Feature-Aligned Parametrization in Penner Coordinates

RYAN CAPOUELLEZ, New York University, USA
RODRIGO SINGH, New York University, USA
MARTIN HEISTERMANN, University of Bern, Switzerland
DAVID BOMMES, University of Bern, Switzerland
DENIS ZORIN, New York University, USA

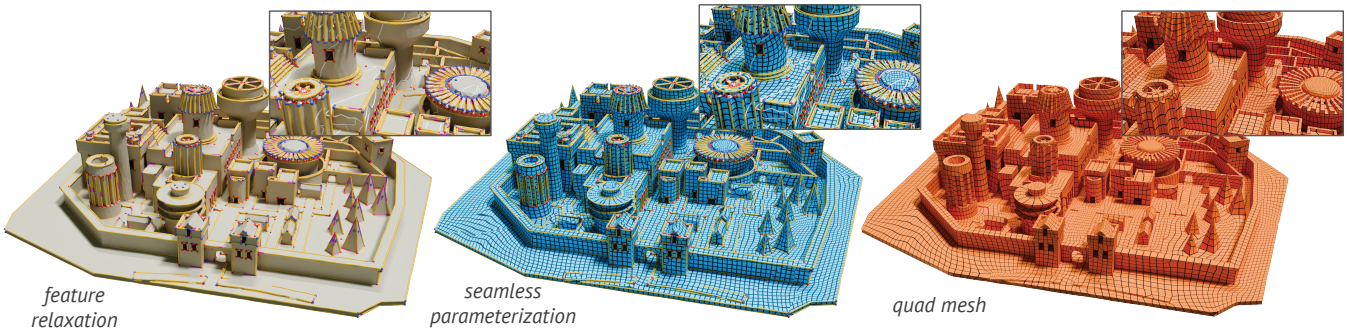


Fig. 1. Our robust feature-aligned seamless parametrization method starts with a mesh annotated with feature edges and parametrization singularities and holonomies, most of which are enforced as hard constraints, and the rest optimized for alignment (left). Our algorithm produces a seamless parametrization with all constraints aligned to a high precision, and has the prescribed cones and holonomies (middle). This parametrization can be used as an initial parameterization for a quantization algorithm producing a quadrangulation (right).

Parametrization is a key element of many geometric modeling tasks. Seamless parametrization, in particular, is needed as a starting point for many algorithms for quadrangulation and conversion to high-order patches, as well as for the construction of seamless texture maps and displacement maps.

Seamless parametrizations are difficult to compute robustly, in part because, in general, it is not known if one exists for a given mesh connectivity or for a particular configuration of singularities. Recently, Penner-coordinate-based methods that allow for connectivity changes have been shown to achieve a perfect success rate on a widely used dataset (Thingi10k).

However, previously proposed Penner coordinate methods do not support sharp feature alignment or soft alignment with preferred directions on the surface, both of which are important for practical applications, especially those involving models with sharp features.

In this paper, we extend Penner coordinates to surfaces with sharp features to which the parametrization needs to be aligned. Our algorithm extends the holonomy signature description of seamless parametrizations to surfaces with marked feature curves. We describe sufficient conditions for obtaining feasible solutions and describe a two-phase method to efficiently enforce feature constraints or minimize residual errors when solutions are unattainable. We demonstrate that the resulting algorithm works robustly on the Thingi10k dataset with automatic feature labeling, and the resulting seamless parametrizations can be optimized, quantized, and quadrangulated, completing the quad mesh generation pipeline.

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

Additional Key Words and Phrases: parameterization, feature alignment, seamless, Newton's method, cone metric, intrinsic triangulation, Penner coordinates, quadrangulation

ACM Reference Format:
Ryan Capouellez, Rodrigo Singh, Martin Heistermann, David Bommes, and Denis Zorin. 2025. Feature-Aligned Parametrization in Penner Coordinates. *ACM Trans. Graph.* 44, 4 (August 2025), 21 pages. https://doi.org/10.1145/3731216

1 INTRODUCTION

Seamless parametrization is a critical tool in geometry processing, used in methods for quad mesh generation, conforming quad and T-mesh layout construction for high-order surface approximation and texturing, surface-to-surface mapping, and other contexts where surface parametrization is essential.

An important variant of the seamless parametrization problem considers parametrization singularities (cones) and their type, and more generally the *holonomy signature* (Section 4) as fixed. These signatures are typically derived from a cross-field on the surface, which allows precise control over cone placement and the preferred resampling directions. This flexibility makes them highly valuable for many applications.

It is common for meshes, especially those obtained from CAD models, to have *sharp features*, either directly known from the source

Authors' addresses: Ryan Capouellez, rjc8237@nyu.edu, New York University, USA; Rodrigo Singh, rs4208@nyu.edu, New York University, USA; Martin Heistermann, martin.heistermann@unibe.ch, University of Bern, Switzerland; David Bommes, david.bommes@unibe.ch, University of Bern, Switzerland; Denis Zorin, dzorin@cs.nyu.edu, New York University, USA.

Please use nonacm option or ACM Engage class to enable CC licenses
This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2025 Copyright held by the owner/author(s).
ACM 0730-0301/2025/8-ART
https://doi.org/10.1145/3731216

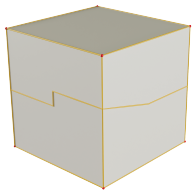


Fig. 2. An example of a feature configuration with no feasible solution. Aligned horizontal features in the middle of faces around the cube force all of them to have the same parametric coordinate (e.g., same u). The vertical short feature line is labeled to be aligned with v, so can only have parametric length 0.

model or inferred, e.g., by heuristics or a data-driven algorithm. For most applications, the parametric directions of the parametrization need to be aligned with these features. For example, if the downstream application is quad meshing, with quads obtained by sampling the parametric lines, these lines should be following feature lines precisely.

Penner coordinates is an approach to introducing coordinates on the space of metrics on a mesh, allowing for mesh connectivity changes, and reducing to just edge lengths for fixed connectivity. Recent methods using the Penner coordinates have proven robust and effective in computing seamless parametrizations for (almost) arbitrary holonomy signatures [Capouellez and Zorin 2024], as demonstrated on a large dataset of models of a broad range of geometric and topological complexity. However, existing methods based on Penner coordinates currently do not support the parametrization feature alignment, i.e., ensuring that marked edges on the input mesh are aligned with coordinate directions.

This paper addresses this gap by introducing a novel approach to feature-aligned seamless parametrization. We aim to extend Penner coordinate-based methods to ensure alignment with features while maintaining robustness and compatibility with downstream applications like quadrangulation.

The introduction of feature alignment requires answering two key questions:

(1) How to handle feature constraints if the connectivity changes? The ability to flip any edge is critical for the robustness of algorithms of Capouellez and Zorin [2024] on which we built; as a consequence, a feature edge may disappear.
(2) How do we handle cases when feature constraints do not have a feasible solution? Penner coordinates-based algorithms of Capouellez and Zorin [2023] and Capouellez and Zorin [2024] demonstrate strong empirical performance when feasible solutions exist, which is pretty much always for the seamless parametrization problem without constraints, subject to mild restrictions [Shen et al. 2022]. This is not the case for feature-aligned parameterizations, for which inconsistent conditions are more common; a simple example is shown in Figure 2.

Unfortunately, the question of the existence of solutions is not mathematically resolved for hard feature constraints with an arbitrary feature graph, and the configurations with no solution appear to be more common. For this reason, we take the relaxation approach. A standard approach like imposing constraints in a weak form, e.g., as a penalty term in an optimization, is not a good match for the task: for the most common application, quadrangulation, even small deviations from feature lines create undesirable artifacts, so in the cases when all constraints cannot be satisfied, distributing the error over all constraints is not appropriate. One can also formalize the problem as finding a maximal feasible set of constraints, but this problem is NP-hard even in simpler settings. We note that one can take two approaches to the competing goals of respecting input feature curve constraints and cone constraints: relax some of the cone constraints (e.g., allow adding cones [Pietroni et al. 2021],) or, as we do, relax some of the feature constraints, leading to different types of artifacts. These approaches are complementary and may be combined in the future for optimal results.

Our approach is based on the conjecture that if the feature graph is a set of trees, then a solution exists, which empirically holds for the large majority of meshes in our dataset. In this way, a relatively small number of constraints are initially eliminated and then reintroduced, if a solution can be found. We demonstrate that this approach results in high feature alignment quality and, at the same time, produces a seamless parametrization for a large dataset of 17,421 meshes. In summary, our contributions include:

- Formulation of the feature-aligned seamless parametrization problem in Penner coordinates, supporting mesh connectivity changes;
- An algorithm for solving the resulting constraint system, based on partitioning the constraints into hard constraints and a smaller number of relaxed constraints.
- Evaluation of a complete quadrangulation pipeline, demonstrating that the resulting seamless parametrizations are suitable as the starting point for quantization algorithms for quadrangulation (we emphasize that our focus is on robustness of the pipeline; we do not aim for state-of-the-art quad mesh quality).

In the spirit of Capouellez and Zorin [2024], conceptually, the algorithm (Section 7) is remarkably simple: formulate the problem as a set of constraints on angles and lengths, and solve this system using a variant of Newton's method; the key to robustness is that the machinery of Penner coordinates tells us how to allow for connectivity changes in the process. While there is additional effort required in the case of feature alignment to make the problem compatible with Penner coordinate use, the final setup is similar.

2 RELATED WORK

We focus on the most closely related work here; more comprehensive surveys on parametrization can be found in Naitsat et al. [2021] and Fu et al. [2021], and on quadrangulation in Bommes et al. [2013b]; Campen [2017].

Our work extends Capouellez and Zorin [2024], and has similar motivation: our feature-aligned seamless parametrizations provide a feasible starting point for conforming quad layout algorithms, such as Bommes et al. [2009]; Campen et al. [2015]; Lyon et al. [2021], T-mesh algorithms, e.g., Campen and Zorin [2017]; Myles et al. [2014] or other parametrization optimization methods, e.g., Liu et al. [2018]; Rabinovich et al. [2017]; Schüller et al. [2013].

Intrinsic methods. Among intrinsic methods, i.e., methods operating on intrinsic variables, typically angles and lengths [Ben-Chen et al. 2008; Kharevych et al. 2006; Sheffer and de Sturler 2001; Springborn et al. 2008], the most closely related to our work, beyond Penner coordinate methods [Capouellez and Zorin 2023, 2024] on which we build, are discrete conformal maps [Campen et al. 2021; Gillespie et al. 2021; Springborn et al. 2008] based on the uniformization theorems of Gu et al. [2018a,b]; Springborn [2020]. None of these methods were designed to handle feature constraints; both Campen et al. [2021]; Gillespie et al. [2021] handle boundaries via mesh doubling. We also use this for features.

Similar to Capouellez and Zorin [2024] we avoid the higher computational complexity of iterative projection for constraints used in earlier work on Penner-coordinate-based parametrization [Capouellez and Zorin 2023], which uses conformal parametrization to project to the space of metrics with given vertex angles in an internal optimization loop. We take advantage of the implicit approximate minimization of the deviation from the initial metric provided by the extended Newton method.

Seamless parametrization constructions. Theoretical guarantees of existence for seamless parametrizations are remarkably hard to obtain even in the absence of features and boundaries. Shen et al. [2022] provides theoretical guarantees, but is both expensive and may be numerically fragile. Related methods Campen et al. [2018] and Zhou et al. [2020] also involve a highly distorted parametrization stage and do not provide control over loop holonomy angles. Other works with partial control of holonomy include Levi [2022, 2023]. None of these works consider feature alignment.

Other methods that do aim for feature alignment, often fail to ensure injectivity or find a feasible solution, e.g., Bommes et al. [2013a, 2009]; Bright et al. [2017]; Campen et al. [2015]; Hefetz et al. [2019]; Lipman [2012].

An alternative approach is to construct a T-mesh partition of the surface that does not necessarily correspond to a valid seamless parametrization, e.g., by tracing a cross-field, and then modify it by inserting or merging singularities [Lyon et al. 2021; Myles et al. 2014; Pietroni et al. 2021]. We aim to preserve the singularities, and more generally the holonomy structure.

We rely on an implementation of the method of Bommes et al. [2009] for generating cross-fields from which we infer the cone positions and angles, and to which our parametrization is aligned. We refer to Vaxman et al. [2016], for a survey.

Quadrangulation. While our focus is on constructing seamless parametrizations, which can be used in many ways, one of the main applications is conforming quad layouts with prescribed cones, holonomies, and feature lines, which requires additional parametrization transformations. The methods of Campen et al. [2015]; Heistermann et al. [2023]; Lyon et al. [2021] describe techniques that, starting with a seamless parametrization, construct a patch partition of a mesh with integer parametric arc length assignments that can be used to produce a perturbed parametrization suitable for conforming quad meshing. We demonstrate that our parametrizations are of sufficient quality to serve as a starting point for these algorithms.

We note that other approaches to quadrangulation, e.g., local [Huang et al. 2018; Jakob et al. 2015] or based on T-mesh tracing [Myles et al. 2014; Pietroni et al. 2021, 2016], can produce quad meshes robustly and with feature alignment, but do not preserve guiding field singularities. Our method belongs to the category of methods that preserve the field holonomy signature exactly.

3 PROBLEM FORMULATION

We begin with the formulation of the feature-aligned seamless parametrization problem on a fixed mesh; a critical feature of our approach is that the triangulation changes as the parametrization is solved for. This also presents the main challenge in dealing with the alignment constraints. We will address this in subsequent sections; here, we establish the problem in its simplest form in intrinsic variables (angles and lengths), which is necessary to apply Penner coordinate-based methods.

3.1 Seamless parametrization

We first describe the seamless parametrization problem in intrinsic variables for meshes without features.

A *discrete metric* is an assignment of positive lengths $\ell : E \to \mathbb{R}^+$ to the edges of the mesh that satisfy the triangle equality for each triangular face $f_{ijk} \in F$. A (mostly) flat parametrization with the sum of angles at almost all vertices equal to 2π is directly related to parameterization, i.e., mapping the surface to the plane. Such mappings can be obtained from an almost everywhere flat metric by using lengths in a breadth-first layout process to determine parametric plane positions for each vertex.

Holonomy signature. Let α_i^{jk} denote the interior angle of the triangle f_{ijk} at vertex v_i in a metric on the mesh (in our approach, the metric is variable, and evolves from the initial 3D metric to the parametrization metric). We define the *cone angle* at vertex $v_i \in V$ to be just the sum of angles

$$\sum_{f_{ijk} \ni v_i} \alpha_i^{jk} = \Theta(v_i) \tag{1}$$

$2\pi - \Theta(v_i)$ is the discrete curvature at v_i. For most vertices, the curvature is zero, and there is typically a set of isolated cones with locations and angles inferred, e.g., from a cross-field.

For a dual loop γ_s of faces on the mesh separated by edges $e_1^s, ..., e_{N_s}^s$, the *holonomy angle* of the loop is

$$\sum_{m=1}^{N_s} d_m^s \alpha_m^s = \kappa_s \tag{2}$$

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 $s = 1 \dots 2g$ (Figure 3).

The *holonomy signature* of the metric is a collection of the N_v cone angles $\Theta(v_i)$, one per vertex, and the holonomy angles $\kappa_1, ...\kappa_{2g}$ for a system of loops on the surface. The holonomy signature uniquely determines the holonomy angle of any dual loop on the mesh. That is, to know the holonomy angle of any dual loop on the surface, it suffices to know the holonomy signature of the metric. If all entries of the holonomy signature are integer multiples of $\pi/2$, then the

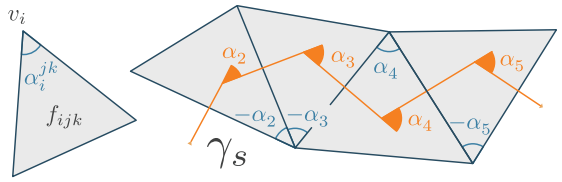


Fig. 3. Holonomy constraint notation, adapted from [Capouellez and Zorin 2024].

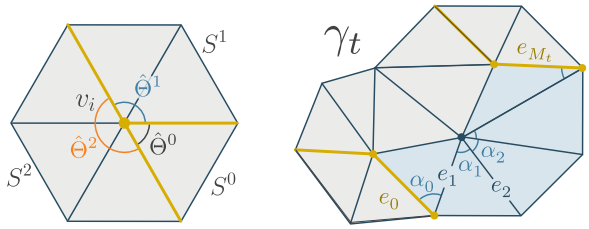


Fig. 4. Left: sector constraints; right: component constraint.

metric is *seamless*. As discussed, e.g., in Capouellez and Zorin [2024], this is equivalent to the notion of seamless parametrization defined in terms of uv coordinates on a mesh, as having $k\pi/2$ rotations of coordinates on all parametrization cuts.

Seamless parametrization problem. The constrained seamless parameterization problem for closed surfaces is to find a metric ℓ such that the holonomy signature of the metric agrees with prescribed values that are all integer multiples of $\pi/2$. Let $\hat{\Theta}$ denote a vector of such $N_v - 1$ target vertex angles for a connected mesh (one is redundant due to the Gauss-Bonnet theorem) and $2g$ target holonomy angle constraints (If the mesh had several connected components, one vertex constraint should be dropped per component). Let $\alpha(\ell)$ denote the vector of all $3N_f$ corner angles of the metric ℓ. The cone angle and holonomy angles are linear combinations of the corner angles α_i^{jk}, so we can express the holonomy signature constraints as the following linear system:

$$C\alpha(\ell) = \hat{\Theta} \tag{3}$$

The problem can also be formulated for surfaces with boundaries and solved by reduction to the closed-surface case (Section 6). This is a nonlinear, underdetermined system. It is well known that for a fixed connectivity M, a solution may not exist for a given holonomy signature, so we must consider retriangulation to allow for solutions to be found. It turns out that it is sufficient to consider all possible triangulations with a fixed vertex set (i.e., there is no need to increase the mesh size, one just needs to flip edges) to ensure solution existence for important subclasses of problems. Before considering this in detail, we extend this formulation to meshes with features.

3.2 Feature alignment in intrinsic variables

We call a mesh M a *mesh with features* if a subset of edges is tagged as aligned with a $\pm u$ or $\pm v$ coordinate direction in the parametric domain. One can already observe the potential for trouble: if mesh connectivity needs to be changed, in the process of optimization, how do we deal with feature edges? We will address this question in Section 4 and subsequent sections.

The constraints for feature alignment in parametric coordinates are almost trivial: the u or v value at the endpoints of the edge should be the same. To recast these constraints in intrinsic variables we formulate these in relative terms. Rather than considering edge alignment with parametric direction, we consider parametric-domain angles between (not necessarily adjacent) edges. For example, if the edge e_1 is supposed to be aligned with u direction, and edge e_2 with v, we may require that the angle between them is $\pi/2$ counted along a chain of faces (dual path) connecting these. One can show that if there is a homotopy of two paths that does not traverse cone vertices, the resulting angle does not depend on the choice of the path. Then if the first edge is aligned with u the second is automatically aligned with v. If the graph of such constraints between pairs of edges is connected, then it is sufficient to align one with a coordinate axis to have all of them aligned. There is a consistency condition on the angle assignments (the discrete Gauss-Bonnet theorem) which is in our tests satisfied automatically by inferring the angles from a feature-aligned cross-field.

We distinguish between two types of alignment constraints: *vertex sector* constraints and *feature component constraints*. Both are linear constraints on the angles of the triangulation.

Suppose a vertex v_i has K_i incident feature edges, numbered sequentially in counterclockwise order around a vertex. We assume that each sector between two sequential feature edges e_r, e_{r+1}, is assigned a target angle $k_r\pi/2$, where k_r is a positive integer, which we denote $\hat{\Theta}^r(v_i)$, and call the sector angle. We define K_i sector constraints for the vertex v:

$$\sum_{f_{ijk} \in S^r(v_i)} \alpha_i^{jk} = \hat{\Theta}^r(v_i) \tag{4}$$

where $S^r(v_i)$ is the set of triangles in sector r. The usual vertex angle constraint is a special case for $K_i = 1$.

If the feature graph has multiple connected components, vertex sector constraints are not sufficient to enforce alignment, as each component may rotate in the parametric domain independently in an arbitrary way. Consider a dual path connecting two edges e_0^t and $e_{M_t}^t$ belonging to separate components of the feature graph, and with edges $e_1^t, \dots e_{M_t-1}^t$ shared by two sequential triangles in the path. We choose the path so that the edges e_m^t are not feature edges. Similar to the holonomy constraints (2), these constraints sum up angles α_m^t between e_m^t and e_{m+1}^t along the dual path γ_t

$$\sum_{m=0}^{M_t-1} d_m^t \alpha_m^t = \kappa_t^{fc} \tag{5}$$

with $d_m^t = \pm 1$ depending on the direction of rotation, and $t = 1 \dots N_{cc} - 1$, where N_{cc} is the number of connected components.

The main difference between (1) and (4), and between (2) and (5), is that the original seamless constraints are defined on closed loops (around a vertex or holonomy loop) and the feature constraints are defined on open paths.

While seemingly minor in a fixed-mesh setting, the difference is drastic when we change mesh connectivity: the two edges the path connects may disappear, and while conceptually the way to deal with closed loops under connectivity changes is straightforward (replace these with a homotopic dual loop), it is much less clear how to approach the problem with feature edges.

We combine all constraints we are solving for, (1), (2), (4), and (5) into a single system of constraints on angles:

$$C^f \alpha = \hat{\Theta} \tag{6}$$

which has exactly the same general form as for the seamless parameterization without features; however, the method of [Capouellez and Zorin 2024] cannot be applied directly to solve this system, for two reasons mentioned above: the path update under connectivity changes is not well defined, and we need to alter the algorithm for solving, and constraints may not be feasible.

4 OVERVIEW OF THE APPROACH

Our approach to extending the Penner coordinate parametrization to meshes with marked features is based on several ideas.

Our main assumption is that the algorithm of Capouellez and Zorin [2024] for meshes without features converges to a solution, which was confirmed by the numerical studies in that paper; with some important caveats, we reduce the problem of computing a parametrization with feature alignment to the feature-free case.

Reduction to seamless parametrization. To address the first problem, we take the following approach.

- We treat the features of the mesh as boundaries, creating two boundary edges for each feature edge, possibly separating the mesh into components. Additional constraints on the lengths of the two sides of the feature are added to ensure that the complete mesh can be stitched back together after parametrization
- We apply the double-cover approach (Figure 5) used in the conformal parametrization of meshes with boundary [Campen et al. 2021], to compute isometry-optimizing parametrizations of the cut mesh components. In this approach, a mesh with a boundary is converted to a closed mesh by gluing a copy along the boundary, and symmetry is enforced between the two.

This idea addresses the issue of combining mesh changes (specifically edge flips) with maintaining feature constraints. In the second step, open-path constraints on features become standard seamless vertex and holonomy constraints, and the feature lines are first transformed into the mesh boundary, and then, in turn, into the symmetry line components on the double cover mesh.

Relaxation. In the case of meshes without feature constraints, it is known [Shen et al. 2022] that under mild assumptions, there is a seamless parametrization for any prescribed holonomy. More specifically, as long as there is at least one cone of valence 3 or 5, and excluding some minimal cone configurations for the torus. As we discussed in the introduction, not much is known about similar conditions for meshes with features. For this reason, most algorithms for feature-aligned parametrization introduce a relaxation of the problem: e.g., Myles et al. [2014] and other works may introduce additional cones and modify holonomies. We take the approach of relaxing feature constraints. A common approach of imposing constraints through a penalty function, is not a good match for the problem, especially if seamless parametrization is used as a starting point for quadrangulation. In this case, the error in the constraints (if these cannot be satisfied exactly) is spread uniformly, likely resulting in misalignment of the resulting quadrangulation with features in many locations. A preferred approach would be to identify a maximal feasible subset of the constraints, but this problem is known to be hard even for linear constraints, and ours are nonlinear.

Instead, we partition the constraints into the initial set of hard constraints and treat the problem as a constrained problem for optimizing the remaining constraints. Our approach can be viewed as similar to Capouellez and Zorin [2023], in which the parametrization distortion optimization problem is solved by a projected gradient descent, with a step along the gradient of the objective is combined with the projection to the constraint set.

We found that the level of robustness similar to the seamless parametrizations without features can be obtained if the feature graph is a tree: the infeasible situations we have observed are mostly related to loops in the graph, so we use it as a starting point, and eliminate more constraints if it does not succeed in a given number of iterations.

In principle, this approach can eliminate all constraints, reducing the problem to seamless parametrization without features for which we have a robust algorithm. In our experiments on a large dataset, we found that nearly the same level of robustness is obtained if the feature graph is reduced to a tree, and all of the relaxed constraints can be recovered for almost all meshes.

Algorithm overview. Before proceeding with the details, we review the algorithm's steps. To simplify, we do not include the additional loop for relaxing the tree constraints further, if it does not succeed. The algorithm's input is a mesh with features and a cross-field (or, at minimum, a set of marked cone vertices and an associated holonomy signature).

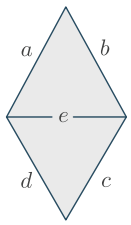
- Construct a forest of spanning trees for the feature edges. The edges in the forest are marked as hard-constraint edges. The feature graph separates the mesh into components; we ensure that each component's boundary contains at least one edge of the spanning tree.
- To simplify the construction of our system of hard constraints, we refine the mesh to ensure every face is adjacent to at most one feature edge.
- Cut the mesh along feature lines, and construct doubles of all connected components.
- Check for known infeasible holonomy signatures for seamless parametrization without features.
- Construct initial dual paths for holonomy and feature connection constraints.
- Construct the initial matrices C^f of angle-based constraints in two versions: all feature constraints, and hard constraints only.
- Iterate the following steps:

  - Use modified Newton's method with hard constraints only to get a seamless metric.
  - Compute a descent direction as the Newton direction for the full constraints.
  - Use backtracking line search with projection to the hard constraints along the full constraint direction.
  - Terminate once the relaxed constraints are satisfied, or the triangle quality reaches a given threshold.
- Parameterize each component with an overlay mesh, an arrangement of the original mesh with the mesh with modified connectivity the parametrization algorithm produces.
- Stitch together connected components along boundaries.
- Simplify the overlay mesh to keep only the edges of the original mesh whenever possible.
- Optionally, optimize the mesh with symmetric Dirichlet and field alignment energies [Bommes et al. 2009; Smith and Schaefer 2015], with the hard and relaxed constraints satisfied within numerical tolerance imposed as hard.

As the test of suitability of the resulting parametrization for quadrangulation, we use it as the input to the quantization pipeline developed in Campen et al. [2015]; Heistermann et al. [2023]; Lyon et al. [2021] provided by the authors. We provide additional details about this stage in Section 8. The pipeline is used largely as it was in the previous work.

5 BACKGROUND: PENNER COORDINATES

Intrinsic Delaunay connectivity. To define Penner coordinates on the space of flat metric with a given vertex set of the same topology (in other words, all metrics that can be obtained by assigning lengths to edges of a mesh obtained from a given mesh by flips), we first introduce a canonical choice of connectivity for a given metric. A (Euclidean) intrinsic edge flip changes the connectivity of the surface mesh without changing the cone metric it describes. There are many equivalent connectivities that can be reached through intrinsic edge flips, but we can obtain an (almost) unique canonical choice by using the *Delaunay* connectivity. That is, for any mesh (M, ℓ), we can compute an equivalent $(\tilde{M}, \tilde{\ell}) = \mathrm{Del}(M, \ell)$ such that each edge e of $\tilde{M}$ satisfies the intrinsic Delaunay condition $\alpha_i + \alpha_j \leq \pi$, where α_i, α_j are the triangle angles opposite edge e.



This condition can be equivalently expressed as a rational expression in edge lengths defined for any choice of these lengths even if these do not satisfy triangle inequality; this fact is important for defining Penner coordinates.

$$\frac{\ell(a)^2 + \ell(b)^2 - \ell(e)^2}{2\ell(a)\ell(b)} + \frac{\ell(c)^2 + \ell(d)^2 - \ell(e)^2}{2\ell(c)\ell(d)} \geq 0 \tag{7}$$

The standard flip algorithm, which simply flips any edge that does not satisfy this local condition, is guaranteed to terminate and produce a Delaunay mesh, so the Delaunay connectivity can be computed efficiently in practice.

Penner coordinates. The Delaunay connectivity is an ideal choice for a fixed cone metric, but it is difficult to maintain the Delaunay property when deforming a metric, e.g., during optimization. Penner coordinates represent all cone metrics $(\tilde{M}, \tilde{\ell})$ on an arbitrary triangulation with the same vertices, using edge-based coordinates ℓ on a single, arbitrarily chosen, connectivity M_0. While for a subset of metric for which M_0 is a Delaunay connectivity, these coincide with edge lengths, an arbitrary assignment of ℓ, not necessarily satisfying the triangle inequality, defines a metric.

We first define a change of coordinates $\tau(M, M') : \mathbb{R}_+^{N_e} \to \mathbb{R}_+^{N_e}$ between *any* two connectivities M, M' with the same topology and vertex set. We use the Ptolemy formula as atomic transition maps when a single edge e is flipped to e':

$$\ell'(e') = \frac{\ell(a)\ell(c) + \ell(b)\ell(d)}{\ell(e)}. \tag{8}$$

This formula can be applied to an assignment of positive numbers to the edges whether these correspond to actual lengths (i.e., satisfy triangle inequality) or not. The coordinates for all other edges are left unchanged. For a sequence of flips of edges $e_1, ..., e_n$ connecting two connectivities M and M_0, we define the transition map $\tau(M, M_0) : \mathbb{R}_+^{N_e} \to \mathbb{R}_+^{N_e}$ as the composition $\tau_n \circ \tau_{n-1} \circ \cdots \circ \tau_1$ of the transition maps for the individual flips. These transition maps are smooth and well-defined, i.e., they do not depend on the sequence of flips used to construct the map [Penner 1987].

Finally, we define the Penner coordinates, based on the ideas introduced in Penner [1987] in the form used in Capouellez and Zorin [2023]:

Definition 5.1. Penner coordinates for a cone metric with Delaunay length coordinates (M, ℓ), with respect to a connectivity M_0 is a vector $P_{M_0}(M, \ell)$ of positive numbers in $\mathbb{R}_+^{N_e}$ defined as $P_{M_0}(M, \ell) = \tau(M, M_0)(\ell)$.

Penner coordinates as optimization variables. Penner coordinates establish a one-to-one correspondence between the space of metrics on a mesh and $\mathbb{R}_+^{N_e}$. For a fixed mesh M_0, any $\ell \in \mathbb{R}_+^{N_e}$ is a vector of Penner coordinates for some metric. Moreover, its canonical Delaunay metric representation $(\tilde{M}, \tilde{\ell})$ can be obtained by the Week's flip algorithm, which is just the standard flip algorithm for Delaunay with Ptolemy intrinsic flips substituted for Euclidean intrinsic flips. By the definition of Penner coordinates, the canonical coordinates $\tilde{\ell}$ are just (Delaunay) length coordinates on $\tilde{M}$, so the corner angles $\alpha(\tilde{\lambda})$ of $\tilde{M}$ are well-defined. Since the change of coordinates determined by the flip algorithm is smooth, these values are also differentiable functions of the Penner coordinates.

Penner coordinates thus provide a remarkably simple optimization space for cone metrics with implicit retriangulation that only needs to be performed temporarily when lengths or angles are required, e.g., for updating gradient or Newton directions. These coordinates have been used successfully to solve metric optimization problems in Capouellez and Zorin [2023] and the constrained seamless parametrization problem in Capouellez and Zorin [2024]. The latter is solved with a simple extended Newton's method for the seamless constraint equation expressed in Penner coordinates:

$$F(\lambda) = C\alpha(\mathrm{Del}(M_0, \lambda)) - \hat{\Theta} = 0 \tag{9}$$

Here, $\lambda = 2 \log \ell$ is the logarithmic Penner coordinate, which is used to avoid the positivity constraint on Penner coordinates. Although

a proof of convergence for $g > 0$ is not available yet, the fast and consistently successful empirical performance on a large dataset [Zhou and Jacobson 2016] suggests that there may be an underlying convex formulation.

6 PROBLEM REDUCTION TO SEAMLESS PARAMETRIZATION WITHOUT FEATURES

We reduce the system of constraints for a mesh with features to a seamless parametrization problem without features in two steps. First, we convert the problem to a seamless parametrization problem with additional constraints on a mesh with boundary, and then, as the second step, we convert the latter to a seamless parametrization problem on a mesh without boundary, matching the setting of Capouellez and Zorin [2024]. The solution of the latter is directly converted to the solution of our original problem.

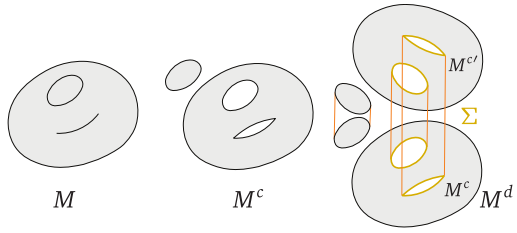


Fig. 5. Reduction of the feature-aligned parametrization problem to seamless parametrization on a closed mesh.

Conversion to a mesh with boundary. For the input mesh M, we construct a *cut mesh* M^c by separating it along all feature edges. The new mesh M has the same set of faces, and a set of vertices V^c, with a map $a : V^c \to V$ mapping each vertex to its ancestor in V. Each vertex v in V^c corresponds to a single sector of its ancestor $S^r(a(v))$. We denote the sector at $a(v)$ corresponding to a vertex v of M^c by $S(v)$. The constraint (4) for a sector $S^r(a(v))$ becomes a constraint on the total vertex angle: if there is only a single sector, and no feature edges, then it is a non-feature vertex constraint (1), otherwise it is a constraint on a boundary vertex total angle. The explicit form of the constraint is

$$\sum_{f_{ijk} \ni v_i} \alpha_{ijk} = \hat{\Theta}(S^r(a(v_i))) \tag{10}$$

The feature graph partitions the mesh into connected components. Each component may have parts of multiple feature graph components on its boundary. As these are disjoint, each corresponds to a separate boundary loop of M^c. The feature component constraints (5) become parametric angle constraints between boundary components. The paths for both the new vertex constraints and boundary component constraints are still open, but now we can apply the double cover approach, following [Campen et al. 2021], to convert them to loops. Similarly to vertices, we have a map for edges $a : E^c \to E$, with two edges mapped to each feature edge of M, and one for the rest. As we choose the path between e_0 and e_{M^r} not to pass through feature edges, it maps to a path in one of the components of M^c. The form of the constraint (5) remains exactly the same, but now the summation is over edges $e_m^{t,c}$ in M^c, such that $a\left(e_m^{t,c}\right) = e_m^t$.

Boundary length constraints. The above angle constraints are all that is required for a seamless parametrization of a mesh with boundary. However, for the features, the lengths of the two edges corresponding to the feature on the boundary should match: $\lambda(e_i) = \lambda(e_j)$, if $a(e_i) = a(e_j)$. As the logarithmic lengths are our variables, this would be a simple additional linear constraint. However, as the mesh connectivity may change, a feature edge may no longer be present in the mesh, and we need a more complex form of this constraint, which we define explicitly below.

Conversion to a closed mesh. While Capouellez and Zorin [2024] does not describe seamless parametrizations of meshes with boundary, the extension to open meshes for conformal maps with just vertex angle constraints was presented in Campen et al. [2021]. The theory of Penner coordinates is only valid for closed meshes, as the intrinsic Delaunay triangulation is an essential part of it. The idea is to extend a mesh with boundary to a closed surface by creating a symmetric double cover M^d, maintain symmetry throughout the optimization process, and extract the final parametrization for meshes by restricting it to one of the symmetric halves. Note that the edges on the boundary are allowed to be modified: now the feature edges, corresponding to the boundary edges in M^c, correspond to the symmetry line of M^d, which may not have an edge on it but can be easily constructed by splitting faces crossing it, which are tracked throughout the optimization.

Double cover. Let $M^{c\prime}$ be a mirrored copy of M^c, reversing the orientation of all faces. We glue M^c and $M^{c\prime}$ together along the common boundaries, identifying the vertices and edges on the boundary, to produce a single closed mesh M^d; we denote the line on M^d along which M^c was glued to $M^{c\prime}$ by Σ. M^d has a reflection symmetry; there is a natural involution map R that maps vertices, edges, and faces to their reflected copies. Given a discrete metric (or Penner coordinates) ℓ defined on M^c, we extend it to $M^{c\prime}$ (and thus M^d) by defining $\ell(R(e)) = \ell(e)$. That is, edge length is preserved by R.

The ancestor map a for the vertices of M^d maps them to the vertices of M^c. In particular, each vertex on the symmetry polyline of M^d corresponds to a sector of a feature vertex of M, and a boundary vertex of M^c.

Now the sector constraints, already transformed into boundary vertex constraints, take the form of vertex constraints (1) on M^d: we consider the union of two paths in M^d around a symmetry vertex v, one in M^c, and the other in $M^{c\prime}$ which form a closed loop, and the total angle in the constraint just doubles. The same is true for the component constraints, already transformed into constraints between boundary loops. Taking the union of the path defined by the edge sequence $e_0^{t,c} \dots e_{M_t}^{t,c}$ and $R\left(e_0^{t,c}\right) \dots R\left(e_{M_t}^{t,c}\right)$, we obtain a closed loop γ_t^d passing through $e_0^{t,c} = R\left(e_0^{t,c}\right)$ and $e_{M_t}^{t,c} = R\left(e_{M_t}^{t,c}\right)$, yielding *vertex and holonomy* constraints:

$$\begin{aligned}\sum_{f_{ijk} \ni v_i} \alpha_i^{jk} &= \hat{\Theta}\left(a(v_i)\right), \text{ for } v_i \text{ not on } \Sigma \\ \sum_{f_{ijk} \ni v_i} \alpha_i^{jk} &= 2\hat{\Theta}\left(S^r(a(v_i))\right), \text{ for } v_i \text{ on } \Sigma.\end{aligned} \tag{11}$$

$$\begin{aligned}\sum_{m=1}^{N_r} d_m^r \alpha_m^r &= \kappa_r, \text{ for } \gamma_r \text{ not crossing } \Sigma,\ r = 1 \dots 4g. \\ \sum_{m=1}^{N_t} d_m^t \alpha_m^t &= 2\kappa_t^{fc}, \text{ for } \gamma_t^d \text{ crossing } \Sigma,\ t = 1 \dots N_{cc} - 1.\end{aligned} \tag{12}$$

We note that although there are twice as many non-symmetry vertex constraints compared to the number of non-feature vertices in M, only one of each symmetric pair needs to be enforced. Similarly, for holonomy constraints out of $4g$ constraints, only $2g$ need to be enforced. Finally, we observe that the loops composed of two component constraint paths are independent of all other loops: as the paths on M^d connect two boundary components, after the double construction, the resulting path is a loop around a tunnel (Figure 5) that forms on the double surface, which was not present in M^c. The total genus of M^d is $2g + N_{cc} - 1$, so we expect $4g + 2N_{cc} - 2$, holonomy constraints. However, the holonomies of loops following the boundary of M^c are determined by the holonomy of the symmetry line, which is in turn determined by the boundary vertex angles because of symmetry enforced on M^d edge lengths, so these are not needed.

We retain the length constraints between pairs of edges on M^c for M^d. As these are constraints on lengths on the symmetry line Σ, their number remains the same.

To summarize, we have converted the feature sector and component constraints to vertex and holonomy constraints on the double of the cut mesh. The symmetry of the double cover ensures that the angles on the two halves of the paths around symmetry vertices and holonomy paths obtained from component paths are equal, i.e., the original constraints are satisfied. The important difference, however, is that now the constraints are compatible with mesh connectivity changes needed by the Penner-coordinate algorithm [Capouellez and Zorin 2024].

Updating constraints for changing connectivity. An essential part of the Penner-coordinate-based algorithms is, for a given choice of Penner coordinates on a fixed connectivity M_0^d, converting these to edge lengths on a different connectivity $M^{d'}$ for which these converted lengths are Delaunay, and computing all geometric quantities there, including the angles and lengths involved in the constraints. An additional complicating factor is that with the change of triangulation, the dual loops involved in the holonomy constraints may change.

For the double M^d, following [Campen et al. 2021], we use symmetric flips, i.e., if an edge e not crossing Σ, is flipped, we also flip $R(e)$. We note that flipping an edge entirely on Σ leads to a symmetric edge crossing Σ. However, the case of an edge with one vertex on Σ requires special attention. In this case, $R(e)$ and e cannot be simultaneously flipped symmetrically. This is related to the fact that a symmetric Delaunay tesselation may have stable configurations of two triangles, forming a symmetric trapezoid, for which choosing either diagonal is valid. A complete list of the possible stable configurations for symmetric flips is given in Campen et al. [2021].

Vertex and holonomy constraints are updated exactly as discussed in Capouellez and Zorin [2024]. The vertex angle constraints remain the sum of angles around a vertex, although some edges may be added or removed.

The boundary length constraints, not present in the previous work, are updated as follows. A feature edge of the original mesh, which became a boundary edge of the mesh M^c, is a symmetry-line edge on M^d in the initial configuration. The flip algorithm applied to M^d to obtain $\mathrm{Del}(M^d, \lambda)$ for a choice of Penner coordinates λ, may eliminate the edge, which would correspond to a line segment connecting two vertices on Σ, but not in the triangulation. It is a straight line in the parametric domain, albeit composed of multiple segments. More precisely, for any edge e with endpoints v_i, v_j in the boundary of M_0, the symmetry structure ensures one of the following holds:

(1) e remains an edge in M.
(2) There are symmetric paths of edges $e_0 \dots, e_n$ and $R(e_n), \dots, R(e_0)$ that bound a union of triangles and quads that cross the section of the line of symmetry between v_i and v_j (Figure 6).

In the first case, the length of the edge $\ell(e)$ can be directly recovered from the intrinsic metric. In the latter case, let $t_0, q_1, \dots q_n, t_1$ be the sequence of faces of $\mathrm{Del}(M^d, \lambda)$, starting and ending with a triangle, and with quad faces in between, crossing e of the original triangulation M^d.

The line of symmetry must, by the reflectional symmetry, be the path between v_i and v_j intersecting these transverse segments at their midpoints. Using the fact that all edges cross the symmetry line orthogonally, for the face t_0 we can compute the length of the edge from the vertex v_i on the symmetry line to the midpoint of the perpendicular opposite edge $e_0^\perp$ using the Pythagorean theorem as follows:

$$\ell_{t_0}^2 = \ell(e_0)^2 - \left(\frac{\ell(e_0^\perp)}{2}\right)^2 \tag{13}$$

and similarly for t_1. The situation is only slightly more complicated for a quadrilateral q_m, which by symmetry must be an isosceles trapezoid, where the length of the line between the midpoints of the transverse edges $e_{m-1}^\perp$ and $e_m^\perp$ can be computed as follows (see Figure 6):

$$\ell_{q_m}^2 = \ell(e_m)^2 - \left(\frac{\ell(e_m^\perp) - \ell(e_{m-1}^\perp)}{2}\right)^2 \tag{14}$$

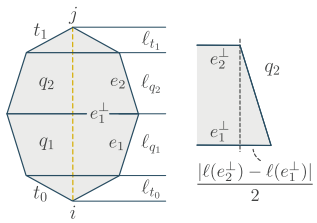


Fig. 6. A possible connectivity near a feature edge on the flipped double cover mesh M^d.

The total length of the symmetry-line edge is, for an edge e that is no longer in the mesh due to flips:

$$\ell_e = \ell_{t_0} + \ell_{q_1} + \cdots + \ell_{q_m} + \ell_{t_1} \tag{15}$$

where the terms of the expression are defined by (13) and (14). If two edges e, e' in M^c arise from the same feature, i.e., $a(e) = a(e')$, then we add the logarithmic variable constraint

$$B_{a(e)}(\lambda) = 2\ln \ell_e(\lambda) - 2\ln \ell_{e'}(\lambda) = 0 \tag{16}$$

Summary. The overall system of constraints we aim to solve consists of the constraints (11), (12), and (15), with the angles α expressed as functions of the (logarithmic) Penner coordinates λ, which we concisely write as

$$F^a(\lambda) = \begin{bmatrix} C\alpha(\lambda) - \hat{\Theta} \\ B(\lambda) \end{bmatrix} = 0 \tag{17}$$

7 THE ALGORITHM

7.1 Constraint relaxation

As discussed in Section 4, solving the problem with the full constraint set is not always possible. For this reason, we construct two constraint systems *all constraints* $F^a(\lambda) = 0$ defined above, and a reduced system of *hard constraints* $F^h(\lambda) = 0$, which is a subset. All length constraints are retained, but angle constraints are modified.

We describe a constraint-reduction procedure that reduces the number of constraints one-by-one and is equivalent to replacing a single feature edge with a regular edge. We construct a forest of spanning trees T on the feature graph G and obtain F^h as a system of constraints for edges excluding $E(G) \setminus E(T)$, where $E(\cdot)$ is the set of edges of a graph. We impose the additional constraint (a) that every feature vertex is adjacent to at least one edge in $E(T)$ and (b) that every boundary component in M^c contains at least one edge in $E(T)$. Note that this means that not all feature edges can be relabeled, but we can, in the extreme case, reach the situation when all feature edges are isolated. As we have not observed the need to relax many edges beyond those needed to reduce a feature graph to a tree, we do not consider additional transformation cases that would be needed to eliminate this restriction.

The feature constraints are defined per vertex sector (4) and path connecting two graph components (5). It is more convenient to reason in terms of the cut mesh M^c, for which these constraints become boundary vertex and boundary loop alignment constraints.

A feature edge e in M corresponds to two separate edges on the boundary of M^c, while non-feature edges correspond to a single edge of M^c. When a single feature edge becomes a non-feature edge, this corresponds to merging two edges on the boundary of M^c. The constraint that no vertex is isolated by the relaxation ensures that all four endpoints of the two boundary edges are distinct, so this merging modifies the constraint system as follows:

- Four boundary vertex constraints are dropped;
- Two boundary vertex constraints are added;
- One boundary-to-boundary constraint is added.

Two pairs of boundary vertices are identified, creating two new vertices. If the angles assigned to the original vertices were $\hat{\Theta}(s(v_m)) = \hat{\Theta}_{m0}$, and $\hat{\Theta}(s(v'_m)) = \hat{\Theta}_{m1}$, $m = i, j$, these four constraints are dropped, and the new vertex constraints are

$$\sum_{f_{mrt} \ni v_m} \alpha^m_{rt} = \hat{\Theta}_{m0} + \hat{\Theta}_{m1}, \ m = i, j.$$

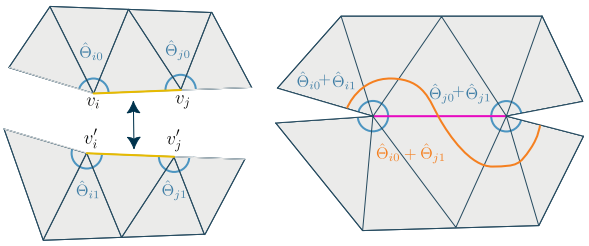


Fig. 7. Transformation of constraints for relaxation of a single edge. Left: four vertex constraints to be removed. Right: two new vertex constraints added, and one boundary-to-boundary constraint.

The additional constraint we add is along a path connecting the edge e_{pi} preceding v_i on the boundary, and $e_{jn'}$, succeeding v'_j with the angle $\hat{\Theta}_{i0} + \hat{\Theta}_{j1}$.

As a boundary-to-boundary constraint is only non-redundant if it connects two boundary loops, we clarify why it needs to be added in all cases. The transformation of the boundary components of M_c is shown in Figure 8. In case 1, a boundary loop of M^c is split into two components, and an additional boundary-to-boundary constraint needs to be added. In case 2, topology changes, and a holonomy constraint needs to be added; one can show that it can be expressed as a composition of a similar boundary-to-boundary constraint as in case 1, and vertex constraints. Finally, in case 3, two connected components of M_c are linked. As one of the vertex constraints in each component is redundant by the Gauss-Bonnet theorem, once components are merged, only one of the two is left redundant; the other needs to be added back. Adding a boundary-to-boundary constraint instead is sufficient, as the missing vertex constraint can be expressed as a linear combination of this constraint and other vertex constraints, so the transformation shown in Figure 7 is sufficient.

The angle equations of the system $F^h(\lambda) = 0$ are obtained by applying the transformation described above (remove 4 constraints, replace with 3) repeatedly.

Note that these are not specific to the set of feature edges we eliminate. In our algorithm, we found that reliable results can be obtained by starting with the forest of spanning trees. However, if failure is detected, but the seamless parametrization algorithm without features succeeds, additional constraints can be removed.

7.2 Algorithm

Algorithm 1 shows the pseudocode for the main components of our algorithm. The main idea of the algorithm is to approximate the solution of $\min \|F^a(\lambda)\|^2$, subject to the constraint $F^h(\lambda) = 0$. We assume that the relaxation from F^a to F^h is sufficient for $F^h(\lambda) = 0$ to be reliably solved by the method of Capouellez and Zorin [2024]: function ProjectToHardConstraint is exactly the extended Newton algorithm from this work, with a different set of

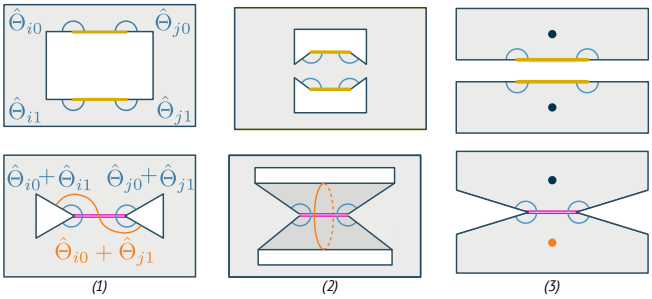


Fig. 8. Removal of a single tagged edge e_{ij}, effects on M^c. (1) The two edges in M^c corresponding to e_{ij} are on the same boundary loop; the operation splits it into two. (2) Two edges are on different boundary loops of the same component; the operation creates a handle. (3) Two edges are on boundary loops of different components; the operation glues these together. Black dots indicate vertices at which redundant vertex constraints are dropped in each component, and the orange dot is the constraint that needs to be reintroduced.

constraints including length. The outer loop in FeatureSeamless similarly solves the full constraint system but enforces the hard constraints at every step. As discussed in Capouellez and Zorin [2024], this approach, involving a pseudoinverse of the Jacobian of the constraints at every iteration, finding a minimal norm solution to the linearized constraints, is effectively approximating minimizing the metric distortion $\|\lambda - \lambda_0\|$ at every iteration, so the algorithm is driving the solution towards constraints with minimal solution norm change at every step. Of course, if the constraints are infeasible, it will never solve $F^a(\lambda) = 0$. Our stopping criterion for the outer iteration is either finding a solution, or degeneration of the triangles of the Delaunay mesh for the current iteration of Penner coordinates λ, which we found to be correlated with infeasibility, or its step decreases close to zero.

We note that the ProjectToHardConstraint can, in principle, also fail similarly, as we have no mathematical guarantees of feasibility for the reduced system $F^h(\lambda) = 0$ or that the iteration converges to a feasible solution even if one exists. An additional iteration preceding this algorithm is needed to find a system of constraints that produces a solution. As a fallback in failure cases, we use a simple greedy algorithm: arbitrarily remove edges from $E(T)$ without isolating feature vertices or removing all edges in a boundary component of M^c, until no more edges can be removed. If this also fails, we simply fall back to the version of the algorithm without feature alignment constraints. Further details of the algorithm are provided in Appendix C.

7.3 Stitching and Simplification

Stitching. Once the metric is determined, we construct an explicit parametrization for the mesh. Capouellez and Zorin [2023] a method to produce a refinement of a single component, possibly with boundary, by tracking the intersection points of flipped edges with the original connectivity and solving an optimization problem to determine the barycentric coordinates of the intersections on the edge. The metric induced on this refined mesh determines a mapping to the plane via a layout procedure akin to that of Springborn et al.

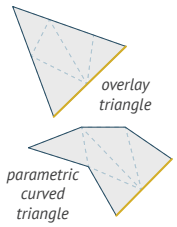
Algorithm 1: Seamless feature-aligned parametrization algorithm.

Input : manifold mesh with features $M = (V, E, F, E_c)$, lengths $\ell^0 = e^{\lambda^0/2}$ satisfying triangle inequality, target angles $\hat{\Theta} > 0$ at non-feature vertices, sectors of feature vertices, a basis of dual loops, and paths between feature graph connected components, respecting the Gauss-Bonnet theorem.

Output : triangle mesh $\tilde{M} = (V, \tilde{E}, \tilde{F})$, edge lengths $e^{\tilde{\lambda}/2}$ satisfying triangle inequality, with angles $\max_\Theta \|\Theta - \hat{\Theta}\| \leq \epsilon_c$, and a subset of feature edges E_c, satisfying constraints.

Function FeatureSeamless($M, \lambda^0, \hat{\Theta}$)**:**
  $\lambda \leftarrow$ ProjectToHardConstraint(M, λ_0)
  while not
  Converged(M, λ) **do**
    $\tilde{M}, \tilde{\lambda}, D, C \leftarrow$ DiffMakeDelaunay(M, λ, C)
    $L \leftarrow \nabla F^a (\nabla F^a)^T$
    Solve $L\mu = -F^a$
    $d \leftarrow (\nabla F^a)^T \mu$
    $\beta \leftarrow$ ProjectedLineSearch(λ, d)
    $\lambda \leftarrow \lambda + \beta d$
  return
  MakeDelaunay(M, λ)
Function ProjectToHardConstraint($M, \lambda^0, \hat{\Theta}$)**:**
  $\lambda \leftarrow \lambda_0$ **while not***Converged*(M, λ) **do**
    $\tilde{M}, \tilde{\lambda}, D, C \leftarrow$ DiffMakeDelaunay(M, λ, C)
    $L \leftarrow \nabla F^h (\nabla F^h)^T$
    Solve $L\mu = -F^h$
    $d \leftarrow (\nabla F^h)^T \mu$
    $\beta \leftarrow$ LineSearch(λ, d)
    $\lambda \leftarrow \lambda + \beta d$
  return
  MakeDelaunay(M, λ)
Function *ProjectedLineSearch*(λ, d)**:**
  $\beta \leftarrow 1$
  $\lambda \leftarrow$ ProjectToHardConstraint($\mathcal{M}, \lambda_0 + \beta d$)
  while $\|F^a(\lambda)\|_2^2 > \|F^a(\lambda_0)\|_2^2$ **do**
    $\beta \leftarrow \beta/2$
    $\lambda \leftarrow$ ProjectToHardConstraint($\mathcal{M}, \lambda_0 + \beta d$)
  return λ
Function *DiffMakeDelaunay*(M, λ, C)**:**
  $\tilde{M}, \tilde{\lambda} \leftarrow M, \lambda$
  $D \leftarrow$ Id
  $Q \leftarrow \{e | \text{NonDelaunay}(M, \lambda, e)\}$
  while $Q \neq \emptyset$ **do**
    remove e from Q
    $\tilde{M}, \tilde{\lambda} \leftarrow$ PtolemyFlip($\tilde{M}, \tilde{\lambda}, e$)
    $D \leftarrow$ DiffPtolemy($\tilde{M}, \tilde{\lambda}, e$) $\cdot D$
    $C \leftarrow$ UpdateConstraints($\tilde{M}, \tilde{\lambda}, C, e$)
  return $\tilde{M}, \tilde{\lambda}, D, C$

[2008]. The particular layout depends on a cut to disk of the refined mesh; we use a cutgraph that only contains edges in the original mesh and not the inserted edges that cut triangles transversely, so the parametrization maps refined triangles in the embedded mesh to connected curved triangles in the parametric domain (see inset).



The extension to multiple components is straightforward, but additional considerations are necessary to stitch components together across the feature edges to construct a single manifold mesh with a parametrization. Since boundary edges may be refined in the overlay mesh and the two edges split by a feature are decoupled, they may have different refinements that need to be stitched together.

Simplification. The final stitched mesh includes edges of both M and the edges of the final mesh obtained from $\mathrm{Del}(M_d, \lambda^f)$, where λ^f is the final metric produced by the algorithm. Capouellez and Zorin [2023] also describe a simplification method to remove refinement where possible without introducing flipped triangles in the parametric domain. This method does not depend on how the refined mesh was produced; it only requires a refined mesh with annotations of the mappings from refined triangles to the original face containing it and from refined vertices to the endpoints of the edge containing it. By some simple additional bookkeeping, we can maintain this information during the stitching process and then apply this simplification procedure directly. Since the inserted vertices on refined feature edges are colinear, this simplification naturally preserves feature alignment.

8 EVALUATION

Datasets. Following previous work, we use two datasets in our evaluation: Myles et al. [2014] and a dataset derived from [Zhou and Jacobson 2016], similar to [Capouellez and Zorin 2024], i.e., obtained by tetrahedral remeshing and extraction of manifold surfaces. As a large fraction of meshes in Zhou and Jacobson [2016] are nonmanifold, and have self-intersections, degenerate triangles, and other flaws, this makes a large percentage unsuitable for testing. Our preprocessing allows us to increase the number and topological complexity of meshes we use, simultaneously improving mesh quality. While feature preservation by TetWild [Hu et al. 2018], which we use for remeshing, is imperfect, these are preserved sufficiently well to test robustness. A simple dihedral-based sharp feature detector was used to generate features automatically. This is not the best approach in many cases: we view curved feature detection as a separate important problem, and our purpose is to test the robustness of the method and the naive feature detector results in challenging feature configurations.

Once features are detected, we use an implementation of Bommes et al. [2009] to generate cross-fields on the surface, using feature edge directions as additional constraints. The iterative rounding step of this method was modified to avoid snapping feature sector angle constraints to 0, which are unsatisfiable. A small number of meshes for which the resulting global vertex angle constraints were also theoretically unsatisfiable (3-5 torus topology [Shen et al. 2022], and tori without cones but with nontrivial holonomy) were modified with simple heuristics (e.g., random cone pair insertion) to make them feasible. In total, we obtained 17,421 meshes by this procedure. Statistics for this dataset are shown in Figure 12, and details of our procedure are provided in Appendix B

Algorithm robustness and constraint satisfaction. The algorithm obtained a feature-aligned seamless parametrization for every mesh in both the dataset of Myles et al. [2014] and the dataset derived from Zhou and Jacobson [2016] dataset, except for one simple mesh with torus topology and four cones. While a seamless parametrization in this case exists, we hypothesize that due to unfortunate cone location, it may be distorted enough to cause problems (Figure 9).

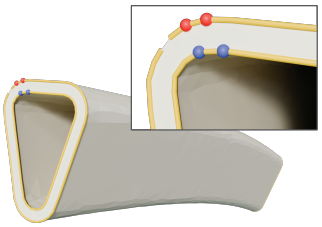


Fig. 9. The mesh for which the parametrization algorithm with feature alignment did not converge. The fallback algorithm with no feature alignment constraints converges as expected. Note the unfortunate location of cones, generated by the cross-field optimization; there are two close pairs of cones, with valences 3 and 5 on the feature curves, and no other cones on the mesh.

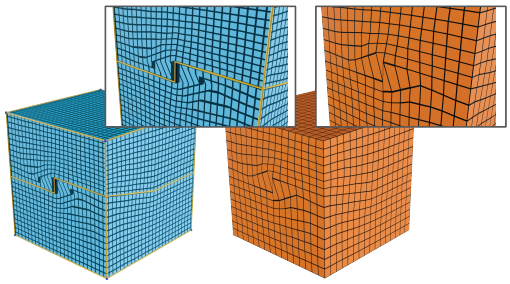


Fig. 10. For an impossible configuration from Section 1, our method relaxes one of the edges, which ends up misaligned. In the visualized example the vertical edge is relaxed. Note that in the resulting parametrization, it is forced by the constraints on other edges to be aligned, but also with the u direction, violating constraints on the sector angles at its endpoints.

It produces parametrizations for which all angle constraints (at cones, on holonomy loops, and alignment) are satisfied with at least 10^{-10} accuracy. All parametric-domain triangles in all parametrizations are guaranteed to have the correct orientation, which is validated by exact predicates [Shewchuk 1996].

As the alignment constraints for some edges are imposed as soft constraints, we report the distribution of the resulting error.

For 16,444 meshes, all soft constraints were satisfied with 10^{-10} accuracy. The distribution of the errors for the remaining constraints is shown in Figure 14. For 54 meshes, the initial projection to hard

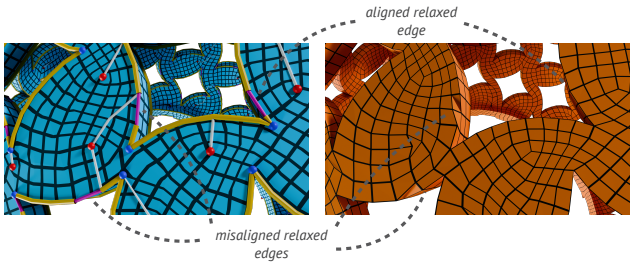


Fig. 11. An example of misaligned relaxed edges on a model in our dataset. Note that many of the other nearby relaxed edges are actually aligned.

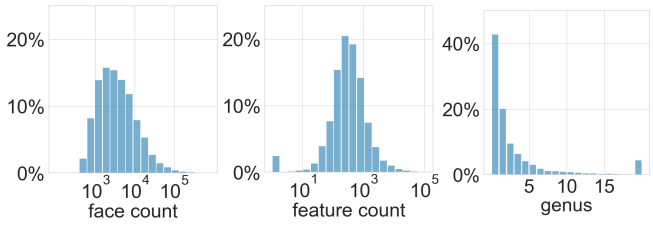


Fig. 12. Thingi10k remeshed dataset statistics. Outliers are aggregated in the rightmost bin.

constraints did not converge or the resulting metric was was too degenerate for downstream applications such as, e.g., quadrangulation, and the fallback method described in Appendix C was employed. We emphasize that the convergence failure in these cases does not necessarily imply that a solution does not exist. Capouellez and Zorin [2024] observed that some form of intrinsic pre-processing is necessary for the convergence of the Newton's method for poorly conditioned meshes. While our remeshed models are initially high quality, challenging feature alignment constraints can quickly result in degraded triangle quality. Indeed, for many of our fallback cases, simply applying the interpolation towards regular Penner coordinates used in Capouellez and Zorin [2024] resulted in convergence without relaxing any additional hard feature constraints.

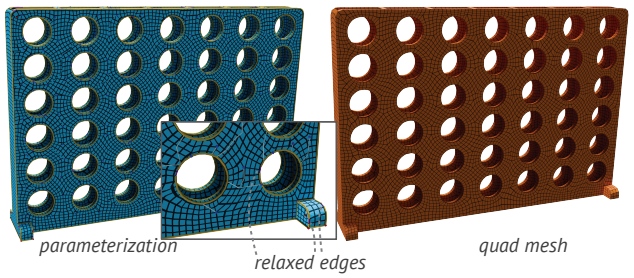


Fig. 13. Left: a seamless parametrization showing hard constraints (yellow) and relaxed feature edges which were perfectly aligned by the algorithm. Right: resulting quad mesh. Note that almost all relaxed feature edges actually do get aligned by our algorithm, resulting in aligned quadrangulations.

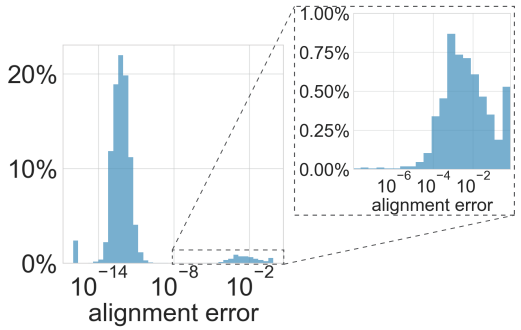


Fig. 14. Distribution of feature alignment errors (worst error per mesh), measured in radians. The histogram on the right shows the zoomed-in view of edges with higher deviation from constraints. The number of meshes with the worst-case alignment error exceeding 0.01 radian, i.e., around 6 degrees, is 346. Meshes with no features trivially have an alignment error of 0.

Metric optimality. Similarly to Capouellez and Zorin [2024], implicitly our algorithm approximately minimizes the norm $||\lambda - \lambda^0||_2$. Figure 15 shows the distributions of distortion with and without feature constraints. There is a significant increase in distortion, but overall, our method keeps the distortion within a reasonable range.

Runtime. As with Capouellez and Zorin [2024], the main bottleneck of our seamless parameterization method is the linear solve to find the modified Newton descent directions. The total runtime of Algorithm 1 is then roughly estimated as the product of the number of linear solves and the average solve time.

For our method, we found that the average number of solves required was 18 and the average solve time was 0.155 seconds. On the same dataset with feature constraints removed, the method of Capouellez and Zorin [2024] had an average solve count of 10 and an average solve time of 0.043 seconds. This increase in runtime is expected as we have more constraints to enforce. Figure 16 shows the distributions of solve counts and solve times for both.

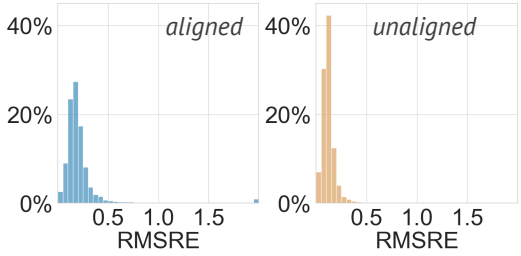


Fig. 15. Distribution of metric errors with and without feature alignment.

Non-degeneracy. While all our parametrizations are valid, some may have triangles close to degenerate (Figure 17). We show the distributions of minimal angles and minimal exterior angles (i.e., the differences between maximal angles and π) for the intrinsic mesh $\mathrm{Del}(M^d, \lambda^f)$ (see Section 5) associated with the parametrization

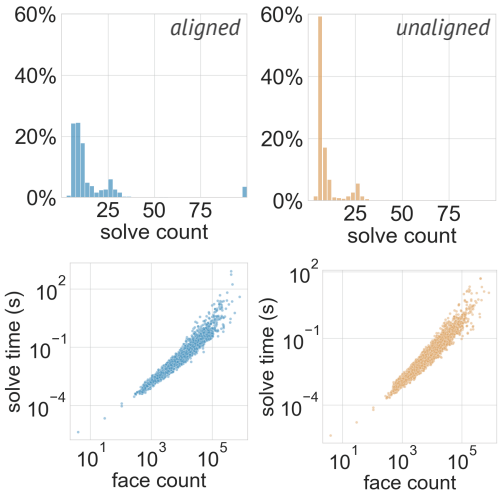


Fig. 16. Distribution of total linear solves required with and without feature alignment. We also plot the average time of the linear solves over the mesh size. With alignment, the average solve count is 18 with an average solve time of 0.155 seconds. Without alignment, the corresponding results are 10 solves and 0.043 seconds.

metric and the refinement of the original mesh needed to remap the parametrization of the intrinsic mesh to the original.

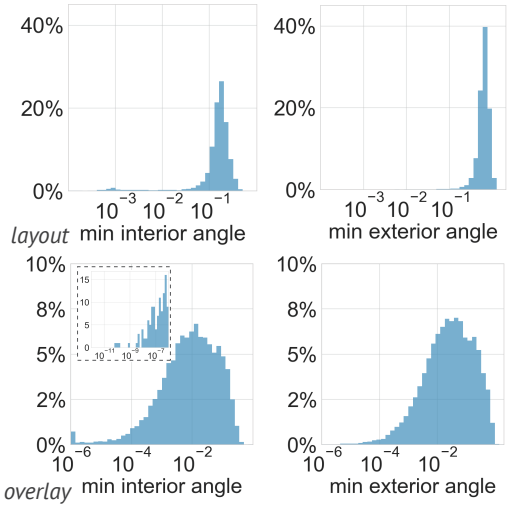


Fig. 17. Distribution of minimal angles and exterior angles for the intrinsic parametrization metric mesh and the refined original mesh. Upper row: distribution for the intrinsic mesh $\mathrm{Del}(M, \lambda^f)$. Lower row: distribution for the overlay mesh, including the distribution of 112 meshes with worst interior angle below 10^{-6}

Quadrangulation suitability. As one of the motivations for seamless parametrization is to provide a reliable starting point for quadrangulation algorithms, we combined our seamless parametrization with the algorithms of Campen et al. [2015] and Heistermann et al. [2023]. For completeness, we briefly summarize the quadrangulation pipeline here:

- Compute a seamless parametrization, P_s
- "Sanitize" it by perturbing the u, v positions, so that the seamless constraints are satisfied bit-exactly [Mandad and Campen 2019].
- Compute a T-mesh partition of the surface, by tracing u and v isolines on P_s [Campen et al. 2015].
- Quantize the parametric arc lengths using greedy Bi-MDF quantization [Heistermann et al. 2023], i.e., assign positive integers to the arclength of the T-mesh, so that the lengths of opposite sides of each quad are equal (To obtain coarser quad meshes, Heistermann et al. [2023] allows for zero-length assignments, which are later eliminated). For a seamless starting point, one always exists, for a sufficiently fine quad meshing scale [Campen et al. 2015; Heistermann et al. 2023].
- Compute an initial parametrization P_q^0 with quantized constraints, parametrizing per patch of the T-mesh.
- Optimize the parametrization, to obtain the final one, P_q, by minimizing the symmetric Dirichlet energy [Smith and Schaefer 2015] with P_s as a reference.
- Trace integer u and v isolines of P_q to obtain a quad mesh [Ebke et al. 2013].

An important technical aspect of these algorithms, as far as the input parametrization is concerned is *sanitization* (cf. [Mandad and Campen 2019]) which makes the seamless constraints exact before the next steps of the quantization algorithms. While our algorithm satisfies the seamless constraints very accurately, for triangles close to degenerate, even a tiny perturbation needed to make the constraints exact may result in an inverted triangle and parametrization not suitable for quadrangulation. For this reason, we measure how robust the parametrization is for such a perturbation process. Specifically, we use two measures: the ratio of the triangle height to the maximal error in the length of edges on which seamless constraints are imposed r_ℓ and the ratio of minimal angle or exterior angle to the error in the seam edge orientation r_α (Figure 18).

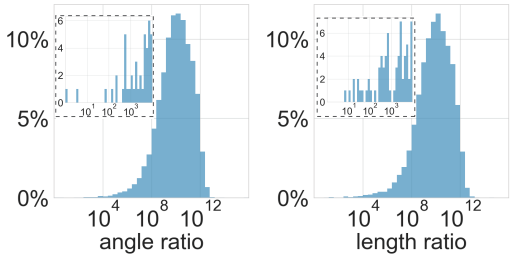


Fig. 18. Distribution of measures of sensitivity to vertex perturbation r_ℓ and r_{angle}, indicating whether the initial step of the quadrangulation process (sanitization) is likely to fail. Only 2 meshes were below 10.

For all models but 2, this ratio exceeds 10, and for these models, we can still successfully obtain a quadrangulation.

Figure 19 shows one extreme example demonstrating the robustness of our method. One model in our dataset has an extremely

spiky surface, with a large fraction of the edges identified as sharp by our naive feature tagging algorithm. The algorithm succeeds in this extreme case, producing a parametrization suitable for quadrangulation (the usefulness of such quadrangulation is unclear, we include this example as empirical evidence of robustness).

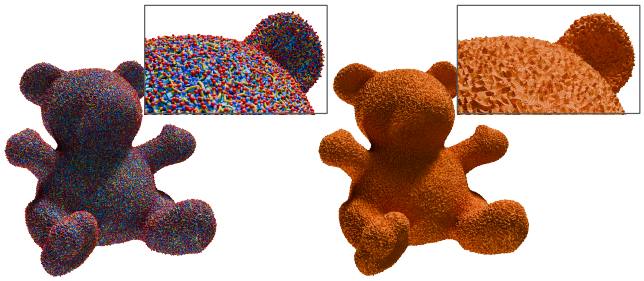


Fig. 19. A extreme model with dense random sharp features. This model has 85,846 cones, and 111,395 sharp edges, out of 838,947 edges. Left: seamless parametrization. Right: extracted quad mesh.

Figures 20, 21, 22 show a collection of models from the dataset with high geometric complexity and salient features based on Zhou and Jacobson [2016]. For each model, we show the seamless, feature-aligned parametrization, quadrangulation based on this parametrization, and histograms of the metric distortion as measured by the deviation per face of the symmetric Dirichlet energy from the optimal value. Our method handles meshes of high topological and geometric complexity, with genus as high as 4292, while preserving the input cross-field topology and feature alignment. Note that, although we are able to obtain a quadrangulation for the highest genus models in our dataset, these surfaces lack salient features, and our simple dihedral-based feature detection produces noisy feature tags akin to those in Figure 19.

Robustness compared to other methods. Our method succeeded in producing a valid parametrization (but possibly with some misaligned edges) on the closed meshes of the dataset introduced in Myles et al. [2014] and used in a few papers, with original connectivity and fields as input, as well as on all of 17,421 mesh dataset, except one model described above.

The method of Myles et al. [2014], uses field tracing (including tracing sharp features exactly) to create an initial T-mesh quad partition, which is then adjusted to make it metric-compatible, and, if this proves to be impossible, inserts cones in some of the quads, thus solving a somewhat different problem – our method aims to keep the holonomy signature fixed. By design, the initial T-mesh will include all sharp features. However, the algorithm may reroute some patch boundaries, including sharp features, in the process of T-mesh adjustment. Like our method, it produces a seamless parametrization on all inputs on the smaller dataset used (around 120 meshes), but inserts cones in some cases (4 meshes, around 3%) Our relaxation is different: we preserve the holonomy signature exactly but may relax some of the feature constraints (which may also be the case for Myles et al. [2014], due to T-mesh modifications required for approximately 30% of the meshes).

Three other methods were evaluated in that paper on the same dataset, to obtain a seamless parametrization, aligned with features if any are present: Bommes et al. [2013a, 2009] and MIQ combined with the convexified bijectivity constraints [Lipman 2012]. The former does not guarantee bijectivity, and the latter narrows the space of admissible solutions by convexifying the bijectivity constraints, and as a consequence did not find solutions in 25% and 17% cases respectively, as well as modified some cones.

Other recent methods, Campen and Zorin [2017] and Shen et al. [2022] do not address feature constraints, and the latter fails on the highest genus models in the Myles et al. [2014] dataset (6% of nontrivial genus models, highest genus around 100).

Another recent method, Pietroni et al. [2021], as far as we know, is the only quadrangulation method that was tested on a dataset derived from the Thingi10k dataset we are using. It uses a construction that avoids the need for seamless parametrization but changes the holonomy signature, similar to Myles et al. [2014]. In terms of robustness it shows a high, but still substantially lower success rate on Thingi10k (for 0.5% of inputs no quad mesh was produced (around 50 meshes out of 9877 in the dataset used), on a variety of models, including some of the more complex shapes. Figures 20,21, and 22 include some of these shapes among others.

In a sense, the approach we propose is complementary to the cone-inserting approaches of Myles et al. [2014] and Pietroni et al. [2021]: one can expect that e.g., for infeasible configurations, some are best handled by relaxing the feature constraints as we do, and other by adding cones.

9 LIMITATIONS AND FUTURE WORK

We have described an approach to solving the seamless parameterization problem with feature alignment robustly, at the cost of relaxing some of the alignment constraints. Our method is based on solving a system of nonlinear equations in intrinsic variables using a straightforward modification of Newton's algorithm while allowing mesh connectivity to change in the process, which is critical for robustness.

Limitations. While the robustness of our method is orders of magnitude higher (based on available data) compared to previously proposed techniques, just like Capouellez and Zorin [2024], it lacks a full theoretical justification of convergence, which is only available for genus 0 version of that method. Even more fundamentally, while the conditions for solution existence for a seamless parametrization problem without feature alignment are known, for the feature-aligned version we consider, there are no known conditions on the feature graph that guarantee that the solution exists.

Moreover, in cases where the soft constraints are not satisfied to machine precision, we have empirically observed that it is often possible to convert a large subset of these soft constraints to hard constraints without impacting convergence. While we have found our hard constraint trees to work remarkably well in practice, developing heuristics to relax as few edges as possible is a potential direction for future work.

Our method is substantially slower in many cases compared to Capouellez and Zorin [2024], as it may require multiple iterations, each requiring a hard constraint nonlinear solve, similar to

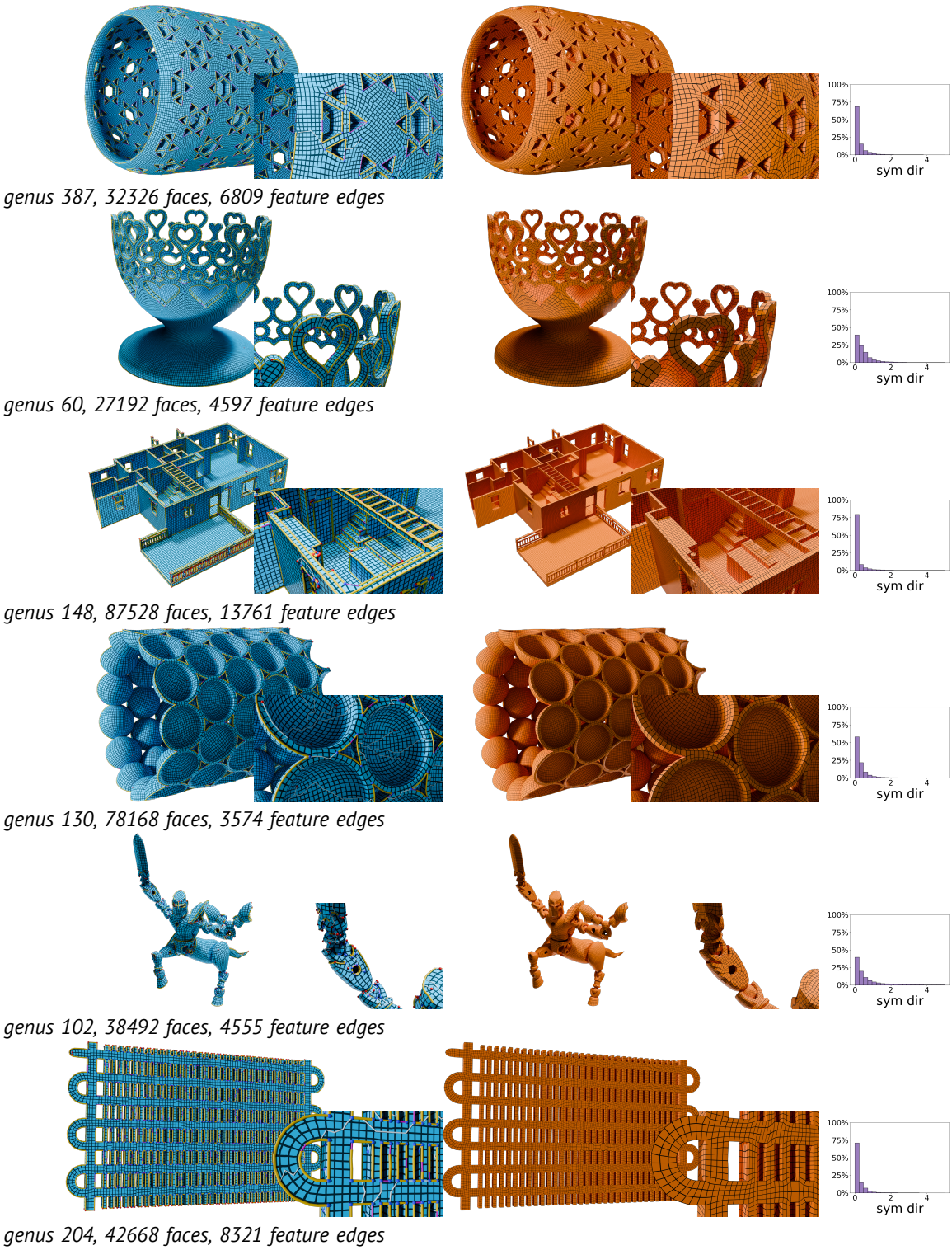


Fig. 20. Examples from our dataset with a complex feature graph, large number of cones, and high genus (set 1). Please note that feature detection was based on simple dihedral angle thresholding, and missing features were not present in the input, rather than removed by our algorithm.

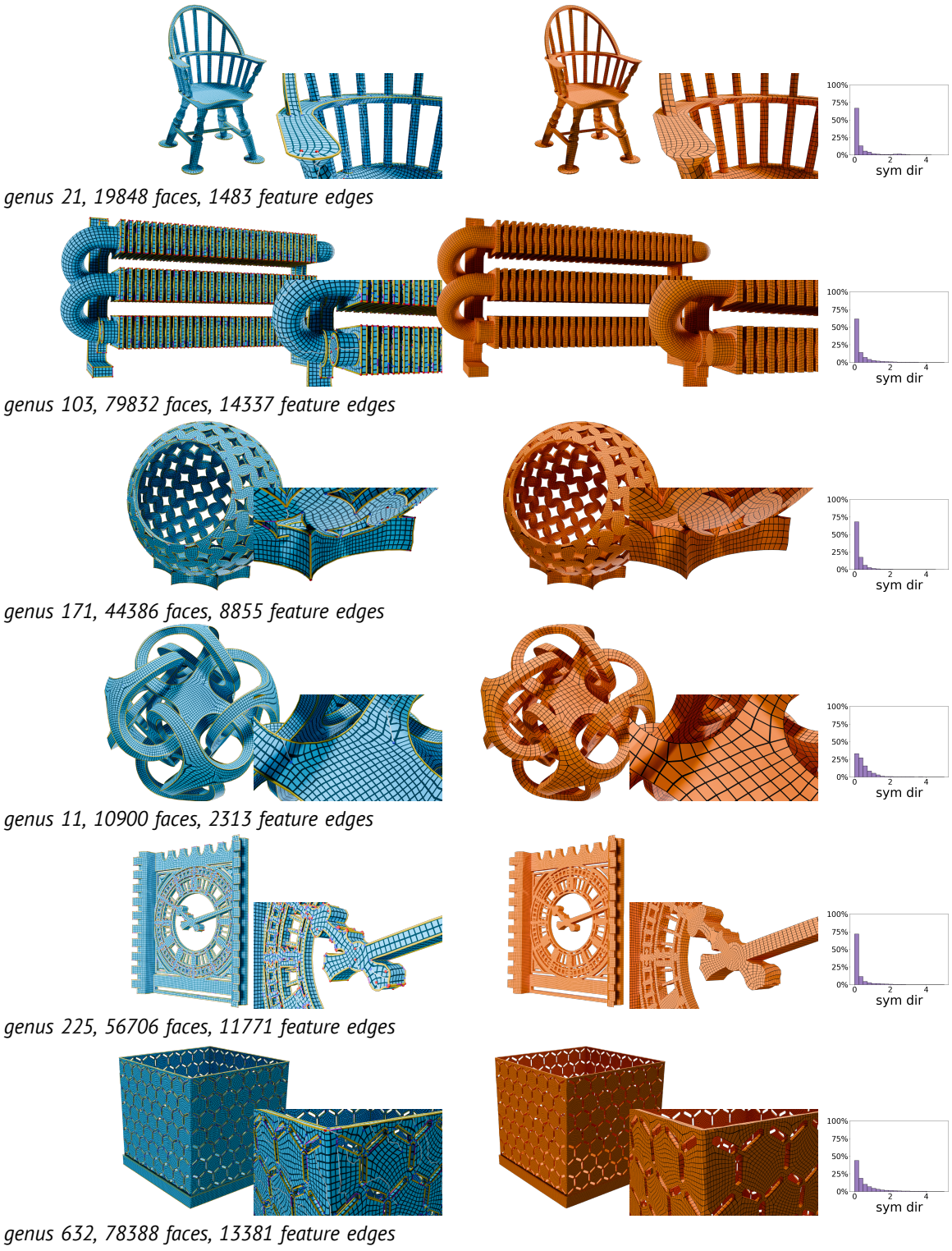
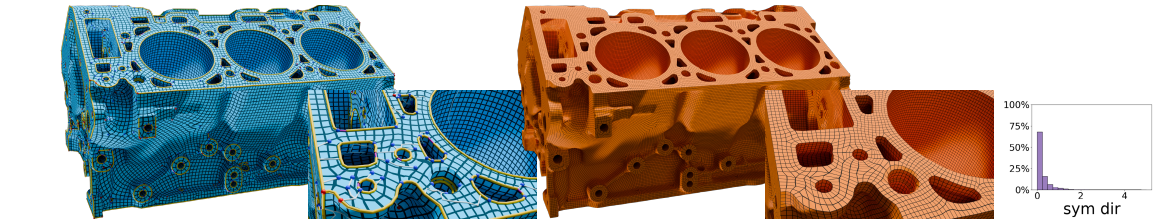
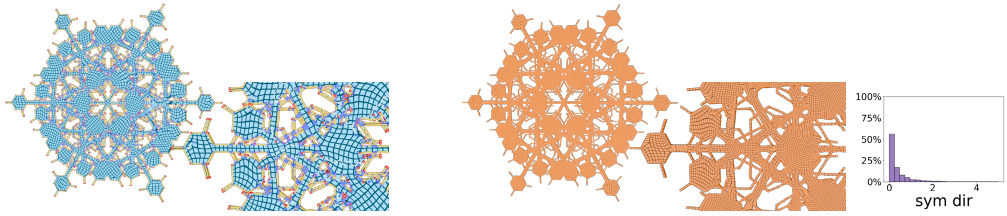


Fig. 21. Examples from our dataset with a complex feature graph, large number of cones, and high genus (set 2). Please note that feature detection was based on simple dihedral angle thresholding, and missing features were not present in the input, rather than removed by our algorithm.



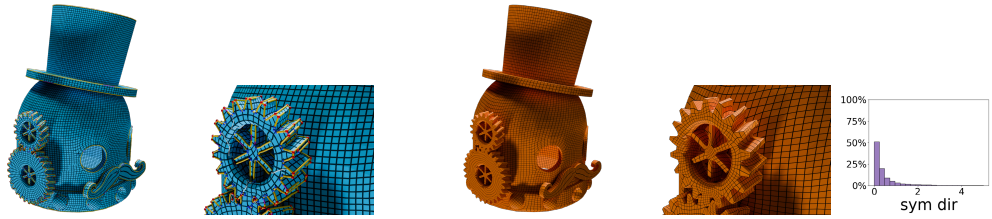
genus 132, 140460 faces, 6116 feature edges



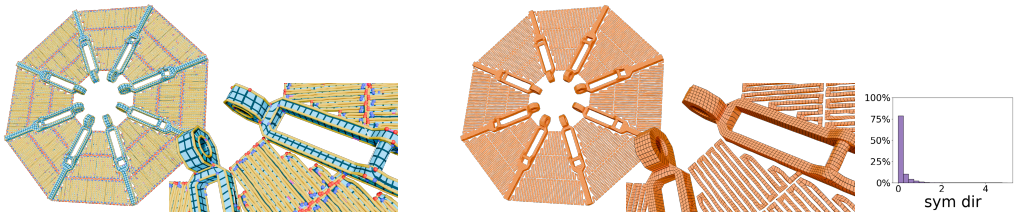
genus 452, 59946 faces, 11721 feature edges



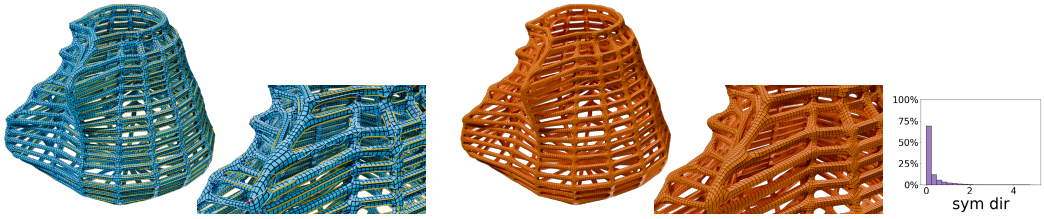
genus 0, 46458 faces, 4425 feature edges



genus 31, 22200 faces, 3009 feature edges



genus 41, 231008 faces, 41426 feature edges



genus 320, 61458 faces, 8900 feature edges

Fig. 22. Examples from our dataset with a complex feature graph, large number of cones, and high genus (set 3). Please note that feature detection was based on simple dihedral angle thresholding, and missing features were not present in the input, rather than removed by our algorithm.

Capouellez and Zorin [2023]. Furthermore, while the parametric-domain meshes with updated connectivity computed using the flip algorithm are of consistently good quality, remapping to the initial domain using an overlay mesh does result in a quality decrease, which can be addressed by an improved overlay construction.

Finally, our parametrization quality in many cases requires improvement by a post-optimization process based on fixed mesh connectivity and symmetric Dirichlet/field-alignment energy, which is often costly. Improving the quality of the parametrization obtained in Penner coordinates is a possible future direction.

ACKNOWLEDGMENTS

This work was supported in part through the NYU IT High Performance Computing resources, services, and staff expertise. This work was also partially supported by the NSF grants OAC-2411349 and IIS-2313156, and a gift from Adobe Research.

D. Bommes and M. Heisterman have received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (AlgoHex, grant agreement No 853343).

We would further like to thank Max Paik for assistance in creating diagrams.

REFERENCES

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

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.

Alon Bright, Edward Chien, and Ofir Weber. 2017. Harmonic Global Parametrization with Rational Holonomy. *ACM Trans. Graph.*36, 4 (2017).

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. https://doi.org/10.1145/3478513.3480557

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

Marcel Campen and Denis Zorin. 2017. Similarity maps and field-guided T-splines: a perfect couple. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–16.

Ryan Capouellez and Denis Zorin. 2023. Metric Optimization in Penner Coordinates. *ACM Trans. Graph.* 42, 6, Article 234 (dec 2023), 19 pages. https://doi.org/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. https://doi.org/10.1145/3658202

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. https://doi.org/10.1145/2508363.2508372

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. 2018a. A discrete uniformization theorem for polyhedral surfaces II. *Journal of Differential Geometry* 109, 3 (2018), 431–466.

Xianfeng Gu, Feng Luo, Jian Sun, and Tianqi Wu. 2018b. A discrete uniformization theorem for polyhedral surfaces. *Journal of Differential Geometry* 109, 2 (2018), 223–256.

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. https://doi.org/10.1145/3386569.3392385

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral meshing in the wild. *ACM Trans. Graph.* 37, 4 (2018), 60–1.

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.

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

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.

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

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

Max Lyon, Marcel Campen, and Leif Kobbelt. 2021. Quad layouts via constrained T-mesh quantization. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 305–314.

Manish Mandad and Marcel Campen. 2019. Exact constraint satisfaction for truly seamless parametrization. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 135–145.

Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. Robust field-aligned global parametrization. *ACM Trans. Graph.* 33, 4 (2014), 135–1.

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.

Daniele Panozzo, Enrico Puppo, and Luigi Rocca. 2010. Efficient multi-scale curvature and crease estimation. https://api.semanticscholar.org/CorpusID:17213450

Robert C Penner. 1987. The decorated Teichmüller space of punctured surfaces. *Communications in Mathematical Physics* 113, 2 (1987), 299–339.

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. https://doi.org/10.1145/3450626.3459941

Nico Pietroni, Enrico Puppo, Giorgio Marcias, Roberto Scopigno, and Paolo Cignoni. 2016. Tracing field-coherent quad layouts. In *Computer graphics forum*, Vol. 35. Wiley Online Library, 485–496.

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.

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

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.

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.

Jonathan Richard Shewchuk. 1996. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Workshop on applied computational geometry*. Springer, 203–222.

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

Boris Springborn. 2020. Ideal Hyperbolic Polyhedra and Discrete Uniformization. *Discrete & Computational Geometry* 64, 1 (2020), 63–108.

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. *Computer Graphics Forum* 35, 2 (2016).

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).

A BOUNDARY PATH LENGTH DERIVATIVES

The boundary are simple expressions in squared length coordinates, but we need to take derivatives of logarithmic lengths $\lambda_f = 2\log \ell_f$ with respect to logarithmic coordinates λ. The unflipped edge is trivial, and the flipped edge formula is

$$\frac{\partial B_e}{\partial \lambda_{e'}} = \frac{\partial \left(2\log \ell_e\right)}{\partial \lambda_{e'}} = \frac{2}{\ell_e}\left(\frac{\partial \ell_{t_0}}{\partial \lambda_{e'}} + \frac{\partial \ell_{q_1}}{\partial \lambda_{e'}} + \cdots + \frac{\partial \ell_{q_m}}{\partial \lambda_{e'}} + \frac{\partial \ell_{t_1}}{\partial \lambda_{e'}}\right)$$

The derivatives of the individual segment length derivatives are straightforward. For the transverse triangle t_0 with transverse edge $e_0^\perp$, we have:

$$\frac{\partial \ell_{t_0}}{\partial \lambda_{e_0^\perp}} = \frac{-\ell(e_0^\perp)^2}{8\ell_{t_0}}, \qquad \frac{\partial \ell_{t_0}}{\partial \lambda_{e_0}} = \frac{\ell(e_0)^2}{2\ell_{t_0}}$$

For a quadrilateral q_m, we have:

$$\frac{\partial \ell_{q_m}}{\partial \lambda_{e_{m-1}^\perp}} = \frac{-\ell(e_{m-1}^\perp)(\ell(e_{m-1}^\perp) - \ell(e_m^\perp))}{8\ell_{q_m}}$$

$$\frac{\partial \ell_{q_m}}{\partial \lambda_{e_m^\perp}} = \frac{-\ell(e_m^\perp)(\ell(e_m^\perp) - \ell(e_{m-1}^\perp))}{8\ell_{q_m}}$$

$$\frac{\partial \ell_{q_m}}{\partial \lambda_{e_m}} = \frac{\ell(e_m)^2}{2\ell_{q_m}}$$

As usual, the full gradient with respect to the Penner coordinates on the original connectivity can be computed by the chain rule as

$$\nabla_{\lambda^0} \lambda_f = \nabla_{\lambda} \lambda_f \cdot \nabla_{\lambda^0} \lambda$$

B DATASET

Manifold remeshing. We remesh all 10,000 models in the Zhou and Jacobson [2016] dataset with the method of Hu et al. [2020]. We split nonmanifold edges and vertices to produce a manifold surface mesh; since the method produces the boundary of a manifold tetrahedral mesh, this manifold splitting is always possible. We separate each (possibly disconnected) manifold surface into components. To avoid, e.g., the trivial isolated tetrahedra often produced by Hu et al. [2020], we discard separated components with fewer than 500 faces, and also discard any components with degenerate faces.

Feature inference. We tag all edges with a dihedral angle defect greater than 60° as a feature edge. If all edges of a triangle are tagged (a trivially impossible constraint for seamless parameterization with alignment), we remove the feature tags from these edges. To avoid spurious features, we also remove any feature graph components with four or fewer edges.

Hard feature edges. To construct the initial hard constraint subset, we compute a maximal spanning forest, with edges weighted by dihedral angle defect. In other words, we select the sharpest feature edges as hard constraints. In order to construct the feature component alignment constraints (Equation 5), we make the simplifying assumption that every boundary component in the feature cut mesh contains at least one hard constraint edge. Rather than attempting to find a spanning forest with this property, we use the following simple heuristic: if a boundary component does not contain at least one spanning forest edge, we refine an arbitrary edge in this component at the midpoint and add one of the two new edges to the spanning forest.

Cross-field optimizaiton. To ensure the cross-field can be feature aligned, we first refine the mesh to ensure every face is adjacent to at most one feature edge. We also estimate principal curvature directions on the feature cut mesh [Panozzo et al. 2010], and we identify salient directions with high anisotropy. We then generate a cross field with the method of Bommes et al. [2009] on the cut mesh, with:

(1) feature alignment for boundary faces,
(2) principal curvature direction constraints for salient interior faces, and
(3) iterative snapping modified to avoid 0 sector angles (which are unsatisfiable) and interior cones with angle $\pi/2$ (which result in self adjacent quads).

C ALGORITHM DETAILS

Infeasible constraint correction. After inferring vertex and holonomy constraints from the cross-field, we check for known infeasible holonomy signatures for seamless parametrization without features. The two known infeasible signatures, described in Shen et al. [2022], are:

- a torus with precisely two cones, with target angles $3\pi/2$ and $5\pi/2$, and
- a torus with no cones, but with nontrivial holonomy constraints.

In the case of a 3-5 torus where the cones are adjacent vertices, we attempt to remove the two cones without introducing any 0 sector angles. If this heuristic fails, we arbitrarily add two additional 3-5 cone pairs. In the case of a torus without cones, we simply remove any loop holonomy constraints.

Full constraint warm start. Before relaxing constraints, we first attempt to find a solution for *all* constraints using the modified Newton's method. If the method succeeds, we proceed directly to generating an explicit parameterization. If the method fails to satisfy all constraints to machine precision within 50 iterations, we use the partially converged output as a warm start for λ^0 in Algorithm 1.

Termination conditions. We terminate Algorithm 1 once the relaxed constraints are satisfied, or if the worst triangle quality (measured as the ratio of inradius to outradius) is below 10^{-3}. We also terminate if the total number of linear solves (i.e., the Newton direction solves for the hard constraint projection as well as the full constraint Newton direction solves) exceeds 100.

Fallback. If the initial projection to hard constraints in Algorithm 1 does not converge in 200 iterations, or if the resulting parameterization is too degenerate for sanitization and quantization, we greedily reduce the hard feature constraint set using the greedy heuristic described in Section 7.2, and we project to these reduced hard constraints from perfectly regular initial Penner coordinates $\lambda^0 = 0$. This approach generates a feature aligned parameterization with minimal geometric degeneracy.

Robust layout. The explicit refinement and metric layout method of Capouellez and Zorin [2023], which we use without modification for each component of the feature cut mesh, can fail for near degenerate metrics in standard double precision due, e.g., to numerical instability in the angle computations. If the method fails to produce a numerically seamless parameterization with accuracy 10^{-10}, we fallback to a higher precision implementation with a 100-bit mantissa, and we round the higher precision result to double precision. We emphasize that higher precision is only used for the layout sub-procedure, and the final output of the method is a numerically seamless parameterization in standard double precision.

D POST-PROCESSING

Given initial u, v coordinates, we optimize a weighted combination of a p-norm symmetric Dirichlet energy term E^p_{symDir} [Smith and Schaefer 2015], which acts a flip-preventing barrier term, and the orientation energy term E_{orient} of Bommes et al. [2009], which aligns the parameterization with the target cross field. That is, we optimize

$$E(u,v) = w_{symDir} E^p_{symDir}(u,v) + w_{orient} E_{orient}(u,v) \qquad (18)$$

Constraints. We enforce the linear seamless and feature alignment constraints through constraint elimination with a rank-revealing QR decomposition. We also fix an arbitrary vertex position per connected component of the parameterization, which ensures that our energy generically has a nonsingular Hessian. In cases where not all soft feature edges were aligned to machine precision, we use an augmented Lagrangian for the alignment constraints for the misaligned edges. Even when these alignment constraints cannot be satisfied, the augmented Lagrangian will still preserve soft alignment during the uv-optimization. We emphasize that edges that were fully aligned by our Penner coordinate method are still enforced by constraint elimination and maintained to machine precision.

Optimization. We use Newton's method to optimize the energy (or augmented Lagrangian) with respect to the reduced variables after constraint elimination. In some cases, the solver may fail while computing the Newton direction due to numerical instability, or the Newton direction may not be a descent direction, e.g. when using the augmented Lagrangian. In such cases, we add an exponentially increasing regularization term to the energy Hessian, which interpolates between the Newton direction and gradient direction, until we obtain a descent direction. We use a backtracking line search to ensure that the optimization energy decreases and that no triangles flip during the line step. We terminate when the gradient of the energy is sufficiently small.

Parameters. In our experiments, we use $w_{symDir} = 10^{-3}$ and $w_{orient} = 1$. We initially attempt to use a 6-norm for the symmetric Dirichlet energy, which reduces the global maximum distortion but is less numerically stable. If a suitably small gradient is not obtained in 500 iterations with the 6-norm, we fallback to the more stable 2-norm, again with a maximum of 500 iterations.

E STITCHING ALGORITHM DETAILS

We describe how two edges can be stitched together. Let e_{ij} be a feature edge split into two boundary edges in M^c, which we denote h_{ij} and h_{ji} respectively. Let $v_{ij}^0, ..., v_{ij}^n$ be the vertices on h_{ij} and $v_{ji}^0, ..., v_{ij}^m$ the vertices on h_{ji}. Generally, the edges of the refined triangles in the parametric domain may be curved. However, as described in Section 6, for a boundary edge the symmetry structure of the double mesh forces the refined edge to be straight. Thus, $v_{ij}^0, ..., v_{ij}^n$ are all colinear, and likewise for $v_{ji}^0, ..., v_{ij}^m$. Consequently, it is possible to express the interior vertices v_{ij}^k in terms of barycentric coordinates t_{ij}^k with respect to the endpoints v_{ij}^0 and v_{ij}^n. We may thus determine a common refinement by simply using edge midpoint refinement. More specifically, for each barycentric coordinate t_{ji}^k on edge h_{ji}, we determine the vertices v_{ij}^l and v_{ij}^{l+1} such that $t_{ij}^l < t_{ji}^k < t_{ij}^{l+1}$ and refine the triangle containing these two vertices such that the newly inserted vertex has barycentric coordinate t_{ji}^k. In the case where $t_{ij}^l = t_{ji}^k$, we do not refine the triangle. The stitching is demonstrated in Figure 23.

Note that this only defines the refinement of the parametric domain mesh. We also need to refine the embedded overlay mesh in a compatible manner. By the overlay construction and layout cut, the corresponding refinement vertices in the overlay mesh are also

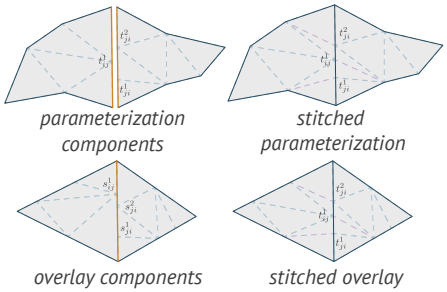


Fig. 23. Stitching of two overlay meshes across a boundary edge. The relative ordering of the inserted vertices for the overlay and the parametrization may be inconsistent, so we shift the vertices on the overlay edge to be consistent with the parametrization.

colinear and induce barycentric coordinates s_{ij}^k and s_{ji}^k. However, the barycentric coordinates these vertices induce will generally be different from those of the parametric domain. If $s_{ji}^k \notin (s_{ij}^l, s_{ij}^{l+1})$, then the midpoint refinement of the triangle is not well defined. In order to resolve this, we simply snap s_{ij}^k to t_{ij}^k so that the barycentric coordinates are consistent in the overlay and parametric domain. Since all inserted vertices are contained in the edges of the original mesh, refined triangles do not change orientation in the plane of the triangle if the refined vertices are moved in this way.

We may iteratively refine all edges by the above procedure, and then stitch together triangles in the overlay mesh in the obvious manner to produce a single closed mesh. The final result is a connected refined mesh with a (possibly disconnected) parametrization.