Rectangular Surface Parameterization

ETIENNE CORMAN, Université de Lorraine, CNRS, LORIA, France
KEENAN CRANE, Carnegie Mellon University, USA

This paper describes a method for computing surface parameterizations that map infinitesimal axis-aligned squares in the plane to infinitesimal rectangles on the surface. Such *rectangular* parameterizations are needed for a broad range of tasks, from physical simulation to geometric modeling to computational fabrication. Our main contribution is a novel strategy for constructing frame fields that are perfectly orthogonal and exactly integrable, in the limit of mesh refinement. In contrast to past strategies for achieving integrability, we obtain maps that are less distorted *and* better preserve target field directions. The method supports user-defined distortion measures, sharp feature alignment, prescribed or automatic cone singularities, and direct control over boundary behavior (*e.g.*, sizing or aspect ratio). By quantizing and contouring these maps we obtain high-quality anisotropic quad meshes, even without element-based optimization. Empirically, we outperform state-of-the-art research and commercial mesh generation algorithms in terms of element quality, accuracy, and asymptotic convergence rate in end-to-end simulation tasks, are competitive with the widely-used *ZBrush* package for automatic retopology, and provide *Chebyshev nets* of superior quality to methods specifically tailored to digital fabrication.

CCS Concepts: • **Mathematics of computing → Mesh generation**.

Additional Key Words and Phrases: surface parameterization, quad meshing

ACM Reference Format:
Etienne Corman and Keenan Crane. 2025. Rectangular Surface Parameterization. *ACM Trans. Graph.* 44, 4 (August 2025), 21 pages. https://doi.org/10.1145/3731176

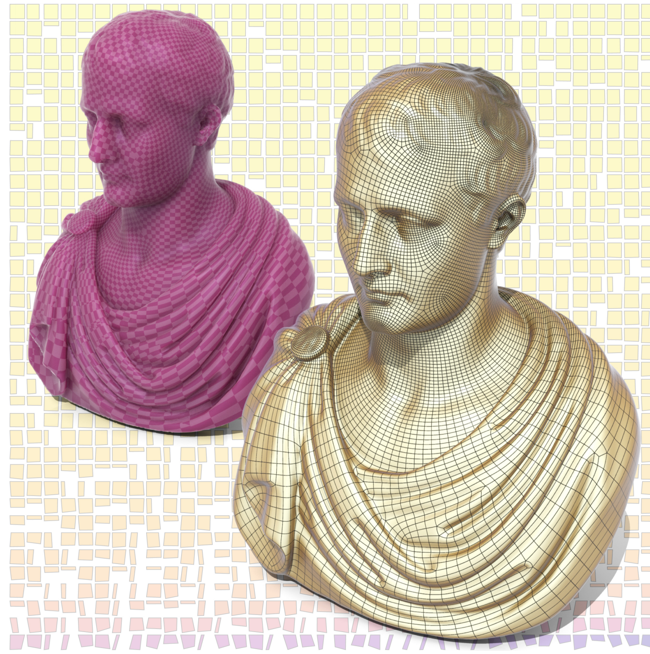


Fig. 1. *Left:* a *rectangular surface parameterization (RSP)* can adapt to long, skinny features, such as the folds of cloth in this bust. *Right:* such parameterizations lead to high-quality anisotropic quad meshes, even without any optimization or smoothing of mesh elements. *Background:* mesh quads sorted from most to least rectangular *(top left to bottom right)*. Non-rectangular elements are largely due to irregular vertices, where more (or fewer) than four quads must fit around a vertex.

1 Introduction

Definition. A surface parameterization $\phi(x,y) : U \to \mathbb{R}^n$ over a region $U \subset \mathbb{R}^2$ is *rectangular* if $\frac{\partial\phi}{\partial x} \cdot \frac{\partial\phi}{\partial y} = 0$, *i.e.*, if the coordinate axes get mapped to orthogonal directions on the surface. In differential geometry, such a map is known as an *orthogonal parameterization* [Do Carmo 2016, Section 2.5]; we opt for "rectangular" since "orthogonal" is often overloaded to mean *orthonormal* (orthogonal and unit length). Equivalently, ϕ is rectangular if its Jacobian can be expressed at each point as $J_\phi = QD$ where Q is orthogonal and D is diagonal, describing an axis-aligned scaling followed by a rotation. At each point, the induced Riemannian metric g is hence represented by a positive diagonal matrix $J_\phi^\mathsf{T} J_\phi = D^\mathsf{T} Q^\mathsf{T} Q D = D^2$, generalizing conformal maps, where it is a positive multiple of the identity. Importantly, not all parameterizations map (axis-aligned) squares to rectangles—not even infinitesimally (Figure 2).

Motivation. Numerous applications benefit from rectangular parameterization. In computational mechanics, it is a valuable starting point for quadrilateral mesh generation, since rectangular elements exhibit *asymptotically* faster convergence for quadrilateral FEM [Arnold et al. 2002] (see Figure 7). In computational architecture, special classes of orthogonal parameterizations correspond to *discrete nets* with essential properties for physical construction [Pottmann et al. 2007]. In digital fabrication, rectangular

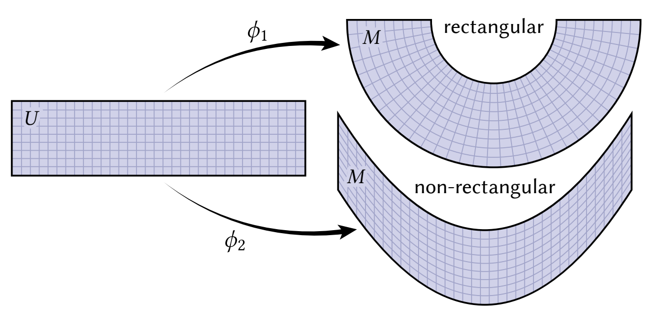


Fig. 2. A map ϕ is *rectangular* if it exhibits no shearing of the coordinate grid *(top)*, but can still apply unequal scaling along the coordinate axes.

Authors' Contact Information: Etienne Corman, etienne.corman@cnrs.fr, Université de Lorraine, CNRS, LORIA, Vandœuvre-lès-Nancy, France; Keenan Crane, kmcrane@cs.cmu.edu, Carnegie Mellon University, Pittsburgh, PA, USA.

ACM 1557-7368/2025/8-ART
https://doi.org/10.1145/3731176

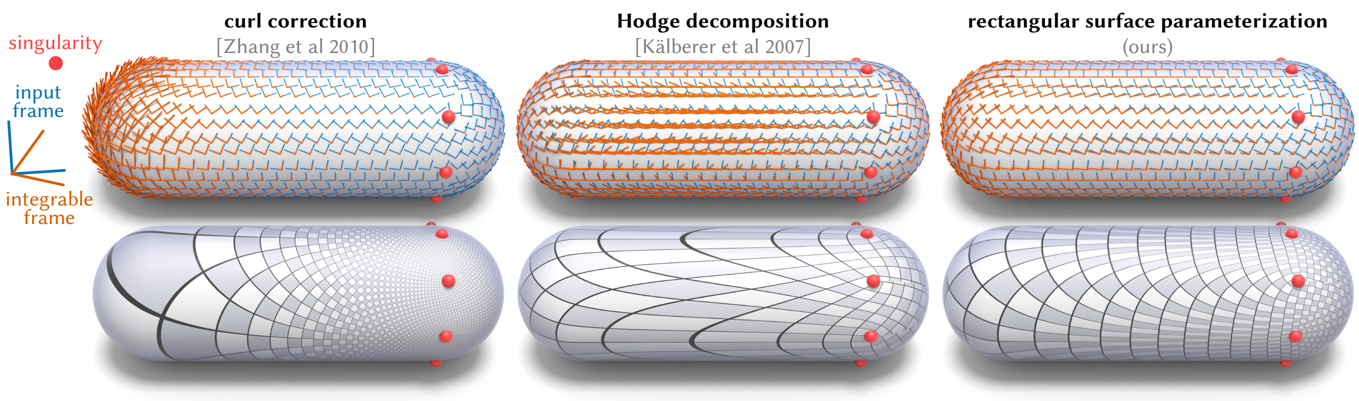


Fig. 3. Past methods for making a field integrable can distort the input field, badly violating target properties such as orthogonality *(bottom center)*, or incurring severe scale distortion *(bottom left)*. Our method explores the full space of integrable rectangular fields, yielding a perfect rectangular parameterization with low metric distortion. Here for instance, our method (rectangular surface parameterization) applies just a gentle rotation of the input, yielding less-distorted and better-aligned results than two standard techniques used in nearly all prior work on field-aligned parameterization (curl correction and Hodge decomposition).

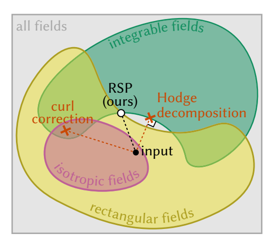
networks are used to design piecewise developable surfaces [Rabinovich et al. 2018], optimize principal stress networks [Pellis and Pottmann 2018], and model textiles via *Chebyshev nets* [Sageman-Furnas et al. 2019]. Finally, in computer graphics, rectangular parameterizations help mimic structural and pigmentation patterns found in nature [Knöppel et al. 2015] (Figure 4); lines of principal curvature form a rectangular network commonly used for illustration and visualization [Hertzmann and Zorin 2000]; and *automatic retopology* guided by principal directions is commonly used for subdivision modeling [Rouca 2019].

Integrability of Frame Fields. Our method for *rectangular surface parameterization (RSP)* is based on field-guided mapping [Bommes et al. 2013b, Section 3.5], where one first solves for a frame field that encodes the principal directions of the mapping, then recovers the mapping itself. At the heart of RSP is a procedure for making a given frame field *integrable*, *i.e.*, modifying it so that each frame axis locally corresponds to the gradient of a coordinate function. This seemingly small change has a big impact on the overall pipeline, since a nonintegrable field distorts the parameterization, which degrades mesh quality, and in turn diminishes performance of downstream applications (Section 6.5).

Though great care has been put into designing frame fields [Vaxman et al. 2016], there has been less intense study of how to preserve essential properties (such as orthogonality) when going from the field to a subsequent parameterization. Historically, two strategies are used almost universally to address nonintegrability:

- **Helmholtz-Hodge Decomposition** makes a field X integrable by extracting the curl-free or *exact* part, i.e., the component Y of X such that $Y = \nabla\phi$ for some scalar function ϕ [Kälberer et al. 2007, Section 4]. Note that Hodge decomposition is used implicitly in any method that seeks a "least squares" type solution to the equation $\nabla\phi = X$, since minimizing the residual $\|\nabla\phi - X\|^2$ yields a normal equation $\Delta\phi = \nabla \cdot X$ equivalent to the Poisson equation used in Hodge decomposition. For instance, all methods based on *mixed integer quadrangulation* [Bommes et al. 2009, Section 5] effectively adopt the Hodge approach to integrability—even if not cast as a direct modification of the field itself.
- **Curl Correction** seeks a pointwise rescaling s of X such that $\mathrm{curl}(sX) = 0$, *i.e.*, such that the rescaled field Y is curl-free [Ray et al. 2006; Zhang et al. 2010]. Since scaling is isotropic, Y describes derivatives of a conformal mapping. Curl correction has been generalized to other settings, such as *polyvector fields* [Diamanti et al. 2015], and is representative of any method that scales field magnitude while keeping directions fixed [Zhang et al. 2010; Simons and Amenta 2024].

An attractive feature of these strategies is that they amount to sparse linear solves. A major drawback, however, is that in both methods the integrable field Y is "anchored" to any flaws in the input field X: Hodge decomposition seeks the closest integrable field Y (in the L^2 sense) to the input X; curl correction keeps the directions of X fixed, and is only able to modify magnitude. Hence, even if X exhibits erroneous directions or a poorly-designed global flow, the final field will retain some "memory" of this bad initial guess (Figure 3).



Integrable Rectangular Fields. We take a fundamentally different approach: we optimize any user-specified quality criterion (smoothness, alignment, *etc.*) over the full space of exactly integrable, rectangular frame fields, *i.e.*, fields where axes are orthogonal and positively-oriented, but need not have the same magnitude. The price we pay is that we must now solve a nonlinear problem rather than a linear one—yet the method remains efficient, and reliably yields optimal results (Figure 21). The benefit is that we often get fields that are better than those produced by linear methods in all respects: they are not only integrable, but are perfectly rectangular *by construction*, and generally lead to lower-distortion mappings than past methods (Figure 3, *right*).

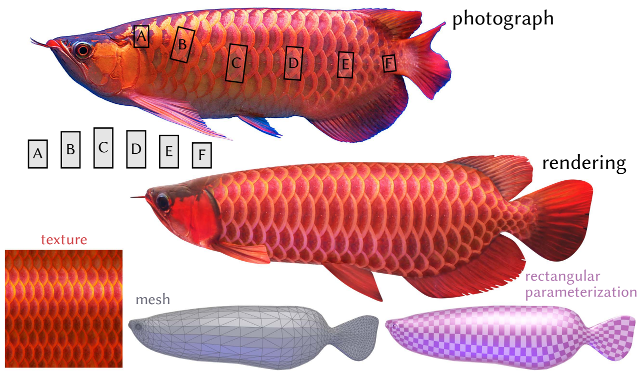


Fig. 4. A variety of natural phenomena exhibit rectangular patterns, where a continuously varying aspect ratio is needed to adapt to changes in curvature and pattern orientation. Our rectangular parameterizations can be used to synthesize patterns with a natural orthogonal structure, as seen here in the scales of *scleropages formosus* (Asian arowana).

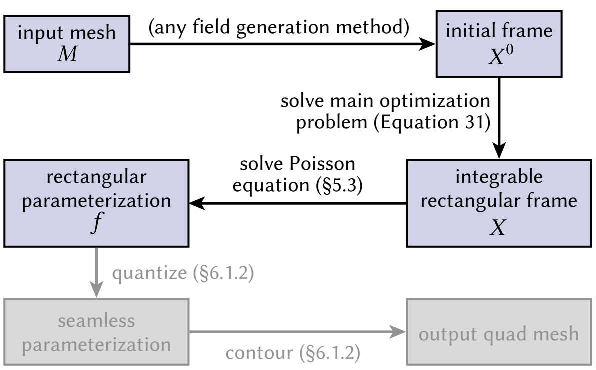


Fig. 5. The basic steps of our algorithm. The main cost is solving the optimization problem for the integrable rectangular frame. Optional steps needed for mesh extraction are shown in gray.

Method Overview. How do we actually compute a rectangular parameterization? One idea is to simply assert that its derivatives must be orthogonal and positively oriented, but directly penalizing the map itself can result in severe violation of injectivity (Figure 10). Instead, we characterize a rectangular parameterization indirectly in terms of the frame field induced by the parameterization (Section 3). In particular, we ask, *"if we already had a rectangular parameterization, what must be true about how the corresponding coordinate frame moves around on the surface?"* Answering this question leads to our main integrability condition (Equation 10) on data encoding the frame at each point, namely, a rotation and positive scaling of an arbitrary reference frame. In turn, we obtain a positively oriented frame—and hence a locally injective map—*by construction.*

Algorithmically, we proceed in the opposite direction (Figure 5). First, we compute an orthonormal reference frame X^0 via any method. Second, we find rotation angles θ and positive scale factors a, b that describe an integrable rectangular frame X relative to X^0. Third, we solve a Poisson equation to recover the final map f from X. Since X is integrable, any deviation from orthogonality is due only to a small amount of discretization error, which vanishes under refinement (Figure 24). The final per-vertex map f can then be used in any downstream application, such as quad meshing.

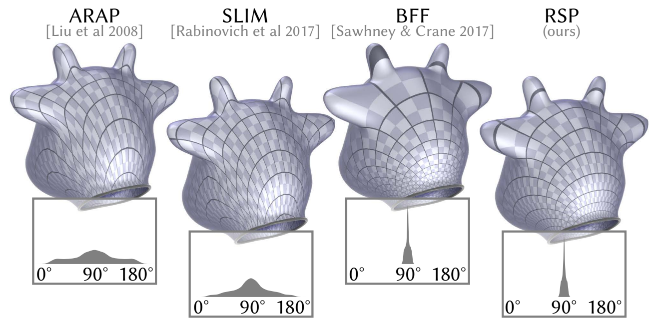


Fig. 6. General-purpose parameterization methods like ARAP and SLIM minimize overall metric distortion, but exhibit severe shearing. Conversely, conformal methods like BFF give an orthogonal parameterization, but with extreme area distortion. A rectangular parameterization offers the best of both worlds: no shearing, yet far less area distortion than conformal maps.

As with the metric scaling approach to conformal maps [Ben-Chen et al. 2008; Springborn et al. 2008], expressing our parameterization in terms of *log scale factors* (which necessarily induce strictly positive scaling) ensures it is flip-free, modulo discretization error. Overall, starting with a principled discretization of an integrable continuous formulation greatly improves results relative to past methods (Section 6).

Outline. After reviewing related work (Section 2), we derive our main integrability condition in the smooth setting (Section 3). We then discretize this condition on triangle meshes (Section 4), and describe how we optimize a frame for integrability, plus additional quality objectives (Section 5). Section 6 numerically validates our approach, via both synthetic metrics and end-to-end application tasks. Section 7 covers limitations and future work.

2 Related Work

Here we detail connections with methods from mesh parameterization (Section 2.1), field-guided mapping (Section 2.2), and quad meshing (Section 2.3). Overall, we find no past method able to optimize a given distortion metric directly over the space of rectangular maps. At one end of the spectrum, methods encourage orthogonality via soft penalties which still permit substantial shear. At the other end, *conformal maps* are exactly orthogonal, but exhibit major scale distortion. Moreover, no past ansiotropic quad meshing scheme (to our knowledge) guarantees convergence to perfect rectangular elements under input mesh refinement.

2.1 Surface Parameterization

	shear	*stretch*	*scale*
conformal	✗	✗	✔
authalic	✔	✔	✗
rectangular	✗	✔	✔

Relative to traditional mesh parameterization, we break from the goal of finding a map that is "as isometric as possible," instead seeking the least-distorted rectangular map. In other words, unless requested by the user, we penalize neither uniform scale nor axis-aligned

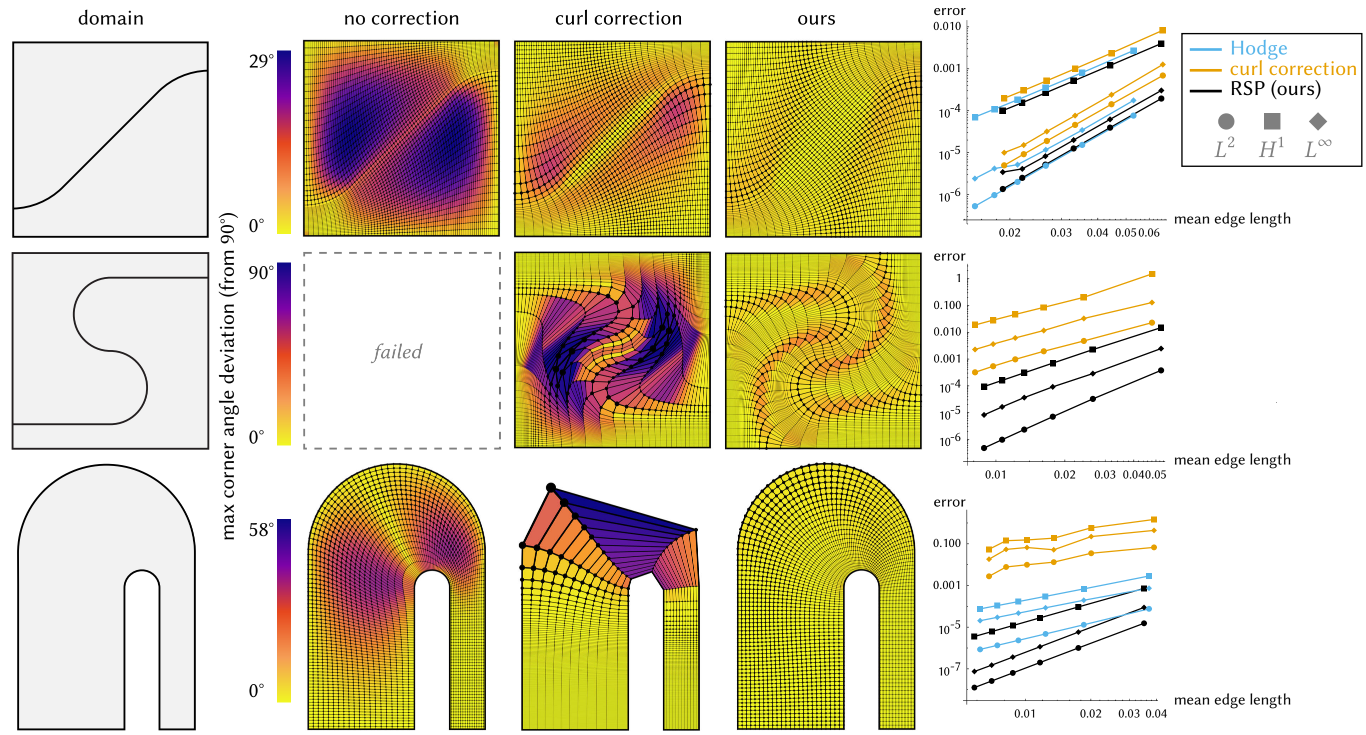


Fig. 7. As shown by [Arnold et al. 2002], quadrilateral FEM converges asymptotically faster on rectangular meshes than on general quad meshes. Empirically, meshes produced by RSP achieve the theoretically-predicted optimal convergence rate on common problems, such as solving a Poisson equation *(far right)*. In contrast, past curl correction schemes lead to highly non-rectangular elements, yielding suboptimal convergence.

stretch—only shear is explicitly forbidden. In this way, rectangular parameterization complements area-preserving or *authalic* maps and angle-preserving or *conformal* maps (see inset). In fact, our approach provides a very natural generalization of the *log scale factors* used to describe conformal maps. In particular, we augment the conformal scale factor u with an additional authalic scale factor v (Section 3.3.1).

The full landscape of parameterization methods is too vast to detail here [Floater and Hormann 2005; Sheffer et al. 2007], but we can get a sense of tradeoffs by comparing to representative methods that penalize non-isometry, such as *ARAP* [Liu et al. 2008] or *SLIM* [Rabinovich et al. 2017] , and to conformal methods like *BFF* [Sawhney and Crane 2017]. Here we use the implementations found in `libigl` example 503 [Jacobson et al. 2013]), *Blender* [2025], and by Sawhney and Crane [2019], *resp.* As seen in Figure 6, RSP achieves the best of both worlds: like a conformal map, it preserves right angles between principal axes, yet like non-conformal methods it does not exhibit extreme area distortion.

Distortion Measures. More broadly, many distortion metrics have been used for surface parameterization (Rabinovich et al. [2017] provide one list), but most are *isotropic* since they are expressed via scalar invariants of the Jacobian. In contrast, rectangular mapping demands an *anisotropic* distortion measure, where both scalar stretch and vectorial orientation are accounted for. In the field-guided setting, Bommes et al. [2009, Section 5.1] use a penalty that promotes a prescribed anisotropy along a fixed direction field. Unlike RSP, this energy still permits large scale non-injectivity (Figure 10), and does not jointly optimize for the directions and anisotropy that best minimize distortion. Similarly, Levi [2023b] describe a *shear reduction* penalty that does not yield exact orthogonality (even under refinement), and requires that an injective map is already given as input. In contrast, RSP is orthogonal by construction, and jointly optimizes both stretch and orientation to minimize distortion.

Local Injectivity. A variety of recent methods offer hard guarantees on local injectivity [Rabinovich et al. 2017; Gillespie et al. 2021b; Fu et al. 2021]. In the smooth setting, the mapping defined by RSP is guaranteed to be locally injective, since positive scale factors $e^{u+v}, e^{u-v} > 0$ ensure the integrable frame (hence derivatives of the map) cannot invert. In the discrete setting, however, edge vectors mapped into the plane do fit together exactly (Section 4.2.1). Hence, our final Poisson step, which integrates edges in a least-squared sense, can in principle produce flipped triangles—though this rarely occurs in practice. Any such failure to close also vanishes under refinement (Figure 24), and is often mitigated by automatic cone placement, which can help account for a reference frame X^0 with poorly chosen global topology (Figure 20). Morever, nothing prevents the *almost* perfectly integrable rectangular field generated by RSP from being used as input to a locally-injective field-guided parameterization method (*e.g.*, [Myles et al. 2014]), rather than our simple Poisson integration scheme.

2.1.1 Conformal Mapping. Mathematically, RSP is a natural generalization of the *metric scaling* approach to conformal flattening [Ben-Chen et al. 2008; Springborn et al. 2008]. Here, a Riemannian metric g is transformed into a flat metric g_0 by a positive scaling $g_0 = e^{2u}g$, where $u : M \rightarrow \mathbb{R}$ is the *log conformal factor*. This transformation preserves angles between tangent vectors, but uniformly scales area. The factor u is obtained by solving the linear *Yamabe equation* [Aubin 2013; Bunin 2008; Sawhney and Crane 2017] or by minimizing a convex energy [Luo 2004; Springborn et al. 2008]. While this approach is efficient, it can induce extreme area distortion, and does not always admit a solution when seeking feature alignment [Myles and Zorin 2013]. RSP overcomes these limitations via an additional log factor $v : M \rightarrow \mathbb{R}$ that *stretches* the domain while preserving area. Our integrability condition (Equation 10) can be viewed as a generalization of [Myles and Zorin 2013, Proposition 3], which gives the relationship between the log conformal factor u and the rotation rate of the frame induced by a conformal mapping.

2.2 Field-Guided Parameterization

Past work on field-guided parameterization has focused heavily on *topological* questions, *e.g.*, the conditions under which a field can be parameterized [Shen et al. 2022], or guarantees of local injectivity [Campen et al. 2019; Levi 2023a]. These properties are necessary to ensure a quad mesh can be generated—but do not by themselves guarantee that mesh elements will have good quality.

Geometric criteria are often secondary, and addressed indirectly through weak requirements like bounded distortion [Chien et al. 2016], or soft penalties on metric distortion [Capouellez and Zorin 2024]. Likewise, while orthogonality of the input field promotes orthogonality of the final map, this property is not achieved exactly unless the field is already integrable (Figure 3, *center*).

2.2.1 Integrable Fields. Past methods for field integrability also treat geometric quality as an "afterthought"—largely prioritizing a topological property (integrability). For example, curl correction achieves integrability through significant isotropic [Ray et al. 2006] or anisotropic scaling [Zhang et al. 2010], but makes no attempt to optimize the direction of anisotropy in order to reduce geometric distortion. Diamanti et al. [2015] and Sageman-Furnas et al. [2019] develop integrability schemes for *polyvector fields*, Pluta et al. [2021] establishes an integrability condition for planar hexagonal meshing, and Coiffier and Corman [2023] use (as we do) moving frames to express integrability constraints for general fields—yet in all these schemes, only soft penalties are used to encourage geometric quality. Our approach to integrability guarantees that, under refinement, the parameterization always approaches a geometrically orthogonal one (Figure 24).

Less attention has been given to integrable *orthogonal* fields. Simons and Amenta [2024] give conditions under which a frame with *fixed* orientation admits a rectangular parameterization after axis-aligned scaling. However, fixing frame directions can lead to severe scale distortion, as seen in Figure 7, *left*. Jezdimirović et al. [2023] study fields with fixed singularities and admissible rotations but fail to provide a practical algorithm both constraining orthogonality and minimizing distortion. Finally, Couplet et al. [2024] propose an integrability condition for orthogonal frame fields represented with orthogonally decomposable *(odeco)* tensors [Palmer et al. 2020]. Here, integrability is imposed only weakly, and the method struggles to maintain orthogonality.

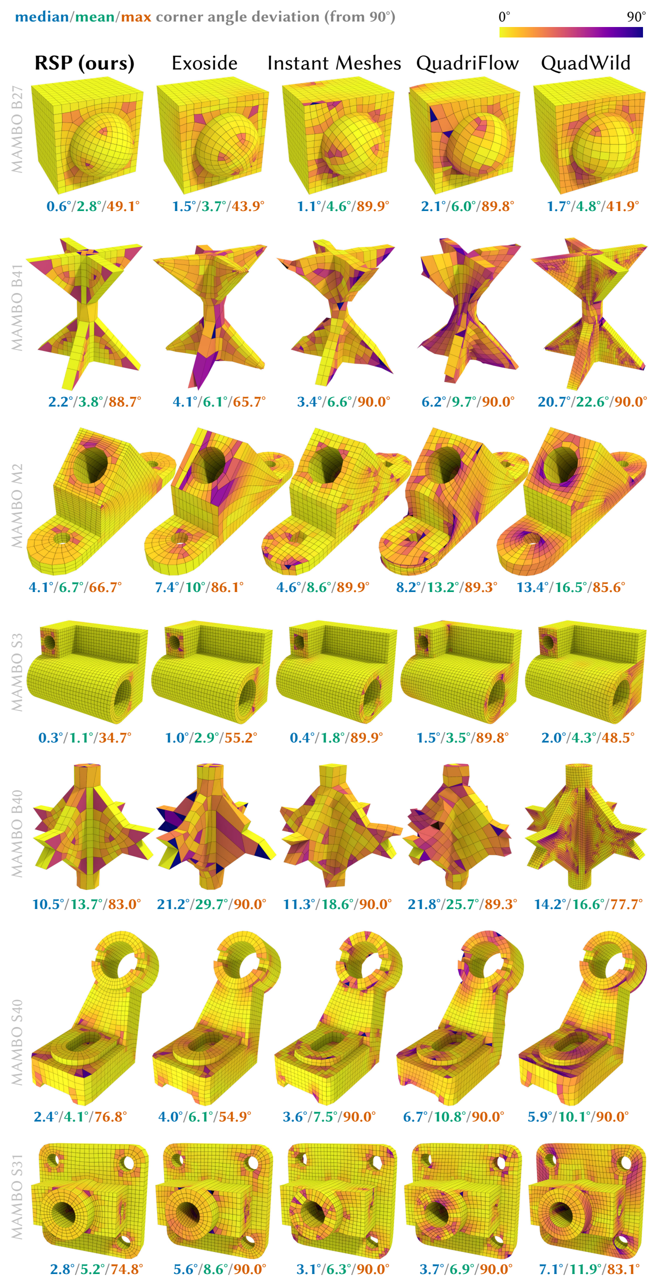


Fig. 8. For field-guided mesh generation, starting with an integrable rectangular field both reduces distortion during the parameterization step, and helps elements adapt to geometric features. Each quad is colored according to the maximum deviation from 90° at any corner, and we give the median/mean/max such deviation across the entire mesh.

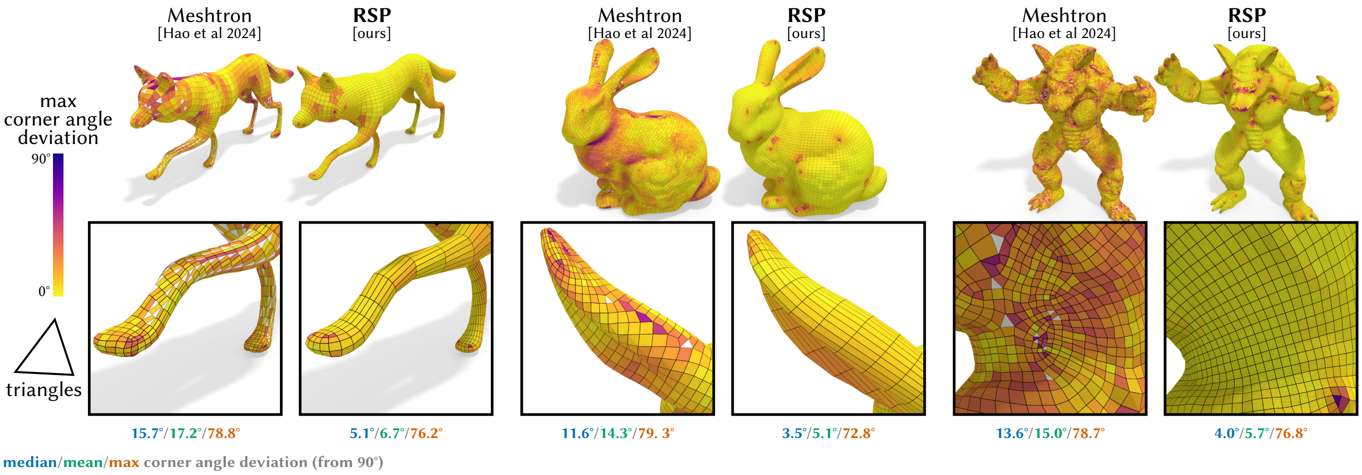


Fig. 9. Compared to recent learning-based methods, our method also produces meshes that more closely resemble models created by professional artists, with cleaner edge flow, fewer irregular regions, and better-shaped quads. Here for instance we compare against the *Meshtron* method of Hao et al [2024], which is directly trained on artist-generated models.

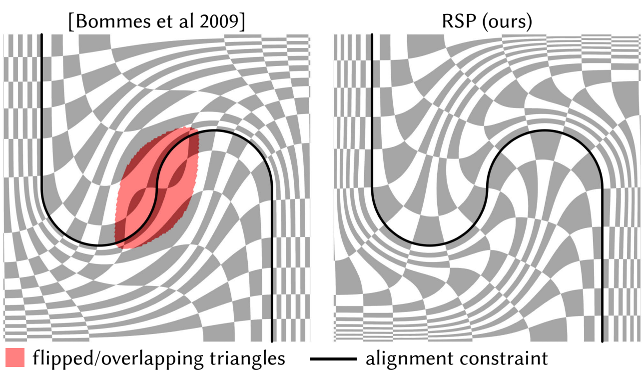


Fig. 10. Attempting to achieve orthogonality through penalty terms is unsatisfactory, since penalties may "fight" with other energy terms. *Left:* fixing field directions and promoting anisotropy during parameterization leads to large noninjective regions, and many angles far from 90°. *Right:* our method directly explores the space of rectangular maps, yielding injective, orthogonal results even for difficult constraints.

2.3 Anisotropic Quad Meshing

Beyond field-guided parameterization, there are a variety of strategies for generating anisotropic rectangular meshes.

In 2D, *boundary layer meshes* concentrate anisotropic elements near the domain boundary to resolve characteristics of, *e.g.*, flow equations, without globally inflating mesh resolution. Here, *paving* or *advancing front* methods incrementally add high-quality elements starting at the boundary, but must introduce highly irregular connectivity to resolve collisions between propagating wavefronts [Merhof et al. 2007]. Likewise, *indirect* methods first triangulate the domain, then merge triangles into quads [Remacle et al. 2013]—again resulting in many singularities and suboptimal, non-rectangular elements (Figure 28, *bottom, center-left*). Explicit control over boundary conditions enables RSP to produce such meshes (Figure 18, *bottom*), while maintaining a regular, highly rectangular grid structure almost everywhere.

In 3D, early methods obtained anisotropic quads by dicing up surfaces along lines of principal curvature [Alliez et al. 2003; Marinov and Kobbelt 2004]. This local approach makes it difficult to obtain good global structure, resulting in many low-quality, nonquadrilateral elements. Field-guided methods addressed many of these issues, with limitations discussed in Section 2.2. Other methods modify the *metric* of the surface to encourage rectangular elements [Kovacs et al. 2010] or user-controlled anisotropy [Jiang et al. 2015]. Yet metric-based approaches still provide no guarantee that the field will be integrable with respect to the modified metric—which means that any subsequent parameterization can violate criteria like orthogonality.

Learning-Based Methods. Finally, some recent methods learn to generate mesh geometry and connectivity based on a given training set [Siddiqui et al. 2024; Son et al. 2024; Shen et al. 2024; Hao et al. 2024]. Here one could train on high-quality rectangular meshes made by artists or engineers. However, current methods leave much to be desired in terms of quality (Figure 9), ability to generalize to unseen object categories, and generation cost (*e.g.*, on the order of minutes–hours for *Meshtron* [Hao et al. 2024], versus seconds–minutes for our method). In this context, RSP provides a way to generate high-quality training data for such networks; our integrability condition (Equation 10) may also prove useful as a regularizer for methods that generate fields [Dielen et al. 2021].

3 Smooth Formulation

We develop our method in the setting of continuous differential geometry. An introduction to relevant concepts (including moving frames) can be found in [O'Neill 2006]; further background on moving frames can be found in [Cartan et al. 2001]. Readers uninterested in derivation of the method may wish to skip to Section 5, where we describe the final algorithm.

In the smooth setting, our algorithm consists of a few basic steps:

I. Pick an orthonormal reference frame field (X_1^0, X_2^0).
II. Solve for rotations θ and per-axis scale factors a, b at each point that transform the reference field into an integrable rectangular field $(aX_1^\theta, bX_2^\theta)$.
III. Use the rectangular field to construct the derivatives μ of a rectangular parameterization.
IV. Integrate the derivative to obtain the parameterization f itself, by solving a Poisson equation $\Delta f = \nabla \cdot \mu$.

This section is devoted to deriving the equation needed for Step II.

3.1 Basic Setup

Consider an oriented smooth surface M with a Riemannian metric g (Figure 11). Let $J_p : T_pM \to T_pM$ denote a counter-clockwise 90° rotation in each tangent space T_pM compatible with the metric, *i.e.*, $g_p(X|_p, JX|_p) = 0$ for all vector fields X, at all points $p \in M$. For brevity we drop subscripts p throughout, implying that relationships hold at all points $p \in M$. We use Δ for the Laplace-Beltrami operator on (M, g), $\nabla \cdot X$ for the divergence operator, and $\nabla_Y X$ for the covariant derivative of X along the direction Y.

3.2 Moving Frames on Surfaces

A *moving frame* is any pair of vector fields X_1, X_2 on M that are orthonormal with respect to g and consistently oriented, *i.e.*, $X_2 = JX_1$ and $g(X_i, X_j) = \delta_{ij}$, where δ_{ij} denotes the Kronecker delta. A classic example is the *Darboux frame*, where X_1, X_2 are the principal curvature directions—though in our case we do not make any special

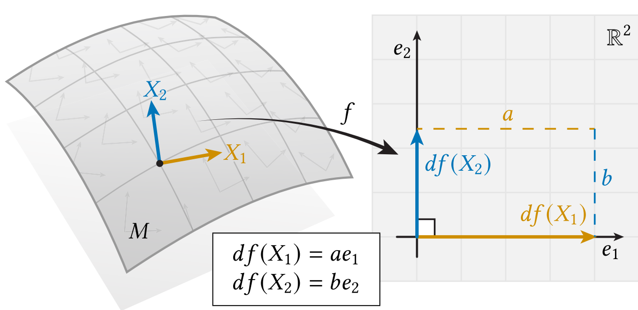
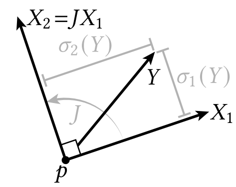


Fig. 11. Given a surface M, we seek a parameterization $f : M \to \mathbb{R}^2$ that takes an orthonormal frame (X_1, X_2) at each point to a rectangular orthogonal frame (ae_1, be_2) in the plane.

quantity	description
$M \subset \mathbb{R}^3$	problem domain (surface)
$f : M \to \mathbb{R}^2$	surface parameterization
$df : TM \to \mathbb{R}^2$	differential / Jacobian of parameterization
$a, b : M \to \mathbb{R}_{>0}$	width, height of infinitesimal rectangles
$u, v : M \to \mathbb{R}$	logarithm of a, b
$e_1, e_2 \in \mathbb{R}^2$	standard basis in the 2D parameter plane
$X_1, X_2 : M \to TM$	axes of infinitesimal rectangles (moving frame)
$\omega_{12} : TM \to \mathbb{R}$	$\omega_{12}(Y)$ is rotation rate of (X_1, X_2) in Y direction
$\sigma_1, \sigma_2 : TM \to \mathbb{R}$	coframe $\sigma_i(X_j) = \delta_{ij}$

Fig. 12. Basic quantities used in our formulation.

choice of frame. We assume only that the frame varies smoothly away from a collection of isolated points, which will correspond to *cone singularities* in the final parameterization. Singularities are further discussed in Section 3.6; for now we consider a frame without singularities.



Coframe. Every moving frame has a corresponding *coframe*, *i.e.*, a pair of differential 1-forms $\sigma_1, \sigma_2 : TM \to \mathbb{R}$ such that

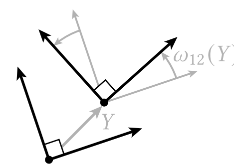
$$\sigma_i(X_j) = \delta_{ij}, \quad i, j \in \{1, 2\},$$

Intuitively, at each point $p \in M$ the coframe measures the extent of any given vector Y along the frame directions, *i.e.*, it gives the local coordinates of Y in the basis X_1, X_2.

Connection Form. We also have differential 1-forms $\omega_{ij} : TM \to \mathbb{R}$ for $i, j \in \{1, 2\}$ defined by the relationship

$$\omega_{ij}(Y) := \sigma_i(\nabla_Y X_j),$$

for all tangent vector fields Y. Intuitively, $\omega_{ij}(Y)$ describes the change in the ith component of the jth frame direction as we move along Y, and hence expresses change in the frame relative to the moving basis of the frame itself. Since the frame is orthonormal, we always have $\omega_{11} = \omega_{22} = 0$, *i.e.*, a unit vector cannot get longer or shorter. Moreover, $\omega_{12} = -\omega_{21}$: from point to point, the frame can only rotate, and $\omega_{12}(Y)$ captures the rate of rotation as we move in the Y direction. We refer to ω_{12} as the *connection form* for the moving frame X_1, X_2, since it effectively tells us how local coordinate systems at nearby points "connect."



Structure Equations. One can show that any σ and ω that come from a frame X must satisfy *Cartan's first structure equation*

$$\begin{aligned} d\sigma_1 &= -\omega_{12} \wedge \sigma_2, \\ d\sigma_2 &= -\omega_{21} \wedge \sigma_1, \end{aligned} \tag{1}$$

where d and $\wedge$ are the exterior derivative and wedge product, *resp.* In fact, for a given σ, ω is the unique 1-form satisfying this equation [O'Neill 2014, Corollary 1.8.5]. Note that for our algorithm we will not need Cartan's *second* structure equation, which in 2D simply asserts that ω is closed ($d\omega = 0$).

3.3 Rectangular Parameterization

We now characterize rectangular parameterizations in terms of moving frames. Suppose in particular we want a parameterization $f : M \to \mathbb{R}^2$ that locally looks like a pure nonuniform scaling along some orthonormal frame X, *i.e.*, no shearing. In particular, let $e_1 = (1, 0), e_2 = (0, 1)$ be the standard basis for $\mathbb{R}^2$. Then we want our parameterization and moving frame to satisfy

$$\begin{aligned} df(X_1) &= ae_1, \\ df(X_2) &= be_2, \end{aligned} \tag{2}$$

where the differential of the parameterization $df : TM \to \mathbb{R}^2$ maps tangent vectors X on the surface to the corresponding vectors in the plane $\mathbb{R}^2$, and $a, b : M \to \mathbb{R}_{>0}$ are strictly positive functions that describe the scaling along each frame direction (Figure 11).

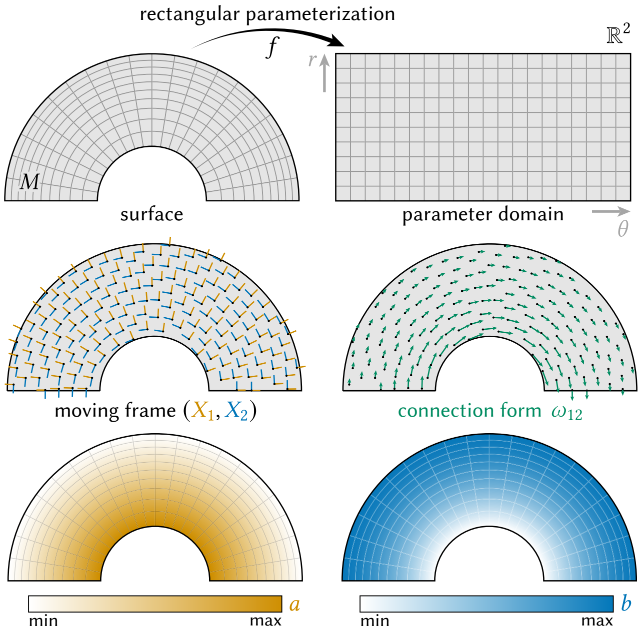


Fig. 13. Any rectangular parameterization $f : M \to \mathbb{R}^2$ can be associated with an orthonormal frame (X_1, X_2) and a pair of scale factors a, b along the two axes. The value $\omega_{12}(Y)$ encodes the speed of rotation in any given direction Y, which can be though of as the inner product of Y with the representative vector for the connection form ω_{12}. Here for instance the frame rotates in the θ direction, but remains parallel in the r direction.

3.3.1 Log Scale Factors. Alternatively, we can express our parameterization as

$$\begin{array}{rcl} df(X_1) & = & e^{u+v}e_1, \\ df(X_2) & = & e^{u-v}e_2, \end{array} \tag{3}$$

for arbitrary (*i.e.*, not necessarily positive) functions $u, v : M \to \mathbb{R}$, which we call the *(log) conformal factor* and *(log) authalic factor*, *resp.* In particular, when $v = 0$ we get $a = b = e^u$, *i.e.*, the width and height of infinitesimal rectangles are scaled by the same amount, preserving angles. When $u = 0$ we get $a = e^v = 1/b$, i.e., the width and height are scaled by reciprocal amounts, preserving local area.

Log factors are natural for several reasons. First, exponentiation guarantees that scale factors a, b are always nonnegative, avoiding overlap (and helping to prevent inverted mesh elements). Second, they yield an integrability condition linear in u, v, as discussed below. Finally, they make it straightforward to express objective functions and boundary conditions in simple geometric terms (*e.g.*, angle or area preservation); see Sections 5.1.1 and 5.1.3. This setup directly generalizes the standard formulation of conformal maps in terms of the log conformal factor (Section 3.4.2).

3.4 Integrability Condition

We next derive an expression for the connection form ω_{12} associated with a rectangular parameterization f, *i.e.*, for the rotation speed of a rectangular frame, in terms of the log scale factors u, v. This relationship serves as an intermediate form of our final integrability condition, which will ultimately be expressed in terms of a rotation and scaling of a fixed reference frame (Section 3.4.1).

First note that the differential df can be expressed via the coframe:

$$df = a\sigma_1 e_1 + b\sigma_2 e_2. \tag{4}$$

Applying the exterior derivative to both sides of this equation (and noting that e_1, e_2 are constant) then yields

$$0 = ddf = (da \wedge \sigma_1 + ad\sigma_1)e_1 + (db \wedge \sigma_2 + bd\sigma_2)e_2. \tag{5}$$

Since e_1, e_2 are linearly independent, we have

$$\begin{array}{rcl} da \wedge \sigma_1 + ad\sigma_1 & = & 0, \\ db \wedge \sigma_2 + bd\sigma_2 & = & 0. \end{array}$$

Applying the first structure equation (Equation 1) and making the substitutions $da = ad(u+v)$, $db = bd(u-v)$ yields

$$\begin{array}{rcl} d(u+v) \wedge \sigma_1 & = & \omega_{12} \wedge \sigma_2, \\ d(u-v) \wedge \sigma_2 & = & \omega_{21} \wedge \sigma_1. \end{array} \tag{6}$$

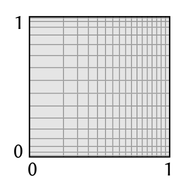
Applying the 2-forms on both sides of these equations to the vectors X_1, X_2 (via the usual identity $\alpha \wedge \beta(X,Y) = \alpha(X)\beta(Y) - \alpha(Y)\beta(X)$), and recalling that $\sigma_i(X_j) = \delta_{ij}$, we get

$$\begin{array}{rcl} d(u+v)(X_2) & = & -\omega_{12}(X_1), \\ d(u-v)(X_1) & = & -\omega_{21}(X_2). \end{array}$$

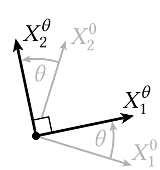
Noting that ω_{12} can be expanded in the coframe as $\omega_{12} = \omega_{12}(X_1)\sigma_1 + \omega_{12}(X_2)\sigma_2$ (and recalling that $\omega_{12} = -\omega_{21}$), we then get an equation for the rotation speed of the rectangular frame X_1, X_2 in terms of the log scale factors u and v:

$$\omega_{12} = -D_{X_2}(u+v)\sigma_1 + D_{X_1}(u-v)\sigma_2. \tag{7}$$

Here $D_X\phi$ denotes the directional derivative of a scalar function ϕ along the vector field X.



Geometric Interpretation. Geometrically, this equation tells us that in order for scale factors to vary over space, an infinitesimal rectangle must exhibit some amount of "bending" (see Figure 13 for an illustration). *E.g.*, when $M \subset \mathbb{R}^2$, $\omega_{12} = 0$ only when the horizontal scaling a is purely a function of x and the vertical scaling b is purely a function of y. In contrast, the inset example $f(x,y) := (\sqrt{x}, \frac{1}{2} - \frac{1}{2}\cos(\pi y))$ on the domain $M = [0,1]^2$ exhibits independent scaling along each axis, and the frame does not rotate.



3.4.1 Final Integrability Condition. We can express the frame X in Equation 7 as a rotation of some fixed reference frame X^0 by a continuously-varying angle $\theta : M \to \mathbb{R}$:

$$\begin{array}{rcl} X_1^\theta & := & \cos(\theta)X_1^0 + \sin(\theta)X_2^0, \\ X_2^\theta & := & -\sin(\theta)X_1^0 + \cos(\theta)X_2^0. \end{array} \tag{8}$$

Since θ is continuous, X^θ will have the same singularities as X^0 (see Section 3.6.2 for further discussion). Likewise, if σ^0 is the coframe of X^0, then

$$\begin{array}{rcl} \sigma_1^\theta & := & \cos(\theta)\sigma_1^0 + \sin(\theta)\sigma_2^0, \\ \sigma_2^\theta & := & -\sin(\theta)\sigma_1^0 + \cos(\theta)\sigma_2^0. \end{array} \tag{9}$$

We then have the following relationship between the rotation speed of the original and rotated frames:

Lemma 3.1. *The connection form* ω_{12}^θ *of the rotated frame differs from the connection form* ω_{12}^0 *of the reference frame by the differential* $d\theta$ *of the angle of rotation,* i.e.,

$$\omega_{12}^\theta = \omega_{12}^0 - d\theta.$$

For example, for a constant reference frame on a flat region $M \subset \mathbb{R}^2$, we get just $\omega_{12}^\theta(Y) = d\theta(Y)$, i.e., the rate of rotation is just the change in angle as we move along any given direction Y. See Appendix A for a proof.

Substituting this expression for ω_{12} into Equation 7 yields our final integrability condition, which says how the rotation angles θ and log scale factors u, v must be related in a rectangular map:

$$\boxed{\omega_{12}^0 - d\theta = -D_{X_2^\theta}(u+v)\sigma_1^\theta + D_{X_1^\theta}(u-v)\sigma_2^\theta.} \tag{10}$$

Note that since the frame X^θ and coframe σ^θ depend on θ, this equation is linear in u, v but nonlinear in θ.

3.4.2 Relationship to Conformal Maps. A conformal map exhibits scaling but not shearing—in terms of scale factors, conformal maps correspond to the case $a = b$, or equivalently $v = 0$ (Equation 3). If we also assume a constant reference frame X^0 (so that $\omega_{12}^0 = 0$), Equation 10 reduces to just

$$d\theta = (D_{X_2}u)\sigma_1 - (D_{X_1}u)\sigma_2,$$

or equivalently,

$$d\theta = *du, \tag{11}$$

where $*$ denotes the Hodge star (since for any 1-form α on a two-dimensional domain, $*\alpha(X) = -\alpha(JX)$ for all X). As discussed in Myles and Zorin [2013, Proposition 3], this equation gives the relationship between the scale factor u and the connection 1-form $\omega_{12} = - * d\theta$ in a conformal parameterization.

3.5 Poisson Equation

Solving Equation 10 for θ, u, v provides data sufficient to determine the differential df of a rectangular parameterization, which we can then integrate to obtain the parameterization f itself. In particular, we know from Equation 4 that we can express the differential as

$$\mu \coloneqq a\sigma_1^\theta e_1 + b\sigma_2^\theta e_2, \tag{12}$$

i.e., as a 1-form that sends X_1^θ to ae_1 and X_2^θ to be_2. In the smooth setting the equation $df = \mu$ is exactly solvable, and we could in principle integrate μ directly to obtain f. In the discrete setting, however, there will be a small amount of nonintegrability arising from discretization error (*i.e.*, the fact that we use a mesh with finitely many elements). Instead, as is standard practice in geometry processing, we seek the parameterization f that minimizes the residual $\|df - \mu\|_{L^2}^2$, leading to a vector-valued Poisson equation

$$\Delta f = \nabla \cdot \mu. \tag{13}$$

Solving this equation yields our final parameterization $f : M \longrightarrow \mathbb{R}^2$.

3.5.1 Local Injectivity. Since (i) $a = e^{u+v}$ and $b = e^{u-v}$ are strictly positive, (ii) (X_1^θ, X_2^θ) is a rotation of a positively-oriented frame, and (iii) μ is exactly integrable, the map f defined by our procedure has a nondegenerate differential (*i.e.*, Jacobian) *by construction.* Hence, in the smooth setting, it is always locally injective. In the discrete setting we no longer have a hard guarantee, due to discretization error. However, since this error goes to zero under refinement (Figure 24), we can reduce any overlap by simply refining the input mesh. In contrast, methods based on minimizing, *e.g.*, an elastic [Liu et al. 2008] or other isometric energy [Rabinovich et al. 2017] have no such guarantee in the smooth setting—putting a far greater burden on discrete optimization to provide injectivity.

3.6 Cone Singularities

An important observation first made by Kharevych et al. [2006] in the context of conformal parameterization is that carefully-placed *cone singularities* can significantly reduce distortion. A complementary perspective is that such cones correspond to singularities in the frame field used for field-guided parameterization.

So far, we have considered a frame field X that varies smoothly across the entire surface. Suppose instead that the field can discontinuously jump in angle by multiples of $\pi/2$ along a network of curves $\Gamma \subset M$. To find a parameterization f aligned with X, we can now imagine cutting M along Γ before mapping it into the plane (Figure 14). If we pull back the metric of the plane under f, we get a metric on M that is intrinsically flat, except at vertices $p \in \Gamma$ that map to a corner $f(p)$ of angle different from 2π (Figure 14, *bottom right*). Gluing such a corner back together along cut edges forms a cone—hence the name *cone singularity.*

In terms of our rectangular parameterization, the only thing we need to consider is the behavior of the log scale factors u and v as we cross the cut network Γ. From Equation 4 and the definition of a, b and σ_1, σ_2, we have

$$df(Y) = \langle e^{u+v}X_1, Y\rangle e_1 + \langle e^{u-v}X_2, Y\rangle e_2,$$

for any Y. Suppose that X experiences a counter-clockwise 90-degree rotation as we cross Γ, *i.e.*, $\widetilde{X}_1 = X_2$ and $\widetilde{X}_2 = -X_1$. Then on the other side of the cut we have

$$\widetilde{df}(Y) = \langle e^{\widetilde{u}+\widetilde{v}}X_2, Y\rangle e_1 - \langle e^{\widetilde{u}-\widetilde{v}}X_1, Y\rangle e_2.$$

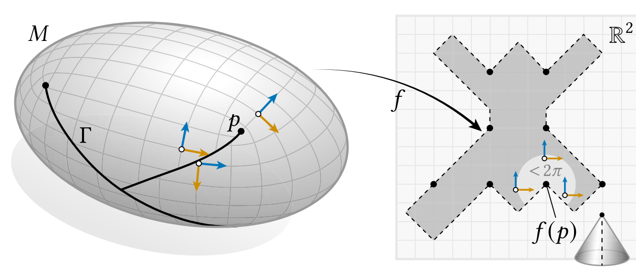


Fig. 14. Parametric distortion can be reduced by allowing *cone singularities* in the parameterization, corresponding to points p around which the total rotation of the frame is nonzero (e.g., ±90°). To do so, we cut the surface along a graph Γ that passes through all singular points and turns the surface M into a topological disk.

Since $df(Y)$ and $d\widetilde{f}(Y)$ are also related by a 90° rotation, we get

$$u = \widetilde{u} \qquad \text{and} \qquad v = -\widetilde{v}. \tag{14}$$

I.e., conformal scaling (which is the same in all directions) is unchanged, whereas authalic scaling (which has reciprocal magnitude for X_1 and X_2) is applied to the opposite axis. A clockwise 90° rotation yields the same relationship. If the frame experiences a 180° rotation, we instead have $\widetilde{u} = u$ and $\widetilde{v} = v$, *i.e.*, both scale factors are continuous at the cut. These relationships become linear conditions in our discrete formulation (Section 4).

Note that only the choice of singularities, and not the particular cut Γ, affects the quality of the final parameterization.

3.6.1 Singularity Index. We can assign each point $p \in \Gamma$ an *index*, equal to the number of times the frame rotates as we walk around p. More precisely, recall that $\omega_{12}(Y)$ gives the angular velocity of the frame in the direction Y. Hence, if γ_p is a small loop around p, then the index of p is

$$\operatorname{ind}(p) := \frac{1}{2\pi}\int_{\gamma_p} \omega^0_{12}.$$

Since we allow only 90° rotations, all singularities in our theory will have index $k/4, k \in \mathbb{Z}$, though one typically wants only singularities of index $\pm 1/4$ to avoid significant distortion (corresponding to irregular vertices of degree beyond 3 or 5 in the resulting quad mesh).

3.6.2 Preservation of Singularities. Recall that the frame X^θ used for rectangular parameterization differs from the reference frame X^0 by an angle θ at each point. Since θ varies continuously, the rotated field will (by construction) exhibit the same singularities as the reference field. In particular, applying Stokes' theorem, the new index around any point p is given by

$$\frac{1}{2\pi}\int_{\gamma_p}(\omega^0_{12} - d\theta) = \operatorname{ind}(p) - \cancelto{0}{\frac{1}{2\pi}\int_{\partial\gamma_p=\emptyset}\theta} = \operatorname{ind}(p).$$

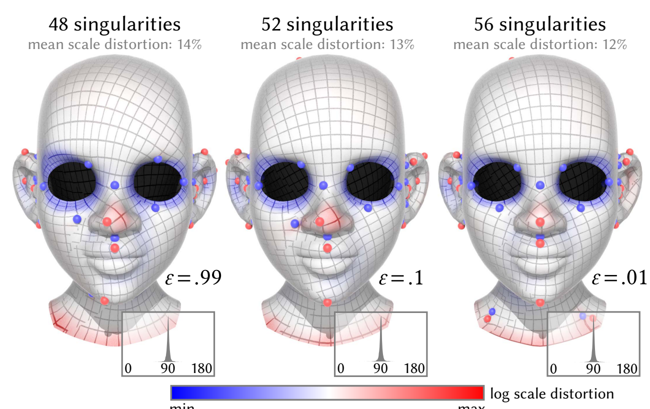


Fig. 15. We adopt a cone placement strategy that does not penalize rectangular stretching, yet still yields lower area distortion as the number of cones increases *(left to right)*. Even with many cones, the discrete parameterization remains almost perfectly rectangular. Inset histograms tabulate the angle between columns of the Jacobian of f^{-1} across triangles.

3.7 Discussion

Note that, due to Equation 5, our formulation implicitly relies on Poincaré's lemma, i.e., that on a simply connected domain, every closed differential form is also exact. However, our final algorithm still applies to nonsimply connected surfaces, since we effectively cut our surface into a disk (and apply boundary conditions that ensure seamless transitions across the cut); see Section 3.6.

It is also natural to wonder whether other, "simpler" equations established earlier in our derivation might provide a suitable starting point for rectangular parameterization. For instance, one could attempt to discretize Equation 2 and solve simultaneously for f, X_1, X_2, a, and b. However, it is unclear in this setting how to incorporate singularities, due to discontinuities in X_1, X_2 as well as a, b; moreover, one must still solve a system of nonlinear equations. Likewise, rather than work with a reference frame, one could try to solve Equation 7 directly, but would have to then incorporate the structure equations into the constraints, and would give no direct way to prescribe singularities. Our formulation, in terms of just an unknown rotation θ and log scale factors u, v, is much closer in spirit to classical conformal mapping, easily incorporates singularities, and leads to a robust discretization, as explored in the next section.

4 Discretization

We next describe how to discretize the two main equations needed for our algorithm: the integrability condition (Equation 10) and the Poisson equation (Equation 13). We consider a manifold oriented triangle mesh $M = (V, E, F)$, using indices $i \in V$ for vertices, pairs of indices $ij \in E$ for edges, and triples of indices $ijk \in F$ for triangles. We let H be the set of oriented *halfedges*, using $\overrightarrow{ij} \neq \overrightarrow{ji}$ to denote two opposite orientations, and let C be the set of all $3|F|$ triangle corners, using ϕ_i^{jk} to denote a quantity ϕ at corner i of triangle ijk.

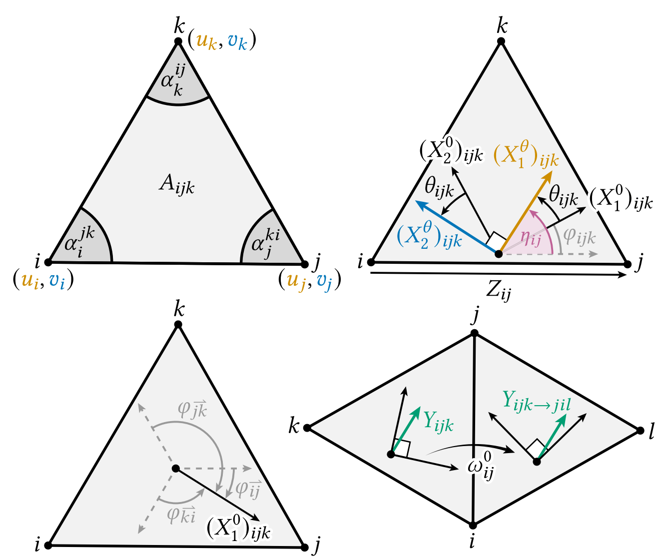


Fig. 16. The discrete quantities used in our formulation.

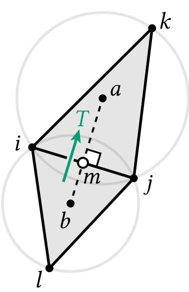
Discrete quantities are illustrated in Figure 16. We assume triangles have nonzero corner angles $\alpha_i^{jk} \in (0,\pi)$ and areas $A_{ijk} \in \mathbb{R}_{>0}$, and use vertex areas $A_i := \frac{1}{3}\sum_{ijk} A_{ijk}$. We let $w_{ij}^k := \frac{1}{2}\cot\alpha_k^{ij}$ denote the *cotan weight* for halfedge ij in triangle ijk [MacNeal 1949, Equation 3.17]; the weight w_{ij} for edge ij is then the sum of cotan weights over its halfedges (two on the interior; one on the boundary). Nominally, we encode log scale factors as values u_i, v_i at each vertex $i \in V$, though to handle singularities we more generally store values v_i^{jk} at corners (Supplement, Appendix E).

The reference frame is encoded as a pair of orthonormal vectors $(X_1^0)_{ijk}, (X_2^0)_{ijk} \in \mathbb{R}^2$ relative to a local coordinate system in each triangle ijk, which by convention we align with the first edge ij. We use φ_{ijk} to denote the angle of $(X_1^0)_{ijk}$ in this coordinate system; we use φ_{ab} to denote the angle of this same vector relative to the direction of any of the three halfedges $\overrightarrow{ab} \in \{\overrightarrow{ij}, \overrightarrow{jk}, \overrightarrow{ki}\}$. We likewise use θ_{ijk} to denote the rotation of the current frame X^θ relative to the reference frame X^0. Hence, the angle of the rotated frame direction X_1^θ is given by $\eta_{ijk}^\theta := \varphi_{ijk} + \theta_{ijk}$. We can *parallel transport* any vector Y_{ijk} to a neighboring triangle jil to get the vector

$$Y_{ijk\to jil} := R_{\varphi_{ji}-\varphi_{ij}+\pi} Y_{ijk}, \tag{15}$$

where R_φ denotes a counter-clockwise rotation by φ. Finally, the 1-form describing the rotation of the reference frame is encoded as a *dual discrete 1-form*, *i.e.*, a value ω_{ij}^0 at each (dual) edge that gives the counter-clockwise angle of rotation from X_{ijk}^0 to X_{jil}^0, where ijk, jil are the two triangles containing edge ij.

4.1 Discretized Integrability Condition



To discretize Equation 10, we integrate it along the dual edge ab associated with each edge $ij \in E$. Throughout we assume each interior edge ij is contained in two triangles ijk, jil, and let a and b be the circumcenters of ijk and jil (*resp.*), which we connect to the edge midpoint m to form the dual edge. We use T to denote the intrinsic unit vector tangent to dual edge ab. The left-hand side of Equation 10 is discretized via *discrete exterior calculus* [Desbrun et al. 2006], yielding

$$\omega_{ij}^0 - (\theta_{ijk} - \theta_{jil}). \tag{16}$$

To discretize the right-hand side, consider any facewise affine function interpolating values ϕ_i at vertices. Integrating over the dual edge ab, we have

$$\int_{ab} (D_{X_1^\theta}\phi)\sigma_2^\theta = \int_{ab} \langle X_1^\theta, \nabla\phi\rangle\langle X_2^\theta, T\rangle\, ds,$$

and similarly for $(D_{X_2^\theta}\phi)\sigma_1^\theta$. To obtain a more explicit formula, we perform integration piecewise over the two segments am and mb. We then evaluate these expressions for $\phi = u - v$ and $\phi = u + v$, and add these terms to Equation 16. After some simplification, the contribution of triangle ijk to the right-hand side is

$$\rho_{ij}^k(u,v,\theta) := \tfrac{1}{2}\cot\alpha_k^{ij}\Big((u_j - u_i) + \cos(2\eta_{ijk}(\theta))(v_j - v_i) + \sin(2\eta_{ijk}(\theta))\big(\cot\alpha_j^{ki}(v_k - v_i) + \cot\alpha_i^{jk}(v_k - v_j)\big)\Big).$$

For any boundary edge ij, we let $\rho_{ji}^l = 0$. Also, since this formula assumes the reference frame is expressed relative to edge ij, we must re-express η relative to edges jk and ki when evaluating terms for those edges. In particular, if $\eta_{ij} := \varphi_{ijk} + \theta_{ijk}$, then

$$\eta_{jk} := \eta_{ij} - (\pi - \alpha_j^{ki}) \quad \text{and} \quad \eta_{ki} := \eta_{jk} - (\pi - \alpha_k^{ij}).$$

To get the full residual corresponding to edge ij, we add the corresponding term ρ_{ji}^l from the neighboring triangle jil, and subtract the discrete left-hand side, yielding our final discrete condition

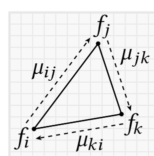
$$\boxed{F_{ij}(u,v,\theta) := \rho_{ij}^k(u,v,\theta) + \rho_{ji}^l(u,v,\theta) + (\theta_{ijk} - \theta_{jil}) - (\omega_{12}^0)_{ij} = 0,} \tag{17}$$

which must hold for each edge $ij \in E$. Assembly of the overall residual and its Jacobian is detailed in Appendix E of the supplement.

When $v = 0$ (pure conformal scaling), Equation 17 reduces to a standard (cotan) discretization of Equation 11. Note also that Equation 17 is invariant to rotations $\eta \mapsto \eta + \pi$, correctly reflecting the fact that we only care about *how much* scaling there is along the horizontal/vertical axis, and not the particular direction.

4.2 Discretized Poisson Equation

We discretize the Laplace and divergence operators in Equation 13 using the usual cotan formula (see Appendix B of the supplement).



4.2.1 Discrete Differential. We discretize the term μ in Equation 13 by integrating the right-hand side of Equation 12 along each edge. Geometrically, $\mu_{ij} \in \mathbb{R}^2$ gives the vector along edge ij in the parameter domain. Ideally, we could sum these vectors incrementally to get final coordinates f_i. Due to discretization error, however, they do not fit together perfectly—we instead solve a discrete Poisson equation (Supplement, Appendix B) that effectively finds a least-squares solution.

More explicitly, for each edge ij we compute average scale factors

$$a_{ij} := e^{(u_i+u_j+v_i+v_j)/2}, \qquad b_{ij} := e^{(u_i+u_j-v_i-v_j)/2},$$

which for $v = 0$ agree with the edge length ratios appearing in the theory of discrete conformal equivalence [Springborn et al. 2008, Equation 2]. Likewise, since $\sigma_i(Z) = \langle X_i^\theta, Z\rangle$, we take the average inner product of the vector Z_{ij} along each edge with the frame directions X_i^θ in the two neighboring triangles—being careful to account for different coordinate systems. Overall, each triangle ijk makes the following contributions to equations along its three edges:

$$\begin{aligned}
\mu_{ij}^k &= \ell_{ij}\begin{bmatrix} +a_{ij}\cos(\eta_{ijk}) \\ +b_{ij}\sin(\eta_{ijk}) \end{bmatrix}, \\
\mu_{jk}^i &= \ell_{jk}\begin{bmatrix} -a_{jk}\cos(\eta_{ijk} + \alpha_j^{ki}) \\ -b_{jk}\sin(\eta_{ijk} + \alpha_j^{ki}) \end{bmatrix}, \\
\mu_{ki}^j &= \ell_{ki}\begin{bmatrix} -a_{ki}\cos(\eta_{ijk} - \alpha_i^{jk}) \\ -b_{ki}\sin(\eta_{ijk} - \alpha_i^{jk}) \end{bmatrix}.
\end{aligned}$$

For each edge ij contained in a pair of triangles ijk, jil we then let

$$\mu_{ij} := \mu_{ij}^k + \mu_{ji}^l. \tag{18}$$

For edges ij on the domain boundary, we let $\mu_{ji} = \mu_{ij}$, *i.e.*, we effectively just use the frame from the single triangle containing ij.

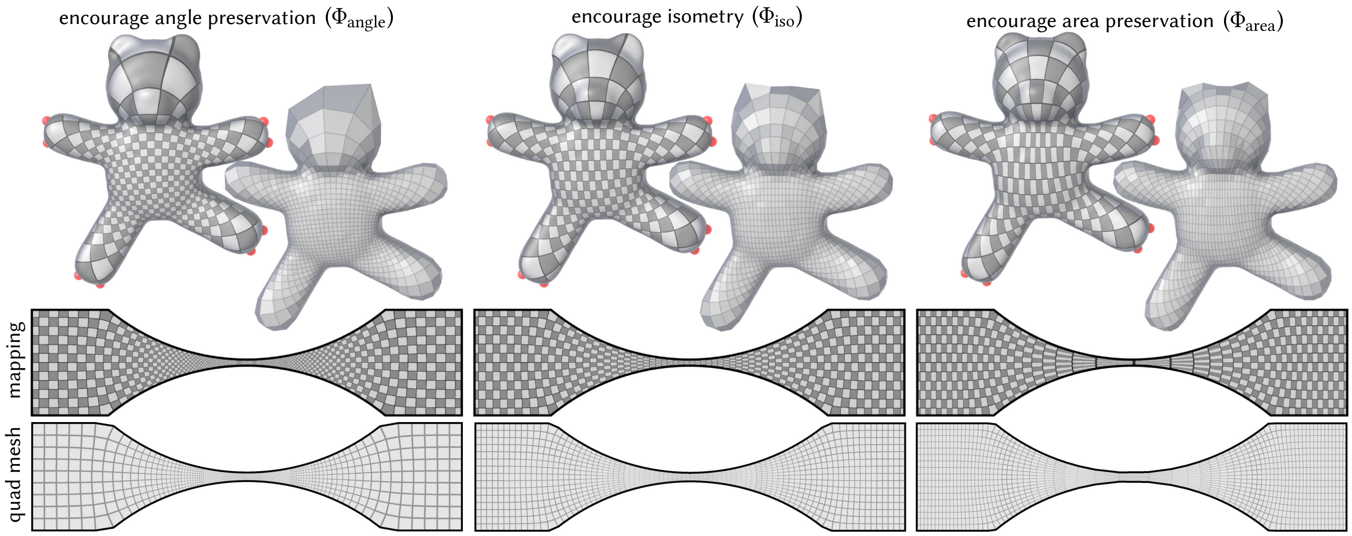


Fig. 17. Since our fields are rectangular by construction, the shear-free condition does not "fight" with other objectives. We are therefore free to minimize any user-defined distortion metric Φ over the space of rectangular parameterizations. Here we minimize angle, metric, and area distortion *(left to right)*.

5 Algorithm

Our final algorithm is comprised of three basic steps:

(1) Compute a reference frame (or cross field) X^0, using any algorithm. Section 6.1 details algorithms used in this paper.
(2) Solve for scale factors u, v and rotations θ that transform X^0 into an integrable rectangular field (Section 5.1).
(3) Recover the final parameterization from this field (Section 5.2).

A diagram of these steps is shown in Figure 5; detailed pseudocode can be found in the supplement (Appendix E).

5.1 Optimization

Given a reference frame X^0, we use constrained optimization to find log scale factors u, v and angles θ that satisfy our integrability condition (Equation 10) while minimizing an objective $\Phi(u, v, \theta)$ that encodes quality criteria such as angle or area preservation (Section 5.1.1 gives several examples). Letting F be our discretized integrability equation from Equation 17, we solve the problem

$$\min_{\substack{u,v:V\to\mathbb{R}\\ \theta:F\to\mathbb{R}}} \Phi(u, v, \theta) \qquad \text{s.t.} \qquad F(u, v, \theta) = 0. \tag{19}$$

5.1.1 Objective Functions. The constraint $F = 0$ restricts our search to shear-free rectangular maps; we can then optimize any other objective Φ over this space. For instance:

- $\Phi_{\text{iso}}(u, v, \theta) := \sum_{i\in V} A_i(u_i^2 + v_i^2)$ — tries to make parameterization as isometric as possible
- $\Phi_{\text{area}}(u, v, \theta) := \sum_{i\in V} A_i u_i^2$ — encourages area preservation by minimizing scale distortion
- $\Phi_{\text{angle}}(u, v, \theta) := \sum_{i\in V} A_i v_i^2$ — encourages angle preservation by minimizing stretching
- $\Phi_{\text{align}}(u, v, \theta) := \sum_{ijk\in F} A_{ijk}\theta_{ijk}^2$ — encourages alignment with the given field X^0

Figure 17 demonstrates the effect of several different objectives. Since u and v are *logarithmic* scale factors, we obtain an infinite energy barrier to degenerate maps—further encouraging local injectivity beyond the reasons already discussed in Section 3.5.1. One can of course use other objectives, *e.g.*, Section 6.6 describes how to find a parameterization close to a *Chebyshev net*.

5.1.2 Regularizer. Equation 10 links the rotation speed of X^θ with derivatives of the log scale factors u and v. In theory, the frame can rotate arbitrarily fast; in practice, however, rotation speed must be limited. For instance, our discretization does not distinguish between a frame that makes a full turn along a dual edge, and one that is not turning at all. This ambiguity introduces numerical instability, preventing convergence of the optimizer. Instead of constraining rotation along dual edges, we regularize variation in the authalic factor v by adding a discrete Dirichlet energy to our objective:

$$\Phi_{\text{reg}}(u, v, \theta) := \varepsilon v^\top L v.$$

As $\varepsilon \to \infty$ the authalic factor is constant, and Equation 10 reduces to the well-behaved linear integrability condition for conformal maps (Equation 11). By default we use $\varepsilon = 10^{-2}$, except in Section 6.6 where more variability is needed and we set $\varepsilon = 10^{-4}$.

5.1.3 Boundary Conditions. Rectangular parameterizations accommodate a variety of boundary conditions, naturally expressed as linear constraints on the log scale factors u, v and frame rotations θ.

Feature Alignment. To align the parameterization to boundary or feature curves (as in Figure 8), we simply constrain θ in triangles adjacent to feature edges, and omit integrability constraints $F_{ij} = 0$ for these edges. In our case, we start with a feature-aligned reference field X^0 and set $\theta = 0$.

Element Shape. We can also prescribe the shape of rectangles—and ultimately, mesh elements—near the boundary (Figure 18). Note

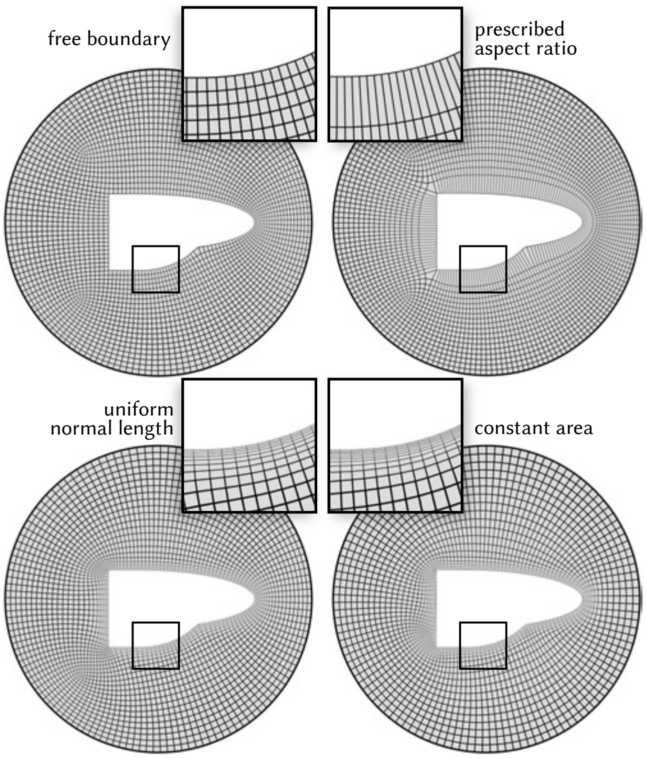


Fig. 18. Using scale factors as primary variables provides direct control over element sizing near the boundary. Here we prescribe values of u and v along the inner boundary, leaving the outer boundary free. Automatic cone singularities help adapt the rest of the field to these boundary conditions.

that ab and a/b quantify local area and aspect ratio, *resp.*, and that

$$u = \tfrac{1}{2}\log(ab) \qquad \text{and} \qquad v = \tfrac{1}{2}\log(a/b).$$

Assuming X_1^0 is tangent to the boundary, we then have:

- $u|_{\partial M} = p$ — prescribed area
- $v|_{\partial M} = p$ — prescribed aspect ratio
- $(u - v)|_{\partial M} = p$ — prescribed normal length
- $(u + v)|_{\partial M} = p$ — prescribed tangent length

for any prescribed function p (defined at boundary vertices). As with conformal maps, not all boundary conditions are feasible [Sawhney and Crane 2017, Section 3.2.1], and we do not have a clear characterization of compatibility conditions—in practice the solver may simply fail if the boundary is overconstrained. Note also that extreme boundary conditions can induce distortion (Supplement, Figure 32), which can be mitigated via cone singularities (Figure 18).

5.1.4 Solver. To solve Equation 19, we apply a constrained Newton method to the first order optimality conditions (see Supplement, Appendix C). In practice we always converge to a valid solution, despite the nonlinearity of this problem. The initial guess for u, v, and θ seems to have no impact on results (Section 6.2). We hence initialize with $u = v = \theta = 0$, *i.e.*, no scaling or rotation of X^0.

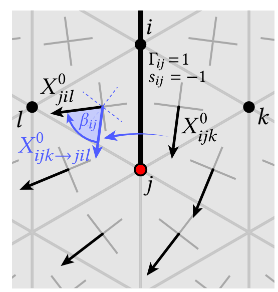
5.2 Recovering Parameterization

After optimization, we have log scale factors u, v and frame rotations θ that describe some rectangular parameterization f. However, we still need to recover f itself, *i.e.*, we need to solve a discrete Poisson equation (corresponding to Equation 13) for the unknown vertex coordinates $f : V \to \mathbb{R}^2$. This equation is solved independently for each scalar coordinate function, possibly after cutting the surface into a topological disk. As in many parameterization methods, the main implementation challenge is handling cuts through cones. See Supplement, Appendix Band Dfor further details.

5.3 Cone Singularities

To incorporate singularities, we start with a *cross field* [Vaxman et al. 2016, Section 2], and extract a frame field X^0 with $\pm 90°$ or $180°$ jumps. Since the initial field is free to rotate, only its singularities—and not the field quality—will have an impact on the solution (unless we explicitly encourage alignment with X^0).

5.3.1 Jump Curve and Sign Bit. We first identify edges where X^0 is discontinuous, corresponding to the cut network Γ from Section 3.6. In accordance with Equation 14, we also track the change in the sign of v as we cross each edge. We mark edges in the cut network via a binary labeling $\Gamma : E \to \{0, 1\}$, and encode flips in the sign of v via values $s : E \to \{-1, +1\}$.



To determine these values, we initialize $\Gamma_{ij} \leftarrow 0$ and $s_{ij} \leftarrow 1$ on all edges. We then perform breadth-first traversal: starting in any triangle ijk, we pick any of the four frames X^0_{ijk} aligned with the cross field, and transport it to each neighboring triangle jil, via Equation 15. If jil has not been visited, we set X^0_{jil} to the cross direction closest to the transported frame $X^0_{ijk\to jil}$. Otherwise, if it has been visited, we let $\beta_{ij} \in [-\pi, \pi)$ be the angle from the transported frame to the existing value of X^0_{jil}. If X^0_{jil} is not the closest of the four possible rotations (*i.e.*, if β_{ij} is greater than $\pm\pi/4$), there is a jump, and we set Γ_{ij} to 1. Moreover, if the closest frame differs by a quarter rotation (*i.e.*, if $\beta \in [\pi/4, 3\pi/4)$ or $\beta \in [-3\pi/4, -\pi/4)$), then v requires a sign flip, and we set s_{ij} to -1. Traversal continues until all faces have been visited.

Per-edge sign bits s_{ij} in turn determine the sign of v_i^{jk} at each triangle corner, relative to a single value v_i stored at vertex $i \in V$. In particular, let $j_0, \ldots, j_m$ be the neighbors of i in counter-clockwise order. Then the value of v at any corner around vertex i is

$$v_i^{j_n j_{n+1}} := s_i^{j_n j_{n+1}} v_i,$$

where the sign s for each corner is given by

$$s_i^{j_n j_{n+1}} := \prod_{p=0}^{n-1} s_{ij_{p+1}}. \tag{20}$$

5.3.2 Automatic Singularities. So far we have assumed cone singularities are fixed ahead of time. However, we can adapt the method of Myles and Zorin [2013] to find singularities that help minimize the objective Φ. We start by cutting the mesh into a disk along edges (as

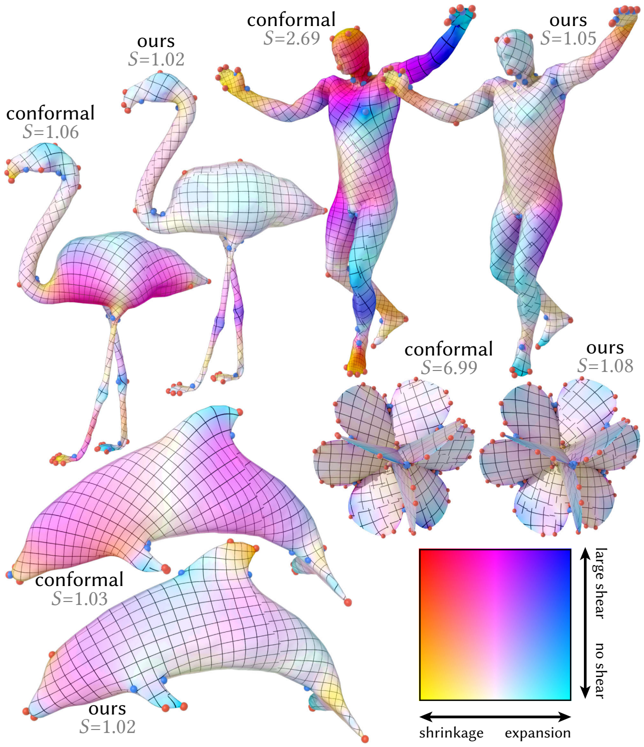


Fig. 19. Our greedy cone placement strategy for rectangular parameterization works better than even optimal cone placement strategies for conformal maps. Here we compare with the method of Li et al. [2022], yielding lower total *root mean square stretch S* [Sander et al. 2001, Section 3]. Colors plot local scaling $\sqrt{\varsigma_1\varsigma_2}$ and shearing ς_1/ς_2 on a log scale, where ς_1, ς_2 are singular values of the affine map from 3D to 2D triangles.

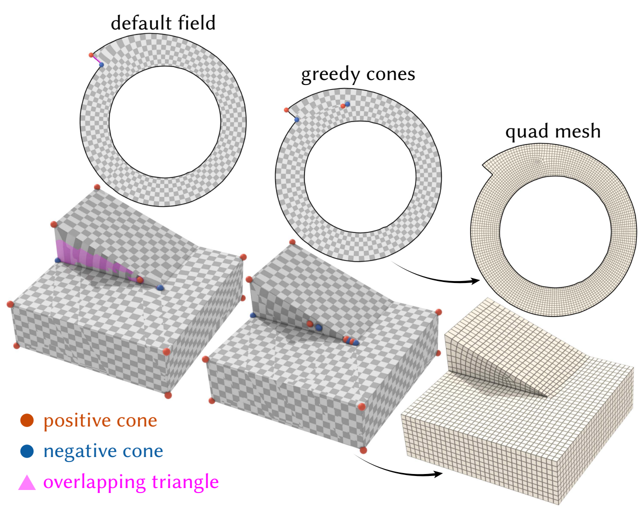
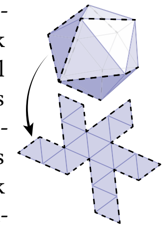


Fig. 20. As with any field-guided method, solution quality depends on the global field topology. Here, for example, our default field generation strategy struggles on meshes with challenging *limit cycles (left)*. Switching to automatic cone placement yields a flip-free parameterization *(center)* that is easily quantized and contoured to generate a quad mesh *(right)*.



shown in the inset), and flatten it using the procedure from Sections 5.1 and 5.2. We then glue back together the edge ij that incurs the smallest residual F_{ij} (Equation 17). If the angle β_{ij} between frames across the glued edge is closest to a nonzero multiple of $\pi/2$, we add this edge to the curve Γ. This process is repeated until all edges are glued back together, at which point Γ determines the configuration of singularities (*à la* Section 5.3.1). To make this process more efficient, we glue batches of edges together after each solve, rather than just a single edge. In particular, we glue the largest batch of edges whose absolute error $|F_{ij}|$ sums to less than π.

Like Myles *et al.*, we do not make any claim of global optimality—but do obtain good results in practice, even for challenging field topology (Figure 20). Conversely, optimal cones for *conformal* parameterization can perform poorly for rectangular parameterization (Figure 19). As always, there is a tradeoff between number of singularities (which reduce field smoothness) and overall distortion. We control this tradeoff with a regularization term

$$\Phi_{\text{smooth}} := \varepsilon_{\text{smooth}} \sum_{ij \in E} w_{ij} ((\omega_{12}^0)_{ij} + \theta_{ijk} - \theta_{jil})^2.$$

The effect of $\varepsilon_{\text{smooth}}$ is illustrated in Figure 15.

6 Evaluation and Examples

6.1 Implementation

We implemented our algorithm in MATLAB, in double precision. Timings were measured a 2019 iMac with a 3.6GHz Intel Core i9 processor and 32GB of RAM. On our largest examples (about 50k input triangles), our method took no more than about a minute for the parameterization step (excluding the cost of reference field generation and mesh extraction).

6.1.1 Reference Field Generation. To generate a reference field X^0 with prescribed singularities, we use the *trivial connections* method of Crane et al. [2010]. Alternatively, for organic surfaces, we start with a *globally optimal direction field à la* Knöppel et al. [2013], then improve this field via iterative *Ginzburg-Landau (GL)* smoothing [Viertel and Osting 2019]. Finally, for hard surface (e.g., CAD) models, we modify the definition of parallel transport near acute corners *à la* Desobry et al. [2022], identify sharp edges where the angle between face normals is greater than 40°, and again apply GL smoothing. For curvature-aligned examples, we use the method of Cohen-Steiner and Morvan [2003] (using a stencil of nine edges from four adjacent triangles), plus a small amount of GL smoothing.

6.1.2 Mesh Extraction. For meshing examples, we need to (i) quantize the parameterization f computed by our algorithm onto a regular grid, and (ii) contour the quantized parameterization to obtain mesh elements. For the first step we use the method of Coudert-Osmont et al. [2024]. For the second step we use the *QEx* method [Ebke et al. 2013], though since the method of Coudert-Osmont et al. [2024] already requires the parameterization to be locally injective, we do not rely on the robustness properties of this method (and simply use the QEx code for convenience).

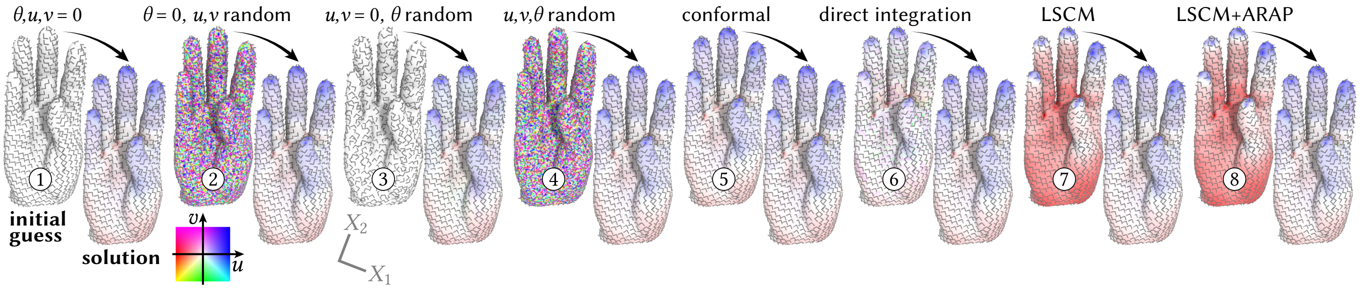


Fig. 21. Empirically, we tend to get the same solution independent of how we initialize the solver—suggesting that our results may be globally optimal (even though the formulation is nonconvex). Here we plot the initial scale factors u, v and frame orientation θ, obtaining the same solution across constant, random, and structured initializations. We consistently observe this behavior for examples throughout the paper.

6.2 Initialization, Optimality, and Robustness

Given the nonconvexity of our optimization problem, it is natural to wonder if we always get a high-quality solution. Importantly, to get a valid, integrable field, it is sufficient to find any feasible point satisfying Equation 17—*we do not need a globally optimal solution.* In practice, however, we do appear to obtain optimal solutions, since very different initialization strategies lead to identical results. *E.g.*, in Figure 21 we initialize with (1) $u, v = 0$ (no scaling) and $\theta = 0$ (no rotation of the reference frame), (2) $\theta = 0$ and u, v sampled uniformly at random from $[-1, 1]$, (3) $u, v = 0$ and uniformly random θ, (4) uniformly random u, v, and θ, (5) an intrinsic conformal parameterization where $v = 0$, u solves $\Delta u = d\omega^0$, and θ solves $\Delta\theta = d * \omega^0$, (6) u, v and θ obtained by breadth-first integration of a nonintegrable reference field, (7) u, v, θ extracted from *LSCM* [Lévy et al. 2002], and (8) u, v, θ extracted from LSCM followed by *ARAP* [Liu et al. 2008]. If we intentionally try to break the solver (*e.g.*, using large random initial values) it can get stuck near a saddle point, but we did not encounter this problem naturally on any of our models. In practice, we therefore default to just $u = v = \theta = 0$.

As seen in Figure 22, RSP is also robust to difficult singularity configurations. In all cases we obtain a valid rectangular parameterization, and highly rectangular quads (away from singular points).

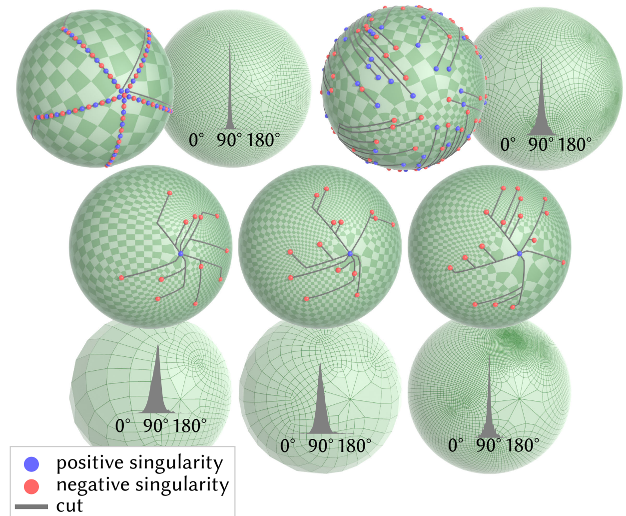


Fig. 22. Starting with an integrable rectangular field helps with robustness throughout the rest of the pipeline. Here we obtain valid, highly-rectangular meshes even for difficult cone configurations that are structured *(top left)*, random *(top right)*, or have increasingly large index *(bottom row)*.

6.3 Convergence

Under refinement of the input mesh M, we observe linear convergence to a smooth rectangular parameterization. Consider for instance Figure 24. Here, letting (Y_1, Y_2) be the Jacobian of the map f, *non-orthogonality* is the deviation of the angle between Y_1 and Y_2 from $\pi/2$; *integration error* is the Frobenius norm of the difference between (X_1^θ, X_2^θ) and (Y_1, Y_2); *angle deviation* is the angle between (X_1^θ, X_2^θ) and the rotation matrix R closest to (Y_1, Y_2); and *scale/stretch deviation* is the difference between u, v and the log magnitudes of (Y_1, Y_2), *resp.* In each case we plot the area-weighted ℓ_2 norm. Figure 23 again indicates that this error is quite small in absolute terms even on coarse meshes, and becomes smaller for fine meshes. Overall, Equation 17 appears to do an excellent job of characterizing integrability even at the discrete level.

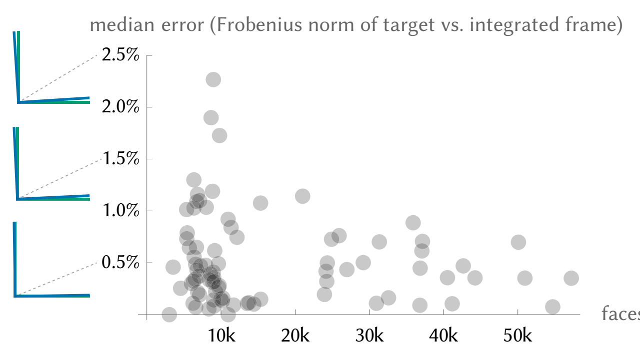


Fig. 23. As predicted by the smooth theory, our discrete frame fields are almost perfectly integrable. Here we plot numerical integration error as a function of mesh size, for the 88 MAMBO models in our dataset; notice that error tends to get smaller as the mesh gets finer. As visualized on the left, integration error is tiny even for very coarse meshes.

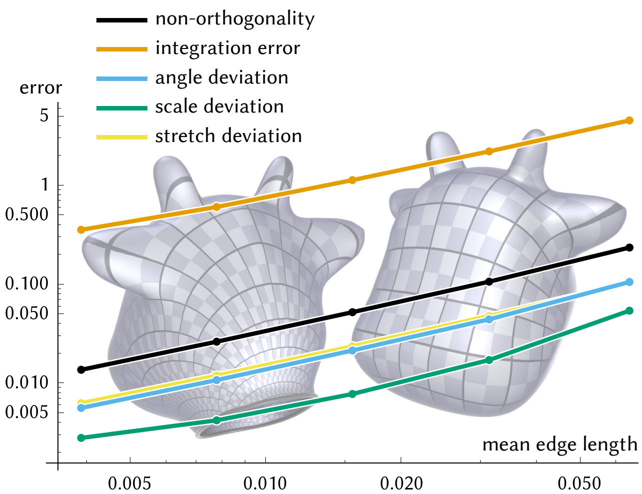


Fig. 24. Our smooth theory guarantees that fields are exactly orthogonal, and exactly integrable. In the discrete case, any failure of integrability is hence due only to discretization error—which vanishes under mesh refinement. Here we compare the gradient of the final piecewise linear parameterization (shown in the background) to the input field, achieving a linear reduction in error with respect to mean edge length of the input mesh.

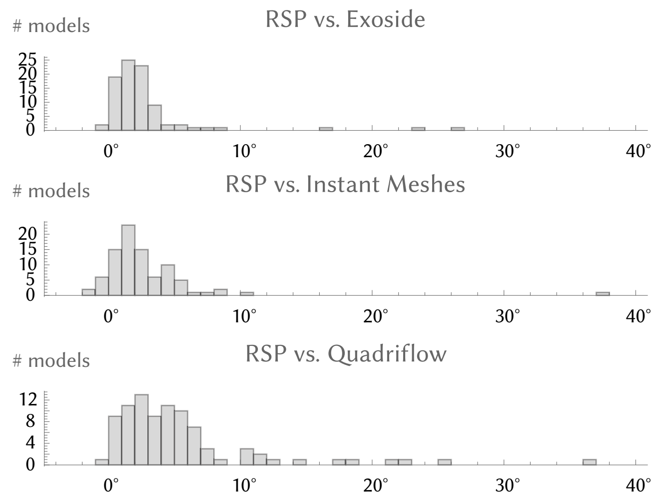


Fig. 25. Increase in error relative to RSP (on the MAMBO dataset), quantified as the mean deviation of corner angles from 90°. Overall, our method produces more rectangular elements than established alternatives, in turn providing better geometry (*e.g.*, planarity and discrete conjugacy) and faster solver convergence (*e.g.*, for quadrilateral FEM).

6.4 Quadrilateral Mesh Generation

A central motivation for orthogonal frame field generation is field-guided quadrilateral remeshing [Bommes et al. 2013b]. As noted in Section 1, rectangular elements are especially desirable in this setting due to, *e.g.*, better asymptotic convergence rates for FEM [Arnold et al. 2002], superior mechanical properties [Pottmann et al. 2007], and optimal geometric approximation (Section 6.4.2). However, an orthogonal frame alone is not sufficient to obtain near-rectangular mesh elements: in general, the initial frame field will be distorted during the parameterization step (Figure 3), quantization of the parameterization (to "snap" it to a regular grid [Bommes et al. 2013a]), and extraction of quadrilateral elements [Ebke et al. 2013], where angles around irregular-degree vertices must deviate from 90°.

Though some distortion is inevitable, starting with a near-integrable rectangular field significantly improves mesh quality relative to state of the art methods. Here we compare against open source implementations of *Instant Meshes* [Jakob et al. 2015], *QuadriFlow* [Huang et al. 2018], and *QuadWild* [Pietroni et al. 2021], as well as *Exoside* [Rouca 2019], commercial software invoked by thousands of users via integration with *ZBrush*. For all methods we use the authors' reference implementation with default parameters, targeting approximately the same element count.

6.4.1 Hard Surface Models. We evaluated performance on hard surface meshing using 88 CAD models from the MAMBO dataset [Ledoux 2022] (Figure 8). Overall, RSP yields more rectangular elements than past methods, on all but a few examples where the mean difference in worst angle is around just 1° (Figure 25). (See supplemental material for mesh data and more detailed statistics.) We omit models where quantization via [Coudert-Osmont et al. 2024] failed—though our field optimization and parameterization steps succeed for all models. In a real implementation, one could achieve greater robustness with a less simplistic integration step and/or more robust quantization (see Section 7).

6.4.2 Organic Models. Asymptotically, the best quadrilateral approximation of a positively-curved smooth surface aligns mesh edges with principal curvature directions [D'Azevedo 1999]. To get such a mesh, we first compute principal directions E_1, E_2 and curvatures κ_1, κ_2 in each triangle, via the method of Cohen-Steiner and Morvan [2003]. These directions are smoothed via a vector diffusion equation $\frac{\partial}{\partial t} E_i = a\Delta^\nabla E_i$ to obtain the reference field X^0, where Δ^∇ is the *connection Laplacian* of Knöppel et al. [2013], and the variable diffusion rate $a := 1/(|K| + \varepsilon)$ helps adapt smoothing to local feature size [Sharp and Crane 2018, Section 5.6] (where K is Gaussian curvature and $\varepsilon = 10^{-3}$). We then minimize an objective

$$\Phi_{\text{curv}}(u, v, \theta) := \Phi_{\text{align}}(u, v, \theta) + \|v - v^0\|_2^2,$$

where $\|\cdot\|_2$ is the area-weighted ℓ_2 norm, and $v^0 := \frac{1}{2}\log(r)$ encourages rectangles to have a target aspect ratio of $r := |\kappa_1/\kappa_2|$, which is optimal for minimizing L_∞ gradient error [Shewchuck 2002; Nebel and Chern 2023]. Examples are shown in Figure 27, and a comparison with other methods is provided in Figure 26.

6.5 Finite Element Simulation

Beyond standard synthetic metrics of element quality, we evaluated how overall mesh quality affects performance in end-to-end simulation of *microfluidic circuits*. These systems drive fluid through narrow channels to induce interactions between chemical compounds or biological species, enabling "lab on a chip" devices [Oh et al. 2012]. Since channels are shallow relative to overall circuit dimensions, they are well-modeled by two-dimensional low-Reynolds-number flows. To track mixing dynamics we use quadrilateral FEM

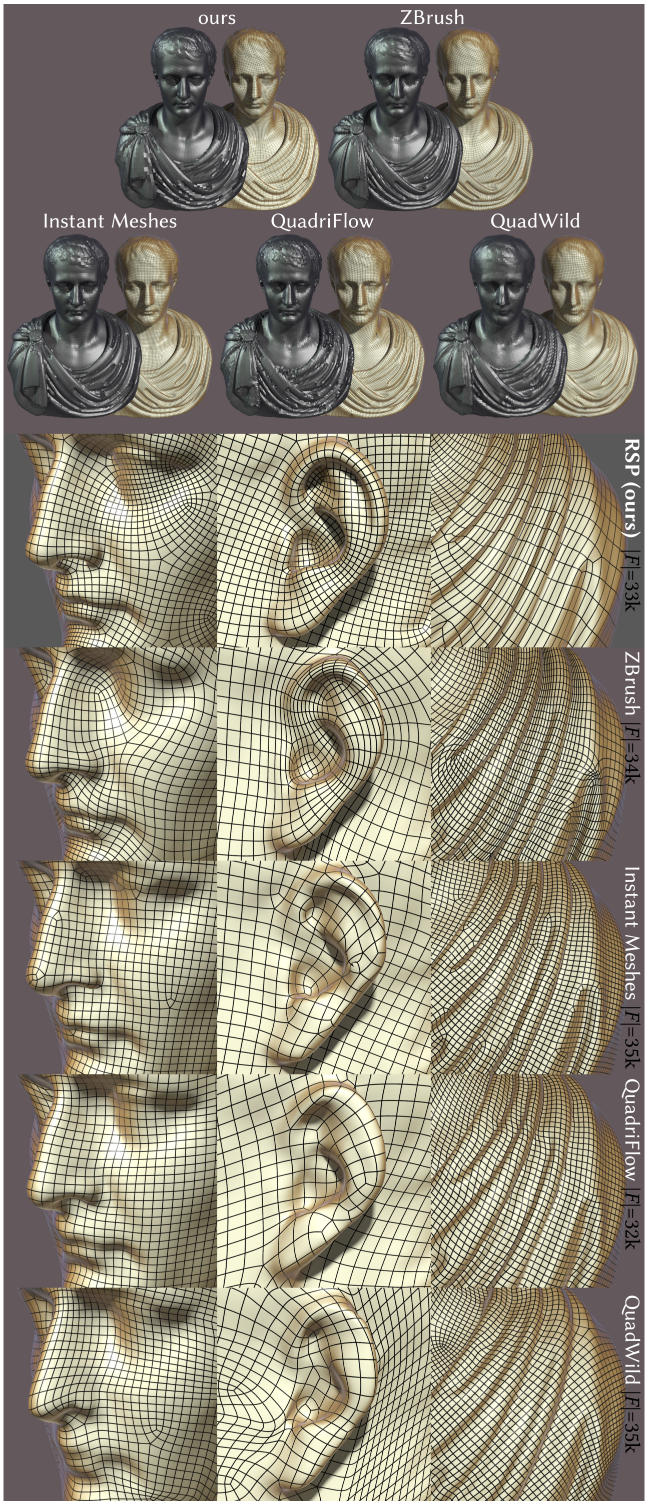


Fig. 26. For remeshing smooth, organic shapes our integrable rectangular approach leads to good normal approximation, and a natural flow of mesh topology that improves even on retopology tools in the commercial *ZBrush* modeling package (Exoside). Here we show both flat shaded (gray) and corresponding Catmull-Clark subdivision (yellow) surfaces.

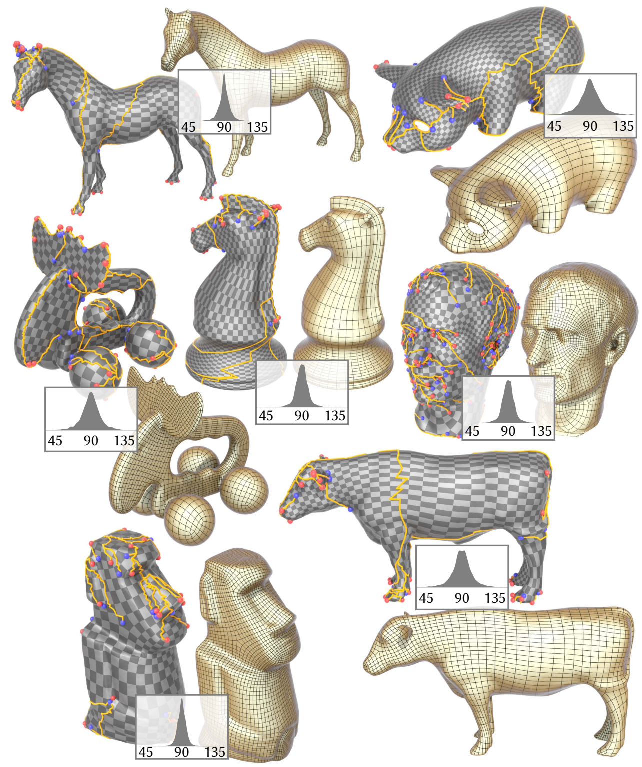


Fig. 27. Curvature-guided parameterizations *(checkerboard pattern)*, showing cuts *(yellow)* and positive/negative cone points *(red/blue)*, as well as the resulting quad mesh *(visualized as Catmull-Clark surfaces)*. Histograms show the distribution of corner angles, which are tightly centered around 90°.

for Stokes flow, and a low-order upwind finite volume scheme for advection–diffusion. In such problems, meshes cannot be tailored to a single solver, but must work well across different discretizations.

Figure 29 demonstrates a *gradient generator*, where compounds progressively flow through several *serpentine mixers* to achieve a uniform concentration gradient. Such mixers compensate for the lack of turbulent mixing in laminar flow. RSP provides an anisotropic quad mesh which, due to boundary alignment, naturally follows flow characteristics. We first solve a time-independent Stokes' equation

$$\Delta \mathbf{u} + \nabla p = 0, \qquad \nabla \cdot \mathbf{u} = 0,$$

for velocity $\mathbf{u}$ and pressure p. Positive Dirichlet conditions on $\mathbf{u}$ at inlets drives two species through the circuit. We use zero-Dirichlet conditions on p at the outlet, and zero-Neumann conditions everywhere else. We then solve a time-dependent advection equation

$$\tfrac{d}{dt} c = -\nabla \cdot (c\mathbf{u})$$

for the concentration ϕ, until reaching equilibrium. At each step we update the concentration c in each quad via

$$c = c + \tau \frac{1}{A} \sum_{i=1}^{4} \ell_i \mathbf{n}_i \cdot \mathbf{u}_i c_i^+,$$

where A is the quad area, ℓ_i is the length of the ith edge, $\mathbf{n}_i$ is its inward unit normal, $\mathbf{u}_i$ is the fluid velocity, and c_i^+ is the upwind

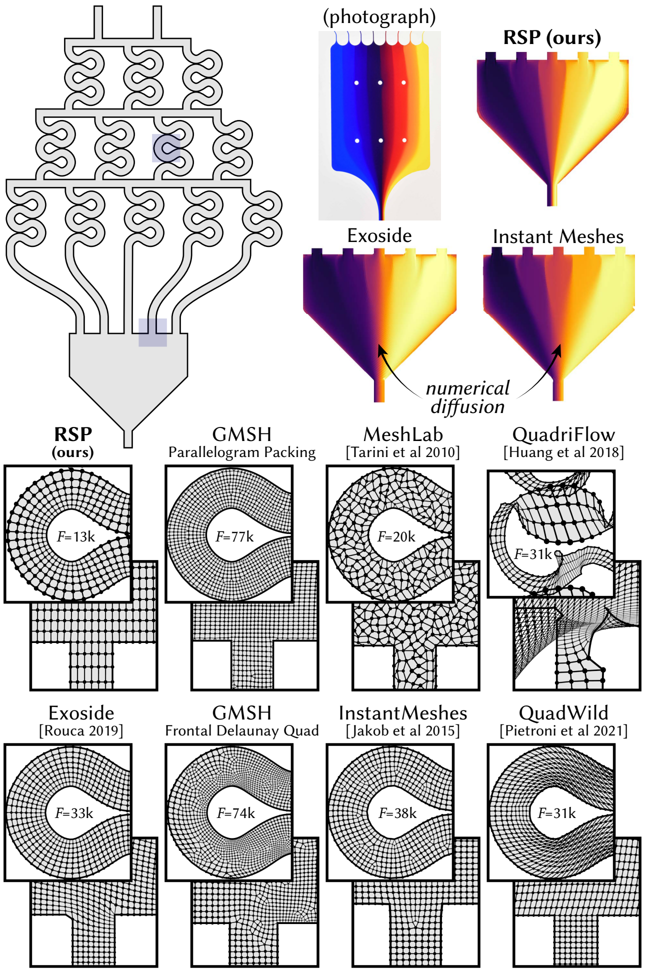


Fig. 28. Though 2D quad meshing is often viewed as a "solved problem," it is deceptively hard to obtain correct results for end-to-end simulation. Here we test a variety of mesh generators on our microfluidic simulation. Most meshes led to INF/NaN values; the few that gave meaningful values still failed to reproduce the piecewise concentration observed in the real physics, due to numerical diffusion (even at higher element counts than RSP).

concentration at edge i, equal to the value of c in either the quad or its neighbor depending on the sign of $\mathbf{n}_i \cdot \mathbf{u}_i$. The time step $\tau > 0$ is chosen according to a CFL condition, namely, τ must be smaller than the minimum over all quads of $A / \sum_{i=1}^{4} |\mathbf{u}_i \ell_i \cdot \mathbf{n}_i|$.

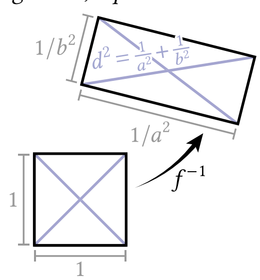
As shown in Figure 29, simulation on the mesh generated by RSP yields not only the expected velocity and linear pressure profile, but also a concentration gradient stratified over the number of mixers in the final stage—just as in physical experiments. Figure 28 compares with other mesh generators, including those listed in Section 6.4, plus tools from *MeshLab* [Tarini et al. 2010] and *Gmsh* [Remacle et al. 2013; Baudouin et al. 2014]. Though some of these methods are "robust" in the sense that they reliably generate a topologically valid mesh, element quality can still be quite poor. As a result, most meshes lead to invalid solutions with INF/NaN values. Moreover, even the successful examples (Exoside, Instant Meshes) exhibit non-physical diffusion—even with significantly more elements than the RSP mesh. Post-processing to improve element orthogonality will not solve this problem, since not all mesh connectivity admits a rectangular mesh aligned to flow characteristics. The fact that RSP provides explicit control over singularities and boundary behavior is hence hence a major advantage for real-world anisotropic mesh generation.

6.6 Chebyshev Nets

A *Chebyshev net* is a mapping $\phi(x, y) : U \to \mathbb{R}^n$ over a region $\Omega \subset \mathbb{R}^2$ such that

$$\left|\frac{\partial \phi}{\partial x}\right| = \left|\frac{\partial \phi}{\partial y}\right| = d$$

for some constant $d > 0$, *i.e.*, that preserve lengths in the two coordinate directions. Such nets provide a model of loosely woven textiles made from inextensible threads, which easily shear but do not scale or stretch (cheesecloth, gauze, fishnet, *etc.*). Finding a Chebyshev parameterization of a surface hence provides a cutting pattern for which the surface can be made out of a piece of cloth. For instance, a hemisphere can be perfectly covered by a single diamond-shaped piece of inextensible cloth, *à la* Figure 30, *top center*.



A slight tweak of RSP enables us to compute high-quality Chebyshev parameterizations. The basic observation is that infinitesimal squares in the plane are mapped to infinitesimal rectangles of constant diagonal, so long as the sum of squares of scale factors is constant (see inset). Since a, b are scale factors going from the surface to the plane, we can express this condition as

$$1/a^2 + 1/b^2 = d^2. \tag{21}$$

Contouring diagonals rather than isolines (equivalently: rotating f by 45°) then yields a Chebyshev net (Figure 30).

For simplicity we let $d = 1$ (since any other d is obtained via global rescaling), and encourage Equation 21 via a corresponding penalty, which is convex in log scale factors:

$$\Phi_{\text{Chebyshev}}(u, v, \theta) := \sum_{ijk \in F} A_{ijk} \left(-2u_{ijk} + \log\left(e^{2v_{ijk}} + e^{-2v_{ijk}}\right)\right)^2,$$

where $u_{ijk} := \frac{1}{3}(u_i + u_j + u_k)$, and likewise for v_{ijk}.

Figure 31 shows that our maps are both smoother and closer to Chebyshev nets than those from the special-purpose algorithm of Oehri et al. [2024]. Like Oehri et al. [2024], and unlike Sageman-Furnas et al. [2019] or Liu et al. [2020], we can generate Chebyshev nets without any singularities—needed for fabricating garments from standard woven fabric (which has no irregular points).

7 Limitations and Future Work

Since Equation 19 is nonconvex, the solver could get stuck at a local minimum or infeasible point. In practice, however, we find our optimization strategy not only succeeds reliably, but produces

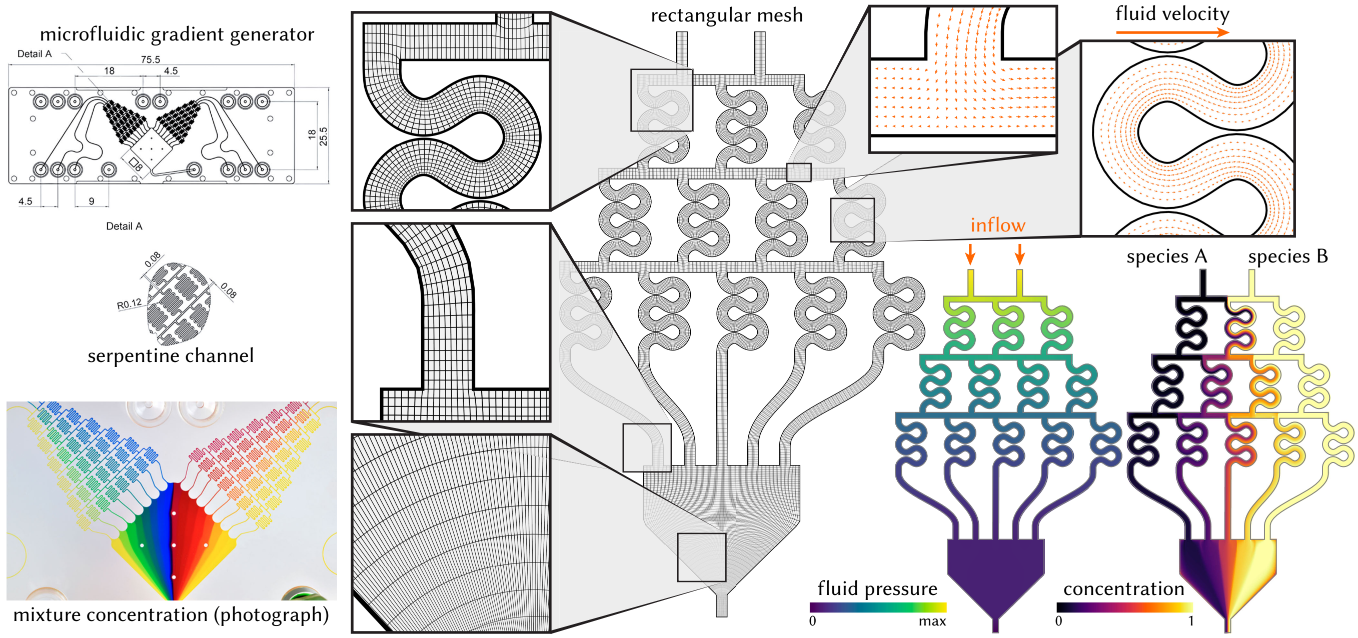


Fig. 29. Our method is suitable for generating high-quality 2D rectangular meshes of domains with complex geometry, as needed for numerical simulation. Here we simulate a microfluidic *gradient generator*, which uses a series of *serpentine mixers* to produce a more uniform concentration gradient of two input species. *Left:* schematic and photograph of a real gradient generator. *Center:* a rectangular mesh automatically generated via RSP captures flow characteristics, without any unwanted singularities. *Right:* simulation on this mesh accurately models the concentration gradient (qualitatively matching the photograph), even with a low-order numerical method.

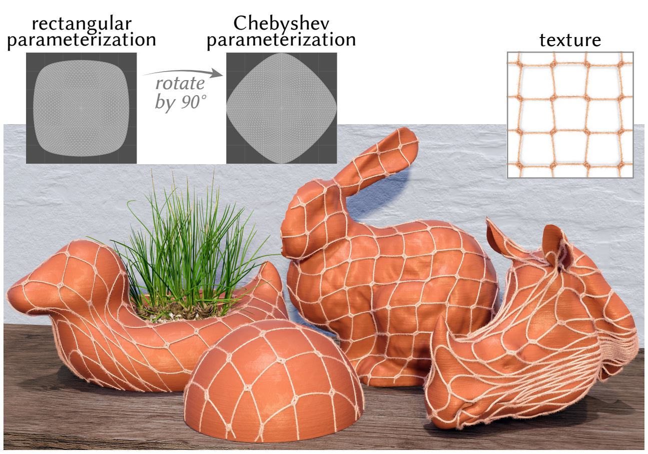


Fig. 30. *Chebyshev nets* are broadly used in architecture, textile design or, as shown here, to design literal *sheet bend* nets that wrap around curved surfaces. Our method can be used to generate high-quality Chebyshev nets by contouring diagonals of a rectangular parameterization rather than its isolines.

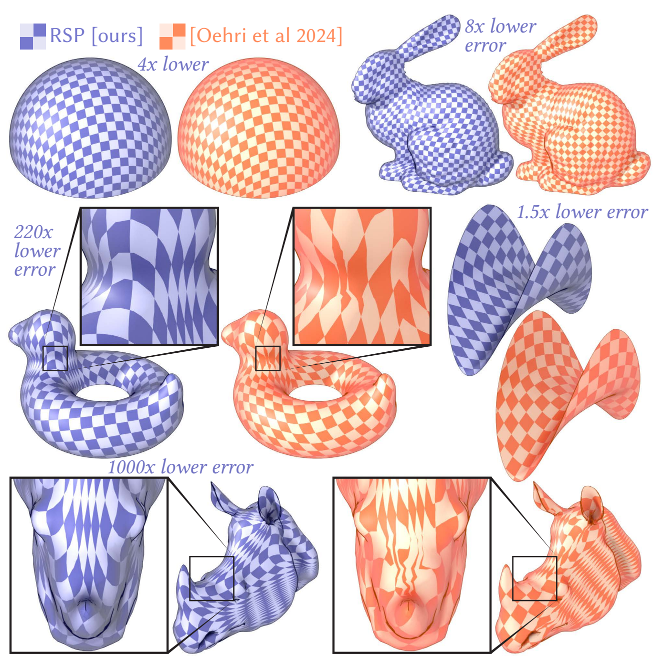


Fig. 31. By simply reinterpreting our rectangular parameterizations, we reliably obtain lower-distortion Chebyshev nets than methods explicitly designed for this purpose. Here we measure error using the metric of Oehri et al. [2024, Equation 12].

the same high-quality minimizer independent of initialization (Section 6.2). This situation is no different from existing field generation techniques, which are largely nonconvex [Viertel and Osting 2019; Sageman-Furnas et al. 2019; Jakob et al. 2015; Diamanti et al. 2015], apart from a few exceptions [Crane et al. 2010; Knöppel et al. 2013]. RSP is also of course more expensive than linear integrability schemes—but well-worth it, given that the real-world engineering bottleneck is typically *human* time spent on tweaking mesh generation and repairing defective meshes.

As discussed in Section 2.1, local injectivity can in principle fail due to discretization error, though rarely happens in practice. If local injectivity is crucial, one might replace our simple Poisson integration with a more robust approach [Myles et al. 2014]. An interesting question is whether there is an exact, *mimetic* discretization of rectangular maps in the spirit of *discrete conformal equivalence* [Gillespie et al. 2021b], which guarantees local injectivity.

Another question is whether existence of a rectangular parameterization implies existence of a *seamless* rectangular parameterization with the same cones/constraints. We have never found a counterexample, and suspect the answer is "yes", but lack a formal proof.

Our automatic cone placement strategy already works quite well (Section 5.3.2), but since singularity placement for *rectangular* parameterization is a new topic, there is of course room for improvement. Here one might build on recent insights from the conformal case [Soliman et al. 2018; Fang et al. 2021b; Li et al. 2022, 2023].

Since our method (like many in this space) discretizes a partial differential equation, its behavior will of course be influenced by the input mesh quality. However, since RSP is purely intrinsic—and we have been careful to express its discretization in intrinsic terms—it can be implemented without modification on *intrinsic triangulations* [Sharp et al. 2021], where *intrinsic Delaunay refinement* can be used to guarantee robustness on essentially arbitrarily bad input meshes [Sharp et al. 2019; Gillespie et al. 2021a].

Finally, our original motivation for this approach was establishing better tools for volumetric *hexahedral* meshing [Pietroni et al. 2022; Fang et al. 2021a]. Since our basic strategy (writing out derivatives of a moving rectangular frame) does not depend inherently on a 2D domain (or holomorphic functions), we are optimistic about its extension to 3D. However, significant open questions remain, *e.g.*, how to move from point-like singularities to 1D curve networks.

Acknowledgments

This work was supported by the French Agence Nationale de la Recherche (ANR) under grant ANR-24-CE48-0895-01 (ORTHOMAP), the National Science Foundation under award 1943123, a Packard Foundation Fellowship, as well as gifts from Adobe and nTopology.

References

Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. 2003. Anisotropic polygonal remeshing. In *ACM SIGGRAPH 2003*.

Douglas Arnold, Daniele Boffi, and Richard Falk. 2002. Approximation by quadrilateral finite elements. *Mathematics of computation* 71, 239 (2002).

Thierry Aubin. 2013. *Some nonlinear problems in Riemannian geometry*. Springer Science & Business Media.

Tristan Carrier Baudouin, Jean-François Remacle, Emilie Marchandise, François Henrotte, and Christophe Geuzaine. 2014. A frontal approach to hex-dominant mesh generation. *Adv. Mod. Sim. Eng. Sci.* 1 (2014), 1–30.

Mirela Ben-Chen, Craig Gotsman, and Guy Bunin. 2008. Conformal flattening by curvature prescription and metric scaling. In *Computer Graphics Forum*, Vol. 27.

David Bommes, Marcel Campen, Hans-Christian Ebke, Pierre Alliez, and Leif Kobbelt. 2013a. Integer-grid maps for reliable quad meshing. *ACM Trans. Graph.* 32, 4 (2013).

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.

David Bommes, Henrik Zimmer, and Leif Kobbelt. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph.* 28, 3 (2009).

Guy Bunin. 2008. A continuum theory for unstructured mesh generation in two dimensions. *Computer Aided Geometric Design* 25, 1 (2008).

Marcel Campen, Hanxiao Shen, Jiaran Zhou, and Denis Zorin. 2019. Seamless parametrization with arbitrary cones for arbitrary genus. *ACM Trans. Graph.* 39, 1 (2019).

Ryan Capouellez and Denis Zorin. 2024. Seamless Parametrization in Penner Coordinates. *ACM Trans. Graph.* 43, 4 (2024).

Élie Cartan et al. 2001. *Riemannian Geometry in an Orthogonal Frame: From Lectures Delivered by lie Cartan at the Sorbonne in 1926-1927.* World Scientific.

Edward Chien, Zohar Levi, and Ofir Weber. 2016. Bounded distortion parametrization in the space of metrics. *ACM Trans. Graph.* 35, 6 (2016).

David Cohen-Steiner and Jean-Marie Morvan. 2003. Restricted delaunay triangulations and normal cycle. In *Symp. Comp. Geom.*

Guillaume Coiffier and Etienne Corman. 2023. The Method of Moving Frames for Surface Global Parametrization. *ACM Trans. Graph.* 42, 5 (2023).

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

Mattéo Couplet, Alexandre Chemin, and Jean-François Remacle. 2024. Integrable Frame Fields using Odeco Tensors. In *Proc. Int. Mesh. Roundtable.* SIAM.

Keenan Crane, Mathieu Desbrun, and Peter Schröder. 2010. Trivial Connections on Discrete Surfaces. *Computer Graphics Forum (SGP)* 29, 5 (2010).

Mathieu Desbrun, Eva Kanso, and Yiying Tong. 2006. Discrete differential forms for computational modeling. In *ACM SIGGRAPH 2006 Courses*.

David Desobry, François Protais, Nicolas Ray, Etienne Corman, and Dmitry Sokolov. 2022. Frame Fields for CAD models. *Lecture Notes in Computer Science* 13018 (Jan. 2022), 421–434. https://inria.hal.science/hal-03537852

Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. Integrable polyvector fields. *ACM Trans. Graph.* 34, 4 (2015).

Alexander Dielen, Isaak Lim, Max Lyon, and Leif Kobbelt. 2021. Learning direction fields for quad mesh generation. In *Computer Graphics Forum*, Vol. 40.

Manfredo P Do Carmo. 2016. *Differential geometry of curves and surfaces: revised and updated second edition.* Courier Dover Publications.

Eduardo D'Azevedo. 1999. On optimal bilinear quadrilateral meshes. *Engineering with Computers* 15 (1999), 219–227.

Hans-Christian Ebke, David Bommes, Marcel Campen, and Leif Kobbelt. 2013. QEx: Robust quad mesh extraction. *ACM Trans. Graph.* 32, 6 (2013).

Qing Fang, Wenqing Ouyang, Mo Li, Ligang Liu, and Xiao-Ming Fu. 2021b. Computing sparse cones with bounded distortion for conformal parameterizations. *ACM Trans. Graph.* 40, 6 (2021).

Xianzhong Fang, Jin Huang, Yiying Tong, and Hujun Bao. 2021a. Metric-driven 3D frame field generation. *IEEE Trans. Vis. and Comp. Graph.* 29, 4 (2021).

Michael S Floater and Kai Hormann. 2005. Surface parameterization: a tutorial and survey. *Advances in multiresolution for geometric modelling* (2005).

Blender Foundation. 2025. Blender: a 3D modelling and rendering package. https://www.blender.org. Version 3.6, accessed January 22, 2025.

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 (2021), 289–318.

Mark Gillespie, Nicholas Sharp, and Keenan Crane. 2021a. Integer Coordinates for Intrinsic Geometry Processing. *ACM Trans. Graph.* 40, 6 (2021).

Mark Gillespie, Boris Springborn, and Keenan Crane. 2021b. Discrete conformal equivalence of polyhedral surfaces. *ACM Trans. Graph.* 40, 4 (2021).

Zekun Hao, David W Romero, Tsung-Yi Lin, and Ming-Yu Liu. 2024. Meshtron: High-Fidelity, Artist-Like 3D Mesh Generation at Scale. *arXiv:2412.09548* (2024).

Aaron Hertzmann and Denis Zorin. 2000. Illustrating smooth surfaces. In *SIGGRAPH*.

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.

Alec Jacobson, Daniele Panozzo, C Schüller, Olga Diamanti, Qingnan Zhou, N Pietroni, et al. 2013. libigl: A simple C++ geometry processing library. (2013).

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

Jovana Jezdimirović, Alexandre Chemin, and Jean-François Remacle. 2023. Integrable Cross-Field Generation Based on Imposed Singularity Configuration—The 2D Manifold Case. In *International Meshing Roundtable*. Springer.

Tengfei Jiang, Xianzhong Fang, Jin Huang, Hujun Bao, Yiying Tong, and Mathieu Desbrun. 2015. Frame field generation through metric customization. *ACM Trans. Graph.* 34, 4 (2015).
Felix Kälberer, Matthias Nieser, and Konrad Polthier. 2007. Quadcover-surface parameterization using branched coverings. In *Comp. Graph. For.*, Vol. 26.
Liliya Kharevych, Boris Springborn, and Peter Schröder. 2006. Discrete Conformal Mappings via Circle Patterns. *ACM Trans. Graph.* 25, 2 (apr 2006), 412–438.
Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Globally optimal direction fields. *ACM Trans. Graph.* 32, 4 (2013).
Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2015. Stripe Patterns on Surfaces. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–11.
Denis Kovacs, Ashish Myles, and Denis Zorin. 2010. Anisotropic quadrangulation. In *Proc. Solid and Phys. Mod.*
Frank Ledoux. 2022. MAMBO. *URL: https://gitlab. com/franck. ledoux/mambo* (2022).
Zohar Levi. 2023a. Seamless Parametrization with Cone and Partial Loop Control. *ACM Trans. Graph.* 42, 5 (2023).
Zohar Levi. 2023b. Shear-reduced seamless parametrization. *Computer Aided Geometric Design* 101 (2023).
Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph.* 21, 3 (2002).
Mo Li, Qing Fang, Wenqing Ouyang, Ligang Liu, and Xiao-Ming Fu. 2022. Computing sparse integer-constrained cones for conformal parameterizations. *ACM Trans. Graph.* 41, 4 (2022).
Mo Li, Qing Fang, Zheng Zhang, Ligang Liu, and Xiao-Ming Fu. 2023. Efficient Cone Singularity Construction for Conformal Parameterizations. *ACM Trans. Graph.* 42, 6 (2023).
Hao-Yu Liu, Zhong-Yuan Liu, Zheng-Yu Zhao, Ligang Liu, and Xiao-Ming Fu. 2020. Practical fabrication of discrete chebyshev nets. In *Computer Graphics Forum*, Vol. 39.
Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J Gortler. 2008. A local/global approach to mesh parameterization. In *Comp. Graph. For.*, Vol. 27.
Feng Luo. 2004. Combinatorial Yamabe flow on surfaces. *Communications in Contemporary Mathematics* 6, 05 (2004), 765–780.
Richard Henri MacNeal. 1949. *The solution of partial differential equations by means of electrical networks.* Ph.D. Dissertation. California Institute of Technology.
Martin Marinov and Leif Kobbelt. 2004. Direct anisotropic quad-dominant remeshing. In *Pac. Conf. Comp. Graph. App.* IEEE.
Dorit Merhof, Roberto Grosso, Udo Tremel, and Günther Greiner. 2007. Anisotropic quadrilateral mesh generation: an indirect approach. *Adv. Eng. Soft.* 38, 11-12 (2007).
Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. Robust field-aligned global parametrization. *ACM Trans. Graph.* 33, 4 (2014).
Ashish Myles and Denis Zorin. 2013. Controlled-distortion constrained global parametrization. *ACM Trans. Graph.* 32, 4 (2013).
Nicolas Nebel and Albert Chern. 2023. Adaptive Surface Meshes from Harmonic Maps. *arXiv e-prints* (2023).
Annika Oehri, Aviv Segall, Jing Ren, and Olga Sorkine-Hornung. 2024. Chebyshev Parameterization for Woven Fabric Modeling. *ACM Trans. Graph.* 43, 6 (2024).
Kwang W Oh, Kangsun Lee, Byungwook Ahn, and Edward P Furlani. 2012. Design of pressure-driven microfluidic networks using electric circuit analogy. *Lab on a Chip* 12, 3 (2012).
Barrett O'Neill. 2006. *Elementary differential geometry.* Elsevier.
Barrett O'Neill. 2014. *The geometry of Kerr black holes.* Courier Corporation.
David Palmer, David Bommes, and Justin Solomon. 2020. Algebraic representations for volumetric frame fields. *ACM Trans. Graph.* 39, 2 (2020).
Davide Pellis and Helmut Pottmann. 2018. Aligning principal stress and curvature directions.. In *Advances in Architectural Geometry.*
Nico Pietroni, Marcel Campen, Alla Sheffer, Gianmarco Cherchi, David Bommes, Xifeng Gao, Riccardo Scateni, Franck Ledoux, Jean Remacle, and Marco Livesu. 2022. Hex-mesh generation and processing: a survey. *ACM Trans. Graph.* 42, 2 (2022).
Nico Pietroni, Stefano Nuvoli, Thomas Alderighi, Paolo Cignoni, Marco Tarini, et al. 2021. Reliable feature-line driven quad-remeshing. *ACM Trans. Graph.* 40, 4 (2021).
Kacper Pluta, Michal Edelstein, Amir Vaxman, and Mirela Ben-Chen. 2021. PH-CPF: planar hexagonal meshing using coordinate power fields. *ACM Trans. Graph.* 40, 4 (2021).
Helmut Pottmann, Andreas Asperl, and Axel Kililan. 2007. *Architectural geometry.* SIAM.
Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2018. Discrete geodesic nets for modeling developable surfaces. *ACM Trans. Graph.* 37, 2 (2018).
Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable locally injective mappings. *ACM Trans. Graph.* 36, 4 (2017).
Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. 2006. Periodic global parameterization. *ACM Trans. Graph.* 25, 4 (2006).
J-F Remacle, François Henrotte, T Carrier-Baudouin, Eric Béchet, E Marchandise, Christophe Geuzaine, and Thibaud Mouton. 2013. A frontal Delaunay quad mesh generator using the L-∞ norm. *Int. J. Num. Meth. Eng.* 94, 5 (2013).
Maxime Rouca. 2019. Exoside Quad Remesher. https://exoside.com/ Automatic quad remeshing software, available as plugins for various 3D software, including Autodesk 3ds Max, Autodesk Maya, Autodesk Fusion 360, Modo, Blender, and Houdini.
Andrew O. Sageman-Furnas, Albert Chern, Mirela Ben-Chen, and Amir Vaxman. 2019. Chebyshev nets from commuting PolyVector fields. *ACM Trans. Graph.* 38, 6 (2019).
Pedro V Sander, John Snyder, Steven J Gortler, and Hugues Hoppe. 2001. Texture mapping progressive meshes. In *SIGGRAPH.*
Rohan Sawhney and Keenan Crane. 2017. Boundary First Flattening. *ACM Trans. Graph.* 37, 1, Article 5 (Dec. 2017), 14 pages.
Rohan Sawhney and Keenan Crane. 2019. boundary-first-flattening. https://github.com/GeometryCollective/boundary-first-flattening. Accessed January 22, 2025.
Nick Sharp and Keenan Crane. 2018. Variational Surface Cutting. *ACM Trans. Graph.* 37, 4 (2018).
Nicholas Sharp, Mark Gillespie, and Keenan Crane. 2021. Geometry Processing with Intrinsic Triangulations. (2021).
Nicholas Sharp, Yousuf Soliman, and Keenan Crane. 2019. Navigating Intrinsic Triangulations. *ACM Trans. Graph.* 38, 4 (2019).
Alla Sheffer, Emil Praun, Kenneth Rose, et al. 2007. Mesh parameterization methods and their applications. *Found. Trends Comp. Graph. Vis.* 2, 2 (2007).
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 Trans. Graph.* 41, 4 (2022).
Tianchang Shen, Zhaoshuo Li, Marc Law, Matan Atzmon, Sanja Fidler, James Lucas, Jun Gao, and Nicholas Sharp. 2024. SpaceMesh: A Continuous Representation for Learning Manifold Surface Meshes. In *SIGGRAPH Asia 2024.*
J Shewchuck. 2002. What is a good linear finite element. *Interpolation, Conditioning, Anisotropy, and Quality Measures* (2002).
Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. 2024. MeshGPT: Generating triangle meshes with decoder-only transformers. In *Proc. Comp. Vis. Pat. Rec.*
Lance Simons and Nina Amenta. 2024. Anisotropy and Cross Fields. In *Computer Graphics Forum*, Vol. 43.
Yousuf Soliman, Dejan Slepčev, and Keenan Crane. 2018. Optimal Cone Singularities for Conformal Flattening. *ACM Trans. Graph.* 37, 4 (2018).
Sanghyun Son, Matheus Gadelha, Yang Zhou, Zexiang Xu, Ming C Lin, and Yi Zhou. 2024. DMesh: A Differentiable Representation for General Meshes. *arXiv e-prints* (2024).
Boris Springborn, Peter Schröder, and Ulrich Pinkall. 2008. Conformal equivalence of triangle meshes. In *ACM SIGGRAPH 2008 papers.*
Marco Tarini, Nico Pietroni, Paolo Cignoni, Daniele Panozzo, and Enrico Puppo. 2010. Practical quad mesh simplification. In *Comp. Graph. For.*, Vol. 29.
Amir Vaxman, Marcel Campen, Olga Diamanti, Daniele Panozzo, David Bommes, Klaus Hildebrandt, and Mirela Ben-Chen. 2016. Directional field synthesis, design, and processing. In *Comp. Graph. For.*, Vol. 35.
Ryan Viertel and Braxton Osting. 2019. An approach to quad meshing based on harmonic cross-valued maps and the Ginzburg–Landau theory. *SIAM Journal on Scientific Computing* 41, 1 (2019).
Muyang Zhang, Jin Huang, Xinguo Liu, and Hujun Bao. 2010. A wave-based anisotropic quadrangulation method. In *ACM SIGGRAPH 2010.*

A Transformation of Connection Forms under Rotation

We here give a proof of Lemma 3.1, showing that the connection form for a frame rotated by an angle θ can be expressed as

$$\omega_{12}^\theta = \omega_{12}^0 - d\theta,$$

where ω_{12}^0 is the connection form for X^0. First note that

$$\begin{aligned} d\sigma_1^\theta \quad &= \quad -\sin(\theta)d\theta \wedge \sigma_1^0 + \cos(\theta)d\sigma_1^0 \\ &\quad\; + \cos(\theta)d\theta \wedge \sigma_2^0 + \sin(\theta)d\sigma_2^0. \end{aligned}$$

Making substitutions $d\sigma_1^0 = -\omega_{12}^0 \wedge \sigma_1^0$ and $d\sigma_2^0 = -\omega_{21}^0 \wedge \sigma_2^0$ from the first structure equation (Equation 1) and collecting terms gives

$$d\sigma_1^\theta = -(\omega_{12}^0 - d\theta) \wedge (-\sin(\theta)\sigma_1^0 + \cos(\theta)\sigma_2^0).$$

From Equation 9, we know that the second factor is equal to σ_2^θ. Hence, we get $d\sigma_1^\theta = -(\omega_{12}^0 - d\theta) \wedge \sigma_2^\theta$. Again invoking the first structure equation $d\sigma_1^\theta = -\omega_{12}^\theta \wedge \sigma_2^\theta$, we must therefore have $\omega_{12}^\theta = \omega_{12}^0 - d\theta$, as desired.