DOI: 10.1111/cgf.70341

EUROGRAPHICS 2026 / B. Masia and J. Thies
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 2

Fast Injective Mesh Parameterization via Beltrami Coefficient Prolongation

G. Fargion and O. Weber

Bar-Ilan University, Israel

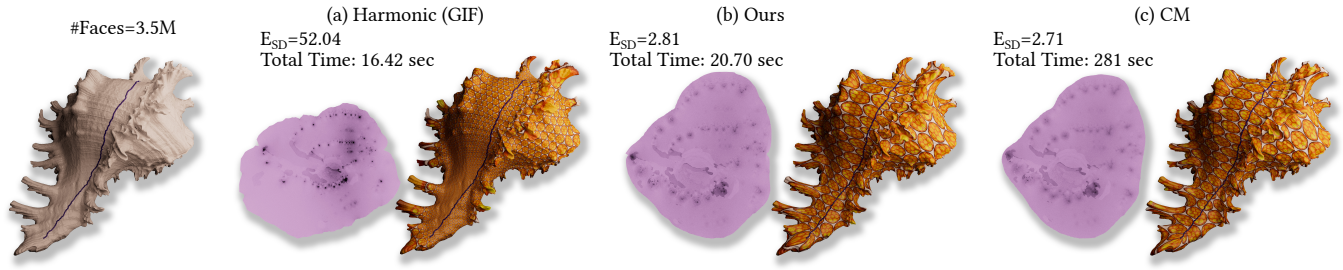


Figure 1: *Comparison on a high-resolution model of a scanned ornamented shell with 3.5M triangles containing numerous tubercles and ridges. Harmonic approaches are particularly effective at producing injective parameterizations. (a) However, due to the accumulated Gaussian curvature that appears in the shell's features, the harmonic GIF method of [FW22] fails to control the isometric distortion on this challenging example (left). (b) Our method guarantees injectivity by construction (middle). It terminates in a fraction of the time required by CM [SPSH*17], while achieving a symmetric Dirichlet energy close to that of CM. (c) The optimal map obtained by CM (on the right) has the lowest level of distortion, but it takes more than 13.5 times longer to compute than our method.*

Abstract

*We present a highly efficient and robust method for free boundary injective parameterization of disk-like triangle meshes with low isometric distortion. Harmonic function–based approaches, grounded in a strong mathematical framework, are widely employed. In particular, harmonic maps are valuable for guaranteeing injectivity under suitable boundary conditions. They are also computationally efficient, as they operate within a linear subspace [FW22]. However, this restriction to a limited subspace often introduces substantial isometric distortion, especially on highly curved surfaces. In contrast, methods that explore the full space of piecewise linear maps [SPSH*17] achieve much lower isometric distortion, but at the expense of increased computational cost. We propose a hybrid method that combines the speed and robustness of harmonic maps with the generality of full-space methods to produce injective maps with low isometric distortion up to 50 times faster than state-of-the-art methods. The core concept is simple but powerful. Instead of searching for the optimal parameterization over the original mesh via a full-space method, we simplify the input fine mesh and parameterize the coarse mesh. We then prolong the Beltrami coefficients from the coarse mesh parameterization to the fine mesh, resulting in a customized Laplacian matrix that gives rise to a harmonic map in a modified metric [WGS23, FW25]. Our method ensures injectivity, offers great computational efficiency, and produces significantly lower isometric distortion compared to harmonic maps.*

CCS Concepts

• ***Computing methodologies*** → ***Mesh geometry models;***

1. Introduction

Computing high-quality surface parameterizations of triangle meshes efficiently is a fundamental problem in geometry processing and computer graphics. These maps are essential for texture mapping, shape correspondence, remeshing, shape analysis, compression, shape-and-image deformation, and high-level learning tasks, to name a few. Despite an extensive body of literature on the subject, computational mapping, and surface parameterization in particular, remains highly active. This is primarily because the underlying mathematical formulations lead to large-scale, nonlinear, and nonconvex optimization problems.

Over the past decade, significant progress has been made in developing fast, high-quality parameterization methods that guarantee injectivity. Most of these approaches operate in the space of (continuous) piecewise linear (P.W.L.) mappings, typically starting with a valid but highly distorted map initialization (e.g., [Tut63]), followed by an iterative optimization process that gradually reduces the geometric distortion across the mesh. Methods such as [SPSH*17, RPPSH17, JSP17, SYLF20] often achieve the lowest overall distortion. Processing meshes of low to medium size can be done in a matter of seconds. However, optimizing models with several million triangles can consume substantial computational resources and take several minutes to process, even with high-end hardware.

Another approach to solving nonlinear optimization problems involves restricting the solution space to a reduced-dimensional linear subspace. For mesh parameterization and deformation tasks, the harmonic subspace is of particular interest since it possesses desirable properties such as smoothness and injectivity guarantees in both the discrete and smooth cases [FW22, HFCW19, LCH*21, CW15].

This restriction significantly reduces the dimensionality of the problem, as the degrees of freedom are effectively limited to the UV coordinates of a small set of boundary sampling points. Compared to full-space methods, harmonic-subspace approaches are substantially faster, often by an order of magnitude [FW22, HFCW19], while still guaranteeing injectivity. These methods typically yield high-quality parameterizations for meshes that do not require high isometric distortion for flattening [SSC18, MZ13]. However, a key limitation arises in regions with accumulated Gaussian curvature. Harmonic and conformal methods tend to introduce unnecessary excessive area distortion instead of tolerating a moderate amount of shear [FW22, Figure 10].

Several recent methods for computing maps [WGS23, FW25, FBC25] are based on the observation that any map from an abstract 2-manifold to the plane is harmonic with respect to some metric. Notably, even the optimal solution produced by a full-space method such as the Composite Majorization (CM) method [SPSH*17] is harmonic under a particular metric. Owing to the linear reproduction property of barycentric coordinates, constructing the cotan Laplacian using the edge lengths of the CM parameterization (a flat metric with low distortion) yields a harmonic subspace that contains this parameterization.

With such a Laplacian, the CM solution becomes accessible to harmonic-subspace methods such as the Globally Injective Flattening (GIF) method [FW22] or the conformal Boundary First Flattening (BFF) method [SC17]. Since CM optimizes over the full space of P.W.L. maps, its result can be regarded as the optimal solution. Thus, constructing a Laplacian based on the CM layout is sufficient for harmonic-subspace methods to reproduce the CM solution. This observation is key across multiple approaches: approximating this Laplacian efficiently and then applying a harmonic-subspace method can achieve results comparable to CM at only a fraction of CM's runtime. This approach is not limited to a particular type of distortion measure or optimization method. The exact same idea can be used to accelerate the solution of multiple energy-optimization problems.

Fargion et al. [FW25] recently used this approach to solve the surface parameterization problem. They trained a neural network to predict the pullback metric tensor of each triangle using a dataset of 30,000 models parameterized with CM, minimizing the so-called symmetric Dirichlet energy. During inference, they used the predicted metric to construct a modified Laplacian, which was then fed into GIF. The results obtained are far more similar to CM's results than to those produced by vanilla harmonic GIF. However, this method relies on a lengthy preprocessing stage, which is based on a specific dataset. Furthermore, the training is tied to a particular distortion measure, and switching to a different measure requires retraining.

In this paper, we provide a conceptually simple and general algorithm for parameterizing high-resolution meshes in a fraction of the time required by state-of-the-art methods. Our method simplifies the fine input mesh into a low-resolution coarse mesh. We then parameterize the coarse mesh using the same geometric optimization procedure that would otherwise be applied to the full-resolution mesh. A key observation is that the fine mesh Laplacian can be estimated in a fraction of the time compared to a full nonlinear solve, simply by prolonging the so-called Beltrami coefficients (but not the map) from the parameterization of the coarse mesh to the high-resolution mesh. These Beltrami coefficients encode the conformal distortion induced by the map and in the discrete case, they define the angles of the deformed triangles, which leads to a modified Laplacian matrix. Finally, we use the harmonic framework to flatten the mesh. This process is extremely fast, comparable to a single linear solve, and reliably maintains injectivity.

2. Previous Work

There is a vast body of work on map computation and mesh parameterization. In this literature review, we focus primarily on full-space injective parameterization methods and harmonic subspace approaches. We also provide a brief overview of techniques that employ simplification or incremental remeshing for general surface mapping.

Injective Parameterization Methods Over the last decade, research on injective mappings has developed significantly, giving rise to a wide range of approaches.

One well-studied family of techniques addresses this issue by reformulating the optimization problem as a convex one. These approaches avoid initialization altogether [Lip12, BCE*13, APL14, KABL14], but their robustness is limited: the convexified feasible set may be empty even when the original nonconvex problem has a solution.

Another approach is to design energies whose global minimizer corresponds to an injective map [XCGL11, WMZ12, DAZ*20, DKZ*21]. Related formulations eliminate singularities at degenerate elements, enabling the optimization process to potentially "recover" injectivity [GKK*21, DKZ*22, WGS23, SLS22, OKN21]. While these strategies often achieve high success rates in practice, they lack theoretical guarantees of convergence or optimality.

In contrast, interior-point methods explicitly preserve injectivity throughout the optimization. Starting with a bijective initialization, these methods incorporate barrier terms that grow

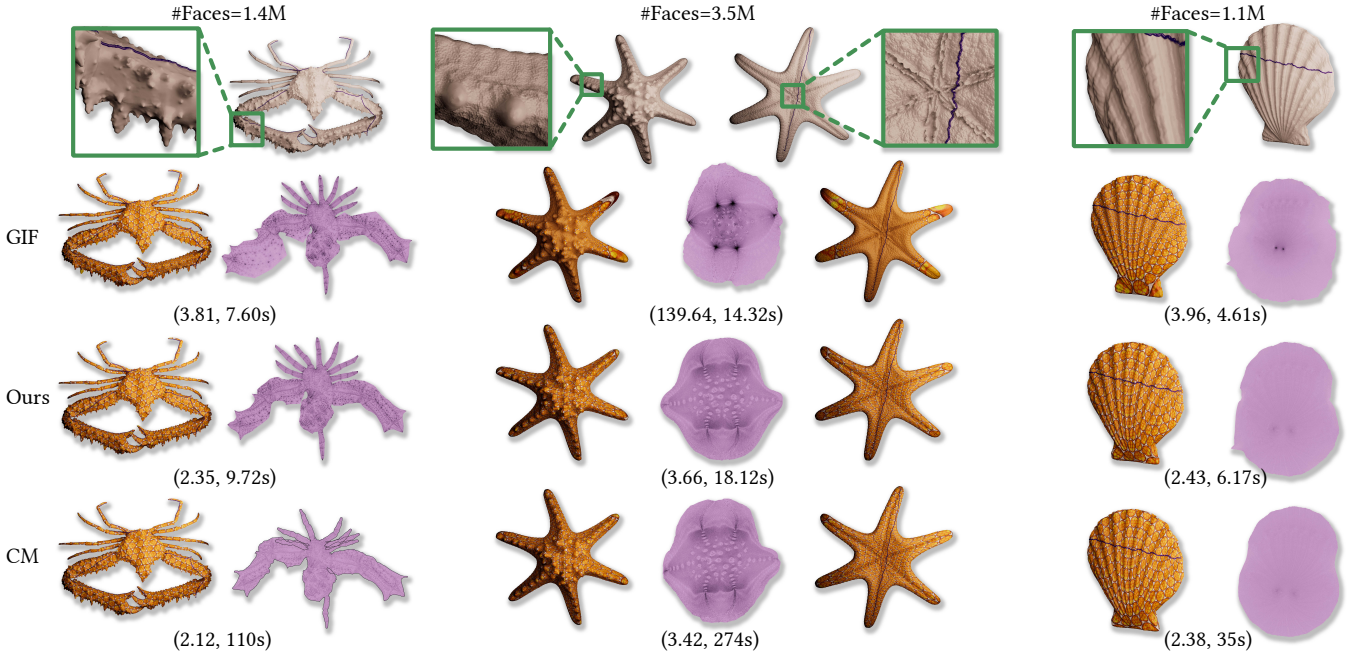


Figure 2: *Comparison to CM and GIF on high-resolution meshes with geometric features across the entire frequency domain. Our method consistently achieves low isometric distortion, which is comparable to CM, in a fraction of the time. GIF results have significantly higher area distortion. The symmetric Dirichlet energy and the runtime in seconds are in brackets.*

unbounded as elements collapse. This maintains local injectivity [SKPSH13, RPPSH17, LYNF18, CBSS17, KGL16, SGK19, SPSH*17, SYLF20]. Commonly used isometric energies in this context include symmetric Dirichlet [SS15], AMIPS [FLG15], Neo-Hookean strain [Ogd97], symmetric As-Rigid-As-Possible (SARAP) [PL16], etc. These methods are applicable when a valid initialization is accessible, such as for disk-like surfaces where Tutte's embedding provides a natural starting point [Tut63]. Nevertheless, these methods guarantee injectivity only in exact arithmetic. In floating-point implementations, even Tutte's approach can fail [SJZP19,FBRCA23]. Our method shares this limitation, though we have not encountered such numerical failures in practice.

The need for valid initializations becomes more restrictive when positional constraints are introduced. Projection-based methods allow optimizations to start from non-injective maps to mitigate this issue, as discussed in [AL13, KABL15, HFCW17].

Together, these strands of work define a well-established landscape of injective parameterization techniques. However, the nonlinear and iterative nature of these algorithms makes them much slower than linear or subspace-based approaches. In this paper, we aim to develop a method that bridges the gap between these two classes by computing the result of the full-space method over a coarse mesh. This allows for a simple, yet powerful modification of the subspace.

Conformal Parameterization Methods Conformal maps are smooth, angle-preserving maps that are locally injective by construction, though they do not preserve lengths. In the field of geometry processing, conformal parameterizations are of central importance and are supported by a robust mathematical foundation [WG10, SSP08, MZ13, SSC18, SC17]. From a computational standpoint, significant progress has been made in recent years regarding the *robust* evaluation of discrete conformal maps [GSC21, CCS*21, SWGL15]. These approaches enable mappings from general surfaces to flat domains, potentially incorporating cone singularities and automatic connectivity changes.

Harmonic Parameterization Methods The space of harmonic maps is a linear subspace that generalizes the space of conformal maps. Unlike conformal maps, harmonic maps are not always injective; however, their injectivity is entirely determined by their boundary behavior. This distinctive property, coupled with their vector space structure, makes harmonic maps widely used [CW15, LW16, CCW16, CW17, HFCW17, WGS23].

In the discrete setting, Convex Combination Maps (CCMs) are the closest analog. Tutte's embedding [Tut63] and its extensions [Flo97] guarantee a bijective mapping of a disk-like mesh onto a convex polygon by solving a single sparse linear system. Gortler et al. [GGT06] further proved that CCMs yield bijective maps to flat tori, and Aigerman et al. [AL15] generalized this idea to Euclidean orbifolds. These methods are advantageous for their simplicity, efficiency, and robustness. However, they are applicable only in restricted settings where boundaries are fixed to convex domains or are entirely absent, often producing severe distortion.

Several works have attempted to overcome these limitations. Weber et al. [WZ14] combined two harmonic maps to enforce injectivity when the target shape is not convex. However, allowing the boundary to move freely typically leads to lower-distortion solu-

tions. Bright et al. [BCW17] established general injectivity conditions for CCMs on arbitrary topologies with free boundaries and cone singularities. Building on this work, Hefetz et al. introduced Fast Locally Injective Harmonic Mapping (FLIM) [HFCW19]. This method efficiently constructs a compact harmonic subspace using the selective inverse technique [KLS13, VCKS17] by extracting the necessary rows of the Laplacian inverse without computing it in full. The cost of this construction is comparable to solving a sparse linear system. Once the subspace is constructed, finding a locally injective solution reduces to a small nonlinear optimization problem, typically involving only boundary degrees of freedom.

A related approach is Globally Injective Flattening (GIF) [FW22], which employs the same harmonic subspace framework as FLIM, together with Jiang et al.'s scaffolding technique [JSP17] to compute globally injective maps with free boundaries. Freeing the boundary consistently yields harmonic maps with minimal isometric distortion within this class.

The downside of harmonic parameterizations is that they struggle with highly curved surfaces and often have significantly higher isometric distortion than non-harmonic techniques. Our method strikes a balance by combining the computational efficiency and robustness of harmonic-based approaches with the distortion levels of more general techniques, all without relying on learning-based techniques.

Quasi-Conformal maps Our method differs from the standard harmonic subspace approach because it substitutes the Laplacian that is induced from the metric of the 3D embedding with an alternative one. Several prior works have explored the use of altered metrics in the context of harmonic and conformal maps, though in different applications.

For instance, [CWKBC13, FBC25] interpolate planar shapes by blending the metric of the source and target meshes. The resulting blended metric has pointwise bounded distortion with respect to the source and target shapes. Since, in general, this blended metric is not flat, they apply conformal (which is also harmonic) flattening, producing a flat metric while preserving the conformal error bound.

Weber and Zorin [WZ14, Section 5.3] modify the original surface metric to approximate an extremal quasi-conformal map under fixed boundary conditions, building on their earlier work [WMZ12]. Similarly, Wang et al. [WGS23] construct diffeomorphisms by optimizing within the space of quasi-harmonic maps that satisfy specialized Cauchy boundary conditions. Their method optimizes the modified metric, expressed in terms of complex Beltrami coefficients, using a variational formulation combined with a nonlinear iterative procedure. This work also contributes a detailed theoretical framework for harmonic maps with variable metrics, both in smooth and discrete settings.

Neural-Based Methods [AGK*22] is the first method to compute surface parameterization using a data-driven approach. The authors trained a neural network to predict the per triangle Jacobian of the parameterization. The inferred non-integrable Jacobians are fed into a Poisson solver in order to recover the final map. In [FW25] a neural network is trained to predict the matrix logarithm of the 3×3 pullback metric tensor of each triangle and computes the final map using harmonic flattening based on a modified Laplacian of the predicted metric. Instead, our method uses a geometric construction to "predict" the modified metric.

Simplification-Based Methods Simplification is used to some extent in map computations. [BCE*13] computes mesh quadrangulation via parameterization and decimates the mesh in the flat parametric space while maintaining a map between the coarse and original meshes throughout the optimization process. [SPK23] computes inter-surface maps between multiple sphere-like meshes by mapping each mesh onto an adaptively refined sphere triangulation. This allows for efficient "direct" energy optimization via a coarse overlay. While efficient, applying this technique into the standard surface parameterization is not trivial. [SLMB05, LSS*98, HGC99] simplify a mesh, perform direct optimization on the simplified model, and prolong the UVs linearly to obtain the parameterization of the original mesh. The main problem with this approach is the fact that the distortion of the triangles in the original mesh is unbounded, leading to frequent near-collapsed and flipped triangles, which cannot be easily fixed. Another problem is that prolonging the UVs usually yields piecewise smooth parameterizations, with sharp changes between patches.

The multigrid approach of [LZBCJ21] prolongs functions across different hierarchy levels based on their intrinsic geometry. It solves iterative geometric optimization efficiently. Albeit, the construction of the hierarchy is time consuming and using it to prolong UVs may violates injectivity and is not suitable for interior point methods. [LGC*23] is a surface simplification method that uses intrinsic geometry to construct a bijective map between the simplified and original meshes. [YSLF20] proposes a surface parameterization approach intended for very large meshes for which a direct solver is not applicable. It optimizes the energy over the entire model where only a few control points are allowed to move. The energy-descent direction for any other vertex is defined by interpolating the descent directions of the control points. Finally, [LLT*20] proposes a spectrum-preserving mesh decimation scheme that simplifies a mesh while ensuring that the Laplacian of the simplified mesh remains spectrally close to that of the original.

3. Algorithm

Given an input 3D disk-like triangle mesh, our algorithm first simplifies it (Section 3.1) and runs a full-space energy-minimizing procedure over it (Section 3.2). Then, we extract the per triangle Beltrami coefficient from the triangles of the simplified mesh, prolong them over the entire input mesh (Section 3.3), and construct a modified Laplacian (Section 3.4). Finally, we use a robust harmonic flattening method to compute an injective (locally or globally) parameterization based on this modified Laplacian (Section 3.5). The main building blocks of our algorithm are illustrated in Figure 3.

Our algorithm can handle any mesh size. However, for the majority of popular distortion measures, most advanced parameterization algorithms can process meshes with several thousand triangles in a fraction of a second. While it is still possible to accelerate such a process, we focus on larger meshes, where the benefit of using our acceleration technique will be substantial.

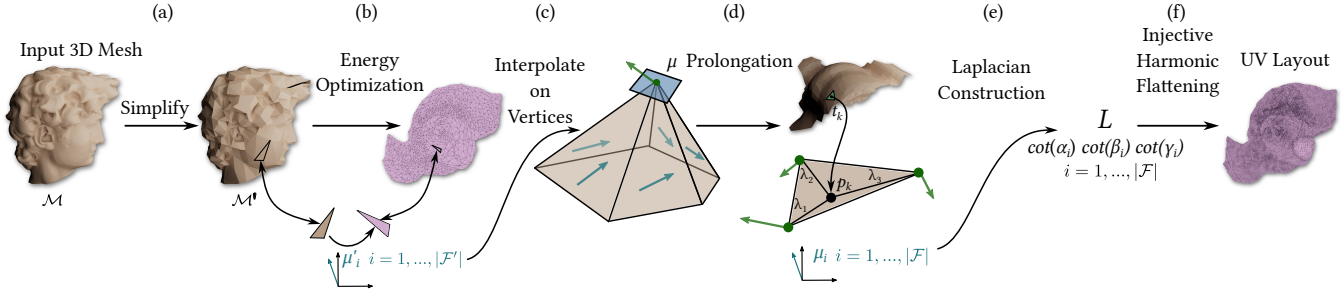


Figure 3: *Steps of our algorithm. The input is a 3D mesh $\mathcal{M}(\mathcal{V},\mathcal{E},\mathcal{F})$, and the output is a UV layout represented by $(u_i,v_i)\in\mathbb{R}^2, i=1\dots|\mathcal{V}|$. After simplification process (a) that creates $\mathcal{M}'(\mathcal{V}',\mathcal{E}',\mathcal{F}')$, we run a global optimization process to flatten it (b). We then extract the Beltrami coefficient of each coarse triangle, and smooth it by interpolating the Beltrami coefficient into the coarse vertices (c). After that, we project each fine triangle onto the coarse mesh (d), such that the projected centroid defines the interpolated Beltrami coefficient of the fine triangle. The Beltrami coefficients define a set of angles for each fine triangle (e), which in turn defines the modified Laplacian used in the harmonic flattening process (f).*

3.1. Mesh Simplification

Given an input 3D disk-like high-resolution triangle mesh, denoted by $\mathcal{M}$ with $(\mathcal{V},\mathcal{E},\mathcal{F})$ vertices, edges, and faces respectively, we first simplify the mesh (Figure 3(a)). In all our experiments (unless stated otherwise), we use a simplified mesh with 1% the number of triangles compared to the input mesh, where for meshes with less than a million triangles, we use a cutoff of 10,000. The simplified mesh is denoted $\mathcal{M}'(\mathcal{V}',\mathcal{E}',\mathcal{F}')$. We used the parallel implementation of the celebrated QSLIM method [GH97], that is part of the open source MeshLib package [Mes25]. Our algorithm is quite robust to the chosen simplification method since even when the approximation is of low quality, an injective parameterization is guaranteed.

3.2. Energy Minimization

Next, we run a full-space energy-minimizing method on the simplified mesh $\mathcal{M}'$, and extract the Beltrami coefficient μ_i of each triangle $t_i\in\mathcal{F}'$ (Figure 3(b)). For all results in this paper, we use the CM full-space energy minimization procedure from [SPSH*17]. Our algorithm is general and is not tied to any particular numerical optimization procedure or a specific choice of a distortion error. However, we focus on isometric measures since conformal energies and harmonic maps are targeted by existing algorithms that are already fast enough.

3.3. Beltrami Coefficient Approximation

Let $f: S\to\mathbb{R}^2$ be an orientation–preserving C^1 map from the surface S to the plane. We denote the 2×2 Jacobian matrix in local coordinates by:

$$J=\begin{pmatrix} u_x & u_y \\ v_x & v_y \end{pmatrix}. \tag{1}$$

The complex Wirtinger derivatives are:

$$f_z=\frac{1}{2}\Big(u_x+v_y+i(v_x-u_y)\Big)\quad f_{\bar z}=\frac{1}{2}\Big(u_x-v_y+i(v_x+u_y)\Big). \tag{2}$$

The *complex Beltrami coefficient* is: $\mu=\frac{f_{\bar z}}{f_z}$. Geometrically, $\mu=ke^{2i\theta}$ encodes the conformal distortion of f. Its magnitude $k=|\mu|\in[0,1)$ the small dilatation, encodes the amount of shear that is induced by f. It is related to the large dilatation $K=\frac{\sigma_1}{\sigma_2}=\frac{1+k}{1-k}$, where σ_1 and σ_2 are the singular values of J which encodes the maximal and minimal stretches. The angle $\theta=\frac{1}{2}\arg\mu$ encodes the direction in which the maximal stretch occurs.

The next step of our algorithm is to transfer the Beltrami coefficients from the simplified mesh to the fine input mesh. On a triangle mesh, if f is a continuous and piecewise linear (P.W.L.) map, then J_i and μ_i are constant per triangle. If we translate the vertices of the source and target triangles such that the first vertex is at the origin and the vertex positions are $(0,v_1,v_2)$ and $(0,v_1',v_2')$ in local coordinates respectively, we get:

$$f_z=\frac{v_1'\overline{v_2}-v_2'\overline{v_1}}{v_1\overline{v_2}-v_2\overline{v_1}},\quad f_{\bar z}=\frac{v_1v_2'-v_2v_1'}{v_1\overline{v_2}-v_2\overline{v_1}},\quad \mu=\frac{v_1v_2'-v_2v_1'}{v_1'\overline{v_2}-v_2'\overline{v_1}}. \tag{3}$$

We observed that transferring μ from a single triangle of the simplified mesh to a patch of multiple corresponding triangles in the input mesh produces UV maps that lack smoothness across patch boundaries. A straightforward remedy is to smooth the UVs in post-processing. However, performing such smoothing efficiently while preserving bounded distortion and \ or maintaining injectivity is challenging. In contrast, smoothing the Beltrami-coefficient field rather than the UVs is simple, efficient, and robust, with qualitative guarantees. To see this, recall that the injectivity of the harmonic map is guaranteed (under mild assumptions) regardless of the choice of Beltrami coefficients. Moreover, by linearly interpolating with barycentric coordinates,

$$\mu=\lambda_1\mu_1+\lambda_2\mu_2+\lambda_3\mu_3, \tag{4}$$

the conformal distortion induced by the blended μ remains bounded by the conformal distortion of μ_1, μ_2, and μ_3. Let $R=\max\{|\mu_1|,|\mu_2|,|\mu_3|\}$. Then the disk of radius R centered at the origin contains μ_1, μ_2, and μ_3. Since the disk is convex and μ is a convex combination of points inside it, μ also lies within the disk. Hence, $k=|\mu|\le R$, or equivalently

$$k\le\max\{k_1,k_2,k_3\},\quad k_i=|\mu_i|,\ i=1\dots3 \tag{5}$$

The Beltrami coefficient μ is represented as a complex scalar. However, just like the pullback metric, μ behaves as a tensor and is only meaningful with respect to a chosen coordinate system. Since our μ is defined on a curved manifold, no globally consistent coordinate system exists. Instead, each triangle carries its own arbitrarily chosen local coordinate system. As a result, directly copying μ from one location on the mesh to another is meaningless. To transfer μ consistently between coordinate systems—for example, from a face to its vertices or vice versa—we rely on parallel transport [SSC19, Section 5].

Our smoothing procedure consists of several steps. In the first step, we transport the Beltrami coefficient from the local coordinate system of each coarse triangle to the coordinate systems of its three vertices. After all coefficients have been expressed in their respective vertex coordinate systems, we compute a weighted average over each one-ring neighborhood. The weights are convex and proportional to the areas of the adjacent triangles, ensuring a smooth and well-balanced interpolation (Figure 3(c)).

Next, we project the fine mesh $\mathcal{M}$ onto the coarse mesh $\mathcal{M}'$. For each fine triangle $t_k \in \mathcal{F}$, we locate the closest point $p_k \in \mathcal{M}'$ to its centroid. This point p_k lies in a coarse triangle $t_j \in \mathcal{F}'$ and is represented by its barycentric coordinates $\lambda_1(p_k), \lambda_2(p_k), \lambda_3(p_k)$ with respect to the vertices of t_j (Figure 3(d)). Our objective is to assign a Beltrami coefficient μ_k to the fine triangle t_k. We achieve this by interpolating the Beltrami coefficients defined on the vertices of the coarse triangle t_j using the barycentric weights of p_k. Since Beltrami coefficients are defined in local coordinate systems, the vertex coefficients of t_j must first be parallel transported into the common coordinate system of t_j before interpolation. This yields an interpolated coefficient μ_k, which at this stage is expressed in the coordinate system of t_j. Finally, to make μ_k compatible with the geometry of the fine mesh, we perform an additional parallel transport from the coarse triangle t_j to the fine triangle t_k as detailed next.

For two adjacent triangles sharing an edge, a standard way to parallel transport a vector from one triangle to another is by conceptually flattening the dihedral angle by rotating one triangle along the common hinge, making the problem trivial. To generalize parallel transport to arbitrary triangles, we adopt the same idea of rotating around a hinge. However, in our case the two triangles may differ in size and even belong to different meshes, so there is no shared edge that can serve as a hinge. Instead, we use the direction of the line of intersection between the two supporting planes as the hinge. If the two 3D triangles have the same unit normals, i.e., $n_k = n_j$, there is no unique intersection line but the problem becomes trivial. Otherwise, the vector $d_{kj} = n_k \times n_j$ gives the hinge direction. Since d_{kj} is orthogonal to both normals, it lies in the tangent plane of both triangles. We encode d_{kj} in the local coordinate systems of the two triangles and compute its angles α_k and α_j with respect to the positive x-axes. Finally, given a vector u_j in t_j's coordinate system represented as a complex number, the transported vector is $u_k = e^{i(\alpha_k - \alpha_j)} u_j$. This process is illustrated in Figure 4. Since μ is transported like a tensor with $\arg\mu = 2\theta$ (where θ encodes the stretch direction), the transport coefficient must be squared, i.e., $\mu_k = e^{2i(\alpha_k - \alpha_j)} \mu_j$.

Note, that our closest point projection does not necessarily produces a bijective map since it is done in an extrinsic fashion, relying on the 3D embedding of $\mathcal{M}$ and $\mathcal{M}'$. Bijective correspondence based on intrinsic geometry would be ideal and can be used, but current implementations, e.g., [LGC*23] turned out to be much more computationally intensive. Moreover, as we'll see in Section 4, bijective correspondence between $\mathcal{M}$ and $\mathcal{M}'$ does not lead to bijective parameterization in general. On the other hand, our non-bijective projection and Beltrami prolongation does not affect our guarantees for the injectivity of the final parameterization since the final harmonic flattening ensures this independently of the modified Laplacian. Therefore, we opted for this simple choice which turned out to be extremely fast while providing excellent quality.

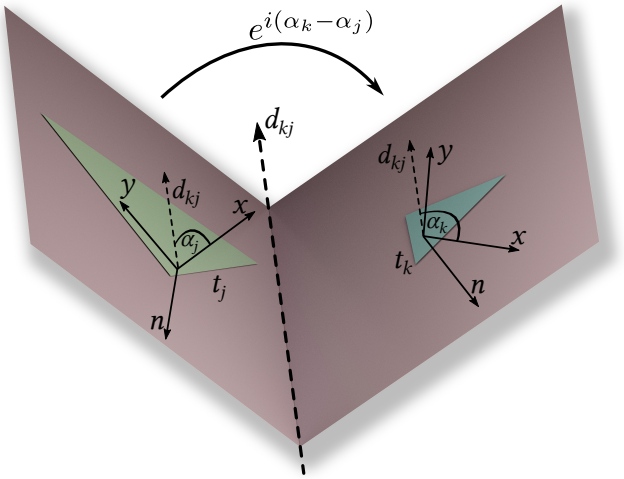


Figure 4: *Parallel transport of a tangent vector from a coarse triangle t_j to a triangle in the input mesh t_k. First, we compute the direction of the intersection line d_{kj} between the supporting planes of the two triangles. Then, we measure the angle of d_{kj} relative to the local x-axis in both coordinate systems, denoted by α_j and α_k. The rotation required to map a 2D vector from t_j to its representation in t_k is $\alpha_k - \alpha_j$. In complex form, this is equivalent to multiplying the scalar by $e^{i(\alpha_k - \alpha_j)}$.*

3.4. Modified Laplacian Construction

After we obtain a Beltrami coefficient $\mu_k, k = 1 \dots |\mathcal{F}|$ for each triangle in the input mesh $\mathcal{M}$, we construct the modified Laplacian matrix. Note, that μ defines the metric only up to isotropic scale. Hence, in general, a globally compatible metric with these Beltrami coefficients does not exist. In [FW25, Sec. 3.1], the pullback metric tensor is predicted per triangle. Such a metric tensor includes an isotropic scale, which is not encoded by μ. However, neighboring triangles may have a common edge with incompatible induced lengths. Since the Laplacian construction only requires angles, [FW25] computes the angles of the neighboring triangles independently, ignoring the incompatibility. This is done by summing the cotangents of the angles opposite to the common edge. Later on, [FBC25, Sec. 6.1.1] used a similar construction, denoting this Laplacian as the Corner Laplace-Beltrami operator.

We go one step further and decide to work with the Beltrami coefficients instead of the metric. Dropping any information re-

lated to isotropic scale, leaving behind edge length compatibility. Working with Beltrami coefficients instead of the metric tensor is simpler and more efficient as μ can be expressed using complex numbers. More importantly, an incompatible metric is not, strictly speaking, a metric. On the other hand, it is known from smooth quasi-conformal theory that for any measurable function $\mu(z) \in L^\infty$ that is bounded up to a set of measure zero with $\|\mu\|_\infty < 1$, there exists an orientation-preserving quasi-conformal parameterization $f : S \to \mathbb{R}^2$, that solves the Beltrami partial differential equation $f_{\bar{z}} = \mu f_z$. The solution is unique up to post-composition by a conformal map. In simple words, for our discrete setting, it is reasonable to assume that for any set of prescribed μ_k, $k = 1 \dots |\mathcal{F}|$, a parameterization with approximately these Beltrami coefficients exists. In the smooth case, such a map is quasi-conformal, i.e. locally injective, orientation preserving, with a bound $\|\mu\|_\infty$ on conformal distortion. The fact that the Beltrami equation is always solvable in the smooth case, producing a map that satisfies prescribed Beltrami coefficients, makes the Beltrami coefficients preferable to alternatives such as the Jacobians. Interpolated Jacobians (even in the smooth setting) often have non-zero curl and require a Poisson projection, which offers no control over the l_∞ conformal distortion and can easily introduce flips.

Our final design choice is to use a harmonic parameterization rather than conformal, since this subspace provides more flexibility and contain conformal maps as a special case. Harmonic maps are easier to work with since they form a linear subspace. Furthermore, it has the potential to lead to a map with less isometric distortion.

Given a triangle t_k, our algorithm computes the cotangents of respective angles after μ_k is applied. Locally, up to an irrelevant similarity transformation (isotropic scale and rotation) a quasi-conformal map with Beltrami coefficient μ acts on an infinitesimal complex vector v by the real-linear map $L_\mu(v) = v + \mu\bar{v}$. Let $(0, v_1, v_2)$ be the triangle vertex positions represented as complex numbers in a local coordinate system. In order to compute the cotan of the angle α that belongs to the first vertex, we use

$$\cot\alpha = -\frac{\mathrm{Re}(q)}{\mathrm{Im}(q)}, \qquad q = (v_1 + \mu\overline{v_1})(\overline{v_2} + \bar{\mu}v_2). \tag{6}$$

We simply go over all triangles t_k and use this formula 3 times, adding corresponding cot terms to to the relevant entries in the Laplacian matrix (Figure 3(e)).

3.5. Harmonic Flattening

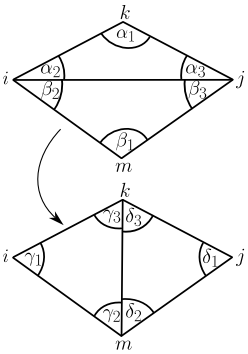
At this point we can use any harmonic algorithm that is fast and robust with respect to injectivity. See Figure 3(f). The simplest choice would be to use a conformal map. The BFF algorithm [SC17] is one choice, though there are no strict guarantees for injectivity, and the restriction to conformality may lead to slightly increased isometric distortion. Other viable choices are the harmonic locally injective method FLIM [HFCW19] or its globally injective variant GIF [FW22], which also has a more optimized implementation. Unless stated otherwise, we use the latter. The computational complexity of both FLIM and GIF is comparable to a single linear system solve. It constitutes the majority of the runtime in our entire pipeline.

3.5.1. Foldovers Handling

A sufficient condition for our final algorithmic step of harmonic flattening [FW22, HFCW19] to guarantee injectivity is positive Laplacian weights. Various techniques exist for ensuring this condition [FW25]. However, this is merely a sufficient but not a necessary condition, and ignoring it still leads to injective map in the vast majority of our results. Flipped triangles happen rather rarely in our experiments, hence, we followed the strategies of [HFCW19] and [FW22] that handle flipped triangles in a post process. If foldovers appear in our final result, we first use the fast local fix of [HFCW19] that reposition vertices that belongs to flipped triangles in the kernel of their 1-ring. This consumes negligible amount of computation time. In case this strategy fails, we revert to the fully robust global approach proposed by [FW22] which requires one additional linear solve using positive weights, which are extracted directly from the UV layout. Hence, have extremely local effect. As expected, applying this hybrid strategy produced injective parameterizations for all the models in our dataset.

3.5.2. Angles-Based Intrinsic Delaunay Triangulation

Another simple and elegant way to deal with negative Lapalacian weights is to use an intrinsic Delaunay triangulation (iDT) [SGC21], which produces a new metric that is isometric to the input metric. This is generally done via a finite series of edge flips, where each iteration flips a non-Delaunay edge, turning it into a Delaunay one, until all edges are Delaunay. Note that the standard iDT procedure requires an input *metric*. In our case, we define the modified harmonic subpsace only using the cotangent triplet of each triangle, and no actual metric is utilized. However, the modified Laplacian after the iDT does not depend on any metric either, but rather yet another set of cotangent triplet per triangle. Additionally, the Delaunay criteria depends solely over the cotangents as well: an edge is Delaunay if and only if the sum of its 2 opposite cotangents is non-negative. We also observe that in each edge flip, the cotangent triplets of the 2 output triangles, depend solely on the cotangent triplets of the 2 input triangles. This essentially means that we can still run the standard iDT procedure, where we equip the triangulation only with a cotangent triplet per triangle, rather than edge lengths. For a given angle θ, denote by $\mathfrak{C}_\theta$ the function $1 + \cot^2\theta$. Given the cotangent triplets of the 2 input triangles (see inset), the cotangent triplets of the output triangles are given by



$$\cot\gamma_1 = \frac{\cot\alpha_2 \cdot \cot\beta_2 - 1}{\cot\alpha_2 + \cot\beta_2}, \qquad \cot\delta_1 = \frac{\cot\alpha_3 \cdot \cot\beta_3 - 1}{\cot\alpha_3 + \cot\beta_3}$$

$$\cot\gamma_2 = \sqrt{\frac{\mathfrak{C}_{\beta_1}\mathfrak{C}_{\alpha_3}\mathfrak{C}_{\gamma_1}}{\mathfrak{C}_{\beta_3}\mathfrak{C}_{\alpha_1}}} - \cot\gamma_1, \qquad \cot\delta_3 = \sqrt{\frac{\mathfrak{C}_{\beta_2}\mathfrak{C}_{\alpha_1}\mathfrak{C}_{\delta_1}}{\mathfrak{C}_{\beta_1}\mathfrak{C}_{\alpha_2}}} - \cot\delta_1$$

$$\cot\gamma_3 = \frac{\cot\alpha_1 \cdot \cot\delta_3 + 1}{\cot\delta_3 - \cot\alpha_1}, \qquad \cot\delta_2 = \frac{\cot\beta_1 \cdot \cot\gamma_2 + 1}{\cot\gamma_2 - \cot\beta_1}$$

The standard iDT procedure, which operates on the actual edge lengths, has been proven to converge after a finite sequence of edge flips [BS07], yielding an iDT metric. The proof consists of two main steps. First, it is shown that every edge flip strictly decreases a global measure defined over the mesh [BS07, Proposition 15], such as the harmonic index [Mus97], which depends only on the triangulation's angles. Second, since each flip of a non-Delaunay edge strictly decreases this measure, and because the number of possible triangulations of the vertex set $\mathcal{V}$ is finite, the procedure must terminate after finitely many flips. The resulting triangulation minimizes the global measure and is therefore Delaunay [BS07, Proposition 12]. These arguments apply directly to our setting. Because the harmonic index depends only on angles, and because flipping an edge in our quadrilateral configuration is equivalent to performing a standard iDT edge flip on a rescaled quadrilateral (with both triangles scaled so that their shared edge lengths match), the harmonic index strictly decreases in our case as well. Moreover, since the number of possible triangulations of $\mathcal{V}$ is finite and each flip strictly decreases the harmonic index, our algorithm must terminate after finitely many flips, with no possibility of revisiting the same connectivity. The final triangulation is guaranteed to be Delaunay, as the procedure systematically flips all non-Delaunay edges until none remain.

The iDT procedure described above extends the standard formulation by relying solely on cotangents rather than edge lengths. When applied over cotangents set derived from an actual edge-length metric, the result is identical to that of the standard iDT. Given an input metric, the iDT procedure provides new connectivity with new edge lengths, describing the same metric with a more robust representation. The key advantage of this representation in our case is that all cotangent weights in the Laplacian of the resulting iDT are positive. As a result, the harmonic flattenings of [FW22, HFCW19] provably yield injective parameterizations without requiring any additional foldover corrections. Using this strategy, we obtained injective parameterizations for all the models in our dataset.

3.5.3. Injectivity

When performing harmonic flatteing in a modified metric, a sufficient condition for injectivity is positive Laplacian weights. The interpolated Beltrami coefficients don't always induce positive cotan weights. In Sections 3.5.1 and 3.5.2 we propose two strategies to address this issue. The first strategy completely ignores this limitation and, if foldovers occur, applies a robust correction technique to eliminate them. The second strategy applies the iDT procedure, based entirely on the cotangent triplets of the triangles, which by construction produces a Delaunay triangulation with positive cotangent weights. Under exact arithmetic, both strategies provably guarantee injectivity and were fully implemented and tested. In practice, we obtained flip-free maps across the entire dataset with either approach, validated using a robust orientation test based on exact predicates [CGA24]. The first strategy was used for all the results in our paper due to its simplicity, connectivity preservation, and to enable a more fair comparison to CM and GIF, which don't allow connectivity changes.

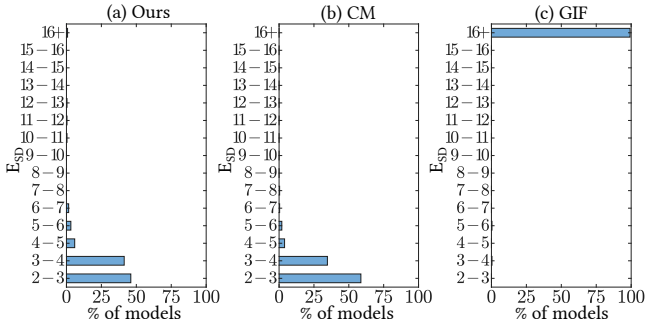


Figure 5: *Distortion distribution of 3 parameterization methods on the entire dataset (252 meshes). Each histogram shows the percentage of meshes in the dataset for different* E_{SD} *levels in the range* $[2-16]$. *GIF produces results with significantly high distortion, with almost all results having* $E_{SD} > 16$. *In contrast, our method produces results with less distortion that better resembles CM.*

4. Results

We implemented our method using C++, Matlab, and Python. All experiments were conducted on a Windows 10 machine with an Intel i9-13900KS CPU and 128GB RAM. We used the authors' implementation of CM [SPSH*17] and GIF [FW22] as the main building blocks of our method, as well as for comparisons. Both CM and GIF utilize the PARDISO 8.2 linear solver [Pan25]. For the closest point queries, we used the functionality from the Open3D Python library [ZPK18]. Our code and data are publicly available on the authors' website.

We mainly tested two types of popular distortion measures. The symmetric Dirichlet energy [SS15] with energy density $E_{SD} = \frac{1}{2}(\sigma_1^2 + \sigma_2^2 + \sigma_1^{-2} + \sigma_2^{-2})$, and the symmetric as-rigid-as-possible energy [PL16] with a density $E_{SARAP} = (\sigma_1 - 1)^2 + (\sigma_2^{-1} - 1)^2$. Since both measures are isometric, they are minimized when $\sigma_1 = \sigma_2 = 1$. E_{SD} and E_{SARAP} have minimal values of 2 and 0 respectively.

To stress test our method, we created a new dataset which is based on models from the D2 group in the dataset of [LYNF18] as opposed to those of GIF [FW22], which are based on the "well cut" group D1 and tend to have low overall symmetric Dirichlet energy. We first excluded models for which CM achieved a symmetric Dirichlet energy below 2.5. The remaining models were subdivided to resolutions of up to 4M triangles. Finally, we uniformly sampled the 20K–4M triangle range to ensure a balanced distribution of mesh sizes, resulting in a dataset of 252 models. In addition to our dataset, we picked additional high-quality high-resolution scans from [Thr, RPPSH17]. These are showcased in Figures 1, 2, and 15.

UV Prolongation Baseline Earlier approaches that try to accelerate mesh parameterization using mesh simplification directly prolong the UVs [SLMB05, LSS*98, HGC99] from a low-resolution mesh to a high-resolution one. These methods do not optimize modern barrier-like isometric measures such as E_{SD} or E_{SARAP} and does not guarantee injectivity. For a fair comparison, we implemented a baseline approach that optimizes E_{SD} on a simplified mesh

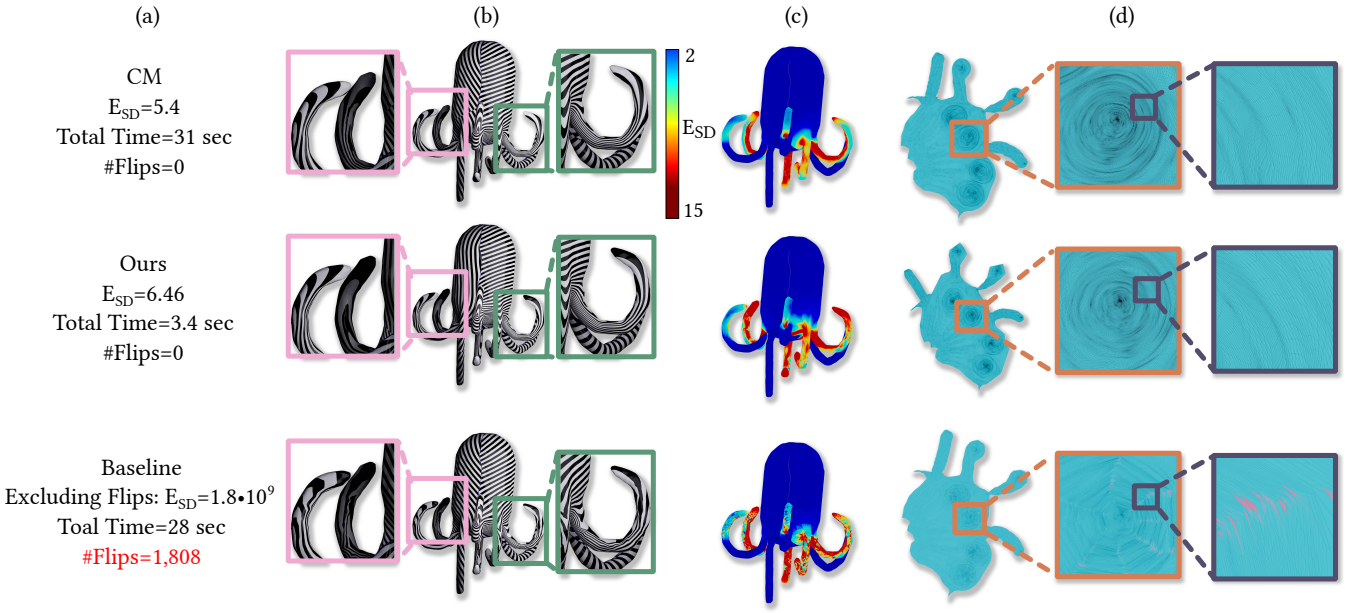


Figure 6: *Comparison to a UV prolongation baseline approach on a tough octopus model with* 303,840 *faces. (a) Both our method and CM produce injective maps with low distortion, while the baseline introduces* 1,808 *flips and has an extremely high energy due to nearly collapsed triangles. Because flips make* E_{SD} *infinite, we excluded them from the baseline's energy computation. (b) Applying a stripe pattern texture yields smooth results for CM and our method, while the UV prolongation approach introduces broken stripes, indicating a non-smooth result. (c) Color visualization of* E_{SD} *values between 2-15. While CM and our method demonstrate smooth changes, the baseline exhibits region-wise constant distortion, due to the UV prolongation. (d) The UV layout with zoom-in views. The baseline result is nonsmooth and introduces flipped triangles (pink) in regions corresponding to transition between coarse triangles.*

and prolong the UVs rather than the Beltrami coefficients to the fine mesh.

We used the state-of-the-art intrinsic simplification method of [LGC*23] which produces a simplified triangle mesh connectivity $\mathcal{M}'$ along with a metric (edge lengths without an embedding). The method also provides a bijective P.W.L. correspondence $g : \mathcal{M} \to \mathcal{M}'$ between the input fine mesh and the simplified one. Unlike extrinsic simplification methods, the intrinsic approach has greater potential to provide a high fidelity coarse representation since it operates in a much larger space of metrics which are not necessarily embeddable in $\mathbb{R}^3$. The position of each vertex v_i of the fine mesh under the correspondence $g(v_i)$ is encoded using the index of the corresponding triangle in $\mathcal{M}'$ and a set of 3 barycentric coordinates. We flatten the simplified mesh by a map $f : \mathcal{M}' \to \mathbb{R}^2$, and obtain the parameterization of the fine mesh by composing the two maps $f \circ g : \mathcal{M} \to \mathbb{R}^2$. Ideally, we would like to prolong the UVs of the simplified mesh to the fine mesh. However, such UV maps are not, in general, linear on a triangle of the fine mesh, but rather P.W.L. (on each triangle). In other words, in order to ensure that $f \circ g$ is injective, there is a need to refine $\mathcal{M}$ by using the common overlay of $\mathcal{M}$ and $\mathcal{M}'$. Failure to do so leads to parameterizations that are not injective. This failure is not unique to [LGC*23] and is shared by any other simplification approach that prolong UVs directly without mesh refinement. Figure 7 illustrates this problem. Moreover, prolonging UVs often lead to extreme distortion which the baseline approach cannot reliably address. Our method bypass this pitfall by prolonging Beltrami coefficients instead of UVs.

In Figure 6 we compare our result against the baseline and CM on a challenging octopus mesh. The baseline approach produces a map with 1,808 flipped triangles and many outliers with extremely high distortion levels. On the other hand, our result is smooth, injective, and has much lower distortion, despite the fact that we use a simple extrinsic simplification and a prolongation that is not guaranteed to be bijective.

Prolonging the Beltrami coefficients with [LGC*23] would in principle be optimal. However, for models exceeding 2M faces, CM was empirically faster on the high-resolution mesh than the intrinsic simplification step of [LGC*23] alone. Thus, although [LGC*23] achieves superior simplification with bijective correspondence, it is currently impractical for our pipeline. On the largest meshes in our dataset, it scaled poorly, and we terminated the baseline after processing 85 meshes. Across these, the baseline produced 33,931 flipped triangles out of 72.5 million. In contrast, using the strategy of Section 3.5.1, our method initially produced only 974 flipped triangles out of 500 million triangles (an average of 3.8 per model). Moreover, for 163 models our parameterization was fully injective, with no flipped triangles. For the 89 non-injective cases, the local post-processing of Section 3.5.1 restored injectivity in nearly all cases; only five required the global fix, which entails a single additional linear solve. Ultimately, all

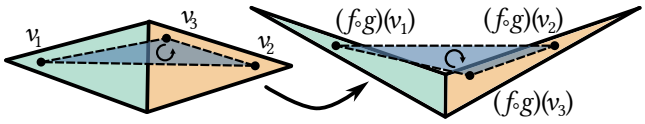


Figure 7: *UV prolongation can lead to flipped triangles even if there is a P.W.L. bijective correspondence between the input mesh and the simplified mesh, and the simplified mesh is flattened in an injective manner. (left) a triangle of the input mesh (blue) is mapped onto the simplified mesh triangles (green and orange) by an orientation preserving map g. (right) The green and orange triangles are mapped by an injective P.W.L. map f to the plane. Due to the distortion of f, when applying the composition of the two P.W.L. maps $(f \circ g)$ on v_1, v_2, v_3, the image triangle has negative orientation (flip). While $(f \circ g)$ is still P.W.L., it is not P.W.L. on the triangles of the input mesh and without mesh refinement the map is not injective.*

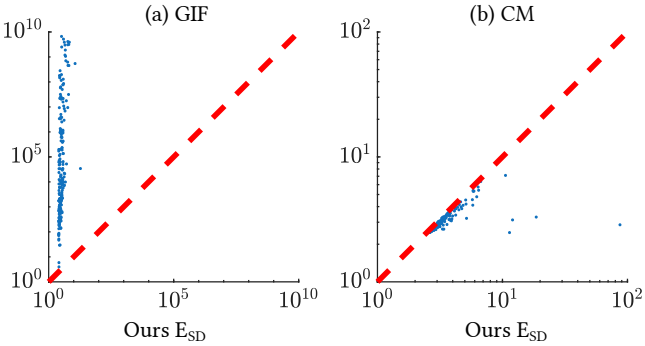


Figure 8: *Distortion comparison between our method, GIF and CM. Each scattered plot compares our method to a different approach, using a logarithmic scale on both axes to accommodate the high dynamic range of distortion values. Each blue dot represents a single mesh from our dataset. (a) our method consistently achieves significantly lower distortion than GIF, as indicated by dots above the red dashed line $y = x$. (b) CM optimizes the distortion on the fine mesh directly and therefore produces lower distortion than our method, and substantially lower distortion than GIF. However, beside several outliers, all results have relatively close distortion.*

parameterizations were made fully injective, as verified with exact arithmetic (CGAL exact predicates [CGA24]).

Finally, the parameterizations produced by [LGC*23] are only region-wise smooth, with sharp discontinuities between patches (Figure 6(b, c)). This occurs because each region of the original mesh is mapped to a single coarse triangle, so the prolonged UVs changes abruptly between these patches.

Beltrami Prolongation In Figure 5, we present a quantitative analysis of the symmetric Dirichlet distortion produced by our full parameterization pipeline with Beltrami coefficient prolongation, compared against CM [SPSH*17] and GIF [FW22]. Our approach effectively combines the high quality of CM with the speed and robustness of GIF. As shown in the histograms, vanilla GIF exhibits significantly higher distortion than our method on this dataset, with almost all meshes exceeding a distortion value of 16. This behavior is common for harmonic and conformal methods in the presence of large accumulations of Gaussian curvature, as illustrated in [FW22, Figure 10].

Figure 8 provides further statistical comparisons against GIF and CM. Our method consistently achieves dramatically lower isometric distortion than GIF. Compared with CM, aside from four outliers, all meshes lie in the vicinity of the red line, indicating that our results closely approximate CM. The E_{SD} values of the four outliers are 87.1, 18.6, 12, and 11.4, which stem from local failures rather than a failure to capture the global structure of the ground truth (CM's solution). These issues primarily arise from poor triangle quality in the original meshes, as illustrated in [FW22, Figure 17]. When considering the 99.9% of triangles with the lowest distortion in these four models, the average distortion remains relatively low: 4, 3.4, 4.2, and 2.6. Overall, CM produces the least distorted results, but at a substantially higher cost.

Figure 9 shows the enormous speedup of our runtime compared to the projected Newton optimization of CM [SPSH*17], which optimizes the symmetric Dirichlet energy on the fine mesh. The speedup of our algorithm becomes more substantial as the mesh size increases. For large meshes with more than 1.5M triangles, the speedup of all meshes but 1 is in the range 10–32. For fairness, we loosened the stopping criteria for CM compared to the criteria used in the authors' implementation, to give CM a small performance advantage as we noticed that it had little effect on the final energy reached by CM. With the more strict stopping criteria, CM sometimes doubled the number of iterations and our speedup was even more pronounced. Since our algorithm produces parameterizations with higher distortion than CM, we also include the speedup when CM terminates early, as soon as it reaches our final distortion. This has little effect on the runtime of CM, retaining the dramatic speedups. The total time needed to process the entire dataset with our method is 37 minutes, while CM runs more than 10 hours and 9.3 hours if it is stopped prematurely at the distortion level reached by our method.

We performed an additional experiment in which we used the parameterization generated by our algorithm only as an initialization for CM, which is otherwise initialized with a simple Tutte embedding. This caused CM to converge faster to the same final energy level. The total runtime of this hybrid procedure on the entire dataset was 3.5 hours, which is more than 6.5 hours less than vanilla CM. The two results were virtually indistinguishable.

We also performed an equal-runtime comparison of distortion. Specifically, we stopped CM prematurely after running it for the same amount of time as our method and compared the resulting distortion. While the entire runtime of our method is dominated by 1-2 linear solves of a $|\mathcal{V}|\times|\mathcal{V}|$ matrix, CM's initialization alone involves a linear solve of a $|\mathcal{V}|\times|\mathcal{V}|$ matrix (Tutte's embedding), followed by an additional symbolic factorization of a $2|\mathcal{V}|\times 2|\mathcal{V}|$ matrix (the energy's Hessian). Empirically, for 93.7% of the 252 models in our dataset, CM's initialization was incomplete by the time our method had already converged. At that point, CM's distortion matched Tutte's, which is significantly higher than ours.

Our total runtime can be divided into four main steps, with the average fraction of time per step being 11% for simplification,

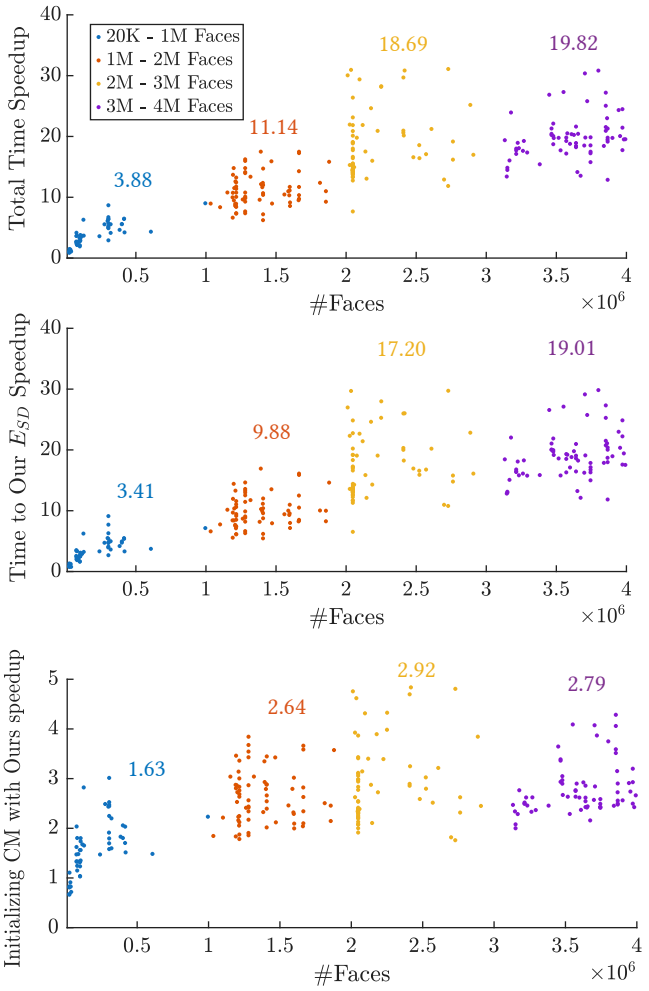


Figure 9: *Runtime speedup with respect to CM across the 252 meshes in our dataset, minimizing the symmetric Dirichlet energy. Each dot represents one mesh. (top) The ratio between the total runtime of CM and ours. The average speedup is noted near each cluster. (middle) Since our algorithm produces parameterizations with higher distortion than CM, we also show the speedup when CM is terminated as soon as it reaches our final distortion. The speed becomes more substantial as mesh size grows. (bottom) We used our results as the initialization for CM rather than the default Tutte embedding initialization. We show the speedup of the overall method (our initialization + CM), compared to vanilla CM. Our initialization speeds up CM by several times, which is significant because the two methods produce indistinguishable results.*

14% for executing CM on the simplified mesh, 4% for prolonging and smoothing the Beltrami coefficients, and 71% for running GIF [FW22]. Figure 10 compares our total runtime to CM when minimizing the symmetric Dirichlet energy. Our algorithm is significantly faster, as its runtime is dominated by GIF, which in turn is dominated by only 1–2 $|\mathcal{V}|\times|\mathcal{V}|$ linear solves, whereas CM requires a $2|\mathcal{V}|\times 2|\mathcal{V}|$ linear solve for each Newton iteration. Moreover, our runtime depends mostly on the mesh size while CM's runtime is much less predictable, since it depends on the number of Newton iterations, which is tightly coupled to the geometry of the mesh.

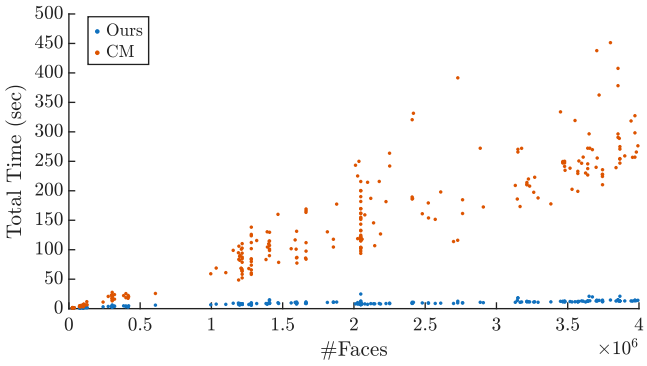


Figure 10: *Runtime comparison on our dataset. Orange dots indicate CM runtime. Blue dots indicate our runtime, which is significantly shorter. Our runtime scales only with the mesh size, as it is dominated by GIF [FW22], whose cost is governed by a linear solver. In contrast, CM's runtime depends both on mesh size and the number of Newton iterations, resulting in substantial variability.*

Figure 1 contains an example of a high resolution scanned ornamented shell containing numerous tubercles and ridges (3,519,558 triangles). Although the model is of high resolution and we simplify to just 35K triangles, we compute a smooth injective parameterization with overall low isometric distortion ($E_{SD} = 2.81$) in a fraction of the time (20.70 seconds) compared to CM. GIF is the fastest, but its parameterization tends to be conformal and has a much higher area distortion ($E_{SD} = 52.04$). CM reaches the lowest distortion of $E_{SD} = 2.71$ in 281 seconds, more than 13.5 times slower than ours. It takes CM 249 seconds to reach our final energy (12.3 times slower). Initializing CM with our result instead of Tutte's embedding reduced CM's overall runtime to 117 seconds, (20.70 seconds of which are consumed by our method), producing virtually the same result 164 seconds faster.

In Figure 2, we present three high-resolution scanned meshes (a crab, a starfish, and a scallop), each containing rich high- and medium-frequency details. Even though the simplified mesh retains only 1% of the original size, our method achieves a dramatic reduction in distortion compared to vanilla GIF.

Finally, Figure 15 shows the results of our method on 7 high-resolution meshes taken from the datasets of [RPPSH17, Thr].

Additional Distortion Measures Our framework was not designed specifically for the projected Newton optimization of CM or for the symmetric Dirichlet energy. It can be applied to other distortion measures for which the distortion of the coarse parameterization represent reliably the distortion of the fine parameterization. To illustrate this, we conducted an additional experiment using E_{SARAP}, the symmetric ARAP energy. Figure 11 compares our method with GIF and CM across our dataset of 252 meshes, minimizing E_{SARAP}. Our approach consistently achieves distortion levels close to CM, significantly outperforming GIF, while also delivering substantial runtime speedups over CM.

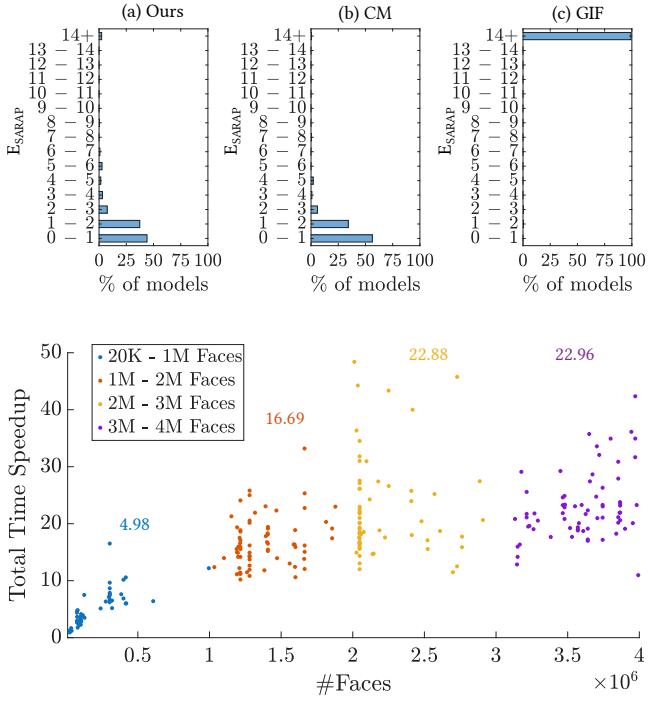


Figure 11: *(top)* E_{SARAP} *distortion distribution of 3 parameterization methods on the entire dataset (252 meshes). Each histogram shows the percentage of meshes in the dataset for different* E_{SARAP} *levels in the range* $[0-14]$. *GIF produces results with significantly high distortion, with almost all the results having* ($E_{SARAP} > 14$). *In contrast, our method produces results with low distortion that more closely resemble those of CM. (bottom) The speedup factors of our runtime compared to CM. The average speedup is noted near each cluster. For meshes with more than 500K triangles, our method is between* $\times 10$ *to* $\times 50$ *times faster.*

Comparison to the Learning Metric Field Method As a data-driven technique, LMF [FW25] requires a pre-trained neural network that has been trained on models similar in nature to those in the test set. The authors trained LMF on a dataset of 30K patches with an average size of 2,658 triangles, which differ from our dataset of real-world objects. Consequently, LMF [FW25] produced relatively high and noncompetitive distortion levels when tested on our dataset of 252 models. To enable a fair comparison, we evaluated both our method and LMF on LMF's dataset, consisting of 2,316 patch-like models, using their published pre-trained network. Figure 13 shows a quantitative analysis of parameterization distortion for our method, LMF, GIF, and CM. As seen in the histograms, the two lowest bars of our method are closest to CM, whereas LMF has 3.7% of models in the 16+ distortion bin, which is empty for our method. In terms of runtime, processing LMF's dataset took 1.77 hours with our method, compared to 2.05 hours for LMF.

Injectivity under Floating-Point Arithmetic Our method guarantees injectivity only under exact arithmetic assumptions, and failures may occur with a floating-point implementation. However, this limitation is also present in the floating-point implementation of any other surface parameterization method. Even the standard Tutte embedding may fail on numerically challenging models [SJZP19, FBRCA23]. Such failures are particularly severe, as they propagate to any downstream method that relies on Tutte initialization, including [SS15, RPPSH17, JSP17, SYLF20], and most notably CM [SPSH*17]. We show that our method is significantly more robust than the standard Tutte (and, consequently, all its downstream methods) under extreme numerical conditions, when considering floating-point implementations.

In contrast to directly applying Tutte on the full-size mesh, our method relies on the standard Tutte embedding only when running CM on a simplified version of the mesh. Applying Tutte to a simplified mesh significantly reduces its susceptibility to numerical instabilities and floating-point rounding errors. This improvement arises because vertex distances increase after simplification, thereby reducing the required numerical precision. Moreover, solving the Laplace equation for the Tutte embedding becomes more numerically stable, since a smaller Laplacian matrix typically has a lower condition number.

Within the modified harmonic subspace, GIF succeeds because the Tutte initialization under the modified metric operates at a far more moderate scale, avoiding the extreme numerical ranges that cause standard Tutte to fail. The Progressive Embedding (PE) method [SJZP19] provides a benchmark dataset of 80 numerically challenging models, constructed by removing a single triangle from sphere-like meshes. This setup makes flattening particularly difficult, as the entire mesh is mapped onto a single triangle, resulting in a boundary with only three vertices. In the continuous setting, this problem effectively corresponds to mapping a sphere onto a puncture, which underlies the severe numerical instability. On this dataset, Tutte fails on all models, and consequently CM also fails in every case. By contrast, our method successfully produces injective parameterizations on 63 out of the 80 models. Achieving injectivity on the majority of this dataset is particularly noteworthy given that Tutte-based methods achieve zero successful embeddings.

Using our default simplified mesh size, Tutte still failed on a subset of these models. To address this, we further reduced the simplified mesh to a fixed size of 1,000 faces, and, for the remaining failure cases, to a constant size of 100 faces.

Finally, since our approach incorporates an energy-minimization process, it produces parameterizations with significantly lower distortion than PE. Figure 12 presents a comparison between our method, PE and CM on the same two representative models shown in [SJZP19, Figure 2].

Robustness to Aggressive Simplification To evaluate the sensitivity of our method to the mesh simplification stage, we conduct an experiment in which we decrease the number of faces in the simplified mesh, to as few as 200 faces (approximately 100 vertices). Figure 14 reports both distortion levels and runtimes for different simplification sizes, evaluated on the models shown in Figure 15.

The x-axis is shown on a logarithmic scale; at each step, the number of faces in the simplified mesh is halved. The rightmost point in each plot corresponds to our method under the default simplification setting. As expected, more aggressive simplification leads

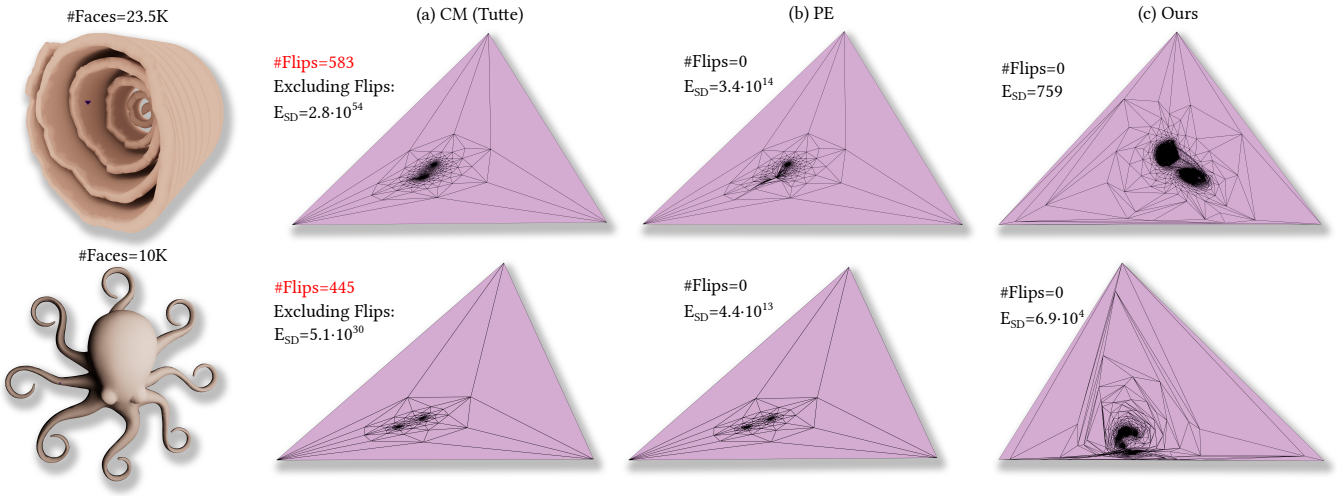


Figure 12: *Comparison on the two representative models shown in [SJZP19, Figure 2]. Each model is produced by removing a single triangle from a sphere-like mesh. (a) Tutte fails to produce injectivity on both models. Consequently any downstream method that relies on it fails as well, particularly CM. As a result, even when excluding the flipped faces, many faces are nearly-collapsed, producing staggering levels of distortion. (b) PE succeeds on both models, but produces enormous distortion levels. (c) Our method produces injective results on both models, with significantly lower distortion than Tutte and PE.*

to an increase in distortion. However, even at extreme simplification levels, the distortion remains moderate and does not escalate to prohibitively large values.

Notably, the runtime remains largely unchanged across all simplification levels. This behavior is explained by the fact that for large models, the dominant computational cost of our pipeline arises from the GIF optimization stage, which is identical in all experiments and therefore unaffected by the size of the simplified mesh. Consequently, aggressive simplification offers no benefit for large models.

5. Summary

We introduced a coarse-to-fine framework for robust and efficient parameterization of disk-like meshes with free boundaries. The key idea is to prolong Beltrami coefficients—rather than UV coordinates—from a parameterization computed on a heavily simplified mesh. This strategy leverages a fast coarse solve, followed by a single harmonic reconstruction on the original high-resolution mesh. The harmonic step is extremely efficient and fully robust with respect to injectivity, yielding low isometric distortion comparable to direct optimization. On large meshes with more than 250K vertices, our method achieves speedups of one to two orders of magnitude. In addition, we proposed a new notion of intrinsic Delaunay triangulation (iDT), formulated purely in terms of cotangent angles, with potential applicability beyond the scope of parameterization.

5.1. Limitations and Future Work

The primary limitation of our method is a modest increase in distortion compared to direct optimization. While this gap is moderate in practice, understanding its roots, and designing improved building blocks to reduce it, is an important direction for future work.

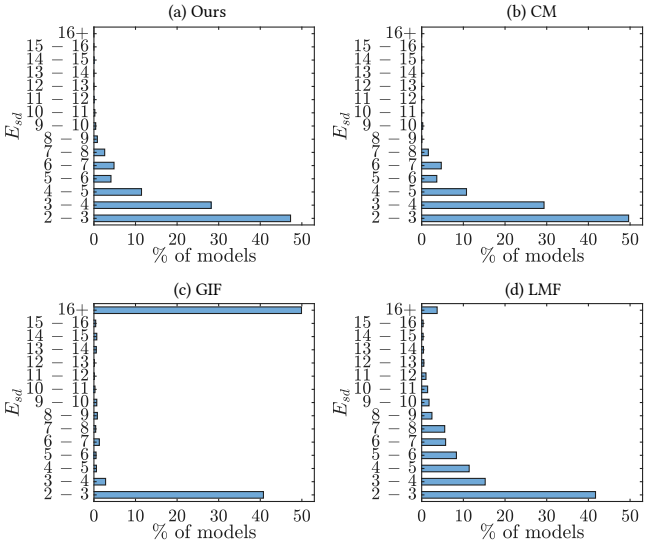


Figure 13: *Distortion distribution of 4 parameterization methods on LMF's dataset. Each histogram shows the percentage of meshes in the dataset for different* E_{SD} *levels in the range* $[2-16]$. *GIF produces results with significantly high distortion, with the majority of results having* ($E_{SD} > 16$). *In contrast, our method and LMF produce results with less distortion, and CM produces the least distortion. However, the 2 lower bars are the closest to CM in our histogram, compared to LMF's histogram. In addition, LMF has 3.7% of the models in the* 16+ *bar, while this bar is empty in our histogram.*

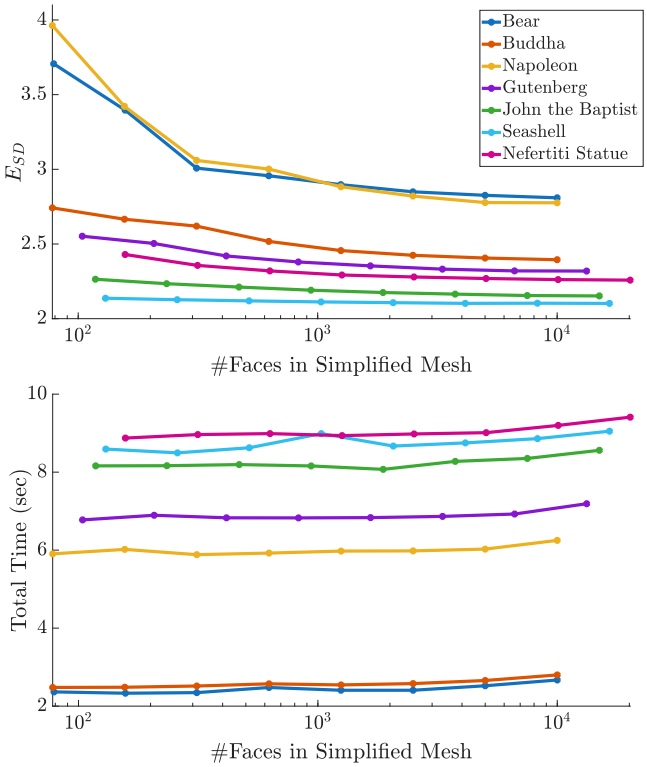


Figure 14: *(top) Distortion levels for progressively more aggressive simplification of the models shown in Figure 15. As expected, distortion increases with more aggressive simplification; however, it remains moderate and does not reach problematic levels. (bottom) Runtimes of our method under increasingly aggressive simplification. The runtime remains largely constant across different simplified mesh sizes, as the primary computational cost comes from GIF, which is unaffected by simplification.*

Promising avenues include distortion-aware simplification strategies and specialized subspaces of Beltrami coefficients that allow direct energy optimization.

Our current prolongation procedure is extrinsic, relying on linear smoothing and simple plane-to-plane parallel transport. This approach only weakly accounts for the geometry of the fine mesh. More sophisticated, geometry-aware or nonlinear prolongation methods could yield further improvements.

The framework is presently restricted to topological disks. Extending it to support more general topologies, cone singularities, or seamless constraints would significantly broaden its applicability.

Finally, although our method guarantees injectivity, it does not directly control the maximum distortion. Incorporating bounded-distortion guarantees, and exploring how our approach could accelerate existing bounded-distortion techniques [Lip12, CLW16], represents another valuable direction for future research.

6. Acknowledgments

This research was supported by the Israel Science Foundation (grant No. 3564/24). We thank the Threedscans team for providing a diverse, high-quality collection of 3D models.

References

[AGK*22] AIGERMAN N., GUPTA K., KIM V. G., CHAUDHURI S., SAITO J., GROUEIX T.: Neural jacobian fields: Learning intrinsic mappings of arbitrary meshes. *ACM Transactions on Graphics 41*, 4 (2022), Article 109, 17 pages. 4

[AL13] AIGERMAN N., LIPMAN Y.: Injective and bounded distortion mappings in 3D. *ACM Transactions on Graphics 32*, 4 (2013), Article 106, 14 pages. 3

[AL15] AIGERMAN N., LIPMAN Y.: Orbifold Tutte embeddings. *ACM Transactions on Graphics 34*, 6 (2015), Article 190, 12 pages. 3

[APL14] AIGERMAN N., PORANNE R., LIPMAN Y.: Lifted bijections for low distortion surface mappings. *ACM Transactions on Graphics 33*, 4 (2014), Article 69, 12 pages. 2

[BCE*13] BOMMES D., CAMPEN M., EBKE H.-C., ALLIEZ P., KOBBELT L.: Integer-grid maps for reliable quad meshing. *ACM Transactions on Graphics 32*, 4 (2013), 98. 2, 4

[BCW17] BRIGHT A., CHIEN E., WEBER O.: Harmonic global parametrization with rational holonomy. *ACM Transactions on Graphics 36*, 4 (2017), Article 89, 15 pages. 4

[BS07] BOBENKO A. I., SPRINGBORN B. A.: A discrete laplace–beltrami operator for simplicial surfaces. *Discrete & Computational Geometry 38*, 4 (2007), 740–756. 8

[CBSS17] CLAICI S., BESSMELTSEV M., SCHAEFER S., SOLOMON J.: Isometry-aware preconditioning for mesh parameterization. *Computer Graphics Forum 36*, 5 (2017), 37–47. 3

[CCS*21] CAMPEN M., CAPOUELLEZ R., SHEN H., ZHU L., PANOZZO D., ZORIN D.: Efficient and robust discrete conformal equivalence with boundary. *ACM Transactions on Graphics 40*, 6 (2021), Article 261, 16 pages. 3

[CCW16] CHIEN E., CHEN R., WEBER O.: Bounded distortion harmonic shape interpolation. *ACM Transactions on Graphics 35*, 4 (2016), Article 105, 15 pages. 3

[CGA24] CGAL PROJECT: CGAL user and reference manual, 2024. URL: https://doc.cgal.org/5.6/Manual/packages.html. 8, 10

[CLW16] CHIEN E., LEVI Z., WEBER O.: Bounded distortion parametrization in the space of metrics. *ACM Transactions on Graphics 35*, 6 (2016), Article 215, 16 pages. 14

[CW15] CHEN R., WEBER O.: Bounded distortion harmonic mappings in the plane. *ACM Transactions on Graphics 34*, 4 (2015), Article 73, 12 pages. 2, 3

[CW17] CHEN R., WEBER O.: GPU-accelerated locally injective shape deformation. *ACM Transactions on Graphics 36*, 6 (2017), Article 214, 13 pages. 3

[CWKBC13] CHEN R., WEBER O., KEREN D., BEN-CHEN M.: Planar shape interpolation with bounded distortion. *ACM Transactions on Graphics 32*, 4 (2013), Article 108, 11 pages. 4

[DAZ*20] DU X., AIGERMAN N., ZHOU Q., KOVALSKY S. Z., YAN Y., KAUFMAN D. M., JU T.: Lifting simplices to find injectivity. *ACM Transactions on Graphics 39*, 4 (2020), 120–1. 2

[DKZ*21] DU X., KAUFMAN D. M., ZHOU Q., KOVALSKY S. Z., YAN Y., AIGERMAN N., JU T.: Optimizing global injectivity for constrained parameterization. *ACM Transactions on Graphics 40*, 6 (2021), 1–18. 2

[DKZ*22] DU X., KAUFMAN D. M., ZHOU Q., KOVALSKY S., YAN Y., AIGERMAN N., JU T.: Isometric energies for recovering injectivity in constrained mapping. In *SIGGRAPH Asia 2022 Conference Papers* (Daegu, Korea, 2022), ACM, pp. Article 36, 9 pages. 2

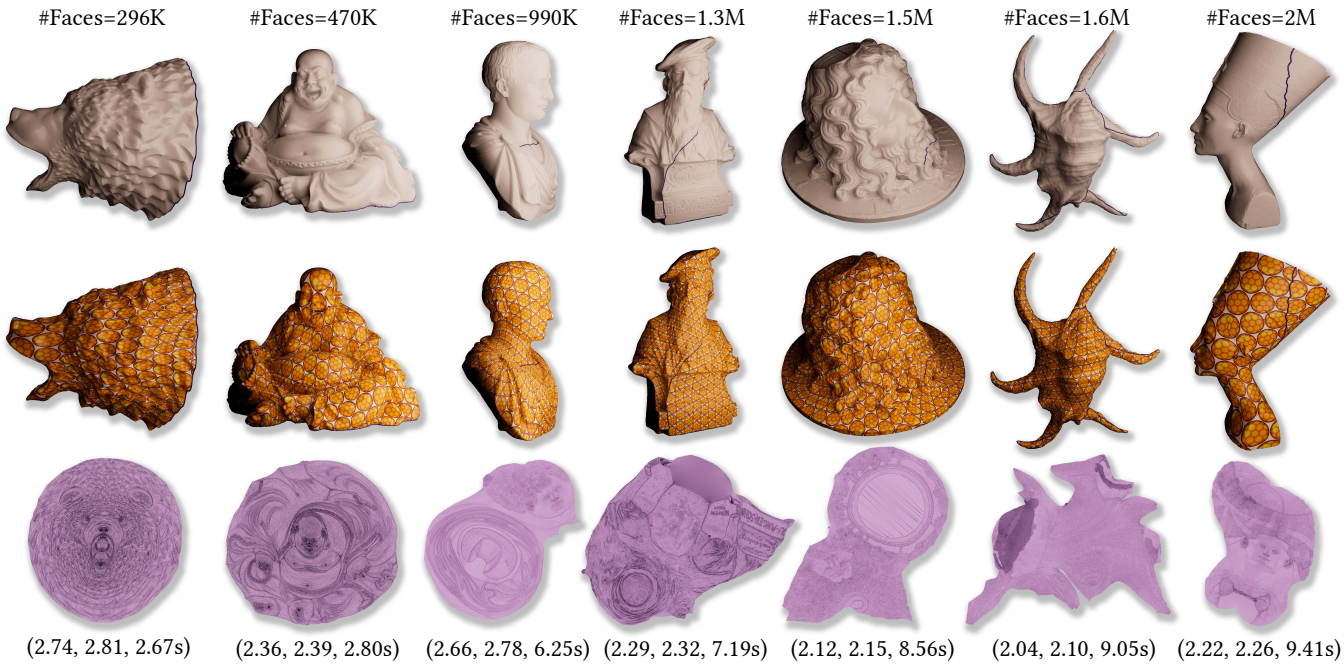


Figure 15: *Additional results of our algorithm applied to 7 high-resolution meshes. In brackets, we have CM's* E_{SD} *for reference, our* E_{SD}*, and our runtime.*

[FBC25] FELDMAN A., BEN-CHEN M.: On planar shape interpolation with logarithmic metric blending. In *Proceedings of SIGGRAPH 2025* (Vancouver, Canada, 2025), ACM. 2, 4, 6

[FBRCA23] FINNENDAHL U., BOGIOKAS D., ROBLES CERVANTES P., ALEXA M.: Efficient embeddings in exact arithmetic. *ACM Transactions on Graphics 42*, 4 (July 2023). 3, 12

[FLG15] FU X.-M., LIU Y., GUO B.: Computing locally injective mappings by advanced mips. *ACM Transactions on Graphics 34*, 4 (2015). 3

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

[FW22] FARGION G., WEBER O.: Globally injective flattening via a reduced harmonic subspace. *ACM Transactions on Graphics 41*, 6 (2022), Article 253, 17 pages. 1, 2, 4, 7, 8, 10, 11

[FW25] FARGION G., WEBER O.: Learning metric fields for fast low-distortion mesh parameterizations. *Computer Graphics Forum 44*, 2 (2025), 1–14. 1, 2, 4, 6, 7, 12

[GGT06] GORTLER S. J., GOTSMAN C., THURSTON D.: Discrete one-forms on meshes and applications to 3D mesh parameterization. *Computer Aided Geometric Design 23*, 2 (2006), 83–112. 3

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), pp. 209–216. 5

[GKK*21] GARANZHA V., KAPORIN I., KUDRYAVTSEVA L., PROTAIS F., RAY N., SOKOLOV D.: Foldover-free maps in 50 lines of code. *ACM Transactions on Graphics 40*, 4 (2021), Article 102, 16 pages. 2

[GSC21] GILLESPIE M., SPRINGBORN B., CRANE K.: Discrete conformal equivalence of polyhedral surfaces. *ACM Transactions on Graphics 40*, 4 (2021), Article 103, 20 pages. 3

[HFCW17] HEFETZ FEDIDA E., CHIEN E., WEBER O.: Fast planar harmonic deformations with alternating tangential projections. *Computer Graphics Forum 36*, 5 (2017), 175–188. Proceedings of Symposium on Geometry Processing 2017. 3

[HFCW19] HEFETZ FEDIDA E., CHIEN E., WEBER O.: A subspace method for fast locally injective harmonic mapping. *Computer Graphics Forum 38*, 2 (2019), 105–119. 2, 4, 7, 8

[HGC99] HORMANN K., GREINER G., CAMPAGNA S.: Hierarchical parametrization of triangulated surfaces. In *Proceedings of Vision, Modeling, and Visualization* (1999), vol. 1999, pp. 219–226. 4, 8

[JSP17] JIANG Z., SCHAEFER S., PANOZZO D.: Simplicial complex augmentation framework for bijective maps. *ACM Transactions on Graphics 36*, 6 (Nov. 2017), Article 186, 9 pages. 2, 4, 12

[KABL14] KOVALSKY S. Z., AIGERMAN N., BASRI R., LIPMAN Y.: Controlling singular values with semidefinite programming. *ACM Transactions on Graphics 33*, 4 (2014), Article 68, 13 pages. 2

[KABL15] KOVALSKY S. Z., AIGERMAN N., BASRI R., LIPMAN Y.: Large-scale bounded distortion mappings. *ACM Transactions on Graphics 34*, 6 (2015), Article 191, 10 pages. 3

[KGL16] KOVALSKY S. Z., GALUN M., LIPMAN Y.: Accelerated quadratic proxy for geometric optimization. *ACM Transactions on Graphics 35*, 4 (2016), Article 134, 11 pages. 3

[KLS13] KUZMIN A., LUISIER M., SCHENK O.: Fast methods for computing selected elements of the green's function in massively parallel nanoelectronic device simulations. In *Euro-Par 2013 Parallel Processing* (Berlin, Heidelberg, 2013), Springer Berlin Heidelberg, pp. 533–544. 4

[LCH*21] LIAO W., CHEN R., HUA Y., LIU L., WEBER O.: Real-time locally injective volumetric deformation. *ACM Transactions on Graphics 40*, 4 (2021), Article 74, 16 pages. 2

[LGC*23] LIU H.-T. D., GILLESPIE M., CHISLETT B., SHARP N., JACOBSON A., CRANE K.: Surface simplification using intrinsic error metrics. *ACM Transactions on Graphics 42*, 4 (2023). 4, 6, 9, 10

[Lip12] LIPMAN Y.: Bounded distortion mapping spaces for triangular meshes. *ACM Transactions on Graphics 31*, 4 (2012), Article 108, 13 pages. 2, 14

[LLT*20] LESCOAT T., LIU H.-T. D., THIERY J.-M., JACOBSON A., BOUBEKEUR T., OVSJANIKOV M.: Spectral mesh simplification. *Computer Graphics Forum 39*, 2 (2020), 315–324. 4

[LSS*98] LEE A., SWELDENS W., SCHRÖDER P., COWSAR L., DOBKIN D.: Maps: Multiresolution adaptive parameterization of surfaces. In *SIGGRAPH 1998 Proceedings* (1998), ACM, pp. 95–104. 4, 8

[LW16] LEVI Z., WEBER O.: On the convexity and feasibility of the bounded distortion harmonic mapping problem. *ACM Transactions on Graphics 35*, 4 (2016), Article 106, 15 pages. 3

[LYNF18] LIU L., YE C., NI R., FU X.-M.: Progressive parameterizations. *ACM Transactions on Graphics 37*, 4 (2018), Article 41, 12 pages. 3, 8

[LZBCJ21] LIU H.-T. D., ZHANG J. E., BEN-CHEN M., JACOBSON A.: Surface multigrid via intrinsic prolongation. *ACM Transactions on Graphics 40*, 4 (2021). 4

[Mes25] MESHLIB: Geometry processing library. https://meshlib.io, 2025. Version 3.0. 5

[Mus97] MUSIN O. R.: Properties of the delaunay triangulation. In *Proceedings of the thirteenth annual symposium on Computational geometry* (1997), pp. 424–426. 8

[MZ13] MYLES A., ZORIN D.: Controlled-distortion constrained global parametrization. *ACM Transactions on Graphics 32*, 4 (2013), Article 105, 14 pages. 2, 3

[Ogd97] OGDEN R. W.: *Non-linear elastic deformations*. Courier Corporation, New York, 1997. 3

[OKN21] OVERBY M., KAUFMAN D., NARAIN R.: Globally injective geometry optimization with non-injective steps. *Computer Graphics Forum 40*, 5 (2021), 111–123. 2

[Pan25] PANUA TECHNOLOGIES: PARDISO 8.2, 2025. URL: https://panua.ch/pardiso. 8

[PL16] PORANNE R., LIPMAN Y.: *Simple Approximations of Planar Deformation Operators*. Tech. rep., ETH Zurich, 2016. 3, 8

[RPPSH17] RABINOVICH M., PORANNE R., PANOZZO D., SORKINE-HORNUNG O.: Scalable locally injective mappings. *ACM Transactions on Graphics 36*, 2 (2017), Article 16, 16 pages. 2, 3, 8, 11, 12

[SC17] SAWHNEY R., CRANE K.: Boundary first flattening. *ACM Transactions on Graphics 37*, 1 (2017), Article 5, 14 pages. 2, 3, 7

[SGC21] SHARP N., GILLESPIE M., CRANE K.: Geometry processing with intrinsic triangulations, 2021. SIGGRAPH 2021 courses. 7

[SGK19] SMITH B., GOES F. D., KIM T.: Analytic eigensystems for isotropic distortion energies. *ACM Transactions on Graphics 38*, 1 (2019). 3

[SJZP19] SHEN H., JIANG Z., ZORIN D., PANOZZO D.: Progressive embedding. *tog 38*, 4 (July 2019). 3, 12, 13

[SKPSH13] SCHÜLLER C., KAVAN L., PANOZZO D., SORKINE-HORNUNG O.: Locally injective mappings. *Computer Graphics Forum 32*, 5 (2013), 125–135. 3

[SLMB05] SHEFFER A., LÉVY B., MOGILNITSKY M., BOGOMYAKOV A.: ABF++: Fast and robust angle based flattening. *ACM Transactions on Graphics 24*, 2 (Apr. 2005), 311–330. 4, 8

[SLS22] STEIN O., LI J., SOLOMON J.: A splitting scheme for flip-free distortion energies. *SIAM Journal on Imaging Sciences 15*, 2 (2022), 925–959. 2

[SPK23] SCHMIDT P., PIEPER D., KOBBELT L.: Surface maps via adaptive triangulations. *Computer Graphics Forum 42*, 2 (2023), 103–117. 4

[SPSH*17] SHTENGEL A., PORANNE R., SORKINE-HORNUNG O., KOVALSKY S. Z., LIPMAN Y.: Geometric optimization via composite majorization. *ACM Transactions on Graphics 36*, 4 (2017), Article 38, 11 pages. 1, 2, 3, 5, 8, 10, 12

[SS15] SMITH J., SCHAEFER S.: Bijective parameterization with free boundaries. *ACM Transactions on Graphics 34*, 4 (2015), Article 70, 9 pages. 3, 8, 12

[SSC18] SOLIMAN Y., SLEPČEV D., CRANE K.: Optimal cone singularities for conformal flattening. *ACM Transactions on Graphics 37*, 4 (2018), Article 105, 17 pages. 2, 3

[SSC19] SHARP N., SOLIMAN Y., CRANE K.: The vector heat method. *ACM Transactions on Graphics (TOG) 38*, 3 (2019), 1–19. 6

[SSP08] SPRINGBORN B., SCHRÖDER P., PINKALL U.: Conformal equivalence of triangle meshes. *ACM Transactions on Graphics 27*, 3 (2008), 77. 3

[SWGL15] SUN J., WU T., GU X., LUO F.: Discrete conformal deformation: Algorithm and experiments. *SIAM Journal on Imaging Sciences 8*, 3 (2015), 1421–1456. 3

[SYLF20] SU J.-P., YE C., LIU L., FU X.-M.: Efficient bijective parameterizations. *ACM Transactions on Graphics 39*, 4 (2020), Article 111, 8 pages. 2, 3, 12

[Thr] THREEDSCANS: Threedscans – free 3d scans archive. https://threedscans.com/. 8, 11

[Tut63] TUTTE W. T.: How to draw a graph. *Proceedings of the London Mathematical Society 3*, 1 (1963), 743–767. 2, 3

[VCKS17] VERBOSIO F., CONINCK A. D., KOUROUNIS D., SCHENK O.: Enhancing the scalability of selected inversion factorization algorithms in genomic prediction. *Journal of Computational Science 22* (2017), 99–108. 4

[WG10] WEBER O., GOTSMAN C.: Controllable conformal maps for shape deformation and interpolation. *ACM Transactions on Graphics 29*, 4 (2010), Article 78, 11 pages. 3

[WGS23] WANG Y., GUO M., SOLOMON J.: Variational quasi-harmonic maps for computing diffeomorphisms. *ACM Transactions on Graphics 42*, 4 (2023), Article 130, 26 pages. 1, 2, 3, 4

[WMZ12] WEBER O., MYLES A., ZORIN D.: Computing extremal quasiconformal maps. *Computer Graphics Forum 31*, 5 (2012), 1679–1689. 2, 4

[WZ14] WEBER O., ZORIN D.: Locally injective parametrization with arbitrary fixed boundaries. *ACM Transactions on Graphics 33*, 4 (2014), Article 75, 12 pages. 3, 4

[XCGL11] XU Y., CHEN R., GOTSMAN C., LIU L.: Embedding a triangular graph within a given boundary. *Computer Aided Geometric Design 28*, 6 (2011), 349–356. 2

[YSLF20] YE C., SU J.-P., LIU L., FU X.-M.: Memory-efficient bijective parameterizations of very-large-scale models. *Computer Graphics Forum 39*, 7 (2020), 1–12. 4

[ZPK18] ZHOU Q.-Y., PARK J., KOLTUN V.: Open3d: A modern library for 3d data processing. *arXiv:1801.09847* (2018). 8