Shape Modeling and Geometry Processing

INTERACTIVE GEOMETRY LAB

March 26, 2025

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

RECAP: Intrinsic Geometry

- First *fundamental* form

Important! 😮

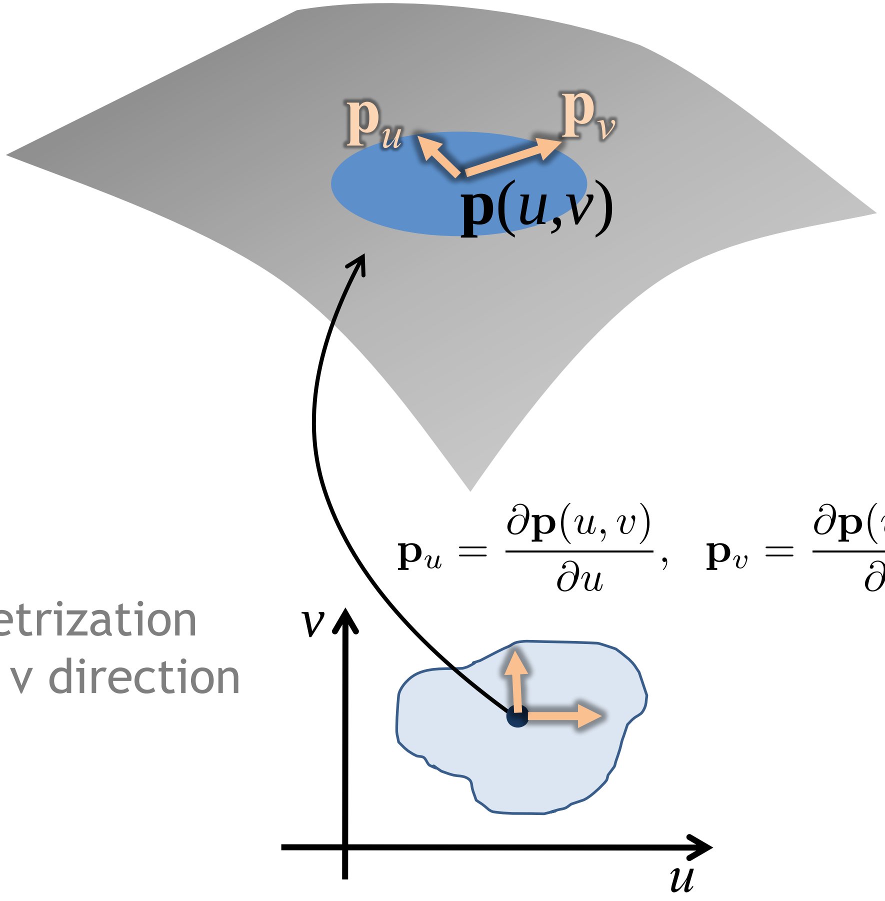
Parametrization speed of u direction

Alignment of parametrizations

$$\mathbf{I} = \begin{pmatrix} E & F \\ F & G \end{pmatrix} = \begin{pmatrix} \mathbf{p}_u^\top \mathbf{p}_u & \mathbf{p}_u^\top \mathbf{p}_v \\ \mathbf{p}_u^\top \mathbf{p}_v & \mathbf{p}_v^\top \mathbf{p}_v \end{pmatrix}$$

Alignment of parametrizations

Parametrization speed of v direction



$$\mathbf{p}_u = \frac{\partial \mathbf{p}(u,v)}{\partial u}, \quad \mathbf{p}_v = \frac{\partial \mathbf{p}(u,v)}{\partial v}$$

- Needed to define lengths, angles and areas.
- It defines the *metric* of the surface.

RECAP: Extrinsic Geometry

Out of plane bending in *u* direction.

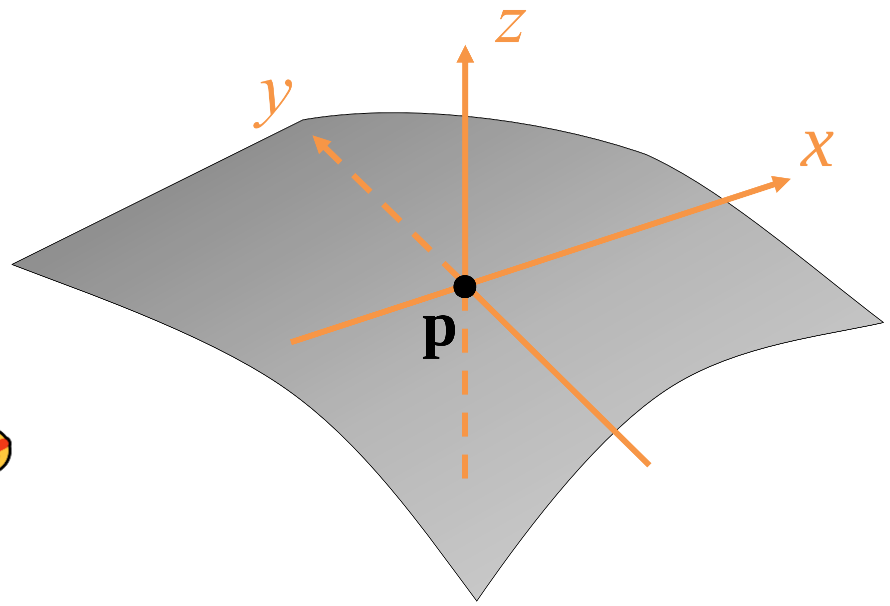
Mixed curvature interaction

$$\mathbf{II} = \begin{pmatrix} e & f \\ f & g \end{pmatrix} = \begin{pmatrix} \mathbf{p}_{uu}^\top \mathbf{n} & \mathbf{p}_{uv}^\top \mathbf{n} \\ \mathbf{p}_{uv}^\top \mathbf{n} & \mathbf{p}_{vv}^\top \mathbf{n} \end{pmatrix}$$

Beautiful! 😍

Mixed curvature interaction

Out of plane bending in *v* direction.



„How is my surface bend relative to the normal plane?“

RECAP: Fundamental Forms

- First fundamental form (first derivative surface behavior)

$$\mathbf{I} = \begin{pmatrix} E & F \\ F & G \end{pmatrix} = \begin{pmatrix} \mathbf{p}_u^\mathsf{T}\mathbf{p}_u & \mathbf{p}_u^\mathsf{T}\mathbf{p}_v \\ \mathbf{p}_u^\mathsf{T}\mathbf{p}_v & \mathbf{p}_v^\mathsf{T}\mathbf{p}_v \end{pmatrix}$$

(determines the metric $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\mathrm{T}\mathbf{I}\mathbf{y}$)

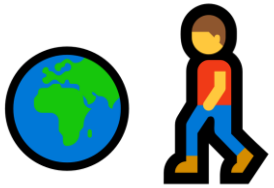
Intrinsic
(Measurements made on surface)

- Second fundamental form (second derivative surface behavior)

$$\mathbf{II} = \begin{pmatrix} e & f \\ f & g \end{pmatrix} = \begin{pmatrix} \mathbf{p}_{uu}^\mathsf{T}\mathbf{n} & \mathbf{p}_{uv}^\mathsf{T}\mathbf{n} \\ \mathbf{p}_{uv}^\mathsf{T}\mathbf{n} & \mathbf{p}_{vv}^\mathsf{T}\mathbf{n} \end{pmatrix}$$

(determines the normal curvature $\frac{\mathbf{x}^\mathrm{T}\mathbf{II}\mathbf{x}}{\mathbf{x}^\mathrm{T}\mathbf{x}}$)

Extrinsic
(3D space properties)



RECAP: Surface Curvatures

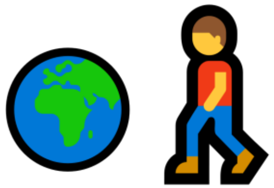
Theorem:
The principal curvatures $\kappa_1 = \kappa_{min}$, $\kappa_2 = \kappa_{max}$ are eigenvalues of $\mathbf{II}$

- Second fundamental form (second derivative surface behavior)

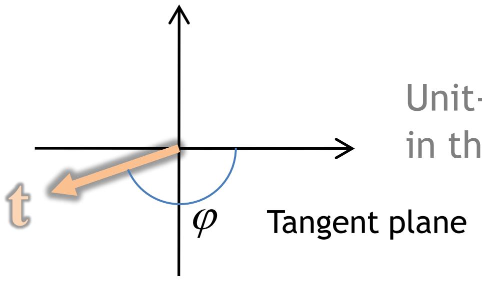
$$\mathbf{II} = \begin{pmatrix} e & f \\ f & g \end{pmatrix} = \begin{pmatrix} \mathbf{p}_{uu}^{\mathsf{T}}\mathbf{n} & \mathbf{p}_{uv}^{\mathsf{T}}\mathbf{n} \\ \mathbf{p}_{uv}^{\mathsf{T}}\mathbf{n} & \mathbf{p}_{vv}^{\mathsf{T}}\mathbf{n} \end{pmatrix}$$

(determines the normal curvature $\kappa_n(\mathbf{x}) = \frac{\mathbf{x}^{\mathsf{T}}\mathbf{II}\mathbf{x}}{\mathbf{x}^{\mathsf{T}}\mathbf{x}}$)

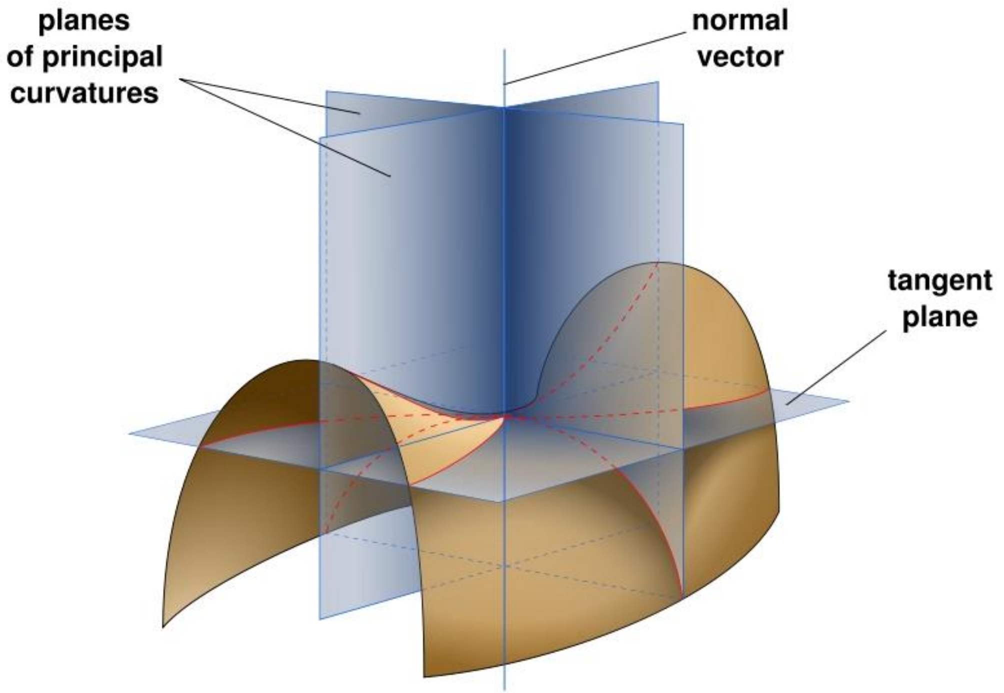
Extrinsic
(3D space properties)



Principal Directions



Unit-length direction $\mathbf{t}$
in the tangent plane:



Euler's Theorem:
Planes of principal curvature are **orthogonal** and independent of parameterization.

$$\kappa_n(\varphi) = \kappa_1 \cos^2 \varphi + \kappa_2 \sin^2 \varphi, \quad \varphi = \text{angle with } \mathbf{t}_1$$

Surface Curvatures

- Principal curvatures
  - Minimal curvature $\kappa_1 = \kappa_{\min} = \min_{\varphi} \kappa_n(\varphi)$
  - Maximal curvature $\kappa_2 = \kappa_{\max} = \max_{\varphi} \kappa_n(\varphi)$

- *Mean* curvature

due to Euler's Theorem

$$H = \frac{1}{2\pi} \int_0^{2\pi} \kappa_n(\varphi) d\varphi = \frac{\kappa_1 + \kappa_2}{2}$$

- *Gaussian* curvature $K = \kappa_1 \cdot \kappa_2$

What does this represent? 🤔

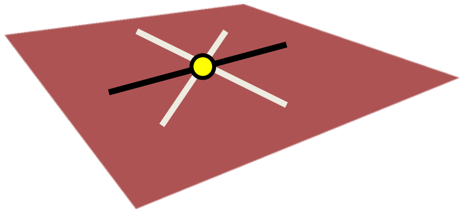
Local Surface Shape By Curvatures

$$K = \kappa_1 \cdot \kappa_2$$

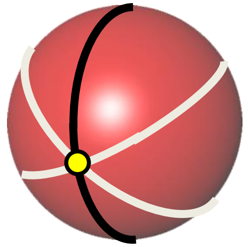
Gaussian curvature: $K = 0$ $\qquad$ $K > 0$

$\kappa_1 = \kappa_2$

Isotropic:
all directions are
principal directions



planar

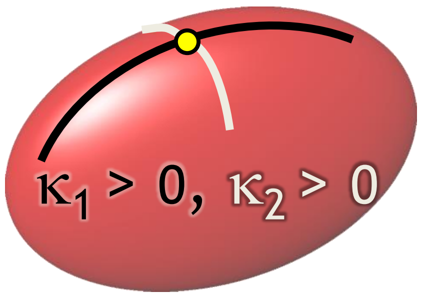


spherical (umbilical)

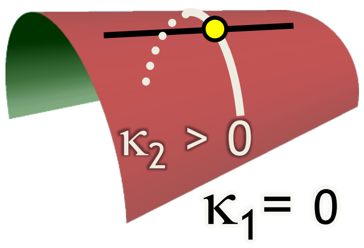
$\kappa_1 \neq \kappa_2$

Anisotropic:
2 distinct
principal
directions

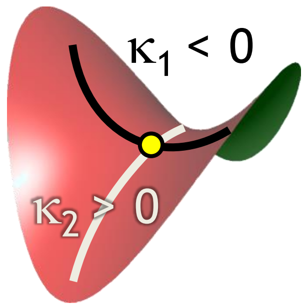
$K > 0$ $\qquad$ $K = 0$ $\qquad$ $K < 0$



elliptic



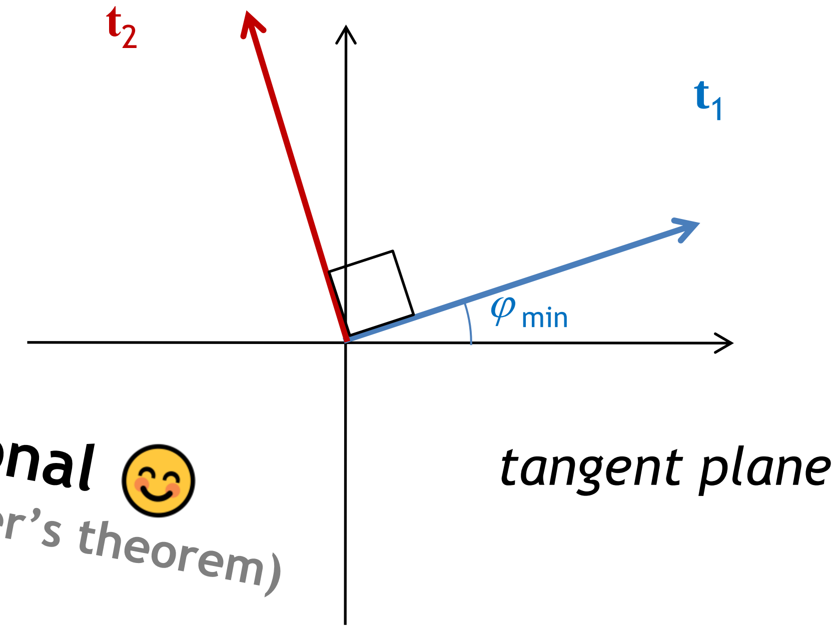
parabolic



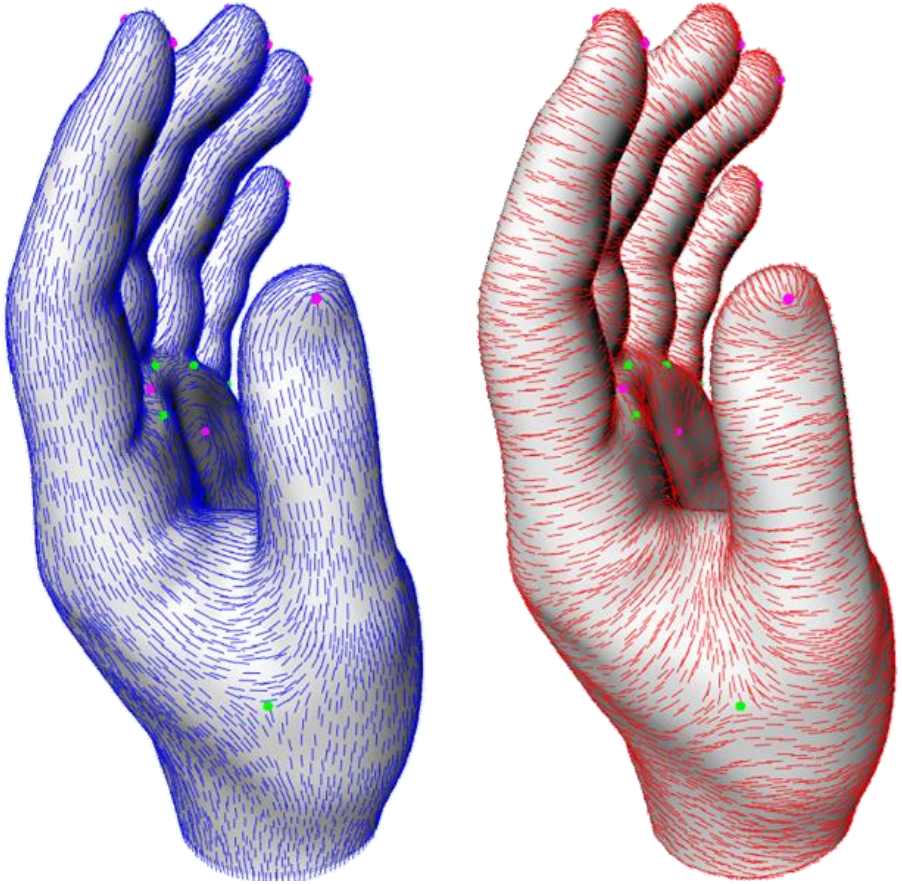
hyperbolic

Principal Directions

- Principal directions:
  tangent vectors corresponding to
  $\phi_{\min}, \phi_{\max}$ pointing towards $\kappa_{\min}, \kappa_{\max}$

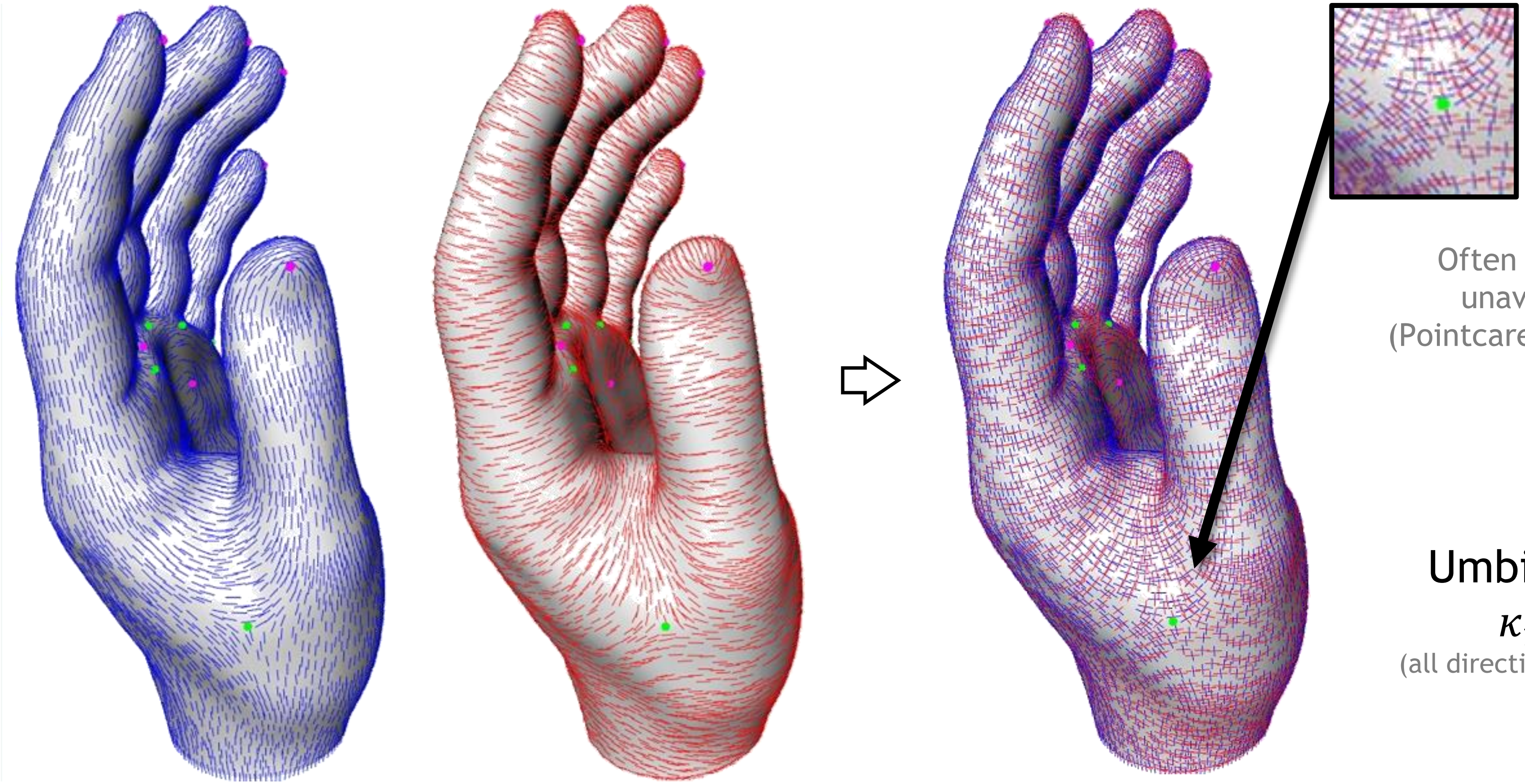


Orthogonal 😊
(Thanks to Euler's theorem)



min curvature max curvature

Principal Directions



Singularity

Often topologically unavoidable 😭 (Pointcare-Hopf Theorem)

Umbilic points

$\kappa_1 = \kappa_2$

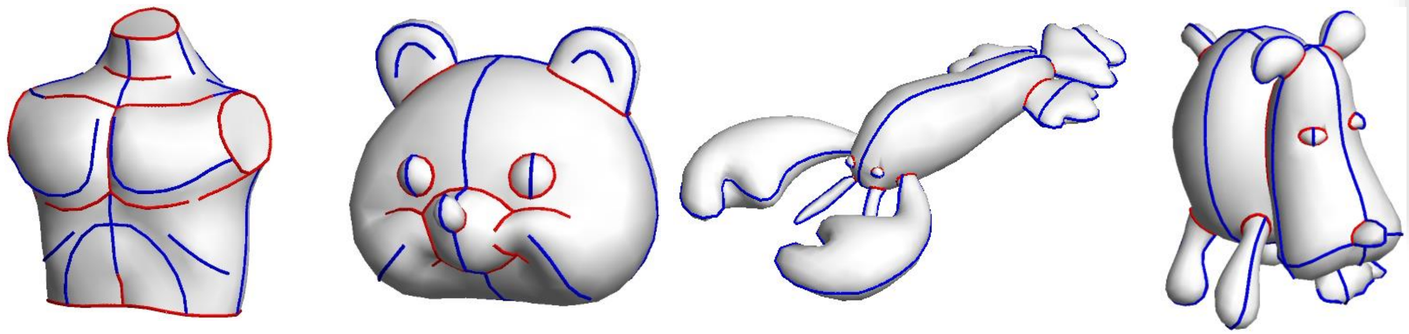
(all directions equal curving)

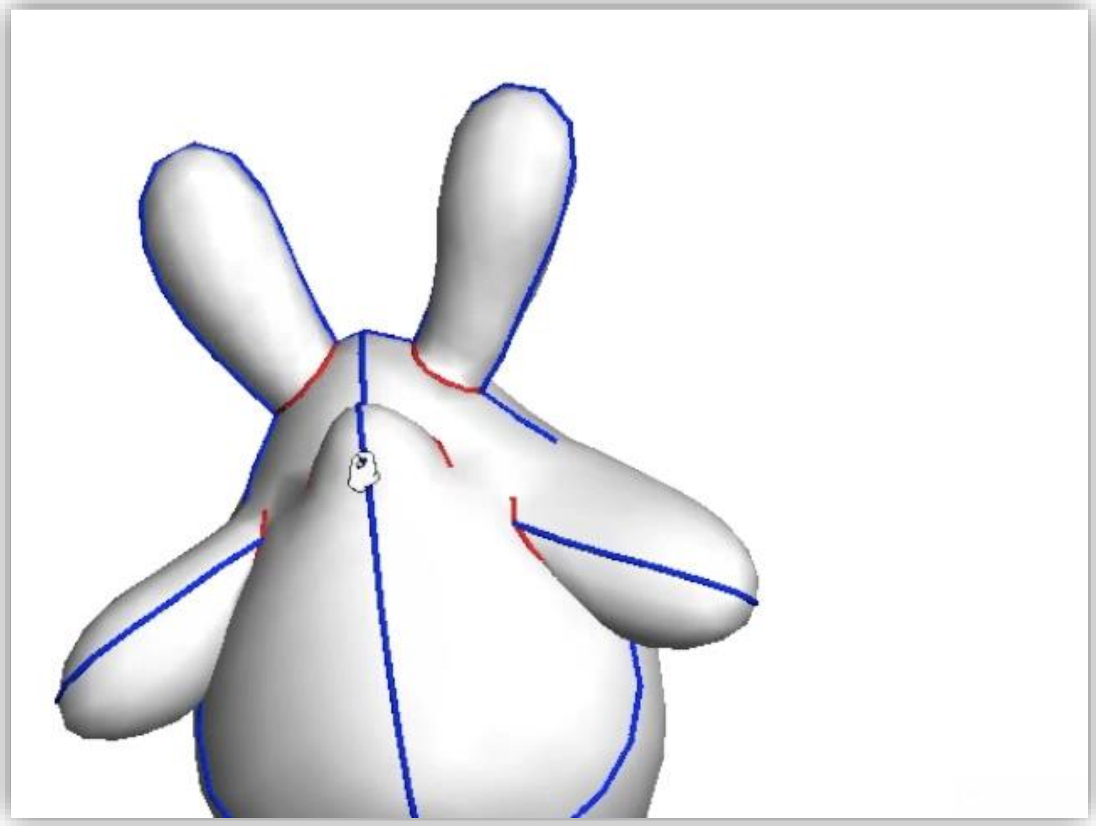
Usage example: define fair surfaces

- FiberMesh

$$\min_{\mathcal{M}} \int_{\mathcal{M}} \left(\frac{d\kappa_n}{d\mathbf{t}_1}\right)^2 + \left(\frac{d\kappa_n}{d\mathbf{t}_2}\right)^2 dA$$

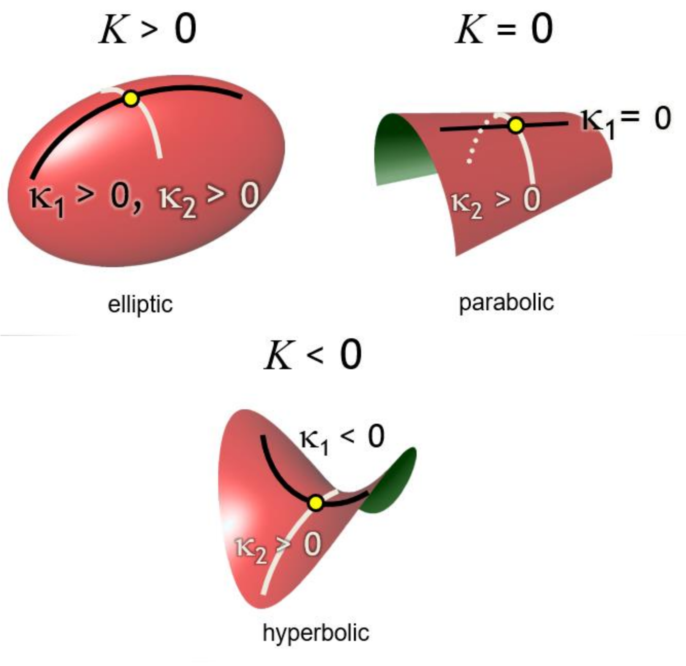
Minimize this energy with M
s.t. M interpolates given curves

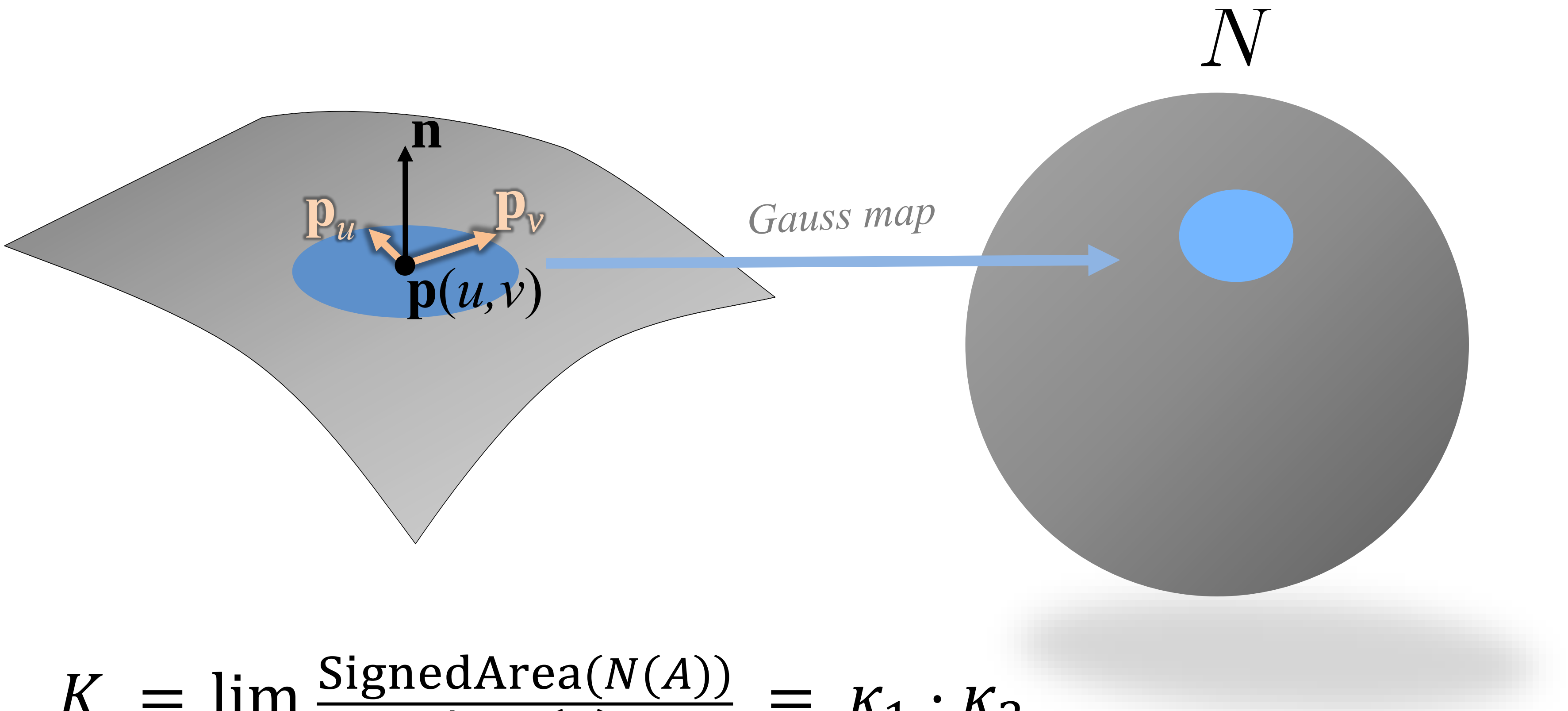




https://igl.ethz.ch/projects/FiberMesh/ (2007)

Gauss map and Gaussian curvature





Relationship to Gauss map

$$K = \lim_{A \to 0} \frac{\mathrm{SignedArea}(N(A))}{\mathrm{Area}(A)} = \kappa_1 \cdot \kappa_2$$

Easier to remember: The Gauss curvature expresses the relative relation of the principle curvatures κ_1, κ_2.

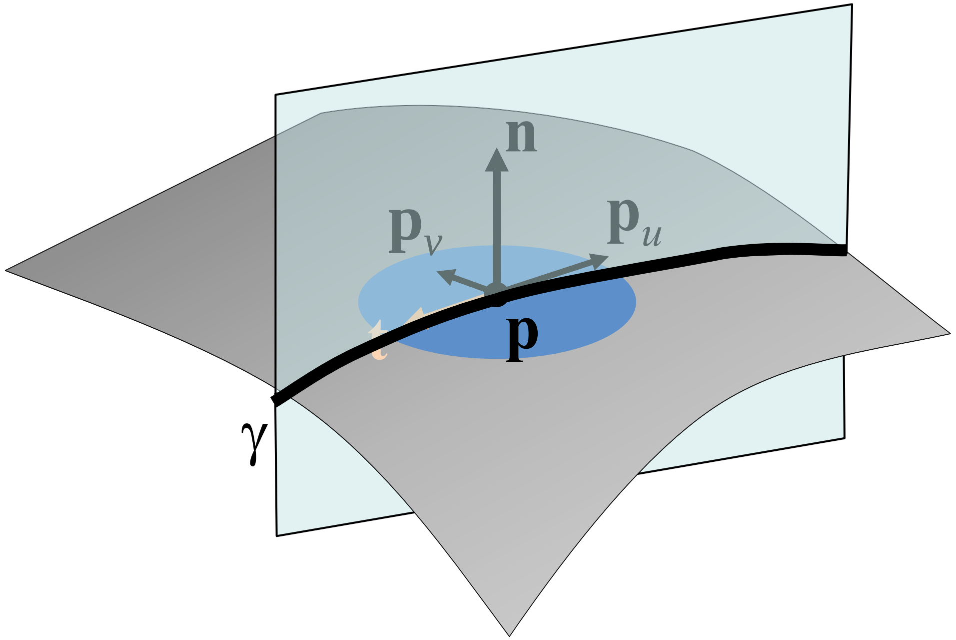
Theorema Egregium

= Exceptional, extraordinary

- Gauss curvature is intrinsic 🤯 🎉
  - (Mean curvature and principal curvatures are not)

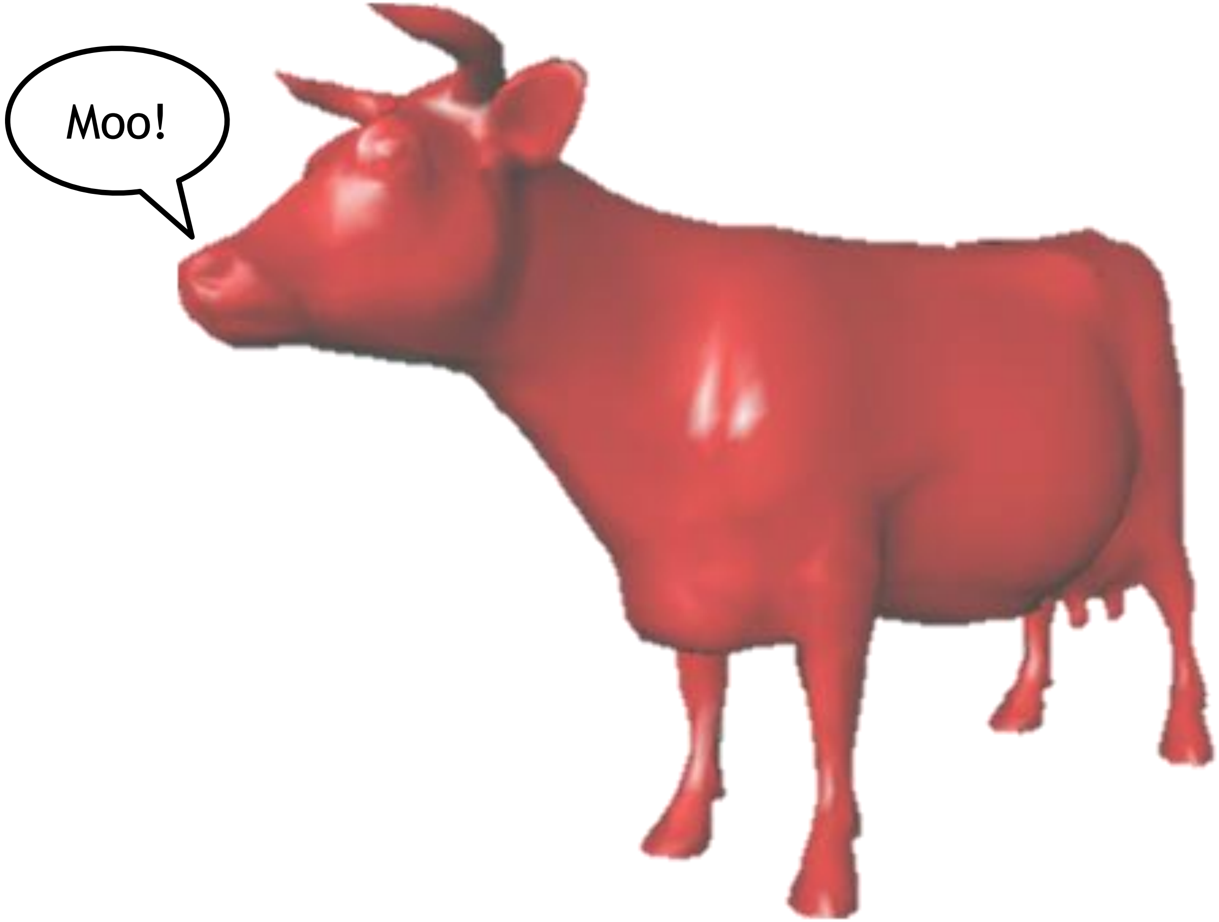
$$K = \kappa_1 \cdot \kappa_2$$

An ant can measure Gauss curvature!



It means, K can be derived solely from the first fundamental form (and its derivatives, without the normal).

What is the total curvature of this?



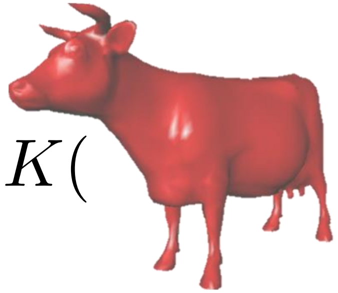
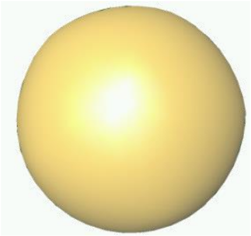
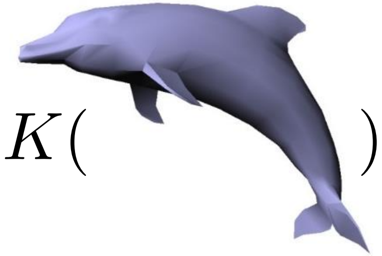
$$\int_{\mathcal{M}} K \, dA = \ ?$$

Gauss-Bonnet Theorem

- For a closed surface M:

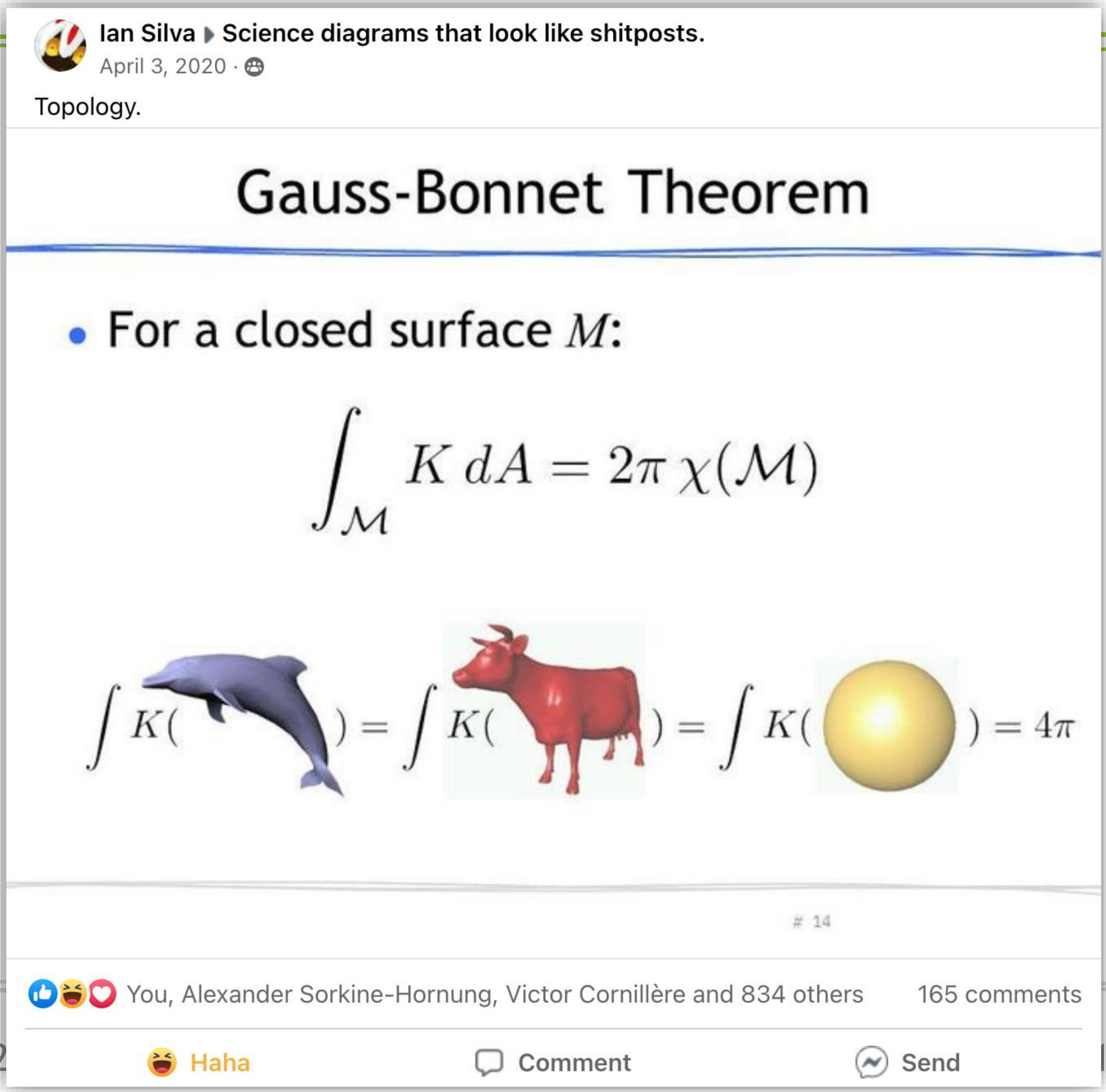
$\chi(M) = 2 - 2g - b$

$$\int_{\mathcal{M}} K \, dA = 2\pi \, \chi(\mathcal{M})$$

$\int K($  $) = \int K($  $) = \int K($  $) = 4\pi$

Total curvature is a topological invariant! 🤩

Moment of fame ☺



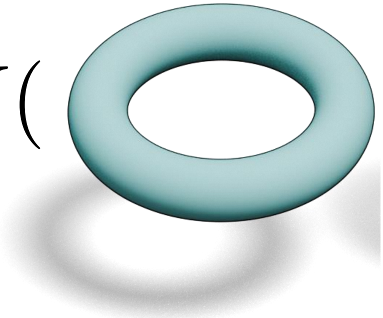
Gauss-Bonnet Theorem

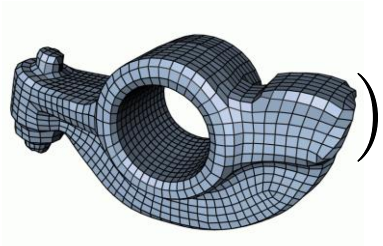
Try it yourself! 👏

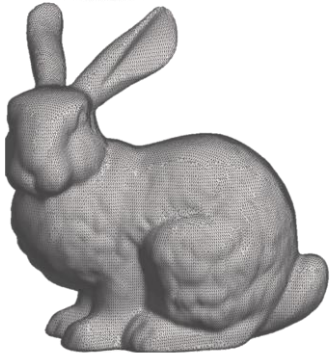
$$\int_{\mathcal{M}} K \, dA = 2\pi \, \chi(\mathcal{M})$$

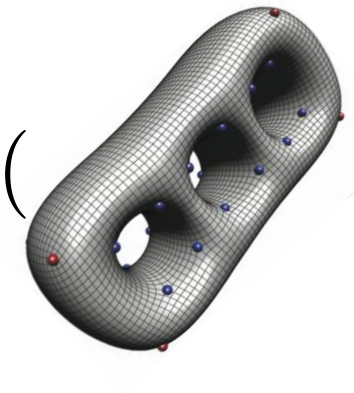
For a closed surface of genus g:

$$\chi(M) = 2 - 2g$$

$\int K($  $) = 0$

$\int K($  $) = 0$

$\int K($  $) = 4\pi$

$\int K($  $) = -8\pi$

Total curvature is a topological invariant! 🤩

Gauss-Bonnet Theorem

https://youtu.be/egEraZP9yXQ?si=tQ-yNGIi0bZue_aS

Gauss-Bonnet Theorem

- For a closed surface M: (Gauss-Bonnet theorem)

$$\int_{\mathcal{M}} K\, dA = 2\pi\, \chi(\mathcal{M})$$

- Compare with planar curves: (Turning number theorem)

$$\int_{\gamma} \kappa\, ds = 2\pi\, k$$

Towards discrete curvatures

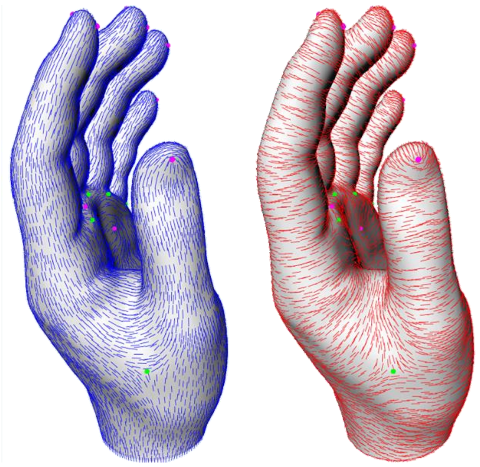
Surface Curvatures

- Principal curvatures
  - Minimal curvature $\kappa_1 = \kappa_{\min} = \min_{\varphi} \kappa_n(\varphi)$
  - Maximal curvature $\kappa_2 = \kappa_{\max} = \max_{\varphi} \kappa_n(\varphi)$

Let's focus on this one for now:

- Mean curvature $\boxed{H = \frac{\kappa_1 + \kappa_2}{2} = \frac{1}{2\pi} \int_0^{2\pi} \kappa_n(\varphi) d\varphi}$

- Gaussian curvature $K = \kappa_1 \cdot \kappa_2$

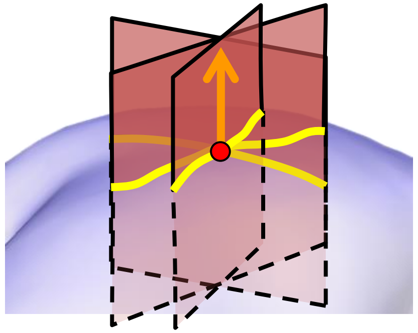


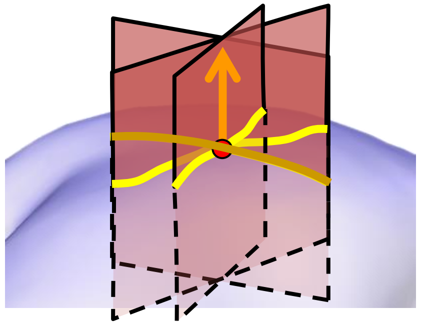
min curvature direction

max curvature direction

Mean Curvature

$$H = \frac{\kappa_1 + \kappa_2}{2} = \frac{1}{2\pi}\int_0^{2\pi} \kappa_n(\varphi)d\varphi$$





The average of all normal curvatures

Laplace Operator

$$f : \mathbb{R}^3 \to \mathbb{R} \qquad \Delta f : \mathbb{R}^3 \to \mathbb{R}$$

Laplace operator

gradient operator

2nd partial derivatives

"A second derivative operator on Euclidean $\mathbb{R}^3$" 🙂

$$\Delta f = \operatorname{div} \nabla f$$

function in Euclidean space

divergence operator

Cartesian coordinates

$$\operatorname{grad} f = \nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right) \qquad \operatorname{div}\mathbf{F} = \nabla \cdot \mathbf{F} = \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z}$$

Laplace equation (heat equation)

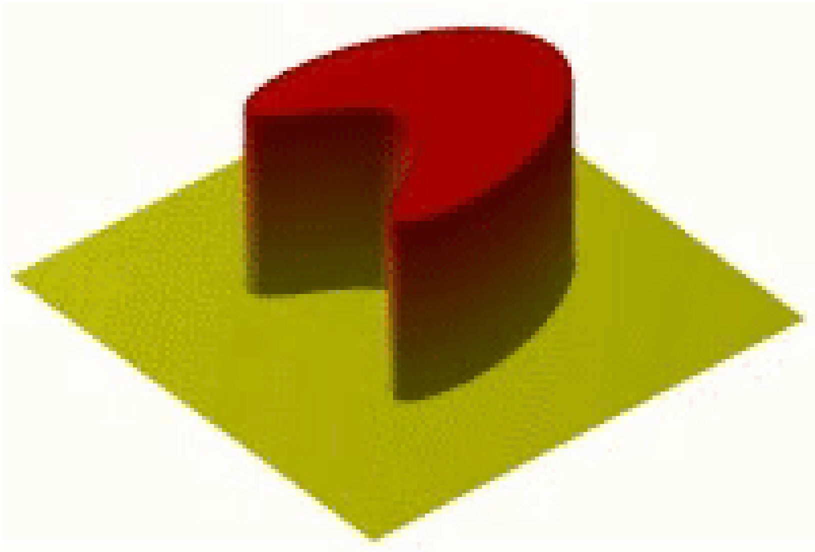
Heat Equation:

$$\Delta f = \frac{\partial f}{\partial t}$$

Steady state
heat equation:

$$\Delta f = 0$$

No heat flow = stable!
(Given suitable boundary conditions)



Heat diffusion simulated in a plane.

https://en.wikipedia.org/wiki/File:Heat_eqn.gif

Laplace equation (heat equation)

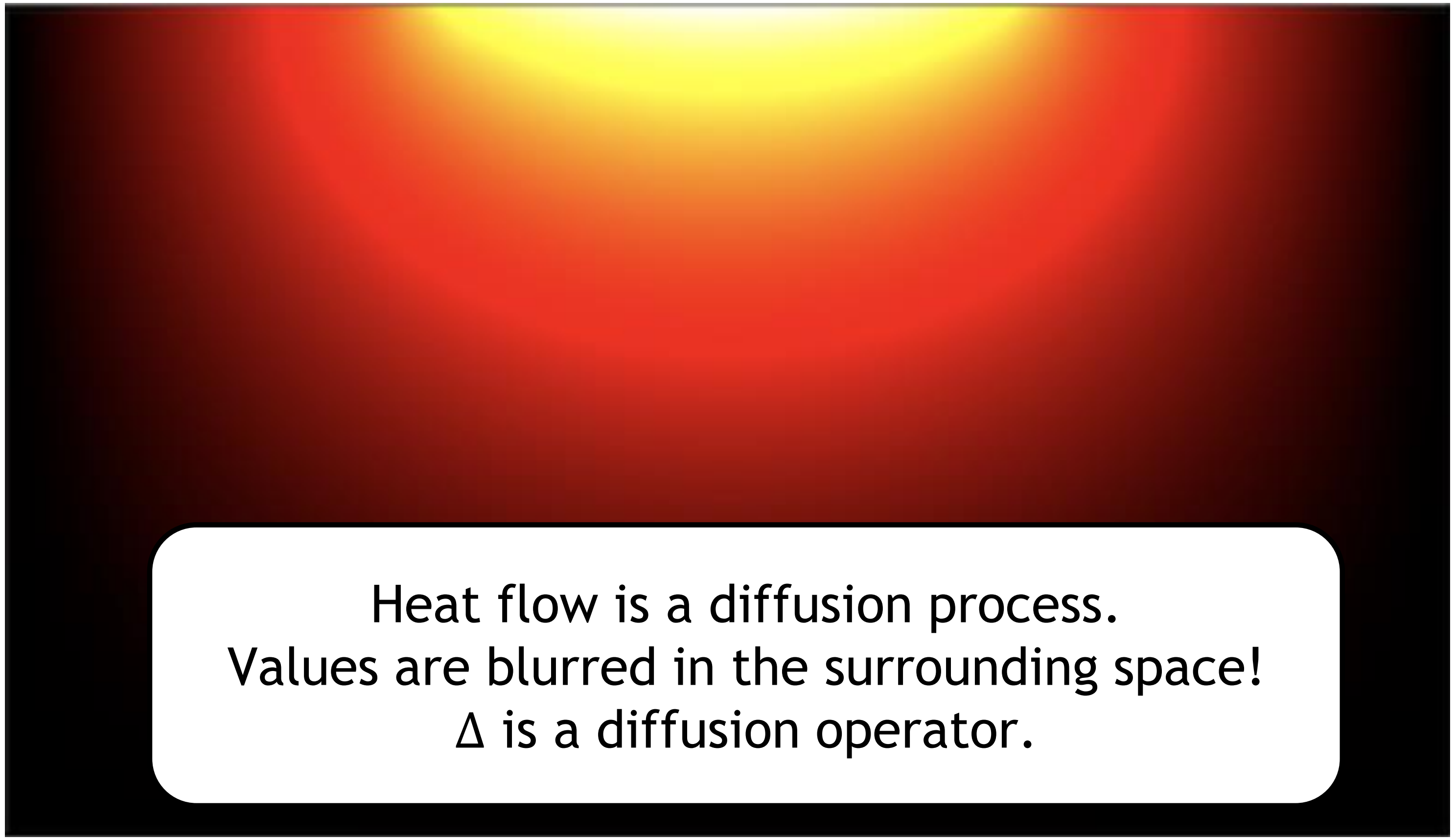
Heat Equation:

$$\Delta f = \frac{\partial f}{\partial t}$$

Steady state
heat equation:

$$\Delta f = 0$$

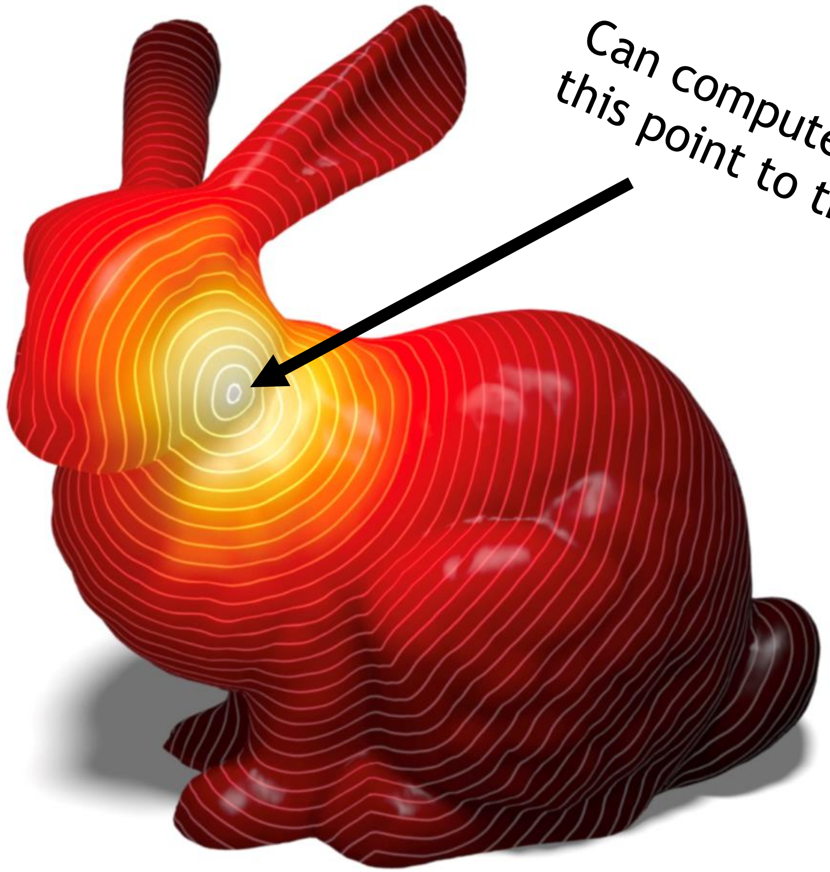
No heat flow = stable!
(Given suitable boundary conditions)



Heat equation on a surface

Poke a bunny with a hot needle 🔥

See how the heat distributes after a millisecond.



Can compute surface distance from this point to the rest of the surface.

How to do heat flow on a surface? 🤔

Geodesic in Heat [Crane et al. 2013]

Laplace-Beltrami Operator

- Extension of Laplace to functions on manifolds

$$f : \boxed{\mathcal{M}} \to \mathbb{R} \qquad \Delta f : \mathcal{M} \to \mathbb{R}$$

On the surface, not in $\mathbb{R}^3$

"A form of second derivative operator on the surface" 🙂

Laplace-Beltrami

Surface gradient operator

$$\Delta_{\mathcal{M}} f = \mathrm{div}_{\mathcal{M}} \nabla_{\mathcal{M}} f$$

function on surface M

Surface divergence operator

Laplace-Beltrami Operator

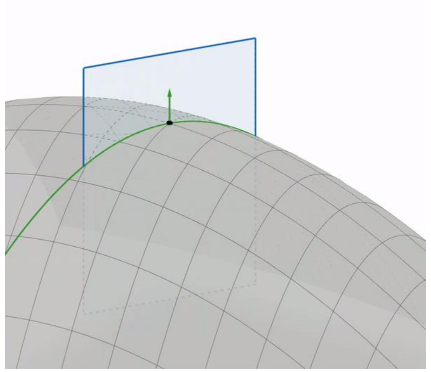
- For coordinate functions: $f(x, y, z) = x$ $\quad\quad \mathbf{p} = (x, y, z)$

Laplace-
Beltrami

"A form of second derivative operator on the surface" 🙂

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

Diffusing
position itself.



Recall:

For curves:

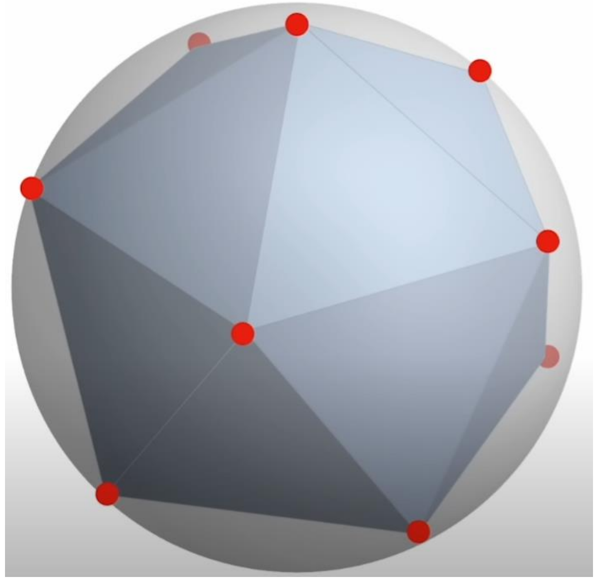
$$\gamma'' = \kappa \mathbf{n}(\gamma)$$

For surfaces:

$$\kappa_n(\varphi) = \kappa_1 \cos^2 \varphi + \kappa_2 \sin^2 \varphi.$$

Differential Geometry on Meshes

How can we discretize curvature, normal and operators? 🤔



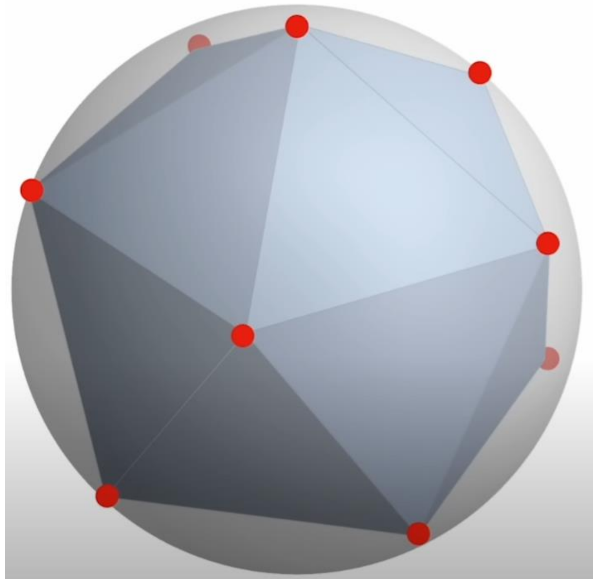
Differential Geometry on Meshes

- *Problem:* 🤨
  Discrete surfaces are not differentiable at edges/corners.

- *Assumption:* ✍️
  meshes are piecewise linear approximations of smooth surfaces

- *Idea:* 💡
  Can try fitting a smooth surface locally
  (say, a polynomial) and find differential quantities analytically

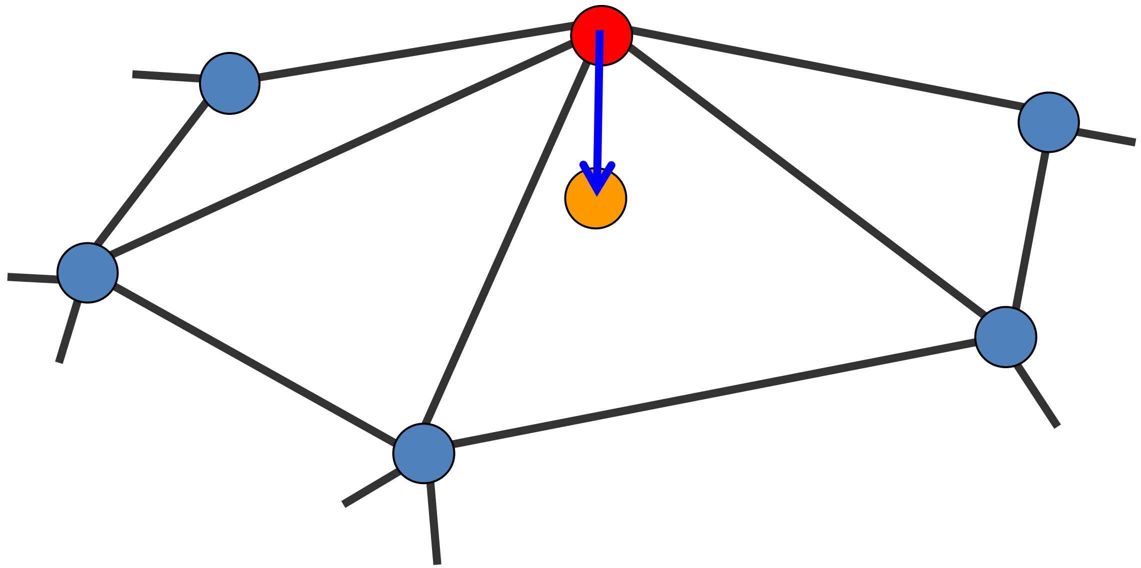
- *But:* ⚠️
  It is often too slow for interactive setting and error prone 😥



Discrete Differential Operators

Approach:
Approximate differential properties at point $\mathbf{v}$ as spatial average over local mesh neighborhood $N(\mathbf{v})$

- $\mathbf{v}$ = mesh vertex
- $N_k(\mathbf{v})$ = k-ring neighborhood



Discrete Laplace-Beltrami

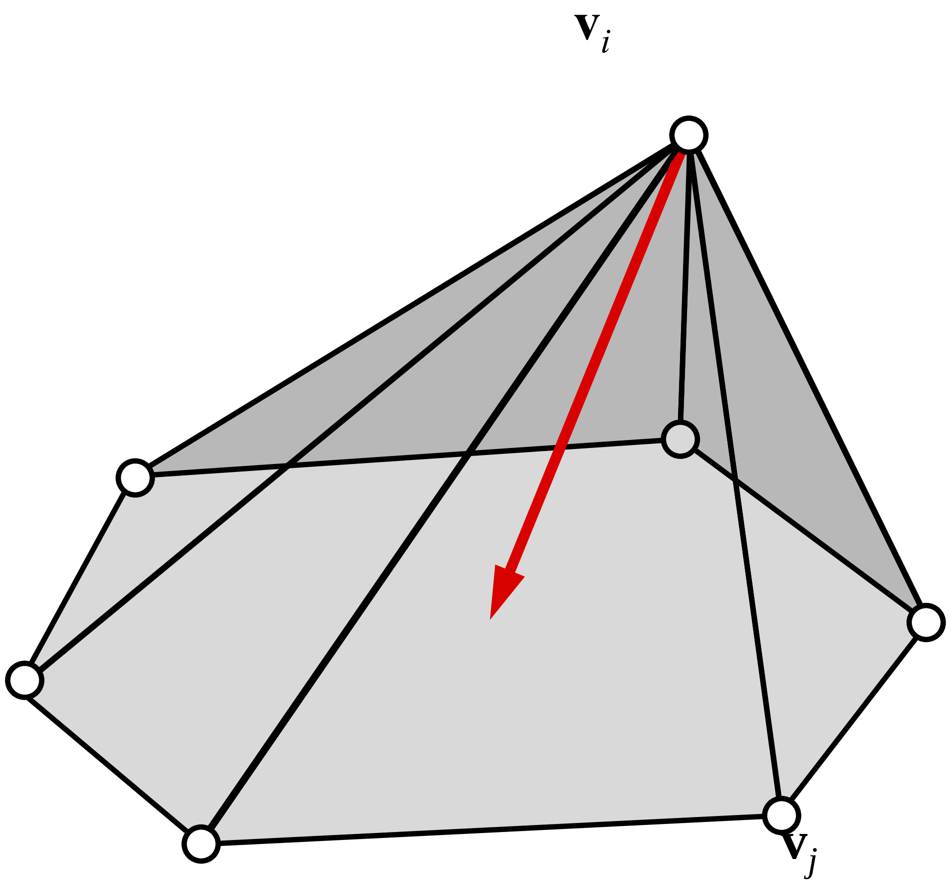
Smooth Laplace relationship

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

- *Uniform* discretization: $L(\mathbf{v})$ or $\Delta\mathbf{v}$

$$L_u(\mathbf{v}_i) = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} (\mathbf{v}_j - \mathbf{v}_i) = \left(\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{v}_j \right) - \mathbf{v}_i$$

"uniform Laplacian"

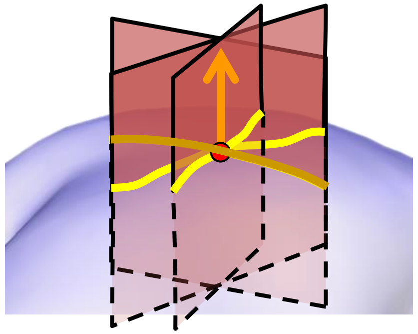


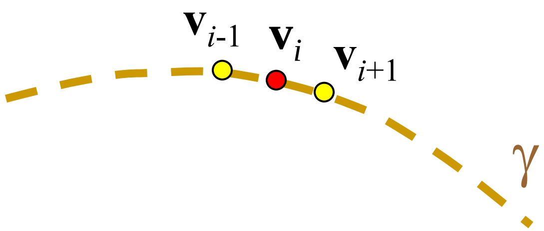
Local averaging, like in heat flow 🔥.

Discrete Laplace-Beltrami

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

- Intuition for uniform discretization – finite differences





$$H = \frac{1}{2\pi}\int_0^{2\pi} \kappa(\varphi)d\varphi$$

Arc-length parameterization assumption

$$\boldsymbol{\kappa}\mathbf{n} = \gamma''$$

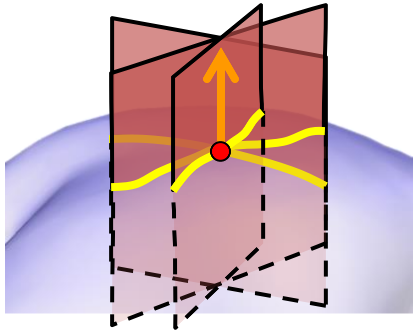
$$\gamma'' \approx \frac{1}{h}\left(\frac{\mathbf{v}_{i+1} - \mathbf{v}_i}{h} - \frac{\mathbf{v}_i - \mathbf{v}_{i-1}}{h}\right) = -\frac{2}{h^2}\left(\frac{1}{2}(\mathbf{v}_{i-1} + \mathbf{v}_{i+1}) - \mathbf{v}_i\right)$$

Finite difference approximation of the second derivative through finite difference approximation of the first derivative. 🤯

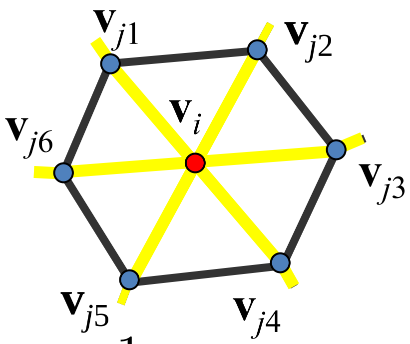
Discrete Laplace-Beltrami

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

- Intuition for uniform discretization – finite differences



$$H = \frac{1}{2\pi}\int_0^{2\pi}\kappa(\varphi)d\varphi$$



$$\frac{1}{2}(\mathbf{v}_{j_1} + \mathbf{v}_{j_4}) - \mathbf{v}_i \ +$$
$$\frac{1}{2}(\mathbf{v}_{j_2} + \mathbf{v}_{j_5}) - \mathbf{v}_i \ +$$
$$\frac{1}{2}(\mathbf{v}_{j_3} + \mathbf{v}_{j_6}) - \mathbf{v}_i =$$
$$= \frac{1}{2}\sum_{j\in\mathcal{N}(i)}\mathbf{v}_j - 3\mathbf{v}_i = 3\left(\frac{1}{6}\sum_{j\in\mathcal{N}(i)}\mathbf{v}_j - \mathbf{v}_i\right)$$

We call this „uniform Laplacian operator“

$\boxed{L_u(\mathbf{v}_i)}$

👍Depends only on connectivity = simple and efficient

👎Bad approximation for irregular triangulations

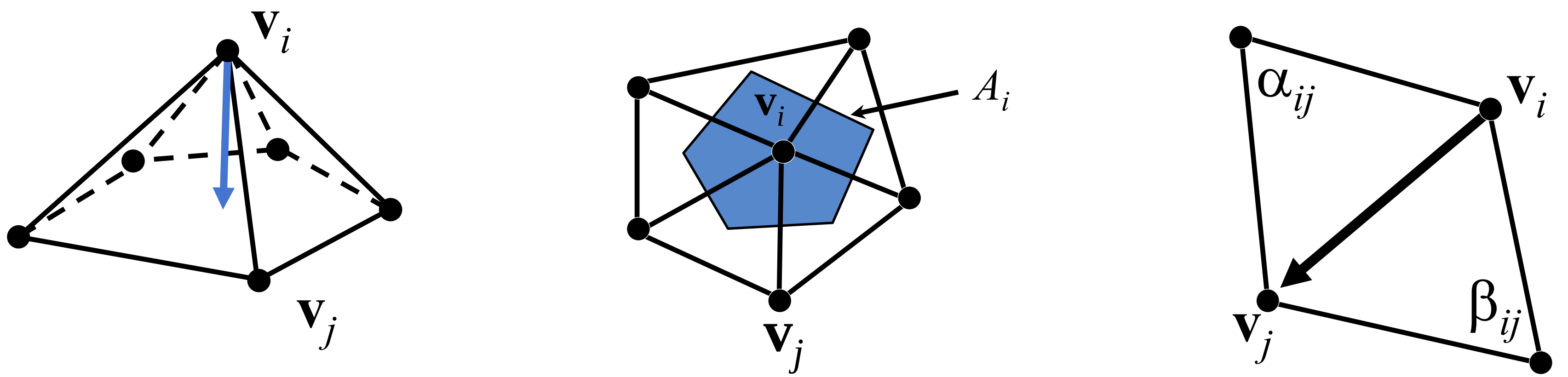
Finite difference approximation of the second derivative through finite difference approximation of the first derivative. 🤯

Discrete Laplace-Beltrami

- Cotangent formula

$$L_c(\mathbf{v}_i) = \frac{1}{A_i} \sum_{j \in \mathcal{N}(i)} \frac{1}{2} (\cot \alpha_{ij} + \cot \beta_{ij})(\mathbf{v}_j - \mathbf{v}_i)$$

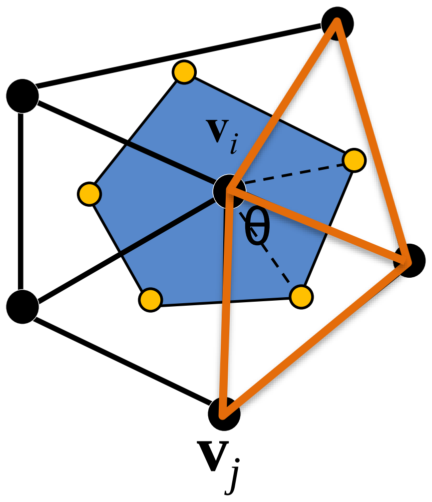
"cotan Laplacian"



Vertex area

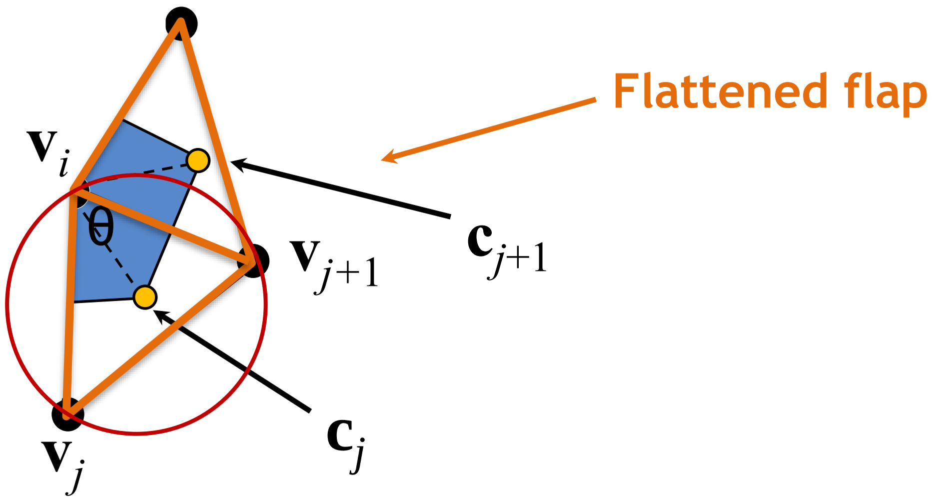
Cotan weights

Voronoi Vertex Area



- Unfold the triangle flap onto the plane (without distortion)

Voronoi Vertex Area



$$\mathbf{c}_j = \begin{cases} \text{circumcenter of } \triangle(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_{j+1}) & \text{if } \theta < \pi/2 \\ \text{midpoint of edge } (\mathbf{v}_j, \mathbf{v}_{j+1}) & \text{if } \theta \geq \pi/2 \end{cases}$$

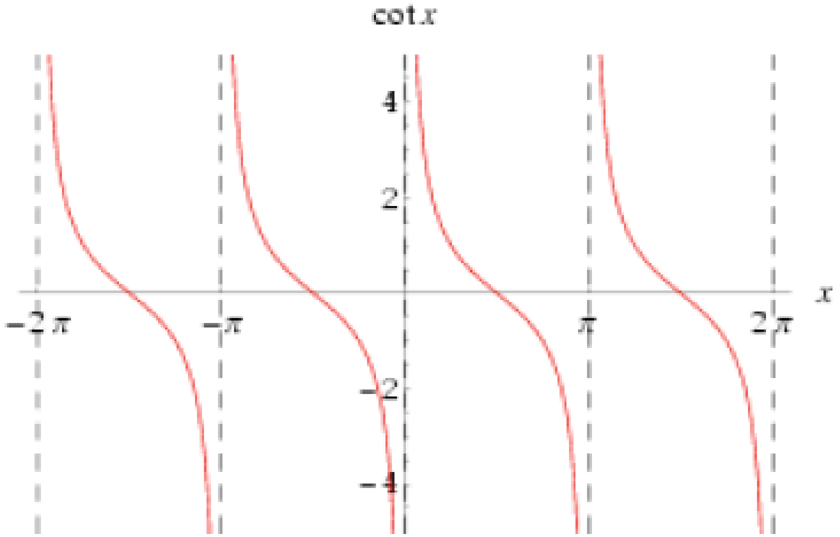
$$A_i = \sum_j \text{Area}\left(\triangle(\mathbf{v}_i, \mathbf{c}_j, \mathbf{c}_{j+1})\right)$$

Discrete Laplace-Beltrami

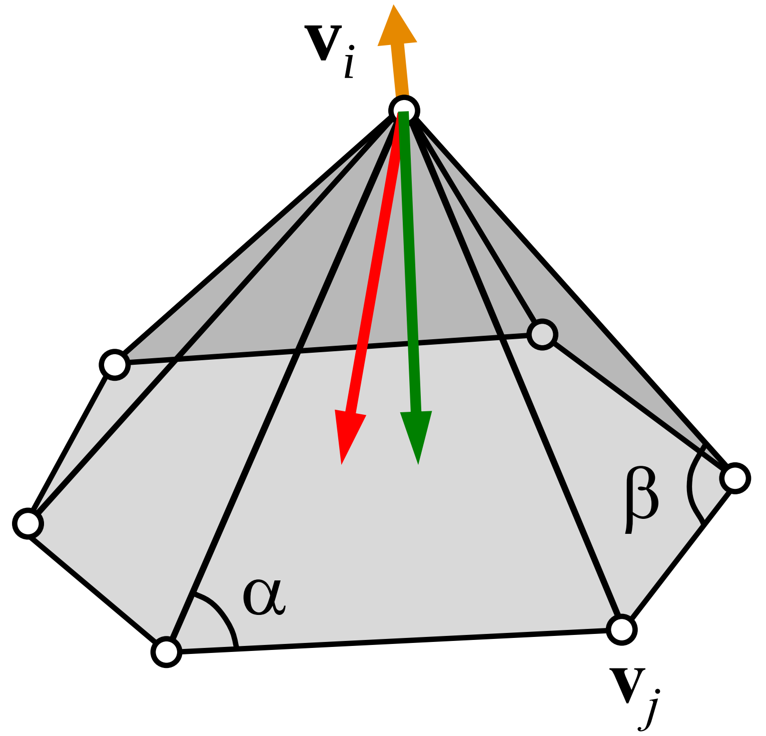
- Cotangent formula

$$L_c(\mathbf{v}_i) = \frac{1}{A_i} \sum_{j \in \mathcal{N}(i)} \frac{1}{2}(\cot \alpha_{ij} + \cot \beta_{ij})(\mathbf{v}_j - \mathbf{v}_i)$$

  - 👍
    - Accounts for mesh geometry
    - Can be derived using linear Finite Elements
    - Nice property: gives zero for planar 1-rings!
  - 👎
    - Potentially negative/ infinite weights



Discrete Laplace-Beltrami



- Normal
- Uniform Laplacian $\mathrm{L}_u(\mathrm{v}_i)$
- Cotangent Laplacian $\mathrm{L}_c(\mathrm{v}_i)$

- For nearly equal edge lengths
  Uniform ≈ Cotangent

Cotan Laplacian allows computing discrete normals 🥳

$$\Delta_M \mathbf{p} = -H\mathbf{n} \;\Rightarrow\; \frac{\Delta_M \mathbf{p}}{||\Delta_M \mathbf{p}||} = \pm\mathbf{n}$$

Also allows us to compute H 🤩

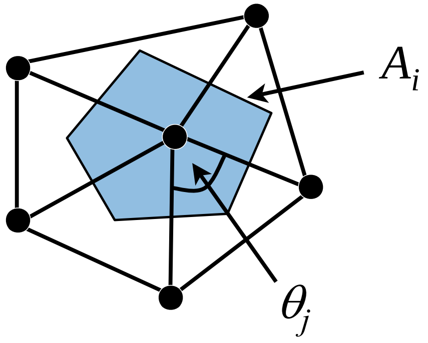
$$\Delta_M \mathbf{p} = -H\mathbf{n} \;\Rightarrow\; H = -\langle \Delta_M \mathbf{p}, \mathbf{n} \rangle$$

Discrete Curvatures

- Discrete Mean curvature (derived from $\Delta_M \mathbf{p} = -H\mathbf{n}$)

$$H = -\langle \mathbf{n}, \Delta_M(\mathbf{p}) \rangle$$

- Principal curvatures (Using $H\&K$. Rearrange $H = \frac{1}{2}(\kappa_1 + \kappa_2)$ and $K = \kappa_1 \kappa_2$)



$$\kappa_1 = H - \sqrt{H^2 - K} \qquad \kappa_2 = H + \sqrt{H^2 - K}$$

- Gaussian curvature

$$K(\mathbf{v}_i) = \ ?$$ 🤔

Discrete Gauss-Bonnet Theorem

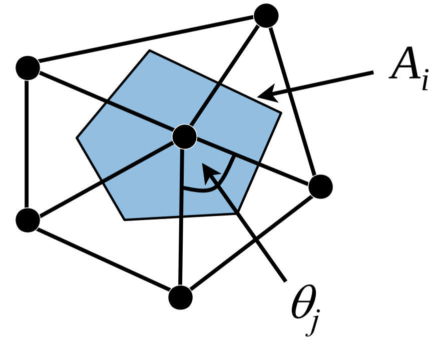
- Total Gaussian curvature should be fixed for a given topology

$$\int_M K dA = 2\pi\chi(M) \xrightarrow{\text{discretize}} \sum_i A_i K(\mathbf{v}_i) = 2\pi\chi(M)$$

😲 Needs to be 0 for flat objects and dependent on intrinsic values

Proposal: $$K(\mathbf{v}_i) = \frac{1}{A_i}\left(2\pi - \sum_j \theta_j\right)$$

Area weighted local angle defect from plane



Discrete Gauss-Bonnet Theorem

Continuous **Gauss-Bonnet Theorem**

$$\int_M K dA = 2\pi\chi(M)$$

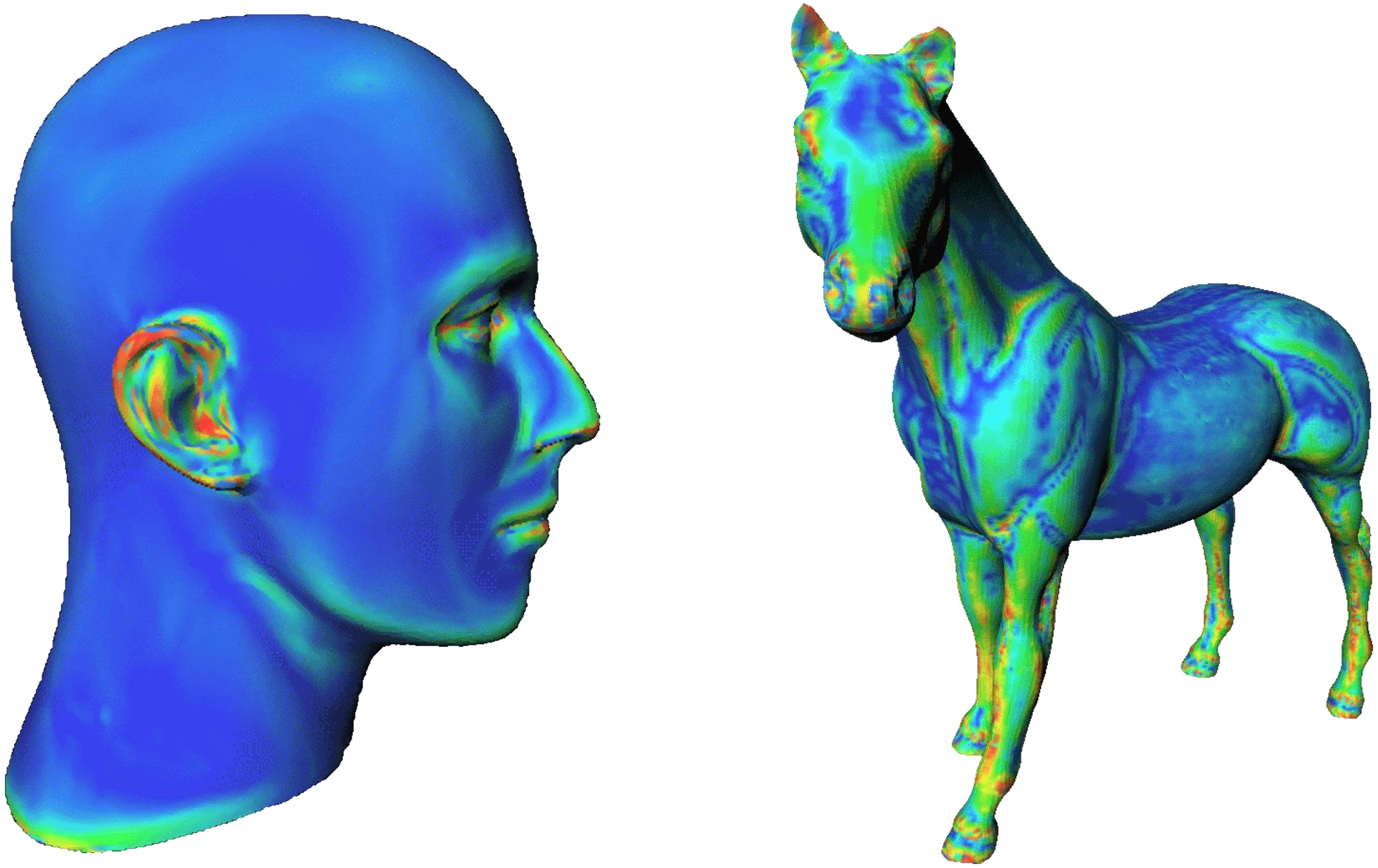
Discrete **Gauss-Bonnet Theorem**

$$\sum_i A_i K(\mathbf{v}_i) = 2\pi\chi(M)$$

if: $K(\mathbf{v}_i) = \frac{1}{A_i}(2\pi - \sum_j \theta_j)$

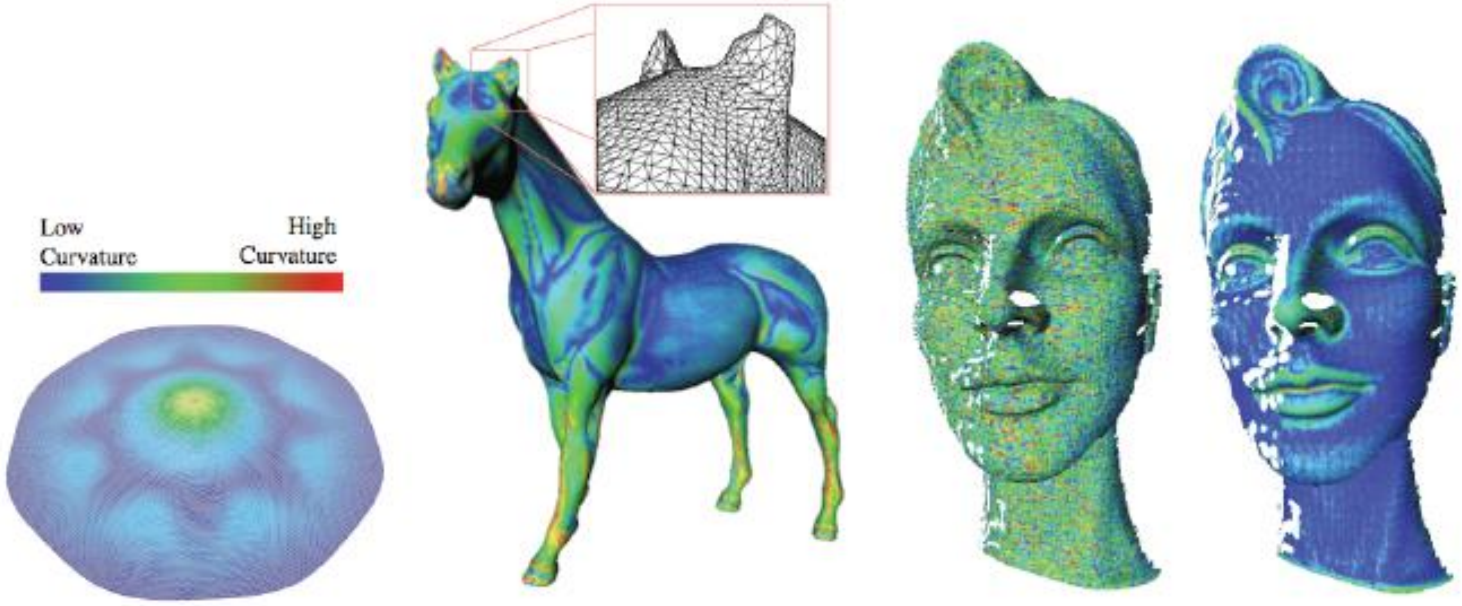
Another structure preservation!

Example: Discrete Mean Curvature



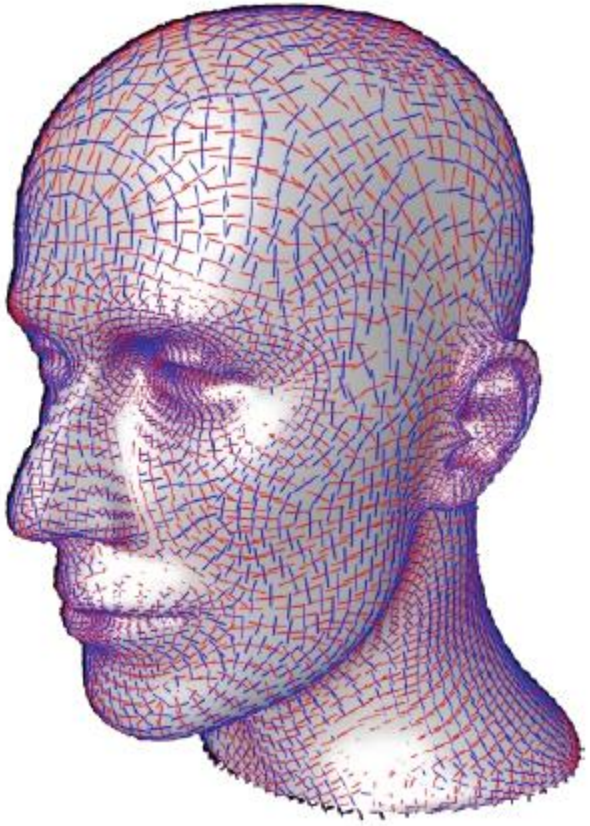
Links and Literature

- M. Meyer, M. Desbrun, P. Schroeder, A. Barr
  Discrete Differential-Geometry Operators for Triangulated 2-Manifolds,
  VisMath, 2002



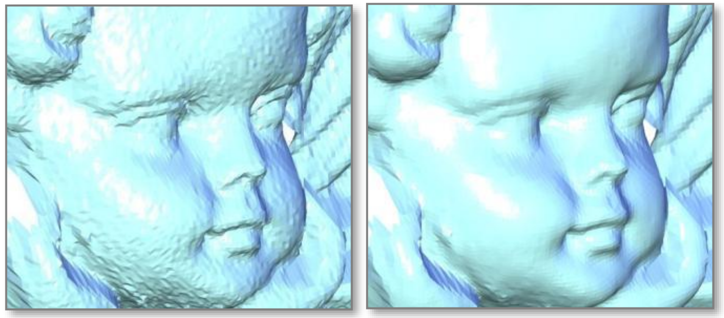
Links and Literature

- libigl implements many discrete differential operators
- See the tutorial! 🥳
- https://libigl.github.io/tutorial/

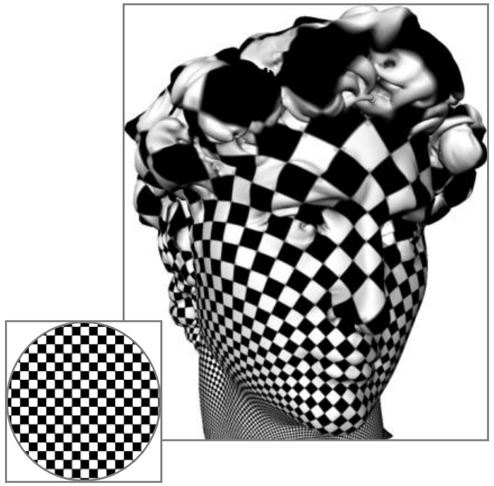


principal directions

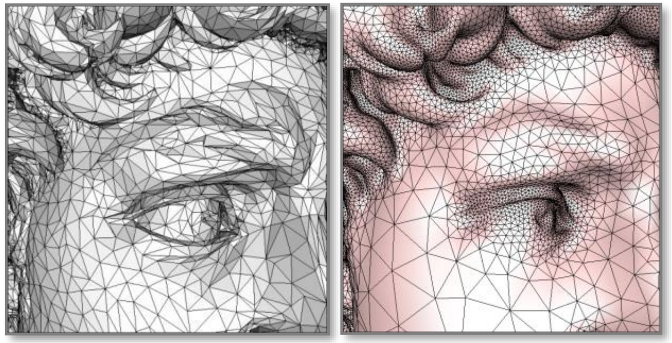
Surface Processing– Topics



Smoothing



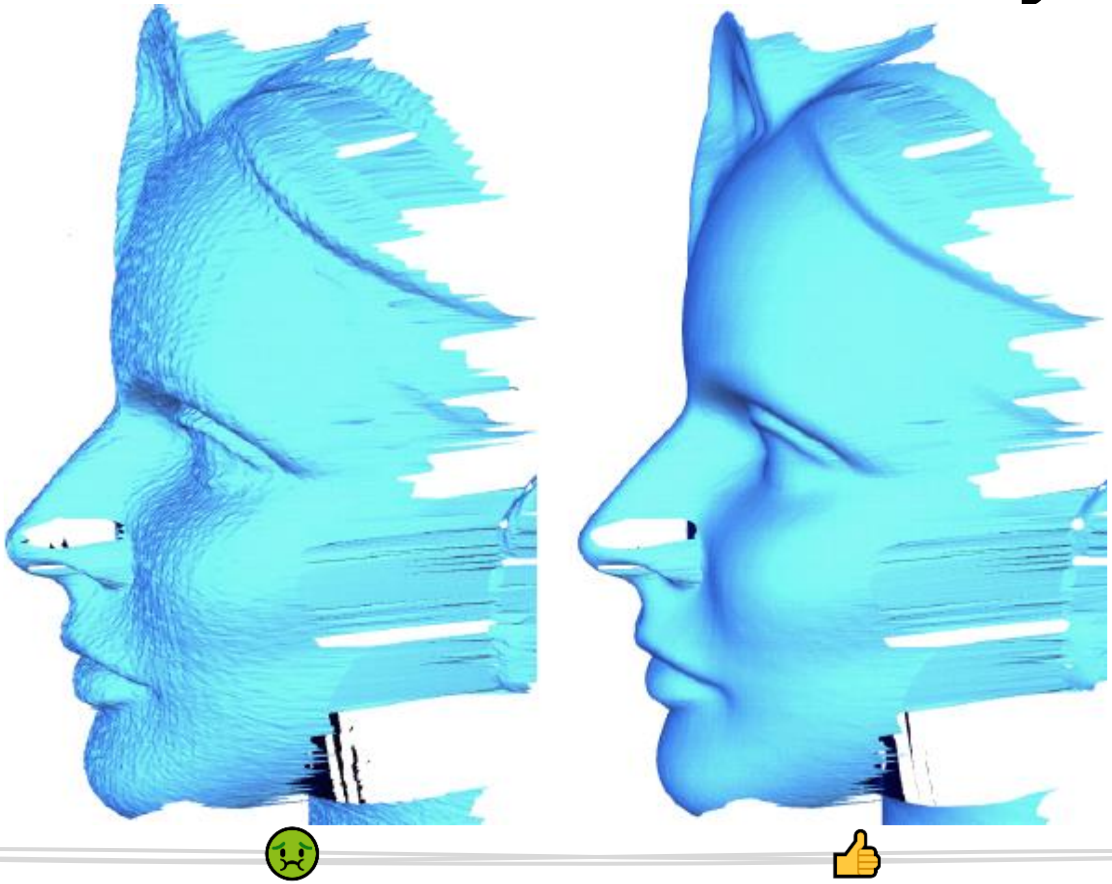
Parametrization



Remeshing

Surface Smoothing – Motivation

- Scanned surfaces can be noisy

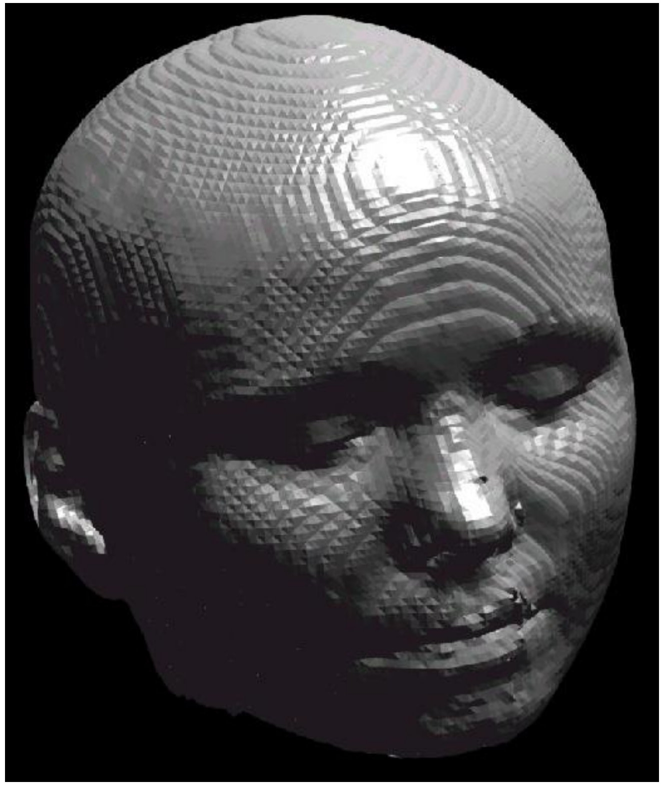


Surface Smoothing – Motivation

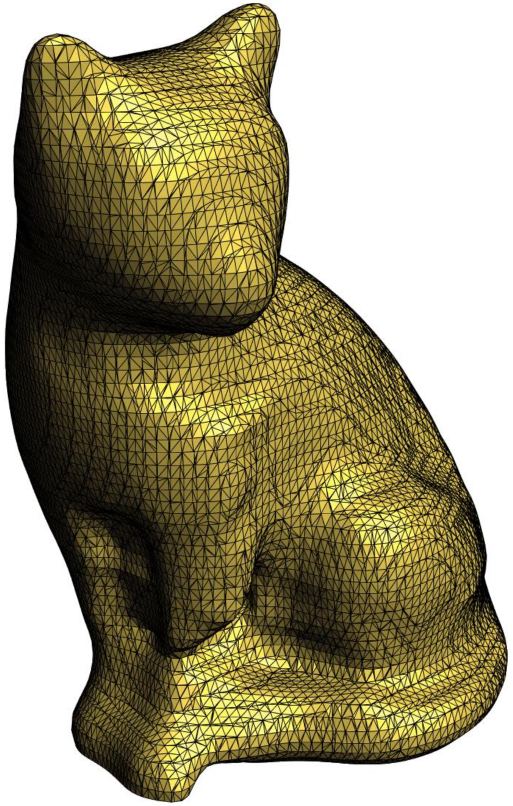
- Scanned surfaces can be noisy

Mesh Fairing – Motivation

- Marching Cubes meshes can be ugly 💀



Why so damn ugly? 👹

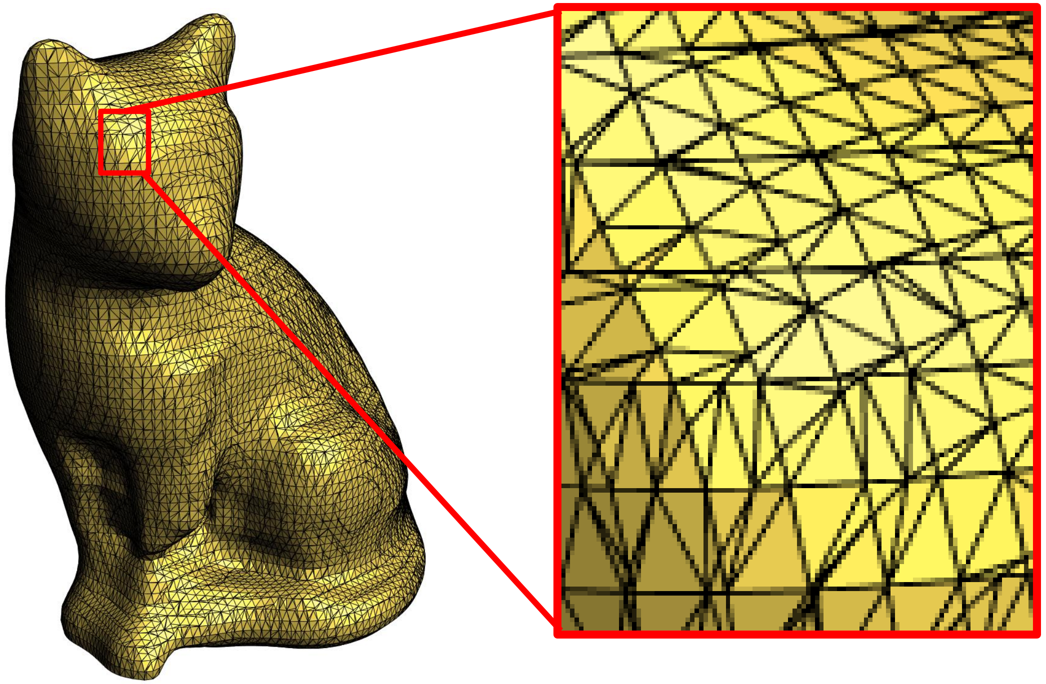


Why so disgusting? 🤮

What is the problem with "bad triangles"? 🤔

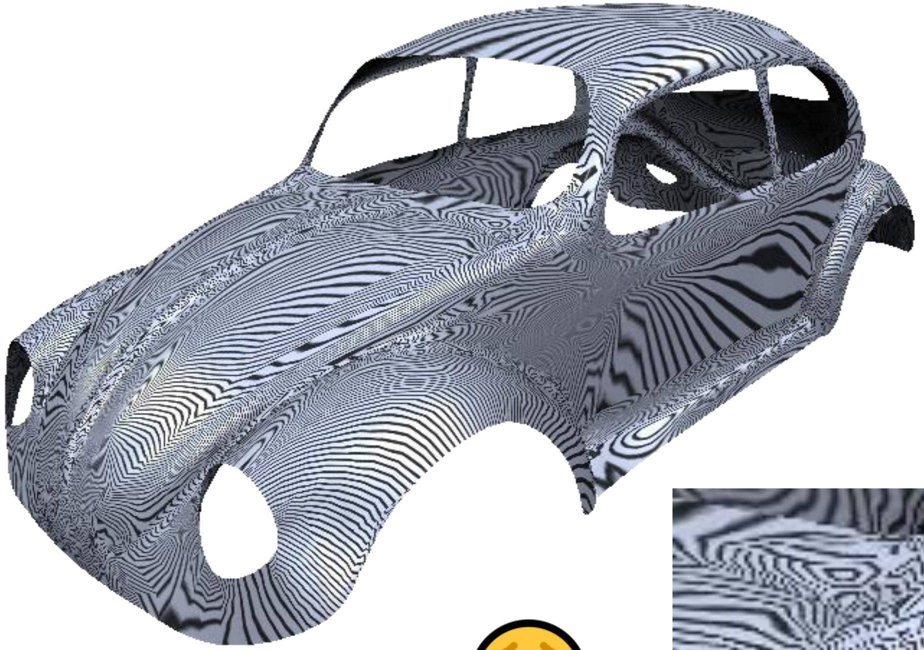
Important rule

👎 Bad triangles = Bad computations 😡



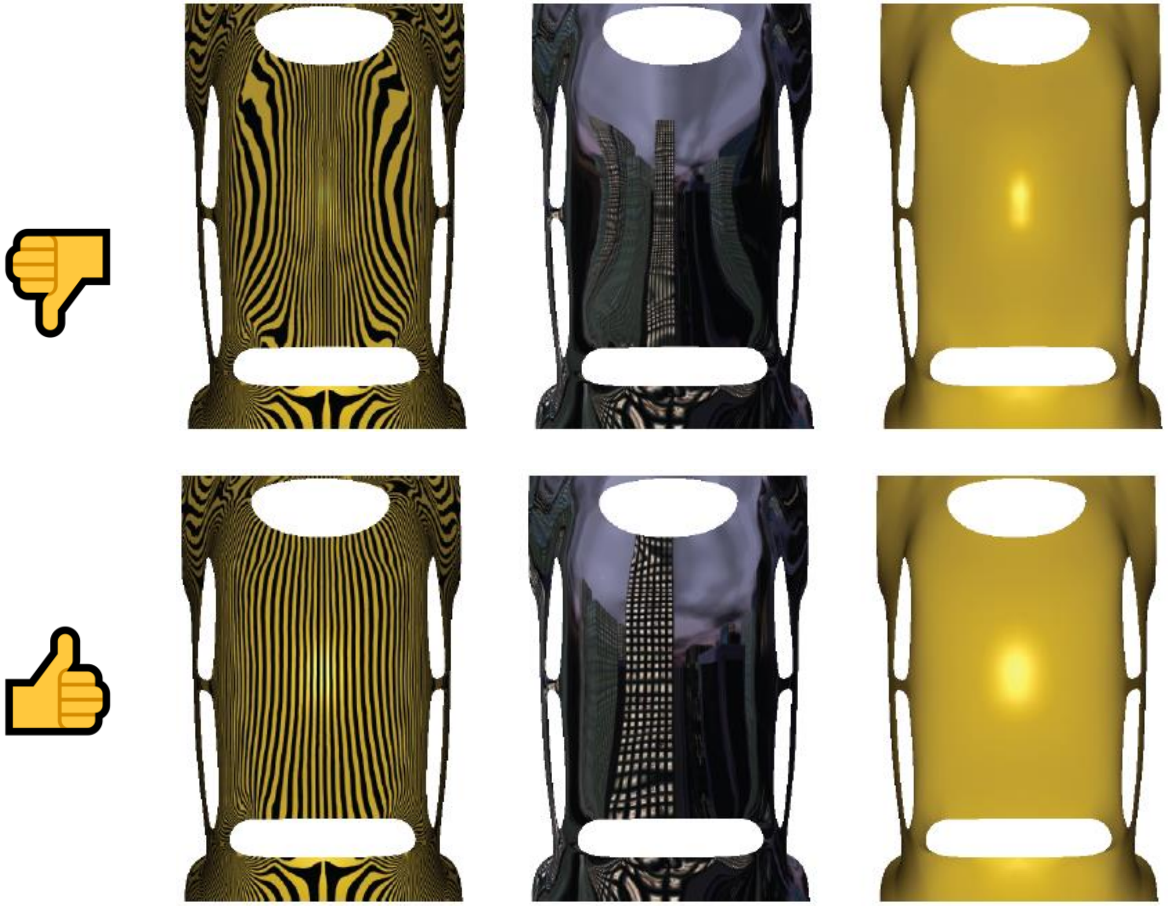
- More triangles than needed 🔥
  - Waste of memory
  - Waste of computation time
- Long triangles 📏
  - Numerical issues
    - e.g. Normal computation
  - Bad Laplacian Matrix
    - Negative weights
    - Large weights
    - Nearly singular matrix
  - Bad for Algorithms
    - Performance & accuracy

Visual Assessment of Surface Smoothness



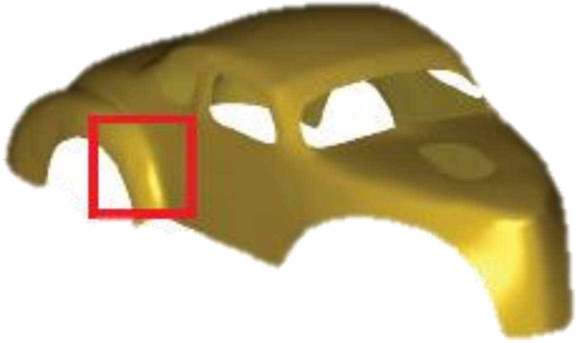
Reflection Lines as an Inspection Tool

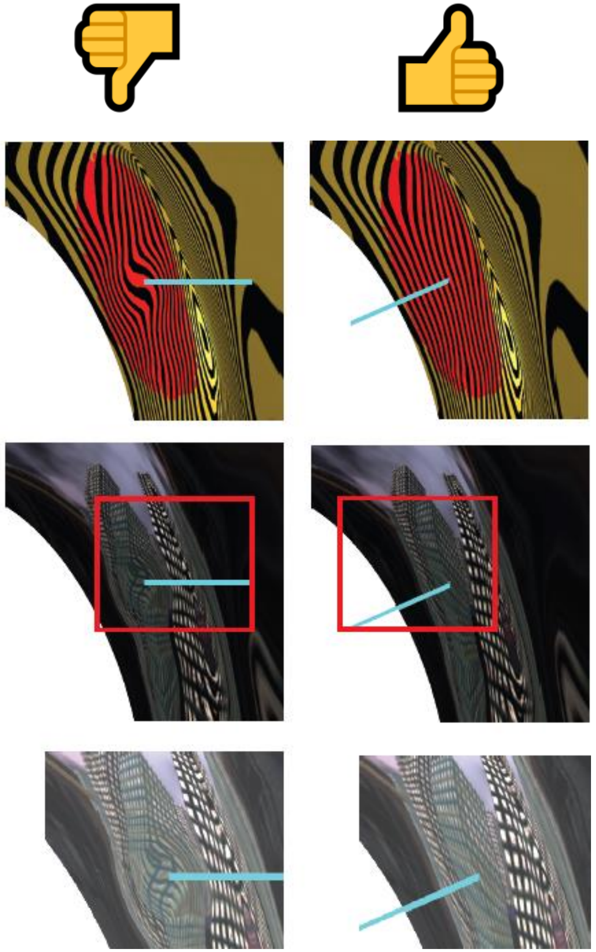
- [Shape optimization using reflection lines](#)
  E. Tosun, Y. I. Gingold, J. Reisman, D. Zorin
  Symposium on Geometry Processing 2007



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





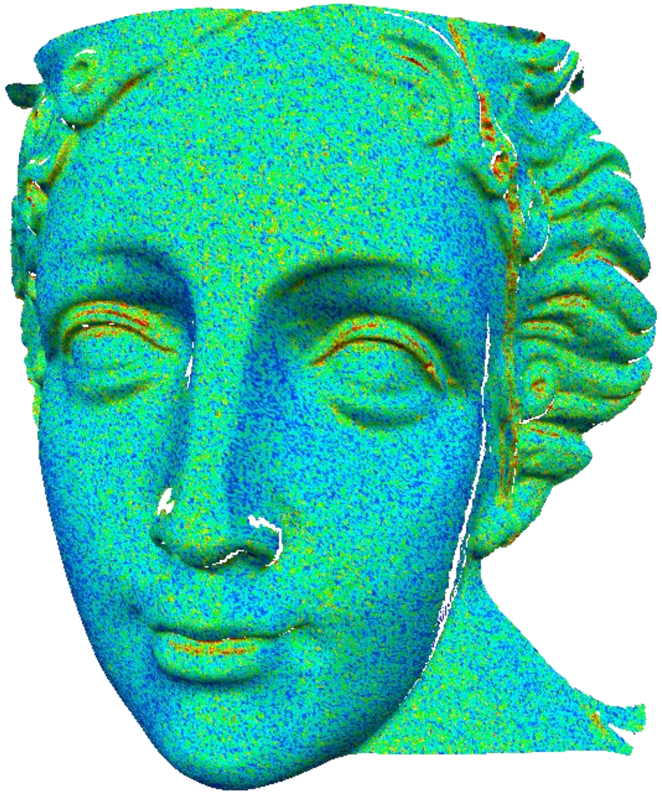
How to measure smoothness?

igl

ETH zürich

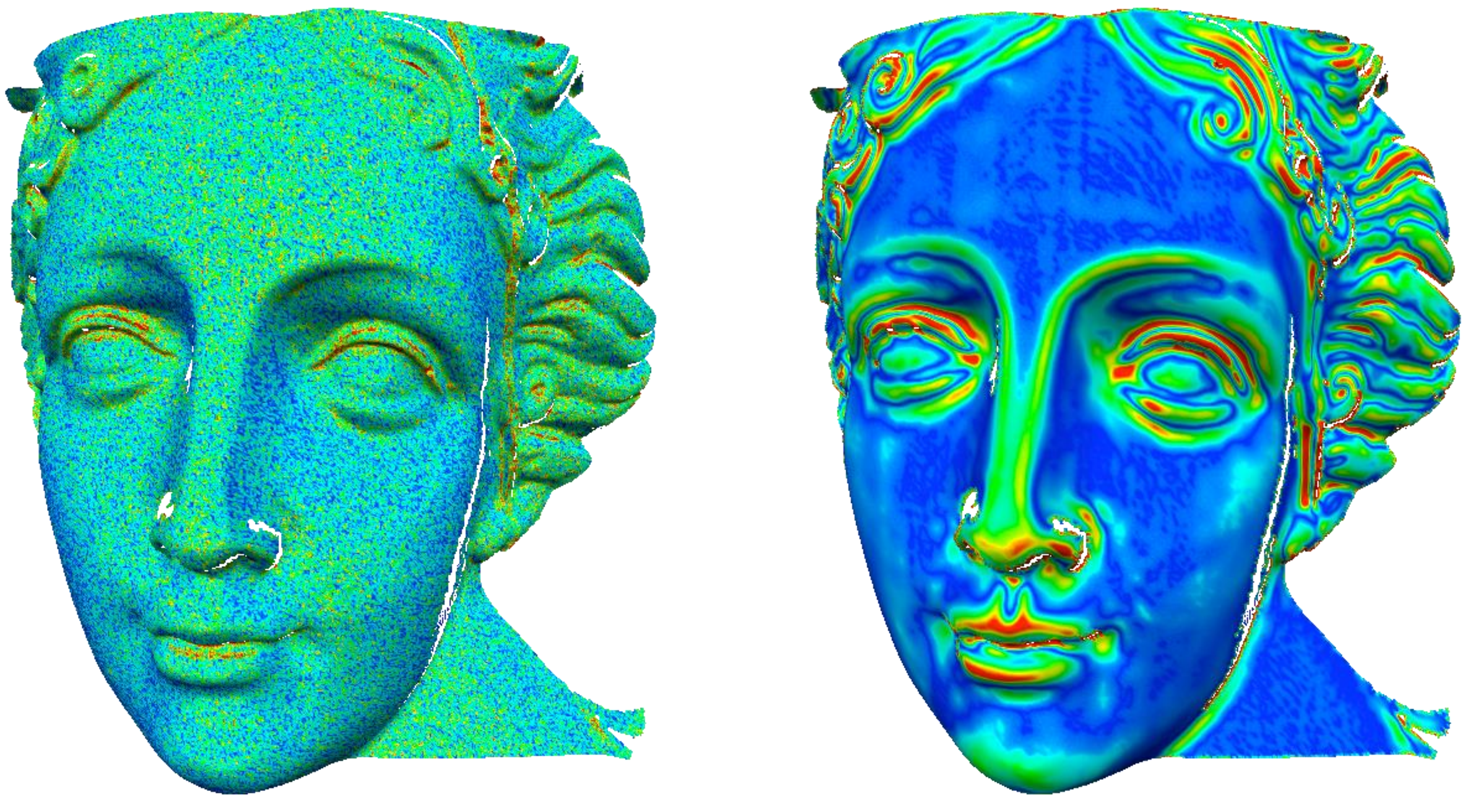
Curvature and Smoothness

Curvature and Smoothness



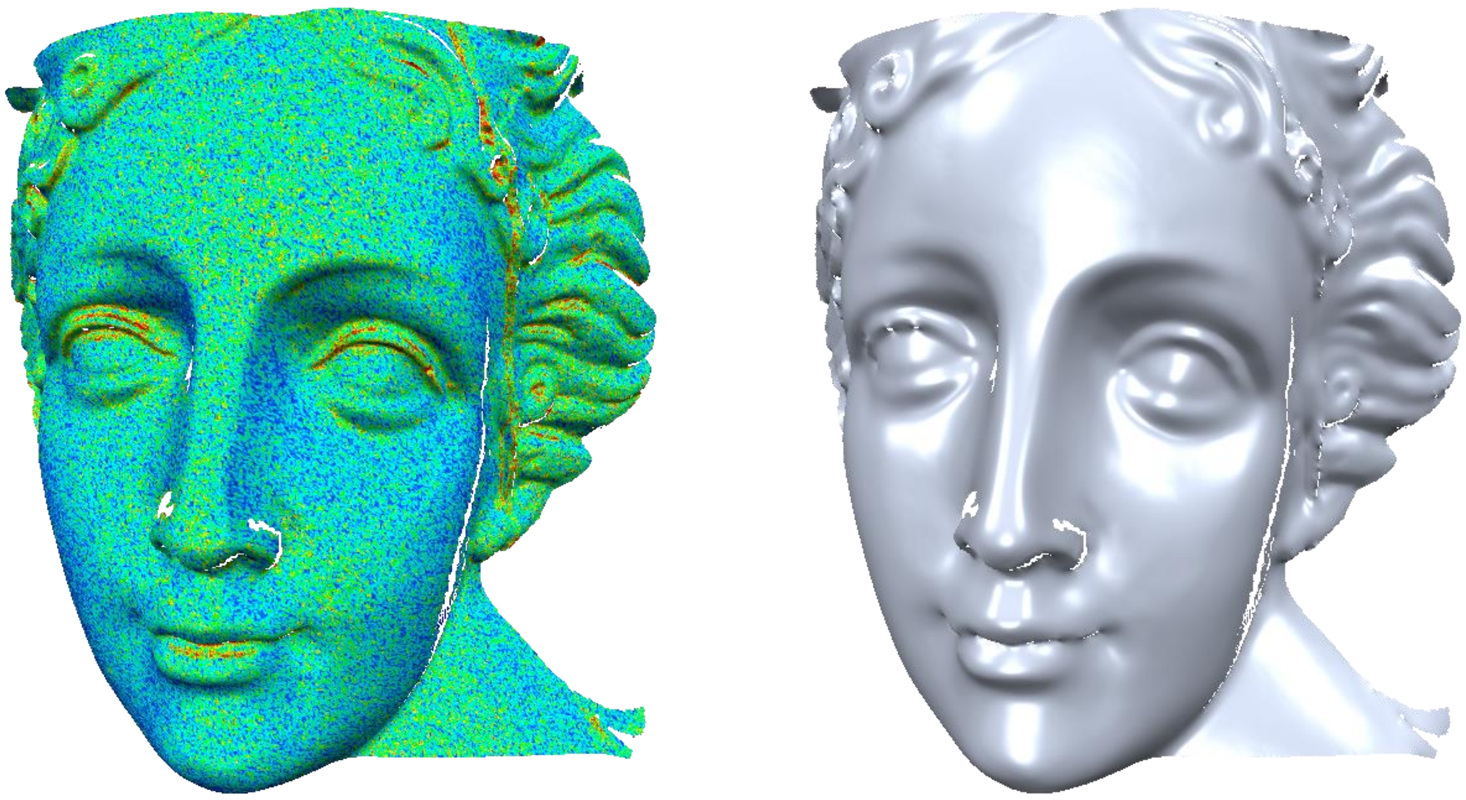
mean curvature plot

Curvature and Smoothness



mean curvature plot

Curvature and Smoothness



mean curvature plot

Curvature and Smoothness

What is *smoothing*? 🤔

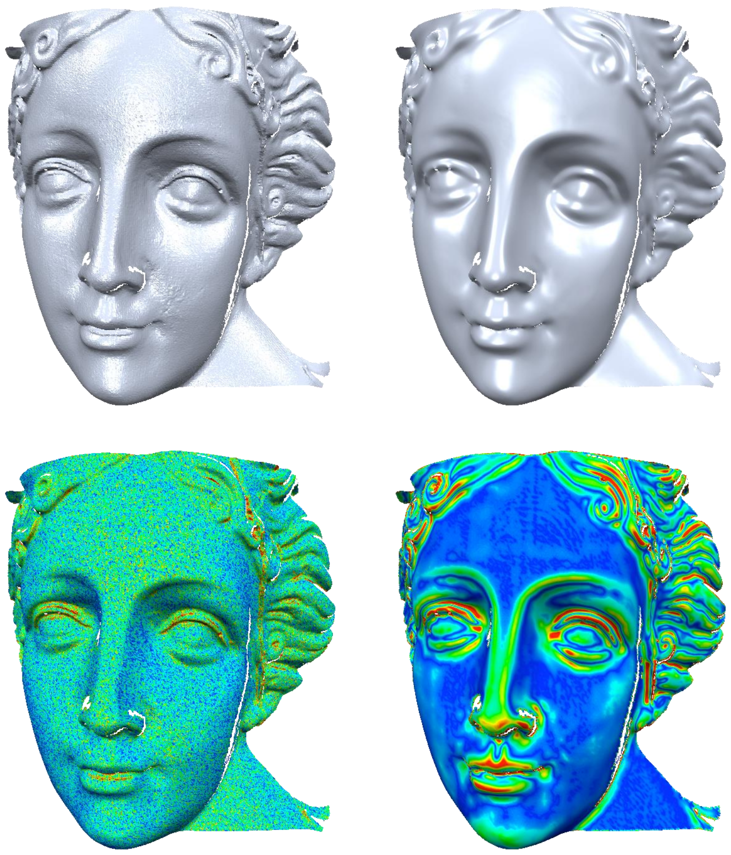
~~Reducing total curvature?~~

Gauss-Bonnet theorem says it is invariant (for closed surfaces) 😵 💫

VS.

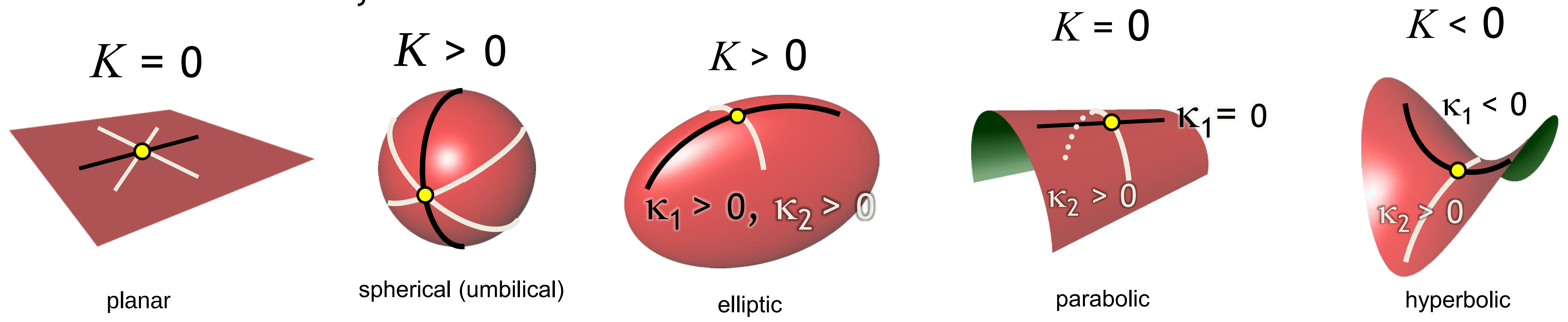
make curvature vary less?

This one is a good choice! 👌



Which curvature to change for smoothing?

Remember Smooth theory Gauss curvature:

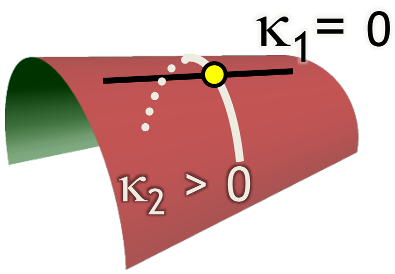


So can we use smooth surfaces by reducing gauss curvature variation?

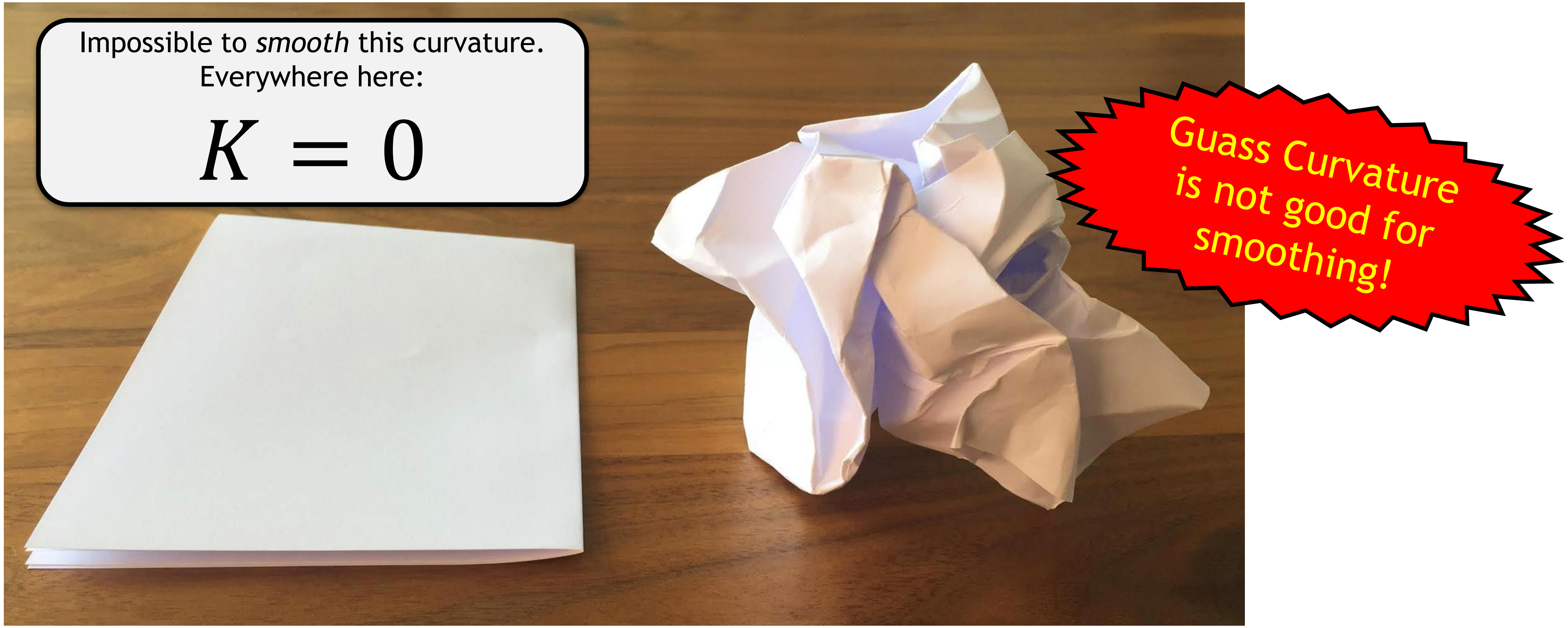
🤔

Can we use Gauss Curvature?

$K = 0$



parabolic



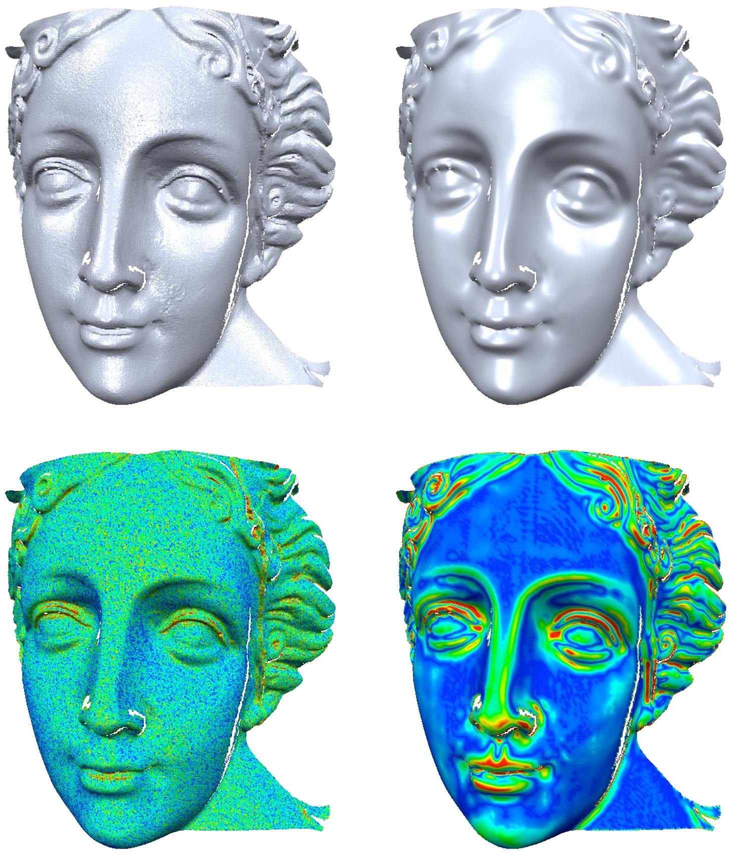
These two surfaces are intrinsically the same ⇒ Equal Gauss curvature!

Which curvature?

- Principal curvatures $\kappa_{\min}, \kappa_{\max}$ ✗
  - Nonlinear and “discontinuous” operator
- Gauss curvature K ✗
  - Intrinsic only, insensitive to embedding
- Mean curvature H ✓
  - Easy to compute by Laplacian Δ

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

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



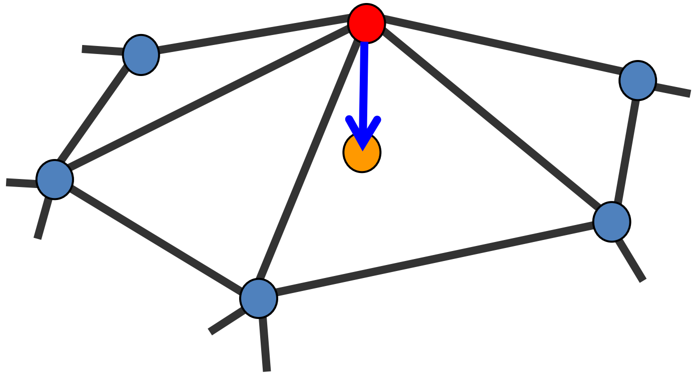
Laplace as a linear operator

Recap: discrete Laplace-Beltrami $\Delta_{\mathcal{M}}$

- Assumption: smoothing = local averaging/diffusion
- Heat equation operator Δ_M is linear $\Rightarrow$ Δ_M has matrix form

Matrix form with unknown entries:

$$\Delta_{\mathcal{M}}(\mathbf{p}_i) = \delta_i = \underbrace{\frac{1}{W_i}}_{\text{vertex weight}} \underbrace{\sum_{j \in \mathcal{N}(i)}}_{\text{local}} \underbrace{w_{ij}(\mathbf{p}_j - \mathbf{p}_i)}_{\text{weighted differences}}$$

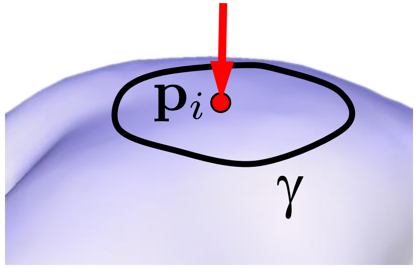


L-B: Weighting Schemes

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

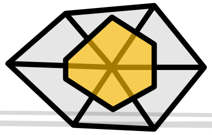
- Ignore geometry

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

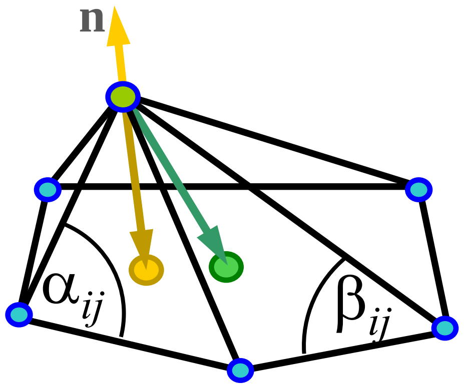


- Integrate over Voronoi region of the vertex

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



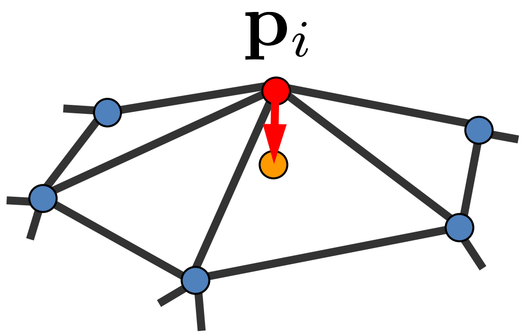
$$W_i = A_i$$



Laplacian Matrix

- The transition between *xyz* and δ is linear:

$$\mathbf{p} = (\mathbf{x}, \mathbf{y}, \mathbf{z})$$



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

$$\mathbf{L}\,\mathbf{x} = \delta_x$$

$$\mathbf{L}\,\mathbf{y} = \delta_y$$

$$\mathbf{L}\,\mathbf{z} = \delta_z$$

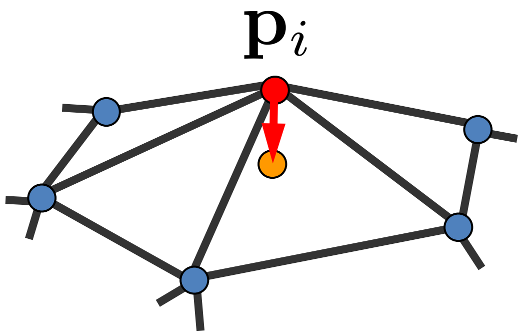
$$\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^n$$

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

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

Laplacian Matrix

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



Matrix Δ		Diagonal	Symmetric
$\mathbf{L}$	=	$\mathbf{M}^{-1}$	$\mathbf{L}_w$

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

How to do the smoothing?

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

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

Idea 1:

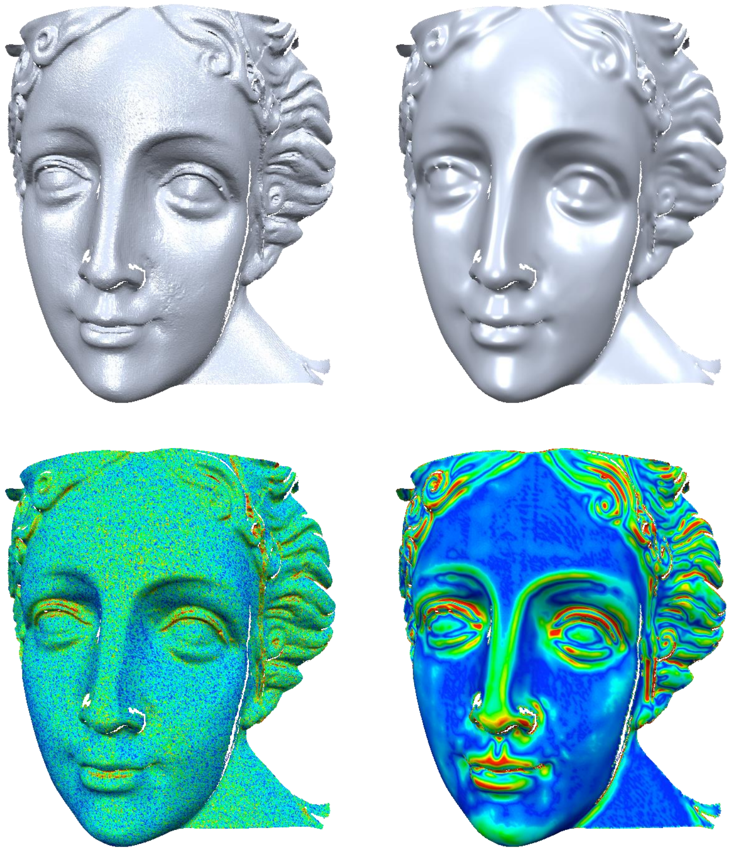
- Smooth $H \Rightarrow \tilde{H}$
- Construct surface that has mean curvature $\tilde{H}$
  - Problem: H doesn't define the surface, $\mathbf{n}$ nonlinear in $\mathbf{p}$

✗

Idea 2:

- If $\Delta_{\mathcal{M}}\mathbf{p}$ small, then H also small.
- Minimize $\Delta_{\mathcal{M}}\mathbf{p}$ by flowing along $\mathbf{n}$
  - Classic gradient descent "*flow*"!
  - Feasible because linear operator!

✓

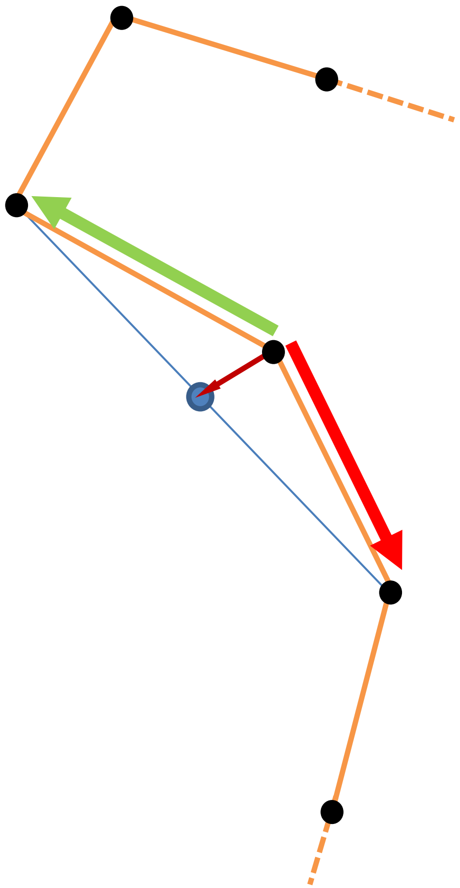


Smoothing by flowing

Example – smoothing curves

- Uniform Laplace in 1D = second derivative.

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



- In matrix-vector form for the whole curve

$L\mathbf{p}$

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

$$L = \frac{1}{2}\begin{pmatrix} -2 & 1 & & & 0 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 0 & & & 1 & -2 \end{pmatrix}$$

Defined by point connectivity

Example – smoothing curves

- Flow to reduce curvature:

„step size“ $0 < \lambda < 1$

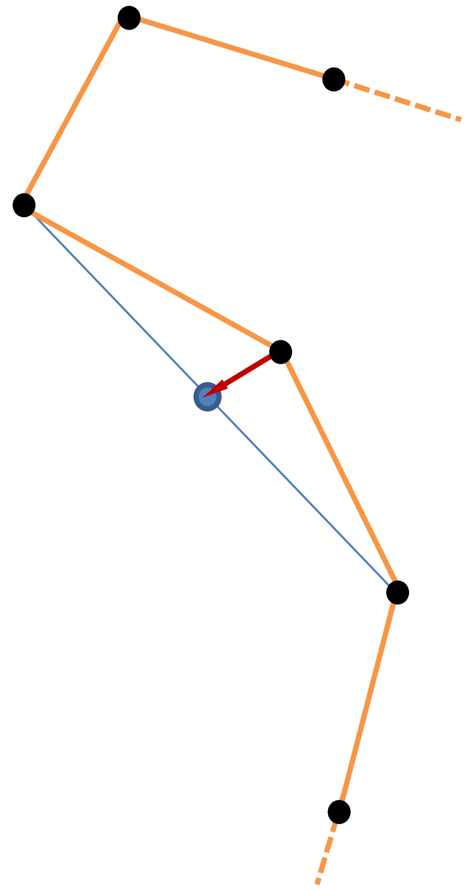
$$\tilde{\mathbf{p}}_i = \mathbf{p}_i + \lambda \frac{d^2}{ds^2}(\mathbf{p}_i)$$

- Matrix-vector form:

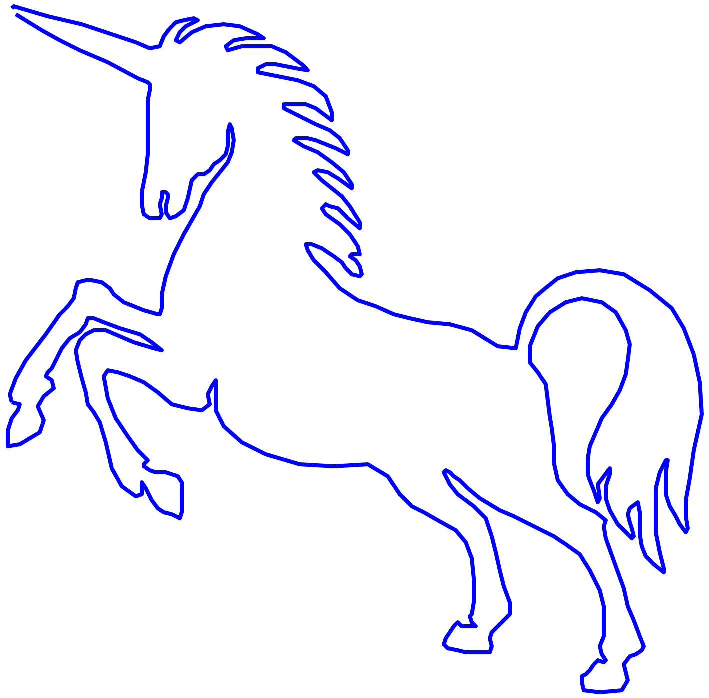
$$\tilde{\mathbf{p}} = \mathbf{p} + \lambda L \mathbf{p}, \quad \mathbf{p} \in \mathbb{R}^{n\times 2}$$

🙂 This flow will reduce curvature step by step

😨 Problem May shrink the shape, can be slow

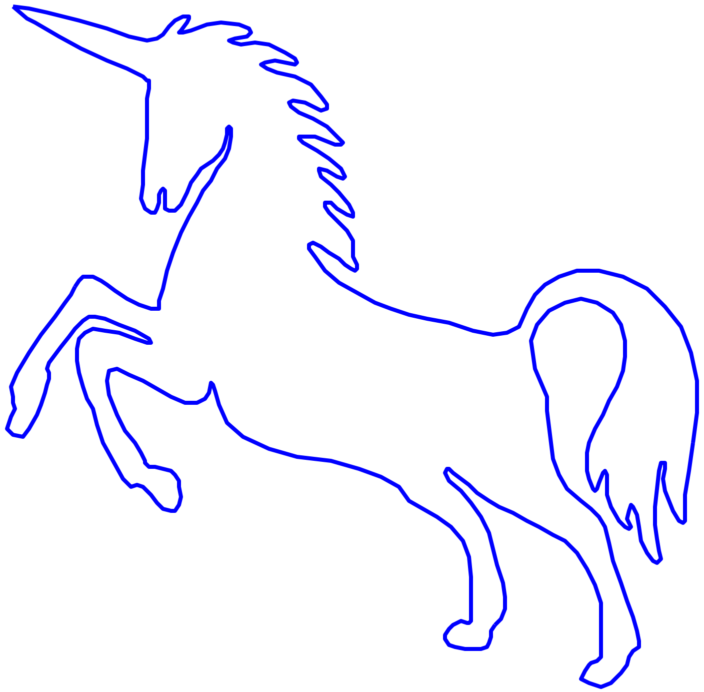


Filtering Curves



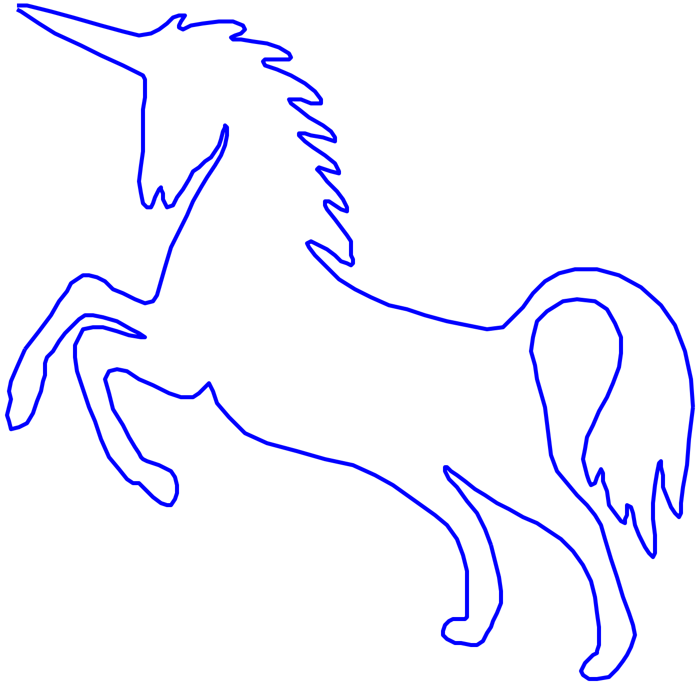
Original curve

Filtering Curves



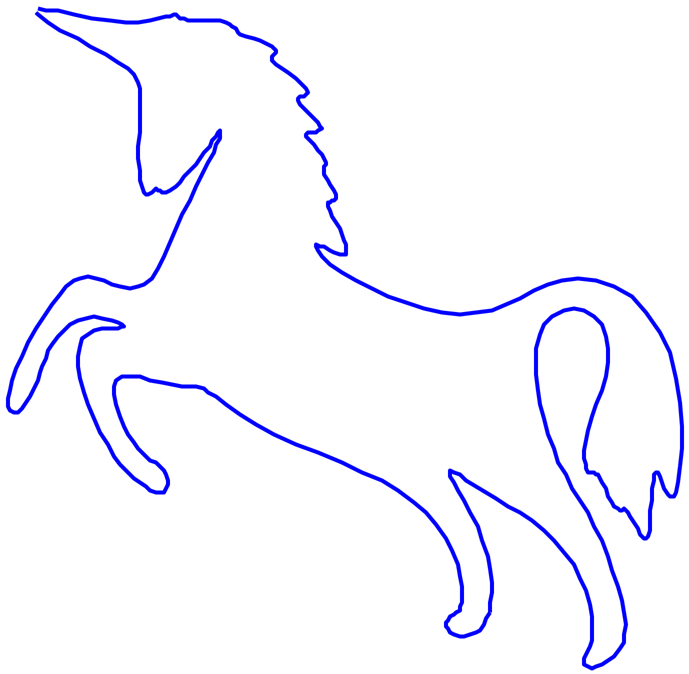
1st iteration; $\lambda=0.5$

Filtering Curves



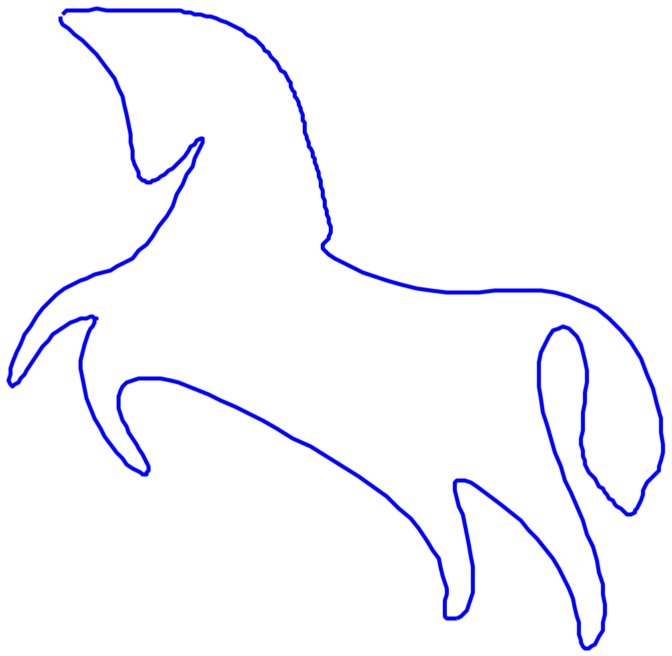
2nd iteration; $\lambda=0.5$

Filtering Curves



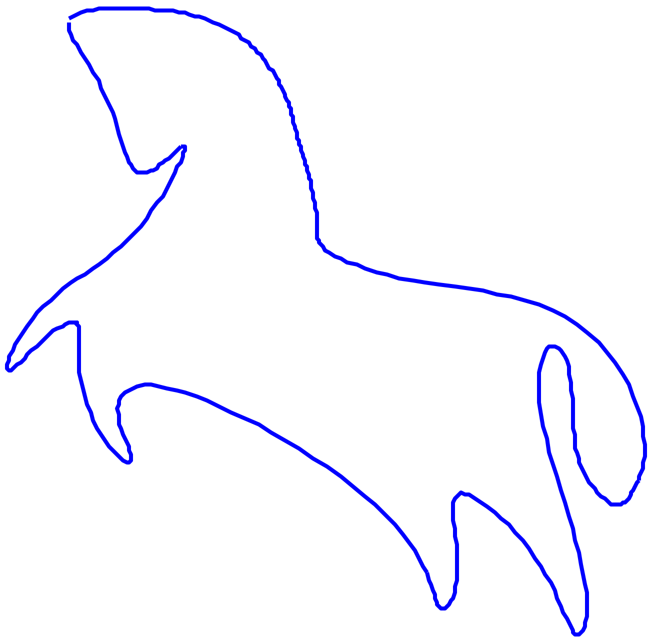
8th iteration; λ=0.5

Filtering Curves



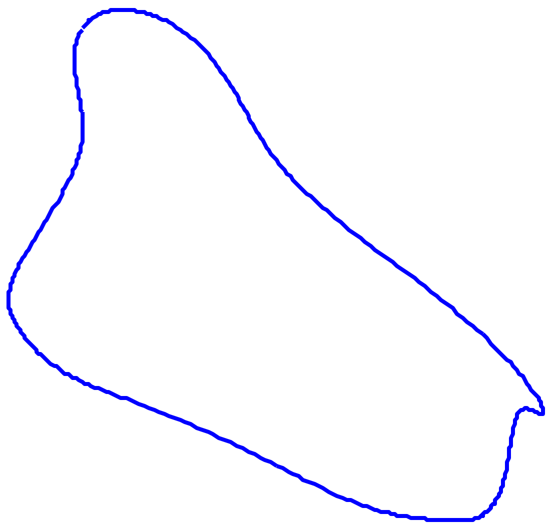
27th iteration; λ=0.5

Filtering Curves



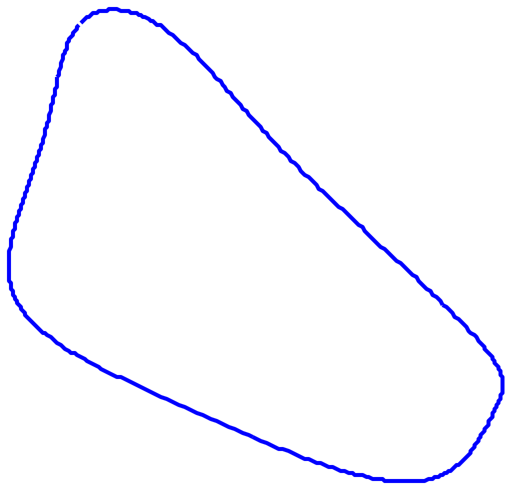
50th iteration; λ=0.5

Filtering Curves



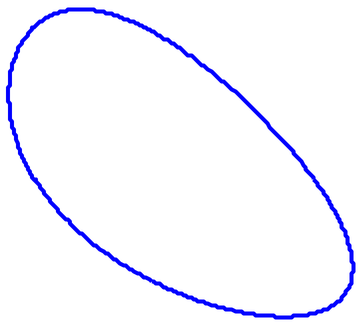
500th iteration; $\lambda=0.5$

Filtering Curves



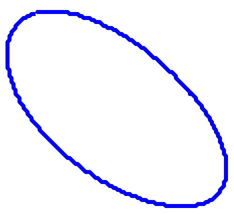
1000th iteration; $\lambda=0.5$

Filtering Curves



5000th iteration; $\lambda=0.5$

Filtering Curves



10000th iteration; λ=0.5

Filtering Curves

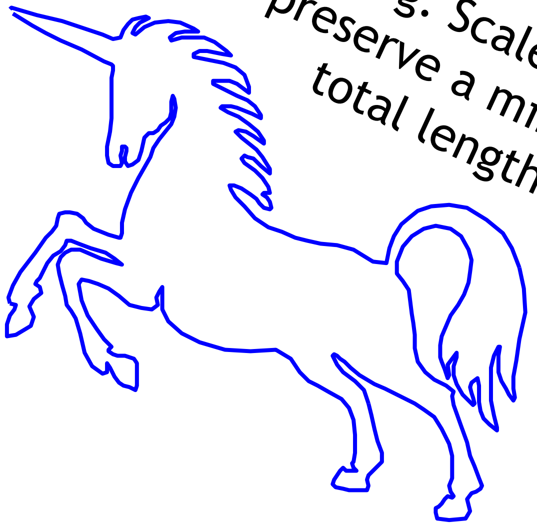
Oh! It shrinks to a point.

(constraints can prevent this)

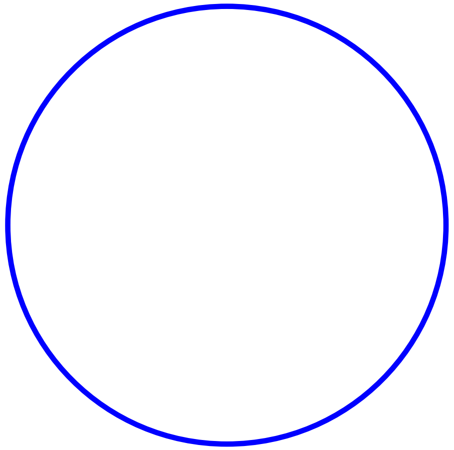
50000th iteration; $\lambda=0.5$

Filtering Curves

e.g. Scale to preserve a miniaml total length.



The circle is the is the smoothest closed curve



Curvature is constant!
& turning number theorem says total curvature can't be zero.

Oh! It shrinks to a point.

(constraints can prevent this)

50000th iteration; $\lambda=0.5$

On meshes: smoothing as mean curvature flow

- Model smoothing as a diffusion process

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

- Discretize in time, forward differences:

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

😄 Explicit integration! Super easy to compute.

😦 Explicit integration! Unstable unless time step dt is small.

Taubin Smoothing: Explicit Steps

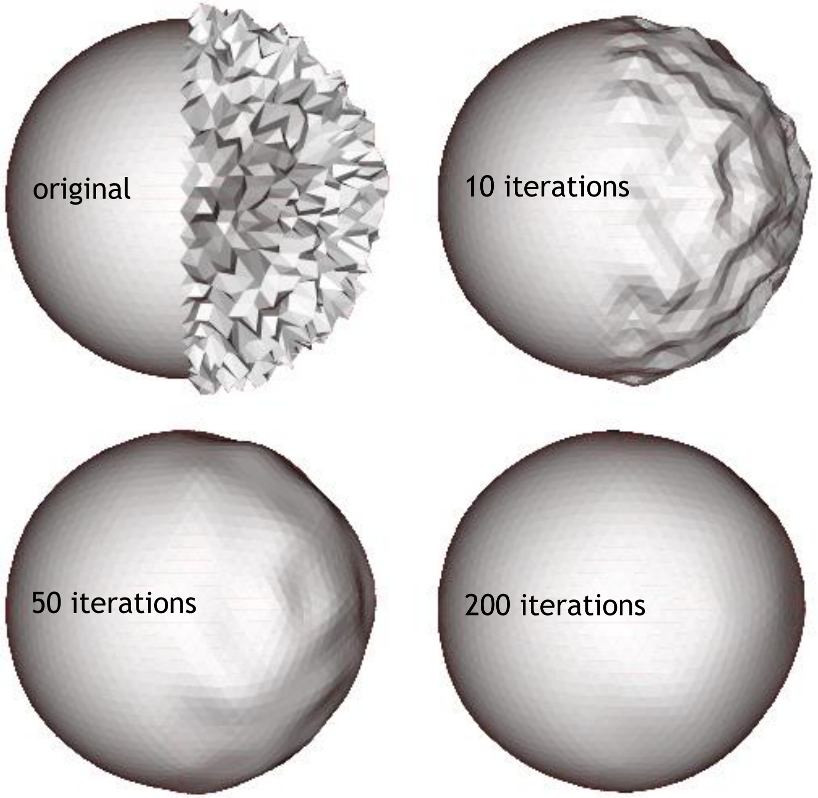
- Simple iteration 👍:

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

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

$\lambda > 0$ to smooth;
$\mu < 0$ to inflate

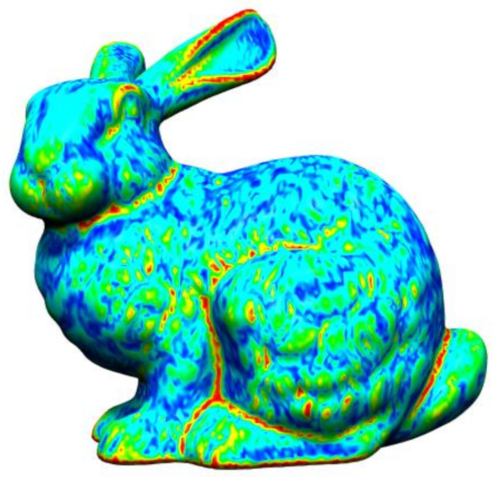
- Disgustingly slow convergence 🤮:
  - Very local
  - λ, μ need tweaking



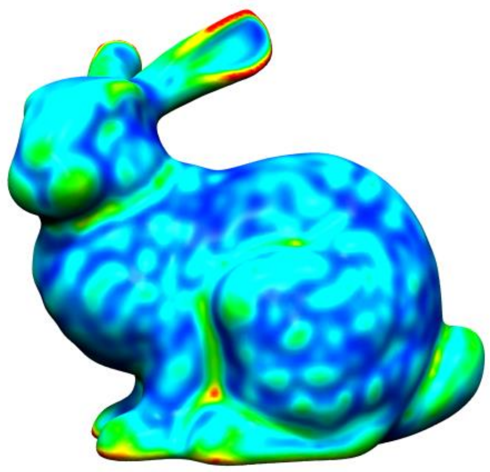
A Signal Processing Approach to Fair Surface Design
Gabriel Taubin
ACM SIGGRAPH 95

Using uniform Laplacian

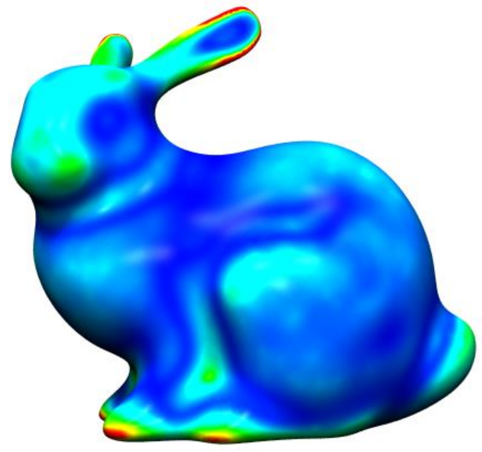
Example



0 iterations



10 iterations



100 iterations

On meshes: smoothing as mean curvature flow

- Model smoothing as a diffusion process

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

Not „local only“ anymore because of the global solve. Faster convergence! 🌏

- Discretize in time, forward differences:

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

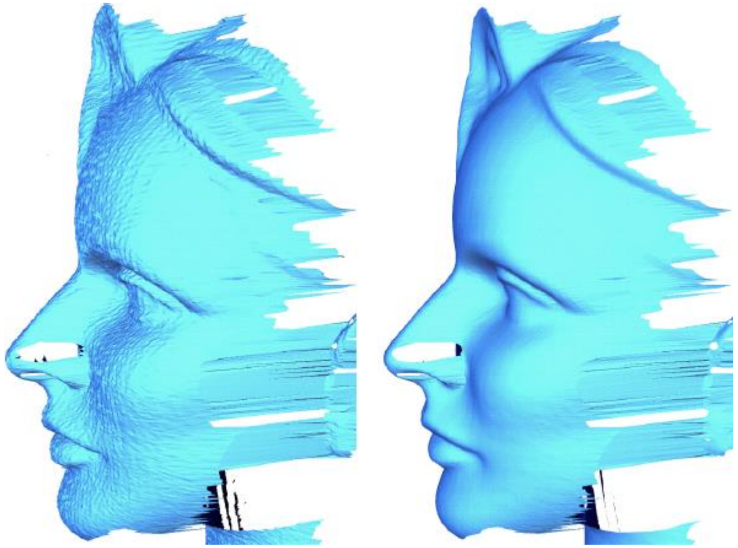
😦 Implicit integration! Harder to compute because of matrix solve.

😄 *Implicit* integration! Much more stable for small steps dt compared to explicit integration

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

Implicit Fairing

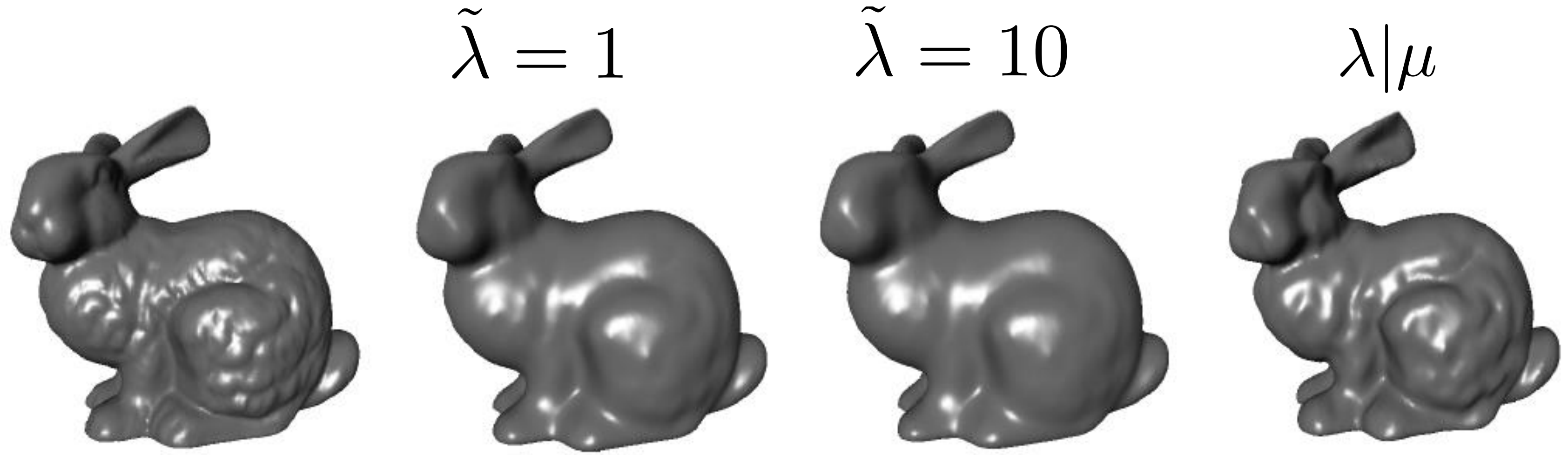


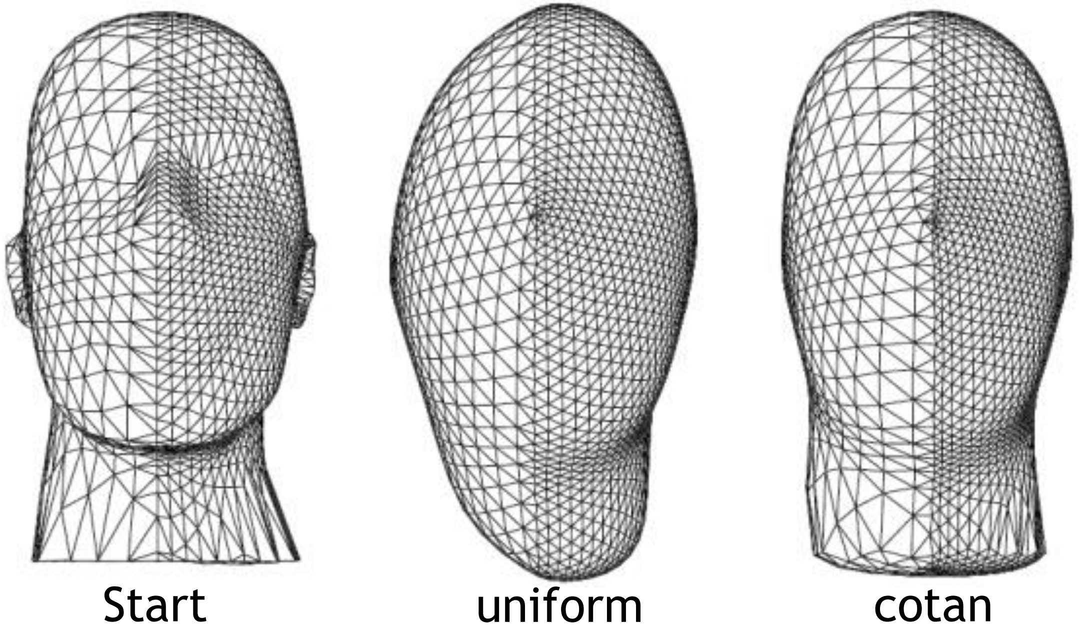
Figure 4: *Stanford bunnies: (a) The original mesh, (b) 10 explicit integrations with $\lambda dt = 1$, (c) 1 implicit integration with $\lambda dt = 10$ that takes only 7 PBCG iterations (30% faster), and (d) 20 passes of the $\lambda|\mu$ algorithm, with $\lambda = 0.6307$ and $\mu = -0.6732$. The implicit integration results in better smoothing than the explicit one for the same, or often less, computing time. If volume preservation is called for, our technique then requires many fewer iterations to smooth the mesh than the $\lambda|\mu$ algorithm.*

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? Because triangle size is not considered. Smaller triangles => smaller values of laplacian

Asymmetric results although underlying geometry is symmetric

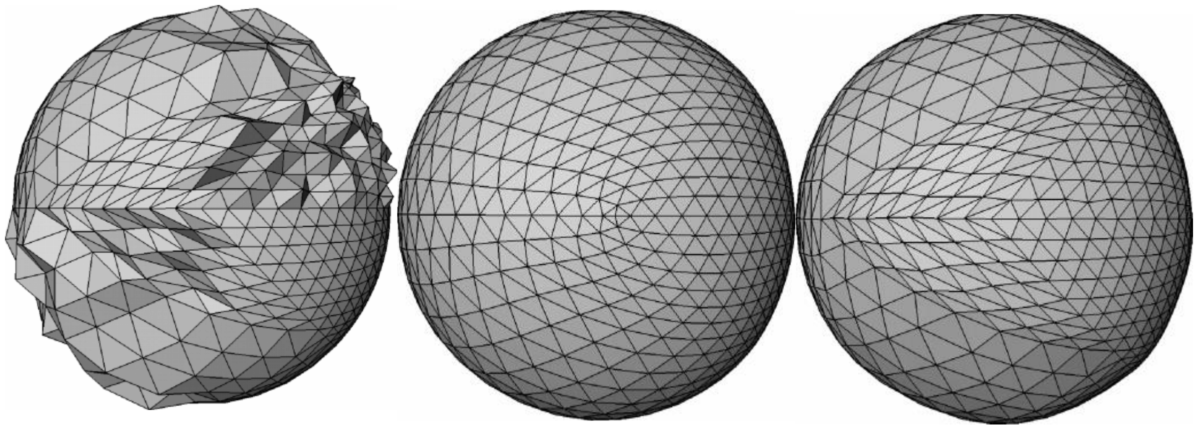


😫 😍

Mesh Independence

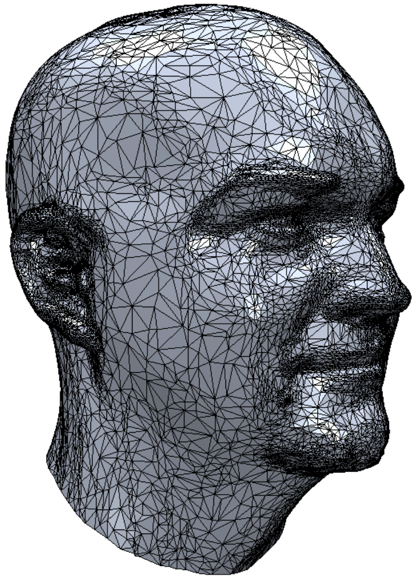
- Result of smoothing with uniform Laplacian depends on triangle density and shape
  - Why? Because triangle size is not considered. Smaller triangles => smaller values of laplacian

Asymmetric results although underlying geometry is symmetric

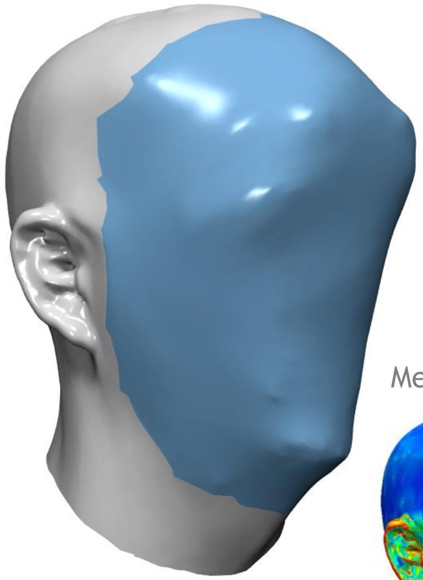


Comparison of the weights

- Implicit fairing with different weights:



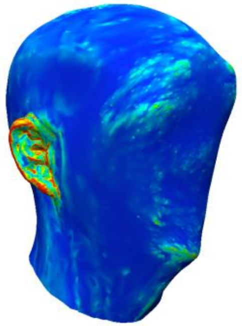
Start

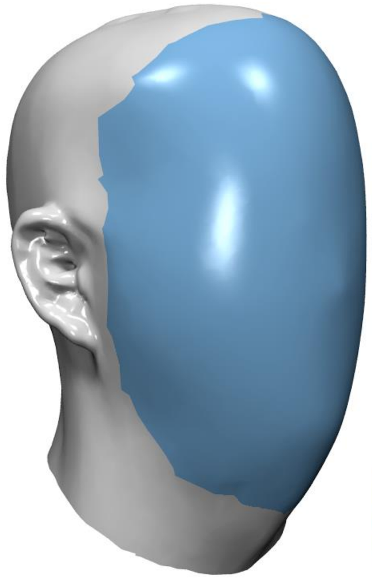


uniform

😫

Mean curvature

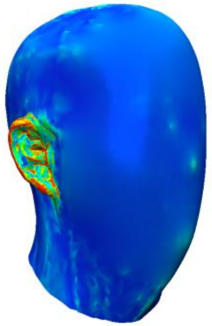




cotan

🥰

Mean curvature



Thank you