

3 Surface Approximation of Complex 3-D Objects

Chia-Wei Liao and Gérard Medioni

Our goal is to generate a surface description of complex objects with parts and holes. We start by fitting a surface, assuming the object is of Genus 0, then analyze the result to further segment the description.

In the first part of our algorithm, the system provides an initial estimated surface which is subject to internal forces (describing implicit continuity properties such as smoothness) and external forces which attract it toward the data points. The problem is cast in terms of energy minimization. We solve this non-convex optimization problem by using the well known Powell algorithm which guarantees convergence and does not require gradient information. The variables are the positions of the control points. The number of control points processed by Powell at one time is controlled. The process is controlled by two parameters only, which are constant for all our experiments.

The above approach is not sufficient for complex objects with cavities, or for more than one object. We therefore propose an approach that can apply simultaneously more than one curve or surface to approximate multiple objects. Using (1) the residual data points, (2) the bad parts of the fitting surface, and (3) appropriate Boolean operations, our approach is able to handle objects more complicated than Genus 0 or with deep cavities, and can perform segmentation if there is more than one underlying object.

3.1 Introduction

Range sensing is a mature technology, and there are many methods, such as time of flight and MRI, collect 3D data based on this technology. In addition to this, 3D data can also be obtained in passive ways like stereo and shape from X methods. The data obtained from the above sources is in the form of points.

But, in computer vision, what we need are some properties such as the curvature, normal, and principal directions. These quantities relate to the underlying surface, which is not made explicit in the original data. Furthermore, it is even more difficult if some ordering relation among the data points is not known. This happens mainly when we gather data points from various sources. Analytical surface construction of a cloud of points (boundary points of the object) becomes important because it is much easier to extract the features from an analytical surfaces. So, we need some tools to construct an analytical description (for example, surface) for the collected 3D

data. A deformable model, which can give a analytical surface representation over a cloud of 3D data points, is able to serve this purpose.

The idea of fitting data by a deformable model can be found in the work of Kass *et al.* [14] in 2D. Such models are generalized in 3D by the same authors [15] for a surface of revolution.

There have been many works [16]-[29] derived from this seminal idea. But most of them either require many parameters, or cannot guarantee convergence (partially due to the use of gradient descent to minimize the energy). Furthermore, they suffer from the following problems:

First, there may be more than one underlying object, and these objects might be close to one another. Most deformable model algorithms assume that there is only one object, that is, the segmentation has been done beforehand. It takes sophisticated segmentation to separate these mixed objects, and it is often the case that segmentation is much more difficult than the fitting process.

Second, they cannot handle very well patterns with deep and narrow cavities. To capture a cavity directly through energy minimization, we need to differentiate between the data points belonging to the cavities from the other points when defining the external energy. But this might lead to a circular problem.

Third, without prior knowledge, most of them are insufficient for objects more complicated than Genus 0.

Our proposed algorithm can deal with these problems appropriately. There are mainly two parts in our algorithm. In the first part, we focus on the Genus 0 surface fitting. We apply a tested numerical method which guarantees convergence. Through an coarse-to-fine approach and a partitioning scheme, the computational time is kept in check. By using the surface fitting algorithm in the first part and appropriate boolean operations, we develop another method that can accomplish data segmentation and handle objects with deep cavities or more complicated than Genus 0.

3.2 Part 1: Genus 0 surface fitting

3.2.1 Issues

Several problems come with the algorithms dealing with deformable models:

1. Huge computational time and space: Assuming $M \times N$ control points on the fitting surface, there are $3MN$ variables for this surface. Theoretically, we can just inject these $3MN$ variables into a minimization algorithm to minimize the energy of the fitting surface. This approach turns out to be impractical due to the unbearable computational time when $3MN$ is large. Furthermore, most minimization algorithms need a matrix of size $(3MN) \times (3MN)$, which also results in huge space complexity. An adaptive approach can just alleviate these problems, but cannot avoid these problems in the worst case, especially when all control points or patches are bad.

2. Constructing a smooth closed surface from the B-spline control points: we obtain a closed surface by closing the top and bottom parts of the rectangular mesh as depicted in Figure 3.1 . The problem is that the poles are not smooth when we try to construct a smooth surface directly from the control points. It is because these two poles are shared by many degenerate patches (triangular patches) around them.

3. A good approximation of the data as the initial surface: the quality of the fitting result depends on the initial surface, and we would like the result to be invariant to the initial surface as long as the initial surface is not too bad.

4. The convergence of the surface to the data points: the convergence of the fitting process should be guaranteed. By using the Powell [30] minimization routine, the convergence is guaranteed.

All problems listed above are well handled by our algorithm for the surface fitting.

3.2.2 Algorithm

Now, we define some terms for further use. We define a Cap to be the triangular patches formed by a Pole and its adjacent control points. So, we always have two Caps. A Meridian (a line of constant u in parameter space) is defined to be the line connecting the two Poles, as depicted in Figure 3.1 .

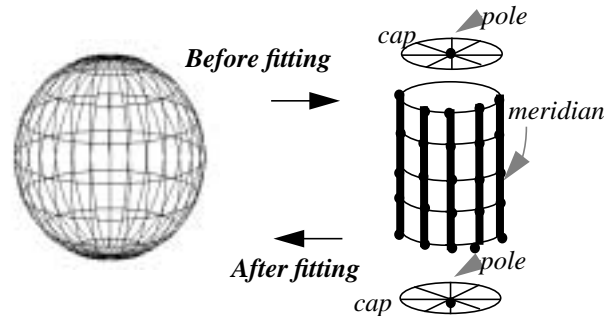


Figure 3.1 The initial closed surface and the definitions of Pole, Meridian, and Cap

A flowchart of our algorithm is in Figure 3.2 .

In our algorithm, we consider a sphere as composed of three parts, which are two caps and an open cylinder as shown in Figure 3.1 . These three parts are processed separately in our algorithm.

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 curve fitting problem followed by a (simpler) mesh fitting problem.

Given that the caps are already in place, we select every other meridian and move their $(M-4)$ control points, which are not shared with the two caps, to minimize their energy. We then select the remaining meridians and move their $(M-4)$ control

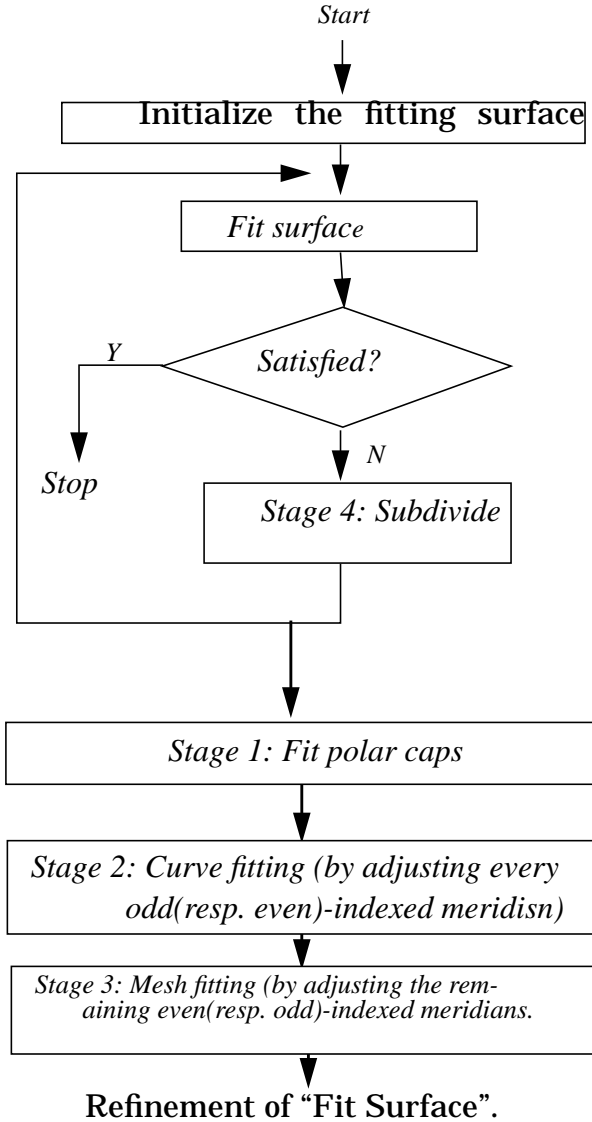


Figure 3.2 The flowchart of the Genus 0 algorithm

points, which are 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 (after the energy minimization, we can guarantee that the associated energy of the bad segments is reduced, but some segments with higher external energy might still be bad). Then, we subdivide all patches in four, and repeat the process until some terminating condition is met.

Our algorithm is a 4-stage one, and during the first three stages, Powell is called frequently for energy minimization.

First, we fit the caps to the data, and force the caps to be planar. This way, all tangent vectors in all directions at the pole are coplanar. When fitting, the cap can change its shape, subject to the planar constraint, in order to get the best fit. By duplicating the control points at the poles and this planar constraint, we can get a smooth caps around the poles when we want to construct a smoother (e.g. C^1) surface.

Second, we perform the curve fitting to some meridians 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 approximating linear B-snake [16]. The only difference between a B-snake and a selected meridian lies in the internal energy. 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 (an example is depicted in Figure 3.8). Then we let them adapt to find the profiles of the target. These selected meridians are not influenced much by the fitting surface (by the internal energy) when moving, so they can detect the object more accurately. Please notice 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. Each strip is bounded by two odd(resp. even)-indexed meridians.

Next, we fit each strip to the target. We select meridians of the other polarity, even(resp. odd)-indexed, to do the mesh 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.

The fourth stage is subdivision (in Figure 3.3). If the fitting surface up to now is

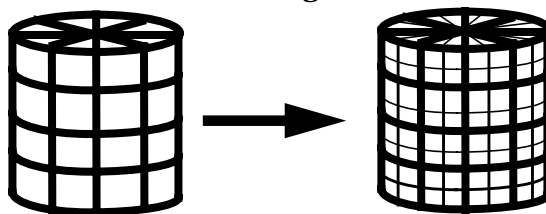


Figure 3.3 Subdivision (except on the caps).

not satisfactory, we subdivide all rectangular patches into four and then go back to stage 1, otherwise, we exit.

3.2.3 Initial guess

What we want to obtain is the outer contour of the target at each stage. The concavities of the objects can be detected later by applying Boolean operations applied at the next stage. We just need to have the initial surface (or curve) covering all data

points. So, the initial fitting surface would 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 each sampled direction. The polygon formed by these extremal data points is used as the initial guess. An illustrative example is shown in Figure 3.4 .

In the 3D case, we have two alternatives. The first one is to use a cylinder covering all data points as the initial surface. In the second approach, we first calculate the center of mass C , too. In order to make the system invariant under translation and scaling, we compute the three eigen vectors of the covariance matrix of the data points, and then use these orthogonal vectors to define another coordinate system with the origin at C . We define the sampled directions according to this coordinate system, and find the farthest data point in each sampled direction. With these farthest points, we can obtain a fairly good initial estimate on the data points. In both approaches, the caps of the initial surface must be initialized to be planar. Examples for these two approaches are in Figure 3.5 and Figure 3.6 , respectively.

In both 2D and 3D 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.

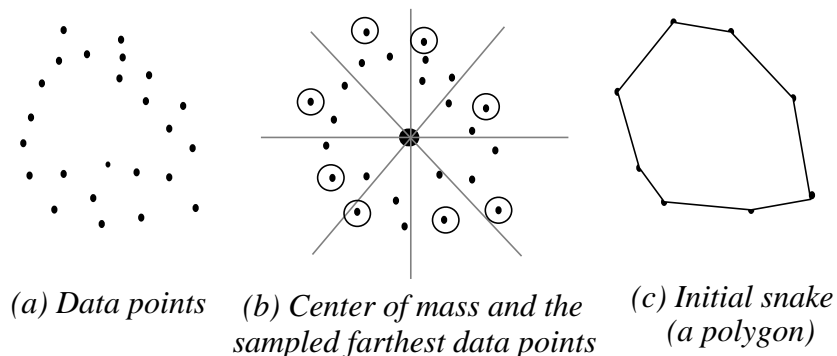


Figure 3.4 Initial guess

3.2.4 Parameters

There are only two important parameters, $ERROR_{\text{threshold}}$ and $RATIO_{\text{ext-to-int}}$, set by the user in this algorithm.

$ERROR_{\text{threshold}}$ is used to determine whether or not a patch of the surface or a span of the snake is good. We only process the bad parts of the snakes and the bad patches on the fitting surface during each iteration. At each iteration, a patch (or a span) is good if its average external energy is smaller than $ERROR_{\text{threshold}}$; otherwise, it is bad.

$RATIO_{\text{ext-to-int}}$ specifies the relative importance of the external energy with respect to the internal energy. After setting $RATIO_{\text{ext-to-int}}$, two internal parameters

W_{ext} and W_{int} , concerning the weights of the external and internal energies, are set by the system. W_{ext} is always 1, and W_{int} is set as below:

$$W_{\text{int}} = \frac{E_{\text{current-ext}}}{E_{\text{current-int}} \times \text{RATIO}_{\text{ext-to-int}}}$$

Where $E_{\text{current-ext}}$ is the current external energy of the fitting surface or snake, and $E_{\text{current-int}}$ is the current internal energy of the fitting surface or snake.

Every time Powell is invoked, W_{int} is re-calculated based on $\text{RATIO}_{\text{ext-to-int}}$ and the current internal and external energies of the fitting surface. So every time, Powell may be called with different W_{int} .

The reasons why we set $\text{RATIO}_{\text{ext-to-int}}$ is that W_{ext} and W_{int} are different measures and thus on different scales. $\text{RATIO}_{\text{ext-to-int}}$ serves to normalize two energies. $\text{RATIO}_{\text{ext-to-int}}$ is always greater than 1; otherwise, the fitting surface is unlikely to conform to the data points, as we now explain.

As we subdivide the surface after each iteration, the fitting surface is approaching the real object. We have more confidence in the fitting surface. So it is suggested that the internal energy weight be reduced as the process goes on. One more advantage of $\text{RATIO}_{\text{ext-to-int}}$ is that the weight of the internal energy tends to decrease as the process goes on, because E_{ext} decreases faster than E_{int} does when $\text{RATIO}_{\text{ext-to-int}}$ is greater than 1. By setting $\text{RATIO}_{\text{ext-to-int}}$ to be a constant greater than 1, we can diminish the importance of the internal energy after each iteration implicitly, and thus obtain a better fitting surface.

On the contrary, if we set W_{int} directly and keep it unchanged, then the internal energy tends to dominate at the later iterations because the external energy decreases faster than the internal energy does. This might not lead to a good fit.

These two parameters could be constants in most cases, which means we can use the same values for most data regardless of the complexity of the underlying object (because the weight of the internal energy decreases automatically as the fitting process goes on). Thus, we can assume the underlying object is smooth, and assign a liberal weight to the internal energy. We use the same parameter values in our experiments.

3.2.5 Summary

In summary, in the first step, we fit the caps. Next, the odd(resp. even)-indexed meridians are used to find the profile or frame of the target, and the surface is divided into strips by these meridians. Then the even(resp. odd)-indexed meridians are applied to fit the strips to the data. Finally in stage four, we subdivide the surface, that is, we divide each rectangular patch at its center into four (we can avoid degenerate patches this way). We break the 3D surface problem into a set of 2D (linear) B-snake

problems in a way. Also notice that, at each step, although we have different ways of calculating the internal energy E_{int} and the external energy E_{ext} the basic idea is the same.

We can see that this is a typical coarse-to-fine approach. We start with few control points and large patches, then we increase the number of the patches and control points at later iterations.

Our algorithm overcomes the time and space complexities, and closed surface problems by (1) breaking the 3D surface problems into several 2D snake problems, which is shown in stages 2 and 3, (2) coarse-to-fine approach, and (3) forcing the cap to be planar. Due to the robustness of Powell, we do not need a good initial guess. Also, the Powell method guarantees convergence.

Please notice that stages 2 and 3 can be performed very fast when there are few control points. We can take advantage of this to get more reliable global information. The computational time could also be largely reduced by parallel processing. It is obvious that (1) the two caps are independent of each other, (2) all odd(resp. even)-indexed meridians are independent of one another, and (3) all even(resp. odd)-indexed meridians are independent of one another, so each stage can be performed in parallel.

3.2.6 Experiments

$ERROR_{threshold}$ and $RATIO_{ext-to-int}$ are 1.0 and 10 in all of our experiments. The first experiment on the head is performed on Sparc 10, and the second on IRIS Indigo.

Figure 3.5 shows the evolution of the fitting surface with the cylindrical initial surface, and both the shaded and wire-frame results are shown. The average external energy of the surface point is initially 20.15 voxels, 1.43 voxels after the first iteration, 1.21 voxels after the second iteration (one sub-division has been done), and 1.18 voxels after the third iteration (two sub-divisions have been done).

An experiment on the Renault part (see Figure 3.6). Around 3820 control points are used. The maximum point error of the fitting surface is 2.0, and the average is 0.37 voxel. The running time here includes the time for constructing the energy field, so it is longer than those for the other experiments. There are three iterations (two subdivision). A $200 \times 200 \times 200$ cube is used to store the external energy. The average error is 10.54 voxels initially, 1.04 voxels after the first iteration, 0.41 voxel after the second iteration, and 0.37 voxel finally. The distribution of the error is also shown. (a) shows the original data. (b) is the shaded result. (c) is the initial surface. (d), (e), and (f) are the results after each iterations. (g), (h), (i), and (j) show the patches with an error above 0.3, 0.6, 0.9, and 1.2, respectively. There are only three patches with an error over 1.2 (and less 2.0). The detailed information is on Tables 1 and 2.

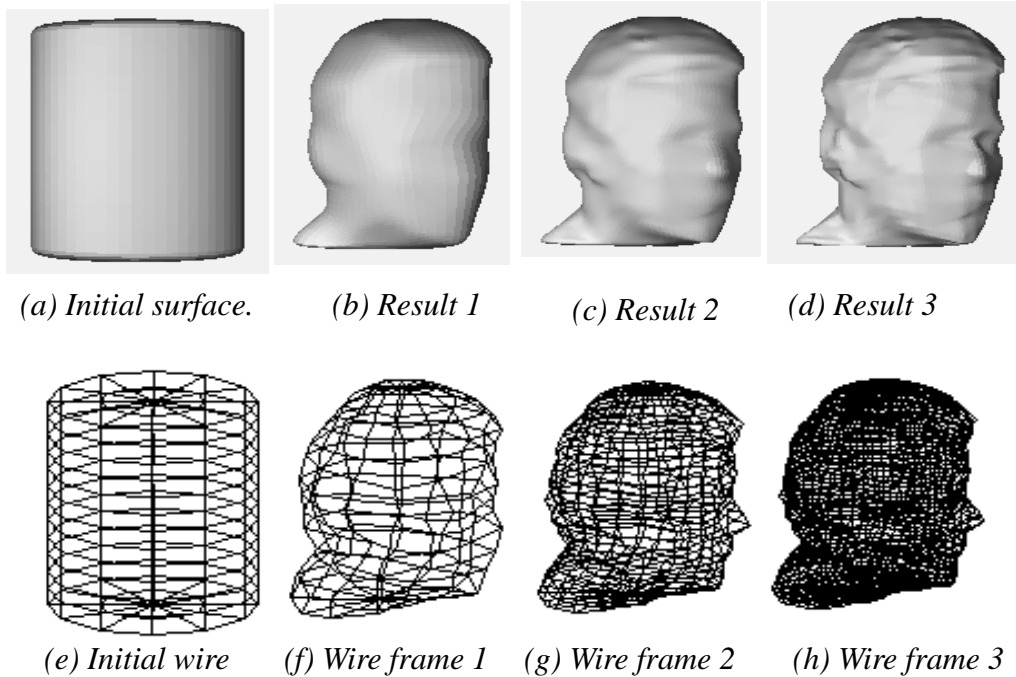


Figure 3.5 The evolution of the experiment of head 1. (a) is the initial surface (cylinder). (b), (c), and (d) are the deformed results for each iteration. The surface has been sub-divided twice. (e), (f), (g), and (h) show the wire frames of (a), (b), (c), and (d).

3.3 Part 2: Surface fitting for complex objects

3.3.1 Issues

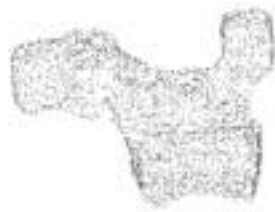
Most deformable algorithms are deficient when (1) there are multiple underlying objects, (2) there are deep cavities, or (3) the underlying objects are more complicated than Genus 0. These are the problems we want to resolve here.

Table 1: Performance on the Renault part experiment

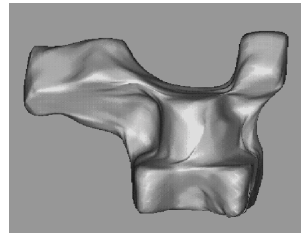
	Initial surface	first iteration	second iteration	third iteration
Error	10.54	1.04	0.41	0.37
Control points	16×18	16×18	32×33	64×63
Computation time (min.)	0.1	21	19	3

Table 2: General information on the Renault part experiment

Number of data points	Time for constructing energy field	No of subdivisions
214100	2.16 minutes	2



(a) Original data



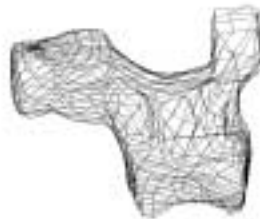
(b) Shaded result



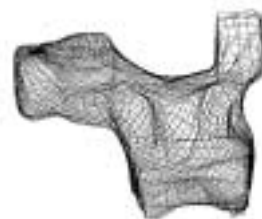
(c) Initial surface



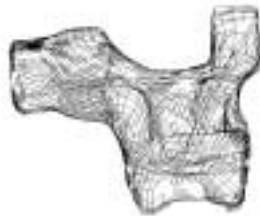
(d) First fit



(e) Second fit



(f) Third fit



(g) Patches with an error > 0.3



(h) Patches with an error > 0.6



(i) Patches with an error > 0.9



(j) Patches with an error > 1.2

Figure 3.6 Experiment on a Renault part.

3.3.2 Algorithm

Our proposed approach is to use a *hierarchy*. We would like to handle a complicated object by representing it as a tree. The underlying object can be obtained by applying Boolean operations recursively to the tree. Every node in the tree, including the root, is supposed to be a simple object without narrow cavities or inner tunnels, that is, each node contains the outer contour of some object. We still use the energy fields mentioned in the previous section to detect the outer contour.

Let B be the outer contour first found. Then, we isolate residual data points that are not well fitted, and cluster these residual data points into groups. Next, we find those bad parts of the fitting curve with high external energy. For each bad part, we check if there is a group of residual data points connected with it. If so, we merge it into this group, otherwise we consider this bad part good because no data points are nearby. Now, we have groups of points. We treat each group of bad data points as an object and find out its contour recursively. Let P be one of the contours.

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

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

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

An illustrative 2D example is given in Figure 3.7. (a) shows a complex object

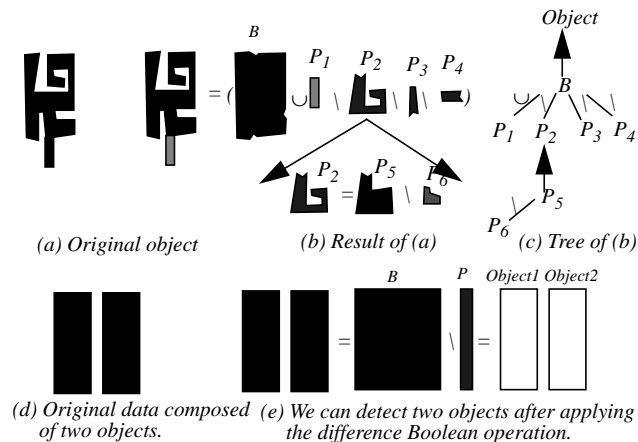


Figure 3.7 Two illustrative examples of object decomposition.

with deep cavities. In (b), by finding the outer profile only, and 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 original object can be restored by applying the appropriate union and difference Boolean operations. (c) shows the tree associated with the result in (b). (d) 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 (e).

3.3.3 Experiments

In the first experiment (in Figure 3.8), we use hand-made data, which consists

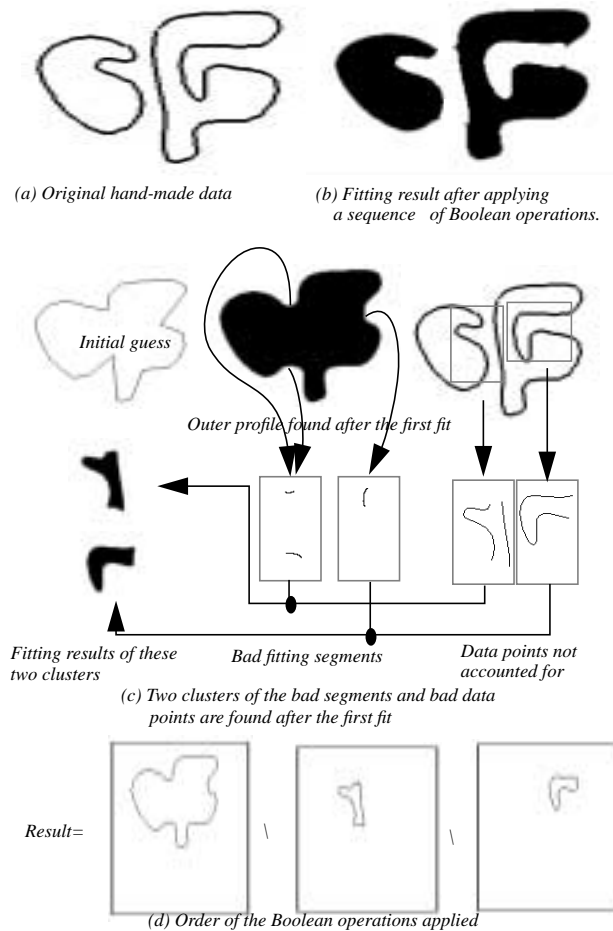


Figure 3.8 Segmentation on two concave patterns.

of two simple objects with deep concavities. After applying a B-snake and appropriate Boolean operations, these two objects are differentiated. (a) is the original data, and (b) is the final result. (c) shows the initial guess, and how the residual data points and the bad B-snake segments merge into two clusters (which form the negative parts of the target). (d) 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 dif-

ference Boolean operations are applied. Finally, the original contour is separated into two

In the second experiment (in Figure 3.9), the data points are the same as those

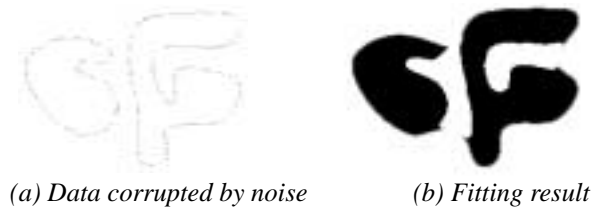


Figure 3.9 Experiments on noisy data.

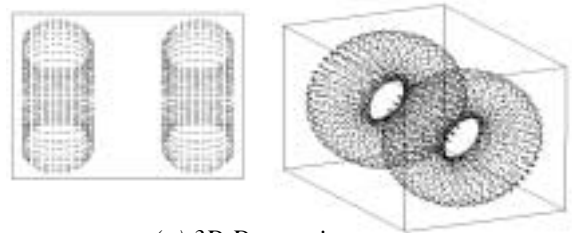
in Figure 3.8 except that (1) half of the data points are deleted randomly, and (2) the data points are randomly shifted at most 3 pixels. (a) shows the data points, which have broken boundaries. (b) is the result. The sequences of the Boolean operations applied are exactly the same as those in the previous 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 third experiment, in Figure 4.10, is on 3D data which is composed of two separate genus 1 toruses. (a) shows the data points, (b) is the result (object A) after the first fit, which results in a dumbbell-like shape, (c) is the residual of the data points that are not accounted for by object A. They are from the inner parts of the two toruses, and (d) shows the bad parts of object A without data points nearby. They are from the two ends and middle of object A. (e) is the merger of points in (c) and (d). (f) is the fitting result (object B) to points in (e). Because object B is inside object A, a difference operation $A \setminus B$ is applied, which leads to two separate entities. (g) shows the shells of the two separate entities, which are toruses. In this experiment, objects (two separate toruses) more complex than genus 0 are well handled, and the data segmentation, which segments the data points into two parts, is automatically done after fitting.

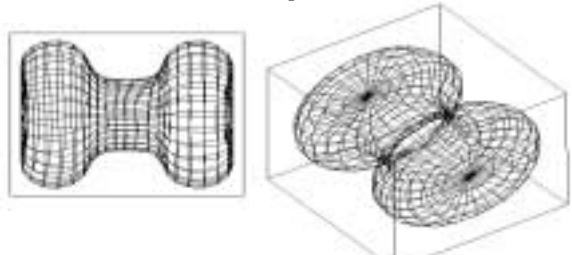
3.4 Discussion

There are several important aspects in this paper:

- Our new scheme is a coarse-to-fine approach. It divides all patches after each iteration. It is efficient because if a patch is really good, then the only operation applied to it in the future is just sub-division, which costs very little. This scheme also preserve the rectangular structure of the surface after each sub-division, which makes generating smooth surface easier and cheaper. This approach is free from the degenerate patch problem because a rectangular patch is always divided into 4 rectangular ones. We prefer the rectangular mesh to the triangular mesh because it is much easier to construct a smoother surface from



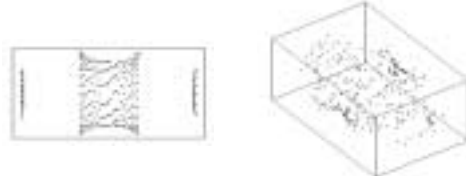
(a) 3D Data points.



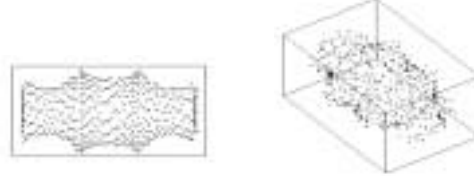
(b) Object A after the first fit.



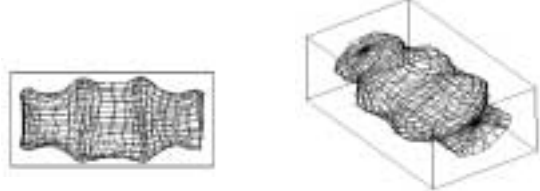
(c) Residual of data points.



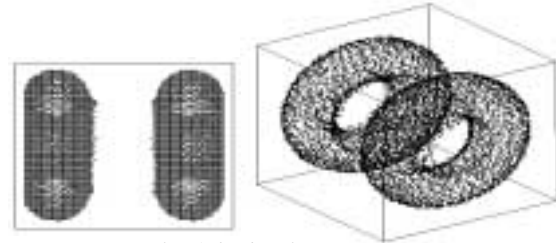
(d) Bad parts of the fitting surface.



(e) Merger of data points in (c) and (d).



(f) Object B after fitting points in (e).



(g) Result of the boolean operation $A \setminus B$.

Figure 4.10 Experiment on two tori

the rectangular mesh, and the properties, such as derivatives, are much easier to obtain.

- There is always a large matrix associated with the minimization algorithm, and the size of the matrix is proportional to the square of the number of the variables. This might result in the memory explosion if there are many control points to handle at a time. Also, the numerical method goes extremely slow under this situation. With the partitioning scheme, we break a 3 dimensional problem down into several 2 dimensional problems, and then the space and time complexities can be reduced significantly. We separate the surface into several strips, so Powell is always called with a limited number of variables. For example, if the fitting surface has $M \times N$ control points, the maximum number of variables sent to Powell is around $3 \times (N-4)$. Only the bad parts of the strips and the meridians are tuned by Powell. So, in practice, the number of variables is far below $3 \times (N-4)$. The caps only have $(N+5)$ variables, which is also low.
- We reduce the weight of the internal energy implicitly as the iteration goes on, because we have more confidence in the fitting surface after each iteration. This way, the discontinuities of the data can be well preserved.
- We use the Powell minimization routine which is more stable, robust, and accurate than the gradient descent approach.
- Due to the independency among the caps and meridians, our algorithm could run in parallel.
- This system is easy to control because there are only two global parameters to adjust.
- The assumptions of (1) one underlying object only, (2) the availability of good initial guess, and (3) geometrically simple objects without deep cavities have been the weakness points of the deformable model algorithms. By applying multiple snakes simultaneously and Boolean operations, objects can be segmented into independent ones, and cavities can also be well handled. Our algorithm makes the deformable model much more versatile.

3.5 Future work

We would like to upgrade all algorithms in this paper completely to 3D ones, and build a working system for both 2D and 3D. In addition, we would like to work out a better 3D surface representation which can handle multiple objects and objects more complicated than Genus 0.

References

- [14] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models", in *International Journal of Computer Vision*, January 1988, pp.321-331.

- [15] D. Terzopoulos, A. Witkin, and M. Kass, "Constraints on deformable models: Recovering 3D Shape and Nonrigid Motion", *Artificial Intelligence*, Vol. 36, 1988, pp. 91-123.
- [16] Sylvie Menet, Philippe Saint-Marc, and Gérard Medioni, "B-snakes: implementation and application to stereo", in *Proceedings of Image Understanding Workshop 1990*, pp.720-726, Pittsburgh, September, 1990.
- [17] Brent, Richard P. 1973, in *Algorithms for Minimization* Laurent D. Cohen, "On Active Contour Models and Balloons", in *Computer Vision, Graphics, and Image Processing*, Vol. 53, No. 2, March 1991, pp.211-218.
- [18] H. Delingette, M. Hebert, K. Ikeuchi, "Shape Representation and Image Segmentation Using Deformable Surfaces", in *Computer Vision and Pattern Recognition*, 1991, pp467-472.
- [19] Scott, G. L. (1987), "The alternative snake - and other animals", in Eklundh, J.-O., editor, *The 1987 Stockholm Workshop on Computational Vision*, Stockholm. Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, TRITA-NA-P8714 CVAP 47.
- [20] Staib, L.H. and Duncan, J.S. (1989). "Parametrically Deformable Contour Models" in *Computer Vision and Pattern Recognition, 98-103, San Diego, CA. IEEE Computer Society Press*.
- [21] Leitner, F., Marque, I., Lavallée, S., and Cinquin, O. (1990)., "Dynamic Segmentation: Finding the Edge with Differential Equations and 'Spline Snakes'". *Technique Report TIMBTIM 3-IMAG, Faculte De Medecine, La Tronche, France*.
- [22] Curwen, R. M., Blake, A., and Cipolla, R. (1991). "Parallel Implementation of Lagrangian Dynamics for Real-Time Snakes", In Mowforth, P., editor, *British Machine Vision Conference, 29-35, Glasgow. Springer-Verlag, London*.
- [23] Cohen, L.D. and Cohen, I. (1990), "A Finite Element Method Applied to New Active Contour Models and 3D Reconstruction from Cross Sections", in *Proc. Third International Conference on Computer Vision*, 587-591. IEEE Computer Society Conference. Osaka, Japan.
- [24] Terzopoulos, D. and Waters, K. (1990), "Analysis of Facial Images Using Physical and Anatomical Models", in *Third International Conference on Computer Vision*, 727-732, Osaka, Japan.
- [25] Carlbom, I., Terzopoulos, D., and Harris, K. M. (1991), "Reconstruction and Visualizing Models of Neuronal Dendrites" in Patrikalakis, N. M., editor, *Science Visualization of Physical Phenomena*, 623-638. Springer-Verlag, New York.
- [26] Gabriel Taubin, "An Improved Algorithm for Algebraic Curve and Surface Fitting" in *International Conference on Computer Vision*, May, 1993.

- [27] Gabriel Taubin, Fernando Cukierman, Steven Sullivan, Jean Ponce, and David J. Kriegman, “Parameterizing and Fitting Bounded Algebraic Curves and Surfaces” in *Computer Vision and pattern Recognition*, 1992
- [28] Gabriel Taubin, Ruud M. Bolle, and David B. Cooper, “Representing and Comparing Shapes Using Shape Polynomials”, in *Computer Vision and pattern Recognition*, 1989.
- [29] Richard Szeliski, David Tonnesen, and Demetri Terzopoulos, “Modeling Surfaces of Arbitrary Topology with Dynamic Particles”, in *Computer Vision and pattern Recognition*, 1993
- [30] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, in *Numerical Recipes in C, The Art of Scientific Computing* (Cambridge), Chapter 10.

