Globally Smooth Parameterizations with Low Distortion

Andrei Khodakovsky
Caltech

Nathan Litke
Caltech

Peter Schröder
Caltech

Abstract

Good parameterizations are of central importance in many digital geometry processing tasks. Typically the behavior of such processing algorithms is related to the smoothness of the parameterization and how much distortion it contains. Since a parameterization maps a bounded region of the plane to the surface, a parameterization for a surface which is not homeomorphic to a disc must be made up of multiple pieces. We present a novel parameterization algorithm for arbitrary topology surface meshes which computes a *globally* smooth parameterization with low distortion. We optimize the patch layout subject to criteria such as shape quality and metric distortion, which are used to steer a mesh simplification approach for base complex construction. Global smoothness is achieved through simultaneous relaxation over all patches, with suitable transition functions between patches incorporated into the relaxation procedure. We demonstrate the quality of our parameterizations through numerical evaluation of distortion measures and the excellent rate distortion performance of semi-regular remeshes produced with these parameterizations. The numerical algorithms required to compute the parameterizations are robust and run on the order of minutes even for large meshes.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

Keywords: Parameterization, compression, resampling, smoothness, rate distortion.

1 Introduction

Many algorithms which process digital geometry, such as surface meshes, require that a parameterization of the mesh be established. Examples include texture mapping [Lévy et al. 2002], morphing [Lee et al. 1999], editing [Biermann et al. 2002], approximation [Lee et al. 2000], and compression [Khodakovsky et al. 2000], among many others.

Due to the importance of parameterizations, many algorithms for this task have been proposed over the years. Early work by Eck and co-workers [1995] formulated the parameterization problem as the minimizer of an energy which measures the quality of the parameterization. They also proposed a procedure for tiling an arbitrary topology surface so as to produce a number of regions, each homeomorphic to a disc. The topological relations between these patches are encoded by a *base complex*, which governs the patch layout and forms the parametric domain of the surface. The MAPS algorithm [Lee et al. 1998] was the first to use mesh simplification [Hoppe 1996] to build the base complex. Given a base complex of the same topology as the original surface, parameterizations can be built in a number of different ways. For example, Wood and co-workers [2000] used an inflating balloon analogy, while Guskov *et al.* [2000] used a recursive piercing procedure. These approaches are global, in that they build a parameterization everywhere at once. Most other work—starting with Eck *et al.* [1995]—iteratively improves local parameterizations by cycling over sub-regions of the mesh, making local improvements each time (*e.g.*, [Guskov et al. 2002]).

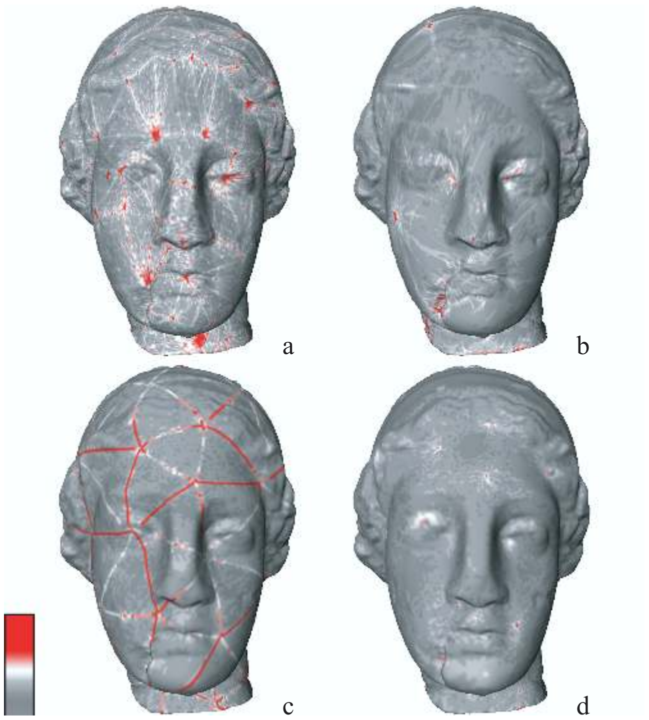


Figure 1: Parameterizations produced by (a) MAPS [1998], (b) Normal Meshes [2000], (c) Hybrid Meshes [2002] and (d) our new method, visualizing derivative magnitudes of the tangent field (gray: small; red: large). Note the discontinuities visible at patch boundaries, which are remedied by our *global* parameterization method.

One of the distinctions between different algorithms is the energy which they minimize. Sander *et al.* [2001] compared a variety of distortion measures with an eye towards controlling texture blur (see also [Sander et al. 2002]). Some approaches define discrete measures [Maillot et al. 1993] while others are based on continuous energies with particular analytic properties [Pinkall and Polthier 1993; Eck et al. 1995; Desbrun et al. 2002; Lévy et al. 2002]. Floater [1997; 2003] has focused on deriving local coordinates which are always guaranteed to be positive, to ensure injectivity of the solution. In all of these cases the ultimate goal is the minimization of the unavoidable distortion when mapping a curved surface to the plane.

Our interest lies in building *globally* smooth parameterizations with low distortion for use in the remeshing of arbitrary topology surfaces. The most important ingredients in this are the *construction of the base complex* and *control of smoothness and distortion*, in particular at patch boundaries (see [Gu and Yau 2003] for an alternative approach). A good patch layout, for example, can greatly decrease the distortion in the parameterization. As an extreme case consider the original mesh itself serving as the parameterization domain. In this case the parameterization would have no distortion at

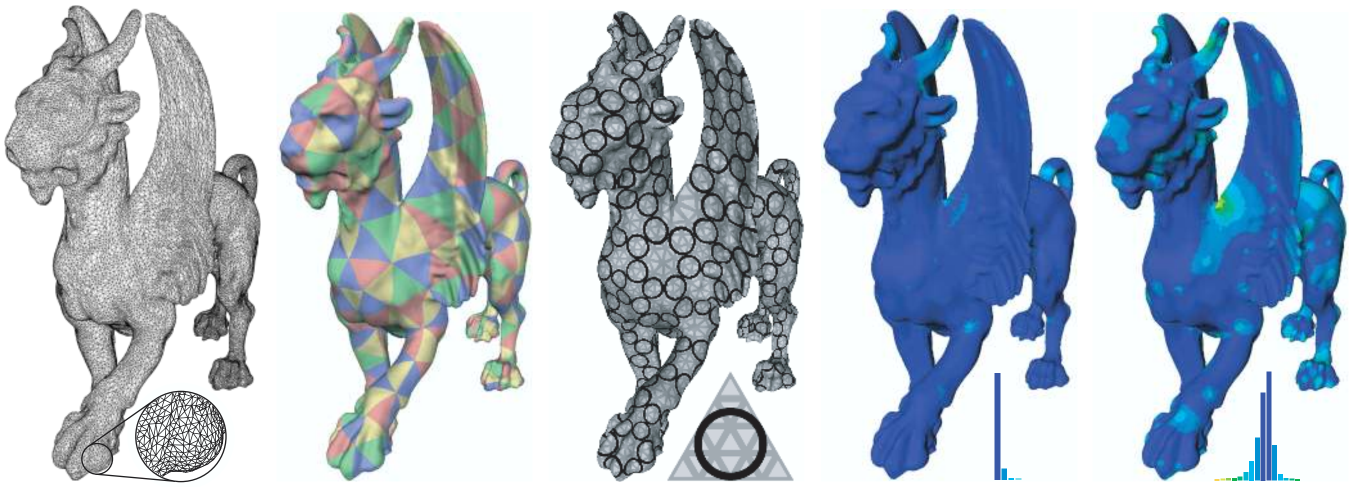


Figure 2: Our algorithm takes an input mesh and generates a base complex for a *globally* smooth parameterization, illustrated with a texture map. The base complex and subsequent parameterization relaxation reduce distortion, shown as anisotropic and area distortion, respectively.

all at the cost of a very large base complex. The other extreme would parameterize the entire surface over a single patch [Gu et al. 2002; Sheffer and Hart 2002] at the cost of having to introduce seams to map a higher genus surface to a disc. Spherical topology does not require cutting, though distortion may be difficult to control (see [Praun and Hoppe 2003; Gotsman et al. 2003] in these proceedings).

A globally smooth parameterization with little distortion lies somewhere between these extremes, balancing the trade-off between a small base complex and low distortion. We were motivated to find such parameterizations by noticing that different *remeshers* can produce markedly different rate distortion (r/d) compression performance for the same object. For example, the normal remesher of Guskov *et al.* [2000] consistently produced the best remeshes. Unfortunately it can only deal with closed surfaces. The alternatives, parameterization algorithms which treat sections of the surface one at a time, tend to suffer from (visible) smoothness artifacts at patch boundaries, which we seek to avoid (Figure 1).

To compare different parameterizations we score them according to their r/d performance. Aside from measuring a quantity relevant in compression applications, r/d performance also captures a measure of smoothness in the discrete surface setting. It is well known from wavelet theory [Strang and Nguyen 1996] that the decay rate of wavelet coefficients characterizes the smoothness class of a given function. Smooth functions have rapidly decaying coefficients leading to better r/d performance. The geometric smoothness of a surface depends, of course, on the input. However, for a given smooth surface there are many possible parameterizations. The coder "sees" the parameterization, not the original surface. In this way its performance becomes a measure of the quality (smoothness and distortion) of the parameterization (Figure 7).

Contributions We present a novel algorithm for the automatic generation of parameterizations, which are smooth across patch boundaries and whose distortion varies gradually over the entire surface. To accomplish this goal we focus on the construction of patch layouts and global parameterization computations:

- the base complex combinatorial structure and its layout are optimized through the use of weighting criteria which take into account sources of local distortion during mesh simplification;
- the parameterization is relaxed globally and smoothness between patches is ensured by a novel set of transition functions.

At present we do not have a theory from which a guaranteed set of simplification criteria can be derived to ensure optimal r/d performance. Instead, we have run extensive experiments with different criteria to develop a set of heuristics that work well in practice. We show results of applying our algorithm to a number of example data sets for the purpose of remeshing, and numerically evaluate distortion measures as well as the r/d performance of these meshes in a progressive geometry encoder. We generally achieve better remeshing error tolerances and better r/d performance, at times markedly so. All numerical computations can be performed on the order of minutes and require only minimal guidance from the user.

2 Smoothness and Distortion

Since the notions of smoothness and distortion are central to our algorithm we begin with their definition.

Smooth Parameterizations A smooth 2-manifold S is a topological 2-manifold for which a smooth atlas of overlapping coordinate charts $\{U_\alpha\}$ is defined [Grimm and Hughes 1995]. Each chart U_α is parameterized over the region $V_\alpha \subset \mathbb{R}^2$ by a parameterization function $\phi_\alpha : V_\alpha \mapsto U_\alpha$. Each ϕ_α is a bijection, and we denote its inverse by ψ_α. A *transition function* $\tau_{\alpha\beta} : V_{\alpha\beta} \mapsto \mathbb{R}^2$ is defined on the intersection of U_α and U_β, where $V_{\alpha\beta} = \phi_\alpha^{-1}(U_\alpha \cap U_\beta)$. The smoothness of the manifold is determined by the smoothness of the transition functions.

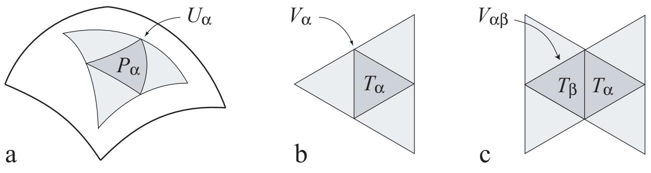


Figure 3: (a) A chart U_α is defined as a patch P_α and its three neighboring patches. (b) A patch P_α and a chart U_α are the images of triangles T_α and V_α, respectively. (c) The intersection of V_α and V_β defines the domain $V_{\alpha\beta}$ where the transition function $\tau_{\alpha\beta}$ acts.

We partition S into triangular patches P_α and define the chart U_α to be the union of P_α and its three edge neighbors (Figure 3a). Each chart U_α is mapped to the set V_α composed of four equilateral triangles, with the center triangle T_α corresponding to P_α (Figure 3b). For any two neighboring patches P_α and P_β, the intersection $U_\alpha \cap U_\beta$ equals the union $P_\alpha \cup P_\beta$ (Figure 3c). Therefore, the transition function $\tau_{\alpha\beta}$ maps $V_\alpha \cap V_\beta \mapsto V_\alpha \cap V_\beta$, where V_α (resp. V_β) is parameterized by barycentric coordinates (u, v, w) with respect to T_α (resp. T_β). Note that some of these coordinates are negative over $V_\alpha \backslash T_\alpha$ (resp. $V_\beta \backslash T_\beta$). Later on we will drop the charts U_α and refer only to their patches P_α. Consequently, transition functions between patches should be understood as transition functions on their associated charts.

We will call a parameterization *globally smooth* if lines of constant parameter value cross patch boundaries smoothly, *i.e.*, the tangent vectors along the iso-parameter lines of one chart transform into tangent vectors along the same lines of the other chart, up to some permutation of the coordinates. The smoothness condition forces the derivatives of the transition functions to be ± 1, and we force $\tau_{\alpha\beta}$ to be the identity function at the common boundary between T_α and T_β. Later on we require our transition functions to be linear. This assumption is not restrictive, because we will only apply them near the boundary between adjacent patches. Under these conditions, the only non-trivial transition function is $\tau_{\alpha\beta} = (-u, 1 - v)$, where we have chosen $u = 0$ for the boundary. This solution has a simple geometric interpretation: it takes the point $(u, v)_\alpha$ with respect to T_α and expresses it as $(u, v)_\beta$ with respect to T_β.

Distortion Measures An embedding ϕ of a manifold in $\mathbb{R}^3$ defines a metric $g_{ij} = \langle \partial_i \phi, \partial_j \phi \rangle$, $i, j \in \{u, v\}$, which expresses the notions of angle, distance and area on the surface. This metric can also be used to define a measurement of the local *distortion* of these quantities. Let $\gamma_{\max}$ and $\gamma_{\min}$ be the largest and smallest eigenvalues of the metric tensor: these values provide bounds on the local distortion in ϕ. Sander *et al.* [2001] define an L^2 distortion measure by taking the root-mean-square of the two values, and define $L^\infty = \gamma_{\max}$. Conversely, Sorkine *et al.* [2002] define their geometric distortion as $\max\{\gamma_{\max}, 1/\gamma_{\min}\}$. We chose a qualitatively equivalent, but more intuitive, pair of distortion measures: $\Gamma_a = \gamma_{max}/\gamma_{min}$ [Hormann and Greiner 2000], which measures the amount of *anisotropic distortion*, and $\Gamma_d = \gamma_{max}\gamma_{min} = \det g$, which measures *area distortion*. Γ_a is a natural measurement for the shear in the parameter function ϕ since it is scale-invariant. In our comparisons, we normalize Γ_d by the total area of the surface (for example, in Figure 2).

3 Algorithms

The parameterization process typically consists of two steps. First, the original input surface mesh is partitioned into disc-like regions called *patches*. Second, for each patch, a map into the parameter domain is computed. We use triangles as patch shapes, but quadrilaterals are possible as well [Guskov et al. 2002]. For remeshing, a third step follows: the parameter domain is refined and the surface is sampled at associated parametric locations until some remeshing error tolerance is met.

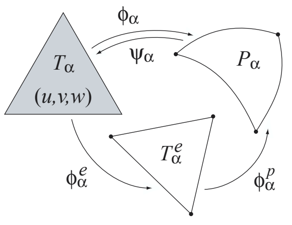
For the construction of the base complex, *i.e.*, the number of patches and their combinatorial structure, we follow MAPS [1998]: a copy of the input mesh serves as an initial parametric domain with the initial parameterization being the identity. The domain is then successively simplified until the base complex is as simple as possible without violating quality measures (Section 3.1). The quality measures are responsible for controlling the distortion in the parameterization (Section 3.1.1).

During simplification the initial parameterization is incrementally updated as in MAPS, carefully ensuring a bijection throughout the process. At the end of the simplification stage, each vertex in the input mesh is associated to some base complex triangle with an appropriate barycentric coordinate. This parameterization is not yet smooth, but serves as a valid starting point for a relaxation process which builds the final smooth parameterization (see Section 3.2).

Before moving on to the algorithm details we will fix some notation that will be useful throughout.

Notation The input to our algorithm is a *mesh* $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$ with vertices, edges, and faces. The mesh is a topological 2-manifold, possibly with boundary. We assume that all faces are triangles and that each vertex carries a point position in $\mathbb{R}^3$: $v_i \mapsto p_i \in S$, which samples the surface S. These point positions induce an embedding $E(\mathcal{M}) \subset \mathbb{R}^3$ of the mesh as a piecewise linear surface. In practice this embedding often has self-intersections and "bad" geometry, *e.g.*, tiny triangles with noisy positions and poor aspect ratios.

During base domain construction it will be useful to think of ϕ_α as being composed of two functions $\phi_\alpha = \phi^p_\alpha \circ \phi^e_\alpha$. The first linearly embeds a domain triangle into $\mathbb{R}^3$, $\phi^e_\alpha : T_\alpha \mapsto T^e_\alpha$. The second function completes the mapping to the surface, $\phi^p_\alpha = \phi_\alpha \circ (\phi^e_\alpha)^{-1}$. The linear embedding $\phi^e_\alpha(T_\alpha)$ is defined by the three point positions on the surface which are associated with the vertices of T_α. These can be thought of as the *corners* of the patch P_α.



3.1 Base Domain Generation

We generate the patch layout through a sequence of topological simplifications of an initial domain complex, $\mathcal{T}^0 = \mathcal{M}$. Using the standard greedy approach, $\mathcal{T}^{i+1}$ is produced from $\mathcal{T}^i$ through a least cost simplification step [Hoppe 1996; Garland and Heckbert 1997]. Following MAPS [1998], we use *vertex removal* followed by the retriangulation of the hole left by the removed vertex and its incident triangles $\mathcal{N}^i \subset \mathcal{T}^i$ (Figure 4a). During retriangulation, any vertices in M whose barycentric coordinates are associated with a triangle in $\mathcal{N}^i$ receive new barycentric coordinates with respect to the triangles $\mathcal{R}^i$ filling the hole (Figure 4b). The new coordinates are computed by flattening $\mathcal{N}^i$ in the plane using the exponential map z^θ, and assigning new coordinates with respect to $\mathcal{R}^i$ [Lee et al. 1998]. This step maintains valid ψ_α mappings. An invariant of this procedure is that each vertex present in the base complex maps to some vertex in the input mesh, *i.e.*, ϕ^e_α is always well defined.

There are many possible retriangulations of a given hole and we choose one that admits little distortion under z^θ. If no good triangulation of this type exists, the vertex is not considered for removal.

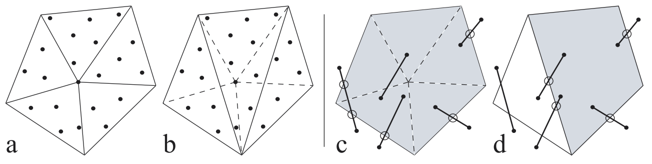


Figure 4: At each simplification step, (a) a vertex and its incident triangles $\mathcal{N}$ are removed, (b) the resulting hole is filled with new triangles $\mathcal{R}$, and each vertex mapped to $\mathcal{N}$ is reassigned to a triangle in $\mathcal{R}$. (c) The edge classification algorithm incrementally updates the list of patches crossed by an edge (d) by examining the crossings one patch at a time as the hole is retriangulated.

3.1.1 Priority Criteria

Standard mesh simplification methods are geared towards maintaining closeness in $\mathbb{R}^3$ between S and the piecewise linear embedding $\phi^e(\mathcal{T})$. In our setting closeness of embeddings is not the relevant

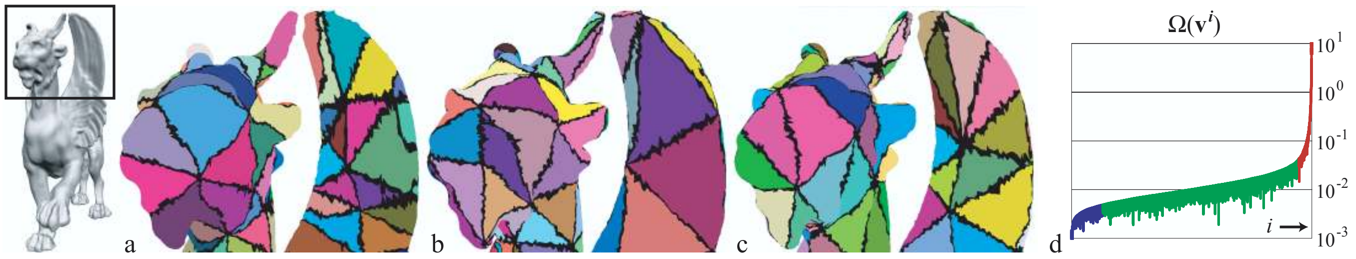


Figure 5: An example of base complexes that optimize (a) triangle quality, (b) metric distortion, and (c) our cost function Ω, which combines the two. The edges which cross patch boundaries are highlighted to emphasize the patches generated by the simplification algorithm. (d) Plotting the cost function Ω reveals three distinct stages during simplification. We find our ideal stopping point before the third stage.

property to track. Instead we care about the ϕ_α having *low distortion*. Some decrease in the amount of distortion can be achieved with the subsequent relaxation algorithm. However, the bulk of the distortion is controlled through the choice of $\mathcal{T}$ and the initial ϕ_α. For example, consider the model of a horse (Figure 8): in principle, a tetrahedron is sufficient to parameterize the surface, but great distortion would have to be tolerated in the legs. Guskov *et al.* [2002] considered this the "skin growth" problem and dealt with it through the introduction of additional base complex pieces during hierarchical refinement. Since our goal is to have a single global parameterization we cannot follow their path.

Ideally, each domain triangle $T_\alpha \in \mathcal{T}$ should map to a patch P_α of "equilateral" shape, with neighboring patches having smoothly varying area. Shewchuk [2002] gives an in-depth discussion of different choices of triangle shapes for approximation. In particular he finds that the approximation error between a function and its piecewise linear approximation is provably minimized if triangles are allowed to be anisotropic and align with curvature directions. Standard mesh simplification criteria such as error quadrics [Garland and Heckbert 1997] tend to favor such triangles and corresponding alignments. However, once gradients of functions are taken into account the situation changes. In this case, both small and large angles in the triangulation are detrimental.

The use of additional criteria for surface approximation was considered by Kobbelt and co-workers [1998]. One of their criteria, the *roundness* of a triangle, also works well for scoring surface parameterizations, an observation which matches Shewchuk's theory in the function approximation setting. During simplification the algorithm of course cannot anticipate the exact properties of the final ϕ_α, so any priority criteria are necessarily heuristic.

To produce good parameterizations $\phi_\alpha = \phi^p_\alpha \circ \phi^e_\alpha$, we consider the effect of ϕ^p_α and ϕ^e_α separately. The *triangle quality* is controlled by ϕ^e_α while the initial *metric distortion* is captured by ϕ^p_α. We rate these functions according to the distortion that they contribute to ϕ_α, and combine these ratings in an oracle which guides the simplification algorithm.

Triangle Quality There are two factors which contribute to the distortion in ϕ^e_α. The *shape* of the triangle $T^e_\alpha = \phi^e_\alpha(T_\alpha)$ affects the gradient inside T_α, and the *area dispersion* between T^e_α and its neighbors T^e_β affects the gradient between the triangles. Both factors are minimized when the triangles are equilateral in $\mathbb{R}^3$. We measure the *shape distortion* $\sigma(T_\alpha)$ by the inverse of the roundness of T^e_α, *i.e.*, the ratio of the longest edge length to the inner circle radius, normalized so that $\sigma(T_\alpha) = 1$ when T^e_α is equilateral. The *area dispersion* $\rho(T_\alpha)$ is calculated as the largest ratio of area between T^e_α and its neighbors T^e_β. Optimizing the patch layout based on triangle quality alone tends to produce nearly equilateral triangles, however the patches do not respect the features of the surface well, as seen in Figure 5a.

Metric Distortion The parameterization ϕ_α generated during simplification gives a conservative estimate of the distortion in the final parameterization computed by our method. This distortion can be measured, for example, on the edges $\mathbf{e}$ which are incident to a vertex with coordinates in T_α. The *metric distortion* $\delta(T_\alpha)$ accumulates the ratio of edge lengths,

$$\delta(T_\alpha) = \sum_{\mathbf{e} \in P_\alpha} \max\left(\frac{\|\phi_\alpha(\mathbf{e})\|}{\|\phi^e_\alpha(\mathbf{e})\|}, \frac{\|\phi^e_\alpha(\mathbf{e})\|}{\|\phi_\alpha(\mathbf{e})\|}\right)^2 \cdot \|\phi_\alpha(\mathbf{e})\|$$

normalized by the sum of edge lengths $\|\phi_\alpha(\mathbf{e})\|$. Patch layouts optimized for metric distortion tend to align patches with flat regions, but also contain long, thin triangles (Figure 5b).

Oracle Given a candidate vertex $\mathbf{v}$, we assign a cost to it by evaluating our heuristic functions for each triangle T_α in the retriangulation $\mathcal{R}$ of the hole left by $\mathbf{v}$. To balance the influence of these functions, we shift them to zero, and introduce a set of coefficients to weight them appropriately. We compute the cost $\Omega(\mathbf{v})$ as

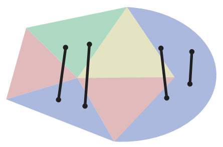
$$\Omega(\mathbf{v}) = \sum_{T_\alpha \in \mathcal{R}} A(T_\alpha) \cdot (\gamma\sigma(T_\alpha)^2 + \mu\rho(T_\alpha)^2 + \delta(T_\alpha)^2)$$

where $A(T_\alpha)$ is the area of P_α on S. Since we have no precise definition of the surface area of a patch, we estimate this quantity by summing the areas of the Voronoi regions of the vertices in P_α, which are computed in a pre-processing step. We find that the coefficient values $\gamma = 2$ and $\mu = 1$ work well in practice. This balance tends to produce patch layouts that align to the features of the object, and tile the surface with patches that change shape gradually (Figure 5c).

The vertex with the lowest cost $\Omega(\mathbf{v}^i)$ is selected to produce $\mathcal{T}^{i+1}$. A plot of the costs $\Omega(\mathbf{v}^i)$ identifies three distinct stages in the simplification (Figure 5d). The first stage quickly corrects ill-conditioned triangles in the input mesh $\mathcal{M}$, and the second stage begins a slow linear ascent as the triangle quality dominates the cost. The final stage is dominated by metric distortion, and we have observed that stopping the simplification before this stage produces the least distortion in our experiments.

3.1.2 Edge Classification

In preparation for the global parameterization solver, we augment the information produced by the MAPS algorithm with *edge classifications*. At the end of the simplification stage, every vertex in the input mesh has barycentric coordinates with respect to some domain triangle in the base complex $\mathcal{T}$, but this information is not sufficient to determine which domain triangles are crossed by a given edge. The end points of an edge may map to the same base domain triangle, or they may map to different triangles. In the latter case, the triangles may be neighbors or they may be further apart.


Edges cannot be classified by their end points alone.

There may even be edges with both end points contained in the same triangle, yet the edge traverses a number of intermediate triangles. These situations are likely to appear near patch corners. Knowledge of the sequence of base domain triangles traversed between two end points of an edge will be important for the compositions of transition functions.

To address this problem we classify all edges during the simplification process. For each edge, we keep the ordered list of domain triangles which are traversed when going from one end point to the other. This information is purely topological, *no geometry is involved*. Most edges will have only one entry in their list: their beginning and end points are mapped to the same domain triangle, and the edge crosses no others. Other edges may have longer lists. For each edge, the transition list defines a sequence of *triangle crossings*, *i.e.*, two adjacent triangles in the list define a crossing over the boundary they share. In addition to the transition lists associated with edges, each domain triangle maintains pointers to the edges whose transition lists contain the given triangle. During simplification, the transition lists are updated incrementally with the help of these pointers.

Incremental Update of Transition Lists Recall that an elementary simplification step removes a vertex and its incident triangles $\mathcal{N}$, and retriangulates the hole with a set of new triangles $\mathcal{R}$. Any vertices with barycentric coordinates in $\mathcal{N}$ will receive new assignments in $\mathcal{R}$. Accordingly, all edges pointed to by the triangles in $\mathcal{N}$ must have their transition lists updated. Note that among these may be edges with both end points outside of $\mathcal{N}$ (Figure 4c).

We prepare to update the edge transition lists by removing from them the triangles in $\mathcal{N}$. Edges whose triangle crossings include a boundary of the hole will be left with at most two partial list segments, since in practice edges with more than two boundary crossings never occur. The rest of the edges now have empty transition lists. Next, we *advance* the transition lists for all the edges that need updating.

A given triangulation of the hole (a simple polygon) always has the property that we can clip off an "ear", *i.e.*, the retriangulation can be split into a single triangle with two edges on the boundary, and a remainder. We incrementally reconstruct the transition lists by processing such an "ear", and then recurse on the remainder of the triangulation.

Advancing Transition Lists Across a Triangle The selected triangle $T_\alpha \in \mathcal{R}$ has two boundaries, which are annotated with the working set of edge transition lists that need to be incremented. T_α is appended to each such list, and we check the working set for completed transition lists. If the end point of a given list is in T_α, the list is complete, otherwise the edge transitions out of T_α and crosses either one of the other two boundaries (Figure 4d). If it crosses the other boundary edge, it must match against a partial list associated with that boundary, and we connect those lists. Otherwise the list is advanced to the "open" boundary. In addition, there may be new edges which begin in T_α. Their lists are added to the working set and associated with the open boundary if their end point does not also belong to T_α. At this stage, we have processed all the relevant transition lists, and each edge crossing the boundary of the (remaining) open hole has a partial list associated to it. The algorithm now recurses over the next "ear", until the hole is filled. Note that at no point are actual geometric intersection computations required: *all triangle crossings are purely topological*. This observation is important since the triangle boundaries do not even possess an embedding.

3.2 Parameterization

Our goal is to compute a globally smooth parameterization of the surface. Traditionally, parameter values are fixed on the patch boundaries and computed inside each patch as a solution of a linear system [Desbrun et al. 2002; Lévy et al. 2002]. This approach produces *continuous* parameterizations since neighboring patches use the same parameter values on the boundary, but in general the result is not smooth. Boundary relaxation can be employed to improve the quality, for example, by smoothing the parameter function in the direction orthogonal to the boundary [Guskov et al. 2002]. But boundary relaxation cannot make both parameter functions smooth, as illustrated in Figure 1c.

We take a different approach: we do not fix patch boundaries. Instead, we solve for parameter functions on all patches simultaneously. The boundary values are fixed only at patch corners and on open boundaries, if any, of the input mesh.

3.2.1 Global Parameterization System

Each vertex of the original mesh is assigned to be either a patch corner, *i.e.*, it maps to one of the base domain vertices, or it belongs to exactly one patch. This is true even of vertices which map exactly to a triangle boundary in the base domain. Therefore the complete description of the parameterization at a non-corner vertex $\mathbf{v}_i$ is a triple $\psi(\mathbf{v}_i) = (u, v, \alpha)$, where α indexes a triangle in the base domain, and correspondingly the patch P_α to which $\mathbf{v}_i$ belongs. A corner vertex is assigned parameter values in each of its incident patches.

A wide class of parameterization schemes can be written in the form:

$$(u_i, v_i) = \sum_{j \in \text{1-ring}(i)} c_{ij} \cdot (u_j, v_j) \qquad (1)$$

where (u_i, v_i) are parameter values at vertex $\mathbf{v}_i$ and c_{ij} are scalar coefficients which depend on the geometry of the original mesh. The various parameterization schemes differ in the way they define these coefficients. Our approach can be applied to any of these schemes. We chose the mean value parameterization [Floater 2003], because it always produces positive weights c_{ij}.

The parameterization equation (1) is a linear combination of parameter variables over the 1-ring neighborhood of a given vertex. In general, these parameters may belong to different patches, and in this situation taking linear combinations is meaningless. Instead, we use the transition functions $\tau_{\alpha\beta}$ which express barycentric coordinates in one patch with respect to a neighboring patch. Using these transition functions, we obtain a *global parameterization system*

$$(u_i, v_i, \alpha_i) = \sum_{j \in \text{1-ring}(i)} c_{ij} \cdot \tau_{ij}(u_j, v_j, \alpha_j). \qquad (2)$$

For any edge $\mathbf{e}(i, j)$, the transition function τ_{ij} takes the parameter values of the vertex $\mathbf{v}_j$ in the patch P_{α_j} and expresses them with respect to the patch P_{α_i} containing vertex $\mathbf{v}_i$. In the most typical case, the edge (i, j) is inside a single patch, so τ_{ij} is the identity. The next most common case has $\mathbf{e}_{(i,j)}$ crossing exactly one patch boundary, so that $\tau_{ij} = \tau_{\alpha_i \alpha_j}$. Occassionally we encounter edges which cross multiple patches, as discussed in Section 3.1.2. In this case, the transition function is defined to be the composition of elementary transition functions, applied in the same order as the edge crosses the patches. The edge classification information collected during simplification is used to determine the appropriate transition function for each edge.

The system (2) is linear since we defined our transition functions to be linear. Our choice of transition functions also ensures another important property: if ψ is a solution of (2), then the result of the reassignment of any vertex $\mathbf{v}$ to another patch is again the same solution.

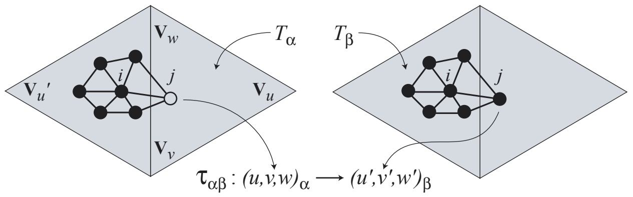


Figure 6: Transition functions are used to transform parameter values across patch boundaries.

We use a bi-conjugate gradient solver to compute a solution for the global parameterization system (2). This solution may have parameter values which are not in the range of valid barycentric coordinates. For every vertex with invalid parameter values, we check if the coordinates are valid when transformed to a neighboring patch, and if so, reassign the vertex to that patch.

Some of the vertices with invalid parameter values may not yet be reassigned. Such vertices occur in the neighborhoods of patch corners. In a subsequent stage of *vertex relaxation*, the patch corners are moved to new locations on the surface, forcing the global parameterization to be recomputed. Since the reassignment of patch corners is a non-linear operation, the theoretical analysis of the existence of a valid solution is a difficult problem. Experimentaly, we have found that after a couple iterations of relaxation and update of the global parameterization, all of the vertices are reassigned so that their parameter values are in their valid range.

3.2.2 Vertex Relaxation

When we compute our global parameterization, the patch corners are fixed while the parameter values of the remaining vertices are calculated. However, the positions of the fixed vertices are somewhat arbitrary, and may not be optimally placed to reduce the distortion in the parameterization. Morever, for some configurations the solution can be so distorted that the parameter values would be out of their valid range. To improve the parameterization, we reposition the patch corners using a vertex relaxation method [Guskov et al. 2000; Guskov et al. 2002].

The relaxation algorithm iterates over all the patch corners and repositions each one by assigning it to a new vertex in the original mesh. To relax a given patch corner, we begin by selecting a region of vertices on the original mesh, typically a few rings around the corner. We map the selected vertices onto a disc using a conformal map z^θ, where $\theta = 6/k$ and k is the number of patches that share the corner. Then we fix the positions of the outermost vertices and compute a local parameterization inside the disc. Using this parameterization, we locate the vertex closest to the origin, and assign it to be the new patch corner. Finally, the parameter values of the selected vertices are updated by applying the inverse conformal map to the local parameter values that we just computed. The patch assignments for these vertices are determined by splitting the disc into k sectors, corresponding to the neighborhood of the patch corner.

3.3 Algorithm Summary

We have described an algorithm for computing a globally smooth parameterization of an input mesh, which proceeds through the following stages:

1. The base complex is generated through simplification of the original mesh. During this process, every vertex that is removed from the base complex is assigned an initial parameter value with respect to a particular patch.
2. A bi-conjugate gradient solver is used to compute parameter values for all the vertices of the original mesh.
3. Vertex relaxation is performed on a small neighborhood of vertices around each patch corner.
4. Steps 2 and 3 are performed alternately until no more relaxation is required.

	# Vertices	# Faces	# Bnd.	# Base	E_r
Venus	50002	100000	0	15	3.55
David	583528	1174602	1	90	22.50
rabbit	67039	134074	0	70	3.68
bunny	34835	69473	4	150	4.22
horse	48485	96966	0	140	5.16
feline	49864	99732	0	280	3.54

Table 1: Statistics for the example meshes: number of vertices, faces, and boundary components in the original mesh; the number of vertices in the base complex; and the remeshing error (relative ℓ_2 error w.r.t. the bounding box diagonal, in units 10^{-5}).

4 Results

Our original motivation behind the research reported here started with the observation that remeshes of the same input model produced by different parameterization algorithms can yield widely varying rate distortion performance in a progressive geometry coder.

We implemented the algorithms described in Section 3 and applied them to a number of common models to allow comparison with previous results (where available). Figure 8 shows the models with patch layouts and resulting remeshes. For purposes of illustration we do not show the finest resolution remesh. The remeshes were produced by successive quadrisection of domain triangles followed by mapping through the parameterization to the surface and sampling it. Table 1 lists the remeshing errors (computed with Metro [Cignoni et al. 1998]) as mean-square magnitudes relative to the bounding box diagonal. Note that the high remeshing error for the David model is due to geometrically large "stalactite" clusters of noise in the interior of the model. Incidentally this model is a challenging stress test of the robustness of any parameterization algorithm.

One way to compare different parameterizations for a single model is demonstrated in Figure 1. The color coding visualizes the magnitude of discrete differences of the first derivatives of the parameterization over the surface. The original MAPS algorithm did not explicitly enforce smoothness (Figure 1a). It applied a modified version of Loop [1987] subdivision to the initial parameterization instead. The mesh in Figure 1c was produced by a solver which explicitly relaxes the parameterization on the interior of edge adjacent patches, holding the boundary of the pair of patches fixed. Here we can clearly see the "breaks" in smoothness across patch boundaries. Figure 1b was produced by the normal remesher of Guskov *et al.* [2000]. Interestingly it never explicitly computes a parameterization. Finally, Figure 1d shows the result achieved with our algorithm.

Another approach for comparing the quality of different parameterizations is to measure their rate distortion performance with the publicly available progressive geometry coder (PGC) of Khodakovsky and co-workers [2000]. Specifically, we compare favorably with their rate distortion results and with the Normal Mesh results from Khodakovsky and Guskov [2003]. Since we are interested in comparing smoothness of the parameterization and not the remeshing error, we do not present results at high bit rates which are dominated by the remeshing error.

The PGC paper mainly featured semi-regular meshes produced with the MAPS algorithm. We see that compression of our remeshes gives significant improvement (about $3 - 5$ dB on average). For Normal Mesh compression we have two sets of results.

One, based on the modified Butterfly [Zorin et al. 1996] wavelet transform, takes advantage of the normality of the mesh. It produces the coefficients with no parametric (tangential) components. Another set of results (Loop) does not recognize normality and leads to inferior performance. However, it is more suitable for measuring the smoothness of the parameterization because the Butterfly wavelet transform does not contain any parameterization information. For the three models for which we found normal compression data, our remeshes lead to better compression than the "Normal Loop". It is not surprising, that we cannot match the "Normal Butterfly" curves since we still need to deal with all three components of the coefficients.

The timing results were measured on an Athlon 1800++. The total time to remesh the David head with 586K vertices is 32 min. Most of this time was taken by two passes of the global solver algorithm with the tolerance for the residual vector set to $5 \cdot 10^{-5}$ in the ℓ_∞ norm. In comparison, the same sequence of steps applied to the simplifed David head of 90K vertices took only 86 seconds.

Throughout our experiments, we relied on the default simplification parameters from Section 3.1.1. Some trial-and-error is needed to select the base domain size, with the help of a simplification cost plot like Figure 5d. Intervention during the parameterization step is minimal, as the user only decides how many iterations of relaxation to perform to lower the distortion to a satisfactory level.

Since we are interested in compressing highly detailed models, our input data consists of densely sampled meshes (Figure 2). For this reason, our decision to restrict patch corners to vertices of the original mesh does not have any noticeable effect on the quality of the parameterizations we produce. For sparsely sampled input data, we locally refine the input mesh to introduce more degrees of freedom for our relaxation algorithm.

5 Summary and Future Work

In this paper we introduced a novel method to compute *globally* smooth parameterizations for arbitrary topology surfaces. Distortion in the parameterization is controlled both through judicious choice of the base domain and relaxation of the intitial parameterization. The solver uses explicit transition functions between patches to ensure global smoothness. The base domain creation is controlled by *triangle quality* and *metric distortion*. These cost heuristics have proven effective in helping to avoid high distortion in the parameterization, and the formation of badly shaped patches. In comparing the rate distortion properties of these meshes against the best previous examples, we found our approach to have comparable and in some cases superior performance.

An interesting avenue for future work is the exploration of global smoothness questions from an analytic point of view. For example, we know from subdivision surface techniques that globally smooth parameterizations are possible even in the presence of irregular vertices. Consider lines of constant parameter values, which define a tangent field on the surface. Our choice of transition functions ensures that lines crossing patch boundaries have continuously varying tangent vectors. Such a tangent field is defined everywhere except at patch corners. If the valence of a corner is six our relaxation procedure produces a smooth field at such points. At irregular patch corners the pattern is visually very close to what is

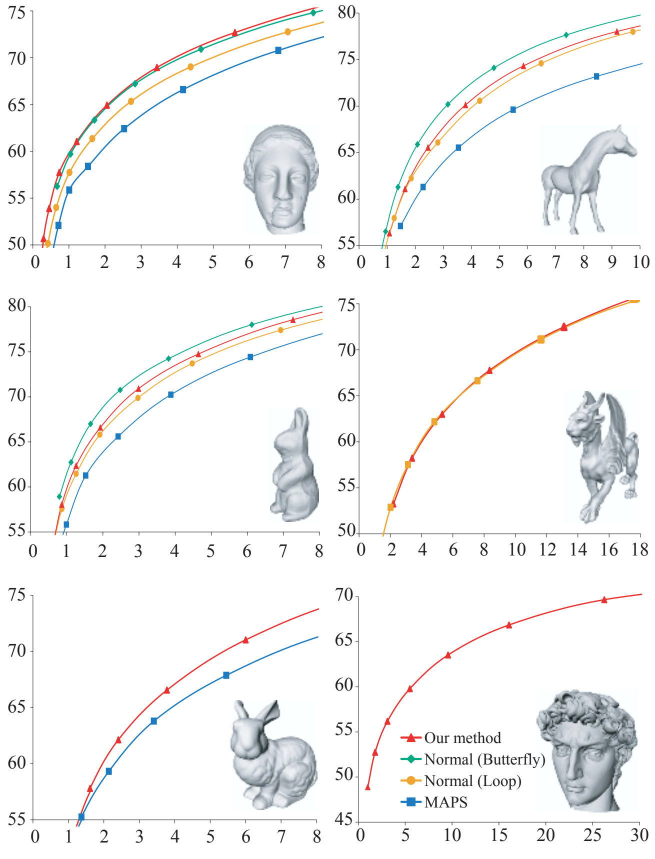


Figure 7: Rate distortion curves provide a measurement of the quality of a surface parameterization. Shown here are our results compared to parameterizations generated by other remeshing techniques (PSNR vs. compressed file size in 10^3 bytes).

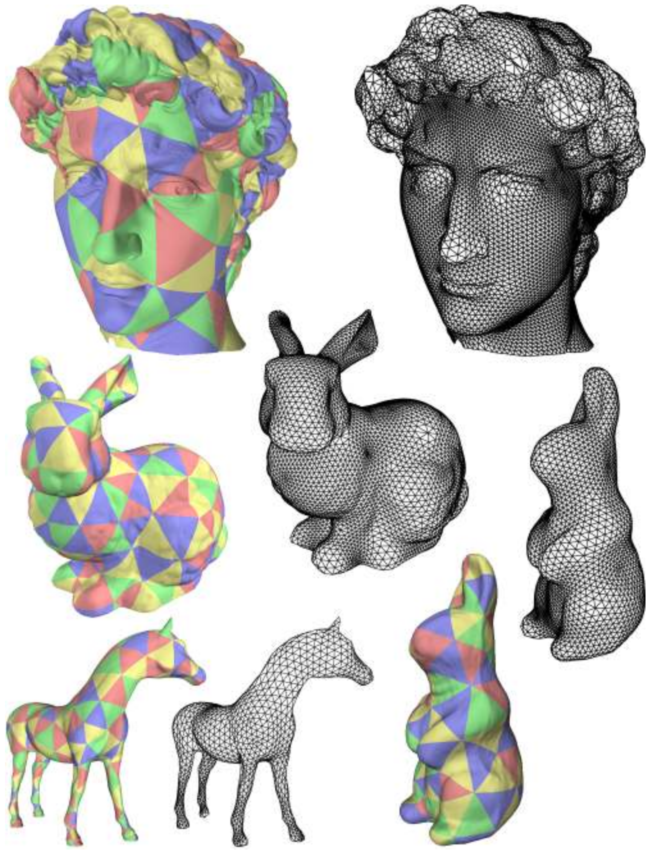


Figure 8: Examples of parameterizations created by our algorithm. A base complex (left) is generated to reduce the distortion in the globally smooth parameterization of the surface (right).

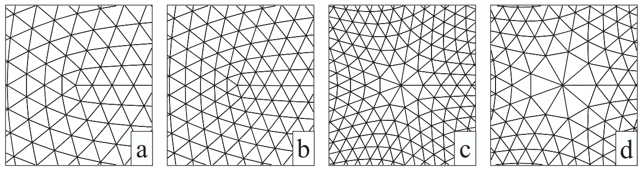


Figure 9: The parameterization produced by Loop subdivision (a, c) is remarkably similar to our own (b, d). Shown here are discs with 5 and 9 sectors, respectively.

seen under smooth subdivision. Figure 9 compares parameterizations of k-sector disk meshes ($k = 5$ and $k = 9$) computed by our method with the parameterization produced through Loop subdivision. The close similarity gives us hope that our parameterization can be proven smooth even at irregular vertices.

It would also be interesting to compare our approach to that of Gu and Yau [2003] who produce globally smooth parameterizations with the least number of irregular vertices. To alleviate extensive distortion they add additional punctures to the surface at extrema of the conformal factor. We hypothesize that placement of these punctures plays a role similar to placement of irregular vertices in our approach. In general, finding methods that allow more control over the number and placement of high and low valence vertices would be very useful in the control of distortion.

Finally, more work is needed to construct efficient numerical solvers for very large unstructured meshes.

Acknowledgments This work was supported in part by NSF (DMS-0220905, DMS-0138458, ACI-0219979), the DOE (W-7405-ENG-48/B341492), nVidia, the Center for Integrated Multiscale Modeling and Simulation, Intel, Alias|Wavefront, Pixar, Microsoft, and the Packard Foundation. Special thanks to Ilja Friedel, Eitan Grinspun, Kai Hormann, Mark Meyer, Steven Schkolne, Zoë Wood, and to our reviewers for their helpful suggestions.

References

BIERMANN, H., MARTIN, I., BERNARDINI, F., AND ZORIN, D. 2002. Cut-and-Paste Editing of Multiresolution Surfaces. *ACM Transactions on Graphics 21*, 3, 312–321.

CIGNONI, P., ROCCHINI, C., AND SCOPIGNO, R. 1998. Metro: Measuring Error on Simplified Surfaces. *Computer Graphics Forum 17*, 2, 167–174.

DESBRUN, M., MEYER, M., AND ALLIEZ, P. 2002. Intrinsic Parameterizations of Surface Meshes. *Computer Graphics Forum 21*, 3, 209–218.

ECK, M., DEROSE, T. D., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., AND STUETZLE, W. 1995. Multiresolution Analysis of Arbitrary Meshes. In *Proceedings of SIGGRAPH*, 173–182.

FLOATER, M. S. 1997. Parameterization and Smooth Approximation of Surface Triangulations. *Computer Aided Geometric Design 14*, 3, 231–250.

FLOATER, M. S. 2003. Mean Value Coordinates. *Computer Aided Geometric Design 20*, 1, 19–27.

GARLAND, M., AND HECKBERT, P. S. 1997. Surface Simplification Using Quadric Error Metrics. In *Proceedings of SIGGRAPH*, 209–216.

GOTSMAN, C., GU, X., AND SHEFFER, A. 2003. Fundamentals of Spherical Parameterization for 3D Meshes. In *Proceedings of SIGGRAPH*.

GRIMM, C. M., AND HUGHES, J. F. 1995. Modeling Surfaces of Arbitrary Topology using Manifolds. In *Proceedings of SIGGRAPH*, 359–368.

GU, X., AND YAU, S.-T. 2003. Global Conformal Surface Parameterization. Submitted for publication, April.

GU, X., GORTLER, S. J., AND HOPPE, H. 2002. Geometry Images. *ACM Transactions on Graphics 21*, 3, 355–361.

GUSKOV, I., VIDIMČE, K., SWELDENS, W., AND SCHRÖDER, P. 2000. Normal Meshes. In *Proceedings of SIGGRAPH*, 95–102.

GUSKOV, I., KHODAKOVSKY, A., SCHRÖDER, P., AND SWELDENS, W. 2002. Hybrid Meshes: Multiresolution using Regular and Irregular Refinement. In *Proceedings of the Eighteenth Annual Symposium on Computational Geometry*, 264–272.

HOPPE, H. 1996. Progressive Meshes. In *Proceedings of SIGGRAPH*, 99–108.

HORMANN, K., AND GREINER, G. 2000. MIPS: An Efficient Global Parametrization Method. In *Curve and Surface Design: Saint-Malo 1999*. Vanderbilt University Press, 153–162.

KHODAKOVSKY, A., AND GUSKOV, I. 2003. Compression of Normal Meshes. In *Geometric Modeling for Scientific Visualization*. Springer-Verlag.

KHODAKOVSKY, A., SCHRÖDER, P., AND SWELDENS, W. 2000. Progressive Geometry Compression. In *Proceedings of SIGGRAPH*, 271–278.

KOBBELT, L., CAMPAGNA, S., AND SEIDEL, H.-P. 1998. A General Framework for Mesh Decimation. In *Graphics Interface '98*, 43–50.

LEE, A. W. F., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. 1998. MAPS: Multiresolution Adaptive Parameterization of Surfaces. In *Proceedings of SIGGRAPH*, 95–104.

LEE, A., DOBKIN, D., SWELDENS, W., AND SCHRÖDER, P. 1999. Multiresolution Mesh Morphing. In *Proceedings of SIGGRAPH*, 343–350.

LEE, A., MORETON, H., AND HOPPE, H. 2000. Displaced Subdivision Surfaces. In *Proceedings of SIGGRAPH*, 85–94.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least Squares Conformal Maps for Automatic Texture Atlas Generation. *ACM Transactions on Graphics 21*, 3, 362–371.

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

MAILLOT, J., YAHIA, H., AND VERROUST, A. 1993. Interactive Texture Mapping. In *Proceedings of SIGGRAPH*, 27–34.

PINKALL, U., AND POLTHIER, K. 1993. Computing Discrete Minimal Surfaces. *Experimental Mathematics 2*, 1, 15–36.

PRAUN, E., AND HOPPE, H. 2003. Spherical Parameterization and Remeshing. In *Proceedings of SIGGRAPH*.

SANDER, P. V., SNYDER, J., GORTLER, S. J., AND HOPPE, H. 2001. Texture Mapping Progressive Meshes. In *Proceedings of SIGGRAPH*, 409–416.

SANDER, P., GORTLER, S., SNYDER, J., AND HOPPE, H. 2002. Signal-Specialized Parameterization. In *Eurographics Workshop on Rendering*, 87–100.

SHEFFER, A., AND HART, J. C. 2002. Seamster: Inconspicuous Low-Distortion Texture Seam Layout. In *IEEE Visualization*, 291–298.

SHEWCHUK, J. R. 2002. What Is a Good Linear Element? Interpolation, Conditioning, and Quality Measures. *Eleventh International Meshing Roundtable*.

SORKINE, O., COHEN-OR, D., GOLDENTHAL, R., AND LISCHINSKI, D. 2002. Bounded-distortion Piecewise Mesh Parameterization. In *IEEE Visualization*, 355–362.

STRANG, G., AND NGUYEN, T. 1996. *Wavelets and Filter Banks*. Wellesley-Cambridge Press.

WOOD, Z. J., DESBRUN, M., SCHRÖDER, P., AND BREEN, D. 2000. Semi-Regular Mesh Extraction from Volumes. In *IEEE Visualization*, 275–282.

ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1996. Interpolating Subdivision for Meshes with Arbitrary Topology. In *Proceedings of SIGGRAPH 96*, 189–192.