

Simultaneous Surface Approximation and Segmentation of Complex Objects

Chia-Wei Liao and Gérard Medioni

Institute for Robotics and Intelligent Systems, University of Southern California, Los Angeles, California 90089-0273
E-mail: cliao@iris.usc.edu

Received January 22, 1997; accepted February 12, 1998

Deformable models represent a useful approach to approximate objects from collected data points. We propose to augment the basic approaches designed to handle mostly compact objects or objects of known topology.

Our approach can fit simultaneously more than one curve or surface to approximate multiple topologically complex objects by using (1) the residual data points, (2) the badly fitting parts of the approximating surface, and (3) appropriate Boolean operations. In 2-D, B-snakes [3] are used to approximate each object (pattern). In 3-D, an analytical surface representation, based on the elements detected, is presented. The global representation of a 3-D object, in terms of elements and their connection, takes the form of B-spline and Bézier surfaces. A Bézier surface is used to connect different elements, and the connecting surface itself conforms to the data points nearby through energy minimization. This way, a G^1 continuity surface is achieved for the underlying 3-D object.

We present experiments on synthetic and real data in 2-D and 3-D. In these experiments, multiple complex patterns and objects with through holes are segmented. The system proceeds automatically without human interaction or any prior knowledge of the topology of the underlying object. © 1999 Academic Press

Key Words: snakes; deformable model; B-spline; segmentation; energy minimization.

1. INTRODUCTION

The inference of an analytical surface from a cloud of points (boundary points of the object) is an important research area, because surface features can be made explicit under this representation. A deformable model, which fits surfaces to data points through minimization, is a good candidate for this purpose.

The idea of fitting data by a deformable model in 2-D, known as a “snake,” can be found in the work of Kass *et al.* [1]. Such models are generalized to 3-D by the same authors [2] for surfaces of revolution. Features, such as edges and curves, can be detected by optimizing models of applied forces (resulting from, for example, curve contrast) and smoothness. Snakes have been used in image analysis applications, which require edge, curve, and boundary segmentation. They are able to conform to object shapes, such as biomedical structures, from noisy observations.

Due to their dynamical nature, snakes are useful to support the interaction between users and computers. A user can interactively edit the shape of an object by adjusting appropriate parameters. Working from a global viewpoint, snakes are quite robust to the presence of noise.

Several variants of snakes exist which are based on Fourier descriptors [38, 39], B-splines [3, 40, 41], finite elements [42], balloons [43], and discrete representation [44, 45], and the literature on 3-D deformable models is also rich [11, 19–37, 48]. Good results have been obtained, but these algorithms handle mostly topologically simple objects and might fail under the following two conditions:

—First, there may be more than one underlying object, and these objects might be close to one another. It takes sophisticated segmentation to separate these mixed objects if the data are corrupted by noise. In this case segmentation is not a trivial job at all.

—Second, they cannot handle objects with deep, narrow, or through cavities, especially when these are winding inside the objects. The reason that most deformable algorithms fail to capture these cavities and holes is that they lack a good external energy definition to estimate the difference between the fitting surface and the underlying object, and a good (or right) initial guess, which is more important, of the object. The fitting process is bound to fail if the topology of the initial surface is wrong. For example, a genus 0 surface cannot approximate a torus accurately. How to infer the right topology for the initial surface might lead to a circular problem.

Taubin *et al.* [15–17] used implicit algebraic curves and surfaces, which assume the form of polynomials, to fit the collected data. They performed segmentation through factorization. The main difficulties are that the Euclidean distance between the data point and the implicit curve (or surface) is not easy to obtain when fitting.

Szeliski *et al.* [18] presented a model based on dynamic particles. By (1) adding intermolecular, coplanarity, conormality, and cocircularity potentials to the internal energy, (2) creating particles appropriately, and (3) doing triangulation, their model can handle arbitrary topology. The results are very encouraging.



DeCarlo and Metaxas [22, 23] proposed blended deformable models, which use a blending function and some primitives to handle objects with through holes. Hierarchical blending is employed to achieve multiple blends; and thus, holes can be handled.

Malladi *et al.* [46] represented shapes by propagating fronts. Unknown shapes are modeled by making the front adhere to the object boundary of interest under a synthesized halting criterion. Topological changes in the front can be handled naturally in this approach. In their experiment, multiple 2-D patterns with holes are segmented successfully.

Whitaker [47] used an implicit model, which is an alternative to parametric models. This implicit model deals with the deformation of higher-dimensional objects. The evolution of this model is straightforward, and two numerical methods, viscosity solution and sparse-field solution, for handling the evolution are proposed.

Here, we propose an approach which segments multiple objects (2-D or 3-D) of arbitrary topologies, and gives each segmented object G^1 analytical representation. This paper is organized as follows: Section 2 presents the goals, issues, and limitations. Section 3 describes the basic idea and shows illustrative examples in 2-D. Section 4 discusses how the same idea is generalized to 3-D, and how a 3-D analytical surface representation is obtained by attaching the elements of the underlying object with connecting surfaces. Then, we show some experimental results in Section 5. Finally, analysis of the system and the conclusion are presented in Sections 6 and 7, respectively.

2. GOAL, ISSUES AND LIMITATIONS

2.1. Goal

Our goal is to design a segmentation and approximation system for both 2-D and 3-D objects which, given dense data points of the underlying objects as the input,

1. segments multiple objects, which are physically disconnected,
2. handles complex objects with through or deep cavities, and
3. generates an analytical surface representation for the underlying objects.

2.2. Issues

Surface representation and initialization are the two fundamental problems associated with the deformable model approach. Since our algorithm is based on the deformable model, these are the main issues we need to address.

2.2.1. Basic Representation: Triangular Mesh vs Rectangular Mesh

The triangular mesh and the rectangular mesh have the same time and space complexities, but the triangular one has a higher constant factor. The rectangular mesh is easier to manipulate, but

it is less versatile than the triangular one in that the rectangular mesh cannot represent an object topologically more complex than genus 1 (e.g., a torus). But because (1) we do not know the topology of the target, and (2) we always, in our system, use multiple genus 0 surfaces to approximate a 3-D object and piece them together later, the triangular mesh does not give any advantage under this condition, while it is more expensive. In our system, the rectangular mesh is adopted for surface representation.

2.2.2. Initialization

In the deformable model, an initial surface is always required, and the initial surfaces conforms to the underlying objects. In the 2-D case, the initial curve is initialized in such a way that it approximates and encloses all of the data points. It is achieved basically by computing the farthest points in many sampled directions centered at the center of gravity. In the 3-D case, initialization is more complex because one has to deal with unknown topology. The algorithm mentioned above for the 2-D case cannot be immediately extended to 3-D and must be adaptively constructed.

2.3. Limitations

There are limitations on data points and representation, which are detailed in the following three paragraphs.

2.3.1. Data Points

We need a dense cloud of 3-D data points, and the underlying objects have to be closed. Just like most deformable algorithms, this system could handle cracks and noise to a certain degree using the internal forces from the smoothness constraints. The fitting process is reliable as long as the global shape could be accounted for by the data points available. Fitting might fail when significant portions of data points are missing.

2.3.2. Pairwise Connections

This system breaks an object into components automatically, and any pair of adjacent components is connected by a connecting Bézier surface. This connection surface serves to account for the data points between two components and also to connect them smoothly. If there is a significant number of data points between two elements, this system would create another element(s) recursively until there is a small number of data points between elements. The connecting surfaces are always placed between two adjacent elements. Due to the assumption of pairwise connection between components, this system might break down when there are more than two components overlapping one another. Figure 1 shows two illustrative examples of the connecting surfaces and decomposition.

2.3.3. Quality of Fit vs Stability of Decomposition

Our decomposition into components is not unique. For example, the number of components depends on the degree of freedom

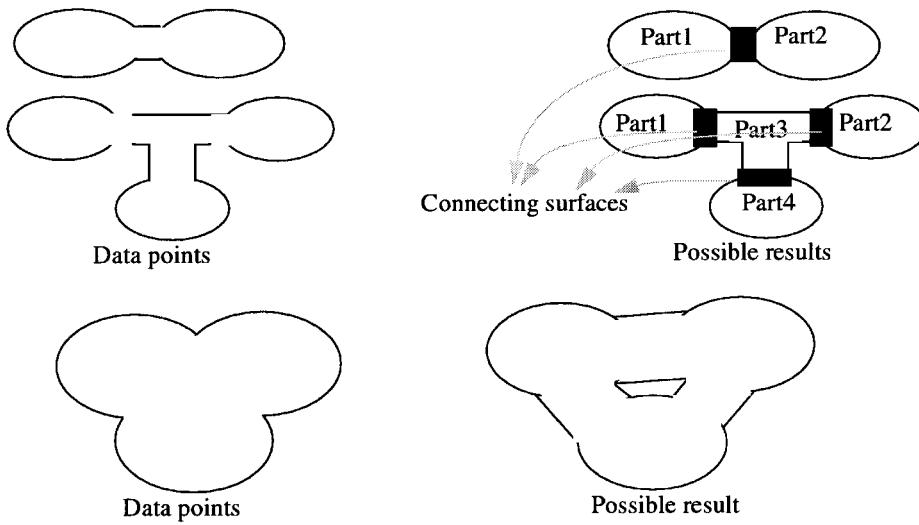
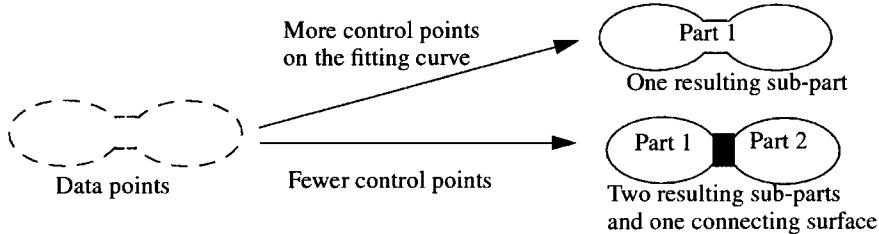


FIG. 1. Two illustrative examples of the connecting surfaces and decomposition. This system assumes pairwise connection between components, and thus might break down when more than two components cluster together.

of the fitting surface (or curve). More control points produce fewer components and vice versa. Of course, if we always use the same number of control points for the fitting surface, we can obtain the same result with the initialization algorithm discussed in the latter section (which takes orientation and distribution into account).

there is a group of residual data points associated with it. If so, we merge it into this group, otherwise we consider this bad span acceptable because no data points are nearby. Now, we have collections of points. We treat each collection of bad data points as an object and extract its contour recursively. Let P be one of the contours.



We briefly present our work in 2-D first, which handles segmentation and complex objects, and then talk about the problems in 3-D and the solutions.

3. OVERVIEW OF THE APPROACH IN 2-D

We wish to obtain an analytical surface representation from the data points collected from unknown 3-D objects, but, for ease of explanation, we start with the 2-D case.

3.1. Basic Idea

Decompose a complex object into a tree, where the leaves are simple primitives, the internal nodes are partial shapes, and the branches represent Boolean operations.

Let B be the outer contour first found. We isolate residual data points that are not well fitted and cluster these residual data points into groups. Next, we extract bad spans of the fitting curve with high external energy. For each span, we check whether

If P is inside B , which means P is a negative element of B , then $B = B \setminus P$;

If P is outside B , which means P is a missing element of B , then $B = B \cup P$.

How do we check whether subelement P is inside or outside body B ? Because the boundary of any object here is a closed continuous B-spline curve, we can differentiate the inside from the outside by setting different gray levels inside these two regions. By examining the gray-level values, we can tell whether a pixel is in B or not and thus determine if P is inside or outside B easily.

An illustrative 2-D example is given Fig. 2. Figure 2a shows a complex object with deep cavities. In Fig. 2b, by (1) finding the outer profile only and (2) using the residual data points and the bad curve segments, we get simple-shaped primitives $B, P_1, P_2, P_3, P_4, P_5$, and P_6 . The complete description of the original object can be displayed by applying the appropriate union and difference Boolean operations. Figure 2c shows the

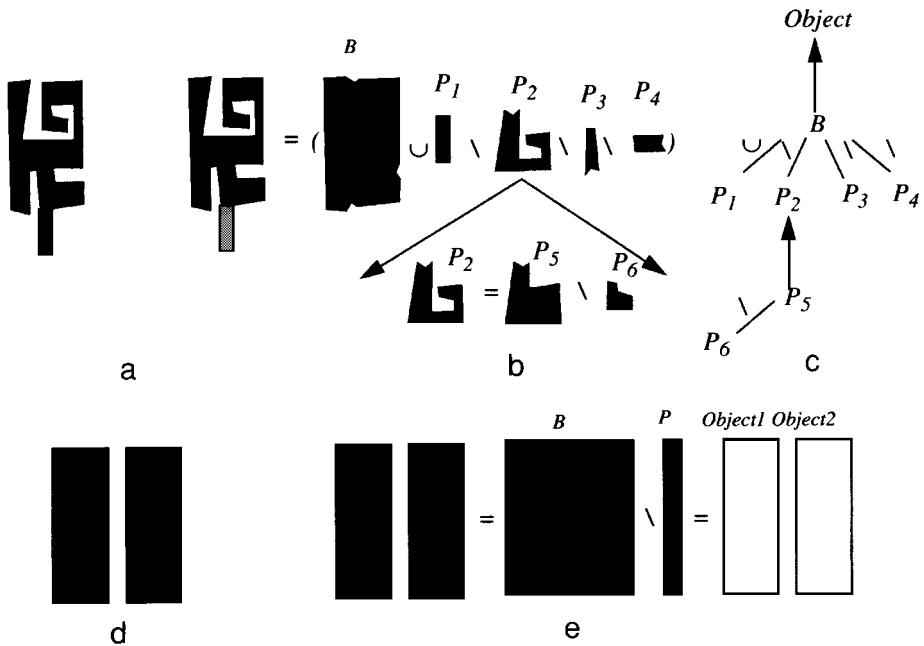


FIG. 2. Two examples of decomposition in 2-D. (a) Original object; (b) result of (a); (c) tree of (b); (d) original data composed of two objects; and finally (e) we can detect two objects after applying the difference Boolean operation.

tree associated with the result in Fig. 2b. Figure 2d shows two objects close to each other. By applying the difference operation, we can classify this set of data points into two different objects as shown in Fig. 2e.

After applying the steps above, we can decompose the input data points into several connected components, each of them corresponding to an underlying object or subobject. Each partial object is the result of Boolean operations applied to its components. All objects here are represented by closed B-spline curves, so the boundaries can be obtained by tracing the edges. There might be some sharp angles along the boundary due to the Boolean operations between two objects. We can inject this boundary as the initial snake into the minimization routine again to get a better result.

See Fig. 3a for a detailed illustration. The number of closed boundaries found is an indicator of the number of underlying objects.

3.2. 2-D Pattern Initialization and Contour Finding

The outer contour of the target at each stage could be obtained using the deformable model. In our case, a *B-snake* [3] is employed. The concavities of the objects can be detected later by applying Boolean operations. We just need to have the initial snake covering all data points. So, the initial fitting curve should be slightly larger than the target, and it shrinks when the energy is being minimized.

First, we compute the center of mass of the data points and extract the farthest data point in sampled directions. The polygon formed by these extremal data points is used as the initial guess. An illustrative example is in Fig. 4.

4. THE 3-D CASE

Figure 3b depicts the 3-D surface fitting process, which uses the same basic idea as in the 2-D case. We first use the B-spline deformable model [19–21] to initialize an extract the surface of each element of the object, and each element is either positive or negative to the entire object. The data point set of each positive or negative element is composed of the residual data points and surface parts resulting from the previous surface fitting, which is the same as the 2-D case. Finally, we put a connection surface between each pair of adjacent elements, and the entire representation should be at least of G^1 continuity.

The difficulty of the 3-D case lies in the unknown topology of the underlying objects. Trying to obtain a correct initial surface might lead to a circular problem. A smooth surface representation of a complex object with holes is difficult when the topology is not known ahead of time. For example, if the object is of *Genus 1*, we might end up with two objects, O_1 (positive) and O_2 (negative). Let O_1 be the main body and O_2 be the hole. To get the complete object, we need a difference Boolean operation such as $O_1 \setminus O_2$. Currently we use a B-spline surface for its ease of smooth surface construction. The problem here is to have O_1 and O_2 smoothly connected without destroying the rectangular topology of the B-spline surface. O_1 and O_2 might have different numbers of control points, which makes the problem even more complicated. One solution to this is to work in the parameter space of the surface, instead of the 3-D space. We use a Bézier blending surface, which achieves G^1 continuity, as a connecting surface between two objects, O_1 and O_2 , which are supposed to be connected. This Bézier blending surface is defined by two

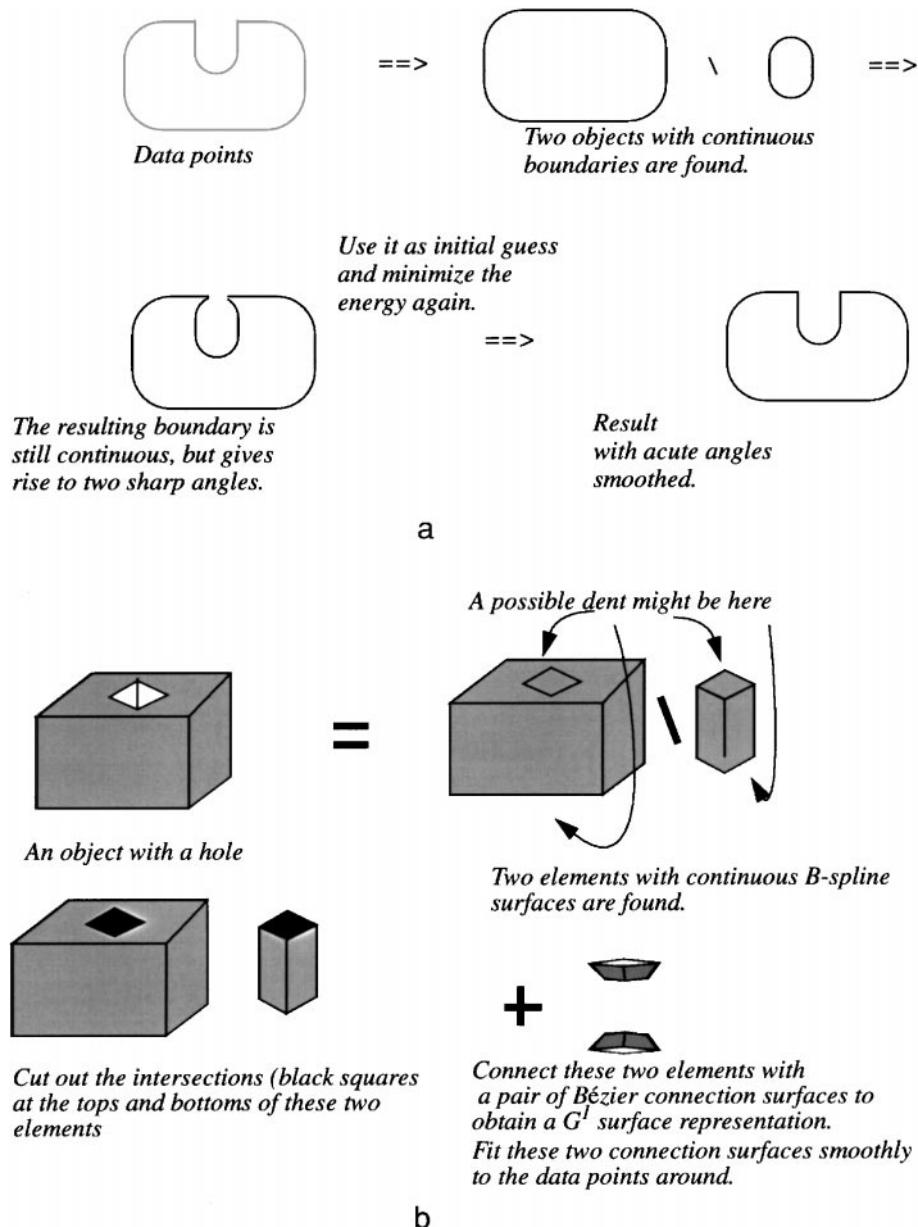


FIG. 3. Further refinement on the 2-D and 3-D fitting result. (a) 2-D snake refinement example. (b) 3-D surface refinement example.

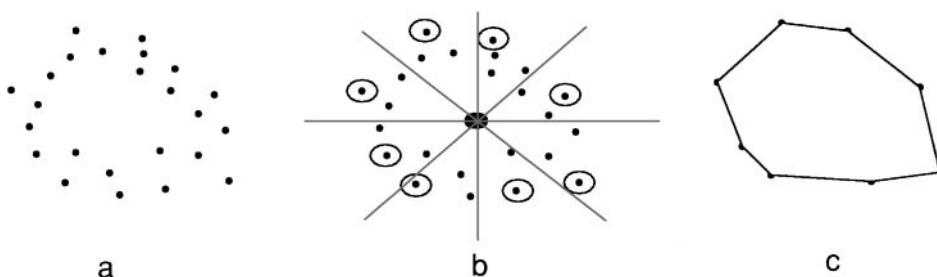


FIG. 4. Initialization. (a) Data points; (b) center of mass and the sampled farthest data points; and (c) initial polygon approximation.

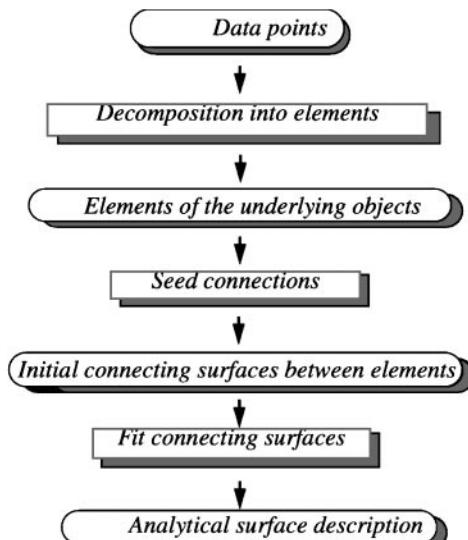


FIG. 5. Flowchart of our approach.

B-spline curves B_1 and B_2 moving on the surfaces of O_1 and O_2 , respectively. Our approach is similar to the work by Filip [24] for CAGD. The main difference is that their system is user controlled, whereas ours is entirely automatic. We automatically detect the need for decomposition into elements and create the blending surface between them.

The flowchart of the approach in the 3-D case is presented in Fig. 5.

In this section, we first talk about surface initialization and fitting for each element of the object.

The fitting process basically serves to decompose objects into elements. Then we show how to bridge two adjacent elements with a Bézier connecting surface.

4.1. 3-D Object Initialization

In the 3-D case, we first calculate the center of mass C . In order to make the system invariant to translation and scaling, we

compute the three eigenvectors of the covariance matrix of the data points and then use these orthogonal vectors to define a new coordinate system with the origin at C . We define the sampled directions according to this new coordinate system and find the farthest data point in sampled directions. With these extreme points, we can obtain a reasonable initial estimate on the data points. An example is shown in Fig. 6.

In both the 2-D and 3-D cases, if there is no data point in the sampled direction, we set the corresponding radius to a predefined constant for the initial estimate of the curve and surface.

4.2. Decomposition into Elements

The 3-D decomposition is conceptually the same as in 2-D. We decompose the underlying objects into elements by recursively fitting B-spline surfaces, and each B-spline surface accounts for an element. An element is either positive or negative, which is decided using the algorithm in Section 3.1. After each fitting, the data points are:

1. the residual data points resulting from the previous fitting, and
2. the parts of the previous fitting surface with no data points nearby.

The 3-D case is an extension of the 2-D one. But, in practice, the 3-D case is much more difficult because the topology is unknown ahead of time and the adjacent elements need to be connected smoothly. These problems are addressed in the following paragraphs.

4.3. Element Fitting

The formalism we are about to establish amounts to deforming an initial surface to conform as closely as possible to the input 3-D data points. This is achieved by defining an attraction force field around the data points to bring the initial surface closer to them. The initial surface is updated by a function minimization algorithm, Powell [5–10].

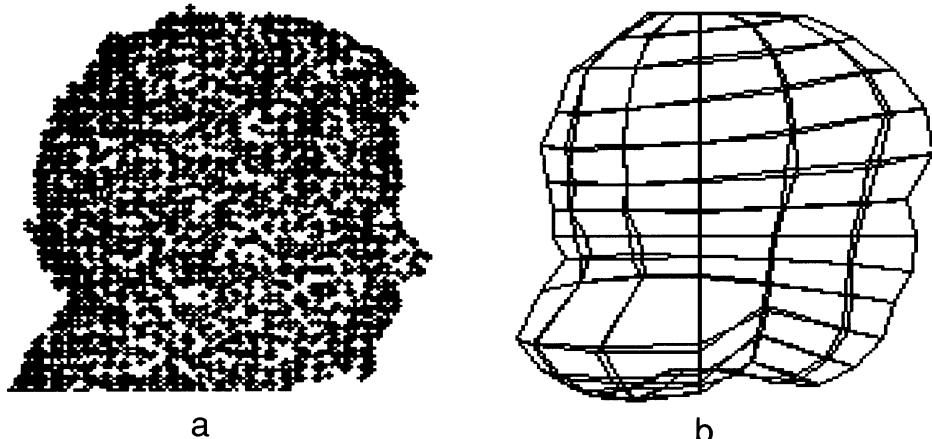


FIG. 6. Example of initial surface for a 3-D object. (a) Data points and (b) initial surface.

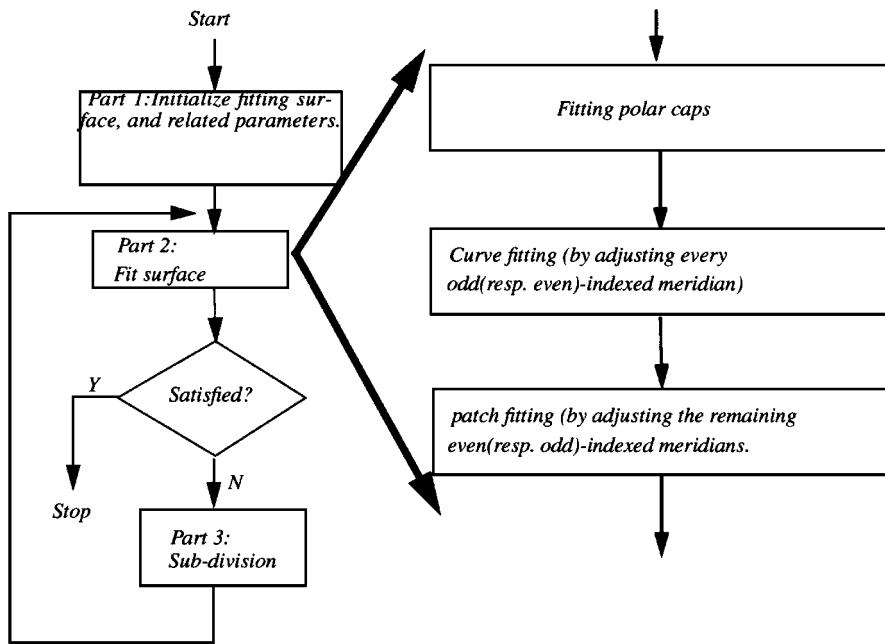


FIG. 7. The flowchart of our 3-D algorithm.

The total energy of the fitting surface is defined as

$$E_{\text{total}} = W_{\text{int}} * E_{\text{int}} + E_{\text{ext}}.$$

E_{ext} expresses the distance between the fitting surface and the data points. E_{int} depends on the constraints, such as smoothness. The definition of E_{int} is subject to change when different constraints are applied. W_{int} is the weight associated with E_{int} .

There are three main parts in this algorithm. The first is to *initialize* the fitting surface and the related parameters, the second to *fit* the surface to the data points, and the third to *subdivide* the fitting surface if necessary (see Fig. 7).

4.3.1. Choice of Representation

We choose rectangular patches over triangular ones as they allow the creation of a smoother (C^1 or C^2) surface. This forces us to deal explicitly with poles, as explained later.

Here, we currently use a linear B-spline surface for its efficiency in computation time and some geometric properties we need. In the linear B-spline surface, each control point only affects its four neighboring patches. This makes it very easy to separate the whole surface into independent strips which will be processed separately later.

The initial surface we use is topologically equivalent to a sphere. Here we consider a sphere as composed of three parts: two caps and an open cylinder as shown in Fig. 8. These three parts are processed separately in our algorithm.

Now, we define some terms for further use. A *cap* is the triangular patch formed by a *pole* and its adjacent control points. Thus, we always have two caps. A *meridian* (a line of constant

u in parameter space) is the curve connecting the two poles, as depicted in Fig. 8.

4.3.2. Surface Fitting

The basic idea here is to keep the time and space complexities under control by breaking down the 3-D problem into several 2-D subproblems.

Instead of injecting all $M \times N$ control points into the minimization procedure (which is possible but extremely expensive), we decompose the problem into a cap fitting problem followed by (simpler) curve fitting and patch fitting problems.

We select every other meridian and move their control points, which are not shared with the two caps, to minimize their energy. We then select the remaining meridians and move their control points, not shared with the two caps, to minimize the energy of the related patches this time. It is important to note, however, that only the bad segments (patches or curves) are injected into the minimization procedure.

Cap fitting. The caps of the surface are initialized planar to make the constructed B-spline surface smooth at the poles. First,

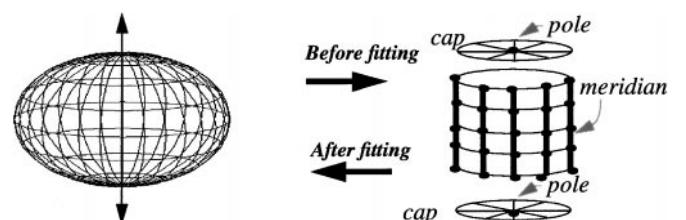


FIG. 8. The initial closed surface and the definitions of pole, meridian, and cap.

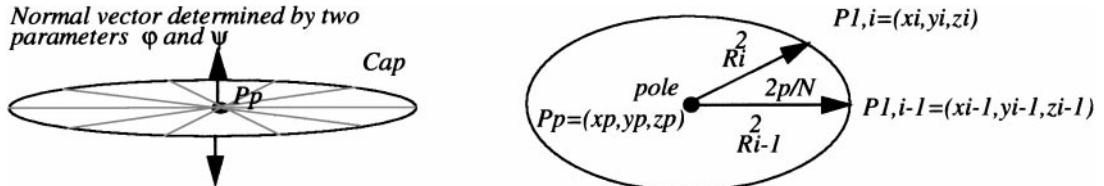


FIG. 9. $P_{1,i}$ and $P_{1,i-1}$ are two consecutive control points on the row immediately adjacent to the pole. Number of variables: $N + 5(NR'_i s + \varphi + \psi + x_p + y_p + z_p = N + 5)$.

we fit the caps to the data. By treating the pole as multiple control points and forcing the caps to be planar (the tangent vectors in all directions at the poles would be coplanar) during fitting, G^1 continuity at the poles can be achieved when we construct a smoother surface. When fitting, the cap changes its shape subject to the planarity constraint. The variables for fitting the caps are depicted in Fig. 9.

Curve fitting. Second, we perform the curve fitting to every other meridian to locate the profile of the target, and this is done by applying energy minimization to these meridians. We select the odd- (resp. even-) indexed meridians and fit them to the data by treating them as linear B-snakes [3]. We first extract the bad spans of the snakes, group the connected ones, and then tune one group at a time. We leave good spans untouched during fitting. An illustrative example is in Fig. 10. When calculating the internal energy of these meridians, we not only consider their own smoothness, but also the smoothness between them and their immediate neighboring even- (resp. odd-) indexed meridians. Then we let these selected meridians adapt to find the profiles of the target. These selected meridians are not influenced much by the remaining part of the fitting surface when moving, so they can detect the object more accurately. Note that the external energy of these selected meridians is defined on the curve without considering the surface nearby. The surface is separated by the selected meridians into independent “strips” in a way, and each strip contains an even- (resp. odd-) indexed meridian.

Mesh fitting. Next, we fit each strip to the target. We select meridians of the other alternate, even- (resp. odd-) indexed, to do patch fitting for each strip. We treat them as regular snakes, except that they are tuned to minimize the external energy (error) of the strip. This means the external energy is not only from the

curve but also from the area it defines. As in curve (snake) fitting, we group connected bad patches and tune the associated set of the control points of each group on the selected meridian one at a time. An example is shown in Fig. 11.

4.4. Connecting the Elements

Here we show how to create a connecting surface between two adjacent elements. First we talk about the surface representation, which is a Bézier surface. Then we talk about the initialization of this connection surface and finally how to conform it to the residual data points between adjacent elements.

4.4.1. Bézier Connecting Surface

A connecting surface serves as the joint between two connected simpler elements. It is defined by a 3-D Bézier blending surface as depicted in Fig. 12. First, we initialize two closed B-splines curves (B-snakes [3]) moving on the B-spline surfaces of the primitives. To be more specific, these two B-snakes are, in fact, tuned on the (u, v) parameter spaces of the B-spline surfaces to fit the connecting surface to the data points in 3-D space.

Let C_1 and C_2 be the boundary curves of the connecting surface on the two objects. We sample N points from C_1 and C_2 , and N should be large enough for both C_1 and C_2 . There are several steps for the Bézier blending surface construction:

1. We need to determine the correspondence between the sampled points on C_1 and C_2 .
2. We need to compute the cross-border tangents CT_1 and CT_2 for each sampled point on C_1 and C_2 .
3. With the correspondence relationship, and the cross-border tangent of the points on both curves, we construct the Bézier blending surface.

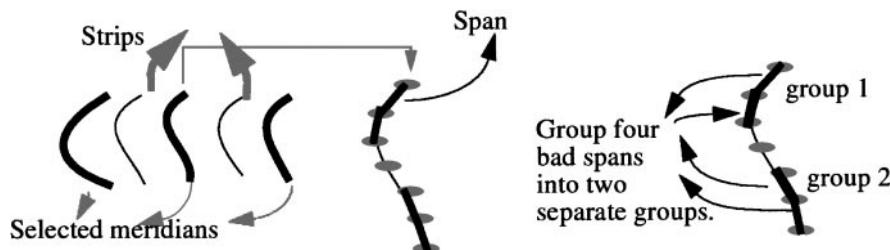


FIG. 10. Illustration of the curve fitting procedure. For each selected meridian, classify the connected bad spans into groups (solid spans in the above). Tune the associated control points of each group at a time during fitting. In the above illustrative example, there are two separate groups, each of which contains three control points.

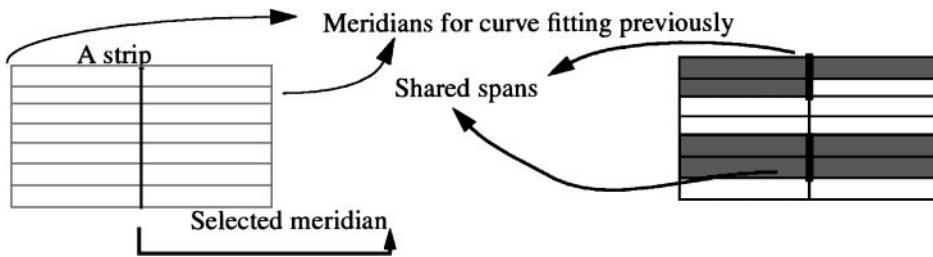


FIG. 11. Illustration of the mesh fitting procedure. Supposing the gray areas are bad patches, we have two groups connected bad patches, each of them containing three control points of the selected meridian. These two sets of control points are tuned separately to fit the strips to the data.

4. Because the normal to each surface point might point in opposite directions, we have to determine which is outward (or inward) when rendering the surface.

After the correspondence between C_1 and C_2 is determined, the two curves can be parameterized by the same parameter, say w . Similarly CT_1 and CT_2 can be parameterized by w , too. The Bézier blending connecting surface can be represented as follows:

$$\begin{aligned} G\langle t, w \rangle = & \langle 1-t \rangle^3 \cdot C_1(w) + 3 \cdot \langle 1-t \rangle^2 \cdot t \cdot \langle C_1(w) \\ & + CT_1(w)/3 \rangle + 3 \cdot (1-t) \cdot t^2 \cdot \langle C_2(w) \\ & + CT_2(w)/3 \rangle + t^3 \cdot C_2(w). \end{aligned}$$

We now give details of these steps.

4.4.2. Point Correspondence

Let O_1 and O_2 be the objects to be connected. Suppose O_2 is a element of O_1 . O_2 could be a positive element (which is outside O_1) or negative element (which is inside O_1 and might be a cavity or a through hole) of O_1 .

Both boundary curves go counterclockwise as a convention. With this convention, we can determine which part of the object surface is inside the connecting surface, and the direction of the cross-border tangent can be correctly set. How to initialize the counterclockwise boundary curve B is discussed in 4.4.6.

If O_2 is a positive element of O_1 , then the boundary curves of the connecting surface on both elements are going in opposite

directions. If O_2 is a negative element, the boundary curves go in the same direction. An example for the positive-element case is shown in Fig. 12. We set the variable D to -1 if O_2 is positive and to 1 otherwise. Let K be the offset of curves C_1 and C_2 , and $P_{C1,i}$ and $P_{C2,i}$, $0 \leq i \leq N$ (N points have been sampled), are points on C_1 and C_2 , respectively. An error function $E(K)$ is defined as follows:

$$E(K) = \sum_{i=0}^N \|P_{C1,i} P_{C2,((i+D \cdot K) \bmod N)}\|^2.$$

Then find K' that minimizes $E(K)$. With K' and D , the correspondence of the points of C_1 and C_2 can be determined. The point $P_{C1,i}$ on C_1 corresponds to point $P_{C2,((i+D \cdot K') \bmod N)}$ on C_2 .

4.4.3. Cross-Border Tangent

In order to construct a G^1 connecting surface, we need the cross-border tangent at each point along the boundary curve C . Figure 13 shows how the cross-border tangent is computed. Let $O(u, v)$ be the surface of the target and point P be a point on boundary curve C (also on $O(u, v)$). We first calculate the normal N_p of O at P . N_p is always pointing outward and can be obtained by $(\partial O(u, v)/\partial u) \times (\partial O(u, v)/\partial v)$. The boundary curve C always goes counterclockwise in our system. Since the boundary curve C has an analytic foundation, the tangent T_p of C at point P can be computed easily. The cross-border tangent CT_p at point P can be calculated by $T_p \times N_p$.

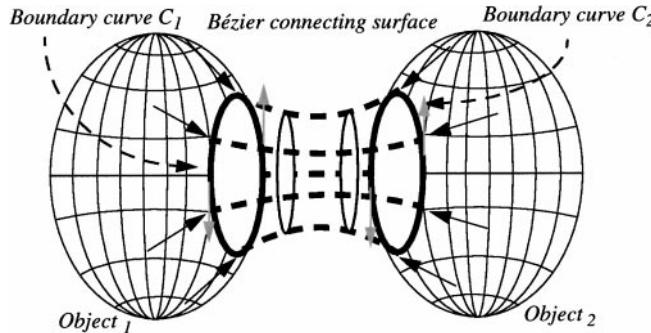


FIG. 12. The connecting surface between two objects.

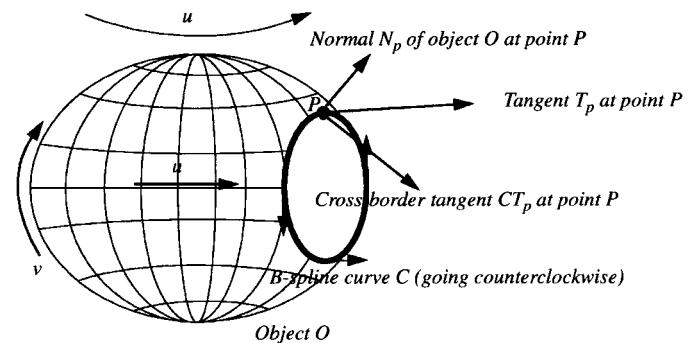
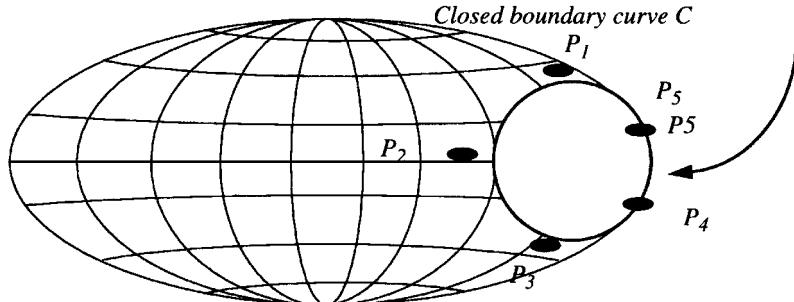
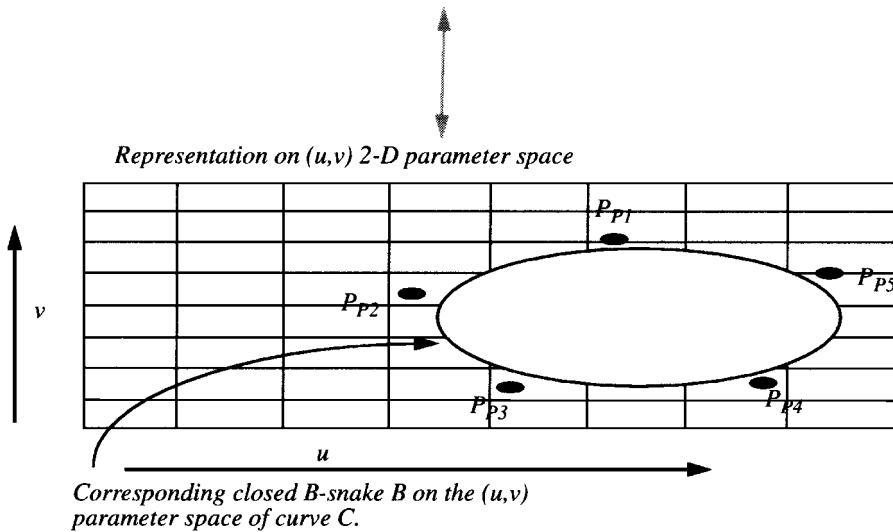


FIG. 13. Illustration of the cross-border tangent.

Representation in 3-D space



The boundary curve C has five control points P_1, P_2, P_3, P_4 , and P_5 , which might not be on the boundary curve C , on the element's surface.



The boundary curve C and its five control points in the 3-D surface correspond to a 2-D B-snake B and five control points ($P_{p1}, P_{p2}, P_{p3}, P_{p4}$, and P_{p5}) on the 2-D (u,v) parameter space.

FIG. 14. The connecting surface boundary on the object is a closed curve C . We tune the control points on the (u, v) parameter space to adapt the connecting surface to the data points nearby.

4.4.4. Connecting Surface Construction

Let $P_{1,i}$ be a point on C_1 , $CT_{P(1,i)}$ the cross-border tangent at $P_{1,i}$, $P_{2,j}$ be the corresponding point of $P_{1,i}$ on C_2 , and $CT_{P(2,j)}$ the cross-border tangent at $P_{2,j}$. With this information in hand, we can construct a Bézier curve connecting $P_{1,i}$ and $P_{2,j}$ and preserve G^1 continuity at the junctions with objects O_1 and O_2 . Constructing a Bézier curve for each pair of corresponding points on C_1 and C_2 , we obtain a Bézier blending surface. The functional representation of this surface was shown earlier in Section 4.4.1. We can scale $CT_{P(1,i)}$ and $CT_{P(2,j)}$ up and down without losing G^1 continuity, which gives the connecting surface more flexibility and degrees of freedom when fitting.

Suppose there are M_1 and M_2 control points for the boundary curves C_1 and C_2 . For each control point P_c , we have three associated variables. Two of them encode the coordinates on the (u, v) parameter space, and one defines the norm of the

cross-border tangent vectors of the part of the boundary curve constructed by P_c . These $3(M_1 + M_2)$ variables are tuned to fit the connecting surface to the data points around.

4.4.5. Determining the Outward Direction of the Surface Normal

We always initialize the fitting surface of an object in such a way that $(\partial O(u, v)/\partial u) \times (\partial O(u, v)/\partial v)$ points outward. But we need to invert the direction when it comes to a negative element of the object. Since our algorithm for the high-genus object works recursively, a negative subelement of a negative element is positive, a negative subelement of a positive element is negative, a positive subelement of a negative element is negative, and a positive subelement of a positive element is positive. After the polarities of all elements are determined, we can determine the outward direction of the connecting surface straightforwardly.

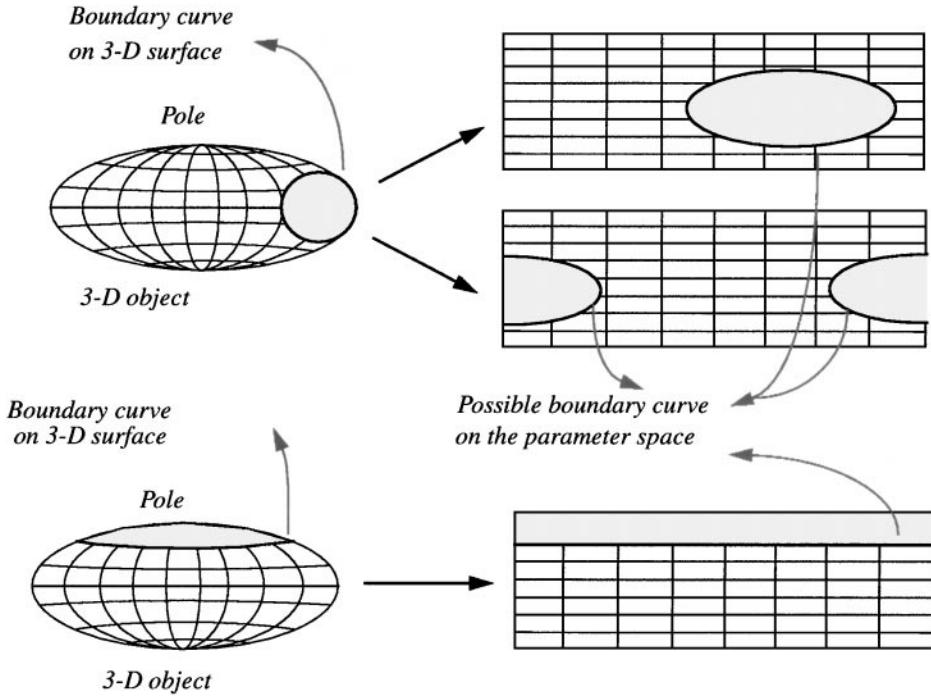


FIG. 15. Boundary curve in 3-D and parameter spaces.

4.4.6. Connecting Surface Initialization

The boundary curve on the element's (part of the 3-D object needs to be connected) surface corresponds to a B-spline curve on the (u, v) parameter space, so we can manipulate the boundary curve by tuning the control points on the (u, v) parameter space. We have to initialize two B-spline curves on the (u, v) parameter spaces of the two connected elements first and then construct a Bézier blending surface from the corresponding 3-D boundary curves on the object surfaces.

The initialization of the connecting surface is a nontrivial task. The corresponding curves might be completely different on the (u, v) parameter space even for simple closed curves, such as circles, when they are at different places of the object surface. An illustrative example is shown in Fig. 15.

We initialize the connecting surface as follows:

1. Extract all data points between objects O_1 and O_2 and then the surface points close to the other object.
2. Cluster these points into groups.
3. Discard groups without residual data points. Now, each group corresponds to a connecting surface between O_1 and O_2 .
4. For each group D , extract the surface points on O_1 and O_2 close to D and find their corresponding points on the (u, v) parameter space. Map these points from the (u, v) parameter space onto the Gaussian sphere G , which is a unit sphere in 3-D space. Compute the center of mass C of the points on the Gaussian sphere. Let O be the origin. Rotate the Gaussian sphere G , point \overline{OC} toward the North pole, and let the G' be the resultant Gaussian sphere. On G' , sample the farthest point along each meridian in ascending order of longitude and find its cor-

responding coordinate on the (u, v) parameter space. Use these points on the (u, v) parameter space as B-spline control points to initialize the boundary curve. Since the longitude increases counterclockwise, we can guarantee that the boundary curve always goes counterclockwise. This is illustrated in Fig. 16. When initializing the surface, we use the residual data points and the surface points. But when fitting the connecting surface, we use the residual data points only to compute the external energy.

4.4.7. Fitting Connecting Surface to Data Points Nearby

Now, we have the boundary curves C_1 and C_2 for each connecting surface. With C_1 and C_2 defined, the coordinates and the cross-border tangents of C_1 and C_2 can be computed directly.

There are two kinds of energy needed during surface fitting: external energy (which is a measure of the difference between the fitting surface and the collected data points) and internal energy (which represents the smoothness of the fitting surface). The fitting surface is brought closer to the data by minimizing these two energies.

We simply inject the associated $3(M_1 + M_2)$ variables (from Section 4.4.4) of the connecting surface into a minimization routine, and the connecting surface conforms to the data points nearby. In our system, we use Powell [5–10] for minimization.

5. EXPERIMENTAL RESULTS

Several experimental results are presented here. We use B-snake [3] mesh fitting [20] as the engines for 2-D curve and 3-D surface fittings in our experiments, respectively.

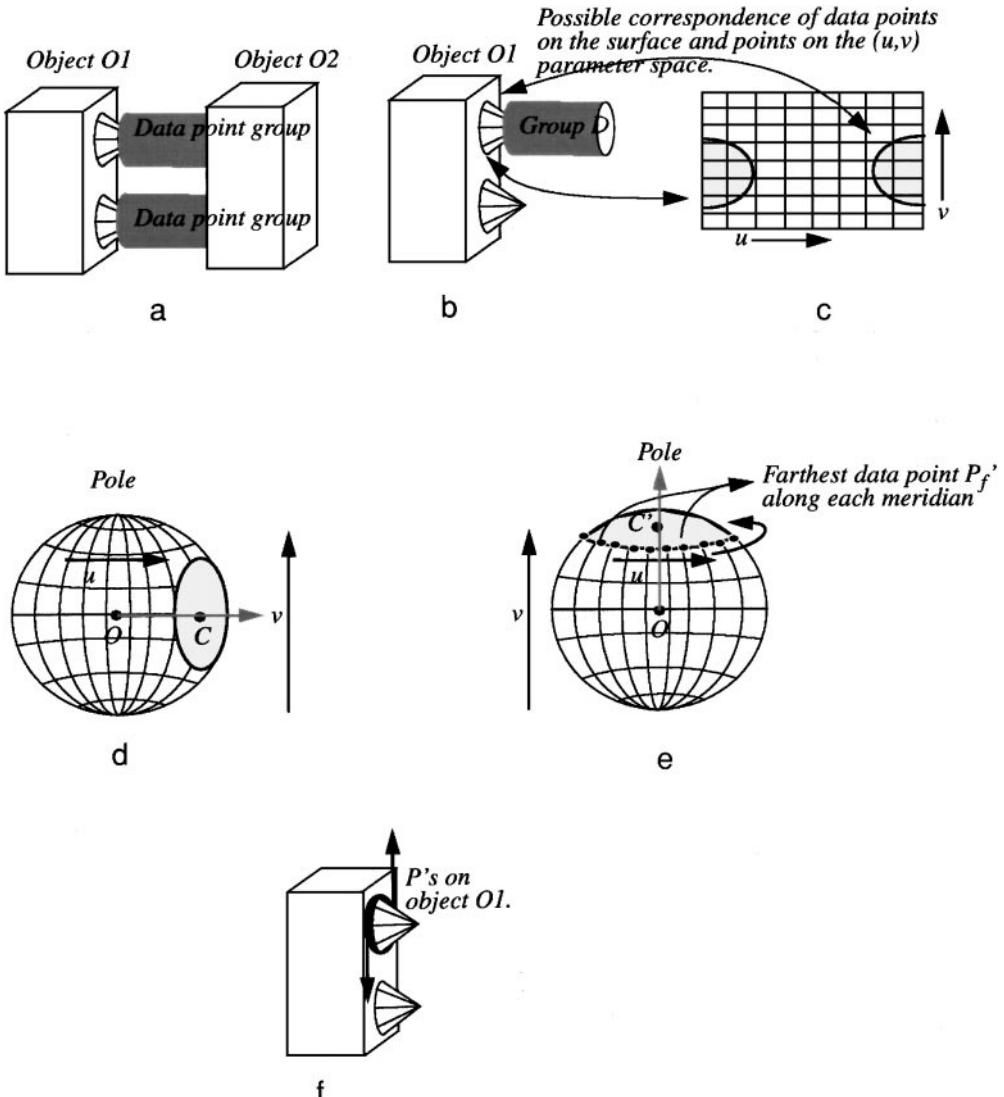


FIG. 16. An illustrative example showing how to initialize the boundary curves of the connecting surface. (a) Two objects, which are well fit already, and two groups of residual data points for the connecting surfaces. (b) Choose one group D of data points and one object O_1 as an example. (c) Parameter space. Points on (u, v) parameter space corresponding to surface points of O_1 that are close to data points in D or object O_2 . (d) Gaussian sphere G . Map the (u, v) parameter space onto a Gaussian sphere G , and find the center of mass C of the surface points. (e) Rotated Gaussian sphere G' . Rotate G to make OC pointing at the North pole. Let G' be the resultant Gaussian sphere. Sample farthest data points P_f 's along each meridian in ascending order of the longitude as shown in (e), and P_f 's go in the counterclockwise direction. (f) Possible corresponding surface points P 's of P_f 's on object O_1 , which also go in the counterclockwise direction. With P 's on O_1 and the ordering information, we can initialize the boundary curve between the connecting surface and object O_1 .

In the 3-D case, for example, the number of the 3-D control points tuned for surface fitting is much larger than that of 2-D control points tuned on the parameter space. The connecting surface is determined completely by its two boundary curves on the surfaces of the elements, so connecting surface fitting is more or less a 2-D curve fitting problem, which is much simpler than the 3-D surface fitting for segmentation. Thus, the computation time of the experiments below is mainly spent on curve (for 2-D patterns) or surface fitting (for 3-D objects) which segments data points into elements. Connecting 3-D element pairs and 2-D patterns only takes a fraction of the overall computation time.

The experiments on 2-D patterns take less than 10 min to run. In the 3-D case it takes about 20 min to segment a 3-D element, and 3–4 min to fit a connection surface to data points. The platform is a Sparc-10 workstation.

B-spline surface is employed here for 3-D object segmentation. Unless otherwise mentioned, each surface has 12 by 14 control points, and each of the two boundary curves of the connecting surface has eight 2-D control points on the corresponding parameter space. As to 2-D pattern segmentation, 30 control points are employed for the B-snake fitting. All of the data points are first projected into a $200 \times 200 \times 200$ 3-D cube,

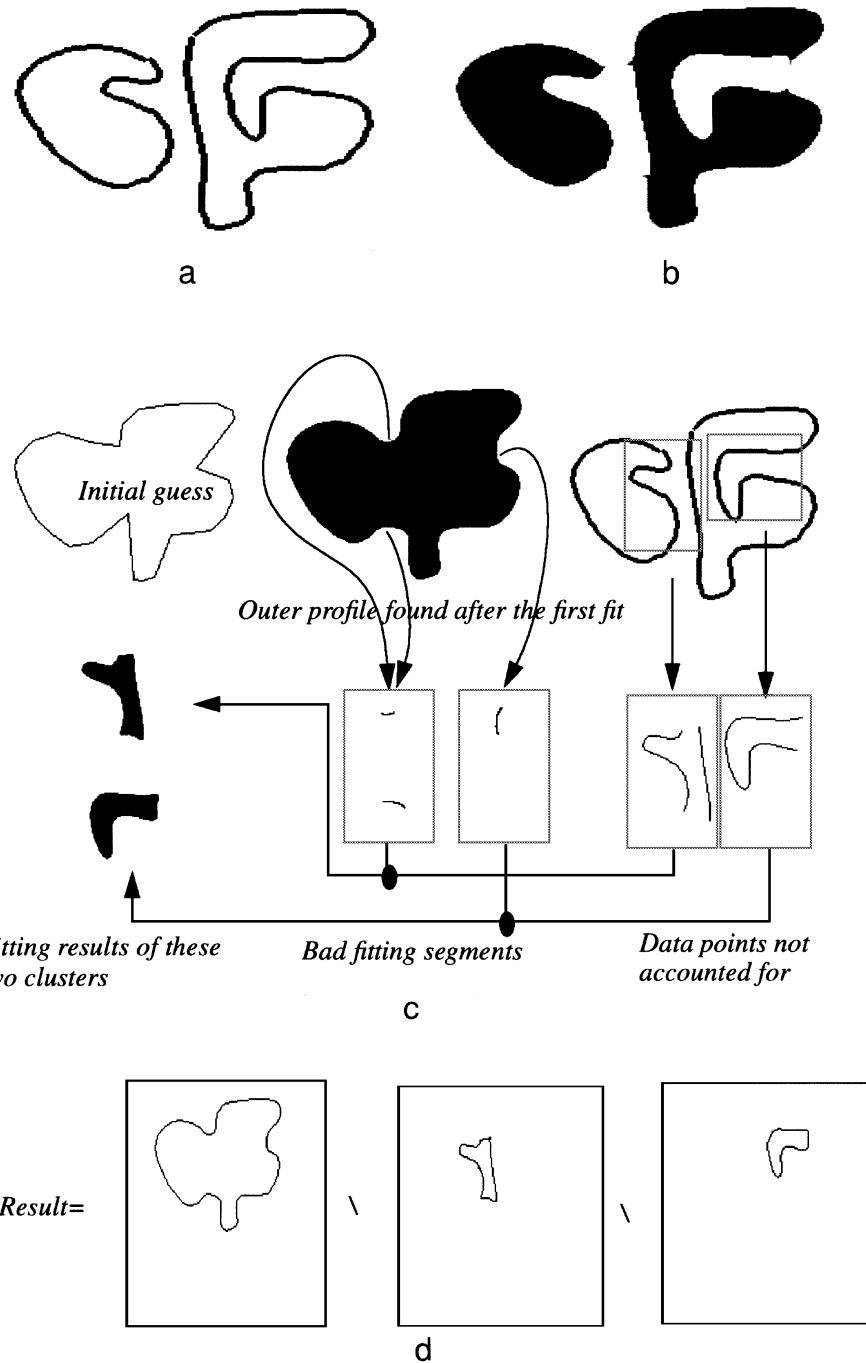


FIG. 17. Segmentation on two concave patterns. (a) Original hand-drawn data. (b) Fitting result after applying a sequence of Boolean operations. (c) Two clusters of the bad segments and bad data points found after the first fit. (d) Order of the Boolean operations applied.

and the fitting process stops when the average error is less than 1 voxel.

In the first four experiments, we did not polish the final results with one more layer of post-processing, which fits a closed B-snake to each contour found. This is why the contour is pointy at the junction between two elements. The contours here are C^0 only.

In the first experiment (in Fig. 17), we use hand-drawn data, which consists of two simple objects with deep concavities. After applying a B-snake and appropriate Boolean operations, these two objects are segmented. Figure 17a is the original data, and Fig. 17b is the final result. Figure 17c shows the initial guess, and how the residual data points and the bad B-snake segments merge into two clusters (which form the negative elements of

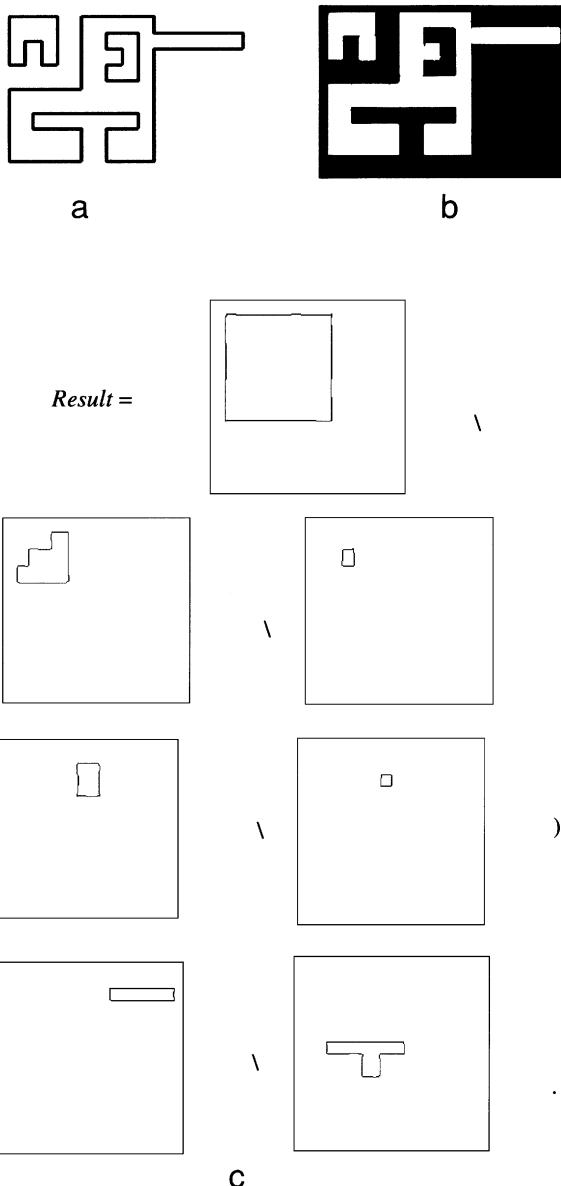


FIG. 18. Segmentation on two complicated patterns. (a) Original data. (b) Fitting result after applying a sequence of Boolean operations. (c) Order of the Boolean operations applied.

the target). Figure 17d shows the Boolean operations applied. At first, the outer contour is found. We find two clusters of residual data points inside the outer contour, so the difference Boolean operations are applied. Finally, the original contour is separated into two.

In the second experiment, we use synthetic data (as shown in Fig. 18a). There are two separate polygons in this synthetic data. One is concave, and the other is a complex object with a concave-like hole. The result is in Fig. 18b. The fitting result is obtained after applying a sequence of Boolean operations. The edges here are represented by a set of B-snakes, so the edges here are perfectly continuous. By tracing along the edges, we

find two different objects and the inner hole in the bigger object, and thus the objects are segmented. Figure 18c shows how the appropriate Boolean operations are applied. Please notice that some residual data points are complicated, so our algorithm needs to be applied recursively in order to capture correctly the shapes of the residuals (each group of residual data points is considered as another object to fit).

In the third experiment (in Fig. 19), the data points are the same as those in Figs. 17 and 18 except that (1) half of the data points are deleted randomly, and (2) the data points are randomly shifted at most three pixels using a uniform-distribution random-number generator. Figure 19a shows the data points, which have broken boundaries. Figure 19b is the result. The sequences of the Boolean operations applied are exactly the same as those in the previous experiments. The boundaries detected here are more irregular, but they are still continuous B-spline curves. So the objects and the hole can still be correctly segmented.

The fourth experiment, in Fig. 20, is on synthetic 3-D data which is composed of two separate genus 1 tori. Figure 20a shows the data points, Fig. 20b is the result (object A) after the first fit, which results in a dumbbell-like shape, and Fig. 20c is the residual of the data points that are not accounted for by object A. They are from the inner parts of the two tori, and Fig. 20d shows the bad parts of object A without data points nearby. They are from the two ends and middle of object A. Figure 20e is the cluster of points in Figs. 20c and 20d. Figure 20f is the fitting result (object B) to points in Fig. 20e. Because object B is inside object A, a difference operation $A \setminus B$ is applied, which leads to two separate entities. Figure 20g shows the shells of the two separate entities, which are tori. In this experiment, objects (two separate tori) more complex than genus 0 are well handled, and the data segmentation, which segments the data points into two elements, is automatically done after fitting.

The fifth experiment is on a synthetic high genus object, which is a sphere with a cross-like through-tunnel inside. In this cross-like tunnel, there is another bar. Figure 21 shows the data points and the result. Figure 21a shows the data point set, which contains a sphere (positive element as shown in Fig. 21b), a cross (negative element as shown in Fig. 21c), a bar (a negative element of Fig. 21c, and a positive element of the underlying object as shown in Fig. 21d). Figures 21b, 21c, and 21d show three surfaces S_1 , S_2 , and S_3 , and are the results after applying the algorithm in [21]. Figures 21e and 21f show two views of the residual data points around the connecting surfaces between surfaces Figs. 21b and 21c and between surfaces Figs. 21c and 21d. Figures 21g and 21h show two views of the objects with initial connecting surfaces. The connecting surfaces do not look smooth because of bad initialization. This problem is automatically fixed by the fitting process later on. Figure 21i shows the negative element of the object, which results from the difference Boolean operation $S_2 \setminus S_3$. A Boolean difference operation is applied because S_3 is a negative element of S_2 . There are two connecting surfaces (connections) between S_2 and S_3 . Since Fig. 21i is a negative element, a difference Boolean operation is

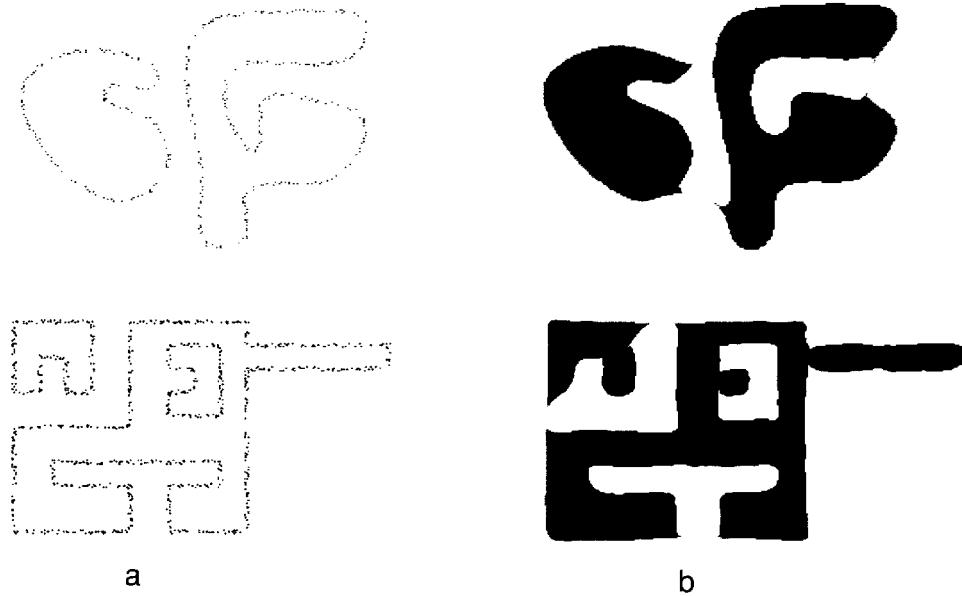


FIG. 19. Experiments on noisy data. (a) Data corrupted by noise. (b) Fitting result.

applied again on S_1 and $(S_2 \setminus S_3)$. There are four connecting surfaces between S_1 and $(S_2 \setminus S_3)$. The results are shown in Figs. 21j and 21k. Figure 22 shows the activities on the parameter spaces. Figures 22a and 22c show the coordinates on the parameter space of the points of surfaces S_1 and S_2 that are close to the other object and the residual data points. There are four connections between surfaces S_1 and S_2 , and the (u, v) coordinates of initial boundary curves are shown in Figs. 22b and 22d, respectively. In Fig. 22d, regions 2, 4, 5, and 6 are for the connections with surface S_1 . There are two connections between surfaces S_2 and S_3 . Figures 22c and 22e show the coordinates on the parameter spaces of the points of surfaces S_2 and S_3 that are close to the other object and the residual data points. The (u, v) coordinates of initial boundary curves are shown in Figs. 22d and 22f, respectively. In Fig. 22d, regions 1 and 3 correspond to the connection with surface S_3 . In this example, we can see that some connections surround the poles of the (u, v) parameter space, and the initial boundary curves are open curves on the (u, v) parameter space.

The sixth experiment, shown in Fig. 23, is on *Genus 1* synthetic data, which is a teapot. Figure 23a shows the data points. Figure 23b shows the result after fitting recursively to the residual data points and bad parts of the fitting surface. There are four elements segmented: knob, handle, spout, and main body, which are disconnected. Figure 23c shows the four initial connecting surfaces between the corresponding pairs of elements. Figure 23d shows the final result. There are four connection surfaces used. In this experiment, the data points around the junctions of the handle and the main body are very close to one another, which blurs the external energy field.

The last experiment shows how to handle the situation when an insufficient number of control points are employed during

fitting. The data is collected from the famous Renault part using the 3-D rangefinder in our lab. A method to determine the correct number of control points ahead of time is still open, since it takes higher level information. The number of control points should be enough to capture the global shape; otherwise the fitting surface might just approximate part of the object. In this case, we need to process the residual data points, as discussed in previous sections, and add a connecting surface in between. Alternatively, we can simply add more control points to the first iteration until there are no residual data points left after fitting. The second approach gives a more compact representation than the first one, so we prefer to add more control points. We use the second approach only when the residual data points result from the difference in topology between the fitting surface and the underlying object. An example showing what happens when an insufficient number of control points is chosen is in Fig. 24. In this example, a 10×12 mesh is employed first to approximate a complex object (Renault part). The fitting surface fails to account for all data points on the first try. Therefore the system automatically fits another 10×12 mesh to the residual data points and gives two elements (subobjects) as the result. In this situation, a connecting surface with a total of 16 control points is added automatically by the system to connect these two elements. Figure 24a shows two separate subobjects represented by two 10×12 meshes. Figure 24b shows the result with a connecting surface between the two subobjects. Figure 24c shows the result using a 14×16 mesh.

6. TIME COMPLEXITY ANALYSIS

In this section, we analyze the system to pinpoint its bottlenecks.

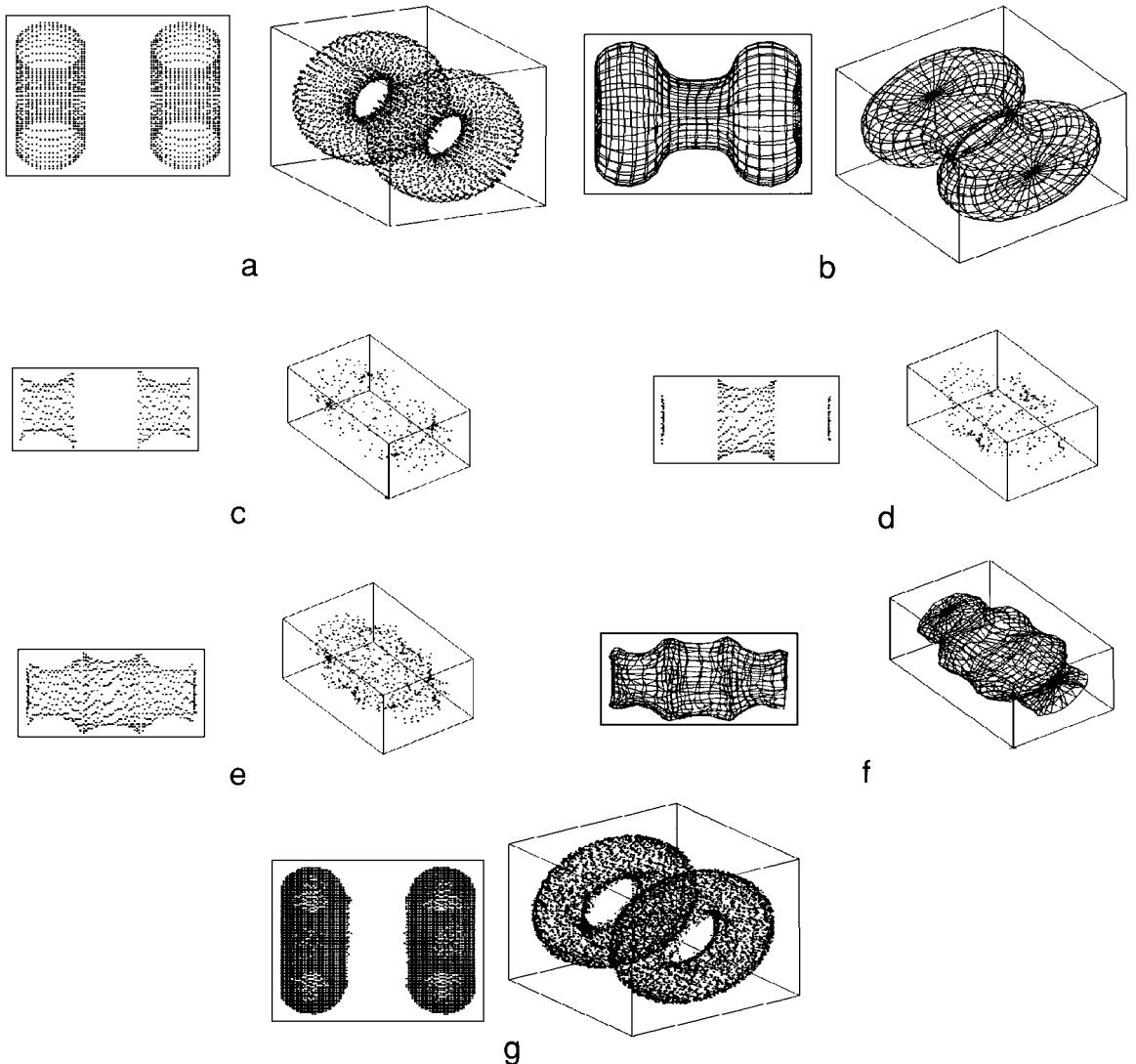


FIG. 20. Segmentation and surface fitting for two tori. (a) 3-D data points. (b) Object A after the first fit. (c) Residual of data points. (d) Bad parts of the fitting surface. (e) Cluster of data points in (c) and (d). (f) Object B after fitting points in (e). (g) Result of the Boolean operation $A \setminus B$.

There are two main subsystems in our work. The first fits the underlying object with a deformable model and is a 3-D surface fitting problem. The second is to connect the elements segmented by the first system properly, which amounts to a 2-D snake fitting problem. The 3-D surface fitting step is much more expensive. The bottleneck obviously is in the first subsystem, and the second takes only a fraction of the overall computation time. Hence, when it comes to performance, we will ignore the second subsystem for connecting elements, and pay attention to the first one only. The surface fitting algorithm is detailed in Section 4.3 and [20, 21].

6.1. Analysis of the Bottleneck

Now, we would like to find the bottleneck of the entire fitting process. We conducted an experiment on a sphere and recorded

the time spent in each step. The sphere is composed of 4413 data points sampled from a synthetic sphere of radius 23 as shown in Fig. 25. We first project a sphere of radius 23 into a $50 \times 50 \times 50$ cube. The voxels traversed by this sphere are picked as the data points, and then random noise between 0.0 and 1.0 is added to each of them. The fitting result is also shown in Fig. 25.

There are three main modules in the system which are for cap fitting, curve fitting, and mesh fitting. Surface fittings (including cap, curve, and mesh fittings) have been repeated 16 times. The mesh fitting module is the main bottleneck (see Table 1).

The number of variables of cap fitting is smaller than the curve fitting's, but the computation times are roughly the same. It is because in cap fitting, $\sin(\cdot)$ and $\cos(\cdot)$ functions are evaluated frequently during minimization, and these two functions are expensive to compute. The performance of cap fitting could be further improved by building a table for these two functions.

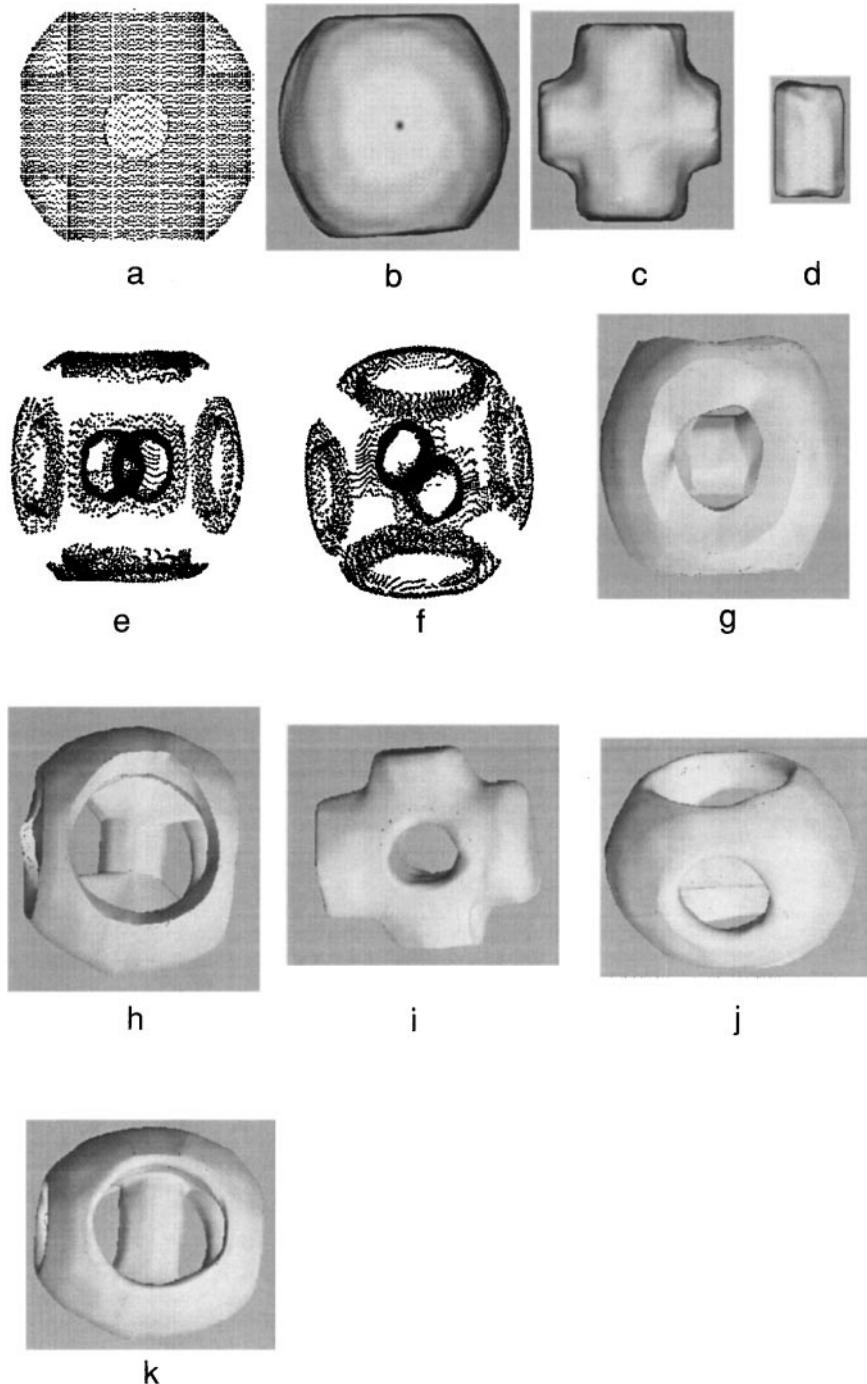
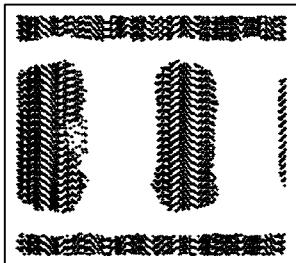


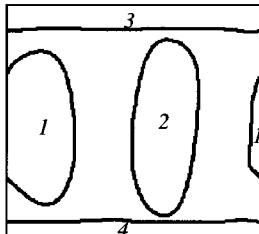
FIG. 21. Segmentation and surface fit for a complex synthetic object. (a) Input data; (b) Outer shell S_1 ; (c) Negative part S_2 ; (d) Bar S_3 inside S_2 ; (e) View 1 of residual data points; (f) View 2 of residual data points; (g) View 1 of the initial connecting surfaces; (h) View 2 of the initial connecting surfaces; (i) Negative part $S_2 \setminus S_3$ of S_1 ; (j) View 1 of the final result $S_1 \setminus (S_2 \setminus S_3)$; and (k) View 2 of the final result $S_1 \setminus (S_2 \setminus S_3)$.

Supposing K and K^2 points are sampled from each span and patch during curve and mesh fitting, respectively, the evaluation of the external energy of a patch would be K times, at least, more expensive than a span's. K is 8 in most experiments. Moreover, computing a surface point (using tensor product) is more expensive than computing a curve point. So computing the external energy of the mesh would be at least $2K$ times more expensive.

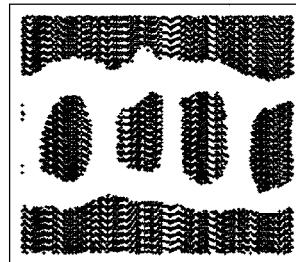
The performance of the minimization routine highly depends on the number of variables and the cost to evaluate the objective function. In terms of the number of variables involved during



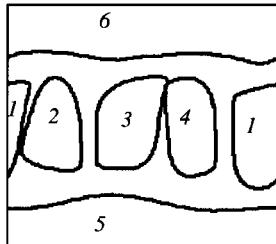
a



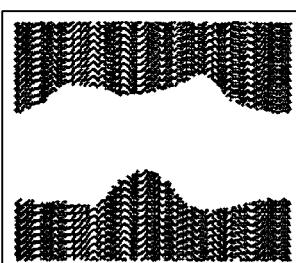
b



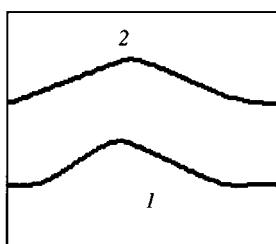
c



d



e



f

FIG. 22. Boundary curves in parameter space. (a) Parameter space of S1. (b) Initial boundary curves of S1 in parameter space. (c) Parameter space of S2. (d) Initial boundary curves of S2 in parameter space. (e) Parameter space of S3. (f) Initial boundary curves of S3 in parameter space.

TABLE 1
Time for Each Module

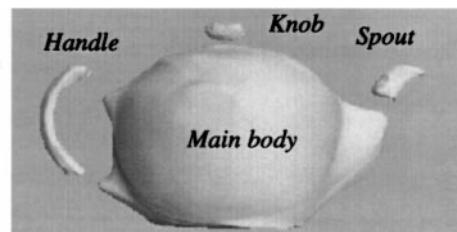
	Average time for each fitting	Percentage
Cap fitting	2.70	2.70/38.36 = 7.04%
Curve fitting	2.58	2.58/38.36 = 6.73%
Mesh fitting	33.08	33.08/38.36 = 86.23%

fitting, the complexity of the curve and mesh fittings is approximately the same. But in terms of the cost of function evaluation, the complexity of the mesh fitting would be about $2K$ more expensive, which explains the significant difference in computation time. One way to remove the bottleneck would be using a multiple processor computer, so all of the patch points and span points can be computed at the same time.

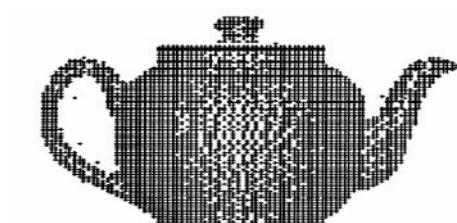
Furthermore, since all curve fittings are independent of one another, all of the curve fittings can go in parallel, and the same goes for the mesh fittings. This way, we can boost the performance of the system even more.

7. CONCLUSION AND LIMITATIONS

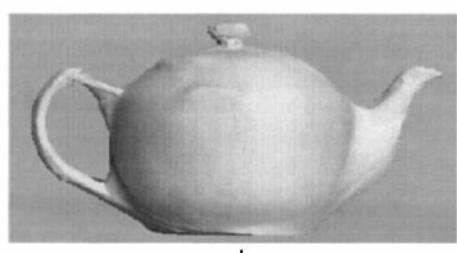
Deformable models provide a compromise between faithfulness and smoothness. With more smoothness, some important features, such as sharp edges, might be blurred. Placing too much weight on faithfulness might lead to jagged fitting results. There is always such a trade-off for deformable models. In our algorithm, we prefer more weight on smoothness because we emphasize the outer contour, which can be detected more stably and easily with more weight on smoothness. The inner parts and the features (such as discontinuity) of the underlying object emerge after applying Boolean operations.



b



a



d

FIG. 23. Experiment on a genus 1 object (teapot). (a) Data points; (b) result of part segmentation; (c) four initial connecting surfaces; and (d) final surface fit.

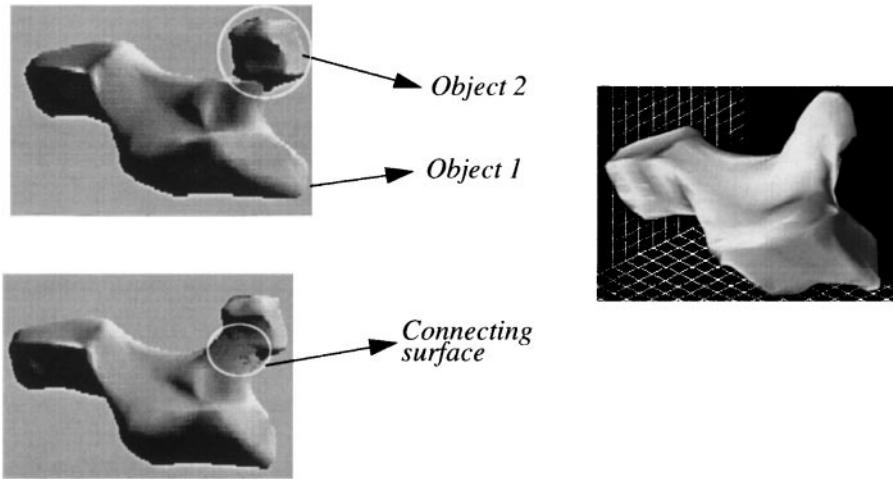


FIG. 24. Illustration of the number of control points. Add a connecting surface when the target is divided into two elements due to the insufficient number of control points for the fitting surface. (a) Two elements (10×12) due to insufficient initial control points. (b) Result of adding a connecting surface (16 control points). (c) Result of adding more control points to the fitting surface (14×16).

The assumptions of (1) one underlying object only, and (2) geometrically simple object without deep (or through) cavities have been the weakness points of the deformable model algorithms. In this paper, we have presented (1) a hierarchical tree structure, whose nodes are the outer profile of the objects, to represent the objects, and (2) a 3-D G^1 surface representation for the topologically complex object. The assumptions above are lifted, and multiple topologically complex objects can be handled properly.

By applying multiple snakes (or fitting surfaces) simultaneously and Boolean operations recursively, objects can be segmented into independent primitives, and cavities can also be well handled. Our algorithm makes the deformable model much more versatile. It breaks down the input data points into several disconnected components, each of them standing for an underlying object or subobject. Each object detected is the result of Boolean operations on its components. The elements of the object extracted first are disconnected and represented by closed B-spline surfaces. A deformable scheme, based on Bézier blending surfaces, is used to connect the elements smoothly, and a smooth surface representation for complex objects can be obtained. The system proceeds automatically without human interaction.

It is difficult to determine the right number of control points for an unknown object ahead of time. Using too many control points might lead to a waste in time while using too few might lead to a result that captures only part of the underlying object. The proposed algorithm here works when there are not enough control points used, which provides another alternative to simply adding more control points and fitting again.

There are also limitations to this approach:

First, we need complete a data set, which is not always easy to obtain. We cannot handle incomplete data points mainly because we do not make any assumptions about the underlying objects in the current implementation. We can always cast the assumptions and prior knowledge, such as size and shape, on the underlying objects into internal energies. This approach could handle the incomplete data points if we have some assumptions or prior knowledge. For example, we could reconstruct a sphere easily using partial data points if we know ahead of time that the underlying object is a sphere.

Second, we assume that the connecting surface exists only between two adjacent elements. This assumption holds most of the time because the connecting surface only accounts for the limited number of residual data points between two adjacent elements. The adjacent elements usually overlap or are very close to each other. We cannot handle the case of more than two elements joined together at one place.

Third, the representation of the object is not unique. This problem could be alleviated by using the same amount of data points on the fitting surface and taking the distribution of the data points into account when initializing the fitting surface. Without noise, ideally, we should always be able to get a unique representation.

Finally, while our scheme is able to decompose an object into elements, there is no direct relationship between these elements and perceptual parts of these objects, and our next step is to

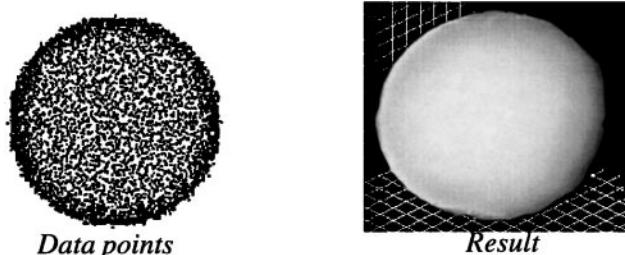


FIG. 25. 4413 data points sampled from a synthetic sphere of radius 23 and the fitting result.

segment “real” parts of the underlying object by analyzing the G^1 surface obtained by our current system.

REFERENCES

1. M. Kass, A. Witkin, and D. Terzopoulos, Snakes: Active contour models, *Int. J. Comput. Vision*, January 1988, 321–331.
2. D. Terzopoulos, A. Witkin, and M. Kass, Constraints on deformable models: Recovering 3-D shape and nonrigid motion, *Artificial Intell.* **36**, 1988, 91–123.
3. S. Menet, P. Saint-Marc, and G. Medioni, B-snakes: implementation and application to stereo, in *Proceedings of Image Understanding Workshop 1990, Pittsburgh, September, 1990*, pp. 720–726.
4. H. Blum, A transformation for extracting new descriptors of shape, in *Proceedings of Symposium on Models for Perception of Speech and Visual Form* (W. Whalen-Dunn, Ed.), MIT Press, Cambridge, MA, 1967.
5. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C, The Art of Scientific Computing*, Chap. 10, Cambridge Univ. Press, Cambridge, UK.
6. R. P. Brent, *Algorithms for Minimization without Derivatives*, Chap. 7, Prentice-Hall, Englewood Cliffs, NJ, 1973.
7. F. S. Acton, *Numerical Methods That Work*, pp. 464–467, Harper and Row, New York, 1970.
8. D. A. H. Jacob, Ed., *The State of the Art in Numerical Analysis*, pp. 259–262, Academic Press, London, 1977.
9. E. Polak, *Computational Methods in Optimization*, Academic Press, New York, 1971.
10. J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, New York, 1980.
11. H. Delingette, M. Herbert, and K. Ikeuchi, Shape representation and image segmentation using deformable surfaces, in *Computer Vision and Pattern Recognition, 1991*, pp. 467–472.
12. L. D. Cohen and I. Cohen, A finite element method applied to new active contour models and 3-D reconstruction from cross sections, in *Proc. Third International Conference on Computer Vision, Osaka, Japan, 1990*, pp. 587–591, IEEE Comput. Soc. Conf.
13. D. Terzopoulos and K. Waters, Analysis of facial images using physical and anatomical models, in *Third International Conference on Computer Vision, Osaka, Japan, 1990*, pp. 727–732.
14. I. Carlom, D. Terzopoulos, and K. M. Harris, Reconstruction and visualizing models of neuronal dendrites, in *Science Visualization of Physical Phenomena* (N. M. Patrikalakis, Ed.), pp. 623–638, Springer-Verlag, New York, 1991.
15. G. Taubin, An improved algorithm for algebraic curve and surface fitting in *International Conference on Computer Vision, May, 1993*.
16. G. Taubin, F. Cukierman, S. Sullivan, J. Ponce, and D. J. Kriegman, Parameterizing and fitting bounded algebraic curves and surfaces, in *Computer Vision and Pattern Recognition, 1992*.
17. G. Taubin, R. M. Bolle, and D. B. Cooper, Representing and comparing shapes using shape polynomials, in *Computer Vision and Pattern Recognition, 1989*.
18. R. Szeliski, D. Tonnesen, and D. Terzopoulos, Modeling surfaces of arbitrary topology with dynamic particles, in *Computer Vision and Pattern Recognition, 1993*.
19. C.-W. Liao and G. Medioni, Representation of range data with B-spline surface patches, in *Proceedings of International Conference on Pattern Recognition 1992, Hague, Netherlands, August, 1992*, pp. 745–748.
20. C.-W. Liao and G. Medioni, Surface approximation of a cloud of 3-D points, in *2nd CAD-based Computer Vision Workshop, Champion, PA, 1994*.
21. C.-W. Liao and G. Medioni, Surface approximation of a cloud of 3-D points, *CVGIP: Graphical Models Image Process.* **57**, 1995, 67–74.
22. D. DeCarlo and D. Metaxas, Adaptive shape evolution using blending, in *Proceedings of International Conference on Computer Vision, Cambridge, MA, 1995*, pp. 834–839.
23. D. DeCarlo and D. Metaxas, Blended deformable models, in *Computer Vision and Pattern Recognition, Seattle, WA, 1994*.
24. D. J. Filip, Blending parametric surfaces, *ACM Trans. Graphics* **8**(3), 1989, 164–173.
25. D. Terzopoulos and D. Metaxas, Dynamic 3-D models with local and global deformations: Deformable superquadrics, *IEEE Trans. Pattern Anal. Mach. Intell.* **13**(7), 1991, 703–714.
26. A. Pentland and S. Sclaroff, Closed-form solutions for physically based shape modeling and recognition, *IEEE Trans. Pattern Anal. Mach. Intell.* **13**(7), 1991, 715–729.
27. I. Cohen, L. D. Cohen, and N. Ayache, Introducing deformable surfaces to segment 3-D images and infer differential structures, in *Proceedings of Computer Vision and Pattern Recognition 1991, Hawaii*, pp. 738–739.
28. S. S. Sinha and B. G. Schunck, Surface approximation using weighted splines, in *Proceedings of Computer Vision and Pattern Recognition 1991, Hawaii*, pp. 44–49.
29. S. S. Sinha and B. G. Schunck, Discontinuity preserving surface reconstruction, in *Proceedings of Computer Vision and Pattern Recognition 1991, Hawaii*, pp. 229–234.
30. D. Terzopoulos and M. Vasilescu, Sampling and reconstruction with adaptive meshes, in *Proceedings of Computer Vision and Pattern Recognition, 1991, Hawaii*, pp. 70–75.
31. M. Vasilescu and D. Terzopoulos, Adaptive meshes and shells: Irregular triangulation, discontinuities, and heretical subdivision, in *Proceedings of Computer Vision and Pattern Recognition 1992, Urbana-Champaign, IL*, pp. 829–832.
32. S. Muraki, Volumetric shape description of range data using “Blobby Model,” *Comput. Graphics* **25**(4), 1991, 227–235.
33. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, Surface reconstruction from unorganized points, *Comput. Graphics* **26**(2), 1992, 71–78.
34. F. Solina and R. Bajcsy, Recovery of parametric models from range images: The case for superquadrics with global deformations, *IEEE Trans. Pattern Anal. Mach. Intell.* **12**(2), 1990, 131–147.
35. S. Han, D. B. Goldgof, and K. W. Bowyer, Using hyperquadrics for shape recovery from range data, in *Proceedings of International Conference on Computer Vision, 1993, Berlin, Germany*, pp. 492–496.
36. C. Nastar and N. Ayache, *Fast Segmentation, Tracking, and Analysis of Deformable Objects*, Technical Report 1783, INRIA, France.
37. W. C. Huang and D. B. Goldgof, Adaptive-size meshes for rigid and nonrigid shape analysis and synthesis, *IEEE Trans. Pattern Anal. Mach. Intell.* **15**(6), 1993.
38. G. L. Scott, The alternative snake—and other animals, in *The 1987 Stockholm Workshop on Computational Vision, Stockholm* (J.-O. Eklundh, Ed.), Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, TRITA-NA-P8714 CVAP 47.
39. L. H. Staib and J. S. Duncan, Parametrically deformable contour models, in *Computer Vision and Pattern Recognition, San Diego, CA, 1989*, IEEE Computer Society Press, pp. 98–103.
40. F. Leitner, I. Marque, S. Lavallée, and O. Cinquin, Dynamic segmentation: Finding the edge with differential equations and “spline snakes”, Technique Report TIMBTIM 3-IMAG, Faculte De Medecine, La Tronche, France, 1990.

41. R. M. Curwen, A. Blake, and R. Cipolla, Parallel implementation of lagrangian dynamics for real-time snakes, in *British Machine Vision Conference, Glasgow, 1991* (P. Mowforth, Ed.), pp. 29–35, Springer-Verlag, London.
42. L. D. Cohen and I. Cohen, A finite element method applied to new active contour models and 3-D reconstruction from cross sections, in *Proc. Third International Conference on Computer Vision, Osaka, Japan, 1990*, pp. 587–591. IEEE Computer Society Conference.
43. L. D. Cohen, On active contour models and balloons, *Comput. Vision Graphics Image Process.* **53**, 1991, 211–218.
44. D. Terzopoulos and K. Waters, Analysis of facial images using physical and anatomical models, in *Third International Conference on Computer Vision, Osaka, Japan, 1990*, pp. 727–732.
45. I. Carlbom, D. Terzopoulos, and K. M. Harris, Reconstruction and visualizing models of neuronal dendrites, in *Science Visualization of Physical Phenomena* (N. M. Patrikalakis, Ed.), pp. 623–638, Springer-Verlag, New York, 1991.
46. R. Malladi, J. A. Sethian, and B. C. Vemuri, Evolutionary fronts of topology-independent shape modeling and recovery, in *Proceedings of European Conference on Computer Vision, Stockholm 1994*, pp. 3–13.
47. R. T. Whitaker, Algorithm for implicit deformable models, in *Proceedings of International Conference on Computer Vision, Cambridge, MA, 1995*, pp. 822–827.
48. A. Gueziec, Large deformable splines, crest lines and matching, in *Proceedings of International Conference on Computer Vision, Berlin, Germany, 1993*, pp. 650–657.