. 2001; Ohtake and Belyaev 2003].

Often the surfaces reconstructed from a volume data using the Marching Cubes algorithm have a higher genus than they should have. Wood et al. [2002] and Bischoff and Kobbelt [2002] have presented algorithms on isosurface topology simplification and isosurface reconstruction with topology control. These algorithms are mainly targeted towards volume data generated from physical scanning devices.

3 Extended Dual Contouring

In this section, we present our improved reconstruction algorithm, Extended Dual Contouring. Given a volumetric grid with atmost one sharp feature per cell, we apply this reconstruction algorithm locally to each grid cell. The algorithm combines an exact edge-intersection test based on directed distances with the dual contouring algorithm.

We start with a brief description of the dual contouring algorithm [Ju et al. 2002]. The dual contouring method operates on a uniform grid in two steps. First, for each cell that exhibits a sign change across the edges, this method examines the set of intersection points and generates a vertex (per cell) such that a quadratic error function is minimized. We refer to this vertex as the *error-minimizing* vertex. Second, for each edge that exhibits a sign change, the contouring method generates a quad connecting the error-minimizing vertices of the four cells sharing the edge. The dual contouring algorithm detects whether the surface intersects the edge based on sign change across the edge. However, when the surface has thin features, this may not be a reliable intersection test. When a surface intersects an edge an even number of times, the edge will not exhibit a sign change. As a result, the reconstructed surface can have additional handles or disjoint components (as shown in Fig. 2).

In this case, both the endpoints A and B have the same sign, and the dual contouring algorithm considers the edge as non-intersecting. We present an exact edge-intersection test that can reliably detect edge intersection. We use this test to design an improved reconstruction algorithm.

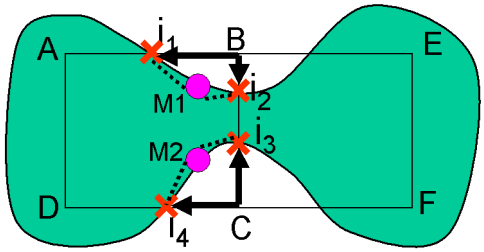


Figure 3: The primitive shown in green intersects cell ABCD such that the edge BC is complex. The intersection points (shown by red cross) $\mathbf{i}_1, \mathbf{i}_2, \mathbf{i}_3, \mathbf{i}_4$ are enumerated and divided into a set of components, $\{\mathbf{i_1}, \mathbf{i_2}\}$ & $\{\mathbf{i_3}, \mathbf{i4}\}$. Error-minimizing vertices (M1 & M2 shown by pink circles) are computed for each component independently.

3.1 Edge Intersection Test

We use directed distances to reliably detect whether the surface intersects an edge. Let the directed distance between a point $\mathbf{p}$ and a surface along a unit vector $\mathbf{v}$ be denoted by $D_{\vec{v}}(\mathbf{p})$. Our test is based on the following property: If an edge AB intersects the surface, then the directed distance at A along the direction vector, $\vec{AB}$, will be strictly less than the Euclidean distance, $D(A,B)$. Based on this fact, we define an edge AB to be intersecting if:

$$|D_{\vec{AB}}(A)| \quad < \quad D(A,B)$$

This is shown in Fig. 2. We define an edge to be *complex* if it is intersecting, but does not exhibit a sign change. The surface intersects a complex edge more than once. Complex edges typically arise when a cell contains a thin feature. The edges, AB, BC, DA are classified as intersecting based on our edge intersection test. No sign change occurs across the edge AB, a complex edge.

We use directed distance not only to detect edge intersections but also for computing intersection points. The directed distance at the two endpoints provides information for two intersection points along that edge.

3.2 Reconstruction from Complex Edges

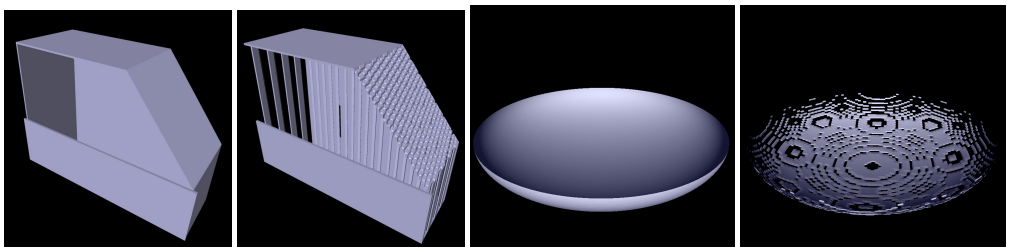
A direct application of the dual contouring algorithm to a grid with complex edges can lead to additional handles in the reconstruction due to the presence of complex edges (see Fig. 1 and Fig. 2(b)). In this section, we present our contouring algorithm that takes complex edges into account. We enumerate all the cell intersection points, separate them into components, and generate an error-minimizing vertex for each component independently.

3.2.1 Intersection Points

We initially consider edges that exhibit a sign change and complex edges that have positive signs at both the endpoints. For each edge of a cell, the directed distances at the two endpoints provide us information for two intersection points. For edges that exhibit a sign change we consider one intersection point for that edge, while for complex edges two intersection points are considered. We enumerate all the intersection points for each edge of the cell. Consider cell $ABCD$ in Fig. 3. Edges AB & CD each have one intersection point. Edge BC is complex and has two intersection points.

3.2.2 Component Separation

The set of intersection points enumerated above may belong to more than one surface component. A component refers to a sheet of the surface within the cell. For example, cell $ABCD$ in Fig. 3



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

Figure 4: Figs (a) & (c) shows reconstruction using extended dual contouring method applied to different benchmarks with thin features on an adaptive grid. Figs (b) & (d) show reconstruction of the models using dual contouring on the same grid. Note that the reconstruction produced by dual contouring has many topological artifacts.

has two components. Next, we present a technique to separate the set of intersection points into components. Given a grid cell, let $\mathbf{I}$ be the set of intersection points. Each intersection point $\mathbf{i} \in \mathbf{I}$, is associated with a unique grid point with a positive sign. We refer to this grid point as the *parent* of $\mathbf{i}$, denoted as $\mathscr{P}(\mathbf{i})$. See Fig. 3. We have $\mathbf{I}= \{\mathbf{i_1},\mathbf{i_2},\mathbf{i_3},\mathbf{i_4}\}$, $\mathscr{P}\{\mathbf{i_1},\mathbf{i_2}\} = B$, $\mathscr{P}\{\mathbf{i_3},\mathbf{i_4}\} = C$. Given two intersection points, $\mathbf{i}$, $\mathbf{j} \in \mathbf{I}$, we define the following equivalence relation:

$\mathbf{i} \rightarrow \mathbf{j}$ if and only if there is a path between $\mathscr{P}(\mathbf{i})$ and $\mathscr{P}(\mathbf{j})$ consisting of only non-intersecting edges in the cell.

The equivalence class defined by this relation induces a partition of $\mathbf{I}$ into separate components. In the case of cell *ABCD*, we have $\mathbf{i_1} \rightarrow \mathbf{i_2}$ and $\mathbf{i_3} \rightarrow \mathbf{i_4}$. There is no non-intersecting path between B and C. As a result, $\mathbf{I}$ gets divided into two components: $\{\mathbf{i_1},\mathbf{i_2}\}$ & $\{\mathbf{i_3},\mathbf{i_4}\}$. It is important to note that the components that result from the above method are accurate only because the edge intersection test is reliable. We compute an error-minimizing vertex for each component separately. Given an intersection point $\mathbf{p}$ along an edge of the cell C, let $M_C(\mathbf{p})$ be its error-minimizing vertex within the cell C.

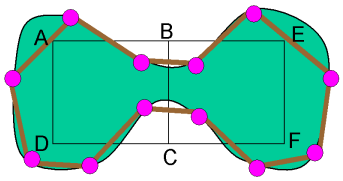


Figure 5: Contouring: Within each cell, an error-minimizing vertex (shown by pink circles) is computed for each component independently. The reconstructed surface (shown in brown) is obtained by connecting the error-minimizing vertices.

3.2.3 Contouring algorithm

The resulting contouring algorithm is very similar to the dual contouring algorithm, but it must consider that an edge can have up to two intersection points and that there can be multiple error-minimizing vertices per cell. Our overall contouring algorithm proceeds in the following manner:

1. for each cell, separate the intersection points into components and generate an error-minimizing vertex for each component independently.
2. for each intersecting edge,
   - for each intersection point $\mathbf{p}$ on the edge
     - for each cell C_i, $i = 1,\ldots,4$, that shares the intersecting edge, select $M_{C_i}(\mathbf{p})$ as the error-minimizing vertex
     - Generate a quad by connecting the selected vertices

Fig. 5 is a 2D illustration of our contouring algorithm. This contouring algorithm also extends to adaptive grids, based on the approach presented in [Ju et al. 2002].

The grid can also have complex edges with negative signs at both the endpoints. Our component separation technique easily extends to handle them. We define the equivalence relation in terms of grid points with a negative sign. The partition induced by this relation separates the intersection points into components. The rest of the algorithm is similar to that of complex edges with positive signs at both the endpoints.

The intuition behind the equivalence relation is that if we walk across the edges of the cell without any intersections, we are always on the exterior (or interior) of the surface. We treat this as a hint to the local topology inside the cell. Figs 1 & 4 show reconstruction using the extended dual contouring algorithm applied to different benchmarks and compare it with dual contouring. The reconstruction produced by dual contouring algorithm has many topological artifacts.

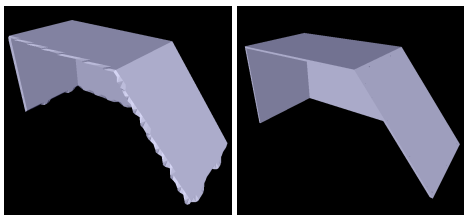
Our reconstruction algorithm requires that there is atmost one sharp feature per grid cell. Next, we present a subdivision algorithm that generates a grid satisfying this requirement.

4 Feature-Sensitive Subdivision

Many geometric processing applications like surface intersection and Boolean operations result in *sharp features* on the boundary of the surface. In the context of this paper, points of non-differentiability (either along a curve or isolated surface points) constitutes a sharp feature. Examples include vertices and edges of polyhedra, or the intersection curve of two smooth surfaces. When we generate the voxel grid for these geometric models, some of the voxels can have more than one sharp feature. In order to use our reconstruction algorithm, we present a new algorithm to detect multiple sharp features per cell.

Sharp features can arise in a number of ways: from polyhedral primitives, as a result of performing geometric operations involving intersecting surfaces or Boolean operations, or due to singularities such as self-intersections. In case of polyhedron, we detect sharp features by tracking vertices and edges of the polyhedron. The problem of computing the intersection curve between two surfaces has been well studied. However, current algorithms are unable to robustly evaluate the intersections or handle degenerate cases for general models [Hoffmann 2001]. As a result, our goal is to develop good approximation algorithms.

In this section, we address the problem of detecting sharp features that are created as a result of geometric operations involving



(a) (b)

Figure 6: Fig (a) shows our reconstruction algorithm applied to a $32 \times 32 \times 32 (\approx 32K)$ uniform grid. Some of the cells in the grid have multiple sharp features. As a result, the reconstruction suffers from aliasing. Fig (b) shows our reconstruction applied to an adaptive grid generated by our adaptive subdivision algorithm that uses $37,456$ voxels. It was able to reconstruct sharp features.

surface intersection between primitives. We also present an approximate technique to compute the number of sharp features per cell and use it for performing adaptive subdivision.

4.1 Sharp Features from Intersecting Surfaces

Consider the case when we are trying to compute the union of two objects S_1 and S_2. Let the continuous signed distance field corresponding to the two objects be $d_1(\mathbf{x})$ and $d_2(\mathbf{x})$, respectively. The distance field, $d(\mathbf{x})$, for $S_1 \cup S_2$ is given by

$$d(\mathbf{x}) = \min(d_1(\mathbf{x}), d_2(\mathbf{x})).$$

This function can also be written as $\mathscr{H}(d_1, d_2)$ (refer to Fig. 7), where

$$\mathscr{H}(p,q) = \frac{(p+q)}{2} + \frac{(q-p)}{2}\,\mathrm{sgn}(p-q). \qquad (1)$$

The function $(\mathrm{sgn}(.) : \mathbb{R} \rightarrow \{-1, 0, 1\})$ is the standard *sign* function. All Boolean operations can be expressed using linear combinations of the operand distance fields along with the $\mathrm{sgn}(.)$ function. From the point of view of signal processing, the presence of the $\mathrm{sgn}(.)$ function in the above expression can result in sharp features during Boolean operations. It is known that this function is not bandlimited. As a result, even if the primitive distance fields are bandlimited signals, a single Boolean operation can destroy this property.

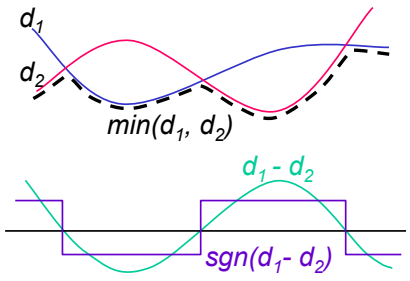


Figure 7: This figure shows a union operation between two functions, d_1 and d_2, and their union expressed as $\min(d_1, d_2)$. The sharp features in the union occur where the function $(d_1\text{-}d_2)$ changes sign i.e., at the zero crossing of the sign function $\mathrm{sgn}(.)$ We use this as a test to detect sharp feature.

We use this formulation to provide a test that detects presence of sharp features within a cell. Since the $\mathrm{sgn}(.)$ function changes value at the zero crossing of its argument, the sharp features of the original operation can be characterized by tracking the zeros of its argument. Let us consider the union operation described above. For distance fields (defined by a metric), d_1 and d_2, we must track the zero-level surface of the function $(d_1 - d_2)$ (called the *bisector* surface) (see Fig. 7 for a one-dimensional example). An alternate geometric justification for this exists. Given two smooth primitives, the sharp features produced as a result of a Boolean operation occur only where the surfaces intersect. The intersection curve between the two solids is strictly contained in the zero set of $(d_1 - d_2)$, where d_1 and d_2 are the distance fields of the two solids.

A simple test to detect the presence of a sharp feature is to track the presence of the bisector surface using its signed values at the grid points. However, this test can be conservative, and even the absence of sharp features can lead to unnecessary subdivisions.

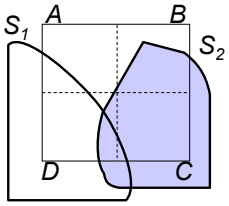


Figure 8: Illustration of our subdivision algorithm in presence of multiple sharp features. Surfaces S_1 and S_2 intersect inside the cell $ABCD$ where S_2 already has a sharp feature. Our algorithm subdivides the cell until it has no more than one sharp feature

4.2 Detecting Multiple Sharp Features

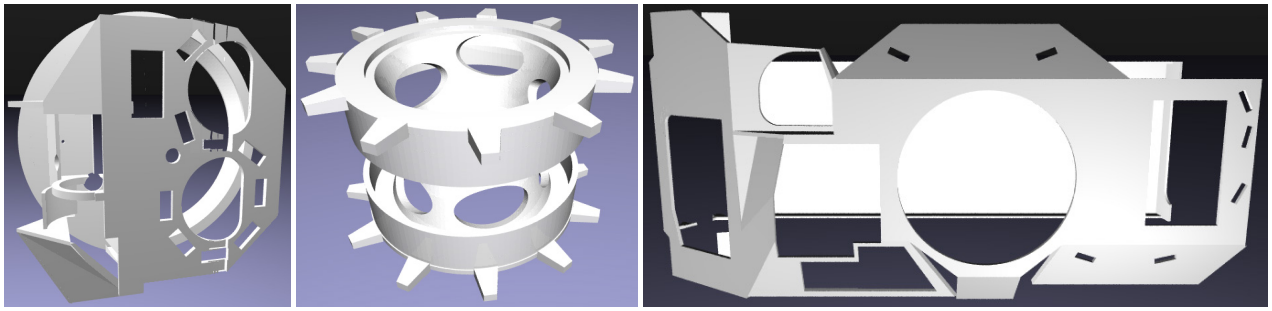
We present an algorithm to estimate whether a cell has more than one sharp feature. We further subdivide a cell based on the outcome of this test.

We assume that at most one new sharp feature is created due to intersecting surfaces or a Boolean operation between a pair of primitives inside a cell. Whenever such an operation is performed between two objects, the number of sharp features inside a cell resulting from the operation is bounded by the number of sharp features of the individual objects (prior to the operation) inside the cell and any extra features created as a result of the intersection between them. We use this property in our algorithm. The main advantages of this approach lie in its simplicity, and its natural extension to a nested sequence of geometric or Boolean operations, say all within the same cell. Let us denote the number of sharp features of an object inside a cell using the $\#(.)$ operator. If $\odot$ is the geometric or Boolean operation between two objects, S_1 and S_2, then we define it as

where

$$\#(S_1 \odot S_2) = \#(S_1) + \#(S_2) + \mathscr{I}(S_1, S_2),$$
$$\mathscr{I}(S_1, S_2) = \begin{cases} 1 & \text{if } S_1 \text{ and } S_2 \text{ intersect,} \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

Since the definition of the $\#()$ function is recursive, we must compute the sharp feature counts of all the leaf-level primitives. This computation is not difficult when dealing with linear and quadric primitives. As we mentioned earlier, in case of polyhedron, we detect sharp features by tracking vertices and edges of the polyhedron. We assume that smooth primitives such as ellipsoids and tori do not have any sharp features. A cylinder has two sharp features corresponding to the edges of its bottom and top caps. However, the main issue is how well we can compute $\mathscr{I}(.,.)$ inside a cell. We choose the *bisector surface* test to determine $\mathscr{I}(.,.)$. Even though this test can be conservative, we perform cell subdivision only if $\#()$ for the cell is greater than 1. This process is illustrated in Fig. 8. Figs. 1 and 6 show the benefit of our subdivision algorithm.



(a) Turret (b) Drivewheel (c) Hull

Figure 9: Benchmarks: Fig. (a) Turret is composed of 41 solids, where each solid defined using $2-19$ Boolean operations. Fig. (b) Drivewheel composed of 30 solids, each defined using $2-7$ Boolean operations. Fig. (c) Hull in the Bradley model composed of 36 solids, each defined using $2-12$ Boolean operations.

Figure 10: Bradley Model: This figure shows a view of the Bradley Fighting Vehicle. This model consists of $1,296$ solids and each solid is defined using $2-20$ CSG operations, for a total of $8,456$ CSG operations for the entire model. It took about 3.5 hours, including subdivision, distance field computation and reconstruction, to generate the approximate boundary on a 2 GHz Pentium 4 PC.

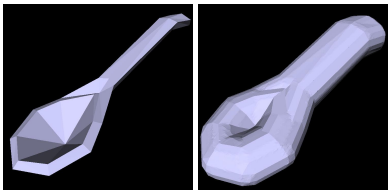


Figure 11: Offset: The left image shows the spoon model and the right image is its offset surface. We decompose the spoon model into 25 convex polytopes, and reduced the problem to computing union of 25 pairwise Minkowski sums. It took about 13 secs to compute the boundary including distance field computation, adaptive subdivision and reconstruction.

5 Implementation and Performance

In this section, we describe the implementation of our subdivision and reconstruction algorithms and highlight their performance on different applications.

5.1 Implementation

We used C++ programming language with the GNU g++ compiler under Linux operating system. For the choice of GUI implementation, GLUT and OpenGL were used. We represented all our primitives in an implicit form and computed distances to the primitives in a lazy manner. We used two types of distances: Euclidean distance to perform the sharp feature test during subdivision and directed distance for reconstruction. As part of the edge intersection test, we perform an inequality test on distances. The correctness of this test depends on the precision of distance computation. To test whether the edge intersects the surface, we first test for a sign change across the edge endpoints and apply our edge intersection test only if no sign change occurs.

5.2 Applications

We used three different applications to test the performance of our algorithms. These include boundary evaluation of complex CAD models, offset computations and polygonization of general implicit models. In case of boundary evaluation of CAD models and offset computation, the problem reduces to performing many Boolean operations on the primitives. Even though the problem of Boolean operations and boundary evaluation has been extensively studied in solid modeling, no good algorithms are known for efficient and robust computation, especially for curved primitives [Hoffmann 2001]. Some recent algorithms based on exact computation [Keyser et al. 2002] can produce accurate results. However, they cannot handle degenerate configurations that arise in real-world applications. We used a number of benchmarks to test the performance of our algorithms and to compute an approximation to the final boundary.

Boundary Evaluation of Bradley Fighting Vehicle: We used the model of a Bradley Fighting Vehicle defined using Boolean operations. It consists of 1,296 solids and each solid is defined using $2-20$ Boolean operations. The total number of Boolean operations is $8,456$ and the primitives consist of polyhedra, quadrics and tori. A view of the Bradley Fighting Vehicle is shown in Fig. 10. Some of the solids are shown in Fig. 9.

Offset Computation: The offset of a surface is defined by taking a fixed offset along the normal direction at each point and computing the envelope of the resulting set of points. We formulate it as the Minkowski sum with a sphere. Given a primitive P and a sphere S of radius r centered at the origin, the set of points in the offset surface are given as: $P \oplus S = \{p+s |\ p \in P, s \in S\}$. No good algorithms are known for exact computation for any arbitrary primitive, P. One approach for approximating the offset surface is based on computing a convex decomposition of P and using the *decomposition property* of Minkowski sums. Computing the Minkowski sum of a convex polytope and a sphere is relatively simple. If P is decomposed into

convex pieces, there can be $O(n)$ pairwise Minkowski sums and the problem reduces to computing their union. We applied our adaptive subdivision and reconstruction framework to approximate a boundary of the union of convex primitives. We used a model of a spoon (shown in Fig. 11) decomposed it into 25 convex polytopes and computed the boundary using 25 union operations.

Polygonization of General Implicit Models: We used our reconstruction algorithm to polygonize the surface of general implicit models. In Fig. 12, we show the image of a heart-shaped implicit model polygonized using extended dual contouring. This model is represented mathematically by a sixth order polynomial, $(2x^2+y^2+z^2-1)^3-(0.1x^2+y^2)z^3=0$.

Figure 12: General Implicit Model: The image shows a heart-shaped implicit model polygonized using extended dual contouring. This model is represented as a sixth order implicit polynomial function.

5.3 Performance

We have applied our algorithms to perform 8,456 Boolean operations in 1,296 solids in the Bradley model, offset computations and polygonization of implicit models. In our implementation, more than 90% of the time is spent in distance computation; applying the multiple sharp features subdivision criteria and computing the isosurface using extended dual contouring algorithm takes the remaining time. On average, it took about 12-15 secs to compute an approximation to each solid in the Bradley model of which 10-12 secs were spent in distance computation. The remainder of the time was spent on adaptive grid generation and iso-surface reconstruction. We accelerated distance computation by computing a distance field (directed as well as Euclidean) at a minimum resolution of $64\times64\times64$ using a hardware-based approach similar to [Hoff et al. 1999] and adaptively computing additional distance values in software. The hardware generated distance values are obtained by rendering a polygonal approximation of distance functions. The average time to generate a grid for each solid using our multiple sharp features test varied from 0.7-0.9 secs. The average time to compute a boundary representation of each solid using the extended dual contouring algorithm was about 1 sec. Table 1 shows timings for the extended dual contouring algorithm applied to different benchmarks.

Model	Dual Contouring		Ext Dual Contouring	
	$N=64$ (s)	$N=128$ (s)	$N=64$ (s)	$N=128$ (s)
Gun (Fig 1)	0.61	2.14	1.14	2.57
Turret (Fig. 9)	0.72	2.55	0.89	2.72
Drivewheel (Fig. 9)	0.85	3.32	1.03	3.56
Hull (Fig. 9)	0.81	3.12	0.95	3.41

Table 1: Performance: This table compares the performance of extended dual contouring algorithm with that of dual contouring on different benchmarks at different grid resolutions ($N\times N\times N$)

We represent the adaptive grid using an octree. For all our benchmarks, we generated an adaptive grid with a minimum resolution of 64 in each dimension and subdivide it in an adaptive manner, such that the maximum resolution in each dimension is bounded by 512. Fig. 13 highlights the performance of our adaptive subdivision algorithm on different benchmarks, showing the level of subdivision.

5.4 Comparison with Previous Approaches

Figs. 1 & 4 show a comparison between the extended dual contouring and dual contouring. Our algorithm produces better reconstructions with fewer levels of subdivision. Table 1 compares the performance of the extended dual contouring algorithm with that of the dual contouring. On an average, it is 10-20% slower compared to dual contouring algorithm. Our algorithm behaves differently from dual contouring only in the cells with complex edges. Because the gun model has many complex edges at a coarse resolution of $64\times64\times64$, our algorithm takes more time.

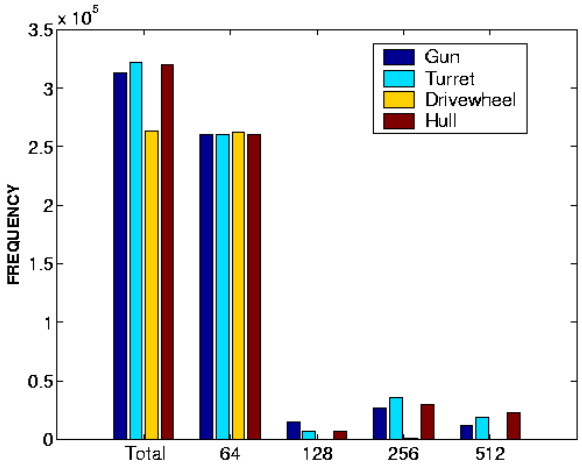


Figure 13: This figure shows the total number of voxels in our adaptive grid for different benchmarks (shown in Fig. 1 and Fig. 9. It highlights the number of voxels generated by our adaptive subdivision algorithm at different resolutions. Note that a very small fraction of the voxels at $64\times64\times64$ resolution are further subdivided by our algorithm.

We have presented an algorithm for generating an adaptive grid that takes into account the number of sharp features per cell. To the best of our knowledge, none of the previous approaches for grid generation took this into account; consequently, we cannot directly compare those approaches to ours. Previous algorithms [Frisken et al. 2000; Perry and Frisken 2001] based on tri-linear interpolation may not work well for curved primitives or when the final surface contains many sharp features. Comparing the performance of our grid generation algorithm with these algorithms proves difficult because distance computation time depends highly on the primitives and the number of Boolean operations per solid. Our benchmarks include curved objects such as quadrics and tori and our solids are generated by performing 2-20 Boolean operations. Moreover, our implementation can be sped up by cache coherence techniques presented in [Perry and Frisken 2001].

Optimization based algorithms [Ohtake et al. 2001; Ohtake and Belyaev 2003] can recover sharp features in many cases, though their applicability to complex models defined using hundreds of Boolean operations and whose final boundary consists of multiple sharp features close to each other is not clear. [Huang et al. 2001] have used a *Complete Distance Field Representation* (CDFR) to capture sharp features of an object. However, it is unclear if the CDFR representation can be used to consistently capture sharp features generated from geometric operations such as Boolean operations.

6 Discussion

In this section, we analyze our algorithms and discuss some of its strengths and limitations. The techniques presented in this paper

address the problem of sampling and reconstruction using distance fields. We address two main issues with respect to generating accurate reconstructions. These are sharp features and additional handles that can arise in Marching Cubes based reconstruction algorithms.

Main Benefits: One of the main challenges with Marching Cubes-based algorithms is accurate reconstruction of all the sharp features. Fundamentally, the problem of computing sharp features based on discrete sampling can be ill-posed. In practice, this imposes a limit on the performance of any algorithm based on a subdivision and reconstruction framework. However, the use of sharp feature test gives us certain properties which can generate better reconstructions. Overall, our algorithm for subdivision and reconstructing surfaces with sharp features is more general and less restrictive than earlier techniques like Extended Marching Cubes or Dual Contouring, which assume at most one sharp feature per cell. The method works quite well on our benchmarks.

Our reconstruction algorithm detects surface features that cannot be found purely based on sign changes along an edge. One alternative to our subdivision and reconstruction approach would have been to subdivide the grid adaptively until there are no complex edges and use dual contouring for reconstruction. However, such a subdivision algorithm can be very conservative and can result in a very high number of cells. One of our major goals is to use fewer grid cells, if possible.

Limitations: Our reconstruction algorithm uses information along the edges of the cell to make a guess about the local topology within the cell. This is not a fool-proof test; in some cases, the topology within the cell can be different. Thus, although it produces better results as compared to earlier algorithms, it is not completely immune from topological inconsistencies. It does not handle edges with more than two intersection points. The algorithm doesn't handle cases where a surface lies completely within a cell or passes through a face of the cell without intersecting any edges. To detect such cases, [Varadhan et al. 2003] have presented an efficient *voxel-intersection test* based on max-norm computation to reliably test whether the surface intersects a voxel. Furthermore, our assumption of two primitives intersecting in only one sharp feature per cell may not hold in some configurations of the primitives. In such cases, our approach may not reconstruct all the sharp features.

7 Conclusion and Future Work

We have presented a novel reconstruction algorithm for extracting isosurfaces from volume data. Our reconstruction algorithm can reconstruct thin features without creation of additional handles. We presented an exact edge-intersection test which can reliably detect intersections of the edge with a surface. We presented new techniques to handle problems related to multiple sharp features. We have applied the results to generate an approximate boundary of a Bradley Fighting Vehicle described using more than 8000 Boolean operations on curved primitives.

Many avenues for future work lie ahead. We have applied our approach to reconstruct the boundaries generated from Boolean operations and implicit functions defined using polynomials. We would like to apply it to inputs defined using blends and warping. We would like to incorporate the voxel-intersection test [Varadhan et al. 2003] in our adaptive subdivision algorithm. We would also like to extend our sharp feature computation algorithm to handle primitives that can intersect in more than one sharp feature per cell. We could use more efficient algorithms for distance field computation [Sud and Manocha 2003]. Finally, we would like to extend our approach to compute good approximations for arrangement computation problems (e.g. envelope computation, cell decomposition, swept volume [Kim et al. 2003], etc.).

8 Acknowledgements

This research is supported in part by ARO Contract DAAD 19-99-1-0162, NSF awards ACI-9876914, ACI-0118743, ONR Contract N00014-01-1-0067 and Intel Corporation. We thank Joe Warren and Scott Schaefer for providing us with dual contouring code and the reviewers for their feedback and suggestions.

References

BISCHOFF, S., AND KOBBELT, L. 2002. Isosurface reconstruction with topology control. *Proc. of Pacific Graphics*, 246–255.

CUISENAIRE, O. 1999. *Distance Transformations: Fast Algorithms and Applications to Medical Image Processing*. PhD thesis, Universite Catholique de Louvain.

FRISKEN, S., PERRY, R., ROCKWOOD, A., AND JONES, R. 2000. Adaptively sampled distance fields: A general representation of shapes for computer graphics. In *Proc. of ACM SIGGRAPH*, 249–254.

GIBSON, S. 1998. Using distance maps for smooth representation in sampled volumes. In *Proc. of IEEE Volume Visualization Symposium*, 23–30.

HOFF, K., CULVER, T., KEYSER, J., LIN, M., AND MANOCHA, D. 1999. Fast computation of generalized voronoi diagrams using graphics hardw are. *Proceedings of ACM SIGGRAPH*, 277–286.

HOFFMANN, C. 2001. Robustness in geometric computations. *Journal of Computing and Information Science in Engineering 1*, 143–156.

HUANG, J., LI, Y., CRAWFIS, R., LU, S. C., AND LIOU, S. Y. 2001. A complete distance field representation. In *Proceedings of the conference on Visualization 2001*, IEEE Press, 247–254.

JU, T., LOSASSO, F., SCHAEFER, S., AND WARREN, J. 2002. Dual contouring of hermite data. *ACM Trans. on Graphics (Proc. SIGGRAPH) 21*, 3.

KEYSER, J., CULVER, T., FOSKEY, M., KRISHNAN, S., AND MANOCHA, D. 2002. Esolid: Exact solid modeling for low-degree curved solids. In *ACM Symposium on Solid Modeling and Applications*.

KIM, Y. J., VARADHAN, G., LIN, M. C., AND MANOCHA, D. 2003. Fast swept volume approximation of complex polyhedral models. *ACM Symposium on Solid Modeling and Applications*.

KOBBELT, L., BOTSCH, M., SCHWANECKE, U., AND SEIDEL, H. P. 2001. Feature-sensitive surface extraction from volume data. In *Proc. of ACM SIGGRAPH*, 57–66.

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching cubes: A high resolution 3D surface construction algorithm. In *Computer Graphics (SIGGRAPH '87 Proceedings)*, vol. 21, 163–169.

OHTAKE, Y., AND BELYAEV, A. G. 2003. Dual-prime mesh optimization for polygonized implicit surfaces with sharp features. *Journal of CISE*. To appear.

OHTAKE, Y., BELYAEV, A. G., AND PASKO, A. 2001. Dynamic meshes for accurate polygonanization of implicit surfaces with sharp features. *Prof. of Shape Modeling International*, 135–158.

PERRY, R., AND FRISKEN, S. 2001. Kizamu: A system for sculpting digital characters. In *Proc. of ACM SIGGRAPH*, 47–56.

SHEKHAR, R., FAYYAD, E., YAGEL, R., AND CORNHILL, F. 1996. Octree-based decimation of marching cubes surfaces. *Proc. of IEEE Visualization*, 335–342.

SUD, A., AND MANOCHA, D. 2003. Fast distance field computation using graphics hardware. Tech. Rep. TR03-026, University of North Carolina.

VARADHAN, G., KRISHNAN, S., KIM, Y. J., DIGGAVI, S., AND MANOCHA, D. 2003. Efficient max-norm distance computation and reliable voxelization. *Eurographics Symposium on Geometry Processing*.

WOOD, Z., HOPPE, H., DESBRUN, M., AND SCHRODER, P. 2002. Isosurface topology simplification. Tech. rep., Microsoft Research, MSR-TR-2002-28.