Practical 3D Frame Field Generation

Nicolas Ray* Dmitry Sokolov† Bruno Lévy‡

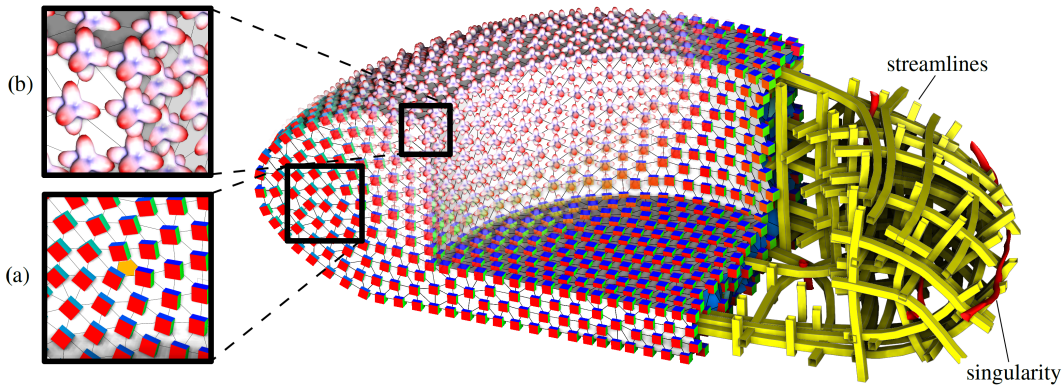

Figure 1: *Our algorithm produces smooth frame fields in volumes. Frames (a) are represented by spherical harmonic functions (b), attached to each vertex of a tetrahedral mesh. Streamlines and singularities of the field are shown in yellow and red, respectively.*

Abstract

Given a tetrahedral mesh, the algorithm described in this article produces a smooth 3D frame field, i.e. a set of three orthogonal directions associated with each vertex of the input mesh. The field varies smoothly inside the volume, and matches the normals of the volume boundary. Such a 3D frame field is a key component for some hexahedral meshing algorithms, where it is used to steer the placement of the generated elements.

We improve the state-of-the art in terms of quality, efficiency and reproducibility. Our main contribution is a non-trivial extension in 3D of the existing least-squares approach used for optimizing a 2D frame field. Our algorithm is inspired by the method proposed by Huang *et al.* [2011], improved with an initialization that directly enforces boundary conditions. Our initialization alone is a fast and easy way to generate frames fields that are suitable for remeshing applications. For better robustness and quality, the field can be further optimized using nonlinear optimization as in Li *et al* [2012]. We make the remark that sampling the field on vertices instead of tetrahedra significantly improves both performance and quality.

Keywords: smooth frame fields, remeshing

Concepts: •**Theory of computation** → **Computational geometry;** •**Computing methodologies** → **Mesh geometry models;**

1 Introduction

Given a volume of interest, a 3D frame field can be defined at each point of the volume as a set of 6 unit vectors, globally invariant by rotations of $\pi/2$ around any of its vectors. One of the main motivations for generating such direction fields is to use them to steer the placement of objects within the volume of interest, as in hex-dominant meshing. Thus, hexahedral meshing is decomposed into two steps: (1) smooth frame field design and (2) hexahedral partitioning of the domain aligned with the frame field.

We make the remark that previous frame field generation methods address the 2D and 3D cases with very different strategies:

- In 2D, *frame* field design can be restated as *vector* field design (a simpler problem), thanks to the introduction of the **"representation vector"**. It replaces a frame (a set of 4 vectors invariant by rotations of $k\pi/2$) with a 2D vector, and defines the distance between two frames as the distance between the corresponding 2D vectors. It leads to a simple optimization problem that can easily support constraints and data fitting terms.

- In 3D, it is (unfortunately) not trivial to extend the idea of a "representation vector". Instead, Huang *et al.* [2011] propose to represent frames by **functions** defined on the unit sphere (refer to Fig. 1). They derive a definition of the field smoothness from this representation. The existing approaches optimize it in two steps: (1) initialize the field by optimizing spherical harmonics coefficients (Huang *et al.*) or propagate the boundary constraints greedily ([Li et al. 2012]) (2) further smooth this result by minimizing a non-convex objective function, using L-BFGS [Liu et al. 1989]. In this smoothing step, the frames are represented by Euler angles.

We experimented with Huang's approach in 2D (§2): each frame is represented by a sine function. The corresponding 2D optimization problem means finding for each frame its coefficients in the Fourier basis (§2.2), that will minimize the L^2 norm between adjacent frames (§2.3), subject to unit function norm constraint (§2.4). **We found that the instantiation of Huang *al.* [2011] in 2D reproduces exactly the same optimization problem as in the current 2D state-of-the-art (§2.5).**

Therefore, using the notations introduced for the 2D case, we can describe the 3D version in §3 and extend the 2D optimization mechanism to 3D in §3.5. In the end, our 3D version is very similar to

*Nicolas.Ray@inria.fr
†Dmitry.Sokolov@univ-lorraine.fr
‡Bruno.Levy@inria.fr

SA '16 Technical Papers,, December 05-08, 2016, Macao
ISBN: 978-1-4503-4514-9/16/12
DOI: http://dx.doi.org/10.1145/2980179.2982408

ACM Reference Format
Ray, N., Sokolov, D., Lévy, B. 2016. Practical 3D Frame Field Generation. ACM Trans. Graph. 35, 6, Article 233 (November 2016), 9 pages. DOI = 10.1145/2980179.2982408
http://doi.acm.org/10.1145/2980179.2982408.

Huang's initialization, except that it exactly enforces boundary constraints.

- The first difficulty is to find the expression of the boundary conditions. On the boundary the frames are free to rotate around the surface normal (§3.4) and it is difficult to enforce the alignment constraint while preserving this rotational degree of freedom. Previous work [Huang et al. 2011] only partially enforces the alignment condition, resulting in a poor initialization of the optimization procedure as evaluated in §4.2.
- The second difficulty is that a frame is represented in a 9D Fourier basis, but the set of admissible functions (those that correspond to a frame) has dimension 3 (all possible rotations of the reference frame). Thus, in 2D one simply needs to normalize the representation vector, whereas in 3D we need to find the nearest point on the 3D manifold of admissible vectors embedded in 9D space.

In the present paper, we discuss some evaluation criteria related to our main envisioned application (hexahedral dominant meshing), and we show that our algorithm outperforms the state of the art methods [Huang et al. 2011; Li et al. 2012]. Furthermore, we show that minor modifications of existing algorithms make them competitive with ours, and that our initialization alone can be sufficient in many cases.

Previous works

Frame field design on surfaces

Optimizing a frame field inside a 2D shape is very similar to optimizing a direction field on a 3D surface. The differences between a 2D shape and a 3D surface come from the curvature of the surface in the 3D case (angle defect, hard constraints, curvature fitting term).

Direction field design on surfaces was introduced by [Hertzmann and Zorin 2000] for orienting strokes in non-photorealistic rendering. In their method, directions are represented by an angle rotation θ per vertex, and the smoothing is performed by a non-linear solver (BFGS). In the reference cited above, the "representation vector" was not yet introduced, and the optimization mechanism they use is very similar to the smoothing step used in [Huang et al. 2011; Li et al. 2012] for the 3D frame fields. Solving with a representation vector $v = (r\cos(\theta), r\sin(\theta))$ for each θ was suggested in [Ray et al. 2006] for faster results, and improved later for better control over the field topology [Ray et al. 2009]. Based on representation vectors, [Palacios and Zhang 2007a] proposed to control the field topology by local operations. For direction fields without constraints, fixing the norm of the entire solution instead of the norm of each representative vector [Knöppel et al. 2013] allows to find optimal direction fields by solving an eigenvector problem.

Directly optimizing for the angle θ makes it possible to perfectly control the field topology, but this is obtained at the expense of solving a mixed-integer system [Ray et al. 2008; Bommes et al. 2009]. In [Palacios and Zhang 2007a], representation vectors are introduced together with their duality with N^{th} order traceless symmetric tensors. This relation is very useful to unify 2D and 3D frame fields.

Frame field design in volumes

In the computer graphics domain, Huang *et al.* [2011] bring the functional representation of 3D frames: they represent frames by a function defined on a sphere and they store its decomposition in the spherical harmonics basis.

Their initialization step first creates a smooth spherical harmonic field. Then, for each sample, they compute the 3D frame that is best aligned with the spherical harmonic. This initial solution is finally improved by applying an optimized rotation to each frame. These rotations are defined by Euler angles and obtained by minimizing the field smoothness with L-BFGS. Boundary alignment is enforced by a penalty term.

Li *et al.* [2012] propose an alternative initialization method. They design a 2D frame field on the volume boundary, convert it to a 3D frame field by adding the surface normal and its opposite vector, and then propagate it inside the volume. The resulting field is finally smoothed by optimizing a rotation for each frame, similarly to what is done in [Huang et al. 2011], but with a better optimization scheme. They also optimize the singularity graph of the field by local combinatorial operations, as done by [Jiang et al. 2014].

Our method optimizes for the same objective function as the references cited above, but with an initialization step that may be sufficient for remeshing applications. Our smoothing (same as [Li et al. 2012] but with a different sampling) is only required for surfaces of revolution. The performance of our method is compared with previous work in §4.

2 Functional representation in 2D

In this section, we explain how 2D algorithms like [Ray et al. 2006; Kowalski et al. 2012] can be reformulated with a functional representation for each frame (instead of a "representation vector"). The goal of this section is to introduce the concepts and notations in the simple 2D setting in order to facilitate understanding the 3D setting (later in Section 3). Note that in the 3D setting there will be no Gauss curvature. For this reason, we discuss here only the *flat* 2D setting (field defined on a subset of $\mathbb{R}^2$ instead of a 3D surface).

2.1 Definitions for the 2D setting

Given a 2D triangulated shape, frame field design in 2D consists in finding a smooth frame field aligned with the boundary of the shape. We formulate it as minimizing the field curvature, based on the following definitions:

- A **frame** is a set of 4 unit vectors $f = \{f_k\}, k \in [0, 3]$ that is invariant by a rotation of $\pi/2$ (Figure 2). It can be represented by the angle θ such that $\forall k, f_k = (\cos(\theta + k\pi/2), \sin(\theta + k\pi/2))$;
- a **frame field** is a frame per vertex of the triangulation;
- the **boundary constraint:** a normal on the boundary must match one of member vectors of the corresponding boundary frame;
- the **rotation angle between two frames** is the angle $\Delta\theta$ of the rotation that transforms one frame into the other. This angle being defined modulo $\pi/2$, we choose the $\Delta\theta$ with minimum absolute value;
- a triangle is said to be **singular** if the sum of the rotation angles over its three edges is not equal to 0.

Representing a frame field by angles is a natural and simple idea, but it makes it difficult to optimize the field curvature [Bommes et al. 2009; Ray et al. 2008]. More importantly, this representation cannot be trivially generalized to 3D frame fields. For these reasons, we propose to study and fully characterize an alternative representation, where each frame field is represented by the graph of a function with the same symmetries as the frame field.

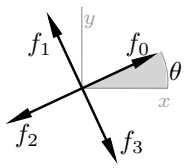

Figure 2: *A 2D frame is a set f of 4 vectors f_0, f_1, f_2, f_3 invariant under rotation by $\pi/2$. Its angle representation is the rotation θ between the global axis x and f_0.*

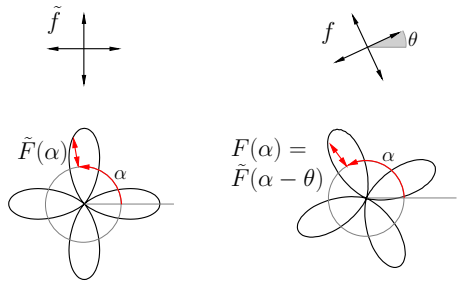

Figure 3: *Parametric plot of the reference frame representation $\tilde{F}(\alpha)$ (left) and an arbitrary frame $F(\alpha)$ (right). The plotted function is defined by $x(\alpha) = (1 + F(\alpha))\cos(\alpha)$ and $y(\alpha) = (1 + F(\alpha))\sin(\alpha)$ for $\alpha \in [0, 2\pi[$. Note that the corresponding frames are aligned with the maxima of the representation functions.*

2.2 Functional approach: frame representation

In what follows, the trivial frame $\tilde{f}$ that is aligned with the coordinate axes is called the *reference frame*. It is defined by $\tilde{f} = \{(1,0),(0,1),(-1,0),(0,-1)\}$. We represent the reference frame by the function $\tilde{F}(\alpha) = \cos(4\alpha)$ with $\alpha \in [0, 2\pi[$ (Figure 3–left). One can notice that the graph of this function $\tilde{F}(\alpha)$ is similar to the frame $\tilde{f}$ it represents[1]. In particular, it exhibits the same symmetries and $\pi/2$ rotation invariance as $\tilde{f}$.

Any frame f can be obtained through a rotation of $\tilde{f}$ by a given angle θ. The functional counterpart is to rotate the graph of the function $\tilde{F}$, namely any frame f can be represented by a function $F(\alpha) = \tilde{F}(\alpha - \theta) = \cos(4(\alpha - \theta))$ with $\alpha \in [0, 2\pi[$ (Figure 3–right).

Since a rotation of the graph does not change the frequencies of the underlying function, *any* frame function $F(\alpha)$ can be *exactly* represented as a weighted sum of Fourier basis functions $\cos(4\alpha)$ and $\sin(4\alpha)$. Indeed, if $F(\alpha)$ represents a rotation of the reference frame by angle θ, then $F(\alpha) = \cos(4\alpha - 4\theta) = \cos(4\theta)\cos(4\alpha) + \sin(4\theta)\sin(4\alpha)$.

Let us now introduce some notations: let $B = (\cos(4\alpha), \sin(4\alpha))$ denotes the row vector of the orthogonal basis functions and $a = (a_0, a_1)^\top = (\cos(4\theta), \sin(4\theta))^\top$ denotes the column vector of coordinates of a function F in the basis B. With these notations, any frame function F can be represented by its coordinates a in the basis B, or $F = Ba$.

A **coefficient vector a is feasible** if and only if there exists θ such that $a = (\cos(4\theta), \sin(4\theta))^\top$. Geometrically, a is constrained to lie on a curve parameterized by θ. This curve represents, in coefficient space, all possible rotations of the reference frame. In 2D, this curve is the unit circle, thus the feasibility constraint on a is simply $a^\top a = 1$.

At this point, we can observe that the coefficient vector a exactly corresponds to the representation vector used in the direction field literature, but it comes with a different viewpoint that will have an importance when moving to the 3D setting in Section 3. Before moving to the 3D setting, we explain how the objective function that expresses the field smoothness can be defined, as well as the constraints it should satisfy.

[1]Note that representing frames by functions is not completely new: when expressed in Cartesian coordinates, our reference frame function $\tilde{F}$ is the polynomial $4(x^4 + y^4) - 3$ restricted to the unit circle, thus this formulation is equivalent to traceless symmetric 4^{th} order tensors (in other words, degree 4 polynomials) manipulated in [Palacios and Zhang 2007b].

2.3 Functional approach: objective function

The objective function that we minimize is defined as a sum over each edge of the squared differences between the frames at the edges extremities. In our formulation, the difference between two frames (at vertices i and j) is not the rotation angle, but the L^2 distance $\int_0^{2\pi}(F^j(\alpha) - F^i(\alpha))^2 d\alpha$, which can be simplified as:

$$\begin{aligned} E &= \sum_{ij}\int_0^{2\pi}(F^j(\alpha) - F^i(\alpha))^2 d\alpha \\ &= \sum_{ij}\int_0^{2\pi}(Ba^j - Ba^i)^2 d\alpha \\ &= \sum_{ij}(a^j - a^i)^\top \left(\int_0^{2\pi} B^\top B d\alpha\right)(a^j - a^i) \\ &= \pi\sum_{ij}\|a^j - a^i\|^2 \end{aligned} \quad (1)$$

In Equation 1 above, the last step is justified because the function basis B is orthogonal ($< \cos(4\alpha), \sin(4\alpha) > \; = 0$). One can also notice that all feasible functions are of norm $\sqrt{\pi}$.

2.4 Functional approach: constraints and boundary conditions

As discussed in §2.2, the first constraint is that the variables a^i must be feasible (i.e. there should exist a frame represented by a^i).

As for boundary conditions, frames on boundary vertices i must have one member vector equal to the normal direction. If θ^i denotes the normal direction, the frame functions on boundary vertices are constrained by two equations:

$$a_0^i = \cos(4\theta^i) \quad ; \quad a_1^i = \sin(4\theta^i)$$

2.5 Implementation

At this point, our optimization problem consists in minimizing the objective function E (eq. (1)) subject to linear equality constraints on boundary vertices (eq. (2)) and quadratic equality constraints ${a^i}^\top a^i = 1$ for each vertex i. We first minimize the quadratic energy without the constraint ${a^i}^\top a^i = 1$, then choose the nearest feasible solution by normalizing a^i.

Here, we relax the feasibility constraints, so we need to minimize the quadratic function E subject to linear boundary constraints. To do that, we simply replace the linear constraints with a strong penalty term in the objective function, leading to a new quadratic

function to optimize. This penalty method is very simple and sufficient in practice.

In more details, the new quadratic function is expressed in the form $\|AX - b\|^2$ where A is a matrix, X our variable vector ($X_{2i} = a_0^i$ and $X_{2i+1} = a_1^i$) and b is a vector. The system of equations $AX - b = 0$ is constructed line-by-line:

- *initial objective function E:* for each edge ij, we add two equations (eq. (1)):

$$\sqrt{\pi}(X_{2i} - X_{2j}) = 0$$
$$\sqrt{\pi}(X_{2i+1} - X_{2j+1}) = 0$$

- *boundary constraints:* for each boundary vertex i, we add two equations (eq. (2)):

$$CX_{2i} = C\cos 4\theta^i$$
$$CX_{2i+1} = C\sin 4\theta^i,$$

  where the constant C is set as $C = 100$ in all our experimental results reported here.

From A and b, we just need to solve the linear system $A^\top AX = A^\top b$ to obtain a minimizer of $\|AX - b\|^2$. Then from X we can obtain an initialization of a^i by projecting the corresponding vectors onto the set of feasible coefficients:

$$a^i \leftarrow (X_{2i}, X_{2i+1})^\top / \|(X_{2i}, X_{2i+1})\|.$$

To solve the linear system, we use the OpenNL library [Lévy 2001], which automatically constructs $A^\top AX = A^\top b$ from the set of linear equations and then solves it by the Jacobi-preconditioned conjugate gradient method. We obtain in the end an algorithm that is very similar to the one in [Knöppel et al. 2013] or to the initialization in [Ray et al. 2006; Kowalski et al. 2012]. Smoothness can be further improved by a nonlinear solver initialized with this initial guess, as done in [Ray et al. 2006; Kowalski et al. 2012]

3 Optimization of 3D frame fields

The representation vector used by previous work was the key to efficiently optimize direction fields in 2D, however, generalizing it to the 3D setting is non-trivial. We now explain how the functional representation viewpoint of the 2D problem (§2) can be naturally generalized to the 3D setting. We present below the functional representation of 3D direction fields, the constraints they need to satisfy and derive an efficient optimization algorithm.

3.1 Definitions for the 3D setting

Given a tetrahedral mesh, our goal is now to create a smooth frame field that is aligned with the boundary of the mesh. We first introduce the following notions:

- The **reference frame** $\tilde{f}$ is the set of 6 unit vectors aligned with coordinate axes (Figure 4);
- a **frame** is the reference frame rotated by a 3×3 orthonormal matrix R: $f = R\tilde{f}$;
- a **frame field** associates a frame to each vertex of the tetrahedral mesh;
- the **boundary constraint** ensures that the frame of a boundary vertex has one of its member vectors equal to the normal at the boundary;
- a tetrahedron is said to be **singular** if any of its triangles is singular. The triangle ijk is singular if and only if $R^{ij} \times R^{jk} \times R^{ki} \neq Id$, where R^{ij} denotes the rotation matrix that brings the frame f^i to the frame f^j.
- the **stable direction** of a singular tetrahedron's facet is the set of 2 unit vectors $\{\vec{v}, -\vec{v}\}$ such that $R^{ij} \times R^{jk} \times R^{ki}\vec{v} = \vec{v}$. Having a stable direction is a necessary condition to produce hexahedral meshes that [Li et al. 2012; Jiang et al. 2014] are enforcing (on the dual mesh). It is also assumed in the definition of singularities [Persson et al. 2014] using local projections.

3.2 Frame representation

Like Huang *et al.*[2011], we represent a frame by a polynomial, whose restriction to the unit sphere exhibits the 24 symmetries of a cube. The lowest degree of such a polynomial is 4, thus our reference frame (up to a scaling factor and a constant summand) is the polynomial $x^4 + y^4 + z^4$ restricted to the unit sphere, refer to Figure 4 for an illustration.

In the 2D setting, we decomposed the reference frame polynomial onto the Fourier basis, and our variables were the coordinates in this basis (they correspond to the representation vector). We apply the same technique in 3D: we decompose the function $x^4 + y^4 + z^4$ onto the basis of spherical harmonics. Note that it is composed of a single component of frequency 4 (we removed the constant summand and normalized the function). Thus, the reference frame $\tilde{f}$ is represented by the function $\tilde{F}$ defined by:

$$\tilde{F} = \sqrt{\frac{7}{12}}Y_{4,0} + \sqrt{\frac{5}{12}}Y_{4,4}, \tag{2}$$

where $Y_{l,m}$ is the real-valued spherical harmonic of degree l and order m (see the supplemental material for more details). These harmonics are sometimes called tesseral [Ferrers 1877, p. 74]. The list of harmonics of degree 4 can be found in [Görller-Walrand and Binnemans 1996, p. 239]). The function $\tilde{F}$ is defined over the entire space $\mathbb{R}^3 \to \mathbb{R}$, but here we are only interested in its restriction to the unit sphere $\mathbb{S}^2 \to \mathbb{R}$.

An arbitrary frame f can be obtained as a rotation of $\tilde{f}$ by a matrix R. Thus, it is represented by the function $F(P) = \tilde{F}(RP)$, where P is a point of the unit sphere (Figure 4).

The family of functions $Y_{l,m}$ forms a function basis over the unit sphere. Clearly, applying a rotation to a spherical harmonic of degree l does not change its degree (it produces another spherical harmonic of degree l). This invariance is quite natural: for example, translating a 1D function does not change its frequencies, thus the degree of functions in the Fourier basis is not changed upon a translation. We represent the reference frame $\tilde{F}$ by a sum of two spherical harmonics of degree 4 (Equation (2)), so any frame function F can be *exactly* represented in the basis $B = (Y_{4,-4}, Y_{4,-3}, \ldots, Y_{4,4})$.

Let us express the reference frame function in this basis: $\tilde{F} = B\tilde{a}$ with $\tilde{a} = \left(0, 0, 0, 0, \sqrt{\frac{7}{12}}, 0, 0, 0, \sqrt{\frac{5}{12}}\right)^\top$, here B is a row vector and $\tilde{a}$ is a column vector, so $\tilde{F}$ is a scalar function defined on the unit sphere. Any other frame $f = R\tilde{f}$ can be represented as $F = Ba$, where $a = R_B\tilde{a}$ with R_B being a 9×9 rotation matrix acting on the coefficients space. The formulas and technical details behind these rotation matrices are also provided in the supplemental material.

A **feasible coefficient vector** a corresponds to the reference vector $\tilde{a}$ transformed by a rotation. In other words, it is a vector that can

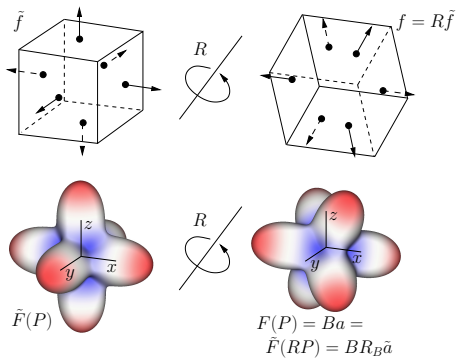

Figure 4: *An arbitrary frame f corresponds to the reference frame $\tilde{f}$ rotated by a 3×3 matrix R. The graphs of the corresponding functions F and $\tilde{F}$ are also rotated by R, and their coefficients vectors verify $a = R_B\tilde{a}$ where R_B is a 9×9 rotation matrix (derivation detailed in the supplemental material).*

be written as $a = R_B\tilde{a}$ where R_B is a 9D rotation matrix that can be derived from a 3D rotation (determined by 3 parameters). Thus, a is geometrically constrained to lie on a manifold of dimension 3 embedded in the 9D coefficient space.

At this point, one can consider the coefficient vector a as an extension of the representative vector used in the direction field literature. It also corresponds to the representation introduced in [Huang et al. 2011].

3.3 Objective function

As in the 2D setting, the objective function is defined as the sum over each edge ij of the squared difference between frames located at the edges extremities. In our formulation, the difference between two frames is the L^2 norm of the difference between the corresponding functions, defined as:

$$E = \sum_{ij} \int_{S^2} (F^j(\alpha) - F^i(\alpha))^2 d\alpha$$

Since the function basis B is orthonormal, E can be simplified as:

$$E = \sum_{ij} \|a^j - a^i\|^2 \tag{3}$$

3.4 Constraints and Boundary Conditions

There are two types of constraints that need to be satisfied by the coefficient vectors: first, each coefficient vector a^i needs to be feasible and second, frames on boundary vertices must have one vector aligned with the normal at the volume boundary. We already mentioned the first constraint at the end of §3.2. It is enforced by our optimizer in a dedicated "projection" step (3D counterpart of the normalization of a in 2D case). More details about it are given in the supplemental material.

We now focus on the boundary conditions, which have more particular cases than in 2D, depending whether the considered boundary vertex is smooth, on a crease or on a corner.

Smooth vertex

We assume first that there is only one normal associated with the vertex, it can be computed as the average normal vector of incident boundary triangles.

Case 1: the normal is equal to the z axis.

Let us first consider the case where the fixed vector (the surface normal) corresponds to the z axis. If we rotate $\tilde{F}$ around z by an angle θ, we obtain $a = R_B\tilde{a}$ with R_B being a rotation around z. The simple structure of R_B together with the zero coefficients of $\tilde{a}$ gives the equation:

$$a = \left(\sqrt{\frac{5}{12}}\sin 4\theta, 0, 0, 0, \sqrt{\frac{7}{12}}, 0, 0, 0, \sqrt{\frac{5}{12}}\cos 4\theta\right)^\top$$

As done in the construction of coefficient vectors in the 2D case, we can get rid of the angle θ by replacing it by a vector $c = (c_0, c_1) = \left(\sqrt{\frac{5}{12}}\cos 4\theta, \sqrt{\frac{5}{12}}\sin 4\theta\right)$.

$$\begin{aligned} a &= \sqrt{\frac{7}{12}}(0,0,0,0,1,0,0,0,0)^\top \\ &+ c_0(0,0,0,0,0,0,0,0,1)^\top \\ &+ c_1(1,0,0,0,0,0,0,0,0)^\top \end{aligned} \tag{4}$$

With this equation, all frames with one of their vector that corresponds to z can be represented by the 2D vector c. As in the 2D case, this equation comes with a norm constraint: $c_0^2 + c_1^2 = \frac{5}{12}$.

The variable c defines the rotation of the frame around the surface normal. In other words, it defines a 2D frame field. Optimizing this 2D frame field using c as variables is exactly what we did in 2D by introducing the coefficient vector a. Our 3D solution restricted to the object boundary is therefore almost [2] equivalent to our 2D solution.

Case 2: the normal is not equal to the z axis.

In this (more general) case, we rotate the constraint. If we want the vector $\vec{n}$ to be preserved, we first compute a rotation that brings the z axis to $\vec{n}$. From this rotation, we compute the corresponding 9D rotation matrix R_B, and derive the constraints:

$$\begin{aligned} a &= \sqrt{\frac{7}{12}}R_B(0,0,0,0,1,0,0,0,0)^\top \\ &+ c_0R_B(0,0,0,0,0,0,0,0,1)^\top \\ &+ c_1R_B(1,0,0,0,0,0,0,0,0)^\top \end{aligned} \tag{5}$$

This expression of the normal constraint gives us a set of 9 linear equations per boundary vertex. It introduces two new variables c_0 and c_1, and a quadratic constraint $c^\top c = 5/12$.

Note As in the 2D case, the boundary constraint has a simpler expression [Huang et al. 2011]: $a^\top R_B(0,0,0,0,1,0,0,0,0)^\top = \sqrt{\frac{7}{12}}$ that is valid only if all a^i are feasible, which is not ensured by their method. As a consequence, it cannot be safely used during the initialization step. We discuss the practical consequences later, and show a comparison in Figure 9.

Non-smooth vertices

Frames of vertices located on non-smooth vertices (creases, hard edges and corners) have to conform to more than one normal. These vertices have multiple normal constraints, we pick two normals that are almost orthogonal, perturb them (by rotations around their cross product vector) to make them orthogonal, and compute the rotation that brings x to the first normal, and y to the second normal. We compute the corresponding coefficient-space rotation R_B and fix the frame coefficient vector a^i to $R_B\tilde{a}$.

[2] The boundary has curvature that is not present in the (flat) 2D setting.

3.5 Implementation

We now have to minimize our objective function (eq. (3)) with linear equality constraints on boundary vertices (eq. (5)), quadratic equality constraints $c^i \cdot {c^i}^\top = 1$ on boundary vertices, and feasibility constraints for a^i. Algorithm 1 describes our optimization process.

First of all, as in the 2D case (in §2.5), our initialization step is formulated as a least squares problem (minimize $\|AX - b\|^2$), constructed *without* the feasibility constraint on a^i (line 1).

Then, the minimizer of the least-squares problem is projected onto the manifold of valid frames (lines 2-4). The projection $f^i \leftarrow$ closest_frame(a^i) onto the set of feasible coefficient vectors is no longer a simple normalization of representation vectors, as it was in 2D. Instead we perform, for each vertex, a gradient descent. In more details, starting with a seed frame, we rotate it gradually in order to minimize the distance between the current frame function and the function to be projected. The gradient is evaluated by calculating the variation of the L^2 norm induced by infinitesimal rotation matrices with Euler's angles. We do not have a formal proof, but we conjecture that there is a single minimum of the L^2 norm.

Finally, starting with the initial field, we smooth it using L-BFGS (lines 5-7). Since at this step we already have valid frames, we use Euler angles (rotation with respect to the reference frame) as variables in the L-BFGS optimization procedure. On the boundary of the volume each frame is represented by one angle representing a rotation around the surface normal. Note that each frame can be represented by 48 triplets of equivalent Euler angles. In our implementation we choose the triplet that maximizes the distance to the nearest gimbal lock. An extensive description of all steps is provided in the supplemental material.

Algorithm 1: Frame field optimization

Input: A tetrahedral mesh $\mathcal{M}$ with n_v vertices including n_l vertices with normal constraint
Output: A frame f^i for each vertex i

1 $\{a^i\}_{i=1}^{n_v} \leftarrow arg\,min\ E(\{a^i\})$ subject to normal (5), but *not* feasibility constraints ; // *solve a linear system*
2 **foreach** $i < n_v$ **do**
3 | $f^i \leftarrow$ closest_frame(a^i); // *project onto feasible manifold*
4 **end**
5 **repeat**
6 | L-BFGS iteration on $E(\{f^i\})$ subject to feasibility and normal constraints
7 **until** $\Delta E/E < 10^{-5}$ *or time exceeds*;

4 Results

First, we observe that minor modifications of the methods introduced in previous work can significantly improve the result. We obtained much better results than Huang *et al.* by sampling the field on vertices instead of tetrahedra facets. It does not avoid degenerate cases, as shown if Figure 9, but better captures the global shape of the object. The running time also significantly decreases. So, this section does not only compare our algorithm to previous work, but also evaluates benefits of our initialization alone and the impact of sampling on vertices (instead of tetrahedra or tetrahedra facets).

The main challenge here is to find a fair way to evaluate the field quality: a natural idea would be to evaluate and compare the values of the objective function (eq.3). However, we noticed that in the global value of the objective function, singular regions of the field have a dominating influence. As a consequence, a lower value does not mean that the corresponding field is better suited to hex-dominant meshing. We illustrate this remark in Figure 5, which shows an example where a field with higher objective function produces a considerably lower proportion of non-hexahedral elements.

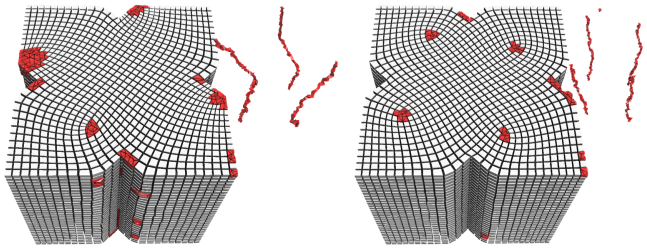

Figure 5: *The frame field initialized by Li et al.'s front propagation (left) has a better smoothness energy than ours (right). However it yields a lower proportion of hexahedra. Therefore, the quality of the singularity graph is more important than the smoothness energy.*

A better option would be to compare the quality of the mesh produced from these fields, but the state-of-the art algorithms [Nieser et al. 2011; Li et al. 2012; Jiang et al. 2014] are not robust enough for testing a large collection of objects. Moreover, it is difficult to separate the impact of the meshing scheme from the underlying field quality. We propose an alternative solution that estimates how the field's singularity graph is suitable for hex remeshing, and compare it in Table 1. The next section describes the protocol that we used. In addition, in the supplemental material, we show the hexahedra produced by CubeCover applied to the original mesh minus the singular tetrahedra. It gives an intuition about what could be expected from a hex-dominant meshing algorithm.

After a brief description of our experimental protocol, we present the results and compare the methods.

4.1 Experimental protocol

We cannot predict whether it is possible to produce a nice hex mesh from a given frame field. However, we know a necessary condition: [Nieser et al. 2011, §2.2] have proven that a singularity of a volume parameterization is always similar to a 2D singularity extruded along the third coordinate. In our setting, it means that the singularities must admit a stable direction (refer to § 3.1 for the definitions of singularities and stable directions) and it must be equal to this third coordinate. Note that only the existence of a stable direction was enforced by local operations in [Li et al. 2012].

According to this observation, we derive the following measure of field quality: first, we extract the singularity graph i.e. for each singular triangle, we create a segment linking the barycenters of its adjacent tetrahedra, then we smooth the geometry of this graph to remove the high-frequency noise that is due to the discretization. Next, for each segment, we determine the stable direction of its corresponding singular triangle. If it exists and its deviation from the segment direction is smaller than $\pi/4$, then we consider that it can be used for hex-meshing. Otherwise this part of the singularity graph is considered to be incompatible with hex meshing (see Figure 6 for an illustration).

Table 1 compares the total length of hex-incompatible singularities and the timings for our algorithm, Huang *et al.*'s algorithm and Li *et al.*'s algorithm. We also evaluate the potential of using our initialization alone, without the subsequent nonlinear smoothing steps, as a much simpler and faster alternative to the overall algorithm. To

Model	Size	Incompat. singularities						Time (s)					
		our method		Huang *et al.*		Li *et al.*		our method		Huang *et al.*		Li *et al.*	
	(K tets)	full	init only	original	modified	original	modified	full	init only	original	modified	original	modified
p1	6	0	2	0	0	0	0	1	0	4	33	1	2
p2	7	1	0	58	1	3	1	0	0	9	22	1	0
p3	11	0	0	3	0	4	0	0	0	10	25	2	0
p4	12	0	0	0	0	1	0	1	0	33	66	2	1
p5	17	2	2	117	2	3	2	1	0	33	174	3	2
p6	17	0	0	0	0	0	0	1	0	20	31	2	2
p7	18	8	16	21	10	12	6	2	0	58	198	4	3
p8	21	0	0	1	0	0	0	1	0	45	34	3	1
p9	22	0	5	11	0	9	0	2	0	21	72	8	3
p10	27	17	293	154	12	47	19	9	0	27	278	52	6
p11	34	71	125	136	78	61	85	2	1	64	248	28	4
p12	40	21	13	50	24	34	32	5	1	62	123	38	11
p13	46	85	170	200	93	145	101	7	1	71	246	72	9
p14	61	0	1	28	0	1	0	5	1	295	419	29	7
p15	69	89	113	69	107	36	94	3	1	184	280	54	4
p16	112	53	101	127	53	103	50	19	3	513	623	50	18
p17	121	25	326	154	18	110	33	30	1	183	606	278	57
p18	130	61	104	162	68	128	72	12	3	663	592	61	24
p19	239	61	34	71	59	42	69	23	6	750	630	145	23
p20	258	0	49	23	7	114	2	59	11	1447	701	603	112
p21	268	9	10	22	15	61	9	26	8	656	619	288	45
p22	328	186	399	694	236	424	242	64	10	1026	632	593	91
p23	335	0	3	20	0	54	0	204	9	810	622	361	252
p24	356	105	134	320	84	220	112	53	9	2563	740	344	82
p25	410	135	454	417	202	474	185	177	9	826	619	605	156
p26	437	0	0	0	0	138	0	42	15	2044	685	428	130
p27	448	37	179	44	22	177	28	276	7	1046	626	607	321
p28	634	46	77	79	57	56	49	98	18	1548	650	607	111
p29	790	12	51	58	12	211	31	120	26	1634	667	542	410
p30	824	357	1477	1334	353	1170	353	358	17	1662	711	611	256
p31	848	43	192	196	52	220	51	494	33	2461	656	614	249
p32	968	1451	3472	4222	1621	2644	1642	109	25	2582	531	608	97
p33	1147	1	0	39	1	0	1	133	35	2739	728	618	77
p34	1425	36	91	93	48	181	38	350	63	2987	738	618	286
p35	1607	265	381	1217	291	1106	289	178	51	5093	992	619	200
p36	1650	402	1201	2100	500	1655	404	605	50	2470	728	618	604

Table 1: *Evaluation of our algorithm (with and without the smoothing step), and Huang et al.'s algorithm (sampled on faces and sampled on vertices) with a high penalty given to the boundary condition, and Li et al.'s algorithm (sampled on tetrahedra and sampled on vertices). Column 1 is the model identifier used in the supplemental material, column 2 gives the model size (in thousands of tetrahedra), columns 3,4,5,6,7,8 report the total length of the singularity graph that is incompatible with hex remeshing (refer to §4.1), columns 9,10,11,12,13,14 report running times in seconds. We stop L-BFGS iterations when* $\Delta E/E < 10^{-5}$.

evaluate the impact of the sampling, we report the statistics of previous works with the initial sampling (on tetrahedra) and with our sampling (on vertices). Note that for Huang *et al.*'s algorithm, our sampling also introduces non-smooth vertices (§3.4), which completely fix the drifting problem encountered on hard edges. The comparison is made on a database of canonical geometrical shapes and industrial models (see supplemental materials for images). It includes a selection of failure cases for each algorithm: objects of revolution(p1, p6, p10, p17, p27), object including long cylindrical shapes (p12, 27), degenerated case for Huang's boundary condition (p4, p5), and "traps" for the front propagation initializer (p16, p17, p20, p21, p23, p26, p27, p29, p34).

4.2 Observations

Our initializer is very fast, easy to implement, and produces fields that are often equivalent to other methods. There is one exception: for objects of revolution, our initializer fails to pick a solution in the infinite set of equivalent solutions. Our method fixes these failure cases with the non-linear solver. On other models, the non-linear solver improves the quality a little bit.

Previous works are slower and have failure cases (see below). However, if we improve them by sampling on vertices instead of sampling on tetrahedra, the nonlinear solver is often able to produce fields with a quality that is similar to ours (see Figure 6). It just takes more time to converge from a weak initialization (see Figure 7).

For dense input tet meshes, even when sampled on vertices, L-BFGS is unable to reach a good configuration from a weak initialization. Such cases are illustrated in Figure 8 with failure cases of the initialization for each algorithm.

Huang *et al.*'s original method suffers from the boundary condition that is only partially enforced during the initialization step. It can lead to dramatic consequences, as illustrated on some simple shapes (Fig. 9), where a constant field perfectly satisfies their incomplete boundary condition whereas it does not match the bound-

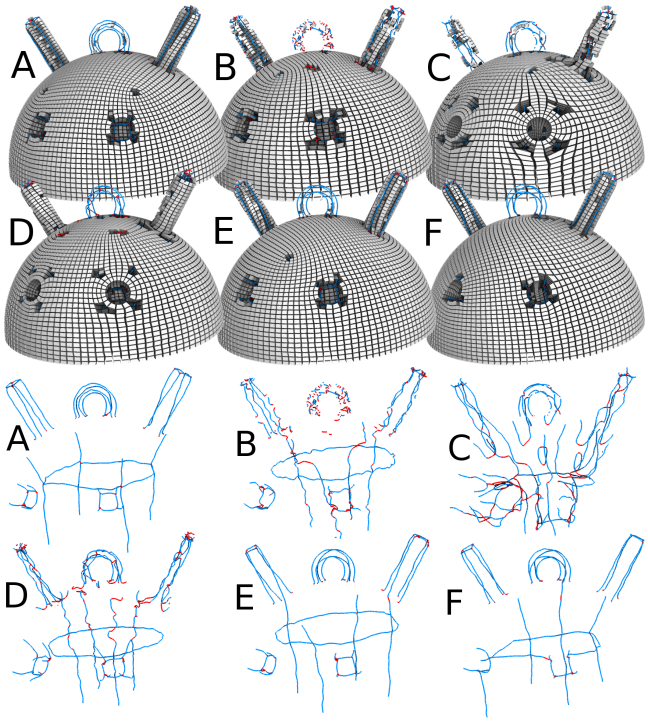

Figure 6: *Visualization of results for model p31 in Table 1. Top: hexahedra extracted from CubeCover. Bottom: singularity graphs, with the hex-incompatible parts colored in red.* ***A:*** *our algorithm;* ***B:*** *Huang's algorithm;* ***C:*** *Li's algorithm;* ***D:*** *our initialization alone;* ***E:*** *our modified version of Huang's algorithm;* ***F:*** *our modified version of Li's algorithm.*

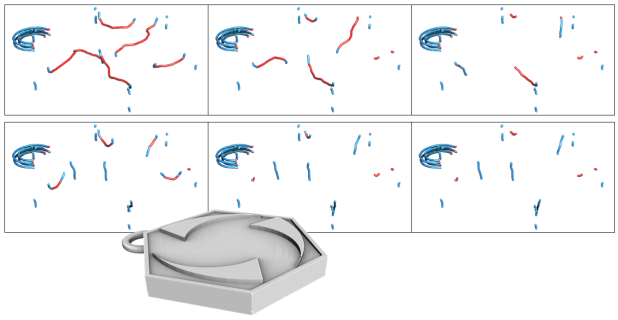

Figure 7: *The upper row shows the singularity graphs obtained by Li et al. sampled on vertices after 1 minute (left), after 3 minutes (middle) and at convergence after 20 minutes (right). The lower row shows our result at the same time steps.*

ary. One can notice that, in such cases, the harmonic function is also very far from being valid.

For less specific objects, results presented in their article are all "polycube like". We preferred to test their algorithm with a higher penalty term associated to the boundary condition: it produces more interesting fields, but also introduces new failure cases as illustrated in Figure 10.

The improved version of their method samples the field on vertices and introduces locked frames that avoid drifting on hard edges. It

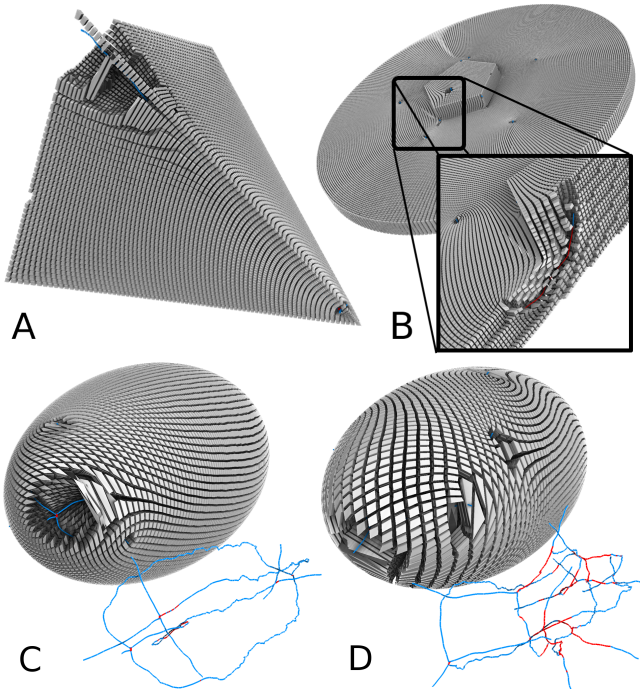

Figure 8: *Importance of the initialization for high-resolution input meshes. Even when sampled on vertices, the nonlinear solver fails to find a good singularity graph from a poor-quality initial field. We used the following initializations:* ***A:*** *improved Huang's method applied to a tetrahedron model (865K tets),* ***B:*** *improved Li's method applied to a higher resolution version of object p26 (2639K tets),* ***C:*** *our initialization on an object of revolution — ellipsoid (2713K tets),* ***D:*** *Li's method on the same ellipsoid.*

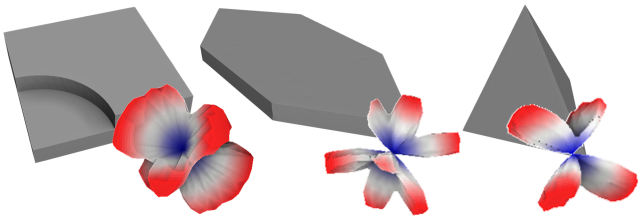

Figure 9: *Typical failure cases of Huang's method are objects where a constant field of SH functions respects all boundary conditions of the model.*

is therefore very similar to our algorithm.

Fields generated by **Li *et al.*'s original algorithm** match the boundary, however they are likely to have a singularity graph that is incompatible with hexahedral remeshing. When their initialization is too much distorted across the medial axis of the object, the nonlinear smoother is not always able to find a good quality singularity graph (Fig. 11). The modified version sampled on vertices is better but the lower quality of the initialization makes the whole algorithm slower than ours (Fig. 7).

Conclusion

In this paper, we draw a bridge between existing 2D and 3D frame field generation algorithms, and use it to extend some 2D frame field design algorithms to 3D. As a practical consequence, the ob-

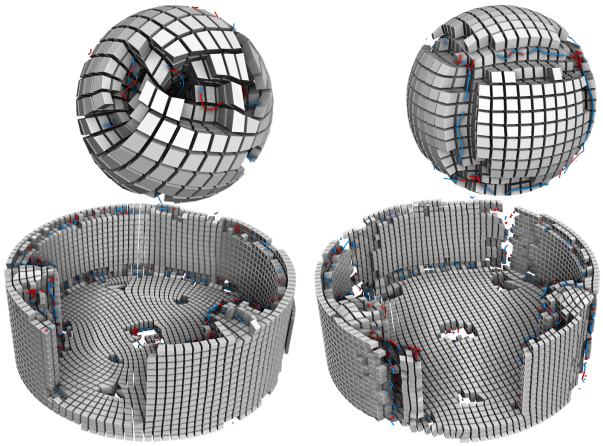

Figure 10: *Impact of penalty term for Huang's normal constraint: high value (left) avoids "polycube like" fields (right), but introduces degenerate cases (sphere).*

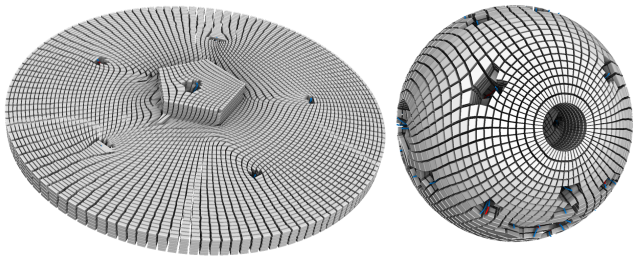

Figure 11: *Typical failure cases of Li's method are object with high field discontinuities across the medial axis.*

tained algorithm computes a better initial solution than previous work in 3D, and can be easily extended with additional constraints and/or internal boundary conditions. The initial solution computed by our algorithm (i.e. without the smoothing step) can be sufficient for some applications and has a virtue of being simple to implement: it requires only a sparse linear system solver.

The smoothness of this initial field can be further improved by a modification of [Li et al. 2012] that samples the field on vertices. The discretization on vertices that we propose makes L-BFGS much more efficient than in the original version of [Li et al. 2012] where the field was sampled on tetrahedra. With this approach we are able to generate (on a laptop) 3D frame fields on models with up to several millions tetrahedra in a matter of minutes.

To ease the reproducibility of our results, extensive explanations with all the derivations and algorithms are provided in the supplemental material.

References

BOMMES, D., ZIMMER, H., AND KOBBELT, L. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3 (July), 77:1–77:10.

FERRERS, N. 1877. *An Elementary Treatise on Spherical Harmonics and Subjects Connected with Them*. Macmillan and Company.

GÖRLLER-WALRAND, C., AND BINNEMANS, K. 1996. Rationalization of crystal field parametrization. *status: published*.

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

HUANG, J., TONG, Y., WEI, H., AND BAO, H. 2011. Boundary aligned smooth 3d cross-frame field. *ACM Trans. Graph. 30*, 6 (Dec.), 143:1–143:8.

JIANG, T., HUANG, J., WANG, Y., TONG, Y., AND BAO, H. 2014. Frame field singularity correction for automatic hexahedralization. *IEEE Transactions on Visualization and Computer Graphics 20*, 8, 1189–1199.

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

KOWALSKI, N., LEDOUX, F., AND FREY, P. 2012. A PDE based approach to multi-domain partitioning and quadrilateral meshing.

LI, Y., LIU, Y., XU, W., WANG, W., AND GUO, B. 2012. All-hex meshing using singularity-restricted field. *ACM Trans. Graph. 31*, 6 (Nov.), 177:1–177:11.

LIU, D. C., NOCEDAL, J., LIU, D. C., AND NOCEDAL, J. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical Programming 45*, 503–528.

LÉVY, B., 2001. OpenNL, Open Numerical Library. http://alice.loria.fr/index.php/software/4-library/23-opennl.html, [Online; accessed 3-May-2016].

MOAKHER, M. 2009. *Visualization and Processing of Tensor Fields : Advances and Perspectives*. D. Laidlaw and J. Weickert, eds., Springer.

NIESER, M., REITEBUCH, U., AND POLTHIER, K. 2011. CubeCover - Parameterization of 3D Volumes. *Computer Graphics Forum 30*, 5, 1397–1406.

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

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

PERSSON, P.-O., STATEN, M. L., KOWALSKI, N., LEDOUX, F., AND FREY, P. 2014. 23rd international meshing roundtable (imr23) block-structured hexahedral meshes for cad models using 3d frame fields. *Procedia Engineering 82*, 59 – 71.

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

RAY, N., VALLET, B., LI, W. C., AND LÉVY, B. 2008. N-symmetry direction field design. *ACM Trans. Graph. 27*, 2 (May), 10:1–10:13.

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