

# Deformable Surfaces for Feature Based Indirect Volume Rendering

Christoph Lürig   Leif Kobbelt   Thomas Ertl  
University of Erlangen, Computer Graphics Group (IMMD IX)  
Am Weichselgarten 9, D-91058 Erlangen, Germany  
Email: {cpluerig,kobbelt,ertl} @immd9.informatik.uni-erlangen.de

## Abstract

*In this paper we present an indirect volume visualization method, based on the deformable surface model, which is a three dimensional extension of the snake segmentation method. In contrast to classical indirect volume visualization methods, this model is not based on iso-values but on boundary information. Physically speaking it simulates a combination of a thin plate and a rubber skin, that is influenced by forces implied by feature information extracted from the given data set. The approach proves to be appropriate for data sets that represent a collection of objects separated by distinct boundaries. These kind of data sets often occur in medical and technical tomography, as we will demonstrate by a few examples.*

*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. This functional tends to produce very regular triangular meshes compared to results of the marching cubes algorithm. It makes this method attractive for meshing in numerical simulation or texture mapping. Red-green triangulation allows an adaptive refinement of the mesh. Special considerations have been made to prevent self inter-penetration of the surfaces.*

## 1. Introduction

Many technical and medical tomographic measurement techniques generate large volumetric scalar valued data sets, that have to be interpreted by means of visualization methods.

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 ad-

vantage 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. Up to now mainly iso-surfaces have been used 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 these kind of data sets the boundary information is usually of interest. Using the iso-surface approach would require to adjust 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 (Lorenson 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 post processing 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].

In contrast to the iso-surface approach, we are looking for surfaces with different properties in this paper. Our aim is to look for surfaces that match the boundaries of a sub-volume. Those are indicated by large gradient magnitude. This contour model has been developed in the pattern recognition community and is known as the snake concept for the two dimensional case (Kass et al. [6]). The snake is a curve that minimizes a potential energy, which consists of an internal and an external part. The external part is the negative of the gradient magnitude. 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 in 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 triangular grid 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 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, and 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 use an error estimator based on the local curvature of the surface instead.

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 has a linear left hand side and a non-linear 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 during this outer loop is evaluated using a fix-point iteration. This is necessary because of the non-linear inhomogeneity 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 multi-level finite difference solver, including the refinement criterion, the refinement operation and the efficient construction of appropriate finite difference weights. 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.

## 2. Deformable Surfaces

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

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

where  $\mathbf{v}^{(i)}$  denotes the  $i$ -th component of  $\mathbf{v}$ ,  $H(\mathbf{v}^{(i)})$  the determinant of the Hessian Matrix of  $\mathbf{v}^{(i)}$  and  $P : R^3 \rightarrow R$  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  $\tau$  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:

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

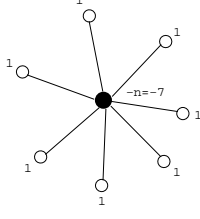
where  $G_{\sigma}$  denotes a Gaussian kernel with variance  $\sigma$ . The kernel is used to generate a smooth potential field out of the data, which will improve the convergence of the iterative solver. This makes the approximation of the gradient by finite differences more reliable and enlarges the 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

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

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 Neuenschwander [10]).

We use a finite difference method to solve this differential equation system. The discretization results in a weakly non-linear equation system. The iteration matrix itself would be linear, but we have the non-linear inhomogeneity  $P$ . The values  $\mathbf{v}$  are the degrees of freedom at a finite set



**Figure 1. Example difference star for the discrete Laplacian operator**

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:

$$U(\mathbf{p}) := \frac{1}{n} \sum_i \mathbf{q}_i - \mathbf{p}, \quad (2)$$

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

$$(u_i, v_i) = \left( h \cos\left(\frac{2\pi i}{n}\right), h \sin\left(\frac{2\pi i}{n}\right) \right) \quad (3)$$

An approximation of the  $\Delta^2$  Operator can be computed by the recursive application of the Umbrella operator

$$U^2(\mathbf{p}) := \frac{1}{n} \sum_i U(\mathbf{q}_i) - U(\mathbf{p}) \quad (4)$$

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 (1).

$$(\tau U - (1 - \tau)U^2)(\mathbf{p}_k + \mathbf{c}) = \nabla P, \quad (5)$$

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

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

with

$$\alpha = 1 + \frac{1}{n} \sum \frac{1}{n_i} \quad (6)$$

and  $n_i$  the valence of the  $i$ -th neighbor of  $\mathbf{q}_i$ , we can solve equation (5) for  $\mathbf{c}$  and get the following representation

$$\mathbf{c} = \gamma(\tau \mathbf{c}_1 + (1 - \tau)\alpha \mathbf{c}_2 - \nabla P), \quad (7)$$

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

$$\mathbf{c}_1 = U(\mathbf{p}) \quad (8)$$

$$\mathbf{c}_2 = -\frac{1}{\alpha}(U^2(\mathbf{p})) \quad (9)$$

The coefficient  $\gamma$  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 damping coefficient  $\gamma$  is also frequently called a viscosity term, which is a more physical interpretation.

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

$$U(\mathbf{p} + \mathbf{c}_1) = 0 \quad (10)$$

$$U^2(\mathbf{p} + \mathbf{c}_2) = 0 \quad (11)$$

The corrected position of the vertex  $\mathbf{p}$  can then be computed according to:

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

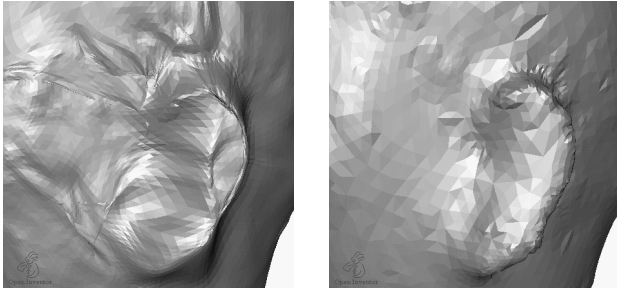
The applied iteration method results in fact in a Gauss-Seidel iterator without the need to construct the iteration matrix explicitly. Since (7) gives rise to a fix-point iteration to compute the solution of one row of the system, the approximation procedure results in two nested iteration schemes. The outer loop is the Gauss-Seidel scheme and the inner one is the approximative solution of a non-linear 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.

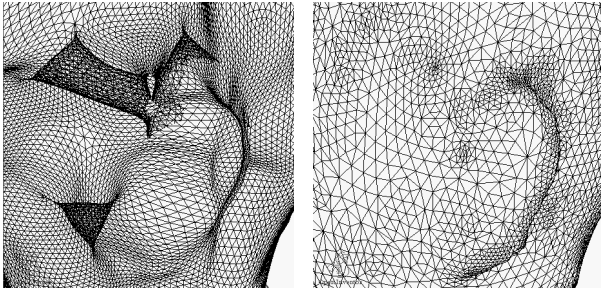
## 2.1. Preventing self-intersections

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



(a) Shaded self intersecting surface

(b) Shaded non self intersecting surface



(c) Self intersecting surface

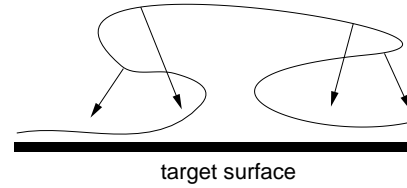
(d) Surface without self intersection

**Figure 2. The problem of self intersecting surfaces**

vector is not necessarily orthogonal to the surface the triangular mesh may be distorted. This problem is illustrated in Fig. 2(c).

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 (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 a another problem might occur as shown in Fig. 3.

We suggest an approach which first computes the scalar product of the 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, 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, in this case an av-



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

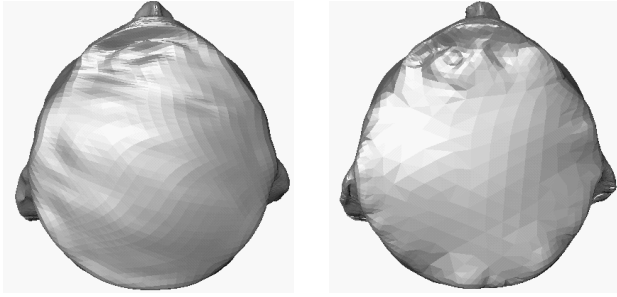
eraged normal vector of the bordering triangles appeared to be more stable. A resulting triangle mesh for the same example as in Fig. 2(c) is shown in Fig. 2(d). In this image we have already used an adaptive triangulator, that will be described in section 3.

### 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 the ability to accelerate 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.

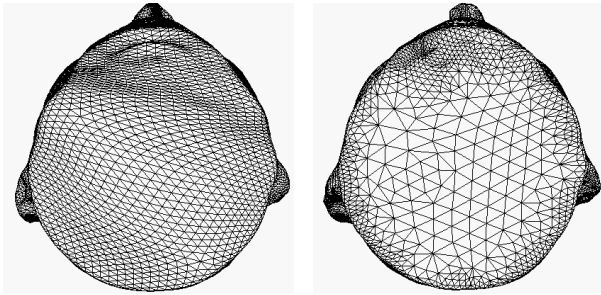
In this paper we are using the main ideas of the multi level approach for the generation of the final surface. The first iterations are done on a coarse grid, that is then refined at positions indicated by a local error estimator. The initial surface used for the iteration process is generated using a semi-automatic modeling tool, which will be described briefly in the next sub-section. A special triangulation method is used to avoid 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 (see eqs. 2, 4) 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) (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 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



(a) Shaded surface of uniform refinement

(b) Shaded surface of adaptive refinement



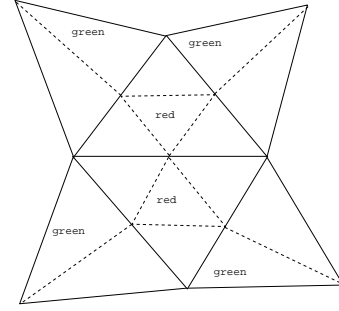
(c) Uniform refinement

(d) Adaptive refinement

**Figure 4. Uniform vs. adaptive mesh refinement**

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 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 triangulator, 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 sub-triangles. 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 sub-triangles 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



**Figure 5. Red and Green refinement**

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. 3). 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 vertex which is separating a red and a green triangle are therefore adjusted as proposed in Fig. 6(a). 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]).

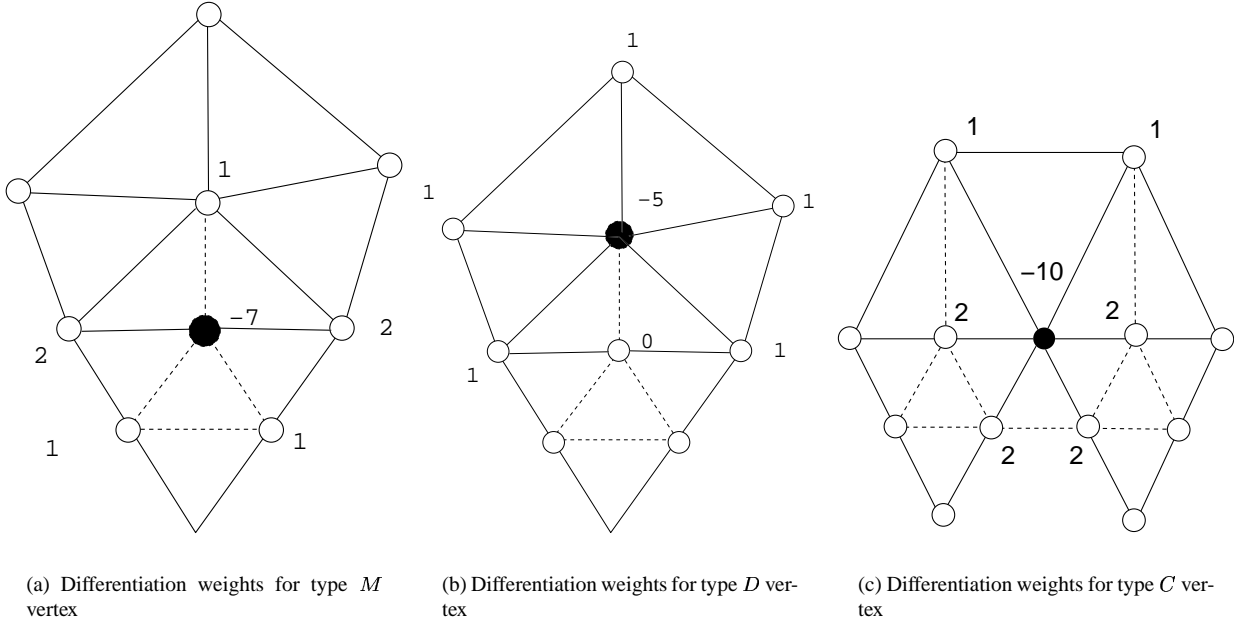
Also special considerations have to be made for the opposite corner of a green refined triangle, to keep the difference scheme balanced. In this case, we have decided to give this former T-vertex a weight of zero (see Fig. 6(b)). This is necessary to avoid the influence of the finer resolved cell to the coarser one and to keep the difference scheme balanced. In general non-symmetric 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).

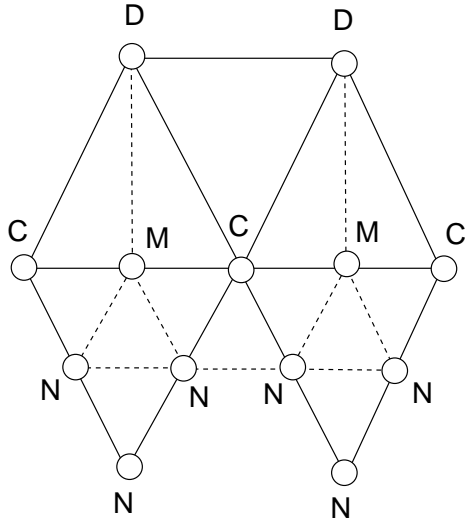
The resulting weights can then be determined using the look up table 1. This table allows to correctly handle most cases. For the sake of efficiency, we ignore some rare special cases of cascading refinement level boundaries. This does not effect the resulting mesh significantly.

In consequence of this generalization of the Umbrella functional the equation (2), which represents the Umbrella functional now becomes the non-uniform Umbrella

$$U(\mathbf{p}) = \frac{1}{\sum_i w_i} \sum_i w_i \mathbf{q}_i - \mathbf{p}$$



**Figure 6. The differentiation weights in the case of the irregular refined cells**



**Figure 7. Labelization of the vertices**

$\leftarrow$	$N$	$C$	$M$	$D$
$N$	1	1	1	1
$C$	2	2	2	1
$M$	1	2	0	1
$D$	1	1	0	1

**Table 1. Look up table for the interpolation coefficients**

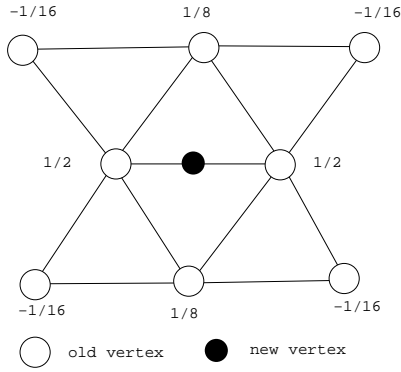
position is computed by the iterative solver.

### 3.1. 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. 9. The selected vertices are then connected using a Delaunay tetrahedrization. In order to model non-convex 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 the fact, that Neuenschwander deletes the tetrahedra in the three dimensional surface view, without the ability

where  $w_i$  are the weights according to the modified difference schemes. Equations (4) and (6) change accordingly.

The position of the new vertices are 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



**Figure 8. Butterfly subdivision scheme**

to control the correctness in the slicing view. If the user is finally satisfied with the result, the outer surface of the tetrahedra 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 10. By modeling different initial meshes different structures may be extracted 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 towards 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 seconds for an iteration with 12288 triangles and 5.9 seconds for a surface with 49152 triangles. These times have been measured 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 is done, 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 a thumb it can be said that about 40 iterations are appropriate between two sub-division 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 minutes. The required time is not comparable to the performance of the marching cubes algorithm and its variations, however in general these surfaces can not 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 the 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 head 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, what makes this technique especially interesting for texture-mapping and numerical applications. In Fig. 10(f) an adaptive triangulation of a cube side is shown. This image clearly shows the stability of the triangulation, especially at the borders of redly and greenly refined triangles.

As a rule of the thumb we figured out, that the damping factor  $\gamma$  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 unstable, as the Lipschitz constant of the iteration scheme is no longer smaller than one. The factor  $\tau$  should be around 0.05 – 0.2, but the algorithm does not behave sensitive with respect to changes of the parameters. If the factor  $\tau$  is zero, then the surface simulates a pure thin plate, that tends to minimize surface curvature. If  $\tau$  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.

## 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 multi-level 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.

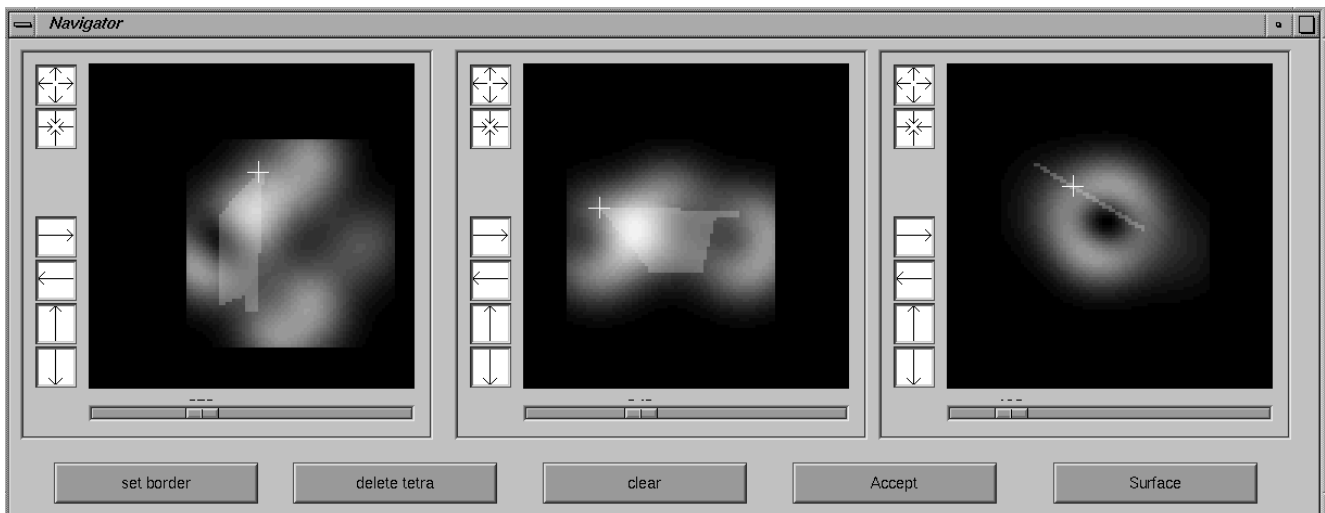
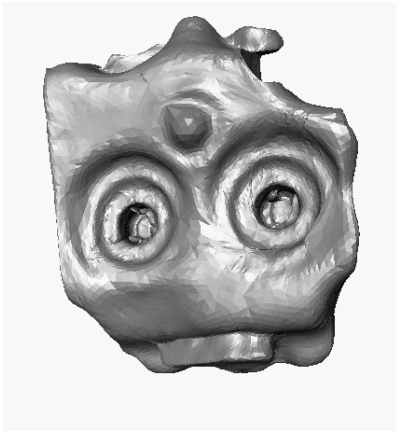


Figure 9. The user interface of the slicing tool

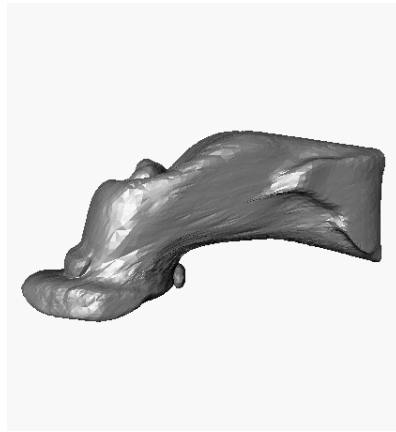
As future work we plan to use the generated surfaces for texture mapping and parameterization purposes. First experiments have also shown the ease of converting the Inventor description to a VRML description, which might offer the possibility of web based applications for this method.

## References

- [1] R. E. Bank. Hierarchical Bases and The Finite Element Method. *Acta Numerica*, 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 Symposium on Volume Visualization*, pages 19–26, 1994.
- [3] L. D. Cohen and I. Cohen. Finite Element Methods for Active Contour Models and Balloons for 2-D and 3-D Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1131–1147, 1993.
- [4] N. Dyn, J. Gregory, and D. Levin. A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control. *ACM Trans. Graph.*, pages 160–169, 1990.
- [5] R. Grosso, C. Lürig, and T. Ertl. The Multilevel Finite Element Method for Adaptive Mesh Optimization and Visualization of Volume Data. In R. Yagel and H. Hagen, editors, *Proceedings IEEE Visualization '97*, pages 387–394, Phoenix AZ, U.S.A., October 1997.
- [6] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, pages 321–331, 1988.
- [7] A. Kaufmann. *Introduction to Volume Visualization*. IEEE Computer Society Press, 1991.
- [8] L. Kobbelt. *Iterative Erzeugung glatter Interpolanten*. PhD thesis, Universität Karlsruhe, 1994.
- [9] W. Lorensen and H. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics*, 21(4):163–169, 1987.
- [10] W. M. Neuenschwander. *Elastic Deformable Contour and Surface Models for 2-D and 3-D Image Segmentation*. PhD thesis, Zürich, Eidgenössische Techn. Hochschule, Dissertation, 1995.
- [11] I. A. Sardajoen and F. H. Post. Deformable Surface Techniques for Field Visualization. In *Eurographics '97*, pages 109–116, 1997.
- [12] W. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. In *Proc. SIGGRAPH '92*, pages 65–70, 1992.
- [13] W. J. Schroeder. A Topology Modifying Progressive Decimation Algorithm. In R. Yagel and H. Hagen, editors, *Proceedings IEEE Visualization '97*, pages 205–219, 1997.
- [14] H. R. Schwarz. *Numerische Mathematik*. Teubner, 1993.
- [15] H.-W. Shen, C. Hansen, Y. Livnat, and C. R. Johnson. Iso-surfacing in span space with utmost efficiency (issue). In *Proceedings IEEE Visualization '96*, pages 287–294, 1996.
- [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 Hierarchical Active Surface Templates. *Pattern Recognition*, 28(10):1599–1609, 1995.
- [17] D. Terzopoulos. Regularization of Inverse Visual Problems Involving Discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 413–424, 1986.
- [18] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-Seeking Models and 3D Object Construction. *International Journal of computer Vision*, 1:211–221, 1987.
- [19] M. Vasilescu and D. Terzopoulos. Adaptive Meshes and Shells: Irregular Triangulation, Discontinuities, and Hierarchical Subdivision. In *Proceedings of Computer Vision and Pattern Recognition conference*, pages 829–832, 1992.
- [20] R. Verfürth. *A review of a posteriori error estimation and adaptive mesh refinement techniques*. Wiley-Teubner, 1996.
- [21] J. Wilhelms and A. V. Gelder. Octrees for Faster Iso-Surface Generation. In *ACM Transactions on Graphics*, pages 201–227, 1992.



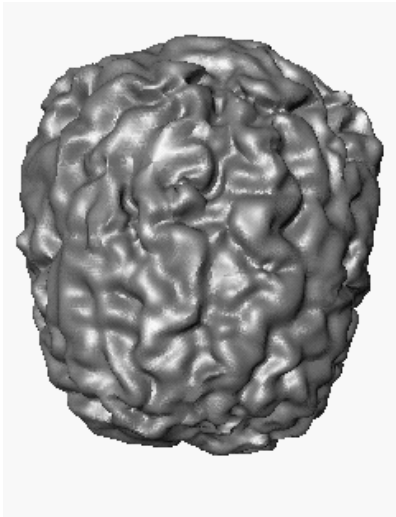
(a) Engine Block



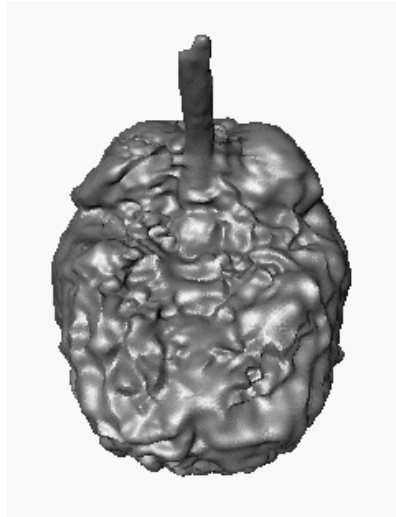
(b) Exhaust of Engine



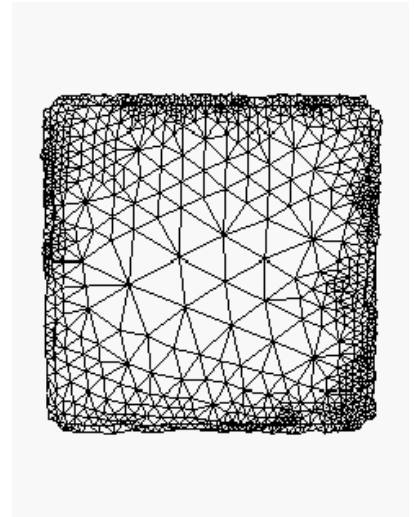
(c) Engine Configuration



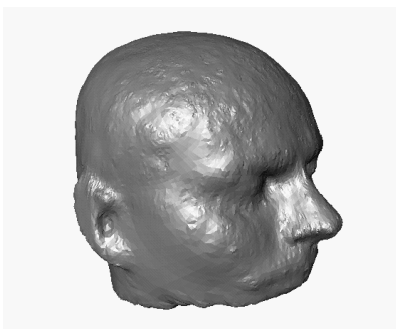
(d) Brain of MRI scan



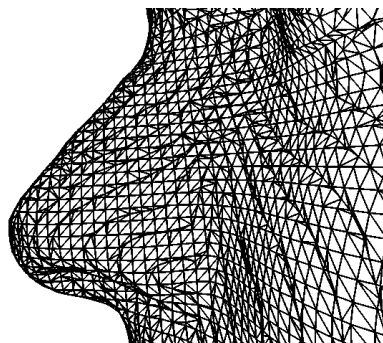
(e) Brain Seen from Below



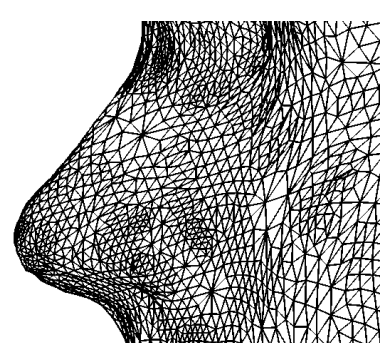
(f) Adaptive Triangulation



(g) Head of MRI Scan



(h) Marching Cubes Mesh



(i) Energy Minimizing Surface Mesh

**Figure 10. Applications of the energy minimizing surface algorithm**