Seminar Report on

Integrable PolyVector Fields

[Dia+15]

Patrick Schmidt

February 29, 2016

Abstract

Designing smooth guiding fields plays a major role in current parametrization-based quad meshing algorithms. While previous methods suffer from expensive mixed-integer optimization, recent publications propose an alternative continuous formulation. New challenges arise in optimizing the resulting fields for integrability and preventing local inversions.

This seminar report presents results from [Dia+14] and [Dia+15] while focusing on the particular application of quadrilateral surface remeshing. It covers the generation of smooth frame fields using complex polynomials as well as a method turning these into curl-free fields using non-linear optimization.

Keywords *Vector field design, integrability, quad meshing*

Contents

1 Introduction **2**
1.1 Quad Remeshing . . . 2
1.2 Parametrization-Based Methods . . . 2
1.3 Challenges in Frame Field Design . . . 3
1.4 Related Work . . . 4

2 Smooth Frame Fields **5**
2.1 Complex Numbers Recap . . . 5
2.2 N-PolyVector Fields . . . 6
2.3 Optimization . . . 7

3 Integrability **9**
3.1 Discrete Curl . . . 9
3.2 Inversions . . . 11
3.3 Optimization . . . 13

4 Results & Conclusion **15**

1 Introduction

In various domains quad meshes, i.e. meshes consisting entirely of four-sided polygons, are the preferred approximation of continuous surfaces. Unlike triangle meshes, these are often capable of aligning their structure to the geometry of the underlying object in a very meaningful way. This is the case if for example, the edge flow of the quad mesh adheres well to the principal curvature directions of the surface.

Applications. One prominent application domain for quad meshes is character animation where artists often work with subdivision surfaces, c.f. [Bom12]. Quad meshes are particularly well suited to this use case because they keep a consistent edge flow under multiple steps of subdivision. If this edge flow is properly aligned to e.g. principal curvature directions, they prevent geometric artefacts like normal noise. For similar reasons, artists also strive to align edges to the expected directions of deformation in the eventual animation.

Quad meshes are also used in physical simulation. Here, their structure heavily influences the stability and approximation quality in e.g. finite element analysis, c.f. [Bom+12].

1.1 Quad Remeshing

Many 3D models are initially created as triangle meshes, for example via a sculpting tool or by triangulating a laser-scanned point cloud. Thus, a common task is to convert an arbitrarily structured triangle mesh into a high quality quad mesh. Since this job is usually still carried out manually (c.f. [Bom12]), researchers try to find algorithms performing the conversion as automatic as possible.

Quality. In practice, quality criteria vary strongly based on the particular application and can even be in conflict to each other. While proper alignment is very important, principal curvature constraints may contradict directions specified by an animation artist.

Furthermore, high regularity is a prevailing goal in general. However, depending on the genus of an object, vertices of valence other than four are inevitable. Even if they are not topologically necessary, they provide more degrees of freedom to capture the curvature of an object. So there exists a natural trade-off between a low number of irregular vertices and well adapted orientation.

Other criteria involve individual element quality like planarity, isotropy and angles being close to 90°. Also feature alignment, i.e. sharp creases in the input surface being directly represented by quad edges, is an important property. A concise list of desired quality criteria is given in [BZK09].

1.2 Parametrization-Based Methods

The currently most successful quad meshing algorithms are indirect methods using a surface parametrization. Intuitively, this means that the input mesh is cut open and developed to a flat 2D surface where it is then covered by a regular Cartesian grid. After that, the object is reassembled to its original 3D shape while keeping the grid as a texture, resulting in a quad mesh drawn onto the surface.

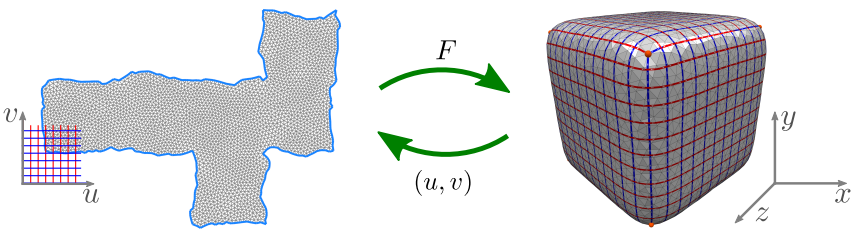


Figure 1: The function F describes an object in parametric form. Inversely, the scalar functions u and v map its vertices to a 2D parameter domain. Figure adapted from [Sch14].

More formally, a surface $\mathcal{S}$ can be described parametrically by a function $F : \mathbb{R}^2 \to \mathbb{R}^3$ mapping from a 2D *parameter domain* to 3D space. Function values of $F(u, v)$ for which either u or v is a constant integer

are called *iso-parameter lines* and indicate quad mesh edges (red and blue lines in Figure 1). Positions where both u and v are integer will result in vertices of the eventual quad mesh.

Usually the inverse of F is computed, consisting of two scalar functions u and v defined on the surface, i.e. $u : \mathcal{S} \to \mathbb{R}$ and $v : \mathcal{S} \to \mathbb{R}$. In a discrete setting these are piecewise linear and map vertices of the input triangle mesh to their locations in the parameter domain. However, at edges where the mesh has been cut, vertices may appear multiple times in the parameter domain. When making no assumptions on how the mesh is cut open, i.e. every edge could be a cut, u and v are actually defined on triangle corners rather than vertices.
The *parametrization* $(u, v) = F^{-1}$ completely determines the eventual quad mesh. Thus, it should be computed in a way that captures the quality criteria mentioned above as well as possible. Yet, not every parametrization implies a valid quad mesh. To assure consistency across seams it is important that both sides of a cut edge are related to each other by a *grid-automorphism* [KNP07] inside the parameter domain, which only allows for full integer translations and rotations by multiples of 90°. Singularities of the parametrization, i.e. vertices spanning an angle other than 360° inside the parameter domain, have to be located at integer positions. This way, they will result in irregular vertices of the quad mesh rather than in irregular faces. In addition, all triangles have to be mapped to the parameter domain with correct orientation. Flipped triangles may prevent the mapped Cartesian grid from forming a pure quad structure. A parametrization fulfilling these necessary properties is called an *integer-grid map* [Bom+13]. Based on such a parametrization, an actual quad mesh can be extracted as shown in [Ebk+13].

1.3 Challenges in Frame Field Design

The gradients of the parametrization coordinate functions u and v are two tangent vector fields defined on the surface, i.e. $\nabla u : \mathcal{S} \to \mathbb{R}^2$ and $\nabla v : \mathcal{S} \to \mathbb{R}^2$ (see Figure 2). On a discrete mesh, these are piecewise constant per triangle.

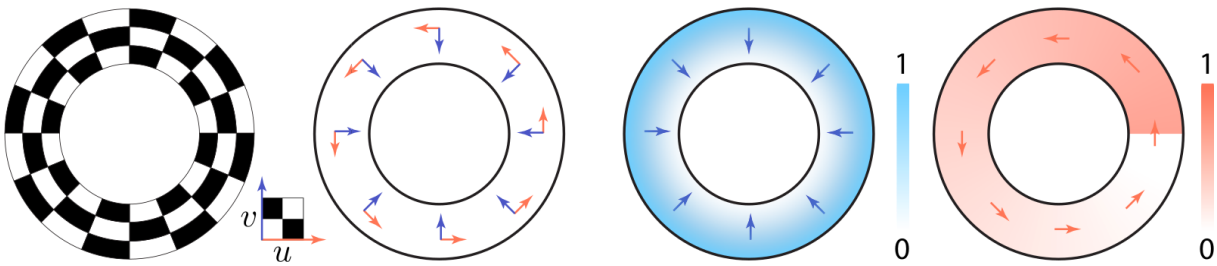


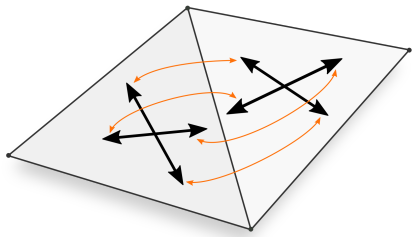
Figure 2: A parametrization consists of two scalar functions u (red) and v (blue). Their gradients are vector fields defined on the surface. Figure adapted from [Dia+15].

Rather than computing a parametrization directly, common methods first compute a guiding field expressing its desired gradients. Because the gradients are related to the local directions of the iso-parameter lines by a rotation of 90° they define the overall alignment of the quad mesh. Initially, a sparse set of directional constraints in form of tangent vectors is specified and then smoothly interpolated over the entire surface. Since it is not known a priori which of the two coordinate functions should follow the prescribed direction—nor is their sign know—four different permutations of the gradient vectors are locally possible. To capture this invariance, a guiding field is defined by four pairwise collinear vectors at each point. Such a field is called a *frame field* if it is allowed to be locally non-orthonormal, i.e. both directions can vary in angle as well as in scale. Orthonormal, i.e. symmetric and unit-length, fields are called *cross fields*.

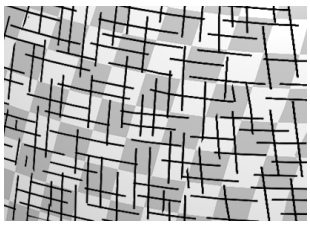
Although these guiding fields are intended to describe the gradients of the parametrization, not every tangent vector field is the gradient of a scalar function. In fact, current methods can only produce parametrizations whose gradients approximately align to the guiding field, e.g. by solving a Poisson equation [BZK09].

The work covered in this report addresses two major challenges in frame field design:

Matchings. In order to provide a measure for the smoothness of a frame field on a discrete triangle mesh, frames in neighboring faces have to be compared to each other. But since there are four vectors in each, it is not immediately clear which vectors correspond to each other. A choice of such pairs across an edge is called a *matching* (see inset). Unfortunately these matchings cannot be computed in advance and are part of the optimization procedure. Thus, many formulations for smooth guiding fields involve unknown integers, resulting in an $\mathcal{NP}$-hard problem.



Integrability. Having computed a guiding field does not mean that there exists a parametrization whose gradient is perfectly aligned to it. For this reason recent methods keep the topology—i.e. number and degree of singularities—of the guiding field but can geometrically have an arbitrarily large deviation from the prescribed gradient (see inset, figure from [Dia+15]). Restricting the design of frame fields to the space of such vector fields that actually are the gradients of some scalar function would allow to obtain the parametrization by simple integration instead of solving a second optimization problem.



[Dia+14] proposes a formulation of frame fields without the use of explicit matchings, thus avoiding integer optimization. [Dia+15] extends the concept and proposes a way to modify such fields in order to favor integrability. Both methods partially serve more general cases than the ones necessary for frame field design. In this report however they will at some points be reduced to the specific case of quad meshing.

1.4 Related Work

Methods to compute fields assigning multiple tangent vectors to each face of a triangle mesh have been proposed by various authors in different levels of generalization.

Cross Fields. Cross fields consist of four orthogonal unit-length vectors per face. Thus, each cross is represented by a single degree of freedom, namely its rotation inside the tangent plane. Cross fields were first introduced in [HZ00] and used for rendering purposes. Smooth cross fields for quad meshing are computed in e.g. [BZK09]. [Ebk+14] computes cross fields on virtually smoothed tangent planes compensating for small-scale noise.

Rotationally Symmetric Fields. More general, rotationally symmetric fields, also called N-RoSy fields, consist of N unit-length vectors in each face. Each consecutive pair of vectors spans an angle of $\frac{2\pi}{N}$. As for cross fields, there is a single degree of freedom per triangle and these fields do not encode sizing or anisotropy. Angle-based N-RoSy fields are computed in [Ray+08] from prescribed singularities or in [Ray+09] from sparse directional constraints. A vector-based representation is proposed in [PZ07]. Note that cross fields are a special case of these, i.e. 4-RoSy fields.

Non-Orthonormal Fields. Recently, fields that in addition encode anisotropy and scaling receive lots of attention. In [Pan+14] frame fields are defined which are cross fields extended by a linear transformation per triangle. The authors of [Jia+15] build upon this idea and obtain frame fields by computing cross fields in a customized metric. Finally, [Dia+14] proposes N-PolyVector fields, in which N unordered vectors are assigned to each face. Each vector can arbitrarily vary in length as well as in angle. In this sense, all above vector fields can be seen as special cases of N-PolyVector fields.

Curl-Free Fields. Apart from [Dia+15], only [Ray+06] and [MPZ14] are known to improve the integrability of frame fields by reducing their discrete curl. The former applies a heuristic modifying the length of vectors before using them to compute a parametrization.

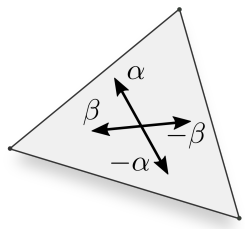
[Dia+15] heavily relies on previous results published in [Dia+14], namely computing smooth frame fields using complex polynomials. For this reason Section 2 is devoted to those details from [Dia+14] that are also an essential part of the method presented in [Dia+15].

From here on, the contents of this report are based on [Dia+14] and [Dia+15] unless stated otherwise.

2 Smooth Frame Fields

As stated above, a major challenge is to compute smooth frame fields, while factoring out the role of matchings from the process. This is achieved in [Dia+14] using a complex number representation.

Frame Fields are formally introduced in [Pan+14]. On a discrete triangle mesh, a frame field assigns two tangent vectors $\alpha, \beta \in \mathbb{R}^2$ to each face of the input mesh. The vectors can vary in angle as well as length but are required to be linearly independent. Because the orientation of the desired gradients ∇u and ∇v is not known in advance, two copies of α and β, each rotated by 180°, are added (see inset). Altogether, a single frame is described as the vector set $\{\alpha, \beta, -\alpha, -\beta\}$.



2.1 Complex Numbers Recap

Complex numbers provide a way to extend the one-dimensional space of real numbers into a second dimension. Fortunately, basic operations on these numbers come with some very intuitive interpretations in 2D geometry, which are often exploited in practice.

This section gives a short summary of definitions and properties used in the following. The contents of this section are primarily based on the lecture notes [Ang11].

A *complex number* $z \in \mathbb{C}$ is a tuple (a, b) of real numbers $a, b \in \mathbb{R}$ that is written as $z = a + bi$. The first component $\mathrm{Re}(z) = a$ is called the *real part* and the second one $\mathrm{Im}(z) = b$ the *imaginary part* of z. The symbol i denotes the *imaginary unit* for which one defines the special property $i^2 = -1$.

Geometrically, a complex number can be visualized as a two-dimensional position vector (a, b) embedded in the so-called *complex plane*. $\mathrm{Re}(z)$ and $\mathrm{Im}(z)$ form the positive axes of a Cartesian coordinate system.

Polar Representation. An alternative representation of a complex number is possible in terms of its *polar coordinates*. This means that instead of a and b, one stores the length $|z|$ of the vector as well as the angle Θ that it spans with the positive real axis. The length, or *absolute value*, of z is defined by the Euclidean norm $|z| = \sqrt{a^2 + b^2}$. The angle, also called the *argument* $\arg(z) = \Theta$, can be computed via the fact that $\tan\Theta = \frac{b}{a}$. Conversely, a complex number can be converted from polar form to its coordinate representation via

$$z = |z| \, (\cos\Theta + \sin\Theta \; i).$$

Note that this representation is invariant in incrementing or decrementing Θ by integer multiples of 2π.

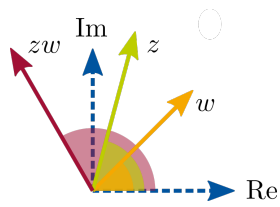
Addition. The sum of two complex numbers $z = a + bi$ and $w = c + di$ is naturally defined by the sum of its individual parts

$$z + w = (a + c) + (b + d)i.$$

It can, just like the addition of two 2D vectors, be seen as a translation.

Multiplication. When taking the product of z and w, the polar representation turns out to be very intuitive. By using the definition $i^2 = -1$ as well as the trigonometric sum and difference formulas one gets:

$$\begin{aligned} zw &= |z| \, (\cos\Theta + \sin\Theta \; i) \, |w| \, (\cos\Psi + \sin\Psi \; i) \\ &= |z||w| \, (\cos(\Theta + \Psi) + \sin(\Theta + \Psi) \; i) \end{aligned}$$



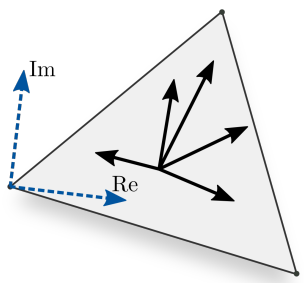
This shows that computing the product of two complex numbers multiplies their lengths and adds their angles. Thus, complex numbers with unit length can be used to describe rotations in the 2D plane. For example the product wz or equivalently zw with $w \in \mathbb{C}$ and $|w| = 1$ rotates z by $\arg(w)$ in counter-clockwise direction (see inset).

Complex Conjugate. An important unary operation is the *complex conjugate* defined as $\bar{z} = a - bi$. This can be visualized as the reflection of a vector across the real axis. Hence, in polar representation the angle Θ changes to $-\Theta$ while the length $|z|$ stays the same. So the inverse of a rotation $r \in \mathbb{C}$ with $|r| = 1$ can be obtained by $\bar{r}$.

Complex Polynomials. Like with real numbers, a polynomial of degree N is a function of the form $P(z) = \sum_{i=0}^{N} c_i z^i$ with complex coefficients $c_i \in \mathbb{C}$ and a complex variable $z \in \mathbb{C}$. The fundamental theorem of algebra states that every non-zero polynomial of degree N has exactly N complex roots. Though, the same root may appear multiple times.

2.2 *N*-PolyVector Fields

In [Dia+14] the notion of *N-PolyVectors fields* is established. These are a generalization of various existing types of fields assigning multiple vectors to each point. In particular, they contain frame fields as a special case. Smooth N-PolyVector fields can be found by solving a sparse linear system.



An *N-PolyVector* for $N \geq 1$ is an unordered set of N two-dimensional vectors. In an *N-PolyVector field*, defined on a discrete triangle mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$, such a set of vectors is assigned to each face in $\mathcal{F}$ (see inset). Thus, an N-PolyVector field can be seen as N vector fields which are piecewise constant per face. However because each vector set is unordered, there is no information on which vector belongs to which field.

Individual vectors live in the tangent space of their respective triangle. Thus, they are two-dimensional vectors embedded in 3D. For a coordinate representation of these vectors an orthonormal basis has to be defined per face. This basis can be selected arbitrarily as all of the following computations are invariant to its choice. When using such a basis to define the complex plane, a single vector can be represented by a complex number $\alpha \in \mathbb{C}$. Finally, an N-PolyVector living in face $f \in \mathcal{F}$ is written as the set $\{\alpha_{f,0}, \alpha_{f,1}, \ldots \alpha_{f,N-1}\}$ with each $\alpha_{f,m} \in \mathbb{C}$. In the following, m will be consistently used as an index between 0 and $N-1$, and the whole vector set of face f can be abbreviated by $\{\alpha_{f,m}\}$.

Representation via Complex Polynomials. As motivated in Section 2, a central challenge is to compute smooth N-PolyVector fields without using explicit matchings. For this reason one is looking for a way to store and compare PolyVectors completely invariant of their order. This is possible by encoding them as the roots of complex polynomials.

A complex polynomial $P(z)$ of degree N has exactly N roots i.e. values of z for which $P(z) = 0$ holds (see Section 2.1). These roots are complex numbers which are uniquely determined by the $N+1$ polynomial coefficients. So it is possible to store the coefficients of P instead of its roots.

Given a PolyVector $\{\alpha_0, \alpha_1, \ldots \alpha_{N-1}\}$, a polynomial with exactly this root set can be set up by a combination of linear factors:

$$P(z) = (z - \alpha_0) \cdot (z - \alpha_1) \cdot \ldots \cdot (z - \alpha_{N-1})$$

Expanding the expression yields the coefficients $x_0, x_1, \ldots x_N \in \mathbb{C}$:

$$P(z) = x_N z^N + x_{N-1} z^{N-1} + \ldots + x_1 z^1 + x_0$$

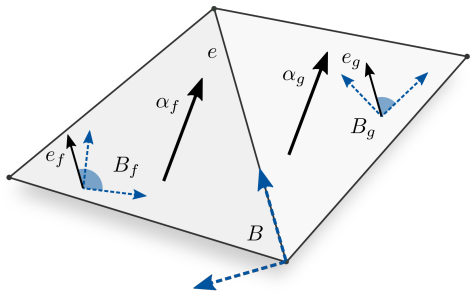
Notice that a polynomial can always be scaled by a factor without changing its zero-crossings. This unnecessary degree of freedom is removed by fixing x_N to 1. This naturally happens when constructing P as above. Now the N vectors $\alpha_0, \alpha_1, \ldots \alpha_{N-1} \in \mathbb{C}$ uniquely determine the N coefficients $x_0, x_1, \ldots x_{N_1} \in \mathbb{C}$ and vice versa (see Section 2.3).

Example. A 2-PolyVector $\{\alpha, \beta\}$ can be encoded by

$$\begin{aligned} P(z) &= (z - \alpha) \cdot (z - \beta) \\ &= z^2 - (\alpha + \beta) z + \alpha\beta. \end{aligned}$$

So the complex coefficients $x_1 = -(\alpha + \beta)$ and $x_0 = \alpha\beta$ are stored.

The Discrete Levi-Civita Connection. To be able to measure smoothness, PolyVectors in adjacent faces have to be compared to each other. As a first step, one needs a definition stating in which case neighboring vectors, living in different tangent spaces, are considered parallel. In differential geometry this concept of parallel transport is defined by a so-called *connection*. It tells how a vector changes if it is moved along a surface. A very natural choice is the *Levi-Civita connection* where, in a discrete setting, a vector is transported to a neighboring face by flattening the two faces onto the same plane along their common edge, moving the vector to the new face and finally reversing the flattening (cf. [Cra+13]).



Eventually, the only thing the Levi-Civita connection does is to compensate the change of bases between adjacent faces. Since orthonormal bases are used here, this boils down to a single 2D rotation per directed dual edge. This rotation is computed by bringing both vectors into a common coordinate system. Assume vectors α_f and $\alpha_g \in \mathbb{C}$ in triangles f and g, expressed in orthonormal bases B_f and B_g. Both faces share a common edge e. The edge vector can be expressed in the local coordinate systems and is called e_f and e_g respectively (see inset). Furthermore, assume that e_f and e_g are normalized. After flattening the triangle flap, the common coordinate system B can for example be chosen with e as the first (real) coordinate axis, and a counter-clockwise rotation by 90° as the second (imaginary) one. To compute a transformation from B_f to B, see that e_f spans an angle $\arg(e_f)$ to the real axis of B_f. Rotating B_f by this angle will align both coordinate systems. But instead of changing the basis, the vector α has to be transported from B_f to B. Thus, the inverse rotation has to be applied to α_f. As seen in Section 2.1, multiplying with e_f rotates a vector by $\arg(e_f)$. Since e_f is normalized, the length will stay the same and $\bar{e}_f$ yields the inverse rotation. Of course the same holds for face g. So $\alpha_f \bar{e}_f$ and $\alpha_g \bar{e}_g$ are vectors expressed in the same coordinate system. One says α_f and α_g are parallel with respect to the Levi-Civita connection or *LC-parallel* if and only if

$$\alpha_f \bar{e}_f = \alpha_g \bar{e}_g.$$

In other words, the Levi-Chivita connection rotates a vector by an angle of $\arg(\bar{e}_f e_g)$ when transporting it from face f to g.

Transporting PolyVector Coefficients. Now the above transport has to be applied to N-PolyVectors, where not the individual vectors but rather their polynomial coefficients are known. Given a PolyVector $\{\alpha_{f,m}\}$ in face f, transporting it to a common basis along e_f gives $\{\alpha_{f,m}\bar{e}_f\}$. In this basis, the factored polynomial looks like

$$P_{f|e_f}(z) = (z - \alpha_{f,0}\bar{e}_f) \cdot (z - \alpha_{f,1}\bar{e}_f) \cdot \ldots \cdot (z - \alpha_{f,N-1}\bar{e}_f).$$

The authors state that if expanding the untransported polynomial yields coefficients $x_0, x_1, \ldots x_N$ as before, expanding $P_{f|e_f}(z)$ gives transported coefficients

$$x_0(\bar{e}_f)^N, x_1(\bar{e}_f)^{N-1}, \ldots x_N(\bar{e}_f)^0$$

where the index of the coefficient and the power of $\bar{e}$ always sum up to N. The coefficient $x_N(\bar{e})^0$ is still 1 so it will be skipped in the following. Now the concept of two single vectors being LC-parallel can be extended to PolyVectors: $\{\alpha_{f,m}\}$ and $\{\alpha_{g,m}\}$ are LC-parallel if and only if their transported polynomials are equal, i.e. their transported coefficients are the same:

$$x_{f,m}(\bar{e}_f)^{N-m} = x_{g,m}(\bar{e}_g)^{N-m}$$

for all $0 \leq m \leq N-1$.

The most important aspect of this formulation is that it works without using an explicit matching between vectors in neighboring faces. If two PolyVectors are LC-parallel, every vector in one face indeed has an LC-parallel counterpart in the other face, but this pairing is not encoded anywhere.

2.3 Optimization

An N-PolyVector field in which all neighboring PolyVectors are LC-parallel to each other is not possible on arbitrary geometry. So instead a field should be found that exactly matches fully specified frames in

a sparse subset of constrained faces and is as close to LC-parallel as possible across every edge. Since the deviation of neighboring frames from being LC-parallel cannot be measured directly, the difference of their polynomial coefficients is regarded instead. As seen above, both formulations yield the same minimum—frames are LC-parallel if and only if their transported coefficients match—however, they might approach this minimum differently. Thus, the following energy measuring the smoothness of the coefficient field represents the smoothness of the associated N-PolyVector field only indirectly:

$$E_{\text{smooth}} = \sum_{(f,g)\in\mathcal{E}} \sum_{m=0}^{N} \left| x_{f,m}(\bar{e}_f)^{N-m} - x_{g,m}(\bar{e}_g)^{N-m} \right|^2$$

For each edge it measures the squared deviation between each pair of transported coefficients. As a sum of squared distances it is separable into an individual energy term E_m for each set of degree-m coefficients. The unknown vector $\hat{x}_m$ denotes the concatenation of all degree m coefficients and each $\hat{x}_m$ can be optimized independently in order to minimize E_m. Since E_m is a Dirichlet energy, its minimizer can be found by solving for the $\hat{x}_m$ whose Laplacian is zero. The authors propose to use a discrete face-based Laplacian weighted by the respective transport terms. This can be written as a complex matrix $L_m \in \mathcal{R}^{|\mathcal{F}\times\mathcal{F}|}$. Each row representing a face f contains the diagonal entry $(\bar{e}_f)^{N-m}$ and three values $-(\bar{e}_g)^{N-m}$ for its neighboring faces. Finally, solving the N independent sparse linear systems $L_m\hat{x} = 0$ gives a smooth coefficient field on the entire mesh.

PolyVector Extraction. The last step is extracting the actual PolyVector field from the coefficient field computed so far. This means that for each face, the the root set $\{\alpha_0, \dots \alpha_{N-1}\}$ of the given polynomial $P(z)$ with coefficients $1, x_{N-1}, \dots x_1, x_0$ has to be found. This is possible by computing the eigenvalues of the companion matrix.

The companion matrix is defined as

$$A = \begin{bmatrix} -x_{N-1} & \dots & -x_1 & -x_0 \\ 1 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 1 & 0 \end{bmatrix}$$

and has the property that $det(zI - A) = P(z)$, (cf. [Fie03]) thus its eigenvalues are the roots of $P(z)$.

Frame Fields. In the special case of frame fields there are four vectors $\{\alpha, \beta, -\alpha, -\beta\}$ per face, i.e. $N = 4$. However, the partial symmetry simplifies this case even further. For face f the polynomial is constructed as

$$\begin{aligned} P(z) &= (z-\alpha)\cdot(z-\beta)\cdot(z+\alpha)\cdot(z+\beta) \\ &= (z^2-\alpha^2)\cdot(z^2-\beta^2) \\ &= z^4 - (\alpha^2+\beta^2)z^2 + \alpha^2\beta^2. \end{aligned}$$

The coefficient vector is $\left(1, 0, -(\alpha^2+\beta^2), 0, \alpha^2\beta^2\right)$ with non-trivial coefficients only in the second and zero order terms. So only $x_2 = -(\alpha^2+\beta^2)$ and $x_0 = \alpha^2\beta^2$ are part of the optimization. Their transport terms are $(\bar{e}_f)^2$ and $(\bar{e}_f)^4$ respectively. By fixing the third and first order coefficients to zero, the smoothness measure for frame fields simplifies to the quartic energy

$$\begin{aligned} E_{\text{smooth}} &= \sum_{(f,g)\in\mathcal{E}} \left| (\alpha_f^2+\beta_f^2)(\bar{e}_f)^2 - (\alpha_g^2+\beta_g^2)(\bar{e}_g)^2 \right|^2 \\ &+ \sum_{(f,g)\in\mathcal{E}} \left| \alpha_f^2\beta_f^2(\bar{e}_f)^4 - \alpha_g^2\beta_g^2(\bar{e}_g)^4 \right|^2. \end{aligned}$$

Singularities. To recover the topology of a frame field defined by its polynomial coefficients x_0 and x_2, explicit matchings are inevitable. The authors show that it is possible to create a related cross field which is known to have the same topology as the frame field. Position and degree of singularities can then be obtained from this cross field.

Geometrically, each of these crosses is created by the bisectors of the corresponding frame. So in a singe triangle, all four vectors $\{\alpha, \beta, -\alpha, -\beta\}$ are ordered by their angle and for each pair of adjacent vectors a new one with their mean angle and unit length is created. The four resulting vectors are always orthogonal and form the bisector cross for this face. Mathematically, the authors show that this is (up to magnitude) the field which is defined by the coefficient $x_0 = \alpha^2\beta^2$ alone.

Having constructed the bisector cross field, so called *principal matchings* can be chosen explicitly across each edge by transporting one cross to the local coordinate system of its neighbor and pairing vectors which are closest to each other.

Finally, the singularity index of a vertex can be obtained by choosing a cross vector of an incident face and following the matchings of the dual 1-ring loop around the vertex. When reaching the source face again and the vector has been rotated counter-clockwise by k multiples of 90°, its index is $\frac{k}{4}$.

3 Integrability

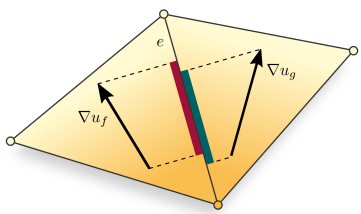
The first challenge—computing smooth frame fields without integer variables—can now be solved as seen in the previous section. In [Dia+15] the problem that these fields do not directly induce a parametrization is addressed. As stated in Section 1.3, in general, there is no parametrization whose gradient vectors fields are exactly aligned with such a given frame field. As a consequence common methods solve a Poisson equation to find a parametrization with a similar gradient. A major drawback of splitting the whole problem into these two subsequent steps is that singularities are completely determined within the first step, without any knowledge about the second one. Thus, constraints imposed during the Poisson step cannot affect the placement of singularities this way.

The authors of [Dia+15] aim for a solution in which the frame field computed in the first step is integrable. This means that it can be decomposed into two consistent vector fields which actually yield a valid parametrization by simple integration, ideally making the second step dispensable. For this purpose they investigate conditions on frame fields that assure their integrability and express them in terms of discrete curl. Sticking to the polynomial representation again allows to formulate all necessary equations without the use of explicit matchings.

To tackle the resulting problem computationally, they propose a non-linear optimization transforming an initial smooth frame field into a similar field with low curl using penalty terms. During this optimization singularities are free to change. Since the iterative procedure does not produce fields whose curl is numerically zero, the following Poisson step still has to be performed. However, its error is low as demonstrated at a large number of examples.

In addition they impose penalty terms on the frame field optimization preventing triangle flips in the eventual parametrization. Other ingredients to obtain an integer-grid map [Bom+13], namely integer transitions at cuts and singularities on integer locations, are not included by design and have to be achieved by subsequently applying a rounding strategy.

3.1 Discrete Curl



A piecewise linear parametrization of a triangle mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$ consists of two scalar functions defined at each vertex. To obtain values of u and v inside triangles, the values at the three incident vertices are linearly interpolated. The gradients of the parametrization are piecewise constant tangent vector fields $\nabla u, \nabla v \in \mathbb{R}^2$ defined per face $f \in \mathcal{F}$. Clearly, values of u and v on an edge are the same, no matter in which incident triangle f or g they are interpolated (assuming there is no cut). Thus, the directional derivative along the edge e is the same, too. Expressed in terms of the gradient this means that the projection of ∇u_f on e is the same as the projection of ∇u_g on e, i.e.

$$\langle \nabla u_f, e \rangle = \langle \nabla u_g, e \rangle.$$

"Integrable ⇒ Curl-Free". In [PP02] $\langle \nabla u_f, e \rangle - \langle \nabla u_g, e \rangle$ is defined as a measure for the *discrete curl* of the vector field ∇u across edge e. As seen above, this is always zero for the gradient of any piecewise linear scalar function defined on the vertices of $\mathcal{M}$.

"Curl-Free ⇒ Integrable". More interesting is the opposite direction: the authors of [PP02] also prove that a curl-free vector field always is the gradient of a scalar function, however under a condition. For this statement to be true, the domain on which the field is defined has to be simply-connected. For a discrete triangle mesh, this means that it contains no handles, i.e. it is of genus zero or has disk topology. This condition can always be satisfied by cutting the mesh to a single patch which is topologically equivalent to a disk. As a result, the integrability of a vector field can be expressed directly via its discrete curl.

Integrating Curl-Free Frame Fields. The above concept can be transferred from vector fields to frame fields. A frame field is curl free if across every edge there is a matching such that each pair of matched vectors has zero curl. Given a curl free frame field, integrating it poses one additional issue. It has to be separated into two consistently oriented vector fields ∇u and ∇v. However, locally it is not clear which vectors from $\{\alpha_f, \beta_f, -\alpha_f, -\beta_f\}$ represent ∇u_f and ∇v_f respectively. If there are no singularities, it is possible to pick an arbitrary triangle f, pick an arbitrary frame vector to be ∇u_f and the next vector in counter-clockwise direction to be ∇v_f. Now this choice can be consistently propagated along a dual spanning tree using the curl-free matchings across each edge.

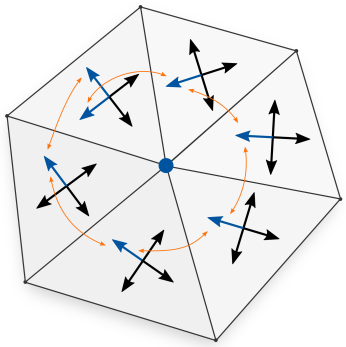


Figure 3: Inconsistent matchings around a singularity.

Yet, in the presence of singularities this procedure is not possible. As shown in Section 2.3, the one-ring matchings around a singular vertex are *inconsistent*. Specifically, this means that if a vector α in an incident face is chosen to be ∇u_f and this choice is propagated around the singularity until reaching the same face again, ∇u_f might now be associated with a different vector (see Figure 3). The problem can be solved by cutting the mesh to disk topology such that all the singularities lie on its boundary, as done in [BZK09] (see Figure 4). It is then possible to consistently separate the frame field into two vector fields as described above. Since the whole surface now consists of a single patch it is simply connected. Thus, if the frame field is completely curl-free, ∇u and ∇v actually are the gradients of scalar functions u and v. These can then be obtained up to a constant by integration.

Across the seams at which the mesh has been cut, the resulting parametrization (u, v) is related only by a permutation and an additive factor. For the purpose of quad meshing, only the integer grid formed by the isolines of the parametrization is of interest. A cyclic permutation of $\{u, v, -u, -v\}$ rotates this grid by integer multiples of 90° which has no effect on the eventual quad structure. The additive factor however is not restricted to full integers, which produces visible seams in the parametrization.

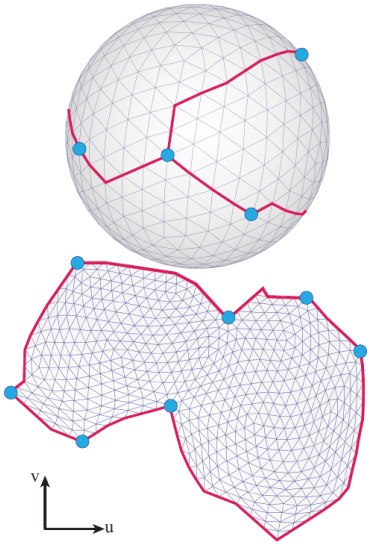


Figure 4: A mesh is cut to a topological disk such that all singularities lie on its boundary. Figure from [Dia+15].

In summary, entirely curl-free frames fields can be directly turned into a parametrization by simple integration. The resulting parametrization is seamless only in its rotational aspect, not in its translational part.

Curl via Real Polynomials. The above definition of discrete curl for frame fields relies on matchings between adjacent frames. Fortunately, it is also possible to formulate it in a complex polynomial representation. As done in Section 2.1 for smoothness, this makes it possible to factor out the role of matchings.

As before, e_f and e_g denote the normalized edge vector expressed in the bases of face f and g respectively. So matched vectors α_f and $\alpha_g \in \mathbb{C}$ in adjacent faces can be transformed from their local bases into a common coordinate system with the shared edge e as the real axis by writing $\alpha_f \bar{e}_f$ and $\alpha_g \bar{e}_g$. The real component of the transformed vectors is now exactly their projection on e (see Figure 5). So the discrete curl

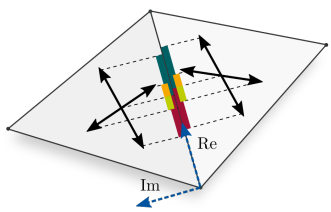


Figure 5: Adjacent frames are curl-free if the set of projections onto to common edge is the same for both frames.

$\langle \alpha_f, e \rangle - \langle \alpha_g, e \rangle$ can also be written as

$$\mathrm{Re}(\alpha_f \bar{e}_f) - \mathrm{Re}(\alpha_g \bar{e}_g).$$

When dealing with frame fields, there are four vectors per face each of which is projected to e. It is again possible to construct a—in this case real—polynomial having these projections as roots:

$$\begin{aligned} P_{f|e}(r) &= (r - \mathrm{Re}(\alpha_f \bar{e}_f)) \cdot (r - \mathrm{Re}(\beta_f \bar{e}_f)) \cdot (r - \mathrm{Re}(-\alpha_f \bar{e}_f)) \cdot (r - \mathrm{Re}(-\beta_f \bar{e}_f)) \\ &= (r^2 - \mathrm{Re}(\alpha_f \bar{e}_f)^2) \cdot (r^2 - \mathrm{Re}(\alpha_g \bar{e}_g)^2) \\ &= r^4 - (\mathrm{Re}(\alpha_f \bar{e}_f)^2 + \mathrm{Re}(\beta_f \bar{e}_f)^2) r^2 + \mathrm{Re}(\alpha_f \bar{e}_f)^2 \, \mathrm{Re}(\beta_f \bar{e}_f)^2. \end{aligned}$$

Again, the fourth order coefficient is one by construction, and the third and first order coefficients vanish. So there are only two non-trivial coefficients:

$$\begin{aligned} c_{f|e,2} &= -(\mathrm{Re}(\alpha_f \bar{e}_f)^2 + \mathrm{Re}(\beta_f \bar{e}_f)^2) \\ c_{f|e,0} &= \mathrm{Re}(\alpha_f \bar{e}_f)^2 \, \mathrm{Re}(\beta_f \bar{e}_f)^2. \end{aligned}$$

If these two coefficients are are equal for neighboring faces across their common edge, i.e. $c_{f|e,2} = c_{g|e,2}$ and $c_{f|e,0} = c_{g|e,0}$, both polynomials are equal and thus have the same roots. This means that the set of projections on e is the same for both frames. Hence, the frame field is curl free across this edge. Note that using this formulation two sets of projections are compared without ever comparing any pair of projections explicitly.

The deviation of two adjacent frames from being curl-free is measured by the squared distance of the coefficients. This per-edge energy is called the PolyCurl:

$$PolyCurl_e = (c_{f|e,2} - c_{g|e,2})^2 + (c_{f|e,0} - c_{g|e,0})^2.$$

Compared to an energy that measures the difference of matched projections explicitly, the PolyCurl expresses this difference only indirectly. The explicit energy is zero if and only if the PolyCurl is zero. However in the vicinity of this minimum, both formulations are simply assumed to behave similarly. The optimization in Section 3.3 aims to eliminate the PolyCurl across every edge.

3.2 Inversions

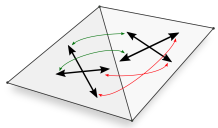
Not all parametrization functions u and v on a surface actually imply a valid quad mesh. [Bom+13] defines three conditions on parametrizations restricting them to the class of *integer-grid maps*. These are: (1) across a cut, u and v are related by an integer-grid automorphism, (2) singularities lie on integer locations, (3) triangles are consistently oriented in the parameter domain, i.e. there are no triangle flips. While (1) and (2) cannot be fully addressed by conditions on frame fields, the authors propose a way to optimize for (3).

Triangle Flips. The gradients ∇u and ∇v have to be defined consistently (see Section 3.1). Commonly their order is chosen to be counter-clockwise, i.e. ∇u and ∇v locally span a convex angle of less than 180°. If for a triangle f this is not the case, the image of this triangle inside the parameter domain flips its orientation. In this case the isolines of the parametrization may not form a quad mesh on the surface. For example faces with other than four edges are possible (c.f. [Bom+13]). The condition on ∇u and ∇v to induce a flip-free parametrization hence is

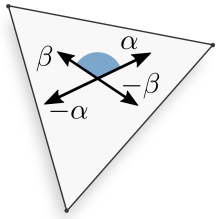
$$\det(\nabla u^T, \nabla v^T) > 0.$$

This way, the cross product of the two gradient vectors always points into the same direction as the surface normal. A parametrization that does not contain triangle flips is also called *inversion-free* or *locally injective*.

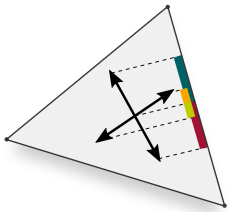
Order-Preserving Matchings. Applied to the setting of curl-free frame fields, this condition is satisfied in practice if the curl-free matchings are order-preserving. This means that for adjacent faces f and g the matching, which is induced by the projections of the frame vectors onto the common edge, is only allowed to be a cyclic permutation. Thus only four valid matchings are possible for each edge. Since all above formulations are completely invariant to any kind of ordering among the vectors of a frame, this property is likely to be violated (see inset).



Geometrically-Ordered Vectors. In the following, the restriction to the four valid matchings will be formulated in the complex setting. For this to be possible, a geometrical order of the frame vectors inside each face has to be established. This means that $\alpha_f, \beta_f, -\alpha_f, -\beta$ are ordered counter-clockwise, i.e. α_f and β_f span a convex angle. In complex notation, this can be expressed as $\mathrm{Im}(\beta_f \bar{\alpha}_f) > 0$, which constrains their angle difference to be strictly between 0° and 180° (see inset). Geometrically-ordered vectors form a precondition for the following expressions.



PolyQuotient. Assuming vanishing PolyCurl and geometrically-ordered roots in each face, a formulation has to be found that distinguishes frame fields inducing order-preserving matchings from other frame fields. However, this has to be expressed indirectly since explicit matchings are not known during the optimization. The authors propose to consider each pair of consecutive vectors within a frame and the ratio of their projections onto an edge:



$$q_{f|e}(\alpha_f, \beta_f) = \frac{\mathrm{Re}(\alpha_f \bar{e}_f)}{\mathrm{Re}(\beta_f \bar{e}_f)}, \qquad q_{f|e}(\beta_f, -\alpha_f) = \frac{\mathrm{Re}(\beta_f \bar{e}_f)}{\mathrm{Re}(-\alpha_f \bar{e}_f)},$$

$$q_{f|e}(-\alpha_f, -\beta_f) = \frac{\mathrm{Re}(-\alpha_f \bar{e}_f)}{\mathrm{Re}(-\beta_f \bar{e}_f)}, \qquad q_{f|e}(-\beta_f, \alpha_f) = \frac{\mathrm{Re}(-\beta_f \bar{e}_f)}{\mathrm{Re}(\alpha_f \bar{e}_f)}.$$

These are either equal, e.g. $q_{f|e}(\alpha_f, \beta_f) = q_{f|e}(-\alpha_f, -\beta_f)$, or the negative reciprocal of each other, e.g. $q_{f|e}(\alpha_f, \beta_f) = -q_{f|e}(\beta_f, -\alpha_f)^{-1}$. For a curl-free matching that is order-preserving, the same relation holds across an edge, i.e. either $q_{f|e}(\alpha_f, \beta_f) = q_{g|e}(\alpha_g, \beta_g)$ or $q_{f|e}(\alpha_f, \beta_f) = -q_{g|e}(\alpha_g, \beta_g)^{-1}$. The authors state that all other matchings are related differently, i.e by either reciprocal only or negative only.

Thus, enforcing the above relation between every pair or adjacent frames yields order-preserving matchings. To model the disjunction (equality or negative reciprocal) two polynomials are set up per edge. In the following $q_{f|e}(\alpha_f, \beta_f)$ and $q_{g|e}(\alpha_g, \beta_g)$ are abbreviated as $q_{f|e}$ and $q_{g|e}$ respectively. The real polynomial

$$\begin{aligned} Q_{f|e}(r) &= (r - q_{f|e}) \cdot (r + q_{f|e}^{-1}) \\ &= r^2 + (q_{f|e}^{-1} - q_{f|e})r - 1 \end{aligned}$$

has $q_{f|e}$ and $q_{f|e}^{-1}$ as its roots. Thus, if two polynomials $Q_{f|e}(r)$ and $Q_{g|e}(r)$ across an edge are equal—i.e. have the same roots—either $q_{f|e} = q_{g|e}$ or $q_{f|e} = -q_{g|e}^{-1}$ holds.

As before this can be expressed by the polynomial coefficients instead of the actual values. Each polynomial $Q_{f|e}(r)$ is of degree two and has only a single non-constant coefficient, namely $q_{f|e}^{-1} - q_{f|e}$. The difference between these coefficients across an edge is called the PolyQuotient

$$\mathrm{PolyQuotient}_e = (q_{f|e}^{-1} - q_{f|e}) - (q_{g|e}^{-1} - q_{g|e}).$$

The fractions contained in $q_{f|e}$ and $q_{g|e}$ may cause divisions by zero. To prevent this, the condition of a vanishing PolyQuotient can be reformulated by eliminating common denominators:

$$\mathrm{PolyQuotient}_e = \left(\frac{\mathrm{Re}(\beta_f \bar{e}_f)}{\mathrm{Re}(\alpha_f \bar{e}_f)} - \frac{\mathrm{Re}(\alpha_f \bar{e}_f)}{\mathrm{Re}(\beta_f \bar{e}_f)}\right) - \left(\frac{\mathrm{Re}(\beta_g \bar{e}_g)}{\mathrm{Re}(\alpha_g \bar{e}_g)} - \frac{\mathrm{Re}(\alpha_g \bar{e}_g)}{\mathrm{Re}(\beta_g \bar{e}_g)}\right) \overset{!}{=} 0$$

$$\Leftrightarrow$$

$$\underbrace{\left(\mathrm{Re}(\beta_f \bar{e}_f)^2 - \mathrm{Re}(\alpha_f \bar{e}_f)^2\right) \mathrm{Re}(\alpha_g \bar{e}_g)\, \mathrm{Re}(\beta_g \bar{e}_g)}_{=:\, \hat{q}_1} - \underbrace{\left(\mathrm{Re}(\beta_g \bar{e}_g)^2 - \mathrm{Re}(\alpha_g \bar{e}_g)^2\right) \mathrm{Re}(\alpha_f \bar{e}_f)\, \mathrm{Re}(\beta_f \bar{e}_f)}_{=:\, \hat{q}_2} \overset{!}{=} 0$$

Ultimately, the squared difference $(\hat{q}_1 - \hat{q}_2)^2$ is used as a measure for the PolyQuotient per edge, because it is zero if and only if the PolyQuotient is zero.

If the frame vectors in each face are ordered geometrically and the PolyQuotient vanishes, a curl-free frame field induces order-preserving matchings, which prevents inverted triangles.

3.3 Optimization

The results explained so far constitute a wish list of desired frame field properties. A frame field should

- be as smooth as possible,
- be curl-free,
- have geometrically ordered vectors in each face,
- induce order-preserving matchings,
- adhere to a set of provided constraints.

Because in general total smoothness cannot be achieved, this can be regarded as a soft condition while all other properties are intended to be hard criteria. Each property can already be expressed in a formulation that works without the use of integer variables, resulting in continuous optimization only. Unfortunately the resulting terms are highly non-linear and non-convex which makes finding the global optimum practically impossible. For this reason, the authors set up a single energy term which is a weighted sum of all desired properties. To minimize the energy, a frame field that is smooth only (see Section 2.3) is used as an initial guess. From there on, Gauss-Newton steps are performed to find a nearby local optimum. The desired hard conditions are only incorporated as penalty terms and thus might not be satisfied exactly.

Variables. Since each frame $\{\alpha_f, \beta_f, -\alpha_f, -\beta_f\}$ is completely determined by the complex numbers α_f and β_f, there are four real-valued unknowns per face. They are referred to as

$$z_f = [\mathrm{Re}(\alpha_f), \mathrm{Im}(\alpha_f), \mathrm{Re}(\beta_f), \mathrm{Im}(\beta_f)]^T.$$

The full variable vector representing the entire frame field is called z and contains $4 \cdot |\mathcal{F}|$ real numbers.

Energy. The full energy term is defined as

$$\begin{aligned} E(z) = & \sum_{(f,g)=e\in\mathcal{E}} \mathcal{S}(z_f, z_g) \;+\; \mathcal{P}(z_f, z_g) \;+\; \mathcal{Q}(z_f, z_g) \\ + & \sum_{f\in\mathcal{F}} \mathcal{B}(z_f) \;+\; \mathcal{C}(z_f) \end{aligned}$$

consisting of terms measuring smoothness, PolyCurl, PolyQuotient, geometric order and deviation from given constraints, respectively.

Smoothness. The smoothness is measured as in Section 2.3 by comparing the squared distance between transported complex polynomial coefficients and an additional weighting factor w_s:

$$\begin{aligned} \mathcal{S}(z_f, z_g) &= w_s \left| x_{f,2}(\bar{e}_f)^2 - x_{g,2}(\bar{e}_g)^2 \right|^2 \\ &+ w_s \left| x_{f,0}(\bar{e}_f)^4 - x_{g,0}(\bar{e}_g)^4 \right|^2 \end{aligned}$$

with

$$\begin{aligned} x_{f,2} &= \alpha_f^2 + \beta_f^2 \\ x_{f,0} &= \alpha_f^2 \beta_f^2. \end{aligned}$$

Edge vectors e_f, e_g are assumed to be normalized.

PolyCurl. As in Section 3.1 the PolyCurl is expressed as the squared difference between the coefficients of real polynomials:

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

with

$$\begin{aligned}c_{f|e,2} &= -(\mathrm{Re}(\alpha_f \bar{e}_f)^2 + \mathrm{Re}(\beta_f \bar{e}_f)^2) \\ c_{f|e,0} &= \mathrm{Re}(\alpha_f \bar{e}_f)^2\, \mathrm{Re}(\beta_f \bar{e}_f)^2.\end{aligned}$$

Here, both coefficients are weighted differently to compensate for their different magnitudes.

Geometric Order. To ensure geometrically—i.e. counter-clockwise—oriented vectors, $\mathrm{Im}(\beta_f \bar{\alpha}_f) > 0$ has to hold per face (see Section 3.2). The authors propose to use a barrier term that only effects the energy if this value is below a threshold s. The barrier function ϕ_s is zero for values greater than s and ∞ for values less or equal to s. In-between, the shifted reciprocal of a cubic polynomial is used to model a smoothly increasing barrier (see Figure 6):

$$\phi_s(x) = \begin{cases} \infty & x \leq s \\ \frac{1}{b_s(x)} - 1 & 0 < x \leq s \\ 0 & x \geq s \end{cases}$$

with

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

Thus:

$$\mathcal{B}(z_f) = w_b[\phi_s(\mathrm{Im}(\beta_f \bar{\alpha}_f))]^2.$$

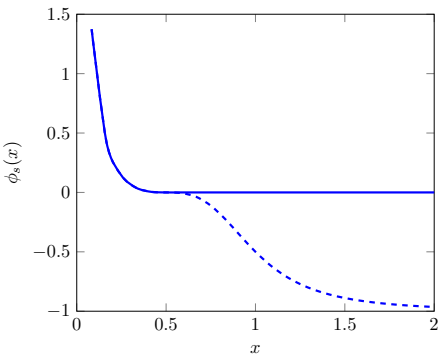


Figure 6: The barrier function $\phi_s(x)$ (solid) smoothly increases from zero to ∞ as x approaches zero. The point of impact is controlled by the position s, at which $\frac{1}{b_s(x)} - 1$ (dashed) has an inflection point, here $s = 0.5$.

PolyQuotient. The PolyQuotient is measured as derived in Section 3.2:

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

with

$$\begin{aligned}\hat{q}_1 &= \left(\mathrm{Re}(\beta_f \bar{e}_f)^2 - \mathrm{Re}(\alpha_f \bar{e}_f)^2\right) \mathrm{Re}(\alpha_g \bar{e}_g)\, \mathrm{Re}(\beta_g \bar{e}_g) \\ \hat{q}_2 &= \left(\mathrm{Re}(\beta_g \bar{e}_g)^2 - \mathrm{Re}(\alpha_g \bar{e}_g)^2\right) \mathrm{Re}(\alpha_f \bar{e}_f)\, \mathrm{Re}(\beta_f \bar{e}_f).\end{aligned}$$

Constraints. The deviation from given frames in constrained faces is also measured by a squared distance. In unconstrained faces, the deviation from the initial smooth frame field is added as regularization:

$$\mathcal{C}(z_f) = \begin{cases} w_{cn}||z_f - z_f^{cn}||^2 & f \text{ is constrained} \\ w_r||z_f - z_f^{prev}||^2 & f \text{ is free} \end{cases}$$

It is also possible to use partial constrains. In this case only α_f is constrained without having to choose if it corresponds to ∇u, ∇v, $-\nabla u$ or $-\nabla v$.

Weights. The authors propose the following weights which were chosen during their experiments:

$$\begin{aligned} w_s &= 1 && \text{(Smoothness)} \\ w_p &= 10 && \text{(PolyCurl)} \\ w_q &= 10 && \text{(PolyQuotient)} \\ w_b &= 0.001 && \text{(Geometric order)} \\ s &= 0.5 && \text{(Barrier threshold)} \\ w_{cn} &= 10 && \text{(Constraints)} \\ w_r &= 0.001 && \text{(Regularization)} \end{aligned}$$

Since the smoothness term might prevent the PolyCurl and PolyQuotient from approaching zero, w_s is divided by two each five iterations.

Optimization. The initial solution $z^{(0)}$ is computed by solving a linear system as described in Section 2.3. Because the energy function is a sum of squared terms, i.e. $E(z) = \sum E_i(z)^2$, the Gauss-Newton algorithm can be applied. This requires setting up a Jacobian matrix and solving a linear system in each step. However, no Hessian matrix is necessary.

A single iteration is an update of the form $z^{(k+1)} = z(k) - \gamma^{(k)}\delta^{(k)}$, where $\delta^{(k)}$ is a direction in the $4|\mathcal{F}|$-dimensional solution space, and $\gamma^{(k)}$ a step size. The direction $\delta^{(k)}$ is obtained by solving the linear system

$$J\left(z^{(k)}\right)^T J\left(z^{(k)}\right)\delta^{(k)} = J\left(z^{(k)}\right)^T \mathbf{E}\left(z^{(k)}\right).$$

Here, $J(z^{(k)})$ is the Jacobian matrix and $\mathbf{E}(z^{(k)})$ the concatenation of all squared terms at iteration k. The step size $\gamma^{(k)}$ for a given direction is then found by backtracking line search.

The authors propose to stop the iterations if there are no flipped triangles and the average error of the Poisson step that integrates u and v is lower than 0.001.

4 Results & Conclusion

Two challenges in guiding field design for parametrization-based quad meshing are computing fields that are integrable and avoiding hard integer optimization. [Dia+14] proposes a way to compute smooth frame fields without integer variables by solving a linear system. [Dia+15] builds upon this idea and finds a formulation for frame fields that are integrable and inversion-free. The formulation is also free of integer variables but results in non-linear optimization.

Convergence. Due to the non-convexity of the problem, the optimal solution cannot be found in practice. Furthermore, there is no proof of convergence for the optimization procedure described in Section 3.3. Moreover, if it converges to a local minimum, there are no guarantees concerning its quality. Thus, the success of the algorithm is only demonstrated empirically. The method was applied to a collection of 116 meshes and in all cases an inversion-free parametrization with an average Poisson residual of less than 0.005 was obtained.

Performance. The authors provide two performance benchmarks. In the first one, the algorithm is stopped as soon as no inverted elements are present. Timings reach from 0.01148 minutes (10402 triangles) to 40.7 minutes (105022 triangles). However, the second slowest example took only 3.718 minutes (100000 triangles). In many cases no iterations were performed because the initial frame field is already inversion-free.

In the second benchmark, the iterations are stopped if the average Poisson error falls below 0.005. Results reach from 0.01323 minutes (480 triangles) to 142.7 minutes (100000 triangles).

Although integer optimization is avoided due to its bad performance, the resulting non-linear optimization is slow compared to state of the art algorithms producing (non-integrable) fields via greedy rounding of integer variables.

Integration. A major motivation for the computation of integrable vector fields is to replace the subsequent Poisson step by simple integration. However, since [Dia+15] does not enforce integrability as a hard constraint it is only achieved approximately. Thus, the Poisson step still has to be performed, but its error is greatly reduced.

Singularity Optimization. Common parametrization methods, i.e. [BZK09] perform a two-step optimization. Here, the position and degree of singularities is completely determined in the first step, which creates a smooth, non-integrable guiding field. The following parametrization step implies further conditions—namely that its gradient is a curl-free vector field—and might benefit from changing the singularity configuration, which is not possible. In [Dia+15], both smoothness and integrability are optimized in a single problem instance and can influence each other.

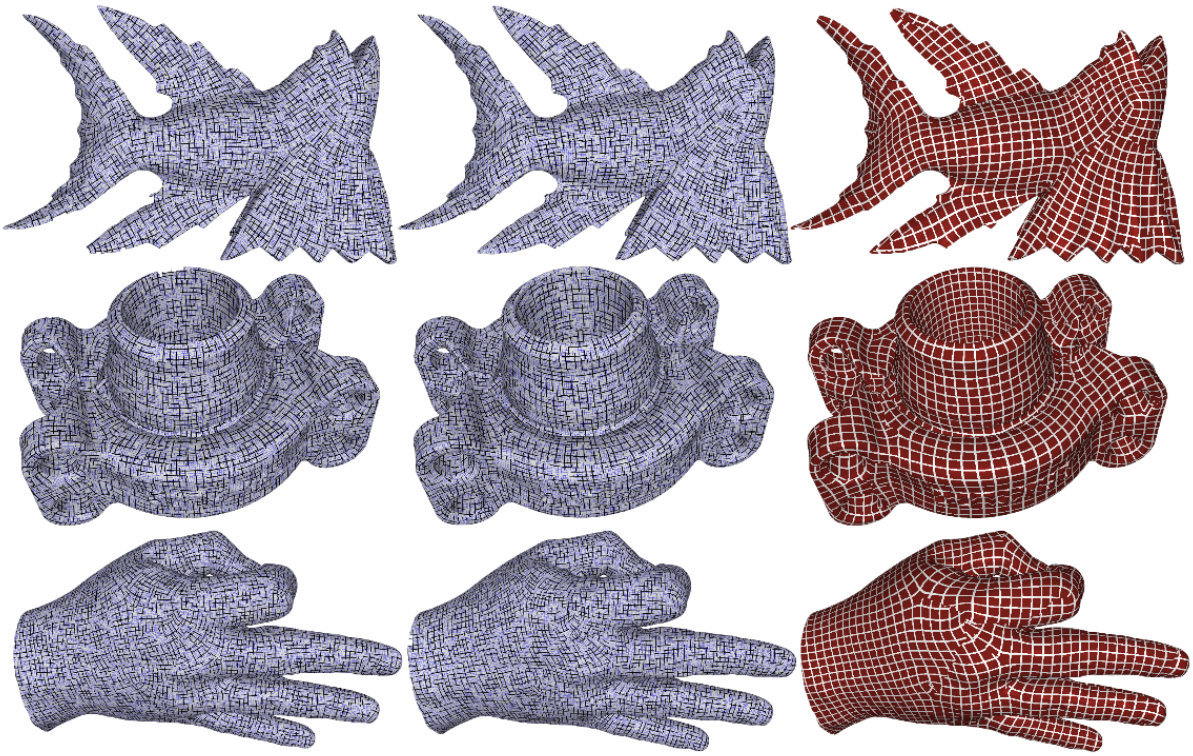


Figure 7: Left: smooth frame fields used as initial solution. Middle: results of the optimization. Right: the parametrizations obtained by the Poisson step. For all meshes, a parametrization with an average Poisson error of less than 0.005 and no flipped triangles was obtained. Run times for the frame field optimization are 13.08 minutes (top, 34910 triangles), 17.61 minutes (middle, 68580 triangles) and 23.69 minutes (bottom, 99999 triangles). Figure from [Dia+15].

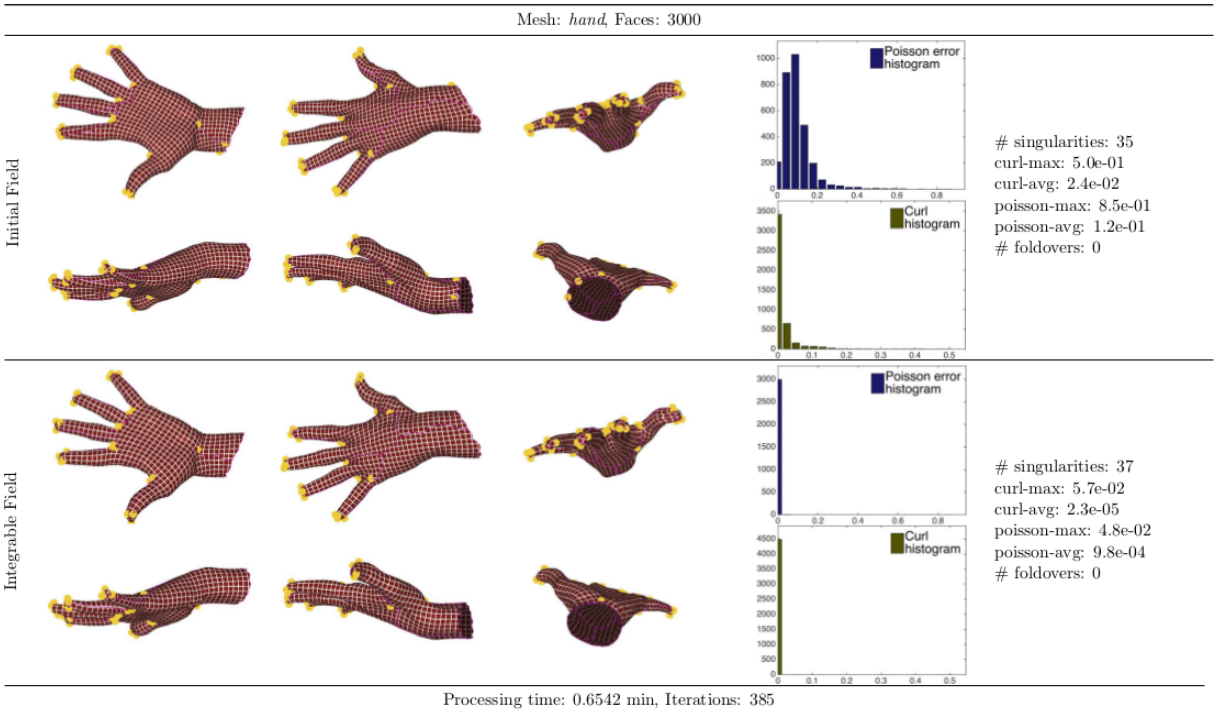


Figure 8: The average Poisson error of a smooth frame field is reduced from 0.12 to 0.001 in 39 seconds (3000 triangles). Figure from [Dia+15].

Integer Rounding. The parametrizations obtained from curl-free frame fields by (approximate) integration exhibit discontinuities across cuts. While the rotational component of the transition function relating both sides of a cut is restricted to integer multiples of 90°, the translational component is continuous. Thus, there are visible seams along cuts. Singularities are also positioned continuously inside the parameter domain and are not restricted to lie at integer coordinates, which results in irregular faces instead of vertices. Both problems have to be solved by subsequently snapping continuous values to integers. The authors apply the iterative rounding scheme from [BZK09]. More recently, [CBK15] proposes a novel greedy strategy tailored specifically to this situation.

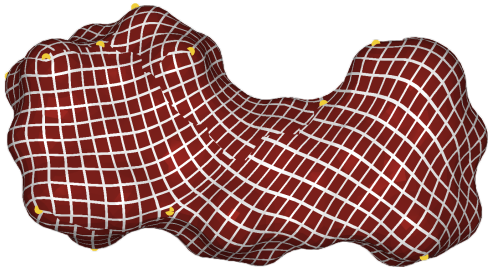


Figure 9: The resulting parametrization has non-integer translations at cuts and singularities located at non-integer positions. Figure from [Dia+15].

The authors of [Dia+15] are the first who directly optimize for curl-free, i.e. integrable vector fields. They obtain reasonable results which can—after subsequent integer rounding—be used for quad meshing. With this, they demonstrate that computing integrable fields is a promising direction towards overcoming the two-step nature of current parametrization algorithms. However, this goal is not yet achieved completely because the integer rounding is still subject to a second step. An interesting research question is whether conditions for valid translations at seams can be incorporated into the optimization. Moreover, it is worth investigating if formulations for curl-free vector fields without explicit matchings exist that do not result in non-convex optimization. Alternatively, improving algorithms resorting to integer variables is another possible direction, since there might be more sophisticated strategies than currently known for optimizing frame fields with explicit matchings.

References

[Ang11] Sigurd Angenent. "Complex Numbers and the Complex Exponential". In: *Math 222 Second Semester Calculus Lecture Notes*. 2011. Chap. 3, pp. 56–73.

[Bom+12] D. Bommes et al. "State of the Art in Quad Meshing". In: *Eurographics STARS*. 2012.

[Bom12] David Bommes. "Quadrilateral Surface Mesh Generation for Animation and Simulation". Dissertation. RWTH Aachen, 2012. URL: `http://darwin.bth.rwth-aachen.de/opus3/volltexte/2013/4373/`.

[BZK09] David Bommes, Henrik Zimmer, and Leif Kobbelt. "Mixed-integer Quadrangulation". In: *ACM Trans. Graph.* 28.3 (July 2009), 77:1–77:10. ISSN: 0730-0301. DOI: `10.1145/1531326.1531383`. URL: `http://doi.acm.org/10.1145/1531326.1531383`.

[Bom+13] David Bommes et al. "Integer-grid Maps for Reliable Quad Meshing". In: *ACM Trans. Graph.* 32.4 (July 2013), 98:1–98:12. ISSN: 0730-0301. DOI: `10.1145/2461912.2462014`. URL: `http://doi.acm.org/10.1145/2461912.2462014`.

[CBK15] Marcel Campen, David Bommes, and Leif Kobbelt. "Quantized Global Parametrization". In: *ACM Trans. Graph.* 34.6 (Oct. 2015), 192:1–192:12. ISSN: 0730-0301. DOI: `10.1145/2816795.2818140`. URL: `http://doi.acm.org/10.1145/2816795.2818140`.

[Cra+13] Keenan Crane et al. "Digital Geometry Processing with Discrete Exterior Calculus". In: *ACM SIGGRAPH 2013 courses*. SIGGRAPH '13. Anaheim, California: ACM, 2013.

[Dia+14] Olga Diamanti et al. "Designing N-PolyVector Fields with Complex Polynomials". In: *Computer Graphics Forum (proceedings of EUROGRAPHICS Symposium on Geometry Processing)* 33.5 (2014), pp. 1–11.

[Dia+15] Olga Diamanti et al. "Integrable PolyVector Fields". In: *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)* 34.4 (2015), 38:1–38:12.

[Ebk+14] Hans-Christian Ebke et al. "Level-of-detail Quad Meshing". In: *ACM Trans. Graph.* 33.6 (Nov. 2014), 184:1–184:11. ISSN: 0730-0301. DOI: `10.1145/2661229.2661240`. URL: `http://doi.acm.org/10.1145/2661229.2661240`.

[Ebk+13] Hans-Christian Ebke et al. "QEx: Robust Quad Mesh Extraction". In: *ACM Trans. Graph.* 32.6 (Nov. 2013), 168:1–168:10. ISSN: 0730-0301. DOI: 10.1145/2508363.2508372. URL: http://doi.acm.org/10.1145/2508363.2508372.

[Fie03] Miroslav Fiedler. "A note on companion matrices". In: *Linear Algebra and its Applications* 372 (2003), pp. 325 –331. ISSN: 0024-3795. DOI: http://dx.doi.org/10.1016/S0024-3795(03)00548-2. URL: http://www.sciencedirect.com/science/article/pii/S0024379503005482.

[HZ00] Aaron Hertzmann and Denis Zorin. "Illustrating Smooth Surfaces". In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '00. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 517–526. ISBN: 1-58113-208-5. DOI: 10.1145/344779.345074. URL: http://dx.doi.org/10.1145/344779.345074.

[Jia+15] Tengfei Jiang et al. "Frame Field Generation Through Metric Customization". In: *ACM Trans. Graph.* 34.4 (July 2015), 40:1–40:11. ISSN: 0730-0301. DOI: 10.1145/2766927. URL: http://doi.acm.org/10.1145/2766927.

[KNP07] Felix Kälberer, Matthias Nieser, and Konrad Polthier. "QuadCover – Surface Parameterization using Branched Coverings". In: *Computer Graphics Forum* 26.3 (2007), pp. 375–384.

[MPZ14] Ashish Myles, Nico Pietroni, and Denis Zorin. "Robust Field-aligned Global Parametrization". In: *ACM Trans. Graph.* 33.4 (July 2014), 135:1–135:14. ISSN: 0730-0301. DOI: 10.1145/2601097.2601154. URL: http://doi.acm.org/10.1145/2601097.2601154.

[PZ07] Jonathan Palacios and Eugene Zhang. "Rotational Symmetry Field Design on Surfaces". In: *ACM Trans. Graph.* 26.3 (July 2007). ISSN: 0730-0301. DOI: 10.1145/1276377.1276446. URL: http://doi.acm.org/10.1145/1276377.1276446.

[Pan+14] Daniele Panozzo et al. "Frame Fields: Anisotropic and Non-orthogonal Cross Fields". In: *ACM Trans. Graph.* 33.4 (July 2014), 134:1–134:11. ISSN: 0730-0301. DOI: 10.1145/2601097.2601179. URL: http://doi.acm.org/10.1145/2601097.2601179.

[PP02] Konrad Polthier and Eike Preuß. "Identifying Vector Field Singularities Using a Discrete Hodge Decomposition". In: Springer Verlag, 2002, pp. 112–134.

[Ray+09] Nicolas Ray et al. "Geometry-aware Direction Field Processing". In: *ACM Trans. Graph.* 29.1 (Dec. 2009), 1:1–1:11. ISSN: 0730-0301. DOI: 10.1145/1640443.1640444. URL: http://doi.acm.org/10.1145/1640443.1640444.

[Ray+08] Nicolas Ray et al. "N-symmetry Direction Field Design". In: *ACM Trans. Graph.* 27.2 (May 2008), 10:1–10:13. ISSN: 0730-0301. DOI: 10.1145/1356682.1356683. URL: http://doi.acm.org/10.1145/1356682.1356683.

[Ray+06] Nicolas Ray et al. "Periodic Global Parameterization". In: *ACM Trans. Graph.* 25.4 (Oct. 2006), pp. 1460–1485. ISSN: 0730-0301. DOI: 10.1145/1183287.1183297. URL: http://doi.acm.org/10.1145/1183287.1183297.

[Sch14] Patrick Schmidt. "Towards Interactive Quadrangulation Using Local Remeshing". Bachelor's Thesis. Visual Computing Institute, RWTH Aachen, 2014.