Efficient and Robust Seamless Parametrization

RYAN CAPOUELLEZ, New York University, USA
ALISHER AIKYN, New York University, USA
DENIS ZORIN, New York University, USA

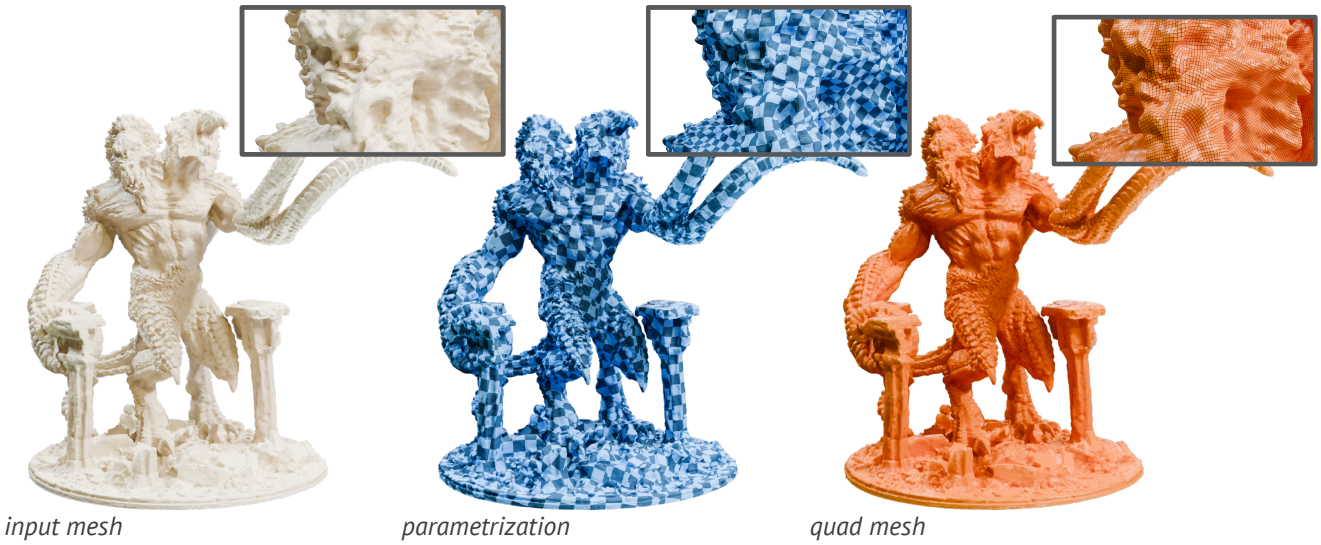


Fig. 1. An example of a scanned high-resolution mesh from Artec3D collection [Artec3D 2025], and corresponding seamless parametrization and fine quad mesh. Our method parametrizes this genus 22 surface with 2,552,268 faces in 963 seconds.

Seamless surface parametrizations support many important geometry processing tasks, including quad meshing, texture mapping, and patch-based surface representations. Recent intrinsic approaches based on Penner coordinates provide a robust mechanism for constructing feasible, seamless parametrizations with prescribed topological structure, determined by a guiding field on a surface and detected feature lines, overcoming many robustness limitations of previous methods. However, these methods rely on highly accurate nonlinear solves, which present a significant obstacle to scalability on large, densely sampled meshes. We address this challenge by observing that Penner-coordinate methods are primarily needed to establish an initial feasible solution whose structure depends on overall guiding field topology and surface topology rather than fine geometric detail, and that high-precision optimization is unnecessary for achieving high-quality results at full resolution, as long as the coarse initial solution can be transferred to the fine levels, while maintaining constraints. We propose a three-scale pipeline that combines an efficient hierarchical approach with robust Penner-coordinate methods. Our method computes guiding fields and holonomy on a medium-resolution mesh, solves the Penner-coordinate feasibility problem efficiently on a coarse mesh that permits connectivity changes, and transfers the resulting structure to the original fine mesh, where only a final, scalable optimization and quantization are performed. Experiments on large benchmark and real-world models demonstrate substantial speedups with no change in robustness in the processing of meshes with millions of triangles.

CCS Concepts: • **Computing methodologies** → **Mesh models**; **Mesh geometry models**.

Additional Key Words and Phrases: parametrization, seamless, cross-field, multiscale, hierarchical, discrete metrics, intrinsic triangulation, Penner coordinates, quadrangulation

ACM Reference Format:
Ryan Capouellez, Alisher Aikyn, and Denis Zorin. 2026. Efficient and Robust Seamless Parametrization. *ACM Trans. Graph.* 45, 6, Article 251 (December 2026), 21 pages. https://doi.org/10.1145/3842536

Authors' Contact Information: Ryan Capouellez, rjc8237@nyu.edu, New York University, USA; Alisher Aikyn, aa13586@nyu.edu, New York University, USA; Denis Zorin, dzorin@cs.nyu.edu, New York University, USA.



This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2026 Copyright held by the owner/author(s).
ACM 1557-7368/2026/12-ART251
https://doi.org/10.1145/3842536

1 Introduction

Seamless parametrizations are an important tool in geometry processing applications that require globally consistent texture coordinates, quad layouts, and conversion to patch-based surface representations. Beyond quad-meshing, seamless parametrization atlases are also an effective representation for high-resolution geometry via textures and displacement maps, where continuity of the parametrization across chart boundaries is highly desirable.

A widely used family of quad-meshing/seamless parametrization methods follows the following overall pipeline: (i) compute a guiding cross-field (typically curvature-aligned in anisotropic regions) and its singularities; (ii) construct an initial *feasible* parametrization

that satisfies local injectivity and seamlessness constraints induced by the field; (iii) improve its quality by minimizing a nonlinear distortion energy while maintaining injectivity; and, optionally (iv) quantize the map to enable tracing integer iso-lines and producing a conforming quadrangulation close to the optimized solution. While this pipeline is conceptually clean and widely used, two of its aspects remain challenging: finding an initial parametrization that simultaneously satisfies seamlessness and local bijectivity constraints (on fixed-connectivity meshes, a solution may not even exist), and scaling it to large meshes.

Recent intrinsic formulations using Penner coordinates provide a robust way to construct seamless parametrizations with prescribed *holonomy signatures*, capturing the cross-field structure, by solving for an intrinsic metric, allowing mesh connectivity changes when necessary. This line of work demonstrates that feasible solutions can be found in essentially all cases, with only known infeasible examples having a restrictively small number of degrees of freedom or cones, or specific configurations of feature constraints that require guiding field singularity changes.

However, the computational profile of these methods presents an obstacle for scaling to large meshes: the Penner-coordinate solve requires a highly accurate nonlinear solution to produce a valid parametrization (the residual measures how far the parametric-domain angles around vertices deviate from 2π). The best-known method to achieve this uses direct sparse solves in the Newton loop, which scale poorly to large meshes and require a lot of memory. Additional complications of the pipeline include constructing a cross-field on the surface, which requires a greedy mixed-integer or another nonlinear solver, as well as the final parametrization optimization, preserving bijectivity, also requiring a nonlinear solver. At the same time, large-scale, densely sampled models may benefit most from semi-regular resampling, as in most cases, connectivity carries little information.

This paper addresses this scaling gap. Our key observation is that Penner-coordinate parametrization is most valuable to find an initial feasible solution for a constrained problem whose structure is primarily determined by the singularities of the cross-field (or more generally, its holonomy structure) and topology of the mesh, not its detailed geometry. This initial solution (or the guiding field needed to produce it) need not be obtained on the full-resolution mesh, as long as the solution and the field can be transferred to the full resolution mesh *while preserving all constraints*. Our second observation is that while the algorithm for obtaining a feasible solution requires a very precise solve, the final mesh quality does *not* require a precise minimization of a global distortion functional, and can be performed relatively inexpensively using scalable iterative solvers.

Based on these observations, we develop a three-scale pipeline that decouples robust initial parametrization computation from resolution. This pipeline resolves the "chicken-and-egg" problem: to construct a coarse-mesh parametrization to propagate to the fine mesh, we need a coarse guiding field that needs to factor the geometry of the fine mesh into consideration. We perform our solves on three meshes, with bijective maps robustly computed between the three: the original, fine mesh, on which only a final optimization is done, a *mid-scale* mesh, capturing large-scale geometry features, to compute the cross-field and extract cones/holonomy reliably, a *coarse* mesh on which we solve the Penner-coordinate system to obtain a feasible seamless parametrization efficiently. Our overall approach is to perform all possible operations on the coarse- and mid-scale meshes, propagating the solution to the fine-scale mesh only for the final quantization and optimization. This strategy removes the major scalability bottlenecks while retaining the robustness benefits of Penner-coordinate feasibility. Key to preserving robustness is our custom fine-to-coarse map construction, which is able to account for changes in the connectivity of the coarse mesh, essential for the robustness of Penner-coordinate methods, while preserving the bijectivity of the mapping.

Contributions. Our contributions include the following:

- A multiscale formulation and overall system architecture for robust seamless parametrization that separates field computation, Penner-coordinate feasibility solves, and final optimization to different resolutions;
- A robust coarsened mesh and bijective fine-to-coarse map construction pipeline based on a combination of techniques for mesh partitioning and overlay mesh construction, capable of handling coarse-mesh connectivity changes;
- An algorithm for the transfer of cross-field topology information between meshes at different resolutions;
- An optimization strategy for the fine-scale mesh parametrization;
- An open-source implementation of the algorithm, to be released at publication time;
- A new dataset of larger meshes for testing parametrization scaling.

Our system is tested on the larger models from a version of a standard dataset (Thingi10k) and on our new benchmark of large-scale models, demonstrating, on models that can be handled by the baseline method, a speedup of up to 87×, and the ability to handle meshes with up to 26 million triangles.

2 Related work

We focus on work most relevant to robust, seamless parametrization and its role in quad-meshing pipelines, as well as prior efforts on large-scale quadrangulation. More comprehensive surveys on parametrization and bijective map optimization can be found in Fu et al. [2021]; Naitsat et al. [2021], and surveys focused on quad meshing and layout construction in Bommes et al. [2013b]; Campen [2017].

Our work builds most directly on many components developed in [Capouellez and Zorin 2024] (Penner-coordinate parametrization), [Eck et al. 1995] and [Klein et al. 1997] (discrete Voronoi coarsening), [Sharp and Crane 2020] (geodesic straightening), [Fisher et al. 2006] (robust intrinsic mesh operations) and [Bommes et al. 2009] (field optimization).

Seamless parametrization and intrinsic feasibility. Seamless parametrizations aim to produce globally consistent transition functions between charts so that integer iso-lines can be traced to form a structured quad layout. Early influential constructions include periodic/global

formulations and branched-covering style parameterizations [Kälberer et al. 2007; Ray et al. 2006]. A recurring practical difficulty is to obtain an *initial feasible* map that simultaneously satisfies seamlessness and local injectivity constraints, especially in the presence of prescribed singularities (cones) and additional constraints.

Our work uses recent algorithms based on Penner coordinates, which construct seamless maps by solving for an intrinsic cone metric with prescribed holonomy/signature and then recovering a planar map [Capouellez and Zorin 2024]. This line of work is closely connected to discrete conformal equivalence methods and uniformization-inspired intrinsic optimization [Ben-Chen et al. 2008; Campen et al. 2021; Gillespie et al. 2021; Kharevych et al. 2006; Springborn et al. 2008], and to earlier intrinsic parametrization frameworks [Sheffer and de Sturler 2001]. Penner-coordinate methods can change intrinsic triangulation during the solve, searching a larger space of metrics with the same set of vertices compared to fixed-connectivity approaches, and, while the guarantees are limited to genus 0 and arbitrary-genus similarity maps, they were demonstrated to succeed on a large dataset derived from Zhou and Jacobson [2016]. However, their computational requirements (high-accuracy nonlinear solves) become a bottleneck on very large meshes [Capouellez and Zorin 2023, 2024]. We note that algorithms with theoretical guarantees for seamless parametrization with prescribed singularities do exist [Campen et al. 2018; Levi 2022, 2023; Shen et al. 2022; Zhou et al. 2020] but are extremely expensive and lead to numerically unstable problems.

Quad-meshing pipelines, quantization, and optimization. Many modern quad-meshing methods follow the same pipeline as our method.

Mixed-integer and integer-grid-map approaches established this pipeline [Bommes et al. 2013a, 2009; Campen et al. 2015]. Subsequent work developed layout/quantization objectives and flow formulations to quantize with minimal deviation from a given map [Coudert-Osmont et al. 2024; Heistermann et al. 2023]. Our method follows the same pipeline, using Penner-coordinate parametrization to find the initial seamless map. We note several works that in various ways simultaneously optimize the parametrization and place singularities, e.g., [Myles and Zorin 2012] and recently [Coiffier and Corman 2023]. These types of methods have not yet reached the robustness level of methods fixing the parametrization topology defined by a cross-field or other means. One advantage of splitting parametrization, topology determination, and parametrization construction is that fields can be manipulated much more easily (e.g., with user-defined or automatically determined constraints), enabling control over parametrization structure if a parametrization can be obtained from any field. Finally, recent approaches [Liang et al. 2025; Pietroni et al. 2021], construct patch layouts based on a guiding field and field line tracing, but do not aim to preserve its topology, or scale these methods to large meshes.

Improving map quality while maintaining injectivity is commonly addressed via locally injective mappings and related barrier-based energies [Lipman 2012; Schüller et al. 2013]. Scalability of these optimization stages has motivated more efficient or reduced formulations [Hefetz et al. 2019; Liu et al. 2018; Rabinovich et al. 2017]. Our final-stage optimization is in this spirit, but we exploit the additional freedom that we do not need high-accuracy energy minimization—only sufficient distortion reduction—once a feasible, seamless map is available.

Coarse domains and multiresolution parameterization. The earliest semi-regular remeshing methods [Eck et al. 1995; Lee et al. 1998] were based on the idea of constructing a coarse domain by simplification or Voronoi partitioning and a map from the fine mesh to this coarse domain. The downside of this approach, which led to the development of field-driven techniques, is that the structure of the domain could take the input geometry into account only in a limited way. Our approach can be viewed as a combination of the ideas from these methods with global seamless parametrization methods.

Other multiresolution frameworks for geometry processing also utilize fine-to-coarse map constructions. Liu et al. [2021] describes a method for mesh coarsening based on edge collapses, which can interpolate functions from the coarse mesh to the fine mesh by composing a sequence of maps associated with the edge collapse operations. However, the mapping is not guaranteed to be bijective, a necessary property for robustly transferring a feasible seamless parametrization from the coarse to fine mesh. Liu et al. [2023] uses a different decimation scheme that does guarantee a bijective mapping in exact arithmetic, but the mapping is not linear in the interior of the faces of the fine mesh. The parametric image of a fine-scale triangle under this fine-to-coarse map may be a nonconvex region, and naively connecting the vertices of the triangle may produce an inverted triangle in the parametric plane. Our construction, in contrast, produces a bijective mapping that is linear within the faces of the fine mesh, which allows us to robustly preserve the local injectivity of the transferred parametrization.

Cross-fields, cone placement and field-guided layout construction. An essential component of our method is a geometry-aligned cross-field construction; we use the basic method of Bommes et al. [2009], but many other methods exist, see the survey in [Vaxman et al. 2016], and related recent work on cone placement [Li et al. 2023]. More recently, machine learning techniques were applied to generate fields and singularities [Dong et al. 2025a,b].

Large-scale quad remeshing and parametrization. The most closely related efforts to ours also aim to make field-aligned quad remeshing practical, similarly combining decimation and multi-scale parametrization transfer. Ebke et al. [2016] present an interactive framework that performs preprocessing on a reduced representation and supports fast updates under user edits; their setting assumes a fixed-connectivity underlying parametrization method at the coarse level and targets interactive refinement. Rather than computing an explicit fine-to-coarse map as we do, the method propagates information from coarse mesh to fine through a decimation log. We have experimented with a related approach but found that it is more prone to introducing large-scale distortion into the initial map, making the final optimization more difficult.

Pietroni et al. [2011] takes a different approach, parametrizing *range image sets* directly, leveraging the acquisition structure to

avoid operating directly on a single large triangulation. It is suitable for constructing lower resolution quadrangulations directly from high-resolution range scans, but at a high level has the same complexity as Bommes et al. [2009].

Fast field-aligned meshing methods (e.g., Huang et al. [2018]; Jakob et al. [2015]) are highly practical for producing quad-dominant or quad meshes at scale, but they do not target the same objective of a globally controlled seamless parametrization structure.

3 Overview

Our method proceeds in several steps, which we summarize informally in this section, providing motivation for our design choices.

Our main goal is to preserve the robustness of the Penner-coordinate methods while creating a framework that allows one to remap the parametrizations it produces to large meshes.

As a way to achieve this goal, we aim to minimize processing on the original high-resolution mesh M, and keep the algorithms that are applied to this mesh or modified meshes at a fine scale arising in the method, primarily local and combinatorial, avoiding dependence on mesh quality.

We use three mesh resolutions in our algorithm: the original fine mesh M, medium-resolution mesh M^m for cross-field construction, and a coarse mesh M^c for parametrization, along with algorithms to transfer quantities (fields and parametrization) between meshes.

Most of the algorithmic steps rely on the construction of mesh embeddings, i.e., pairs of meshes such that the vertices of one are a subset of the vertices of the other, and the edges of one correspond to edge chains on the other. Three key algorithms are used in different parts of the pipeline: a Voronoi-tiling-based algorithm (Section 6.1), to construct coarser meshes M^m and M^c along with maps between fine and coarse meshes, an intrinsic flip algorithm, used for geodesic edge straightening (Section 6.2), and Penner-coordinate parametrization (Section 6.5). None of these algorithms uses geometric information to determine correct connectivity, a feature critical for robustness.

Cross-field construction. The first mesh constructed in the pipeline, M^m, is used to eliminate the bottleneck in constructing the guiding cross-field for the parametrization, as well as reduce the number of singularities and high-frequency direction variations in the resulting field. It can also be used for user-defined input (e.g., marking field constraints), although this is not implemented in our framework.

We note that a specific choice of method for constructing the cross-field is not an essential part of our approach: we used a commonly used algorithm of [Bommes et al. 2009], but any more advanced recent techniques (cf. work mentioned in Section 2) that can generate a field, or at minimum, a set of singularities on M^m, can be used.

The choice to use a medium-resolution mesh for the field is due to balancing three considerations. First, from the performance perspective, the mixed-integer solve for the field can be the bottleneck on a large mesh, so making the mesh for the field as coarse as possible is desirable. Second, if the mesh is too fine, for large-scale high-resolution geometry, aside from computational cost, it may also lead to many spurious singularities capturing small-scale geometry or geometric noise. These singularities make the final quadrangulation output less regular. On the other hand, the mesh needs to be sufficiently fine for the field to capture the shape of the object (e.g., be aligned with the cylindrical parts). While optimizing the tradeoff between these three considerations is not our main focus, we describe the approach we found satisfactory in Section 6.1.

Coarse mesh parametrization. The second mesh, M^c, is needed to eliminate the bottleneck of the high-accuracy nonlinear solve needed by the Penner-coordinate parametrization. We use a coarse mesh, while still including the singularities of the field that determine the parametrization structure, where these accurate solves can be performed inexpensively. Once the coarse mesh parametrization is computed (and the coarse mesh M^c is modified in the process to a new mesh $M^{c'}$, an essential feature of the Penner parametrization algorithm), we transfer it to a refined version of M.

There is again a balance to be struck in the choice of the coarse mesh resolution. Using a mesh that is as coarse as possible minimizes the computational cost of the initial feasible parameterization. However, if the mesh is too coarse, the parameterization that is transferred to the refined version of M will be heavily distorted, and the final optimization of the fine-scale mesh will require more time to reach an acceptable solution. We demonstrate this tradeoff in Section 7 (Figure 23).

While conceptually the process is simple (compute a field on simplified M^m, construct M^c and transfer the cross-field there, parametrize M^c, transfer the parametrization to M), additional complexity arises from the need for moderate additional refinement at every step, needed for the algorithms to guarantee that their output meets the downstream requirements.

The only stages of the algorithm that may suffer from numerical issues are Penner-coordinate parametrization, which was demonstrated to work reliably on a large dataset ([Capouellez et al. 2025; Capouellez and Zorin 2024]), and Tutte's embedding, used in mesh-to-mesh mappings, and well-known to be reliable in the simple setting we consider (a map from a Voronoi region to a single triangle).

Final parametrization. We transfer the field from M^m to the fine mesh, and perform the final optimization starting from this point. As our starting point is already close to the desired solution, the optimization can be performed using efficient and scalable methods. We also use an existing quantization and quadrangulation code to extract a quadrangulation from this parametrization.

Overall, in our pipeline, we follow the approach of robustly constructing a valid solution satisfying all constraints first, and optimizing quality at the end.

4 Background

In this section, we briefly review four underlying techniques to set up the notation for the method.

4.1 Seamless parametrizations

We summarize two views of seamless parametrization, both of which are needed in our method: the parametric plane (uv) coordinate view,

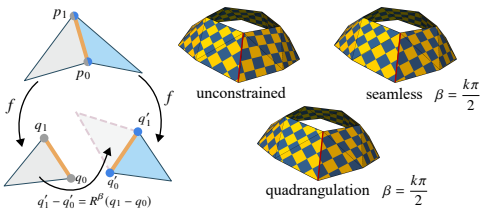


Fig. 2. Seamless parametrization, after [Myles and Zorin 2013].

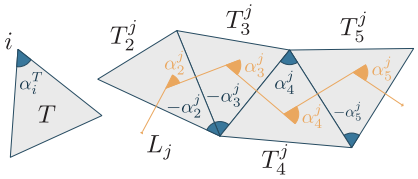


Fig. 3. Holonomy notation, adapted from [Capouellez and Zorin 2024].

and the intrinsic metric view, in terms of parametric lengths and induced angles, and how these two views relate to each other.

Parametric uv coordinate view. To simplify exposition, we assume that the input mesh M does not have a boundary. Let disk(M) be a *cut mesh* of a mesh M obtained by cutting it along a set of edges (*seam* edges) to a topological disk. A *parametrization* of M is an assignment of planar uv coordinates to the vertices of disk(M), defining a piecewise affine continuous map $f : \text{disk}(M) \to \mathbb{R}^2$. We require that f is locally bijective on triangles, i.e., all images of triangles in the parametric domain have the same orientation (no folds). For a seam edge, at least one endpoint has different uv coordinates for two incident triangles. For a *seamless parametrization*, on seam edges (p_0, p_1) shared by triangles T and T' the uv coordinates $q = [u, v]$ satisfy

$$q_1' - q_0' = R^\beta(q_1 - q_0) \tag{1}$$

where q_i, q_i', $i = 0, 1$ are parametric coordinates of p_i in triangles T and T', and R^β is a rotation by an angle $\beta = k\pi/2$ (Figure 2) for an integer k. Informally, in seamless parametrizations, parametric lines $u = \text{const}$ or $v = \text{const}$ continue straight (in parametric domain) across the seams, although u and v may switch if $R^\beta \neq Id$. This is a critical property for constructing quad meshes or seamless textures. From the seamlessness conditions, one can infer that the sum of all parametric angles around a vertex is $k_v\pi/2$ for an integer k_v. Vertices with $k_v \neq 4$, i.e., with angles different from 2π are called *cones*.

The choices of k's on seams are not arbitrary, as these determine the overall structure of the field, typically inferred from a cross-field, as described below.

Intrinsic view. A seamless parametrization, as any parametrization, can also be described in terms of a discrete metric. A *discrete metric* $\mathcal{M} = (M, \ell)$ with connectivity M, is an assignment of positive lengths $\ell : E \to \mathbb{R}^+$ to the edges of the mesh that satisfy the triangle inequality for each triangular face $f \in F$. We can obtain a uv assignment from a metric by using lengths in a breadth-first layout traversal, incrementally assigning uv positions to the vertices of the traversed triangles using assigned parametric lengths. This assignment is highly non-unique: the layout depends on the choice of the BFS root. However, these are all equivalent, as these can be transformed into each other by choosing a different cut for obtaining disk(M), without changing the shape of individual triangles in the parametric domain.

Holonomy signature. Holonomy signature captures the overall structure of the parametrization: which vertices are cones, and how the parametric lines rotate if we traverse a loop on the surface. It can be specified by a collection of angles

$$\{\Theta_i, i = 0 \dots N_c - 1;\ \kappa_s, s = 0 \dots 2g - 1\}$$

where N_c denotes the number of cones and κ_s correspond to a chosen basis of $2g$ homology loops for genus g.

Let α_i^{jk} be the angle of the triangle f_{ijk} at vertex v_i in a mesh M, defined by the metric $\mathcal{M}$. We define the *vertex angle* at vertex $v_i \in V$ to be the sum of angles

$$\sum_{f_{ijk} \ni v_i} \alpha_i^{jk} = \Theta_i \tag{2}$$

For cone vertices, we refer to this angle as the cone angle.

Let the loop γ_s be a sequence of faces sharing sequential edges $e_0^s, \dots, e_{n_s-1}^s$. We assume it to be *locally simple*, i.e., we assume that any three sequential faces are distinct. Then the *holonomy angle* of the loop is defined as

$$\sum_{m=0}^{n_s-1} d_m^s \alpha_m^s = \kappa_s \tag{3}$$

where α_m^s is the angle between e_{m-1}^s and e_m^s, and d_m^s is 1 if the rotation from e_{m-1}^s to e_m^s is counterclockwise and -1 if it is clockwise, and $s = 0 \dots 2g - 1$ (Figure 3). Note that local simplicity ensures that the angles and signs d_m^s are defined unambiguously.

If all angles in the holonomy signature are integer multiples of $\pi/2$, then the metric is seamless, i.e., corresponds to a seamless parametrization.

4.2 Cross-fields

The parametrization gradients $\nabla u, \nabla v$ with respect to the surface metric, as well as the rotations $\beta = k\pi/2$ are often inferred from a *guiding cross-field*. We now describe piecewise-constant cross-fields introduced in Ray et al. [2008] and used in [Bommes et al. 2009]. We reiterate that this is only one of the possible ways to determine the structure of the parametrization.

A cross-field is a quadruple of unit vectors, obtained by rotating a single vector by multiples of $\pi/2$. It is represented by two quantities, an angle $\theta : F \to \mathbb{R}$ assigned to each triangle f, and a *period jump* $p : E \to \mathbb{Z}$ assigned to each edge (Figure 4). The angle $\theta(f)$ is the angle between one of four field directions and a reference edge of f. A period jump $p(e)$ determines corresponding cross-field directions on two triangles sharing the edge e [Bommes et al. 2009]. If directions are numbered $0 \dots 3$, a period jump i means that direction k in triangle 1 corresponds to $(k + i)$ mod 4 in triangle 2. The field period jumps determine the holonomy signature of the parametrization precisely, as we describe below. Two orthogonal

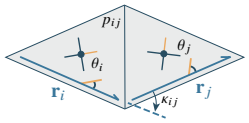


Fig. 4. Cross-field parameters on a triangle.

directions determined by θ on each triangle f are used as a fitting target for the parametrization gradients $\nabla u, \nabla v$ on f, as described in Bommes et al. [2009].

Bommes et al. [2009] also describes an efficient mixed-integer solver that we use to solve the optimization problem.

The field index at a vertex is a direct analog of cones for parametrizations, and the field rotations along locally simple dual loops are analogs of holonomy angles on loops for parametrizations. The period jumps are the field analogs of the rotations R^{β} on seam edges with p_{ij} giving the rotation angle measured in units of $\pi/2$. The rotation of a cross-field direction across an edge e_{ij} shared by faces f_i and f_j is given by

$$\Delta_{ij} = \left(\theta(f_i) + \kappa_{ij} + \frac{\pi}{2}\, p(e_{ij}) - \theta(f_j)\right) \tag{4}$$

where κ_{ij} is the angle between reference directions in f_i and f_j with respect to which θ is measured. The rotation angles on loops can be computed by summing these Δ_{ij} up along dual paths. Note that on closed loops all θ_i cancel, and indices depend only on the reference direction rotations κ_{ij} and period jumps.

The topological structure of the field is determined implicitly by period jumps, and more explicitly by the indices at singularities (equivalently, rotation angles Φ_i of the field on loops around vertices), and a set of rotation angles ψ_j on a system of $2g$ dual loops γ_j which can be selected in the same way as for the parametrization. This is a direct analog of the holonomy signature for the parametrization, so we refer to the set of numbers $\{\Phi_i, i = 0 \dots N_s - 1, \psi_j, j = 0 \dots 2g - 1\}$ as *field holonomy signature*.

In the types of seamless parametrization we consider, we aim to obtain a parametrization with a holonomy signature matching the field holonomy signature for a given field, i.e., with cone angles and loop holonomies determined as $\Theta_i = 2\pi - \Phi_i$, and $\kappa_i = 2\pi - \psi_i$ respectively.

4.3 Penner-coordinate parametrization

We briefly review the intrinsic metric representation underlying Penner-coordinate methods for robust seamless parametrization, as we aim to scale these methods to large meshes.

Let $M = (V, E, F)$ be a mesh with a fixed vertex set V and connectivity, and with a metric $\ell : E \to \mathbb{R}_{>0}$. For a prescribed set of cones and holonomy angles, it may not be possible to find, for a given connectivity, a metric that has the desired holonomy signature, i.e., there may be no parametrization matching the desired cones and gradient rotations on the homology loops. By relaxing the requirement of fixed connectivity, Penner-coordinate methods, originating in the discrete uniformization theory [Bobenko et al. 2015; Gu et al. 2018; Springborn et al. 2008], can find feasible solutions, satisfying holonomy constraints on angles, reliably. This is achieved by considering the space of all metrics with a given vertex set but with arbitrary connectivity, and performing optimization in this space. Remarkably, this space can be parametrized by *Penner coordinates*, which are also an assignment of positive numbers (or, if one switches to logarithms, arbitrary real numbers) to edges on a fixed connectivity, $\ell : E \to \mathbb{R}_{>0}$, but without the constraint of satisfying the triangle inequality. Each such assignment on a fixed connectivity can be remapped, by a Delaunay flip-like algorithm (Weeks' algorithm), to a different connectivity M' with an associated assignment of lengths ℓ', for which M' is Delaunay. This provides a *unique* way of associating a metric with coordinates ℓ on fixed connectivity. We refer to [Capouellez and Zorin 2023, 2024] for details.

From the perspective of this work, the key feature of the Penner-coordinate parametrization is that given an input mesh M, it returns a mesh with the same vertices but *different* connectivity M', and a seamless parametrization of this new mesh M'. With an extra step of constructing an overlay mesh M_1 of M and M', i.e., a mesh which is a refinement of both M and M' (equivalently, M and M' are both *embedded* in M_1 as we define precisely later), we can obtain a parametrization $f : M_1 \to \mathbb{R}^2$.

4.4 Parameterization Optimization

We measure the optimality of a seamless parameterization with a weighted sum of field alignment [Bommes et al. 2009] and symmetric Dirichlet objectives [Smith and Schaefer 2015]:

$$E = \int_S w_{SD} E_{SD}(q) + (1 - w_{SD}) E_{FA}(q) dS \tag{5}$$

where $q_i = (u_i, v_i)$ are the parametric coordinates of a vertex i of disk(M_3), with linear seamless constraints applied. The first term is $E_{SD} = \|F_q\|_F^2 (1 + (\det F_q)^{-2})$, where $\|\cdot\|_F$ denotes the Frobenius norm, and F_q is the linear map from the 3D mesh to the parametric plane defined by the positions of points q_i. The second term is $E_{FA} = \|F_q - F^{trg}\|_F^2$, where F^{trg} denotes the target per-triangle linear map, inferred from two orthogonal vectors of the cross-field, interpreted as the gradients of u and v components of the map respectively.

Note that both terms are non-dimensional. We use $w_{SD} = 10^{-3}$, as the symmetric Dirichlet term is primarily used as a barrier function to ensure that the initial locally bijective parametrization retains this property.

5 Method

Most of the steps of our algorithm deal with meshes of different resolutions and the need to transfer data (cross-field, parametrization) between these. The detailed summary of the algorithmic steps is shown in Table 1, and Figure 6, with the main meshes involved illustrated in Figure 7.

We transfer data between meshes when the edge graph of one is *embedded* in the other. (By a slight abuse of terminology, we will say that the mesh itself is embedded.)

A particular example of embedding is the overlay mesh of two meshes A and B with the same vertex set, with each embedded in the overlay. We define embedding more formally as follows.

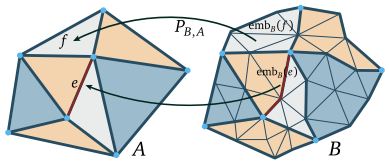


Fig. 5. Embedded meshes: the coarse mesh A on the left is embedded in B. The embeddings $\text{emb}_B(f)$ of a face f and $\text{emb}_B(e)$ of an edge e are shown. The map $P_{B,A}$ maps $\text{emb}_B(f)$ to f.

Mesh embeddings. A mesh A is *embedded* in a mesh B (denoted $A \hookrightarrow B$), if there are maps emb_B, defined for all faces and edges, and the following conditions hold:

- The vertex set of A is a subset of the vertex set of B.
- For each edge e of A, there exists a non–self–intersecting chain of edges $\text{emb}_B(e)$ in B connecting the endpoints of e.
- No two such edge chains intersect except at the endpoints.
- B is partitioned into domains by the edge chains $\text{emb}_B(e)$ each with 3 vertices of A on the boundary.

For each face $f = (i, j, k)$ of A, the three edge chains $\text{emb}_B(e_{ij})$, $\text{emb}_B(e_{jk})$, and $\text{emb}_B(e_{ki})$ bound a simply connected submesh of B, which we denote by $\text{emb}_B(f)$. These embedding maps are illustrated in Figure 5.

Embedded mesh mappings. If a mesh A is embedded in a mesh B, we define $P_{B,A}$ to be a piecewise linear (PL) map from B to $|A|$. Here $|A|$ denotes a reference surface constructed from a set of reference triangles T^{ref}, one for each face of A, glued along edges corresponding to shared edges in A. Each triangle T^{ref} is a copy of the triangle with vertices $(0, 0)$, $(1, 0)$, $(0, 1)$.

The map $P_{B,A}$ is defined on each submesh $\text{emb}_B(f^A)$. On such a submesh, the vertices of B are assigned barycentric coordinates with respect to f^A. These assignments are required to be consistent across shared edges, so that for any two adjacent faces f_1^A and f_2^A of A, the induced mappings agree along their common edge.

To pass data (fields and parametrizations) between meshes, we use the maps $P_{B,A}$ associated with mesh embeddings: if A is embedded in B, we can transfer a field or parametrization between these. Several meshes are used as inputs and outputs of algorithmic stages (Figures 6 and 7):

- The original mesh M;
- The mid-scale mesh M^m, for computing the cross-field, embedded in a refinement M_1 of M.
- The coarse-scale mesh M^c used for Penner parametrization, embedded in a different refinement M_2 of M and parametrization output $M^{c'}$, a refinement of M^c.
- Associated cross-fields C^m on M^m, C^c on M^c, and C on M_3, a refinement of M_2.

We obtain coarse meshes M^m and M^c by subsampling vertices of M, constructing an approximate Voronoi partition of M with respect to these vertices and then connecting them with paths dual to Voronoi cell boundary edges (see details in Section 6.1). In general it may not be possible to approximate all edges of M^m or M^c by chains of edges of M (e.g., it is possible for a vertex to have higher valence in M^c than in M). M_1 and M_2 define refinements of M in which M^m and M^c can be embedded. A further refinement M_3 of M_2 is needed to embed additional edges arising from the connectivity changes of M^c produced by Penner-coordinate methods.

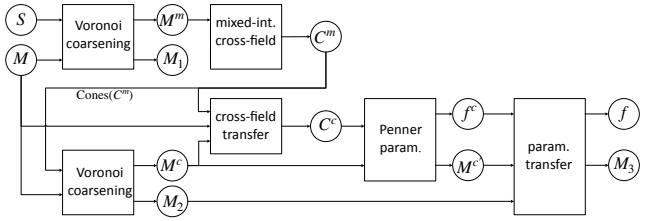


Fig. 6. Algorithm pipeline for the initial parametrization. Data (meshes, cross-fields, and parametrizations) are shown in circles, algorithmic blocks in rectangles. M_1 and M_2 are needed for the cross-field transfer from M^m to M^c, connection not shown.

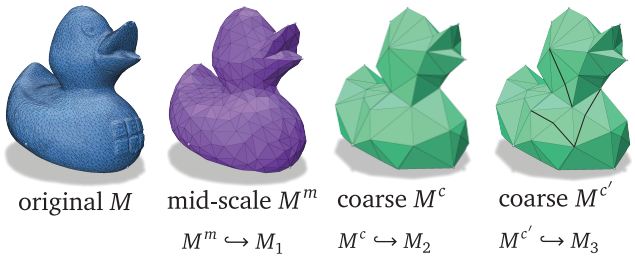


Fig. 7. Meshes used in the algorithm. M_1, M_2, and M_3 are refinements of M, in which M^m, M^c and $M^{c'}$ are embedded. In addition, M_3 is a refinement of M_2.

6 Algorithmic details

6.1 Voronoi coarsening

We use a variant of the method of [Klein et al. 1997] to create an approximate Voronoi tiling of our surface, to construct meshes M^m and M^c. Given a mesh A and an initial set of seed vertices V_s^{init}, the method constructs a Voronoi tiling consisting of tiles $\text{Vor}(v)$, for seeds $v \in V_s$, which includes V_s^{init}, and then traces edge paths E_{ij} connecting vertices v_i and v_j in V_s, possibly refining some edges to keep these disjoint away from endpoints. This results in a refinement of A, which we call A_1, and a new triangle mesh B, with vertex set V_s, and edges corresponding to the edge chains E_{ij} traced between vertices, with A_1 partitioned into domains corresponding to faces of B, i.e., defining an embedding emb_{A_1}.

We use different choices of V_s^{init} for M^c and M^m. For the coarse mesh M^c, we use the set of cones $\text{Cone}(C^m)$ of the cross-field on M^m, which ensures that the coarsened mesh contains these cone vertices. For the mid-scale mesh M^m, which is constructed before a cone set has been determined, we use a single arbitrary vertex v_0 as the initial seed set, with the rest placed adaptively until a target maximal tile size is reached.

For completeness, we recap the algorithm briefly. For meshes without boundary, the algorithm enforces the following requirements:

Table 1. Main steps of the algorithm

Algorithm step	Input	Output	Description	Described in
Cross-field construction				
Mid-scale mesh construction for the cross-field	Input mesh M	Refinement M_1 of M, mid-scale mesh $M^m \hookrightarrow M_1$, the map P_{M_1,M^m}	Compute M^m for cross-field construction, using the discrete Voronoi algorithm	Section 6.1 and 6.3
Cross-field optimization	M^m	Cross-field $C^m = (\theta^m, p^m)$ on M^m	Compute a cross-field on M^m using greedy mixed-integer cross-field optimization.	[Bommes et al. 2009]
Field transfer to the fine mesh	$M^m \hookrightarrow M_1$, $M \hookrightarrow M_1$, C^m	Cross-field $C = (\theta, p)$ on M	Remap angles θ^m and period jumps p^m from M^m to M_1 and then transfer to M using the embedding $M \hookrightarrow M_1$.	Section 6.4
Coarse mesh construction and parametrization				
Coarse mesh construction from cones	M, cone set Cone(C)	Refinement M_2 of M, coarse mesh $M^c \hookrightarrow M_2$, map P_{M_2,M^c}	Construct the coarse mesh M^c for Penner parametrization, using the discrete Voronoi algorithm with geodesic straightening.	Section 6.1
Field transfer to the coarse mesh	$M \hookrightarrow M_2$, $M^c \hookrightarrow M_2$, C	Cross-field C^c on M^c	Transfer the field C to the coarse mesh M^c using embedding $M^c \hookrightarrow M_2$	Section 6.4
Penner-coordinate parametrization	M^c, C^c	Refinement $M^{c'}$ of M^c, parametrization $f^c : \text{disk}(M^{c'}) \to \mathbb{R}^2$	Compute a field-aligned seamless parametrization f^c matching the holonomy signature of C^c.	[Capouellez and Zorin 2024]
Parametrization transfer to the fine mesh and optimization				
Mapping modified coarse mesh to the fine mesh	$M^{c'} \hookrightarrow M^c$, $M^c \hookrightarrow M_2$	Refinement M_3 of M_2 with $M^{c'} \hookrightarrow M_3$	Since Penner-coordinate parametrization changes connectivity from $M^c \hookrightarrow M_2$ to $M^{c'}$, trace edges of $M^{c'}$ on M_2 to get M_3.	Section 6.5
Parametrization transfer to the fine mesh	$M^{c'} \hookrightarrow M_3$, coarse param. f^c	Initial parametrization $f : \text{disk}(M_3) \to \mathbb{R}^2$	Use $P_{M_3,M^{c'}}$ to extend f^c from $M^{c'}$ to M_3.	Section 6.3
Refined mesh quality improvement	$M \hookrightarrow M_3$, param. f	Improved mesh M_3', parameterization f'	Improve the quality of the refined mesh M_3 in 3D and the initial parameterization to improve convergence of the optimization.	Section 6.3, Section 6.6
Final optimization	M_3', C, parametrization f'	Final optimized parametrization f^{final} on M_3'	Run the iterative solver to optimize the symmetric Dirichlet energy combined with field-alignment terms, with f' as a starting point.	Section 6.7

(1) Every tile is homeomorphic to a disk;
(2) No more than 3 tiles meet at a vertex;
(3) The intersection of two adjacent tiles is a single connected path.

These requirements are needed to ensure that the dual of this partition is a surface triangulation.

The algorithm is vertex-based, i.e., the seed of each tiling is a vertex, and tilings are composed of *derived neighborhoods of vertices* (Figure 8). These neighborhoods are defined as unions of quads in all incident triangles, connecting the vertex, two midpoints on incident edges, and the triangle centroid. Boundaries between tiles are paths of dual edges connecting face centroids. This means that Condition 2 is automatically satisfied (centroid degree is at most 3). The algorithm uses multi-source Dijkstra with lengths as weights to assign vertices to tiles, checking condition 1, and making a vertex a seed vertex, instead of adding it to a tile, if it results in a violation.

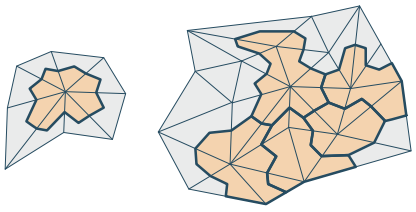


Fig. 8. Derived neighborhood of a vertex, and a collection of such neighborhoods forming a Voronoi cell.

After all vertices have been assigned to a tile, Condition 3 is checked. If it is violated, additional vertices adjacent to the offending tile intersections are added as seeds until it is satisfied. Additionally, if two adjacent tile intersections constitute less than 10% of the length of the tile perimeter, a vertex adjacent to these intersections is also added as a seed. This is a heuristic to improve the aspect ratio of the coarse triangulation.

Adaptive sampling. If the set of seed vertices V_s^{init} is too sparse, then the coarsening produced by this method may not sample the surface sufficiently densely to capture the overall shape. By adding the following additional requirement, we can control the density of the coarsened vertex set:

(4) No vertex is more than ℓ_{max} away from the nearest seed

We iteratively add vertices that do not meet this requirement to the seed set until all vertices are sufficiently close to a seed. In our experiments, we use

$$\ell_{\max} = \sqrt{\frac{A}{\rho \frac{\sqrt{3}}{2}|V|}},$$

where A is the area of the mesh and $\rho \in (0, 1]$ is a target ratio of vertices between the coarse and original mesh. In order to make our simplification adaptive and to sample vertices more densely in curved regions than flat regions, we replace the standard Euclidean lengths with the embedded Gauss map lengths of [Kovacs et al. 2010]. To be precise, we use

$$\ell_{ij} = \|\mathbf{p}_j - \mathbf{p}_i\| + \alpha\|\mathbf{n}_j - \mathbf{n}_i\|$$

where $\mathbf{p}_i$ is the position of vertex v_i and $\mathbf{n}_i$ is the vertex normal. We use $\alpha = \ell_{\text{avg}}$, the average edge length on the mesh.

Dual construction. [Eck et al. 1995] constructs the dual of the Voronoi partition by mapping the Voronoi regions to the plane, and tracing straight lines there. We take a different approach, tracing edge paths on the mesh directly, and then rectifying these paths, to avoid the extra mapping step as well as the need to compute edge-line intersections, which may create slivers. To construct the initial dual (Delaunay) triangulation, we construct edge chains E_{ij} between the Voronoi tile seeds v_i and v_j for tiles sharing a boundary segment. We first find the vertex v_m on the Voronoi tile boundary edge between $\text{Vor}(v_i)$ and $\text{Vor}(v_j)$ with minimal distance to v_i and v_j. We then trace the shortest path between v_i and v_m in $\text{Vor}(v_i)$ and the shortest path between v_m and v_j in $\text{Vor}(v_j)$. We ensure that the traced paths are disjoint in each Voronoi tile, so that the resulting paths partition the mesh into triangular domains (Figure 10).

This property is obtained in the following way. Let D_v be the set of vertices in the Delaunay triangulation, which is the same as the set of Voronoi seeds V_s, and let D_e be the set of edges in the Delaunay triangulation. To ensure that we can always find a path between vertices that does not intersect already traced paths, after tracing each Delaunay edge path, we refine any edges $e_{ij} \notin D_e$ such that $v_i, v_j \in D_v$. Then a path always exists between any Voronoi tile seed v_i and any point on the boundary of $\text{Vor}(v_i)$.

After the initial tracing of dual edge paths, we remove the original Voronoi domain boundaries and retrace the dual path without the constraint of staying within the same Voronoi patch for each half, while maintaining the non-intersection constraint (Figure 9).

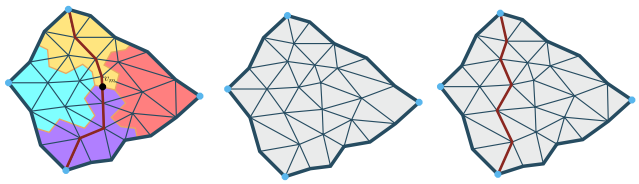


Fig. 9. Retracing a Delaunay edge while maintaining non-intersection constraints. Left: the initial traced path must pass through the midpoint vertex v_m on the Voronoi tile boundaries, creating unnecessary distortion. Middle: a single traced path is removed, while leaving the other paths fixed. Right: a new, shorter path is traced that is still non-intersecting and topologically equivalent to the original path

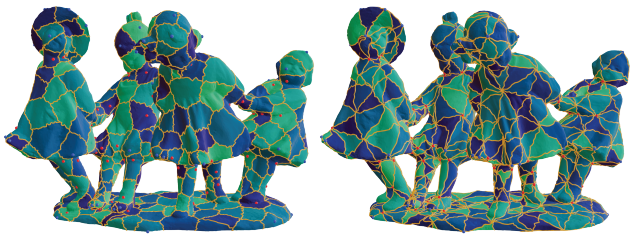


Fig. 10. Voronoi partition of a mesh, with seeds at cones of the cross-field C, and the dual triangulation.

Simplification. Once the Delaunay paths are traced, we remove the unneeded edges we inserted. More specifically,

(1) Remove all segments that are not parts of original edges, and that can be removed without changing the mesh genus and manifoldness.
(2) Remove all degree one vertices, and their adjacent edges.
(3) Remove degree two vertices inserted in faces of the original mesh A, unless this leads to a degenerate triangle.

After these edges and vertices are removed, each polygon of the resulting mesh is triangulated.

Mid-scale and coarse mesh. The only differences between applying this algorithm to the construction of M^m and M^c are in how the seed vertices are initialized and the target vertex density ρ.

For the mid-scale mesh M^m, we use a single arbitrary vertex v_0 as the seed set V_s^{init}, and we set the target vertex ratio as $\rho = 0.1$ to produce a moderately dense sampling that captures the shape of the surface.

For the coarse mesh M^c, the initial set of seeds is the set of cones $\text{Cone}(C^m)$ of the cross-field on M^m, and we set $\rho = 0.02$. This produces a coarser sampling that is guaranteed to contain the cone vertices, which is necessary in order to transfer the guiding cross-field to this mesh.

6.2 Geodesic straightening

The quality of our coarse-level parametrization f^c on M^c, and as a consequence, the initial fine parametrization f^{init} that needs to be optimized at the final stage, depends on how well the coarse mesh approximates the fine mesh, and, related to this, how much distortion is introduced by the fine-to-coarse map $P_{M_3,M^{c'}}$. This is largely determined by the placement of cones and the edge chains we trace in the construction of M^c. Since the edges of the coarse triangles are mapped to straight lines in the parametric domain, we expect the distortion to be minimized if the coarse triangle paths are also straight lines, i.e., geodesics, on the mesh surface. In order to straighten the paths, we adapt an algorithm of Sharp and Crane [2020] to our setting (Figure 11). This work describes an algorithm to find geodesics isotopic to an input path just by flipping edges of the mesh.

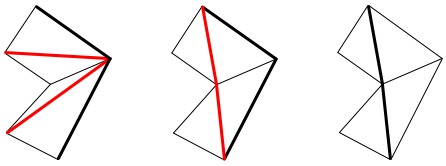


Fig. 11. FlipOut algorithm for geodesic straightening [Sharp and Crane 2020]. In the fan between two black edges at a vertex, red edges are flipped to create a shorter path between the endpoints of these edges.

Our algorithm is based on the idea of the original method, but changes it in two important ways. First, the signpost data structure, used in the algorithm, depends on floating-point operations that can create incorrect connectivity due to numerical problems, especially on bad-quality meshes. Second, while the method is guaranteed to preserve the isotopy class of the input paths, the geodesics are allowed to tangentially intersect other vertices and paths of the curve network when it is not possible to further straighten the path without crossing another edge transversally.

To address the first issue, as well as to be able to maintain the embedding of the coarse meshes we construct in a refinement of the original mesh, we use the overlay mesh data structure and edge flip algorithm of [Fisher et al. 2006]. The overlay mesh is a representation of two embeddings. In the case of the Voronoi mesh coarsening described above, the original mesh B, and the transformed mesh B' with the same vertex set, obtained from B by flips, both embedded in the overlay mesh A'_1, are modified with every flip. Initially, $B = B'$, and $A'_1 = A_1$. A key consequence of allowing only flip operations is that in the intrinsic metric on the original mesh, all flipped edges remain straight: the intrinsic flips do not change the metric. This makes it possible to develop an algorithm that determines the sequence of edges of B that are traversed by the flipped edge, not relying on floating-point computations. We use this robust algorithm for edge flips, to modify both the embedded coarse mesh B, and the mesh A_1 it is embedded in.

To address the second issue and ensure that edge chains corresponding to the edges of A' do not intersect, as a part of straightening, other than at vertices, we prohibit any operation that would result in an intersection. While the result will not be a geodesic, it is still guaranteed to be a shorter curve than the initial path. The result of this straightening method is illustrated in Figure 12.

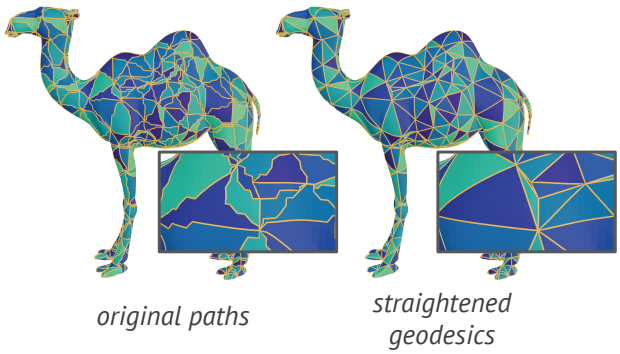


Fig. 12. M^m embedded in M_1 before and after geodesic straightening.

6.3 Mappings of embedded meshes

If a mesh A is embedded in a finer mesh B, it is straightforward to construct piecewise-affine maps from faces of B to A, forming a bijective continuous map from B to A. Each map $P_{B,A}(f^B)$ for a face f^B is a map to the reference triangle $T^{\text{ref}} = [(0,0),(1,0),(0,1)]$. All vertices v_i of the edge chain $\text{emb}_B(e^A)$ for an edge e^A are placed uniformly on the edges of T^{ref}, and fixed. For vertices v of triangles f^B of B, belonging to faces f^A with $\text{parent}(f^B) = f^A$ but not on edges, the map is determined by Tutte's embedding with uniform weights (we use uniform weights for robustness, although this increases the initial map distortion). We consider the submesh $B(f^A)$ of B that corresponds to f^A, and solve the linear system for the positions of vertices of $B(f^A)$ that are not on the boundary, which is guaranteed to be bijective after some additional edge refinement to ensure that no interior edge of $B(f^A)$ connects two boundary vertices with the same parent edge e^A. This yields a map to T^{ref}, i.e., barycentric coordinates of the vertices of $B(f^A)$ on f^A. Due to the consistent choice of positions on edges, these maps form a continuous map on all of B. Note that this map only depends on the connectivity of f^B, which results in numerical robustness even for near-degenerate triangles. While it also results in higher average distortion than a more geometrically aware map, this distortion is local and is quickly removed by the final optimization.

Additionally, we use Tutte's embeddings on "micro-scale" to improve the quality of the 3D triangles of the refinement M_3 of M, at the end of the process: the points of M_3 that are on edges of M are distributed uniformly, and the barycentric coordinates of the points interior to faces of M are updated using Tutte's maps. Typically, there are very few such vertices on each face of M, and while this step is done on a fine mesh, only a small fraction of faces require these adjustments, and the overall cost is relatively low. The goal of this step is to improve the convergence of the optimization at the last step of the algorithm, as it is affected by the 3D triangle quality.

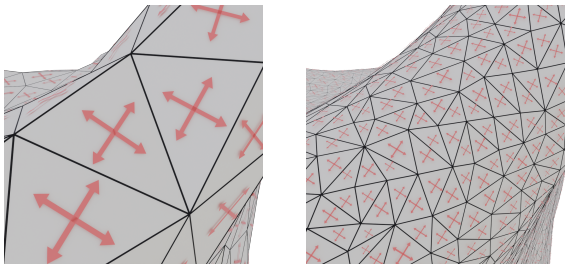


Fig. 13. Field on a mid-scale mesh M^m and the transfer to the fine mesh.

6.4 Field transfer

We need to transfer the field $C^m = (\theta^m, p^m)$ defined on M^m, to the coarse mesh M^c and the fine mesh M_3. An example of transfer is shown in Figure 13. We consider two transfer operations, from a coarse mesh to a fine mesh and from fine to coarse mesh, and then clarify how these are combined to implement the transfers we need in the algorithm.

We consider two meshes A and B, with the coarser mesh A embedded in B, and the corresponding map $P_{B,A}$, with triangles of B mapped to faces of A. We further assume that the singularities of the field θ_B defined on B, are at vertices also present in A.

There are two parts to this task: transferring the period jumps p and the angles θ.

Transfer of period jumps. Period jump transfer from a coarse mesh A embedded in a fine mesh B is straightforward: we assign a period jump of a coarse edge e^A to its set of child edges e_i^B, with $\text{parent}(e_i^B) = e^A$.

The main challenge is in the fine-to-coarse transfer – note that we define the field on M^m, which is in general finer than M^c.

Instead of transferring period jumps, we transfer the field holonomy signature. We use the equivalence between period jumps for a fixed choice of the seam cut, and holonomy signatures for a fixed choice of loops, with the additional constraint that period jumps are non-zero at seams. This equivalence is given by the zippering algorithm of Ray et al. [2008], which describes how to convert a set of indices of singularities and rotation angles on a system of loops to period jumps and vice versa.

The transfer of singularity rotation angles, as we assume that the singular vertices are also present in the coarse mesh A, is direct. It remains to identify loops on a coarse mesh A that can be assigned the rotation angles γ_i of the field on B.

Let $f_1^B, ..., f_n^B$ be the dual loop on the fine mesh B. Let $f_i^A = \text{parent}(f_i^B)$. The dual loop $f_0^A, ..., f_{n-1}^A$ is already a dual loop on the surface, but it is not locally simple, as required by the definition of the rotation angles.

To make it locally simple, we first remove any repeated elements $f_i^A = f_{i+1}^A$ from the loop. We then iteratively remove any cusps, i.e., subsequences $f_i^A, f_{i+1}^A, f_{i+2}^A$ where $f_i^A = f_{i+2}^A$. Once all cusps are removed, the resulting loop is locally simple. Once we have identified the equivalent loops on the coarse surface A, we use the zippering algorithm to recover the period jumps on A.

Angle transfer. For coarse-to-fine, the angle transfer is again straightforward; we project the direction in $\mathbb{R}^3$ defined by θ^A on a single triangle f^A of A from the plane of f^A to the planes of the child faces f_i^B with $\text{parent}(f^B) = f^A$.

For fine-to-coarse, we need to ensure that the correspondences between the four field directions are consistent with the period jumps on both the fine and coarse mesh. To do this, we first ensure that the period jumps are trivial on a dual spanning tree of A, which is always possible using local period jump modifications without changing the topological structure of the field [Bommes et al. 2009]. We then choose an arbitrary face of the fine mesh, and we determine the corresponding choice of direction in all other faces using a breadth-first traversal that *does not* cross any edge e^B where the period jump of $\text{parent}(e^B)$ is nontrivial. The corresponding directions from all children of f^A are then projected to the tangent plane of f^A and averaged. Akin to Bommes et al. [2009], we also apply several iterations of local Gauss-Seidel iterations to reduce the sum of squared rotations (4) across the mesh edges with the fixed transferred period jumps, which quickly smooths sampling artifacts in the directions.

The initial field computed on M^m is transferred first to M_1 in which M^m is embedded (a simple coarse-to-fine transfer), then to M, also embedded in M_1. As M_2 is a refinement of M, another coarse-to-fine transfer yields a field on M_2 in which M^c is embedded. The final fine-to-coarse transfer maps it from M_2 to M^c. We note that if we construct M_2 as a refinement of M_1 (i.e., construct the Voronoi partition for M^c on M_1 instead of M) we could avoid one of fine-to-coarse transfers, at the expense of some additional refinement of M_2.

6.5 Parametrization, and Parametrization Transfer

Parametrization. We perform the Penner-coordinate parametrization as described in [Capouellez and Zorin 2024]. The algorithm computes a parametrization on a mesh obtained from the original mesh M^c by flips; the same intrinsic flip algorithm on the overlay mesh, as used above in geodesic straightening, is used to obtain the parametrization of the refinement $M^{c'}$ of the original mesh. We emphasize that the topological guarantees of this algorithm, which are only conditional on the convergence of a Newton method with strong empirical performance, are a key to the robustness of the overall pipeline.

We also optimize the parameterization distortion energy E on $M^{c'}$ using the post-processing optimization described in Capouellez et al. [2025] to align the parameterization with the transferred field C^c and to reduce parametric distortion. Since this optimization occurs at the coarse level, we can use Newton's method with a direct solver. We find that this initial coarse optimization, which serves as a simple hierarchical method in the spirit of Aksoylu et al. [2005] and Sheffer et al. [2005], substantially improves the convergence of the final fine-scale optimization.

Parametrization transfer. Once the coarse parametrization f^c is computed on $M^{c'}$, we need to transfer to M, or rather, its refined version M_3. The main difficulty here is that we start with a mesh M^c embedded in M_2 (so we could easily transfer from M^c to M_2 using P_{M_2,M^c}), but Penner-coordinate parametrization necessarily

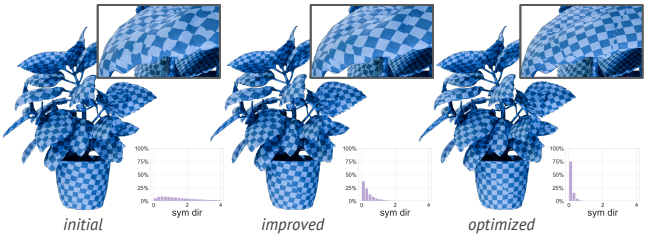


Fig. 14. Initial parametrization transferred from a coarse mesh M^c for complex geometry/topology may be quite distorted, but the distortion tends to be local, rather than, e.g., large-scale smooth variation of scale, and is quickly eliminated by the local uv optimization during the mesh improvement. The final optimization then further reduces the distortion and improves the global field alignment.

changes connectivity of M^c to $M^{c'}$, embedded in M^c. In contrast with coarsening methods based on mesh decimation, with our embedded mesh construction it is straightforward to propagate these connectivity changes to our fine-to-coarse mapping. We simply use the maps P_{M_2,M^c} to insert edges of $M^{c'}$ that are not edges of M^c into M_3.

Suppose w_{ij} is a vertex of $M^{c'}$ inserted by the parametrization algorithm on the edge e_{ij} of M^c with barycentric coordinate s on the edge. We parametrize the edge chain $\mathrm{emb}_{M_2}(e_{ij})$ of edges of M_2 corresponding to e_{ij} on the interval $[0, 1]$. We choose an edge e on the path such that the point with coordinate s along the path is contained in e. We split this edge and insert a new vertex v corresponding to w_{ij}. After all vertices of $M^{c'}$ which are not in M^c are inserted into M_2, this refinement of M_2 is triangulated, and the shortest edge paths corresponding to the edges of $M^{c'}$ which are not in M^c are traced. We constrain these paths to be non-intersecting. If such an edge of $M^{c'}$ is contained in a face f of M^c we make sure that the path stays in $\mathrm{emb}_{M_2}(f)$, the set of faces of M_2 corresponding to f. We insert vertices if needed, as in Section 6.1. These newly inserted edge chains are then also straightened using the algorithm of Section 6.2, and the excess refinement is removed, using the simplification algorithm described in Section 6.1. The resulting refinement of M_2 is the mesh M_3, on which the parametrization will be defined.

6.6 Mesh improvement

The parameterization transferred from the coarse mesh $M^{c'}$ usually has high distortion due to the maps $P_{M_3,M^{c'}}$, and the refinement often introduces short edges and bad-aspect-ratio triangles (Figure 14). To improve the quality of our parameterized meshes, we use local mesh improvement to simultaneously improve the mesh triangle quality in the 3D surface and in the parameterization.

We first remove, whenever possible, the inserted vertices in M_3 using the refinement simplification method of Capouellez and Zorin [2023]. We then apply the following local mesh operations:

(1) Collapse edges smaller than ℓ_{collapse};
(2) Split edges larger than ℓ_{split};
(3) Swap nearly planar edges with dihedral angle within θ_{swap} of 180° if it improves the mesh quality.

We use $\ell_{\text{collapse}} = 10^{-2} \cdot \ell_{\text{avg}}$, $\ell_{\text{split}} = 10 \cdot \ell_{\text{avg}}$, and $\theta_{\text{swap}} = 2^\circ$.

Along with standard checks to make sure that these operations do not change the topology of the surface, we also reject edge collapse and swaps that increase the following quality measure:

$$Q = \sum_{f \in F} \frac{r_f^{\text{circ}}}{r_f^{\text{in}}}$$

where r_f^{circ} is the circumscribed circle radius and r_f^{in} is the inscribed circle radius of face f. Note that this measure approaches infinity if any triangle degenerates. We measure the triangle quality in both the 3D and parametric geometry, and we reject the operation if it increases either.

Seamless constraint preservation. An additional challenge in our setting is the need to preserve the seamless constraints in the parameterization. We reject any operation that would result in an inverted element in the plane. We also prohibit edge collapses between two cone vertices. Before swapping an edge on the parametrization seam, we modify the seam connectivity to avoid the edge, and we reassign the uv coordinates for the adjacent triangles using a local layout procedure.

Local distortion optimization. To remove the local parametric distortion, we also apply a local coordinate descent procedure to iteratively optimize the distortion energy E for the parametric coordinates of a single vertex, while keeping all other vertices fixed. We use a simple Newton backtracking line search to ensure that the new vertex position decreases the energy and does not introduce inverted parametric triangles. For vertices away from the seam of the parameterization, this is a straightforward optimization of two uv coordinates. For vertices on the seam, we eliminate the seamless constraints, which for non-cone vertices always results in two degrees of freedom. We skip the optimization for cones.

We apply a fixed number of rounds (we use five) of collapses, splits, swaps, and local vertex uv optimization.

6.7 Final optimization and quadrangulation

While the coarse parameterization optimization and local mesh improvement reduce E substantially, additional global optimization may be needed to reach high parametrization quality. We generally follow the optimization approach described in Appendix D of Capouellez et al. [2025], which uses Newton's method with a backtracking line search, but with several key changes to render this approach scalable.

Compared to [Capouellez et al. 2025], one important change is using an iterative solver (PCG) with Jacobi preconditioner, instead of a direct solver, for computing our Newton descent directions. Additionally, we project the Hessian at each iteration to be positive semidefinite by clamping the negative eigenvalues of the local per-face Hessians of the symmetric Dirichlet energy. We found that preconditioned CG is a good match for the task, as the final optimization does not require finding the minimum of energy exactly (this is in contrast to the Penner-coordinate solve, which requires exact solution for edge lengths, with at least 10^{-10} accuracy in most cases). The most significant aspect of the optimization affecting visual quality, as well as quantitative distortion of highly-distorted triangles, is the removal of local variations of uv coordinates, due to

the remappings $P_{M^{c'},M_3}$, i.e., elimination of high-frequency errors in the approximate solution that were not removed by the local mesh improvement. Iterative solvers reduce this type of error efficiently.

Iterative solvers for symmetric positive-definite systems are well-known to be more stable compared to indefinite or non-symmetric matrices. For this reason, we eliminate the constraints explicitly rather than solving the indefinite system obtained by enforcing constraints with Lagrange multipliers.

We eliminate the constraints (1) for the seam parametric positions. To achieve this, we mark each seam *uv* position as dependent or independent, and for the dependent ones, compute the vector of coefficients expressing them in terms of independent degrees of freedom. The marking is done by traversing the boundary of disk(M_3), which includes two sides of each edge, and whenever three of the four degrees of freedom on a seam edge (q_0, q_1, q_0', q_1') are already visited and marked as independent or dependent, it is marked as dependent, otherwise as independent. One exception is the nodes of the seam where more than two seam edges meet, which we always mark as independent – a heuristic we found to help to reduce the number of degrees of freedom substantially. When we mark a degree of freedom as dependent, we compute its vector of coefficients based on the vectors of other dependent vertices present at the edge, and the constraint (1) associated with the edge. At the end of this process, we still have some constraints not eliminated, as the number of edges on the boundary of disk(M_3) is $2m$ where m is the number of seam edges, and the number of constraints is m, exactly m *uv* positions should be marked as dependent, which in general does not hold, as we skip the nodes. The final elimination of the remaining constraints is done by QR, which we found to be very fast, as only a few dependencies are left.

Another key difference from [Capouellez et al. 2025] is that we alternate the Newton line steps with rounds of the local per-vertex optimization (Section 6.6). The per-vertex optimizations can cheaply remove local artifacts caused by the global Newton steps without precise convergence to a local minima.

We terminate our optimization once the energy stops changing significantly, i.e., the relative decrease is less than 10^{-2} per step over multiple steps (we use 3). This is similar to the approach taken in other hierarchical methods, e.g., Liu et al. [2018] and Meyer et al. [2026], where the primary goal is a practical reduction in distortion, not convergence to a precise local minimum.

Finally, we use the algorithms of Campen et al. [2015], [Ebke et al. 2013], and Heistermann et al. [2023] to generate fine quadrangulations. These algorithms start with our optimized seamless parametrization and quantize it (we use the average edge length for quantization, as our focus is not on creating conforming layouts with large patches). As a consequence, the quantized parametrization and the quad mesh itself stay very close to our initial parametrization. The steps of the current implementation, provided by the authors, are the same as described in [Capouellez et al. 2025].

7 Evaluation

Datasets. We use two datasets in our evaluation. The first one is derived from the widely used Thingi10k dataset, processed in the same way as in [Capouellez et al. 2025], consisting of 17,421 models. However, the number of faces in most models in Thingi10k is relatively small, and we primarily use it to validate the robustness of our approach and test scalability using artificially refined data.

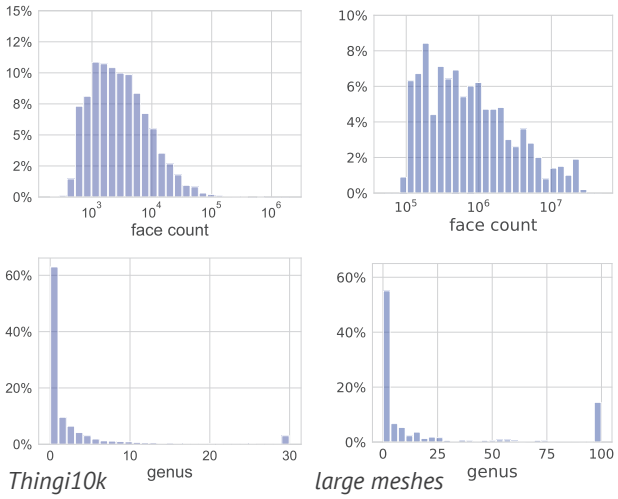


Fig. 15. Dataset statistics for Thingi10k and our large mesh dataset. Outliers are aggregated in the rightmost bin. 145 meshes in the large mesh dataset have genus above 100, with a maximum value of 33,472.

A second dataset of larger models was collected from freely available online 3D scanned models, provided under the Creative Commons license, by Artec3D [Artec3D 2025], Smithsonian Institution [Smithsonian Institution 2025], and AIM@Shape repository v5.0 [Falcidieno 2004]. When possible, we used the meshes obtained from these repositories without modification. However, many of these scanned models are non-manifold or contain degenerate triangles. We remesh surfaces that are not closed manifold surfaces or with a minimum corner angle less than 10^{-4} using Hu et al. [2020], preserving the original resolution as closely as possible. After separating all meshes into connected manifold components and discarding components with fewer than 100,000 faces, we obtained 1,000 meshes, with the largest having 26+ million faces. The full collection of large meshes is shown in Figure 29, and basic statistics on both datasets are shown in Figure 15. Timings for our evaluation were measured on cluster nodes with Intel Xeon Platinum 8592+, with 128 GB of RAM and up to eight threads.

Field Generation. We compute fields for all models in our datasets on the mid-scale mesh M^m using the method of Bommes et al. [2009], which is then transferred to the original mesh. We compare this approach with applying the method of Bommes et al. [2009] directly on the fine mesh M (Figure 16) with a time out of 6 hours. Solving for the field on the mid-scale mesh results in a significant speedup, and also, on average, a smaller set of cones. Nevertheless, our curvature adaptive mid-scale mesh still captures salient geometric features of the surface (Figure 17).

Performance improvements. Our method is targeting larger meshes, with performance improvements from moderate to large-scale models. We compare our parameterization method to the Penner-coordinate

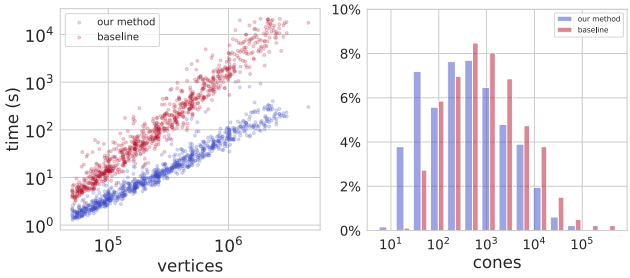


Fig. 16. Runtime and cone counts of our field generation and transfer from M^m compared with the baseline field generation directly on M. Our method achieves an average 15× speedup over the fine mesh generation, with a 45× speedup on meshes with over one million vertices.

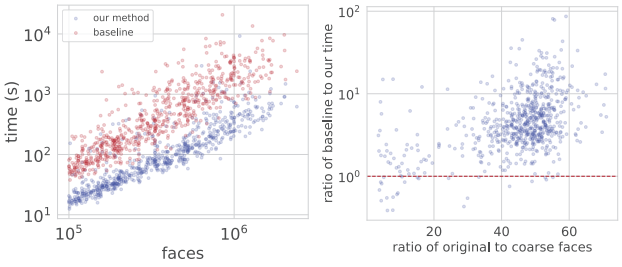


Fig. 18. Comparison of runtimes between our method and the baseline on a subset of our large mesh dataset. On average, our method achieves a 6.4× speedup over the baseline, with more significant improvements for surfaces that can be simplified substantially.

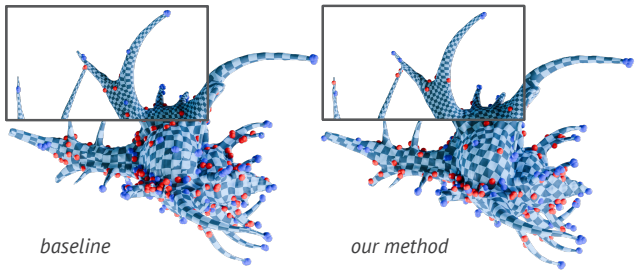


Fig. 17. Left: parametrization for a field produced on the original fine mesh (419 cones). Right: parametrization for a field produced by our method on a medium resolution mesh (316 cones). Note that our method correctly places cones at the tips of the spikes, and results in a parametrization aligned with the principal curvature directions on the cylindrical regions of the surface.

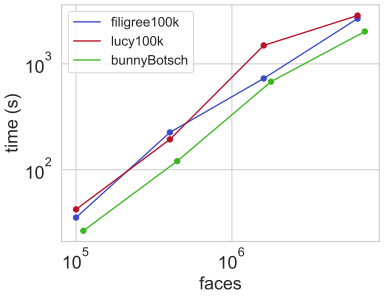


Fig. 19. Scaling of runtime for 3 representative meshes vs. resolution (finer resolutions obtained by subdivision). Observe the variation in the runtime due to differences in complexity: filigree100k mesh has high topological and geometric complexity, lucy100k has relatively low topological complexity, but fine detail, and the bunny has lower complexity both for topology and geometry.

parameterization of Capouellez and Zorin [2024] combined with the final optimization algorithm described in the appendix of Capouellez et al. [2025] using a direct solver. Since the difficulty of the parameterization task depends heavily on the topology of the field and on the choice of optimization energy, we run this baseline on the same fields transferred from M^m as our method. We also use the same energy E and the same relative decrease stopping criteria. We select for comparison the 678 meshes from the large mesh dataset where the baseline method did not run out of memory and successfully obtained an optimized parameterization within 6 hours.

On this subset, our method achieves an average a 6.4× speedup (Figure 18). The baseline Penner-coordinate parameterization needed to obtain a feasible initial solution on the fine mesh alone takes on average 3.9× longer than our entire method. We observe that the speedup varies considerably, with up to a 87× improvement, and in rare cases a slowdown of up to 2.6×. To a significant extent, it is determined by the balance between coarse and fine-scale complexity of the models. As an example, a model with an extremely complex topology and a relatively low polygon/genus ratio will not be accelerated a lot, as the topology-preserving coarse-level mesh has to be large. On the other hand, a model with a relatively simple overall shape and low genus, but with a lot of small-scale geometric detail, as is common for scanned objects, sees more significant speedups.

Figure 19 demonstrates how our method scales on meshes with the same geometric complexity but higher resolution, with finer meshes of the surface obtained by subdivision.

Parametrization quality. We also compare the quality of our results to this baseline method. The distribution of symmetric Dirichlet values in the parameterizations produced by our method is nearly identical (Figure 20). The qualitative appearance of the parameterizations we produce is also comparable to the optimized baseline (Figure 21).

Since the termination condition may affect the quality of the optimized parameterization, for these comparisons we use the original stopping criteria from Capouellez et al. [2025] for the baseline uv optimization ($\|\nabla E\|_\infty < 10^{-4}$, or up to 500 iterations). We emphasize that, *for our method*, we use the same relative decrease stopping criteria as used for the runtimes reported in Figure 18. We observe that the original stopping criteria for the baseline results in a significantly longer runtime than the relative stopping condition, but

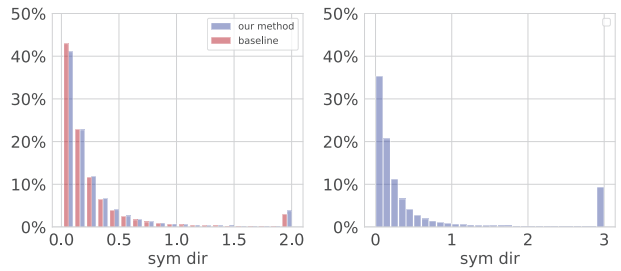


Fig. 20. Left: Comparison of the distribution of per-face symmetric Dirichlet values between our method and the baseline on a subset of the large mesh dataset, with outliers aggregated in the rightmost bin. Right: Distribution of per-face symmetric Dirichlet values of our method on the full dataset, which contains larger and generally more challenging models, e.g., with meshing artifacts in undersampled regions.

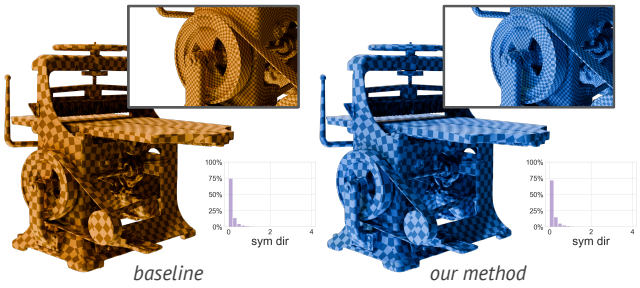


Fig. 21. Left: the parametrization produced by the baseline method in 3,637 seconds. Right: the parametrization produced by our method in 509 seconds. Note the comparable scale and alignment of the uv-isolines.

the difference in the distribution of symmetric Dirichlet values it produces is negligible.

Further examples of the parametrizations produced by our method are shown in Figures 26, 27, and 28, as well as the extracted quad meshes. For each model, we report the distribution of the per-face symmetric Dirichlet energy and the average angle deviation of the quadrangulation from 90°.

Scalability. Figure 22 shows the runtime of our method on the full large dataset of 1,000 meshes, as well as the breakdown by algorithmic step. Note that the bottleneck of our method is now the final optimization on the fine mesh, while obtaining the initial feasible solution is comparatively cheap. We also observe that, although substantial refinement of M may be needed to produce the mesh M_3 on which the parametrization is initially defined, the final amount of refinement is modest after applying the simplification method of Capouellez and Zorin [2023]. The average increase in the number of faces for the parametrized mesh is 1.0%.

Parameters. For most models, using the lowest resolution mesh (i.e., with most vertices being cones) is adequate to obtain a feasible initial seamless parameterization. However, we observe that very coarse meshes result in extremely high distortion that requires more iterations of the final optimization to resolve, in which case the

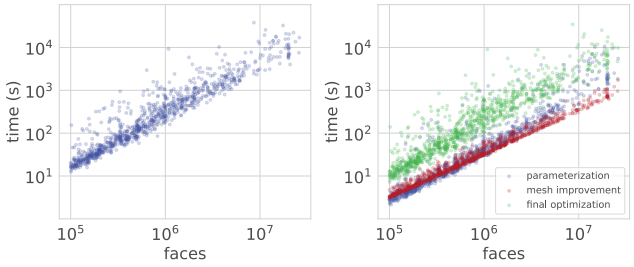


Fig. 22. Left: total runtime; Right: runtime by phase.

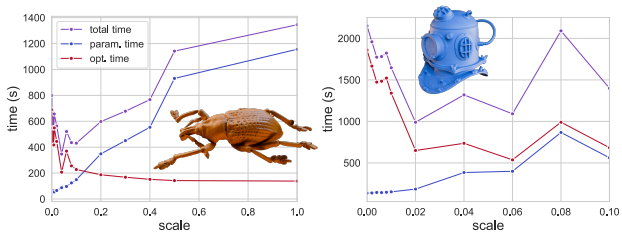


Fig. 23. Coarse-scale mesh resolution parameter ρ impact on performance. While not surprising, the parameterization time increases rapidly with the coarse mesh size; however, if the scale is too coarse, we observe an increase in the optimization time, due to higher distortion of the parametrization f^c, and the need for more solver iterations.

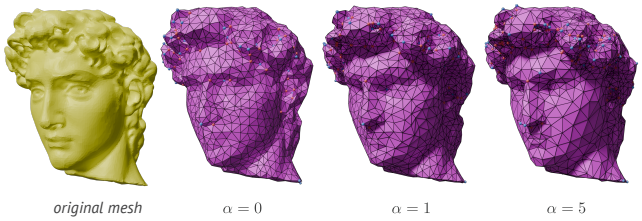


Fig. 24. Impact of parameter α in mid-scale map construction. Each coarse mesh has approximately the same number of faces, but with higher values of α the vertices are concentrated at high curvature regions.

overall performance degrades. Figure 23 demonstrates the tradeoff in the choice of ρ between parameterization time and optimization time, motivating the use of a small but nonzero value.

Figure 24 shows the impact on varying α on the mid-scale mesh construction. We find $\alpha = 1$ to give good balance between regularity and curvature adaptivity.

Robustness. We evaluate robustness by verifying that the initial parametrization can be consistently obtained for a large variety of meshes. We have run on the Thingi10k-derived dataset with 100% success rate.

On the large mesh dataset, we obtained parameterizations for all but five meshes using our default parameters. Four of these meshes had numerically degenerate inverted parametric triangles due to numerical issues in the remapping. However, we were able



Fig. 25. Failure cases for our method. Left: a genus 33,472 surface with 519,144 cones in the target field. Right: a surface with numerous meshing artifacts due to undersampled regions.

to successfully parameterize all four of these meshes by increasing the resolution parameter ρ for the coarse mesh. We observe that these meshes contain numerous meshing artifacts (Figure 25, Right), which results in a challenging field topology and a greater chance of numerical problems in Tutte's mapping. The remaining failure case is a genus 33,472 surface with 519,144 cones (Figure 25, Left), which, even with the coarsest triangulation possible for this vertex set, is too large for a direct solver.

Relation to other methods. Our work builds on [Capouellez and Zorin 2024] with a focus on improving performance, to enable parametrization of large meshes with the same guarantees, and inheriting its robustness, as demonstrated by running it on the Thingi10k-derived dataset. The relation to previous work, in terms of robustness, is identical to the discussion in [Capouellez and Zorin 2024] and [Capouellez et al. 2025], which we briefly repeat here.

Among older methods, the method of Myles et al. [2014] succeeds in producing a seamless parametrization (along with T-mesh partition) for all shapes in a small set of 120 meshes, but needed to add cones or change the holonomy signature in about 30% of the cases. In contrast, our method, as the Penner-coordinate method, succeeds on 100% of a 17,421 mesh dataset. While the method for obtaining the initial parametrization discussed there is fast, the final optimization, as implemented, required solving constrained cone problems. That method was compared to the earlier work [Bommes et al. 2009], and [Bommes et al. 2013a] on which [Ebke et al. 2016] is based, which did not find a valid solution in 25% of cases. Another recent method with theoretical guarantees Shen et al. [2022] is not robust numerically, not finding solutions for 6% of the [Myles et al. 2014] cases. A more recent method, Pietroni et al. [2021] is tested on another Thingi10k-derived dataset of 9,877 models. It did not produce output on about 0.5% of meshes, and does not necessarily preserve the input parametrization topology structure. A more recent method, similar in spirit [Liang et al. 2025], demonstrates a higher degree of robustness. The papers Pietroni et al. [2021] and [Liang et al. 2025] do not aim to make the methods scalable or reproduce fine surface detail, with greater focus on mechanical models.

8 Limitations and Conclusion

Our method demonstrates that Penner-coordinate parametrization can be extended to large-scale meshes, without sacrificing the robustness of the original method, and achieving similar quality. We have demonstrated its performance and scalability on a diverse set of models of medium- to large scale. There are several factors that limit the current version of the method and present possible directions for future work.

- In its current form, the method does not handle meshes with boundaries; while it primarily targets meshes obtained by reconstructing from measured data (images, range scans, or volumetric imaging), and many reconstruction methods produce closed surfaces, in the case of partial data, it is preferable not to mesh the artificially added parts that make the surface closed. While no significant changes are needed to handle boundaries or sharp features, a number of adjustments need to be made to the coarsening, straightening, and Penner-coordinate parametrization algorithms (cf. treatment of boundaries via reflection [Campen et al. 2021]).
- The Tutte's mapping using a floating-point solve is the only remaining source of potential non-robustness of the method; this can be further improved by using, e.g., Finnendahl et al. [2023].
- Significant speedups may be possible to achieve in some cases by further improvements to the iterative solver for handling the high-distortion triangles, as well as systematic parallelization of all stages, similar to [Ebke et al. 2016].
- The method is inherently topology preserving, and high-resolution scanned meshes often contain topology artifacts. Combining it with a topology simplification approach is a promising direction.

Acknowledgments

This work was supported in part through the NYU IT High Performance Computing resources, services, and staff expertise. This work was also supported by the NSF grant No 2411349. We would further like to thank Jiacheng Dai for assistance in creating diagrams.

References

Burak Aksoylu, Andrei Khodakovsky, and Peter Schröder. 2005. Multilevel Solvers for Unstructured Surface Meshes. *SIAM Journal on Scientific Computing* 26, 4 (2005), 1146–1165. arXiv:https://doi.org/10.1137/S1064827503430138 doi:10.1137/S1064827503430138

Artec3D. 2025. *3D Models Scanned using Artec3D technology.* https://www.artec3d.com/3d-models/

Mirela Ben-Chen, Craig Gotsman, and Guy Bunin. 2008. Conformal Flattening by Curvature Prescription and Metric Scaling. *Computer Graphics Forum* 27, 2 (2008).

Alexander I Bobenko, Ulrich Pinkall, and Boris A Springborn. 2015. Discrete conformal maps and ideal hyperbolic polyhedra. *Geometry & Topology* 19, 4 (2015), 2155–2215.

David Bommes, Marcel Campen, Hans-Christian Ebke, Pierre Alliez, and Leif Kobbelt. 2013a. Integer-grid maps for reliable quad meshing. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–12.

David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Claudio Silva, Marco Tarini, and Denis Zorin. 2013b. Quad-mesh generation and processing: A survey. In *Computer graphics forum*, Vol. 32. Wiley Online Library, 51–76.

David Bommes, Henrik Zimmer, and Leif Kobbelt. 2009. Mixed-integer quadrangulation. *ACM transactions on graphics (TOG)* 28, 3 (2009), 1–10.

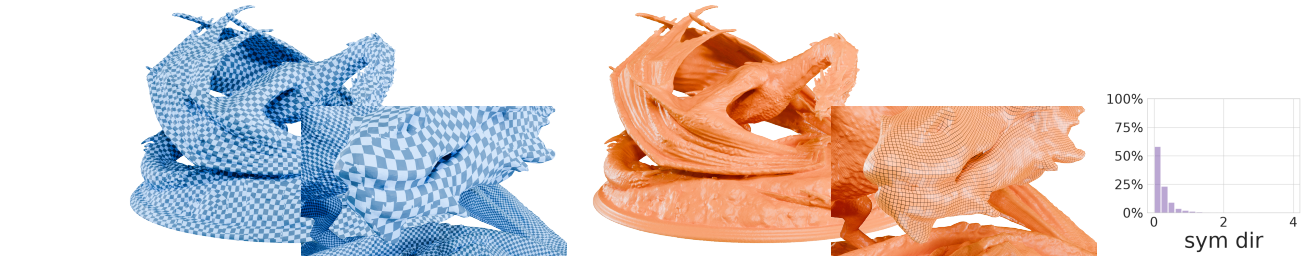
Marcel Campen. 2017. Partitioning surfaces into quadrilateral patches: A survey. In *Computer graphics forum*, Vol. 36. Wiley Online Library, 567–588.

Marcel Campen, David Bommes, and Leif Kobbelt. 2015. Quantized global parametrization. *Acm Transactions On Graphics (tog)* 34, 6 (2015), 1–12.

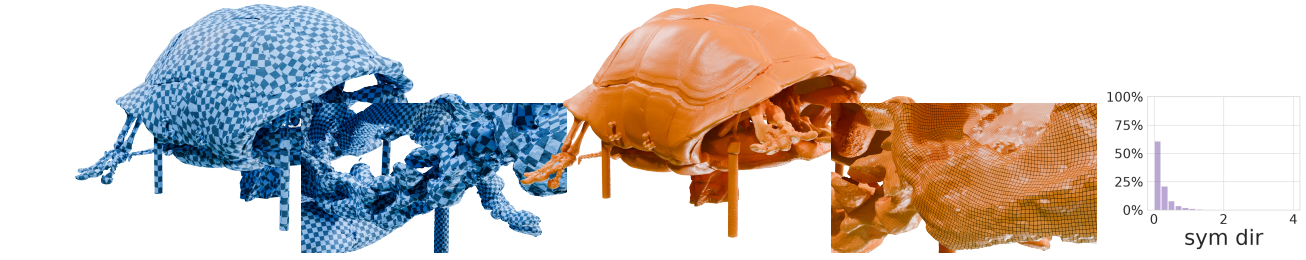
Marcel Campen, Ryan Capouellez, Hanxiao Shen, Leyi Zhu, Daniele Panozzo, and Denis Zorin. 2021. Efficient and Robust Discrete Conformal Equivalence with Boundary. *ACM Trans. Graph.* 40, 6, Article 261 (dec 2021), 16 pages. doi:10.1145/3478513.3480557



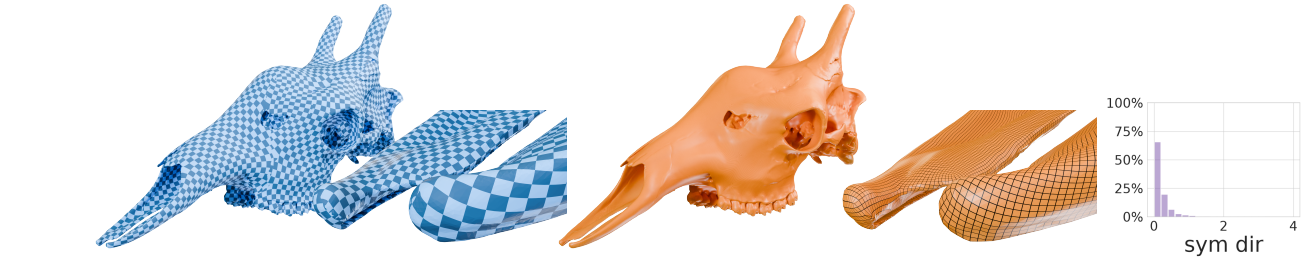
genus 152, 1831170 faces, 2721 cones, 6.3 angle deviation, 696 s



genus 16, 5000000 faces, 1274 cones, 7.9 angle deviation, 1068 s



genus 295, 20091670 faces, 24194 cones, 8.0 angle deviation, 9455 s



genus 20, 1489370 faces, 1442 cones, 7.9 angle deviation, 429 s



genus 61, 1952354 faces, 5715 cones, 5.5 angle deviation, 844 s

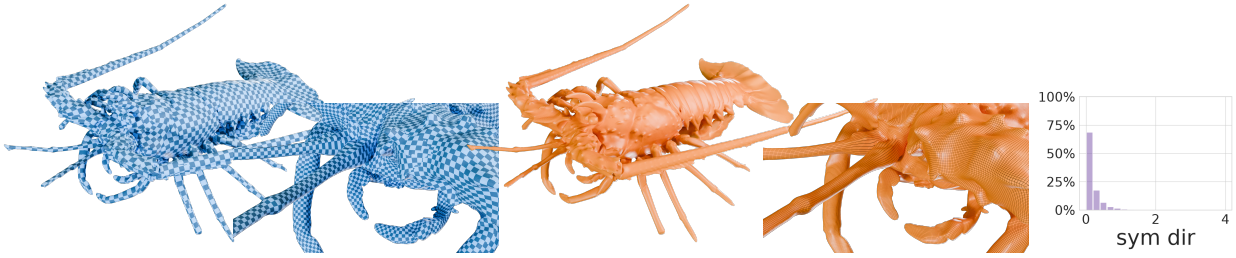
Fig. 26. Examples of meshes from the large-mesh dataset, parametrized and quadrangulated using our method.



genus 2168, 20596784 faces, 75264 cones, 9.4 angle deviation, 7057 s



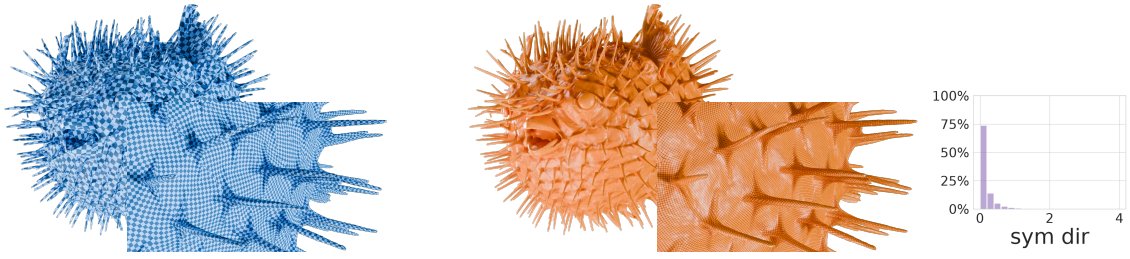
genus 12058, 17502560 faces, 226647 cones, 10.6 angle deviation, 32553 s



genus 25, 1492052 faces, 1405 cones, 6.0 angle deviation, 765 s

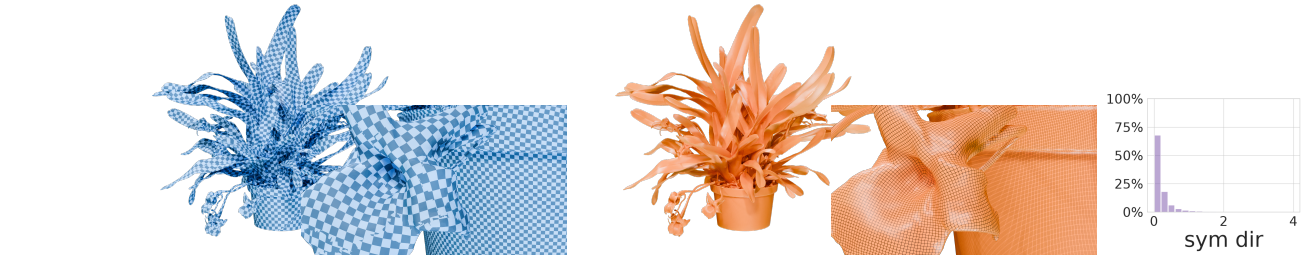


genus 0, 26062720 faces, 1937 cones, 10.2 angle deviation, 17026 s



genus 45, 998270 faces, 3611 cones, 6.9 angle deviation, 347 s

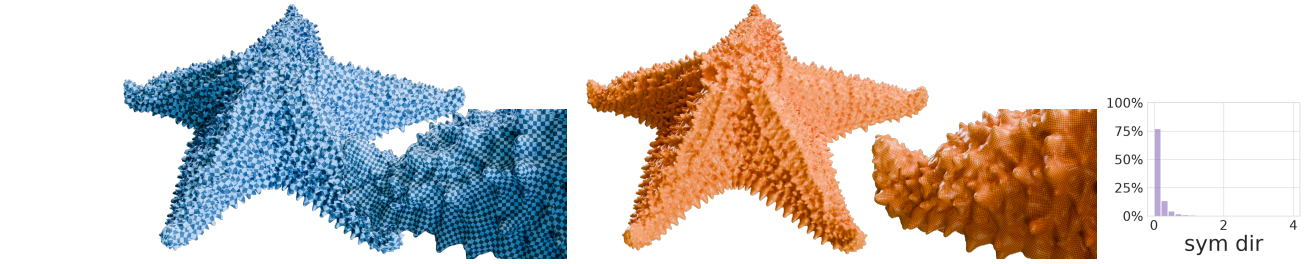
Fig. 27. Examples of meshes from the large-mesh dataset, parametrized and quadrangulated using our method.



genus 113, 4485154 faces, 4451 cones, 7.2 angle deviation, 1743 s



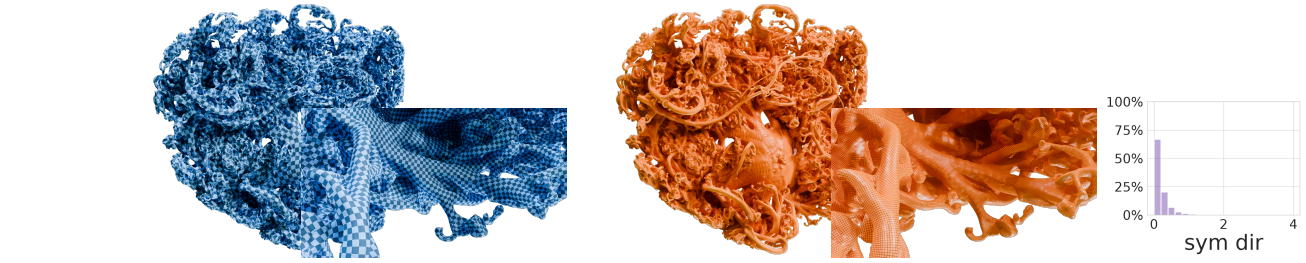
genus 581, 9857904 faces, 16441 cones, 6.5 angle deviation, 12393 s



genus 47, 1473634 faces, 10348 cones, 6.9 angle deviation, 586 s



genus 443, 3930108 faces, 17500 cones, 7.7 angle deviation, 2204 s



genus 436, 5794044 faces, 22379 cones, 6.7 angle deviation, 2158 s

Fig. 28. Examples of meshes from the large-mesh dataset, parametrized and quadrangulated using our method.



Fig. 29. Our large mesh dataset, consisting of manifold meshes with more than 100,000 faces, and up to 26 million triangles.

Marcel Campen, Hanxiao Shen, Jiaran Zhou, and Denis Zorin. 2018. Seamless Parametrization with Arbitrarily Prescribed Cones. *arXiv preprint arXiv:1810.02460* (2018).

Ryan Capouellez, Rodrigo Singh, Martin Heistermann, David Bommes, and Denis Zorin. 2025. Feature-Aligned Parametrization in Penner Coordinates. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–21.

Ryan Capouellez and Denis Zorin. 2023. Metric Optimization in Penner Coordinates. *ACM Trans. Graph.* 42, 6, Article 234 (dec 2023), 19 pages. doi:10.1145/3618394

Ryan Capouellez and Denis Zorin. 2024. Seamless Parametrization in Penner Coordinates. *ACM Trans. Graph.* 43, 4, Article 61 (jul 2024), 13 pages. doi:10.1145/3658202

Guillaume Coiffier and Etienne Corman. 2023. The Method of Moving Frames for Surface Global Parametrization. *ACM Transactions on Graphics* 42, 5 (2023), 1–18.

Yoann Coudert-Osmont, David Desobry, Martin Heistermann, David Bommes, Nicolas Ray, and Dmitry Sokolov. 2024. Quad Mesh Quantization Without a T-Mesh. In *Computer Graphics Forum*, Vol. 43. Wiley Online Library, e14928.

Qiujie Dong, Jiepeng Wang, Rui Xu, Cheng Lin, Yuan Liu, Shiqing Xin, Zichun Zhong, Xin Li, Changhe Tu, Taku Komura, et al. 2025a. CrossGen: Learning and Generating Cross Fields for Quad Meshing. *ACM Transactions on Graphics (TOG)* 44, 6 (2025), 1–15.

Qiujie Dong, Huibiao Wen, Rui Xu, Shuangmin Chen, Jiaran Zhou, Shiqing Xin, Changhe Tu, Taku Komura, and Wenping Wang. 2025b. NeurCross: A Neural Approach to Computing Cross Fields for Quad Mesh Generation. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–17.

Hans-Christian Ebke, David Bommes, Marcel Campen, and Leif Kobbelt. 2013. QEx: robust quad mesh extraction. *ACM Trans. Graph.* 32, 6, Article 168 (Nov. 2013), 10 pages. doi:10.1145/2508363.2508372

Hans-Christian Ebke, Patrick Schmidt, Marcel Campen, and Leif Kobbelt. 2016. Interactively controlled quad remeshing of high resolution 3d models. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–13.

Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. 1995. Multiresolution analysis of arbitrary meshes. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. 173–182.

B. Falcidieno. 2004. AIM@SHAPE project presentation. In *Proceedings Shape Modeling Applications, 2004.* 329.

Ugo Finnendahl, Dimitrios Bogiokas, Pablo Robles Cervantes, and Marc Alexa. 2023. Efficient Embeddings in Exact Arithmetic. *ACM Trans. Graph.* 42, 4, Article 71 (July 2023), 17 pages.

Matthew Fisher, Boris Springborn, Alexander I. Bobenko, and Peter Schroder. 2006. An algorithm for the construction of intrinsic delaunay triangulations with applications to digital geometry processing. In *ACM SIGGRAPH 2006 Courses* (Boston, Massachusetts) *(SIGGRAPH '06)*. Association for Computing Machinery, New York, NY, USA, 69–74. doi:10.1145/1185657.1185668

Xiao-Ming Fu, Jian-Ping Su, Zheng-Yu Zhao, Qing Fang, Chunyang Ye, and Ligang Liu. 2021. Inversion-free geometric mapping construction: A survey. *Computational Visual Media* 7, 3 (2021), 289–318.

Mark Gillespie, Boris Springborn, and Keenan Crane. 2021. Discrete conformal equivalence of polyhedral surfaces. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–20.

Xianfeng Gu, Ren Guo, Feng Luo, Jian Sun, and Tianqi Wu. 2018. A discrete uniformization theorem for polyhedral surfaces II. *Journal of Differential Geometry* 109, 3 (2018), 431–466.

Eden Fedida Hefetz, Edward Chien, and Ofir Weber. 2019. A Subspace Method for Fast Locally Injective Harmonic Mapping. *Computer Graphics Forum* 38, 2 (2019), 105–119.

Martin Heistermann, Jethro Warnett, and David Bommes. 2023. Min-Deviation-Flow in Bi-directed Graphs for T-Mesh Quantization. *ACM Trans. Graph.* 42, 4 (2023), 70–1.

Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. 2020. Fast Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 39, 4, Article 117 (July 2020), 18 pages. doi:10.1145/3386569.3392385

Jingwei Huang, Yichao Zhou, Matthias Niessner, Jonathan Richard Shewchuk, and Leonidas J Guibas. 2018. Quadriflow: A scalable and robust method for quadrangulation. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 147–160.

Wenzel Jakob, Marco Tarini, Daniele Panozzo, Olga Sorkine-Hornung, et al. 2015. Instant field-aligned meshes. *ACM Trans. Graph.* 34, 6 (2015), 189–1.

Felix Kälberer, Matthias Nieser, and Konrad Polthier. 2007. Quadcover-surface parameterization using branched coverings. In *Computer graphics forum*, Vol. 26. Wiley Online Library, 375–384.

Liliya Kharevych, Boris Springborn, and Peter Schröder. 2006. Discrete conformal mappings via circle patterns. *ACM Trans. Graph.* 25 (April 2006), 412–438. Issue 2.

A Klein, A Certain, T DeRose, T Duchamp, and W Stuetzle. 1997. *Vertex-based Delaunay triangulation of meshes of arbitrary topological type.* Technical Report. Tech. rep., University of Washington.

Denis Kovacs, Ashish Myles, and Denis Zorin. 2010. Anisotropic quadrangulation. In *Proceedings of the 14th ACM Symposium on solid and physical modeling*. 137–146.

Aaron WF Lee, Wim Sweldens, Peter Schröder, Lawrence Cowsar, and David Dobkin. 1998. MAPS: Multiresolution adaptive parameterization of surfaces. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 95–104.

Zohar Levi. 2022. Seamless parametrization of spheres with controlled singularities. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 57–68.

Zohar Levi. 2023. Seamless Parametrization with Cone and Partial Loop Control. *ACM Transactions on Graphics* 42, 5 (2023), 1–22.

Mo Li, Qing Fang, Zheng Zhang, Ligang Liu, and Xiao-Ming Fu. 2023. Efficient cone singularity construction for conformal parameterizations. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–13.

Zhongxuan Liang, Wei Du, and Xiao-Ming Fu. 2025. Field Smoothness-Controlled Partition for Quadrangulation. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–15.

Yaron Lipman. 2012. Bounded Distortion Mapping Spaces for Triangular Meshes. *ACM Trans. Graph.* 31, 4 (2012), 108:1–108:13.

Hsueh-Ti Derek Liu, Mark Gillespie, Benjamin Chislett, Nicholas Sharp, Alec Jacobson, and Keenan Crane. 2023. Surface Simplification using Intrinsic Error Metrics. *ACM Trans. Graph.* 42, 4, Article 118 (July 2023), 17 pages. doi:10.1145/3592403

Hsueh-Ti Derek Liu, Jiayi Eris Zhang, Mirela Ben-Chen, and Alec Jacobson. 2021. Surface multigrid via intrinsic prolongation. *ACM Trans. Graph.* 40, 4, Article 80 (July 2021), 13 pages. doi:10.1145/3450626.3459768

Ligang Liu, Chunyang Ye, Ruiqi Ni, and Xiao-Ming Fu. 2018. Progressive parameterizations. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

Hendrik Meyer, Bernd Bickel, and Marc Alexa. 2026. Hierarchical Optimization of the As-Rigid-As-Possible Energy. *Computer Graphics Forum* n/a, n/a (2026), e70404. doi:10.1111/cgf.70404

Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. Robust field-aligned global parametrization. *ACM Trans. Graph.* 33, 4, Article 135 (July 2014), 14 pages. doi:10.1145/2601097.2601154

Ashish Myles and Denis Zorin. 2012. Global parametrization by incremental flattening. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–11.

Ashish Myles and Denis Zorin. 2013. Controlled-distortion constrained global parametrization. *ACM Transactions on Graphics* 32, 4 (2013), 105.

Alexander Naitsat, Gregory Naitzat, and Yehoshua Y Zeevi. 2021. On Inversion-Free Mapping and Distortion Minimization. *Journal of Mathematical Imaging and Vision* (2021), 1–36.

Nico Pietroni, Stefano Nuvoli, Thomas Alderighi, Paolo Cignoni, and Marco Tarini. 2021. Reliable feature-line driven quad-remeshing. *ACM Trans. Graph.* 40, 4, Article 155 (July 2021), 17 pages. doi:10.1145/3450626.3459941

Nico Pietroni, Marco Tarini, Olga Sorkine, and Denis Zorin. 2011. Global parametrization of range image sets. In *Proceedings of the 2011 SIGGRAPH Asia Conference*. 1–10.

Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable Locally Injective Mappings. *ACM Trans. Graph.* 36, 2 (2017), 16:1–16:16.

Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. 2006. Periodic global parameterization. *ACM Transactions on Graphics (TOG)* 25, 4 (2006), 1460–1485.

Nicolas Ray, Bruno Vallet, Wan Chiu Li, and Bruno Lévy. 2008. N-symmetry direction field design. *ACM Transactions on Graphics (TOG)* 27, 2 (2008), 1–13.

Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. 2013. Locally Injective Mappings. *Computer Graphics Forum* 32, 5 (2013), 125–135.

Nicholas Sharp and Keenan Crane. 2020. You Can Find Geodesic Paths in Triangle Meshes by Just Flipping Edges. *ACM Trans. Graph.* 39, 6 (2020).

Alla Sheffer and Eric de Sturler. 2001. Parameterization of faceted surfaces for meshing using angle-based flattening. *Engineering with computers* 17, 3 (2001), 326–337.

Alla Sheffer, Bruno Lévy, Maxim Mogilnitsky, and Alexander Bogomyakov. 2005. ABF++: fast and robust angle based flattening. *ACM Trans. Graph.* 24, 2 (April 2005), 311–330. doi:10.1145/1061347.1061354

Hanxiao Shen, Leyi Zhu, Ryan Capouellez, Daniele Panozzo, Marcel Campen, and Denis Zorin. 2022. Which cross fields can be quadrangulated? global parameterization from prescribed holonomy signatures. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–12.

Jason Smith and Scott Schaefer. 2015. Bijective parameterization with free boundaries. *ACM Trans. Graph.* 34, 4, Article 70 (July 2015), 9 pages. doi:10.1145/2766947

Smithsonian Institution. 2025. *Smithsonian 3D Digitization.* https://3d.si.edu

Boris Springborn, Peter Schröder, and Ulrich Pinkall. 2008. Conformal Equivalence of Triangle Meshes. *ACM Transactions on Graphics* 27, 3 (2008), 1–11.

Amir Vaxman, Marcel Campen, Olga Diamanti, Daniele Panozzo, David Bommes, Klaus Hildebrandt, and Mirela Ben-Chen. 2016. Directional field synthesis, design, and processing. In *Computer graphics forum*, Vol. 35. Wiley Online Library, 545–572.

Jiaran Zhou, Changhe Tu, Denis Zorin, and Marcel Campen. 2020. Combinatorial construction of seamless parameter domains. In *Computer graphics forum*, Vol. 39. Wiley Online Library, 179–190.

Qingnan Zhou and Alec Jacobson. 2016. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797* (2016).