Intelligent Balloon: A Subdivision-Based Deformable Model For Surface Reconstruction Of Arbitrary Topology

Ye Duan   Hong Qin

Department of Computer Science
State University of New York at Stony Brook
{yduan | qin}@cs.sunysb.edu

Abstract

In this paper, we develop a novel subdivision-based model---Intelligent Balloon---which is capable of recovering arbitrary, complicated shape geometry as well as its unknown topology simultaneously. Our Intelligent Balloon is a parameterized subdivision surface whose geometry and its deformable behaviors are governed by the principle of energy minimization. Our algorithm starts from a simple seed model (of genus zero) that can be arbitrarily initiated by users within regions of interest. The growing behavior of our model is controlled by a locally defined objective function associated with each vertex. Through the numerical integration 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), correctly evolve the model towards the region 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 topologically flexible models to such applications as reverse engineering from range data and surface reconstruction from volumetric image data. Our new models prove to be very powerful and extremely useful for boundary representation of complicated solids of arbitrary topology, shape recovery and segmentation for medical imaging, and iso-surface extraction for visualization.

Keywords: Energy Optimization, Geometric and topological representations, Biomedical applications, Reverse engineering.

1 INTRODUCTION

Despite the significant advances of modeling techniques and system functionalities in Computer-Aided Design (CAD) and computer graphics during the past ten years, current state-of-the-art modeling systems are still lacking of some of the unique visual and physical advantages inherent in real-world clay models. As a result, clay models remain to be unreplacable especially in the presence of 3-D data acquisition technology. In particular, they have been extensively used by engineers in areas such as automobile design. 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.

On the other hand, recent hardware advances based on new imaging modalities such as CT, MRI and Ultrasound as well as sensoring techniques have been able to generate more volumetric, range and image data than what scientists and engineers can handle. It remains a challenging task for researchers and practitioners on how to model, manipulate and display the tremendous amount of data in a useful and intuitive way in shape modeling, CAD, visualization, and medical imaging. To date, many algorithms and techniques have been developed to effectively deal with the acquired data for modeling and rendering applications. In general, existing approaches can be roughly 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 modeling for shape reconstruction and recognition. Among a wide array of model-driven techniques, deformable models have been extremely popular with great success primarily because they offer a unified and powerful approach that can combine the knowledge from geometry, physics and approximation theory to tackle many challenging problems and offer an elegant solution for many applications. 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 phase) and remain unchanged throughout the time integration of model deformations.

By contrast, our Intelligent Balloon model can be used to overcome this limitation. It can recover both complicated shape geometry and its arbitrary, unknown topology simultaneously. It provides user a unified approach that can deal with both volumetric data and range data, therefore our new model is efficient, flexible and powerful. Moreover, Intelligent Balloon is a subdivision-based deformable model. In using our system for reverse engineering, 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 matured mesh optimization techniques, has proven to be effective and can generate a good, high-quality polygonal mesh, which 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. We briefly discuss related work in Section 2. Section 3 reviews the principle of energy minimization techniques that govern the deformable behavior of our Intelligent Balloon towards the correct recovery of both geometry and topology. Section 4 details all key components of our algorithm for the Intelligent Balloon model. Experimental results are shown in section 5. Finally, we conclude our paper in Section 6 and point out possible future directions in Section 7.

2 RELATED WORK

During the recent years, a lot of research has been conducted in the areas of surface reconstruction, reverse engineering, and shape recovery in medical imaging. The majority of the published results falls into two groups: static, geometric techniques and dynamic, energy-based techniques. Among the static methods, one popular algorithm was proposed by Hoppe et al. [3,4,5]. 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 static approaches uses *Voronoi diagram* and *Delaunay triangulation*. For instance, Edelsbrunner et al. [2] 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, Witkin and Terzopoulos [6]. 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. Recently, McInerney and Terzopoulos [11] extended the snake model to be able to recover 3D shapes of arbitrary topology. The basic idea of their method is to superimpose a simplical grid on the image domain and iteratively reparameterize the geometry of deforming snakes. 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 balloon 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, and the algorithm only deal with the volumetric data type. In contrast, Our Intelligent Balloon model further generalizes their work and can overcome some limitations of their algorithm. In particular, our technique is capable of recovering geometric shape of arbitrary, unknown topology from either volumetric data or range data. Recently, Qin and Mandal [14, 9] proposed dynamic subdivision surfaces for surface reconstruction. 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. In addition, their algorithm allows the direct manipulation of the limit surfaces defined by the subdivision process on the initial control mesh. When applying the dynamic subdivision surfaces to solve shape reconstruction problems, however, 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.

Our Intelligent Balloon is a polygonal model with the capability of recursive refinements through surface subdivision. It can automatically construct the new subdivision mesh during the deformation phase. Besides the aforementioned work, two other research advances are also of relevance. One is the work of Welch and Witkin [16,17]. 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. [10]. Their goal is to generate a triangle mesh to approximate the surface implicitly defined by the "skeletons".

3 GEOMETRIC CONSTRAINTS AND ENERGY-BASED OPTIMIZATION

The deformable behavior of our Intelligent Balloon is governed by the principle of energy-based minimization. A locally defined cost function is associated with each vertex of the polygonal model. Through the minimization process of the cost function, our model will inflate like a typical balloon until it reaches the boundary of the modeled objects represented by the data set. In this paper, the cost function is a weighted linear combination of four constraints that are selected to achieve the desired behaviors being simulated in the model. We shall briefly review these four components in Section 3.1 followed by the minimization algorithm 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 -- $D(x,y,z)$

The 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 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

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

where $Image(x,y,z)$ is the grey-level intensity distribution of the voxel at (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 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 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 -- $V(x,y,z)$

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 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, $1 \le j, k \le n.$ 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 Section 4 later.

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

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

3.2 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 reconstruction. A vertex of the model will move along the direction of 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 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 marked as non-active and is not allowed to move any further.

4 INTELLIGENT BALLOON MODEL

The previous section highlights the key mathematical concept of balloon-like deformable models---defining and associating a local cost function with each vertex of the underlying polygonal models. Through the minimization of the cost function, the model can grow until it reaches the boundary of the object.

In order to recover shape of arbitrary, unknown topology, we generalize the balloon-like deformable models with additional capabilities. Our Intelligent Balloon model is a subdivision-based deformable model. Our model is initialized interactively by users as a very simple *seed* model that consists of a small number of faces (e.g., a cube). Following the previous discussion, our model can inflate according to its local cost function associated with each vertex of the model. In addition, local adaptive subdivision schemes can be applied on the model in order to increase the degrees of freedom and make our model highly flexible for various complicated geometric data sets while ensuring the tight control on the fitting error. Most importantly, our model has the ability to automatically detect self-collision and modify its topology in accordance with the data set. Meanwhile, mesh optimization techniques are used throughout the deformation process to maintain the good quality of the mesh. Figure 1 shows the flowchart of our algorithm.

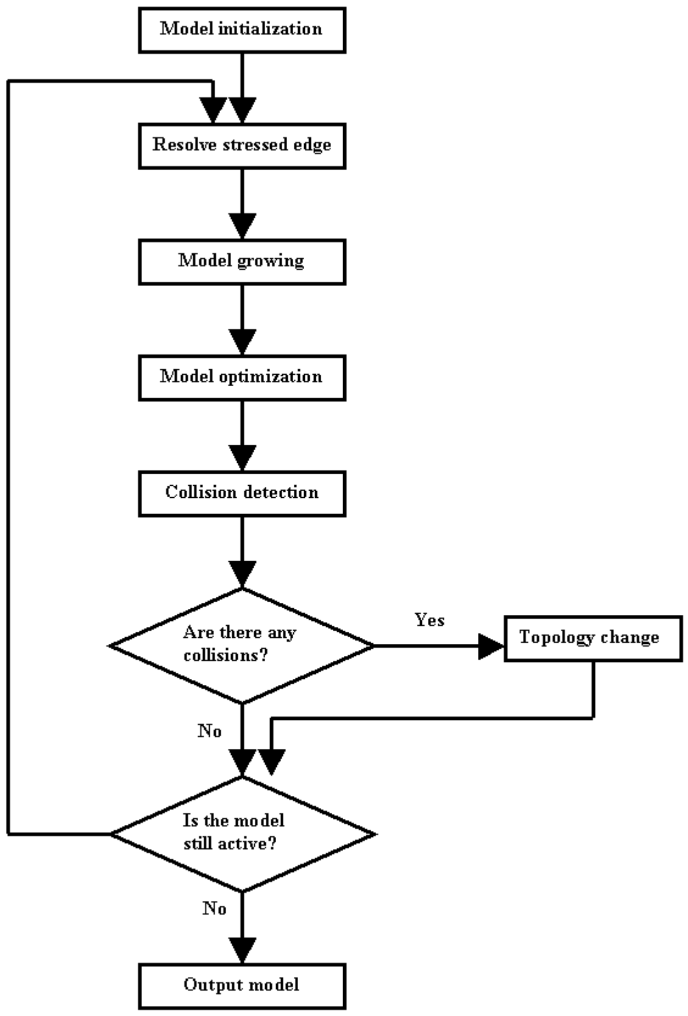


Figure 1: Overview of the Intelligent Balloon model.

4.1 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 data set. Furthermore, 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.

4.2 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 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 splitting the adjacent edge they share with the central triangle BDE.

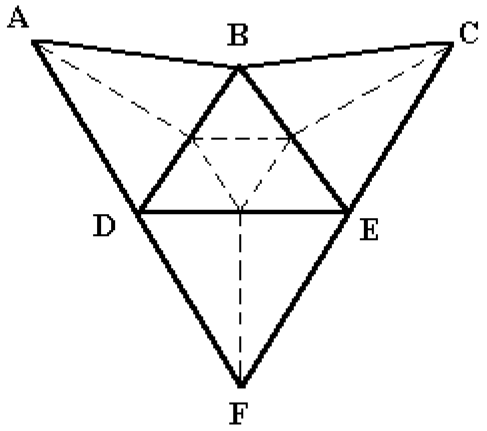


Figure 2: Local adaptive subdivision scheme. The solid lines are the old edges, the dashed lines are the new edges. The center triangle BDE is divided into 4 smaller triangles by connecting the three middle points of the old edges. Each of the three adjacent triangles ADB, CBE and DFE is split into two smaller triangles.

4.3 Improving Mesh Quality

The Intelligent Balloon is a polygonal based model. Therefore, it's critical to improve and maintain the mesh quality throughout the deformation 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.

4.3.1 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 for the Intelligent Balloon, 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 $C_i(x,y,z)$ 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.

4.3.2 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.

4.3.3 Node 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, we also need to perform the following two operations: (1) edge split and (2) edge collapse. First, all the edge lengths are measured in 3-D, and 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. 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.

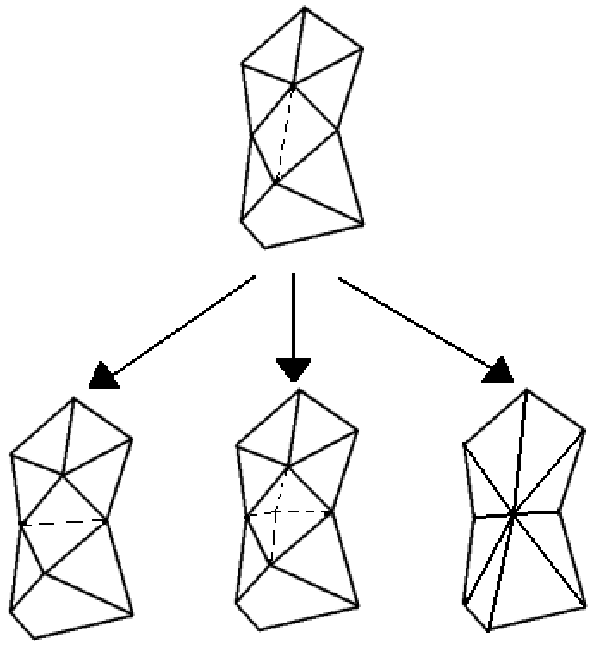


Figure 3: Mesh optimization techniques.

4.4 Prevention of Local Inter-penetration

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). Marksoian et al. [10] solve this problem by passively registering the collision points and letting other non-stressed edges continue to evolve. Eventually, the creases will be resolved by themselves. This method works very well within interactive design environment. But it is not very applicable to our purpose for shape reconstruction and reverse engineering. We propose a more aggressive approach, which, when combined with the aforementioned mesh optimization techniques, can significantly prevent the occurrence of local inter-penetrations.

The idea is to resolve stressed edges whenever they appear. Before each step of deformation, we calculate the dihedral angle of the two adjacent faces for all the active edges, mark all the edges whose dihedral angle is less than certain threshold as stressed edges. Each stressed edge is split into two small edges at the middle point. Then the middle point is moved to the middle position of the two opposite vertices. If the two adjacent faces of the stressed edge are very close to each other, then at the following mesh optimization steps, the new edges will be collapsed, i.e., the two adjacent faces will be merged. Figure 4 demonstrates our method of resolving stressed edges. Edge BD is marked as the stressed edge because the dihedral angle between its two adjacent faces ABD and EBD is smaller than the threshold (Fig. 4(a)). Thus it is split at its middle point F and is connected with the four neighbor vertices A, B, D and E (Fig. 4(b)). Finally, F is moved to the middle position of the two opposite vertices A and E (Fig. 4(c)).

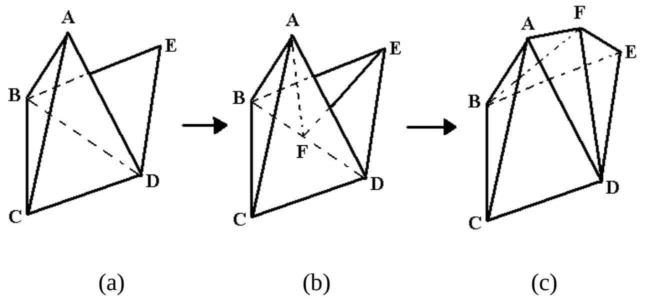


Figure 4: Resolve stressed edges. (a) Edge BD is marked as stressed edge because the dihedral angle between its two adjacent faces ABD and EBD is less than the threshold. (b) Edge BD is split at the middle, and the middle point F of edge BD is connected with vertices A, B, D and E. (c) Finally, F is moved to the middle of vertices A and E.

4.5 Collision Detection and Topology Changes

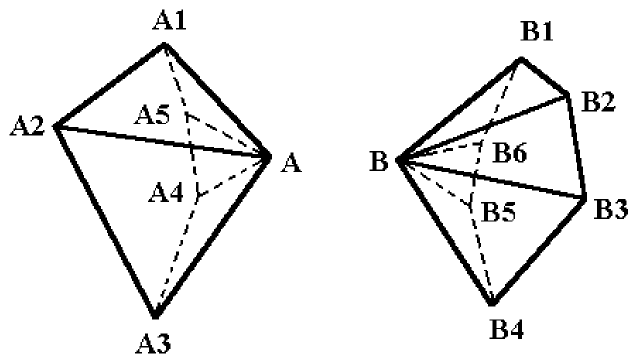
In order to recover a shape of arbitrary, unknown topology, the model must be able to change its topology properly whenever a collision with other parts of the model is detected. Various kinds of collisions can be considered, such as face-to-face, edge-to-edge, vertex-to-vertex, edge-to-face, etc. Techniques 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 shows the following three steps of the algorithm: (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.

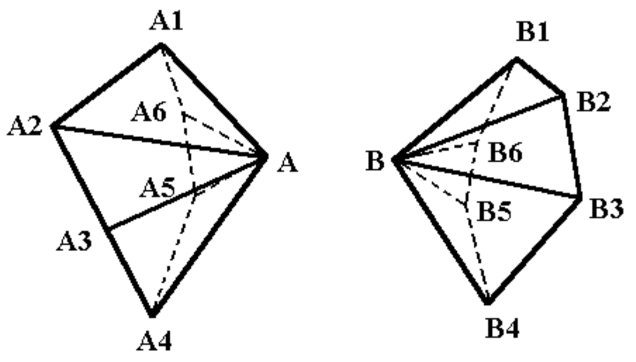
Identify one-neighborhoods and put them into correspondence: To merge the two parts of the model. First, we need to identify and collect all the one-neighborhood points for each of these two vertices. Then these two sets of points (i.e., one-neighborhood points) are sequenced separately and are put into correspondence. To do so, we use the same procedure as Welch and Witkin [16]: 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, …, A6} 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)).

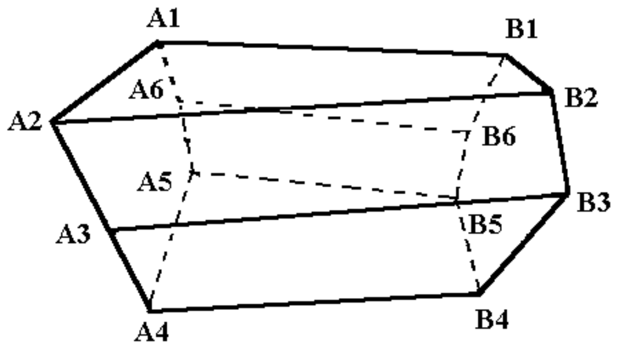
The mesh optimization processes will quickly smooth out any artifacts that may result from the matching procedure once the merge has been completed.



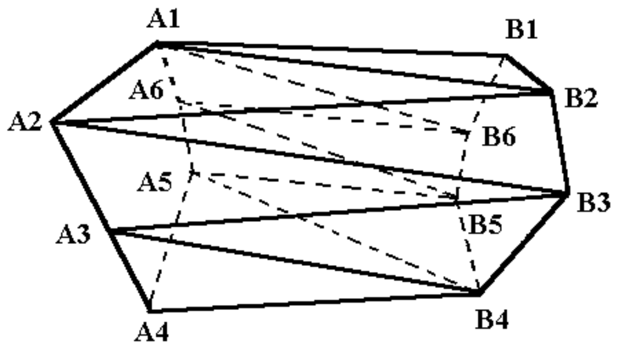
(a)



(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 Level of Detail

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 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. Note that, 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.

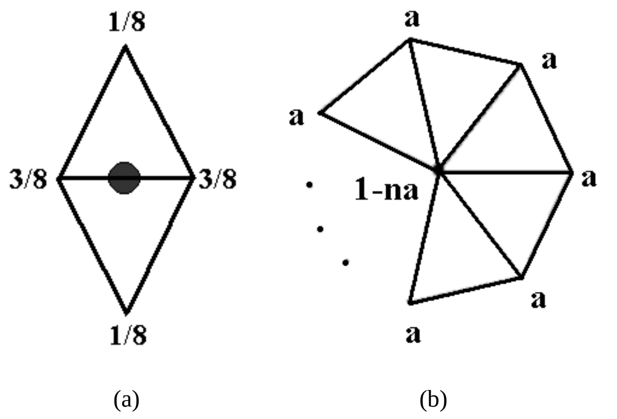


Figure 6: Subdivision rules for Loop's scheme: (a) Edge point rule. (b) Vertex point rule. $a = \frac{3}{8n}$ for n > 3 and $a = \frac{3}{16}$ for n = 3, n is the valence of the vertex.

5 RESULTS

We have built an experimental system of the Intelligent Balloon model using C++ language and FLTK. Figure 7 to 10 show some of the experiments we have conducted using this system.

The input data set of Figure 7 is a synthesized range data with three holes. Fig. 7(a) shows the initialized model within the range data. Fig. 7(b) and Fig. 7(c) are two snapshots of our model during the deformation process. Fig. 7(d) is the final shape. Fig. 7(e) is the refined shape after one level of global subdivision using Loop's scheme. By comparing Fig. 7(d) and Fig. 7(e), we can clearly see the improvement of the fitting accuracy.

Figure 8 shows our experiment on a synthesized range data which is acquired from a torus. Fig. 8(a) shows the initialized model within the range data, followed by a growing model to its right (Fig. 8(b)). Fig. 8(c) represents the final shape after the recovery of the topological information. Fig. 8(d) illustrates the more refined model after one level of subdivision.

Figure 9 illustrates the shape reconstruction process from a volumetric image data with six holes. Fig. 9(a) and Fig. 9(b) are the snapshots of our model while they are still growing and deforming, and the final shape estimation is shown on Fig. 9(c). Fig. 9(d) shows the more refined model after one level of subdivision. Fig. 9(e) is the volume rendering of the volumetric image data with six holes.

Figure 10 demonstrates our experiment on a volumetric image data which is acquired from a nut. Fig. 10(a) shows the still growing model, followed by the final shape on Fig. 10(b). Fig. 10(c) and Fig. 10(d) are the refined shape after one and two levels of global subdivision. The groves on the inner-surface of the nut are recovered very well (Fig. 10(d)). The volume rendering of the nut is shown on Fig. 10(e). Fig. 10(f), Fig. 10(g) and Fig. 10(h) are the error maps of the final shape of Fig. 10(b), Fig. 10(c) and Fig. 10(d), 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 illustrates 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 mainly the regions around the sharp edges and corners.

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 the examples. In particular, Table 2 summarizes the information of the input data, including data type and the data size. Table 3 gives the information of the recovered shape, such as the number of vertices, edges and faces for each model, the running time, and the maximum fitting error. The running time is measured on an AMD K6 475MHZ Notebook PC with 64MB internal memory.

a_0	a_1	a_2	a_3
1	1	1.6	1

Table 1: Weighting coefficients.

Figure #	Data Type	Data Size
7	Synthesized range data	4348 points
8	Synthesized range data	6400 points
9	Volumetric image data	32 x 32 x 64 voxels
10	Volumetric image data	69 x 41 x 59 voxels

Table 2: Input data information.

Figure#	#Vertices	#Edges	#Faces	Time (sec)	Fitting error(%)
7(d)	509	1539	1026	41	4.37
7(e)	2104	6324	4216	139	2.29
8(c)	211	633	422	30	4.80
8(d)	924	2772	1848	125	2.79
9(c)	2379	4774	7161	120	1.26
9(d)	9848	29568	19712	315	0.94
10(b)	172	344	516	24	2.79
10(c)	756	1512	2268	31	2.03
10(d)	3053	6106	9159	218	1.85

Table 3: Recovered shape information.

6 CONCLUSIONS

We have developed and presented a novel subdivision based deformable model—Intelligent Balloon. Our Intelligent Balloon can overcome several limitations of the conventional deformable models. It offers users a unified approach to deal with both volumetric data and range data. By using a novel distanced based collision detection scheme, Intelligent Balloon can recover the shape of arbitrary geometry and unknown topology simultaneously. We also proposed a novel technique to prevent any local inter-penetration by aggressively resolving all the stressed edges whenever they appear. Our algorithm, combined with mesh optimization techniques, can guarantee a very high quality mesh during the deformation process. Since our model is a subdivision-based model, it supports levels of detail naturally. After an initial estimation of both topology and geometry of the data set is accomplished, the user can control the levels of detail easily by specifying the number of levels of global subdivision. Our model can also be paralleled, i.e., multiple seed models can be placed at different positions at the same time, and each seed model will grow on its own and will merge with other models whenever a collision is detected. This will speed up the algorithm and is especially useful in areas such as medical imaging when a priori knowledge about the data set is always available.

7 FUTURE WORK

Several improvements are possible. First, we currently use a brute-force searching algorithm for collision detection and the closest-point searching algorithm for the range data. We shall continue to improve their time performance by using techniques such as hierarchical bounding box. Second, our current algorithms require users to interactively select a few parameters during the initialization stage. 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 data set and its property. It would be ideal to refine 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. Furthermore, the initial position of the seed model ought to be chosen automatically.

In addition, we plan to extend the functionality of our model and its associated system in the following directions in the future. First, we shall enhance our model so that it can recover sharp features such as corners and creases. Second, 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). It is essential in solid modeling to obtain a more concise, spline-like representation of the modeled data. Third, we shall apply the techniques and algorithms of our Intelligent Balloon to other visual computing fields such as interactive design and engineering analysis.

ACKNOWLEDGMENTS

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 making the volumetric data sets (used in this paper) available to us. We also thank Kevin Kreeger and Michael Ming Wan for their valuable assistance on dealing with the volumetric data.

REFERENCES

[1] N. Amenta, M. Bern and M. Kamvysselis. A new Voronoi-based surface reconstruction algorithm. Computer Graphics (SIGGRAPH'98 Proceedings), pages 415-421, July 1998.

[2] H. Edelsbrunner and E.P. Mucke. Three-dimensional Alpha Shapes. ACM Transactions on Graphics 13:43-72, 1994.

[3] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. Computer Graphics (SIGGRAPH'92 Proceedings), pages 71-78, July 1992.

[4] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. Computer Graphics (SIGGRAPH'93 Proceedings), pages 19-26, August 1993.

[5] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. Computer Graphics (SIGGRAPH'94 Proceedings), pages 295-302, July 1994.

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

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

[8] 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.

[9] 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), pages 191-202, Ann Arbor, Michigan, June 1999.

[10] 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.

[11] T. McInerney and D. Terzopoulos. Topologically adaptable snakes. In Proc. Fifth international conference on computer vision (ICCV'95), Cambridge, MA, June 1995. Pages 840-845.

[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.

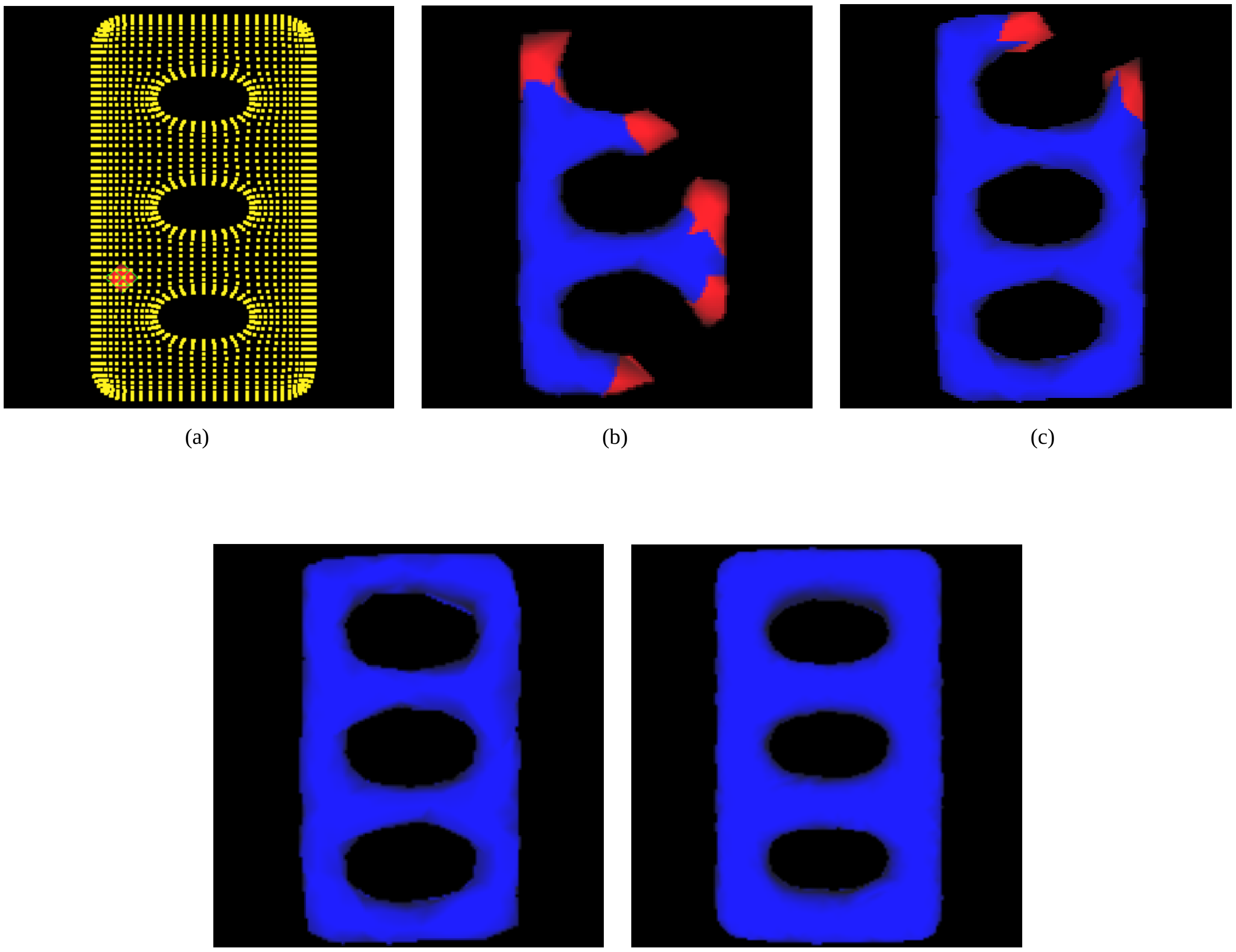
[14] H. Qin, C. Mandal, and B.C. Vemuri. Dynamic Catmull-Clark subdivision surfaces. IEEE Transactions on

Visualization and Computer Graphics, pages 215-229, July 1998.

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

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

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



(a) (b) (c)

(d) (e)

Figure 7: Surface reconstruction from synthesized range data with three holes. (a) Seed model initialized inside the range data. (b) A snapshot of the model during the deformation process. (c) Another snapshot of the model during the deformation process, the topology of the model has been changed. (d) The final shape of the model. (e) Refined shape of the model, one level of subdivision has been applied. Active regions of the model are shown as red, non-active regions of the model are shown as blue, and range data points are shown as gold.

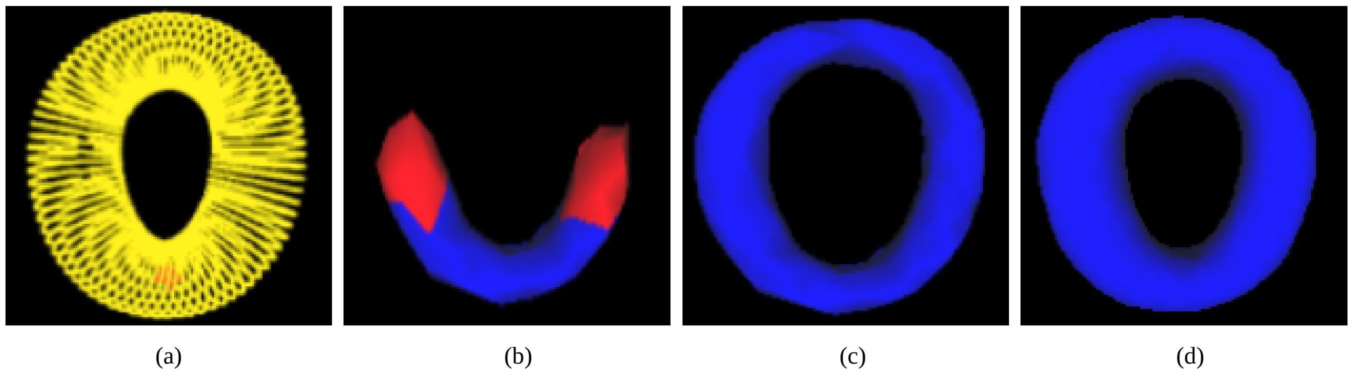


Figure 8: Surface reconstruction of a torus from synthesized range data. (a) Seed model initialized inside the range data. (b) A snapshot of the model during the deformation process. (c) The final shape of the model. (d) Refined shape of the model, one level of subdivision has been applied. Active regions of the model are shown as red, non-active regions of the model are shown as blue, and range data points are shown as gold.

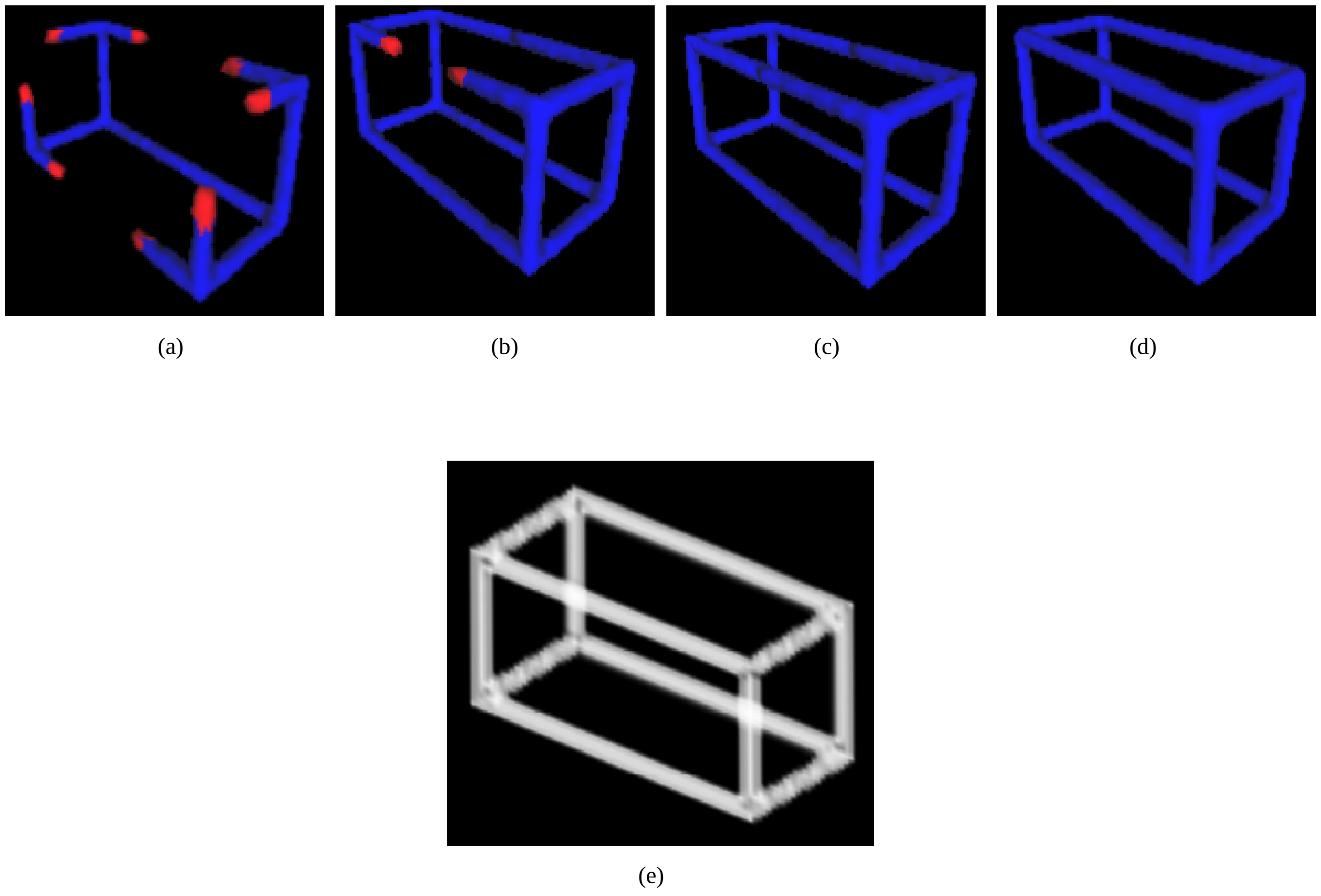


Figure 9: Surface reconstruction from volumetric image data with six holes. (a) A snapshot of the model during the deformation process. (b) Another snapshot of the model during the deformation process, the topology of the model has been changed. (c) The final shape of the model. (d) Refined shape of the model, one level of subdivision has been applied. (e) Volume rendering of the volumetric image data with six holes. Active regions of the model are shown as red, non-active regions of the model are shown as blue.

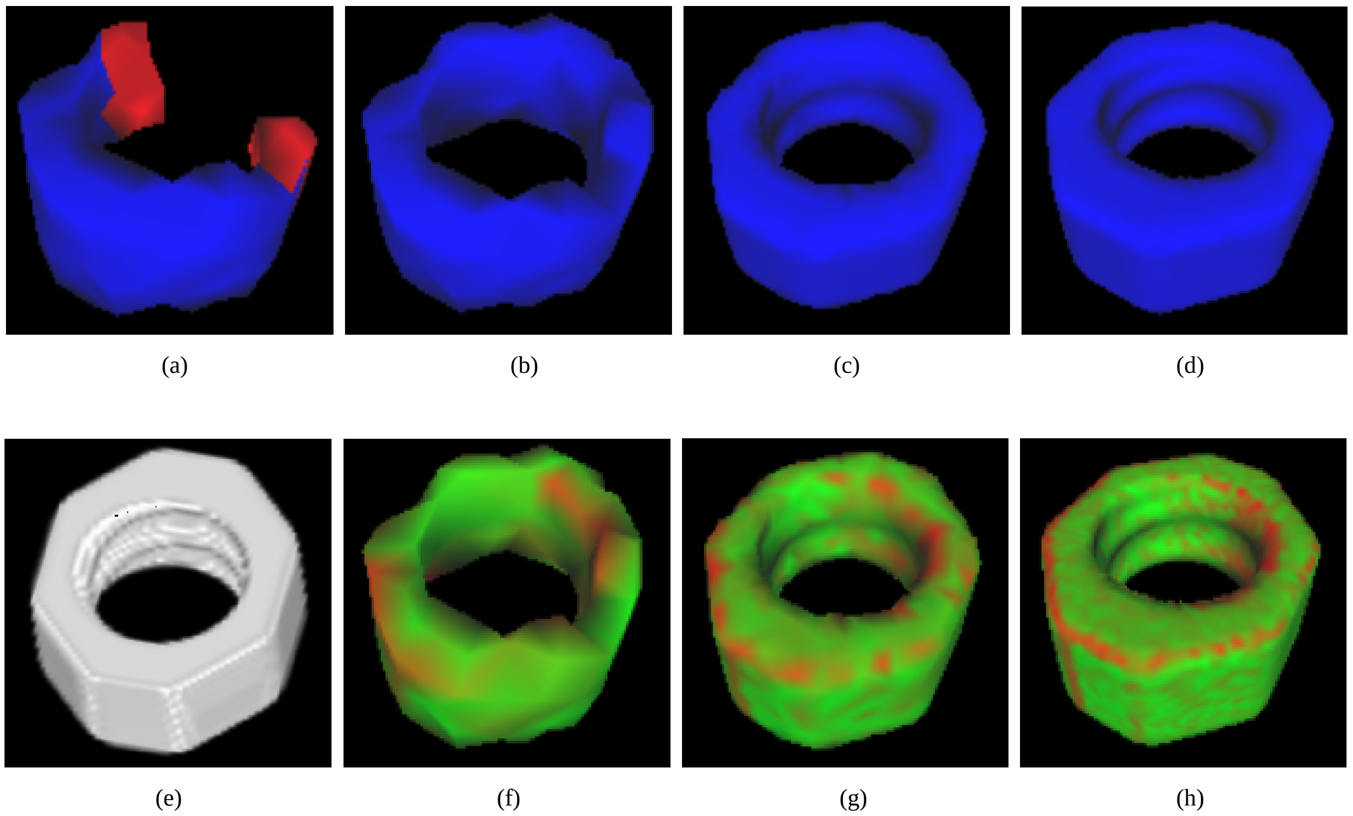


Figure 10: Surface reconstruction of a nut from volumetric image data. (a) A snapshot of the model during the deformation process. (b) The final shape of the model. (c) Refined shape of the model, one level of subdivision has been applied. (d) Refined shape of the model after two levels of subdivision. (e) Volume rendering of the volumetric image data of nut. Active regions of the model are shown as red, non-active regions of the model are shown as blue. (f), (g) and (h) are the error maps of models on (b), (c) and (d), respectively. Green color shows regions whose fitting error are less than 0.5%, red color represents regions whose fitting error is greater than 2%.