Graphical Models **62,** 2–18 (2000)
doi:10.1006/gmod.1999.0515, available online at http://www.idealibrary.com on IDEAL®

Hierarchical Solutions for the Deformable Surface Problem in Visualization

Christoph Lürig, Leif Kobbelt, and Thomas Ertl

University of Erlangen, Computer Graphics Group (IMMD IX), Am Weichselgarten 9, D-91058 Erlangen, Germany
E-mail: cpluerig@informatik.uni-erlangen.de, kobbelt@informatik.uni-erlangen.de, ertl@informatik.uni-erlangen.de

Received November 25, 1998; revised August 16, 1999; accepted October 13, 1999

In this paper we present a hierarchical approach for the deformable surface technique. This technique is a three dimensional extension of the snake segmentation method. We use it in the context of visualizing three dimensional scalar data sets. In contrast to classical indirect volume visualization methods, this reconstruction is not based on iso-values but on boundary information derived from discontinuities in the data. We propose a multilevel adaptive finite difference solver, which generates a target surface minimizing an energy functional based on an internal energy of the surface and an outer energy induced by the gradient of the volume. The method is attractive for preprocessing in numerical simulation or texture mapping. Red-green triangulation allows adaptive refinement of the mesh. Special considerations help to prevent self interpenetration of the surfaces. We will also show some techniques that introduce the hierarchical aspect into the inhomogeneity of the partial differential equation. The approach proves to be appropriate for data sets that contain a collection of objects separated by distinct boundaries. These kind of data sets often occur in medical and technical tomography, as we will demonstrate in a few examples.

1. INTRODUCTION

The aim of the presented technique is to detect boundary structures in a volumetric data set using a hierarchical solver for the deformable surface problem. Those boundary structures are indicated by large gradient magnitudes. The contour model that we are going to use has been developed in the pattern recognition community and it is known as the snake concept for the two dimensional case (Kass *et al*. [6]). The snake is a curve embedded into a scalar data field f defined on $\mathbf{R}^2$ that minimizes a potential energy, which consists of an internal and an external component. The external part is the negative of the gradient

1524-0703/00 $35.00

magnitude ∇f. This attracts the snake to the boundaries. Many considerations concerning this external force can be found in Cohen *et al*. [3]. Internal forces are introduced to stabilize the convergence of this method. These forces tend to minimize a weighted sum of the first and second order derivatives of the curve. The external forces account for the structure of the data while the internal forces provide some global regularization properties (see also Neuenschwander [10]).

First extensions of this concept to three dimensions were based on surfaces with spine topology, which yields a three dimensional model whose projection into an image plane fits a given image (Terzopoulos *et al*. [18]).

An application of this concept to tomographical data sets has been presented by Snell *et al*. [16]. They segment brain surfaces of MRI scans by deforming an initial brain atlas that is parameterized over four individual domains.

In our approach we are using manifolds of arbitrary topology. This approach is intended for visualization purposes. Special consideration will be paid to the generated triangles to be well shaped and to prevent self interpenetration of the surface. In order to apply the finite difference technique, we adopt the local reparameterization approach proposed by Neuenschwander [10].

An additional approach that uses adaptive refinement for the surface has been presented by Sardajoen *et al*. [11]. In contrast to our goal, they do not use internal energy terms, but just approximate iso-surfaces from scalar data fields. The displacement vectors of the vertices are computed using a Newton iteration scheme with the correction vector projected on the surface normal direction. As a refinement criterion an error estimator based on the face size and the remaining distance to the target surface is used. We will apply an error estimator based on the local curvature of the surface instead.

As a new approach we will present a method that treats not only the differential operators and the solution space in a hierarchical manner, but also the right hand side of the equation. The idea behind this technique is to enlarge the basin of attraction of the boundary structures when the discretization of the deformable surface is still on a coarse level. As we will show in the results this leads to a significant improvement in the convergence process for some data sets.

The deformable surface technique has been applied here to visualize three dimensional scalar data sets. For the visualization of volumetric data sets two main categories of methods are in use: the direct and the indirect visualization methods. The direct methods are mostly based on variations of the volume ray-casting technique (Kaufmann [7]). The indirect visualization methods extract geometric objects from the data set to be visualized. The advantage of indirect visualization methods is the possibility to recover relevant surface manifolds and the ease of display. In consequence the crucial question in indirect volume visualization is how to find meaningful surfaces. Until now isosurfaces have been used mainly for visualization purposes. They are a powerful tool for data sets with smoothly varying function values, a heat flux simulation for instance, but they turn out to be problematic in data sets with large gradients. In this kind of data set the boundary information is usually of interest. Using the iso-surface approach would require the adjustment of an iso-value to obtain a specific boundary, which is not always possible.

The most common iso-surface extraction scheme that is in use today is the marching cubes algorithm (Lorensen *et al*. [9]). Most research in indirect volume visualization has been done in the area of the acceleration of iso-surface extraction and decimation. One popular acceleration technique is to presort the cells according to the value range of the

volume in order to eliminate cells in advance that do not contribute triangles to the requested iso-surface (Wilhelm *et al*. [21] and Shen *et al*. [15]). Another acceleration technique is the decimation of the produced polygons in a postprocessing step as done by Schroeder [12, 13] or the adaptive reduction of the volume data itself to generate fewer polygons more quickly as it was done by Cignoni *et al*. [2] or Grosso *et al*. [5]. The deformable surface approach falls into the category of the indirect visualization methods. By emphasizing boundary structures instead of function values, it provides a different insight into the data sets.

In Section 2 we introduce the mathematical concept of the deformable surface and its discretization. We first explain the continuous formulation of the minimization problem. From this formulation we then derive the discrete representation of the problem. The resulting equation exhibits a linear left hand side and a nonlinear right hand side. In order to solve this system we use a nested iteration scheme that applies a Gauss–Seidel-type iteration to solve the overall system. Each single step within this outer loop is evaluated using a fix-point iteration. This is necessary because of the nonlinear inhomogenity on the right hand side. A methodology to prevent self-intrusion of the surface will also be presented here. Section 3 will cover the construction of a multilevel finite difference solver, including the refinement criterion, the refinement operation, and the efficient construction of appropriate finite difference weights. This section concludes with a description of how to treat the right hand side of the used partial differential equation in a hierarchical way and a description of how to generate the initial model.

In Section 4 we will show a few examples to demonstrate the behavior of the algorithm and the quality of the generated meshes. A medical and a technical application will also be presented there. Finally we analyze the different convergence behavior of the hierarchical method that assumes a scale invariante influence of the right hand side of the equation and the method that also varies the influence of this part of the equation in a hierarchical way.

2. DEFORMABLE SURFACES

Let the volume to be analyzed be described by a function $f: R^3 \to R$ and a surface by $\mathbf{v}: \Omega \subset R^2 \to R^3$. The surface $\mathbf{v}$ has to minimize the following functional (Terzopoulos [17]),

$$\int_\Omega \sum_{i=1}^{3} \left(\tau \left(\nabla \mathbf{v}^{(i)}\right)^2 + (1-\tau)\left(\left(\Delta \mathbf{v}^{(i)}\right)^2 - 2H\left(\mathbf{v}^{(i)}\right)\right)\right) + P(\mathbf{v})\, dA \to 0,$$

where $\mathbf{v}^{(i)}$ denotes the ith component of $\mathbf{v}$, $H(\mathbf{v}^{(i)})$ the determinant of the Hessian matrix of $\mathbf{v}^{(i)}$, and $P: R^3 \to R$ the potential field induced by the volume data f. The $\nabla \mathbf{v}$ term denotes a membrane term that tends to minimize the surface area, which simulates the behavior of a rubber skin. The term $((\Delta \mathbf{v}^{(i)})^2 - 2H(\mathbf{v}^{(i)})$ denotes the total curvature and simulates a thin plate. The coefficient τ is a balancing factor which controls the relative influence of the rubber skin and the thin plate aspect. Both terms stabilize the optimization process. The external energy term P is defined as

$$P(\mathbf{v}) = -(w_{edge} \|\nabla (G_\sigma * f(\mathbf{v}))\| + w_{image} f(\mathbf{v})), \tag{1}$$

where G_σ denotes a Gaussian kernel with variance σ. The kernel is used to generate a smooth potential field from the data, which will improve the convergence of the iterative

solver. This makes the approximation of the gradient by finite differences more reliable and enlarges the regions of attraction near sharp boundaries in the volume data set. The second part of the potential field term may be used to detect regions with high intensity values. The coefficient w_{edge} weights the impact of the boundary influence and the coefficient w_{image} describes the direct influence of the intensity value of the data set.

In order to solve the described problem we derive the corresponding Euler–Lagrange differential equation

$$\tau \Delta \mathbf{v} - (1 - \tau)\Delta^2 \mathbf{v} = \frac{\partial P}{\partial \mathbf{v}}, \tag{2}$$

where we combined the three equations corresponding to the partial derivatives with respect to the components of $\mathbf{v}$. This differential equation lacks a well-defined solution in the absence of boundary conditions. In our approach we use boundary conditions at singular points in the beginning of the iteration process. Consequently we do not get a classical but just a weak solution to this problem (see Neunschwander [10]).

We use a finite difference method to solve this weakly nonlinear differential equation system. The iteration matrix itself would be linear, but we have the nonlinear inhomogeneity P. The values $\mathbf{v}$ are the degrees of freedom at a finite set of given vertices and have to be computed by the iteration process.

In order to approximate this system of differential equations, we need a discrete representation of the differential operators in terms of divided differences. We are making use of the Umbrella function U to achieve this aim (Kobbelt [8], Neuenschwander [10]). The Umbrella function U is defined as follows,

$$U(\mathbf{p}) := \frac{1}{n} \sum_i \mathbf{q}_i - \mathbf{p}, \tag{3}$$

where $\mathbf{p}$ contains the coordinates of the considered vertex and $\mathbf{q}_i$ are the neighbors in the triangular mesh we use to represent the surface with n being the total number of neighbors (valence of $\mathbf{p}$). The Umbrella function has been constructed directly from the difference star shown in Fig. 1. The Umbrella of $\mathbf{p}$ is a discrete approximation of the Laplacian if we assume a symmetric parameterization of the neighborhood of $\mathbf{p}$, i.e., we will associate the adjacent vertex $\mathbf{q}_i$ with

$$(u_i, v_i) = \left(h \cos\left(\frac{2\pi i}{n}\right), h \sin\left(\frac{2\pi i}{n}\right) \right). \tag{4}$$

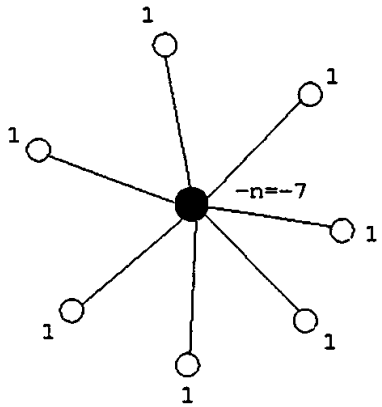

FIG. 1. Example difference star for the discrete Laplacian operator.

An approximation of the Δ^2 operator can be computed by the recursive application of the Umbrella operator

$$U^2(\mathbf{p}) := \frac{1}{n}\sum_i U(\mathbf{q_i}) - U(\mathbf{p}). \tag{5}$$

We assume the external force to be constant and compute the correction vector $\mathbf{c}$ for every vertex by iteration. This way one approximates a solution for the discrete version of (2)

$$(\tau U - (1-\tau)U^2)(\mathbf{p}_k + \mathbf{c}) = \nabla P, \tag{6}$$

where $\mathbf{p}_k$ corresponds to the kth row of the equation system. As the following equations hold,

$$U(\mathbf{p} + \mathbf{c}_1) = U(\mathbf{p}) - \mathbf{c}_1$$
$$U^2(\mathbf{p} + \mathbf{c}_2) = U^2(\mathbf{p}) + \alpha\mathbf{c}_2,$$

with

$$\alpha = 1 + \frac{1}{n}\sum \frac{1}{n_i} \tag{7}$$

and n_i the valence of the ith neighbor of $\mathbf{q}_i$, we can solve Eq. (6) for $\mathbf{c}$ and get the following representation,

$$\mathbf{c} = \gamma(\tau\mathbf{c}_1 + (1-\tau)\alpha\mathbf{c}_2 - \nabla P), \tag{8}$$

where the correction vectors $\mathbf{c}_1, \mathbf{c}_2$ are defined as follows:

$$\mathbf{c}_1 = U(\mathbf{p}) \tag{9}$$
$$\mathbf{c}_2 = -\frac{1}{\alpha}(U^2(\mathbf{p})). \tag{10}$$

The coefficient γ is a damping factor that has to be chosen small enough to guarantee for the convergence of the iteration procedure (Fix-point theory of Banach). The Lipschitz continuity coefficient of the iterative function has to be smaller than one to guarantee for the existence of the fix-point. Along with the existence of the fix-point the theory also guarantees that this point might be estimated by an iteration.

The damping coefficient γ is also frequently called a viscosity term, which is a more physical interpretation. The coefficient γ also contains a factor that is somehow dependent on the valence of the regarded node. This coefficient, however, is not computed in the program as this costs extra computational time and does not influence the final result in a noticeable way. The explanation of this phenomenon is that almost every vertex has a valence of six in this application.

The correction vectors are equivalent to the solutions that satisfy the following equations:

$$U(\mathbf{p} + \mathbf{c}_1) = 0 \tag{11}$$
$$U^2(\mathbf{p} + \mathbf{c}_2) = 0. \tag{12}$$

These equations are direct consequences of the combination of the equations (7), (7), (9), and (10). The corrected position of the vertex $\mathbf{p}$ can then be computed according to

$$\tilde{\mathbf{p}} = \mathbf{p} + \mathbf{c}.$$

The applied iteration method results in fact in a Gauss–Seidel iterator without the need to construct the iteration matrix explicitly. Since (8) gives rise to a fix-point iteration to compute the solution of one row of the system, the approximation procedure results in two nested iteration schemes. The outer loop is the Gauss–Seidel scheme and the inner one is the approximative solution of a nonlinear equation. This contribution is computed using a fix-point iteration scheme.

This fix-point iteration should be repeated several times within every step of the Gauss–Seidel iteration. But in our case, we decided to just perform one single iteration step since the boundary conditions also change during the Gauss–Seidel iteration that encapsulates this fix-point iteration. During each iteration the positions of all vertices are corrected exactly once using the computed correction vector.

2.1. *Preventing Self-intersections*

The definition of the Umbrella functional guarantees for a stable triangulation of the surface if the geometry of the mesh and of the surface to be extracted are already quite similar. The only component that might cause trouble is the impact of the potential function. As the additional impact vector is not necessarily orthogonal to the surface the triangular mesh may be distorted. This problem is illustrated in Fig. 2c.

The first experiments were done using just the component of the potential gradient that points into the direction of the estimated surface normal at this vertex as suggested in

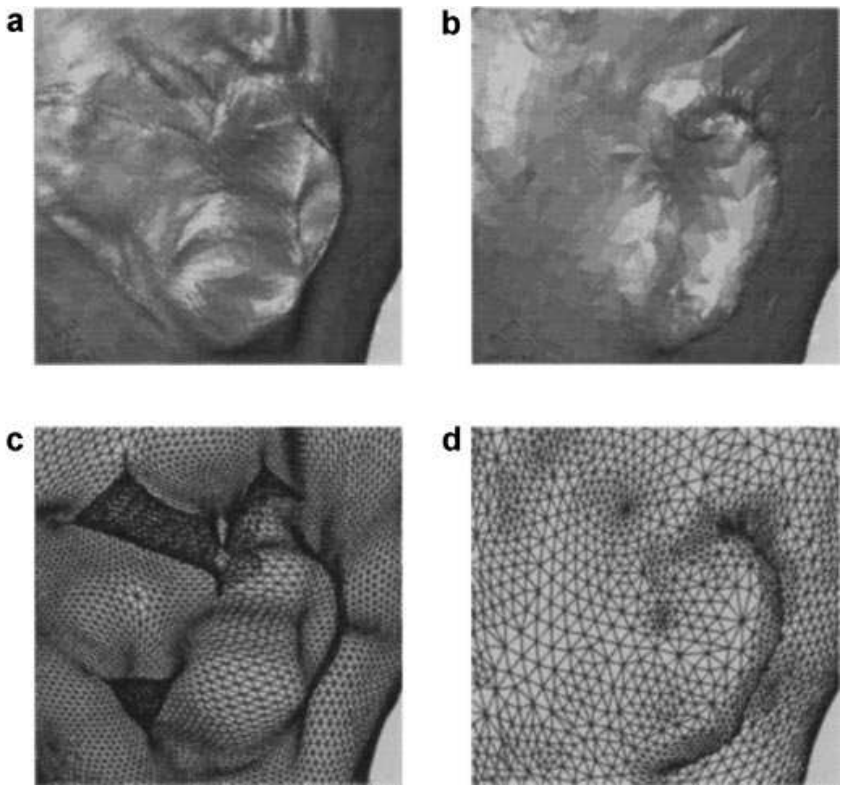

FIG. 2. The problem of self-intersecting surfaces: (a) shaded self-intersecting surface, (b) shaded non-self-intersecting surface, (c) self-intersecting surface, and (d) surface without self-intersection.

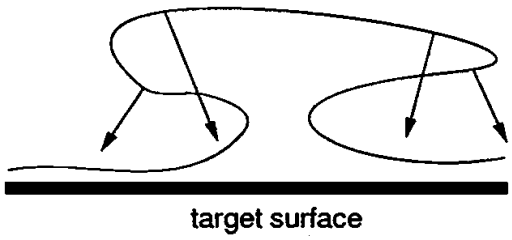

FIG. 3. The problem of using the normal vector direction during the iteration process.

Sardajoen [11]. This approach prevented the surface from becoming distorted when the convergence of the iteration method was already almost achieved. But during this conversion process another problem might occur as shown in Fig. 3.

We suggest an approach which first computes the scalar product of potential gradient with the correction vector that emerges from the Umbrella function. If the result is positive, the direction of this correction vector is used; otherwise the estimated surface normal will designate the direction of correction. By using the Umbrella vector the anomaly as shown in Fig. 3 is avoided. If iteration using this method would continue, the slope shown in the image would be folded down the surface and produce a self-intersection.

On the other hand using the Umbrella vector even if the resulting scalar product is negative existing anomalies would become even stronger and would possibly lead to self-intrusion. The newly computed correction vector would cause an enlargement of the pocket displayed in Fig. 3.

In the case of a negative scalar product of the potential gradient with the correction vector the normal vector of the bordering triangles appeared to be more stable. A resulting triangle mesh for the same example as in Fig. 2c is shown in Fig. 2d. In this image we have already used an adaptive triangulator that will be described in Section 3.

3. HIERARCHICAL APPROXIMATION

The multilevel approach for discretizing partial differential equations has especially become popular in the finite element community the last few years (see Bank [1]). Besides to ability to accelerate the numerical solution, we also want to automate the decision about the discretization granularity. The approach first computes the low frequency contribution of the final solution on the coarse grid. The higher frequency contributions are added later on during the computation on the finer grids.

In this paper we are using the main ideas of the multilevel approach for the generation of the final surface. The first iterations are done on a coarse grid that is then refined at positions indicated by a local error estimator. The initial surface for the iteration process is generated using a semiautomatic modeling tool, which will be described briefly in the next subsection. A special triangulation method avoids T-vertices in the new generated grid. The initial values on the finer grid are computed from the coarser grid using a subdivision scheme. The weights for the finite difference operators (3), (5) are constructed according to the local connectivity.

In order to implement a local refinement strategy a criterion for deciding where to refine the mesh is required. The standard approach used is to construct an error estimator either based on higher order basis functions (p-method) or temporary local refinement (h-method)

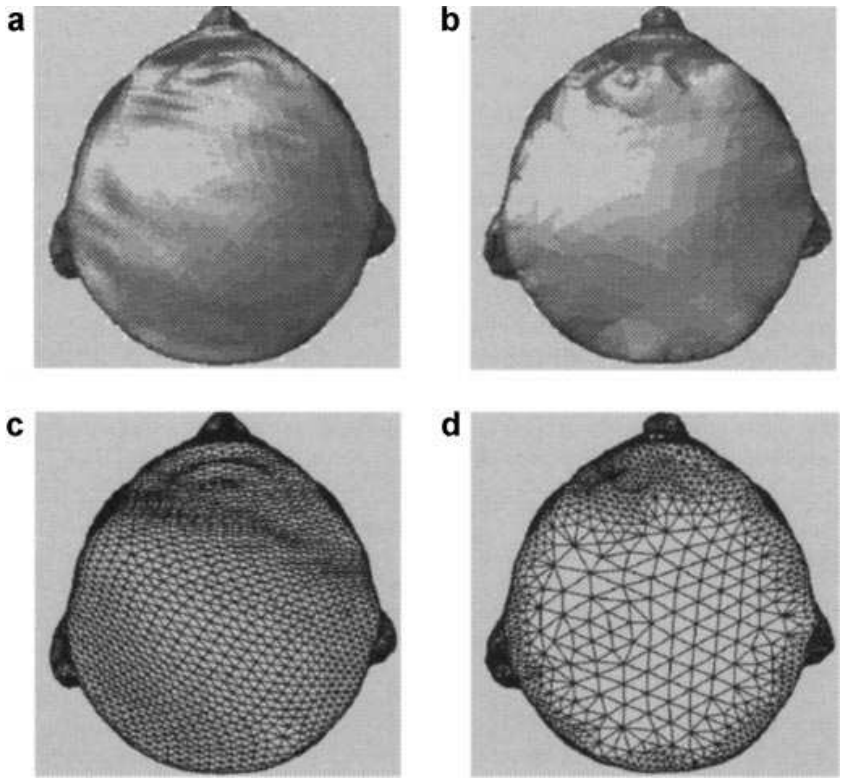

FIG. 4. Uniform vs adaptive mesh refinement: (a) shaded surface of uniform refinement, (b) shaded surface of adaptive refinement, (c) uniform refinement, and (d) adaptive refinement.

(see Verfürth [20]). We have decided to use the influence of the inner forces during the last iteration as some kind of local error estimator to avoid computational overhead. We stop the iteration and initiate mesh refinement if the average correction during the last Gauss–Seidel iteration has become very small. If there is a high impact of the inner forces (local distortion) on a particular vertex during the last iteration, this generally means that there also is a strong contour force implied by the potential field which is approximately the same size and compensates the inner force (actio=reactio). This is an indication that finer detail is present in the neighborhood of that vertex. A triangle is marked for refinement if the average indicator value of its vertices is above a given threshold. In Fig. 4 the result of an adaptively and uniformly refined surface is shown. The refinement concentrates in regions of high curvature.

The local mesh refinement is performed using a red–green triangular, which especially avoids the problem of T-vertices (see Verfürth [20]). This triangulation method consists of two different refinement rules. The red refinement rule subdivides a triangle into four subtriangles. This rule is applied to triangles that have been marked by the local error estimator. The green refinement rule is applied to triangles that are next to the triangles which are red refined. The green refined triangles are divided into two subtriangles to avoid the T-vertex (see Fig. 5). In order to control the aspect ratio of the triangulation green refined triangles must not be refined again. If a green triangle is marked for refinement, the green-cut is undone and a complete red-cut is performed instead. This approach is related to the one described in Vasilescu *et al*. [19].

On the irregularly refined grid, the finite difference scheme has to be adapted, as the underlying parameterization can no longer be assumed to be symmetric (see Eq. (4)). If no special care is taken especially for the vertices connecting a red and green refined triangle, they would tend to drift and would severely distort the adjacent triangles. The weights of a

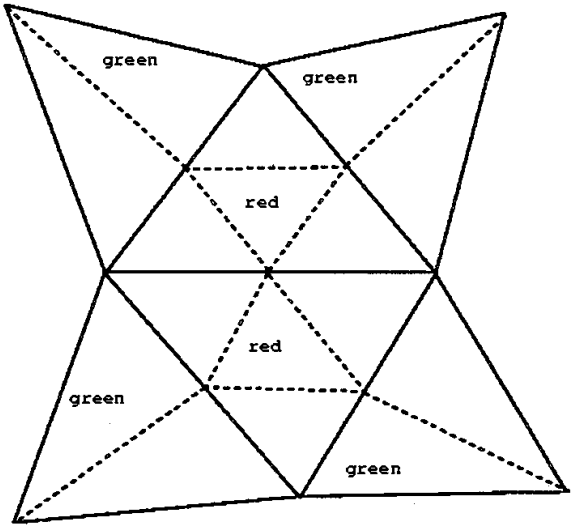

FIG. 5. Red and green refinement.

vertex which is separating a red and a green triangle are therefore adjusted as proposed in Fig. 6a. These weights can be computed by differentiating the interpolating polynomials, as it is traditionally done in the construction of divided difference operators (see Schwarz [14]).

Also special considerations have to be made for the opposite corner of a green refined triangle, to keep the difference scheme balanced. In this case, we have decided to give this former T-vertex a weight of zero (see Fig. 6b). This is necessary to avoid the influence of the finer resolved cell to the coarser one and to keep the difference scheme balanced. In general nonsymmetric Umbrella masks become necessary if the neighboring vertices are not all from the same generation.

The decision which mask of weights to choose during the calculation can be easily done by labeling the vertices. In the initial mesh all vertices are labeled N. If a triangle is refined red, all new vertices are also labeled N. If a triangle is refined green, the newly generated vertex is labeled M, the vertices that are on the same edge of this vertex are labeled C, and the vertex that is on the opposite site of the triangle is labeled D (see Fig. 7).

The resulting weights can then be determined using the lookup table 1. This table allows the correct handling of most cases. For the sake of efficiency, we ignore some rare special

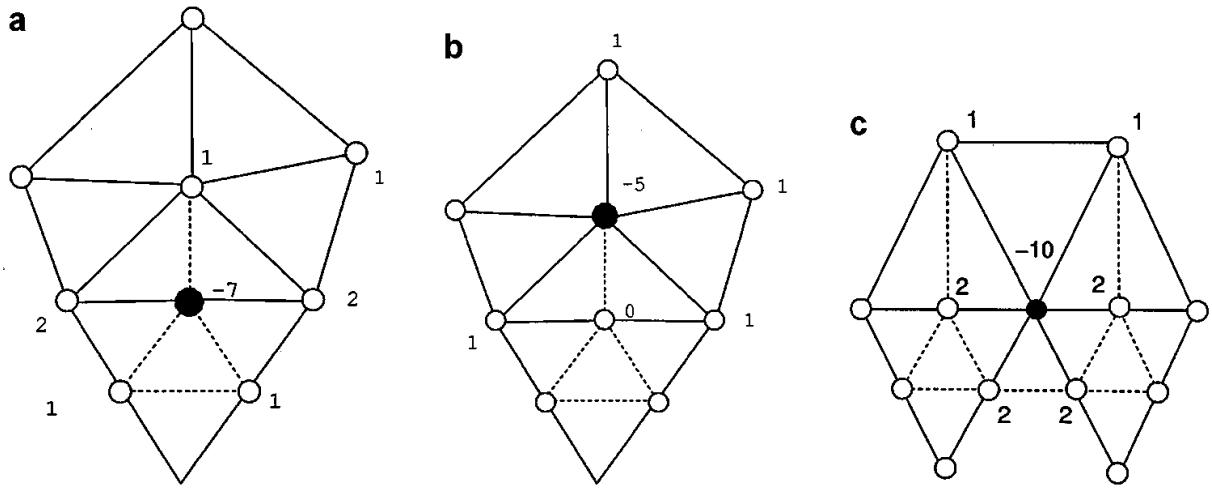

FIG. 6. The differentiation weights in the case of the irregularly refined cells: (a) differentiation weights for type M vertex, (b) differentiation weights for type D vertex, and (c) differentiation weights for type C vertex.

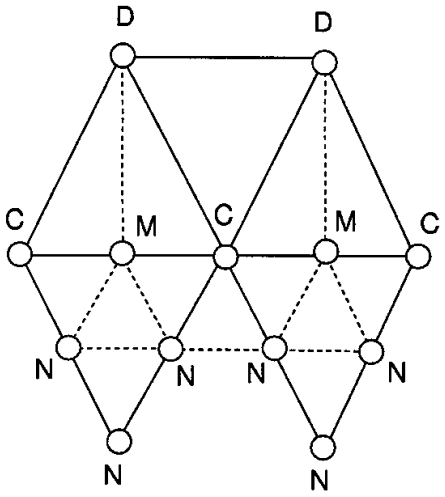

FIG. 7. Labelization of the vertices.

cases of cascading refinement level boundaries. This does not affect the resulting mesh significantly.

In consequence of this generalization of the Umbrella functional the Eq. (3), which represents the Umbrella functional, now becomes the nonuniform Umbrella

$$U(\mathbf{p}) = \frac{1}{\sum_i w_i} \sum_i w_i \mathbf{q_i} - \mathbf{p},$$

where w_i are the weights according to the modified difference schemes. Equations (5) and (7) change accordingly.

To reflect the different triangle sizes in the parameterization domain, triangles from different subdivision levels should also have different differentiation weights. If one analyzes the Gauss–Seidel iteration scheme, one finds out that ignoring this subject effectively results in an overemphasis of the volume gradient influence. However, as the smaller triangles are assumed to fit to regions of higher curvature this is justified.

The position of the new vertices is inserted according to the subdivision scheme shown in Fig. 8, which has been proposed in Dyn *et al*. [4]. This scheme tends to generate a smooth surface. The butterfly subdivision step is applied as a prolongation operator that generally generates better initial values than simple linear interpolation. The final vertex position is computed by the iterative solver.

3.1. *Hierarchical Analysis of the Volume Gradient*

In order to optimize the convergence behavior of the surface in some cases, the analysis of the gradient volume is also done in a hierarchical manner. The basic idea of

TABLE 1
Lookup Table for the Interpolation Coefficients

←	*N*	*C*	*M*	*D*
N	1	1	1	1
C	2	2	2	1
M	1	2	0	1
D	1	1	0	1

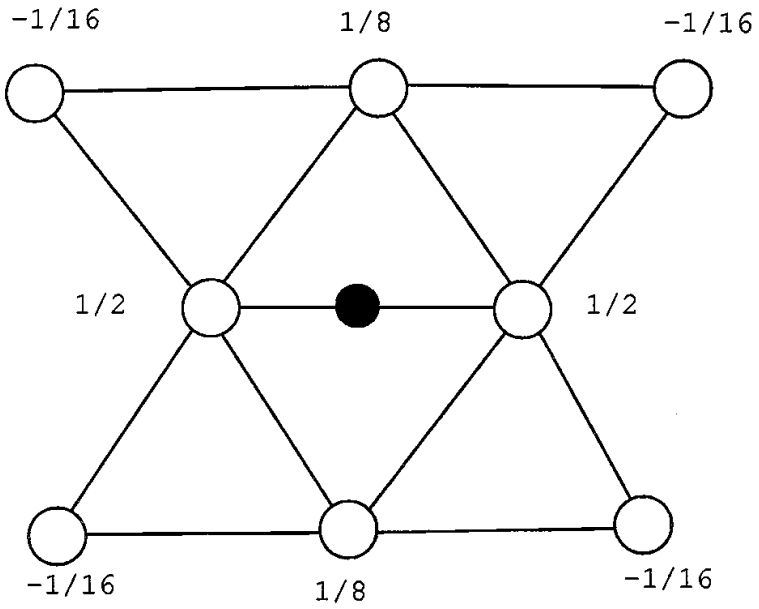

FIG. 8. Butterfly subdivision scheme: ○, old vertex; ●, new vertex.

multilevel methods is that the solution at the coarse resolution level reflects the low frequency component of the final solution. The finer frequency parts are added later on during the subdivision process.

Due to the fact that only low frequency information is reflected in the solution at a coarser level it makes sense to increase the variance in the convolution kernel in Eq. (1). The size of the convolution kernel is also increased to maintain numerical accuracy.

Figure 9 shows a sketch of the situation with a sharp boundary structure and several gradient magnitudes estimated with different convolution kernels of increasing variance. One can clearly see that the basin of attraction of the boundary structure enlarges with increasing variance. On the other hand, the detail information in the data set gets lost as the variance and the size of the convolution kernel is increased. The low pass filtering effect of the Gaussian suppresses the high frequency detail information and thus also enlarges the basin of attraction of boundary structures.

The variance and the support of the convolution kernel is maximal when the iteration process begins and is bisected at each subdivision step. During the first iterations a rough

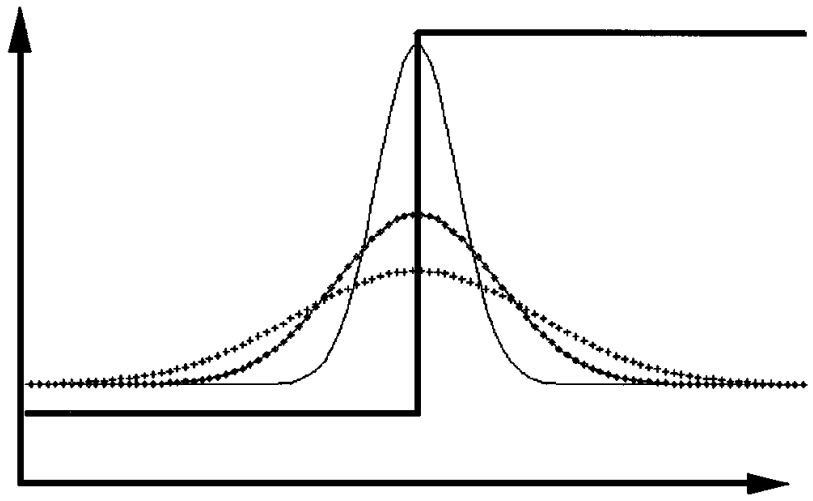

FIG. 9. Application of different variances: ——, original function; ——, gradient magnitude low variance; ◆◆◆, gradient magnitude middle variance; +++, gradient magnitude high variance.

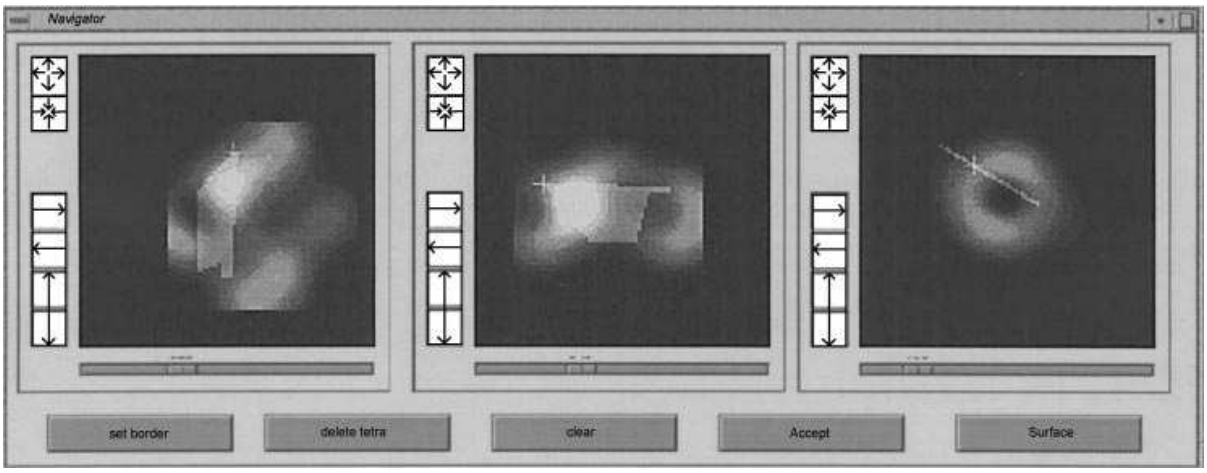

FIG. 10. The user interface of the slicing tool.

positioning of the initial surface is accomplished and during further iterations an exact positioning with smaller variances and more high frequency information of the volume can be performed. The reduction of the variance and of the size of the convolution kernel by 50% steps has been chosen, as the length of a triangle edge is also bisected during this process.

The initial size and the variance for the convolution kernel is estimated by the spatial extent of the volume. We choose an initial variance and support as $1/6$ of the average extension of the data set in every dimension. This value of $1/6$ should be adjusted to the content of the data set.

In order to keep the complexity of the computation of the convolution at a reasonable level, the gradient is estimated using one dimensional filters only. This is less stable than the application of three dimensional filters, but also much less computing intensive, as the amount of operations needed also just increases linearly with the extension of the convolution kernel.

3.2. *Generation of the Initial Surface*

In order to apply the deformable surface algorithm, an initial surface with a coarse triangulation has to be defined first. This surface has to be in the neighborhood of the final destination surface. First the user selects some boundary points using a slicing tool that is shown in Fig. 10. The selected vertices are then connected using a Delaunay tetrahedrization. In order to model nonconvex structures, tetrahedra may be deleted by the user in a post-processing step. The deletion of tetrahedra is done again by picking into the slicing view to delete individual tetrahedra. The positions of the tetrahedra are indicated by color. This technique is related to the modeler presented in Neuenschwander [10], except for the fact that Neuenschwander deletes the tetrahedra in the three dimensional surface view without the ability to control the correctness in the slicing view. If the user is finally satisfied with the result, the outer surface of the tetrahedral complex is used as an initial surface.

4. RESULTS

We have applied the deformable surface algorithm to medical and technical data sets as shown in Fig. 11. By modeling different initial meshes different structures may be extracted

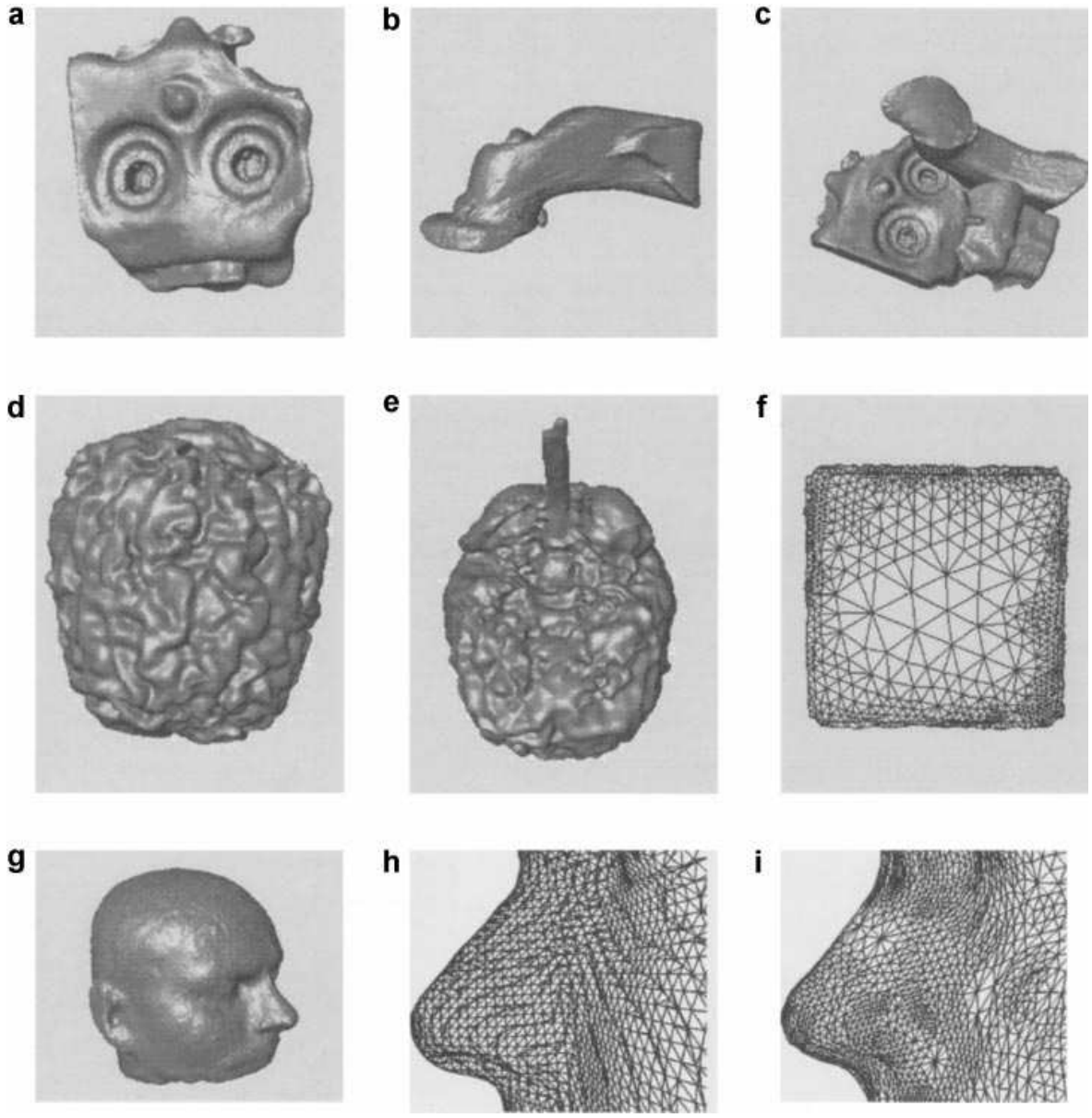

FIG. 11. Applications of the energy minimizing surface algorithm: (a) engine block, (b) exhaust of engine, (c) engine configuration, (d) brain of MRI scan, (e) brain seen from below, (f) adaptive triangulation, (g) head of MRI scan, (h) marching cubes mesh, and (h) energy minimizing surface mesh.

from the same data set. The three parts of the engine block are extracted from the same data set. The decision which structure has to be visualized is done by editing the initial surface appropriately. The surface tends to drift toward the closest boundary with respect to its starting position.

We have implemented this algorithm in C++ using the Open Inventor library as a means of displaying the generated surfaces. This provides the possibility to tune the parameters during the iteration process and the possibility to generate an Inventor file description of the generated surface that may be used later on.

As a performance result we have measured 1.4 s for an iteration with 12288 triangles and 5.9 s for a surface with 49152 triangles. These times have been measures on a 175 MHz R10000 SGI O2. It is difficult to give an estimation for the overall construction of a complete surface as the computation time is influenced by the requested quality. The finer the subdivision, the more time is required. The overall time also depends on the quality of the surface to be approximated, as different surfaces require a different amount of iterations. As

a rule of thumb it can said that about 40 iterations are appropriate between two subdivision steps. In lower resolutions there are fewer and in higher resolutions there are more iterations necessary as finer details are added to the surface. The segmentation of the brain took about half an hour including user-interaction to adopt the parameters of the iterative solver during the computation. Processing the engine block took about 20 min. The required time is not comparable to the performance of the marching cubes algorithm and its variations; however, in general these surfaces cannot be extracted by an iso-surface algorithm. This is especially true for the extraction of the brain surface of the MRI scan. An iso-surface extraction algorithm like marching cubes generates surfaces that indicate a distinct value within the data set and does not represent boundary information, which is usually of higher interest in data sets that represent certain objects. The intensity values may differ along a boundary in the algorithm presented here, which does not influence the appearance as the iso-surface approach would. In contrast to the iso-surface approach our algorithm tends to generate smooth surfaces which are tolerant to small disturbances in the analyzed data set. An iso-surface extraction algorithm is not able to isolate single connected objects as our algorithm does.

The last two images show a comparison of the generated triangular mesh of the marching cubes algorithm and the energy minimization approach. The iso-value has been adjusted to show the skin surface of the MRI scan. For the deformable surface approach, the initial surface has been wrapped around the head. Our algorithm generates a much more regular triangulation than the marching cubes does, which makes this technique especially interesting for texture mapping and numerical applications. In Fig. 11f an adaptive triangulation of a cube side is shown. This image clearly shows the stability of the triangulation, especially at the borders of red and green refined triangles.

As a rule of the thumb we figured out that the damping factor γ should be about 0.05–0.1. This damping factor influences the step size during the iteration. A smaller value reduces the convergence rate. If the value is too high the iteration scheme behaves unstably, as the Lipschitz constant of the iteration scheme is no longer smaller than one. The factor τ should be around 0.05–0.2, but the algorithm does not behave sensitively with respect to changes of the parameters. If the factor τ is zero, then the surface simulates a pure thin plate that tends to minimize surface curvature. If τ is one, this approach simulates a pure rubber skin that tends to minimize the area of the surface. Any other value simulates a mixture of these two aspects.

In Fig. 12 we have shown a comparison of the convergence process without and with the adaption of the right hand side of the equation. In this example we have applied the cubic data set that has a value of one in a small cubic area in the center and a value of zero outside this region. As an initial surface for the convergence process a cube surface is defined that encloses the whole extent of the data set. This surface is shown in image 12a.

The image series 12b–12d shows the convergence process of the described surface with a nonchanging variance and a constant size of convolution kernel. The first image shows the surface after two subdivision steps, the second one shows the image after three steps, and the last one after four. The problem of the relatively small convolution kernel and the low variance becomes obvious in this series of images. The surface shrinks very slowly at the beginning of the iteration process and finally starts locking onto the boundary structure of the cube. This can be seen at the singularity in image 12d. This locking process happens quite late and it needs still many more iterations to correctly approximate the boundary structure under discussion.

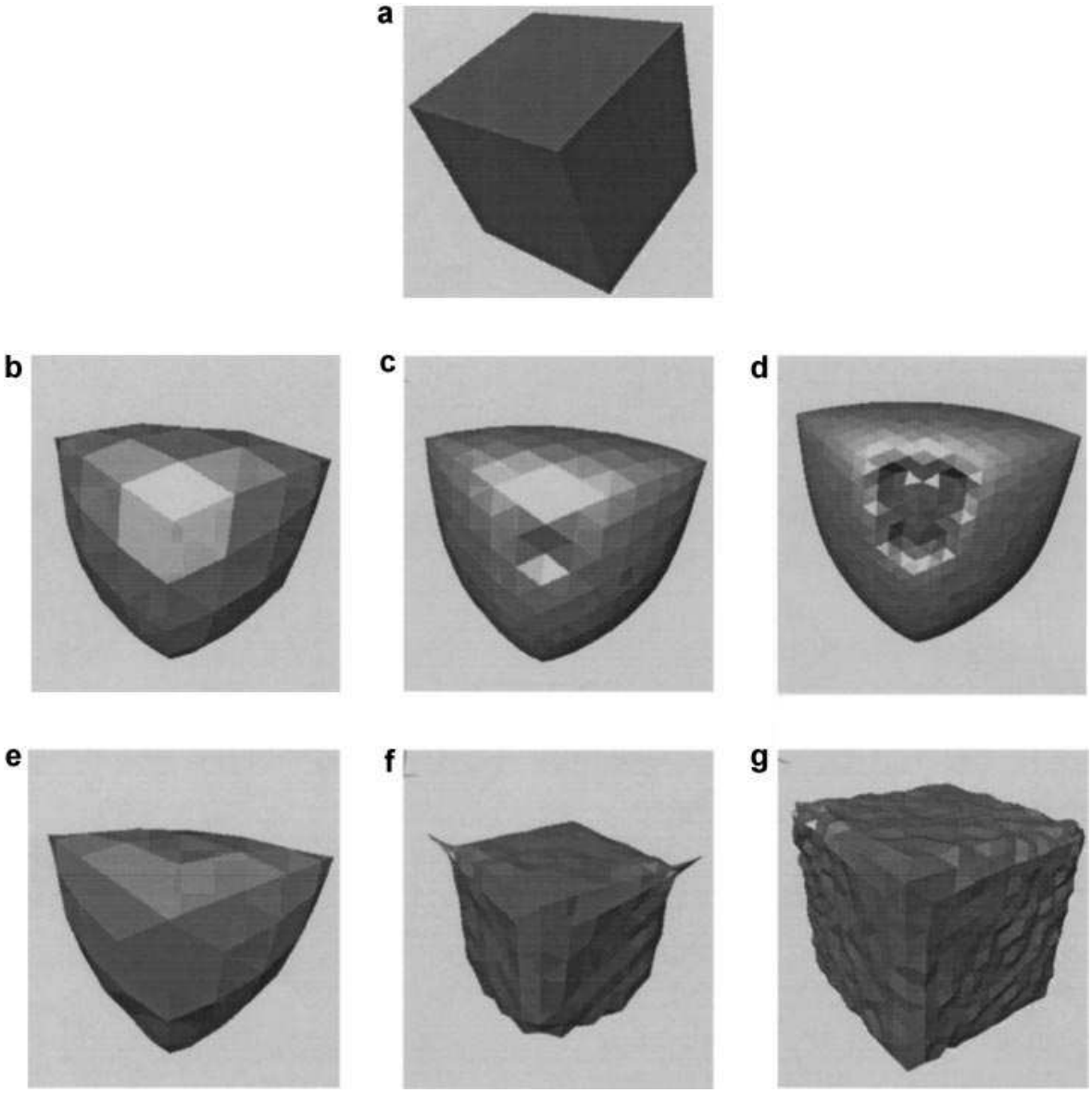

FIG. 12. Comparison of the two adaptive techniques (scaling increasing from left to right): (a) start surface, (b) two subdivisions with constant variance, (c) three subdivisions with constant variance, (d) four subdivisions with constant variance, (e) two subdivisions with adaptive variance, (f) three subdivisions with adaptive variance, and (g) four subdivisions with adaptive variance.

The second series of images 12e–12g shows the convergence process with the successively reduced variance and the size of the convolution kernel. One can clearly see that the shape of the cube is approximated much faster than in the process, where the right hand side of the equation is not adopted.

The presented data set is particularly suited for the adaptive change of the convolution kernel and the variance, as there is just one sharp singularity in this data set. The larger variance at the beginning of the iteration process attracts the surface to the boundary structure. The convolution kernel generates gradient information in the homogeneous regions of the data set especially at the beginning of the iteration process. This is not the case for the smaller variance and convolution kernels in the first series of images. In this case the initial shrinking process is purely based on the internal energy terms.

However not all data sets are suited for this kind of hierarchical method. This is especially true for data sets where local minima of the overall energy function are searched for. If there is a weak boundary structure rather close to a strong one, it can only be extracted using

the nonadaptive technique for the right hand side. If the initial surface is positioned rather close to this boundary structure it is attracted by it if the size of the convolution kernel and the variance is kept minimal from the beginning. But if the hierarchical approach for the inhomogeneity would be used instead, the surface would become attracted by the stronger boundary structure which is farther away than the weaker one. This is caused by the lower pass filter that is applied here.

5. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a new indirect volume visualization method that is based on the deformable surface approach. We have described how to construct an adaptive multilevel finite difference solver and demonstrated the applicability for medical and technical data sets. This visualization method is a general approach suitable for a great variety of scalar data sets that contain boundaries.

As future work we plan to use the generated surfaces for texture mapping and parameterization purposes. First experiments have also shown the ease of converting the Inventor description to a VRML description, which might offer the possibility of Web-based applications for this method.

REFERENCES

1. R. E. Bank, Hierarchical bases and the finite element method, *Acta Numer.* 1996.
2. P. Cignoni, L. De Floriani, C. Montani, E. Puppo, and R. Scopigno, Multiresolution modeling and visualization of volume data based on simplicial complexes, in *IEEE '94 Symposion on Volume Visualization, 1994*, pp. 19–26.
3. L. D. Cohen and I. Cohen, Finite element methods for active contour models and balloons for 2-D and 3-D images, *IEEE Trans. Pattern Anal. Mach. Intelligence* **15**, 1993, 1131–1147.
4. N. Dyn, J. Gregory, and D. Levin, A butterfly subdivision scheme for surface interpolation with tension control, *ACM Trans. Graph.* 1990, 160–169.
5. R. Grosso, C. Lürig, and T. Ertl, The multilevel finite element method for adaptive mesh optimization and visualization for volume data, in *Proceedings IEEE Visualization '97, Phoenix AZ, October 1997* (R. Yagel and H. Hagen, Eds.), pp. 387–394.
6. M. Kass, A. Witkin, and D. Terzopoulus, Snakes: Active contour models, *Internat. J. Comput. Vision* 1988, 321–331.
7. A. Kaufmann, *Introduction to Volume Visualization*, IEEE Computer Society Press, Los Alamitos, CA, 1991.
8. L. Kobbelt, *Iterative Erzeugung glatter Interpolanten*, Ph.D. thesis, Universität Karlsruhe, 1994.
9. W. Lorensen and H. Cline, Marching cubes: A high resolution 3D surface construction algorithm, *Comput. Graphics* **21**(4), 1987, 163–169.
10. W. M. Neuenschwander, *Elastic Deformable Contour and Surface Models for 2-D and 3-D Image Segmentation*, Ph.D. thesis, Zürich, Eidgenössische Techn. Hochschule, Dissertation, 1995.
11. I. A. Sardajoen and F. H. Post, Deformable surface techniques for field visualization, in *Eurographics '97, 1997*, pp. 109–116.
12. W. Schroeder, J. A. Zarge, and W. E. Lorensen, Decimation of triangle meshes, in *Proc. SIGGRAPH '92, 1992*, pp. 65–70.
13. W. J. Schroeder, A topology modifying progressive decimation algorithm, in *Proceedings IEEE Visualization '97, 1997*, (R. Yagel and H. Hagen, Eds.), pp. 205–219.
14. H. R. Schwarz, *Numerische Mathematik*, Teubner, Leipzig, 1993.
15. H.-W. Shen, C. Hansen, Y. Livnat, and C. R. Johnson, Isosurfacing in span space with utmost efficiency (ISSUE), in *Proceedings IEEE Visualization '96 1996*, pp. 287–294.

16. J. W. Snell, M. B. Merickel, J. M. Ortega, J. Goble, J. R. Brookeman, and N. F. Kassell, Model-based boundary estimation of complex objects using hierachical active surface templates, *Pattern Recog.* **28**, 1995, 1599–1609.
17. D. Terzopoulos. Regularization of inverse visual problems involving discontinuities, *IEEE Trans. Pattern Anal. Mach. Intelligence* 1986, 413–424.
18. D. Terzopoulos, A. Witkin, and M. Kass, Symmetry-seeking models and 3D object construction, *Internat. J. Comput. Vision* **1**, 1987, 211–221.
19. M. Vasilescu and D. Terzopoulos, Adaptive meshes and shells: Irregular triangulation, discontinuities, and hierarechical subdivision, in *Proceedings of Computer Vision and Pattern Recognition Conference, 1992*, pp. 829–832.
20. R. Verfürth, *A review of a Posteriori Error Estimation and Adaptive Mesh Refinement Techniques*, Wiley–Teubner, Berlin, 1996.
21. J. Wilhelms and A. V. Gelder, Octrees for faster iso-surface generation, in *ACM Trans. Graphics*, 1992, 201–227.