Volumetric Basis Reduction for Global Seamless Parameterization of Meshes

Julian Panetta
New York University

Michael Kazhdan
John Hopkins University

Denis Zorin
New York University

Abstract

We present an efficient method for generating global parameterizations of large meshes. Following the state-of-the art work on global parameterization, we use a cross-field guided approach: we fit the parameterization to a field aligned with surface geometry while constraining the mapping to be seamless, in the sense of local continuity of parametric lines. We extend these approaches by showing that the constraints on the mapping can be decoupled from the mesh tessellation, allowing us to formulate the problem of seamless mesh parameterization over a general function basis. In particular, we show that by adapting a recently proposed volumetric basis, we can develop a highly efficient solver for mesh parameterization.

Using the solver, we compute parameterizations for meshes consisting of up to 20 million vertices in 1600 seconds, providing a solution that outperforms state-of-the-art direct solvers in both time and memory.

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

Keywords: geometric modeling, parameterization

Figure 1: *Parameterization of a large (7M triangle) mesh computed using our solver*

1 Introduction

Computing global seamless parameterizations is useful for a variety of geometry processing tasks. In particular, it is an essential component of a class of quadrangulation algorithms with many attractive features (feature alignment, automatic singularity placement, smoothness, and element shape control). Existing versions of global parameterization methods use piecewise-linear discretizations tied to mesh connectivity in combination with direct linear systems solvers. While highly efficient and accurate for problems of moderate size, direct solvers are limited in their applicability to larger problems by their high memory usage, by the relatively high time complexity of their factorization stage ($O(n^{3/2})$ for these types of systems), and by how difficult they are to parallelize.

In this paper, we present a global parameterization technique based on the ideas developed in [Chuang et al. 2009] to enable handling of large meshes. The key idea of [Chuang et al. 2009] is to use restrictions of volumetric basis functions to the surface to define a basis. The key advantage of this approach is that *the basis functions used by the solver are decoupled from the mesh connectivity and resolution.* In particular, this makes it possible to compute approximations to the parameterization of a large mesh at a chosen scale.

This is of particular importance in the context of mesh parameterization, which is most often guided by a smooth field, so the full resolution of the mesh is not required in order to accurately compute the mapping.

While demonstrating good scalability, accuracy, and mesh-independent behavior, the method of [Chuang et al. 2009] has a number of limitations: it was developed to solve a scalar Poisson equation on the surface as opposed to the more complex constrained versions of Poisson equations that need to be solved for global parameterization: in contrast to the scalar Poisson equation, which easily admits a standard Galerkin discretization in any basis, the global parameterization problems are formulated in [Kälberer et al. 2007; Bommes et al. 2009] directly in terms of piecewise-linear and piecewise-constant functions on the mesh.

Our work addresses these difficulties. The main components of our solver include:

- a generalized formulation of the global seamless parameterization problem, not tied to the piecewise-linear basis.
- use of the piecewise-linear basis and a direct solver for post-processing the subset of the mesh on which we expect the volumetric solution to be inaccurate.

For sufficiently large meshes, we obtain better time and space efficiency: for the largest meshes, compared to a direct solution of the system in the mesh's piecewise-linear basis, we are roughly five times more efficient in space and ten in speed, making it possible to parameterize a mesh containing 20 million triangles in 1600 seconds.

2 Related work

Parameterization was studied from many different points of view (see [Hormann et al. 2007; Sheffer et al. 2006] for surveys). We briefly consider the most closely related work.

Our method is based on [Kälberer et al. 2007] and [Bommes et al. 2009]: we find a parameterization of a given shape by fitting its gradient to a given smooth *cross-field* capturing surface features (for example, to smoothed principal curvature directions or a field constructed from features). [Ray et al. 2006] is a similar nonlinear feature-aligned method. A recent technique of [Zhang et al. 2010] uses a wave-based approach, which combines some of the features of spectral quadrangulation for feature alignment and provides anisotropy control. In all previous work, parameterization

techniques are applied to small to medium size meshes. Most recently, [Pietroni et al. 2011] extends field-based techniques to parameterization of range image sets, which can be used as intermediate representations for parametrizing large models. We note that while this method can construct coarse quadrangulations of large-scale models easily, it does not produce in general a parameterization of the original models.

Other methods use global harmonic or conformal parameterizations with singularities [Gu and Yau 2003; Dong et al. 2006; Tong et al. 2006; Ben-Chen et al. 2008; Springborn et al. 2008; Kovacs et al. 2009].

A number of algorithms [Eck et al. 1995; Lee et al. 1998; Khodakovsky et al. 2003; Marinov and Kobbelt 2005; Daniels et al. 2009a; Daniels et al. 2009b; Pietroni et al. 2009; Tarini et al. 2010] use simplification techniques for constructing a conforming domain mesh. These techniques make it possible to obtain very coarse domains with good user control over the domain size. While some degree of feature alignment is possible (cf. [Lee et al. 1998], [Marinov and Kobbelt 2005]), it is limited by the difficulty of preserving features in simplification. These techniques are fundamentally different from the PDE-based techniques we are considering, and comparisons of potential scalability and quality are difficult.

Fitting the gradients requires solving a sparse linear system, and a variety of methods have been proposed for addressing this challenge. These have included both direct solvers [Davis and Hager 2005; Schenk et al. 2001] and multigrid solvers. The multigrid solvers have ranged from "black-box" algebraic multigrid solvers that rely solely on the algebraic information encoded in the system matrices [Ruge and Stueben 1987] to more geometry-driven solvers that support multigrid through the design of mesh hierarchies [Kobbelt et al. 1998; Clarenz et al. 2000; Schneider and Kobbelt 2001; Ray and Levy 2003; Aksoylu et al. 2005; Shi et al. 2006].

3 Global parameterization in general bases

3.1 Global parameterization

We start with a brief review of global parameterization. In general, surfaces must be cut for a mapping to the plane to be possible. It is common to cut a surface M to a topological disk, M_c, along a 1D *seam* and construct a map $q : M_c \to \mathbb{C}$, $q = (u,v) = u + \mathbf{i}v$ (we use complex representation of parameterization coordinates with $\mathbf{i} = \sqrt{-1}$). Each point of the seam is split into multiple boundary points of the cut mesh and has multiple parametric positions. The seam consists of a set of *seam curves*, Σ, with each seam curve $\sigma \in \Sigma$ consisting of points split into exactly two points in M_c. These curves terminate at *seam nodes*, which can split into more than two points in M_c or can remain unsplit.

Parameterization defines a flat metric on the surface; conversely, if a flat metric is defined and the surface is cut to a disk, a parameterization is uniquely determined up to rigid transforms. Due to the Gauss-Bonnet theorem, an everywhere flat metric may not exist: the integral of the curvature should match the Euler characteristic. Forcing the metric to be flat everywhere may also lead to high distortion. For these reasons, a number of *singularities* (*cones*) with nonzero curvature need to be introduced.

The seam needs to pass through singularities as no local continuous parameterization can be defined in their neighborhoods. At any other point, a consistent local parameterization can be constructed from the metric, which has to coincide with the original one up to rigid transforms of the parts of the cut mesh M_c separated by seam curves.

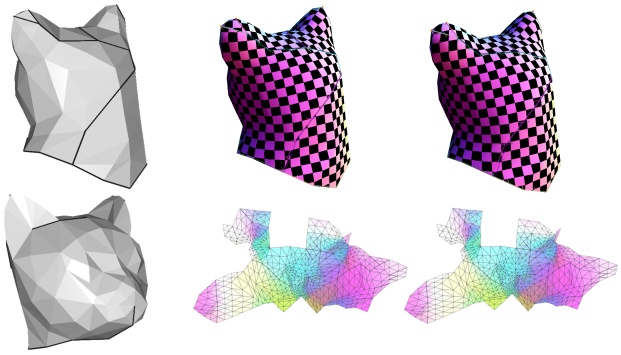

Figure 2: *A cut cat model (left), parameterized to a 2D domain (bottom). Without enforcing integer constraints on the translational coefficients, the parameterization results in discontinuities across the seam (center). Fixing the translational coefficients at integer positions guarantees a continuous parameterization (right).*

We are interested in *seamless* parameterizations, which satisfy additional constraints at the seam: the change of the coordinates across the seam is a restricted rigid transform, X. The rotational part of this transform is restricted to the symmetries of the coordinate grid ($k\pi/2$ rotations) so that the parametric lines in the u and v directions continue across the seam smoothly. More precisely, we require the following constraint to hold for the two parametric positions q and q' associated with a point on the seam:

$$q' = Xq = rq + t, \tag{1}$$

where r is a rotation by $k\pi/2$ (in complex form, ± 1 or $\pm\mathbf{i}$), and t is a translation. For many applications, it is also important that all t are of the form hj, where j is a point on the integer lattice, and h is a global scale factor (Figure 2).

The parameterization is typically constructed by minimizing an energy that is invariant with respect to these rigid transformations. We use the energy of [Kälberer et al. 2007; Bommes et al. 2009]; this energy attempts to align the gradients of the parameterization with a pair of orthogonal unit vectors $(\mathbf{u}_T, \mathbf{v}_T)$, and can be written in the complex form as

$$E_T = A_T \|\nabla q - w_T\|^2, \tag{2}$$

where $w_T = \mathbf{u}_T + \mathbf{i}\mathbf{v}_T$. The unordered quadruple of vectors $(\mathbf{u}_T, \mathbf{v}_T, -\mathbf{u}_T, -\mathbf{v}_T)$ forms a continuous *cross-field* on the surface, except at a number of isolated singularities, which are also parameterization singularities.

The *rotations* r in the constraint (1) are inferred from the cross-field as discussed in [Bommes et al. 2009]; the *translations* remain free variables in the optimization, but they are constrained to be integer. We adopt the convention that all seam curves are oriented, and the rotation and translation associated with the curve is the one for which q' corresponds to the domain on the *right* of the curve, when looking towards its tip.

3.2 Parameterization with general basis functions

We compute our parameterizations using a basis derived from a general basis $\{B_i\}$ $(i = 1 \ldots N)$ on the mesh M. We do not make any assumptions on the basis functions B_i, other than that

- the basis functions are continuous;
- they form a partition of unity; and
- we can compute gradients of these functions on each triangle.

Hat basis functions on the mesh are a special case of our construction, but our primary target is the basis functions obtained by restricting a B-spline basis defined on a regular lattice to the mesh, as described in the next section. These functions can have a large and complex support. In particular, the support can contain multiple components or have nontrivial topology.

The seam conditions (1) result in constraints on the basis functions' coefficients. In the solver used in [Bommes et al. 2009], constraints on the coefficients of the hat basis functions (that is, on parametric positions of mesh vertices) are constructed explicitly and then eliminated using Gaussian elimination on the constraint matrix. This approach, while simple and flexible, does not scale well and is unsuitable for multigrid formulations.

Instead, we use geometric considerations to obtain a *reduced basis* for which constraints are satisfied automatically. This approach resembles the formulation of [Kälberer et al. 2007] for the basis of closed 1-forms (but we do not use a covering surface).

Constraints on the basis function coefficients. Suppose the support $\operatorname{supp} B_i$ is split by the seam into disconnected components D_{ij}. Let B_{ij} be the basis functions obtained from B_i by splitting it up into individual basis functions each supported on D_{ij}. The parameterization $F(p)$ of M_c is defined as

$$F(p) = \sum_{i,j} c_{ij} B_{ij}(p), \tag{3}$$

where c_{ij} are complex coefficients. Each coefficient is associated with a domain D_{ij}.

Seam curve σ splits the incident domain components into two (not necessarily disjoint) sets: regions to the left and regions to the right. Indexing the regions on the left as D_{ij}, we denote the corresponding components across σ by $D_{ij'}$. As B_i is continuous, $B_{ij'}(p) = B_{ij}(p)$ for a seam point p. Then, the constraint (1) can be written as:

$$\begin{aligned} 0 &= \sum_{i,j} c_{ij'} B_{ij'}(p) - r_\sigma \sum_{i,j} c_{ij} B_{ij}(p) - t_\sigma \\ &= \sum_{i,j} (c_{ij'} - r_\sigma c_{ij} - t_\sigma) B_{ij}(p), \end{aligned}$$

where we have used the partition of unity property of the basis. In general, this equation can be satisfied for all p along the seam curve only if

$$c_{ij'} = r_\sigma c_{ij} + t_\sigma \tag{4}$$

for any B_i with $\operatorname{supp} B_i$ overlapping the seam curve σ.

Restricted basis example. To explain our formulas for the basis satisfying (4), we consider an example basis function B_i with support overlapping a single seam σ. The seam splits the support into two parts: D_{i0} and D_{i1}. Since $c_{i1} = r_\sigma c_{i0} + t_\sigma$, we can rewrite the linear combination $c_{i0}B_{i0} + c_{i1}B_{i1}$ as $c_{i0}(B_{i0} + r_\sigma B_{i1}) + t_\sigma B_{i1}$. The dependent coefficient c_{i1} is eliminated, and two new basis functions appear: $B_{i0} + r_\sigma B_{i1}$ and B_{i1}, the latter corresponding to the translation variable t_σ. If the whole surface has a single seam (e.g. a disk with a hole), the complete new basis consists of $\tilde{B}_i = B_{i0} + r_\sigma B_{i1}$ for all B_i overlapping σ, $\tilde{B}_i = B_i$ for the rest, and a single additional basis function $\hat{B}^\sigma = \sum_i B_{i1}$; then $F(p)$ has the form $\sum_i c_{i0}\tilde{B}_i + t_\sigma \hat{B}^\sigma$, and it is easy to check that it satisfies the seam constraint for any choice of variables c_{i0} and t_σ.

More generally, a *restricted basis* consists of (a) for each connected component of $\operatorname{supp} B_i$ not overlapping a singularity with noninteger index, a single basis function $\tilde{B}_i$ that is a linear combination of B_{ij}; and (b) a set of basis functions $\hat{B}^\sigma$ for *independent* translation variables t_σ, which are obtained by Gaussian elimination on a small constraint system of complexity proportional to the number of seam curves.

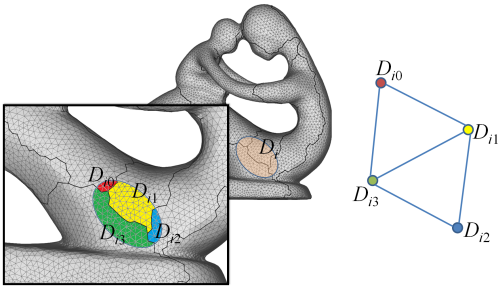

Figure 3: *Example of a local domain graph: A function B_i has support D_i on the mesh. This support is the union of four disconnected regions $D_{i0}, \ldots, D_{i3}$ on the cut mesh. The nodes in the local domain graph (right) correspond to the disconnected regions and edges connect nodes whose associated regions meet along a seam.*

We call the first type of basis functions *positional*, as in the simplest case of the hat function basis, these define the parametric positions of vertices. We call the second type *translational* basis functions.

General restricted basis and constraints. Next, we briefly explain how these two types of basis functions are constructed using constraints (4). A more detailed derivation is found in the supplementary document.

As in the simple example above, we initially construct a single new positional basis function $\tilde{B}_i$ for each B_i of the original basis and a translational basis function $\hat{B}^\sigma$ for each seam curve. This translational basis function will accumulate a contribution from each original basis function B_i whose support overlaps σ. These new basis functions, however, are not generally independent: each B_i with support overlapping one or more seams may contribute a constraint, either expressing a positional variable in terms of translations or constraining the translational variables.

In the following, we consider one original basis function B_i at a time, so we drop the index i. For simplicity, assume that its support is connected (disconnected parts of support are effectively treated as separate basis functions).

Local domain graph. The *local domain graph* $\mathcal{G}_D$ of $\operatorname{supp} B_i$ is a directed graph with nodes representing the domains D_j and edges representing shared seam curves σ (Figure 3). Typically this graph either has one node or a single pair of edges connecting two nodes, but the structure is more complicated for basis functions overlapping multiple seam curves and singularities. To be able to handle general volumetric basis functions, we consider the problem in full generality.

Constraints $c_{j'} = r_\sigma c_j + t_\sigma$ are associated with the edges in the graph connecting D_j and $D_{j'}$ and will accumulate for paths.

For an edge path $\pi = (e_1, \ldots, e_k)$ in $\mathcal{G}_D$, we define $R(\pi)$ to be the product of rotations $r_{\ell(e_k)} r_{\ell(e_{k-1})} \ldots r_{\ell(e_1)}$, where $\ell(e)$ is the seam corresponding to edge e. For an empty path π, $R(\pi) = 1$.

Consider, for example, a path π with 3 edges, connecting domains $D_0, \ldots, D_3$, and crossing seam curves $\sigma_1, \sigma_2, \sigma_3$. Composing constraints along this path, we get $c_3 = r_3(r_2(r_1 c_0 + t_1) + t_2) + t_3 = r_3 r_2 r_1 c_0 + r_3 r_2 t_1 + r_3 t_2 + t_3$. Therefore, the term $c_3 B_3$ contributes a term $r_3 r_2 B_3$ to the translational basis function $\hat{B}^{\sigma_1}$, $r_3 B_3$ to $\hat{B}^{\sigma_2}$, and B_3 to $\hat{B}^{\sigma_3}$.

More generally, for a path π connecting D_i and D_j with edges $e_1 \dots e_k$, we have a constraint of the form

$$c_j = R(\pi)c_i + \sum_{\sigma\in\Sigma} \frac{T^\sigma(\pi)t_\sigma}{2},$$

where $T^\sigma(\pi)$ represents the coefficients accumulated onto t_σ during traversal of π and is given by the formula

$$T^\sigma(\pi) = \sum_{e\in\pi,\ell(e)=\sigma} R(\pi_e^+) - \sum_{e\in\pi,\ell(e^{-1})=\sigma} R(\pi_e^+)r_\sigma^{-1}. \quad (5)$$

Here, π_e^+ is the part of π following e. Every time an edge in the path crosses σ, it contributes to the constraint transform a term of either t_σ or $t_{\sigma^{-1}} = -r_\sigma^{-1}t_\sigma$ (depending on the crossing direction) onto which the subsequent seam crossings compose a rotation. For an empty path π, $T^\sigma(\pi) = 0$.

The division by two in the constraint's translation term is needed because Σ contains both σ and its oppositely oriented pair σ^{-1}, causing two copies of the same translation term to accumulate for each seam crossing.

Proposition 1. *The definitions of $R(\pi)$ and $T^\sigma(\pi)$ depend only on the homotopy class of the path π: if two paths can be deformed to each other, they have equal $R(\pi)$ and $T^\sigma(\pi)$.*

Proposition 2. *Fix a domain D_0 (reference domain) in the support of B, and let π_j be a path connecting it to D_j. Constraints* (4) *imply that the basis function $\tilde{B}$ and the term B contributes to $\hat{B}^\sigma$ for each seam σ are given by*

$$\begin{aligned} \tilde{B} &= \sum_j R(\pi_j)B_j \\ \hat{B}^\sigma_{loc} &= \sum_j T^\sigma(\pi_j)B_j/2. \end{aligned} \quad (6)$$

These propositions are proved in the supplementary material.

Constraints for local graphs with nontrivial homology. If the local graph is a tree, there is a unique path from the reference domain to each D_j, and there is only one way to define c_j from c_0. On the other hand, if the graph has a nontrivial loop, there may be multiple paths π_j from D_0 to D_j yielding different expressions for c_j that all must agree. This leads to constraints on the coefficients.

Due to the homotopy invariance of the expressions (Proposition 1), it is sufficient to find a homology basis in the graph to define all independent constraints. Choosing a homotopy basis with base point D_0, we obtain for each loop in the basis, π:

$$c_0 = R(\pi)c_0 + \sum_{\sigma\in\Sigma} \frac{T^\sigma(\pi)}{2}t_\sigma. \quad (7)$$

There are two possibilities. When $R(\pi) = 1$ we get a constraint on translations only,

$$\sum_{\sigma\in\Sigma} \frac{T^\sigma(\pi)}{2}t_\sigma = 0, \quad (8)$$

as c_0 does not enter the equation. When $R(\pi) \neq 1$, the constraint yields an expression for c_0 (and consequently for all other c_j) in terms of translations:

$$c_0 = \frac{1}{1-R(\pi)} \sum_{\sigma\in\Sigma} \frac{T^\sigma(\pi)}{2}t_\sigma. \quad (9)$$

The latter constraints simply eliminate some functions $\tilde{B}$ from the basis. The former are assembled into a small system, which is solved for an independent set of translations. Introducing the index sets I^t_{ind} and I^t_{dep} for independent and dependent translations (and $I^t_{all} = I^t_{ind} \cup I^t_{dep}$), we can compute weights W expressing all translations in terms of the independent ones:

$$t_m = \sum_{n\in I^t_{ind}} w_{mn}t_n \quad \forall\, m \in I^t_{all}.$$

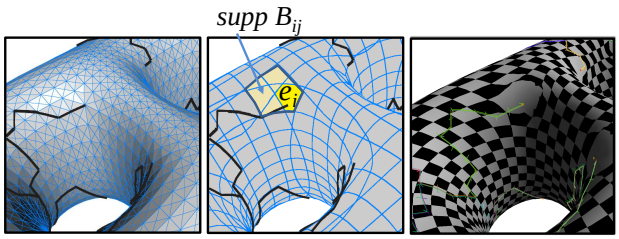

Figure 4: *Triangle collapse: When the tessellation of the mesh (left) is much finer than the resolution of the finite-elements (center), triangles in the vicinity of non-integer cones collapse to a single point in the parameterization domain. In texture mapping applications, this causes severe distortion near these cones (right).*

Local constancy of parameterization. It is important to observe that every basis function whose domain graph includes the cycle in (9) has an identical positional constraint (the expression for c_{i0} does not depend on i). In common cases (e.g. a single singularity in the common overlap area Ω, which is separated by seams into domains D_j) it follows that all basis functions overlapping Ω have the same coefficients. Combining this with the partition of unity property, we see that the parameterization is *constant on each domain D_j*. In other words, it collapses these domains to single points (Figure 4). We note that this does not happen if only a *single* basis function overlaps any singularity, which is the case for the hat basis.

This is an important limitation of parameterizations obtained using a general basis; we discuss its implications in Section 5.

Complete basis description. The basis satisfying all constraints by construction is defined in the following proposition, which follows from the formulas above. We must now reintroduce the indices from our original basis $\{B_i\}$, which means that the quantities defined in (6) for each B_i are renamed $\tilde{B}_i$ and $\hat{B}^\sigma_{loc_i}$.

Proposition 3. *Let I^p_{ind} be the index set for basis functions B_i whose local graphs contain only homology loops π with $R(\pi) = 1$, i.e. whose positional coefficients have no constraint like* (9)*. Let I^p_{dep} be the remaining basis function indices. Then the reduced basis consists of (a) positional functions $\tilde{B}_i$ defined by* (6) *for each $i \in I^p_{ind}$; and (b) translational basis functions $\hat{B}^n$ defined by*

$$\hat{B}^n = \sum_{m\in I^t_{all}} w_{mn} \left[\sum_{i=1}^{N} \hat{B}^{\sigma_m}_{loc_i} + \sum_{d\in I^p_{dep}} \frac{1}{1-R(\pi_d)} \frac{T^{\sigma_m}(\pi_d)}{2} \tilde{B}_d \right]$$

for each $n \in I^t_{ind}$. Here π_d is a noncontractible closed path in $\mathcal{G}_D(B_d)$ starting and ending at D_{d0} and having $R(\pi_d) \neq 1$. Notice that this formula implicitly uses the fact that each translation variable t_m originated from some seam σ_m.

The formulas are relatively complex, as few assumptions can be made about the structure of the local graphs for the basis functions.

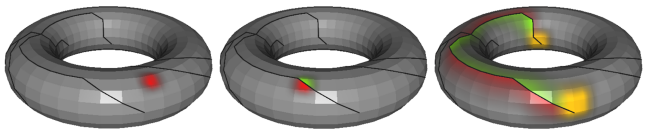

Figure 5: *Reduced basis functions: A positional basis function in the octree basis not overlapping a seam (left). A positional function overlapping a seam (center). A translational basis function (right). Complex values are visualized using HSV, with hue corresponding to the angle and value corresponding to norm. Thus, the first function is continuous over the mesh, while the second and third exhibit a rotation as they cross the seam.*

They can be greatly simplified if, e.g., the support of a basis function contains only a single node of the seam, is simply connected, and crosses each seam exactly once.

These conditions hold for the hat basis, which can be reduced to a basis similar to [Kälberer et al. 2007]. For non-seam vertex basis functions, the local graphs are trivial. For a seam node, the local domain graph is a single loop, with vertices corresponding to seam edge-separated sectors in the node's one-ring of triangles.

For this reason, there is only one constraint associated with each seam node: there is only one non-contractible path π in the loop. If the node is nonsingular or has integer index, then $R(\pi) = 1$ and a constraint on translations is added, as in (8).

Otherwise, the position of the vertex is determined by translations as in (9). Each translational basis function ends up following a set of seam curves and differences to 1 across these seams.

Discretizing the energy using the new basis. We use a standard Galerkin formulation for the system: to minimize $\|\nabla q - w\|^2$ over a basis ϕ_i, we compute the system matrix L and right-hand-side b as

$$L_{ij} = \int_{\operatorname{supp}\phi_i \cap \phi_j} \langle \nabla_M \phi_i, \nabla_M \phi_j \rangle dA,$$
$$b_i = \int_{\operatorname{supp}\phi_i} \langle \nabla_M \phi_i, w \rangle dA,$$

using our restricted basis functions (both positional and translational) for ϕ_i.

4 Volumetric basis functions

We begin with a brief review of the volumetric system described in [Chuang et al. 2009] and then describe how the method can be adapted to support mesh parameterization.

4.1 The basis

To define a basis, [Chuang et al. 2009] embed the mesh M in a regular $2^D \times 2^D \times 2^D$ voxel grid. Then, first-order B-splines are associated with the corners of the grid cells, and test functions $\{B_1(p), \ldots, B_n(p)\} : M \to \mathbb{R}$ are defined by restricting the B-splines to the surface of the mesh. Since only those B-splines whose support overlaps the geometry need to be considered in defining the linear system, the coefficients of the system can be indexed by the corners of all depth-D nodes of an octree that is adapted to the surface. For first-order B-splines, this means that only corners of cells overlapping the surface need to be considered.

4.2 Defining the linear system

To use the volumetric basis in the task of mesh parameterization, we need to construct the associated linear system.

To simplify the integrations, we partition the cut mesh M_c into disjoint elements e_n whose intersections with regions D_{ij} are trivial (i.e. either the intersection is equal to e_n, or it is empty). In the context of the hat function basis, these are simply the triangles of the mesh, and each element is in the support of three basis functions (corresponding to the hat functions centered at the three vertices of the triangle). In the context of B-splines, the elements correspond to the connected components of the intersection of M_c with voxels, and an element is in the support of eight basis functions (one for each of the eight corners of the voxel containing the element). Figure 4 (center) shows a visualization of the elements for the octree basis.

Using these, we define *element functions* ϕ_{ijn}, obtained by restricting the function B_{ij} to e_n. Since the intersection of the support of each B_{ij} with e_n is trivial by construction, we can express the functions B_{ij} as sums of the ϕ_{ijn}. Furthermore, since the reduced basis functions $\hat{B}^m$ and $\tilde{B}_i$ are themselves expressed as the linear combinations of the B_{ij}, we can define linear transformations $\hat{P}$ and $\tilde{P}$ that take a vector corresponding to the coefficients of a function with respect to the reduced basis and return the coefficients with respect to the element functions ϕ_{ijn}.

Using these functions, we can now define our linear system. We first define the element function matrix and constraints:

$$L_{ij} = \int_M \langle \nabla_M \phi_i, \nabla_M \phi_j \rangle dp \qquad b_i = \int_M \langle \nabla_M \phi_i, w_T \rangle dp.$$

Then, the linear system with respect to the reduced basis becomes:

$$\begin{pmatrix} \hat{P}^* L \hat{P} & \hat{P}^* L \tilde{P} \\ \tilde{P}^* L \hat{P} & \tilde{P}^* L \tilde{P} \end{pmatrix} \begin{pmatrix} \hat{x} \\ \tilde{x} \end{pmatrix} = \begin{pmatrix} \hat{P}^* b \\ \tilde{P}^* b \end{pmatrix},$$

where $\hat{x}$ are the coefficients of the solution with respect to the reduced translational basis functions $\hat{B}^m$, $\tilde{x}$ are the coefficients of the solution with respect to the reduced positional basis functions $\tilde{B}_i$, and $\hat{P}^*$ (resp. $\tilde{P}^*$) is the transpose conjugate of $\hat{P}$ (resp. $\tilde{P}$).

Restoring Translational Degrees of Freedom. While this approach ensures a consistent parameterization, it is possible at low grid resolutions that too many translational constraints will be introduced and the system will not have sufficient degrees of freedom to provide a high-quality solution. To address this, we modify our implementation to use the local domain graphs only to define the reduced basis, but not to define the translation constraints. For the translational constraints, we use the minimal set of constraints that must be satisfied by any basis – the constraints defined by using the hat basis.

Unfortunately, using such an approach no longer guarantees that we generate a consistent basis for the parameterization. We correct this by (1) identifying the basis functions whose local domain graphs generate additional translation constraints (these are constraints that are linearly independent of the minimal constraint set), (2) marking the triangles in the support of these basis functions as triangles on which we may have an inconsistent parameterization, and (3) performing a subsequent solve on the interior of the marked region, defining a linear system with the hat basis and using a direct solver. A solution before and after this post-processing solve is shown in Figure 6.

While this stage requires the use of a direct solver on a system defined by the hat basis, only a small fraction of the mesh is re-solved, and this solve's cost is often negligible.

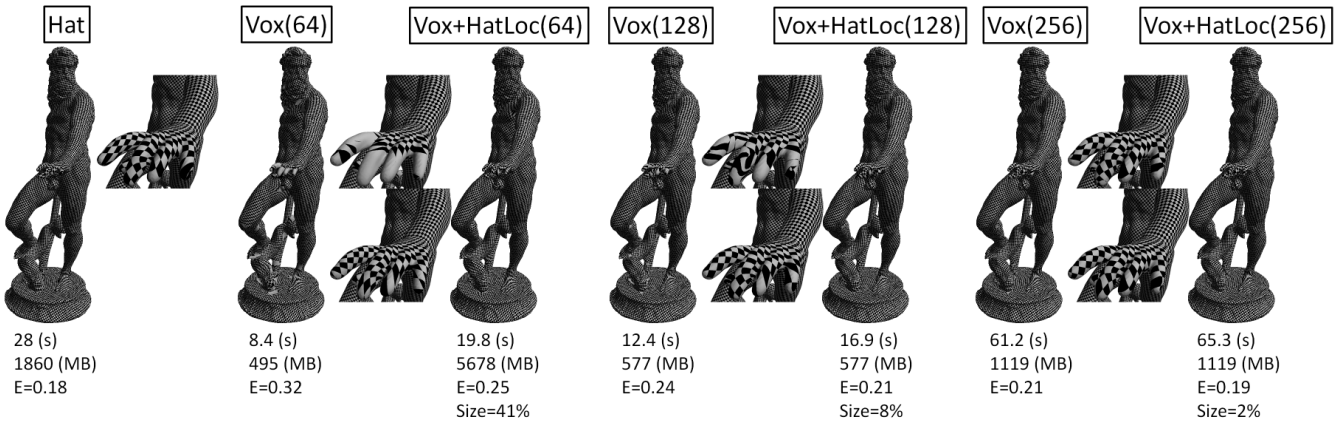

Figure 7: *The traditional hat basis parameterization (left) compared to the parameterizations obtained using the volumetric basis (right) with voxel grids of resolution 64^3, 128^3, and 256^3 (from left to right). For each volume basis, solutions without (left and top) and with (right and bottom) local refinement are shown. The solve time and memory requirements are reported below each result, along with the result's L^2 gradient error. For solves with local refinement, the percentage of mesh vertices in the refinement region is shown.*

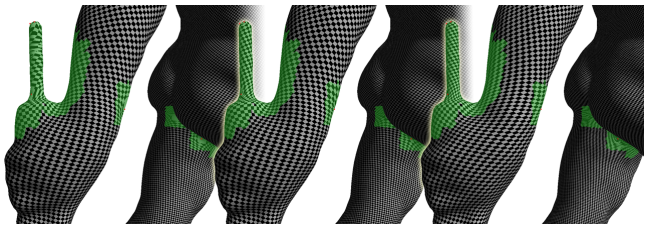

Figure 6: *Left: inconsistent solution computed by the underconstrained volume basis solver (note the high distortion on the narrow feature on the left). Center: a local refinement in the hat basis fixes the inconsistency. Right: a full hat solve for comparison. Triangles in the support of the cone-overlapping basis functions (on which the local solve is run) are shown in green.*

4.3 Solving for a seamless parameterization with integer translations

We compute the global parameterization in four steps.

1. Using the system described above, we solve for the independent positional and translational coefficients with respect to the reduced system. This gives a solution that satisfies the rotational constraints across the seams but is not seamless, Figure 8(a).

2. We round the translational coefficients to integer values so that integer isolines match across the boundaries. The resulting solution satisfies the constraints, but the rounding introduces high-frequency error, Figure 8(b).

3. We solve the system using the volumetric basis a second time, this time fixing the translational coefficients at their rounded values. This gives a seamless solution that is smooth with respect to the restricted B-splines system, Figure 8(c). However, this solution may have collapsed some regions to a single point in the parametric domain.

4. We use the hat basis to define a linear system for the interior of all triangles marked for post-processing. We then solve this system for the region's positions and free translations using a direct solver (locking the boundary values), Figure 8(d).

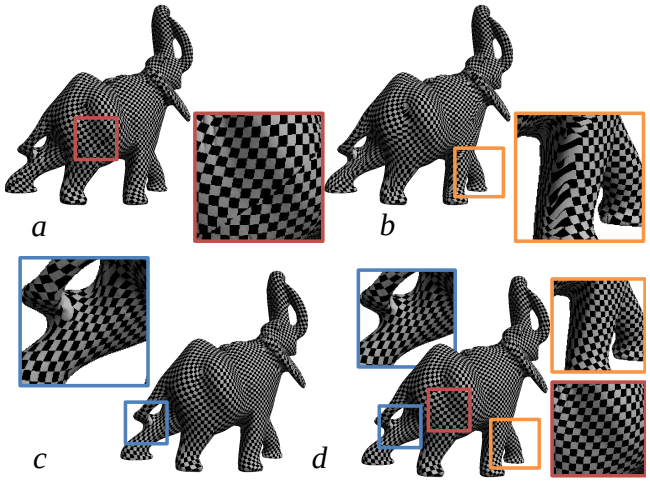

Figure 8: *Computing the parameterization: (a) We solve for the translational and positional constraints using the volumetric solver. (b) We round the translation constraints to obtain a seamless (but not smooth) solution. (c) Keeping the translational coefficients locked, we perform a second solve using the volumetric basis to refine the positional coefficients. (d) We perform a direct hat basis solve on the regions where the parameterization may be inconsistent across seams (if we chose to omit those seams' translational constraints) or where the solution collapsed.*

Alternatively, the more precise but expensive greedy rounding of [Bommes et al. 2009] can be used.

5 Results

In this section, we explore the performance of our method and compare it to the solution obtained using the hat basis.

Volumetric solutions at different resolutions. First, we look at the solutions using the volumetric basis at different resolution voxel grids to demonstrate the impact of discretization (Figure 7). The

image to the left shows a checkerboard pattern for the parameterization computed using the traditional hat basis. The pairs of images to the right show the parameterizations obtained using the volumetric basis at voxel resolutions 64^3, 128^3, and 256^3 both with (right and bottom inset) and without (left and top inset) a subsequent local refinement using the hat basis.

Examining the figure, we make several observations. First, while the volumetric basis generates a parameterization with noticeable artifacts, particularly near singularities, these are effectively removed using the localized refinement: starting at resolution 128^3, the solution obtained using the volumetric basis in conjunction with localized refinement is qualitatively similar to the solution returned by the hat solver. This qualitative behavior can be validated quantitatively by measuring the L^2-difference between the parameterizations' gradients and the guidance field. As the errors reported in Figure 7 show, incorporating the localized refinement makes the volumetric solution error similar to that of the hat solution. Since the original guidance field was defined over the triangles of the mesh, we expect the hat basis to perform slightly better since its level-of-detail exactly matches that of the constraints.

Finally, we note that as the resolution of the grid grows coarser, the region requiring local refinement grows, and there is a trade-off between the efficiency of the volumetric solver and the inefficiency of the localized refinement at coarser resolution.

Comparative performance of the volumetric and hat solvers. We perform two comparisons. First, we benchmark the memory and time performance of the volumetric solver (Vox) and the volumetric solver followed by a hat solver localized to regions of inconsistency and collapse (Vox+HatLoc), comparing them against the global hat basis solve (Hat). We used the direct solver CHOLMOD [Davis and Hager 2005] to solve all of our sparse linear systems. (We also experimented with using the multigrid solver of [Chuang et al. 2009] but found that the system was small enough at low voxel resolutions that the highly optimized direct solver was competitive with the hierarchical solver.) In these evaluations, we use a resolution 128^3 voxel grid for the volumetric basis; we found this basis resolution generates parameterizations that, relative to the hat basis parameterizations, have similar numbers of foldovers and have gradient deviation magnitudes that are never more than 25% greater.

The results of these comparisons over a broad set of meshes are provided in Figure 9. Our meshes are shown in Figure 11 and include high-resolution geometry obtained from range scans and meshed using [Kazhdan et al. 2006]: the Awakening mesh at depths 9, 10, and 11 (284K, 1M, and 5M triangles); the Neptune mesh at depths 8, 9, 10, and 11 (89K, 322K, 1M, and 6M triangles); the Lucy mesh at depth 11 (7M triangles); and the highest resolution mesh in our data set, the David head mesh at depth 11 (20M triangles). We also included artificially subdivided meshes: the hand mesh, subdivided three times using 1-to-4 subdivision (65K, 262K, 1M, and 4M triangles) and the Fertility mesh, subdivided four times (27K, 112K, 447K, 2M, and 7M triangles).

As the plots in Figure 9 indicate, though the hat basis provides a more efficient solution for coarse meshes, the relative performance of the hat basis solver deteriorates as the meshes get larger. In particular, we see that the volumetric solver (defined over a 128^3 grid) becomes more time- and space-efficient as the model sizes begin to exceed 250K triangles, approximately the size of models typically used in the previous work on global parameterization.

Note the substantial variation in performance of both hat and volume bases for meshes of approximately the same size. This variation is due to a significant difference in geometric complexity and, consequently, a significant variation in the number of cones. Although the associated number of translational degrees of freedom

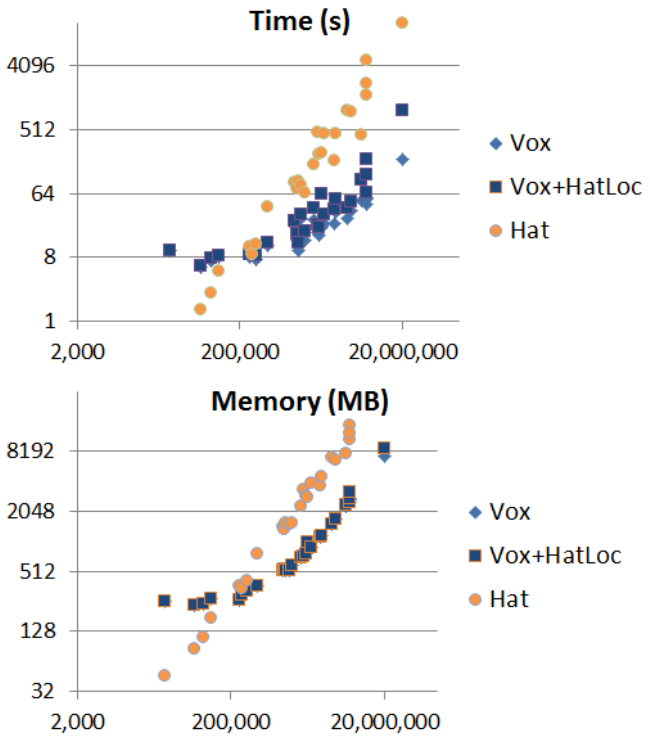

Figure 9: *Performance analysis for all meshes in our dataset. Running time and memory consumption of the volumetric solver at grid resolution* 128^3*, both without and with local refinement, is compared against the hat solver.*

is small in all cases, it has a significant impact on the sparsity structure of the matrix, as each translational basis function has relatively large support and many more nonzero entries in its row (typically hundreds or thousands).

To better understand the characteristics of the volumetric solver, we perform a second comparison, this time over a set of successively higher resolution meshes of a single geometry, using the same cross-field projected to each mesh. In this way, the effects of the translational degrees of freedom can be excluded. The results of these experiments, one for the Fertility statue and one for Neptune statue, are shown in Figure 10. The figure compares the performance of the hat solver to the volumetric solver at resolutions 64^3 and 128^3 with local refinement.

6 Conclusions

In this work, we have generalized the constraint framework for generating seamless mesh parameterizations to extend beyond the classical hat basis. We have shown that this allows us to define a finite-elements solution to the parameterization problem using general function spaces, and we have adopted the recent volumetric system to support a solver that can seamlessly parameterize large meshes more quickly and with significantly less memory than using the hat basis.

In future work, we would like to consider several areas of research. First, we want to extend our approach to support matrix stiffening to reduce the fold-overs. (Our finite-elements formulation is particularly well-suited to this task since introducing stiffening weights only requires scaling the element coefficients appropriately.) We

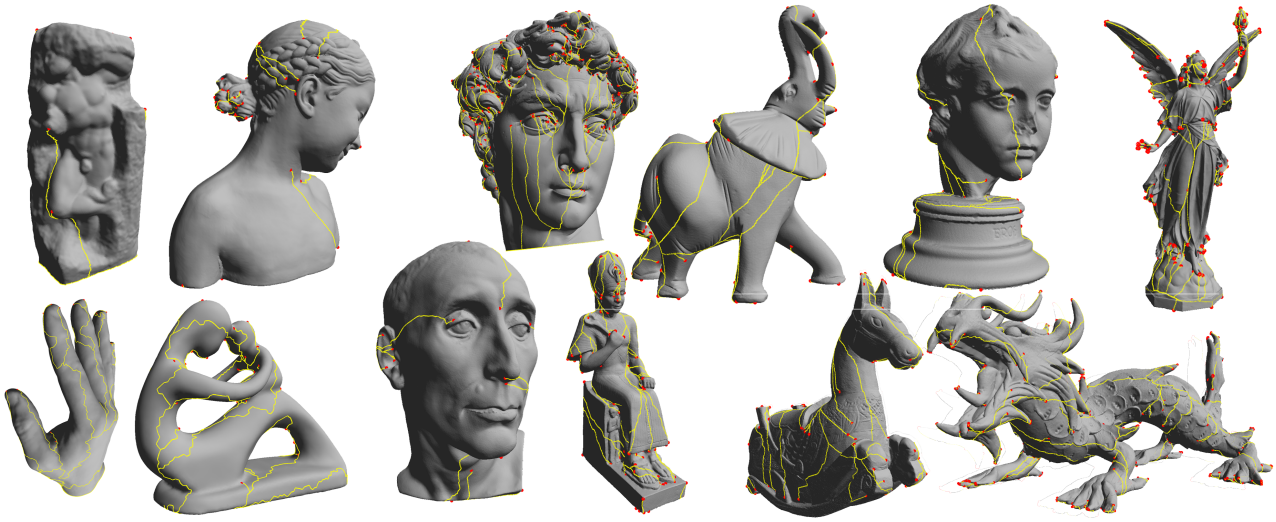

Figure 11: *Meshes used in our experiments (left-right, top-bottom): Awakening (multiple resolutions); Bimba (1M triangles); David head (20M triangles); Elephant (3M triangles); Eros (950K triangles); Lucy (7M triangles); Hand (65K triangles, subdivided up to 4M); Fertility (27K triangles, subdivided up to 7M); Nicolo (2M triangles); Ramesses (2M triangles); Isidore Horse (2M triangles); and Dragon (7M triangle original mesh, and a 2M triangle downsampling). Data set also includes multiple resolutions of the Neptune mesh, seen in Figure 7.*

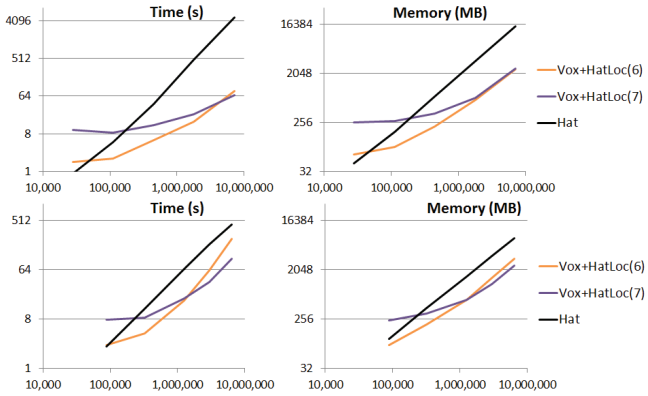

Figure 10: *Performance analysis for successively higher subdivisions of the Fertility mesh (top) and higher resolution reconstructions of the Neptune scans (bottom): running times and memory consumptions of the resolution 64^3 and 128^3 volumetric solver (with local refinement) are compared against the hat solver.*

want to explore more robust methods for rounding translations in order to ensure that two cones do not get rounded to the same lattice position. Finally, we would also like to explore a more adaptive discretization in which the resolution of the grid can be refined around cones in order to minimize the effects of triangle collapse.

References

AKSOYLU, B., KHODAKOVSKY, A., AND SCHRÖDER, P. 2005. Multilevel solvers for unstructured surface meshes. *SIAM Journal of Scientific Computing 26*, 1146–1165.

BEN-CHEN, M., GOTSMAN, C., AND BUNIN, G. 2008. Conformal Flattening by Curvature Prescription and Metric Scaling. In *Computer Graphics Forum*, vol. 27, Blackwell Synergy, 449–458.

BOMMES, D., ZIMMER, H., AND KOBBELT, L. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3, 77.

CHUANG, M., LUO, L., BROWN, B., RUSINKIEWICZ, S., AND KAZHDAN, M. 2009. Estimating the laplace-beltrami operator by restricting 3d functions. In *Proceedings of the Symposium on Geometry Processing*, Eurographics Association, 1475–1484.

CLARENZ, U., DIEWALD, U., AND RUMPF, M. 2000. Anisotropic geometric diffusion in surface processing. In *Visualization '00: Proceedings of the 11th IEEE Visualization 2000 Conference (VIS 2000)*, 497–405.

DANIELS, J., SILVA, C., AND COHEN, E. 2009. Semi-regular quadrilateral-only remeshing from simplified base domains. In *Computer Graphics Forum*, vol. 28, Blackwell Publishing Ltd, 1427–1435.

DANIELS, J., SILVA, C. T., AND COHEN, E. 2009. Localized quadrilateral coarsening. *Computer Graphics Forum 28*, 5, 1437–1444.

DAVIS, T., AND HAGER, W. 2005. Row modifications of a sparse cholesky factorization. *SIAM Journal on Matrix Analysis and Applications 26*, 3, 621–639.

DONG, S., BREMER, P., GARLAND, M., PASCUCCI, V., AND HART, J. 2006. Spectral surface quadrangulation. *ACM Trans. Graph. 25*, 3, 1057–1066.

ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., AND STUETZLE, W. 1995. Multiresolution analysis of arbitrary meshes. *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 173–182.

GU, X., AND YAU, S. 2003. Global conformal surface parameterization. *Symposium on Geometry Processing*, 127–137.

HORMANN, K., LÉVY, B., AND SHEFFER, A. 2007. Mesh parameterization: Theory and practice. *SIGGRAPH Course Notes*.

KÄLBERER, F., NIESER, M., AND POLTHIER, K. 2007. QuadCover: Surface Parameterization using Branched Coverings. *Computer Graphics Forum 26*, 3, 375–384.

KAZHDAN, M., BOLITHO, M., AND HOPPE, H. 2006. Poisson surface reconstruction. *Computer Graphics Forum (SGP 2006) 25*, 61–70.

KHODAKOVSKY, A., LITKE, N., AND SCHRÖDER, P. 2003. Globally smooth parameterizations with low distortion. *ACM Trans. Graph. 22*, 3, 350–357.

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H. 1998. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 105–114.

KOVACS, D., MYLES, A., AND ZORIN, D. 2009. Anisotropic harmonic quadrangulation. In *Symposium on Geometry Processing 2009 Poster*.

LEE, A., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. 1998. MAPS: multiresolution adaptive parameterization of surfaces. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM New York, NY, USA, 95–104.

MARINOV, M., AND KOBBELT, L. 2005. Automatic generation of structure preserving multiresolution models. In *Computer Graphics Forum*, vol. 24, Amsterdam: North Holland, 1982-, 479–486.

PIETRONI, N., TARINI, M., AND CIGNONI, P. 2009. Almost isometric mesh parameterization through abstract domains. *IEEE Transactions on Visualization and Computer Graphics 99*, RapidPosts.

PIETRONI, N., TARINI, M., SORKINE, O., AND ZORIN, D. 2011. Global parametrization of range image sets. *ACM Transactions on Graphics (TOG) 30*, 6, 149.

RAY, N., AND LEVY, B. 2003. Hierarchical least squares conformal map. In *Pacific Graphics*, 263.

RAY, N., LI, W., LÉVY, B., SHEFFER, A., AND ALLIEZ, P. 2006. Periodic global parameterization. *ACM Trans. Graph. 25*, 4, 1460–1485.

RUGE, J., AND STUEBEN, K. 1987. Algebraic multigrid. 73–130.

SCHENK, O., GÄRTNER, K., FICHTNER, W., AND STRICKER, A. 2001. PARDISO: A high-performance serial and parallel sparse linear solver in semiconductor device simulation. *Journal of Future Generation Computers Systems 18*, 69–78.

SCHNEIDER, R., AND KOBBELT, L. 2001. Geometric fairing of irregular meshes for free-form surface design. *Computer Aided Geometric Design 18*, 359–379.

SHEFFER, A., PRAUN, E., AND ROSE, K. 2006. Mesh parameterization methods and their applications. *Foundations and Trends® in Computer Graphics and Vision 2*, 2, 171.

SHI, L., YU, Y., BELL, N., AND FENG, W.-W. 2006. A fast multigrid algorithm for mesh deformation. *ACM Transactions on Graph (SIGGRAPH '06) 25*, 3, 1108–1117.

SPRINGBORN, B., SCHRÖDER, P., AND PINKALL, U. 2008. Conformal equivalence of triangle meshes.

TARINI, M., PIETRONI, N., CIGNONI, P., PANOZZO, D., AND PUPPO, E. 2010. Practical quad mesh simplification. *Computer Graphics Forum 29*, 2.

TONG, Y., ALLIEZ, P., COHEN-STEINER, D., AND DESBRUN, M. 2006. Designing quadrangulations with discrete harmonic forms. *Symposium on Geometry Processing*, 201–210.

ZHANG, M., HUANG, J., LIU, X., AND BAO, H. 2010. A wave-based anisotropic quadrangulation method. *ACM Transactions on Graphics (TOG) 29*, 4, 1–8.