Controlled Topology Simplification

Taosong He[‡], Lichan Hong[‡], Amitabh Varshney[‡], and Sidney Wang[*]

[‡]Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400, U.S.A.

[*]Sony-Kihara Research Center, Inc.
1-14-10 Higashigotanda
Shinagawa-ku, Tokyo, 141 Japan

Abstract

We present a simple, robust, and practical method for object simplification for applications where gradual elimination of high frequency details is desired. This is accomplished by converting an object into multi-resolution volume rasters using a controlled filtering and sampling technique. A multi-resolution triangle-mesh hierarchy can then be generated by applying the Marching Cubes algorithm. We further propose an adaptive surface generation algorithm to reduce the number of triangles generated by the standard Marching Cubes. Our method simplifies the topology of objects in a controlled fashion. In addition, at each level of detail, multi-layered meshes can be used for an efficient antialiased rendering.

1. Introduction

Interactive and realistic rendering is of importance in many applications such as scientific visualization and virtual reality. However, there has always been a conflict between the ever larger datasets and the limited rendering capabilities of graphics engines. Object simplification provides one way to reconcile scene realism with interactivity. The basic idea is to use object simplification to automatically generate a multi-resolution object hierarchy, and perform level-of-detail-based rendering. A level-of-detail-based rendering scheme uses the perceptual importance of a given object in the scene to select its appropriate level of representation in the multi-resolution object hierarchy [6, 8, 15]. Thus, higher detail representations are used when the object is perceptually more important and lower detail representations are used when the object is perceptually less significant. This method allows one to achieve higher frame update rates while maintaining good visual realism.

Most existing algorithms in the area of object simplification preserve the object topology [10, 12, 21, 35, 37, 38]. Topology in this context means the properties such as the holes, tunnels, and cavities of an object. Preservation of topology is crucial for certain applications, such as molecular surface modeling, where the presence (or absence) of interior tunnels and cavities in a molecule conveys important structural and chemical information to the biochemist. Clearly, if the target application demands topology preservation, then the simplification algorithm should adhere to it. However, if the primary goal is fast and realistic rendering, such as for virtual reality or some other time critical applications, the topology preservation criterion could stand in the way of efficient simplification.

Let us consider a virtual fly-through in a CAD model. A tiny hole on the surface of a mechanical part in this model will gradually disappear as the observer moves away from the part. However, topology-preserving simplification of this object will retain such features, thereby reducing simplification rates due to limits on the amount of geometry-simplification one can achieve while preserving topology. Another disadvantage is that rendering of a simplified object retaining high frequency details would increase image-space aliasing due to undersampling, especially in perspective viewing, thereby causing distracting effects such as flickering. On the other hand, by appropriately

simplifying the topology of the model, both simplification rates and visual realism can be increased. This idea was previously demonstrated in [8], where a chair was shown at three levels of detail with no preservation of the topology across them.

We therefore classify an object simplification process into the following two stages:
(a) geometry simplification, in which the number of geometry primitives, such as vertices, edges, and faces, is reduced;
(b) topology simplification, in which the number of holes, tunnels, cavities, as well as the number of geometry primitives, is reduced.
Depending upon the target application, these two stages should be performed either independently or jointly. For example, topology simplification itself naturally includes the reduction of geometry primitives, while geometry simplification can be applied on a topology simplified model to further reduce its complexity. However, most of the existing work for object simplification deals exclusively with geometry simplification, and the extension to topology simplification is usually difficult and complicated. The primary goal of our research is to address the topology simplification stage in a simple and robust way, and thereby also help the geometry simplification algorithms to achieve better results for certain applications.

In this paper we present a voxel-based topology simplification algorithm for the generation of multi-resolution object hierarchy, with gradual elimination of high-frequency features including, but not limited to, tiny holes, tunnels, and cavities. In our approach all formats of input objects are first converted into three-dimensional volume rasters by applying a controlled filtering and sampling technique, which is an extension of the volume sampling method proposed by Wang and Kaufman [39]. Then a surface-fitting technique such as Marching Cubes can be applied on the volume rasters to produce simplified polygon meshes. By simply adjusting the size of each voxel, thereby adjusting the resolution of the volume raster, the desired level of detail can be achieved, and consequently, a multi-resolution hierarchy of polygon meshes can be generated.

An earlier version of this work has been presented in [17]. One of the major potential problems left unresolved in [17] is that surface-fitting techniques, such as Marching Cubes, could generate a large number of redundant triangles in the regions of low surface curvature. To alleviate this problem, we adopt the idea of adaptive subdivision of volume space [30], and present in this paper an adaptive Marching Cubes algorithm. Since surface extraction from the multi-resolution volume rasters should preserve the already simplified topology of the model, our adaptive Marching Cubes algorithm guarantees that the simplified mesh is within a given bound of the mesh generated by the standard Marching Cubes.

Although our controlled filtering and sampling technique effectively eliminates the object-space aliasing in the multi-resolution volume representations, both image-space and object-space aliasings are re-introduced when the binary surface-fitting technique is applied. To solve this problem, we have developed a multi-layered triangle mesh rendering algorithm. Our idea is to smooth out the transition between the boundary of an object and empty space surrounding it by using multiple layers of triangle mesh with increasing translucency from the innermost layer to the outermost one. Unlike the earlier antialiasing techniques presented in [1, 5], the prefiltering of the projected objects in image-space is replaced here by a view-independent filtering in object space, which is performed only once in a pre-rendering stage.

The rest of the paper is organized as follows. We first summarize the previous work on object simplification in Section 2. We then present our voxel-based topology simplification in Section 3. We

discuss the adaptive Marching Cubes in Section 4, and introduce the multi-layered Marching Cubes for antialiasing in Section 5. We have implemented our algorithm and tested it on several kinds of objects, and we summarize our results in Section 6. Conclusions and some ideas on future work appear in Section 7.

2. Previous Work on Object Simplification

The last few years have seen extensive research in the area of object simplification for level-of-detail-based rendering. Thus far no single algorithm works well for all kinds of objects under all conditions. Some algorithms work best on smooth objects with no sharp edges, whereas others work best for objects that have large areas that are almost coplanar, and yet others are fine-tuned to exploit special object properties such as convexity. It is therefore natural that research on hierarchy generation has evolved around different classes of objects. These are mainly convex polytopes, polyhedral terrains, and arbitrary three-dimensional polygonal objects.

Convex Objects: Automatic simplification of convex objects is now a relatively well-understood subject, due mainly to some recent seminal papers on this topic. It has been shown that computing the minimal-facet approximation within a certain error bound is NP-hard for convex polytopes [9]. Thus algorithms for approximation of convex objects focus mainly on fast heuristics that produce approximations close to the optimal [3, 7, 28].

Polyhedral Terrains: Simplification of polyhedral terrains has been an active area of research for almost two decades because of its considerable importance to the GIS (Geographical Information System) community. It is impossible to do full justice to such a vast area in a mere section. We would, however, like to point interested readers to a recent paper on this topic by Heckbert and Garland [18], for a comprehensive survey of the field.

General Three-Dimensional Objects: Research on simplification of general (non-convex, non-terrain, possibly high genus) three-dimensional objects has spanned the entire gamut of highly local to purely global algorithms, with several approaches in between that have both local and global steps. Local algorithms work by applying a set of local rules, which primarily work under some definition of a *local neighborhood*, for simplifying an object. The local rules are iteratively applied under a set of constraints, and the algorithm terminates when it is no longer possible to apply the local rule without violating some constraint. The global algorithms optimize the simplification process over the whole object, and are not necessarily limited to the small neighborhood regions on the object.

Some of the local rules that have appeared in the literature are mentioned below:

• Vertex Deletion: Delete a vertex with its adjacent triangles and retriangulate the resulting hole. This is used by Schroeder et al. in [35] with some very good results. A generalization of this, deleting several neighboring vertices at once and retriangulating the resulting hole, is proposed by Varshney [38].

• Vertex Collapsing: Merge all the vertices that satisfy a given criterion into one vertex. This is used in conjunction with a global grid by Rossignac and Borrel [32].

• Edge Collapsing: Merge the two vertices of an edge into one, thereby deleting the two adjacent triangles of the edge. This is used as a subroutine in the mesh optimization algorithm by Hoppe et al. [21].

• Polygon Merging: Merge the adjacent coplanar polygons into larger polygons [20].

Some of the global rules that have been used are:

• Uniform Distribution of Fewer Vertices: In a re-tiling approach to simplification outlined by Turk [37], the program first distributes a given number of vertices over the surface and then repositions them based on a global repulsion method to uniformly spread them as a function of the curvature. These new vertices are then retriangulated and the old vertices deleted to obtain the approximation mesh.

• Minimization of Energy Function: Hoppe et al. [21] globally optimize the energy function representing (a) the sum of squared distances from the mesh, (b) the number of vertices, and (c) the edge lengths over the vertices of the newer mesh. The overall optimization procedure alternates between local and global optimization steps.

• Minimization by Set Partitioning: Varshney [38] uses a greedy set-partitioning-based minimization approach to reducing the number of triangles. His method can relate the quality of the approximation produced to the optimal for the same ε tolerance.

• Wavelets: An interesting solution to the problem of polygonal simplification by using wavelets is presented in [12, 26], where arbitrary polygonal meshes are first subdivided into patches with *subdivision connectivity* and then multi-resolution wavelet analysis is used over each patch.

The issue of preservation or simplification of topology is independent of whether an algorithm uses local rules, or global rules, or both to simplify. With the exception of Rossignac and Borrel [32], all other papers cited above preserve the topology of the input object. Preservation of input topology is mathematically elegant and aesthetically pleasing. However, if interactivity is the bottomline, as is often the case in interactive three-dimensional graphics and visualization applications, topology can and should be sacrificed if the topology simplification (a) does not directly impact the application underlying the visualization and (b) does not decrease visual realism. Both of these goals are easier to achieve if the simplification of the topology is finely *controlled* and has a sound mathematical basis. In the next section we outline our approach, which has these properties and is global in nature.

3. Controlled Topology Simplification

In order to better motivate our approach to object simplification, we turn to the classic example of rendering a tilted checkerboard, in which the top of the checkerboard is placed further away from the viewer than the bottom of the board. Once rendered, the Moire patterns are especially noticeable at the top of the image, where too many details from the checkerboard are forced into too few image pixels. This is mainly due to the pyramidal viewing frustum of perspective projection. The same problem occurs when highly detailed objects far from the viewer are rendered. Therefore, our goals for object simplification are twofold. First, we would like to avoid the Moire patterns by gradually eliminating detailed features of an object as it moves away from the viewer. Second, as in the case of most existing level-of-detail algorithms, we would like to increase the frame rate by establishing a multi-resolution object representation, and using simplified models for distant objects.

A flow diagram illustrating our overall object simplification algorithm is outlined in Fig. 1. The algorithm starts by first overlaying the object with a three-dimensional grid and applying a three-dimensional low-pass filter at each grid point. A three-dimensional volume raster data-structure is used to store these filtered grid-point values. Once the filtering and sampling process is completed, a reconstruction process is employed to generate the detail-eliminated object from the set of filtered sample points represented in the volume raster. In this section, we first explain the controlled filtering and sampling process, then discuss the establishment of the hierarchical representation and the smooth transition between levels of detail. The reconstruction process will be presented in Section 4.

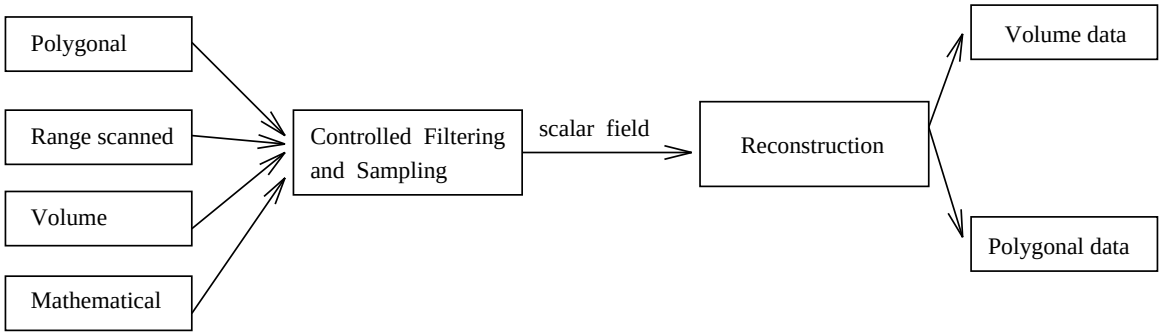


Fig. 1: *Pipeline for controlled topology simplification.*

3.1. Controlled filtering and sampling

To simplify the topology in a controlled fashion, we adopt a signal-processing approach to object detail-elimination by low-pass filtering the object to gradually remove the high frequencies (i.e., detailed features) from the object. The class of input objects that our algorithm can accept and process includes polygonal meshes, volume datasets, objects derived from range-scanners, and algebraic mathematical functions such as fractals. Our algorithm is backed by a sound and elegant mathematical framework of sampling and filtering theory. In fact, a similar sampling and filtering principle has been used extensively in the image processing communities to reduce noise and smooth sharp features in 2D images. Wang and Kaufman [39] generalized the concept into 3D to remove aliasing in volume-based modeling of geometric objects. Our approach utilizes their volume sampling approach, and extends it to incorporate more precise control over the filtering and sampling process. In the following discussion we assume readers are familiar with the basic concepts of sampling and filtering theory. A good reference text is [43].

Fourier analysis tells us that a signal's (or an object's) shape is determined by its frequency spectrum. The more details the signal contains, the richer it is in high-frequency components of its spectrum. Therefore, to gradually eliminate the detail features from an object, we create a smoother signal by removing the offending high frequencies from the original signal. This process is known as low-pass filtering, or band-limiting the signal, and is described mathematically in frequency domain as:

$$FT(f_{new}) = FT(f_{orig}) \cdot H(\nu) \tag{1}$$

where

$$H(\nu) = \begin{cases} 1 & -k \le \nu \le k \\ 0 & otherwise \end{cases}. \tag{2}$$

The more high frequencies we remove (i.e., the smaller the k), the more details that are eliminated from the object. Since the multiplication in the frequency domain corresponds exactly to convolution in the spatial domain, the equations can be rewritten to operate in the spatial domain as:

$$f_{new} = f_{orig} * sinc \tag{3}$$

where $*$ is the convolution operator, and *sinc* is the ideal low-pass filter. Although analytic evaluation of Equation 3 is sometimes possible for objects which are represented by algebraic mathematical functions, for general mathematical functions, polygon meshes, or volume datasets, an analytical solution either does not exist or is too expensive to be calculated. For such cases, a discrete approximation must be used. To minimize the discrete approximation error, the original object is kept in its continuous form while the three-dimensional filter is divided into a number of bins, each having a precomputed filter weight. Note that the resolutions of the bins should be finer than that of the sampling grid to achieve good approximation. These weights are computed by evaluating the filter function at these discrete bin positions and multiplying them by a normalization factor to ensure that the sum of all weights equals unity. Thus, during convolution, a lookup table is used to obtain the corresponding set of weights, which is then applied to the intersected region between the filter kernel and the object. That is, for a grid point (i, j, k) in the volume raster, the resulting filtered density $f(i, j, k)$ is calculated as:

$$f(i, j, k) = \int\int\int h(i-\alpha, j-\beta, k-\gamma)\, S(\alpha, \beta, \gamma)\, d\gamma\, d\beta\, d\alpha \tag{4}$$

where h is a low-pass filter of choice and $S(\alpha, \beta, \gamma)$ is a binary function defined as:

$$S(\alpha, \beta, \gamma) = \begin{cases} 1 & \text{if point } (\alpha, \beta, \gamma) \in \text{object} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Thus an important criterion of our input object is that for a given point (α, β, γ), it can be determined whether that point is inside or outside of the object. For example, only those polygonal meshes that form the boundary of a solid can be treated by the algorithm.

The issues that still remain to be addressed are the determination of the appropriate resolution of the sampling volume raster and the appropriate size of the filter support. From Shannon's sampling theorem, these two variables are directly related to each other. That is, the volume raster must be sampled at a frequency that is greater than twice f_h, the highest frequency component in the signal. This lower bound on the sampling rate is known as the Nyquist rate, or Nyquist frequency *NF*. Suppose that we have a volume raster consisting of $X \times Y \times Z$ sampling resolution, which is used to represent a rectangular volume region of $p \times q \times r$ $unit^3$ of space; then the Nyquist frequency and the sampling frequency f_v of the volume raster are:

$$NF = f_v = \left\{ f_{v_x},\ f_{v_y},\ f_{v_z} \right\} = \left\{ \frac{X}{p}, \frac{Y}{q}, \frac{Z}{r} \right\} \tag{6}$$

Hence, ideally, the cut-off frequency f_g of the low-pass filter must be set to $NF/2$ in order to filter out all offending high frequencies. However, in practice, since the ideal low-pass filter is rarely used, f_h is usually set far less than $NF/2$. In our experiments, approximate filters such as Gaussian filters and hyper-cone filters are often employed [17]. The use of these non-ideal filters results in a combination of frequency leakages and non-unity gains. Fortunately, substantial improvement can be made by filtering out more high frequencies from the objects to allow some error margins caused by the non-ideal filters. This is achieved by the following calculation of f_h:

$$f_h = \frac{NF}{2} \cdot error_{non\_ideal} \tag{7}$$

where $0 < error_{non\_ideal} < 1$. Generally, depending on the type of non-ideal filter used, an $error_{non\_ideal}$

factor between 0.5 and 0.85 achieves satisfactory results. More in-depth discussions on the selection and error estimation of three-dimensional filters can be found in [4, 27]. Given the cut-off frequency f_g, the size of the corresponding low-pass filter support is determined accordingly. High f_h, therefore high resolution of the volume raster, corresponds to small support in the spatial domain, and vice versa.

3.2. Multi-resolution hierarchy

Now that we have established the direct correspondence between the size of the filter support and the resolution of the volume raster, a hierarchical level-of-detail object representation can be easily constructed by controlling the amount of high frequency removed from the spectrum, as one goes from one level of the hierarchy to another. In frequency domain, that is,

$$FT(f_i) = FT(f_{orig}) \cdot H_i(\nu), \qquad 0 \le i \le L \tag{8}$$

and

$$H_i(\nu) = \begin{cases} 1 & -k - \delta \cdot (L - i) \le \nu \le k + \delta \cdot (L - i) \\ 0 & otherwise \end{cases} \tag{9}$$

where f_i represents the i-th level of the level-of-detail hierarchy and L represents the total number of levels.

The base of the proposed hierarchy contains the most detailed and the highest resolution version of the object, and the top contains the blurriest and lowest resolution version of the object. Thus, during volume raster hierarchy construction, as one moves up the hierarchy, the sampling resolution of the volume raster decreases. Consequently, a low-pass filter with wider support is applied. Finally, if desired, a surface-fitting technique can be used to reconstruct a polygon mesh model for each level of the volume raster hierarchy. During rendering, the appropriate level of the hierarchy is selected for each object in the scene. The heuristic that we have used is that the footprint of each filtered sample point covers approximately one and a half times the area of a pixel.

Furthermore, in order to reduce temporal aliasing during animation, smooth interpolation between two adjacent resolution meshes should be generated, which is generally a non-trivial task. However, in our algorithm, an arbitrary integer resolution volume raster can be generated by adjusting the low-pass filter support. It is also straightforward and efficient to directly interpolate between two adjacent resolution volume rasters to generate an in-between resolution volume raster. This is achieved by first linearly interpolating the resolution of the volume rasters at adjacent levels. Then the density at a grid point of the in-between volume raster is decided by linearly interpolating the densities at the corresponding positions of the two adjacent resolution volume rasters. A two-dimensional example of this process is demonstrated in Fig. 2.

The topology simplification algorithm presented above is simple, robust, and widely applicable. By continuously adjusting the filter support, the user can control the elimination of appropriate amount of high frequency in the model. The desired passband for filtering can be precisely calculated according to the distance from the model to the eyepoint. At first glance, it might seem somewhat similar to the clustering scheme [32] or the three-dimensional "mip-map" approach [24, 33]. However, our approach follows a control-based filtering for gradual elimination of high frequencies, which is different from the locality-based clustering of geometry as presented in [32]. As for the three-dimensional "mip-map" approach, every level of the hierarchy is formed by averaging several voxels

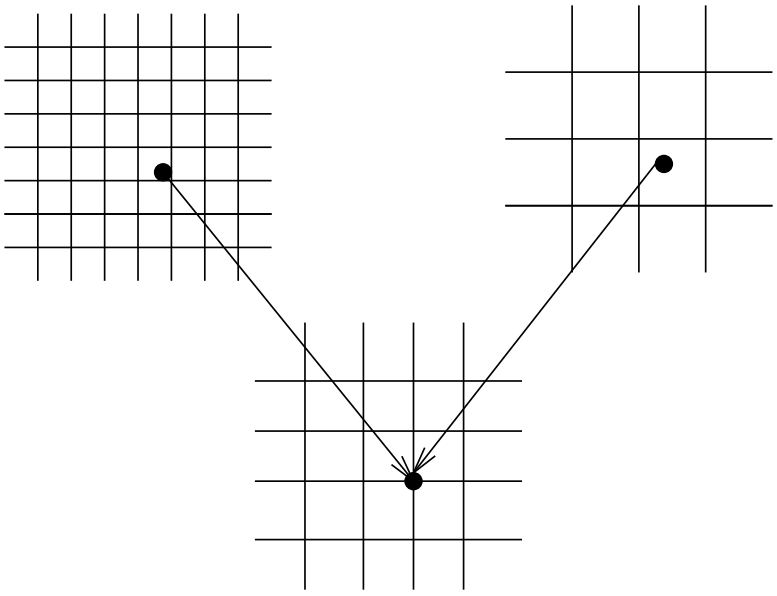


Fig. 2: *Interpolation between two adjacent resolution volume rasters.*

from the previous level. In our approach, every level of the volume raster hierarchy is created by convolving the original object with a low-pass filter of an appropriate support, whose size can theoretically be any positive real number. Thus, errors caused by a non-ideal filter do not propagate and accumulate from level to level. Furthermore, the sampling resolution of our volume raster hierarchy need not be the same for all three axes nor even be required to be a power of two.

4. Surface Reconstruction

Once the filtered value for each volume raster grid point is generated, surface-fitting techniques can be used to reconstruct the isodensity surfaces, if desired. Alternatively, if a polygonal model is not required for the simplification result, one can delay the reconstruction process until rendering, instead of reconstructing the isodensity surface using polygonal elements. In other words, if the direct volume rendering [23, 40] technique is employed to render the volume raster of filtered values, then the reconstruction process is done during rendering, without resorting to intermediate polygonal representation. In this paper, we focus on the polygon reconstruction.

Marching Cubes, originally proposed by Lorensen and Cline [25], has been considered the standard approach to surface extraction from a volume raster of scalar values. In this algorithm, an isodensity surface is approximated by determining its intersections with edges of every voxel in the volume raster. Up to five triangles are used to approximate the surface within a voxel, depending on the configurations of the voxel with respect to an isodensity value. One advantage of Marching Cubes is that it can be efficiently implemented using a precomputed lookup table for the various arrangements of surface-voxel intersections. However, despite its extensive applications, the original algorithm proposed by Lorensen and Cline [25] has some particular problems, in turn provoking substantial research aimed at the solutions. One of the problems is that the 15 basic configurations proposed in [25] are incomplete, and could generate topology inconsistent surfaces due to the ambiguities [11]. Several solutions have been proposed to add additional configurations [31, 41].

Recently, Schroeder, Martin, and Lorensen have published the Visualization Toolkit [36]. It contains an implementation of a topology-consistent Marching Cubes based on a complete set of 256 configurations. Since this implementation is simple and available, it has been adopted in this paper. Another problem of Marching Cubes is that the time of computation spent for empty voxels with no surface intersection could be considerable. It can be solved by avoiding the visiting and testing of empty regions [42].

For the purpose of object simplification, the number of triangles generated by Marching Cubes is particularly important. Since the maximum size of the triangles is limited by the regular grid spacing of the volume raster, there could be excessive fragmentation of the output data even in the area of low curvature. The solutions proposed to solve this problem can be classified into either filter-based or adaptive techniques. A filter-based technique starts with a large number of primitives and removes or replaces them to reduce the model size. For example, Montani et al. [29] discretize the intersection points between the surface and the edges of voxel and merge the coplanar facets. Schroeder et al. [35] have proposed a decimation algorithm based on multiple filtering passes and vertex deletion. It analyzes the geometry and topology of a triangle mesh locally and recursively removes vertices that pass a minimum distance or curvature-angle criterion. The advantage of this approach is that any level of reduction can be obtained, on the condition that a sufficiently coarse approximation threshold is set. Kalvin and Taylor [22] have proposed a *superface* algorithm by merging faces. It guarantees a bounded approximation and can be applied on any polyhedral mesh that is a valid manifold. On the other hand, adaptive techniques produce more primitives in selected areas, such as an area with highly detailed features. For example, Schmitt [34] starts with a rough bi-cubic patch approximation to sample data, and then subdivides those patches that are not sufficiently close to the underlying samples. Adaptive techniques have been used for terrains [13], implicit modeling [2], and general polygon meshes [10].

As mentioned in the introduction, the surface-fitting technique used for our algorithm must preserve the topology of the model, since the topology simplification has been appropriately achieved in the stage of controlled filtering and sampling. One method is to first generate a triangle mesh using the standard Marching Cubes, and then apply the existing topology preserving geometry simplification algorithm on the mesh. As an alternative, we adopt the adaptive idea and propose a simple algorithm based directly on Marching Cubes. The basic idea is to adapt the size of the generated triangles and hence their number to the shape of the isodensity surface. To achieve this, Muller and Stark [30] have proposed a Splitting-box algorithm. Our *adaptive Marching Cubes* is based on Splitting-box, but with some major improvements. In this section, we first introduce the Splitting-box algorithm, following the description in [30], and then discuss the difference between it and our algorithm.

4.1. Splitting-box algorithm

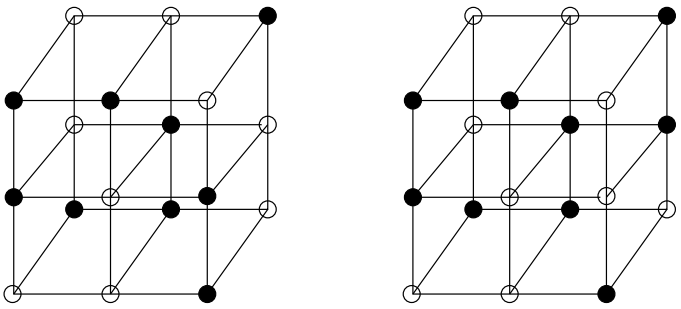
The input of the Splitting-box algorithm, like Marching Cubes, consists of a regular 3D grid of scalar values and an isodensity value for the surface. A vertex of the grid is called *black* if its value is greater than or equal to the isodensity, and *white* otherwise. A box is a rectangular parallelepiped, whose edges are induced by linear sequences of vertices of the grid. For Marching Cubes, the assumption is that the box edge length is always 2, and the surface has one and only one intersection at the box edges whose vertices are of different colors. For Splitting-box, a box can have longer edges, but the second assumption still holds. An edge is called *MC* if the color changes at most once along its grid vertex sequences. A face is called *MC* if its four edges are *MC* edges, and a box is *MC* if its six

faces are *MC* faces., An example of an *MC* box is shown in Fig. 3a.

Splitting-box starts with the box given by the input grid. This box is bisected perpendicular to its longest edge into two sub-boxes. The process of bisection is recursively performed until a 2×2×2 box is reached. Meanwhile, the bisection is postponed for the boxes arising during the bisection process if they are recognized *MC*. Instead, triangle chains are generated for such boxes, according to the rules of Marching Cubes configurations. One important point here is that a triangle vertex on an *MC* edge is interpolated between the pair of consecutive vertices of different colors on the edge similar to Marching Cubes. Thus, the vertex coincides exactly with the one generated by Marching Cubes. After the generation of the triangle chains for an *MC* box, the bisection is continued to check the quality of approximation of the triangle chains with respect to the true triangle chains both on the faces of and inside the *MC* box. If the approximation is acceptable, it will be part of the output. Otherwise, the chain is discarded, and a new approximation is tested in the sub-boxes. A satisfactory approximation is generated, at worst, at the level of a 2×2×2 box.

The purpose of checking the quality of the approximation of the triangle chains is to preserve the topology. With the bisection of an *MC* box B into two sub-boxes B_1 and B_2, if one of B_1 and B_2 is not *MC*, then essentially the approximation of B will not be acceptable. If both B_1 and B_2 are *MC* boxes, then a satisfactory approximation of B must satisfy the following two criteria. First, the intersections between the chains in B and the new edges on the common face of B_1 and B_2 must lie between a pair of consecutive vertices with different colors. If this condition is satisfied, the respective triangle vertices in B_1 and B_2 are replaced by the intersection point. Second, the triangle chain of B must be geometrically coincident with those of B_1 and B_2. Using these criteria, Splitting-box preserves the exact separation of black and white vertices, and guarantees that the topology of the surface coincides with and is not more than the sampling distance apart from the Marching Cubes solution.

Splitting-box provides a simple and practical framework for reducing the number of triangles generated by Marching Cubes. One of the problems of this approach, however, is that the algorithm achieves only a fixed-bound approximation. In other words, the Splitting-box mesh is always within sampling distance of the Marching Cubes mesh, and this bound can not be changed by the users. This is due to the requirement to preserve the exact separation of black and white vertices. This requirement also limits the possible reduction of polygons, even with the sampling distance bound.



(a) (b)

Fig. 3: *(a) An MC box for Splitting Box. (b) An AMC box for adaptive Marching Cubes*.

4.2. Adaptive Marching Cubes

To solve this problem, we propose an adaptive Marching Cubes algorithm. By slightly change the definition of *MC* of Splitting-box, we propose the concept of *AMC*. The definition of *AMC* edges and *AMC* faces are the same as of Splitting-box. However, a box in a 3D grid is *AMC* if and only if all the edges induced by linear sequences of grid vertices on the face and inside the box are *AMC* (Fig. 3b). In other words, when we define an *AMC* box, we not only consider the faces, but also the interior. Therefore, together with the quality checking process discussed below, we guarantee that a satisfactory approximation in an *AMC* box does not affect the topology of the model.

The adaptive Marching Cubes algorithm follows a similar bisection process to Splitting-box. The process is recursively performed, but postponed to generate the triangle chain according to Marching Cubes configurations when a box is recognized *AMC*. The quality of the triangle chain approximation of this *AMC* box is then checked, and bisection is continued if the approximation is not satisfactory.

However, there are several important differences between the two algorithms. First, for Splitting-box, a box is always bisected perpendicular to its longest edge. In some situations, this could cause unnecessary bisections. For example, if all the edges along a certain axis are *AMC* in a non-*AMC* box, bisecting along this axis would still generate *non-AMC* boxes. On the other hand, for adaptive Marching Cubes, only an *AMC* box is still bisected perpendicular to its longest edges. For non-*AMC*, we first find those axes along which there is at least one non-*AMC* edge, then bisect perpendicular to the longest edges along those axes. An example is shown in Fig. 4, where more than three bisections must be performed to make all the sub-boxes *AMC* using the Splitting-box approach, while only one bisection is needed perpendicular to the shortest edge along the Z axis using the adaptive Marching Cubes approach.

The major difference between the two algorithms lies in the quality checking process. Given a user-specified bound ε, the goal of our algorithm is to satisfy the following conditions:
(1) The set of adaptive Marching Cubes generated mesh vertices is a subset of the set of Marching Cubes generated mesh vertices.
(2) The topology of the standard Marching Cubes generated mesh is preserved in the *adaptive Marching Cubes* generated mesh.
(3) All the vertices of the standard Marching Cubes generated mesh are within ε distance of the

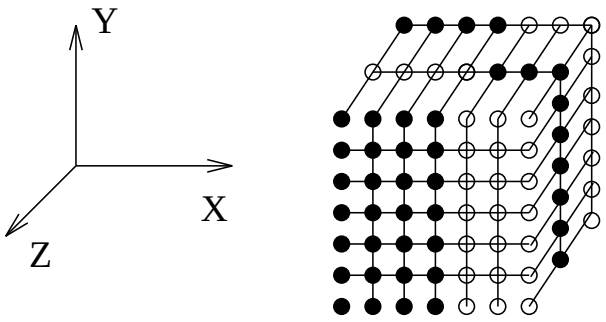


Fig. 4: *Bisecting of a non-AMC Box.*

adaptive Marching Cubes generated mesh.

The first condition is met through the method of generating the approximating triangle chain in an *AMC* box, and the second condition is satisfied by approximating only the mesh in an *AMC* box. To satisfy the third condition, for each *AMC* box *B*, we test the distance between all the Marching Cubes vertices to the approximating chain. To check whether the distance between a point and a triangle is within a certain bound, we conservatively approximate the distance by calculating the distance between the point and the triangle along X, Y, and Z axes. Since quality checking is always performed between the approximating chain and the Marching Cubes vertices, it can be achieved without the recursive bisection of Splitting-box.

A potential problem for adaptive approximation is the cracks generated on the common face between different levels of the hierarchy. An example of the possible cracks is shown in Fig. 5a. To make things clear, we present only triangles related to the cracks in Fig. 5. For Splitting-box, the crack is first detected, then eliminated by exploiting the restriction that the intersections between the approximation chains in an *MC* box and the common faces of the sub-boxes lie between a pair of consecutive vertices with different colors. Since this restriction is released for adaptive Marching Cubes, we develop a simple *stitching* algorithm for the elimination of the cracks. The basic idea is to first find those potential crack edges on the common faces, then retriangulate them to stitch the cracks. The edge on a common face between B_1 and B_2 is defined as *good* if it appears in more than one triangle in the union of B_1 chain and B_2 chain, and *crack* otherwise. The retriangulation is performed among the triangles with at least one crack edge. An example of such retriangulation is shown in Fig. 5b, where one triangle in the coarse resolution *AMC* box on the right is split into two triangles. Notice that many additional triangles could be generated using this simple stitching algorithm. Another possible retriangulation with fewer triangles but more complicated implementations is shown in Fig. 5c, where the position of the corresponding vertices of the triangles in the fine resolution *AMC* box on the left is moved.

5. Multi-Layered Marching Cubes Rendering

The controlled filtering and sampling technique discussed in Section 3 effectively eliminates the high frequencies above the Nyquist frequency in the multi-resolution volume rasters. They can be appropriately rendered through antialiased volume rendering. However, as discussed above, surface-fitting techniques, such as our adaptive Marching Cubes, are sometimes needed to generate an isodensity surface from the volume representations. Generally, these techniques apply a binary surface classification to extract an isodensity surface from the 3D grid. Although the isodensity surface is considered to be good from the point of view of modeling, it does introduce infinitely high frequencies. Since these frequencies cannot be fully represented by a discrete image, they can cause image-space aliasing. As an alternative to the commonly used hardware-supported antialiasing for rendering, we have developed a multi-layered Marching Cubes antialiased rendering algorithm. This approach takes advantage of the low-pass filtering applied during the controlled filtering and sampling stage. The non-binary surface classifier that we have used permits surfaces to be associated with a continuous range of densities, thereby allowing a smooth transition from the object-boundary to the empty space. Another motivation for this approach is to more appropriately represent the low-pass-filtered models using polygon meshes. Since a simplified model is represented in a volume raster with densities ranging continuously from 0 to 1, a single layer of surface with one isodensity sometimes cannot always produce a good simplified effect, especially when the object is far away and a very low

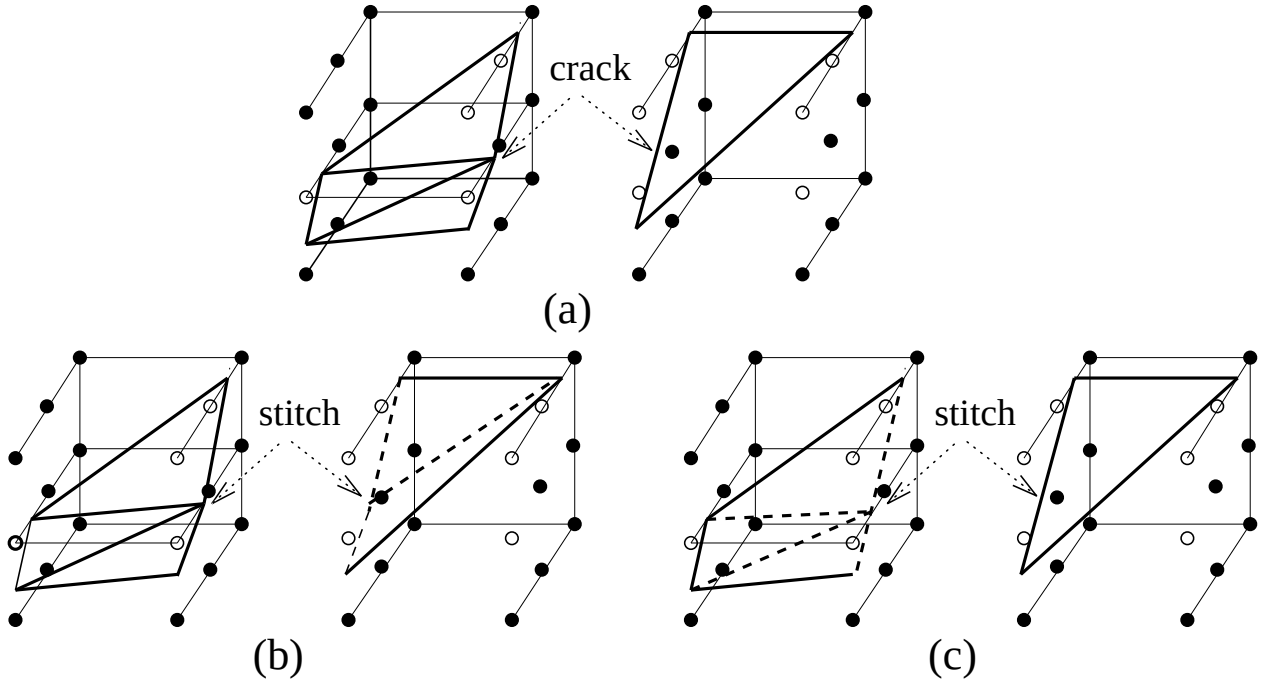


Fig. 5: *Crack Stitching: (a) Crack, (b, c) Examples of solutions.*

resolution volume raster is used. For these situations, object-space aliasing introduced by surface-fitting needs to be reduced.

Besides our work in [17], the idea of utilizing multiple layers of triangle meshes generated by Marching Cubes has been independently proposed by Heidrich et al. [19] for the purpose of interactive maximum projection. However, their algorithm is not concerned about the composition of semi-transparent layers. Fujishiro et al. [14] have generalized Marching Cubes to handle an interval volume with isodensities falling into a close interval $[\alpha, \beta]$. Their algorithm generates polyhedra, instead of triangles, but still cannot handle semi-transparent interval volumes. Guo [16] has proposed using

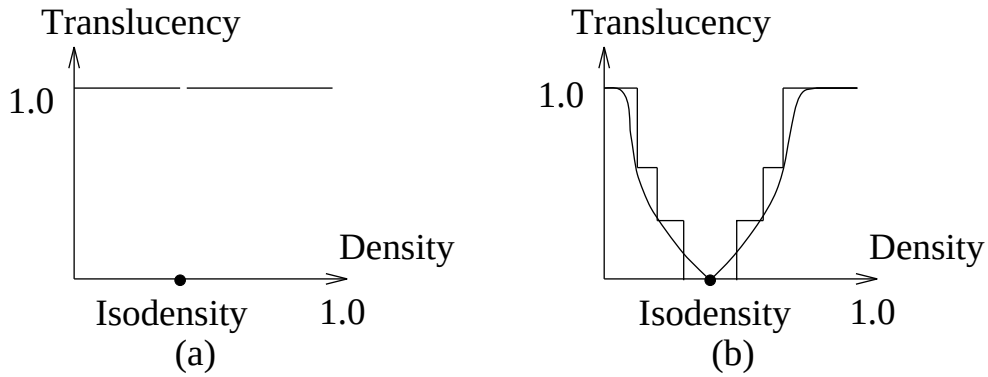


Fig. 6: *(a) Binary surface classification. (b) Continuous surface classification and discrete approximation.*

α-shape to approximate the interval volumes. The advantage of his approach is that the α-shape can be rendered as RGBA clouds, thereby producing a correct semi-transparent effect. However, too many tetrahedra would be generated if the interval $[\alpha, \beta]$ is large.

The basic idea of our algorithm is to discretely approximate the surface boundary by generating several layers of triangle meshes using Marching Cubes, with increasing translucency from the innermost layer to the outermost one (Fig. 6b). Then, by appropriately compositing these layers of triangle meshes using hardware-assisted blending, a high frame rate of antialiased rendering can be achieved. The accuracy of the discrete approximation of the continuous surface classification is determined by the number of mesh layers used and their corresponding isodensities. Better approximation can be achieved with more layers, at the cost of increased storage space and rendering time. The minimum number of layers needed for the approximation to be within a user-specified error bound depends upon both the low-pass filter employed to generate the volume raster and the geometry of the original polygon mesh (e.g., Fig. 7). However, it can be computed approximately by the following method.

First, it is assumed that the translucency of a point with a certain density d is

$$1 - \frac{d}{m} \tag{10}$$

where m is the maximum isodensity value associated with the innermost layer M of the multi-layered surfaces. Therefore, the problem of approximating a translucency function is simplified into the approximation of a density function. Then, by assuming that the density of a point is decided solely by its distance to M, we can approximate the density at every point. Mathematically, centering the low-pass filter h with support R at a point with distance r from M, and assuming the filter intersects a planar surface (Fig. 7a), the density of this point is:

$$d(r) = \int_r^R \int_{-\sqrt{R^2-\alpha^2}}^{\sqrt{R^2-\alpha^2}} \int_{-\sqrt{R^2-\alpha^2-\beta^2}}^{\sqrt{R^2-\alpha^2-\beta^2}} h(\alpha, \beta, \gamma) d\gamma\, d\beta\, d\alpha \tag{11}$$

Therefore, given an error bound ε, the minimum number of layers needed and their corresponding isodensities are decided by a piecewise constant function p with a minimum number of segments which satisfies:

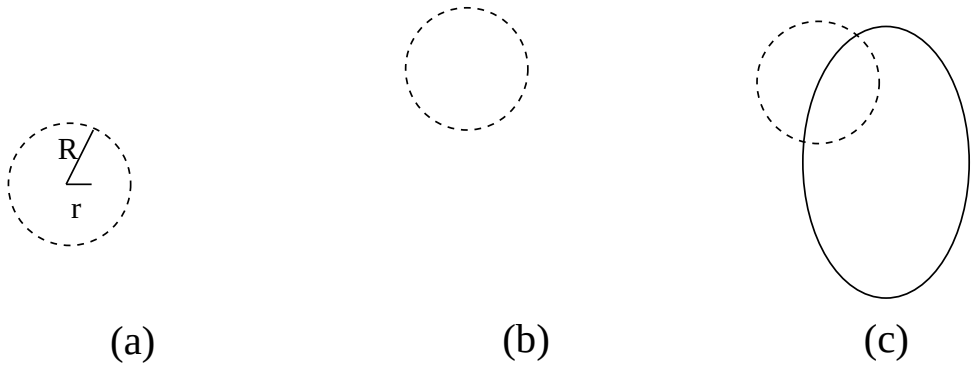


Fig. 7: *Different intersections between a surface (solid) and the filter (dashed).*

$$\int_0^R |p(x) - d(x)|\, dx \le \varepsilon \tag{12}$$

The optimal piecewise constant function p might not be analytically derivable for certain filters. However, by discretely approximating d, and using the heuristic that more layers should be placed where the change of the function d is high, sub-optimal p can be recursively generated. The number of layers of triangle meshes is then equal to the number of segments in the function p, and the isodensities of the meshes are the corresponding constants of that function. In addition, the corresponding translucency can be computed using Equation 10.

In order to generate the correct composition of semi-transparent meshes, the triangles should be projected in a back-to-front or a front-to-back order, either of which generally involves an expensive sorting process. A nice property of a Marching Cubes generated mesh is that it is associated with a volume raster. As a result, sorting can be accomplished by traversing only the surface-intersected voxels in a slice-by-slice fashion. However, because of the generation of multiple layer meshes with different isodensities, when adaptive Marching Cubes is applied, the sorting is not always possible. Projection of multiple objects is even more complicated. One simple solution is to perform the sorting on bounding boxes of the volume rasters associated with the objects. A more accurate and still efficient sorting algorithm takes advantage of the volume rasters associated with the meshes of these objects, since intersections among the regularly partitioned volume rasters are easy to compute. Therefore, all the surface-intersected voxels can be rapidly traversed in an almost correct order.

6. Results

We have implemented our controlled topology simplification algorithm and applied it on a variety of objects. The results have been very encouraging and are summarized below. All the experiments were conducted on a Silicon Graphics Onyx VTX, equipped with two 100Mhz R4400 processors and 128MB of RAM. Only one of the processors was used.

An interesting feature of our voxel-based topology simplification algorithm is that it can simplify not only individual objects but also collections of objects. This is achieved by filtering and sampling the object cluster into one volume raster hierarchy. Fig. 8 illustrates the triangle-mesh hierarchy of a fractal ellipsoid-flake with 820 ellipsoids in the original model. The original triangle mesh, shown in Fig. 8a, is reconstructed from a high resolution volume raster to preserve the details. By convolving the original fractal functions with Gaussian filters with different radius supports, we decrease the resolution of volume rasters accordingly, and the resulting number of triangles in the simplified mesh is reduced. The simplification results are presented in Table 1, with the index specifying the corresponding image in Fig. 8. The discrete approximations of the applied Gaussian filters are at resolution $11 \times 11 \times 11$. The surfaces have been reconstructed using standard Marching Cubes from multi-resolution volume rasters using an isodensity of 0.5 on a normalized scale of 0 to 1. The running time is from several seconds to several minutes.

To further reduce the number of triangles in the simplified model, we have applied our adaptive Marching Cubes on the multi-resolution ellipsoid-flake volume rasters with different approximation bounds. Fig. 9 illustrates the results of applying this geometry simplification algorithm on the volume raster with 200×200×200 resolution. The approximation bound is 0.0 in Fig. 9a, 0.5 in Fig. 9b, 1.0 in Fig. 9c, and 2.0 in Fig. 9d, and the running times are 190sec, 260sec, 190sec, and 163sec, respectively.

Table 1: *Simplification of a fractal ellipsoid-flake*

Index	Resolution	Triangles
a	200×200×200	320455
b	100×100×100	64687
c	50×50×50	13589
d	30×30×30	3746
e	15×15×15	640
f	5×5×5	8

The simplification results of all the multi-resolution ellipsoid-flake volume rasters are presented in Table 2. In this example we use the simple stitching algorithm as presented in Fig. 5b.

Fig. 10 demonstrates a mechanical part generated by CSG operations using volume-sampled voxelized primitives [39]. The level-of-detail meshes established by applying Gaussian filters of different radius supports are presented in Table 3, with the index specifying the corresponding image in Fig. 10. The surfaces have been reconstructed with Marching Cubes from multi-resolution volume rasters using an isodensity of 0.5 on a normalized scale of 0 to 1. These images have been rendered using a solid steel texture. From these results, it can be seen that our algorithm provides a controlled way to gradually reduce the genus and small features. Fig. 11 presents the effect of the simplification on an assembly of identical mechanical parts at different resolutions, as shown in Fig. 10. The selection of resolution depends on the distance of the parts from the viewpoint. To reduce the temporal aliasing, we apply the smooth interpolation algorithm as described in Section 3. The effect is illustrated in Fig. 12, where the interpolation is performed between the 100×100×60 volume raster (top left) and 50×50×30 volume raster (bottom right). It should be noted that to generate the polygon mesh from interpolated volume rasters using Marching Cubes, only those voxels which might contain surfaces are examined. An interpolated voxel might contain a surface only if at least one of the corresponding regions in the two volume rasters contains a surface, or exactly one of the corresponding regions in the two adjacent resolution volume rasters is inside the surface. Such voxels can be efficiently generated since the regions in the two volume rasters satisfying above conditions can be pre-computed.

Table 2: *Triangle number of a multi-resolution fractal ellipsoid-flake mesh using adaptive Marching Cubes*

Bound	0.0	0.5	1.0	2.0
Resolution				
200×200×200	321400	108103	79338	71364
100×100×100	64460	22519	16117	13267
50×50×50	13348	6500	4740	4199
30×30×30	3692	1813	1175	1033
15×15×15	620	314	217	205
5×5×5	4	4	4	4

We have also applied our algorithm on volumetric datasets. Fig. 13 presents the result of simplifying the head and neck of the Visible Man fresh CT data [44]. The data is first aligned and downsampled from 512×512×217×16 bits to 256×256×117×8 bits. Then, the simplified meshes reconstructed using Marching Cubes are presented in Table 4, with the index specifying the corresponding images in Fig. 13.

Unlike the volume raster generated from a solid object, a sampled or simulated volumetric dataset generally does not have a well-defined surface. However, for a given point, it is still possible to test whether this point is inside or outside the surface by tri-linearly interpolating the point value from the neighboring eight vertices and comparing it to the isodensity, and therefore Equations 4 and 5 can still be applied. Another method of simplifying volumetric datasets without well-defined surfaces is to directly apply the reconstruction filters with different radius supports to the original volumes. The application of 3D reconstruction filter for volumetric datasets has been previously discussed for volume rendering [40].

We also tested our adaptive Marching Cubes algorithm on the multi-resolution head and neck volume rasters with different approximation bounds. Fig. 14 illustrates the results of applying the algorithm on the original model with 256×256×225 resolution. The approximation bound is 0.0 in Fig. 14a, 0.5 in Fig. 14b, 1.0 in Fig. 14c, and 2.0 in Fig. 14d, and the running times are 223sec, 416sec, 312sec, and 280sec, respectively. The simplification results of all the multi-resolution volume rasters are presented in Table 5. In this example, we apply the complicated stitching algorithm as presented in Fig. 5c.

Table 3: *Simplification of a CSG mechanical part*

Index	Resolution	Triangles
a	200×200×120	271504
b	100×100×60	64344
c	50×50×30	13292
d	40×40×24	8660
e	20×20×12	1508
f	5×5×3	88

Table 4: *Simplification of the head and neck of Visible Man Fresh CT*

Index	Resolution	Triangles
a	256×256×117	334564
b	192×192×88	180996
c	128×128×59	76088
d	64×64×30	16852
e	32×32×15	3284
f	16×16×8	568

Table 5: *Triangle number of a multi-resolution head and neck mesh using adaptive Marching Cubes*

Bound	0.0	0.5	1.0	2.0
Resolution				
256×256×256	333259	102253	75739	68859
192×192×88	180310	64181	47357	42853
128×128×59	75790	30739	23278	21452
64×64×30	16784	7935	6254	5820
32×32×15	3236	1654	1326	1273
16×16×8	536	276	248	242

The effect of our antialiasing algorithm is demonstrated by employing five layers of meshes on a bolt, shown in the bottom half of Fig. 15, and contrasted with the aliased result of applying the traditional algorithm with binary surface classification shown at the top half of Fig. 15. The antialiased effect can be clearly seen in the zoom view. Fig. 16 presents another example of applying the multi-layered Marching Cubes rendering on a lamp cover. It should be emphasized that the multi-layered Marching Cubes rendering generally requires more memory, and the rendering speed might be slower than other hardware-supported antialiasing algorithms. However, it provides a competitive object-space antialiasing method, and is quite useful when a high-quality antialiasing effect is required. It also helps the algorithm more appropriately represent a filtered model.

7. Conclusions and Future Work

Object simplification is an important research area for interactive applications. While most of the existing work focuses on geometry simplification, we have outlined in this paper a practical and robust method for topology simplification by controlled filtering and sampling an object into alias-free multi-resolution volume rasters. The strengths of our method are that it (a) works for a wide variety of objects; (b) simplifies the object topology in a controlled way; (c) is relatively easy to implement; and (d) is based on the robust theoretical foundation of signal-processing theory. To reduce the potentially large number of redundant triangles generated by the traditional surface-fitting algorithms, we have presented an adaptive Marching Cubes, which adheres to the topology preservation criterion, and guarantees that the generated mesh is within the user-specified error bound of the mesh generated by the standard Marching Cubes. To overcome the problems caused by the binary surface classification, we have further introduced a multi-layered Marching Cubes algorithm for hardware-assisted antialiasing.

Surface generation from multi-resolution volume rasters is an important step in our object simplification process. Our adaptive Marching Cubes, as well as the other existing geometry simplification algorithms, can be used to simplify the geometry of a model as a postprocess of the topology simplification stage. As part of our ongoing research in this area, we are currently developing a method of controlled low-pass filtering and sampling a polygon mesh or other formats of object into volume raster with adaptive size voxels, where the high curvature areas are represented by small voxels and the smooth areas by large voxels. The adaptive Marching Cubes is then modified to combine the topology and geometry simplification into one stage.

One of the restrictions of our algorithm, as mentioned in Section 3.1, is that it only works properly for closed surfaces. A possible solution for open polygonal patches is to first close them with dummy patches. Another area that promises to be of interest, and one that we are currently exploring, is the use of multi-resolution object hierarchies for collision detection. The idea here is to recursively perform collision detection among the multi-resolution descriptions of objects, starting from the lowest resolution representations and moving up to the higher resolutions only when an intersection is suspected. This approach works because every time a low-pass filter is applied with a larger support, the area affected by it becomes a superset since a larger filter support is applied. Thus, computation time is saved by avoiding intersection detection in regions that cannot possibly collide. Furthermore, this hierarchical approach can be interrupted, allowing users to trade accuracy for speed.

Acknowledgments

This work has been partially supported by the National Science Foundation under grants CCR-9205047 and CCR-9502239 and by the Department of Energy under the PICS grant. We thank Arie Kaufman for his contribution to the original ideas of this project. The source of the Visible Human data set is the National Library of Medicine and the Visible Human Project.

References

1. Amanatides, J., ‘‘Ray Tracing with Cones’’, *Computer Graphics (SIGGRAPH ’84 Proceedings)*, **18**, 3 (July 1984), 129-135.
2. Bloomenthal, J., ‘‘Polygonization of Implicit Surfaces’’, *Computer Aided Geometric Design*, **5**, (1988), 341-355.
3. Bronnimann, H. and Goodrich, M., ‘‘Almost optimal set covers in finite VC-dimension’’, *Proceedings Tenth ACM Symposium on Computational Geometry*, 1994, 293-302.
4. Carlbom, I., ‘‘Optimal Filter Design for Volume Reconstruction and Visualization’’, *IEEE Visualization’93 Proceedings*, San Jose, CA, October 1993, 54-61.
5. Carpenter, L., ‘‘The A-buffer, an Antialiased Hidden Surface Method’’, *Computer Graphics (SIGGRAPH ’84 Proceedings)*, **18**, 3 (July 1984), 103-108.
6. Clark, J., ‘‘Hierarchical Geometric Models for Visible Surface Algorithms’’, *Communications of the ACM*, **19**, 10 (1976), 547-554.
7. Clarkson, K. L., ‘‘Algorithms for Polytope Covering and Approximation’’, *Proc. 3rd Workshop Algorithms Data Structure, Lecture Notes in Computer Science*, 1993.
8. Crow, F. C., ‘‘A More Flexible Image Generation Environment’’, *Computer Graphics (SIGGRAPH ’82 Proceedings)*, **16**, 3 (1982), 9-18.
9. Das, G. and Joseph, D., ‘‘The complexity of minimum convex nested polyhedra’’, *Proc. 2nd Canad. Conf. Comput. Geom.*, 1990, 296-301.
10. DeHaemer, Jr., M. and Zyda, M. J., ‘‘Simplification of objects rendered by Polygonal Approximations’’, *Computers and Graphics*, **15**, 2 (1991), 175-184.
11. Durst, M., ‘‘Letters: Additional Reference to Marching Cubes’’, *Computer Graphics*, **22**, 2 (1988), .

12. Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M. and Stuetzle, W., ‘‘Multiresolution Analysis of Arbitrary Meshes’’, *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH*, August 1995, 173-182.

13. Fowler, R. and Little, J., ‘‘Automatic Extraction of Irregular Network Digital Terrain Models’’, *Computer Graphics*, **13**, 2 (August 1979), 199-207.

14. Fujishiro, I., Maeda, Y. and Sato, H., ‘‘Interval Volume: A Solid Fitting Technique for Volumetric Data Display and Analysis’’, *IEEE Visualization’95 Proceedings*, Atlanta, GA, October 1995, 151-158.

15. Funkhouser, T. A. and Sequin, C. H., ‘‘Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments’’, *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH*, August 1993, 247-254.

16. Guo, B., ‘‘Interval Set: A Volume Rendering Technique Generalizing Isosurface Extraction’’, *IEEE Visualization’95 Proceedings*, Atlanta, GA, October 1995, 3-10.

17. He, T., Hong, L., Kaufman, A., Varshney, A. and Wang, S., ‘‘Voxel-Based Object Simplification’’, *IEEE Visualization’95 Proceedings*, Atlanta, GA, October 1995, 296-303.

18. Heckbert, P. and Garland, M., ‘‘Fast Polygonal Approximation of Terrains and Height Fields’’, *Technical Report CMU-CS-95-181*, September 1995.

19. Heidrich, W., McCool, M. and Stevens, J., ‘‘Interactive Maximum Projection Volume Rendering’’, *IEEE Visualization’95 Proceedings*, Atlanta, GA, October 1995, 11-18.

20. Hinker, P. and Hansen, C., ‘‘Geometric Optimization’’, *IEEE Visualization’93 Proceedings*, San Jose, CA, October 1993, 189-195.

21. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. and Stuetzle, W., ‘‘Mesh Optimization’’, *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH*, August 1993, 19-26.

22. Kalvin, A. D. and Taylor, R. H., ‘‘SuperFaces: Polyhedral Approximation with Bounded Error’’, *Technical Report RC 19808*, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY 10958, 1994.

23. Levoy, M., ‘‘Display of Surfaces from Volume Data’’, *IEEE Computer Graphics and Applications*, **8**, 5 (May 1988), 29-37.

24. Levoy, M. and Whitaker, R., ‘‘Gaze-Directed Volume Rendering’’, *Computer Graphics (Proc. 1990 Symposium on Interactive 3D Graphics)*, **24**, 2 (March 1990), 217-223.

25. Lorensen, W. E. and Cline, H. E., ‘‘Marching Cubes: A High Resolution 3D Surface Construction Algorithm’’, *Computer Graphics (SIGGRAPH ’87 Proceedings)*, **21**, 4 (July 1987), 163-169.

26. Lounsbery, M., DeRose, T. D. and Warren, J., ‘‘Multiresolution Analysis for Surfaces of Arbitrary Topological Type’’, *Tech. Rep. 93-10-05B*, University of Washington at Seattle, January 1994.

27. Machiraju, R. and Yagel, R., ‘‘Accuracy Control of Reconstruction Errors in Volume Slicing’’, *Biomedical Visualization Proceedings’95*, Atlanta, GA, October 1995, 50-57.

28. Mitchell, J. and Suri, S., ‘‘Separation and approximation of polyhedral surfaces’’, *Proceedings of 3rd ACM-SIAM Symposium on Discrete Algorithms*, 1992, 296-306.

29. Montani, C., Scateni, R. and Scopigno, R., ‘‘Discretized Marching Cubes’’, *IEEE Visualization’94 Proceedings*, Washington, D.C., 1994, 281-287.

30. Muller, H. and Stark, M., ‘‘Adaptive generation of surface in volume data’’, *The Visual Computer*, 1993, 182-199.

31. Payne, B. and Toga, A., ‘‘Surface Mapping Brain Functions on 3D models’’, *IEEE Computer Graphics and Applications*, **10**, 2 (July 1992), 41-53.

32. Rossignac, J. and Borrel, P., ‘‘Multi-Resolution 3D Approximations for Rendering Complex Scenes’’, in *Modeling in Computer Graphics*, B. Falcidieno and T. L. Kunni, (eds.), Springer-Verlag, 1993, 455-465.

33. Sakas, G. and Hartig, J., ‘‘Interactive Visualization of Large Scalar Voxel Fields’’, *IEEE Visualization’92 Proceedings*, Boston, MA, October 1992, 29-36.

34. Schmitt, F. J., Barsky, B. A. and Du, W., ‘‘An Adaptive Subdivision Method for Surface-fitting from Sample Data’’, *Computer Graphics (SIGGRAPH ’86 Proceedings)*, **20**, 4 (1986), 179-188.

35. Schroeder, W., Zarge, J. and Lorensen, W., ‘‘Decimation of Triangle Meshes’’, *Computer Graphics (SIGGRAPH ’92 Proceedings)*, **26**, 2 (July 1992), 65-70.

36. Schroeder, W., Martin, K. and Lorensen, W., *The Visualization Toolkit*, Prentice Hall, 1996.

37. Turk, G., ‘‘Re-Tiling Polygonal Surfaces’’, *Computer Graphics (SIGGRAPH ’92 Proceedings)*, **26**, 2 (July 1992), 55-64.

38. Varshney, A., ‘‘Hierarchical Geometric Approximations’’, Doctoral Dissertation, Department of Computer Science, Tech. Rep.-050-1994, Department of Computer Science, University of North Carolina, Chapel Hill, NC 27599-3175, 1994.

39. Wang, S. W. and Kaufman, A. E., ‘‘Volume-Sampled 3D Modeling’’, *IEEE Computer Graphics & Applications*, **14**, 5 (September 1994), 26-32.

40. Westover, L., ‘‘Footprint Evaluation for Volume Rendering’’, *Computer Graphics (SIGGRAPH’90 Proceedings)*, **24**, 4 (August 1990), 367-376.

41. Wilhelms, J. and Van Gelder, A., ‘‘Topological Consideration in Isosurface Generation’’, *ACM Computer Graphics*, **24**, 5 (November 1990), 79-86.

42. Wilhelms, J. and Gelder, A. V., ‘‘Octree for Faster Isosurface Generation’’, *ACM Computer Graphics*, **24**, 5 (November 1990), 57-62.

43. Wolberg, G., *Digital Image Warping*, IEEE Computer Science Press, 1990.

44. ‘‘National Library of Medicine. Electronic Imageings: Report of the Board of Regions.’’, *NIH Publications 90-2197*, National Insistitute of Health, 1990.