

Mathematics and Implementation of
Computer Graphics Techniques 2015

Boundary Aligned Smooth 3D Cross-Frame Field

Jin Huang, Yiyong Tong, Hongyu Wei, Hujun Bao
SIGGRAPH Asia 2011

Kenshi Takayama

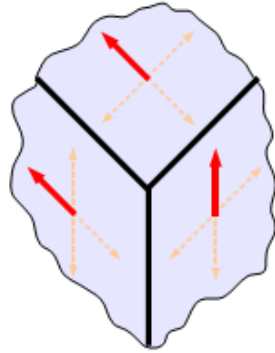
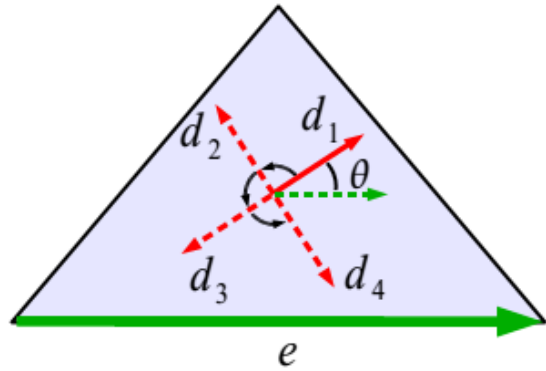
Assistant Prof @ National Institute of Informatics

takayama@nii.ac.jp

2nd round

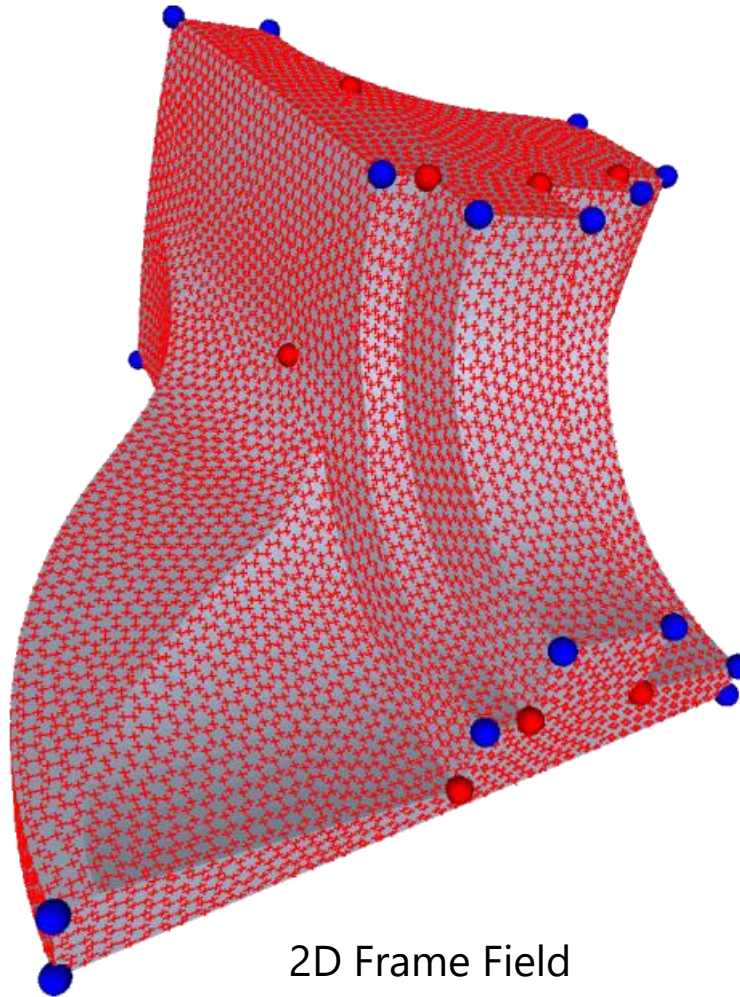
3 October 2015

Background: 2D Frame Field & Quad Meshing

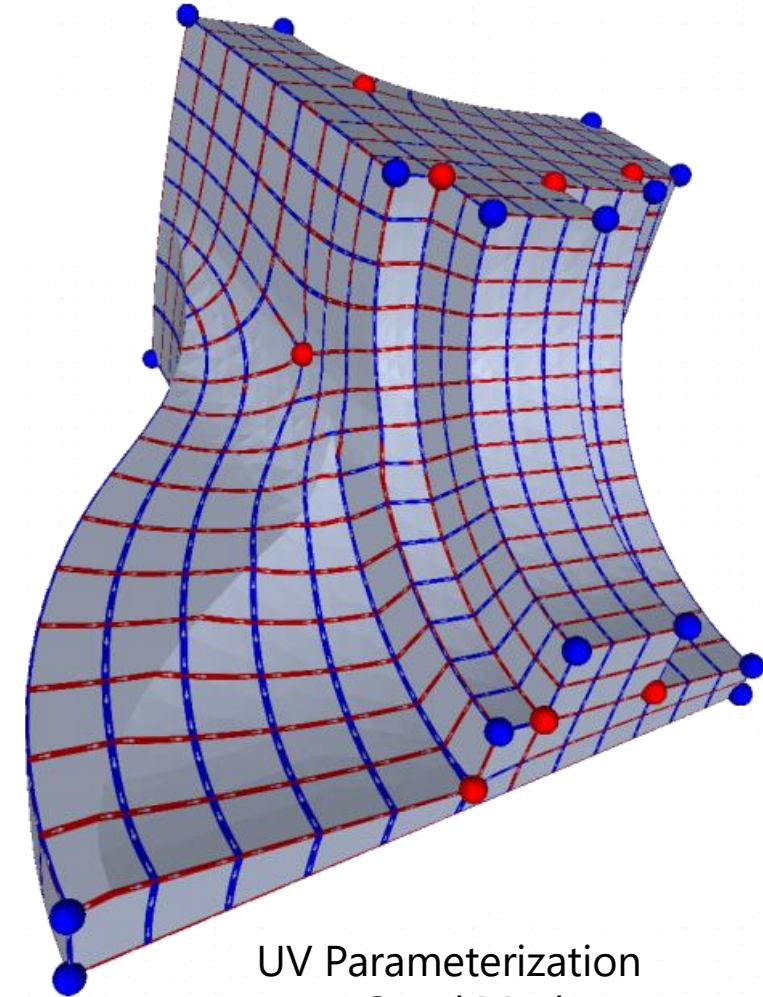


- 2D Frame Field
← Auto-computed

- UV Parameterization
← Auto-computed

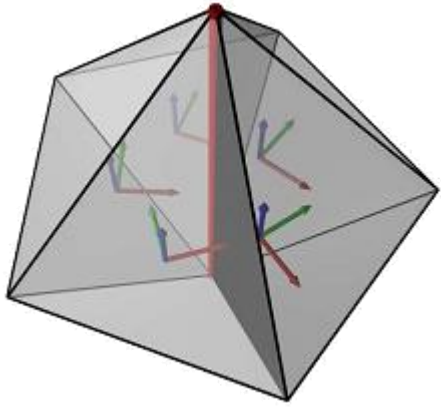


2D Frame Field



UV Parameterization
= Quad Mesh

Background: 3D Frame Field & Hex Meshing

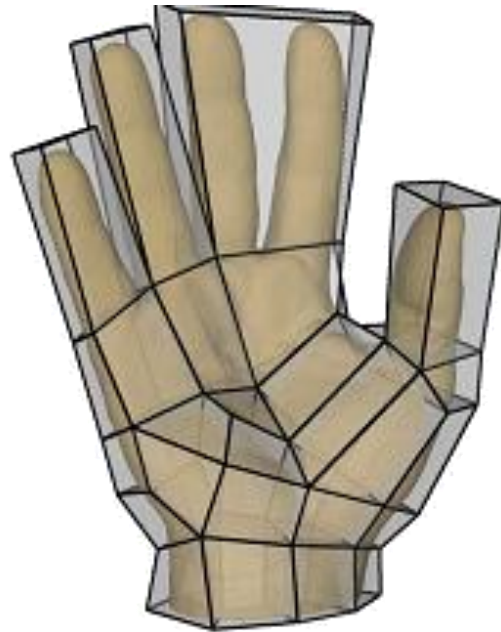


- 3D Frame Field

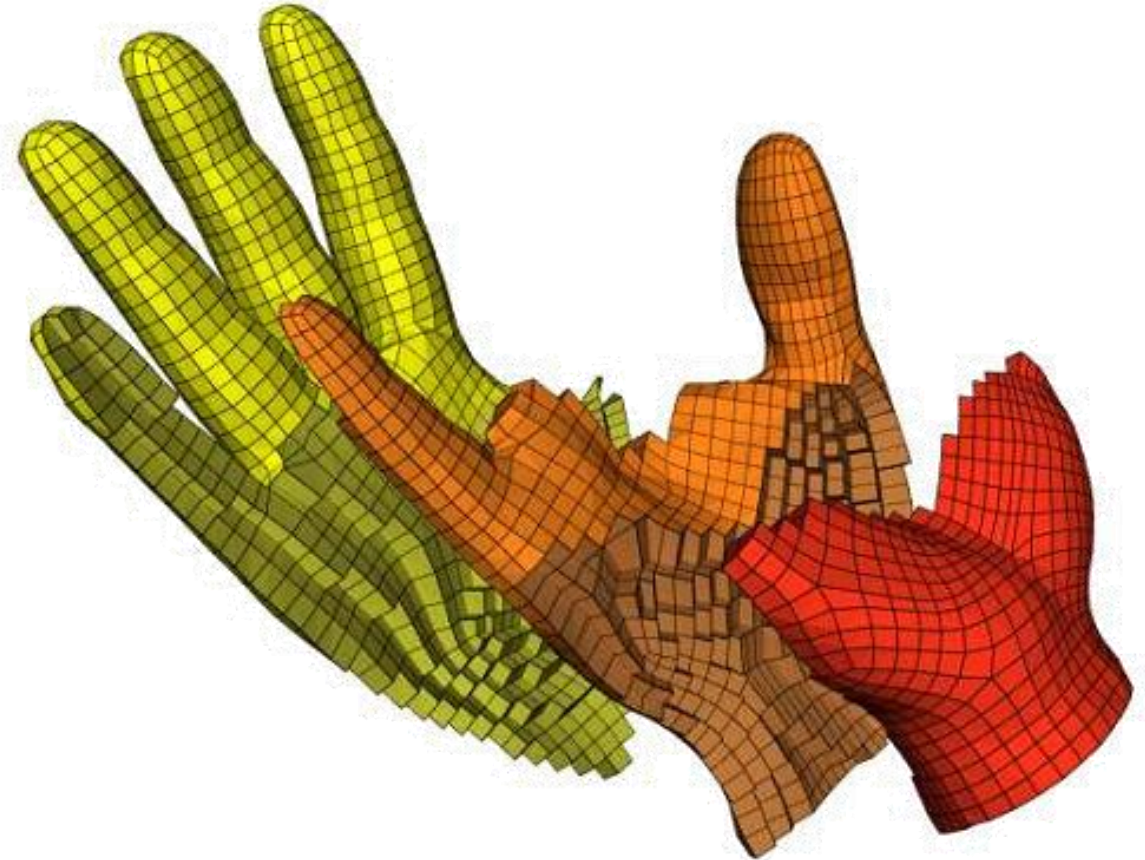
← Heuristic

- UVW Parameterization

← Auto-computed

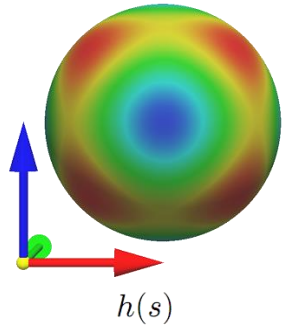


"Meta-Mesh" to define
3D Frame Field

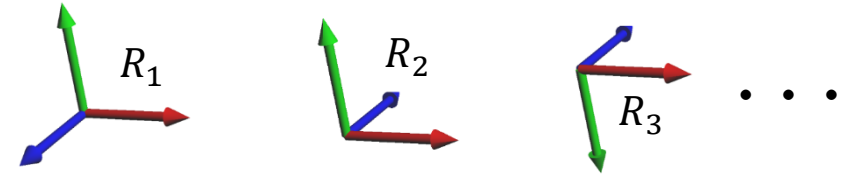


UVW Parameterization
= Hex Mesh

Distance between 3D Frames



$$h(s) := s_x^2 s_y^2 + s_y^2 s_z^2 + s_z^2 s_x^2$$



$$d(R_a, R_b) := \int_{s \in S^2} \left[\begin{array}{c} \text{Sphere } h(R_a s) \\ - \\ \text{Sphere } h(R_b s) \end{array} \right]^2 ds$$

- Integral over an entire sphere → **Spherical Harmonics!**

Representing 3D frames using SH coeffs

$$\begin{aligned}
 h(Rs) &= \lambda_{-4} \times \begin{matrix} \text{Y}_4^{-4} \\ \text{Y}_4^{-4} \end{matrix} + \lambda_{-3} \times \begin{matrix} \text{Y}_4^{-3} \\ \text{Y}_4^{-3} \end{matrix} + \lambda_{-2} \times \begin{matrix} \text{Y}_4^{-2} \\ \text{Y}_4^{-2} \end{matrix} \\
 &+ \lambda_{-1} \times \begin{matrix} \text{Y}_4^{-1} \\ \text{Y}_4^{-1} \end{matrix} + \lambda_0 \times \begin{matrix} \text{Y}_4^0 \\ \text{Y}_4^0 \end{matrix} + \lambda_1 \times \begin{matrix} \text{Y}_4^1 \\ \text{Y}_4^1 \end{matrix} \\
 &+ \lambda_2 \times \begin{matrix} \text{Y}_4^2 \\ \text{Y}_4^2 \end{matrix} + \lambda_3 \times \begin{matrix} \text{Y}_4^3 \\ \text{Y}_4^3 \end{matrix} + \lambda_4 \times \begin{matrix} \text{Y}_4^4 \\ \text{Y}_4^4 \end{matrix}
 \end{aligned}$$

$$\begin{aligned}
 h(s) &= s_x^2 s_y^2 + s_y^2 s_z^2 + s_z^2 s_x^2 \\
 &= \sqrt{7} \times \begin{matrix} \text{Y}_4^0 \\ \text{Y}_4^0 \end{matrix} + \sqrt{5} \times \begin{matrix} \text{Y}_4^4 \\ \text{Y}_4^4 \end{matrix}
 \end{aligned}$$

$$\begin{pmatrix} \lambda_{-4} \\ \lambda_{-3} \\ \lambda_{-2} \\ \lambda_{-1} \\ \lambda_0 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{pmatrix} = \begin{pmatrix} \hat{R} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \sqrt{7} \\ 0 \\ 0 \\ 0 \\ \sqrt{5} \end{pmatrix}$$

Representing \hat{R} using ZYZ Euler angles

$$R(\alpha, \beta, \gamma) = R_Z(\gamma) R_Y(\beta) R_Z(\alpha)$$

Rotation about Z axis by α

$$= R_Z(\gamma) R_X\left(-\frac{\pi}{2}\right) R_Z(\beta) R_X\left(\frac{\pi}{2}\right) R_Z(\alpha)$$

$$= \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \cos \beta & -\sin \beta & 0 \\ \sin \beta & \cos \beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\hat{R}(\alpha, \beta, \gamma) = \hat{R}_Z(\gamma) \hat{R}_X\left(-\frac{\pi}{2}\right) \hat{R}_Z(\beta) \hat{R}_X\left(\frac{\pi}{2}\right) \hat{R}_Z(\alpha)$$

Deriving $\hat{R}_Z(\alpha)$ & $\hat{R}_X\left(\frac{\pi}{2}\right)$

https://en.wikipedia.org/wiki/Table_of_spherical_harmonics#Real_spherical_harmonics

$$Y_4^{-4}(x, y, z) = \frac{3}{4} \sqrt{\frac{35}{\pi}} xy(x^2 - y^2)$$

$$Y_4^0(x, y, z) = \frac{3}{16} \sqrt{\frac{1}{\pi}} (35z^4 - 30z^2 + 3)$$

$$Y_4^4(x, y, z) = \frac{3}{16} \sqrt{\frac{35}{\pi}} (x^2(x^2 - 3y^2) - y^2(3x^2 - y^2))$$

$$Y_4^{-3}(x, y, z) = \frac{3}{4} \sqrt{\frac{35}{2\pi}} (3x^2 - y^2)yz$$

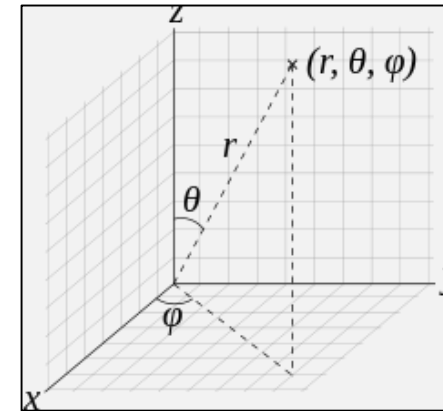
$$Y_4^1(x, y, z) = \frac{3}{4} \sqrt{\frac{5}{2\pi}} xz(7z^2 - 3)$$

$$Y_4^{-2}(x, y, z) = \frac{3}{4} \sqrt{\frac{5}{\pi}} xy(7z^2 - 1)$$

$$Y_4^2(x, y, z) = \frac{3}{8} \sqrt{\frac{5}{\pi}} (x^2 - y^2)(7z^2 - 1)$$

$$Y_4^{-1}(x, y, z) = \frac{3}{4} \sqrt{\frac{5}{2\pi}} yz(7z^2 - 3)$$

$$Y_4^3(x, y, z) = \frac{3}{4} \sqrt{\frac{35}{2\pi}} (x^2 - 3y^2)xz$$



$$\begin{aligned} x &= \sin \theta \cos \phi \\ y &= \sin \theta \sin \phi \\ z &= \cos \theta \end{aligned}$$

$$\hat{R}_Z(\alpha)_{i,j} = \int_0^{2\pi} \int_0^\pi Y_4^i(\sin \theta \cos(\phi + \alpha), \sin \theta \sin(\phi + \alpha), \cos \theta) \cdot Y_4^j(\sin \theta \cos \phi, \sin \theta \sin \phi, \cos \theta) \sin \theta d\theta d\phi$$

$$\hat{R}_X\left(\frac{\pi}{2}\right)_{i,j} = \int_0^{2\pi} \int_0^\pi Y_4^i(\sin \theta \cos \phi, -\cos \theta, \sin \theta \sin \phi) \cdot Y_4^j(\sin \theta \cos \phi, \sin \theta \sin \phi, \cos \theta) \sin \theta d\theta d\phi$$

→ use Computer Algebra Systems (CAS): Mathematica, Maple, Sage

Sage code (<https://cloud.sagemath.com>)

```
Y4_4(x,y,z) = 3/4*sqrt(35/pi)*x*y*(x^2 - y^2)
Y4_3(x,y,z) = 3/4*sqrt(35/(2*pi))*(3*x^2 - y^2)*y*z
Y4_2(x,y,z) = 3/4*sqrt(5/pi)*x*y*(7*z^2 - 1)
Y4_1(x,y,z) = 3/4*sqrt(5/(2*pi))*y*z*(7*z^2 - 3)
Y40(x,y,z) = 3/16*sqrt(1/pi)*(35*z^4 - 30*z^2 + 3)
Y41(x,y,z) = 3/4*sqrt(5/(2*pi))*x*z*(7*z^2 - 3)
Y42(x,y,z) = 3/8*sqrt(5/pi)*(x^2 - y^2)*(7*z^2 - 1)
Y43(x,y,z) = 3/4*sqrt(35/(2*pi))*(x^2 - 3*y^2)*x*z
Y44(x,y,z) = 3/16*sqrt(35/pi)*(x^2*(x^2 - 3*y^2) - y^2*(3*x^2 - y^2))

Y4 = [Y4_4, Y4_3, Y4_2, Y4_1, Y40, Y41, Y42, Y43, Y44]

for i in range(9):
    v = []
    for j in range(9):
        Si(theta, phi) = Y4[i](sin(theta)*cos(phi+a), sin(theta)*sin(phi+a), cos(theta))
        Sj(theta, phi) = Y4[j](sin(theta)*cos(phi), sin(theta)*sin(phi), cos(theta))
        v.append(integral(Si(theta, phi) * Sj(theta, phi) * sin(theta), theta, 0, pi).integrate(phi, 0, 2*pi))
    print v

for i in range(9):
    v = []
    for j in range(9):
        Si(theta, phi) = Y4[i](sin(theta)*cos(phi), -cos(theta), sin(theta)*sin(phi))
        Sj(theta, phi) = Y4[j](sin(theta)*cos(phi), sin(theta)*sin(phi), cos(theta))
        v.append(integral(Si(theta, phi) * Sj(theta, phi) * sin(theta), theta, 0, pi).integrate(phi, 0, 2*pi))
    print v
```


Matrices written down

$$\hat{R}_Z(\alpha) = \begin{pmatrix} \cos 4\alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sin 4\alpha \\ 0 & \cos 3\alpha & 0 & 0 & 0 & 0 & 0 & \sin 3\alpha & 0 \\ 0 & 0 & \cos 2\alpha & 0 & 0 & 0 & \sin 2\alpha & 0 & 0 \\ 0 & 0 & 0 & \cos \alpha & 0 & \sin \alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\sin \alpha & 0 & \cos \alpha & 0 & 0 & 0 \\ 0 & 0 & -\sin 2\alpha & 0 & 0 & 0 & \cos 2\alpha & 0 & 0 \\ 0 & -\sin 3\alpha & 0 & 0 & 0 & 0 & 0 & \cos 3\alpha & 0 \\ -\sin 4\alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cos 4\alpha \end{pmatrix}$$

$$\hat{R}_X\left(\frac{\pi}{2}\right) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & \sqrt{14}/4 & 0 & -\sqrt{2}/4 & 0 \\ 0 & -3/4 & 0 & \sqrt{7}/4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sqrt{2}/4 & 0 & \sqrt{14}/4 & 0 \\ 0 & \sqrt{7}/4 & 0 & 3/4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3/8 & 0 & \sqrt{5}/4 & 0 & \sqrt{35}/8 \\ -\sqrt{14}/4 & 0 & -\sqrt{2}/4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{5}/4 & 0 & 1/2 & 0 & -\sqrt{7}/4 \\ \sqrt{2}/4 & 0 & -\sqrt{14}/4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{35}/8 & 0 & -\sqrt{7}/4 & 0 & 1/8 \end{pmatrix}$$

Going between ZYZ (3D) \Leftrightarrow SH (9D)

- ZYZ \rightarrow SH

$$\mathbf{f}(\alpha, \beta, \gamma) := \hat{R}_Z(\gamma) \cdot \hat{R}_X\left(-\frac{\pi}{2}\right) \cdot \hat{R}_Z(\beta) \cdot \hat{R}_X\left(\frac{\pi}{2}\right) \cdot \hat{R}_Z(\alpha) \cdot \hat{h}$$

$$\hat{h} := \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \sqrt{7} \\ 0 \\ 0 \\ 0 \\ \sqrt{5} \end{pmatrix}$$

- SH \rightarrow ZYZ

$$\mathbf{g}(\mathbf{f}_{\text{in}}) := \underset{(\alpha, \beta, \gamma) \in \mathbb{R}^3}{\text{arg min}} \|\mathbf{f}_{\text{in}} - \mathbf{f}(\alpha, \beta, \gamma)\|^2$$

- Minimize $E(\alpha, \beta, \gamma)$ using Conjugate Gradient

$$\frac{dE}{d\alpha} = -2(\mathbf{f}_{\text{in}} - \mathbf{f}(\alpha, \beta, \gamma))^\top \frac{d\mathbf{f}}{d\alpha}$$

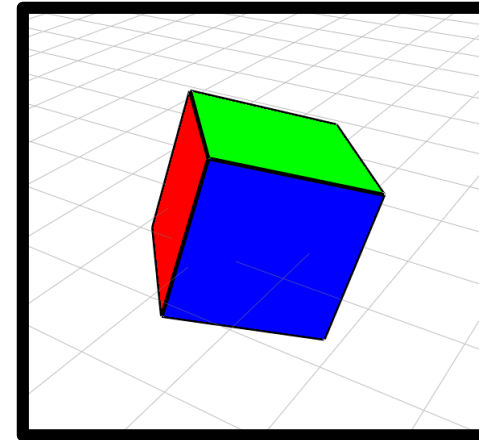
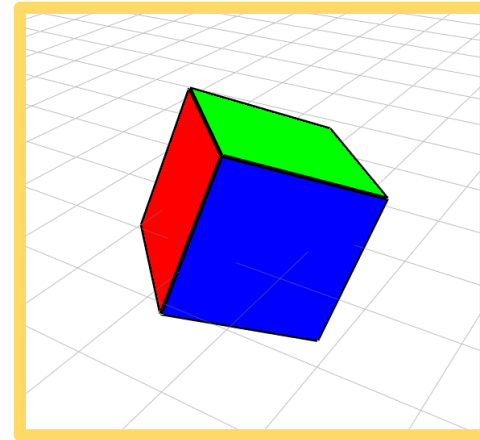
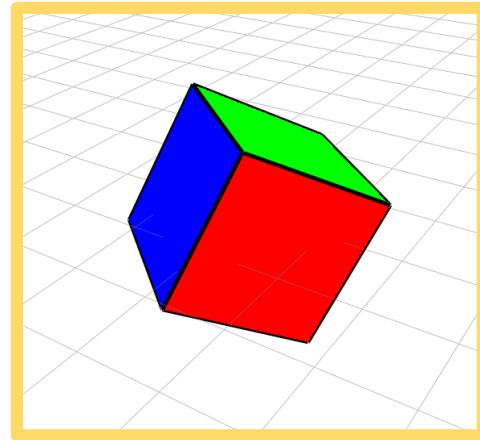
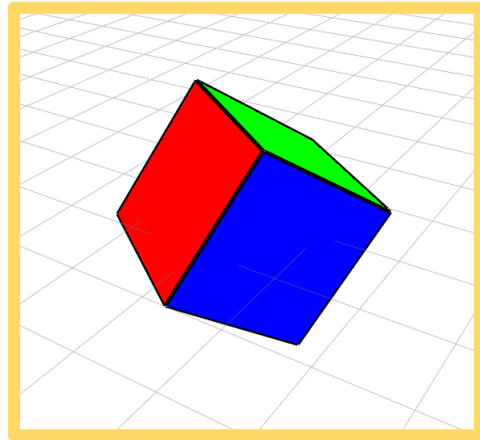
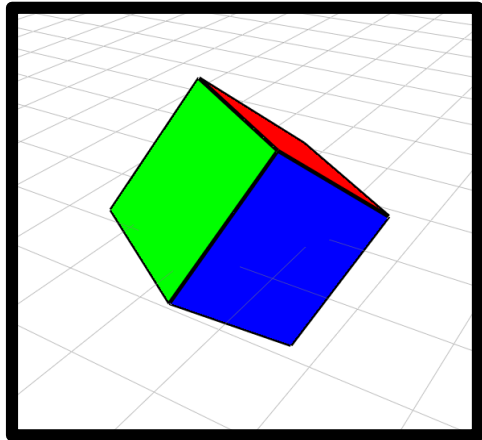
$$\frac{d\mathbf{f}}{d\alpha} = \hat{R}_Z(\gamma) \cdot \hat{R}_X\left(-\frac{\pi}{2}\right) \cdot \hat{R}_Z(\beta) \cdot \hat{R}_X\left(\frac{\pi}{2}\right) \cdot \frac{d\hat{R}_Z}{d\alpha} \cdot \hat{h}$$

$$\frac{d\hat{R}_Z}{d\alpha} = \begin{pmatrix} -4 \sin 4\alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \cos 4\alpha \\ 0 & -3 \sin 3\alpha & 0 & 0 & 0 & 0 & 0 & 3 \cos 3\alpha & 0 \\ 0 & 0 & -2 \sin 2\alpha & 0 & 0 & 0 & 2 \cos 2\alpha & 0 & 0 \\ 0 & 0 & 0 & -\sin \alpha & 0 & \cos \alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\cos \alpha & 0 & -\sin \alpha & 0 & 0 & 0 \\ 0 & 0 & -2 \cos 2\alpha & 0 & 0 & 0 & -2 \sin 2\alpha & 0 & 0 \\ 0 & -3 \cos 3\alpha & 0 & 0 & 0 & 0 & 0 & -3 \sin 3\alpha & 0 \\ -4 \cos 4\alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4 \sin 4\alpha \end{pmatrix}$$

Toy example: interpolating two frames

$t = 0$ ←

→ $t = 1$

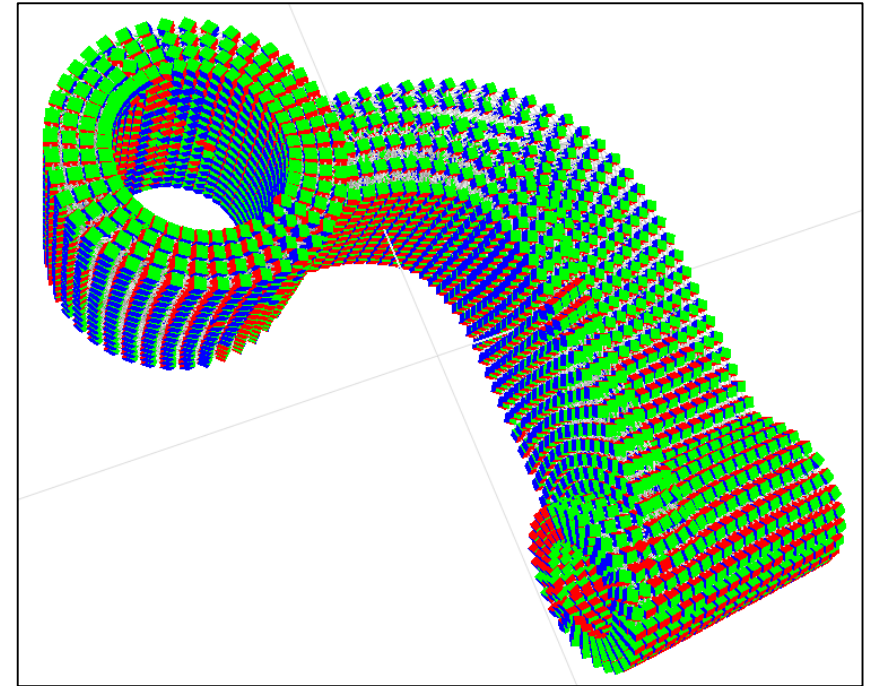
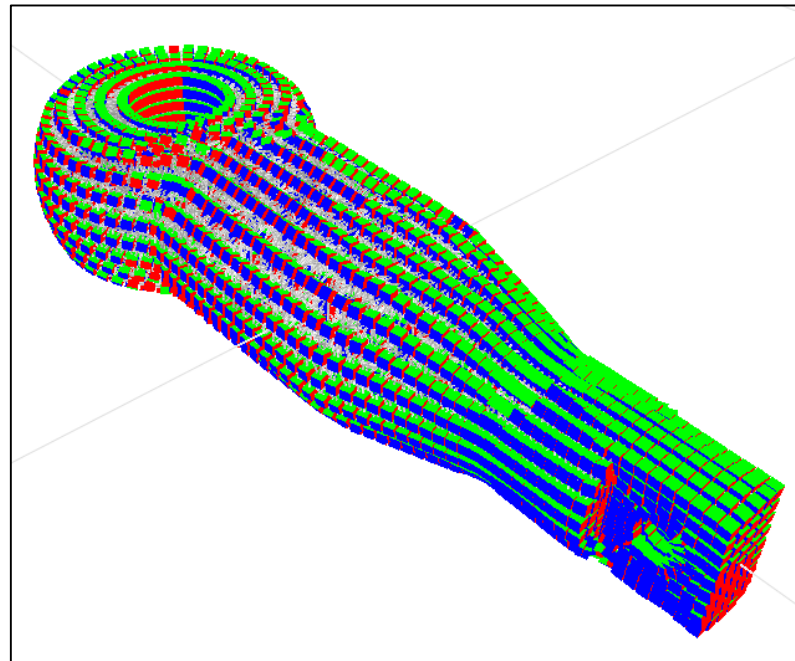
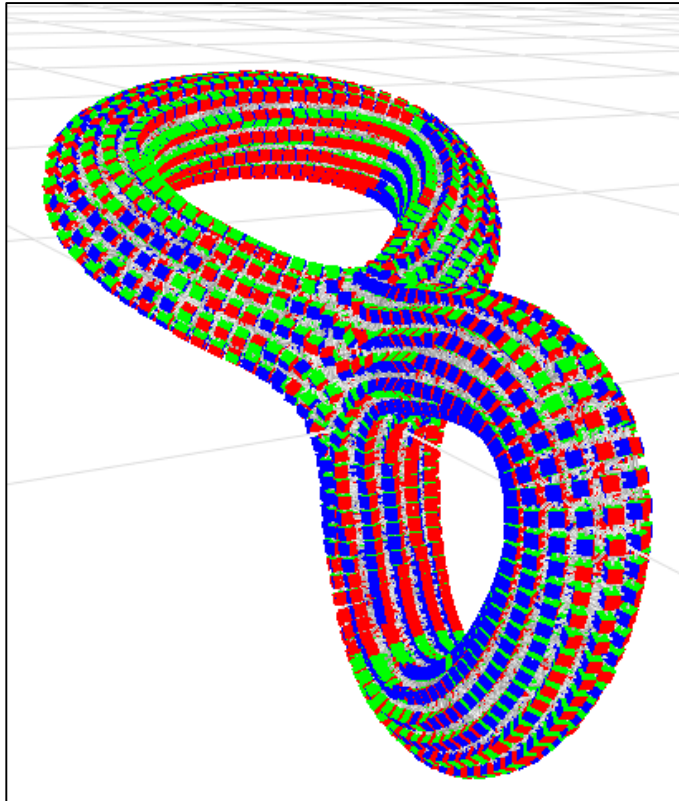
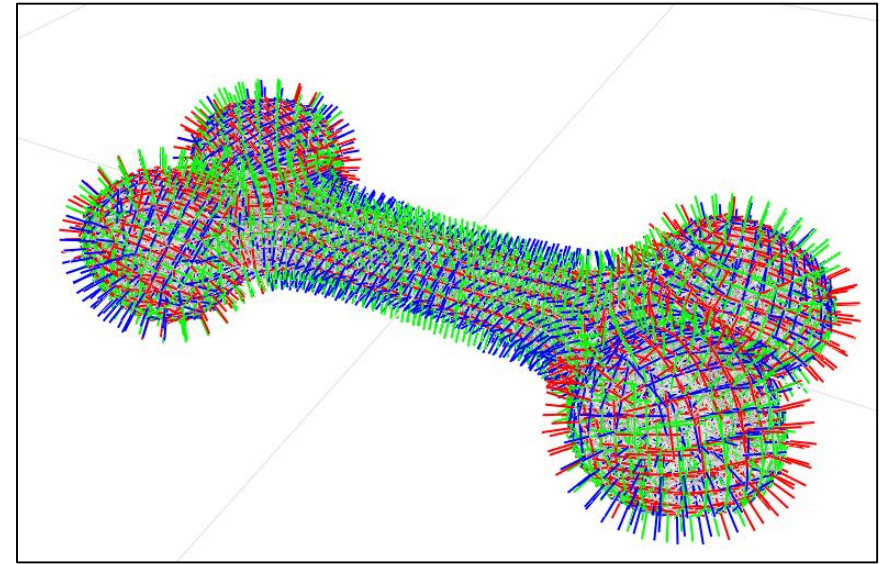
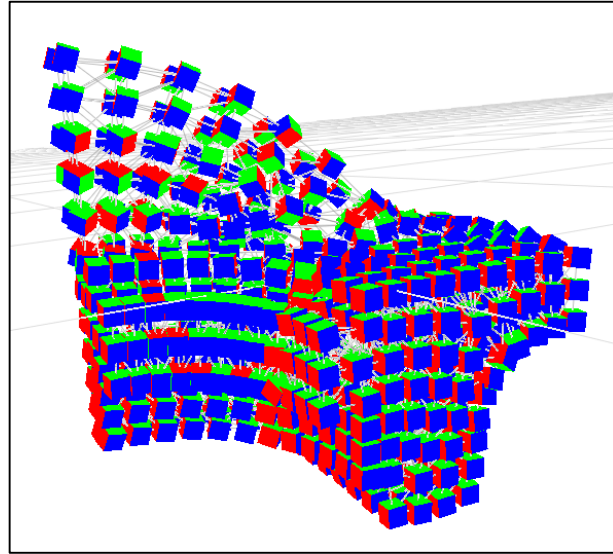
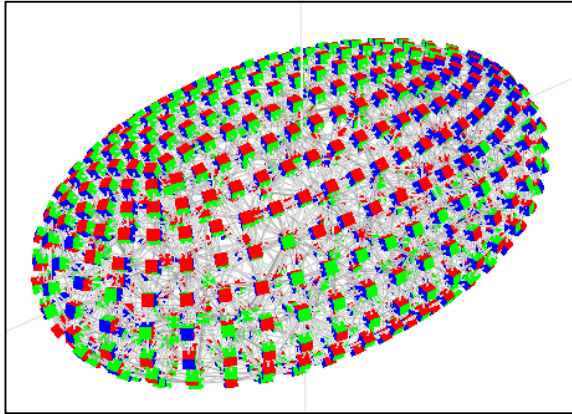


$$\mathbf{f}_0 = \mathbf{f}(\alpha_0, \beta_0, \gamma_0)$$
$$\alpha_0 = 53^\circ$$
$$\beta_0 = 60^\circ$$
$$\gamma_0 = 356^\circ$$

$$(\alpha_t, \beta_t, \gamma_t) = \underset{(\alpha, \beta, \gamma) \in \mathbb{R}^3}{\operatorname{arg\,min}} \|(1-t)\mathbf{f}_0 + t\mathbf{f}_1 - \mathbf{f}(\alpha, \beta, \gamma)\|^2$$

$$\mathbf{f}_1 = \mathbf{f}(\alpha_1, \beta_1, \gamma_1)$$
$$\alpha_1 = 160^\circ$$
$$\beta_1 = 43^\circ$$
$$\gamma_1 = 2^\circ$$

Real examples with tetrahedral meshes



Small differences from [Huang11]

- Per-vertex discretization
 - \Leftrightarrow per-face (Crouzeix-Raviart) discretization
 - $\#vertices \ll \#faces \rightarrow$ much smaller problem size (x0.1)
 - Normal alignment properly handled
- No global nonlinear solve w.r.t. $\{ (\alpha_i, \beta_i, \gamma_i) \}$
 - Only Laplacian smoothing + per-vertex nonlinear projection

Recent arXiv paper [Ray & Sokolov 2015]

- Unified view toward 2D & 3D problems
- Better handling of normal alignment
- “Feasibility” constraint linearized & integrated into iterative solve
- SH cookbook, concise pseudocode

arXiv:1507.03351v1 [cs.GR] 13 Jul 2015

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). The examples show 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 minimize 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, π -meshing
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 vectors). 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 from 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 §4.5, the only required tool is a linear solver, all calculations are given explicitly.*

The 2D algorithm with frames represented by functions

Solutions developed for 2D are very different from 3D 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 formalize the corresponding optimization

ACM Transactions on Graphics, Vol. 34, No. 4, Article 106, Publication date: August 2005.

Authors' address: Nicolas.Ray@inria.fr, dmitry.sokolov@inria.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 10133-0701 USA, fax +1 (212) 609-6481, or permissions@acm.org.
© 2009 ACM 0730-0301/2009/13-ART106 \$10.00
DOI: 10.1145/1559755.1559763
http://dx.doi.org/10.1145/1559755.1559763

<http://arxiv.org/abs/1507.03351>

Small ideas for further improvements

- Fix 3D frames on boundary using 2D frames
 - Decouple 9 SH coeffs \rightarrow x1/9 dimensionality
- Jacobi-style iteration \rightarrow simple & parallel
- Other scattered-data-interpolation tools (RBF / MLS) ?
- (Not my main focus anyway)

