HAL
open science

On Smooth Frame Field Design

Nicolas Ray, Dmitry Sokolov

To cite this version:

Nicolas Ray, Dmitry Sokolov. On Smooth Frame Field Design. 2016. hal-01245657

HAL Id: hal-01245657

https://inria.hal.science/hal-01245657v1

Preprint submitted on 8 Jan 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

On Smooth Frame Field Design

N. RAY
INRIA Lorraine
and
D. SOKOLOV
Université de Lorraine

We analyze actual methods that generate smooth frame fields both in $2D$ and in $3D$. We formalize the $2D$ problem by representing frames as functions (as it was done in $3D$), and show that the derived optimization problem is the one that previous work obtain via "representation vectors". We show (in $2D$) why this non linear optimization problem is easier to solve than directly minimizing the rotation angle of the field, and observe that the $2D$ algorithm is able to find good fields.

Now, the $2D$ and the $3D$ optimization problems are derived from the same formulation (based on representing frames by functions). Their energies share some similarities from an optimization point of view (smoothness, local minima, bounds of partial derivatives, etc.), so we applied the $2D$ resolution mechanism to the $3D$ problem. Our evaluation of all existing $3D$ methods suggests to initialize the field by this new algorithm, but possibly use another method for further smoothing.

Categories and Subject Descriptors: I.3.5 [**Computational Geometry and Object Modeling**]: Curve, surface, solid, and object representations

General Terms: Frame field, $3D$ mesh

Additional Key Words and Phrases: smooth frame fields, remeshing

ACM Reference Format:

1. INTRODUCTION

In computer graphics, a frame field can be defined on a surface ($2D$) or inside a volume ($3D$). For each point of the domain, it defines a set of 4 (resp. 6) unit vectors invariant by rotations of $\pi/2$ around the surface normal (resp. around any of its member vector). The main motivation to study these fields is to split the quad and hexahedral remeshing problems into two steps: (1) the design of a smooth frame field, (2) and the partitioning of the domain by quads or hexes aligned with the frame field. Our objective is to unify the formulation of the $2D$ and $3D$ frame field design problems, and to use it to extend an efficient $2D$ algorithm to the $3D$ case.

In most cases, frame field design is formalized as the following optimization problem: find the smoothest frame field subject to some constraints. What makes them different from each others is obviously the dimension of the frames ($2D$ or $3D$), but also the definition of the field smoothness, the expression of the constraints, and the optimization method. Interestingly, the $2D$ case and the $3D$ case are addressed by very different strategies:

—In $2D$, the frame field design problem can be restated as a vector field design problem thanks to the introduction of the **"representation vector"**. In local polar coordinates, each vector of a frame has the same angle modulo $\pi/2$, if we multiply it by 4 we obtain a unique representation vector (modulo 2π). It is easy to derive optimization algorithms acting on the representation vectors. For simplicity reasons, we limit ourselves to planar frame fields and use the algorithm proposed by Kowalski *et al.* [Kowalski et al. 2012] as reference.

—In $3D$, it is not possible to extend the idea of "representation vector". Instead, Huang *et al.* [Huang et al. 2011] propose to represent frames by **functions** defined on the sphere, refer to figure 1 for an illustration. A definition of the field smoothness is derived from this representation and optimized in a two step procedure: (1) initialization based on optimization of spherical harmonics coefficients in [Huang et al. 2011] or front propagation of boundary constraints in [Li et al. 2012], followed by (2) smoothing iterations performed by L-BFGS on Euler angles representation of frames.

Thus our goal is to better understand how $2D$ and $3D$ problems are related to each other and to extend [Kowalski et al. 2012] to $3D$. We first show that [Kowalski et al. 2012] can be derived with the formalization approach inherited from the $3D$ case, and then we extend it to $3D$ by the same logical flow. *Busy readers interested in only reproduction of the method can skip to implementation section §3.5, the only required tool is a linear solver, all calculations are given explicitely.*

The $2D$ algorithm with frames represented by functions

Solutions developed for $3D$ are very different from $2D$ solutions because the "representation vector" trick does not extend nicely into $3D$. To unify both problems, we propose to go in the other direction §2: we apply the functional frame representation to the $2D$ case. By doing so, we found another way to introduce the "representation vectors": they appear as coefficient vector of the function decomposed in the Fourier function basis §2.2. Following the logical flow introduced for the $3D$ case, we derive an estimation of the field smoothness §2.3 and formulate the corresponding optimiza-

Authors' addresses: Nicolas.Ray@inria.fr dmitry.sokolov@loria.fr.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2009 ACM 0730-0301/2009/15-ART106 $10.00
DOI 10.1145/1559755.1559763
http://doi.acm.org/10.1145/1559755.1559763

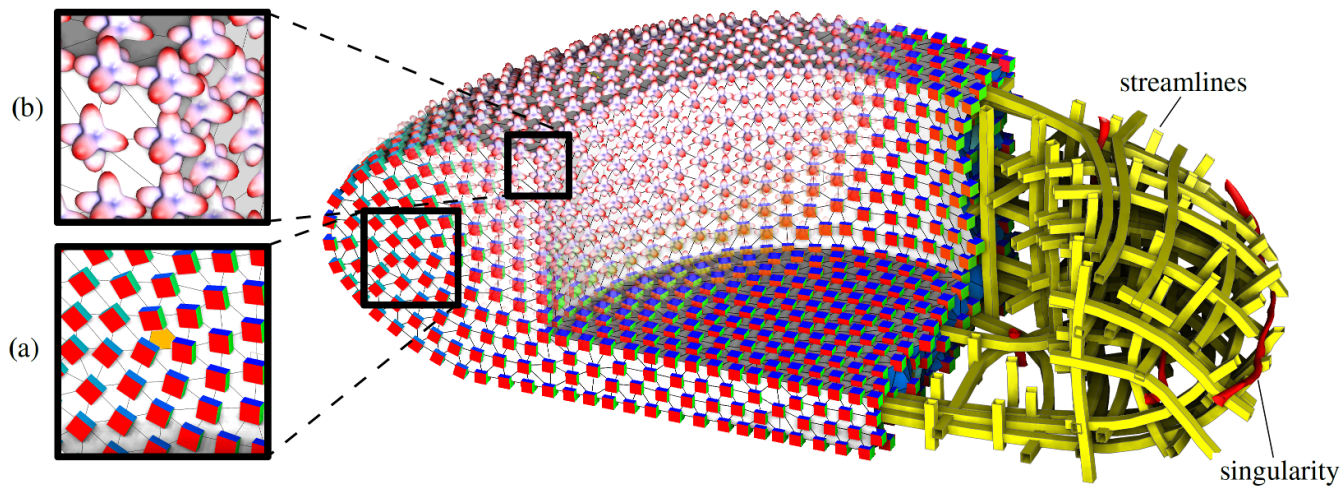

Fig. 1. Orthonormal frames (close-up (a)) are represented by spherical harmonic functions (close-up (b)), attached to each vertex of a tetrahedral mesh. Streamlines and singularities of the field are shown in yellow and red, respectively.

tion problem §2.6. We end up with exactly the formulation of $2D$ methods based on the "representation vectors".

This common formulation of the $2D$ problem is not the simplest one: the difference between adjacent frames is not evaluated by the rotation between them as in [Bommes et al. 2013; Ray et al. 2008], but approximated by the Euclidian distance in a plane. We observe the impact of this approximation on the objective function §2.7, and show how it simplifies the optimization problem §2.8.

The resolution mechanisms of the $2D$ optimization problem are strongly inspired by the geometric intuition of the representation vector field. To make abstraction of this intuition, we re-explain the algorithm from [Kowalski et al. 2012] with the notations/vocabulary introduced by the functional representation of frames.

Extension of the $2D$ algorithm to $3D$

Now that we can describe the $2D$ algorithm without referring to the representation vector, it is possible to extend it to $3D$. Using the notations introduced in the $2D$ case, we describe the $3D$ version in §3 and extend the $2D$ optimization mechanism to $3D$ §3.5.

A first difficulty was to find the expression of the boundary conditions because the boundary frames are free to rotate around the surface normal §3.4. Incomplete enforcing of this condition by Huang *et al.* [Huang et al. 2011] results in a poor initialization of the optimization procedure as evaluated in §3.6.1.

The second difficulty is that the frame is represented in a $9D$ function basis, but the set of functions that corresponds to a frame has dimension 3 (the $3D$ rotation that brings the axis aligned frame to the current frame). The extension of the normalization of the representation vector in [Kowalski et al. 2012] becomes: find the nearest point on the $3D$ manifold of the set of functions representing a frame.

Our extension of [Kowalski et al. 2012] nicely completes the tool set of $3D$ frame field design algorithms §3.6. Our initial field has lower energy, often a better topology, and is more robust to surface noise. The optimization step is easy to implement from the initialization step, is competitive when the initial topology is good enough, but does not outperform the current state of the art.

Previous works

The orientation of objects is commonly represented by symmetric tensors in physics to model the orientation distribution of fibers. For example, Moakher [Moakher 2009] introduced the notion of cubic orientation distribution functions. However, in computer graphics, more compact representations are often preferred: "representation vectors" in $2D$ and vectors of spherical harmonics coefficients in $3D$.

On surface. The optimization of a frame field inside a $2D$ shape is very similar to the optimization of direction fields on surfaces [1].

The problem of direction field design on surfaces was introduced by [Hertzmann and Zorin 2000] for orienting strokes in non photo realistic rendering. Directions are represented by an angle rotation θ per vertex, and the smoothing is performed by a non linear solver (BFGS). The "representation vector" was not introduced yet, and the optimization mechanism was similar to actual approach of $3D$ frame fields smoothing. Solving with a representation vector $v = (r\cos(\theta), r\sin(\theta))$ for each θ was used in [Ray et al. 2006] for faster results, and improved later for better control over the field topology [Ray et al. 2009]. Based on the similar representative vectors, [Palacios and Zhang 2007a] propose to control the field topology by local operations. For direction fields without constraints, [Knöppel et al. 2013] proved that the norm of the representative vector does not affects the result, leading to finding optimal direction fields by solving an eigenvector problem.

Directly working with angle θ allows to perfectly control the field topology, but at the expense of solving a mixed integer system [Ray et al. 2008; Bommes et al. 2009]. In [Palacios and Zhang 2007a], representative vectors are introduced with its duality with N^{th} order traceless symmetric tensors. This relation is very useful to unify $2D$ and $3D$ frame fields.

Inside a volume. The pioneer work of [Huang et al. 2011] discovered that each frame can be represented by a spherical har-

[1] The differences between these two problems (angle defect, hard constraints, curvature fitting term, etc.) have an impact on the optimization algorithm, as detailed in the supplemental material

monic. Their initialization step defines a smooth spherical harmonic field then, for each sample, defines the frame that better aligns with the spherical harmonic. This initial solution is then improved by rotating each frame. These rotations are defined by Euler angles and obtained by minimizing the field smoothness with L-BFGS and enforcing the boundary alignment by a penalty term.

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

Our extension of [Kowalski et al. 2012] is optimizing for the same objective function, but with a very different solution mechanism. It performances are compared with previous work in §3.6.

2. FUNCTIONAL REPRESENTATION OF FRAMES IN $2D$

This section introduces how to optimize $2D$ frame fields using a functional representation for each frame. While we do not claim any technical contribution in this section, we think that it is important to reformulate existing concepts using the functional representation, because $3D$ case inherits exactly the same difficulties and the intuition we gained in $2D$ helps to motivate the choices made for $3D$ fields. We derive an energy and the boundary conditions from this new representation. The resulting optimization problem is exactly the one usually solved by direction field algorithms based on the standard "representation vector". We show on a toy and a real examples that this optimization problem is much easier than directly minimizing the frame rotation. We then present the algorithm [Kowalski et al. 2012] that we extend to $3D$ in the following section.

2.1 Problem settings

Given a $2D$ shape, frame field design in $2D$ consists of 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 [2]:

—A **frame** is a set of 4 unit vectors $f = \{f_i\}, i \in [0,3]$ that is invariant by a rotation of $\pi/2$ (Figure 2). It can be represented by the angle θ such that $\forall i, f_i = (\cos(\theta + i\pi/2), \sin(\theta + i\pi/2))$.

—A **frame field** is a frame per vertex of a $2D$ shape triangulation.

—The **boundary constraint:** a frame located on a boundary vertex must have one of its member vectors equal to the normal on the boundary.

—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.

—The **curvature of a frame field** is the sum over each edge of the squared rotation angle between frames that are defined at the edge extremities.

—A triangle is said to be **singular** [3] if the sum of the rotation angles over his boundary is not equal to 0.

[2]The problem is directly presented in discrete settings. Interesting results on its (non trivial) continuous counterpart are given in supplemental materials.
[3]Frame field topology is further discussed in supplemental material.

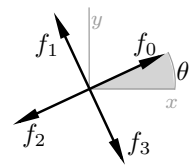

Fig. 2. A $2D$ frame is a set f of 4 vectors f_0, f_1, f_2, f_3 invariant under rotation by $\pi/2$. Its angle representation is the rotation θ between the global axis x and f_0.

Representing frames by angles is simple, but it makes the field curvature hard to optimize [Bommes et al. 2009; Ray et al. 2008], and it does not extend nicely in $3D$. For these reasons, we propose an alternative representation based on functions, and use this curvature based formalization as a reference.

2.2 Functional approach: frame representation

The frame aligned with coordinate axes is called the reference frame $\tilde{f} = \{(1,0),(0,1),(-1,0),(0,-1)\}$. Instead of using the angle approach, we represent it by the function $\tilde{F}(\alpha) = \cos(4\alpha)$ with $\alpha \in [0, 2\pi[$ (Figure 3–left), that exhibits the same $\pi/2$ rotation invariance as the frame.

Any other frame f can be obtained by a rotation of $\tilde{f}$ by 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).

A compact representation of these functions is given by the trigonometric relation $f(\alpha) = \cos(4\alpha - 4\theta) = \cos(4\theta)\cos(4\alpha) + \sin(4\theta)\sin(4\alpha)$: we see that a frame function F can be represented by a $2D$ vector of coefficients $a = (a_0, a_1)^\top = (\cos(4\theta), \sin(4\theta))^\top$ in Fourier basis $B = (\cos(4\alpha), \sin(4\alpha))$ i.e. $F = Ba$.

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 live 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, so the constraint on a is simply : $a^\top a = 1$.

At this point, we can observe that the coefficient vector a is exactly the representative vector used in the direction field literature. We can also notice that 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 it is also equivalent to the traceless symmetric 4th order tensors manipulated in [Palacios and Zhang 2007b].

2.3 Functional approach: objective function

We have defined the field curvature as the sum over each edge of the squared difference between frames at the edges extremities. In our formalization, the difference between two frames (at vertices i and j) is no longer the rotation angle, but the L^2 norm of the difference between the corresponding functions : $\int_0^{2\pi} (F^j(\alpha) - F^i(\alpha))^2 d\alpha$.

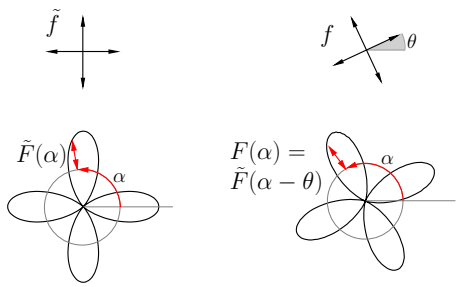

Fig. 3. Parametric plot of the reference frame representation $\tilde{F}(\alpha)$ (left) and an arbitrary frame $F(\alpha)$ (right). The plot is made using $x(\alpha) = (1 + F(\alpha))\cos(\alpha)$ and $y(\alpha) = (1 + F(\alpha))\sin(\alpha)$ for $\alpha \in [0, 2\pi[$. It is easy to see that corresponding frames are aligned with maxima of the representation functions.

It leads to the new objective function:

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

As the function basis B is orthogonal, and all functions are of norm $\sqrt{\pi}$, so the expression simplifies to:

$$E = \pi \sum_{ij} \|a^j - a^i\|^2 \tag{1}$$

2.4 Functional approach: constraints

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

The second constraint is that frames of boundary vertices i must have one member vector equals to the normal direction. If θ^i denotes the normal direction, the frame can be directly fixed by satisfying two equations:

$$\begin{aligned} a_0^i &= \cos(4\theta^i) \\ a_1^i &= \sin(4\theta^i) \end{aligned} \tag{2}$$

However, as we already have the feasibility constraint $a^{i\top} a^i = 1$, enforcing only one equation has the same effect:

$$a_0^i \cos(4\theta^i) + a_1^i \sin(4\theta^i) = 1. \tag{3}$$

2.5 Toy example

It is natural to ask the question: *"Does minimizing our energy minimizes the field curvature as well?"*

Two frames f^i and f^j are represented by a^i and a^j, both located on the unit circle. The field curvature measures the circle arc length between them, whereas our L^2 norm is the chord length between a^i and a^j.

From a practical point of view, we want to produce smooth fields, so most edges will have low curvature. In this case the objective function E is almost proportional to the field curvature. If, however, two adjacent frames are not similar (e.g. they are close to singularities), then the function E is smoother than the field curvature, making it easier for the optimization algorithm to move singularities.

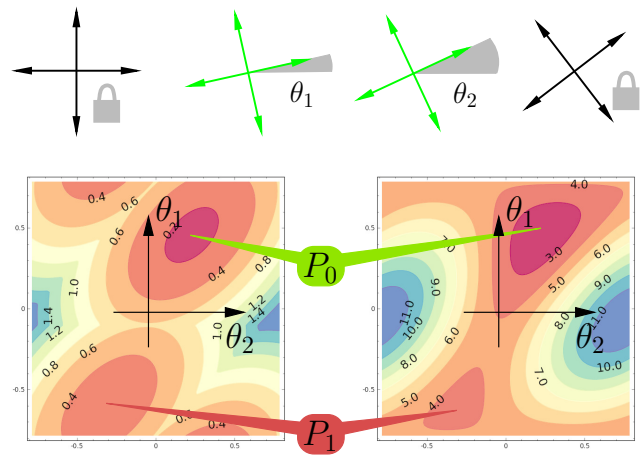

Fig. 4. Top row: interpolation problem with two locked extremity frames. Bottom row, left: field curvature plot. Bottom row, right: our objective function E. The plots are made as functions of the rotations θ_1, θ_2 of interpolated frames. Both functions share same local minima P_0 and P_1.

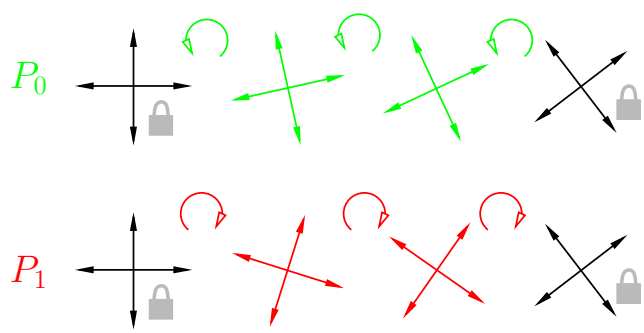

Fig. 5. Two minima for the toy problem shown in Figure 4. P_0 turns the frames counterclockwise while P_1 turns clockwise. P_0 minimizes energy E and has better field curvature.

Let us illustrate our intuition on a very simple interpolation example: a chain of four vertices having its extremities locked. The toy problem is therefore to find two frames interpolating the extremity frames.

If we represent two free frames by their angle θ_1 and θ_2, we can plot and compare the field curvature versus our objective function E (Figure 4). The field curvature is not smooth (it is piecewise quadratic) and we can observe that there are two local minima. Our objective function is smooth, and has exactly the same minima on this example. Note that it could also have a smaller number of minima e.g. if the constrained frames are more similar.

Figure 5 shows the frames corresponding to minima P_0 and P_1: they differ by the sense of rotation. The point P_0 minimizes objective function E and has better field curvature. In next two sections we expose current state of the art approach to minimization of the objective function.

2.6 Implementation

We have to minimize our objective function E (eq. (1)) with linear equality constraints on boundary vertices (eq. (2) or eq. (3)) and quadratic equality constraints ${a^i}^\top a^i = 1$ for each vertex i.

We use [Kowalski et al. 2012]'s algorithm to solve this problem. It finds an initial solution by relaxing the quadratic feasibility constraints on a^i and finding the nearest feasible solution. Then it performs a number of smoothing iterations to ameliorate the solution. In order to respect the feasibility, the quadratic constraints are linearized at each smoothing step.

Initialization. Here, we relax the feasibility constraints, so we need to minimize the quadratic function E subject to linear boundary constraints. To do it, we simply replace the linear constraints by 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.

More precisely, 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)):

$$\lambda X_{2i} = \lambda \cos 4\theta^i$$
$$\lambda X_{2i+1} = \lambda \sin 4\theta^i,$$

where we set $\lambda = 100$ in our experiments.

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 corresponding vectors on the set of feasible coefficients:

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

Smoothing iterations. Each smoothing iteration is similar to the initialization problem, except that we add to the objective function a new quadratic penalty term that corresponds to a linear approximation of the feasibility constraint as done in [Kowalski et al. 2012, p. 6]. As before it is expressed by a new set of linear equations when constructing A and b: for each vertex i, we add one equation $\lambda(X_{2i}a_0^i + X_{2i+1}a_1^i - 1) = 0$, where a^i denotes the solution obtained at the previous iteration.

To solve linear systems we use OpenNL library [Lévy]: it automatically constructs $A^\top AX = A^\top b$ from the set of linear equations and then solves it by the conjugate gradient method.

2.7 Toy problem revisited

This section explains our optimization approach on the toy problem already presented in § 2.5. As we mentioned before, at the initialization step we relax the constraints of feasibility of a^i. Unfortunately we can not plot the corresponding energy since without the constraints it becomes four-dimensional.

Top row of Figure 6 shows the solution of the initialization stage. Intuitively, we allow the points a^i not to be on the unit circle. Hence a^1 and a^2 lie on the chord between locked points a^0 and a^3. Second row of Figure 6 shows the corresponding functions and the third row gives the frames obtained by projecting points a^i on the circle of constraints.

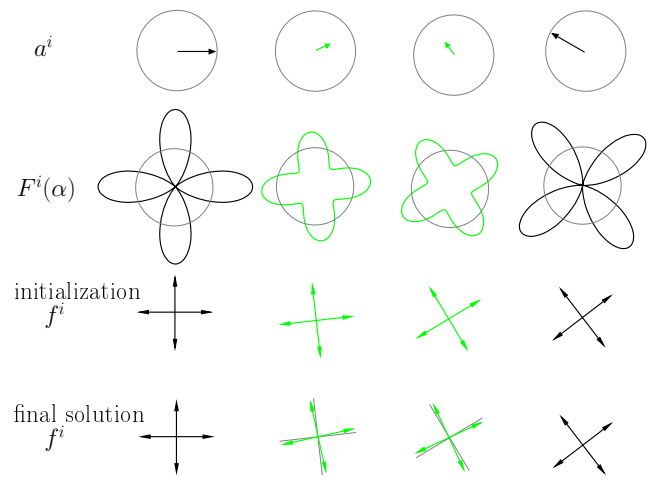

Fig. 6. First row: initialization stage solution. Second row: corresponding functional interpretation. Third and fourth rows: frames obtained by the projection of initialization a^i and after smoothing iterations.

Note that the initialization stage produces the correct sense of rotation (Figure 5). However it does not directly produce the optimum point P_0. The reason being that the initialization stage produces points a^1 and a^2 (before normalization) in the way that all three chord segments are equal: $\|a_0 - a_1\| = \|a_1 - a_2\| = \|a_2 - a_3\|$. But after projecting the points onto the feasible circle (3rd row of Figure 6) the corresponding arc lengths are not equal. Therefore, we need a few smoothing iterations to reach the optimum (Figure 5—bottom row).

2.8 Results

Figure. 7–left shows the optimization of the field curvature by a gradient descent algorithm, initialized by an axis aligned frame field. We obtain a frame field that bends to match the boundary constraints, but its singularities remain on the boundary. The system is only able to reach the local minima corresponding to the initial field topology. The field curvature [4] is 34.21.

Figure 7–middle shows the optimization using only the smoothing iterations, initialized by an axis aligned frame field. We observe that smoothing iterations were able to slightly move the singularities away from the border and even to merge two singularities. It results in a better field curvature (equal to 31.41, 24.01 and 23.95 after 10^2, 10^3 and 10^6 iterations).

Figure 7–right shows that the initialization step alone finds a solution with a simple topology and a lower field curvature (20.84). Smoothing iterations further decrease the field curvature (to 17.91).

These observations suggest that our initialization step is mandatory, and smoothing iterations further improve the result. However, it is important to notice that each iteration takes approximately the same time as the initialization step.

Boundary constraints. When working only with feasible solutions a single equation (equation (3)) is sufficient to enforce each

[4]The value of the field curvature is relevant only for comparing different fields on the same mesh.

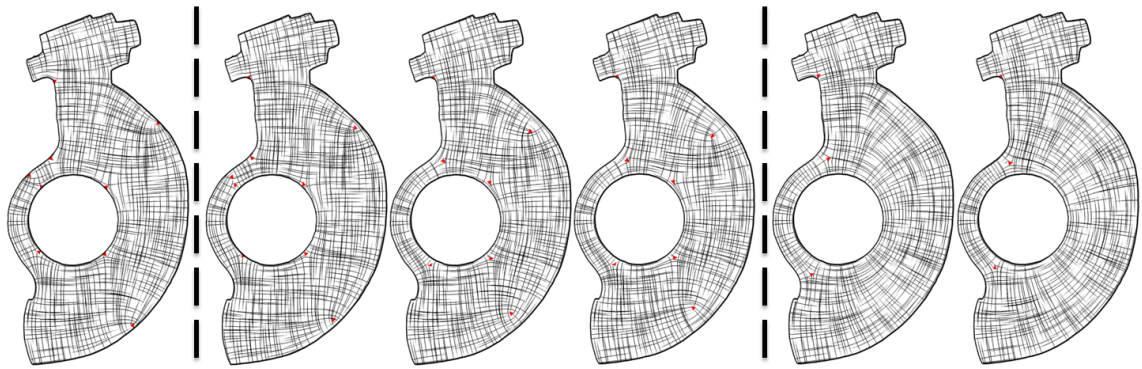

Fig. 7. *Evaluation of $2D$ frame field optimization algorithms:* singular triangles are highlighted in red. Compared algorithms are: **(left)** steepest descent of the field curvature, initialized with an axis aligned frame field, **(middle)** smoothing iterations with our objective function E initialized with axis aligned frame field after 10^2, 10^3 and 10^6 iterations (from left to right), **(right)** initialization step alone (left) and the initialization step followed by 10^3 smoothing iterations (right).

boundary constraint. Using it for the initialization step is wrong: for example, a normal constraint of angle $\theta = 0$ forces $a_0 = 1$ but let a_1 free. As illustrated in Figure 8, it can produce very bad frames fields. Therefore we must use two separate equations (2).

There exists a similar issue in $3D$: Huang et al [Huang et al. 2011] use a $3D$ boundary condition that is not sufficient for the initialization step. It leads to a poor initialization for their smoothing algorithm, making it very slow, and getting possibly locked with a bad initial topology. This issue is discussed in details in §3.6.

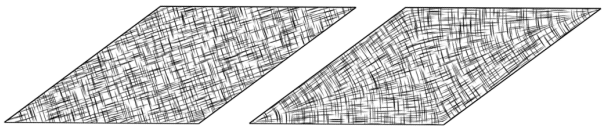

Fig. 8. *Boundary constraints for global optimization:* if we only use one constraint (eq.3), the initialization step finds a constant frame field on a parallelogram (left). It is therefore mandatory to lock the frames to get the proper boundary constraints (right).

3. OPTIMIZATION OF $3D$ FRAME FIELDS

Our objective is to extend to $3D$ the optimization algorithm presented in previous section.

In $2D$, our framework allows to retrieve the representation vector that was the key to efficient optimization of direction fields. We now extend our framework to $3D$, find a generalization of this representation vector, express the boundary alignment condition with respect to this representation, and derive the optimization algorithm.

3.1 Problem settings

The problem is to define, inside a $3D$ shape, a smooth frame field that is aligned with the boundary of the shape. We are working in discrete settings on a tet mesh. The problem to minimize the field curvature is defined as follows:

—The **reference frame** $\tilde{f}$ is the set of 6 unit vectors forming normals of a cube aligned with coordinate axes (Figure 9).

—A **frame** is the reference frame rotated by a 3×3 matrix R: $f = R\tilde{f}$. Note that multiplying a matrix by a set is a slight abuse of notation, it means that we obtain a new set where each vector is rotated by the given matrix.

—A **frame field** is the definition of a frame for each vertex of the tet mesh.

—The **boundary constraint**: The frame of a boundary vertex must have one of its member vectors equal to the normal of the boundary.

—The **rotation angle between two frames** is the minimal angle of rotation that brings one frame to the other.

—The **curvature of a frame field** is the sum, over each edge, of the squared rotation angle between adjacent frames.

—A tet is called **singular** if any of its triangles is singular.[5] 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.

3.2 Frame representation

The reference frame $\tilde{f}$ is represented by the function $\tilde{F} = \sqrt{\frac{7}{12}}Y_{4,0} + \sqrt{\frac{5}{12}}Y_{4,4}$, where $Y_{l,m}$ is the real valued spherical harmonic of degree l and order m. 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]). Function $\tilde{F}$ is defined as $\mathbb{R}^3 \to \mathbb{R}$, but we are interested by its restriction to the unit sphere $\mathbb{S}^2 \to \mathbb{R}$.

Any other frame f can be obtained as a rotation of $\tilde{f}$ by a matrix R. It is represented by the function $F(P) = \tilde{F}(RP)$, where P is a point of the unit sphere (Figure 9).

$Y_{l,m}$ forms a functional basis over the unit sphere with an interesting property: applying a rotation to a spherical harmonic of degree l produces another harmonic of degree l. As a consequence, since we represent the reference frame by a sum of two spherical harmonics of degree 4, each frame function F can be represented in the basis $B = (Y_{4,-4}, Y_{4,-3}, \ldots, Y_{4,4})$. Using it, we can rewrite the expression for the reference frame function as $\tilde{F} = B\tilde{a}$ with

[5]We can define what a singular tet is, but we are not able to characterize them by an equivalent of the index in $2D$. This fact is discussed in the supplemental material.

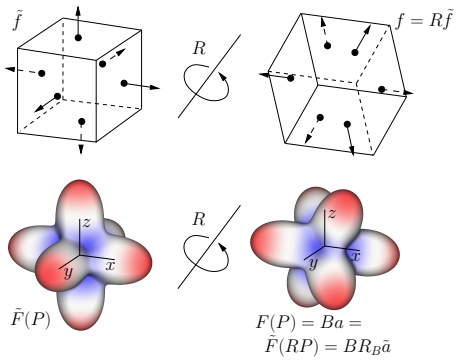

Fig. 9. A frame f is the reference frame $\tilde{f}$ rotated by a 3×3 matrix R. The plots of the corresponding functions F and $\tilde{F}$ are also rotated by R, and their coefficients vectors verify $a = R_B\tilde{a}$ where R_B is a 9×9 rotation matrix given in §A.1.

$\tilde{a} = \left(0, 0, 0, 0, \sqrt{\frac{7}{12}}, 0, 0, 0, \sqrt{\frac{5}{12}}\right)^\top$. Any other frame $f = R\tilde{f}$ can be represented by $F = Ba$, where $a = R_B\tilde{a}$ with R_B being a 9×9 rotation matrix acting on coefficients space. Appendix A.1 describes the construction of the rotation matrices.

A **feasible coefficient vector** a is a vector that can be written as $a = R_B\tilde{a}$ where R_B is a $9D$ rotation matrix that can be derived from a $3D$ rotation. Geometrically, a is constrained to be on a manifold of dimension 3 embedded in the $9D$ coefficient space.

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

3.3 Objective function

As in $2D$, the objective function is the sum, over each edge ij, of the squared difference between frames located at the edges extremities. In our formalism, the difference between two frames is not the rotation angle, but the L^2 norm of the difference between the corresponding functions: $\int_0^{2\pi}(F^j(\alpha) - F^i(\alpha))^2 d\alpha$. It gives the energy:

$$E = \sum_{ij} \int_0^{2\pi} (F^j(\alpha) - F^i(\alpha))^2 d\alpha$$

Here, the function basis B is orthonormal, so the expression simplifies to:

$$E = \sum_{ij} \|a^j - a^i\|^2 \tag{4}$$

3.4 Constraints

There are two types of constraints: each coefficient vector a^i must be feasible, and boundary frames must have one vector aligned with the normal of the volume boundary. The first constraint is presented in the frame representation section, and will be enforced by our optimizer in a dedicated "projection" step (the $3D$ counterpart of the normalization of a in the $2D$ case). Here we focus on the boundary constraint.

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.

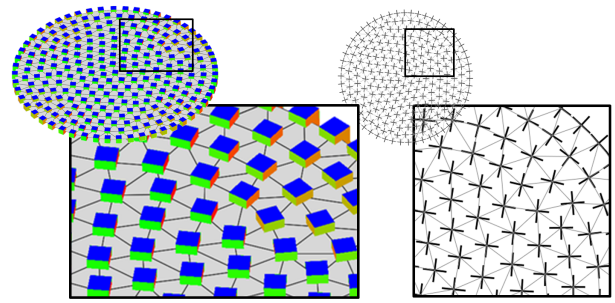

Fig. 10. A $3D$ frame field produced on a ($2D$) disk (Left) produces the $2D$ frame field we could obtain from the $2D$ algorithm (Right).

Let us first consider the case where the fixed vector (the surface normal) is the axis z. If we rotate $\tilde{F}$ around z by 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 null coefficients of $\tilde{a}$ gives the equation:

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

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 $c = (c_0, c_1) = \left(\sqrt{\frac{5}{12}}\cos 4\theta, \sqrt{\frac{5}{12}}\sin 4\theta\right)$.

$$\begin{aligned} a &= \sqrt{\frac{7}{12}}(0,0,0,0,1,0,0,0,0)^\top && (6)\\ &+ c_0(0,0,0,0,0,0,0,0,1)^\top && (7)\\ &+ c_1(1,0,0,0,0,0,0,0,0)^\top && (8)\end{aligned}$$

With this equation, all frames having a vector equal to z can be represented by the $2D$ vector c. As in the $2D$ case it 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 i.e. a $2D$ frame field. The optimization of 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 [6] equivalent to our $2D$ solution (Figure 10).

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

To handle 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 z axis to $\vec{n}$. From this rotation, we compute the corresponding $9D$ rotation matrix R_B, and derive the constraints:

$$\begin{aligned} a &= \sqrt{\frac{7}{12}}R_B(0,0,0,0,1,0,0,0,0)^\top && (9)\\ &+ c_0R_B(0,0,0,0,0,0,0,0,1)^\top && (10)\\ &+ c_1R_B(1,0,0,0,0,0,0,0,0)^\top && (11)\end{aligned}$$

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. Consequently, it cannot be used safely during the initialization step (see Figure 17).

[6] The boundary has curvature that was not assumed in our $2D$ frame fields.

Non smooth vertices. Frames of vertices located on hard edges 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 $R_B\tilde{a}$.

3.5 Implementation

We have to minimize our objective function (eq. (4)) with linear equality constraints on boundary vertices eq. (9), quadratic equality constraints $c^i \cdot {c^i}^\top = 1$ on boundary vertices, and the constraint that each a^i is feasible.

As in the $2D$ case (in §2.6), our minimization algorithm (Algo. 1) is formulated as a series of least squares problems (minimize $\|AX - b\|^2$), where A and b are constructed without the feasibility constraint of a^i at the first iteration (initialization), and with a linear approximation of it in the subsequent iterations (smoothing iterations).

—**Initialization:** Our variable vector X must represent the representation vectors a^i but also the c^i variables introduced to express the boundary constraint. To do so, we first reorder vertices to have boundary vertices first [7]. We can then organize the variable vector X by blocks: $X[9i+d] = a^i_d$, and $X[9n_v+2i+d] = c^i_d$ where n_v is the number of vertices.

As in the $2D$ case, the matrix A and the vector b are constructed iteratively by adding new equations for the objective function (algorithm 3) and the boundary constraints (algorithm 2). In our approach, we do not explicitly enforce the feasibility of c^i ($c^i \cdot {c^i}^\top = 5/7$), but it will be indirectly respected by the feasibility of a^i.

The projection of a^i on the set of feasible coefficient vectors is no longer a simple normalization. Instead we perform, for each vertex, a gradient descent (algorithm 5) initialized by $\tilde{a}$. More precisely, starting with $\tilde{a}$ we rotate our current frame 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.

—**Smoothing iteration:** For the linearized feasibility constraint of a^i, we must also add 3 extra variables per vertex to our system. These variables account for the position in a local basis of the tangent space of the $3D$ manifold of feasible a^i. The introduction of these constraints in the matrix A is detailed in algorithm 4.

This frame field design algorithm can be implemented without being expert in spherical harmonics. We give explicit construction of matrices R_B, E^x, E^y, E^z in the appendix. The system to solve $A^\top AX = A^\top b$ is simply a linear system with a positive definite matrix. We use the OpenNL library [Lévy] because the system can be directly constructed from the equations (lines of A and elements of b).

In order to keep the algorithm easy to read, we did not detail how to lock frames for vertices with multiple normal constraints.

[7] It is possible to increase the performances by $\approx 30\%$ by doing a Hilbert sort in the boundary vertices block, and another one in the free vertices block

Algorithm 1: Frame field optimization

Input:
—A tetrahedral mesh $\mathcal{M}$ with:
  —n_v vertices including n_l vertices with normal constraint
  —edges $\mathcal{E}$
—number of smoothing iterations N

Output: A frame f^i for each vertex i

sort($\mathcal{M}, \mathcal{E}$); *// vertex i is a boundary vertex $\iff i < n_l$*
foreach $I \in 0 \dots N$ **do**
  // A and b will be constructed iteratively
  create matrix A with 0 rows and $9n_v + 2n_l + 3n_v$ columns;
  create vector b of size 0;
  add_smoothing_terms($\mathcal{M}$, A, b);
  add_normal_constraints($\mathcal{M}$, A, b);
  // add constraints only if we are in a smoothing iteration
  if $I > 0$ **then**
    add_local_optim_constraints($\{a^i\}$, $\mathcal{M}$, A, b);
  end
  // solve $A^\top AX = A^\top b$
  $X \leftarrow$ call_least_squares_solver(A, b);
  // find the frame for each vertex
  foreach $i < n_v$ **do**
    $a^i \leftarrow X[9i \dots 9i + 8]$;
    $(f^i, a^i) \leftarrow$ closest_frame(a^i);
  end
end

Algorithm 2: add_normal_constraints

Input: A tetrahedral mesh $\mathcal{M}$, matrix A, row b
Output: Modified matrix A and vector b

// enforcing normal constraints by quadratic penalty
foreach $i < n_l$ **do**
  estimate normal n at vertex i;
  find Euler angles (α, β, γ) to rotate z-axis to n;
  find 9×9 rotation matrix R_B; *// see appendix A.1*
  $h_0 \leftarrow R_B \times (1, 0, 0, 0, 0, 0, 0, 0, 0)^\top$;
  $h_4 \leftarrow R_B \times (0, 0, 0, 0, 1, 0, 0, 0, 0)^\top$;
  $h_8 \leftarrow R_B \times (0, 0, 0, 0, 0, 0, 0, 0, 1)^\top$;
  $\lambda \leftarrow 100$; *// quadratic penalty multiplier*
  foreach $d \in 0 \dots 8$ **do**
    create vector row;
    $row[9i + d] \leftarrow \lambda$;
    $row[9n_v + 2i + 0] \leftarrow \lambda h_0[d]$;
    $row[9n_v + 2i + 1] \leftarrow \lambda h_8[d]$;
    A.add_row(row);
    b.push($\lambda\sqrt{7/12}h_4[d]$);
  end
end

3.6 Results

It is impossible to compare frame field design algorithms only from the images and results presented in the state of the art papers. First of all, our implementation of [Huang et al. 2011] has significantly better performances compared to what was presented in the orig-

Algorithm 3: add_smoothing_terms

Input: A tetrahedral mesh $\mathcal{M}$, matrix A, row b
Output: Modified matrix A and vector b

```
foreach ij ∈ E do
    foreach d ∈ 0 ... 8 do
        create vector row;
        row[9i + d] ← 1;
        row[9j + d] ← −1;
        A.add_row(row);
        b.push(0);
    end
end
```

Algorithm 4: add_local_optim_constraints

Input: Previous solution $\{a^i\}_{i<n_v}$, tetrahedral mesh $\mathcal{M}$, matrix A, row b
Output: Modified matrix A and vector b

```
foreach i < n_v do
    c_x ← E^x × a^i; // see appendix A.2
    c_y ← E^y × a^i;
    c_z ← E^z × a^i;
    λ ← 100; // quadratic penalty multiplier
    foreach d ∈ 0 ... 8 do
        create vector row;
        row[9i + d] ← λ;
        row[9n_v + 2n_l + 3i + 0] ← −λc_x[d];
        row[9n_v + 2n_l + 3i + 1] ← −λc_y[d];
        row[9n_v + 2n_l + 3i + 2] ← −λc_z[d];
        A.add_row(row);
        b.push(λa^i[d]);
    end
end
```

Algorithm 5: closest_frame

Input: 9-component vector q
Output: A frame f and its representation vector a

```
f ← f̃;
a ← ã;
s ← 10^−1; // optimization step size
ε ← 10^−4; // step threshold
q ← q/|q|;
while True do
    g ← (q^⊤ E^x_B a, q^⊤ E^y_B a, q^⊤ E^z_B a); // gradient in point a
    if ||g|| < ε then
        break;
    end
    R_B ← R^x_B(s · g[0]) × R^y_B(s · g[1]) × R^z_B(s · g[2]);
    R ← R^x(s · g[0]) × R^y(s · g[1]) × R^z(s · g[2]);
    a ← R_B a;
    f ← Rf;
end
return f, a;
```

inal paper. Next, Li *et al.* [Li et al. 2012] did not present any frame field results, but only hex meshes that was the main focus of their work. Therefore, we implemented both methods; there are few points worth noting:

Sampling: In previous works the frame fields were sampled either on each tet face or on each tet. Instead we sample it on vertices, otherwise we would not be able to compare corresponding energies.

Gimbal Lock: Both Huang and Li use Euler angles as variables in their L-BFGS optimization, which have numerical issues when the angles are close to the gimbal lock. Note that each frame can be represented by 48 triplets of equivalent Euler angles. In our implementation we choose the triplet maximizing the distance to the nearest gimbal lock.

Rendering: For rendering purposes, we rely on a combination of techniques (Figure 16) to show the spherical harmonics field, the frame field (locally and globally) and the field topology.

3.6.1 *Comparison with Huang's method.* Recall that Huang *et al.* proposed a method in two steps:

—find an initial frame field by solving a linear system and projecting the solution onto the manifold of feasible solutions

—represent each frame by a triplet of Euler angles and optimize the smoothness using an L-BFGS descent method.

Our implementation produces results very similar to those presented in [Huang et al. 2011], but with significantly better timings. For example, the rockarm (Figure 11) with one million tetrahedra takes about 10 minutes on a single thread application on a Dell M6600 laptop compared to 155 minutes obtained by Huang *et al.* on a two-thread i7 processor.

Huang's initialization is very similar to ours, however their boundary condition is not sufficient in this case (it requires the SH coefficients to be feasible). Moreover, they enforce the boundary condition with a penalty term that is very light (10^{-2} weight). As a consequence, their initial fields are almost constant everywhere (it maximizes the smoothness), with a topology very far from being optimal. The smoothing iterations are performed with much higher weight (10^3) of the penalty term using L-BFGS.

After the initialization step we measured the deviation of the field from the given constraints on the rockarm model. Note that the penalty term being the sum of deviations over all vertices, we can conclude that deviation at a given vertex belongs to $[0, 2\sqrt{7/12}]$. Thus on the rockarm the initial frame field has the average deviation of 0.34, whereas the maximum frame deviation is 0.96.

If we use a much higher penalty weight to enforce the boundary constraint, we obtain an initial frame field with average deviation from constraints equal to 0.07 and maximum deviation equal to 0.75. The field has better topology and L-BFGS converges faster for this initialization. The initialization provided by our method has average deviation from constraints equal to 10^{-8} with maximum deviation of 10^{-7}.

Figure 11 gives an illustration, it compares three methods: Huang's algorithm (left image) Huang's algorithm with much higher penalty weight (middle image) and our initialization followed by Huang's smoothing iterations (right).

Huang's paper is the pioneer work and the main contribution in [Huang et al. 2011] was the introduction of the energy used in frame field optimization, however the initialization is not very good and smoothing stage was also outperformed by later works (see §3.6.3).

3.6.2 *Comparison with Li's method.* Li's initialization computes a $2D$ frame field on the surface, then propagates it inside the volume by advancing front. As a consequence, the initial field

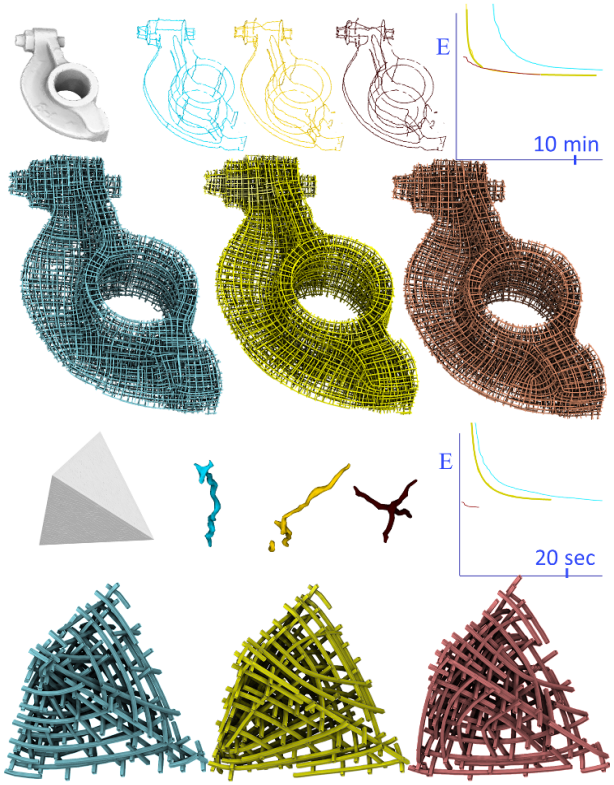

Fig. 11. Comparison of initialization of Huang's algorithm with their weight of boundary penalty term (10^{-2}) (left/blue), with a much higher weight (10^{2}) (middle/yellow), and our initialization (right/red). All smoothing iterations are performed by Huang's algorithm. For the rocker model, the energies (we take the run with our initialization for 100%) are respectively $94,7\%$, $101,6\%$ and (obviously) 100%. For the tet model, we obtain 91%, $89,7\%$ and 100%.

perfectly matches boundary constraints, but is discontinuous across its medial axis.

The smoothing iterations are performed by L-BFGS. It acts on a new set of variables that characterizes, per vertex, the rotation that brings the reference frame to the current frame. For the frames located inside the object, variables are the Euler angles as in Huang's method. For frames located on the object boundary, the rotation is characterized by a single rotation angle around the normal vector.

The frame field results presented in their paper were limited to an ellipsoid and a sphere (frame field design was not the primary objective). We guess that most of presented results were not fully automatically generated frame fields, as they wrote: *"For instance, we use guiding boxes to modify the frames inside the narrow ears of Bunny (Figure 13-a) and the head of rock arm (Figure 13-b) to reduce singularities"*.

Moreover, before implementing the method, we thought that their algorithm was strongly limited by the original field topology from the sentence: *"However, our propagation-based frame field initialization likely generates singular edges around the medial axis of the volume, and most of them cannot be eliminated by frame optimization."* Surprisingly, our implementation of their method is able to generate smooth frame fields automatically in most situations, even when the topology of the initial field is complex close to the medial axis. Our implementation is slightly different:

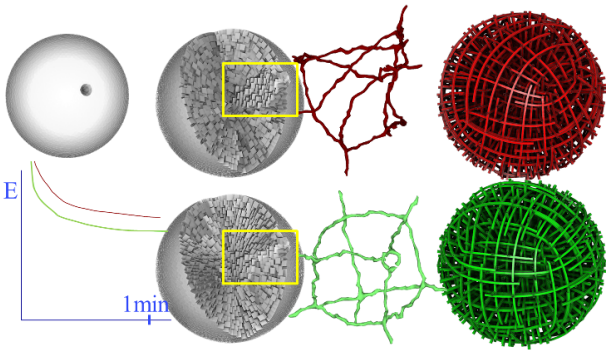

Fig. 12. Li's algorithm (red) is compared to our algorithm (green) on a one-finger bowling ball. The hole has a huge impact on the initialization due to the advancing front approach (second column, inside the yellow box). As a result, their initialization provides a field with a poor topology and smoothing iterations are not sufficient to find the expected topology (like ours). Our final energy is 86% of theirs.

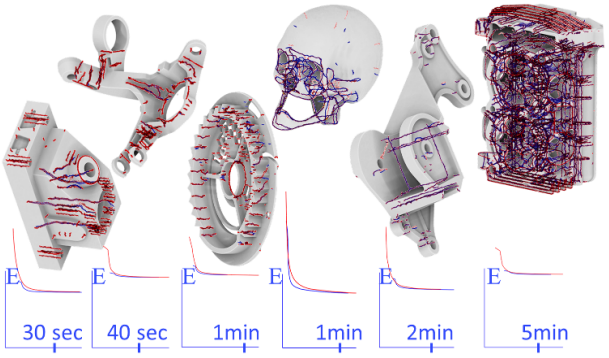

Fig. 13. Comparison of two fields generated by Li *et al.* smoothing iterations: using their initialization (red) or ours (blue). Our energy in percent of theirs is 99%, 99.87%, 100.5%, 99.97%, 99%, 99.6%. The difference is always very low ($< 1\%$) but always in our advantage except for the third model.

—we initialize the $2D$ field by our $3D$ algorithm restricted to boundary vertices

—we sample the field on vertices

—and we prevent gimbal locks by a proper initialization of Euler angles.

The only real failure case we found using their method is due to the front propagation algorithm: when a boundary frame is copied to a large number on inner samples. In this case, the L-BFGS solver is sometimes locked with a bad topology (see Fig. 12).

On more complex examples, we have compared their algorithm against our initialization followed by their smoothing iterations. In most cases, we obtain an energy that is a bit better (Fig 13). We also observed that their topology often differs from ours (Fig 14), so we conjecture that our topology is somehow better. However, the quality of the field topology depends on the application, and is not well evaluated by the energy, even for topologies very far from being optimal (see e.g. Fig 11).

3.6.3 *Comparison of smoothing iterations.* In the previous sections we have shown (Fig. 11 and 12) that our method pro-

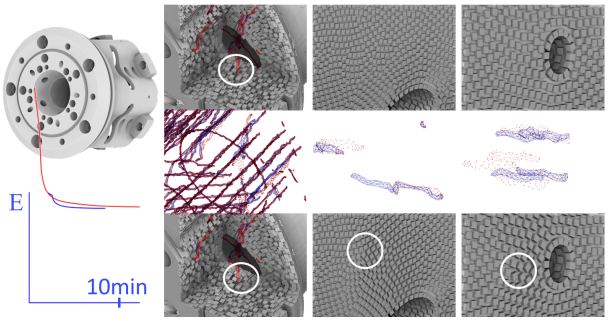

Fig. 14. Comparison of two fields generated by Li's smoothing iterations: using their initialization (red) or ours (blue). Our energy in percent of theirs is 98.5%. Close-ups show the field where the singularity graphs diverge: one is inside the volume (leftmost) and others are on the object boundary. Our results are on the top row and theirs are on the bottom row with singularity encircled in white.

vides the best initialization, however our smoothing iterations are not clearly better than others.

Figure 15 shows a comparison of three different smoothing strategies (our linearization, L-BFGS proposed by Huang *et al.* and L-BFGS proposed by Li *et al.*). In all three cases we use our method to initialize the field. L-BFGS smoothing proposed by Huang *et al.* is the slowest in all test cases. First of all, in our implementation the time to evaluate the energy and the gradient is four times slower for the method by Huang *et al.* than for the method by Li *et al.* Second, the cruicial difference between these two methods is the way to enforce the boundary alignment: Huang *et al.* use a penalty term, whereas Li *et al.* use the set of variables directly satisfying the boundary constraints. In our test we noticed that usage of penalty terms increases the number of iterations to converge and interferes with topological choices to be made, leading to inferior final fields.

We also noticed that the behaviour of linearization changes in function of how far the initialization is from the final solution.

—First row of figure 15 shows a simple case without topology changes, the smoothing iterations change the field geometry only. In this case the linearization of feasibility constraints works flawlessly, in two iterations the method converges, taking about the same time as the smoothing by Li *et al.*

—Middle row shows a second case, where few topology changes must be made. It slowes the linearization down, even if two iterations produce a reasonably good field.

—Finally, the bottom row shows the case where the initial topology is really bad. The linearization method fails on this model: two first iterations are still very far from the final solution and to reach the minimum it requires four times more time than the method by Li *et al.*

We conclude that the best solution is our initialization followed by the optimization of Li *et al.*. In practice, the implementation of our linearization smoothing iterations is almost free (incremental with respect to our initialization algorithm), whereas Li *et al.* smoothing algorithm is more difficult to implement. Moreover, in most cases few iterations of linearization steps suffice to obtain a fairly good field. As a consequence, we suggest starting with our smoothing iterations (almost free to implement), then possibly replace it with Li *et al.* smoothing algorithm if performances are not sufficient.

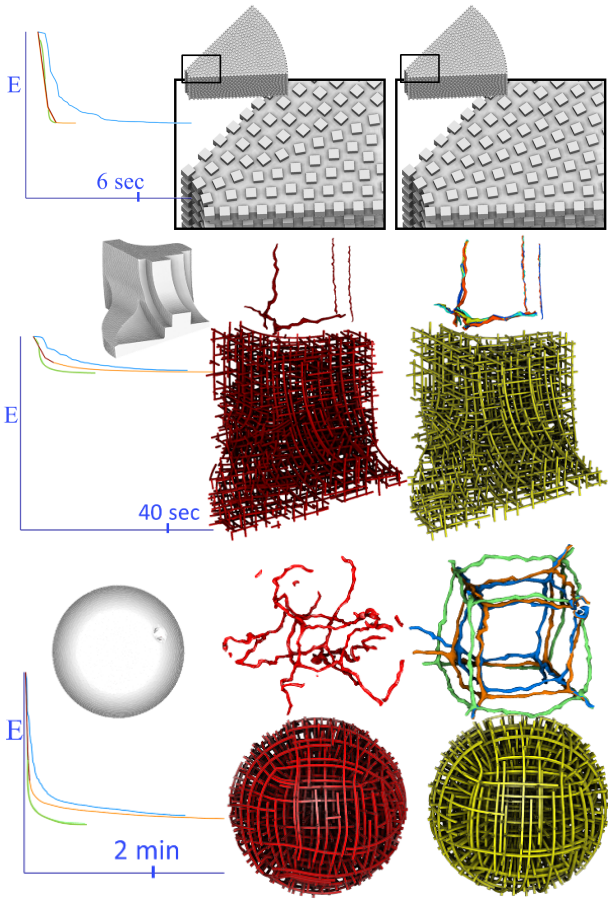

Fig. 15. Comparison of smoothing iteration algorithms combined with our initialization algorithm. We compare Huang's method (blue), Li's method (green), our method (orange), and our method limited to two iterations (red). Top row compares the initialization (left) with the field after two iterations of our algorithm (right). Middle and bottom rows compare the field with two iterations of our smoothing algorithm with other smoothing strategies. Singularity graphs reflect nicely the convergence of thee algorithms. We obtain energy (we take Li's result for 100%) of resp. 99.98%, 100%, 99.98% and 100.5% on the sector, 101.5%, 100%, 100.4%, and 106.5% on the fandisk, and 116%, 100%, 109%, 185% on the one-finger bowling ball.

Future works

From an application point of view, our method makes it fast and easy to produce smooth frame fields. The smoothness is not necessary the optimal objective for applications such as hex remeshing (see Fig. 20), but can be used as a regulation term for more complex energies. It is also very easy to modify our method to add constraints inside a volume. Figure 19 shows a frame field constrained by faults in geological data. Note that the field is not constrained by the boundary of the model. Such a frame field is useful to steer element placement for fluid simulation used in oil exploration.

From a theoritical point of view, it would be interesting to better understand the shape of the feasible set of a^i (the $3D$ manifold embedded in $9D$). It could help to find a better projection algorithm than our current gradient descent.

Conclusion

This work unifies the frame field design problem in $2D$ and $3D$. Both problems are formulated with a similar representation of frames, constraints and objective function. As a consequence, they can also be solved by similar algorithms.

From this analysis, we discovered that the best actual solution to produce smooth $3D$ frame fields is to initialize it by our proposed extension of [Kowalski et al. 2012], followed by smoothing iterations of [Li et al. 2012]. The main drawback of this solution is requirement to implement two very different approaches (a sparse linear system solver and a L-BFGS descent). A fair alternative is to use our extension of [Kowalski et al. 2012], it is simple to implement and requires a linear system solver only. In practice for our models we perform only two or three linearization iterations, however if the initialization is a bad guess (e.g. the sphere), it can be insufficient. With this approach we are able to generate (on a laptop) fields on the models up to few millions tetrahedra in less than 10 minutes, refer to Figure 18 for an illustration.

APPENDIX

A. SH COOKBOOK

A.1 $9D$ rotation

Let us denote by R^x, R^y and R^z 3×3 matrices of rotation around axes x, y and z respectively. Any frame f can be obtained as a composed rotation of the reference frame $\tilde{f}$, where the reference frame is the set of 6 unit vectors aligned with coordinate axes:

$$f = R^x(\alpha) \times R^y(\beta) \times R^z(\gamma) \times \tilde{f}.$$

If (α, β, γ) are Euler angles of rotation between a frame f and $\tilde{f}$, the representation vector a is calculated as $a = R^x_B(\alpha) \times R^y_B(\beta) \times R^z_B(\gamma) \times \tilde{a}$, where R^x_B, R^y_B and R^z_B are 9×9 matrices of rotation defined as follows:

$$R^z_B(\gamma) = \begin{bmatrix} \cos(4\gamma) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sin(4\gamma) \\ 0 & \cos(3\gamma) & 0 & 0 & 0 & 0 & 0 & \sin(3\gamma) & 0 \\ 0 & 0 & \cos(2\gamma) & 0 & 0 & 0 & \sin(2\gamma) & 0 & 0 \\ 0 & 0 & 0 & \cos(\gamma) & 0 & \sin(\gamma) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\sin(\gamma) & 0 & \cos(\gamma) & 0 & 0 & 0 \\ 0 & 0 & -\sin(2\gamma) & 0 & 0 & 0 & \cos(2\gamma) & 0 & 0 \\ 0 & -\sin(3\gamma) & 0 & 0 & 0 & 0 & 0 & \cos(3\gamma) & 0 \\ -\sin(4\gamma) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cos(4\gamma) \end{bmatrix}$$

$$R^x_B(\pi/2) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \sqrt{14}/4 & 0 & -\sqrt{2}/4 & 0 \\ 0 & -3/4 & 0 & \sqrt{7}/4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sqrt{2}/4 & 0 & \sqrt{14}/4 & 0 \\ 0 & \sqrt{7}/4 & 0 & 3/4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3/8 & 0 & \sqrt{5}/4 & 0 & \sqrt{35}/8 \\ -\sqrt{14}/4 & 0 & -\sqrt{2}/4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{5}/4 & 0 & 1/2 & 0 & -\sqrt{7}/4 \\ \sqrt{2}/4 & 0 & -\sqrt{14}/4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{35}/8 & 0 & -\sqrt{7}/4 & 0 & 1/8 \end{bmatrix}$$

$$R^y_B(\beta) = R^x_B(\pi/2) \times R^z_B(\beta) \times R^x_B(\pi/2)^\top$$

$$R^x_B(\alpha) = R^y_B(\pi/2)^\top \times R^z_B(\alpha) \times R^y_B(\pi/2)$$

These matrices are called Wigner D-matrices and the literature on their construction is vast [Collado et al. 1989; Blanco et al. 1997; Choi et al. 1999; Ivanic and Ruedenberg 1996]. However, as we are using degree 4 harmonics only, we performed the symbolic computation as follows.

The matrix $R^z_B(\gamma)$ is easy to compute, given a spherical harmonic $Y_{4,m}(\theta, \phi)$ we can rotate it around the z-axis by the angle γ by evaluating $Y_{4,m}(\theta, \phi + \gamma)$. Then the element (i, j) of the matrix R^z_B is simply $\int_0^\pi \int_0^{2\pi} Y_{4,i-5}(\theta, \phi) \cdot Y_{4,j-5}(\theta, \phi + \gamma) \sin\theta d\theta d\phi$.

Matrices $R^y_B(\beta)$ and $R^x_B(\alpha)$ are trickier to get, however we can use the fact that arbitrary rotation around the y-axis can be decomposed into the rotation around x-axis by $\pi/2$ followed by a rotation around z-axis. To rotate a spherical harmonics $Y_{4,m}(\theta, \phi)$ around the x-axis by $\pi/2$ we can perform the following substitution:

$$(\theta, \phi) \leftarrow (\arccos(-\sin(\theta)\sin(\phi)), \operatorname{atan2}(\cos(\theta), \sin(\theta)\cos(\phi))).$$

Then the matrix $R^x_B(\pi/2)$ is calculated by evaluating double integrals of all possible products between basis functions and rotated functions.

A.2 Linearization

In order to linearize the constraints in our linear solver we define matrices E^x_B, E^y_B and E^z_B as follows:

$$E^x_B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\sqrt{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\sqrt{7/2} & 0 & -\sqrt{2} \\ 0 & 0 & 0 & 0 & 0 & -3/\sqrt{2} & 0 & -\sqrt{7/2} & 0 \\ 0 & 0 & 0 & 0 & -\sqrt{10} & 0 & -3/\sqrt{2} & 0 & 0 \\ 0 & 0 & 0 & \sqrt{10} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3/\sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sqrt{7/2} & 0 & 3/\sqrt{2} & 0 & 0 & 0 & 0 & 0 \\ \sqrt{2} & 0 & \sqrt{7/2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$E^y_B = \begin{bmatrix} 0 & \sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\sqrt{2} & 0 & \sqrt{7/2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\sqrt{7/2} & 0 & 3/\sqrt{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -3/\sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\sqrt{10} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{10} & 0 & -3/\sqrt{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3/\sqrt{2} & 0 & -\sqrt{7/2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{7/2} & 0 & -\sqrt{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{2} & 0 \end{bmatrix}$$

$$E^z_B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

It is easy to verify that these matrices are chosen to verify the following equations for small rotations α, β, γ:

$$\begin{aligned} R^x_B(\alpha) &= I_{9\times 9} + \alpha E^x_B + o(|\alpha|) \\ R^y_B(\beta) &= I_{9\times 9} + \beta E^y_B + o(|\beta|) \\ R^z_B(\gamma) &= I_{9\times 9} + \gamma E^z_B + o(|\gamma|). \end{aligned}$$

Finally, for small rotations the multiplication is commutative:

$$\begin{aligned} R_B(\alpha, \beta, \gamma) &= R^x_B(\alpha) \times R^y_B(\beta) \times R^z_B(\gamma) = \\ &= I_{9\times 9} + \alpha E^x_B + \beta E^y_B + \gamma E^z_B + o(|\alpha| + |\beta| + |\gamma|). \end{aligned}$$

REFERENCES

BLANCO, M. A., FLÓREZ, M., AND BERMEJO, M. 1997. Evaluation of the rotation matrices in the basis of real spherical harmonics. *Journal of Molecular Structure: {THEOCHEM} 419,* 1–3, 19 – 27.

BOMMES, D., CAMPEN, M., EBKE, H.-C., ALLIEZ, P., AND KOBBELT, L. 2013. Integer-grid maps for reliable quad meshing. *ACM Trans. Graph. 32,* 4 (July), 98:1–98:12.

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

CHOI, C. H., IVANIC, J., GORDON, M. S., AND RUEDENBERG, K. 1999. Rapid and stable determination of rotation matrices between spherical harmonics by direct recursion. *The Journal of Chemical Physics 111,* 19, 8825–8831.

COLLADO, J. R. A., RICO, J. F., LÓPEZ, R., PANIAGUA, M., AND RAMÍREZ, G. 1989. Rotation of real spherical harmonics. *Computer Physics Communications 52,* 3, 323 – 331.

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.

IVANIC, J. AND RUEDENBERG, K. 1996. Rotation matrices for real spherical harmonics. direct determination by recursion. *The Journal of Physical Chemistry 100,* 15, 6342–6347.

JIANG, T., HUANG, J., WANG, Y., TONG, Y., AND BAO, H. 2014. Frame field singularity correctionfor 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.

LÉVY, B. OPENNL, Open Numerical Library. http://alice.loria.fr/index.php/software/4-library/23-opennl.html.

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

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

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

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

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.

Received September 2008; accepted March 2009

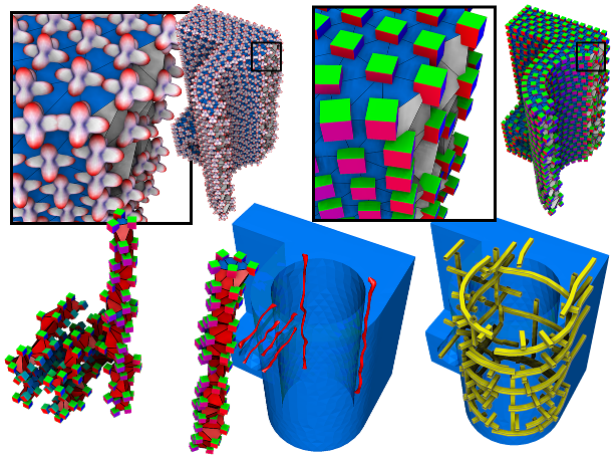

Fig. 16. Our results are shown using combinations of the following rendering techniques. We can plot for each vertex i its F^i (upper left), or its frame as a cube(upper right). We can show the singular tets (lower left), or a smoothed and refined version (lower middle) to better see it in $3D$ (thanks to the lighting). The field inside the volume can be rendered by curved french fries.

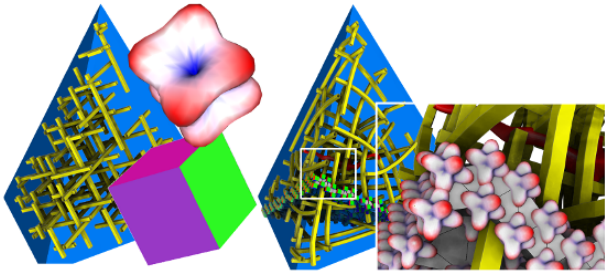

Fig. 17. The initialization of [Huang et al. 2011] (left) is a constant frame field whereas ours (right) is aligned to the boundary. Their F^i are all equal, and very far from being feasible, making it possible to violate the boundary condition.

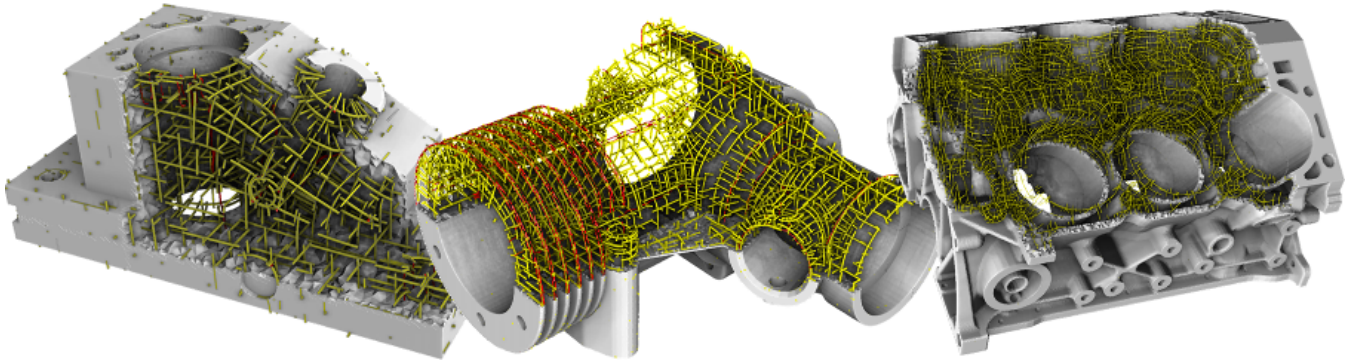

Fig. 18. Results on CAD objects. Names are (from left to right): Anc, Crank, 40head.

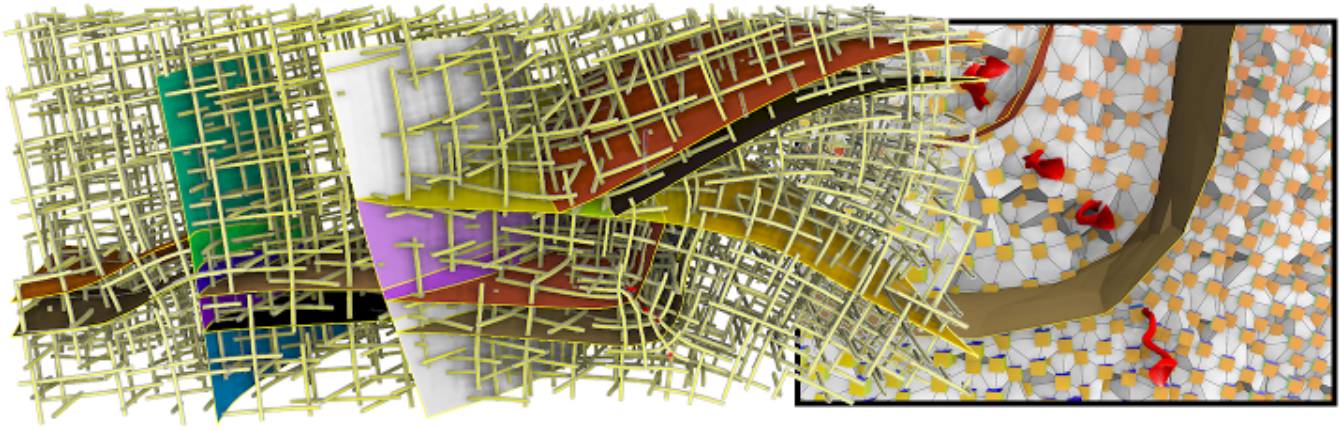

Fig. 19. 3D frame field constrained by faults in geological data.

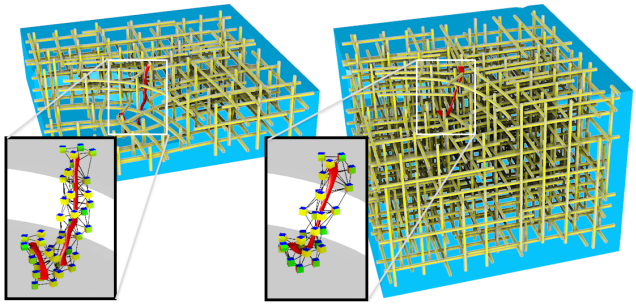

Fig. 20. The thin object (left) frame field is suitable to produce a hex mesh: its singularity graph is basically two singularities of index $1/4$ and $-1/4$ extruded in the z direction. The frame field of fat object (right) has a singularity graph that does not correspond to a hex mesh.