09 - Smoothing and Fairing

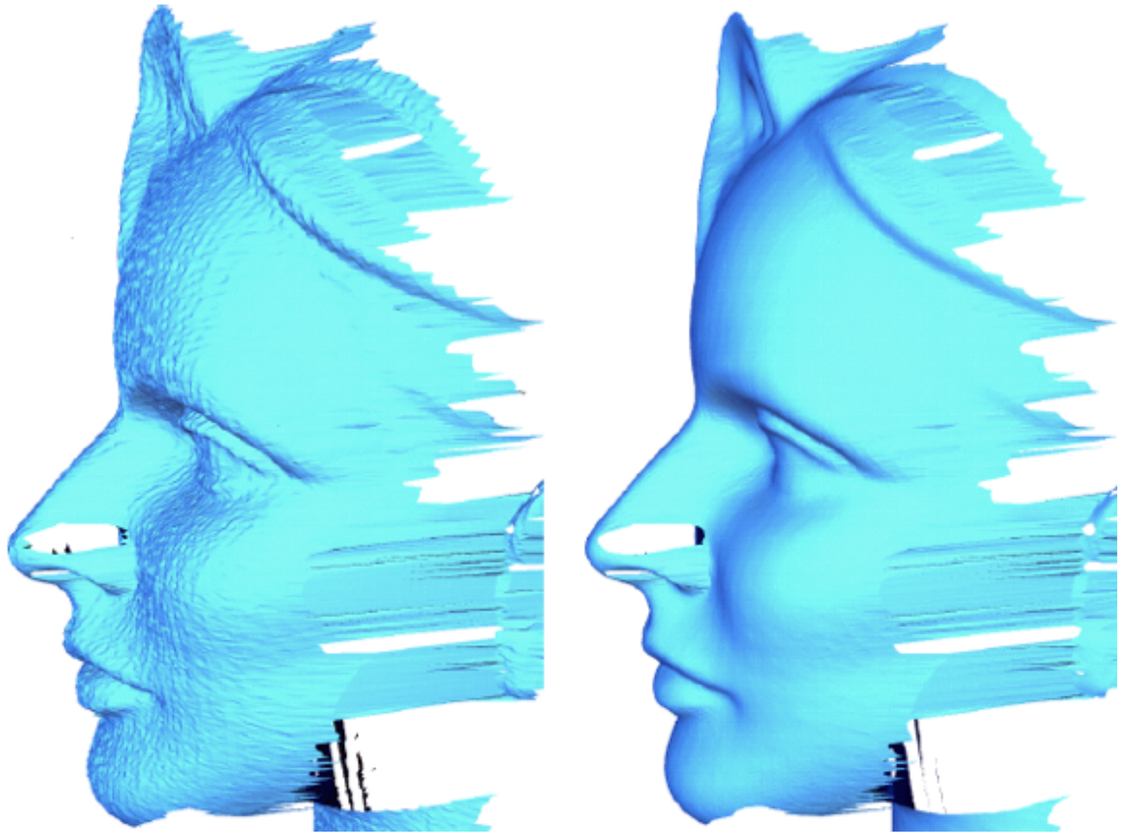
Acknowledgements: Daniele Panozzo

In this lecture

- First problem in geometry processing
- Application of Laplace-Beltrami operator
- Laplace equation: PDE with a linear solution
  - solution of sparse linear systems

Surface Smoothing – Motivation

- Scanned surfaces can be noisy
- Noise produces artefacts at high frequency
- Removing noise makes the surface smoother
- Details at a scale smaller than noise are smoothed out

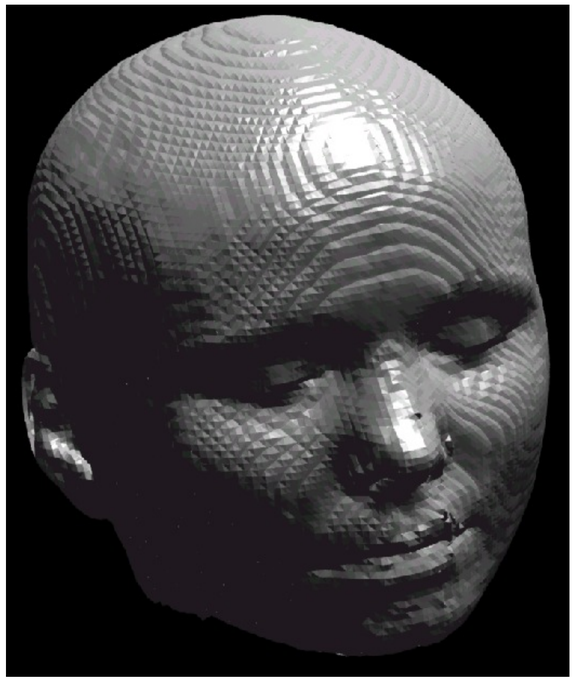


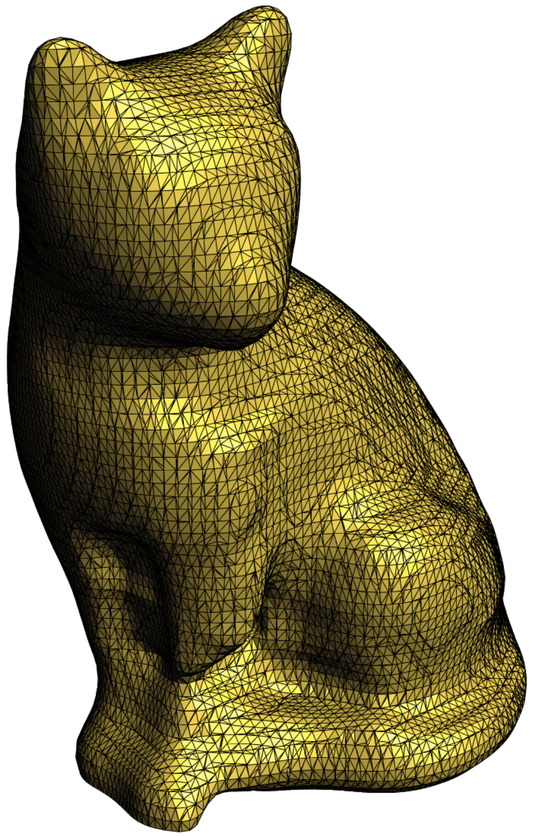
Surface Smoothing – Motivation

- Scanned surfaces can be noisy…

Surface Smoothing – Motivation

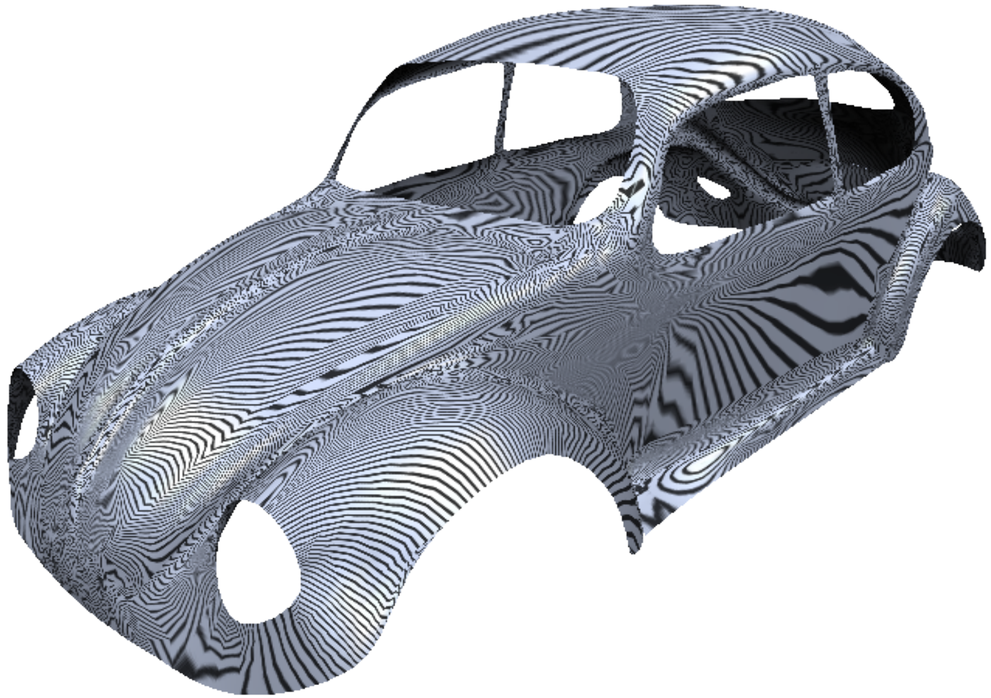
- Marching Cubes meshes can be ugly





- Why is the left mesh ugly?
- Why is the right mesh ugly?
  - What is the problem with such triangles?

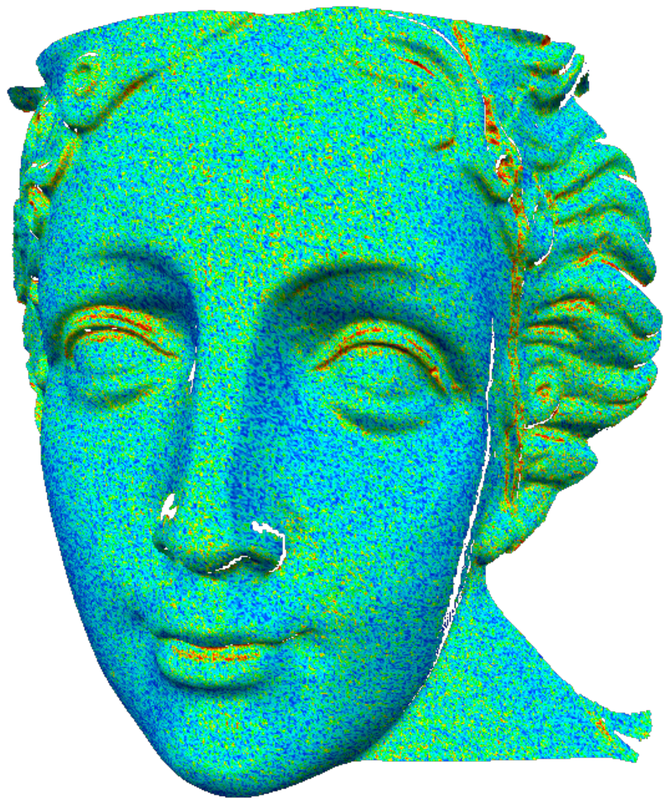
Visual Assessment of Surface Smoothness



How to measure smoothness?

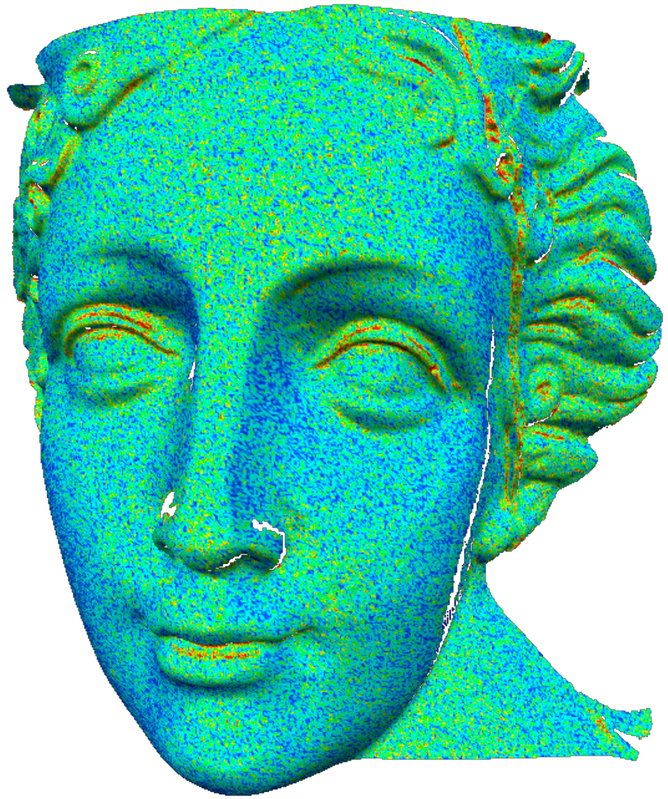
Curvature and Smoothness

Curvature and Smoothness



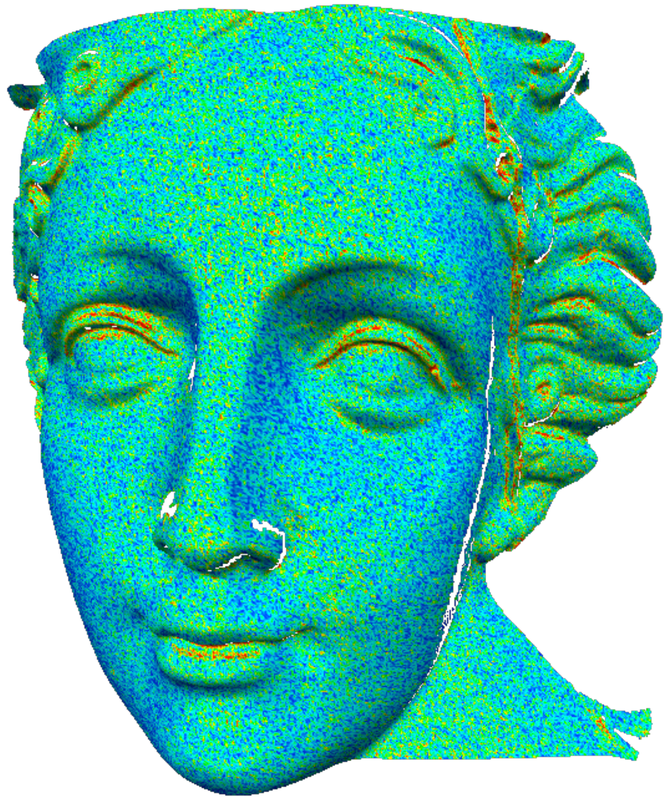
mean curvature plot

Curvature and Smoothness

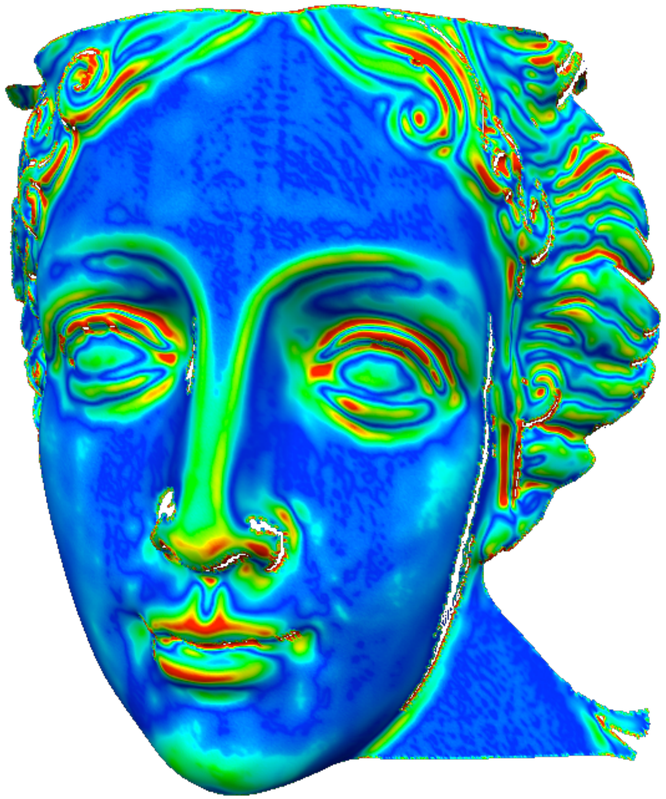


mean curvature plot

Curvature and Smoothness



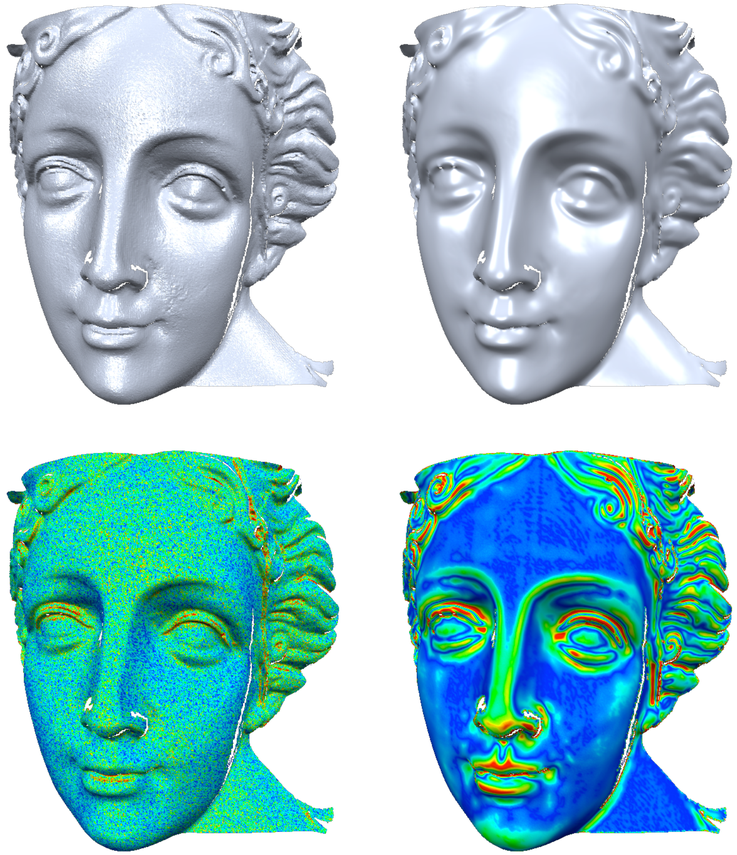
mean curvature plot



mean curvature plot
of smoothed model

Curvature and Smoothness

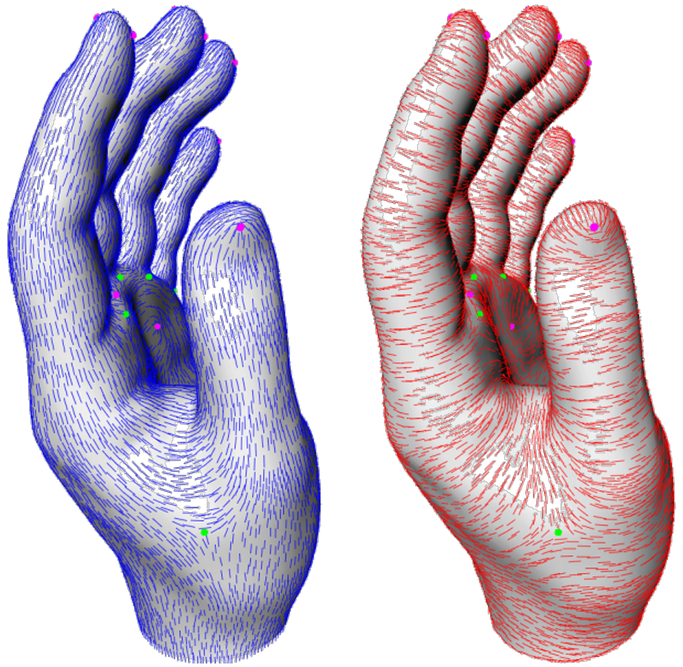
- Smoothing =
  reducing curvature?
- Smoothing =
  make curvature vary less?



Which curvature?

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

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



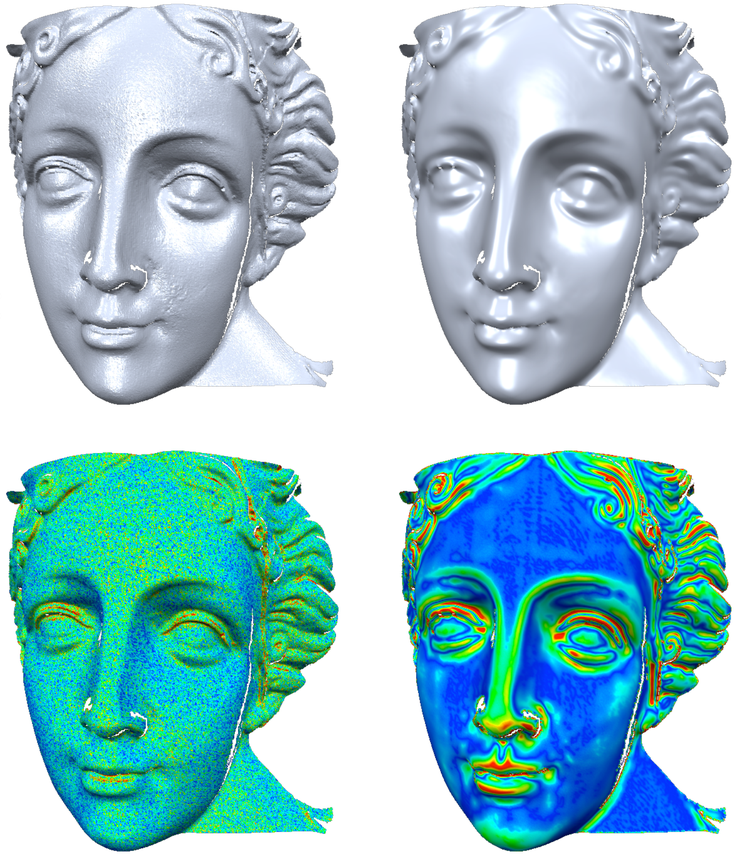
principal directions

Which curvature?

- Principal curvatures
  - Nonlinear and “discontinuous” operator in the definition (min, max)
- Gauss curvature K
  - Intrinsic-only, insensitive to embedding in 3D space

Which curvature?

- Principal curvatures
  - Nonlinear and “discontinuous” operator in the definition (min, max)
- Gauss curvature K
  - Intrinsic-only, insensitive to embedding in 3D space
- Mean curvature H
  - Relatively simple to extract on meshes via Laplace-Beltrami:



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

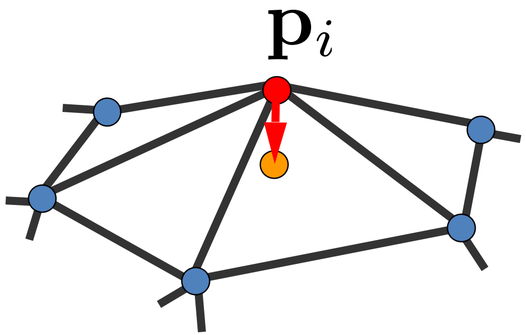
goal: $H = 0$ or $H = \text{const}$

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

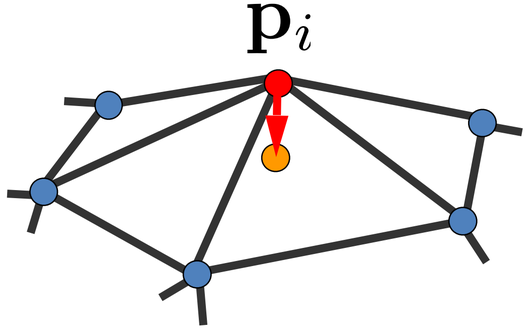


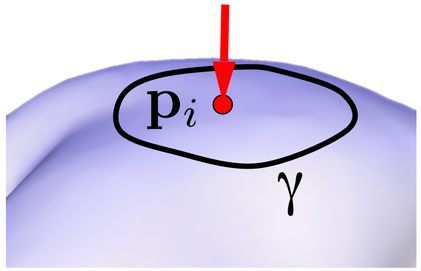
$$\Delta_{\mathcal{M}}(\mathbf{p}_i) = \delta_i = \frac{1}{W_i} \sum_{j \in \mathcal{N}(i)} w_{ij}(\mathbf{p}_j - \mathbf{p}_i)$$

Recap: Laplace-Beltrami

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

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





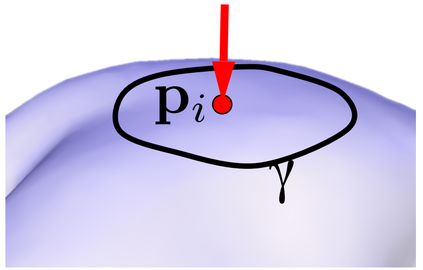
$$\delta_i = \frac{1}{W_i} \sum_{j \in \mathcal{N}(i)} w_{ij} (\mathbf{p}_j - \mathbf{p}_i) \qquad \delta_i = \frac{1}{\text{len}(\gamma)} \int_\gamma (\gamma(s) - \mathbf{p}_i)\, ds$$

$$\lim_{\text{len}(\gamma) \to 0} \frac{1}{\text{len}(\gamma)} \int_\gamma (\gamma(s) - \mathbf{p}_i)\, ds = -2H(\mathbf{p}_i)\, \mathbf{n}_i$$

L-B: Weighting Schemes

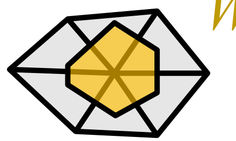
$$\delta_i = \frac{1}{W_i} \sum_{j \in \mathcal{N}(i)} w_{ij} (\mathbf{p}_j - \mathbf{p}_i)$$

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

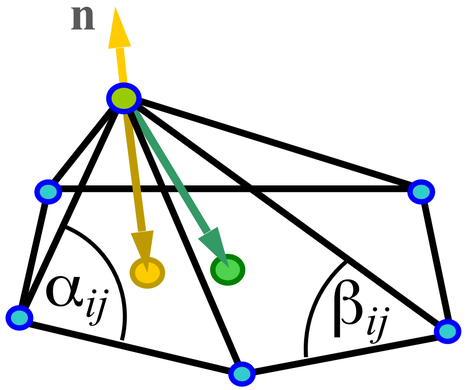


- Integrate over Voronoi region of the vertex

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

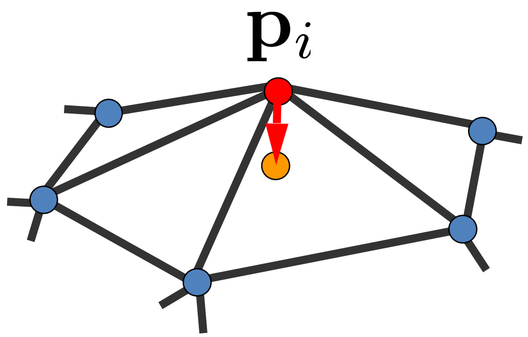


$W_i = A_i$

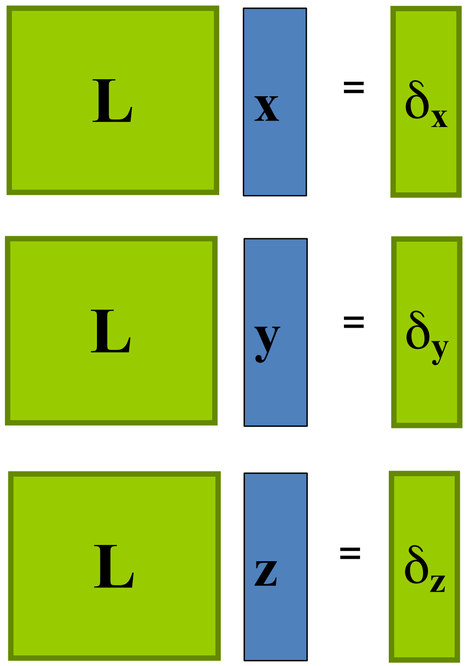


Laplacian Matrix

- The transition between xyz and δ is linear:



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



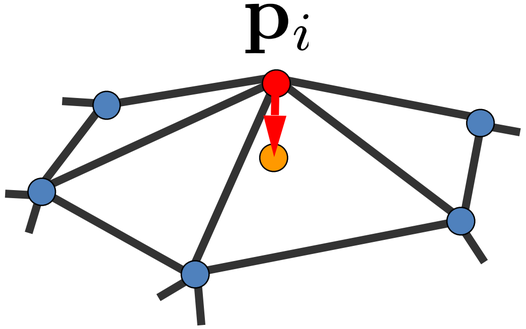
$$\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^n$$

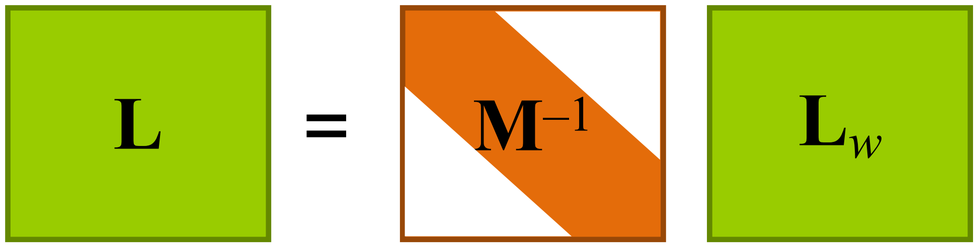
$$\mathbf{L} \in \mathbb{R}^{n \times n}$$

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

Laplacian Matrix

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





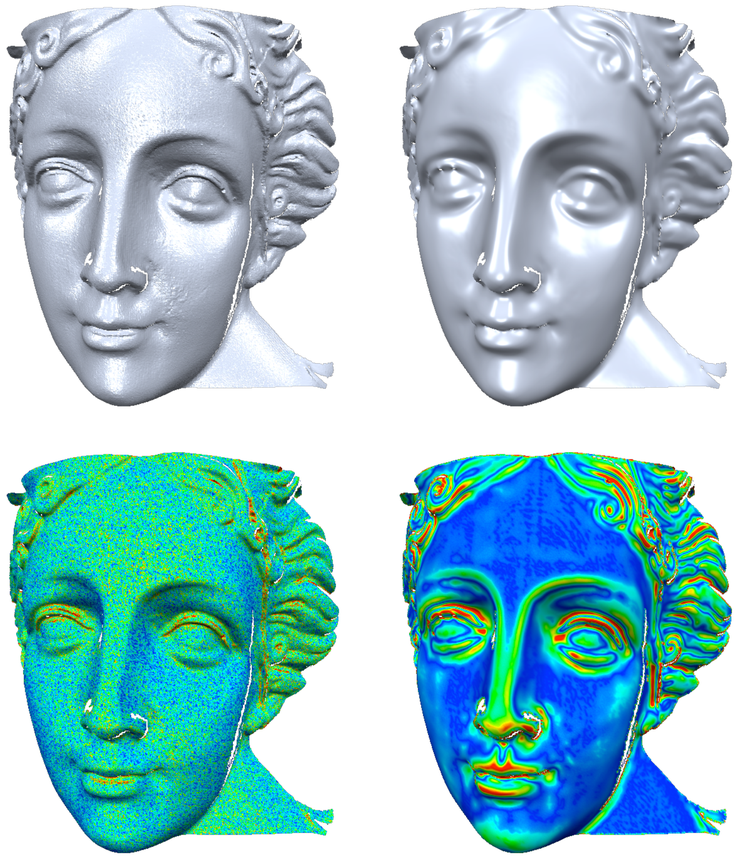
$$\delta_i = \frac{1}{W_i} \sum_{j \in \mathcal{N}(i)} w_{ij}(\mathbf{p}_j - \mathbf{p}_i)$$

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 does not define the surface
  - $\mathbf{n}$ non-linear in $\mathbf{p}$

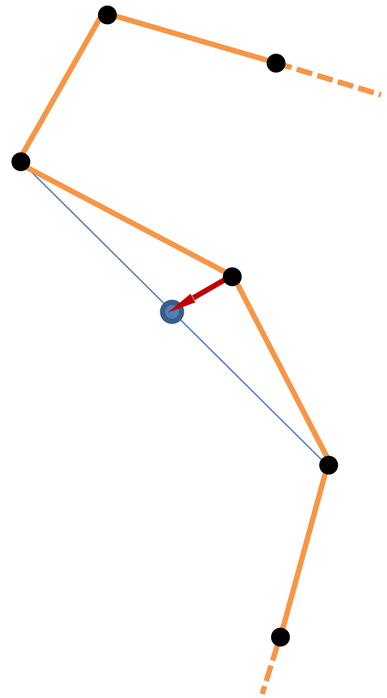


Smoothing by diffusion flow

Example – smoothing curves

- Laplace in 1D = second derivative:

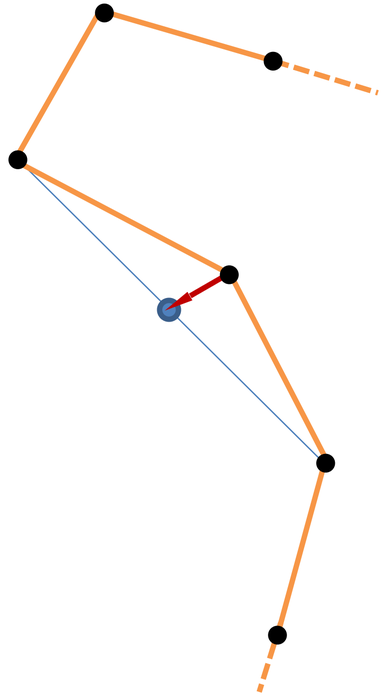
$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i)$$



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

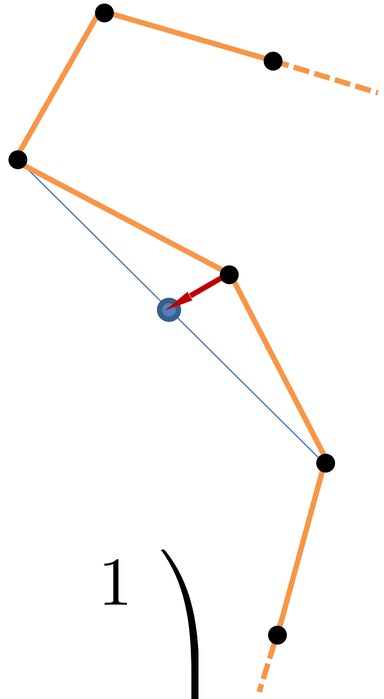
$$L\mathbf{p}$$

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

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

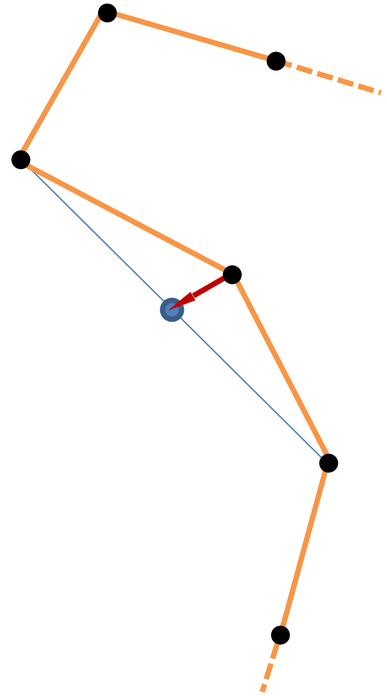
$$L\mathbf{p}$$

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

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:

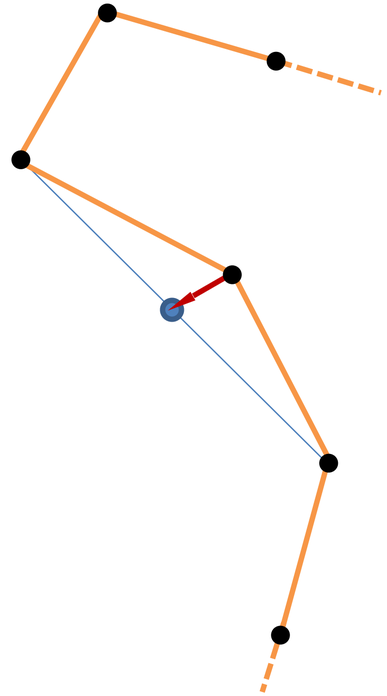
$$\tilde{\mathbf{p}} = \mathbf{p} + \lambda L \mathbf{p}, \quad \mathbf{p} \in \mathbb{R}^{n \times 2}$$

- Drawbacks?

Example – smoothing curves

- Flow to reduce curvature:

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

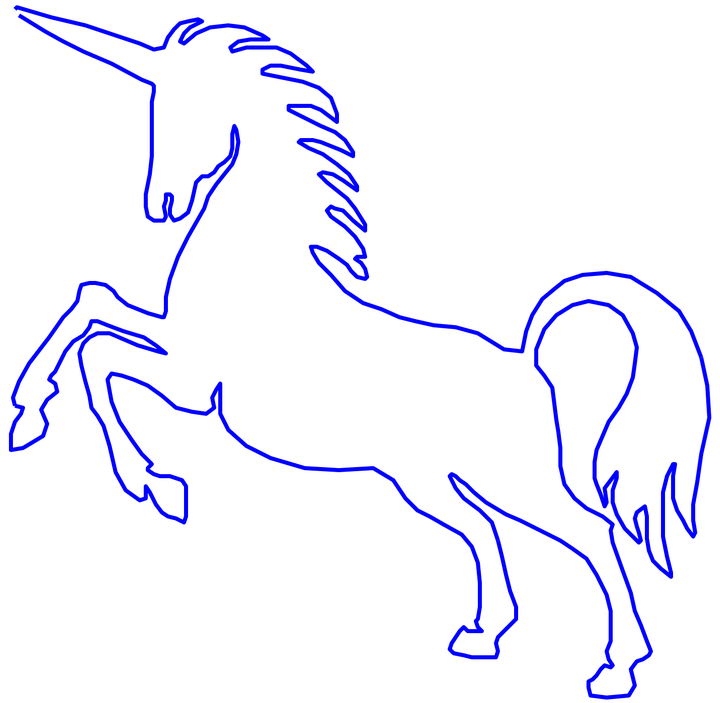


- Scale factor $0 < \lambda < 1$
- Matrix-vector form:

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

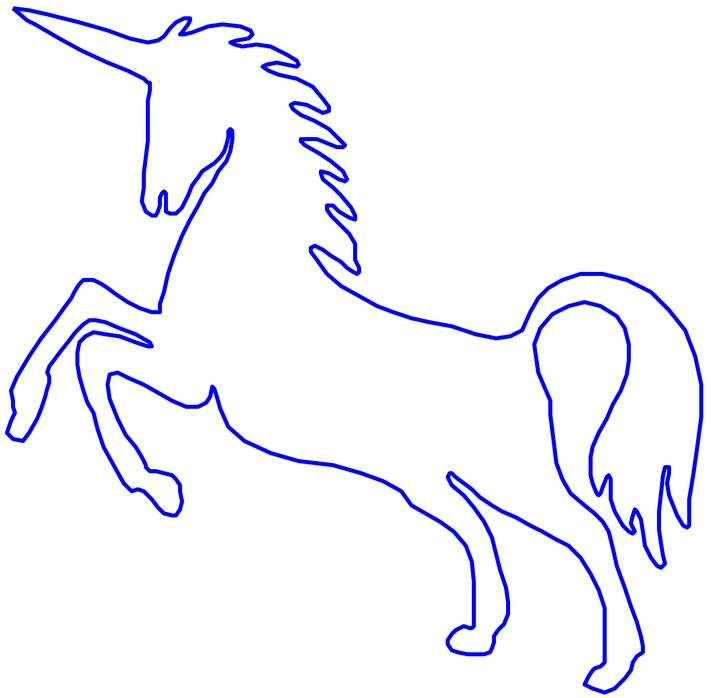
- May shrink the shape; can be slow

Filtering Curves



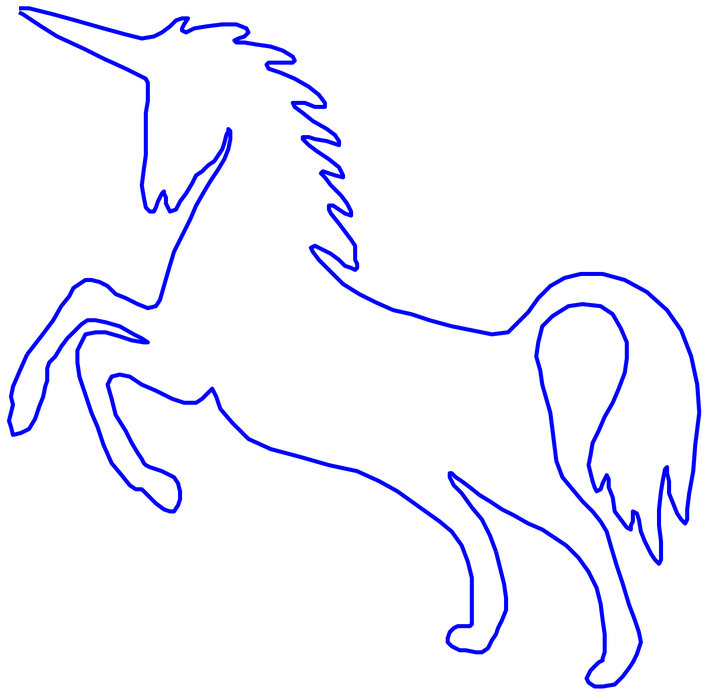
Original curve

Filtering Curves



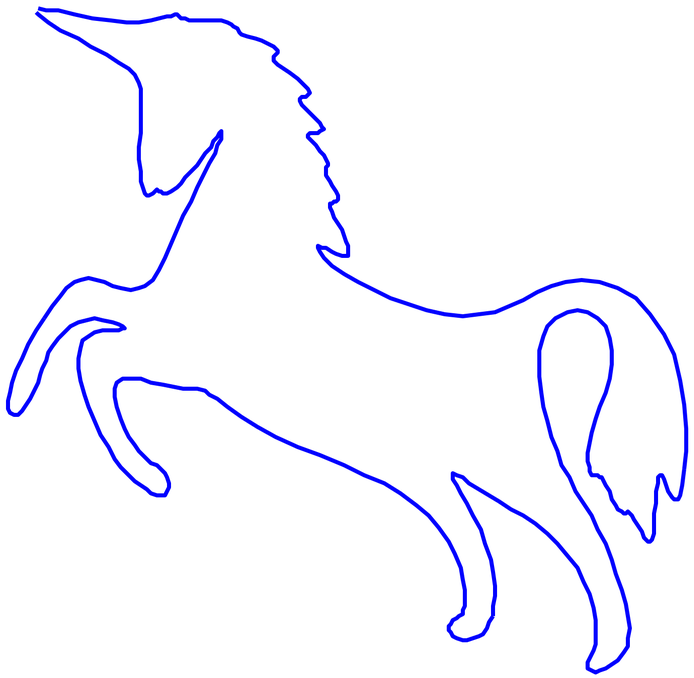
1st iteration; $\lambda=0.5$

Filtering Curves



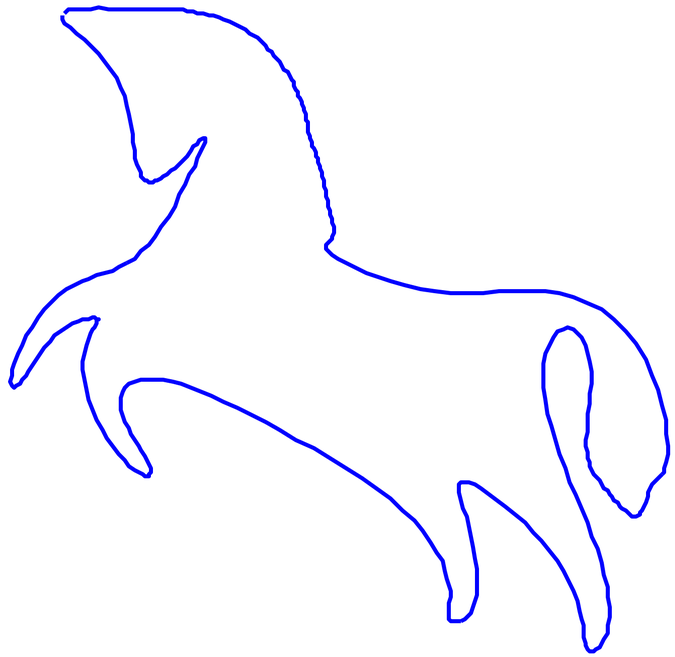
2nd iteration; $\lambda=0.5$

Filtering Curves



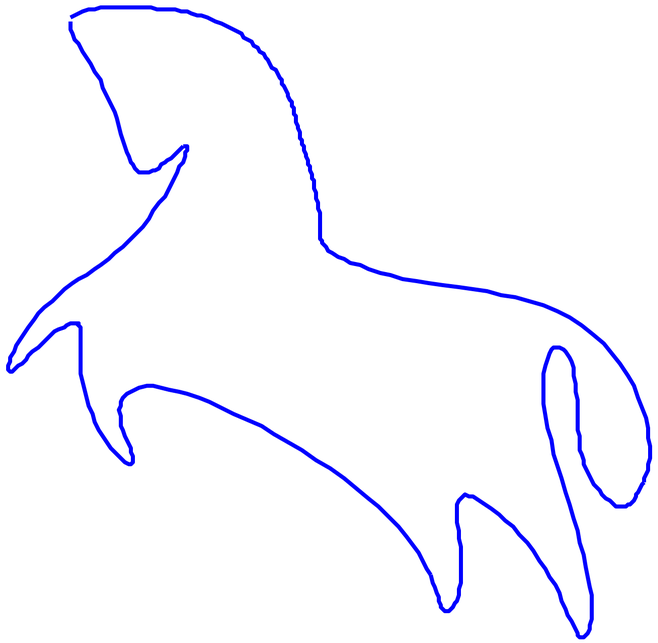
8th iteration; $\lambda=0.5$

Filtering Curves



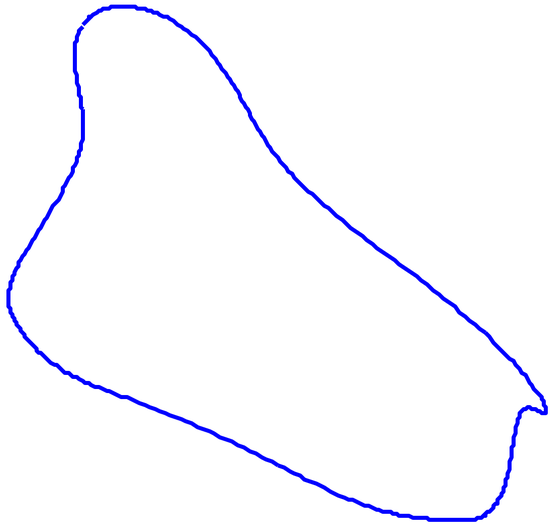
27th iteration; $\lambda=0.5$

Filtering Curves



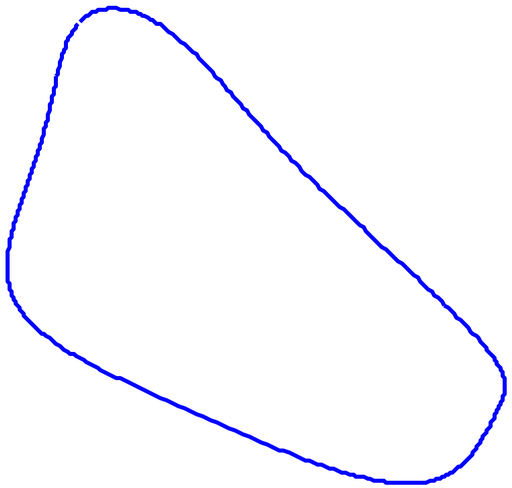
50th iteration; $\lambda=0.5$

Filtering Curves



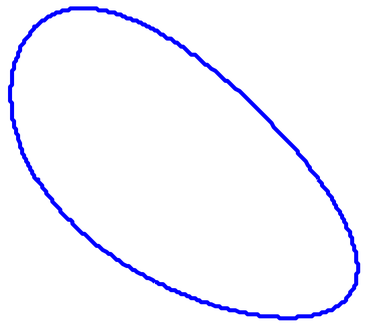
500th iteration; $\lambda=0.5$

Filtering Curves



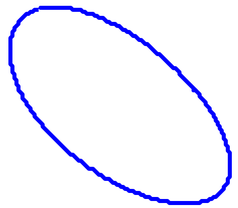
1000th iteration; $\lambda=0.5$

Filtering Curves



5000th iteration; $\lambda=0.5$

Filtering Curves



10000th iteration; $\lambda=0.5$

Filtering Curves

50000th iteration; $\lambda=0.5$

Diffusion flow - general scheme

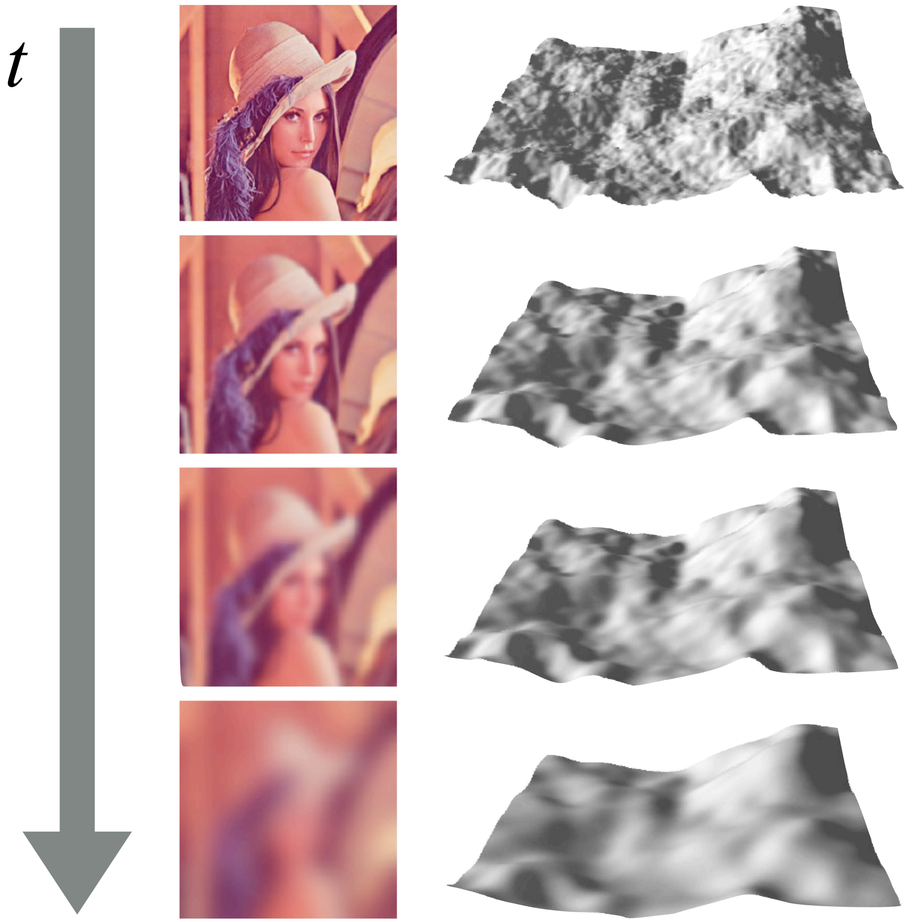
- We consider the *diffusion equation* (a.k.a. *heat equation*) describing the evolution of a signal over time

$$\frac{\partial f(\mathbf{x},t)}{\partial t} = \lambda \Delta f(\mathbf{x},t)$$

- A function f obeying to the above equation becomes smoother and smoother for increasing values of t
- $f(\mathrm{x},0)$ is the function at its initial state
- parameter λ sets the speed at which the function is smoothed

Diffusion flow - general scheme

- widely used to blur images and smooth terrain surfaces
- build a *scale space* describing the evolution of data through time under the blurring/ smoothing process



Diffusion flow - general scheme

- the diffusion equation is a Partial Differential Equation
- we discretize it both in space and in time:
  - sample f at mesh vertices: $\mathbf{f}(t) = (f(v_1, t), \ldots, f(v_n, t))^T$
  - divide time in discrete steps of uniform width h

$$\frac{\partial f(v_i, t)}{\partial t} \approx \frac{f(v_i, t+h) - f(v_i, t)}{h}$$

On meshes: smoothing as mean curvature flow

- Model smoothing as a diffusion process

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

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

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

- Discretize in time, forward differences:

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

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

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

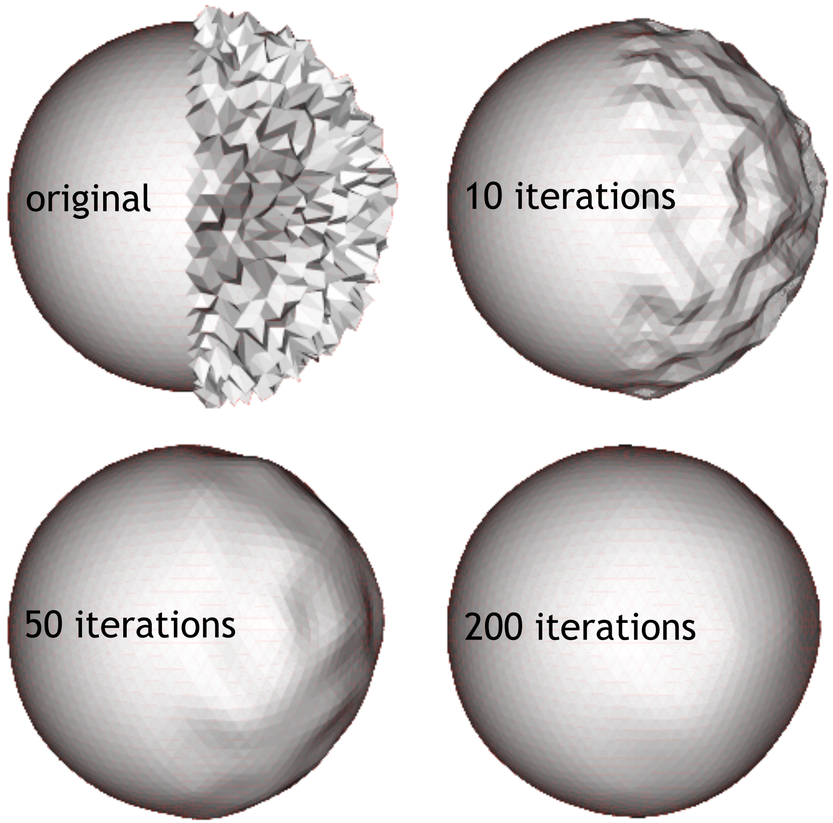
Explicit integration! Unstable unless time step *dt* is small

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



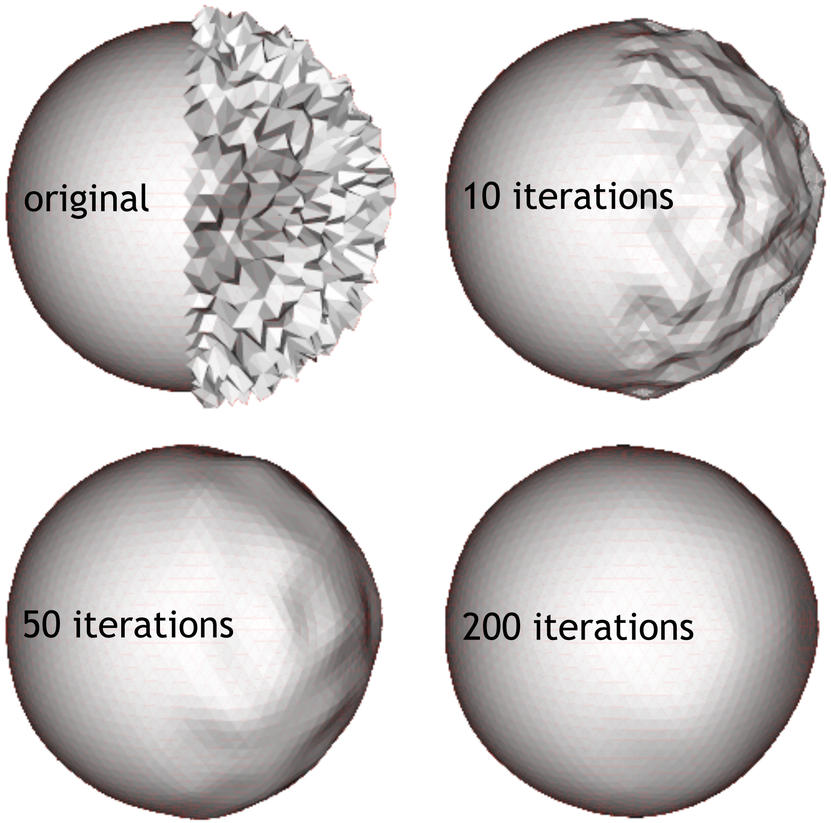
A Signal Processing Approach to Fair Surface Design
Gabriel Taubin, **ACM SIGGRAPH 95**

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**

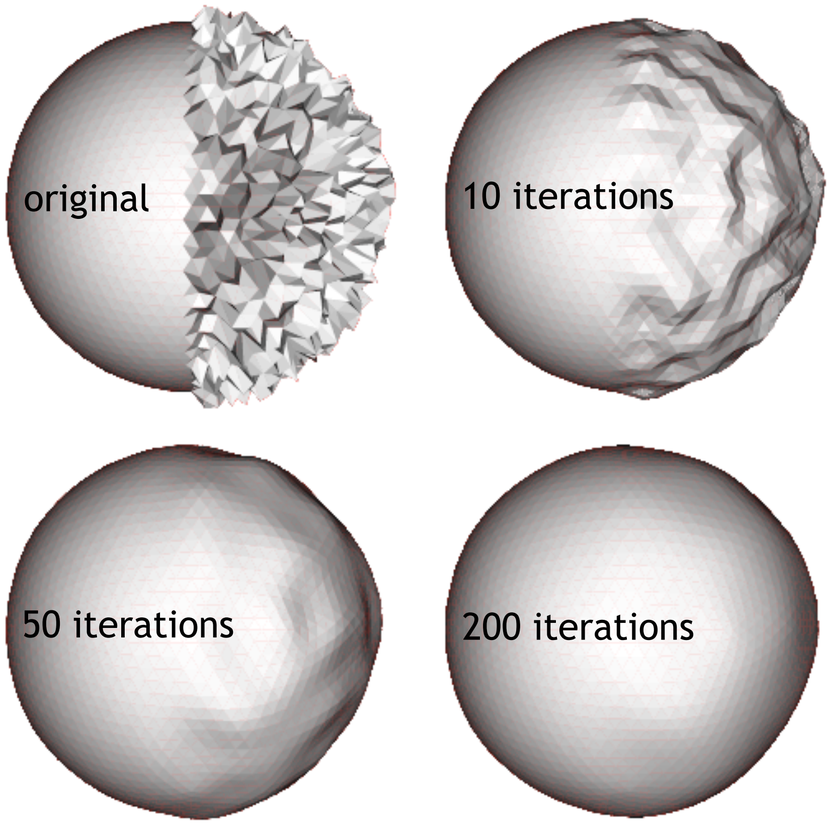
Taubin Smoothing: Explicit Steps

- Per-vertex iterations

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

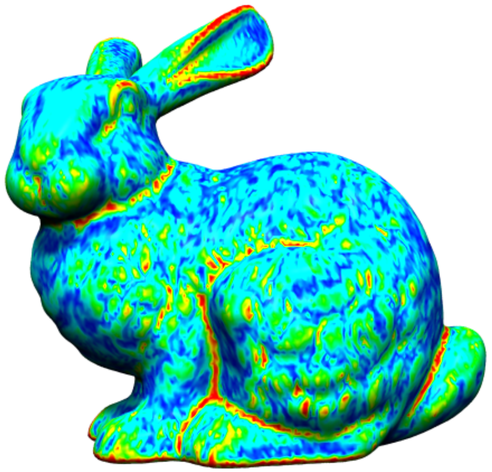
- Simple to implement

- Requires many iterations
- Need to tweak λ, u

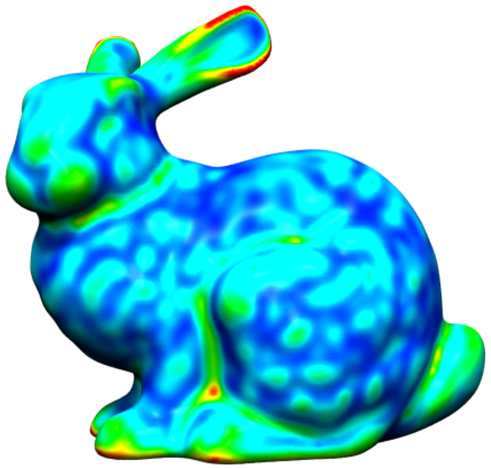


A Signal Processing Approach to Fair Surface Design
Gabriel Taubin, **ACM SIGGRAPH 95**

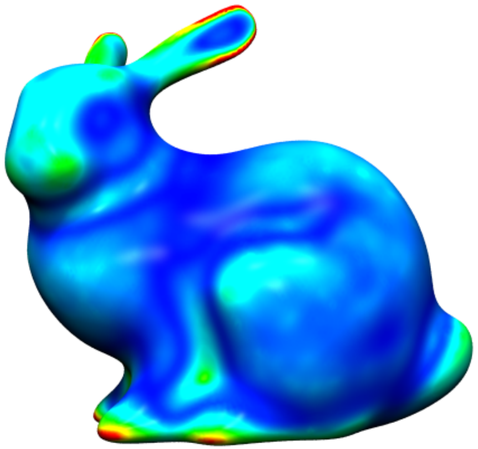
Example



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

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

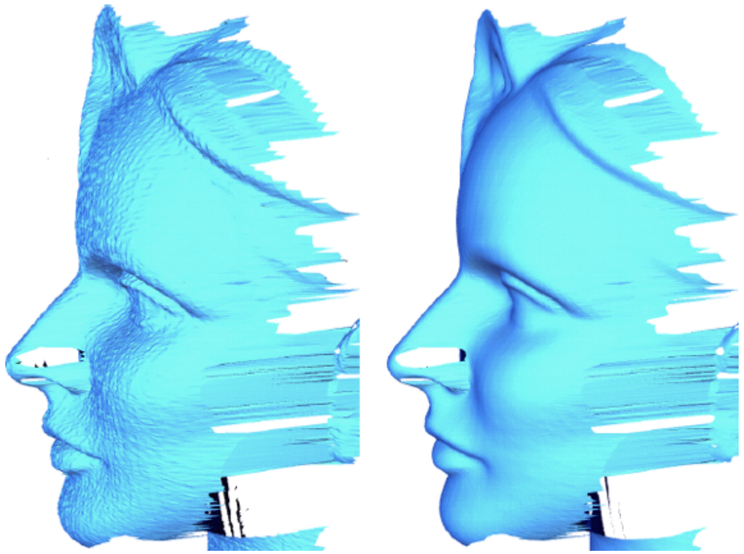
$$\mathbf{p}^{(n+1)} - \mathbf{p}^{(n)} = dt\, \lambda L \mathbf{p}^{(n+1)}$$

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

Implicit Fairing: Implicit Euler Steps

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

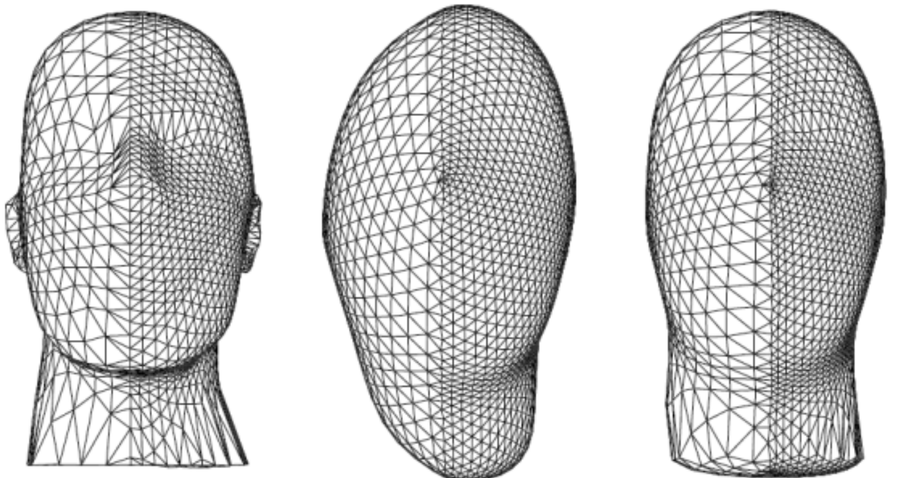
$$(I - \lambda L)\tilde{\mathbf{p}} = \mathbf{p}$$



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

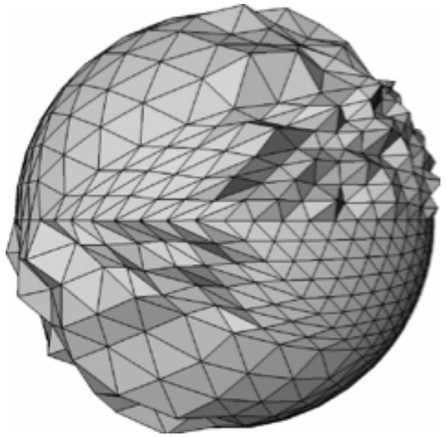


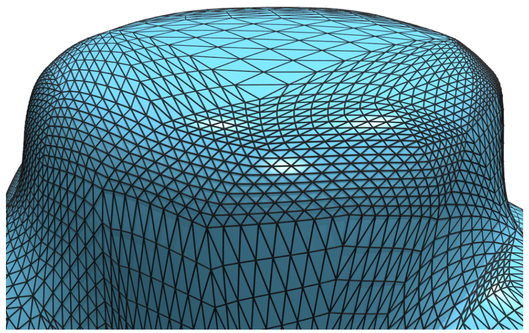
original uniform cotan

Comparison of the weights

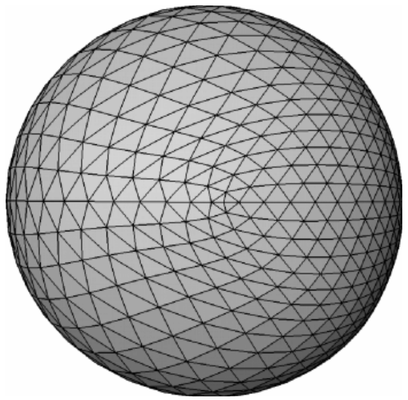
- Explicit flow with different weights:

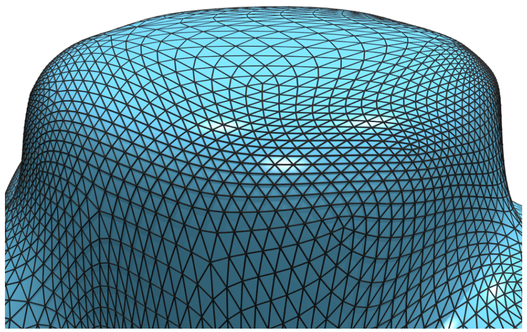
original



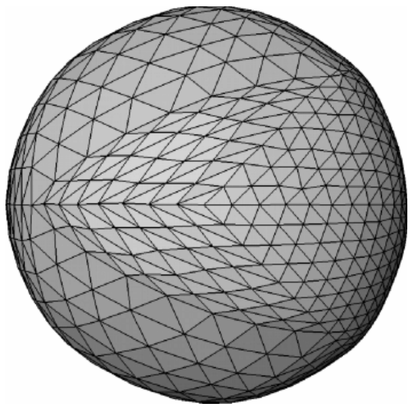


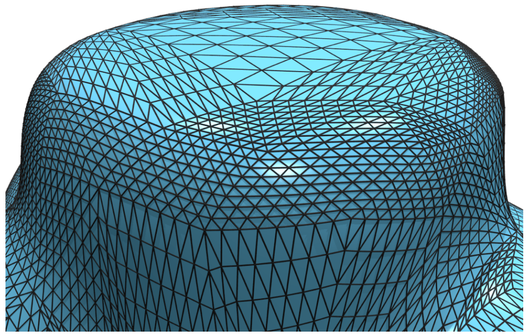
uniform





cotan





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 $H = \text{const}$

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

Laplace equation

- only trivial solution, no connection to initial surface $\mathbf{p}$

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!

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

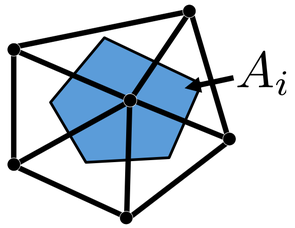
$$\min_{\tilde{\mathbf{p}}} \int_{\mathcal{M}} \underbrace{\|\Delta_{\mathcal{M}} \tilde{\mathbf{p}}\|^2}_{\text{small } H} + \underbrace{w\|\tilde{\mathbf{p}} - \mathbf{p}\|^2}_{\text{stay close to original surface}}$$

weighting factor (like $1/\lambda$)

Minimizing a smoothness energy

- Discretize:

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



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

- Minimize!

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

Minimizing a smoothness energy

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

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

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

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

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

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

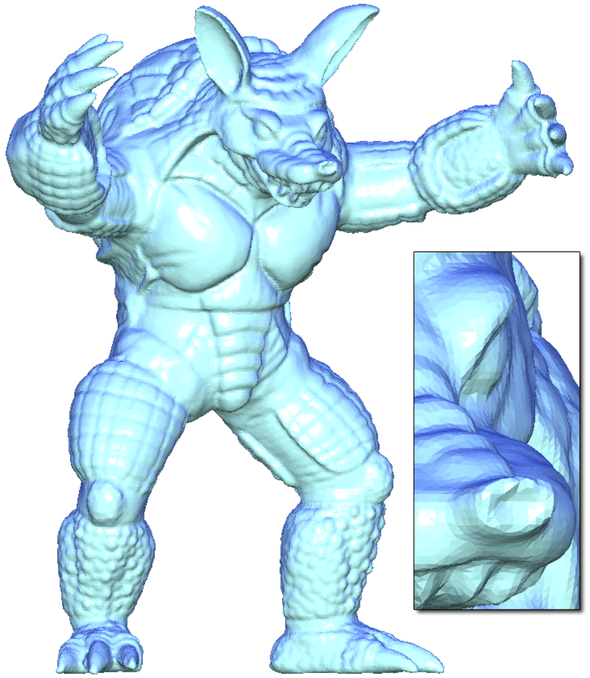
compare with implicit Euler!

tip: google “The Matrix Cookbook”

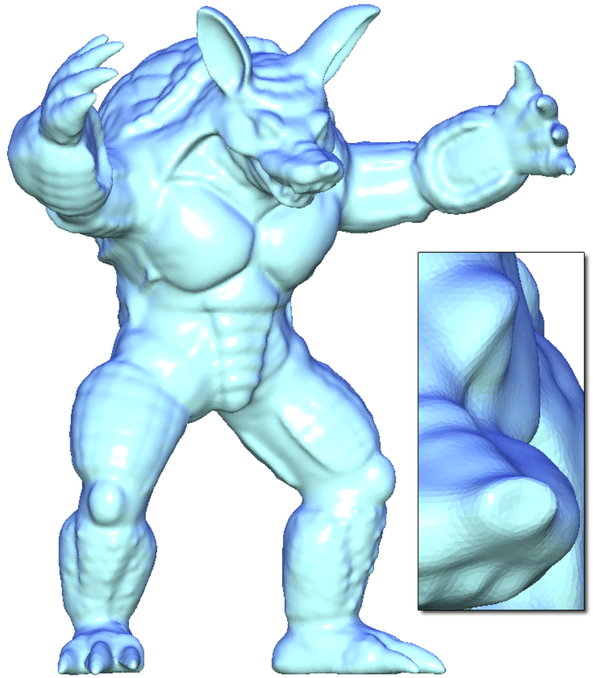
$NB:$

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

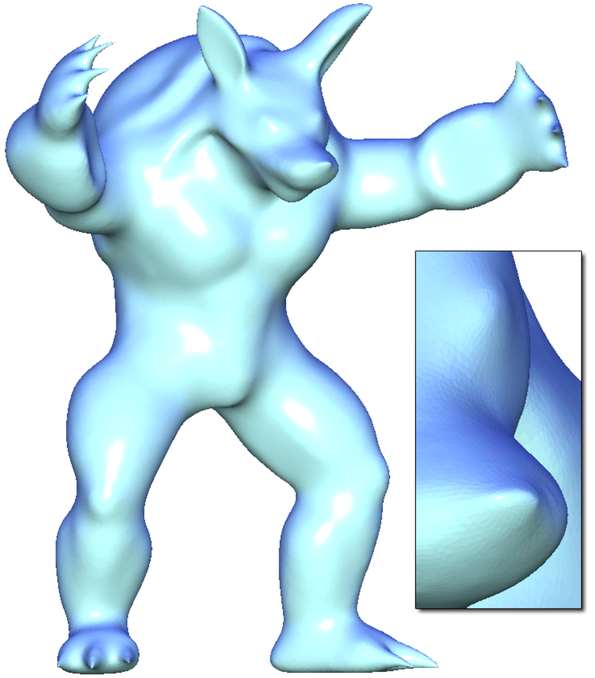
Results



original



$w = 0.2$



$w = 0.02$

Customize the energy functional

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

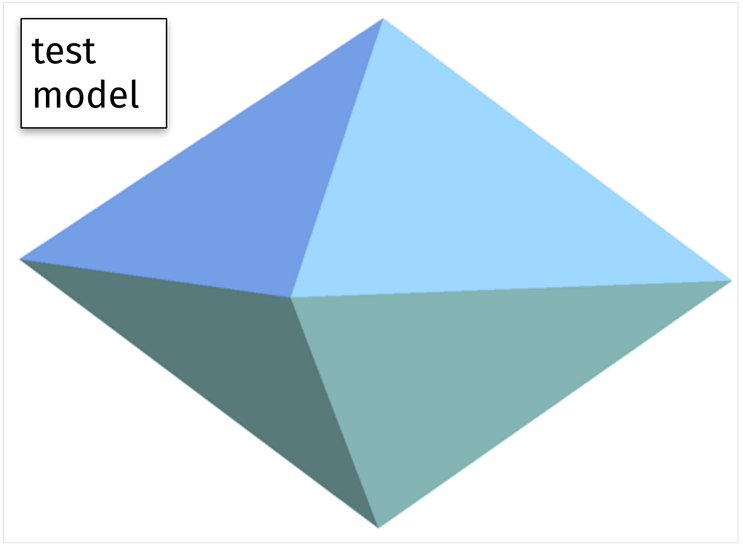
add a varying weight here to control feature preservation

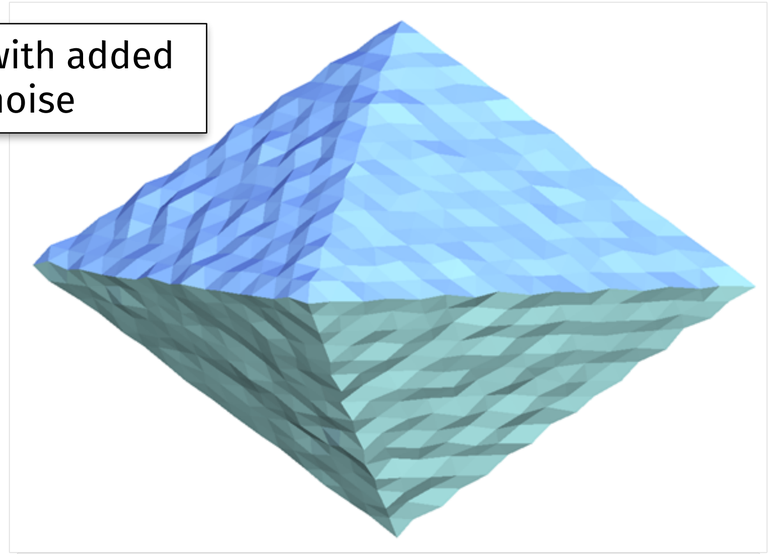
$$E(\tilde{\mathbf{p}}) = \tilde{\mathbf{p}}^T L^T (M^{0.5} W_f M^{0.5}) L\tilde{\mathbf{p}} + w(\tilde{\mathbf{p}} - \mathbf{p})^T M(\tilde{\mathbf{p}} - \mathbf{p})$$

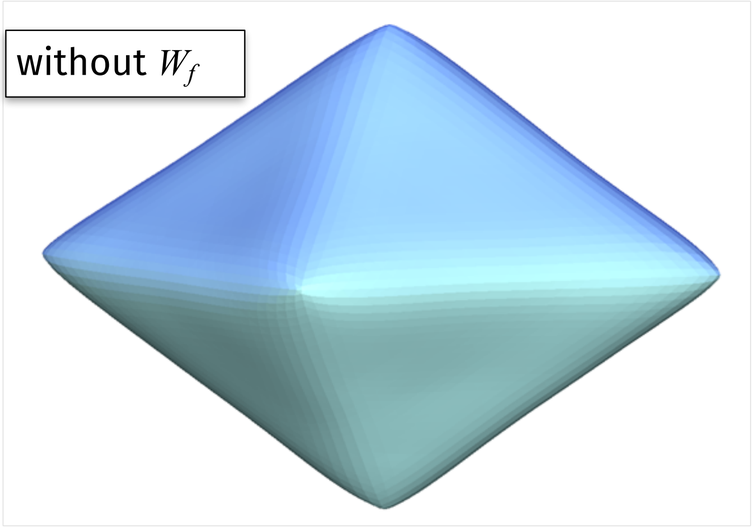
$(W_f)_{ii}$ can be inverse-proportional to e.g. curvature at vertex *i*

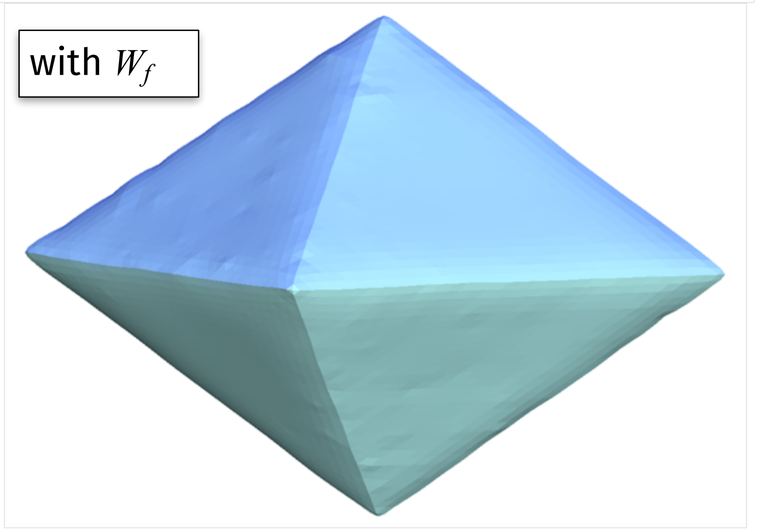
(slight abuse of notation, the energy is actually the sum of the *xyz* components, $E(\tilde{\mathbf{p}}) = \sum_{\mathbf{v}\in\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \tilde{\mathbf{v}}^T L^T \ldots$ like in slide #58)

Using W_f



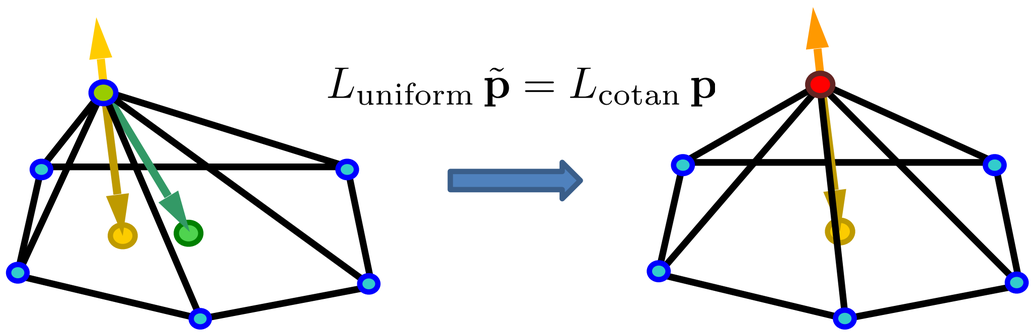






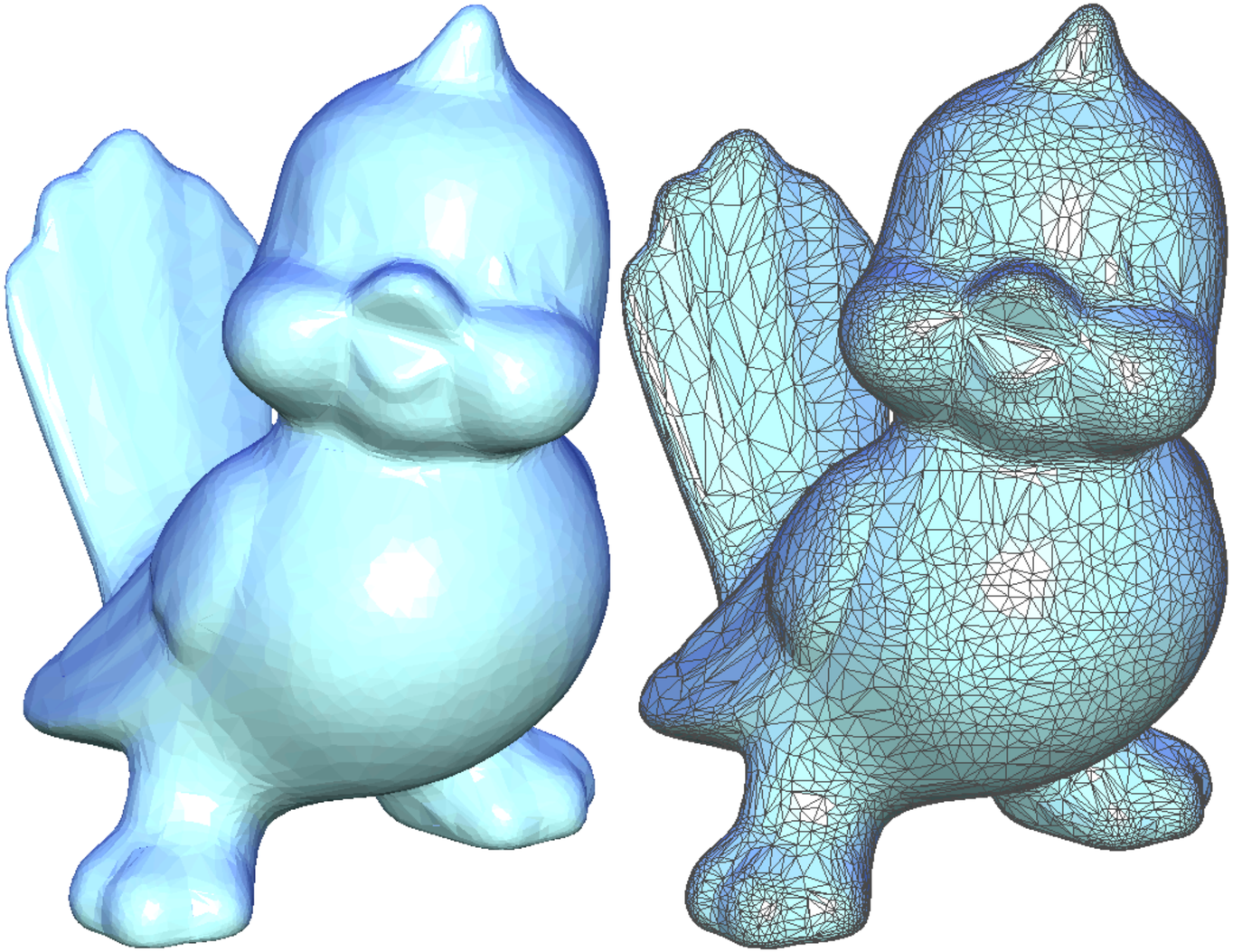
Customize the energy functional

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

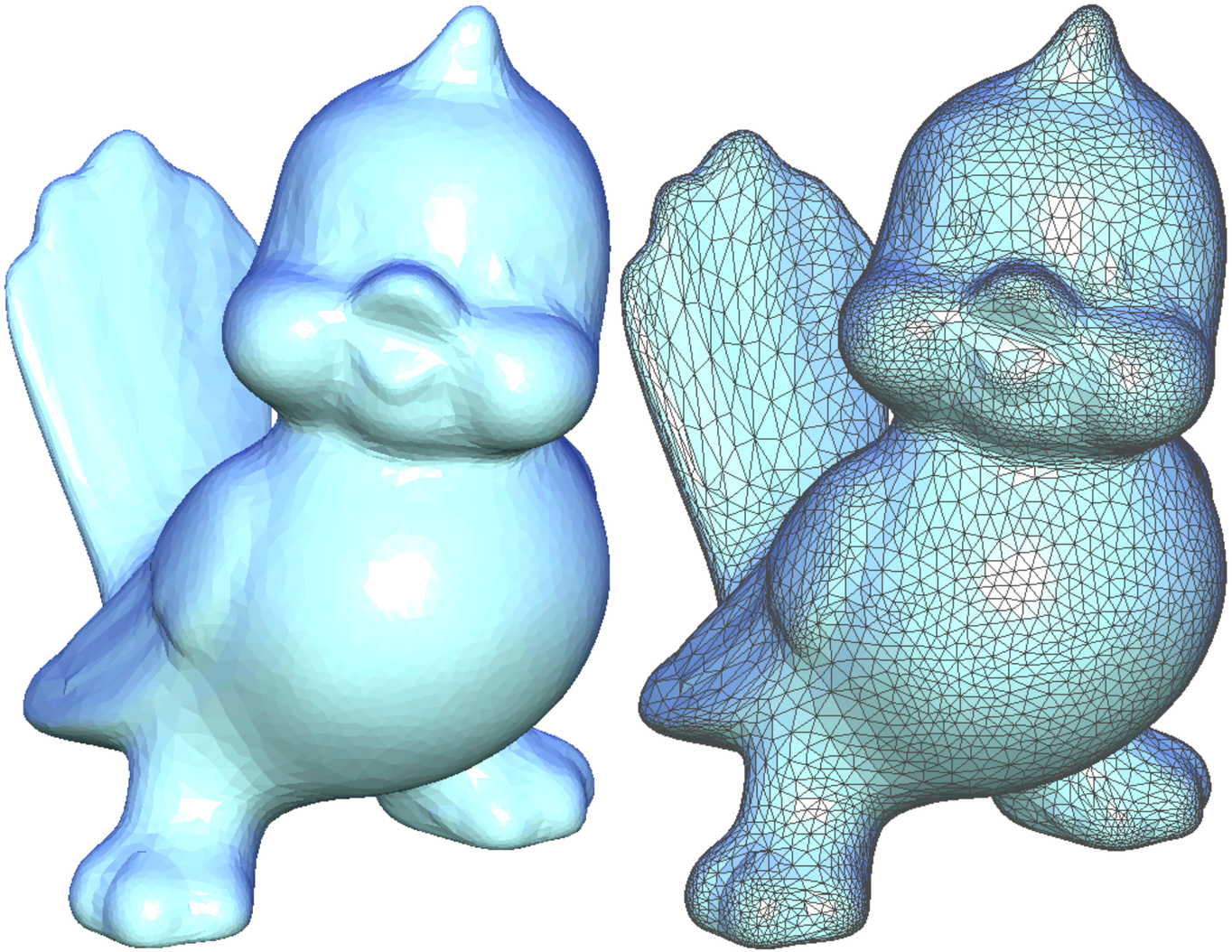


$$\min_{\tilde{\mathbf{p}}} \sum_{i=1}^{n} A_i \left(\| L_{\text{uniform}}\,\tilde{\mathbf{p}}_i - L_{\text{cotan}}\,\mathbf{p}_i \|^2 + w \| \tilde{\mathbf{p}}_i - \mathbf{p}_i \|^2 \right)$$

Original



Triangle Shape Optimization

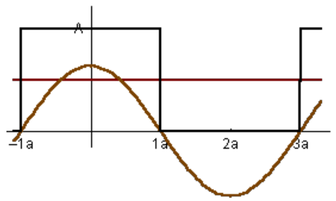


Smoothing as filtering

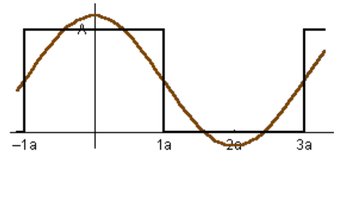
Fourier analysis

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

basis functions



weighted sum

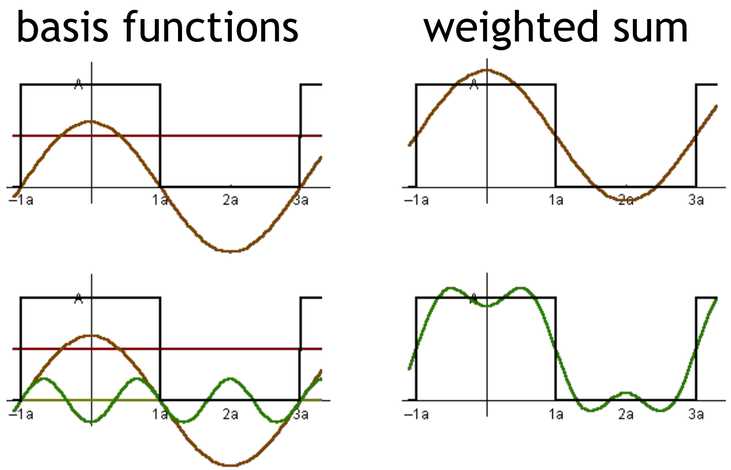


Joseph Fourier 1768 - 1830

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

Fourier analysis

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

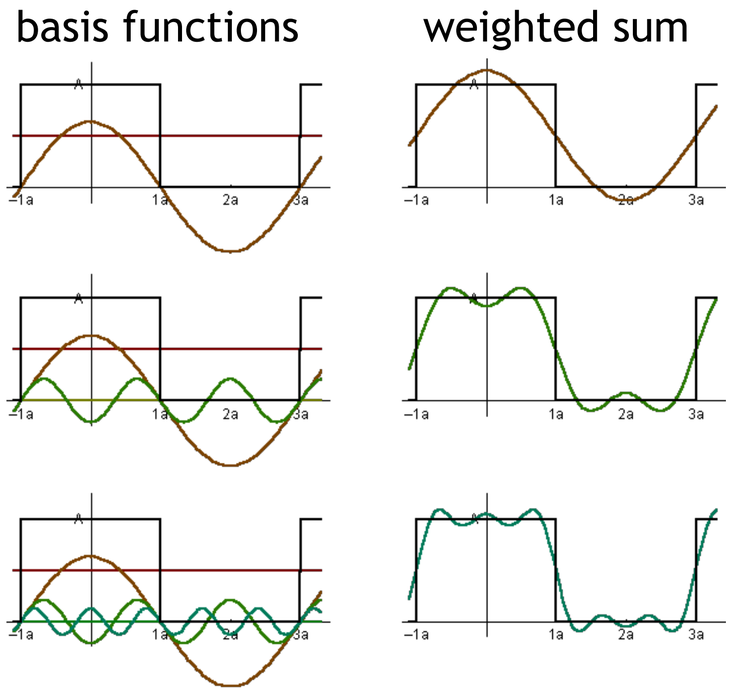


Joseph Fourier 1768 - 1830

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

Fourier analysis

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



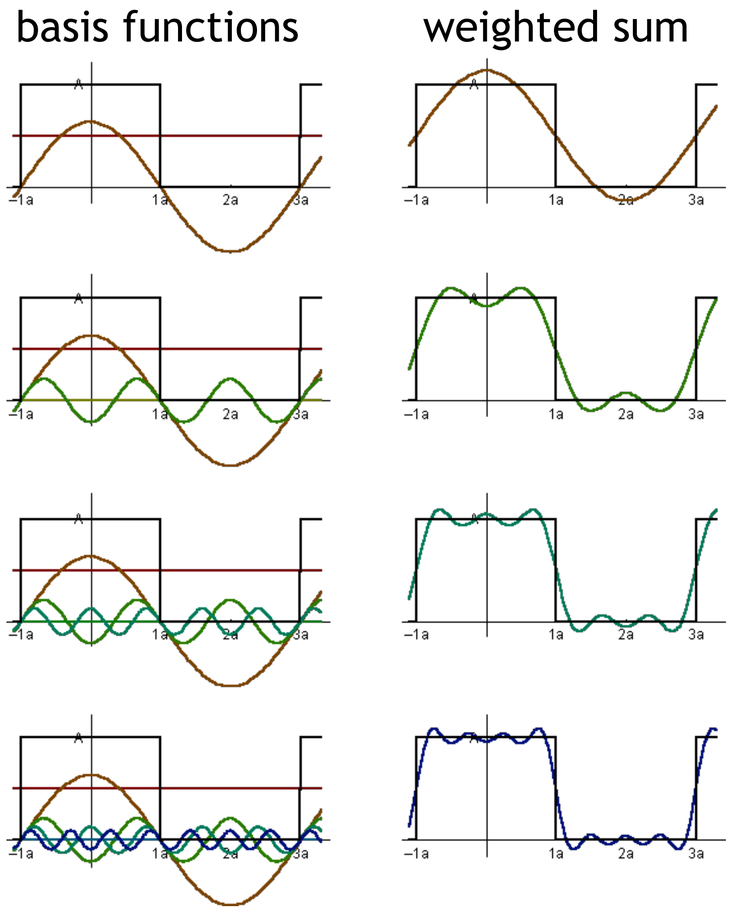
Joseph Fourier 1768 - 1830

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

Fourier analysis

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

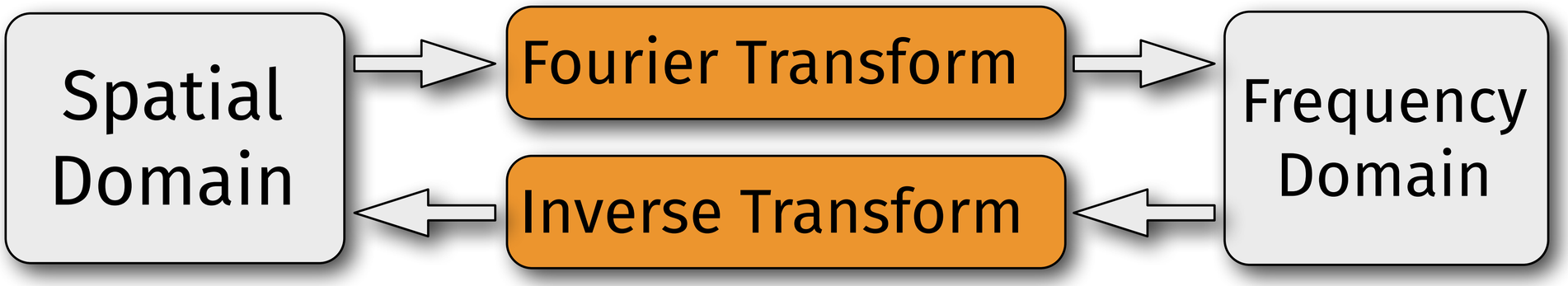
Joseph Fourier 1768 - 1830



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

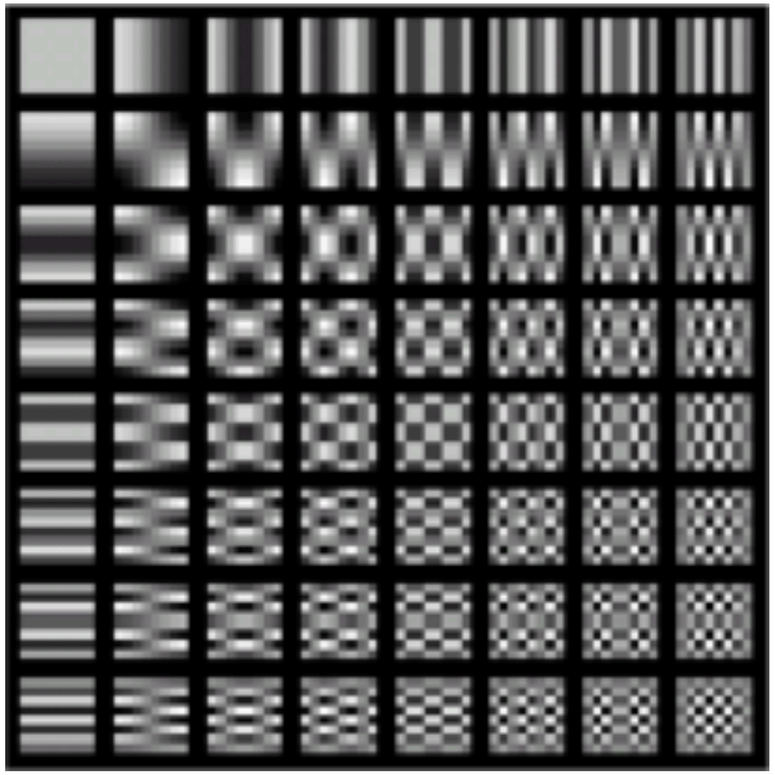
Fourier analysis

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



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

Also works on rectangular 2D domains

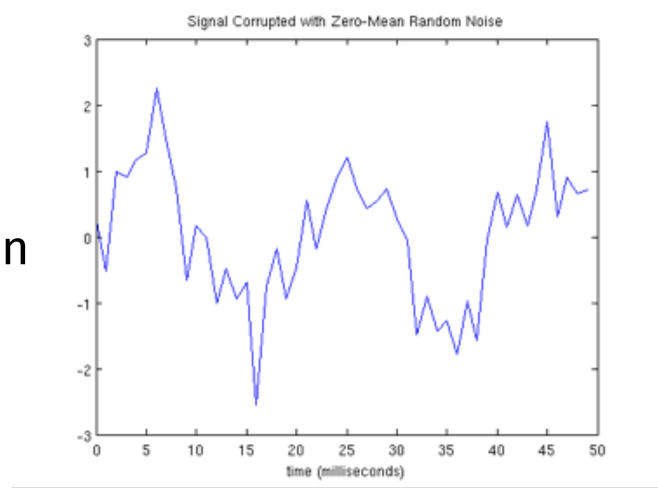


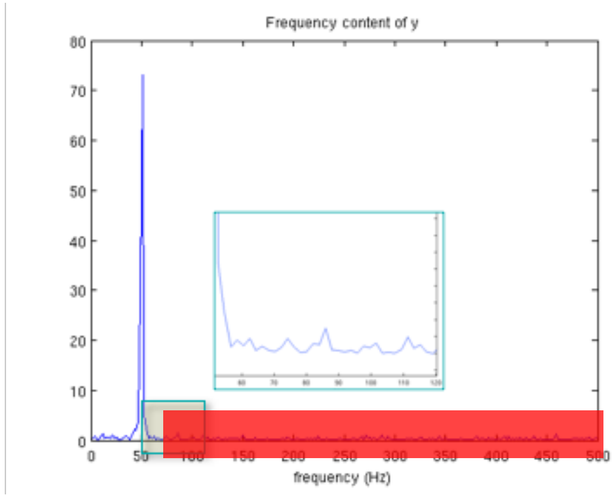
Fourier (DCT) basis functions for 8x8 grayscale images

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

Smoothing = filtering high frequencies

spatial domain

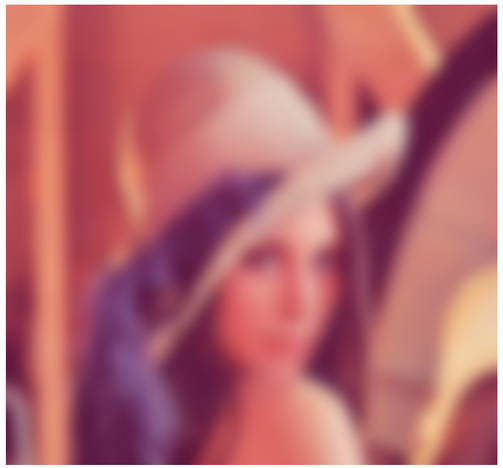




frequency domain

suppress high frequencies

original



filtered

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 mesh geometry from these eigenvectors:

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

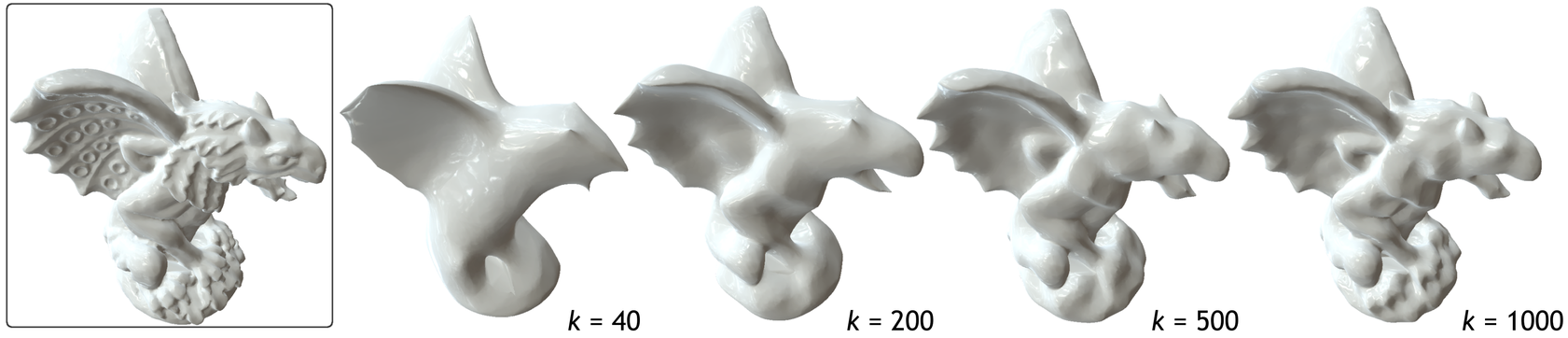
$$\tilde{\mathbf{x}} = \sum_{i=1}^{k} \left(\mathbf{x}^T \mathbf{e}_i\right) \mathbf{e}_i \qquad \tilde{\mathbf{y}} = \sum_{i=1}^{k} \left(\mathbf{y}^T \mathbf{e}_i\right) \mathbf{e}_i \qquad \tilde{\mathbf{z}} = \sum_{i=1}^{k} \left(\mathbf{z}^T \mathbf{e}_i\right) \mathbf{e}_i$$

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

Spectral analysis on meshes

too expensive for large 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 mesh geometry from these eigenvectors:



Fairing

(overview)

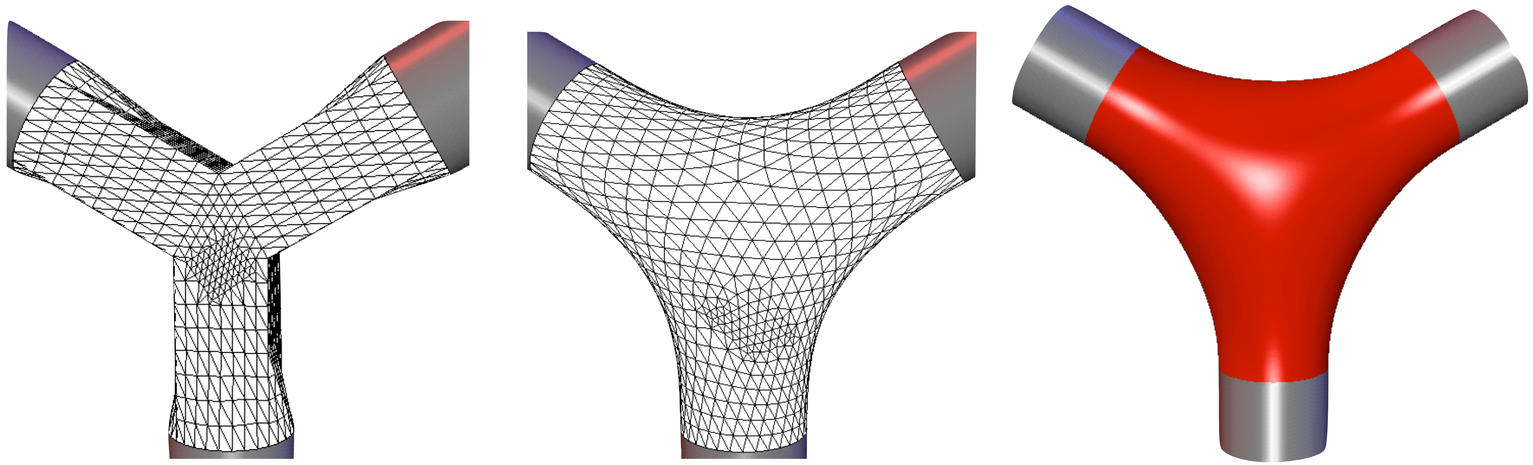
Fairing

- Fairing has the purpose to compute surfaces that are *as smooth as possible*
- Actual measure of smoothness depends on application
- *Principle of simplest shape*: the surface should be free of any unnecessary details or oscillations
- General method:
  - fixed topology
  - boundary constraints (fixed position for vertices at the boundary)
  - minimize an energy depending on the position of vertices

Fairing

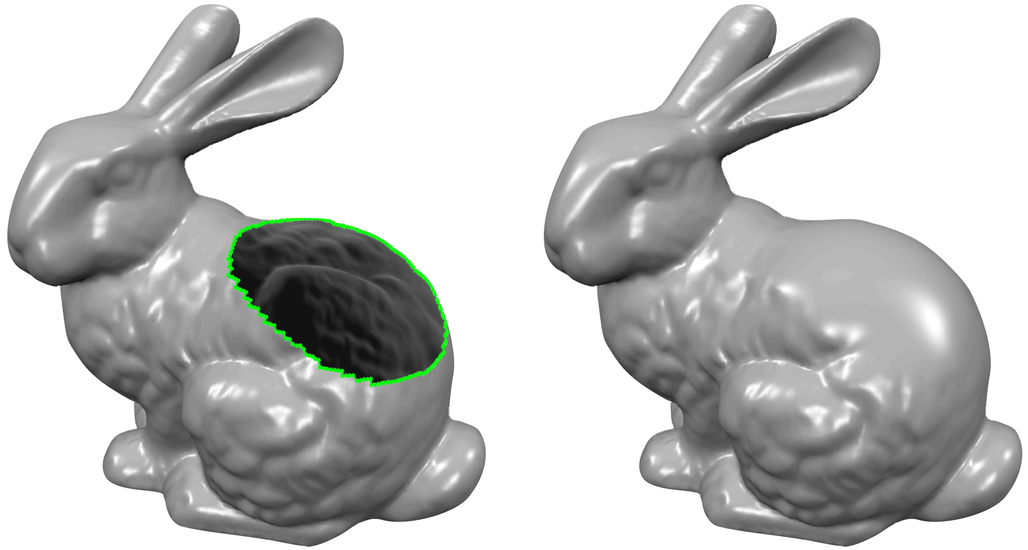
Applications:

- Blend parts, joints
- Fill holes



Care needed with setting constraints:

- boundary vertices
- outer ring of triangles



Firts-order fairing

- *Membrane energy*: measures area

$$E_M(\mathbf{x}) = \iint_\Omega \sqrt{\det(\mathbf{I})}\mathrm{d}u\mathrm{d}v$$

  highly non-linear, thus difficult to minimize

- Surrogate (linearization): *Dirichlet energy*

$$\tilde{E}_M(\mathbf{x}) = \iint_\Omega ||\mathbf{x}_u||^2 + ||\mathbf{x}_v||^2 \mathrm{d}u\mathrm{d}v$$

First-order fairing

- Minimization of energy functional is studied with *calculus of variations*
- It can be proved that the Dirichlet energy is minimized by the function that satisfies the *Laplace equation*:

$$\mathbf{Lx} = \mathbf{0}$$

- Boundary conditions fix the position of some of the unknowns
- The system is sparse and, under suitable manipulations, symmetric and positive definite
- Efficient solvers can be used (e.g., *cholmod*)

Second-order fairing

- *Thin-plate energy*: measures curvature

$$E_{TP}(\mathbf{x}) = \iint_{\Omega} \kappa_1^2 + \kappa_2^2 \, \mathrm{d}u\mathrm{d}v$$

- Linearization:

$$\tilde{E}_{TP}(\mathbf{x}) = \iint_{\Omega} ||\mathbf{x}_{uu}||^2 + 2||\mathbf{x}_{uv}||^2 + |\mathbf{x}_{vv}||^2 \, \mathrm{d}u\mathrm{d}v$$

- Solved by the bi-Laplacian system: $\mathbf{L}^2\mathbf{x} = 0$

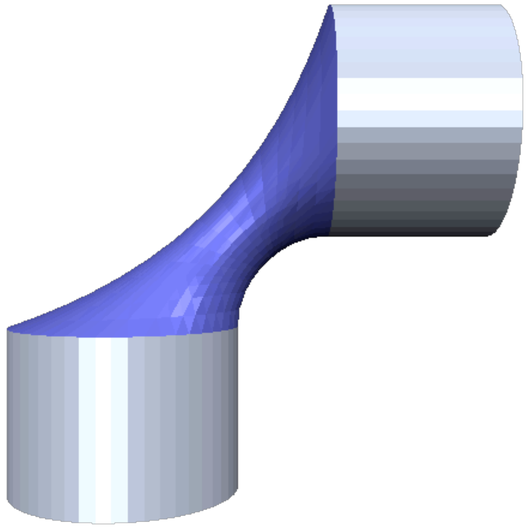
Third-order fairing

- *Higher-order energy* measuring variation of curvature

$$E_{TP}(\mathbf{x}) = \iint_{\Omega} \left(\frac{\partial \kappa_1}{\partial \mathbf{t}_1} \right)^2 + \left(\frac{\partial \kappa_2}{\partial \mathbf{t}_2} \right)^2 \, \mathrm{d}u\mathrm{d}v$$

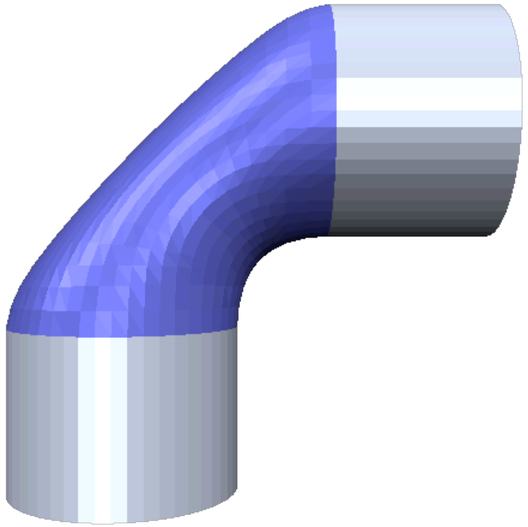
- can be also linearized and solved by the tri-Laplacian system:

$$\mathbf{L}^3\mathbf{x} = 0$$



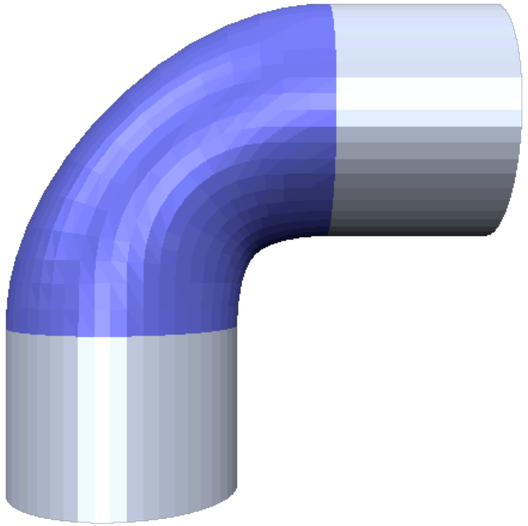
Membrane

$\Delta_{\mathcal{S}}\mathbf{p} = 0$



Thin Plate

$\Delta^2_{\mathcal{S}}\mathbf{p} = 0$



$\Delta^3_{\mathcal{S}}\mathbf{p} = 0$

Relation between fairing and diffusion flow

- A fair surface satisfying $\mathbf{L}^k\mathbf{x} = 0$ is a steady state for the flow

$$\frac{\partial f(\mathbf{x}, t)}{\partial t} = \lambda \Delta^k f(\mathbf{x}, t)$$

- Thus, fair surfaces are *as smooth as possible*
- Explicit integration of the Laplacian flow is equivalent to one Jacobi iteration to solve the related Laplace equation

Thank you