Available online at www.sciencedirect.com
SCIENCE DIRECT®
Graphical Models 66 (2004) 181–202
Graphical Models
www.elsevier.com/locate/gmod

A subdivision-based deformable model for surface reconstruction of unknown topology

Ye Duan[a,*] and Hong Qin[b]

[a] *University of Missouri at Columbia, USA*
[b] *State University of New York at Stony Brook, USA*

Received 1 August 2002; received in revised form 9 April 2004; accepted 21 May 2004

Abstract

This paper presents a surface reconstruction algorithm that can recover correct shape geometry as well as its unknown topology from both volumetric images and unorganized point clouds. The algorithm starts from a simple seed model (of genus zero) that can be arbitrarily initiated within any datasets. The deformable behavior of the model is 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 refinement. Commonly used mesh optimization techniques are employed throughout the geometric deformation and topological variation to ensure the model both locally smooth and globally well defined. Our experiments have demonstrated that the new modeling algorithm is valuable for iso-surface extraction in visualization, shape recovery and segmentation in medical imaging, and surface reconstruction in reverse engineering.
Published by Elsevier Inc.

Keywords: Energy optimization; Geometric and topological representations; Biomedical applications; Reverse engineering

* Corresponding author.
E-mail address: duanye@missouri.edu (Y. Duan).

1524-0703/$ - see front matter. Published by Elsevier Inc.
doi:10.1016/j.gmod.2004.05.004

1. Introduction

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 and range 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 visualization, medical imaging, and computer graphics. One of its important applications that have proven to be essential in numerous medical and engineering fields is the non-invasive evaluation of an object's internal structure. For example, it allows the examination of internal organs without operating on the patient and the inspection of mechanical parts without destroying the product.

On the other hand, despite the significant advances of modeling techniques and system functionalities in computer-aided design (CAD) and computer graphics during the past 10 years, current state-of-the-art modeling systems are still lacking some of the unique visual and physical advantages inherent in real-world clay models. As a result, clay models remain to be irreplaceable especially in the presence of 3-D data acquisition technology. In particular, they have been extensively used by engineers in areas such as automotive and aerospace industries. Besides conventional interactive techniques via editing on CAD models, 3-D laser range scanners offer a powerful, alternative means of acquiring geometric models. Small or large-scale objects can be initially sculptured in real world and subsequently scanned into CAD formats for future applications such as manipulation, analysis, and evaluation. In a nutshell, 3-D scanning technology facilitates the process of reverse engineering, i.e., natural and manufactured parts can be digitally converted into CAD systems and then being modified using a range of CAD tools.

At present, many algorithms and techniques have been developed to effectively deal with the acquired datasets 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 [15,29–31] 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.

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, versatile, and powerful. The underlying model is a subdivision-based deformable model that further generalizes the polygonal model of Miller et al. [24,25]. 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 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. 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.

The rest of the paper is organized as follows. The next section summarizes 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 details in Section 4. Section 5 demonstrates the experimental results we obtained using our algorithm. Finally, a conclusion is given in Section 6.

2. Background

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. The static methods can be further classified as methods designed for range datasets and methods designed for volumetric datasets. Among the methods for volumetric datasets, one of the first algorithms was devised by Fuchs et al. [11]. They developed a means of stitching a series of 2-D 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 [18] developed an algorithm called marching cubes that has proven very useful for generating a 3-D 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 3-D 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.

Among various methods for range datasets, one popular algorithm was proposed by Hoppe et al. [12–14]. They first use all the input points to define a signed distance function on R^3, and then interpolate and polygonize the zero-set of this function through the use of the marching-cube algorithm to generate the desirable output mesh. Another type of approaches uses Voronoi diagram and Delaunay triangulation. For instance, Edelsbrunner and Mucke [10] generalize the mathematical notion of convex hull to formally define a family of surfaces based on the input point set. They call the new set of polyhedra α-shapes. A simplex (i.e., edge, triangle, or tetrahedron) belongs to the α-shape if it has some circumsphere with interior empty of sample points, of radius at most α. Therefore, any α-shape consists of a number of appropriate simplices, which can be considered as modeling primitives for the α-shape. The overall shape and its natural dimensionality of the point set can be modified by changing the values of α. Recently, Amenta et al. [1] proposed a new Voronoi-based algorithm called Crust. Using their method, the parameter can be computed automatically for the purpose of shape reconstruction.

In the category of dynamic approaches, the most famous one is the snake model proposed by Kass et al. [15]. 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 et al. [31] generalized the concept of snakes into symmetry-seeking models. They derive a 3-D shape from a 2-D 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 [5], Terzopoulos and Metaxas [30], and McInerney and Terzopoulos [22].

Miller et al. [24,25] later 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 et al. [20,27] proposed dynamic subdivision surfaces for surface reconstruction. 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 new deformable models have been proposed [2,3,6,16,19,23,28,34]. Among them, implicit-function-based methods [3,19,34] are becoming very popular. The key part of these schemes is the modeling of an evolving level set of some implicitly defined functions. Despite the advantages of topological and geometric flexibility, implicit, level-set models are in general computationally more expensive, and are not very easy for user interaction. In a different approach, Szeliski et al. [28] 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. McInerney and Terzopoulos [23] proposed topological adaptable snake, which is a parametric snake 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. Recently, another approach has been chosen by Delingette [6]. He proposed simplex meshes and suggested to use them as a geometric model suitable for deformation.

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. [24,25] and can overcome some limitations of their algorithm. In particular, our technique is capable of automatically change the model's topology during the deformation process. Besides the aforementioned work, two other research advances are also of relevance. One is the work of Welch and Witkin [32,33]. 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. [21]. Their goal is to generate a triangle mesh to approximate the surface implicitly defined by the "skeletons."

The preliminary results of our work have been published in [7–9]. This paper further extends the previous works with several new capabilities. For example, the previously used topology modification algorithm may not work properly in some rare cases if the parameters are not set correctly. In this paper, we have significantly enhanced the topology modification algorithm and now it is much more robust and efficient. After a collision is detected, a preprocessing step is conducted to align the two merging parts so that the two parts will face exactly towards each other. This preprocessing step will ensure the success of the subsequent topology-merge operation. After the topology-merge operation, a post-processing step will proceed

to smooth out the connecting region. The details of the algorithm will be explained in Section 4.

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 model vertex i at location (x,y,z) 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 constraint, $V(x,y,z)$ is the curvature constraint, and $A(x,y,z)$ is the angular constraint. In addition, a_0, a_1, a_2, a_3 are the four corresponding non-negative weighting parameters.

3.1.1. Deformation potential

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). 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 polyhedral surface. Fig. 1 shows a 2-D illustration of the normal tracking method. At the beginning of each deformation, each model vertex P will be assigned a local focal point P_f. The focal point P_f is located at the normal direction of the vertex P and is a constant distance away from the vertex P. The deformation potential associated with the vertex P is then computed as the distance between the vertex P and the focal point P_f. Hence, during each evolving step, every vertex moves in the general direction of the local surface normal to decrease its deformation potential.

During the refinement process (local and global subdivision) which we will discuss in details in Section 4, it is possible that new vertices are added to the model on the opposite side of the data boundary. To move these model vertices 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 vertices is defined to point inwards in the opposite direction. The effect is that a model vertex will migrate towards the true boundary of the object regardless of whether the model vertex is located inside or outside the object boundary. Hence, as long as the initial model intersects the object boundary, i.e., some of the model vertices are inside the object,

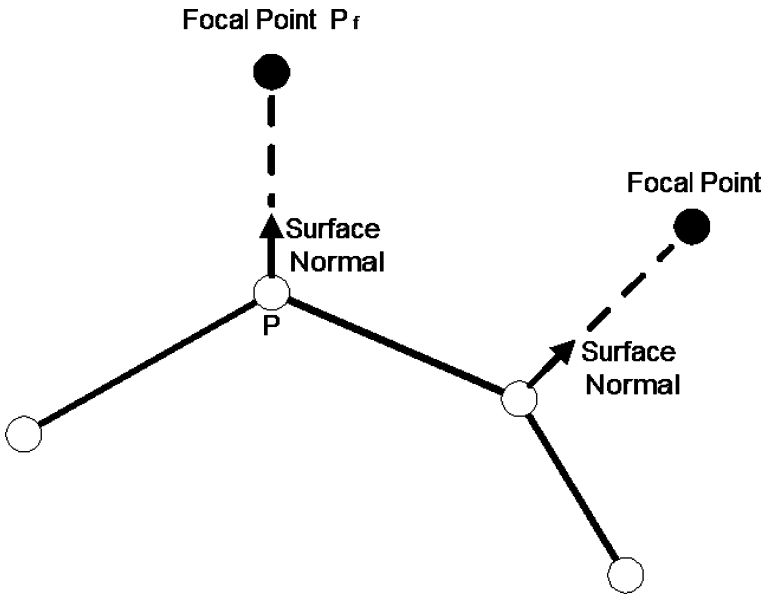

Fig. 1. Deformation potential defined by normal tracking (Section 3.1.1).

the remainders are outside the object, the model tends to seek out the true boundary of the object.

3.1.2. Boundary constraint

Boundary constraint $B(x,y,z)$ affords the mechanism for the model to interact with the data set 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

$$B(x,y,z) = \begin{cases} \text{Image}(x,y,z) - T & \text{if Image}(x,y,z) \geqslant T, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where $\text{Image}(x,y,z)$ is the grey-level intensity of the voxel at location (x,y,z), 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 are no grids 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 vertex will be marked as non-active and is no longer allowed to move. This mechanism will ensure that the model is always inside the range data. The 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.

3.1.3. Curvature constraint

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

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

where (x,y,z) is the position of the current model vertex i, n is the number of neighbors to the current model vertex, $(x_j,y_j,z_j),(x_k,y_k,z_k)$ are the positions of the neighbors of the current model vertex i, where $1 \leqslant j,k \leqslant n$. Curvature constraint $V(x,y,z)$ 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

The fourth constraint—angular constraint $A(x,y,z)$ is used to simulate the effect of attaching a very stiff spring 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, all the edges whose two endpoints are 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 surface reconstruction 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 until 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. Fig. 2 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.

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. The initial position of the seed model can be set interactively by users anywhere within the data set. For volumetric data, the seed model does not need to be completely inside the data set. This is because the model will flip the normal tracking direction of the vertex if the vertex is detected to be outside the data set. Furthermore, for volumetric datasets, the model can be automatically initialized by the system. This is done in a preprocessing step that will search through the input volume dataset and find a non-boundary voxel. This voxel is then identified as the initial center position of the seed model.

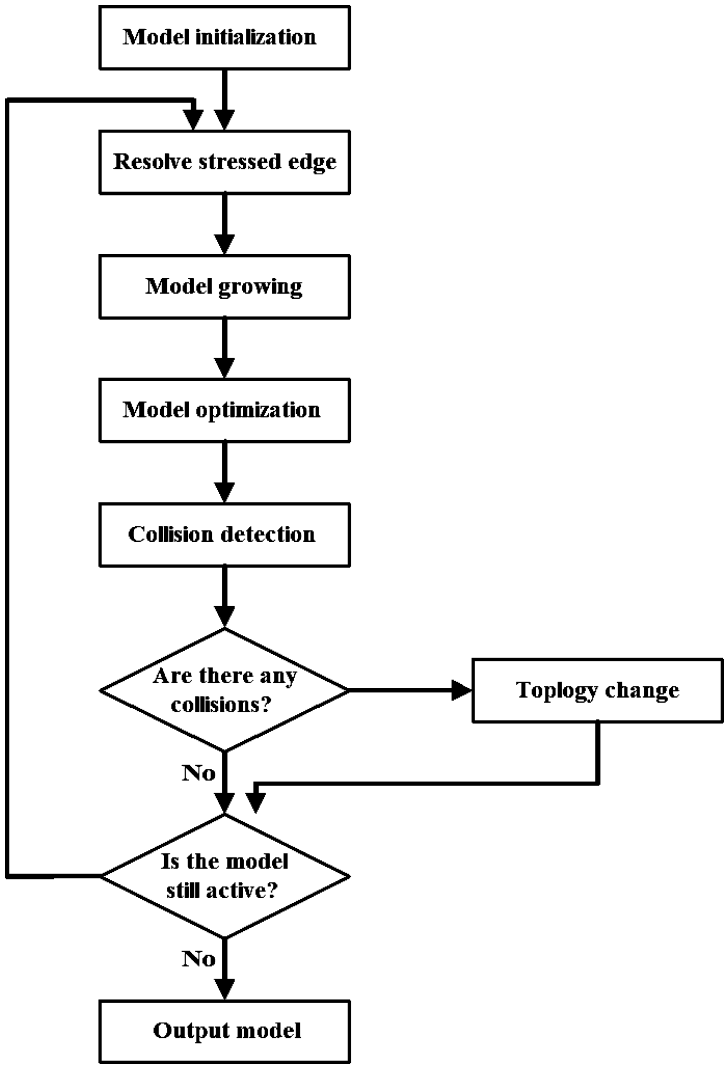

Fig. 2. Pipeline of the algorithm (Section 4).

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 the dihedral angle of its two adjacent faces is less than a certain threshold (we use 60° in our experiments). In this paper, we propose a simple, yet very powerful method that can efficiently solve this problem. At

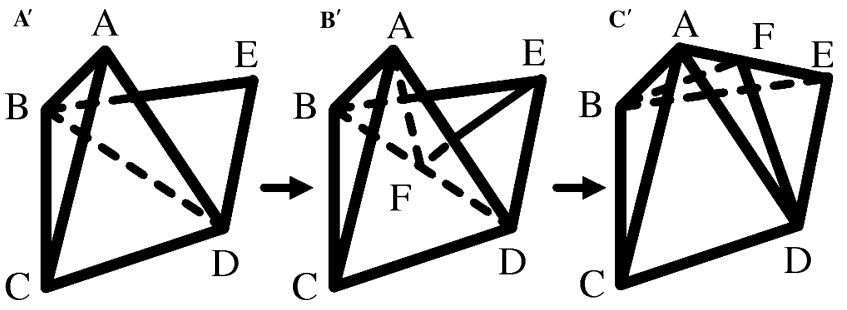

Fig. 3. Stressed edge resolution (Section 4.2). (A′) Edge BD is marked as stressed edge because the dihedral angle between its two adjacent faces ABD and EBD is less than the threshold. (B′) Edge BD is split at the middle, and the middle point F of edge BD is connected with vertices A, B, D, and E. (C′) Finally, F is moved to the middle of vertices A and E.

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 smaller edges at the middle point and the middle point is further moved to the middle position of the two opposite vertices. This is in fact equivalent to an edge flip operation followed by an edge split operation that will discussed in Section 4.4. Fig. 3 demonstrates our method of resolving stressed edges.

4.3. Local adaptive subdivision

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 Fig. 4, to subdivide the central triangle BDE, all three adjacent triangles ADB, CBE, and DFE need to be subdivided as well. Each of these three triangles is subdivided into two smaller ones by splitting the adjacent edge they share with the central triangle BDE.

4.4. Mesh optimization

The algorithm can automatically construct the new subdivision mesh during the deformation phase. Therefore, it is critical to improve and maintain the mesh quality

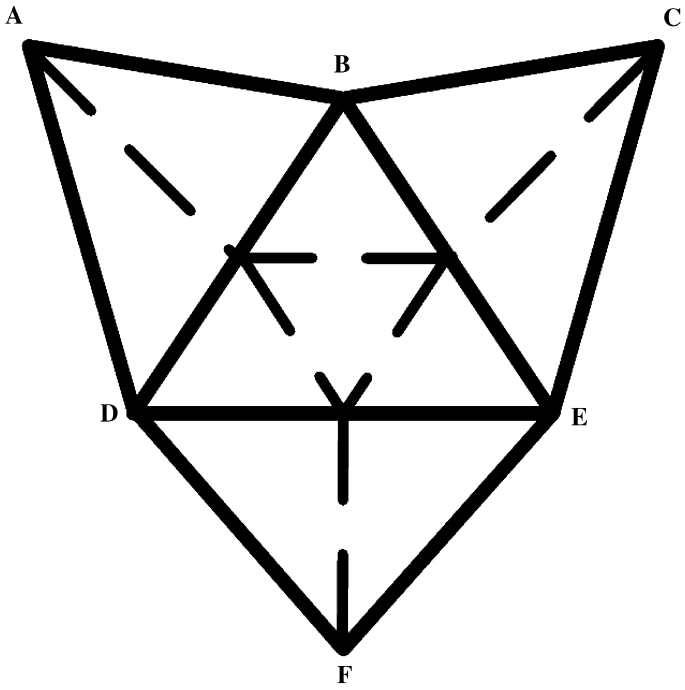

Fig. 4. Local adaptive subdivision scheme (Section 4.3).

throughout the process to keep the model both locally smooth and globally well conditioned. In general, three issues must be considered as suggested by Welch and Witkin [32]: (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 Eq. (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 [4]. 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/or 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. Fig. 5 illustrates the three mesh operations.

4.5. Collision detection and topology changes

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. Instead we use a simple distance based collision detection scheme that is fast and efficient. We will discuss the collision detection scheme in Section 4.5.1. After a collision is detected, a topology-merge operation will be triggered. There are five steps in the topology-merge operation:

(1) Shift the two center vertices into the center of its one-neighborhood.
(2) Align the two one-neighborhoods so that they are facing exactly towards each other.
(3) Put the two one-neighborhoods into correspondence.
(4) Reconnect the two one-neighborhoods.
(5) Smooth out the connecting region by applying the stressed edge resolution scheme.

Fig. 6 illustrates the whole collision detection and topology change algorithm.

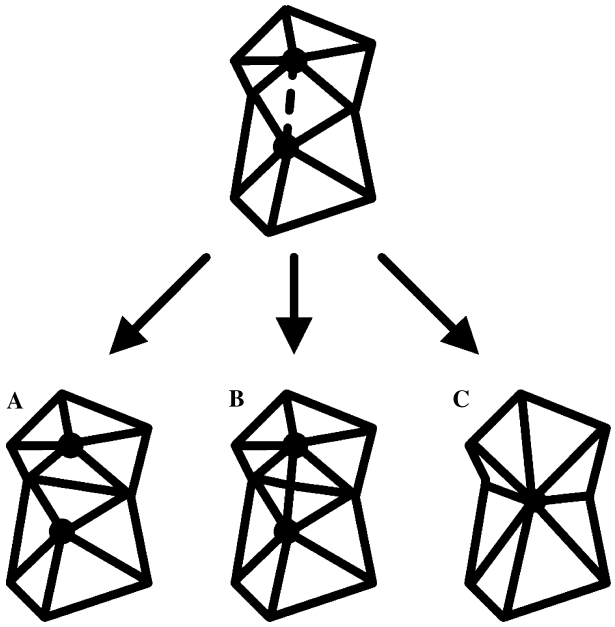

Fig. 5. Mesh optimization operations (Section 4.4).

4.5.1. Collision detection

If the distance of two non-neighboring 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. 6A, 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.

4.5.2. Shift the center vertices

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 we will shift these two vertices to the center of its one-neighborhoods by the following formula:

$$P = \frac{1}{n}\sum_{i=1}^{n} Q_i, \tag{4}$$

where P is the center vertex and $Q_1, \ldots, Q_n$ are its n neighboring vertices. This step will ensure that the two center vertices are in the general position so that the subsequent step of aligning the two one-neighborhoods will proceeds successfully.

4.5.3. Align the two one-neighborhoods

Before the two one-neighborhoods are merged, we will iteratively shift the position of the vertices in the two one-neighborhoods so that the two one-neighborhoods are facing exactly towards each other. This is achieved by applying a

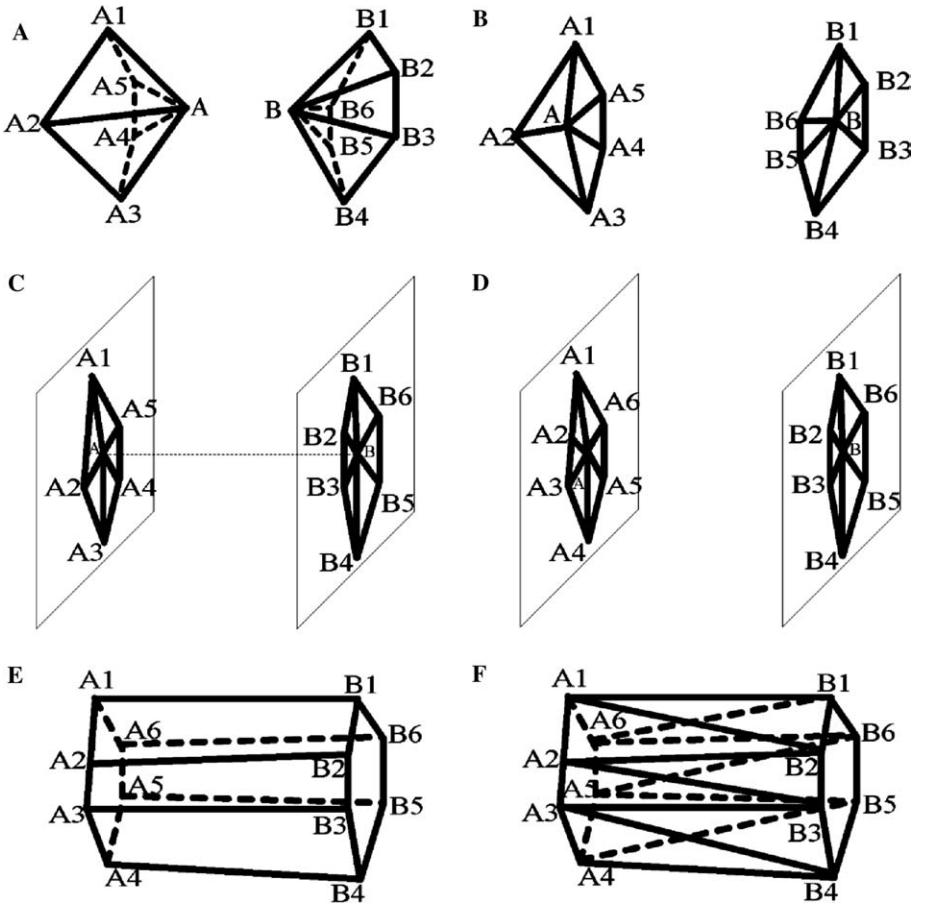

Fig. 6. Collision detection and topology change (Section 4.5).

modified version of the normal diffusion flow recently proposed by Ohtake et al. [26]. To make this paper self-contained, we shall briefly review the normal flow in the following.

The basic idea of the normal flow is to modify the position of each mesh vertex such that the normal of each triangle matches with its corresponding target normal as accurate as possible. For example, in Fig. 7, in order for the triangle normal $n(T)$ becomes closer to the target normal $m(T)$ at the centroid C of each triangle T, each vertex position is updated by the following operator:

$$\begin{aligned} P_{\text{new}} &= P_{\text{old}} + N(P_{\text{old}}), \\ N(p) &= \frac{1}{\sum A(T)} \sum A(T)V(T), \end{aligned} \tag{5}$$

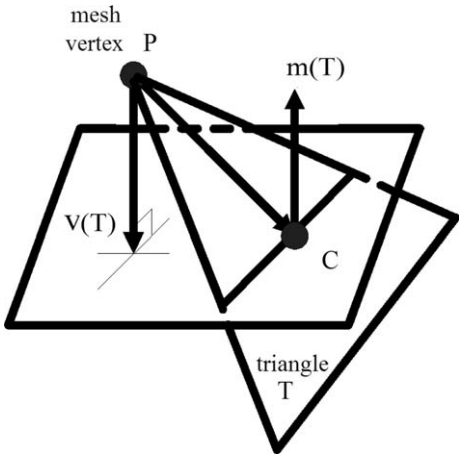

Fig. 7. Update vertex position using the normal flow (Section 4.5.3).

where $V(T) = [\overrightarrow{PC} \cdot m(T)]m(T)$ is the projection of the vector $\overrightarrow{PC}$ along the $m(T)$ direction, $A(T)$ denotes the area of T, and the summations are taken over all P-incident triangles (see Fig. 7).

In our applications, to let the two one-neighborhoods to face towards each other, we apply the aforementioned normal flow to all the vertices in the one-neighborhoods by defining the target normal $m(T)$ at the centers of all the faces in the one-neighborhoods as the normalized vector pointing from the current center vertex towards the opposite center vertex. For example in Fig. 6B, the target normal $m(T)$ of the five faces in the one-neighborhood of the center vertex A is the normalized vector pointing from vertex A to vertex B, same argument holds for the one-neighborhood of center vertex B. A few iterations of the normal flow will quickly align the two one-neighborhoods to face exactly towards each other (Fig. 6C). Since the faces in the one-neighborhood will have the same normal vector, they will stay on the same plane. The planes that contain the two one-neighborhoods will be perpendicular to the vector connecting the two center vertices A and B.

4.5.4. Put the one-neighborhoods into correspondence

After the two one-neighborhoods are aligned to face each other, these two sets of one-neighborhood points will be sequenced separately and put into correspondence. To do so, we use the same procedure as [32]: Iteratively refine the one-neighborhood who has fewer edges by splitting its longest edge until both of the two one-neighborhoods have the same number of nodes, then choose an alignment that minimizes the sum of squared distances between corresponding nodes of the two one-neighborhoods. For example, in Fig. 6C, 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 A1 and A2, 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. 6D, point set A1, A2,..., A5 are corresponding to B1, B2,..., B6, respectively.

4.5.5. Change the topology

After the two sets of one-neighborhood points are put into correspondence, each point is connected with its corresponding point in the opposite one-neighborhood. The two center vertices and all its incident edges are removed (Fig. 6E). The newly created quadrilaterals are further triangulated by splitting each quadrilateral into two triangles along one of its diagonals (Fig. 6F).

4.5.6. Smooth out the connecting region

After the topology-merge operation, the stressed edge resolution scheme described in Section 4.2 will be employed to smooth out any stressed edges that may be generated around the connecting region.

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 [17] in our model though other schemes would also achieve this goal. Fig. 8 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. 8A. Each old vertex will generate a new vertex point using the rule shown in Fig. 8B. By connecting each vertex point with its two adjacent edge points and connect the three edge points with each

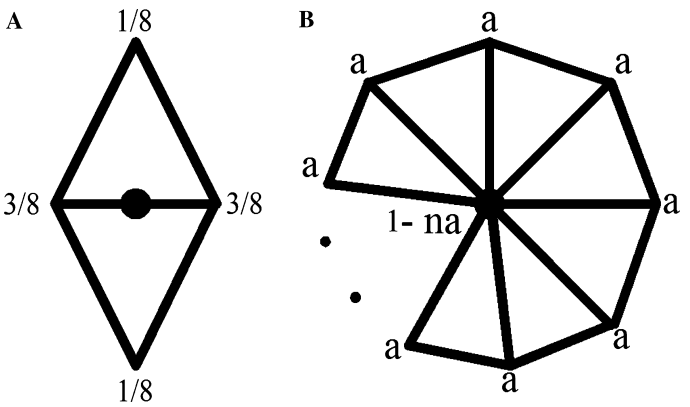

Fig. 8. Subdivision rules for Loop's scheme (Section 4.6). (A) Edge point rule. (B) Vertex point rule. $a = \frac{3}{8n}$ for $n > 3$, and $a = \frac{3}{16}$ for $n = 3$, n is the valence of the vertex.

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.

5. Experimental results

We have developed an experimental system using C++ and FLTK. Figs. 9–13 show some of the experimental results we have conducted using this system.

Figs. 9–11 demonstrate the surface reconstruction process from volumetric image data. In these three figures, (A) shows a volume-rendered image of the original volume datasets. (B and C) Two snapshots of the model during the deformation process. Red color shows the regions of the model that are still active, while the non-active regions of the model are colored in blue. (D) Initial estimation of geometry and topology of the model. (E) 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.

Our algorithm also supports multiple-seed model initialization. For example, in Fig. 11B, four seeds are initialized at four different positions at the same time. Each model will grow independently (Fig. 11C) and will merge with other models whenever a collision is detected (Fig. 11D).

Figs. 12 and 13 illustrate the shape recovery process from range datasets. The input dataset of Fig. 12 is obtained by sampling a subdivision surface with 4348 data points. The input dataset of Fig. 13 is a real range data of 56,340 data points. The leftmost figures (Figs. 12A and 13A) are the range data with the seed model inside. The middle two figures (Figs. 12B and C and 13B and C) are the two snapshots of the model while they are still growing and deforming. Figs. 12D and 13D are the initially recovered shapes. Figs. 12E and 13E are the refined shapes. Figs. 13D and E are shown in wireframe to illustrate the good mesh quality of the model.

Table 1 lists the four weighting coefficients for calculating the local cost function associated with each vertex using Eq. (1). Table 2 gives the information of the recovered shape, such as the number of vertices, edges for each model, the running time, and the maximum fitting error. The running time is measured on an AMD K6 475MHZ Notebook PC with 64 MB internal memory. The fitting error is calculated by dividing the distance between the model vertex and object boundary by the diameter of the smallest bounding sphere of the object.

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

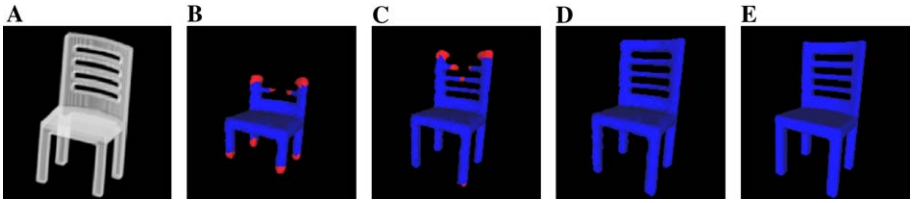

Fig. 9. Surface reconstruction from volumetric image data of a chair.

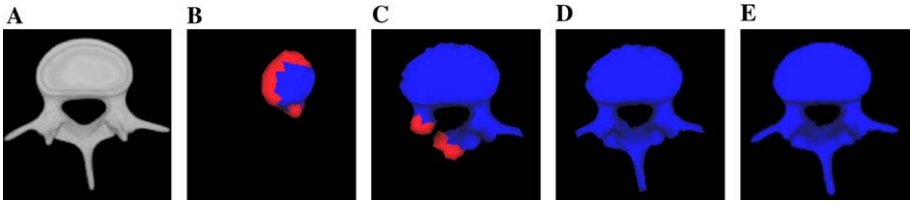

Fig. 10. Surface reconstruction from volumetric image data of a phantom vertebra.

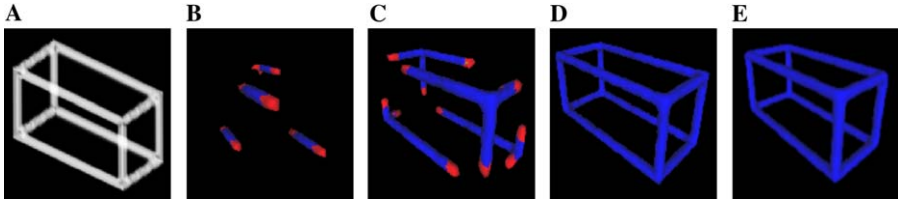

Fig. 11. Surface reconstruction from volumetric image data using multiple seeds.

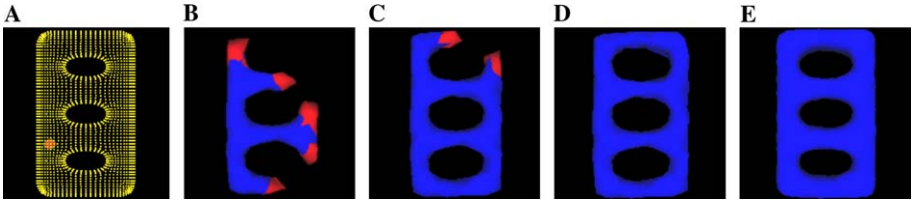

Fig. 12. Shape recovery from synthetic range data.

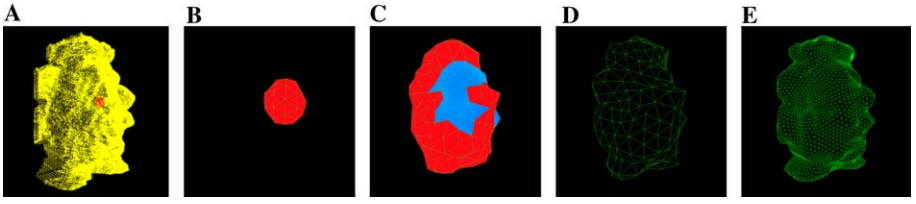

Fig. 13. Shape recovery from real range data.

Table 1
Weighting coefficients

a_0	a_1	a_2	a_3
1	1	1.6	1

Table 2
Recovered model information

Figure #	# Vertices	# Edges	Times (s)	Maximum fitting error (%)
9D	2491	7491	157	1.05
9E	9889	29,685	342	0.92
10D	1005	3015	73	0.533
10E	4299	12,897	147	0.38
11D	2367	7141	96	1.25
11E	9735	29,146	285	0.93
12D	509	1539	41	4.37
12E	2104	6324	139	2.29
13D	270	804	8	6.43
13E	4557	13,665	138	2.68

possible for the algorithm to automatically determine the proper values for these parameters.

6. Discussion

In this paper, we presented a new modeling algorithm that can recover correct shape geometry as well as its unknown topology from either volumetric images or point clouds. Because the underlying model is a subdivision-based model, it naturally supports levels of detail. After the initial estimation of the topology and geometry is achieved, the user can control the fitting quality easily by specifying the number of levels of global subdivision. To further improve the performance, the algorithm can be easily multi-threaded, i.e., multiple-seed models can be initialized at different locations at the same time.

The deformation behavior of the model is guided by the principle of energy minimization. Specifically, each vertex of the model is associated by a local defined cost function $C_i(x, y, z)$ Eq. (1). (1) is in fact a very flexible framework, different cost functions can be employed for different applications. For example, to facilitate user interaction, a point attraction constraint $S(x, y, z)$ can be added in the right-hand side of Eq. 1 to simulate a spring force attached at fixed position (x_0, y_0, z_0)

$$S(x,y,z) = k(||(x,y,z) - (x_0,y_0,z_0)||)^2 \cdot \delta(||(x,y,z) - (x_0,y_0,z_0)||), \tag{6}$$

where δ is a distribution function that is centered at position (x_0, y_0, z_0) with finite support (such as a guassian filter). It is used to control the range of influence of the

spring constraints. The spring constraint $S(x, y, z)$ will allow the user to easily pin point some fixed positions in the domain space to ensure the final shape of the model will interpolate those fixed positions.

We expect our modeling algorithm to be valuable in areas such as visualization, computer graphics, medical imaging, and CAD. 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 volumetric data which is essential in motion tracking).

Acknowledgments

We would like to thank Professor Arie Kaufman for providing the volume rendering facilities in our visualization lab and providing most of the volumetric datasets used in this paper. We thank Professor Tim McInerney for providing the phantom vertebral image. The Potato Head dataset is courtesy of Liu Yang and Professor Dimitris Samaras. We are very grateful for the help from Kevin Kreeger, Ming Wan, Kevin McDonnell, Haixia Du, Jing Hua, Meijing Zhang, Hui Xie, Nan Zhang, Bin Zhang, and Xiaoling Li. We thank for the reviewers for their constructive suggestions.

References

[1] N. Amenta, M. Bern, M. Kamvysselis, A new Voronoi-based surface reconstruction algorithm, in: Computer Graphics (SIGGRAPH'98 Proceedings), July 1998, pp. 415–421.
[2] J. Bredno, T.M. Lehmann, K. Spitzer, A general discrete contour model in two, three, and four dimensions for topology-adaptive multichannel segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 25 (5) (2003) 550–563.
[3] V. Caselles, R. Kimmel, G. Sapiro, Geodisc active contours, in: Proceedings of the Fifth International Conference on Computer Vision (ICCV'95), 1995, June, pp. 694–699.
[4] L. Chew, Guaranteed-quality mesh generation for curved surfaces, in: Proceedings of the Ninth Symposium on Computational Geometry, 1993, pp. 274–280.
[5] L.D. Cohen, I. Cohen, inite element methods for active contour models and balloons for 2D and 3D images, IEEE Trans. Pattern Anal. Mach. Intell. 15 (11) (1993) 1131–1147.
[6] H. Delingette, General object reconstruction based on simplex meshes, Intl. J. Comput. Vision 32 (2) (1999) 111–146.
[7] Y. Duan, H. Qin, Intelligent Balloon: a subdivision-based deformable model for surface reconstruction of arbitrary topology, in: Proceedings of Sixth ACM Symposium on Solid Modeling and Applications (Solid Modeling'01), Ann Arbor, Michigan, 2001, pp. 47–58.
[8] Y. Duan, H. Qin, Extracting Boundary Surface of Arbitrary Topology from Volumetric Datasets. Volume Graphics 2001, Springer, Vienna, June 2001, pp. 237–248.
[9] Y. Duan, H. Qin, A novel modeling algorithm for shape recovery of unknown topology, in: Proceedings of The Eighth IEEE International Conference on Computer Vision (ICCV 2001), Vancouver, Canada, 2001, pp. 402–409.
[10] H. Edelsbrunner, E.P. Mucke, Three-dimensional alpha shapes, ACM Trans. Graph. 13 (1994) 43–72.
[11] H. Fuchs, Z.M. Kedem, S.P. Uselton, Optimal surface reconstruction from planar contours, Commun. ACM 20 (10) (1977) 693–702.
[12] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, in: Computer Graphics (SIGGRAPH'92 Proceedings), July 1992, pp. 71–78.

[13] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Mesh optimization, in: Computer Graphics (SIGGRAPH'93 Proceedings), August 1993, pp. 19–26.
[14] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. jin, J. McDonald, J. Schweitzer, W. Stuetzle, Piecewise smooth surface reconstruction, in: Computer Graphics (SIGGRAPH'94 Proceedings), July 1994, pp. 295–302.
[15] M. Kass, A. Witkin, D. Terzopoulos, Snakes: active contour models, Intl. J. Comput. Vision 1 (4) (1988) 321–331.
[16] J.-O. Lachaud, A. Montanvert, Deformable meshes with automated topology changes for coarse-to-fine 3D surface extraction, Med. Image Anal. 3 (2) (1999) 187–207.
[17] Charles Loop, Smooth subdivision surfaces based on triangles. Master's thesis, Department of Mathematics, University of Utah, August 1987.
[18] W.E. Lorensen, H.E. Cline, Marching cubes: a high resolution 3D surface construction algorithm, in: Computer Graphics (SIGGRAPH'87 Proceedings), July 1987, pp. 163–169.
[19] R. Malladi, J. Sethian, B. Vemuri, Shape modeling with front propagation: a level set approach, IEEE Trans. Pattern Anal. Mach. Intell. 17 (2) (1995) 158–175.
[20] C. Mandal, H. Qin, 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), Ann Arbor, Michigan, June 1999, pp. 191–202.
[21] L. Markosian, J.M. Cohen, T. Crulli, J.F. Hughes. Skin: a constructive approach to modeling free-form shapes, in: Computer Graphics (SIGGRAPH'99 Proceedings), August 1999, pp. 393–400.
[22] T. McInerney, D. Terzopoulos, A dynamic finite element surface model for segmentation and tracking in multidimensional medical images with applications to cardiac 4D image analysis, Comput. Med. Imaging Graphics 19 (1) (1995) 69–83.
[23] T. McInerney, D. Terzopoulos, Topology adaptive deformable surfaces for medical image volume segmentation, IEEE Trans. Med. Imaging 18 (10) (1999) 840–850.
[24] 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.
[25] J.V. Miller, D.E. Breen, W.E. Lorensen, R.M. O'Bara, M.J. Wozny, Geometric deformed models: a method for extracting closed geometric models from volume data, in: Computer Graphics (SIGGRAPH'91 Proceedings), July 1991, pp. 217–226.
[26] Yu. Ohatake, A.G. Belyaev, A. Pasko, Dynamic meshes for accurate polygonization of implicit surfaces with sharp features, in: Shape Modeling International 2001, Genova, Italy, May 2001, pp. 74-81.
[27] H. Qin, C. Mandal, B.C. Vemuri, Dynamic Catmull–Clark subdivision surfaces, IEEE Trans. Visualization Comput. Graphics 4 (3) (1998) 215–229.
[28] R. Szeliski, D. Tonnesen, D. Terzopoulos, Modeling surfaces of arbitrary topology with dynamic particles, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'93), June 1993, pp. 82–87.
[29] D. Terzopoulos, K. Fleischer, Deformable models, Visual Comput. 4 (6) (1988) 306–331.
[30] D. Terzopoulos, D. Metaxas, Dynamic 3D models with local and global deformations: deformable superquadrics, IEEE Trans. Pattern Anal. Mach. Intell. 13 (7) (1991) 703–714.
[31] D. Terzopoulos, A. Witkin, M. Kass, Symmetry-seeking models and 3D object reconstruction, Intl. J. Comput. Vision 1 (3) (1987) 211–221.
[32] W. Welch and A. Witkin, Free-form shape design using triangulated surfaces, in: Computer Graphics (SIGGRAPH'94 Proceedings), July 1994, pp. 247–256.
[33] W. Welch, A. Witkin, Serious Putty: topological design for variational curves and surfaces. PhD thesis, Carnegie Mellon University, June 1995.
[34] R.T. Whitaker, A level-set approach to 3D reconstruction from range data, Intl. J. Comput. Vision 29 (3) (1998) 203–231.