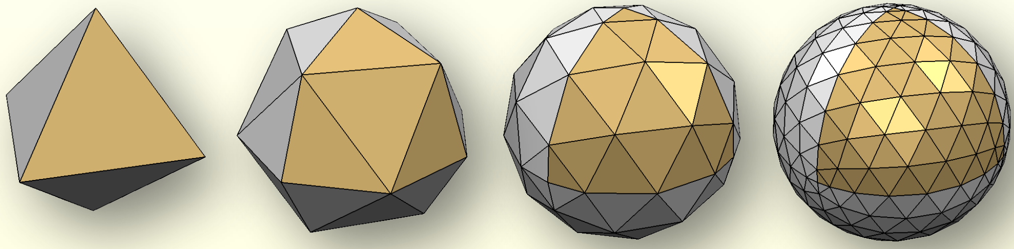
CS348a:
Geometry Processing

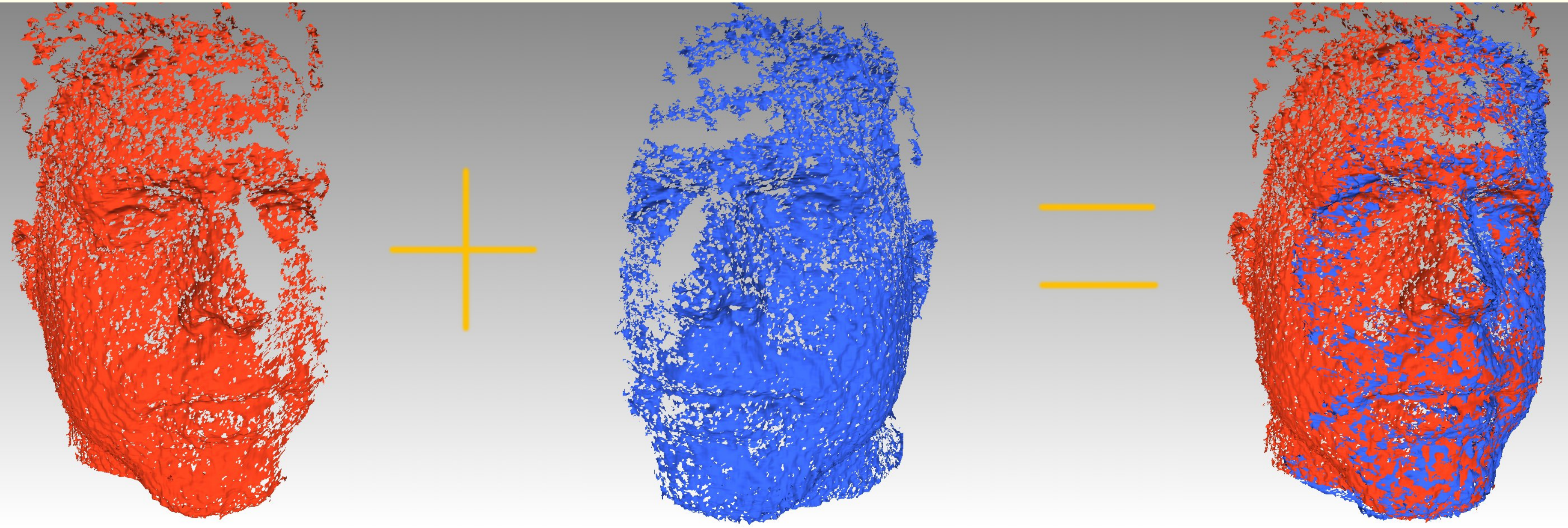


Reconstruction / Fairing
(also: Laplace-Beltrami)

March 13, 2017 Olga Diamanti

In Previous Lecture

- Point cloud registration



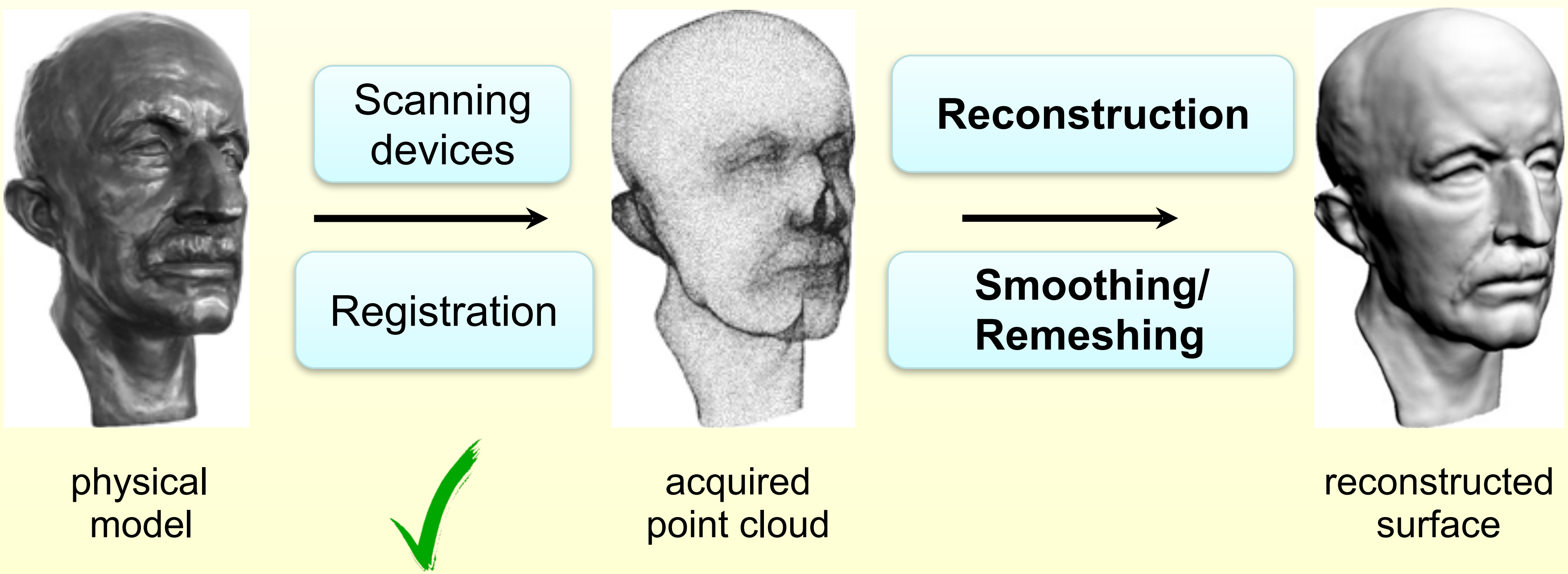
This Lecture

- Point clouds not directly usable by most CG applications
  - Rendering, editing/deformation, texturing, simulation, …!
- Need a semi-”continuous” surface instead!

Partial slides credits: Olga Sorkine-Hornung, ETH Zurich, Mario Botsch, and many others

This Lecture

- Point clouds not directly usable by most CG applications
  - Rendering, editing/deformation, texturing, simulation, …!
- Need a semi-”continuous” surface instead!



Partial slides credits: Olga Sorkine-Hornung, ETH Zurich, Mario Botsch, and many others

Input to Reconstruction Process

- Input option 1: just a set of 3D points, irregularly spaced
  - Need to estimate normals

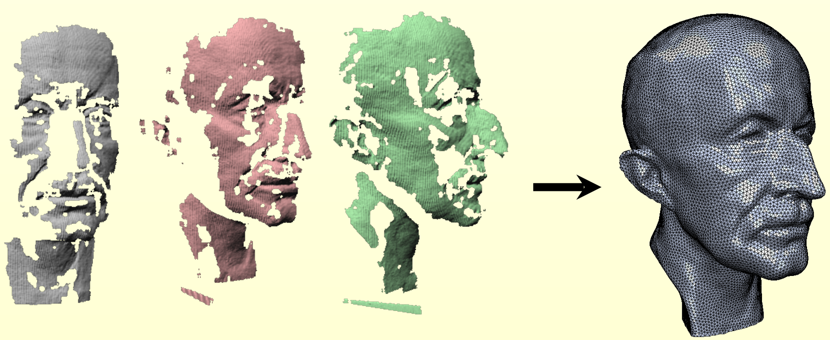
    ➔ reminder: PCA - intro class!

    ➔ Hoppe et al. 92, “Surface reconstruction from unorganized points”

Input to Reconstruction Process

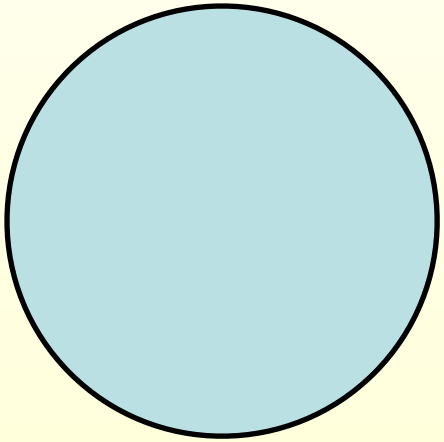
- Input option 1: just a set of 3D points, irregularly spaced
  - Need to estimate normals
    ➔ reminder: PCA - intro class!
    ➔ Hoppe et al. 92, “Surface reconstruction from unorganized points”

- Input option 2: normals come from the range scans



set of raw scans → reconstructed model

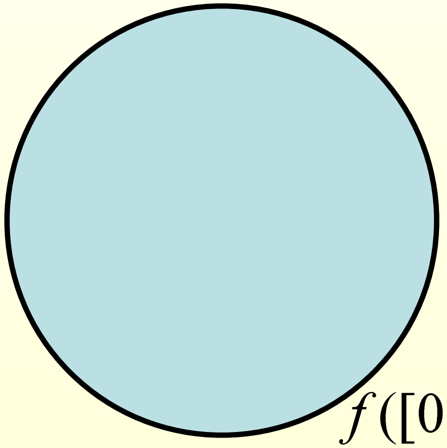
What is a surface?



What is a surface?

- Explicit representation
  - Image of parameterization

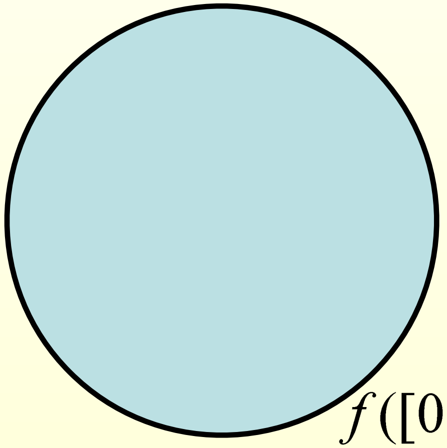
$$f(t) = (x(t), y(t)) = (r\cos(t), r\sin(t))$$



What is a surface?

- Explicit representation
  - Image of parameterization

$$f(t) = (x(t), y(t)) = (r\cos(t), r\sin(t))$$



- Implicit representation
  - Zero set of distance function

$$F(x,y) = \sqrt{x^2+y^2} - r$$

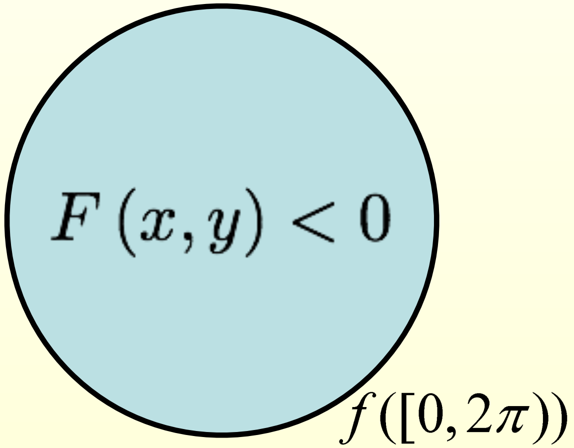
What is a surface?

- Explicit representation
  - Image of parameterization

$$f(t) = (x(t), y(t)) = (r\cos(t), r\sin(t))$$

- Implicit representation
  - Zero set of distance function

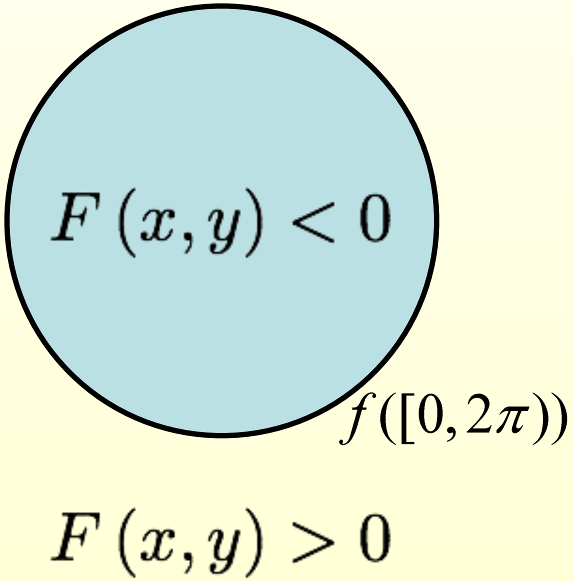
$$F(x,y) = \sqrt{x^2 + y^2} - r$$



What is a surface?

- Explicit representation
  - Image of parameterization

$$f(t) = (x(t), y(t)) = (r\cos(t), r\sin(t))$$



- Implicit representation
  - Zero set of distance function

$$F(x,y) = \sqrt{x^2 + y^2} - r$$

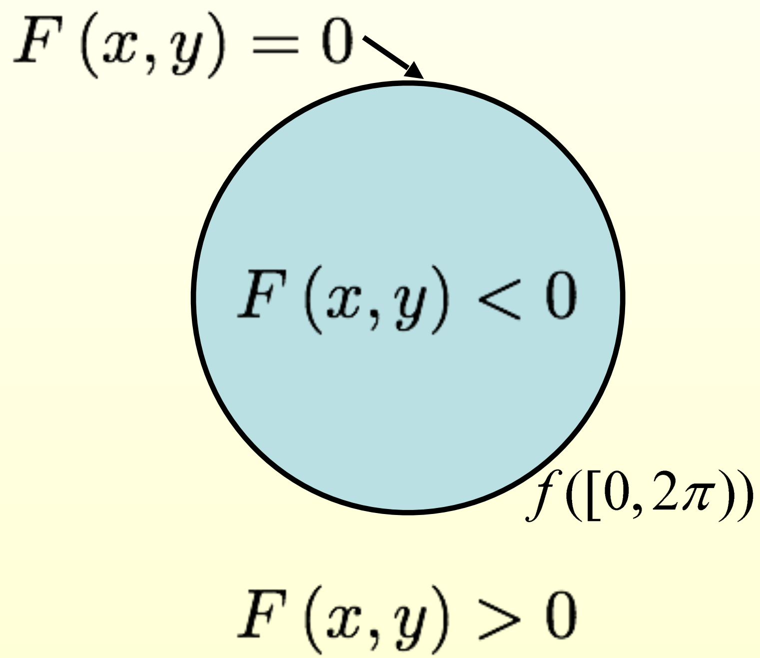
What is a surface?

- Explicit representation
  - Image of parameterization

$$f(t) = (x(t), y(t)) = (r\cos(t), r\sin(t))$$

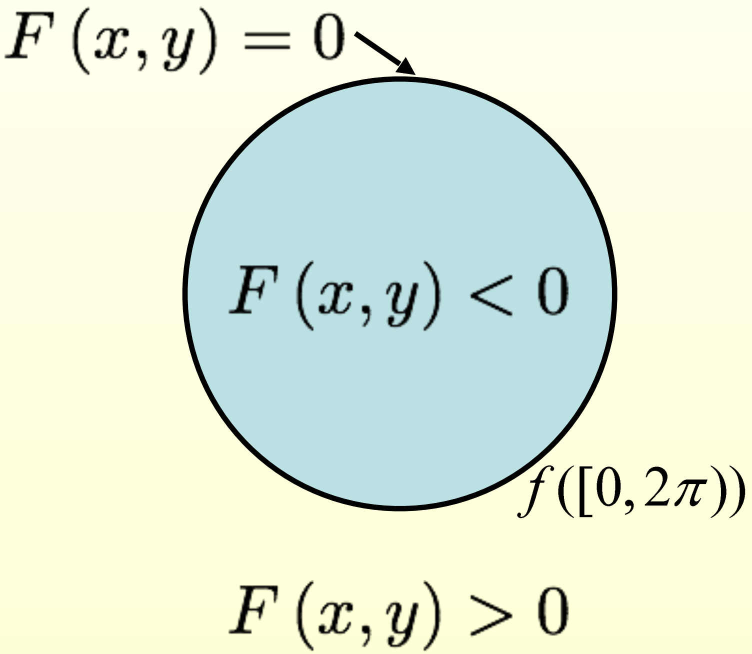
- Implicit representation
  - Zero set of distance function

$$F(x,y) = \sqrt{x^2 + y^2} - r$$



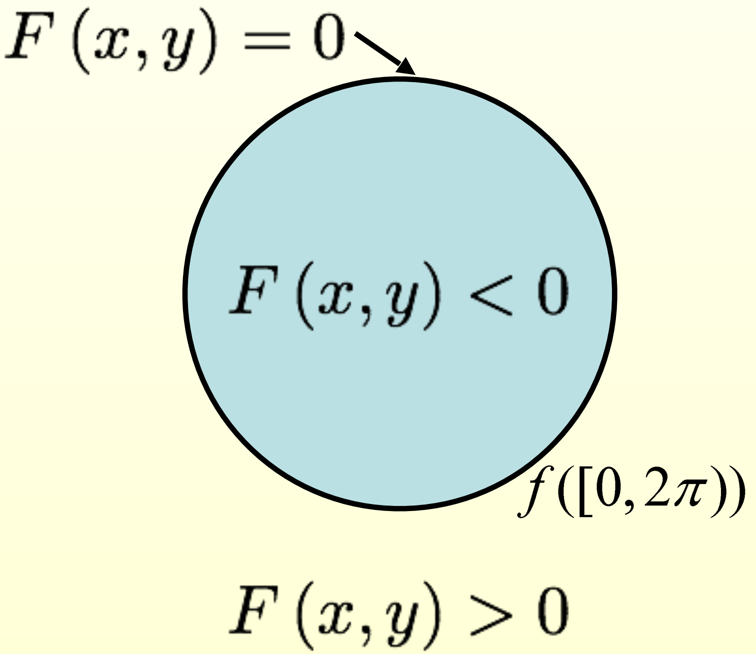
Explicit / Implicit

- Explicit representation
  - Image of parameterization
- Implicit representation
  - Zero set of distance function



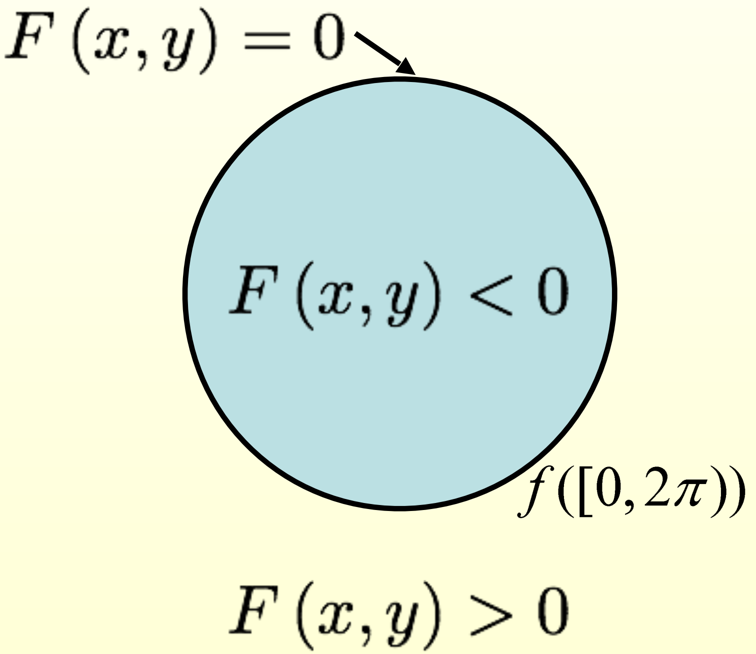
Explicit / Implicit

- Explicit representation
  - Image of parameterization
  - Easy to find points on shape
- Implicit representation
  - Zero set of distance function



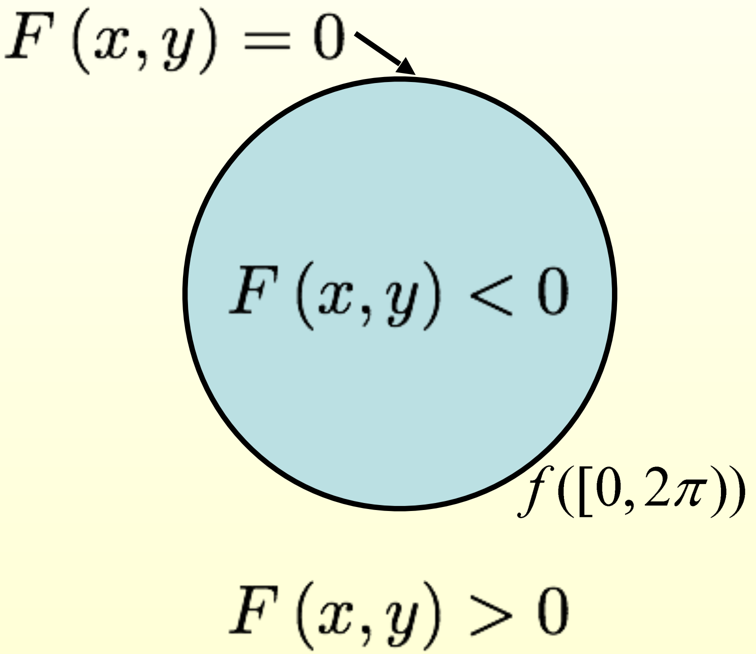
Explicit / Implicit

- Explicit representation
  - Image of parameterization
  - Easy to find points on shape
  - Can defer problems to param domain
- Implicit representation
  - Zero set of distance function



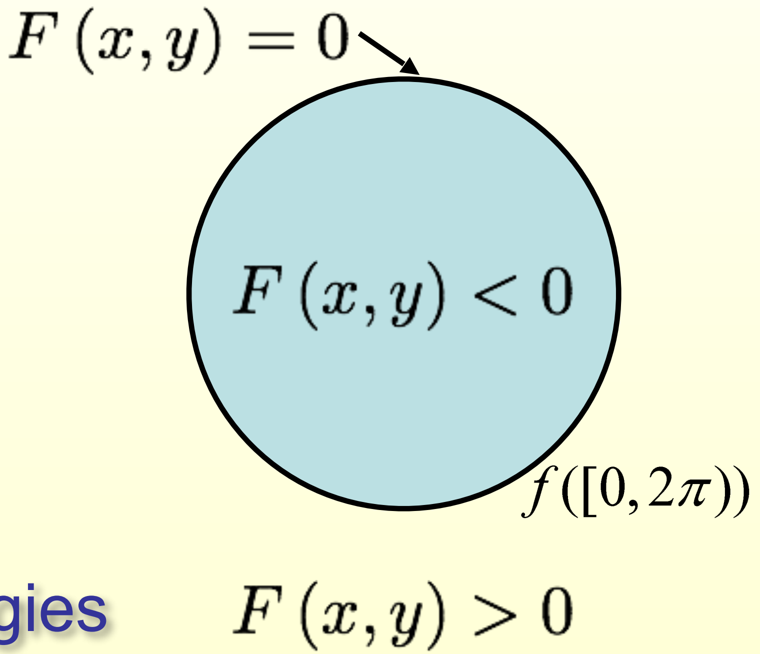
Explicit / Implicit

- Explicit representation
  - Image of parameterization
  - Easy to find points on shape
  - Can defer problems to param domain
- Implicit representation
  - Zero set of distance function
  - Easy in/out/distance test



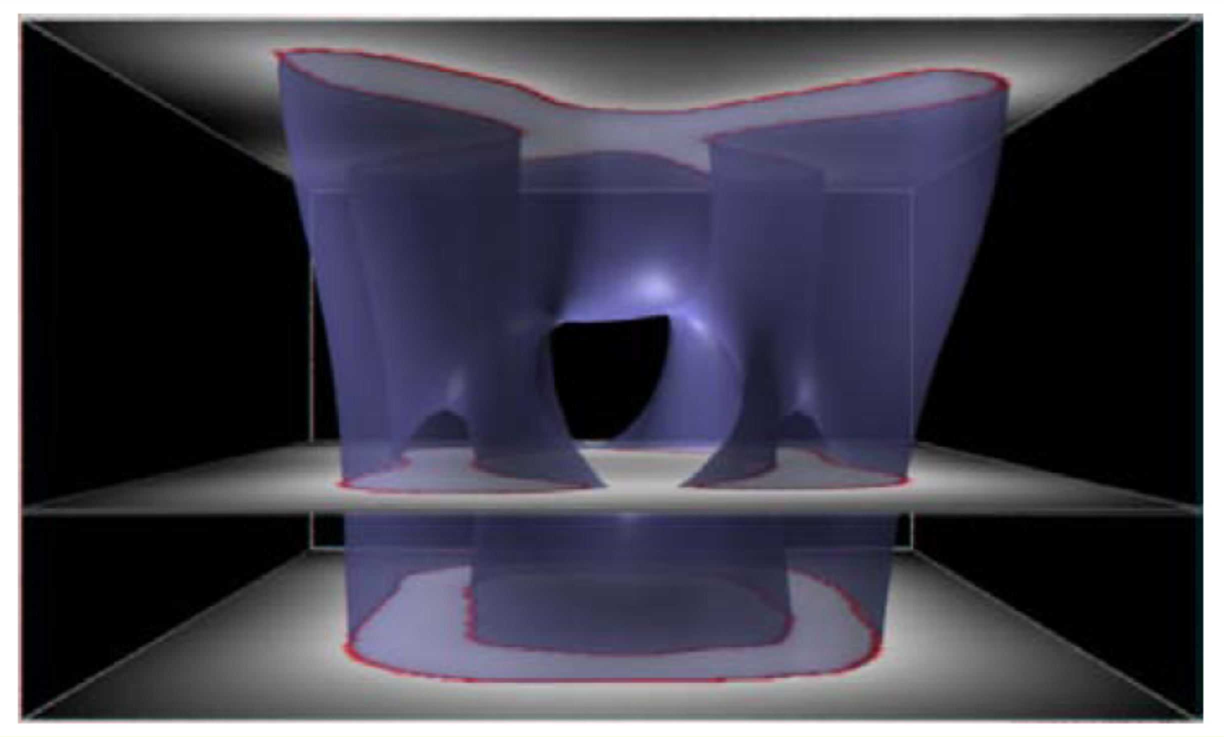
Explicit / Implicit

- Explicit representation
  - Image of parameterization
  - Easy to find points on shape
  - Can defer problems to param domain
- Implicit representation
  - Zero set of distance function
  - Easy in/out/distance test
  - Easy to handle different topologies



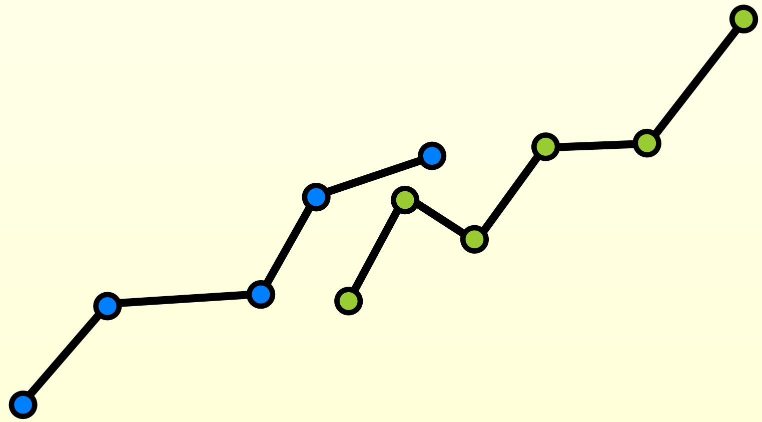
Implicit Representations

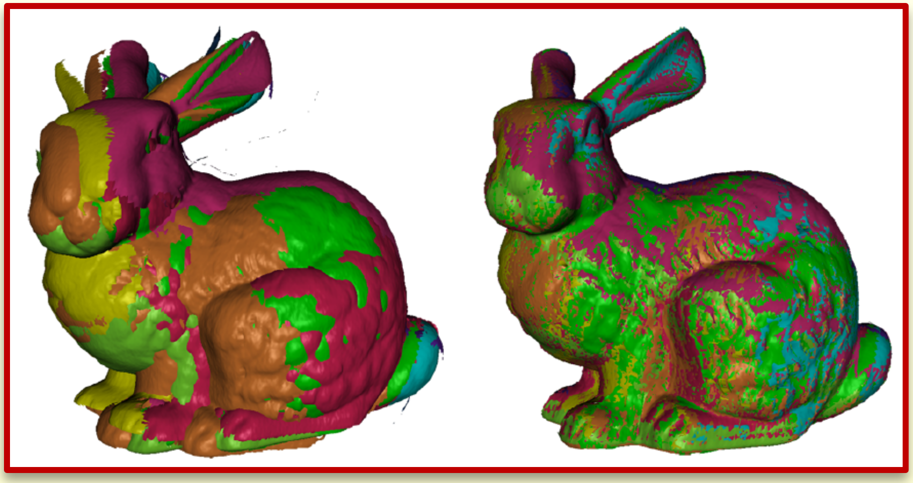
- Easy to handle different topologies



How to Connect the Dots?

- **Explicit reconstruction**:
stitch the range scans together

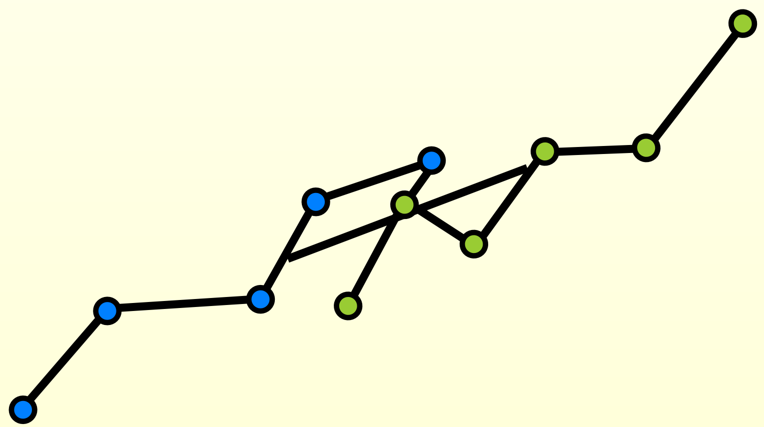




"Zippered Polygon Meshes from Range Images", Greg Turk and Marc Levoy, ACM SIGGRAPH 1994

How to Connect the Dots?

- **Explicit reconstruction**:
  stitch the range scans together



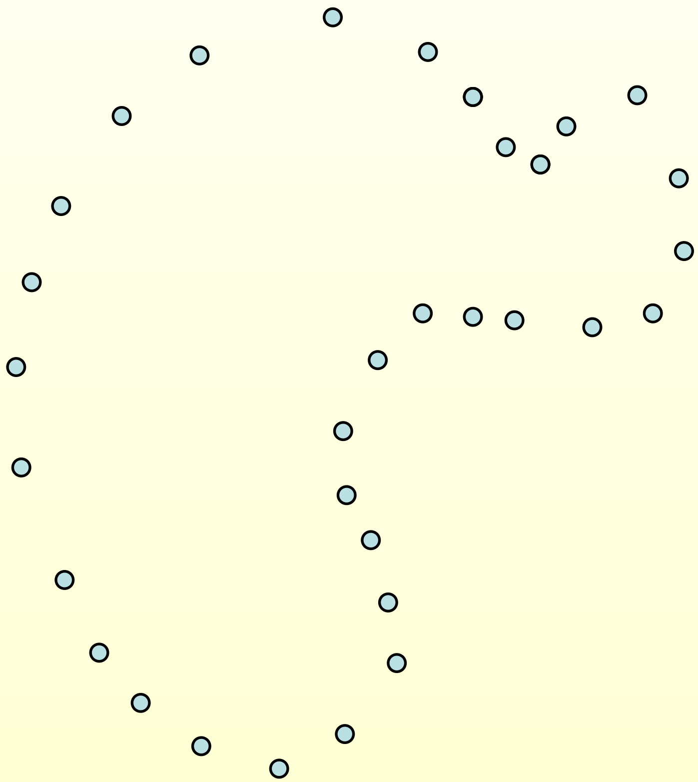
- Connect sample points by triangles
- Exact interpolation of sample points
- Bad for noisy or misaligned data
- Can lead to holes or non-manifold situations

Implicit Function Approach

- Define a function

$$f : R^3 \to R$$

with value < 0 outside the shape and > 0 inside

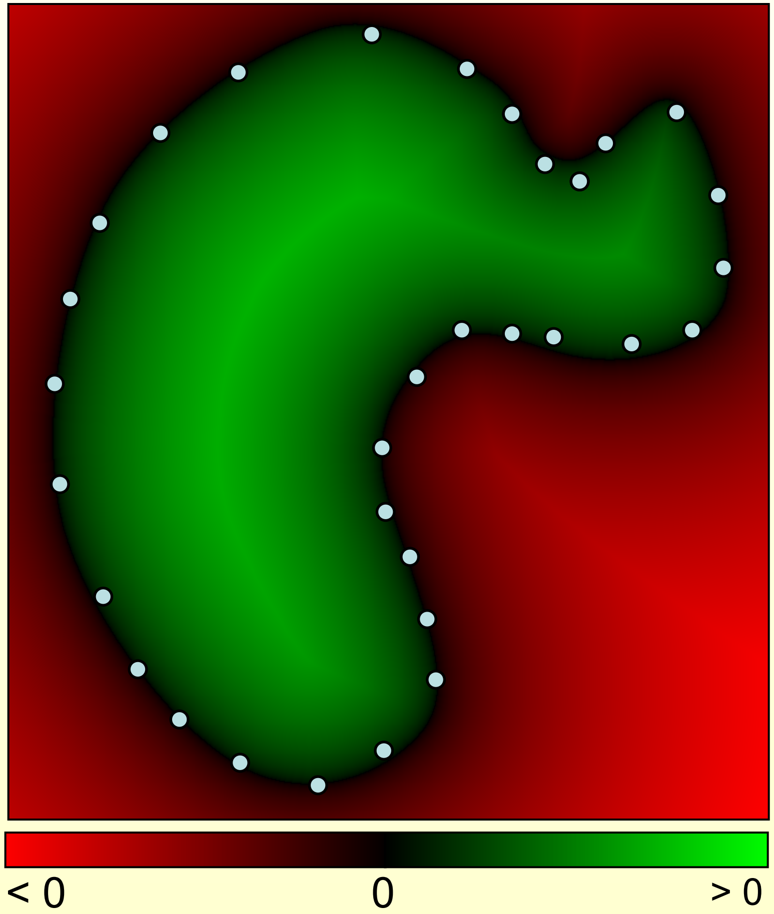


Implicit Function Approach

- Define a function

$$f : R^3 \rightarrow R$$

with value < 0 outside the shape and > 0 inside



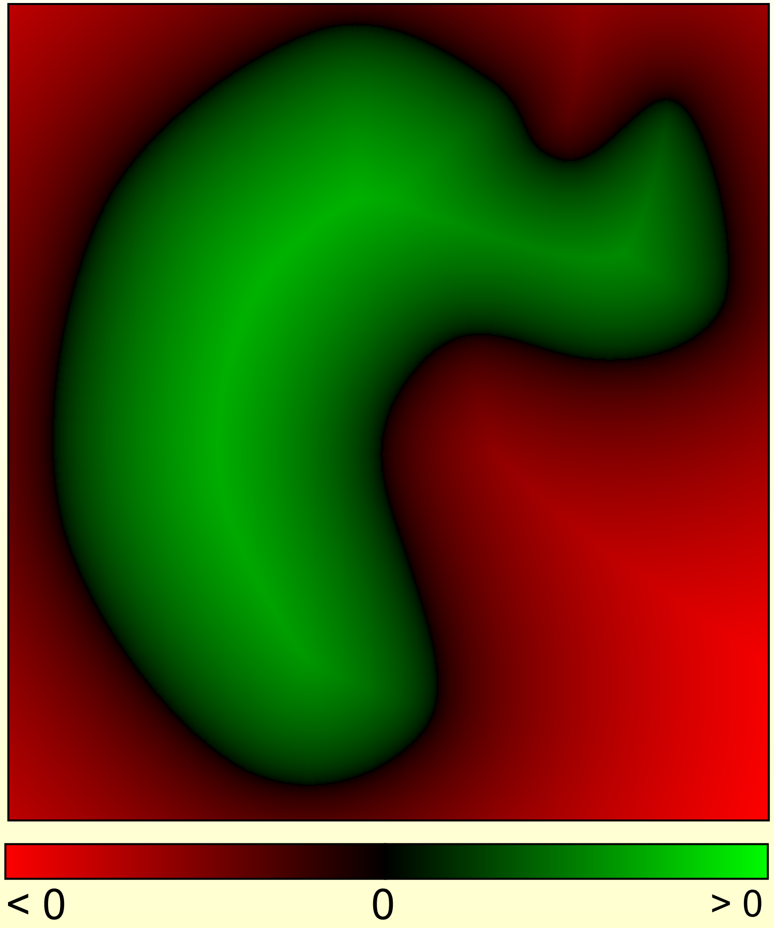
Implicit Function Approach

- Define a function

$$f : R^3 \rightarrow R$$

with value < 0 outside the shape and > 0 inside

- Extract the zero-set



Implicit Function Approach

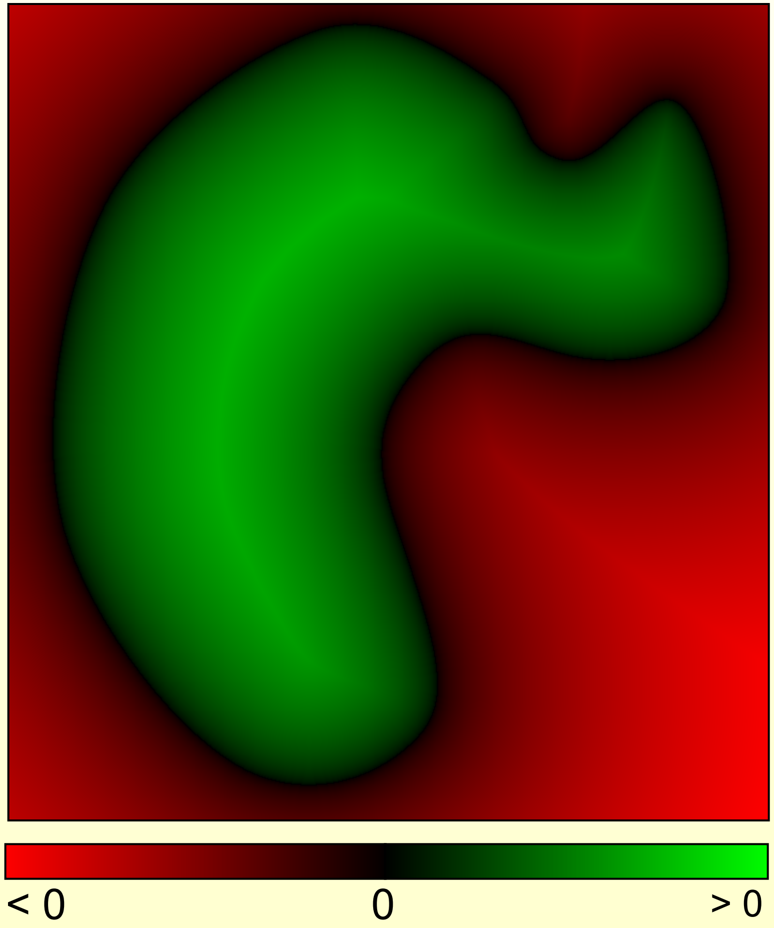
- Define a function

$$f : R^3 \to R$$

with value < 0 outside the shape and > 0 inside

- Extract the zero-set

$$\{x : f(x) = 0\}$$



Implicit Function Approach

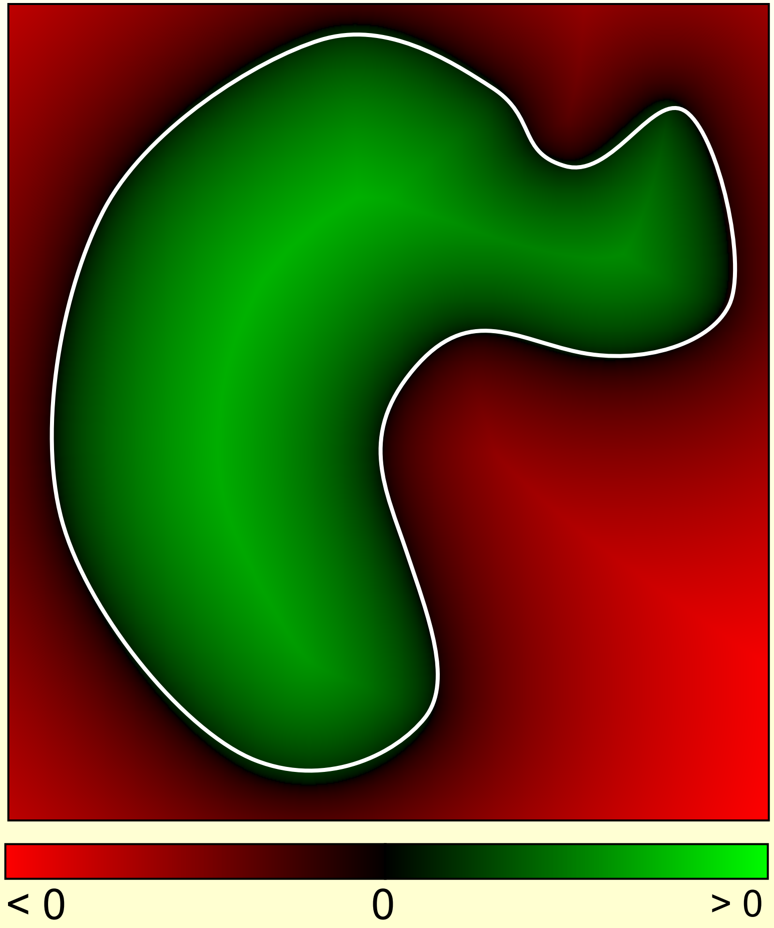
- Define a function

  $$f : R^3 \rightarrow R$$

  with value < 0 outside the shape and > 0 inside

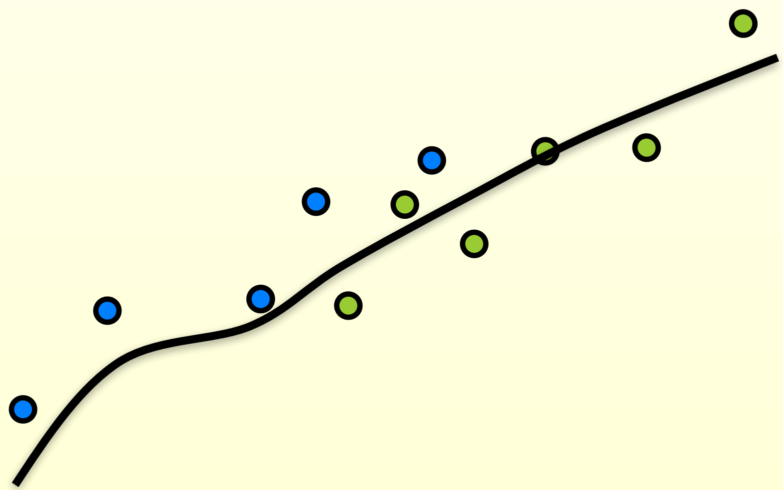
- Extract the zero-set

  $$\{x : f(x) = 0\}$$



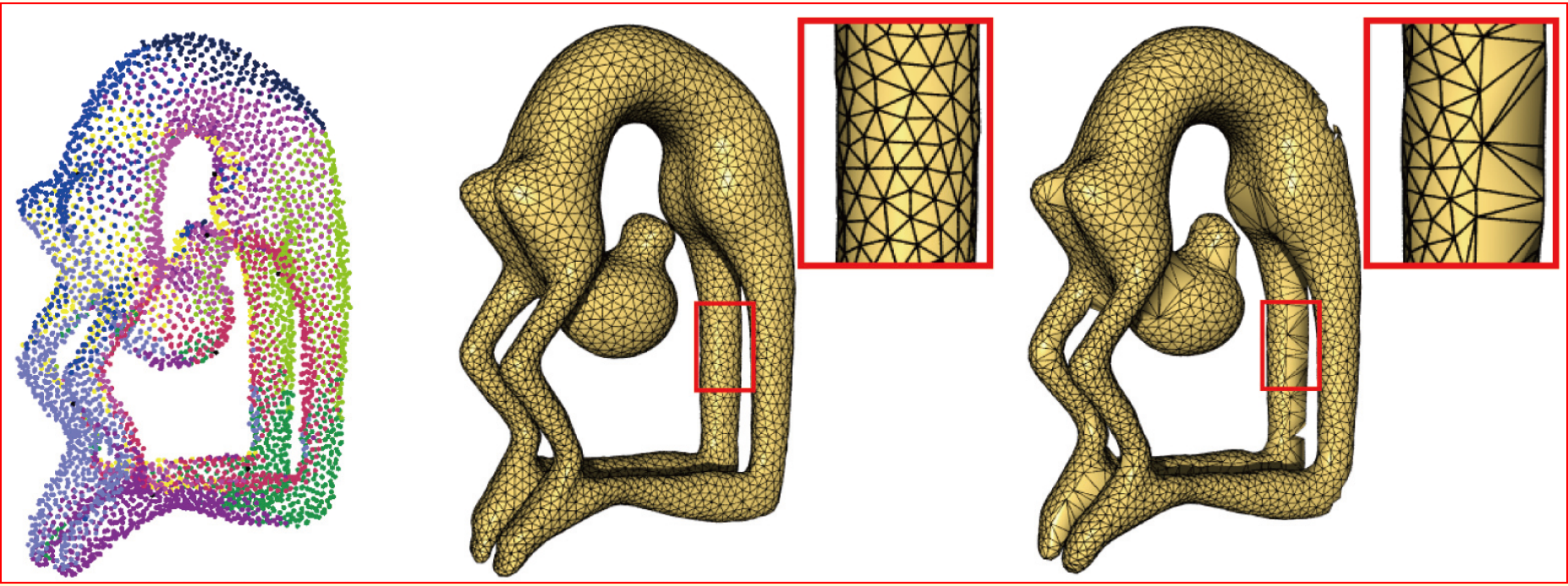
How to Connect the Dots?

- **Implicit reconstruction**: estimate a signed distance function (SDF); extract 0-level set mesh using Marching Cubes



  - Approximation of input points
  - Watertight manifold results by construction

Implicit vs. Explicit

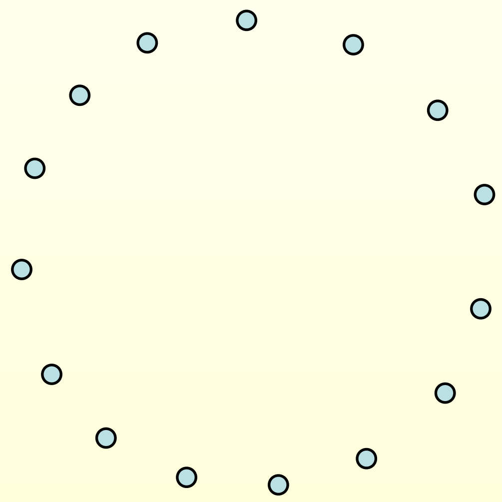


Input Implicit Explicit

SDF from Points and Normals

- Compute signed distance to the tangent plane of the closest point

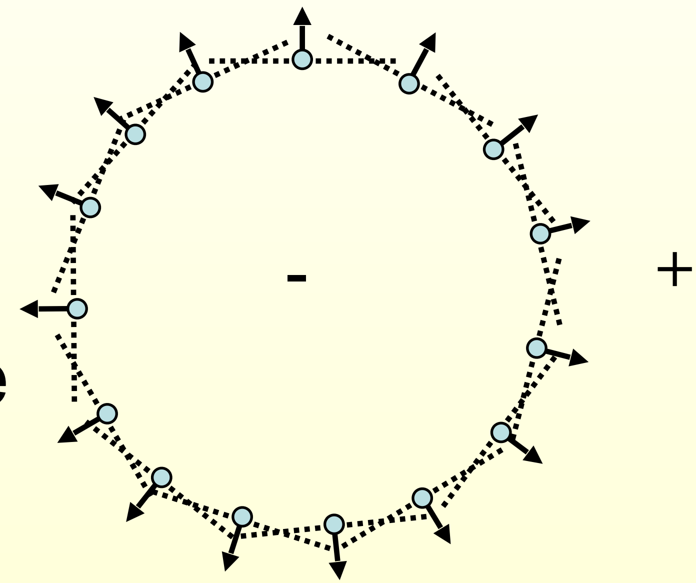
- Normals help to distinguish between inside and outside



"Surface reconstruction from unorganized points", Hoppe et al., ACM SIGGRAPH 1992
http://research.microsoft.com/en-us/um/people/hoppe/proj/recon/

SDF from Points and Normals

- Compute signed distance to the tangent plane of the closest point
- Normals help to distinguish between inside and outside

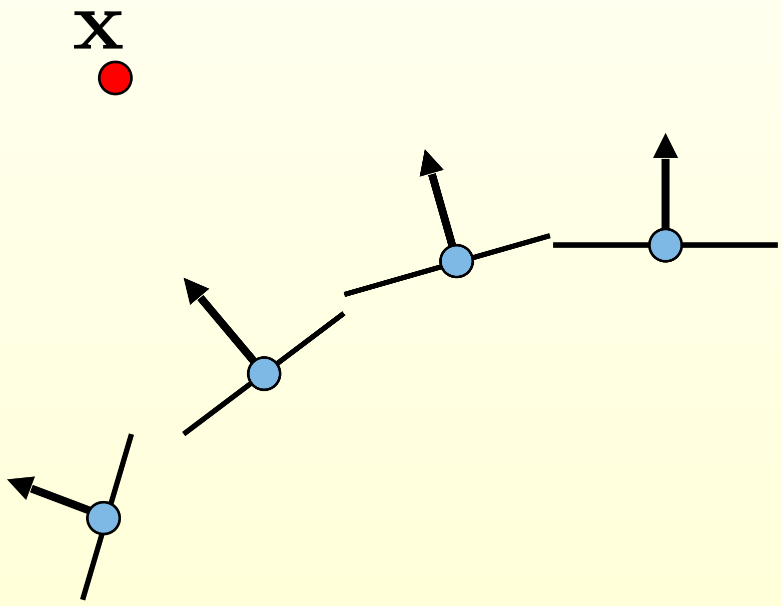


"Surface reconstruction from unorganized points", Hoppe et al., ACM SIGGRAPH 1992
http://research.microsoft.com/en-us/um/people/hoppe/proj/recon/

SDF from Points and Normals

- Compute signed distance to the tangent plane of the closest point

$$f(x) = (x-p)^T \mathbf{n}_p$$

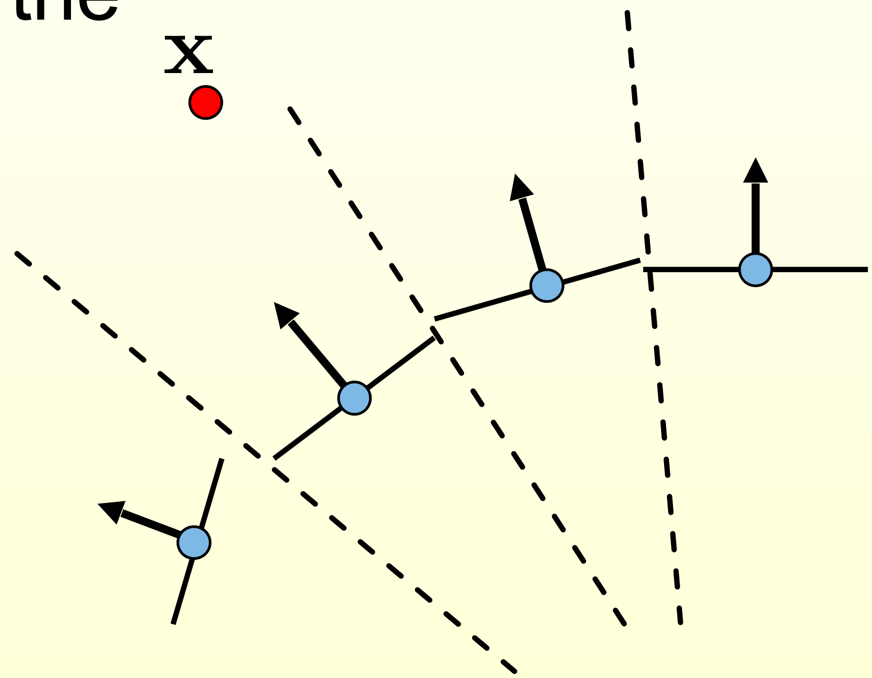


"Surface reconstruction from unorganized points", Hoppe et al., ACM SIGGRAPH 1992
http://research.microsoft.com/en-us/um/people/hoppe/proj/recon/

SDF from Points and Normals

- Compute signed distance to the tangent plane of the closest point

$$f(x) = (x - p)^T \mathbf{n}_p$$

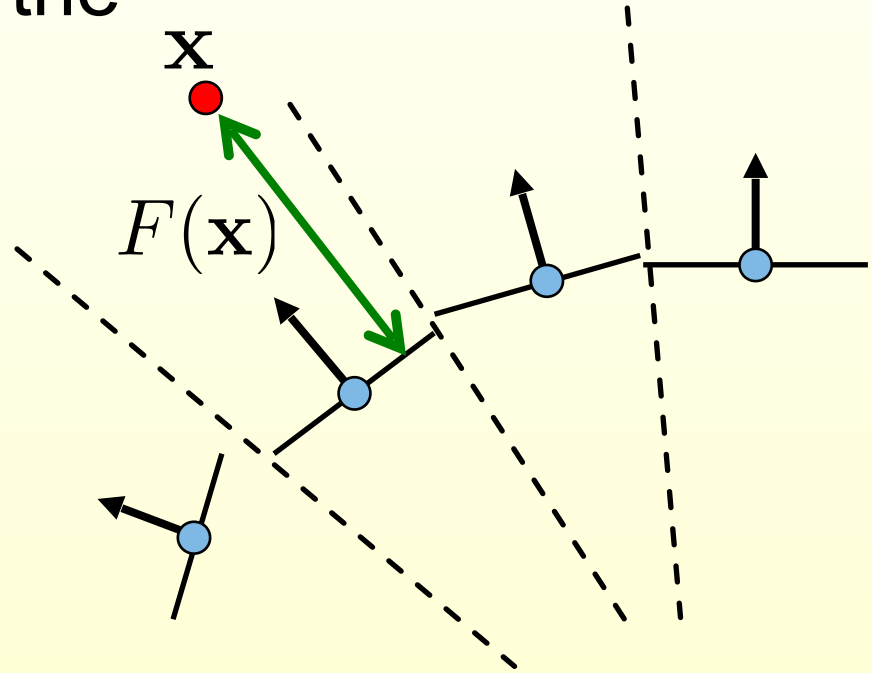


“Surface reconstruction from unorganized points”, Hoppe et al., ACM SIGGRAPH 1992
http://research.microsoft.com/en-us/um/people/hoppe/proj/recon/

SDF from Points and Normals

- Compute signed distance to the tangent plane of the closest point

$$f(x) = (x - p)^T \mathbf{n}_p$$



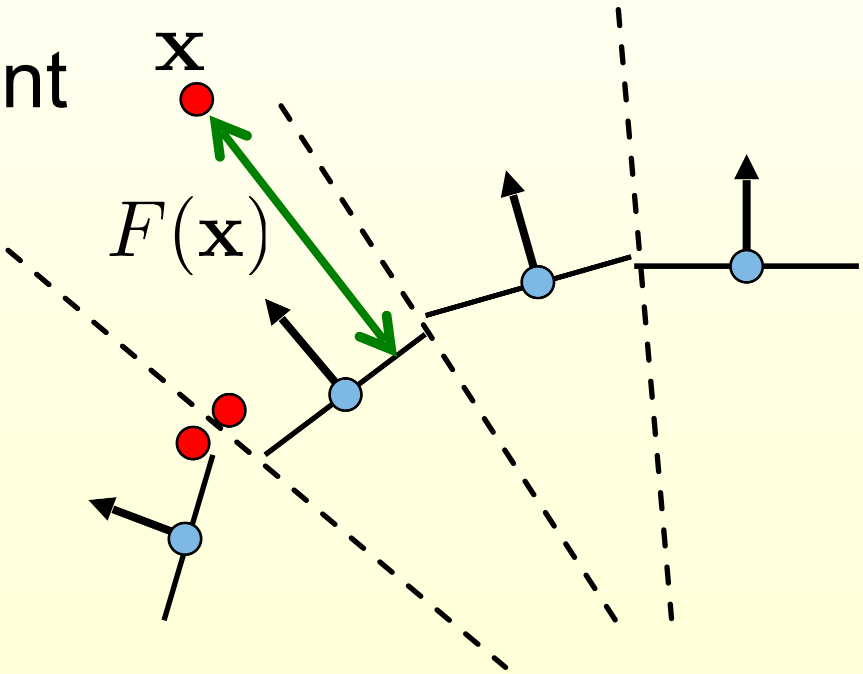
"Surface reconstruction from unorganized points", Hoppe et al., ACM SIGGRAPH 1992
http://research.microsoft.com/en-us/um/people/hoppe/proj/recon/

SDF from Points and Normals

- Compute signed distance to the tangent plane of the closest point

$$f(x) = (x - p)^T \mathbf{n}_p$$

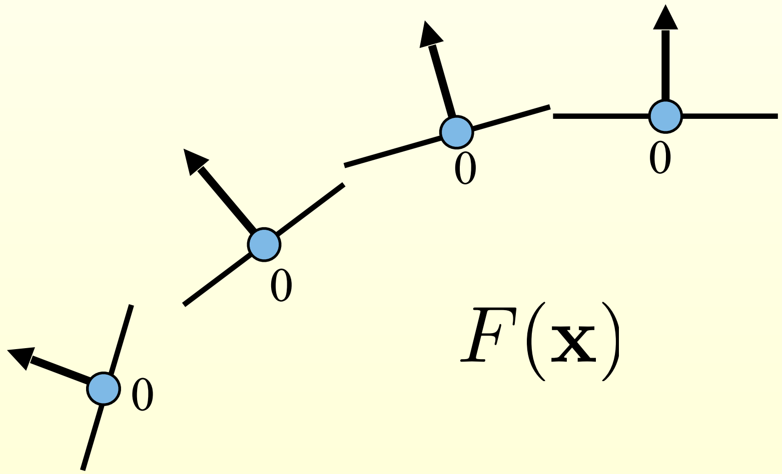
- The function will be discontinuous



"Surface reconstruction from unorganized points", Hoppe et al., ACM SIGGRAPH 1992
http://research.microsoft.com/en-us/um/people/hoppe/proj/recon/

Smooth SDF

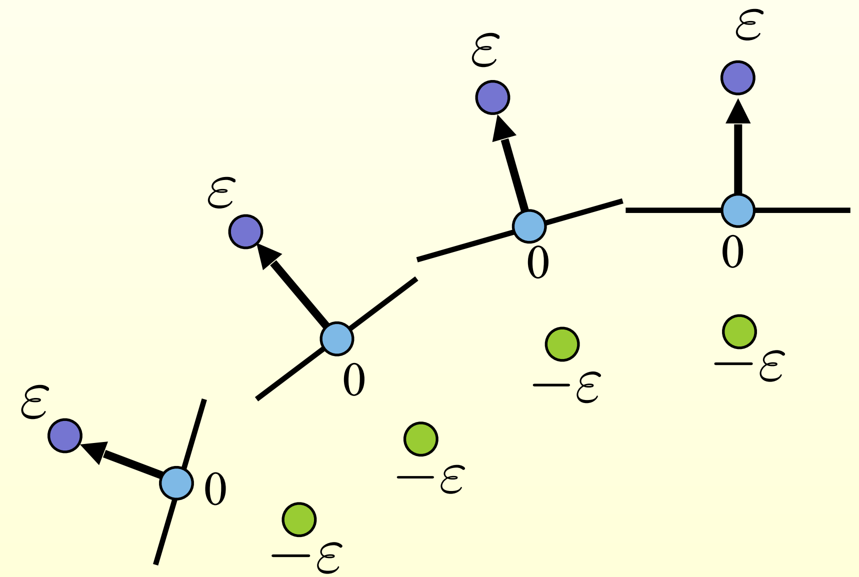
- Instead find a smooth formulation for F.
- Scattered data interpolation:
  - $F(\mathbf{p}_i) = 0$
  - F is smooth
  - Avoid trivial $F \equiv 0$



"Reconstruction and representation of 3D objects with radial basis functions", Carr et al., ACM SIGGRAPH 2001

Smooth SDF

- Scattered data interpolation:
  - $F(\mathbf{p}_i) = 0$
  - F is smooth
  - Avoid trivial $F \equiv 0$
- Add off-surface constraints



$$F(\mathbf{p}_i + \varepsilon \mathbf{n}_i) = \varepsilon$$

$$F(\mathbf{p}_i - \varepsilon \mathbf{n}_i) = -\varepsilon$$

"Reconstruction and representation of 3D objects with radial basis functions", Carr et al., ACM SIGGRAPH 2001

Radial Basis Function Interpolation

- **RBF**: Weighted sum of shifted, smooth kernels

$$F(\mathbf{x}) = \sum_{i=0}^{N-1} w_i\, \varphi(\|\mathbf{x} - \mathbf{c}_i\|) \quad N = 3n$$

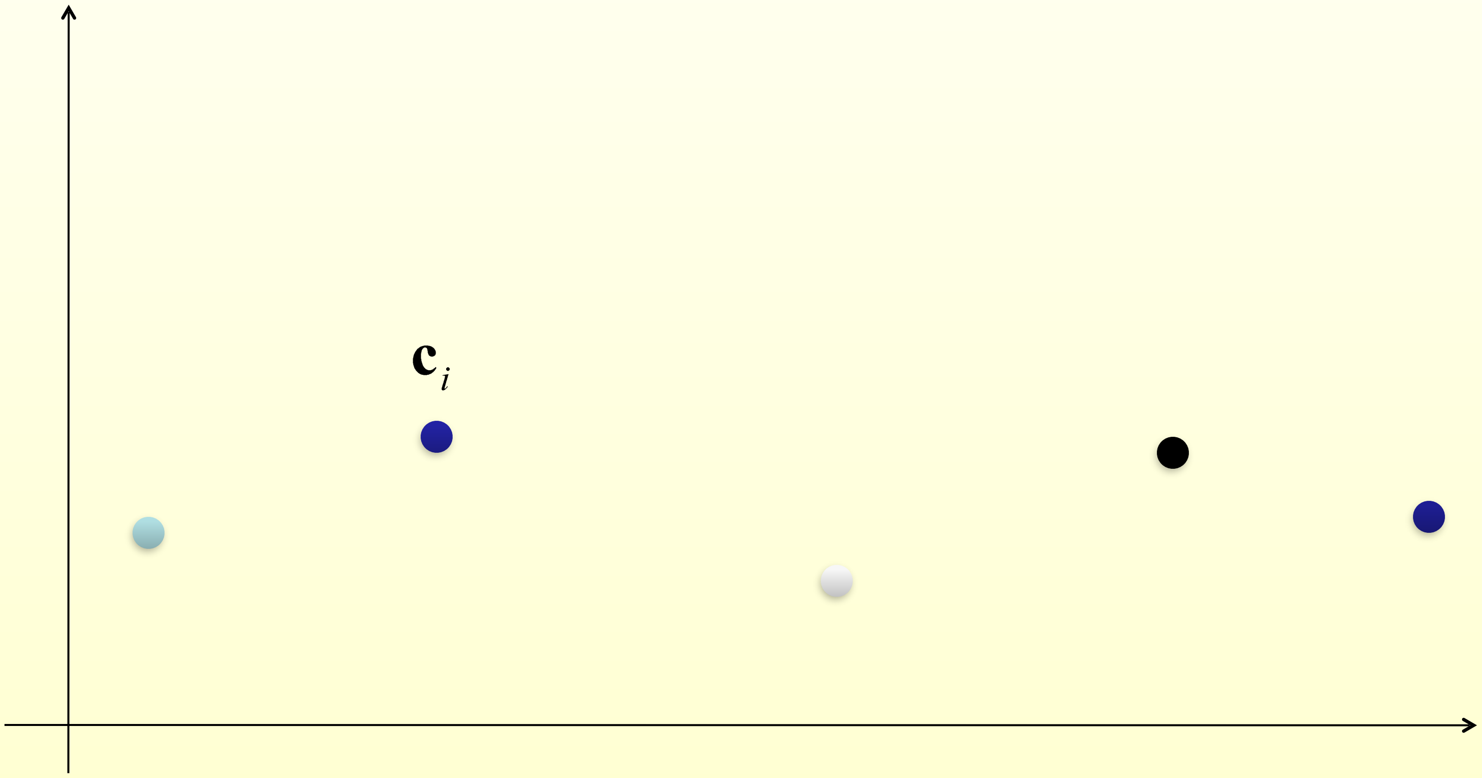
Scalar weights
Unknowns

Smooth kernels
(basis functions)
centered at constrained
points.
For example:

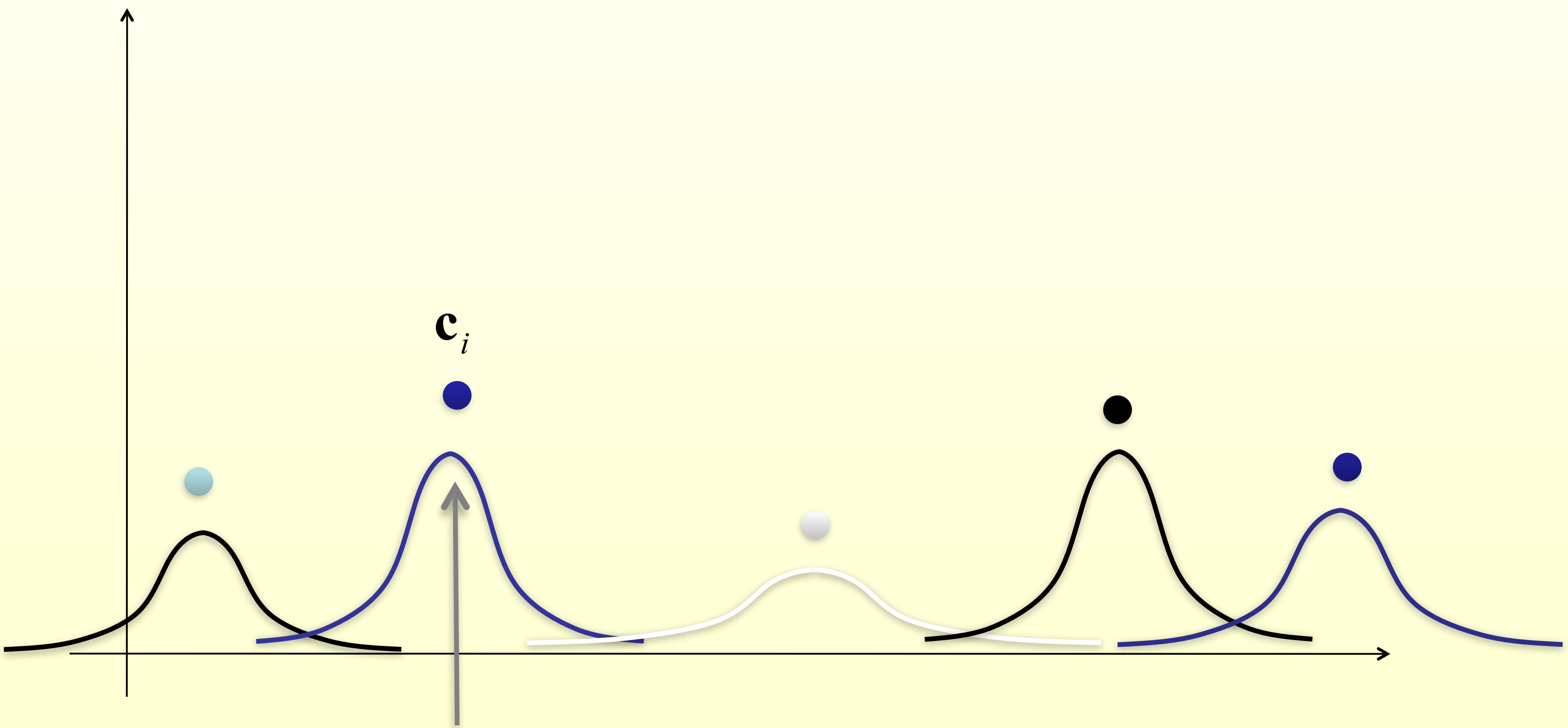
$$\varphi(r) = r^3$$

Radial Basis Functions Interpolation

Radial Basis Functions Interpolation



Radial Basis Functions Interpolation

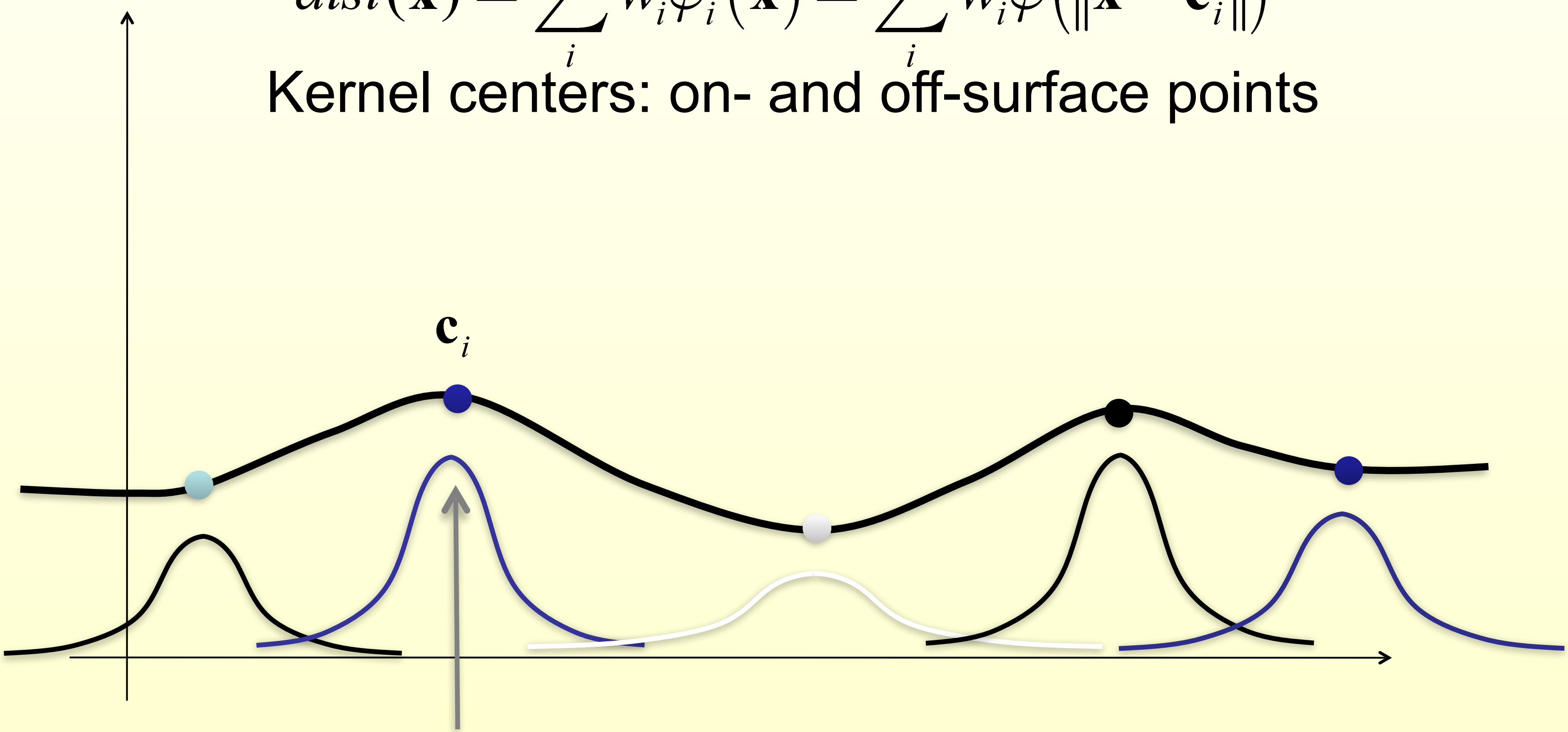


$$\varphi_i(\mathbf{x}) = \varphi\left(\left\|\mathbf{x} - \mathbf{c}_i\right\|\right)$$

Radial Basis Functions Interpolation

$$dist(\mathbf{x}) = \sum_i w_i \varphi_i(\mathbf{x}) = \sum_i w_i \varphi\left(\left\|\mathbf{x} - \mathbf{c}_i\right\|\right)$$

Kernel centers: on- and off-surface points



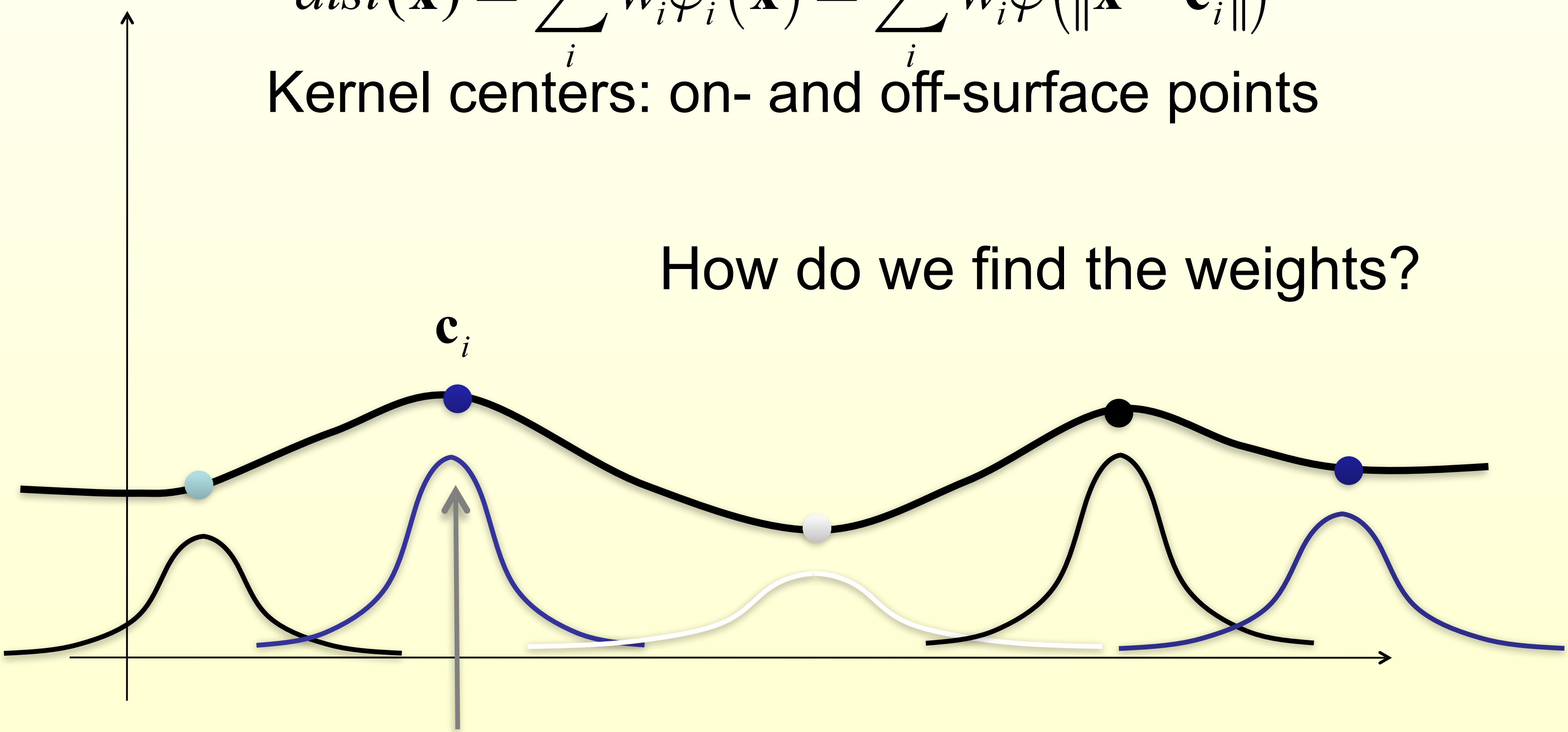
$$\varphi_i(\mathbf{x}) = \varphi\left(\left\|\mathbf{x} - \mathbf{c}_i\right\|\right)$$

Radial Basis Functions Interpolation

$$dist(\mathbf{x}) = \sum_i w_i \varphi_i(\mathbf{x}) = \sum_i w_i \varphi\left(\left\|\mathbf{x} - \mathbf{c}_i\right\|\right)$$

Kernel centers: on- and off-surface points

How do we find the weights?



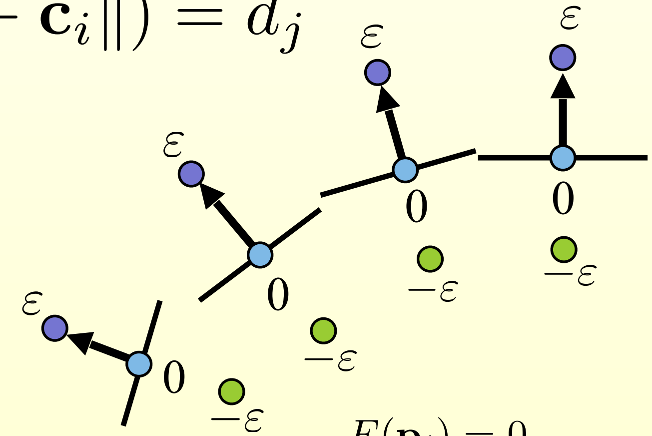
$$\varphi_i(\mathbf{x}) = \varphi\left(\left\|\mathbf{x} - \mathbf{c}_i\right\|\right)$$

Radial Basis Function Interpolation

- Interpolate the constraints:

$$\{\mathbf{c}_{3i}, \mathbf{c}_{3i+1}, \mathbf{c}_{3i+2}\} = \{\mathbf{p}_i,\ \mathbf{p}_i + \varepsilon \mathbf{n}_i,\ \mathbf{p}_i - \varepsilon \mathbf{n}_i\}$$

$$\forall j = 0, \ldots, N-1, \quad \sum_{i=0}^{N-1} w_i\, \varphi(\|\mathbf{c}_j - \mathbf{c}_i\|) = d_j$$



$$F(\mathbf{p}_i) = 0$$
$$F(\mathbf{p}_i + \varepsilon \mathbf{n}_i) = \varepsilon$$
$$F(\mathbf{p}_i - \varepsilon \mathbf{n}_i) = -\varepsilon$$

Radial Basis Function Interpolation

- Interpolate the constraints:

$$\{\mathbf{c}_{3i}, \mathbf{c}_{3i+1}, \mathbf{c}_{3i+2}\} = \{\mathbf{p}_i,\ \mathbf{p}_i + \varepsilon \mathbf{n}_i,\ \mathbf{p}_i - \varepsilon \mathbf{n}_i\}$$

- Symmetric linear system to get the weights:

$$\begin{pmatrix} \varphi(\|\mathbf{c}_0 - \mathbf{c}_0\|) & \cdots & \varphi(\|\mathbf{c}_0 - \mathbf{c}_{N-1}\| \\ \vdots & \ddots & \vdots \\ \varphi(\|\mathbf{c}_{N-1} - \mathbf{c}_0\|) & \cdots & \varphi(\|\mathbf{c}_{N-1} - \mathbf{c}_{N-1}\|) \end{pmatrix} \begin{pmatrix} w_0 \\ \vdots \\ w_{N-1} \end{pmatrix} = \begin{pmatrix} d_0 \\ \vdots \\ d_{N-1} \end{pmatrix}$$

$3n$ equations

$3n$ variables

RBF Kernels

$$\varphi(r) = r^3$$

- Triharmonic:
  - Globally supported
  - Leads to dense symmetric linear system
  - C^2 smoothness
  - Works well for highly irregular sampling

RBF Kernels

- Polyharmonic spline
  - $\varphi(r) = r^k \log(r), \;\; k = 2, 4, 6 \ldots$
  - $\varphi(r) = r^k, \;\; k = 1, 3, 5 \ldots$

- Multiquadratic

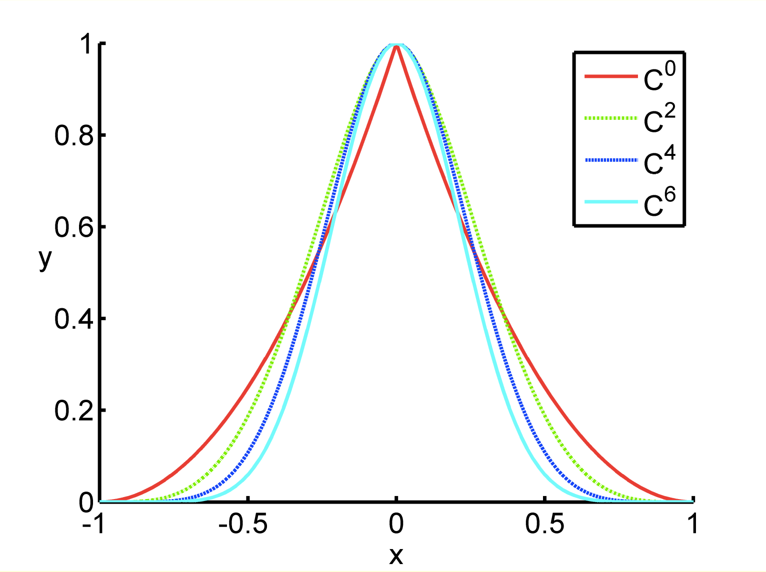
$$\varphi(r) = \sqrt{r^2 + \beta^2}$$

- Gaussian

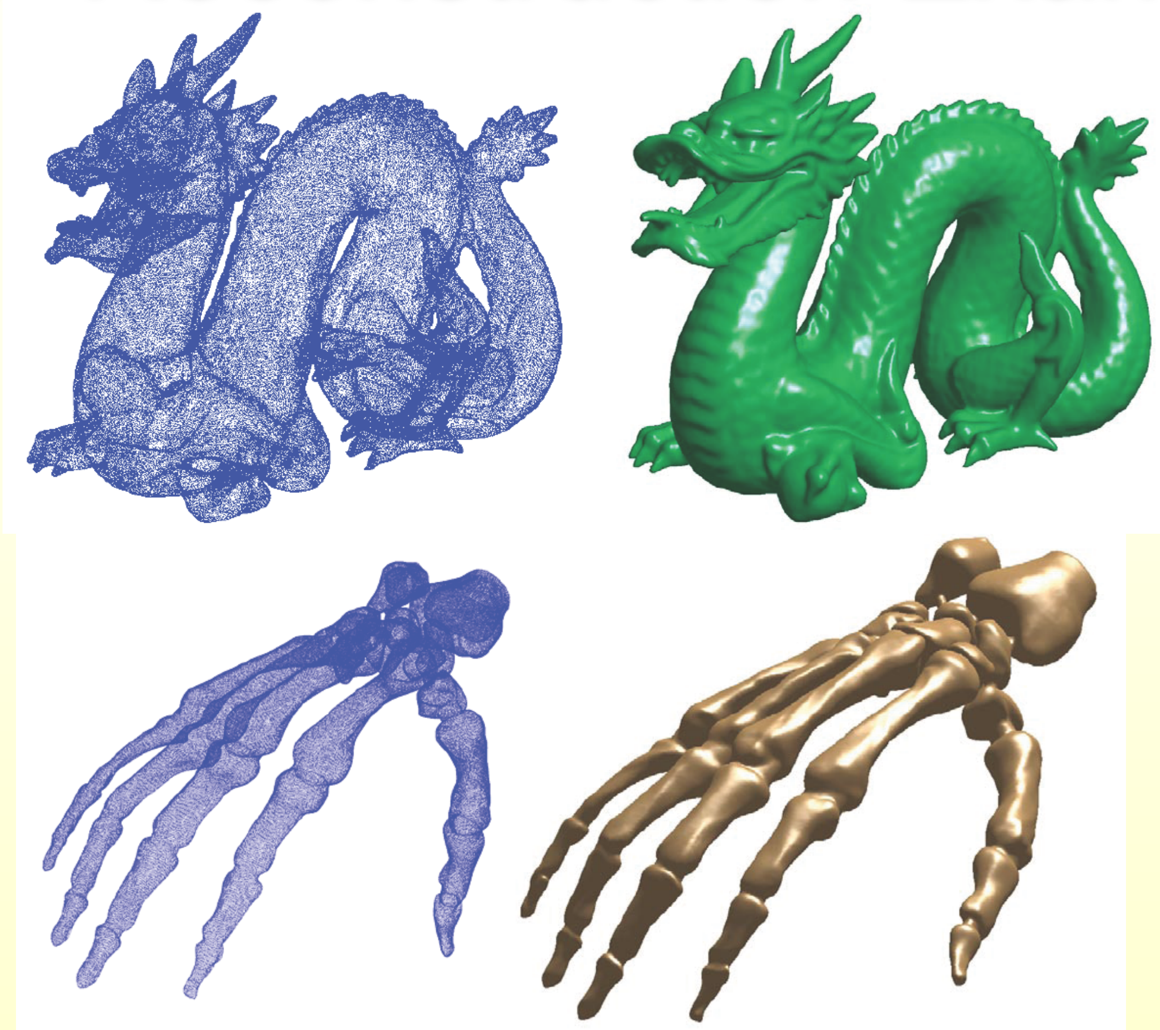
$$\varphi(r) = e^{-\beta r^2}$$

- B-Spline (compact support)

$$\varphi(r) = \text{piecewise-polynomial}(r)$$

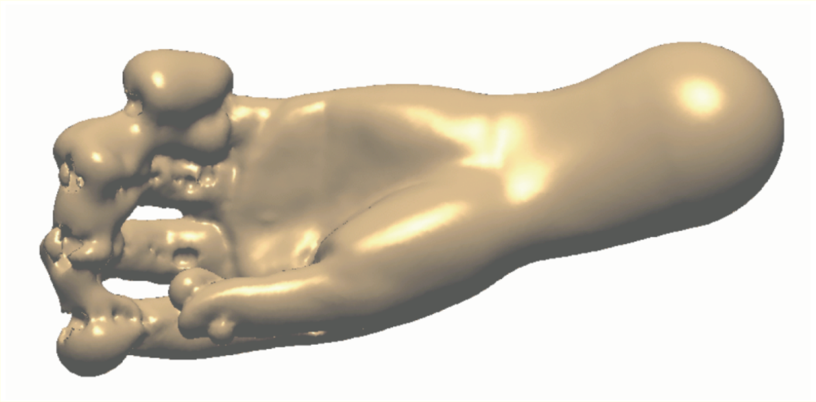


RBF Reconstruction Examples

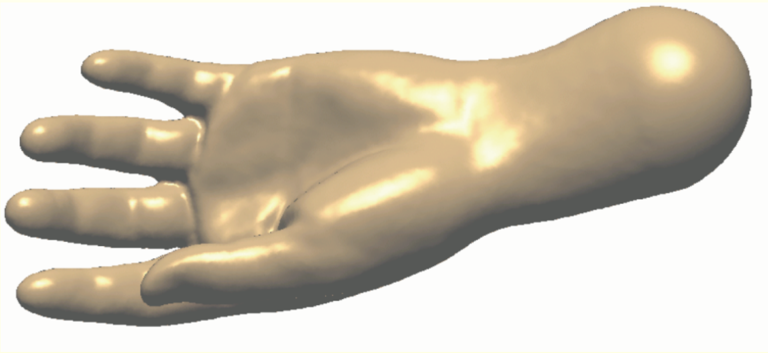


“Reconstruction and representation of 3D objects with radial basis functions”, Carr et al., ACM SIGGRAPH 2001

Off-Surface Points



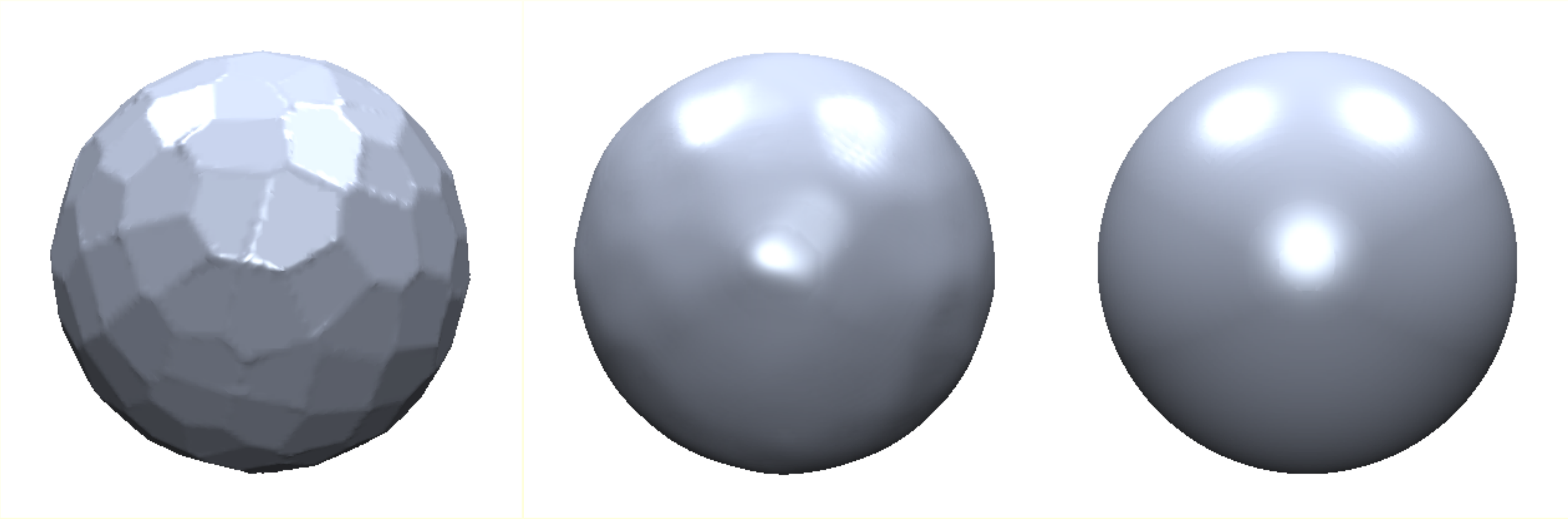
Insufficient number/
badly placed off-surface points



Properly chosen off-surface points

"Reconstruction and representation of 3D objects with radial basis functions", Carr et al., ACM SIGGRAPH 2001

Comparison of the various SDFs so far



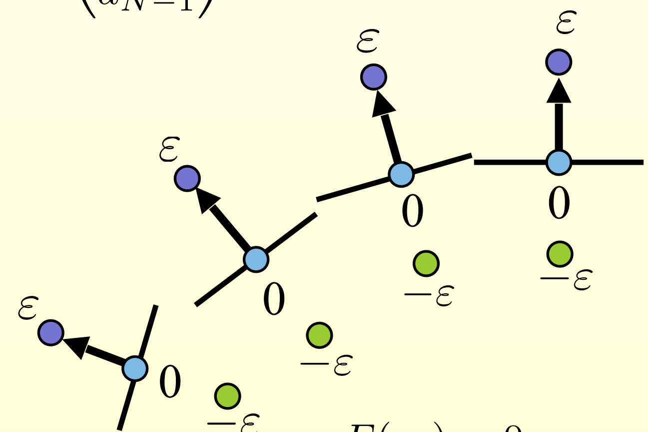
Distance to plane | Compact RBF | Global RBF Triharmonic

RBF – Discussion

- Global definition! $$F(\mathbf{x}) = \sum_{i=0}^{N-1} w_i \, \varphi(\|\mathbf{x} - \mathbf{c}_i\|)$$

$$\begin{pmatrix} \varphi(\|\mathbf{c}_0 - \mathbf{c}_0\|) & \dots & \varphi(\|\mathbf{c}_0 - \mathbf{c}_{N-1}\| \\ \vdots & \ddots & \vdots \\ \varphi(\|\mathbf{c}_{N-1} - \mathbf{c}_0\|) & \dots & \varphi(\|\mathbf{c}_{N-1} - \mathbf{c}_{N-1}\|) \end{pmatrix} \begin{pmatrix} w_0 \\ \vdots \\ w_{N-1} \end{pmatrix} = \begin{pmatrix} d_0 \\ \vdots \\ d_{N-1} \end{pmatrix}$$

- Global optimization of the weights, even if the basis functions are local



$$F(\mathbf{p}_i) = 0$$
$$F(\mathbf{p}_i + \varepsilon \mathbf{n}_i) = \varepsilon$$
$$F(\mathbf{p}_i - \varepsilon \mathbf{n}_i) = -\varepsilon$$

Complexity Issues

Complexity Issues

- Solve the linear system for RBF weights
  - Hard to solve for large number of samples

Complexity Issues

- Solve the linear system for RBF weights
  - Hard to solve for large number of samples
- Compactly supported RBFs
  - Sparse linear system
  - Efficient solvers
  - .. but less smooth!

Complexity Issues

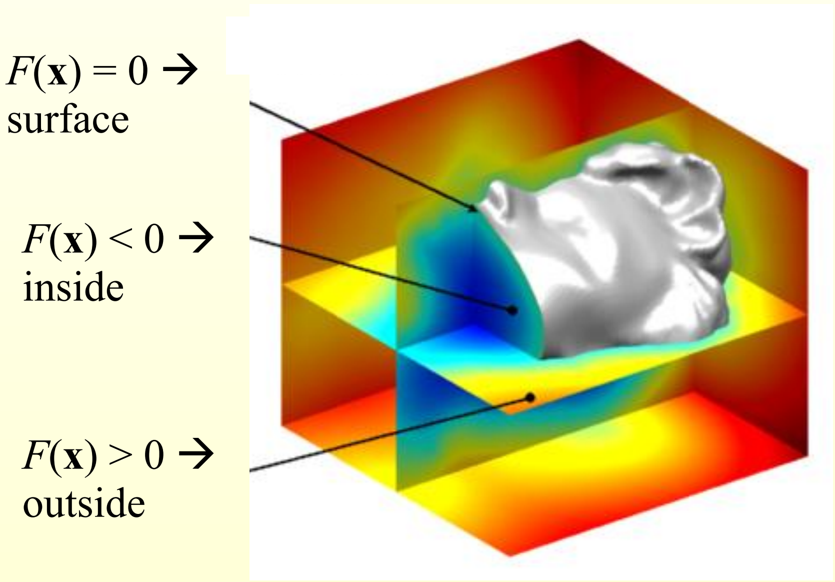
- Solve the linear system for RBF weights
  - Hard to solve for large number of samples
- Compactly supported RBFs
  - Sparse linear system
  - Efficient solvers
  - .. but less smooth!
- Greedy RBF fitting
  - Start with a few RBFs only
  - Add more RBFs in region of large error

Complexity Issues

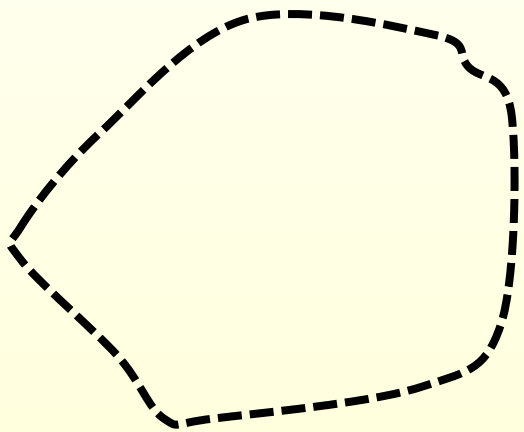
- Solve the linear system for RBF weights
  - Hard to solve for large number of samples
- Compactly supported RBFs
  - Sparse linear system
  - Efficient solvers
  - .. but less smooth!
- Greedy RBF fitting
  - Start with a few RBFs only
  - Add more RBFs in region of large error
- Even better: Moving Least Squares! (maybe later….)

Extracting the Surface

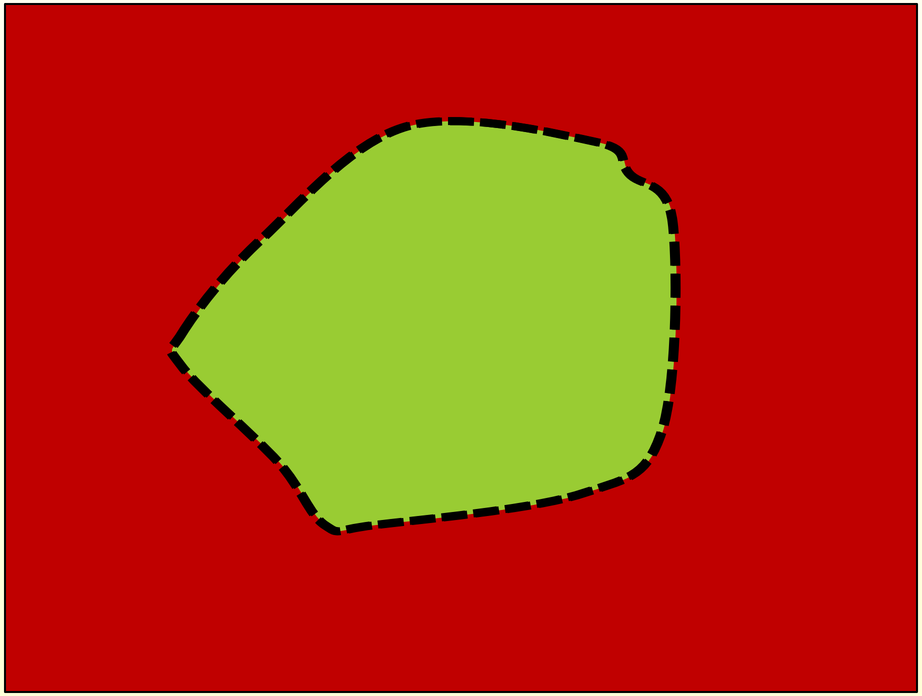
- Wish to compute a manifold mesh of the level set



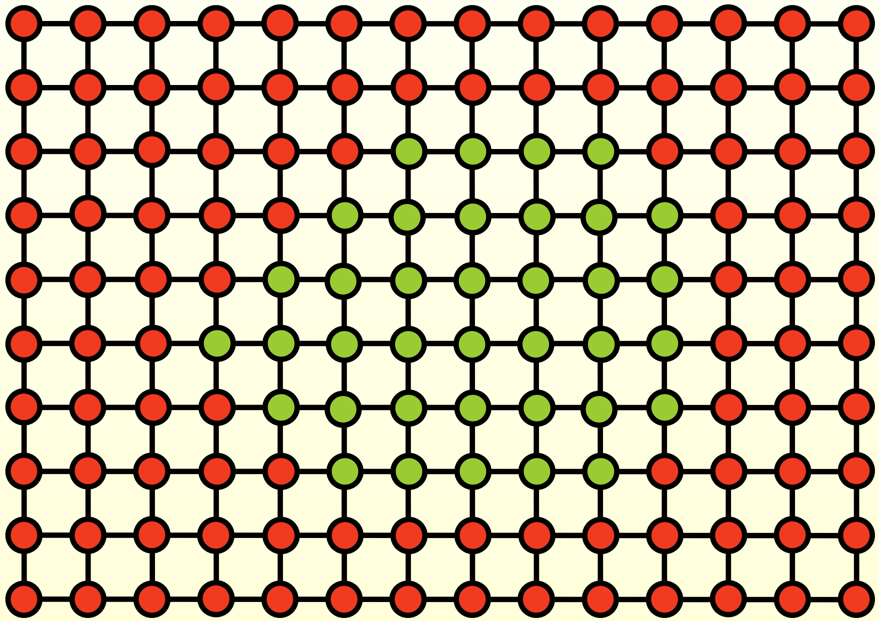
Sample the SDF



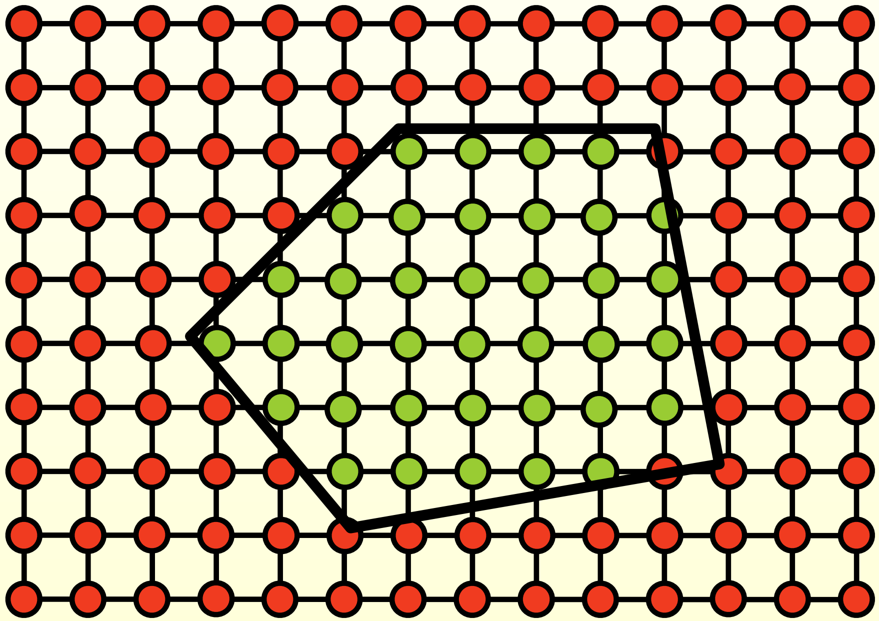
Sample the SDF

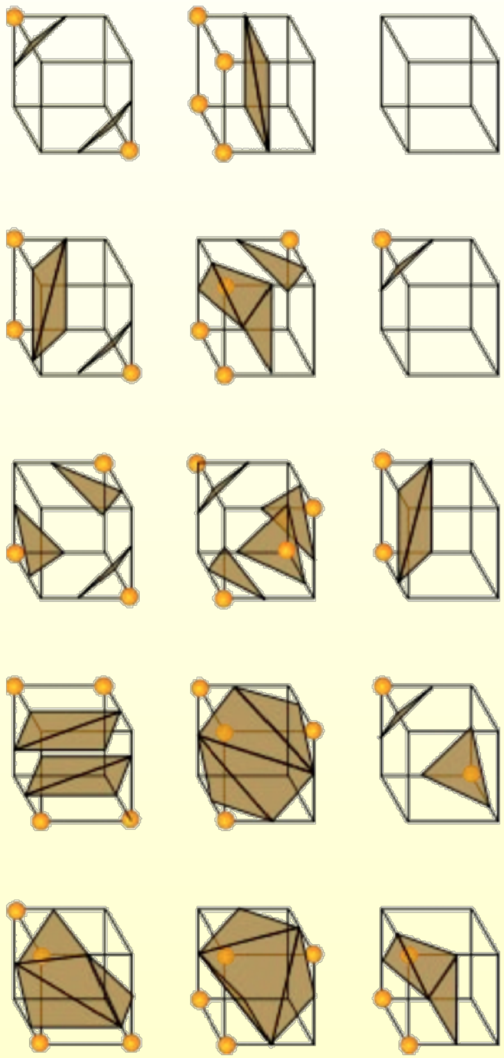


Sample the SDF



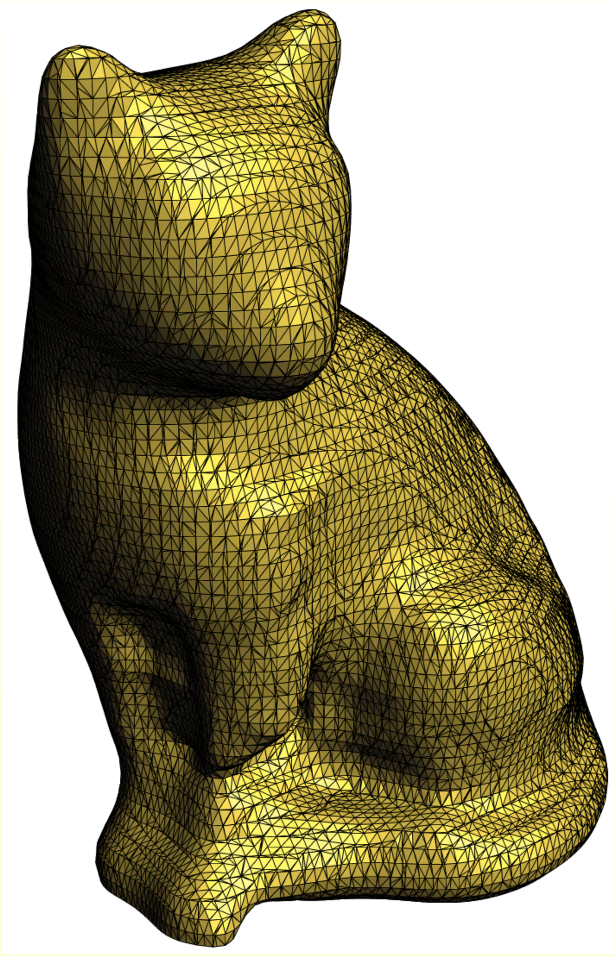
Sample the SDF





Marching Cubes! (in previous lecture)

Example: Reconstruction



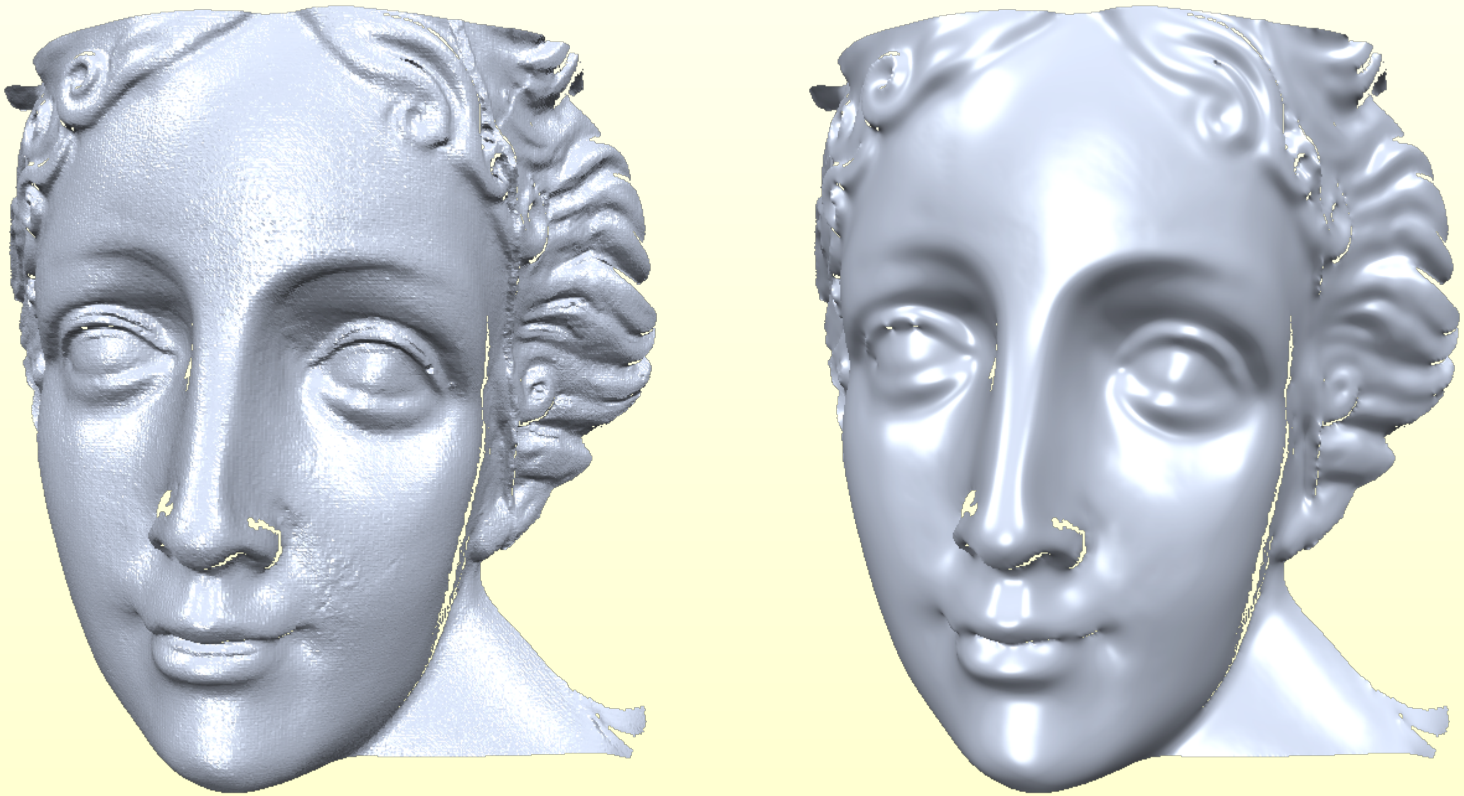
Other Methods

- Better use of normals: [Shen et al. SIGGRAPH 2004]

- ***Poisson Reconstruction*** : Kazhdan et al., SGP 2006
http://www.cs.jhu.edu/~misha/Code/PoissonRecon/

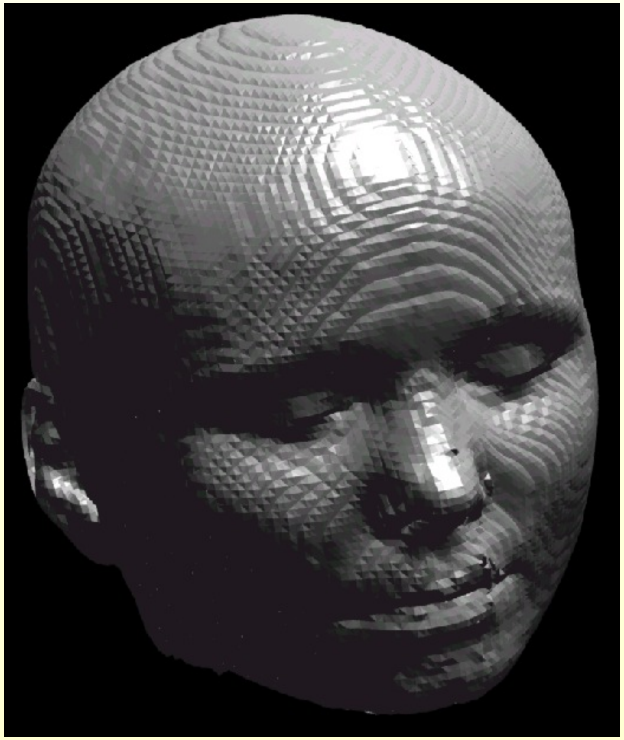
Smoothing & Remeshing– Motivation

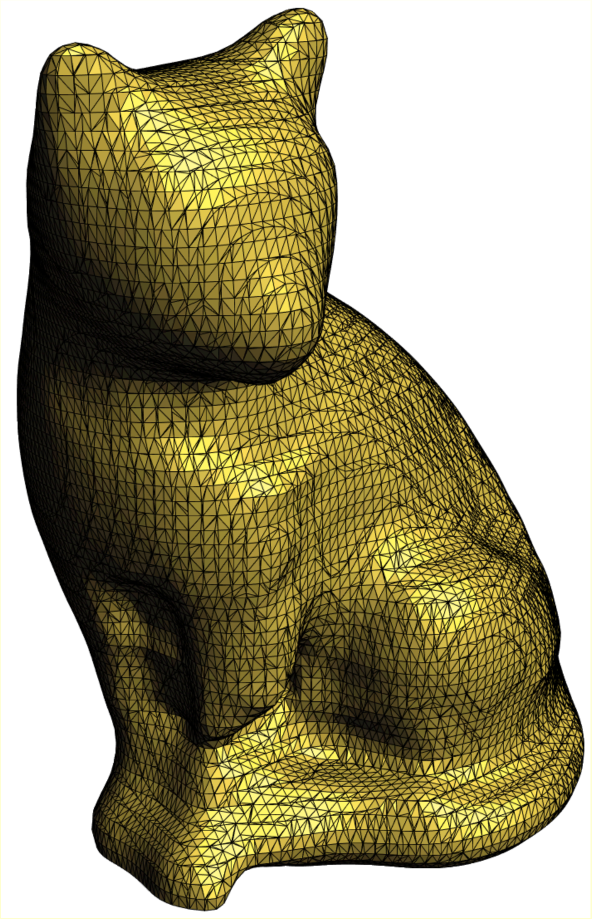
- Scanned surfaces can be noisy



Smoothing & Remeshing– Motivation

- Marching Cubes meshes can be ugly





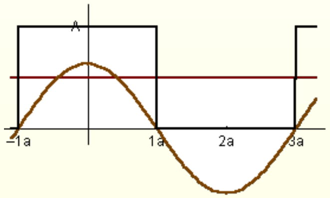
Fourier analysis - Example

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

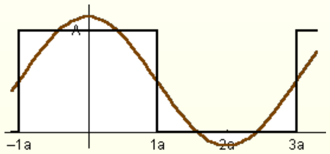
Joseph Fourier 1768 - 1830

Fourier analysis - Example

basis functions



weighted sum



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

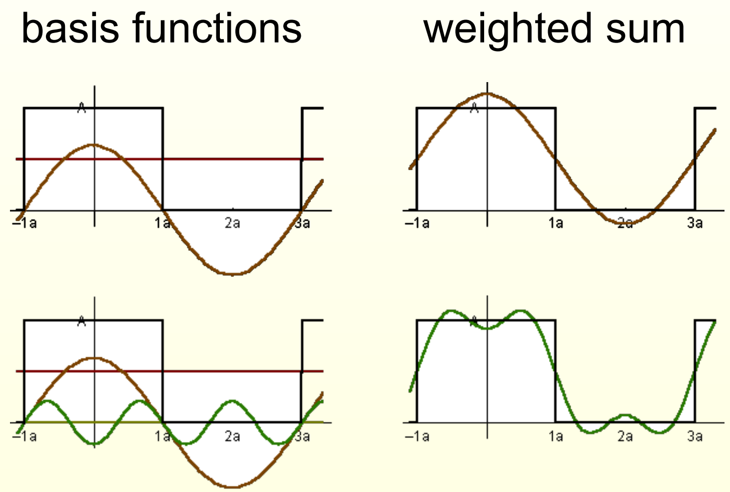
Joseph Fourier 1768 - 1830

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

Coefficients : co-integrate function with basis

Fourier analysis - Example

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



Joseph Fourier 1768 - 1830

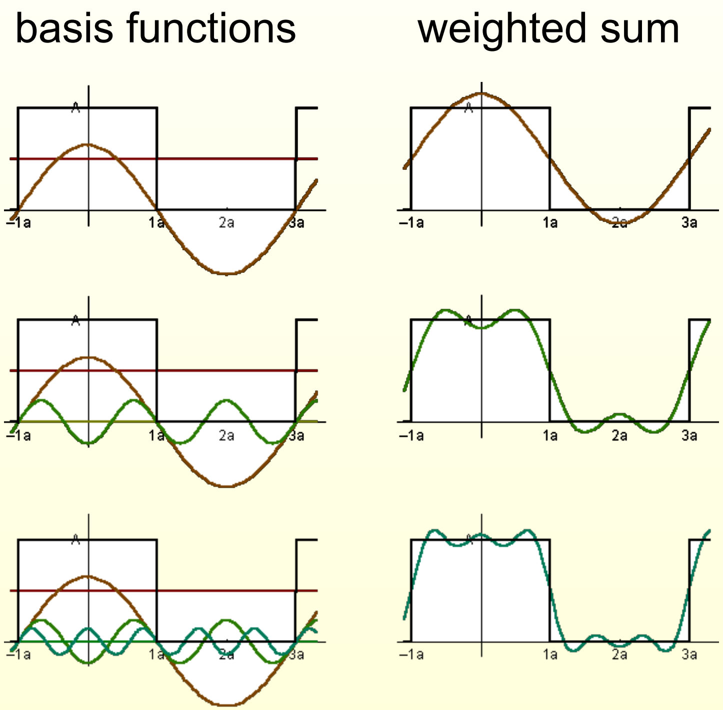
$$f(x) = a_0 + a_1 \cos(x) + a_2 \cos(3x)$$

Coefficients : co-integrate function with basis

Fourier analysis - Example

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

Joseph Fourier 1768 - 1830



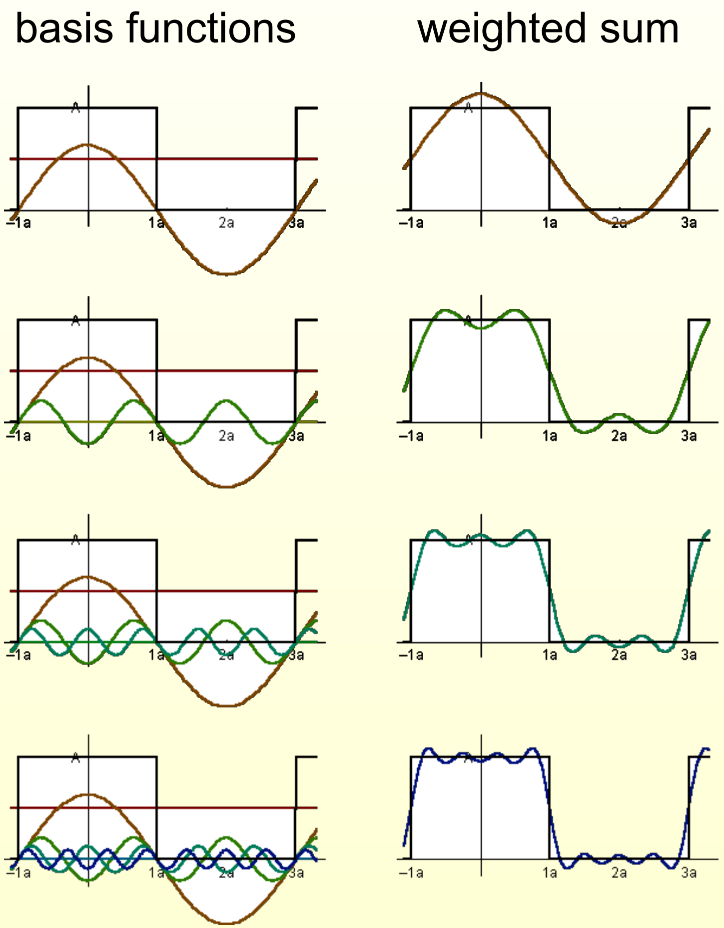
$$f(x) = a_0 + a_1 \cos(x) + a_2 \cos(3x) + a_3 \cos(5x)$$

Coefficients : co-integrate function with basis

Fourier analysis - Example

- 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 $\langle f, g \rangle := \int_{-\infty}^{\infty} f(x)\,\overline{g(x)}\,\mathrm{d}x$
- Orthonormal basis : complex “waves”

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

More generally - Fourier analysis

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

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

Spatial Domain

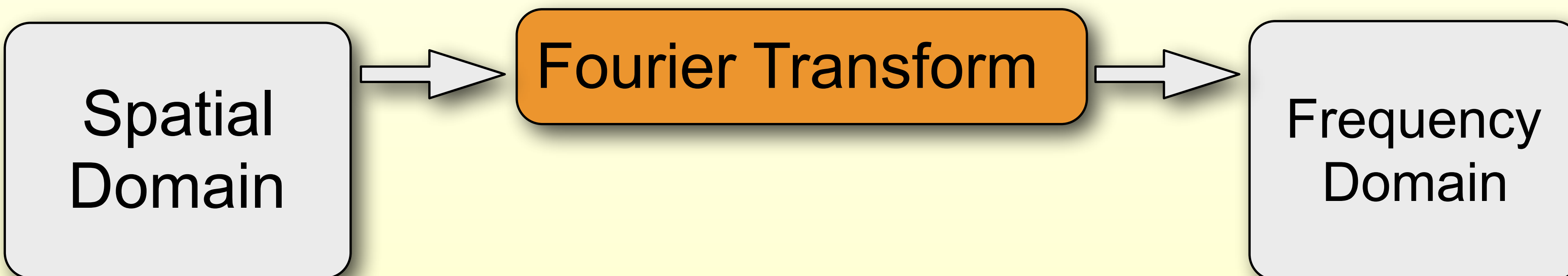
Frequency Domain

More generally - Fourier analysis

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

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

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

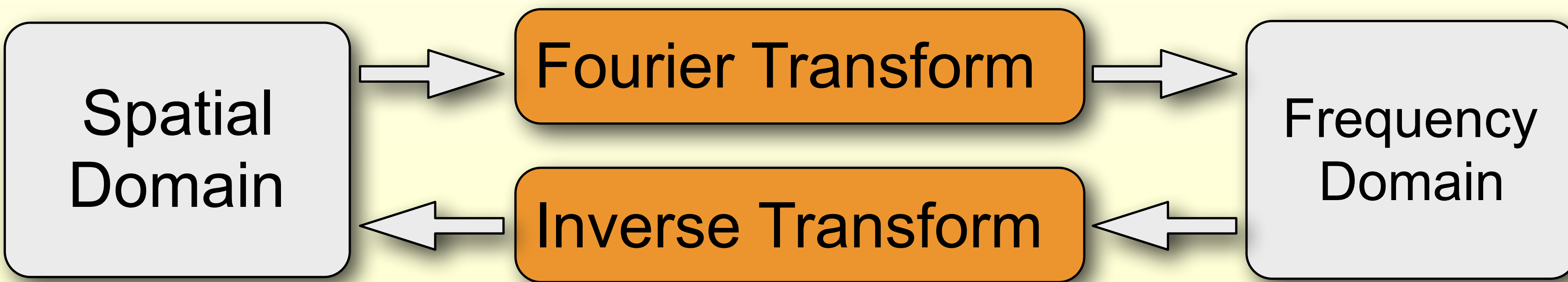


More generally - Fourier analysis

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

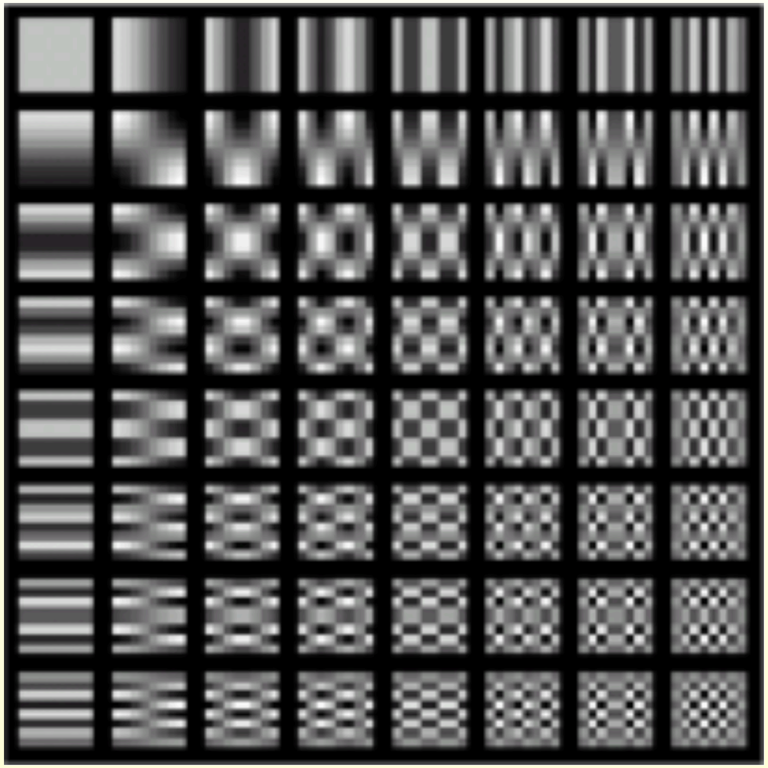
$$e_u(x) \;:=\; \mathrm{e}^{\mathrm{i}2\pi ux} \;=\; \cos(2\pi ux) - \mathrm{i}\sin(2\pi ux)$$

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



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

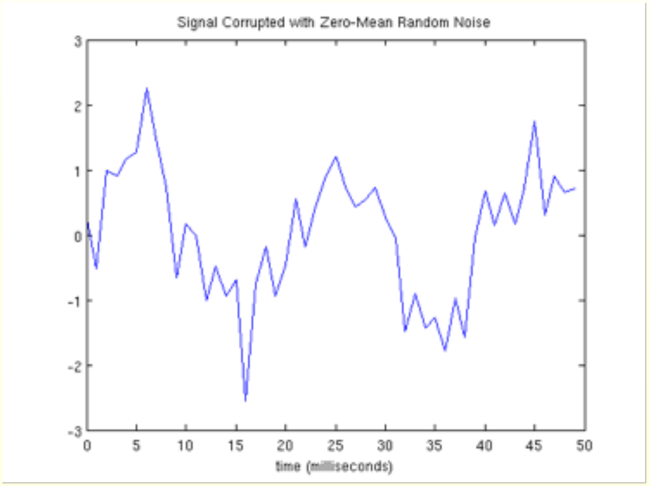
We can also Fourier on rectangular 2D domains



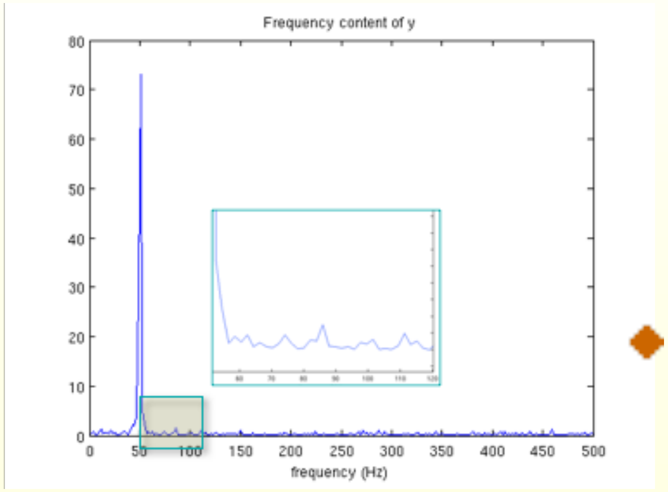
Fourier (DCT) basis functions for 8x8 grayscale images

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

Smoothing = filtering high frequencies out



spatial domain



frequency domain

- Spatial domain $f(x) \rightarrow$ Frequency domain $F(u)$

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

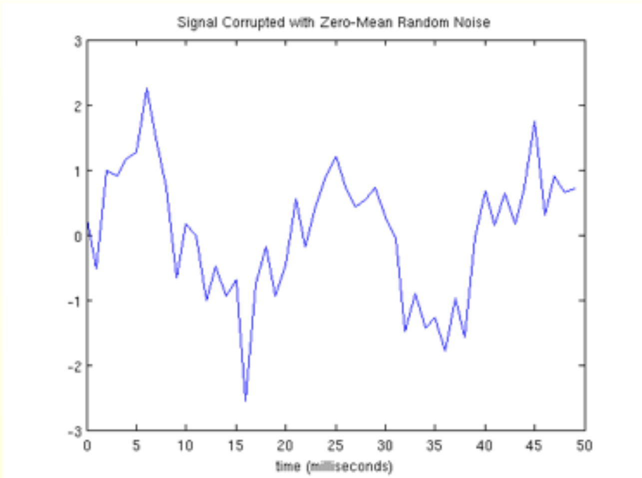
- Multiply by low-pass filter $G(u)$

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

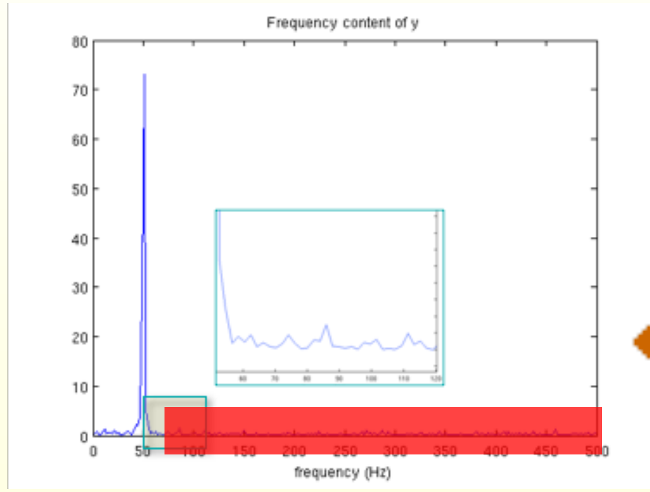
- Frequency domain $F(u) \rightarrow$ Spatial domain $f(x)$

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

Smoothing = filtering high frequencies out



spatial domain



frequency domain

- Spatial domain $f(x) \rightarrow$ Frequency domain $F(u)$

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

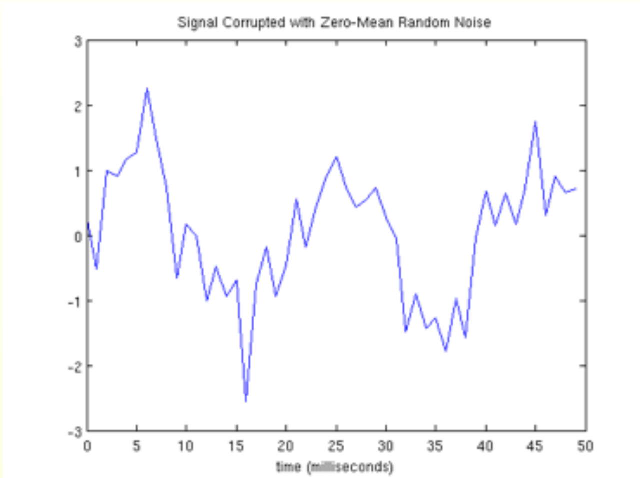
- Multiply by low-pass filter $G(u)$

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

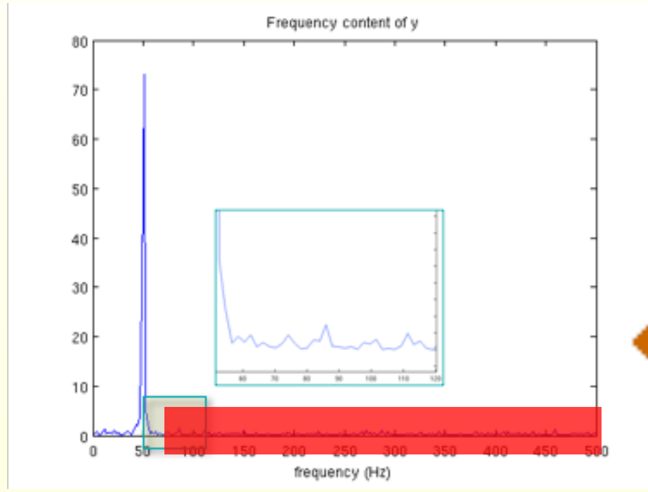
- Frequency domain $F(u) \rightarrow$ Spatial domain $f(x)$

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

Smoothing = filtering high frequencies out



spatial domain



frequency domain

- Spatial domain $f(x) \rightarrow$ Frequency domain $F(u)$

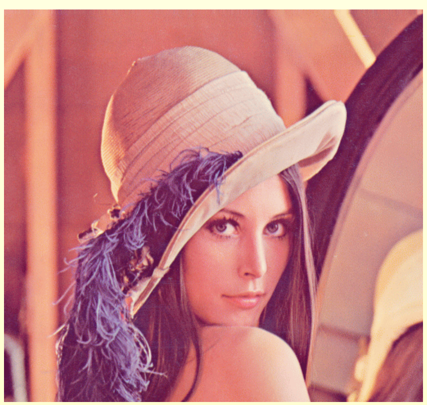
$$F(u) = \int_{-\infty}^{\infty} f(x)\, \mathrm{e}^{-\mathrm{i}2\pi ux} \mathrm{d}x$$

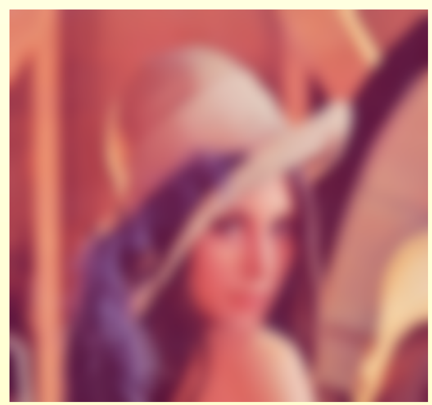
- Multiply by low-pass filter $G(u)$

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

- Frequency domain $F(u) \rightarrow$ Spatial domain $f(x)$

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





Extend Fourier to meshes?

Extend Fourier to meshes?

- So far, our functions have been defined on parameterized patches $f(u,v)$
  - Generalize to meshes!

Extend Fourier to meshes?

- So far, our functions have been defined on parameterized patches $f(u,v)$
  - Generalize to meshes!

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

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

Extend Fourier to meshes?

- So far, our functions have been defined on parameterized patches $f(u,v)$
  - Generalize to meshes!

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

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

- We need a discrete (mesh-based) version of this operator!

Continuous Laplace Operator

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

Continuous Laplace Operator

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

Laplace
operator

$$\Delta f$$

function in
Euclidean space

Continuous Laplace Operator

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

Laplace operator

2nd partial derivatives

$$\Delta f \qquad = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \ldots$$

function in Euclidean space

Cartesian coordinates

Continuous Laplace Operator

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

Laplace operator

gradient operator

2nd partial derivatives

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

function in Euclidean space

divergence operator

Cartesian coordinates

Continuous 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

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

function in Euclidean space

divergence operator

Cartesian coordinates

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

Continuous 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

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

function in Euclidean space

divergence operator

Cartesian coordinates

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

Continuous Laplace-Beltrami Operator

- Extension of Laplace operator to functions on manifolds

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

Continuous Laplace-Beltrami Operator

- Extension of Laplace operator to functions on manifolds

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

Laplace-Beltrami

gradient operator

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

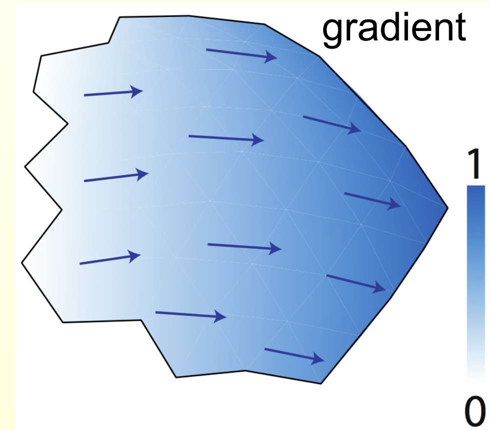
function on surface M

divergence operator

Continuous Laplace-Beltrami Operator

- Extension of Laplace operator to functions on manifolds

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



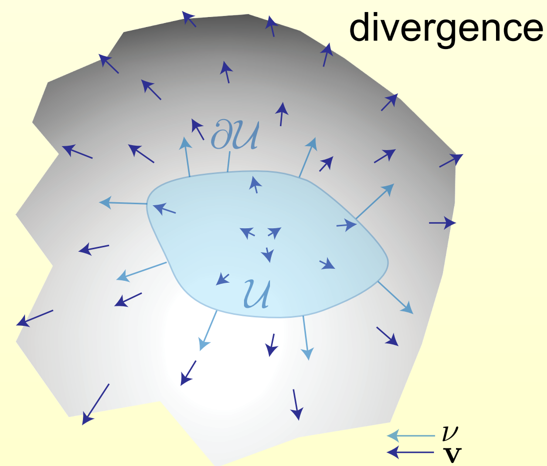
Laplace-Beltrami

gradient operator

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

function on surface M

divergence operator



Differential Properties on Meshes

Differential Properties on Meshes

- So for Laplacian, we need differential quantities (gradient, divergence…)

Differential Properties on Meshes

- So for Laplacian, we need differential quantities (gradient, divergence…)
- Assumption: meshes are piecewise linear approximations of smooth surfaces

Differential Properties on Meshes

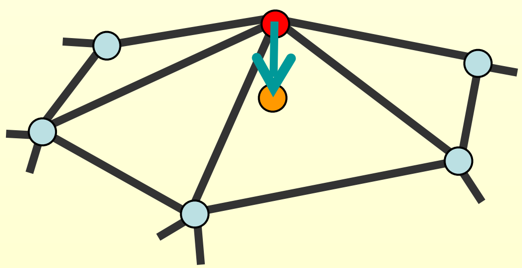
- So for Laplacian, we need differential quantities (gradient, divergence…)
- Assumption: meshes are piecewise linear approximations of smooth surfaces
- Can try fitting a smooth surface locally (say, a polynomial) and find differential quantities analytically

Differential Properties on Meshes

- So for Laplacian, we need differential quantities (gradient, divergence…)
- Assumption: meshes are piecewise linear approximations of smooth surfaces
- 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 $N(\mathbf{v})$ where typically
  - $\mathbf{v}$ = mesh vertex
  - $N_k(\mathbf{v})$ = k-ring neighborhood



Discrete Laplace-Beltrami

- Uniform discretization: $L(f)$ or Δf

$$\begin{aligned}\Delta f(\mathbf{v}) &= \sum_{v_j \in N(v)} \left(f(\mathbf{v}_j) - f(\mathbf{v}) \right) \\ &= \sum_{v_j \in N(v)} f(\mathbf{v}_j) - k f(\mathbf{v}),\ k = |N(v)|\end{aligned}$$

	1	
1	-4	1
	1	

- Similar to 5 point stencil for images!
- Depends only on connectivity : simple and efficient
- Bad approximation for irregular triangulations

Discrete Laplace-Beltrami Operator

- In matrix form

$$\Delta f(\mathbf{v}) = \sum_{v_j \in N(v)} f(\mathbf{v}_j) - k f(\mathbf{v}), \;\; k = |N(v)|$$

Discrete Laplace-Beltrami Operator

- In matrix form

$$\Delta f(\mathbf{v}) = \sum_{v_j \in N(v)} f(\mathbf{v}_j) - k f(\mathbf{v}), \ k = |N(v)|$$

$$\mathbf{F} = \begin{bmatrix} f(\mathbf{v}_1) \\ f(\mathbf{v}_2) \\ f(\mathbf{v}_3) \\ \cdots \\ f(\mathbf{v}_N) \end{bmatrix}$$

Discrete Laplace-Beltrami Operator

- In matrix form

$$\Delta f(\mathbf{v}) \;=\; \sum_{v_j \in N(v)} f(\mathbf{v}_j) - k f(\mathbf{v}),\;\; k = |N(v)|$$

$$\mathbf{Y} = \begin{bmatrix} \Delta f(\mathbf{v}_1) \\ \Delta f(\mathbf{v}_2) \\ \Delta f(\mathbf{v}_3) \\ \cdots \\ \Delta f(\mathbf{v}_N) \end{bmatrix}$$

$$\mathbf{F} = \begin{bmatrix} f(\mathbf{v}_1) \\ f(\mathbf{v}_2) \\ f(\mathbf{v}_3) \\ \cdots \\ f(\mathbf{v}_N) \end{bmatrix}$$

Discrete Laplace-Beltrami Operator

- In matrix form

$$\Delta f(\mathbf{v}) = \sum_{v_j \in N(v)} f(\mathbf{v}_j) - k f(\mathbf{v}),\ k = |N(v)|$$

$$\mathbf{Y} = \mathbf{LF}$$

$$\mathbf{Y} = \begin{bmatrix} \Delta f(\mathbf{v}_1) \\ \Delta f(\mathbf{v}_2) \\ \Delta f(\mathbf{v}_3) \\ \cdots \\ \Delta f(\mathbf{v}_N) \end{bmatrix} \qquad \mathbf{L} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \cdots & \vdots \\ w_{N1} & w_{N2} & \cdots & w_{NN} \end{bmatrix} = \{w_{ij}\} \qquad \mathbf{F} = \begin{bmatrix} f(\mathbf{v}_1) \\ f(\mathbf{v}_2) \\ f(\mathbf{v}_3) \\ \cdots \\ f(\mathbf{v}_N) \end{bmatrix}$$

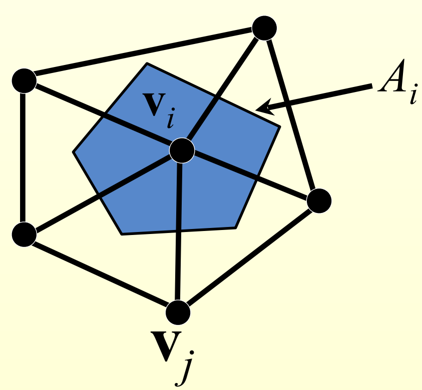
$$w_{ij} = \begin{cases} 0 & i \neq j,\ \nexists \text{ edge } (i,j) \\ 1 & i \neq j,\ \exists \text{ edge } (i,j) \\ -|N(v_i)| & i = j \end{cases}$$

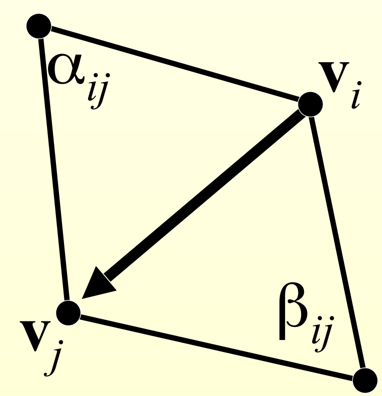
Discrete Laplace-Beltrami

- Better: cotangent formula

$$\Delta_S f(v_i) := \frac{1}{2A_i} \sum_{v_j \in \mathcal{N}_1(v_i)} (\cot\alpha_{ij} + \cot\beta_{ij}) \left(f(v_j) - f(v_i)\right)$$

A_i : vertex area (Voronoi, barycentric..)





Can be derived by discretizing continuous L-B via linear Finite Elements!

Now we can Fourier-smooth!

- Take your favorite L-B matrix L
- Compute eigenvectors $\mathbf{e}_1$, $\mathbf{e}_2$, …, $\mathbf{e}_k$ with the k smallest eigenvalues $\Longrightarrow$ matrix eigenvalues!
- Reconstruct mesh geometry (= coordinate functions, e.g. $f(x,y,z) = x$) from the eigenvectors:

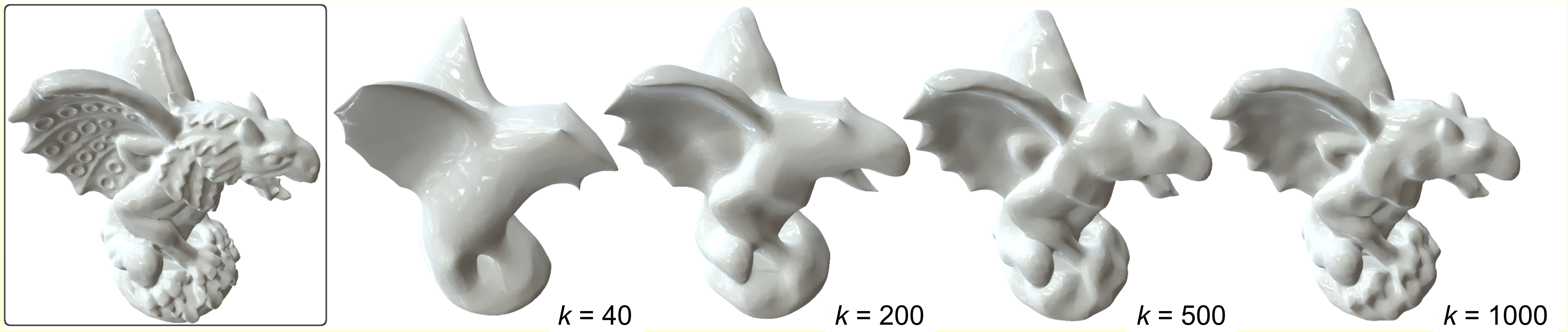
$$\mathbf{x} = [x_1, \ldots, x_n]^T \qquad \mathbf{y} = [y_1, \ldots, y_n]^T \qquad \mathbf{z} = [z_1, \ldots, z_n]^T$$

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

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

Spectral analysis on meshes

- Take your favorite L-B matrix L
- Compute eigenvectors $\mathbf{e}_1$, $\mathbf{e}_2$, …, $\mathbf{e}_k$ with the k smallest eigenvalues
- Reconstruct mesh geometry (= coordinate functions, e.g. $f(x, y, z) = x$) from the eigenvectors:

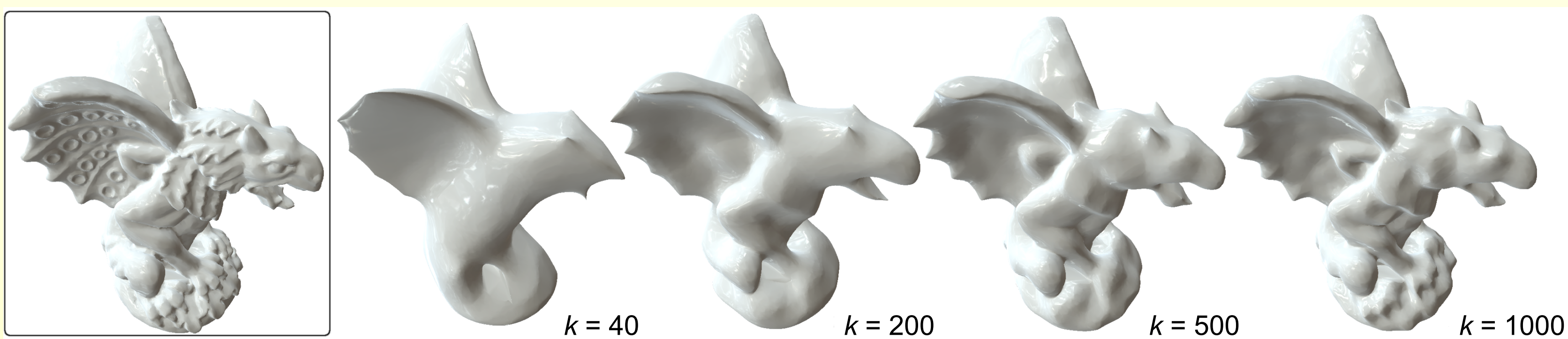


Spectral analysis on meshes

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

too expensive for large meshes

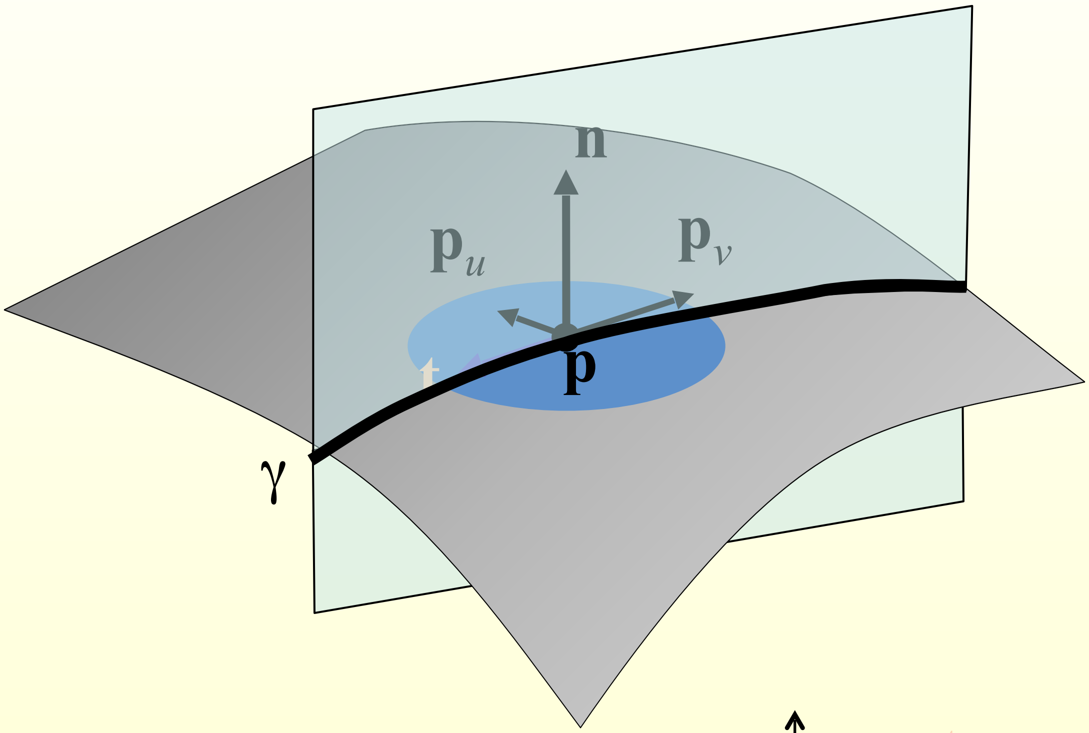
- Reconstruct mesh geometry (= coordinate functions, e.g. $f(x,y,z)=x$) from the eigenvectors:

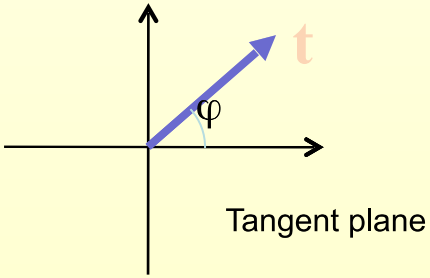


An alternative approach

- Laplace – Beltrami operator relates to mesh curvature!
- Smoothing is “reducing the curvature” !

What is Curvature?





Measure how much the surface "changes" along a the various tangential directions.

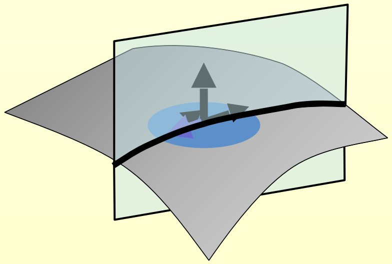
For given tangential direction $\mathbf{t}$:
Take curve γ – intersection of surface with the plane through $\mathbf{n}$ and $\mathbf{t}$.

Normal curvature:

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

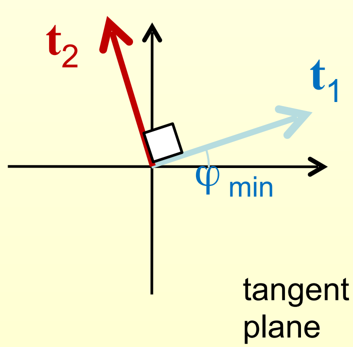
Surface Curvatures

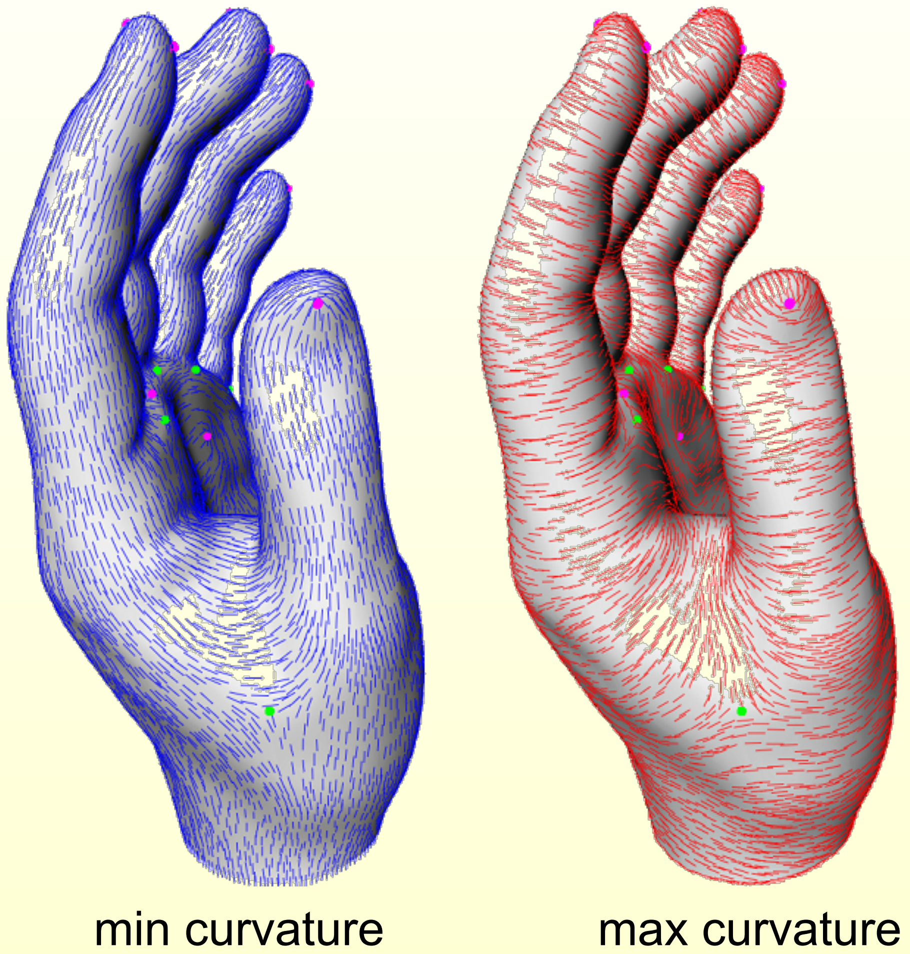
- Principal curvatures
  - Minimal curvature $$\kappa_1 = \kappa_{\min} = \min_{\varphi} \kappa_n(\varphi)$$
  - Maximal curvature $$\kappa_2 = \kappa_{\max} = \max_{\varphi} \kappa_n(\varphi)$$
- Mean curvature $$H = \frac{\kappa_1 + \kappa_2}{2} = \frac{1}{2\pi} \int_0^{2\pi} \kappa_n(\varphi) d\varphi$$
- Gaussian curvature $$K = \kappa_1 \cdot \kappa_2$$



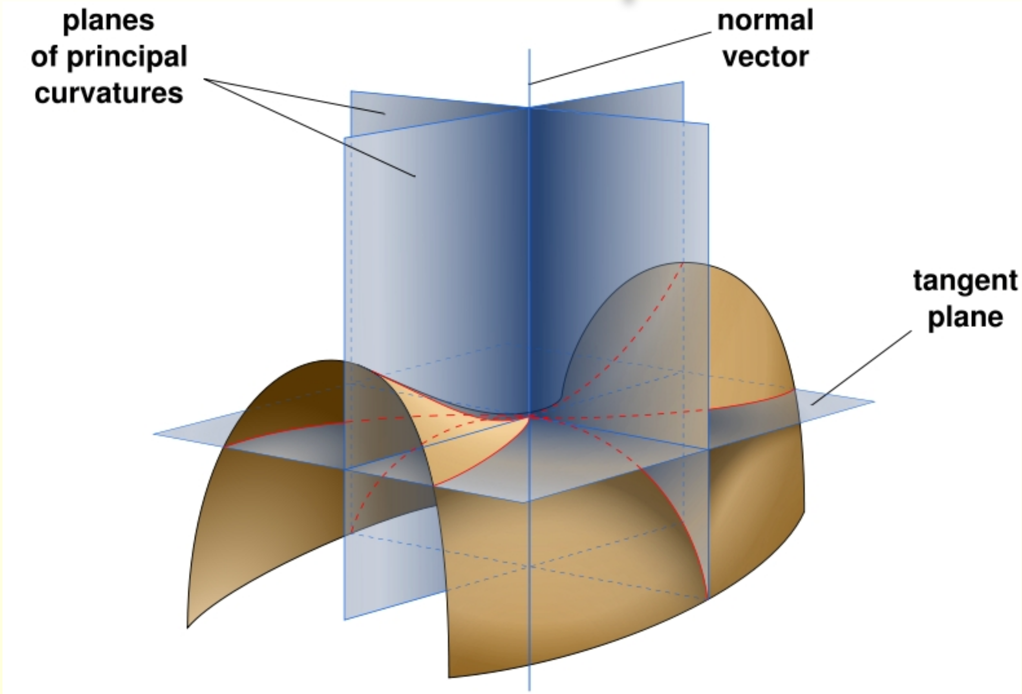
Principal Directions

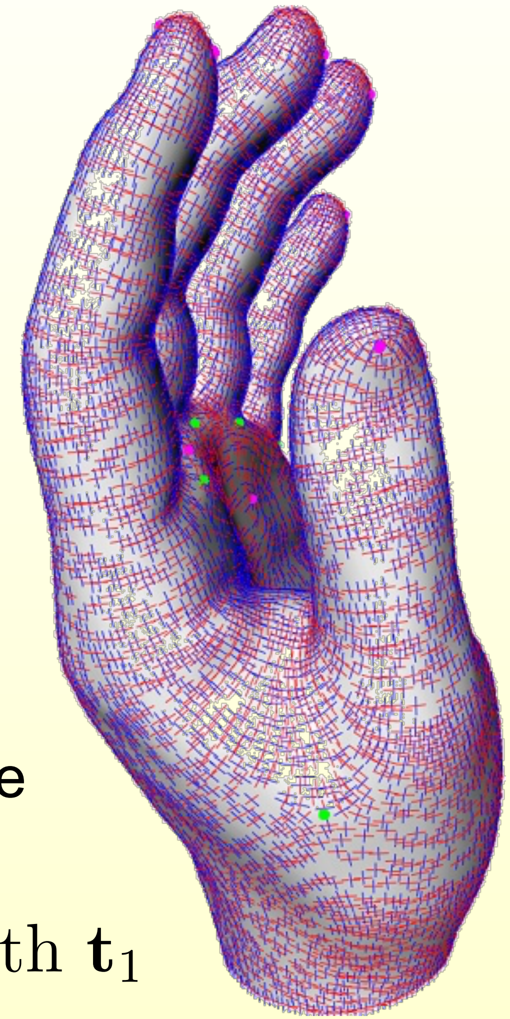
- Principal directions: tangent vectors corresponding to φ_{max} and φ_{min}





Principal Directions





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

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

Laplace-Beltrami and Curvature

- Apply operator to coordinate functions

Laplace-Beltrami

gradient operator

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

coordinate functions on surface M

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

divergence operator

Laplace-Beltrami and Curvature

- Apply operator to coordinate functions

Laplace-Beltrami

gradient operator

mean curvature

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

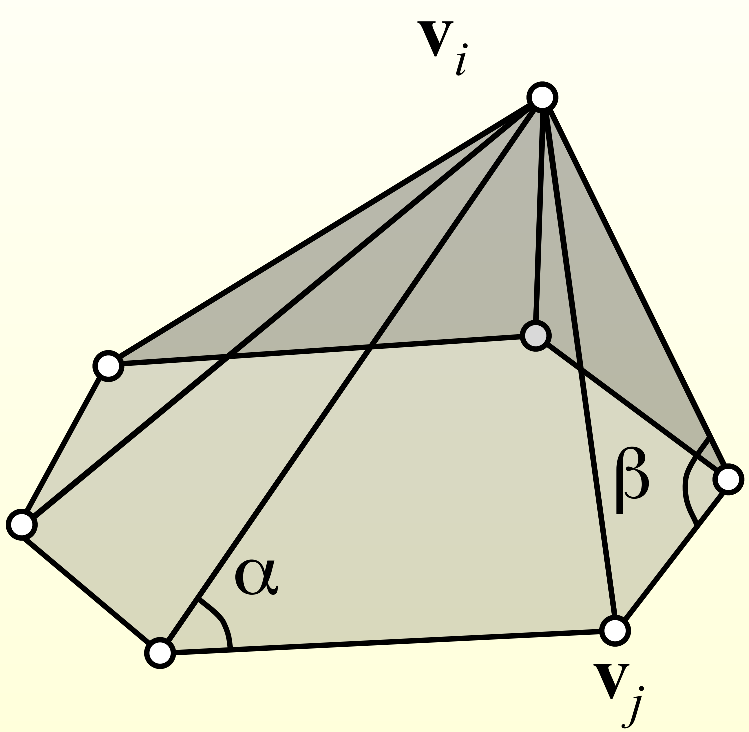
coordinate functions on surface M

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

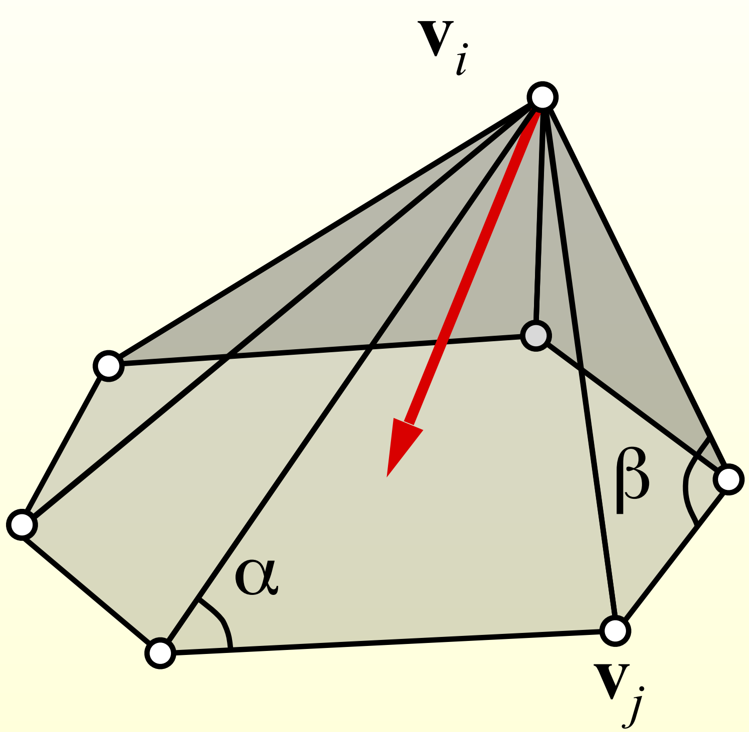
divergence operator

unit surface normal

Effect of the Discretization

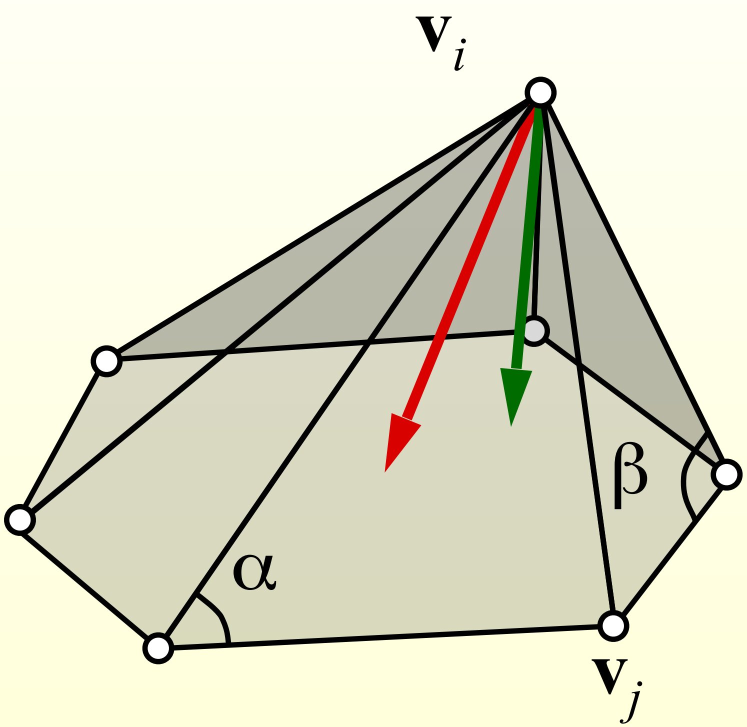


Effect of the Discretization



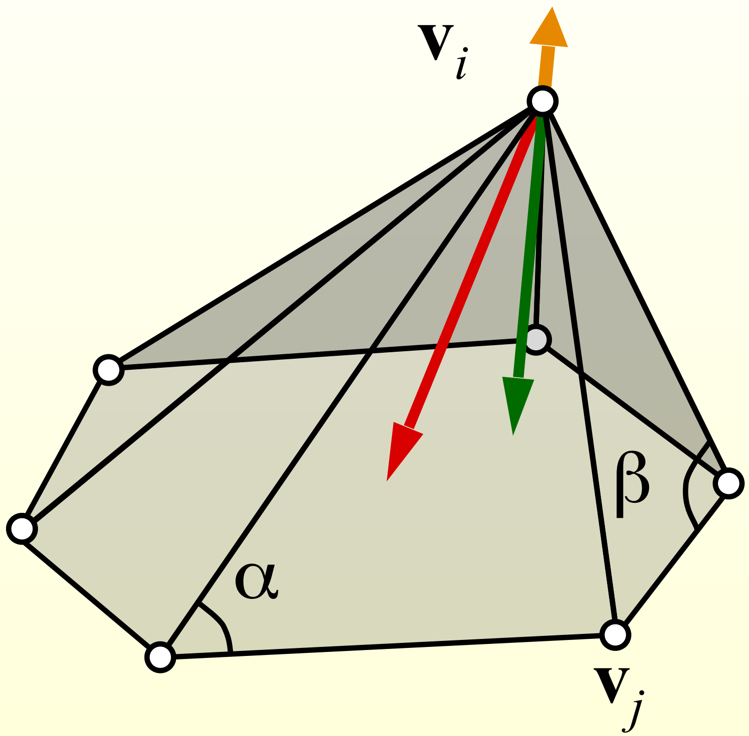
- **Uniform Laplacian $\mathbf{L}_u(\mathbf{v}_i)$**

Effect of the Discretization



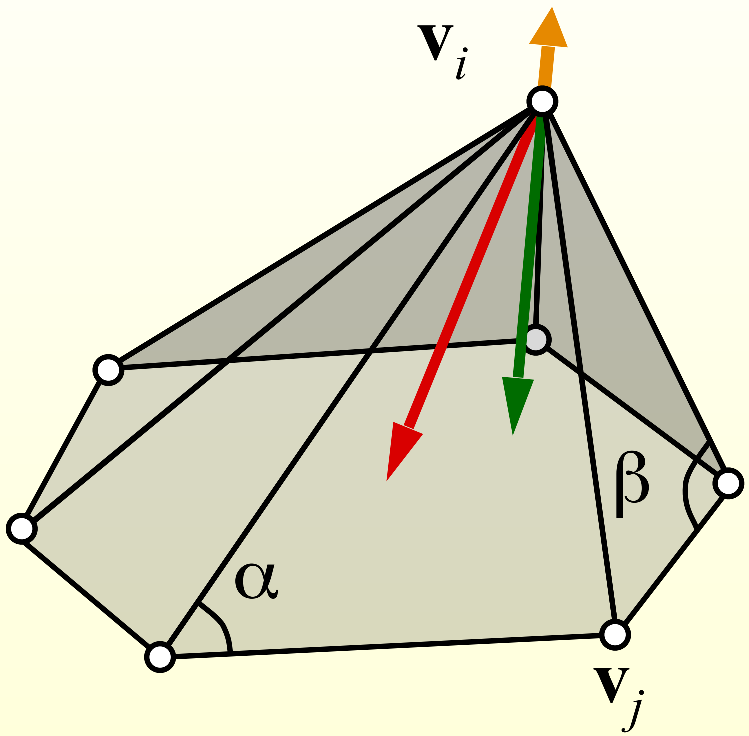
- Uniform Laplacian $\mathbf{L}_u(\mathbf{v}_i)$
- Cotangent Laplacian $\mathbf{L}_c(\mathbf{v}_i)$

Effect of the Discretization



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

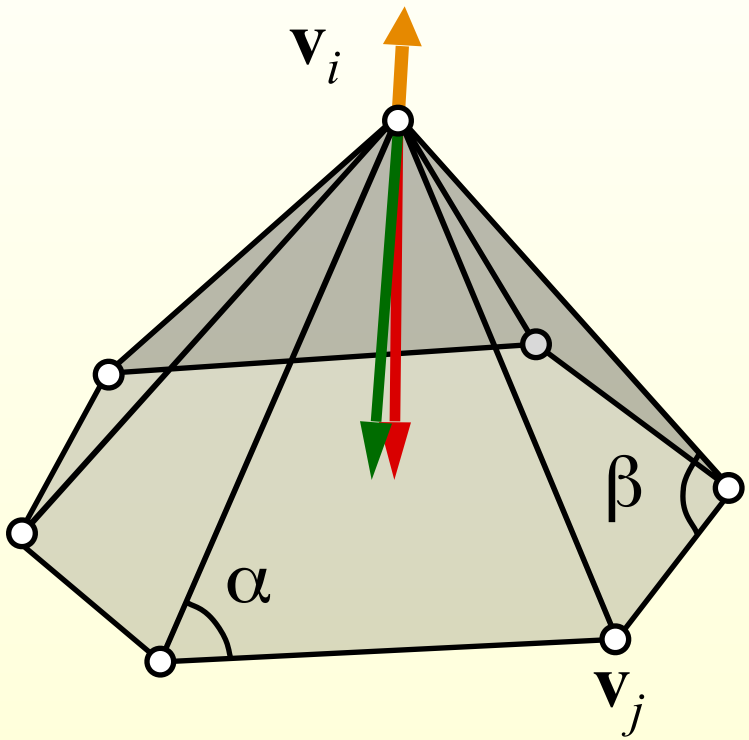
Effect of the Discretization



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

- **For nearly equal edge lengths**
  Uniform ≈ Cotangent

Effect of the Discretization



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

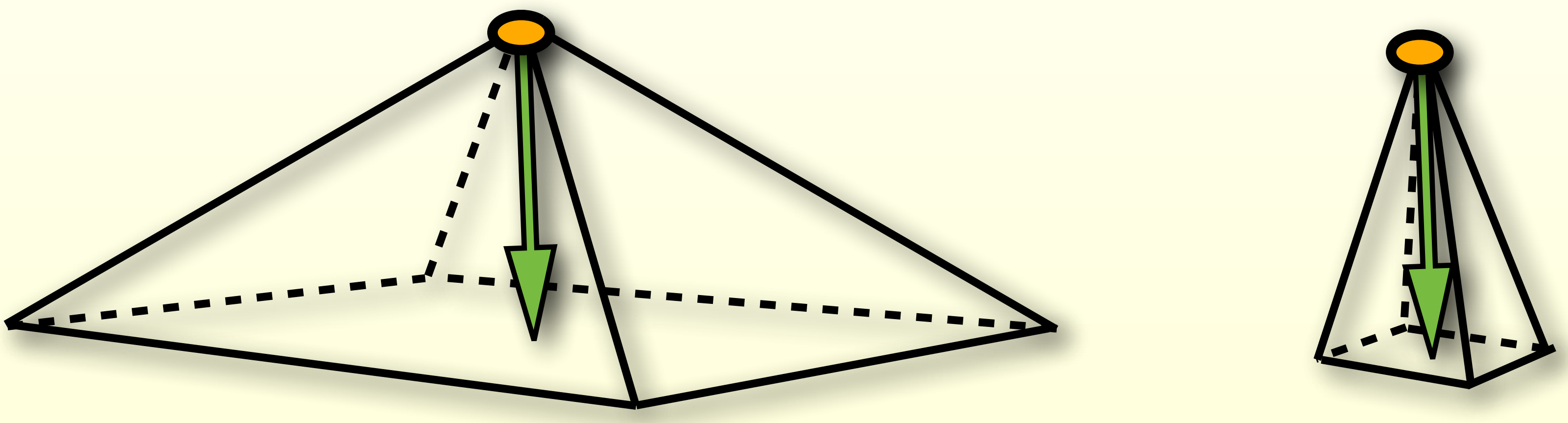
- **For nearly equal edge lengths**
  Uniform ≈ Cotangent

Cotan Laplacian allows computing discrete normal

Nice property: gives zero for planar 1-rings!

Effect of the Discretization

Uniform Laplacian: Frequency Mixup!

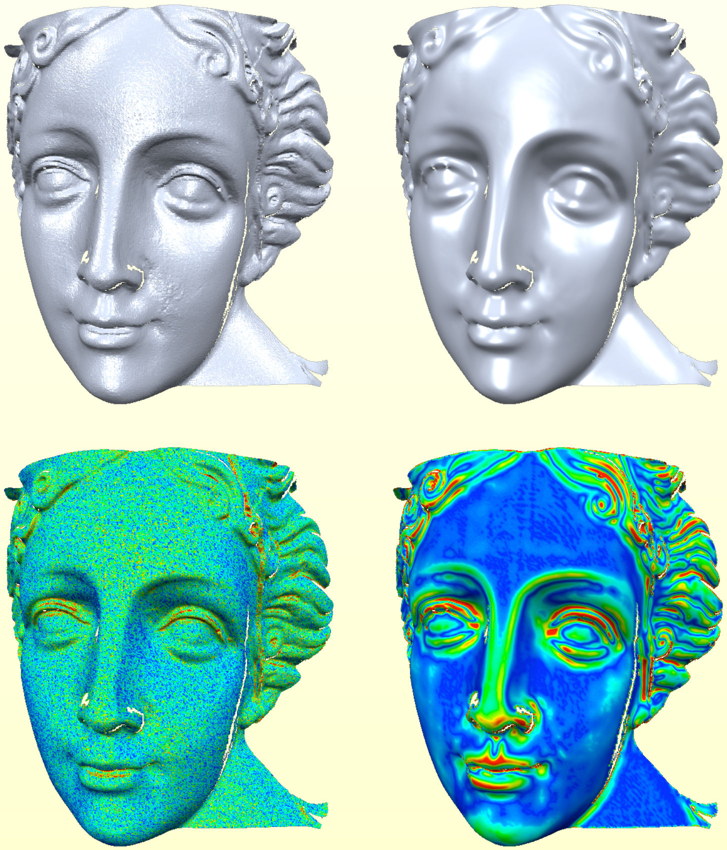


How to use curvature relation for smoothing?

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

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

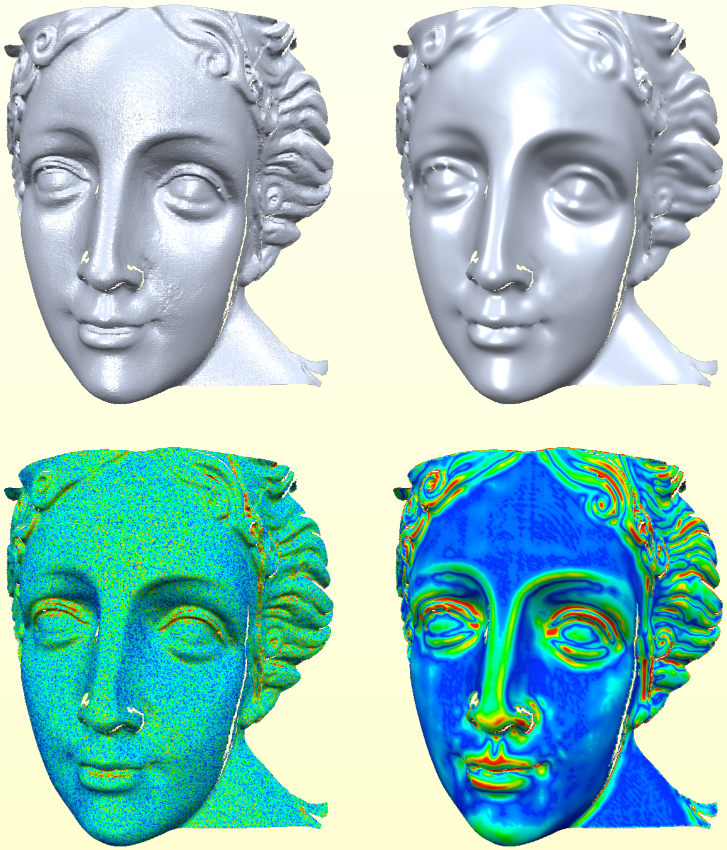
- Smooth H, obtain $\tilde{H}$
- Find a surface that has $\tilde{H}$ as mean curvature
  - H doesn't define the surface
  - $\mathbf{n}$ nonlinear in $\mathbf{p}$



How to use curvature relation for smoothing?

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

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



- Another idea:
  - Keep the old $\mathbf{n}$
  - "Flow" along $\mathbf{n}$ to decrease H

Diffusion Flow on Height Fields

- Diffusion equation

diffusion constant

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

Laplace operator

Diffusion flow on Meshes

Diffusion flow on Meshes

- Model smoothing as a diffusion process

Diffusion flow on Meshes

- Model smoothing as a diffusion process

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

Diffusion flow on Meshes

- Model smoothing as a diffusion process

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

- Discretize in time, forward differences:

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

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

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

Diffusion flow on Meshes

- Model smoothing as a diffusion process

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

- Discretize in time, forward differences:

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

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

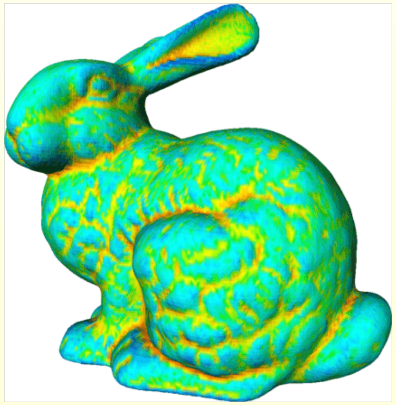
$$\boxed{\mathbf{p}^{(n+1)} = (I + dt\, \lambda L) \mathbf{p}^{(n)}}$$

Explicit integration! Unstable unless time step dt is small

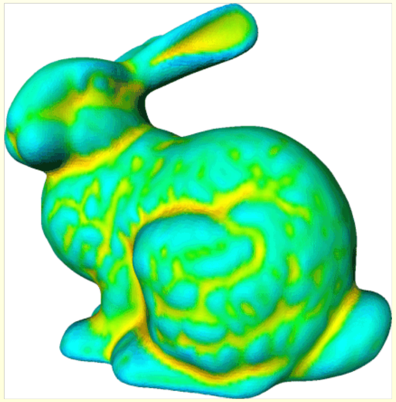
Diffusion Flow on Meshes

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \lambda \Delta \mathbf{p}_i$$

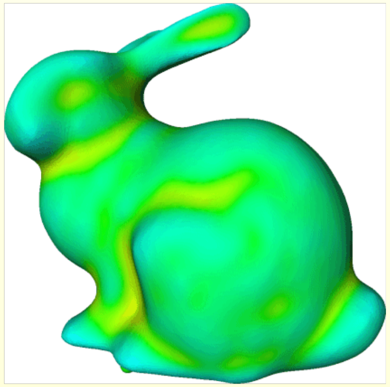
- Iterate



0 Iterations



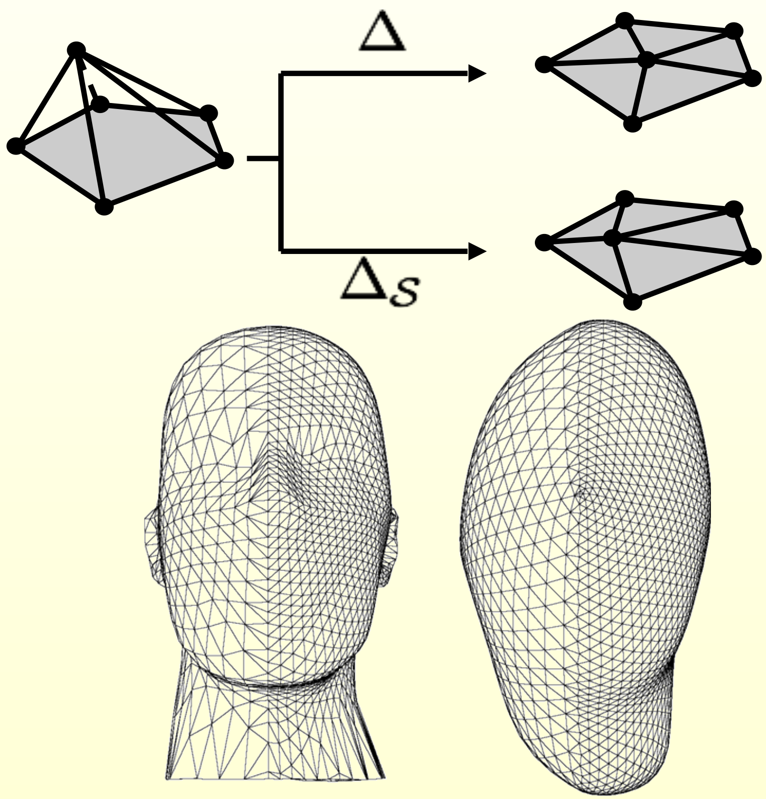
5 Iterations



20 Iterations

Effect of Laplace Discretization

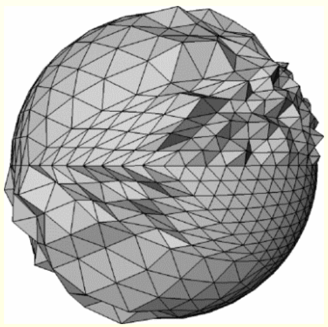
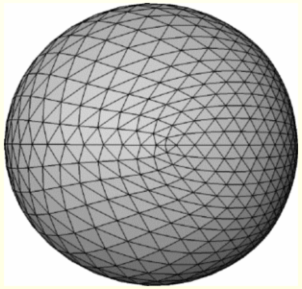
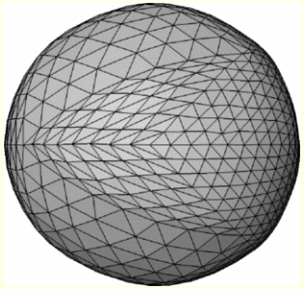
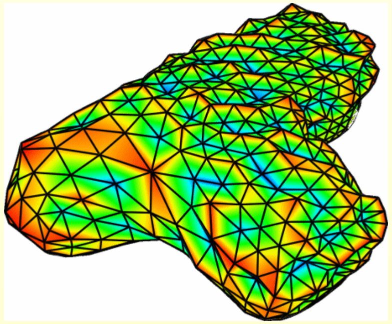
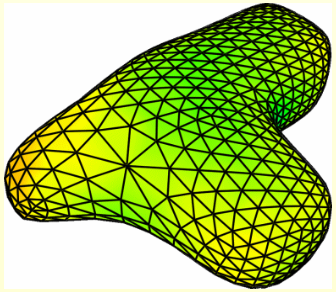
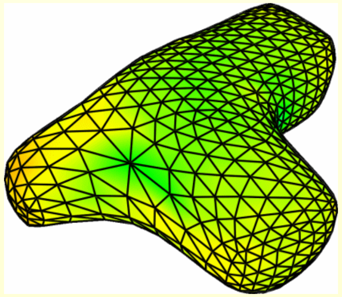
- Uniform Laplace smooths geometry **and** triangulation
- Can be non-zero even for planar triangulations
- Vertex drift can lead to distortions
- Might be desired for mesh regularization



Desbrun et al., Siggraph 1999

Comparison

Original | Uniform Laplace | Laplace-Beltrami

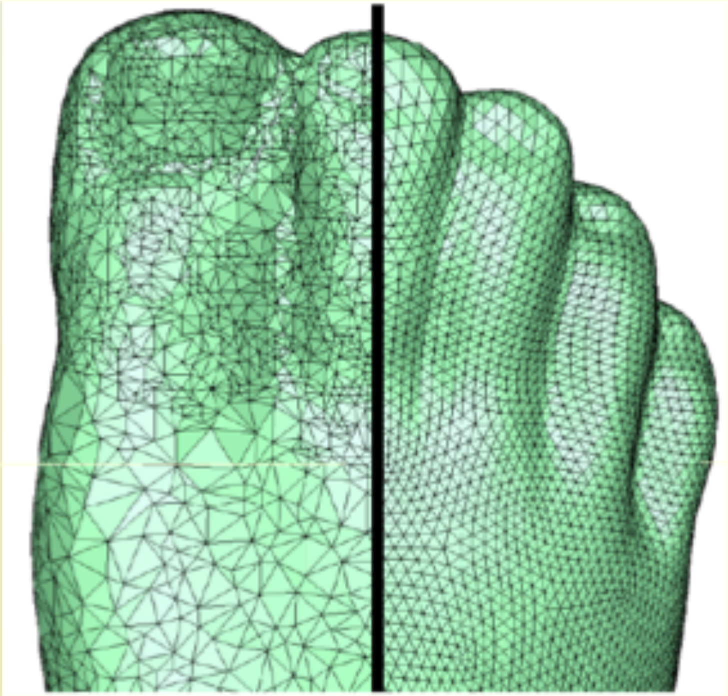
Remeshing

Remeshing

- Improved geometry (positions)
  - How about connectivity?

Remeshing

- Improved geometry (positions)
  - How about connectivity?
- Remeshing: Given a 3D mesh, improve its triangulation while preserving its geometry.

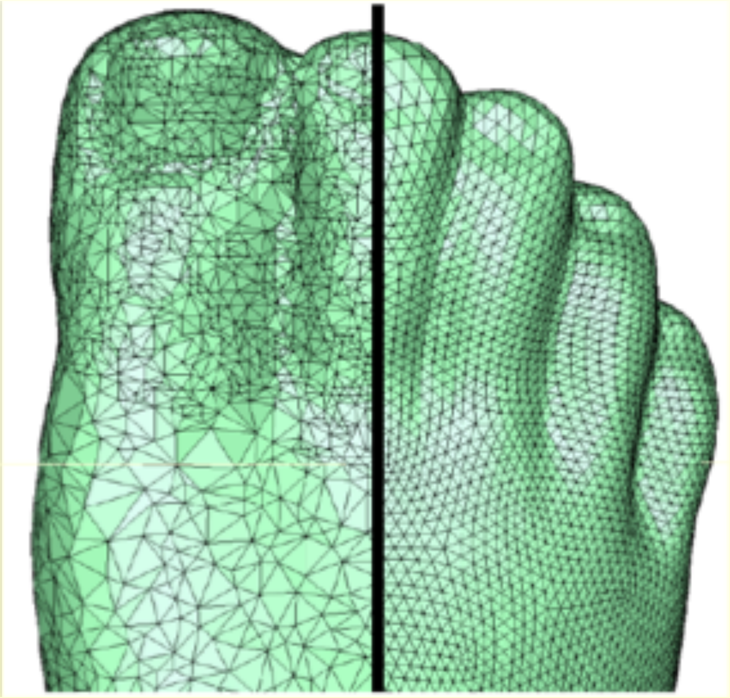


Remeshing

- Improved geometry (positions)
  - How about connectivity?
- Remeshing: Given a 3D mesh, improve its triangulation while preserving its geometry.

Meshing Quality Checklist

- Equal edge lengths
- Equilateral triangles
- Valence close to 6
- Uniform vs. adaptive sampling
- Feature preservation
- Alignment to curvature lines
- Isotropic vs. anisotropic
- Triangles vs. quadrangles



Two Fundamental Approaches

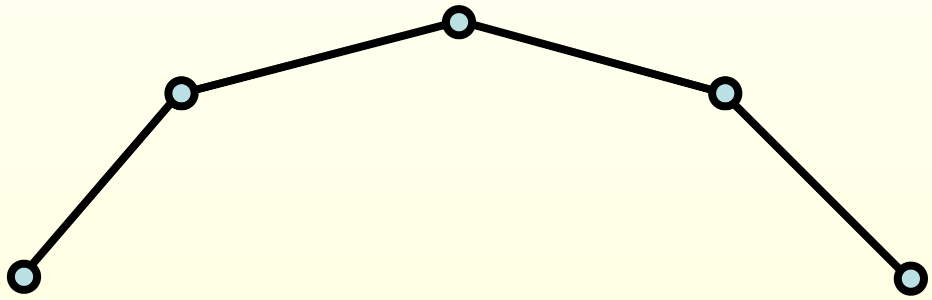
Two Fundamental Approaches

- Parametrization based
  - map to 2D domain / 2D problem
  - computationally more expensive
  - works even for coarse resolution remeshing

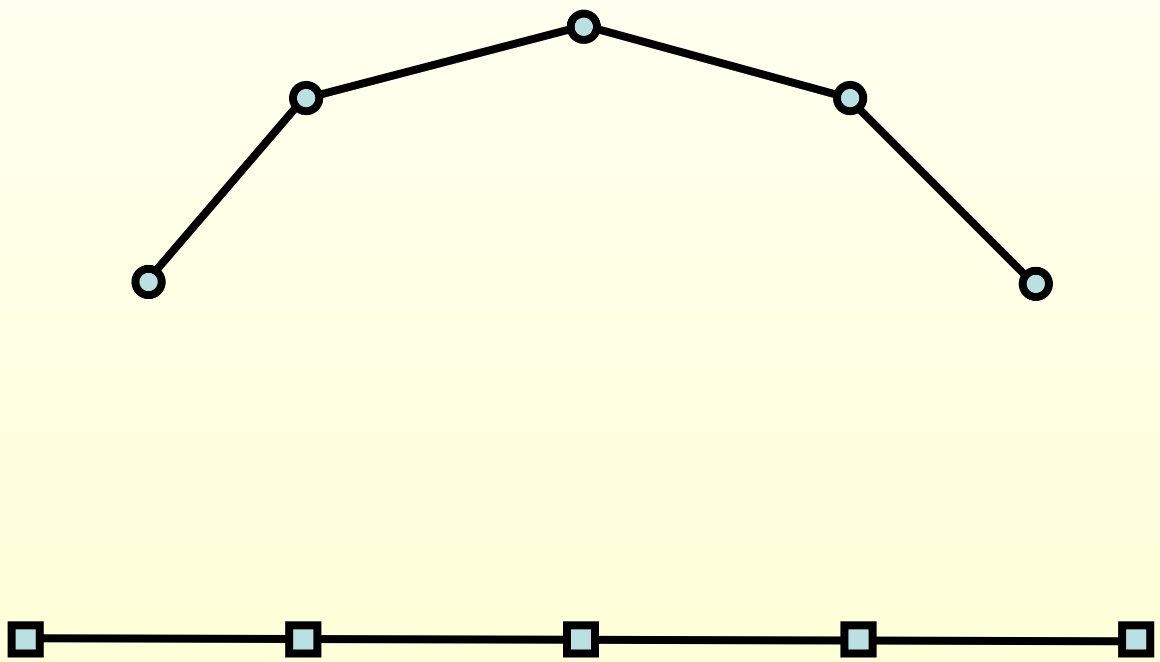
Two Fundamental Approaches

- Parametrization based
  - map to 2D domain / 2D problem
  - computationally more expensive
  - works even for coarse resolution remeshing
- Surface oriented
  - operate directly of the surface
  - treat surface as a set of points / polygons in space
  - efficient for high resolution remeshing

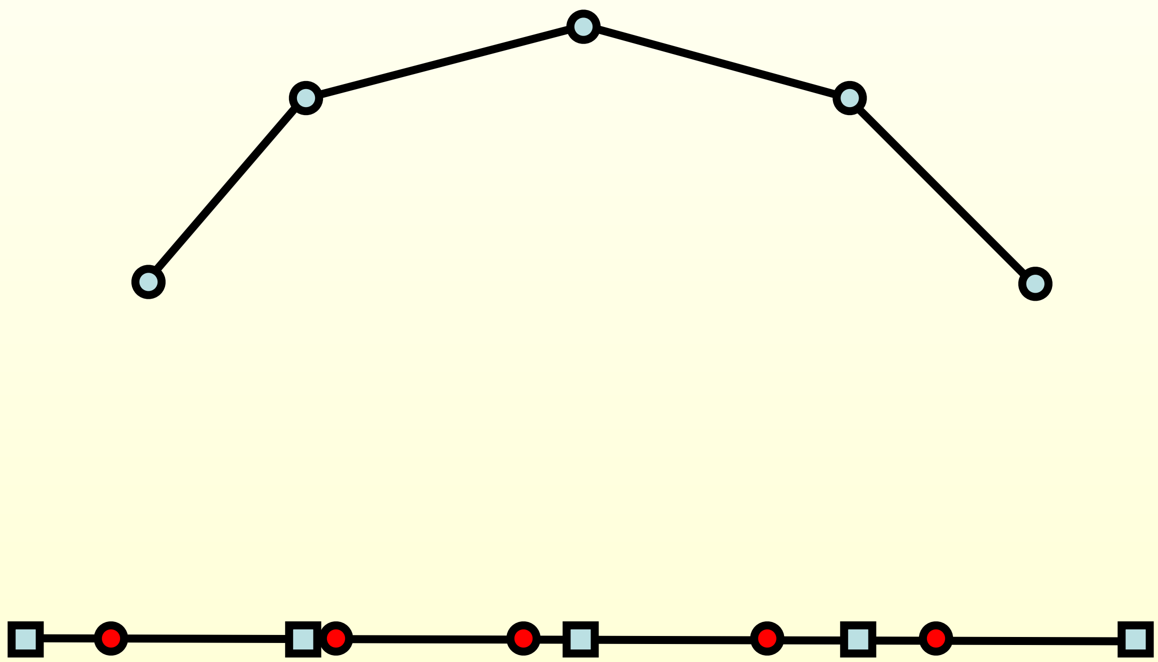
Parametrization Based



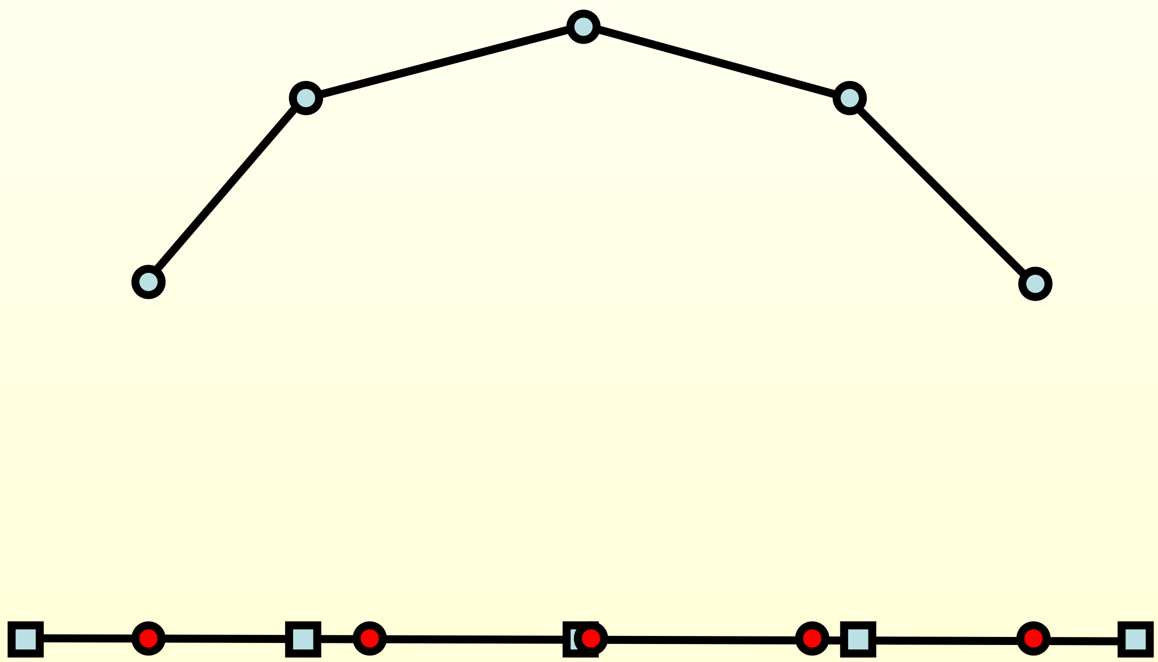
Parametrization Based



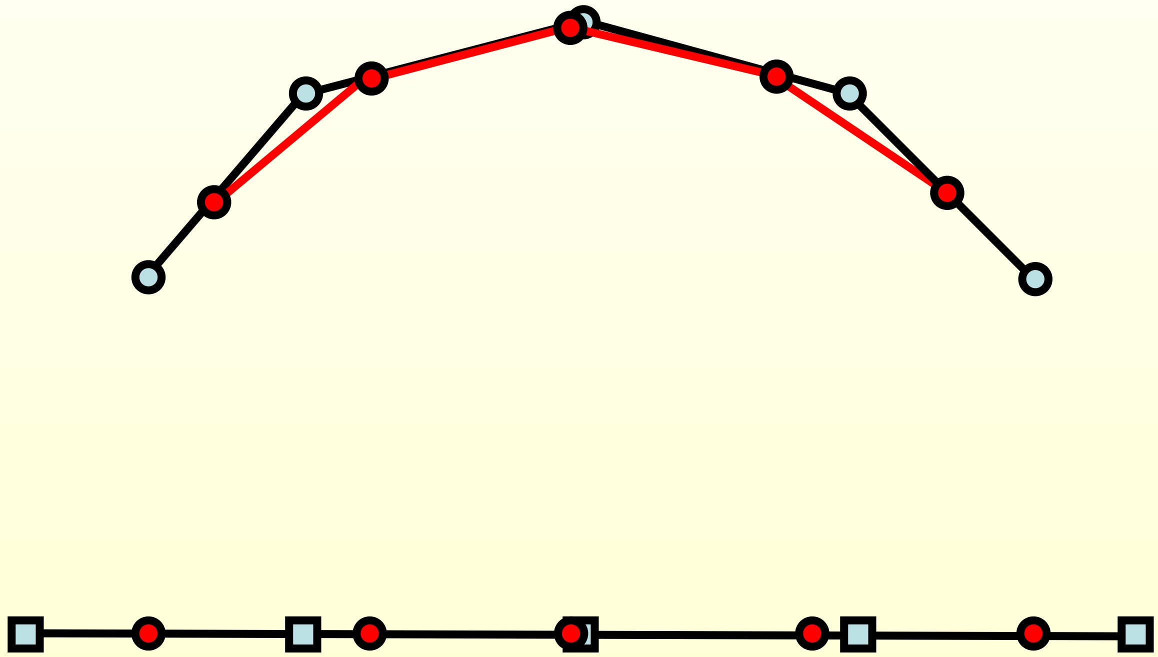
Parametrization Based



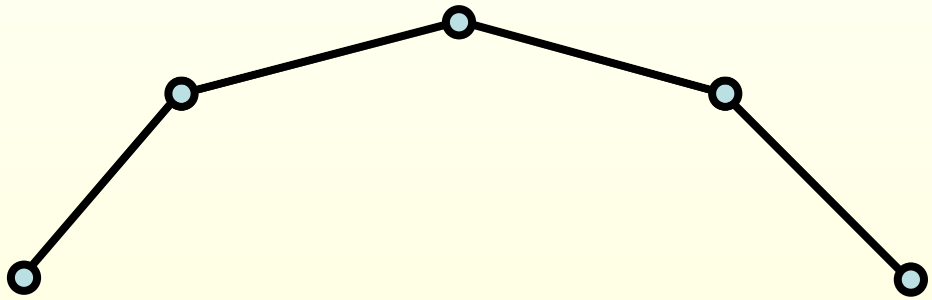
Parametrization Based



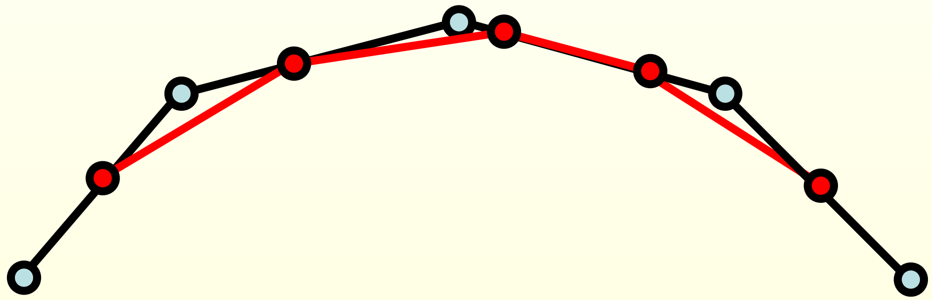
Parametrization Based



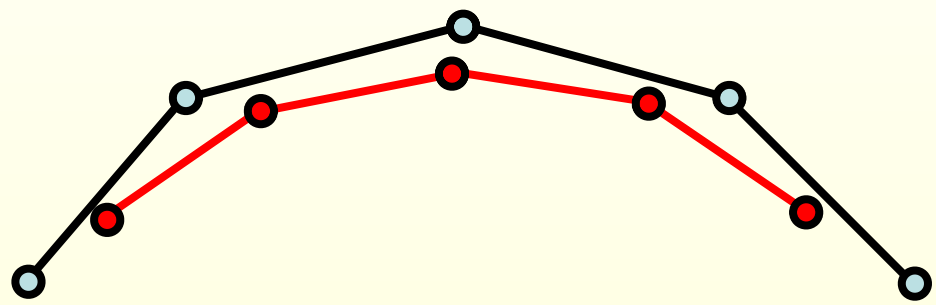
Surface Oriented



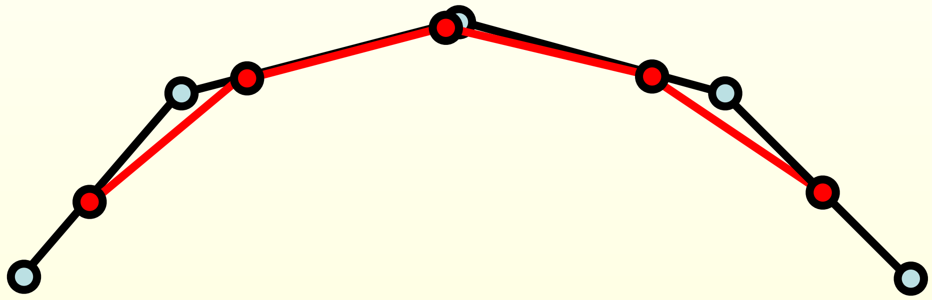
Surface Oriented



Surface Oriented



Surface Oriented

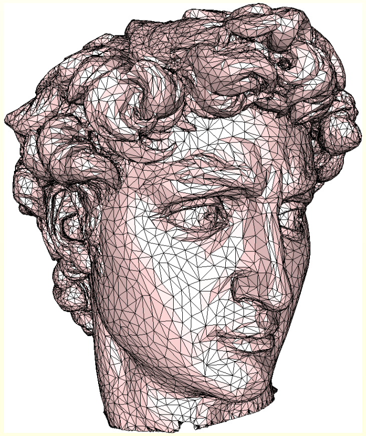


Parameterization-Based Approach

- Motivation: 2D remeshing is much easier
  - Sample distribution
  - Delaunay triangulation
  - Centroidal Voronoi diagram

- Which parameterization method to choose?
  - Next time!

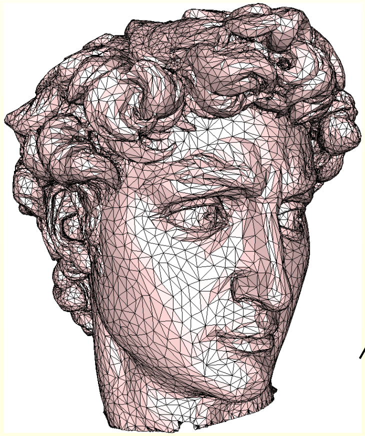
Parameterization- Based Isotropic Remeshing

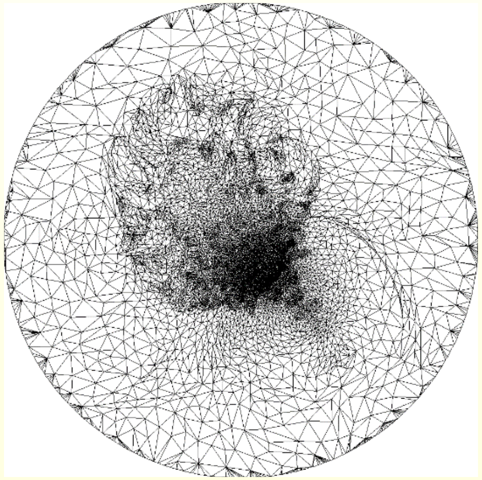


Alliez et al. 2002,

Interactive Geometry Remeshing

Parameterization- Based Isotropic Remeshing

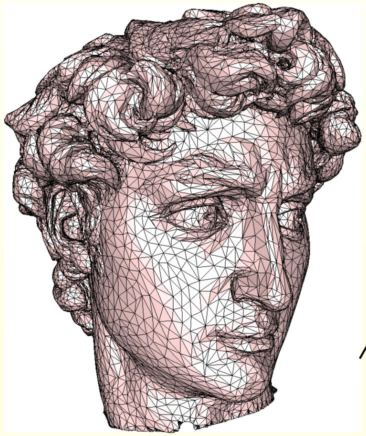


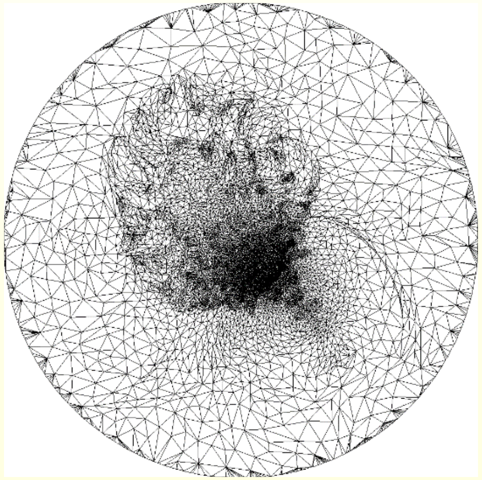


Alliez et al. 2002,

Interactive Geometry Remeshing

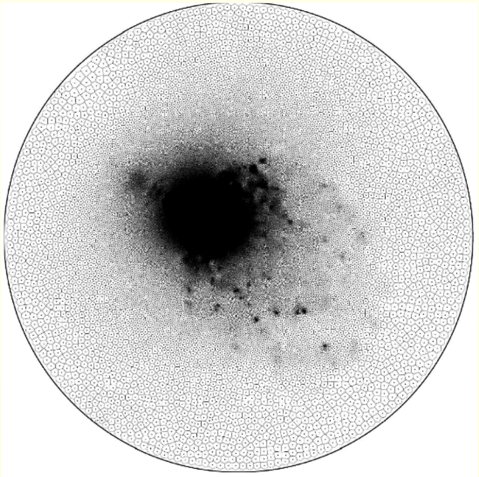
Parameterization- Based Isotropic Remeshing





Alliez et al. 2002,

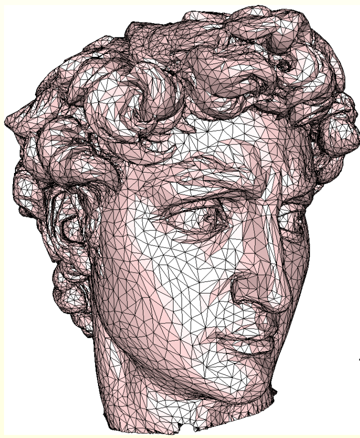
Interactive Geometry Remeshing

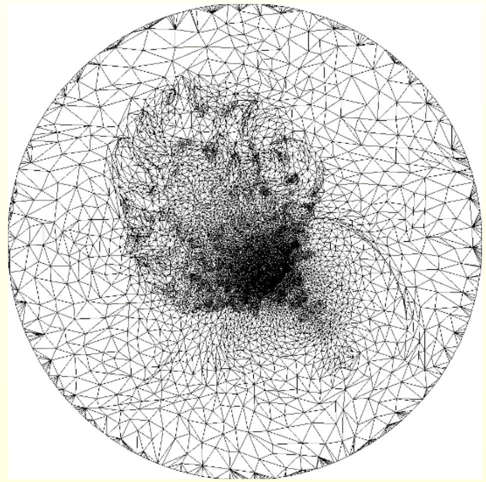


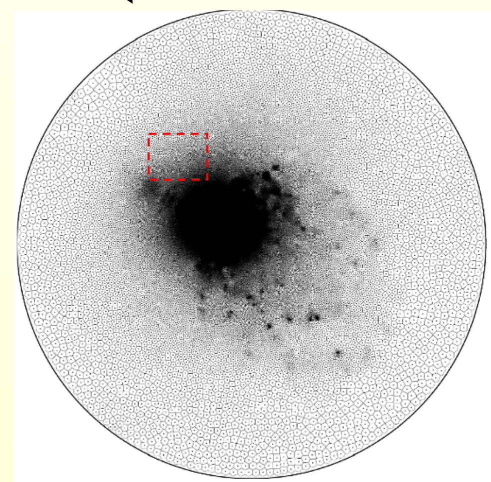
Parameterization- Based Isotropic Remeshing

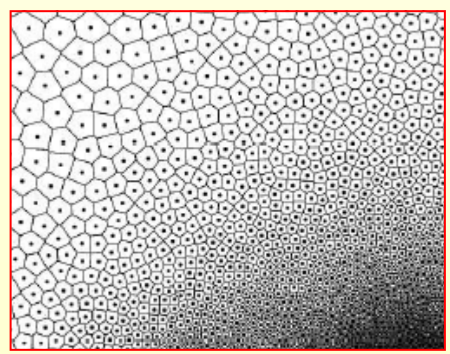
Alliez et al. 2002,

Interactive Geometry Remeshing







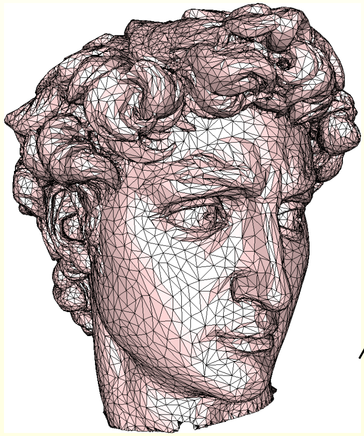


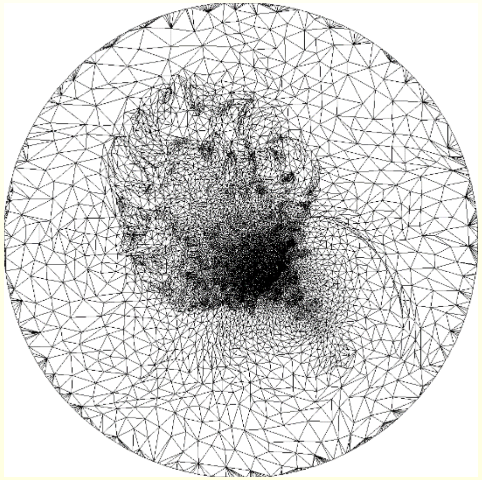
Weighted Centroidal Voronoi tessellation

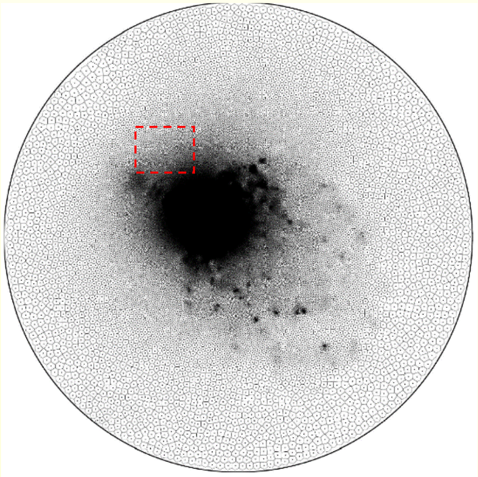
Parameterization- Based Isotropic Remeshing

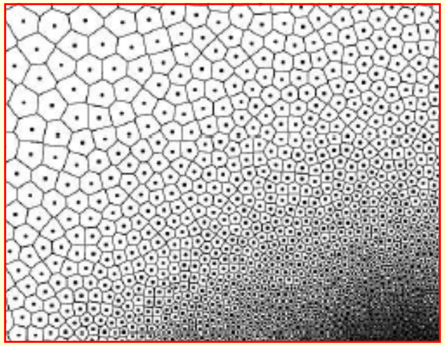
Alliez et al. 2002,

Interactive Geometry Remeshing





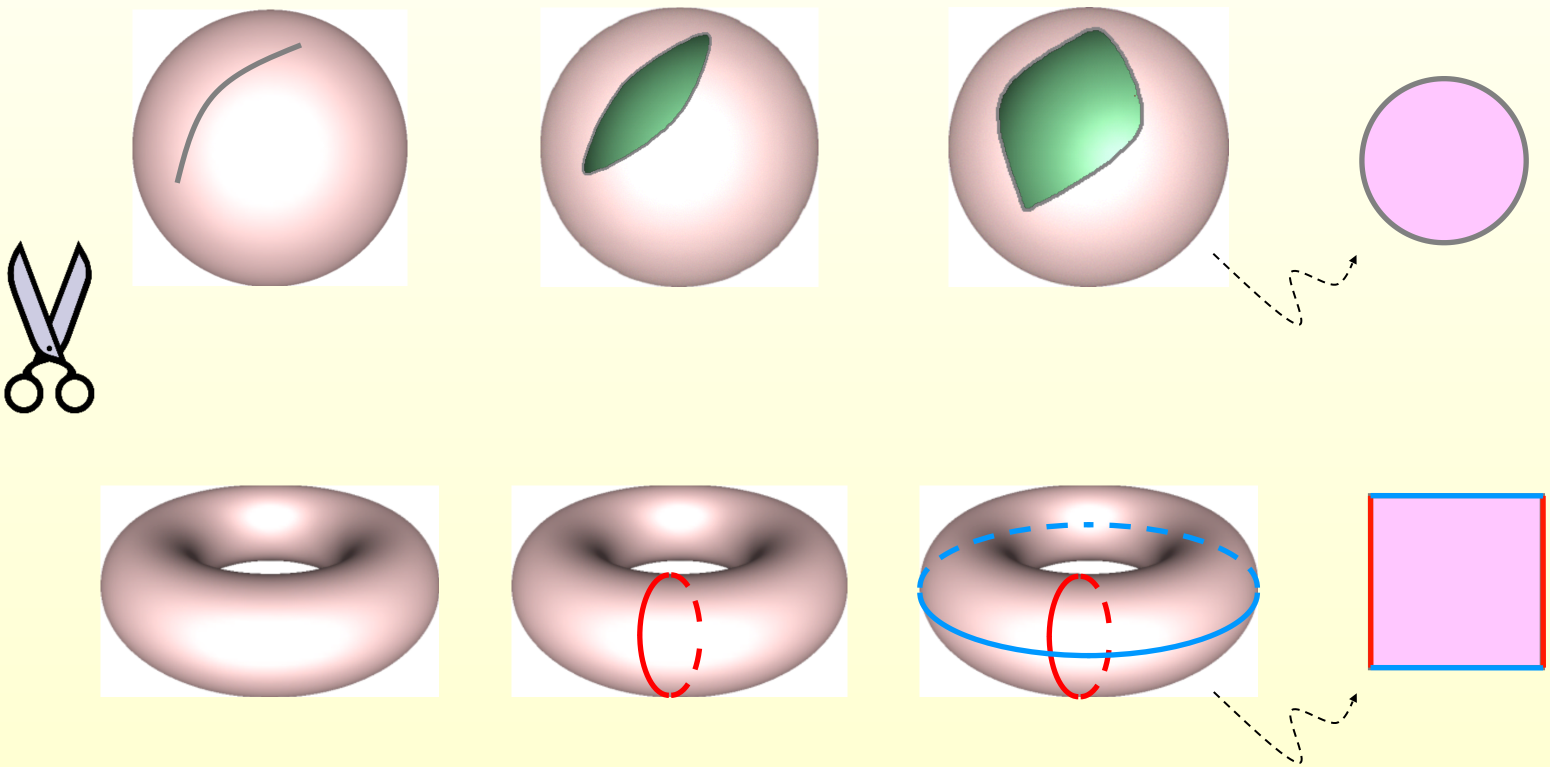




Weighted Centroidal Voronoi tessellation

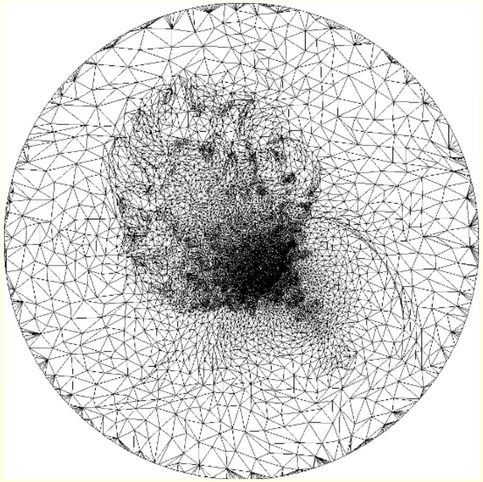
Need disk-like topology

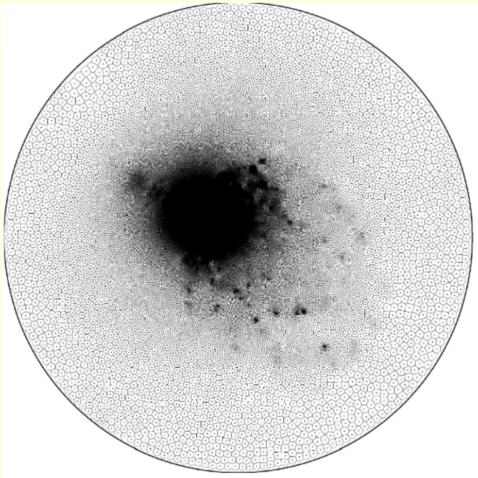
- Introduce cuts on the mesh



Distortion-Based Sampling

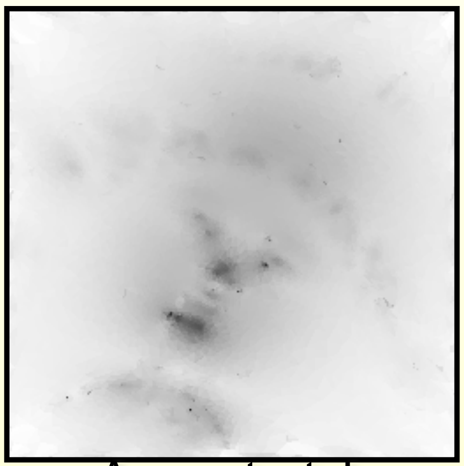
- Randomly sample triangles
  - Weighted by area and density
  - Density: curvature or user-defined sizing field
- Compensate area distortion when sampling in the parameter domain
  - Distortion = 3D area / 2D area





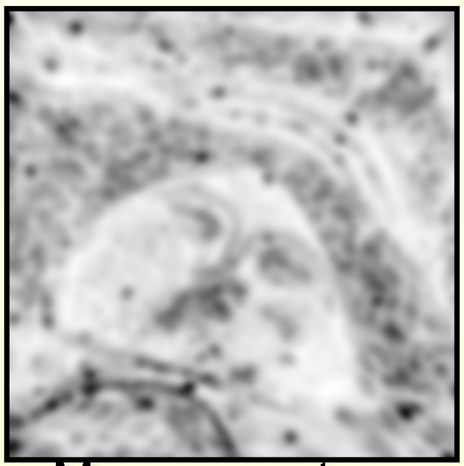
Distortion-Based Sampling

- Compose importance map



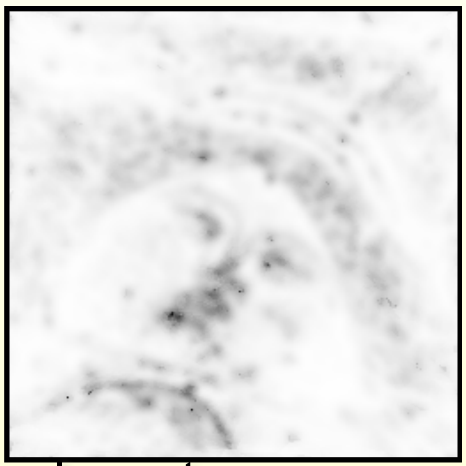
Area stretch

$\cdot$



Mean curvature

$=$

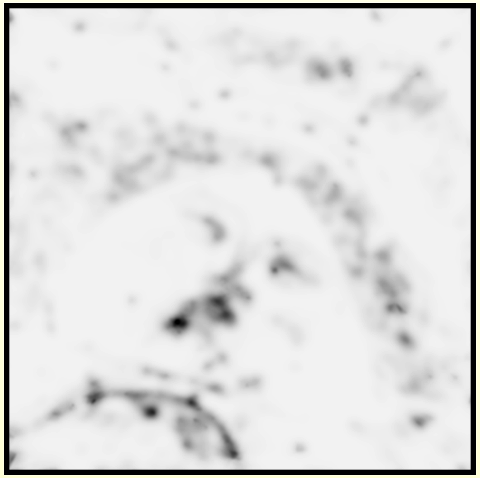


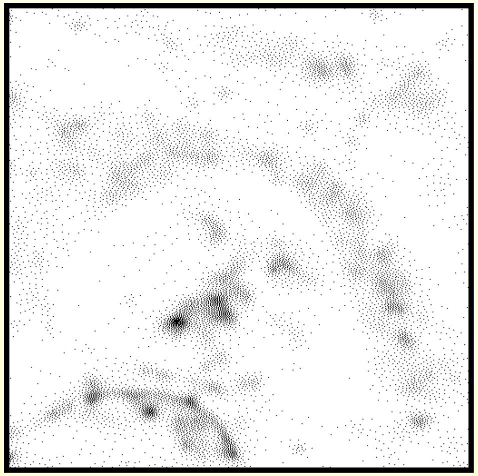
Importance map

- At parameterization time: Keep track of where each point/triangle lands!

Distortion-Based Sampling

- 2D error diffusion on importance map
  - Half-toning, dithering

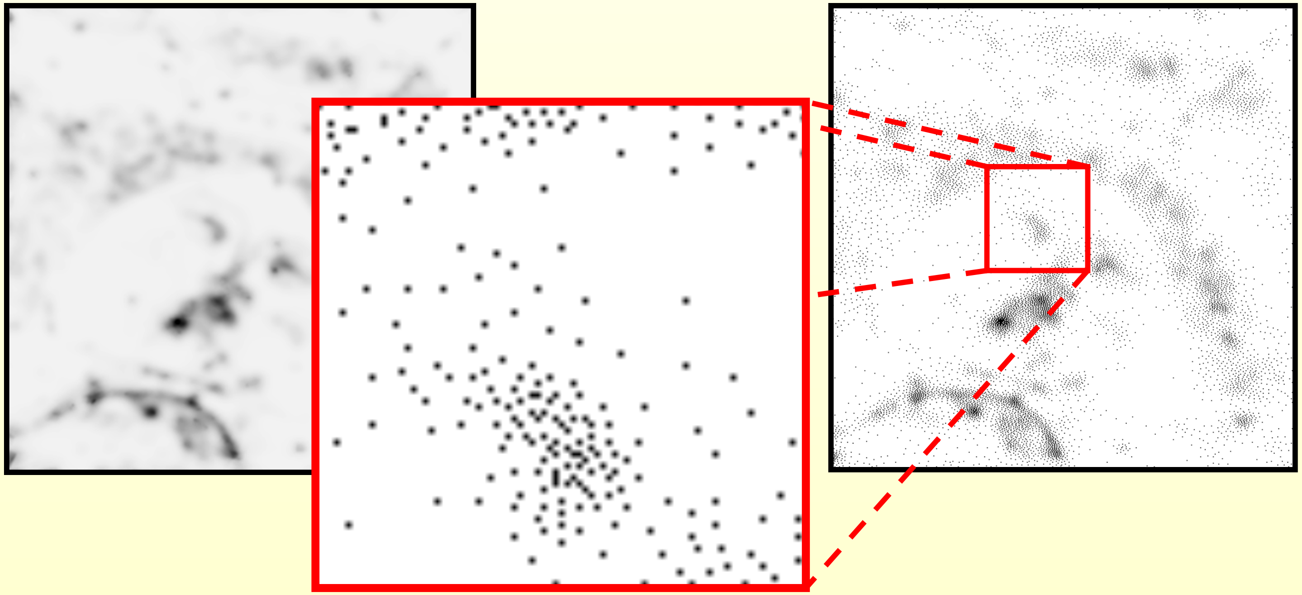




Floyd-Steinberg dithering

Distortion-Based Sampling

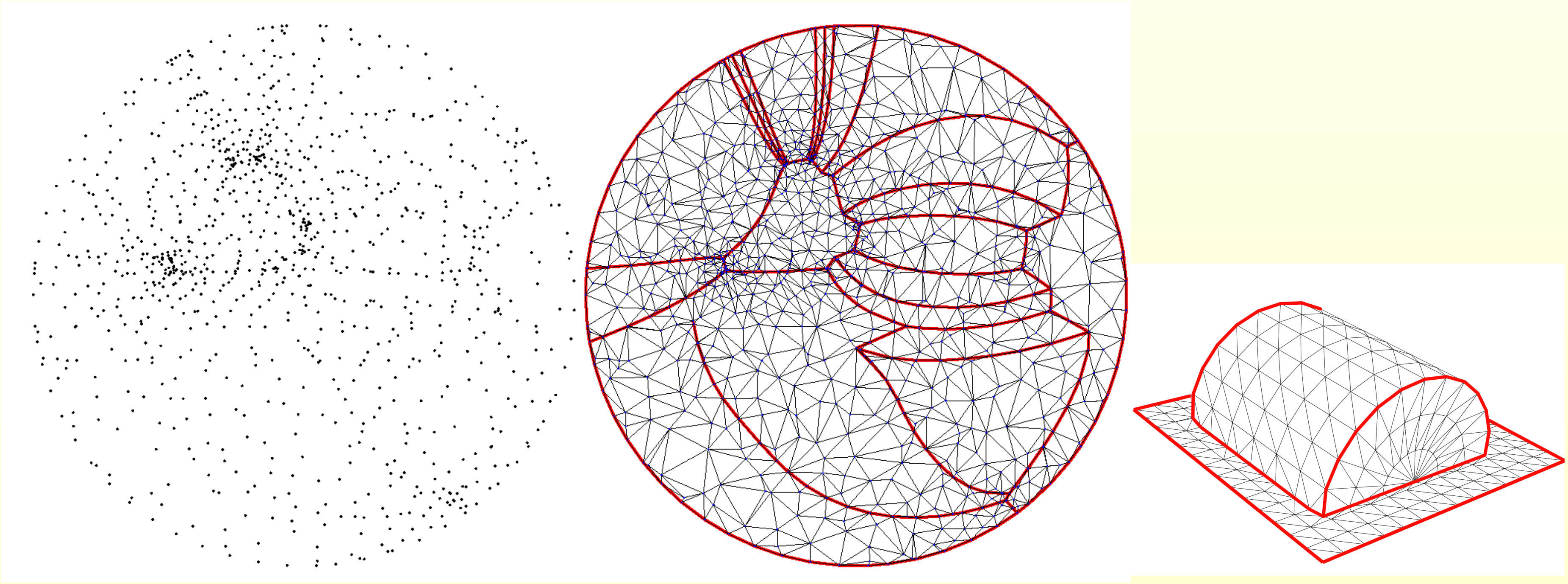
- 2D error diffusion on importance map
  - Half-toning, dithering



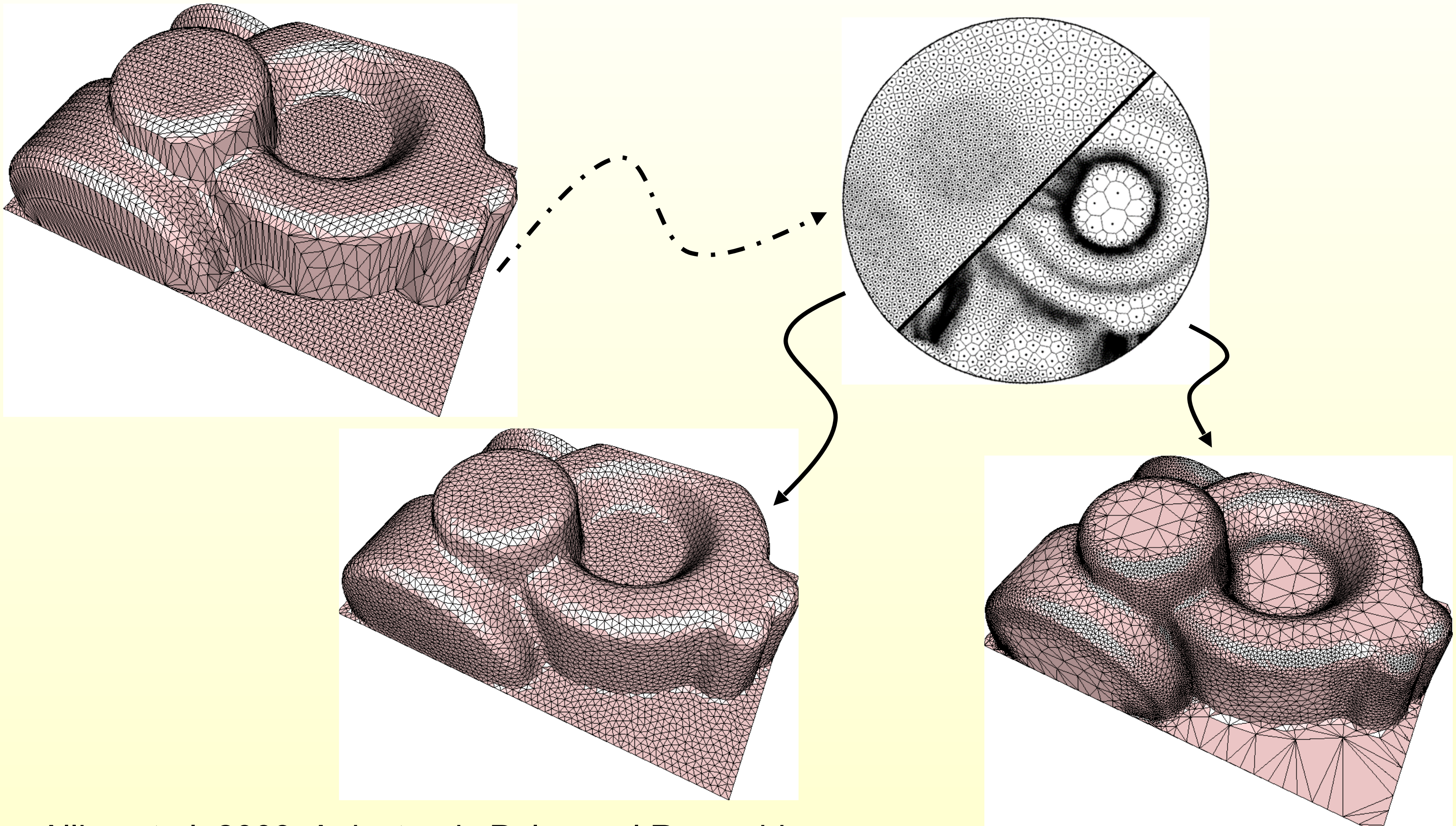
Floyd-Steinberg dithering

Connecting the samples

- 2D constrained Delaunay triangulation



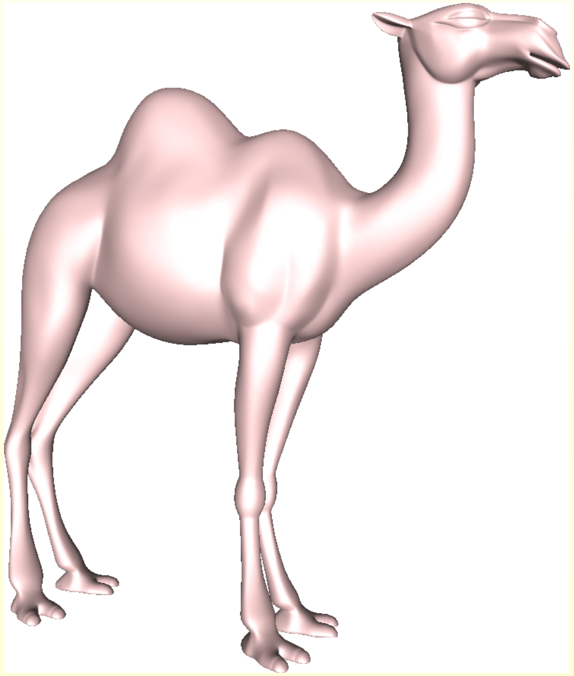
Uniform vs. Adaptive



Alliez et al. 2003, Anisotropic Polygonal Remeshing

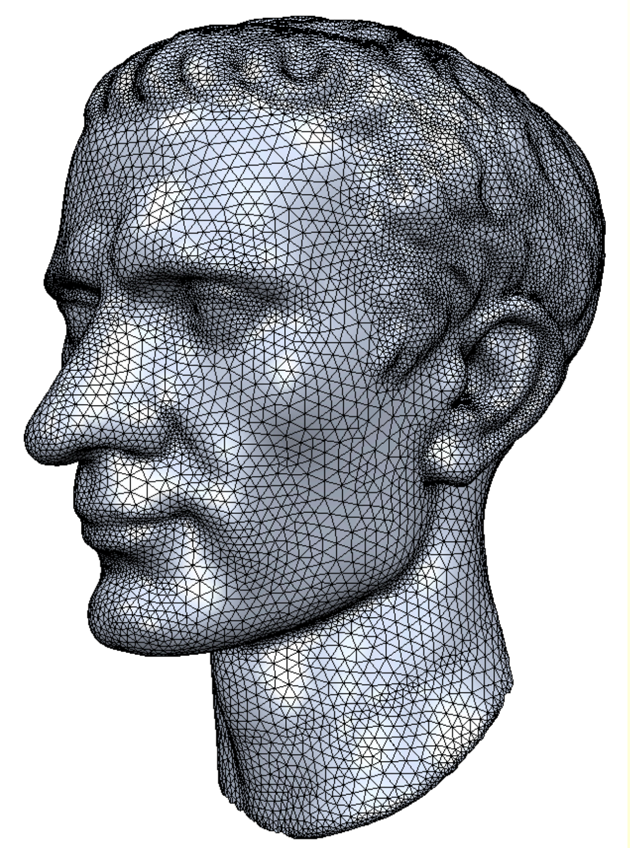
Limitations

- Closed genus 0
  - May need a good cut
  - Stitch seams afterwards
- Protruding legs
  - Sampling
  - Numerical problems



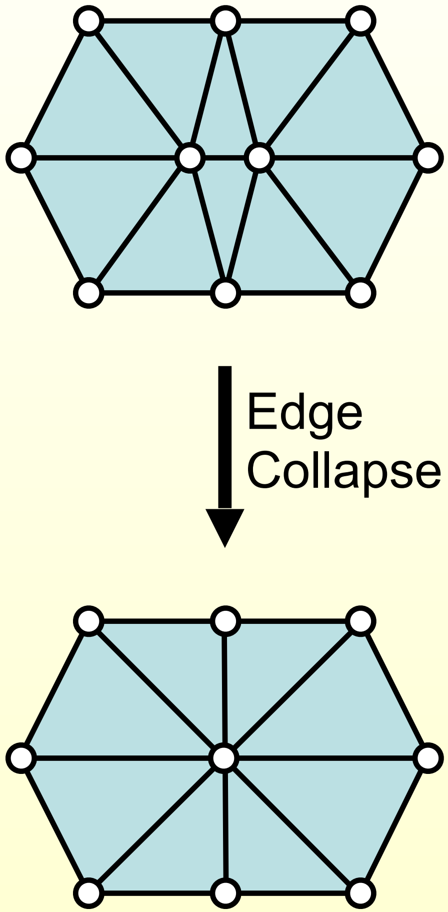
Direct Surface Remeshing

- Avoid global parametrization
  - Numerically very sensitive
  - Topological restrictions
- Use local operators & back-projections
  - Resampling of 100k triangles in < 5s

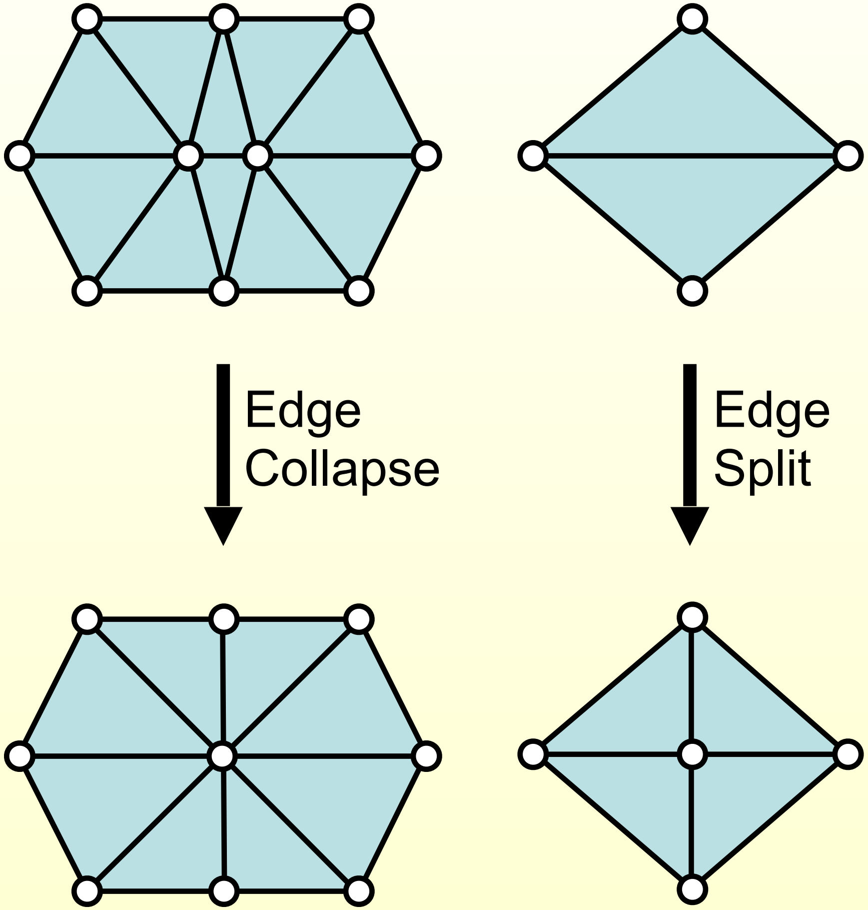


Botsch et al. 2004, “A Remeshing Approach to Multiresolution Modeling”

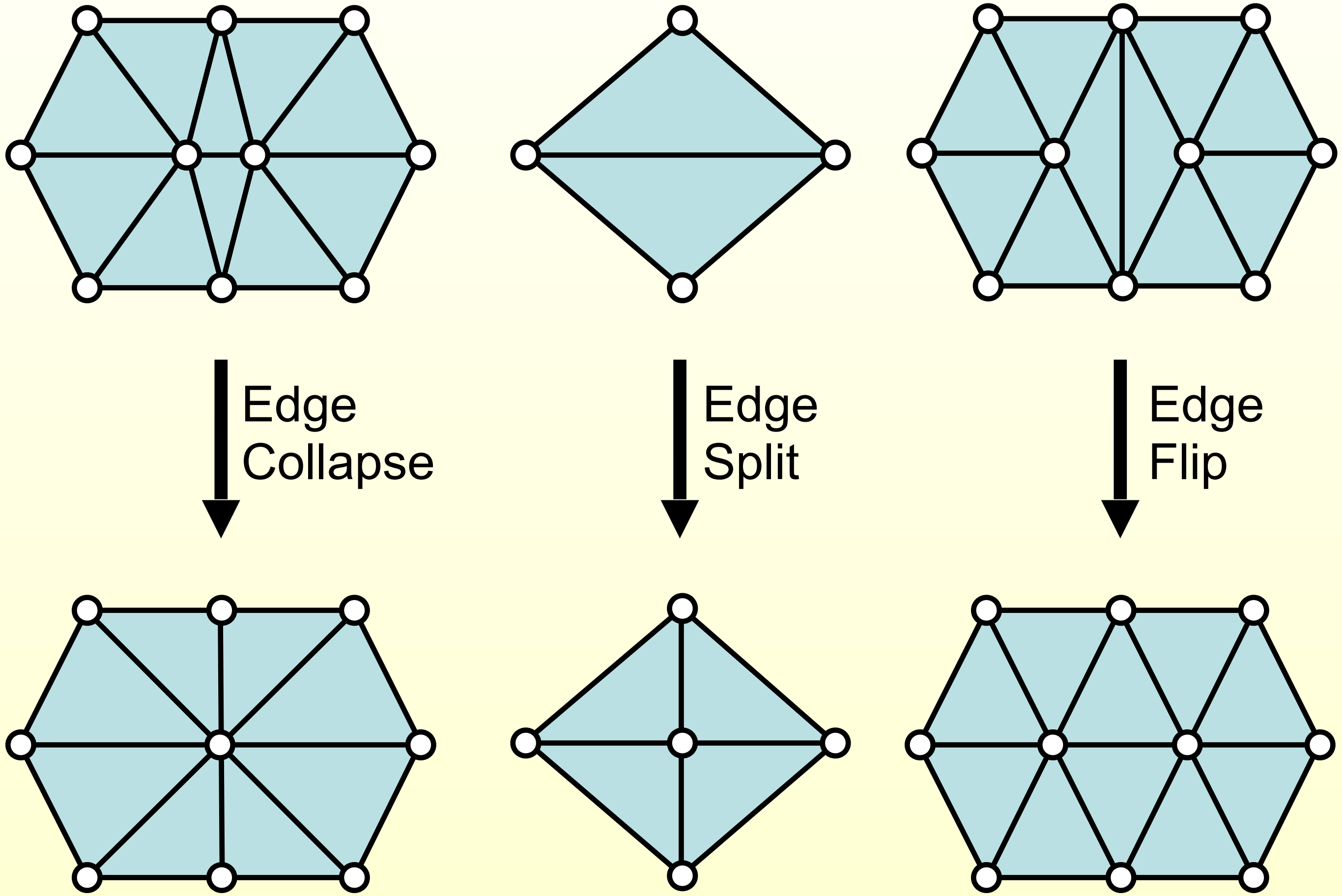
Local Remeshing Operators



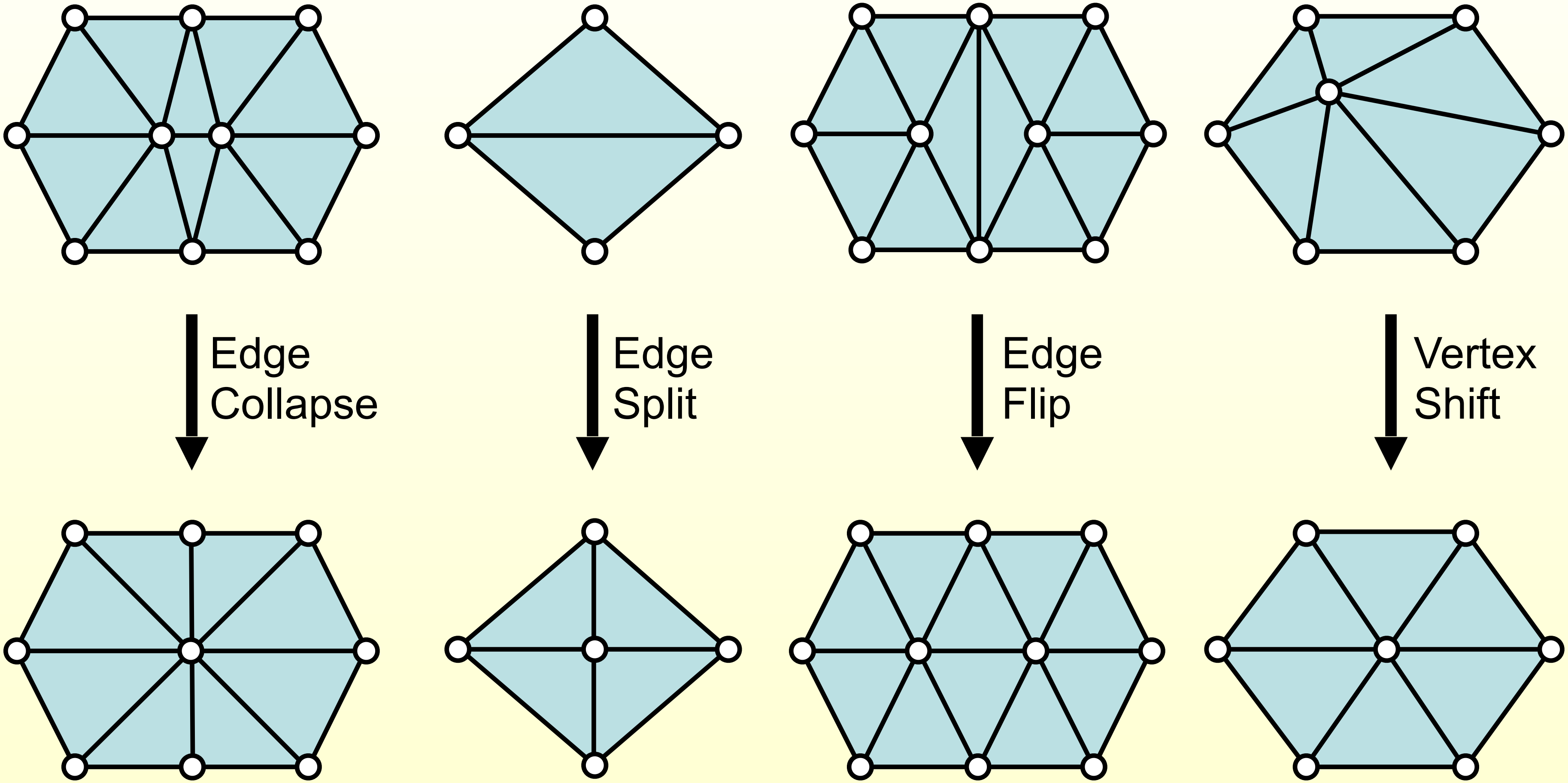
Local Remeshing Operators



Local Remeshing Operators



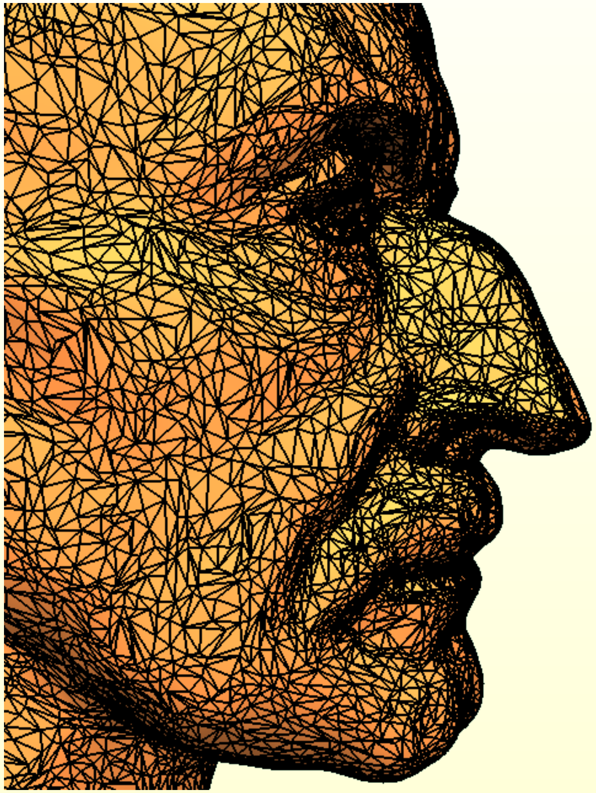
Local Remeshing Operators



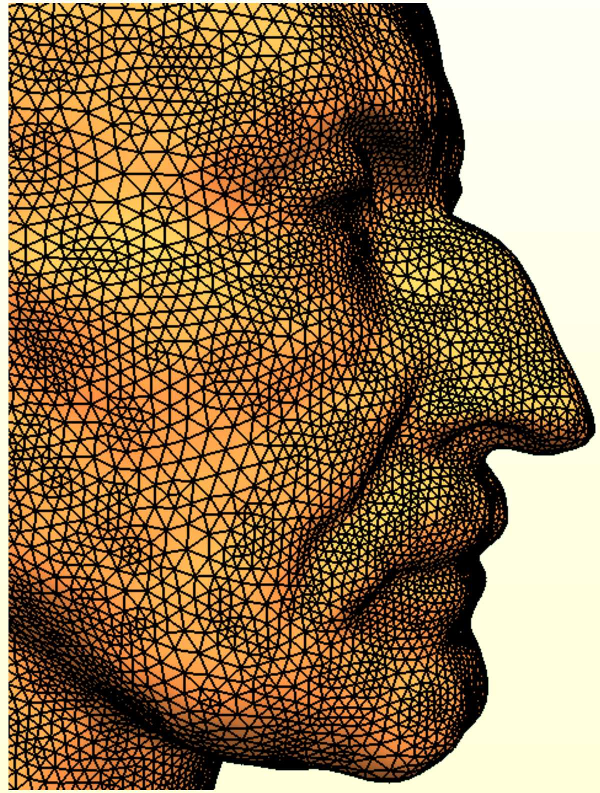
Isotropic Remeshing

- Specify target edge length L
- Iterate:
  1. Split edges longer than L_{max}
  2. Collapse edges shorter than L_{min}
  3. Flip edges to get closer to valence 6
  4. Vertex shift towards neighbor average by tangential relaxation
  5. Project vertices onto reference mesh

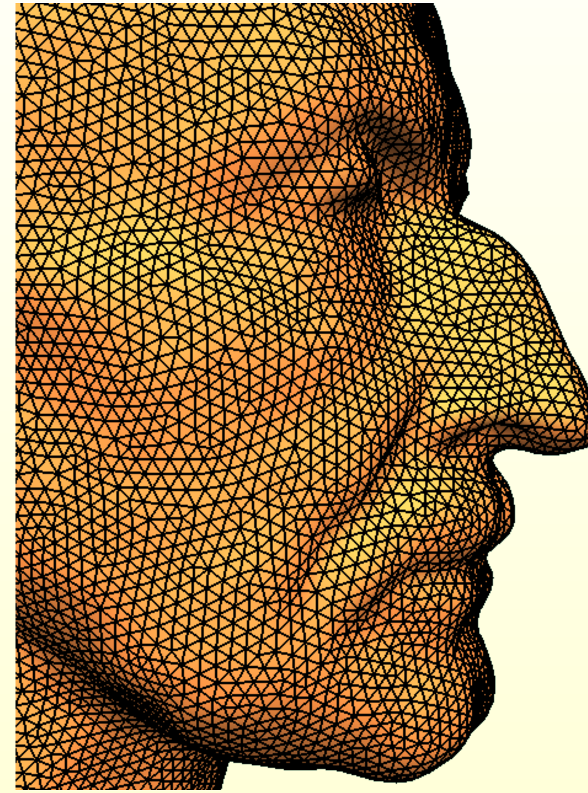
Remeshing Results



Original



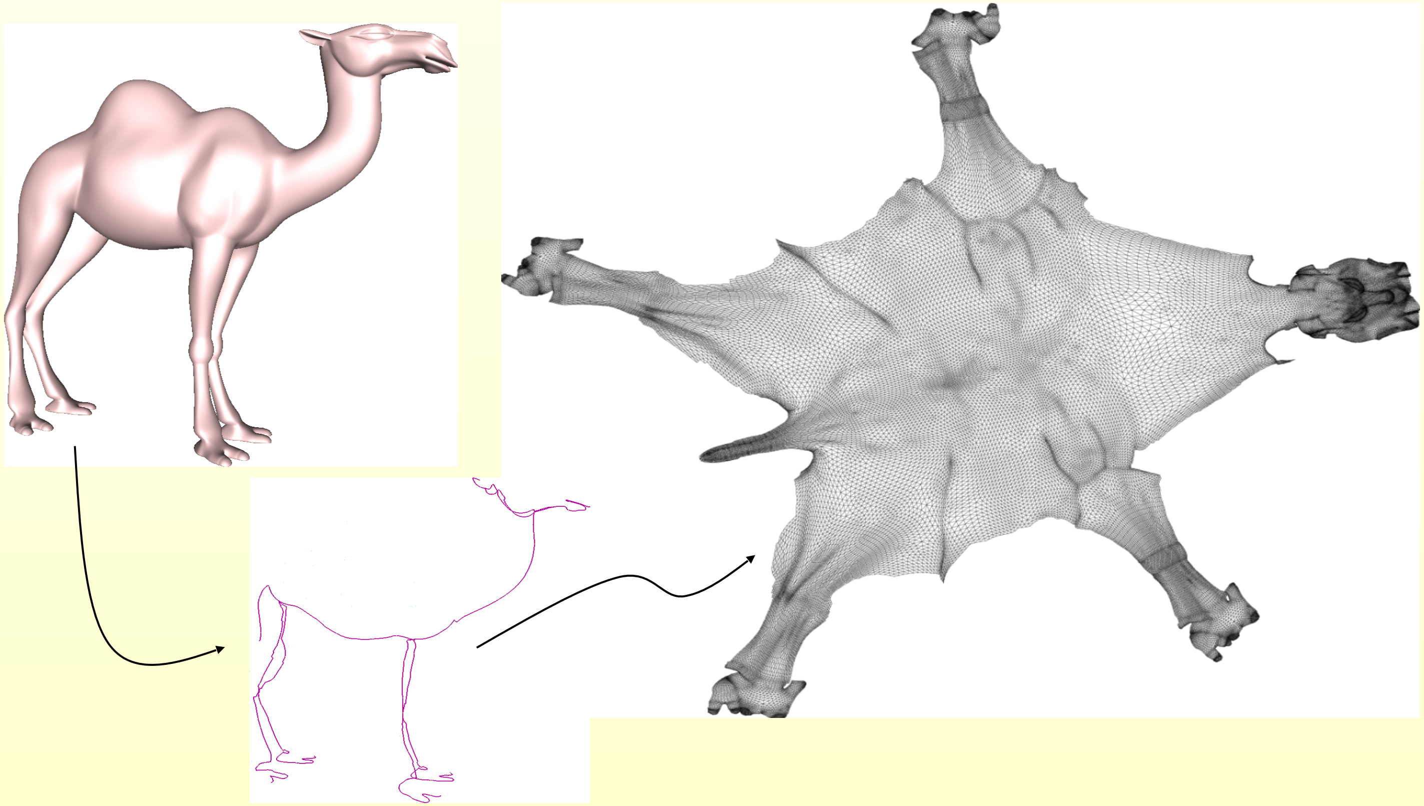
$\left(\frac{1}{2}, 2\right)$



$\left(\frac{4}{5}, \frac{4}{3}\right)$

Next Time

- Parameterization!



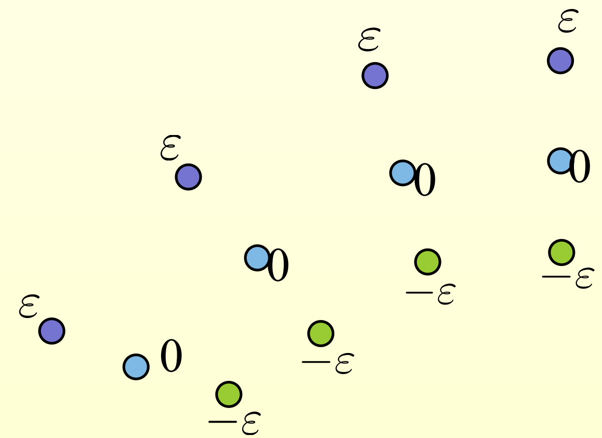
EXTRAS

Moving Least Squares (Reconstruction)
Implicit integration (Smoothing)
Vertex areas (Laplace-Beltrami)
More details on Remeshing Ops (Remeshing)

Moving Least Squares (MLS)

- Do purely **local** approximation of the SDF
- Weights change depending on where we are evaluating

- The beauty: the "stitching" of all local approximations, seen as one function $F(\mathbf{x})$, is smooth everywhere!
  - We get a **globally** smooth function but only do **local** computation

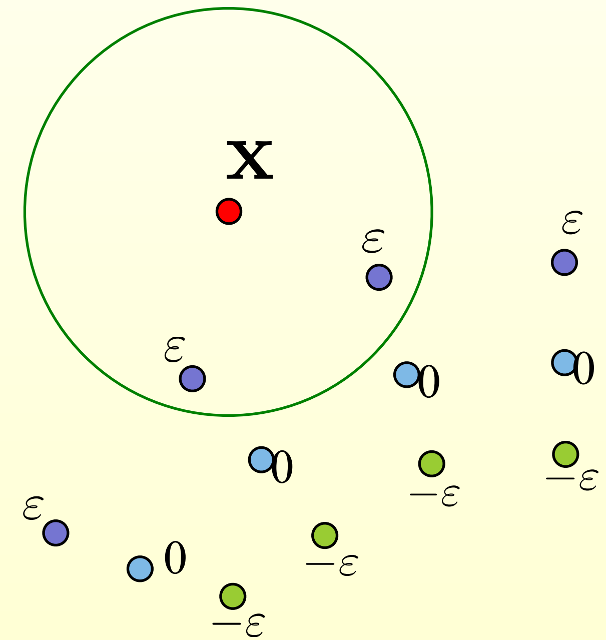


"Interpolating and Approximating Implicit Surfaces from Polygon Soup", Shen et al., ACM SIGGRAPH 2004
http://graphics.berkeley.edu/papers/Shen-IAI-2004-08/index.html

Moving Least Squares (MLS)

- Do purely **local** approximation of the SDF
- Weights change depending on where we are evaluating

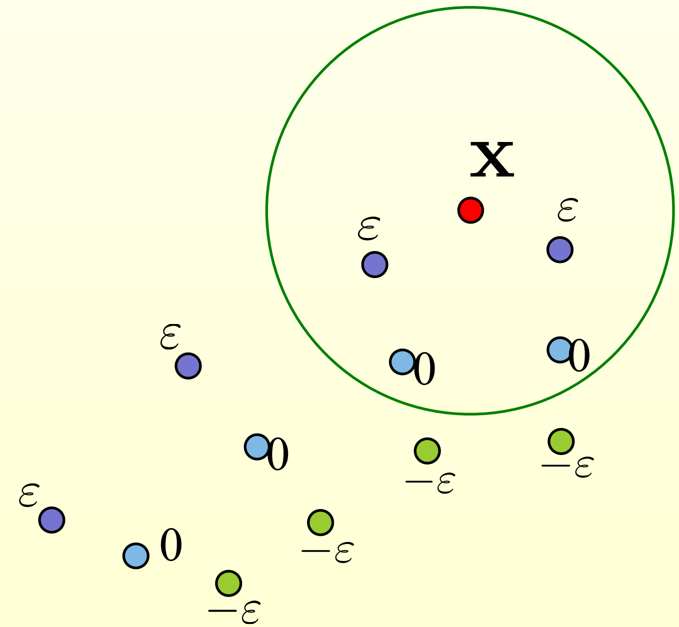
- The beauty: the "stitching" of all local approximations, seen as one function $F(\mathbf{x})$, is smooth everywhere!
  - We get a **globally** smooth function but only do **local** computation



Moving Least Squares (MLS)

- Do purely **local** approximation of the SDF
- Weights change depending on where we are evaluating

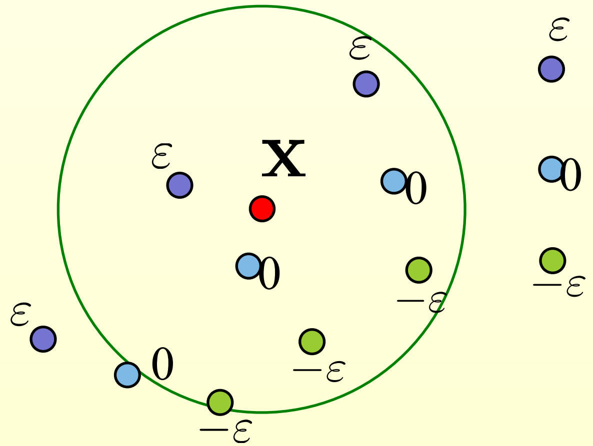
- The beauty: the “stitching” of all local approximations, seen as one function $F(\mathbf{x})$, is smooth everywhere!
  - We get a **globally** smooth function but only do **local** computation



Moving Least Squares (MLS)

- Do purely **local** approximation of the SDF
- Weights change depending on where we are evaluating

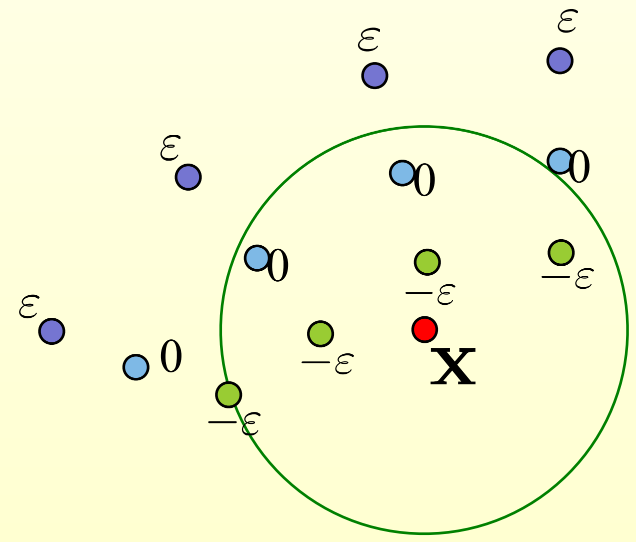
- The beauty: the “stitching” of all local approximations, seen as one function $F(\mathbf{x})$, is smooth everywhere!
  - We get a **globally** smooth function but only do **local** computation



Moving Least Squares (MLS)

- Do purely **local** approximation of the SDF
- Weights change depending on where we are evaluating

- The beauty: the “stitching” of all local approximations, seen as one function $F(\mathbf{x})$, is smooth everywhere!
  - We get a **globally** smooth function but only do **local** computation



Least-Squares Approximation

$$f \in \Pi_k^3 : f(x,y,z) = a_0 + a_1 x + a_2 y + a_3 z + a_4 x^2 + a_5 xy + \ldots + a_* z^k$$

Least-Squares Approximation

$$f \in \Pi_k^3 : f(x, y, z) = a_0 + a_1 x + a_2 y + a_3 z + a_4 x^2 + a_5 xy + \ldots + a_* z^k$$

$$f(\mathbf{x}) = \mathbf{b}(\mathbf{x})^T \, \mathbf{a}$$

Least-Squares Approximation

$$f \in \Pi_k^3 : f(x,y,z) = a_0 + a_1 x + a_2 y + a_3 z + a_4 x^2 + a_5 xy + \ldots + a_* z^k$$

$$f(\mathbf{x}) = \mathbf{b}(\mathbf{x})^T \mathbf{a}$$

$$\mathbf{a} = (a_1, a_2, \ldots, a_*)^T, \ \mathbf{b}(\mathbf{x})^T = (1, x, y, z, x^2, xy, \ldots, z^k)$$

Least-Squares Approximation

- Polynomial least-squares approximation
  - Choose degree, k

$$f \in \Pi_k^3 : f(x,y,z) = a_0 + a_1 x + a_2 y + a_3 z + a_4 x^2 + a_5 xy + \ldots + a_* z^k$$

$$f(\mathbf{x}) = \mathbf{b}(\mathbf{x})^T \mathbf{a}$$

$$\mathbf{a} = (a_1, a_2, \ldots, a_*)^T, \ \mathbf{b}(\mathbf{x})^T = (1, x, y, z, x^2, xy, \ldots, z^k)$$

  - Find $\mathbf{a}$ that minimizes sum of squared differences

Least-Squares Approximation

- Polynomial least-squares approximation
  - Choose degree, k

$$f \in \Pi_k^3 : f(x,y,z) = a_0 + a_1 x + a_2 y + a_3 z + a_4 x^2 + a_5 xy + \ldots + a_* z^k$$

$$f(\mathbf{x}) = \mathbf{b}(\mathbf{x})^T \mathbf{a}$$

$$\mathbf{a} = (a_1, a_2, \ldots, a_*)^T, \ \mathbf{b}(\mathbf{x})^T = (1, x, y, z, x^2, xy, \ldots, z^k)$$

  - Find $\mathbf{a}$ that minimizes sum of squared differences

$$\underset{f \in \Pi_k^3}{\operatorname{argmin}} \sum_{i=0}^{N-1} (f(\mathbf{c}_i) - d_i)^2 \quad \text{or:} \quad \underset{\mathbf{a}}{\operatorname{argmin}} \sum_{i=0}^{N-1} \left(\mathbf{b}(\mathbf{c}_i)^T \mathbf{a} - d_i\right)^2$$

MOVING Least-Squares Approximation

- Polynomial least-squares approximation
  - Choose degree, k

$$f \in \Pi_k^3 : f(x,y,z) = a_0 + a_1 x + a_2 y + a_3 z + a_4 x^2 + a_5 xy + \ldots + a_* z^k$$

$$f(\mathbf{x}) = \mathbf{b}(\mathbf{x})^T \mathbf{a}$$

$$\mathbf{a} = (a_1, a_2, \ldots, a_*)^T, \ \mathbf{b}(\mathbf{x})^T = (1, x, y, z, x^2, xy, \ldots, z^k)$$

  - Find $\mathbf{a}_\mathbf{x}$ that minimizes WEIGHTED sum of squared differences

$$f_\mathbf{x} = \underset{f \in \Pi_k^3}{\operatorname{argmin}} \sum_{i=0}^{N-1} \theta(\|\mathbf{x} - \mathbf{c}_i\|) \left(f(\mathbf{c}_i) - d_i\right)^2 \ \text{ or:}$$

$$\mathbf{a}_\mathbf{x} = \underset{\mathbf{a}}{\operatorname{argmin}} \sum_{i=0}^{N-1} \theta(\|\mathbf{x} - \mathbf{c}_i\|) \left(\mathbf{b}(\mathbf{c}_i)^T \mathbf{a} - d_i\right)^2$$

MOVING Least-Squares Approximation

- Polynomial least-squares approximation
  - Choose degree, k

$$f \in \Pi_k^3 : f(x, y, z) = a_0 + a_1 x + a_2 y + a_3 z + a_4 x^2 + a_5 xy + \ldots + a_* z^k$$

$$f(\mathbf{x}) = \mathbf{b}(\mathbf{x})^T \mathbf{a}$$

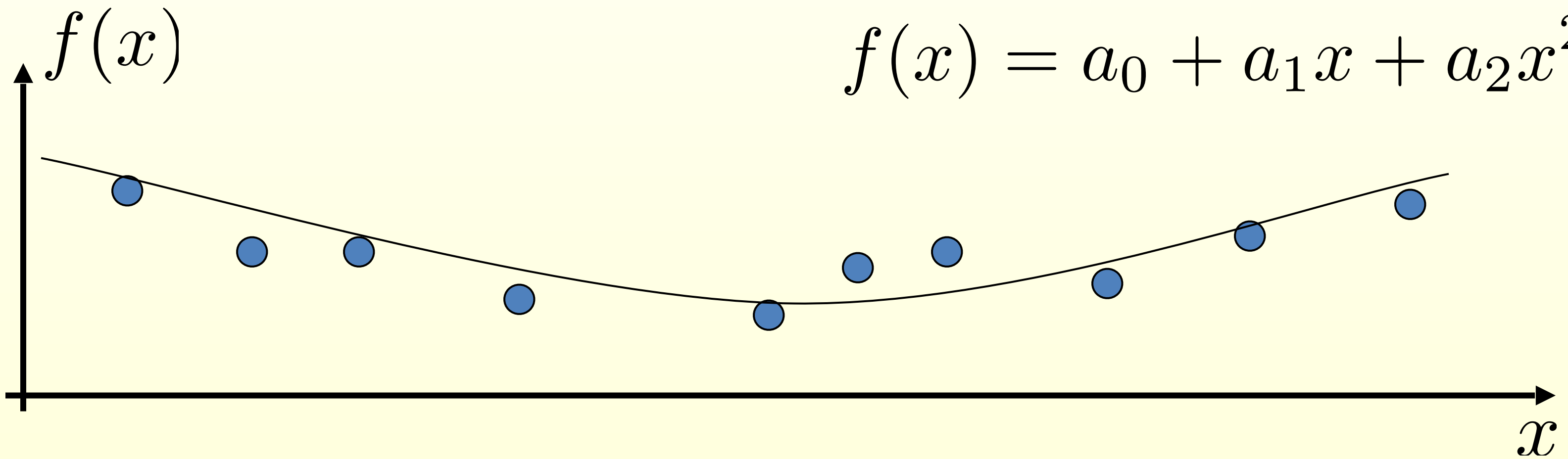
$$\mathbf{a} = (a_1, a_2, \ldots, a_*)^T, \ \mathbf{b}(\mathbf{x})^T = (1, x, y, z, x^2, xy, \ldots, z^k)$$

  - Find $\mathbf{a}_\mathbf{x}$ that minimizes WEIGHTED sum of squared differences
  - The value of the SDF is the obtained approximation evaluated at $\mathbf{x}$:

$$\boxed{F(\mathbf{x}) = f_\mathbf{x}(\mathbf{x}) = \mathbf{b}(\mathbf{x})^T \mathbf{a}_\mathbf{x}}$$

MLS – 1D Example

- Global approximation in Π_2^1

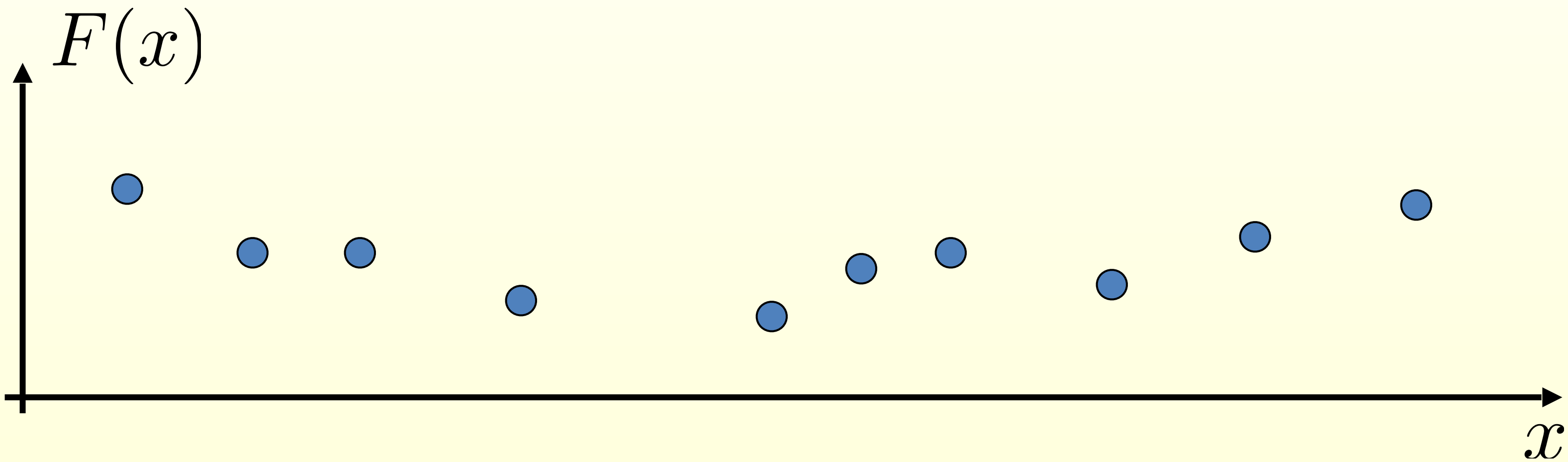


$$f(x) = a_0 + a_1 x + a_2 x^2$$

$$f = \underset{f \in \Pi_2^1}{\operatorname{argmin}} \sum_{i=0}^{N-1} \left(f(c_i) - d_i\right)^2$$

MLS – 1D Example

- MLS approximation using functions in Π_2^1

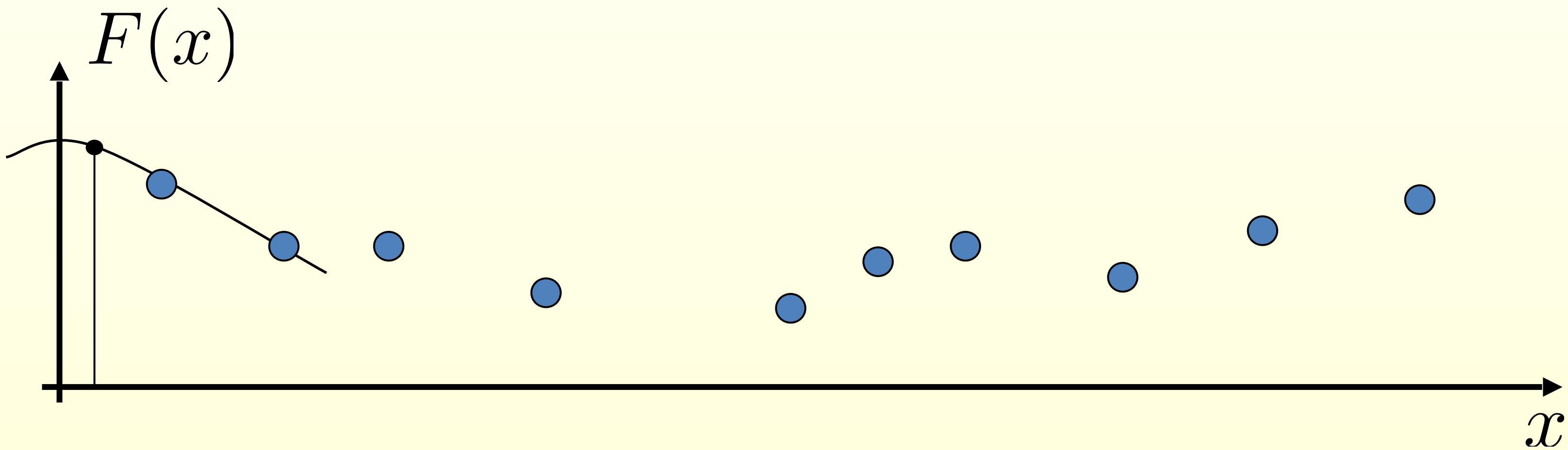


$$F(x) = f_x(x), \quad f_x = \underset{f \in \Pi_2^1}{\operatorname{argmin}} \sum_{i=0}^{N-1} \theta(\|c_i - x\|) \left(f(c_i) - d_i\right)^2$$

Evaluation of the SDF now involves a small optimization problem (linear system)

MLS – 1D Example

- MLS approximation using functions in Π_2^1

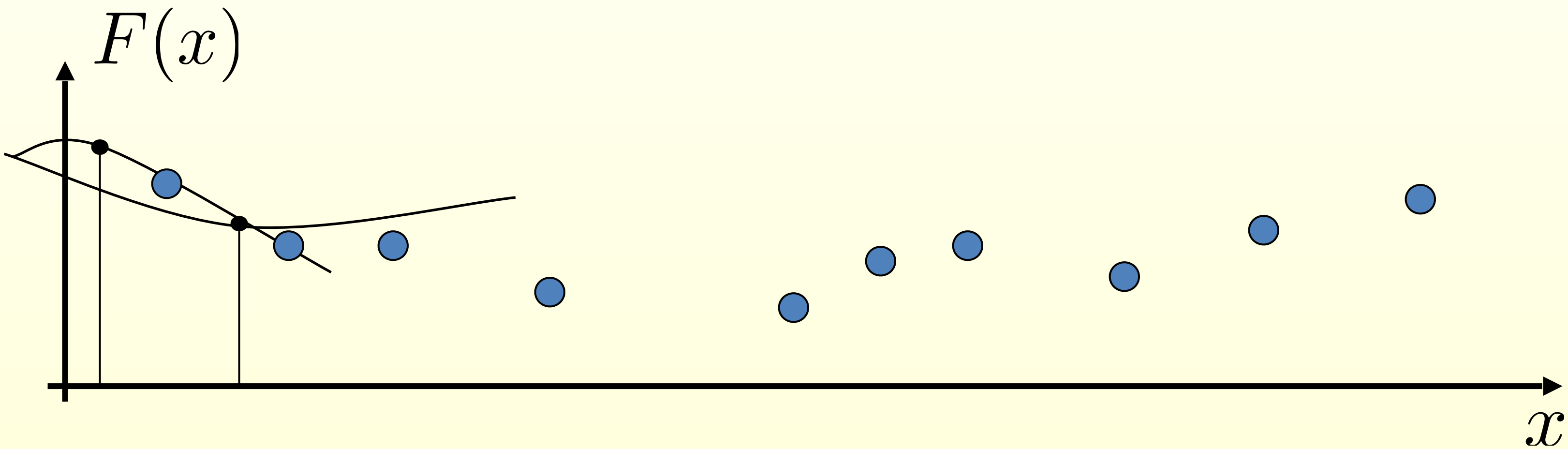


$$F(x) = f_x(x), \quad f_x = \underset{f \in \Pi_2^1}{\operatorname{argmin}} \sum_{i=0}^{N-1} \theta(\|c_i - x\|)\,(f(c_i) - d_i)^2$$

Evaluation of the SDF now involves a small optimization problem (linear system)

MLS – 1D Example

- MLS approximation using functions in Π_2^1

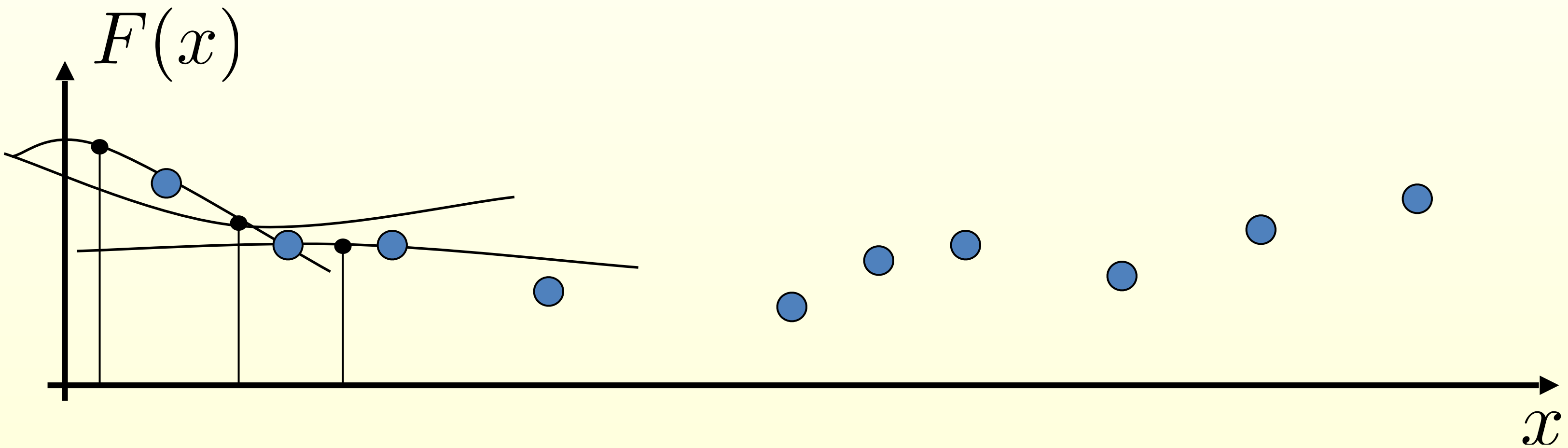


$$F(x) = f_x(x), \quad f_x = \operatorname*{argmin}_{f \in \Pi_2^1} \sum_{i=0}^{N-1} \theta(\|c_i - x\|)\,(f(c_i) - d_i)^2$$

Evaluation of the SDF now involves a small optimization problem (linear system)

MLS – 1D Example

- MLS approximation using functions in Π_2^1

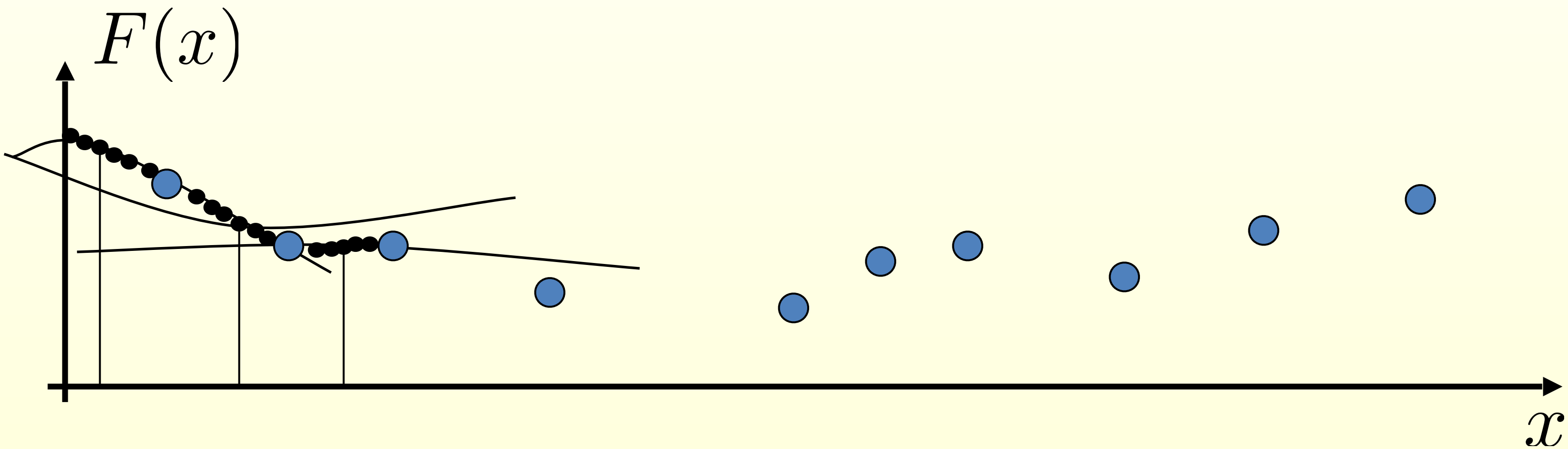


$$F(x) = f_x(x), \quad f_x = \underset{f \in \Pi_2^1}{\operatorname{argmin}} \sum_{i=0}^{N-1} \theta(\|c_i - x\|)\,(f(c_i) - d_i)^2$$

Evaluation of the SDF now involves a small optimization problem (linear system)

MLS – 1D Example

- MLS approximation using functions in Π_2^1

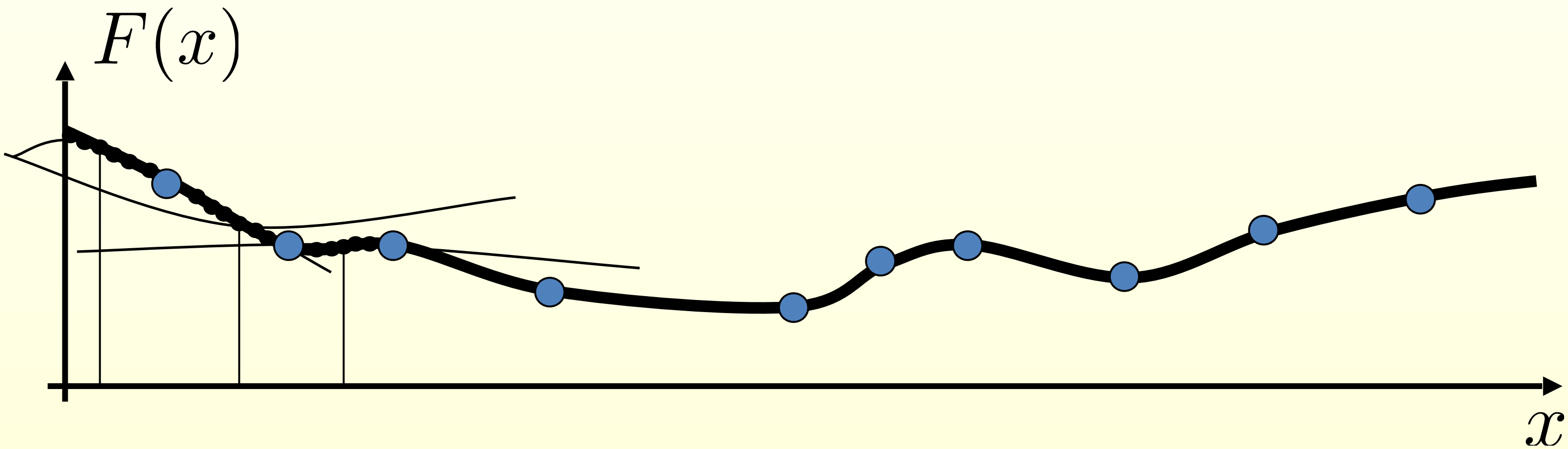


$$F(x) = f_x(x), \quad f_x = \operatorname*{argmin}_{f \in \Pi_2^1} \sum_{i=0}^{N-1} \theta(\|c_i - x\|) \left(f(c_i) - d_i\right)^2$$

Evaluation of the SDF now involves a small optimization problem (linear system)

MLS – 1D Example

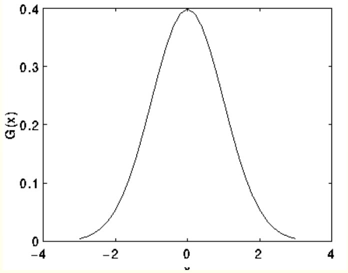
- MLS approximation using functions in Π_2^1



$$F(x) = f_x(x), \quad f_x = \underset{f \in \Pi_2^1}{\operatorname{argmin}} \sum_{i=0}^{N-1} \theta(\|c_i - x\|)\,(f(c_i) - d_i)^2$$

Evaluation of the SDF now involves a small optimization problem (linear system)

Weight Functions



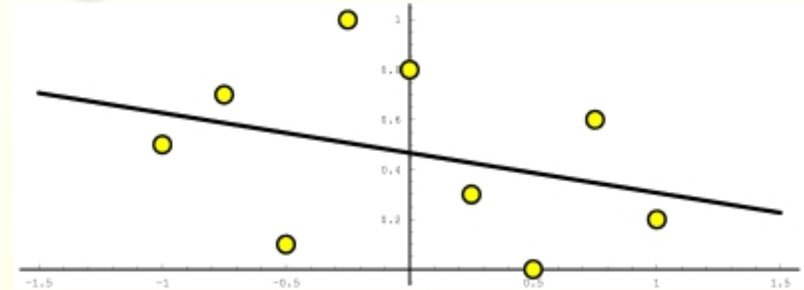
- Gaussian $$\theta(r) = e^{-\frac{r^2}{h^2}}$$
  - h is a smoothing parameter
- Wendland function $$\theta(r) = (1 - r/h)^4(4r/h + 1)$$
  - Defined in $[0, h]$ and

$$\theta(0) = 1,\ \theta(h) = 0,\ \theta'(h) = 0,\ \theta''(h) = 0$$

- Singular function $$\theta(r) = \frac{1}{r^2 + \varepsilon^2}$$
  - For small ε, weights large near r=0 (interpolation)

Dependence on Weight Function

- Global least squares with linear basis

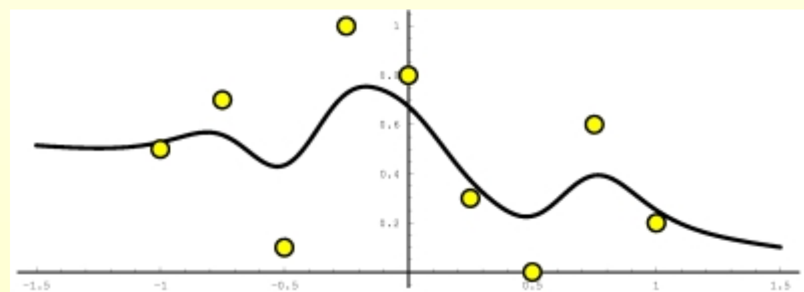


- MLS with (nearly) singular weight function

$$\theta(r) = \frac{1}{r^2 + \varepsilon^2}$$

- MLS with approximating weight function

$$\theta(r) = e^{-\frac{r^2}{h^2}}$$



Dependence on Weight Function

- The MLS function F is continuously differentiable if and only if the weight function θ is continuously differentiable
- In general, F is as smooth as θ

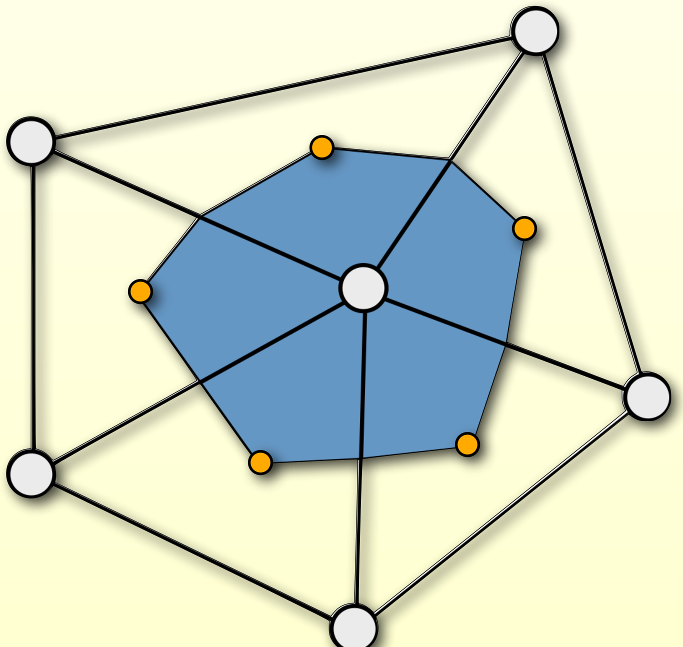
$$F(\mathbf{x}) = f_{\mathbf{x}}(\mathbf{x}), \quad f_{\mathbf{x}} = \operatorname*{argmin}_{f \in \Pi_k^d} \sum_{i=0}^{N-1} \theta(\|\mathbf{c}_i - \mathbf{x}\|) \left(f(\mathbf{c}_i) - d_i\right)^2$$

Global RBF vs. Local MLS

- RBF:
  - sees the whole data set, can make for very smooth surfaces
  - global (dense) system to solve – expensive
- MLS:
  - sees only a small part of the dataset, can get confused by noise
  - local linear solves – cheap

Vertex Area - Barycentric

- **Barycentric area**
  - Connect edge midpoints and triangle barycenters
  - Each of the incident triangles contributes 1/3 of its area to all its vertices, regardless of the placement



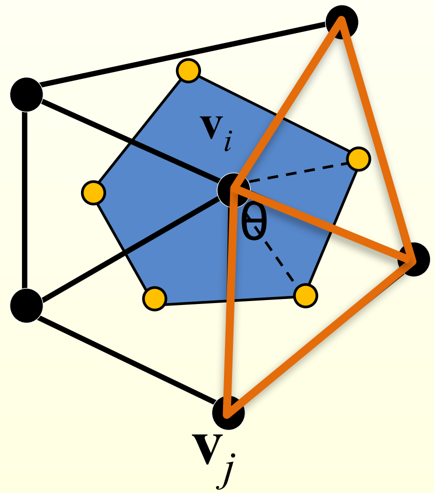
+ Simple to compute

+ Always positive weights

- Heavily connectivity dependent

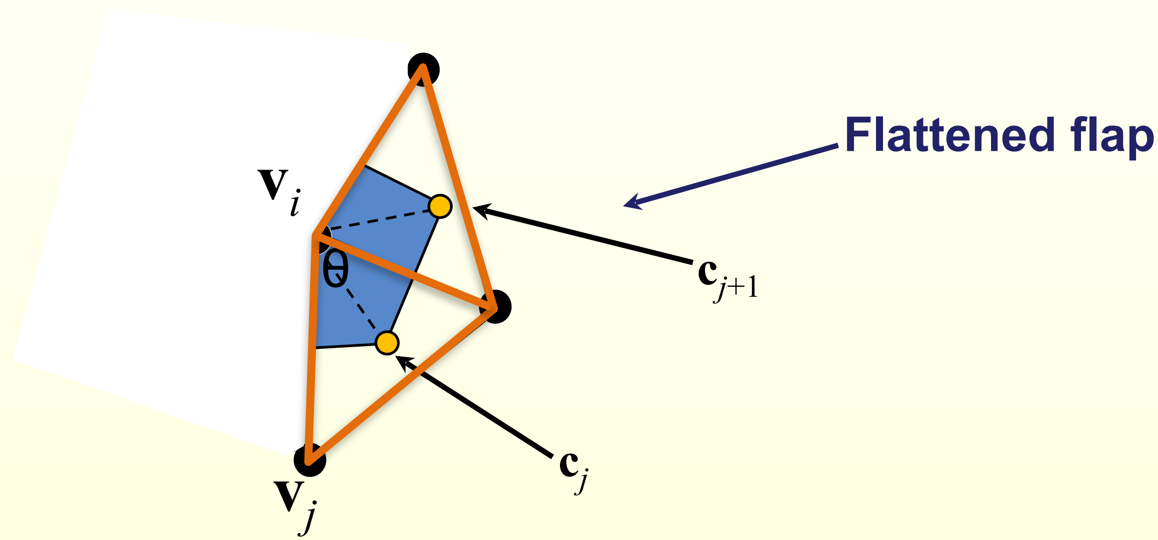
- Changes if edges are flipped

Vertex Area - Voronoi



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

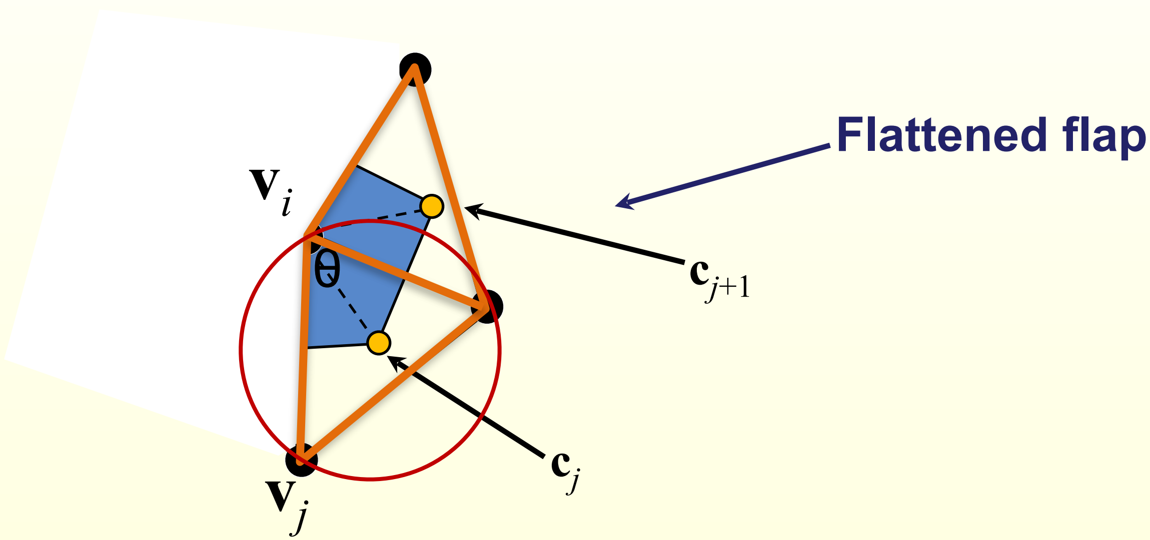
Voronoi Vertex Area



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

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

Voronoi Vertex Area



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

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

Smoothing and Numerical Integration

Smoothing and Numerical Integration

- Explicit integration of diffusion can be unstable

$$\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} + \lambda \Delta \mathbf{p}_i^{(t)}$$
$$\mathbf{P}^{(t)} = \left(\mathbf{p}_1^{(t)}, \ldots, \mathbf{p}_n^{(t)}\right)^T \in \mathbb{R}^{n \times 3}$$

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

Smoothing and Numerical Integration

- Explicit integration of diffusion can be unstable

$$\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} + \lambda \Delta \mathbf{p}_i^{(t)}$$
$$\mathbf{P}^{(t)} = \left(\mathbf{p}_1^{(t)}, \ldots, \mathbf{p}_n^{(t)}\right)^T \in \mathbb{R}^{n \times 3}$$

- Implicit integration is unconditionally stable

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

Smoothing and Numerical Integration

- Explicit integration of diffusion can be unstable

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

$$\mathbf{P}^{(t)} = \left(\mathbf{p}_1^{(t)}, \ldots, \mathbf{p}_n^{(t)}\right)^T \in \mathbb{R}^{n \times 3}$$

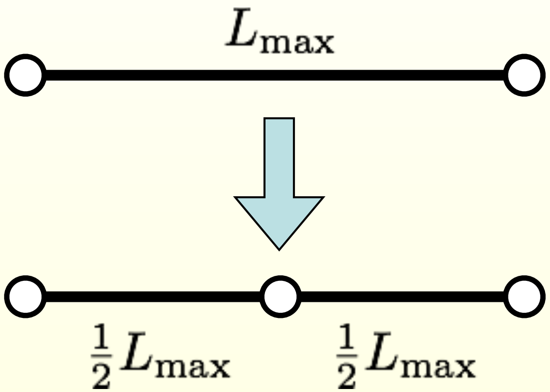
- Implicit integration is unconditionally stable

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

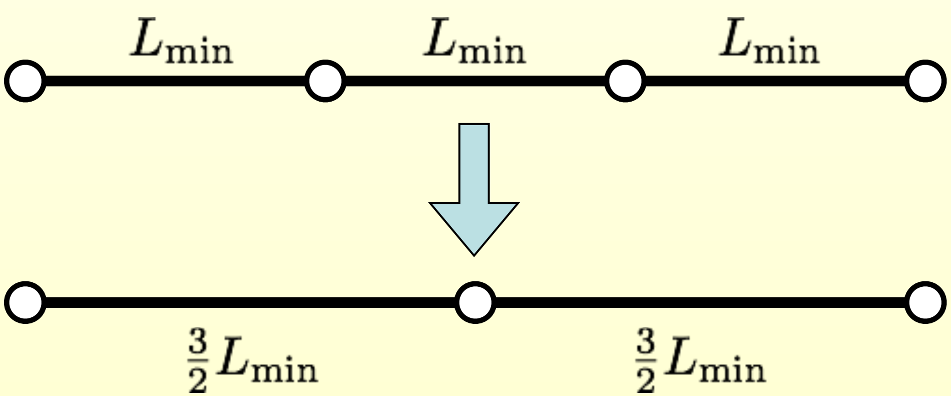
$$(\mathbf{I} - \lambda \mathbf{L})\, \mathbf{P}^{(t+1)} = \mathbf{P}^{(t)}$$

- … boils down to a sparse symmetric positive definite system solve
  - Iterative conjugate gradients, sparse Cholesky

Edge Collapse / Split



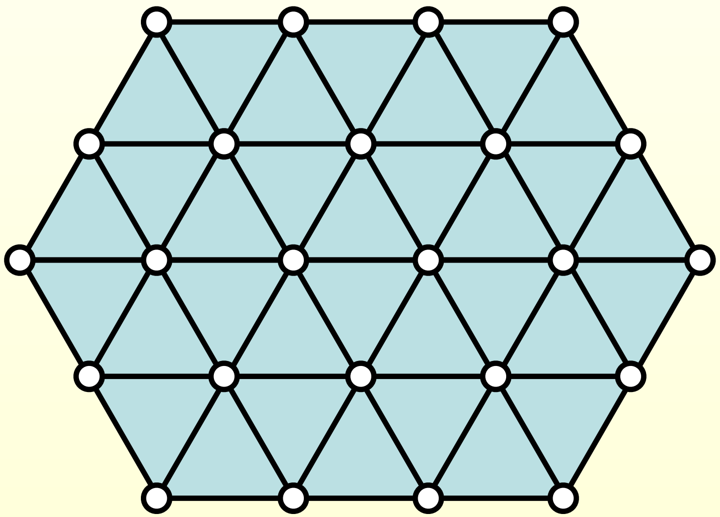
$$|L_{\max} - L| = \left|\tfrac{1}{2}L_{\max} - L\right|$$
$$\Rightarrow L_{\max} = \tfrac{4}{3}L$$



$$|L_{\min} - L| = \left|\tfrac{3}{2}L_{\max} - L\right|$$
$$\Rightarrow L_{\min} = \tfrac{4}{5}L$$

Edge Flip

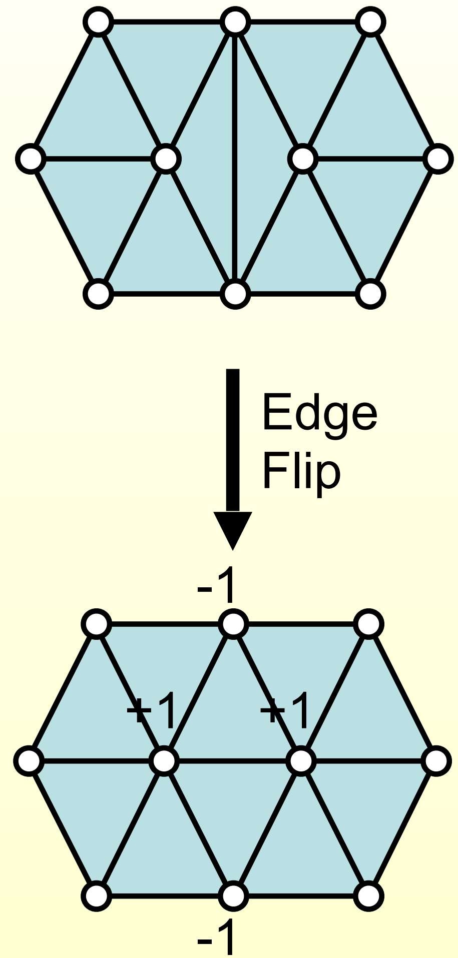
- Improve valences
  - Avg. valence is 6 (Euler)
  - Reduce variation
- Optimal valence is
  - 6 for interior vertices
  - 4 for boundary vertices



Edge Flip

- Improve valences
  - Avg. valence is 6 (Euler)
  - Reduce variation
- Optimal valence is
  - 6 for interior vertices
  - 4 for boundary vertices
- Minimize valence excess

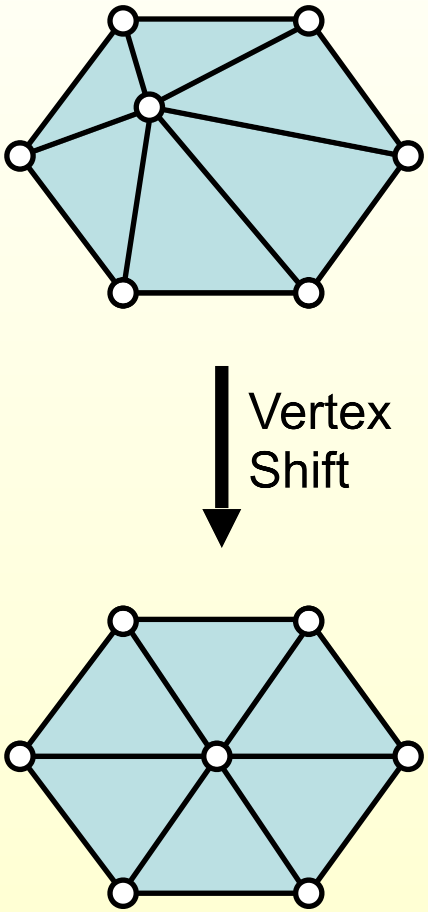
$$\sum_{i=1}^{4} (\text{valence}(v_i) - \text{opt\_valence}(v_i))^2$$



Vertex Shift

- Local "spring" relaxation
  - Uniform Laplacian smoothing
  - Bary-center of one-ring neighbors

$$\mathbf{c}_i = \frac{1}{\text{valence}(v_i)} \sum_{j \in N(v_i)} \mathbf{p}_j$$



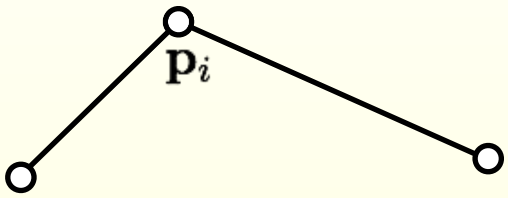
Vertex Shift

- Local “spring” relaxation
  - Uniform Laplacian smoothing
  - Bary-center of one-ring neighbors

$$\mathbf{c}_i = \frac{1}{\text{valence}\,(v_i)} \sum_{j \in N(v_i)} \mathbf{p}_j$$

Vertex Shift

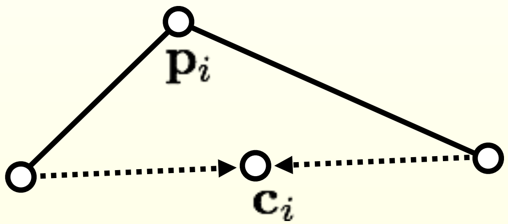
- Local "spring" relaxation
  - Uniform Laplacian smoothing
  - Bary-center of one-ring neighbors



$$\mathbf{c}_i = \frac{1}{\text{valence}\,(v_i)} \sum_{j \in N(v_i)} \mathbf{p}_j$$

Vertex Shift

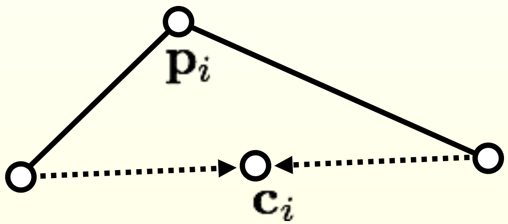
- Local "spring" relaxation
  - Uniform Laplacian smoothing
  - Bary-center of one-ring neighbors



$$\mathbf{c}_i = \frac{1}{\text{valence}\,(v_i)} \sum_{j \in N(v_i)} \mathbf{p}_j$$

Vertex Shift

- Local "spring" relaxation
  - Uniform Laplacian smoothing
  - Bary-center of one-ring neighbors



$$\mathbf{c}_i = \frac{1}{\text{valence}(v_i)} \sum_{j \in N(v_i)} \mathbf{p}_j$$

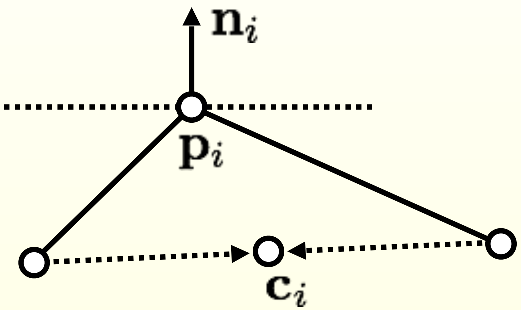
- Keep vertex (approx.) of surface
  - Restrict movement to tangent plane

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \lambda \left(I - \mathbf{n}_i \mathbf{n}_i^T\right) (\mathbf{c}_i - \mathbf{p}_i)$$

Vertex Shift

- Local "spring" relaxation
  - Uniform Laplacian smoothing
  - Bary-center of one-ring neighbors

$$\mathbf{c}_i = \frac{1}{\text{valence}\,(v_i)} \sum_{j \in N(v_i)} \mathbf{p}_j$$



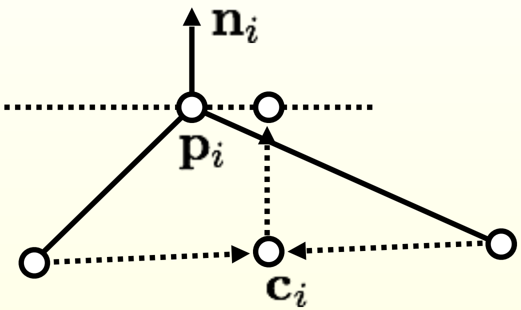
- Keep vertex (approx.) of surface
  - Restrict movement to tangent plane

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \lambda \left(I - \mathbf{n}_i \mathbf{n}_i^T\right) (\mathbf{c}_i - \mathbf{p}_i)$$

Vertex Shift

- Local "spring" relaxation
  - Uniform Laplacian smoothing
  - Bary-center of one-ring neighbors

$$\mathbf{c}_i = \frac{1}{\text{valence}\,(v_i)} \sum_{j \in N(v_i)} \mathbf{p}_j$$



- Keep vertex (approx.) of surface
  - Restrict movement to tangent plane

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \lambda \left(I - \mathbf{n}_i \mathbf{n}_i^T\right) \left(\mathbf{c}_i - \mathbf{p}_i\right)$$

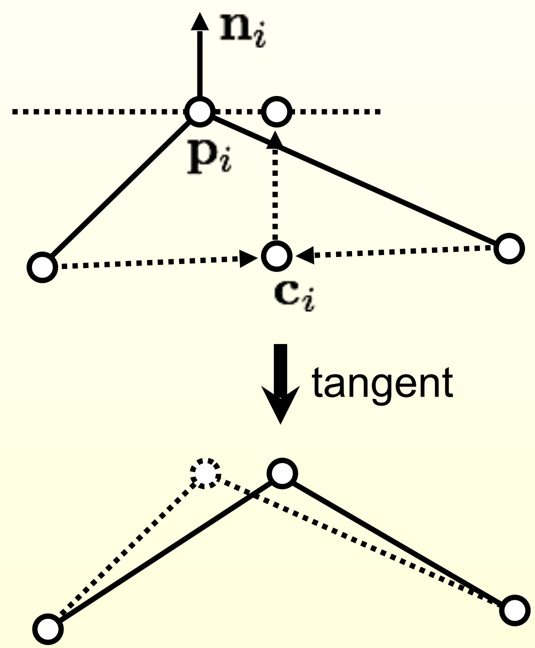
Vertex Shift

- Local “spring” relaxation
  - Uniform Laplacian smoothing
  - Bary-center of one-ring neighbors

$$\mathbf{c}_i = \frac{1}{\text{valence}(v_i)} \sum_{j \in N(v_i)} \mathbf{p}_j$$

- Keep vertex (approx.) of surface
  - Restrict movement to tangent plane

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \lambda \left(I - \mathbf{n}_i \mathbf{n}_i^T \right) (\mathbf{c}_i - \mathbf{p}_i)$$



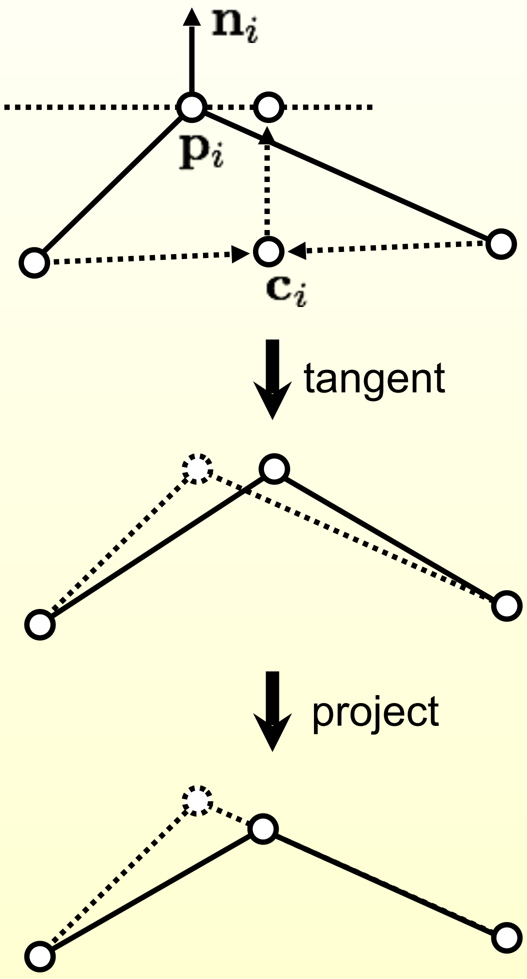
Vertex Shift

- Local “spring” relaxation
  - Uniform Laplacian smoothing
  - Bary-center of one-ring neighbors

$$\mathbf{c}_i = \frac{1}{\text{valence}\,(v_i)} \sum_{j \in N(v_i)} \mathbf{p}_j$$

- Keep vertex (approx.) of surface
  - Restrict movement to tangent plane

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \lambda \left(I - \mathbf{n}_i \mathbf{n}_i^T\right) (\mathbf{c}_i - \mathbf{p}_i)$$



Vertex Projection

- Project vertices onto original reference mesh
  - Static reference mesh
  - Precompute BSP
- Assign position & interpolated normal