

# CS233, CME251: Geometric and Topological Data Analysis

Leonidas Guibas  
Computer Science Department  
Stanford University



Lecture 11  
11 May 2020



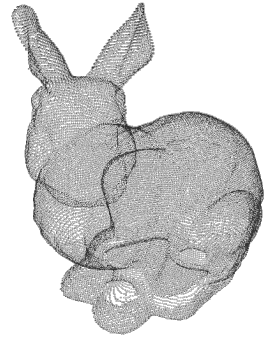
# Class Midterm

- The class midterm will be on **Monday, May 18**, during the regular class time. It may cover material relevant to any previous lecture.
- Unlike the homeworks, **the exam will consist of a number of short problems** (that also have short answers) testing basic understanding of the concepts covered in class. All problems will have the same weight and there will be some choice (do X out of Y).
- In doing the exam, you may use any of the on-line class materials, as well as your own notes. **Internet searches, however, are not allowed -- nor is any kind of human help or collaboration on the solutions.**

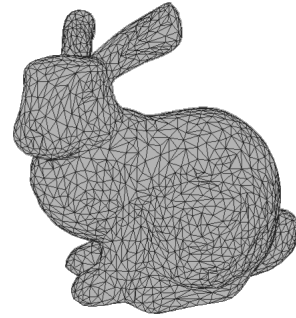
# Last Time: Geometry Representations in 3D



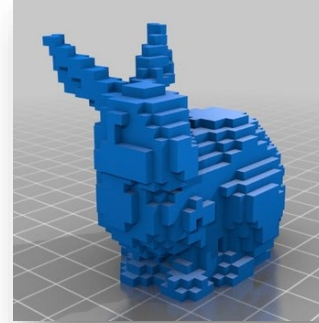
# In 3D There is Representation Diversity



Point Cloud



Mesh



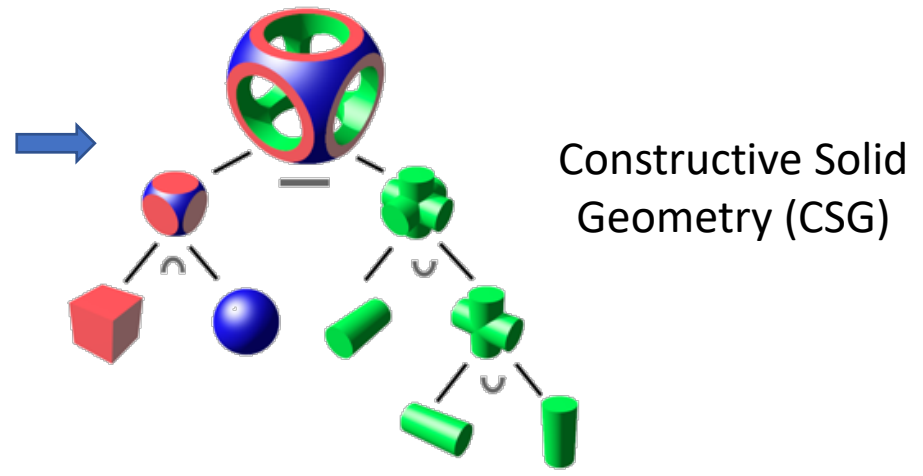
Voxels



Multiple Projected Views  
RGB(D)

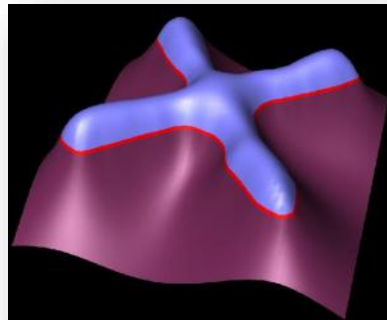
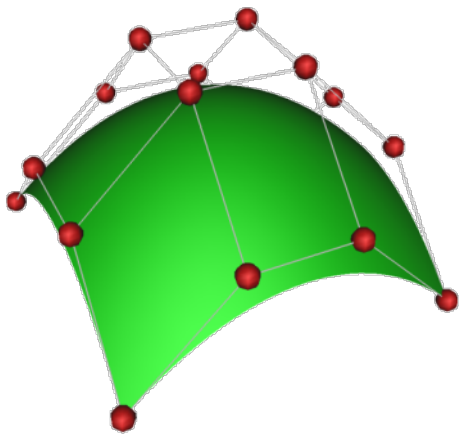
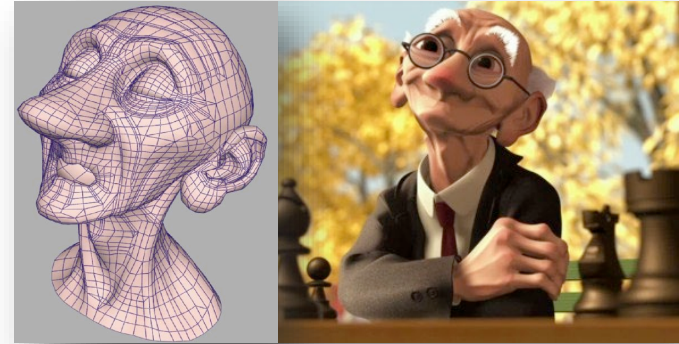
...

These are irregular representations

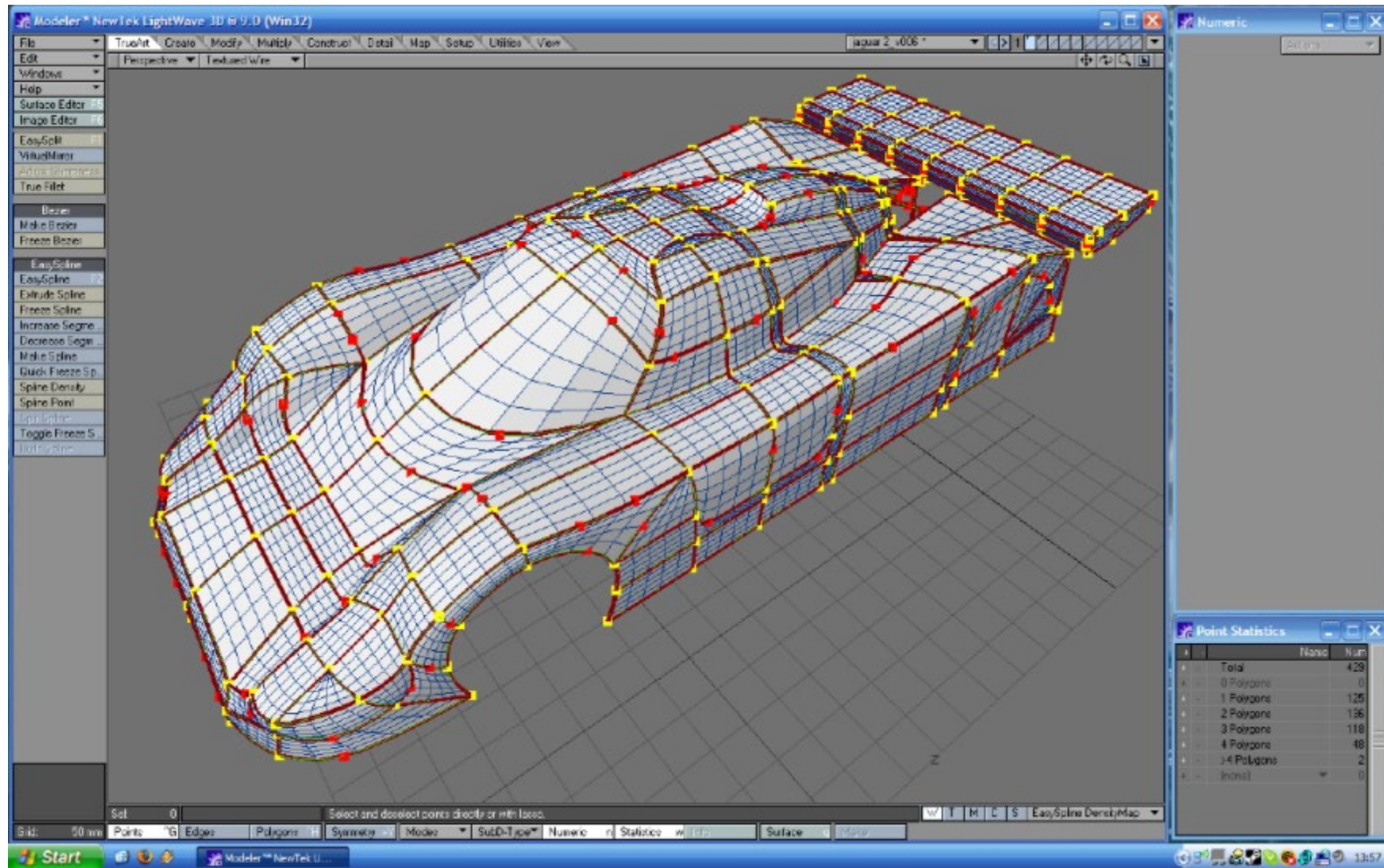


# Boundary-Reps: High-Level

- Parametric surfaces / Splines
- Implicit functions
- Subdivision surfaces



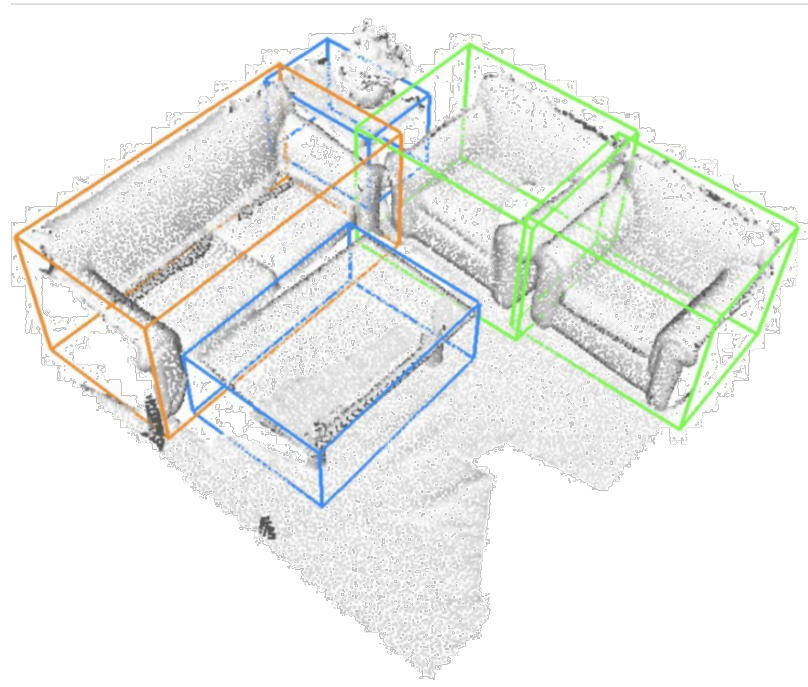
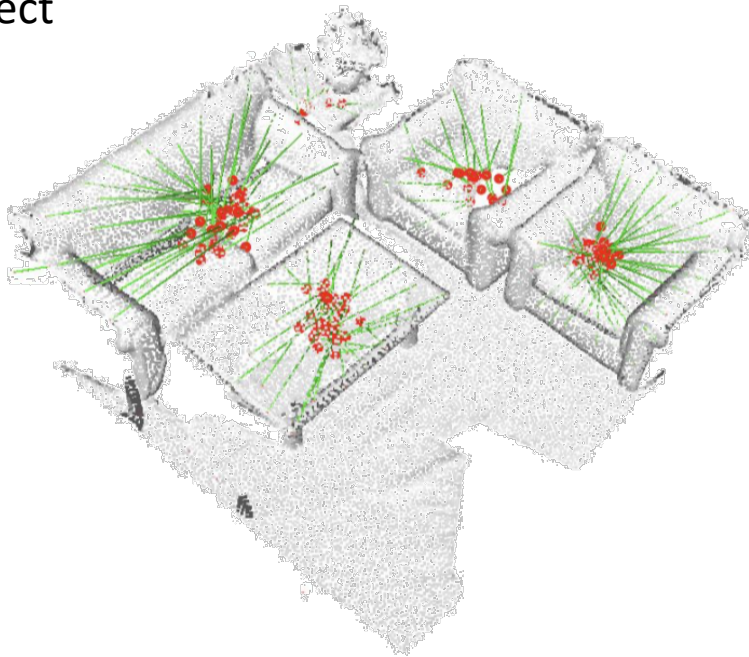
# CAGD: Computer-Aided Geometric Design



# Point Cloud Data



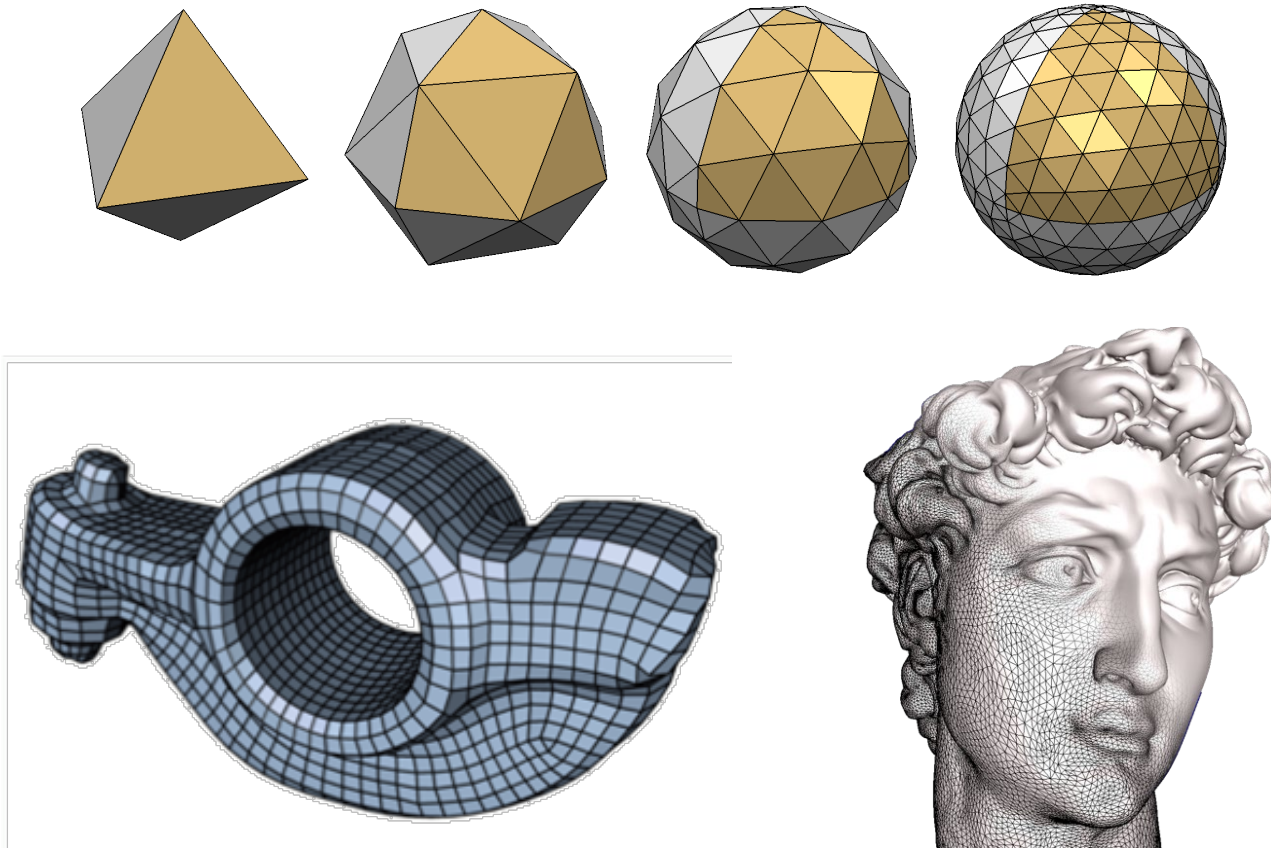
Microsoft Kinect



Intel RealSense

# Polygonal Meshes

- Boundary representations of objects using polygonal primitives



# Representation Conversions

Points → Implicit  
Implicit → Mesh  
**Mesh → Points**

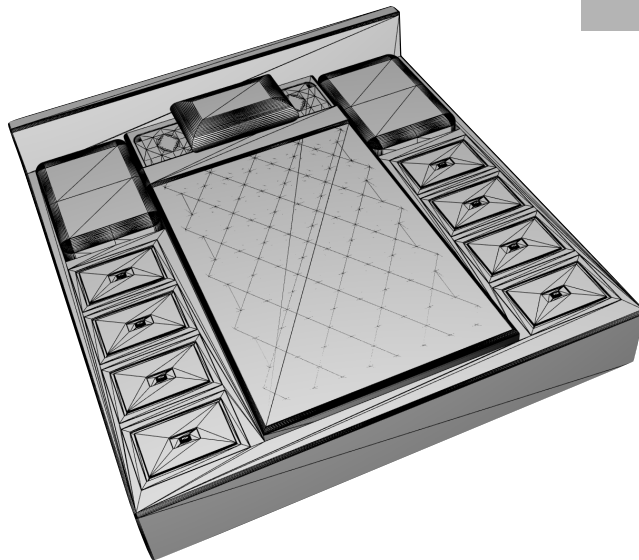
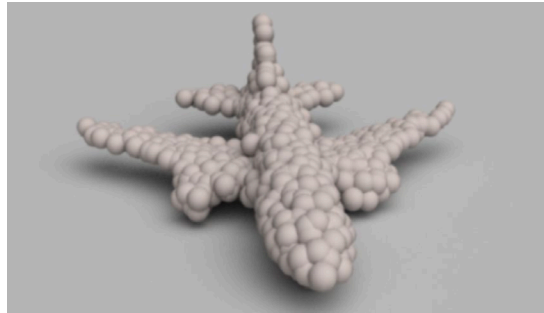


# MESH-> POINT CLOUD

Sampling

# From Surface to Point Cloud Why?

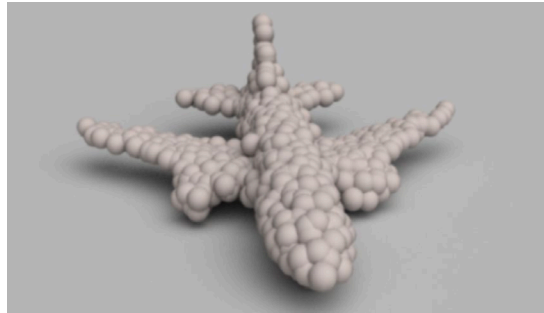
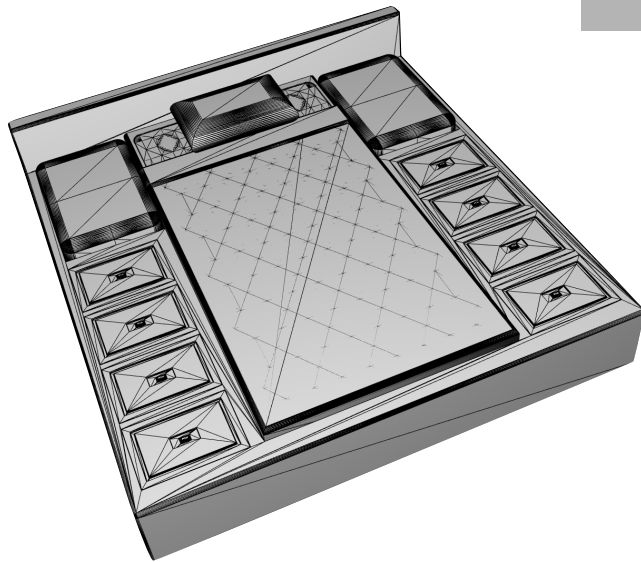
- Points are simple but expressive
- Flexible, unstructured, few constraints
- Also: ML applications!



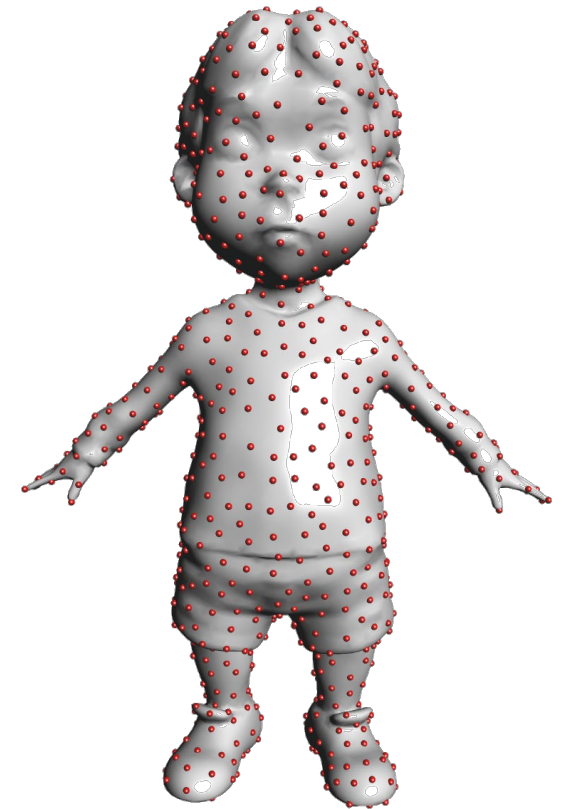
CAD meshes:  
many components  
bad triangles  
connectivity problems

# From Surface to Point Cloud Why?

- Points are simple but expressive
- Flexible, unstructured, few constraints
- Also: ML applications!



CAD meshes:  
many components  
bad triangles  
connectivity problems



the problem:  
sampling the mesh

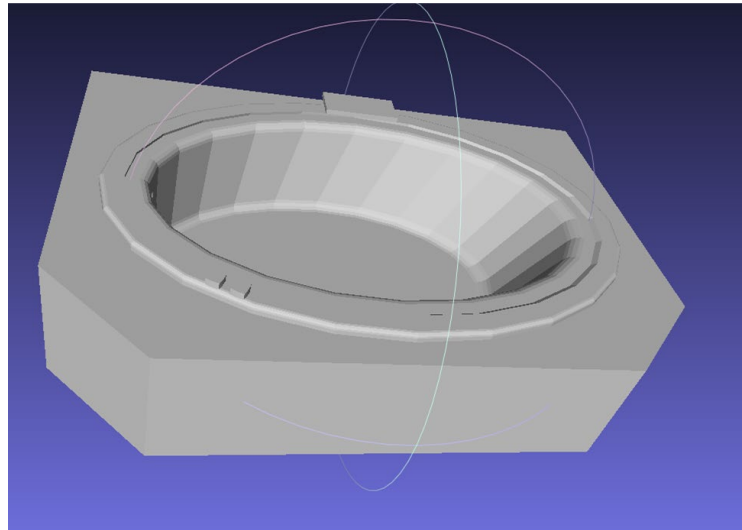
# Furthest Point Sampling (FPS)

1. Create an initial sample point set  $S$ 
  - Keypoints + additional random points.
2. Find the point which is the farthest from all point in  $S$

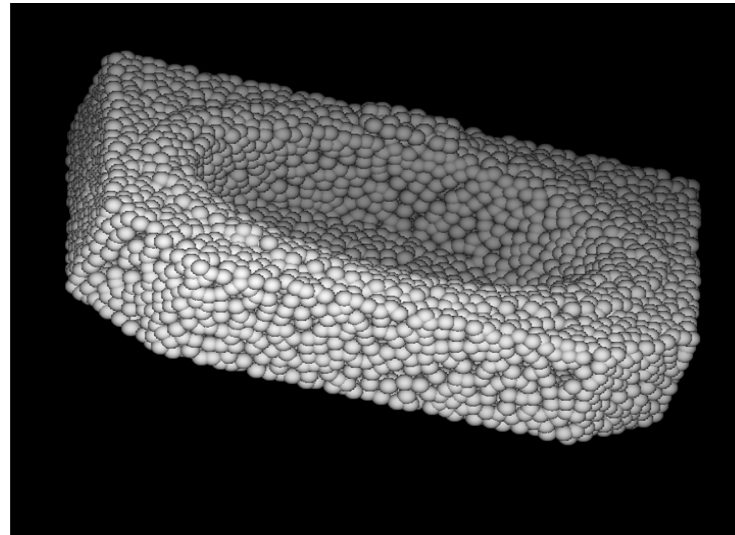
$$\begin{aligned}d(p, S) &= \max_{q \in A} (d(q, S)) \\ &= \max_{q \in A} \left( \min_{0 \leq i < N} (d(q, s_i)) \right)\end{aligned}$$

3. Insert the point to  $S$  and update the distances
4. While more points are needed, iterate

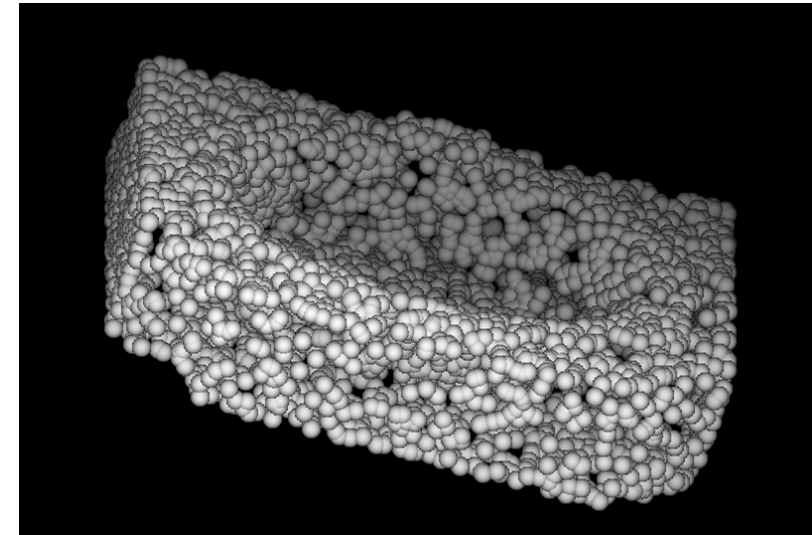
# FPS vs. Random



Mesh



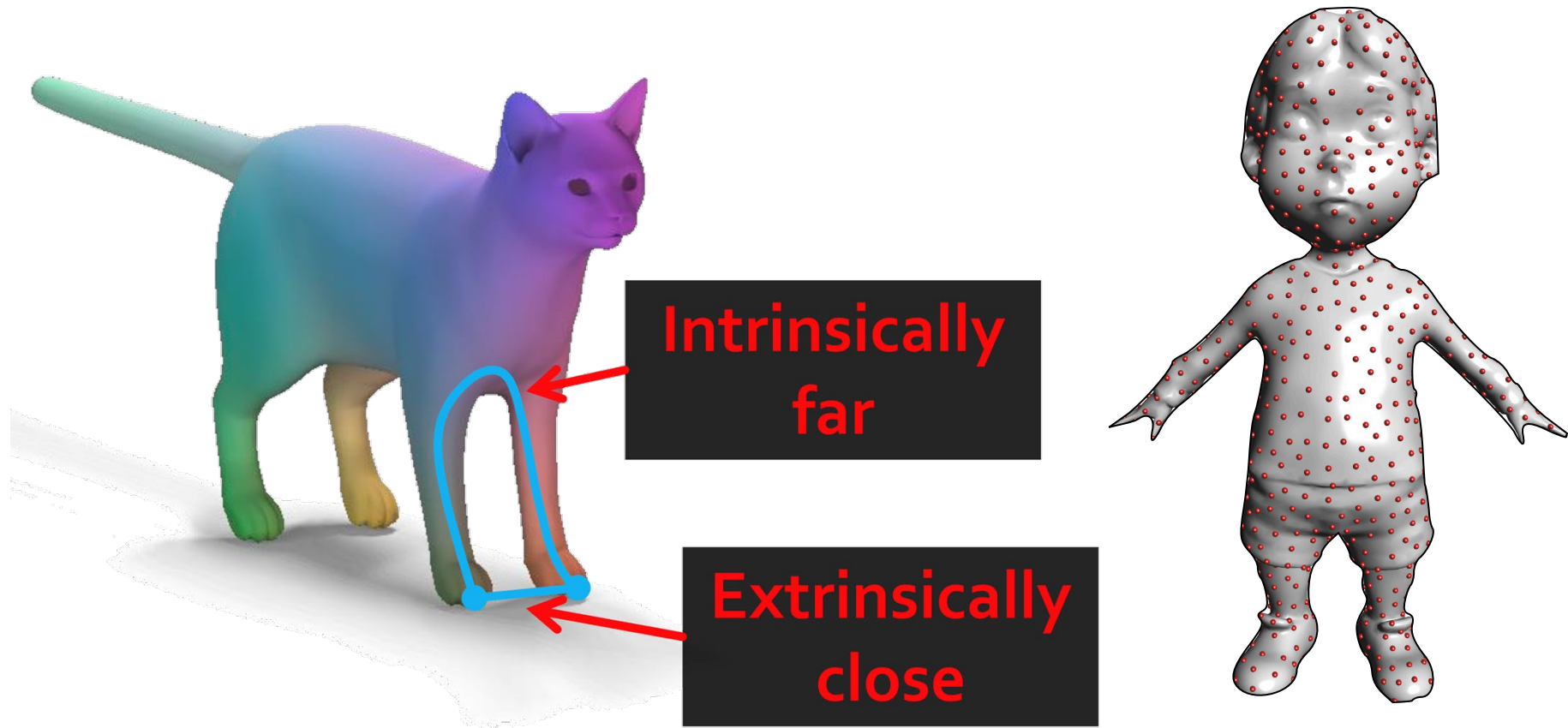
FPS



Random

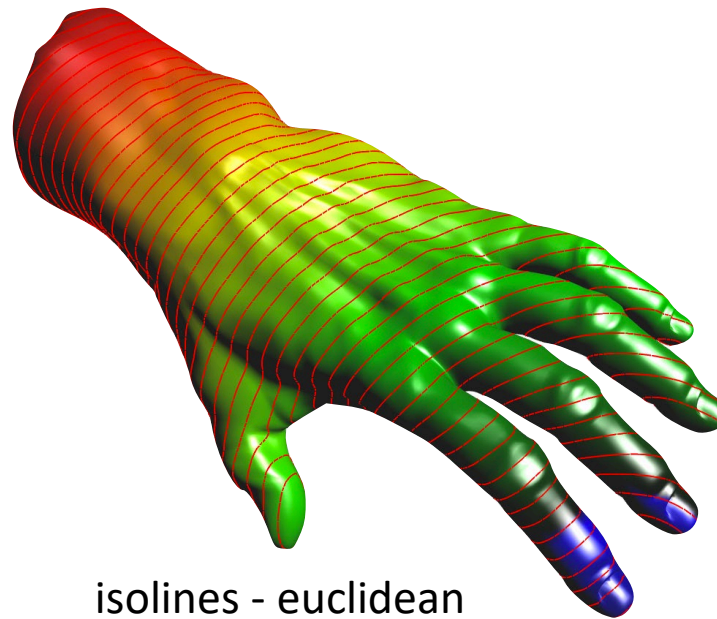
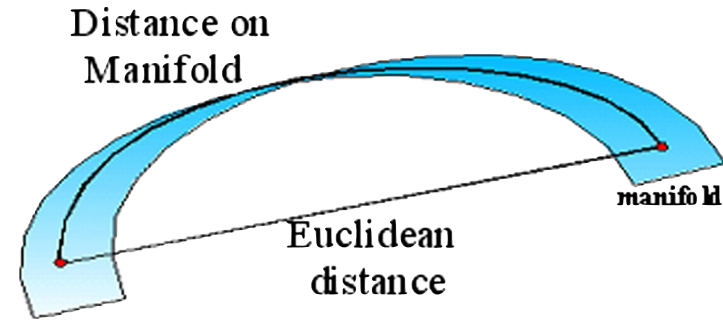
# FPS on surfaces

• What's an appropriate distance?

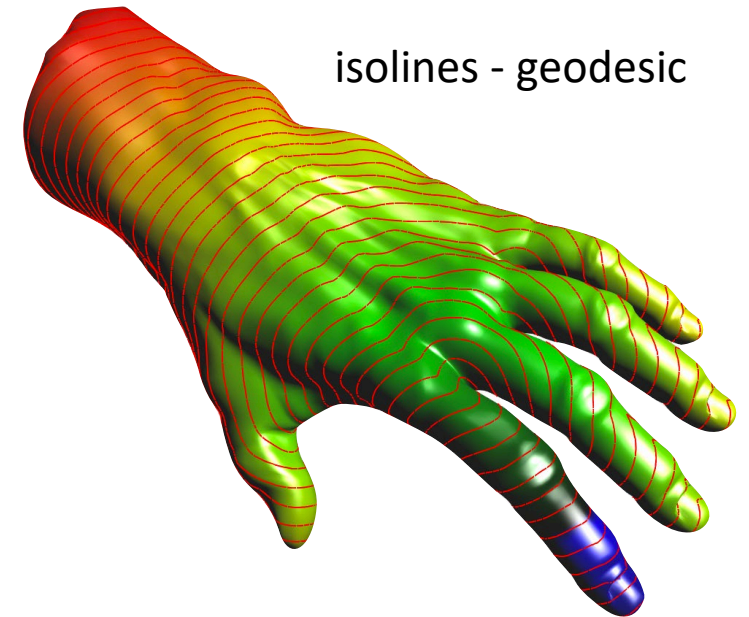


# On-Surface Distances

• Geodesics: Straightest and **locally shortest** curves



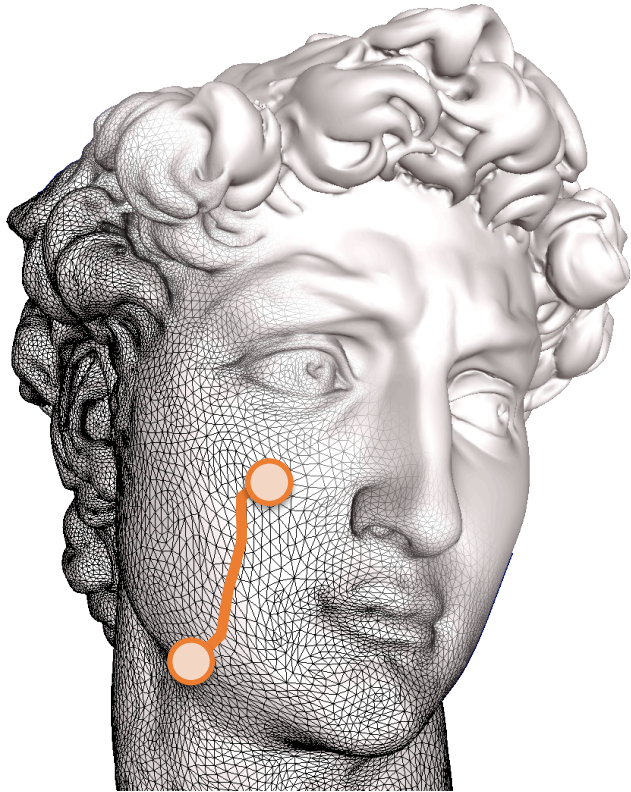
isolines - euclidean



isolines - geodesic

# Discrete Geodesics

- Recall: a mesh is a graph!
- Approximate geodesics as paths along edges



```
 $v_0$  = initial vertex  
 $d_i$  = current distance to vertex  $i$   
 $S$  = vertices with known optimal distance
```

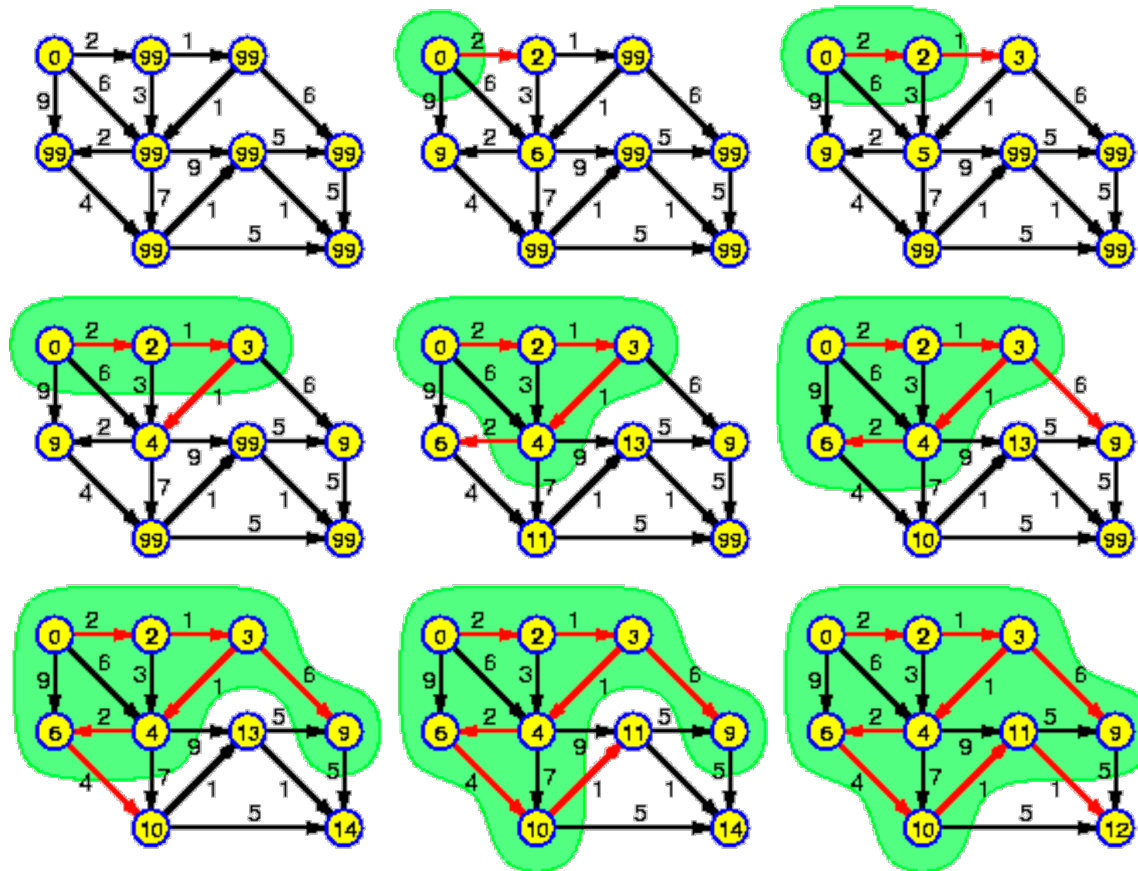
```
# initialize  
 $d_0 = 0$   
 $d_i = [\text{inf for } d \text{ in } d_i]$   
 $S = \{\}$ 
```

**Dijkstra's  
algorithm!**

```
for each iteration  $k$ :  
  # update  
   $k = \text{argmin}(d_k)$ , for  $v_k$  not in  $S$   
   $S.append(v_k)$   
  for neighbors index  $v_l$  of  $v_k$ :  
     $d_l = \min([d_l, d_k + d_{kl}])$ 
```

# Dijkstra Geodesics

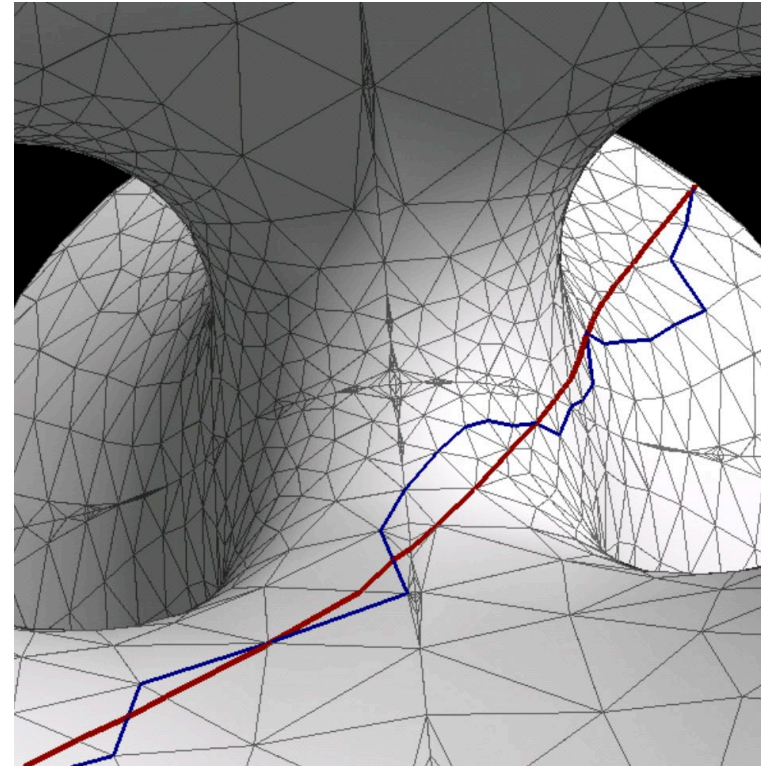
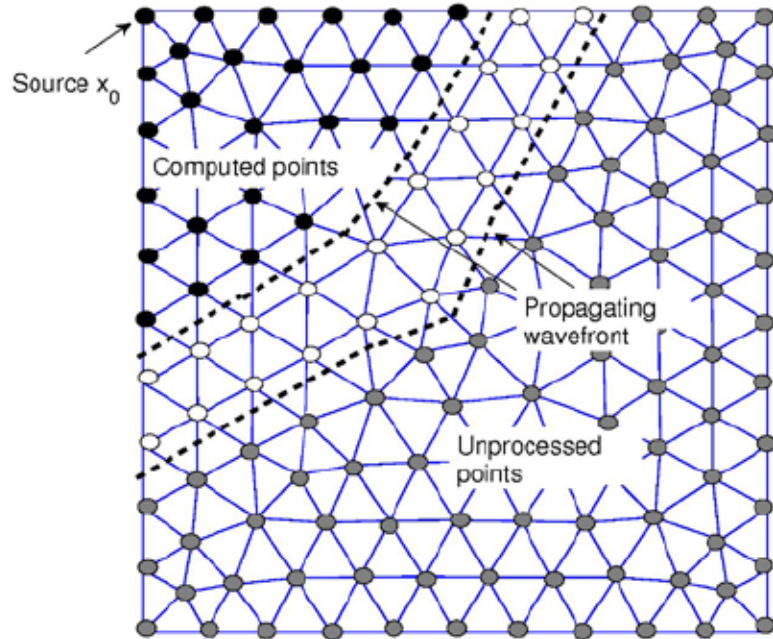
## ● Dijkstra as wave front propagation





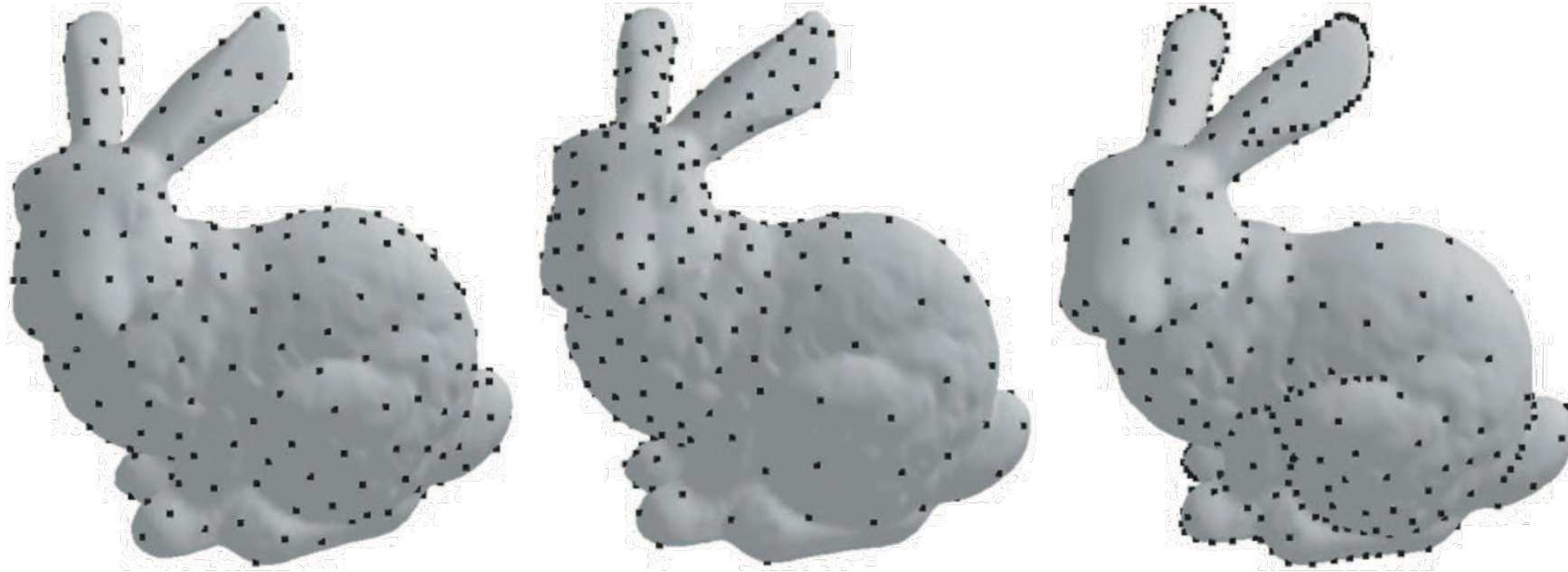
# Fast Marching Geodesics

- A better approximation: allow fronts to cross triangles!



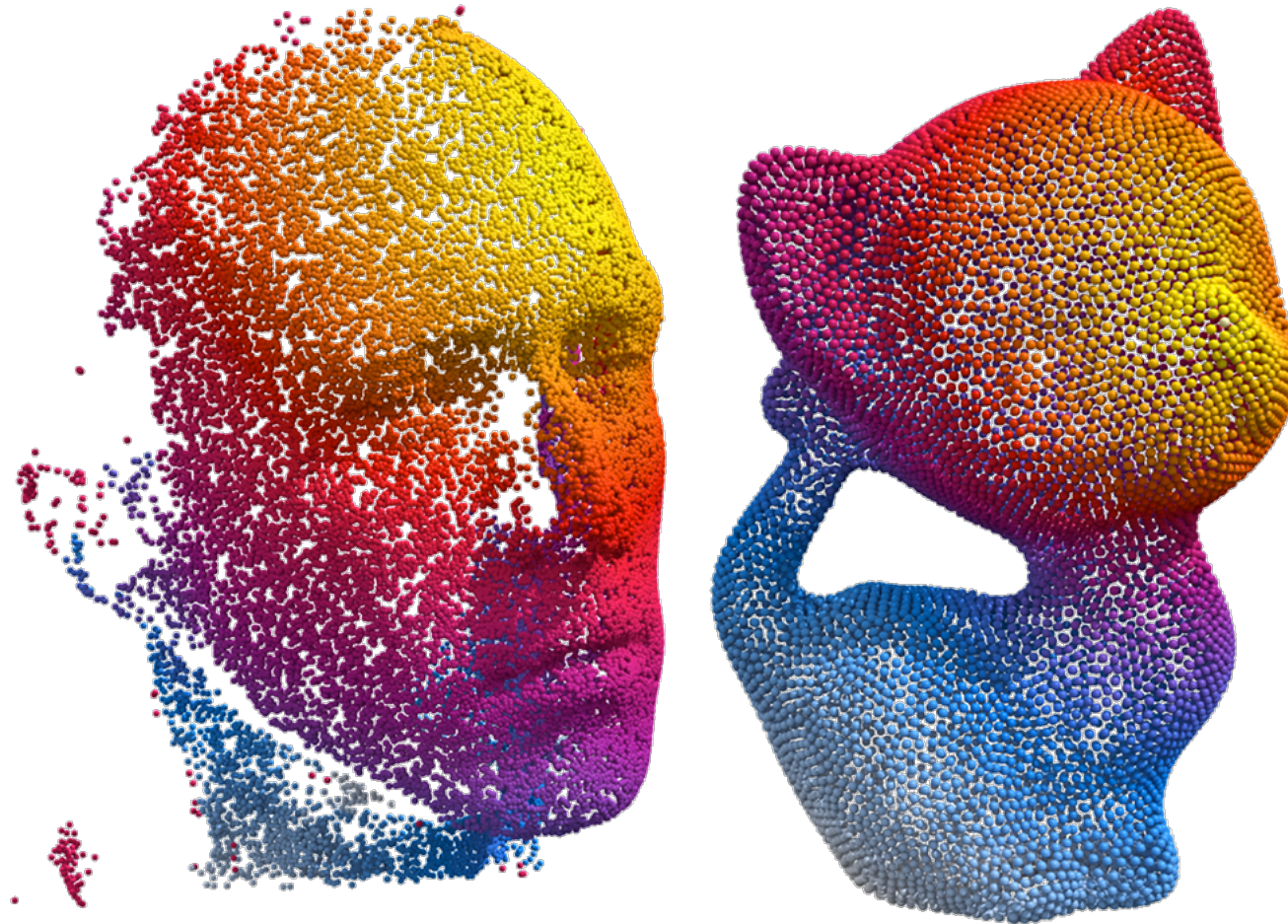
Kimmel and Sethian 1997, "Computing Geodesic Paths on Manifolds"

# FPS on a Mesh



Peyré and Cohen 2003, Geodesic Remeshing Using Front Propagation

# Faster Distance Approximations



Carne, Weischedel, and Wardetzky 2017,  
The Heat Method for Distance Computation

Today: Differential Geometry of  
Surfaces and the Laplace-  
Beltrami Operator

# A Primer on Differential Geometry

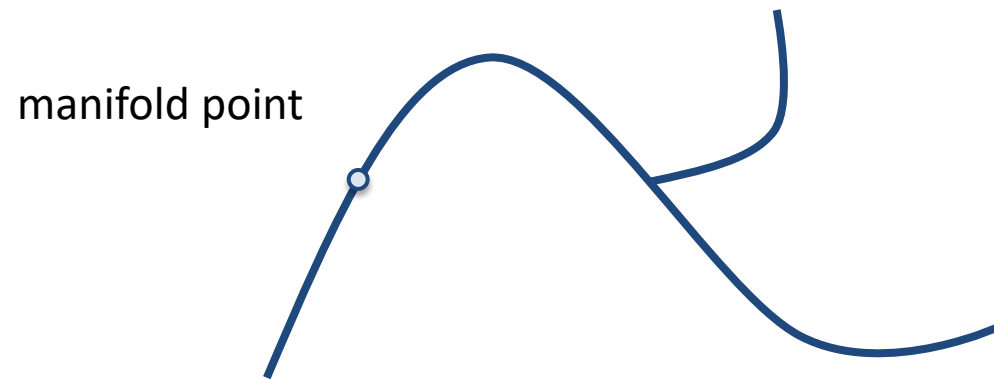
# Differential Geometry Basics

- Geometry of manifolds
- Properties that can be discovered by local observation: point + neighborhood



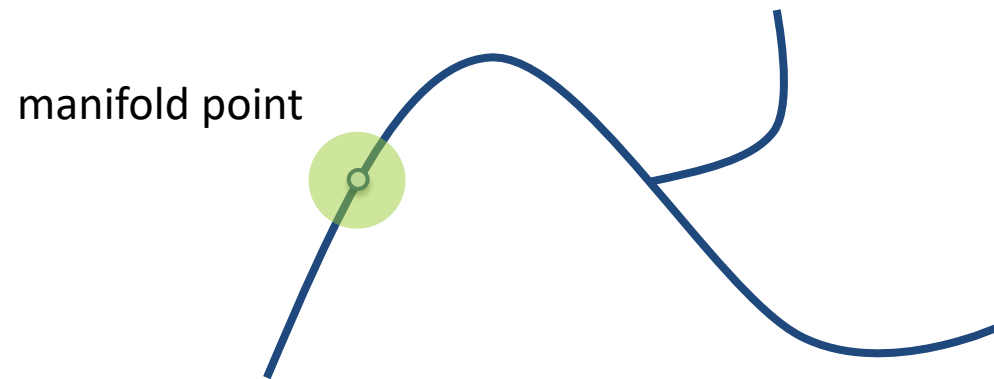
# Differential Geometry Basics 2D

- Geometry of manifolds
- Properties that can be discovered by local observation: point + neighborhood



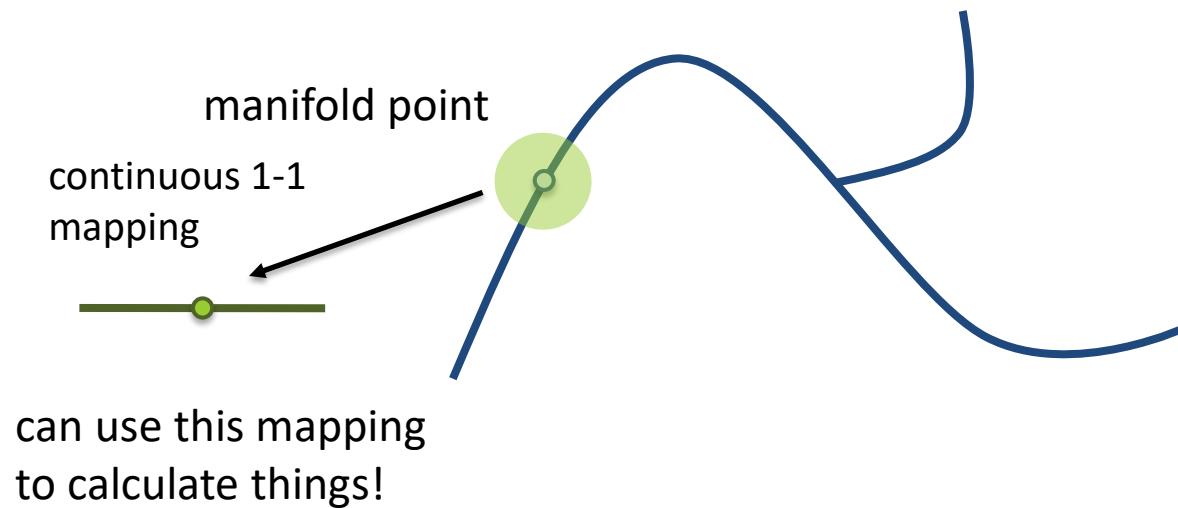
# Differential Geometry Basics 2D

- Geometry of manifolds
- Properties that can be discovered by local observation: point + neighborhood



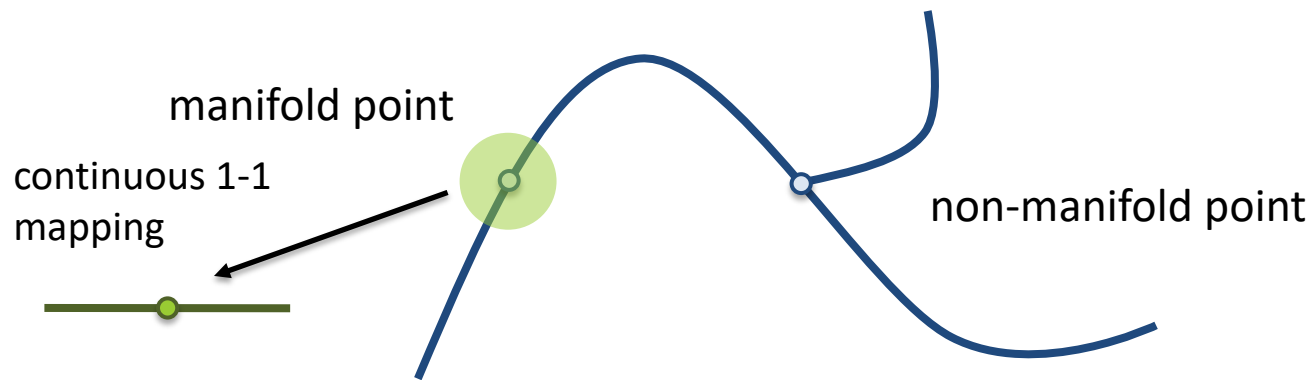
# Differential Geometry Basics 2D

- Geometry of manifolds
- Properties that can be discovered by local observation: point + neighborhood



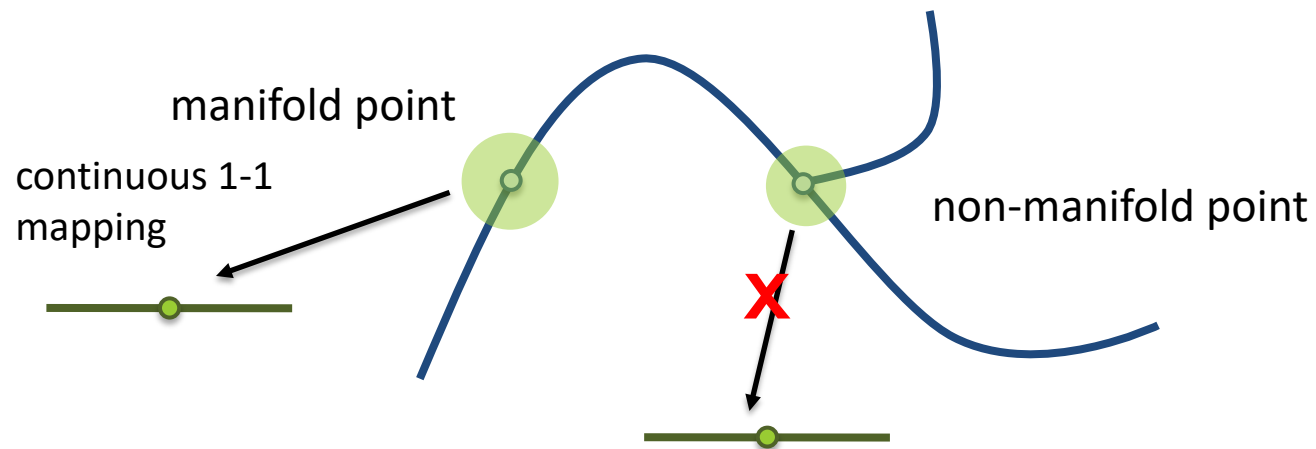
# Differential Geometry Basics 2D

- Geometry of manifolds
- Properties that can be discovered by local observation: point + neighborhood



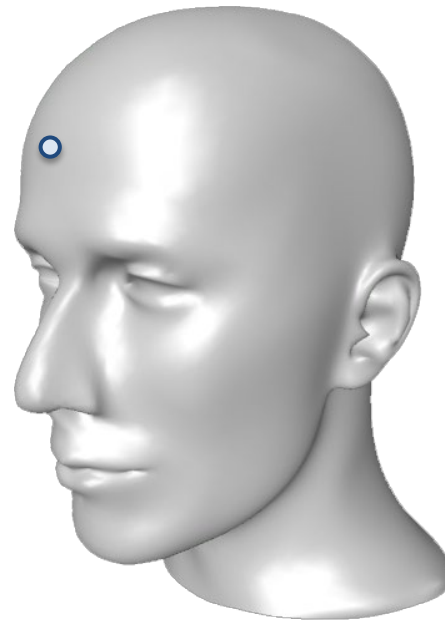
# Differential Geometry Basics

- Geometry of manifolds
- Properties that can be discovered by local observation: point + neighborhood



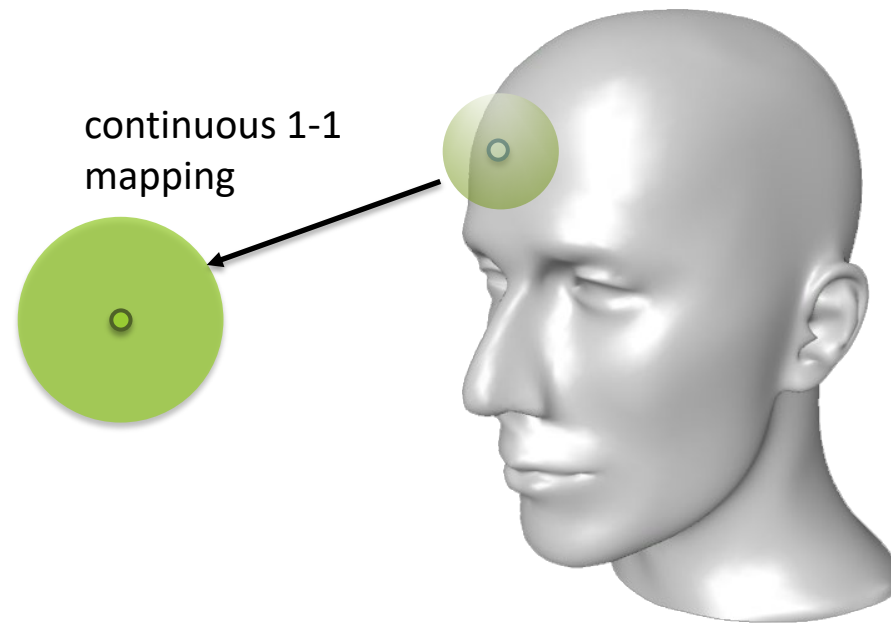
# Differential Geometry Basics 3D

- Geometry of manifolds
- Properties that can be discovered by local observation: point + neighborhood



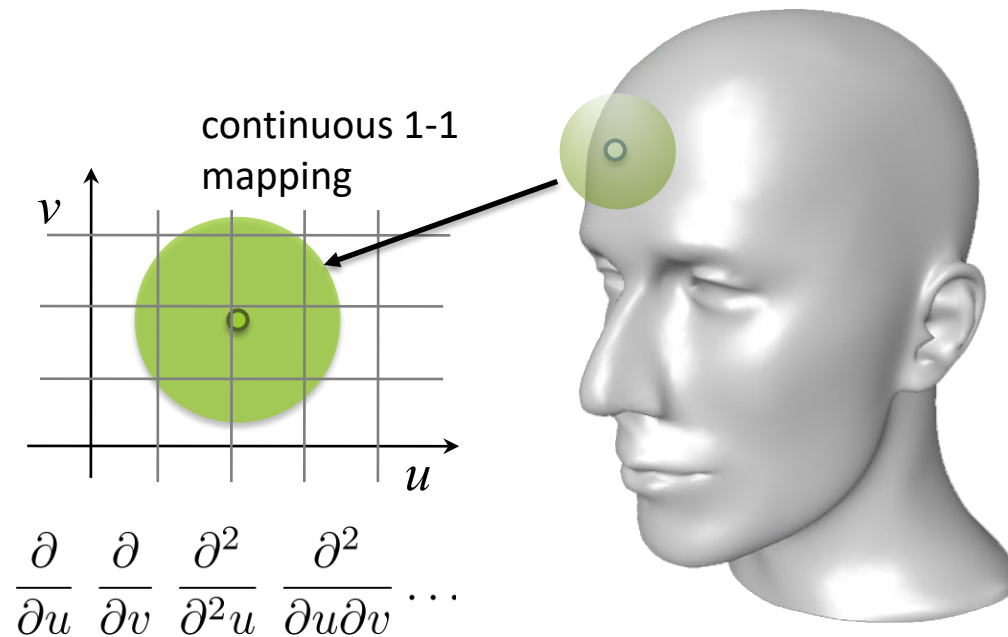
# Differential Geometry Basics 3D

- Geometry of manifolds
- Properties that can be discovered by local observation: point + neighborhood



# Differential Geometry Basics 3D

- Geometry of manifolds
- Properties that can be discovered by local observation: point + neighborhood

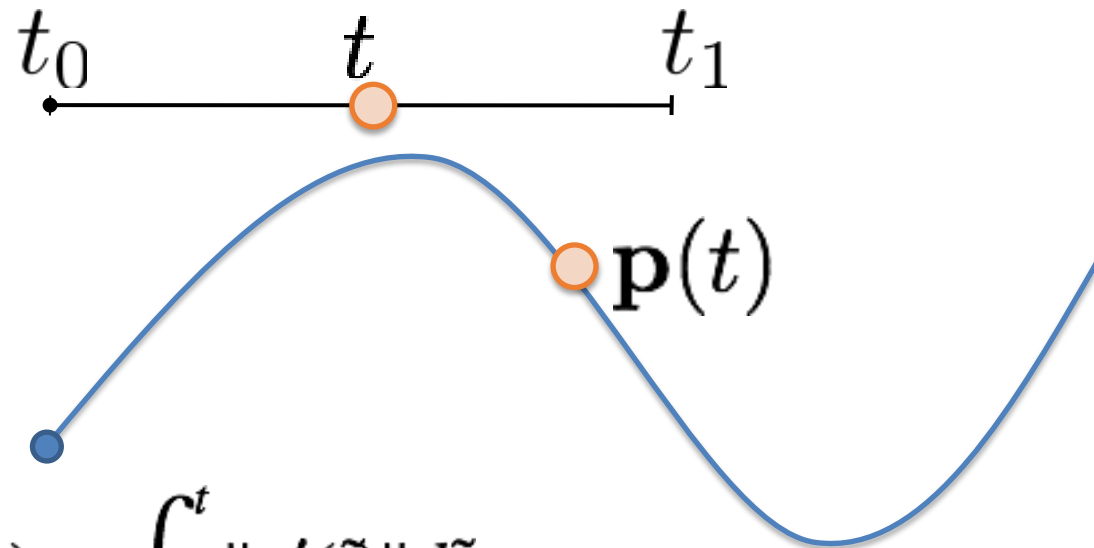


If a sufficiently smooth mapping can be constructed, we can look at its first and second derivatives

**Tangents, normals,  
curvatures, curve angles,  
distances**

# Parametric Curves

- 2D:  $\mathbf{p}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}, t \in [t_0, t_1]$
- $\mathbf{p}(t)$  must be continuous

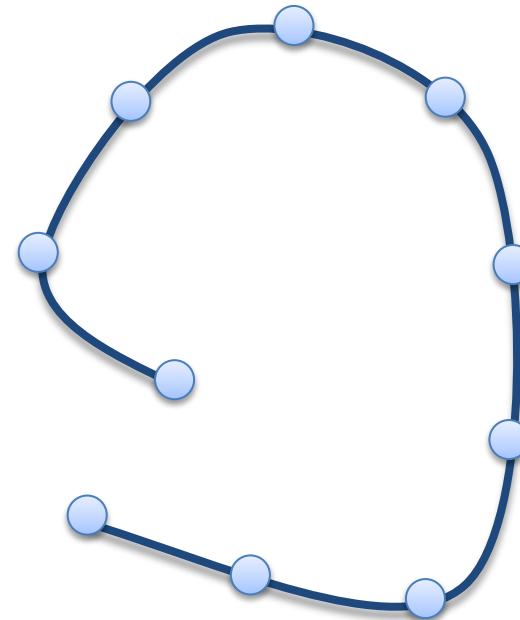


$$\text{len}(\mathbf{p}(t_0), \mathbf{p}(t)) = \int_{t_0}^t \|\mathbf{p}'(\tilde{t})\| d\tilde{t}$$

# Arc Length Parameterization

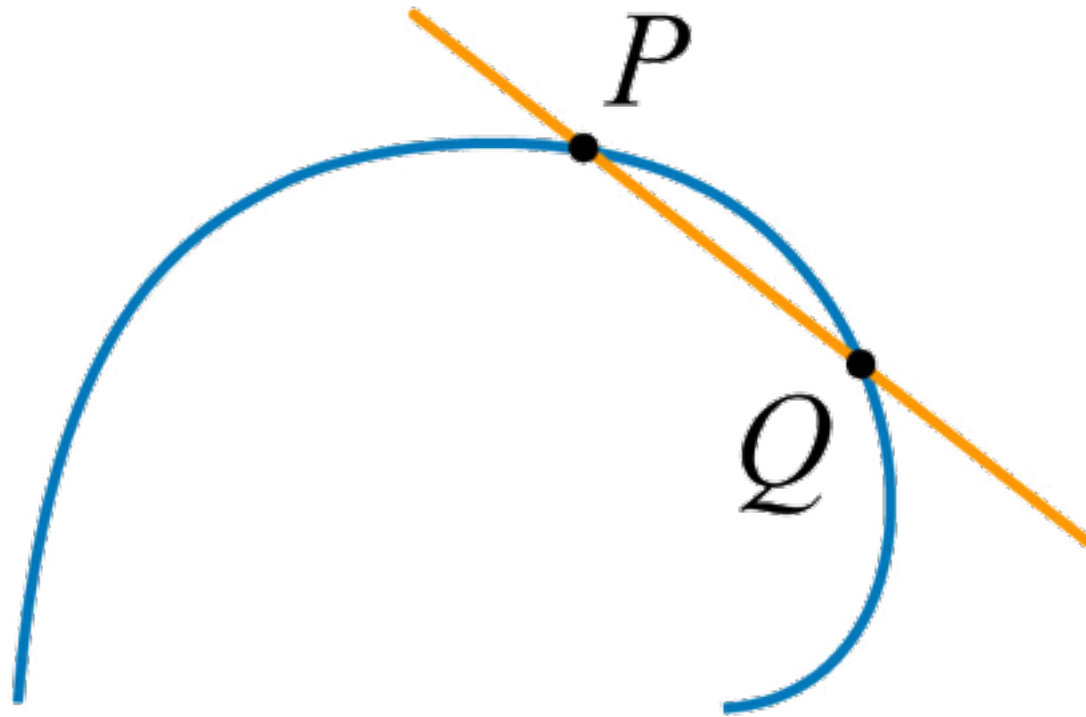
- Equal pace of the parameter along the curve
- $len(\mathbf{p}(s_1), \mathbf{p}(s_2)) = |s_1 - s_2|$
- Now parameter goes from 0 to L

$$\|\mathbf{p}'(s)\| = 1$$



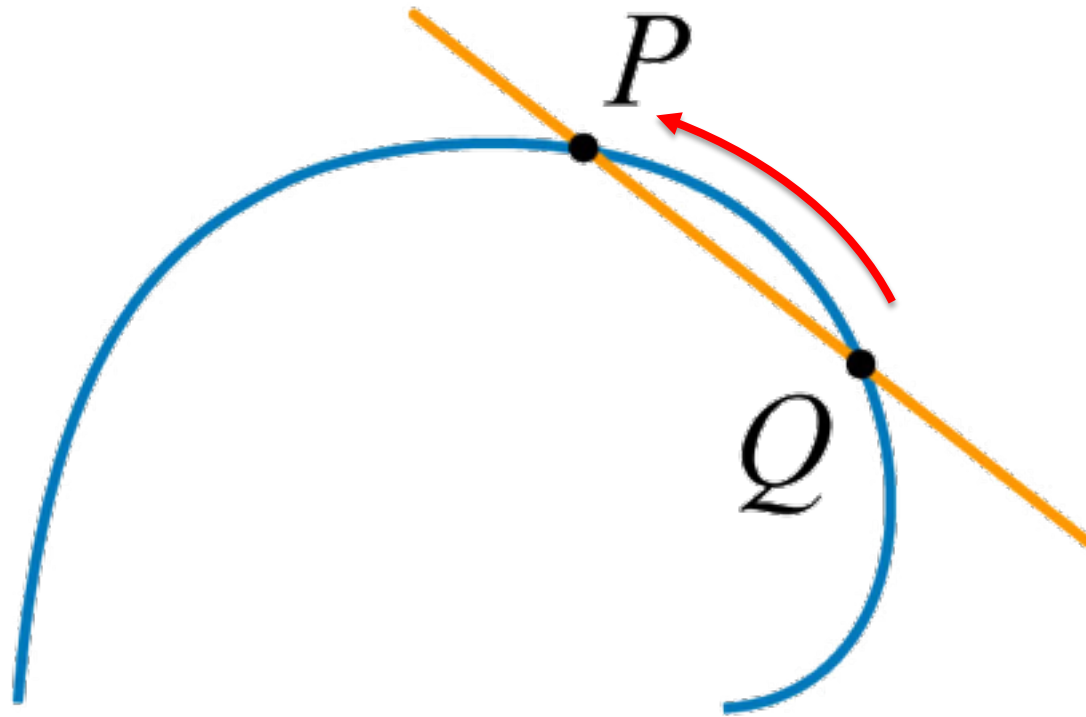
# Secant

- A line through two points on the curve.



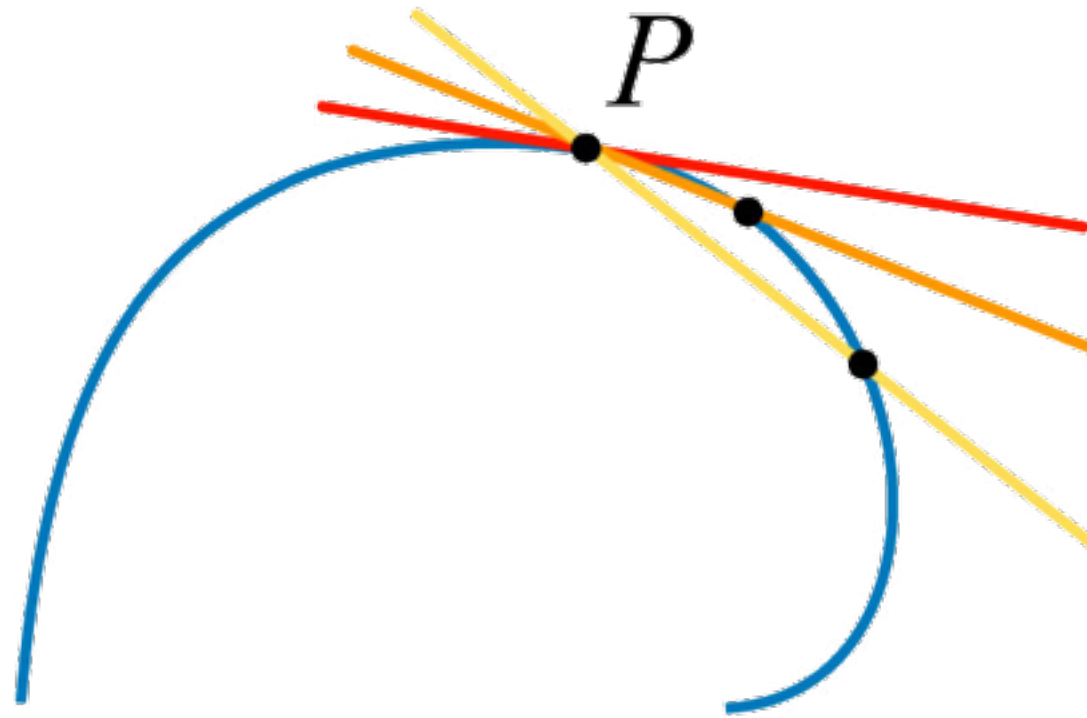
# Secant

- A line through two points on the curve.



# Tangent

- The limiting secant as the two points come together.



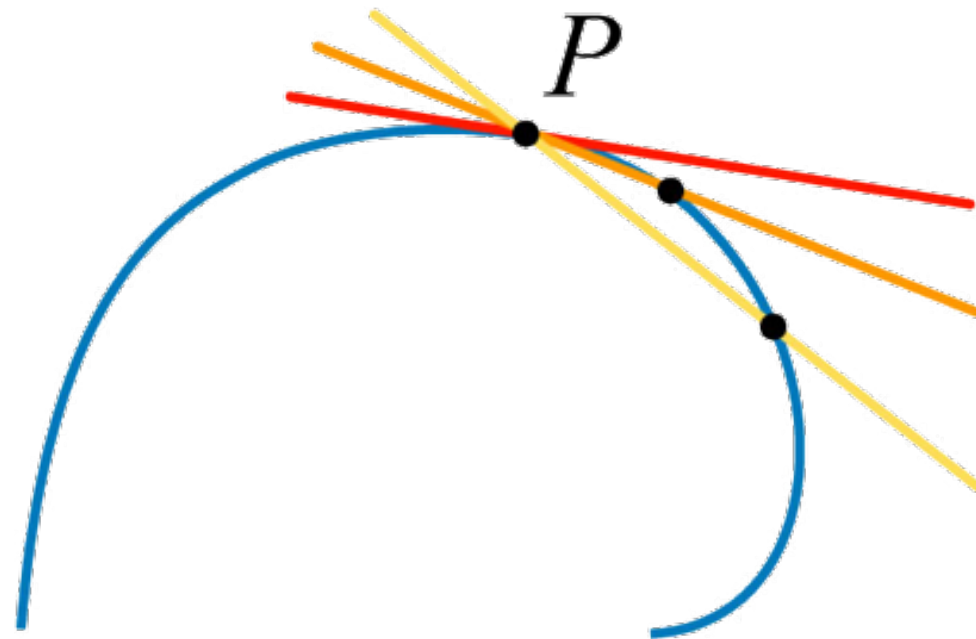
# Secant and Tangent – Parametric Form

- Secant:  $\mathbf{p}(t) - \mathbf{p}(s)$
- Tangent:  $\mathbf{p}'(t) = (x'(t), y'(t), \dots)^T$
- If  $t$  is arc-length:

$$\|\mathbf{p}'(t)\| = 1$$

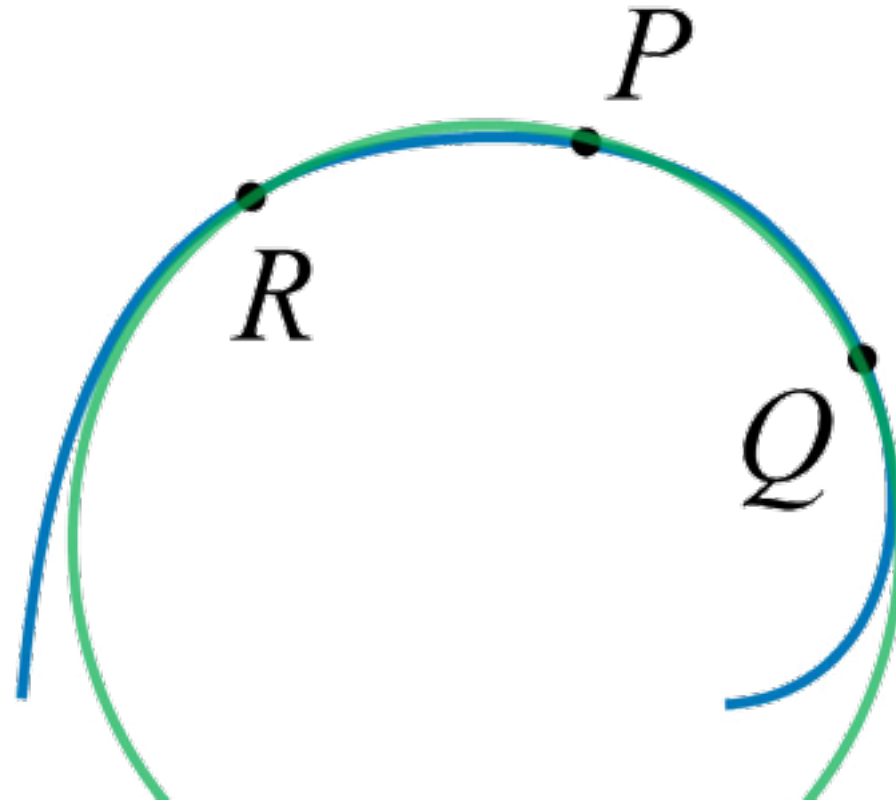
Recall

$$\text{len}(\mathbf{p}(t_0), \mathbf{p}(t)) = \int_0^t \|\mathbf{p}'(t)\| dt$$



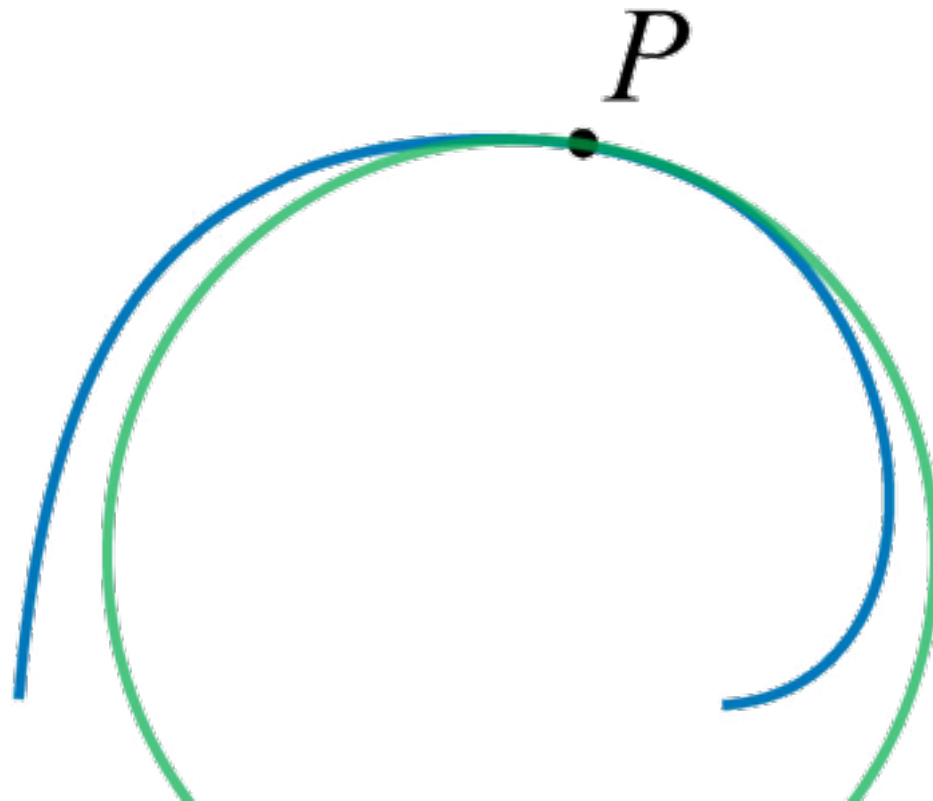
# Circle of Curvature

- Consider the circle passing through three points on the curve...

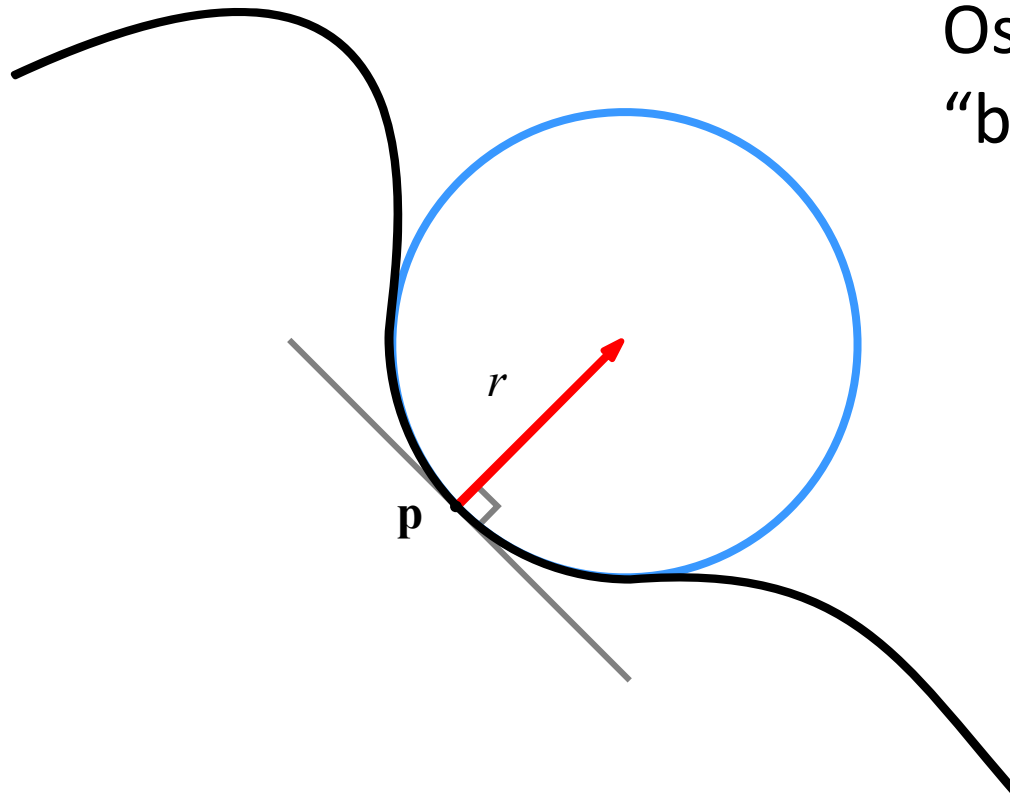


# Circle of Curvature

•...the limiting circle as three points come together.



# Tangent, normal, radius of curvature

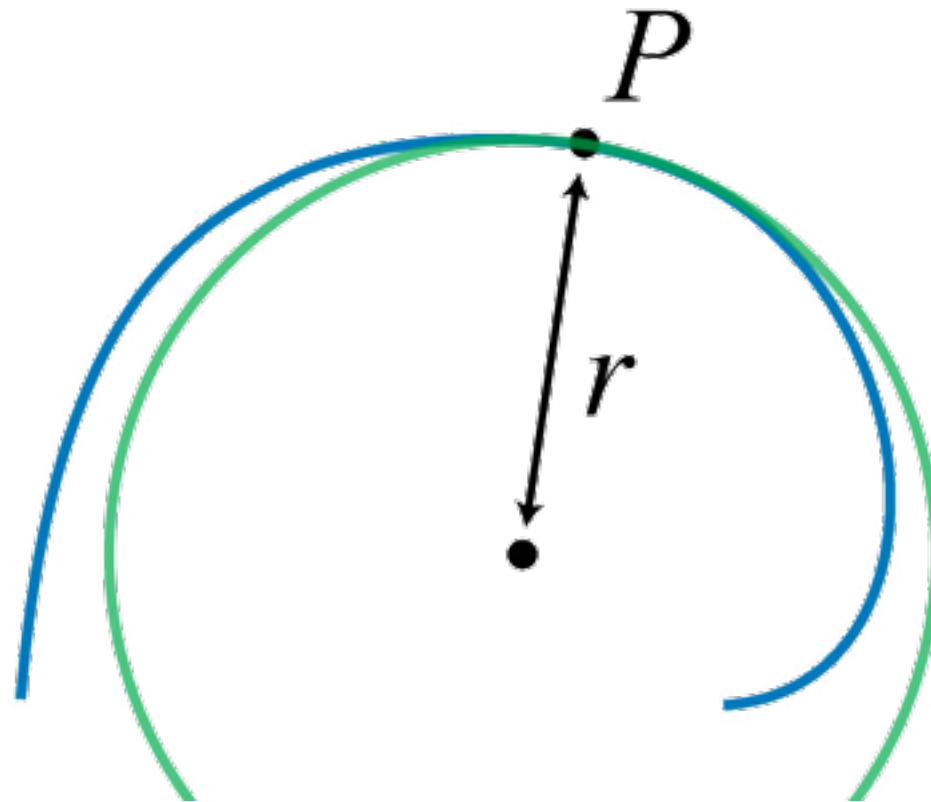


Osculating circle  
“best fitting circle”

# Radius of Curvature, $r = 1/\kappa$

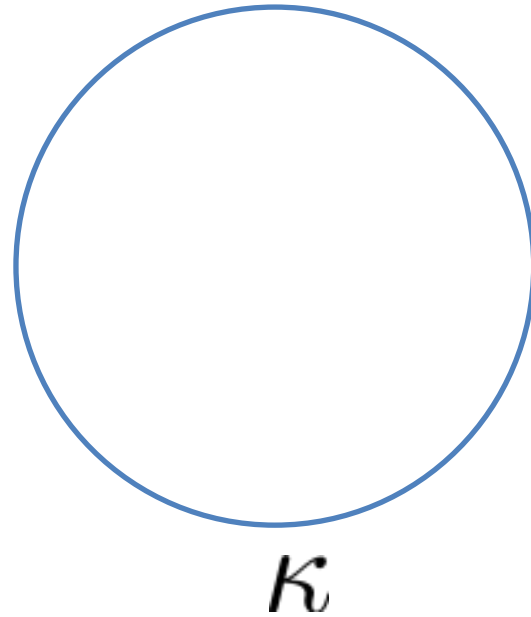
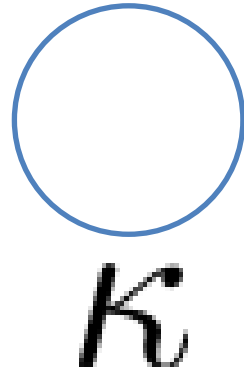
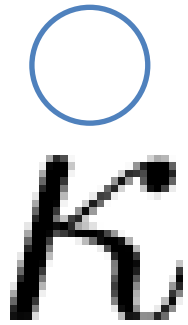
Curvature

$$\kappa = \frac{1}{r}$$



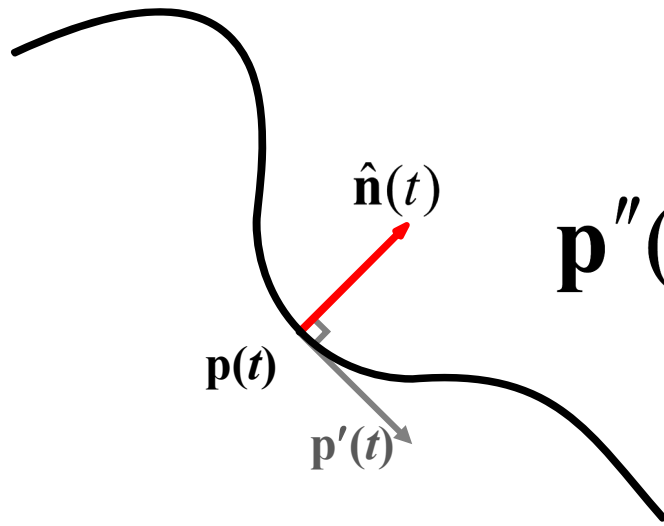
# Curvature is scale dependent

$$\kappa = \frac{1}{r}$$

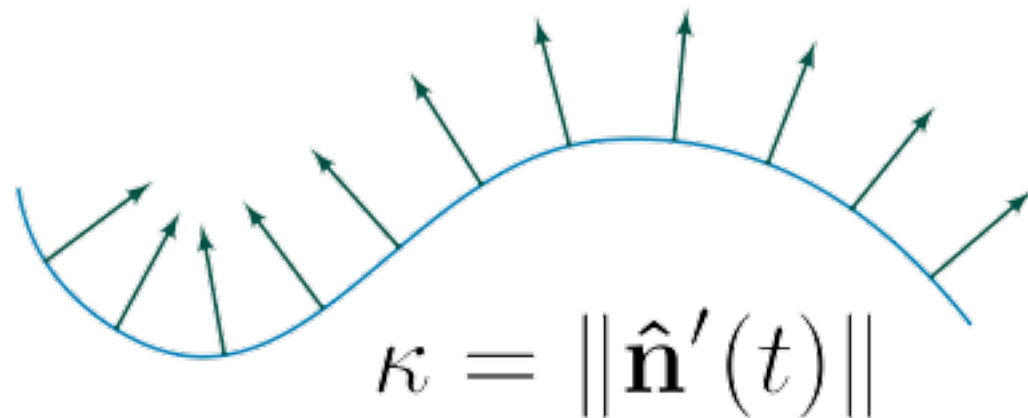


# Curvature and Normal

- Assuming  $t$  is arc-length parameter:



$$\mathbf{p}''(t) = \kappa \hat{\mathbf{n}}(t) \text{ normal to the curve}$$



# Surfaces, Parametric Form

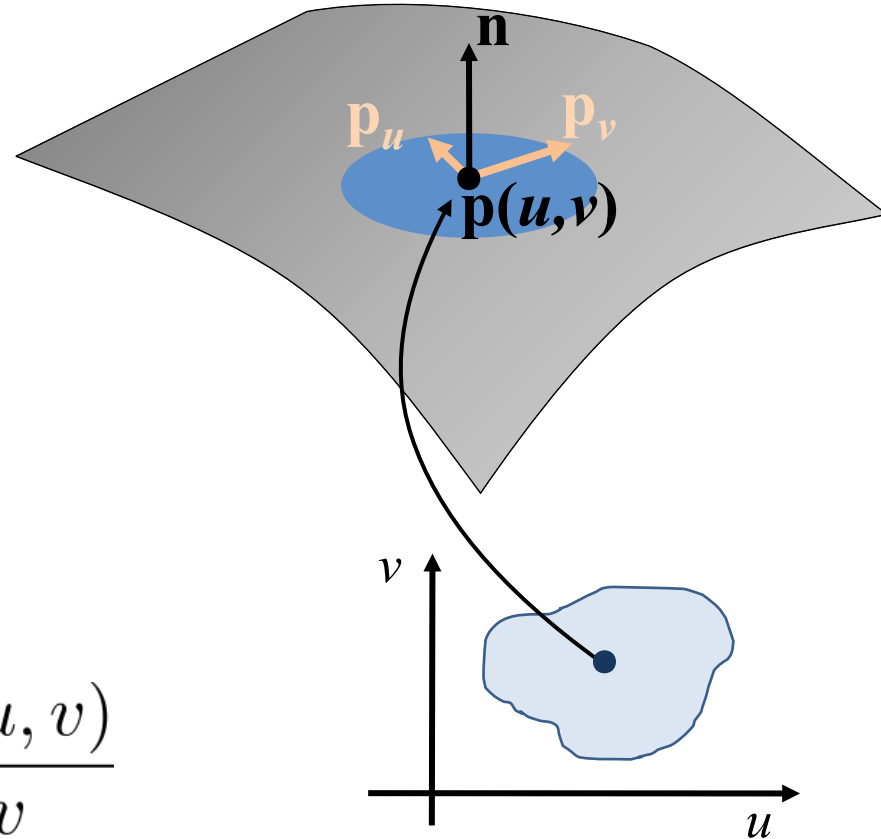
## • Continuous surface

$$\mathbf{p}(u, v) = \begin{pmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{pmatrix}, \quad (u, v) \in \mathbb{R}^2$$

## • Tangent plane at point $\mathbf{p}(u, v)$ is spanned by

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

These vectors don't have to be orthogonal



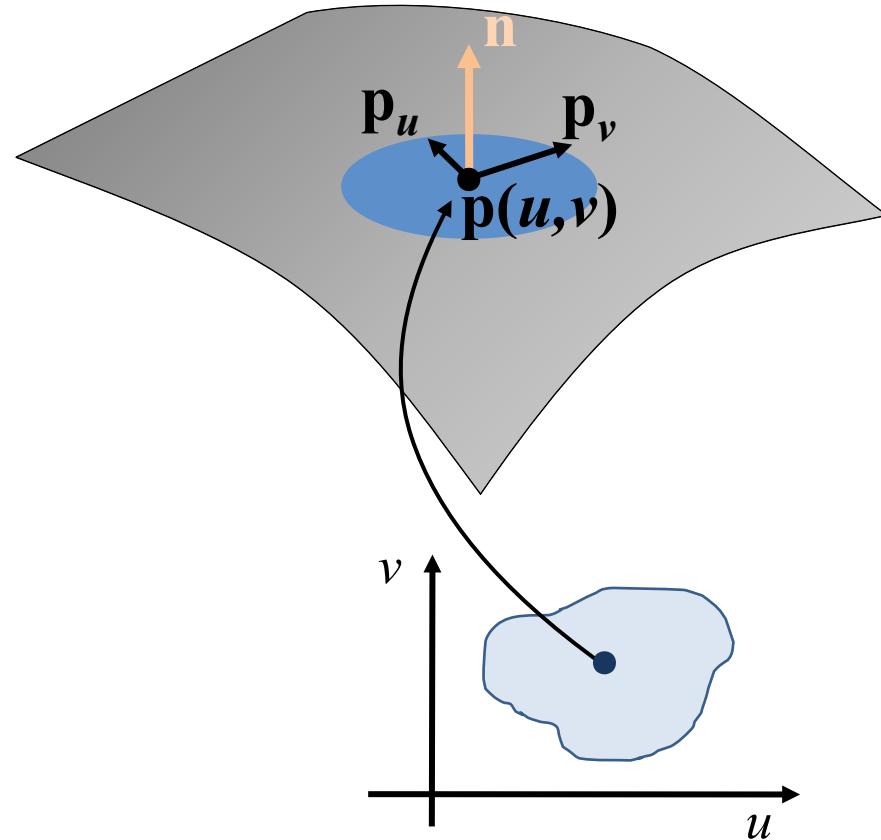
# Surface Normals

• Surface normal:

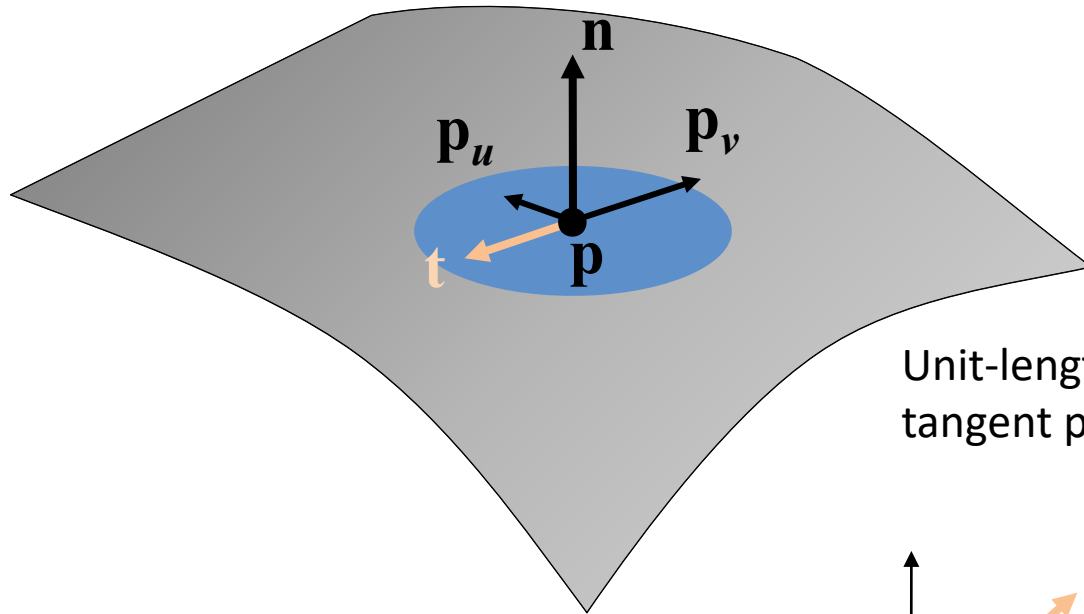
$$\mathbf{n}(u, v) = \frac{\mathbf{p}_u \times \mathbf{p}_v}{\|\mathbf{p}_u \times \mathbf{p}_v\|}$$

• Assuming *regular* parameterization, i.e.,

$$\mathbf{p}_u \times \mathbf{p}_v \neq 0$$



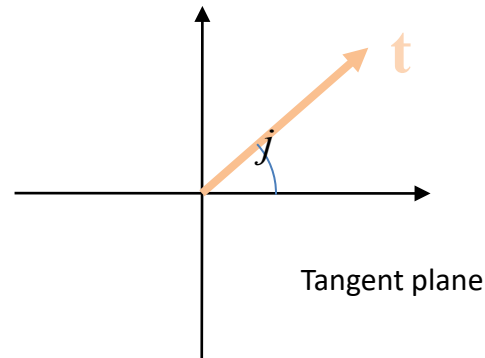
# Normal Curvature



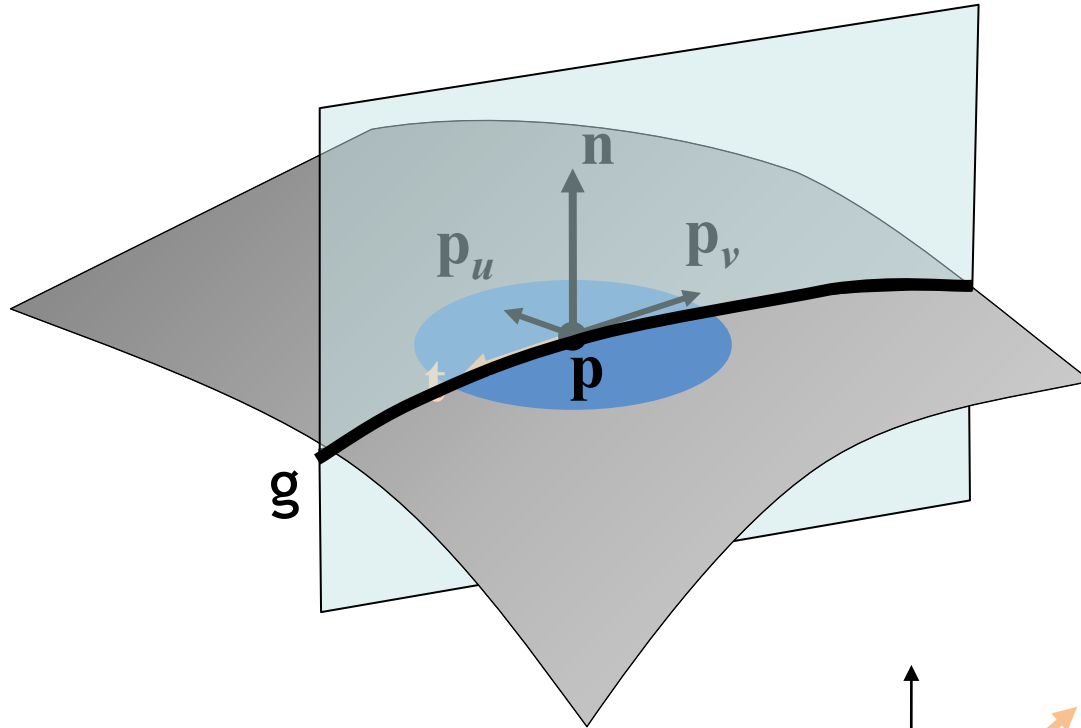
$$\mathbf{n}(u, v) = \frac{\mathbf{p}_u \times \mathbf{p}_v}{\|\mathbf{p}_u \times \mathbf{p}_v\|}$$

Unit-length direction  $\mathbf{t}$  in the tangent plane (if  $\mathbf{p}_u$  and  $\mathbf{p}_v$  are orthogonal):

$$\mathbf{t} = \cos \varphi \frac{\mathbf{p}_u}{\|\mathbf{p}_u\|} + \sin \varphi \frac{\mathbf{p}_v}{\|\mathbf{p}_v\|}$$



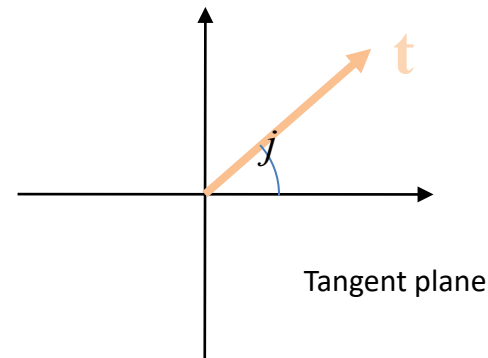
# Normal Curvature



The curve  $\gamma$  is the intersection of the surface with the plane through  $\mathbf{n}$  and  $\mathbf{t}$  - a normal section.

**Normal curvature:**

$$\kappa_n(\varphi) = \kappa(\gamma(\mathbf{p}))$$



# Surface Curvatures

## Principal curvatures

- Minimal curvature
- Maximal curvature

$$\kappa_1 = \kappa_{\min} = \min_{\varphi} \kappa_n(\varphi)$$

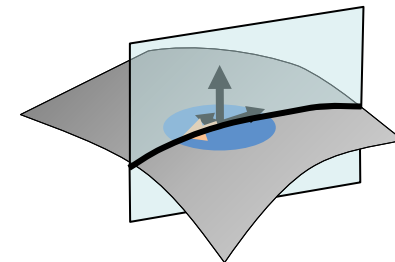
$$\kappa_2 = \kappa_{\max} = \max_{\varphi} \kappa_n(\varphi)$$

## Mean curvature

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

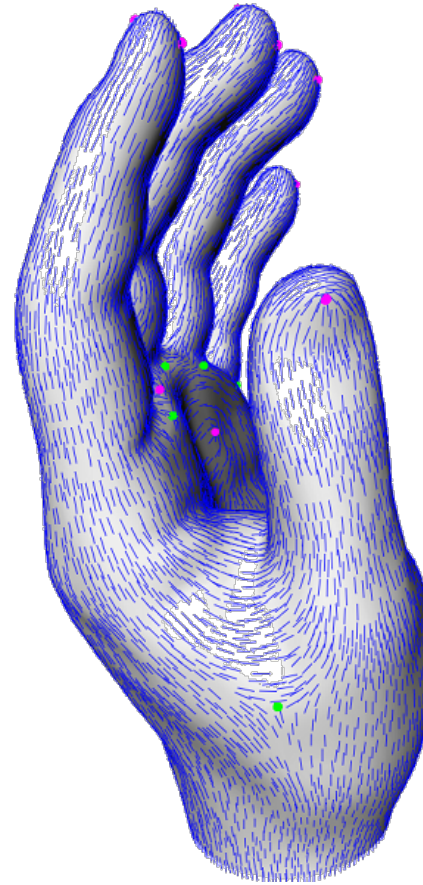
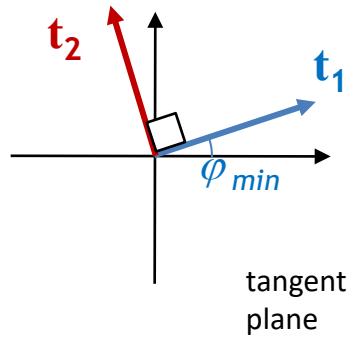
## Gaussian curvature

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

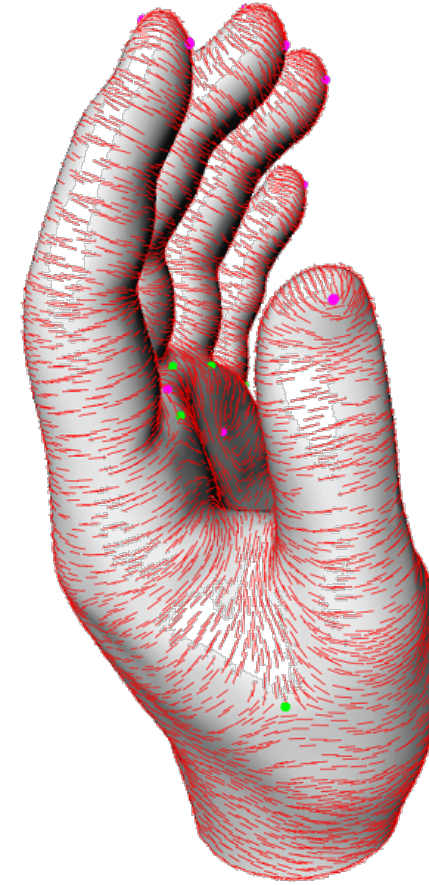


# Principal Directions

- Principal directions: tangent vectors corresponding to  $\varphi_{\max}$  and  $\varphi_{\min}$

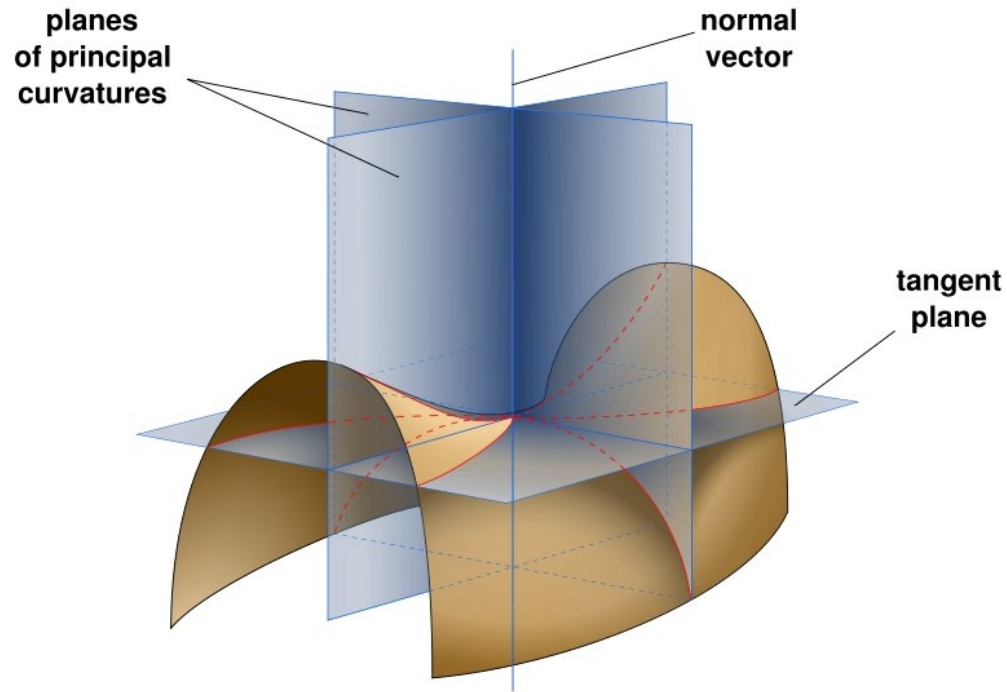


min curvature



max curvature

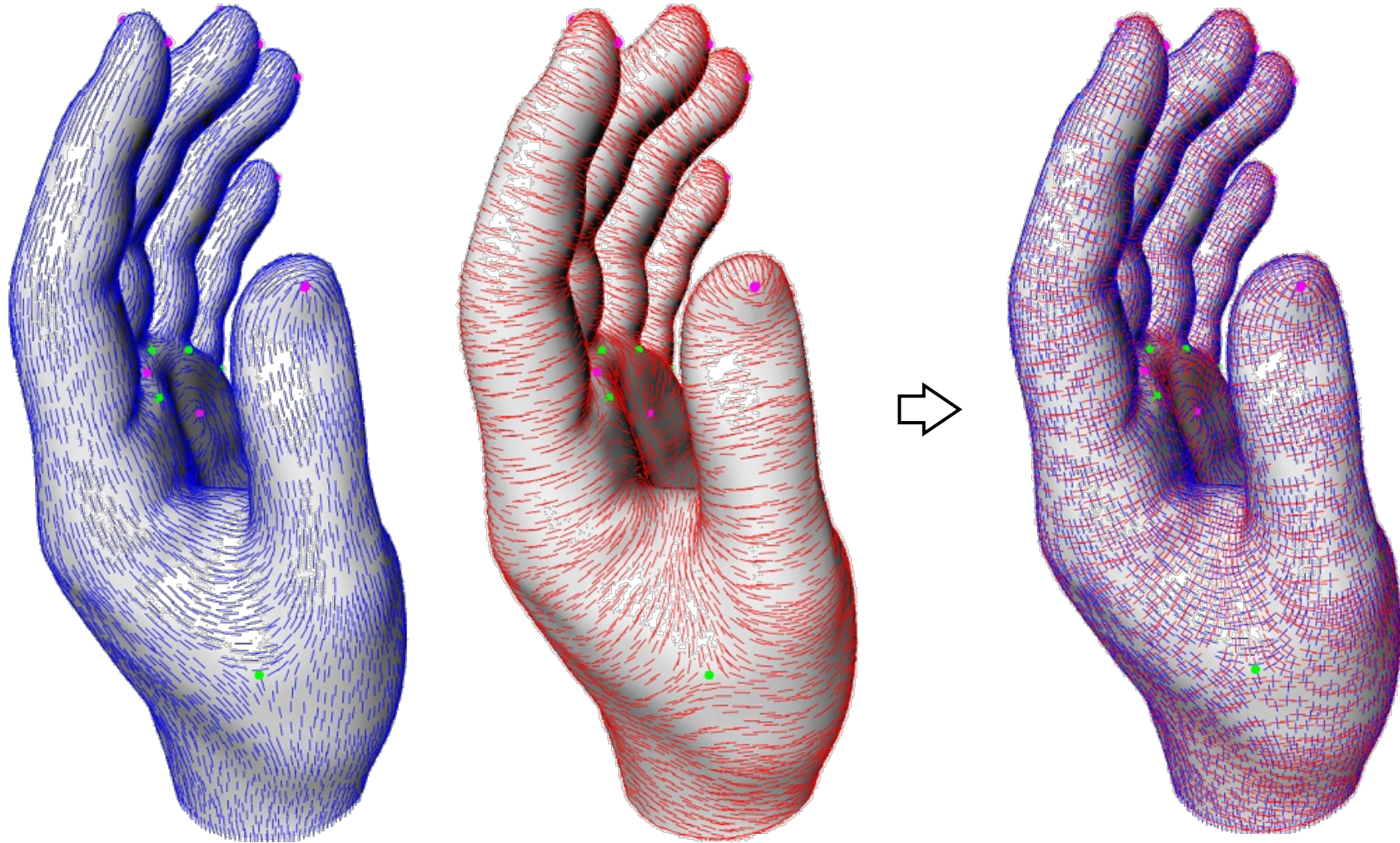
# Principal Directions



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

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

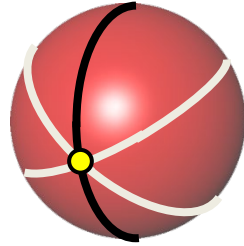
# Principal Directions



# Local Surface Shape By Curvatures

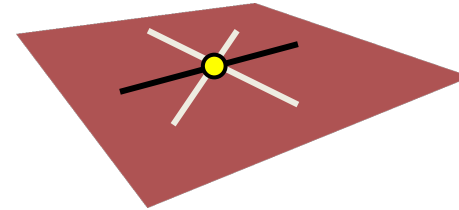
**Isotropic:**  
all directions are  
principal directions

$$K > 0, \kappa_1 = \kappa_2$$



spherical (umbilical)

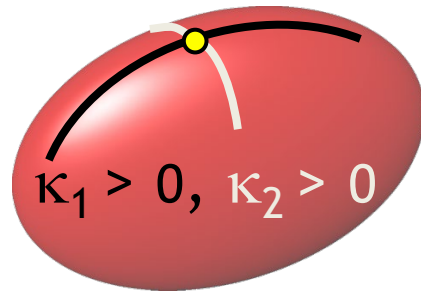
$$K = 0$$



planar

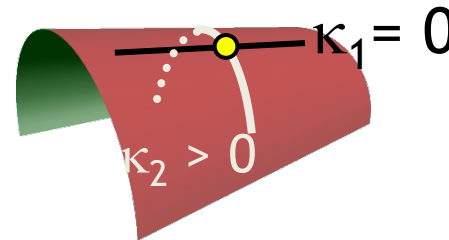
**Anisotropic:**  
2 distinct principal  
directions

$$K > 0$$



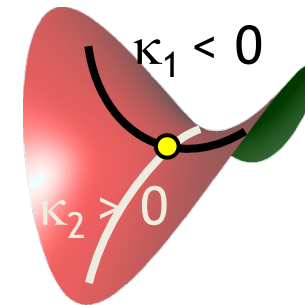
elliptic

$$K = 0$$



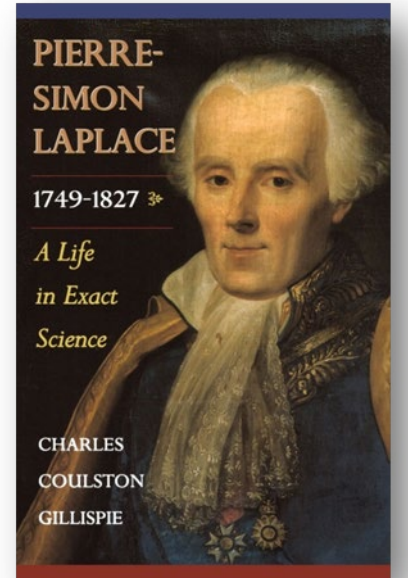
parabolic

$$K < 0$$



hyperbolic

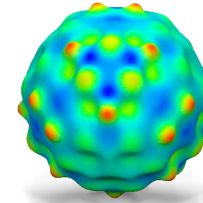
# The Laplace-Beltrami Operator



# Knowledge as Functions over Data



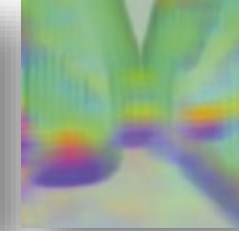
Knowledge towers over visual data: function spaces



Curvature

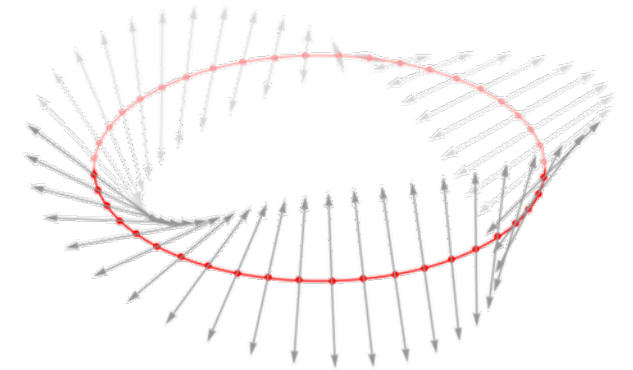


Parts



SIFT flow, C. Liu 2011

Vector bundles and sheaves



# Motivation: Information Representation

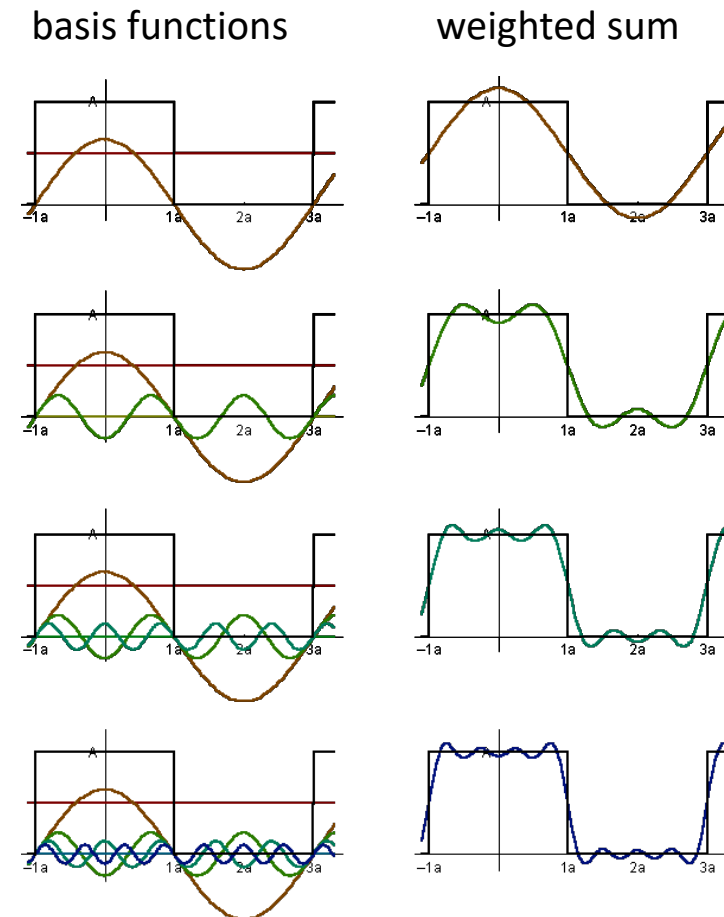
- Various surface properties can be represented as functions “living” on the surface
  - Curvatures ( $k_1, k_2, K, H$ )
  - Part indicator functions
  - Vector-valued functions, like surface coordinates, normals and texture
  - ...
- We need a tool to work with functions on surfaces
- Operators map functions to functions

# Reminder: Fourier Analysis on the Real Line

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



Joseph Fourier 1768 - 1830



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

Coefficients : co-integrate function with basis

# More Generally - Fourier Analysis

- ◆ Inner product for  $L^2$  function space  $\langle f, g \rangle := \int_{-\infty}^{\infty} f(x) \overline{g(x)} dx$
- ◆ Orthonormal basis : complex “waves”

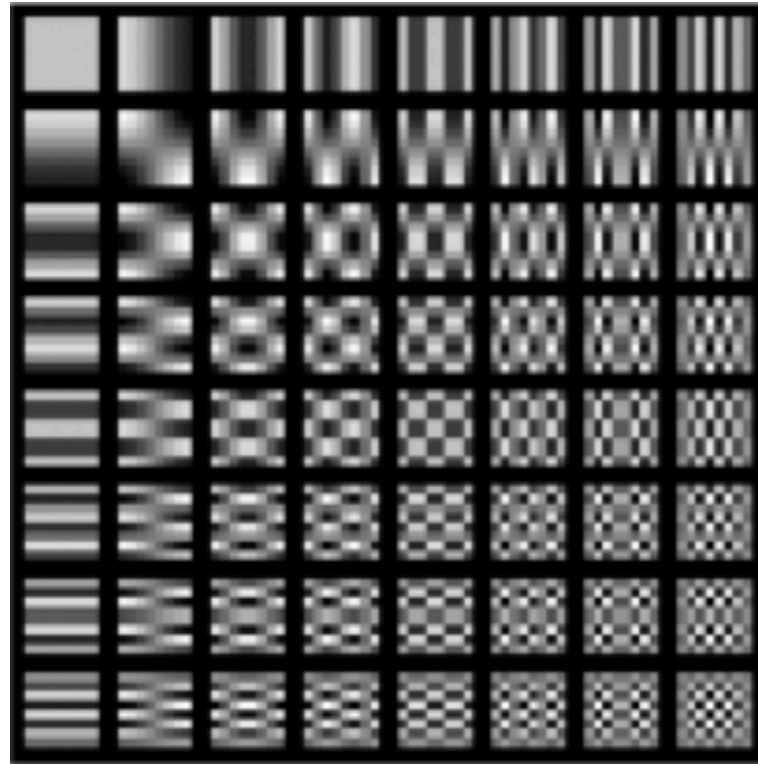
$$e_u(x) := e^{i2\pi ux} = \cos(2\pi ux) - i \sin(2\pi ux)$$

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



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

# Fourier on Rectangular 2D (or 3D) Domains

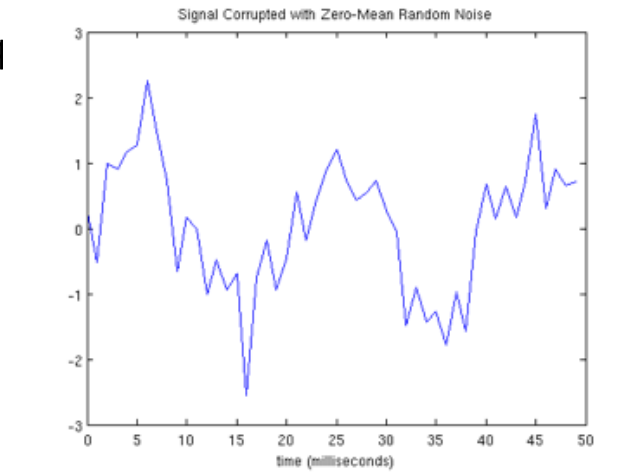


Fourier (DCT) basis functions for 8x8 grayscale images

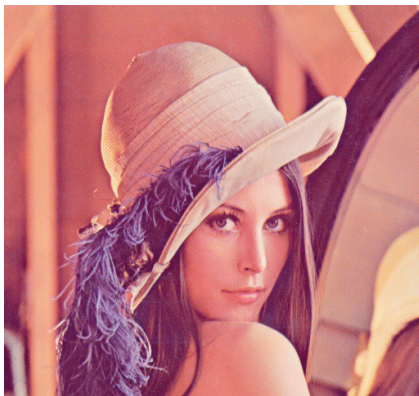
$$\cos(2\pi\omega_h) \cos(2\pi\omega_v)$$

# Image Processing: Filtering

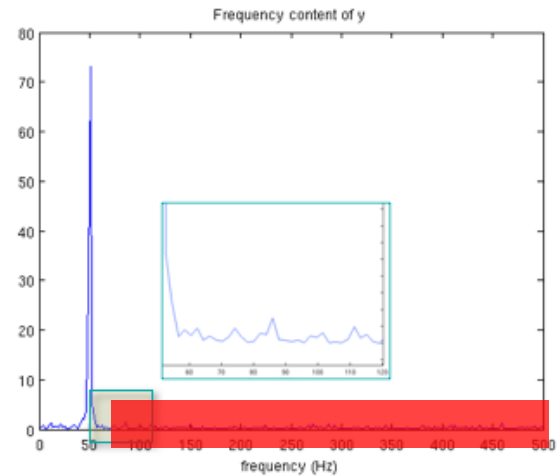
## ◆ Freq



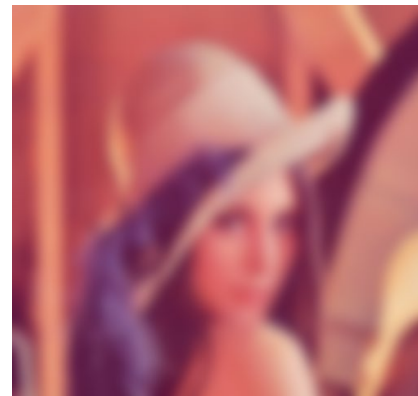
spatial domain



air



frequency domain



- ◆ Spatial domain  $f(x)$  → Frequency domain  $F(u)$

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi ux} dx$$

- ◆ Multiply by low-pass filter  $G(u)$

$$F(u) \leftarrow F(u) \cdot G(u)$$

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{i2\pi ux} du$$

# Extend Fourier to Meshes?

- ◆ Fourier basis functions are eigenfunctions of the (standard) Laplace operator  $\Delta: L^2 \rightarrow L^2$

$$\Delta f = \operatorname{div} \nabla f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

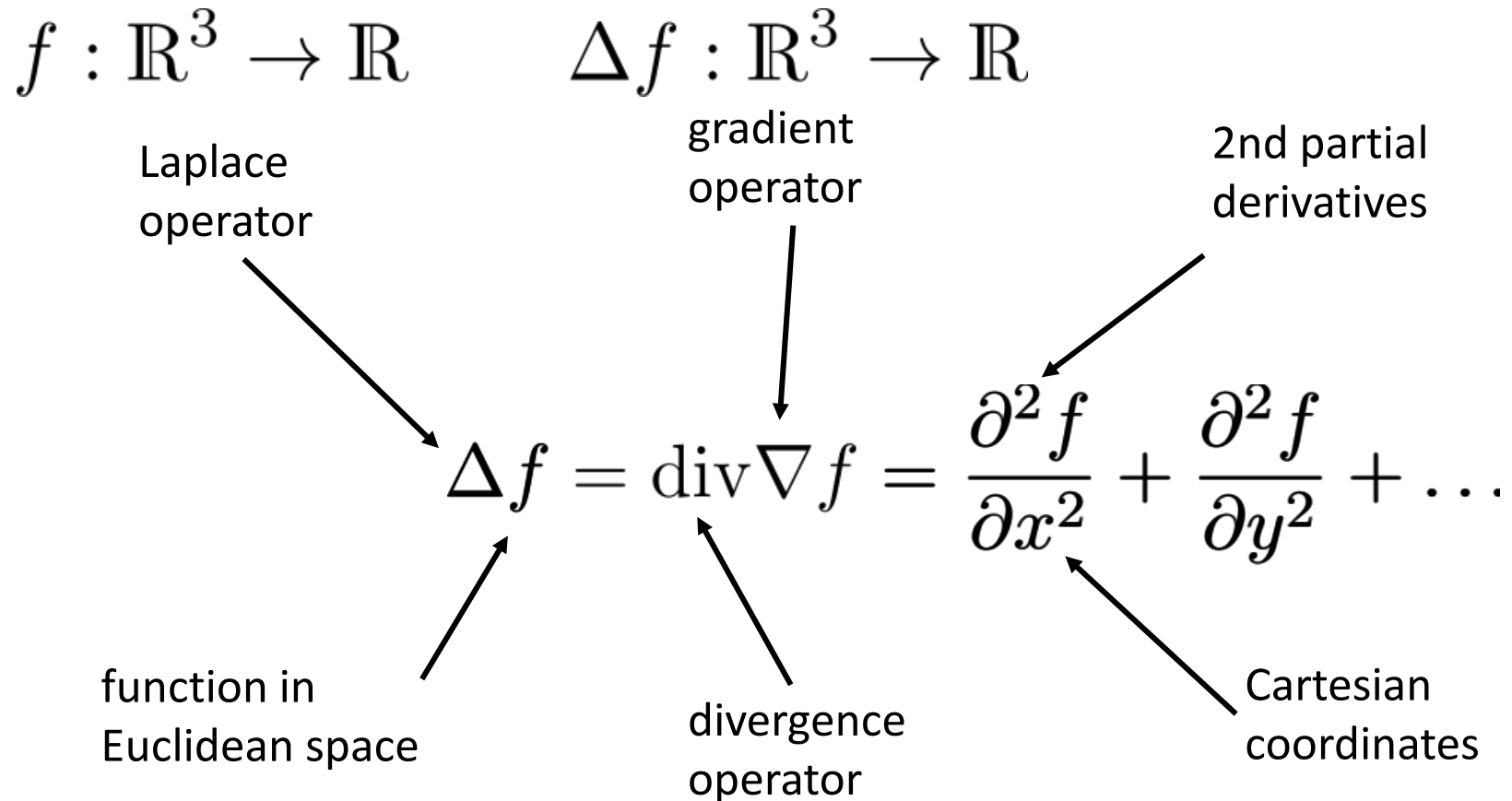
$$\Delta (e^{2\pi i \omega x}) = \frac{\partial^2}{\partial x^2} e^{2\pi i \omega x} = -(2\pi \omega)^2 e^{2\pi i \omega x}$$

- ◆ We need
  - ◆ A version of this operator for 2D manifolds, and
  - ◆ A discrete (mesh-based) version!

# Laplace-Beltrami on 2-Manifolds

Fourier analysis on manifolds

# Continuous Laplace Operator in $\mathbb{R}^3$



$$\operatorname{grad} f = \nabla f = \left( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)$$

$$\operatorname{div} \mathbf{F} = \nabla \cdot \mathbf{F} = \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z}$$

# Continuous Laplace-Beltrami Operator

- Extension of Laplace operator to functions on manifolds

$$f : \mathcal{M} \rightarrow \mathbb{R}$$

function on surface  $M$

$$\Delta f : \mathcal{M} \rightarrow \mathbb{R}$$

Laplace-Beltrami

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

divergence operator

gradient operator

gradient

divergence

# Continuous Setting: Laplace-Beltrami

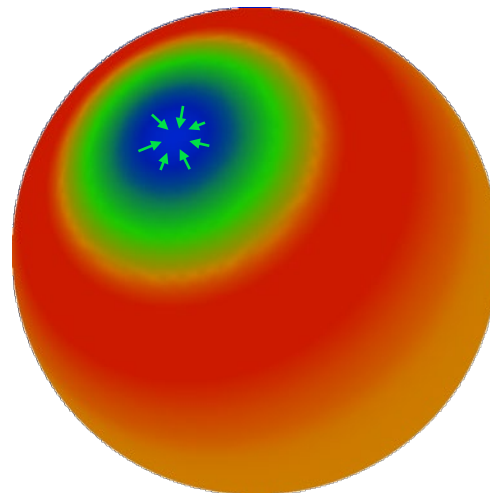
## Definition:

Given a surface without boundary  $\mathcal{M}$ , define the Laplace-Beltrami operator on  $\mathcal{M}$

$$\Delta : C^\infty(\mathcal{M}) \rightarrow C^\infty(\mathcal{M}), \Delta f = \operatorname{div} \nabla f$$

divergence of the gradient of  $f$ .

$\operatorname{div} \nabla f$



# Continuous Setting: Laplace-Beltrami

## Definition:

Given a surface without boundary, define the Laplace-Beltrami operator

$$\Delta : C^\infty(\mathcal{M}) \rightarrow C^\infty(\mathcal{M}), \quad \Delta f = \operatorname{div} \nabla f$$

divergence of the gradient of  $f$ .

Divergence key property:

$$\int_{\mathcal{M}} -f \operatorname{div} X d\mu = \int_{\mathcal{M}} \langle X, \nabla f \rangle d\mu.$$

Thus, on a surface we also have, for any pair of smooth functions:

$$\int_{\mathcal{M}} g \Delta f d\mu = \int_{\mathcal{M}} \langle \nabla g, \nabla f \rangle d\mu = \int_{\mathcal{M}} f \Delta g d\mu$$

# Continuous Setting: Laplace-Beltrami

## Definition:

Given a surface, define the inner product between two functions as:

$$\langle f, g \rangle = \int_{\mathcal{M}} f(x)g(x)d\mu(x)$$

Then:  $\Delta$  is a symmetric positive semi-definite operator:

$$\langle f, \Delta g \rangle = \langle g, \Delta f \rangle$$

And:

$$\langle f, \Delta f \rangle = \int_{\mathcal{M}} \|\nabla f\|^2 d\mu$$

As such,  $\Delta$  has a countable set of *eigenfunctions* with real positive, eigenvalues .

# Continuous Setting: Laplace-Beltrami

## Basic Properties:

For a compact surface,  $\mathcal{M}$ , the Laplace-Beltrami operator  $\Delta$  has a countable set of eigenfunctions with real positive, eigenvalues

- Each eigenvalue has finite multiplicity.
- Multiplicity of 0 is the number of connected components
- Eigenfunctions are orthonormal:

$$\langle \phi_i, \phi_j \rangle = 0, \quad \langle \phi_i, \phi_i \rangle = 1$$

- For any square integrable function:

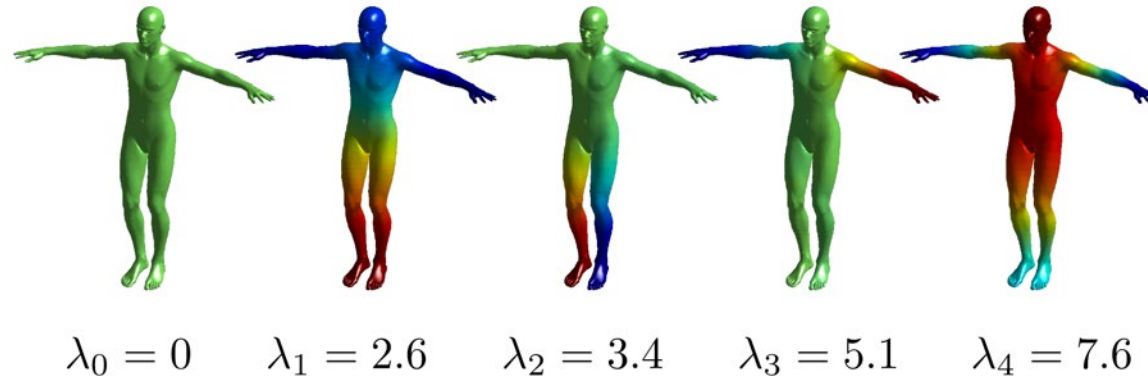
$$f(x) = \sum_{i=0}^{\infty} a_i \phi_i(x)$$

$$a_i = \langle f, \phi_i \rangle$$

# Continuous Setting: Laplace-Beltrami

## Multiscale nature of the spectrum:

Intuitively, eigenfunctions corresponding to larger eigenvalues, capture *smaller details* (higher frequency) of the geometry.



- $n$ -th eigenfunction has at most  $n$  nodal domains.
- Integral of the gradient increases.

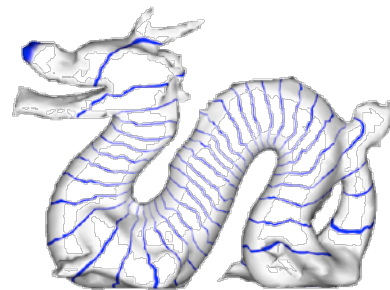
$$\lambda_i = \int_{\mathcal{M}} \phi_i \Delta \phi_i d\mu = \int_{\mathcal{M}} \|\nabla \phi_i\|^2 d\mu$$

# Continuous Setting: Laplace-Beltrami

## Multiscale nature of the spectrum:

- Fiedler *function*: minimizes the gradient, being orthogonal to the constant.

$$\phi_1 = \operatorname{argmin}_f \frac{\langle f, \Delta f \rangle}{\|f\|^2} = \frac{\int_{\mathcal{M}} \|\nabla f\|^2 d\mu}{\|f\|^2} \text{ s.t. } \int_{\mathcal{M}} f d\mu = 0$$



# Continuous Setting: Laplace-Beltrami

## Signal Processing on a manifold (generalizing Fourier analysis):

Given a function  $f : \mathcal{M} \rightarrow \mathbb{R}$

$$f(x) = \sum_{i=0}^{\infty} \phi_i(x) \langle \phi_i, f \rangle$$

Filter out high frequency “noise”, by truncating the series early:

$$f'(x) = \sum_{i=0}^N \phi_i(x) \langle \phi_i, f \rangle$$

New function will preserve the “global” properties of  $f$ .

# Laplace-Beltrami and Curvature

- ◆ Apply operator to coordinate functions

Laplace-Beltrami

coordinate functions on surface  $M$   
 $\mathbf{p} = (x, y, z)$

gradient operator

divergence operator

mean curvature

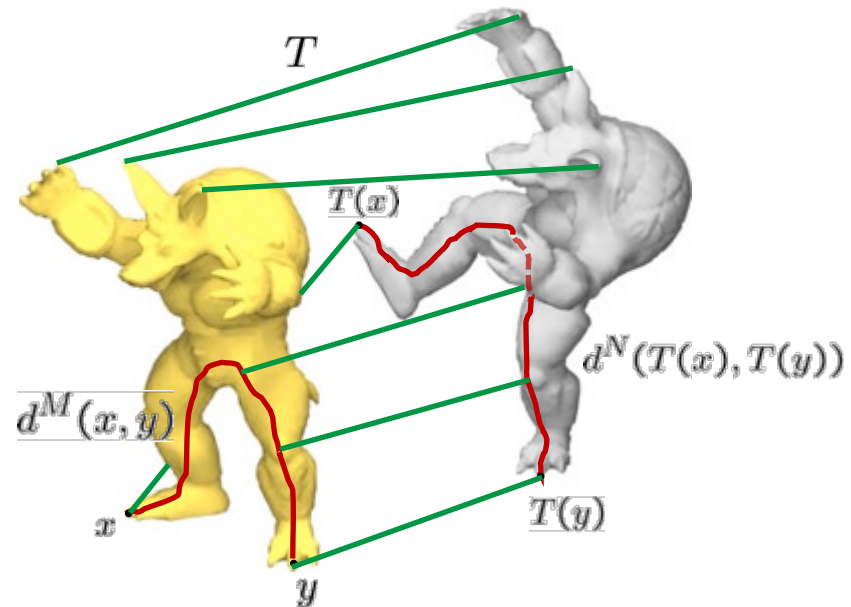
unit surface normal

$$\Delta_{\mathcal{M}} \mathbf{p} = \operatorname{div}_{\mathcal{M}} \nabla_{\mathcal{M}} \mathbf{p} = -2H \mathbf{n} \in \mathbb{R}^3$$

# Laplace-Beltrami – Isometry Invariant

## Isometry invariance (LB is intrinsic)

If two shapes are isometric then their LB operators agree.

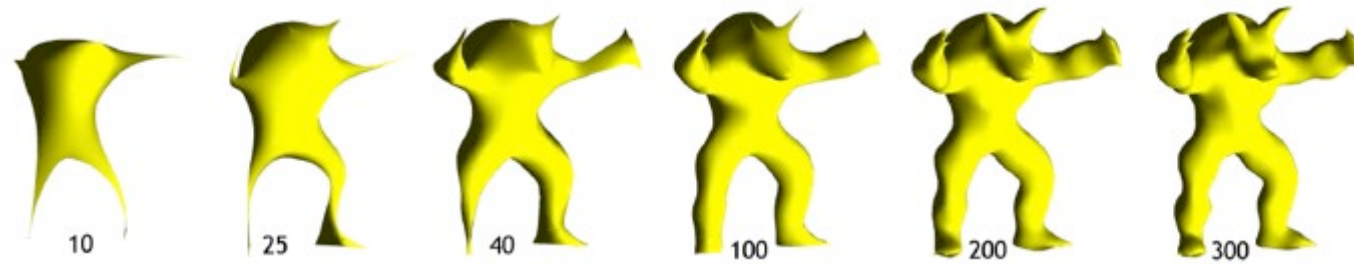


Any quantity derived from the LB operator has to be invariant to isometries.

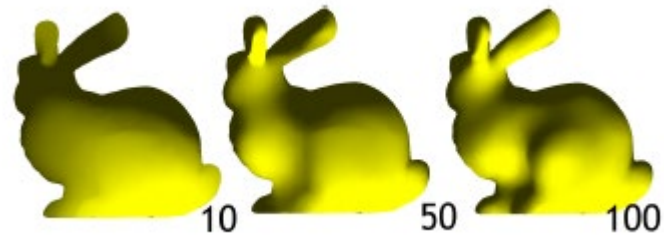
# Continuous Setting: Laplace-Beltrami

**Signal Processing on a manifold (generalizing Fourier analysis):**

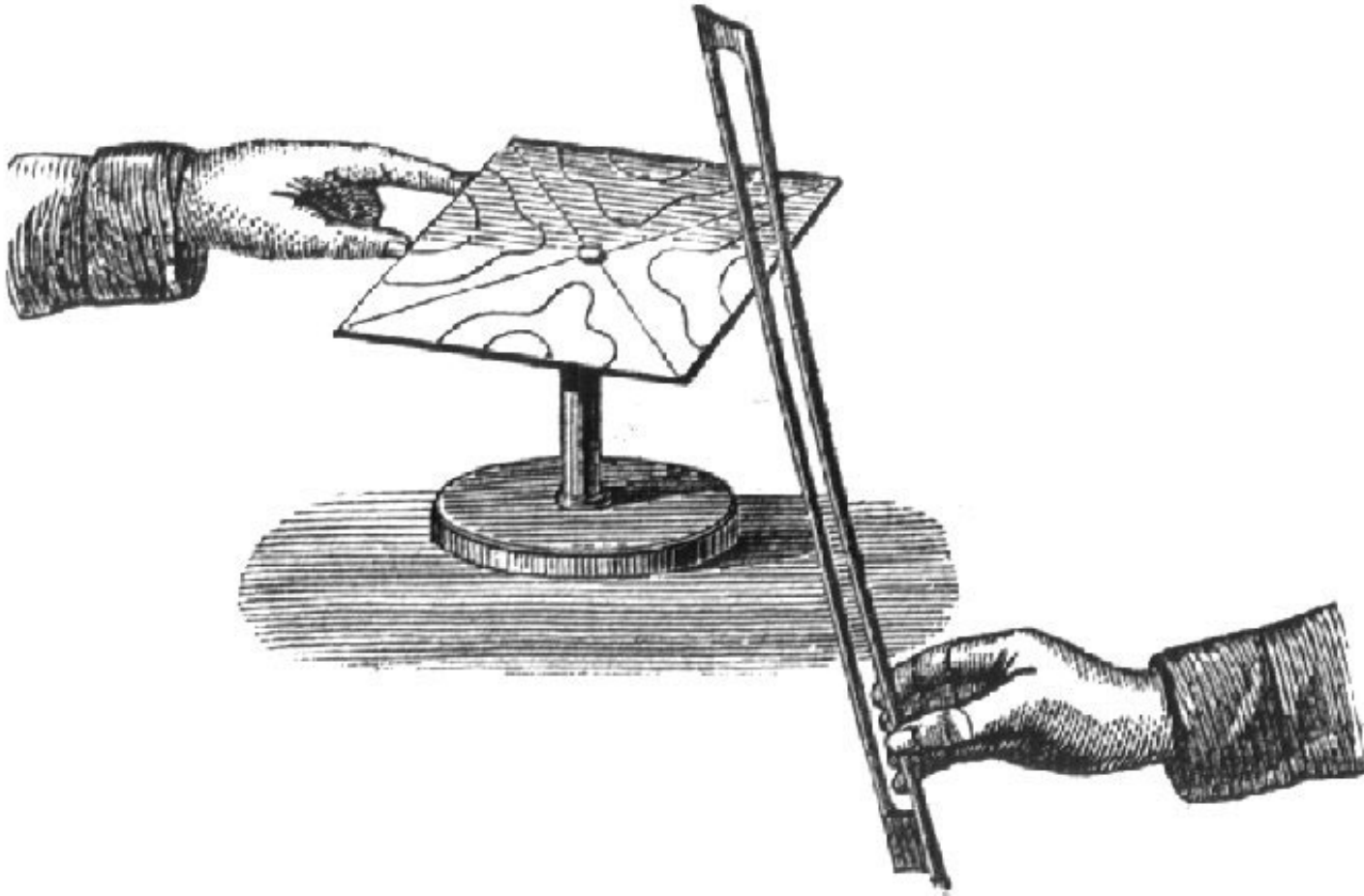
Reconstructing Geometry:



Reconstructing Normals:



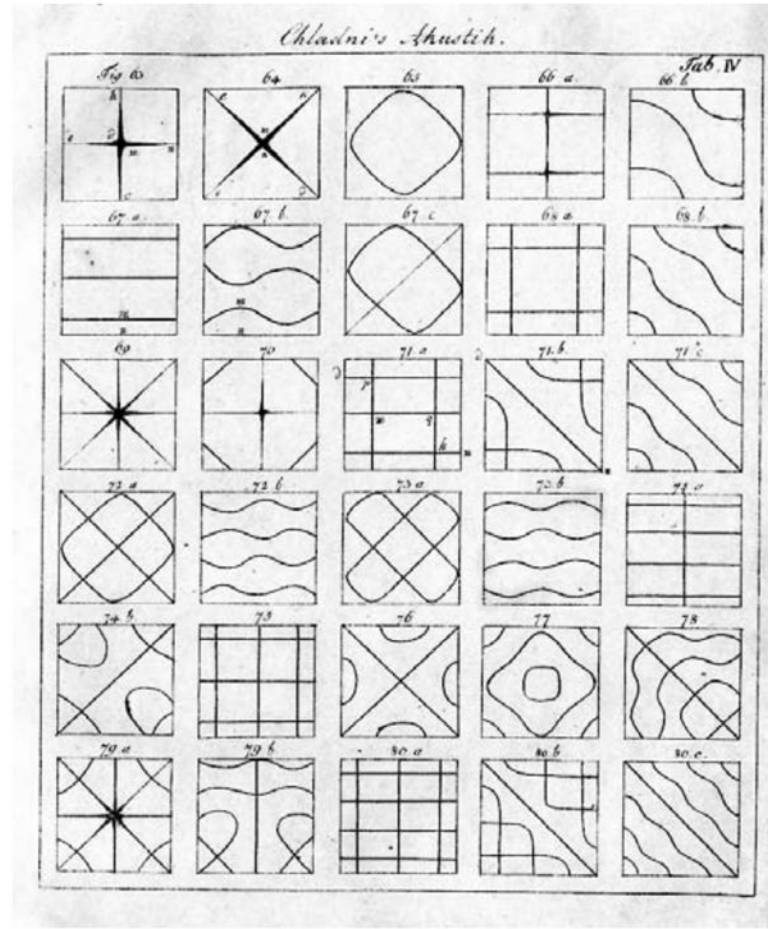
# Digression: Seeing Sound



Ernst Chladni ['kladni]  
(1715-1782)

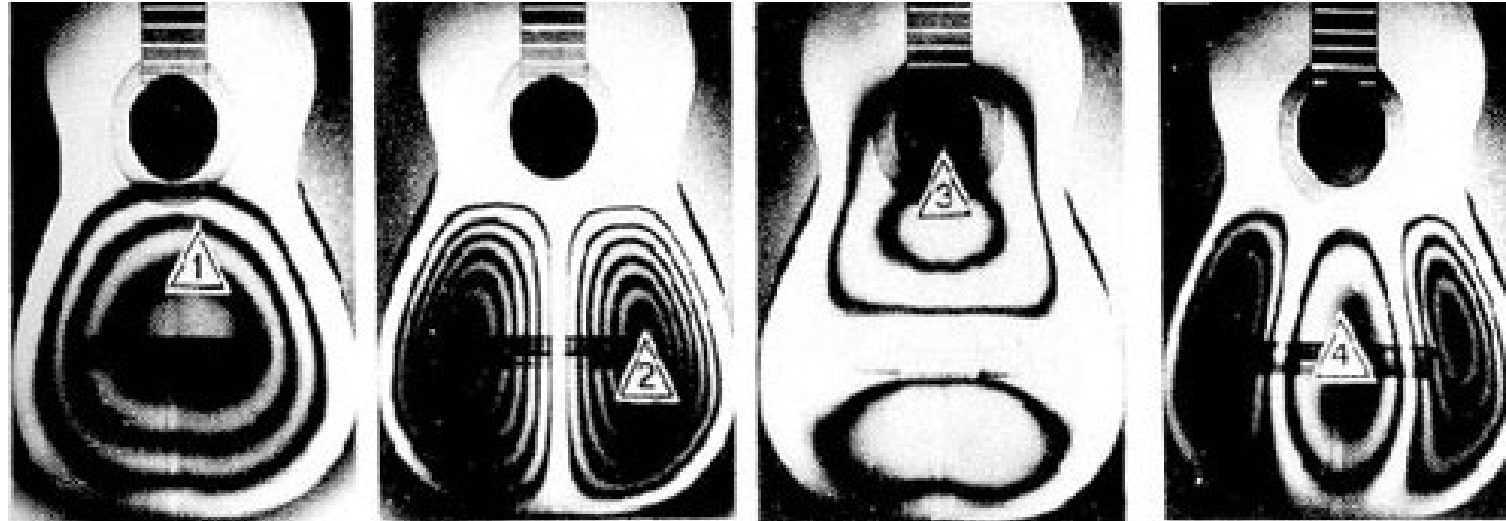
**Chladni's experimental setup allowing to visualize acoustic waves**

# Seeing Sound – Chladni Plates



Gander, Martin J., and Gerhard Wanner. "From Euler, Ritz, and Galerkin to modern computing." *Siam Review* 54.4 (2012): 627-666.

# Chladni Plates

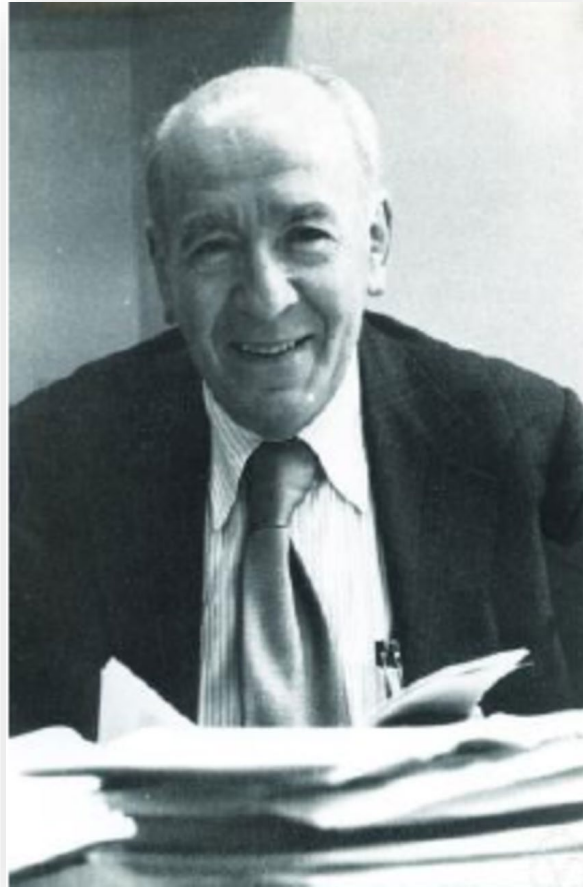


Patterns seen by Chladni are solutions to **stationary Helmholtz equation**

$$\Delta_X f = \lambda f$$

Solutions of this equation are **eigenfunctions** of Laplace-Beltrami operator

# Can One Hear the Shape of a Drum?



Mark Kac  
(1914-1984)



**More prosaically: can one reconstruct the shape  
(up to an isometry) from its Laplace-Beltrami spectrum?**

# Can One Hear the Shape of a Drum?

In Chladni's experiments, the spectrum describes acoustic characteristics of the plates ("modes" of vibrations)

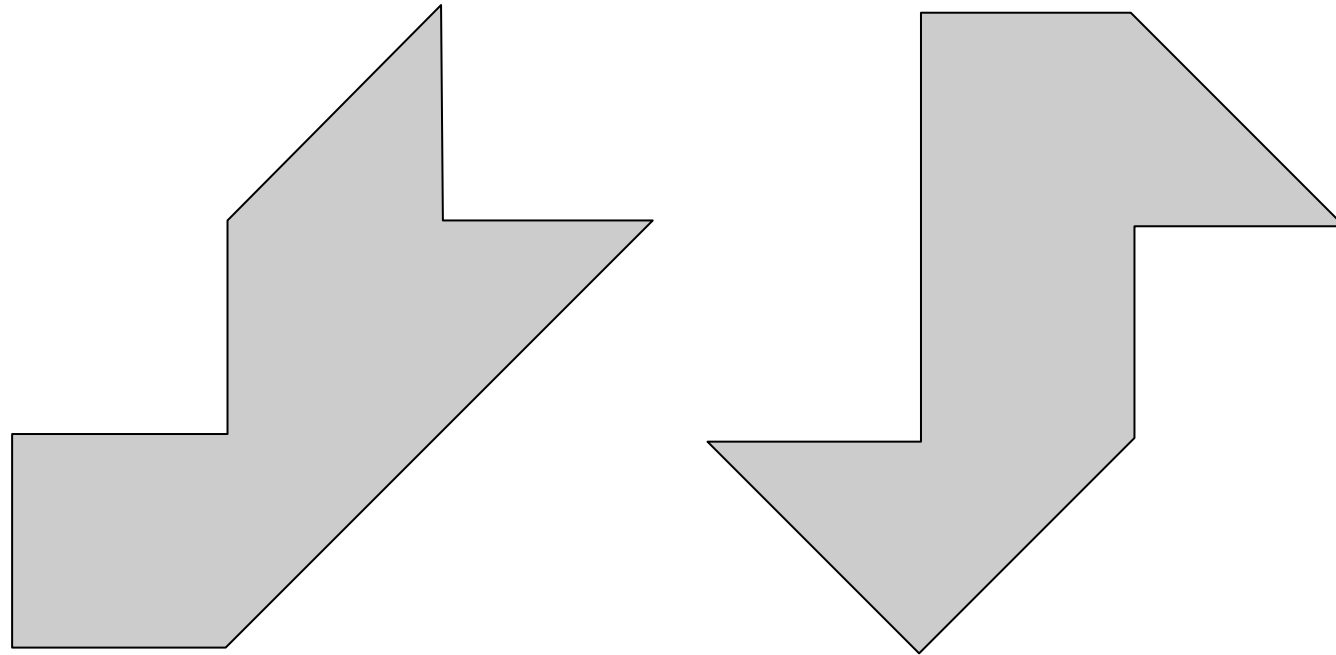
What can be "heard" from the spectrum:

- Total Gaussian curvature
- Euler characteristic
- Area

Can we "hear" the metric (shape)?

# One Cannot Hear the Shape of a Drum

**[Gordon et al. 1991]:**

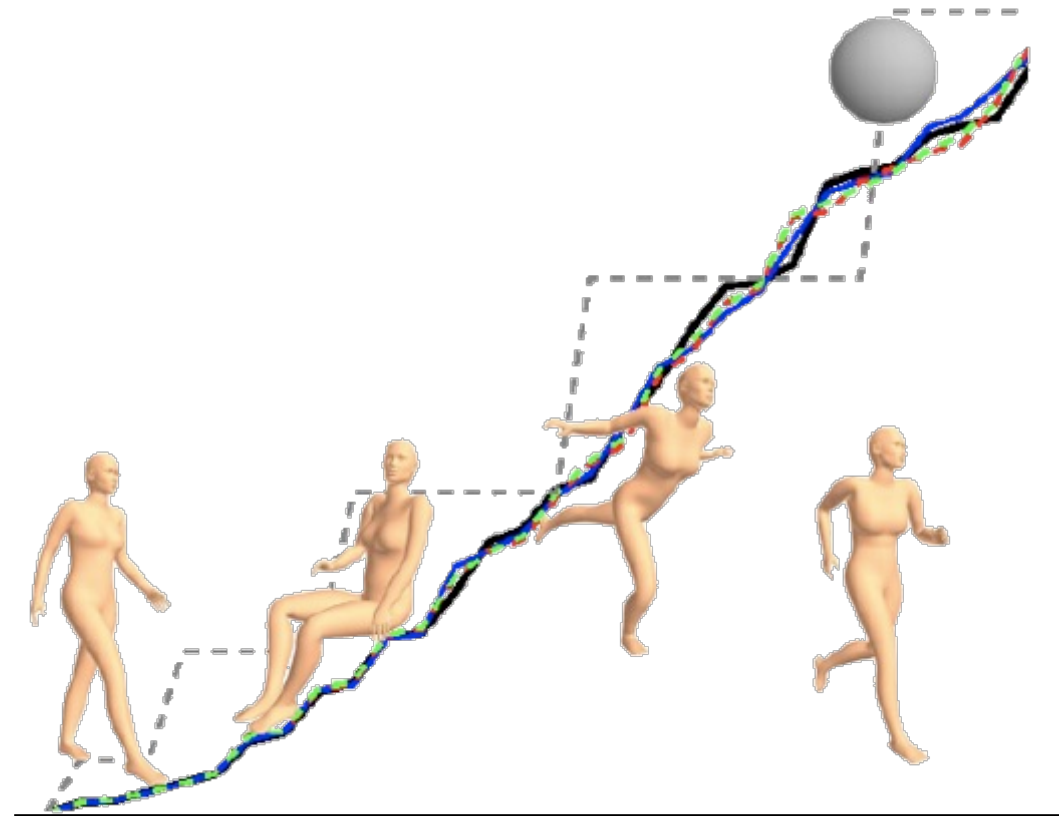


**Counter-example of isospectral but not isometric shapes**

Carolyn Gordon, David Webb, and Scott Wolpert, 1992

# Shape DNA

[Reuter et al. 2006]: use the Laplace-Beltrami spectrum  $\{\lambda_i\}_{i \geq 1}$  as an isometry-invariant shape descriptor ("shape DNA")

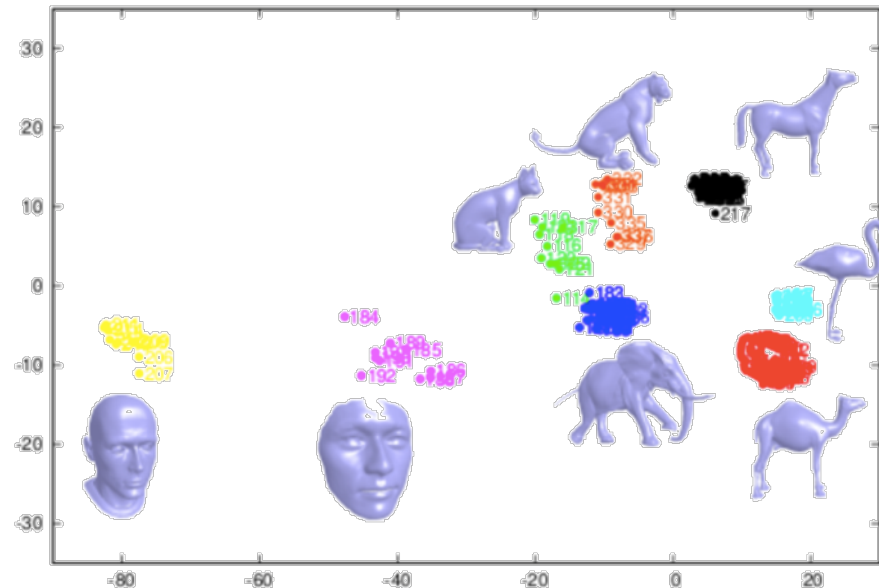


Laplace-Beltrami spectrum

Images: Reuter et al.

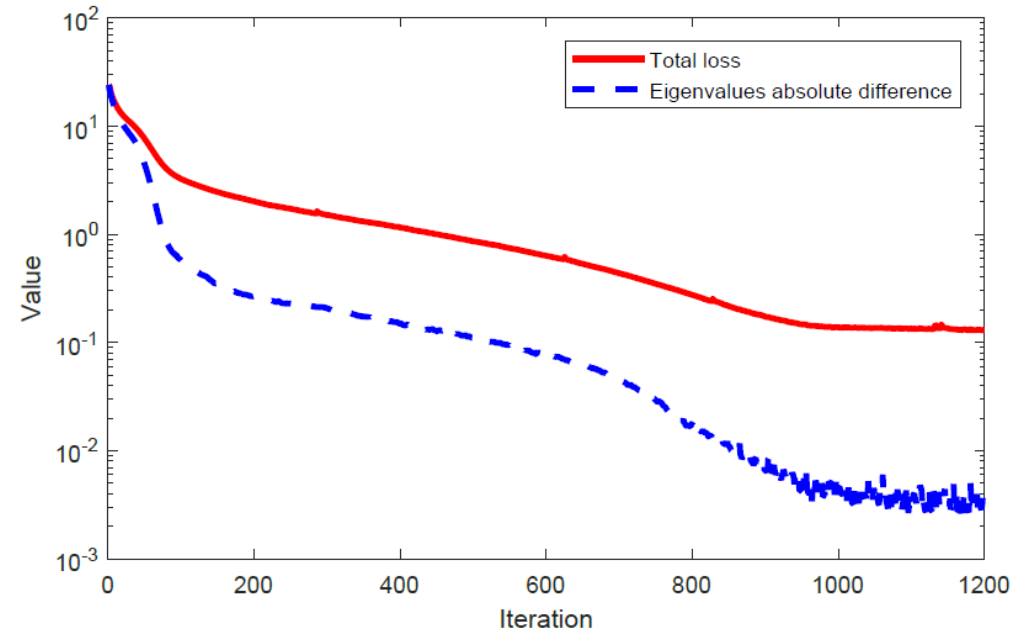
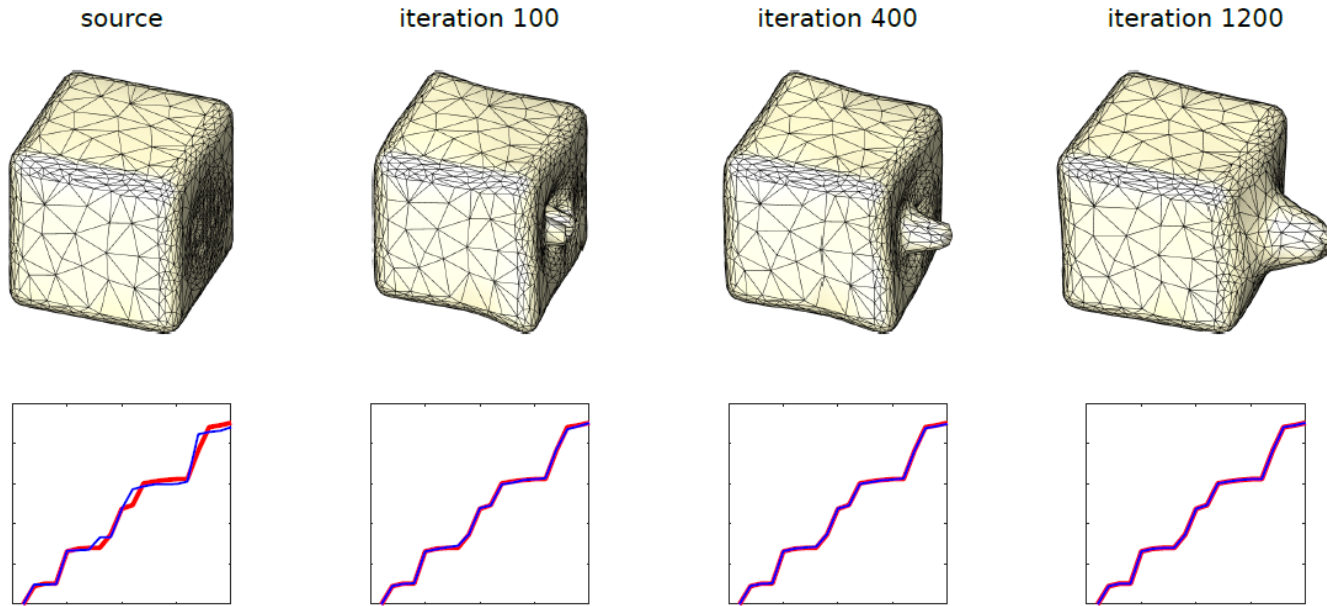
# Shape DNA

1. For each shape in the collection, compute its LB operator.
2. Find the  $k$  smallest eigenvalues and store them in a vector.
3. Compare the shapes by comparing the corresponding vectors.



Reuter et al., Laplace–Beltrami spectra as 'Shape-DNA', 2006

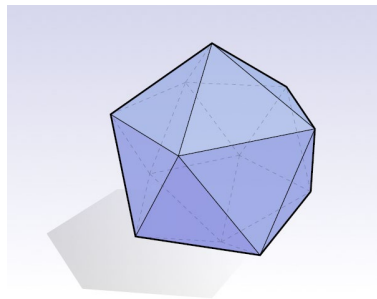
# Reconstruction from LB Spectrum: Isospectralization



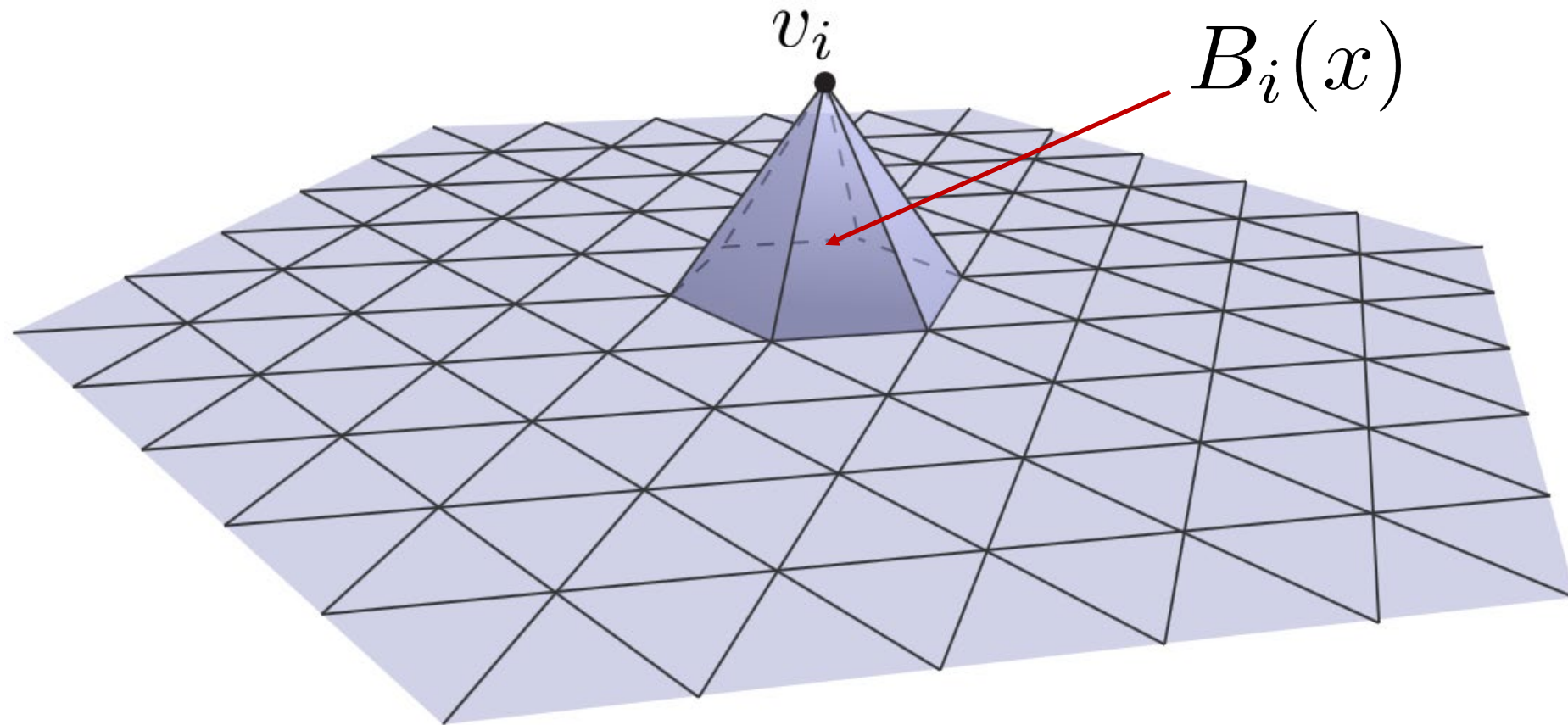
Luca Cosmo, Mikhail Panine, Arianna Rampini, Maks Ovsjanikov, Michael M. Bronstein, Emanuele Rodolà  
Isospectralization, or How to Hear Shape, Style, and Correspondence, 2019

# Discretization: Laplace-Beltrami on Meshes

Fourier analysis on meshes



# Discretized Functions on Meshes



# Mesh Setting: Laplace Beltrami

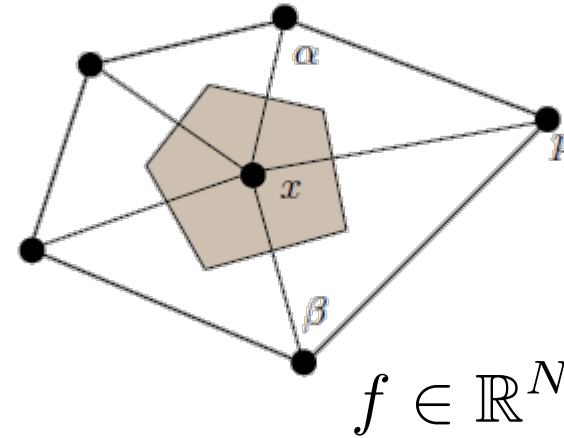
**Computing Laplace-Beltrami operator on a mesh:**

Functions are real-values defined on the triangles

Strategy: find an operator  $L$  that will satisfy

The discrete version of divergence theorem:

$$\int f L(g) dx = \int \langle \nabla f, \nabla g \rangle dx \quad \forall f, g$$



# Mesh Setting: Laplace Beltrami

Computing Laplace-Beltrami operator on a mesh:

Functions are real-values defined on the triangles

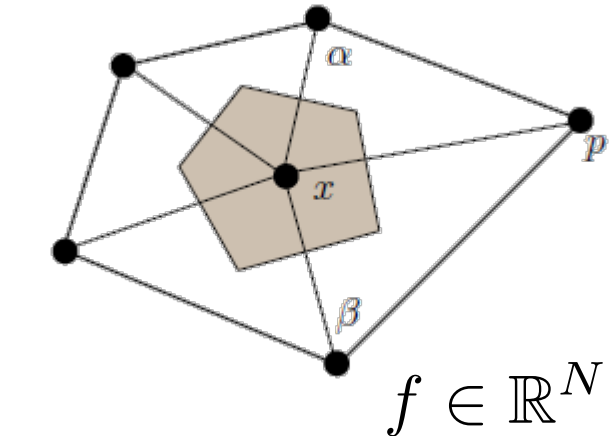
To discretize the integral, we use:

$$\int f dx = \sum_i f_i A_i$$

Here,  $A_i$  is the local area element associated with vertex  $i$ .

The simplest choice:

$$A_i = \frac{1}{3} \sum_{t \sim i} A(t)$$



Sum of the areas of adjacent triangles.

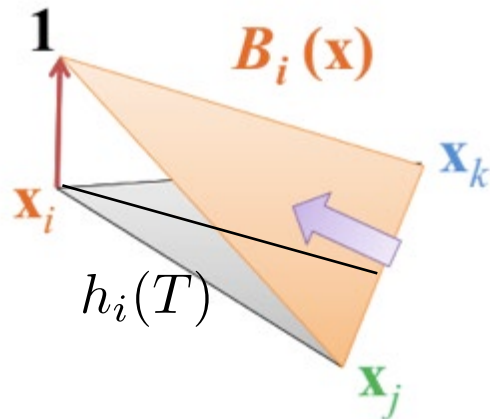
# Mesh Setting: Laplace Beltrami

## Computing the gradient of a function

Inside a single triangle, use piecewise-linear interpolation:

$$f(x) = f_i B_i(x) + f_j B_j(x) + f_k B_k(x)$$

$$\nabla f(x) = f_i \nabla B_i(x) + f_j \nabla B_j(x) + f_k \nabla B_k(x)$$



Steepest ascent direction perpendicular to opposite edge

$$\nabla B_i(x) = \frac{1}{h_i(T)}$$

Gradient is constant on a triangle.

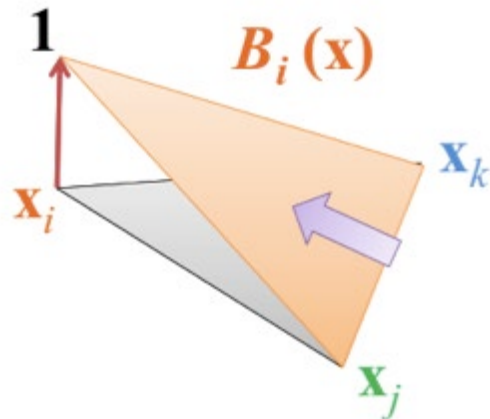
# Mesh Setting: Laplace Beltrami

## Computing the gradient of a function

Inside a single triangle, use piecewise-linear interpolation:

$$f(x) = f_i B_i(x) + f_j B_j(x) + f_k B_k(x)$$

$$\nabla f(x) = f_i \nabla B_i(x) + f_j \nabla B_j(x) + f_k \nabla B_k(x)$$



Steepest ascent direction perpendicular to opposite edge

$$\nabla B_i(x) = \nabla B_i = \frac{(x_k - x_j)^\perp}{2A_T}$$

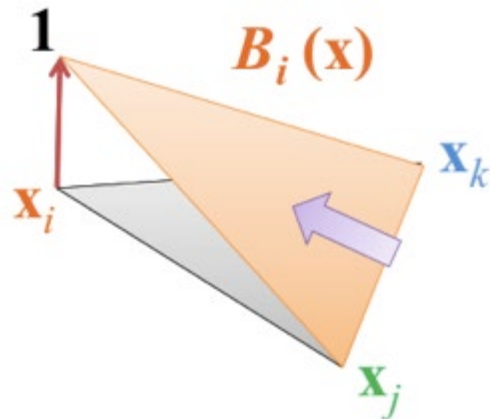
Gradient is constant on a triangle.

# Mesh Setting: Laplace Beltrami

## Computing the gradient of a function

Inside a single triangle, use piecewise-linear interpolation:

$$\begin{aligned}\nabla f(x) &= f_i \nabla B_i(x) + f_j \nabla B_j(x) + f_k \nabla B_k(x) \\ &= \frac{f_i}{2A_T} (x_k - x_j)^\perp + \frac{f_j}{2A_T} (x_i - x_k)^\perp + \frac{f_k}{2A_T} (x_j - x_i)^\perp\end{aligned}$$



Steepest ascent direction perpendicular to opposite edge

$$\nabla B_i(x) = \nabla B_i = \frac{(x_k - x_j)^\perp}{2A_T}$$

Gradient is constant on a triangle.

# Mesh Setting: Laplace Beltrami

Back to our strategy:

$$\int f L(g) dx = \int \langle \nabla f, \nabla g \rangle dx \quad \forall f, g$$

In the discrete case:

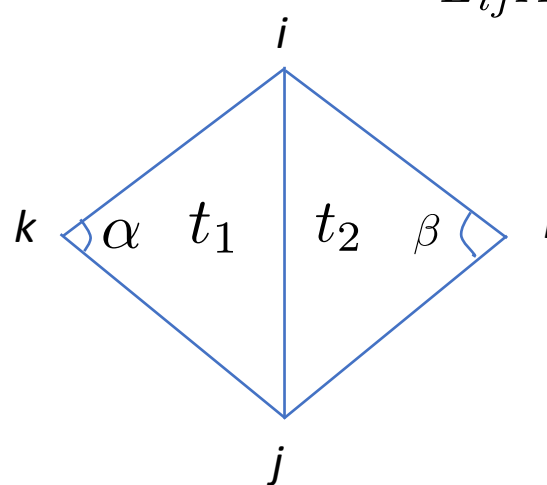
$$\sum_i f_i L(g)_i A(i) = \sum_t \langle \nabla f(t), \nabla g(t) \rangle A(t)$$

# Mesh Setting: Laplace Beltrami

In the discrete case:

$$\sum_i f_i L(g)_i A(i) = \sum_t \langle \nabla f(t), \nabla g(t) \rangle A(t)$$

Plugging in indicator functions of individual vertices we get:



The diagram shows a diamond-shaped mesh element with vertices labeled  $i$  (top),  $j$  (bottom),  $k$  (left), and  $l$  (right). A vertical line segment connects vertices  $i$  and  $j$ , dividing the diamond into two triangles,  $t_1$  (left) and  $t_2$  (right). The angle at vertex  $k$  is labeled  $\alpha$ , and the angle at vertex  $l$  is labeled  $\beta$ .

$$L_{ij}A(j) = \langle \frac{1}{2A(t_1)}(x_k - x_j)^\perp, \frac{1}{2A(t_1)}(x_i - x_k)^\perp \rangle A(t_1) \\ + \langle \frac{1}{2A(t_2)}(x_l - x_j)^\perp, \frac{1}{2A(t_2)}(x_i - x_l)^\perp \rangle A(t_2)$$

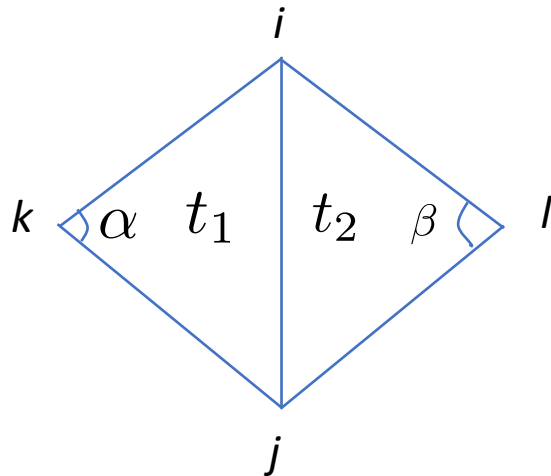
# Mesh Setting: Laplace Beltrami

In the discrete case:

$$\sum_i f_i L(g)_i A(i) = \sum_t \langle \nabla f(t), \nabla g(t) \rangle A(t)$$

Plugging in indicator functions of individual vertices we get:

$$= \frac{1}{4A(t_1)} \cos(\alpha) \|e_{ki}\| \|e_{kj}\| + \frac{1}{4A(t_2)} \cos(\beta) \|e_{li}\| \|e_{lj}\|$$
$$A(t_1) = \frac{1}{2} \sin(\alpha) \|e_{ki}\| \|e_{kj}\|$$



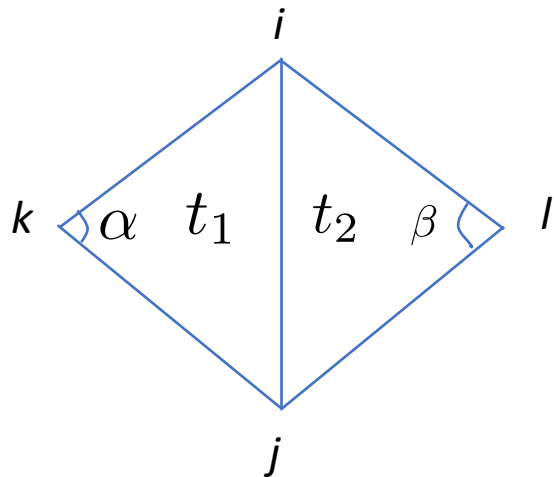
$$L_{ij} A(j) = \frac{1}{2} \cot(\alpha) + \frac{1}{2} \cot(\beta)$$

# Mesh Setting: Laplace Beltrami

In the discrete case:

$$L_{ij}A(j) = \frac{1}{2} \cot(\alpha) + \frac{1}{2} \cot(\beta)$$

In matrix notation:  $L = A^{-1}W$



$$W_{ij} = \begin{cases} -\frac{1}{2} (\cot(\alpha) + \cot(\beta)) & \text{if } i \sim j \\ -\sum_j W_{ij} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

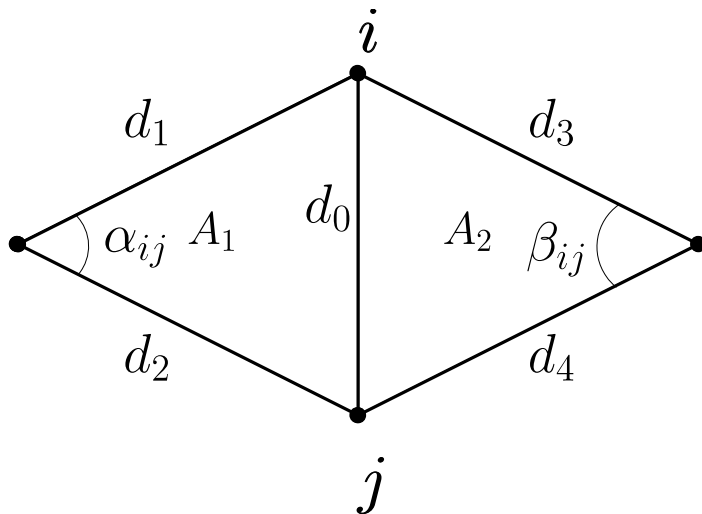
$$A_{ij} = \begin{cases} A(j) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

# Mesh Setting: Laplace Beltrami

In the discrete case:

$$L_{ij}A(j) = \frac{1}{2} \cot(\alpha) + \frac{1}{2} \cot(\beta)$$

In matrix notation:  $L = A^{-1}W$



Small computational trick:

$$\begin{aligned} -W_{ij} &= \frac{1}{2} \cot(\alpha_{ij}) + \frac{1}{2} \cot(\beta_{ij}) \\ &= \frac{1}{8A_1} (d_0^2 - d_1^2 - d_2^2) \\ &\quad + \frac{1}{8A_2} (d_0^2 - d_3^2 - d_4^2) \end{aligned}$$

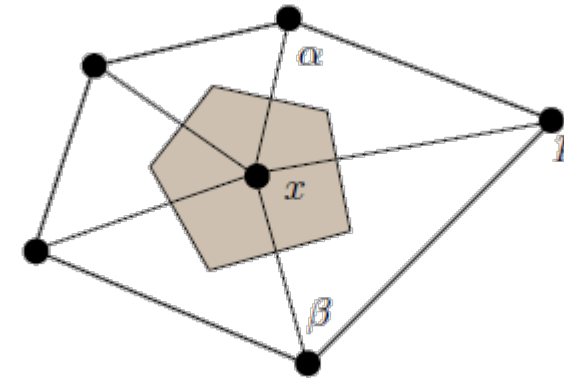
# Mesh Setting: Laplace Beltrami

**Computing Laplace-Beltrami from a mesh:**

Classical discretization

$$L = A^{-1}W$$

Very simple to compute, produces sparse matrices.



Using this discretization, for any two real-valued functions  $f, g$  we obtain

$$\int f L(g) dx = \int \langle \nabla f, \nabla g \rangle dx \quad \forall f, g$$

# Mesh Setting: Laplace Beltrami

## Computing Laplace-Beltrami from a mesh:

Classical discretization

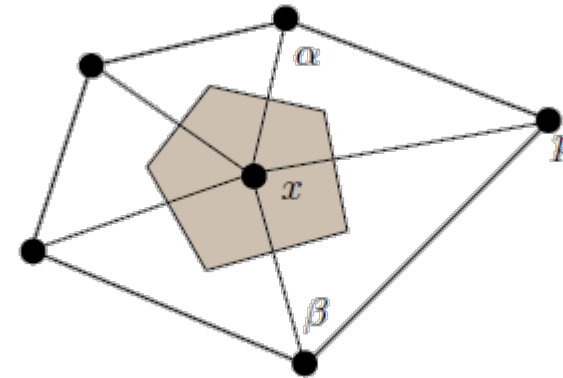
$$L = A^{-1}W$$

Very simple to compute, produces sparse matrices.

To find eigenvalues, solve *generalized* eigenproblem:

$$W\phi_i = \lambda_i A\phi_i$$

Guarantee real solution since W is symmetric and A is SPD.



# Mesh Setting: Laplace Beltrami

## Good news:

Cotangent Laplacian converges weakly under mesh refinement for “nice enough” meshes

M. Wardetzky. Convergence of the cotangent formula: An overview. 2008

## Bad news: No free lunch

**There is no discrete LB operator that is**

- Symmetric
- Local
- Has Linear Precision
- Always has positive weights.

M. Wardetzky. Discrete Laplace operators: No free lunch, 2007

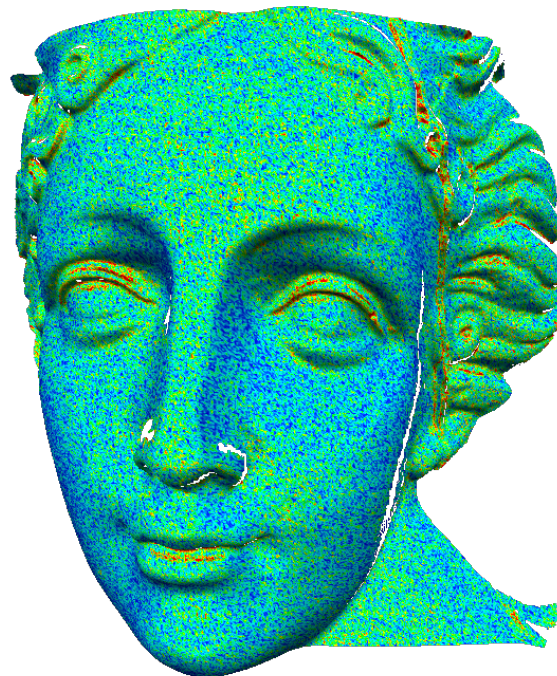
# Laplace-Beltrami – Some Applications

**Input:** Noisy mesh (scanned or other)

**Output:** Smooth mesh

**How:** Filter out high frequency noise

Mesh Smoothing



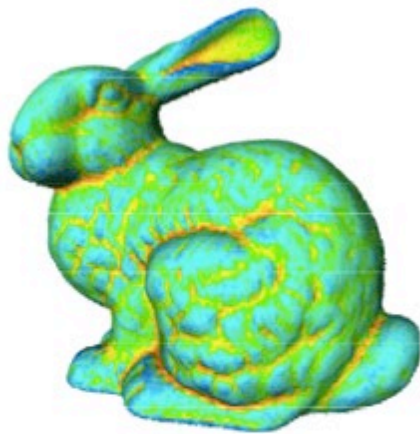
# Laplace-Beltrami – Some Applications

## Laplacian Smoothing:

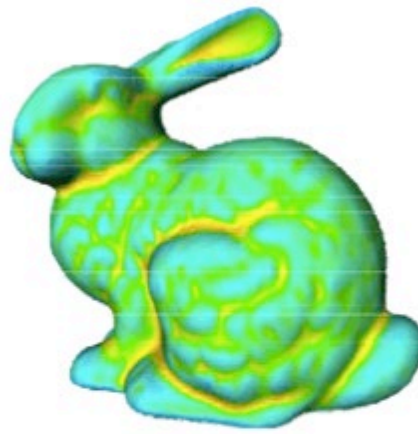
Given the matrix of 3D coordinates, apply a few iterations of

$$\mathbf{P}^{(t)} = \mathbf{P}^{(t-1)} - \lambda L \mathbf{P}^{(t-1)}$$

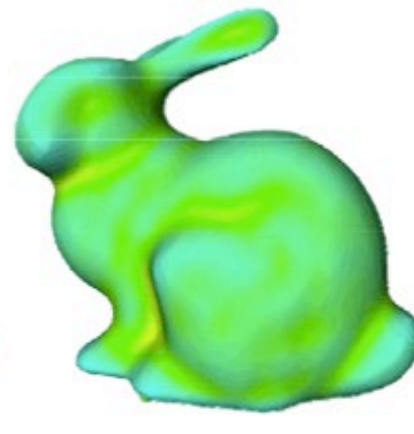
Move each vertex towards the mean of its neighbors



0 Iterations



5 Iterations



20 Iterations

# Spectral Analysis

Then:  $\mathbf{P}^{(t+1)} = \mathbf{P}^{(t)} - \lambda \mathbf{L} \mathbf{P}^{(t)} = (\mathbf{I} - \lambda \mathbf{L}) \mathbf{P}^{(t)}$

After  $m$  iterations:  $\mathbf{P}^{(m)} = (\mathbf{I} - \lambda \mathbf{L})^m \mathbf{P}^{(0)}$

Can be described using eigen-decomposition of  $\mathbf{L}$

$$\mathbf{L} = \mathbf{V} \mathbf{D} \mathbf{V}^T$$

$\mathbf{V} = \begin{pmatrix} | & | & \dots & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \\ | & | & \dots & | \end{pmatrix}, \mathbf{D} = \begin{pmatrix} k_1 & & & \\ & k_2 & & \\ & & \dots & \\ & & & k_n \end{pmatrix}$

$\mathbf{P}^{(m)} = \mathbf{V} (\mathbf{I} - \lambda \mathbf{D})^m \mathbf{V}^T \mathbf{P}^{(0)}$

Filtering high frequencies

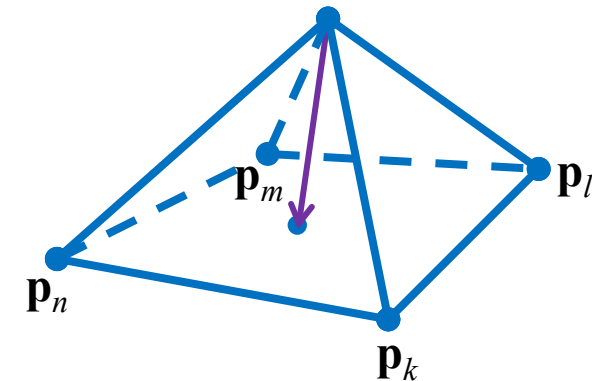
# Laplacian Smoothing on Meshes

What is  $\Delta \mathbf{p}_i$  ?

$$\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} - \lambda \Delta \mathbf{p}_i^{(t)}$$



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



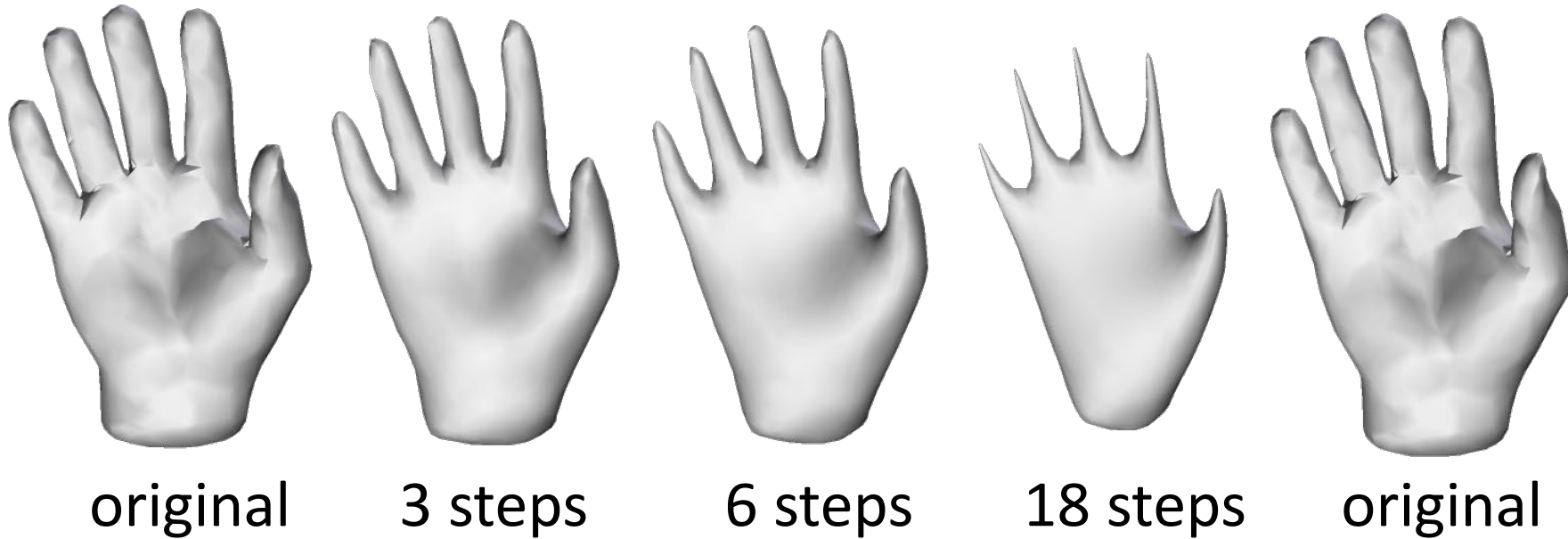
$$N_i = \{k, l, m, n\}$$

$$\mathbf{p}_i = (x_i, y_i, z_i)$$

$$\frac{1}{|N_i|} \left( \sum_{j \in N_i} \mathbf{p}_j \right) - \mathbf{p}_i$$

# Problem - Shrinkage

Repeated iterations of Laplacian smoothing shrinks the mesh

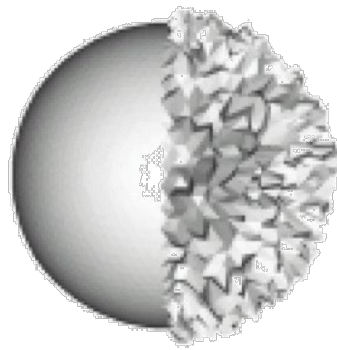


# Taubin Smoothing

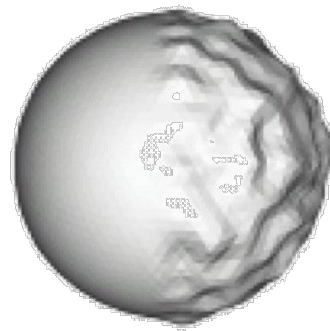
Iterate:  $\mathbf{p}_i \leftarrow \mathbf{p}_i - \lambda \Delta \mathbf{p}_i$  Shrink

$\mathbf{p}_i \leftarrow \mathbf{p}_i - \mu \Delta \mathbf{p}_i$  Inflate

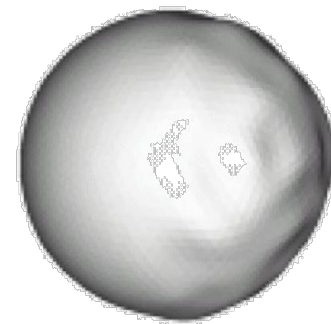
with  $\lambda > 0$  and  $\mu < 0$



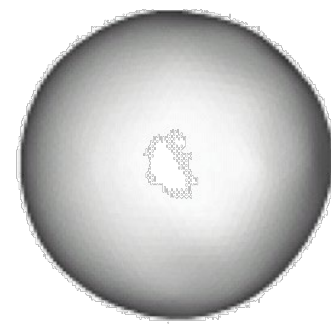
original



10 steps



50 steps



200 steps

# Laplacian Smoothing

$$\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} - \lambda \Delta \mathbf{p}_i^{(t)}$$

$\Delta \mathbf{p}_i$  = mean curvature normal

mean curvature flow



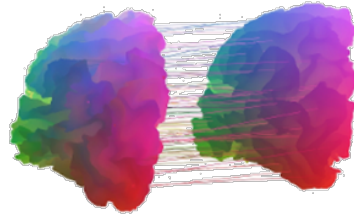
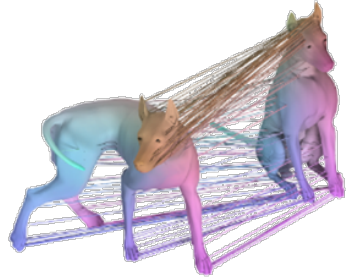
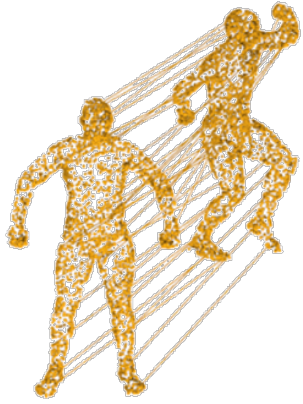
$$\Delta_{\mathcal{M}} \mathbf{p} = \operatorname{div}_{\mathcal{M}} \nabla_{\mathcal{M}} \mathbf{p} = -2H \mathbf{n} \in \mathbb{R}^3$$

# Conclusion

- Spectral Methods in Shape Analysis
  - Mesh Laplacians -- Laplace-Beltrami operator and its properties
  - Isometry invariance
  - Relatively easy to compute
  - Some applications
- Key message:
  - Laplacian matrices allow us to organize shape information in a multi-scale, easy to manipulate way.

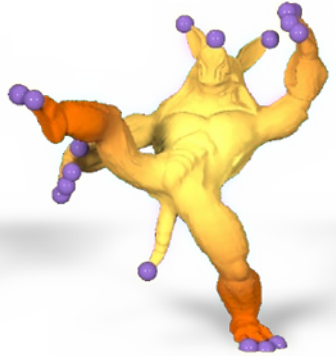
# Shape Alignments and Correspondences

# Alignment and Correspondences, Shape Features



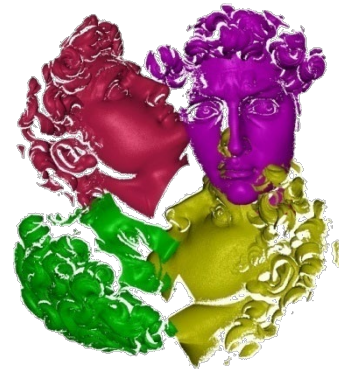
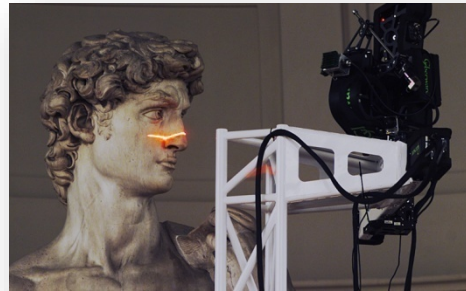
A

B



# Aligning and Registering (Partial or Entire) Shapes

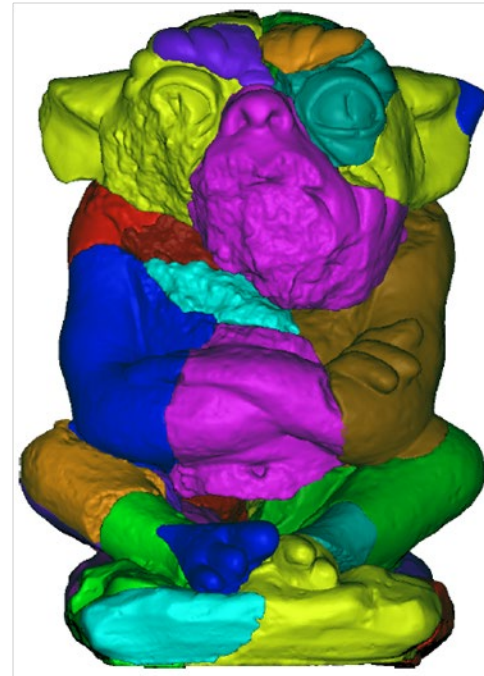
- The need to align and match shapes arises in many domains
- A classic example is shape acquisition through a 3D scanner



# Simultaneous Localization and Mapping (SLAM)



# Cultural Artifact Reconstruction



# Protein Structure Alignment



Human (red) and fly (yellow) thioredoxins, compared

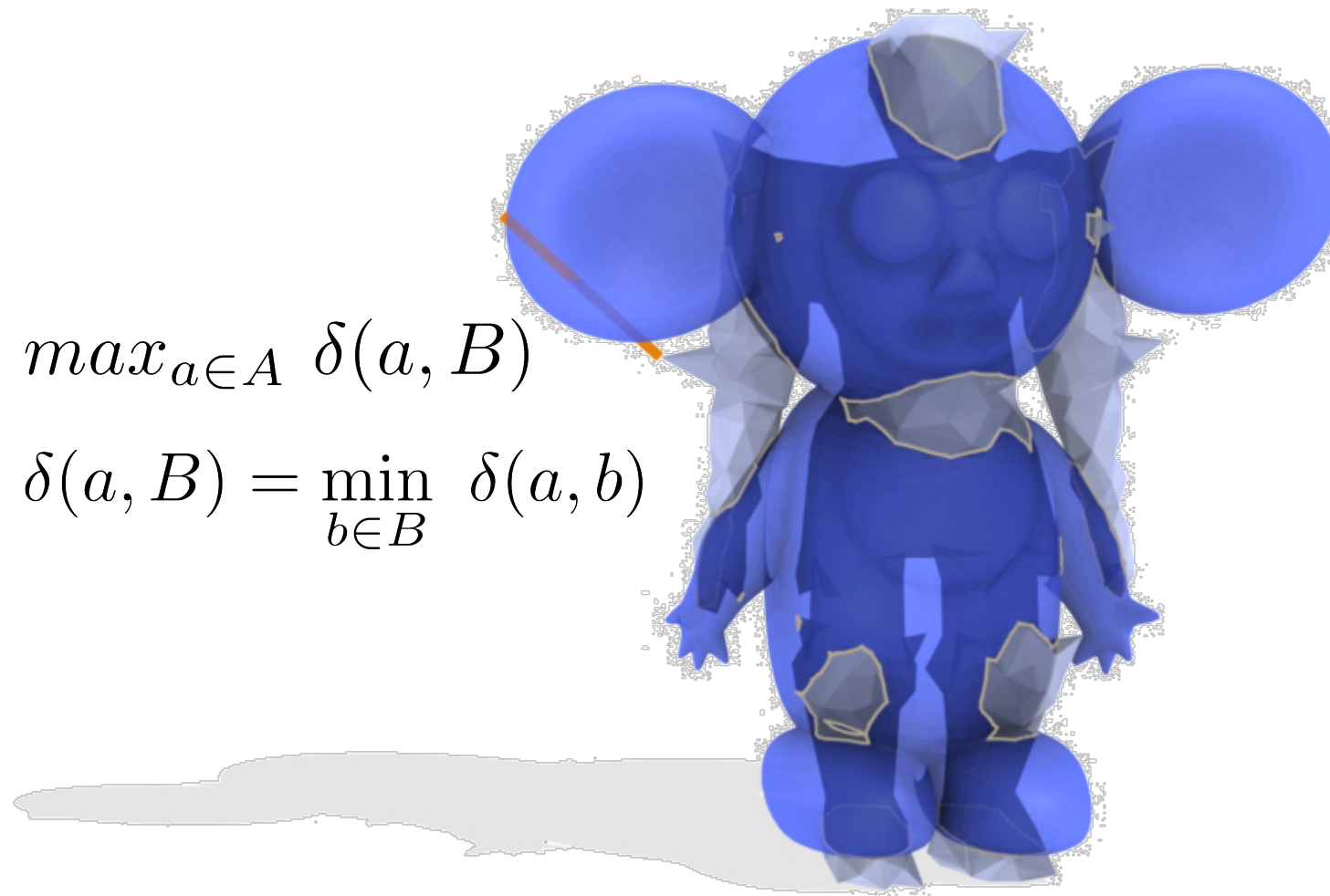
# Measuring Success: Shape Distances

Given two shapes  $A$  and  $B$ , we are interested in defining a distance or (dis-)similarity measure

$$\min_T \delta(A, T(B)) \quad \text{[extrinsic]}$$

Such measures are crucial in shape similarity search, shape classification, and in general for defining ML loss functions.

# Hausdorff Distance



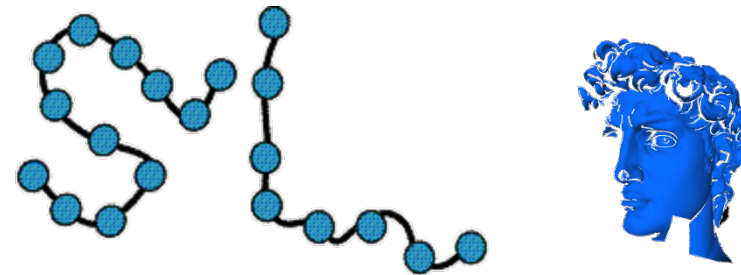
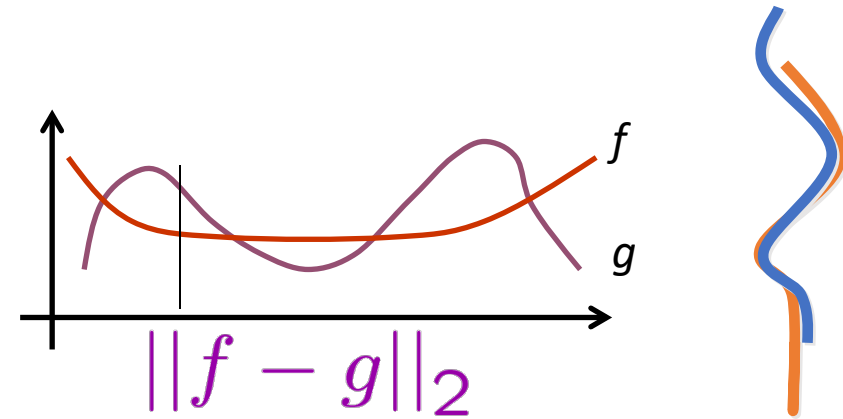
$$\max_{a \in A} \delta(a, B)$$

$$\delta(a, B) = \min_{b \in B} \delta(a, b)$$

Hausdorff is a worst-case error – in some cases we prefer MSE

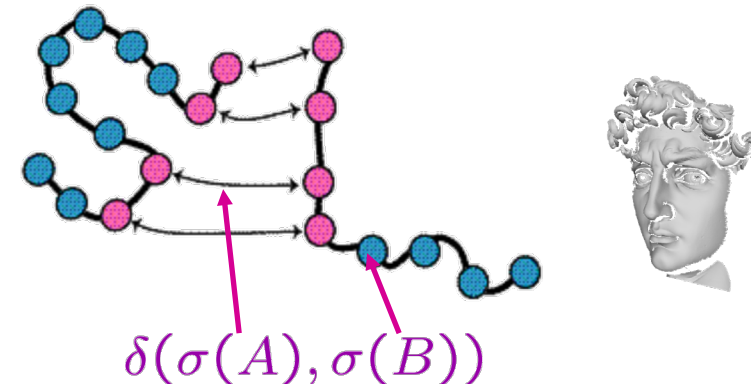
# Issues about Distance Metrics

- We are all familiar with function norms ( $L_2$ , etc.). The **common parametrization** establishes **correspondences**. We don't have that for structures or shapes.
- Partial matches need to be considered -- notion of **support**  $\sigma$  for the match.
- What group of **aligning transforms** is to be considered?
- Is the resulting distance a **metric**?



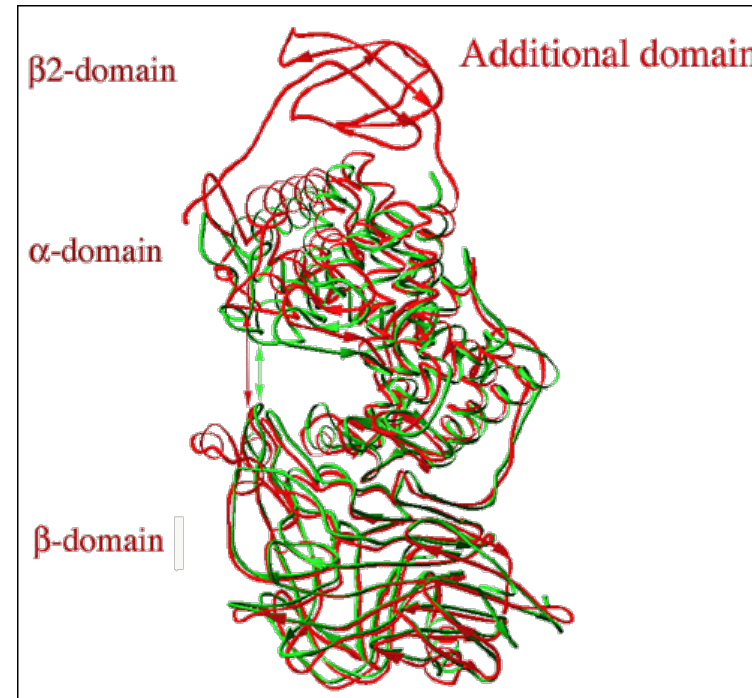
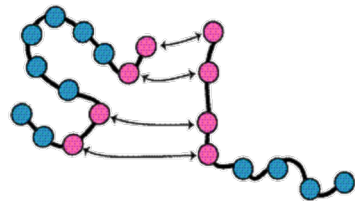
$$\delta(A, C) \leq \delta(A, B) + \delta(B, C)$$

Not for partial matches



# Simultaneous Estimation

- We are given two shapes  $A$  and  $B$ , each in its own coordinate system
- We must establish **correspondences** between certain parts (the **alignment supports**) of  $A$  and  $B$
- We must find an optimal **transform** that best **aligns** the supports of  $A$  and  $B$
- We must **score** this choice of supports and transform to produce a **distance measure**  $\delta$



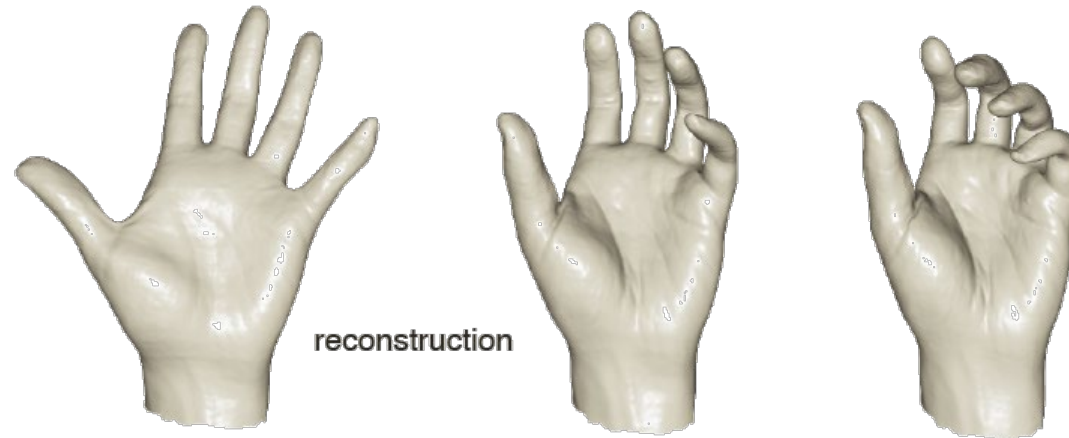
In computing the score,

1. what distance metric do we use?
2. how do we aggregate distances?
3. how do we trade-off larger supports for larger aggregate distance?

# Degrees of Freedom

- Transform estimation

- A rigid motion has 6 degrees of freedom (3 for translation and 3 for rotation)
- We typically estimate the motion using many more pairs of corresponding points, so the problem is **overdetermined** (which is good, given noise, outliers, etc – use least squares approaches)
- More general transforms require more degrees of freedom. When shape deformations are allowed, the degrees of freedom can grow very rapidly



# A Double Whammy

- Estimate correspondences
- Estimate the aligning transform
- Gives rise to combinatorial searches
- Transforms can be non-linear

Hard optimization problems!

Good features help

Low-dimensionality of some transforms helps



The End