

# A Novel Modeling Algorithm for Shape Recovery of Unknown Topology

Ye Duan and Hong Qin

Department of Computer Science, State University of New York at Stony Brook  
Stony Brook, New York, USA 11794

## Abstract

*This paper presents a novel modeling algorithm that is capable of simultaneously recovering correct shape geometry as well as its unknown topology from arbitrarily complicated datasets. Our algorithm starts from a simple seed model (of genus zero) that can be arbitrarily initiated by users within any dataset. The deformable behavior of our model is governed by a locally defined objective function associated with each vertex of the model. Through the numerical computation of function optimization, our 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 conditioned. We have applied our algorithm to various real/synthetic range data as well as volumetric image data in order to empirically verify and validate its usefulness. Based on our experiments, the new modeling algorithm proves to be very powerful and extremely valuable for shape recovery in computer vision, reverse engineering in computer graphics, and iso-surface extraction in visualization.*

## 1. Introduction, motivation, and background

Advances from new imaging modalities such as CT, MRI and Ultrasound as well as other 3-D scanning technologies have given rise to massive volumetric, range, and image datasets available in modern computer era. How to recover and reconstruct the shape of 3-D objects from various datasets accurately and efficiently remains to be non-trivial and challenging for researchers in computer vision. In addition, extracting both meaningful geometry and correct topology from the underlying dataset is fundamental to other vision tasks such as segmentation, reconstruction, recognition, and reasoning. Many algorithms and techniques have been developed to achieve the goal of shape extraction from datasets. In general, existing approaches can be classified as either model-free techniques or model-based approaches. One major

rationale for model-based approaches is that they both provides the great potential for users to effectively interact with the dataset (especially regions of interest) and facilitate other subsequent vision processes such as shape recognition and analysis. Moreover, the inherent continuity and smoothness of the model can compensate for the unwanted sampling artifacts such as noise, gaps, and other irregularities at object boundaries. Hence, model-based approaches are more robust, especially in the existence of noise-corrupted datasets. Among the wide spectrum of model-driven techniques, deformable models [1, 10, 11] have been extremely popular with great success primarily because they offer a unified and powerful approach that combines the knowledge from geometry, physics, approximation theory, and functional analysis. One well-known deformable model is the snake model proposed by Kass, Witkin and Terzopoulos [1]. In essence, a snake is a spline model that can dynamically minimize its potential energy. The total potential energy of the snake model results from three different sources: (1) the internal energy of the spline, (2) image forces, and (3) external constraints. Through minimization of the spline's internal energy, the snake will always remain smooth. The image forces guide the snake toward lines, edges, or other more general low-level features of interest, while the external constraints allow the user to identify specific high-level features to model, recognize, and analyze in the subsequent stages. Later, Miller et al. proposed a polygonal-based deformable balloon model [6, 7]. The behavior of their model is determined by a local cost function associated with each vertex of the model. The cost function is a weighted linear combination of three terms that give rise to: (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 balloon expansion, and (3) a term that constrains the motion of each vertex to remain not too far away from the centroid of its immediate neighbors. Similar to the snake model, the topological variation in Miller et al.'s work is not permitted. The modeled dataset must be homomorphic to a sphere, and the algorithm can only handle volumetric datasets. More recently, Qin and Mandal [8,4] proposed dynamic subdivision surfaces for shape recovery and recognition. Their approaches combine the advantages of free-form

deformable models with the nice properties of subdivision surfaces---smooth limit surfaces with few degrees of freedom. Nonetheless, the topology of these models must be determined before the geometric deformation, i.e., only geometric aspects of the underlying dataset are reconstructed through physics-based simulation. All aforementioned deformable models suffer from one severe limitation: the topology of the underlying shape either is very simple (e.g., genus zero) or must be known a priori (i.e., is determined elsewhere in a separate pre-processing phase) and remain unchanged throughout the time integration of model deformation.

Several researchers have attempted to address this limitation. McInerney and Terzopoulos [5] proposed topological adaptable snake that can overcome this limitation. 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. [9] 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.

In this paper, we develop a new modeling algorithm that can recover both the complicated shape geometry and the arbitrary unknown topology simultaneously from any datasets. The algorithm provides the user a unified approach that not only can deal with both volumetric data and range data, but also is efficient, flexible and powerful. The underlying model is a subdivision-based deformable model that further generalizes the polygonal balloon model of Miller et al. [6,7]. The geometry and the deformable behavior of the model are governed by the principle of energy minimization. When using our algorithm for shape recovery, users can interactively *seed* a simple model at the initialization stage, 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 very effective and can generate a good, high-quality polygonal mesh that can recover both the geometry and the arbitrary topology from any complicated dataset through model deformation.

## 2. 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 that aim to achieve the desired behaviors being simulated in the model. We shall briefly review these four components in Section 2.1 followed by the minimization method in Section 2.2.

### 2.1. Constraints modeling

The cost function  $C_i(x, y, z)$  associated with the current location of a model point is explicitly formulated as  $C_i(x, y, z) = a_0 D(x, y, z) + a_1 B(x, y, z) + a_2 V(x, y, z) + a_3 A(x, y, z)$  (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.

*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.

*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. Note that, volumetric data and range data are treated separately.

For volumetric data, 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).

For range data, however, since there is no grid inherited in the underlying data, the aforementioned method cannot function properly. Instead, we use a distance-based constraint. For each vertex, the algorithm finds out the closest data point to the vertex and calculates the distance. If the distance is smaller than the threshold, the boundary constraint will become very large, otherwise, it is set as zero. Therefore a vertex is not allowed to proceed at a deformation cycle if it is close enough to the boundary of the object. This mechanism will ensure the model to be always inside the range data. The distance threshold used here is the sampling rate of the range data. Intuitively, we can consider the sampling rate as the smallest radius of spheres that are centered at each point of the range data set and can tightly cover the entire boundary area of the modeled object without having any gaps on the surface region.

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. 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.

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 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 which will both be discussed later in this paper, will effectively prevent the local inter-penetration of adjacent faces.

## 2.2. Energy optimization method

An implicit 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 perturb. The amount that a vertex can move is adjusted based upon the current configuration of the cost space. The step size can be reduced four 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.

## 3. 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 change.
7. Global subdivision.

After the model is 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. We have highlights the mechanism of model growing (step three) in the previous section. In this section, we will detail the other six steps of the algorithm.

### 3.1. Model initialization

The *seed* model may be any kind of closed polyhedron. For simplicity and without loss of generality, we use a sphere-like polyhedron consisting of 24 triangles of equal size. The initial position of the *seed* model can be put interactively by users anywhere within the dataset. For volumetric data, the *seed* model does not need to be completely inside the dataset, since the model will flip the

normal tracking direction of the vertex if the vertex is detected to be outside the dataset. In the future, the model initialization should be automated by determining only one voxel of the dataset.

### 3.2. Stressed edge resolution

One phenomenon which oftentimes appears in a polygonal 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 1 demonstrates our method of resolving stressed edges.

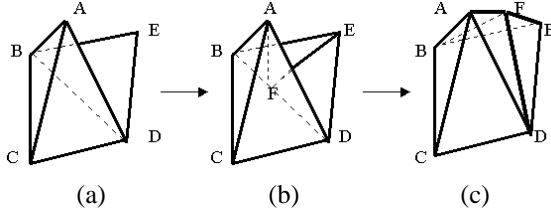


Figure 1: Stressed edge resolution. (a) Edge BD is marked as stressed edge because the dihedral angle between its two adjacent faces ABD and EBD is less than the threshold. (b) Edge BD is split at the middle, and the middle point F of edge BD is connected with vertices A, B, D and E. (c) Finally, F is moved to the middle of vertices A and E.

### 3.3. 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 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 2, 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 split the adjacent edge they share with the central triangle BDE.

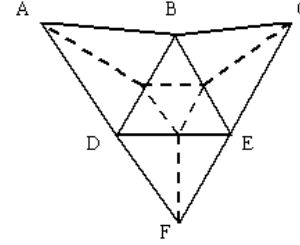


Figure 2: Local adaptive subdivision scheme. The solid lines are the old edges, the dashed lines are the new edges.

### 3.4. 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 ensure the model both locally smooth and globally well conditioned. Three mesh operations: *edge swap*, *edge split*, and *edge collapse* are applied at this step to achieve this goal.

*Edge swap* is employed 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 *Delauny triangulation* at the end of the procedure. That is, it maximizes the minimum angle on all the triangles of the mesh. And it is well known that the best possible surface triangulation over a set of points with known topology is the *Delauny triangulation*. In practice, an edge is swapped only if its local minimum-angle will be increased by certain small minimum. Also, edges with sharp dihedral angle (smaller than 90 degrees in our case) are not allowed to swap. These two conditions will guarantee that the edge-swapping algorithm always functions correctly and terminates eventually.

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 3 illustrates the three mesh operations.

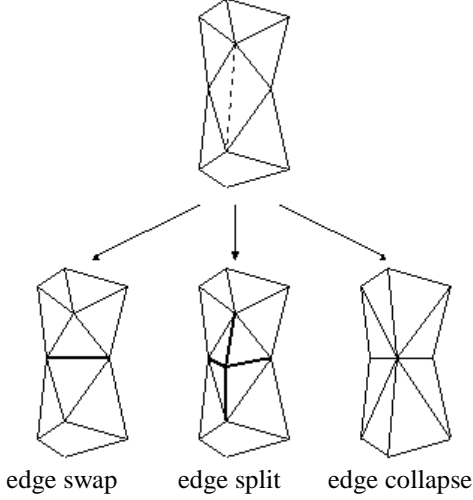


Figure 3: Mesh optimization operations.

### 3.5. Collision detection and topology change

In order to recover 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 4 illustrates the three steps of the scheme: (1) *collision detection*, (2) *identify one-neighborhood and put them into correspondence*, and (3) *change the topology*.

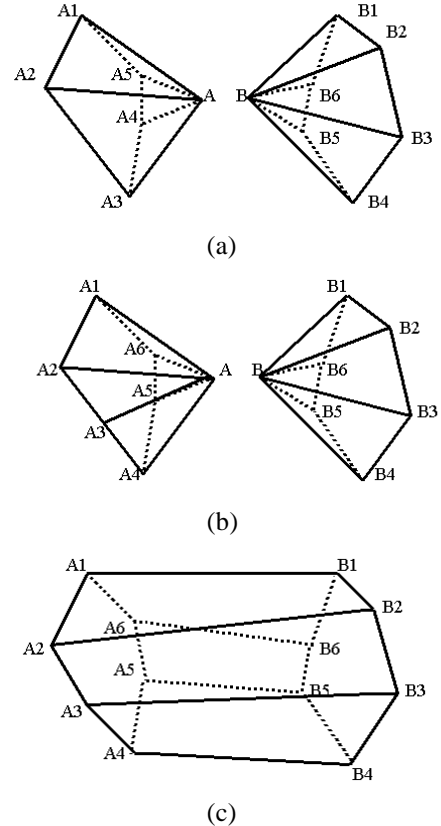
**Collision detection:** If the distance of two non-neighbor active vertices is smaller than the threshold, a collision will be identified and a merge-operation is triggered. If the distance between several pairs of active vertices is smaller than the threshold, the closest pair of vertices is chosen. For example, in Figure 4(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 [12]: 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 Figure 4(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 Figure 4(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 points is connected with its corresponding points in the opposite point set. The two center vertices and all its incident edges are removed (Figure 4(c)). The newly created quadrilaterals are further triangulated by split each quadrilateral into two triangles along one of its diagonal (Figure 4 (d)).

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



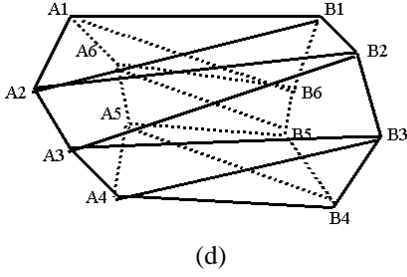


Figure 4: Collision detection and topology change. (a) A collision is detected between the region around vertex A and the region around vertex B. (b) The one-neighborhoods of vertex A and vertex B are put into correspondence. (c) The corresponding vertices between the one-neighborhoods of vertex A and 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.

### 3.6. 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 5 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 Figure 5(a). Each old vertex will generate a new vertex point using the rule shown in Figure 5(b). By connecting each vertex point with its two adjacent edge point 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 and topology change at this stage.

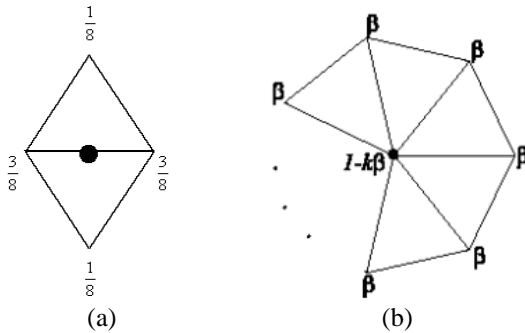


Figure 5: Subdivision rules for Loop's scheme: (a) Edge point rule. (b) Vertex point rule.  $\beta = \frac{3}{8k}$  for  $k > 3$  and

$$\beta = \frac{3}{16} \text{ for } k=3, k \text{ is the valence of the vertex.}$$

## 4. Experimental results

We have built an experimental system using C++ and FLTK. Figure 6 to 9 show some of the experimental results we conducted using this system.

Figure 6 and 7 demonstrate the shape recovery process from volumetric image data. The input dataset of Figure 6 is a synthetic data of a torus with 84 by 23 by 84 voxels. The input dataset of Figure 7 is obtained from scanning a phantom of vertebral with 128 by 120 by 52 voxels. Starting from the leftmost, each figure shows the snapshot of different stages during the deformation process: Fig. 6(a) and Fig. 7(a) are the model initialization stages; Fig. 6(b) and Fig. 7(b) are the model growing stages without any topology changes; Fig. 6(c) and Fig. 7(c) are the first round recovered shape, the topology of the object has been correctly recovered; Fig. 6(d) and Fig. 7(d) are the refined shape after one level of global subdivision, the fitting accuracy of the geometry of the object has been greatly improved. In all figures, red regions represent parts of the model that are still active and growing, blue region represent parts of the model that have already reached the boundary of the object and are not active anymore. The algorithm also supports multiple seed model initialization. For example, in Figure 8(a), four seeds are initialized at different positions at the same time. Each model will grow independently (Fig. 8(b)) and will merge with other models whenever a collision is detected (Fig. 8(c)). Fig. 8(d) shows the more refined model after one level of global subdivision.

Figure 9 and 10 illustrate the shape recovery process from synthetic range datasets. The input dataset of Figure 9 is obtained by sampling a sweeping surface with 10000 data points. The input dataset of Figure 10 is obtained by sampling a subdivision surface with 6140 data points. The leftmost figures (Fig. 9(a) and Fig. 10(a)) are the range data with the seed model inside. The middle two figures (Fig. 9(b), Fig. 9(c) and Fig. 10(b), Fig. 10(c)) are the two snapshots of the model while they are still growing and deforming. The rightmost figures (Fig. 9(d) and Fig. 10(d)) are the final shapes of the models.

Table 1 summarizes the statistics of the examples. The fitting error is calculated by dividing the distance between the model vertex and the boundary of the object by the diameter of the smallest bounding sphere of the object. Table 2 lists the four weighting coefficients for calculating the local cost function associated with each vertex using equation (1). Currently, several parameters need to be set by the user at the beginning of the deformation process.

They are: the face area threshold for local adaptive subdivision, the distance threshold for collision detection, and 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 parameters.

Table 1: Recovered shape information.

Figure #	#Vertices	#Edges	#Faces	Max. fitting error ( % )
6(c)	371	1113	742	1.76
6(d)	1489	4467	2978	1.36
7(c)	1005	3015	2010	0.533
7(d)	4299	8598	12897	0.38
8(c)	2379	4774	7161	1.26
8(d)	9848	29568	19712	0.94
9(d)	5101	15303	10202	1.63
10(d)	821	1650	2475	2.50

Table 2: Weighting coefficients.

$a_0$	$a_1$	$a_2$	$a_3$
1	1	1.6	1

## 5. Conclusion

We have developed a novel modeling algorithm for shape recovery and representation in computer vision. 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. It offers users a unified approach to deal with both volumetric image data and range data. The algorithm can recover the shape of arbitrary geometry and its unknown topology simultaneously. Because the underlying model is a subdivision-based model, it supports levels of detail naturally. After the initial estimation of both topology and geometry of the dataset is accomplished, the user can control the fitting quality easily by specifying the number of levels of global subdivision. Furthermore, the algorithm can be multi-threaded, i.e., multiple seed models can be initialized at different locations at the same time. Throughout the deformation process, each seed model will grow independently and will merge with neighboring models whenever a collision occurs. Several improvements are possible in the near future. The time performance of the algorithm can be further enhanced by employing techniques such as hierarchical bounding box instead of the brute-force searching algorithm currently used for collision detection. Also, the initial position of the seed model should be automatically inferred from data distribution without user intervention.

## References

- [1] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, pages 321-331, 1988.
- [2] Charles Loop. Smooth subdivision surfaces based on triangles. Master's thesis, Department of Mathematics, University of Utah, August 1987.
- [3] 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.
- [4] C. Mandal, B.C. Vemuri, and H. Qin. Shape recovery using dynamic subdivision surfaces. In *Proceedings of IEEE International Conference on Computer Vision (ICCV'98)*, pages 805-810, Bombay, India, January 1998.
- [5] T. McInerney and D. Terzopoulos. Topologically adaptable snakes. In *Proceedings of IEEE International Conference on Computer Vision (ICCV'95)*, pages 840-845, Cambridge, MA, June, 1995.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] 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, New York, NY, June 1993.
- [10] D. Terzopoulos, and K. Fleischer. Deformable models. *The Visual Computer* 4(6), pages 306-331, 1988.
- [11] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-seeking models and 3d object reconstruction. *International Journal of Computer Vision*, pages 211-221, 1987.
- [12] W. Welch and A. Witkin. Free-form shape design using triangulated surfaces. *Computer Graphics (SIGGRAPH'94 Proceedings)*, pages 247-256, July 1994.

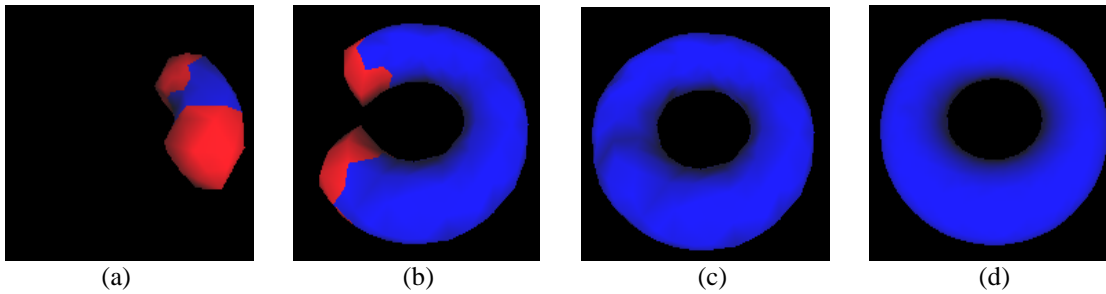


Figure 6: Shape recovery from volumetric image data of a torus.

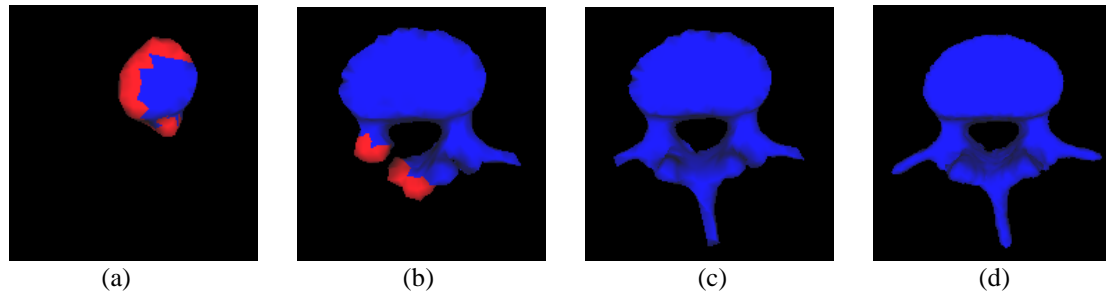


Figure 7: Shape recovery from volumetric image data of a phantom vertebral.

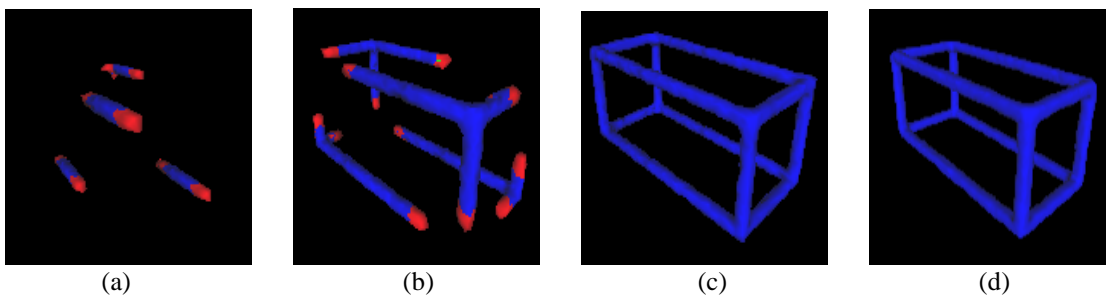


Figure 8: Shape recovery from volumetric image data using multiple seeds.

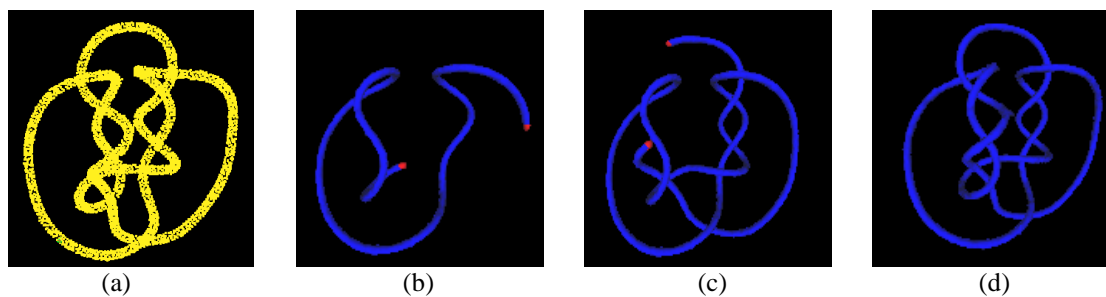


Figure 9: Shape recovery from synthetic range data of a sweeping surface.

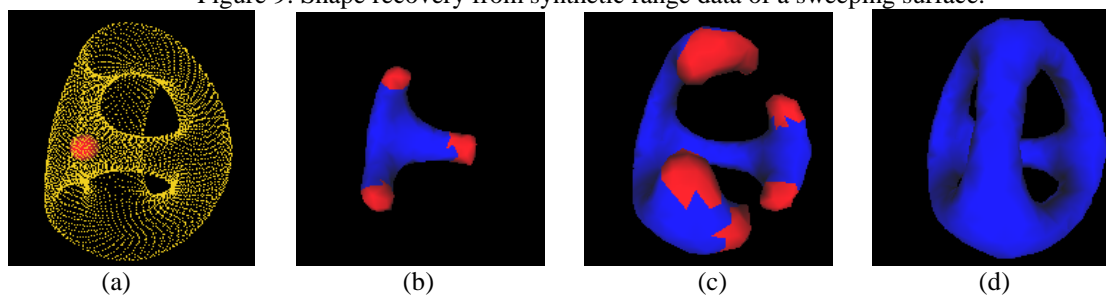


Figure 10: Shape recovery from synthetic range data of a subdivision surface.