Extracting Boundary Surface of Arbitrary Topology from Volumetric Datasets

Ye Duan **Hong Qin**

Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400
{yduan, qin}@cs.sunysb.edu

ABSTRACT

This paper presents a novel, powerful reconstruction algorithm that can recover correct shape geometry as well as its unknown topology from arbitrarily complicated volumetric datasets. The algorithm starts from a simple seed model (of genus zero) that can be initialized automatically without user intervention. The deformable behavior of the model is then governed by a locally defined objective function associated with each vertex of the model. Through the numerical computation of function optimization, the algorithm can adaptively subdivide the model geometry, automatically detect self-collision of the model, properly modify its topology (because of the occurrence of self-collision), continuously evolve the model towards the object boundary, and reduce fitting error and improve fitting quality via global subdivision. Commonly used mesh optimization techniques are employed throughout the geometric deformation and topological variation in order to ensure the model both locally smooth and globally well-defined. We have applied our algorithm to various real and synthetic volumetric datasets in order to empirically verify and validate its utility and efficacy. Our experiments have demonstrated that the new modeling algorithm is extremely valuable for surface reconstruction in volume graphics, volume segmentation in medical imaging, and iso-surface extraction in visualization.

1. INTRODUCTION

Recent technical breakthroughs in new imaging modalities such as CT, MRI and Ultrasound as well as other 3-D scanning technologies have given rise to massive volumetric datasets available in modern computer era. How to extract and reconstruct the shape of 3-D objects from these datasets accurately and efficiently remains to be both extremely challenging and significant in volume graphics, medical imaging, and visualization. One of its important applications that have proven to be essential in numerous engineering and medical fields is the non-invasive evaluation of an object's internal structure. For example, it allows the inspection of mechanical parts without destroying the product and the examination of internal organs without operating on the patient.

At present, many algorithms and techniques have been developed to effectively deal with the acquired volume data for various modeling and rendering tasks. In general, existing approaches can be classified into two different categories: they are either model-less techniques such as direct volume-rendering from voxel datasets or model-centered techniques such as deformable models. One major rationale for model-based approaches is that they provide the great potential for users to effectively interact with the dataset (especially regions of interest) and facilitate other subsequent processes such as segmentation, shape representation, matching, and motion tracking. Moreover, the inherent continuity and smoothness of the model can compensate for the unwanted sampling artifacts such as noise, gaps, and other irregularities on object boundaries. Hence, model-based approaches are more robust, especially for noise-corrupted datasets. Among the wide spectrum of model-driven techniques, deformable models [3, 15, 16, 17] have been extremely popular and successful primarily because they offer a unified and powerful approach that combines the knowledge from geometry, physics, approximation theory, and functional analysis. Nevertheless, there are several limitations associated with deformable models that are currently available. Among them, one of the most severe limitations is that the topology of the underlying shape either is very simple (such as genus zero) or must be known a priori (i.e., is determined elsewhere in a separate pre-processing stage) and remains unchanged throughout the time integration of model deformation. Another limitation of traditional deformable models is that users are often required to manually place the initial model inside the dataset.

In this paper, we propose a new modeling algorithm that can be employed to overcome these limitations. It can recover both complicated shape geometry and arbitrary unknown topology simultaneously from any volume datasets. Furthermore, the model can be automatically initialized by the system. The geometry and the deformable behavior of the model are governed by the principle of energy minimization. After a simple seed model is initialized, the model will deform and grow towards the boundary of the modeled dataset in accordance with the local cost function associated with each vertex of the model. During the process of model deformation, both global and local/adaptive subdivision operations on the model can be automatically applied whenever necessary in order to refine the model to an appropriate resolution and achieve different levels of detail. More importantly, by using

a novel distance-based collision detection scheme, the model can automatically detect self-collision and modify its topology accordingly. In order to ensure the recovery of the correct topology from arbitrary datasets, we develop a novel, yet simple scheme that can prevent inter-penetration in the vicinity of any vertex of the model. This scheme, combined with mature mesh optimization techniques, has proven to be effective and can generate a good, high-quality polygonal mesh which can both reconstruct the data geometry and extract the arbitrary topology from any complicated dataset through model deformation.

The rest of the paper is organized as follows. The next section summarizes all the important literatures that are relevant to our work. Section 3 introduces the energy-based minimization method used in our paper, which is the key mechanism behind the model-growing step of the algorithm. The other six main steps of our algorithm are discussed in detail in Section 4. Section 5 demonstrates the experimental results we obtained using our algorithm. Finally, the conclusion and some future work are given in Section 6 and Section 7, respectively.

2. RELATED WORK

During recent years, a lot of research has been conducted in the areas of surface reconstruction, volume segmentation, and iso-surface extraction. The majority of the published results falls into two groups: (1) static, geometric techniques; and (2) dynamic, energy-based techniques. Among the static methods, one of the first algorithms was devised by Fuchs et al. [3]. They developed a means of stitching a series of two-dimensional contours together by fitting a triangular strip between adjacent contours. The main drawback of this approach is that the user must manually identify a contour in every slice that comprises the object. Later on, Lorenson and Cline developed an algorithm called marching cubes [6] that has proven very useful for generating three-dimensional polygonal surface from volume data with no connectivity information. In their algorithm, a cube is bounded by eight pixels located on two adjacent slices. Each vertex is coded as either inside or outside the object relative to the surface-defining threshold. Based on the configuration of vertices that lie inside and outside the object, the cube is triangulated. The triangles indicate where the surface passes through the cube. The technique of marching cubes provides an accurate method for creating three-dimensional polygonal surfaces from slice data that can then be manipulated and visualized. However, the marching cubes model records all the details associated with the original data regardless of whether these details are insignificant or sampling artifacts. Also, since marching cubes generate at least one triangle per voxel through which the surface passes. This results in an enormous number of extremely small triangles, thus making it difficult to interactively render these models. On the other hand, our underlying model is a subdivision-based deformable model. It can produce models of varying resolution and can remove noise much easier.

In the category of dynamic approaches, the most famous one is the snake model proposed by Kass, Witkin and Terzopoulos [4]. A snake is essentially a spline that minimizes the energy associated with the spline. The total energy of the snake model is contributed from three different sources: (1) the internal energy of the spline, (2) image forces, and (3) external constraints. Through the minimization of the spline's internal energy, the snake will always remain smooth. The image forces guide the snake toward lines and edges of interest, while the external constraints allow the user to identify specific features to model. The original snake model only behaves and deforms on a 2-D plane, and can only model the topology of simple 2-D objects. Later on, Terzopoulos, Kass and Witkin generalized the concept of snakes into symmetry-seeking models [18]. They derive a three-dimensional shape from a two-dimensional image by modeling an axis-symmetric elastic skin spread over a flexible spine. Finite element methods are also explored in deformable models by several researchers, including Cohen and Cohen [2], Terzopoulos and Metaxas [17], and McInerney and Terzopoulos [10]. On the other hand, Miller et al. [12, 13] proposed a polygon-based deformable model. The behavior of the model is determined by a local cost function associated with each model vertex. The cost function is a weighted linear combination of three terms: (1) a deformation potential that pushes the model vertices towards the object boundary, (2) an image term that identifies features such as edges and acts against the model expansion, and (3) a term that constrains the motion of each vertex to remain not far from the centroid of its neighbors. Similar to the snake model, the topological variation in Miller et al.'s work is not allowed. The modeled dataset must be homomorphic to a sphere. Recently, Qin and Mandal [14, 8] proposed dynamic subdivision surfaces for surface reconstruction. Their approaches combine the advantages of free-form deformable models with the nice properties of subdivision surfaces, and their algorithm allows the direct manipulation of the limit surfaces defined by the subdivision process on the initial control mesh. One severe limitation of all aforementioned deformable models is that the topology must be determined before the geometric deformation, i.e., only geometric aspects of the underlying dataset are reconstructed through energy-based simulation.

To overcome this limitation, several researchers have proposed implicit-based methods [1, 7, 21]. The key part of these schemes is the modeling of an evolving level set of some implicitly defined function. Despite the advantages of topological and geometric flexibility, implicit models are in general not very convenient for shape analysis and visualization, and also very difficult for user interaction. Recently, McInerney and Terzopoulos [11] proposed topological adaptable snake, which is a parametric snakes model that has the power of an implicit formulation. The basic idea is to superimpose a simplicial grid on the image domain and iteratively reparameterize the geometry of deforming snakes. In a different approach, Szeliski et al. [15] use a dynamic, self-organizing oriented particle system to model the surface boundary of objects. The particles can reconstruct objects with complex shapes and topologies by "flowing" over the data, extracting and conforming to meaningful surfaces. A triangulation is then performed which connects the particles to form a continuous global model that is consistent with the inferred surface of the underlying object.

Our algorithm is based on a polygonal model with the capability of recursive refinements through surface subdivision. It further generalizes the work of Miller et al. [12, 13] and can overcome some limitations of their algorithm. In particular, our technique is capable of recovering geometric shape of arbitrary, unknown

topology from volume data and the initial model is automatically placed within the dataset. Besides the aforementioned work, two other research advances are also of relevance. One is the work of Welch and Witkin [19, 20]. They use a triangle mesh to approximate the underlying smooth variational surface for free-form surface design. Another one is the more recent work called "skin" algorithm proposed by Marksoian et al. [9]. Their goal is to generate a triangle mesh to approximate the surface implicitly defined by the "skeletons".

3. ENERGY-BASED OPTIMIZATION

The deformable behavior of the model is governed by the principle of energy-based minimization. A locally defined cost function is associated with each vertex of the polygonal model. The cost function is a weighted linear combination of four constraints whose objectives are to achieve the desired behaviors in the simulated model. We shall briefly review these four components in Section 3.1 followed by the minimization method in Section 3.2.

3.1 Constraint Modeling

The energy function $C_i(x,y,z)$ associated with the current location of each model is explicitly formulated as

$$C_i(x,y,z) = a_0 D(x,y,z) + a_1 B(x,y,z) + a_2 V(x,y,z) + a_3 A(x,y,z) \quad (1)$$

where $D(x,y,z)$ is the deformation potential, $B(x,y,z)$ is the boundary constraints, $V(x,y,z)$ is the curvature constraint, and $A(x,y,z)$ is the angular constraint. a_0, a_1, a_2, a_3 are the four corresponding non-negative weighting parameters.

3.1.1 Deformation potential -- $D(x,y,z)$

Deformation potential $D(x,y,z)$ offers the mechanism to inflate the model. It defines a scalar field where each position in space is assigned a value based on the frame of reference. The vertex will move along the direction of the lowest local potential (in absence of other constraints). In order to model concave objects, the *normal tracking* method is used, i.e., each vertex is attracted to a point located in the vicinity of normal direction of the polyhedron surface. During each evolving step, every vertex moves in the general direction of the local surface normal in order to decrease its deformation potential.

During the refinement process (local and global subdivision) which we will discuss in detail in Section 4, it is possible that new vertices are added to the model on the opposite of the boundary. In order to move these model points to the other side of the boundary and hence increase the accuracy and quality of the model, the surface normal used in the deformation potential of these model points is defined to point in the opposite direction. The effect is that a model point will migrate towards the true boundary of the object regardless of whether the model point is located inside or outside the object. Hence, as long as the initial model intersects the object boundary, i.e. some of the model points are inside object, the remainders are outside the object, the model tends to seek out the true boundary of the object.

3.1.2 Boundary constraint -- $B(x,y,z)$

Boundary constraint $B(x,y,z)$ affords the mechanism for the model to interact with the dataset and identify the boundary. It is used to counter-balance the deformation potential and will restrict, direct, and counter-act the general progression of the deformation. We make use of a shifted threshold operator:

$$B(x,y,z) = \begin{cases} Image(x,y,z) - T & Image(x,y,z) \geq T \\ 0 & Image(x,y,z) < T \end{cases} \quad (2)$$

where $Image(x,y,z)$ is the gray-level intensity distribution of the voxel at location *(x,y,z)*, and *T* is the threshold value that identifies the object.

When a model point steps over the edge of an object, the algorithm returns a value that should increase the overall cost of the system. Therefore, the minimization process is required to either move the vertex by a smaller amount or not move the vertex at all. Hence the vertex will approach the boundary without crossing over it (unless its neighbors pull it over the edge).

3.1.3 Curvature constraint -- $V(x,y,z)$

The first two constraints have the ability to grow the model until all the vertices reach the boundary of the underlying object. During the deformation process, it is desirable for a vertex not to stray far away from its neighbors. This suggests the use of *Curvature constraint* $V(x,y,z)$ which is a reasonable approximant of the local curvature, and it is defined as the ratio of the distance from the current model point to the centroid of its neighbors over the maximum distance among all the neighbors of the current model point:

$$V(x,y,z) = \frac{\| (x,y,z) - \frac{1}{n}\sum_{1}^{n}(x_j, y_j, z_j) \|}{\max_{j,k}(\| (x_j, y_j, z_j) - (x_k, y_k, z_k) \|)} \quad (3)$$

where (x,y,z) is the current model point, n is the number of neighbors to the current model point, $(x_j, y_j, z_j), (x_k, y_k, z_k)$ are the neighbors of the current model point, $1 \leq j,k \leq n$. Curvature constraint also has the effect of keeping the vertices well distributed during the deformation process. We will discuss this issue in more details in the next section.

3.1.4 Angular constraint $A(x,y,z)$

The fourth constraint--*Angular constraint* $A(x,y,z)$ is used to simulate the effect of attaching a very stiff string between any two adjacent faces. Similar to the boundary constraint, the value of angular constraint is either zero or very large. At each deformation step, the edges on the one-neighborhood of each vertex are identified, and all the dihedral angles between the two adjacent faces of these edges are calculated. If the next move of the vertex will cause any of these dihedral angles to become smaller than the threshold, the angular constraint will become very large and the vertex is not allowed to move at this deformation cycle. Otherwise, the angular constraint is zero. Angular constraint can effectively keep any two adjacent faces from being too close to each other. This constraint, used in concert with the more aggressive stressed-edge resolution approach and the mesh optimization techniques that will both be discussed later in this paper, will effectively prevent the local inter-penetration of adjacent faces.

3.2 Optimization Method

An iterative method is employed to numerically compute the minimization of our cost function explained above. The advantage of this approach is that it is extremely general and can offer an accurate, stable solution even for very large systems, therefore, it is well suited for our purpose in shape recovery of large datasets. A vertex of the model will move along the direction of the steepest descent along the cost surface, which is opposite to the gradient of the cost function C_i. The gradient $(\frac{\partial C_i}{\partial x}, \frac{\partial C_i}{\partial y}, \frac{\partial C_i}{\partial z})$ is numerically approximated using the central difference of the overall cost function for the current position of the model vertex with a very small perturbation. The amount that a vertex can move is adjusted based upon the current configuration of the cost space. The step size can be reduced several times if the magnitude of the current step size results in an increase in the cost function. If a step size is no longer able to reduce the cost of the vertex, then the vertex is not allowed to move at this step. If a vertex has not moved for a certain number of deformation cycles, the vertex will be marked as non-active and will be excluded from future numerical integrations.

4. ALGORITHM

The entire pipeline of the modeling algorithm consists of the following seven main steps:

1. Model initialization.
2. Stressed edge resolution.
3. Model growing.
4. Local adaptive subdivision.
5. Mesh optimization.
6. Collision detection and topology changes.
7. Global subdivision.

After the model is automatically initialized at step one, the model will start its deformation process. It will loop through step two to step six at each deformation cycle. The deformation process stops when the model reaches its equilibrium, i.e. all the vertices of the model have been marked as non-active. Finally, the model can be globally subdivided several times until a user-given error criterion is met. Figure 1 shows the flow chart of the algorithm. We have highlighted the mechanism of model growing (step 3) in the previous section. In this section, we will detail the other six steps of the algorithm.

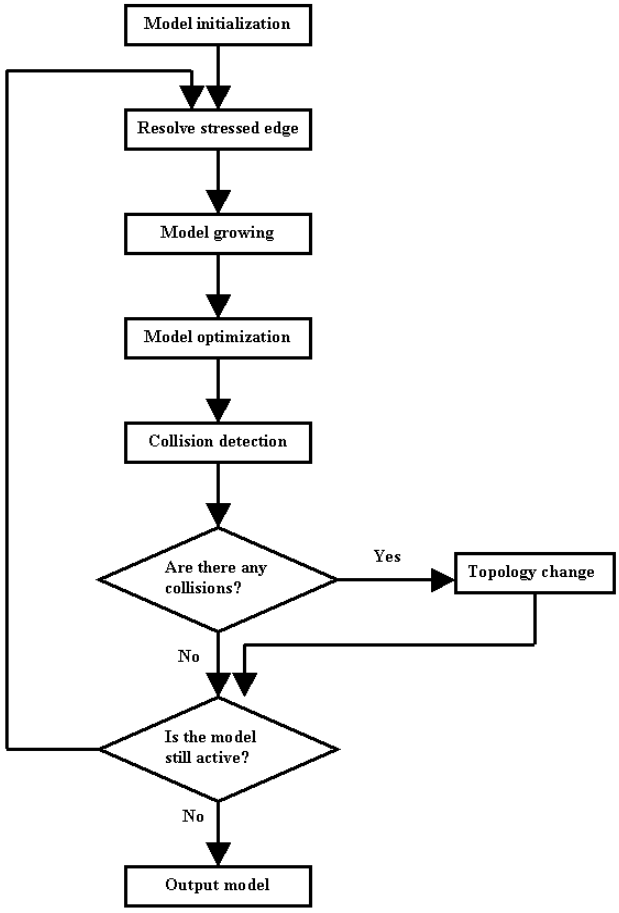


Figure 1: Pipeline of the algorithm.

4.1 Model Initialization

The *seed* model may be any kind of closed polyhedra. For simplicity and without loss of generality, we use a sphere-like polyhedron consisting of 24 triangles of equal size. Before the deformation process starts, the algorithm will search through the input volume datasets and find a non-boundary voxel. This voxel is then identified as the initial center position of the seed model. Note that the *seed* model does not need to be completely inside the dataset because the model will flip the normal tracking direction of the vertex if the vertex is detected to be outside the dataset.

4.2 Stressed Edge Resolution

One phenomenon which oftentimes appears in a polygon based deformable model is the local inter-penetration of neighboring faces. Local inter-penetration typically occurs between two portions of the surface separated by a chain of stressed edges. In practice, a stressed edge is identified if its two adjacent faces form an angle of less than 60 degrees (this value may vary across different systems). In this paper, we propose a simple, yet very powerful method that can efficiently solve this problem. At the beginning of each deformation cycle, all the stressed edges are detected by calculating the dihedral angle. Then each stressed edge is split into two small edges at the middle point and the middle point is further moved to the middle position of the two opposite vertices. Figure 2 demonstrates our method of resolving stressed edges.

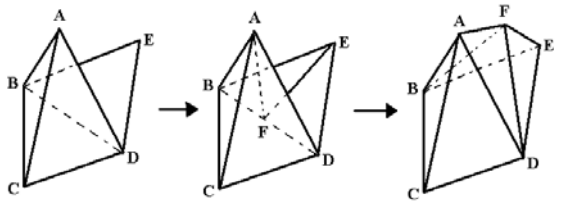


Figure 2: Stressed edge resolution. (a) Edge BD is marked as stressed edge because the dihedral angle between its two adjacent faces ABD and EBD is less than the threshold. (b) Edge BD is split at the middle, and the middle point F of edge BD is connected with vertices A, B, D and E. (c) Finally, F is moved to the middle of vertices A and E.

4.3 Local Adaptive Subdivision

In order to control the smoothness of the model and the size of each polygon during the model-growing phase, we must allow the model to be able to increase its degrees of freedom during the deformation process. One simple, straightforward technique is global subdivision, i.e., globally subdivide the model whenever necessary. The drawback of the global subdivision approach is that it may generate a lot of unnecessary vertices on surface regions where a good approximation to the data boundary has already been achieved. Alternatively, we take advantage of the local adaptive subdivision approach, i.e., we only need to subdivide active regions that are still growing. A face is subdivided if its area is larger than a certain user-defined threshold, and moreover, at least one of its three vertices is still active. The typical subdivision rule is as follows. The algorithm will introduce a new vertex at the middle position of each old edge, and connect all the three new vertices. Thus four smaller new faces are generated from each old face. To maintain subdivision connectivity, all the triangles adjacent to the current face also need to be subdivided correspondingly. For example, in Figure 3, in order to subdivide the central triangle BDE, all the three adjacent triangles ADB, CBE and DFE need to be subdivided as well. And each of these three triangles is subdivided into two smaller ones by splitting the adjacent edge they share with the central triangle BDE.

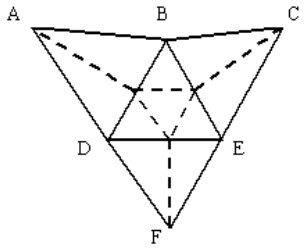


Figure 3: Local adaptive subdivision scheme. The solid lines are the old edges, the dashed lines are the new edges. The center triangle BDE is divided into four smaller triangles by connecting the three middle points of the old edges. Each of the three adjacent triangles ADB, CBE and DFE is split into two smaller triangles.

4.4 Mesh Optimization

The algorithm can automatically construct the new subdivision mesh during the deformation phase. Therefore, it's critical to improve and maintain the mesh quality throughout the process to keep the model both locally smooth and globally well conditioned. In general, three issues must be considered as also observed by Welch et al. [17]: (1) how to keep the nodes well distributed; (2) how to keep the triangles well shaped; and (3) how to keep an appropriate node density.

4.4.1 Nodes distribution

A popular scheme for keeping the nodes well distributed is called *Laplacian Smoothing*. It can be implemented by iteratively moving each node to the centroid of its neighbors. In our algorithm, we decide not to implement this scheme because of the high numerical cost associated with it. Instead, we rely on the curvature constraint $V(x,y,z)$ in our local cost function $C_i(x,y,z)$ in equation (1) associated with each vertex to keep vertices from straying too far away from the centroid of their neighboring vertices. We observe that our curvature constraint behaves well in maintaining a good distribution of the nodes.

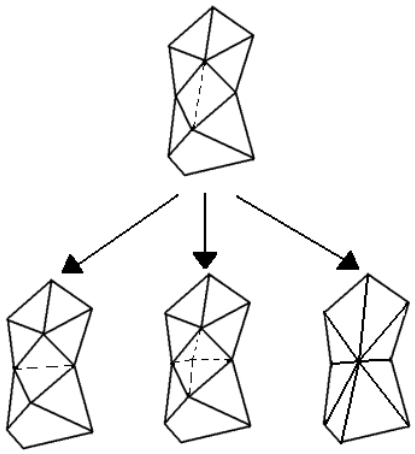
4.4.2 Triangle shape

A triangulation with nodes well distributed can still have many skinny triangles. It is well known that the best possible surface triangulation over a set of points with known topology is the *Delaunay triangulation*. In addition, a *Delaunay triangulation* of an arbitrary surface can be incrementally recovered from a valid initial surface triangulation through edge swapping. We swap an edge if doing so will increase the minimum angle within its adjacent faces. Repeated applications of this swap operation always keep increasing the minimum angle and hence result in a *Delaunay triangulation* at the end of the procedure. That is, it maximizes the minimum angle on all the triangles of the mesh. In practice, an edge is eligible for swapping only if the dihedral angle between its two adjacent faces is larger than a certain user-

defined threshold, i.e., the local surface across the edge is flat enough. Moreover, an edge is swapped only if its local minimum-angle will be increased by a certain small minimum (specified by users and heuristically determined by the algorithm). These two conditions can guarantee that the edge-swapping algorithm always functions correctly and terminates eventually.

4.4.3 Nodes density

During the deformation process, some nodes may cluster with each other, and some other nodes may be too far away from each other. To maintain an appropriate node density, two other operations are needed here: *edge split* and *edge collapse*. An edge-split is triggered if any two neighbors are too far apart. Similarly, if any node is too close to each of its neighbors, the node is destroyed using the edge collapse. In addition, skinny triangles are also eliminated at this step by edge collapsing. All the three inner-angles of each triangle are calculated. If any one of the three inner-angles of a triangle is too small, then the triangle containing the inner-angle will be eliminated by collapsing the edge opposite to this inner-angle. To restore a quality mesh, the edge swapping is always applied after any edge split and edge collapse operations. Figure 4 illustrates the three mesh operations.



(a) (b) (c)

Figure 4: Mesh optimization operations. (a) Edge swap. (b) Edge split. (c) Edge collapse.

4.5 Collision Detection and Topology Changes

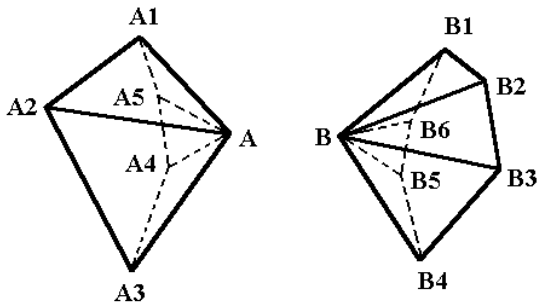
In order to recover a shape of arbitrary, unknown topology, the model must be able to change its topology properly whenever a collision with other parts of the model is detected. Various kinds of collisions can be considered, such as face-to-face, edge-to-edge, vertex-to-vertex, edge-to-face, etc. Techniques such as surface-surface intersection and trimming have been proposed to solve collision detections. However, these techniques are usually very time consuming. We propose a novel distance based collision detection scheme that is simple, fast and efficient. Figure 5 illustrates the three steps of the scheme: (1) *collision detection,* (2) *identify one-neighborhood and put them into correspondence,* and (3) *change the topology.*

Collision detection: If the distance of two non-neighbor active vertices is smaller than the threshold, a collision will be identified and a merge-operation is triggered. If the distance between several pairs of active vertices is smaller than the threshold, the closest pair of vertices is chosen. For example, in Fig. 5(a), because the distance between two active vertices A and B is smaller than the threshold, a collision between regions around vertex A and B is detected and a merge operation is triggered.

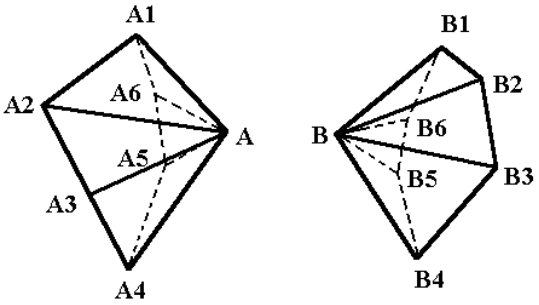
Identify one-neighborhoods and put them into correspondence: To merge the two parts of the model. First, we need to identify and collect all the one-neighborhood points for each of these two vertices. Then these two sets of points (i.e., one-neighborhood points) are sequenced separately and are put into correspondence. To do so, we use the same procedure as [19]: Iteratively refine the neighborhood with fewer edges by splitting its longest edge until both have the same number of nodes, then choose the alignment that minimizes the sum of squared distances between nodes. In Fig. 5(a), originally the one-neighborhood of vertex A has five nodes: {A1, A2, A3, A4, A5}, the one-neighborhood of vertex B has six nodes: {B1, B2, B3, B4, B5, B6}. To make these two one-neighborhoods have the same number of nodes, we first find the longest edge of the one-neighborhood of vertex A, which is the edge between nodes A2 and A3. And then split this edge into two edges and insert a new node in between. Finally, we put these two sets of points into correspondence by finding the alignment that minimizes the sum of squared distances between nodes. In Fig. 5(b), point set {A1, A2, …, A5} are corresponding to {B1, B2, …, B6} respectively.

Change the topology: After the two sets of points are put into correspondence, each point is connected with its corresponding point in the opposite point set. The two center vertices and all its incident edges are removed (Fig. 5(c)). The newly created quadrilaterals are further triangulated by splitting each quadrilateral into two triangles along one of its diagonals (Fig. 5(d)).

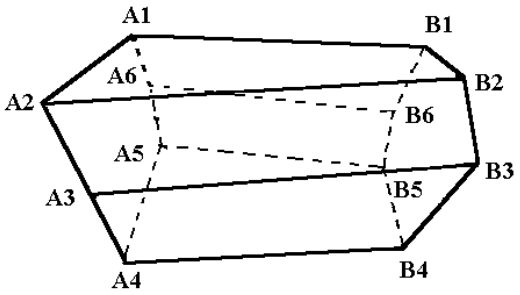
The mesh optimization processes will quickly smooth out any artifacts that may result from the matching procedure once the merge has been completed.



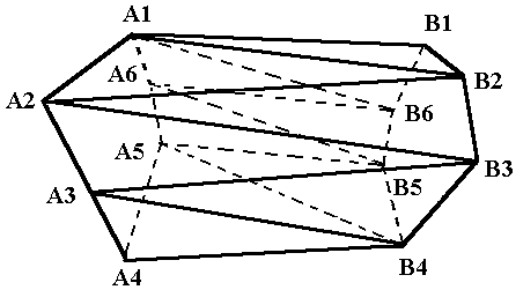
(a)



(b)



(c)

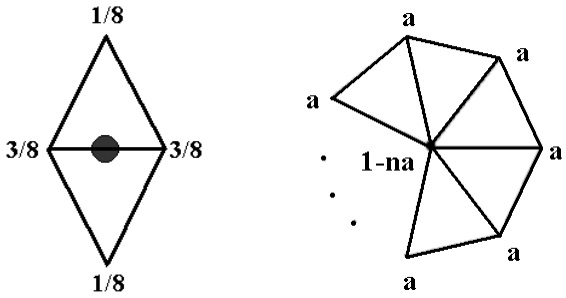


(d)

Figure 5: Collision detection and topology change. (a) A collision is detected between the region around vertex A and the region around vertex B. (b) The one-neighborhoods of vertex A and vertex B are put into correspondence. (c) The corresponding vertices between the one-neighborhoods of vertex A and vertex B are connected. Vertex A and vertex B and their incident edges are removed. The topology of the model is modified. (d) Each of the newly created quadrilaterals is split into two triangles.

4.6 Global Subdivision

Once a rough estimation of the topology and geometry of a shape is achieved, the model can be subdivided several times to improve the fitting accuracy. We choose Loop's scheme [7] in our model though other schemes would also achieve this goal. Figure 6 shows the Loop's subdivision scheme. There are two kinds of new vertices generated at each level of subdivision: edge points and vertex points. Each old edge will generate a new edge point using the rule shown in Fig. 6(a). Each old vertex will generate a new vertex point using the rule shown in Fig. 6(b). By connecting each vertex point with its two adjacent edge points and connect the three edge points with each other, four smaller triangles are generated from each old triangle. After one level of global subdivision, the model will deform again based on the cost function explained above, and will arrive at a more accurate configuration of the shape because we now have more degrees of freedom for the model. Since the unknown topology of the underlying data set has already been recovered, there is no need for collision detection at this stage.



(a) (b)

Figure 6: Subdivision rules for Loop's scheme: (a) Edge point rule. (b) Vertex point rule. $a = \frac{3}{8n}$ for $n > 3$ and $a = \frac{3}{16}$ for $n = 3$, n is the valence of the vertex.

5. EXPERIMENTAL RESULTS

We have developed an experimental system using C++ and FLTK.

Figures 7, 8, 9, and 10 show some of the experimental results we have conducted using this system. In all figures, (a) shows a volume-rendered image of the original volume datasets. (b) and (c) are the two snapshots of the model during the deformation process. (d) is the first round estimation of geometry and topology of the model. Red color shows the regions of the model that are still active, while the non-active regions of the model are colored as blue. (e) is the refined shape of the model after one level of global subdivision. By comparing (d) and (e), we can clearly see the improvement of the fitting accuracy of the model after one level of global subdivision.

In addition, Fig. 10(f) shows the refined shape of the model after two levels of global subdivision, the groves on the inner-surface of the nut are recovered very well. Fig. 10(g), (h) and (i) are the error maps of the models of Fig. 10(c), (d) and (f), respectively. The fitting error is calculated by dividing the distance between the model vertex and the closest volume boundary voxel by the diameter of the smallest bounding sphere of the object. The green

color shows regions whose fitting error is less than 0.5%. The red color represents regions whose fitting error is greater than 2%. These three error maps illustrate that the fitting error is greatly reduced after two levels of global subdivision. Since our model currently cannot recover sharp edges and corners, the regions that are still red after two levels of subdivision are primarily the regions in the vicinity of sharp edges and corners.

Our algorithm also supports multiple seed model initialization. For example, in Fig. 9(b), four seeds are initialized at four different positions at the same time. Each model will grow independently (Fig. 9(c)) and will merge with other models whenever a collision is detected (Fig. 9(d)).

Table 1 lists the four weighting coefficients for calculating the local cost function associated with each vertex using Equation (1). Table 2 and Table 3 summarize the statistics of our examples. In particular, Table 2 is the input of three dimensions of the volumetric image data. Table 3 lists the size of recovered shape of the model along with the maximum fitting error of each model.

Currently, several parameters need to be set by the user at the start of the deformation process. They are: (1) the face area threshold for local adaptive subdivision, (2) the distance threshold for collision detection, and (3) the edge length threshold for mesh operations such as edge split and edge collapse. In the future, we plan to simplify these parameters by conducting a preprocessing step and normalize the input dataset to the same scale. Then it should be possible for the algorithm to automatically set the proper values for these parameters.

a_0	a_1	a_2	a_3
1	1	1.6	1

Table 1: Weighting coefficients.

Figure#	X(#voxels)	Y(#voxels)	Z(#voxels)
7	67	127	67
8	128	120	47
9	32	32	64
10	68	41	59

Table 2: The dimensions of the input volume datasets.

Figure#	#Vertices	#Edges	#Faces	Max. fitting error (%)
7(d)	2491	7491	4994	1.05
7(e)	9889	29685	19790	0.92
8(d)	1005	3015	2010	0.533
8(e)	4299	12897	8598	0.38
9(d)	2379	7161	4774	1.26
9(e)	9848	29568	19712	0.94
10(b)	187	561	374	3.88
10(c)	774	2322	1548	2.22
10(d)	3141	9423	6282	1.85

Table 3: Recovered model information.

6. CONCLUSIONS

In this paper, we have presented a new modeling algorithm for the extraction of boundary surfaces from volumetric datasets. Through the use of a new collision-detection method and a novel stressed-edge resolution scheme, coupled with mesh optimization techniques, the algorithm is able to overcome several limitations associated with conventional deformable models. The algorithm can recover the shape of arbitrary geometry and its unknown topology simultaneously. Because the underlying model is a subdivision-based model, it naturally supports levels of detail. After the initial estimation of both topology and geometry of the dataset is achieved, the user can control the fitting quality easily by specifying the number of levels of global subdivision. Furthermore, the algorithm can be multi-threaded for improved performance, i.e., multiple seed models can be initialized at different locations at the same time. Hence, parallel implementation is readily available. Throughout the deformation process, each seed model will grow independently and will merge with neighboring models whenever a collision occurs.

We expect our modeling algorithm to be extremely valuable in such areas as computer graphics, medical imaging, computer aided design, and visualization. It can be used to extract the internal organs for medicine, or to model scanned mechanical parts for engineering. Furthermore, our model can be easily extended to higher dimensional spaces (e.g., to model a series of time varying volume data which is essential in motion tracking).

7. FUTURE WORK

Several improvements are possible. First, we currently use a brute-force searching algorithm for collision detection. We shall continue to improve their time performance by using techniques such as hierarchical bounding box. Second, our current modeling algorithm functions in a semi-automatic fashion. Although the seed model is automatically initialized, users have to interactively select several parameters before the deformation process starts. This would require certain knowledge for users. Hence, the current version of our system is perhaps more appropriate for domain specialists who are more familiar with the underlying datasets and their attributes. It would be ideal to fully automate our system so that all relevant parameters can be determined heuristically without user intervention. Making all the parameters transparent in our system would appeal to naive users.

We also plan to extend the functionality of our model and its associated system along the following directions in the future. We shall enhance our model so that it can recover sharp features such as corners and creases. Also, besides the coarse-to-fine levels of detail currently available in our system, we shall explore the data-reduction capability (i.e., from fine to coarse), this will enable us to use very few degrees of freedom to model complicated shapes of geometry and topology.

ACKNOWLEDGEMENTS

This research was supported in part by the NSF CAREER award CCR-9896123, the NSF grant DMI-9896170, the NSF ITR grant IIS-0082035, and a research grant from Ford Motor Company. We would like to thank Professor Arie Kaufman for providing the volume rendering facilities in our visualization lab and providing most of the volume datasets used in this paper. We want to thank Professor Tim McInerney for providing the phantom vertebral image. We are also very grateful for the help from Kevin Kreeger, Ming Wan, Kevin McDonnell, Hui Xie, Haixia Du and Meijing Zhang.

REFERENCES

[1] V. Caselles, R. Kimmel, and G. Sapiro. Geodisc active contours. In Proceedings of the Fifth International Conference on Computer Vision (ICCV'95), pages 694-699, June 1995.

[2] L.D. Cohen and I. Cohen. Finite element methods for active contour models and balloons for 2D and 3D images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 15(11), pages 1131-1147, November 1993.

[3] H. Fuchs, Z.M. Kedem, and S.P. Uselton. Optimal surface reconstruction from planar contours. Communication of ACM, 20(10), pages 693-702, 1977.

[4] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. International Journal of Computer Vision, 1(4), pages 321-331, 1988.

[5] Charles Loop. Smooth subdivision surfaces based on triangles. Master's thesis, Department of Mathematics, University of Utah, August 1987.

[6] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. Computer Graphics (SIGGRAPH'87 Proceedings), pages 163-169, July 1987.

[7] R. Malladi, J. Sethian, and B. Vemuri. Shape modeling with front propagation: A level set approach. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(2), pages 158-175, February, 1995.

[8] C. Mandal, H. Qin, and B.C. Vemuri. A novel FEM-based dynamic framework for subdivision surfaces. In Proceedings of Fifth ACM Symposium on Solid Modeling and Applications (Solid Modeling'99*)*, pages 191-202, Ann Arbor, Michigan, June 1999.

[9] L. Markosian, J. M. Cohen, T. Crulli, and J. F. Hughes. Skin: a constructive approach to modeling free-form shapes. Computer Graphics (SIGGRAPH'99 Proceedings), pages 393-400, August 1999.

[10] T. McInerney and D. Terzopoulos. A dynamic finite element surface model for segmentation and tracking in multidimensional medical images with applications to cardiac 4D image analysis. Computerized Medical Imaging and Graphics, 19(1), pages 69-83, 1995.

[11] T. McInerney and D. Terzopoulos. Topologically adaptable snakes. In Proceedings of the Fifth International Conference on Computer Vision (ICCV'95), pages 840-845, June 1995.

[12] J.V. Miller. On GDM's: Geometrically deformed models for the extraction of closed shapes from volume data. Masters thesis, Rensselaer Polytechnic Institute, Troy, New York, December 1990.

[13] J.V. Miller, D.E. Breen, W.E. Lorensen, R.M. O'Bara, and M.J. Wozny. Geometric deformed models: a method for extracting closed geometric models from volume data. Computer Graphics (SIGGRAPH'91 Proceedings), pages 217-226, July 1991.

[14] H. Qin, C. Mandal, and B.C. Vemuri. Dynamic Catmull-Clark subdivision surfaces. IEEE Transactions on Visualization and Computer Graphics, 4(3): 215-229, July 1998.

[15] R. Szeliski, D. Tonnesen, and D. Terzopoulos. Modeling surfaces of arbitrary topology with dynamic particles. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'93), pages 82-87, June 1993

[16] D. Terzopoulos, and K. Fleischer. Deformable models. The Visual Computer 4(6), pages 306-331, 1988.

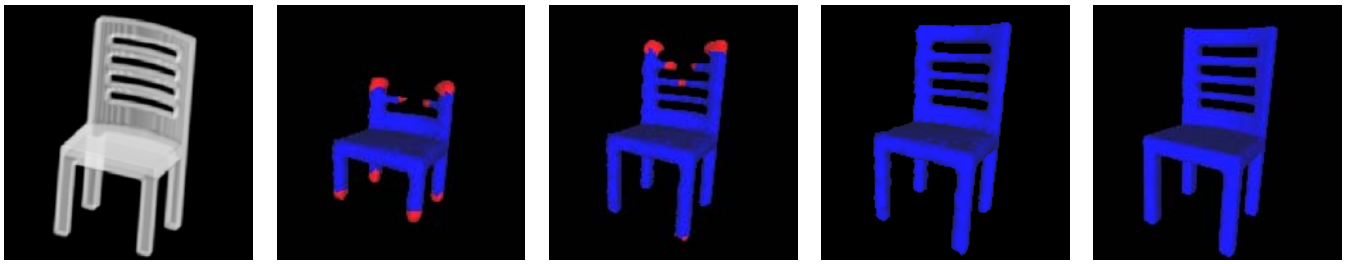
[17] D. Terzopoulos, and D. Metaxas. Dynamic 3D models with local and global deformations: Deformable superquadrics. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(7), pages 703-714, July 1991.

[18] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-seeking models and 3d object reconstruction. International Journal of Computer Vision, 1(3), pages 211-221, 1987.

[19] W. Welch and A. Witkin. Free-form shape design using triangulated surfaces. Computer Graphics (SIGGRAPH'94 Proceedings), pages 247-256, July 1994.

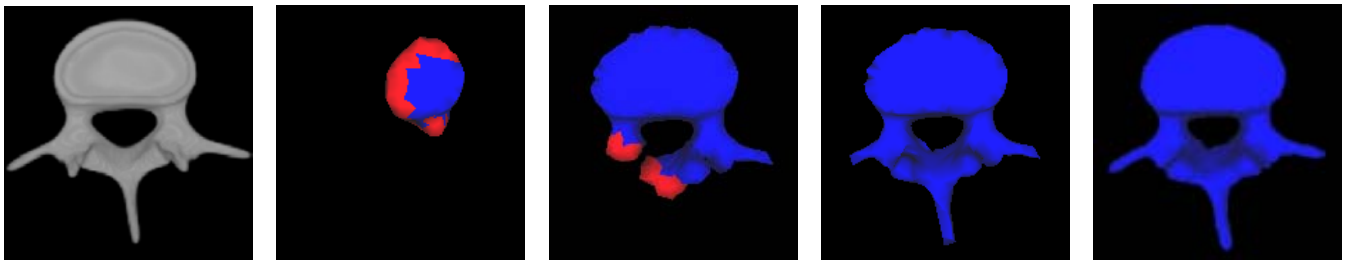
[20] W. Welch and A. Witkin. Serious Putty: topological design for variational curves and surfaces. PhD thesis, Carnegie Mellon University, June 1995.

[21] R. T. Whitaker. A level-set approach to 3d reconstruction from range data. International Journal of Computer Vision, 29(3), pages 203-231, September 1998.



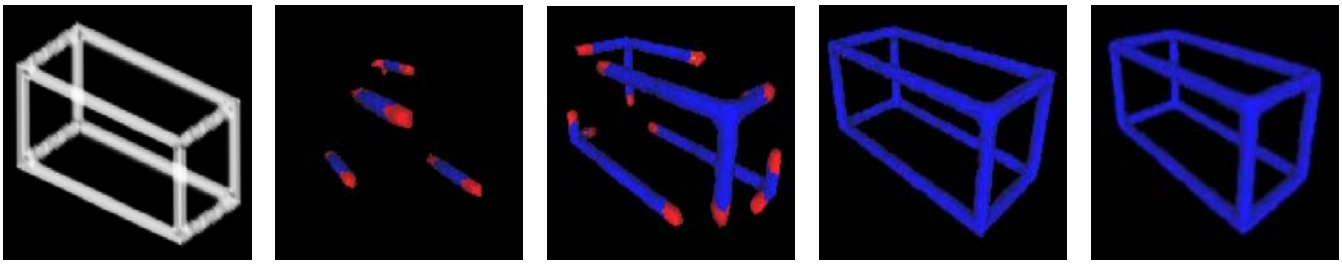
(a) (b) (c) (d) (e)

Figure 7: Surface reconstruction from volumetric image data of a chair.



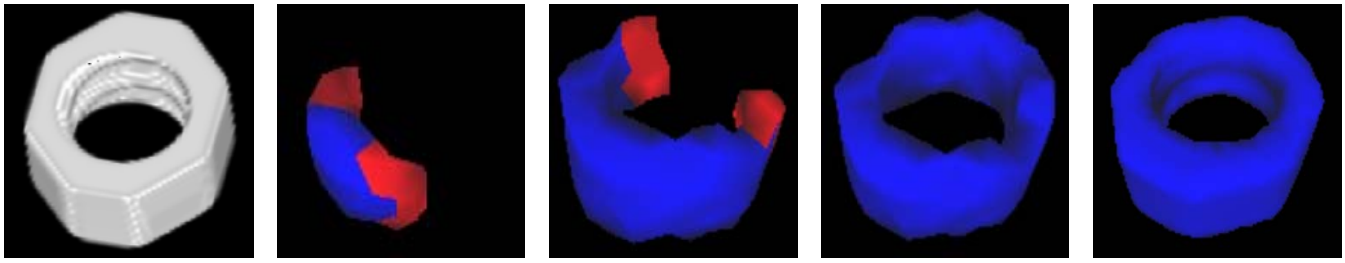
(a) (b) (c) (d) (e)

Figure 8: Surface reconstruction from volumetric image data of a phantom vertebral.

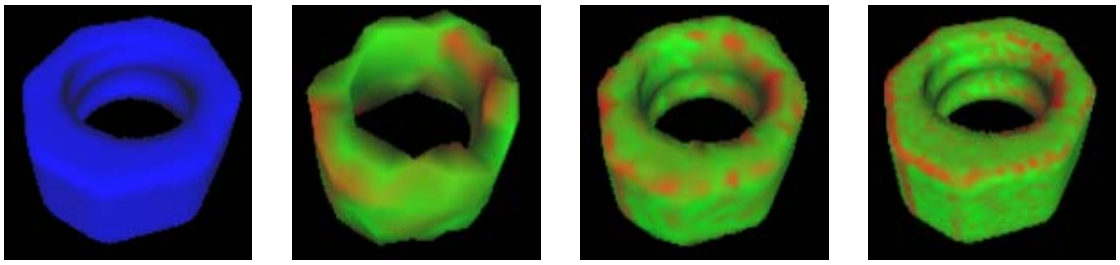


(a) (b) (c) (d) (e)

Figure 9: Surface reconstruction from volumetric image data using multiple seeds.



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



(f) (g) (h) (i)

Figure 10: Surface reconstruction from volumetric image data of a nut along with the corresponding error maps.