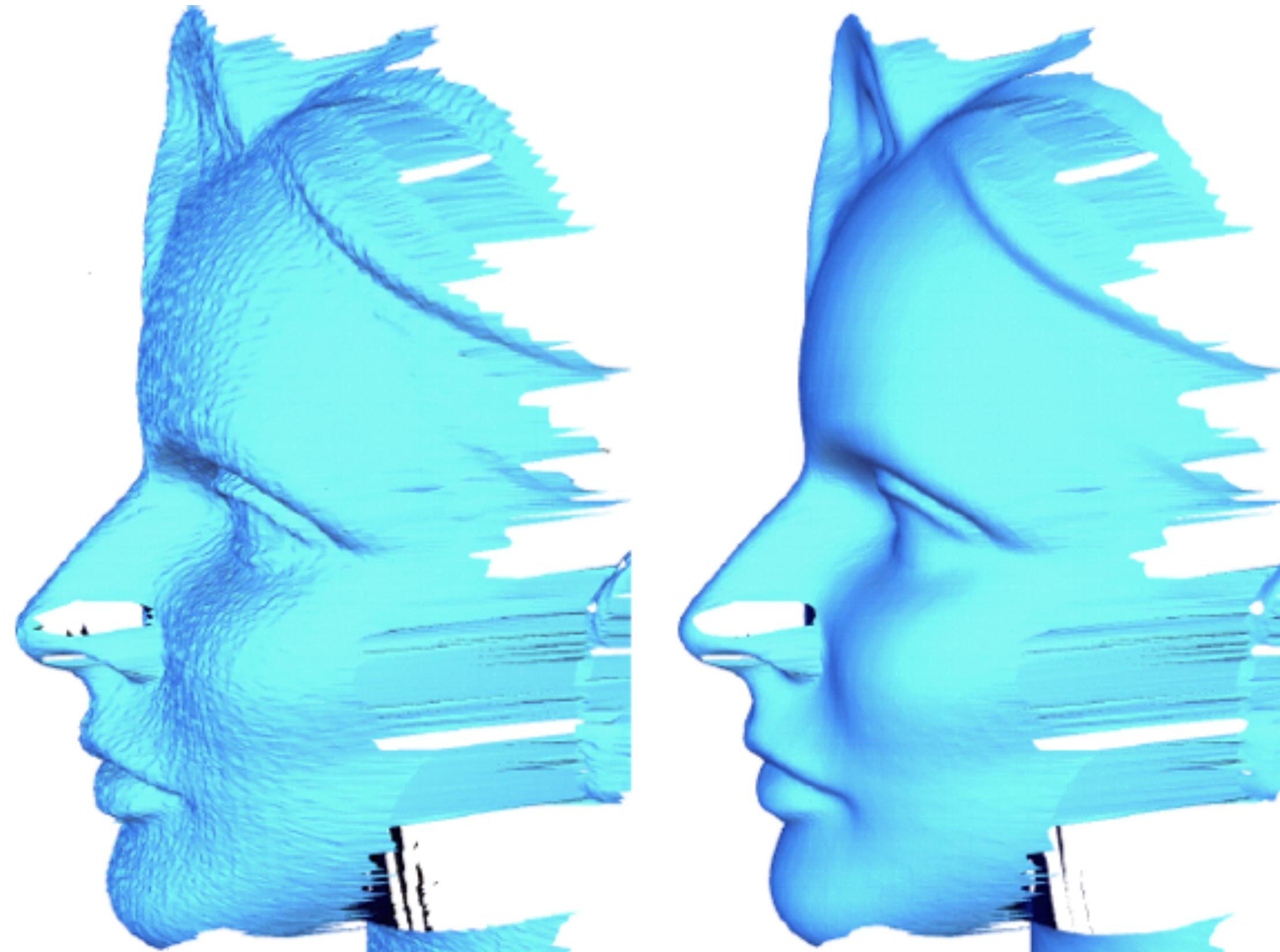


Smoothing

Acknowledgements: Olga Sorkine-Hornung and Mario Botsch
CSC 486B/586B - Geometric Modeling - Teseo Schneider

Motivation

- Scanned surfaces can be noisy



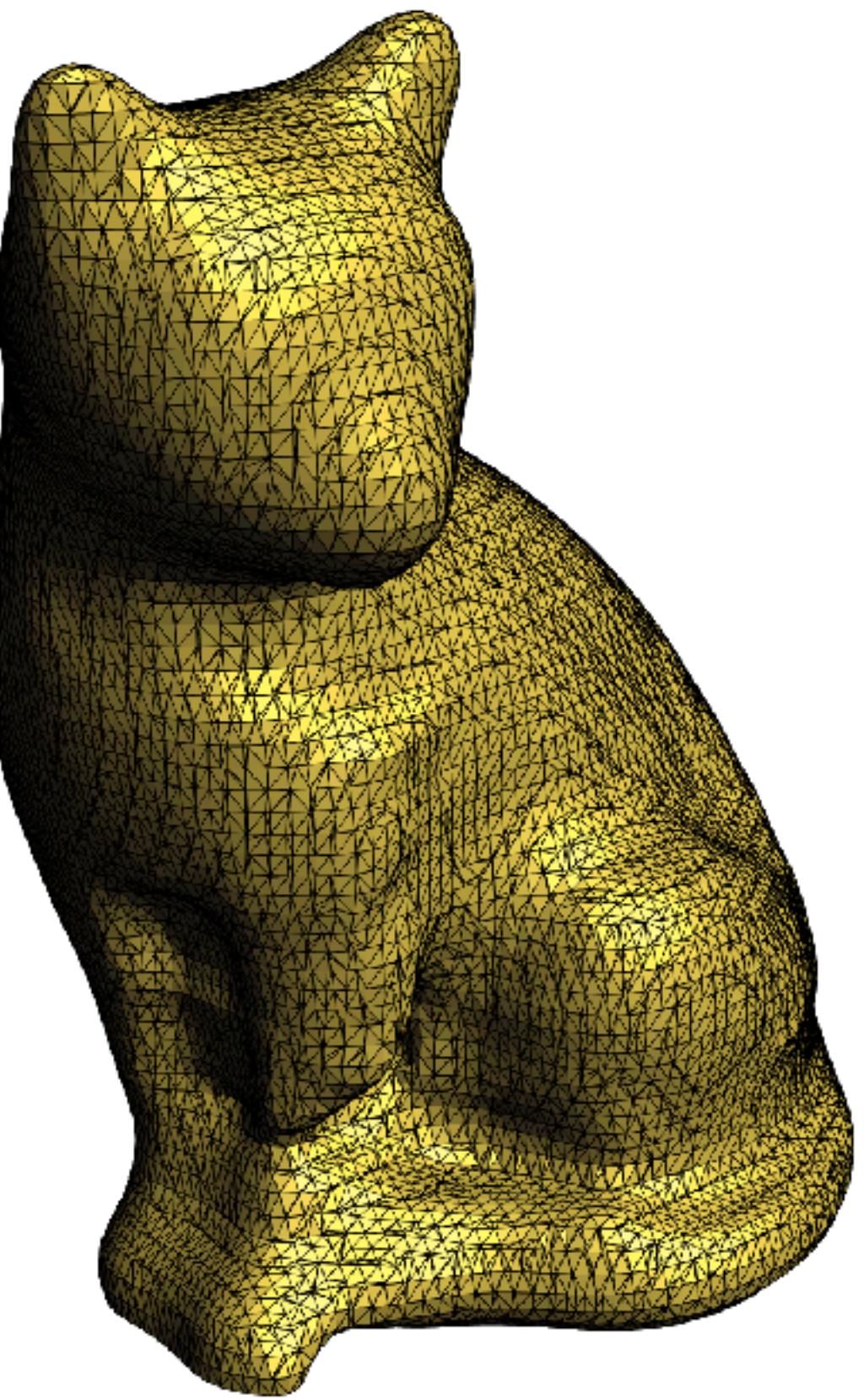
Motivation

- Scanned surfaces can be noisy

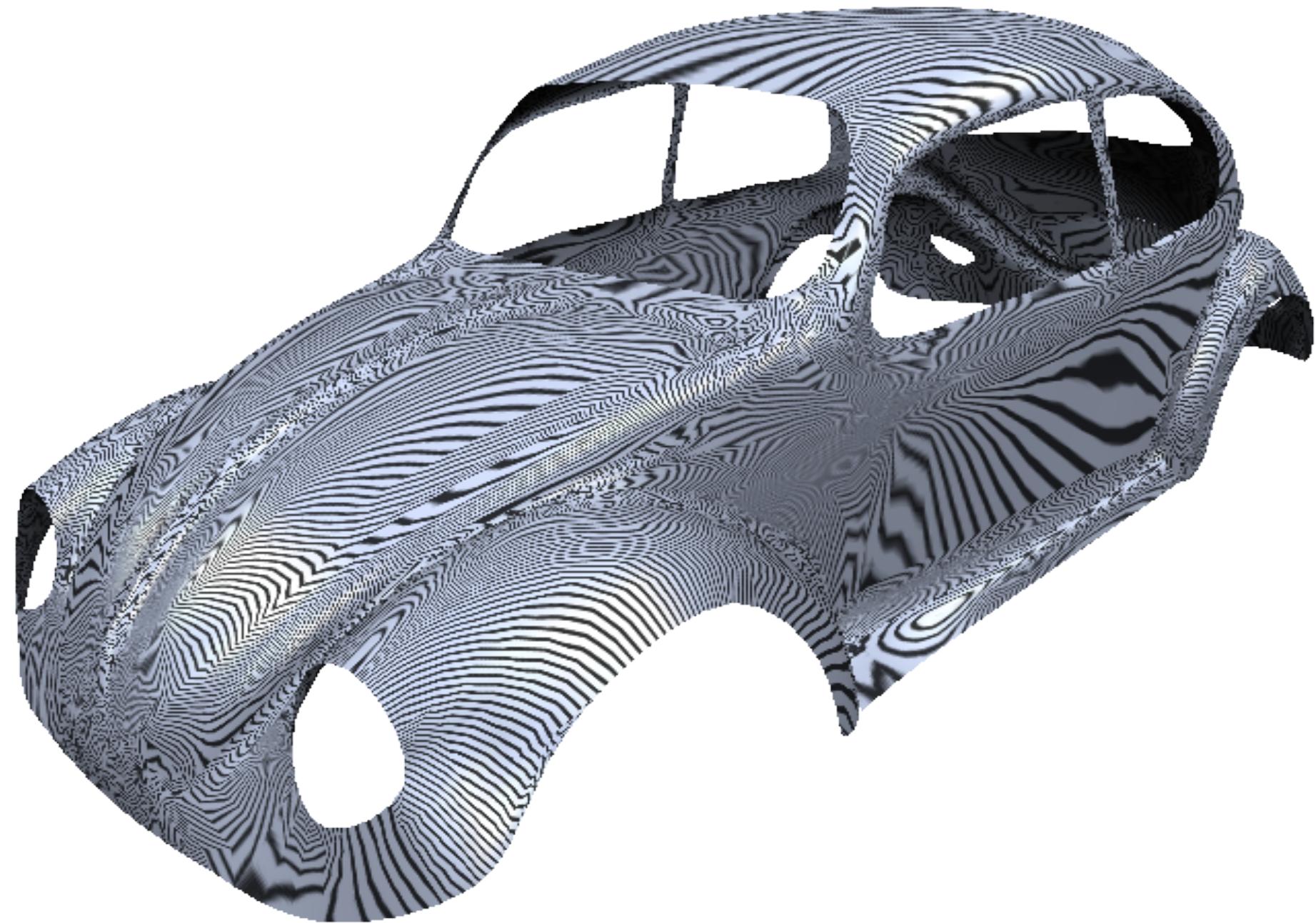
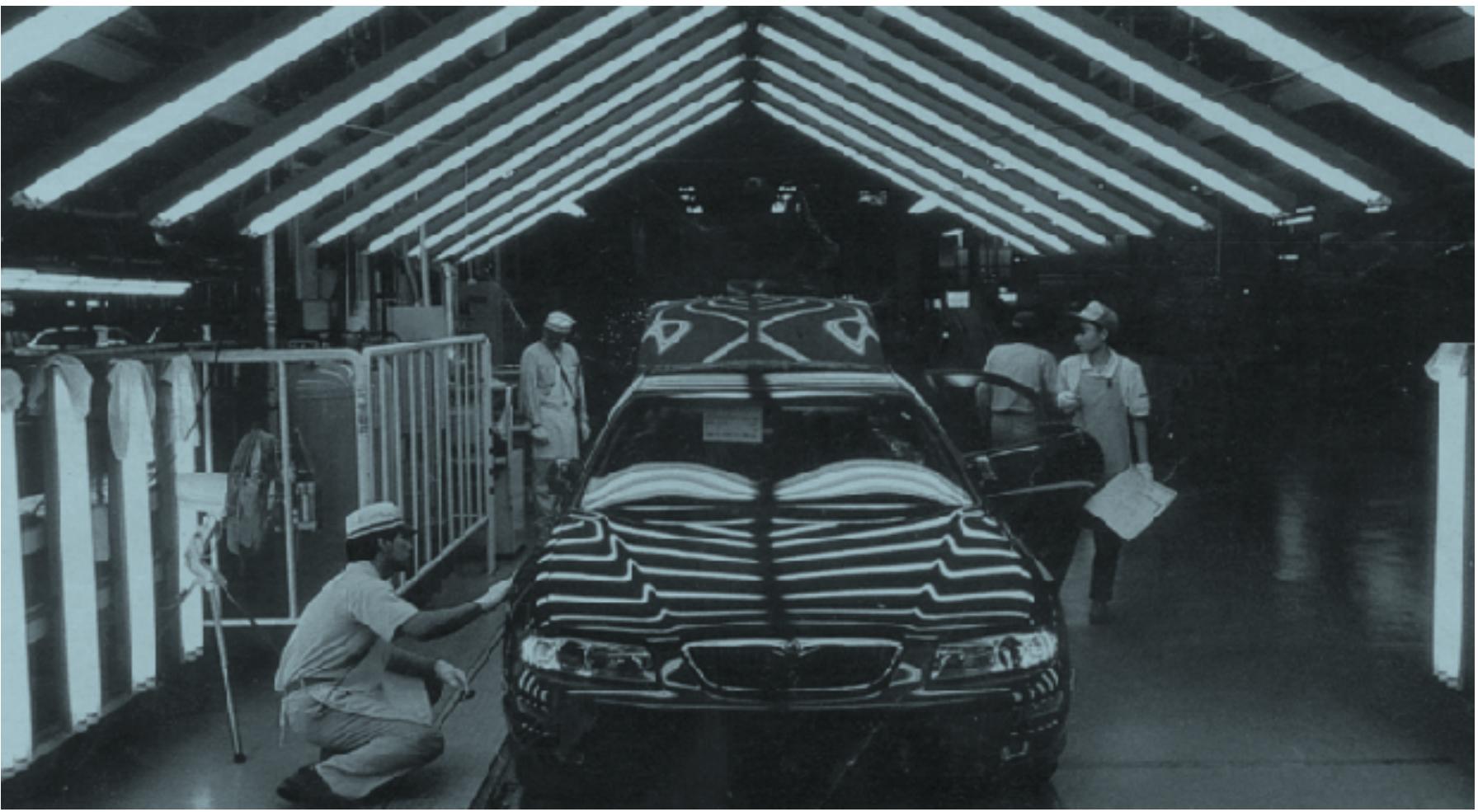


Motivation

- Marching Tetrahedra/Cubes meshes can be problematic



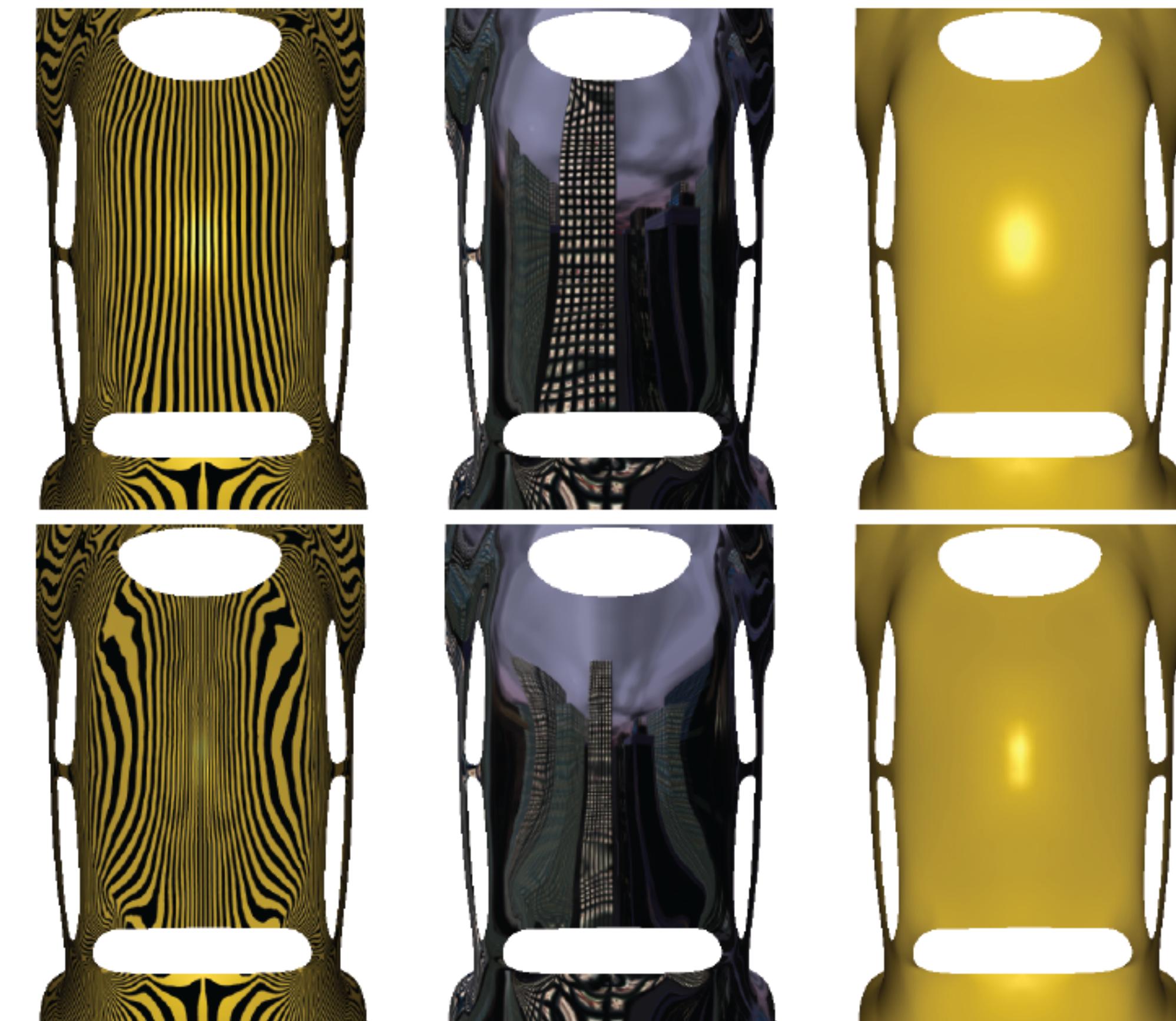
Visual Assessment of Surface Smoothness



Reflection Lines as an Inspection Tool

- Shape optimization using reflection lines

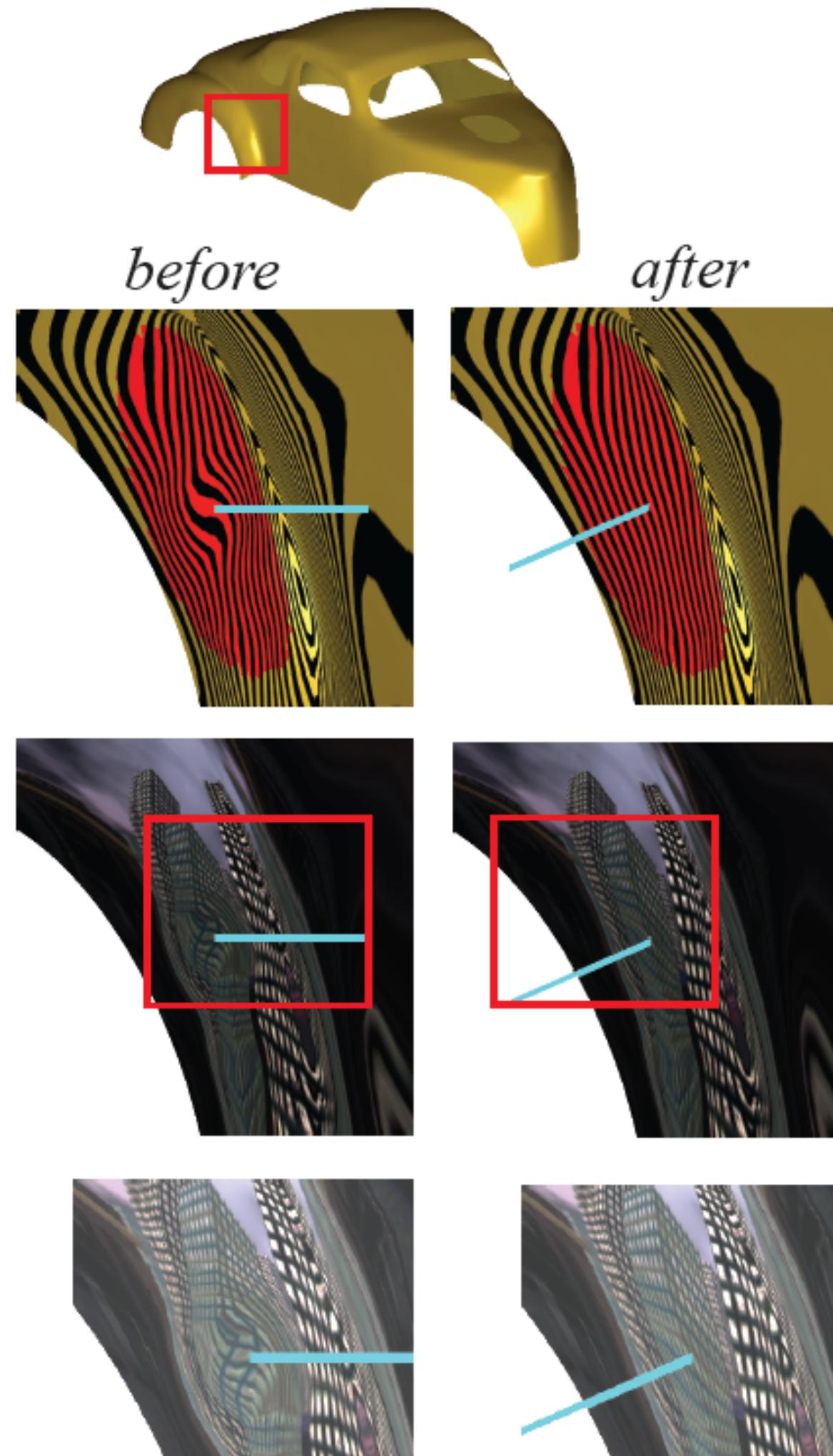
*E. Tosun, Y. I. Gingold, J.
Reisman, D. Zorin
Symposium on Geometry
Processing 2007*



Reflection Lines as an Inspection Tool

- Shape optimization using reflection lines

*E. Tosun, Y. I. Gingold, J.
Reisman, D. Zorin
Symposium on Geometry
Processing 2007*



How to measure smoothness?

Curvature and Smoothness

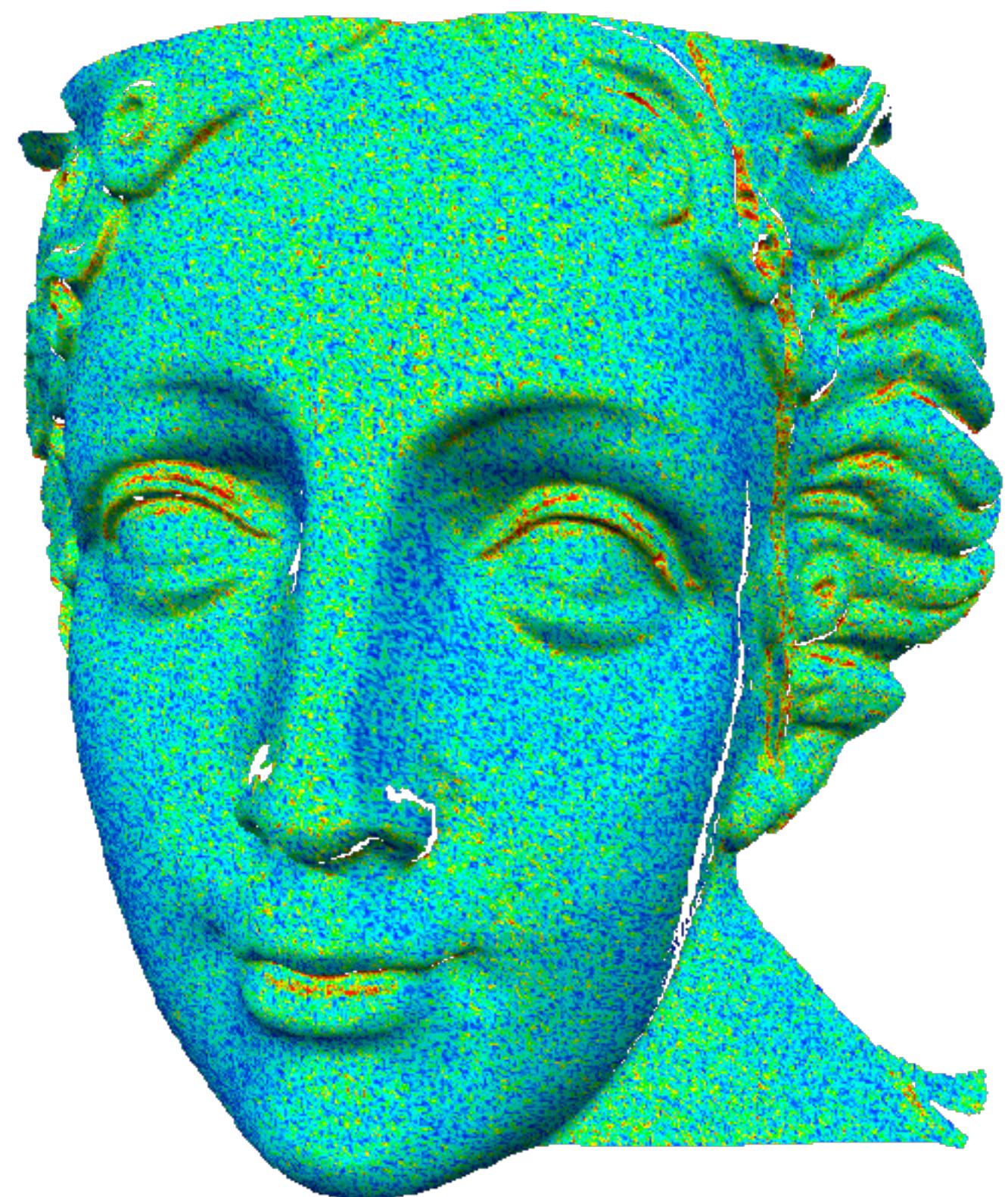


Curvature and Smoothness

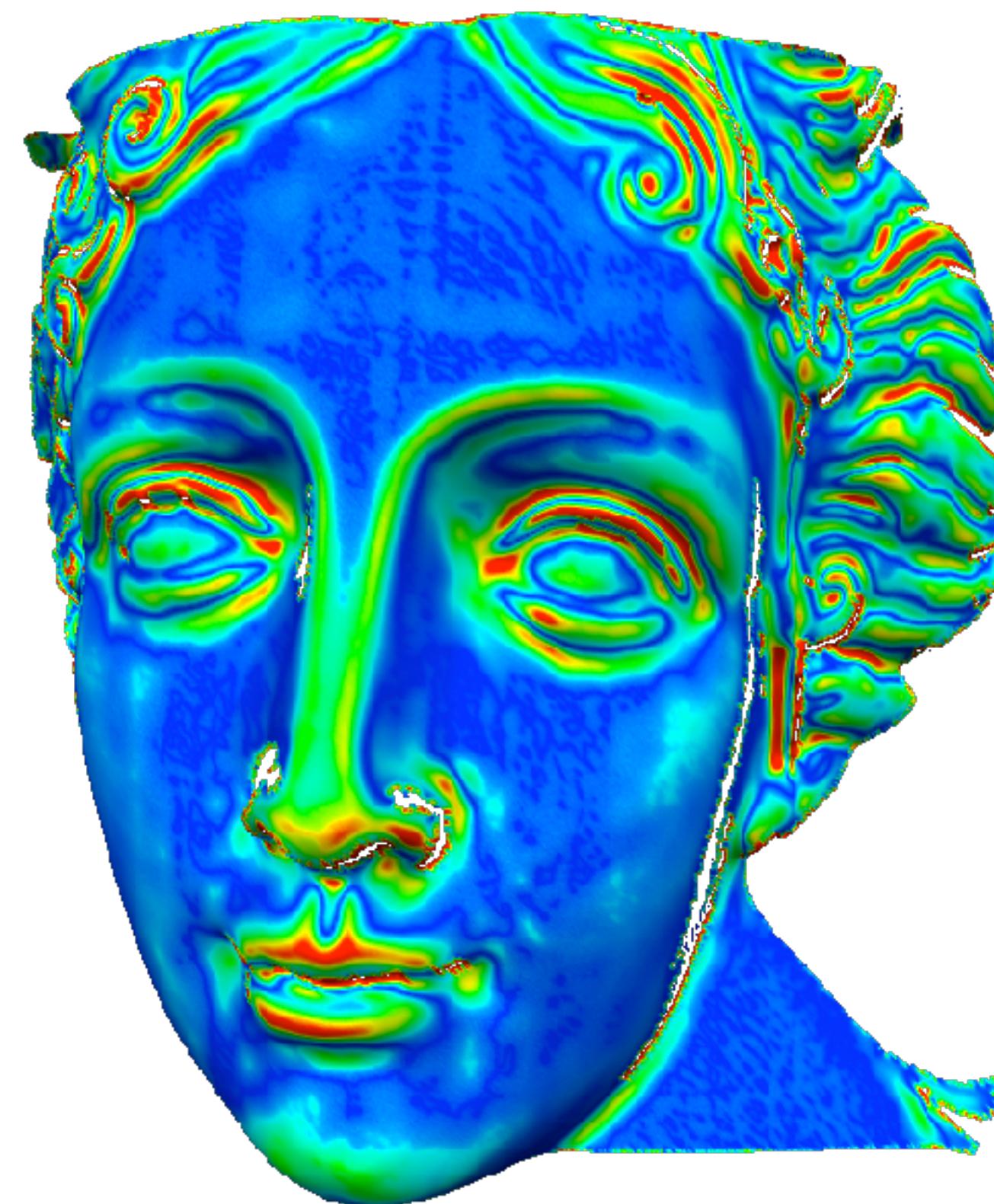


mean curvature plot

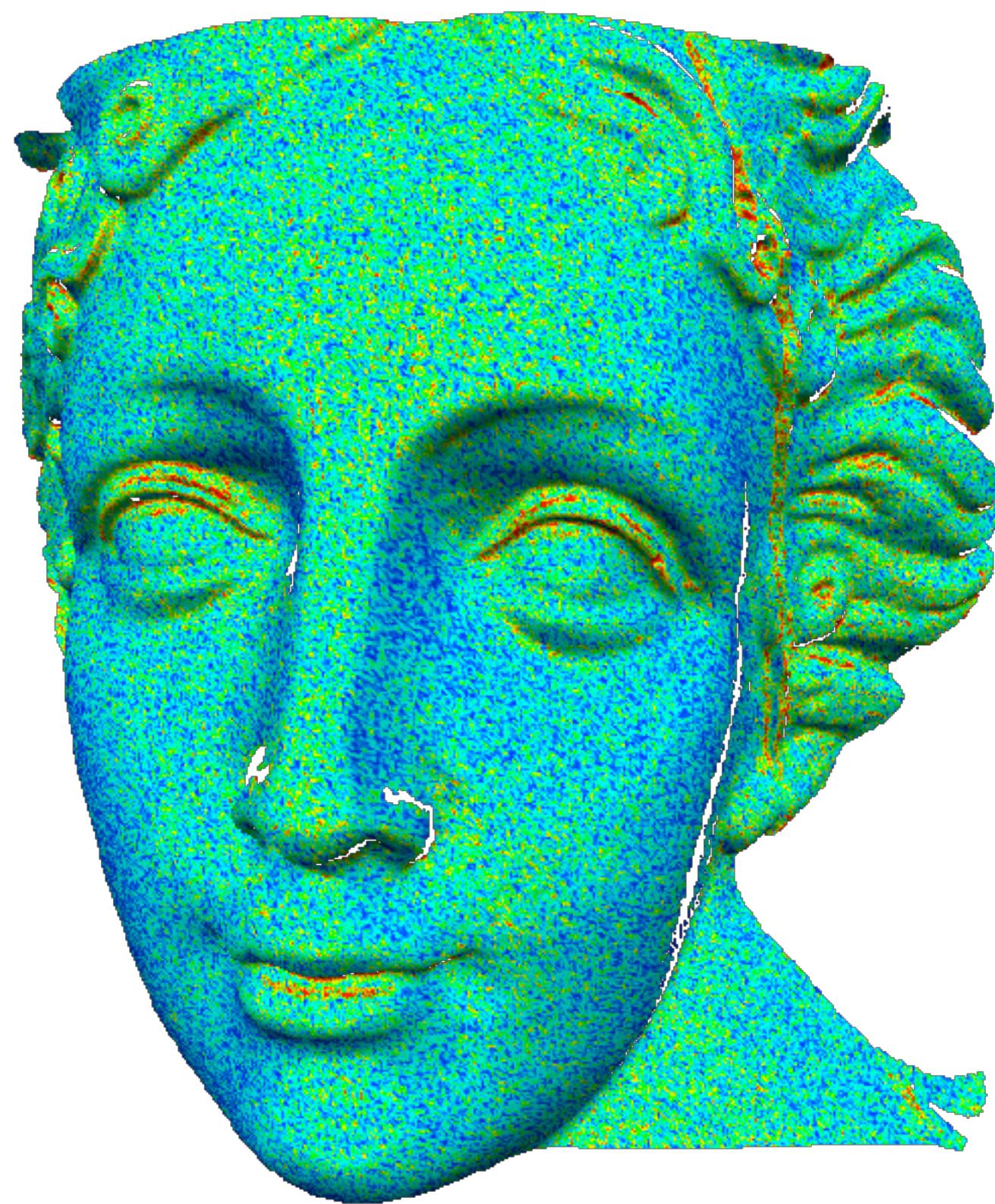
Curvature and Smoothness



mean curvature plot



Curvature and Smoothness

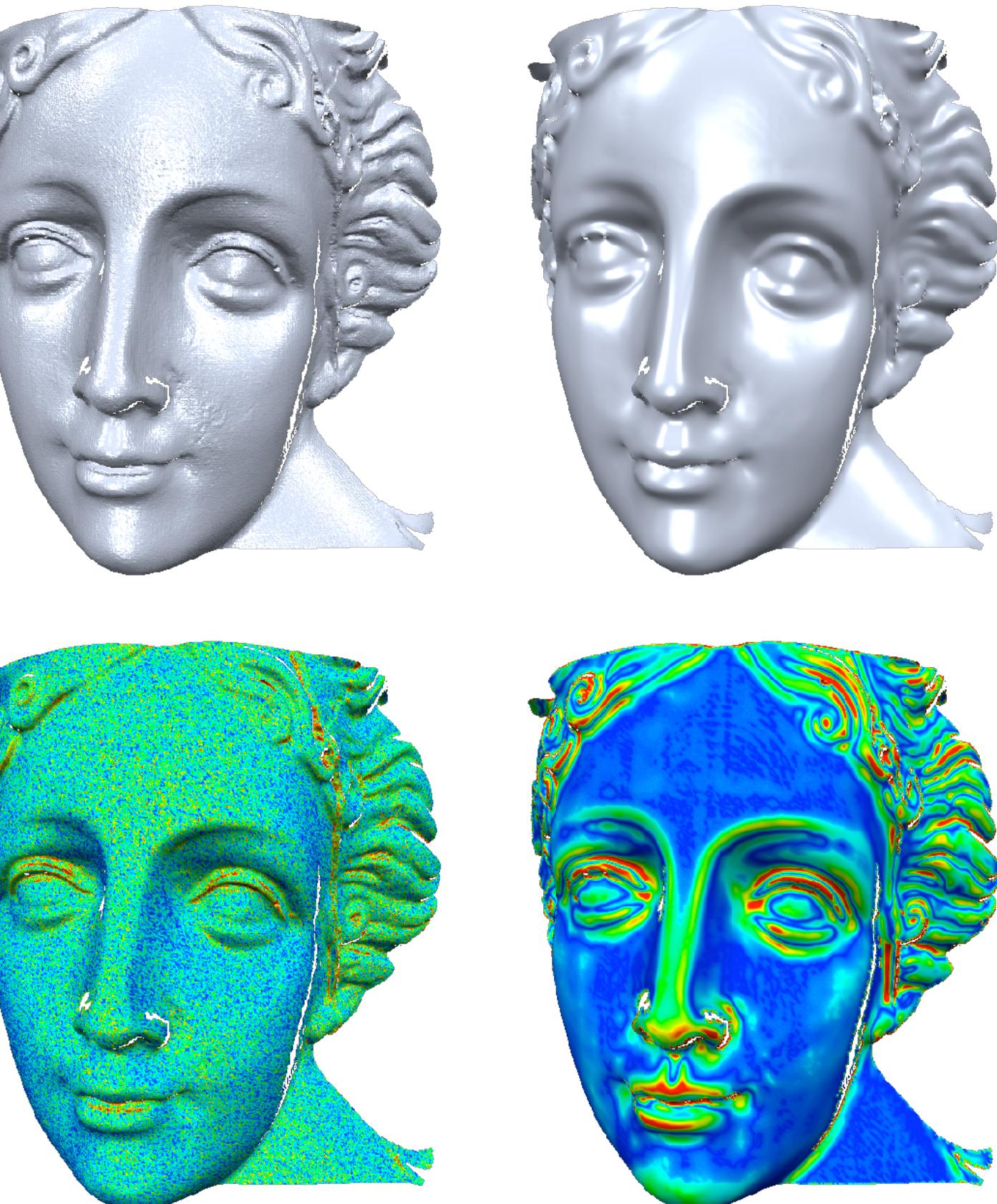


mean curvature plot



Curvature and Smoothness

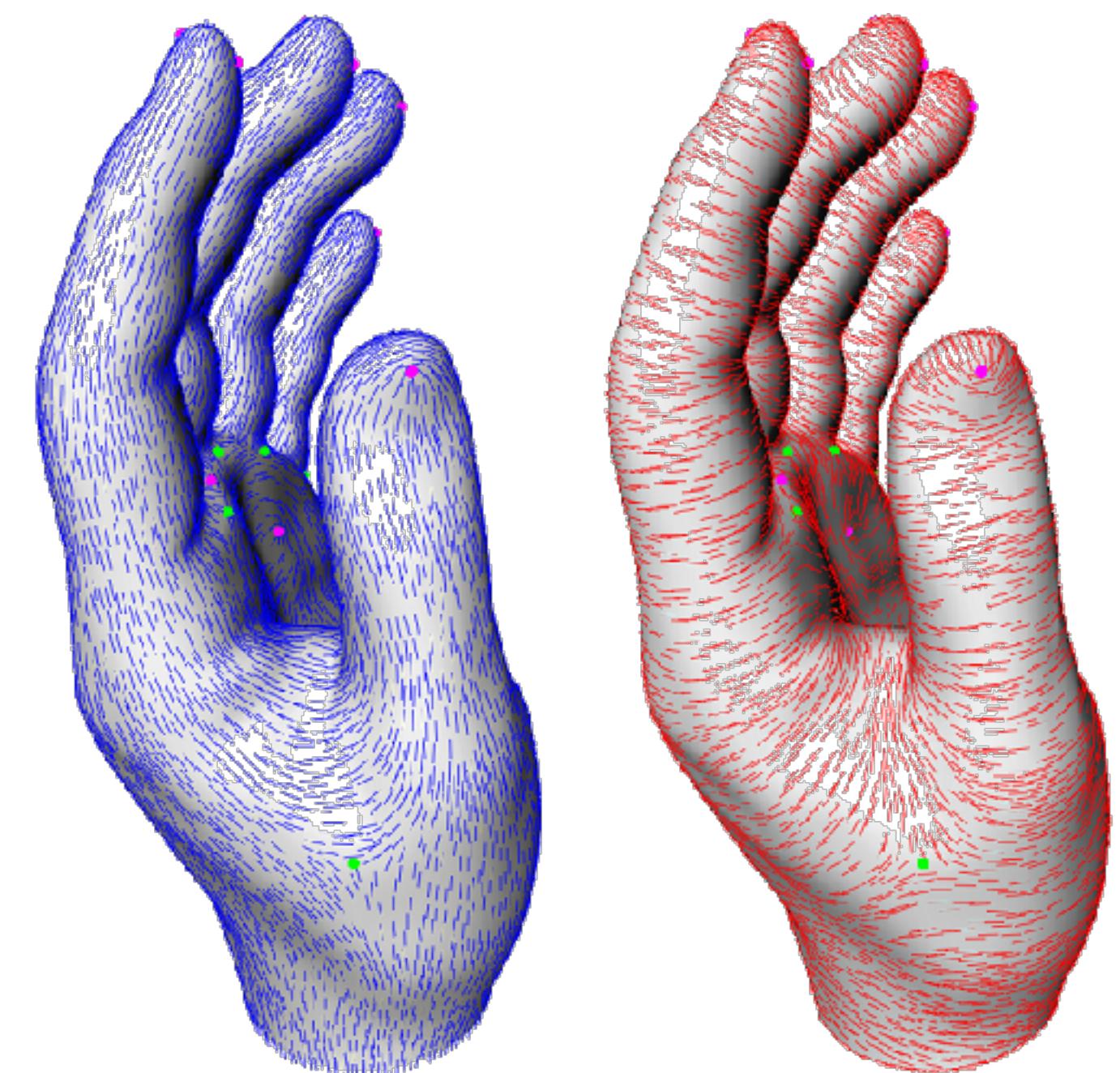
- Smoothing = reducing curvature?
- Smoothing = make curvature vary less?



Which curvature?

- Principal curvatures
 - Nonlinear and “discontinuous” operator in the definition (min, max)

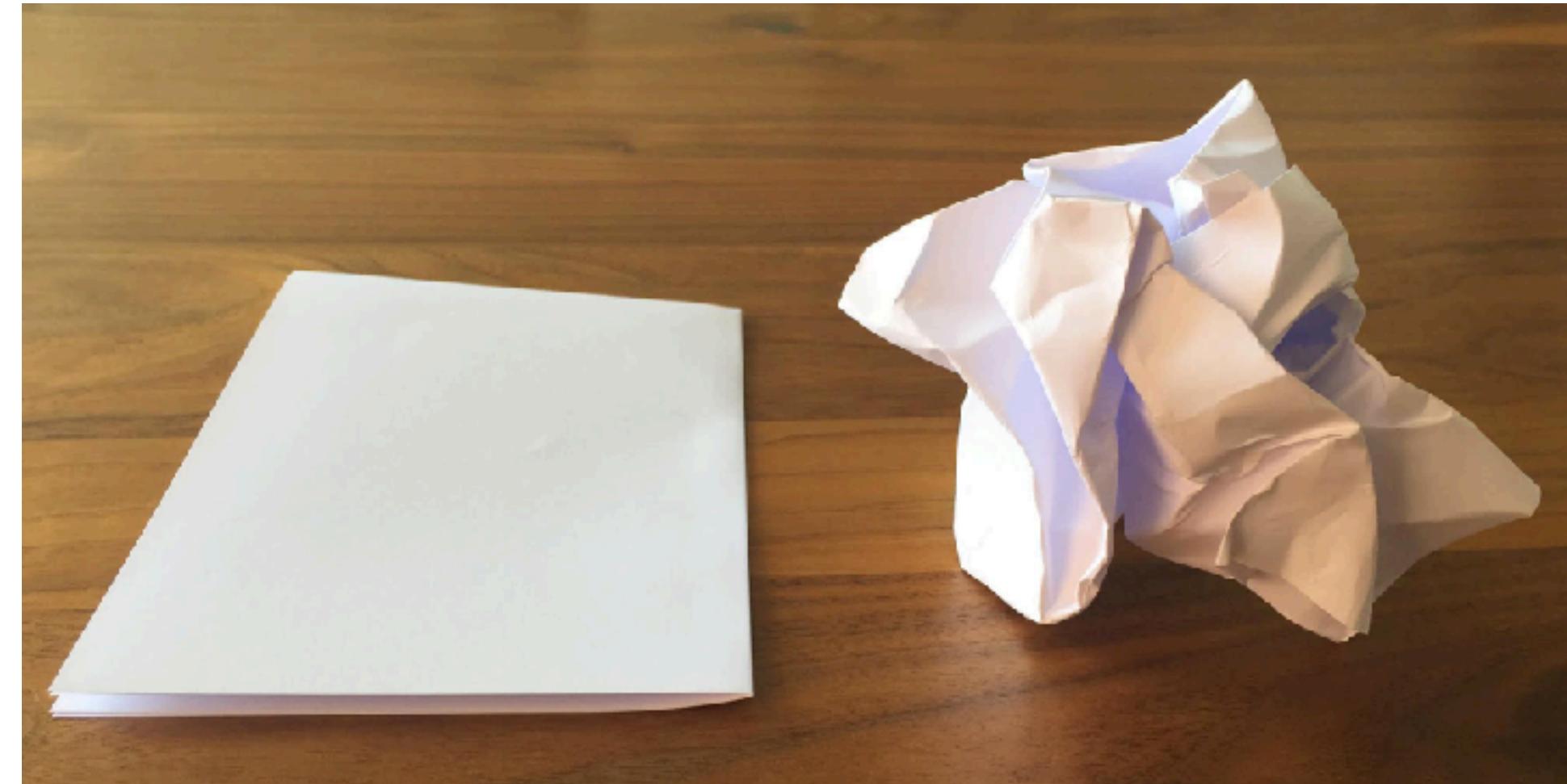
$\kappa_{\min}, \kappa_{\max}$



principal directions

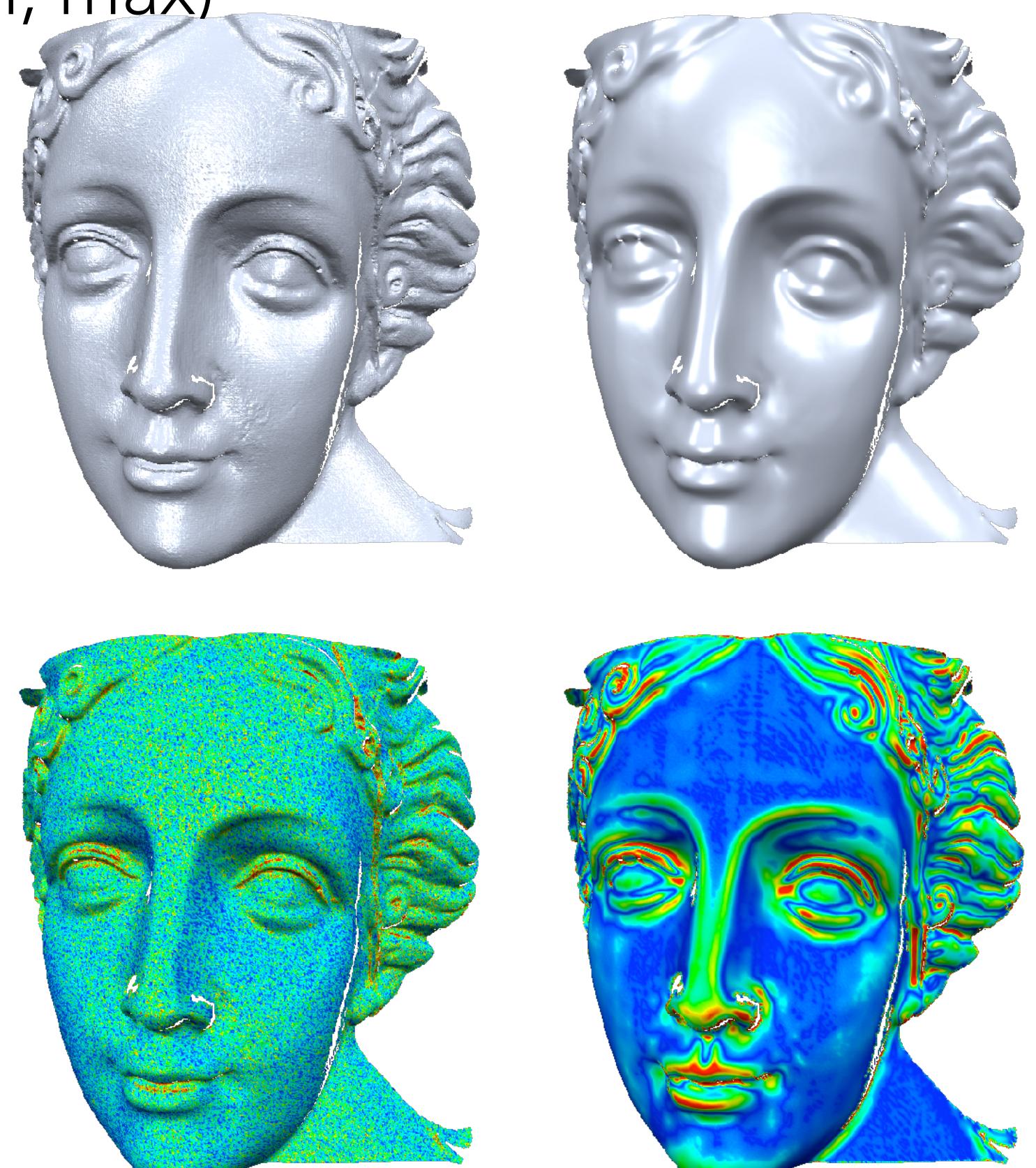
Which curvature?

- Principal curvatures
 - Nonlinear and “discontinuous” operator in the definition (min, max)
- Gauss curvature K
 - Intrinsic-only, insensitive to embedding in



Which curvature?

- Principal curvatures
 - Nonlinear and “discontinuous” operator in the definition (min, max)
- Gauss curvature K
 - Intrinsic-only, insensitive to embedding in
- Mean curvature H
 - Relatively simple to extract on meshes via Laplace-Beltrami:



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

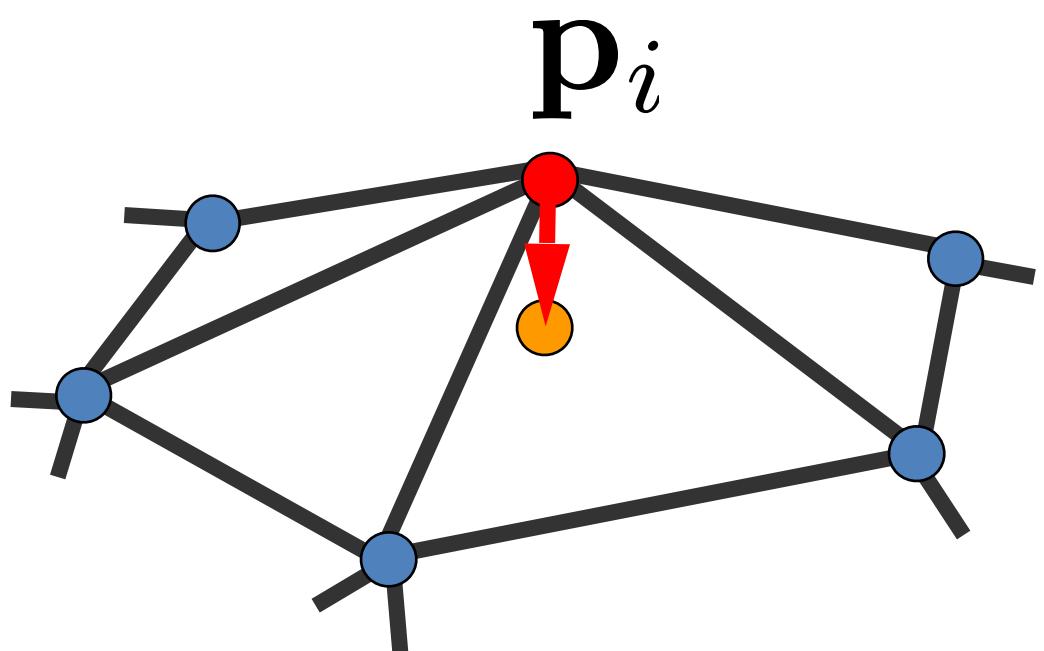
goal: $H = 0$ or $H = \text{const}$

Laplace as a Linear Operator

Recap: Laplace-Beltrami

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

- High-pass filter: extracts local surface detail
 - Detail = *smooth*(surface) – surface
 - Assumption: smoothing = averaging

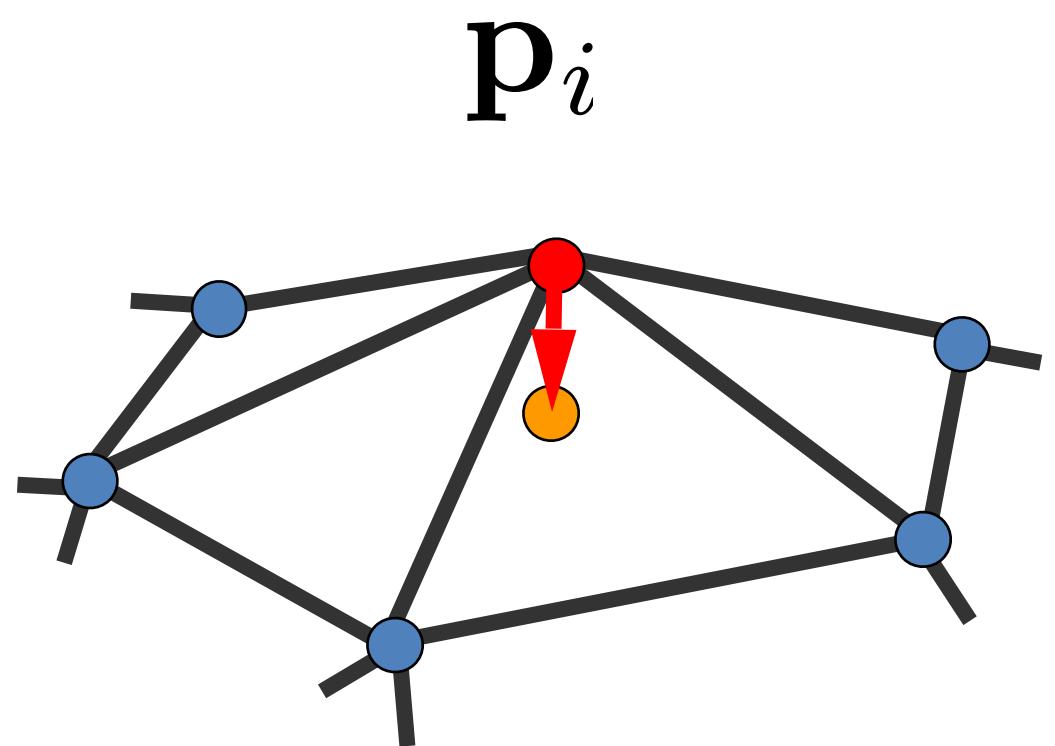


$$\Delta_{\mathcal{M}}(\mathbf{p}_i) = \delta_i = \frac{1}{W_i} \sum_{j \in \mathcal{N}(i)} w_{ij} (\mathbf{p}_j - \mathbf{p}_i)$$

Recap: Laplace-Beltrami

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

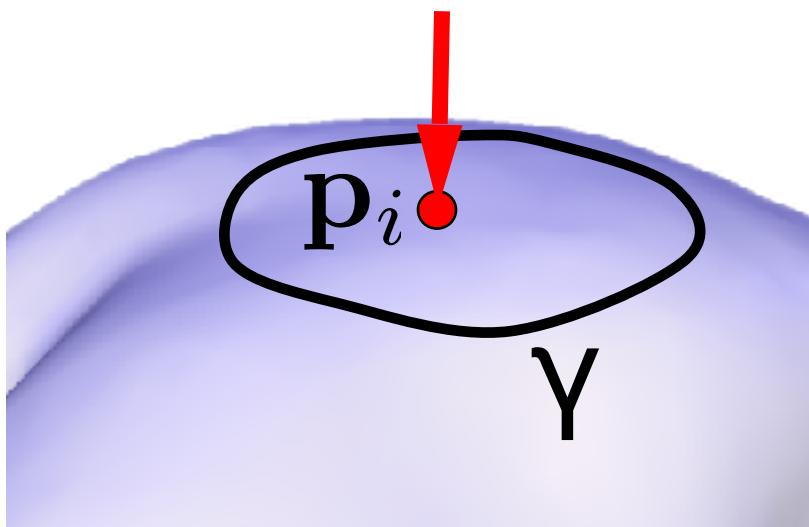
- The direction of δ_i approximates the normal
- The size approximates the mean curvature



$$\delta_i = \frac{1}{W_i} \sum_{j \in \mathcal{N}(i)} w_{ij} (\mathbf{p}_j - \mathbf{p}_i)$$

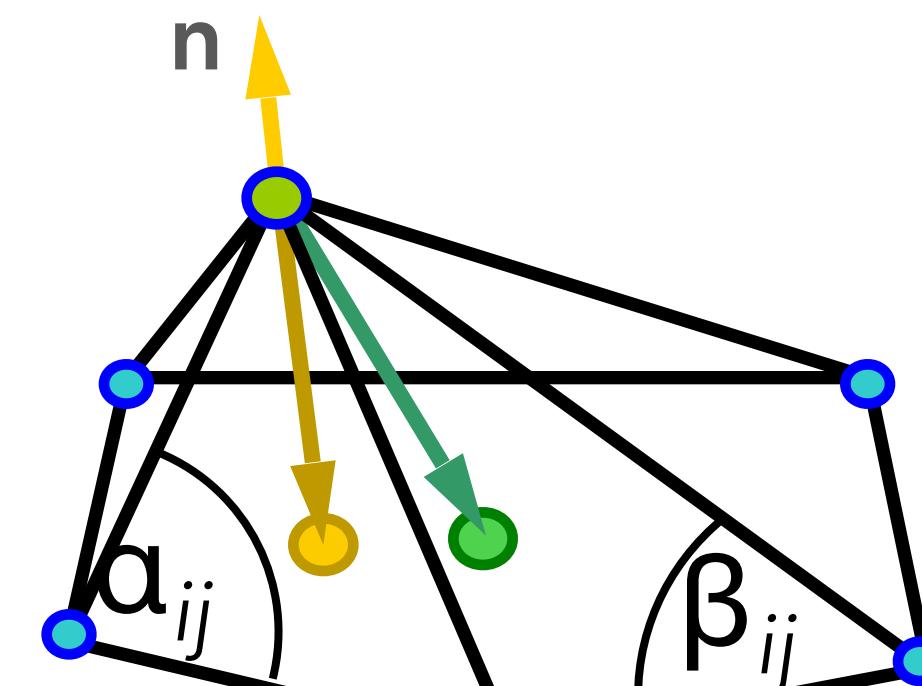
L-B: Weighting Schemes

- Ignore geometry $\delta_i = \frac{1}{W_i} \sum_{j \in \mathcal{N}(i)} w_{ij} (\mathbf{p}_j - \mathbf{p}_i)$
- Integrate over the Voronoi region of the vertex



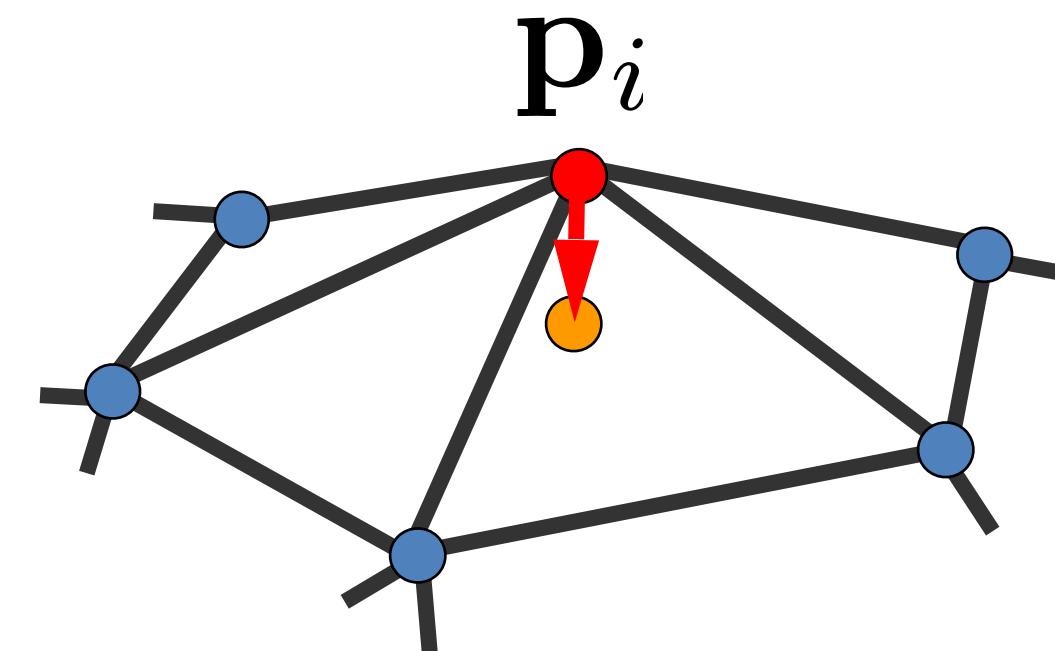
$\delta_{\text{uniform}} : W_i = 1, w_{ij} = 1/|N(i)|$

$\delta_{\text{cotan}} : w_{ij} = 0.5(\cot \alpha_{ij} + \cot \beta_{ij})$



Laplacian Matrix

- The transition between xyz and δ is linear:



$$\delta_i = \frac{1}{W_i} \sum_{j \in \mathcal{N}(i)} w_{ij} (\mathbf{p}_j - \mathbf{p}_i)$$

$$\begin{array}{c} \mathbf{L} \quad \mathbf{x} = \delta_x \\ \mathbf{L} \quad \mathbf{y} = \delta_y \\ \mathbf{L} \quad \mathbf{z} = \delta_z \end{array}$$

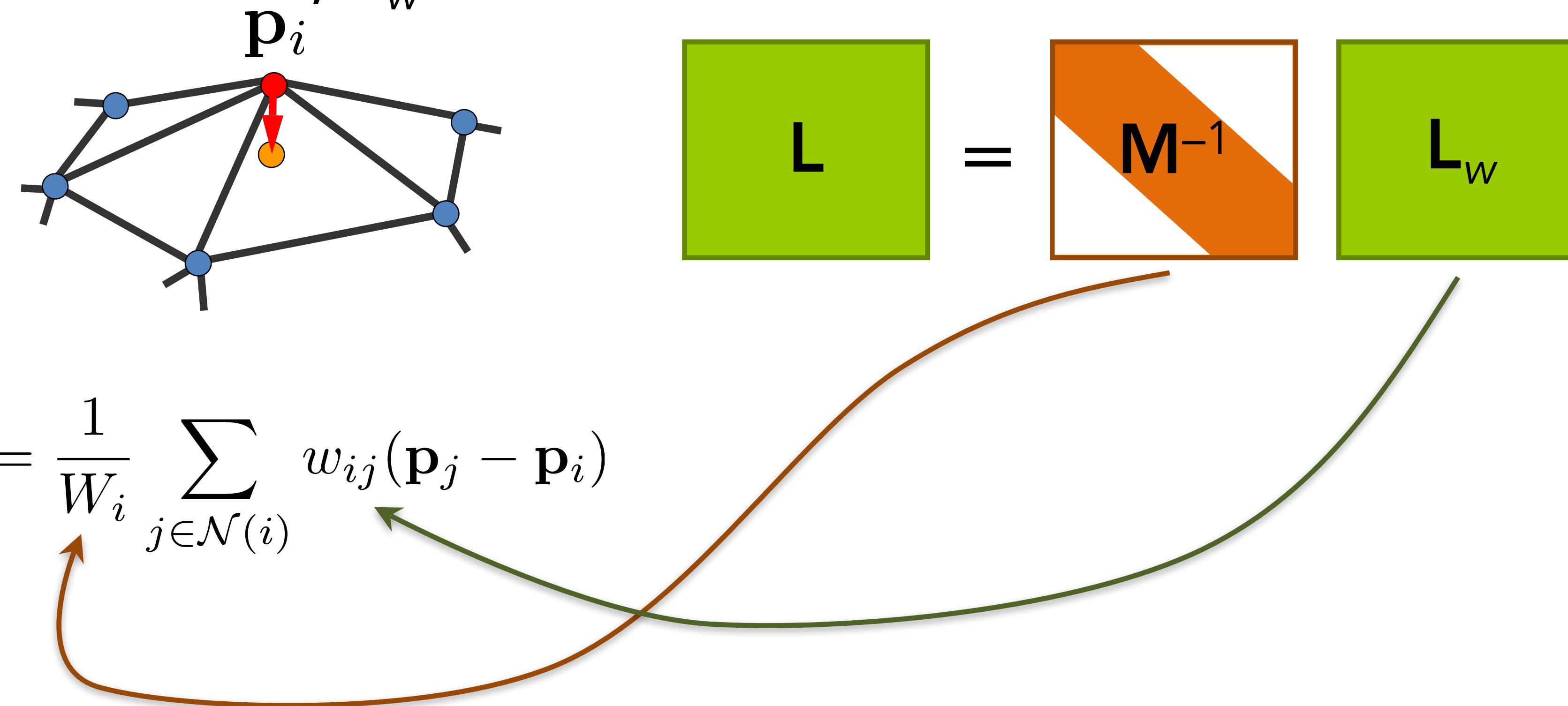
$$\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^n$$

$$\mathbf{L} \in \mathbb{R}^{n \times n}$$

$$\delta_x, \delta_y, \delta_z \in \mathbb{R}^n$$

Laplacian Matrix

- Breaking down the Laplace matrix:
- \mathbf{M} = mass matrix; \mathbf{L}_w = stiffness matrix

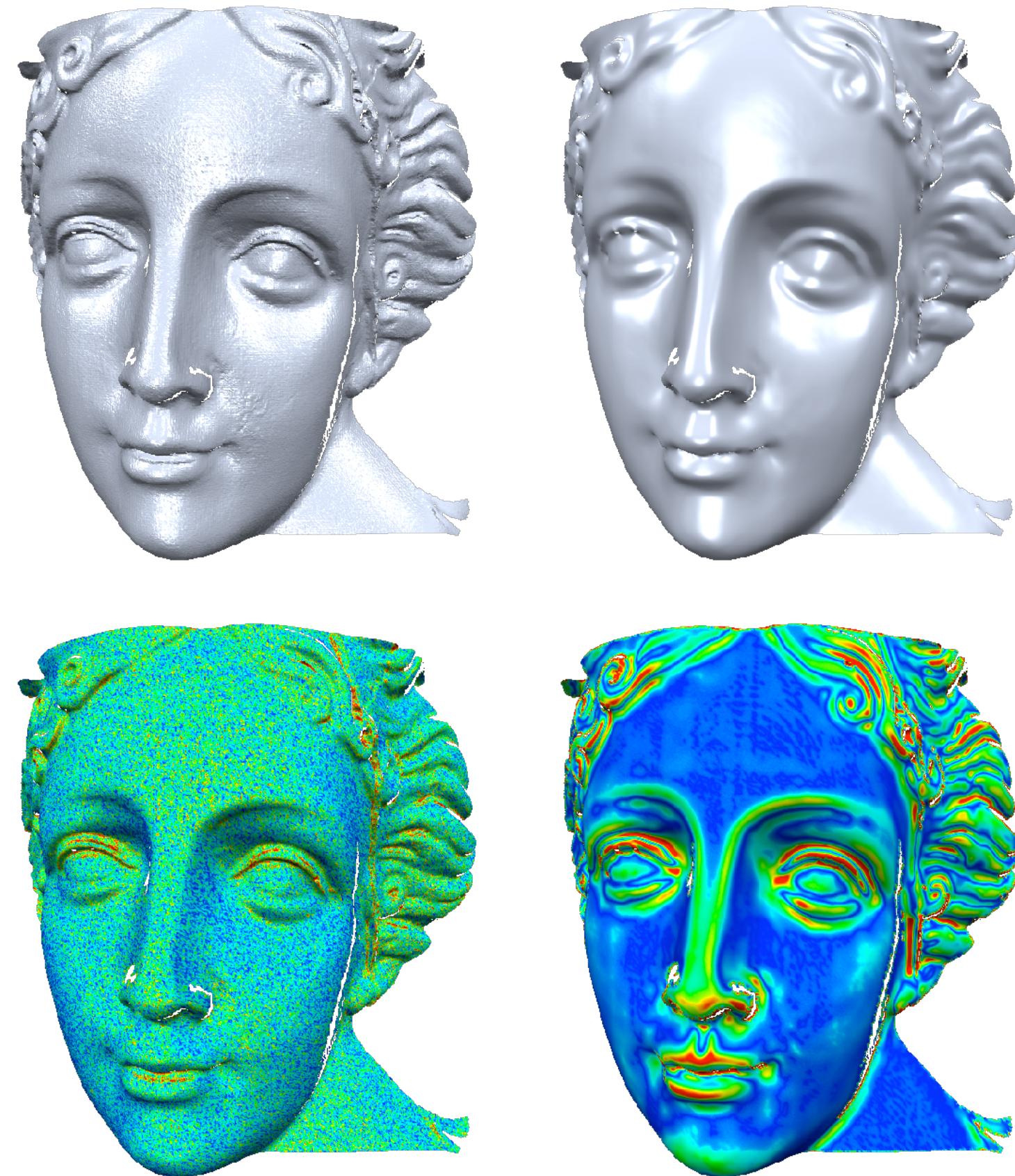


How to do the smoothing?

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

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

- Smooth H , obtain \tilde{H}
- Find a surface that has \tilde{H} as mean curvature
 - H doesn't define the surface
 - \mathbf{n} nonlinear in \mathbf{p}

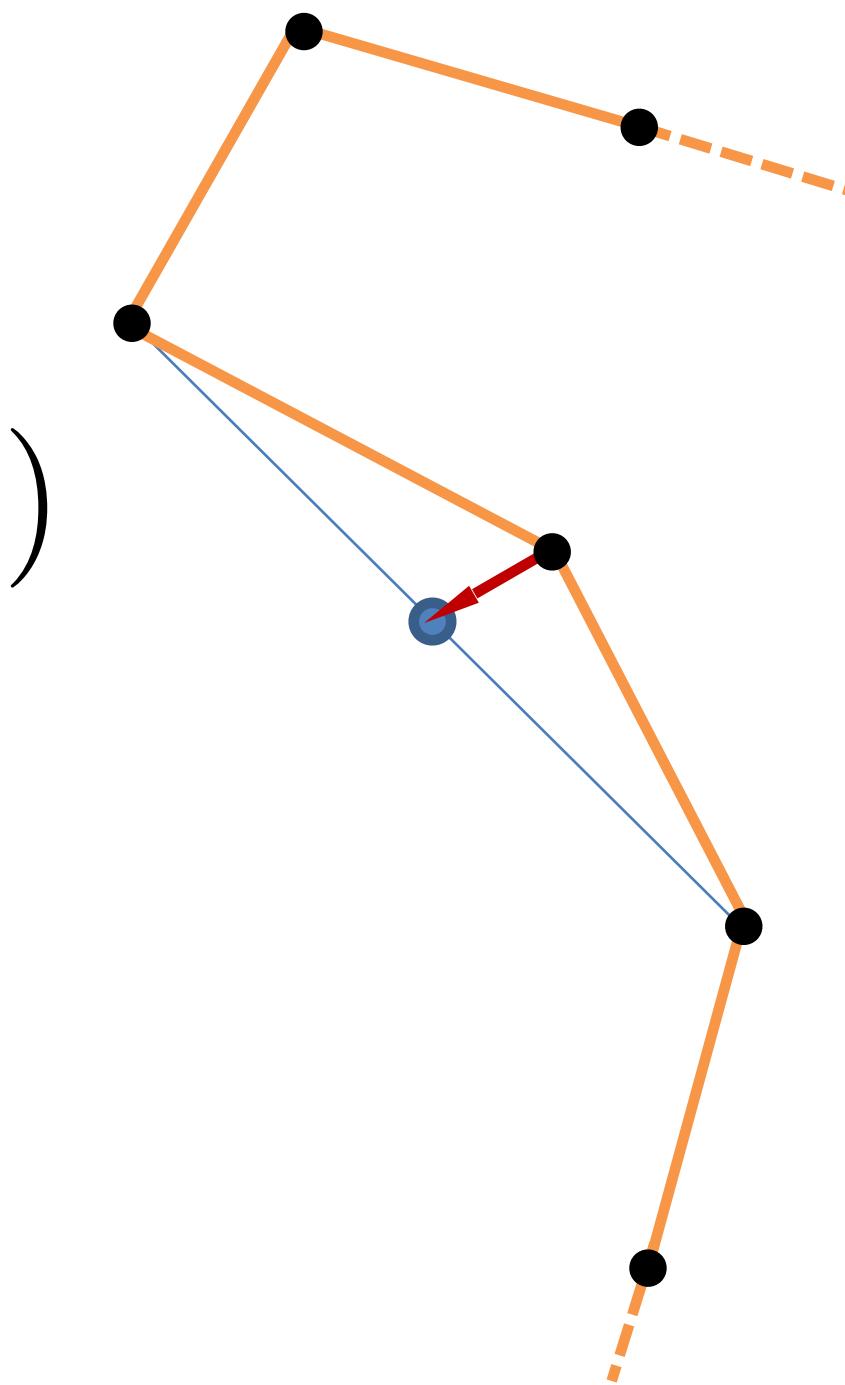


Method 1: Smoothing by Flowing

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

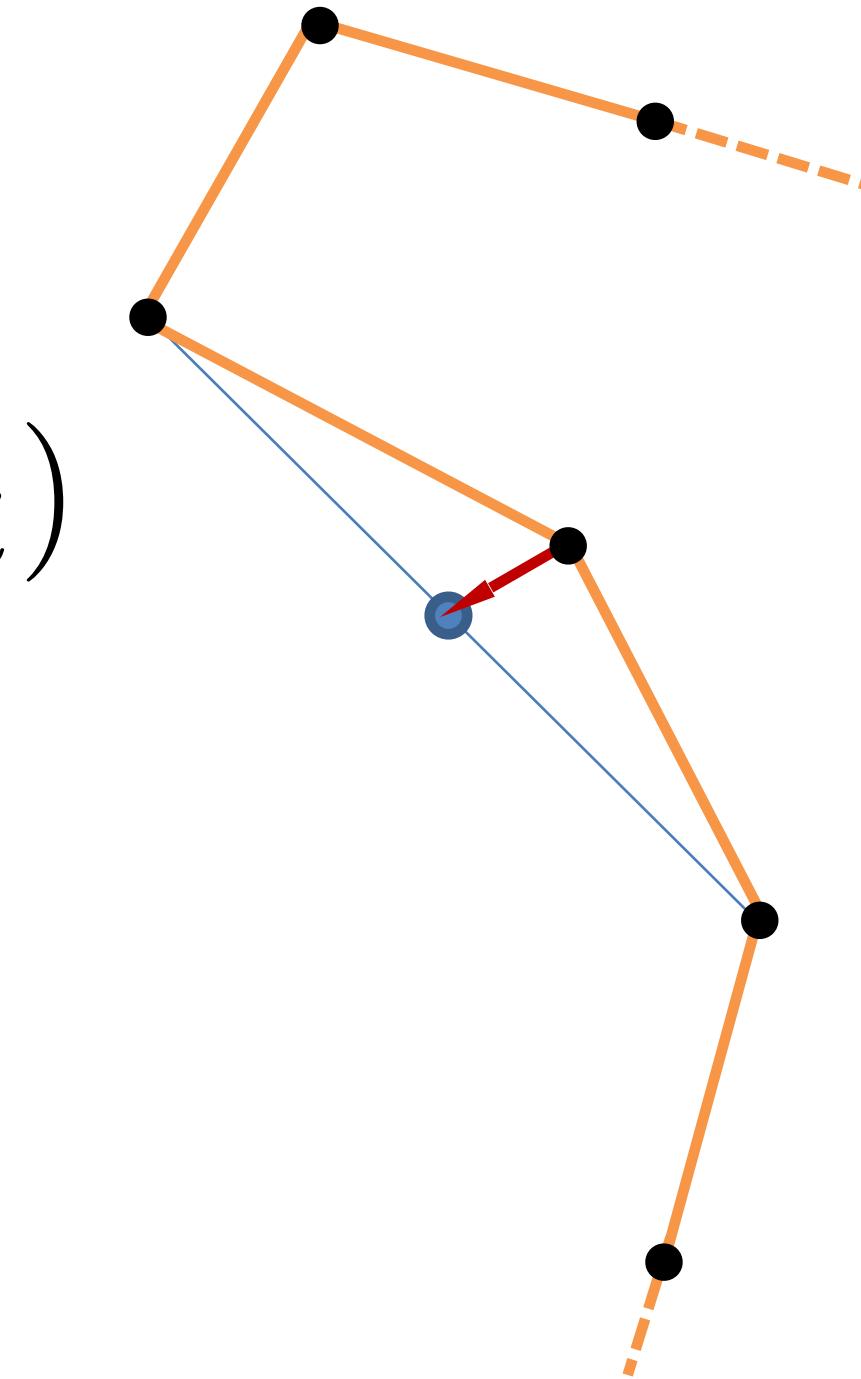
- Laplace in 1D = second derivative:

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

- In matrix-vector form for the whole curve

$L\mathbf{p}$

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



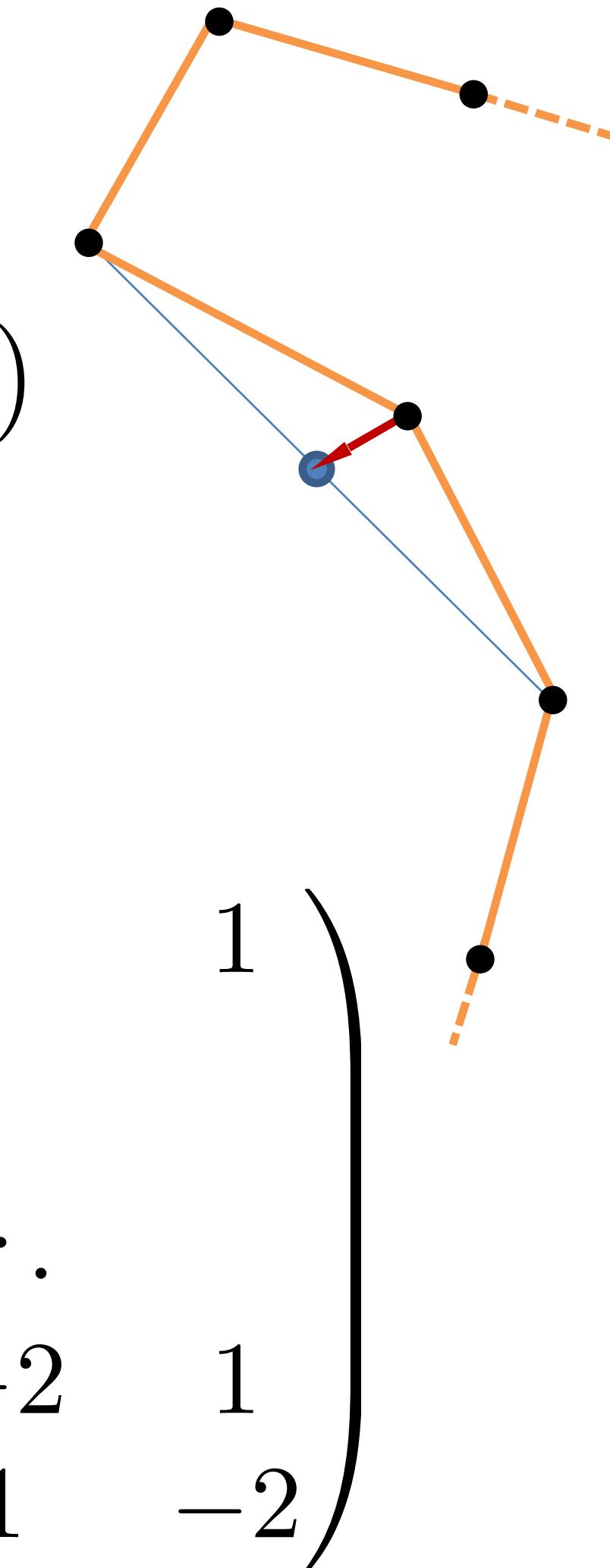
Example – smoothing curves

- 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

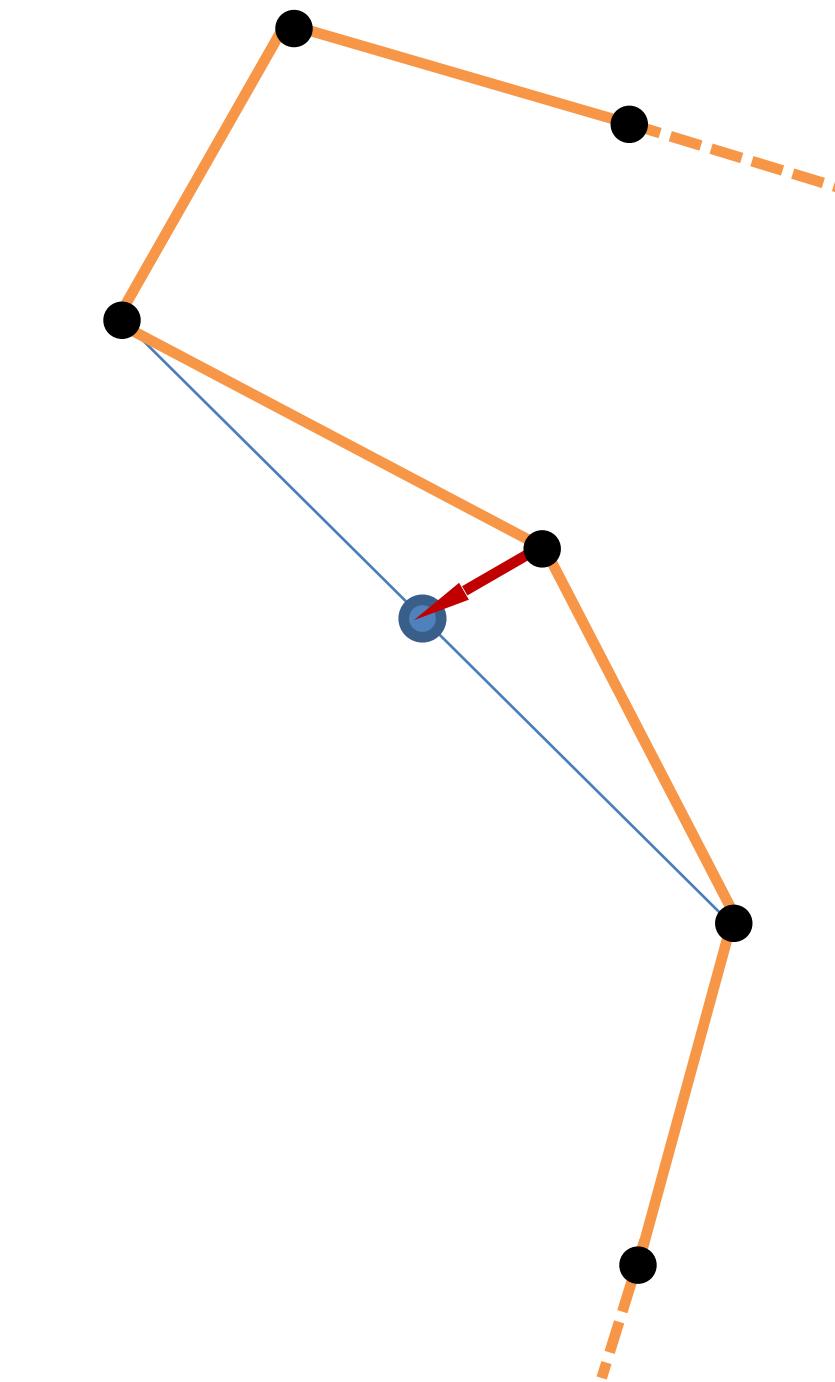
$$L\mathbf{p}$$
$$\mathbf{p} = [\mathbf{x} \ \mathbf{y}] \in \mathbb{R}^{n \times 2} \quad L = \frac{1}{2} \begin{pmatrix} -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 1 & & 1 & -2 & \end{pmatrix}$$



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:

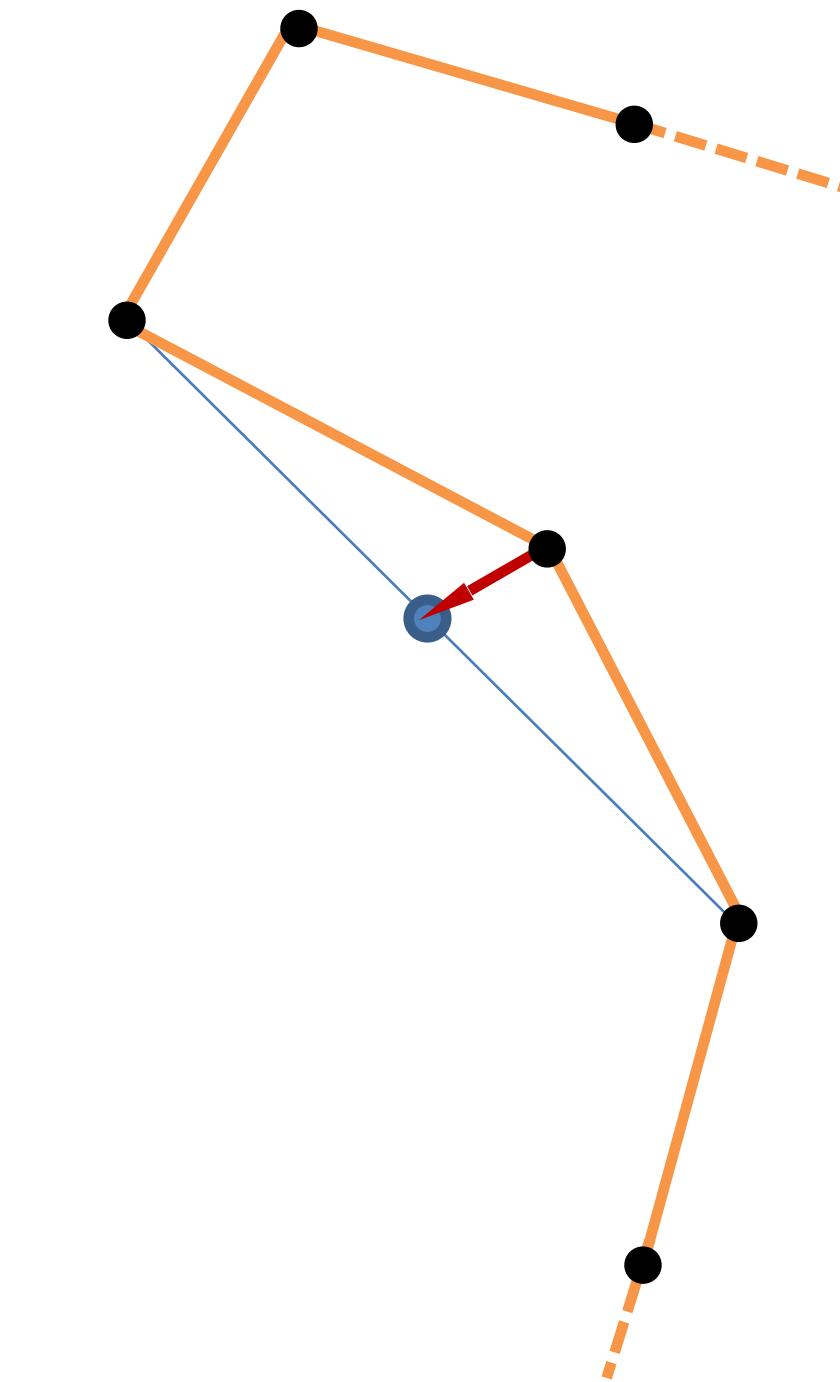
$$\tilde{\mathbf{p}} = \mathbf{p} + \lambda L \mathbf{p}, \quad \mathbf{p} \in \mathbb{R}^{n \times 2}$$

- Drawbacks?

Example – smoothing curves

- Flow to reduce curvature:

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

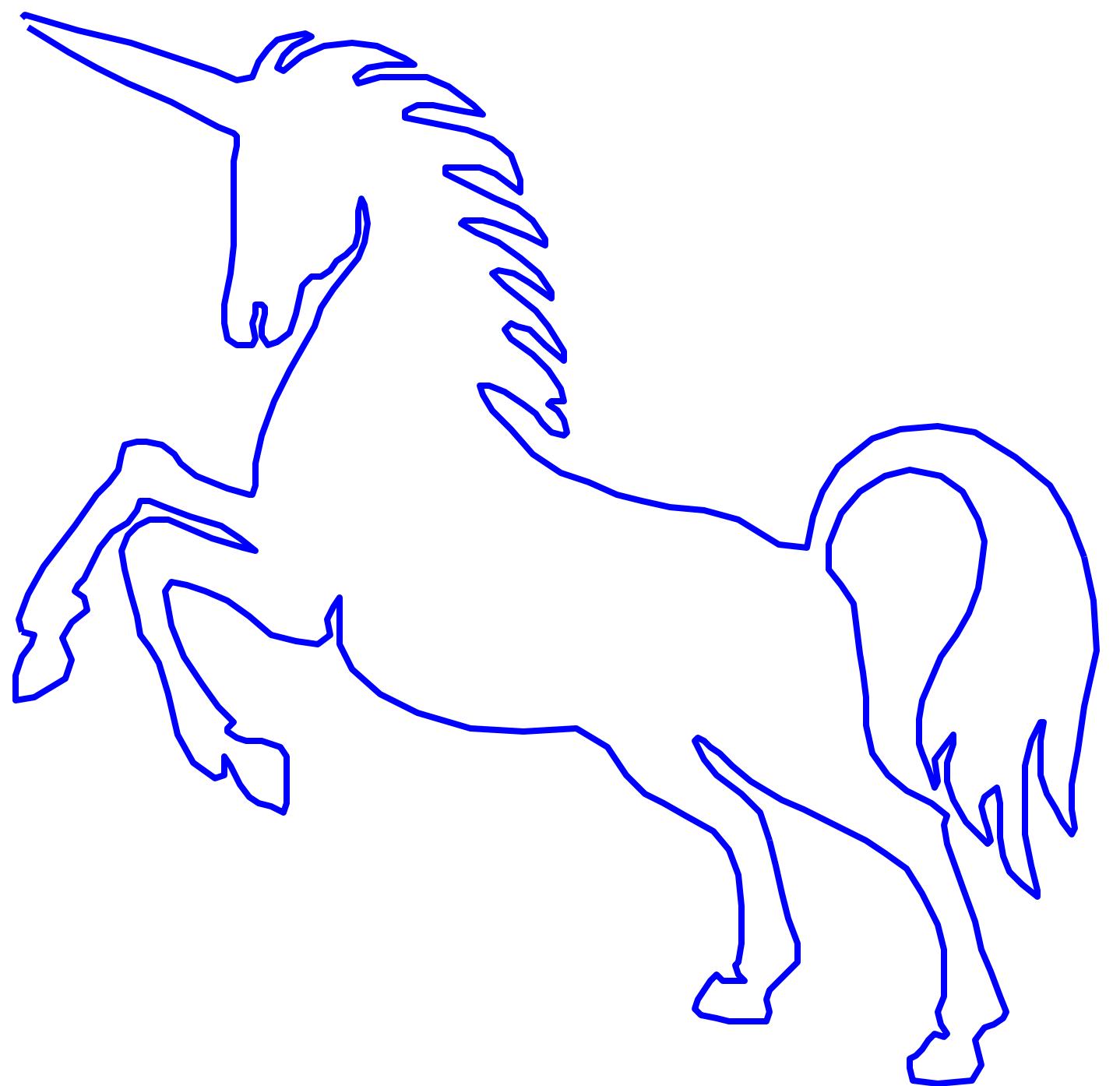


- Scale factor $0 < \lambda < 1$
- Matrix-vector form:

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

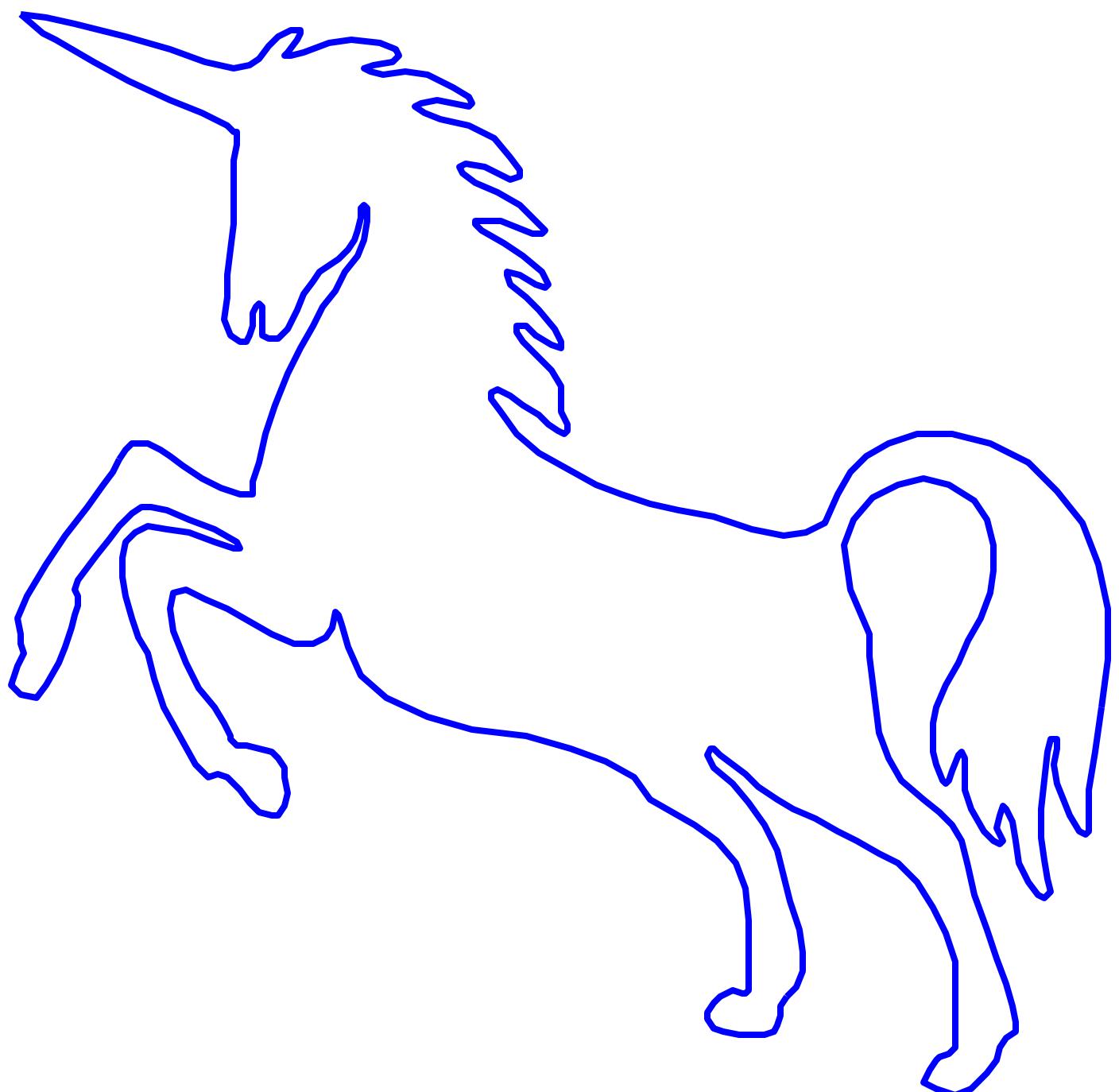
- May shrink the shape; can be slow

Filtering Curves



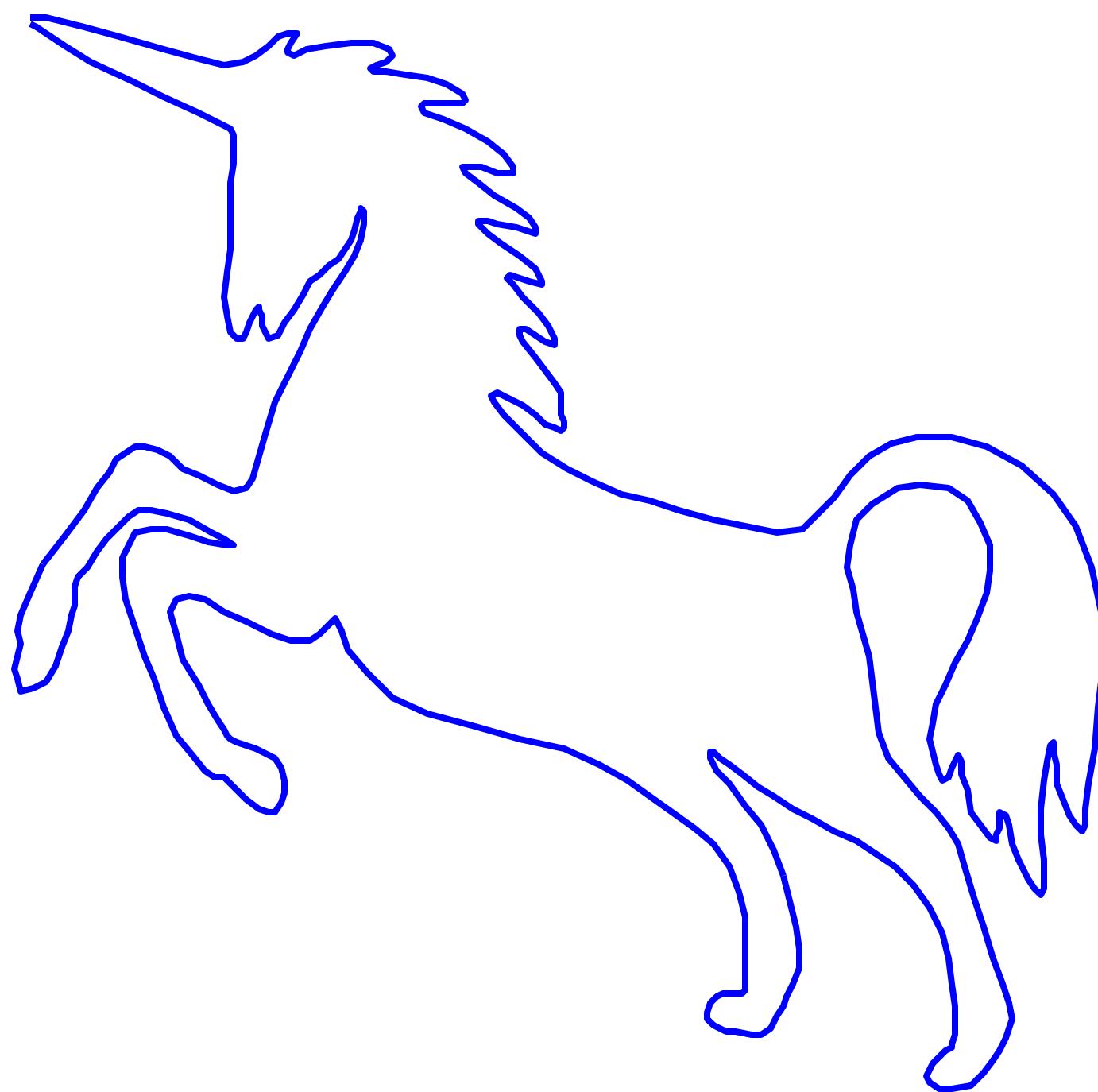
Original curve

Filtering Curves



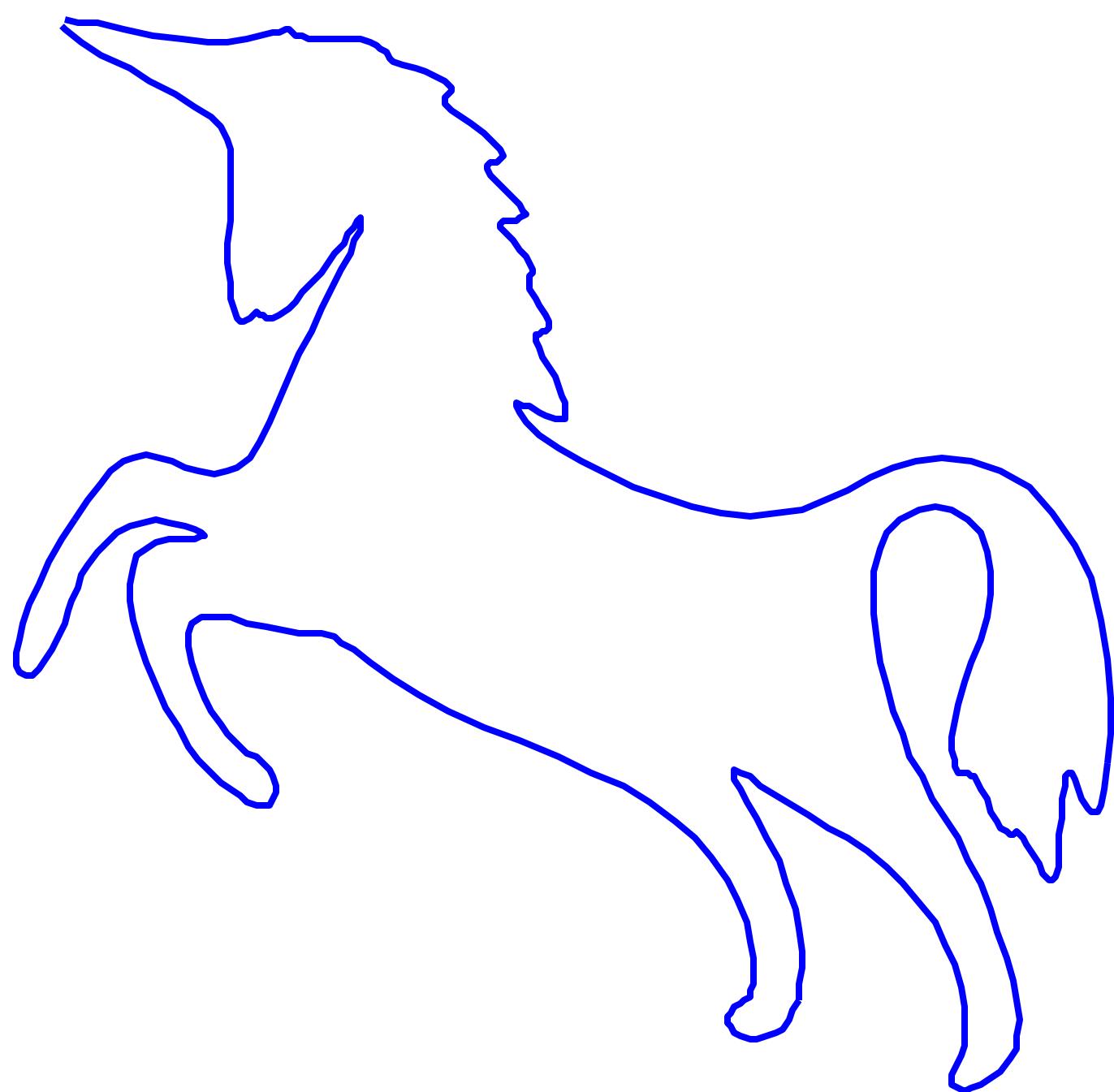
1st iteration; $\lambda=0.5$

Filtering Curves



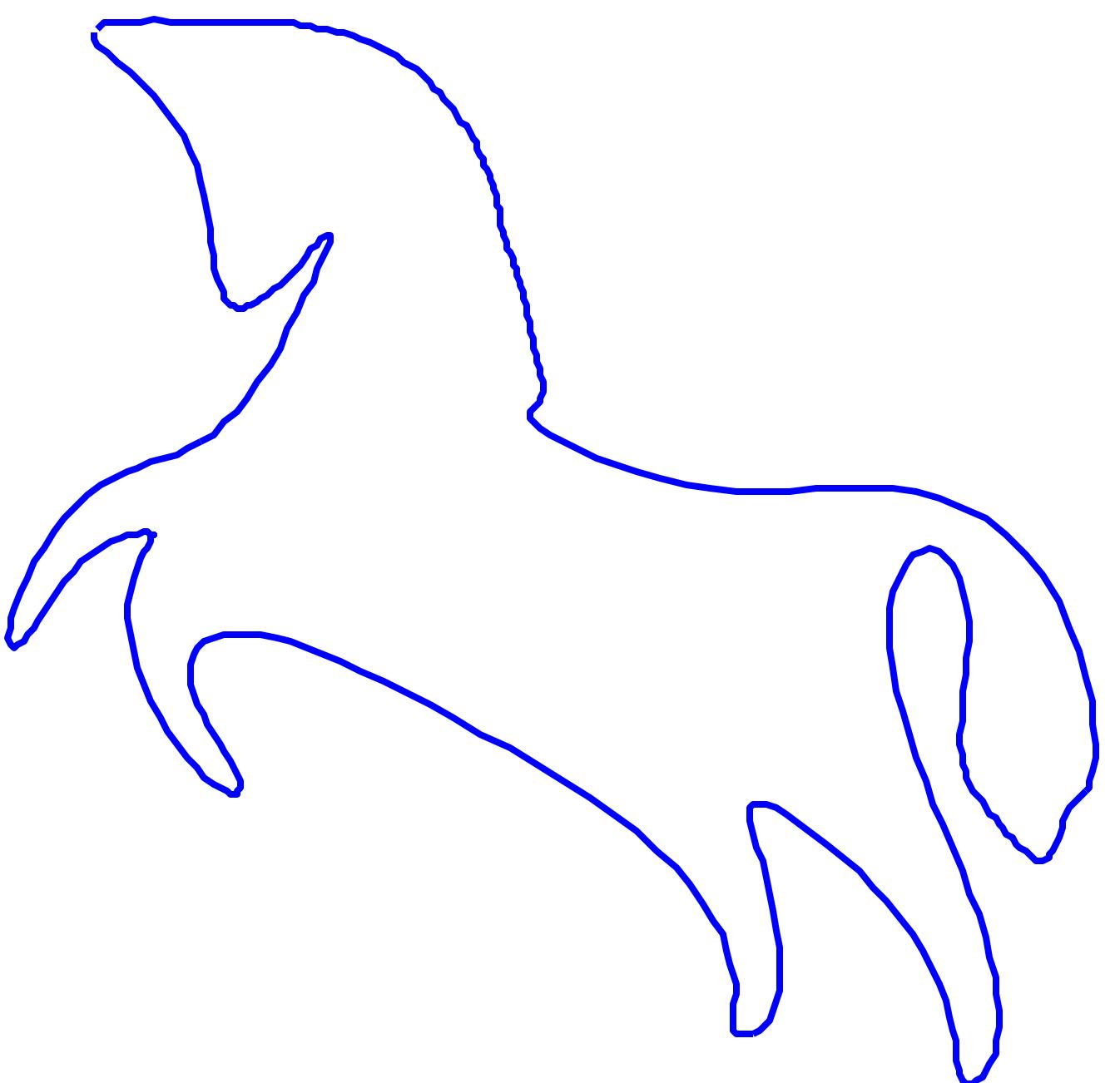
2nd iteration; $\lambda=0.5$

Filtering Curves



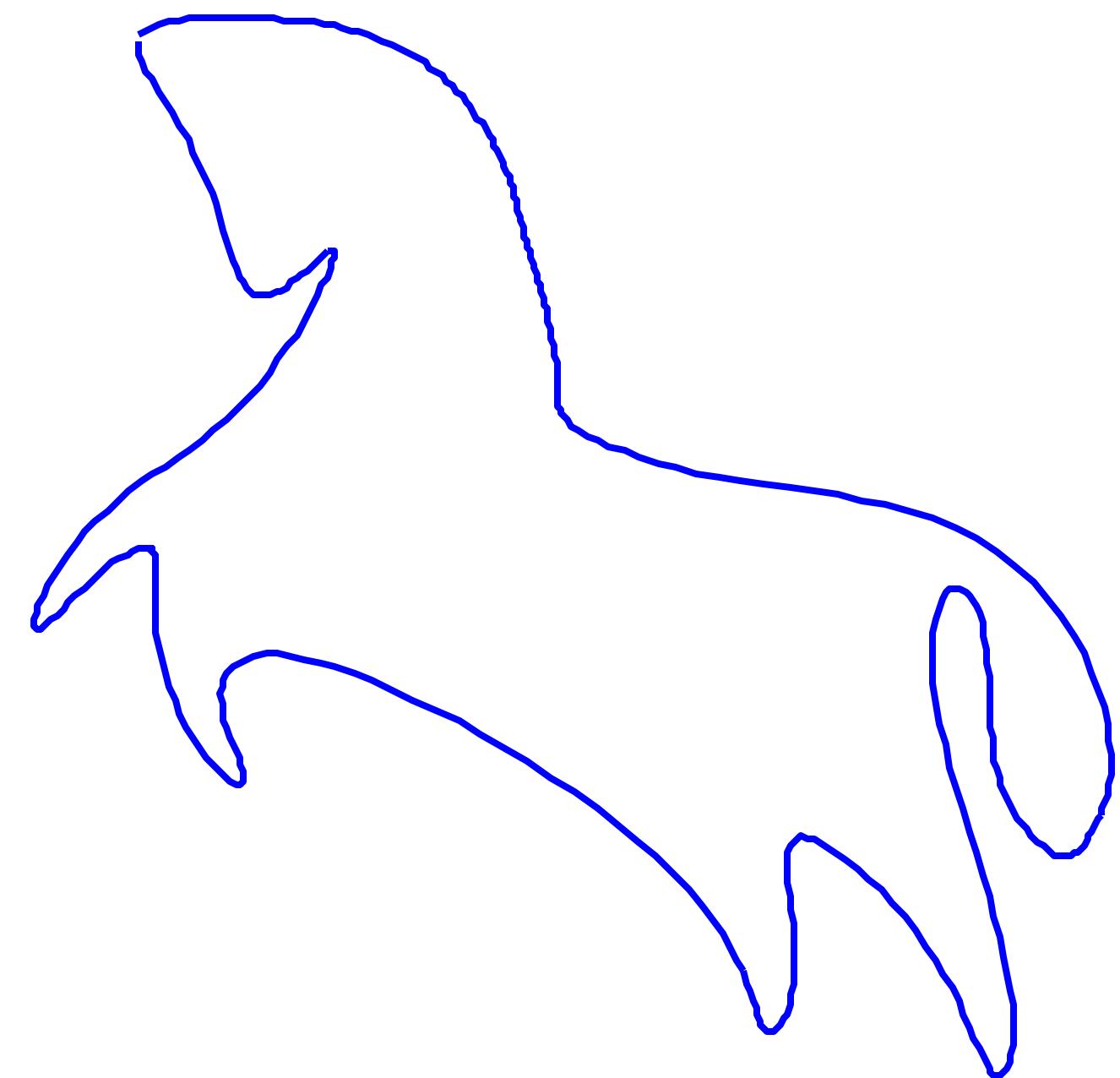
8th iteration; $\lambda=0.5$

Filtering Curves



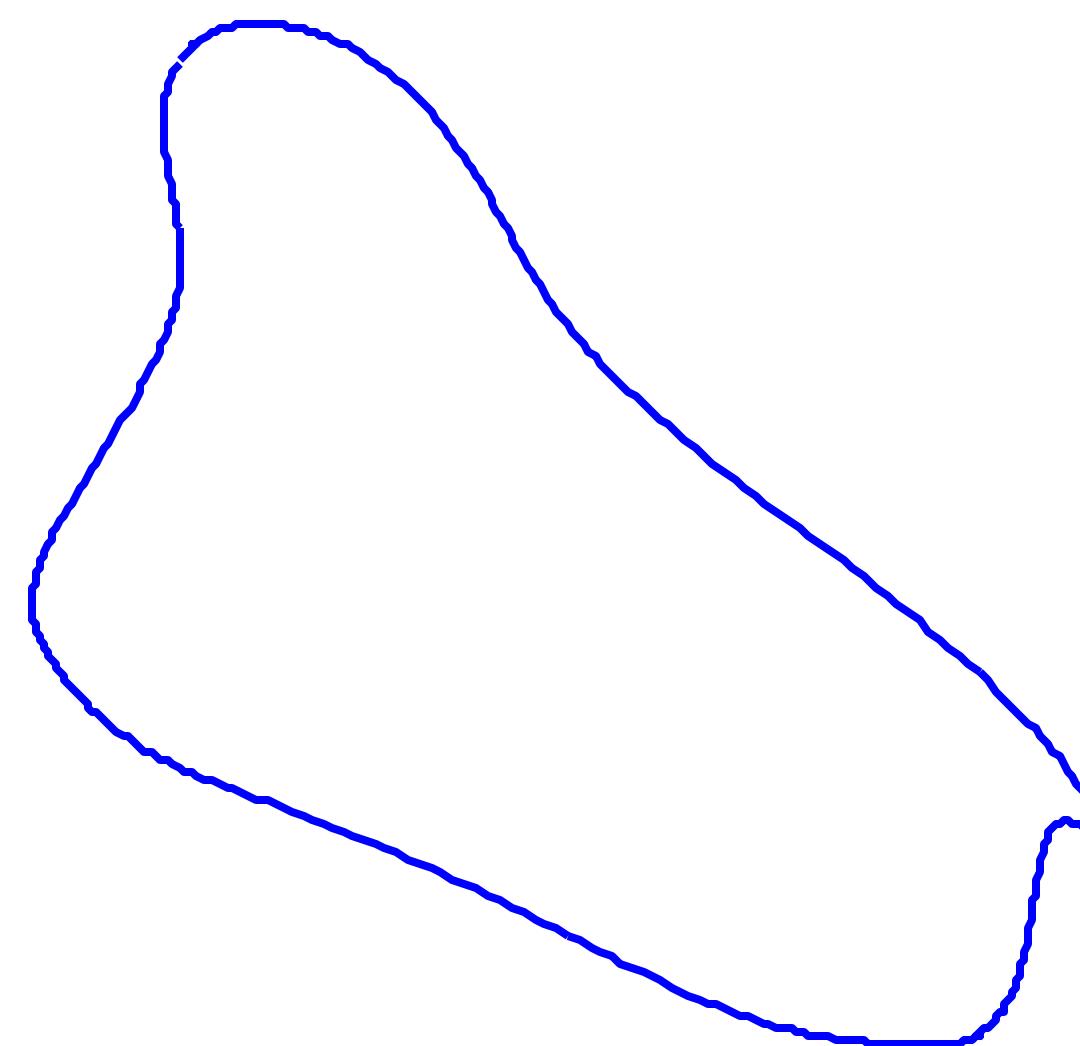
27th iteration; $\lambda=0.5$

Filtering Curves



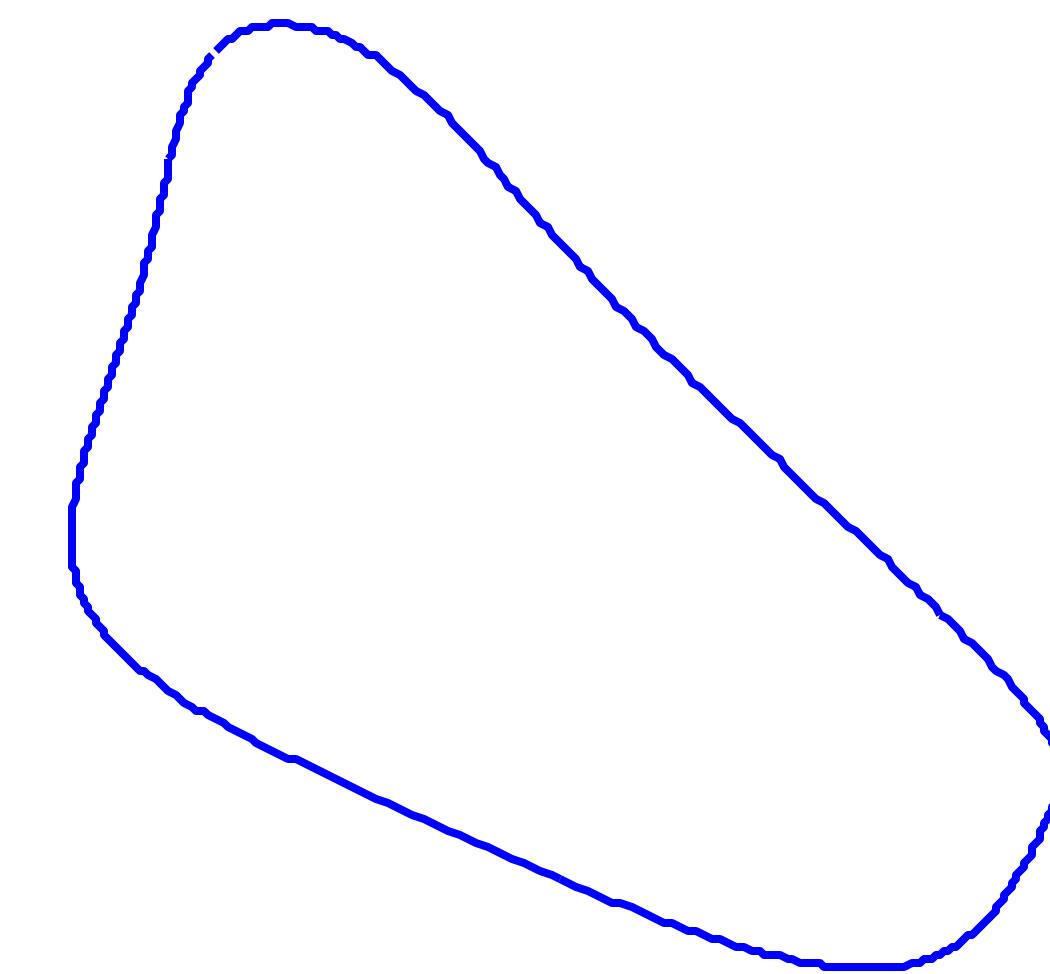
50th iteration; $\lambda=0.5$

Filtering Curves



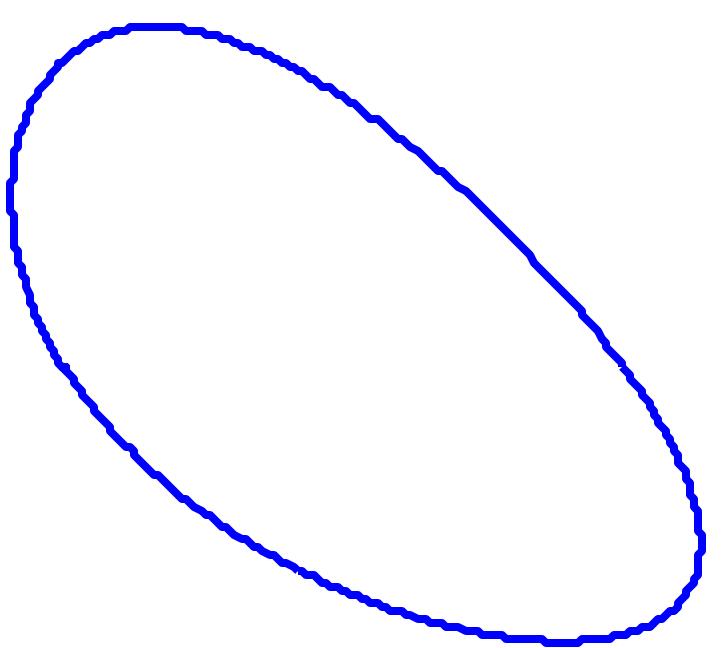
500th iteration; $\lambda=0.5$

Filtering Curves



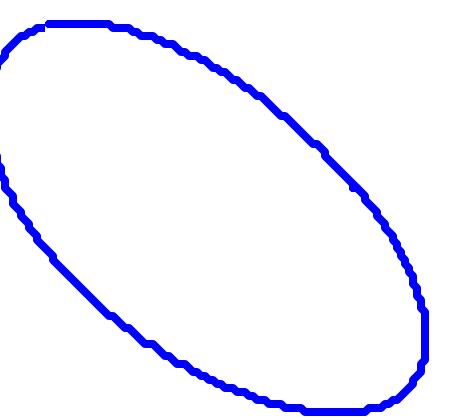
1000th iteration; $\lambda=0.5$

Filtering Curves



5000th iteration; $\lambda=0.5$

Filtering Curves



10000th iteration; $\lambda=0.5$

Filtering Curves

.

50000th iteration; $\lambda=0.5$

Smoothing as Mean Curvature Flow

- Model smoothing as a diffusion process

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

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:

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

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:

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

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

Smoothing as Mean Curvature Flow

- Model smoothing as a diffusion process

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

- Discretize in time, forward differences:

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

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

Explicit integration!
Unstable unless time step dt is small

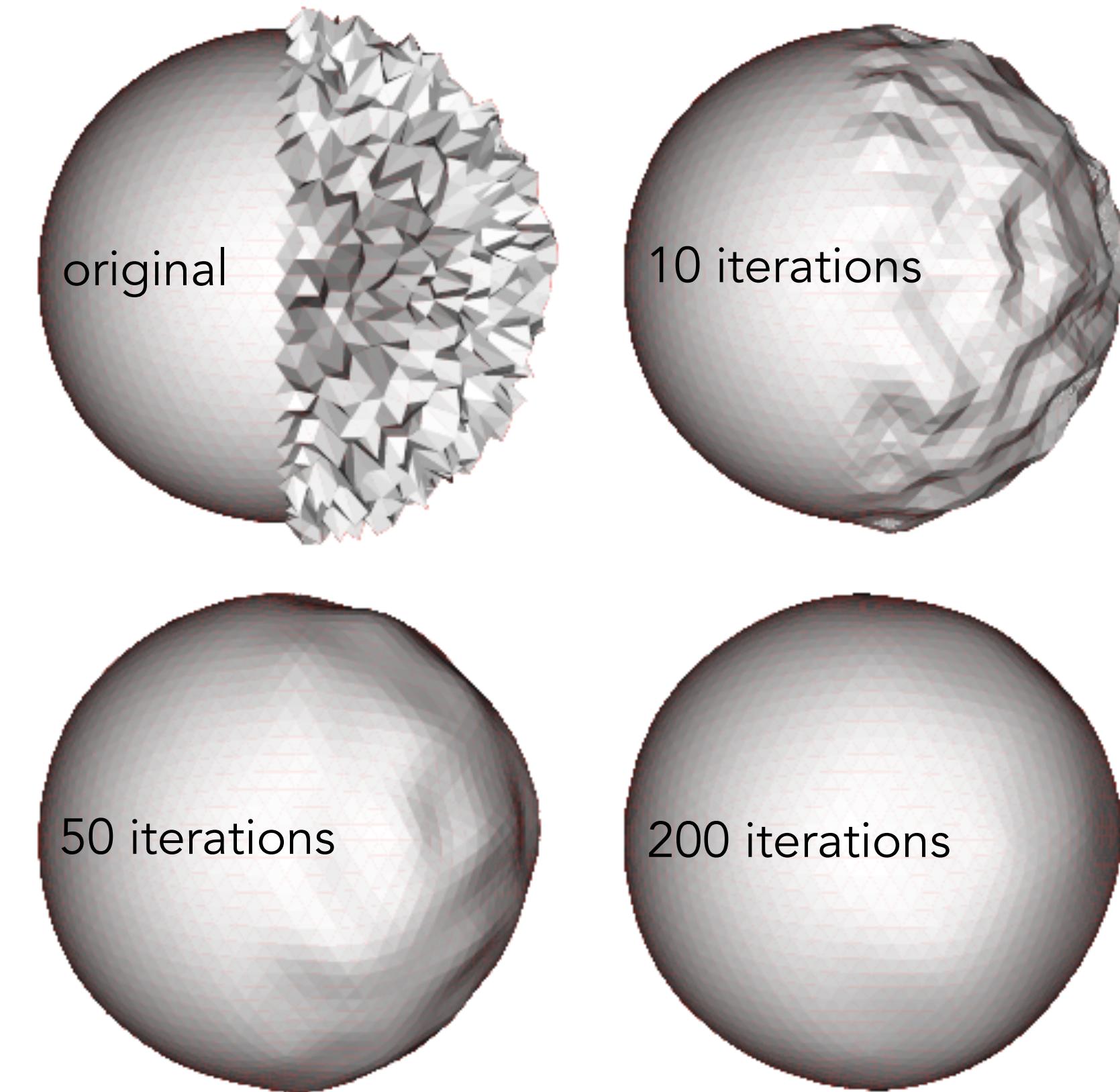
$$\mathbf{p}^{(n+1)} = (I + dt \lambda L) \mathbf{p}^{(n)}$$

Taubin Smoothing: Explicit Steps

- Iterate:

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

- $\lambda > 0$ to smooth
- $\lambda < 0$ to inflate
- Originally proposed with uniform Laplacian weights



A Signal Processing Approach to Fair Surface Design
Gabriel Taubin, ACM SIGGRAPH 95

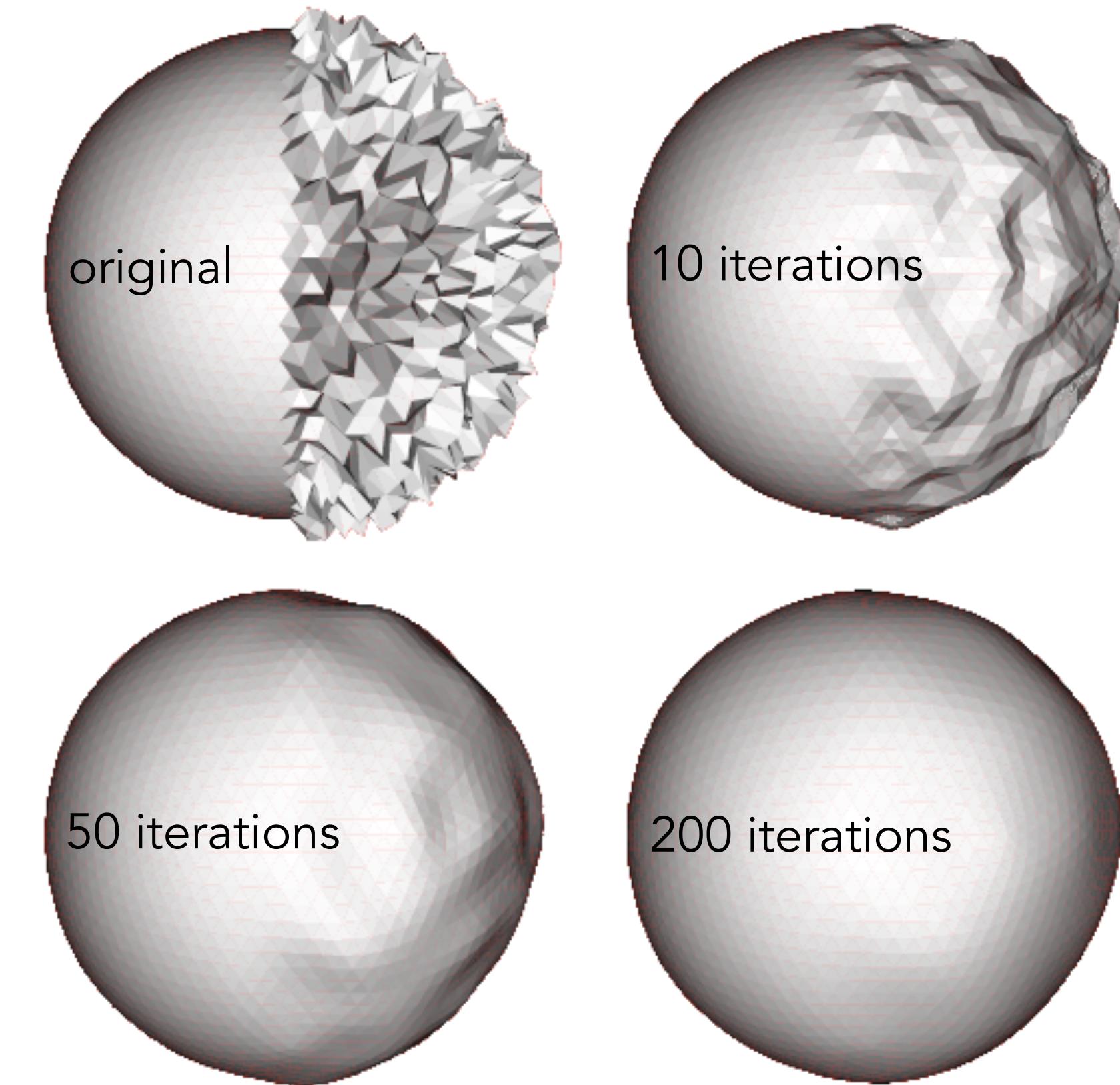
Taubin Smoothing: Explicit Steps

- Iterate:

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

$$\tilde{\mathbf{p}} = \mathbf{p} + \mu L \mathbf{p} = (I + \mu L) \mathbf{p}$$

- $\lambda > 0$ to smooth
- $\mu < 0$ to inflate
- Originally proposed with uniform Laplacian weights



A Signal Processing Approach to Fair Surface Design
Gabriel Taubin, ACM SIGGRAPH 95

Taubin Smoothing: Explicit Steps

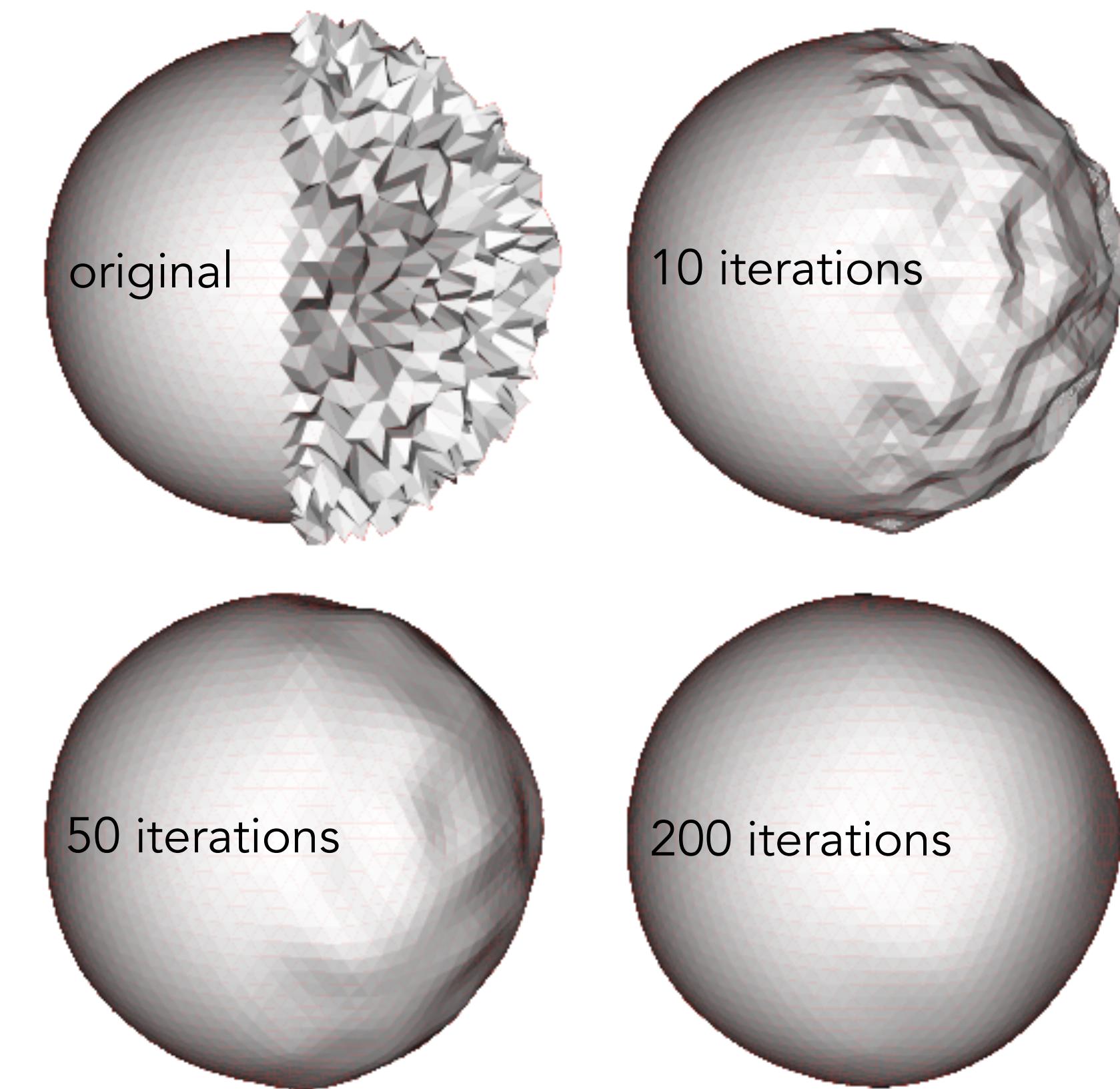
- Per-vertex iterations

$$\tilde{\mathbf{p}}_i = \mathbf{p}_i + \lambda L(\mathbf{p}_i)$$

$$\tilde{\mathbf{p}}_i = \mathbf{p}_i + \mu L(\mathbf{p}_i)$$

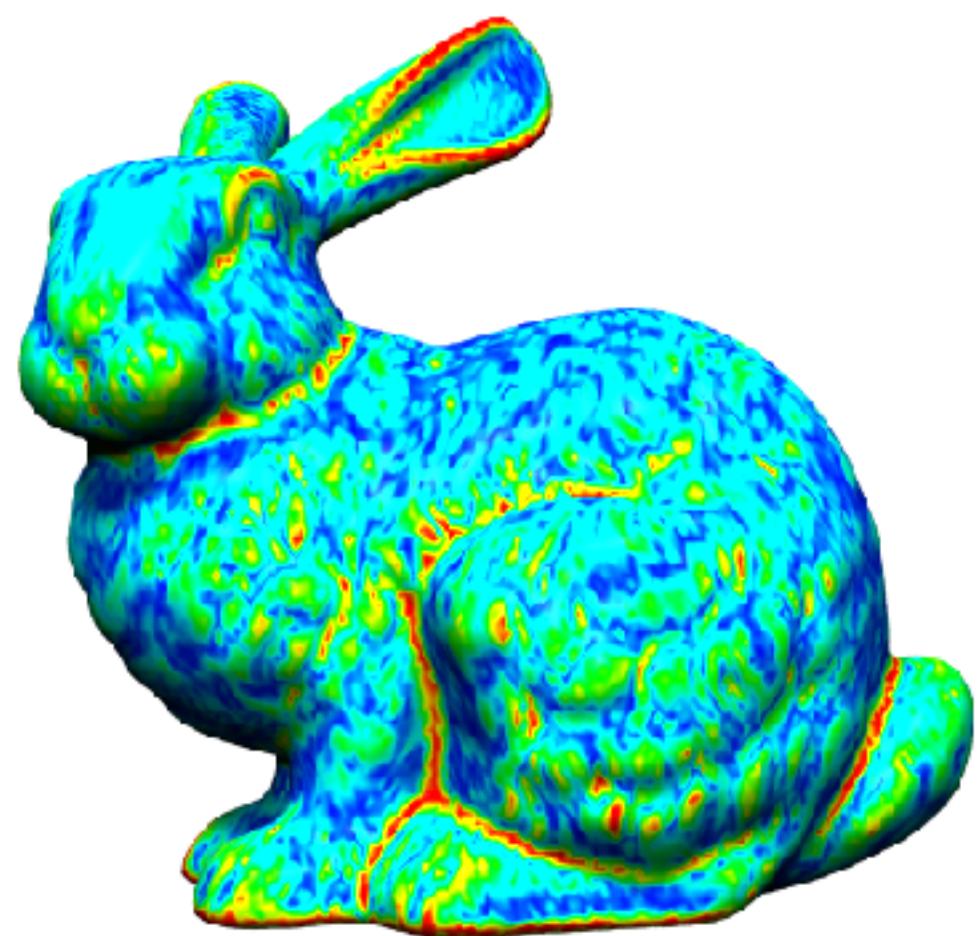
- Simple to implement

- Requires many iterations
- Need to tweak λ, u

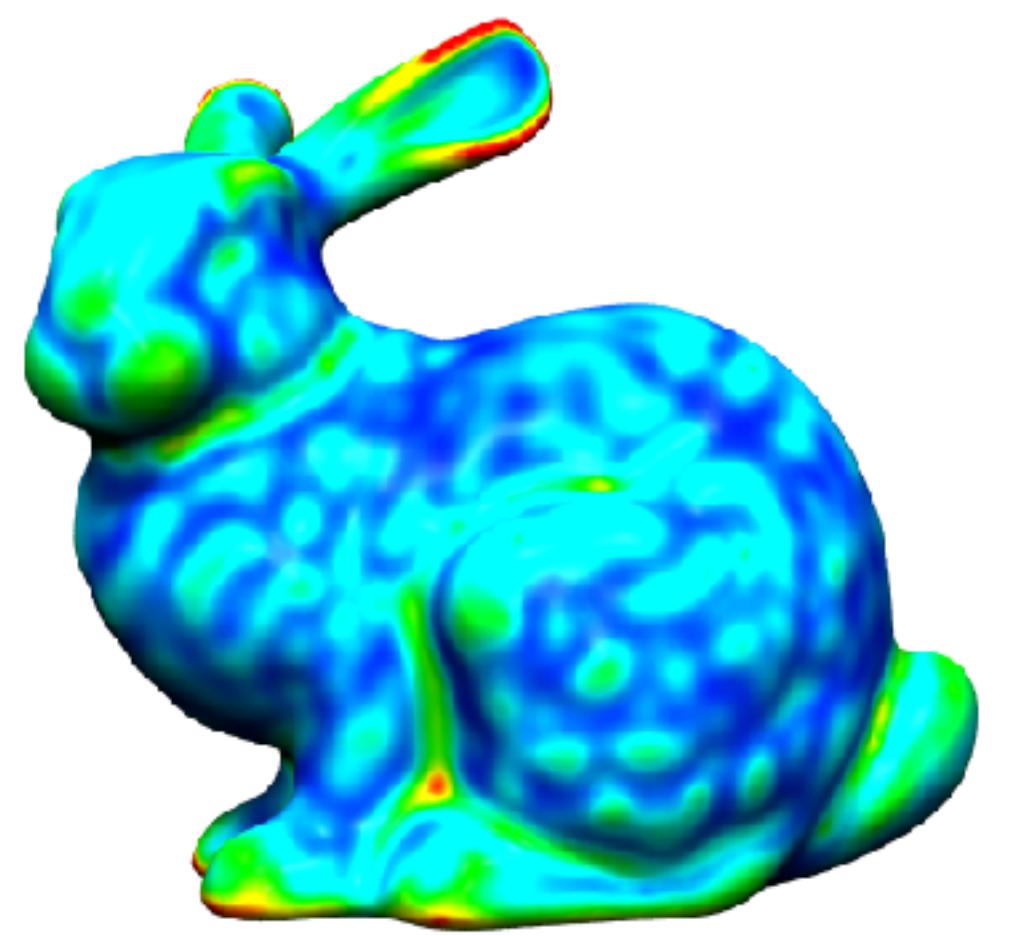


A Signal Processing Approach to Fair Surface Design
Gabriel Taubin, ACM SIGGRAPH 95

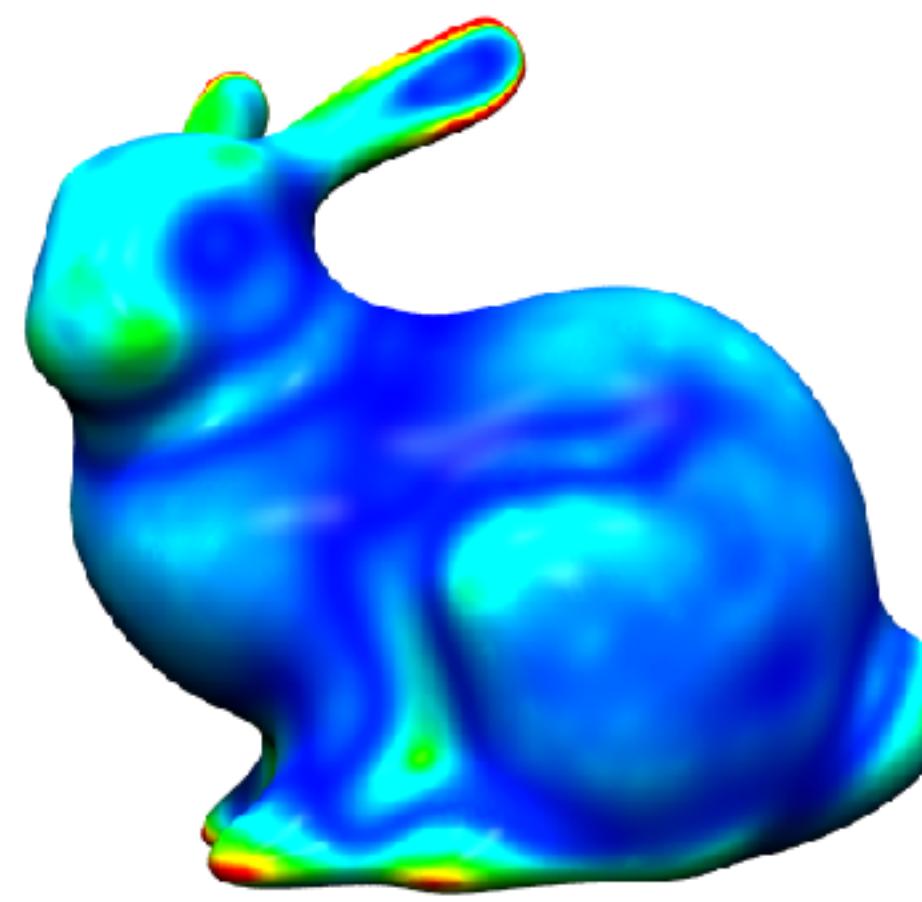
Example



0 iterations



10 iterations



100 iterations

Smoothing as (mean curvature) Flow

- Model smoothing as a diffusion process

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

- Backward Euler for unconditional stability

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

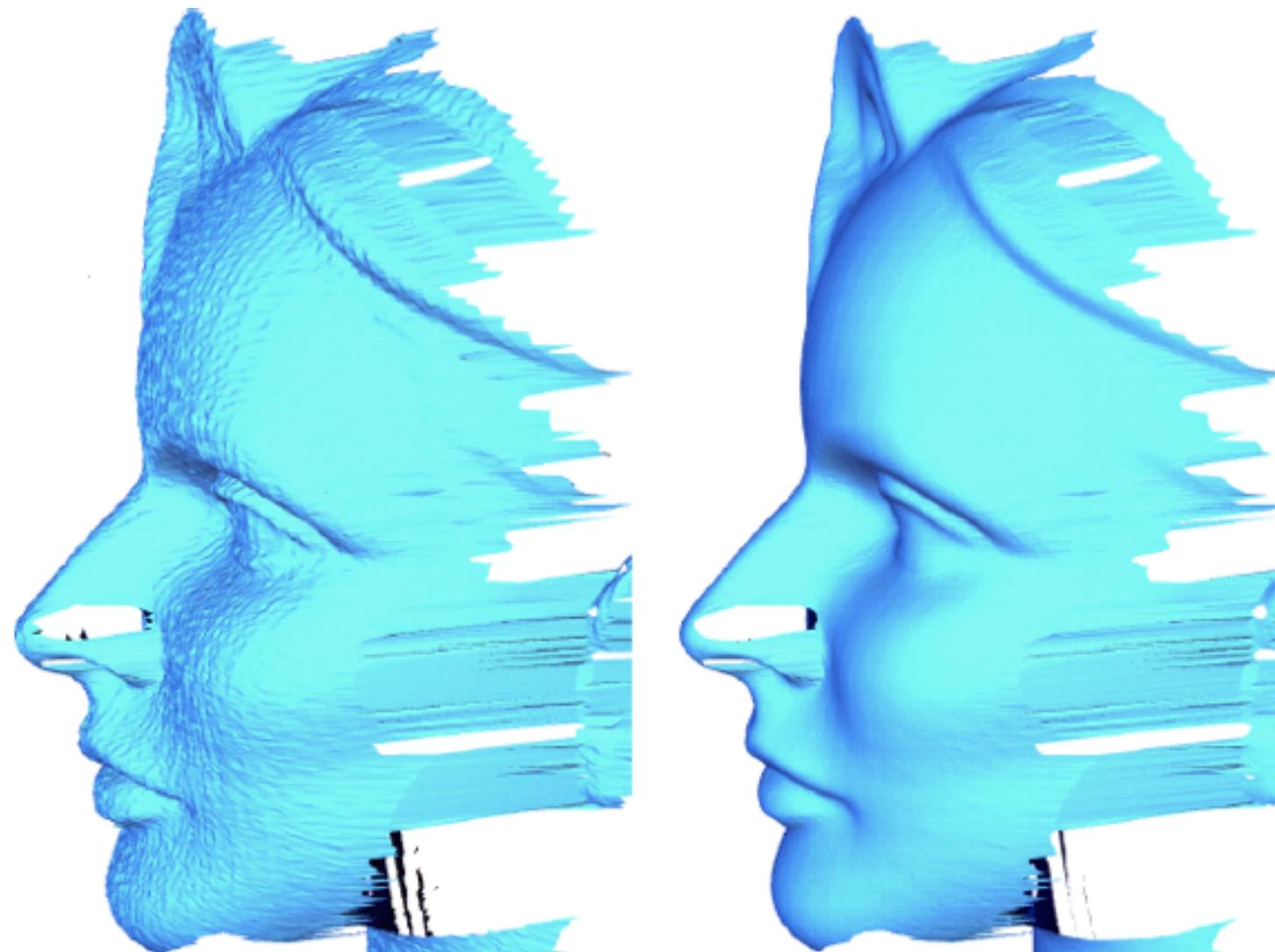
$$\mathbf{p}^{(n+1)} - \mathbf{p}^{(n)} = dt \lambda L \mathbf{p}^{(n+1)}$$

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

Implicit Fairing: Implicit Euler Steps

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

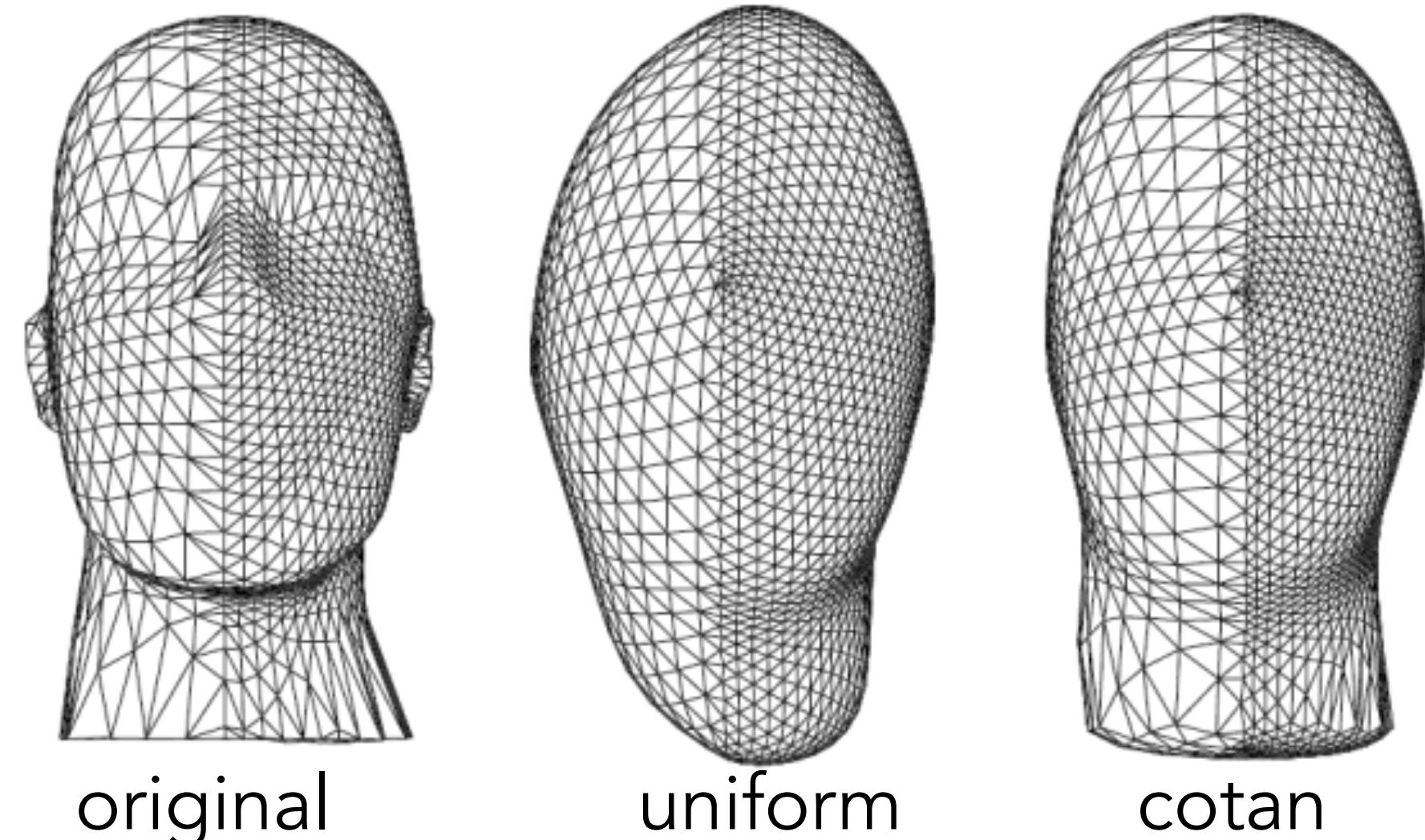
$$(I - \lambda L)\tilde{\mathbf{p}} = \mathbf{p}$$



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

Mesh Independence

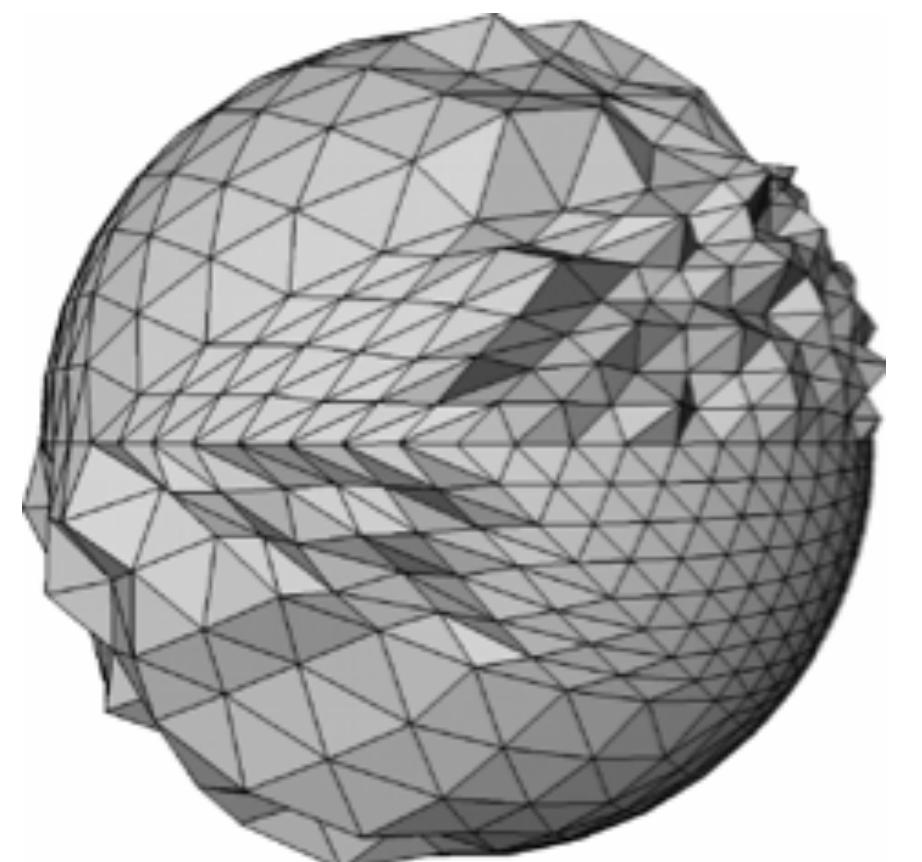
- Result of smoothing with uniform Laplacian depends on triangle density and shape
 - Why?
- Asymmetric results although underlying geometry is symmetric



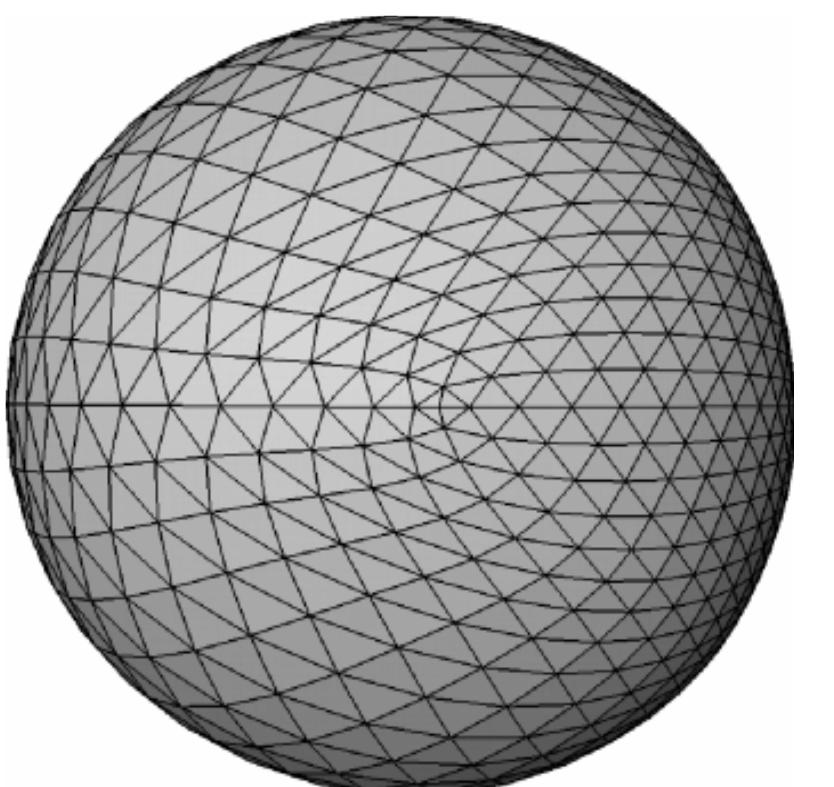
Comparison of the weights

- Explicit flow with different weights:

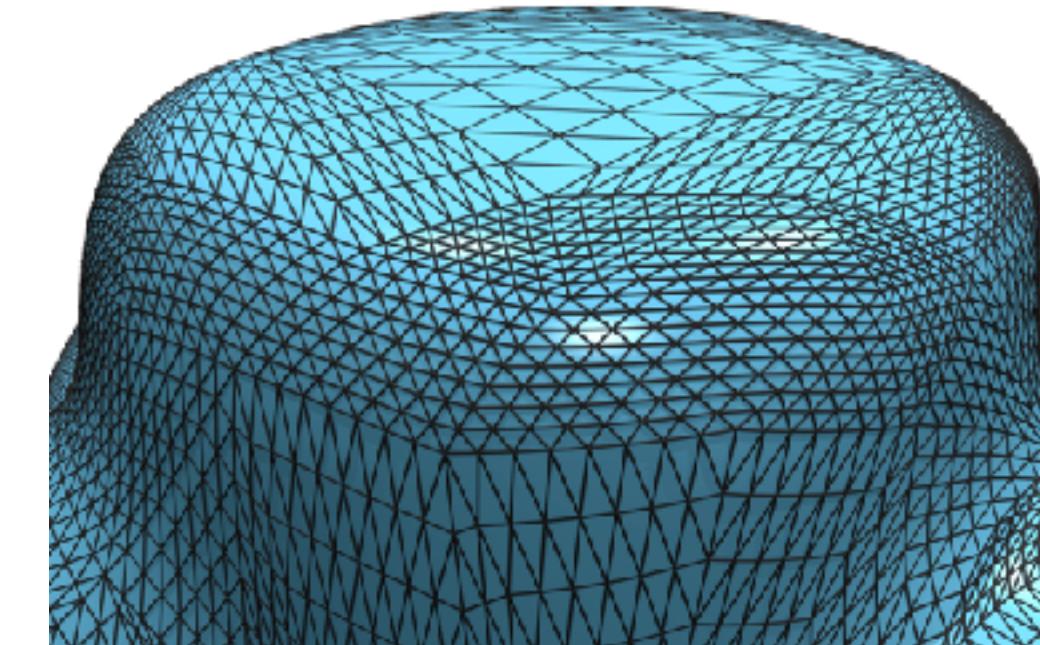
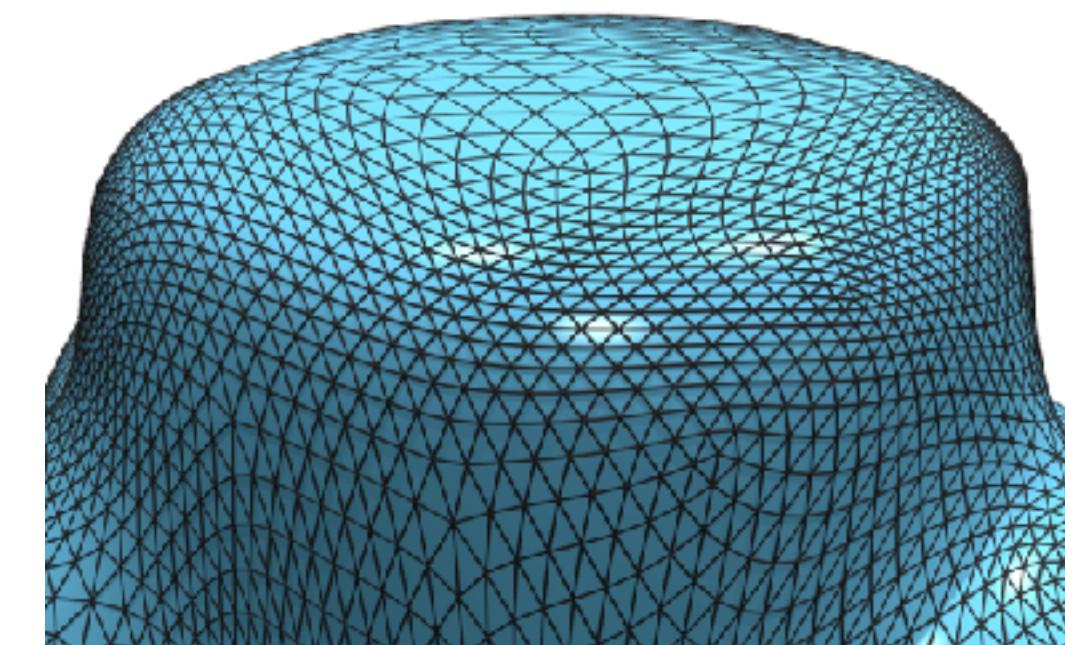
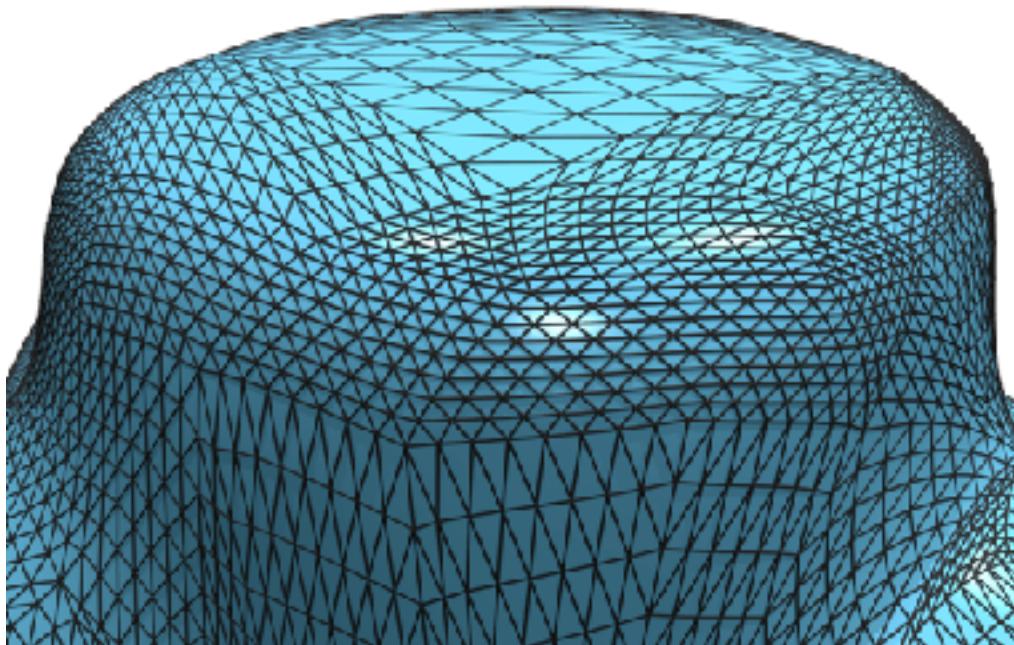
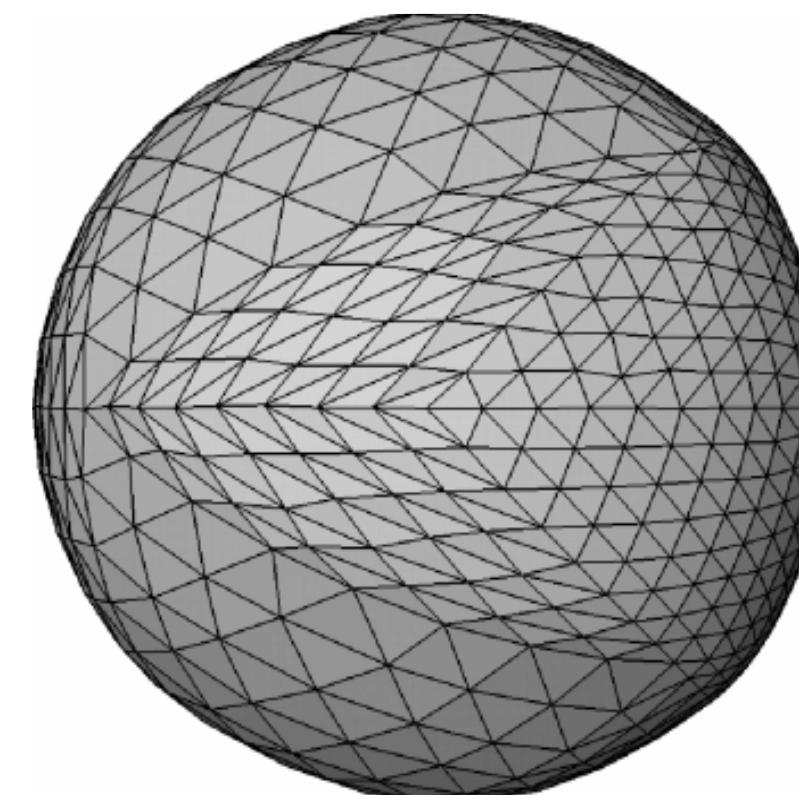
original



uniform

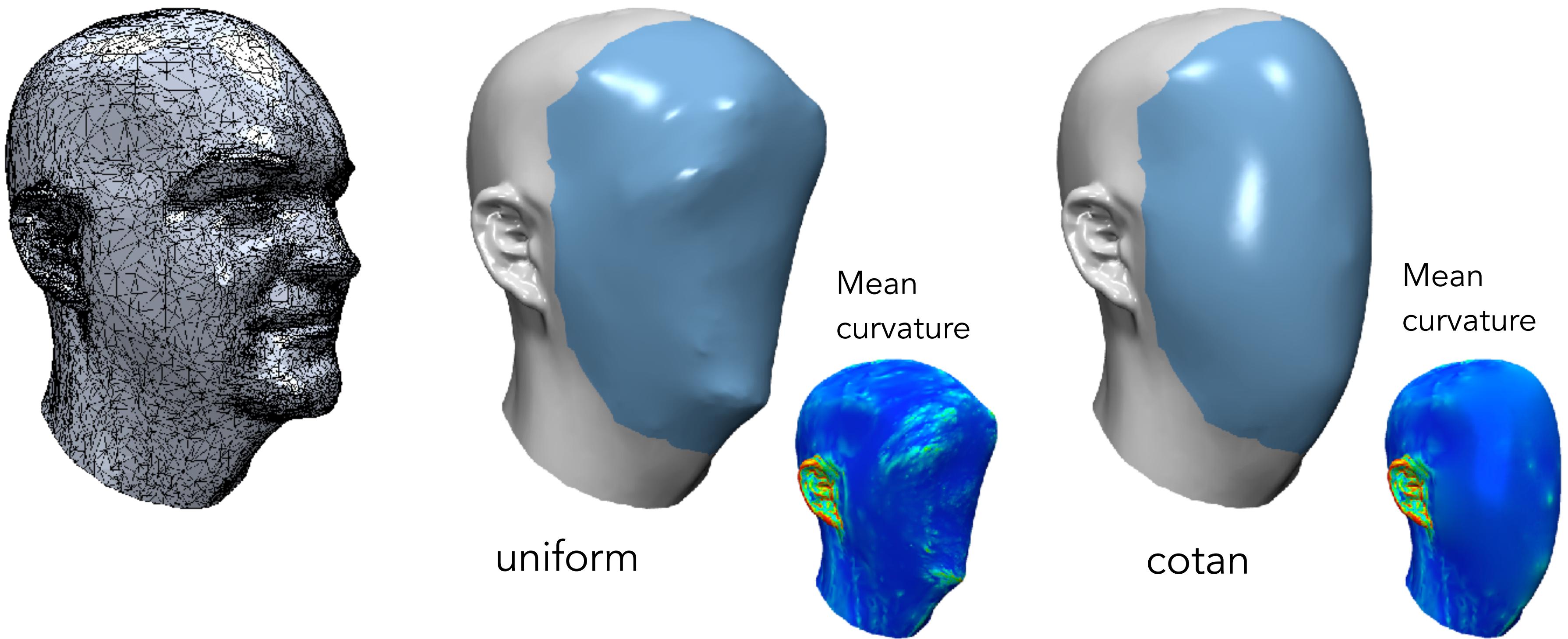


cotan



Implicit Fairing

- The importance of using the right weights



Method 2: Smoothing as Optimization

Minimizing a smoothness energy

- Let's go for $H = 0$ $\Delta_{\mathcal{M}} \mathbf{p} = -2H\mathbf{n}$
goal: $H = 0$ or $H = \text{const}$
- only trivial solution, no connection to initial surface \mathbf{p}

$$\Delta_{\mathcal{M}} \tilde{\mathbf{p}} = 0$$

Minimizing a smoothness energy

- Let's go for $H = 0$ $\Delta_{\mathcal{M}} \mathbf{p} = -2H\mathbf{n}$
goal: $H = 0$ or $H = \text{const}$
- only trivial solution, no connection to initial surface \mathbf{p}
- Let's regularize!

$$\Delta_{\mathcal{M}} \tilde{\mathbf{p}} = 0$$

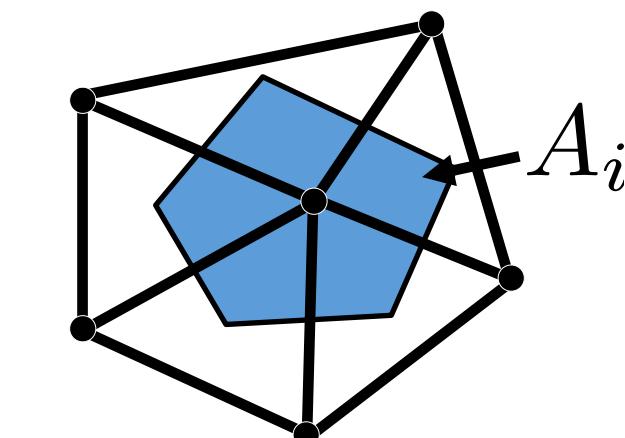
$$\min_{\tilde{\mathbf{p}}} \int_{\mathcal{M}} \frac{\|\Delta_{\mathcal{M}} \tilde{\mathbf{p}}\|^2}{\text{small } H} + \underbrace{w \|\tilde{\mathbf{p}} - \mathbf{p}\|^2}_{\text{stay close to original surface}}$$

weighting factor

Minimizing a smoothness energy

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

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



- 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: google "The Matrix Cookbook"

Minimizing a smoothness energy

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

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

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

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

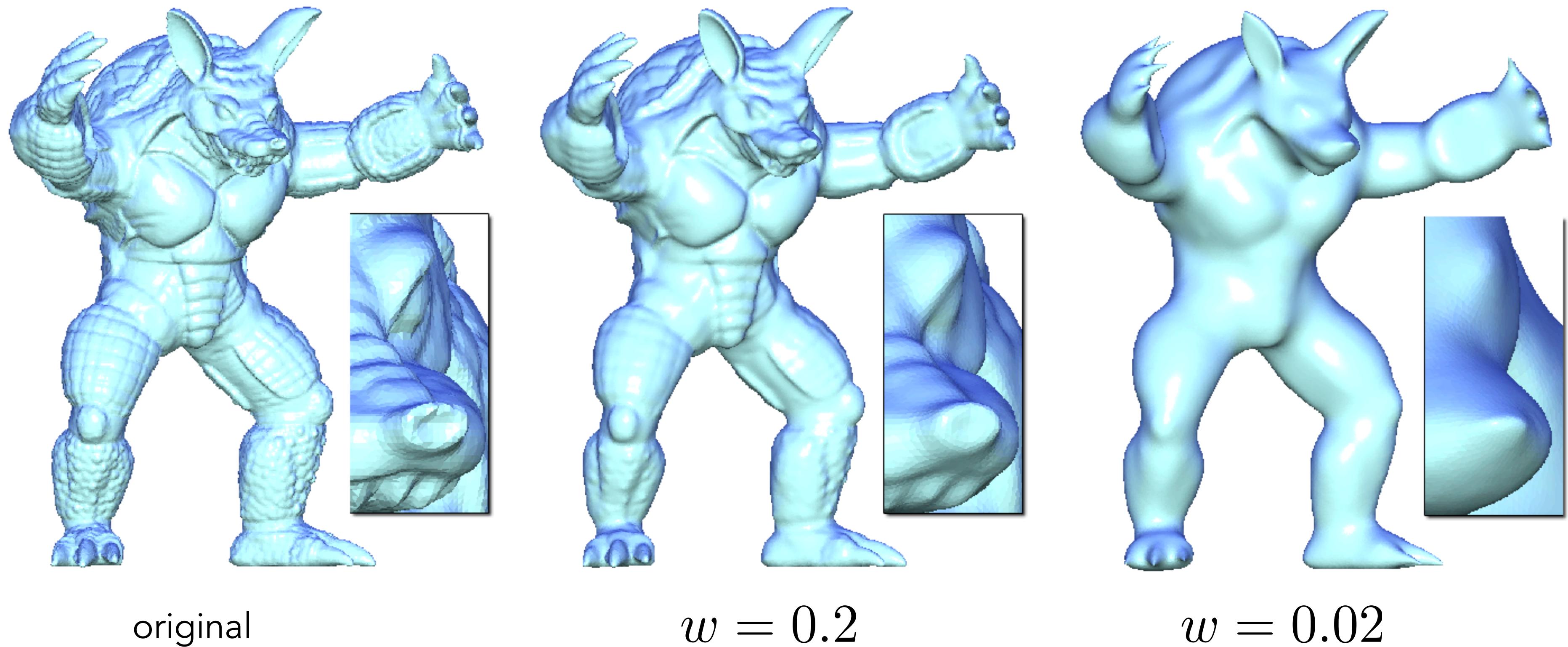
$$\Rightarrow \underline{(L^T M L + wM)} \tilde{\mathbf{p}} = wM\mathbf{p}$$

compare with implicit Euler!

NB :

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

Results



Customize the energy functional

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

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

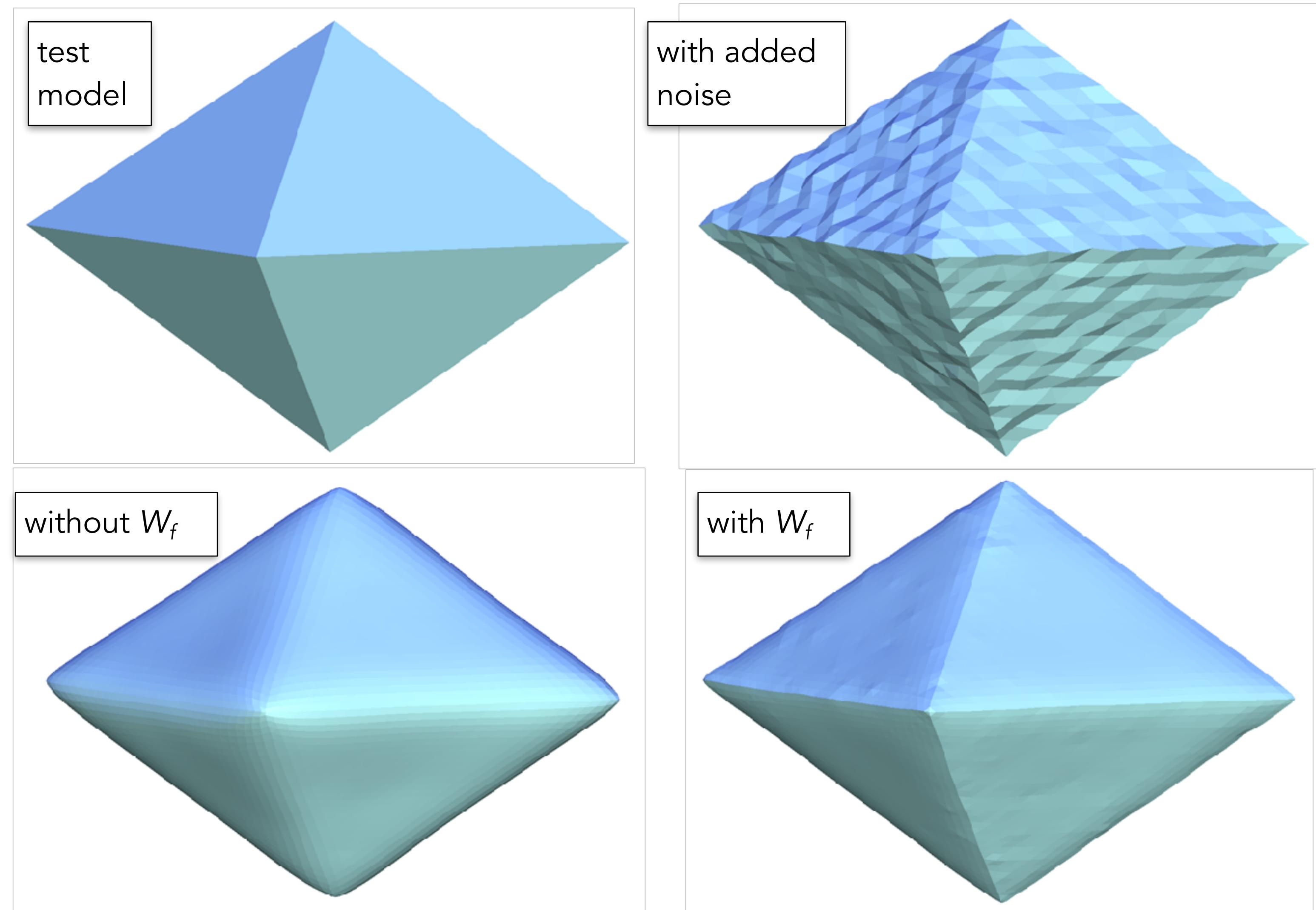
(slight abuse of notation, the energy is
actually the sum of the xyz components, $E(\tilde{\mathbf{p}}) = \sum_{\mathbf{v} \in \{\mathbf{x}, \mathbf{y}, \mathbf{z}\}} \tilde{\mathbf{v}}^T L^T \dots$
like on slide #62)



University
of Victoria

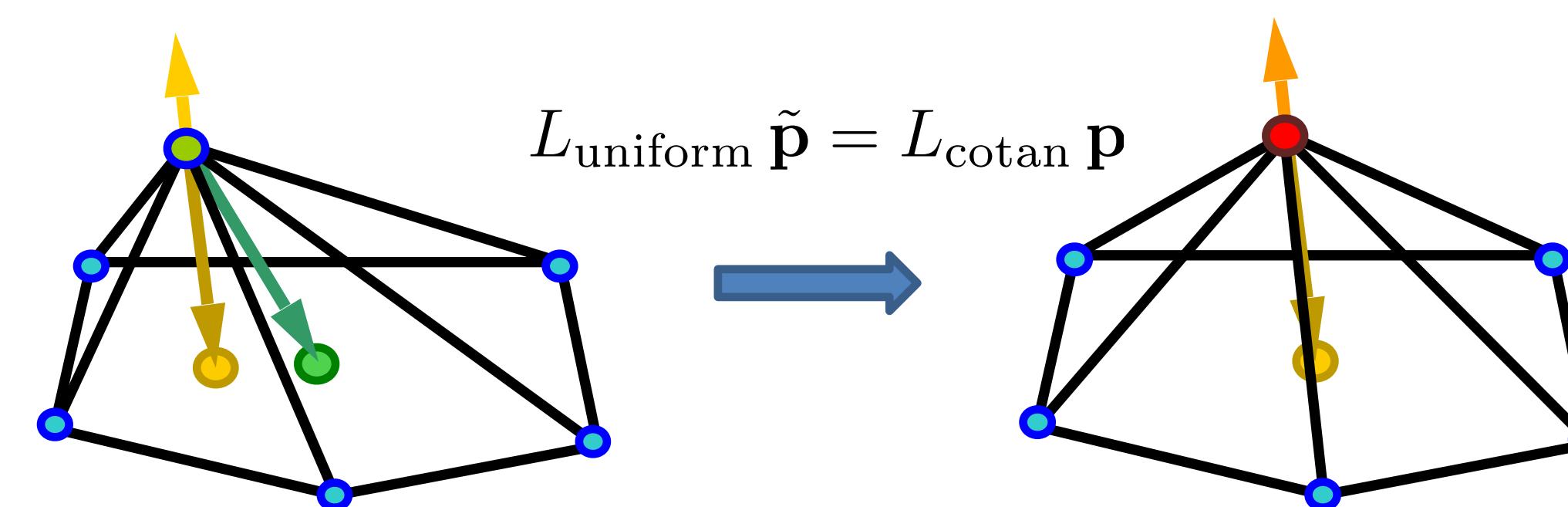
Computer Science

Using W_f



Customize the energy functional

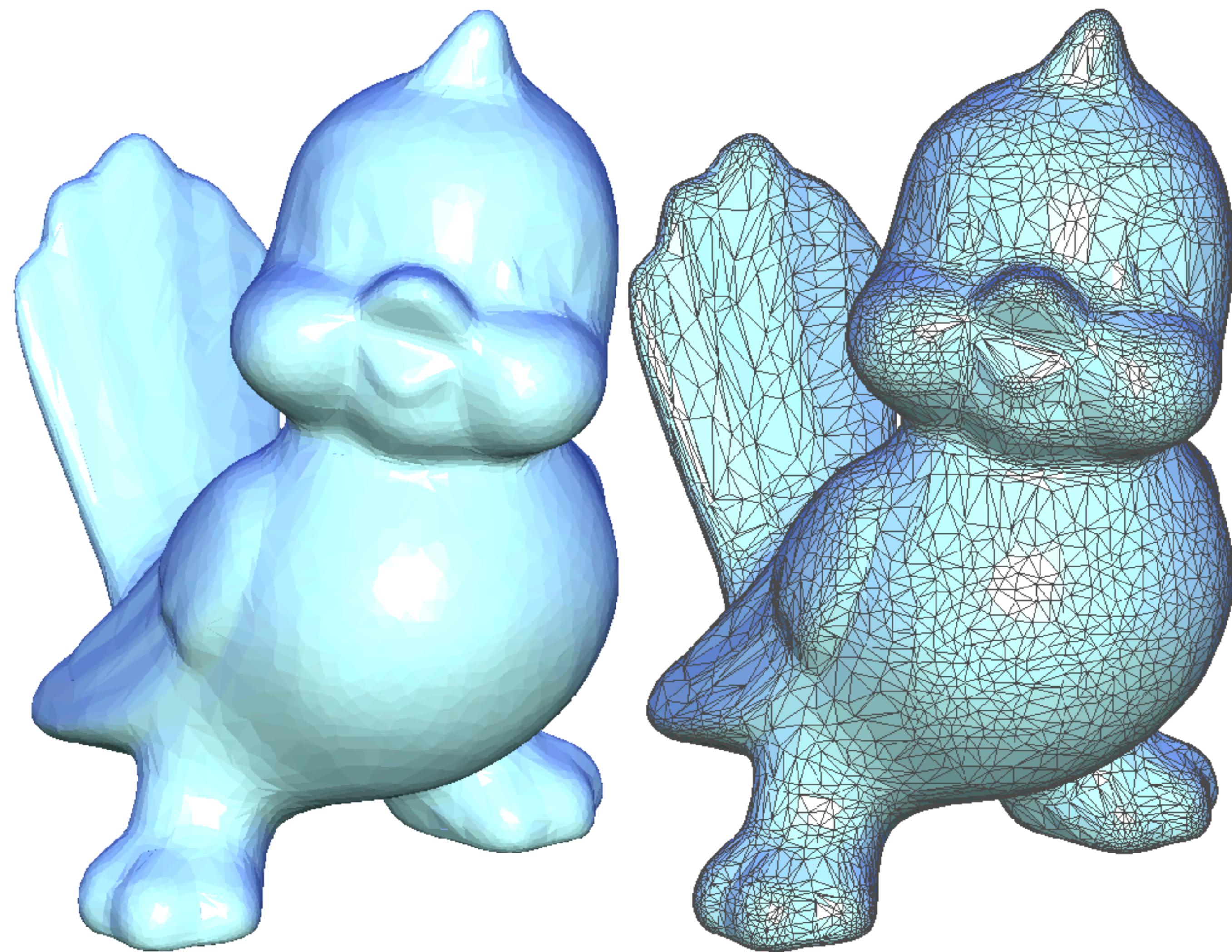
- Can do *tangential* smoothing!
 - Improve the shapes of mesh triangles without changing the surface shape



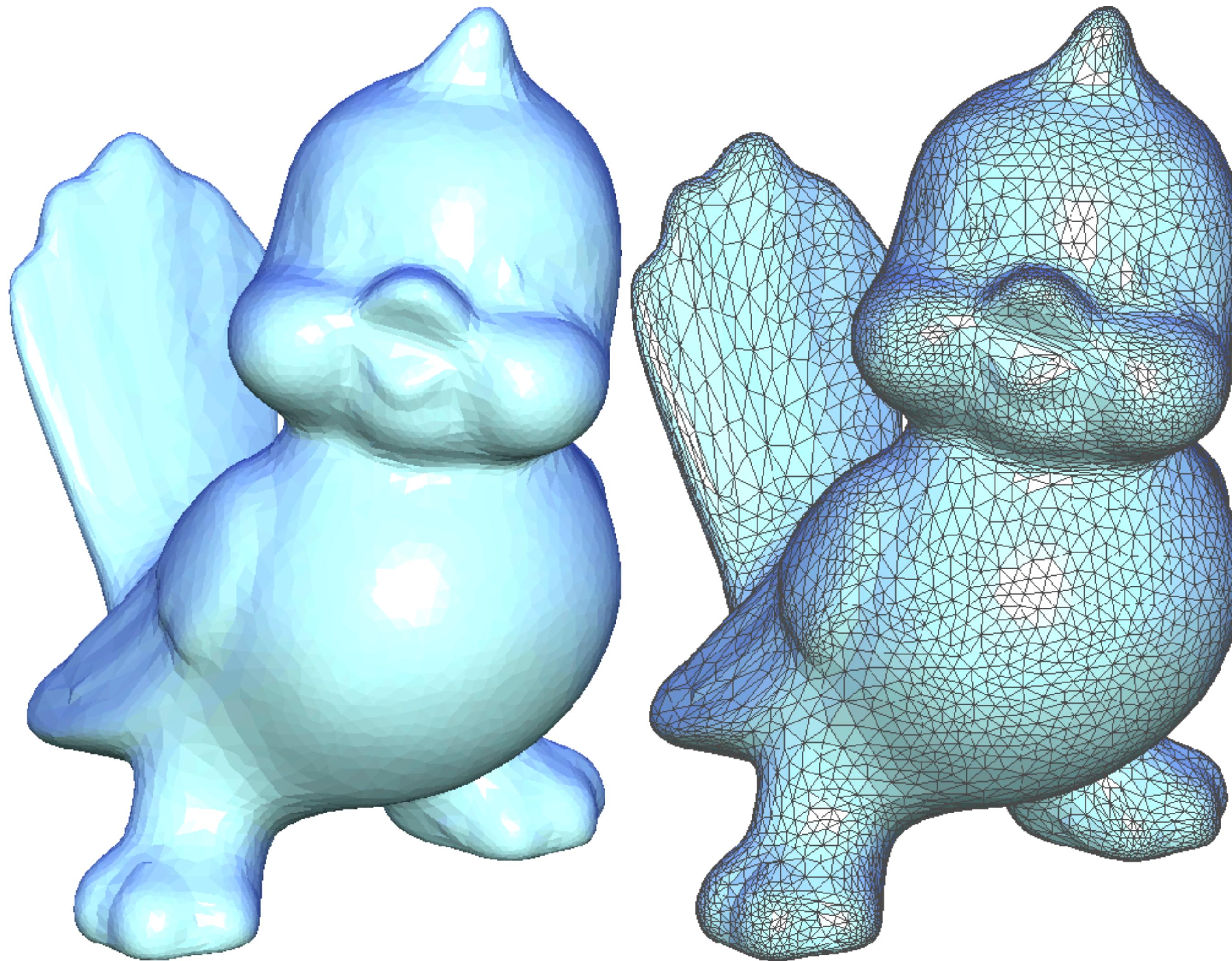
$$\min_{\tilde{\mathbf{p}}} \sum_{i=1}^n A_i (\|L_{\text{uniform}} \tilde{\mathbf{p}}_i - L_{\text{cotan}} \mathbf{p}_i\|^2 + w \|\tilde{\mathbf{p}}_i - \mathbf{p}_i\|^2)$$

<https://graphics.uni-bielefeld.de/downloads/publications/2004/sgp04.pdf>

Original



Triangle Shape Optimization



Method 3: Smoothing as Filtering

Fourier analysis

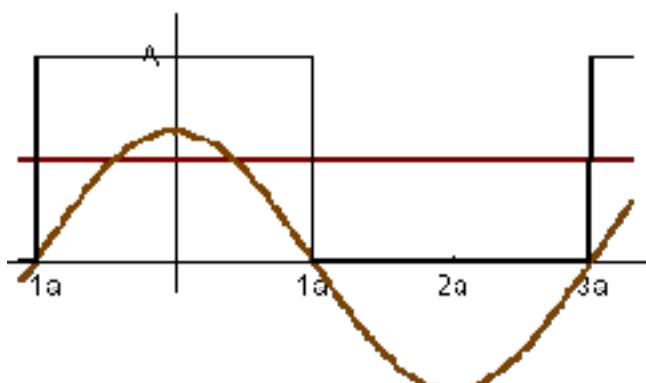
- Represent a function as a weighted sum of sines and cosines (basis functions)



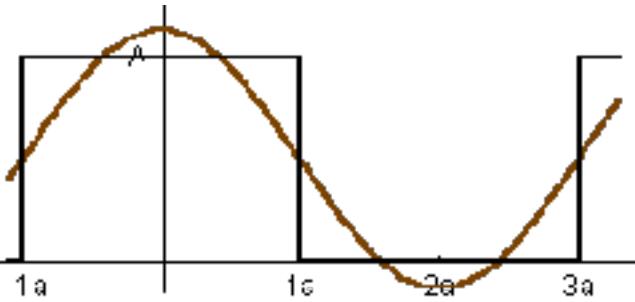
Joseph Fourier 1768 - 1830

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

basis functions



weighted sum



Fourier analysis

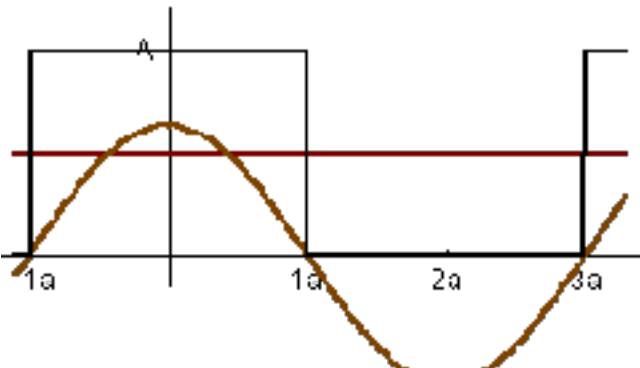
- Represent a function as a weighted sum of sines and cosines (basis functions)



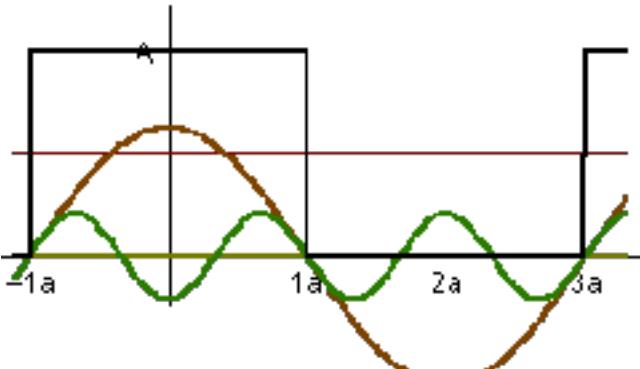
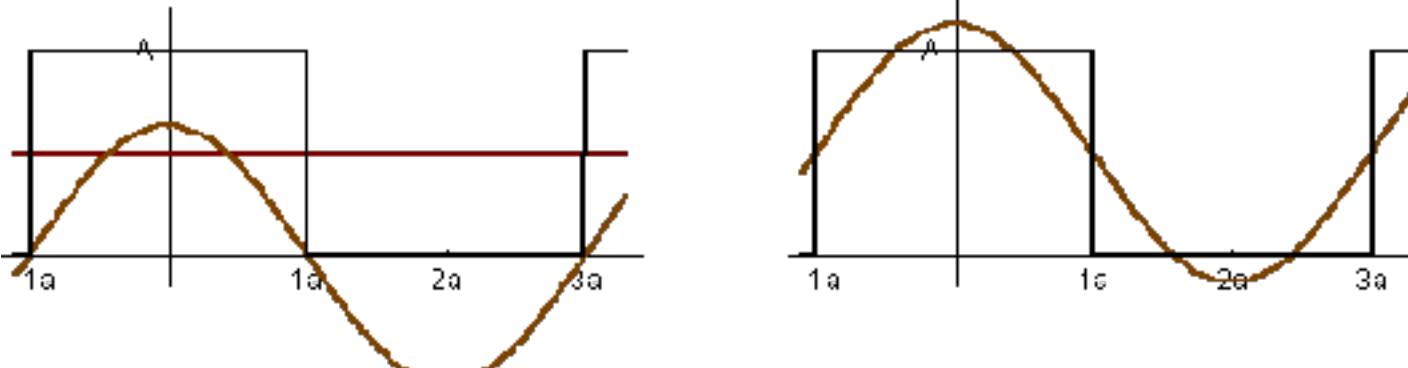
Joseph Fourier 1768 - 1830

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

basis functions



weighted sum



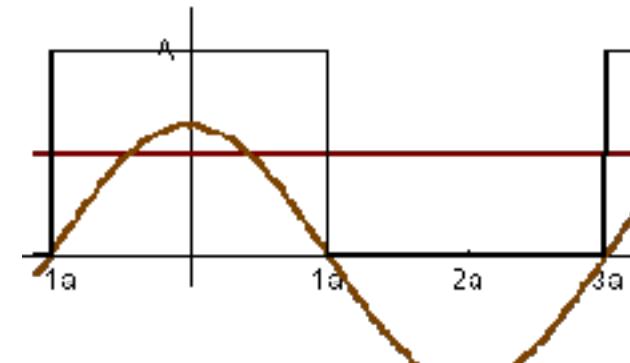
Fourier analysis

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

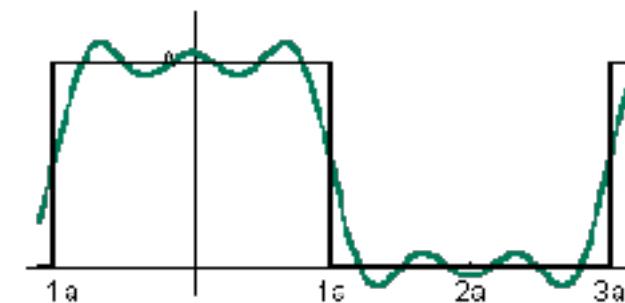
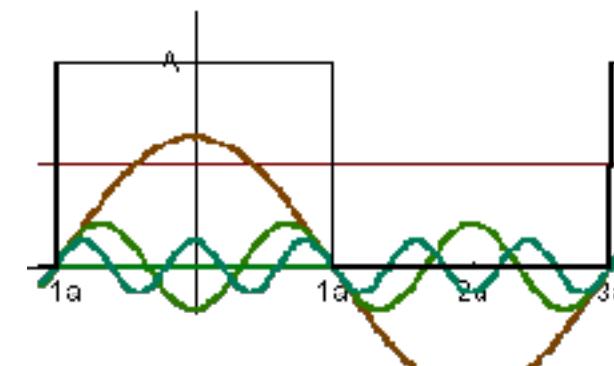
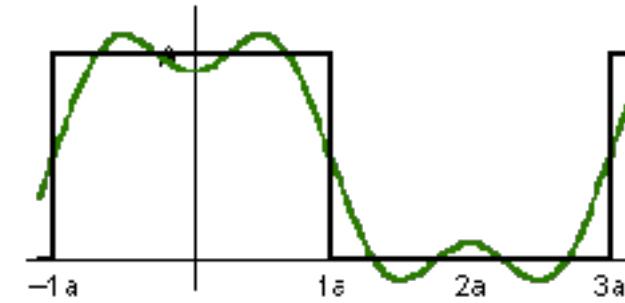
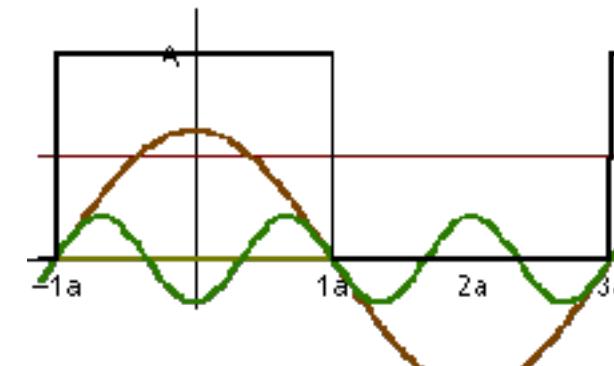
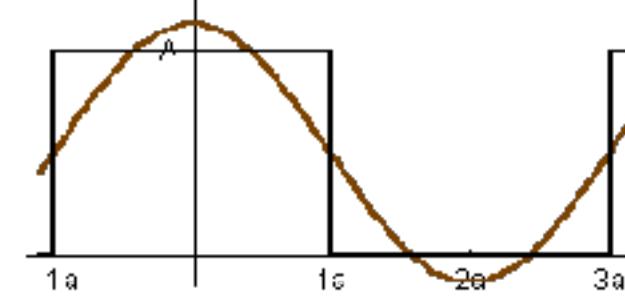


Joseph Fourier 1768 - 1830

basis functions



weighted sum



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

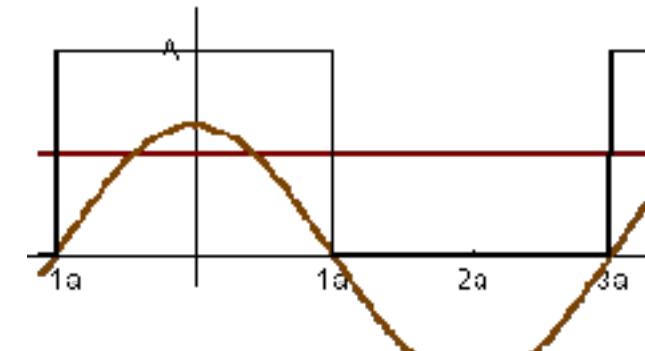
Fourier analysis

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

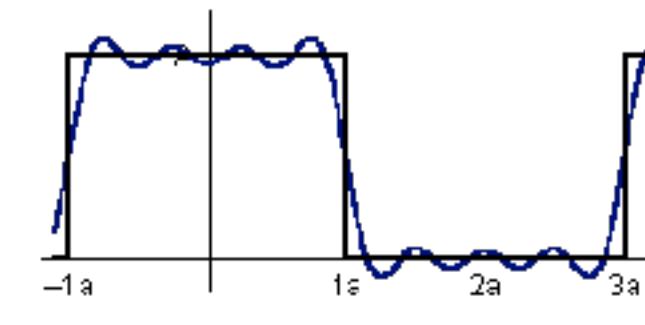
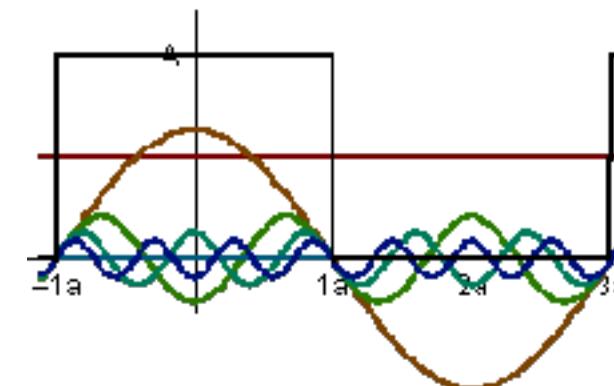
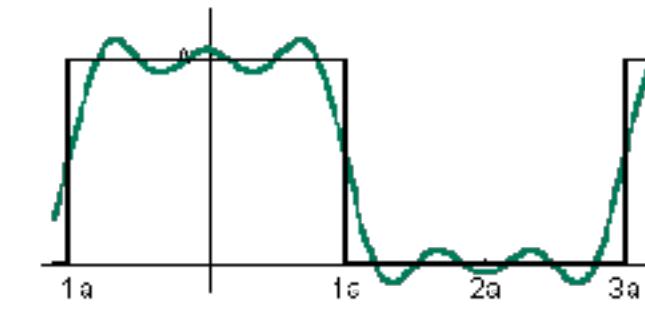
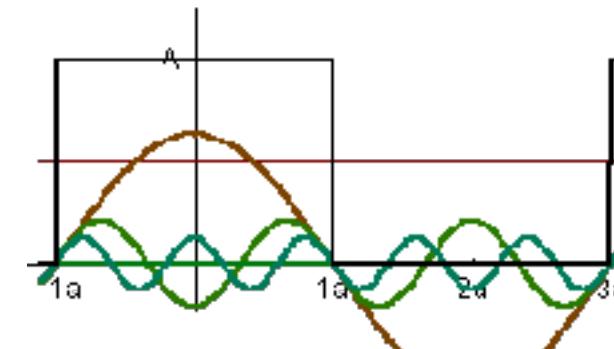
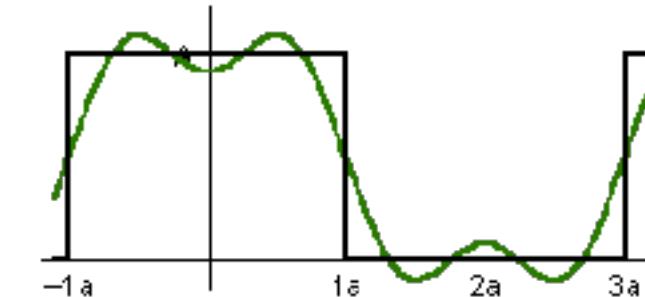
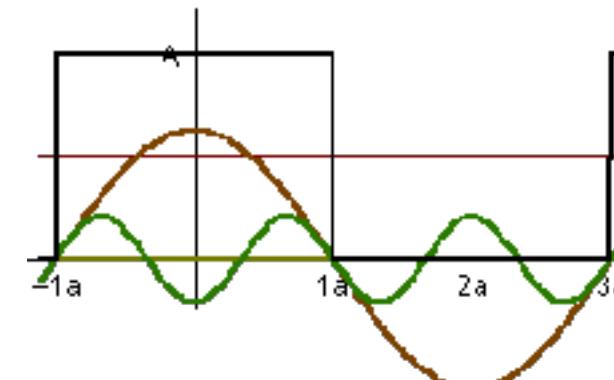
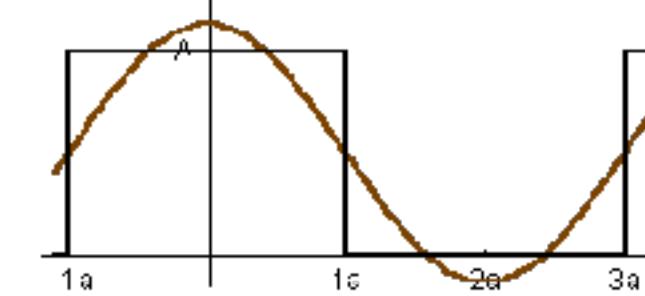


Joseph Fourier 1768 - 1830

basis functions



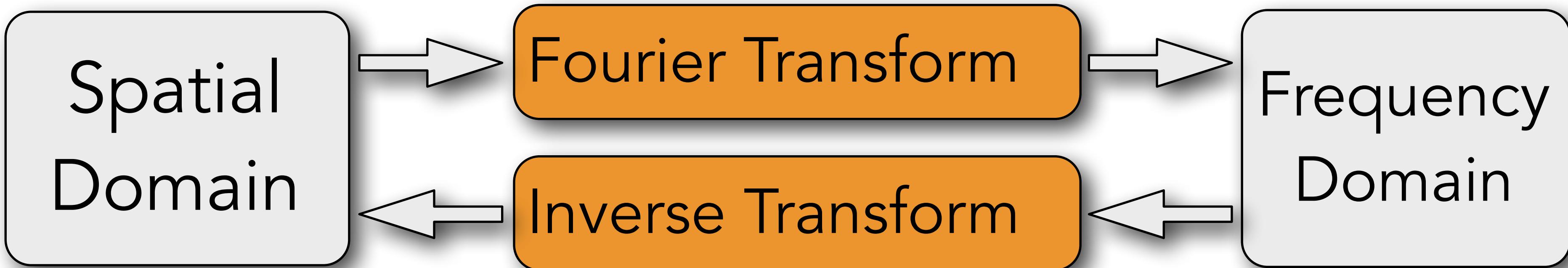
weighted sum



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

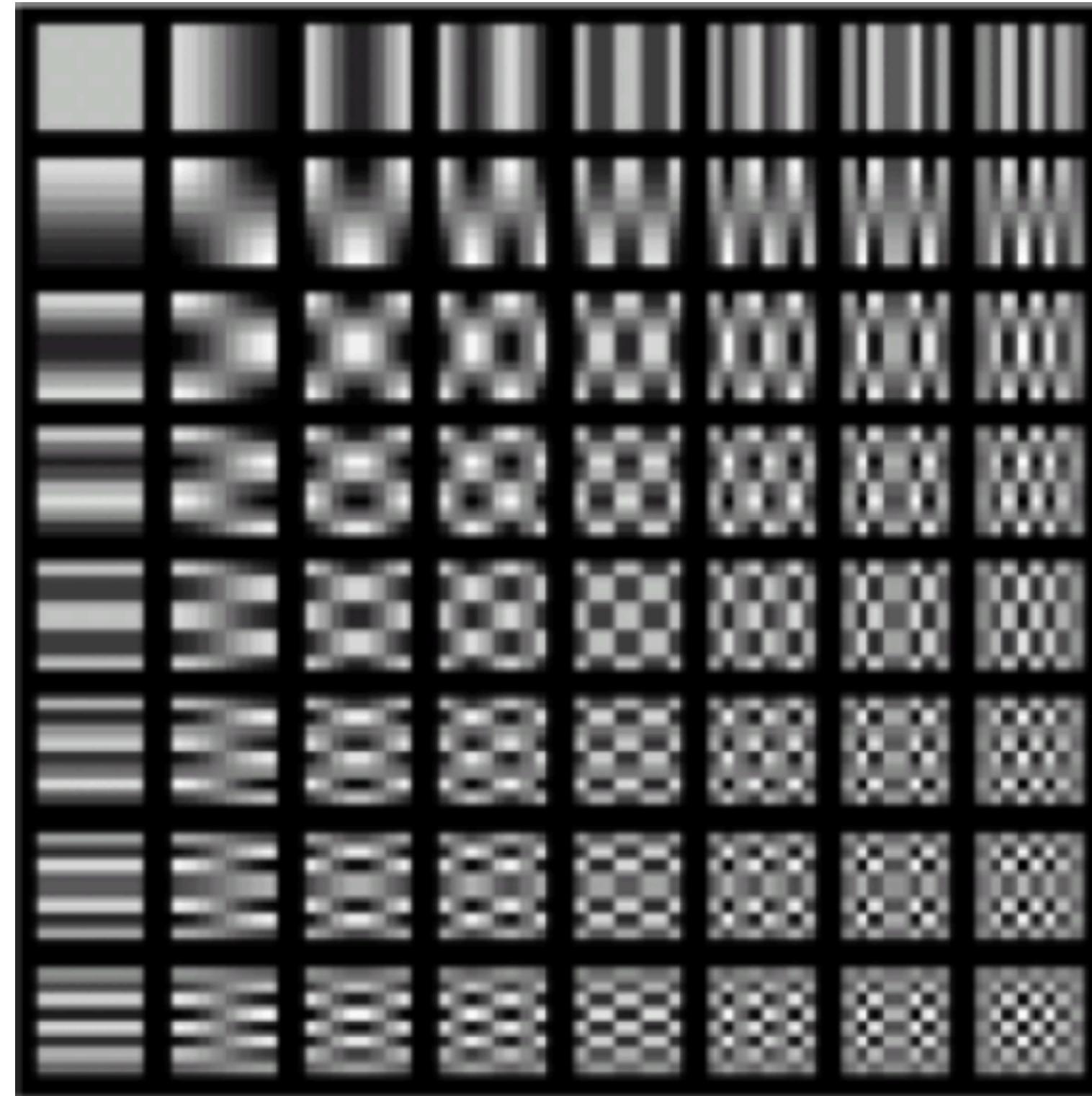
Fourier analysis

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



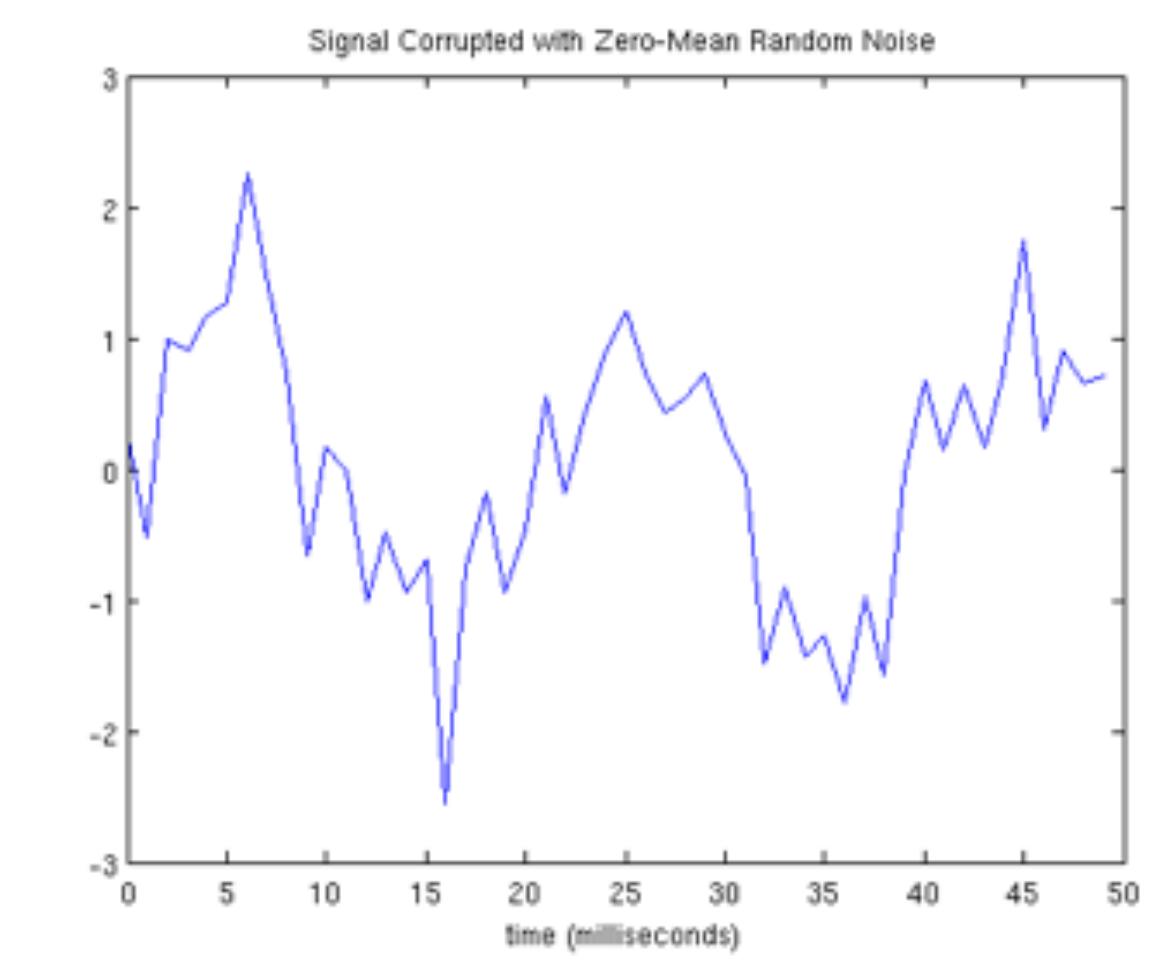
$$f(x) = \int_{-\infty}^{\infty} F(\omega)e^{2\pi i \omega x} d\omega$$

Also works on rectangular 2D domains

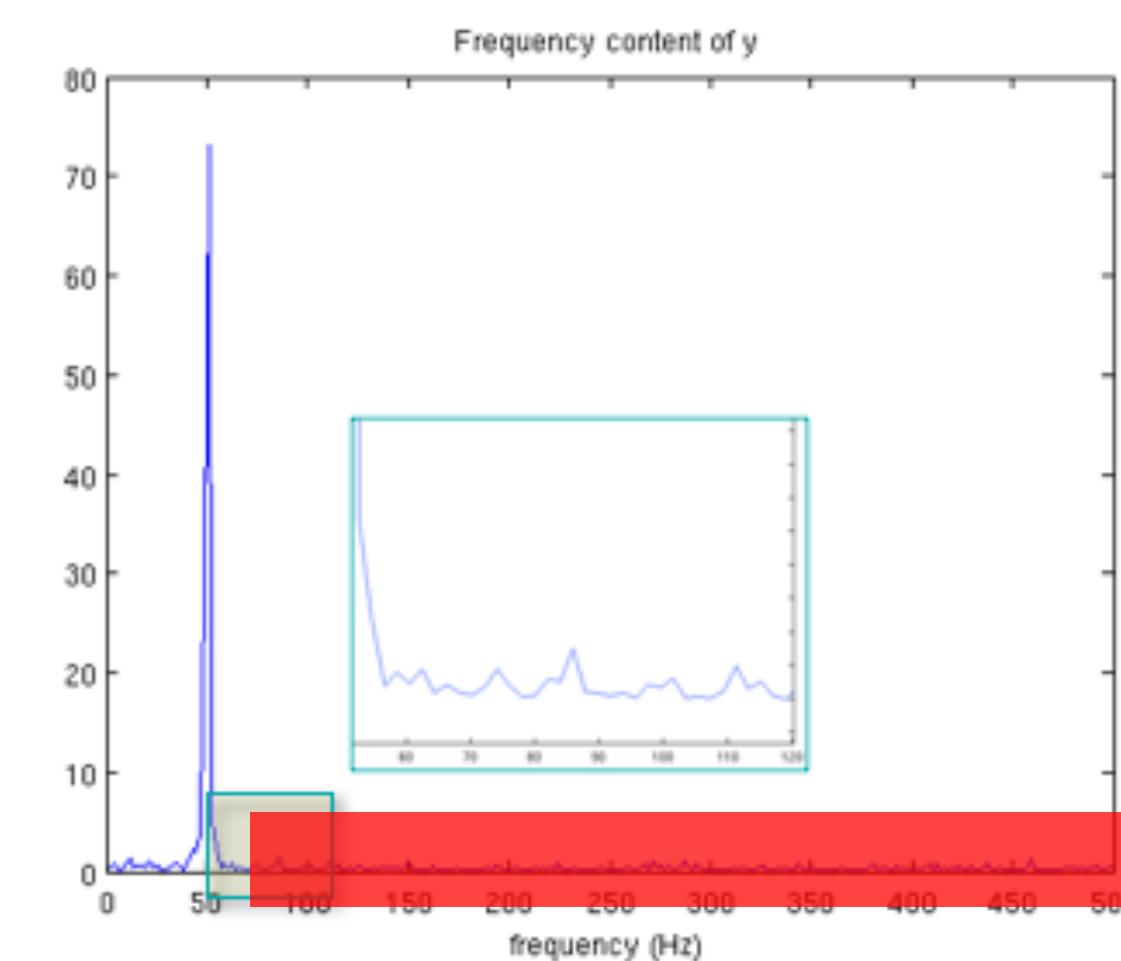
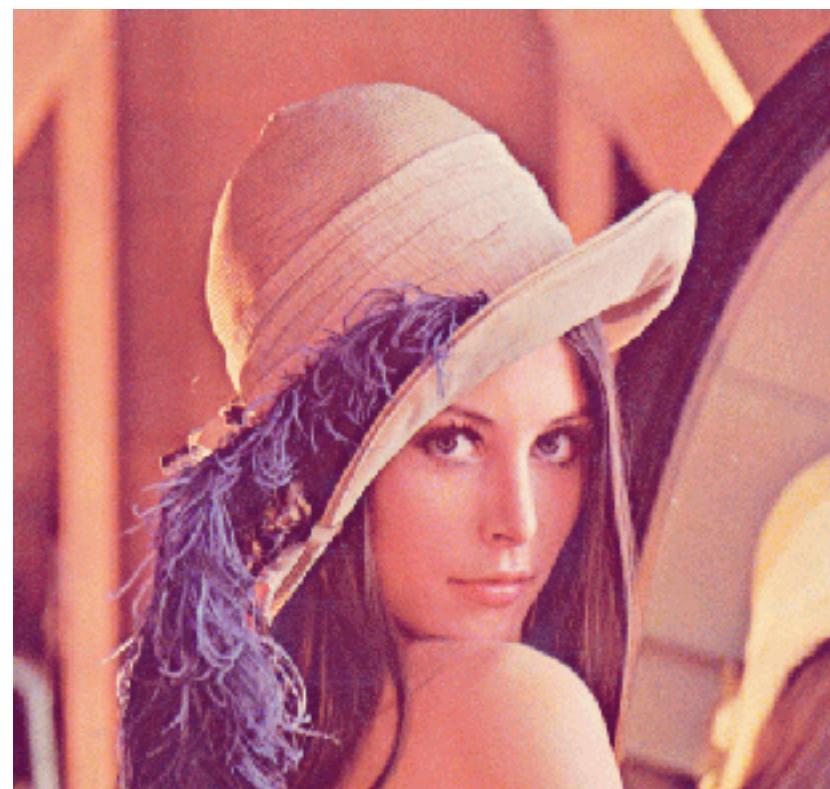


Fourier (DCT) basis functions for 8x8 grayscale images
 $\cos(2\pi\omega_h) \cos(2\pi\omega_v)$

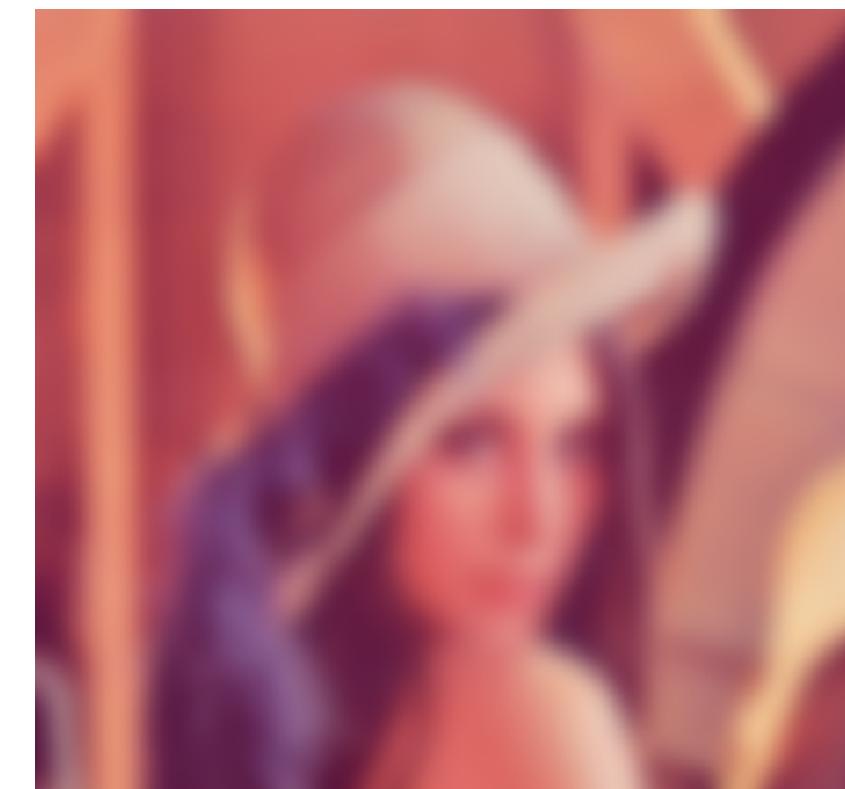
Smoothing = filtering high frequencies out



spatial domain



frequency domain



Extend Fourier to meshes?

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

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

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

Spectral analysis on meshes

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

$$\mathbf{x} = [x_1, \dots, x_n]^T$$

$$\mathbf{y} = [y_1, \dots, y_n]^T$$

$$\mathbf{z} = [z_1, \dots, z_n]^T$$

$$\tilde{\mathbf{x}} = \sum_{i=1}^k (\mathbf{x}^T \mathbf{e}_i) \mathbf{e}_i$$

$$\tilde{\mathbf{y}} = \sum_{i=1}^k (\mathbf{y}^T \mathbf{e}_i) \mathbf{e}_i$$

$$\tilde{\mathbf{z}} = \sum_{i=1}^k (\mathbf{z}^T \mathbf{e}_i) \mathbf{e}_i$$

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

Spectral analysis on meshes

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



too expensive for
large meshes

References

- Polygon Mesh Processing, Chapter 4