252-0538-00L, Spring 2018

Shape Modeling and Geometry Processing

Smoothing (continued)
Deformations

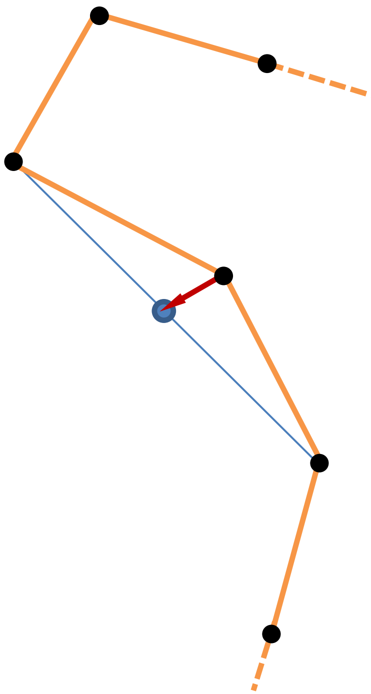
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Smoothing by flowing

Example – smoothing curves

Laplace in 1D = second derivative:

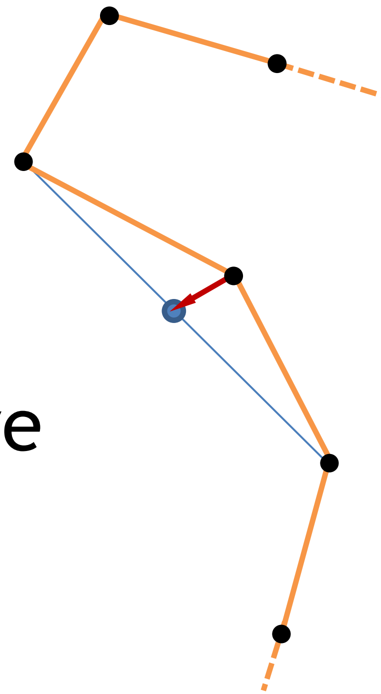
$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i)$$



Example – smoothing curves

Laplace in 1D = second derivative:

$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i)$$



In matrix-vector form for the whole curve

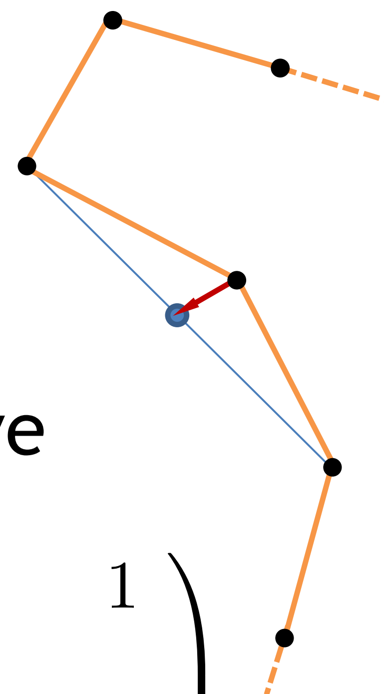
$$L\mathbf{p}$$

$$\mathbf{p} = [\mathbf{x}\ \mathbf{y}] \in \mathbb{R}^{n\times 2}$$

Example – smoothing curves

Laplace in 1D = second derivative:

$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i)$$



In matrix-vector form for the whole curve

$$L\mathbf{p}$$

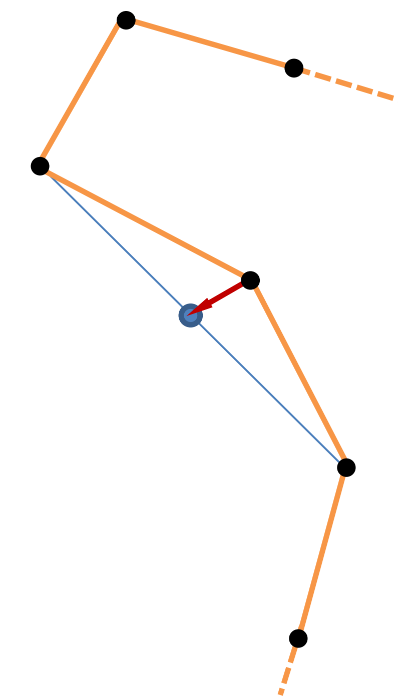
$$\mathbf{p} = [\mathbf{x}\ \mathbf{y}] \in \mathbb{R}^{n\times 2}$$

$$L = \frac{1}{2}\begin{pmatrix} -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 1 & & & 1 & -2 \end{pmatrix}$$

Example – smoothing curves

Flow to reduce curvature:

$$\tilde{\mathbf{p}}_i = \mathbf{p}_i + \lambda \frac{d^2}{ds^2}(\mathbf{p}_i)$$



Scale factor $0 < \lambda < 1$

Matrix-vector form:

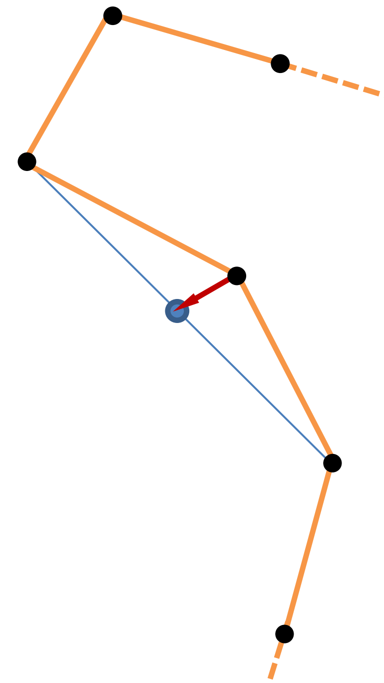
$$\tilde{\mathbf{p}} = \mathbf{p} + \lambda L \mathbf{p}, \quad \mathbf{p} \in \mathbb{R}^{n \times 2}$$

ETH zürich

Example – smoothing curves

Flow to reduce curvature:

$$\tilde{\mathbf{p}}_i = \mathbf{p}_i + \lambda L(\mathbf{p}_i) = \mathbf{p}_i + \lambda \frac{d^2}{ds^2}(\mathbf{p}_i)$$



Scale factor $0 < \lambda < 1$

Matrix-vector form:

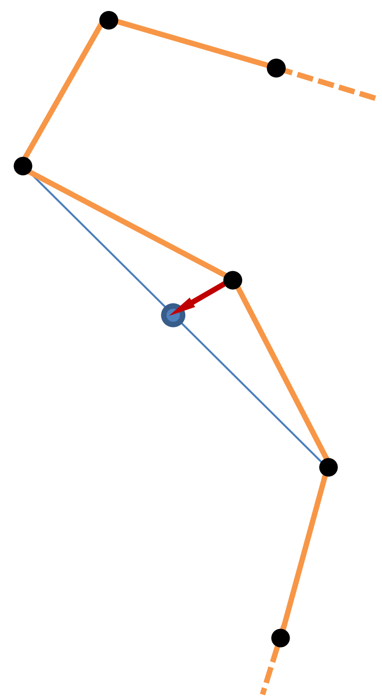
$$\tilde{\mathbf{p}} = \mathbf{p} + \lambda L \mathbf{p}, \quad \mathbf{p} \in \mathbb{R}^{n \times 2}$$

Drawbacks?

Example – smoothing curves

Flow to reduce curvature:

$$\tilde{\mathbf{p}}_i = \mathbf{p}_i + \lambda\, L(\mathbf{p}_i) = \mathbf{p}_i + \lambda \frac{d^2}{ds^2}(\mathbf{p}_i)$$



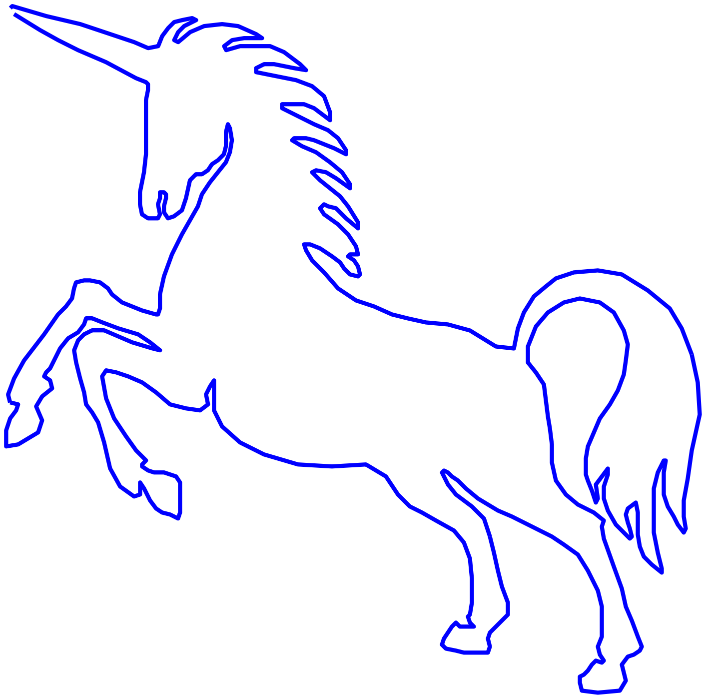
Scale factor $0 < \lambda < 1$

Matrix-vector form:

$$\tilde{\mathbf{p}} = \mathbf{p} + \lambda L \mathbf{p}, \quad \mathbf{p} \in \mathbb{R}^{n \times 2}$$

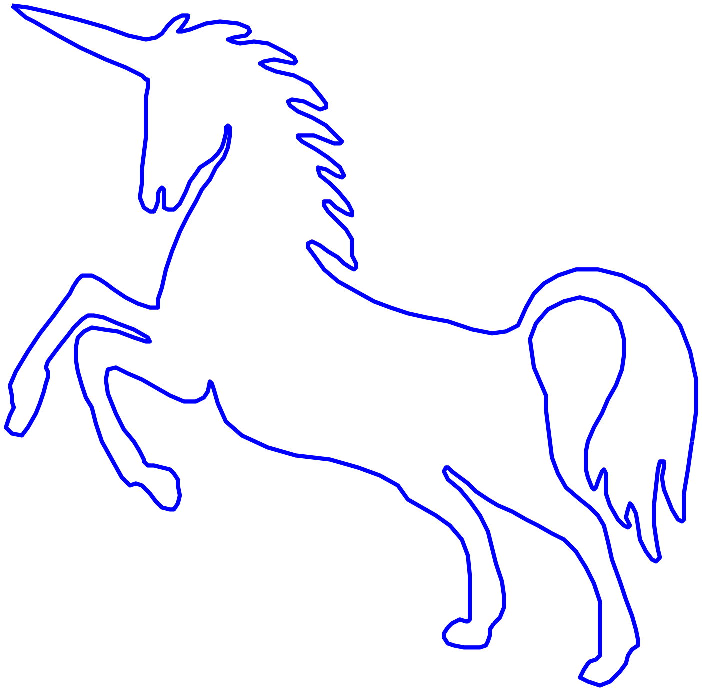
May shrink the shape; can be slow

Filtering Curves



Original curve

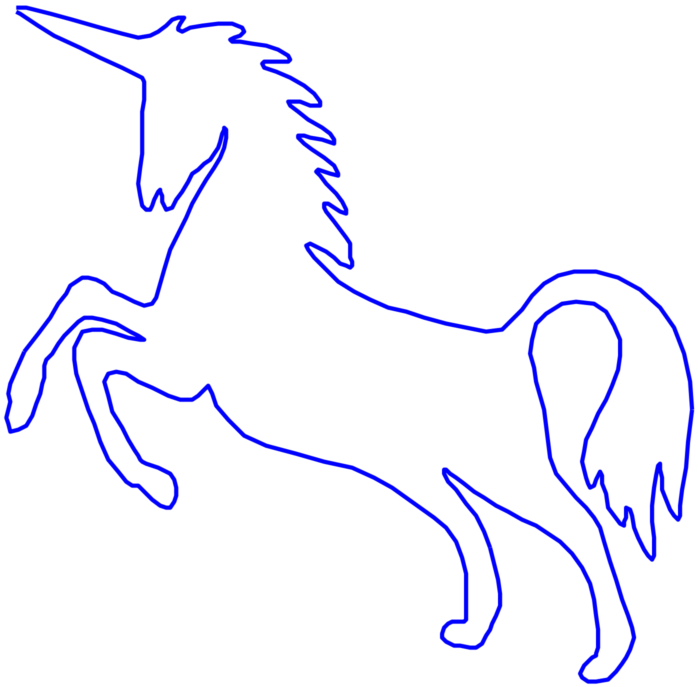
Filtering Curves



1st iteration; λ=0.5

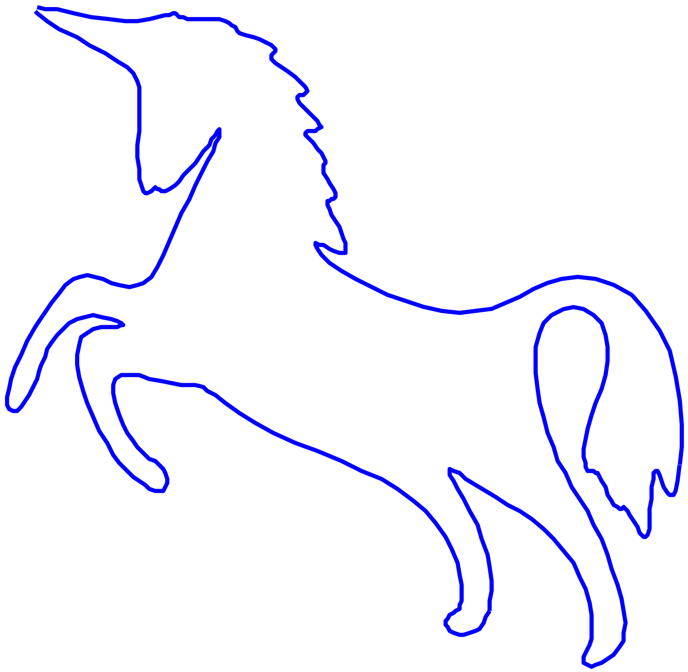
ETH zürich

Filtering Curves



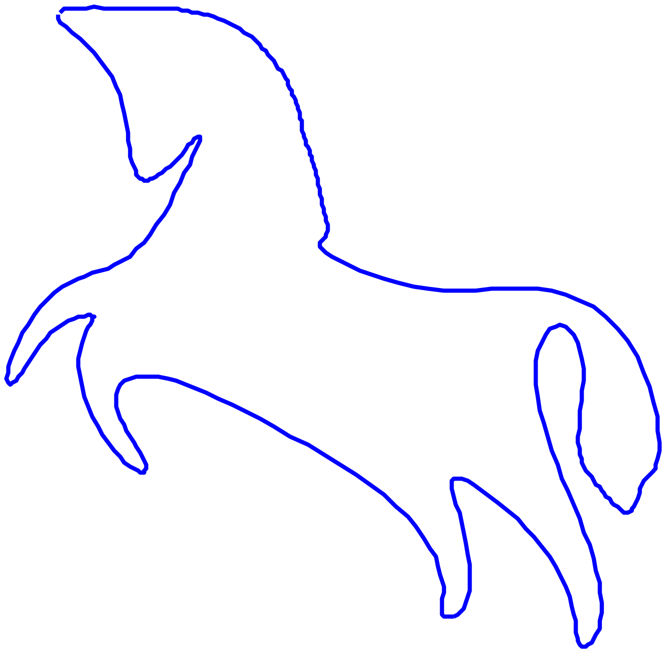
2nd iteration; λ=0.5

Filtering Curves



8th iteration; λ=0.5

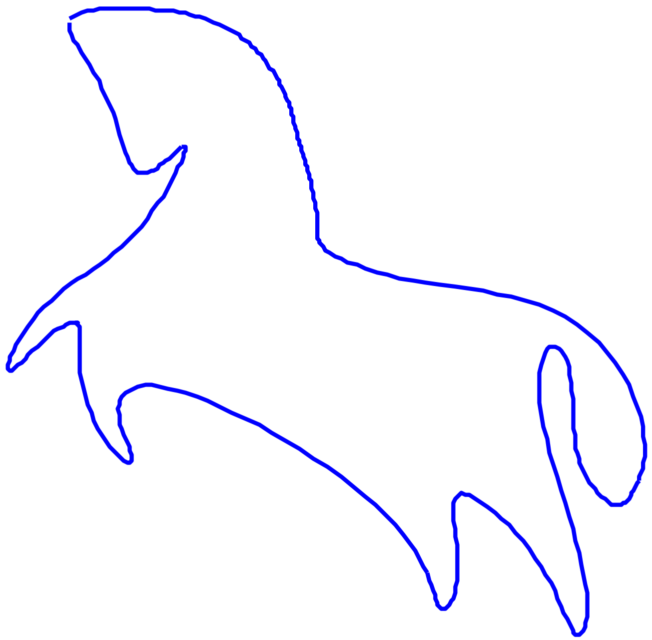
Filtering Curves



27th iteration; λ=0.5

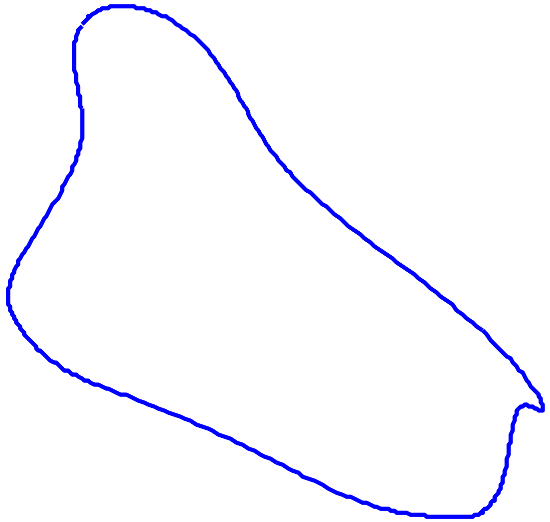
ETH zürich

Filtering Curves



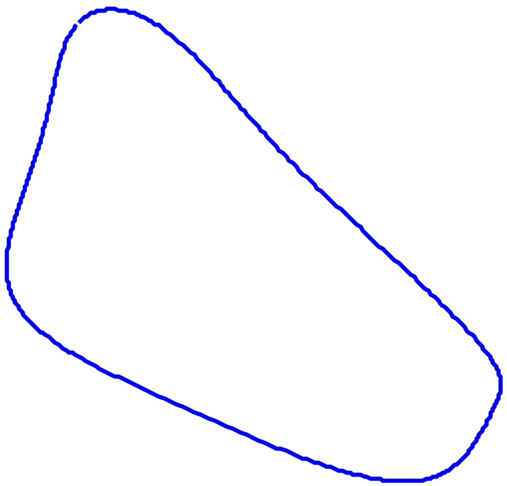
50th iteration; $\lambda=0.5$

Filtering Curves



500th iteration; $\lambda=0.5$

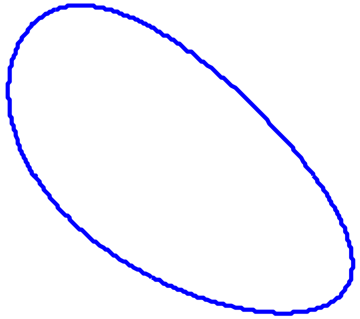
Filtering Curves



1000th iteration; $\lambda=0.5$

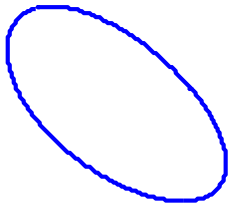
ETH zürich

Filtering Curves



5000th iteration; λ=0.5

Filtering Curves



10000th iteration; λ=0.5

Filtering Curves

50000th iteration; λ=0.5

On meshes: smoothing as mean curvature flow

Model smoothing as a diffusion process

$$\boxed{\frac{\partial \mathbf{p}}{\partial t} = \lambda \Delta \mathbf{p}} = -2\lambda H \mathbf{n}$$

On meshes: smoothing as mean curvature flow

Model smoothing as a diffusion process

$$\boxed{\frac{\partial \mathbf{p}}{\partial t} = \lambda \Delta \mathbf{p}} = -2\lambda H \mathbf{n}$$

Discretize in time, forward differences:

$$\frac{\mathbf{p}^{(n+1)} - \mathbf{p}^{(n)}}{dt} = \lambda L \mathbf{p}^{(n)}$$

On meshes: smoothing as mean curvature flow

Model smoothing as a diffusion process

$$\boxed{\frac{\partial \mathbf{p}}{\partial t} = \lambda \Delta \mathbf{p}} = -2\lambda H \mathbf{n}$$

Discretize in time, forward differences:

$$\frac{\mathbf{p}^{(n+1)} - \mathbf{p}^{(n)}}{dt} = \lambda L \mathbf{p}^{(n)}$$

$$\mathbf{p}^{(n+1)} - \mathbf{p}^{(n)} = dt\,\lambda L \mathbf{p}^{(n)}$$

On meshes: smoothing as mean curvature flow

Model smoothing as a diffusion process

$$\boxed{\frac{\partial \mathbf{p}}{\partial t} = \lambda \Delta \mathbf{p}} = -2\lambda H \mathbf{n}$$

Discretize in time, forward differences:

$$\frac{\mathbf{p}^{(n+1)} - \mathbf{p}^{(n)}}{dt} = \lambda L \mathbf{p}^{(n)}$$

$$\mathbf{p}^{(n+1)} - \mathbf{p}^{(n)} = dt\, \lambda L \mathbf{p}^{(n)}$$

$$\boxed{\mathbf{p}^{(n+1)} = (I + dt\, \lambda L)\mathbf{p}^{(n)}}$$

Explicit integration! Unstable unless time step dt is small

Smoothing as Mean Curvature Flow

Model smoothing as a diffusion process

$$\boxed{\frac{\partial \mathbf{p}}{\partial t} = \lambda \Delta \mathbf{p}} = -2\lambda H \mathbf{n}$$

Backward Euler for unconditional stability

$$\frac{\mathbf{p}^{(n+1)} - \mathbf{p}^{(n)}}{dt} = \lambda L \mathbf{p}^{\boxed{(n+1)}}$$

$$\mathbf{p}^{(n+1)} - \mathbf{p}^{(n)} = dt\, \lambda L \mathbf{p}^{(n+1)}$$

Smoothing as Mean Curvature Flow

Model smoothing as a diffusion process

$$\boxed{\frac{\partial \mathbf{p}}{\partial t} = \lambda \Delta \mathbf{p}} = -2\lambda H \mathbf{n}$$

Backward Euler for unconditional stability

$$\frac{\mathbf{p}^{(n+1)} - \mathbf{p}^{(n)}}{dt} = \lambda L \mathbf{p}^{\boxed{(n+1)}}$$

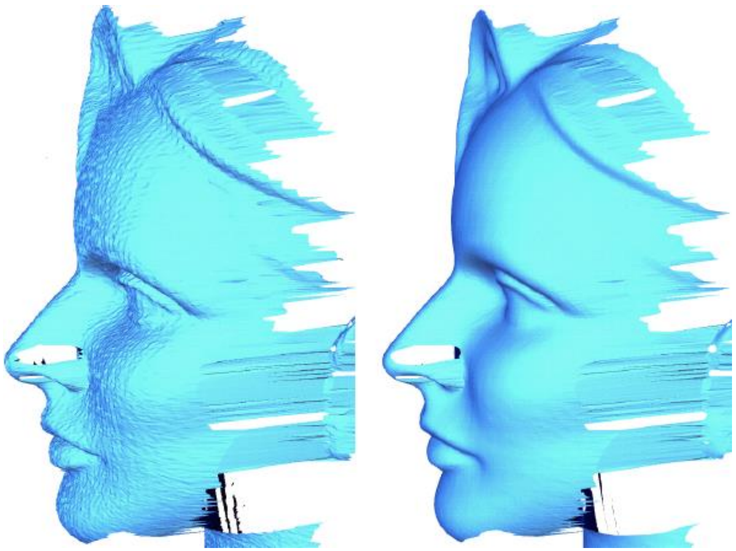
$$\mathbf{p}^{(n+1)} - \mathbf{p}^{(n)} = dt\, \lambda L \mathbf{p}^{(n+1)}$$

$$\boxed{(I - dt\, \lambda L)\mathbf{p}^{(n+1)} = \mathbf{p}^{(n)}}$$

Implicit Fairing: Implicit Euler Steps

In each iteration, solve for the smoothed $\tilde{\mathbf{p}}$:

$$(I - \tilde{\lambda}\, L)\tilde{\mathbf{p}} = \mathbf{p}$$



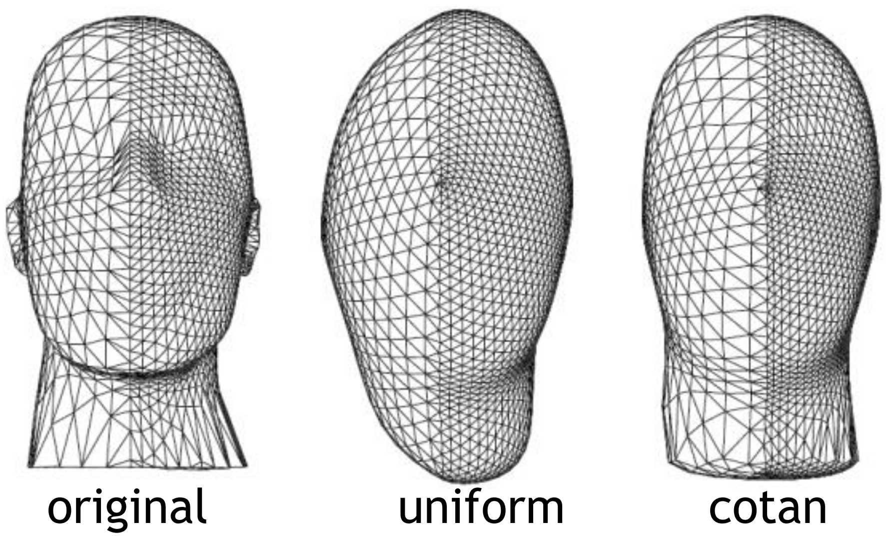
Implicit fairing of irregular meshes using diffusion and curvature flow
M. Desbrun, M. Meyer, P. Schroeder, A. Barr
ACM SIGGRAPH 99

Mesh Independence

Result of smoothing with uniform Laplacian depends on triangle density and shape

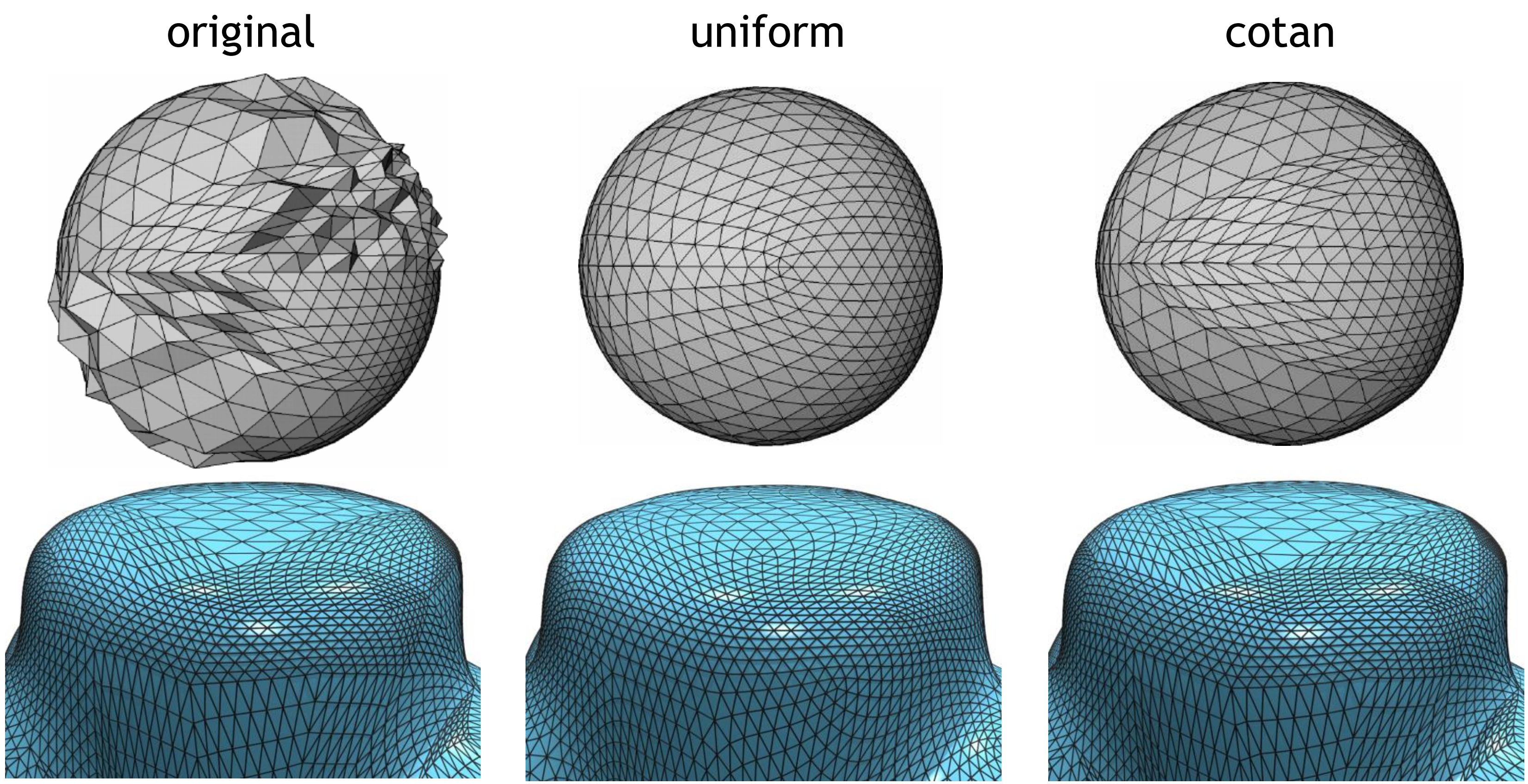
- Why?

Asymmetric results although underlying geometry is symmetric



Comparison of the weights

Explicit flow with different weights:



ETH zürich

Smoothing by optimization

Minimizing a smoothness energy

$$\Delta_{\mathcal{M}} \mathbf{p} = -2H\mathbf{n}$$

goal: $H = 0$ or $H = \mathrm{const}$

Let's go for $H = 0$

$\Delta_{\mathcal{M}} \tilde{\mathbf{p}} = 0$ only trivial solution, no connection to initial surface $\mathbf{p}$

Minimizing a smoothness energy

$$\Delta_{\mathcal{M}}\mathbf{p} = -2H\mathbf{n}$$

goal: $H = 0$ or $H = \text{const}$

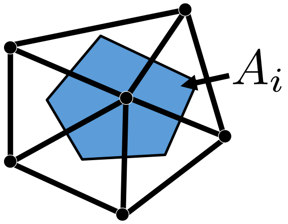
- Let's go for $H = 0$
- $\Delta_{\mathcal{M}}\tilde{\mathbf{p}} = 0$ only trivial solution, no connection to initial surface $\mathbf{p}$
- Let's regularize!

weighting factor (like $1/\lambda$)

$$\min_{\tilde{\mathbf{p}}} \int_{\mathcal{M}} \underbrace{\|\Delta_{\mathcal{M}}\tilde{\mathbf{p}}\|^2}_{\text{small } H} + w\underbrace{\|\tilde{\mathbf{p}} - \mathbf{p}\|^2}_{\text{stay close to original surface}}$$

Minimizing a smoothness energy

$$\min_{\tilde{\mathbf{p}}} \int_{\mathcal{M}} \|\Delta_{\mathcal{M}} \tilde{\mathbf{p}}\|^2 + w\|\tilde{\mathbf{p}} - \mathbf{p}\|^2$$



- Discretize:

$$\min_{\tilde{\mathbf{p}}} \sum_{i=1}^{n} A_i \left(\|L\tilde{\mathbf{p}}_i\|^2 + w\|\tilde{\mathbf{p}}_i - \mathbf{p}_i\|^2\right)$$

- Minimize!

$$\frac{\partial}{\partial \tilde{\mathbf{p}}} = 0$$

remember: $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T A \mathbf{x}) = (A + A^T)\mathbf{x}$

tip: “The Matrix Cookbook”

Minimizing a smoothness energy

$$\tilde{\mathbf{p}} = [\tilde{\mathbf{x}}\ \tilde{\mathbf{y}}\ \tilde{\mathbf{z}}] = \begin{pmatrix} \tilde{p}_{1x} & \tilde{p}_{1y} & \tilde{p}_{1z} \\ \tilde{p}_{2x} & \tilde{p}_{2y} & \tilde{p}_{2z} \\ \vdots & \vdots & \vdots \\ \tilde{p}_{nx} & \tilde{p}_{ny} & \tilde{p}_{nz} \end{pmatrix} \in \mathbb{R}^{n\times 3}$$

$$E(\tilde{\mathbf{p}}) = \sum_{\mathbf{v}\in\{\mathbf{x},\mathbf{y},\mathbf{z}\}} (L\tilde{\mathbf{v}})^T M (L\tilde{\mathbf{v}}) + w(\tilde{\mathbf{v}} - \mathbf{v})^T M(\tilde{\mathbf{v}} - \mathbf{v})$$

$$\frac{\partial E}{\partial \tilde{\mathbf{p}}} = 2L^T M L\tilde{\mathbf{p}} + 2wM(\tilde{\mathbf{p}} - \mathbf{p}) \overset{!}{=} 0$$

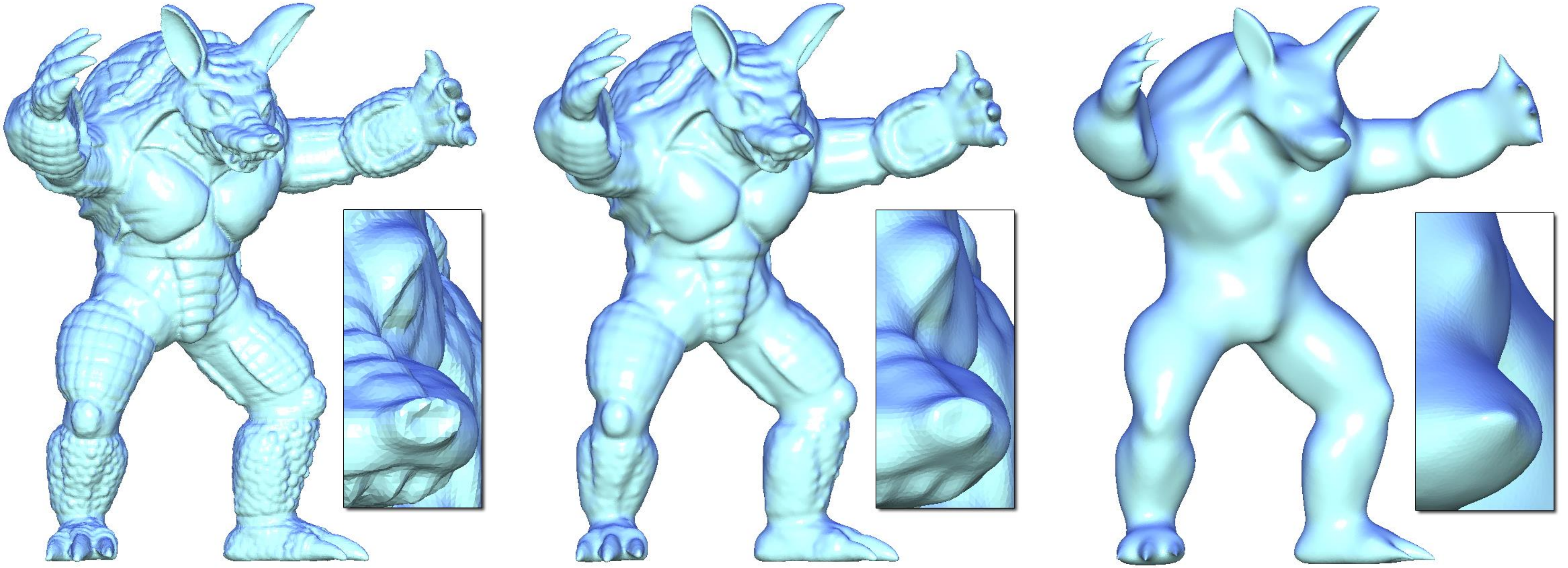
$$\Rightarrow \underline{\left(L^T M L + wM\right) \tilde{\mathbf{p}} = wM\mathbf{p}}$$

compare with implicit Euler!

$$NB:$$

$$L = M^{-1}L_w \;\Rightarrow\; L^T M L = L_w M^{-1} L_w$$

Results



original $\qquad w = 0.2 \qquad w = 0.02$

Customize the energy functional

$$\min_{\tilde{\mathbf{p}}} \sum_{i=1}^{n} A_i \left(\|L\tilde{\mathbf{p}}_i\|^2 + w\|\tilde{\mathbf{p}}_i - \mathbf{p}_i\|^2 \right)$$

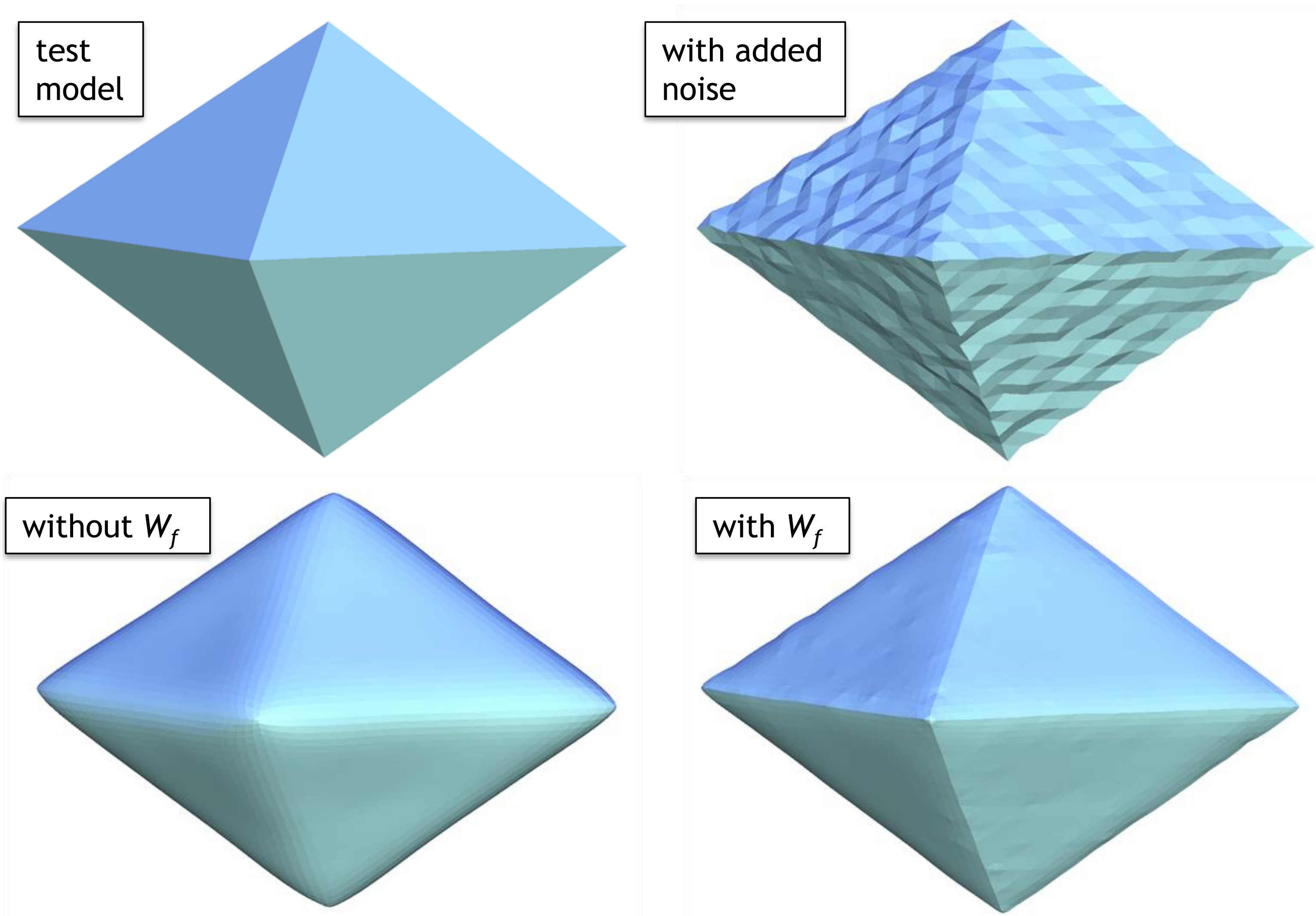
add a varying weight
here to control
feature preservation

$$E(\tilde{\mathbf{p}}) = \tilde{\mathbf{p}}^T L^T (M^{0.5} \underline{W_f} M^{0.5}) L\tilde{\mathbf{p}} + w(\tilde{\mathbf{p}} - \mathbf{p})^T M (\tilde{\mathbf{p}} - \mathbf{p})$$

$(W_f)_{ii}$ can be inverse-proportional to e.g. curvature at vertex *i*

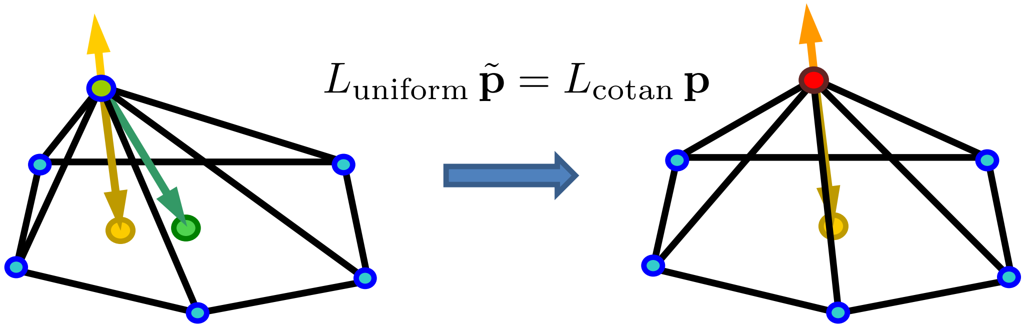
(slight abuse of notation, the energy is
actually the sum of the *xyz* components, $E(\tilde{\mathbf{p}}) = \sum_{\mathbf{v} \in \{\mathbf{x},\mathbf{y},\mathbf{z}\}} \tilde{\mathbf{v}}^T L^T \ldots$
like on slide #62)

Using W_f



Customize the energy functional

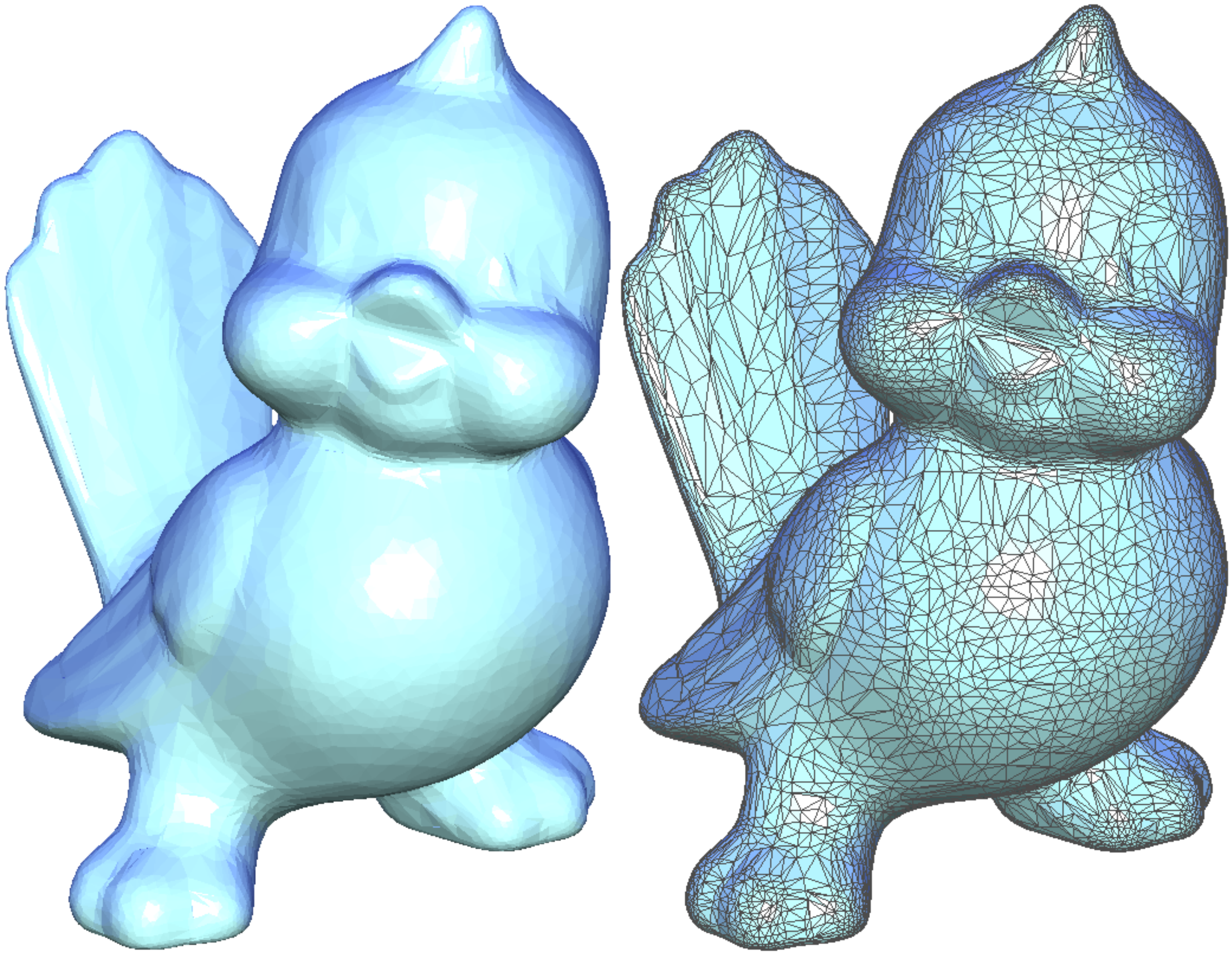
- Can do *tangential* smoothing!
  - Improve the shapes of mesh triangles without changing the surface shape



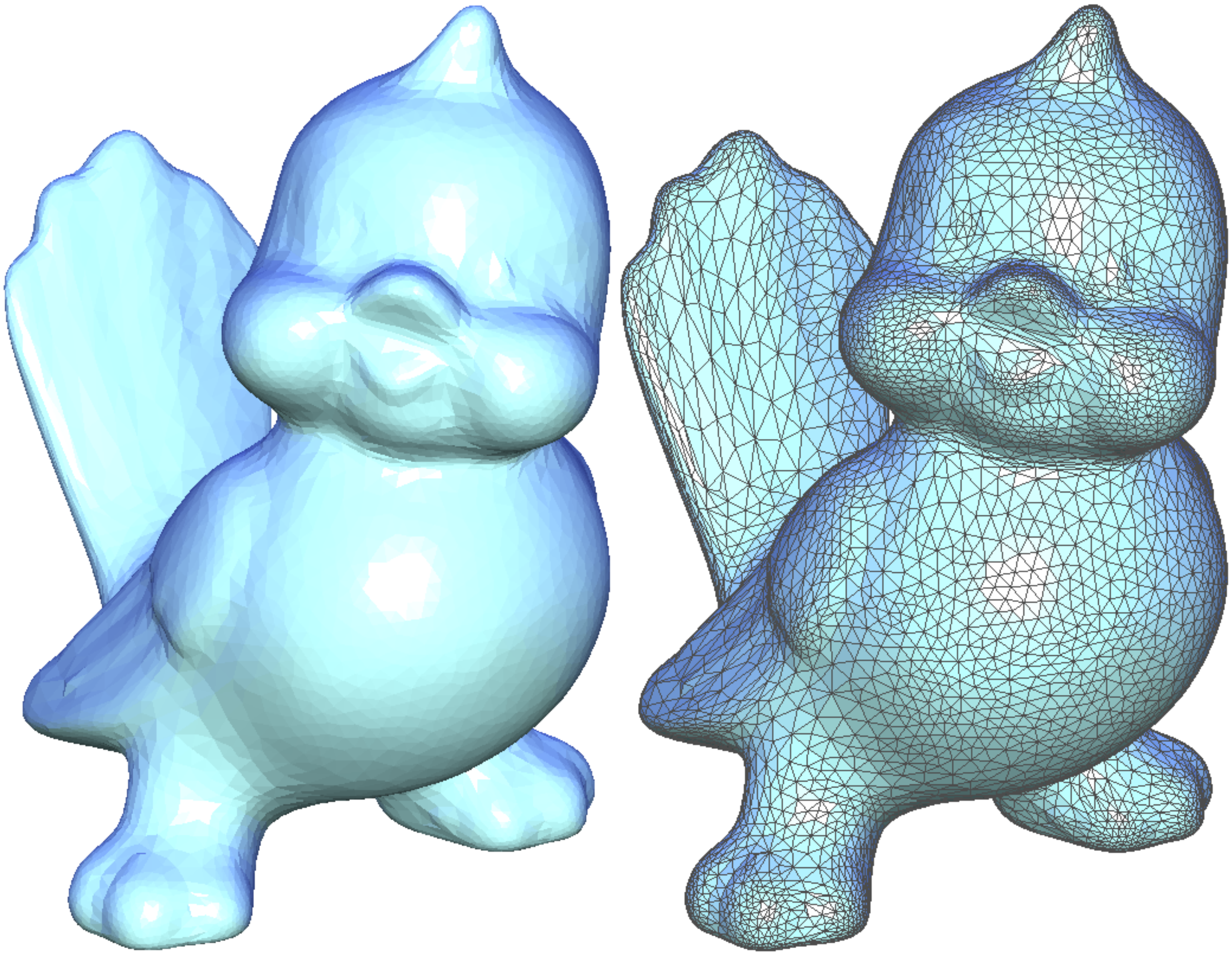
$$L_{\text{uniform}}\, \tilde{\mathbf{p}} = L_{\text{cotan}}\, \mathbf{p}$$

$$\min_{\tilde{\mathbf{p}}} \sum_{i=1}^{n} A_i \left(\| L_{\text{uniform}}\, \tilde{\mathbf{p}}_i - L_{\text{cotan}}\, \mathbf{p}_i \|^2 + w \| \tilde{\mathbf{p}}_i - \mathbf{p}_i \|^2 \right)$$

Original



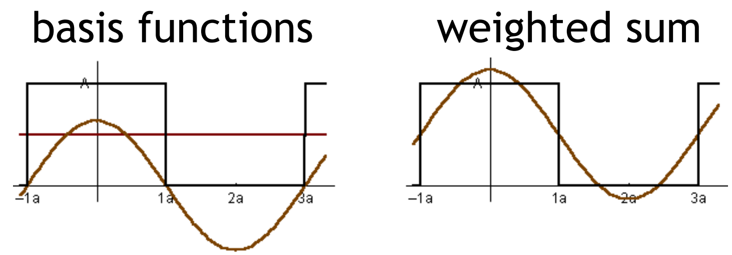
Triangle Shape Optimization



Smoothing by flitering

Fourier analysis

- Represent a function as a weighted sum of sines and cosines (basis functions)



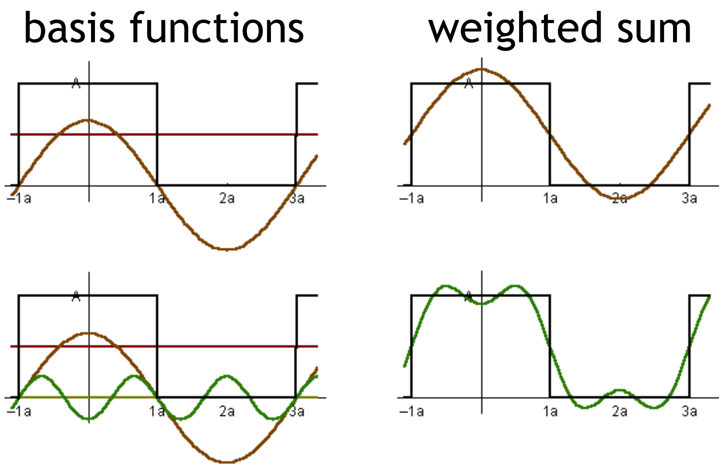
Joseph Fourier 1768 - 1830

$$f(x) = a_0 + a_1 \cos(x)$$

ETH zürich

Fourier analysis

- Represent a function as a weighted sum of sines and cosines (basis functions)



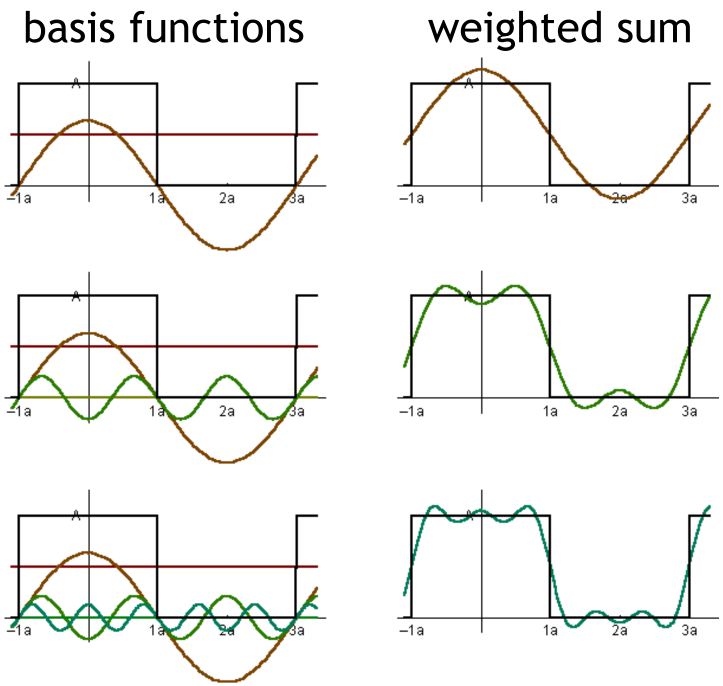
Joseph Fourier 1768 - 1830

$$f(x) = a_0 + a_1 \cos(x) + a_2 \cos(3x)$$

ETH zürich

Fourier analysis

- Represent a function as a weighted sum of sines and cosines (basis functions)



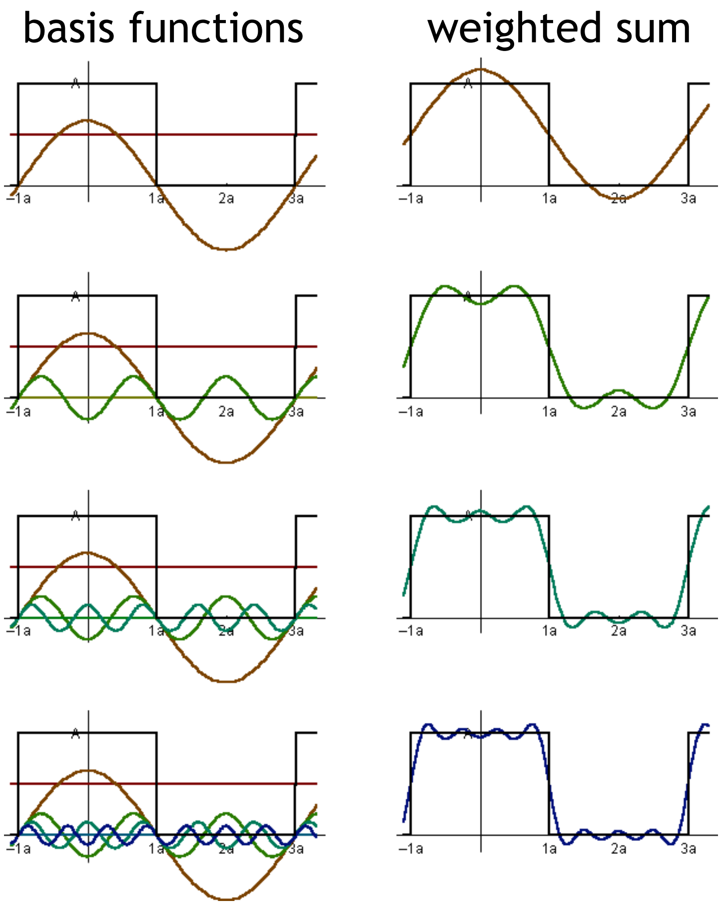
Joseph Fourier 1768 - 1830

$$f(x) = a_0 + a_1 \cos(x) + a_2 \cos(3x) + a_3 \cos(5x)$$

Fourier analysis

- Represent a function as a weighted sum of sines and cosines (basis functions)

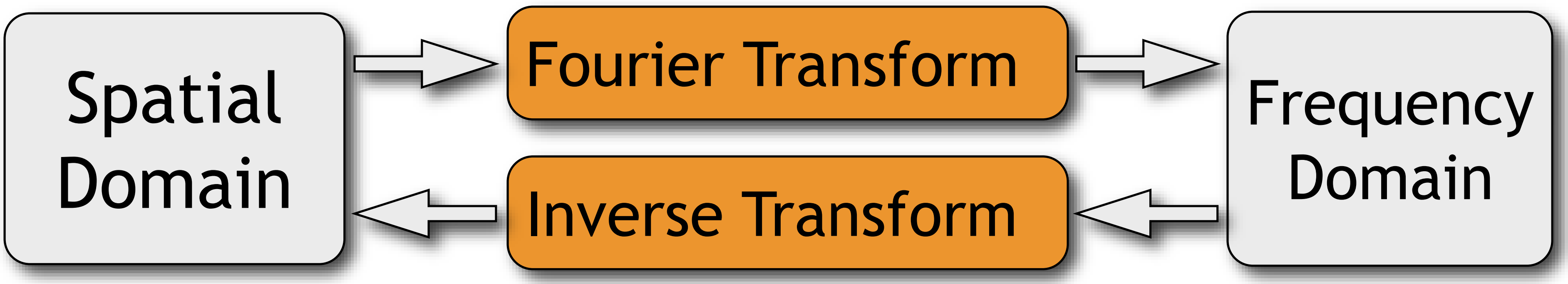
Joseph Fourier 1768 - 1830



$$f(x) = a_0 + a_1 \cos(x) + a_2 \cos(3x) + a_3 \cos(5x) + a_4 \cos(7x) + \ldots$$

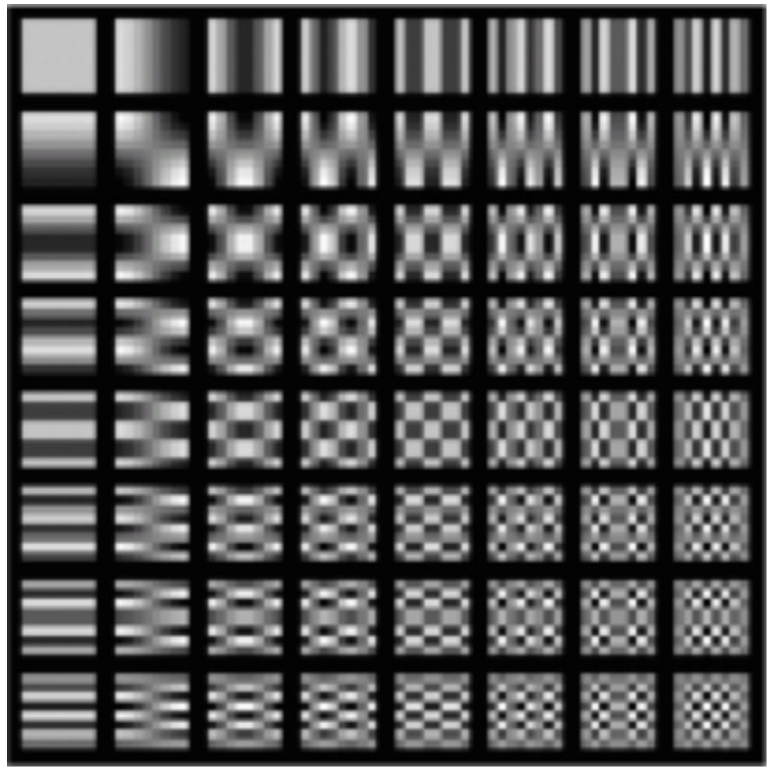
Fourier analysis

$$F(\omega) = \int_{-\infty}^{\infty} f(x) \mathrm{e}^{-2\pi i \omega x}\, dx$$



$$f(x) = \int_{-\infty}^{\infty} F(\omega) \mathrm{e}^{2\pi i \omega x}\, d\omega$$

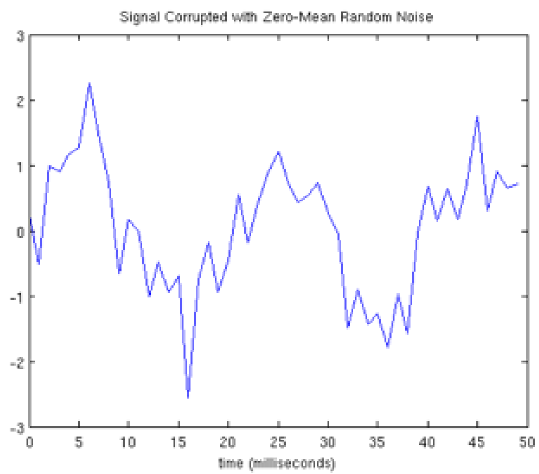
Also works on rectangular 2D domains



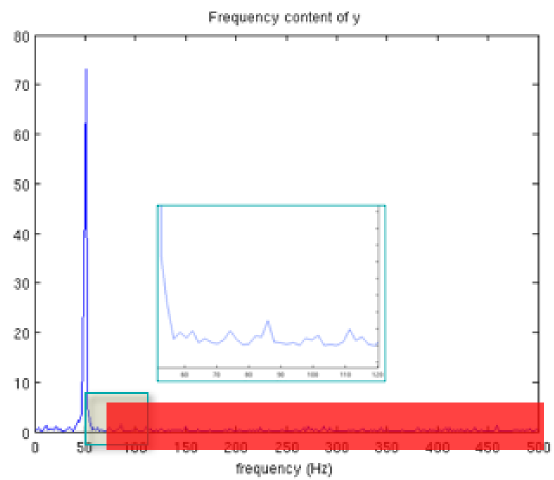
Fourier (DCT) basis functions for 8x8 grayscale images

$$\cos(2\pi\omega_h)\cos(2\pi\omega_v)$$

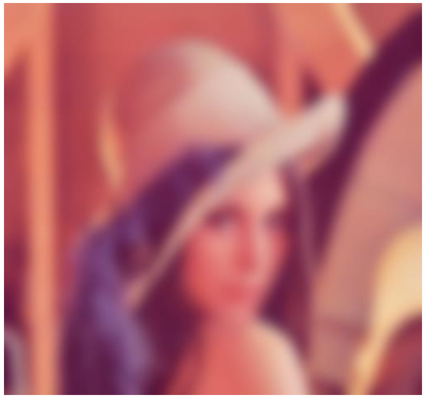
Smoothing = filtering high frequencies out



spatial domain



frequency domain



Extend Fourier to meshes?

- Basis functions??
- Fourier basis functions are eigenfunctions of the (standard) Laplace operator:

$$\Delta\left(e^{2\pi i\omega x}\right) = \frac{\partial^2}{\partial x^2} e^{2\pi i\omega x} = -(2\pi\omega)^2 e^{2\pi i\omega x}$$

- On meshes: take the eigenvectors of the Laplace-Beltrami matrix!

Spectral analysis on meshes

- Take your favorite L-B matrix L
- Compute eigenvectors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_k$ with the k smallest eigenvalues
- Reconstruct the smoothed mesh geometry from these eigenvectors:

$$\mathbf{x} = [x_1, \ldots, x_n]^T \qquad \mathbf{y} = [y_1, \ldots, y_n]^T \qquad \mathbf{z} = [z_1, \ldots, z_n]^T$$

$$\tilde{\mathbf{x}} = \sum_{i=1}^{k} \left(\mathbf{x}^T \mathbf{e}_i\right) \mathbf{e}_i \qquad \tilde{\mathbf{y}} = \sum_{i=1}^{k} \left(\mathbf{y}^T \mathbf{e}_i\right) \mathbf{e}_i \qquad \tilde{\mathbf{z}} = \sum_{i=1}^{k} \left(\mathbf{z}^T \mathbf{e}_i\right) \mathbf{e}_i$$

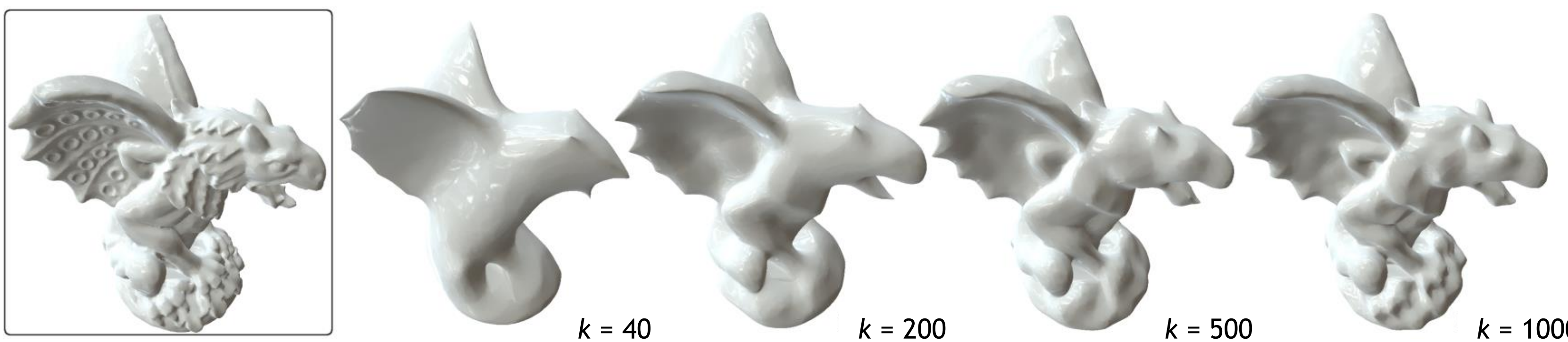
$$\tilde{\mathbf{p}} = [\tilde{\mathbf{x}}\ \tilde{\mathbf{y}}\ \tilde{\mathbf{z}}] \in \mathbb{R}^{n \times 3}$$

Spectral analysis on meshes

- Take your favorite L-B matrix L
- Compute eigenvectors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_k$ with the k smallest eigenvalues

too expensive
for large meshes

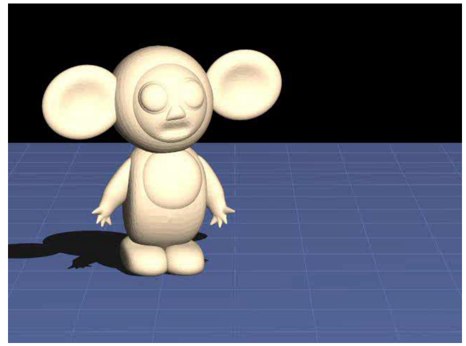
- Reconstruct the smoothed mesh geometry from these eigenvectors:



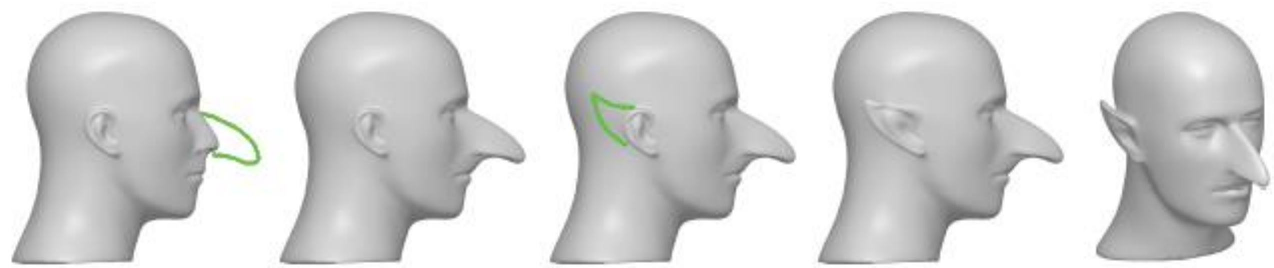
Deformation

Why Shape Deformation?

Animation



Editing

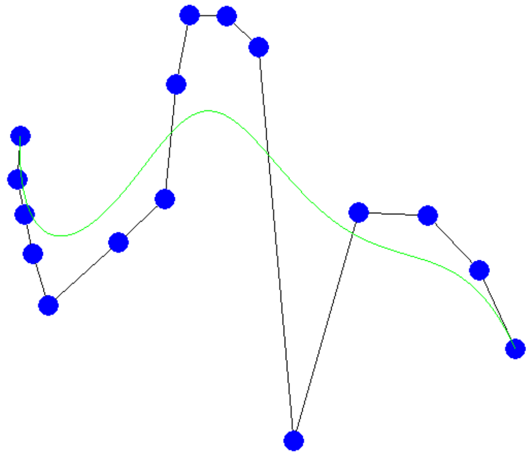


Simulation

Parametric Curves and Surfaces

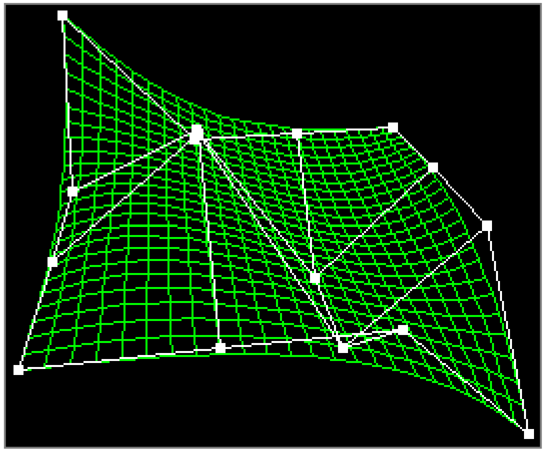
Deformation by control point manipulation

Show example

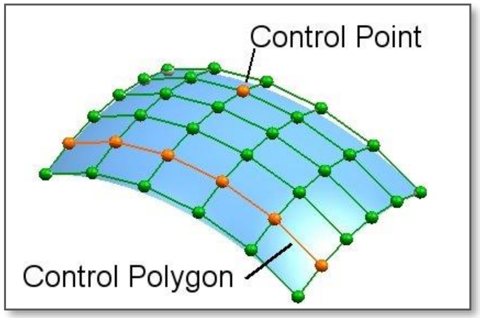


Built-in deformation mechanism

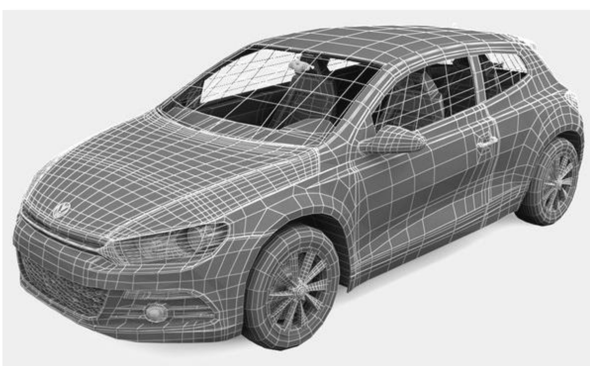
Control structure is pre-set (can't pull on arbitrary points)



Traditional CAD vs Unstructured Meshes



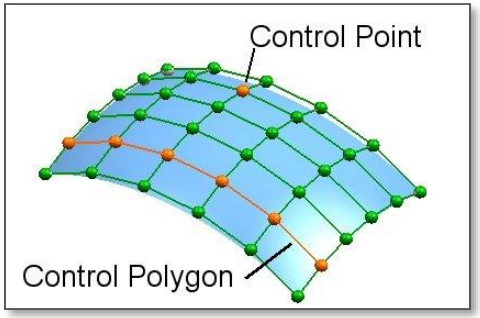
$$s(u,v) = \sum_{i,j} \mathbf{p}_{i,j} B_i(u) B_j(v)$$



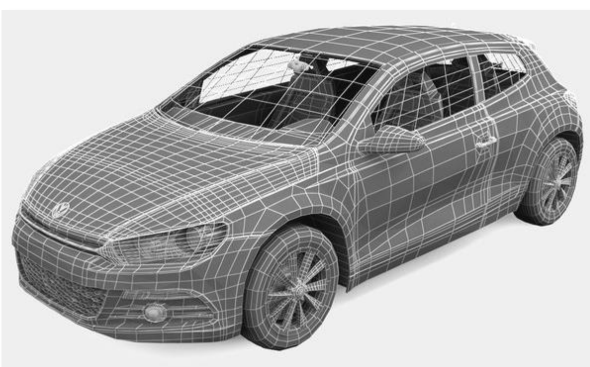
$$\min_{\mathbf{x}'} \int_{\mathcal{S}} \|\Delta_{\mathcal{S}} \mathbf{x}' - \delta_0\|^2$$
$$s.t.\ \mathbf{x}'|_{\mathcal{C}} = \mathbf{x}_{\text{fixed}}$$

images from Jacobson et al., SGP 2010

Traditional CAD vs Unstructured Meshes



$$s(u,v) = \sum_{i,j} \mathbf{p}_{i,j} B_i(u) B_j(v)$$



$$\min_{\mathbf{x}'} \int_{\mathcal{S}} \|\Delta_{\mathcal{S}} \mathbf{x}' - \delta_0\|^2$$
$$s.t.\ \mathbf{x}'|_{\mathcal{C}} = \mathbf{x}_{\text{fixed}}$$

Deformation metaphors

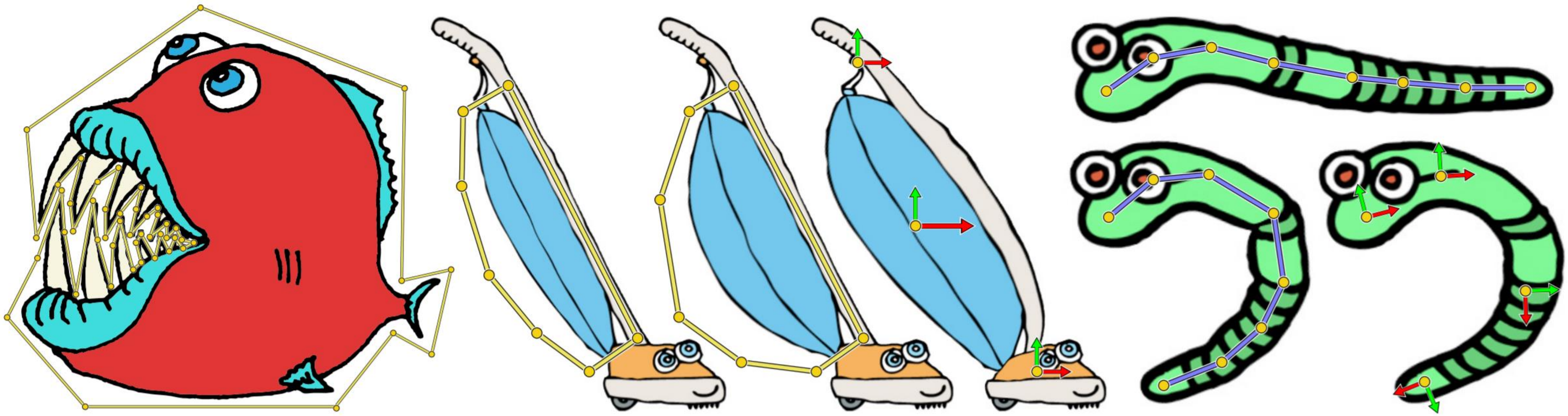
Bounded Biharmonic Weights for Real-Time Deformation

Alec Jacobson[1]
Ilya Baran[2]
Jovan Popović[3]
Olga Sorkine[1,4]

[1]New York University
[2]Disney Research, Zurich
[3]Adobe Systems, Inc.
[4]ETH Zurich

This video contains narration

Deformation metaphors

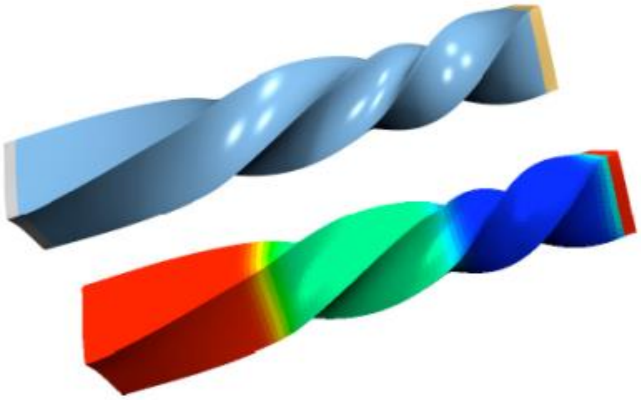


Deformation: Common Paradigms

Surface based deformation

Optimization on the surface

Physically motivated: variants of elastic energy minimization



Space deformation

Deforms some 2D/3D space using a *cage*

Deformation propagation to all points in the space

Independent of shape representation

ETH zürich

General Framework

Find a mesh that optimizes an objective and satisfies modeling constraints

$$\mathbf{x}_{\mathrm{def}} = \arg\min E(\underbrace{\mathbf{x}}_{\text{Original mesh}}, \underbrace{\mathbf{x}'}_{\text{Candidate mesh}}) \quad s.t. \mathbf{x}_i' = \mathbf{c}_i$$

ETH zürich

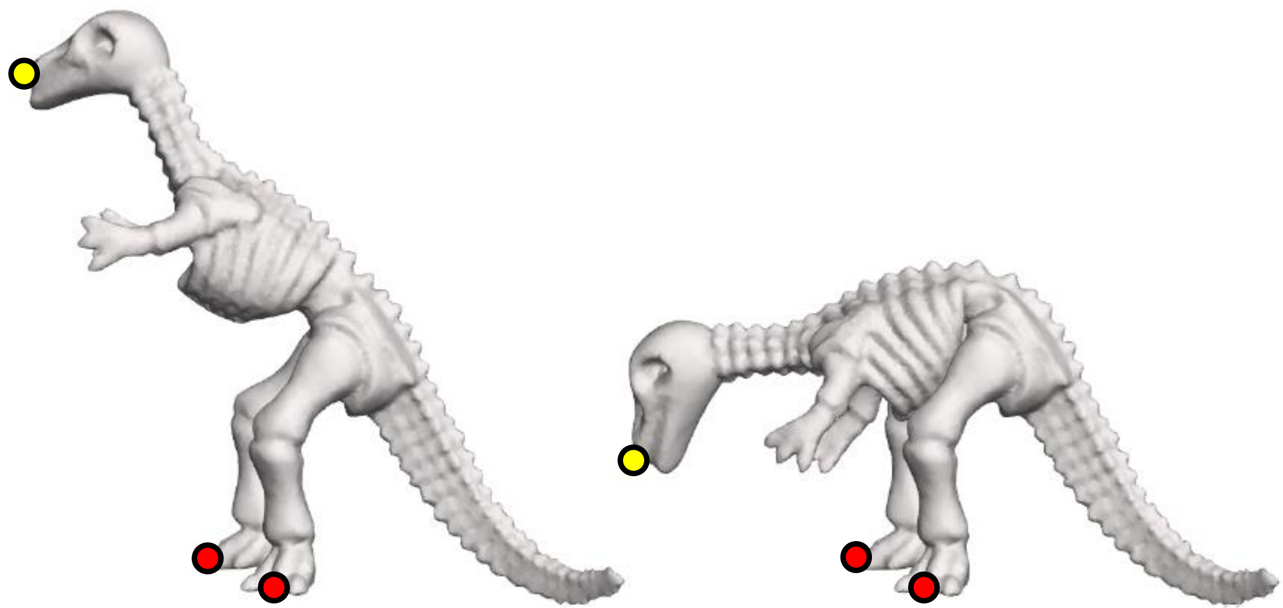
How to Define $E(\mathbf{x}, \mathbf{x}')$?

Goal: Intuitive deformations

- Smooth deformation on the global scale
- Preserve local details (curvatures)

Invariants:

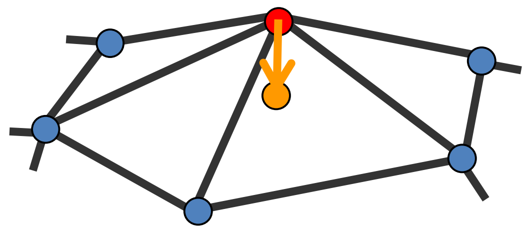
- $E(\mathbf{x}') = 0$ if $\mathbf{x}'$ is a rigid transformation of $\mathbf{x}$



Differential Coordinates

Detail = *smooth*(surface) – surface

Smoothing = averaging



$$\delta_i = \frac{1}{W_i} \sum_{j \in \mathcal{N}(i)} w_{ij} (\mathbf{x}_j - \mathbf{x}_i) \approx -2H_i \mathbf{n}_i$$

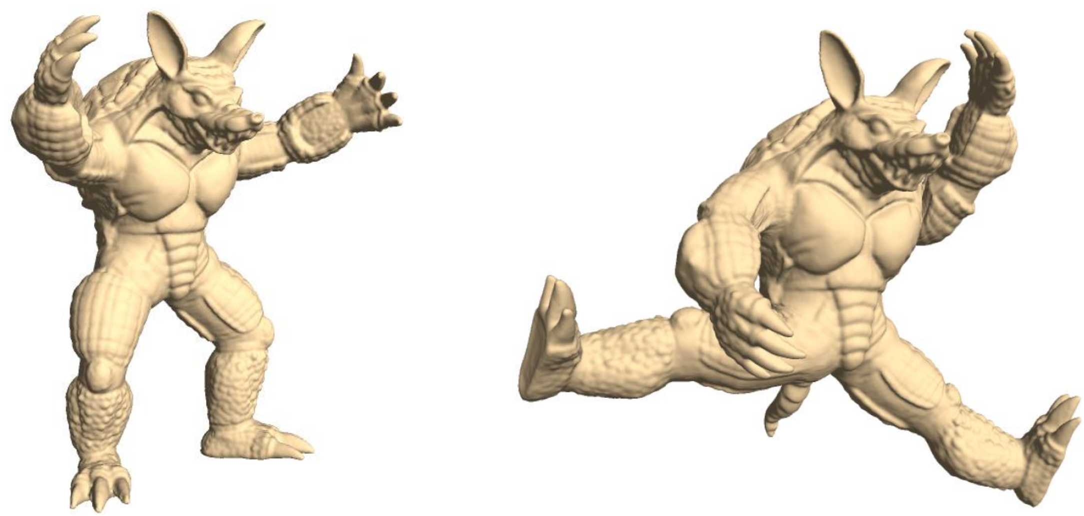
Differential Coordinates

Represent ***local detail*** at each surface point

- More descriptive of the shape than just xyz

Linear transition from xyz to δ

Useful for operations on surfaces where surface details are important



Simple Laplacian Editing

Preserve mean curvature normal [≈differential coordinates] at every point in the ROI [≈ every vertex of the ROI]

continuous: $$E(\mathcal{S}') = \int_{\mathcal{S}'} \|\Delta \mathbf{x}' - \delta\|^2 d\mathbf{x}'$$

discrete: $$E(\mathbf{x}') = \sum_{i=1}^{n} A_i \|\Delta(\mathbf{x}_i') - \delta_i\|^2$$

ETH zürich

Simplifying the Laplacian Energy

$$E(\mathbf{x}') = \sum_{i=1}^{n} A_i \|\Delta(\mathbf{x}'_i) - \delta_i\|^2 = \sum_{i=1}^{n} A_i \left(\Delta(\mathbf{x}'_i)^T \Delta(\mathbf{x}'_i) - 2\Delta(\mathbf{x}'_i)^T \delta_i + \delta_i^T \delta_i\right) =$$

$$= \mathbf{x}'^T \underline{L^T M L} \mathbf{x}' - 2\mathbf{x}'^T \underline{L^T M} \delta + \text{const}$$

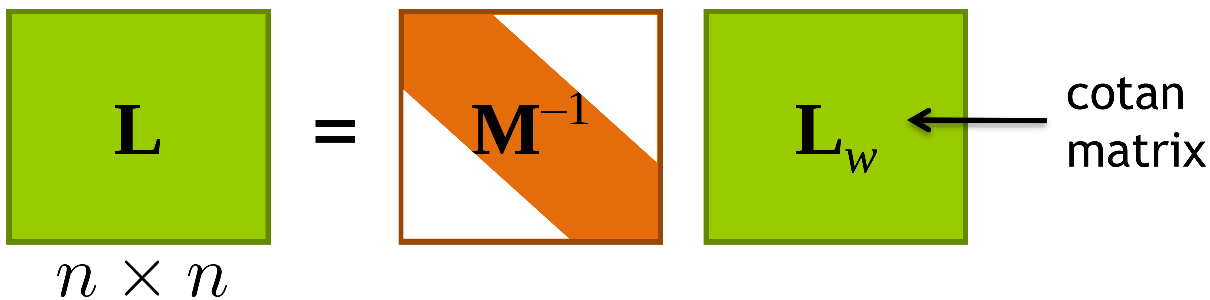
$$\underset{n \times n}{\mathbf{L}} = \mathbf{M}^{-1} \mathbf{L}_w$$

cotan matrix

Simplifying the Laplacian Energy

$$E(\mathbf{x}') = \sum_{i=1}^{n} A_i \|\Delta(\mathbf{x}'_i) - \delta_i\|^2 = \sum_{i=1}^{n} A_i \left(\Delta(\mathbf{x}'_i)^T \Delta(\mathbf{x}'_i) - 2\Delta(\mathbf{x}'_i)^T \delta_i + \delta_i^T \delta_i\right) =$$

$$= \ \mathbf{x}'^T \underline{L^T M L} \mathbf{x}' - 2\mathbf{x}'^T \underline{L^T M} \delta + \text{const}$$



$$L^T M L = (M^{-1} L_w)^T M (M^{-1} L_w) = L_w M^{-1} M M^{-1} L_w =$$

$$= L_w M^{-1} L_w \longleftarrow \text{Symmetric sparse matrix!}$$

Minimizing the Laplacian Energy

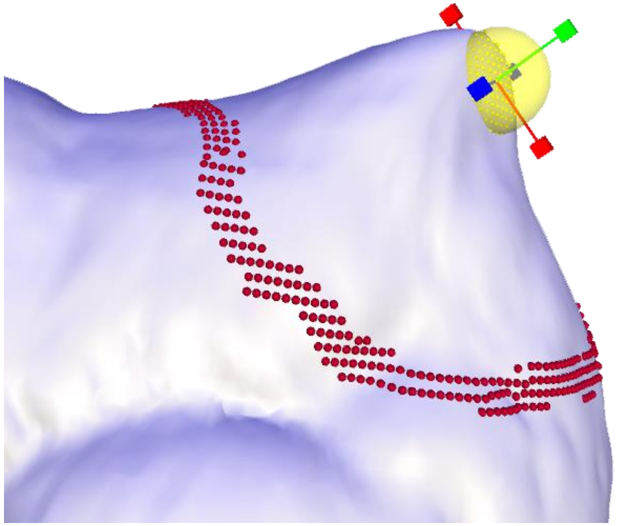
$$E(\mathbf{x}') = \ \mathbf{x}'^T L_w M^{-1} L_w \mathbf{x}' - 2\mathbf{x}'^T L_w \delta + \mathrm{const}$$

$$\frac{\partial}{\partial \mathbf{x}'} E(\mathbf{x}') = 2 L_w M^{-1} L_w \mathbf{x}' - 2 L_w \delta$$

- To find the minimum, gradient = 0 and substitute the modeling constraints $\mathbf{x}'_i = \mathbf{c}_i, \ i \in \mathcal{C}$

Minimizing the Laplacian Energy

$$A\mathbf{x}' = \mathbf{b}$$

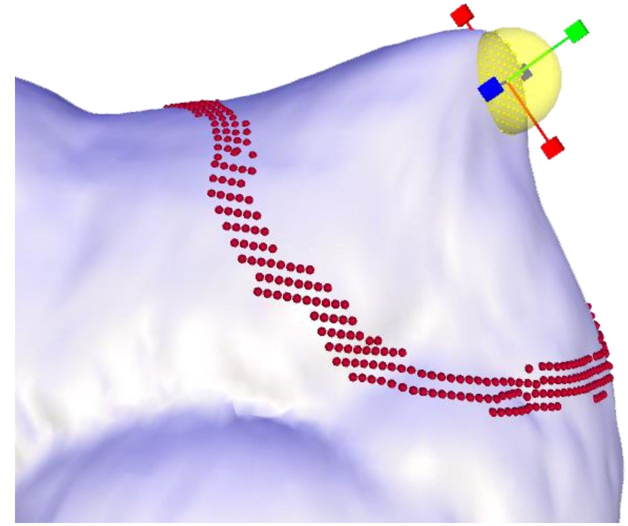


Matrix depends on the initial mesh and the indices of the constraints only. Matrix is fixed!

Right-hand side contains the coordinates of the constraints (handles)

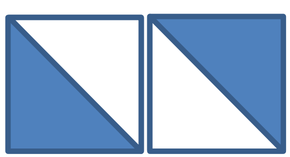
Minimizing the Laplacian Energy

$$A\mathbf{x}' = \mathbf{b}$$



Sparse Cholesky decomposition:

$$A = L_{\mathrm{chol}} L_{\mathrm{chol}}^T$$



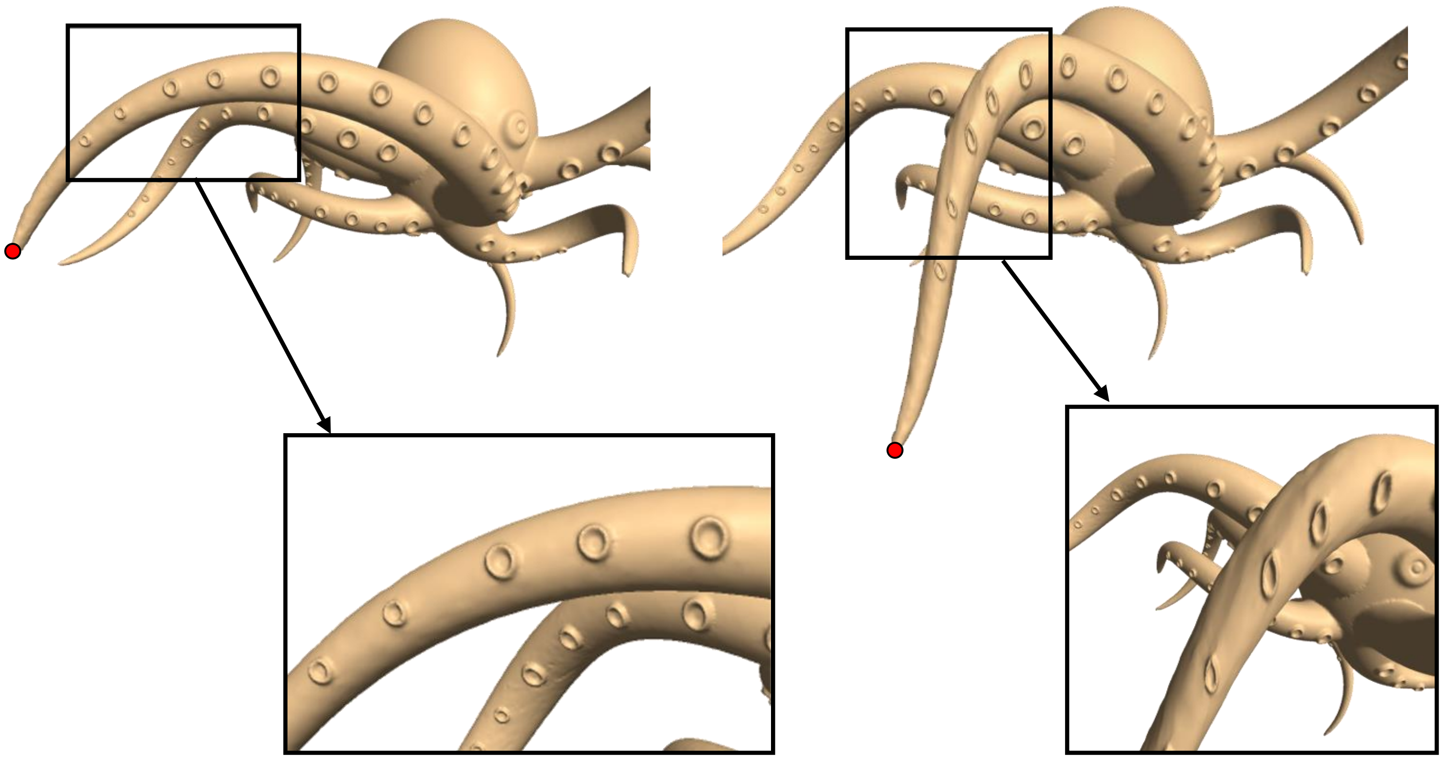
At run-time: just back-substitution!

$$L_{\mathrm{chol}}\mathbf{y} = \mathbf{b}$$

$$L_{\mathrm{chol}}^T \mathbf{x}' = \mathbf{y}$$

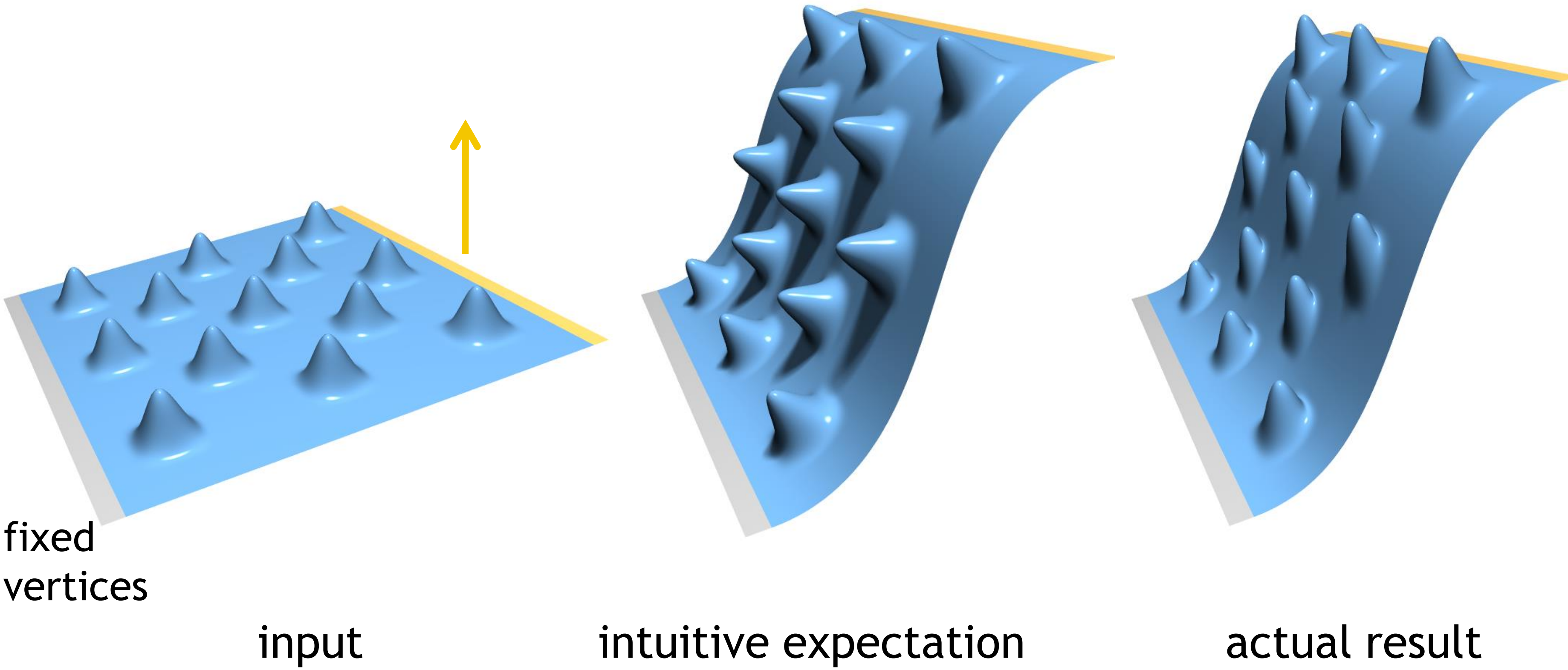
Fundamental Problem: Invariance to Transformations

- The basic Laplacian operator is ***translation***-invariant, but not **rotation-**invariant
- E(**x'**) attempts to preserve the **original global** orientation of the details (the normal directions)



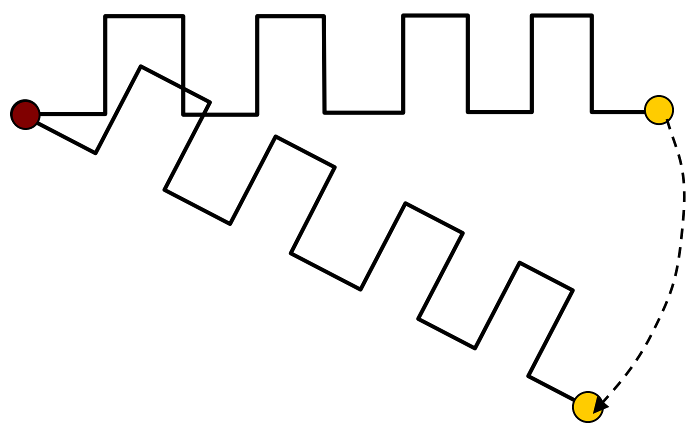
ETH zürich

Fundamental Problem: Invariance to Transformations



Fundamental Problem: Invariance to Transformations

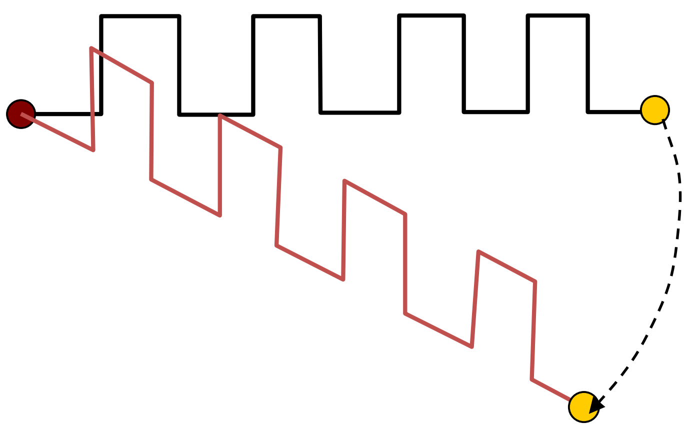
- The basic Laplacian operator is ***translation***-invariant, but not **rotation-**invariant
- E(**x'**) attempts to preserve the **original global** orientation of the details (the normal directions)



ETH zürich

Fundamental Problem: Invariance to Transformations

- The basic Laplacian operator is ***translation***-invariant, but not **rotation-**invariant
- E(**x'**) attempts to preserve the **original global** orientation of the details (the normal directions)



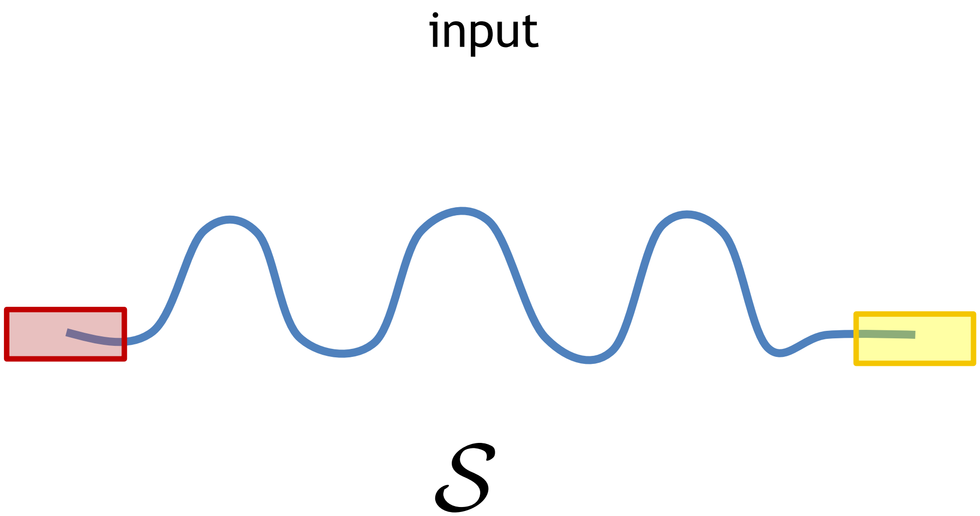
Energy Functional

- We need a rigid-invariant energy…

$$E(\mathbf{x}') = \sum_{i=1}^{n} A_i \|\Delta(\mathbf{x}'_i) - \delta_i\|^2$$

Need to locally rotate the *target* m.c. normals

Fixing Local Rotations: Multiresolution Approach



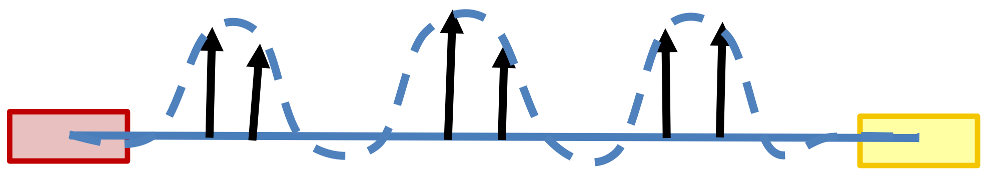
Fixing Local Rotations: Multiresolution Approach

Smooth base surface

$\mathcal{B}$

Fixing Local Rotations: Multiresolution Approach

Details – displacement vectors

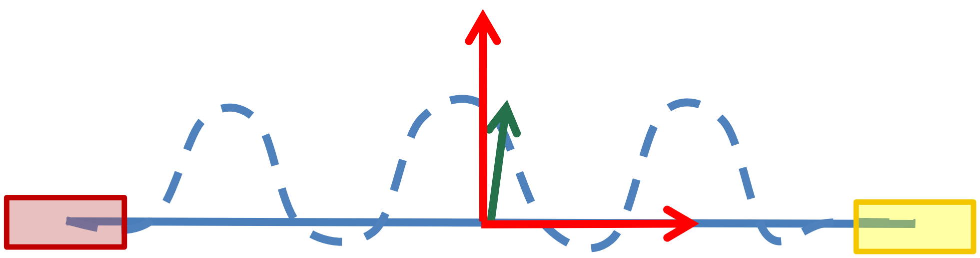


$$\mathcal{S} - \mathcal{B}$$

ETH zürich

Fixing Local Rotations: Multiresolution Approach

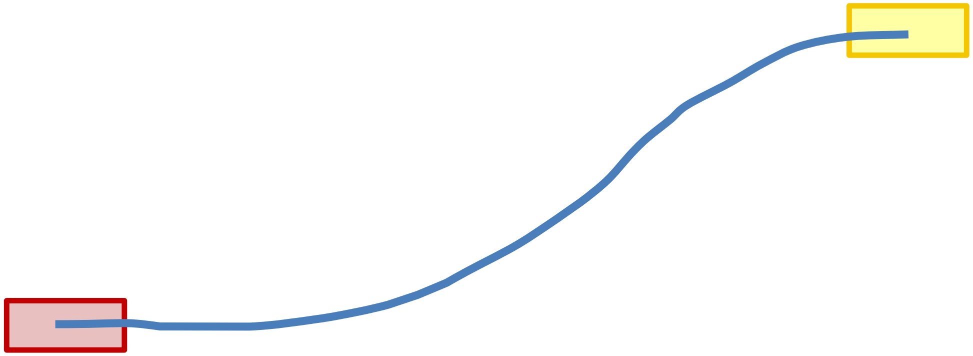
Encode details in the local frame of B



$$\mathbf{d}_i = a_1\mathbf{t}_i + a_2\mathbf{n}_i$$

Fixing Local Rotations: Multiresolution Approach

Deform smooth base surface

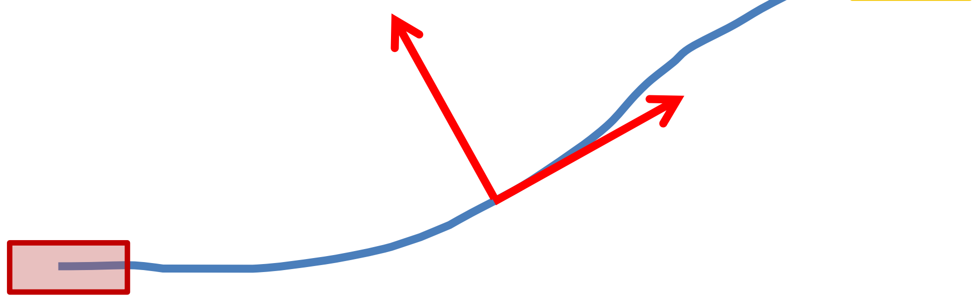


$\mathcal{B}'$

ETH zürich

Fixing Local Rotations: Multiresolution Approach

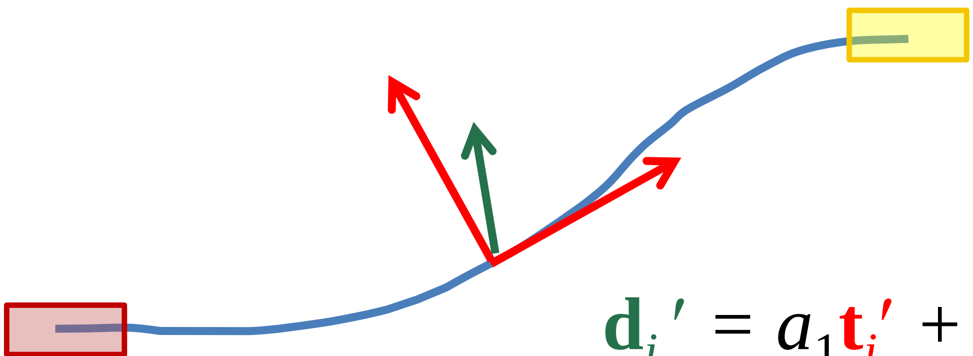
Local frames on B'



$$\mathcal{B}'$$

Fixing Local Rotations: Multiresolution Approach

Add details back – in local frame!



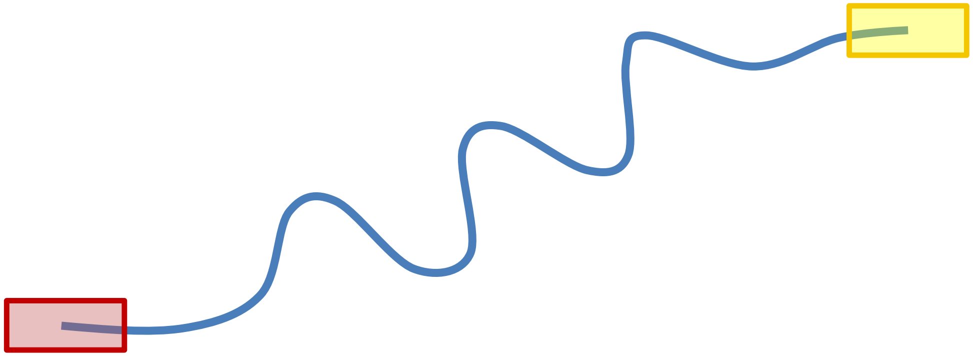
$$\mathbf{d}_i{}' = a_1\mathbf{t}_i{}' + a_2\mathbf{n}_i{}'$$

$$\mathcal{B}'$$

ETH zürich

Fixing Local Rotations: Multiresolution Approach

Displace the vertices to get the result

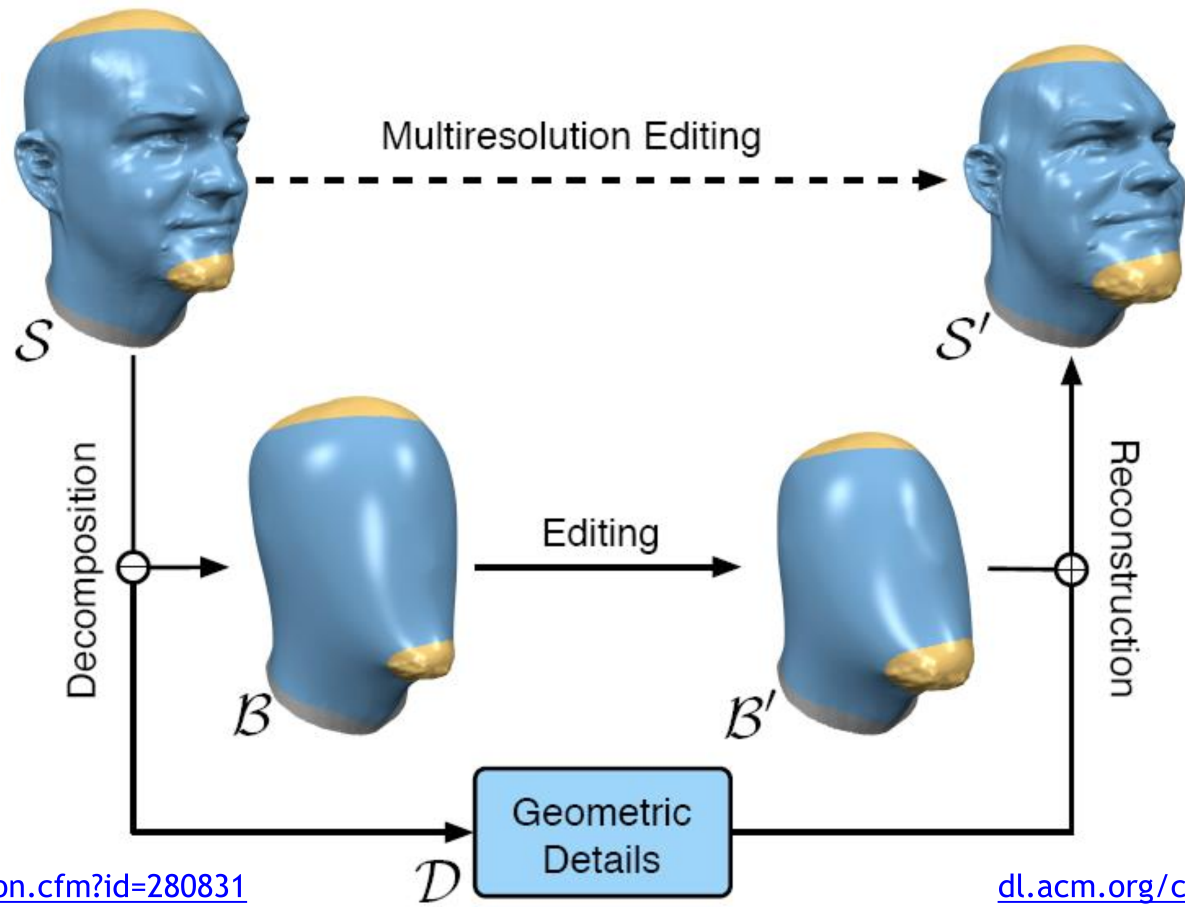


$$\mathcal{S}'$$

ETH zürich

Fixing Local Rotations: Multiresolution Approach

- Kobbelt et al. SIGGRAPH 98, Botsch and Kobbelt SIGGRAPH 2004

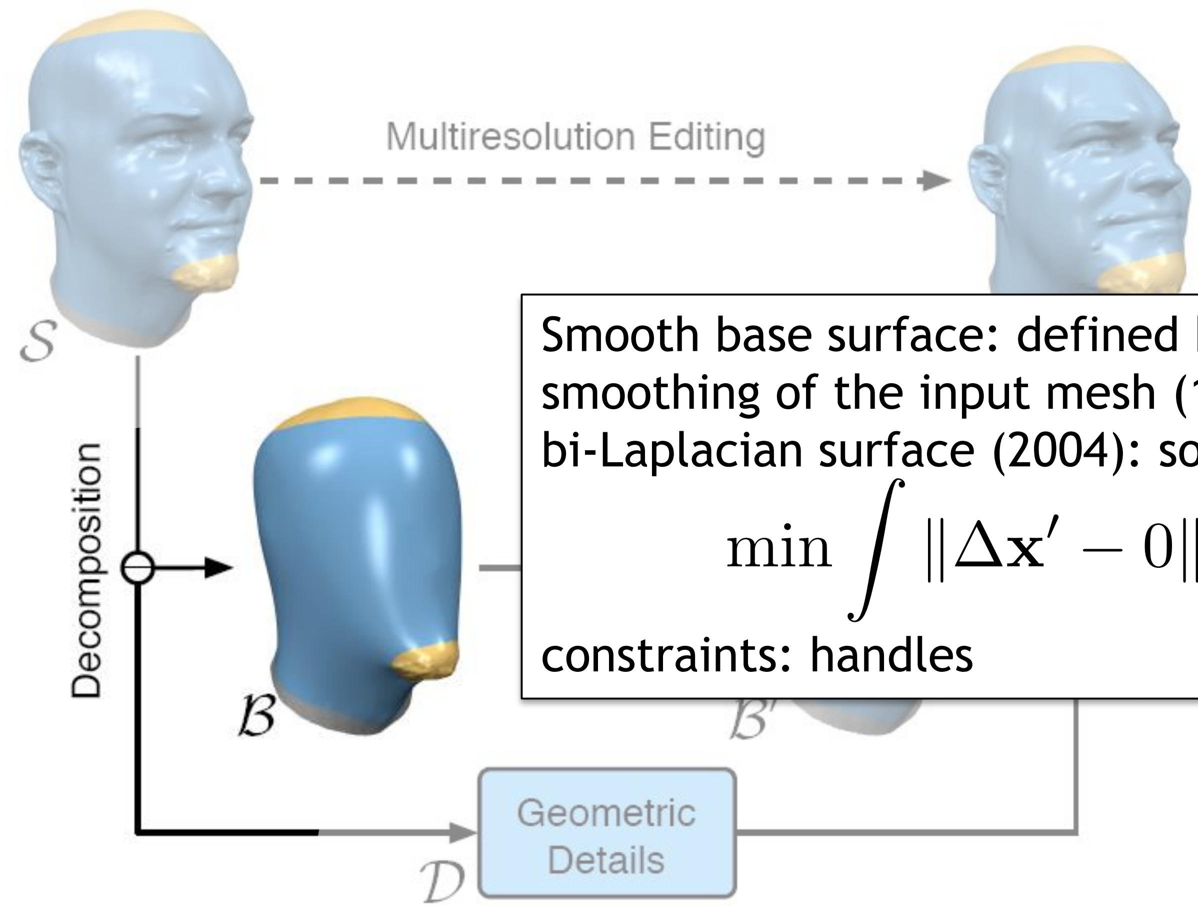


http://dl.acm.org/citation.cfm?id=280831

dl.acm.org/citation.cfm?id=1015772

Fixing Local Rotations: Multiresolution Approach

- Kobbelt et al. SIGGRAPH 98, Botsch and Kobbelt SIGGRAPH 2004



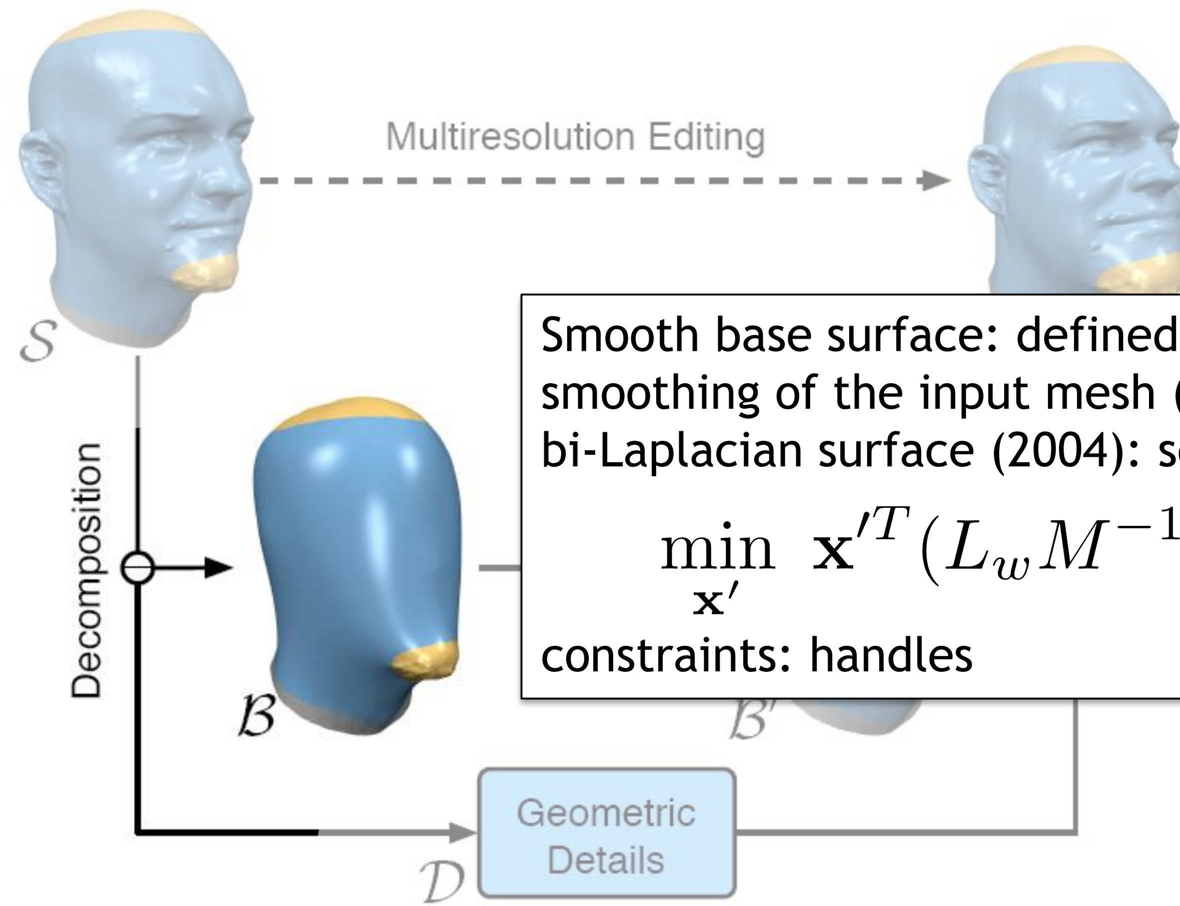
Smooth base surface: defined by Laplacian smoothing of the input mesh (1998) or the bi-Laplacian surface (2004): solve

$$\min \int \|\Delta \mathbf{x}' - 0\|^2 d\mathbf{x}'$$

constraints: handles

Fixing Local Rotations: Multiresolution Approach

- Kobbelt et al. SIGGRAPH 98, Botsch and Kobbelt SIGGRAPH 2004



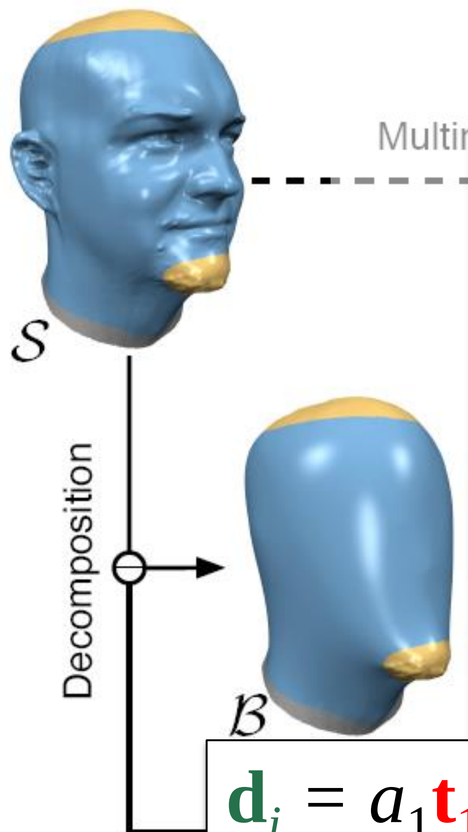
Smooth base surface: defined by Laplacian smoothing of the input mesh (1998) or the bi-Laplacian surface (2004): solve

$$\min_{\mathbf{x}'} \ \mathbf{x}'^T (L_w M^{-1} L_w)\, \mathbf{x}'$$

constraints: handles

Fixing Local Rotations: Multiresolution Approach

- Kobbelt et al. SIGGRAPH 98, Botsch and Kobbelt SIGGRAPH 2004

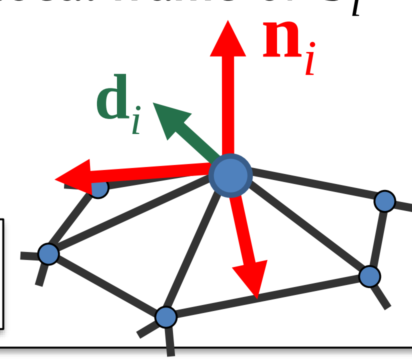


Geometric details: simply the difference vectors between base surface B and input S

For each vertex, compute

$$\mathbf{d}_i = \mathbf{v}_i - \mathbf{b}_i$$

Represent $\mathbf{d}_i$ in the **local frame** of $\mathbf{b}_i$



$$\mathbf{d}_i = a_1\mathbf{t}_1 + a_2\mathbf{t}_2 + a_3\mathbf{n}_i$$

ETH zürich

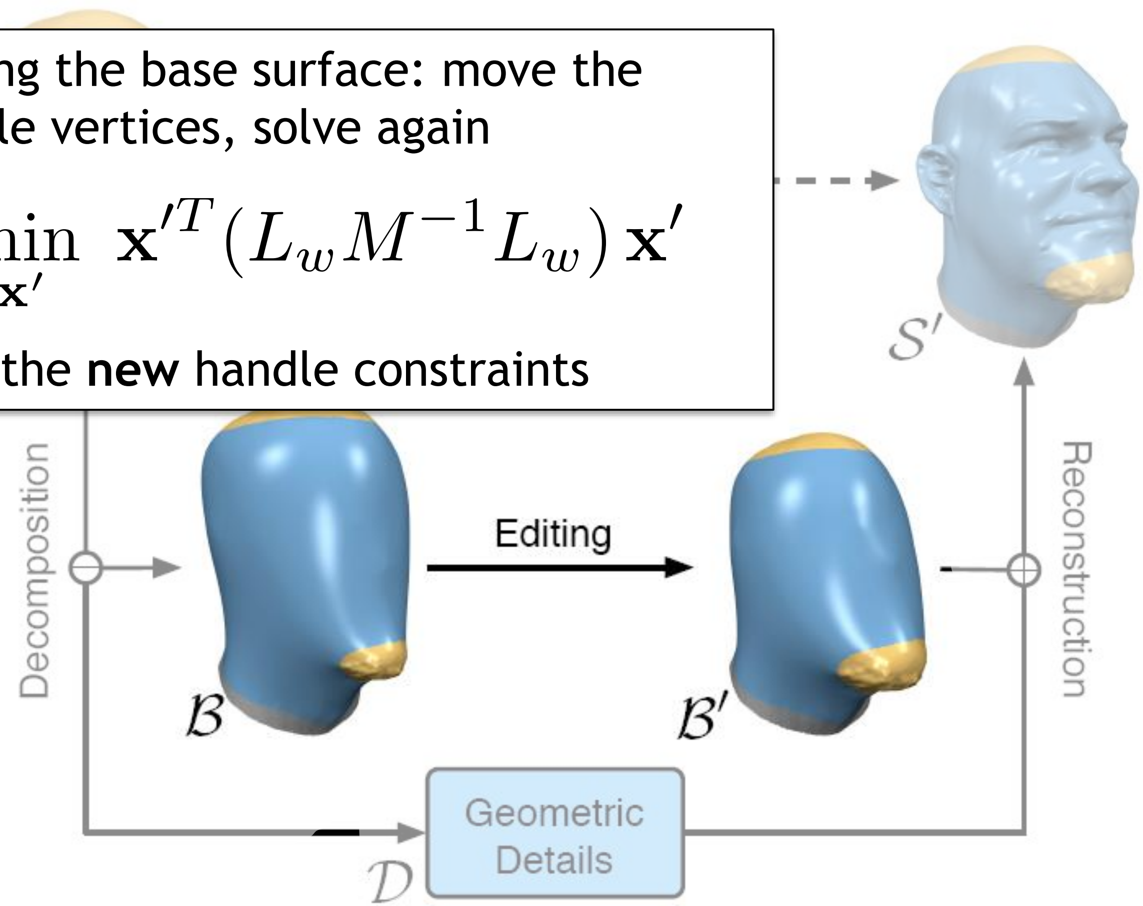
Fixing Local Rotations: Multiresolution Approach

- Kobbelt et al. SIGGRAPH 98, Botsch and Kobbelt SIGGRAPH 2004

Editing the base surface: move the handle vertices, solve again

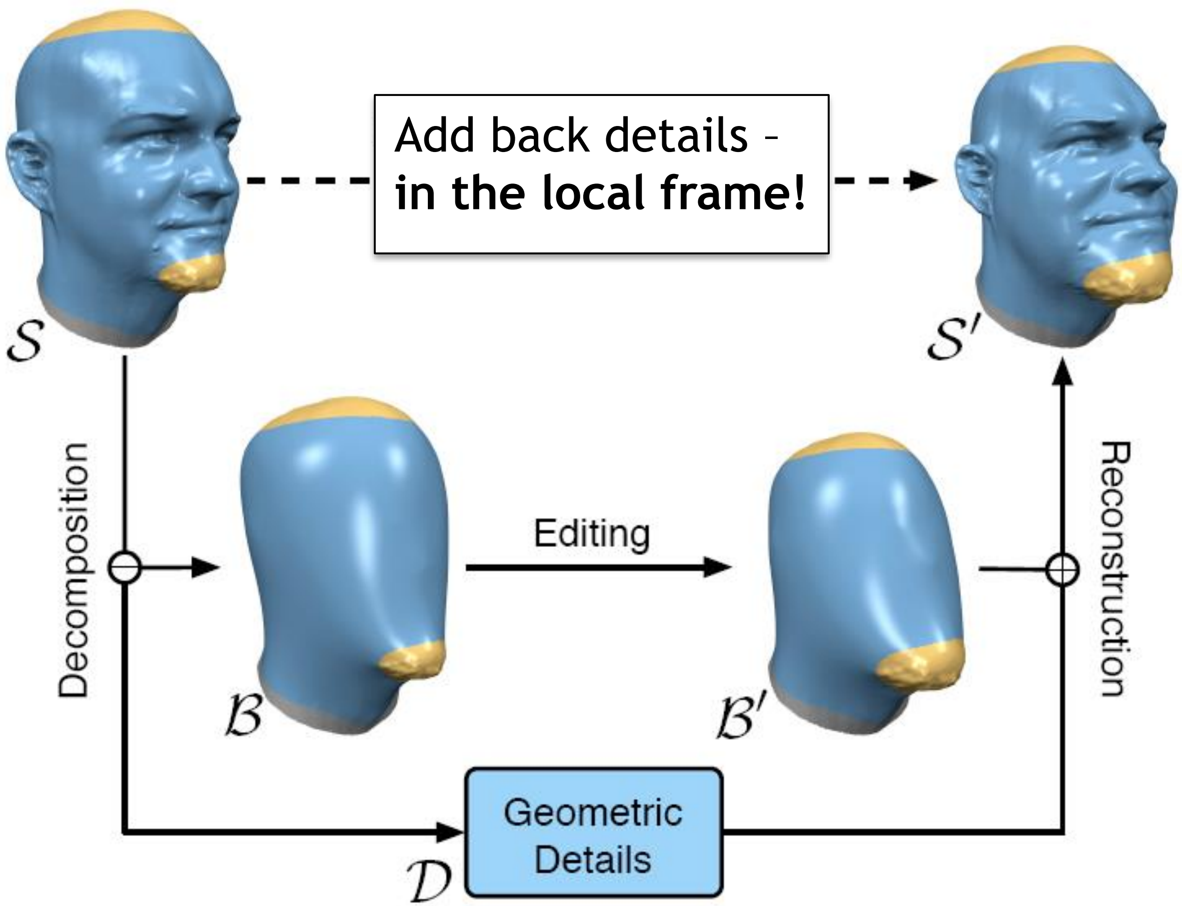
$$\min_{\mathbf{x}'} \ \mathbf{x}'^T (L_w M^{-1} L_w)\, \mathbf{x}'$$

with the **new** handle constraints



Fixing Local Rotations: Multiresolution Approach

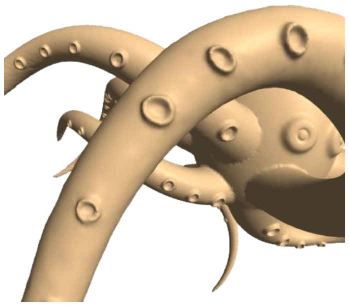
- Kobbelt et al. SIGGRAPH 98, Botsch and Kobbelt SIGGRAPH 2004

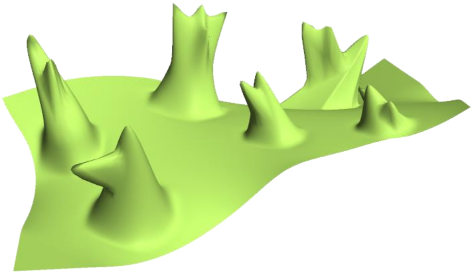


ETH zürich

Multiresolution Framework: Discussion

- Advantages:
  - Fast! Linear solve for the base surface deformation, and then add back displacements
  - Intuitive, easy to implement
- Problem: works only for small height fields (when details vectors are small)

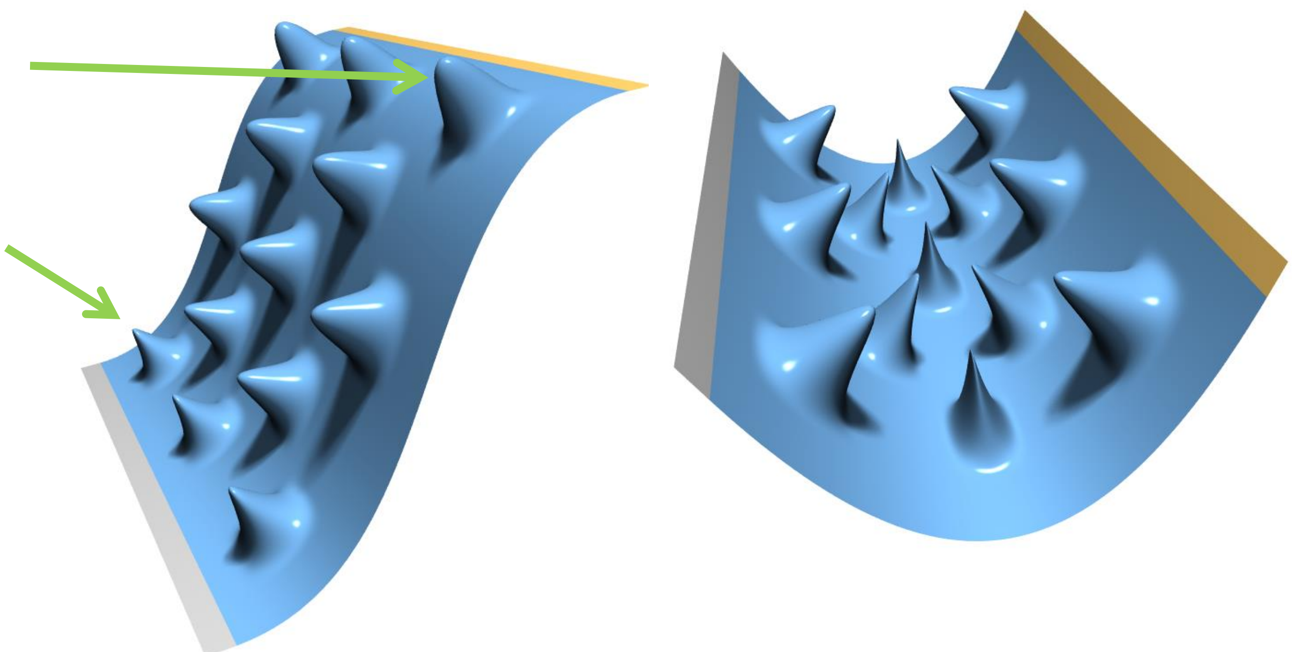

almost a height field


not a height field

ETH zürich

Multiresolution Framework: Discussion

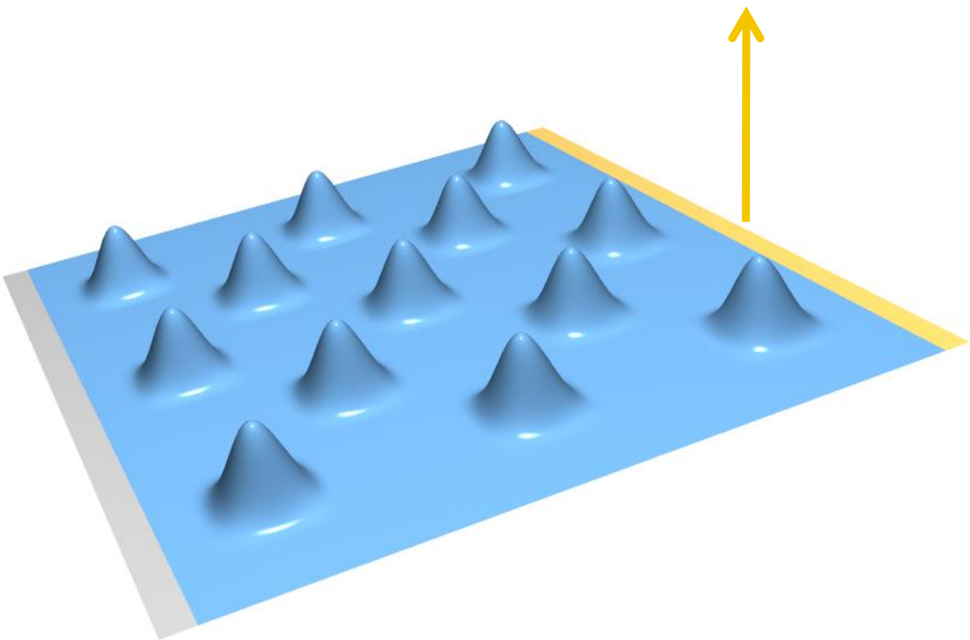
- Problem: If detail vectors are too big we get overshooting and **self-intersections**, especially in concave cases



ETH zürich

Local Rotations – Single Resolution Solutions

- Come up with a rotation field on the surface based on the modeling constraints
- Rotate the differential coordinates; solve

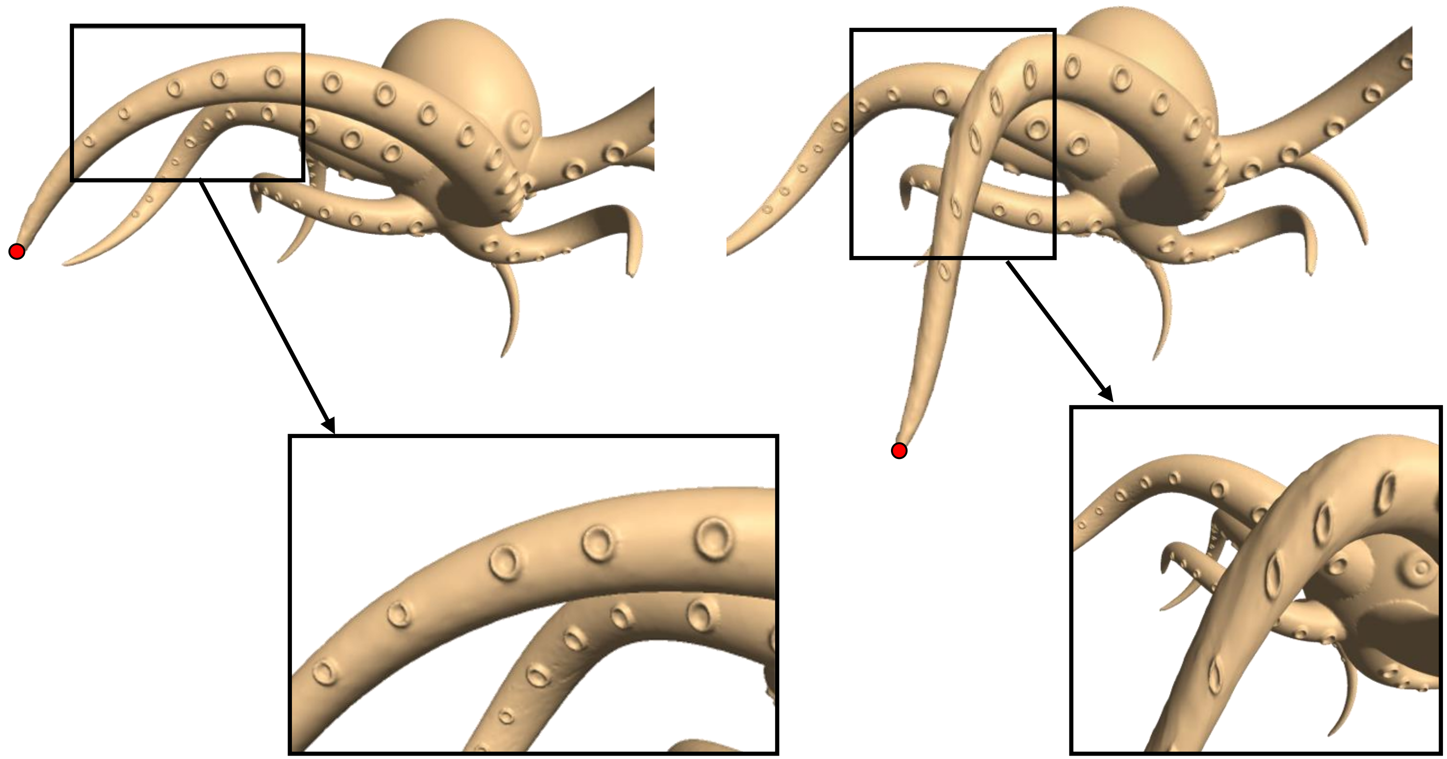


Estimation of Rotations

Lipman et al. 2004

http://igl.ethz.ch/projects/Laplacian-mesh-processing/Laplacian-mesh-editing/diffcoords-editing.pdf

- Edit the surface using the original Laplacians δ (naïve Laplacian editing)
- Compute smoothed local frames, estimate rotation



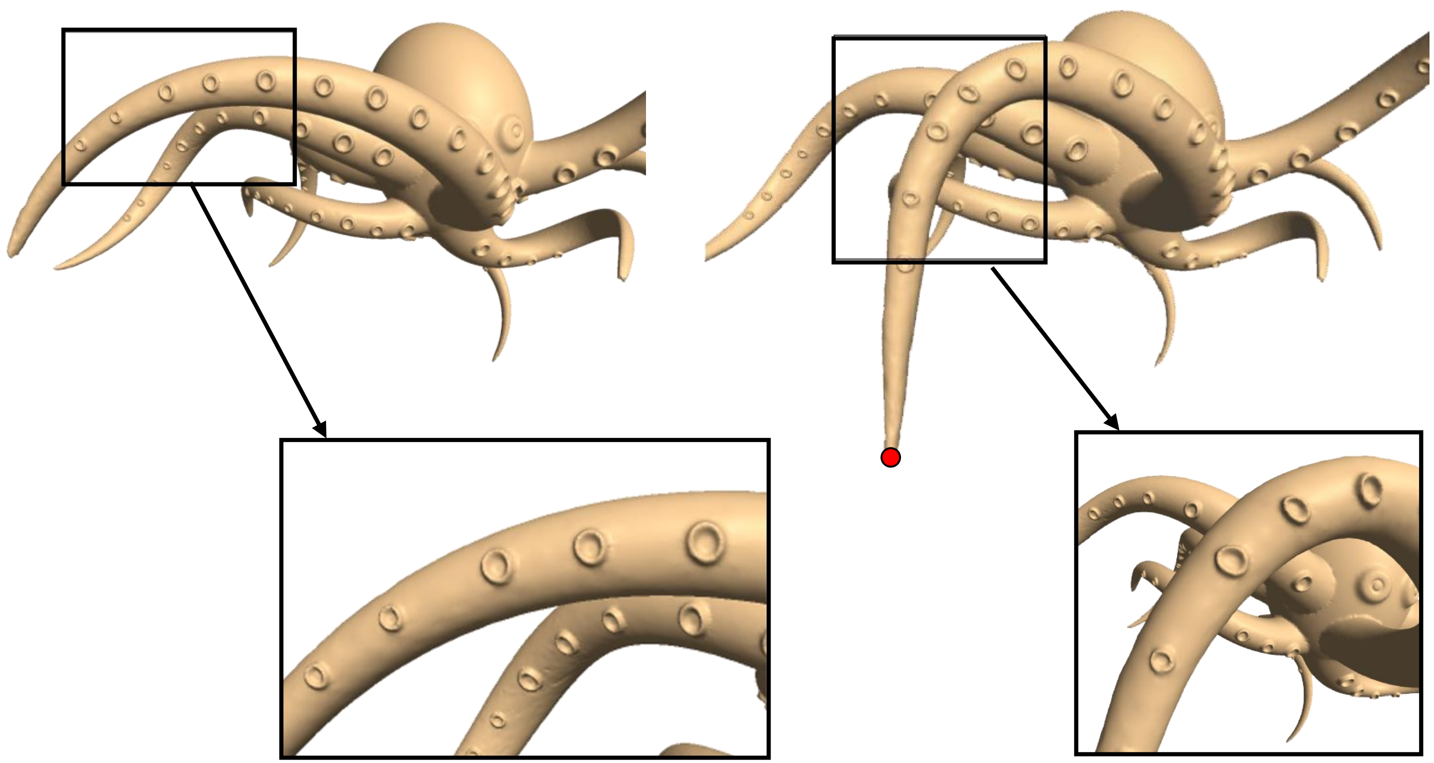
ETH zürich

Estimation of Rotations

Lipman et al. 2004

http://igl.ethz.ch/projects/Laplacian-mesh-processing/Laplacian-mesh-editing/diffcoords-editing.pdf

- Then solve the optimization again with the **rotated** δ's!



Estimation of Rotations

Lipman et al. 2004

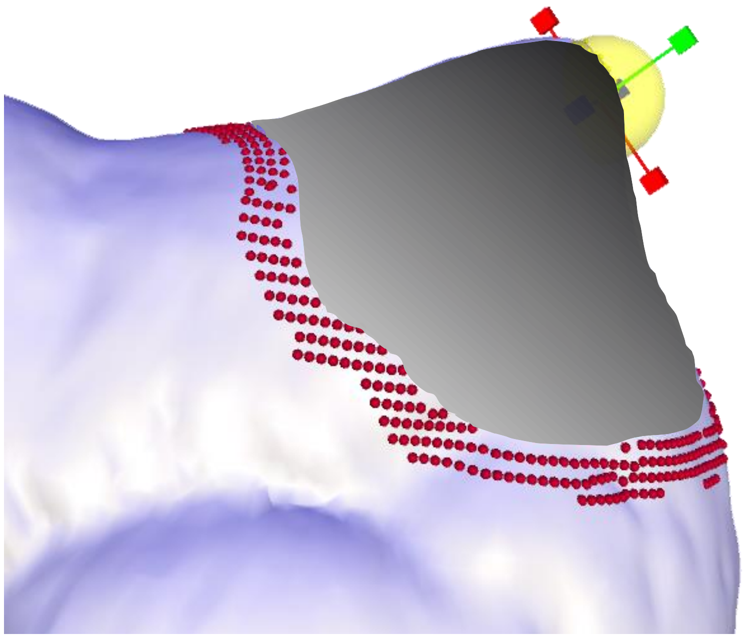
http://igl.ethz.ch/projects/Laplacian-mesh-processing/Laplacian-mesh-editing/diffcoords-editing.pdf

- Advantages:
  - Sparse linear solve
  - Less or no self-intersections thanks to global optimization (no more local displacements that fight each other)
- Disadvantages:
  - Heuristic estimation of the rotations
  - Speed depends on the support of the smooth local frame estimation operator; for highly detailed surfaces it must be large
  - Unclear how much we need to smooth (what is detail?)

Rotation Propagation

[Yu et al. SIGGRAPH 2004][Zayer et al. EG 2005][Lipman et al. SIGGRAPH 2005]

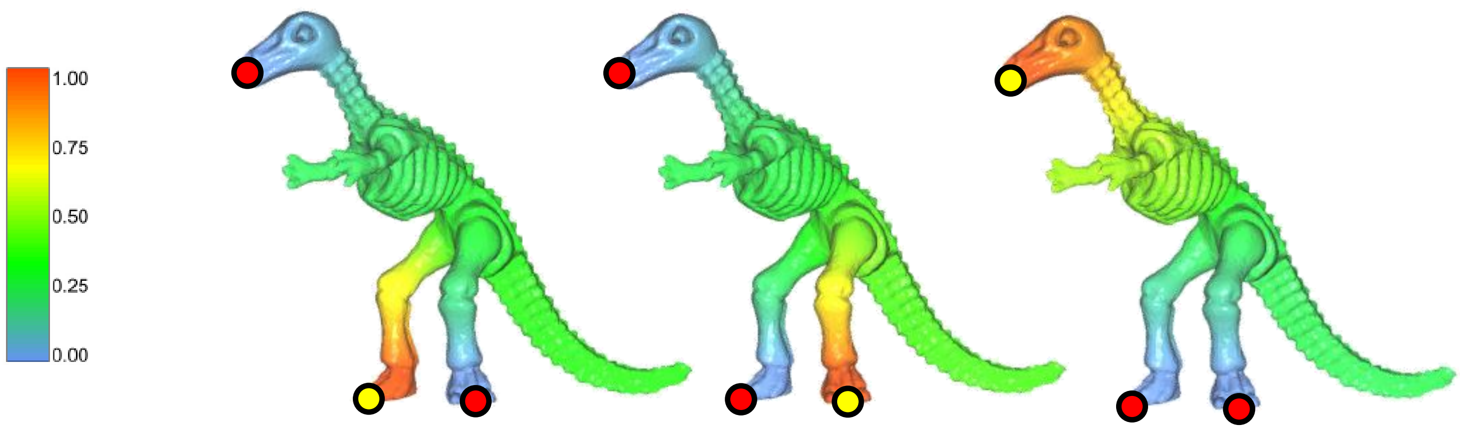
- Assume more user input: the user also specifies handle rotation, not just translation
- The rotation is diffused to the rest of the ROI



ETH zürich

Rotation Propagation

- Geodesic distance [Yu et al. 2004]
- Harmonic field [Zayer et al. 2005]
- Optimization [Lipman et al. 2005, 2007]



Harmonic field

ETH zürich

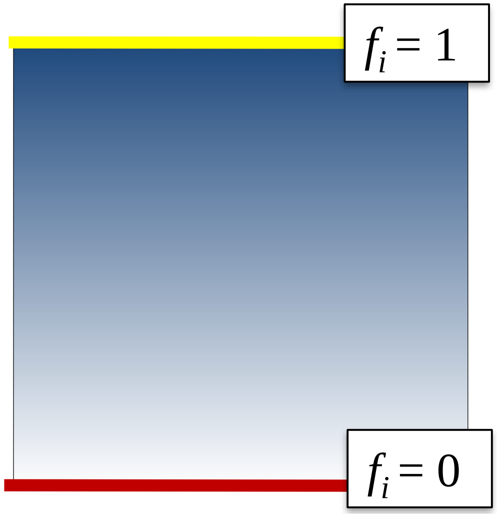
Harmonic Fields on Meshes

- Scalar function, attains 1 on the active handle, 0 on the static handles
- Smooth away from handles, no local extrema (maximum principle)
- Solve:

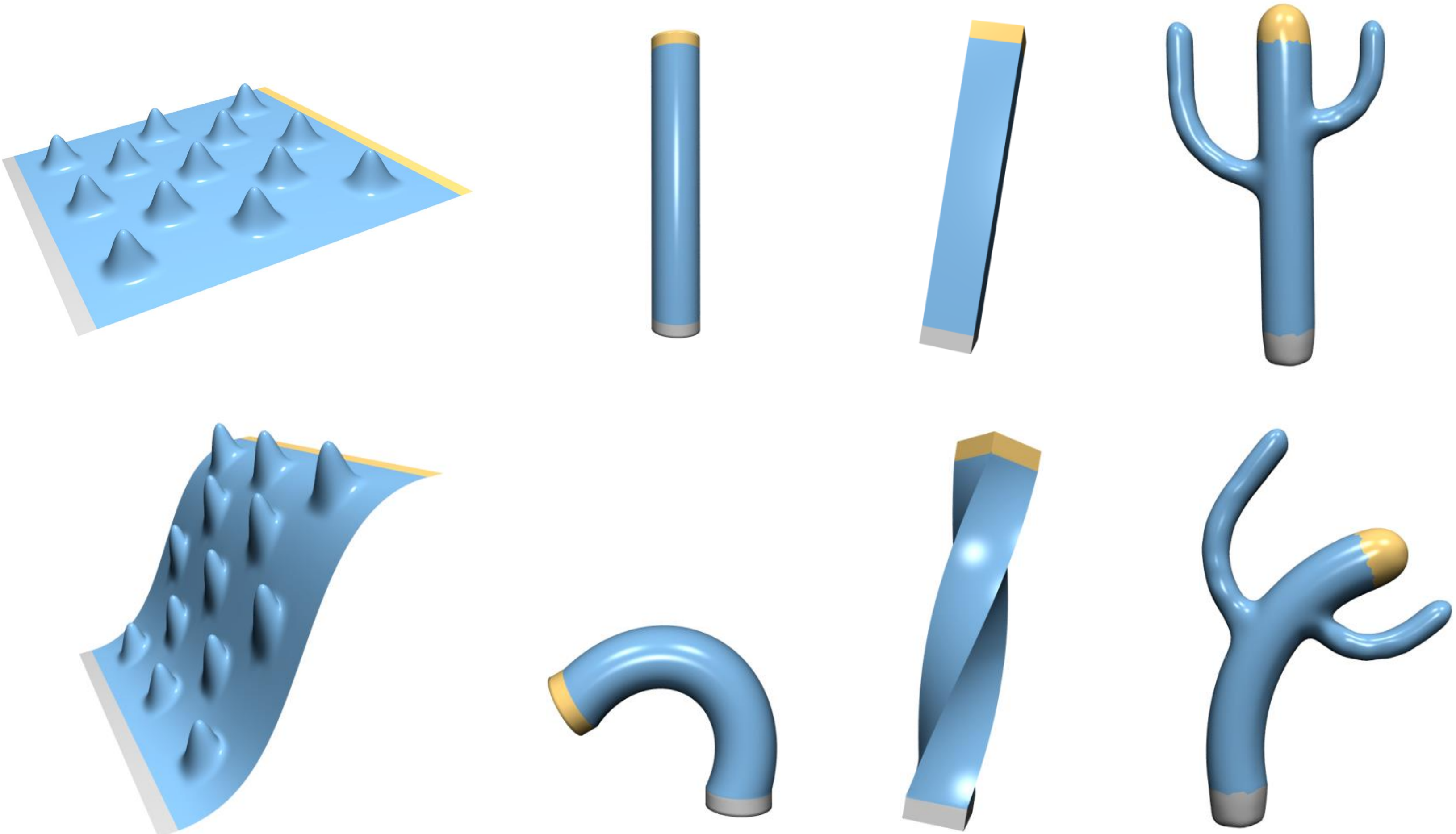
$$\Delta \mathbf{f} = 0$$

with constraints $f_i = 1$ on active handle, $f_i = 0$ on static handle

Example: in this simple case, the harmonic field is a just a linear ramp



Rotation Propagation w/Harmonic Fields



Why does this happen?

ETH zürich

Rotation Propagation w/Harmonic Fields

- If rotations are provided and consistent with the desired transformation, this works well
- However, the method is translation-insensitive (doesn’t generate rotations when there are none provided)

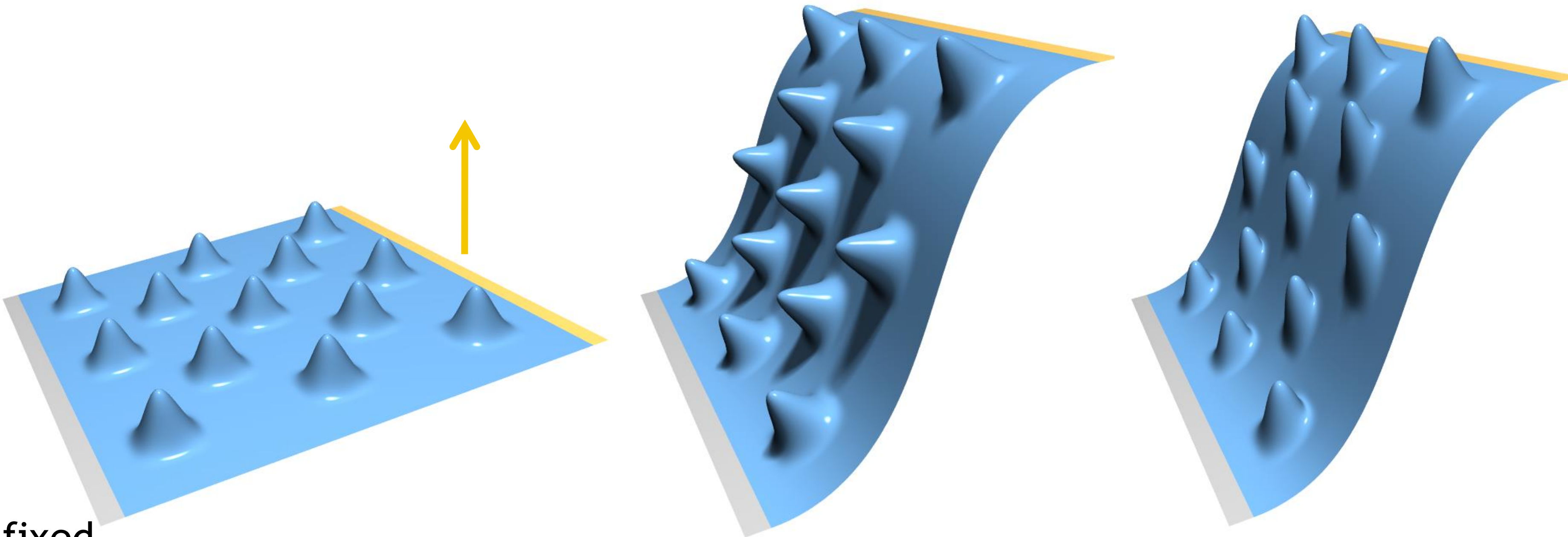
Literature – rotation propagation

- Yu et al. 2004: **Mesh Editing with Poisson-Based Gradient Field Manipulation,** ACM SIGGRAPH 2004
- Lipman et al. 2005: **Linear Rotation-Invariant Coordinates for Meshes,** ACM SIGGRAPH 2005
- Zayer et al. 2005: **Harmonic Guidance for Surface Deformation,** EUROGRAPHICS 2005
- Lipman et al. 2007: **Volume and Shape Preservation via Moving Frame Manipulation,** ACM Transactions on Graphics 26(1), 2007

Demo

- Libigl tutorial 401, 402

Fundamental Problem: Invariance to Transformations

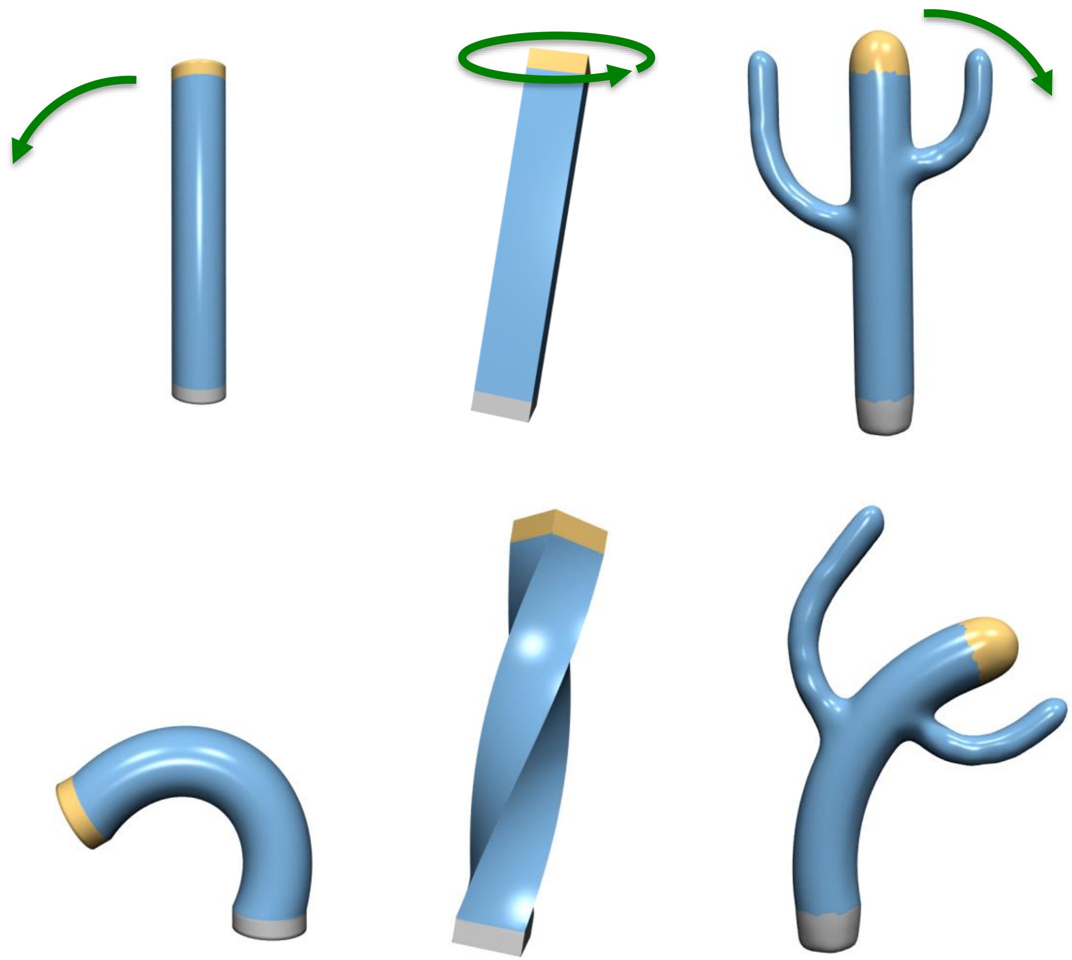


fixed vertices

input | intuitive expectation | actual result

Rotation Propagation

- Propagate using a harmonic scalar field from handle to fixed region
- Works for single handle and user prescribes same rotation for all vertices in the handle

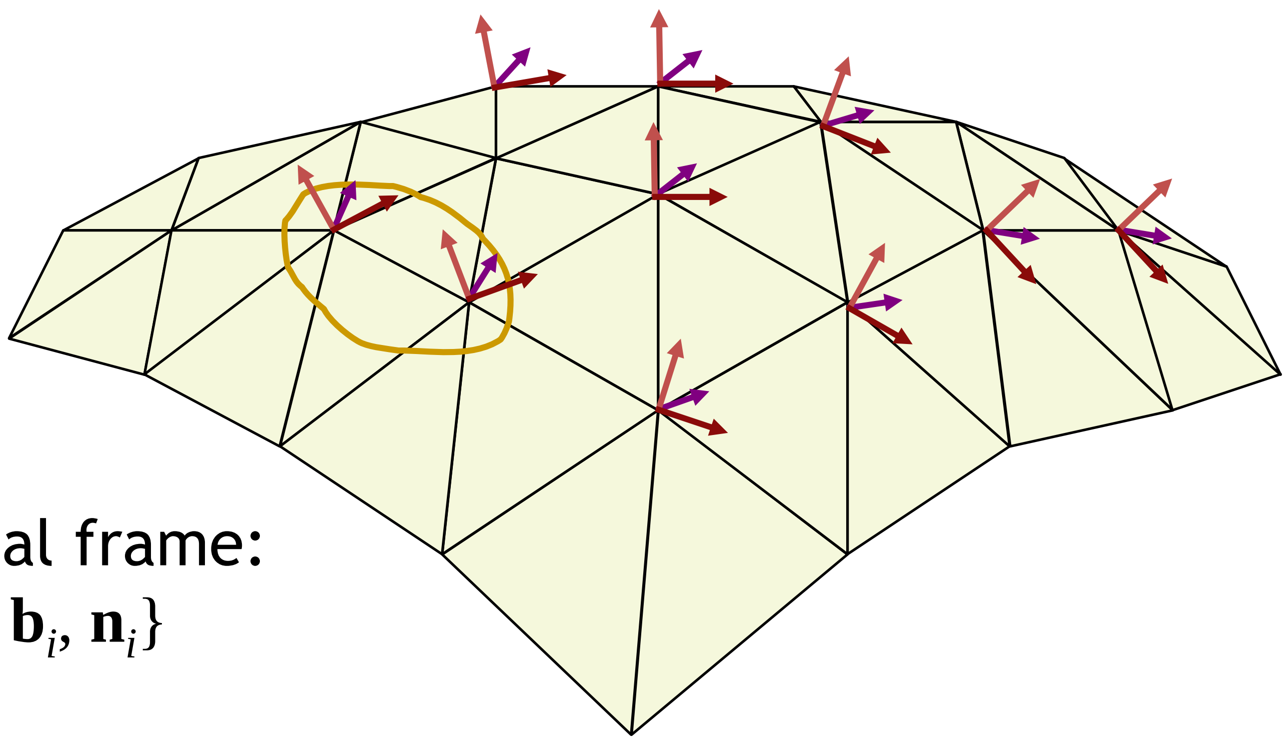


ETH zürich

Optimization of Rotation Propagation

Lipman et al. 2005

- Keep a local orthonormal frame at each vertex
- Prescribe changes to some selected frames (rotation/scaling)



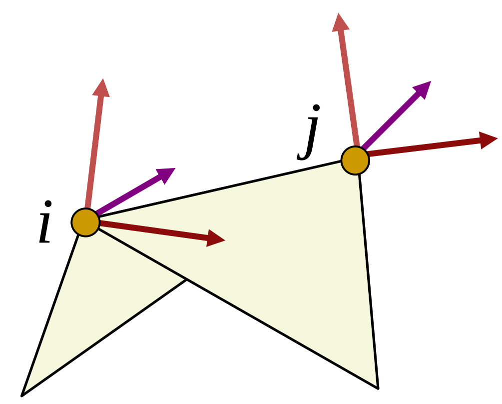
Local frame:
$\{\mathbf{a}_i, \mathbf{b}_i, \mathbf{n}_i\}$

Optimization of Rotation Propagation

Lipman et al. 2005

- Encode the differences between adjacent frames in the original mesh – the numbers □ □ □ for each edge
  - Represented in the local frame coord. system!
- Want the deformed surface to have the same local differences
  - Rotation-invariant representation!

$$
\begin{aligned}
\mathbf{a}_i - \mathbf{a}_j &= \alpha_{i,1}\mathbf{a}_i + \alpha_{i,2}\mathbf{b}_i + \alpha_{i,3}\mathbf{n}_i \\
\mathbf{b}_i - \mathbf{b}_j &= \beta_{i,1}\mathbf{a}_i + \beta_{i,2}\mathbf{b}_i + \beta_{i,3}\mathbf{n}_i \\
\mathbf{n}_i - \mathbf{n}_j &= \gamma_{i,1}\mathbf{a}_i + \gamma_{i,2}\mathbf{b}_i + \gamma_{i,3}\mathbf{n}_i
\end{aligned}
$$



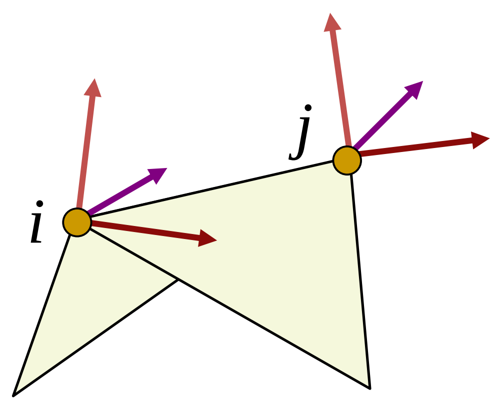
Optimization of Rotation Propagation

Lipman et al. 2005

- Solve for the new frames in least-squares sense
- Need to orthogonalize (and normalize) post-hoc

$$\min_{\mathbf{a},\mathbf{b},\mathbf{n}} \sum_{i=1}^{n} \begin{aligned} &\|(\mathbf{a}'_i - \mathbf{a}'_j) - (\alpha_{i,1}\mathbf{a}'_i + \alpha_{i,2}\mathbf{b}'_i + \alpha_{i,3}\mathbf{n}'_i)\|^2 + \\ &\|(\mathbf{b}'_i - \mathbf{b}'_j) - (\beta_{i,1}\mathbf{a}'_i + \beta_{i,2}\mathbf{b}'_i + \beta_{i,3}\mathbf{n}'_i)\|^2 + \\ &\|(\mathbf{n}'_i - \mathbf{n}'_j) - (\gamma_{i,1}\mathbf{a}'_i + \gamma_{i,2}\mathbf{b}'_i + \gamma_{i,3}\mathbf{n}'_i)\|^2 \end{aligned}$$

$$s.t. \quad (\mathbf{a}'_k, \mathbf{b}'_k, \mathbf{n}'_k) = \mathbf{M}_k,\ k \in \mathcal{C}$$

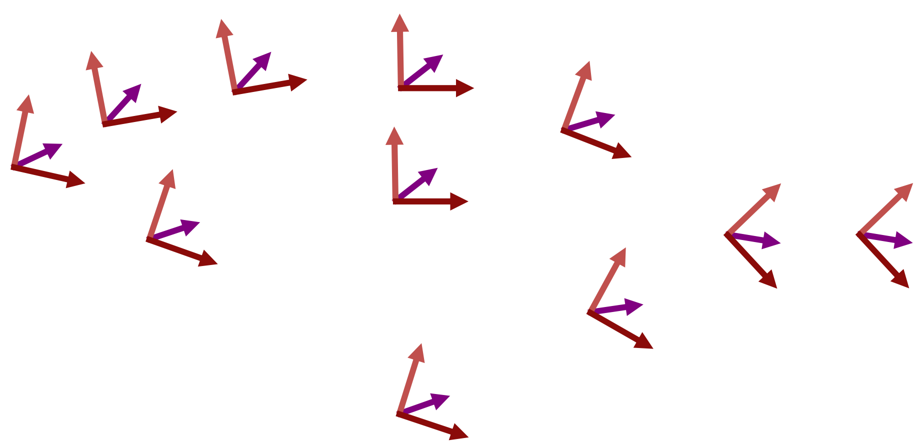


ETH zürich

Optimization of Rotation Propagation

Lipman et al. 2005

- After solving the frames, solve for positions using e.g. naïve Laplacian editing (rotate each delta-vector…)

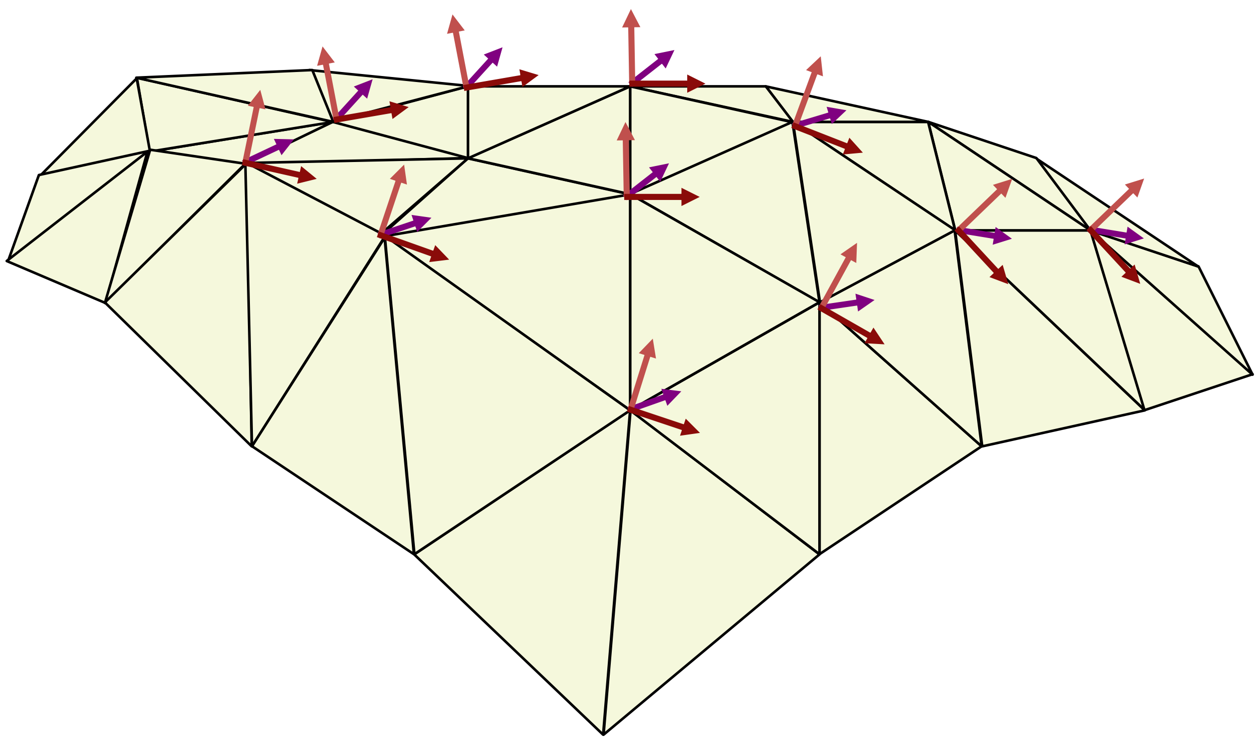


ETH zürich

Optimization of Rotation Propagation

Lipman et al. 2005

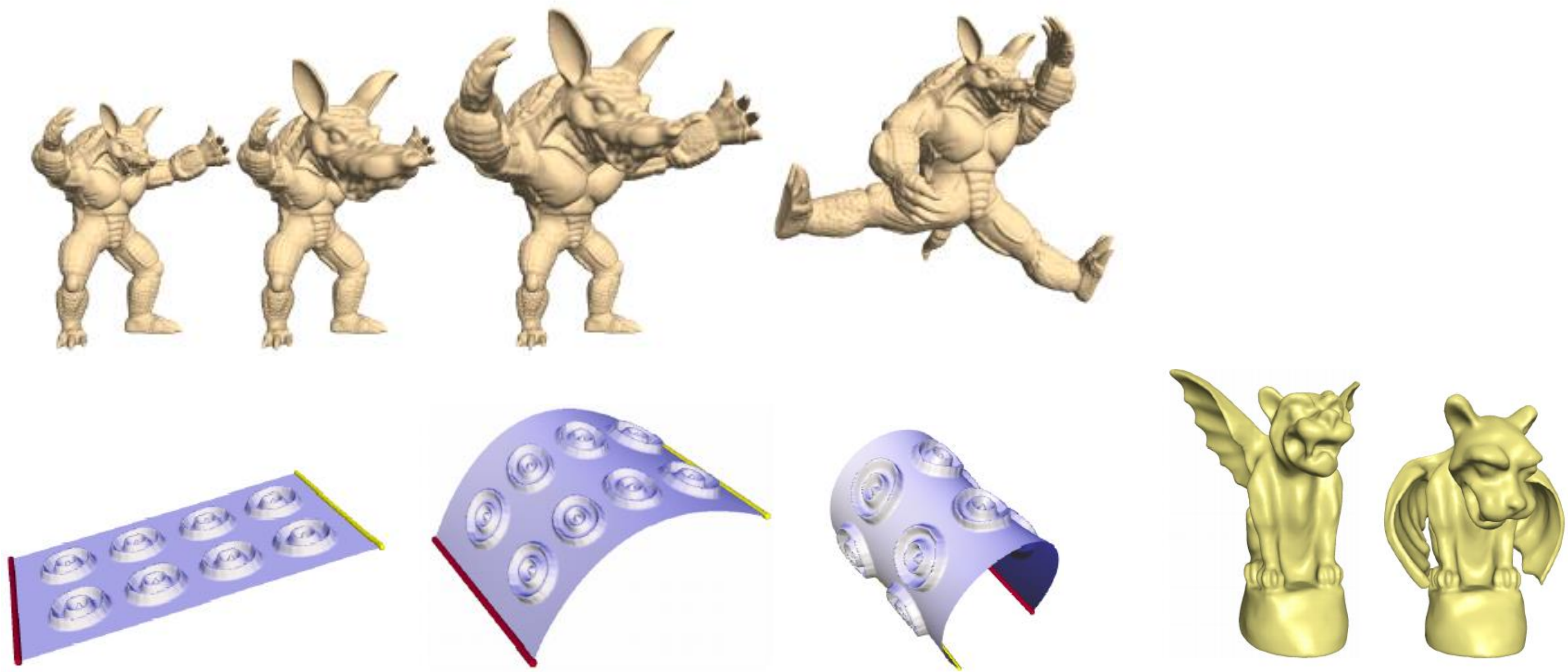
- After solving the frames, solve for positions using e.g. naïve Laplacian editing (rotate each delta-vector…)



Optimization of Rotation Propagation

Lipman et al. 2005

- Some results



Optimization of Rotation Propagation

Lipman et al. 2005

- Can use this representation for shape interpolation

Linear Rotation-Invariant Coordinates
for Meshes

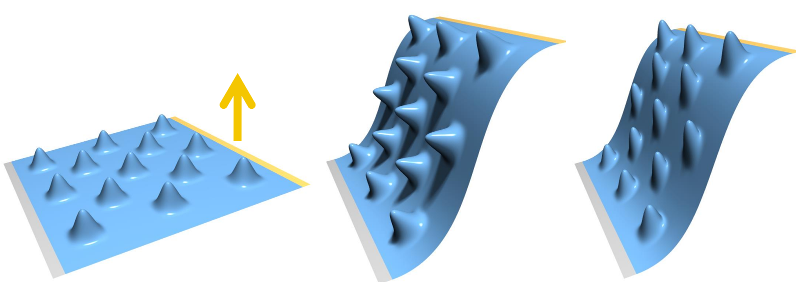
Yaron Lipman
Olga Sorkine
David Levin
Daniel Cohen-Or

Tel Aviv University

Rotation Propagation – Summary

- Linear optimization to find the local frames of the deformed surface
- Works well even for large rotations of the handle

- Does not work if there is no rotation to propagate – translation insensitivity



ETH zürich

Implicit Definition of Transformations

Sorkine et al. 2004

- The idea: solve for local transformations AND the edited surface simultaneously!
- Estimate the local transformations $\mathbf{T}_i$ from the eventual solution

$$E(\mathbf{x}') = \sum_{i=1}^{n} A_i \|\Delta(\mathbf{x}'_i) - \mathbf{T}_i \delta_i\|^2$$

Linear transformation of the local frame

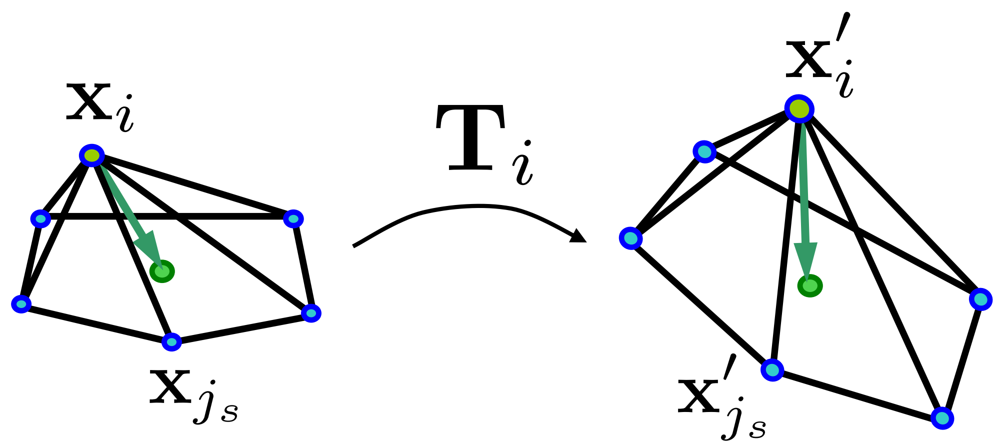
ETH zürich

Defining $\mathbf{T}_i$

$$E(\mathbf{x}') = \sum_{i=1}^{n} A_i \|\Delta(\mathbf{x}'_i) - \mathbf{T}_i \delta_i\|^2$$

- How to formulate $\mathbf{T}_i$?
  - Based on the local (1-ring) neighborhood
  - Linear dependence on the unknown $\mathbf{x}'_i$

$$\mathbf{x}'_{j1} - \mathbf{x}'_i = \mathbf{T}_i(\mathbf{x}_{j1} - \mathbf{x}_i)$$
$$\vdots$$
$$\mathbf{x}'_{jk} - \mathbf{x}'_i = \mathbf{T}_i(\mathbf{x}_{jk} - \mathbf{x}_i)$$



Defining $\mathbf{T}_i$

- First attempt: define $\mathbf{T}_i$ simply by solving

$$\mathbf{T}_i = \underset{\mathbf{T}_i}{\operatorname{argmin}} \sum_{s=1}^{k} \left\| (\mathbf{x}'_{j_s} - \mathbf{x}'_i) - \mathbf{T}_i (\mathbf{x}_{j_s} - \mathbf{x}_i) \right\|^2$$

$$\Downarrow$$

$$\mathbf{T}_i = \begin{pmatrix} | & | & & | \\ (\mathbf{x}'_{j_1} - \mathbf{x}'_i) & (\mathbf{x}'_{j_2} - \mathbf{x}'_i) & \cdots & (\mathbf{x}'_{j_k} - \mathbf{x}'_i) \\ | & | & & | \end{pmatrix} \begin{pmatrix} | & | & & | \\ (\mathbf{x}_{j_1} - \mathbf{x}_i) & (\mathbf{x}_{j_2} - \mathbf{x}_i) & \cdots & (\mathbf{x}_{j_k} - \mathbf{x}_i) \\ | & | & & | \end{pmatrix}^{+}$$

pseudoinverse

Defining $\mathbf{T}_i$

- Plug the expressions for $\mathbf{T}_i$ into the energy formula:

$$E(\mathbf{x}') = \sum_{i=1}^{n} A_i \|\Delta(\mathbf{x}'_i) - \mathbf{T}_i \delta_i\|^2$$

Linear combination of the unknown $\mathbf{x}'$

But: we didn't solve anything since $\mathbf{T}_i$ is an arbitrary linear transformation, i.e. admits distorting shears

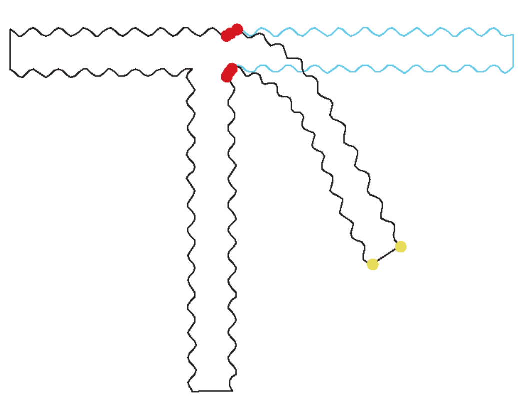
Constraining $\mathbf{T}_i$

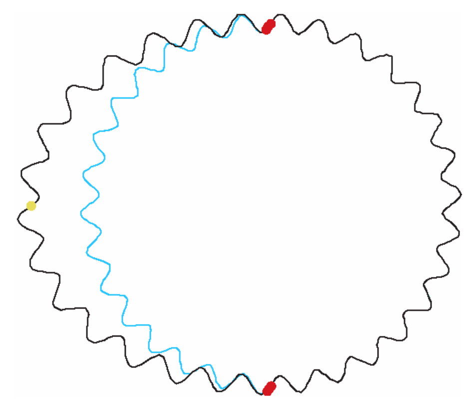
- Rotation + scale (i.e., similarity) is easy in 2D:

$$\mathbf{T}_i = \begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} = \begin{pmatrix} a & b \\ -b & a \end{pmatrix}$$

  - Rotation alone is nonlinear (bounds on a and b)

- Can edit 2D curves:





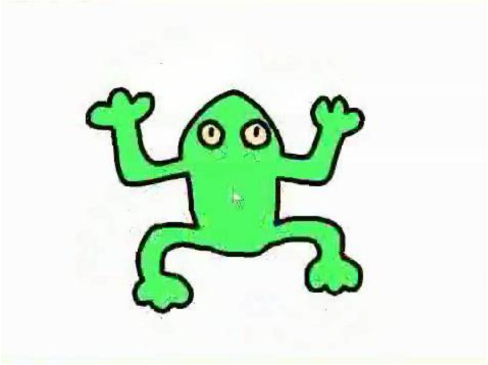
Constraining $\mathbf{T}_i$

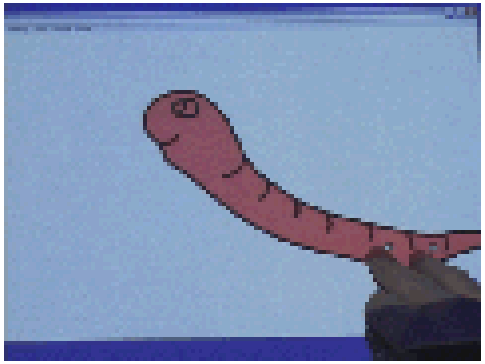
- Rotation + scale (i.e., similarity) is easy in 2D:

$$\mathbf{T}_i = \begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} = \begin{pmatrix} a & b \\ -b & a \end{pmatrix}$$

  - Rotation alone is nonlinear (bounds on a and b)

- Similar idea applied in [Igarashi et al. 2005] for 2D shape manipulation:





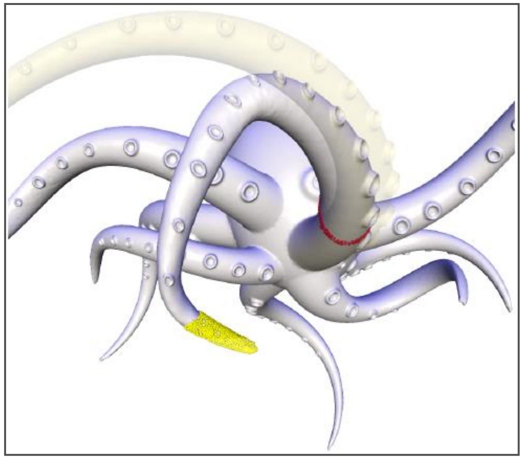
ETH zürich

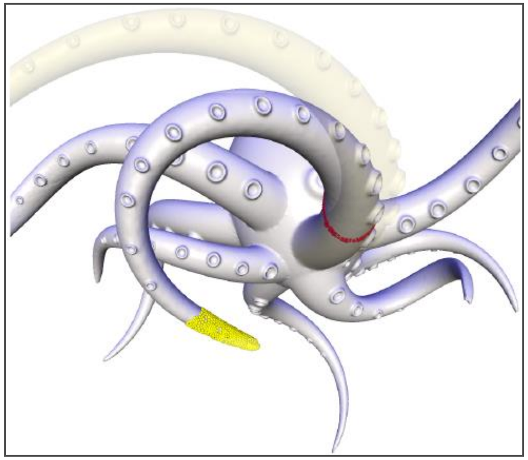
Constraining $\mathbf{T}_i$

- In 3D: even similarity has nonlinear form. Linearization of rotations – first order Taylor approximation:

$$\mathbf{T}_i = \begin{pmatrix} 1 & -h_3 & h_2 \\ h_3 & 1 & -h_1 \\ -h_2 & h_1 & 1 \end{pmatrix}$$

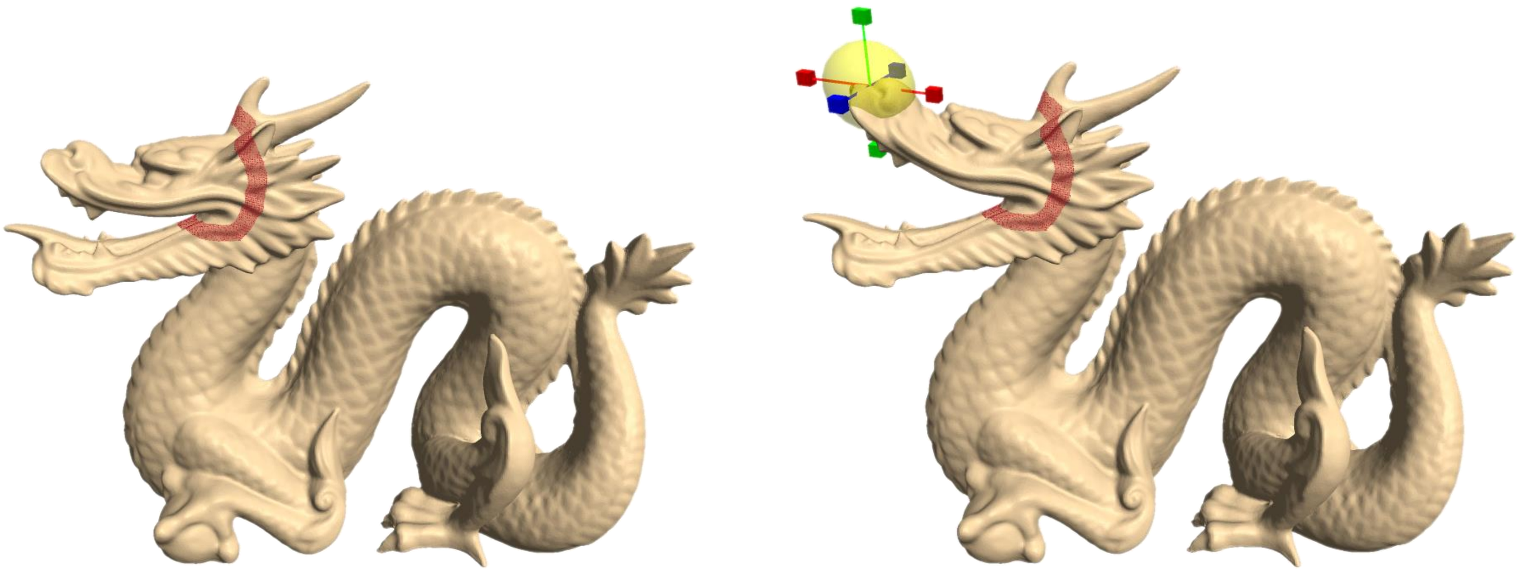
- Works well for moderate rotations, problems with large rotation angles



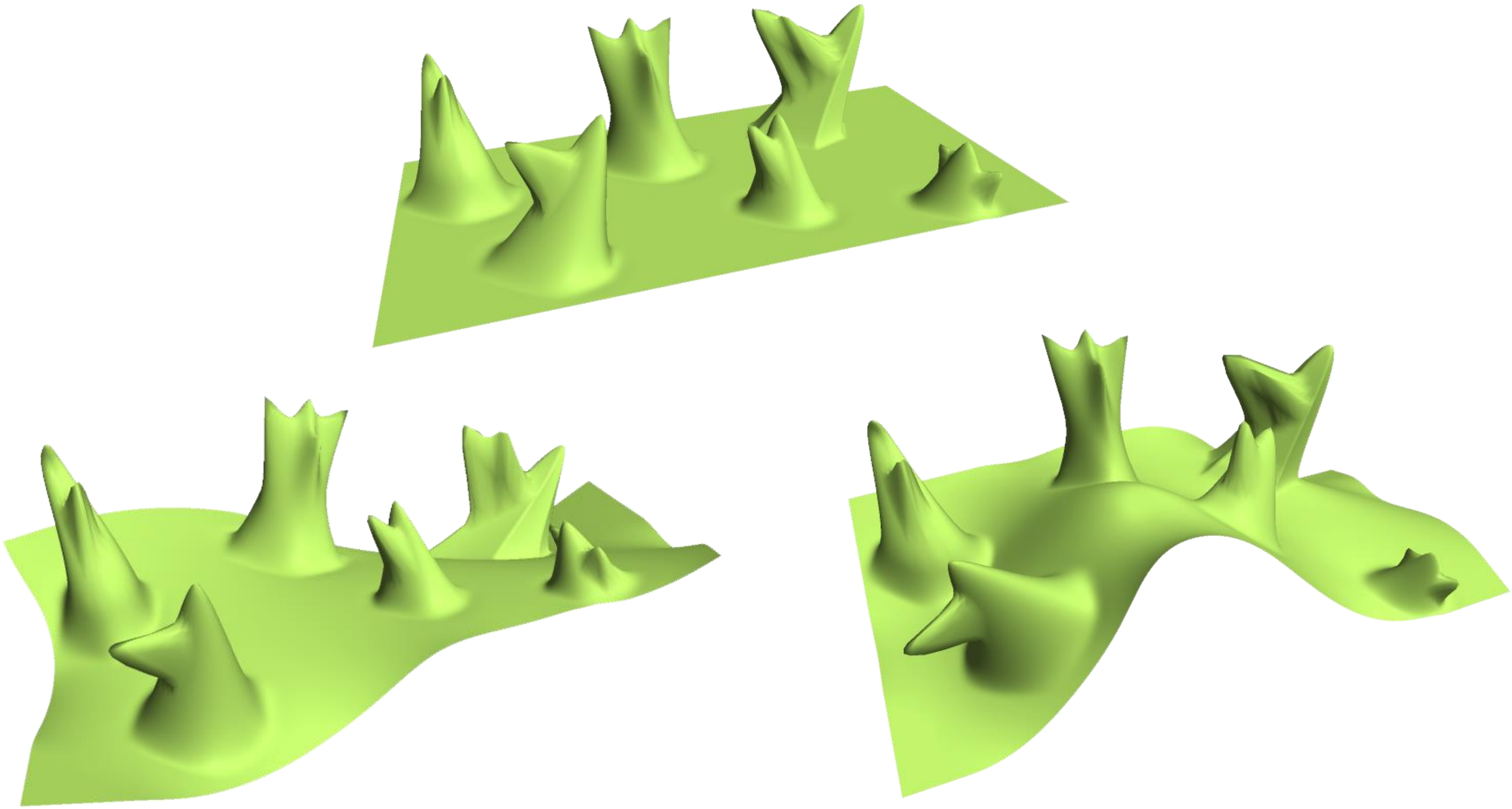


ETH zürich

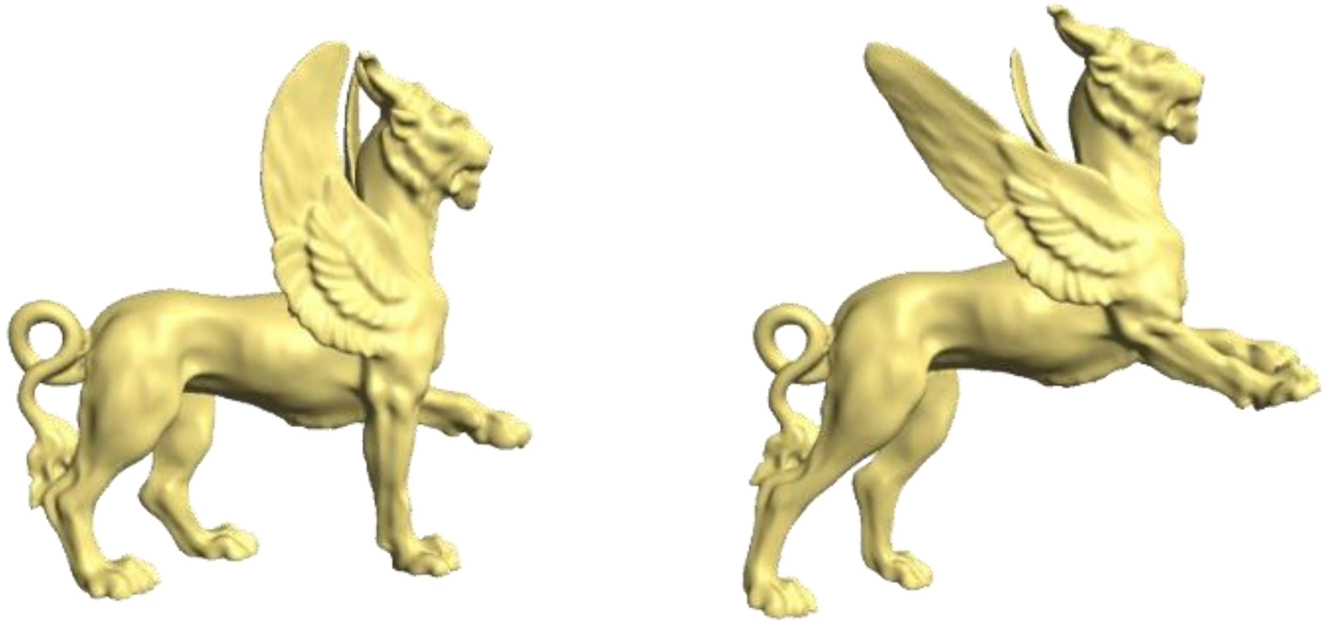
Laplacian Editing Results



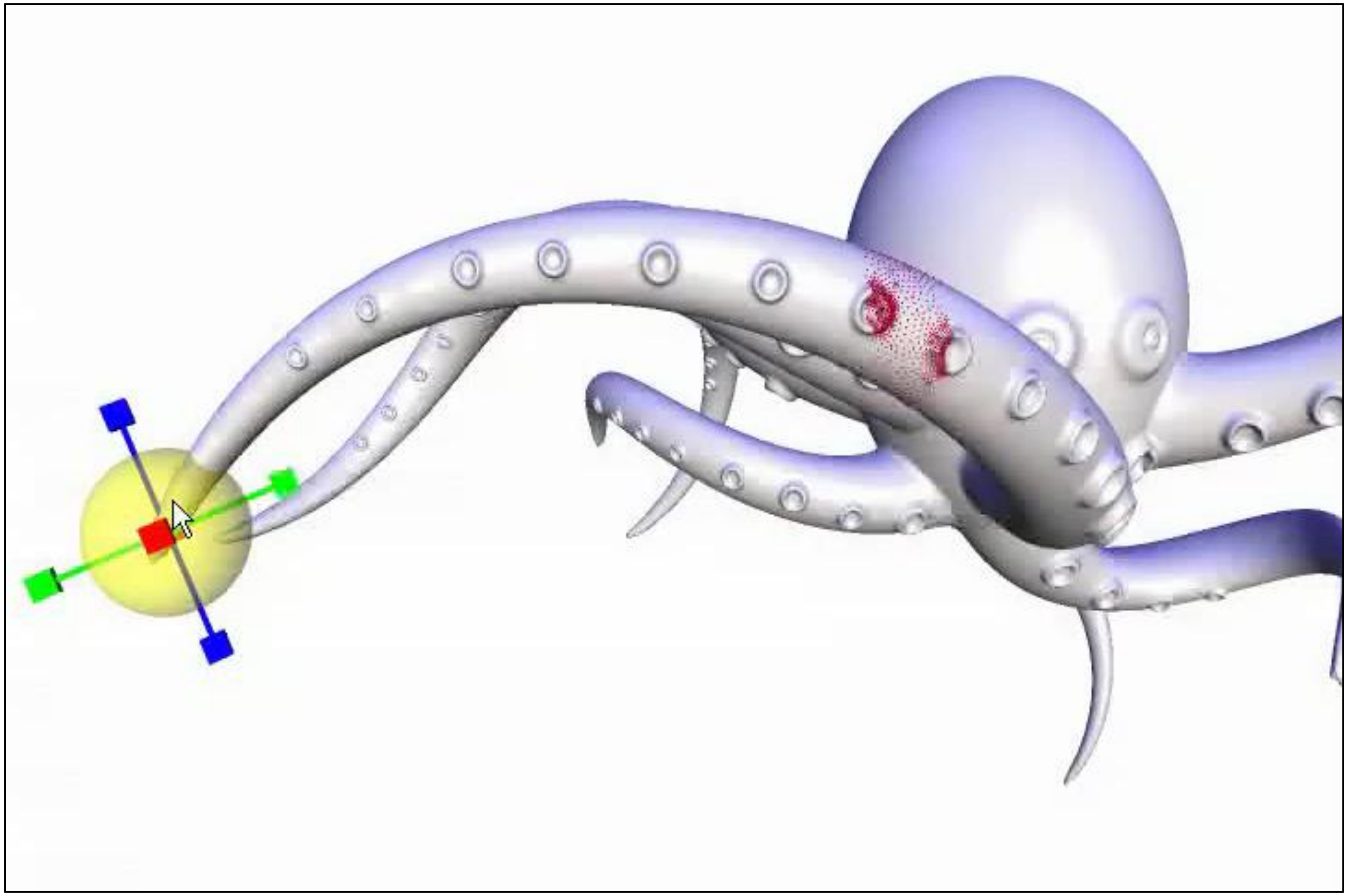
Laplacian Editing Results



Laplacian Editing Results



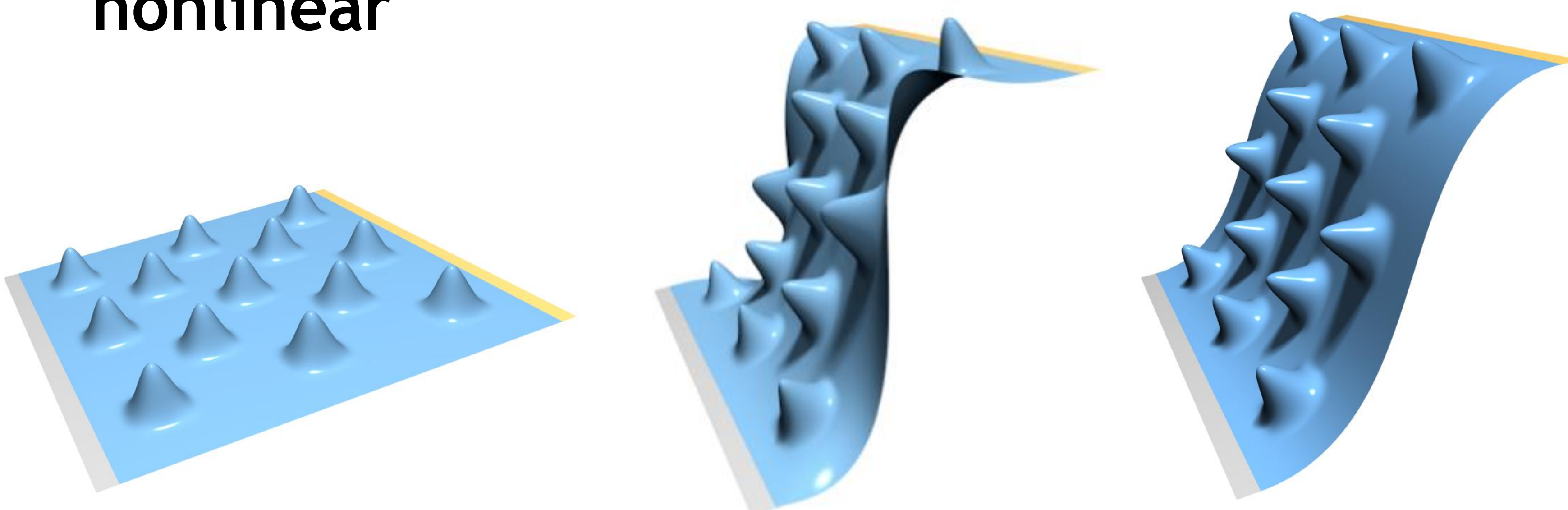
Laplacian Editing Results



ETH zürich

Linear Deformation Methods: Summary

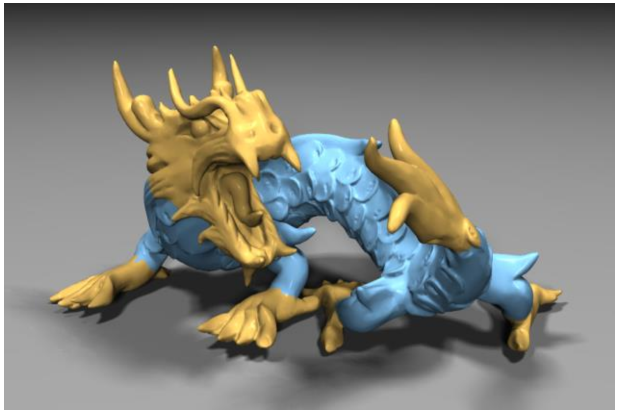
- Involve **linear** global optimization (efficient)
- Suffer from artifacts because of **local rotations**
- The relationship between the **translation** of a handle and the local **rotation** is inherently **nonlinear**



ETH zürich

Nonlinear Surface-based Deformations

- Formulate a nonlinear functional $E(\mathbf{x}')$ that handles local rotations properly
- Still need an efficient minimization method…

252-0538-00L, Spring 2017

Shape Modeling and Geometry Processing

As-Rigid-As-Possible Surface Modeling

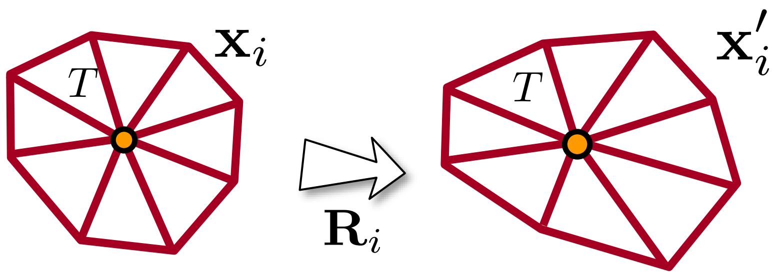
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Demo

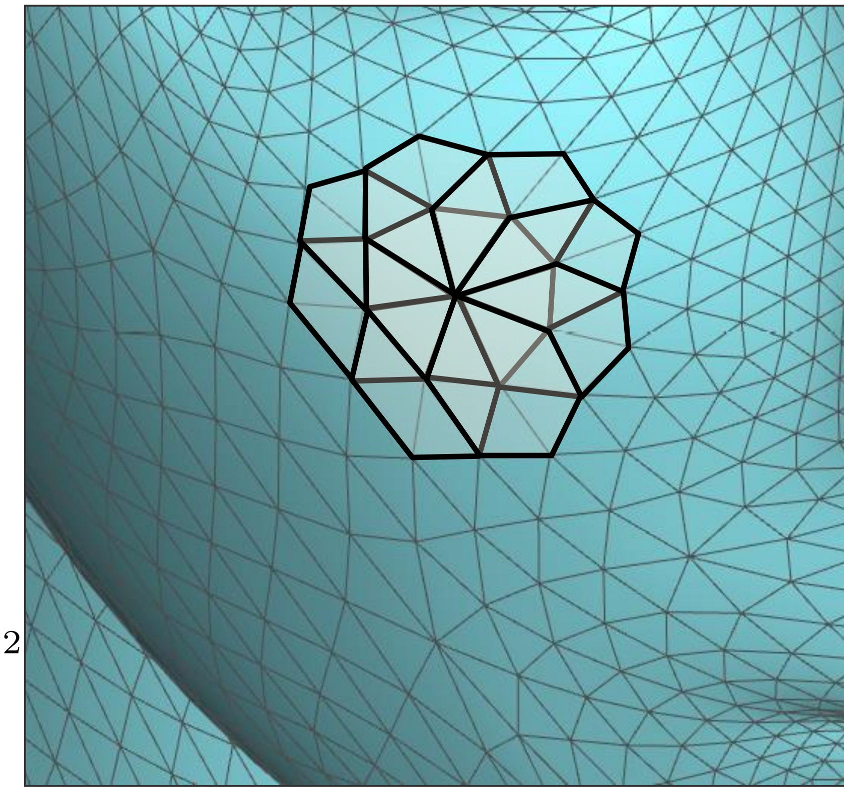
- Libigl demo 405

As-Rigid-As-Possible Deformation

- Preserve shape of cells covering the surface
- Ask each cell i to transform **rigidly** by best-fitting rotation $\mathbf{R}_i$

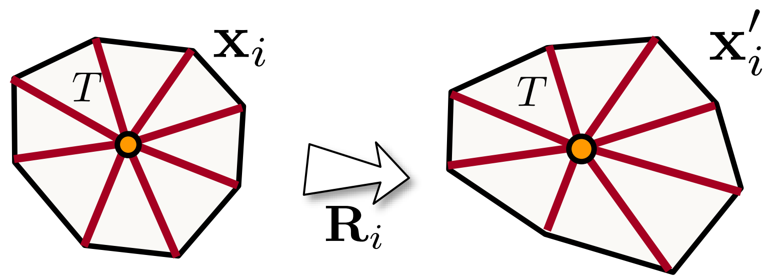


$$\min \sum_{T \in \mathrm{Cell}_i} \sum_{(j,k) \in T} \|(\mathbf{x}_j' - \mathbf{x}_k') - \mathbf{R}_i(\mathbf{x}_j - \mathbf{x}_k)\|^2$$



As-Rigid-As-Possible Deformation

- Optimal $\mathbf{R}_i$ is uniquely defined by $\mathbf{x}_i$, $\mathbf{x}'_i$

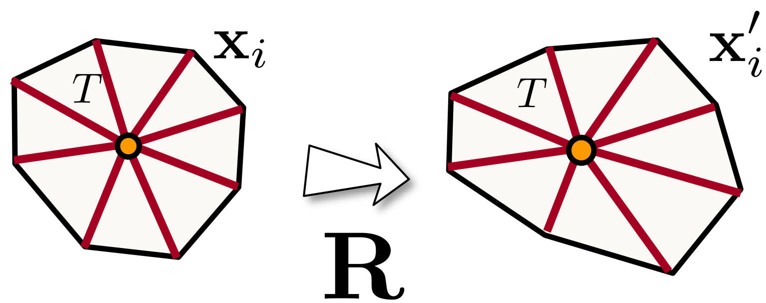


$$\min \sum_{T \in \mathrm{Cell}_i} \sum_{(j,k) \in T} \|(\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}_i(\mathbf{x}_j - \mathbf{x}_k)\|^2$$

- so-called shape-matching problem, solved by a 3x3 SVD

$\mathbf{R}_i$ is a nonlinear function of $\mathbf{x}$

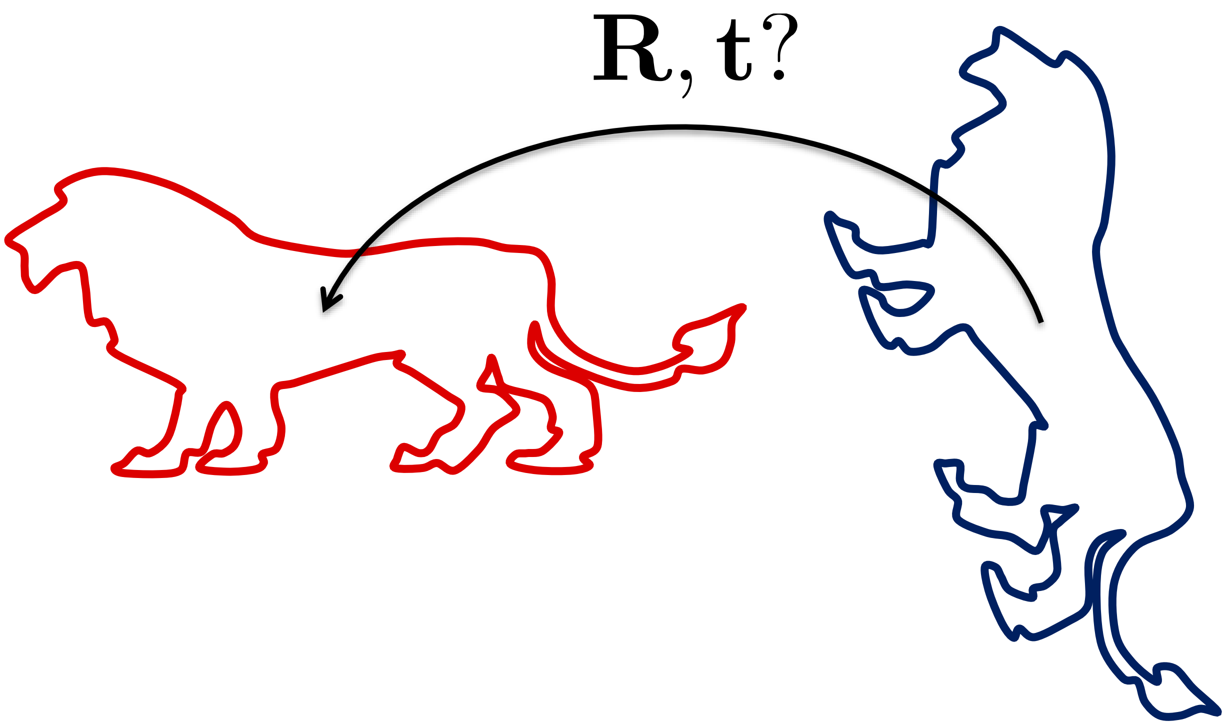
Optimal Rotation



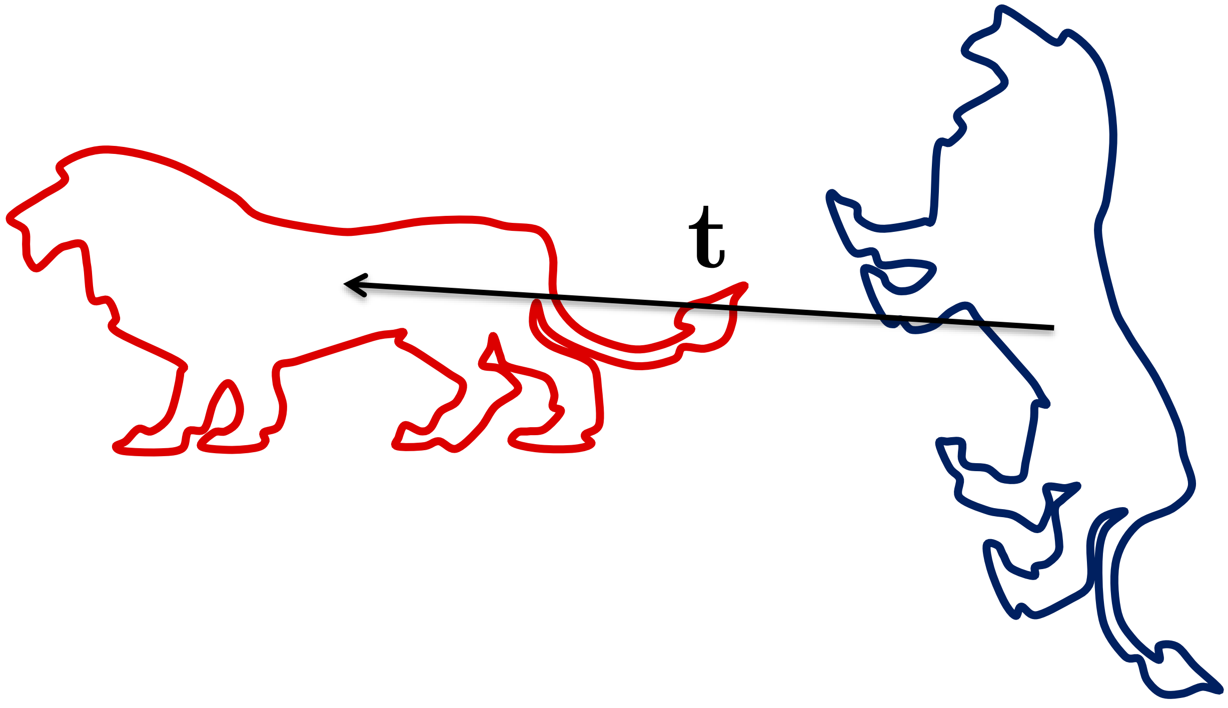
$$\min_{\mathbf{R}\in SO(3)} \sum_{T\in \mathrm{Cell}_i} \sum_{(j,k)\in T} \|(\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}(\mathbf{x}_j - \mathbf{x}_k)\|^2$$

Rotation group

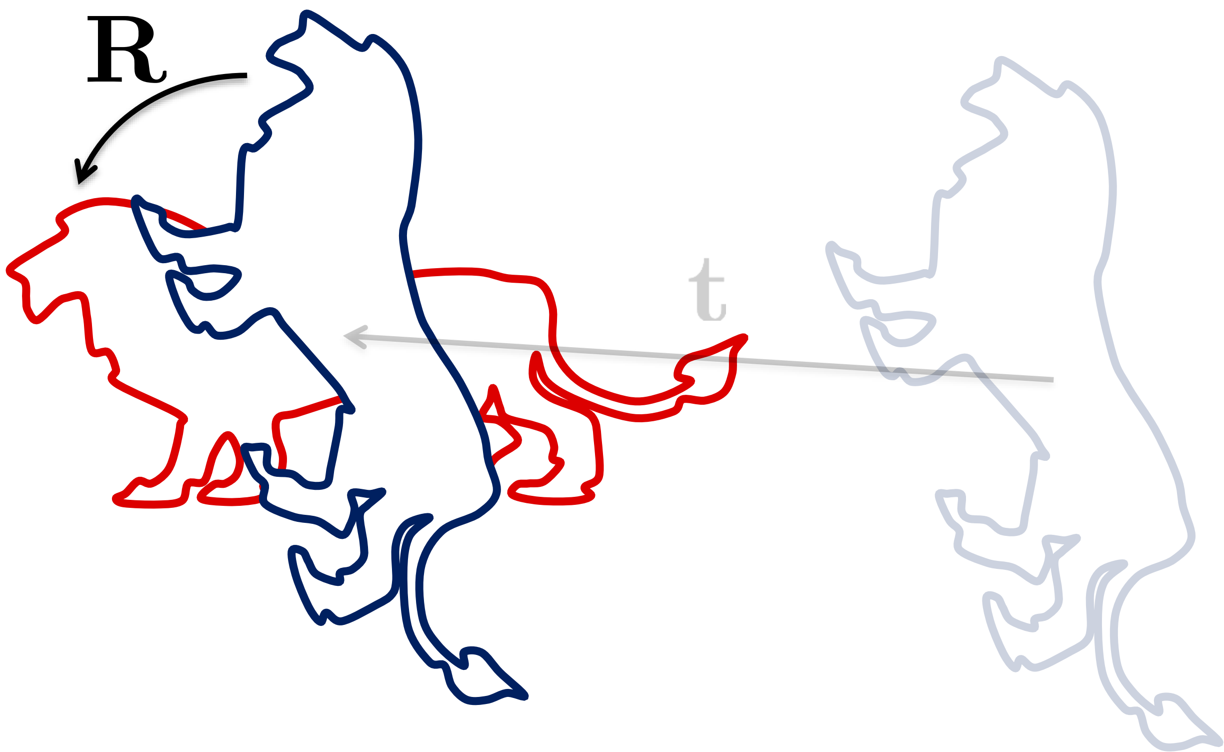
Shape Matching Problem



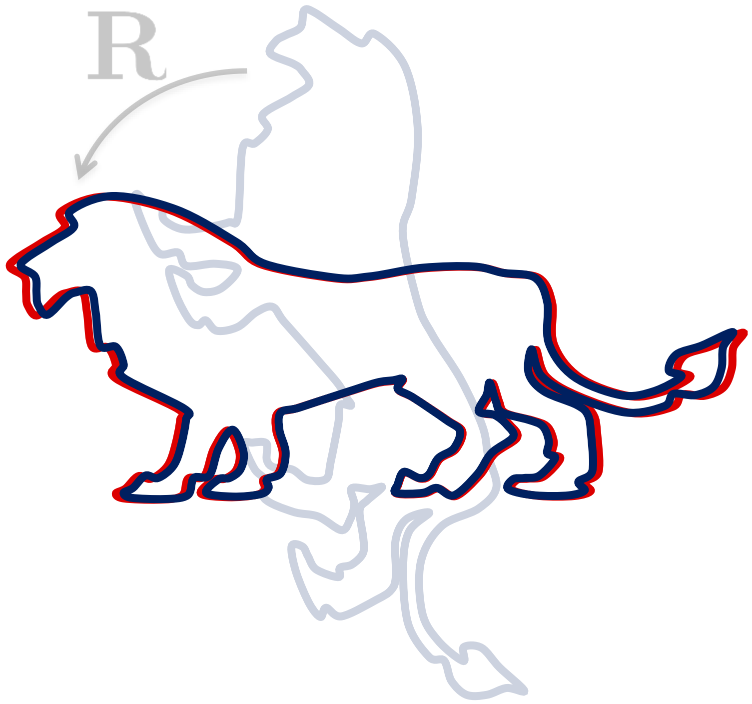
Shape Matching Problem



Shape Matching Problem



Shape Matching Problem



ETH zürich

Shape Matching Problem

- Align two point sets

$$\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \text{ and } \mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_n\}$$

- Find a translation vector $\mathbf{t}$ and rotation matrix $\mathbf{R}$ so that

$$\sum_{i=1}^{n} \|(\mathbf{R}\mathbf{p}_i + \mathbf{t}) - \mathbf{q}_i\|^2 \text{ is minimized}$$

Shape Matching – Solution

- Solve for translation first (w.r.t. $\mathbf{R}$, $\mathbf{p}$, and $\mathbf{q}$)

$$\frac{\partial}{\partial \mathbf{t}} \sum_{i=1}^{n} \|(\mathbf{R}\mathbf{p}_i + \mathbf{t}) - \mathbf{q}_i\|^2 = \sum_{i=1}^{n} 2\left((\mathbf{R}\mathbf{p}_i + \mathbf{t}) - \mathbf{q}_i\right) \stackrel{!}{=} 0$$

$$\mathbf{R}\sum_{i=1}^{n} \mathbf{p}_i + \sum_{i=1}^{n} \mathbf{t} - \sum_{i=1}^{n} \mathbf{q}_i = 0$$

$$\mathbf{t} = \underbrace{\left(\frac{1}{n}\sum_{i=1}^{n} \mathbf{q}_i\right)}_{\bar{\mathbf{q}}} - \mathbf{R}\underbrace{\left(\frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_i\right)}_{\bar{\mathbf{p}}}$$

Take a look at the Matrix Cookbook!

Point sets $\{\mathbf{q}_i\}$ and $\{\mathbf{R}\mathbf{p}_i\}$ have the same center of mass

ETH zürich

Finding the Rotation $\mathbf{R}$

- To find the optimal $\mathbf{R}$, we bring the centroids of both point sets to the origin

$$\mathbf{v}_i = \mathbf{p}_i - \bar{\mathbf{p}}, \qquad \mathbf{v}'_i = \mathbf{q}_i - \bar{\mathbf{q}}$$

- We want to find $\mathbf{R}$ that minimizes

$$\sum_{i=1}^{n} \|\mathbf{R}\mathbf{v}_i - \mathbf{v}'_i\|^2$$

Finding the Rotation $\mathbf{R}$

$$\sum_{i=1}^{n} \|\mathbf{R}\mathbf{v}_i - \mathbf{v}_i'\|^2 = \sum_{i=1}^{n} (\mathbf{R}\mathbf{v}_i - \mathbf{v}_i')^T (\mathbf{R}\mathbf{v}_i - \mathbf{v}_i') =$$

$$= \sum_{i=1}^{n} \Big(\mathbf{v}_i^T \underbrace{\mathbf{R}^T \mathbf{R}}_{\mathbf{I}} \mathbf{v}_i - \mathbf{v}_i'^T \mathbf{R}\mathbf{v}_i - \mathbf{v}_i^T \mathbf{R}^T \mathbf{v}_i' + \mathbf{v}_i'^T \mathbf{v}_i' \Big)$$

These terms do not depend on $\mathbf{R}$,
so we can ignore them in the minimization

Finding the Rotation $\mathbf{R}$

$$\underset{\mathbf{R}\in SO(3)}{\operatorname{argmin}} \sum_{i=1}^{n} \left(-\mathbf{v}_i'^T \mathbf{R} \mathbf{v}_i - \mathbf{v}_i^T \mathbf{R}^T \mathbf{v}_i' \right) = \underset{\mathbf{R}\in SO(3)}{\operatorname{argmax}} \sum_{i=1}^{n} \left(\mathbf{v}_i'^T \mathbf{R} \mathbf{v}_i + \underbrace{\mathbf{v}_i^T \mathbf{R}^T \mathbf{v}_i'} \right) =$$

$$= \boxed{\underset{\mathbf{R}\in SO(3)}{\operatorname{argmax}} \sum_{i=1}^{n} \mathbf{v}_i'^T \mathbf{R} \mathbf{v}_i}$$

$$\mathbf{v}_i^T \mathbf{R}^T \mathbf{v}_i' = \left(\mathbf{v}_i^T \mathbf{R}^T \mathbf{v}_i' \right)^T = \mathbf{v}_i'^T \mathbf{R} \mathbf{v}_i$$

Finding the Rotation $\mathbf{R}$

$$\sum_{i=1}^{n} \mathbf{v}_i'^{T} \mathbf{R} \mathbf{v}_i = tr\left(\mathbf{V}'^{T} \mathbf{R} \mathbf{V}\right)$$

$$\underbrace{\begin{bmatrix} \mathbf{v}_1'^{T} \\ \mathbf{v}_2'^{T} \\ \vdots \\ \mathbf{v}_n'^{T} \end{bmatrix}}_{\mathbf{V}'^{T}} \begin{bmatrix} \mathbf{R} \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix}}_{\mathbf{V}} = \begin{bmatrix} \mathbf{v}_1'^{T} \\ \mathbf{v}_2'^{T} \\ \vdots \\ \mathbf{v}_n'^{T} \end{bmatrix} \begin{bmatrix} \mathbf{R}\mathbf{v}_1 & \mathbf{R}\mathbf{v}_2 & \cdots & \mathbf{R}\mathbf{v}_n \end{bmatrix}$$

Finding the Rotation $\mathbf{R}$

$$\sum_{i=1}^{n} \mathbf{v}_i'^T \mathbf{R}\mathbf{v}_i = tr\left(\mathbf{V}'^T \mathbf{R}\mathbf{V}\right)$$

$$\begin{bmatrix} \mathbf{v}_1'^T \\ \mathbf{v}_2'^T \\ \vdots \\ \mathbf{v}_n'^T \end{bmatrix} \begin{bmatrix} \mathbf{R}\mathbf{v}_1 & \mathbf{R}\mathbf{v}_2 & \cdots & \mathbf{R}\mathbf{v}_n \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1'^T\mathbf{R}\mathbf{v}_1 & & & \\ & \mathbf{v}_2'^T\mathbf{R}\mathbf{v}_2 & & \\ & & \ddots & \\ & & & \mathbf{v}_n'^T\mathbf{R}\mathbf{v}_n \end{bmatrix}$$

ETH zürich

Finding the Rotation $\mathbf{R}$

- Find $\mathbf{R}$ that maximizes

Take a look at the Matrix Cookbook!

$$tr\left(\mathbf{V'}^T\mathbf{R}\mathbf{V}\right) = tr\left(\mathbf{R}\mathbf{V}\mathbf{V'}^T\right)$$

- SVD:

$$\mathbf{V}\mathbf{V'}^T = \mathbf{U}\boldsymbol{\Sigma}\tilde{\mathbf{U}}^T$$

$$tr\left(\mathbf{R}\mathbf{V}\mathbf{V'}^T\right) = tr\left(\underbrace{\mathbf{R}\,\mathbf{U}}\underbrace{\Sigma\tilde{\mathbf{U}}^T}\right) = tr\left(\Sigma\,\underbrace{\tilde{\mathbf{U}}^T\mathbf{R}\mathbf{U}}_{\text{orthogonal matrix}}\right)$$

ETH zürich

Finding the Rotation $\mathbf{R}$

- We want to maximize

$$tr\left(\Sigma\,\mathbf{M}\right)$$

$\mathbf{M}$: orthogonal matrix
all coeffs $\leq$ 1

$$\begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{bmatrix} \begin{bmatrix} m_{11} & \cdots & \\ \vdots & m_{22} & \vdots \\ & \cdots & m_{33} \end{bmatrix}$$

$$tr\left(\Sigma\,\mathbf{M}\right) = \sum_{i=1}^{3} \sigma_i\, m_{ii} \leq \sum_{i=1}^{3} \sigma_i$$

Finding the Rotation $\mathbf{R}$

$$tr\left(\Sigma\,\mathbf{M}\right) = \sum_{i=1}^{3} \sigma_i\, m_{ii} \leq \sum_{i=1}^{3} \sigma_i$$

- Our best shot is $m_{ii} = 1$, i.e. to make $\mathbf{M} = \mathbf{I}$

$$\begin{aligned} \mathbf{M} = \tilde{\mathbf{U}}^T \mathbf{R}\mathbf{U} &\overset{!}{=} \mathbf{I} \\ \mathbf{R}\mathbf{U} &= \tilde{\mathbf{U}} \\ \boxed{\mathbf{R} = \tilde{\mathbf{U}}\mathbf{U}^T} \end{aligned}$$

ETH zürich

Summary of Rigid Alignment

- Translate the input points to the centroids

$$\mathbf{v}_i = \mathbf{p}_i - \bar{\mathbf{p}}, \qquad \mathbf{v}'_i = \mathbf{q}_i - \bar{\mathbf{q}}$$

- Compute the "covariance matrix" $\mathbf{V}\mathbf{V}'^T$
- Compute its SVD:

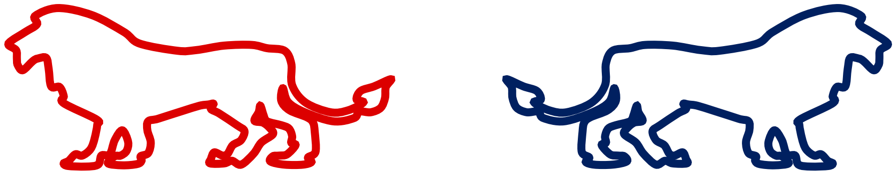
$$\mathbf{V}\mathbf{V}'^T = \mathbf{U}\boldsymbol{\Sigma}\tilde{\mathbf{U}}^T$$

- The optimal orthogonal $\mathbf{R}$ is

$$\mathbf{R} = \tilde{\mathbf{U}}\mathbf{U}^T$$

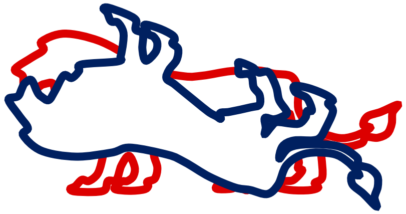
Sign Correction

- It is possible that $\det(\tilde{\mathbf{U}}\mathbf{U}^T) = -1$: sometimes reflection is the best orthogonal transform



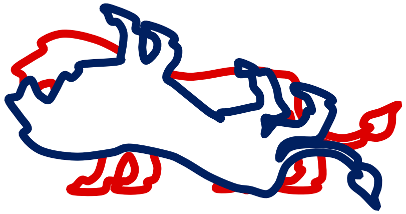
Sign Correction

- It is possible that $\det(\tilde{\mathbf{U}}\mathbf{U}^T) = -1$: sometimes reflection is the best orthogonal transform



Sign Correction

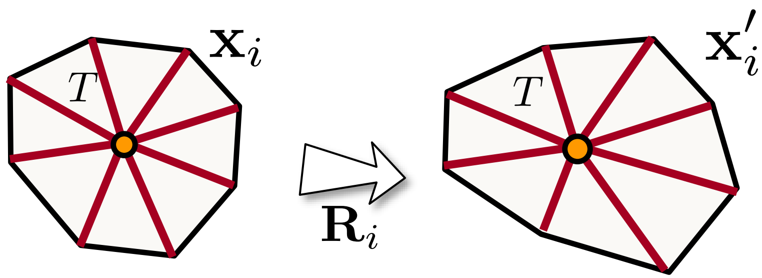
- To restrict ourselves to rotations only:
  take the last column of $\mathbf{U}$ (corresponding to the smallest singular value) and invert its sign.



- Why? See http://igl.ethz.ch/projects/ARAP/svd_rot.pdf

As-Rigid-As-Possible Deformation

- Optimal $\mathbf{R}_i$ is uniquely defined by $\mathbf{x}_i\ ,\mathbf{x}'_i$



$$\min \sum_{T \in \mathrm{Cell}_i} \sum_{(j,k) \in T} \|(\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}_i(\mathbf{x}_j - \mathbf{x}_k)\|^2$$

- so-called shape-matching problem, solved by a 3x3 SVD

$\mathbf{R}_i$ is a nonlinear function of $\mathbf{x}$

As-Rigid-As-Possible Deformation

- Total ARAP energy: sum up for all the cells i

$$\sum_{i} \sum_{T \in \mathrm{Cell}_i} \sum_{(j,k) \in T} \left\| (\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}_i (\mathbf{x}_j - \mathbf{x}_k) \right\|^2$$

- Treat $\mathbf{x}$ and $\mathbf{R}$ as separate sets of variables
- Simple **local-global** iterative optimization process
  - Decreases the energy at each step

As-Rigid-As-Possible Deformation

- Total ARAP energy: sum up for all the cells i

$$\sum_{i} \sum_{T \in \mathrm{Cell}_i} \sum_{(j,k) \in T} \left\| (\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}_i (\mathbf{x}_j - \mathbf{x}_k) \right\|^2$$

- Local step: keep $\mathbf{x}$' fixed, find optimal $\mathbf{R}_i$ per cell i
- Global step: keep $\mathbf{R}_i$ fixed, solve for $\mathbf{x}$' – quadratic minimization problem ➡ $\mathbf{L}\mathbf{x}' = \mathbf{b}$

As-Rigid-As-Possible Deformation

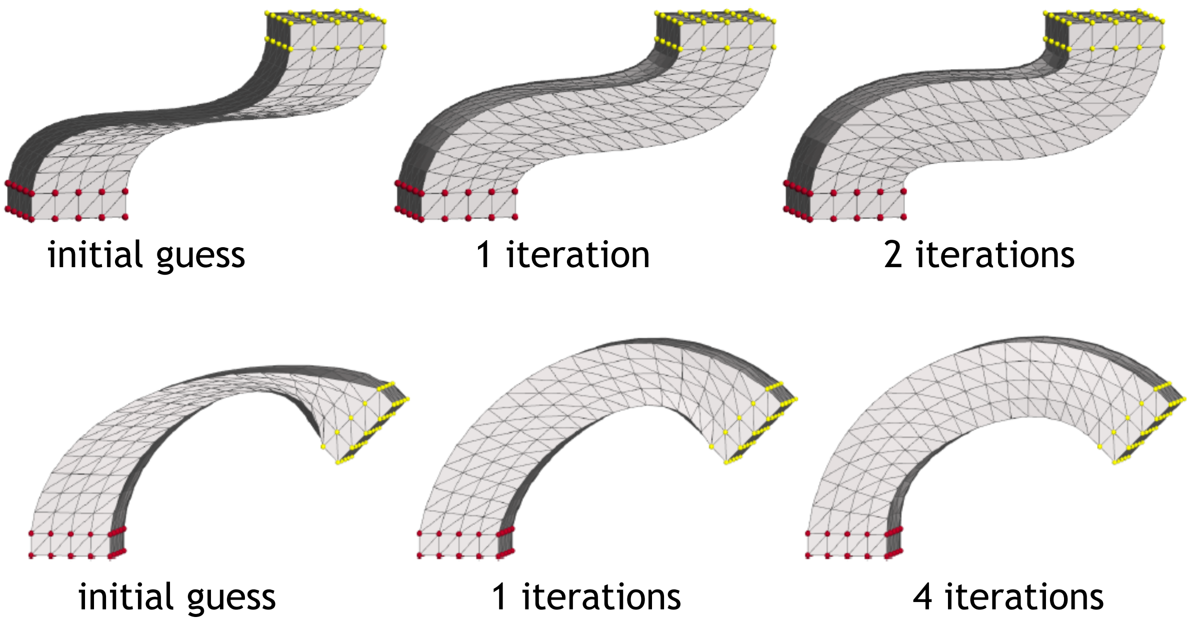
- Total ARAP energy: sum up for all the cells i

$$\sum_i \sum_{T \in \mathrm{Cell}_i} \sum_{(j,k) \in T} \left\| (\mathbf{x}'_j - \mathbf{x}'_k) - \mathbf{R}_i (\mathbf{x}_j - \mathbf{x}_k) \right\|^2$$

- Local step: keep $\mathbf{x}'$ fixed, find optimal $\mathbf{R}_i$ per cell i
- Global step: keep $\mathbf{R}_i$ fixed, solve for $\mathbf{x}'$ – quadratic minimization problem ➡ $\mathbf{L}\mathbf{x}' = \mathbf{b}$
  - The matrix $\mathbf{L}$ stays fixed, can pre-factorize

Initial Guess

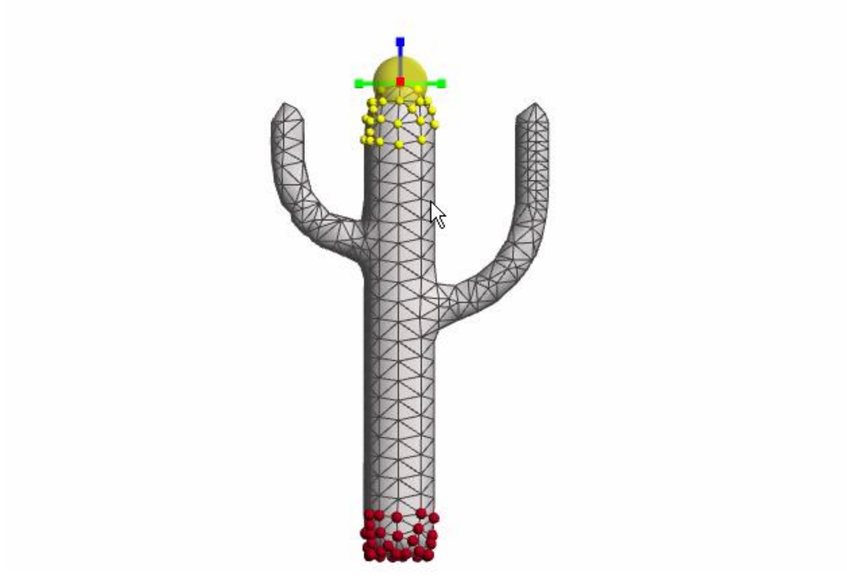
- Can use naïve Laplacian editing



ETH zürich

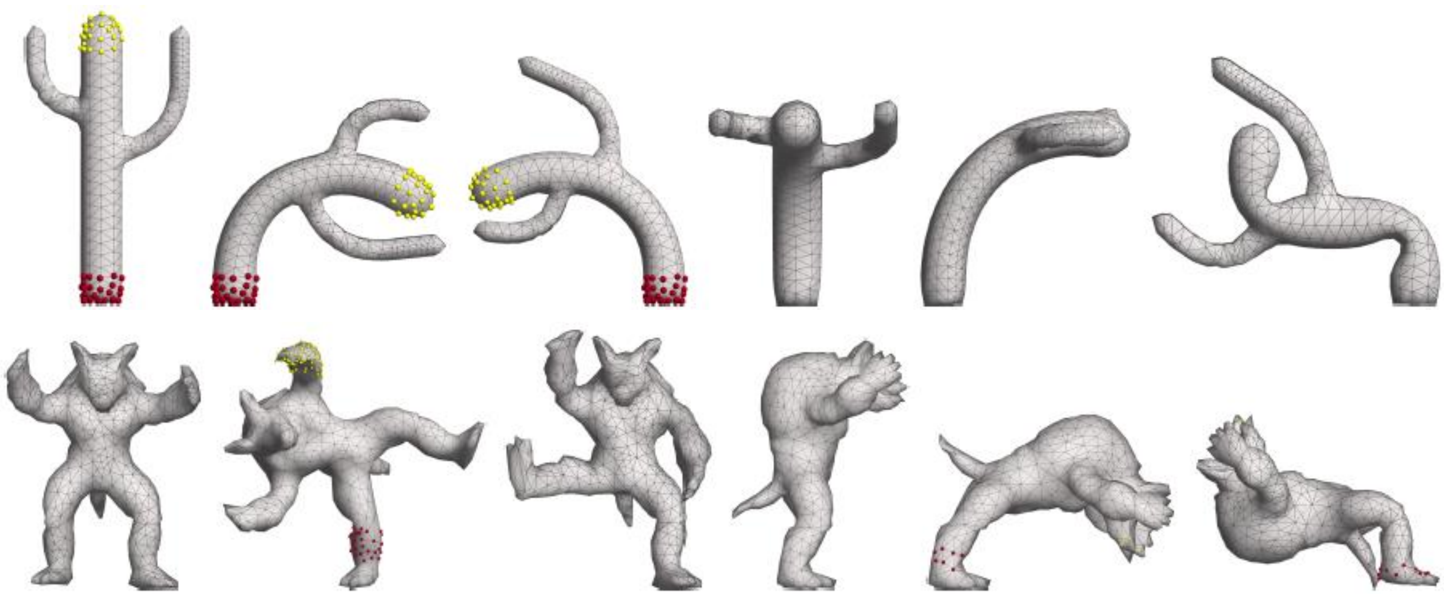
Initial Guess

- Can also use the previous frame
- Replace all handle vertex positions by the currently prescribed ones
- Fast convergence

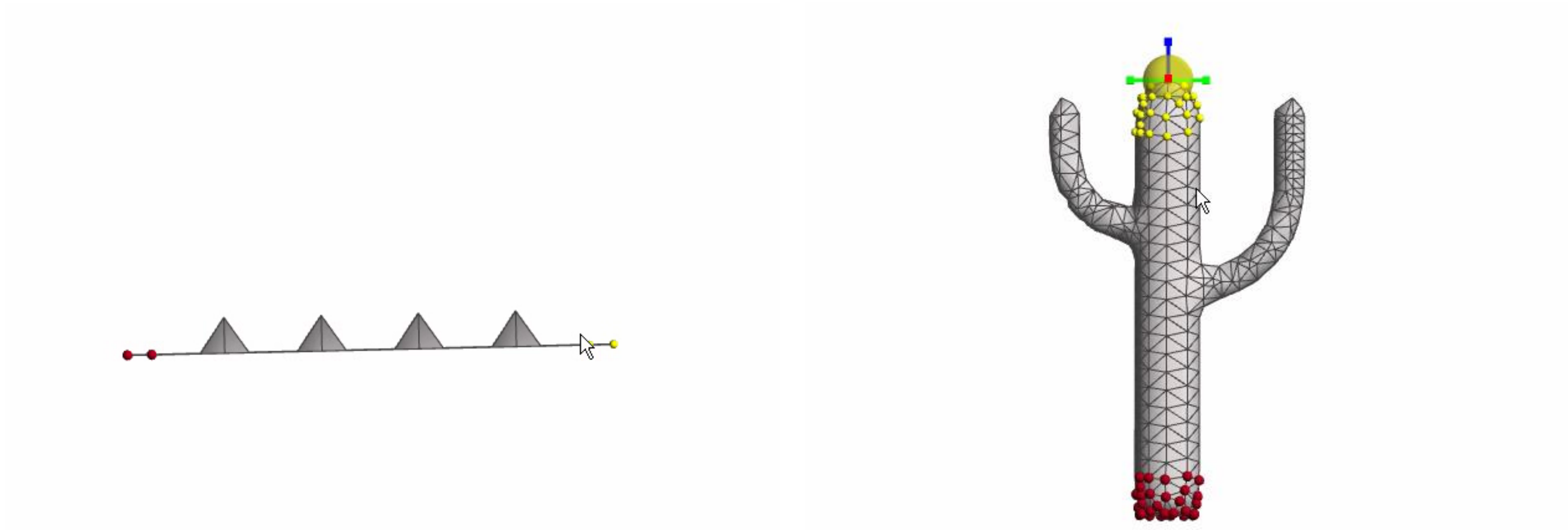


Large Rotations

- Use previous frame as the initial guess



Examples



Discussion

- Nonlinear deformation that models a kind of elastic behavior
- Very simple to implement, no parameters to tune except number of iterations
- Each step is guaranteed to not increase the energy
  - Compare with Gauss-Newton…
- Each iteration is relatively cheap, no matrix re-factorization necessary

Discussion

- Works fine on small meshes
- On larger meshes: slow convergence
  - Each iteration is more expensive
  - Need more iterations because the conditioning of the system becomes worse as the matrix grows
- Material stiffness depends on the cell size
  - lots of wrinkles for fine meshes when using 1-rings as cells

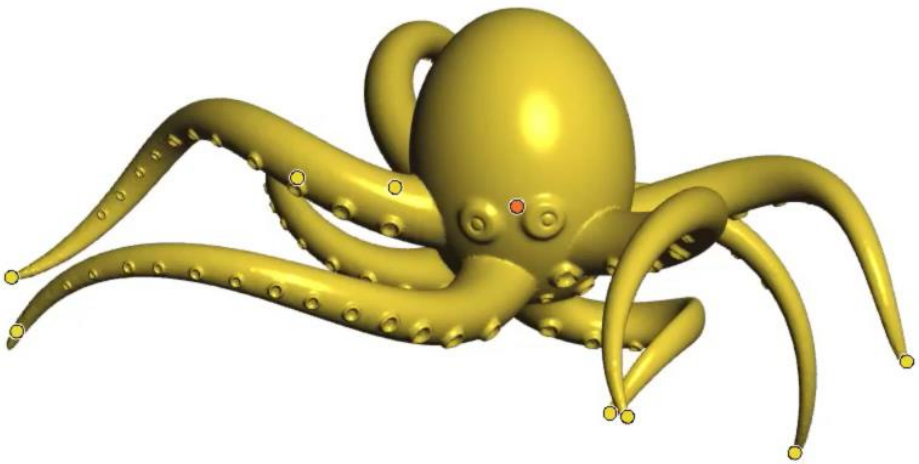
Acceleration using Subspace Techniques

- Subspace created by influence weight functions for each handle
- Drastically reduces the number of degrees of freedom in the optimization

Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. [“Fast Automatic Skinning Transformations,” 2012.]

Acceleration using Subspace Techniques

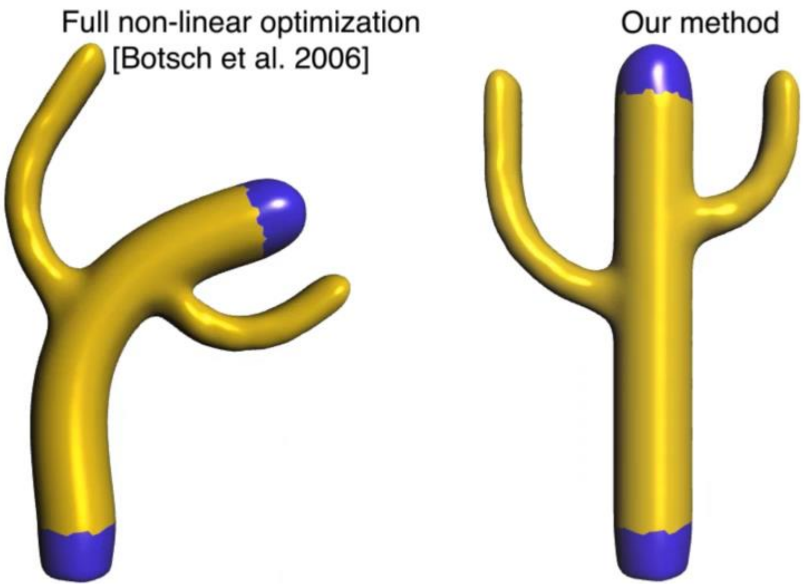
- Subspace created by influence weight functions for each handle
- Drastically reduces the number of degrees of freedom in the optimization



Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. [“Fast Automatic Skinning Transformations,” 2012.]

Acceleration using Subspace Techniques

- Subspace created by influence weight functions for each handle
- Drastically reduces the number of degrees of freedom in the optimization



Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. [“Fast Automatic Skinning Transformations,” 2012.]()

Demo

- Libigl demo 406