Integrable PolyVector Fields

Olga Diamanti* ETH Zurich
Amir Vaxman Vienna University of Technology
Daniele Panozzo ETH Zurich
Olga Sorkine-Hornung ETH Zurich

Abstract

We present a framework for designing curl-free tangent vector fields on discrete surfaces. Such vector fields are gradients of locally-defined scalar functions, and this property is beneficial for creating surface parameterizations, since the gradients of the parameterization coordinate functions are then exactly aligned with the designed fields. We introduce a novel definition for discrete curl between unordered sets of vectors (PolyVectors), and devise a curl-eliminating continuous optimization that is independent of the matchings between them. Our algorithm naturally places the singularities required to satisfy the user-provided alignment constraints, and our fields are the gradients of an inversion-free parameterization by design.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

Keywords: PolyVectors, curl-free fields, quad meshing

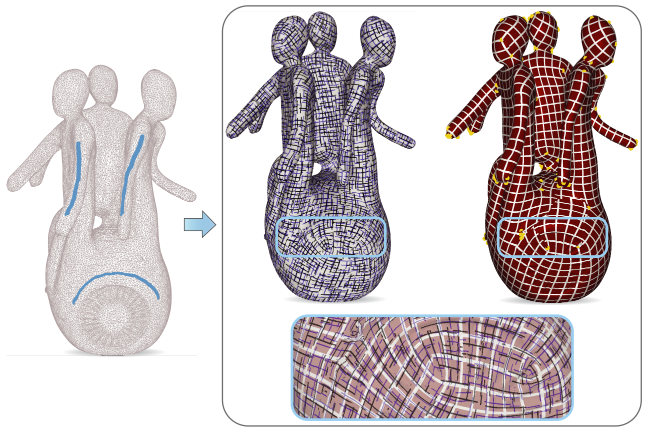


Figure 1: *Parameterization using integrable fields. Given a set of user-provided constraints (left), we optimize for an integrable field, which exactly aligns to the corresponding parameterization (right), while staying close to the constraints.*

1 Introduction

The design of vector fields on discrete surfaces is an important problem in geometry processing, with applications in surface parameterization, texture synthesis, stylized rendering, remeshing and architectural geometry [Hertzmann and Zorin 2000; Lefebvre and Hoppe 2006; Palacios and Zhang 2007; Li et al. 2011; Liu et al. 2011; Bommes et al. 2013b]. In most of these applications, vector fields are computed to serve as a guiding basis for the construction of global parameterizations, which is also the focus of this paper.

Vector fields are typically designed by prescribing a small set of user-defined alignment constraints, which are then interpolated. The resulting field is prescribed as the desired gradient of a parametrization function, and the parameterization is computed by minimizing the difference between the tangent field and the gradient of the function in the least-squares sense, leading to a Poisson equation. In this optimization problem, perfect alignment of the gradient of the resulting parameterization function to the sparse set of constraints can be enforced using linear equality constraints [Kälberer et al. 2007; Bommes et al. 2009]. However, the computation of the function by a least-squares solution may introduce unbounded errors away from the constraints and compromise local injectivity, resulting in inverted elements . Inverted and degenerate elements make the global parametrization unsuitable for practical purposes (e.g., they lead to holes during remeshing [Ebke et al. 2013]). Previous works avoid them using heuristic weighting [Bommes et al. 2009] or relaxation by a reduction of the search space [Bommes et al. 2013a]. All these methods further increase the parametrization distortion.

The key insight that guides our approach is that the gradients of a global parametrization, with or without inversions, are curl-free vector fields. We thus directly devise conditions that guarantee that the final vector field will be curl-free, and that the corresponding parameterization will additionally be free of inverted elements. We propose an algorithm that penalizes the violation of these conditions. Using our algorithm automatically places the singularities in a way that minimizes the distortions.

We formally introduce the notion of integrability and curl for PolyVectors, which are unordered sets of tangent vectors defined on every face of the surface. We develop a formula to measure the PolyVector curl, and optimize for its removal; our formulation does not require explicit pairings of vectors between the unordered sets on adjacent elements. In this manner, the individual vector pairings (often called matchings) are free to change during the optimization, naturally inducing field singularities. Our algorithm relies on a nonlinear optimization, involving no integer variables, and supports user-provided directional constraints, which may be free to align with either the u or the v coordinate of the parameterization, without the necessity to constrain such a choice.

The conditions that ensure an inversion-free parameterization are not imposed as hard constraints by our algorithm, but as a penalization term that tends to avoid invalid configurations. This term is then minimized using a nonlinear and nonconvex optimization. Thus our algorithm is theoretically not guaranteed to find a global minimum, i.e., a fully inversion-free parameterization. However, we demonstrate its robustness and applicability by running it on a global parameterization benchmark proposed in [Myles et al. 2014], which contains 116 models. Our algorithm successfully produces integrable fields on all models, and the induced parameterizations contain no inversions.

*e-mail:olga.diamanti@inf.ethz.ch

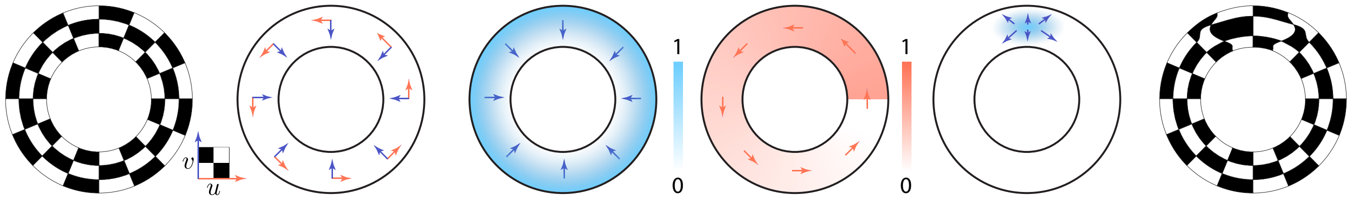


Figure 2: *Our algorithm optimizes for general curl-free fields. From left to right: a parameterization, its gradients, the scalar function corresponding to v (blue), the periodic function corresponding to u (orange), a scalar function whose gradient is added to the original gradients of the parameterization, the resulting parameterization.*

2 Related work

The design of tangent vector fields on surfaces has been pioneered in [Zhang et al. 2006; Fisher et al. 2007]. These methods compute smooth fields that satisfy alignment constraints, and can also generate divergence-free or curl-free fields. A recent method by Azencot *et al.* [2013], which is based on a functional representation, allows joint design of fields on multiple surfaces, in addition to the design of symmetric fields.

Cross fields. Cross fields were initially motivated by nonphotorealistic cross-hatching rendering, where a set of four orthogonal directions is necessary at each point. A cross field is a set of four unit-length vectors related by a rotation of $\pi/2$ radians. They have been formally defined and generalized to a wider set of symmetries, e.g. N-RoSy, in [Kälberer et al. 2007; Palacios and Zhang 2007; Ray et al. 2008].

A cross field can be created by specifying its singularities [Ray et al. 2008; Crane et al. 2010] or by prescribing a sparse set of directional constraints and letting the topology emerge from a smoothing process [Ray et al. 2009]. A smoothness measure on cross fields is defined as the relative angle difference between the two vectors of adjacent crosses that have the smallest relative angle difference. Such smoothness can be equivalently defined as the angle difference between *any* two vectors of adjacent crosses up to an integer in-plane rotation by $\pi/2$ [Bommes et al. 2009], or by factoring out the symmetry using a complex representation and complex powers to encode it [Knöppel et al. 2013]. The design of cross fields on symmetric surfaces has been studied in [Panozzo et al. 2012].

Non-orthogonal and non-homogeneous N-RoSy fields. N-RoSy fields do not encode scale and anisotropy, and the angles between their vectors are fixed to $2\pi/N$. To lift the angle restriction and generalize integer matching, permutation matrices have been introduced in [Liu et al. 2011] and employed to represent conjugate fields, which can be converted into planar quadrilateral meshes. Frame fields, which are pairs of line fields with independent scales and angle, can be encoded by a composition of cross fields and affine maps [Panozzo et al. 2014]; such frame fields can be used for anisotropic quadrangulation with varying element size.

PolyVector fields. A general formulation, which includes all previously mentioned fields as special cases, has been proposed in [Diamanti et al. 2014]. PolyVectors are general unordered vector sets, encoded as the roots of a complex polynomial. The coefficients of the polynomial can be interpolated over a surface by solving a linear system, indirectly interpolating the polynomial roots. Additional constraints, such as angle bounds or conjugacy, can be easily enforced with a local-global optimization. Our algorithm enhances the PolyVector framework by introducing a new set of properties that guarantees the integrability of the fields, while maintaining the continuous and general formulation.

Integrability. To the best of our knowledge, the approach of Ray *et al.* [2006] is the only method that explicitly attempts to improve the integrability of a vector field prior to using it in the computation of a field-aligned global parameterization. Ray *et al.* propose to rescale the vectors to minimize the curl as a heuristic that improves the integrability, but rescaling alone is too restrictive to guarantee a truly integrable field on surfaces which are not homeomorphic to a disk. Apart from [Ray et al. 2006], all other field-design methods focus on creating *smooth* fields that, in the majority of the applications, are used as ideal (target) gradient fields for a global parameterization. Smoothness is correlated with integrability, as we discuss in Section 3, albeit not guaranteeing it. Therefore, the resulting parameterizations are not precisely aligned with the desired field and may even have inversions. [Myles et al. 2014] compute a global parameterization by tracing the separatrices of a given field and then parameterizing each patch of the induced partitioning. Heuristic steps are used to reduce alignment artifacts due the non-integrability of the field, which can lead to arbitrary misalignments in the resulting parameterization. We provide a more direct and simple algorithm that controls the alignment, while avoiding numerical tracing and heavy post-processing.

Integrable poly-vector field topology. A global parametrization is a map from a surface with a non-flat metric to the plane, inducing distortion for any non-developable surface. In order to control the distortion and generate a locally-injective map, cone singularities must be introduced. Singularities are heuristically computed for a smooth field in [Bommes et al. 2009], or optimized for directly on the parameterization function in [Myles and Zorin 2012; Myles and Zorin 2013]. [Lipman 2012; Bommes et al. 2013a; Ray et al. 2006] guarantee the generation of a locally-injective map by fixing the field topology, at the price of limiting the solution space. In our case, the singularities are automatically introduced by the algorithm where needed.

Quadrangulation. Tangent fields can be used directly to create quad-dominant meshes, without computing an intermediate parameterization, by tracing their separatrices [Alliez et al. 2003; Marinov and Kobbelt 2004]. The resulting quad-dominant meshes can be converted to pure quadrilateral meshes with a step of Catmull-Clark subdivision. However, such methods can handle singularities only by subdivision, without any control of quad quality near the singularities. In light of this fact, quad meshes are usually computed using a parameterization; the parametric domain is then regularly meshed by a grid and lifted back to the surface [Kälberer et al. 2007; Bommes et al. 2009; Bommes et al. 2013a; Ebke et al. 2013; Ebke et al. 2014]. The main difficulty in creating a quadrangulation over a general parameterization is that the parameterization must be *seamless*, i.e., close up on parameterization seams with perfect alignment and grid translation, in order for the quadrilateral mesh to be consistent. The gradients of such parameterizations are by definition curl-free everywhere, including across seams, and in addition integrate into integers around singularities for grid consistency. Seams can be avoided by assuming a rotational symmetry between the gradients

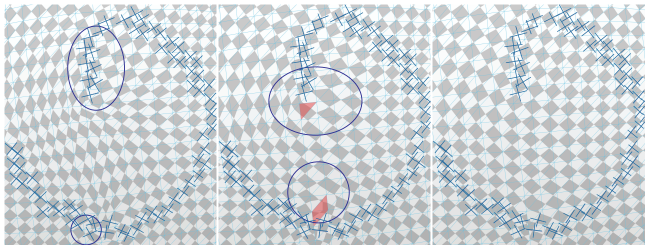


Figure 3: *Solving a Poisson equation can result in arbitrary misalignments between the input field and the gradients of the reconstructed scalar function (left). Alignment can be enforced using harder constraints (middle), potentially leading to inverted elements (in red) and/or higher distortion. Our method optimizes for a field that avoids inversions in the parameterization and adheres well to the constraints.*

of the parameterization functions, by using a periodic trigonometric representation [Ray et al. 2006]. However, this only applies when the goal is a conformal parameterization represented by 4-RoSy fields, and cannot be easily generalized to frame fields.

A completely different approach to mesh quadrangulation consists in designing scalar functions whose Morse-Smale complex is a quadrilateral grid on the surface [Dong et al. 2006; Zhang et al. 2010; Ling et al. 2014]. Unfortunately, aligning to constraints is difficult to achieve in this setting. To the best of our knowledge, the only methods that guarantee perfect alignment of the mesh edges to (arbitrarily many) user-defined constraints require extensive manual input, e.g. in the form of sketches [Takayama et al. 2013; Campen and Kobbelt 2014]. If our fields are used to compute a seamless parameterization for quadrangulation, we can guarantee that all edges of the quadrangulation correspond to the designed field and continuously align everywhere, up to the integer roundings. As we demonstrate in Section 6, the integer grid misalignment becomes smaller as we increase the resolution of the output mesh.

3 Gradients of scalar functions and field-aligned parameterizations

We address the design of tangent vector fields for the purpose of surface parameterization, i.e., fields that are the gradients of the scalar coordinate functions of the parameterization. The isolines of the parameterization are then orthogonal to these vector fields by design. The fields should satisfy the user-provided alignment constraints if they are not contradictory; otherwise, comply to them as much as possible. Ideally, a field-based parameterization design algorithm should provide as much freedom as possible in the choice of imposed alignment constraints, and at the same time always compute a valid parameterization. Finally, the resulting parameterization should introduce minimal metric distortion and avoid inversions.

To achieve these goals in the context of vector-field design, we first need to study how these parameterization properties relate to the properties of vector fields. We discuss multiple vertex-based scalar functions on discrete surfaces and the properties of their gradients as sets of integrable vector fields. We then focus on special sets, defined per surface point, that consist of four ordered vectors and are of particular interest for inversion-free parameterizations.

3.1 Vertex-based scalar functions

We denote the input discrete surface mesh by $\mathcal{S} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$, where $\mathcal{V}, \mathcal{E}, \mathcal{F}$ denote the set of vertices, edges and triangle faces, respectively. Discrete scalar functions $h : \mathcal{S} \to \mathbb{R}$ are defined by their values on the mesh vertices; their values inside a triangle are obtained by linearly interpolating the function values at the triangle vertices, using hat functions.

A *tangent vector field* assigns a vector α_f to each face $f \in \mathcal{F}$; α_f lies on the plane spanned by the face. The gradients ∇h of discrete scalar functions are a particular case; since the functions are piecewise linear, their gradients are piecewise constant, i.e., constant on every face $f \in \mathcal{F}$.

Discrete curl. The linearly-interpolated values of a discrete scalar function h on two adjacent faces $f, g \in \mathcal{F}$ coincide on their mutual edge e. The derivatives of h are well-defined inside triangles f and g, since h is linear there, and the directional derivatives along edge e coincide. This can be expressed using the per-face gradients $\nabla h_f, \nabla h_g$ and the edge vector e as:

$$\langle \nabla h_f, e\rangle = \langle \nabla h_g, e\rangle.$$

It trivially follows that $\langle \nabla h_f, e\rangle - \langle \nabla h_g, e\rangle$ is zero for any function h. Replacing $\nabla h_f, \nabla h_g$ with general tangent vectors α_f, α_g, we obtain $\langle \alpha_f, e\rangle - \langle \alpha_g, e\rangle$, which is a well-established definition of the *discrete curl* of the vector field α on the edge [Polthier and Preuß 2003; Kälberer et al. 2007]. A vector field α is hence curl-free across edge e if and only if:

$$\langle \alpha_f, e\rangle = \langle \alpha_g, e\rangle. \quad (1)$$

Similarly to the continuous case, the gradients of a discrete scalar function have zero curl. However, the converse is true only with an important restriction: a curl-free vector field α in a *simply-connected* surface patch $\mathcal{P} \subset \mathcal{S}$ is always the gradient of some scalar function h defined within the patch $\mathcal{P}$. If the entire mesh $\mathcal{S}$ is simply-connected, then the scalar function h can be defined globally on all vertices.

Poisson integration. Given a tangent vector field α on a simply-connected surface, which is also the gradient of a scalar function h, we can reproduce h up to an additive factor τ by solving a linear system, using the discrete gradient operator $G : \mathcal{V} \to \mathcal{F}$ [Botsch et al. 2010]:

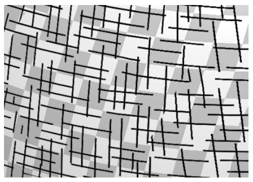
$$Gh = \alpha. \quad (2)$$

Since $|\mathcal{F}| \geq |\mathcal{V}|$, this system is overdetermined. For a general field α, this system has a solution only if α is curl-free (which trivially holds if α is the gradient of a scalar function). If α is not curl-free, or if the surface is not simply-connected, we can solve this system in the least-squares sense, i.e. the Poisson equation:

$$\min_h \|Gh - \alpha\|_2^2 \;\; \Rightarrow \;\; \Delta h = \operatorname{div} \alpha, \quad (3)$$

where Δ is a discrete Laplace operator. This system always has a solution. The gradient of the resulting scalar function is a projection of α onto the space of gradient vector fields, and, as such, onto the space of curl-free vector fields as well.

Solving the above Poisson problem to produce a scalar function from a vector field is numerically inexpensive, as it reduces to



solving a sparse linear system. It is thus commonly used in many geometry processing algorithms, where a tangent field is designed as a candidate in lieu of the exact gradient of a field-aligned parameterization. Unfortunately, the actual gradient of the Poisson-integrated function might have arbitrarily large deviations in alignment and sizing from the originally designed field (see inset, and Figure 3). Introducing a weighting scheme to enforce better alignment of the gradients in the constrained parts often works, but might introduce an unpredictable and uncontrolled deviation of the gradient of the parameterization from the tangent vector field in unconstrained regions, unless the tangent vector field α is curl-free by intended design.

In light of this fact, designing curl-free vector fields for the purpose of constructing scalar functions is an attractive approach, as it ensures that the resulting field is exactly the gradient of the desired scalar function, without introducing alignment or sizing errors.

Harmonic fields. Curl-free vector fields are synonymous with gradients of scalar functions only on simply-connected patches. For non-simply-connected domains, there exist curl-free fields which are not gradients of any globally-defined continuous function, but which play an important role in our framework. Consider the annulus in Figure 2: the gradients of the two parameterization coordinate functions are both curl-free, yet only ∇v (blue) is the gradient of a continuous scalar function. There is no continuous scalar function whose gradient corresponds to the field that circles the annulus (orange). However, it is the gradient of a periodic function (Figure 2, middle), which can be converted into a continuous scalar function by cutting its domain until it is simply connected. Such vector fields are called *harmonic* fields, and they are vital for creating periodic functions on surfaces: they are the part of the vector field which "closes up" the parameterization around any chosen seam, creating the seamless effect. The gradient of any scalar function can be added to the harmonic field without breaking this effect, while providing additional degrees of freedom for the parameterization (Figure 2, right). Our approach designs general curl-free fields, consisting of harmonic fields and gradients of scalar functions, which are then the proper fields for designing seamless parameterizations.

3.2 Multiple scalar parameterization functions

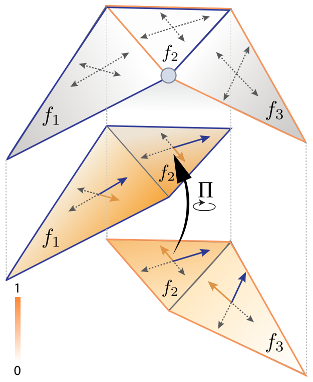
A mesh parameterization is a pair of scalar functions $(u, v) : \mathcal{V} \to \mathbb{R}^2$ representing the mapping between surface points and points on a plane. This is a special case of multiple scalar functions defined on a single surface; their gradients are a collection of vectors on every face $f \in \mathcal{F}$, which we call a *vector-set field*.

Matchings and curl-free sets. Let us consider two adjacent faces f, g with their respective vector-sets. We define a *matching* as a bijective pairing of the individual vectors in the vector-set of f with those of g. If every pair of matched vectors has zero curl on the edge shared between f, g, we denote the pairing as a *curl-free matching*. If for every two adjacent faces on the mesh we have a curl-free matching, the entire vector-set field is *curl-free*.

Inconsistencies and singularities. Consider an ordered 1-ring of consecutive faces $\{f_1, \ldots, f_m\}$ around a vertex j and assume there is a matching defined for all edges. Pick a single vector in face f_1, follow into face f_2 by picking its paired vector, and continue until face f_1 is visited again. If we land on the same vector we started with, the matching is *consistent* around vertex j, which is then called a *regular vertex*. Otherwise, the matching is *inconsistent* and vertex j is a *singularity*.

Separable scalar functions. Suppose we have a vector-set field on a simply-connected surface patch $\mathcal{P} \subset \mathcal{S}$ that does not contain any singularity. In this case, the vector-set field can be decomposed into a collection of tangent vector fields, each field containing one vector per face: starting from one vector on a face, we follow the matchings (on edges) to the adjacent faces, until the entire patch is visited. We then do the same for any remaining vector, until the entire field is claimed. If the matchings are curl-free, we can integrate each tangent vector field into a separate scalar function; thus, the vector-set field represents a set of scalar functions, such as the u and v coordinate functions of a parameterization.

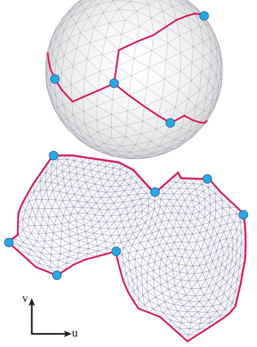
Singularities and overlapping patches. The global decomposition into separate tangent vector fields cannot be done if the vector-set field on a patch contains singularities, and thus we cannot integrate it into separate scalar functions. However, for any regular point, we can find a neighborhood with no singularities in its interior (the neighborhood may still have singularities on its boundary). In such a neighborhood, we can locally integrate the vector-set field to obtain scalar functions. Next, consider two overlapping neighborhoods, where the vector sets are integrated into scalar functions in each neighborhood. The integrated functions in the region shared by both neighborhoods are related by permutations and additive factors, as we show next with an example. Consider two simple overlapping patches (inset): $\mathcal{P}_1 = (f_1, f_2), \mathcal{P}_2 = (f_2, f_3)$, where f_1, f_2, f_3 are consecutive faces around the 1-ring of a singularity vertex. Since these patches do not contain the singularity in their interior (it is on their mutual boundary), we can decompose and integrate the vector-set field of $\mathcal{P}_1$ into separate scalar functions $A_1 \ldots A_n$, and the vector-set field of $\mathcal{P}_2$ into functions $B_1 \ldots B_n$ (we only show the function corresponding to the orange vectors in the figure). Since both scalar function sets $\mathcal{A}$ and $\mathcal{B}$ share the same vector-set α_2 on face f_2, they basically have the same gradients there, but up to order. Therefore, the elements in the sets $\mathcal{A}$ and $\mathcal{B}$ are related by additive factors (due to the integration) and a permutation:



$$\left.\begin{pmatrix} A_1 \\ \vdots \\ A_n \end{pmatrix}\right|_{f_2} = \Pi \left.\begin{pmatrix} B_1 \\ \vdots \\ B_n \end{pmatrix}\right|_{f_2} + \begin{pmatrix} \tau_1 \\ \vdots \\ \tau_n \end{pmatrix}, \tag{4}$$

where $\Pi : \mathcal{A}|f_2 \to \mathcal{B}|f_2$ is a permutation operator, and $\tau_i \in \mathbb{R}$ are the additive factors (in the inset, the orange vector on the left flap is permuted with the opposite of the orange vector on the right flap). Thus, curl-free vector-set fields can be integrated with zero error into separate scalar functions on overlapping patches. The functions on the different patches are *interchangeable* in the shared region with the ones on another patch: their relation is completely defined by permutations and additive factors . This is particularly appealing for parameterization applications that are invariant to certain permutations: for example, for the purpose of quadrangulating a mesh via global parameterization, the exact assignment of u and v directions is not important, as they are equivalent up to rotation and translation by certain factors. Intuitively, rotating a regular quadrilateral grid image by $\pi/2$ radians (permuting coordinates), or translating it by a constant offset (additive factor) produces an identical image, and a similar invariance holds when the grid is lifted onto the surface.

Seams. The concept of interchangeable functions is well-established in computer graphics, and is usually formulated in a different, but equivalent, way. Instead of defining scalar functions on overlapping neighborhoods, it is possible to equivalently cut the mesh into a single, disk-topology patch with all singularities on its boundary (inset). The boundary segments of this patch are commonly referred to as *seams*, and the entire boundary is a tree of mesh edges connecting all singularities (as well as cut handles



for surfaces with a non-vanishing genus). Since there are no singularities in the interior of this patch (which covers the whole mesh), and since the vector-set field is curl-free, we can always separate it into individual tangent vector fields, and subsequently integrate those into globally-consistent scalar functions, defined on the vertices of this patch. Note that these vertices correspond to the mesh vertices only in the patch interior; the mesh vertices on the seams appear multiple times as patch corners. Thus, the integration returns multiple scalar values for the mesh vertices on the seams, which are related by a permutation and an additive factor, as discussed earlier.

Seamless quadrangulations and frame fields. For the purpose of quadrangulation, a square grid texturing the plane is lifted back onto the original surface using the inverse of the parameterization. It is desirable that the (inverse) parameterization function embeds the planar quad grid onto the surface in a *seamless* manner, i.e., the grid appears regular everywhere. However, only topological disks can be parameterized without introducing seams; if seams are present, they might be visible following this lifting.

Note that the appearance of a quadrilateral grid is invariant to both rotation by $\pi/2$ and integer translation. As long as the planar image of the parameterization on the seams only transforms by such rotations and translations, the seams will not be visible. In terms of parameterization functions, this amounts to representing the parameterization with four (dependent) functions $\{u, v, -u, -v\}$, each representing the four (signed) alignments of a quad mesh. These functions are related across seams by a permutation and/or translation by an integer multiple of grid edge length, producing an identically looking grid on the surface. Note that the gradients of these functions are not generally related by rotational symmetries unless the parameterization is conformal. Seamless quadrangulations can then be computed in three steps:

1. Construct a curl-free vector-set field $\{\alpha, \beta, -\alpha, -\beta\}$ (a curl-free frame field).
2. Cut the mesh into a disk so that the singularities are on its boundary, and integrate the field to obtain the scalar functions.
3. Make sure the additive factors on the cuts are integer-valued.

We tackle the first two steps in our work: if the frame field is curl-free, then we can compute scalar parameterization functions whose gradient is exactly aligned with the field. On the cuts, the scalar functions agree up to permutations, which corresponds to the invariance to permutations of the integer grid. Unfortunately, the curl-free property does not guarantee that the additive factors would be integer-valued; this has to be taken into account in a separate design. However, we demonstrate that while curl-free fields only guarantee seamless integration up to integer consistency, a low integration error is achieved after the final integer rounding step.

To conclude, the design of multiple scalar functions is reducible to the design of curl-free vector-set fields. If the singularities are prescribed in advance, the problem can be reduced (by cutting the mesh and separating the vector-set into single vector fields) into the problem of designing curl-free vector fields with the given singularities. However, this is not a good practice in general, since positioning singularities and determining matchings a priori considerably shrinks the available search space, potentially leading to suboptimal solutions. Instead, the entire set of possible local matchings can be encoded with integer rotation variables [Bommes et al. 2009] or permutations [Liu et al. 2011], and an extra curl condition can be incorporated per matching. Unfortunately, the introduction of combinatorial choices increases the computational complexity and, again, might lead to suboptimal results.

Smooth vs. curl-free fields Alternatively, it is a common practice to ignore the curl-free constraints and design smooth vector-set fields, either symmetric [Knöppel et al. 2013] or general [Diamanti et al. 2014]. Two vector sets are perfectly smooth across an edge if there is a matching between them, so that every matched pair is parallel. Parallelity is measured by flattening the related triangle flap to the plane. It is clear that the curl of the sets of the edge is zero. Perfect smoothness cannot be attained in general, due to curvature, and it is then common to compute "as-smooth-as-possible" vector-set fields [Crane et al. 2010; Knöppel et al. 2013; Diamanti et al. 2014]. However, the resulting fields and the smooth matchings (the ones that match the most parallel vectors) are only approximately curl-free matchings at best. While such a solution avoids the difficult curl-free constraints for vector sets altogether, it is susceptible to unexpected errors and misalignments in the integrated scalar functions after the Poisson step (Figure 4).

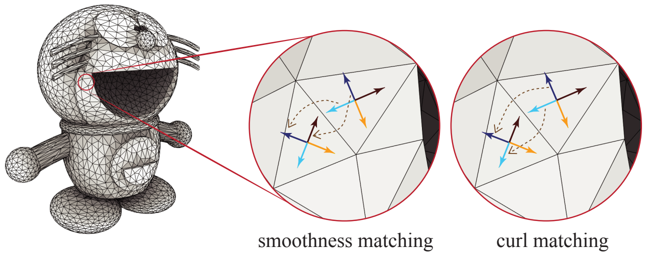


Figure 4: *Smooth matchings might not be curl-free matchings, inducing errors and misalignments in the integrated scalar function.*

3.3 Inversion-free parameterizations

In addition to the "seamlessness" property discussed earlier, a basic requirement for parameterization functions (or mappings) is to be inversion-free. A mapping that does not satisfy this property results in inverted faces in the parametric domain, and thus cannot be used for practical purposes, such as texture mapping or surface quadrangulation.

An equivalent requirement is that the cross product of ∇u and ∇v on the surface always points in the same direction as the surface normal. It follows that the angle between ∇u and ∇v must be convex (between 0 and π). This condition can be directly imposed on the gradients of u and v, when put as row vectors, as follows:

$$\det\left(\nabla u^{\top}, \nabla v^{\top}\right) > 0. \tag{5}$$

The mapping induced by such u and v is then inversion-free.

If u and v are obtained via Poisson integration from two general (not curl-free) tangent vector fields, this condition might not be satisfied: u and v are integrated separately, while the injectivity property is a joint condition. Methods that guarantee an inversion-free parameterization thus either couple the optimization of the two functions u and v in a nonlinear term [Schüller et al. 2013] or solve a sequence of convex subproblems [Lipman 2012; Bommes et al. 2013a], ensuring that the parameterization is free of inversions.

If a curl-free frame field is used to design a quadrangulation, Eq. (5) readily translates into a condition on the matchings: a curl-free frame field induces an inversion-free parameterization if and only if its curl-free matching is order-preserving, meaning that the geometric order of the vectors on two adjacent faces is identical after the matching (see Figure 5).

In summary, to design an inversion-free parameterization, it suffices to produce a vector-set field with the following properties:

- The vector-set field should satisfy the user-provided constraints and be as smooth as possible.

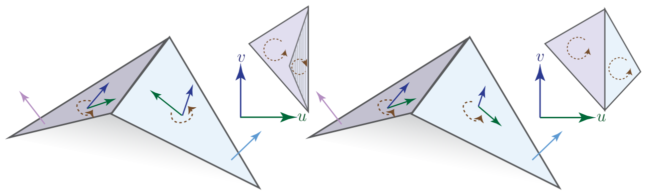


Figure 5: *A curl-free frame field induces an inversion-free parameterization if and only if its curl-free matching is order-preserving.*

- The vector-set field should be curl-free, to ensure exact alignment of the parameterization to the designed field.
- The curl-free matchings must be order-preserving, to ensure local injectivity of the parameterization.
- The vectors in every face should be properly oriented (see Equation 15 below).

In the next section, we formulate a framework to solve for vector-set fields which precisely satisfy these properties; our method performs a continuous optimization that incorporates all possible matchings, and thus naturally induces singularities.

4 Integrable PolyVector fields

Equipped with the formal properties necessary for vector-sets to be the gradients of the coordinate functions of an inversion-free seamless parameterization, we next present the mathematical elements used in our algorithm for designing such fields. To represent vector-sets, we employ PolyVectors [Diamanti et al. 2014]. We briefly summarize the basic concepts and properties of PolyVectors to make this paper self-contained.

Vector fields as complex numbers. We define a local coordinate system for every face, which we identify with the complex plane and use to represent tangent vectors as complex numbers. We use a *discrete connection* to compare two vectors $\alpha_f, \alpha_g \in \mathbb{C}$ that are defined in adjacent faces f, g with a mutual edge e; this amounts to translating one coordinate system onto another by using the normalized edge vector as a common denominator. This formulation is equivalent to isometrically unfolding the triangle flap f, g onto a plane, and then comparing vectors directly after the alignment of their local coordinate systems. The vectors α_f and α_g are said to be equal if and only if:

$$\alpha_f \overline{e_f} = \alpha_g \overline{e_g}, \tag{6}$$

where $\overline{e_f}$ and $\overline{e_g}$ are the complex conjugates of the representations of the edge vector e in the local complex coordinate systems of faces f and g, respectively. This is a discrete *Levi-Civita* (LC) connection [Crane et al. 2010]. The smoothness of a vector field can then be measured by the following Dirichlet energy, which penalizes deviation from the LC connection:

$$E_D(\alpha) = \sum_{(f,g)\sim e \in \mathcal{E}} \|\alpha_f \overline{e_f} - \alpha_g \overline{e_g}\|^2. \tag{7}$$

We define a *smooth* vector field to be a vector field that minimizes this energy, thus being "as-LC-as-possible" [Diamanti et al. 2014].

PolyVector fields. A PolyVector [Diamanti et al. 2014] encodes an unordered set of n tangent vectors $\{\alpha_{1,f}, \cdots, \alpha_{n,f}\}$ on a face f. The vectors are considered as the roots of a unique monic complex polynomial, and thus the unordered set is encoded by its complex coefficients. Partial symmetry of vectors within the set, including N-RoSy fields [Knöppel et al. 2013], can be encoded by the PolyVector representation, producing special cases of polynomials. For instance, an n-RoSy field is encoded as $z^n - a^n$, where a is an arbitrary single vector from the n-RoSy set.

The smoothness of PolyVectors can then be reduced to the smoothness of their polynomial coefficients (as measured by the Dirichlet energy in Eq. (7)), and these coefficients are compared using the proper LC-connection [Diamanti et al. 2014].

4.1 Curl-free PolyVector fields

We define an n-PolyVector field to be curl-free if and only if for each edge e between two faces f, g, there is a matching such that every matched vector-pair is curl-free per the definition in Eq. (1). Given the complex representation of the field vectors, the condition for zero curl across the edge now becomes

$$\mathrm{Re}(\alpha_f \overline{e_f}) = \mathrm{Re}(\alpha_g \overline{e_g}). \tag{8}$$

PolyCurl. Using the PolyVector framework, we can write an optimization that factors out the matchings, allowing us to optimize for curl-free fields by minimizing a continuous energy. Assume again two adjacent faces f, g sharing edge e, and an n-PolyVector field α consisting of the tangent vectors $\{\alpha_{1,f}, \cdots \alpha_{n,f}\}$ on face f and $\{\alpha_{1,g} \cdots \alpha_{n,g}\}$ on face g. We define the following two real *curl polynomials* (where $r \in \mathbb{R}$):

$$\begin{aligned} P_{f|e}(r) &= (r - \mathrm{Re}(\alpha_{1,f}\,\overline{e_f})) \cdots (r - \mathrm{Re}(\alpha_{n,f}\,\overline{e_f}))\,, \\ P_{g|e}(r) &= (r - \mathrm{Re}(\alpha_{1,g}\,\overline{e_g})) \cdots (r - \mathrm{Re}(\alpha_{n,g}\,\overline{e_g}))\,. \end{aligned} \tag{9}$$

Lemma 4.1. *There is a curl-free matching of vectors between the vector sets in f and g if and only if $P_{f|e}$ is equivalent to $P_{g|e}$.*

Proof. Two polynomials are equivalent if and only if they have matching roots, which means that there is an induced matching of vectors with the same dot product with the edge-vector e, and thus the set is curl-free by definition (Eq. (1)). □

Intuitively, the "distance" between these two polynomials, represented as the distance between their respective coefficient vectors $(c_{f|e,0},\ c_{f|e,1}, ...)$ and $(c_{g|e,0},\ c_{g|e,1}, ...)$, is measuring the edge curl while being independent of the matching:

$$\mathrm{PolyCurl}_e = \sum_k \left(c_{f|e,k} - c_{g|e,k}\right)^2. \tag{10}$$

We call this quantity *PolyCurl*: if it is zero, then there must be at least one zero-curl matching for all the vectors in the two adjacent n-PolyVectors. Note that the real-valued PolyCurl does not use the LC-connection to transport vectors; each polynomial $P_{f|e}$ is constructed using the edge e represented in the reference system of f, and similarly for $P_{g|e}$.

Frame fields. Since our main application is parameterization for the purpose of quadrangulation, we now restrict the rest of the paper to special 4-PolyVector fields called *frame fields*. Note, however, that all of the following concepts can be readily extended to general n-PolyVectors.

Frame fields are 4-PolyVector fields comprising two symmetric line fields, each with its own scale, and they are ideal for representing gradients of a global parameterization (Section 3.2). For each face $f \in \mathcal{F}$, a frame field consists of an unordered vector set $\{\alpha_f, \beta_f, -\alpha_f, -\beta_f\}$, $\alpha_f, \beta_f \in \mathbb{C}$. Therefore, the polynomial representing an element of a frame field is $P(z) = (z^2 - \alpha_f^2)(z^2 - \beta_f^2)$ and the corresponding 4-PolyVector is $\left(0, -(\alpha_f^2 + \beta_f^2), 0, \alpha_f^2\beta_f^2\right)$.

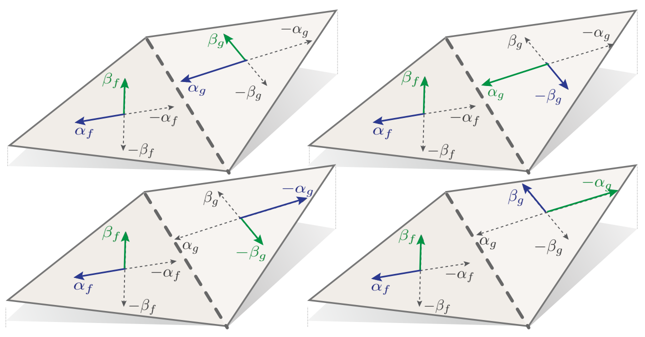


Figure 6: *For frame fields, only four possible matchings per edge are order-preserving. The curl quotients for these four cases are related by equality or reciprocal negation.*

For frame fields, the non-vanishing coefficients of the monic curl polynomial for face f (Eq. (9)) correspond to the free and the second-degree coefficients (note that the curl polynomial is in fact quartic):

$$\begin{aligned} c_{f|e,0} &= \mathrm{Re}(\alpha_f \overline{e_f})^2 \mathrm{Re}(\beta_f \overline{e_f})^2, \\ c_{f|e,2} &= -\left(\mathrm{Re}(\alpha_f \overline{e_f})^2 + \mathrm{Re}(\beta_f \overline{e_f})^2\right). \end{aligned} \tag{11}$$

When $c_{f|e,0} = c_{g|e,0}$ and $c_{f|e,2} = c_{g|e,2}$, the frame field on faces f, g is curl-free. Hence we denote the distance $(c_{f|e,0} - c_{g|e,0})^2 + (c_{f|e,2} - c_{g|e,2})^2$ as the PolyCurl of the frame field.

The disadvantage of trying to minimize the PolyCurl to compute curl-free fields is paradoxically caused by its advantage: the formulation is indifferent to the ordering of the vectors, and thus can easily produce a curl-free field whose curl-matching is not order-preserving, a property that we require for inversion-free parameterizations. In the following, we formulate an additional measure called *PolyQuotient* that, when minimized, prevents matchings that are not order-preserving, and thus guarantees that there are no inverted triangles in the corresponding parametrization.

4.2 Order-preserving matchings

Consider a curl-free frame field on two adjacent faces $f, g \in \mathcal{F}$, i.e., $\{\alpha_f, \beta_f, -\alpha_f, -\beta_f\}$ and $\{\alpha_g, \beta_g, -\alpha_g, -\beta_g\}$, with *geometrically-ordered* roots (e.g., counterclockwise-ordered). We next wish to restrict our matchings to preserve the ordering between the vectors (Section 3.3). If we assume that the roots are ordered geometrically, then a matching that preserves this order is either the matching $\alpha_f \rightarrow \alpha_g,\ \beta_f \rightarrow \beta_g$ and the rest respectively, or any cyclic shifting of this matching (e.g., $\alpha_f \rightarrow \beta_g,\ \beta_f \rightarrow -\alpha_g$), giving four viable order-preserving matchings (see Figure 6). We would like to determine an invariant of this kind of matching that distinguishes them from the rest. Consider the following curl quotients, defined on a consecutive pair of ordered vectors in face f:

$$q_{f|e}(\alpha_f, \beta_f) = \frac{\mathrm{Re}(\alpha_f\,\overline{e_f})}{\mathrm{Re}(\beta_f\,\overline{e_f})},\ q_{f|e}(\beta_f, -\alpha_f) = \frac{\mathrm{Re}(-\beta_f\,\overline{e_f})}{\mathrm{Re}(\alpha_f\,\overline{e_f})}, \ldots \tag{12}$$

These four numbers are pairwise related by an equality, e.g., $q_{f|e}(-\beta_f, -\alpha_f) = q_{f|e}(\beta_f, \alpha_f)$, or a reciprocal negation, e.g., $q_{f|e}(-\beta_f, \alpha_f) = -1/q_{f|e}(\alpha_f, \beta_f)$. Thus, we can compactly choose $q_{f|e}(\alpha_f, \beta_f)$ to represent them all. For the sake of clarity, we omit the vectors from the notation: $q_{f|e} = q_{f|e}(\alpha_f, \beta_f)$.

Since we assume to have vanishing PolyCurl, we immediately get the following:

Lemma 4.2. *The curl-free matching is order-preserving if and only if either $q_{f|e} = q_{g|e}$ or $q_{f|e} = -1/q_{g|e}$.*

Proof. If the PolyCurl matching is order-preserving, the proof is trivial: α_f matched to α_g produces $q_{f|e} = q_{g|e}$, the next shift produces $q_{f|e} = -1/q_{g|e}$, and so on (see Figure 6). We prove the reverse direction (equal curl quotient leads to order-preserving matching) by contradiction: for every non-order-preserving matching, such as α_f to α_g and β_f to $-\beta_g$, we must have that either $q_{f|e} = -q_{g|e}$ or $q_{f|e} = 1/q_{g|e}$. This can be checked by a simple enumeration of all "bad" matchings. □

We can thus formulate our condition: a PolyVector field is curl-free and the curl-free matching is order-preserving if and only if the PolyCurl is zero, and the following two *quotient curl polynomials* are equivalent:

$$\begin{aligned} Q_{f|e}(r) &= \left(r - q_{f|e}\right)\left(r + 1/q_{f|e}\right), \\ Q_{g|e}(r) &= \left(r - q_{g|e}\right)\left(r + 1/q_{g|e}\right). \end{aligned} \tag{13}$$

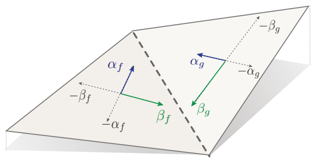
The non-vanishing coefficients of the monic quotient curl polynomial for face f, $Q_{f|e}$, are $q_{f|e} + 1/q_{f|e}$ and -1; similarly for face g. Thus, in the PolyVector spirit, we measure the difference between the non-constant coefficients and denote it as *PolyQuotient*:

$$\mathrm{PolyQuotient}_e = (q_{f|e} + 1/q_{f|e}) - (q_{g|e} + 1/q_{g|e}). \tag{14}$$

Avoiding the division-by-zero. There is an inherent problem in the definition of $q_{f|e}$, since the denominator may be zero. However, we never use $q_{f|e}$ as a variable, and instead incorporate it in the optimization of the PolyQuotient by eliminating common denominators between equation sides (see Section 5 for further details). The formulation is naturally still valid after such an elimination.

Note again that using PolyQuotient does not encode a specific matching between sets; rather, it restricts the matching to the four admissible order-preserving possibilities, and in that it does not compromise the degrees of freedom required by our continuous optimization.

Geometrically-ordered roots. The PolyQuotient constrains the combinatorial order of matched roots. Minimizing the PolyQuotient, however, is not sufficient to ensure that an order-preserving, curl-free matching will be selected. If the geometric order of the four vectors inside each face is not preserved (see inset), the PolyQuotient might be zero, and the matching could be unordered. Thus, in order to preserve local injectivity, we must additionally make sure that the roots are geometrically ordered within each face. This can be conveniently expressed in complex notation as an inequality per face that enforces a convex angle between α_f and β_f, thus guaranteeing a counter-clockwise order of all roots:



$$\mathrm{Im}(\beta_f\,\overline{\alpha_f}) > 0. \tag{15}$$

5 Optimization

Using the definitions in the previous sections, we formulate a non-linear optimization problem that accepts a smooth frame field (and optional directional constraints) as input and outputs a frame field that is as similar as possible to the input, but where both PolyCurl and PolyQuotient are eliminated. The optimized field is thus integrable and induces an inversion-free parameterization. Smoothness of the field and alignment to user-provided constraints are also taken into account in the optimization.

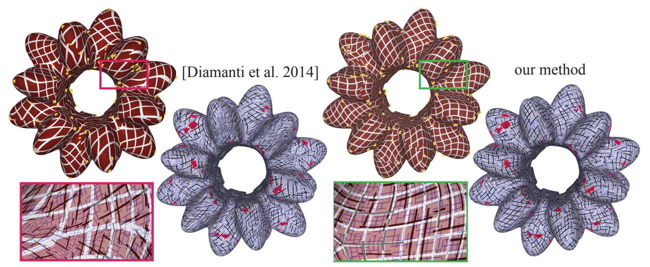


Figure 7: *Our algorithm can be used as a controllable projection from any non-integrable field onto the space of integrable fields, which are exactly aligned to parameterizations. Starting from any non-integrable field that interpolates sparse constraints (e.g.[Diamanti et al. 2014]), we can produce new fields that are reasonably similar but also integrable. The input constraints are highlighted in red.*

Variables. We encode the frame field with two representative vectors $\{\alpha_f, \beta_f\}$ for each face $f \in \mathcal{F}$. These vectors are expressed as complex numbers in the local coordinate system of the plane that contains f. This coordinate system can be arbitrarily chosen; we use the first edge of the face as the real axis. The variable vector per face is $z_f = [\mathrm{Re}(\alpha_f), \mathrm{Im}(\alpha_f), \mathrm{Re}(\beta_f), \mathrm{Im}(\beta_f)]^\top$, and the complete variable vector $z \in \mathbb{R}^{4|\mathcal{F}|}$ for the mesh is the vertical concatenation of these vectors for all faces.

Energy. Our objective function is a sum of five squared residuals; the first three measure smoothness, PolyCurl and PolyQuotient, respectively; the fourth term ensures that the geometric order of the vectors in each face is preserved, and the fifth term minimizes deviation from user-provided alignment constraints, while also acting as a regularization term to improve the numerical stability.

$$\begin{aligned} E(z) &= \sum_{\{f,g\}\sim e} \mathcal{S}(z_f, z_g) + \mathcal{P}(z_f, z_g) + \mathcal{Q}(z_f, z_g) \\ &\quad + \mathcal{B}(z) + \mathcal{C}(z). \end{aligned} \tag{16}$$

Each term is weighted with fixed weights w_* that we discuss below.

Smoothness. The monic frame field polynomial is quartic with two non-zero complex coefficients, denoted for a given face f as $\mathcal{C}_{f,0}, \mathcal{C}_{f,2} \in \mathbb{C}$. The relation between roots and coefficients is:

$$\begin{aligned} \mathcal{C}_{f,0}(z_f) &= (\alpha_f)^2(\beta_f)^2 \\ \mathcal{C}_{f,2}(z_f) &= -\left[(\alpha_f)^2 + (\beta_f)^2\right]. \end{aligned} \tag{17}$$

The smoothness is measured with the Dirichlet energy of the polynomial coefficients [Diamanti et al. 2014] on each edge (f, g); assuming normalized edge vectors, this reduces to:

$$\begin{aligned} \mathcal{S}(z_f, z_g) &= w_s \left|\mathcal{C}_{f,0}(z_f)(t_{fg})^4 - \mathcal{C}_{g,0}(z_g)\right|^2 + \\ &\quad w_s \left|\mathcal{C}_{f,2}(z_f)(t_{fg})^2 - \mathcal{C}_{g,2}(z_g)\right|^2, \end{aligned} \tag{18}$$

where the constant transport term $t_{fg} = \overline{e_f}(\overline{e_g}^{-1})$ is precomputed for each edge.

PolyCurl. The PolyCurl term is similar to the smoothness term, as it measures the difference of polynomial coefficients. Assuming normalized edges, the per-edge PolyCurl term (Eq. (10)) reduces to:

$$\begin{aligned} \mathcal{P}(z_f, z_g) &= w_p^2 \left(c_{f|e,0}(z_f) - c_{g|e,0}(z_g)\right)^2 + \\ &\quad w_p \left(c_{f|e,2}(z_f) - c_{g|e,2}(z_g)\right)^2. \end{aligned} \tag{19}$$

Note that the two coefficients are scaled differently in the above sum. This is to counteract the relative differences in their magnitudes, resulting from the different monomial degree these coefficients belong to.

PolyQuotient. The PolyQuotient term follows from a straightforward manipulation of the PolyQuotient polynomial (Eq. (13)), which yields:

$$\hat{q}_1(z_f, z_g) = \hat{q}_2(z_f, z_g), \tag{20}$$

where

$$\hat{q}_1(z_f, z_g) = \left(\mathrm{Re}(\alpha_f\,\bar{e})^2 + \mathrm{Re}(\beta_f\,\bar{e})^2\right)\mathrm{Re}(\alpha_g\,\bar{e})\,\mathrm{Re}(\beta_g\,\bar{e}),$$

$$\hat{q}_2(z_f, z_g) = \left(\mathrm{Re}(\alpha_g\,\bar{e})^2 + \mathrm{Re}(\beta_g\,\bar{e})^2\right)\mathrm{Re}(\alpha_f\,\bar{e})\,\mathrm{Re}(\beta_f\,\bar{e}).$$

Similarly to the previous terms, this energy term is defined for each edge $e \sim (f, g)$ as follows:

$$\mathcal{Q}(z_f, z_g) = w_q\left(\hat{q}_1(z_f, z_g) - \hat{q}_2(z_f, z_g)\right)^2. \tag{21}$$

Note that we eliminate the division by zero in the PolyQuotient original expression by multiplying with the common denominator.

Geometric order. The geometric order term ensures that the vectors in each face cannot change their order during the optimization (Eq. (15)). We observe that this term is necessary only in a small subset of faces, since the vector order tends to be preserved by our optimization. We thus model it as a cubic-spline barrier ([Schüller et al. 2013]) that only affects the total energy whenever the left-hand side of Eq. (15) is close to zero, and vanishes otherwise. We define the barrier as follows:

$$\phi_s(x) = \begin{cases} \infty & \text{if } x \le 0; \\ \frac{1}{b(x)} - 1 & \text{if } 0 < x < s; \\ 0 & \text{if } x \ge s. \end{cases}$$

where

$$b_s(x) = \frac{x^3}{s^3} - \frac{3x^2}{s^2} + \frac{3x}{s},$$

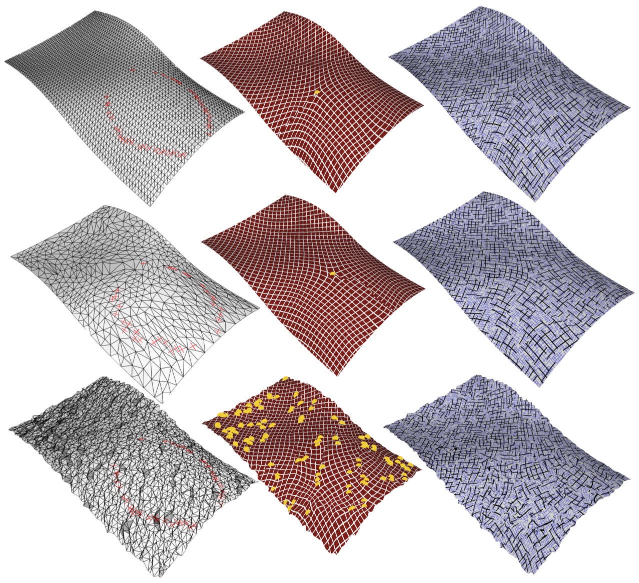


Figure 8: *We demonstrate the robustness of our algorithm to different meshing (middle) and to geometric noise (bottom). In the examples above, all fields have Poisson integration error less than* 0.005*, and the parameterizations contain no inverted triangles.*

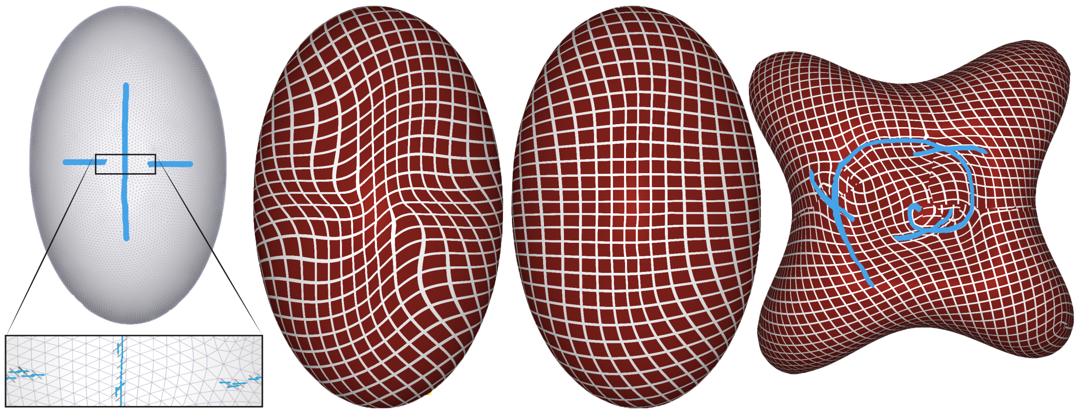


Figure 9: *We prescribe non-orthogonal frame field constraints (first column) which result in a skewed parameterization that is perfectly aligned with them (second column). Using this field as the initial guess, we drop the set of constraints not aligned to the strokes (in gray in the inset image), leaving only the partial constraints in blue. Our algorithm assigns the u coordinate to the horizontal sketch and the v coordinate to the other one, obtaining a smoother solution (third column). Multiple intersecting sets of partial constraints are also supported (fourth column).*

and s is a parameter controlling the point at which the barrier term starts having an effect on the energy. The geometric order term, summed up over all faces, becomes:

$$\mathcal{B}(z) = \sum_{f\in\mathcal{F}} w_b \left[\phi_s\left(\mathrm{Im}(\beta_f \overline{\alpha_f})\right)\right]^2. \tag{22}$$

User constraints and regularization. Alignment of the frame field to user-provided directional constraints is modeled by a quadratic closeness term on the constrained faces. In addition, we model a dampening term that regularizes the solution in the same manner, favoring frame fields that are close to the initial guess whenever multiple solutions have a similar energy value. We combine both terms as follows:

$$\mathcal{C}(z) = \begin{cases} \sum_{f\in\mathcal{F}} w_{\mathrm{cn}} \|z_f - z_f^{\mathrm{cn}}\|^2, & f \text{ is constrained;} \\ \sum_{f\in\mathcal{F}} w_r \|z_f - z_f^{\mathrm{prev}}\|^2, & f \text{ is free.} \end{cases} \tag{23}$$

Note that this formulation naturally supports partial directional constraints. By constraining only the first root in a face, we leave the other free to minimize our energy. In the context of parameterization, this means that the given constraints fix a direction that corresponds to either the u or the v coordinate of the parameterization. We remind that this does not mean that we limit this direction to *globally* become either u or v, since the matching between faces is still free (up to order preservation).

Note. Users commonly use constraints to indicate the directions that the isolines of the parameterization should follow. The u (resp. v) isoline is perpendicular to the gradient of the u (resp. v) coordinate function; this means that our desired curl-free field should always be rotated by $\pi/2$ with respect to the constraints in order for those constraints to actually guide the direction of the isolines. We thus rotate the constraints before providing them to the solver and, in all our figures, we always show the output field rotated by $\pi/2$; if the original (non-rotated) field is curl-free, then its rotation will be exactly aligned to the parameterization.

Optimization. Our energy function $E(z) = \sum E_i(z)^2$ is a sum of squared residuals, each of which is evaluated either on an edge or on a face. We minimize E using Gauss-Newton iterations, which have the advantage of not requiring an explicit Hessian. This algorithm starts with an initial guess $z^{(0)}$ and proceeds from step k to $k+1$ by the iterations

$$z^{(k+1)} = z^{(k)} - \gamma^{(k)}\delta^{(k)}.$$

The direction $\delta^{(k)}$ is given by the solution to the following linear system;

$$J\big(z^{(k)}\big)^\top J\big(z^{(k)}\big)\delta^{(k)} = J\big(z^{(k)}\big)^\top \mathbf{E}(z^{(k)}),$$

where $\mathbf{E}(z^{(k)})$ is the vector of all the concatenated least-squares terms $E_i(z^{(k)})$ in the energy and $J\big(z^{(k)}\big)$ is the Jacobian matrix, both evaluated at the current solution $z^{(k)}$. The i,j-th element of $J\big(z^{(k)}\big)$ contains the gradient of the energy term E_i with respect to the j-th variable in z, i.e., $J\big(z^{(k)}\big)_{ij} = \partial E_i(z^{(k)})/\partial z_j$. The formulas for the calculation of all gradients of our functions can be found in the additional material.

The step size $\gamma^{(k)}$ is iteratively set via backtracking line search: we start with an initial value of 1 and iteratively halve it and compute an updated solution until the energy becomes smaller than the starting point of the line search; once such a solution is found, we double the value of γ and keep it for the next iteration.

To compute the initial guess $z^{(0)}$, we use the interpolation method proposed in [Diamanti et al. 2014]. The initial solution is hence a smooth PolyVector field that conforms to the user constraints, computed by solving a linear system in the PolyVector coefficients.

Parameters and smoothness dampening. The parameters of our algorithm were fixed in all our experiments to $w_s = 1$, $w_p = 10$, $w_q = 10$, $w_b = 0.001$, $s = 0.5$, $w_r = 1e-3$, $w_{\mathrm{cn}} = 10$. If no user constraints are provided, our algorithm finds an integrable frame field close to the initial guess. For these experiments we use $w_r = 1$. The values for these parameters were set after some rough experimentation to ensure that the relative magnitudes of the energy components are in the correct range, e.g., the curl terms should dominate over smoothness, which in turn should be significantly higher than the regularization term. The barrier term should be kept just high enough to prevent element inversion but not affect the overall energy too much. As long as these guidelines are respected, we obtain similar results regardless of the exact parameter values (i.e., for variations around half an order of magnitude).

The smoothness term is useful to regularize the results toward smoothness and hence fewer singularities, but it might prevent PolyCurl and PolyQuotient to reach the numerical zero. We thus decrease it by half every 5 iterations.

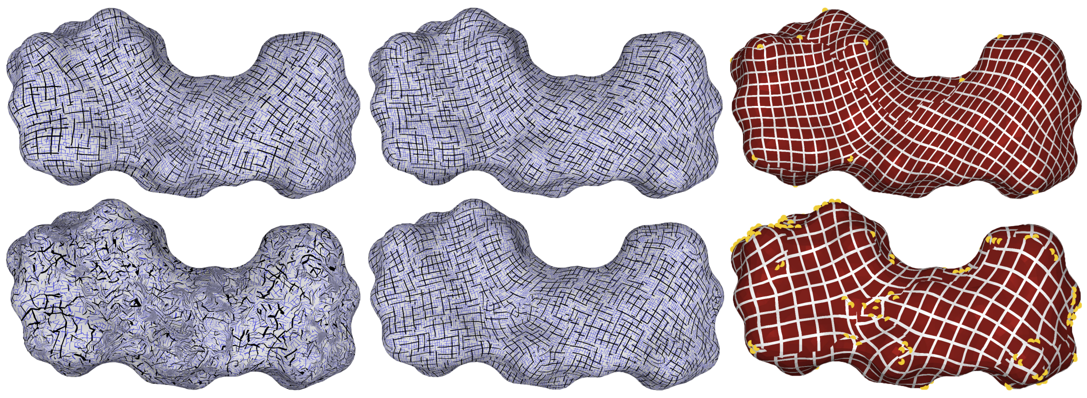


Figure 10: *Top row, left to right: the smooth vector field that is used as the starting point for our optimization, the resulting integrable field and the corresponding parameterization. Bottom row: we add 50% noise to the starting point (left); our algorithm still converges to a perfectly integrable field (middle), inserting approximately 200 additional singularities.*

Implementation details. We stop the iterations when no inversions are present in the parameterization and the average residual of the Poisson step is lower than $1e-3$ (this value is normalized over the lengths of the reconstructed field). In our implementation, we use PARDISO [Kuzmin et al. 2013] to solve the large, yet sparse, linear system in every Gauss-Newton iteration.

6 Results

We implemented our algorithm in C++, and we run our experiments on a workstation with a 2.7 GHz 12-Core Intel Xeon E5 and 64 GB of memory.

Projecting to the space of integrable fields. Fields designed to align to sparse constraints (e.g., the directions prescribed in Figure 7) can arbitrarily deviate from the resulting parameterization due to the Poisson step. Our algorithm can controllably project such fields onto the space of integrable fields, producing integrable fields that are reasonably close to the input fields while ensuring that no unexpected deviations will occur in the parameterization. The input field in Figure 7 was generated with [Diamanti et al. 2014], using a principal direction and a direction at 45 degrees angle to it as constraints, since [Diamanti et al. 2014] supports arbitrary frame fields. Note that any other method can be used for the input field (e.g. [Bommes et al. 2009], [Knöppel et al. 2013]).

Partial constraints. Our method supports partial prescription of constraints, i.e., it is possible to fix only one of the two directions of the field. This requires no modifications to our energy and, since the matchings are free to change in our optimization, it will automatically constrain the direction that is ideal for minimizing the energy, which favors smooth fields. Note that multiple strokes are possible, and they can intersect each other. In our implementation, each stroke is converted into a curl-free vector field on the faces which intersect the stroke by using the Helmholtz decomposition.

Robustness to the initial guess. Our optimization needs a starting point; in all our examples we use the smooth field interpolated using [Diamanti et al. 2014]. We test the robustness of our method by adding noise to our initial solution; the algorithm still produces a perfectly integrable field even in extreme cases (Figure 10).

Robustness to input mesh quality. Our method is robust to mesh resolution and triangle quality. In Figure 8, we compute an integrable frame field aligned with the curvature constraints highlighted with red crosses. The same experiment is executed with a decimated mesh (middle), obtaining a very similar result. Finally, we perturb the mesh with high geometric noise (bottom), and even in this case our algorithm succeeds in finding a smooth integrable frame field, introducing 160 extra singularities.

Angle preservation. The geometric order term (Eq. (22)) is closely related to the angle between the representative directions α and β; by setting the parameter s to a value close to 1, we favor solutions where α and β are as orthogonal as possible (Figure 11). This is particularly useful if the parameterization is used for quadrangulation purposes.

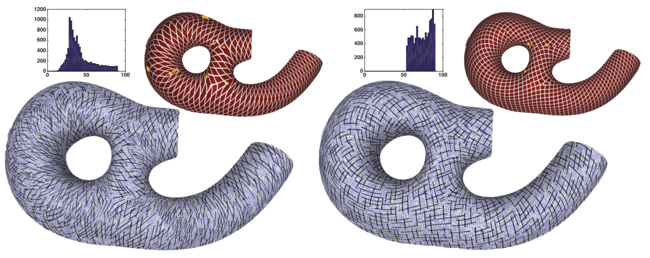


Figure 11: *In both examples, we use the same field constraints, but a different value of s. On the left ($s = 0.1$) the optimization is free to add angle distortion to minimize the curl of the field. On the right ($s = 0.9$) the angles are closer to $\pi/2$, at the price of introducing more singularities. Both fields are integrable.*

Integer rounding. The parameterization induced by an integrable field requires additional processing before it can be used for quadrangulation: our optimization ensures that it is seamless, but the additive factors on the cuts might not be integers (see Section 3). To round them, we apply the parameterization algorithm proposed in [Bommes et al. 2009] to our fields. This step introduces a Poisson error (and in some cases element inversions) that decreases as we increase the resolution of the output mesh (see Figure 12).

Benchmark and timings. We use the benchmark of [Myles et al. 2014] to evaluate the robustness and performance of our algorithm. We stop our iterations as soon as the field induces a parameterization with no inverted faces after the Poisson step (since this was the main objective of the benchmark). Our algorithm succeeds in all the 116 meshes in the database, a few of which are shown in Figure 13. The mesh size varies from 480 to 150k faces, and our algorithm takes from 0.8 to 40 minutes per model, with an average of 1 minute per model (more than 99% of the meshes require less than four minutes). We provide the full statistics in the supplemental material. Addi-

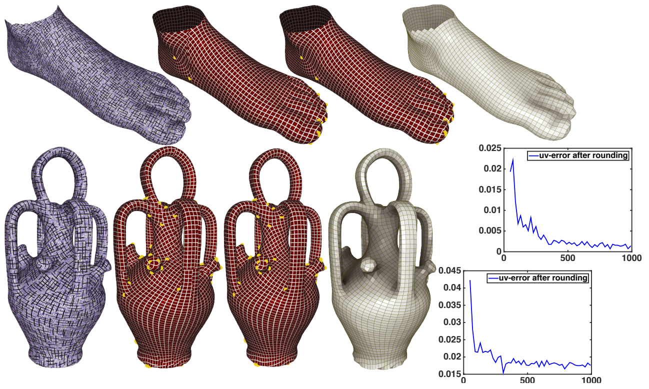


Figure 12: *We apply the rounding scheme of [Bommes et al. 2009] to remesh surfaces into quadrilateral meshes whose edges are aligned with our fields. The graphs show the normalized difference of the per-corner (u, v) coordinates before and after the integer rounding. The x axis shows the number of quads per unit-length vector, which directly correlates to the quad grid resolution.*

tionally, we also run our iterations until the average of the Poisson error is lower than 0.005, which, according to our experiments, is a reasonable threshold to avoid any visible misalignment between the field and the parameterization. Our method succeeds on all models, which we attach in the supplemental material, together with our fields, parameterizations and their screenshots. Our numerical algorithm behaved well in the examples shown, reducing the objective function and its gradient by at least 8 orders of magnitude (from $1e4$ to $1e-4$).

7 Conclusion

Our algorithm is a first attempt to design *integrable* PolyVector fields that trivially induce field-aligned, inversion-free, global parameterizations. We apply our theory and algorithm to the design of quadrangulations, where the user-given directional constraints directly control the edge alignment. In addition to integrability, our formulation supports a novel way to adhere to partial directional constraints, without having to choose the specific isoline a priori.

While our optimization does not rely on integer variables, it is still nonlinear and non-convex, and we cannot formally prove its convergence or guarantee that the result will in fact induce a completely inversion-free parameterization. However, we empirically find our algorithm to be very robust: using default parameters, we obtain integrable fields on hundreds of models without experiencing any convergence issues.

When applied to creating quadrangulations, our fields cannot be immediately used: the boundaries of our parameterizations still need to be rounded to guarantee seamless transitions on the cuts. Tackling this problem without resorting to integer variables is an interesting topic for future theoretical research. In practice, we observe that the rounding has a minor effect when applied to our integrable fields, especially if the output mesh has a high resolution. Finally, it would also be interesting to integrate additional quality criteria, such as an angle bound or anisotropy bound, or to devise additional constraints, such as vector conjugacy for planar-quad meshing.

We note that our definition of inversion-free parametrization optimizes for a positive Jacobian in each face; it is a necessary, but not sufficient condition for true local injectivity [Weber and Zorin 2014], since the boundary can intersect itself in theory. Nevertheless, we conjecture that this not possible with our framework, since such a situation is probably the result of integer-valued singularities. We do not witness this case in practice, and leave the formal proof for future work.

Our paper focuses on frame fields for the purpose of quadrangulation. Nevertheless, our theory and algorithms readily extend to general N-PolyVectors. We are currently investigating other problems that can benefit from the design of integrable N-PolyVector fields, such as hexagonal remeshing, physically-based simulation and surface deformation.

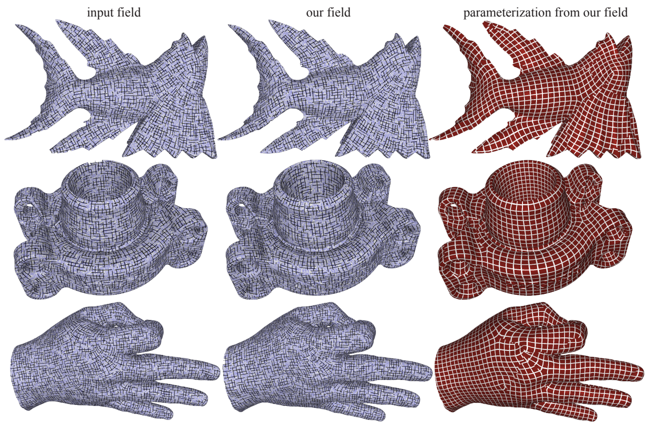


Figure 13: *Sample results from the benchmark. From left to right: reference field, output of our method and parameterization. All three results contain no flips in the parameterization.*

8 Acknowledgements

We thank Marco Tarini for providing the software for the visualization of the vector fields, and Christian Schüller for discussions on optimization with barriers. This work was supported in part by the ERC Starting Grant iModel (StG-2012-306877), the Lise-Meitner grant M1618-N25 and grant P23735-N13 of the Austrian Science Fund (FWF).

References

Alliez, P., Cohen-Steiner, D., Devillers, O., Lévy, B., and Desbrun, M. 2003. Anisotropic polygonal remeshing. *ACM Trans. Graph. 22*, 3, 485–493.

Azencot, O., Ben-Chen, M., Chazal, F., and Ovsjanikov, M. 2013. An operator approach to tangent vector field processing. *Comput. Graph. Forum 32*, 5, 73–82.

Bommes, D., Zimmer, H., and Kobbelt, L. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3, 77:1–77:10.

Bommes, D., Campen, M., Ebke, H.-C., Alliez, P., and Kobbelt, L. 2013. Integer-grid maps for reliable quad meshing. *ACM Trans. Graph. 32*, 4, 98:1–98:12.

Bommes, D., Lévy, B., Pietroni, N., Puppo, E., Silva, C., Tarini, M., and Zorin, D. 2013. Quad-mesh generation and processing: A survey. *Computer Graphics Forum 32*, 6, 51–76.

Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., and Lévy, B. 2010. *Polygon Mesh Processing*. AK Peters.

Campen, M., and Kobbelt, L. 2014. Dual strip weaving: Interactive design of quad layouts using elastica strips. *ACM Trans. Graph. 33*, 6, 183:1–183:10.

CRANE, K., DESBRUN, M., AND SCHRÖDER, P. 2010. Trivial connections on discrete surfaces. *Comput. Graph. Forum 29*, 5.

DIAMANTI, O., VAXMAN, A., PANOZZO, D., AND SORKINE-HORNUNG, O. 2014. Designing N-PolyVector fields with complex polynomials. *Computer Graphics Forum 33*, 5, 1–11.

DONG, S., BREMER, P.-T., GARLAND, M., PASCUCCI, V., AND HART, J. C. 2006. Spectral surface quadrangulation. *ACM Trans. Graph. 25*, 3 (July), 1057–1066.

EBKE, H.-C., BOMMES, D., CAMPEN, M., AND KOBBELT, L. 2013. QEx: Robust quad mesh extraction. *ACM Trans. Graph. 32*, 6, 168:1–168:10.

EBKE, H.-C., CAMPEN, M., BOMMES, D., AND KOBBELT, L. 2014. Level-of-detail quad meshing. *ACM Trans. Graph. 33*, 6, 184:1–184:11.

FISHER, M., SCHRÖDER, P., DESBRUN, M., AND HOPPE, H. 2007. Design of tangent vector fields. *ACM Trans. Graph. 26*, 3.

HERTZMANN, A., AND ZORIN, D. 2000. Illustrating smooth surfaces. In *Proc. ACM SIGGRAPH*, 517–526.

KÄLBERER, F., NIESER, M., AND POLTHIER, K. 2007. QuadCover – surface parameterization using branched coverings. *Computer Graphics Forum 26*, 3, 375–384.

KNÖPPEL, F., CRANE, K., PINKALL, U., AND SCHRÖDER, P. 2013. Globally optimal direction fields. *ACM Trans. Graph. 32*, 4.

KUZMIN, A., LUISIER, M., AND SCHENK, O. 2013. Fast methods for computing selected elements of the Green's function in massively parallel nanoelectronic device simulations. In *Proc. Euro-Par*, 533–544.

LEFEBVRE, S., AND HOPPE, H. 2006. Appearance-space texture synthesis. *ACM Trans. Graph. 25*, 3, 541–548.

LI, Y., BAO, F., ZHANG, E., KOBAYASHI, Y., AND WONKA, P. 2011. Geometry synthesis on surfaces using field-guided shape grammars. *IEEE Trans. Vis. Comput. Graph. 17*, 2, 231–243.

LING, R., HUANG, J., JÜTTLER, B., SUN, F., BAO, H., AND WANG, W. 2014. Spectral quadrangulation with feature curve alignment and element size control. *ACM Trans. Graph. 34*, 1.

LIPMAN, Y. 2012. Bounded distortion mapping spaces for triangular meshes. *ACM Trans. Graph. 31*, 4, 108:1–108:13.

LIU, Y., XU, W., WANG, J., ZHU, L., GUO, B., CHEN, F., AND WANG, G. 2011. General planar quadrilateral mesh design using conjugate direction field. *ACM Trans. Graph. 30*, 6.

MARINOV, M., AND KOBBELT, L. 2004. Direct anisotropic quad-dominant remeshing. In *Proc. Pacific Graphics*, 207–216.

MYLES, A., AND ZORIN, D. 2012. Global parametrization by incremental flattening. *ACM Trans. Graph. 31*, 4.

MYLES, A., AND ZORIN, D. 2013. Controlled-distortion constrained global parametrization. *ACM Trans. Graph. 32*, 4.

MYLES, A., PIETRONI, N., AND ZORIN, D. 2014. Robust field-aligned global parametrization. *ACM Trans. Graph. 33*, 4.

PALACIOS, J., AND ZHANG, E. 2007. Rotational symmetry field design on surfaces. *ACM Trans. Graph. 26*, 3.

PANOZZO, D., LIPMAN, Y., PUPPO, E., AND ZORIN, D. 2012. Fields on symmetric surfaces. *ACM Trans. Graph. 31*, 4.

PANOZZO, D., PUPPO, E., TARINI, M., AND SORKINE-HORNUNG, O. 2014. Frame fields: Anisotropic and non-orthogonal cross fields. *ACM Trans. Graph. 33*, 4, 134:1–134:11.

POLTHIER, K., AND PREUSS, E. 2003. Identifying vector field singularities using a discrete Hodge decomposition. In *Visualization and Mathematics III*, 113–134.

RAY, N., LI, W. C., LÉVY, B., SHEFFER, A., AND ALLIEZ, P. 2006. Periodic global parameterization. *ACM Trans. Graph. 25*, 4, 1460–1485.

RAY, N., VALLET, B., LI, W. C., AND LÉVY, B. 2008. N-symmetry direction field design. *ACM Trans. Graph. 27*, 2.

RAY, N., VALLET, B., ALONSO, L., AND LÉVY, B. 2009. Geometry-aware direction field processing. *ACM Trans. Graph. 29*, 1, 1:1–1:11.

SCHÜLLER, C., KAVAN, L., PANOZZO, D., AND SORKINE-HORNUNG, O. 2013. Locally injective mappings. *Computer Graphics Forum 32*, 5, 125–135.

TAKAYAMA, K., PANOZZO, D., SORKINE-HORNUNG, A., AND SORKINE-HORNUNG, O. 2013. Sketch-based generation and editing of quad meshes. *ACM Trans. Graph. 32*, 4, 97:1–97:8.

WEBER, O., AND ZORIN, D. 2014. Locally injective parametrization with arbitrary fixed boundaries. *ACM Trans. Graph. 33*, 4 (July), 75:1–75:12.

ZHANG, E., MISCHAIKOW, K., AND TURK, G. 2006. Vector field design on surfaces. *ACM Trans. Graph. 25*, 4, 1294–1326.

ZHANG, M., HUANG, J., LIU, X., AND BAO, H. 2010. A wave-based anisotropic quadrangulation method. *ACM Trans. Graph. 29*, 4, 118:1–118:8.