MAPS: Multiresolution Adaptive Parameterization of Surfaces

Aaron W. F. Lee* Princeton University | Wim Sweldens† Bell Laboratories | Peter Schröder‡ Caltech | Lawrence Cowsar§ Bell Laboratories | David Dobkin¶ Princeton University

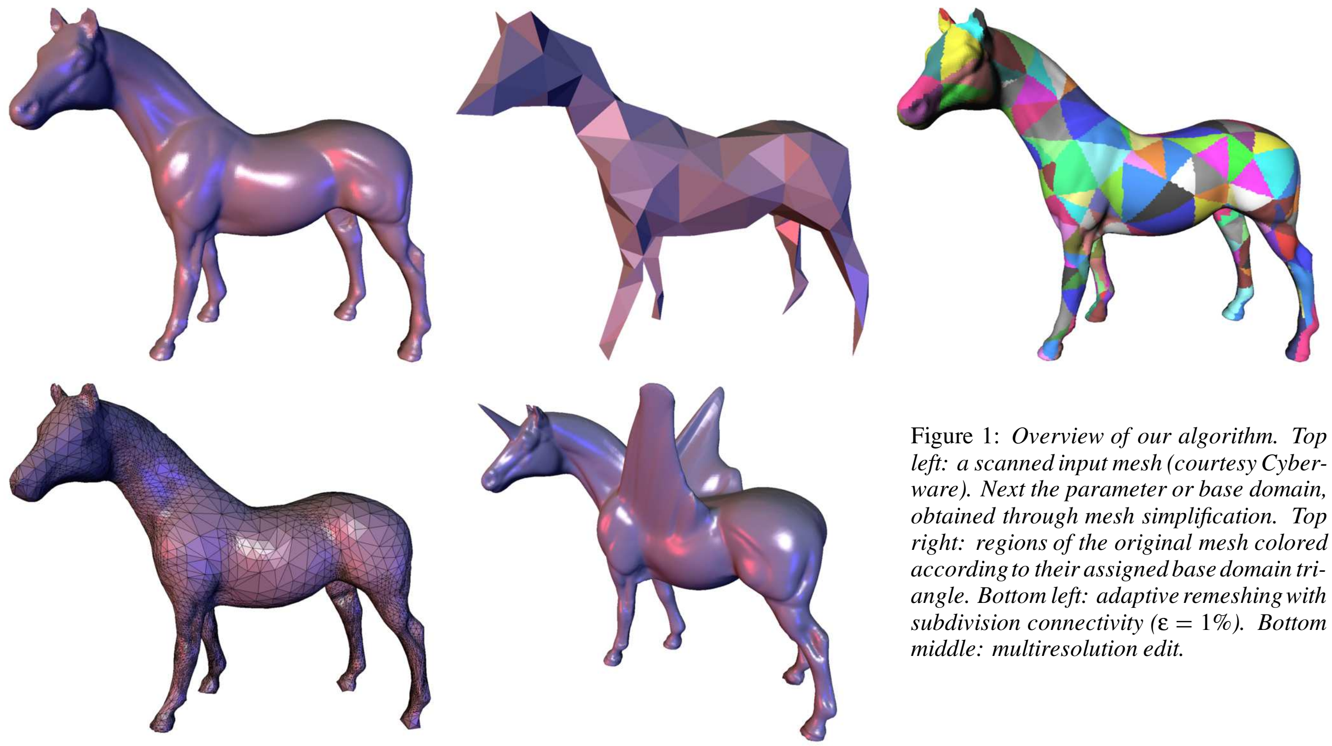

Figure 1: *Overview of our algorithm. Top left: a scanned input mesh (courtesy Cyberware). Next the parameter or base domain, obtained through mesh simplification. Top right: regions of the original mesh colored according to their assigned base domain triangle. Bottom left: adaptive remeshing with subdivision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

Abstract

We construct smooth parameterizations of irregular connectivity triangulations of arbitrary genus 2-manifolds. Our algorithm uses hierarchical simplification to efficiently induce a parameterization of the original mesh over a base domain consisting of a small number of triangles. This initial parameterization is further improved through a hierarchical smoothing procedure based on Loop subdivision applied in the parameter domain. Our method supports both fully automatic and user constrained operations. In the latter, we accommodate point and edge constraints to force the alignment of iso-parameter lines with desired features. We show how to use the parameterization for fast, hierarchical subdivision connectivity remeshing with guaranteed error bounds. The remeshing algorithm constructs an adaptively subdivided mesh directly without first resorting to uniform subdivision followed by subsequent sparsification. It thus avoids the exponential cost of the latter. Our parameterizations are also useful for texture mapping and morphing applications, among others.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: *Picture/Image Generation – Display Algorithms, Viewing Algorithms*; I.3.5 [Computer Graphics]: *Computational Geometry and Object Modeling - Curve, Surface, Solid and Object Representations, Hierarchy and Geometric Transformations, Object Hierarchies*.

Additional Key Words and Phrases: Meshes, surface parameterization, mesh simplification, remeshing, texture mapping, multiresolution, subdivision surfaces, Loop scheme.

*wailee@cs.princeton.edu
†wim@bell-labs.com
‡ps@cs.caltech.edu
§cowsar@bell-labs.com
¶dpd@cs.princeton.edu

1 Introduction

Dense triangular meshes routinely result from a number of 3D acquisition techniques, e.g., laser range scanning and MRI volumetric imaging followed by iso-surface extraction (see Figure 1 top left). The triangulations form a surface of arbitrary topology—genus, boundaries, connected components—and have irregular connectivity. Because of their complex structure and tremendous size, these meshes are awkward to handle in such common tasks as storage, display, editing, and transmission.

Multiresolution representations are now established as a fundamental component in addressing these issues. Two schools exist. One approach extends classical multiresolution analysis and subdivision techniques to arbitrary topology surfaces [19, 20, 7, 3]. The alternative is more general and is based on sequential mesh simplification, e.g., progressive meshes (PM) [12]; see [11] for a review. In either case, the objective is to represent triangulated 2-manifolds in an efficient and flexible way, and to use this description in fast algorithms addressing the challenges mentioned above. Our approach fits in the first group, but draws on ideas from the second group.

An important element in the design of algorithms which manipulate mesh approximations of 2-manifolds is the construction of "nice" parameterizations when none are given. Ideally, the manifold is parameterized over a base domain consisting of a small number of triangles. Once a surface is understood as a function from the base domain into $\mathbf{R}^3$ (or higher-D when surface attributes are considered), many tools from areas such as approximation theory, signal processing, and numerical analysis are at our disposal. In particular, classical multiresolution analysis can be used in the design and analysis of algorithms. For example, error controlled, adaptive remeshing can be performed easily and efficiently. Figure 1 shows the outline of our procedure: beginning with an irregular input mesh (top left), we find a base domain 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). Additionally, there are other practical payoffs to good parameterizations, for example in texture mapping and morphing.

In this paper we present an algorithm for the fast computation of smooth parameterizations of dense 2-manifold meshes with arbitrary topology. Specifically, we make the following contributions

- We describe an $O(N \log N)$ time and storage algorithm to construct a logarithmic level hierarchy of arbitrary topology, irregular connectivity meshes based on the Dobkin-Kirkpatrick (DK) algorithm. Our algorithm accommodates geometric criteria such as area and curvature as well as vertex and edge constraints.
- We construct a smooth parameterization of the original mesh over the base domain. This parameterization is derived through repeated conformal remapping during graph simplification followed by a parameter space smoothing procedure based on the Loop scheme. The resulting parameterizations are of high visual and numerical quality.
- Using the smooth parameterization, we describe an algorithm for adaptive, hierarchical remeshing of arbitrary meshes into subdivision connectivity meshes. The procedure is fully automatic, but also allows for user intervention in the form of fixing point or path features in the original mesh. The remeshed manifold meets conservative approximation bounds.

Even though the ingredients of our construction are reminiscent of mesh simplification algorithms, we emphasize that our goal is not the construction of another mesh simplification procedure, but rather the construction of smooth parameterizations. We are particularly interested in using these parameterizations for remeshing, although they are useful for a variety of applications.

1.1 Related Work

A number of researchers have considered—either explicitly or implicitly—the problem of building parameterizations for arbitrary topology, triangulated surfaces. This work falls into two main categories: (1) algorithms which build a smoothly parameterized approximation of a set of samples (e.g. [14, 1, 17]), and (2) algorithms which remesh an existing mesh with the goal of applying classical multiresolution approaches [7, 8].

A related, though quite different problem, is the maintenance of a *given* parameterization during mesh simplification [4]. We emphasize that our goal is the *construction* of mappings when none are given.

In the following two sections, we discuss related work and contrast it to our approach.

1.1.1 Approximation of a Given Set of Samples

Hoppe and co-workers [14] describe a fully automatic algorithm to approximate a given polyhedral mesh with Loop subdivision patches [18] 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 over the base domain. Since the approach only uses subdivision, small features in the original mesh can only be resolved accurately by increasing the number of triangles in the base domain accordingly. A similar approach, albeit using A-patches, was described by Bajaj and co-workers [1]. 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 goal surface.

This problem was addressed in work of Krishnamurthy and Levoy [17]. They approximate densely sampled geometry with bicubic spline patches and displacement maps. Arguing that a fully automatic system cannot put iso-parameter lines where a skilled 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 mesh whose base domain need not mimic small scale geometric features.

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 surface in regions of high curvature [17, Figure 7].

1.1.2 Remeshing

Lounsbery and co-workers [19, 20] 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 (e.g. [22, 7, 27, 8, 3, 28]). The central requirement of these methods is that the input mesh have subdivision connectivity. This is generally not true for meshes derived from 3D scanning sources.

To overcome this problem, Eck and co-workers [7] developed an algorithm to compute smooth parameterizations of high resolution polyhedral meshes over a low face count base domain. Using such a mapping, the original surface can be remeshed using subdivision connectivity. After this conversion step, adaptive simplification, compression, progressive transmission, rendering, and editing become simple and efficient operations [3, 8, 28].

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 [7, 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 [6], who reduced the harmonic map computations from their initial $O(N^2)$ complexity to $O(N \log N)$ through hierarchical preconditioning. The hierarchy construction they employed for use in a multigrid solver is related to our hierarchy construction.

The initial Voronoi tile construction relies on a number of heuristics which render the overall algorithm fragile (for an improved version see [16]). Moreover, there is no explicit control over the number of triangles in the base domain or the placement of patch boundaries.

The algorithm generates only uniformly subdivided meshes which later can be decimated through classical wavelet methods. Many extra globally subdivided levels may be needed to resolve one small local feature; moreover, each additional level quadruples the amount of work and storage. This can lead to the intermediate construction of many more triangles than were contained in the input mesh.

1.2 Features of MAPS

Our algorithm was designed to overcome the drawbacks of previous work as well as to introduce new features. We use a fast coarsification strategy to define the base domain, avoiding the potential difficulties of finding Voronoi tiles [7, 16]. Since our algorithm proceeds from fine to coarse, correspondence problems found in coarse to fine strategies [17] are avoided, and all features are correctly resolved. We use conformal maps for continued remapping during coarsification to immediately produce a global parameterization of the original mesh. This map is further improved through the use of a hierarchical Loop smoothing procedure obviating the need for iterative numerical solvers [7]. Since the procedure is performed globally, derivative discontinuities at the edges of the base domain are avoided [7]. In contrast to fully automatic methods [7], the algorithm supports vertex and edge tags [14] to constrain the parameterization to align with selected features; however, the user is not required to specify the entire patch network [17]. During remeshing we take advantage of the original fine to coarse hierarchy to output a sparse, adaptive, subdivision connectivity mesh directly without resorting to a depth first oracle [22] or the need to produce a uniform subdivision connectivity mesh at exponential cost followed by wavelet thresholding [3].

2 Hierarchical Surface Representation

In this section we describe the main components of our algorithm, coarsification and map construction. We begin by fixing our notation.

2.1 Notation

When describing surfaces mathematically, it is useful to separate the topological and geometric information. To this end we introduce some notation adapted from [24]. We denote a triangular mesh as a pair $(\mathcal{P}, \mathcal{K})$, where $\mathcal{P}$ is a set of N point positions $p_i = (x_i, y_i, z_i) \in \mathbf{R}^3$ with $1 \leq i \leq N$, 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 $v = \{i\} \in \mathcal{K}$, edges $e = \{i, j\} \in \mathcal{K}$, and faces $f = \{i, j, k\} \in \mathcal{K}$, 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 e_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 $\{e_i \mid i \in s\}$. Thus $|\{i\}| = e_i$, $|\{i, j\}|$ is the open line segment between e_i and e_j, and $|\{i, j, k\}|$ 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 $\varphi(e_i) = p_i$. The resulting polyhedron consists of points, segments, and triangles in $\mathbf{R}^3$.

Two vertices $\{i\}$ and $\{j\}$ are *neighbors* if $\{i, j\} \in \mathcal{K}$. 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 3, left side). The 1-ring neighborhood of a vertex $\{i\}$ is the set

$$\mathcal{N}(i) = \{ j \mid \{i, j\} \in \mathcal{K}\}.$$

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

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

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

We define a conservative curvature estimate, $\kappa(i) = |\kappa_1| + |\kappa_2|$ at p_i, using the principal curvatures κ_1 and κ_2. These are estimated by the standard procedure of first establishing a tangent plane at p_i and then using a second degree polynomial to approximate $\varphi(|\text{star}(i)|)$.

2.2 Mesh Hierarchies

An important part of our 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 4).

Several approaches for such mesh simplification have been proposed, most notably progressive meshes (PM) [12]. 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 [25, 13], which defines levels in a hierarchy. The depth of these hierarchies appears "reasonable" in practice, though can vary considerably for the same dataset [13].

Our approach is similar in spirit, but inspired by the hierarchy proposed by Dobkin and Kirkpatrick (DK) [5], which guarantees that the number of levels L is $O(\log N)$. While the original DK hierarchy is built for convex polyhedra, we show how the idea behind DK can be used for general polyhedra. 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 remove with a particular retriangulation of the resulting hole (see Figure 2). The main reason we chose an algorithm based on the ideas of the DK hierarchy is that it guarantees a logarithmic bound on the number of levels. However, we emphasize that the ideas behind our map constructions apply equally well to PM type algorithms.

2.3 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 3). To find such a set, the original DK algorithm used a greedy approach based only on *topological* information. Instead, we use a priority queue based on both *geometric and topological* information.

At the start of each level of the original 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 of outdegree 12 or

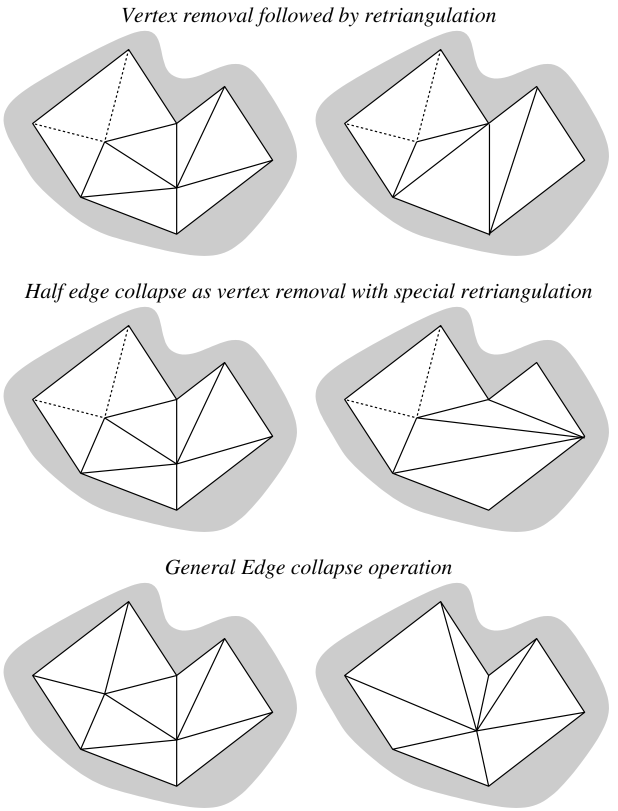

Figure 2: *Examples of different atomic mesh simplification steps. At the top vertex removal, in the middle half-edge collapse, and edge collapse at the bottom.*

more. Thus it is guaranteed that at least $1/24$ of the vertices will be removed at each level [5]. In practice, it turns out one can remove roughly $1/4$ of the vertices reflecting the fact that the graph is four-colorable. 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.

In our approach, we stay in the DK framework, but replace the random selection of vertices by a priority queue based on geometric information. Roughly speaking, vertices with small and flat 1-ring neighborhoods will be chosen first. At level l, for a vertex $p_i \in \mathcal{P}^l$, we consider its 1-ring neighborhood $\varphi(|\mathrm{star}(i)|)$ and compute its area $a(i)$ and estimate its curvature $\kappa(i)$. These quantities are computed relative to $\mathcal{K}^l$, the current level. We assign a priority to $\{i\}$ inversely proportional to a convex combination of relative area and curvature

$$w(\lambda, i) = \lambda \frac{a(i)}{\max_{p_i \in \mathcal{P}^l} a(i)} + (1-\lambda)\frac{\kappa(i)}{\max_{p_i \in \mathcal{P}^l} \kappa(i)}.$$

(We found $\lambda = 1/2$ to work well in our experiments.) Omitting all vertices of outdegree greater than 12 from the queue, removal of a constant fraction of vertices is still guaranteed. Because of the sort implied by the priority queue, the complexity of building the entire hierarchy grows to $O(N \log N)$.

Figure 4 shows three stages (original, intermediary, coarsest) of the DK hierarchy. Given that the coarsest mesh is homeomorphic to the original mesh, it can be used as the domain of a parameterization.

2.4 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

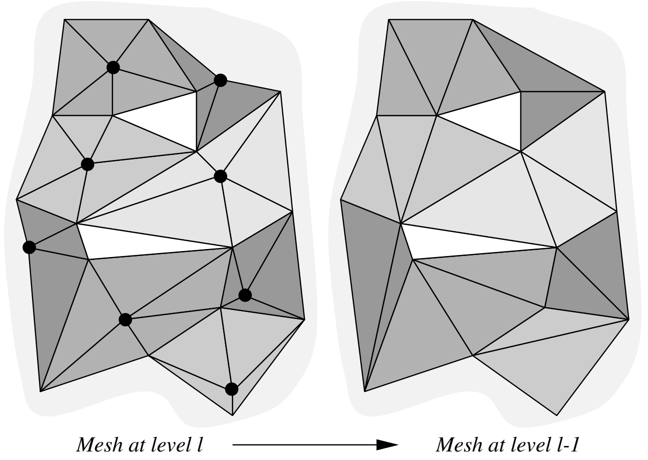

Figure 3: *On the left a mesh with a maximally independent set of vertices marked by heavy dots. Each vertex in the independent set has its respective star highlighted. Note that the star's of the independent set do not tile the mesh (two triangles are left white). The right side gives the retriangulation after vertex removal.*

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, finding such a plane, which may not even exist, can be expensive and involves linear programming [4].

Instead, we use the conformal map z^a [6] which minimizes metric distortion to map the neighborhood of a removed vertex into the plane. Let $\{i\}$ be a vertex to be removed. Enumerate cyclically the K_i vertices in the 1-ring $\mathcal{N}(i) = \{j_k \mid 1 \le k \le K_i\}$ such that $\{j_{k-1}, i, j_k\} \in \mathcal{K}^l$ with $j_0 = j_{K_i}$. A piecewise linear approximation of z^a, which we denote by μ_i, is defined by its values for the center point and 1-ring neighbors; namely, $\mu_i(p_i) = 0$ and $\mu_i(p_{j_k}) = r_k^a \exp(i\theta_k a)$, where $r_k = \|p_i - p_{j_k}\|$,

$$\theta_k = \sum_{l=1}^{k} \angle(p_{j_{l-1}}, p_i, p_{j_l}),$$

and $a = 2\pi/\theta_{K_i}$. 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, for example, a constrained Delaunay triangulation (CDT) (see Figure 5). 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 disk by setting $a = \pi/\theta_{K_i}$ (assuming j_1 and j_{K_i} are boundary vertices and setting $\theta_1 = 0$). Retriangulation is again performed with a CDT.

3 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 $p^0 = \Pi(v) \in \varphi(|\mathcal{K}^0|)$, which can be written as

$$p^0 = \alpha\, p_i + \beta\, p_j + \gamma p_k,$$

where $\{i, j, k\} \in \mathcal{K}^0$ 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

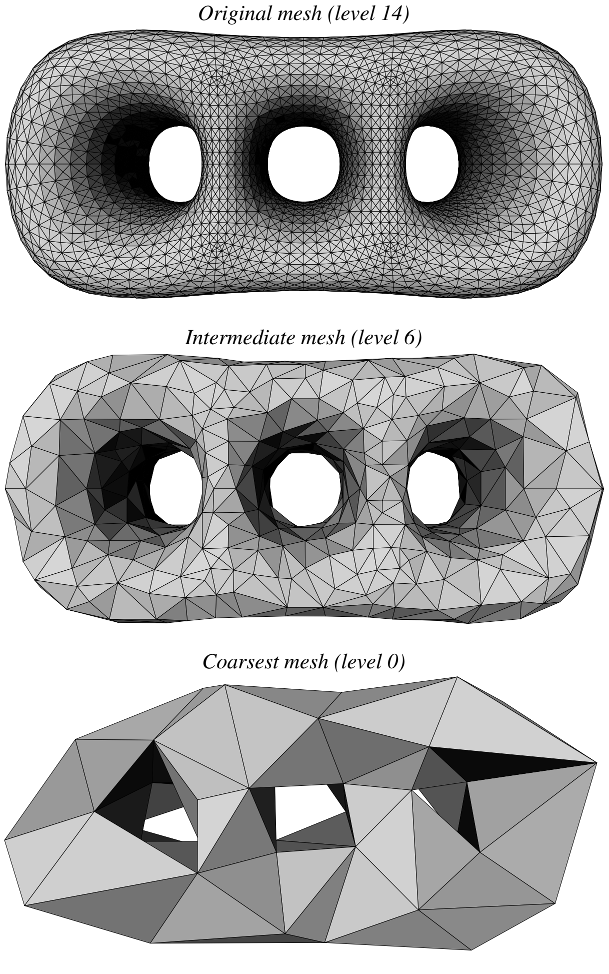

Figure 4: *Example of a modified DK mesh hierarchy. At the top the finest (original) mesh $\varphi(|\mathcal{K}^L|)$ followed by an intermediate mesh, and the coarsest (base) mesh $\varphi(|\mathcal{K}^0|)$ at the bottom (original dataset courtesy University of Washington).*

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

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

2. $\{i\} \in \mathcal{K}^l \setminus \mathcal{K}^{l-1}$: The vertex gets removed when going from l to $l-1$. Consider the flattening of the 1-ring around p_i (see Figure 5). After retriangulation, the origin lies in a triangle which corresponds to some face $t = \{j,k,m\} \in \mathcal{K}^{l-1}$ and has barycentric coordinates (α,β,γ) with respect to the vertices of that face, i.e., $\alpha\mu_i(p_j)+\beta\mu_i(p_k)+\gamma\mu_i(p_m)$ (see Figure 6). In that case, let $\Pi^{l-1}(p_i) = \alpha\, p_j + \beta\, p_k + \gamma p_m$.

3. $\{i\} \in \mathcal{K}^L \setminus \mathcal{K}^l$: The vertex was removed earlier, thus $\Pi^l(p_i) = \alpha' p_{j'} + \beta' p_{k'} + \gamma' p_{m'}$ for some triangle $t' = \{j',k',m'\} \in \mathcal{K}^l$. If $t' \in \mathcal{K}^{l-1}$, nothing needs to be done; otherwise, the independent set guarantees that exactly one vertex of t' is removed, say $\{j'\}$. Consider the conformal map $\mu_{j'}$ (Figure 6). After retriangulation, the $\mu_{j'}(p_i)$ lies in a triangle which corresponds to some face $t = \{j,k,m\} \in \mathcal{K}^{l-1}$ with barycentric coordinates (α,β,γ) (black dots within highlighted face in Figure 6). In that case, let $\Pi^{l-1}(p_i) = \alpha\, p_j + \beta\, p_k + \gamma p_m$ (i.e., all vertices in Figure 6 are reparameterized in this way).

[1] In the vicinity of vertices in $\mathcal{K}^l$ a triangle $\{i,j,k\} \in \mathcal{K}^L$ can straddle multiple triangles in $\mathcal{K}^l$. In this case the map depends on the flattening strategy used (see Section 2.4).

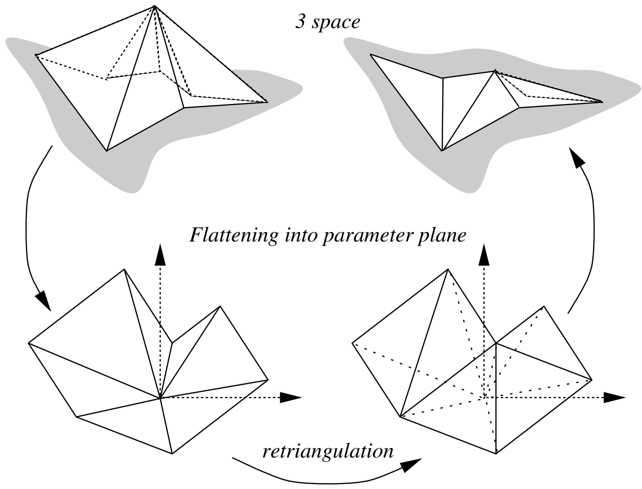

Figure 5: *In order to remove a vertex p_i, its star(i) is mapped from 3-space to a plane using the map z^a. In the plane the central vertex is removed and the resulting hole retriangulated (bottom right).*

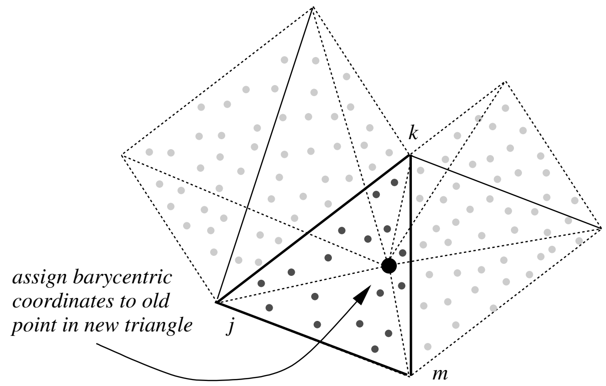

Figure 6: *After retriangulation of a hole in the plane (see Figure 5), the just removed vertex gets assigned barycentric coordinates with respect to the containing triangle on the coarser level. Similarly, all the finest level vertices that were mapped to a triangle of the hole now need to be reassigned to a triangle of the coarser level.*

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 7 visualizes the mapping we just computed. For each point p_i from the original mesh, its mapping $\Pi(p_i)$ is shown with a dot on the base domain.

Caution: Given that every association between a 1-ring and its retriangulated hole is a bijection, so is the mapping Π. However, Π does not necessarily map a finest level triangle to a triangular region in the base domain. Instead the image of a triangle may be a non-convex region. In that case connecting the mapped vertices with straight lines can cause flipping, i.e., triangles may end up on top of each other (see Figure 8 for an example). Two methods ex-

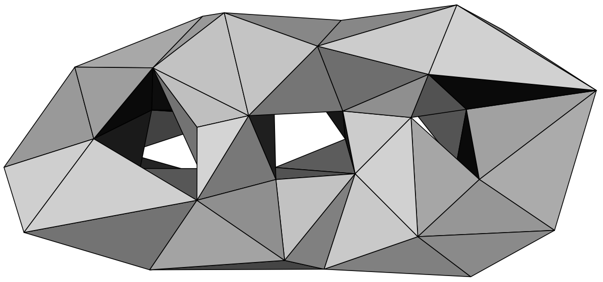

Figure 7: *Base domain $\varphi(|\mathcal{K}^0|)$. For each point p_i from the original mesh, its mapping $\Pi(p_i)$ is shown with a dot on the base domain.*

ist for dealing with this problem. First one could further subdivide the original mesh in the problem regions. Given that the underlying continuous map is a bijection, this is guaranteed to fix the problem. The alternative is to use some brute force triangle unflipping mechanism. We have found the following scheme to work well: adjust the parameter values of every vertex whose 2-neighborhood contains a flipped triangle, by replacing them with the averaged parameter values of its 1-ring neighbors [7].

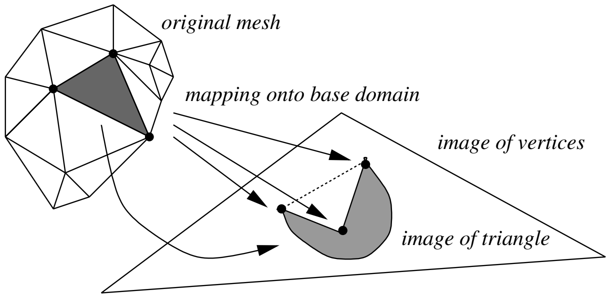

Figure 8: *Although the mapping Π from the original mesh to a base domain triangle is a bijection, triangles do not in general get mapped to triangles. Three vertices of the original mesh get mapped to a concave configuration on the base domain, causing the piecewise linear approximation of the map to flip the triangle.*

3.1 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 which one wants to preserve in the base domain. These features can either 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 $\{v_i \mid 1 \leq i \leq I\} \subset \mathcal{K}^L$ be a set of vertices on the finest level which form a path, i.e., $\{v_i, v_{i+1}\}$ is an edge. Tag all the edges in the path as feature edges. First tag v_1 and v_I, so called *dart points* [14], as unremovable so they are guaranteed to end up in the base domain. Let v_i 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_{i-1} and v_{i+1} cannot be removed and therefore must belong to $\mathcal{K}^{l-1}$. When flattening the hole around v_i, tagged edges are treated like a boundary. We first straighten out the edges $\{v_{i-1}, v_i\}$ and $\{v_i, v_{i+1}\}$ along the x-axis, and use two boundary type conformal maps to the half disk above and below (cf. the last paragraph of Section 2.4). When retriangulating the hole around v_i, we put the edge $\{v_{i-1}, v_{i+1}\}$ in $\mathcal{K}^{l-1}$, tag it as a feature edge, and compute a CDT on the upper and lower parts (see Figure 9). 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) on the base domain. This guarantees that Π maps the curved feature path onto the coarsest level edge(s) between v_1 and v_I.

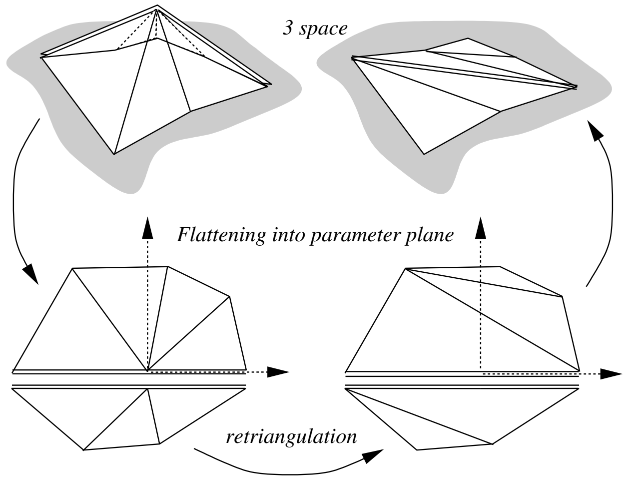

Figure 9: *When a vertex with two incident feature edges is removed, we want to ensure that the subsequent retriangulation adds a new feature edge to replace the two old ones.*

In general, there will be multiple feature paths which may be closed or cross each other. As usual, a vertex with more than 2 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 13.

3.2 A Quick Review

Before we consider the problem of remeshing, it may be helpful to review what we have at this point. We have established an initial bijection Π of the original surface $\varphi(|\mathcal{K}^L|)$ onto a base domain $\varphi(|\mathcal{K}^0|)$ consisting of a small number of triangles (e.g. Figure 7). We use a simplification hierarchy (Figure 4) in which the holes after vertex removal are flattened and retriangulated (Figures 5 and 9). Original mesh points get successively reparametrized over coarser triangulations (Figure 6). The resulting mapping is always a bijection; triangle flipping (Figure 8) is possible but can be corrected.

4 Remeshing

In this section, we consider remeshing using subdivision connectivity triangulations since it is both a convenient way to illustrate the properties of a parameterization and is an important subject in its own right. In the process, we compute a smoothed version of our initial parameterization. We also show how to efficiently construct an adaptive remeshing with guaranteed error bounds.

4.1 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 [7]: regularly (1:4) subdivide the base domain and use the inverse map to obtain a regular connectivity remeshing. This introduces a hierarchy of regular meshes $(\mathcal{Q}^m, \mathcal{R}^m)$ ($\mathcal{Q}$ is the point set and $\mathcal{R}$ is the complex) obtained from m-fold midpoint subdivision of the base domain $(\mathcal{P}^0, \mathcal{K}^0) = (\mathcal{Q}^0, \mathcal{R}^0)$. Midpoint subdivision implies that all new domain points lie *in* the base domain, $\mathcal{Q}^m \subset \varphi(|\mathcal{R}^0|)$ and $|\mathcal{R}^m| = |\mathcal{R}^0|$. All vertices of $\mathcal{R}^m \setminus \mathcal{R}^0$ have outdegree 6. The uniform remeshing of the original mesh on level m is given by $(\Pi^{-1}(\mathcal{Q}^m), \mathcal{R}^m)$.

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. We use the point location algorithm of Brown and Faigle [2] which avoids looping that can occur with non-Delaunay meshes [10, 9]. Once we have found the triangle $\{i, j, k\}$ which contains q, we can write q as

$$q = \alpha\Pi(p_i) + \beta\Pi(p_j) + \gamma\Pi(p_k),$$

and thus

$$\Pi^{-1}(q) = \alpha p_i + \beta p_j + \gamma p_k \in \varphi(|\mathcal{K}^L|).$$

Figure 10 shows the result of this procedure: a level 3 uniform remeshing of a 3-holed torus using the Π^{-1} map.

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 [15] 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).

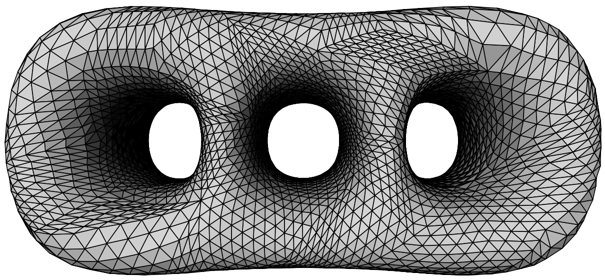

Figure 10: *Remeshing of 3 holed torus using midpoint subdivision. The parameterization is smooth within each base domain triangle, but clearly not across base domain triangles.*

4.2 Smoothing the Parameterization

It is clear from Figure 10 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. We have found computation of mappings to topological realizations that live in a high dimensional space to be needlessly cumbersome.

Instead, we use a much simpler and cheaper smoothing technique based on Loop subdivision. The main idea is to compute Π^{-1} at a smoothed version of the dyadic points, rather then at the dyadic points themselves (which can equivalently be viewed as changing the parameterization). To that end, we define a map $\mathcal{L}$ from the base domain to itself by the following modification of Loop:

- 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 $(\Pi^{-1} \circ \mathcal{L}(\mathcal{Q}^m), \mathcal{R}^m)$, where $\mathcal{Q}^m$, as before, are the dyadic points on the base domain. Figure 11 shows the result of this procedure: a level 3 uniform remeshing of a 3-holed torus using the smoothed parameterization.

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 modified versions of Loop which can deal with corners, dart points and feature edges [14, 23, 26] (see Figure 13).

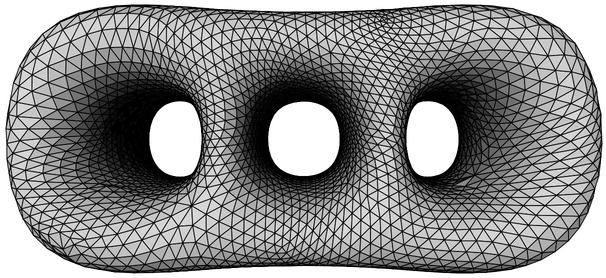

Figure 11: *The same remeshing of the 3-holed torus as in Figure 10, but this time with respect to a Loop smoothed parameterization.* **Note:** *Because the Loop scheme only enters in smoothing the* parameterization *the surface shown is still a sampling of the original mesh,* not *a Loop surface approximation of the original.*

4.3 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. [7, 8] as well as Certain et al. [3] follow a similar approach: 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, e.g., 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.

In this section we present an algorithm which avoids first building a full tree and later pruning it. Instead, we 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, we let the irregular DK hierarchy "drive" the adaptive construction of the regular pyramid.

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

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

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 $p_i \in \mathcal{P}^L$ consider the triangle t of $\mathcal{K}^0$ to which it it is currently assigned. Next consider the 4 children of t on level 1, t_j with $j = 0, 1, 2, 3$ and compute the distance between p_i and each of the $\varphi(|t_j|)$. Assign p_i 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 $i-1$ [28]. 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 (lower left), Figure 12, and Figure 13.

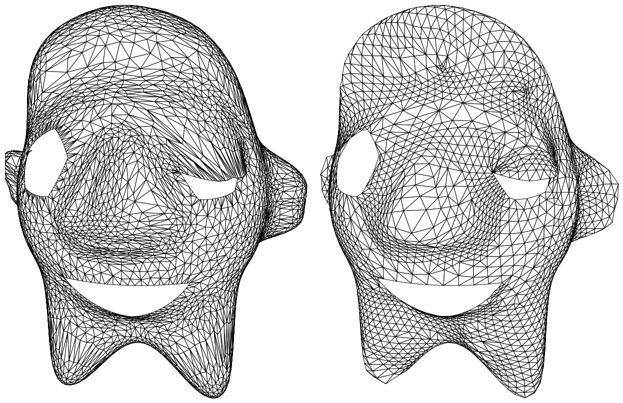

Figure 12: *Example remesh of a surface with boundaries.*

5 Results

We have implemented MAPS as described above and applied it to a number of well known example datasets, as well as some new ones. The application was written in C++ using standard computational geometry data structures, see e.g. [21], and all timings reported in this section were measured on a 200 MHz PentiumPro personal computer.

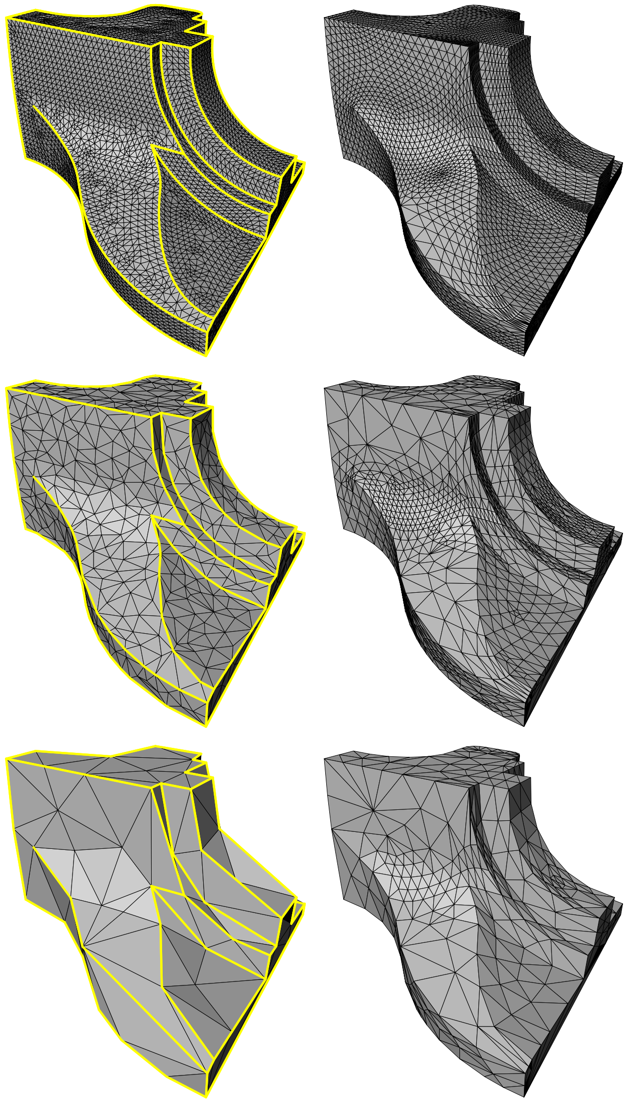

Figure 13: *Left (top to bottom): three levels in the DK pyramid, finest ($L = 15$) with 12946, intermediate ($l = 8$) with 1530, and coarsest ($l = 0$) with 168 triangles. Feature edges, dart and corner vertices survive on the base domain. Right (bottom to top): adaptive mesh with $\varepsilon = 5\%$ and 1120 triangles (bottom), $\varepsilon = 1\%$ and 3430 triangles (middle), and uniform level 3 (top). (Original dataset courtesy University of Washington.)*

The first example used throughout the text is the 3-holed torus. The original mesh contained 11776 faces. These were reduced in the DK hierarchy to 120 faces over 14 levels implying an average removal of 30% of the faces on a given level. The remesh of Figure 11 used 4 levels of uniform subdivision for a total of 30720 triangles.

The original sampled geometry of the 3-holed torus is smooth and did not involve any feature constraints. A more challenging case is presented by the fandisk shown in Figure 13. The original mesh (top left) contains 12946 triangles which were reduced to 168 faces in the base domain over 15 levels (25% average face removal per level). The initial mesh had all edges with dihedral angles below

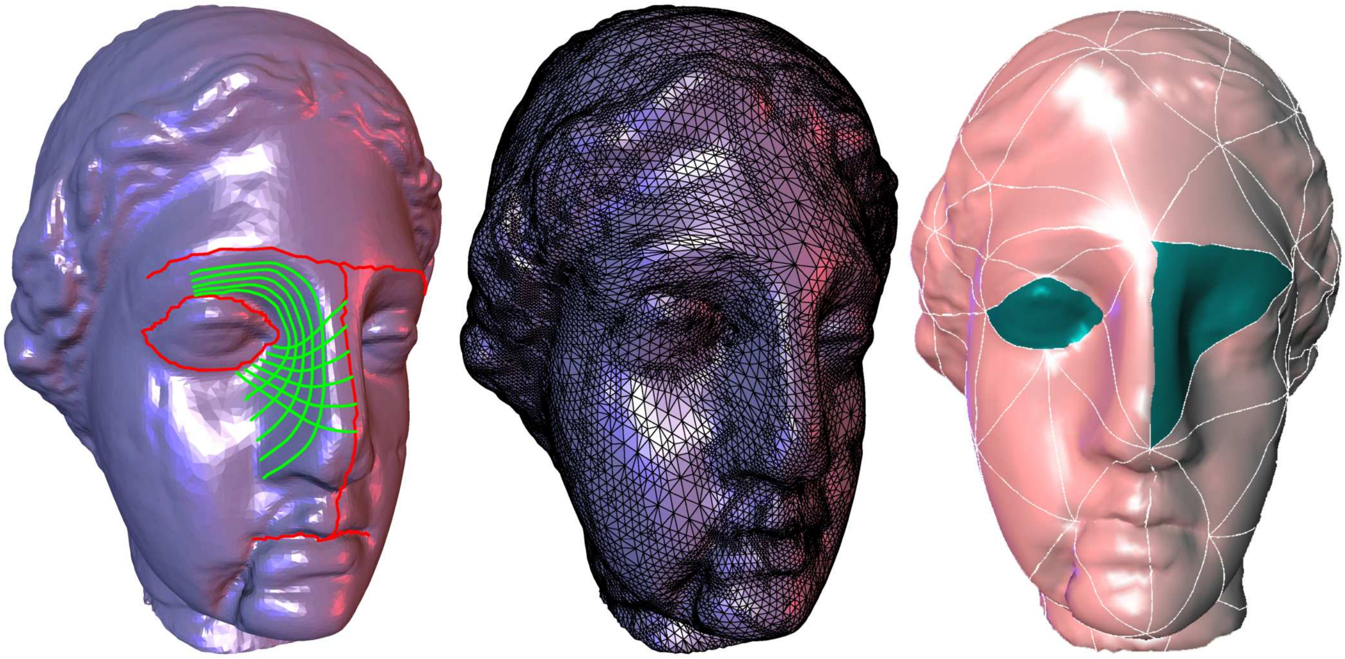

Figure 14: *Example of a constrained parameterization based on user input. Top: original input mesh (100000 triangles) with edge tags superimposed in red, green lines show some smooth iso-parameter lines of our parameterization. The middle shows an adaptive subdivision connectivity remesh. The bottom 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.)*

75^o tagged (1487 edges), resulting in 141 tagged edges at the coarsest level. Adaptive remeshing to within $\varepsilon = 5\%$ and $\varepsilon = 1\%$ (fraction of longest bounding box side) error results in the meshes shown in the right column. The top right image shows a uniform resampling to level 3, in effect showing iso-parameter lines of the parameterization used for remeshing. Note how the iso-parameter lines conform perfectly to the initially tagged features.

This dataset demonstrates one of the advantages of our method—inclusion of feature constraints—over the earlier work of Eck et al. [7]. In the original PM paper [12, Figure 12], Hoppe shows the simplification of the fandisk based on Eck's algorithm which does not use tagging. He points out that the multiresolution approximation is quite poor at low triangle counts and consequently requires many triangles to achieve high accuracy. The comparison between our Figure 13 and Figure 12 in [12] demonstrates that our multiresolution algorithm which incorporates feature tagging solves these problems.

Another example of constrained parameterization and subsequent adaptive remeshing is shown in Figure 14. The original dataset (100000 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. Those can be used for computing texture coordinates. The middle image shows an adaptive subdivision connectivity remesh with 74698 triangles ($\varepsilon = 0.5\%$). On the right we have highlighted a group of patches, 2 over the right (constrained) eye and 1 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). This example illustrates how one places constraints like Krishnamurthy and Levoy [17]. We remove the need in their algorithms to specify the entire base domain. A user may want to control patch outlines for editing in one region (e.g., on the face), but may not care about what happens in other regions (e.g., the back of the head).

We present a final example in Figure 1. The original mesh (96966 triangles) is shown on the top left, with the adaptive, subdivision connectivity remesh on the bottom left. This remesh was subsequently edited in a interactive multiresolution editing system [28] and the result is shown on the bottom middle.

6 Conclusions and Future Research

We have described an algorithm which establishes smooth parameterizations for irregular connectivity, 2-manifold triangular meshes of arbitrary topology. Using a variant of the DK hierarchy construction, we simplify the original mesh and use piecewise linear approximations of conformal mappings to incrementally build a parameterization of the original mesh over a low face count base domain. This parameterization is further improved through a hierarchical smoothing procedure which is based on Loop smoothing in parameter space. The resulting parameterizations are of high quality, and we demonstrated their utility in an adaptive, subdivision connectivity remeshing algorithm that has guaranteed error bounds. The new meshes satisfy the requirements of multiresolution representations which generalize classical wavelet representations and are thus of immediate use in applications such as multiresolution editing and compression. Using edge and vertex constraints, the parameterizations can be forced to respect feature lines of interest without requiring specification of the entire patch network.

In this paper we have chosen remeshing as the primary application to demonstrate the usefulness of the parameterizations we produce. The resulting meshes may also find application in numerical analysis algorithms, such as fast multigrid solvers. Clearly there

Dataset	Input size (triangles)	Hierarchy creation	Levels	$\mathcal{P}^0$ size (triangles)	Remeshing tolerance	Remesh creation	Output size (triangles)
3-hole	11776	18 (s)	14	120	(NA)	8 (s)	30720
fandisk	12946	23 (s)	15	168	1%	10 (s)	3430
fandisk	12946	23 (s)	15	168	5%	5 (s)	1130
head	100000	160 (s)	22	180	0.5%	440 (s)	74698
horse	96966	163 (s)	21	254	1%	60 (s)	15684
horse	96966	163 (s)	21	254	0.5%	314 (s)	63060

Table 1: *Selected statistics for the examples discussed in the text. All times are in seconds on a 200 MHz PentiumPro.*

are many other applications which benefit from smooth parameterizations, e.g., texture mapping and morphing, which would be interesting to pursue in future work. Because of its independent set selection the standard DK hierarchy creates topologically uniform simplifications. We have begun to explore how the selection can be controlled using geometric properties. Alternatively, one could use a PM framework to control geometric criteria of simplification. Perhaps the most interesting question for future research is how to incorporate topology changes into the MAPS construction.

Acknowledgments

Aaron Lee and David Dobkin were partially supported by NSF Grant CCR-9643913 and the US Army Research Office Grant DAAH04-96-1-0181. Aaron Lee was also partially supported by a Wu Graduate Fellowship and a Summer Internship at Bell Laboratories, Lucent Technologies. Peter Schröder was partially supported by grants from the Intel Corporation, the Sloan Foundation, an NSF CAREER award (ASC-9624957), a MURI (AFOSR F49620-96-1-0471), and Bell Laboratories, Lucent Technologies. Special thanks to Timothy Baker, Ken Clarkson, Tom Duchamp, Tom Funkhouser, Amanda Galtman, and Ralph Howard for many interesting and stimulation discussions. Special thanks also to Andrei Khodakovsky, Louis Thomas, and Gary Wu for invaluable help in the production of the paper. Our implementation uses the triangle facet data structure and code of Ernst Mücke.

References

[1] Bajaj, C. L., Bernadini, F., Chen, J., and Schikore, D. R. Automatic Reconstruction of 3D CAD Models. Tech. Rep. 96-015, Purdue University, February 1996.

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

[3] 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.

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

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

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

[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)*, 173–182, 1995.

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

[9] 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.

[10] 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.

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

[12] Hoppe, H. Progressive Meshes. In *Computer Graphics (SIGGRAPH 96 Proceedings)*, 99–108, 1996.

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

[14] 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.

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

[16] 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.

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

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

[19] Lounsbery, M. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type*. PhD thesis, Department of Computer Science, University of Washington, 1994.

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

[21] Mücke, E. P. Shapes and Implementations in Three-Dimensional Geometry. Technical Report UIUCDCS-R-93-1836, University of Illinois at Urbana-Champaign, 1993.

[22] Schröder, P., and Sweldens, W. Spherical Wavelets: Efficiently Representing Functions on the Sphere. In *Computer Graphics (SIGGRAPH 95 Proceedings)*, Annual Conference Series, 1995.

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

[24] Spanier, E. H. *Algebraic Topology*. McGraw-Hill, New York, 1966.

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

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

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

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