PROCEEDINGS

Edited by
D. LEVIATAN
School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel

Contents

Contributors

T. Lyche
Dept. of Informatics, University of Oslo, P.O. Box 1080 Blindern, 0316 Oslo, Norway

K. Mørken
Dept. of Informatics, University of Oslo, P.O. Box 1080 Blindern, 0316 Oslo, Norway

E. Quak
SINTEF Applied Mathematics, P.O. Box 124 Blindern, 0314 Oslo, Norway

A. Cohen
Laboratoire d'Analyse Numérique, Université Pierre et Marie Curie, Paris, France

P. Schröder
Department of Computer Science, California Institute of Technology, Pasadena, CA 91125, USA

1

Subdivision, multiresolution and the construction of scalable algorithms in computer graphics

P. SCHRÖDER

Department of Computer Science
California Institute of Technology
Pasadena, CA 91125
USA

Abstract

Multiresolution representations are a critical tool in addressing complexity issues (time and memory) for the large scenes typically found in computer graphics applications. Many of these techniques are based on classical subdivision techniques and their generalizations. In this paper we review two exemplary applications from this area, multiresolution surface editing and semi-regular remeshing. The former is directed towards building algorithms which are fast enough for interactive manipulation of complex surfaces of arbitrary topology. The latter is concerned with constructing smooth parameterizations for arbitrary topology surfaces as they typically arise from 3D scanning techniques. Remeshing such surfaces then allows the use of classical subdivision ideas. We focus in particular on the practical aspects of making the well understood mathematical machinery applicable and accessible to the very general settings encountered in practice.

1.1 Introduction

Many applications in computer graphics are dominated by the need to deal efficiently with large datasets. This need arises from the goal of providing the user with a responsive system (ideally) allowing interactive work, i.e., update rates of several frames per second. However, compelling or realistic datasets such as topographical maps or geometric models appearing in entertainment and engineering tend to have large amounts of geometric detail. One measure of the latter is the number of triangles or patches necessary to provide a good approximation of a given complex surface. Not only must these datasets be displayed and transmitted rapidly, but in many cases expensive numerical computations must be performed on this data. Examples include surface denoising, fairing, compression, editing, optimization, and finite element (FEM) computations. The target machines for such computations are typically small workstations or home PCs with hardware accelerated graphics adapters.

These needs provide the motivation to search for efficient representations and algorithms which perform a variety of discrete and continuous computations. Some of the properties these representations and algorithms should satisfy are

- **Scalability:** their time and space complexity should ideally be linear in the input size. Conversely algorithms which require quadratic space or time are generally impractical as input sizes reach into the tens of thousands or more primitives. Note that scalability can also mean the ability to take advantage of large parallel computing resources. We will not discuss this issue further as our attention is directed towards small, widely available computing platforms.
- **Speed/Fidelity tradeoffs:** to provide the user with an interactive experience, algorithms which are capable of providing approximate answers fast, as well as more precise answers with additional time (and storage) are preferred over algorithms which always provide highly accurate answers after long computing times. For example, this requirement favors iterative solvers over direct solvers, or wavelet methods which provide "control knobs" for approximation quality versus computational time and storage.
- **Intuitive control:** most applications in computer graphics involve a possibly naive user, thus the parameters exposed by the system must be intuitive and should not require a deep understanding of the underlying representations and algorithms. For example, it is very difficult to manipulate Fourier coefficients directly to achieve a desired effect, while locally supported functions such as B-splines provide an easy to grasp relationship between cause and effect.
- **Robustness:** the larger the input dataset the higher the probability that all manner of "screw cases" will appear in a given dataset. Algorithms must handle these in a robust fashion and ideally be insenstive to them.
- **Uniformity:** a system which attempts to deal with many different representations concurrently tends to be more complicated and fragile than one which is based on a single, and very general representation.

One general concept which is now widely accepted as fundamental to the construction of scalable algorithms is the idea of multiresolution. In the context of computer graphics multiresolution comes in two fundamental flavors: (a) constructions which are based on classical notions of multiresolution as

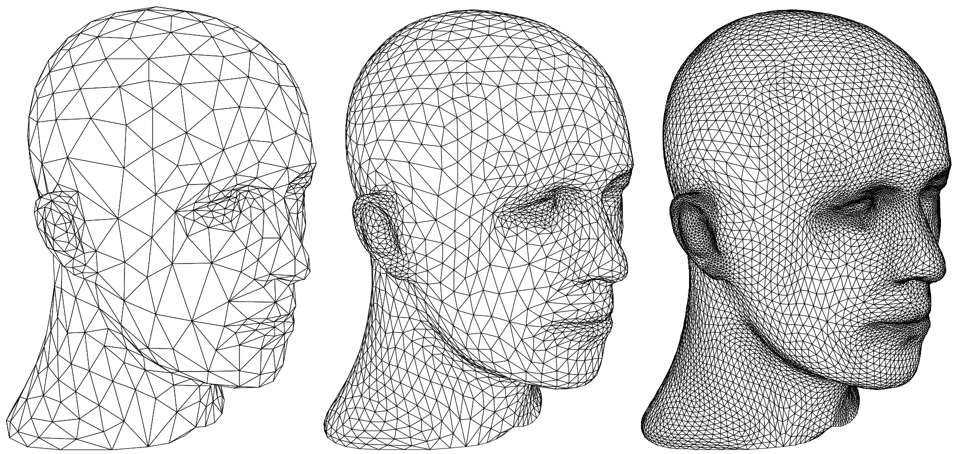

Fig. 1.1. Multiresolution as it is defined through subdivision. Starting with a coarsest level mesh of arbitrary connectivity successive levels of a multiresolution representation are produced through subdivision. (Original dataset courtesy Hugues Hoppe.)

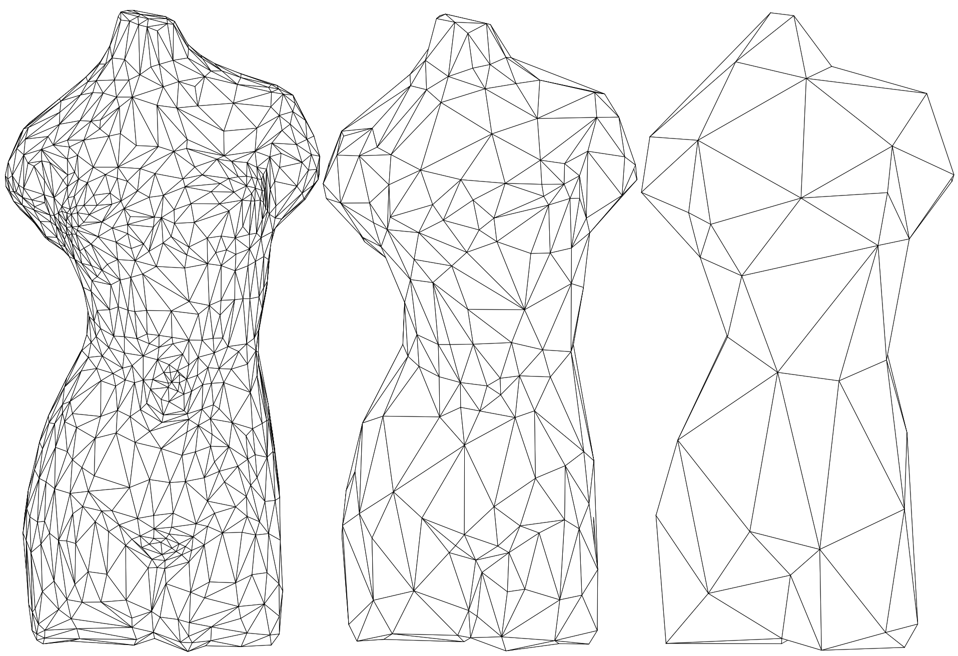

Fig. 1.2. Multiresolution as it is defined through mesh simplification. Starting with a finest level mesh of arbitrary connectivity, successive levels of a multiresolution representation are produced through simplification. (Original dataset courtesy Avalon.)

they appear in wavelets and subdivision; and (b) constructions based on mesh simplification (see Figures 1.1 and 1.2 respectively).

In the following we study two representative examples of these ideas. The first concerns the construction of a multiresolution surface editor which allows the manipulation of intricate, arbitrary topology surfaces at interactive rates on low end platforms. It uses classical subdivision surfaces (Loop) and extends them with the help of a Burt-Adelson type pyramid to a true multiresolution setting, i.e., a setting with non-trivial details at finer resolutions. The second application concerns the construction of smooth parameterizations for arbitrary connectivity meshes. Today, meshes that describe detailed geometric objects are typically built through 3D scanning. Before these meshes become amenable to a number of numerical algorithms they must be provided with a smooth parameterization. In particular this is the first step in transforming ("remeshing") them into a setting suitable for subdivision based multiresolution approaches.

The work discussed in the following was performed jointly with my (former) student Denis Zorin and long term collaborator Wim Sweldens. David Dobkin and his student Aaron Lee participated in this work as well and I am grateful to all of them.

1.2 Interactive Multiresolution Surface Editing

Applications such as industrial and engineering design as well as special effects and animation require creation and manipulation of intricate geometric models of arbitrary global topology (genus, number of boundaries, and connected components). Like real world geometry, these models can carry detail at many scales (see Figure 1.3). Such models are often created with 3D range sensing devices such as laser scanners (see for example the work of Curless and Levoy [6]), and the resulting meshes can be composed of as many as a million or more triangles. Manipulating such fine meshes is difficult, especially when they are to be edited or animated. Even without accounting for any computation on the mesh itself, available rendering resources alone, may not be able to cope with the sheer size of the data.

A popular approach to address the complexity challenge posed by such geometry is based on polygonal simplification [23]. These approaches aim to produce a lower resolution polygonal model which closely approximates the original geometry based on various error measures, such as Haussdorf distance or Sobolev norms. While these methods can produce approximations using fewer polygons ultimately a *smooth* rather than piecewise linear representations is desired. For example, a set of spline patches which approx-

Fig. 1.3. Before the Armadillo started working out he was flabby, complete with a double chin. Now he exercises regularly. The original is on the right (courtesy Venkat Krischnamurthy, Stanford University). The edited version on the left illustrates large scale edits, such as his belly, and smaller scale edits such as his double chin; all edits were performed at about 5 frames per second on an Indigo R10000 Solid Impact.

imates the original geometry. Finding such approximations is made difficult by the fact that we are dealing with arbitrary topology initial meshes. In this setting parametric methods are not straightforward to apply, since a-priori no simple parameterization is given and certainly none over a convenient domain such as the unit square.

Hoppe et al. [26] have described an algorithm which can produce a set of (piecewise) smooth patches in the arbitrary topology setting using subdivision. However, such an approximation is typically associated with a loss of high frequency details or suffers from a very large number of patches to maintain high accuracy in the presence of fine scale detail. Lost detail can be reintroduced by combining patches with displacement maps [31]. Unfortunately such hybrid representations are difficult to manage in the arbitrary topology setting.

Certainly, having a compact representation is useful when considering storage and transmission costs, but we seek more. It is desirable to work with representations which also support editing. Effective editing strategies must account for the fact that a designer will typically desire to make coarse smooth changes with the same ease as detailed changes to the geometry. This requires some form of multiresolution. In the traditional spline patch based editing setting such editing semantics were first supported by hierarchical splines (H-splines) proposed by Forsey and Bartels [18]. H-splines are constructed from regular splines (typically bi-cubic) by adding finer resolution B-splines onto an existing coarser resolution spline patch. By repeating this process, one can build very complicated shapes which are entirely parameterized over the unit square. A critical observation first made by Forsey and Bartels concerns the parameterization of the finer level basis functions. The most natural choice, using a global coordinate frame, leads to non-intuitive behavior of the surface during editing. Instead Forsey and Bartels suggested to parameterize the details with respect to coordinate frames induced by the coarser level geometry, i.e., through its partial derivatives and normal. Even though the resulting representations now become non-linear they are preferable since they exhibit the right editing behavior. A feature attached to the local surface normal will "travel" in a natural way with the surface as the coarse shape of the surface is adjusted (see Figure 1.4).

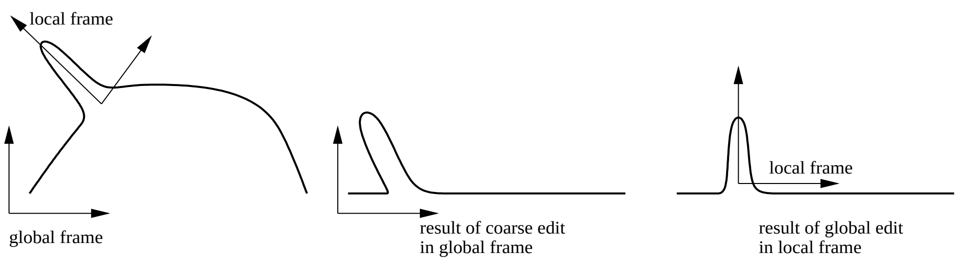

Fig. 1.4. Finer level details should be parameterized with respect to a local frame induced by the coarser level surface to achieve intuitive editing behavior.

Forsey and Bartels' original work focused on the ab initio design setting, i.e., building surfaces from "scratch." In this setting the user's help is enlisted in defining what is meant by different levels of resolution. The user decides where to add detail and manipulates the corresponding controls. To edit an a priori given model it is important to have a general procedure to define coarser levels and compute details between levels. We refer to this as the *analysis* algorithm in analogy to the wavelet setting. An H-spline analysis algorithm based on weighted least squares was introduced [17], but is too expensive to run interactively. Note that even in an ab initio design setting analysis is needed since a long sequence of editing steps often leads to overly refined H-splines which need to be consolidated for the sake of performance.

One particular avenue to make these notions of hierarchy, multiresolution approximation, and projection of fine scale geometry into coarse scales rigorous is through the use of wavelets. Finkelstein and Salesin [16], for example, used B-spline wavelets to describe multiresolution editing of curves. As in H-splines, parameterization of details with respect to a coordinate frame induced by the coarser level approximation is required to get correct editing semantics. Gortler and Cohen [20] pointed out that wavelet representations of details tend to behave in undesirable ways during editing and returned to a pure B-spline representation as used in H-splines. In general we find that critically sampled representations tend to break down under editing operations. One way to understand this is to consider the fact that in a critically sampled wavelet setting there exists a very carefully tuned balance between the low pass (scaling functions) and high pass (wavelets) filters. One cancels the aliases of the other. When a curve is edited, however, this balance is disturbed in a non-linear fashion because of the deformation of the underlying frames which are attached to the curve. As a consequence one observes in practice that the alias cancellation property is lost and undesirable "wiggles" appear in the geometry. These problems can be remedied through the use of overrepresentations such as Burt-Adelson [2] type pyramids (see Section 1.4).

All the approaches mentioned above rely on a setting which admits parameterization over a trivial domain such as the unit interval (curve editing) or the unit square. Carrying these ideas over into the arbitrary topology surface framework is not straightforward. The first to do so were Lounsbery et al. [35] who exploited the connection between wavelets and subdivision to define different levels of resolution for arbitrary topology geometry. While subdivision provides the scaling functions, corresponding wavelets can be constructed with the help of the Lifting Scheme [41, 42, 38]. The original constructions were limited to polyhedral wavelets, i.e., those constructed with the help of interpolating scaling functions such as linear splines or the limit functions of the (modified) Butterfly subdivision scheme [13, 46]. The latter is not as suitable for editing purposes because of negative weights in the subdivision masks. During editing operations, when one adjusts ("pulls on") a control point undesired ripples will appear on the surface.

1.2.1 Goals

We are seeking a geometry representation which unifies the advantages of the approaches discussed above and remedies their shortcomings. It should address the arbitary topology challenge, support hierarchical editing semantics and span the spectrum from patches to finely detailed triangle meshes. Subdivision surfaces based on generalizations of spline patches, combined with detail based extensions as in the H-spline approach provide such a representation. They provide a path from patches to fine meshes using a single primitive (hierarchical triangle mesh), and support the patch-type semantics of manipulation *and* finest level detail polyhedral edits equally well. Because they are the limit of successive refinement, subdivision surfaces support multiresolution algorithms, such as level-of-detail rendering, compression, wavelets, and numerical multigrid in a natural way. The basic computational kernels are simple, resulting in easy to implement and efficient codes. Boundaries and features such as creases can also be included through modified rules [26, 39, 32, 9], reducing the need for trimming curves. Even though the surfaces are of arbitrary topology they possess simple parameterizations and efficient algorithms exist for exact evaluation of values and derivatives of the limit surface at arbitrary parameter locations [40]. The main challenge is to make the basic algorithms fast enough to escape the exponential time and space growth of naive subdivision.

1.2.2 Notation

Before beginning with the description of the details of our algorithms we fix some notation. We need enough machinery to talk about meshes both in terms of their topology and geometry, i.e., their embedding in $\mathbf{R}^3$. For this purpose it is useful to carefully distinguish between the topological and geometric information respectively. We denote a triangle mesh as a pair $(\mathcal{P}, \mathcal{K})$, where $\mathcal{P}$ is a set of N point positions $p_a = (x_a, y_a, z_a) \in \mathbf{R}^3$ with $1 \leq a \leq N$, and $\mathcal{K}$ is an *abstract simplicial complex* which contains all the topological, i.e., adjacency information. The complex $\mathcal{K}$ is a set of subsets of $\{1, \ldots, N\}$. These subsets are called simplices and come in 3 types: vertices $v = \{a\} \in V \subset \mathcal{K}$, edges $e = \{a, b\} \in E \subset \mathcal{K}$, and faces $f = \{a, b, c\} \in T \subset \mathcal{K}$, so that any non-empty subset of a simplex of $\mathcal{K}$ is again a simplex of $\mathcal{K}$, e.g., if a face is present so are its edges and vertices.

Let δ_i denote the standard i-th basis vector in $\mathbf{R}^N$. For each simplex s, its *topological realization* $|s|$ is the strictly convex hull of $\{\delta_i \mid i \in s\}$. Thus $|\{a\}| = \delta_a$, $|\{a, b\}|$ is the open line segment between δ_a and δ_b, and $|\{a, b, c\}|$ is an open equilateral triangle. The topological realization $|\mathcal{K}|$ is defined as $\cup_{s \in \mathcal{K}} |s|$. The *geometric realization* $\varphi(|\mathcal{K}|)$ relies on a linear map $\varphi : \mathbf{R}^N \to \mathbf{R}^3$ defined by $\varphi(\delta_a) = p_a$. Abusing notation a bit we will abbreviate the geometric realization by writing $p_a = p(a)$ below. The resulting polyhedron consists of points, segments, and triangles in $\mathbf{R}^3$.

Two vertices $\{a\}$ and $\{b\}$ are *neighbors* if $\{a, b\} \in E$. A set of vertices is *independent* if no two vertices are neighbors. A set of vertices is *maximally independent* if no larger independent set contains it (see Figure 1.18, left side). The 1-ring neighborhood of a vertex $\{a\}$ is the set

$$\mathcal{N}(a) = \{b \mid \{a, b\} \in E\}.$$

The *outdegree* K_a of a vertex is its number of neighbors. The *star* of a vertex $\{a\}$ is the set of simplices

$$\text{star}\,(a) = \bigcup_{a \in s,\, s \in \mathcal{K}} s.$$

We say that $|\mathcal{K}|$ is a two dimensional manifold (or 2-manifold) with boundaries if for each a, $|\text{star}\,(a)|$ is homeomorphic to a disk (interior vertex) or half-disk (boundary vertex) in $\mathbf{R}^2$. An edge $e = \{a, b\}$ is called a *boundary edge* if there is only one face f with $e \subset f$.

1.3 Subdivision

For our purposes subdivision has two distinct components, one given by topological operations on the simplicial complex and the other by operations on the geometric realization of the mesh, i.e., the association of a *point* in 3D with every *vertex* in the complex. Starting with an initial mesh $(\mathcal{P}^0, \mathcal{K}^0 = (V^0, E^0, T^0))$ subdivision builds a new mesh $(\mathcal{P}^1, \mathcal{K}^1)$ by refining each face $\{a, b, c\} \in T^0$ into four faces $\{a, c_e, b_e\}$, $\{b, a_e, c_e\}$, $\{c, b_e, a_e\}$, $\{a_e, b_e, c_e\} \in T^1$. The new vertices $\{a_e\}, \{b_e\}, \{c_e\} \in V^1$ can be thought of as children of the edges across from the vertices $\{a\}, \{b\}, \{c\} \in V^0$ respectively. Continuing this construction recursively results in meshes $(\mathcal{P}^i, \mathcal{K}^i)$. The superscript i indicates the *level* of triangles and vertices respectively. The vertex sets are nested as $V^j \subset V^i$ if $j < i$. We define *odd* vertices on level i as $M^i = V^{i+1} \setminus V^i$. V^{i+1} consists of two disjoint sets: *even* vertices (V^i) and *odd* vertices (M^i). We define the *level* of a vertex a as the smallest i for which $a \in V^i$. The level of a is $i + 1$ if and only if $a \in M^i$.

With each set V^i we associate a geometric realization $p^i(a) \in \mathbf{R}^3$, for all $a \in V^i$. The set p^i contains all points on level i, $p^i = \{p^i(a) \mid a \in V^i\}$. Finally, a *subdivision scheme* is a linear operator S which takes the points from level i to points on the *finer* level $i + 1$: $p^{i+1} = S\, p^i$, defining the geometric realization at level $i + 1$.

Assuming that the subdivision converges, we can define a limit surface σ as

$$\sigma = \lim_{k \to \infty} S^k\, p^0.$$

$\sigma(a) \in \mathbf{R}^3$ denotes the point on the limit surface associated with vertex a.

In order to later define detail offsets with respect to a local frame we also need tangent vectors and a normal. For the subdivision schemes that we use, such vectors can be defined through the application of linear operators D^1 and D^2 acting on p^i so that $\partial_1^i(a) = (D^1 p^i)(a)$ and $\partial_2^i(a) = (D^2 p^i)(a)$ are linearly independent tangent vectors at $\sigma(a)$. Together with an orientation they define a local orthonormal frame $F^i(a) = (n^i(a), \partial_1^i(a), \partial_2^i(a))$. For reasons of efficiency, it is important to note that in general it is not necessary to use precise normals and tangents during editing; as long as the frame vectors are affinely related to the positions of vertices of the mesh, we can expect intuitive editing behavior.

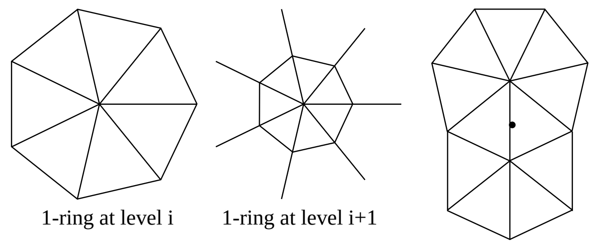

Fig. 1.5. An even vertex has a 1-ring of neighbors at each level of refinement (left/middle). Odd vertices—in the middle of edges—have 1-rings around each of the vertices at either end of their edge (right).

We are particularly interested in subdivision schemes which belong to the class of *1-ring schemes*. In these schemes points at level $i+1$ depend only on 1-ring neighborhoods of points at level i. Let $a \in V^i$ (a even) then the point $p^{i+1}(a)$ is a function of $p^i(a_n)$ for $a_n \in \mathcal{N}^i(a)$, i.e., only the immediate neighbors of a (see Figure 1.5 left/middle). If $m \in M^i$ (m odd), it is the vertex inserted when splitting an edge of the mesh. In this case the point $p^{i+1}(m)$ is a function of the 1-rings around the vertices at the ends of the edge (see Figure 1.5 right).

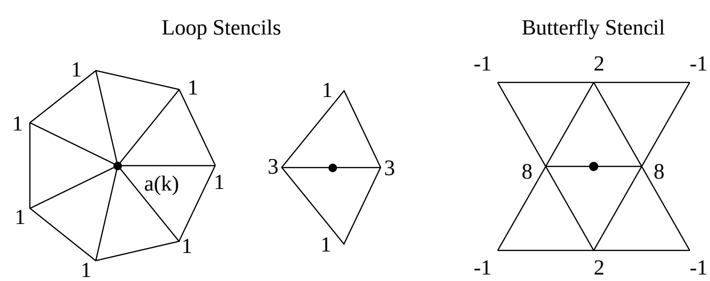

Fig. 1.6. Stencils for Loop subdivision with unnormalized weights for even and odd vertices (left) and for the regular Butterfly (right). Note that the Butterfly scheme, since it is interpolating, only has a stencil for odd vertices.

Two examples of such schemes are Loop's [33], which is an approximating scheme generalizing quartic box splines, and the (modified) Butterfly scheme [13, 46, 37], which is interpolating. The stencil weights for these schemes are shown in Figure 1.6.

Discussion There are many possible choices of subdivision shemes to use for a surface editor, each of which has certain properties which make it more suitable for one task than another. Broadly, we can distinguish interpolating and approximating schemes, those based on triangles or quadrilaterals, as well as primal versus dual schemes. Interpolating schemes such as the Butterfly scheme [13, 46] (triangles) or Kobbelt's quad scheme [29] have the advantage that any vertex has only one associated point, which is a sample of the limit surface. This simplifies the implementation and makes adaptive subdivision particularly easy. For example, it suffices to place a simple restriction criterion on the quad trees that hold the point positions. A disadvantage of interpolating schemes arises from the fact that the fundamental solutions of the subdivision process by necessity have negative lobes. As a consequence the surface develops oscillations when a control point is moved. In general, interpolating schemes tend to produce surfaces which lack fairness and can respond in non-intuitive ways when a user manipulates the control points. An exception are variational subdivision schemes [30] which are defined to produce fair surfaces that are also interpolating. Unfortunately they are computationally intensive, as each subdivision step requires the inversion of a (sparse) system of linear equations. Additionally it is not clear how to support adaptive subdivision in this setting.

In contrast approximating schemes, which are largely based on generalizations of splines, such as Loop [33], Catmull-Clark [3], and Doo-Sabin [11] tend to result in much nicer editing behavior. Since the basis functions do not exhibit any undulations, pulling on a control point results in a smooth, bump like surface change. Implementation of these schemes is slightly more involved as each vertex now has to hold a number of point positions, one for every level of the subdivision process on which it exists. For an efficient implementation this requires dynamically adjustable storage at each vertex. Adaptive subdivision is still possible through the use of a restriction criterion, but care must be taken to reference the point position at the proper level. Alternatively each vertex can carry one extra point position which is a sample of the limit surface (easily computed through the application of an associated mask corresponding to the left eigenvector of the eigenvalue 1).

The choice between triangles and quadrilaterals as basic shape is mostly dictated by the application domain. Triangles are somewhat more general as primitives. For example, when drawing the surface any graphics rendering hardware will convert arbitrary polygons to triangles. On the other hand, many objects exhibit two major directions ("left-right/top-bottom," "north-south/west-east"), which favors quadrilaterals as basic primitives. If the shapes are to be used in finite element analysis one may prefer triangles or quadrilaterals, depending on considerations such as approximation properties.

Finally we point out the difference between primal and dual schemes. Consider the generalization of bi-quadratic splines in the form of Doo-Sabin's subdivision scheme (a similar argument applies to the schemes studied in [36]). When applying subdivision to the control mesh, each vertex at the coarser level becomes the parent of a face at the next finer level. Similarly each edge generates a face at the next finer level and so does each face. They are corner cutting schemes. We call these dual since they can be thought of as subdivision on the dual complex. Consider the complex of the coarser mesh. Take its dual and refine it by quadrisection, i.e., each quadrilateral face receives a new vertex, as do all edges. Finally, dualize the resulting complex again to arrive at the mesh of the finer level of the original scheme. Unfortunately this observation only applies to the topology, not the geometry. The resulting datastructures are awkward and it is not clear how to support adaptive subdivision in this framework. This is the main reason for our preference of primal schemes. The latter include Catmull-Clark and Loop.

Another consideration in implementations concerns the support size of the stencils and basis functions respectively. For example, the interpolating constructions mentioned above have larger support leading to larger support stencils when computing tangent vectors. Aside from performance considerations, larger stencils also imply many more special cases near the boundary and near extraordinary vertices, i.e., those with valence other than 6 (triangles) or 4 (quadrilaterals). All these considerations lead us to prefer Loop's scheme. It is based on triangles, the support for limit point evaluation or tangent evaluation is a 1-ring, the basis functions are non-negative, and adaptive subdivision is easily supported through a restriction criterion imposed on the quadtrees which hold the point positions of the control meshes at different levels of resolution. The latter is particularly critical in escaping the exponential time and memory growth of naive subdivision.

1.4 Multiresolution Extensions of Subdivision

Using subdivision alone one could easily build a very efficient geometric modeling system. The user manipulates the coarsest level control points and thus shapes the limit surface. Suitably high performance can be delivered through the use of adaptive subdivision, for example, allocating larger triangles in areas which are relatively flat and smaller triangles in areas of high curvature. Figure 1.7 shows an instance of such adaptation which is naturally supported by the recursive nature of subdivision. Simply stop a particular branch of the recursion when a local approximation criterion has been satisfied. These criteria can also be varied across the surface so as to allocate resources in regions of particular interest without compromising overall performance (see Figure 1.8).

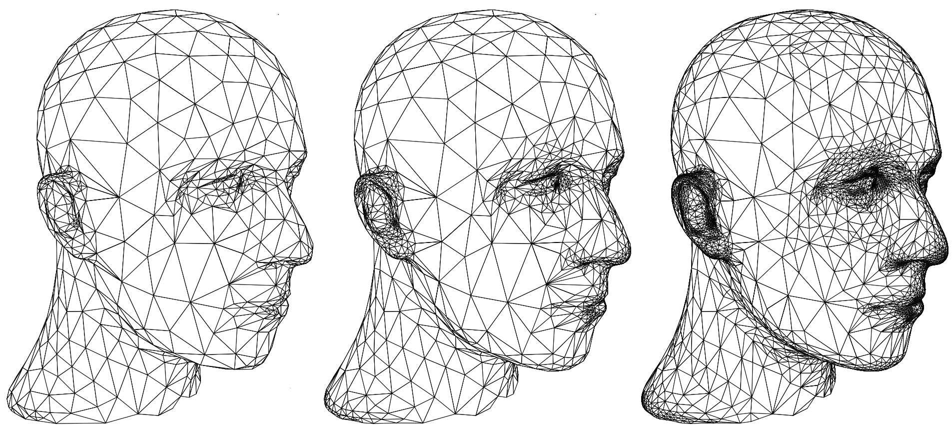

Fig. 1.7. Subdivision describes a smooth surface as the limit of a sequence of refined polyhedra. The meshes show several levels of an adaptive Loop surface.

The capabilities of such an editing system as presented to the user are very similar to what one would find in more traditional patch based modelers. The only difference being that the system can deal with arbitrary topology surfaces and not just those describable over a tensor product domain. What is missing is the ability to define detail at many different levels of a hierarchy. Figure 1.9 shows an example of the head of Armadillo man. Here the top level control points were subdivided four times, yielding the surface on the left side. It is very smooth as one would expect. What the designer would like is the ability to include details at finer levels. Conceptually this is particularly easy to accomplish in the subdivision setting. Simply allow the control mesh points to be perturbed by the addition of "detail" vectors after each subdivision step. The result of allowing such a perturbation are demonstrated in the middle and right picture in Figure 1.9. The middle picture shows the effect of allowing additional displacements to be added after the first subdivision step, while the right image shows the effect of allowing displacements to be added after the first subdivision step and once

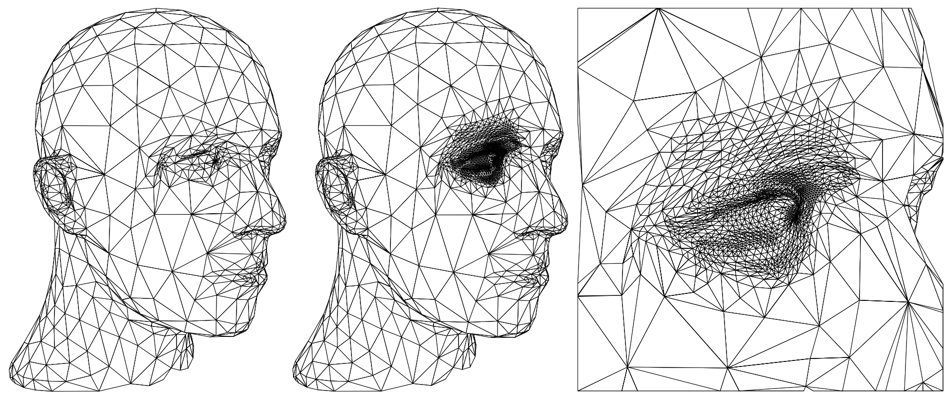

Fig. 1.8. It is easy to change the surface approximation quality criterion locally. Here a "lens" was applied to the right eye of the Mannequin head through a locally decreased approximation quality ϵ to force very fine resolution of the mesh around the eye. This example shows a pure subdivision surface with no multiresolution details.

more after the second subdivision step. In this way the introduction of detail in an "octave" like fashion of finer and finer scales becomes straightforward.

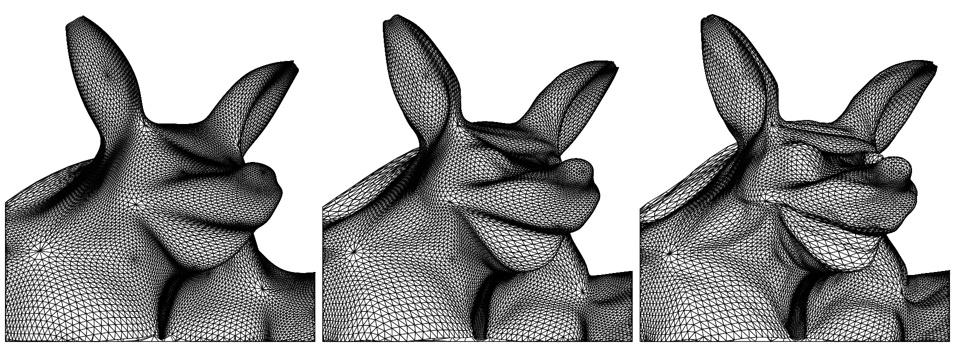

Fig. 1.9. Wireframe renderings of virtual surfaces representing the first three levels of control points.

To put these ideas on a solid basis we must discuss the analysis operator. Subdivision, or synthesis, goes from coarse to fine, while analysis goes from fine to coarse and computes the detail vectors. We first need *smoothing* and downsampling, i.e., a linear operation H to build a smooth coarse mesh at level $i-1$ from a fine mesh at level i:

$$p^{i-1} = H\, p^i.$$

Several options are available here:

- **Least squares:** One could define analysis to be optimal in the least squares sense,

$$\min_{p^{i-1}} \|p^i - S\,p^{i-1}\|^2.$$

  The solution may have unwanted undulations and is too expensive to compute interactively [17].
- **Fairing:** A coarse surface could be obtained as the solution to a global variational problem. This is too expensive as well. An alternative is presented by Taubin [43], who uses a *local* smoothing approach.

Because of its computational simplicity we decided to use a version of Taubin smoothing. As before let $a \in V^i$ have K neighbors $a_k \in V^i$. Use the average, $\overline{p}^i(v) = K^{-1} \sum_{k=1}^{K} p^i(a_k)$, to define the discrete Laplacian $\mathcal{L}(a) = \overline{p}^i(v) - p^i(a)$. On this basis Taubin gives a Gaussian-like smoother with reduced shrinkage problems

$$H := (I + \mu\,\mathcal{L})\,(I + \lambda\,\mathcal{L}),$$

with μ and λ tuned to ameliorate the shrinkage problem inherent in straight Gaussian smoothing [43]. With subdivision and smoothing in place, we can describe the transform needed to support multiresolution editing.

Recall that for multiresolution editing we want the difference between successive levels expressed with respect to a frame induced by the coarser level, i.e., the offsets are relative to the smoother level. With each vertex a and each level $i > 0$ we associate a *detail vector*, $d^i(a) \in \mathbf{R}^3$. The set d^i contains all detail vectors on level i, $d^i = \{d^i(a) \mid a \in V^i\}$. As indicated in Figure 1.10 the detail vectors are defined as

$$d^i = (F^i)^T\,(p^i - S\,p^{i-1}) = (F^i)^T\,(I - S\,H)\,p^i,$$

i.e., the detail vectors at level i record how much the points at level i differ from the result of subdividing the points at level $i-1$. This difference is then represented with respect to the local frame F^i to obtain coordinate independence.

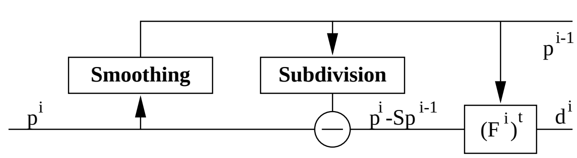

Fig. 1.10. Wiring diagram of the multiresolution transform.

Since detail vectors are sampled on the fine level mesh V^i, this transformation yields an overrepresentation in the spirit of the Burt-Adelson pyramid [2]. The only difference is that the smoothing filter (Taubin) is not the dual of the subdivision filter (Loop). Theoretically it would be possible to subsample the detail vectors and only record a detail per odd vertex (elements of M^{i-1}). This is what happens in the wavelet transform. However, subsampling the details severely restricts the family of smoothing operators that can be used and is not desirable in an editing environment in either case (see our comments in Section 1.2).

Figure 1.11 shows an example of analysis applied to a shape with a strong feature at a fine level. This feature yields details at coarser levels which are smoothed versions of the finer level feature.

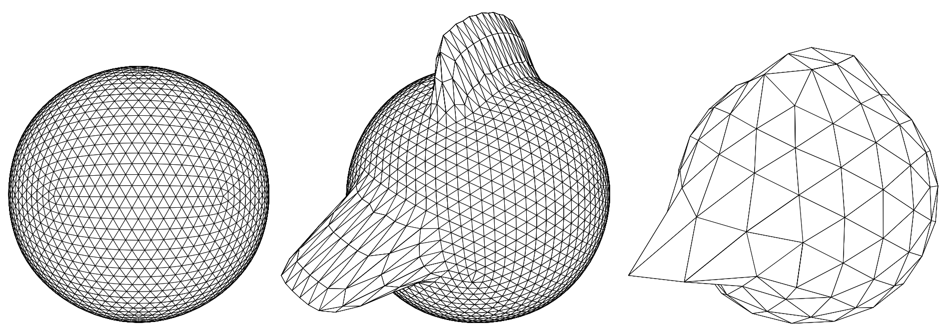

Fig. 1.11. Analysis propagates the changes on finer levels to coarser levels, keeping the magnitude of details under control. Left: The initial mesh. Center: A simple edit on level 3. Right: The effect of the edit on level 2. A significant part of the change was absorbed by higher level details.

Figure 1.12 shows two triangle mesh approximations of the Armadillo head with multiresolution details. Approximately the same number of triangles are used for the adaptive and uniform mesh. The adaptive rendering took local surface approximation quality and magnitudes of details at finer levels into account.

The performance of the system is good enough to allow the editing of intricate shapes such as Armadillo man on a PC. The Armadillo man has approximately 172,000 triangles on 6 levels of subdivision. Display list creation took 3 seconds on the PC for the full model. We adjusted the approximation criteria so that the model would render at 5 frames per second. On the PC this amounted to a model approximation with 35,000 triangles. Note that even though only an approximation is rendered on the screen, the entire model is maintained in memory. Thanks to the surface relative local

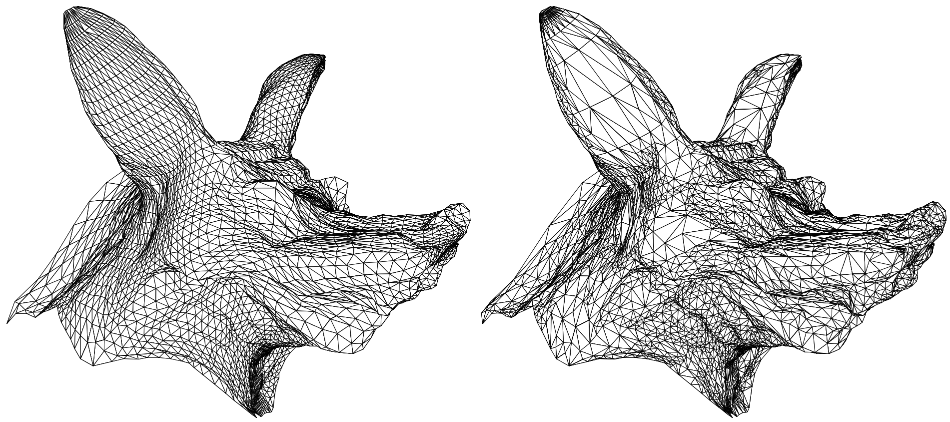

Fig. 1.12. On the left a uniformly subdivided mesh with approximately 11k triangles. On the right a mesh with the same number of triangles but subdivided non-uniformly for a better approximation within the same rendering budget.

frames for the detail vectors, any edits performed at coarse levels, even if fine levels are not displayed, are correctly reflected at fine levels. Figure 1.13 shows a coarse level edit in which control points associated with the tip of the ears are pulled, while Figure 1.14 shows a fine level edit in which the user moves control points at a fine subdivision level in the vicinity of the eye.

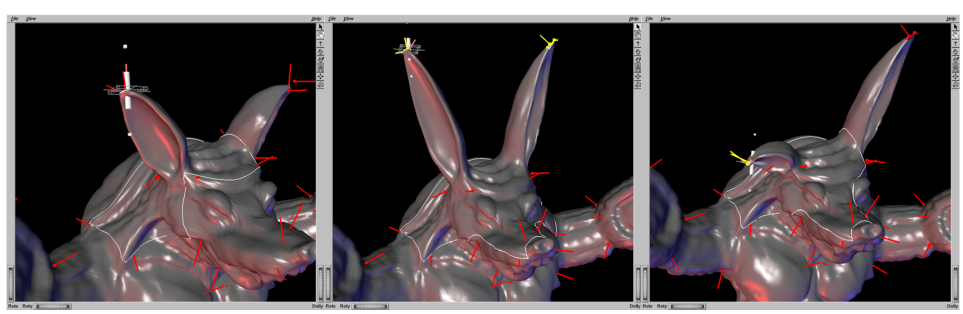

Fig. 1.13. Example of a coarse level edit in which a group of control points at the tip of the ears is pulled up and then subsequently only the control points at the tip of one ear are pushed down. Note that fine level detail on the shape moves along with the surface as one would intuitively expect.

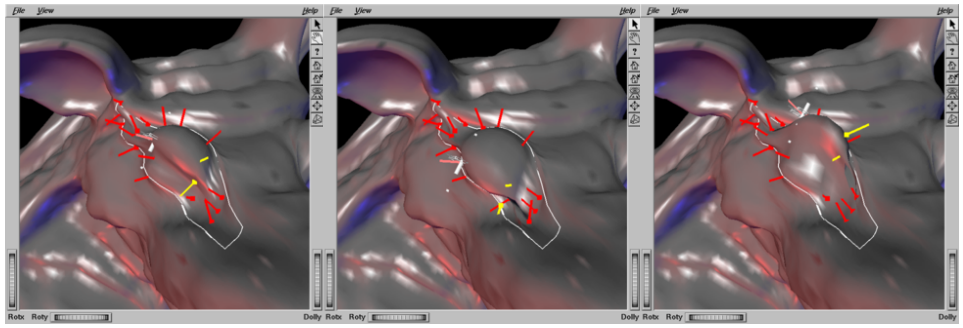

Fig. 1.14. Example of a fine level edit. Here control points at the third subdivision level are exposed and the user selects a group of them around the eye lid, affecting only a very small region around the eye.

1.5 Irregular Meshes

The approach presented so far relies on the special structure of the underlying meshes for its efficiency. We assumed that there exists a coarsest level mesh which is recursively subdivided. The resulting type of meshes have a *semi-regular* structure, i.e, all but the coarsest level vertices have valence 6. When geometry is built "from scratch" it is an easy matter to enforce this requirement. The situation changes when one wants to work with an arbitrary input mesh coming from some other application. For example, meshes coming from 3D acquisition devices typically are completely unstructured, or irregular. If we wish to apply multiresolution analysis and associated algorithms to these we must first convert them to semi-regular meshes, or *remesh* them. A first step in such a procedure, and for that matter in many other numerical algorithms which aim to deal with such meshes, is the establishment of a smooth parameterization. In a very fundamental sense we are seeking methods to build a proper manifold structure, i.e., charts and an atlas, for such settings. While an arbitrary triangle mesh can always be thought of as parameterized over itself this is not a useful observation in practice: a mesh which consists of hundreds of thousands of triangles is trivially parameterized over its combinatorial complex using the geometric realization φ. But this object is unwieldy and not smooth (piecewise linear only). What we seek instead is a parameterization of the original mesh over a homeomorphic complex with very few triangles. Once the original mesh, or a close approximation of it, is realized in this fashion many numerical modeling tasks are greatly facilitated. In particular it is then an easy matter to resample the original mesh onto a semi-regular mesh as it appears in traditional subdivision, allowing us to apply the algorithms described in the previous section.

A procedure to establish such a parameterization is the subject of this section. Figure 1.15 shows the outline of the procedure: beginning with an irregular input mesh (top left) in a first step a base domain is established through mesh simplification (top middle). Concurrent with simplification, a mapping is constructed which assigns every vertex from the original mesh to a base triangle (top right). Using this mapping an adaptive remesh with subdivision connectivity can be built (bottom left) which is now suitable for such applications as multiresolution editing (bottom middle).

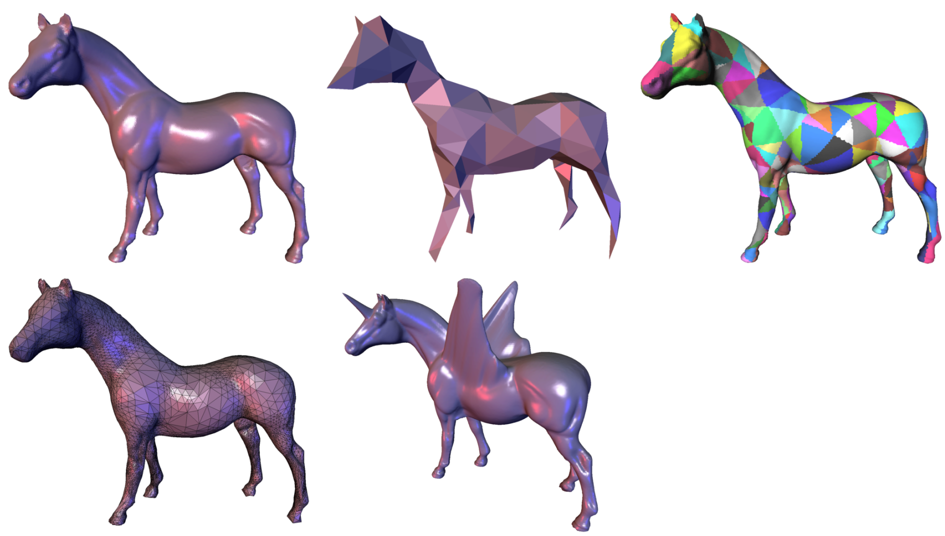

Fig. 1.15. Overview of the remeshing algorithm. Top left: a scanned input mesh (courtesy Cyberware). Next the parameter or base domain, obtained through mesh simplification. Top right: regions of the original mesh colored according to their assigned base domain triangle. Bottom left: adaptive remeshing with subdivision connectivity ($\epsilon = 1\%$). Bottom middle: multiresolution edit.

There are many possible ways to approach this problem and before describing the details of one such algorithm we first consider some other possible approaches. These fall into two broad categories, those that are geared towards approximating an initial unstructured mesh with smooth patches, and those which expicitly aim at constructing parameterizations for remeshing.

1.5.1 Approximation of a Given Set of Samples

Hoppe and co-workers [26] describe a fully automatic algorithm to approximate a given polyhedral mesh with (modified) Loop subdivision patches [33] respecting features such as edges and corners. Their algorithm uses a non-linear optimization procedure taking into account approximation error and the number of triangles of the base domain. The result is a smooth parameterization of the original polyhedral mesh with a (hopefully) small number of patches. Since the approach only uses subdivision, small features in the original mesh can only be resolved accurately by increasing the number of patches accordingly. ¿From the point of view of constructing parameterizations, the main drawback of algorithms in this class is that the number of triangles in the base domain depends heavily on the *geometric* complexity of the original surface.

This latter problem was addressed in work of Krishnamurthy and Levoy [31]. They approximate densely sampled geometry with bi-cubic spline patches and displacement maps. Arguing that a fully automatic system cannot put iso-parameter lines where a skilled modeler or animator would want them, they require the user to lay out the entire network of top level spline patch boundaries. A coarse to fine matching procedure with relaxation is used to arrive at a high quality patch based surface whose base domain need not mimic small scale geometric features. Any fine level features are captured by allowing the addition of a displacement map on top of the smooth spline patches. The principal drawback of their procedure is that the user is required to define the *entire* base domain rather then only selected features. Additionally, given that the procedure works from coarse to fine, it is possible for the procedure to "latch" onto the wrong part of the surface in regions of high curvature [31, Figure 7].

1.5.2 Remeshing

Lounsbery and co-workers [34, 35] were the first to propose algorithms to extend classical multiresolution analysis to arbitrary topology surfaces. Because of its connection to the mathematical foundations of wavelets, this approach has proven very attractive [38, 14, 46, 15, 4, 47]. The algorithm described in the first half of this paper belongs to this class. Since the central requirement of these methods is that the input mesh have subdivision connectivity remeshing is generally required to apply these approaches. Eck and co-workers [14] were the first to develop such a remeshing algorithm, computing a smooth parameterization of high resolution triangle meshes over a low face count base domain, followed by a semi-regular resampling step. After this conversion step, adaptive simplification, compression, progressive transmission, rendering, and editing become simple and efficient operations [4, 15, 47].

Eck et al. arrive at the base domain through a Voronoi tiling of the original mesh. Using a sequence of local harmonic maps, a parameterization which is smooth over each triangle in the base domain and which meets with C^0 continuity at base domain edges [14, Plate 1(f)] is constructed. Runtimes for the algorithm can be long because of the many harmonic map computations. This problem was recently addressed by Duchamp and co-workers [12], who reduced the harmonic map computations from their initial $O(N^2)$ complexity to $O(N \log N)$ through hierarchical preconditioning.† The initial Voronoi tile construction relies on a number of heuristics which render the overall algorithm fragile (for an improved version see [28]). Moreover, there is no explicit control over the number of triangles in the base domain or the placement of patch boundaries. The latter is critical in applications as one wants to avoid unneccessary subdivision near sharp features.

1.5.3 An Alternative Parameterization Algorithm

We describe here an algorithm which was designed to overcome the drawbacks of previous work as well as to introduce new features. A fast coarsification strategy is used to define the base domain, avoiding the potential difficulties of finding Voronoi tiles [14, 28]. Since the algorithm proceeds from fine to coarse, correspondence problems found in coarse to fine strategies [31] are avoided, and all features are correctly resolved. Piecewise linear approximations to conformal maps are used during coarsification to produce a global parameterization of the original mesh. This map is then further improved through the use of a hierarchical Loop smoothing procedure obviating the need for iterative numerical solvers [14]. Since the procedure is performed globally, derivative discontinuities at the edges of the base domain are avoided [14]. In contrast to fully automatic methods [14], the algorithm supports vertex and edge tags [26] to constrain the final parameterization to align with selected features; however, the user is not required to specify the entire patch network [31]. During remeshing we take advantage of the original fine to coarse simplification hierarchy to output a sparse, adaptive, semi-regular mesh directly without resorting to a depth first oracle [38] or the need to produce a uniform subdivision connectivity mesh at exponential cost followed by wavelet thresholding [4].

† The hierarchy construction they employed for use in a multigrid solver is closely related to the hierarchy construction described below.

1.6 Hierarchical Surface Representation

An important part of the algorithm is the construction of a mesh hierarchy. The original mesh $(\mathcal{P}, \mathcal{K}) = (\mathcal{P}^L, \mathcal{K}^L)$ is successively simplified into a series of homeomorphic meshes $(\mathcal{P}^l, \mathcal{K}^l)$ with $0 \leq l < L$, where $(\mathcal{P}^0, \mathcal{K}^0)$ is the coarsest or base mesh (see Figure 1.16). Several approaches for such mesh simplification have been proposed, most notably progressive meshes (PM) [24]. In PM the basic operation is the "edge collapse." A sequence of such atomic operations is prioritized based on approximation error. The linear sequence of edge collapses can be partially ordered based on topological dependence [44, 25], which defines levels in a hierarchy. The depth of these hierarchies appears "reasonable" in practice, though can vary considerably for the same dataset [25].

Our approach is similar in spirit, but inspired by the hierarchy proposed by Dobkin and Kirkpatrick (DK) [10], which guarantees that the number of levels L is $O(\log N)$. While the original DK hierarchy is built for convex meshes, we show how the idea behind DK can be used for general meshes. The DK atomic simplification step is a *vertex remove*, followed by a retriangulation of the hole. The two basic operations "vertex remove" and "edge collapse" are related since an edge collapse into one of its endpoints corresponds to a vertex removal with a particular retriangulation of the resulting hole (see Figure 1.17). The main reason to choose an algorithm based on the ideas of the DK hierarchy is that it guarantees a logarithmic bound on the number of levels.

1.6.1 Vertex Removal

One DK simplification step $\mathcal{K}^l \to \mathcal{K}^{l-1}$ consists of removing a maximally independent set of vertices with low outdegree (see Figure 1.18). Recall that an independent set of vertices has the property that no two vertices in the set are connected by an edge. To find such a set, the original DK algorithm used a greedy approach based only on *topological* information. Instead it is desirable to take geometric information into account as well. This can be done by selecting candidates for the independent set based on a priority queue.

At the start of each level of the DK algorithm, none of the vertices are marked and the set to be removed is empty. The algorithm randomly selects a non-marked vertex of outdegree less than 12, removes it and its star from $\mathcal{K}^l$, marks its neighbors as unremovable and iterates this until no further vertices can be removed. In a triangulated surface the average outdegree of a vertex is 6. Consequently, no more than half of the vertices can be

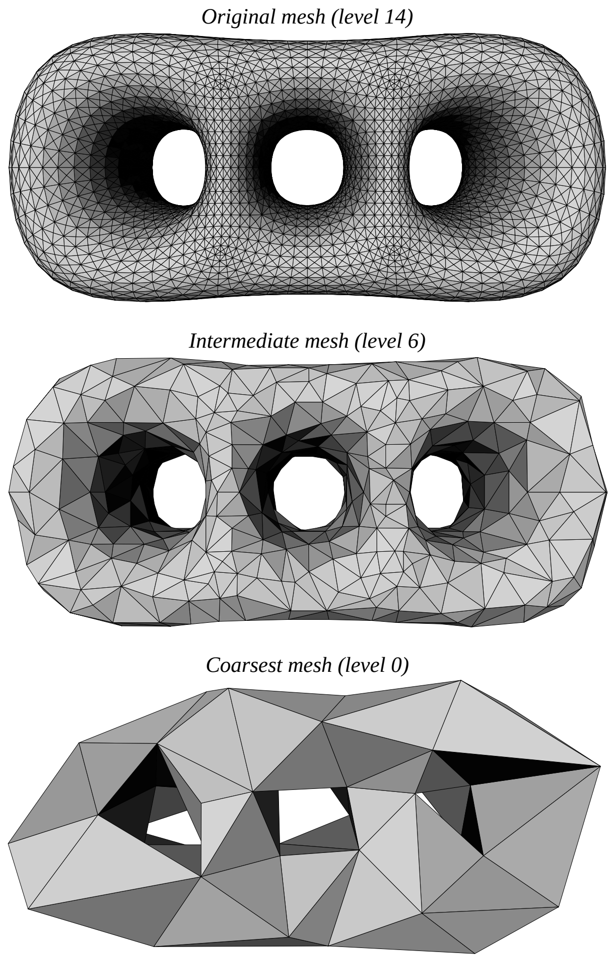

Fig. 1.16. Example of a modified DK mesh hierarchy. At the top the finest (original) mesh $\varphi(|\mathcal{K}^L|)$ followed by an intermediate mesh, and the coarsest (base) mesh $\varphi(|\mathcal{K}^0|)$ at the bottom (original dataset courtesy University of Washington).

of outdegree 12 or more. Thus it is guaranteed that at least $1/24$ of the vertices will be removed at each level [10]. In practice, it turns out one can remove roughly $1/4$ of the vertices reflecting the fact that the graph is four-colorable. In any case, given that a constant fraction can be removed on each level, the number of levels behaves as $O(\log N)$. The entire hierarchy can thus be constructed in linear time. A better approach is to stay in the DK framework, but to replace the random selection of vertices by a priority queue based on geometric information. Roughly speaking, vertices with small and flat 1-ring neighborhoods should be chosen first. Note that

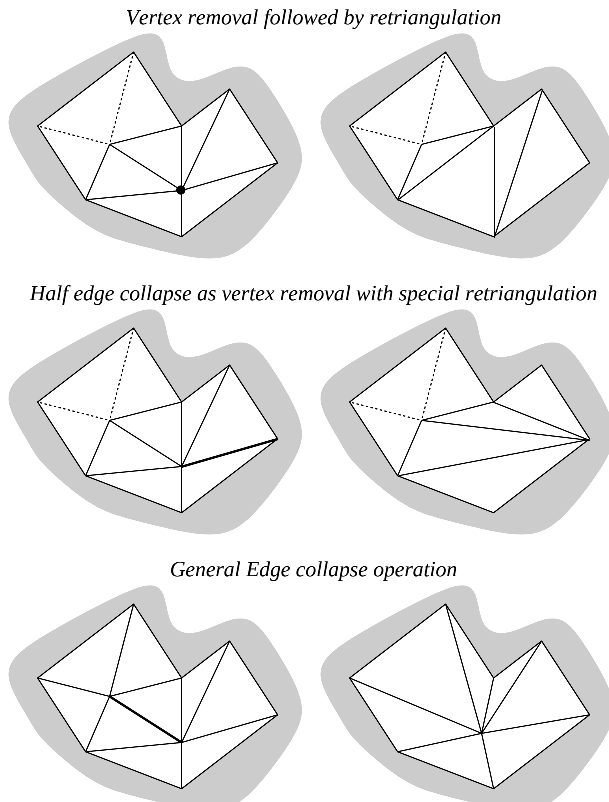

Fig. 1.17. Examples of different atomic mesh simplification steps. At the top vertex removal, in the middle half-edge collapse, and edge collapse at the bottom.

the complexity of the overall construction grows to $O(N \log N)$ because of the priority queue.

Figure 1.16 shows three stages (original, intermediary, coarsest) of such a DK hierarchy. The main observation is that the coarsest mesh can be used as the domain of a parameterization of the original mesh since all meshes in the DK hierarchy are homeomorphic. The construction of these homeomorphisms is the subject of the next section.

1.6.2 Flattening and Retriangulation

To find $\mathcal{K}^{l-1}$, we need to retriangulate the holes left by removing the independent set. One possibility is to find a plane into which to project the 1-ring neighborhood $\varphi(|\mathrm{star}\,(i)|)$ of a removed vertex $\varphi(|i|)$ without overlapping triangles and then retriangulate the hole in that plane. However,

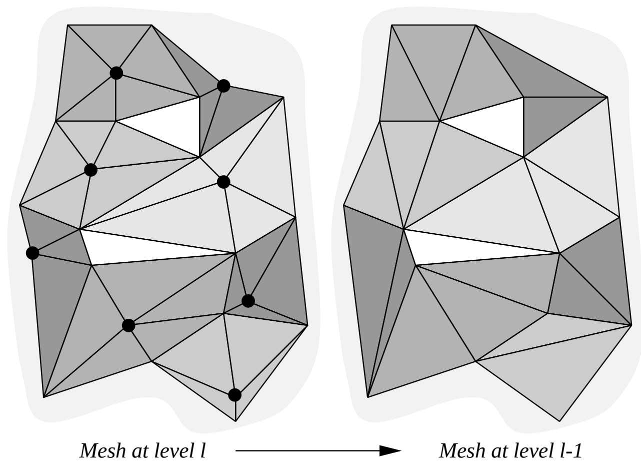

Fig. 1.18. On the left a mesh with a maximally independent set of vertices marked by heavy dots. Each vertex in the independent set has its respective star highlighted. Note that the star's of the independent set do not tile the mesh (two triangles are left white). The right side gives the retriangulation after vertex removal.

finding such a plane, which may not even exist, can be expensive and involves linear programming [5].

Instead, it is much easier and more robust to use the conformal map z^α [12] which minimizes metric distortion when mapping the neighborhood of a removed vertex into the plane. Let $\{a\}$ be a vertex to be removed. Enumerate cyclically the K_a vertices in the 1-ring $\mathcal{N}(i) = \{b_k \mid 1 \le k \le K_a\}$ such that $\{b_{k-1}, a, b_k\} \in \mathcal{K}^l$ with $b_0 = b_{K_a}$. A piecewise linear approximation of z^α, which we denote by μ_a, is defined by its values for the center point and 1-ring neighbors; namely, $\mu_a(p_a) = 0$ and $\mu_a(p_{b_k}) = r_k^\alpha \exp(i\theta_k\,\alpha)$, where $r_k = \|p_a - p_{b_k}\|$,

$$\theta_k = \sum_{l=1}^{K_a} \angle(p_{b_{l-1}}, p_a, p_{b_l}),$$

and $\alpha = 2\pi/\theta_{K_a}$. The advantages of the conformal map are numerous: it always exists, it is easy to compute, it minimizes metric distortion, and it is a bijection and thus never maps two triangles on top of each other. Once the 1-ring is flattened, we can retriangulate the hole using a constrained Delaunay triangulation (CDT), for example (see Figure 1.19). This tells us how to build $\mathcal{K}^{l-1}$.

When the vertex to be removed is a boundary vertex, we map to a half

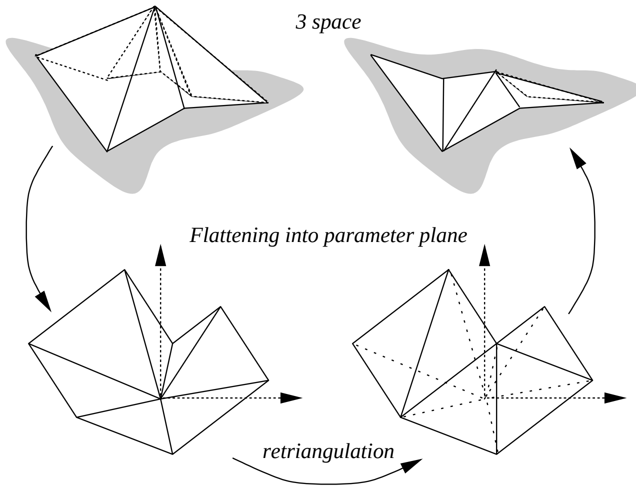

Fig. 1.19. In order to remove a vertex p_a, its star(a) is mapped from 3-space to a plane using the map z^α. In the plane the central vertex is removed and the resulting hole retriangulated (bottom right).

disk by setting $\alpha = \pi/\theta_{K_a}$ (assuming b_1 and b_{K_a} are boundary vertices and setting $\theta_1 = 0$). Retriangulation is again performed with a CDT.

1.7 Initial Parameterization

To find a parameterization, we begin by constructing a bijection Π from $\varphi(|\mathcal{K}^L|)$ to $\varphi(|\mathcal{K}^0|)$. The parameterization of the original mesh over the base domain follows from $\Pi^{-1}(\varphi(|\mathcal{K}^0|))$. In other words, the mapping of a point $p \in \varphi(|\mathcal{K}^L|)$ through Π is a point $p^0 = \Pi(a) \in \varphi(|\mathcal{K}^0|)$, which can be written as

$$p^0 = \alpha\, p_a + \beta\, p_b + \gamma\, p_c,$$

where $\{a, b, c\} \in \mathcal{K}^0$ is a face of the base domain and α, β and γ are barycentric coordinates, i.e., $\alpha + \beta + \gamma = 1$.

The mapping can be computed concurrently with the hierarchy construction. The basic idea is to successively compute piecewise linear bijections Π^l between $\varphi(|\mathcal{K}^L|)$ and $\varphi(|\mathcal{K}^l|)$ starting with Π^L, which is the identity, and ending with $\Pi^0 = \Pi$.

Notice that we only need to compute the value of Π^l at the vertices of $\mathcal{K}^L$. At any other point it follows from piecewise linearity.† Assume we are

† In the vicinity of vertices in $\mathcal{K}^l$ a triangle $\{a, b, c\} \in \mathcal{K}^L$ can straddle multiple triangles in $\mathcal{K}^l$. In this case the map depends on the flattening strategy used (see Section 1.6.2).

given Π^l and want to compute Π^{l-1}. Each vertex $\{a\} \in \mathcal{K}^L$ falls into one of the following categories:

(i) $\{a\} \in \mathcal{K}^{l-1}$: The vertex is not removed on level l and survives on level $l-1$. In this case nothing needs to be done. $\Pi^{l-1}(p_a) = \Pi^l(p_a) = p_a$.

(ii) $\{a\} \in \mathcal{K}^l \setminus \mathcal{K}^{l-1}$: The vertex gets removed when going from l to $l-1$. Consider the flattening of the 1-ring around p_a (see Figure 1.19). After retriangulation, the origin lies in a triangle which corresponds to some face $t = \{d, e, f\} \in \mathcal{K}^{l-1}$ and has barycentric coordinates (α, β, γ) with respect to the vertices of that face, i.e., $\alpha\, \mu_a(p_d) + \beta\, \mu_a(p_e) + \gamma\, \mu_a(p_f)$ (see Figure 1.20). In that case, let $\Pi^{l-1}(p_a) = \alpha\, p_d + \beta\, p_e + \gamma\, p_f$.

(iii) $\{a\} \in \mathcal{K}^L \setminus \mathcal{K}^l$: The vertex was removed earlier, thus $\Pi^l(p_a) = \alpha'\, p_{a'} + \beta'\, p_{b'} + \gamma'\, p_{c'}$ for some triangle $t' = \{a', b', c'\} \in \mathcal{K}^l$. If $t' \in \mathcal{K}^{l-1}$, nothing needs to be done; otherwise, the independent set property guarantees that exactly one vertex of t' is removed, say $\{a'\}$. Consider the conformal map $\mu_{a'}$ (Figure 1.20). After retriangulation, the $\mu_{a'}(p_a)$ lies in a triangle which corresponds to some face $t = \{d, e, f\} \in \mathcal{K}^{l-1}$ with barycentric coordinates (α, β, γ) (black dots within highlighted face in Figure 1.20). In that case, let $\Pi^{l-1}(p_a) = \alpha\, p_d + \beta\, p_e + \gamma\, p_f$ (i.e., all vertices in Figure 1.20 are reparameterized in this way).

Note that on every level, the algorithm requires a sweep through all the vertices of the finest level resulting in an overall complexity of $O(N \log N)$. Figure 1.21 visualizes the mapping we just computed.

1.7.1 Tagging and Feature Lines

In the algorithm described so far, there is no *a priori* control over which vertices end up in the base domain or how they will be connected. However, often there are features such as sharp creases in the input mesh. It is desirable to ensure that these features align with particular iso-parameter lines in the parameterization. Such features could be detected automatically or specified by the user. We consider two types of features on the finest mesh: vertices and paths of edges. Guaranteeing that a certain vertex of the original mesh ends up in the base domain is straightforward. Simply mark that vertex as unremovable throughout the DK hierarchy.

We now describe an algorithm to guarantee that a certain path of edges on the finest mesh gets mapped to an edge of the base domain. Let $\{v_a \mid 1 \leq a \leq I\} \subset \mathcal{K}^L$ be a set of vertices on the finest level which form a path,

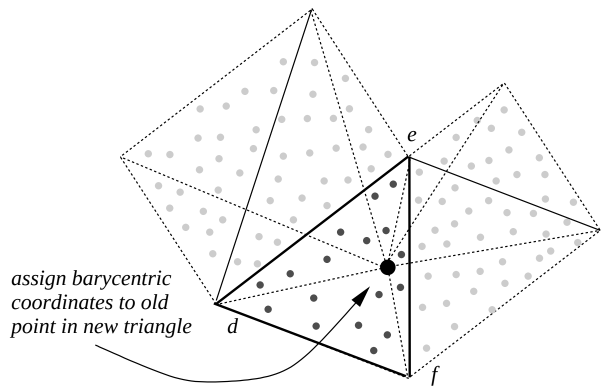

Fig. 1.20. After retriangulation of a hole in the plane (see Figure 1.19), the just removed vertex gets assigned barycentric coordinates with respect to the containing triangle on the coarser level. Similarly, all the finest level vertices that were mapped to a triangle of the hole now need to be reassigned to a triangle of the coarser level.

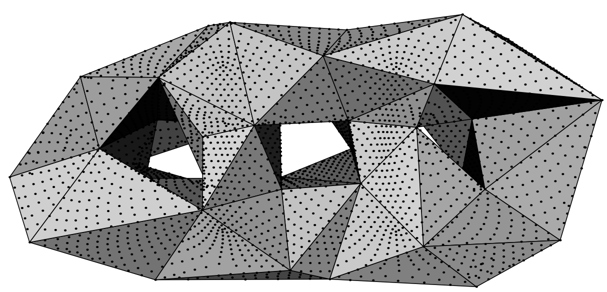

Fig. 1.21. Base domain $\varphi(|\mathcal{K}^0|)$. For each point p_a from the original mesh, its mapping $\Pi(p_a)$ is shown with a dot on the base domain.

i.e., $\{v_a, v_{a+1}\}$ is an edge. Tag all the edges in the path as feature edges. First tag v_1 and v_I, so called *dart points* [26], as unremovable so they are guaranteed to end up in the base domain. Let v_a be the first vertex on the interior of the path which gets marked for removal in the DK hierarchy, say, when going from level l to $l-1$. Because of the independent set property, v_{a-1} and v_{a+1} cannot be removed and therefore must belong to $\mathcal{K}^{l-1}$. When flattening the hole around v_a, tagged edges are treated the same as boundary edges. We first straighten out the edges $\{v_{a-1}, v_a\}$ and $\{v_a, v_{a+1}\}$ along the x-axis, and use two boundary type conformal maps to the half disk above and below (see the last paragraph of Section 1.6.2). When retriangulating the hole around v_a, we put the edge $\{v_{a-1}, v_{a+1}\}$ in $\mathcal{K}^{l-1}$, tag it as a feature edge, and compute a CDT on the upper and lower parts (see Figure 1.22).

If we apply similar procedures on coarser levels, we ensure that v_1 and v_I remain connected by a path (potentially a single edge) in the base domain. This guarantees that Π maps the curved feature path onto the coarsest level edge(s) between v_1 and v_I.

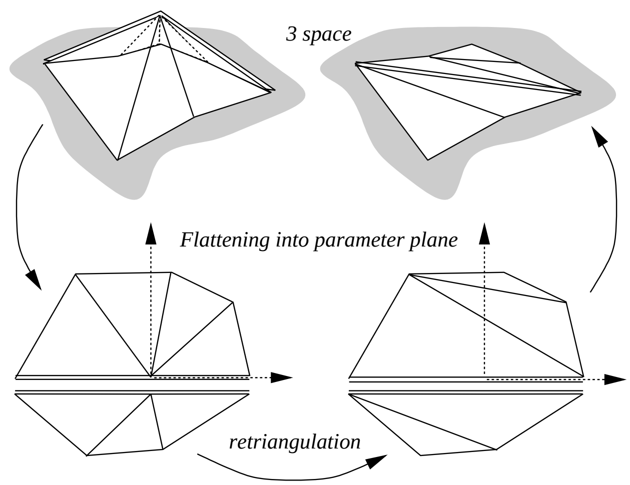

Fig. 1.22. When a vertex with two incident feature edges is removed, we want to ensure that the subsequent retriangulation adds a new feature edge to replace the two old ones.

In general, there will be multiple feature paths which may be closed or cross each other. As usual, a vertex with more than two incident feature edges is considered a corner, and marked as unremovable.

The feature vertices and paths can be provided by the user or detected automatically. As an example of the latter case, we consider every edge whose dihedral angle is below a certain threshold to be a feature edge, and every vertex whose curvature is above a certain threshold to be a feature vertex. An example of this strategy is illustrated in Figure 1.25.

1.8 Remeshing

In this section, we consider remeshing using semi-regular triangulations. In the process, we compute a smoothed version of our initial parameterization. We also show how to efficiently construct an adaptive remesh with guaranteed error bounds.

1.8.1 Uniform Remeshing

Since Π is a bijection, we can use Π^{-1} to map the base domain to the original mesh. We follow the strategy used in [14], quadrisecting the base domain and using the inverse map to obtain a semi-regular connectivity remeshing. This introduces a hierarchy of semi-regular meshes $(\tilde{\mathcal{P}}^m, \tilde{\mathcal{K}}^m)$ ($\tilde{\mathcal{P}}$ is the point set and $\tilde{\mathcal{K}}$ is the complex) obtained from m-fold midpoint subdivision of the base domain $(\mathcal{P}^0, \mathcal{K}^0) = (\tilde{\mathcal{P}}^0, \tilde{\mathcal{K}}^0)$. Midpoint subdivision implies that all new domain points lie *in* the base domain, $\tilde{\mathcal{P}}^m \subset \varphi(|\tilde{\mathcal{K}}^0|)$ and $|\tilde{\mathcal{K}}^m| = |\tilde{\mathcal{K}}^0|$. All vertices of $\tilde{\mathcal{K}}^m \setminus \tilde{\mathcal{K}}^0$ have outdegree 6. The uniform remeshing of the original mesh on level m is given by $(\Pi^{-1}(\tilde{\mathcal{P}}^m), \tilde{\mathcal{K}}^m)$.

We thus need to compute $\Pi^{-1}(q)$ where q is a point in the base domain with dyadic barycentric coordinates. In particular, we need to compute which triangle of $\varphi(|\mathcal{K}^L|)$ contains $\Pi^{-1}(q)$, or, equivalently, which triangle of $\Pi(\varphi(|\mathcal{K}^L|))$ contains q. This is a standard *point location* problem in an irregular triangulation. This problem can be solved with the point location algorithm of Brown and Faigle [1] which avoids looping that can occur with non-Delaunay meshes [21, 19]. Once we have found the triangle $\{a, b, c\}$ which contains q, we can write q as

$$q = \alpha\, \Pi(p_a) + \beta\, \Pi(p_b) + \gamma\, \Pi(p_c),$$

and thus

$$\Pi^{-1}(q) = \alpha\, p_a + \beta\, p_b + \gamma\, p_c \in \varphi(|\mathcal{K}^L|).$$

Figure 1.23 shows the result of this procedure: a level 3 uniform remeshing of a 3-holed torus using the Π^{-1} map.

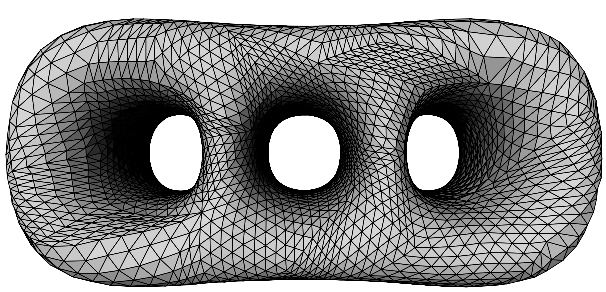

Fig. 1.23. Remeshing of 3 holed torus using midpoint subdivision. The parameterization is smooth within each base domain triangle, but clearly not across base domain triangles.

A note on complexity: The point location algorithm is essentially a walk on the finest level mesh with complexity $O(\sqrt{N})$. Hierarchical point location algorithms, which have asymptotic complexity $O(\log N)$, exist [27] but have a much larger constant. Given that we schedule the queries in a systematic order, we almost always have an excellent starting guess and observe a constant number of steps. In practice, the finest level "walking" algorithm beats the hierarchical point location algorithms for all meshes we encountered (up to $100K$ faces).

1.8.2 Smoothing the Parameterization

It is clear from Figure 1.23 that the mapping we used is not smooth across global edges. One way to obtain global smoothness is to consider a map that minimizes a global smoothness functional and goes from $\varphi(|\mathcal{K}^L|)$ to $|\mathcal{K}^0|$ rather than to $\varphi(|\mathcal{K}^0|)$. This would require an iterative PDE solver. However, computation of mappings to topological realizations that live in a high dimensional space are needlessly cumbersome and a much simpler smoothing procedure suffices.

The main idea is to compute Π^{-1} at a smoothed version of the dyadic points, rather then at the dyadic points themselves. This can equivalently be viewed as changing the parameterization. To that end, define a map $\mathcal{L}$ from the base domain to itself by the following modification of classic Loop subdivision:

- If all the points of the stencil needed for computing either a new point or smoothing an old point are inside the same triangle of the base domain, we can simply apply the Loop weights and the new points will be in that same face.
- If the stencil stretches across two faces of the base domain, we flatten them out using a "hinge" map at their common edge. We then compute the point's position in this flattened domain and extract the triangle in which the point lies together with its barycentric coordinates.
- If the stencil stretches across multiple faces, we use the conformal flattening strategy discussed earlier.

Note that the modifications to Loop force $\mathcal{L}$ to map the base domain onto the base domain. We emphasize that we do *not* apply the classic Loop scheme (which would produce a "blobby" version of the base domain). Nor are the surface approximations that we later produce Loop surfaces.

The composite map $\Pi^{-1} \circ \mathcal{L}$ is our *smoothed parameterization* that maps the base domain onto the original surface. The m-th level of uniform remeshing with the smoothed parameterization is $(\Pi^{-1} \circ \mathcal{L}(\tilde{\mathcal{P}}^m), \tilde{\mathcal{K}}^m)$, where $\tilde{\mathcal{P}}^m$, as before, are the dyadic points on the base domain. Figure 1.24 shows the result of this procedure: a level 3 uniform remeshing of a 3-holed torus using the smoothed parameterization.

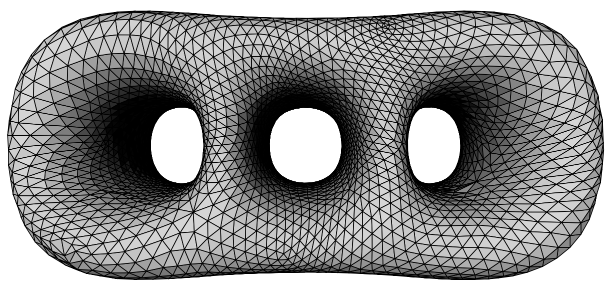

Fig. 1.24. The same remeshing of the 3-holed torus as in Figure 1.23, but this time with respect to a Loop smoothed parameterization. **Note:** Because the Loop scheme only enters in smoothing the *parameterization* the surface shown is still a sampling of the original mesh, *not* a Loop surface approximation of the original.

When the mesh is tagged, we cannot apply smoothing across the tagged edges since this would break the alignment with the features. Therefore, we use a modified versions of Loop which can deal with corners, dart points and feature edges [26, 39, 45] (see Figure 1.25).

1.8.3 Adaptive Remeshing

One of the advantages of meshes with subdivision connectivity is that classical multiresolution and wavelet algorithms can be employed. The standard wavelet algorithms used, e.g., in image compression, start from the finest level, compute the wavelet transform, and then obtain an efficient representation by discarding small wavelet coefficients. Eck et al. [14, 15] as well as Certain et al. [4] follow a similar approach. They remesh using a uniformly subdivided grid followed by decimation through wavelet thresholding. This has the drawback that in order to resolve a small local feature on the original mesh, one may need to subdivide to a very fine level. Each extra level quadruples the number of triangles, most of which will later be decimated using the wavelet procedure. Imagine, for example, a plane which is coarsely triangulated except for a narrow spike. Making the spike width sufficiently small, the number of levels needed to resolve it can be made arbitrarily high.

Instead of first building a uniform remesh and then pruning it, we can immediately build the adaptive mesh with a guaranteed conservative error bound. This is possible because the DK hierarchy contains the information on how much subdivision is needed in any given area. Essentially, one can let the irregular DK hierarchy "drive" the adaptive construction of the semi-regular pyramid.

First compute for each triangle $t \in \mathcal{K}^0$ the following error quantity:

$$E(t) = \max_{p_a \in \mathcal{P}^L \text{and } \Pi(p_a) \in \varphi(|t|)} \operatorname{dist}(p_a, \varphi(|t|)).$$

This measures the distance between one triangle in the base domain and the vertices of the finest level mapped to that triangle.

The adaptive algorithm is now straightforward. Set a certain relative error threshold ϵ. Compute $E(t)$ for all triangles of the base domain. If $E(t)/B$, where B is the largest side of the bounding box, is larger than ϵ, subdivide the domain triangle using the Loop procedure above. Next, we need to reassign vertices to the triangles of level $m = 1$. This is done as follows: For each point $p_a \in \mathcal{P}^L$ consider the triangle t of $\mathcal{K}^0$ to which it it is currently assigned. Next consider the four children of t on level 1, t_j with $j = 0, 1, 2, 3$ and compute the distance between p_a and each of the $\varphi(|t_j|)$. Assign p_a to the closest child. Once the finest level vertices have been reassigned to level 1 triangles, the errors for those triangles can be computed. Now iterate this procedure until all triangles have an error below the threshold. Because all errors are computed from the finest level, we are guaranteed to resolve all features within the error bound. Note that we are not computing the true distance between the original vertices and a given approximation, but rather an easy to compute upper bound for it.

In order to be able to compute the Loop smoothing map $\mathcal{L}$ on an adaptively subdivided grid, the grid needs to satisfy a *vertex restriction criterion*, i.e., if a vertex has a triangle incident to it with depth i, then it must have a complete 1-ring at level $i-1$ [47]. This restriction may necessitate subdividing some triangles even if they are below the error threshold. Examples of adaptive remeshing can be seen in Figure 1.15 (lower left) and Figure 1.25.

Figure 1.26 shows an example of a constrained parameterization and subsequent adaptive remeshing. The original dataset of 100,000 triangles is shown on the left. The red lines indicate user supplied feature constraints which may facilitate subsequent animation. The green lines show some representative iso-parameter lines of our parameterization subject to the red feature constraints. The middle image shows an adaptive subdivision connectivity remesh with 74698 triangles ($\epsilon = 0.5\%$). On the right we have highlighted a group of patches, two over the right (constrained) eye and

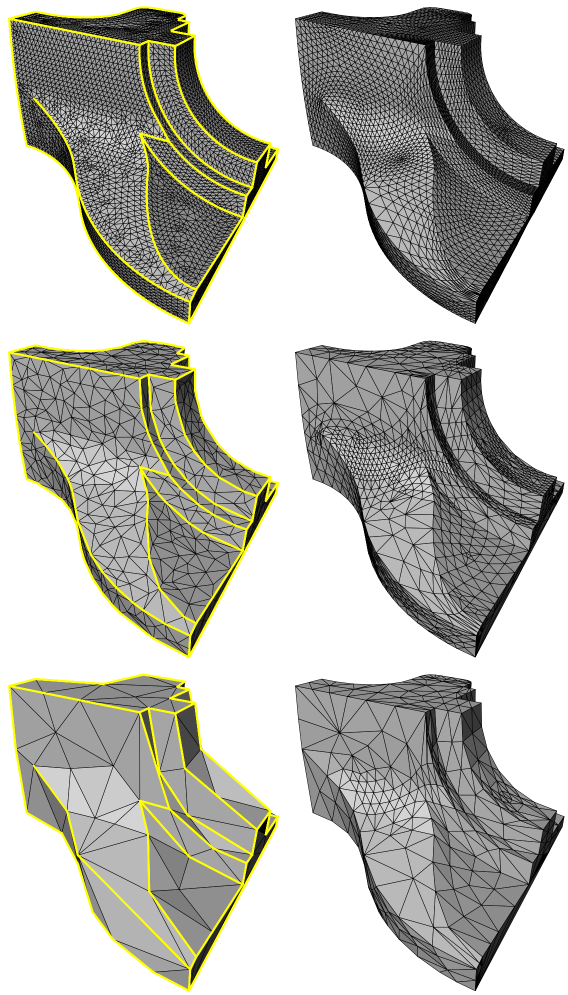

Fig. 1.25. Left (top to bottom): three levels in the DK pyramid, finest ($L = 15$) with 12946, intermediate ($l = 8$) with 1530, and coarsest ($l = 0$) with 168 triangles. Feature edges, dart and corner vertices survive on the base domain. Right (bottom to top): adaptive mesh with $\epsilon = 5\%$ and 1120 triangles (bottom), $\epsilon = 1\%$ and 3430 triangles (middle), and uniform level 3 (top). (Original dataset courtesy Hugues Hoppe.)

one over the left (unconstrained) eye. This indicates how user supplied constraints force domain patches to align with desired features. Other enforced patch boundaries are the eyebrows, center of the nose, and middle of lips (see red lines in left image).

Finally, the example shown in Figure 1.15 starts with an original mesh

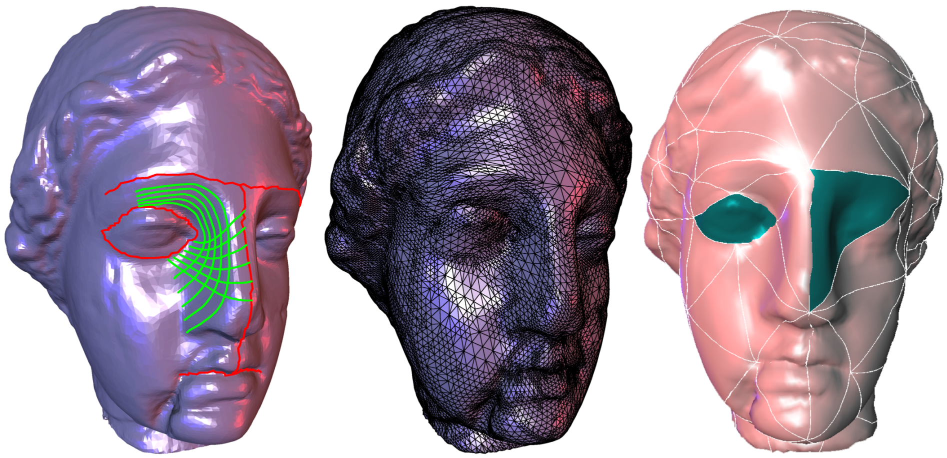

Fig. 1.26. Example of a constrained parameterization based on user input. Top: original input mesh (100,000 triangles) with edge tags superimposed in red. Green lines show some smooth iso-parameter lines of our parameterization. The middle shows an adaptive subdivision connectivity remesh. The bottom one patches corresponding to the eye regions (right eye was constrained, left eye was not) are highlighted to indicate the resulting alignment of top level patches with the feature lines. (Dataset courtesy Cyberware.)

(96966 triangles) on the top left and results in an adaptive, subdivision connectivity remesh on the bottom left. This remesh was subsequently edited in the interactive multiresolution editing system described earlier (bottom middle).

1.9 Summary

Exploiting multiresolution is now widely considered an important ingredient in the construction of scalable algorithms for many computer graphics applications. Such constructions fall into two basic categories, those generalizing classical notions of multiresolution which are intricately connected with subdivision and wavelets, and those based on mesh simplification. The former can leverage the considerable mathematical machinery developed for the classical setting. The latter are still very new and little is known so far about the construction of smooth functions over the associated mesh hierarchies (for first steps in this direction see [8, 7, 22]).

In this paper we primarily considered subdivision for surfaces and its multiresolution extension through the introduction of detail vectors at successive levels of subdivision. The resulting representations are very flexible since they can represent arbitrary 2-manifold surfaces (with boundary). Because of the hierarchical structure many algorithms which are geared towards delivering interactive performance on small computers are immediately applicable. For example, adaptive approximation based on local criteria is easy to include.

Unfortunately many examples of complex geometry, in particular those coming from 3D scanning sources are not imbued with the particular semi-regular mesh structure required by subdivision based algorithms. One way to address this challenge is through remeshing, i.e., resampling of the original geometry onto a semi-regular mesh. Once this step is performed all the advantages of the semi-regular setting apply. We described one such algorithm in detail. It exploits mesh simplification to discover a low complexity base complex which can be employed as the domain for the coordinate functions describing the original geometry. It is then an easy matter to resample the original geometry with guaranteed error bounds.

1.10 Outlook

The demands of real applications are often not satisfied by the traditional constructions arising from the "mathematical clean room." Examples include, irregular samples, non-trivial measures, arbitrary topology 2-manifolds, small computational budgets, and very large datasets. Because of this we have witnessed over the last few years increasing attention being devoted to the generalization of multiresolution from the infinite, regular, tensor product setting, to the semi-regular, and, much more recently, to the irregular setting. In many of these cases algorithms are running ahead of the associated theory, which often requires completely new tools to analyze settings such as irregular samples, or manifolds with non-trivial structure.

The interchange between applications in computer graphics and, more generally, computer modeling, simulation, and design, and the classical mathematical treatment of multiresolution has led to significant advances in both theory and applications and we expect this to continue into the future. While processing power keeps increasing there appears no end in sight for our ability to absorb all avaialble cycles and still be left wanting more. Similar observations apply to availalbe network bandwidth. Multiresolution in all its forms will continue to play a central role and the field is wide open and wanting for further, significant advances.

Acknowledgments

The author was supported in part by NSF (ACI-9624957, ACI-9721349, DMS-9874082, DMS-9872890), DOE (W-7405-ENG-48), and the NSF STC for Computer Graphics and Scientific Visualization. Other support was provided by Alias|wavefront and a Packard Fellowship.

None of this work could have happened without my students and collaborators and in particular Wim Sweldens who has been part of developing these ideas for the last 5 years.

Bibliography

Brown, P. J. C., and Faigle, C. T. A Robust Efficient Algorithm for Point Location in Triangulations. Tech. rep., Cambridge University, February 1997.

Burt, P. J., and Adelson, E. H. Laplacian Pyramid as a Compact Image Code. *IEEE Trans. Commun. 31*, 4 (1983), 532–540.

Catmull, E., and Clark, J. Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes. *Computer Aided Design 10*, 6 (1978), 350–355.

Certain, A., Popović, J., DeRose, T., Duchamp, T., Salesin, D., and Stuetzle, W. Interactive Multiresolution Surface Viewing. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, 91–98, 1996.

Cohen, J., Manocha, D., and Olano, M. Simplifying Polygonal Models Using Successive Mappings. In *Proceedings IEEE Visualization 97*, 395–402, October 1997.

Curless, B., and Levoy, M. A Volumetric Method for Building Complex Models from Range Images. In *SIGGRAPH 96 Conference Proceedings*, H. Rushmeier, Ed., Annual Conference Series, 303–312, August 1996.

Daubechies, I., Guskov, I., and Sweldens, W. Commutation for Irregular Subdivision. Tech. rep., Bell Laboratories, Lucent Technologies, 1998.

Daubechies, I., Guskov, I., and Sweldens, W. Regularity of irregular subdivision. *Const. Approx.* (1999), to appear.

DeRose, T., Kass, M., and Truong, T. Subdivision Surfaces in Character Animation. *Computer Graphics (SIGGRAPH '99 Proceedings)* (1998), 85–94.

Dobkin, D., and Kirkpatrick, D. A Linear Algorithm for Determining the Separation of Convex Polyhedra. *Journal of Algorithms 6* (1985), 381–392.

Doo, D., and Sabin, M. Analysis of the Behaviour of Recursive Division Surfaces near Extraordinary Points. *Computer Aided Design 10*, 6 (1978), 356–360.

Duchamp, T., Certain, A., DeRose, T., and Stuetzle, W. Hierarchical Computation of PL harmonic Embeddings. Tech. rep., University of Washington, July 1997.

Dyn, N., Levin, D., and Gregory, J. A. A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control. *ACM Transactions on Graphics 9*, 2 (April 1990), 160–169.

Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzle, W. Multiresolution Analysis of Arbitrary Meshes. In *Computer Graphics (SIGGRAPH '95 Proceedings)*, 173–182, 1995.

Eck, M., and Hoppe, H. Automatic Reconstruction of B-Spline Surfaces of Arbitrary Topological Type. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, 325–334, 1996.

Finkelstein, A., and Salesin, D. H. Multiresolution Curves. *Computer Graphics* Proceedings, Annual Conference Series, 261–268, July 1994.

Forsey, D., and Wong, D. Multiresolution Surface Reconstruction for Hierarchical B-splines. Tech. rep., University of British Columbia, 1995.

Forsey, D. R., and Bartels, R. H. Hierarchical B-Spline Refinement. *Computer Graphics (SIGGRAPH '88 Proceedings)*, Vol. 22, No. 4, pp. 205–212, August 1988.

Garland, M., and Heckbert, P. S. Fast Polygonal Approximation of Terrains and Height Fields. Tech. Rep. CMU-CS-95-181, CS Dept., Carnegie Mellon U., September 1995.

Gortler, S. J., and Cohen, M. F. Hierarchical and Variational Geometric Modeling with Wavelets. In *Proceedings Symposium on Interactive 3D Graphics*, May 1995.

Guibas, L., and Stolfi, J. Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams. *ACM Transactions on Graphics 4*, 2 (April 1985), 74–123.

Guskov, I., Sweldens, W., and Schröder, P. Multiresolution Signal Processing for Meshes. Tech. Rep. PACM TR 99-01, Princeton University, 1999. submitted.

Heckbert, P. S., and Garland, M. Survey of Polygonal Surface Simplification Algorithms. Tech. rep., Carnegie Mellon University, 1997.

Hoppe, H. Progressive Meshes. In *SIGGRAPH 96 Conference Proceedings*, H. Rushmeier, Ed., Annual Conference Series, 99–108, August 1996.

Hoppe, H. View-Dependent Refinement of Progressive Meshes. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, 189–198, 1997.

Hoppe, H., DeRose, T., Duchamp, T., Halstead, M., Jin, H., McDonald, J., Schweitzer, J., and Stuetzle, W. Piecewise Smooth Surface Reconstruction. In *Computer Graphics (SIGGRAPH '94 Proceedings)*, 295–302, 1994.

Kirkpatrick, D. Optimal Search in Planar Subdivisions. *SIAM J. Comput. 12* (1983), 28–35.

Klein, A., Certain, A., DeRose, T., Duchamp, T., and Stuetzle, W. Vertex-based Delaunay Triangulation of Meshes of Arbitrary Topological Type. Tech. rep., University of Washington, July 1997.

Kobbelt, L. Interpolatory Subdivision on Open Quadrilateral Nets with Arbitrary Topology. In *Proceedings of Eurographics 96*, Computer Graphics Forum, 409–420, 1996.

Kobbelt, L. A Variational approach to subdivision. *Comput. Aided Geom. Des. 13* (1996), 743–761.

Krishnamurthy, V., and Levoy, M. Fitting Smooth Surfaces to Dense Polygon Meshes. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, 313–324, 1996.

Levin, A. Combined Subdivision Schemes for the Design of Surfaces Satisfying Boundary Conditions. To appear in CAGD, 1999.

Loop, C. Smooth Subdivision Surfaces Based on Triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.

Lounsbery, M. *Multiresolution Analysis for Surfaces of Arbitrary Topological*

Type. PhD thesis, Department of Computer Science, University of Washington, 1994.

Lounsbery, M., DeRose, T., and Warren, J. Multiresolution Analysis for Surfaces of Arbitrary Topological Type. *Transactions on Graphics 16*, 1 (January 1997), 34–73.

Peters, J., and Reif, U. The simplest subdivision scheme for smoothing polyhedra. *ACM Transactions on Graphics 16(4)* (October 1997).

Prautzsch, H., and Umlauf, G. Improved Triangular Subdivision Schemes. In *Proceedings of the CGI '98*, 1998.

Schröder, P., and Sweldens, W. Spherical Wavelets: Efficiently Representing Functions on the Sphere. In *Computer Graphics (SIGGRAPH '95 Proceedings)*, 1995.

Schweitzer, J. E. *Analysis and Application of Subdivision Surfaces*. PhD thesis, University of Washington, 1996.

Stam, J. Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values. In *Computer Graphics (SIGGRAPH '98 Proceedings)*, 395–404, July 1998.

Sweldens, W. The lifting scheme: A custom-design construction of biorthogonal wavelets. *Appl. Comput. Harmon. Anal. 3*, 2 (1996), 186–200.

Sweldens, W., and Schröder, P. Building your own wavelets at home. In *Wavelets in Computer Graphics*. ACM SIGGRAPH Course notes, 1996, pp. 15–87.

Taubin, G. A Signal Processing Approach to Fair Surface Design. In *SIGGRAPH 95 Conference Proceedings*, R. Cook, Ed., Annual Conference Series, 351–358, August 1995.

Xia, J. C., and Varshney, A. Dynamic View-Dependent Simplification for Polygonal Models. In *Proceedings Visualization 96*, 327–334, October 1996.

Zorin, D. *Stationary Subdivision and Multiresolution Surface Representations*. PhD thesis, California Institute of Technology, 1997.

Zorin, D., Schröder, P., and Sweldens, W. Interpolating Subdivision for Meshes with Arbitrary Topology. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, 189–192, 1996.

Zorin, D., Schröder, P., and Sweldens, W. Interactive Multiresolution Mesh Editing. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, 259–268, 1997.