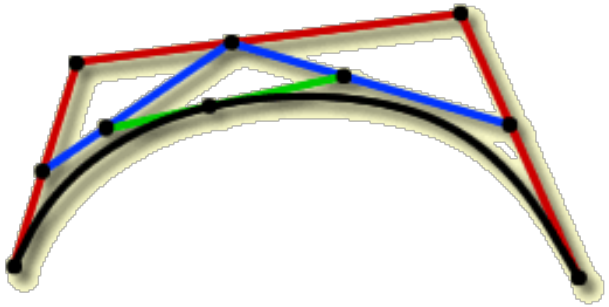
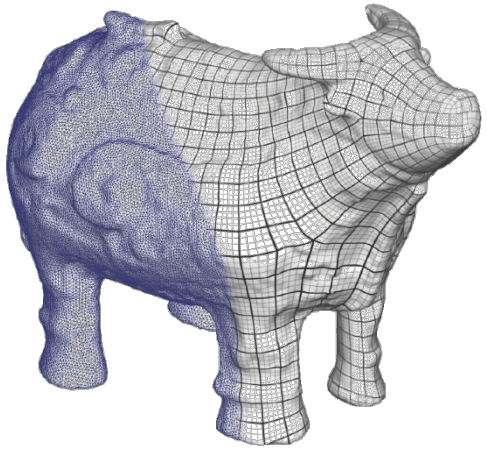
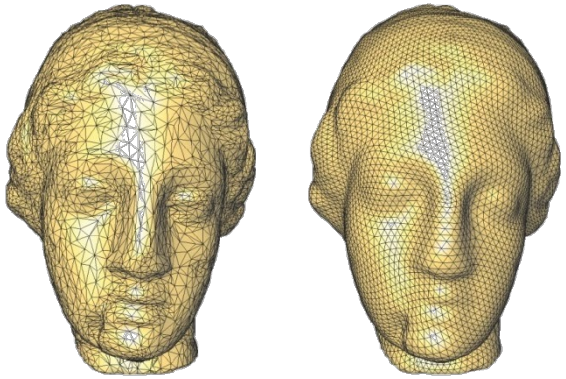
CS348a: Geometric Modeling and Processing



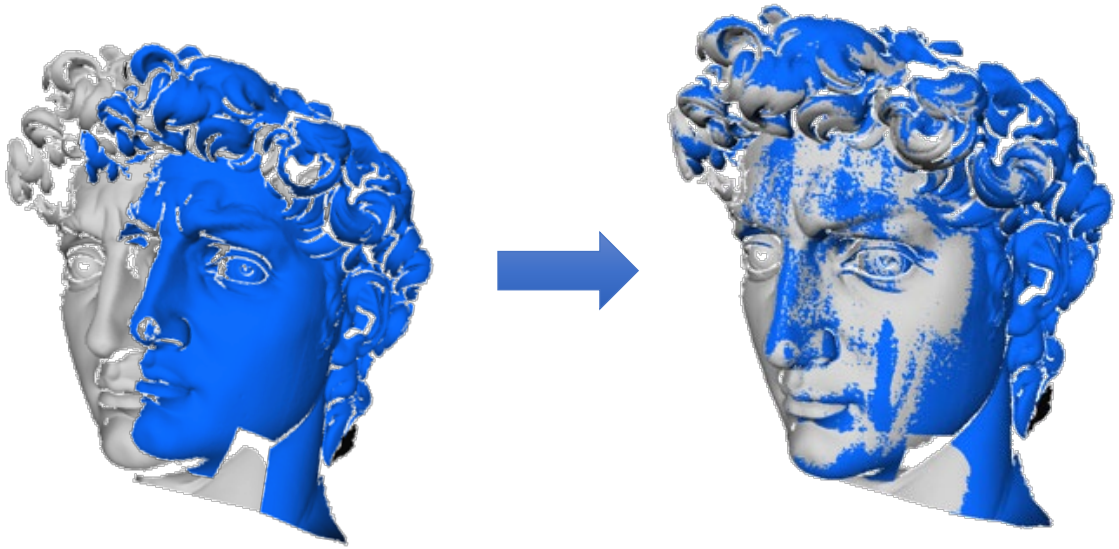


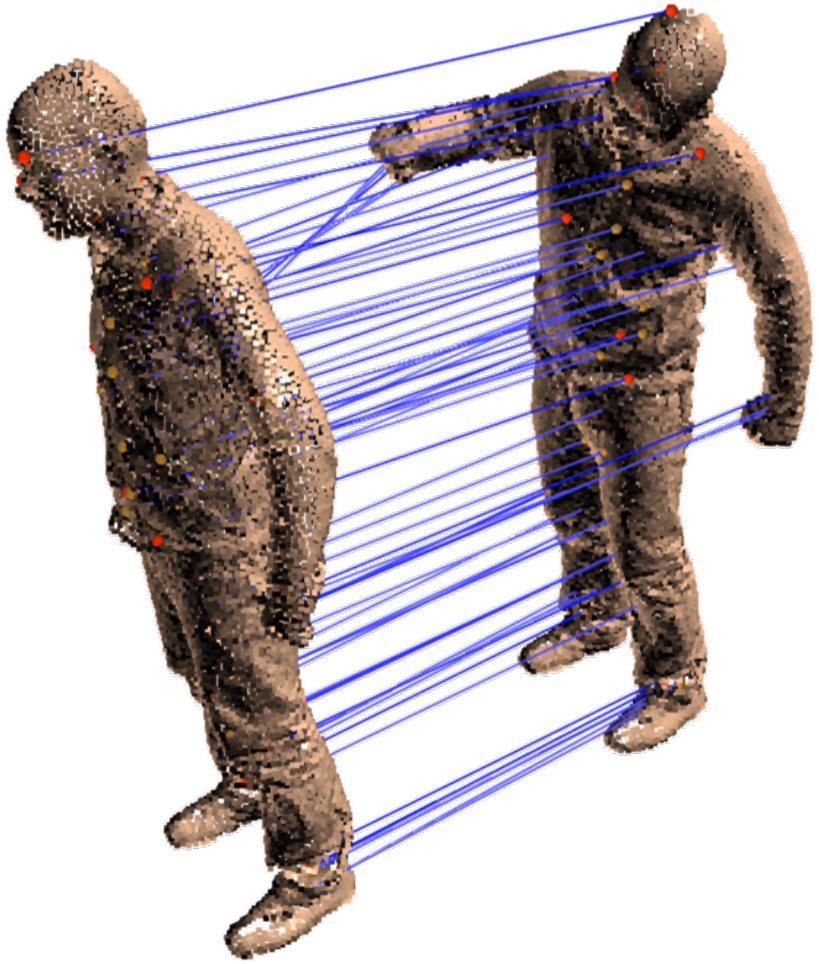


Leonidas Guibas
Computer Science Department
Stanford University

Last Time:
Shape Registration, Matching, Correspondences

Registration and Matching

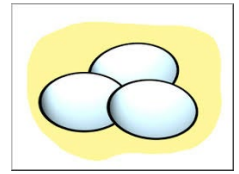




How to Get Correspondences?

A chicken-and-egg problem: if we knew the optimal aligning transform, then we could get correspondences by proximity (possibly with the aid of some global adjustment, e.g., dynamic programming)

Transform ⟺ Correspondences



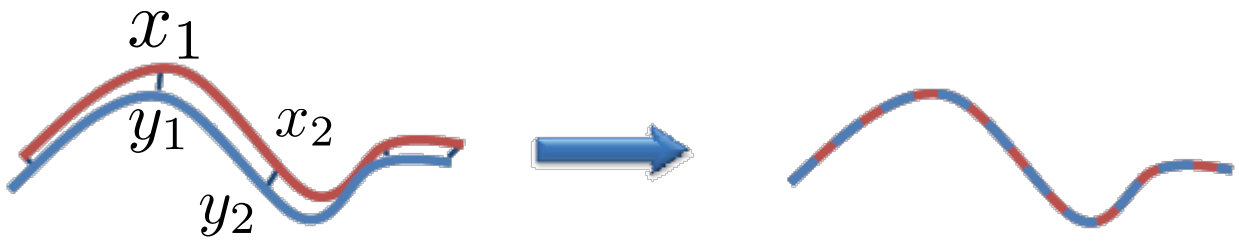
Guess one, estimate the other, and *iterate!*

EM like

- Correspondences from proximity (Iterated Closest Pair)
- Correspondences from local shape descriptors (Shape Features)
- Transform from voting schemes (Geometric Hashing)
- Combinations

Iterative Closest Point (ICP)

- Approach: iterate between finding correspondences and finding the transformation:



Given a pair of shapes, X and Y, iterate:

1. For each $x_i \in X$ find **nearest** neighbor $y_i \in Y$.
2. Find rigid motion $\mathbf{R}, t$ minimizing:

$$\sum_{i=1}^{N} \|\mathbf{R}x_i + t - y_i\|_2^2$$

Simplest Case: Rigid Alignment, Given Correspondences

- We are given two sets of corresponding points $x_1, x_2, ..., x_n$ and $y_1, y_2, ..., y_n$ in $\Re^3$. We wish to compute the rigid transform T that best aligns x_1 to y_1, x_2 to y_2, ..., and x_n to y_n.
- We define the error to be minimized by

$$\min_T \sum_{i=1}^{n} ||T(x_i) - y_i||^2$$

MSE error, RMS distance, …

- Old Problem:
  - Known and solved as the *orthogonal Procrustes problem* in Factor Analysis (Statistics) [Shönemann, 1966]
  - Known and solved as the *absolute orientation problem* in Photogrammetry [Horn, 1986]
  - Also in robotics, graphics, medical image analysis, statistical theories of shape, etc ...

SVD-Based Solution

- A rigid motion *T* is a combination of a translation *a* and a rotation *R*, so that $T(x) = R(x) + a$.
- If we place the origin of our coordinate system at the mean of the x_i's, then the quantity to be minimized simplifies to (up to some constants):

$$\min_{a,R} \left(\sum_{i=1}^{n} |y_i - a|^2 - 2 \sum_{i=1}^{n} \langle R(x_i), y_i \rangle \right)$$

T

R

- Note that the translational and rotational parts separate. The translational part *a* can easily be seen to be optimized by

$$a = \frac{1}{n} \sum_{i=1}^{n} y_i$$

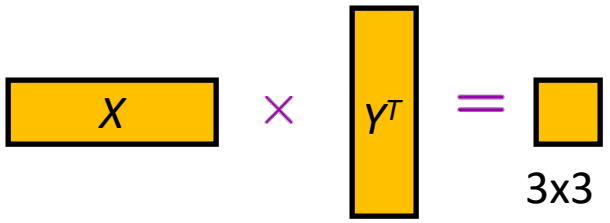
The centroids of the two point sets have to be aligned!

The Rotation Part via SVD

- Define

$$\begin{aligned} \overline{x} &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ \overline{y} &= \frac{1}{n}\sum_{i=1}^{n} y_i \\ X &= [x_1 - \overline{x}, \ldots x_n - \overline{x}]^T \\ Y &= [y_1 - \overline{y}, \ldots y_n - \overline{y}]^T \end{aligned}$$

- Here *X* and *Y* are 3 by *n* matrices.



*SVD = singular value decomposition

- Now compute the SVD*

$$XY^T = UDV^T \quad (3 \times 3)$$

- *U* and *V* are 3 by 3 orthogonal matrices, and *D* is a diagonal matrix with decreasing non-negative entries along the diagonal (the singular values).

- Define *S* by

$$S = \begin{cases} I, & \text{if } \det U \det V = 1 \\ \text{diag}(1, \ldots, 1, -1), & \\ \text{otherwise} & \end{cases}$$

- Then $\boxed{R = USV^T}$

O(*n*) algorithm!

Iterative Closest Point

Given a pair of shapes, X and Y, iterate:

1. For each $x_i \in X$ find **nearest** neighbor $y_i \in Y$.
2. Find rigid motion $\mathbf{R}, t$ minimizing: $\sum_{i=1}^{N} \|\mathbf{R}x_i + t - y_i\|_2^2$

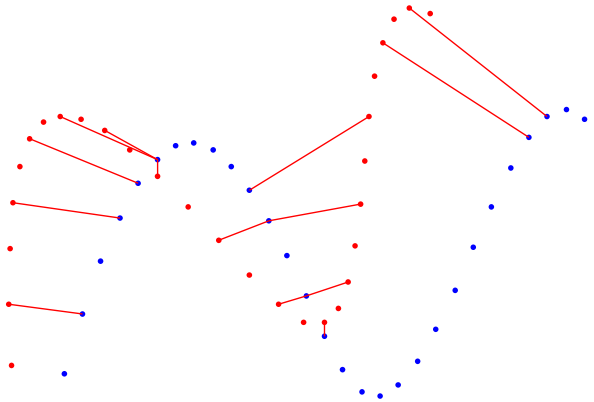
Convergence:

- at each iteration $\sum_{i=1}^{N} d^2(x_i, Y)$ decreases.
- Converges to local minimum
- Good initial guess: global minimum.

[Besl&McKay92]

Variations of ICP

1. Selecting source points (from one or both scans): sampling
2. Matching to points in the other mesh
3. Weighting the correspondences
4. Rejecting certain (outlier) point pairs
5. Assigning an error metric to the current transform
6. Minimizing the error metric w.r.t. the transformation



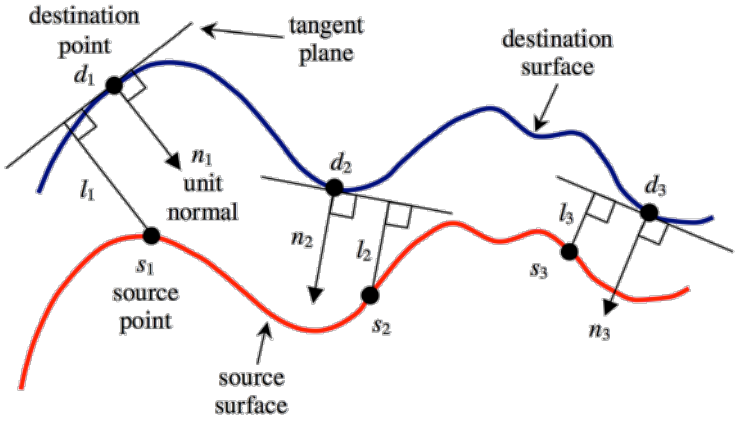
Iterative Closest Point

Given a pair of shapes, X and Y, iterate:

1. For each $x_i \in X$ find **nearest** neighbor $y_i \in Y$.
2. Find rigid motion $\mathbf{R}, t$ minimizing:

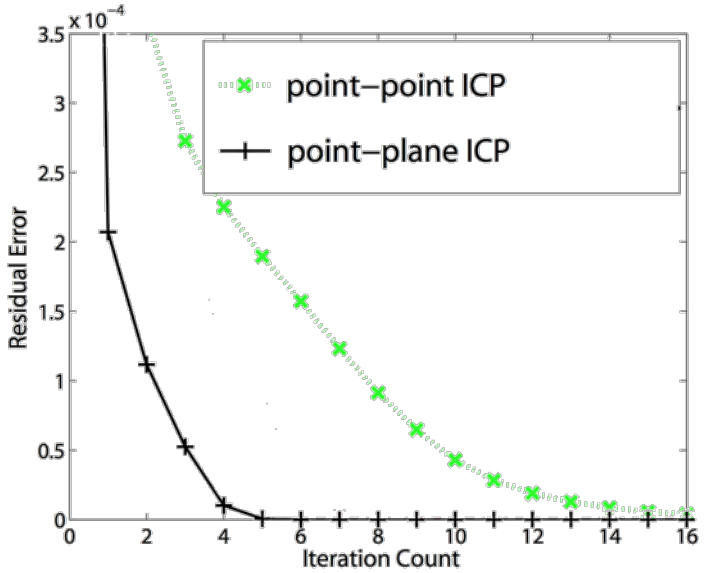
$$\sum_{i=1}^{N} d(\mathbf{R}x_i + t, P(y_i))^2 = \sum_{i=1}^{N} \left((\mathbf{R}x_i + t - y_i)^T \mathbf{n}_{y_i}\right)$$

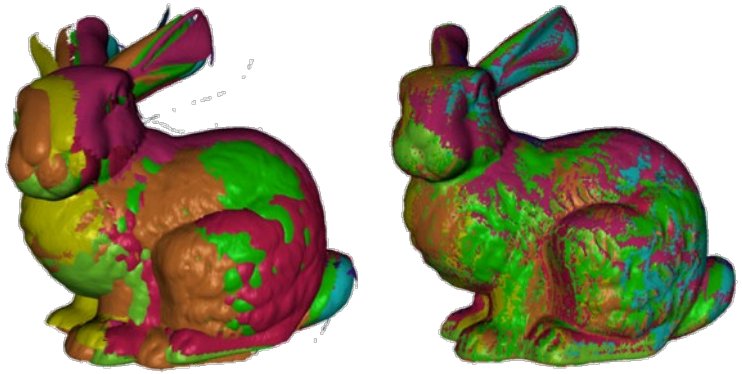
Solution:
Minimize distance to the tangent plane



Kok-Lim Low, '04
[linear least squares]

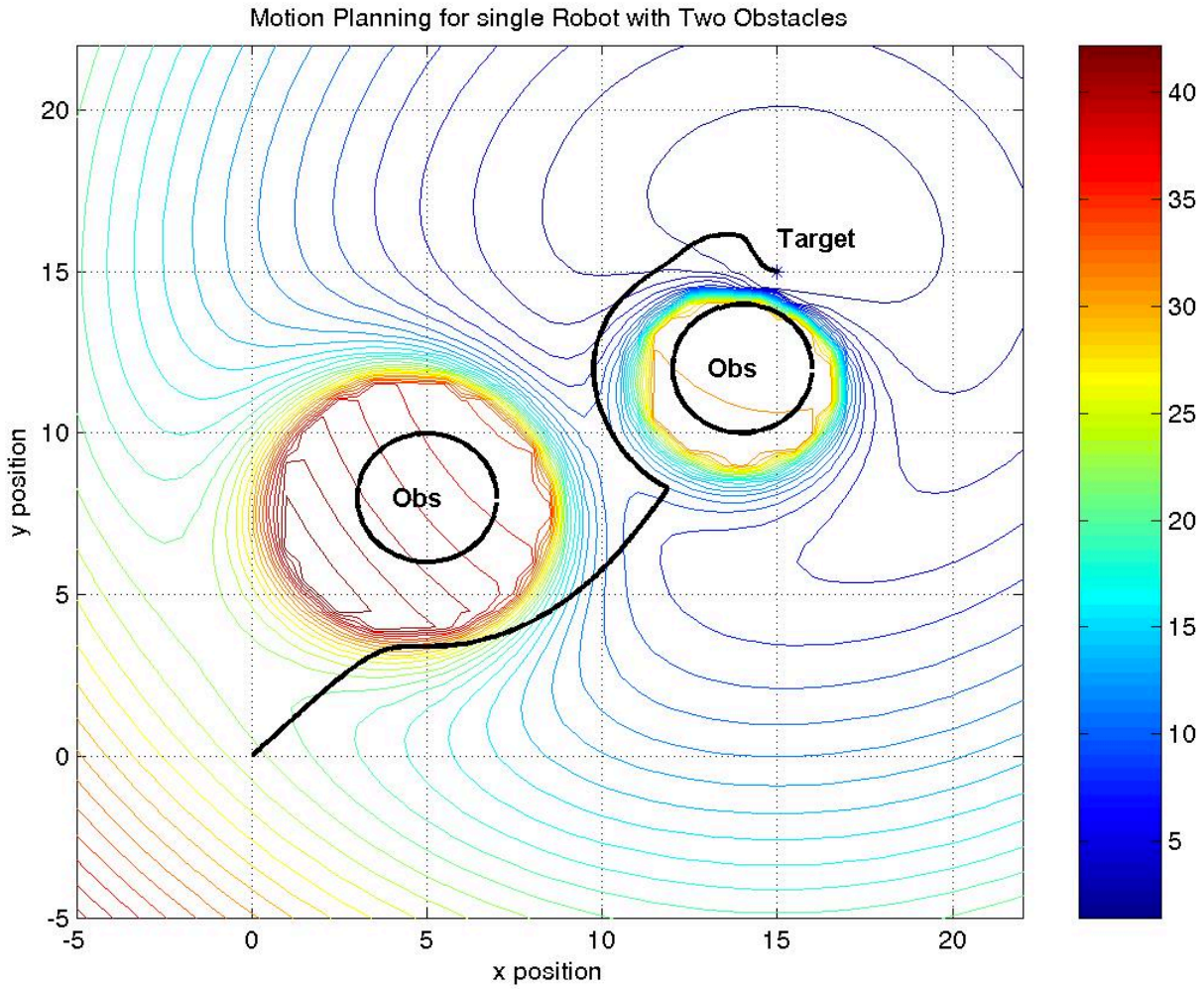
Iterative Closest Point





Aligning the bunny to itself:
Point-to-plane always wins in the end-game.

“Gravitational” Potential



Robot motion planning via potential fields

“Gravitational” Potential

- Given two related shapes, the “data” A and the “model” B, create a potential field that pulls B to the correct alignment with A

- Key tasks
  - Define the potential field
  - Formulate the optimization problem
  - Do gradient descent using approximate linearization
  - Iterative approach

Approximate Squared Distance

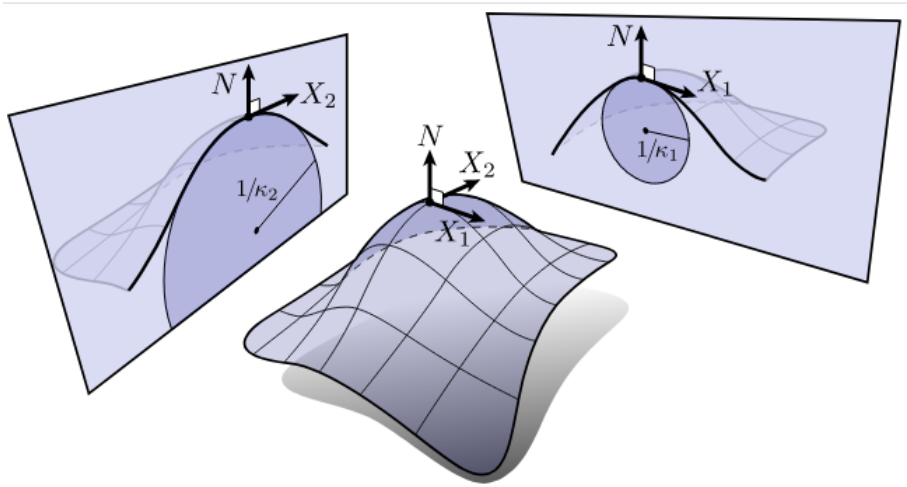
For a curve Ψ, to second order:

$$d^2(y, \Psi) \approx \frac{d}{d - \rho_1} x_1^2 + x_2^2$$

For a surface Φ, to second order:

$$d^2(y, \Phi) \approx \frac{d}{d - \rho_1} x_1^2 + \frac{d}{d - \rho_2} x_2^2 + x_3^2$$

$\rho_1 = 1/\kappa_1$ and $\rho_2 = 1/\kappa_2$ are inverse principal curvatures



[Pottmann and Hofer 2003]

Approximate Squared Distance

For a surface Φ, to second order:

$$d^2(y, \Phi) \approx \frac{d}{d - \rho_1} x_1^2 + \frac{d}{d - \rho_2} x_2^2 + x_3^2$$

$\rho_1 = 1/\kappa_1$ and $\rho_2 = 1/\kappa_2$ are inverse principal curvatures

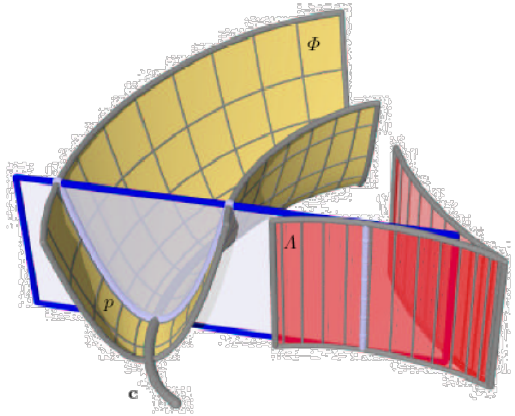
Note that as $d \to 0, \quad d^2(y, \Phi) \to x_3^2$ point-to-plane

$d \to \infty, \quad d^2(y, \Phi) \to x_1^2 + x_2^2 + x_3^2$ point-to-point

ICP Without Correspondences

- ICP without correspondences
  - define a quadratic approximant to the square distance function

[Pottman & Hofer, 02]

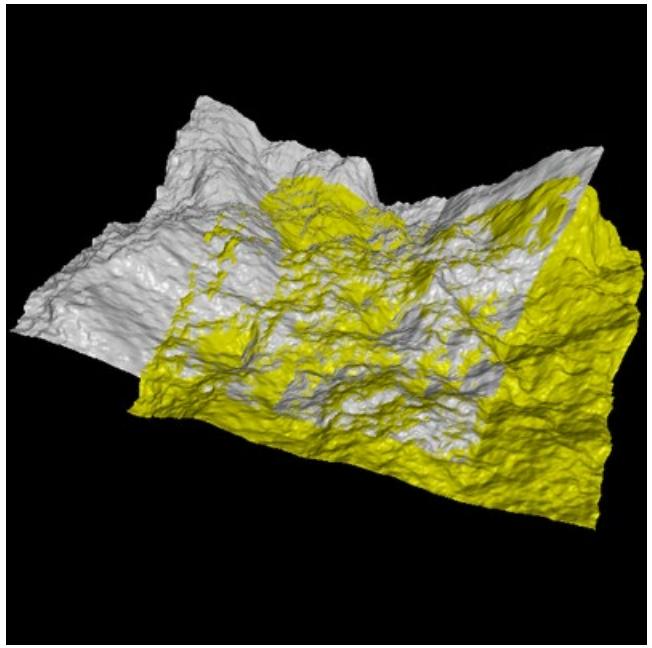


$$F(x_1, x_2, x_3) = \frac{d}{d-\rho_1}x_1^2 + \frac{d}{d-\rho_2}x_2^2 + x_3^2$$

$$F(x_1, x_2) = \frac{d}{d-\rho}x_1^2 + x_2^2$$

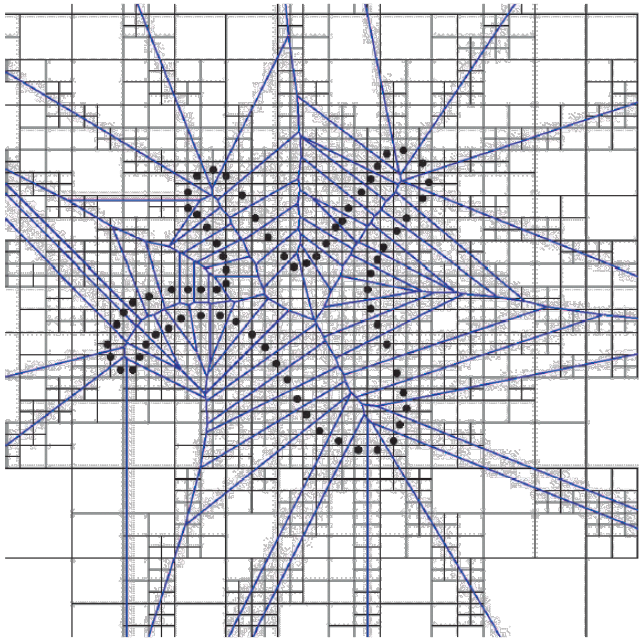
curvature

- perform iterative gradient-descent in this field
- point to foot-point distance
  - case *d* is large: classical ICP
  - case *d* is small: point-to-plane ICP

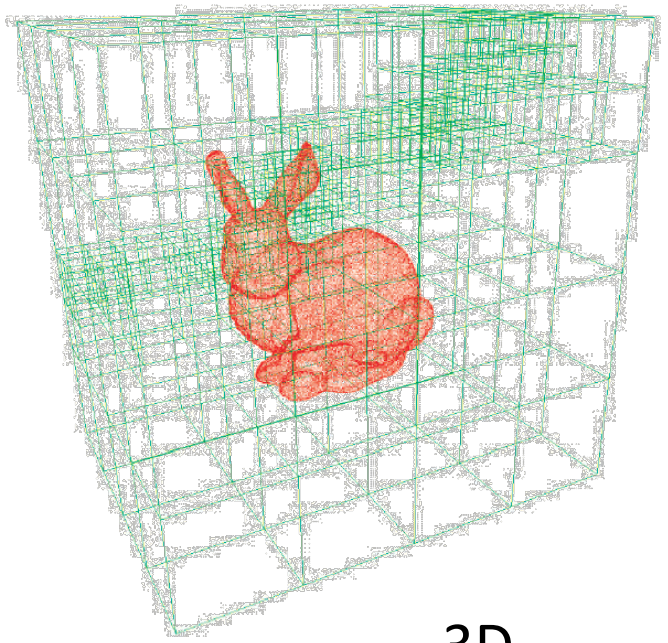


$d^2(y, \Phi_p)$ Using d2 Tree

Partition the space into cells where each cell stores a quadratic approximant of the squared distance function.



2D

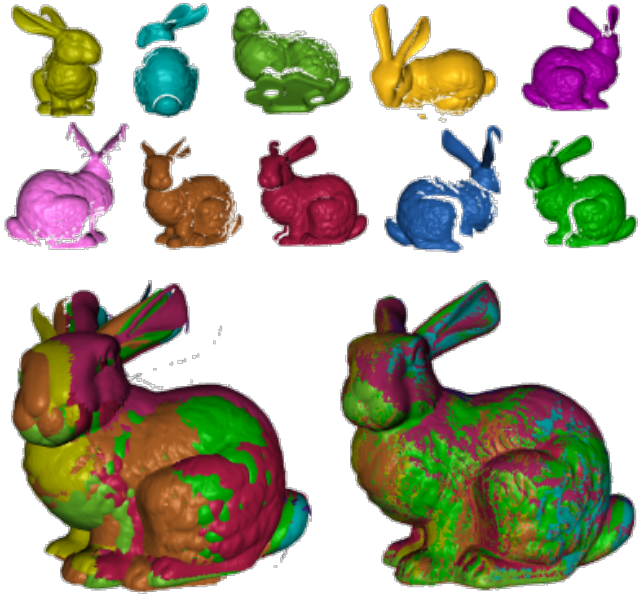


3D

Leopoldseder S et al. *d2-tree: A hierarchical representation of the squared distance function*

Global Matching

Given shapes in *arbitrary* positions, find their alignment:



Robust Global Registration
Gelfand et al. SGP 2005

Can be approximate, since will refine later using e.g. ICP

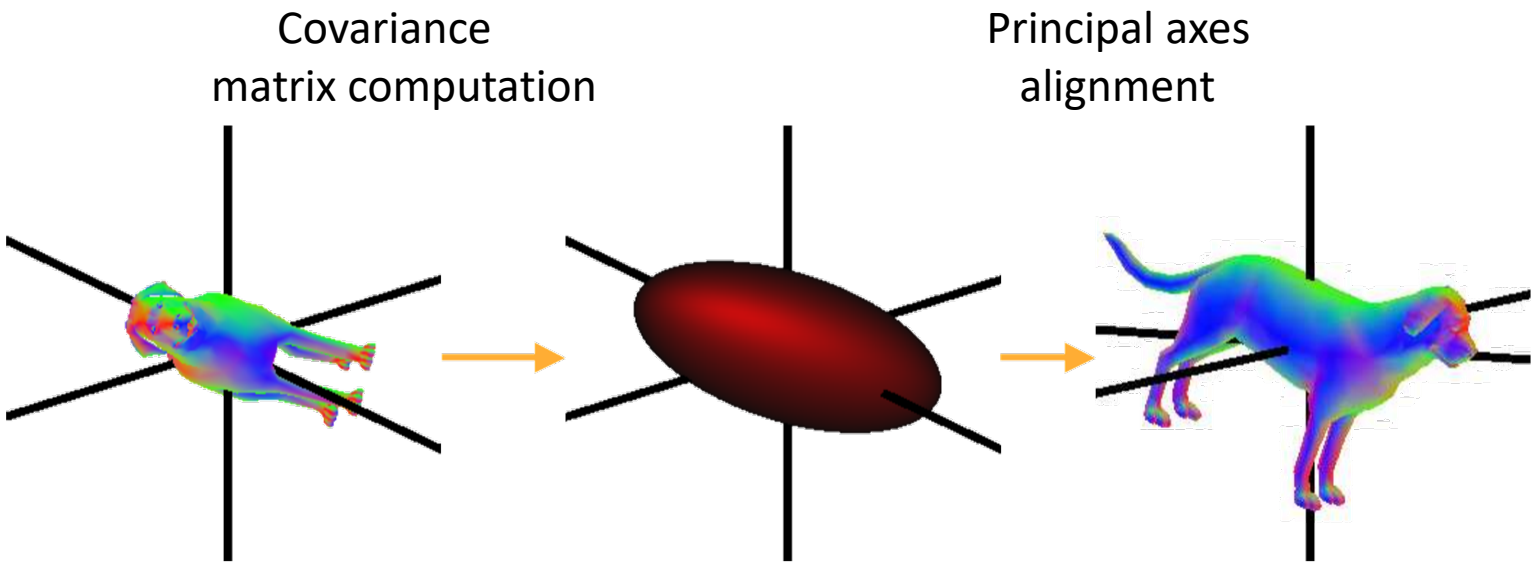
Global Matching – Approaches

Several classes of approaches:

1. Exhaustive Search
2. Normalization (PCA)
3. Random Sampling (RANSAC)
4. Invariant Features

PCA-Based Alignment

- Use PCA to place models into canonical coordinate frames
- Then align those frames

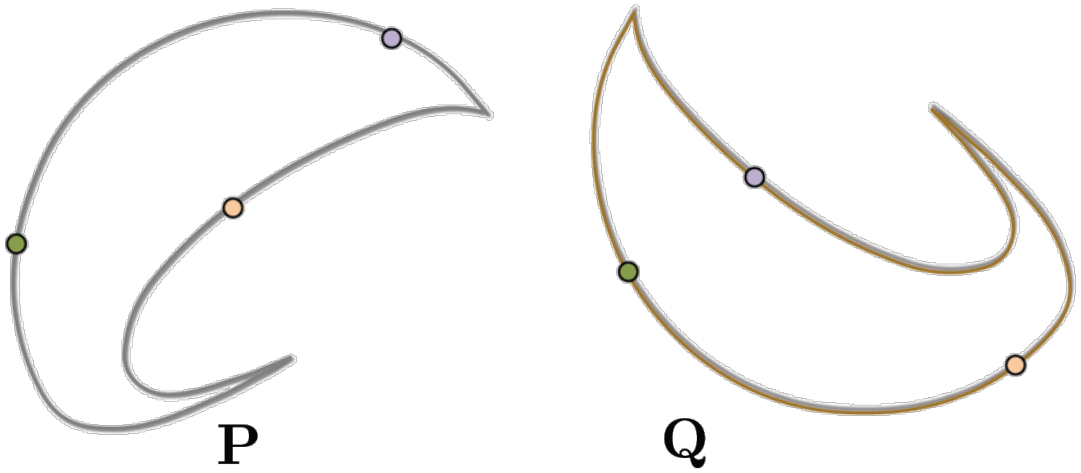


Random Sampling (RANSAC)

ICP only needs 3 point pairs!

Robust and simple approach. Iterate between:

1. Pick a random pair of 3 points on model & scan
2. Estimate alignment, and check for error.



Guess and verify

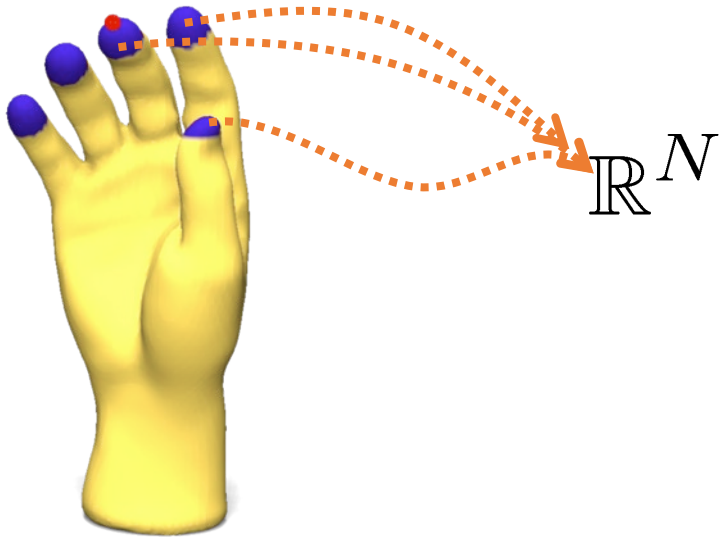
Global Matching – Invariant Features

Try to characterize the shape using properties that are invariant under the desired set of transformations.

Conflicting interests – invariance vs. informativeness.

The most common pipeline:

1. identify salient feature points
2. compute informative and commensurable descriptors.



Spin Images

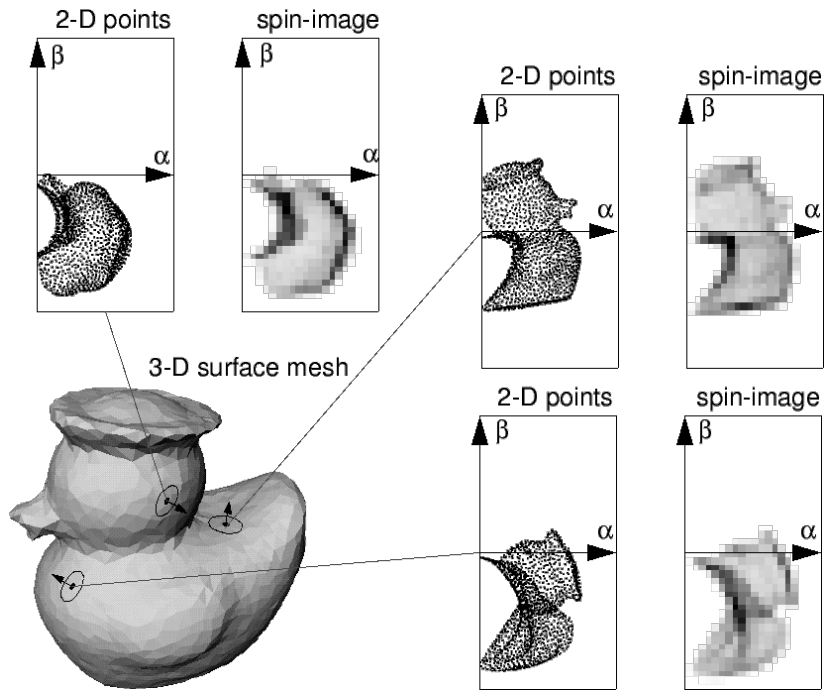
Creates an image associated with a neighborhood of a point.

Compare points by comparing their *spin images* (2D).

Given a point and a normal, every other point is indexed by two parameters:

β distance to tangent plane

α distance to normal line

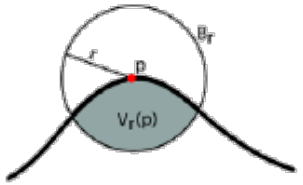


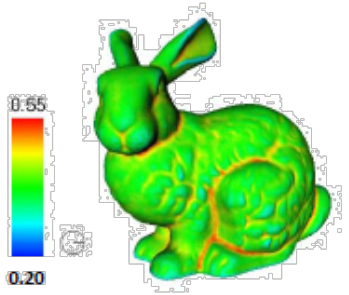
Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes
Johnson et al, PAMI 99

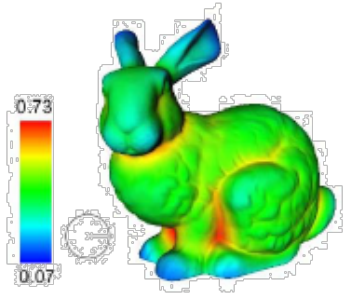
Integral Volume Descriptor

Integral invariant signatures, Manay et al. ECCV 2004
Integral Invariants for Robust Geometry Processing, Pottmann et al. 2007-2009

$$V_r(p) = \int_{B_r(p) \cap S} dx$$







Relation to mean curvature

$$V_r(\mathbf{p}) = \frac{2\pi}{3} r^3 - \frac{\pi H}{4} r^4 + O(r^5)$$

Robust Global Registration, Gelfand et al. 2005

Registration Method Taxonomy

Local vs. Global

refinement (e.g. ICP) | alignment (search)

Rigid vs. Deformable

rotation, translation | general deformation

Pair vs. Collection

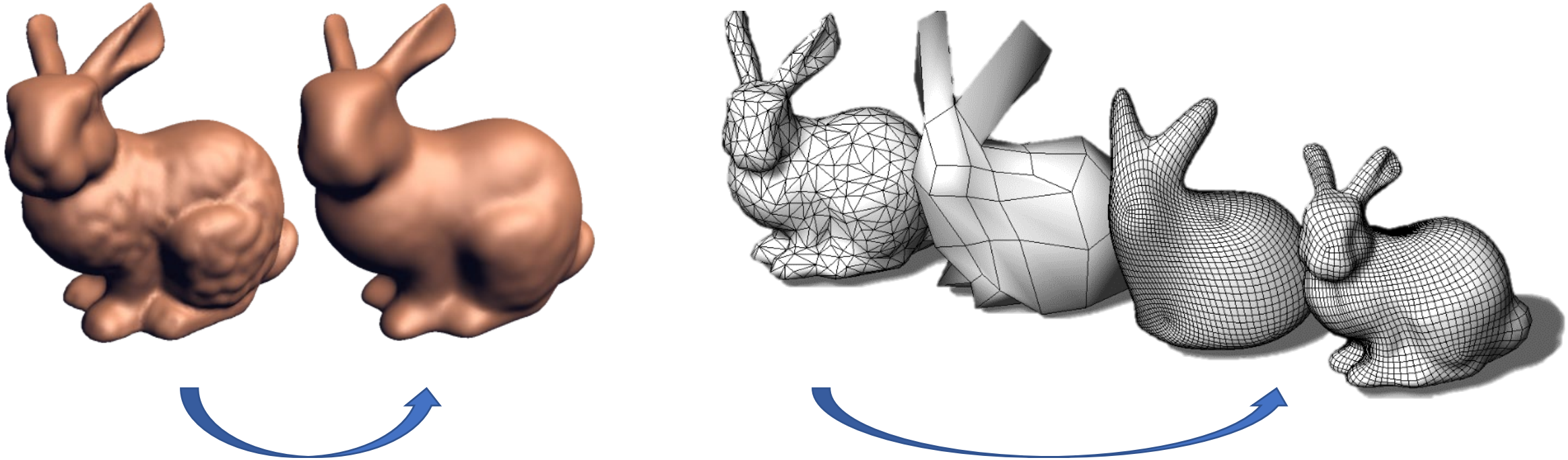
two shapes | multiple shapes

Today:
Shape Smoothing, Parametrization, and Remeshing

Outline

- **Smoothing Motivation**
- Spectral Analysis
- Diffusion Flow
- Remeshing Motivation
- Parametrization
- Global parametrization remeshing
- Direct surface remeshing

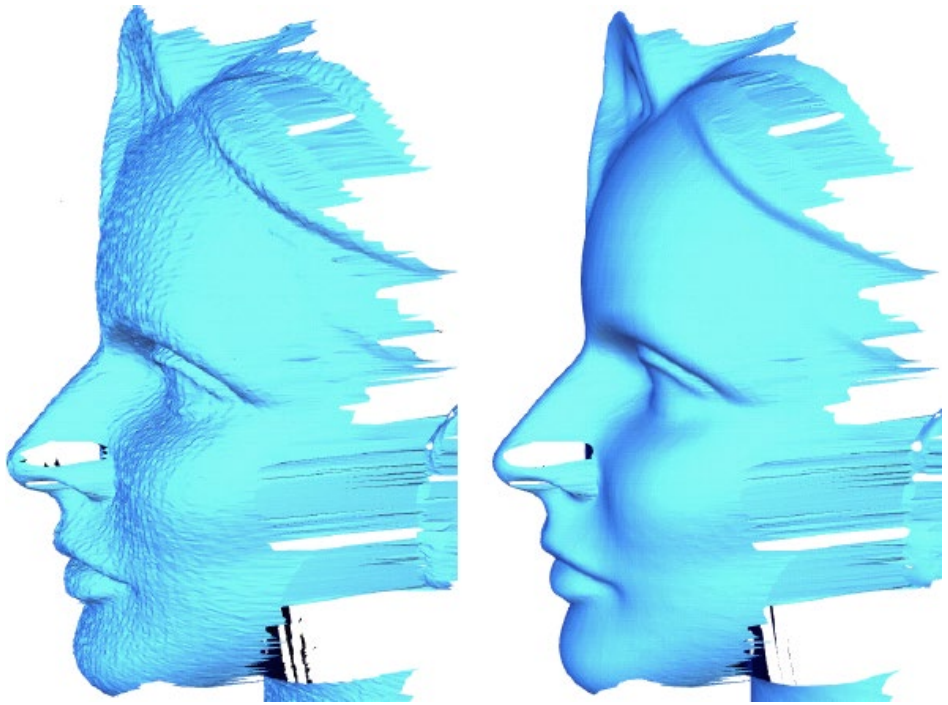
Mesh Smoothing & Remeshing



[slides via Mario Botsch]

Mesh Smoothing: Motivation

- Filter out high frequency noise



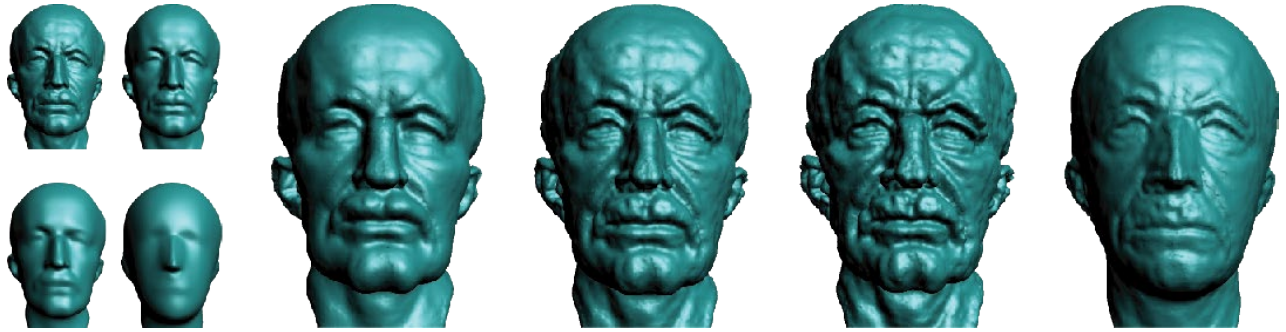
Desbrun, Meyer, Schroeder, Barr: *Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow*, SIGGRAPH 99

Motivation -- Denoising

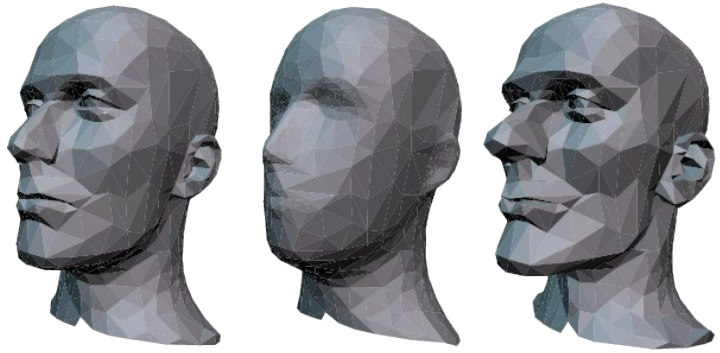
- Filter out high frequency noise

Motivation – Multi-Resolution Editing

- Advanced filtering



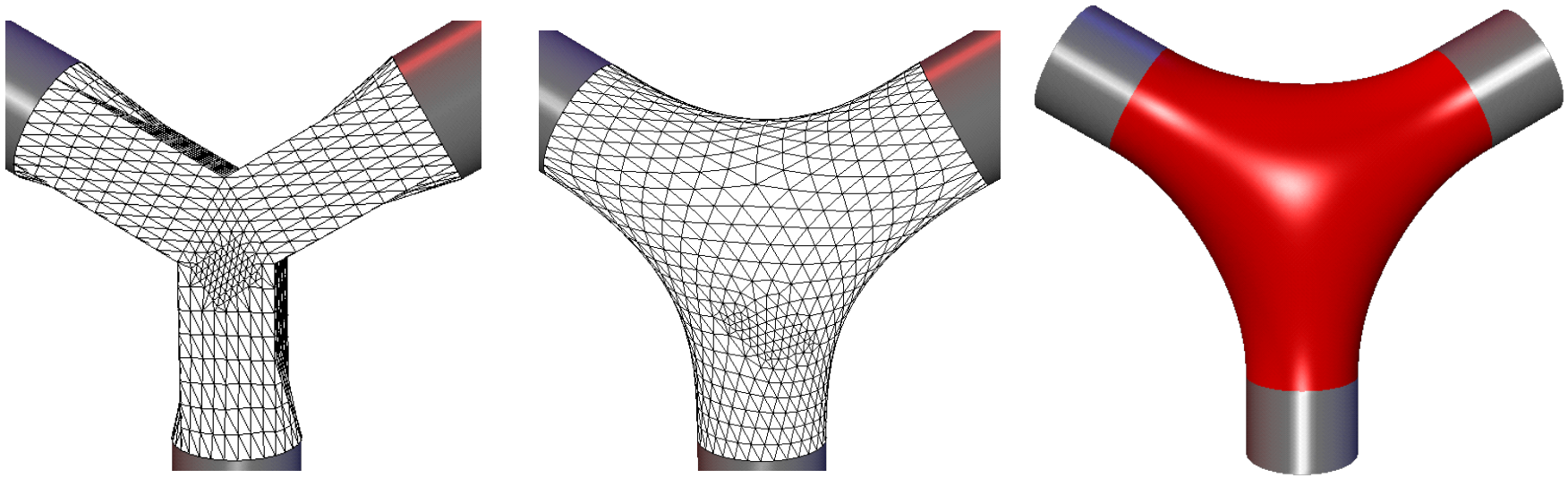
Pauly, Kobbelt, Gross: *Point-Based Multi-Scale Surface Representation*, ACM TOG 2006



Guskow, Sweldens, Schroeder: *Multiresolution Signal Processing for Meshes*, SIGGRAPH 99

Motivation: Surface Fairing

- Fair Surface Design



Schneider, Kobbelt: *Geometric fairing of irregular meshes for free-form surface design*, CAGD 18(4), 2001

Outline

- Smoothing Motivation
- **Spectral Analysis**
- Diffusion Flow
- Remeshing Motivation
- Parametrization
- Global parameterization
- Direct surface remeshing

Fourier Transform

- Spatial domain $f(x) \longrightarrow$ 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 $F(u) \longrightarrow$ Spatial domain $f(x)$

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

Fourier Transform

- Consider L^2 function space with inner product

$$\langle f, g \rangle := \int_{-\infty}^{\infty} f(x)\,\overline{g(x)}\,\mathrm{d}x$$

- Complex “waves” build an orthonormal basis

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

- Fourier transform is a change of basis

$$f(x) = \int_{-\infty}^{\infty} \langle f, e_u \rangle\; e_u\,\mathrm{d}u$$

Extend Fourier to Meshes?

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

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

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

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

- Use Eigenfunctions of a discrete Laplace-Beltrami operator (generalization of Laplace to meshes)

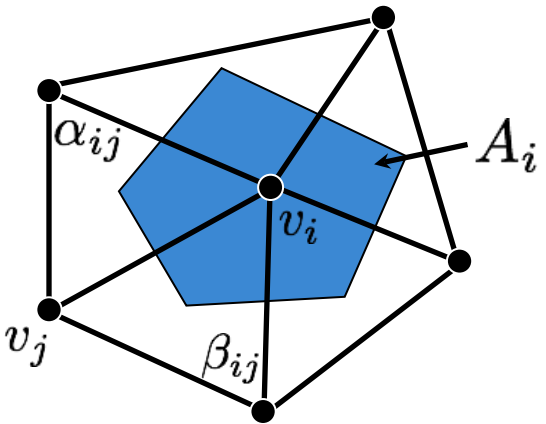
Discrete Laplace-Beltrami

- Discrete function sampled at mesh vertices

$$\mathbf{f} = [f_1, f_2, \ldots, f_n] \in \mathbb{R}^n$$

- Discrete Laplace-Beltrami (per vertex)

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

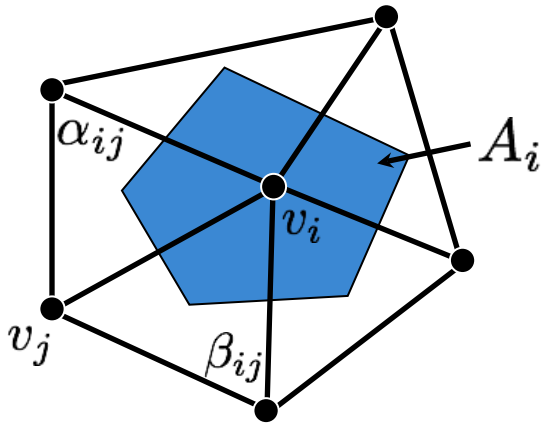


Discrete Laplace-Beltrami

- Discrete Laplace *operator* (per mesh)
  - Sparse matrix

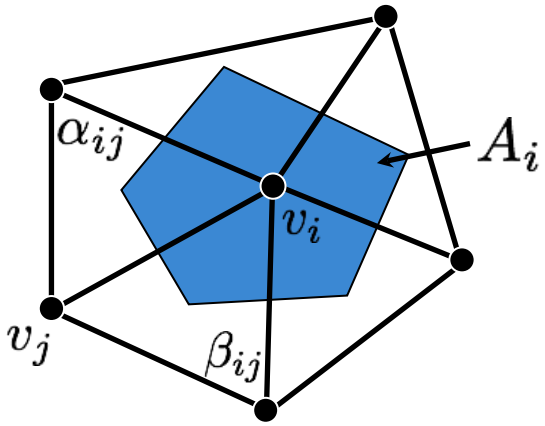
$$\begin{pmatrix} \vdots \\ \Delta_S f(v_i) \\ \vdots \end{pmatrix} = \mathbf{L} \cdot \begin{pmatrix} \vdots \\ f(v_i) \\ \vdots \end{pmatrix}$$

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



Discrete Laplace-Beltrami

- Discrete Laplace *operator* (per mesh)
  - Sparse matrix



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

$$\mathbf{D} = \mathrm{diag}\left(\ldots, \frac{1}{2Ai}, \ldots \right)$$

[inverse of area matrix *A*]

$$\mathbf{M}_{ij} = \begin{cases} \cot\alpha_{ij} + \cot\beta_{ij}, & i \neq j\,,\ j \in \mathcal{N}_1(v_i) \\ -\sum_{v_j \in \mathcal{N}_1(v_i)} (\cot\alpha_{ij} + \cot\beta_{ij}) & i = j \\ 0 & \text{otherwise} \end{cases}$$

[stiffness matrix *M*]

Spectral Analysis of Discrete Laplace-Beltrami

- Discrete function sampled at mesh vertices

$$\mathbf{f} = [f_1, f_2, \ldots, f_n] \in \mathbb{R}^n$$

- Discrete Laplace-Beltrami (per vertex)

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

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

- Discrete Laplace-Beltrami matrix $\mathbf{L}$
  - Eigenvectors are “natural vibrations”

Spectral Reconstruction

1. Setup Laplace-Beltrami matrix $\mathbf{L}$
2. Compute k smallest eigenvectors $\{\mathbf{e}_1,\ldots,\mathbf{e}_k\}$
3. Reconstruct mesh from those (component-wise)

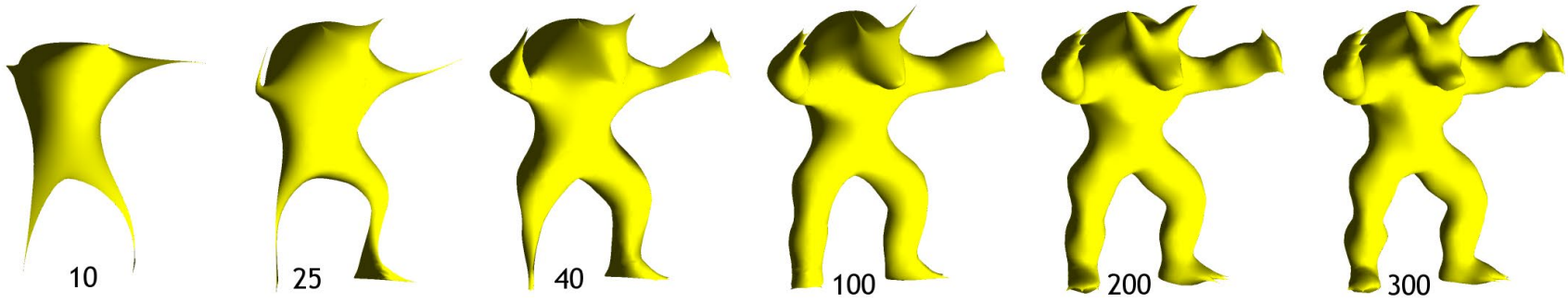
$$\mathbf{x} := [x_1,\ldots,x_n] \qquad \mathbf{y} := [y_1,\ldots,y_n] \qquad \mathbf{z} := [z_1,\ldots,z_n]$$

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

Spectral Reconstruction

1. Setup Laplace-Beltrami matrix $\mathbf{L}$
2. Compute k smallest eigenvectors $\{\mathbf{e}_1,\ldots,\mathbf{e}_k\}$
3. Reconstruct mesh from those

Too complex for large meshes! (cubic in *n*)



Bruno Levy: *Laplace-Beltrami Eigenfunctions: Towards an algorithm that understands geometry*, Shape Modeling and Applications, 2006

Outline

- Smoothing Motivation
- Spectral Analysis
- **Diffusion Flow**
- Remeshing Motivation
- Parametrization
- Global parametrization remeshing
- Direct surface remeshing

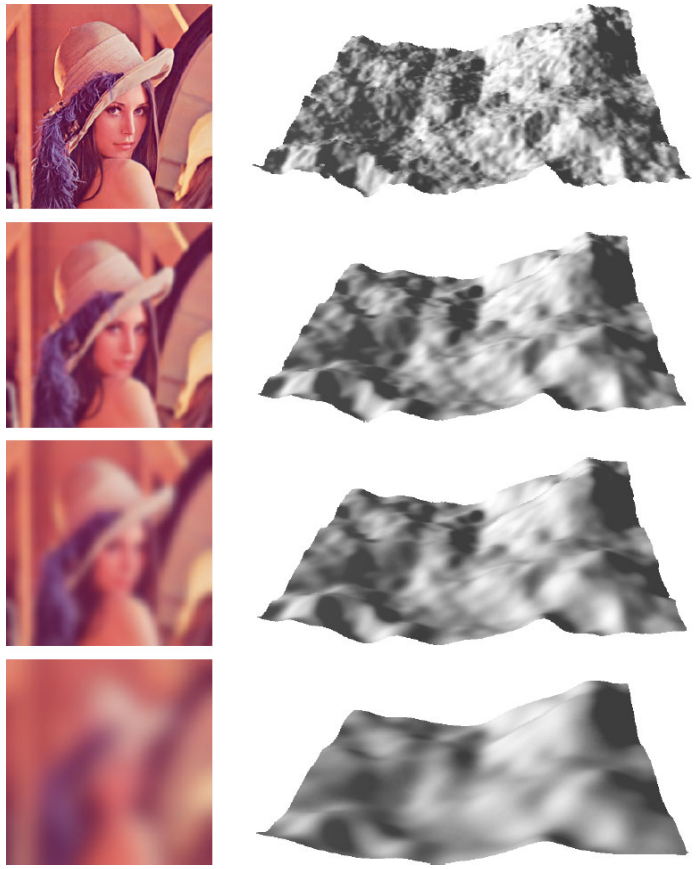
Diffusion Flow on Height Fields

- Diffusion equation

diffusion constant

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

Laplace operator

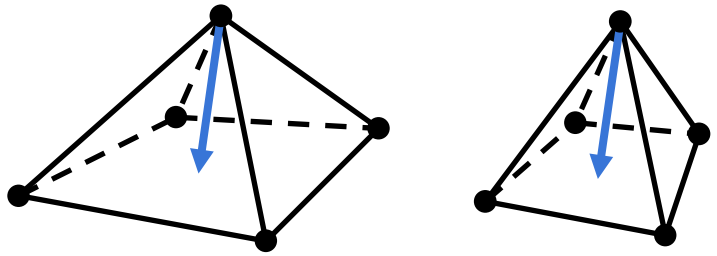


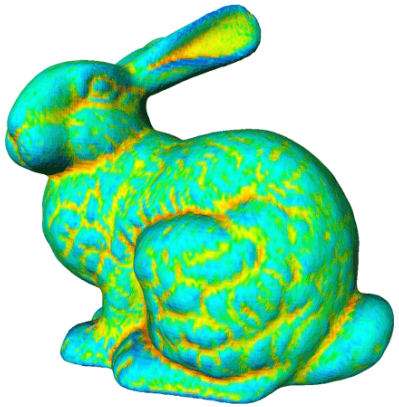
Diffusion Flow on Meshes

- Iterate

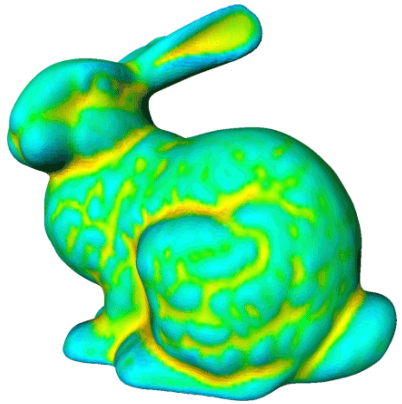
$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \lambda \Delta \mathbf{p}_i$$

[Δ: Mesh graph Laplacian (“uniform Laplace”)]

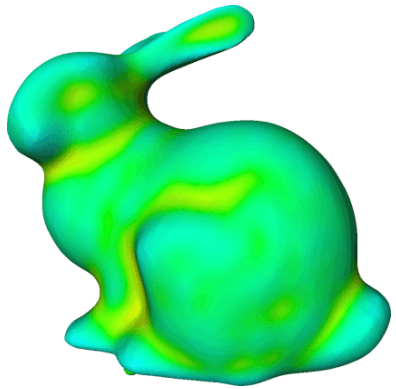




0 Iterations



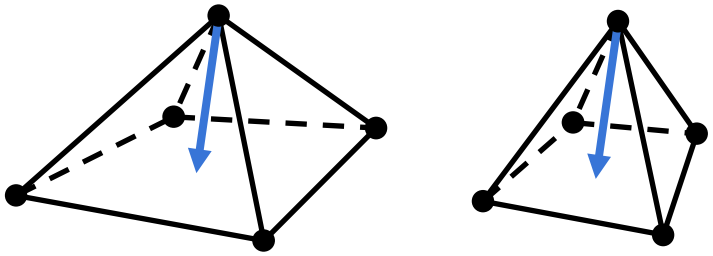
5 Iterations

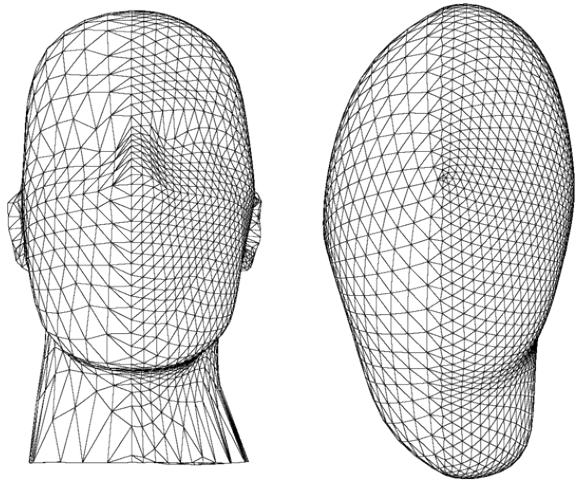


20 Iterations

Uniform Laplace Discretization

- 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

Curvature Flow (Laplace-Beltrami)

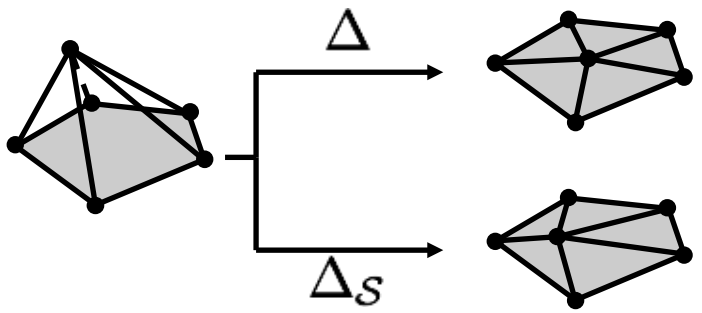
- Use diffusion flow with Laplace-Beltrami

$$\frac{\partial \mathbf{p}}{\partial t} = \lambda \Delta_S \mathbf{p}$$

- Laplace-Beltrami is parallel to surface normal

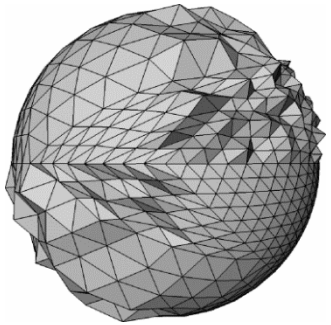
$$\frac{\partial \mathbf{p}}{\partial t} = -2\lambda H \mathbf{n}$$

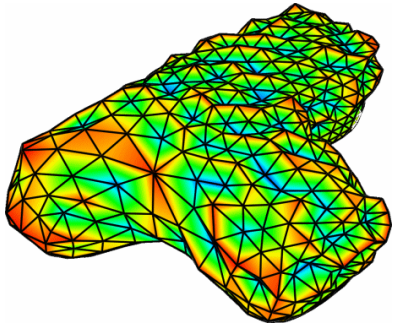
→ Avoids vertex drift on surface



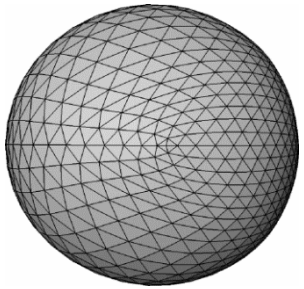
Comparison

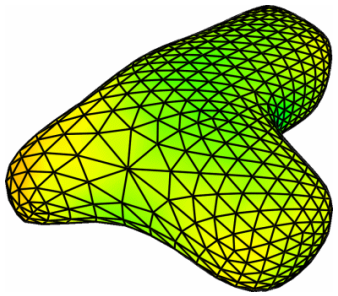
Original



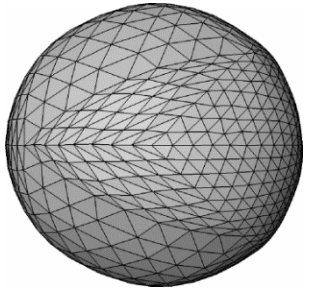


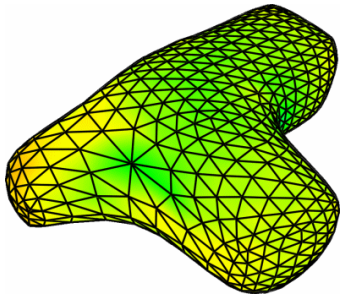
Uniform Laplace





Laplace-Beltrami





Numerical Integration

- Write update $\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} + \lambda \Delta \mathbf{p}_i^{(t)}$ in matrix notation with

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

- Corresponds to *explicit* integration

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

Requires small λ for stability!

- Implicit integration is unconditionally stable

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

Implementation

- Solve linear system each iteration

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

- Matrix $\mathbf{L} = \mathbf{DM}$ is not symmetric because of multiplication by $\mathbf{D}$
  - → Symmetrize by multiplying $\mathbf{D}^{-1}$ from left

$$\left(\mathbf{D}^{-1} - \lambda \mathbf{M}\right) \mathbf{P}^{(t+1)} = \mathbf{D}^{-1} \mathbf{P}^{(t)}$$

- Solve sparse symmetric positive definite system
  - Iterative conjugate gradients, sparse Cholesky

References

- Levy: *Laplace-Beltrami Eigenfunctions: Towards an algorithm that understands geometry*, Shape Modeling and Applications, 2006
- Taubin: *A signal processing approach to fair surface design*, SIGGRAPH 1996
- Desbrun, Meyer, Schroeder, Barr: *Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow*, SIGGRAPH 1999

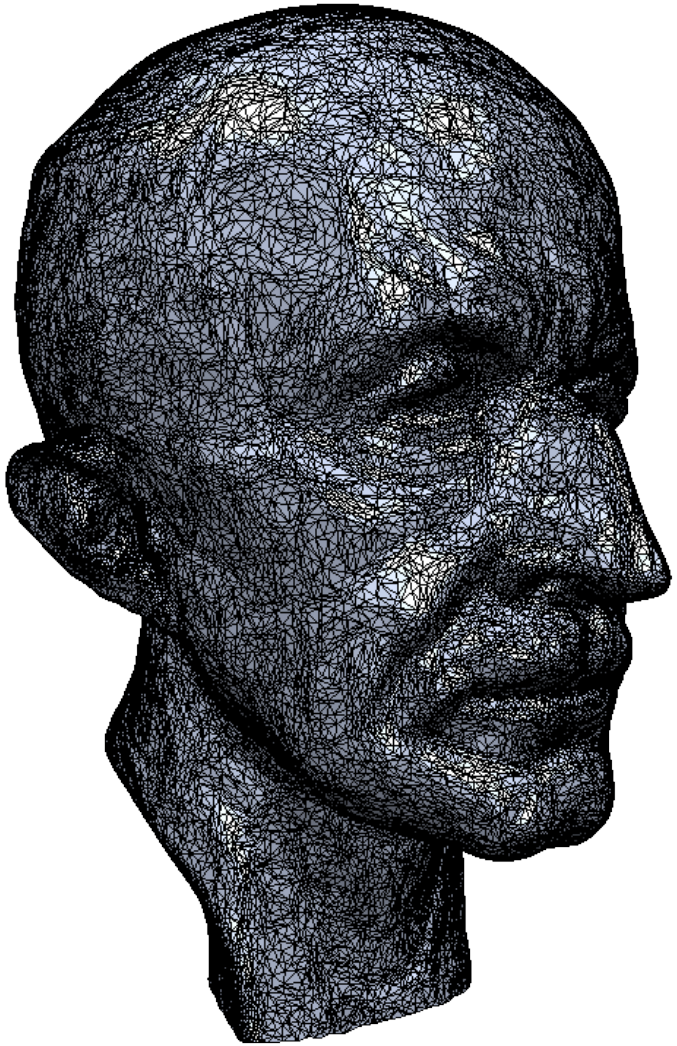
Outline

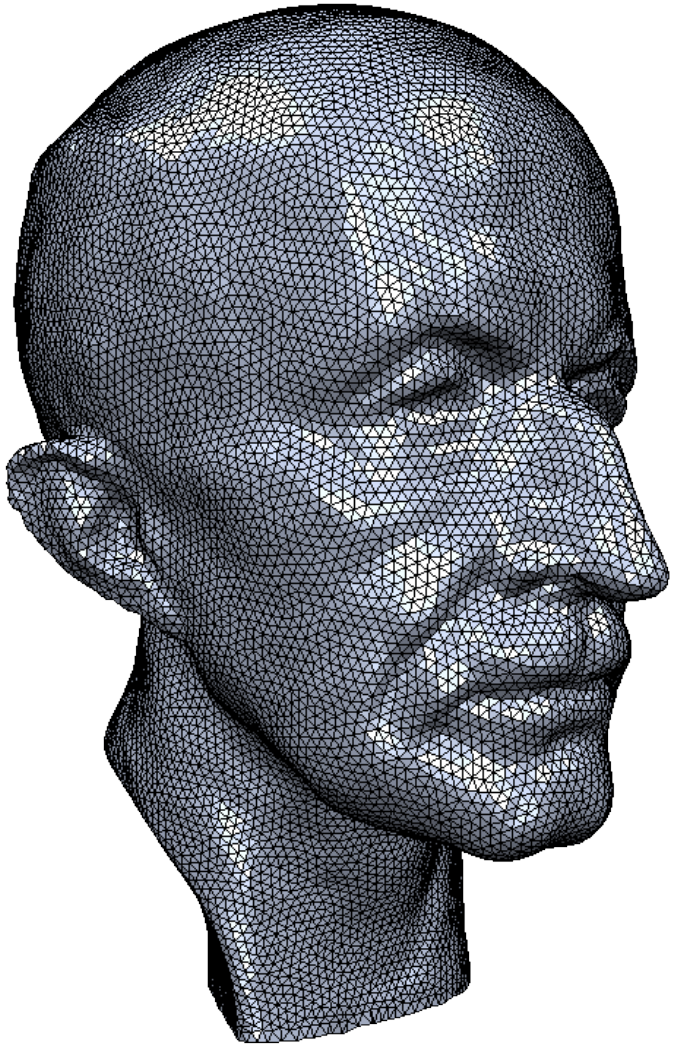
- Smoothing Motivation
- Spectral Analysis
- Diffusion Flow
- **Remeshing Motivation**
- Parametrization
- Global parametrization remeshing
- Direct surface remeshing

Remeshing

"Given a 3D mesh, improve its triangulation
while preserving its geometry."

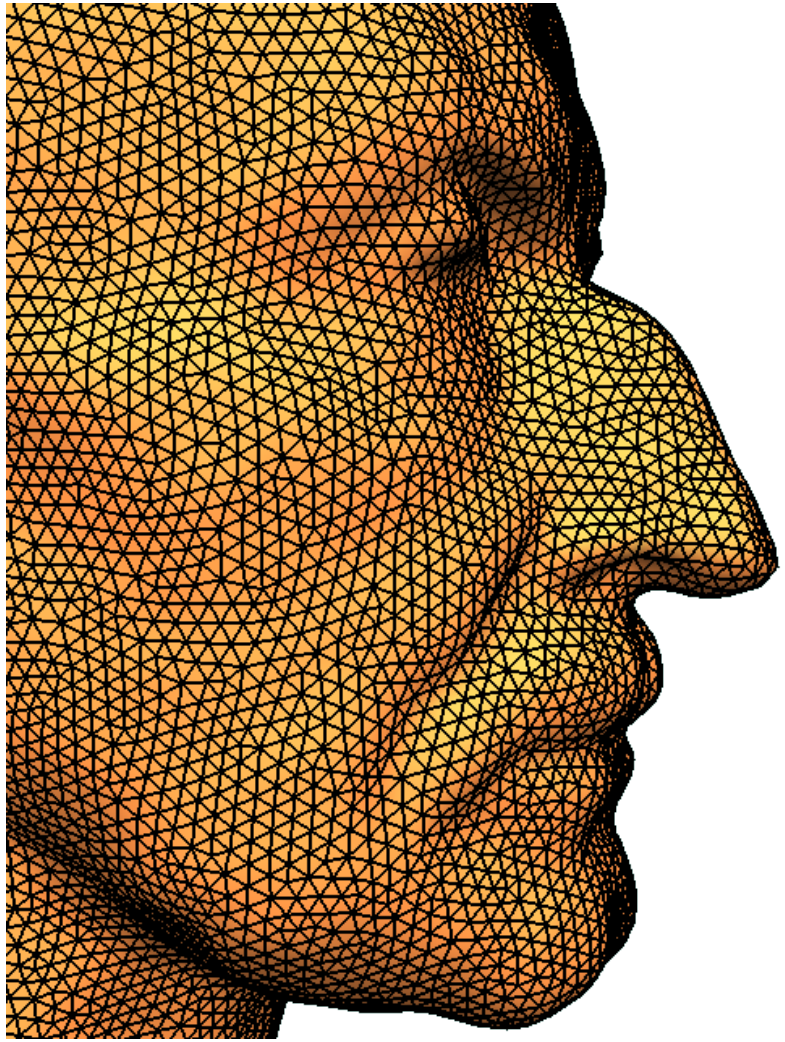
Review: What is a good mesh?





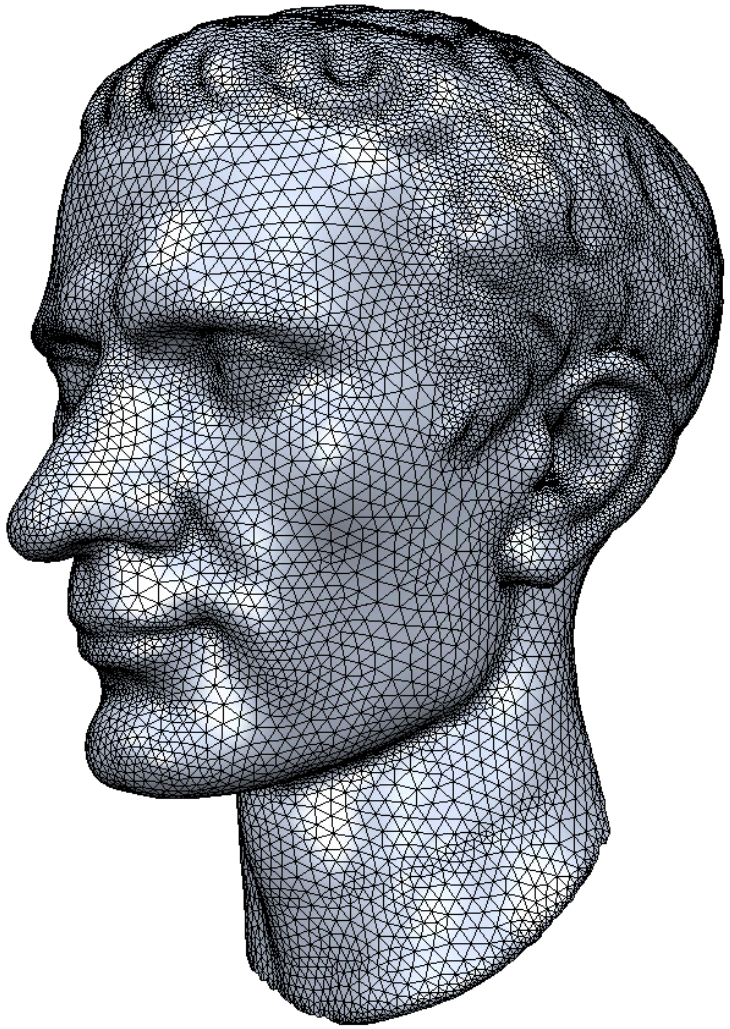
What is a good mesh?

- Equal edge lengths
- Equilateral triangles
- Valence close to 6



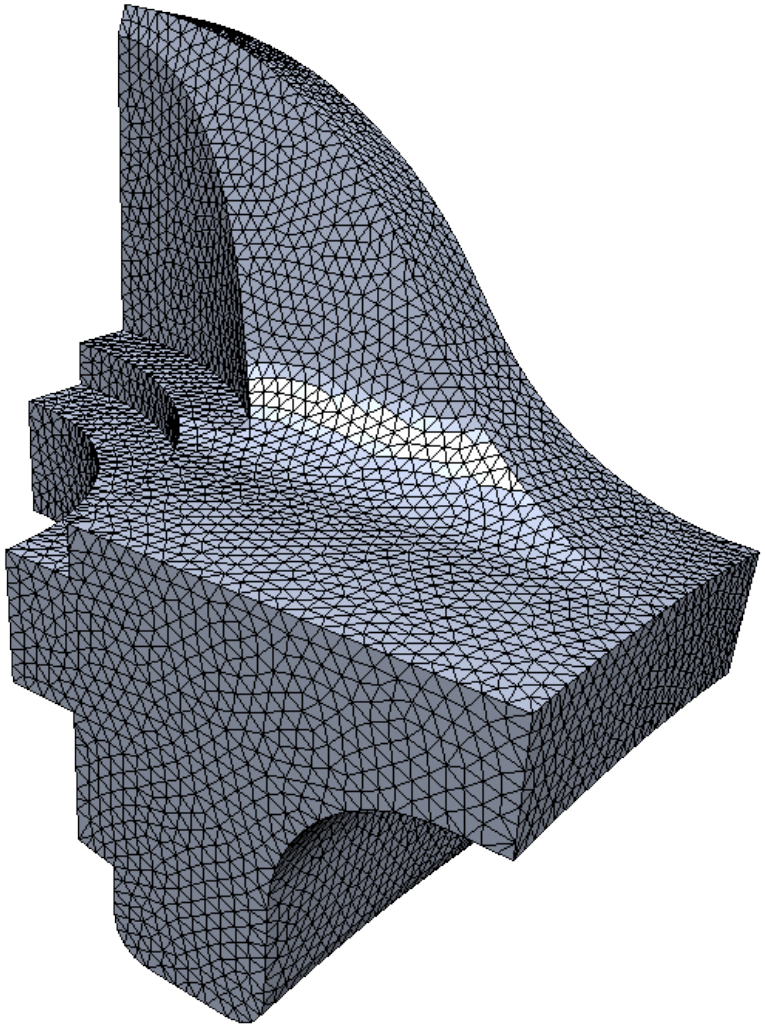
What is a good mesh?

- Equal edge lengths
- Equilateral triangles
- Valence close to 6
- Uniform vs. adaptive sampling



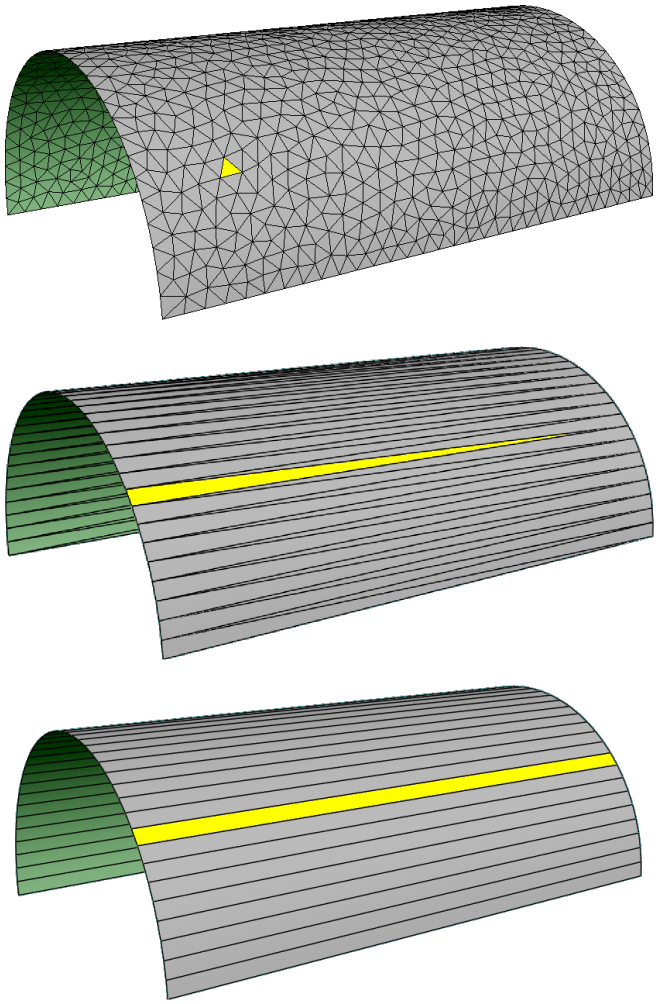
What is a good mesh?

- Equal edge lengths
- Equilateral triangles
- Valence close to 6
- Uniform vs. adaptive sampling
- Feature preservation



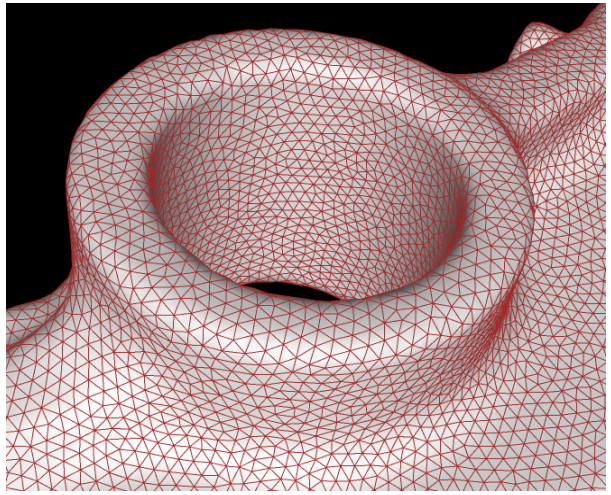
What is a good mesh?

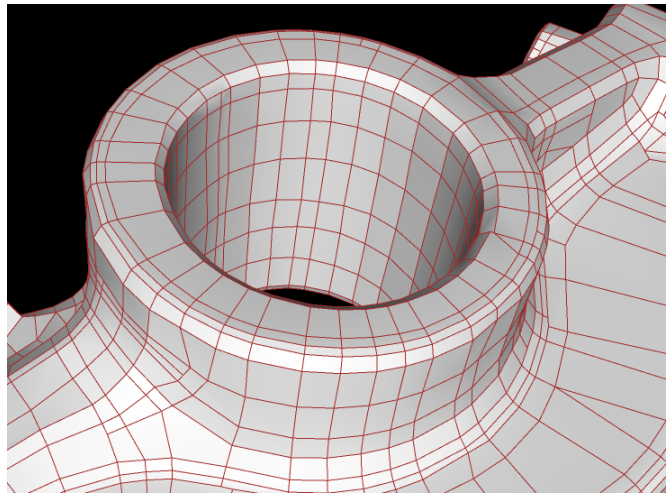
- Equal edge lengths
- Equilateral triangles
- Valence close to 6
- Uniform vs. adaptive sampling
- Feature preservation
- Alignment to curvature lines
- Isotropic vs. anisotropic
- Triangles vs. quadrangles



What is a good mesh?

- 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





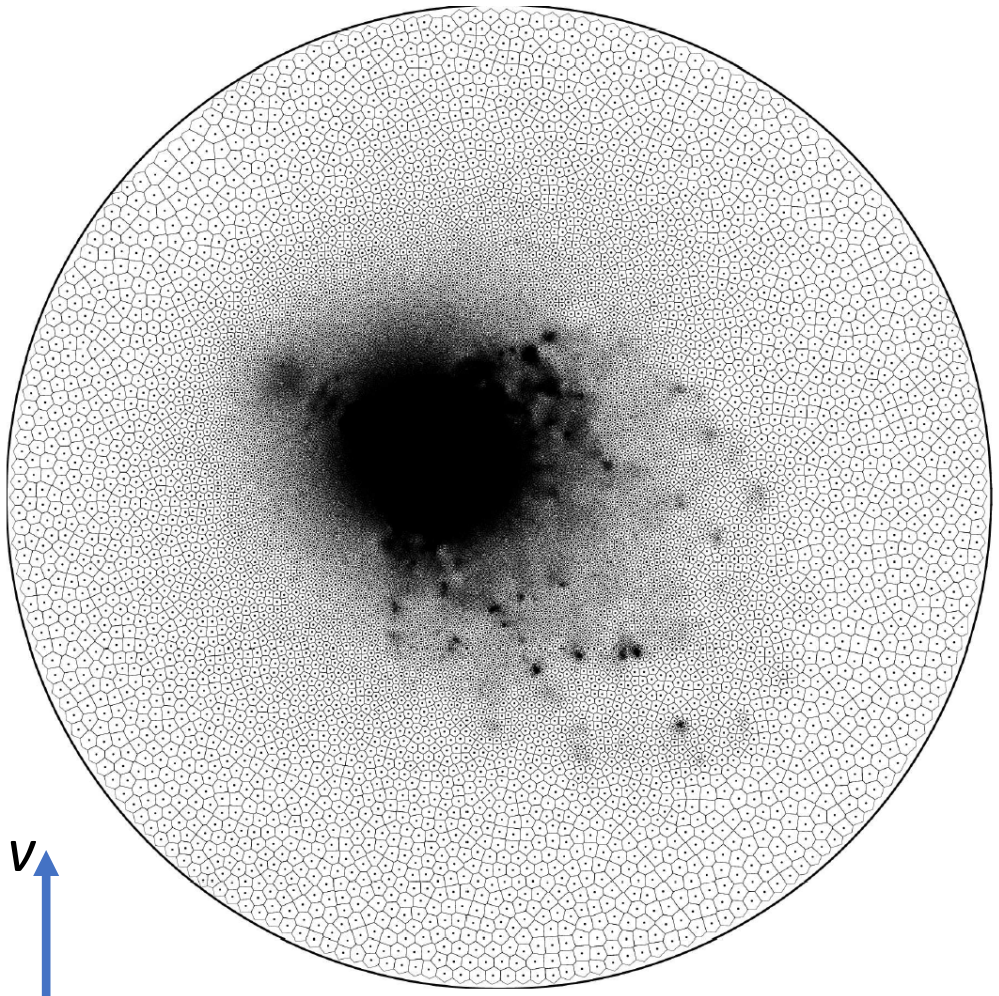
Two Fundamental Remeshing Approaches

- 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

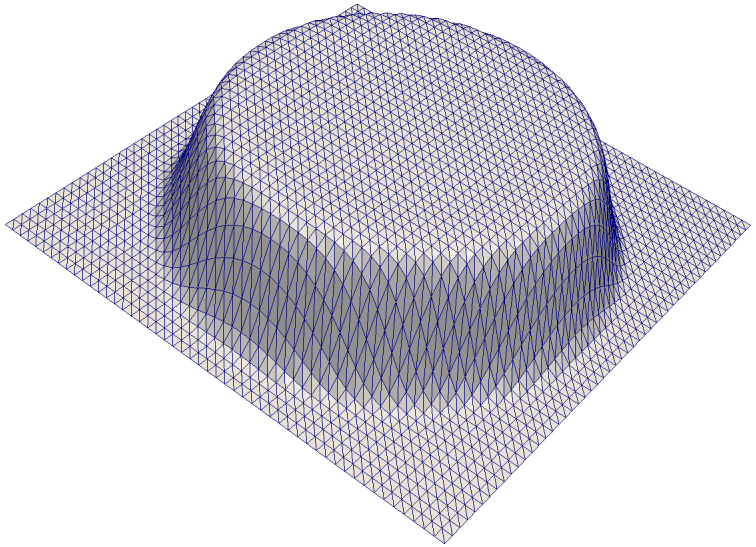
Outline

- Smoothing Motivation
- Spectral Analysis
- Diffusion Flow
- Remeshing Motivation
- **Parametrization**
- Global parametrization remeshing
- Direct surface remeshing

Parametrization

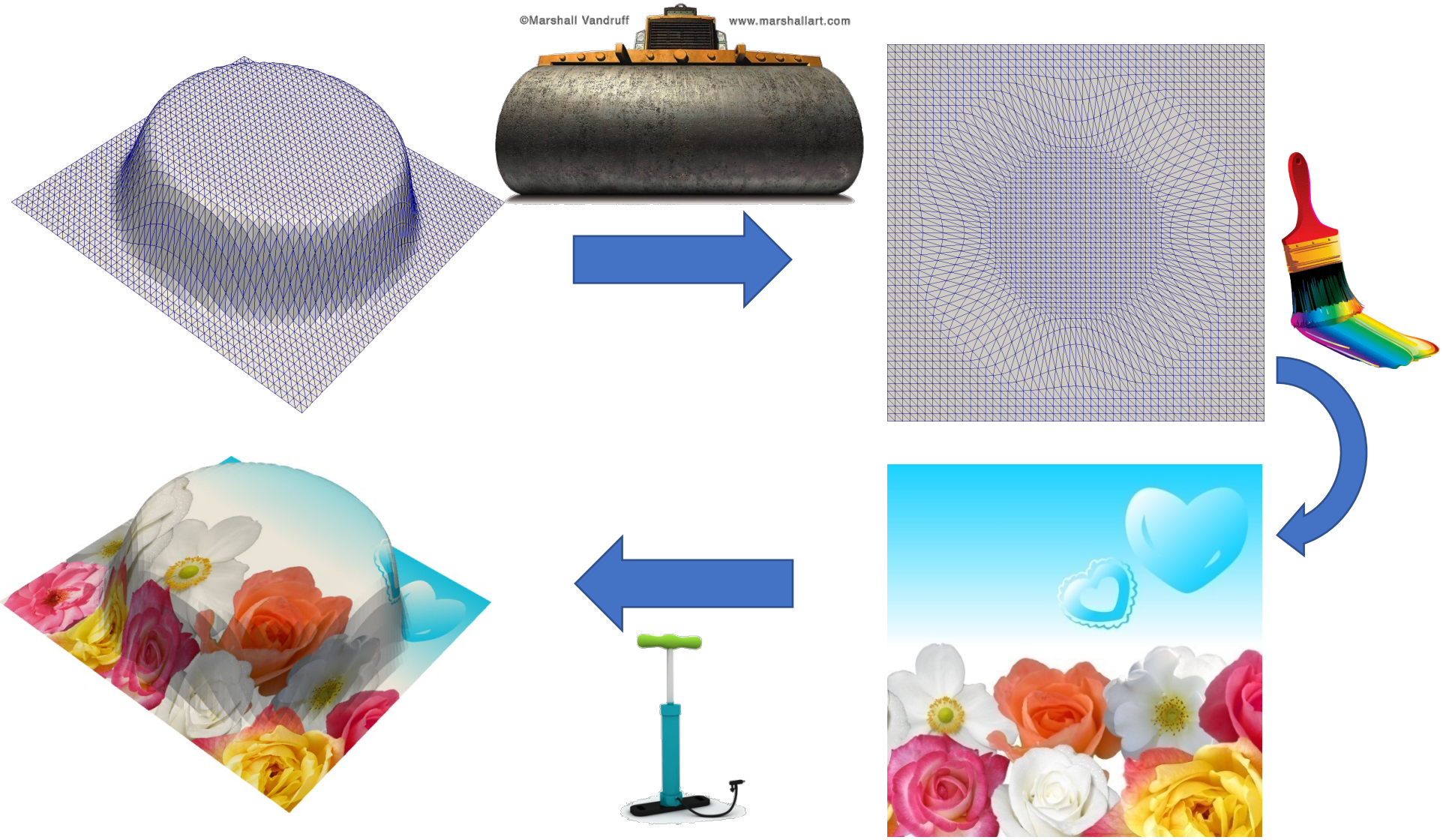


Parametrization



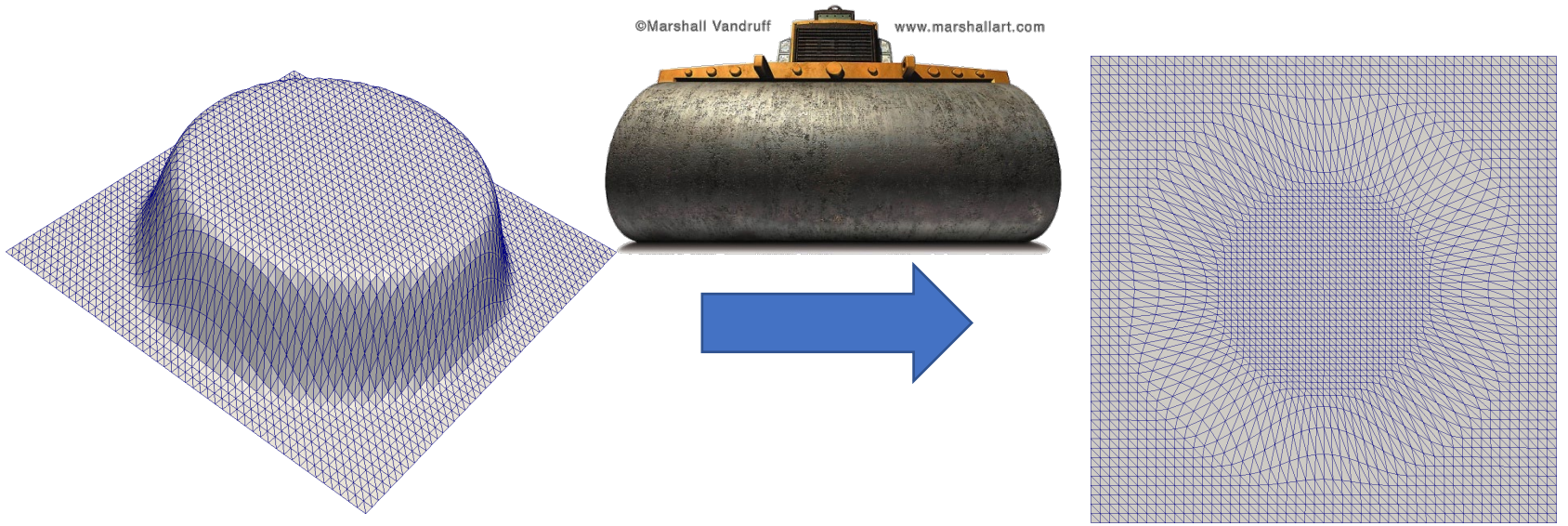
?

Solution

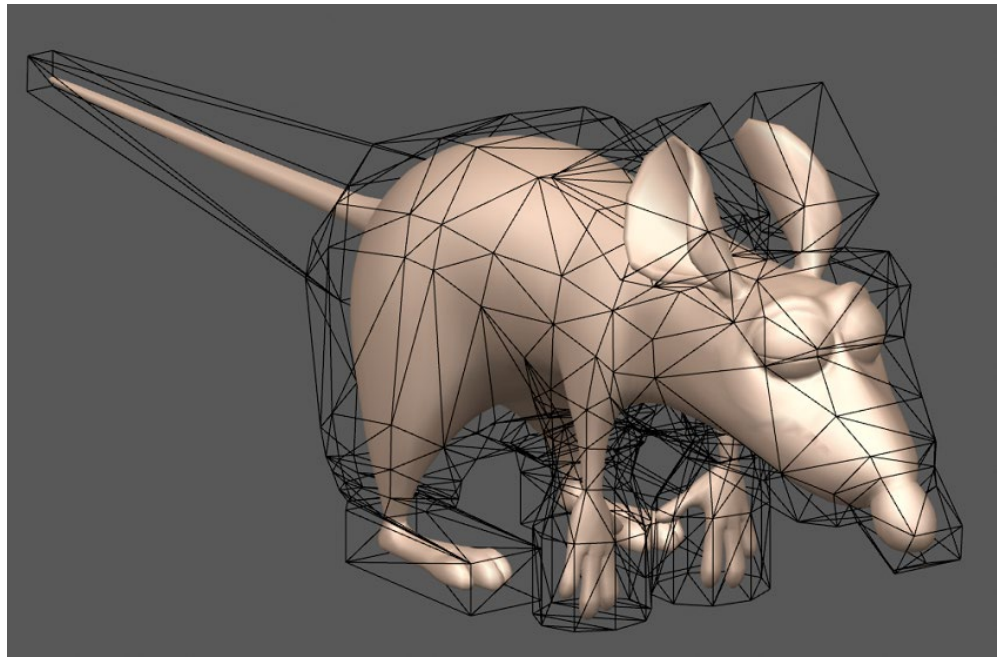


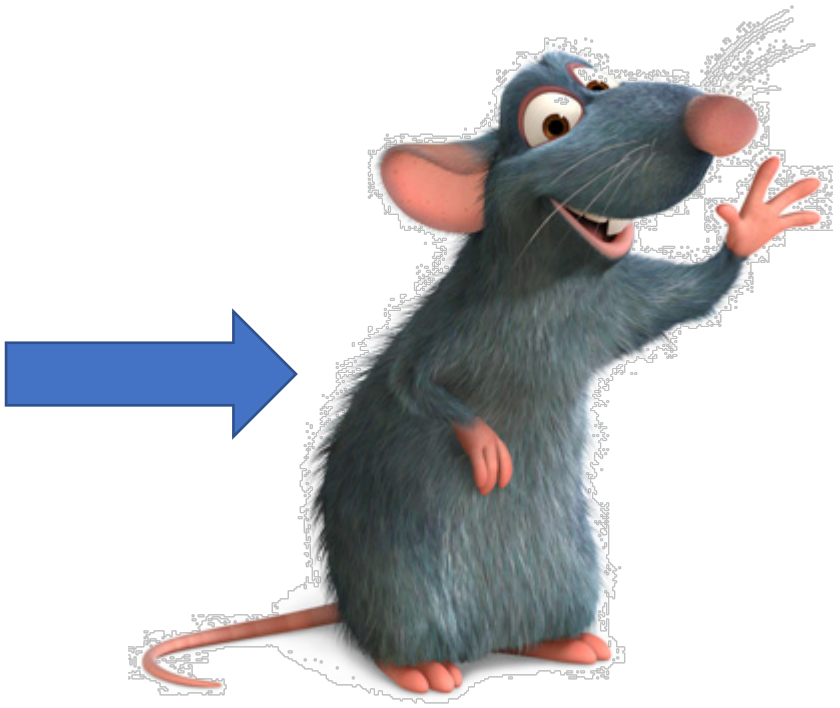
Flattening Surfaces

Parameterization is …



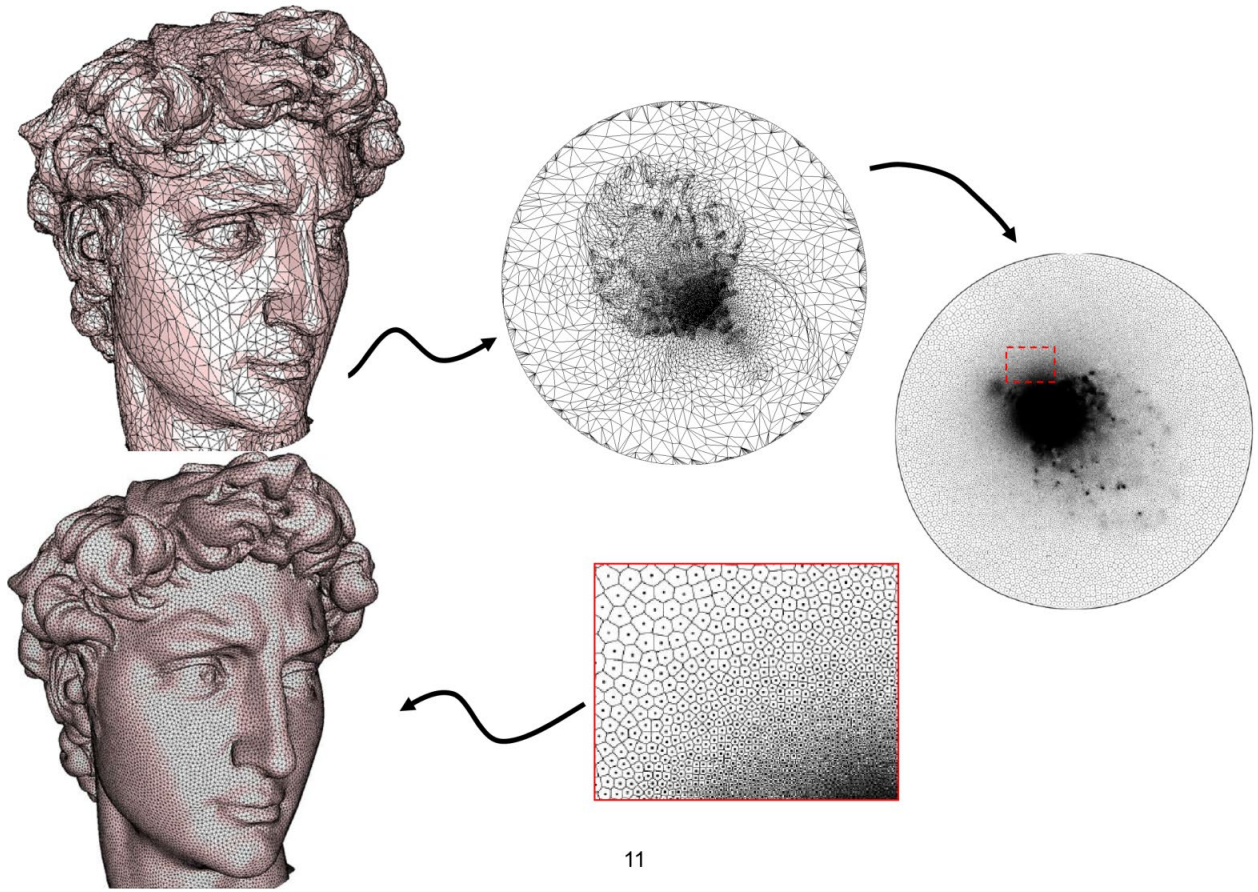
Why Parametrize?





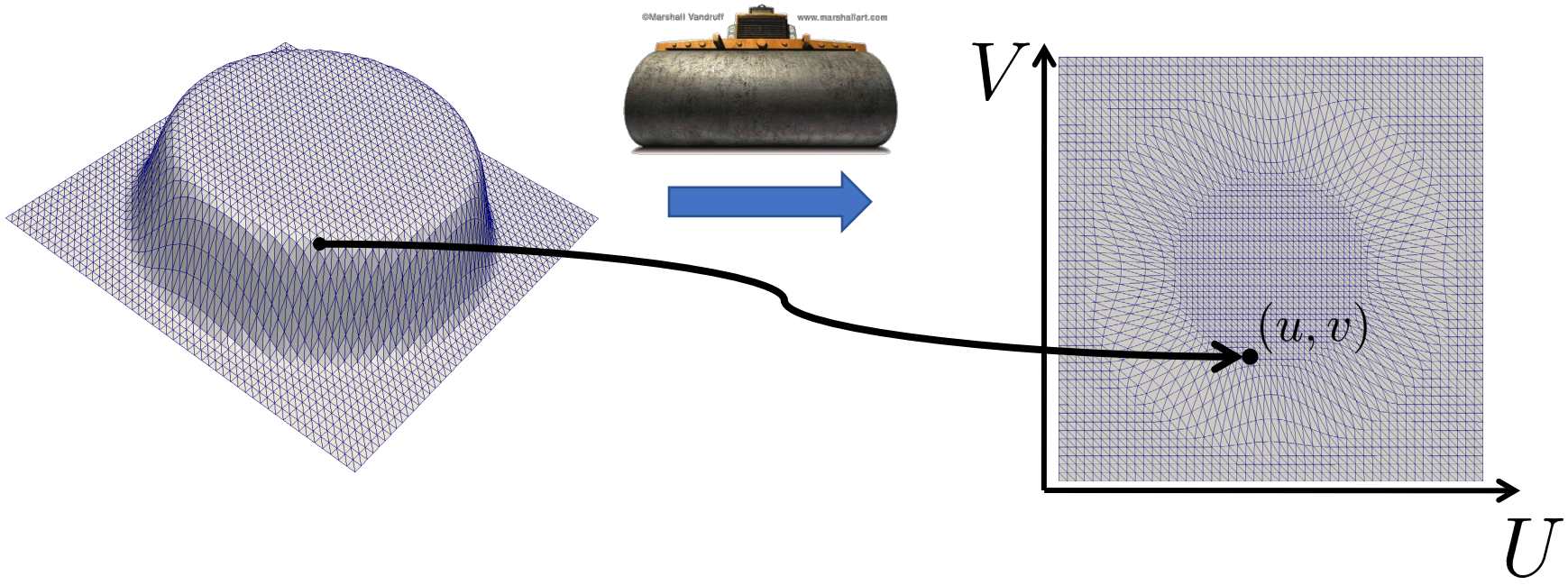
Texture Mapping

Why Parametrize?



Remeshing

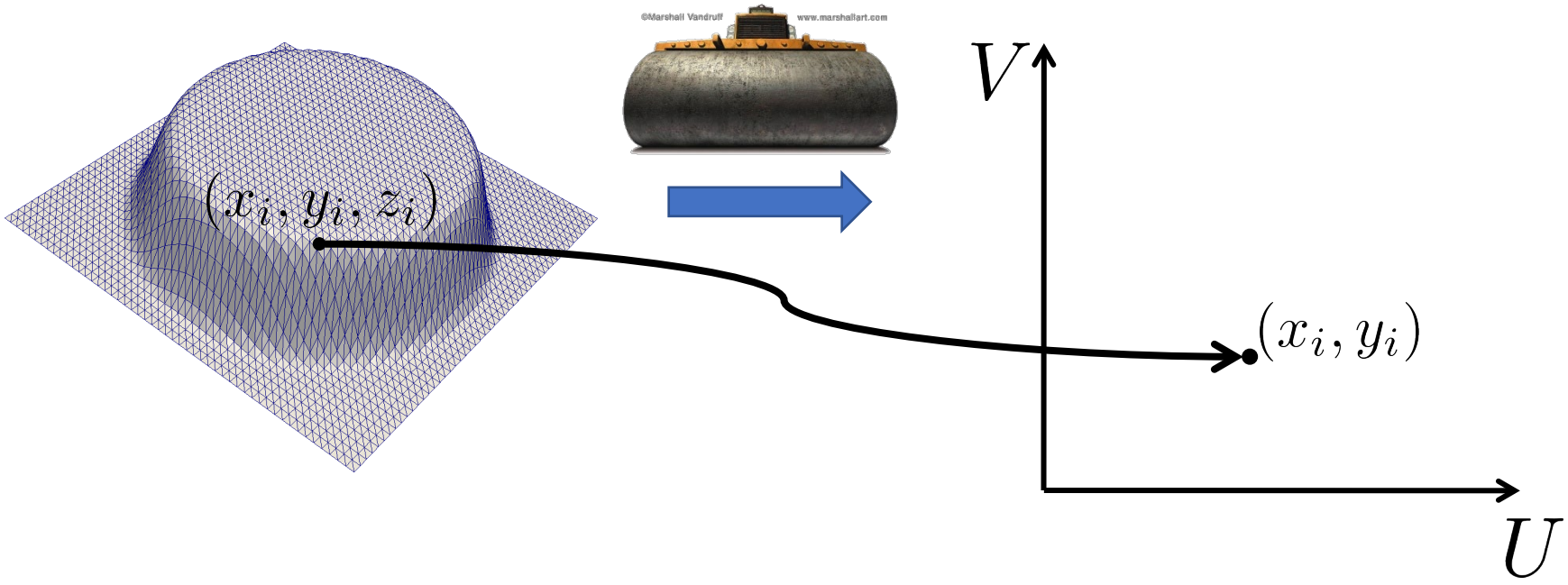
UV-Coordinates



For every mesh vertex determine its (u,v) coordinates

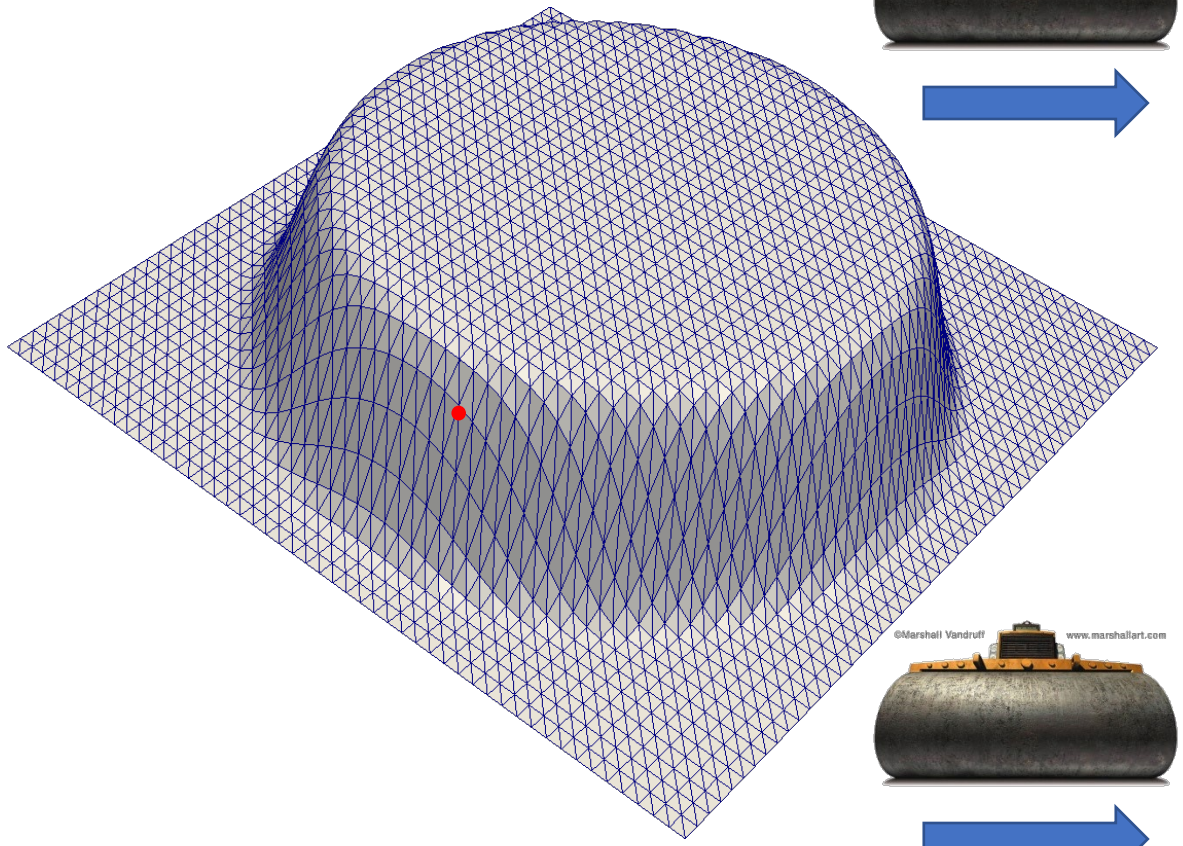
"Texture Coordinates"

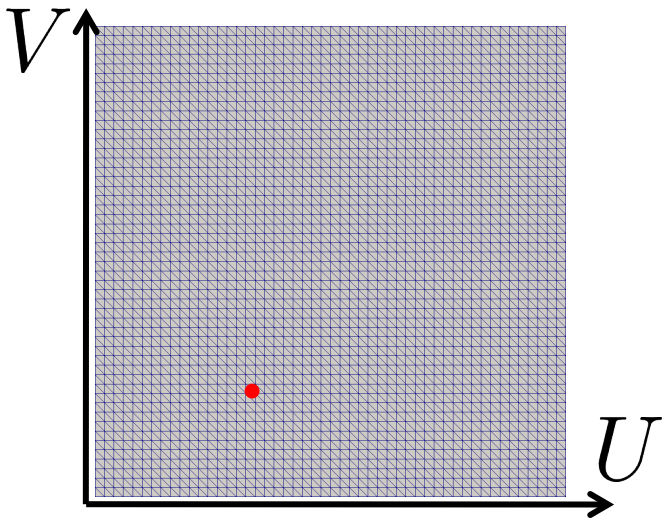
Naïve Approach

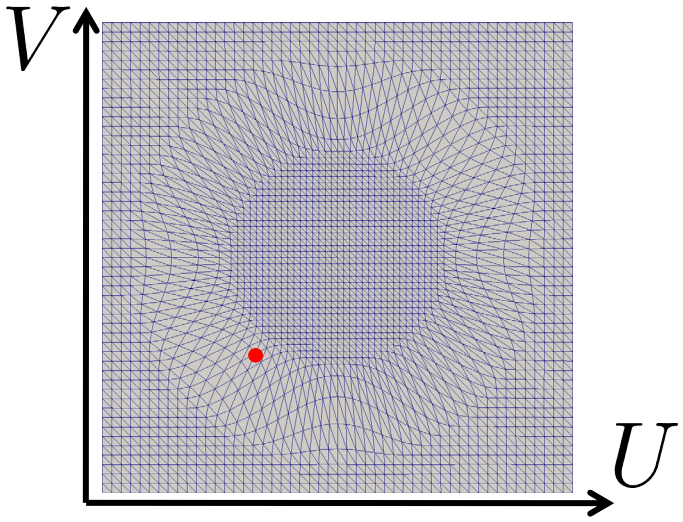


Normalize so that UV-coordinates are in [0,1]

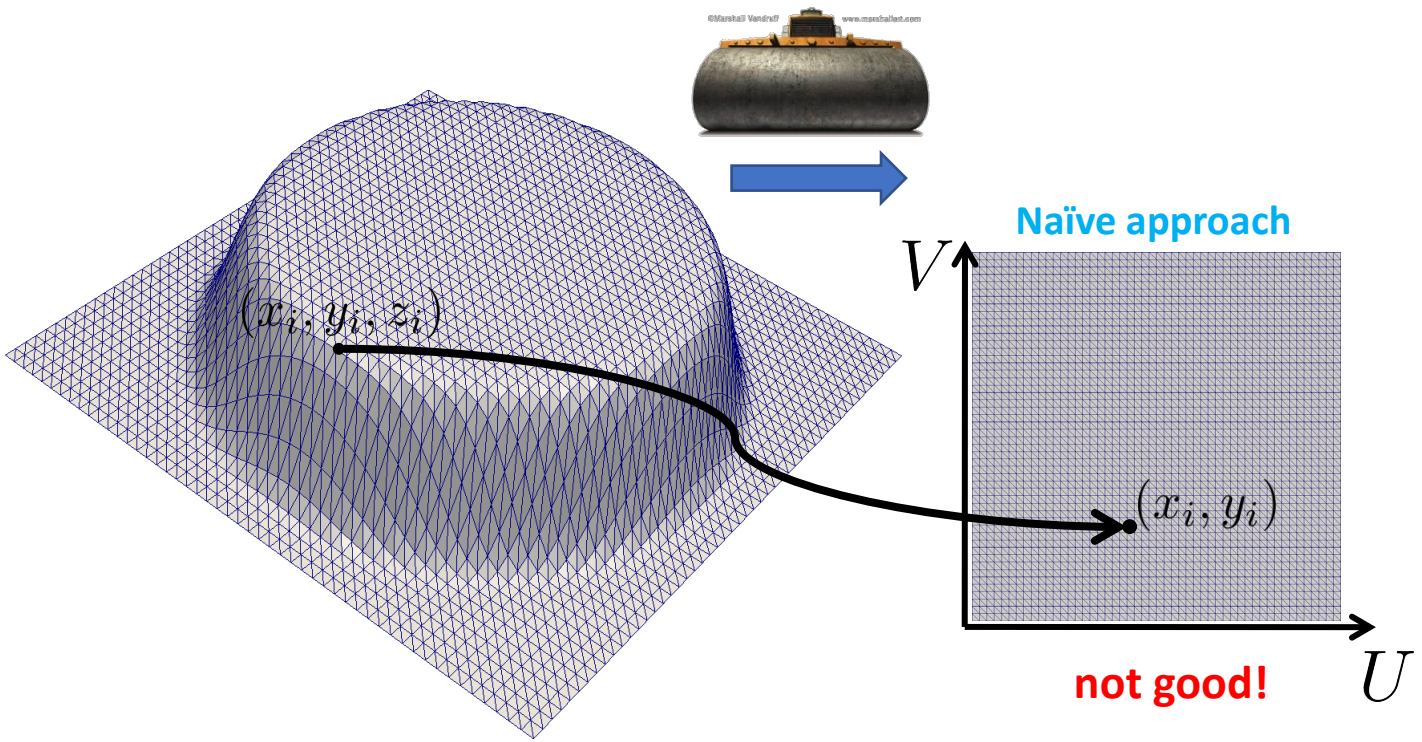
Which is Better?







Main Issue: Distortion

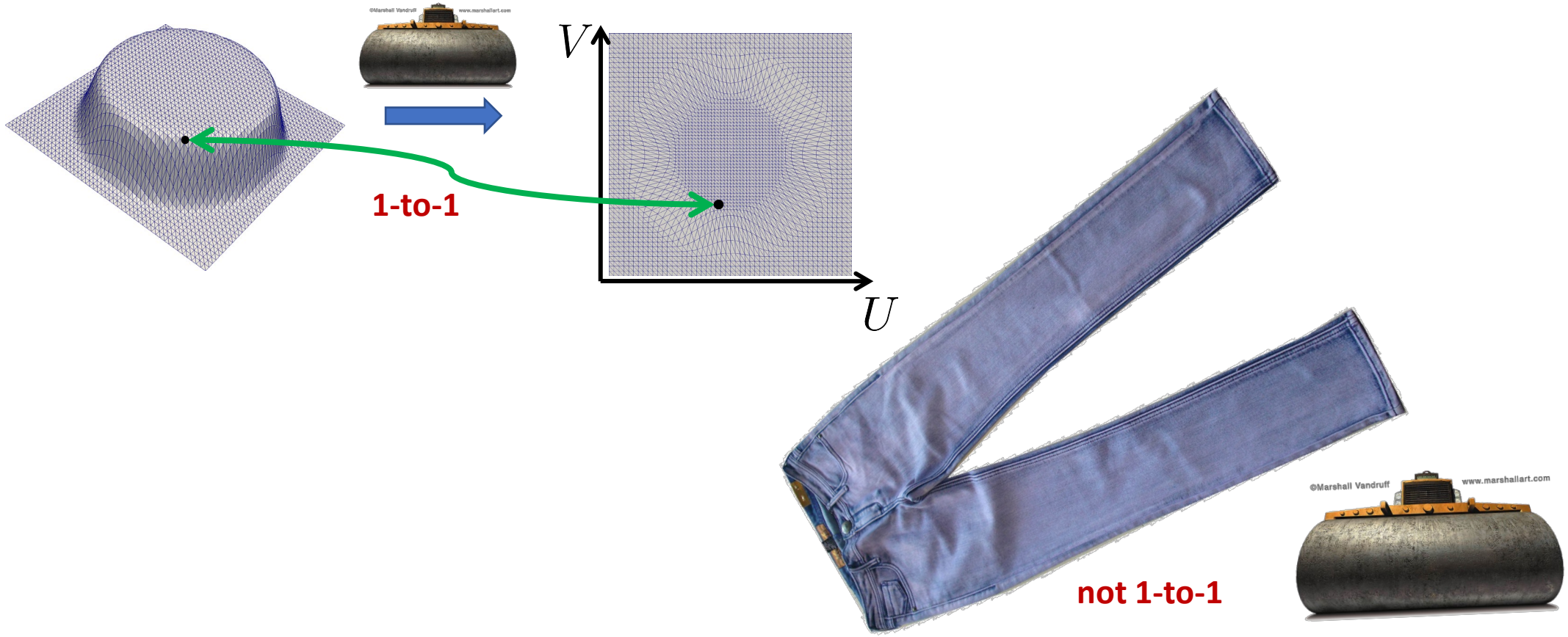


Triangle shapes (angles) and sizes (area) are not preserved!

Preserve shape & size = Triangle congruence (aka isometry)

An Old Problem: Maps

Desirable Characteristics



Bijective: no fold overs

Desirable Characteristics

Conformal: Preserves angles

Desirable Characteristics

Equiareal: Preserves areas

http://en.wikipedia.org/wiki/Bonne_projection

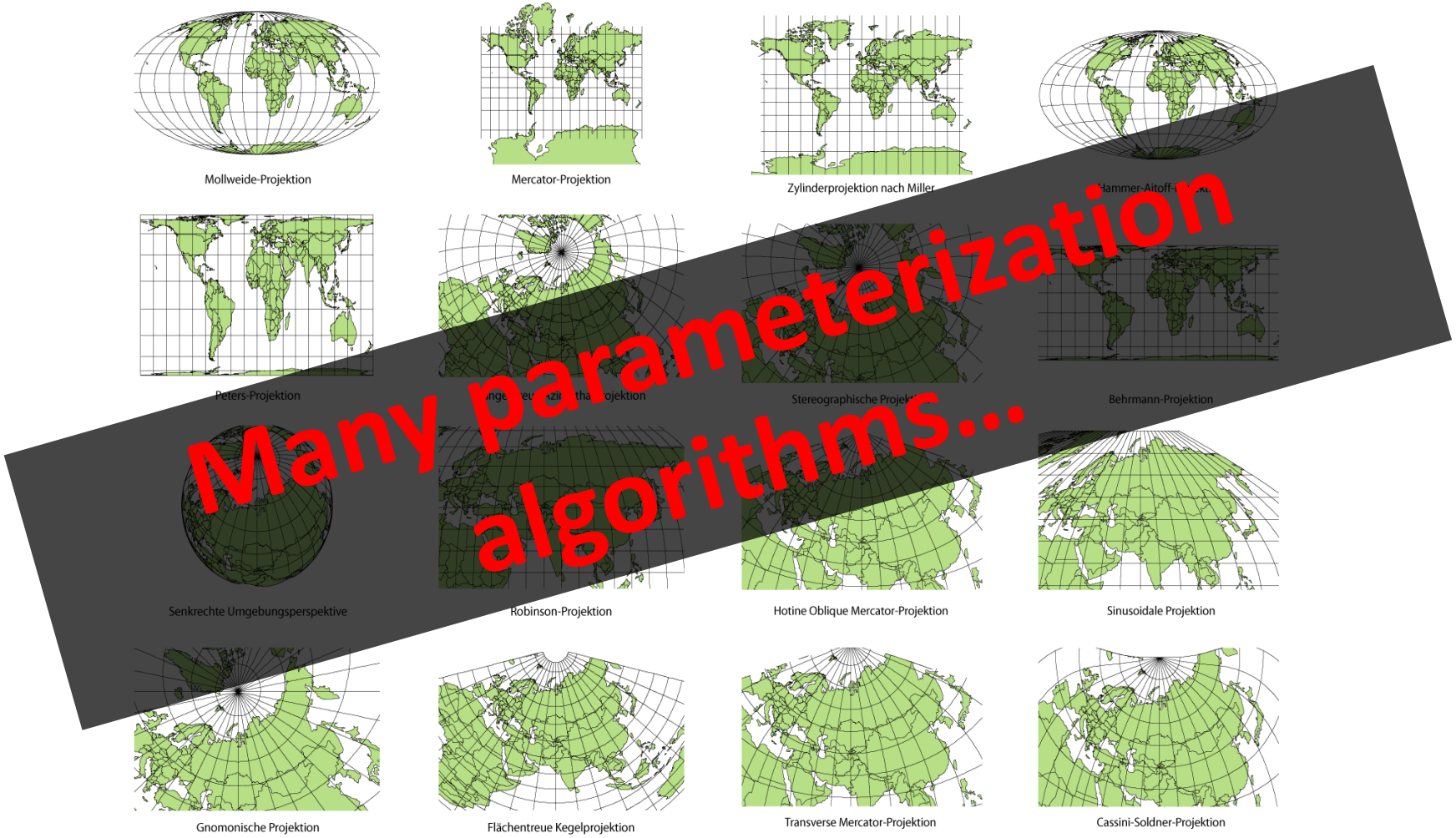
Desirable Characteristics

Isometric: conformal and equiareal

Sad Fact

Very few surfaces can be mapped **isometrically** to the plane.

Reason for so Many Types of Maps

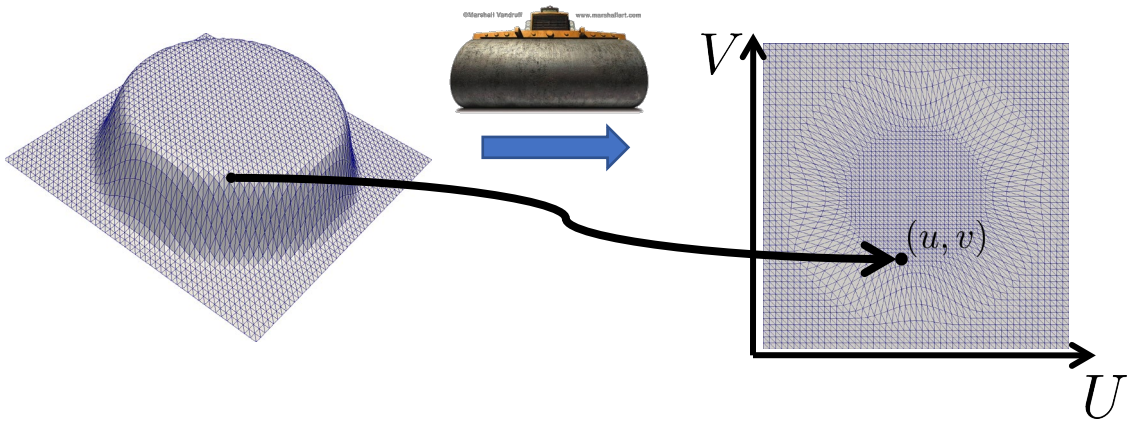


Tutte's Theorem

- If the (u,v) coordinates of the boundary lie on a convex polygon, and if coordinates of the internal vertices are convex combination of their neighbors, then these (u,v) coordinates give a valid (bijective) parameterization.

- Convex combination = center of mass
- Can have different masses at different vertices
- Masses should be positive

Simple Realization



Goal: Assign (*u*,*v*) coordinate to each mesh vertex.

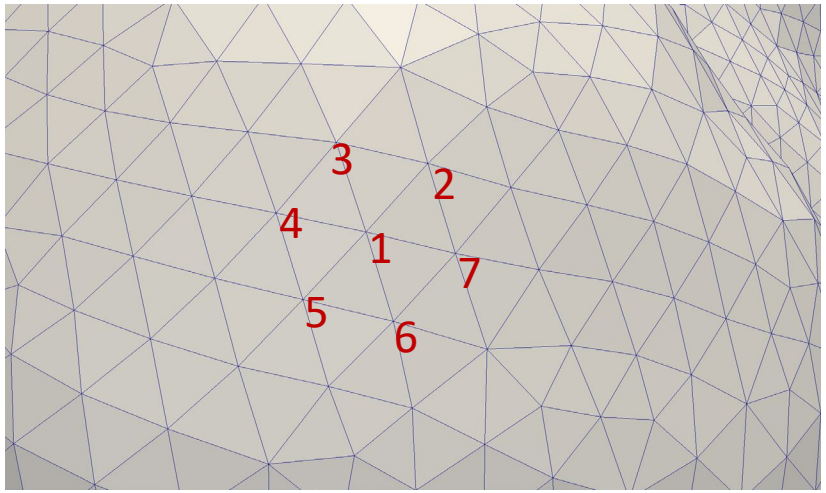
1. Fix (*u*,*v*) coordinates of boundary.
2. Want interior vertices to be at the center of mass of neighbors:

$$u_i = \frac{1}{|N(i)|} \sum_{j \in N(i)} u_j \qquad v_i = \frac{1}{|N(i)|} \sum_{j \in N(i)} v_j$$

Algorithm

1. Fix (*u*,*v*) coordinates of boundary.
2. Initialize (u,v) of interior points (e.g. using naïve).
3. While not converged: for each interior vertex, set:

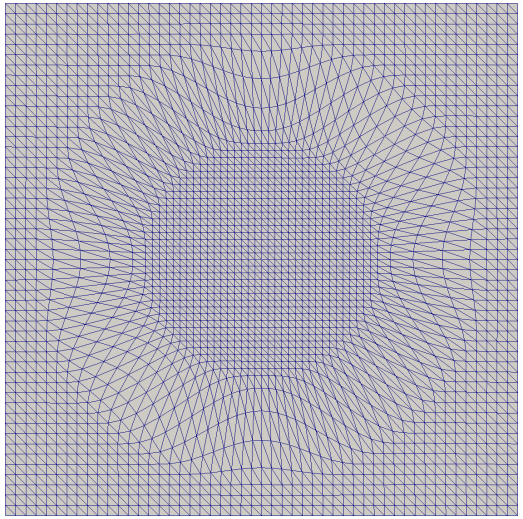
$$u_i \leftarrow \frac{1}{|N(i)|} \sum_{j \in N(i)} u_j \qquad v_i \leftarrow \frac{1}{|N(i)|} \sum_{j \in N(i)} v_j$$



$$u_1 \leftarrow \frac{u_2 + u_3 + u_4 + u_5 + u_6 + u_7}{6}$$

$$v_1 \leftarrow \frac{v_2 + v_3 + v_4 + v_5 + v_6 + v_7}{6}$$

What do you think?

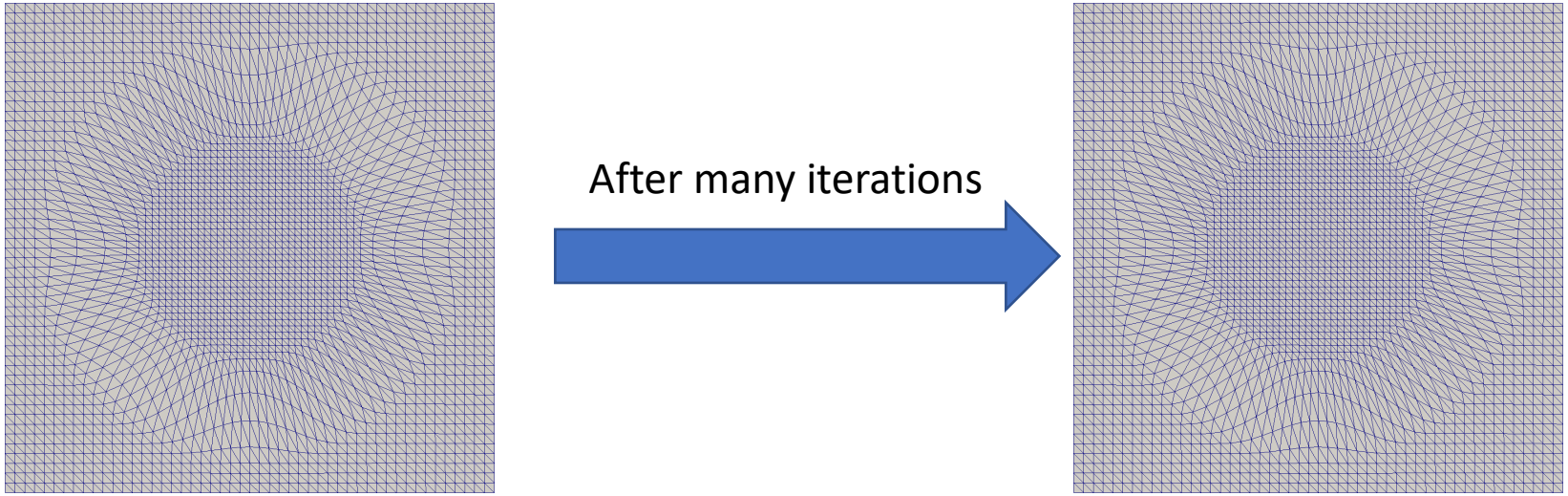


Some random planar mesh

After many iterations

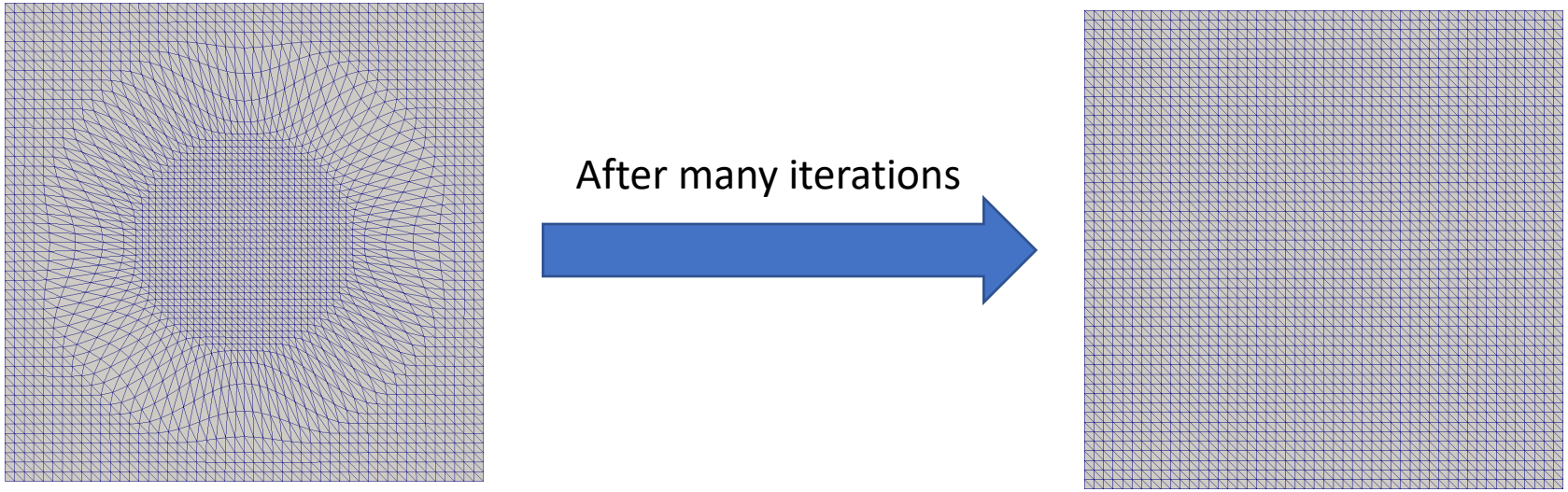
?

Expectation



It is already planar: best parameterization = itself

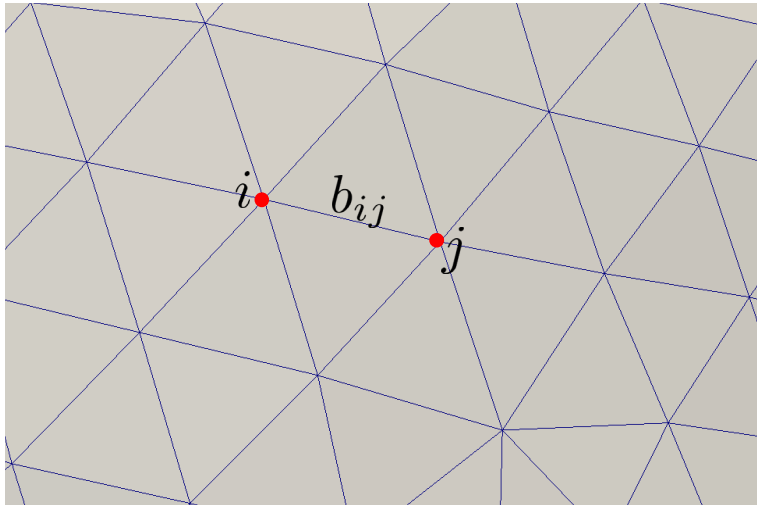
Reality… Why? How to Avoid?



Converges to a somewhat uniform grid!

Triangle shapes and sizes are not preserved!

Algorithm with Weights



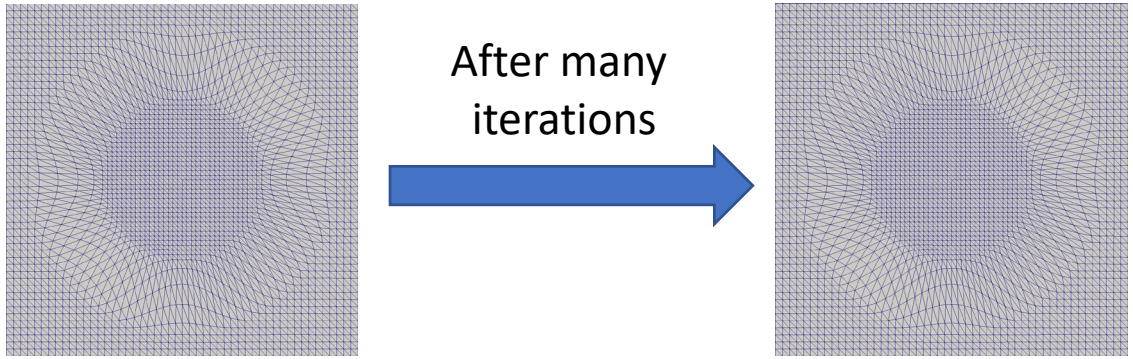
Introduce weights to capture geometric information:

1. Fix (*u*,*v*) coordinates of boundary.
2. Initialize (u,v) of interior points (e.g. using naïve).
3. While not converged: for each interior vertex, set:

$$u_i \leftarrow \sum_{j \in N(i)} b_{ij} u_j \qquad v_i \leftarrow \sum_{j \in N(i)} b_{ij} v_j$$

Weight Properties

GOAL:



For planar mesh with vertex coordinates (x_i, y_i):

$$\sum_{j \in N(i)} b_{ij} x_j = x_i$$

$$\sum_{j \in N(i)} b_{ij} y_j = y_i$$

With naïve initialization,
iterations converge immediately!

$$u_i \leftarrow \sum_{j \in N(i)} b_{ij} u_j \qquad v_i \leftarrow \sum_{j \in N(i)} b_{ij} v_j$$

Weight Properties

$$\sum_{j \in N(i)} b_{ij} = 1$$

If weights w_{ij}, don't satisfy, then normalize:

$$b_{ij} = \frac{w_{ij}}{\sum_{j \in N(i)} w_{ij}}$$

For planar mesh with vertex coordinates (x_i, y_i):

$$\sum_{j \in N(i)} b_{ij} x_j = x_i$$

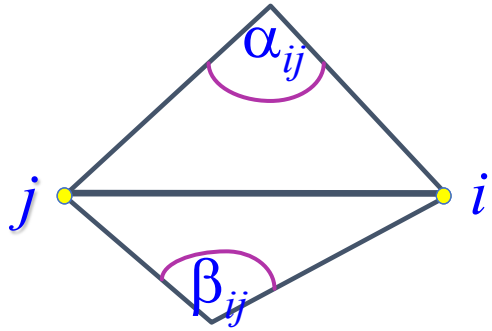
$$\sum_{j \in N(i)} b_{ij} y_j = y_i$$

For Tutte's theorem to hold, $b_{ij} > 0$

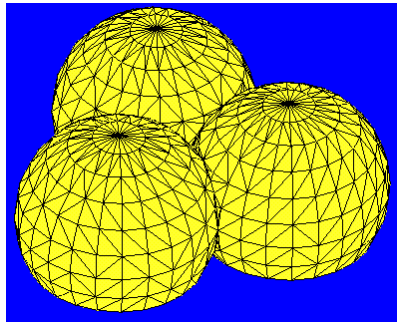
Barycentric Coordinates

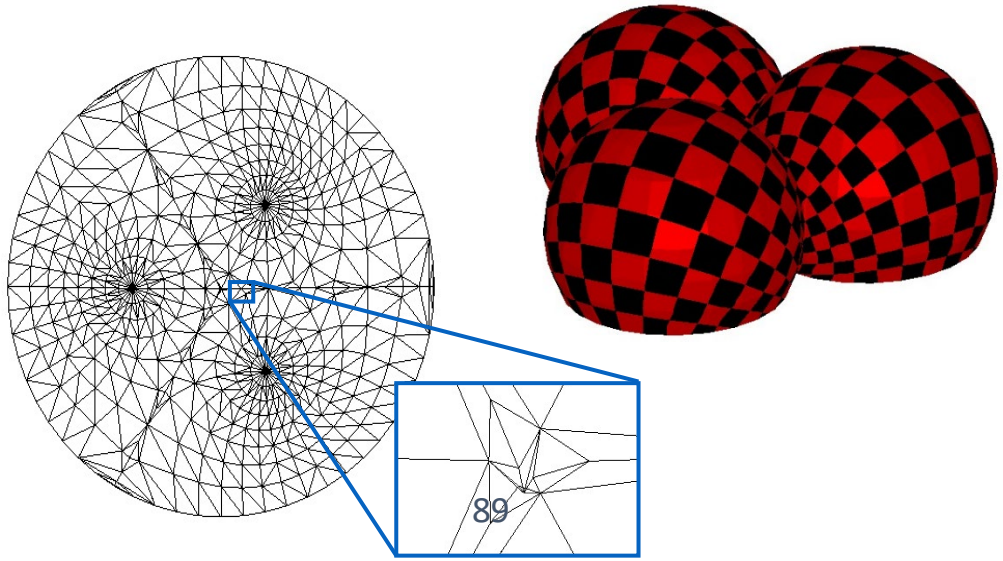
Harmonic Weights

$$w_{ij} = \frac{\cot(\alpha_{ij}) + \cot(\beta_{ij})}{2}$$



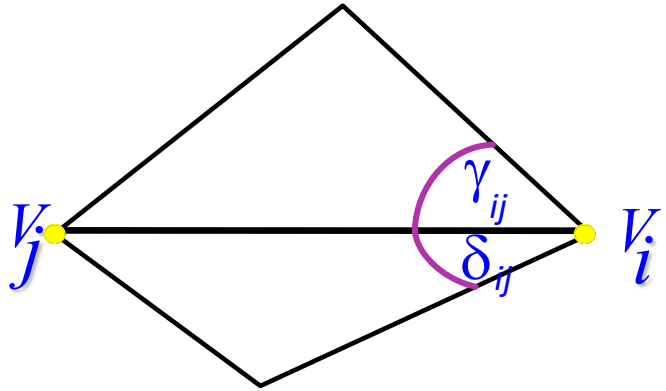
- Weights can be negative – not always bijective
- Weights depend only on angles - close to conformal
- 2D reproducible



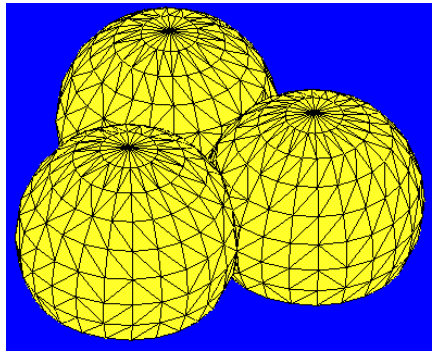


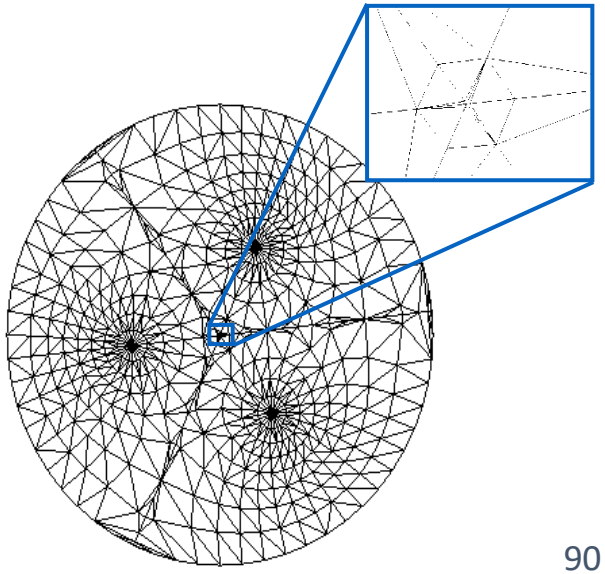
Mean-Value Weights

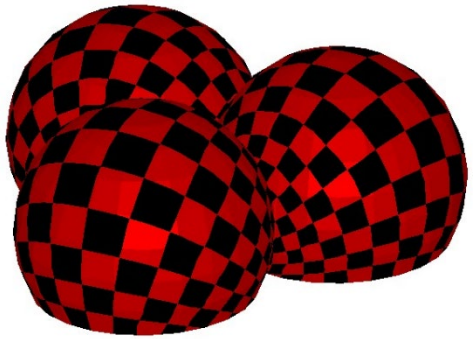
$$w_{ij} = \frac{\tan(\gamma_{ij}/2) + \tan(\delta_{ij}/2)}{2\,\|V_i - V_j\|}$$



- Result visually similar to harmonic
- No negative weights – always bijective
- 2D reproducible

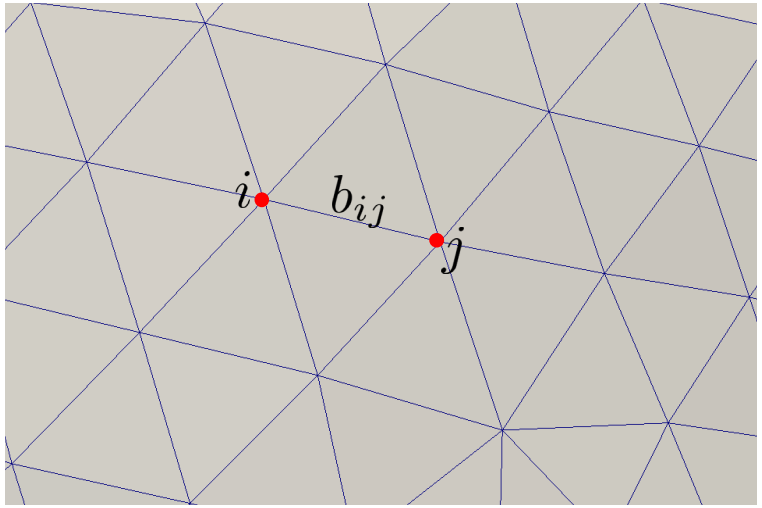






90

Recap: Algorithm with Weights

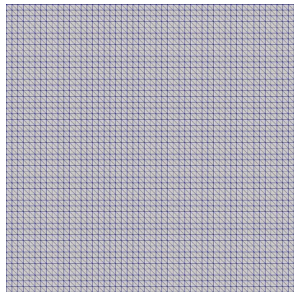


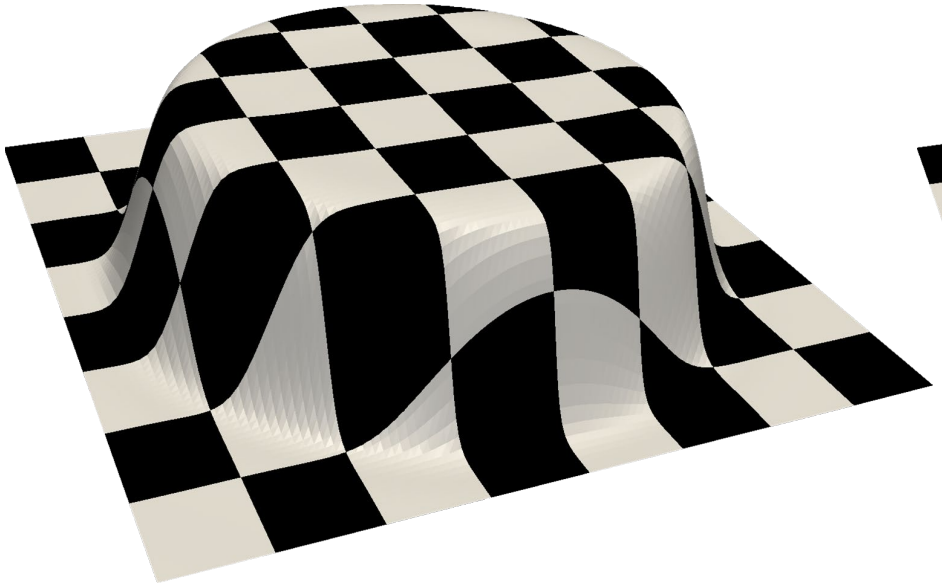
0. Pick some kind of barycentric coordinates as weights to capture geometric information.

1. Fix (*u*,*v*) coordinates of boundary.
2. Initialize (u,v) of interior points (e.g. using naïve).
3. While not converged: for each interior vertex, set:

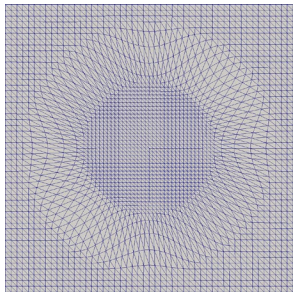
$$u_i \leftarrow \sum_{j \in N(i)} b_{ij} u_j \qquad v_i \leftarrow \sum_{j \in N(i)} b_{ij} v_j$$

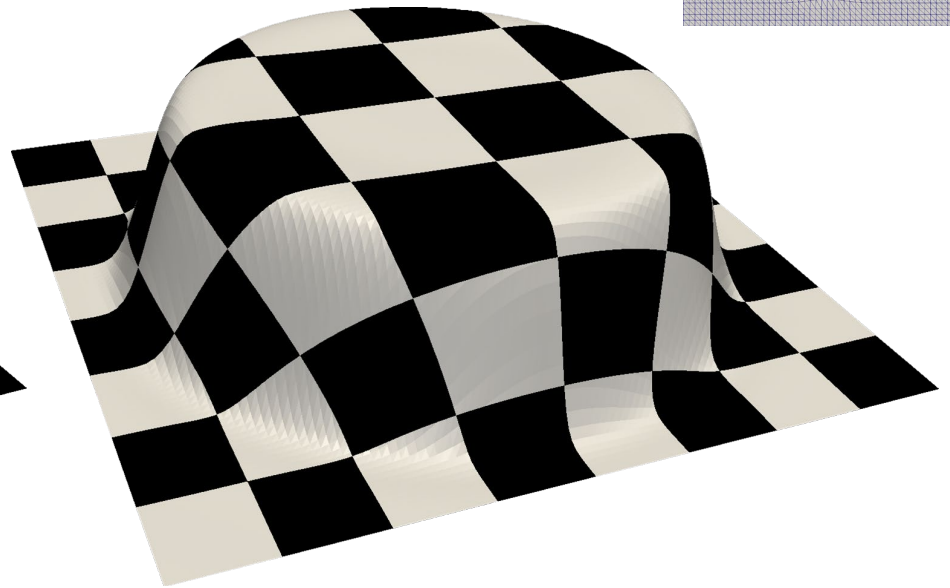
Results





Naive

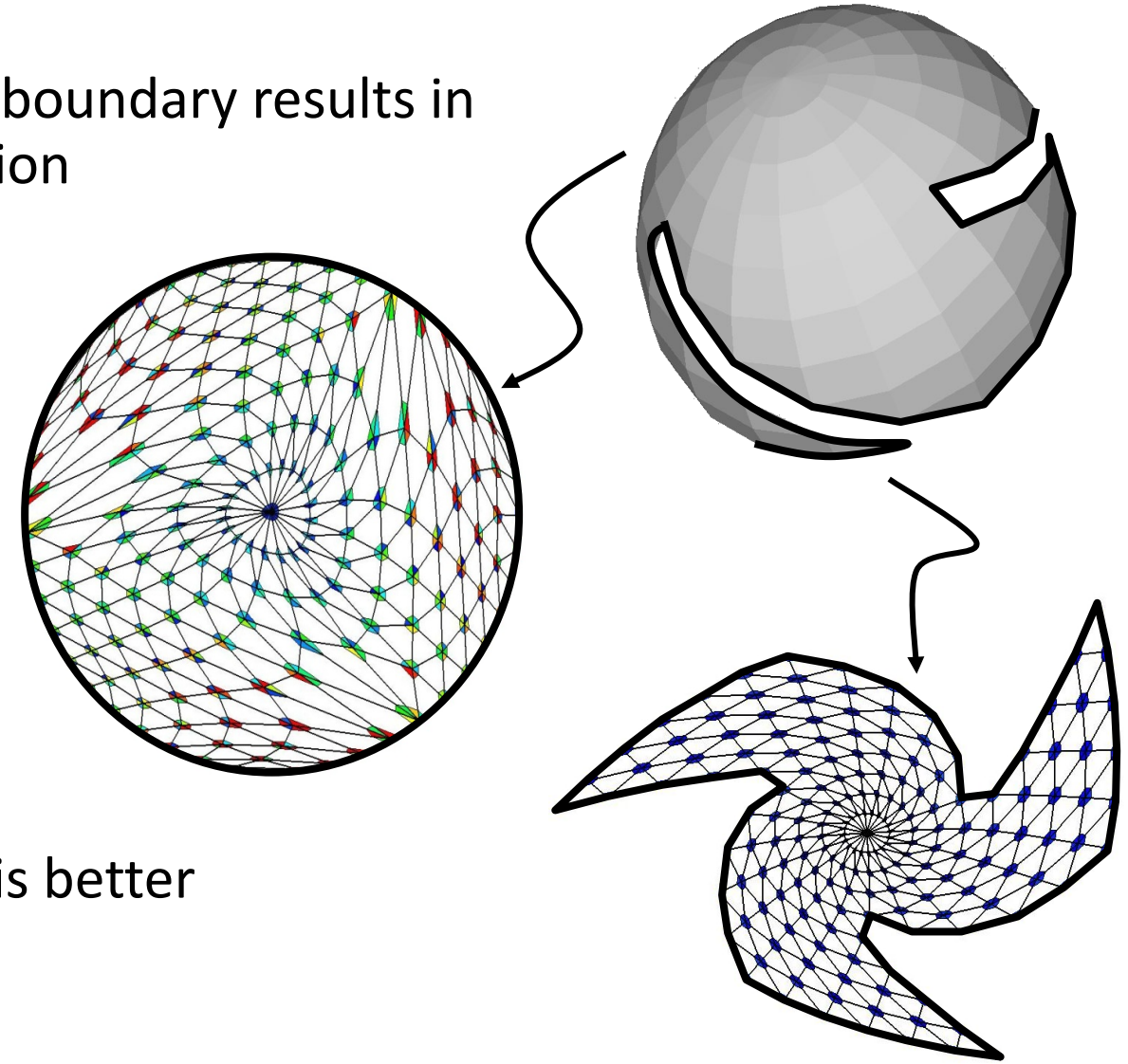




Harmonic

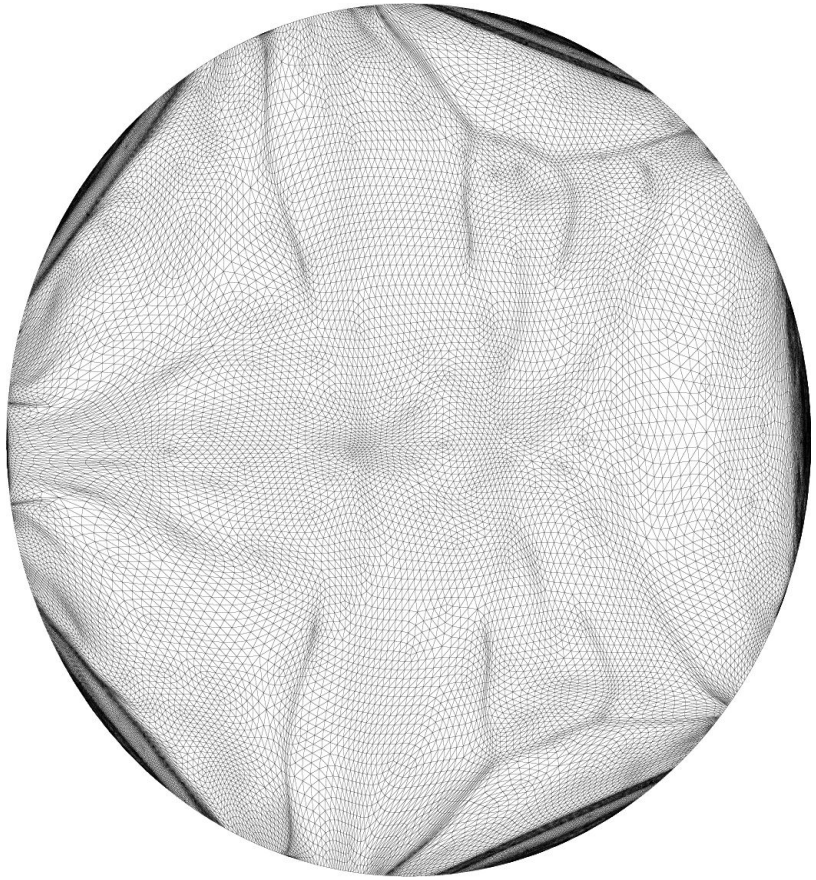
Non-Convex Boundary

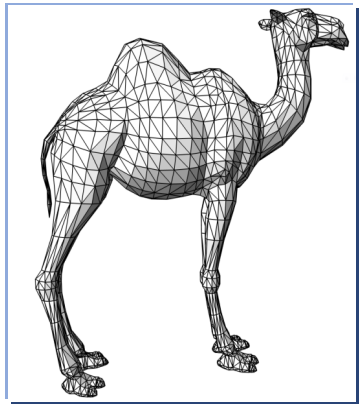
- Requiring convex boundary results in significant distortion



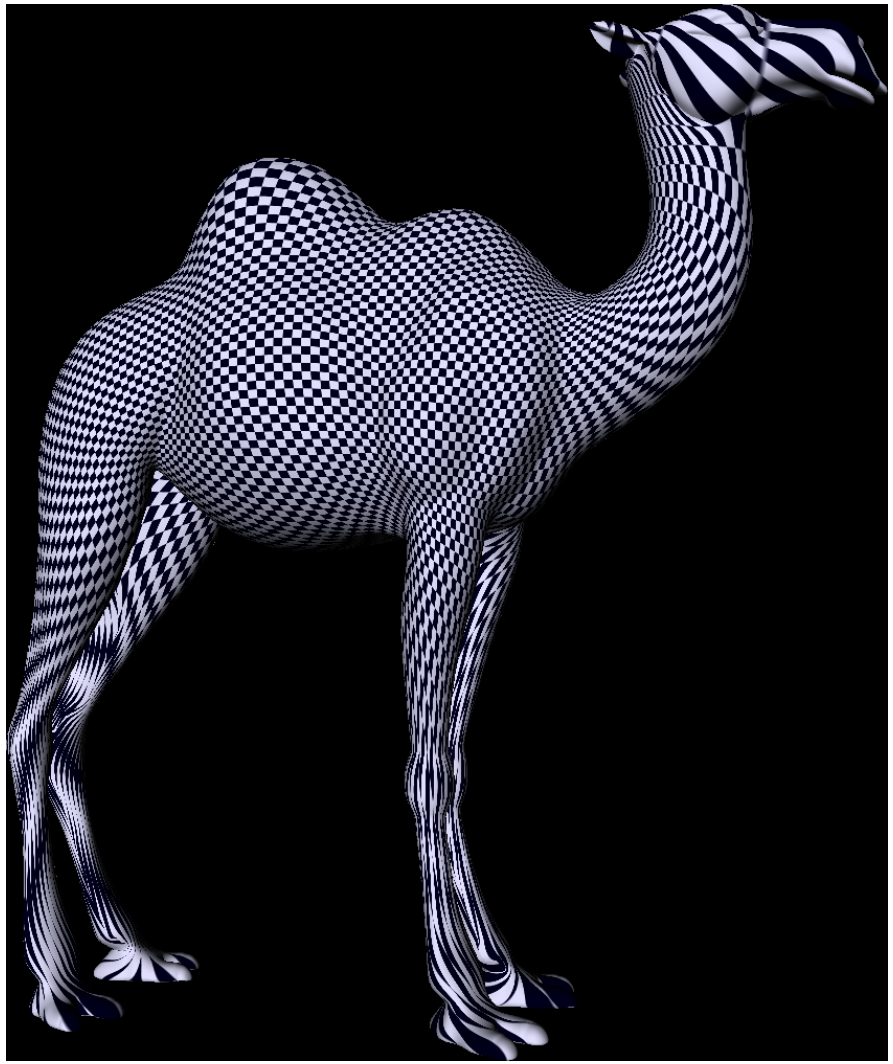
- “Free” boundary is better

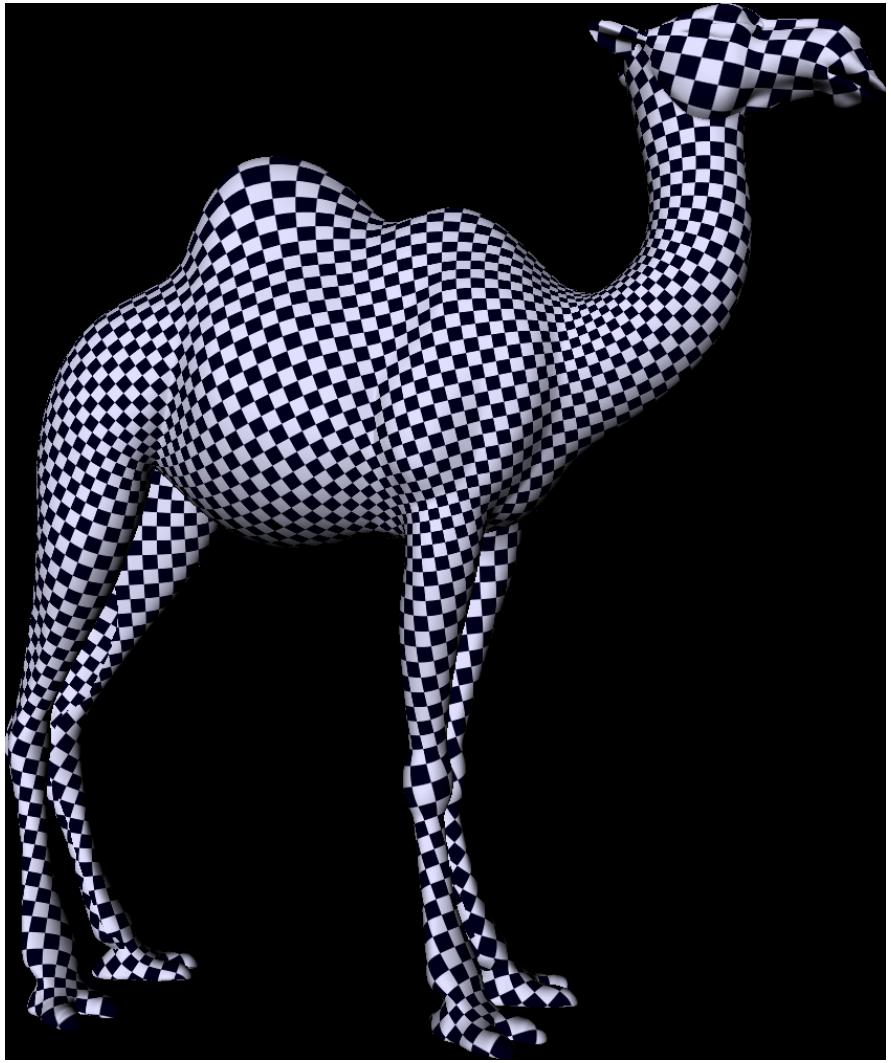
Fixed vs Free Boundary





Fixed vs Free Boundary





Old Algorithm

1. Fix (*u*,*v*) coordinates of boundary.
2. Initialize (u,v) of interior points (e.g. using naïve).
3. While not converged: for each interior vertex, set:

$$u_i \leftarrow \sum_{j \in N(i)} b_{ij} u_j \qquad v_i \leftarrow \sum_{j \in N(i)} b_{ij} v_j$$

Old Algorithm

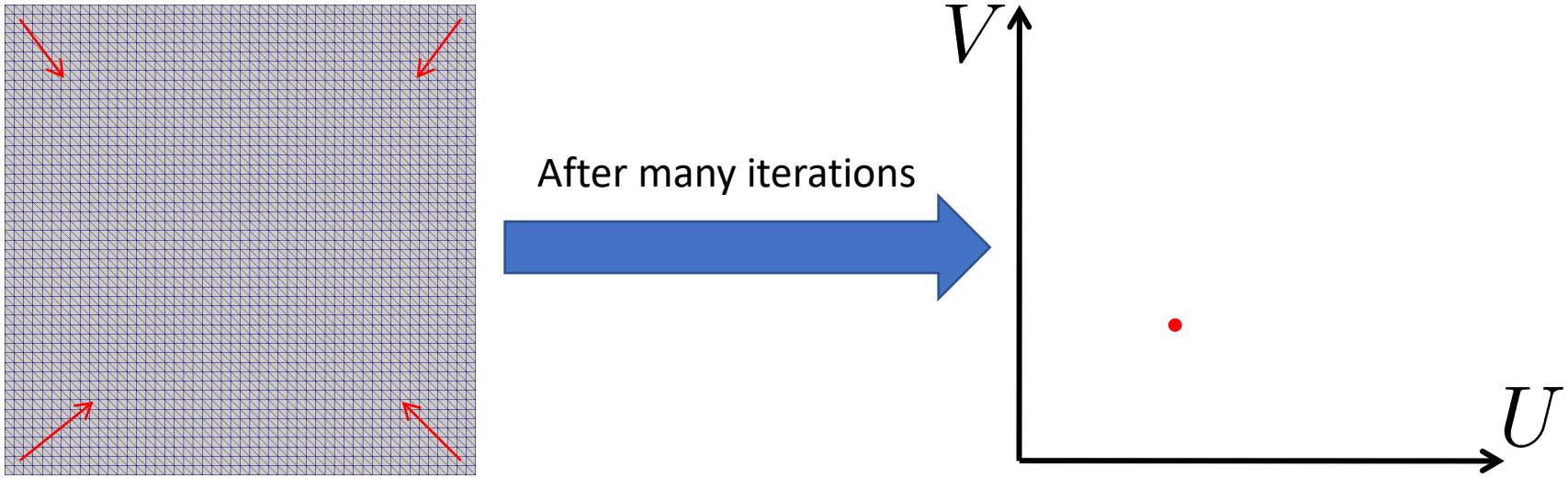
1. Initialize (u,v) of all points (e.g. using naïve).
2. While not converged: for each vertex, set:

$$u_i \leftarrow \sum_{j \in N(i)} b_{ij} u_j \qquad v_i \leftarrow \sum_{j \in N(i)} b_{ij} v_j$$

Why this would be problematic? How to fix this?

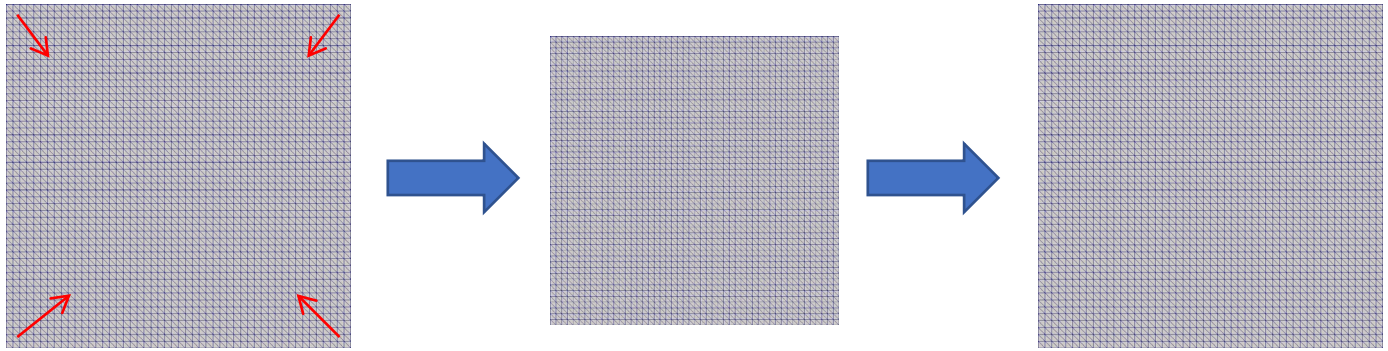
Problem

- Boundary vertices are pulled towards interior
- Shrinkage happens
- Collapse to a single point!



How to Fix this?

- Un-shrink at every iteration:
  - Move the center of mass at origin of UV-plane
  - Rescale in U direction to make std. deviation = 1
  - Rescale in V direction to make std. deviation = 1
  - Make sure covariance between U and V = 0
    - Subtract an appropriate multiple of U from V.



Fixed Algorithm

Initialize (u,v) for all vertices (e.g. using naïve)
While not converged:
    For several times:
        For each vertex, set:

$$u_i \leftarrow \sum_{j \in N(i)} b_{ij} u_j \quad v_i \leftarrow \sum_{j \in N(i)} b_{ij} v_j$$

        End For
    End For
    Un-shrink
End While

Fixed Algorithm = Eigenmap

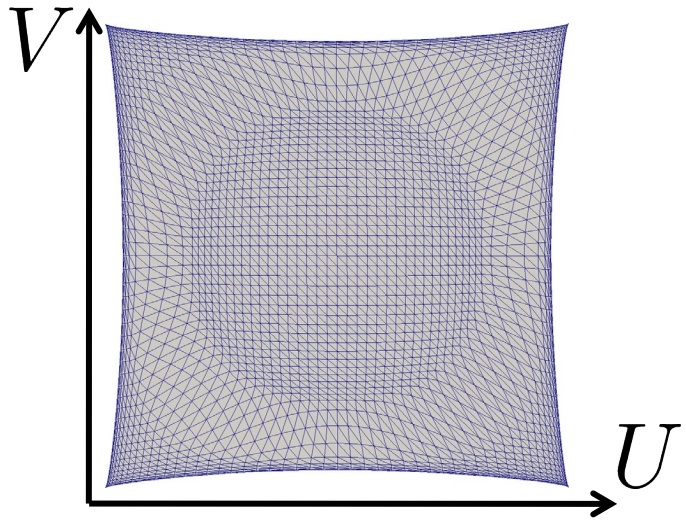
- Equivalent to inverse power iteration for solving an eigenvalue/eigenvector problem of type

$$Ax = \lambda x$$

- Pick the smallest two non-constant eigenvectors. Call these
- Set (u,v) coordinates as: $x^{(1)}, x^{(2)}$

$$u_i = x_i^{(1)} \qquad\qquad v_i = x_i^{(2)}$$

Eigenmap result



Connection Between Methods

- Fixed boundary, solve Ax=b
- Free boundary, solve Ax =0…
  - Problem: then solution x = 0!
  - Solution: solve instead eigenvalue problem

$$Ax = \lambda x$$

and pick eigenvectors corresponding to the smallest non-zero eigenvalues.

Parametrization Summary

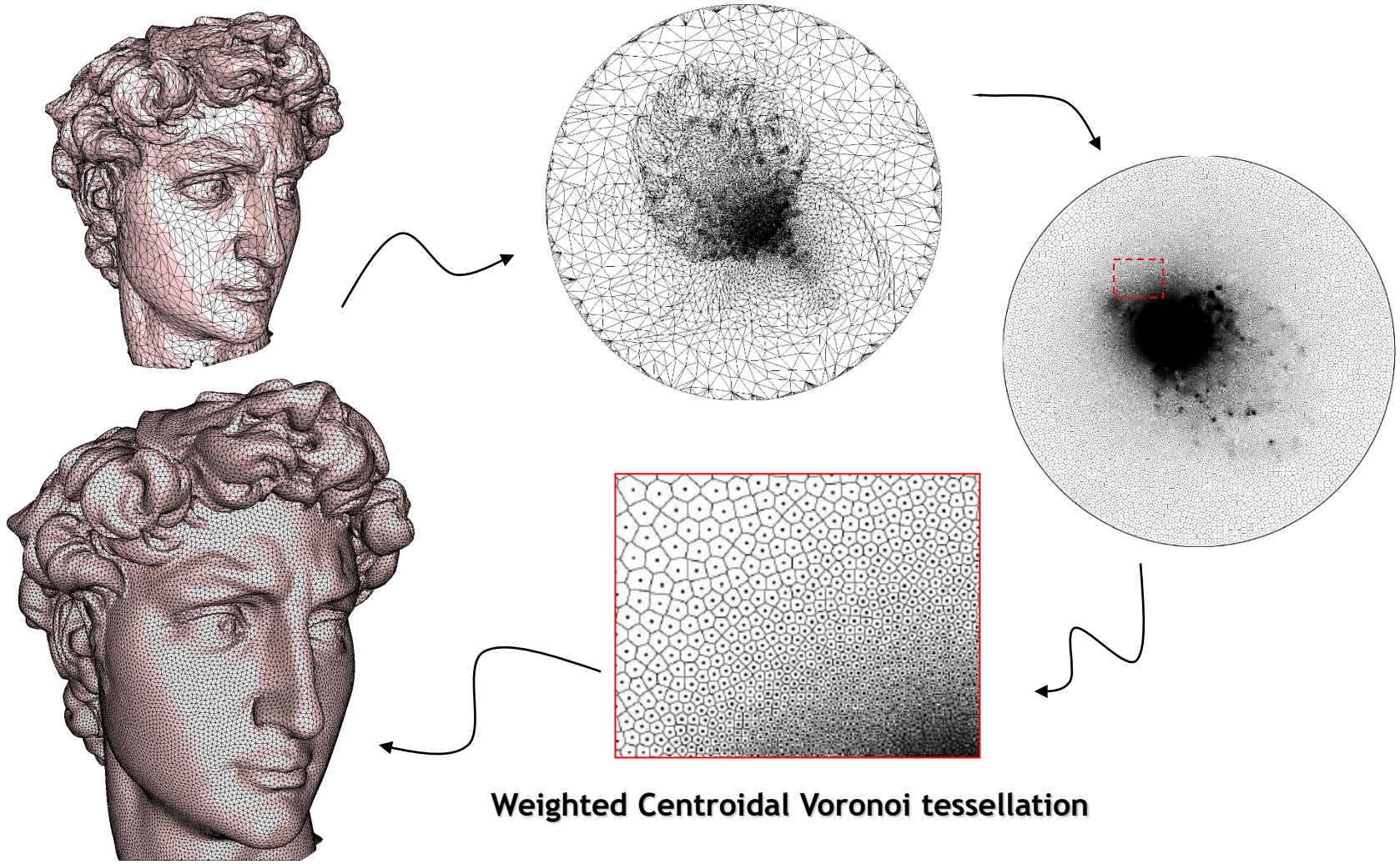
- Mesh parameterization = flattening
- Inspecting and classifying distortion:
  - Conformal/Equiareal/Isometric
- Methods
  - Fixed bndry: Harmonic parameterization
  - Free bndry: Eigenmaps
- These are easy to implement!

Outline

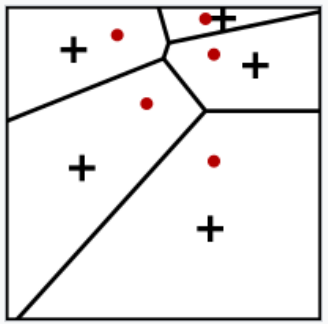
- Smoothing Motivation
- Spectral Analysis
- Diffusion Flow
- Remeshing Motivation
- Parametrization
- **Global parameterization remeshing**
- Direct surface remeshing

Isotropic Remeshing via Parametrization

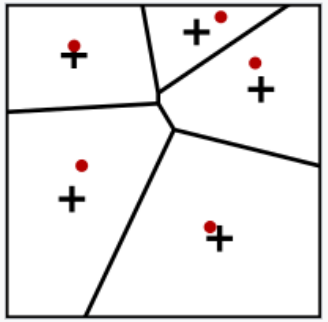
Remesh in the plane!



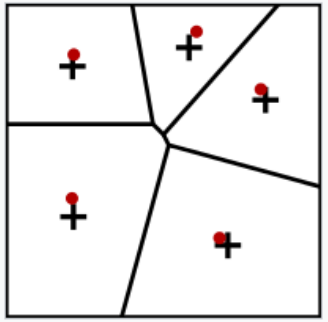
Weighted Centroidal Voronoi tessellation



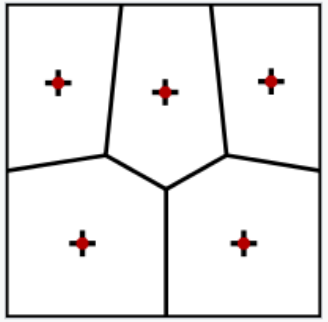
Iteration 1



Iteration 2



Iteration 3



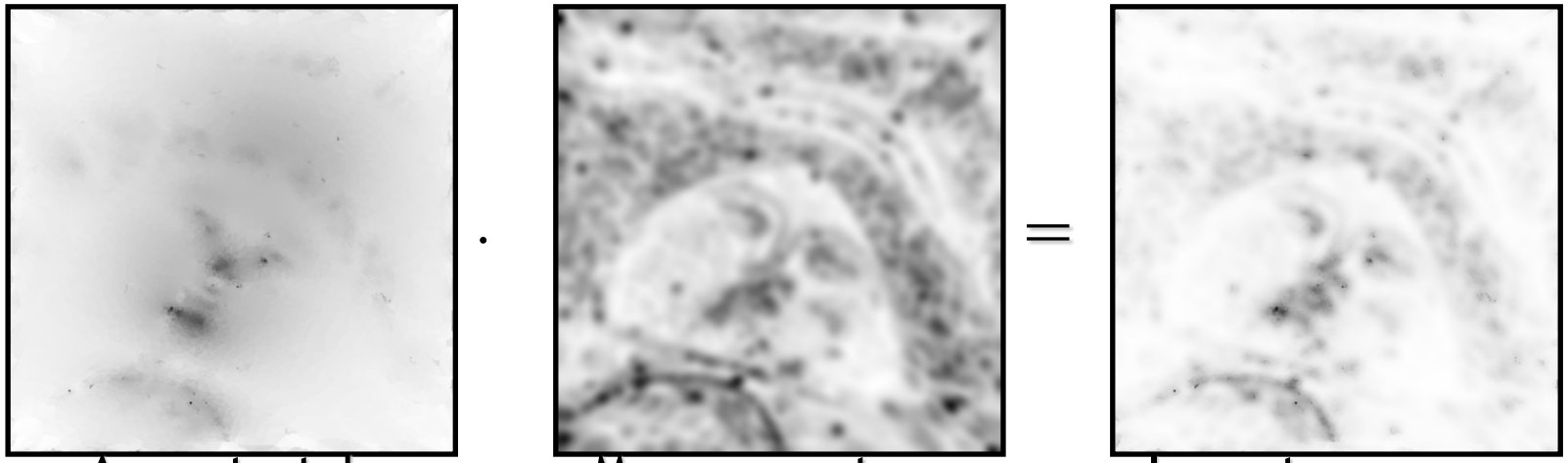
Iteration 15

Initial 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

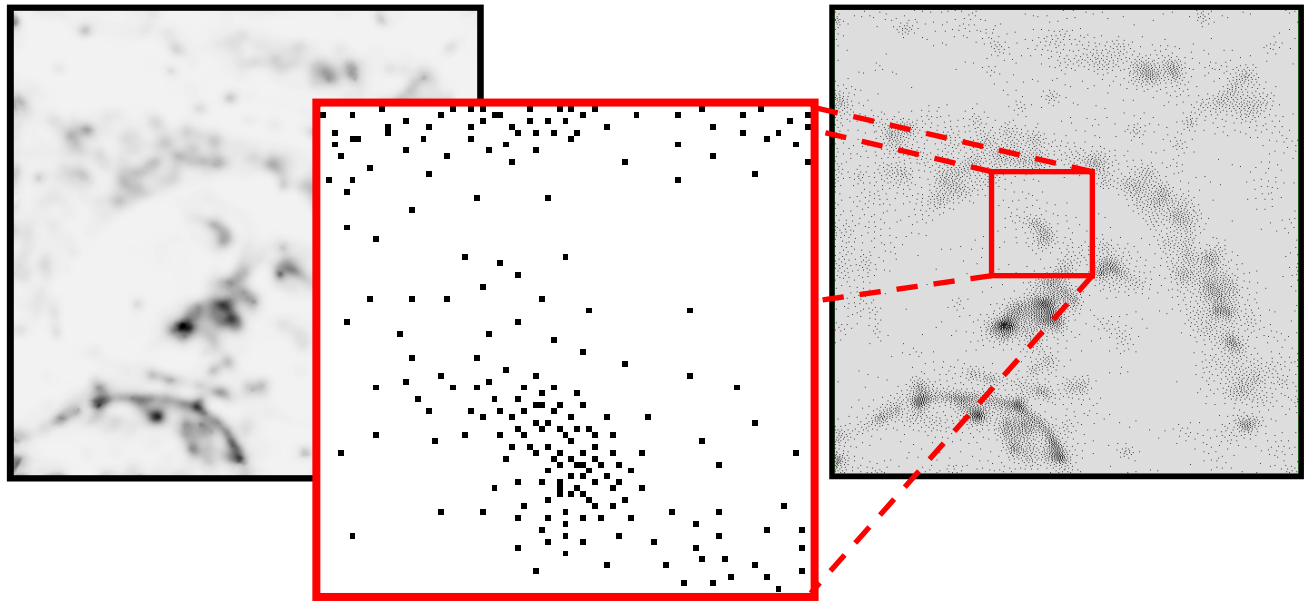
Initial Sampling

- Compose importance map



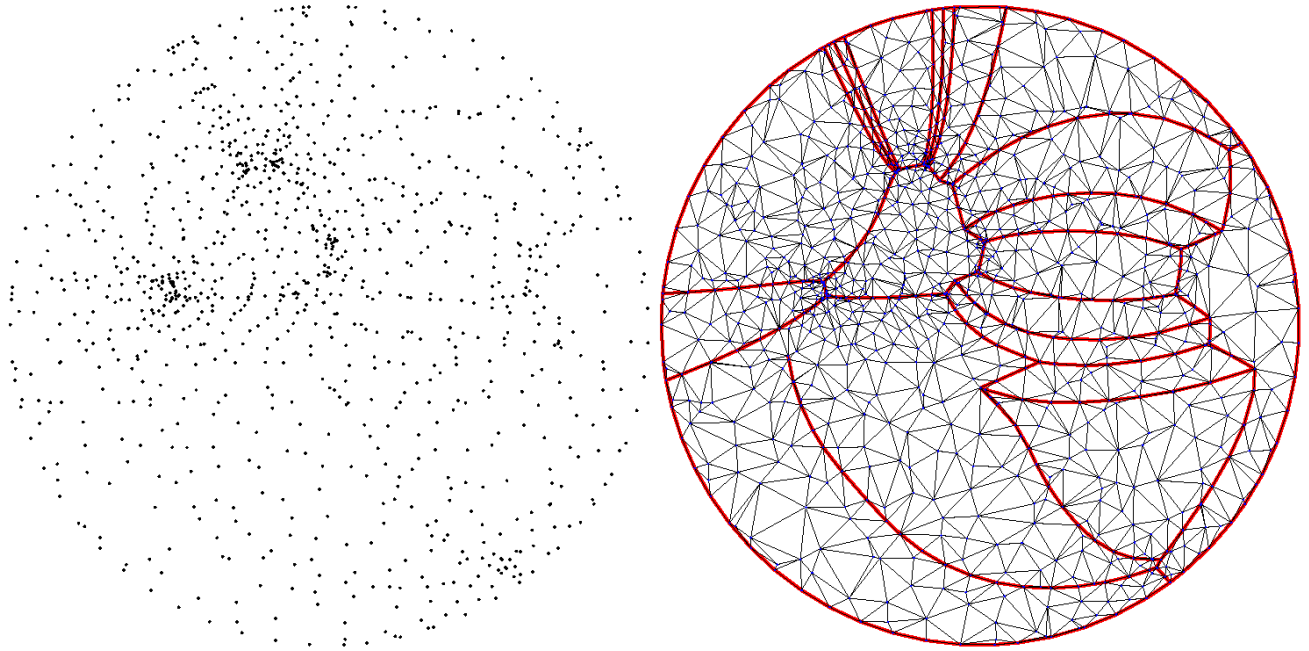
Initial Sampling [Alliez02]

- 2D error diffusion on importance map
  - Half-toning, dithering
  - Can also be done on 3D meshes [Alliez03]



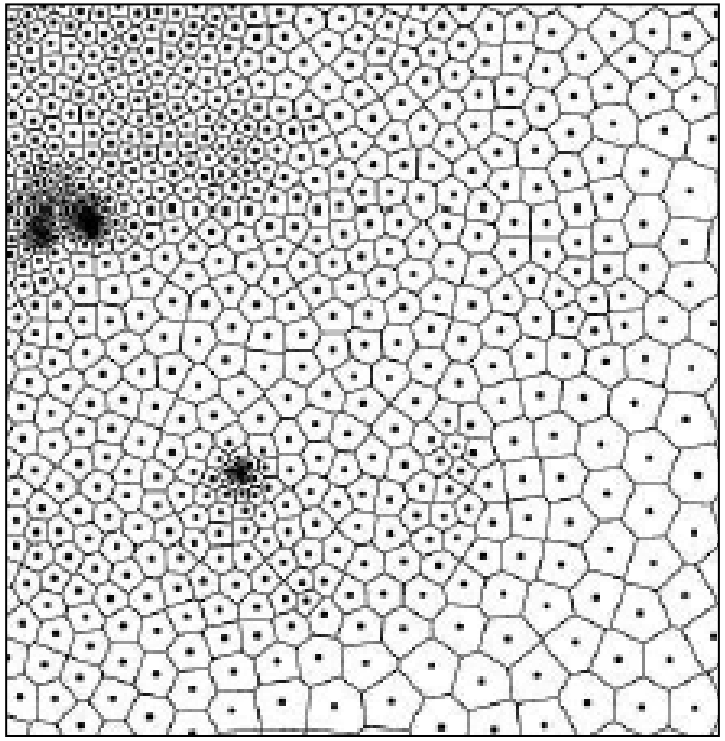
Initial Meshing

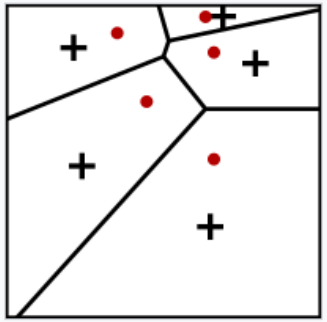
- 2D constrained Delaunay triangulation
- CGAL library provides robust implementation



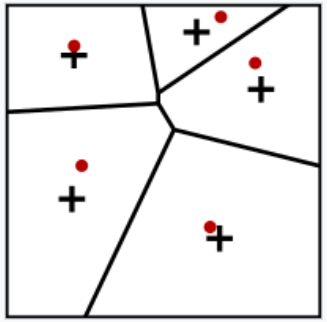
Optimize Sampling / Meshing

- Density-weighted centroidal Voronoi diagram
  - Equal mass enclosing
  - Tiles as compact as possible
  - Highly isotropic sampling
  - Lloyd clustering

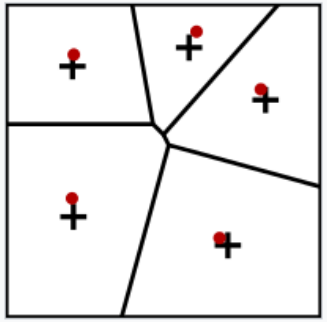




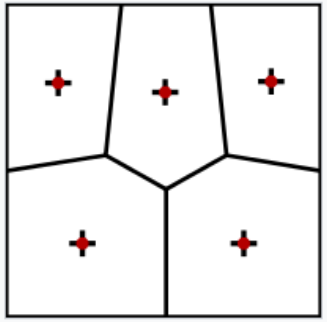
Iteration 1



Iteration 2

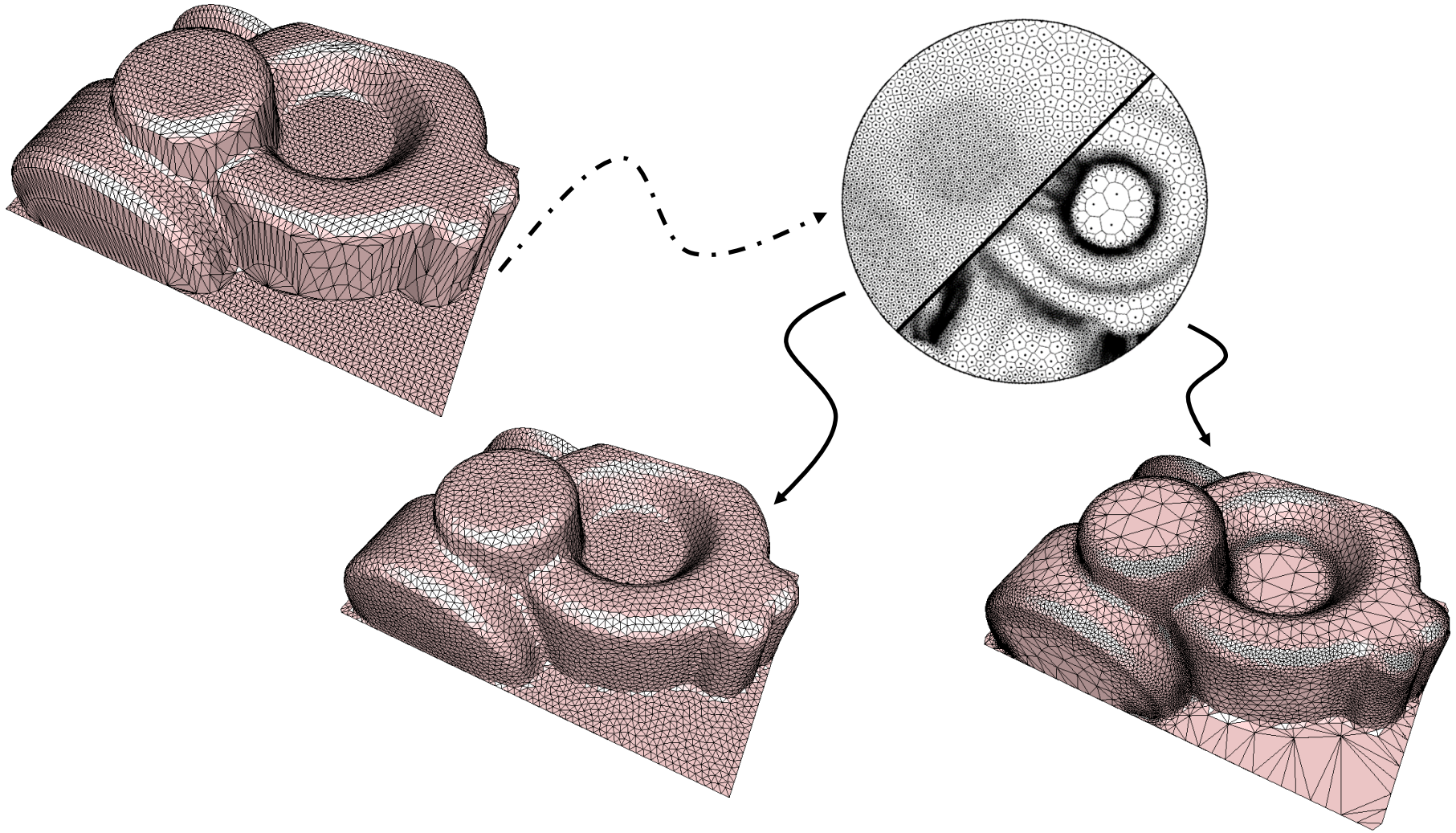


Iteration 3

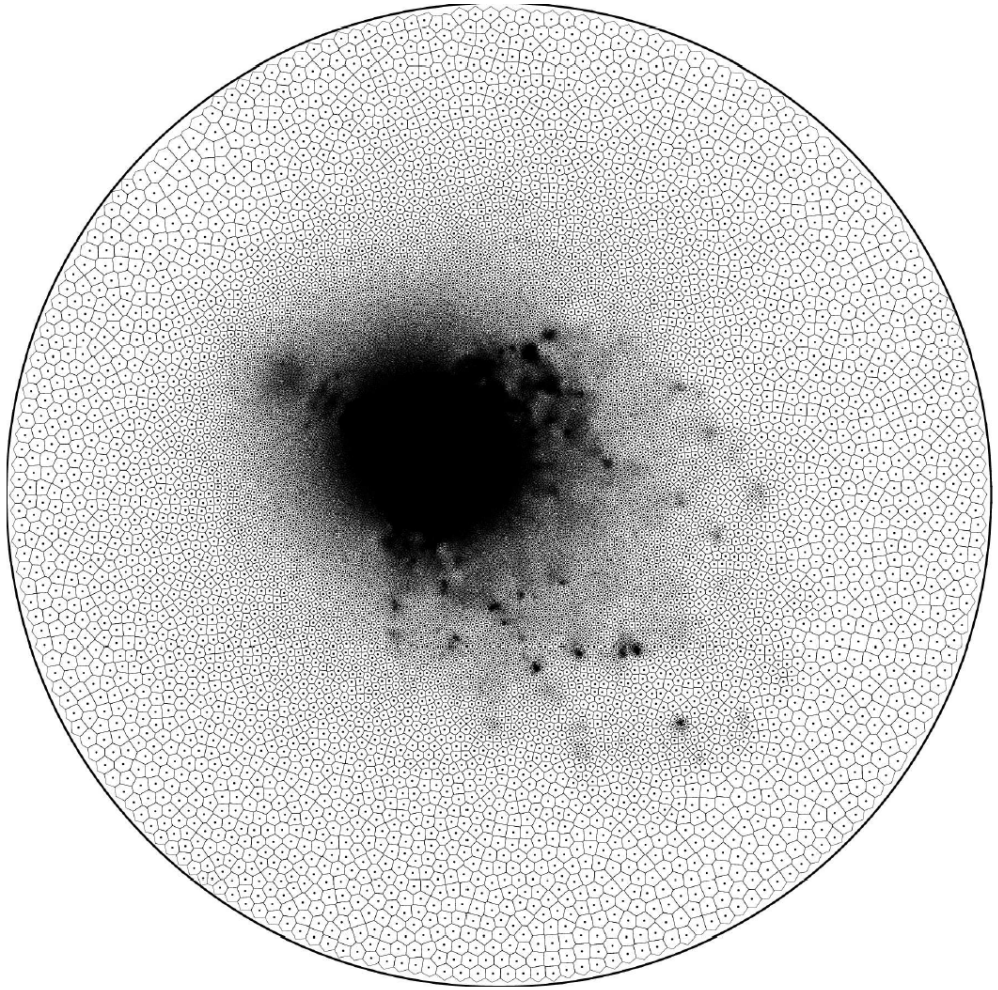


Iteration 15

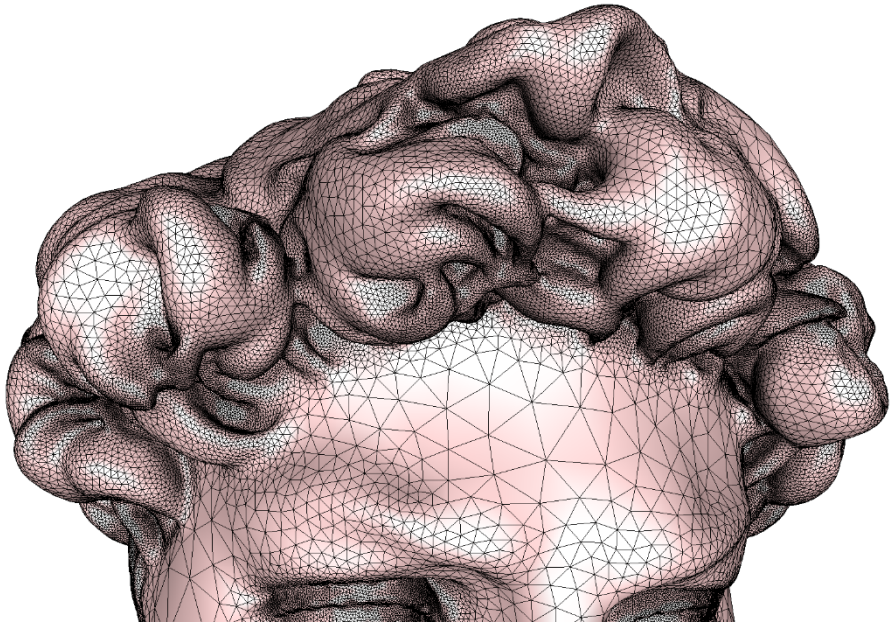
Uniform vs. Adaptive

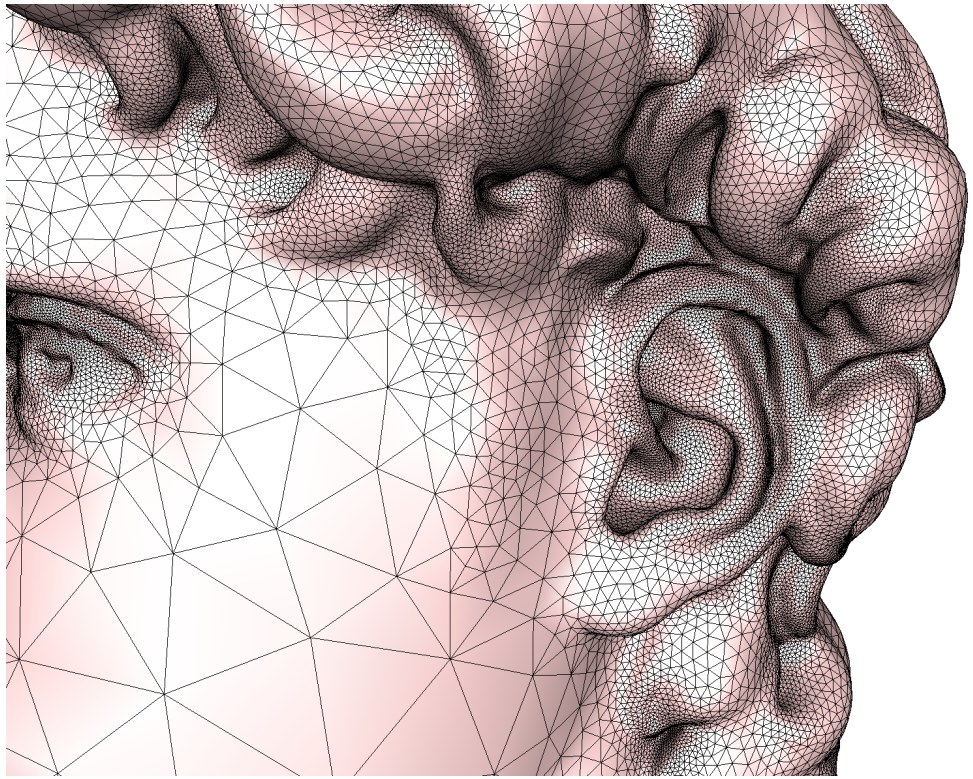


Uniform Sampling



Adaptive Sampling



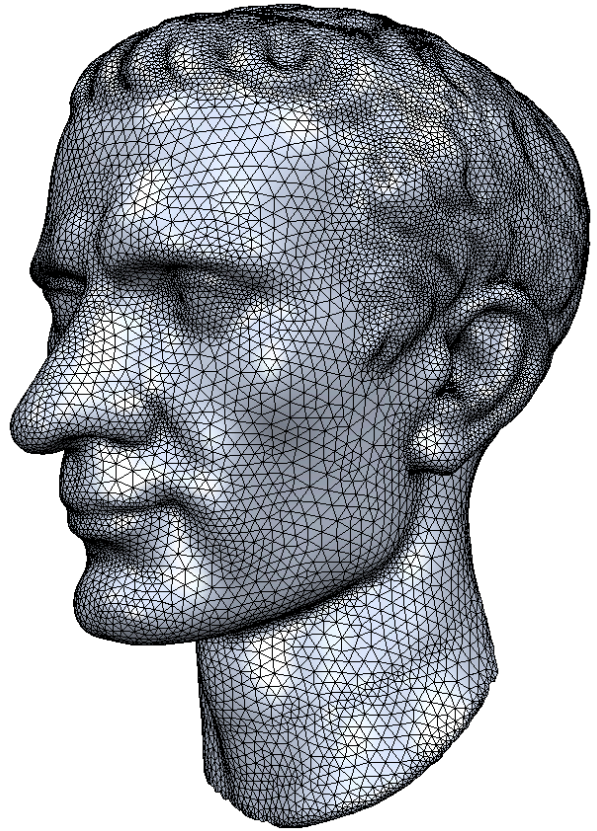


Outline

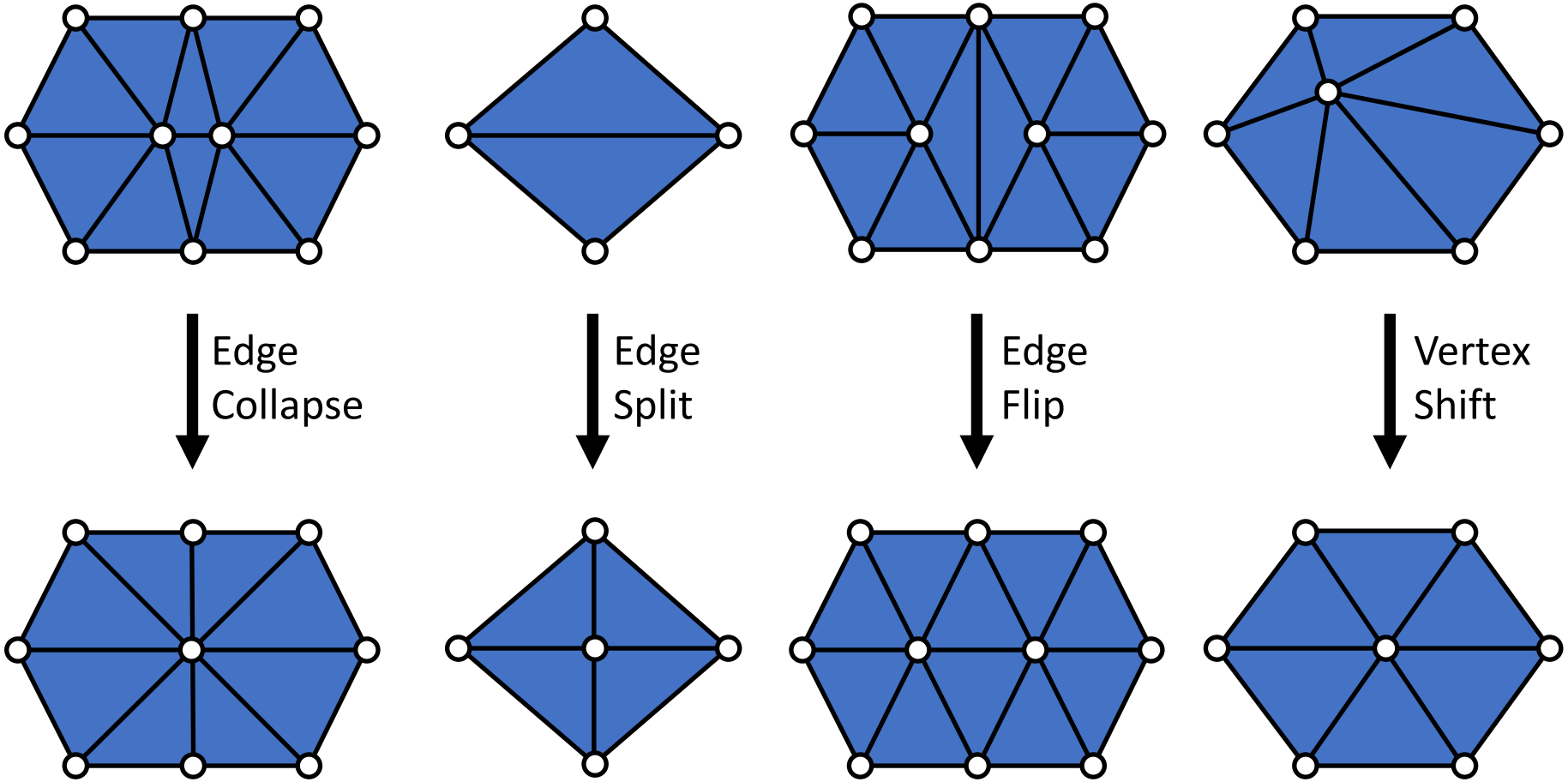
- Smoothing Motivation
- Spectral Analysis
- Diffusion Flow
- Remeshing Motivation
- Parametrization
- Global parameterization remeshing
- **Direct surface remeshing**

Direct Surface Remeshing

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



Local Remeshing Operators

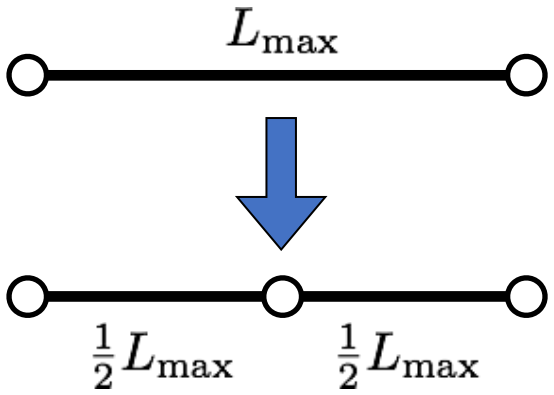


Isotropic Remeshing

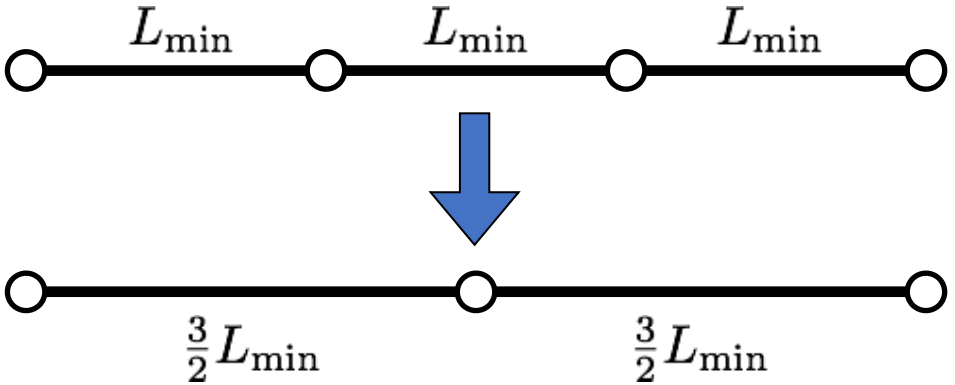
- Specify target edge length L

- 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 by tangential relaxation
  5. Project vertices onto reference mesh

Edge Collapse / Split



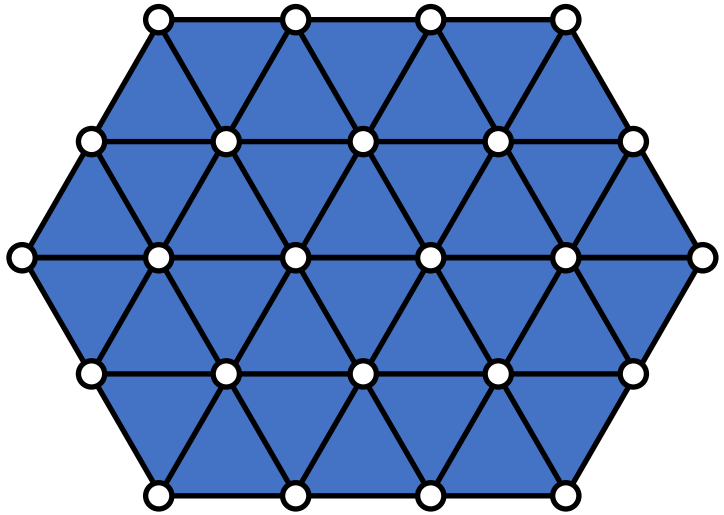
$$|L_{\max} - L| = \left|\tfrac{1}{2}L_{\max} - L\right|$$
$$\Rightarrow L_{\max} = \tfrac{4}{3}L$$



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

Edge Flip

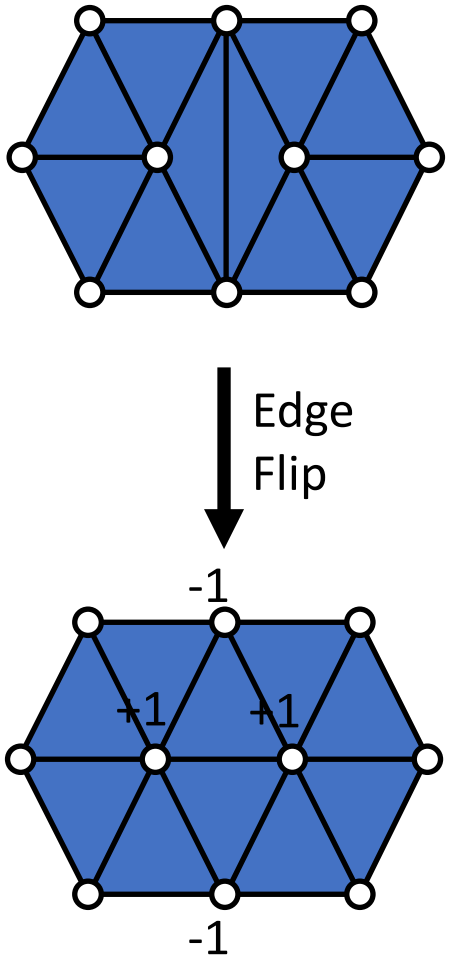
- Improve valences
  - Avg. valence is 6 (Euler)
  - Reduce variation
- Optimal valence is
  - 6 for interior vertices
  - 4 for boundary vertices



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

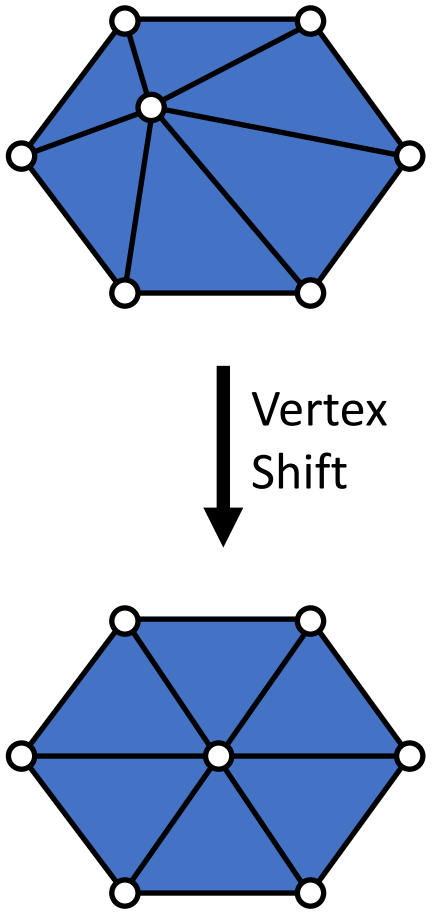
$$\sum_{i=1}^{4} \left(\text{valence}\left(v_i\right) - \text{opt\_valence}\left(v_i\right)\right)^2$$



Vertex Shift

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

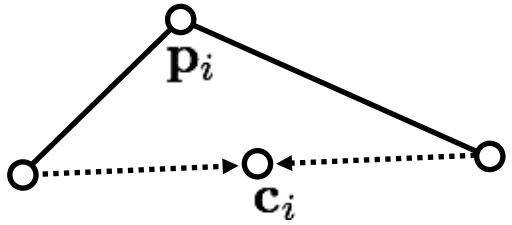
$$\mathbf{c}_i = \frac{1}{\text{valence}\,(v_i)} \sum_{j \in N(v_i)} \mathbf{p}_j$$



Vertex Shift

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

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



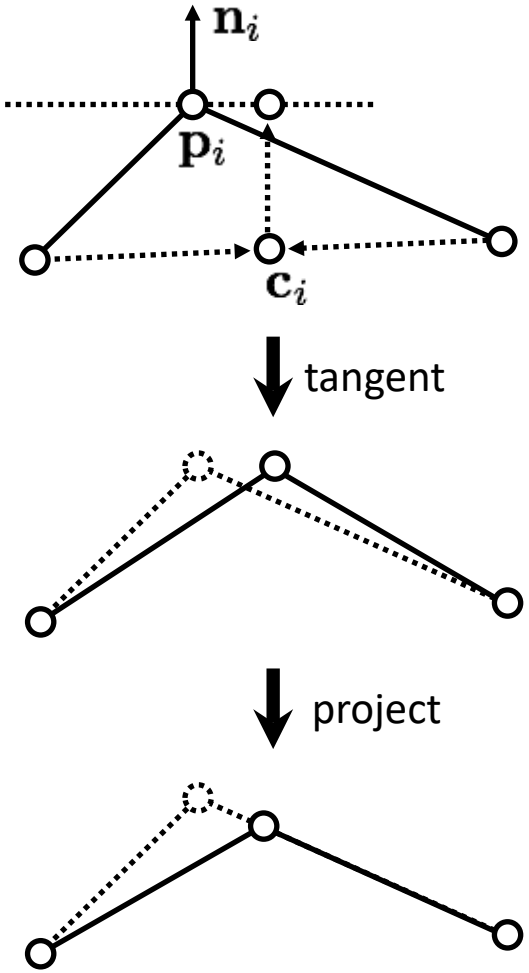
Vertex Shift

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

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

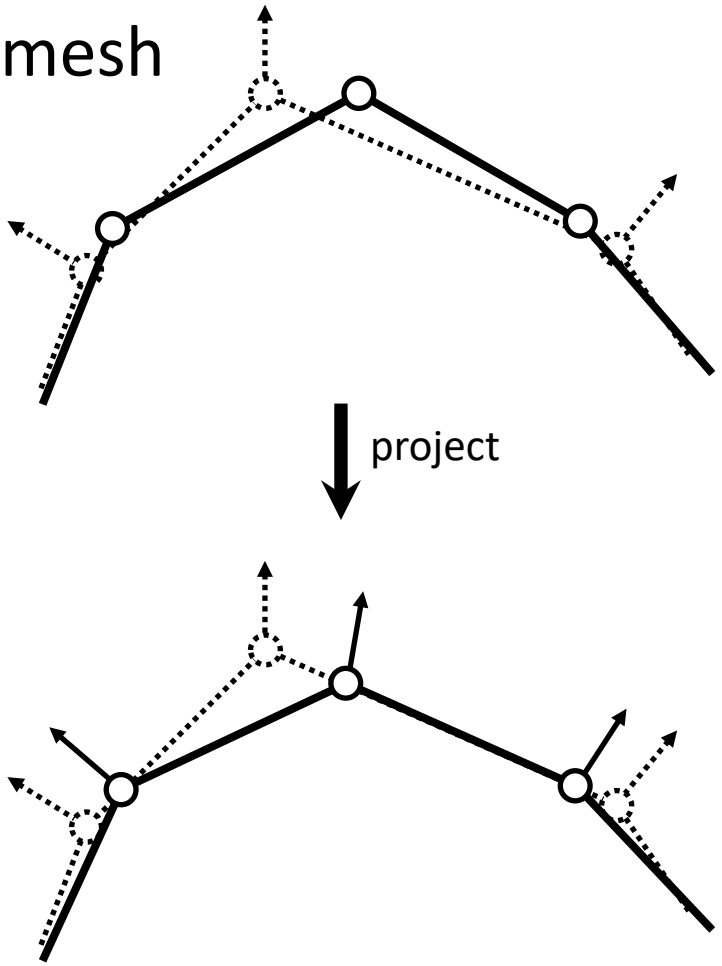
- Keep vertex (approx.) of surface
  - Restrict movement to tangent plane

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



Vertex Projection

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

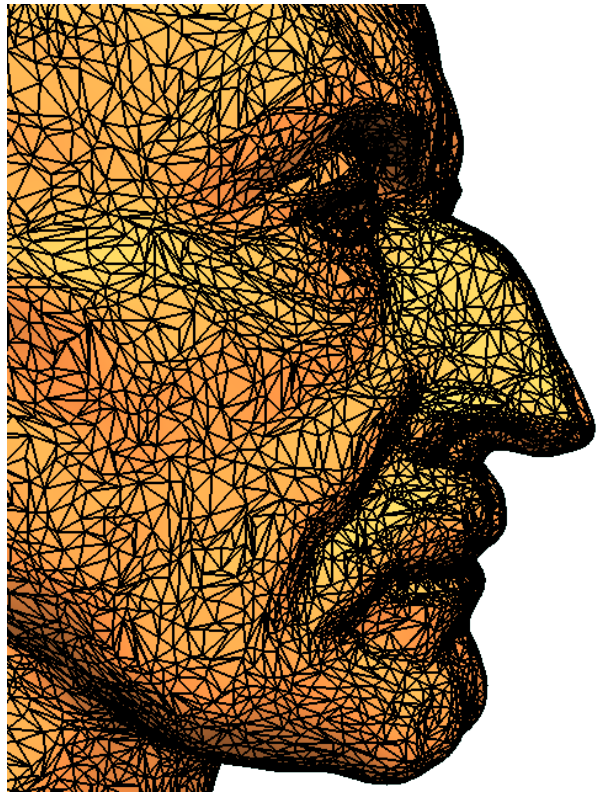


Isotropic Remeshing

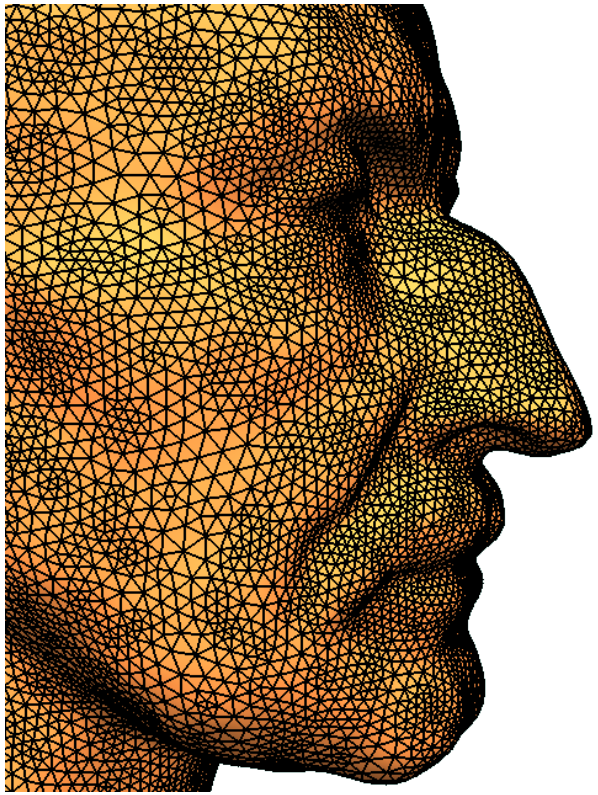
- Specify target edge length L

- 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 by tangential relaxation
  5. Project vertices onto reference mesh

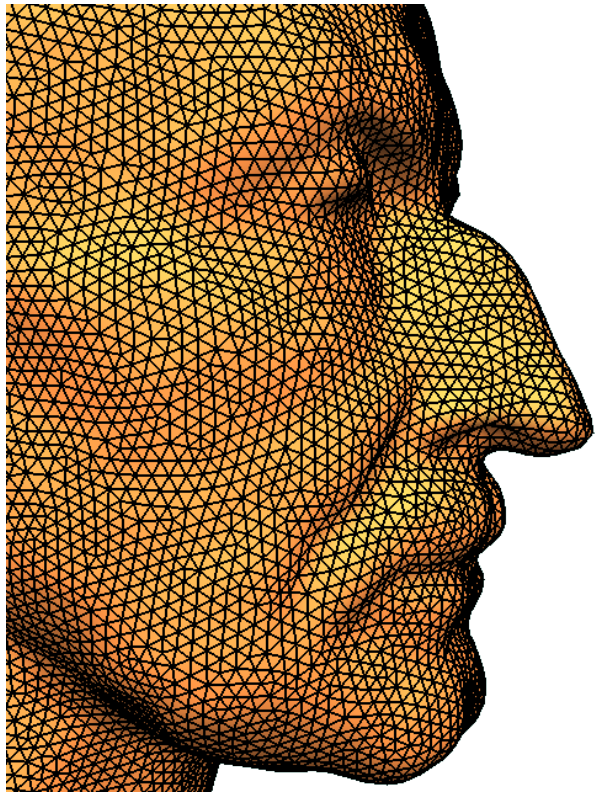
Remeshing Results



Original

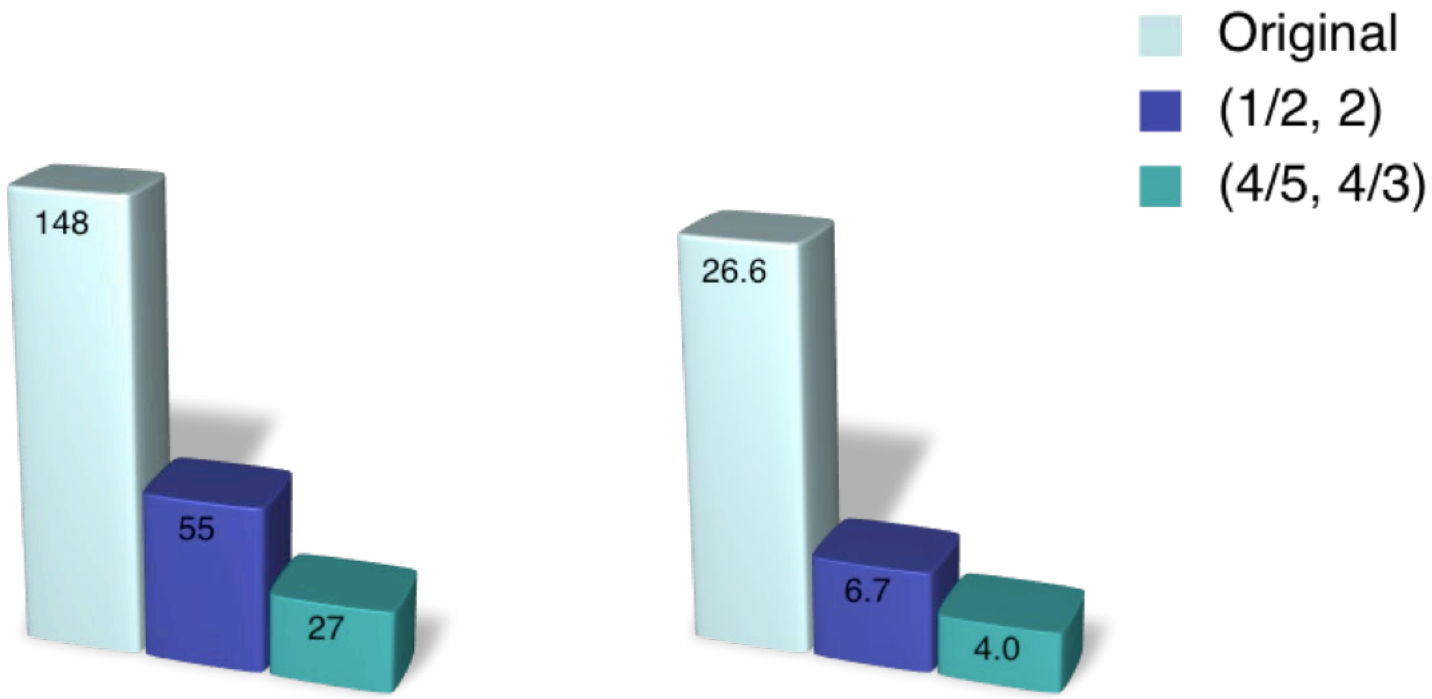


$\left(\frac{1}{2}, 2\right)$



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

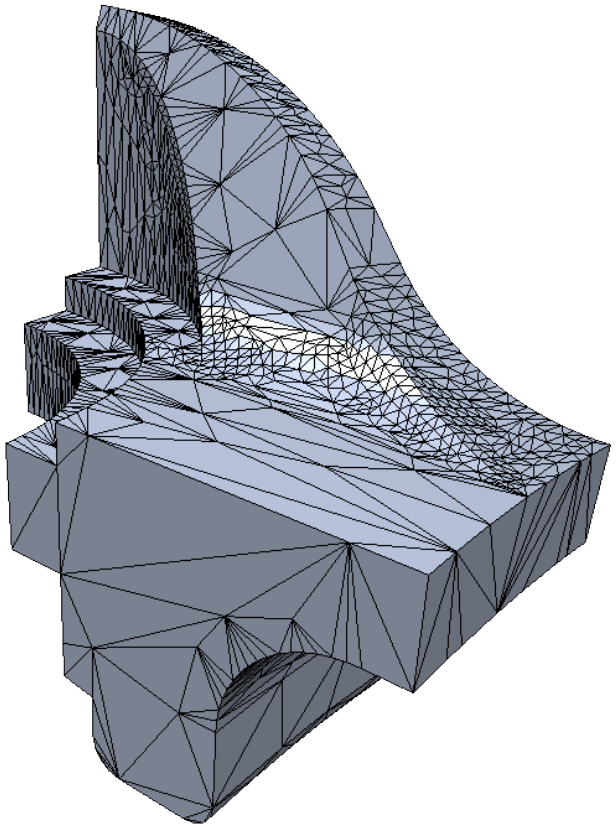
Remeshing Results

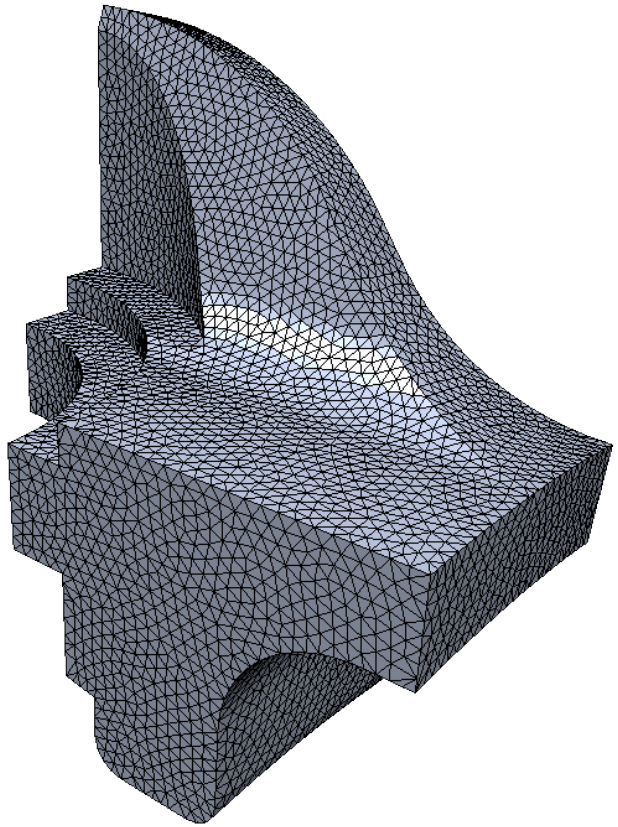


Edge length deviation (%)

Deviation from 60º

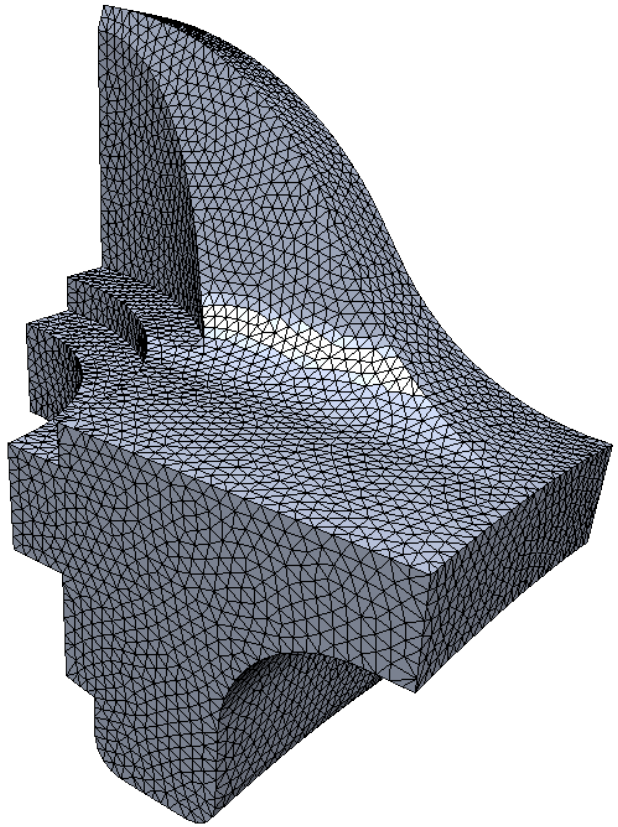
Feature Preservation?





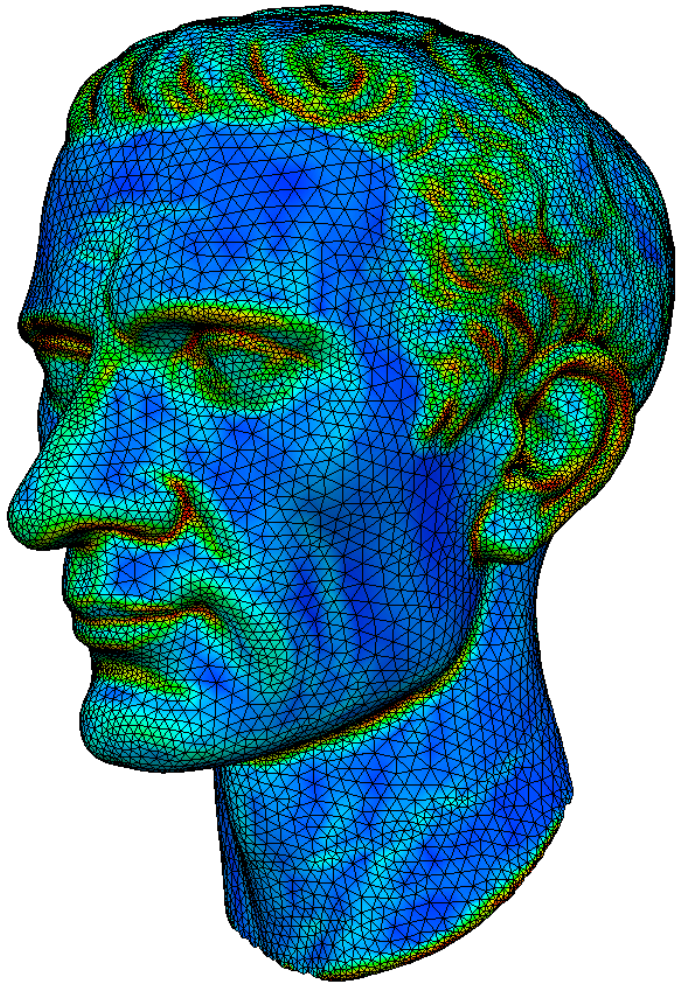
Feature Preservation

- Define features
  - Sharp edges
  - Material boundaries
- Adjust local operators
  - Don’t flip
  - Collapse only along features
  - Univariate smoothing
  - Project to feature curves



Adaptive Remeshing

- Precompute max. curvature on referen
- Target edge length locally determined k
- Adjust split / collapse criteria



Summary: Two Fundamental Approaches

- Surface oriented
  - operate directly of the surface
  - treat surface as a set of points / polygons in space
  - efficient for high resolution remeshing

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

References

- Alliez et al, *“Interactive geometry remeshing”*, SIGGRAPH 2002
- Alliez et al, *“Isotropic surface remeshing”*, SMI 2003
- Botsch & Kobbelt, *“A remeshing approach to multiresolution modeling”*, SGP 2004
- **Alliez et al, *“Recent advances in remeshing of surfaces”*, AIM@Shape state of the art report, 2006**

That’s All