Eurographics Symposium on Geometry Processing 2017
J. A. Bærentzen and K. Hildebrandt
(Guest Editors)

Volume 36 (2017), Number 5

Isometry-Aware Preconditioning for Mesh Parameterization

S. Claici,[1] M. Bessmeltsev,[1] S. Schaefer,[2] and J. Solomon[1]

[1]MIT, CSAIL, Cambridge, USA
[2]Texas A&M University, College Station, USA

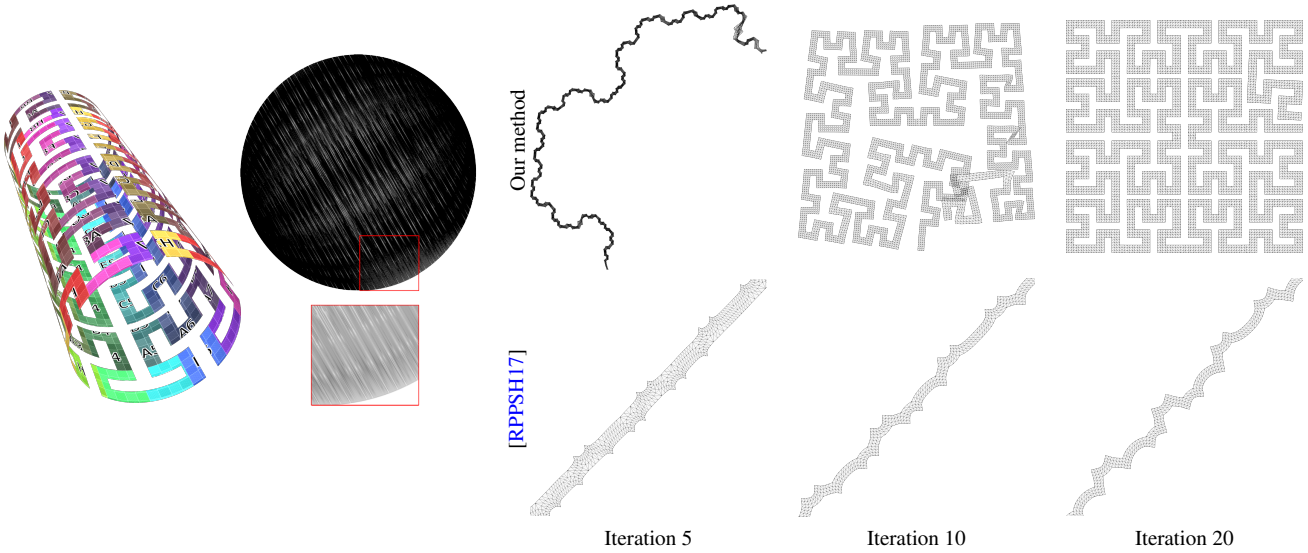


Figure 1: *Comparison of our method against SLIM [RPPSH17] on a particularly challenging test case. Both methods were run for 20 iterations. The first two columns show the mesh and initial parameterization shared by both methods. The remaining columns show the UV maps obtained by our algorithm (top) and SLIM (bottom) at iterations 5, 10, and 20. Our AKVF preconditioner converges quickly to a distortion error near the lower bound, while SLIM makes slow progress after the first few iterations. For detail, we show a zoomed-in version of the SLIM parameterization at each iteration. Unlike SLIM, we do not require Newton iterations to reach the global optimum.*

Abstract

This paper presents a new preconditioning technique for large-scale geometric optimization problems, inspired by applications in mesh parameterization. Our positive (semi-)definite preconditioner acts on the gradients of optimization problems whose variables are positions of the vertices of a triangle mesh in $\mathbb{R}^2$ or of a tetrahedral mesh in $\mathbb{R}^3$, converting localized distortion gradients into the velocity of a globally near-rigid motion via a linear solve. We pose our preconditioning tool in terms of the Killing energy of a deformation field and provide new efficient formulas for constructing Killing operators on triangle and tetrahedral meshes. We demonstrate that our method is competitive with state-of-the-art algorithms for locally injective parameterization using a variety of optimization objectives and show applications to two- and three-dimensional mesh deformation.

1. Introduction

Mesh parameterization is a fundamental member of the geometry processing toolkit for computer graphics. Among other applications, parameterization is critical for texture mapping, remeshing, decimation, and attribute transfer.

From a geometric standpoint, the language of discrete differential geometry suggests elegant formulations of mesh parameterization objective functions. As a result, the objectives optimized by modern parameterization algorithms are endowed with both smooth and discrete interpretations. These interpretations allow the objectives to be understood as discretized adaptations of continuous

quantities on a smooth surface as well as purely discrete measures of triangle distortion.

With this geometric sophistication, however, comes demand for complex optimization algorithms. From an optimization standpoint, parameterization problems involve minimization of nonconvex objectives measuring distortion of a map into the plane as well as bijectivity constraints preventing triangles from flipping over. These problems are highly structured, generally built from triplet terms reflecting the topology of the underlying triangle mesh. Moreover, the parameterization objectives may exhibit special properties like rotation invariance, which present challenges for generic optimization tools.

In this paper, we take inspiration from recent *preconditioners* that accelerate first-order (gradient-based) optimization for mesh parameterization [SS15, KGL16, RPPSH17]. We formulate a new preconditioner specifically designed for parameterization problems, using the language of vector field design. Our preconditioner is built from the intuition that a triangle with zero distortion should be moved rigidly as the parameterization is updated elsewhere on the mesh. We formalize this idea using the framework of approximate Killing vector fields (AKVFs) on two-dimensional shapes [BCBSG10, SBCBG11a].

Our method achieves state-of-the-art performance on challenging parameterization tasks. To accelerate application of our preconditioner in each iteration, we present a simple formula for assembling KVF operators directly from mesh geometry. From a theoretical standpoint, we present a Riemannian interpretation of our preconditioner, showing how the preconditioner navigates the space of mesh parameterizations endowed with a natural metric. Beyond mesh parameterization, we show how an analogous operator can precondition problems on tetrahedral meshes and demonstrate its application to deformation of volumetric objects.

Contributions. This paper contributes several new ideas to parameterization and related optimization problems in geometry. We stress that our method matches or exceeds the performance of existing methods [KGL16, RPPSH17] while enjoying additional theoretical understanding and a remarkably simple matrix construction. Specific contributions include the following:

- A positive (semi-)definite, rotation-invariant preconditioner for gradient fields that penalizes non-isometric deformations and is suited to minimization problems involving distortion energies
- Simple, easy-to-code, closed-form expressions for our preconditioner both in the planar and volumetric cases in terms of basic mesh elements, which can be assembled in parallel without matrix multiplication [SBCBG11a] or per-element eigenvector/SVD computation [RPPSH17]
- Interpretation of our preconditioner as the gradient with respect to a natural AKVF-based metric on the "space of parameterizations" of a mesh with fixed topology
- Application of our method to mesh parameterization problems, with improved performance over state-of-the-art
- Application to volumetric problems over tetrahedral meshes

2. Related Work

2.1. Mesh Parameterization

Surface parameterization is a fundamental and well-studied problem in computer graphics. We refer the reader to several surveys for more complete background [FH05, SPR06].

Early work on parameterization concentrated on solving linear systems of equations to find parameterizations. These solutions, however, typically require constrained boundaries [Tut63, Flo97] to guarantee locally injective parameterizations or suffer from excessive shrinking [LPRM02]. Hence, more recent work has concentrated on nonlinear problems that produce high-quality parameterizations.

As-rigid-as-possible parameterization (ARAP) [LZX*08a] uses a local-global optimization approach to optimize a measure of isometric distortion. This algorithm involves repeatedly finding the best rigid transformation for each triangle and stitching the triangles together through a global, linear system of equations. This choice of objective function, however, does not guarantee that the parameterization is locally injective. Other nonlinear metrics guarantee that the parameterizations will be locally injective with the use of an appropriate interior point solver. Examples include measures of conformal distortion [HG00, DMK03] as well as measures of isometric distortion [SCOGL02, APL14]. We concentrate on the symmetric Dirichlet energy [SS15] here for its high-quality results and simple expression, although our preconditioner applies to any rotationally-invariant distortion energy.

Minimizing such nonlinear distortion measures requires nonlinear optimization procedures, and a variety of such methods have been proposed in the past. Hormann and Greiner [HG00] utilize a single-vertex-at-a-time update to guarantee injective parameterizations, though such an optimization is slow in practice. Fu et al. [FLG15] employ a highly-parallel version of localized gradient descent to obtain reasonable optimization times. Smith and Schaefer [SS15] use LBFGS coupled with an explicit bound on the line search to guarantee a bijective parameterization. Other algorithms optimize for injective parameterizations with bounded distortion through a change of basis utilizing edge lengths [CLW16] or affine transformations [FL16]. Sheffer et al. [SLMB05] propose a hierarchical method for conformal distortion.

More recently, several researchers have developed optimization techniques specific to rotationally-invariant metrics. Kovalsky et al. [KGL16] use a Newton-like algorithm for rotationally invariant functions that can be written as a quadratic energy whose Hessian is the mesh cotangent Laplacian [PP93] plus a nonlinear correction term. The search direction for this nonlinear method is obtained by applying the inverse of the Laplacian, which is constant from iteration to iteration, to the negative gradient direction of the full energy function. Such a modification significantly decreases the number of iterations needed to converge versus a naïve quasi-Newton method [SS15]. Our method resembles this approach, but we utilize the inverse of the KVF operator for the current parameterization instead of the Laplacian of the 3D mesh.

Rabinovich et al. [RPPSH17] employ a modified version of ARAP to optimize rotationally-invariant energies. The idea is to modify the local-global approach of ARAP to reweight an ARAP

proxy energy such that the gradient of the reweighted energy is the same as the gradient of the parameterization objective. Their algorithm minimizes this proxy to obtain a search direction for the nonlinear optimization. The result is a very fast optimization method that can be applied to massive meshes. We compare our results with this technique in §6.

These ideas bear similarity to Sobolev gradient methods [MJBC13, RN95]; specifically, preconditioning via the Laplacian can be seen as a gradient in the Sobolev space H^1. If the energy functional lies in a Sobolev space H^n, using the associated Sobolev gradient may lead to better performance, since the Sobolev gradient maintains smoothness of the functional. These methods, however, are agnostic to the underlying task and, to our knowledge, have not been applied to mesh parameterization.

2.2. Killing Vector Fields

Our preconditioner for mesh parameterization is constructed by approximating optimization gradients with Killing vector fields (KVFs). The idea of computing approximate Killing vector fields (AKVFs) first appeared in the geometry processing literature in [BCBSG10], in the context of vector field design on surfaces. This approach, specialized to the case of two-dimensional shapes in the plane, was applied by Solomon et al. [SBCBG11a] to planar deformation; it is their discretization of the AKVF operator that we use for parameterization, although we contribute a novel formula that accelerates matrix construction.

Alternative AKVF discretizations include [ABCCO13, AOCBC15], which use functional maps [OBCS*12] to represent AKVFs as differential operators, and [dGLB*14], which builds a discrete theory of 2-tensor fields on triangulated polygons in the plane using an edge-based representation. Some papers have considered alternative discretizations and applications for AKVFs. Solomon et al. [SBCBG11b] use AKVFs to partition surfaces into parts with intrinsic symmetries, and Tao et al. [TSB16] use AKVFs on a voxel grid to track moving level set surfaces. AKVF-type operators also appear in approximately developable deformation [KMP07], fluid simulation [NVW12], and nonrigid registration for microscopy [CW13].

AKVF computation more broadly is a task in vector field design. See recent surveys by de Goes et al. [dGDT15] and Vaxman et al. [VCD*16] for summaries of recent work in this area.

2.3. Manifold Approaches to Shape Analysis

We motivate our preconditioner as a gradient in the "manifold of mesh parameterizations," isomorphic to the $(2|V|-3)$-dimensional space of vertex positions modulo rigid motion. This idea of defining a Riemannian metric in shape space has appeared in previous works across a few communities. In computer graphics, this approach appears in work on shape interpolation including [KMP07], which defines Riemannian metrics for triangle stretching in shape interpolation, and [HRWW12], which interpolates between surfaces using a metric inspired by thin-shell deformation. Other appearances of a similar idea include applications to image registration [DGM98], volumetric shape deformation [FW06], 2D curve deformation [MM04], surface registration [BB11], and manipulation of parameterized surfaces [KKDS10, KKG*12].

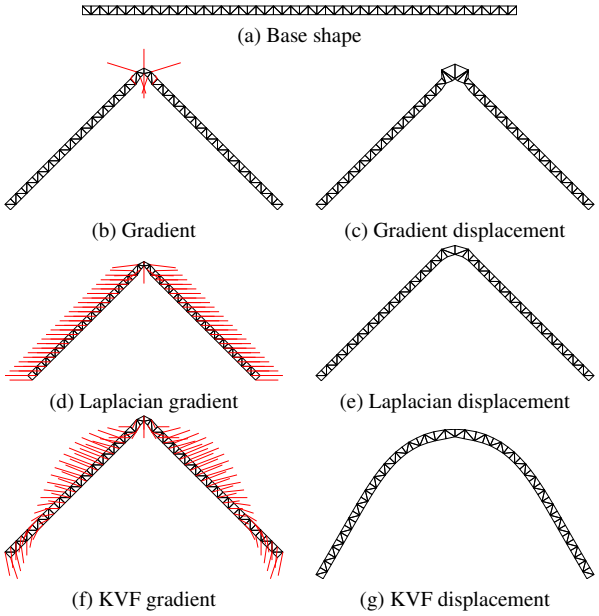


Figure 2: *Motivation for our preconditioner: let the bent rod (b) be our initial parameterization of (a). Most individual faces have 0 distortion. By linearity, the gradient will be non-zero only at the vertices adjacent to the deformed faces, which leads to non-rigid motion. In contrast, our AKVF preconditioner aims to maintain a search direction that is as isometric as possible. The preconditioned gradient of (f) seeks to "un-bend" the parameterization back into a rectangle. We observe similar behavior, though less pronounced, when using the mesh Laplacian as a preconditioner in (d) and (e).*

3. Preliminaries and Motivation

As motivation for our work and to establish notation, we will consider a simple example illustrated in Figure 2: the task of parameterizing a rectangular object (Figure 2(a)) in the plane.

If the mesh has vertices V and triangular faces F, following the notation of Rabinovich et al. [RPPSH17] we can view parameterization algorithms as techniques for minimizing a functional

$$E(\mathbf{x}) := \sum_{f \in F} \hat{A}_f \mathcal{D}(\mathbf{J}_f(\mathbf{x})). \tag{1}$$

Here, $\mathbf{x} \in \mathbb{R}^{2|V|}$ encodes the positions of all points in the parameterization, $\hat{A}_f \in \mathbb{R}_+$ is the area of the undistorted triangle f, $\mathbf{J}_f \in \mathbb{R}^{2\times 2}$ is the Jacobian of the affine deformation of the triangle $f \in F$ modulo a rigid transformation of the triangle from $3D$ to the plane, and $\mathcal{D} : \mathbb{R}^{2\times 2} \to \mathbb{R}$ is a distortion measure per triangle. Intuitively, (1) states that the distortion of mapping a mesh into the plane is the sum of the distortions of its individual triangles.

Examples include the as-rigid-as-possible energy [LZX*08b]

$$\mathcal{D}_{\text{ARAP}}(\mathbf{J}_f(\mathbf{x})) := \|\mathbf{J}_f(\mathbf{x}) - R(\mathbf{J}_f(\mathbf{x}))\|^2_{\text{Fro}},$$

where $R(\cdot)$ projects onto the closest rotation matrix, and the flip-preventing symmetric Dirichlet energy [SAPH04, SS15]

$$\mathcal{D}(\mathbf{J}_f(\mathbf{x})) := \|\mathbf{J}_f(\mathbf{x})\|_{\text{Fro}}^2 + \|\mathbf{J}_f^{-1}(\mathbf{x})\|_{\text{Fro}}^2. \tag{2}$$

First-order techniques iteratively improve a parameterization from an initial guess $\mathbf{x}^0$ by moving along the negative gradient $-\nabla_{\mathbf{x}} E(\mathbf{x})$. By linearity, the gradient of Equation 1 is

$$\nabla_{\mathbf{x}} E(\mathbf{x}) = \sum_{f \in F} \hat{A}_f \nabla_{\mathbf{x}} \mathcal{D}(\mathbf{J}_f(\mathbf{x})). \tag{3}$$

As an extreme example of what can go wrong, suppose $\mathbf{x}^0$ is a bent bar in the example in Figure 2. Here, most triangles undergo only rotation from 3D to 2D, and hence their distortion is zero, yielding $\nabla_{\mathbf{x}} \mathcal{D}(\mathbf{J}_f(\mathbf{x})) = 0$. The weighted sum (3) is comprised of a few terms involving vertices adjacent to distorted triangles. This leads to a *sparse* gradient field, as in Figure 2(b).

Minimizing $E(\cdot)$ will seek to "un-bend" $\mathbf{x}^0$ back into a rectangle. But if we search along $-\nabla_{\mathbf{x}} E(\mathbf{x})$, this does not happen. Rather, we would only move the vertices adjacent to distorted triangles (Figure 2(c)). The gradient disregards the fact that if we displace the distorted vertices alone, their neighbors become distorted instead. What we would prefer is a *near-rigid motion* of the undistorted vertices whose velocity shown in Figure 2(f) yields a smooth shape as in Figure 2(g).

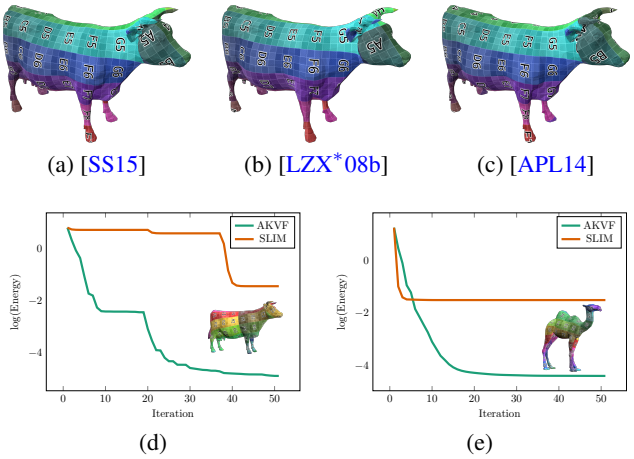
Our preconditioner transforms vector fields like the one in Figure 2(b) to fields like the one in Figure 2(f) using the Killing operator $K(\mathbf{x})$. $K(\mathbf{x})$ measures the deviation of a vector field on $\mathbf{x}$ from being a rigid motion—specifically rotation or translation—so the product $-K(\mathbf{x})^+ \nabla_{\mathbf{x}} E(\mathbf{x})$ transforms the descent direction into an approximately rigid motion when possible. Since $K(\mathbf{x})$ is symmetric and positive (semi-)definite, this transformation maps descent directions to descent directions, preserving gradient descent (with or without line search) as an effective means for optimization. The null space of $K(\mathbf{x})$ corresponds to rigid motions that can be ignored in parameterization problems, as translating/rotating a parameterization does not affect its quality.

4. As-Killing-As-Possible Preconditioner

Our insight is that an operator proposed for shape deformation and vector field design serves as an effective preconditioner for parameterization and related tasks. This operator is easily computed from mesh geometry via new closed-form expressions we present below.

4.1. Motivation for Use in Parameterization

Suppose we are in the process of parameterizing a triangle mesh. The vertex positions of the current parameterization are stored in a vector $\mathbf{x} \in \mathbb{R}^{2|V|}$. In an iteration, we might update $\mathbf{x} \mapsto \mathbf{x} + \mathbf{d}$ for some search direction $\mathbf{d} \in \mathbb{R}^{2|V|}$. While $\mathbf{d}$ is a $2|V|$-dimensional vector, its role is different from $\mathbf{x}$: whereas $\mathbf{x}$ determines a parameterization, $\mathbf{d}$ is a *change* in parameterization. That is, $\mathbf{d}$ lies in the tangent space of the set of parameterizations. Note that $\mathbf{d}$ is a *vector field* along the planar region determined by $\mathbf{x}$; we interpret $\mathbf{d}$ as a velocity field rather than a displacement field, under the assumption it is small relative to $\mathbf{x}$.



(a) [SS15] (b) [LZX*08b] (c) [APL14]

(d) (e)

Figure 3: *Minimizing isometric distortion energies for the Cow model. Our preconditioner can be applied generally to any gradient descent method. Here we show results after 100 iterations using (a) the symmetric Dirichlet energy [SS15] (b) as rigid as possible (ARAP) energy [LZX*08b] (c) an isometric distortion energy [APL14]. In (d) and (e) we compare with SLIM using the ARAP energy [LZX*08b]; SLIM's poor performance is a result of ARAP's performance near local minima.*

Suppose we measure the *magnitude* $\|\mathbf{d}\|$ of $\mathbf{d}$, such that large changes in the parameterization correspond to $\mathbf{d}$'s with large magnitude. It is tempting to use the Euclidean norm $\|\mathbf{d}\|_2$, but this arguably is incorrect. In particular, any $\mathbf{d}$ representing a rigid motion (rotation or translation) of $\mathbf{x}$ may be nonzero but does not affect the parameterization and should be considered to have zero magnitude. More broadly, a $\mathbf{d}$ encoding *nearly*-rigid motion likely should have low magnitude, even if $\|\mathbf{d}\|_2$ is large.

A vector field $\mathbf{d}$ is the velocity of a rigid motion exactly when its Jacobian $J_{\mathbf{d}} \in \mathbb{R}^{2\times 2}$ is antisymmetric. This gives rise to the *Killing energy* of a vector field $\mathbf{d}$ [BCBSG10], defined as

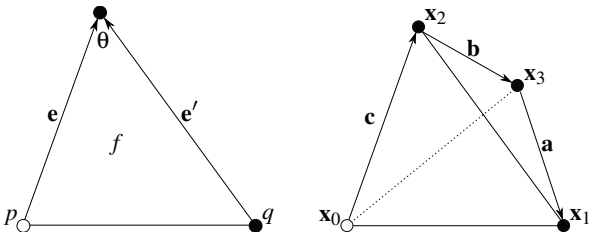
$$\|\mathbf{d}\|_{\text{KVF}}^2 := \frac{1}{2} \int_{\Omega(\mathbf{x})} \|J_{\mathbf{d}} + J_{\mathbf{d}}^\top\|_{\text{Fro}}^2 \, dA, \tag{4}$$

where $\Omega(\mathbf{x}) \subseteq \mathbb{R}^2$ is the planar region defined by the parameterization $\mathbf{x}$. Unlike $\|\mathbf{d}\|_2$, the norm $\|\mathbf{d}\|_{\text{KVF}}^2$ vanishes any time $\mathbf{d}$ is the velocity of a rigid motion. Since $J_{\mathbf{d}}$ is linear in $\mathbf{d}$, the entire expression for $\|\mathbf{d}\|_{\text{KVF}}^2$ is quadratic in $\mathbf{d}$, showing that we can write $\|\mathbf{d}\|_{\text{KVF}}^2 = \mathbf{d}^\top K(\mathbf{x}) \mathbf{d}$ for some symmetric, positive semidefinite matrix $K(\mathbf{x}) \in \mathbb{R}^{2|V|\times 2|V|}$. $K(\mathbf{x})$ is the *Killing operator* associated with the parameterization $\mathbf{x}$.

4.2. Discretization and Efficient Construction

4.2.1. Planar Case

Although there are many mesh-based discretizations of vector fields [dGDT15], our application must update $\mathbf{x}$, and hence we choose to represent $\mathbf{d}$ using one vector per vertex ($\mathbf{d} \in \mathbb{R}^{2|V|}$). Identical to [SBCBG11a], we use first-order (piecewise-linear) finite



(a) Triangle (b) Tetrahedron

Figure 4: *Notation for AKVF operator.*

elements (FEM) to compute the Killing energy (4) as a quadratic form $\|\mathbf{d}\|^2_{\mathrm{KVF}} = \mathbf{d}^\top K(\mathbf{x})\mathbf{d}$.

To maximize efficiency, we must *assemble* $K(\mathbf{x})$ from the parameterization $\mathbf{x}$ as quickly as possible. Whereas Solomon et al. [SBCBG11a] assemble $K(\mathbf{x})$ as a product of FEM matrices, we employ an explicit formula. Hence we prove:

Proposition 1 For a triangle mesh, $K(\mathbf{x})$ can be computed in 2×2 blocks corresponding to (x,y) coordinates as follows:

$$K_{pq} = \begin{cases} \sum_{f\sim(p,q)} \left[\frac{\mathbf{e}'\mathbf{e}^\top}{4\hat{A}_f} - (\cot\theta) I_{2\times 2}\right] & \text{if } p\sim q \\ -\sum_{r\sim p} K_{pr} & \text{if } p = q \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Here, $p\sim q$ indicates that $p,q\in V$ are adjacent, and $f\sim(p,q)$ indicates triangle $f\in F$ is adjacent to edge (p,q) (at most two terms). Figure 4(a) illustrates this notation where the edges $\mathbf{e},\mathbf{e}'$, area A_f, and angle θ are functions of $\mathbf{x}$.

While the cross term in (5) is an outer product $\mathbf{e}'\mathbf{e}^\top$, the cotangent term can be computed using an inner product $\cot\theta = \mathbf{e}^\top\mathbf{e}'/2A_f$.

We will prove the following more general formula.

Proposition 2 For a discretization $\mathbf{x}$ of a n-dimensional manifold, $K(\mathbf{x})$ can be computed in $n\times n$ blocks corresponding to $(x_1,\ldots,x_n)$ coordinates as follows:

$$K_{ij} = \begin{cases} \sum_{f\sim(i,j)} A_f \left[\nabla\phi_j\nabla\phi_i^\top + (\nabla\phi_i\cdot\nabla\phi_j) I_{n\times n}\right] & \text{if } i\sim j \\ -\sum_{k\sim i} K_{ik} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Here ϕ_i is the piecewise linear basis function that is 1 at vertex i and 0 elsewhere.

Proof Write an arbitrary vector $\mathbf{v}$ using linear interpolation functions: $\mathbf{v} = \sum_i \mathbf{v_i}\phi_i$. The Killing operator energy for a single triangle with vertices $i\in\{1,2,3\}$ is given by

$$\|K\mathbf{v}\|^2_{\mathrm{Fro}} = \frac{1}{2}\left\|\sum_i \mathbf{v_i}\nabla\phi_i^\top + \nabla\phi_i\mathbf{v_i}^\top\right\|^2_{\mathrm{Fro}}.$$

Using $\|A\|^2_{\mathrm{Fro}} = \mathrm{Tr}(AA^\top)$, we expand to

$$\|K\mathbf{v}\|^2_{\mathrm{Fro}} = \frac{1}{2}\mathrm{Tr}\Big(\sum_{i,j}[(\mathbf{v_i}\nabla\phi_i^\top)(\mathbf{v_j}\nabla\phi_j^\top)^\top + (\nabla\phi_i\mathbf{v_i}^\top)(\nabla\phi_j\mathbf{v_j}^\top)^\top + (\mathbf{v_i}\nabla\phi_i^\top)(\nabla\phi_j\mathbf{v_j}^\top)^\top + (\nabla\phi_i\mathbf{v_i}^\top)(\mathbf{v_j}\nabla\phi_j^\top)^\top]\Big).$$

Using $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$ and pulling out scalar terms when possible, we can simplify the above to

$$\|K\mathbf{v}\|^2_{\mathrm{Fro}} = \sum_{i,j}\mathbf{v_i}^\top(\nabla\phi_j\nabla\phi_i^\top + (\nabla\phi_i\cdot\nabla\phi_j)I_{n\times n})\mathbf{v_j},$$

which yields

$$K_{ij} = \nabla\phi_j\nabla\phi_i^\top + (\nabla\phi_i\cdot\nabla\phi_j)I_{n\times n}. \tag{7}$$

To obtain the Killing operator for the entire mesh, we scale by area weights and sum over all triangles. □

Equation (6) gives a general form for the KVF operator for n dimensional embeddings and can be specialized to 2D (Equation (5)) and 3D (Equation (10)) with the inclusion of area (respectively volume) terms to account for non-uniform weighting. Specifically, the individual terms in (6) match the terms in (5) and (10). We remark on the similarity between our Equations (5) and (10), and the partial differential operator that also minimizes an AKVF energy given in [TSB16].

Different from [SBCBG11a], for parameterization applications we choose to integrate the piecewise-constant KVF energy $\|J_{\mathbf{d}} + J_{\mathbf{d}}^\top\|^2_{\mathrm{Fro}}$ over the 3D surface rather than the 2D parameterization. The intuition comes from the fact that we are deforming a 3D surface into 2D in the parameterization. Small triangles in the parameterization will be given small weight if we integrate the KVF energy over the parameterization. These triangles, however, may correspond to large triangles in 3D and should affect the parameterization energy disproportionately as the energy is integrated over the 3D triangles in (1). To make this modification, we simply scale the quantities in (5) by $\frac{\hat{A}_f}{A_f}$. We found that such a change increases the performance of the preconditioner in practice.

4.2.2. Volumetric Case

We can extend the formula for $K(\mathbf{x})$ to tetrahedral meshes embedded in $\mathbb{R}^3$ as well, which is useful for optimization problems like the ones documented in §6.2. First-order FEM still suffices, and our vector fields are still defined on vertices. Now, the fields are interpolated piecewise-linearly from the vertices of a tetrahedral mesh to the interiors of the tetrahedra.

To simplify notation, we first provide a 3×3 off-diagonal block of the KVF operator for a single tetrahedron; the full KVF operator $K(\mathbf{x})$ for a tet is 12×12 (four vertices, each with xyz coordinates). Suppose a tet has vertices $(\mathbf{x}_0,\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3)$. As illustrated in Figure 4(b), define

$$\begin{gathered} \mathbf{a} := \mathbf{x}_1 - \mathbf{x}_3 \qquad \mathbf{b} := \mathbf{x}_3 - \mathbf{x}_2 \qquad \mathbf{c} := \mathbf{x}_2 - \mathbf{x}_0 \\ \mathbf{n}_1 := \mathbf{b}\times\mathbf{a} \qquad \mathbf{n}_2 := \mathbf{b}\times\mathbf{c} \end{gathered} \tag{8}$$

The block of $K(\mathbf{x})$ corresponding to edge $(\mathbf{x}_0,\mathbf{x}_1)$ is then

$$K_{\mathrm{edge}}(\mathbf{x}_0,\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3) := \frac{1}{36\mathrm{Vol}}[\mathbf{n}_2\mathbf{n}_1^\top + (\mathbf{n}_1\cdot\mathbf{n}_2)I_{3\times 3}], \tag{9}$$

where Vol denotes the (unsigned) volume of the tetrahedron.

With this notation in place, we provide a formula for assembling the volumetric version of $K(\mathbf{x})$:

Proposition 3 For a tetrahedral mesh, $K(\mathbf{x})$ can be computed in 3×3 blocks corresponding to (x,y,z) coordinates as follows:

$$K_{pq} = \begin{cases} \sum_{(p,q,r,s)\in T} K_{\text{edge}}(\mathbf{x}_p,\mathbf{x}_q,\mathbf{x}_r,\mathbf{x}_s) & \text{if } p\sim q \\ -\sum_{r\sim p} K_{pr} & \text{if } p=q \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Here, $p\sim q$ indicates that $p,q\in V$ are adjacent, and T is the set of tetrahedra.

4.3. Preconditioned Gradient

At this point we have developed a Riemannian-style approach to parameterization. The "manifold of parameterizations" $\mathcal{M}$ is isomorphic to the set of $\mathbf{x}\in\mathbb{R}^{2|V|}$, modulo rigid motion. The tangent space is the set of vector fields $\mathbf{d}\in\mathbb{R}^{2|V|}$ endowed with the inner product $\langle \mathbf{d},\mathbf{d}'\rangle_{\mathbf{x}} := \mathbf{d}^\top K(\mathbf{x})\mathbf{d}'$.

Parameterization can be viewed as searching for $\mathbf{x}$ that minimizes a distortion function $E(\mathbf{x})$. To properly carry out gradient descent on this manifold, we search along the *Riemannian gradient* with respect to $\langle\cdot,\cdot\rangle_{\mathbf{x}}$, defined as

$$\nabla_{\text{KVF}} E(\mathbf{x}) := K(\mathbf{x})^+ \nabla_{\mathbf{x}} E(\mathbf{x}). \tag{11}$$

Here, $\nabla_{\mathbf{x}}$ denotes the Euclidean (coordinate-wise) gradient typically used for optimization, and $K(\mathbf{x})^+$ is the Moore–Penrose pseudoinverse of $K(\mathbf{x})$. Here we require the pseudoinverse because $K(\mathbf{x})$ has a three-dimensional null space corresponding to x translation, y translation, and rotation about the origin. Since $K(\mathbf{x})$ is otherwise full rank, applying this pseudoinverse can be accomplished using standard sparse matrix factorization techniques.

5. Algorithm

Our approach for minimizing equation (1) uses the AKVF operator to formulate an efficient and easily-implemented algorithm. Our proposed method, shown in Algorithm 1, starts from a feasible initialization $\mathbf{x}^0$ and iteratively produces a sequence of approximations $\mathbf{x}^k$ that are guaranteed to converge to a local minimum. We describe each step of our algorithm in more detail below.

5.1. Iterative technique

Similar to [RPPSH17, KGL16], we solve a Newton-style system using the AKVF operator as a proxy for the Hessian of our problem. Specifically, if $\mathbf{x}^k$ is the parameterization at iteration k, we generate $\mathbf{x}^{k+1}$ satisfying

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha K^+ \nabla_{\mathbf{x}} E(\mathbf{x}^k).$$

It is important to note that the positive (semi-)definiteness of K ensures that $-K^+\nabla_{\mathbf{x}}E(\mathbf{x}^k)$ will be a descent direction.

To avoid ill-conditioned gradients, we project out the three-dimensional null space of K at each iteration. Since our preconditioner is the Killing operator, the null space is known *a priori* as two constant vectors corresponding to translation of the entire parameterization in the x or y axes, and a rotation vector about the origin in $\mathbb{R}^{2|V|}$ given by $(y_1,\ldots,y_n,-x_1,\ldots,-x_n)$.

Algorithm 1: AKVF Parameterization

Input : A mesh M with vertices V and faces F.
Output: A parameterization $\mathbf{x}$ minimizing the Symmetric Dirichlet energy.

$\mathbf{x}^0$ = Tutte(V, F)
Perform flap optimization
while $\|\nabla_{\mathbf{x}}E(\mathbf{x}^k)\| > \varepsilon$ **do**
  Compute AKVF operator K
  Compute $\nabla_{\mathbf{x}}E(\mathbf{x}^k)$
  Let $\mathbf{d}^k = -K^+\nabla_{\mathbf{x}}E(\mathbf{x}^k)$
  Perform backtracking line search to find a step size α
  $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha\mathbf{d}^k$
end

5.2. Computing the Search Direction

To find the Riemannian gradient $\nabla_{\text{KVF}}E(\mathbf{x}) := K(\mathbf{x})^+\nabla_{\mathbf{x}}E(\mathbf{x})$, we solve a linear system for the search direction $\mathbf{d}$:

$$K(\mathbf{x})\mathbf{d} = \nabla_{\mathbf{x}}E(\mathbf{x}). \tag{12}$$

Instead of naïvely solving (12), we leverage our knowledge of K to improve performance. First, because K is positive (semi-)definite, we can compute the pseudoinverse of K efficiently using algorithms tailored for positive (semi-)definite matrices. In our implementation, we use PARDISO [PSLG14, PSA14] to leverage recent advances in multi-core parallel solvers. Second, while our operator changes numerically from iteration to iteration, its sparsity pattern is constant. This allows us to symbolically factorize the matrix only once, significantly reducing computation time.

5.3. Line Search Parameters

At each step, we iterate $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha\mathbf{d}^k$, where α is a step size, and $\mathbf{d}^k$ is the search direction from §5.2. Ideally, we want an α that does not cause any of the triangles in our parameterization to flip orientation. Following [SS15], we define a maximal step size $\alpha_{\max}$ and perform backtracking line search for the optimal α within $[0, 0.8\alpha_{\max}]$. Specifically, we iteratively scale $\alpha^k := \rho\alpha^{k-1}$ with $\rho = 0.9$ and $\alpha^0 = 0.8\alpha_{\max}$ until we satisfy the first Wolfe condition

$$E(\mathbf{x}^k + \alpha^k\mathbf{d}^k) \le E(\mathbf{x}^k) + c_1\alpha^k\nabla E(\mathbf{x}^k)^T\mathbf{d}^k,$$

with $c_1 = 10^{-5}$.

5.4. Flap Optimization

In many cases there may be vertices along the boundary of the parameterized surface that are adjacent to only one triangle. We call the triangles that contain such vertices *flap triangles*. While the error functions we minimize are nonlinear, flap triangles have a closed-form solution for error functions that measure isometric distortion. Let $x_0,x_1,x_2\in\mathbb{R}^2$ be vertices of a flap triangle where x_0 is contained by one triangle, which we call the *flap vertex*. Furthermore, let $\hat{x}_0,\hat{x}_1,\hat{x}_2\in\mathbb{R}^3$ be the corresponding locations of the points on the 3D surface.

Figure 5: *The UV map obtained after running SLIM on the mesh in Figure 1 for 500 iterations.*

Assume the x_1, x_2 vertices are constrained. In this case, the Jacobian $J(x)$ that minimizes an isometric distortion will have singular vectors that aligns with the constrained edge $\overrightarrow{x_1x_2}$ and its orthogonal direction. The singular value corresponding to the constrained edge will simply be $\frac{|x_1-x_2|}{|\hat{x}_1-\hat{x}_2|}$. The remaining singular value will have a minimum of 1 for isometric distortion measures.

Using this information, we can derive the optimal location of x_0 given x_1, x_2. Let $\beta = \frac{(\hat{x}_0-\hat{x}_1)\cdot(\hat{x}_2-\hat{x}_1)}{|\hat{x}_2-\hat{x}_1|^2}$ and $\gamma = |\hat{x}_0 - \hat{x}_1 - \beta(\hat{x}_2-\hat{x}_1)|$. Then the optimal location of x_0 is

$$x_0 = (1-\beta)x_1 + \beta x_2 + \frac{\gamma}{|x_2-x_1|}(x_2-x_1)^\perp,$$

where $(x_2-x_1)^\perp$ represents the rotation of the vector x_2-x_1 by 90 degrees in the plane.

While there are typically not many flap triangles in a parameterization, we can identify them at the beginning of the optimization. We apply this flap optimization after the initial Tutte embedding to the unit disk to move each flap vertex to the optimal position for the current parameterization. Experiments show that this simple trick reduces the number of iterations necessary to converge by 13% on average.

5.5. Convergence Criteria

Note that simply using the value of the function as a stopping threshold is not sufficient to determine convergence. This phenomenon is apparent from Figure 1 and its error graph in Figure 9. While the error graphs of SLIM and AKVF appear similar after 10 iterations, Figure 1 demonstrates the actual parameterizations are far from each other, which indicates the need for a more sophisticated stopping criterion.

In §6, we use several convergence criteria. For our purposes, we use the first order criterion $\|\nabla_{\mathbf{x}} E(\mathbf{x})\| \leq 10^{-5}$. However, in our comparisons we have observed that competing methods falter when close to the minimum, and thus we relax our criteria to either a fixed number of iterates, or a weaker first order condition $\|\nabla_{\mathbf{x}} E(\mathbf{x})\| \leq 10^{-3}$ when comparing with other methods.

6. Results

In this section we illustrate the utility of our preconditioner on a variety of tasks where geometry-aware search directions are crucial to obtain fast and effective solutions. We first showcase our approach on 2D mesh parameterization, where we achieve state-of-the-art performance on a variety of tasks ranging both in mesh and geometric complexity, and then provide examples of AKVF-based preconditioning for problems on tetrahedral meshes.

All methods are run on an i7-6700K at 4GHz running Ubuntu 14.04. All methods are implemented in C++11. We utilize all four cores when solving the sparse linear system in (12) using the PARDISO solver. For a fair comparison with [RPPSH17] and [KGL16], our method was implemented in the SLIM framework provided by the authors and has only minor changes apart from the preconditioner. We have reimplemented AQP in C++ within the SLIM framework, as the original MATLAB version is significantly slower. The results for AQP are accelerated using the acceleration constant defined in [KGL16].

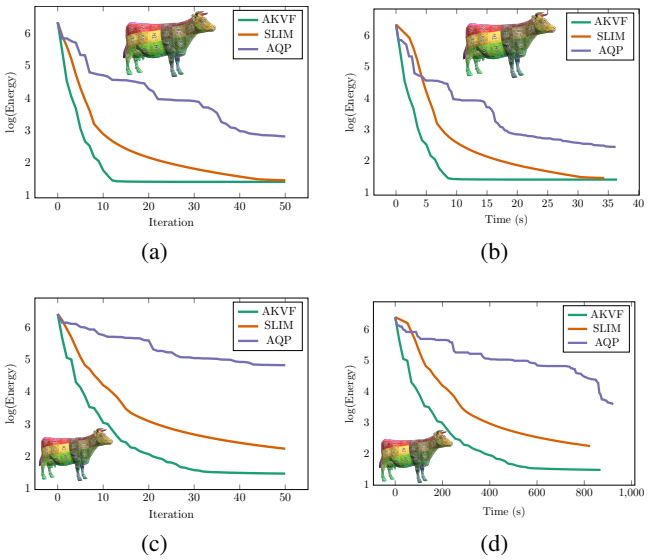


Figure 6: *Comparison with SLIM and AQP for larger meshes. The mesh in (a) and (b) has 550K triangles, and the one in (c) and (d) has 8.8M triangles. We observe improved performance for our KVF preconditioner with increased mesh complexity both with respect to the number of iterations and overall time to convergence.*

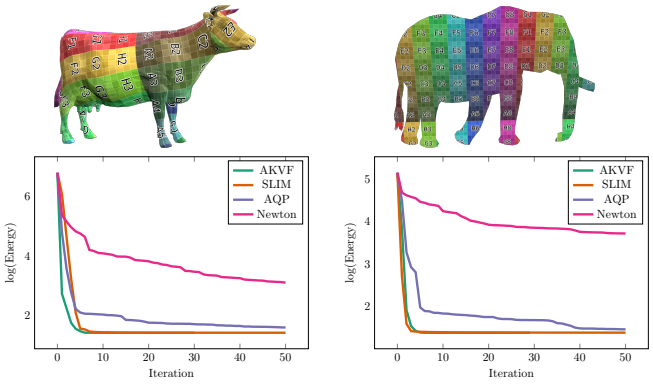


Figure 7: *Comparison with Newton's method on a few meshes. As observed in [RPPSH17] and [KGL16], Newton's method has poorer performance for mesh parameterization.*

6.1. 2D Parameterization

In what follows, we use the energy in (2) as our objective. Our method can, however, be applied generally to any isometric distortion energy (Figure 3).

Figure 9 showcases our method on nine meshes. The first column shows the parameterization obtained by our algorithm displayed as a UV color grid on the input mesh; the second column displays the 2D UV map; the third column plots objective value as a function of iterations; and the fourth column plots objective value as a function of time. Our plots compare performance to state-of-the-art algorithms for mesh parameterizations applied to the same nonlinear objective function. In most cases, we achieve parameterizations that approach or reach the lower bound deformation energy within fewer than 100 iterations. This yields state-of-the-art performance that is comparable to two currently available methods: Scalable Locally-Injective Maps (SLIM) [RPPSH17] and Accelerated Quadratic Proxy (AQP) [KGL16]. In addition, as highlighted in Table 1, we always show faster convergence with respect to a first order convergence condition than SLIM, which showcases the effectiveness of the search directions recovered by our preconditioner even in the region of the minimum.

We stress additionally that our method is significantly easier to implement than SLIM as it requires only a simple, Laplacian-like matrix construction (§4.2.1) that can be plugged into any standard gradient descent algorithm.

While there are two meshes on which our method is somewhat outperformed by SLIM (the camel head and hand in Figure 9), we note that the gradient vector field of those parameterizations will always be far from a rigid motion due to poor seam choice. Thus, the parameterizations obtained by all algorithms have high distortion error caused primarily by the vertices in the center of the initial parameterization.

One particular advantage of our method is its performance when the objective value is *nearly* minimal; in this case, reaching the full minimum can still require large-scale motion of the parameterization to pivot about a few deformed triangles.

As illustrated in Figure 1, our method outperforms SLIM on a difficult test example with this property. We show our method on the top row and SLIM on the bottom row. The first column shows the initial parameterization. We use the same starting parameterization as SLIM in this example to highlight differences. The following columns show the parameterization at iterations 5, 10, and 20. Our method and SLIM exhibit similar performance within the first few iterations, achieving a large reduction in distortion error. However, we recover better search directions in the following iterations and converge to the lower-bound energy value of 4. We continue running SLIM until we reach 500 iterations (Figure 5), and while more iterations improves the parameterization, it still has larger error than our parameterization after 10 iterations.

As we rely on a positive (semi-)definite preconditioner in a similar vein to a quasi-Newton method, we can guarantee convergence to a local minimum. Table 1 gives convergence times under a first order convergence criterion given by $\|\nabla_{\mathbf{x}} E(\mathbf{x})\| < 10^{-3}$. Notice the discrepancy between the results reported in Table 1 and those given in the performance plots of Figure 9. In our experiments, we have frequently observed that beyond running 30 iterations, further improvements are marginal and are not visible qualitatively.

In Figure 6, we compare with SLIM and AQP on significantly larger meshes. Our method outperforms both SLIM and AQP on larger meshes due to the simplicity of Equation (6) when compared with SLIM, which has to solve 2×2 SVDs on each triangle of the mesh. While our implementation of SLIM performs all SVDs in parallel, the timing differences are still significant.

In Figure 7, we show a comparison with Newton's method on two meshes on which we observed good performance for Newton's method. This method can get stuck if the Hessian is not positive semidefinite at a given iteration. We have chosen not to regularize by projecting on the space of PSD matrices. As observed in [RPPSH17], regularization does not significantly improve the results. Confirming results in [RPPSH17, KGL16], our method vastly outperforms Newton's method on parameterization tasks.

6.2. 3D Deformation

AKVF-based preconditioning can be applied to volumetric mesh deformation in a similar fashion to 2D parameterization. Here the deformation energy measures the distortion from some tetrahedral rest shape to its deformed shape, typically with point constraints. We can use our operator defined in (10) to modify the search direction of the gradient of this volumetric error function as well.

We compare the performance of this volumetric approach on the cube deformation example from [RPPSH17] with the same source model and constraints. In this experiment, we minimize the exponential Dirichlet energy augmented with a least-squares term $w\sum_{\mathbf{x}}(\mathbf{x}_i - p_i)^2$. To make the comparison fair, identically to SLIM we use Tikhonov regularization to precondition using the matrix $K(\mathbf{x}) + cI$; we reuse parameter values from their experiment ($w = 10^5, c = 10^{-4}$) as well as the same line search procedure. Again to match the SLIM implementation, we invert our preconditioner using the conjugate gradient (CG) method; we find that a 1% tolerance is acceptable for CG convergence. Figure 8 demonstrates the results on the experiment, in which we ran 100 iterations of each method. The top row shows that the deformation results are qualitatively similar, while the bottom row demonstrates that performance of the methods is comparable. In particular, our method and AQP take similar amounts of time, while SLIM is slightly slower.

7. Discussion and Conclusion

Our results and theoretical development demonstrate that KVFs find value not just in shape analysis but also for identifying structure in otherwise challenging optimization problems. As verified in the experiments above, the KVF operator serves as a versatile preconditioner agnostic to the details of a chosen objective function. This makes the KVF operator a reasonable "black-box" choice for accelerating first-order optimization techniques involving vertex positions, reducing the need to compute a potentially dense or computationally expensive Hessian.

We foresee many additional applications of our machinery outside the ones tested in §6. For instance, physical simulation software can require solution of complex variational problems for sta-

Model	#Faces	#Vertices	SLIM (s)	AKVF (s)
Camel	3576	2032	10.76	0.127
Cow	5804	3195	9.296	0.207
Elephant	1796	1105	1.593	0.036
Hilbert Curve	9174	6120	8.319	0.176
Horse	39698	20636	22.92	1.560
Tricera	5660	3163	14.88	0.319
Sine Wave	4406	2339	0.470	0.065
Camel Head	22704	11381	13.97	3.052
Hand	4677	2356	34.34	13.05

Table 1: *Time to reach convergence criterion* $\|\nabla_{\mathbf{x}} E(\mathbf{x})\| < 10^{-3}$. *We observe good performance close to the minimum using our algorithm, and thus significantly reduced convergence time. We achieve equal or lower distortion energy than SLIM in all examples.*

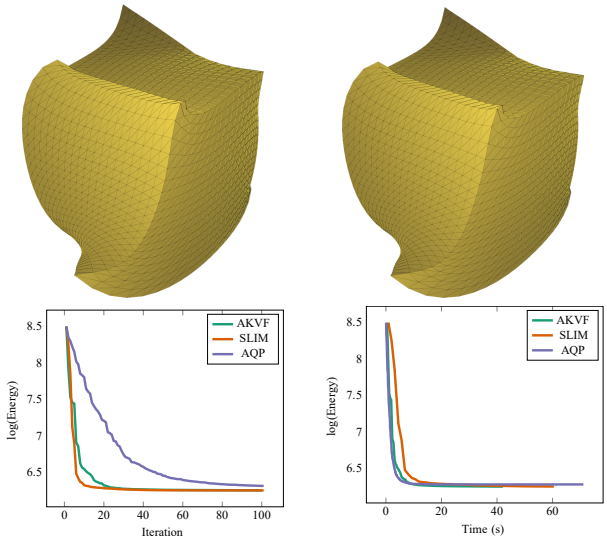


Figure 8: *Comparison of our method against SLIM on a tetrahedral deformation test case from their paper. Top row: deformation results, SLIM (left) and ours (right).*

ble implicit time-stepping [GSS*15]. Automatic tools for interpolating between frames of an animated mesh sequence similarly must minimize objectives constructed out of nonconvex elastic terms and could benefit from preconditioned acceleration [FB11, HRWW12, HRS*16].

A drawback of the preconditioners we have explored in this paper is that they are defined for codimension-zero meshes, e.g. triangle meshes in the plane or tetrahedral meshes in space. While KVFs are most often described in intrinsic language, it may be that similar differential equations seeking infinitesimal isometries of polyhedra or shell surfaces, e.g. the one considered in [HWAG09], can be used to extend our preconditioning technique to triangle meshes in $\mathbb{R}^3$ and other important cases for geometry processing.

The description of our technique in Riemannian language in §4.3 suggests potential avenues for improved optimization. In particular, while our search direction is chosen as the Riemannian descent direction, the subsequent line search is not along a geodesic with respect to the KVF metric. Such a geodesic is likely complicated (and related e.g. to geodesics in the space of elastic shells [RW13]), but it may be possible to extend line search to search along circles or curved arcs like the logarithmic spirals in [SBCBG11a]. The challenge here will be to approximate geodesic curves in the space of parameterizations with the KVF metric while staying in a space simple enough for efficiency purposes.

Specifically for mesh parameterization, a critical next-step will be automatic computation of seams dividing the surface into disk patches to be parameterized. This challenging problem involves minimizing distortion of the mapped patches, choosing an appropriate topology for the set of cut curves, and consideration of artistic constraints involving the visibility of the seams on the surface as it is rendered.

Even without these improvements, KVF-based preconditioning stands as a simple way to improve the efficiency of numerical techniques in geometry processing. We anticipate this preconditioner will serve as a tool in the rapidly-developing numerical toolbox for shape-based optimization problems.

Acknowledgments This work was supported in part by NSF CAREER award IIS 1148976. J. Solomon acknowledges funding from an MIT Skoltech Seed Fund grant ("Boundary Element Methods for Shape Analysis") and from the MIT Research Support Committee ("Structured Optimization for Geometric Problems"), as well as Army Research Office grant W911NF-12-R-0011 ("Smooth Modeling of Flows on Graphs").

References

[ABCCO13] AZENCOT O., BEN-CHEN M., CHAZAL F., OVSJANIKOV M.: An operator approach to tangent vector field processing. In *Computer Graphics Forum* (2013), vol. 32, pp. 73–82. 3

[AOCBC15] AZENCOT O., OVSJANIKOV M., CHAZAL F., BEN-CHEN M.: Discrete derivatives of vector fields on surfaces—An operator approach. *ACM Trans. Graph. 34*, 3 (2015), 29. 3

[APL14] AIGERMAN N., PORANNE R., LIPMAN Y.: Lifted bijections for low distortion surface mappings. *ACM Trans. Graph. 33*, 4 (2014), 69:1–69:12. 2, 4

[BB11] BAUER M., BRUVERIS M.: A new Riemannian setting for surface registration. In *MICCAI Workshop on Mathematical Foundations of Computational Anatomy* (2011), pp. 182–193. 3

[BCBSG10] BEN-CHEN M., BUTSCHER A., SOLOMON J., GUIBAS L.: On discrete Killing vector fields and patterns on surfaces. In *Computer Graphics Forum* (2010), vol. 29, pp. 1701–1711. 2, 3, 4

[CLW16] CHIEN E., LEVI Z., WEBER O.: Bounded distortion parametrization in the space of metrics. *ACM Trans. Graph. 35*, 6 (2016), 215:1–215:16. 2

[CW13] CHAN K. Y., WAN J. W.: Reconstruction of missing cells by a Killing energy minimizing nonrigid image registration. In *International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (2013), pp. 3000–3003. 3

[dGDT15] DE GOES F., DESBRUN M., TONG Y.: Vector field processing on triangle meshes. In *SIGGRAPH Asia 2015 Courses* (2015), p. 17. 3, 4

[dGLB*14] DE GOES F., LIU B., BUDNINSKIY M., TONG Y., DESBRUN M.: Discrete 2-tensor fields on triangulations. In *Computer Graphics Forum* (2014), vol. 33, pp. 13–24. 3

[DGM98] DUPUIS P., GRENANDER U., MILLER M. I.: Variational problems on flows of diffeomorphisms for image matching. *Quarterly of Applied Mathematics* (1998), 587–600. 3

[DMK03] DEGENER P., MESETH J., KLEIN R.: An adaptable surface parameterization method. In *Proceedings of the 12th International Meshing Roundtable* (2003), pp. 201–213. 2

[FB11] FRÖHLICH S., BOTSCH M.: Example-driven deformations based on discrete shells. In *Computer Graphics Forum* (2011), vol. 30, pp. 2246–2257. 9

[FH05] FLOATER M. S., HORMANN K.: *Surface Parameterization: a Tutorial and Survey*. Springer Berlin Heidelberg, 2005, pp. 157–186. 2

[FL16] FU X.-M., LIU Y.: Computing inversion-free mappings by simplex assembly. *ACM Trans. Graph. 35*, 6 (2016), 216:1–216:12. 2

[FLG15] FU X.-M., LIU Y., GUO B.: Computing locally injective mappings by advanced MIPS. *ACM Trans. Graph. 34*, 4 (2015), 71:1–71:12. 2

[Flo97] FLOATER M. S.: Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design 14*, 3 (1997), 231 – 250. 2

[FW06] FLETCHER T., WHITAKER R.: Riemannian metrics on the space of solid shapes. In *MICCAI Workshop on Mathematical Foundations of Computational Anatomy* (2006), pp. 47–57. 3

[GSS*15] GAST T. F., SCHROEDER C., STOMAKHIN A., JIANG C., TERAN J. M.: Optimization integrator for large time steps. *TVCG 21*, 10 (2015), 1103–1115. 9

[HG00] HORMANN K., GREINER G.: MIPS: An efficient global parameterization method. In *Curve and Surface Design: Saint-Malo 1999* (2000), pp. 153–162. 2

[HRS*16] HEEREN B., RUMPF M., SCHRÖDER P., WARDETZKY M., WIRTH B.: Splines in the space of shells. In *Computer Graphics Forum* (2016), vol. 35, pp. 111–120. 9

[HRWW12] HEEREN B., RUMPF M., WARDETZKY M., WIRTH B.: Time-discrete geodesics in the space of shells. In *Computer Graphics Forum* (2012), vol. 31, pp. 1755–1764. 3, 9

[HWAG09] HUANG Q.-X., WICKE M., ADAMS B., GUIBAS L.: Shape decomposition using modal analysis. In *Computer Graphics Forum* (2009), vol. 28, pp. 407–416. 9

[KGL16] KOVALSKY S. Z., GALUN M., LIPMAN Y.: Accelerated quadratic proxy for geometric optimization. *ACM Trans. Graph. 35*, 4 (July 2016), 134:1–134:11. 2, 6, 7, 8, 11

[KKDS10] KURTEK S., KLASSEN E., DING Z., SRIVASTAVA A.: A novel Riemannian framework for shape analysis of 3d objects. In *Proc. CVPR* (2010), IEEE, pp. 1625–1632. 3

[KKG*12] KURTEK S., KLASSEN E., GORE J. C., DING Z., SRIVASTAVA A.: Elastic geodesic paths in shape space of parameterized surfaces. *Proc. PAMI 34*, 9 (2012), 1717–1730. 3

[KMP07] KILIAN M., MITRA N. J., POTTMANN H.: Geometric modeling in shape space. In *ACM Trans. Graph.* (2007), vol. 26, ACM, p. 64. 3

[LPRM02] LÉVY B., PETITJEAN S., RAY N., MAILLOT J.: Least squares conformal maps for automatic texture atlas generation. In *Proceedings of SIGGRAPH* (2002), pp. 362–371. 2

[LZX*08a] LIU L., ZHANG L., XU Y., GOTSMAN C., GORTLER S. J.: A local/global approach to mesh parameterization. In *Proceedings of the Symposium on Geometry Processing* (2008), pp. 1495–1504. 2

[LZX*08b] LIU L., ZHANG L., XU Y., GOTSMAN C., GORTLER S. J.: A local/global approach to mesh parameterization. In *Computer Graphics Forum* (2008), vol. 27, pp. 1495–1504. 3, 4

[MJBC13] MARTIN T., JOSHI P., BERGOU M., CARR N.: Efficient non-linear optimization via multi-scale gradient filtering. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 89–100. 3

[MM04] MICHOR P. W., MUMFORD D.: Riemannian geometries on spaces of plane curves. In *J. Eur. Math. Soc.* (2004). 3

[Neu85] NEUBERGER J.: Steepest descent and differential equations. *Journal of the Mathematical Society of Japan 37*, 2 (1985), 187–195.

[NVW12] NITSCHKE I., VOIGT A., WENSCH J.: A finite element approach to incompressible two-phase flow on manifolds. *Journal of Fluid Mechanics 708* (2012), 418–438. 3

[NW06] NOCEDAL J., WRIGHT S.: *Numerical Optimization*. Springer, 2006.

[OBCS*12] OVSJANIKOV M., BEN-CHEN M., SOLOMON J., BUTSCHER A., GUIBAS L.: Functional maps: a flexible representation of maps between shapes. *ACM Trans. Graph. 31*, 4 (2012), 30. 3

[PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics 2* (1993), 15–36. 2

[PSA14] PETRA C. G., SCHENK O., ANITESCU M.: Real-time stochastic optimization of complex energy systems on high-performance computers. *Computing in Science & Engineering 16*, 5 (2014), 32–42. 6

[PSLG14] PETRA C. G., SCHENK O., LUBIN M., GÄRTNER K.: An augmented incomplete factorization approach for computing the Schur complement in stochastic optimization. *SIAM J. Sci. Comp. 36*, 2 (2014), C139–C162. 6

[RN95] RENKA R. J., NEUBERGER J.: Minimal surfaces and Sobolev gradients. *SIAM J. Sci. Comp. 16*, 6 (1995), 1412–1427. 3

[RPPSH17] RABINOVICH M., PORANNE R., PANOZZO D., SORKINE-HORNUNG O.: Scalable locally injective mappings. *ACM Trans. Graph. 36*, 2 (2017), 16. 1, 2, 3, 6, 7, 8, 11

[RW13] RUMPF M., WIRTH B.: Discrete geodesic calculus in shape space and applications in the space of viscous fluidic objects. *Journal on Imaging Sciences 6*, 4 (2013), 2581–2602. 9

[SAPH04] SCHREINER J., ASIRVATHAM A., PRAUN E., HOPPE H.: Inter-surface mapping. In *ACM Trans. Graph.* (2004), vol. 23, ACM, pp. 870–877. 3

[SBCBG11a] SOLOMON J., BEN-CHEN M., BUTSCHER A., GUIBAS L.: As-Killing-as-possible vector fields for planar deformation. In *Computer Graphics Forum* (2011), vol. 30, pp. 1543–1552. 2, 3, 4, 5, 9

[SBCBG11b] SOLOMON J., BEN-CHEN M., BUTSCHER A., GUIBAS L.: Discovery of intrinsic primitives on triangle meshes. In *Computer Graphics Forum* (2011), vol. 30, pp. 365–374. 3

[SCOGL02] SORKINE O., COHEN-OR D., GOLDENTHAL R., LISCHINSKI D.: Bounded-distortion piecewise mesh parameterization. In *Proc. Visualization* (2002), pp. 355–362. 2

[SLMB05] SHEFFER A., LÉVY B., MOGILNITSKY M., BOGOMYAKOV A.: ABF++: Fast and robust angle based flattening. *ACM Trans. Graph. 24*, 2 (2005), 311–330. 2

[SPR06] SHEFFER A., PRAUN E., ROSE K.: Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis. 2*, 2 (2006), 105–171. 2

[SS15] SMITH J., SCHAEFER S.: Bijective parameterization with free boundaries. *ACM Trans. Graph. 34*, 4 (2015), 70. 2, 3, 4, 6

[TSB16] TAO M., SOLOMON J., BUTSCHER A.: Near-isometric level set tracking. In *Computer Graphics Forum* (2016), vol. 35, pp. 65–77. 3, 5

[Tut63] TUTTE W. T.: How to draw a graph. *Proc Lond Math Soc 13* (1963), 743–767. 2

[VCD*16] VAXMAN A., CAMPEN M., DIAMANTI O., PANOZZO D., BOMMES D., HILDEBRANDT K., BEN-CHEN M.: Directional field synthesis, design, and processing. In *Computer Graphics Forum* (2016), vol. 35, pp. 545–572. 3

[WGTY04] WANG Y., GU X., THOMPSON P. M., YAU S.-T.: 3d harmonic mapping and tetrahedral meshing of brain imaging data. *Proc. MICCAI* (2004).

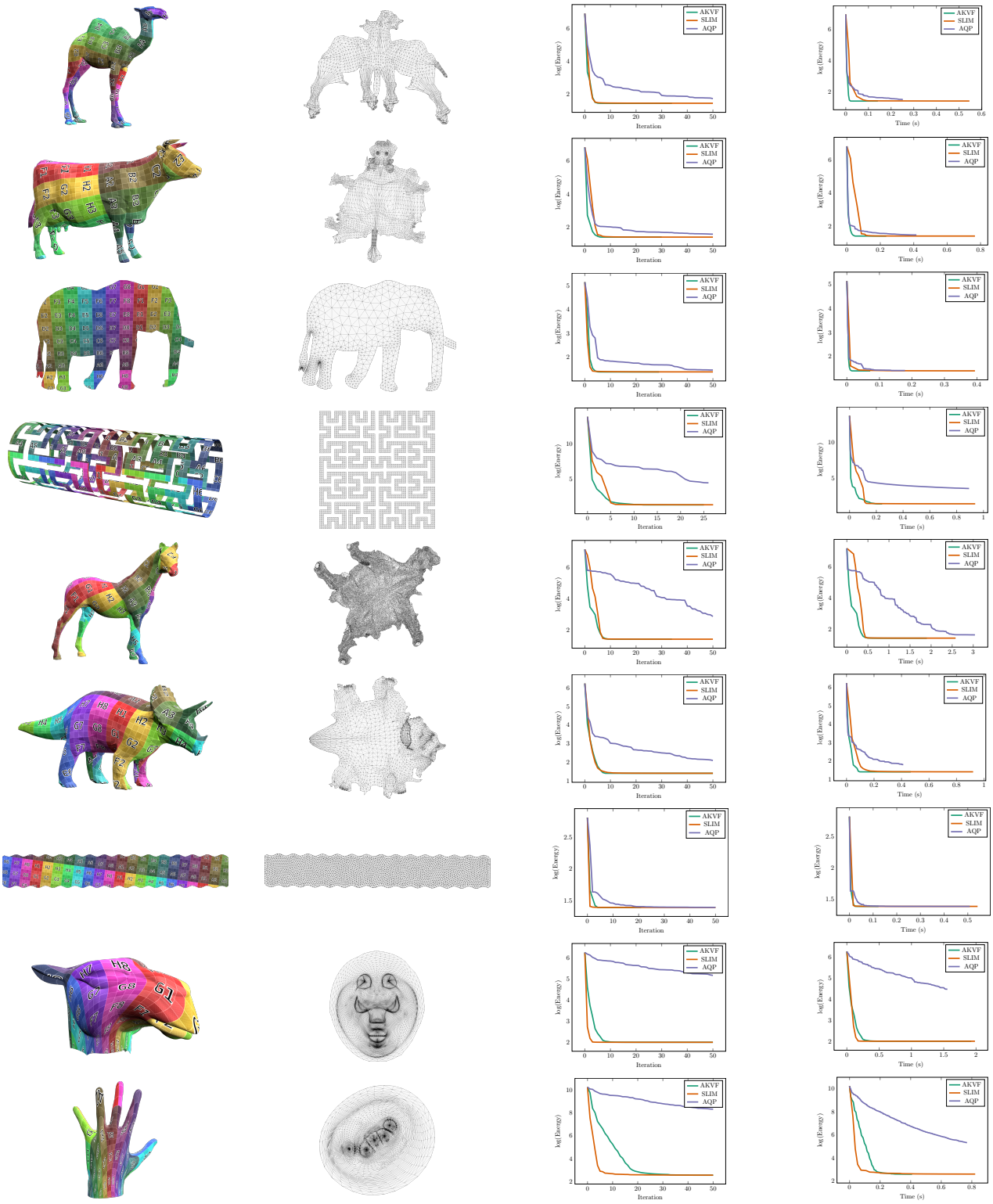


Figure 9: *Summary of results on several test meshes using the symmetric Dirichlet energy. From left to right: a color grid showing the parameterization recovered by our algorithm on the original 3D mesh; the 2D UV map of the parametrization; energy plotted against iterations for our algorithm, SLIM [RPPSH17] and AQP [KGL16]; energy plotted against time for our algorithm, SLIM and AQP. We note that the two meshes on which our method is outperformed by SLIM (the camel head and hand) are difficult to parameterize due to poor seam placement.*