Simplification and Repair of Polygonal Models Using Volumetric Techniques

Fakir S. Nooruddin and Greg Turk, *Member*, *IEEE*

Abstract—Two important tools for manipulating polygonal models are simplification and repair and we present voxel-based methods for performing both of these tasks. We describe a method for converting polygonal models to a volumetric representation in a way that handles models with holes, double walls, and intersecting parts. This allows us to perform polygon model repair simply by converting a model to and from the volumetric domain. We also describe a new topology-altering simplification method that is based on 3D morphological operators. Visually unimportant features such as tubes and holes may be eliminated from a model by the *open* and *close* morphological operators. Our simplification approach accepts polygonal models as input, scan converts these to create a volumetric description, performs topology modification, and then converts the results back to polygons. We then apply a topology-preserving polygon simplification technique to produce a final model. Our simplification method produces results that are everywhere manifold.

Index Terms—Mesh simplification, mesh repair, volumetric models, morphological operators.

1 Introduction

We are in the midst of an explosion in the production of very large geometric models. Advances in many technical areas are fueling this trend: remote sensing, medical scanning, scientific computing, CAD. Remote sensing devices such as synthetic aperture radar produce enormous terrain data sets. Medical sensing technology such as MRI, CT, and PET scanners produce large volume data sets that lead to the creation of large isosurfaces. Scientific computing for applications such as structural analysis, synthetic wind tunnels, and weather prediction result in large data sets that may vary over time. Finally, computer-aided design is used routinely for large tasks in architecture and mechanical design. Polygon representations of CAD models may run into the hundreds of thousands of polygons. We require robust methods for manipulating such large models. Two important tools are repair of models that are nonmanifold and simplification of models. *Repair* is the process of taking a model that may have undesirable features such as cracks or self-intersections and creating a new model similar to the original but that has none of its flaws. *Simplification* of a polygonal model produces another model that has much the same appearance as the original but has many fewer polygons. Our paper addresses both of these tasks.

Many algorithms and applications require well-behaved polygon models as input. T-joints, cracks, holes, double walls, and more than two polygons meeting at an edge are just a few of the possible degeneracies that are often disallowed by various algorithms. Unfortunately, it is all too common to find polygonal models that have such problems. Applications that may require "clean" models include finite element analysis, radiosity, shape transformation, surface smoothing, calculation of moments of inertia, automatic model simplification, and stereolithography. Several approaches to polygonal model repair have been presented in the graphics literature. Unfortunately, most of these proposed methods are complex to program and some do not scale well as the polygon count increases. We present a method of scan-converting polygons into a voxel representation that yields a simple yet effective solution to polygon repair. The same voxelization process is also an important step in our simplification method.

Much work has been published recently in the area of automatic simplification of polygonal models and yet there are still many problems that need to be addressed. One of the important issues is the elimination of unnecessary fine details such as small holes or thin struts— a task that implies making changes to the topology of a model. Many of the earlier published simplification methods made an effort to preserve the topology of the original model. It eventually became evident, however, that topology is often a limiting factor in the simplification of a given object. Consider a box with 100 tiny holes punched all the way through it. A simplification method that preserves topology must retain at least three polygons to represent each hole and thus will retain at least 300 polygons, yet the model can be fairly well represented using just six faces. This problem has led several researchers to relax the restriction on topology preservation in order to remove small features such as small holes or thin bars and pipes. One important issue in topology simplification is whether a user may specify the exact size of the features to be removed from a model. A second issue is whether the simplification method produces manifold surfaces. Additional issues include the simplicity of programming, the memory requirements, and the computational cost and these are important, regardless of the treatment of topology.

• *The authors are with the College of Computing, Georgia Institute of Technology, 801 Atlantic Dr., Atlanta, GA 30084.*
E-mail: fakir@netscape.com, turk@cc.gatech.edu.

Manuscript received 9 Sept. 1999; accepted 13 Sept. 2000.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 110561.

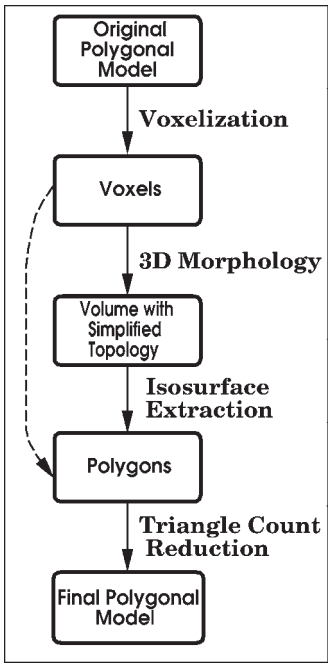


Fig. 1. The simplification pipeline. Dotted arrow shows the path used for model repair.

We have pursued a volumetric approach to geometric simplification. There are several reasons for this choice. First, volume models have none of the topological ambiguities that a polygonal model may have. For example, it is possible for a polygonal model to contain three or more polygons that share an edge—a nonmanifold situation. Purists may argue that such models should never be created in the first place, but the fact is that models with nonmanifold surfaces are only too common. We feel it is necessary to handle these common cases and we do this during the step that converts polygonal models to a volumetric representation. A second reason for working in the volume domain is that we then have access to a wide array of techniques that have been developed for image processing since volumes have the same regular structure as images but in one higher dimension. Finally, there are dozens of polygon-based methods for performing simplification and, in contrast, there have been relatively few proposed methods that make use of a volumetric representation.

Fig. 1 shows a schematic diagram of our simplification pipeline. There are four stages in our simplification method: voxelization, 3D morphology, isosurface extraction, and triangle count reduction. If the goal of the user is only to repair a polygonal model, then the morphological operations are not performed.

The remainder of this paper is organized as follows: In Section 3, we present a brief literature review of polygon simplification, voxelization, and model repair. In Section 4, we describe our new method for converting a polygonal model into a volumetric representation. In Section 5, we describe volumetric morphology and show how it is used to simplify the topology of an object. In Section 6, we discuss isosurface extraction and topology preserving triangle count reduction for producing the final model. Section 7 presents the results of our approach when used on a variety of models, discusses these results, and gives timing information. Section 8 summarizes the characteristics of our approach and describes possible future work.

2 Related Work

In this section, we review previous work in simplification and model repair. Because conversion of polygons into voxels is an important step in our approach, we also review related work in voxelization.

2.1 Simplification

A large number of approaches to geometric simplification have been published in the graphics literature. Rather than attempting to cover all of them, we will concentrate our attention on those simplification methods that allow the topology of a model to be changed.

Rossignac and Borrel created one of the earliest methods of performing polygonal simplification that allows topological changes [26]. Their approach is to group the vertices of a model into clusters that fall within the cubes formed by a uniform spatial subdivision. Those vertices that fall within one cell are merged into a single vertex and the degenerate triangles that are created by this are removed from the model. More recently, Low and Tan have enhanced this approach by making the vertex clustering independent of the position of the model in 3D and they also select the position of the new vertices using new heuristics [21]. Luebke and Erikson also used such a vertex clustering scheme in their view-dependent simplification approach [22]. Due to the dynamic nature of view-dependent simplification, they used a tree data structure in which to store a hierarchy of potential vertex clusters.

Schroeder et al. created one of the earliest polygonal simplification algorithms that successively removes vertices near relatively flat regions [28]. The original algorithm preserved topology, but, in more recent work, Schroeder extended this method to allow topological changes [29]. When no more vertices can be removed from the model due to topological restrictions of the algorithm, the method splits apart the polygons adjacent to a vertex. This allows greater freedom in vertex removal and thus allows the model to be further simplified. This newer algorithm also tracks error bounds at vertices, allowing bounds to be put on the amount of error incurred during simplification.

Garland and Heckbert demonstrated a topology-modifying simplification algorithm based on a generalization of the edge collapse operator [8]. An edge collapse replaces two vertices that share an edge with a single vertex, removing two triangles in the process. Their more general *vertex pair contraction* operator merges together any two vertices, regardless of whether they are joined by an edge or not. Garland and Heckbert use a quadric error metric to determine the best vertex pair contraction during simplification. Popovic and Hoppe take a similar approach to simplification, also using vertex pair contraction to reduce a model's complexity [27]. They use a cost function for a

contraction that includes a measure of distance to the original surface as well as a term that penalizes contractions that would merge vertices which have different material properties.

El-Sana and Varshney use an approach that is inspired by *alpha-hulls* (a distance-controlled portion of the Delaunay triangulation) to identify small holes and protrusions that can be removed from a model [5]. Sharp edges are marked as candidates that are likely to surround a hole. Then, an *alpha-prism* is used to determine whether a candidate hole is small enough to be filled. Identified holes have their associated polygons removed and the boundary edges that are created are filled using triangulation. They use the same process to identify and remove thin structures that protrude from a model.

Quite a different approach is taken by He et al. to perform topology-modifying simplifications of models [13]. They convert models into the volumetric domain, perform low-pass filtering, and then use isosurface extraction to produce a new polygonal model. Low-pass filtering of the volume model eliminates fine details such as thin tubes and surfaces and also closes small holes in the model. Unfortunately, low-pass filtering does not offer strict control over the topological changes that are to be made to an object. For instance, a hole of radius r might be filled if the hole is in the middle of an otherwise unbroken surface. A hole of the exact same size, however, can help create a larger hole if it is near one or more additional holes. In addition, large, thin surfaces of a model that should be retained can be accidentally eliminated by low-pass filtering. Despite these shortcomings, the volumetric filtering method has much to recommend it. Inspired by this approach, we created the new volumetric simplification method that we present in this paper.

2.2 Voxelization

Converting a polygonal model into a volume is an integral step in our method, thus we briefly review previous techniques that convert polygonal models into volumes. Wang and Kaufman use a method that samples and filters the voxels in 3D space to produce alias-free 3D volume models [33]. They place an appropriately shaped filter at the voxel centers and filter the geometric primitives (e.g., polygons) that lie inside the region of support of the filter kernel and this produces the final density value for the voxel. Their paper does not describe how to determine whether a point is interior to a collection of polygons. This is an issue that needs to be addressed if a solid rather than a thin-shelled model is to be created.

Huang et al. describe *separability* and *minimality* as two desirable features of a discrete representation [15]. If a discrete surface is thick enough to prevent ray penetration, it is said to meet the separability condition. If it contains only those voxels that are indispensable for separability, then it also satisfies the minimality condition. They use bounding spheres around vertices, bounding cylinders around edges, and bounding planes around each edge of each polygon to produce surfaces that meet both the separability and minimality conditions. The volumetric representations produced by this method are thin-shelled.

Schroeder and Lorensen create volumetric models by calculating a distance map from the input polygonal model [30]. Using this distance map, they find the closest polygon to a given voxel and use the polygon's normal to classify the voxel as interior or exterior. They then use a distance threshold to obtain an isosurface from this distance map. They use the resulting offset surface to generate swept surfaces for the purpose of path planning for object assembly.

2.3 Model Repair

There are several different approaches that have been taken toward repairing polygonal models, including user-guided repair, crack identification and filling, and creating manifold connectivity.

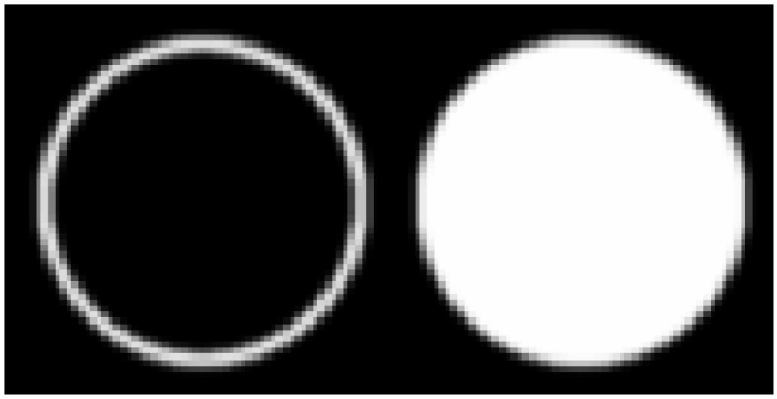
Several interactive systems have been proposed for fixing errors in polygonal models such as cracks and T-joints. Two such systems that used manual intervention to repair architectural models are described in [7] and [17]. Morvan and Fadel proposed a virtual environment in which to perform user-directed repair for layered manufacturing [24]. Interactive techniques for model repair become unattractive as the size of the models becomes large.

A number of other model repair methods have concentrated on automatic crack identification and filling. Bohn and Wozny use Jordan curve construction and local hole filling to fix models with cracks [3]. Barequet and Sharir describe a method for crack finding and filling by triangulation [2] and Barequet and Kumar improve upon this method by sometimes shifting vertices to eliminate cracks [1]. Murali and Funkhouser create a BSP-tree representation of a model and then construct and solve a linear system of equations in order to determine which cells of the BSP-tree are solid or nonsolid [25].

A third approach to model repair is presented by Gueziec et al. [12]. The goal of their repair method is to produce models that are everywhere manifold (perhaps with boundaries) and they are not concerned with eliminating cracks or self-intersections. Their method separates edges between polygons and then selectively stitches together some of these edges in a manner that avoids nonmanifold configurations. This method operates entirely upon the connectivity between polygons and does not examine the 3D positions of the vertices.

All of the repair methods described above operate directly upon a polygon or half-space description of a given model. The method of model repair that we present in this paper is unique in that we convert a model into voxels in order to perform repair.

Now that we have reviewed the related work, we will describe the components of our model manipulation pipeline for simplification and repair.



(a) (b)

Fig. 2. (a) Slice through a thin-shelled volumetric representation of a sphere. (b) Slice through a solid volumetric representation of a sphere.

3 Polygons to Voxels

In order to use morphological operators to simplify topology, we must first voxelize the given polygonal model. In this section, we present two new methods of voxelization, the *parity-count* and the *ray-stabbing* methods. At the end of this section, we describe how voxelization provides a simple method for performing model repair.

A voxel representation of a model is a regular grid of cells, in our case a rectilinear grid, in which each cell (voxel) contains a density value in the range of zero to one. In this paper, we will use a voxel-value of zero to represent a portion of unoccupied space and a value of one to represent a voxel that is entirely interior to our model. Values between zero and one represent voxels that are near the surface of an object.

As described above, there are several published methods for performing voxelization of polygons [15], [30], [33]. Unfortunately, none of the published techniques are satisfactory for our needs. For our purposes, the voxel representation should not be thin-shelled. A thin-shelled voxelization of polygons is one in which only voxels that are near a polygon of the original model have a nonzero voxel value. Thin-shelled voxelization is performed by finding the distance between a given voxel and the nearest polygon [18], [33]. A thin-shelled representation of a sphere, for instance, would contain nonzero voxels only near the sphere's surface. Such a sphere would have a large region of zero-valued voxels inside its boundary. Fig. 2a shows a slice through such a thin-shelled sphere model. Performing isosurface extraction on such a model would produce a polygonal model that had two surfaces that are very near one other. In contrast, the voxel models that we use have voxel values of one in the interior of the object so that isosurface extraction yields a single surface. Fig. 2b shows a slice through such a voxel model of a sphere.

3.1 Parity Count

To produce voxel models with true interiors, the exterior/interior classification of a voxel must take into account nonlocal aspects of the polygonal model. We will first discuss our *parity count* method of voxel classification when used on manifold polygonal models that are watertight (have no cracks or boundaries). For such models, we classify a voxel V by counting the number of times that a ray with its origin at the center of V intersects polygons of the model. An odd number of intersections means that V is interior to the model and an even number means it is outside. This is simply the 3D extension to the parity count method of determining whether a point is interior to a polygon in 2D. Note that, for manifold models, the direction of the ray is unimportant and we can take advantage of this to speed up the voxel classification. In essence, we cast many parallel rays through the polygonal model and each one of these rays classifies all of the voxels along the ray. For an $N \times N \times N$ volume, we need to cast only $N \times N$ rays, with each ray passing through N voxel centers. Instead of using ray-tracing, however, we actually use orthographic projection and polygon scan-conversion to create a "deep" z-buffer. Each pixel in the z-buffer retains not just the nearest polygon, but a linked list of depth samples. Each of these depth samples records an intersection with one polygon. Thus, a "deep" pixel represents one of the parallel rays that has been cast through the model. Each voxel behind a given pixel can be rapidly classified by counting how many depth samples are behind or in front of the voxel center. Fig. 3a shows a 2D representation of this process. In this figure, each blue circle represents a depth sample along a ray. Polygon scan-conversion takes advantage of incremental calculations, so this process is much faster than a ray-tracing approach would be.

Although the parity count method works well for manifold models, many polygonal models have various degeneracies that require us to modify the voxelization process. One common problem is for a model to have small cracks or holes in the surface. The Stanford Bunny model, for example, has several holes on its base and the Utah Teapot contains a hole at the tip of the spout. Fig. 3b illustrates the problem. To voxelize such models, we extend the parity count method by using k different directions of orthographic projection and by scan-converting the model once for each direction. Each of the k projections votes on the classification of a voxel (interior or exterior), and the

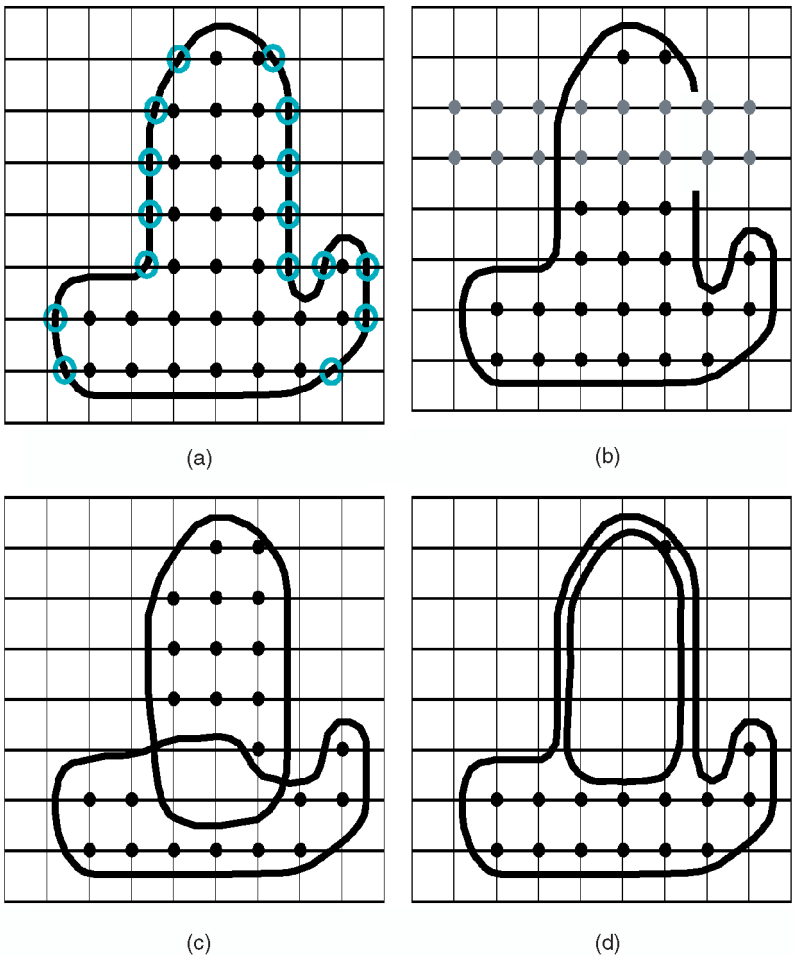


Fig. 3. Scan-converting polygonal models with a variety of degeneracies. (a) Scan-converting a closed model using the parity count method. The black dots represent voxels that are inside the model. the blue circles show where the scanlines intersect the model. (b) Scan-converting a model with ahole. The gray dots represent voxels for which we do not know whether they are inside or outside the model. Scan converting from multiple directions solves this problem. (c) Scan-converting a model with intersecting parts using parity count. This is another instance where reay stabbing yields better results. (d) Scan-converting a double-walled model using parity count. This shows why some models require the ray stabbing method.

majority vote is the voxel's final classification. For water-tight models, all of the votes will agree. This is not the case, however, for models that have a crack through which a ray may pass. Rays that pass through a single crack or hole will have an odd number of depth samples and these rays are marked as invalid and do not vote. It can happen on rare occasions that one ray will pass through two cracks and this will cause the ray to improperly classify many of the voxels. The majority voting between the directions of projection overrules the voting of such rays. Typically, we perform three orthographic projections, one in each of the major axis directions. For troublesome models, we project in 13 directions, three along the major axes and 10 directions that are described by the surface normals of an icosahedron. By choosing an odd number of projection directions, we avoid having many ties in voting. Voting ties can still occur due to invalid rays and we mark such voxels as being exterior to the model.

Fig. 4a and Fig. 4d are two views of the Stanford Bunny polygonal model and Fig. 4d shows the large holes in its base. Fig. 4b and Fig. 4e show the result of using a single orthographic projection for the parity count voxelization method. The holes in Fig. 4b and Fig. 4e are the result of the algorithm classifying invalid columns of voxels (an odd number of ray intersections) as being exterior to the model. Using 13 projections creates a water-tight model, shown in Fig. 4c and Fig. 4f. This repaired model has none of the holes that were in the original model. In addition to the bunny model, both the turbine blade and the chair models (Figs. 6 and 9) were voxelized using the parity count method.

Our representation of the "deep" z-buffer is similar to the Layered Depth Images [31] work of Shade et. al. Their Layered Depth Images store multiple depth pixels at each image pixel and each of these depth pixels represents the pixels along one line of sight. In our case, each pixel in the depth-buffers represents multiple intersections between a ray and the polygonal model.

We note that Schroeder and Lorensen also have converted polygonal models to voxel models that have true interiors [30]. They find the closest polygon to a given voxel and classify the voxel based on the polygon's normal. Their method is tolerant of models with small cracks, but it would produce poor results for polygonal models that are double-

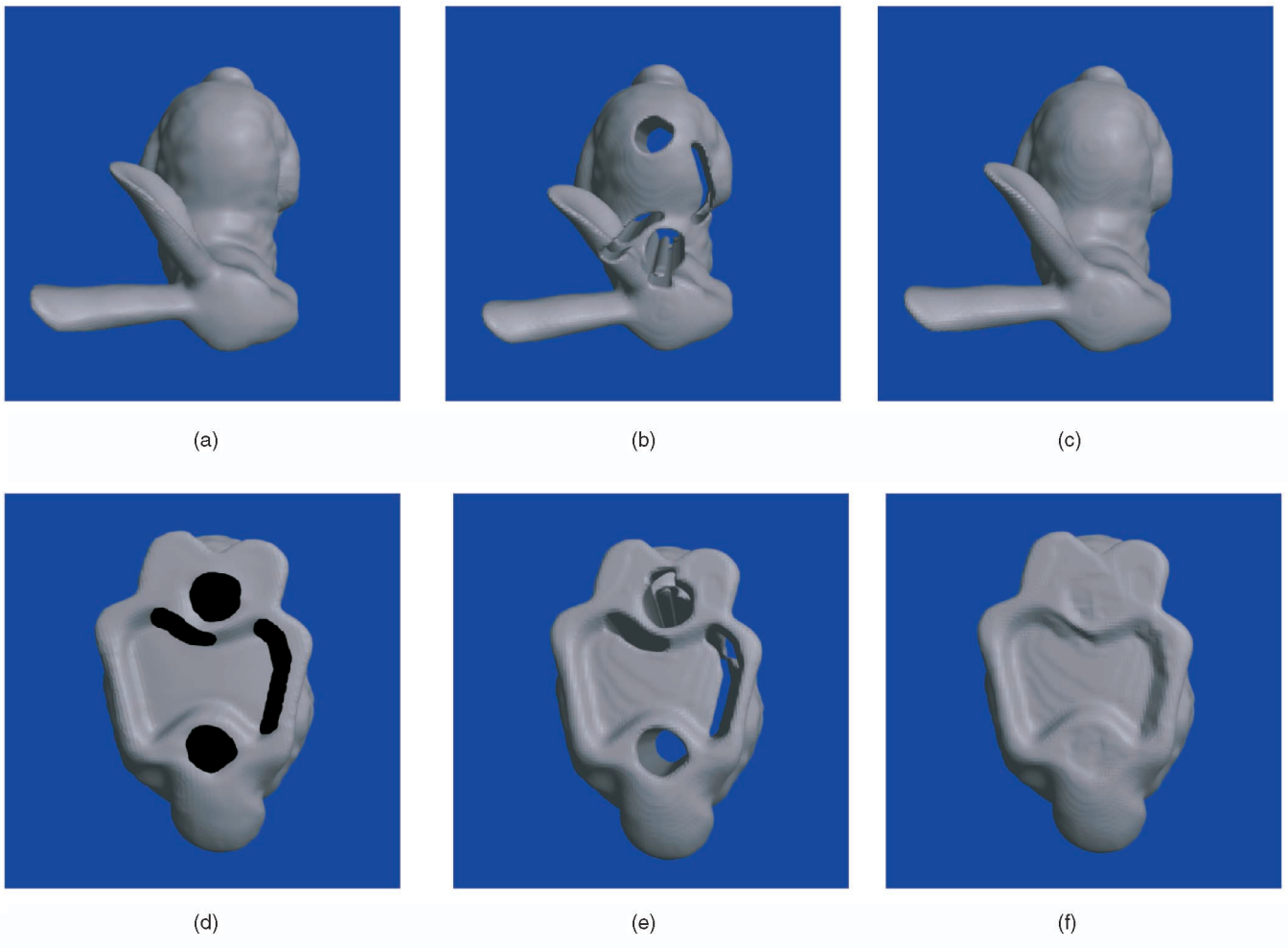


Fig. 4. (a) Original bunny model (top view) 69,451 faces. (b) Results of parity count using one scanning direction (top view) 134,920 faces. (c) Results of parity count using 13 scanning directions (top view) 101,536 faces. (d) Bunny model (bottom view) 69,451 faces. (e) Results of parity count using one scanning direction (bottom view) 134,920 faces. (f) Results of parity count using 13 scanning directions (bottom view) 101,536

walled or that have intersecting surfaces. We handle such models using our ray stabbing approach.

3.2 Ray Stabbing

Unfortunately, cracks and holes are not the only kind of troublesome degeneracies in polygonal models. One other common problem is to have a model that is composed of several interpenetrating subparts. This is often found in articulated figures of humans and animals, where each limb or limb segment is a separate closed surface. For instance, an upper arm might be placed so that portions of its surface are inside the torso. This is not a problem if we are just rendering such a model. The parity count method, however, would incorrectly classify the overlapped portions of the arm and torso as being outside of the model. Fig. 3c shows an example of two objects intersecting in this manner. Another common problem is to have a polygonal model in which there is more than one polygon at or near the same location in space. This is often the case for mechanical models where two subcomponents are made exactly adjacent, but where the shared surface is represented by polygons from both subcomponents. Double walls in building models are a similar problem. Such redundant polygons may cause problems for the parity count method as well. Fig. 3d shows that the parity count method would create an empty interior for such a model. To voxelize this kind of model, we have created the *ray-stabbing* method of voxel classification.

The ray-stabbing method also makes use of orthographic projections of a polygonal model. It differs from the parity count method, however, in the way it interprets the depth samples of a ray. The ray stabbing method only retains the first and last depth sample along each ray. In effect, each ray only keeps those points of intersection where the ray first stabs the surface of the model. Keeping both the first and last depth samples is equivalent to stabbing the surface from two directions at once, at no extra cost. A voxel is classified by a ray to be interior if the voxel lies between these two extreme depth samples; otherwise, it is classified as an exterior voxel. For a single direction of projection, this can cause some voxels to be misclassified as being interior to the surface. To avoid this, we perform several projection in different directions. If *any* of the projections classify a voxel as exterior, it is given an exterior final classification. Only those voxels that are classified as interior for *all*

TABLE 1
This Table Summarizes the Types of Degeneracies that the Ray-Stabbing and Parity-Count Methods Are Able to Fix

Types of Degeneracies		
	Parity Count	Ray Stabbing
Fixes T-Joints	Y	Y
Fixes Cracks / Holes	Y	N
Retains Interior Detail	Y	N
Merges Interpenetrating Surfaces	N	Y
Fixes Non-manifold edges and vertices	Y	Y

projections are given the final classification of interior. Although reasonable voxel models result from three projections, we typically perform 13 projections for the ray-stabbing approach. Both the Al Capone and motor models of Figs. 7 and 8 were voxelized using the ray stabbing method.

3.3 Polygonal Model Repair

We perform polygon repair by converting a model to a volumetric representation and then converting it back to polygons using isosurface extraction. This produces an everywhere manifold polygonal model that is free of holes, cracks, T-joints, double walls, and interpenetrating polygons. The number of polygons produced by the conversion to and from the voxel domain is a function of the resolution of the voxel representation. If the polygon count for the model should be small, we reduce the number of polygons using standard polygon-based simplification. Our polygon repair method uses the same basic pipeline of operations as our simplification approach, but we skip the volumetric morphology step, as indicated by the dotted line in Fig. 1. Fig. 7 shows an example of polygon repair of a model with a number of interpenetrating parts and Fig. 4 illustrates repair of a model with several large holes.

The final results of polygon repair are significantly improved if proper sampling and antialiasing are performed during voxelization. To do so, we use supersampling and filtering in our implementation. We have implemented several filter kernels and our best results are from a Gaussian filter kernel with a radius of two voxels and a standard deviation of 0.7 voxels. For an excellent survey of filter kernels for volume antialiasing, see [23]. We typically use $3 \times 3 \times 3$ supersampling to achieve high quality results, but using $2 \times 2 \times 2$ often produces results that are quite acceptable.

It is left to the user to choose between the parity-count method and the ray-stabbing method when repairing a given model. However, we can offer some guidelines on which method to use based on the types of degeneracies that are present in the model to be repaired. If the model does not have any interpenetrating parts, then the parity-count method should be used. As shown in Table 1, the parity count method is able to repair most of the commonly occuring degeneracies in polygonal models such as non-manifold vertices and edges, T-joints, etc. In the case that the model does have interpenetrating parts, then the ray-stabbing method should be used to eliminate them. Unfortunately, neither the ray-stabbing nor the parity-count method can deal with models that have small cracks *and* interpenetrating parts. One possibility that exists in dealing with such models is that the user can apply a polygon-based model repair method such as that presented in [12] to remove the cracks from the model. Once that has been done, the ray-stabbing method can then be used to eliminate the interpenetrating parts of the model.

4 Morphological Operations

The morphological operators constitute the heart of our topology simplification algorithm. These operators are well suited to simplifying the topology of objects because they present a clean and efficient way to remove small features, close holes, and join disconnected components of a model. In addition, *openings* and *closings* provide precise tolerances so that the user can specify the size of the feature to be removed or the size of the hole to be closed. Finally, because these operations are done in the volume domain, we are able to recover a manifold mesh after the topology simplification has taken place.

The first step in using morphological operators is the calculation of a distance map. Given a binary volume that is classified into *feature* and *nonfeature* voxels, a distance map associates with each voxel the distance to the nearest feature voxel. Feature voxels are those that are inside the object and nonfeature voxels are those that lie outside the object. Feature voxels have a distance map value of zero. We used Danielsson's algorithm [4] to calculate the distance map on our volumes. Specifically, we chose to implement the 4SED (four-point sequential Euclidean distance mapping) algorithm proposed by Danielsson. This algorithm is fast, but it is known to give slightly incorrect distances in some situations due to the fact that only the four immediate neighbors of a pixel contribute to its distance value. However, Danielsson reports that the absolute error in this case is less than 0.29 pixel units, which is quite acceptable for our purposes. When more accuracy is necessary, the user may simply use a finer voxel grid.

Below, we explain how the algorithm works on 2D images, after which we give a brief outline of how this is extended to 3D in order to create distance maps for volumes. Danielsson's 2D algorithm produces a floating-point distance map that contains one scalar value per entry. During the calculation of the distance map, the distances at each pixel are represented as two-dimensional integer

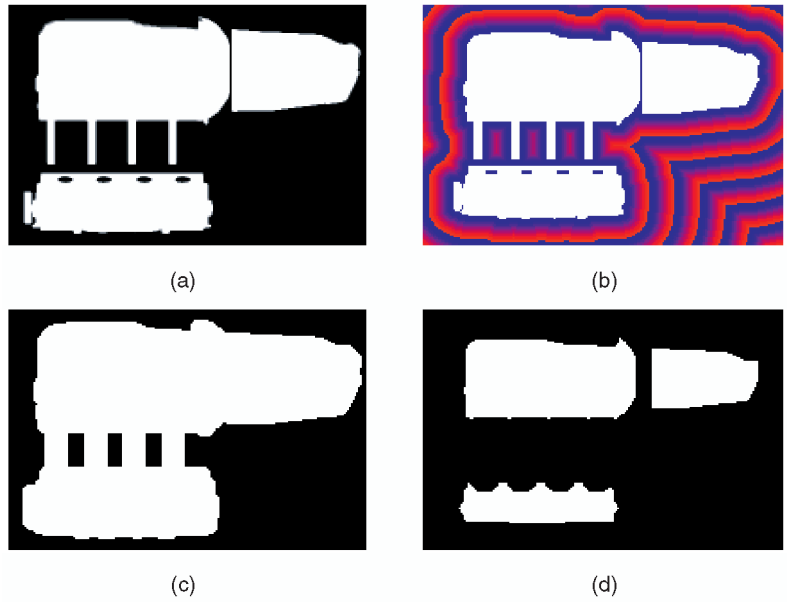


Fig. 5. (a) Slice through voxelized motor, (b) distance map, (c) dilation, (d) erosion.

vectors. For a given pixel, its distance map vector gives the integer distance in the x and y directions to the nearest feature pixel. The final step in the algorithm involves calculating the magnitude of these vectors, yielding the final scalar floating-point distance map.

The 2D algorithm starts by assigning a distance of zero to all feature pixels and a value of MAXVAL to the nonfeature pixels. After the distance map is initialized, the image is scanned from bottom to top (the j direction). A pixel's distance map value changes if its distance map value is greater than that of its neighbors. Thus, distance map values propagate from the sources of change (the feature pixels) to the nonfeature pixels. For every j scan of the image, new values are propagated left, right, and from the row of pixels below. This bottom-to-top scan only propagates information about a given feature pixel horizontally and upward. The image is then scanned a second time, from top to bottom, so that the distance values are propagated downward as well.

First loop of Danielsson's algorithm (sweeping from bottom-to-top)

for j = 1 to dy -1

Examine pixels below the current row
for i = 0 to dx - 1
if $\text{mag}(\overrightarrow{D(i,j)}) \leq mag(\overrightarrow{D(i,j-1)} + \overrightarrow{< 0,1 >})$
$\overrightarrow{D(i,j)} = \overrightarrow{D(i,j-1)} + \overrightarrow{< 0,1 >}$

Examine pixels to the left of each pixel in a row
for i = 0 to dx - 1
if $\text{mag}(\overrightarrow{D(i,j)}) \leq mag(\overrightarrow{D(i-1,j)} + \overrightarrow{< 1,0 >})$
$\overrightarrow{D(i,j)} = \overrightarrow{D(i-1,j)} + \overrightarrow{< 1,0 >}$

Examine pixels to the right of each pixel in a row
for i = dx - 2 downto 0
if $\text{mag}(\overrightarrow{D(i,j)}) \leq mag(\overrightarrow{D(i+1,j)} + \overrightarrow{< 1,0 >})$
$\overrightarrow{D(i,j)} = \overrightarrow{D(i+1,j)} + \overrightarrow{< 1,0 >}$

The above pseudocode is that of the bottom-to-top scan of an image. The second loop (top-to-bottom scan of the image) of Danielsson's 2D algorithm is similar to the loop shown above. The extension of this algorithm to 3D involves applying Danielsson's algorithm on a slice by slice basis to the volume. There are two passes done through the volume: one each in the forward and backward directions in the k dimension. For each of these passes, the distance map for each slice is calculated as described above. In 3D, a voxel's distance map value is calculated from the distance map values of its six immediate neighbors.

The two atomic morphological operations are *erosion* and *dilation*. They take as input the volume, the distance map, and an erosion/dilation distance. For dilation, we look through the distance map and any nonfeature voxel that has distance less than or equal to the threshold is turned into a feature voxel. Erosion is the complement of dilation. In this case, we negate the volume (i.e., a feature voxel becomes nonfeature and vice versa), calculate the distance map, and then perform a dilation. After this, the volume is negated again to obtain the final result. These basic morphological operations are commonly used in image processing [16].

Fig. 5 shows the results of applying erosion and dilation to a 2D image. Fig. 5a shows the original image. This is a slice of the volumetric description of the motor model. Fig. 5b shows a colorized distance map in which the colors indicate the distance of a pixel from the surface. Near the surface, the blue color indicates a small distance, while the red color indicates a large distance. The colors cycle at greater distances. Using this distance map, we performed dilation and erosion on the image. Fig. 5c shows the result of performing dilation on the input image. It demonstrates how dilation will close small holes and join previously unconnected parts of the input image. The result of

performing erosion is shown in Fig. 5d. Erosion eliminates thin structures and increases the distance separating two unconnected parts of the image.

While useful by themselves, erosion and dilation are usually used in conjunction with each other. The reason is that, if they are used in isolation, then they increase (in the case of a dilation) or decrease (in the case of an erosion) the bounds of the volume. When an erosion is done followed by a dilation, it is called an *opening*. This is due to the fact that this operation will widen holes, eliminate small features, and disconnect parts of the model that are connected by thin structures. The complement of this operation is a *closing*, which is a dilation followed by an erosion. This will close holes and connect previously disconnected parts of the model. There is a connection between the resolution at which a polygon model is scan-converted and the distance parameter that is used by the morphological operators to simplify the volumetric description of the input polygonal model. The size of features such as tunnels and thin-structures grows larger in the volumetric description as the scan-conversion resolution is increased. Therefore, the distance parameter used by the morphological operators to eliminate such features must also be increased. In our experiments, we have found that the polygon model should be scan-converted at approximately 10 times the distance value used by the morphological operators. Table 3 shows the sizes of the models as they move through the simplification pipeline along with the closing distances that were used for the Turbine Blade and Chair models. For both these models, a closing distance of 10 voxels was used. The closing parameter is dictated by the voxel resolution and the largest hole that needs to be closed. Notice that if erosion or an opening is performed on a thin-shelled volume model, the erosion will completely destroy the surface. This is another reason we require that our voxelization process not produce a thin-shelled volumetric representation.

5 Polygonal Model Creation and Triangle Count Reduction

Now that we have seen how to remove small features using volumetric morphology, we turn our attention to converting the model back to polygons. There are two steps involved in this: isosurface extraction and polygon simplification.

5.1 Isosurface Extraction

To create a manifold polygonal model, we extract an isosurface from our volumetric representation of the model. We do this using a modified version of the standard Marching cubes algorithm [20]. This algorithm works by examining the eight adjacent voxels at the corners of a cube. Using a threshold value, the corners of this cube are classified as being either inside or outside the surface. This classification scheme yields 256 possible configurations. A lookup table is used to generate triangles within a cube based on the configuration of its corners. The Marching Cubes algorithm can produce up to 11 triangles from each cube. The original algorithm proposed in [20] produces ill-formed isosurfaces in some cases. We use the modifications to Marching Cubes proposed in [10] to extract isosurfaces that are everywhere manifold.

As shown in Fig. 1, there are two possible paths through our simplification pipeline: one where volumetric morphology is performed and the other where we extract the isosurface directly after scan-converting the input polygonal model into a volumetric representation. We omit the morphology stage if our goal is either to repair a polygonal model or to eliminate its interior detail. If no morphological operations are performed, then the resulting isosurface is smooth. This results from the fact that, during scan conversion, we use supersampling to obtain voxel values that vary between 0 and 1. On the other hand, because morphological operations act on binary volumes, the isosurfaces extracted after volume morphology have voxelization artifacts. We use Taubin's smoothing technique to reduce these artifacts [32]. Taubin uses a low-pass filter over the position of the vertices to create a new surface that is smoother than the original. One of the design goals of this smoothing method was that it be able to reduce the voxelization artifacts that are introduced during the voxelization stage. Gortler et al. use the same method in The Lumigraph to smooth polygonal models that they produce via isosurface extraction [11].

5.2 Triangle Count Reduction

The isosurface we extract usually has simpler topology than the input model. In addition, because the Marching cubes algorithm considers cubes in isolation, it frequently overtessellates the surface. These two properties of the isosurface allow us to drastically reduce its triangle count without degrading the model's quality. To achieve this, we use Garland and Heckbert's polygon-based simplification method, which is guided by a quadric error measure [8]. We use their method because it is efficient and produces high quality results. Garland and Heckbert use the planes passing through a vertex to estimate the amount of error introduced by an edge collapse. As discussed in Section 3, their simplification process is based on a generalized form of the edge collapse operation called vertex pair contraction. A vertex merge may join together two vertices that do not share an edge, altering the topology of a model. Since we have already performed topological modifications using volumetric morphology, we only allow the merging of vertices that share an edge when using their simplification method. One consequence of using this edge-collapse-based simplification method to reduce the triangle count of the isosurface is that, as this technique does not guarantee the preservation of the volume of a model, the simplified/repaired meshes are smaller in size than the original model.

When volume morphology is used to simplify the topology of an object, the resulting volume typically has no small holes, interior details, or tunnels. Thus, all the polygons of the resulting isosurface can be used to represent the exterior surface of the object. This enables us to reduce the triangle count of a given model to much lower levels than would have been possible with the original model.

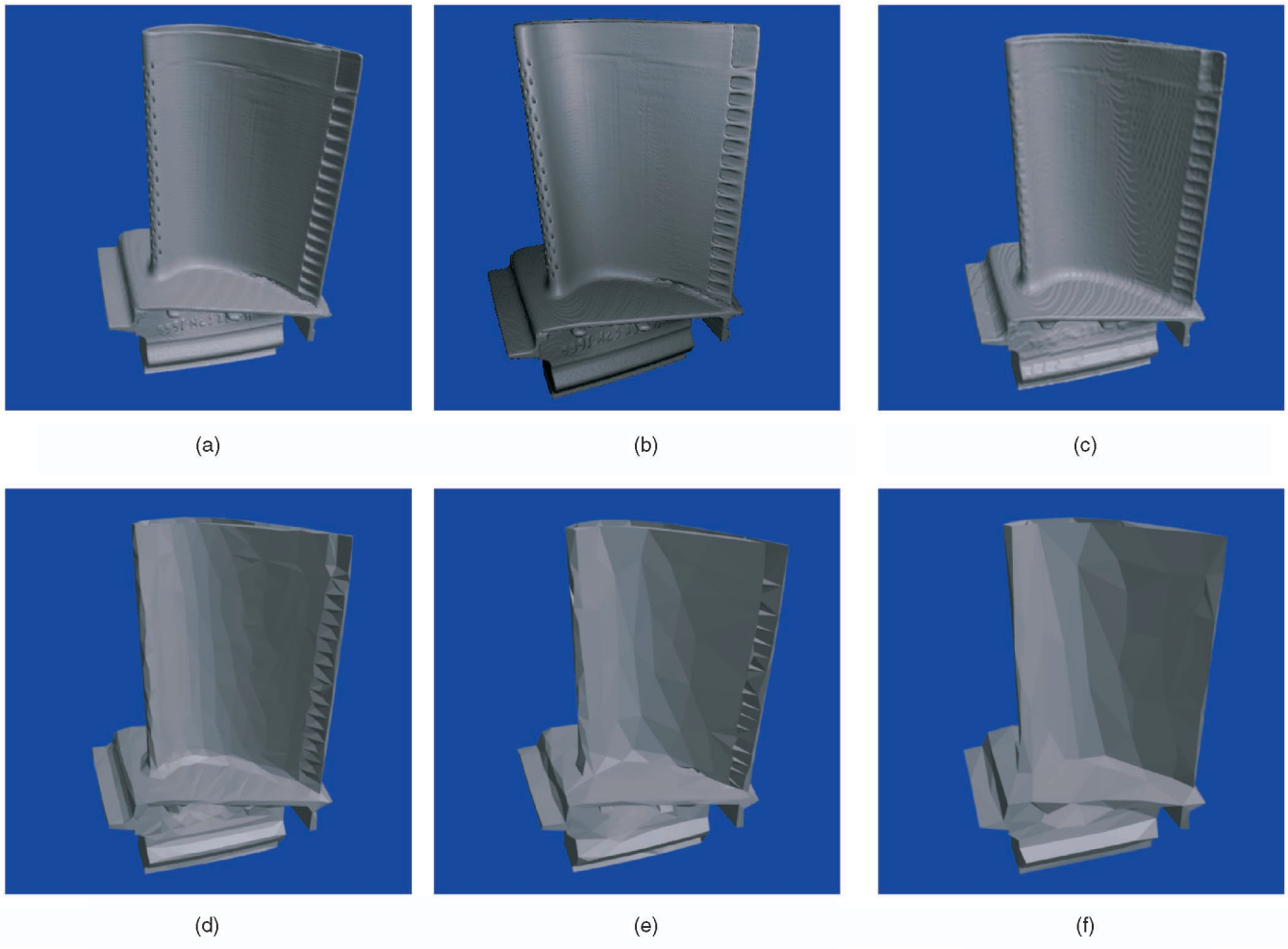


Fig. 6. (a) Original turbine blade model, 1,726,892 faces. (b) Voxelized model (volume rendered), $643 \times 300 \times 382$ voxels. (c) Isosurface after morphological closing, 578,098 faces. (d) Simplified model, 2,200 faces. (e) Model simplified by Qslim, 2,200 faces. (f) Simplified model, 500 faces.

In order to maintain the color of a given model, we record the color values of the polygonal model during voxelization. These color values are associated with *surface* voxels. Surface voxels are those that intersect a polygon of the input model. During scan conversion, when we detect the intersection of a polygon with a voxel, we record the color of the polygon and associate it with the voxel that the polygon intersects. These color values are then carried through the rest of the simplification pipeline. During the morphology stage, we process a volumetric representation of an object that has color by dividing the morphological operations into two steps. In the first step, a distance map is calculated based on the density values. This distance map is then used to perform either the opening or closing operation. After the morphological operations have been performed on the density values, a second distance map is calculated using the color voxels. This distance map is used to find the color of the nearest surface voxel (in the density volume) to each voxel in the color volume. The density volume and the color volume are combined to form the new volumetric representation of the input polygonal model. The next step in the simplification pipeline is isosurface extraction. Here, we extend the Marching Cubes algorithm to take into account color when generating triangles. In our version of the Marching Cubes algorithm, we simply interpolate the color values associated with voxels when we are generating triangles within a cube. These interpolated color values are then associated with the triangles that are generated. The final step in our simplification algorithm is triangle count reduction. In this step, we use Garland and Heckbert's newer simplification method that preserves the color of the original surface [9]. This extension to their previous algorithm simplifies meshes that have color associated with the vertices by adding the red, green, and blue color coordinates to the quadric error measure.

6 Results

Figs. 6, 7, 8, and 9 show the results of our algorithm on four models. The model of Fig. 6 is a CT scan of a turbine blade. This model poses a challenge to most simplification algorithms due to its size, fine interior detail, and complex topology (small holes, thin tunnels, etc.) Many methods cannot simplify this model beyond a certain point because of its complex topology. Fig. 6a shows the original model, Fig. 6b is a volume rendering of the voxelized model, and

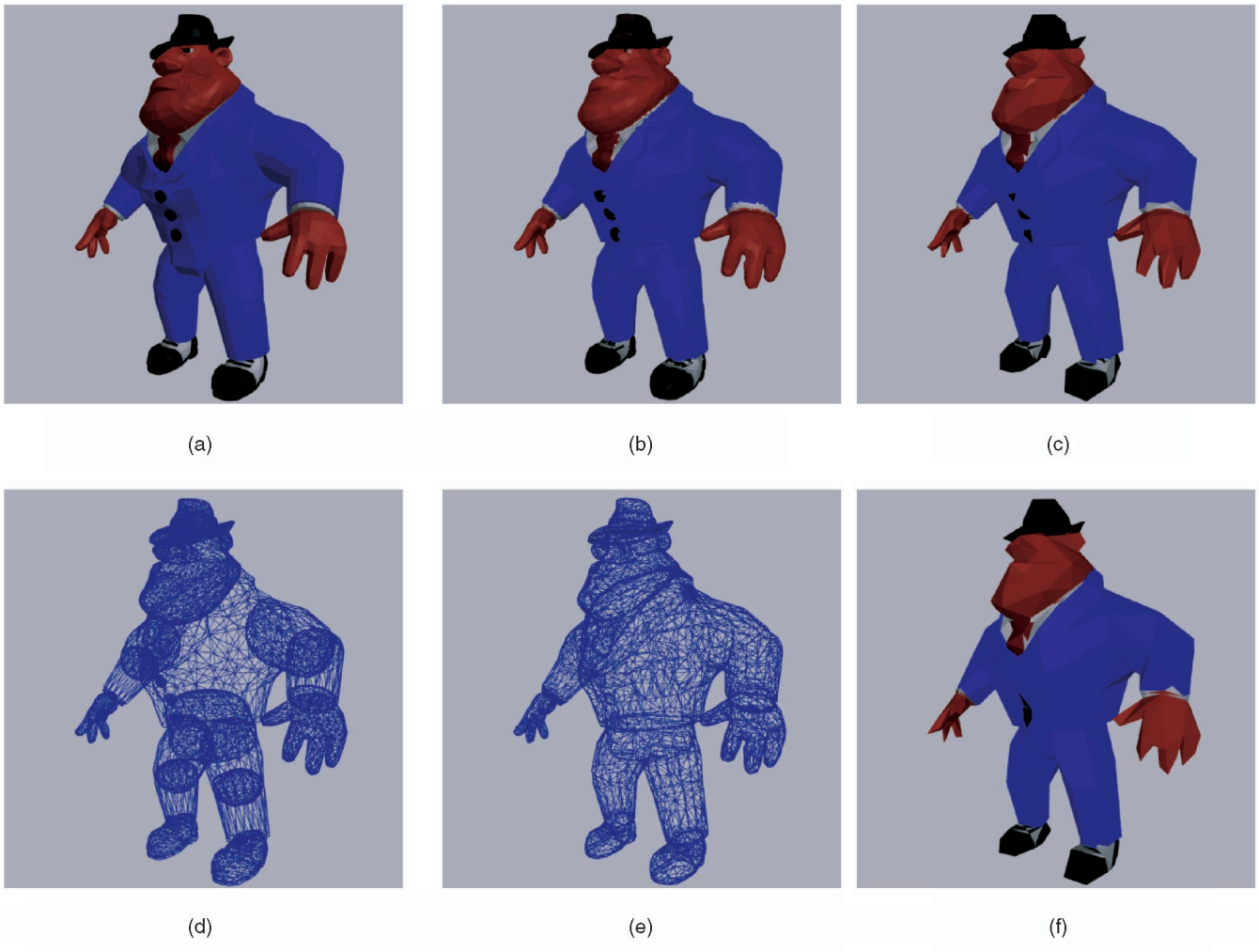


Fig. 7. (a) Original Al Capone model, 13,476 faces. (b) Repaired model, 13,476 faces. (c) Simplified model, 1,721 faces. (d) Wireframe of original model (shows intersecting parts), 13,476 faces. (e) Wireframe of repaired model (shows manifold surfaces), 13,476 faces. (f) Simplified model, 860 faces.

Fig. 6c shows the model after morphological closing and isosurface extraction. Fig. 6d and Fig. 6f show the surface after the polygon reduction stage. For comparison to Fig. 6d, the effect of using Garland and Heckbert's quadric simplification method alone is shown in Fig. 6e. As can be seen from the images, our approach retains more details than Qslim (Garland and Heckbert's method) alone for the same number of faces in the simplified model. We note that there are many parameters for Garland and Heckbert's method. For fairness, we use the same choice of parameters to produce both Fig. 6d and Fig. 6e. (Recall that Qslim is a component of our simplification pipeline.)

Fig. 7 demonstrates our approach on a model of Al Capone. This hand-constructed model has 15 interpenetrating parts. The head, arms, and legs continue into the torso region, as can be seen by the darker regions in the wireframe rendering of Fig. 7d. In addition, this model also has color, which is an attribute that we preserve during the simplification and repair. For this model, we do not perform any morphology on the model, but instead recover an isosurface after the model has been scan-converted. The resulting isosurface has no intersecting parts and is everywhere manifold. This guarantees that the different body parts will not become disjoint from one another when the object is simplified, as shown in Fig. 7c and Fig. 7f. After the isosurface was extracted, we reduced its triangle count to match that of the input model. The resulting surface can be seen in Fig. 7b and Fig. 7e of the figure.

Fig. 8 demonstrates simplification of a car motor. This model contains many degeneracies such as T-joints, zero-area faces, and nonmanifold vertices. In addition, the motor model has interesting topology in that there are a number of parts connected by thin structures and a lot of interior detail. Again, in the comparison with Qslim, our method retains more detail at comparable levels of complexity.

Fig. 9 shows simplification of a model chair. This model demonstrates the topology modification capabilities of our algorithm. Our approach closes the holes that are present in the back and the seat of the model. We do not know of any other simplification algorithms that would smoothly join the slats of the chair into a single component and produce a manifold surface.

The results shown in these figures illustrate the improvement that our morphological operators make when

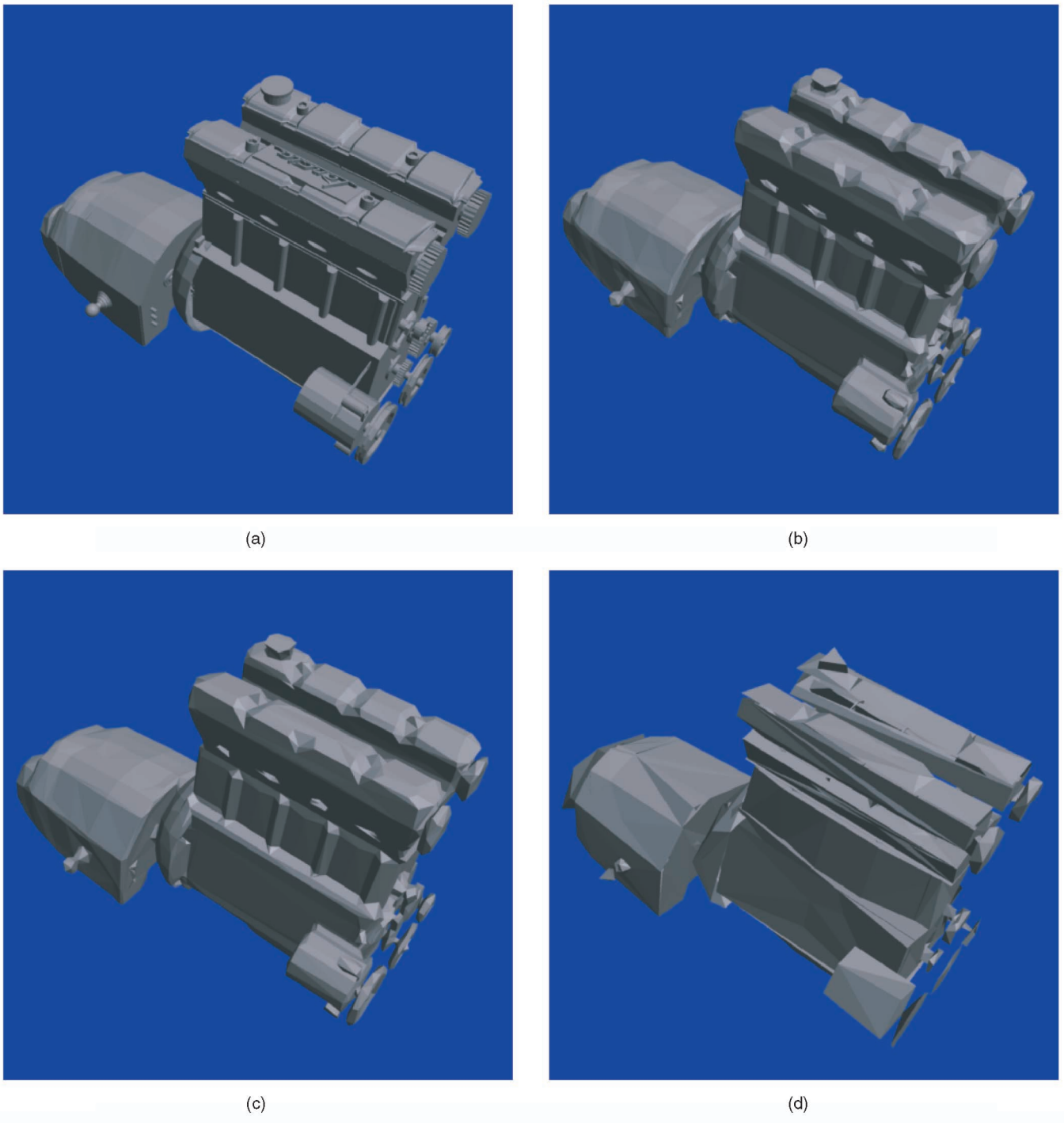


Fig. 8. (a) Original motor model, 140,113 faces. (b) Simplified model, 5,000 faces. (c) Simplified model, 3,300 faces. (d) Model simplified by Qslim, 3,300 faces.

used in conjunction with Garland and Heckbert's polygon-based simplification approach. We believe that similar benefits will result if our morphological technique is used in conjunction with any other polygon-based method. Many polygon-based simplification methods are affected during the later stages of simplification by small features that were present in the original model, even if those features have been removed by the simplification process. The memory of such small features is often used in distance measures that help guide the simplification process, such as are used in [8], [14]. Other simplification methods, such as [19], [30], do not measure distances based on the original geometry, but are still influenced by the original topological features such as small holes. The benefit of the morphological stage of our pipeline is that it produces models in which the small features are completely absent. When morphological changes are performed before polygonal simplification, the polygon-based simplification stage never needs to be concerned with the small features in any way. The polygon-based method is never penalized for creating a surface that

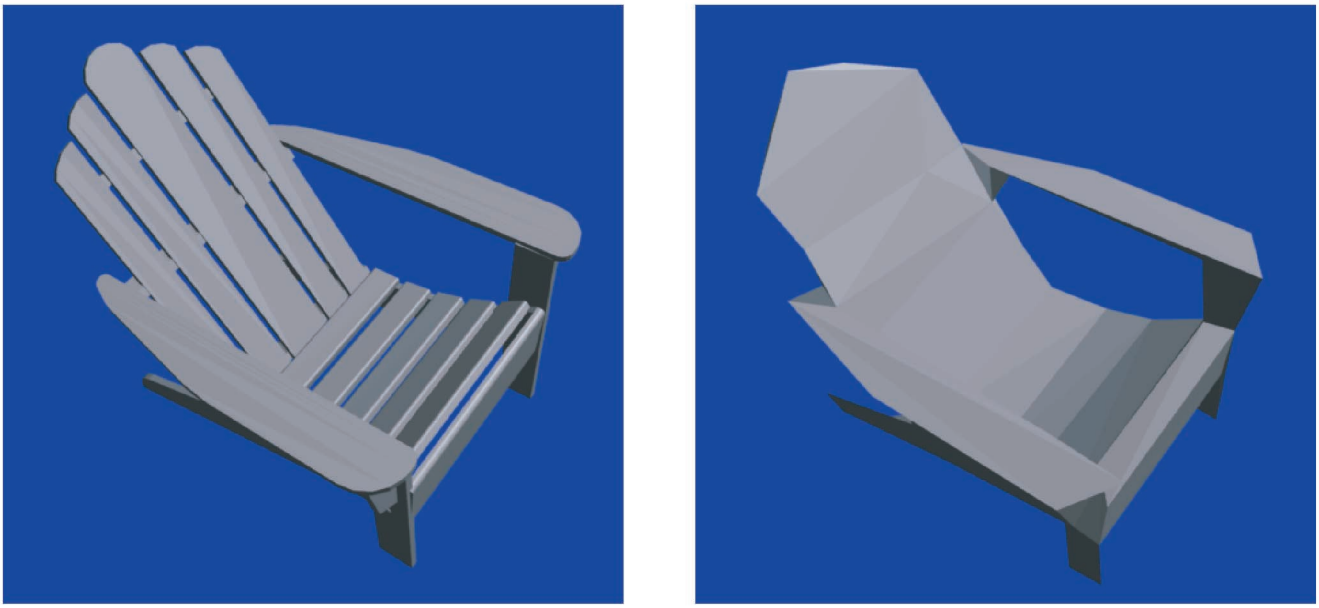


Fig. 9. (a) Original chair model (3,261 faces). (b) Simplified model (170 faces).

is distant from the small features because it never has knowledge of these small features. This results in better simplified models.

Table 2 shows the timing results for each stage of the simplification pipeline. Table 3 contains the sizes of the different representations of a model as it moves through the pipeline. To collect timing statistics that reflect all of the stages of the pipeline, we performed an extreme amount of simplification on all of the models—simplifying each model down to 200 faces. (Note that these 200 face models are not the models shown in the figures.) When the input model contains a large number of faces, then voxelization is the most time consuming stage of the pipeline. This is evident from the timing results of the turbine blade model, where voxelization accounted for 78 percent of the total time required to simplify the model. In other cases, most of the stages in the simplification pipeline took about the same amount of time. By way of comparison, we note that it took Qslim 4:08 minutes to simplify the blade down to 200 polygons and 15 seconds to simplify the motor model. As reported in [19], Qslim is one of the fastest published simplification algorithms. Therefore, it is not surprising to see it perform so well on these models. However, we believe that the increased quality of the simplified models

TABLE 2
This Table Shows the Timing Information for Each Stage of Our Simplification Pipeline

Simplification Pipeline Timing (minutes)						
	Voxelization	Morphology	Isosurface Extraction	Smoothing	Triangle Count Reduction	Total
Blade	19.06 (78 %)	1.03 (4 %)	0.76 (3 %)	1.9 (8 %)	1.6 (7 %)	24.35 (100 %)
Motor	2.6 (29 %)	*	0.63 (7 %)	3.63 (41 %)	2.02 (23 %)	8.88 (100 %)
Al	9.7 (47 %)	*	1.7 (8 %)	3.4 (17 %)	5.7 (28 %)	20.46 (100 %)
Chair	0.9 (25 %)	0.9 (25 %)	0.4 (11 %)	0.9 (25 %)	0.5 (14 %)	3.6 (100 %)

All the timing measurements were taken on a four processor SGI Onyx with 1 Gigabyte of main memory, although only one processor was used.

TABLE 3
This Table Contains the Sizes of the Polygonal Models as They Move through the Simplification Pipeline

Dimensions of Models					
	Original Model (# faces)	Volume Representation	Closing Distance (# voxels)	Isosurface (# faces)	Simplified Model (# faces)
Blade	1,729,892	268 × 128 × 161	10	578,098	200
Motor	140,113	386 × 256 × 197	*	177,158	200
Al	13,476	107 × 256 × 276	*	489,552	200
Chair	1,087	153 × 128 × 150	10	32,750	200

Table 2 give the timing information for these model sizes.

that are obtained by performing the topology modification in the volumetric domain versus running Qslim on the original polygonal models justifies the additional time required by our simplification pipeline.

Table 3 contains the sizes of models as they go through the pipeline. There is only one free parameter when selecting the voxel resolution for a model. The other two dimensions of the volume are calculated automatically so that all the voxels that are produced are cubes. We used a closing distance of 10 for both the Turbine Blade and the Chair models. The closing distance for a model is dictated by both the resolution at which the model was scan-converted and the size of the largest hole to be closed. A closing was not needed for either the Al Capone model or the Motor model since they did not have any holes. In the case of the Chair model, the distance between the slats of the chair dictated the closing distance.

7 Conclusion and Future Work

In this paper, we have introduced a new surface simplification technique that makes use of morphological operations in the volume domain to simplify the topology of an object. Specific advantages of the simplification method include:

- Performs controlled topology modification, allowing extreme simplification,
- Accepts arbitrary collections of polygons as input,
- Produces manifold meshes as output,
- Preserves surface attributes such as color.

A benefit of converting the input polygonal models into volumes is that we can repair a number of degeneracies in polygonal models. This model repair method is simple to program and it produces clean models that are everywhere manifold.

There are several possible avenues for future research. The erosion operator eliminates thin surfaces, thus large thin parts of the model can be eliminated resulting in a large perceptual error. For this reason, we always perform dilation before erosion (which together are an opening), but we are investigating possible solutions to this issue. One area of future research would be to extend the morphological operators so that the distance parameter would vary in different parts of the surface. This would allow models that have thin structures to be processed by the erosion operator. Another future direction would be to extend other 2D image processing techniques into 3D, possibly resulting in other new and useful methods of manipulating volumetric models.

Acknowledgments

The authors thank the many people at Georgia Tech who have helped and encouraged them. This work was funded by US Office of Naval Research grant N00014-97-1-0223. The authors would like to thank the reviewers whose excellent comments lead to the current paper.

References

[1] G. Barequet and S. Kumar, "Repairing CAD Models," *Proc. IEEE Visualization '97,* pp. 363-370, Oct. 1997.

[2] G. Barequet and M. Sharir, "Filling Gaps in the Boundary of a Polyhedron," *Computer Aided Geometric Design,* vol. 12, no. 2, pp. 207-229, 1995.

[3] J.H. Bohn and W.J. Wozny, "A Topology-Based Approach for Shell-Closure," *Geometric Modeling for Product Realization,* P.R. Wilson et al., pp. 297-319, North-Holland, 1993.

[4] P. Danielsson, "Euclidean Distance Mapping," *Computer Graphics and Image Processing,* vol. 14, pp. 227-248, 1980.

[5] J. El-Sana and A. Varshney, "Controlled Simplification of Genus for Polygonal Models," *Proc. IEEE Visualization,* pp. 403-412, Aug. 1997.

[6] J. El-Sana and A. Varshney, "Topology Simplification for Polygonal Virtual Environments," *IEEE Trans. Visualization and Computer Graphics,* vol. 4, no. 2, pp. 133-144, Apr.-June 1998.

[7] C. Erikson, "Error Correction of a Large Architectural Model: The Henderson County Courthouse," Technical Report TR95-013, Dept. of Computer Science, Univ. of North Carolina at Chapel Hill, 1995.

[8] M. Garland and P.S. Heckbert, "Surface Simplification Using Quadric Error Metrics," *Proc. SIGGRAPH '97, Computer Graphics Proc., Ann. Conf. Series,* pp. 209-216, 1997.

[9] M. Garland and P.S. Heckbert, "Simplifying Surfaces with Color and Texture Using Quadric Error Metrics," *Proc. IEEE Visualization '98,* pp. 263-269, Oct. 1998.

[10] A. Gelder and J. Wilhelms, "Topological Considerations in Isosurface Generation," *IEEE Trans. Graphics,* vol. 13, no. 4, pp. 337-375, Nov. 1994.

[11] S.J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen, "The Lumigraph," *SIGGRAPH '96 Proc.,* pp. 43-54, Aug. 1996.

[12] A. Gueziec, G. Taubin, F. Lazarus, and W. Horn, "Converting Sets of Polygons to Manifold Surfaces by Cutting and Stitching," *Proc. IEEE Visualization 1998,* pp. 383-390, Oct. 1998.

[13] L. He, L. Hong, A.E. Kaufman, A. Varshney, and S. Wang, "Voxel-Based Object Simplification," *IEEE Visualization '95 Proc.,* pp. 296-303, Oct. 1995.

[14] H. Hoppe, "Progressive Meshes," *Proc. SIGGRAPH '96, Computer Graphics Proc., Ann. Conf. Series,* pp. 99-108, 1996.

[15] J. Huang, R. Yagel, V. Filippov, and Y. Kurzion, "An Accurate Method for Voxelizing Polygonal Meshes," *Proc. Symp. Volume Visualization,* pp. 119-126, Oct. 1998.

[16] A. Jain, *Fundamentals of Digital Image Processing.* Englewood Cliffs, N.J.: Prentice Hall, 1989.

[17] D. Khorramabdi, "A Walk through the Planned CS Building," Technical Report UCB/CSD 91/652, Computer Science Dept., Univ. of California at Berkeley, 1991.

[18] M. Levoy, "A Hybrid Ray Tracer for Rendering Polygon and Volume Data," *IEEE Computer Graphics and Applications,* vol. 11, no. 2, pp. 33-40, Mar. 1990.

[19] P. Lindstrom and G. Turk, "Evaluation of Memoryless Simplification," *IEEE Trans. Visualization and Computer Graphics,* vol. 5, no. 2, pp. 98-115, Apr.-June 1999.

[20] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3-D Surface Construction Algorithm," *Proc. SIGGRAPH '87, Computer Graphics,* pp. 163-169, July 1987.

[21] K.L. Low and T.-S. Tan, "Model Simplification Using Vertex-Clustering," *Proc. Interactive 3D Graphics* pp. 75-81, Apr. 1997.

[22] D. Luebke and C. Erikson, "View-Dependent Simplification of Arbitrary Polygonal Environments," *Proc. SIGGRAPH '97, Computer Graphics,* pp. 199-208, Aug. 1997.

[23] S.R. Marschner and R.J. Lobb, "An Evaluation of Reconstruction Filters for Volume Rendering," *IEEE Proc. Visualization '94,* pp. 100-107, Oct. 1994.

[24] S.M. Morvan and G.M. Fadel, "IVECS: An Interactive Virtual Environment for the Correction of .STL files," *Proc. Conf. Virtual Design,* Aug. 1996.

[25] T.M. Murali and T. Funkhouser, "Consistent Solid and Boundary Representations from Arbitrary Polygonal Data," *Proc. 1997 Symp. Interactive 3D Graphics,* pp. 155-162, Apr. 1997.

[26] J. Rossignac and P. Borrel, "Multi-Resolution 3D Approximations for Rendering Complex Scenes," *Modeling in Computer Graphics: Methods and Applications,* pp. 455-465, June 1993.

[27] J. Popovic and H. Hoppe, "Progressive Simplicial Complexes," *Proc. SIGGRAPH '97 Computer Graphics,* pp. 217-224, Aug. 1997.

[28] W.J. Schroeder, J.A. Zarge, and W.E. Lorensen, "Decimation of Triangle Meshes," Proc. SIGGRAPH '92, Computer Graphics, pp. 65-70, July 1992.
[29] W.J. Schroeder, "A Topology Modifying Progressive Decimation Algorithm," *Proc. IEEE Visualization '97,* pp. 205-212, Oct. 1997.
[30] W.J. Schroeder and W.E. Lorensen, "Implicit Modeling of Swept Surfaces and Volumes," *Proc. Visualization '94,* pp. 40-45, Oct. 1994.
[31] J. Shade, S.J. Gortler, L.-w. He, and R. Szeliski, "Layered Depth Images," *Proc. SIGGRAPH '98, Computer Graphics Proc., Ann. Conf. Series,* pp. 209-216, 1998.
[32] G.A. Taubin, "Signal Processing Approach to Fair Surface Design," *Proc. SIGGRAPH '95, Computer Graphics,* pp. 351-358, July 1995.
[33] S. Wang and A.E. Kaufman, "Volume Sampled Voxelization of Geometric Primitives," *IEEE Visualization '93 Proc.,* pp. 78-84, Oct. 1993.

Fakir S. Nooruddin graduated from the University of Iowa in 1995 with the BS degree in computer science. He took up a position as a software engineer at the Image Analysis Facility at the University of Iowa Hospital and Clinics developing volume rendering and image analysis tools. In 1996, he joined the Georgia Tech PhD program in computer science, where he worked with Dr. Greg Turk. He is currently on leave from Georgia Tech and is working for America Online Inc. in Mountain View, California.

Greg Turk received the PhD degree in computer science in 1992 from the University of North Carolina (UNC) at Chapel Hill. He was a postdoctoral researcher at Stanford University for two years, followed by two years as a research scientist at UNC Chapel Hill. He is currently an associate professor at the Georgia Institute of Technology, where he is a member of the College of Computing and the Graphics, Visualization, and Usability Center. His research interests include computer graphics, computer vision, and scientific visualization. He is a member of the IEEE.