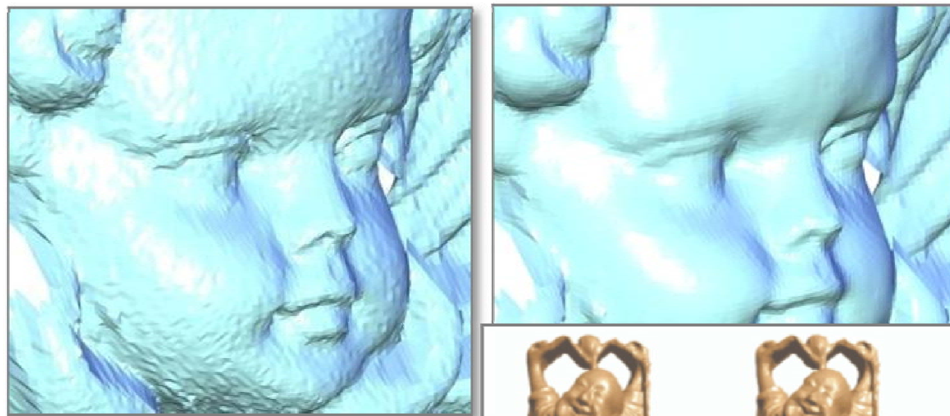
G22.3033-008, Spring 2010

Geometric Modeling

Digital Geometry Processing

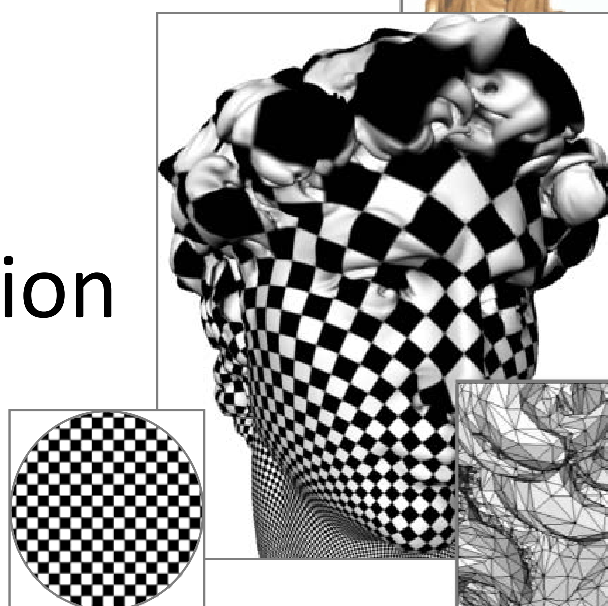
Topics

- Smoothing

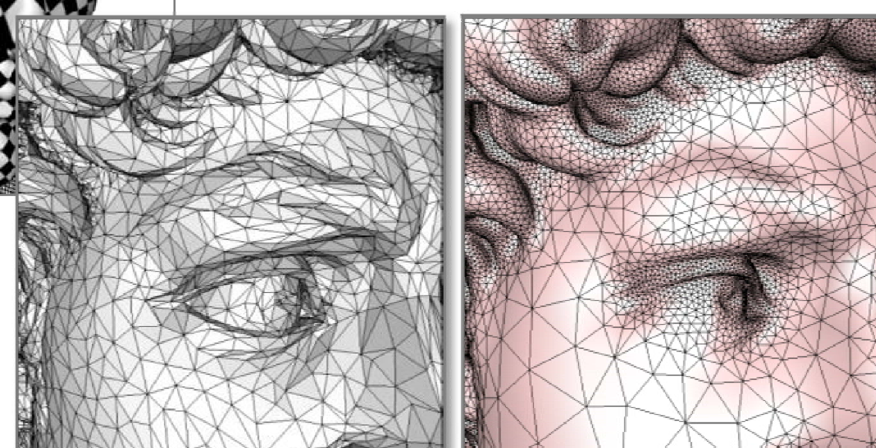


- Simplification

- Parameterization



- Remeshing



Mesh Smoothing

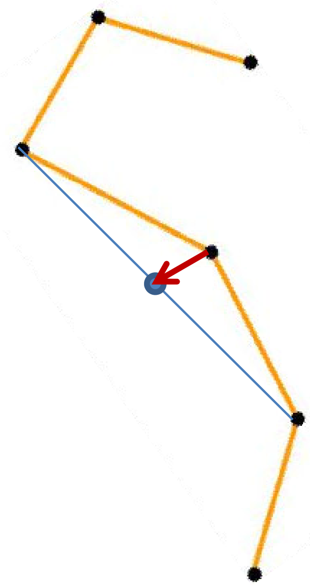
Laplacian curve and surface smoothing

Implicit fairing

Laplacian mesh optimization

Laplacian smoothing

2D Curve



- Discrete Laplacian for a single vertex

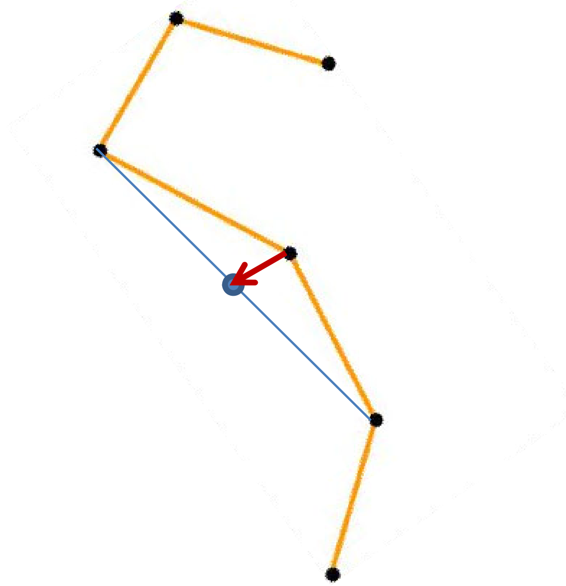
$$L(\mathbf{x}_i) = \frac{1}{2}(\mathbf{x}_{i\text{-}1} - \mathbf{x}_i) + \frac{1}{2}(\mathbf{x}_{i+1} - \mathbf{x}_i)$$

- In matrix-vector form for the whole curve

$$L\mathbf{x}$$

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

Smoothing



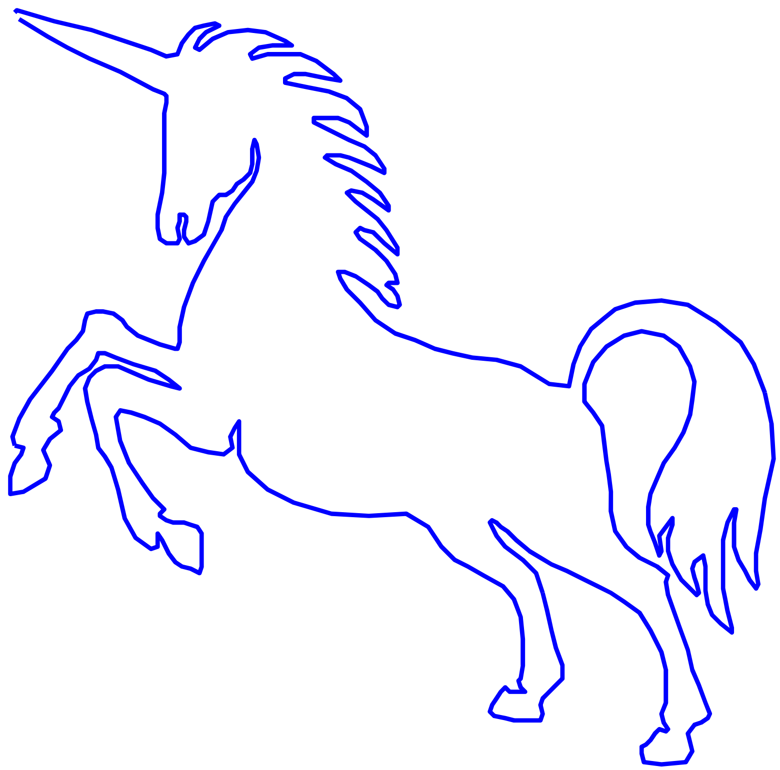
- Gaussian filtering

$$\mathbf{x}_i' = \mathbf{x}_i + \lambda L(\mathbf{x}_i)$$

- Scale factor $0 < \lambda < 1$
- Matrix-vector form $\mathbf{x}' = \mathbf{x} + \lambda L\mathbf{x}$
- Works identical for surface smoothing
  - Choose (normalized) Laplacian weights
- Drawbacks
  - Causes the curve/mesh to shrink

Laplacian smoothing

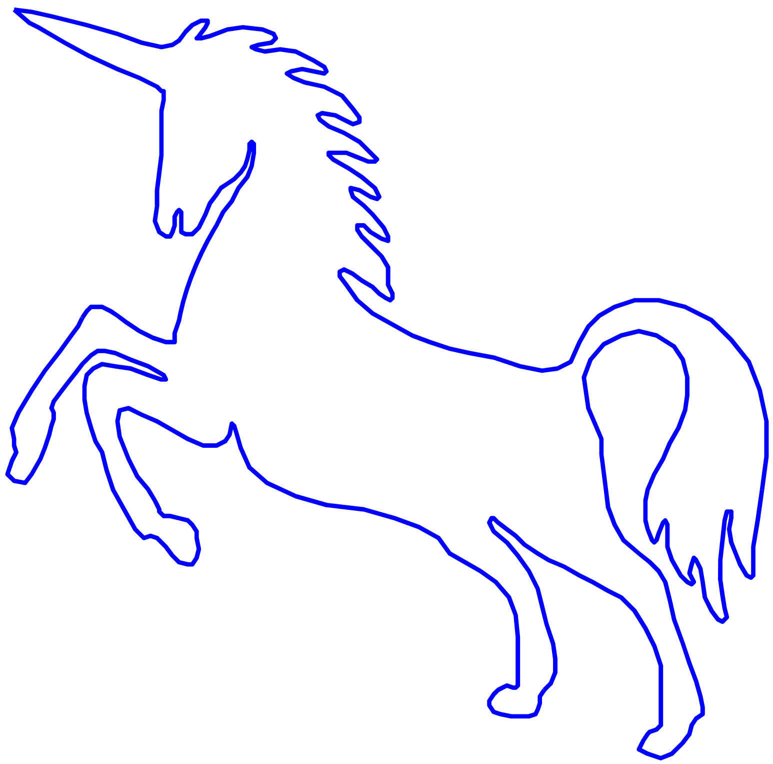
2D Curve – Example



Original curve

Laplacian smoothing

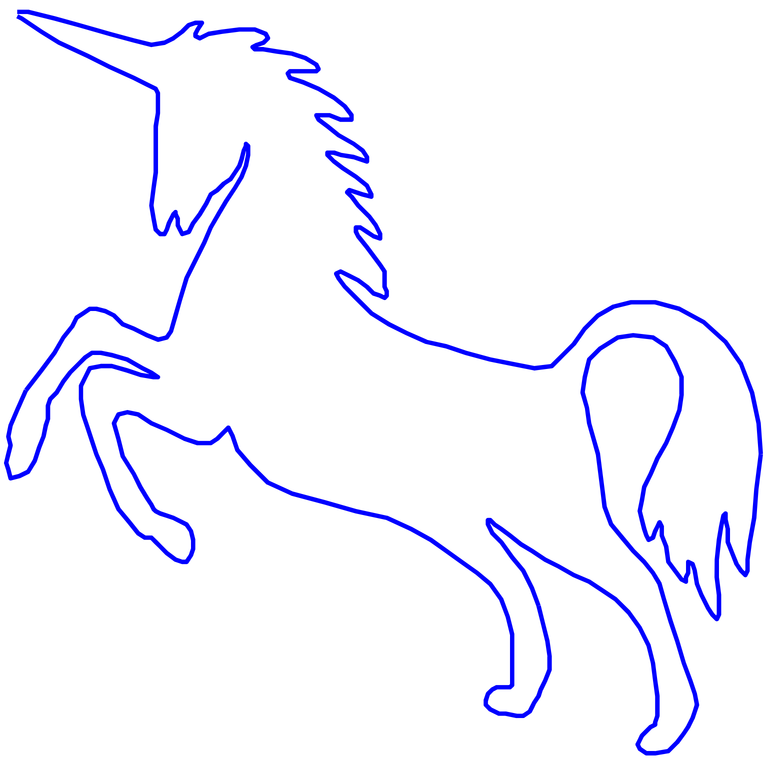
2D Curve – Example



1st iteration; $\lambda=0.5$

Laplacian smoothing

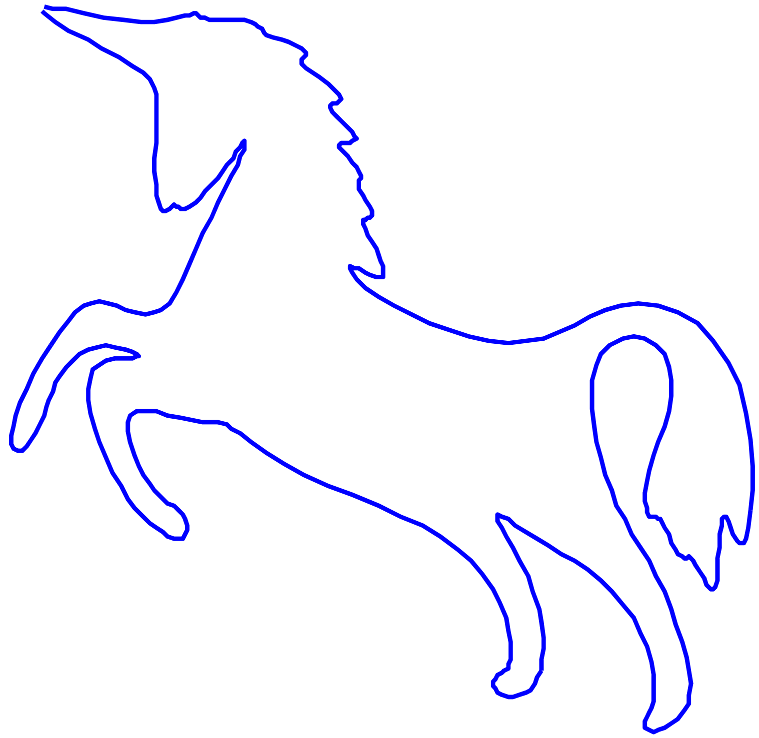
2D Curve – Example



2nd iteration; $\lambda=0.5$

Laplacian smoothing

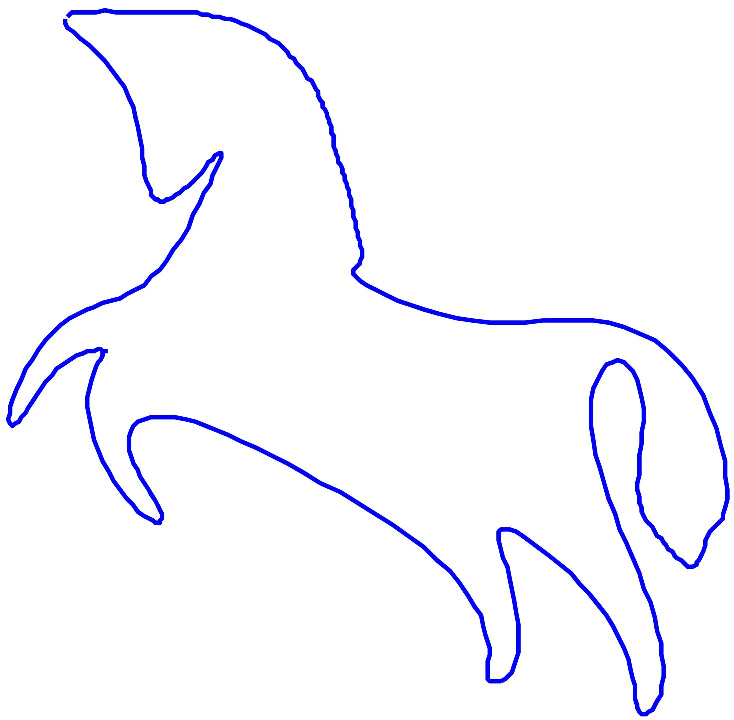
2D Curve – Example



8th iteration; $\lambda=0.5$

Laplacian smoothing

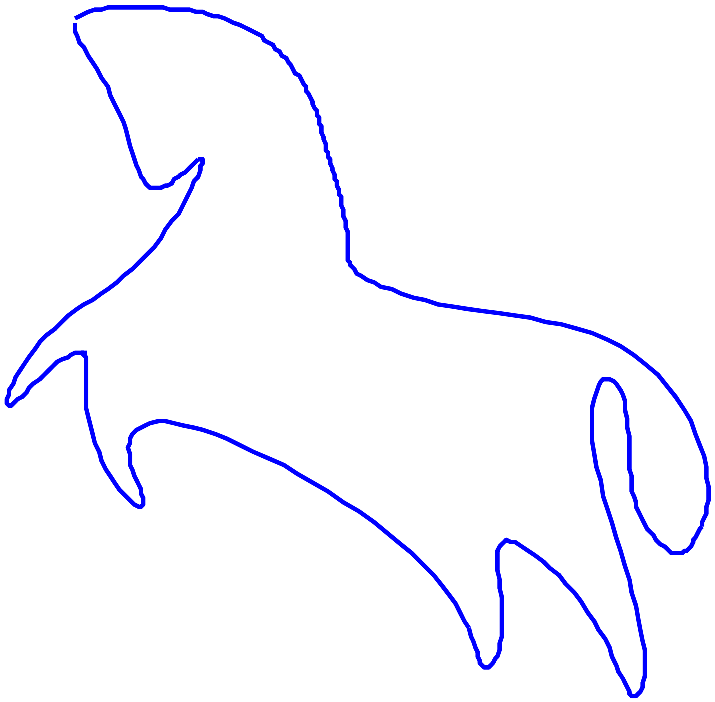
2D Curve – Example



27th iteration; $\lambda=0.5$

Laplacian smoothing

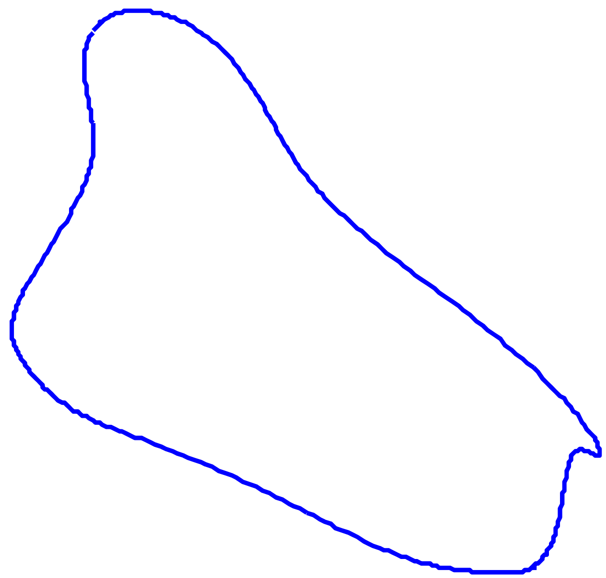
2D Curve – Example



50th iteration; $\lambda=0.5$

Laplacian smoothing

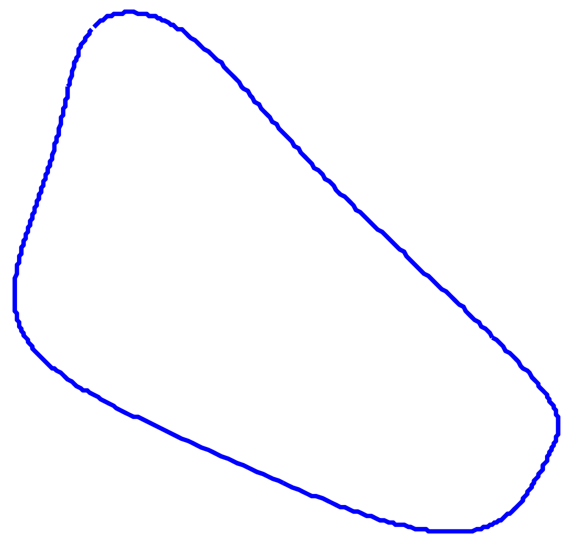
2D Curve – Example



500th iteration; $\lambda=0.5$

Laplacian smoothing

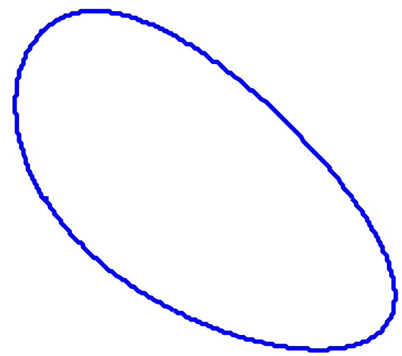
2D Curve – Example



1000th iteration; $\lambda=0.5$

Laplacian smoothing

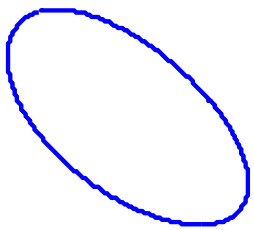
2D Curve – Example



5000th iteration; $\lambda=0.5$

Laplacian smoothing

2D Curve – Example



10000th iteration; $\lambda=0.5$

Laplacian smoothing

2D Curve – Example

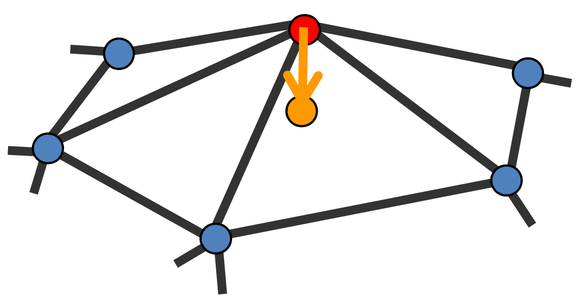
50000th iteration; $\lambda=0.5$

$$\Delta_M \mathbf{p} = -H\mathbf{n}$$

Recap

Laplace-Beltrami operator

- High-pass filter: extracts local surface detail
- Detail = *smooth*(surface) – surface
- Smoothing = averaging



First attempt at definition: uniform weighting

$$\boldsymbol{\delta}_i = \frac{1}{d_i} \sum_{\mathbf{v}_j \in N_1(\mathbf{v}_i)} \mathbf{v}_j - \mathbf{v}_i$$

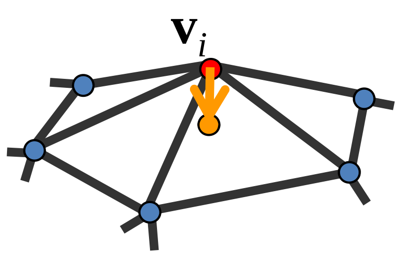
$$\boldsymbol{\delta}_i = \sum_{\mathbf{v}_j \in N_1(\mathbf{v}_i)} \frac{1}{d_i} \left(\mathbf{v}_j - \mathbf{v}_i \right)$$

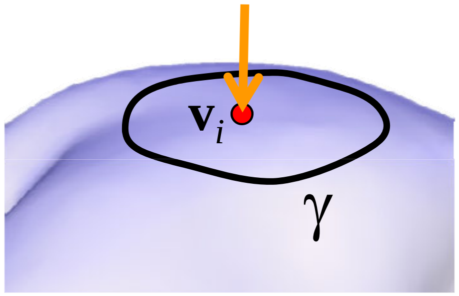
$$\Delta_M \mathbf{p} = -H\mathbf{n}$$

Recap

Laplace-Beltrami operator

- The direction of $\boldsymbol{\delta}_i$ approximates the normal
- The size approximates the mean curvature





$$\boldsymbol{\delta}_i = \frac{1}{d_i} \sum_{\mathbf{v}_j \in N_1(\mathbf{v}_i)} \left(\mathbf{v}_j - \mathbf{v}_i\right) \qquad \boldsymbol{\delta}_i = \frac{1}{len(\gamma)} \int_{s=a}^{b} (\gamma(s) - \mathbf{v}_i)\, ds$$

$$\lim_{len(\gamma)\to 0} \frac{1}{len(\gamma)} \int_{s=a}^{b} (\gamma(s) - \mathbf{v}_i)\, ds = -H(\mathbf{v}_i)\,\mathbf{n}_i$$

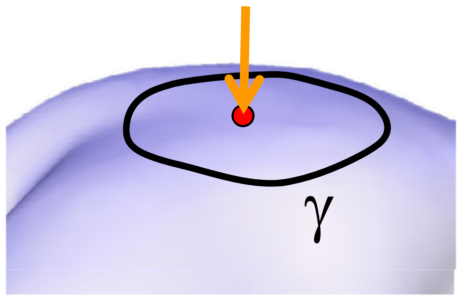
Recap

Discrete Laplace-Beltrami operator – weighting schemes

$$\boldsymbol{\delta}_i = \frac{1}{A_i} \sum_{\mathbf{v}_j \in N_1(\mathbf{v}_i)} w_{ij} \left(\mathbf{v}_j - \mathbf{v}_i \right)$$

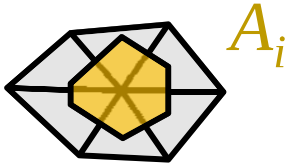
- Ignore geometry

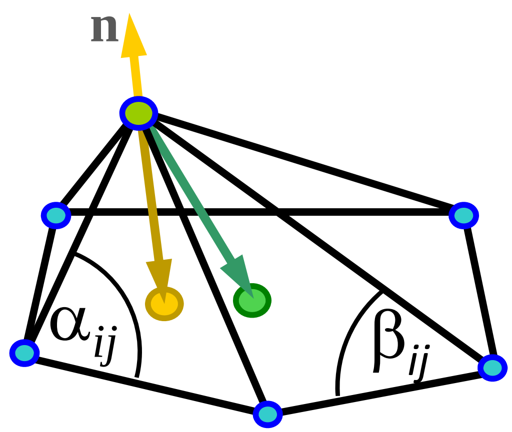
  $\delta_{\text{umbrella}} : A_i = 1,\ w_{ij} = 1/d_i$



- Integrate over Voronoi region of the vertex

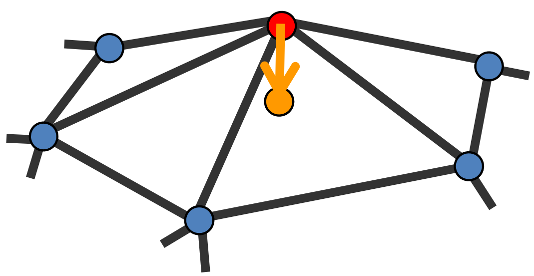
  $\delta_{\text{cotangent}} : w_{ij} = \cot \alpha_{ij} + \cot \beta_{ij}$





Laplacian matrix

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



$$\boldsymbol{\delta}_i = \frac{1}{A_i} \sum_{\mathbf{v}_j \in N_1(\mathbf{v}_i)} w_{ij} \left(\mathbf{v}_j - \mathbf{v}_i \right)$$

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

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

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

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

Taubin smoothing

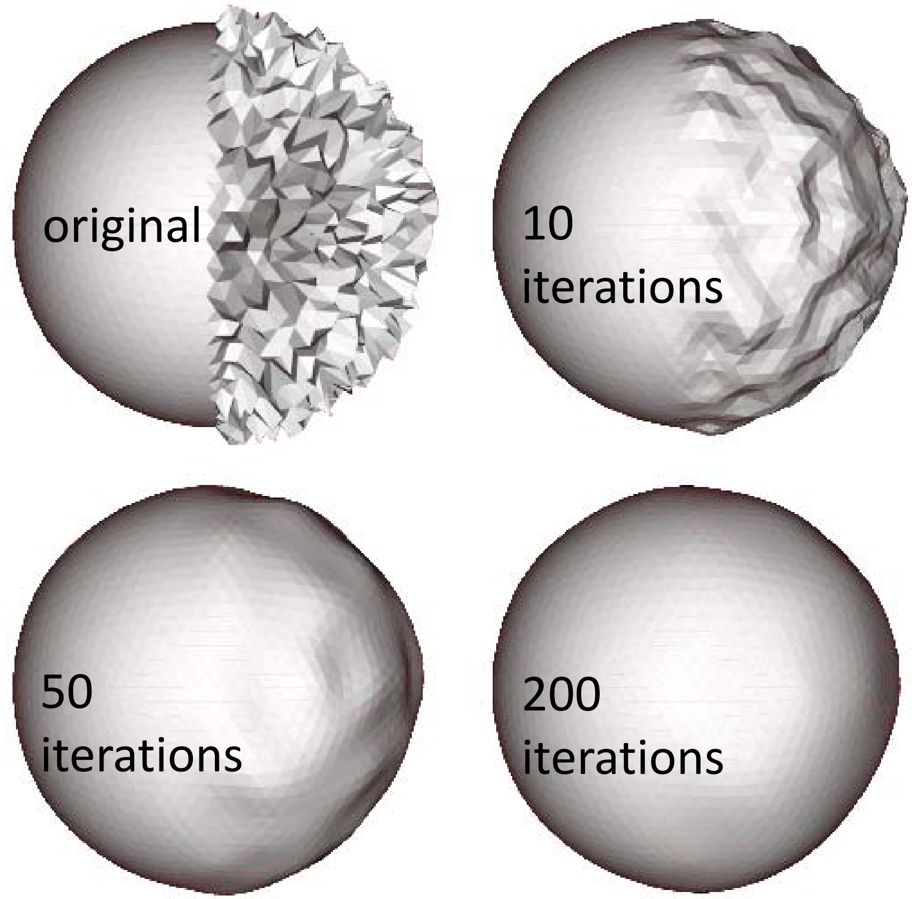
Explicit steps

- Iterate:

$$\mathbf{x}' = \mathbf{x} + \lambda L\mathbf{x} = (\mathrm{I} + \lambda L)\mathbf{x}$$

$$\mathbf{x}' = \mathbf{x} + \mu L\mathbf{x} = (\mathrm{I} + \mu L)\mathbf{x}$$

- $\lambda > 0$ to smooth; $\mu < 0$ to inflate
- Originally proposed with uniform Laplacian weights



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

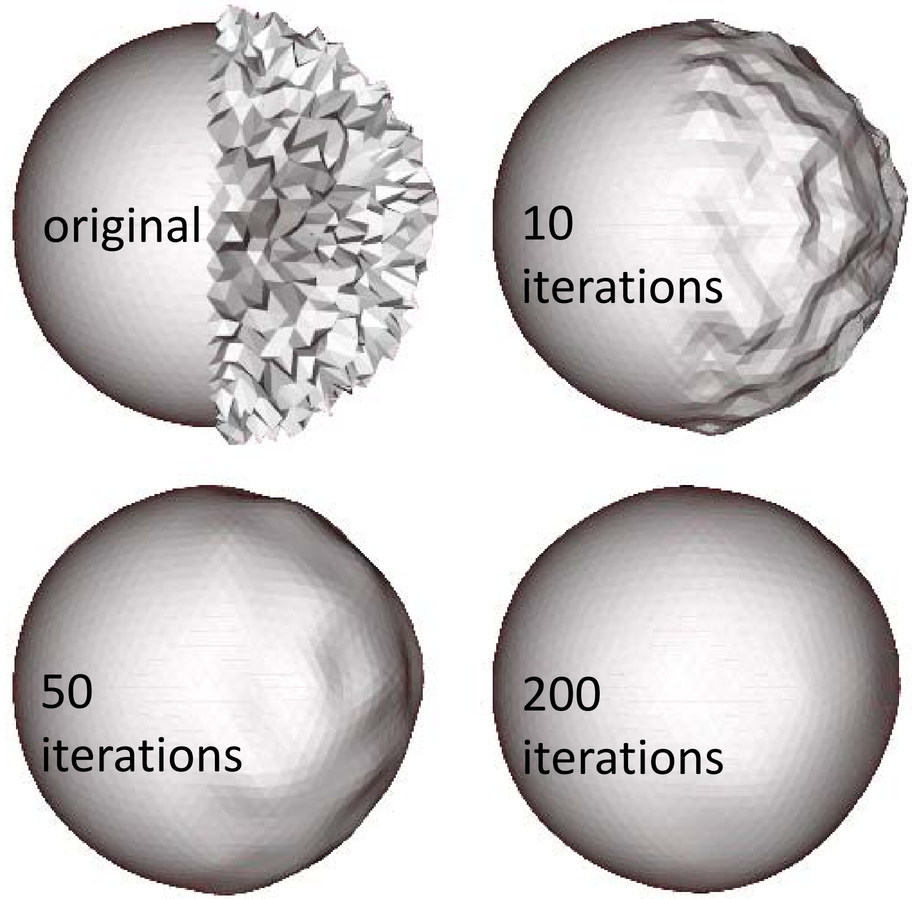
Taubin smoothing

Explicit steps

- Per-vertex iterations

$$\mathbf{x}_i' = \mathbf{x}_i + \lambda L(\mathbf{x}_i)$$
$$\mathbf{x}_i' = \mathbf{x}_i + \mu L(\mathbf{x}_i)$$

- Simple to implement
- Requires many iterations
- Need to tweak μ and λ

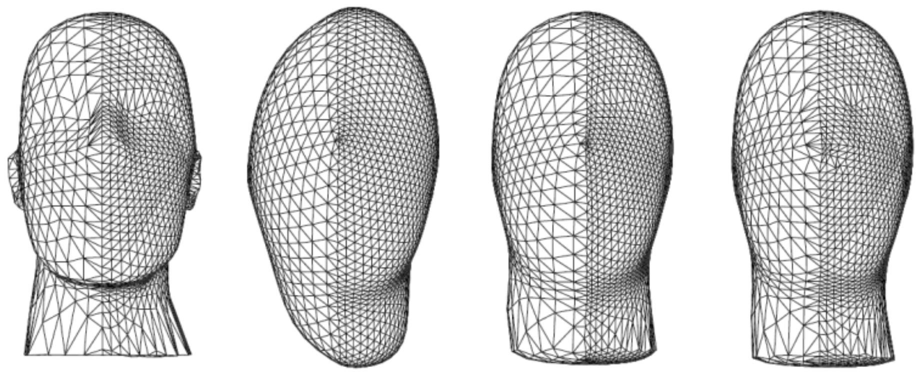


Implicit fairing of irregular meshes using diffusion and curvature flow
M. Desbrun, M. Meyer, P. Schroeder, A. Barr
ACM SIGGRAPH 99

Implicit fairing

Implicit Euler steps

- Use cotangent instead of uniform Laplacian



- In each iteration, solve for the smoothed $\mathbf{x}'$:

$$(I - \lambda L)\mathbf{x}' = \mathbf{x}$$

Implicit fairing of irregular meshes using diffusion and curvature flow
M. Desbrun, M. Meyer, P. Schroeder, A. Barr
ACM SIGGRAPH 99

Implicit fairing

- Model smoothing as a diffusion process

$$\frac{\partial \mathbf{x}}{\partial t} = \lambda L(\mathbf{x})$$

- Scale λ by simulation parameter time t

$$\mathbf{x}^{n+1} = (I + \lambda \, \mathrm{d}t \, L) \, \mathbf{x}^n$$

- Backward Euler for unconditional stability

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \lambda \, \mathrm{d}t \, L(\mathbf{x}^{n+1})$$

$$(I - \lambda \, \mathrm{d}t \, \mathrm{L})\mathbf{x}^{n+1} = \mathbf{x}^n$$

Implicit fairing of irregular meshes using diffusion and curvature flow
M. Desbrun, M. Meyer, P. Schroeder, A. Barr
ACM SIGGRAPH 99

Implicit fairing

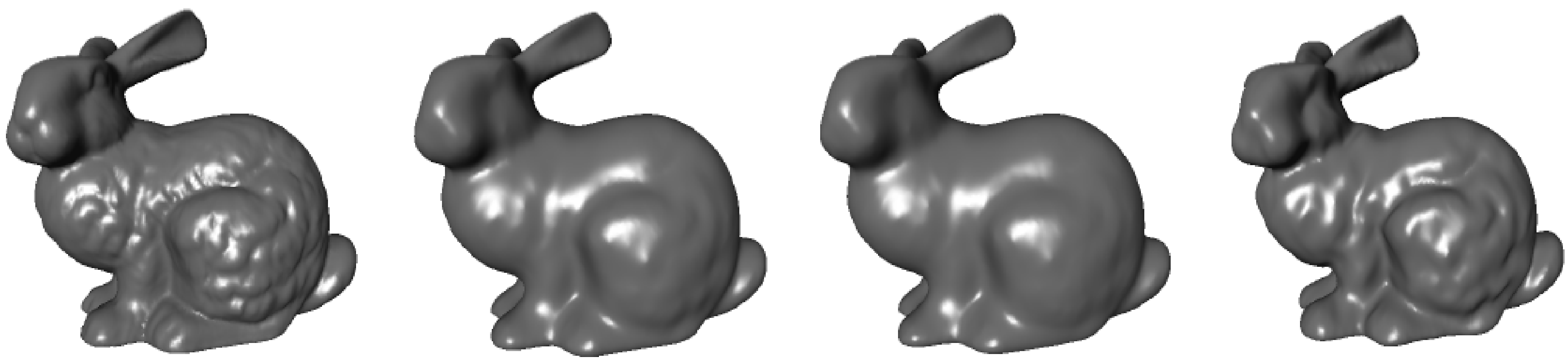
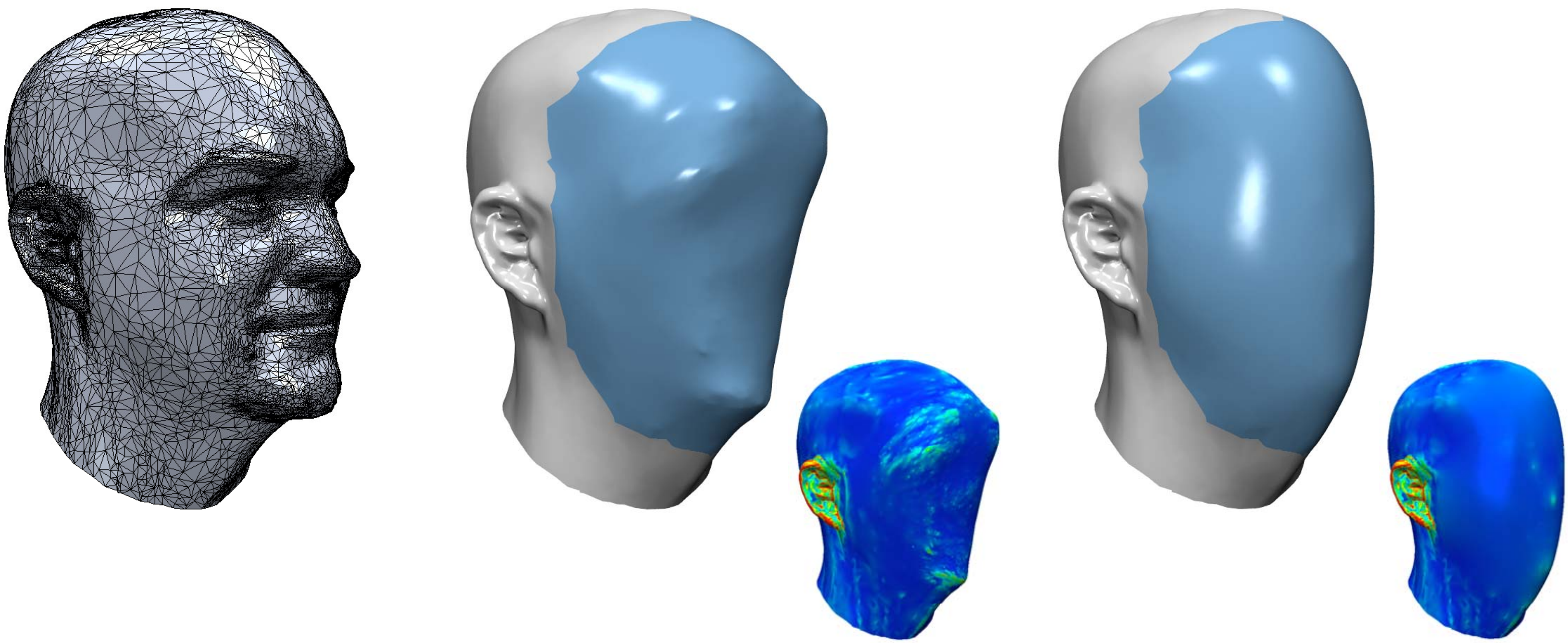


Figure 4: *Stanford bunnies: (a) The original mesh, (b) 10 explicit integrations with* $\lambda dt = 1$*, (c) 1 implicit integration with* $\lambda dt = 10$ *that takes only 7 PBCG iterations (30% faster), and (d) 20 passes of the* $\lambda|\mu$ *algorithm, with* $\lambda = 0.6307$ *and* $\mu = -0.6732$*.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

Implicit fairing

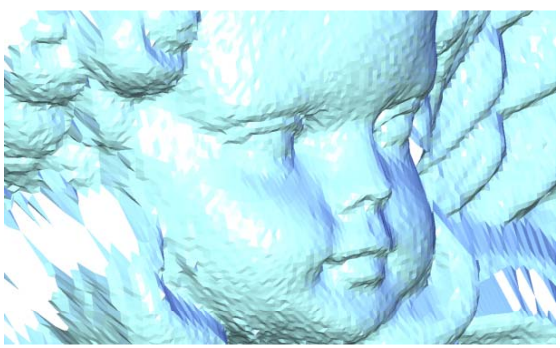
- Use cotangent instead of uniform Laplacian

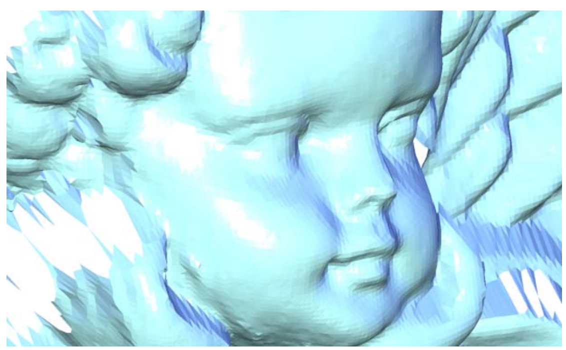


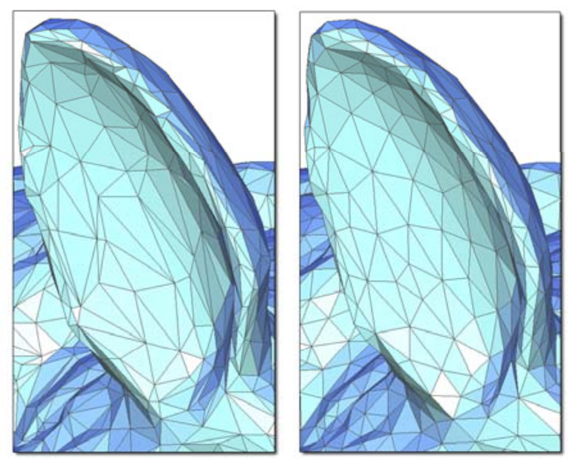
Laplacian mesh optimization

[Nealen et al. 2006]

- Smoothing, improving of triangle shapes
- Basic idea: formulate a “shopping list”, solve with LS
  - Smooth mesh: $L\mathbf{x}' = 0$
  - Passes close to input data set: $\mathbf{x}' = \mathbf{x}$
  - Well-shaped triangles: $L_{\text{uni}}\,\mathbf{x}' = L_{\text{cot}}\,\mathbf{x}$
  - The terms are weighted according to importance and geo.

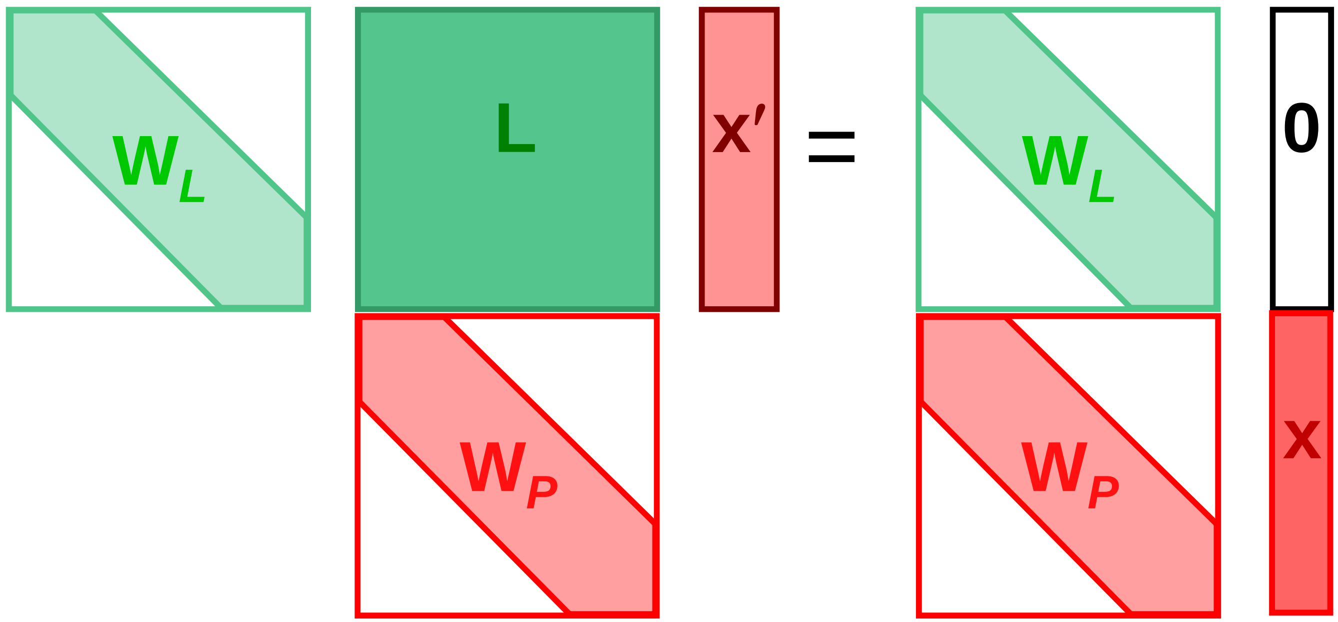






Laplacian mesh optimization

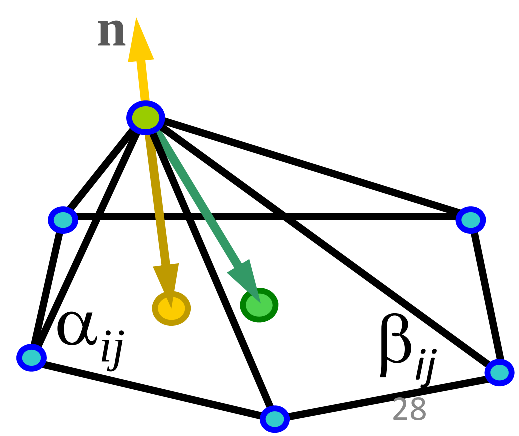
Smoothing

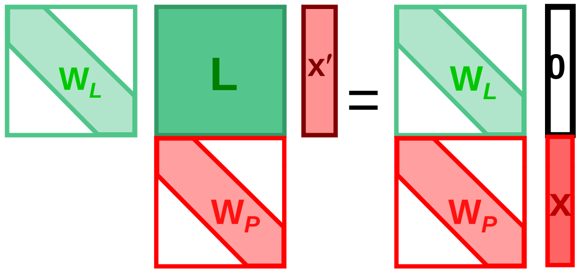


$$L\mathbf{x}' = 0$$

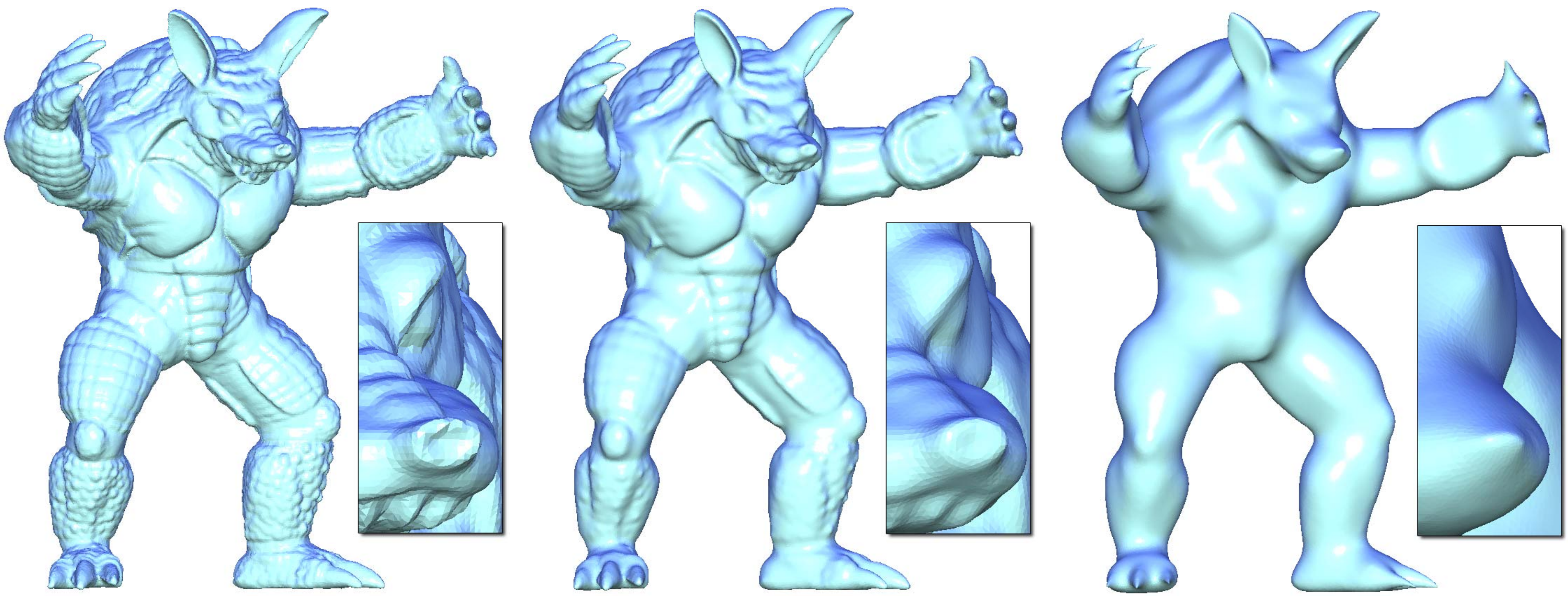
$$\mathbf{x}' = \mathbf{x}$$

- **Mesh smoothing** $L = L_{\mathrm{cot}}$ (outer fairness) or $L = L_{\mathrm{uni}}$ (outer and inner fairness)
- Controlled by W_{P} and W_{L} (Intensity, Features)





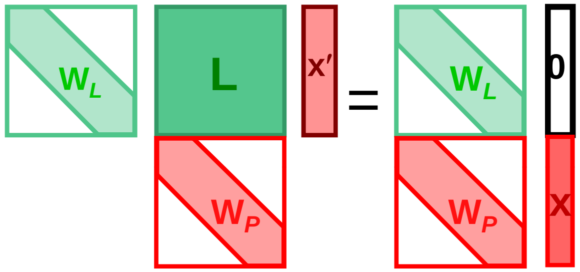
Using W_P



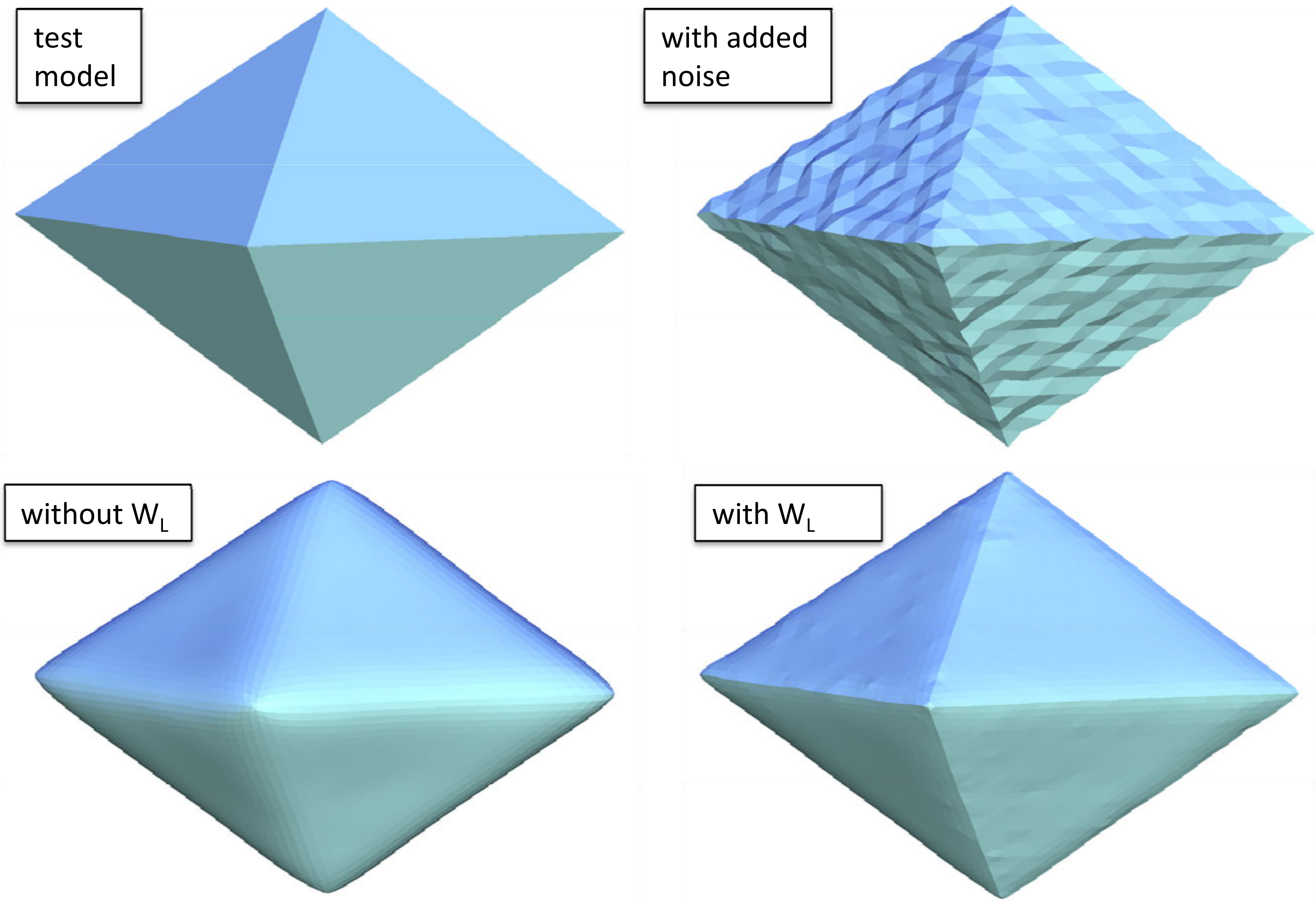
original

w = 0.2

w = 0.02

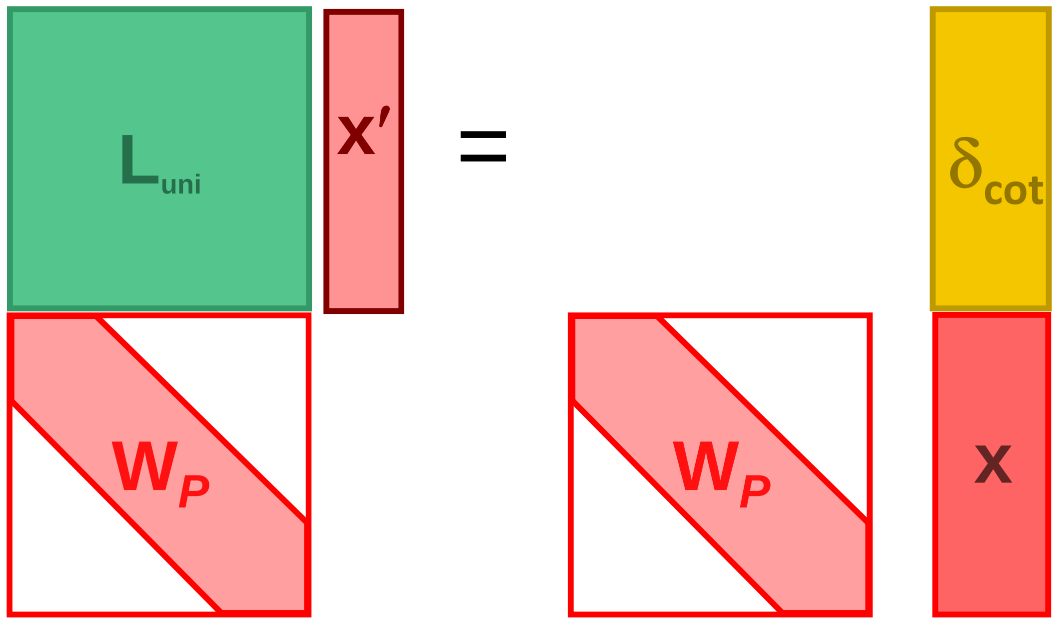


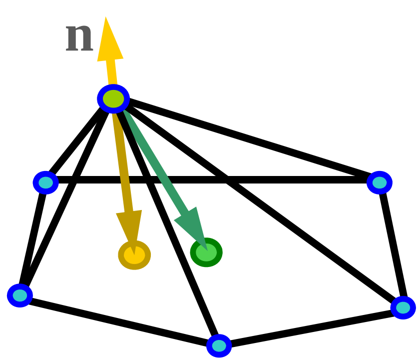
Using W_P and W_L



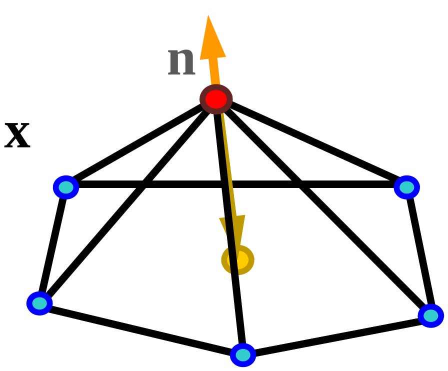
Triangle shape Optimization

By global vertex relocation

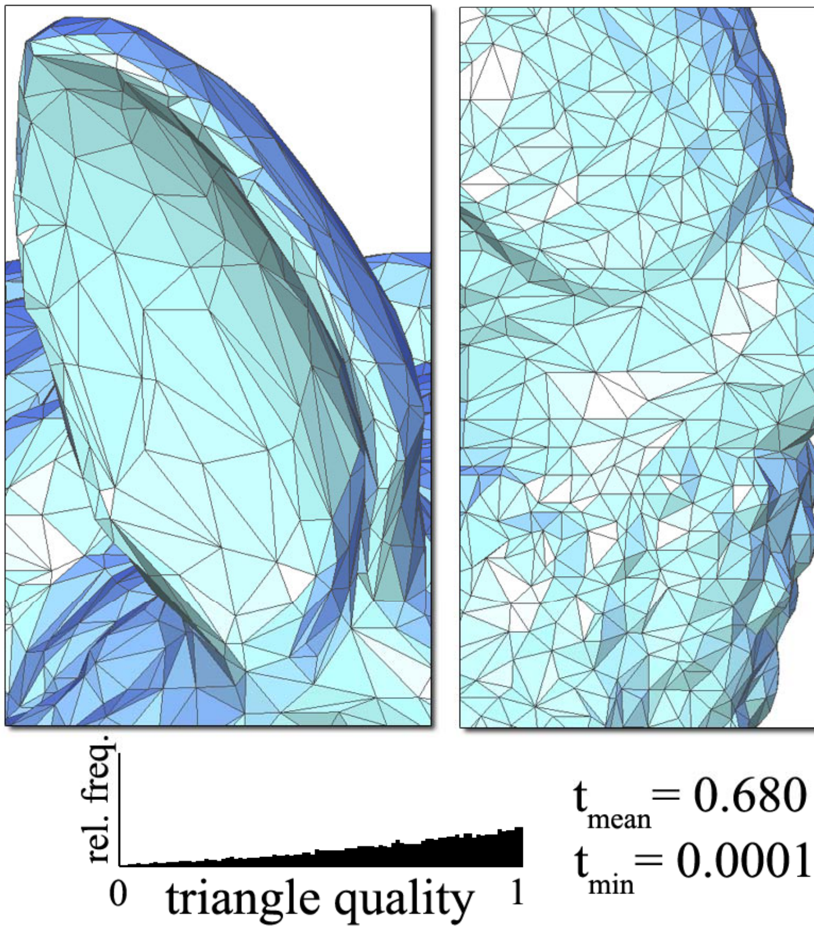




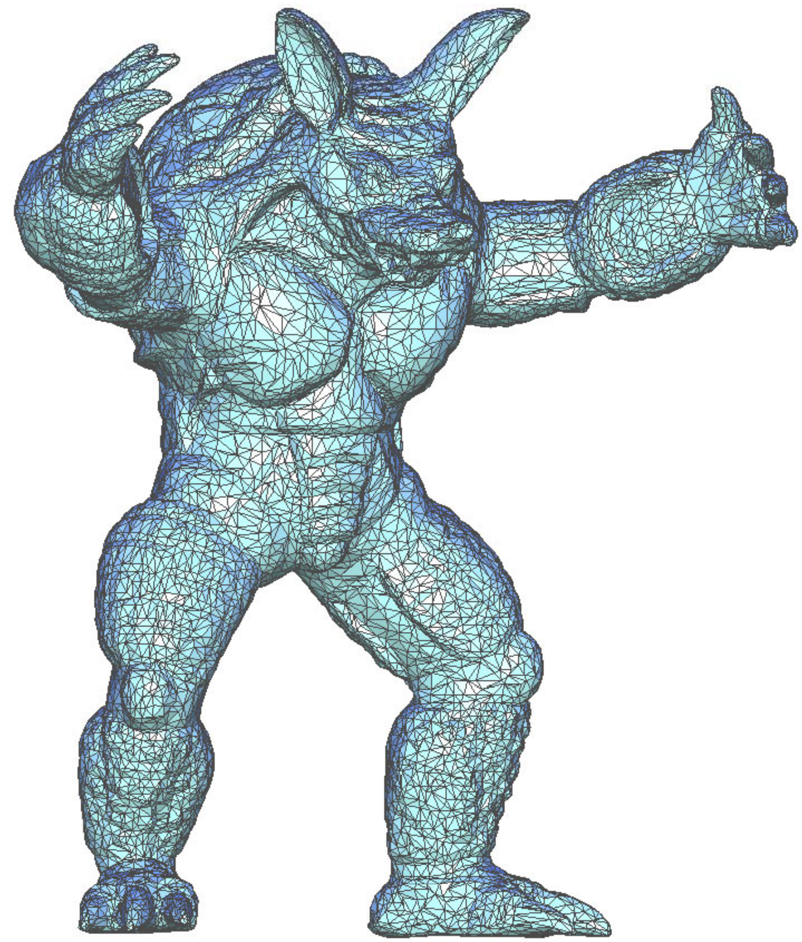
$$L_{uni}\,\mathbf{x}' = L_{cot}\,\mathbf{x}$$



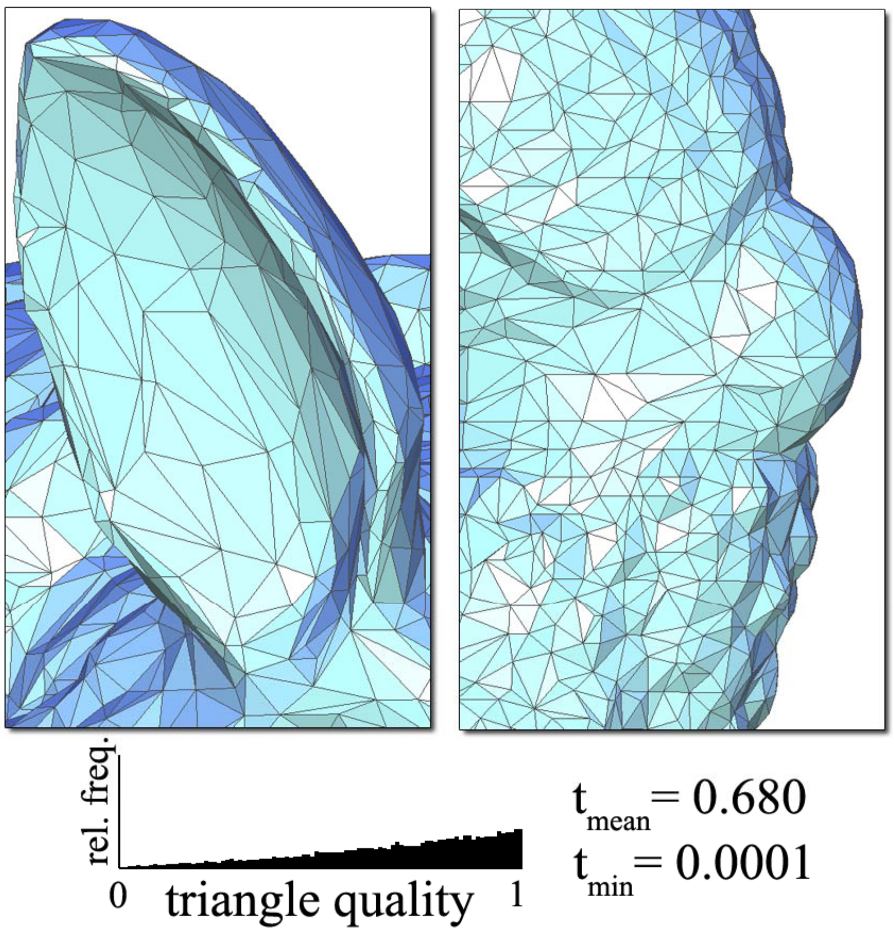
Positional Weights



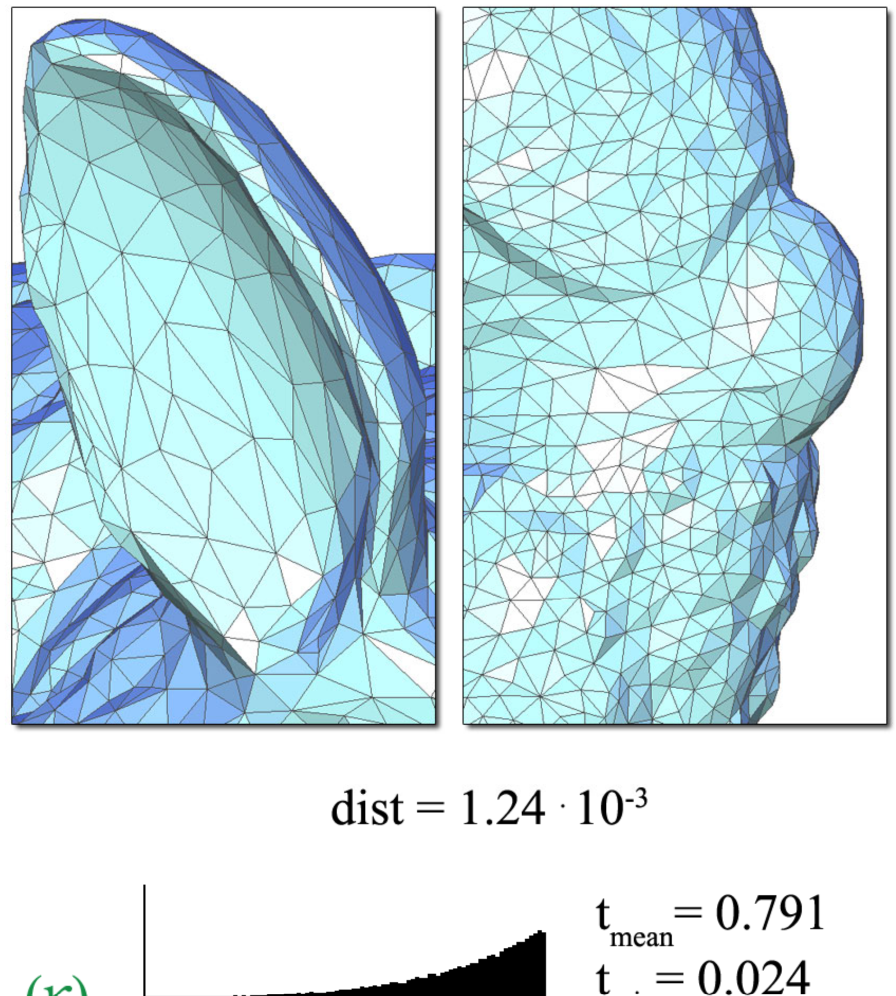
(a) original (17k)



Constant Weights

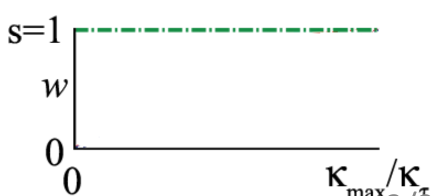


$t_{mean} = 0.680$
$t_{min} = 0.0001$



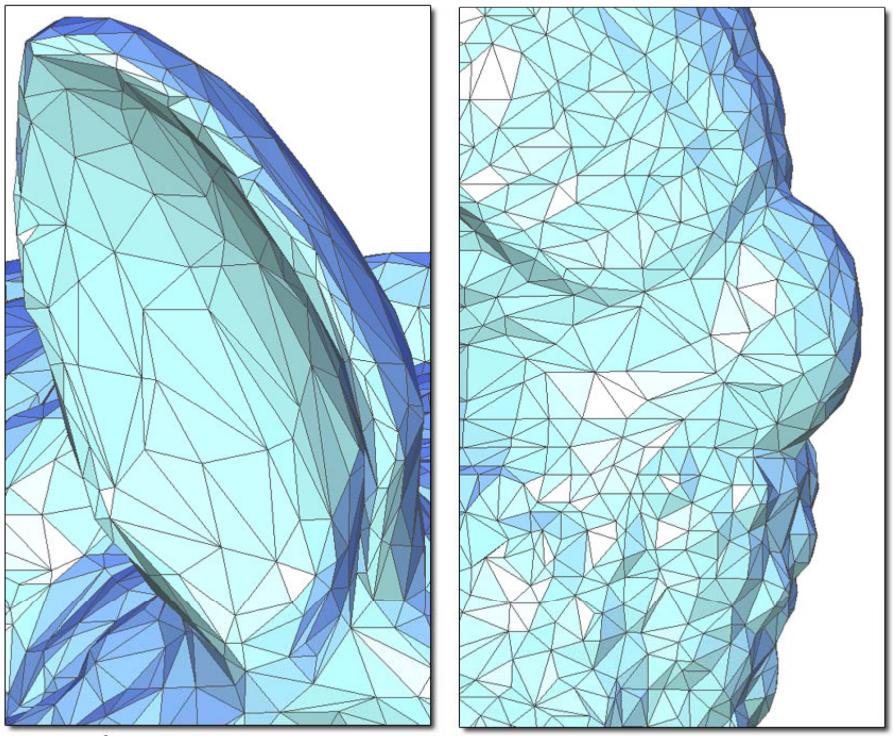
$dist = 1.24 \cdot 10^{-3}$

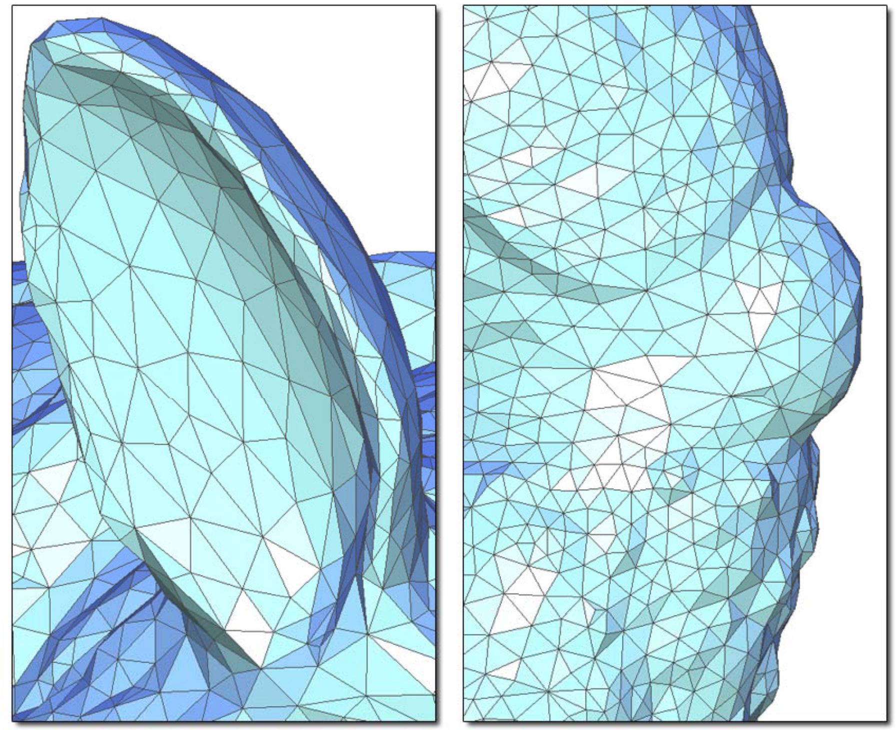
$t_{mean} = 0.791$
$t_{min} = 0.024$

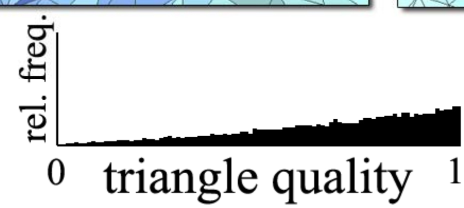


$w_{const}(\kappa)$

Linear Weights



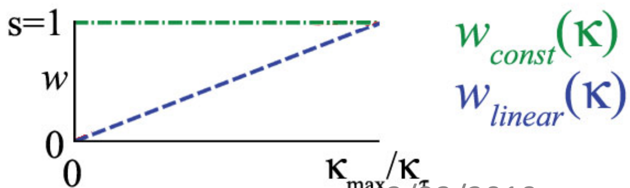




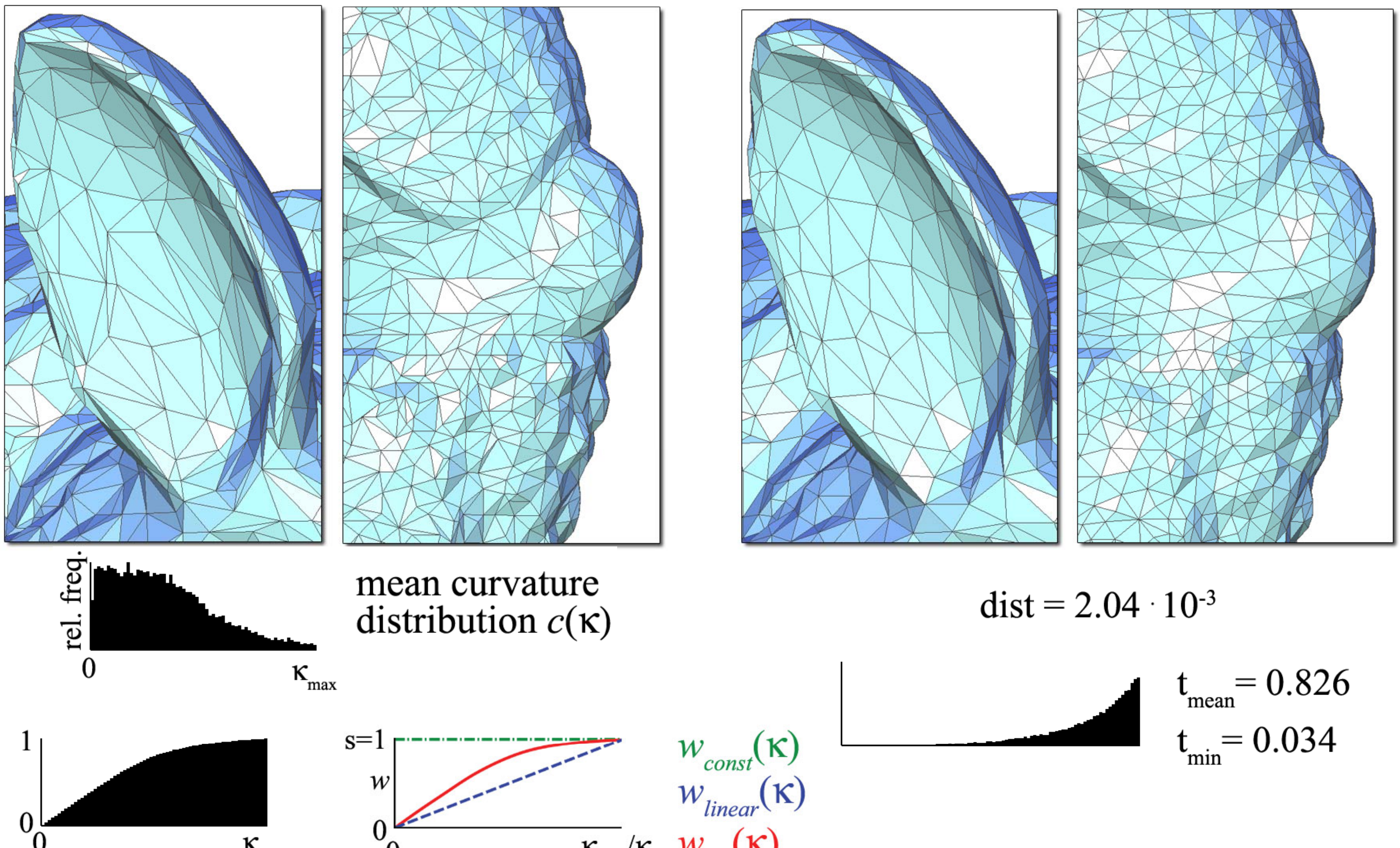
$t_{mean} = 0.680$
$t_{min} = 0.0001$

$dist = 2.53 \cdot 10^{-3}$

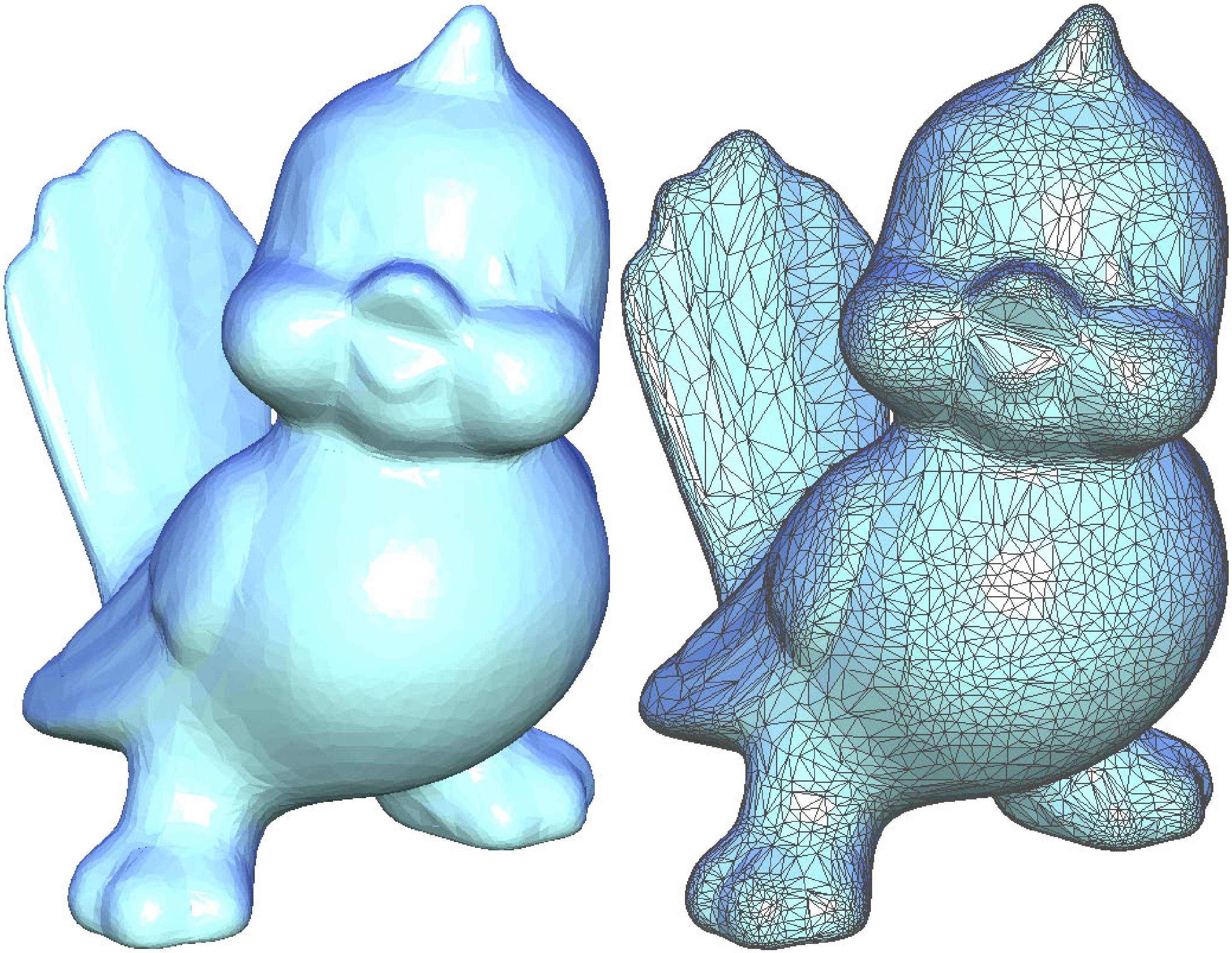
$t_{mean} = 0.842$
$t_{min} = 0.040$



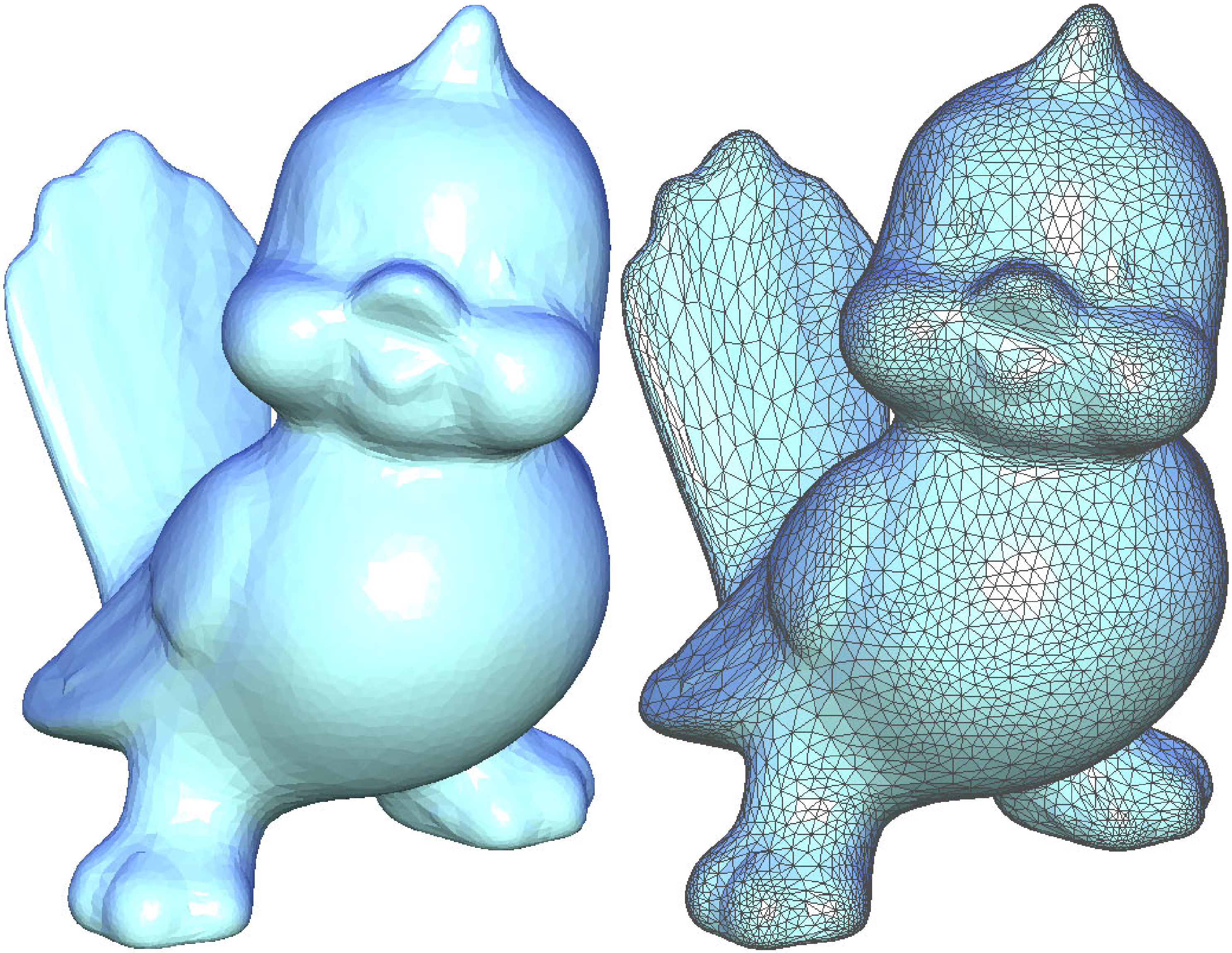
CDF Weights


rel. freq.
0
κmax
mean curvature
distribution c(κ)
dist = 2.04 · 10-3
tmean = 0.826
tmin = 0.034
1
0
0
κmax
s=1
w
0
0
κmax/κτ
wconst(κ)
wlinear(κ)
wcdf(κ)

Original



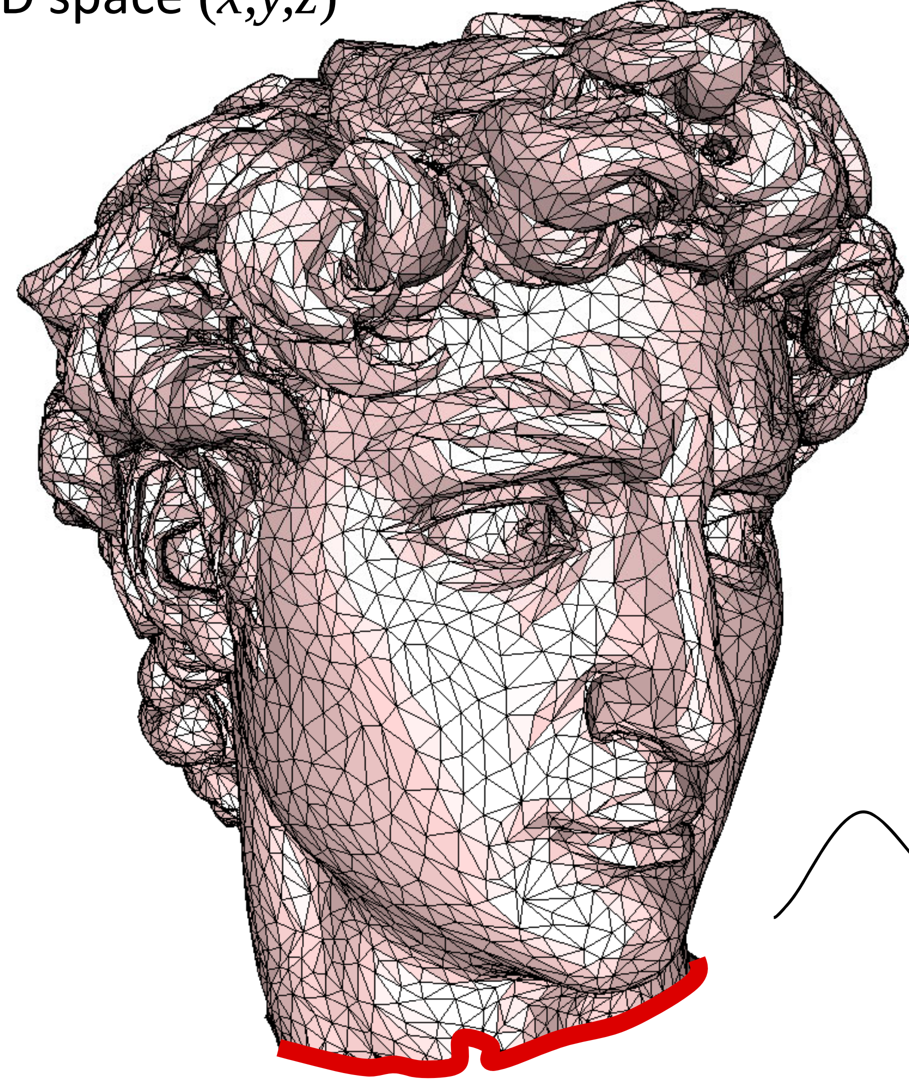
Tri Shape Optimization



Parameterization and Remeshing

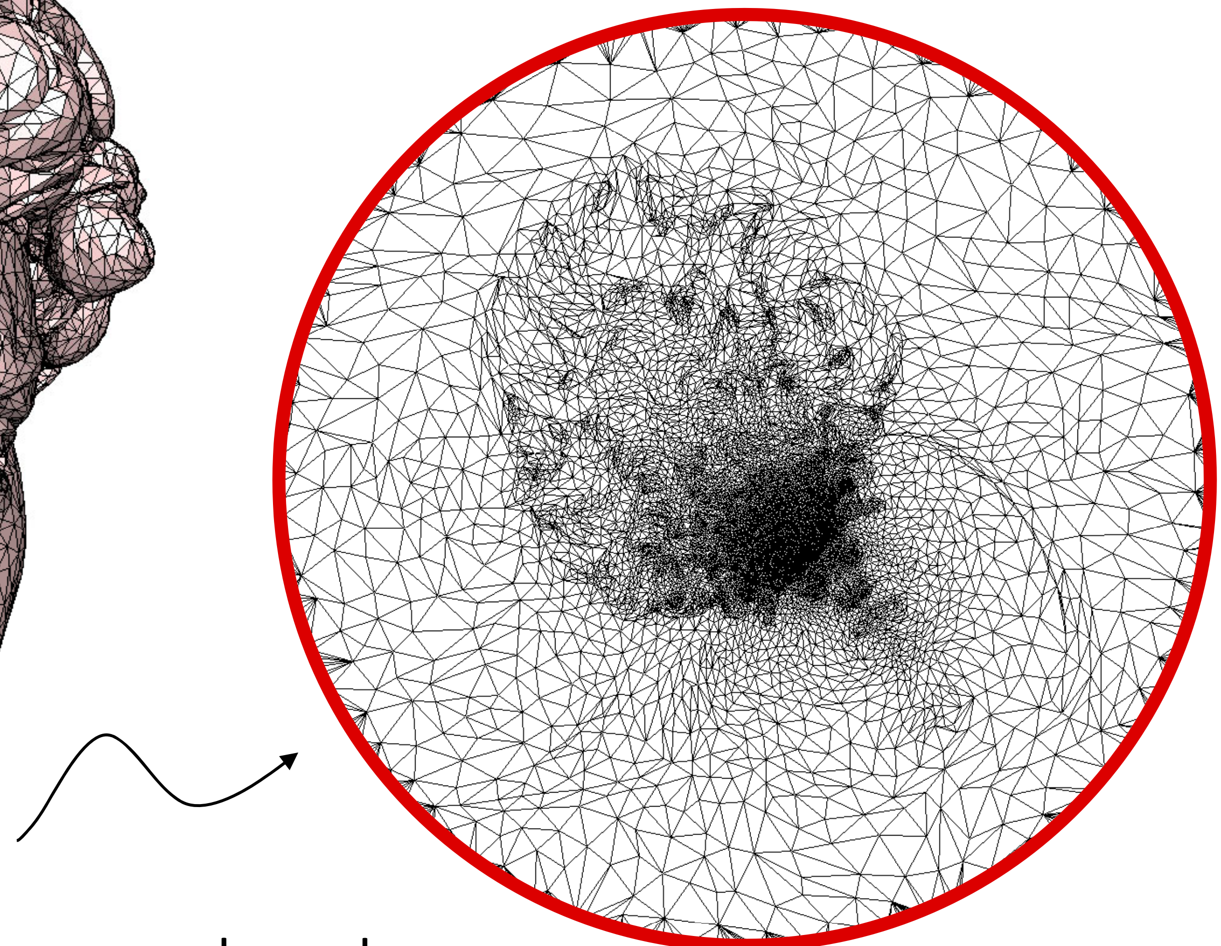
Surface parameterization

3D space (x,y,z)



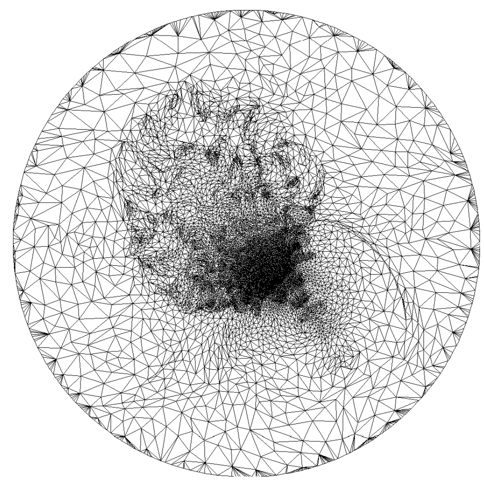
boundary

2D parameter domain (u,v)



boundary

Texture mapping



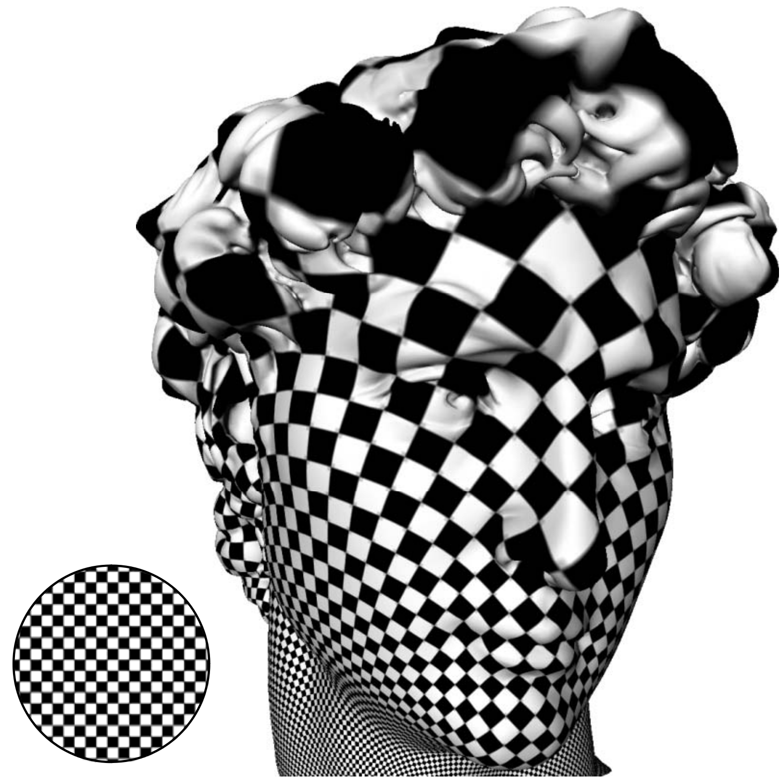
Texture mapping

Image from Vallet and Levy, techreport INRIA

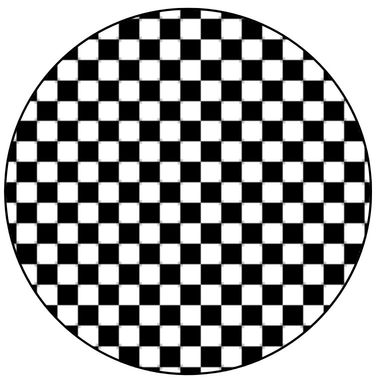
Mesh parameterization

Requirements

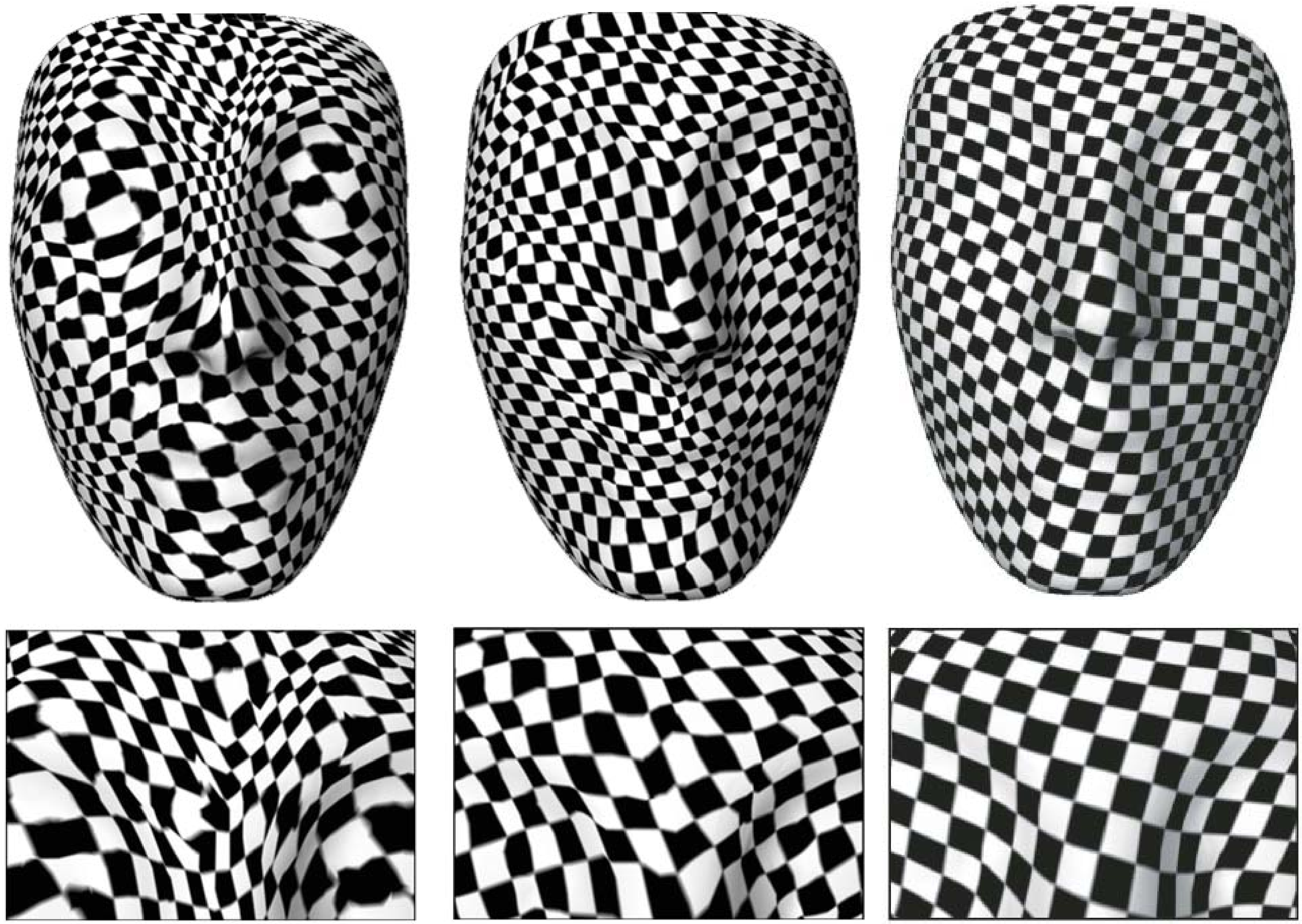
- Bijective (1-1 and onto): No triangles fold over.
- Minimal "distortion"
  - Preserve 3D angles
  - Preserve 3D distances
  - Preserve 3D areas
  - No "stretch"



Distortion minimization

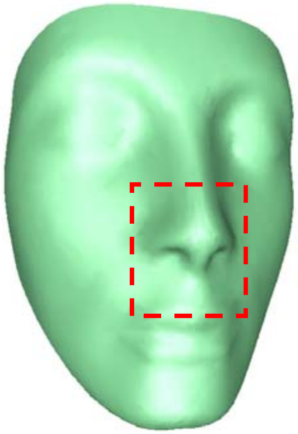


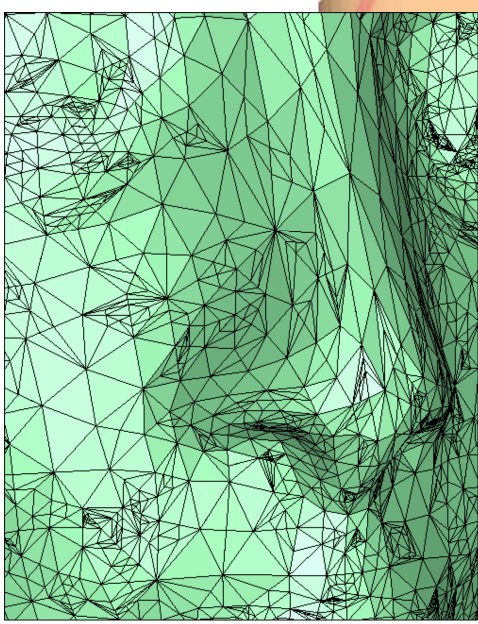
Texture map

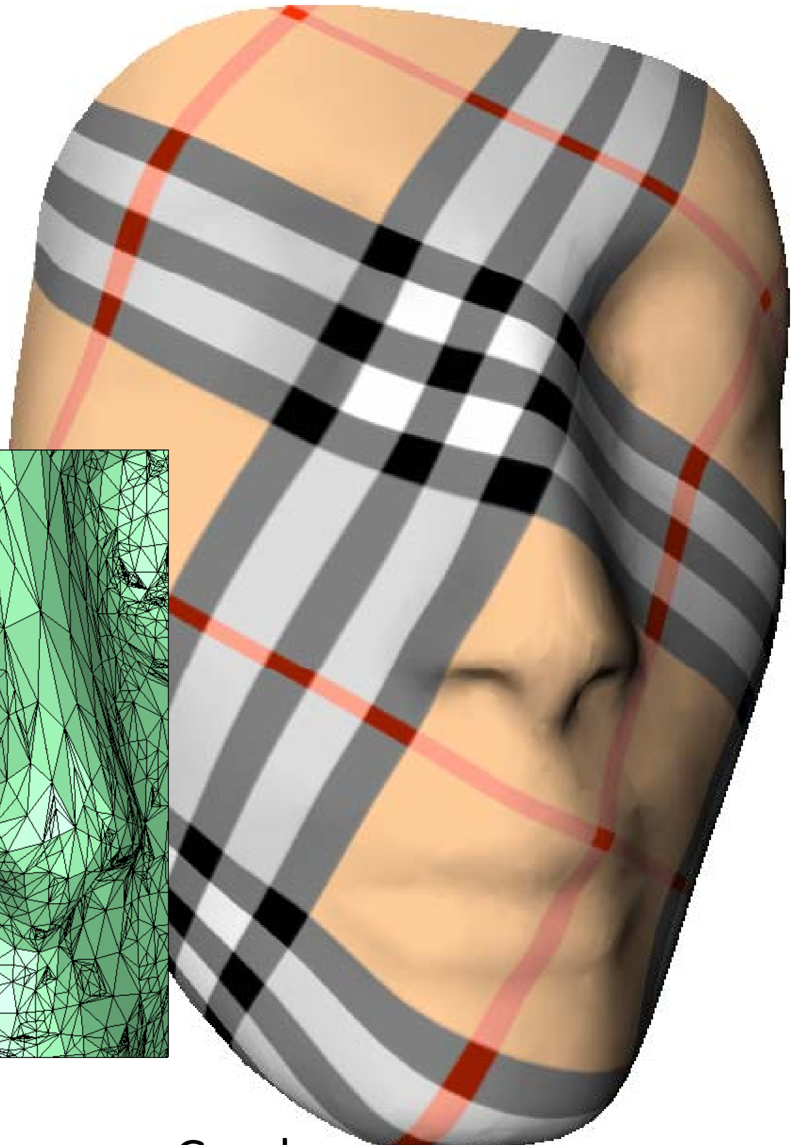


Kent et al '92 | Floater 97 | Sander et al '01

Sensitivity to mesh quality



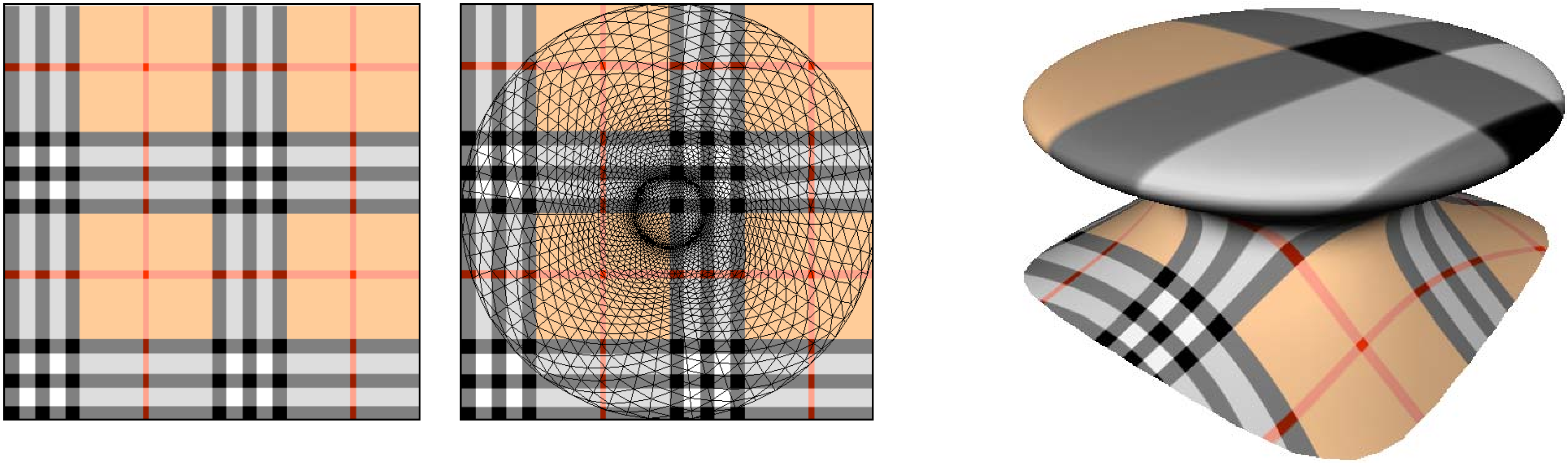




Good
parameterization algorithm

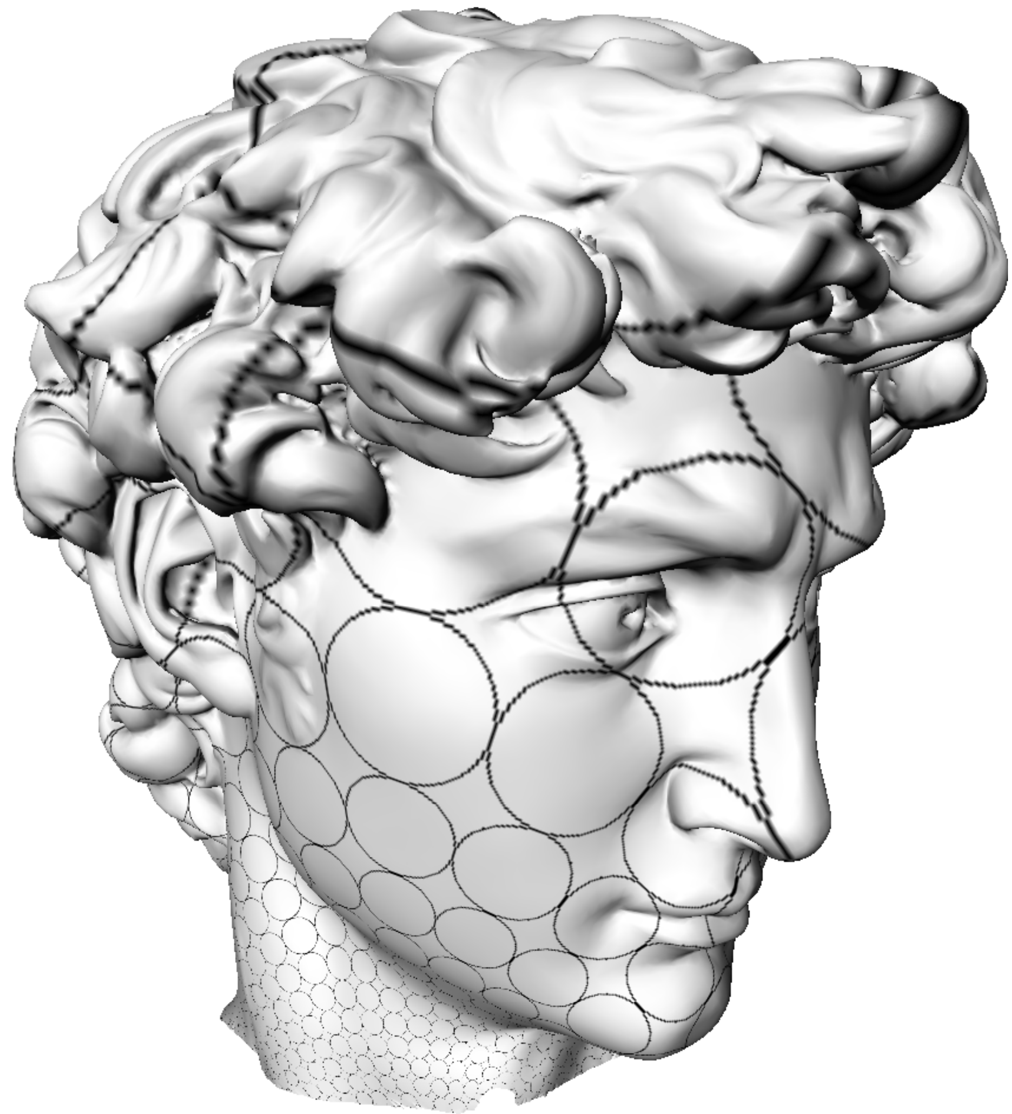
Not so good
parameterization algorithm

Area distortion vs. angle distortion

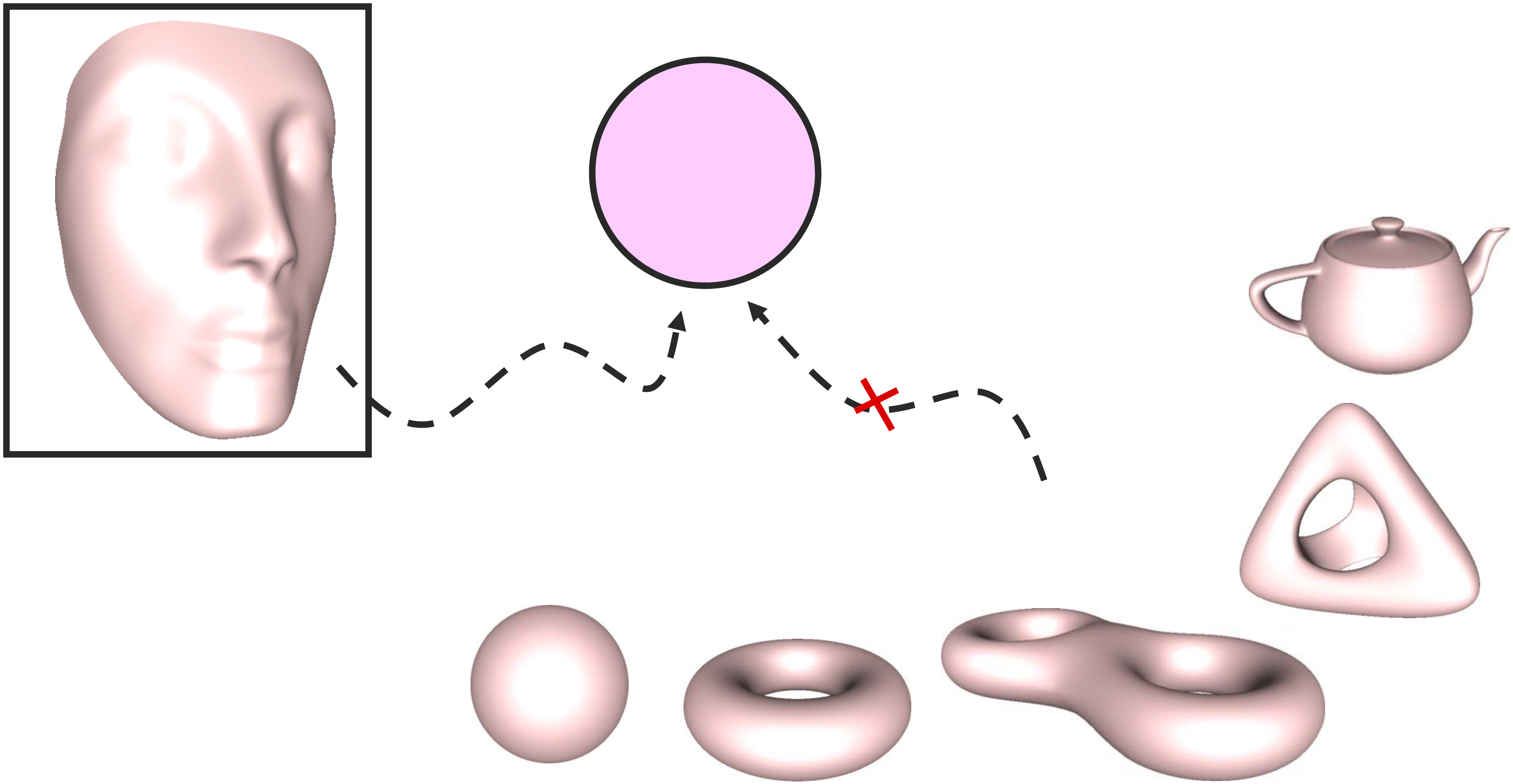


Conformal parameteriztion

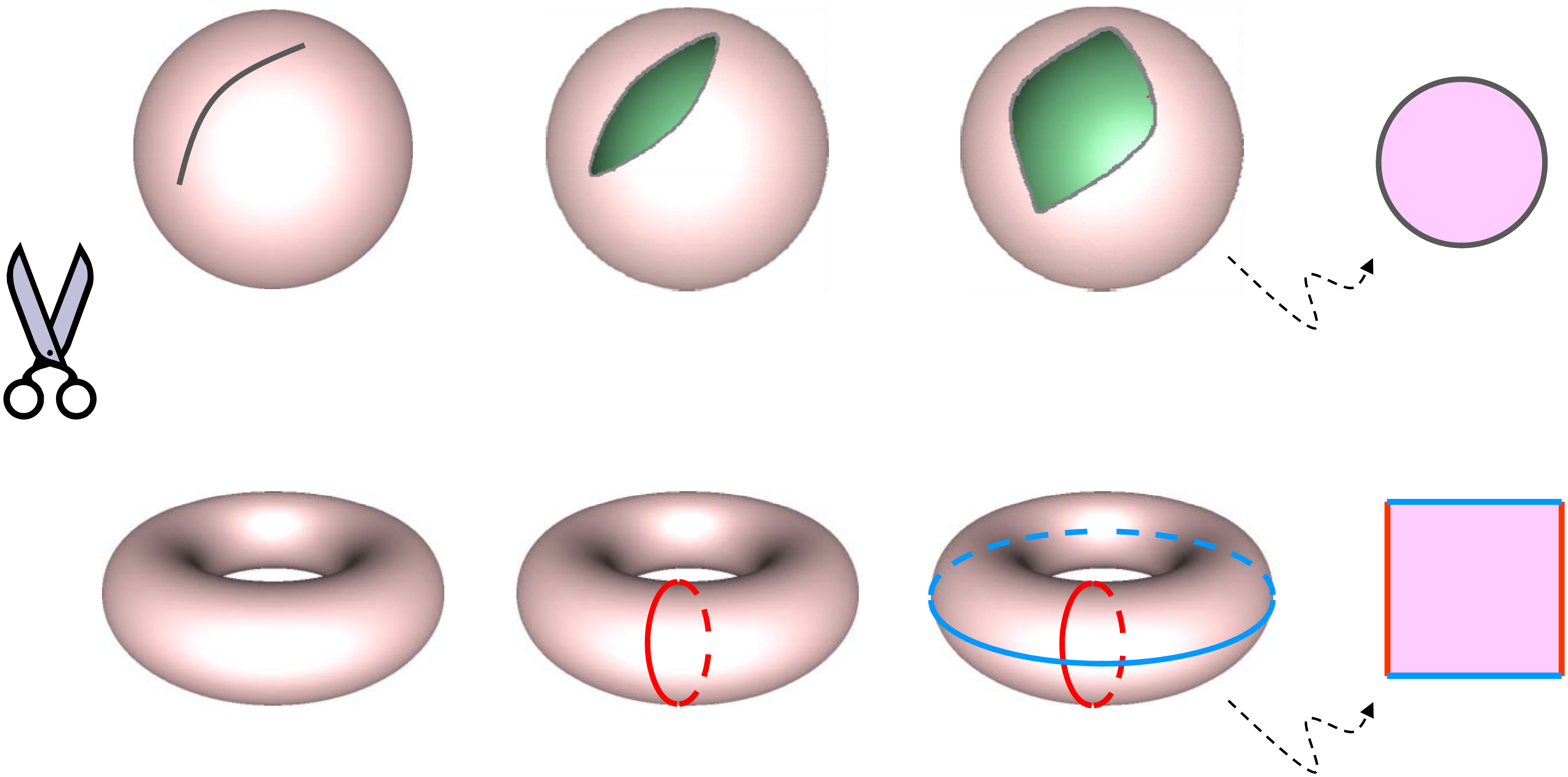
angle preservation; circles are mapped to circles



Non-disk domains



Cutting

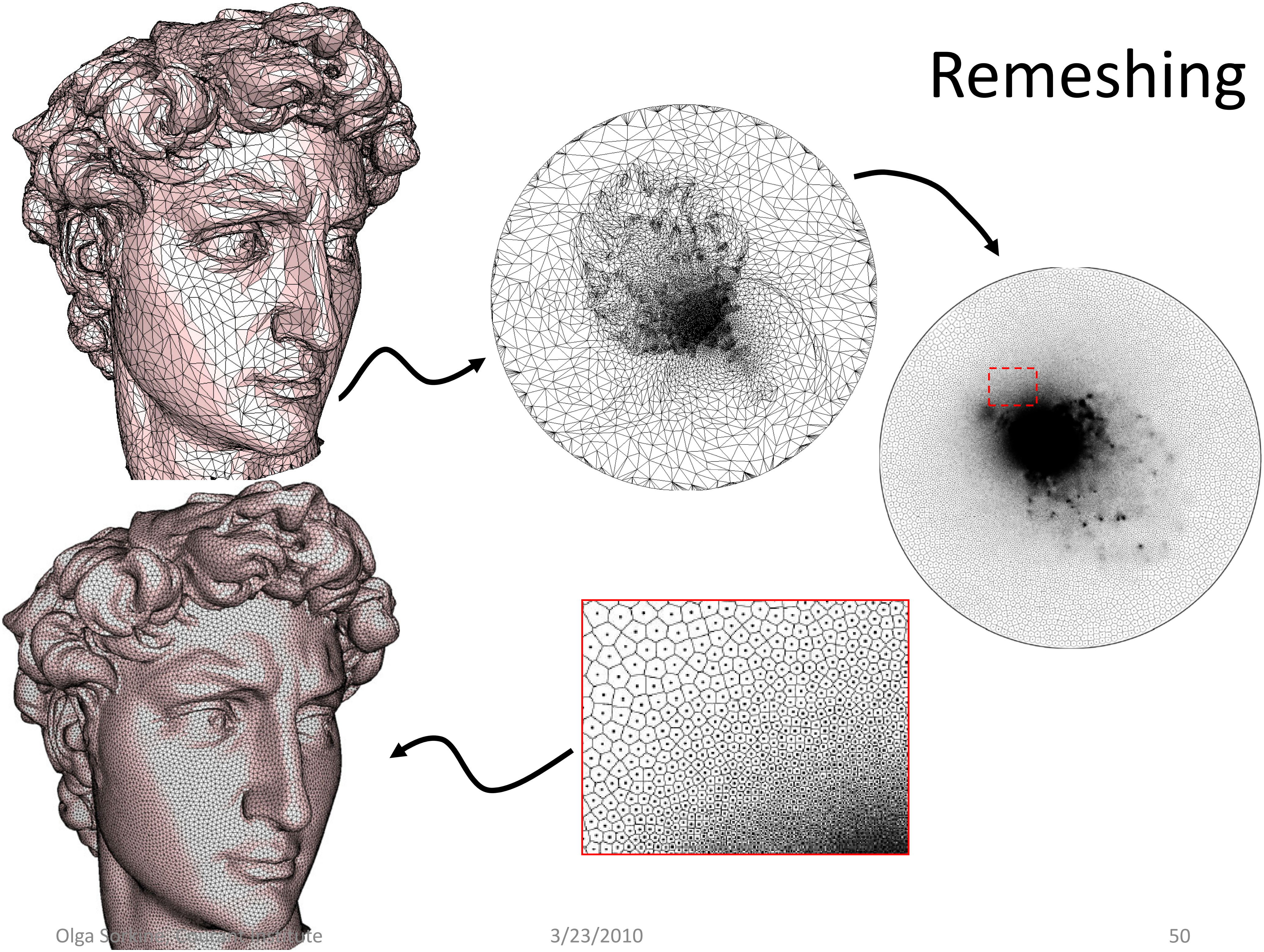


Why parameterization?

- Allows us to do many things in 2D and then map those actions onto the 3D surface
- It is often easier to operate in the 2D domain

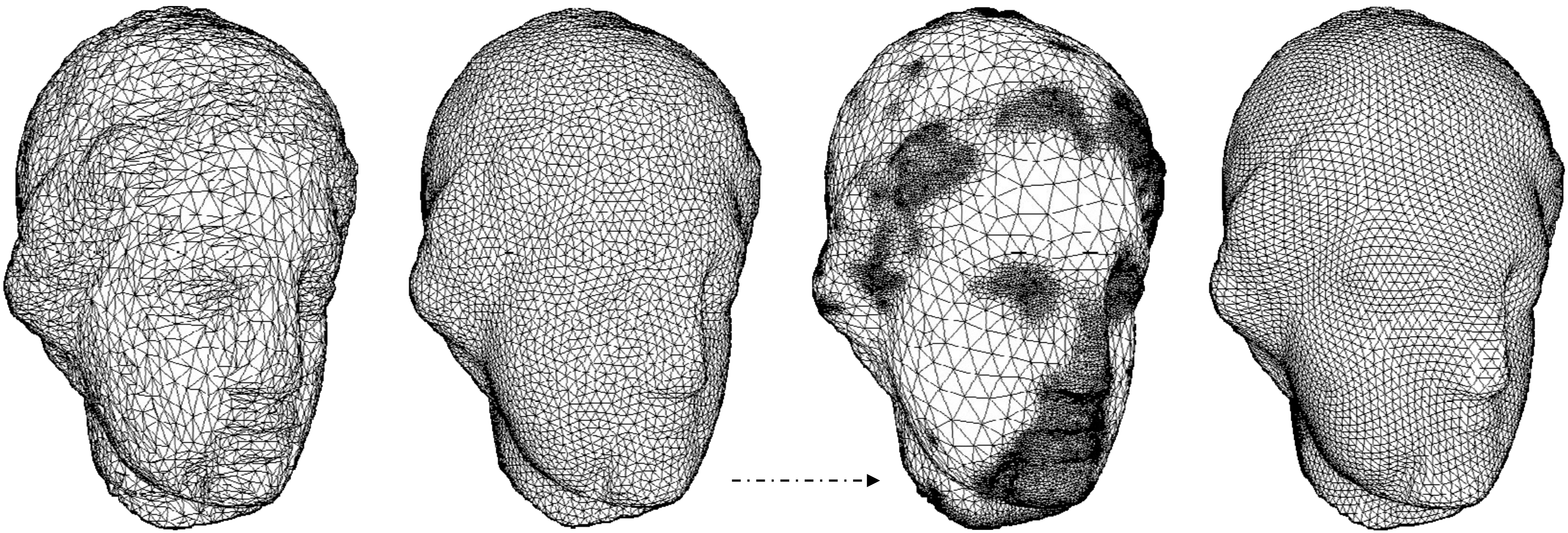
- Mesh parameterization allows to use some notions from continuous surface theory

Remeshing



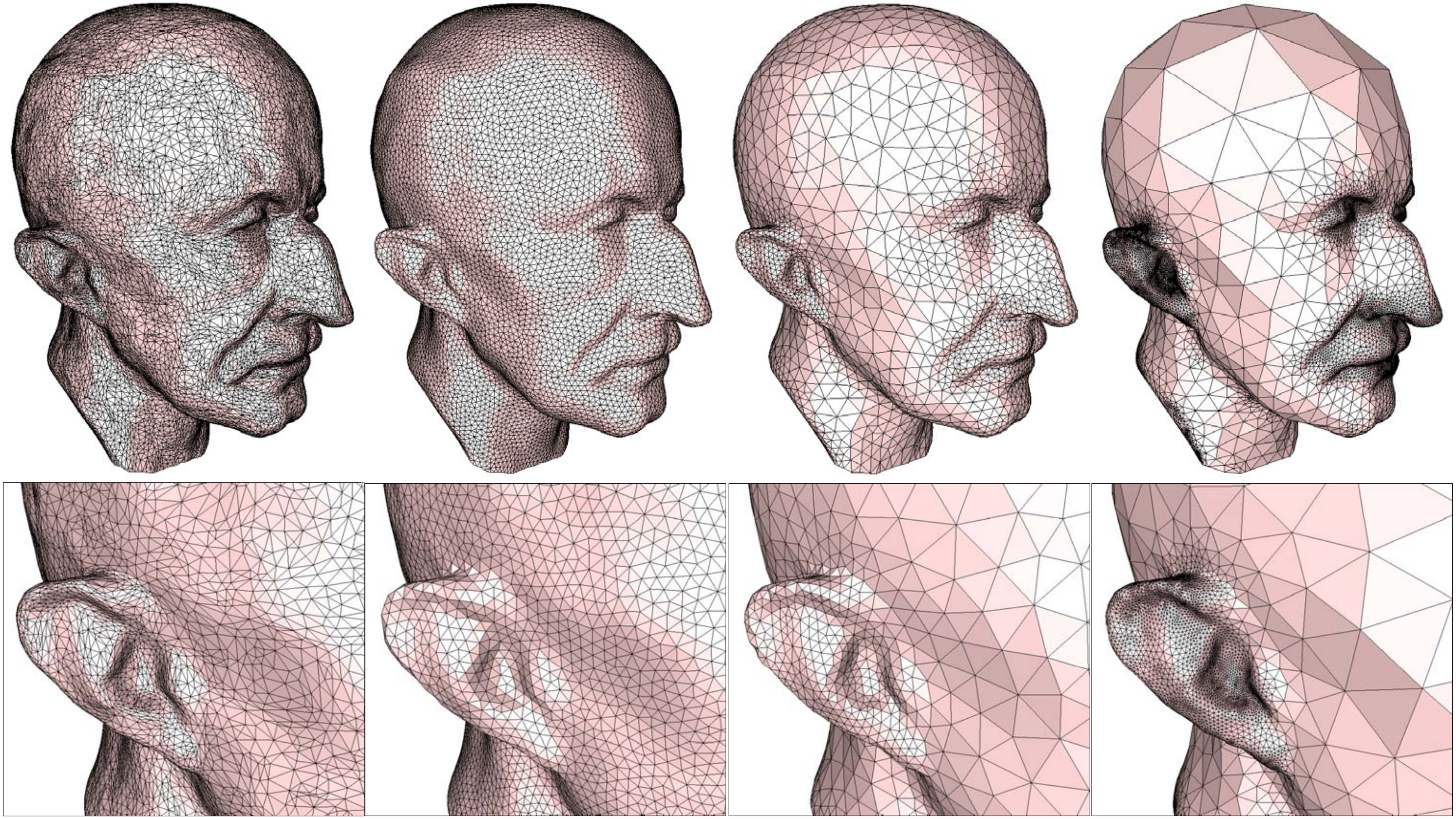
Remeshing

- Particular remeshing type according to application



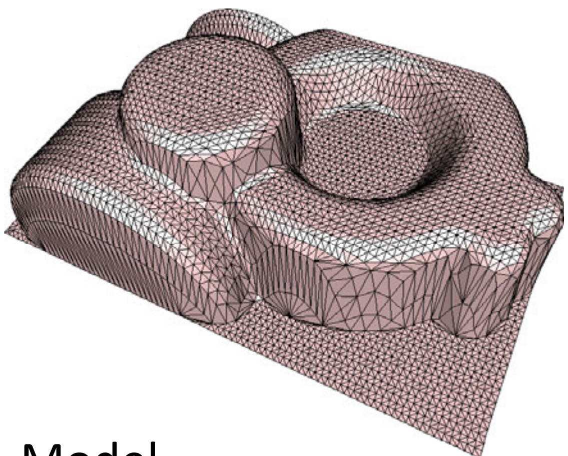
original uniform adapted semi-regular

Remeshing examples

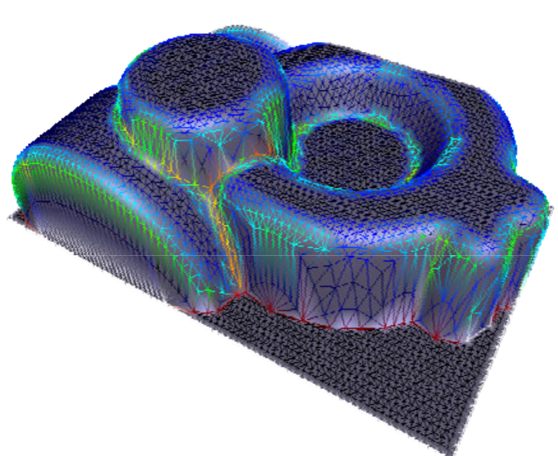


Interactive geometry remeshing

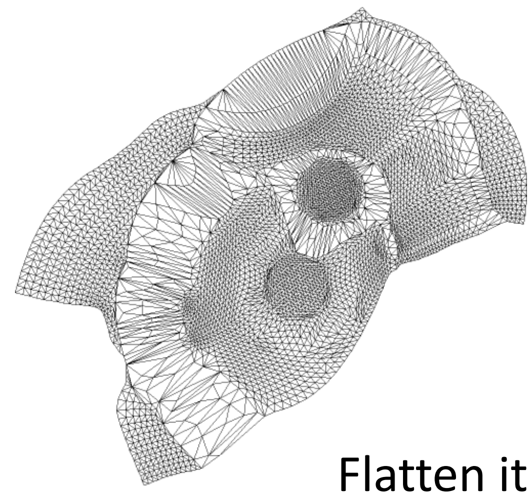
[Alliez et al., SIGGRAPH 2002]



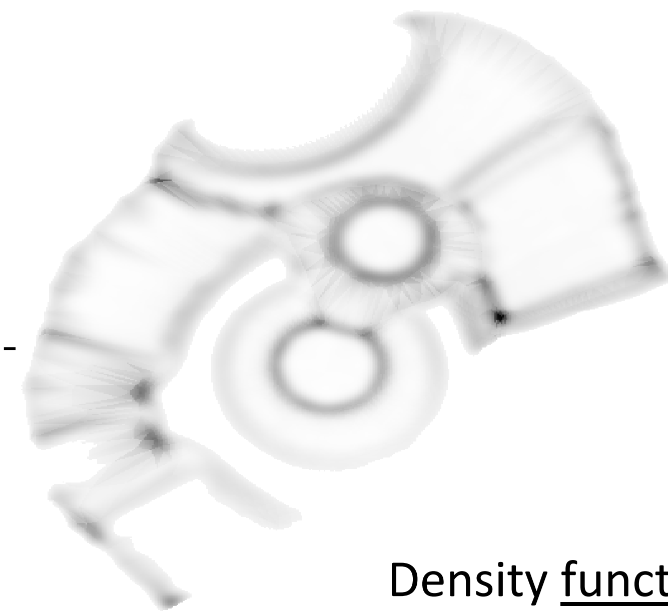
Model



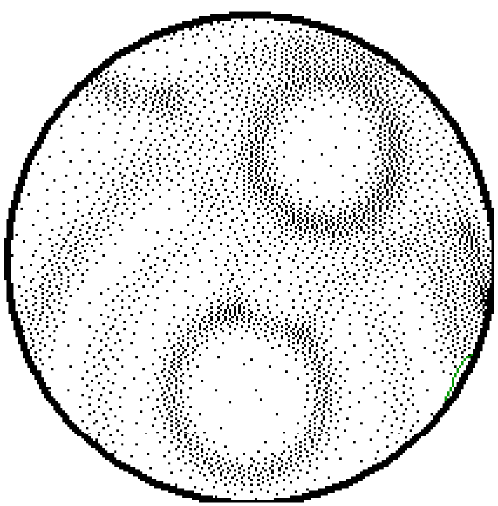
Measure curvature



Flatten it **conformally**



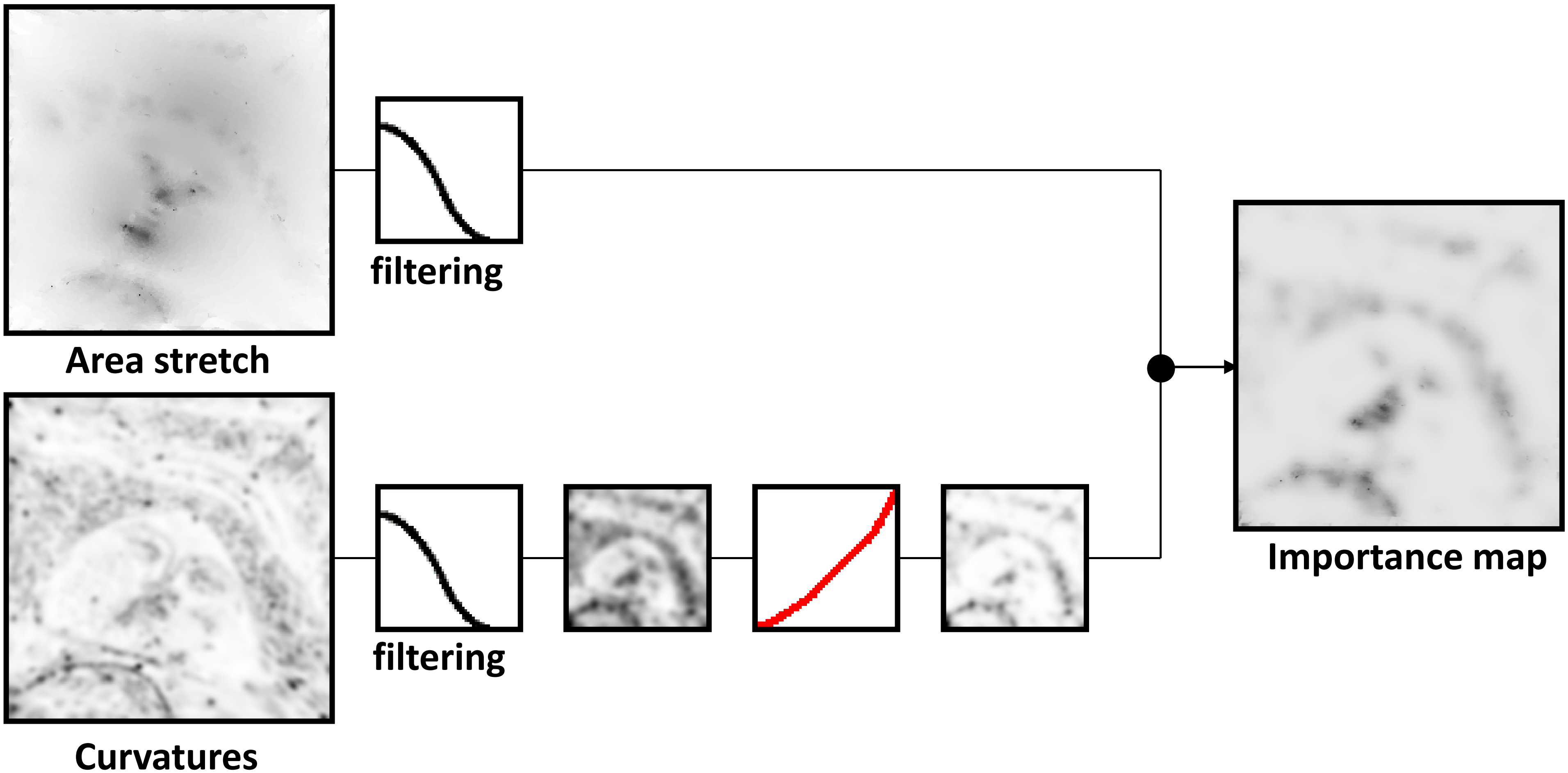
Density function in parameter space



Interactive geometry remeshing

[Alliez et al., SIGGRAPH 2002]

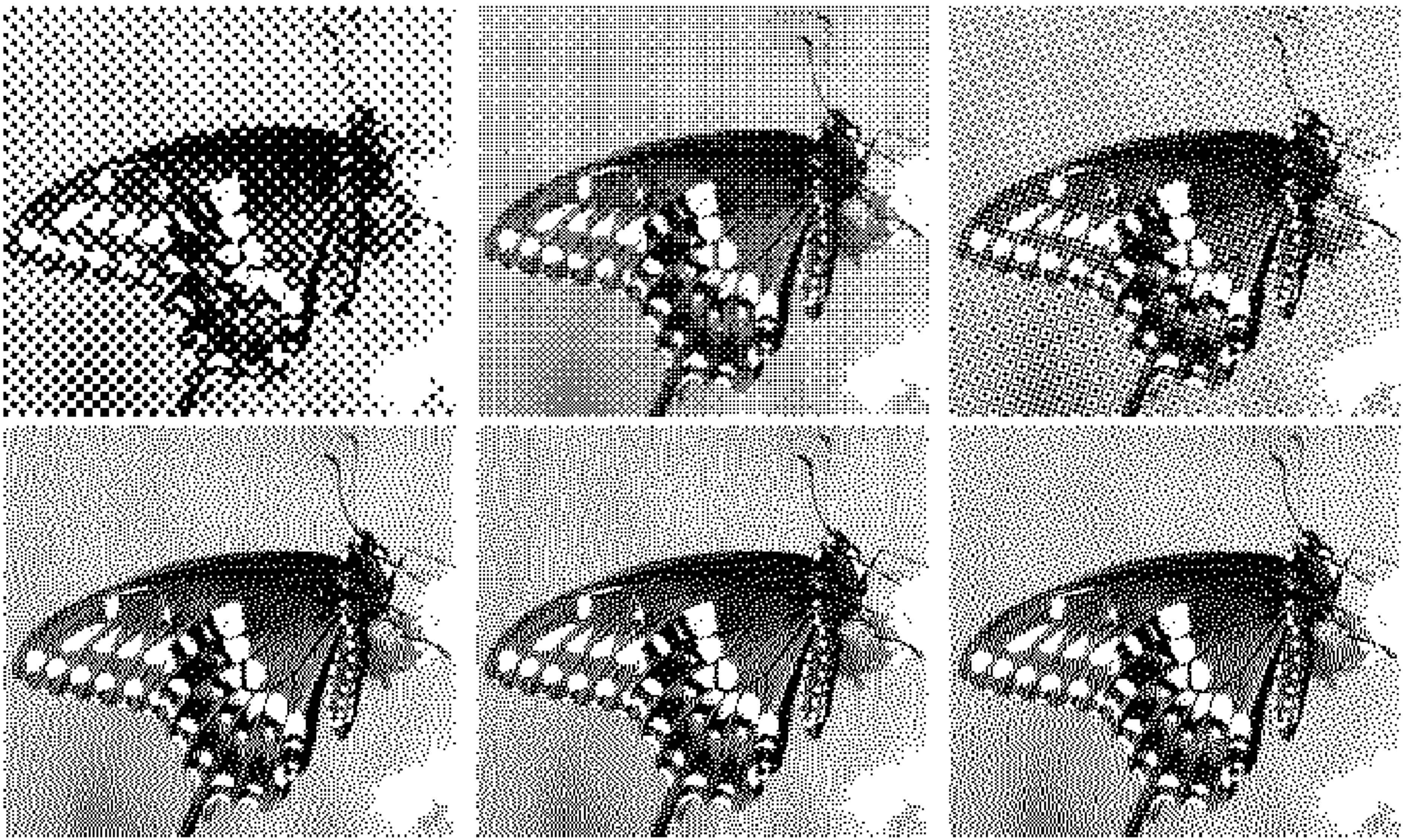
- Importance map created according to application needs



Interactive geometry remeshing

[Alliez et al., SIGGRAPH 2002]

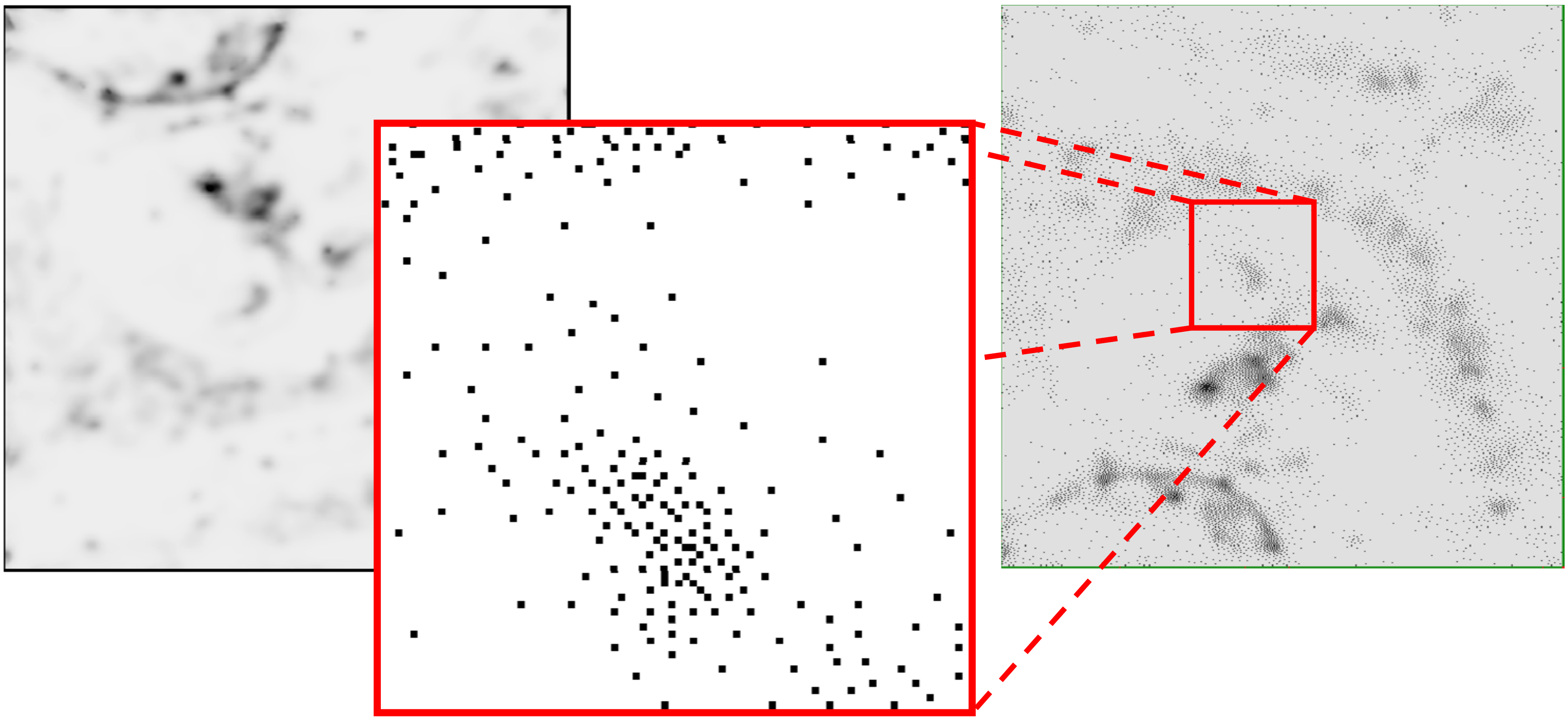
- Importance map is sampled by points – as in halftoning



Interactive geometry remeshing

[Alliez et al., SIGGRAPH 2002]

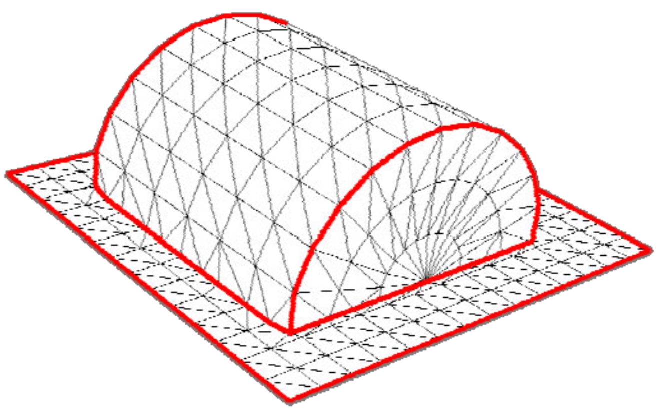
- Importance map is sampled by points – as in halftoning (error diffusion process)

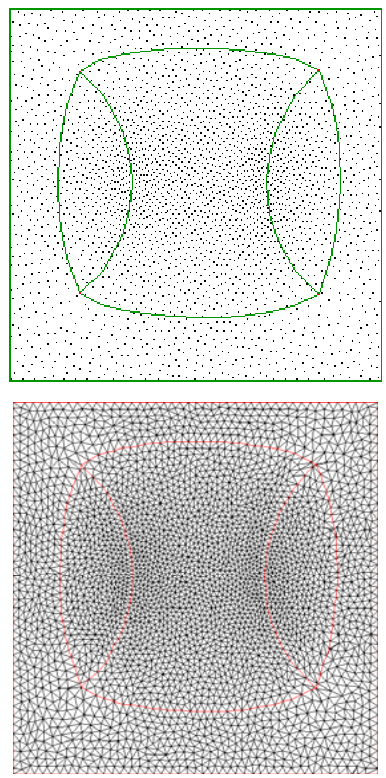


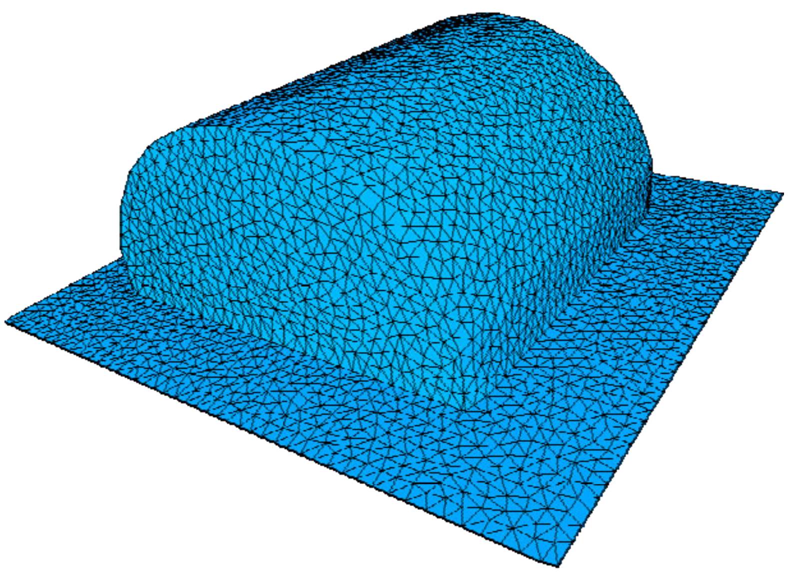
Interactive geometry remeshing

[Alliez et al., SIGGRAPH 2002]

- Sampled points are triangulated using Delaunay
- Using the parameterization, the 2D points are lifted back into 3D



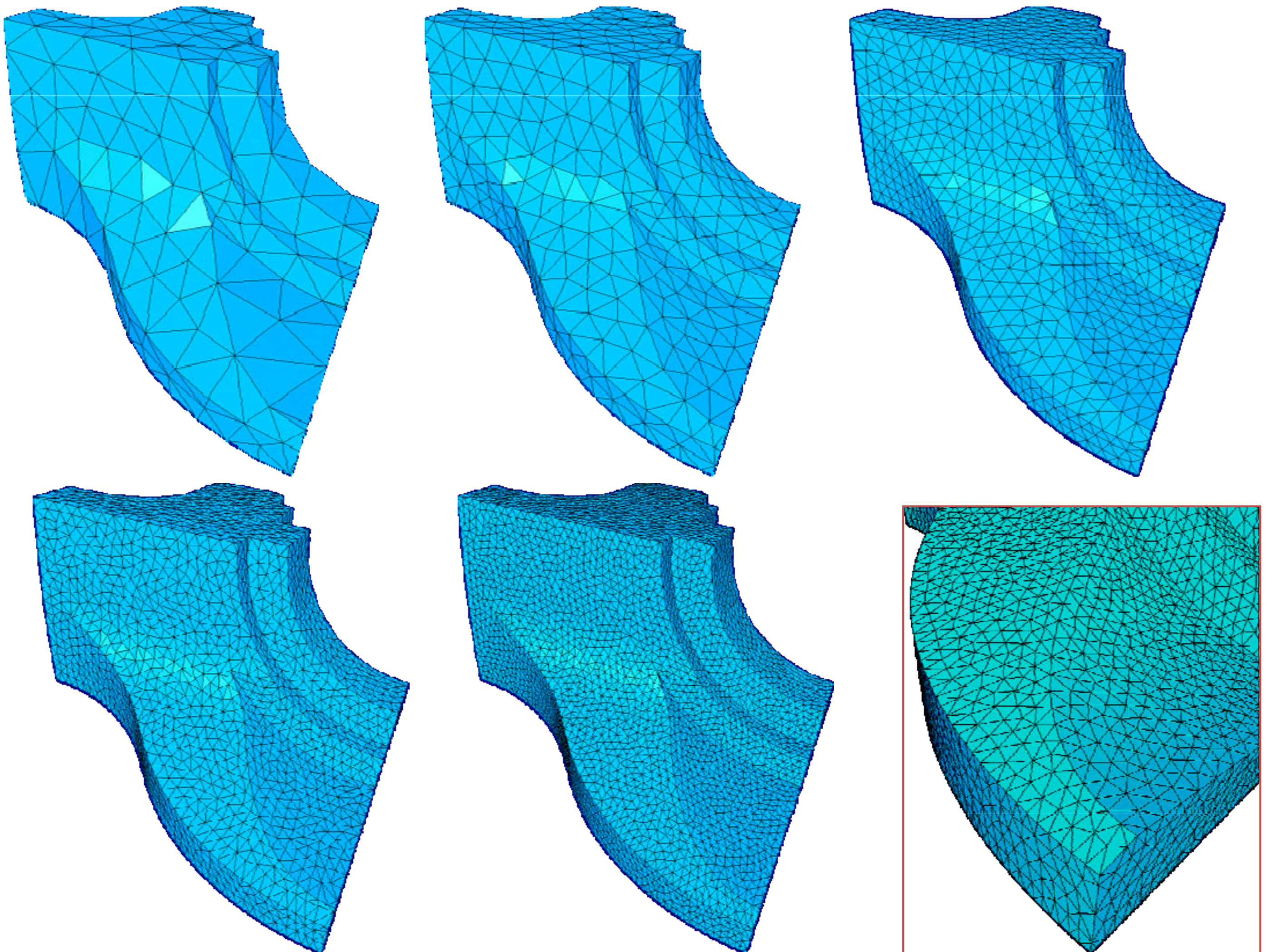




Interactive geometry remeshing

[Alliez et al., SIGGRAPH 2002]

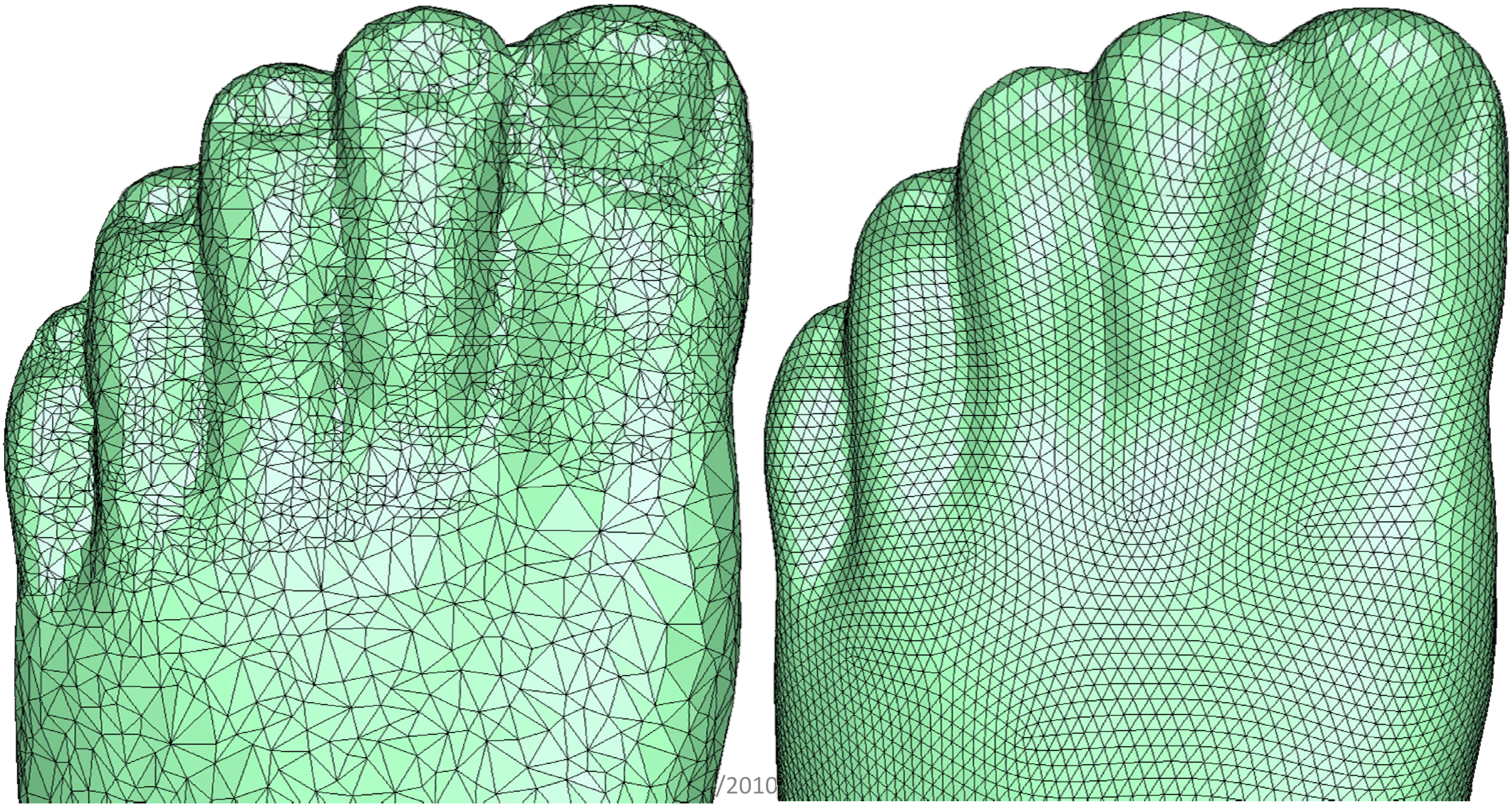
- More results



Interactive geometry remeshing

[Alliez et al., SIGGRAPH 2002]

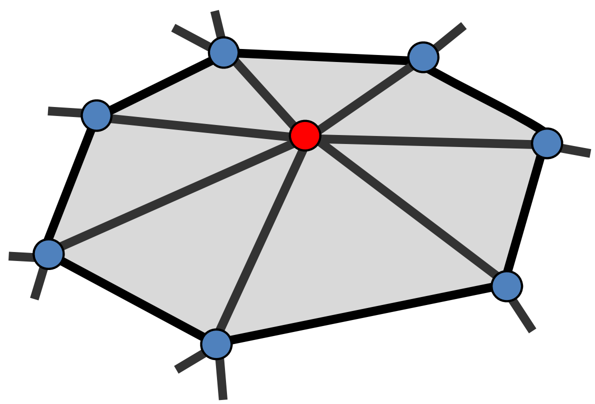
- More results



Computing parameterizations

General idea

- Want to flatten the mesh $\rightarrow$ no curvature $\rightarrow$ Laplace operator gives zero.



$\mathbf{v} = (u,v)$ domain

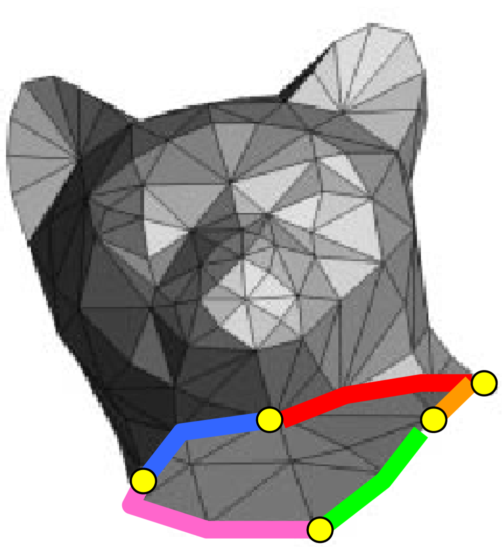
$$L\mathbf{v} = 0$$

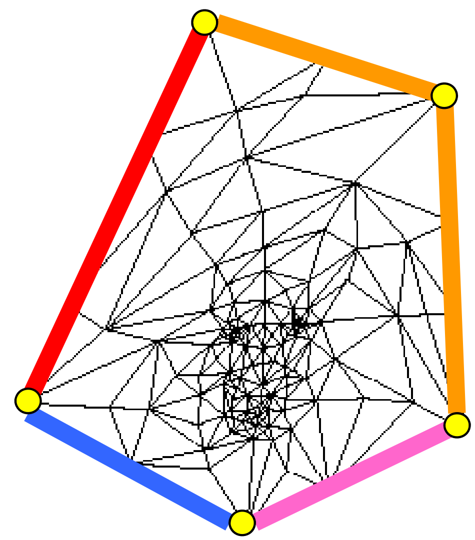
need boundary constraints to prevent trivial solution;

which Laplacian operator (which weights?)

Convex mapping (Tutte, Floater)

- Boundary vertices are fixed
- Convex weights





$\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ – inner vertices; $\mathbf{v}_n, \mathbf{v}_{n+1}, \ldots, \mathbf{v}_N$ – boundary vertices

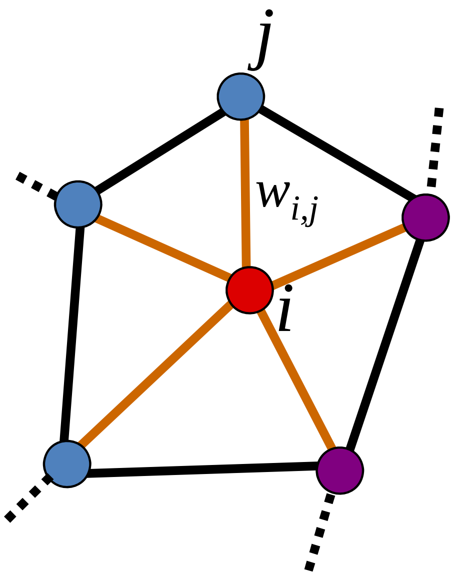
Convex mapping (Tutte, Floater)

- Boundary vertices are fixed
- Convex weights

$$L(\mathbf{v}_i) = \sum_{\mathbf{v}_j \in N_1(\mathbf{v}_i)} w_{ij}\left(\mathbf{v}_j - \mathbf{v}_i\right) = 0$$

$$w_{ij} > 0$$

$$\sum_{\mathbf{v}_j \in N_1(\mathbf{v}_i)} w_{ij} = 1$$



$\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ – inner vertices; $\mathbf{v}_n, \mathbf{v}_{n+1}, \ldots, \mathbf{v}_N$ – boundary vertices

Convex mapping (Tutte, Floater)

- Solve the linear system

$$L\mathbf{v} = 0$$

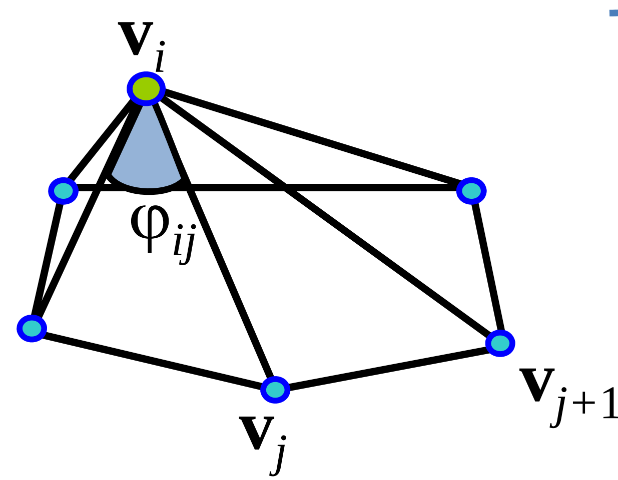
- Remember to substitute the fixed boundary vertices
- Property: if boundary is **convex** and weights are **convex**, the flattening is **guaranteed** to be **legal** (no fold-overs)

Convex weights

- Tutte (1966): uniform weights $w_{ij} = \frac{1}{d_i}$
  - Only depend on connectivity
  - High distortion
- Floater (1997): shape-preserving weights
- Floater (2003): mean-value weights

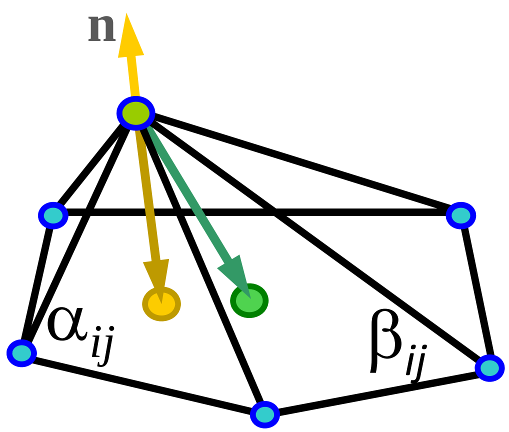
weights taken from 3D mesh

$$w_{ij} = \tan\frac{\varphi_{i,j}}{2} + \tan\frac{\varphi_{i,j+1}}{2}$$



Conformal parameterization

- Preserve angles as much as possible
- "Project" each vertex along its normal direction



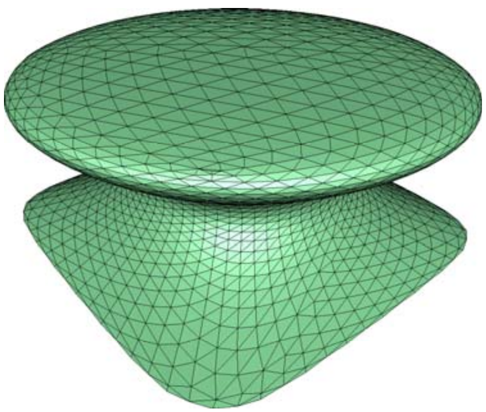
$$L_{cot}\mathbf{v} = 0$$

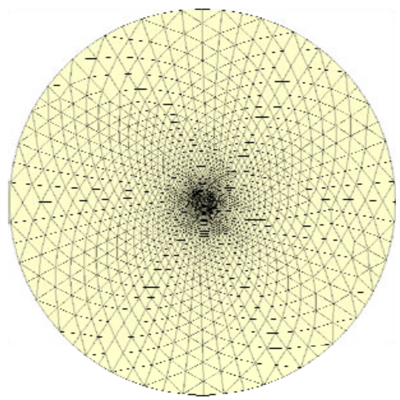
$$w_{ij} = \cot \alpha_{ij} + \cot \beta_{ij}$$

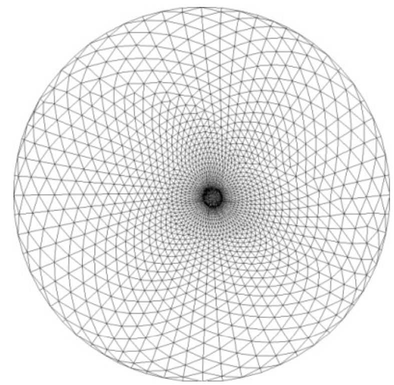
$\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ – inner vertices; $\mathbf{v}_n, \mathbf{v}_{n+1}, \ldots, \mathbf{v}_N$ – fixed boundary vertices

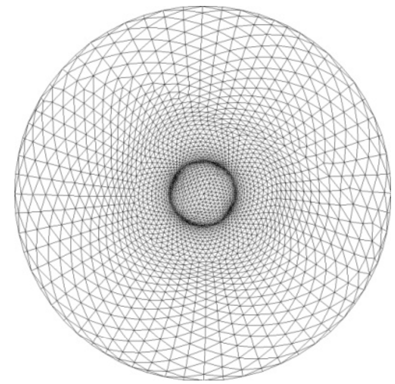
Comparison

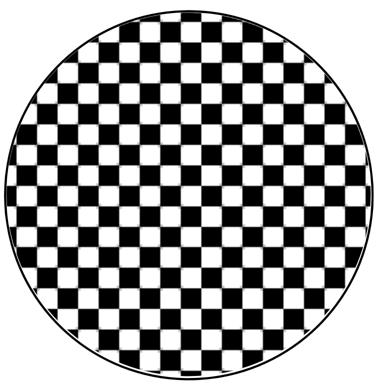
Tutte | Shape-preserving | Conformal







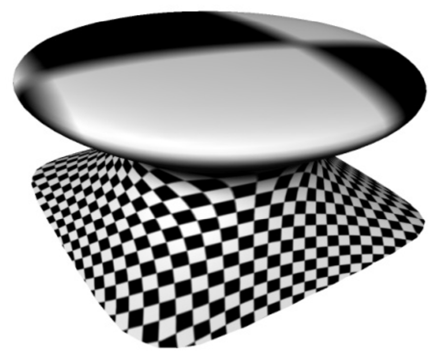


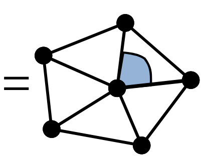


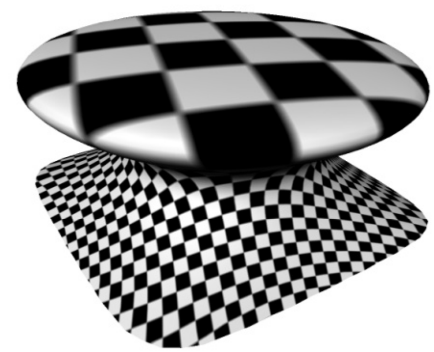
Texture map

$w_{ij} = 1/d_i$

depends on connectivity only



$w_{ij} =$ 



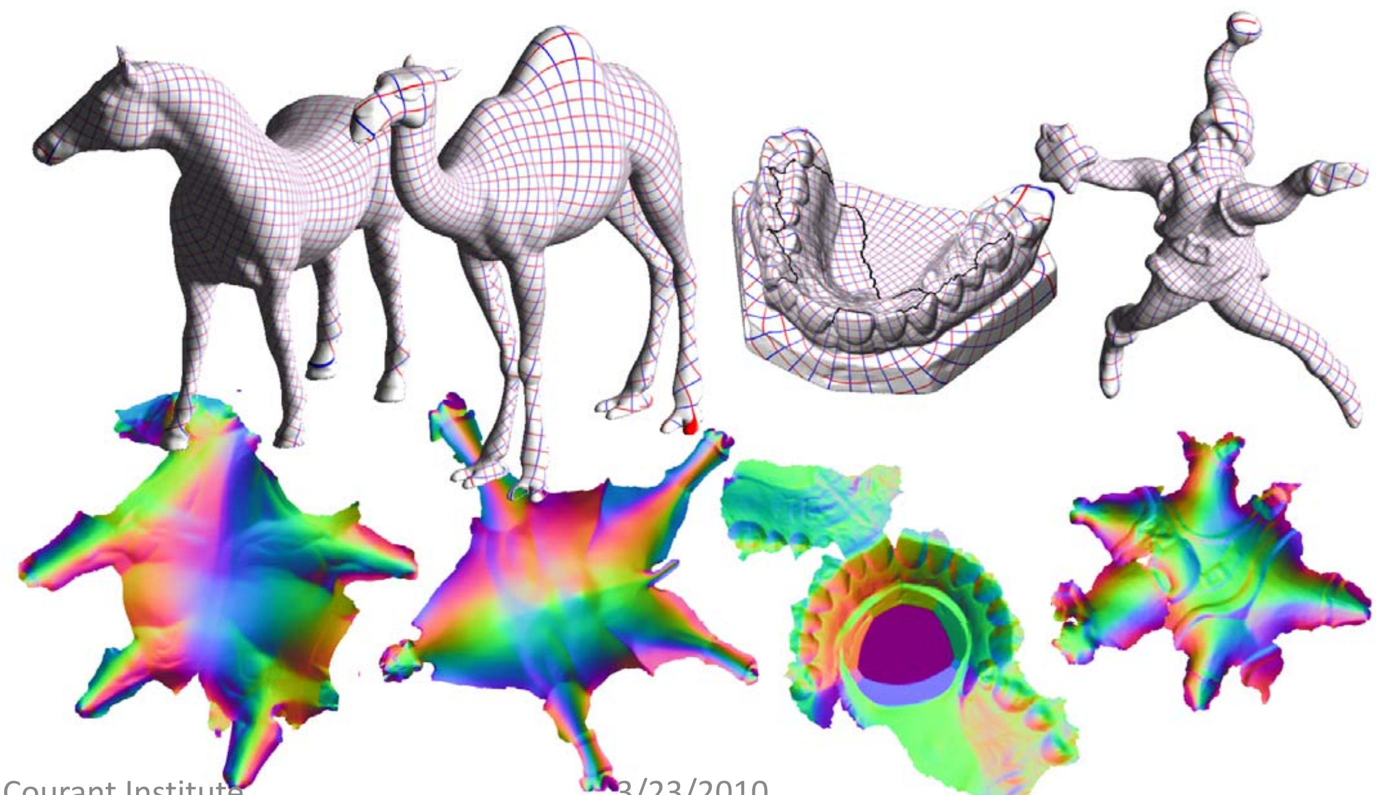
$w_{ij} = \cot \alpha_{ij} + \cot \beta_{ij}$

Discussion

- The results of harmonic mapping are better than those of convex mapping (local area and angles preservation).
- But: the mapping is not always legal (the weights can be negative for badly-shaped triangles…)

Discussion

- Both mappings have the problem of fixed boundary – it constrains the minimization and causes distortion.
- More advanced methods do not require boundary conditions (see references on the website).



ABF++ method, Sheffer et al.