EUROGRAPHICS '2000 / M. Gross and F.R.A. Hopgood
(Guest Editors)

Volume 19, (*2000*), *Number 3*

Multiresolution Shape Deformations for Meshes with Dynamic Vertex Connectivity

Leif P. Kobbelt† Thilo Bareuther Hans-Peter Seidel

Abstract

Multiresolution shape representation is a very effective way to decompose surface geometry into several levels of detail. Geometric modeling with such representations enables flexible modifications of the global shape while preserving the detail information. Many schemes for modeling with multiresolution decompositions based on splines, polygonal meshes and subdivision surfaces have been proposed recently. In this paper we modify the classical concept of multiresolution representation by no longer requiring a global hierarchical structure that links the different levels of detail. Instead we represent the detail information implicitly by the geometric difference between independent meshes. The detail function is evaluated by shooting rays in normal direction from one surface to the other without assuming a consistent tesselation. In the context of multiresolution shape deformation, we propose a dynamic mesh representation which adapts the connectivity during the modification in order to maintain a prescribed mesh quality. Combining the two techniques leads to an efficient mechanism which enables extreme deformations of the global shape while preventing the mesh from degenerating. During the deformation, the detail is reconstructed in a natural and robust way. The key to the intuitive detail preservation is a transformation map which associates points on the original and the modified geometry with minimum distortion. We show several examples which demonstrate the effectiveness and robustness of our approach including the editing of multiresolution models and models with texture.

1. Introduction

With the increasing resolution and complexity of geometric models in computer graphics applications, the necessity of hierarchical representations is getting more and more important. Multiresolution decompositions for highly detailed surface geometries are likely to become the future standard since they enable the surface complexity to be adapted to the available (hardware) resources and the quality requirements of a given application. In addition to complexity control, multiresolution representations enable intuitive and powerful modeling operations since modifications can be applied on any level of detail without affecting coarser levels and with automatic adaption of the finer detail features.

A multiresolution representation consists of a sequence of differently detailed approximations $\mathcal{S}_0, \ldots, \mathcal{S}_m$ of an original surface $\mathcal{S} = \mathcal{S}_m$. Alternatively, $\mathcal{S}_m$ can also be decomposed into a sequence of detail components $\mathcal{D}_i = \mathcal{S}_{i+1} - \mathcal{S}_i$. The efficient use of multiresolution representations requires to find an appropriate mathematical description for the surfaces $\mathcal{S}_i$. For this purpose *hierarchical splines* have been used in [9, 10] where the displacement $\mathcal{D}_i$ between successive levels $\mathcal{S}_i$ is defined by pasting tensor-product splines onto each other.

To get rid of the topological restrictions emerging from the use of tensor-product basis functions, *wavelet techniques* have been generalized to surfaces with arbitrary topology in [25, 30, 32, 39]. Here, the surfaces $\mathcal{S}_i$ are represented by *subdivision schemes* what can be considered as a compromise between spline-surfaces and triangle meshes.

Subdivision schemes exploit the convergence of spline control meshes to the associated surface under knot insertion. The corresponding algorithms apply a set of generalized knot insertion operators (*refinement rules*) which generate a sequence of finer and finer meshes eventually converging to a smooth limit surface. [2, 6, 24, 17, 38]. Consequently, subdivision surfaces are in fact triangle meshes but with an additional mechanism to change the mesh resolution. This is the reason why subdivision techniques are very well suited for computer graphics applications where most of the data is represented by triangle meshes anyway.

In the context of subdivision schemes, the surfaces $\mathcal{S}_i$ naturally emerge from an i-times refined base mesh $\mathcal{M}_0$. The major drawback of this surface representation is that *subdi-*

† Max-Planck-Institute for Computer Sciences, Im Stadtwald, 66123 Saarbrücken, Germany, kobbelt@mpi-sb.mpg.de

vision connectivity is mandatory for the original (fine) mesh $\mathcal{S}_m = \mathcal{M}_m$. As most of the mesh data sets do not come in this form, remeshing is required as a preprocessing step [7, 22, 20].

In [19] and [13] multiresolution techniques have been generalized to meshes with arbitrary connectivity by exploiting fine-to-coarse hierarchies emerging from the application of a mesh decimation algorithm [31, 29, 15, 11, 18]. This enables multiresolution functionality for arbitrary meshes. However, the progressive mesh type multiresolution representations are still associated with the structure of the given mesh. This means that the techniques proposed there cannot be generalized to modeling operations that change the mesh connectivity.

What is common to all these approaches is the fact that the multiresolution representations always impose a *global hierarchical structure* on the data. In the case of hierarchical splines and subdivision surfaces, this structure is provided by the common domain Ω on which all surfaces $\mathcal{S}_i$ are parameterized. For the fine-to-coarse mesh hierarchies this structure is determined by the connectivity of the finest resolution mesh.

The existence of a global structure implies that the representations of the different levels of detail $\mathcal{S}_i$ are not independent from each other. Hence, we cannot perform arbitrary changes on each level. In the context of surface modeling this is a severe restriction since drastic modifications of the geometry cause strong changes in the surface metric. Hence, reparameterization (of the surface) or restructuring (of the mesh) might be necessary to preserve the surface quality.

In this paper we therefore propose a new approach to multiresolution representation and modeling which enables the decomposition of arbitrary geometries and does not require any compatibility condition between the levels $\mathcal{S}_i$ (other than that they are differently detailed approximations of the same surface). The basic idea is to represent the detail information $\mathcal{D}_i$ between the levels by a displacement field without assuming any specific parameterization of the involved surfaces $\mathcal{S}_i$ and $\mathcal{S}_{i+1}$. This enables us to perform arbitrary modifications on each level of a multiresolution decomposition.

The paper is organized as follows: In Section 2, we explain the details of the multiresolution representation which is the basis for the multiresolution shape deformation technique explained in Section 3. Section 4 introduces meshes with dynamic connectivity that enable arbitrary shape modifications while preserving the quality of the tesselation. In Section 5 we explain a simple animation technique based on control ellipsoids which span a membrane surface. In Section 6 we show how the multiresolution shape deformations can be used to globally deform meshes with changing connectivity while the geometric detail is preserved. In Section 7 we demonstrate that the same techniques can also be applied to the deformation of textured models. Finally, we give some hints concerning the efficient implementation of the algorithm in Section 8.

2. Multiresolution Representation

Given an arbitrary surface $\mathcal{S}_m$, a multiresolution *decomposition* consists of a sequence of topologically equivalent surfaces $\mathcal{S}_{m-1}, \ldots, \mathcal{S}_0$ that approximate $\mathcal{S}_m$ with decreasing level of detail. The difference $\mathcal{D}_i = \mathcal{S}_{i+1} - \mathcal{S}_i$ between two successive surfaces is the *detail* on level i which is added or removed when switching between the two approximations. The *reconstruction*

$$\mathcal{S}_m \;=\; \mathcal{S}_i + \mathcal{D}_i + \cdots + \mathcal{D}_{m-1}$$

of the original surface $\mathcal{S}_m$ can start on any level of detail $\mathcal{S}_i$.

Multiresolution *editing* means that on some level of detail, the surface $\mathcal{S}_i$ is replaced by $\mathcal{S}_i'$. This operation does not have any effect on $\mathcal{S}_0, \ldots, \mathcal{S}_{i-1}$. However $\mathcal{D}_{i-1}$ and hence $\mathcal{S}_{i+1}, \ldots, \mathcal{S}_m$ change since the (unchanged) detail information $\mathcal{D}_i, \ldots, \mathcal{D}_{m-1}$ is now added to the modified base surface $\mathcal{S}_i'$ for the reconstruction of $\mathcal{S}_m'$.

To guarantee the intuitive preservation of shape characteristics after a modification on some lower level of detail, this basic setting has to be extended in the sense that the detail information $\mathcal{D}_i$ is encoded with respect to *local frames*. These frames are aligned to the surface geometry of $\mathcal{S}_i$ [9, 10, 19, 39]. If the surfaces $\mathcal{S}_i$ are defined as parameteric functions over a common parameter domain Ω, the use of local frame coding for the details $\mathcal{D}_i$ makes the definition of a proper hierarchy of nested function spaces (*multiresolution analysis*) impossible since the *shape* of $\mathcal{S}_i$ influences the structure of all higher frequency subspaces.

A closely related concept to local frame coding is the representation of detail information $\mathcal{D}_i$ by a *displacement map* on $\mathcal{S}_i$ [3, 21, 28]. Let N_i be a continuous vector field defined on the surface $\mathcal{S}_i$ which assigns a normal vector $N_i(\mathbf{p})$ to every point $\mathbf{p} \in \mathcal{S}_i$. Then the detail information $\mathcal{D}_i$ can be represented by a scalar function λ_i such that

$$\mathcal{S}_{i+1} \;=\; \Big\{ \mathbf{q} \mid \exists \mathbf{p} \in \mathcal{S}_i \,:\, \mathbf{q} = \mathbf{p} + \lambda_i(\mathbf{p})\, N_i(\mathbf{p}) \Big\}.$$

From the continuity of the normal field N_i it follows that $\mathcal{S}_i$ and $\mathcal{S}_{i+1}$ have the same *topology*. Nevertheless, self intersections of $\mathcal{S}_{i+1}$ are possible if $\lambda_i(\mathbf{p})$ is larger than the minimum curvature radius at $\mathbf{p}$. As $\lambda_i(\mathbf{p})$ is unique, no point of $\mathcal{S}_i$ can be mapped to several points on $\mathcal{S}_{i+1}$. This implies some restrictions on the geometric relation between $\mathcal{S}_i$ and $\mathcal{S}_{i+1}$.

When $\mathcal{S}_i$ is transformed into $\mathcal{S}_i'$ by some modeling operation, the reconstruction of $\mathcal{S}_{i+1}'$ is done by using the unchanged detail function λ_i with respect to the new normal field N_i' on $\mathcal{S}_i'$ (cf. Fig. 2). Hence, to perform a multiresolution modification, we need a (inverse) *transformation map* T which associates each point $\mathbf{p}'$ on the new surface $\mathcal{S}_i'$ to a location $\mathbf{p} = T(\mathbf{p}')$ on the old surface $\mathcal{S}_i$. The modified surface $\mathcal{S}_{i+1}'$ is then given by

$$\mathcal{S}_{i+1}' \;=\; \Big\{ \mathbf{q}' \mid \exists \mathbf{p}' \in \mathcal{S}_i' \,:\, \mathbf{q}' = \mathbf{p}' + \lambda_i'(\mathbf{p}')\, N_i'(\mathbf{p}') \Big\}$$

with

$$\lambda_i'(\mathbf{p}') \;=\; \lambda_i\Big(T(\mathbf{p}')\Big).$$

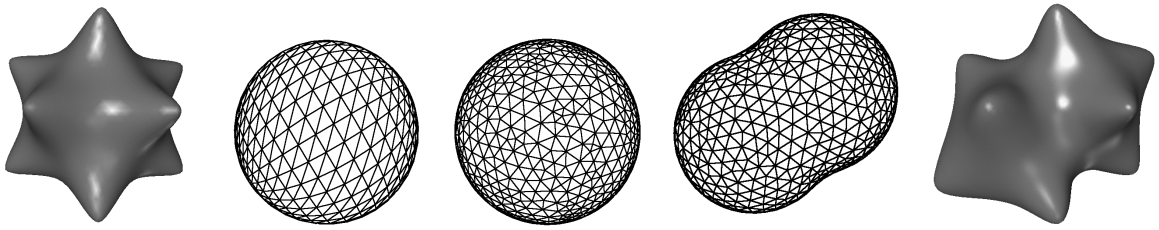

Figure 1: *The multiresolution shape deformation is evaluated from left to right: $\mathcal{S}_1$, $\mathcal{S}_0$, $\mathcal{T}$, $\mathcal{S}_0'$, and $\mathcal{S}_1'$. First we have the original geometry $\mathcal{S}_1$ and its low-frequency approximation $\mathcal{S}_0$. The displacement function λ is sampled by shooting rays from $\mathcal{S}_0$ in normal direction onto $\mathcal{S}_1$. The middle mesh is the* linking mesh *$\mathcal{T}$ between the original and the modified low frequency meshes $\mathcal{S}_0$ and $\mathcal{S}_0'$ respectively. The vertices of $\mathcal{T}$ are projected onto the geometry of $\mathcal{S}_0$. Hence, the detail function λ does not change significantly if we replace $\mathcal{S}_0$ by $\mathcal{T}$ when sampling λ. On the other hand, $\mathcal{T}$ has the same connectivity as $\mathcal{S}_0'$ (middle right) which establishes a one-to-one correspondence between the points on both surfaces. Finally, the deformed surface $\mathcal{S}_1'$ with reconstructed detail is obtained by shifting the vertices of $\mathcal{S}_0'$ in normal direction by the amount λ.*

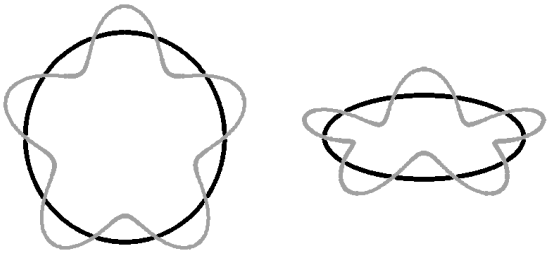

Figure 2: *Left: The geometry $\mathcal{S}_{i+1}$ (grey) is defined relative to the circle $\mathcal{S}_i$ (black). For $\mathbf{p} = [\cos(t), \sin(t)]^T \in \mathcal{S}_i$ we have $\lambda(\mathbf{p}) = \frac{1}{4}\cos(5t)$. Right: When deforming the base geometry $\mathcal{S}_i \to \mathcal{S}_i'$ the modified geometry $\mathcal{S}_{i+1}'$ is reconstructed by using the same detail function λ with respect to the new normal field N_i' on $\mathcal{S}_i'$. To evaluate λ at the correct location, the points on $\mathcal{S}_i'$ and $\mathcal{S}_i$ must be linked by a transformation map T.*

This general definition for multiresolution representations enables us to define the different levels of detail completely independent from each other. There is no need for a global hierarchical structure like in the approaches that are based on a nested sequence of spaces or grids [25, 32, 39]. We do not even have to use the same surface type (parametric, implicit, polygonal meshes) for all surfaces $\mathcal{S}_i$. The only requirements are that we have to be able to compute normal vectors N_i and find ray-intersections. Under these assumptions it is not necessary to set up an explicit formulation for the detail functions λ_i since the *evaluation* of λ_i at some point $\mathbf{p} \in \mathcal{S}_i$ can be accomplished by computing the intersection of the ray $\mathbf{p} + \lambda N_i(\mathbf{p})$ with the surface $\mathcal{S}_{i+1}$ on demand.

3. Multiresolution shape deformation

In computer graphics and geometric modeling, we have several standard techniques for the definition of surface geometry. Besides the implicit definition (*iso-surfaces*) and the parameteric definition (*spline-patches*) we find the explicit definition by *polygonal meshes* particularly useful. Especially *triangle meshes* can be considered the most versatile representation for general free-form surface geometry. Due to the simplicity and robustness of algorithms operating on triangle meshes, we can approximate arbitrarily complicated objects by sufficiently refined tesselations. With increasing resolution (*refinement level*) the non-smooth character of the piecewise linear approximations is quickly vanishing and can be forced below any prescribed tolerance. The complexity of explicit representations can be reduced by efficient schemes to compress the mesh data [4, 12, 35].

In the sequel we will therefore focus on the use of triangle meshes for the representation of the surfaces $\mathcal{S}_i$ in our multiresolution decomposition. Notice that despite their discrete nature, triangle meshes represent *continuous* surfaces. Moreover, we can easily define a continuous normal field by linear interpolation of vertex normals. Hence, given two meshes $\mathcal{S}_i$ and $\mathcal{S}_{i+1}$, we can evaluate the detail function λ_i at arbitrary locations on the mesh $\mathcal{S}_i$. This means in particular that we can arbitrarily super-sample the detail function without any restrictions imposed by the actual resolution of the underlying tesselation. This is an important observation that we exploit when a multiresolution shape deformation strongly changes (e.g., stretches) the global shape of an object.

For simplicity we restrict our explanation to a *two-band decomposition* of the given geometry. The generalization to multi-band decompositions is straightforward since the evaluation/reconstruction procedure on the *i*th level of detail recursively calls the same evaluation procedure for the next coarser approximation level. Without loss of generality we further assume that a single multiresolution modeling operation applies modifications only on one frequency band.

Let the two-band representation of a surface be given by two triangle meshes $\mathcal{S}_0$ and $\mathcal{S}_1$ with arbitrary connectivity each. The multiresolution deformation replaces the mesh $\mathcal{S}_0$ by $\mathcal{S}_0'$ (again with a completely different connectivity). We want to reconstruct the result of the operation $\mathcal{S}_1'$.

According to the multiresolution representation of the last section, we find points $\mathbf{q}'$ on $\mathcal{S}_1'$ by displacing points $\mathbf{p}' \in \mathcal{S}_0'$

in normal direction. The distance $\lambda'(\mathbf{p}')$ by which $\mathbf{p}'$ is moved, has to be sampled from the displacement map that is implicitly defined by the geometries of the two original surfaces $\mathcal{S}_0$ and $\mathcal{S}_1$. Hence, suppose we know the transformation map T, we can compute the point $\mathbf{p} = T(\mathbf{p}')$ on $\mathcal{S}_0$ and shoot a ray $\mathbf{p} + \lambda N_0(\mathbf{p})$. The distance to the first intersection point on $\mathcal{S}_1$ yields the value $\lambda(\mathbf{p}) =: \lambda'(\mathbf{p}')$.

The remaining question is how to find an appropriate map

$$T : \mathcal{S}'_0 \rightarrow \mathcal{S}_0.$$

This is not trivial since $\mathcal{S}_0$ and $\mathcal{S}'_0$ can have different geometry *and* different connectivity. Yet, the quality of the detail reconstruction strongly depends on the *distortion* caused by T.

We propose to generate the map T by constructing an additional mesh $\mathcal{T}$ which has the same connectivity as $\mathcal{S}'_0$ and the same geometry as $\mathcal{S}_0$, i.e. whose vertices $\mathbf{p} \in \mathcal{T}$ lie on the continuous surface $\mathcal{S}_0$. Due to the fact that $\mathcal{T}$ and $\mathcal{S}_0$ are smooth tesselations (low frequency approximations) which have approximately the same geometry, the difference between the displacement function λ sampled from either mesh is neglectible. One the other hand, since $\mathcal{T}$ and $\mathcal{S}'_0$ have the same connectivity, it is trivial to establish a one-to-one correspondence between both by using a barycentric parameterization within each triangle. Hence, for a given point $\mathbf{p}'$ on $\mathcal{S}'_0$ we can compute the displacement value $\lambda'(\mathbf{p}')$ by shooting a ray in normal direction from the corresponding point $\mathbf{p}$ on $\mathcal{T}$.

Fig. 1 shows all the meshes involved in this multiresolution modeling operation. The distortion of the transformation map T and hence the quality of the detail reconstruction is determined by the *distribution* of the vertices of $\mathcal{T}$ on the surface $\mathcal{S}_0$. This distribution has to optimize two conflicting objectives. First, the points should be distributed evenly over $\mathcal{S}_0$ in order to guarantee a uniform sampling densitiy for λ. On the other hand, the relative position of neighboring vertices should be similar to the corresponding configuration on the mesh $\mathcal{S}'_0$ in order to minimize local distortions (e.g. preserve the triangle's aspect ratios). The degree of difficulty for this optimization task depends on the geometric difference between the shapes of $\mathcal{S}_0$ and $\mathcal{S}'_0$.

Let us assume that the shape difference between $\mathcal{S}_0$ and $\mathcal{S}'_0$ is small, i.e. both meshes have only a small Hausdorff-distance from each other. This assumption does not restrict the set of possible modeling operations since we can decompose every drastic shape deformation into a sequence of small deformations (in the Hausdorff-sense). In order to compute the transformation map T for a big deformation we simply concatenate the corresponding maps for the small deformations.

For the generation of $\mathcal{T}$ we start by copying the connectivity of $\mathcal{S}'_0$ and apply two operators that control the distribution of the vertices $\mathbf{p} \in \mathcal{T}$ on the surface $\mathcal{S}_0$. The first operator **P** (*project*) computes initial positions for each vertex $\mathbf{p} \in \mathcal{T}$. The second operator **D** (*distribute*) then shifts the points $\mathbf{p}$ in order to distribute them more evenly over the surface $\mathcal{S}_0$. This operator reduces the global distortion of the map T.

In Sect. 8 we explain a simple and efficient method to find the closest triangle $F = \triangle(A,B,C) \in \mathcal{S}_0$ to a vertex $\mathbf{q}' \in \mathcal{S}'_0$. The **P** operator projects each vertex $\mathbf{q}' \in \mathcal{S}'_0$ onto the nearest point $\tilde{\mathbf{q}}$ on F (this is not necessarily an orthogonal projection) which yields the initial position for the corresponding vertex of $\mathcal{T}$. After the application of the **P** operator, the distribution of the vertices in the initial mesh $\mathcal{T}$ are "as similar as possible" to the distribution of the vertices in $\mathcal{S}'_0$.

The **D** operator equalizes the density of the vertices $\mathbf{q} \in \mathcal{T}$ scattered over the surface $\mathcal{S}_0$. For this, we first estimate the local density d for each vertex $\mathbf{q}$, e.g., by computing the average length of the adjacent edges. We then update the vertex positions for all $\mathbf{q} \in \mathcal{T}$ by the *density weighted umbrella rule* [19, 5]

$$\mathbf{q} \leftarrow \frac{1}{\sum_j d_j} \sum_{j=0}^{n-1} d_j \, \mathbf{q}_j \qquad (1)$$

where n is the valence of $\mathbf{q}$ and $\mathbf{q}_j$ are its direct neighbors with their corresponding density d_j. The effect of the weight coefficients is that vertices with low point density (large average edge length) pull harder. Hence, the density egalization is performed more aggressively. On the other hand, neighboring vertices with high density have reduced attraction which avoids clustering effects (*shrinking*) known from the uniform umbrella operator [19, 34].

After the application of (1) the vertices $\mathbf{q}$ are no longer lying on the surface $\mathcal{S}_0$. Hence, another projection **P** onto the nearest triangle concludes the **D** operation. Usually, the **D** operator has to be applied several times to obtain a good vertex distribution.

Fig. 1 shows a simple example for the detail reconstruction based on the transformation map T after the shape deformation. In Sect. 6 and 7 we will see more examples which demonstrate that this basic technique enables strong multiresolution deformations while keeping the distortion of the map T minimal. However, before we go into the details of multiresolution animation, we have to define a triangle mesh based surface representation which is flexible enough to perform arbitrary shape deformations.

4. Dynamic Connectivity Meshes

In general, a triangle mesh is defined by the position of its vertices $\mathbf{p}_i$ and their connectivity $(\mathbf{p}_i, \mathbf{p}_j)$. When modifying the *shape* of an object (not its *topology*) it is often considered sufficient to merely change the position of the vertices but not to adapt their connectivity. Since the actual continuous surface of a triangle mesh is defined by piecewise linear interpolation between the vertices, the modification of each control vertex $\mathbf{p}$ is equivalent to changing the vector valued coefficient of a locally supported fixed piecewise linear basis function (*hat-function*) centered at $\mathbf{p}$. As global modifications usually require to update the positions of many vertices, such modeling operations are controlled by coarse scale basis functions (*shape functions*) which are super-imposed over a whole subregion of the given mesh.

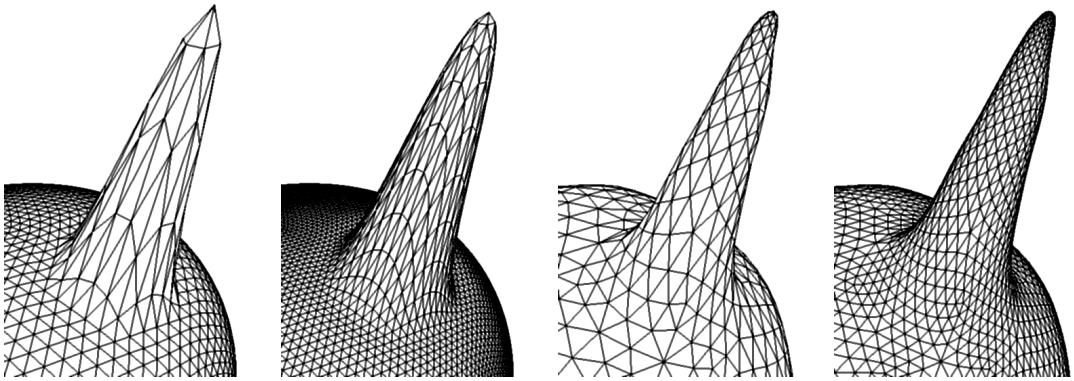

Figure 3: *Moving a selection of vertices in a triangle mesh causes distortion of the adjacent triangles (far left). This effect cannot be removed by uniform subdivision (center left). However, dynamically restructuring the mesh yields better shaped triangles (right). The resolution of the mesh can be controlled by prescribing the minimum and maximum edge length.*

The major problem emerging from this kind of geometric modification is that moving a selection of mesh vertices causes a distortion of the adjacent triangles and hence affects the quality of the mesh (cf. Fig. 3). Applying uniform 1-to-4 splits in regions where the discrete curvature is above some threshold or the length of the edges exceeds a prescribed bound [39] does not solve this problem since uniform splits do not improve the aspect ratio of the triangles. Moreover, we are interested in finding a surface representation where the number of triangles is *proportional* to the total surface area. However, when using uniform refinement (*subdivision connectivity*), the number of triangles increases *quadratically* if the mesh is streched in one direction (cf. Fig. 3).

We therefore prefer a mesh optimization technique similar to [16, 33, 37] which uses simple operations to improve the quality of a given mesh by changing its connectivity. Our goal is to find a mesh representation which guarantees a certain mesh quality according to the following requirements [1]:

- No edge should be shorter than $\varepsilon_{\min}$
- No edge should be longer than $\varepsilon_{\max}$
- A vertex' valence should be six.
- Long and thin triangles should be avoided

We call this representation *dynamic triangle meshes* since every geometric modification (alteration of the vertices' position) is followed by a *restructuring* of the mesh, i.e., by the application of simple operations which change the mesh connectivity in order to re-establish the above quality requirements. As the surface representation is not static anyway, it is not necessary to treat the quality requirements as hard limits. In order to avoid strong distortions in the mesh it is sufficient to at least reduce the number of violations with every restructuring step. This guarantees that the mesh quality stays sufficiently close to an optimal configuration.

The restructuring is performed in several stages. First all edges which are shorter than $\varepsilon_{\min}$ are removed by collapsing the two end-vertices. To avoid problems with "chains" of short edges, we collapse that end-vertex with lower valence into the one with higher (cf. Fig. 4). Then, all edges which are longer than $\varepsilon_{\max}$ are split by inserting a new vertex at its mid-point. The two adjacent triangles are bisected accordingly (cf. Fig. 5). Notice that the upper and the lower bound on the edge lengths are only compatible if $\varepsilon_{\max} > 2\varepsilon_{\min}$ since otherwise the edge split operation would generate two invalid edges.

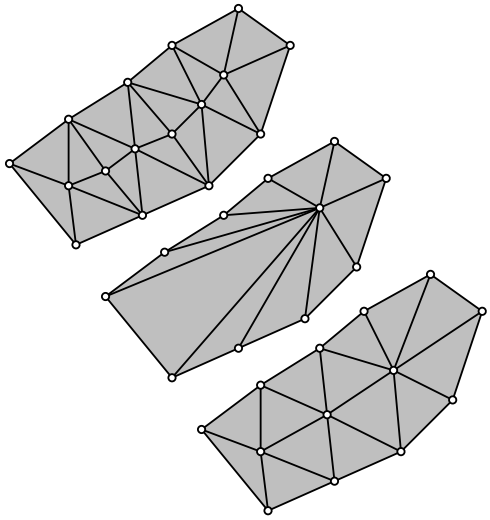

Figure 4: *Small edges are removed by edge collapses. The accumulation of edge collapses (middle) is prevented by collapsing into the vertex with higher valence (bottom). This simple heuristic works because high valence vertices stay fixed and every collapse reduces the number of adjacent short edges. Moving a high valence vertex* **v***, however, can lead to an unbounded accumulation of edge collapses since new short edges can become adjacent to* **v**.

After all edge lengths lie within the target interval $[\varepsilon_{\min}, \varepsilon_{\max}]$ we perform edge-flipping in order to regularize

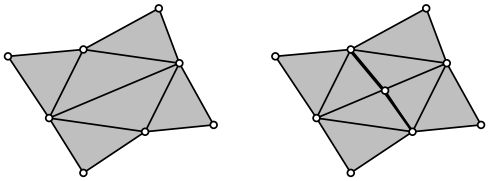

Figure 5: *Long edges are removed by mid-point insertion. If this operation generates short edges (the thick ones) with length* $< \varepsilon_{\max}$*, these will be removed in the* next *restructuring step.*

the connectivity. This becomes necessary since both operations, i.e., the edge collapse and the edge split, do affect the valence balance for the involved vertices. For example the edge-splitting always introduces vertices of valence four and hence disturbes the balance. If the edge flipping should generate edges which are too long or too short, those will be removed in the *next* restructuring step.

For every two neighboring triangles $\triangle(A,B,C)$ and $\triangle(C,B,D)$ we maximize the number of vertices with valence six by flipping the diagonal $\overline{BC}$ if the total valence excess

$$\sum_{\mathbf{p}\in\{A,B,C,D\}} \Big(\text{valence}(\mathbf{p})-6\Big)^2$$

is reduced. The exponent 2 disallows to trade, e.g., two valence 5 vertices for one valence 8 vertex. Fig. 6 shows the different stages of the restructuring process.

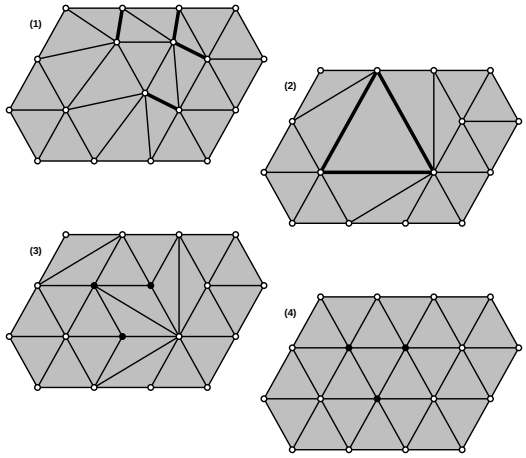

Figure 6: *The quality of a modified mesh (1) can be improved by (2) collapsing short edges, (3) splitting long edges, and (4) flipping edges where the valence excess can be reduced.*

The last quality requirement concerning the aspect ratio of the triangles turns out to be automatically satisfied by the above restructuring algorithm since the edge length ratio is bounded by $\varepsilon_{\max}/\varepsilon_{\min}$ and all vertex valences are close to six which restricts the minimum angle.

5. Animation by control ellipsoids

We suggest a simple mesh animation technique in order to demonstrate how dynamic connectivity meshes can be used to generate high quality meshes for each frame of an animation sequence. The goal is to have a smooth free form object like a bubble which moves and deforms in time. This was motivated by watching a lava lamp.

The shape deformations in a lava lamp can be modeled in the following manner: Two or more ellipsoids move in space. Their radii, aspect ratio, and orientation changes in time. Around these *control ellipsoids* we span a membrane surface which tends to minimize its surface area due to internal forces. The result is a physically based model for the skin of lava lamp bubbles.

In [19, 5] a simple scheme is proposed to generate triangle meshes which are optimal with respect to the (discrete) membrane energy. Solving the optimization problem for a given mesh by an iterative Gauß-Seidel solver requires to compute only simple linear combinations of vertex positions. For each vertex $\mathbf{p}$ with neighbors $\mathbf{p}_0,\ldots,\mathbf{p}_{n-1}$ the Gauß-Seidel update rule is

$$\mathbf{p} \leftarrow \frac{1}{n}\sum_{i=0}^{n-1}\mathbf{p}_i. \qquad (2)$$

In order to make the optimization problem well-defined, it is necessary to impose additional boundary conditions. However, without boundary conditions the update rules of the iterative solver can still be applied as a smoothing convolution filter [34].

In our case we provide boundary conditions by using the control ellipsoids. After each Gauß-Seidel iteration of the umbrella algorithm, we check for every mesh vertex whether it happens to lie in the interior of one of the ellipsoids. If this is the case then we project it back to the corresponding ellipsoid's surface. In order to generate a lava lamp animation we simply have to prescribe the ellipsoid parameters for every frame. The bubble surface for a given time step t_{i+1} is computed by projecting the vertices of the surface from the previous time step t_i onto the modified control ellipsoids, applying umbrella iterations, and re-enforcing the boundary constraints.

These operations only affect the geometric position of the vertices. In order to additionally maintain the mesh quality, the restructuring algorithm of the last section which updates the connectivity, is applied to the resulting mesh. Fig. 7 shows several frames of an animation created by this simple technique.

As mentioned in [19], the convergence of the Gauß-Seidel scheme applied to the membrane energy optimization, is rather slow. A standard technique in numerical analysis to cope with this problem is to use multi-grid solvers [14] which compute solutions to the optimization problem on coarser tesselations in order to find better starting values.

Usually, such multi-level algorithms are applied to a sequence of nested grids which are generated from coarse to

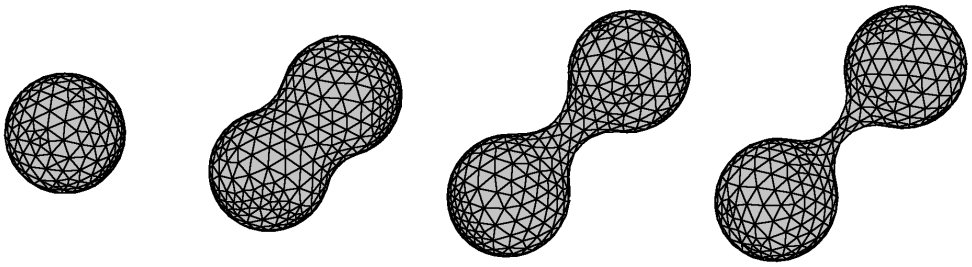

Figure 7: *A bubble object can be animated by spanning a membrane skin around a set of control ellipsoids. Since the skin is represented by a dynamic connectivity mesh, the quality of the triangular faces is guaranteed by restructuring the mesh between the frames.*

fine by uniform refinement of a coarse base mesh. In [19] the method is generalized to meshes with arbitrary connectivity where a hierarchy of nested grids can be generated from fine to coarse by incremental mesh decimation [11, 15, 18, 29, 31]. In the case of dynamic connectivity meshes, however, we cannot generate a sequence of nested grids since the actual connectivity is determined by the restructuring step and unknown a priori.

We therfore use a multi-level solver which controls the coarseness of the tesselation by adapting the bounds for the edge lengths $\varepsilon_{\min}$ and $\varepsilon_{\max}$. For every frame of the animation sequence, we start with rather large values for $\varepsilon_{\min}$ and $\varepsilon_{\max}$ and apply several steps of alternating umbrella updates and projection onto the control ellipsoids (*inner loop*). Then we slowly decrease the bounds by some factor $q < 1$. In the restructuring step this triggers edge split operations which refine the mesh. The alternating mesh optimization and restructuring is repeated until the target resolution is reached (*outer loop*). The following pseudo-code implements the multi-level solver.

```
for each frame
     init εmin, εmax
     while (εmax > τ)
          while not convergence
               minimize energy
               project to ellipsoid
          εmin *= q, εmax *= q
          restructure
```

Fig. 8 shows several intermediate meshes of this process. The generalized multi-level version of the membrane energy minimization is significantly faster than plain Gauß-Seidel iterations — especially for high target resolutions. For moderately complex models ($\approx 10^4 \triangle$) we can easily compute several frames per second on a standard graphics workstation (SGI O2, R10K).

As demonstrated in [27], the control ellipsoid animation technique can also be applied to geometric modeling. The designer can use control ellipsoids as modeling tools to push or pull the surface. Even more general shapes are possible for the tool geometry as long as the inside test can be computed effectively and the projection onto the tool's surface is well-defined. Similar to [19, 37] we could also use the thin-plate energy instead of the membrane energy and we could restrict the influence of a modification to a presribed sub-region of the mesh.

6. Animating a multiresolution object

We now apply the multiresolution shape deformation technique to surface representations based on triangle meshes with dynamic connectivity. This enables us to compute animation sequences where the deformation of the global shape is prescribed while the detail information is preserved in a natural fashion. The shape deformation during the animation might require to dynamically adapt the connectivity from frame to frame in order to maintain the mesh quality. Hence, we have to be able to resample the detail function λ at arbitrary locations since we cannot tell a priori where additional detail information will be required.

The basic idea is to take an arbitrary triangle mesh $\mathcal{M}_1$ and a low-frequency approximation $\mathcal{M}_0$ as a two-band multiresolution representation of the input data. The mesh $\mathcal{M}_0$ is then animated, e.g., by using the control ellipsoid technique. For every frame, the detail will be reconstructed by shifting the vertices of the current mesh in normal direction according to the resampled detail function λ.

For the computation of the animation sequence we need four different meshes. First we need the low-frequency approximation of the current frame $\mathcal{S}_{\text{current}}$ and the linking mesh $\mathcal{T}_{\text{current}}$ which has the same connectivity as $\mathcal{S}_{\text{current}}$ but with vertex positions lying on the surface $\mathcal{M}_0$. This mesh represents the current transformation map T_{current} that links the points $\mathbf{p}$ on $\mathcal{S}_{\text{current}}$ to locations $T(\mathbf{p})$ on $\mathcal{M}_0$ where the detail function λ has to be sampled in order to compute the displacement for the detail reconstruction at $\mathbf{p}$. For the first frame, both meshes are indentical to $\mathcal{M}_0$ (*starting configuration*).

As the animation progresses to the next frame, the new low-frequency shape of the animated object is stored in $\mathcal{S}_{\text{next}}$. We do not assume any coherence between the *connectivities* of $\mathcal{S}_{\text{current}}$ and $\mathcal{S}_{\text{next}}$. For the proper reconstruction of the detail, we have to find the new transformation map $T_{\text{next}} : \mathcal{S}_{\text{next}} \rightarrow \mathcal{M}_0$ which is difficult to compute directly

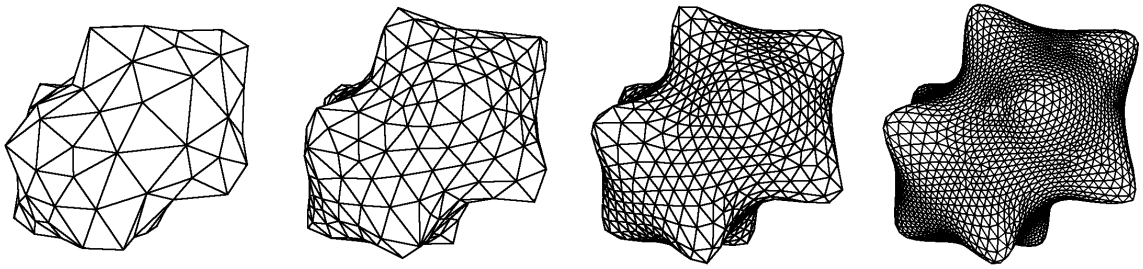

Figure 8: *The concept of multi-level smoothing can be generalized to meshes with dynamic connectivity. The different hierarchy levels are defined by adapting the upper and lower bound for the edge lengths. From left to right, the values* $\varepsilon_{\min}$ *and* $\varepsilon_{\max}$ *are halved in every step. Notice that these meshes are* ***not*** *generated by uniform refinement but by dynamic restructuring.*

since $\mathcal{S}_{\text{next}}$ and $\mathcal{M}_0$ can have completely different shapes. However, as it is usually the case in smooth animations, we can assume that the geometric shape of $\mathcal{S}_{\text{current}}$ and $\mathcal{S}_{\text{next}}$ do not differ very much. Hence, we can try to adapt the current transformation map T_{current} to the next frame.

As explained in Sect. 3, the map T_{next} is completely determined by the vertex positions of the mesh $\mathcal{T}_{\text{next}}$ which have to lie on the surface $\mathcal{M}_0$. The distribution of these vertices should to be optimized in order to minimize the distortion of the resulting map T_{next}.

To accomplish this, we start by computing the projection of the vertices of $\mathcal{S}_{\text{next}}$ onto the mesh $\mathcal{S}_{\text{current}}$. This can be done rather efficiently as explained in Sect. 8. If the vertex $\mathbf{p} \in \mathcal{S}_{\text{next}}$ is projected onto the triangle $\triangle(A,B,C) \in \mathcal{S}_{\text{current}}$ then the projection point $\tilde{\mathbf{p}}$ can be written in barycentric coordinates

$$\tilde{\mathbf{p}} = \alpha A + \beta B + \gamma C.$$

By applying the same linear combination to the corresponding vertices of $\mathcal{T}_{\text{current}}$ we obtain a good initial guess for the position of the vertex $\mathbf{p}' \in \mathcal{T}_{\text{next}}$.

The quality of the mesh $\mathcal{T}_{\text{next}}$ (and hence of the transformation map T_{next}) can be further improved by several applications of the **D** operator of Sect. 3. If we use triangle meshes with dynamic connectivity as explained in Sect. 4, we can accelerate the convergence of the density weighted umbrella algorithm by the multi-level smoothing technique of Sect. 5. As $\mathcal{S}_{\text{next}}$ and $\mathcal{T}_{\text{next}}$ have to have identical connectivities anyway we can perform the restructuring operations on both meshes in parallel. Of course, since the purpose of the restructuring is to improve the quality of the mesh $\mathcal{S}_{\text{next}}$, all edge collapses and splits are triggered by the edge lengths of $\mathcal{S}_{\text{next}}$ alone! During the smoothing phase on each level of resolution, the uniform umbrella operator (2) is applied to $\mathcal{S}_{\text{next}}$ while the density weighted umbrella operator (1) is applied to $\mathcal{T}_{\text{next}}$. The vertices of $\mathcal{T}_{\text{next}}$ have to be projected back to the surface $\mathcal{M}_0$ after every smoothing step.

7. Animating a textured object

The same technique for multiresolution shape deformation can also be applied to the animation of textured objects. In fact, the texture coordinates which are assigned to the points of a surface can be considered as a special type of detail information. The handling of texture information is easier than the treatment of geometric detail since no local frame coding is necessary.

Many authors (e.g., [26, 23]) proposed sophisticated techniques to optimize the mapping of plane texture data to parameteric surfaces in space. In our case, we assume the texture coordinates (u_j, v_j) are already assigned to the vertices $\mathbf{p}_j$ of the input mesh $\mathcal{M}_0$. The problem is then to adapt the texture coordinates during the shape deformation in order to minimize the distortion.

The animation of a textured object works exactly as the two-band multiresolution animation in the last section. The only difference is that the high-band $\mathcal{M}_1$ is replaced by the texture information. We still use the four meshes $\mathcal{S}_{\text{current}}$, $\mathcal{T}_{\text{current}}$, $\mathcal{S}_{\text{next}}$, and $\mathcal{T}_{\text{next}}$ to adapt the transformation map T from frame to frame.

Fig. 10 shows several examples of a deformed textured globe. The deformation is modeled by control ellipsoids and the mesh representation is based on dynamic connectivity meshes. The **D** operator in the computation of the transformation map T obviously works very effective in reducing the texture distortion.

8. Implementation

Besides the mapping and smoothing operations explained in the previous sections, we need to optimize some lower level operations to let the algorithm run efficiently. One very time consuming (and frequent) step in the multiresolution deformation process is the projection of points from one mesh onto the triangles of another. We can greatly reduce the computational complexity by caching some local information and by exploiting spatial coherence in the meshes.

In the **P** operation we have to find for each vertex $\mathbf{p}$ of the deformed mesh $\mathcal{S}_0'$, the nearest point $\tilde{\mathbf{p}}$ on the previous mesh $\mathcal{S}_0$. We cannot build a global space partition like an octree to accelerate this search since the geometry and connectivity are changing dynamically and hence the maintainance of the space partition would be too expensive. One important

observation, however, that leads to an efficient algorithm, is that for two nearby vertices on $\mathcal{S}_0'$ the corresponding closest points on $\mathcal{S}_0$ are very likely to be nearby as well. This is true because $\mathcal{S}_0$ and $\mathcal{S}_0'$ only have a rather small Hausdorff-distance (compared to their size).

As the **P** operation has to project *all* vertices of $\mathcal{S}_0'$, we can exploit this observation by enumerating the vertices in a recursive traversal algorithm such that the expected distance between successive vertices is small. In this case we can use the projection point of the previous vertex as a starting point for finding the next projection point.

Suppose the vertex $\mathbf{p} \in \mathcal{S}_0'$ is projected onto $\tilde{\mathbf{p}} \in \triangle(A,B,C) \subset \mathcal{S}_0$ and $\mathbf{q} \in \mathcal{S}_0'$ is a vertex close to $\mathbf{p}$ then we start with the triangle $\triangle(A,B,C)$ and compute

$$\mathbf{q} - A = \alpha N + \beta(B-A) + \gamma(C-A)$$

where $N = (B-A) \times (C-A)$ is the normal vector to $\triangle(A,B,C)$. As we are only interested in the orthogonal projection $(1-\beta-\gamma)A + \beta B + \gamma C$, it is sufficient to solve the symmetric (2×2)-system

$$\begin{pmatrix} (B-A)^T(B-A) & (C-A)^T(B-A) \\ (B-A)^T(C-A) & (C-A)^T(C-A) \end{pmatrix} \begin{pmatrix} \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} (P-A)^T(B-A) \\ (P-A)^T(C-A) \end{pmatrix}.$$

If the barycentric coordinates are positive, we have found the projected point $\tilde{\mathbf{q}}$. Otherwise we proceed in any direction indicated by a negative barycentric coordinate. To terminate the marching algorithm correctly we have to check for loops, i.e., for triangles that are visited twice. If this happens, then the closest point to $\mathbf{q}$ lies on an edge of $\mathcal{S}_0$. Once we detect a loop, we can easily compute the projected point by scaling the non-negative barycentric coordinates such that they sum to one. For sufficiently smooth meshes $\mathcal{S}_0$, this case is relatively unlikely to occur.

The traversal by which the vertices of $\mathcal{S}_0'$ are enumerated is similar to algorithms that generate triangle strips [8]. However in our case we are not that much interested in long strips but more in avoiding far jumps during the traversal. Our algorithm starts with an arbitrary edge in the mesh. For the first vertex we have to find the projection point by brute-force testing. For the second and all subsequent vertices we can use the above marching technique.

We recursively enumerate the vertices of $\mathcal{S}_0'$ by depth-first traversal of a binary tree which is implicitly defined by the neighbor relation between the triangles (each node has three neighbors, i.e., one parent and two children) [36]. For every new vertex $\mathbf{q}$ for which we have to compute the projection, we find at least one direct neighbor $\mathbf{p}$ that has already been processed. Hence, we immediately find a nearby triangle on $\mathcal{S}_0$ to start the marching algorithm. Practical experiments proved that the marching usually has to bridge only rather small distances. For example if both geometries $\mathcal{S}_0$ and $\mathcal{S}_0'$ have the same resolution we need less than two marching steps in average to reach the destination triangle.

In some rare cases, this procedure can fail to find the correct projection point $\tilde{\mathbf{q}}$. In these cases the wrongly reported point typically lies quite far away on the opposite side of the mesh. This malfunction occurs if the starting triangle for the marching procedure is too far from the actual projection point. We can easily check the feasibility of a reported projection point $\tilde{\mathbf{q}}$ by testing its distance to the point $\mathbf{q}$. For example in the animations controlled by moving ellipsoids, we know that the surface cannot move further than the maximum shift of one of the ellipsoids plus its maximum change of radius. Hence projection points which lie farther away than this distance must be wrong.

In practice, it turned out that the occurence of this error is very rare. So we use a brute-force fall-back solution which simply tests the vertex agains all triangles of the mesh $\mathcal{S}_0'$. This does not affect the overall performance of the algorithm.

9. Conclusions

In this paper we presented a new approach to multiresolution free-form deformations. Instead of defining a global hierarchy across all levels of the multiresolution decomposition, we define detail information as the geometric difference between two differently detailed approximations $\mathcal{S}_i$ and $\mathcal{S}_{i+1}$ of the same object with no further restrictions. The detail information is implicitly represented as a displacement field with respect to the normal vectors on $\mathcal{S}_i$. The detail can be sampled (evaluated) at arbitrary points by computing ray intersections with $\mathcal{S}_{i+1}$. A global parameterization of the surfaces $\mathcal{S}_i$ is not necessary.

The technique enables very general multiresolution deformations and animations with very stable reconstruction of the detail information. The underlying surface representation is based on triangle meshes with dynamic connectivity which guarantees high mesh quality since the tesselation automatically adapts to the deformation. Using multi-level algorithms for the mesh smoothing significantly accelerates the animation such that realtime editing with moderately complex meshes is possible on standard graphics workstations.

The technique has also been applied to the animation of textured objects. In this case the low distortion of the transformation map which transfers the detail information from the original surface to the deformed one, generates realistic texture maps.

References

1. BERN, M., AND EPPSTEIN, D. Mesh generation and optimal triangulation. *Computing in Euclidean Geometry* (1992), 23 – 90. Lecture Notes on Computing, D. Du and F. Hwang (eds.).

2. CATMULL, E., AND CLARK, J. Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes. *Computer Aided Design 10*, 6 (Nov. 1978), 239–248.

3. COOK, R. Shade trees. In *Computer Graphics (SIGGRAPH 84 Proceedings)* (1984), pp. 223–231.

4. DEERING, M. Geometry Compression. In *Computer Graphics (SIGGRAPH 95 Proceedings)* (1995), pp. 13–20.

5. DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Computer Graphics (SIGGRAPH 99 Proceedings)* (1999), pp. 317 – 324.

6. DOO, D., AND SABIN, M. Behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design 10* (1978), 356–360.

7. ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., AND STUETZLE, W. Multiresolution Analysis of Arbitrary Meshes. In *Computer Graphics (SIGGRAPH 95 Proceedings)* (1995), pp. 173–182.

8. EVANS, F., SKIENA, S., AND VARSHNEY, A. Optimizing triangle strips for fast rendering. Technical Report, State University of New York, Stony Brook.

9. FORSEY, D., AND WONG, D. Multiresolution surface reconstruction for hierarchical B-splines. Tech. rep., University of British Columbia, 1995.

10. FORSEY, D. R., AND BARTELS, R. H. Hierarchical B-spline refinement. In *Computer Graphics (SIGGRAPH 88 Proceedings)* (1988), pp. 205–212.

11. GARLAND, M., AND HECKBERT, P. S. Surface Simplification Using Quadric Error Metrics. In *Computer Graphics (SIGGRAPH 97 Proceedings)* (1997), pp. 209–218.

12. GUMHOLD, S., AND STRASSER, W. Real time compression of triangle mesh connectivity. In *Computer Graphics (SIGGRAPH 98 Proceedings)* (1998), pp. 133 – 140.

13. GUSKOV, I., SWELDENS, W., AND SCHRÖDER, P. Multiresolution signal processing for meshes. In *Computer Graphics (SIGGRAPH 99 Proceedings)* (1999), pp. 325 – 334.

14. HACKBUSCH, W. *Multi-Grid Methods and Applications*. Springer Verlag, Berlin, 1985.

15. HOPPE, H. Progressive Meshes. In *Computer Graphics (SIGGRAPH 96 Proceedings)* (1996), pp. 99–108.

16. HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. Mesh Optimization. In *Computer Graphics (SIGGRAPH 93 Proceedings)* (1993), pp. 19–26.

17. KOBBELT, L. Interpolatory Subdivision on Open Quadrilateral Nets with Arbitrary Topology. In *Computer Graphics Forum, Proceedings of Eurographics '96* (1996), pp. C407–C420.

18. KOBBELT, L., CAMPAGNA, S., AND SEIDEL, H.-P. A general framework for mesh decimation. In *Proceedings of the Graphics Interface conference '98* (1998).

19. KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. Interactive multi-resolution modeling on arbitrary meshes. In *Computer Graphics (SIGGRAPH 98 Proceedings)* (1998), pp. 105 – 114.

20. KOBBELT, L., VORSATZ, J., LABSIK, U., AND SEIDEL, H.-P. A shrink wrapping approach to remeshing polygonal surfaces. In *Computer Graphics Forum 18* (1999), pp. 119 – 130. Eurographics '99 issue.

21. KRISHNAMURTHY, V., AND LEVOY, M. Fitting smooth surfaces to dense polygon meshes. In *Computer Graphics (SIGGRAPH 96 Proceedings)* (1996), pp. 313–324.

22. LEE, A., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. Multiresolution adaptive parameterization of surfaces. In *Computer Graphics (SIGGRAPH 98 Proceedings)* (1998), pp. 95 – 104.

23. LEVY, B., AND MALLET, J. Non-distorted texture mapping for sheared triangulated meshes. In *Computer Graphics (SIGGRAPH 98 Proceedings)* (1998), pp. 343 – 352.

24. LOOP, C. Smooth subdivision surfaces based on triangles, 1987. Master's thesis, Utah University, USA.

25. LOUNSBERY, M., DEROSE, T., AND WARREN, J. Multiresolution Analysis for Surfaces of Arbitrary Topological Type. *ACM Transactions on Graphics 16*, 1 (January 1997), 34–73.

26. MAILLOT, J., YAHIA, H., AND VERROUST, A. Interactive texture mapping. In *Computer Graphics (SIGGRAPH 93 Proceedings)* (1993), pp. 27 – 34.

27. MARKOSIAN, L., COHEN, J., CRULLI, T., AND HUGHES, J. Skin: A constructive approach to modeling free-form shapes. In *Computer Graphics (SIGGRAPH 99 Proceedings)* (1999), pp. 393 – 400.

28. PEDERSEN, H. Displacement mapping using flow fields. In *Computer Graphics (SIGGRAPH 94 Proceedings)* (1994), pp. 279–286.

29. RONFARD, R., AND ROSSIGNAC, J. Full-Range Approximation of Triangulated Polyhedra. In *Computer Graphics Forum, Proceedings of Eurographics '96* (1996), pp. C67–C76.

30. SCHRÖDER, P., AND SWELDENS, W. Spherical wavelets: Efficiently representing functions on the sphere. In *Computer Graphics (SIGGRAPH 95 Proceedings)* (1995), pp. 161–172.

31. SCHROEDER, W. J., ZARGE, J. A., AND LORENSEN, W. E. Decimation of Triangle Meshes. In *Computer Graphics (SIGGRAPH 92 Proceedings)* (1992), pp. 65–70.

32. STOLLNITZ, E., DEROSE, T., AND SALESIN, D. *Wavelets for Computer Graphics*. Morgan Kaufmann Publishers, 1996.

33. SUZUKI, H., SAKURAI, Y., KANAI, T., AND KIMURA, F. Interactive mesh dragging with adaptive remeshing technique. In *Pacific Graphics '98 proceedings* (1998), pp. 188 – 197.

34. TAUBIN, G. A signal processing approach to fair surface design. In *Computer Graphics (SIGGRAPH 95 Proceedings)* (1995), pp. 351–358.

35. TAUBIN, G., GUÉZIEC, A., HORN, W., AND LAZARUS, F. Progressive forest split compression. In *Computer Graphics (SIGGRAPH 98 Proceedings)* (1998), pp. 123 – 132.

36. TAUBIN, G., AND ROSSIGNAC, J. Geometric Compression through Topological Surgery. *ACM Transaction on Graphics* (1998).

37. WELCH, W., AND WITKIN, A. Free–Form shape design using triangulated surfaces. In *Computer Graphics (SIGGRAPH 94 Proceedings)* (1994), A. Glassner, Ed., pp. 247–256.

38. ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. Interpolating subdivision for meshes with arbitrary topology. In *Computer Graphics (SIGGRAPH 96 Proceedings)* (1996), pp. 189–192.

39. ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. Interactive multiresolution mesh editing. In *Computer Graphics (SIGGRAPH 97 Proceedings)* (1997), pp. 259–268.

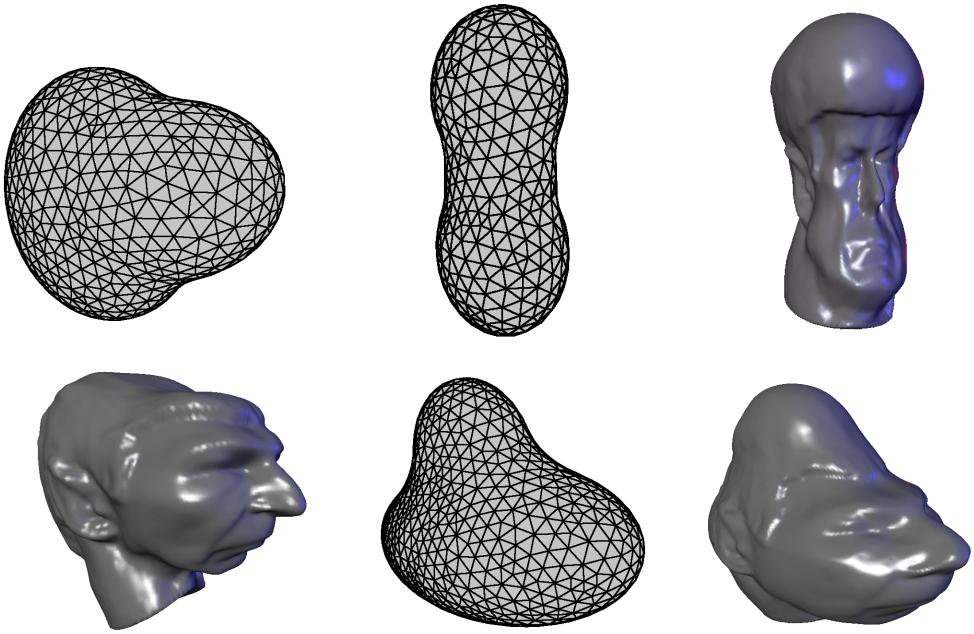

Figure 9: *Examples for multiresolution deformations. The multiresolution decomposition of the input data uses the well-known Spock-head model $\mathcal{M}_1$ and a sphere as its low-frequency approximation $\mathcal{M}_0$. We show the modified low-frequency geometries after applying a modeling operation based on control ellipsoids (wireframe) and the resulting reconstruction of the detail information (solid). The intuitive preservation of the detail is accomplished by constructing a transformation map T from the modified low-frequency geometry to the original mesh $\mathcal{M}_0$. This map is used for resampling the displacement function λ.*

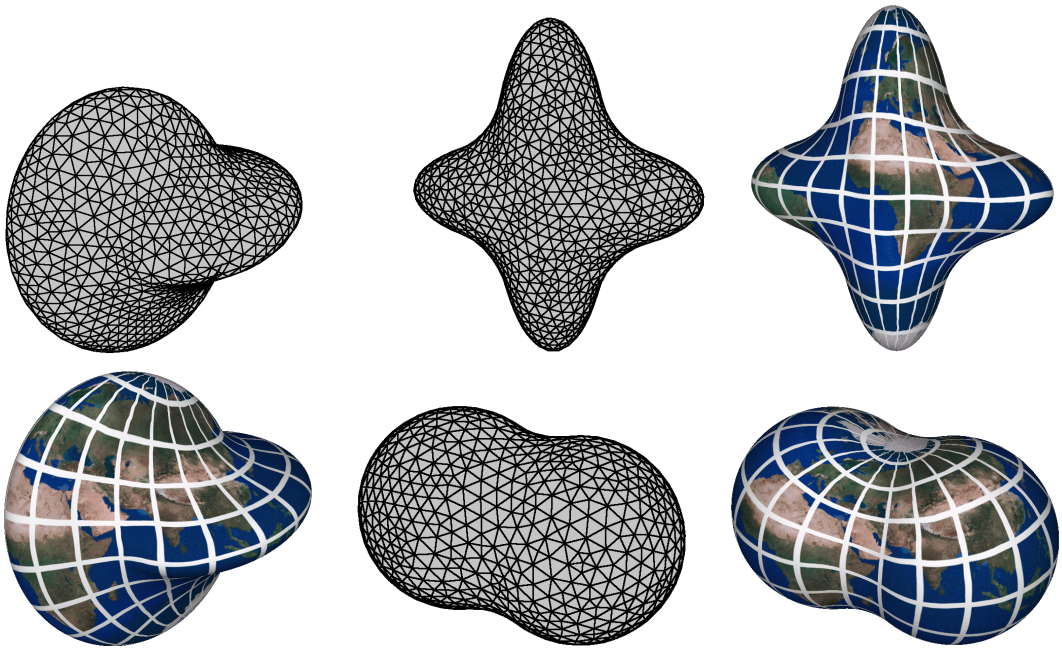

Figure 10: *A textured sphere is deformed by the control ellipsoid technique. Again we show the deformed models together with their underlying triangle mesh whose dynamic connectivity guarantees high quality in terms of the triangular faces' aspect ratio. We super-imposed a longitude/latitude system in order to demonstrate the small local distortion of the texture. This is due to the construction of the map T which associates points on the modified surface with points on the original sphere. A colored version of this figure can be found in the color section.*