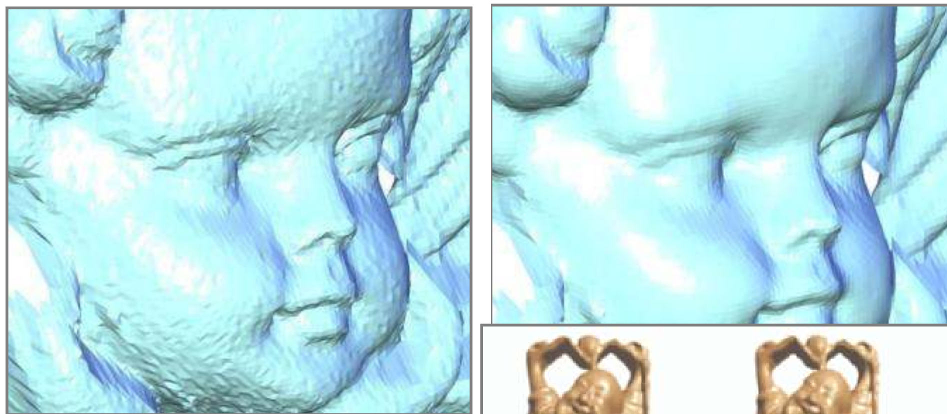
CS 523: Computer Graphics, Spring 2011

Shape Modeling

Digital Geometry Processing

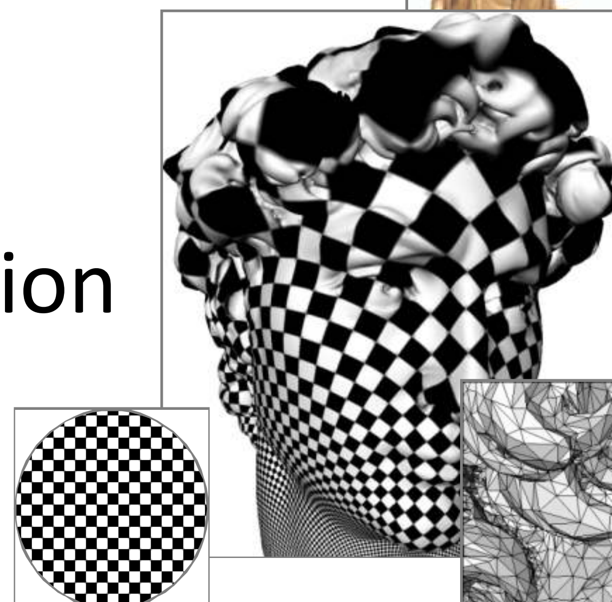
Topics

- Smoothing

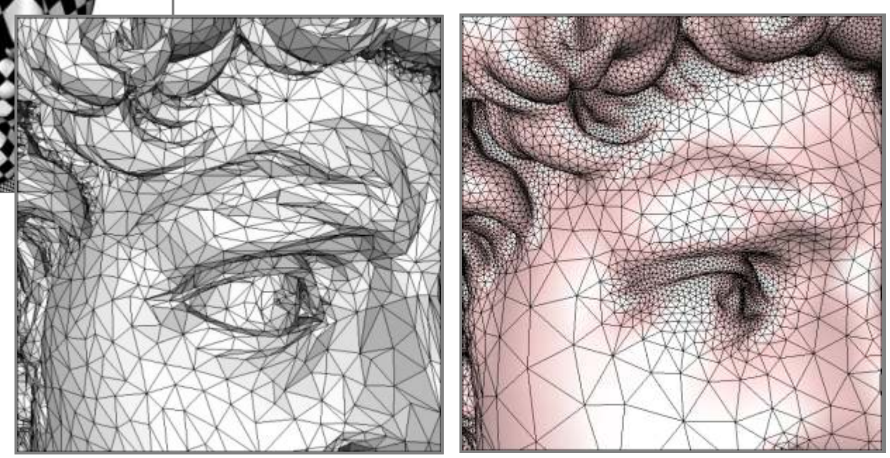


- Simplification

- Parameterization



- Remeshing



Mesh Smoothing

Curve smoothing

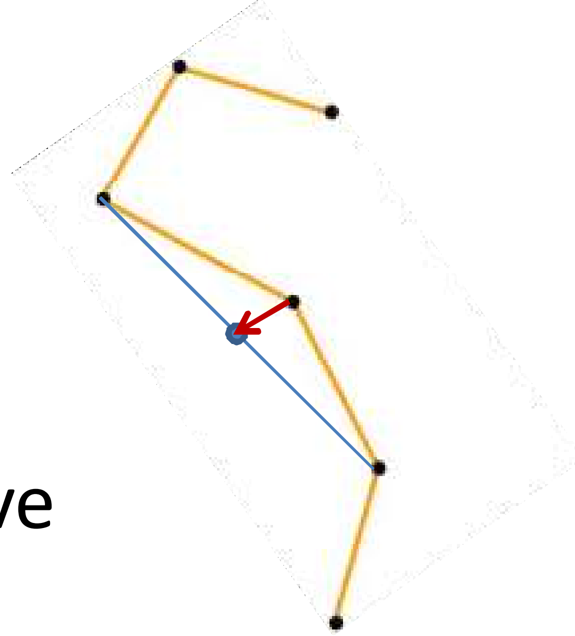
Taubin smoothing

Implicit fairing

Laplacian mesh optimization

Laplacian smoothing

2D Curve



- Discrete Laplacian for a single vertex

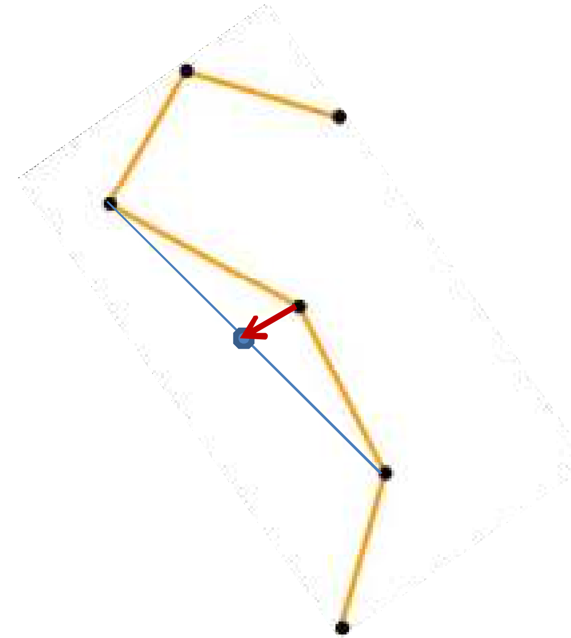
$$\Delta \mathbf{x}_i = \frac{1}{2}\left(\mathbf{x}_{i\text{-}1} - \mathbf{x}_i\right) + \frac{1}{2}\left(\mathbf{x}_{i+1} - \mathbf{x}_i\right)$$

- In matrix-vector form for the whole curve

$$\Delta \mathbf{x} = -\mathrm{K}\mathbf{x}$$

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

Smoothing



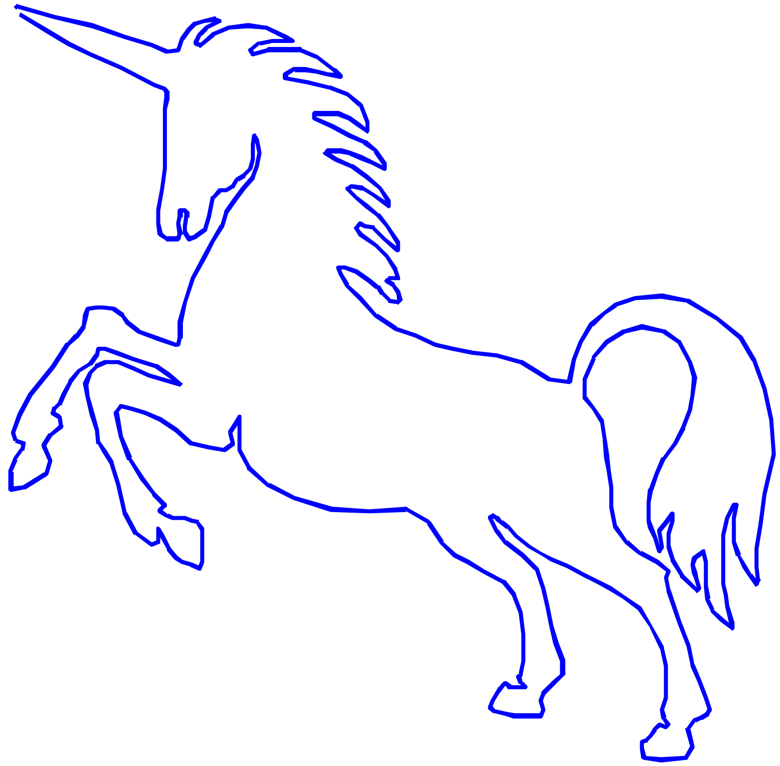
- Gaussian filtering

$$\mathbf{x}_i' = \mathbf{x}_i + \lambda \Delta \mathbf{x}_i$$

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

Laplacian smoothing

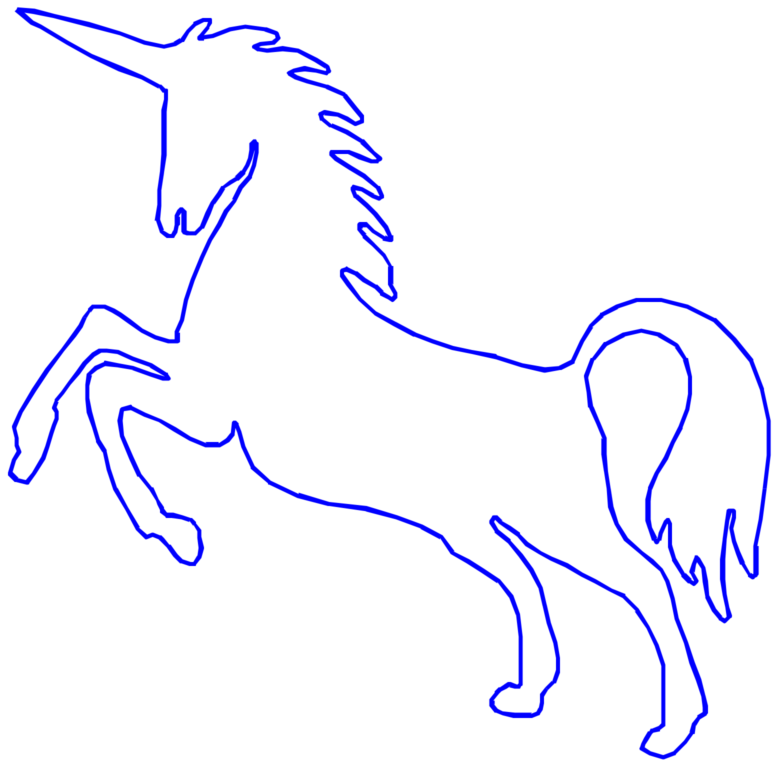
2D Curve – Example



Original curve

Laplacian smoothing

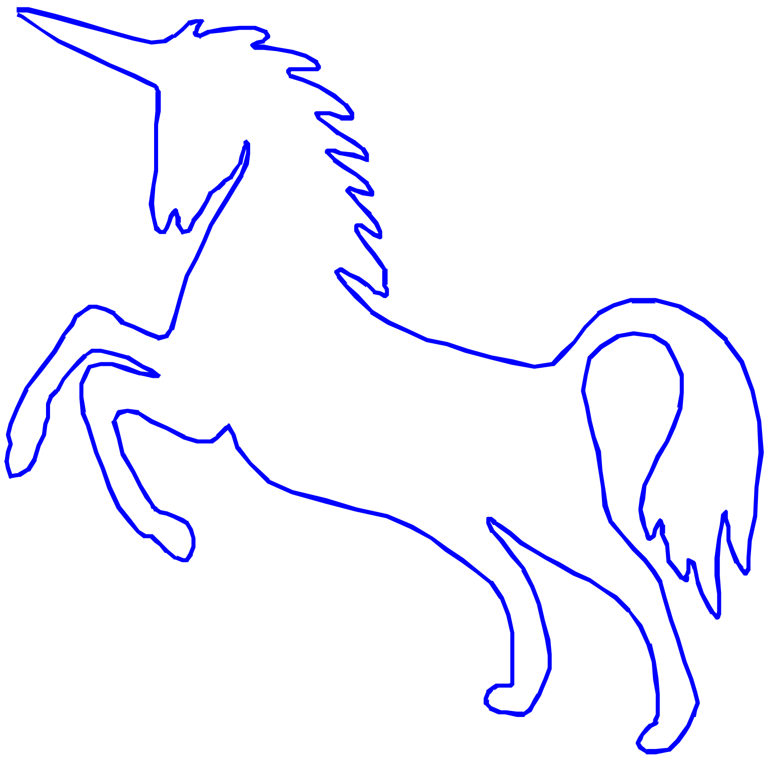
2D Curve – Example



1st iteration; $\lambda=0.5$

Laplacian smoothing

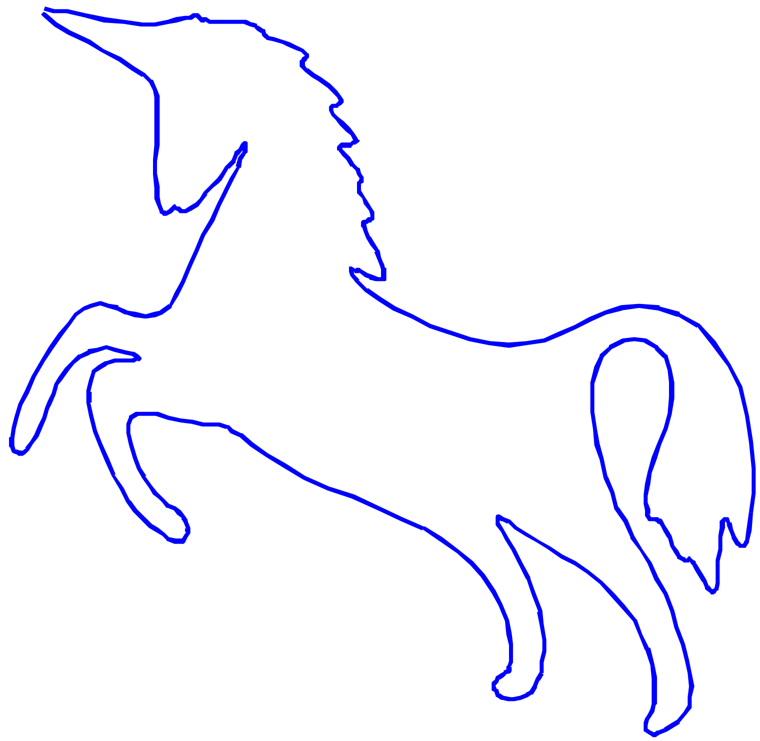
2D Curve – Example



2nd iteration; $\lambda=0.5$

Laplacian smoothing

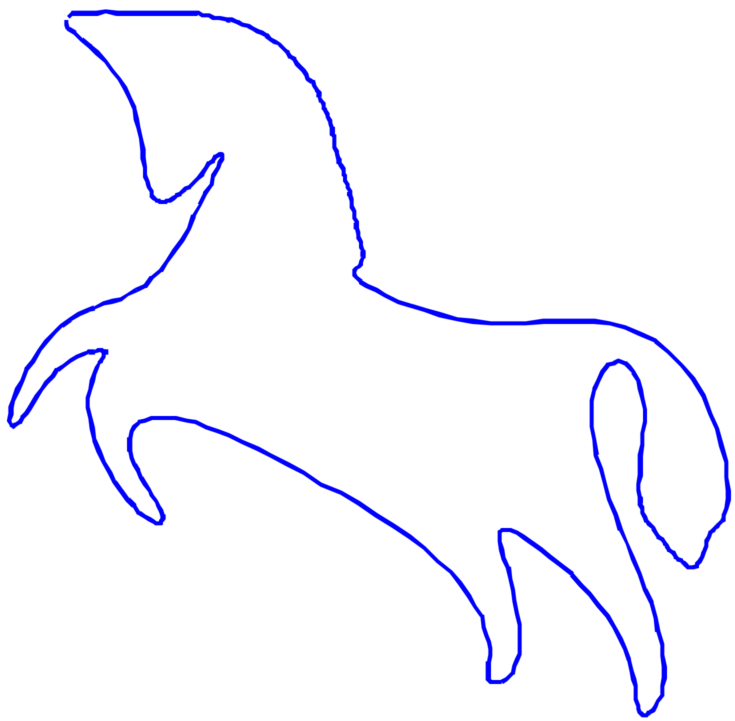
2D Curve – Example



8th iteration; $\lambda=0.5$

Laplacian smoothing

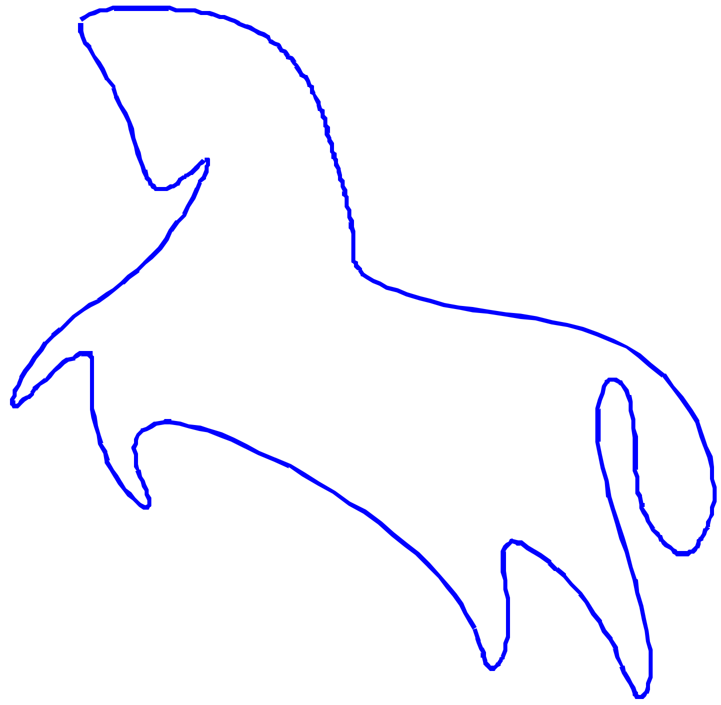
2D Curve – Example



27th iteration; $\lambda=0.5$

Laplacian smoothing

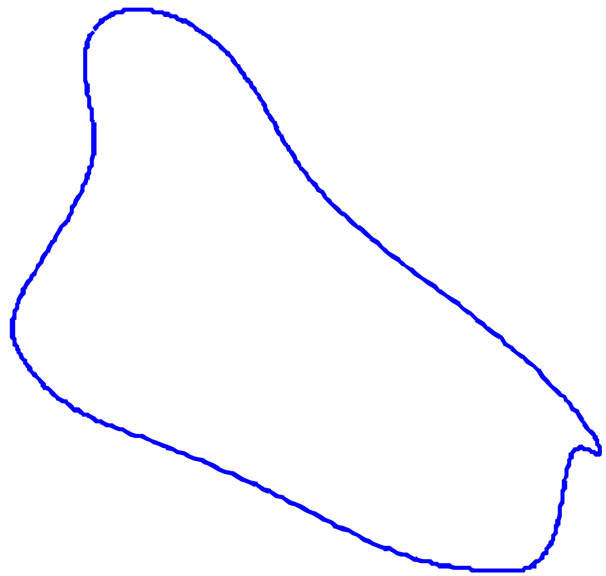
2D Curve – Example



50th iteration; $\lambda=0.5$

Laplacian smoothing

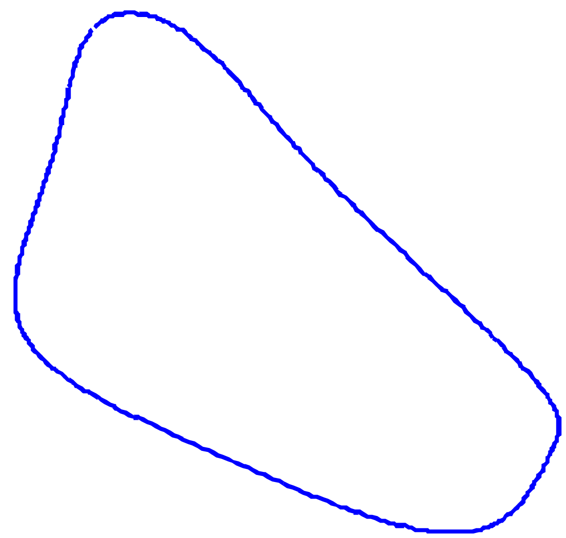
2D Curve – Example



500th iteration; $\lambda=0.5$

Laplacian smoothing

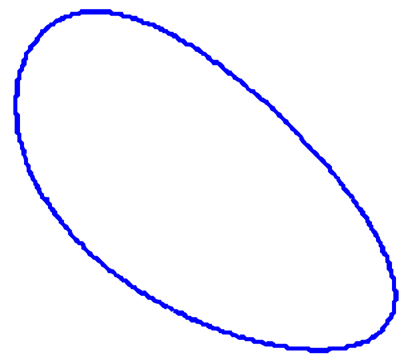
2D Curve – Example



1000th iteration; $\lambda=0.5$

Laplacian smoothing

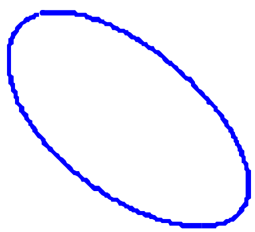
2D Curve – Example



5000th iteration; $\lambda=0.5$

Laplacian smoothing

2D Curve – Example



10000th iteration; $\lambda=0.5$

Laplacian smoothing

2D Curve – Example

50000th iteration; $\lambda=0.5$

Surface smoothing

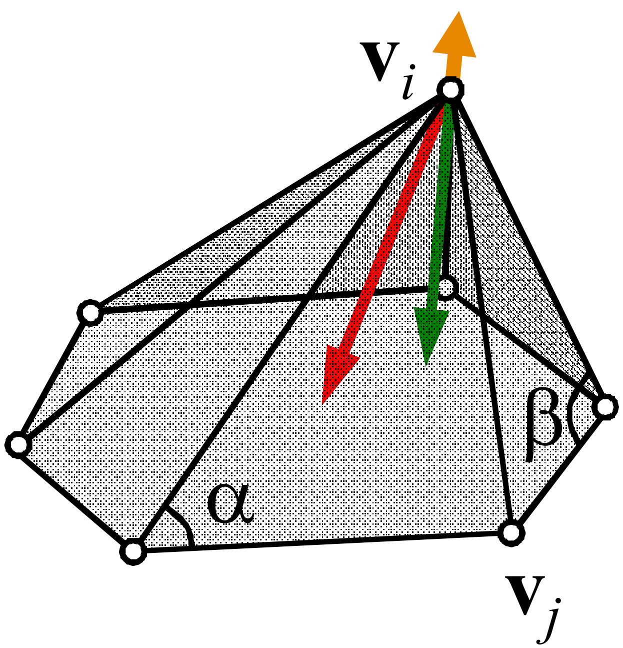
- Normalized Laplacian weights $\sum_{\{i,j\} \in E} w_{ij} = 1$

$$\delta_i = \sum_{\{i,j\} \in \mathbf{E}} w_{ij}(\mathbf{v}_j - \mathbf{v}_i) = \left[\sum_{\{i,j\} \in \mathbf{E}} w_{ij} \mathbf{v}_j \right] - \mathbf{v}_i$$

$$w_{ij} = \frac{\omega_{ij}}{\sum_{\{i,k\} \in \mathbf{E}} \omega_{ik}}$$

$$\omega_{ij} = 1,$$

$$\omega_{ij} = cot\ \alpha + cot\ \beta$$



Surface smoothing

- Matrix-vector notation for $L(\mathbf{x}) = \mathbf{L}\mathbf{x}$

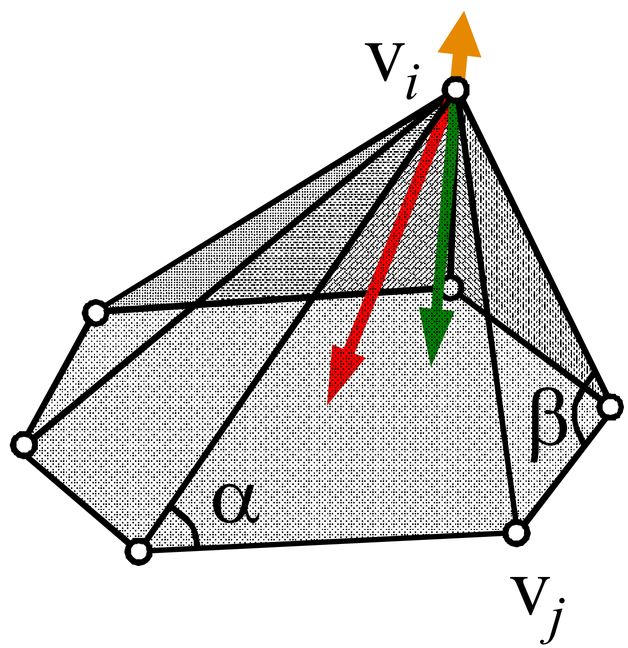
$$\mathbf{L}_{ij} = \begin{cases} -1 & i = j \\ w_{ij} & (i,j) \in \mathbf{E} \\ 0 & \text{otherwise} \end{cases}$$

$\mathbf{L}$ is the n x n Laplacian Matrix

$$w_{ij} = \frac{\omega_{ij}}{\sum_{\{i,k\} \in \mathbf{E}} \omega_{ik}}$$

$$\omega_{ij} = 1,$$

$$\omega_{ij} = cot\ \alpha + cot\ \beta$$



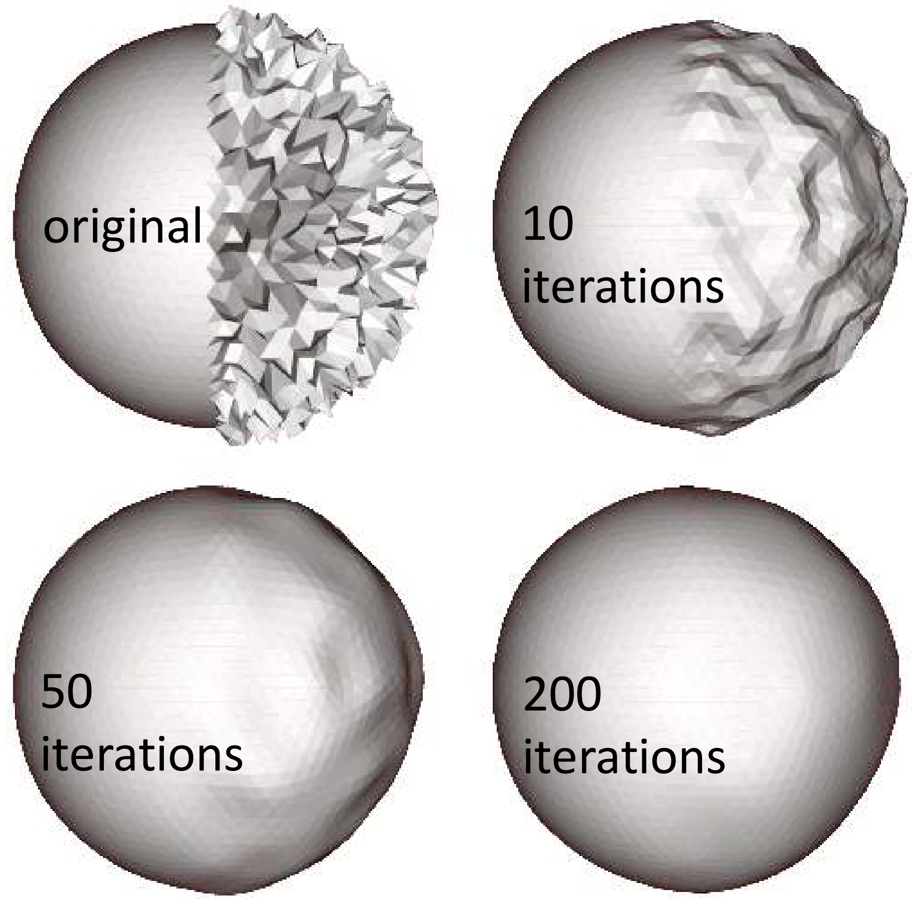
Taubin smoothing

- Idea: perform *inflation* after shrinking step
- Pick a $\mu < -\lambda$
- Iterate the following two steps

$$x_i' = x_i + \lambda \Delta x_i$$

$$x_i' = x_i + \mu \Delta x_i$$

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



Implicit fairing

- Model smoothing as a diffusion process

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

- Scale λ by simulation parameter time t

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

- Backward Euler for unconditional stability

$$X^{n+1} = X^n + \lambda dt\, L(X^{n+1})$$

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

Implicit fairing

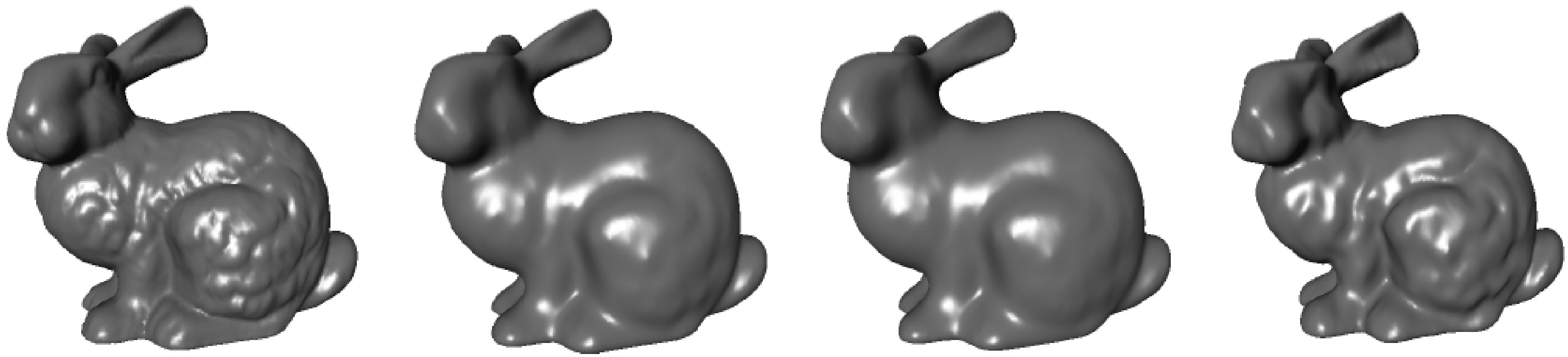
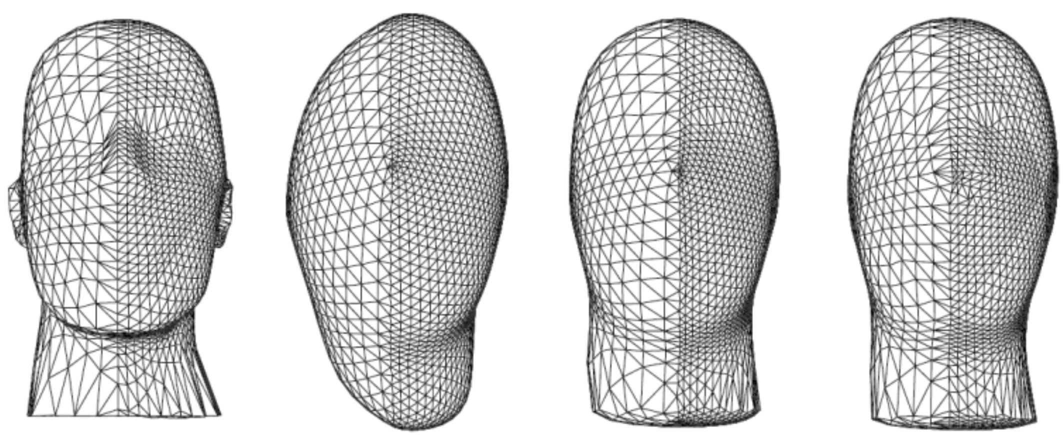
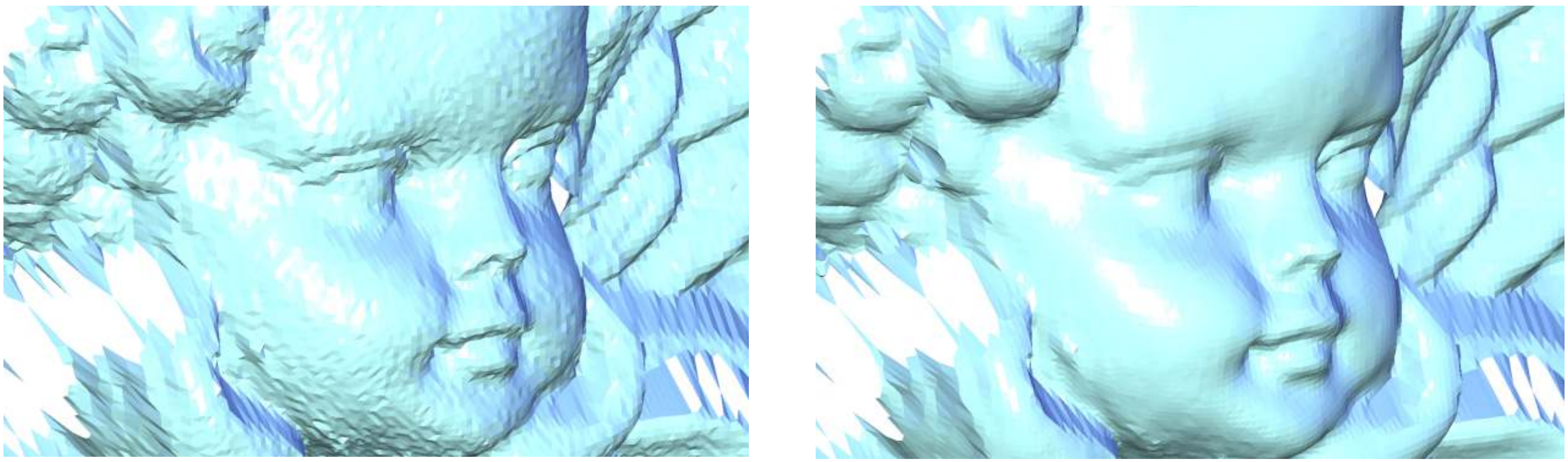


Figure 4: *Stanford bunnies: (a) The original mesh, (b) 10 explicit integrations with $\lambda dt = 1$, (c) 1 implicit integration with $\lambda dt = 10$ that takes only 7 PBCG iterations (30% faster), and (d) 20 passes of the $\lambda|\mu$ algorithm, with $\lambda = 0.6307$ and $\mu = -0.6732$. The 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.*

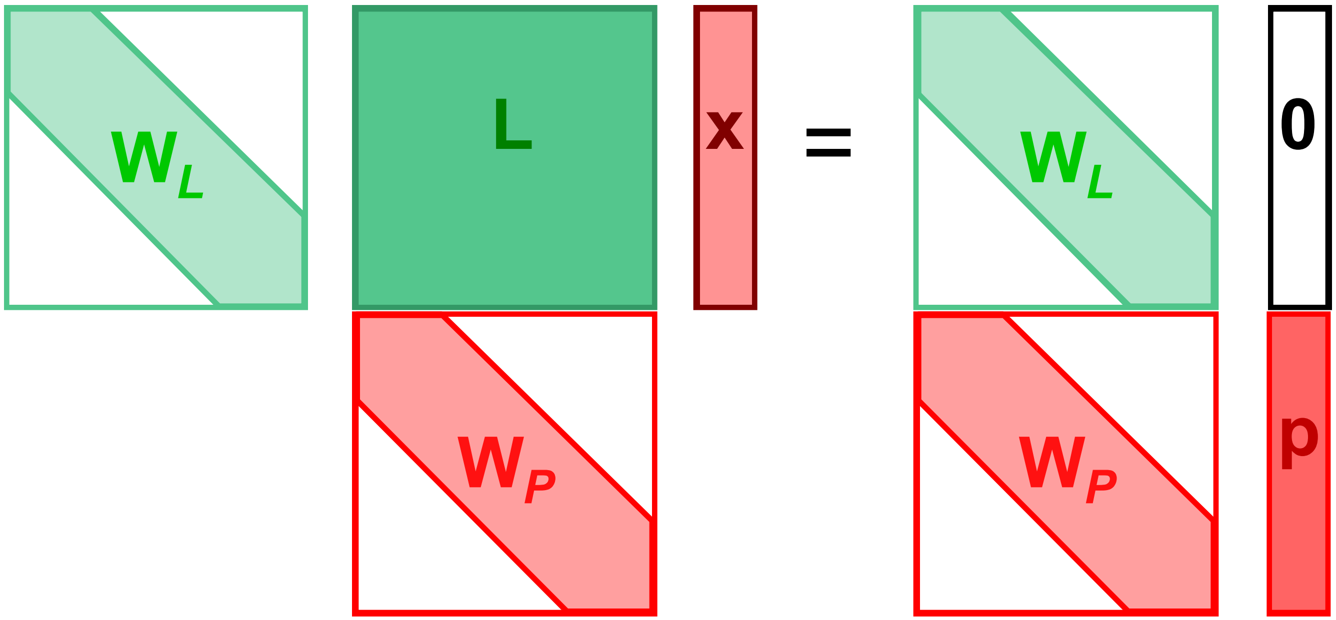
- Use cotangent instead of uniform Laplacian



Laplacian mesh optimization



Laplacian mesh optimization

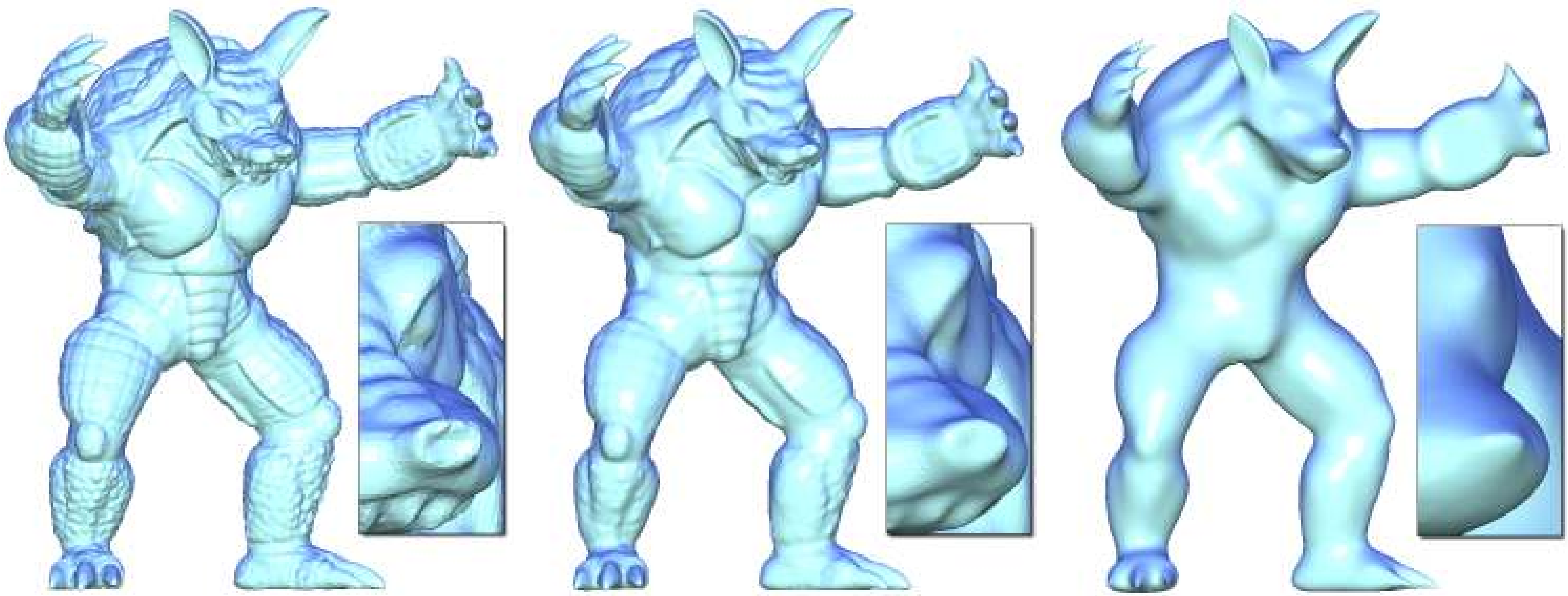


- **Mesh smoothing** $L = L_{cot}$ (outer fairness) or $L = L_{uni}$ (outer and inner fairness)
- Controlled by W_P and W_L (Intensity, Features)
- Least squares solve using normal equations

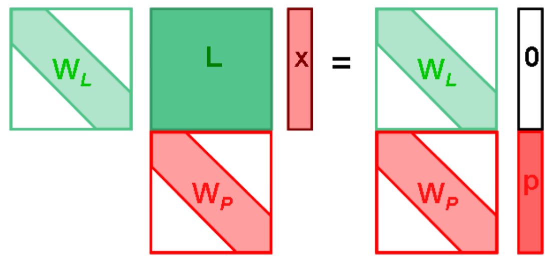
$$A^T A \; x = A^T b$$
$$x = (A^T A)^{-1} A^T b$$

$$\begin{bmatrix} W_L L \\ W_P \end{bmatrix} x = \begin{bmatrix} W_L 0 \\ W_P p \end{bmatrix}$$

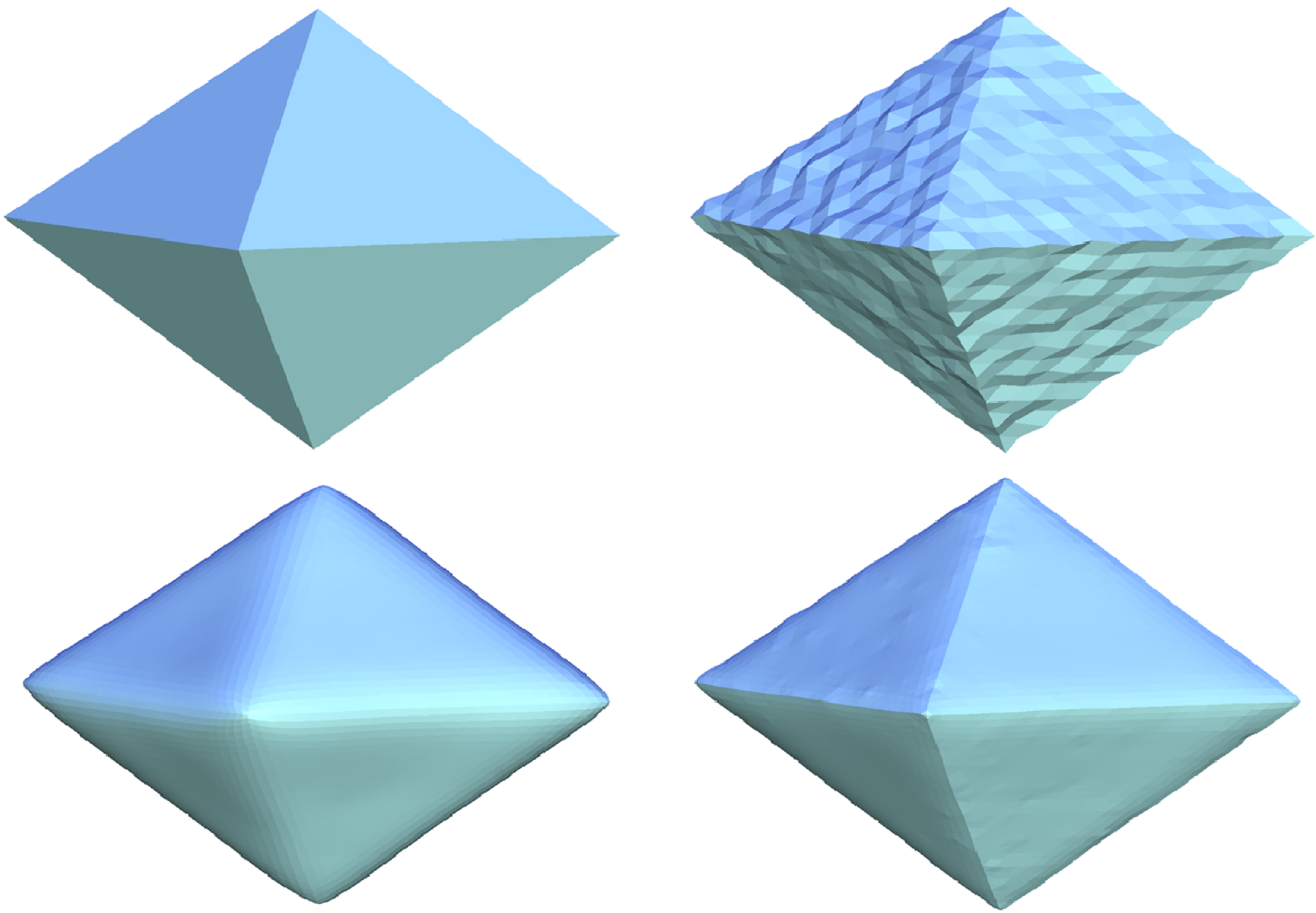
Using W_P



(a) original (173k) (c) cdf weights (s = 0.2) (f) cdf weights (s = 0.02)



Using W_P and W_L



Mesh Simplification

Level of detail (LOD)

Hausdorff distance

Mesh optimization

Error quadrics

Progessive representations

- Polygon simplification and level of detail (LOD) reduce geometric complexity of (small) objects

Preprocess
Compute Levels of Detail (LOD)

Runtime
Select LOD based on screen size

Simplification

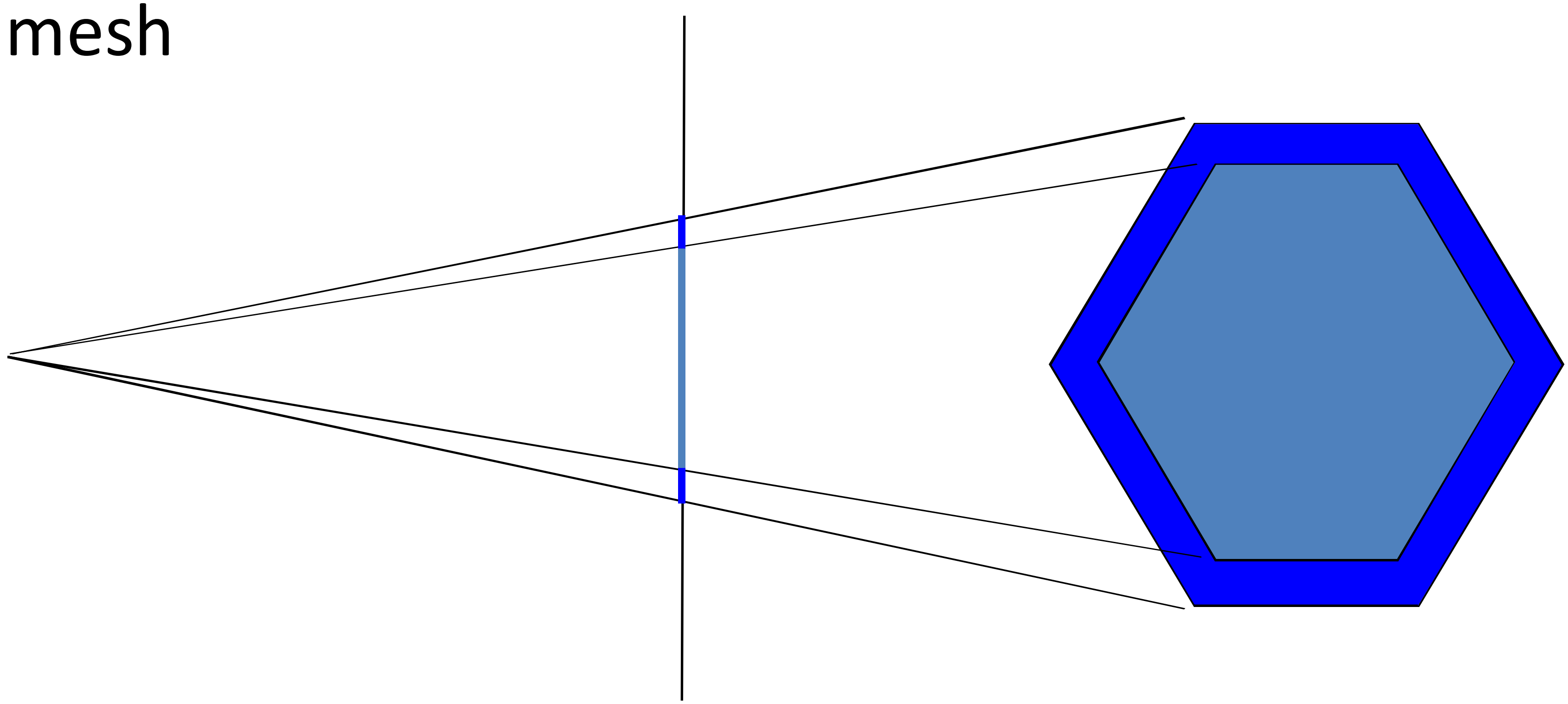
Criteria

- Usually based on visual error
  - Difficult to quantify
  - Often used
    - Geometric distance between original und simplified object
    - Volume between original and simplified object
    - Difference between surface normals between original and simplified object
    - Combination of multiple criteria

Simplification

Criteria

- Why geometric distance?
  - Fails for occluded/foreshortened parts of the mesh

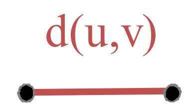


Approximation criteria

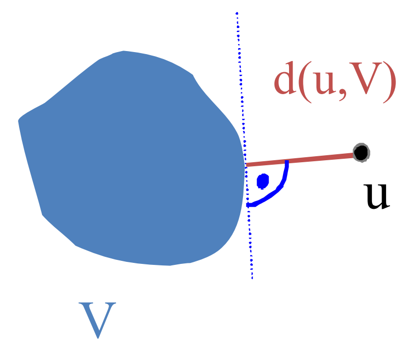
Geometric distance

- Hausdorff distance (distance between point sets):

$$\mathbf{u},\mathbf{v} \in \mathrm{R}^d : d(\mathbf{u},\mathbf{v}) = \left\|\mathbf{u} - \mathbf{v}\right\|$$



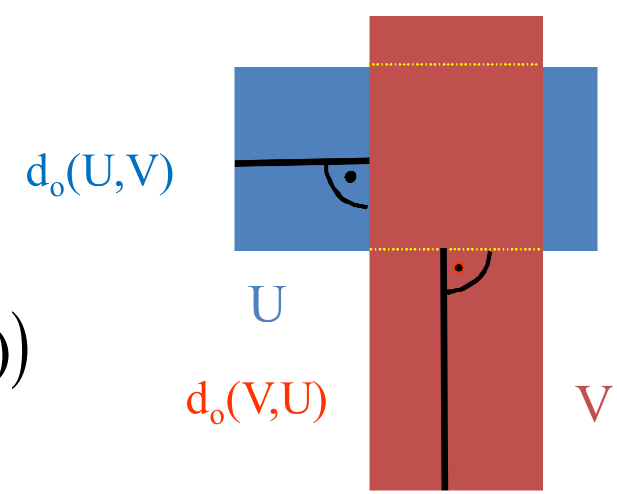
$$\mathbf{u} \in \mathrm{R}^d, \mathbf{V} \subset \mathrm{R}^d : d(\mathbf{u},\mathbf{V}) = \inf_{\mathbf{v} \in \mathbf{V}} d(\mathbf{u},\mathbf{v})$$



$$\mathbf{U} \subset \mathrm{R}^d, \mathbf{V} \subset \mathrm{R}^d : d_{\mathrm{O}}(\mathbf{U},\mathbf{V}) = \sup_{\mathbf{u} \in \mathbf{U}} \inf_{\mathbf{v} \in \mathbf{V}} d(\mathbf{u},\mathbf{v})$$

$$d_{\mathrm{O}}(\mathbf{U},\mathbf{V}) \neq d_{\mathrm{O}}(\mathbf{V},\mathbf{U})$$

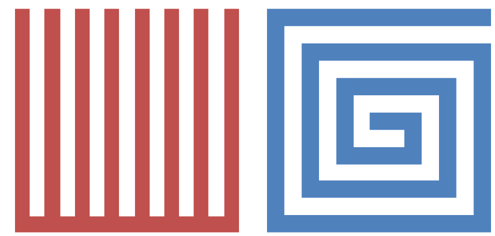
$$\mathbf{U} \subset \mathrm{R}^d, \mathbf{V} \subset \mathrm{R}^d : d_{\mathrm{H}}(\mathbf{U},\mathbf{V}) = \max\left(d_{\mathrm{O}}(\mathbf{U},\mathbf{V}), d_{\mathrm{O}}(\mathbf{V},\mathbf{U})\right)$$

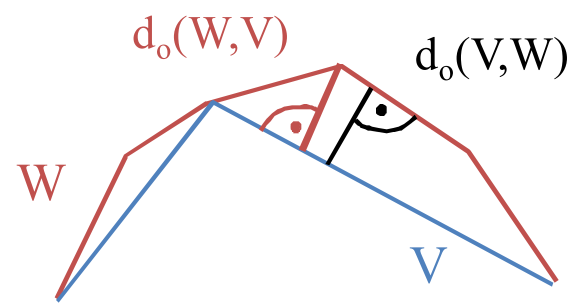


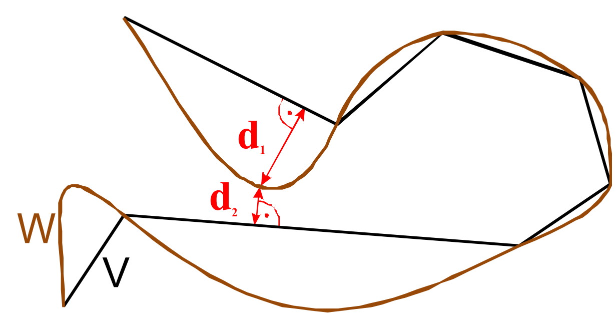
Approximation criteria

Geometric distance

- Hausdorff distance
  - Generally bad for orientation or shape comparison
  - Very good metric for comparing original to simplified model
  - Topolocial correspondence improves results



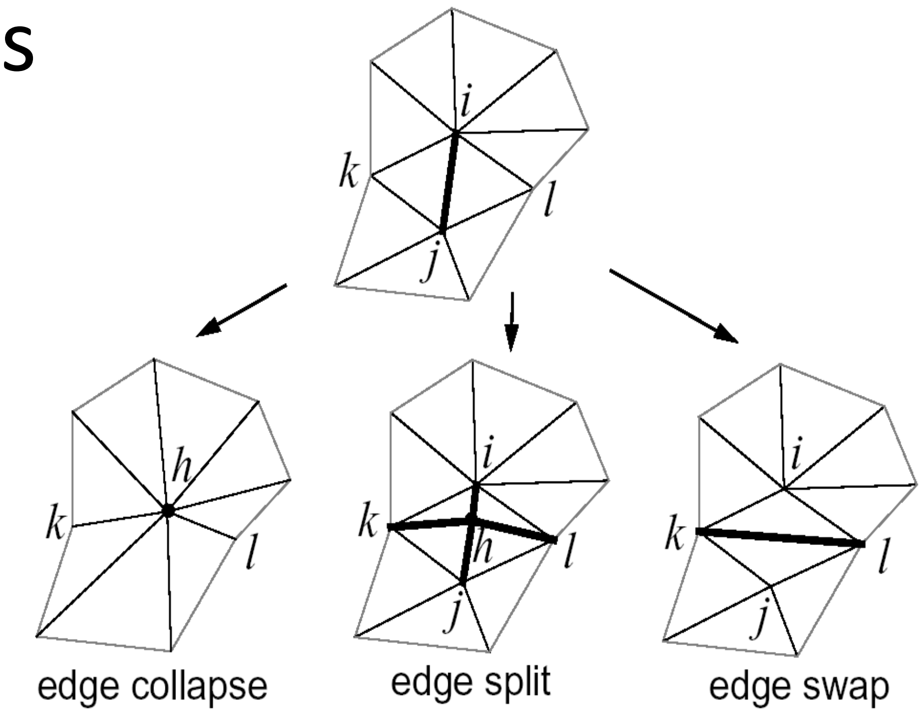




Mesh optimization

Local operations

- Three local operations **[Hoppe 1993]**
  - Edge Collapse
  - Edge Split
  - Edge Swap/Flip

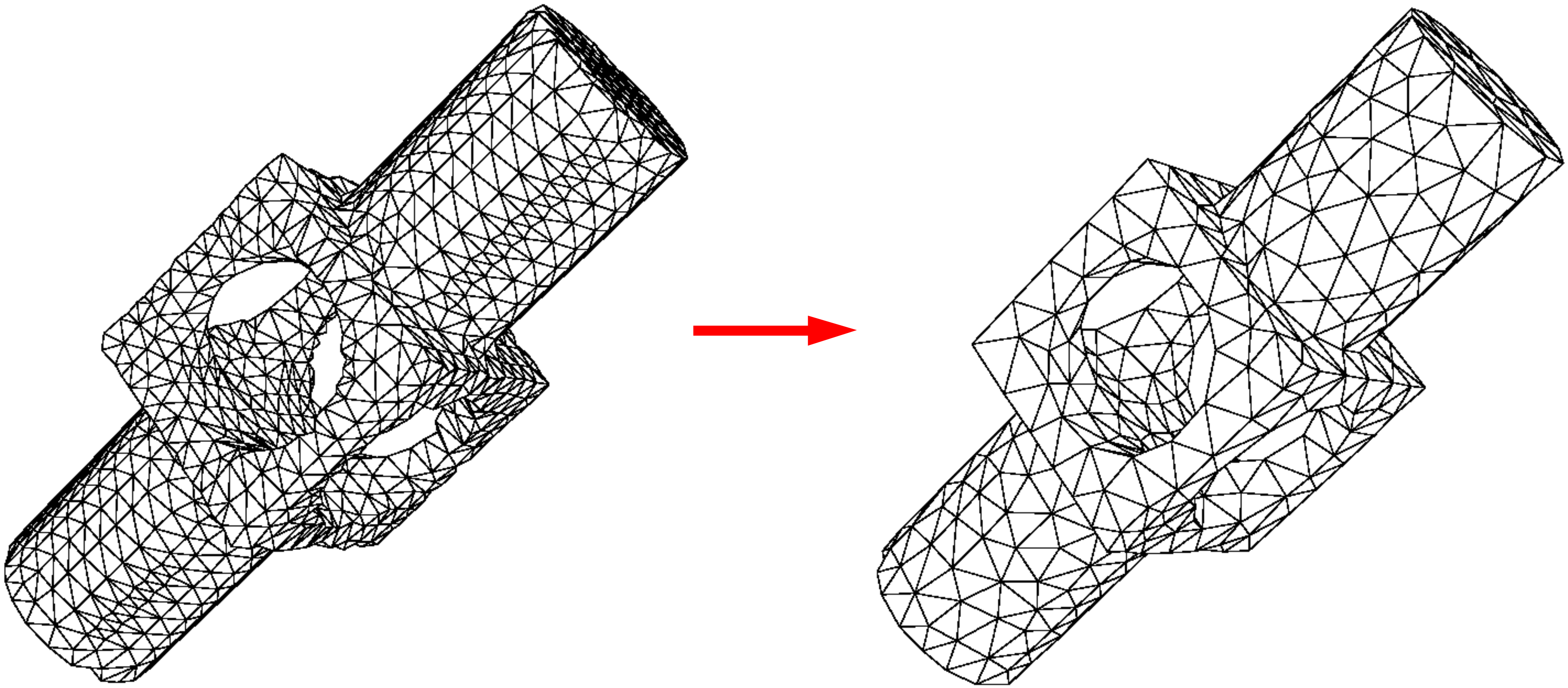


- Minimize some global energy

$$E(K,V) = E_{dist}(K,V) + E_{rep}(K) + E_{spring}(K,V)$$

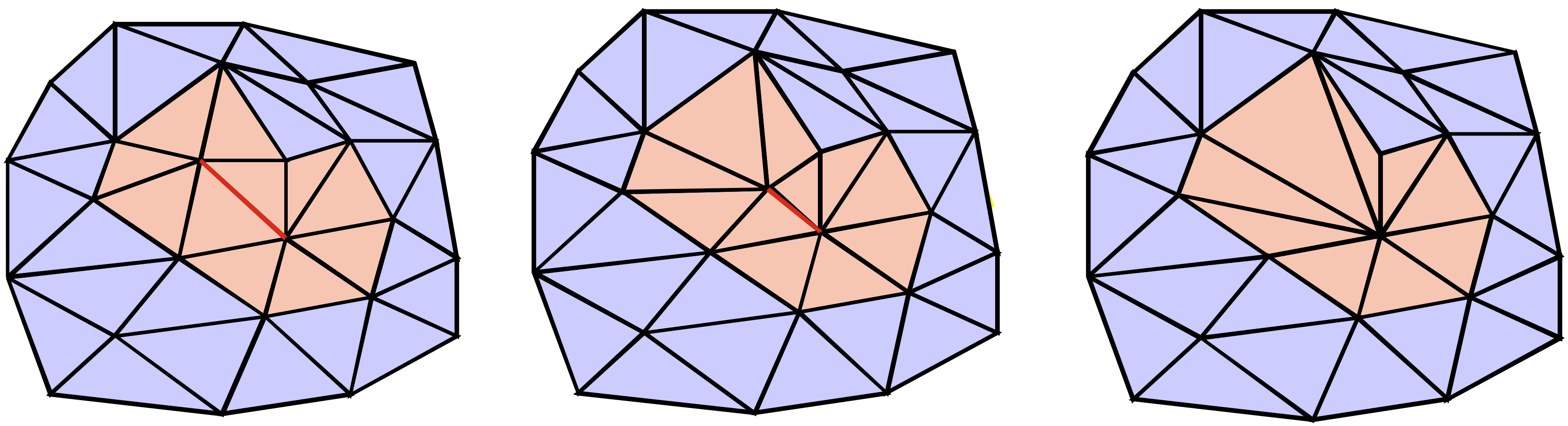
Mesh optimization

Results



Simplification

Edge collapse



Simplification

Evaluating the operations

- For edge collapse selection, the approximation error must be evaluated

- Possibilities:
  - Hausdorff distance (expensive)
  - One-sided Hausdorff distance
  - Accumulate squared distances to planes (Error quadrics)
    - Basic operation: edge collapse

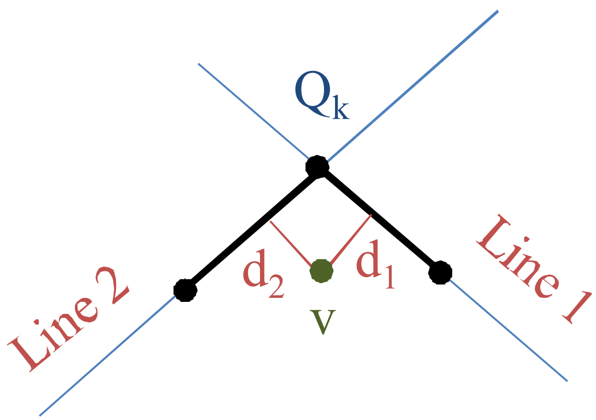
Simplification

Error quadrics

- Compute squared distance to planes
- Example: planar polygons

$$g_1 : a_1 x + b_1 y + c_1, \quad a_1^2 + b_1^2 = 1$$

$$g_2 : a_2 x + b_2 y + c_2, \quad a_2^2 + b_2^2 = 1$$



$$d_1^2 + d_1^2 = \mathbf{v}^{\mathrm{T}} \mathbf{Q}_k \mathbf{v}$$

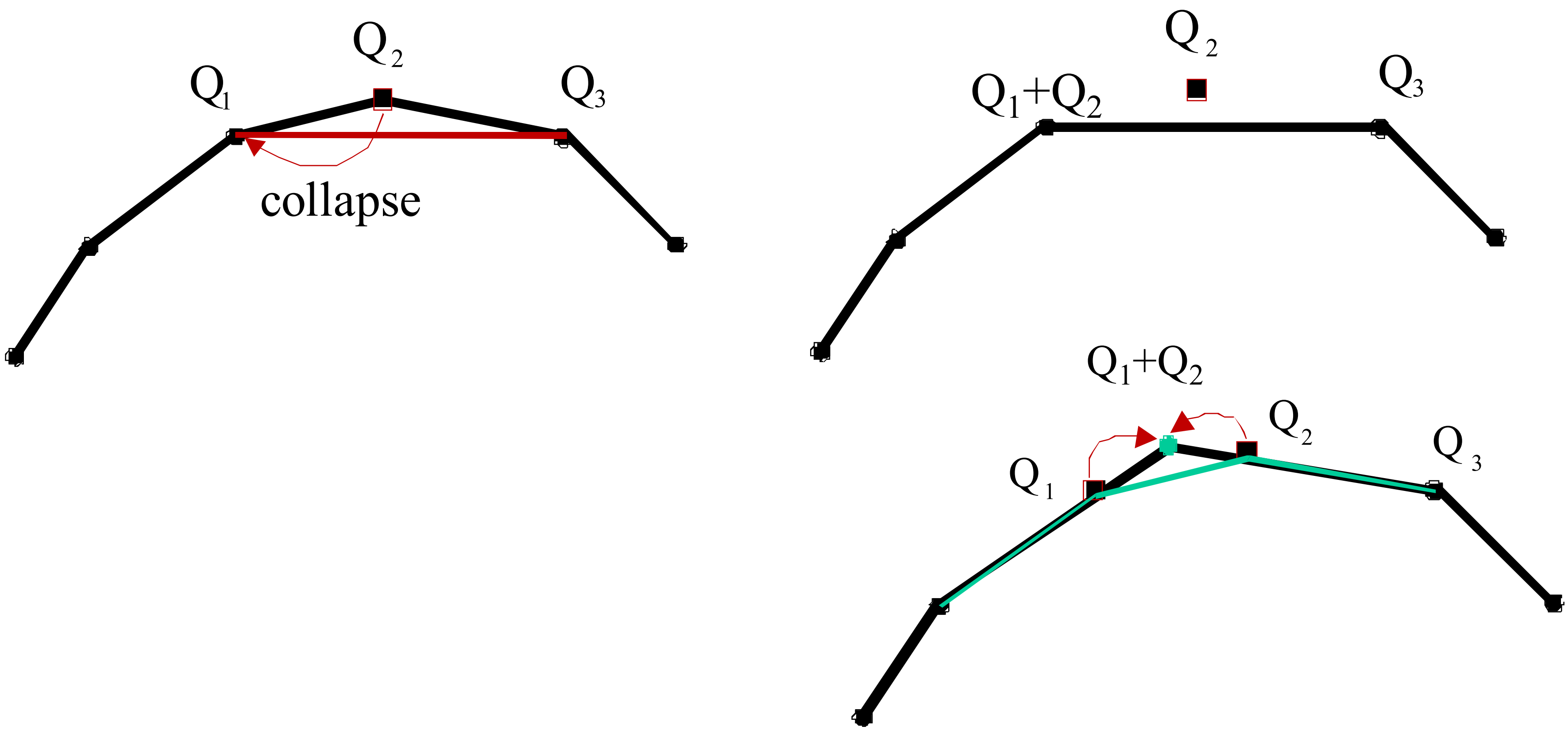
$$\begin{aligned} d_i^2 &= (a_i v_x + b_i v_y + c_i)^2 \\ &= \left((v_x, v_y, 1)(a_i, b_i, c_i)^T\right)^2 \\ &= (v_x, v_y, 1)\mathbf{Q}_{ik}(v_x, v_y, 1)^T, \\ \mathbf{Q}_{ik} &= (a_i, b_i, c_i)(a_i, b_i, c_i)^T \\ &= \begin{pmatrix} a_i^2 & a_i b_i & a_i c_i \\ a_i b_i & b_i^2 & b_i c_i \\ a_i c_i & b_i c_i & c_i^2 \end{pmatrix} \end{aligned}$$

$$\mathbf{Q}_k = \mathbf{Q}_{1k} + \mathbf{Q}_{2k}$$

Simplification

Error quadrics

- **[Garland/Heckbert 97]**:
  propagate error quadric by addition

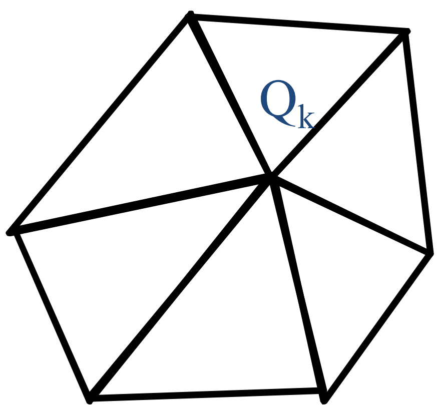


Simplification

Error quadrics

- Triangle meshes

$$p_{ik}: a_i x + b_i y + c_i z + d, \quad a_1^2 + b_1^2 + c_i^2 = 1$$



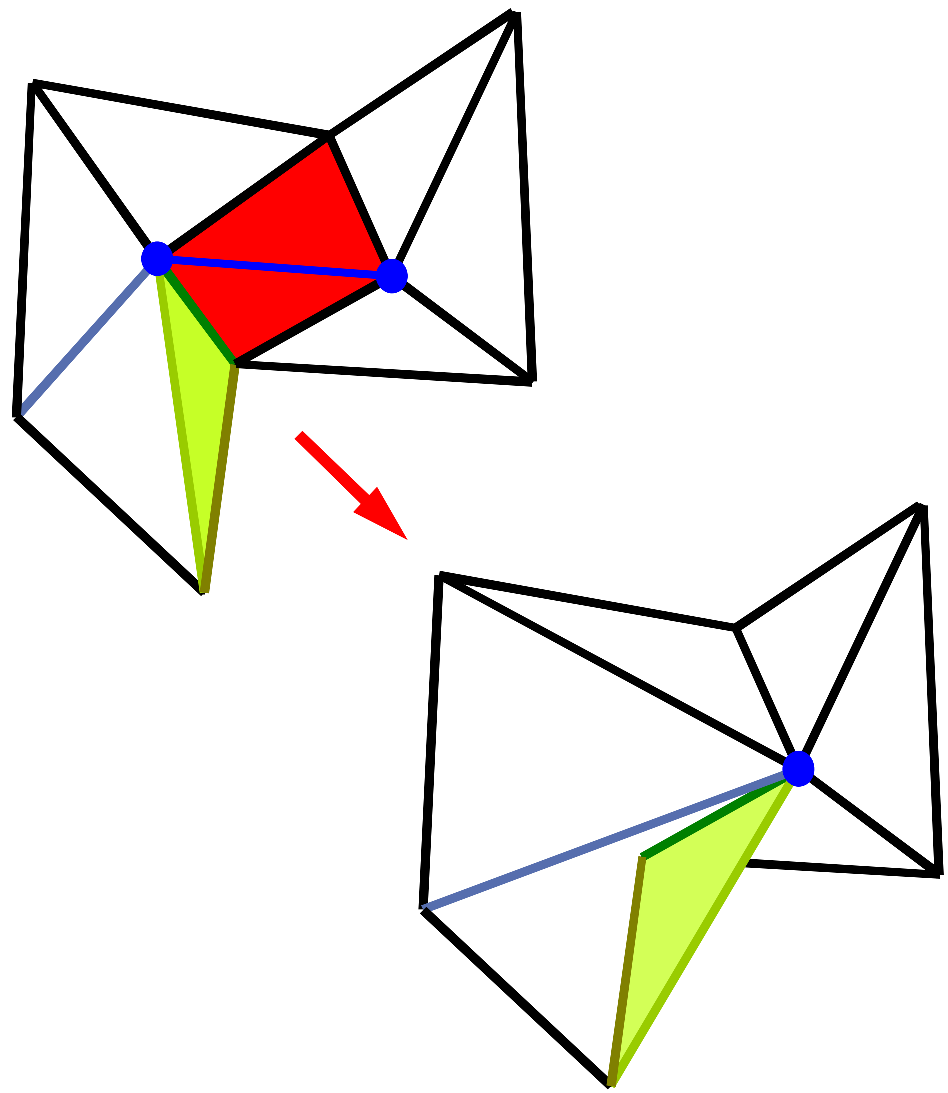
$$
\begin{aligned}
e_i^2 &= (a_i v_x + b_i v_y + c_i v_z + d_i)^2 \\
&= \left((v_x, v_y, v_z, 1)(a_i, b_i, c_i, d_i)\right)^2 \\
&= (v_x, v_y, v_z, 1)\mathbf{Q}_{ik}(v_x, v_y, v_z, 1)^T, \\
\mathbf{Q}_{ik} &= (a_i, b_i, c_i, d_i)(a_i, b_i, c_i, d_i)^T \\
&= \begin{pmatrix} a_i^2 & a_i b_i & a_i c_i & a_i d_i \\ a_i b_i & b_i^2 & b_i c_i & b_i d_i \\ a_i c_i & b_i c_i & c_i^2 & c_i d_i \\ a_i d_i & b_i d_i & c_i d_i & d_i^2 \end{pmatrix}
\end{aligned}
$$

$$\mathbf{Q}_k = \sum_i \mathbf{Q}_{ik}$$

Simplification

Error quadrics

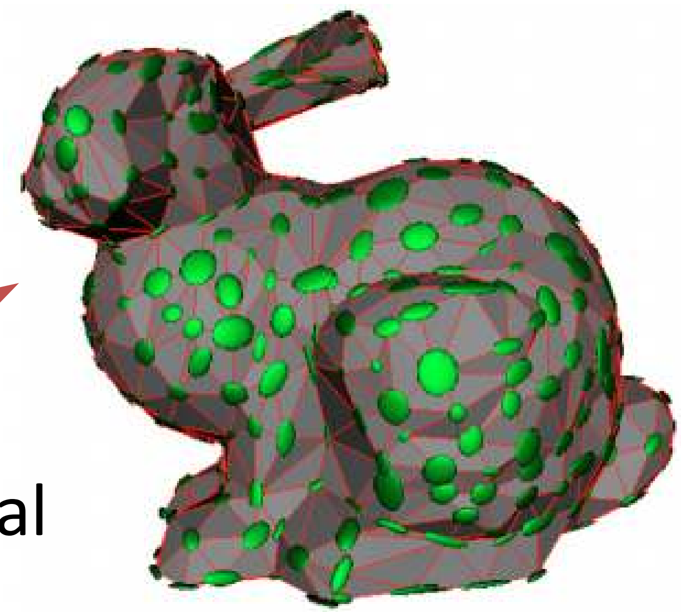
- Special cases
  - Face Flips: in collapsing the blue edge, the normal of the green face has flipped
  - These cases must be detected and avoided
  - OpenMesh takes care of this for you
  - OpenMesh has error quadrics implemented

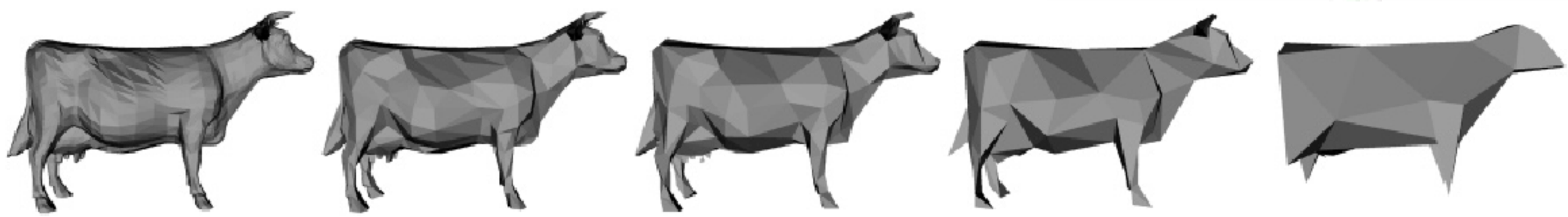


Simplification

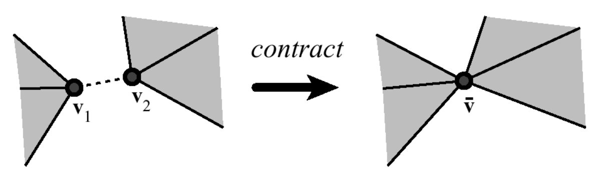
Error quadrics

- Advantages
  - Simple error computation
  - Very fast
  - Geometric interpretation
  - Good approximation of the original





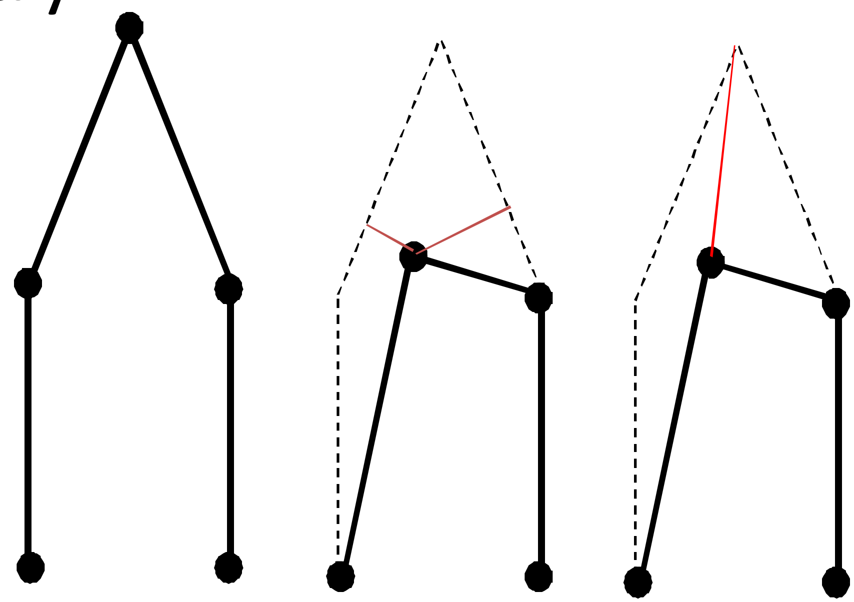
  - Can collapse vertices that are not connected via edges



Simplification

Error quadrics

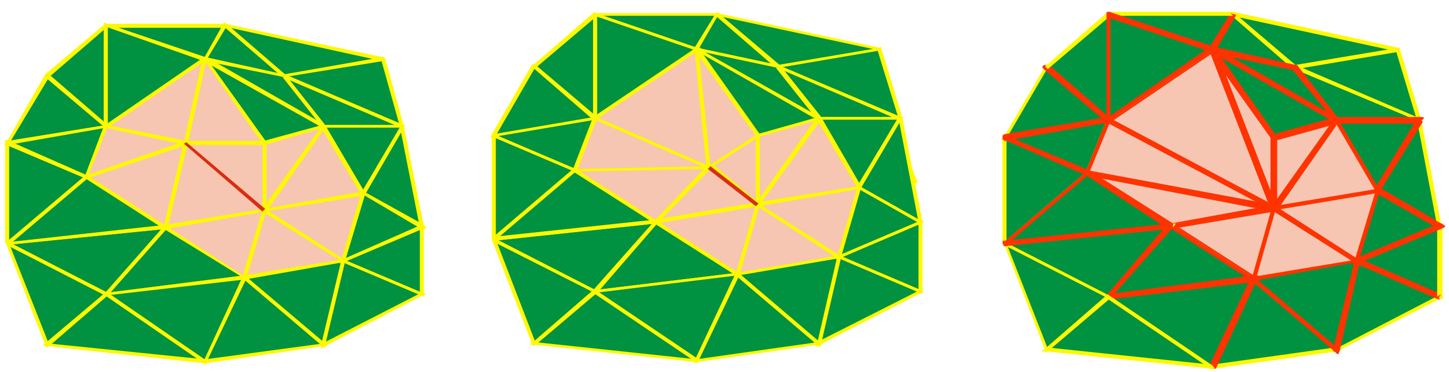
- Disadvantages
  - One sided distance only (from new to original)
  - Loss of symmetry



Simplification

Algorithm

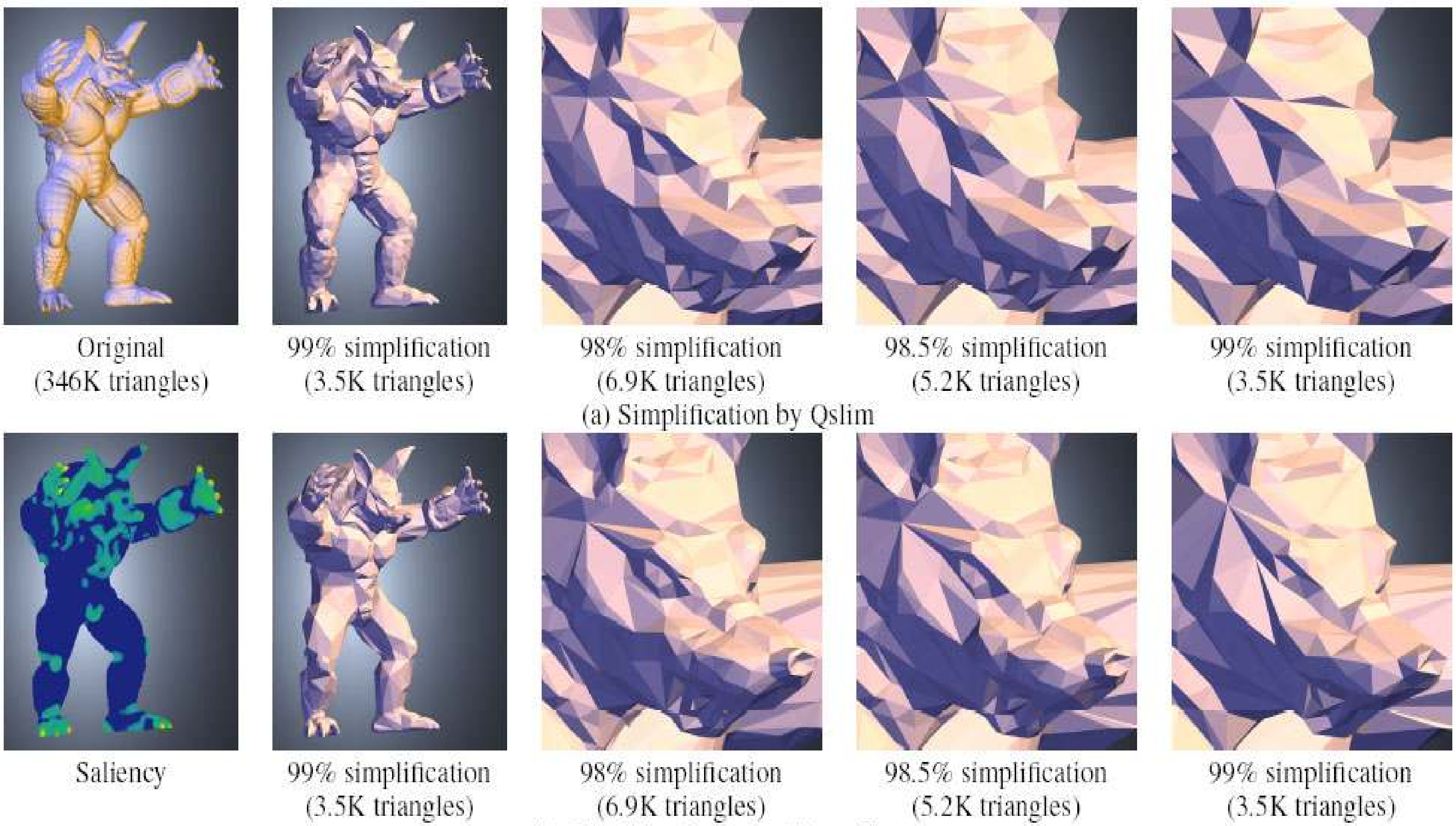
- Operation and error metric define algorithm
  - Compute error for every atomic simplification operation
  - Create a priority queue based on the error
  - While the queue is non-empty
    - Perform first simplification operation and remove from queue
    - Recompute the error for neighboring operations and update the priority queue accordingly



Simplification

Mesh Saliency [Lee et al. 2005]

- Take salient features into account when simplifying the mesh



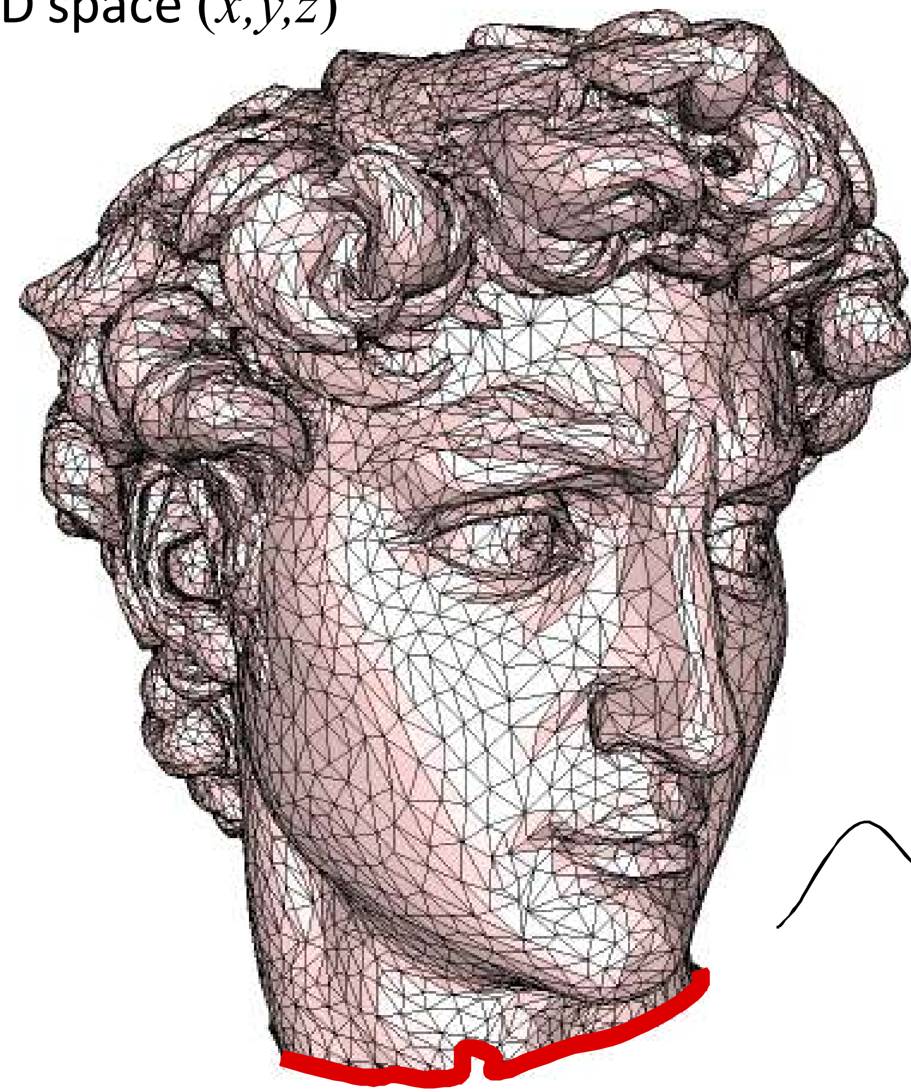
(a) Simplification by Qslim

(b) Simplification guided by saliency

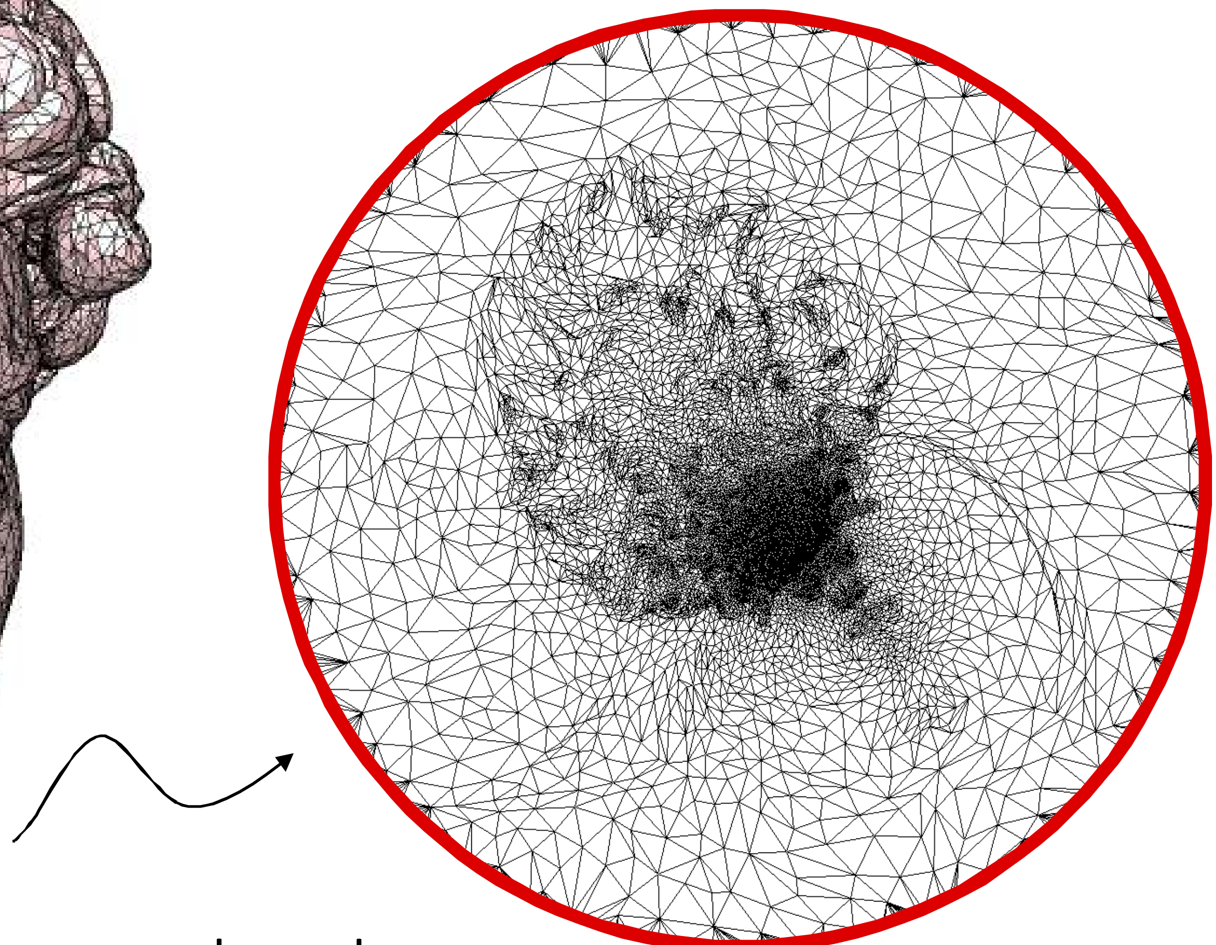
Parameterization and Remeshing

Surface parameterization

3D space (x,y,z)



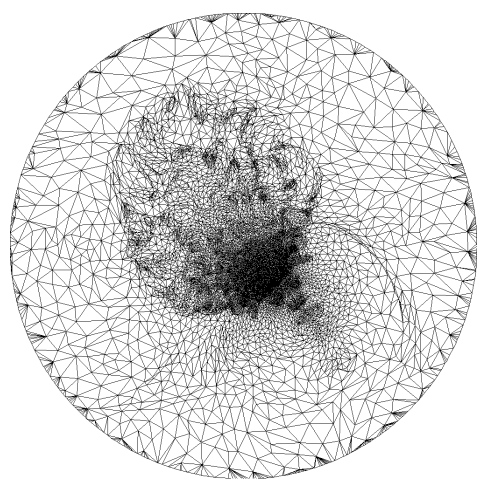
2D parameter domain (u,v)

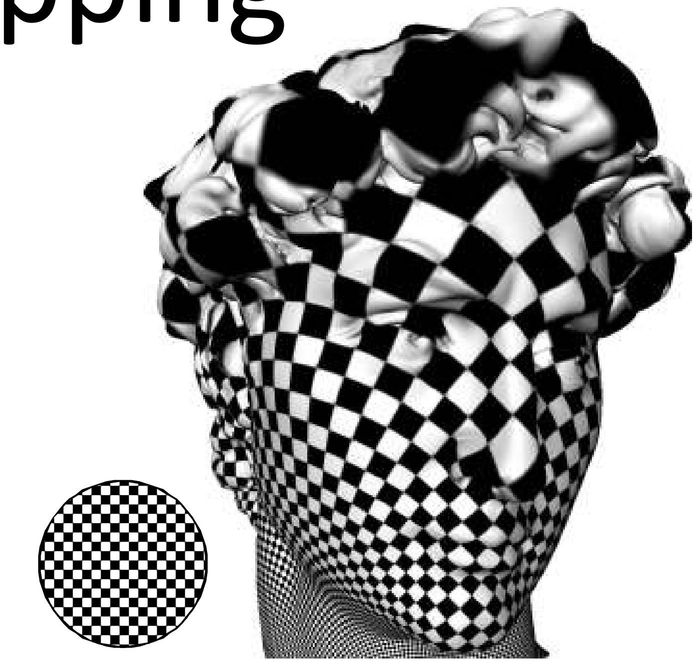


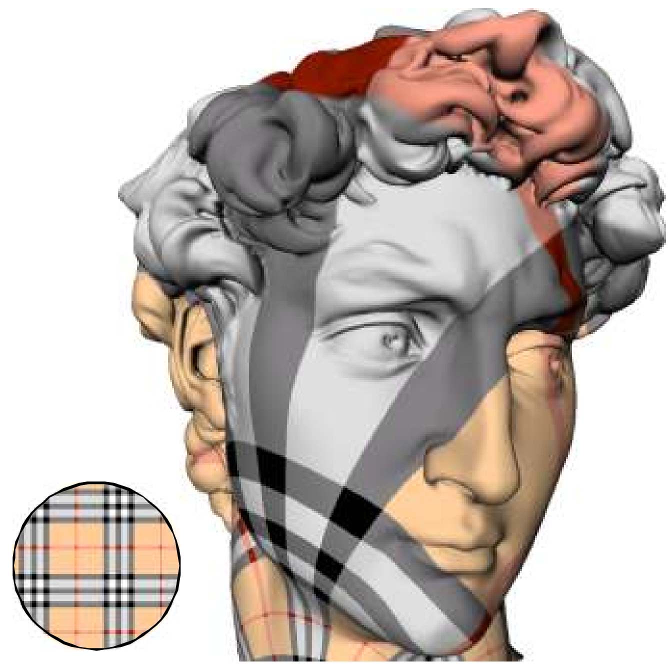
boundary

boundary

Texture mapping







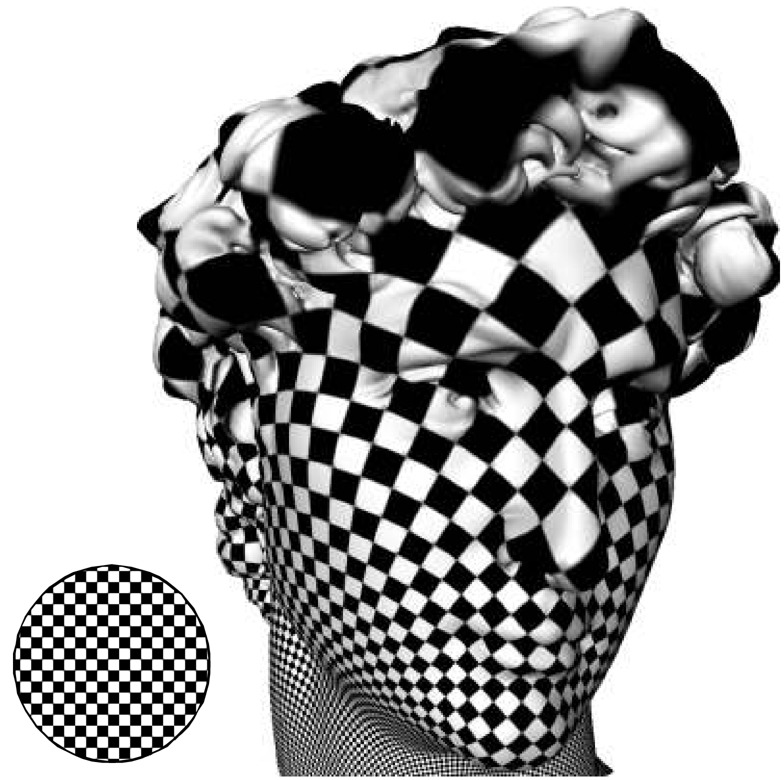
Texture mapping

Image from Vallet and Levy, techreport INRIA

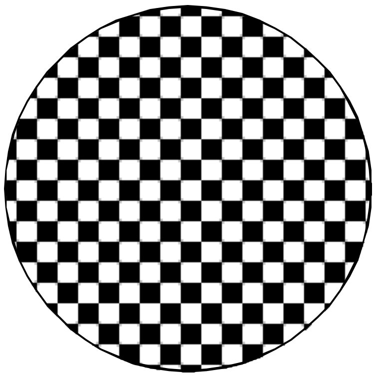
Mesh parameterization

Requirements

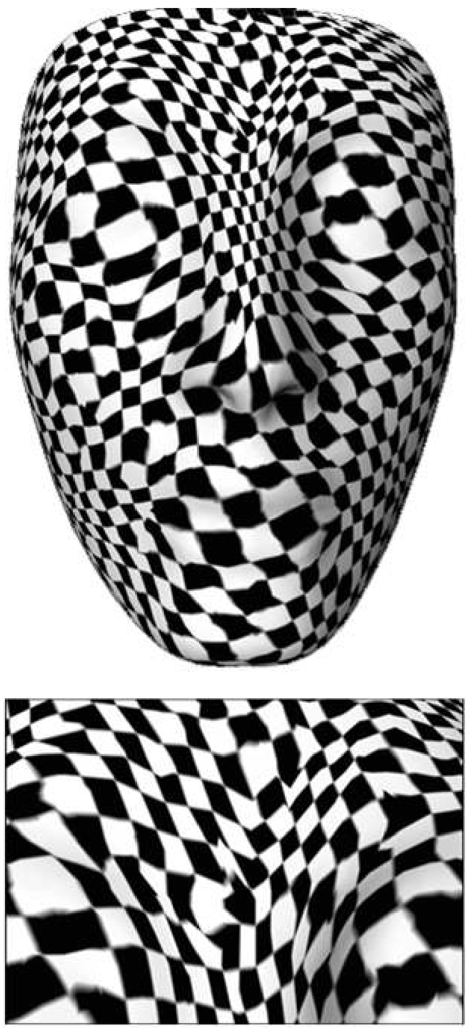
- Bijective (1-1 and onto): No triangles fold over.
- Minimal "distortion"
  - Preserve 3D angles
  - Preserve 3D distances
  - Preserve 3D areas
  - No "stretch"

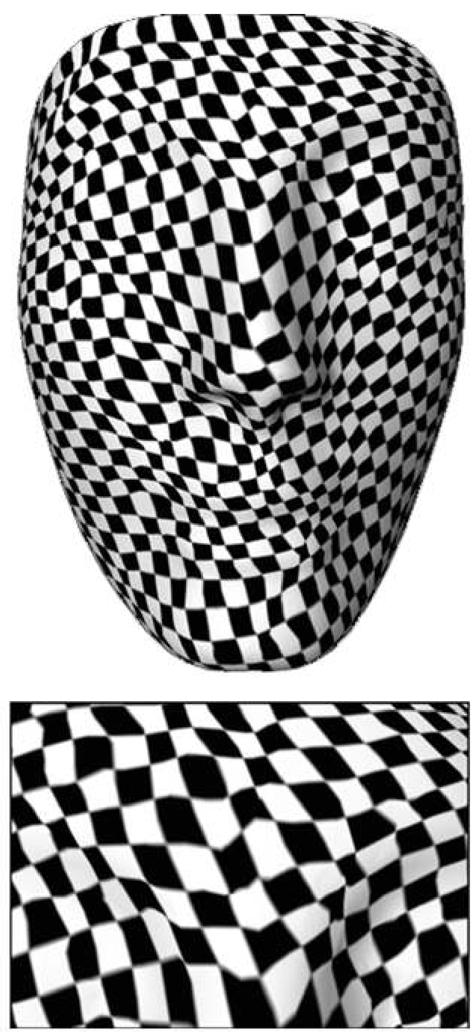


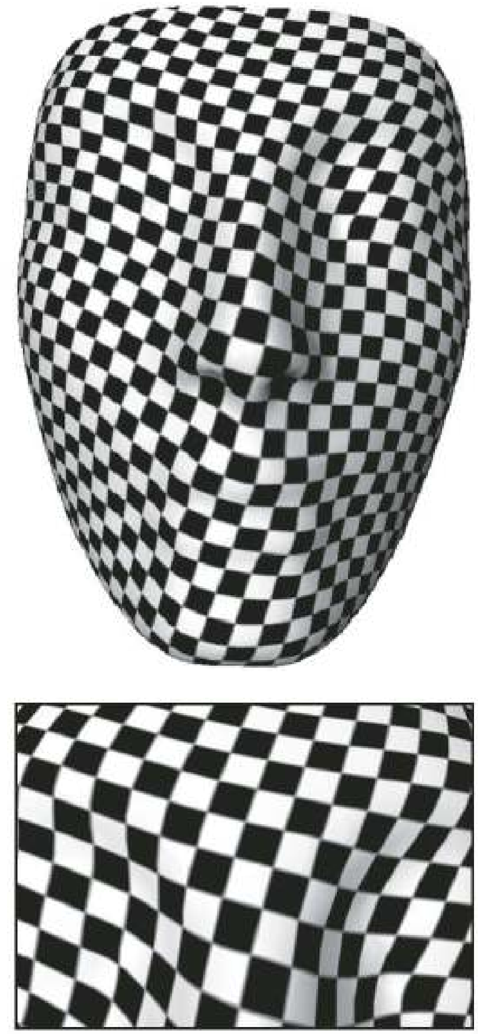
Distortion minimization



Texture map

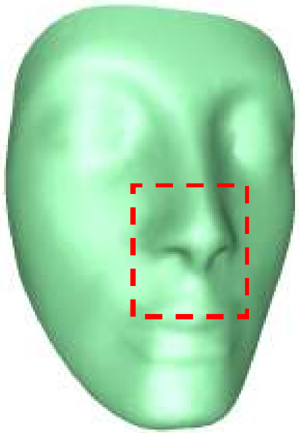


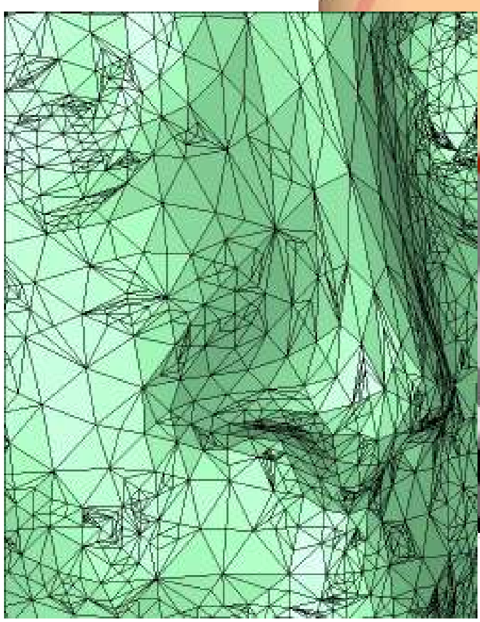


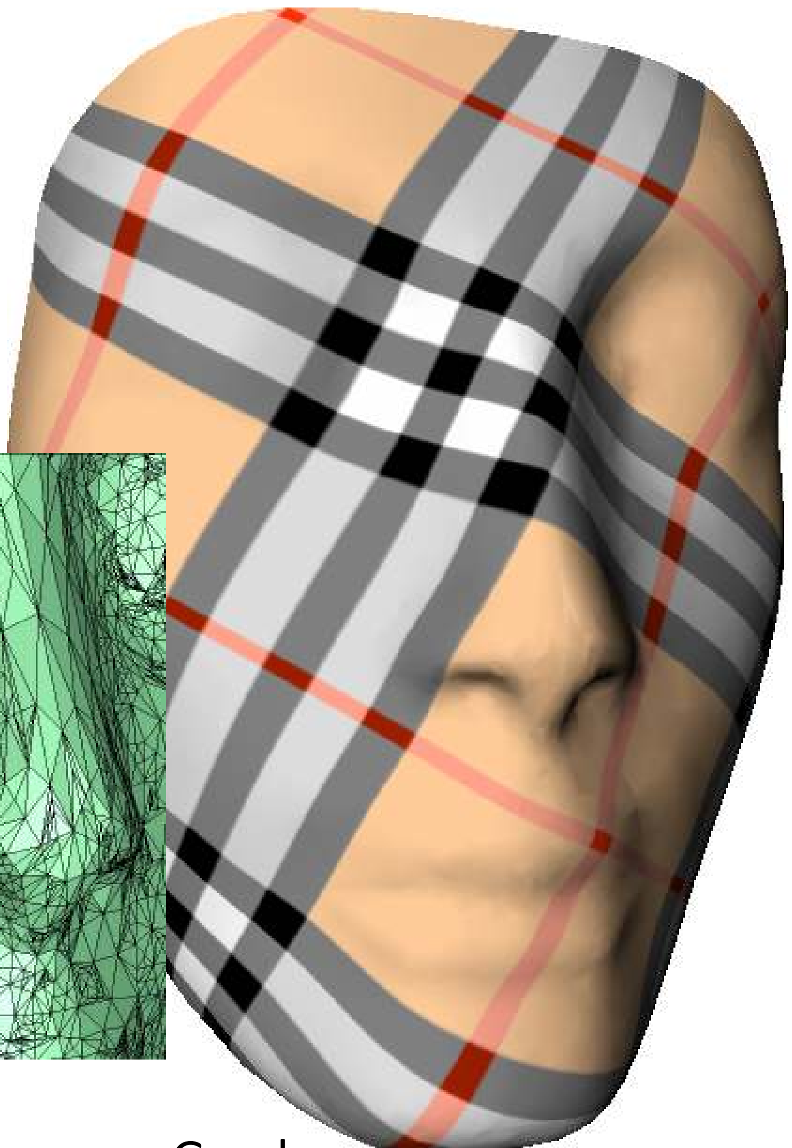


Kent et al ‘92　　Floater 97　　Sander et al ‘01

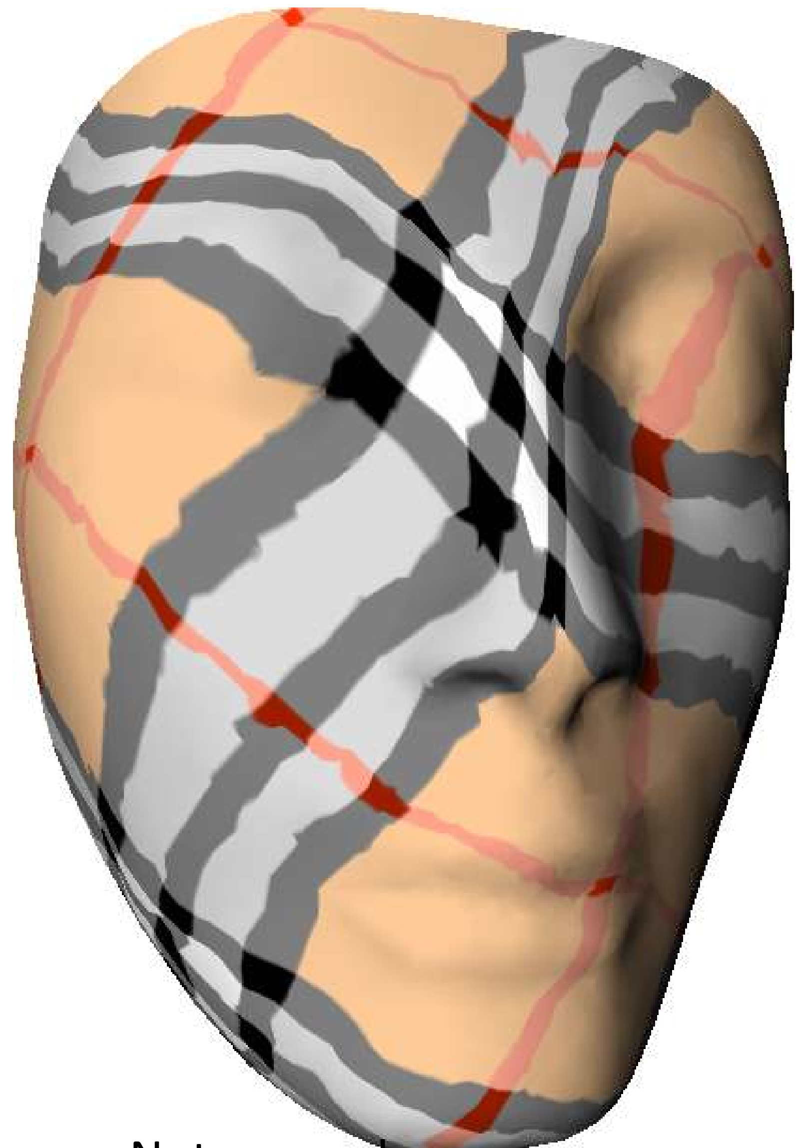
Sensitivity to mesh quality





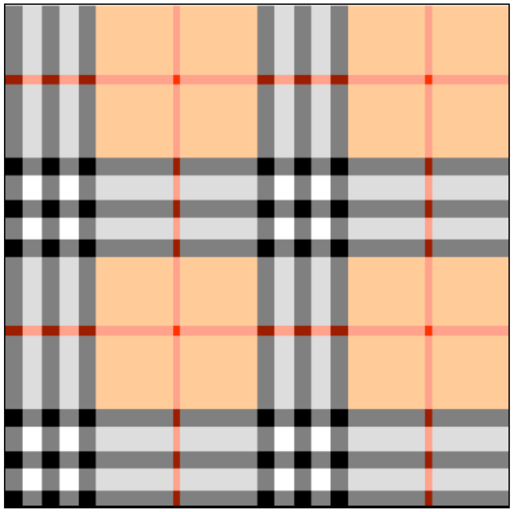
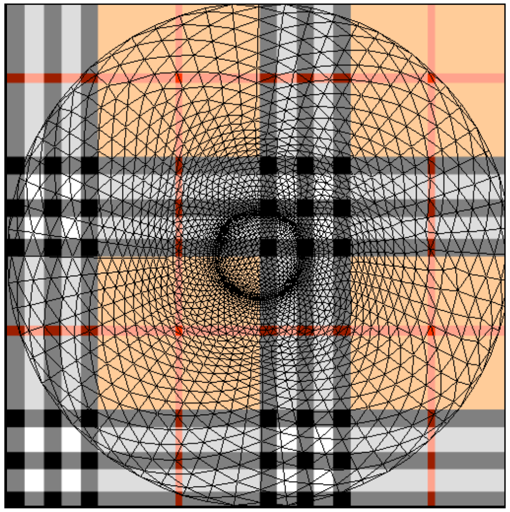
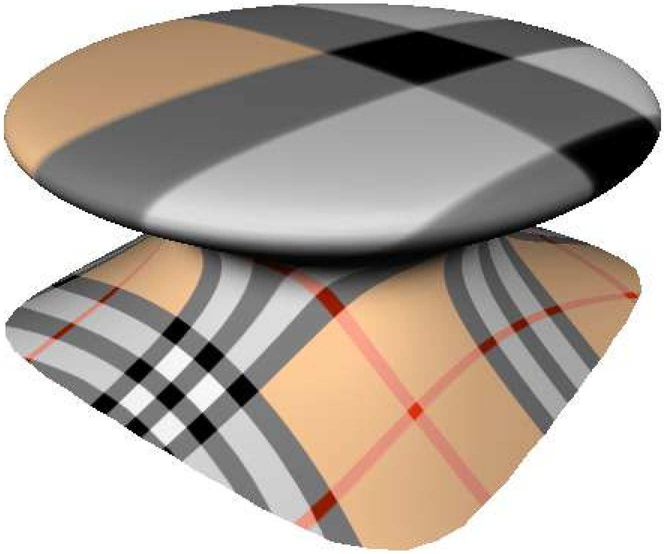


Good parameterization algorithm

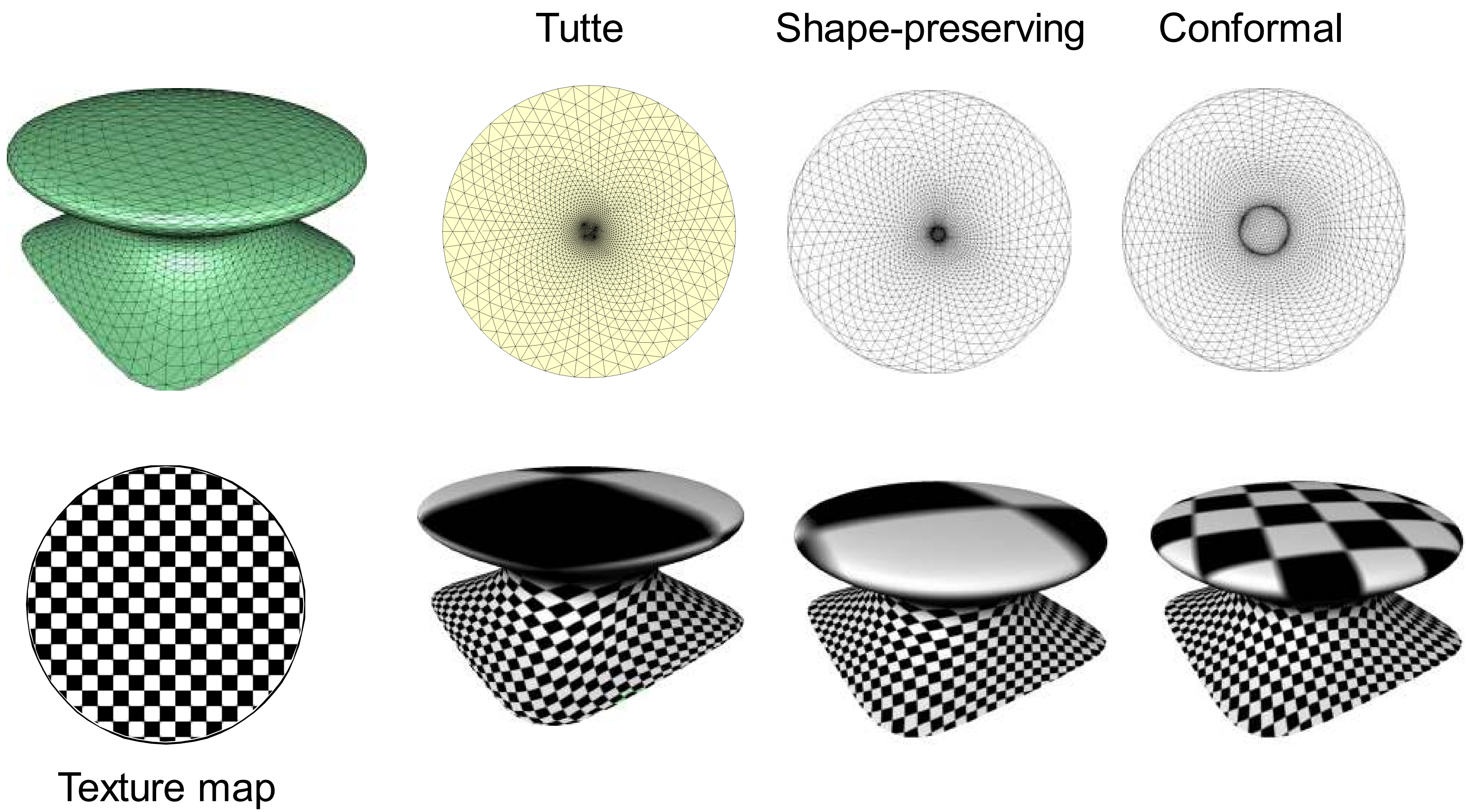


Not so good parameterization algorithm

Area distortion vs. angle distortion

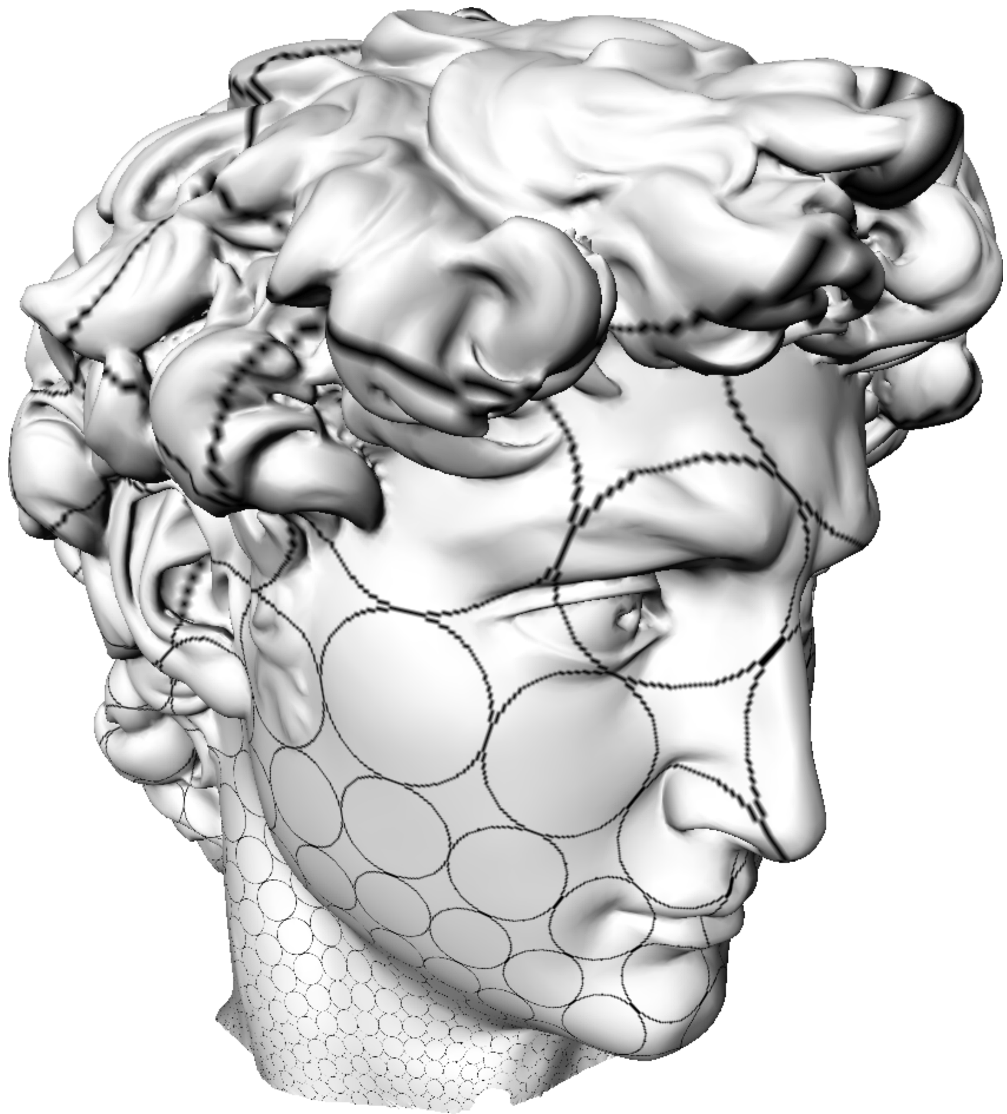




Conformal parameterization

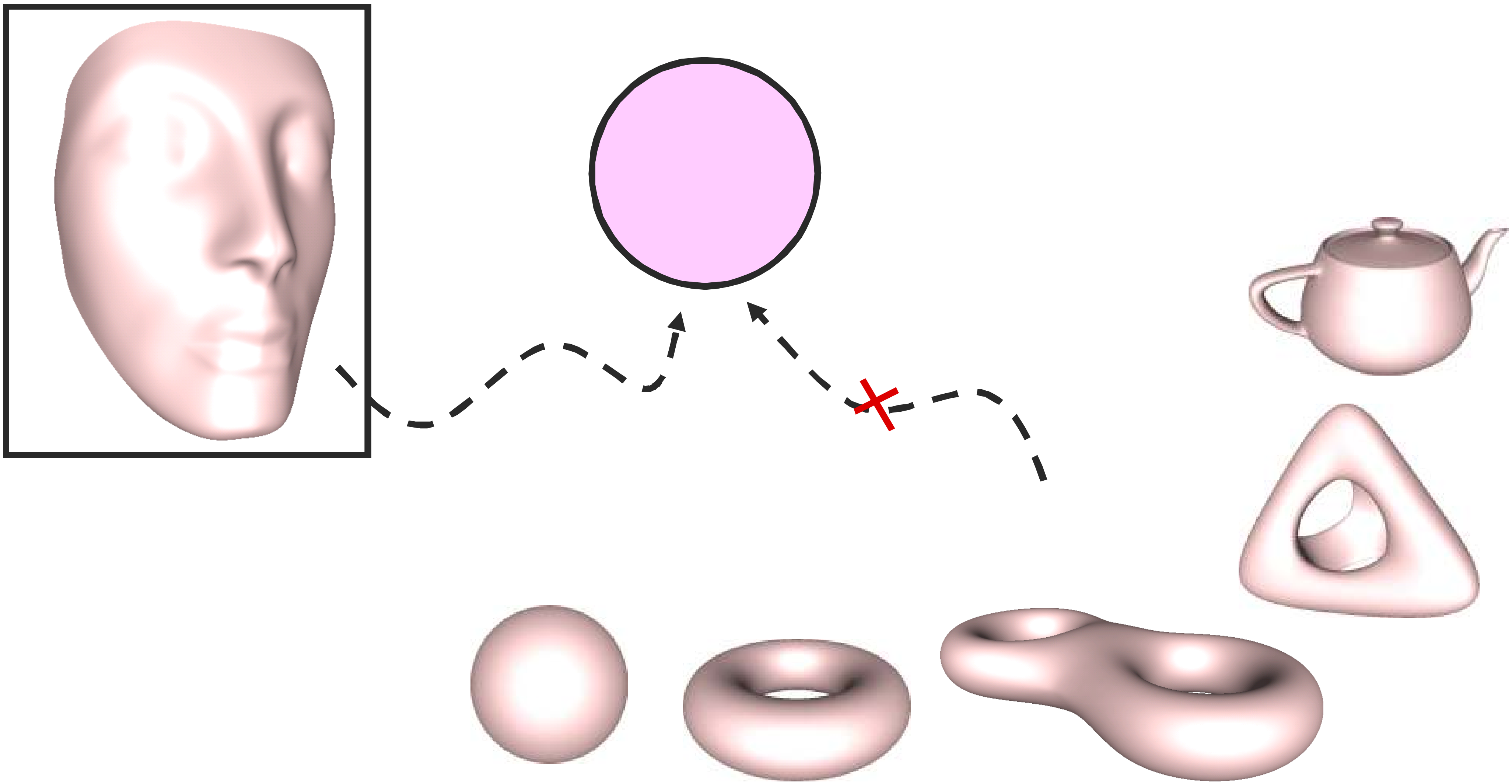


Conformal parameteriztion

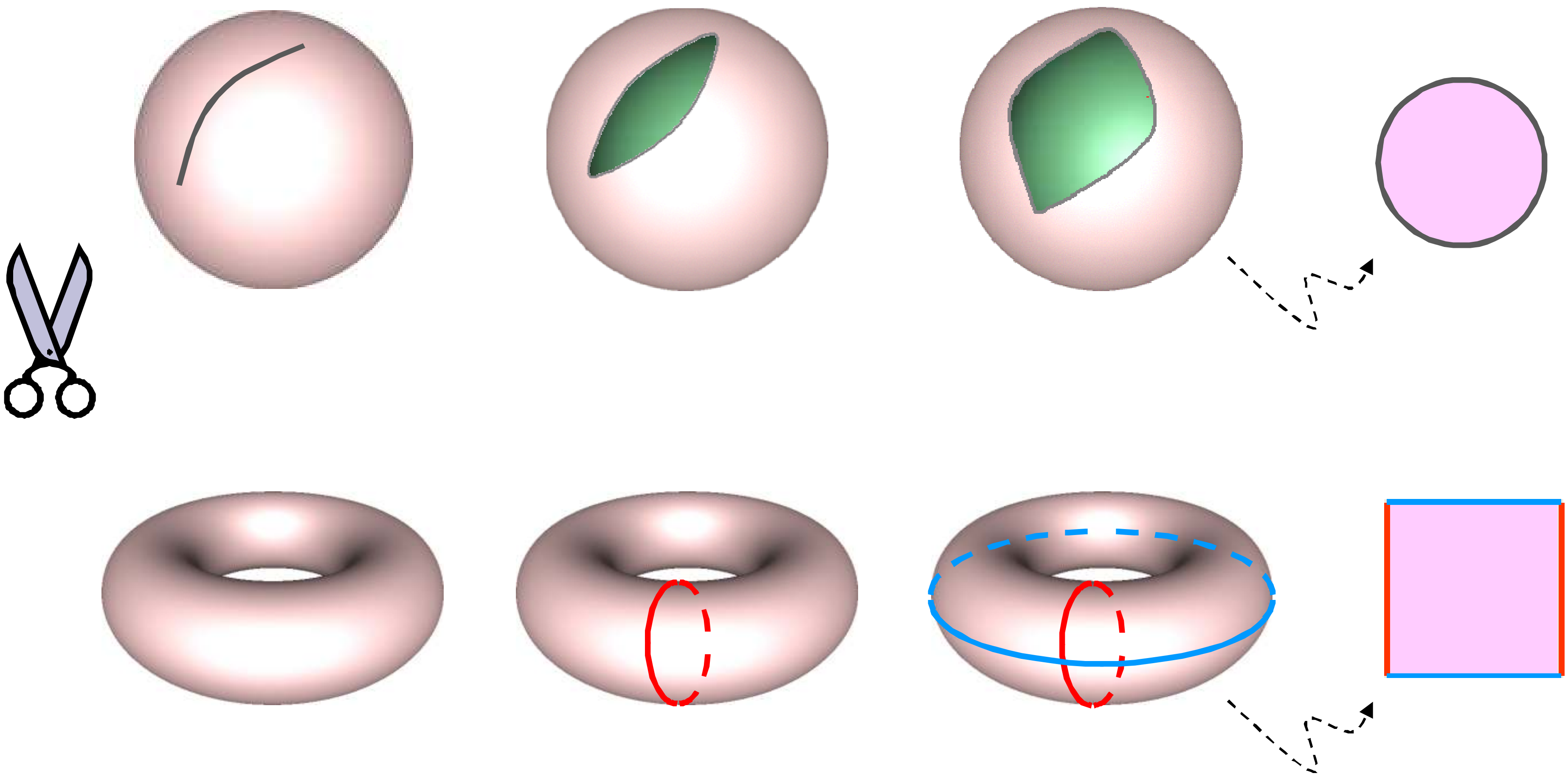
angle preservation; circles are mapped to circles



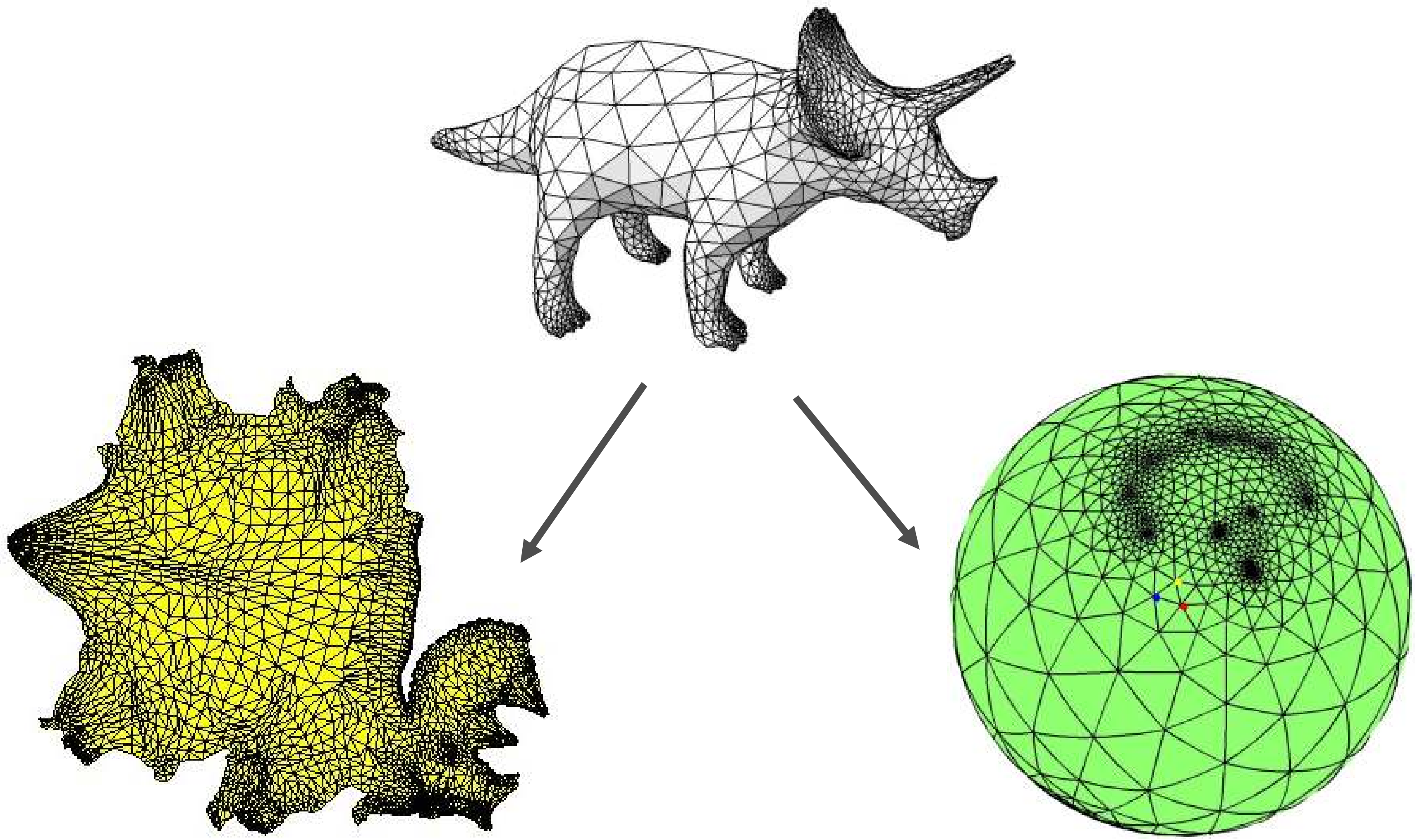
Non-disk domains



Cutting



Parameterization of closed genus-0 triangle meshes



Non-Constrained Planar

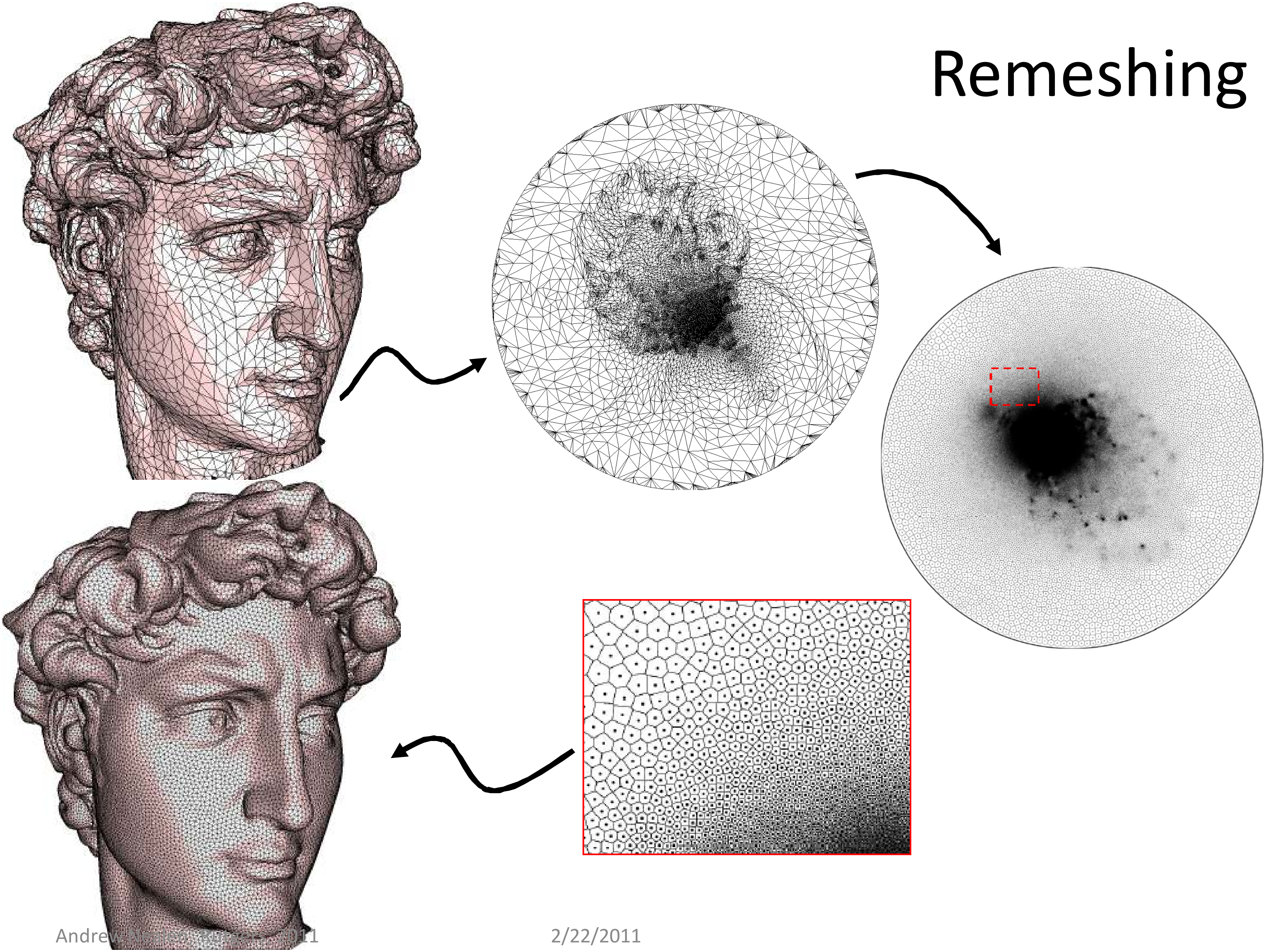
Spherical

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

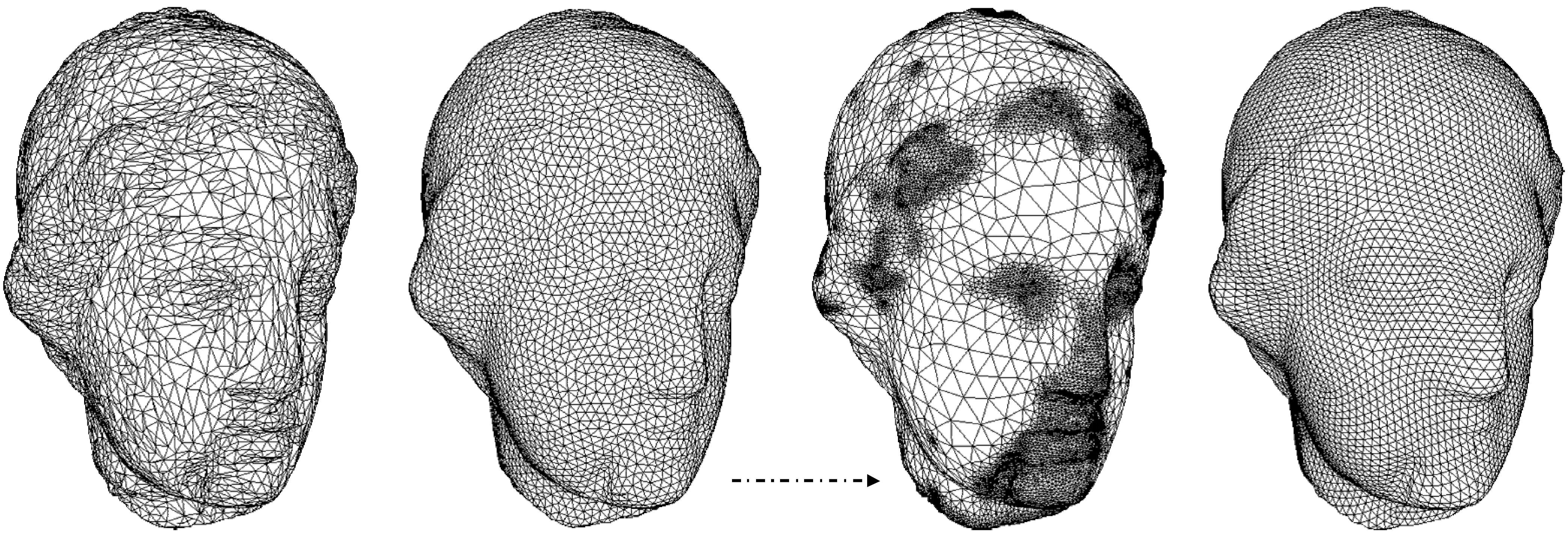
- Mesh parameterization allows to use some notions from continuous surface theory

Remeshing



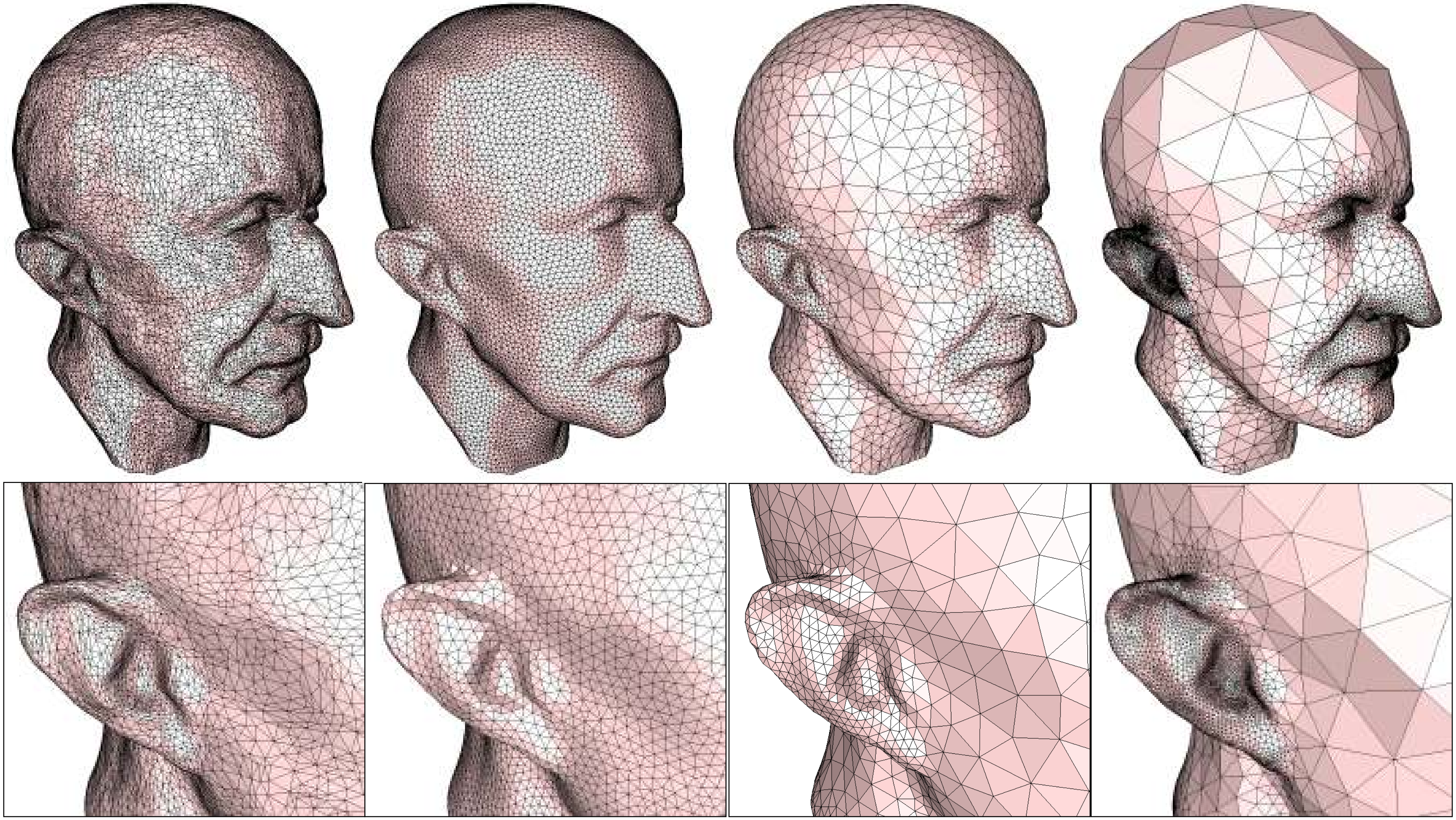
Remeshing

- Particular remeshing type according to application



original uniform adapted semi-regular

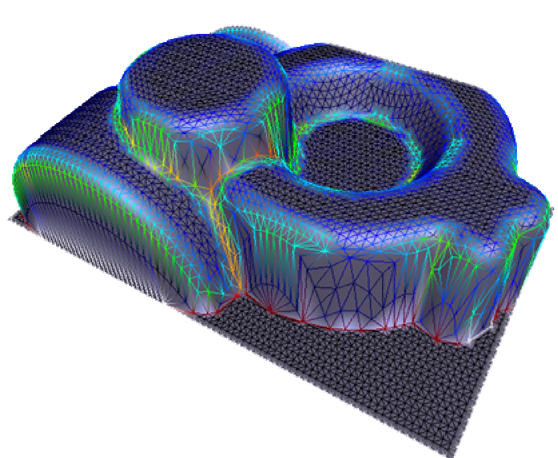
Remeshing examples



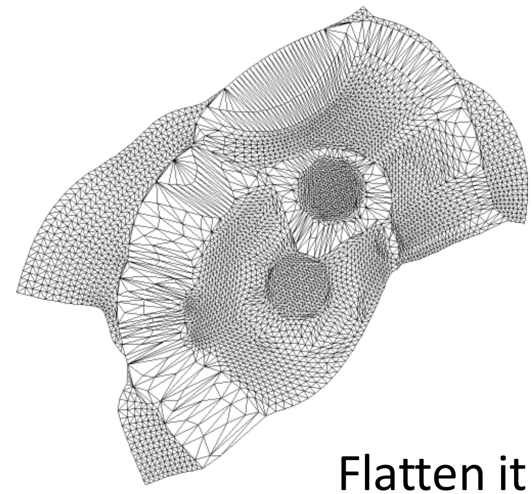
Interactive geometry remeshing

[Alliez et al., SIGGRAPH 2002]

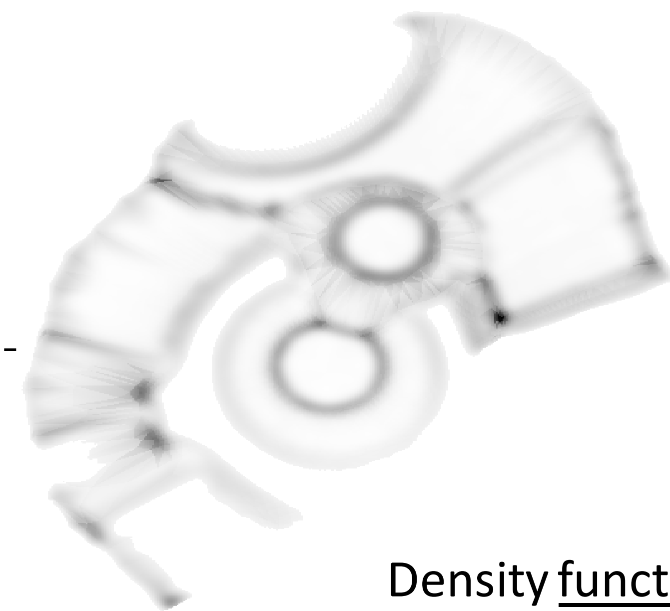
Model



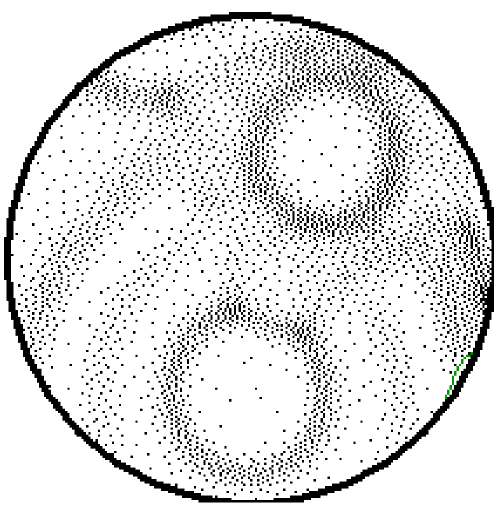
Measure curvature



Flatten it **conformally**



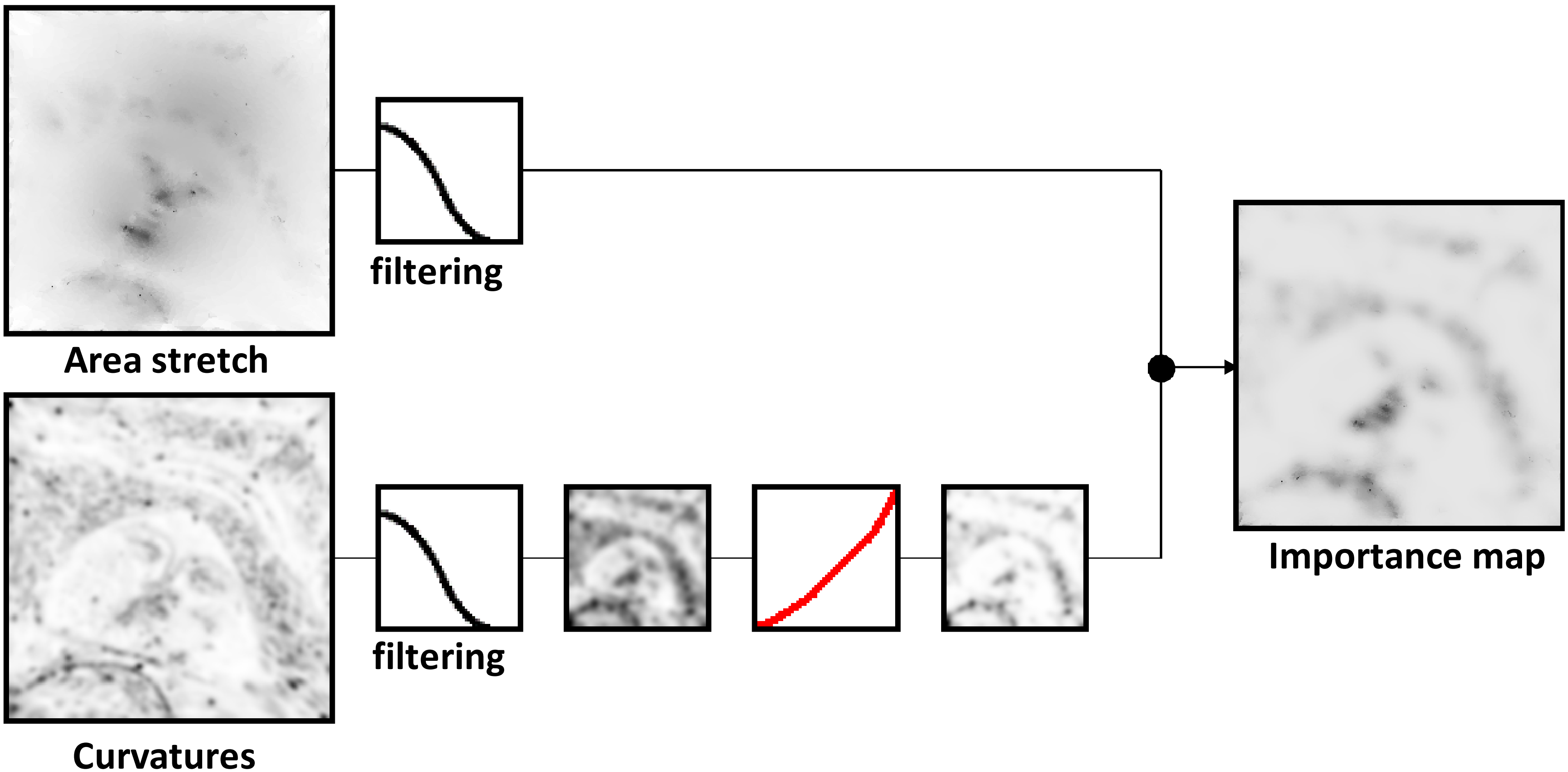
Density function in parameter space



Interactive geometry remeshing

[Alliez et al., SIGGRAPH 2002]

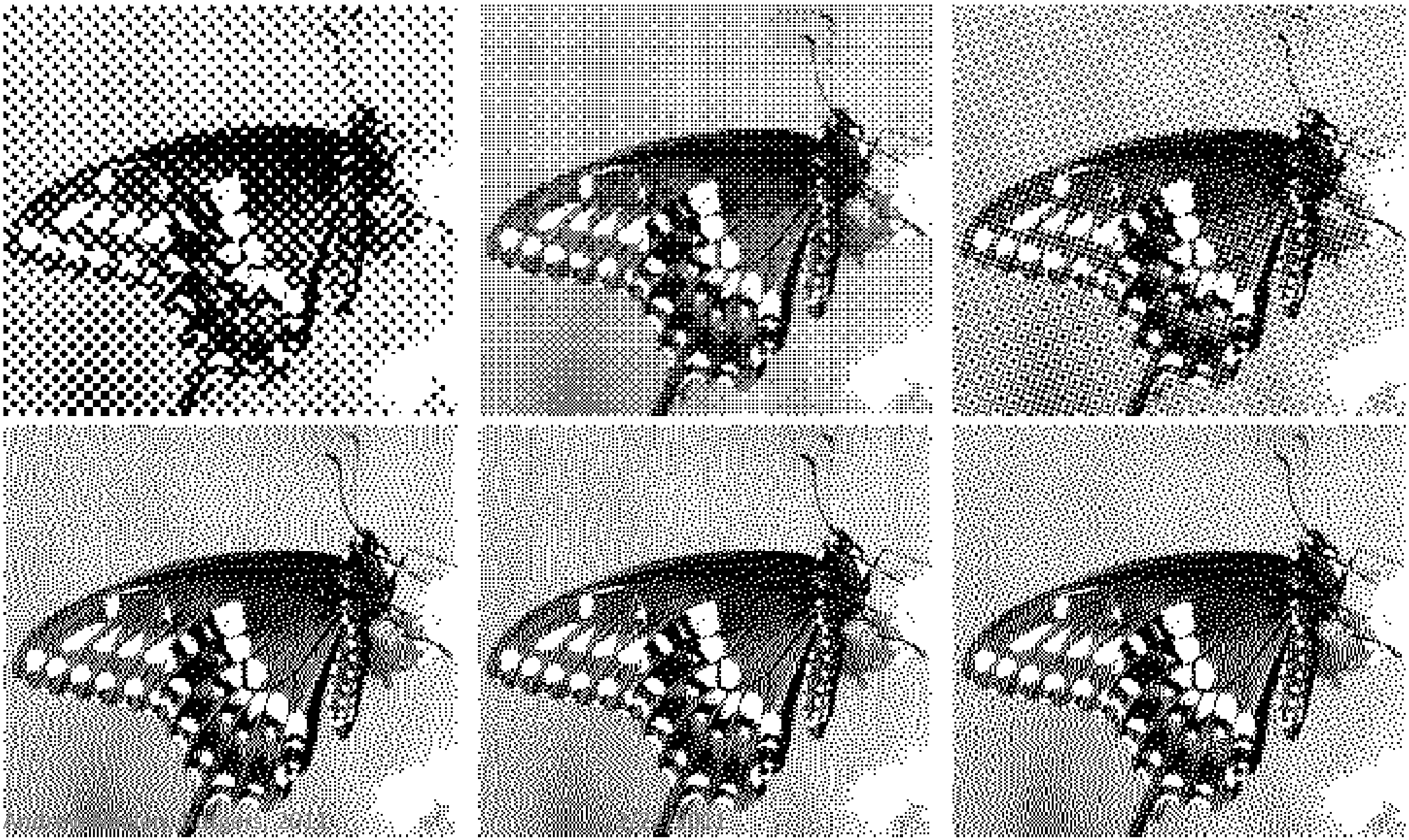
- Importance map created according to application needs



Interactive geometry remeshing

[Alliez et al., SIGGRAPH 2002]

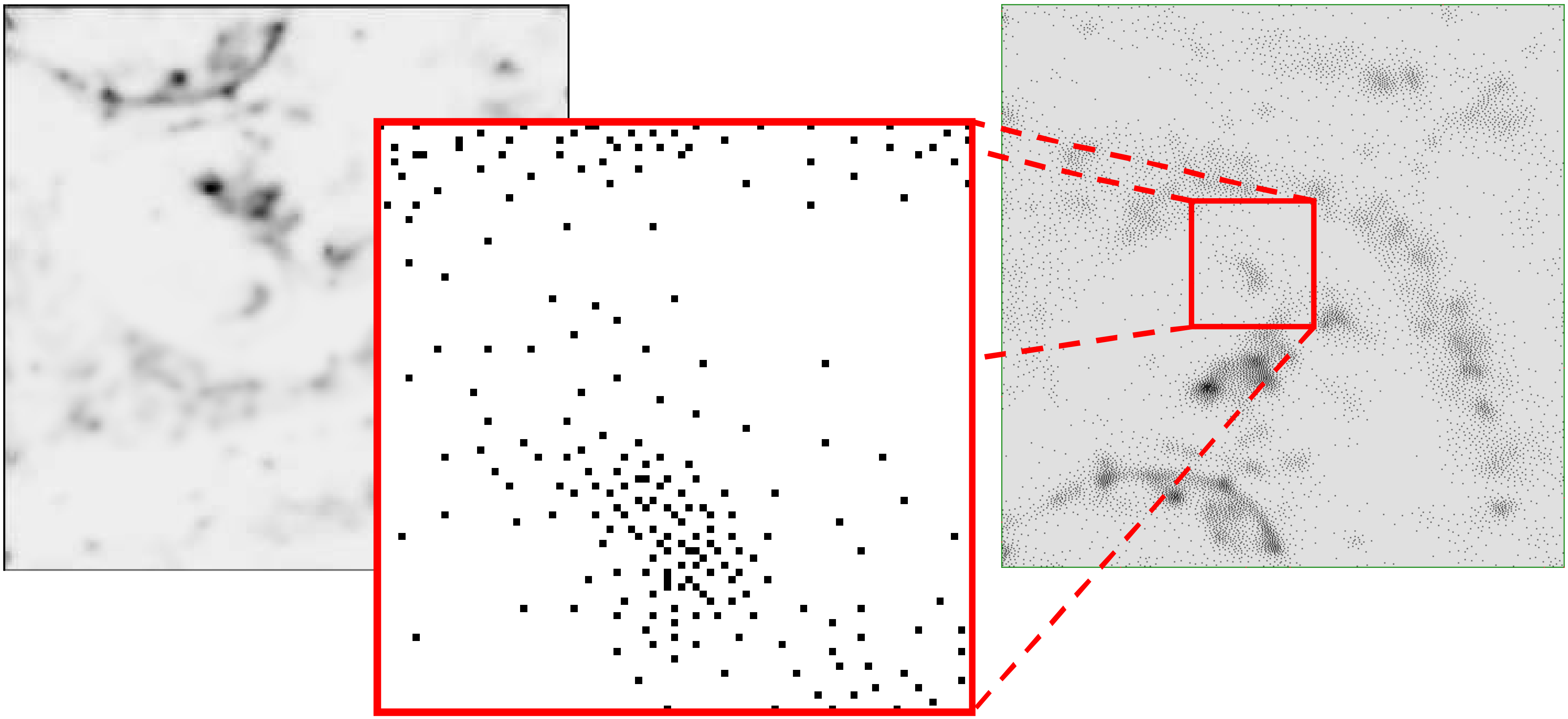
- Importance map is sampled by points – as in halftoning



Interactive geometry remeshing

[Alliez et al., SIGGRAPH 2002]

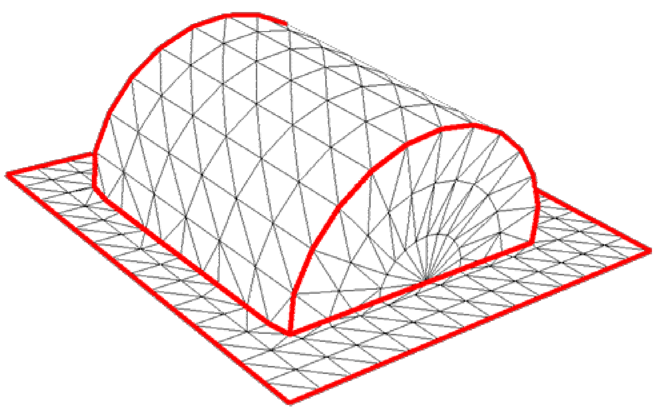
- Importance map is sampled by points – as in halftoning (error diffusion process)

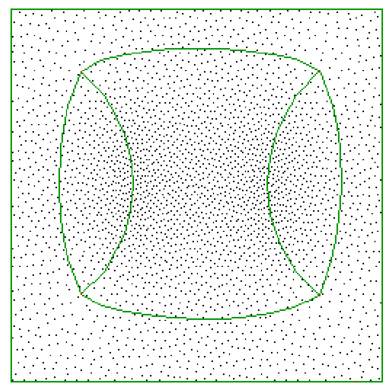


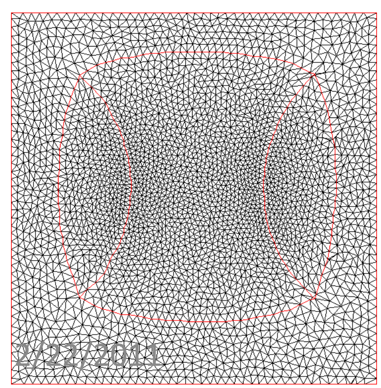
Interactive geometry remeshing

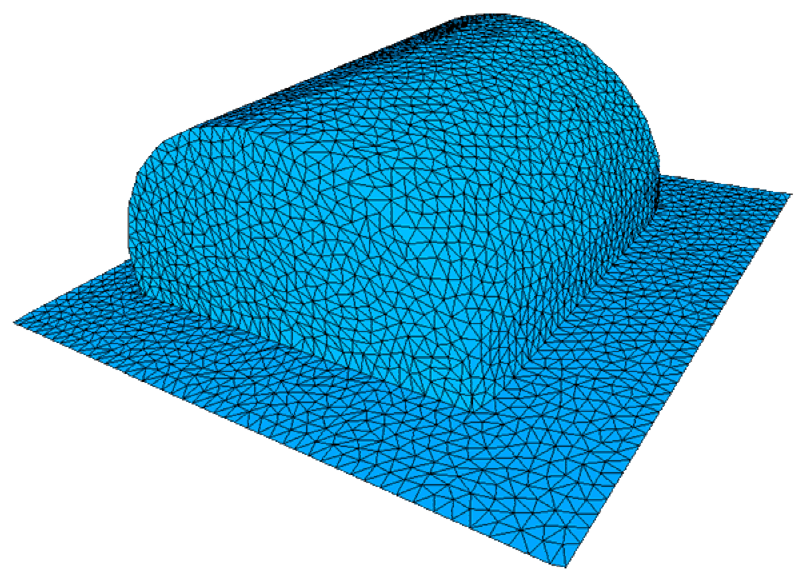
[Alliez et al., SIGGRAPH 2002]

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



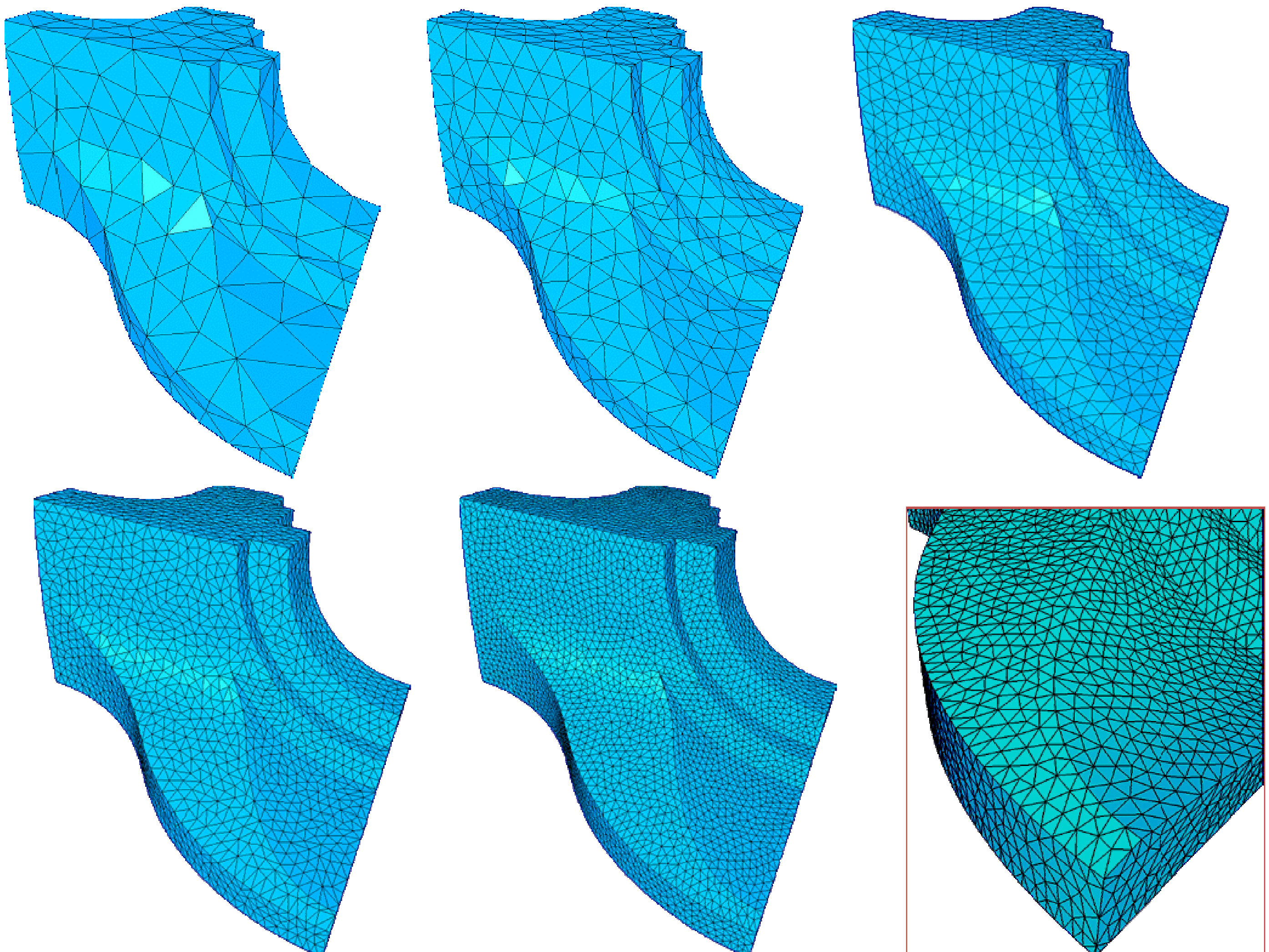






Interactive geometry remeshing
[Alliez et al., SIGGRAPH 2002]

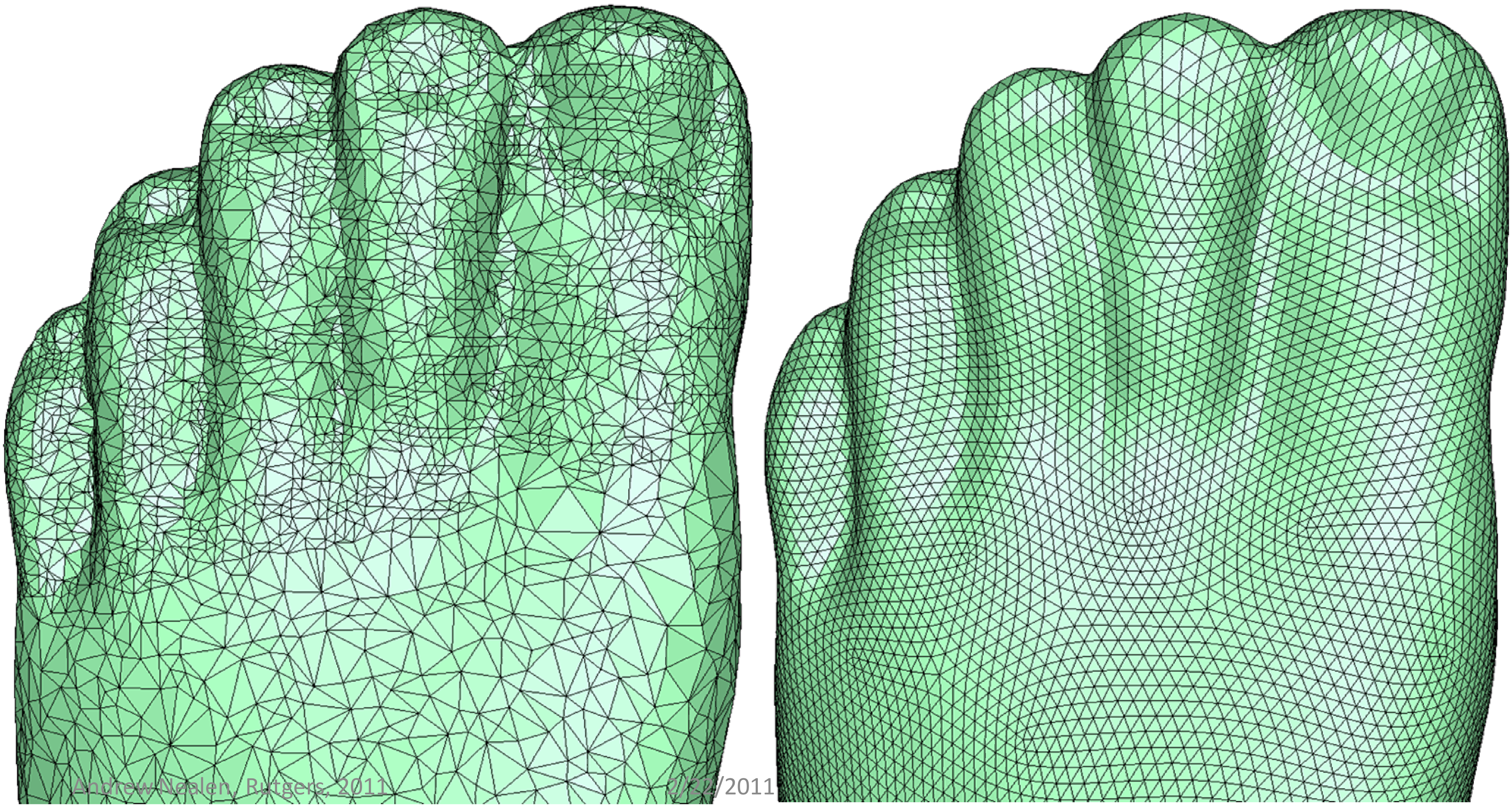
- More results



Interactive geometry remeshing

[Alliez et al., SIGGRAPH 2002]

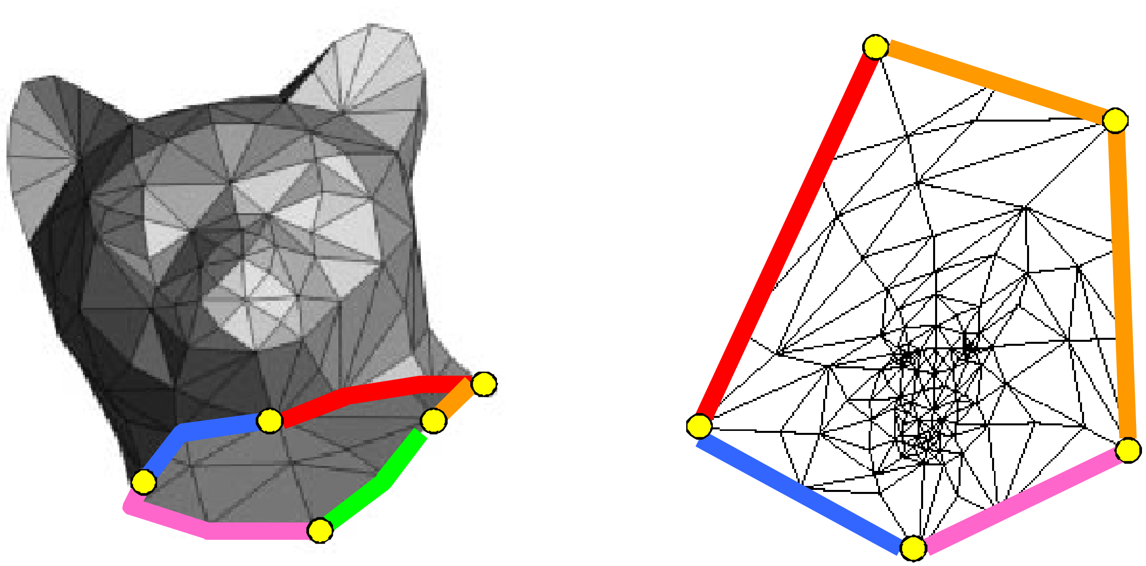
- More results



Computing parameterizations

Convex mapping (Tutte, Floater)

- Works for meshes equivalent to a disk
- First, we map the boundary to a convex polygon
- Then we find the inner vertices positions



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

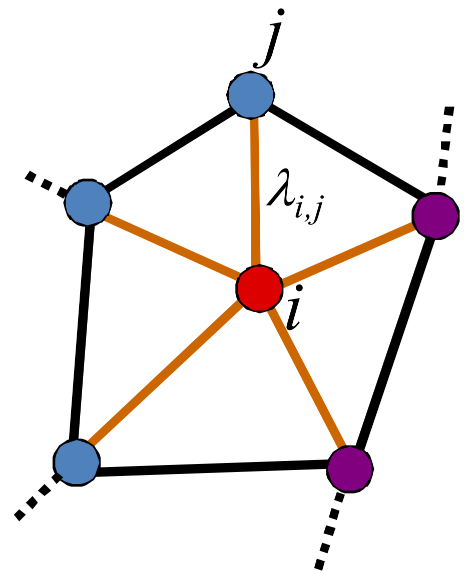
Inner vertices

- We constrain each inner vertex to be a weighted average of its neighbors:

$$\boxed{\mathbf{v}_i = \sum_{j \in \mathrm{N}(i)} \lambda_{i,j} \mathbf{v}_j, \quad i = 1, 2, \ldots, \mathrm{n}}$$

$$\lambda_{i,j} = \begin{cases} 0 & i, j \text{ are not neighbors} \\ > 0 & (i, j) \in E \text{ (neighbours)} \end{cases}$$

$$\sum_{j \in N(i)} \lambda_{i,j} = 1$$



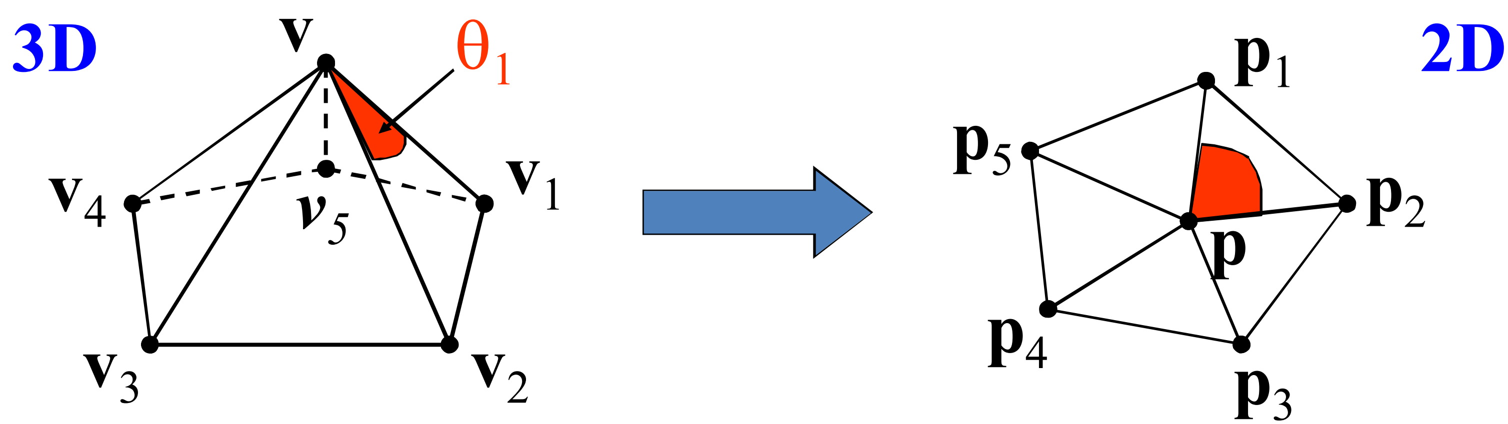
Linear system of equations

$$\mathbf{v}_i - \sum_{j \in \mathrm{N}(i)} \lambda_{i,j} \mathbf{v}_j = 0, \qquad i = 1, 2, \ldots, \mathrm{n}$$

$$\mathbf{v}_i - \sum_{j \in \mathrm{N}(i) \setminus \mathrm{B}} \lambda_{i,j} \mathbf{v}_j = \sum_{k \in \mathrm{N}(i) \cap \mathrm{B}} \lambda_{i,k} \mathbf{v}_k, \quad i = 1, 2, \ldots, \mathrm{n}$$

$$\begin{pmatrix} 1 & & -\lambda_{1,j_1} & & -\lambda_{1,j_{d1}} \\ & 1 & & & \\ & & 1 & & \\ & -\lambda_{4,j_1} & & \ddots & \\ & & -\lambda_{\mathrm{n},j_5} & & 1 \end{pmatrix} \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \\ \vdots \\ \mathbf{v}_\mathrm{n} \end{pmatrix} = \begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \\ \\ \sigma_\mathrm{n} \end{pmatrix}$$

Shape preserving weights



To compute $\lambda_1, \ldots, \lambda_5$, a local embedding of the patch is found:

1) $\| \mathbf{p}_j - \mathbf{p} \| = \| \mathbf{v}_j - \mathbf{v} \|$

2) $angle(\mathbf{p}_j, \mathbf{p}, \mathbf{p}_{j+1}) = (2\pi / \Sigma \theta_j)\, angle(\mathbf{v}_j, \mathbf{v}, \mathbf{v}_{j+1})$

$$\exists \lambda_i, \quad \begin{cases} \mathbf{p} = \Sigma \lambda_i \mathbf{p}_i \\ \lambda_i > 0 \\ \Sigma \lambda_i = 1 \end{cases}$$

$\Rightarrow$ use these λ as edge weights.

Linear system of equations

- A unique solution always exists
- Important: the solution is legal (bijective). The proof is not trivial.

- The system is sparse, thus fast numerical solution is possible
- Numerical problems (because the vertices in the middle might get very dense…)

Conformal mapping

Also called harmonic

- Another way to find inner vertices
- Strives to preserve angles (conformal)
- We treat the mesh as a system of springs.
- Define spring energy:

$$\boxed{\mathrm{E}_{\mathrm{harm}} = \frac{1}{2}\sum_{(i,j)\in \mathrm{E}} k_{i,j}\left\|\mathbf{v}_i - \mathbf{v}_j\right\|^2}$$

where $\mathbf{v}_i$ are the flat position (remember that the boundary vertices $\mathbf{v}_n$, $\mathbf{v}_{n+1}$, …, $\mathbf{v}_N$ are constrained).

Energy minimization – least squares

- We want to find flat positions that minimize the energy.
- Solve the linear least squares problem!

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

$$\mathrm{E}_{\mathrm{harm}}(x_1,\ldots,x_n,y_1,\ldots,y_n) = \frac{1}{2}\sum_{(\mathrm{i},\mathrm{j})\in \mathrm{E}} k_{i,j}\left\|\mathbf{v}_i - \mathbf{v}_j\right\|^2 =$$

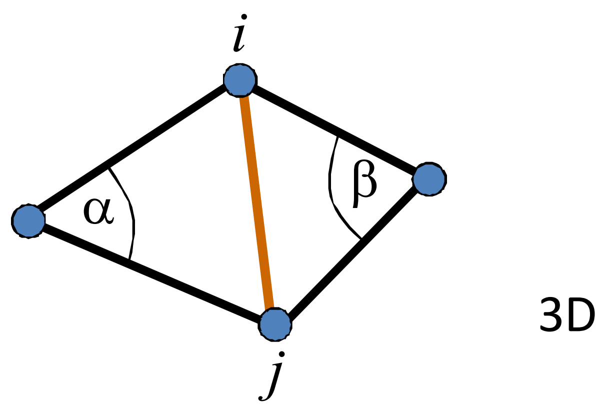
$$= \frac{1}{2}\sum_{(i,j)\in \mathrm{E}} k_{i,j}\left((x_i - x_j)^2 + (y_i - y_j)^2\right).$$

E_{harm} is function of $2\mathrm{n}$ variables

The spring constants $k_{i,j}$

- The weights $k_{i,j}$ are chosen to minimize angle distortion:

$$\boxed{k_{i,j} = \cot\alpha + \cot\beta}$$



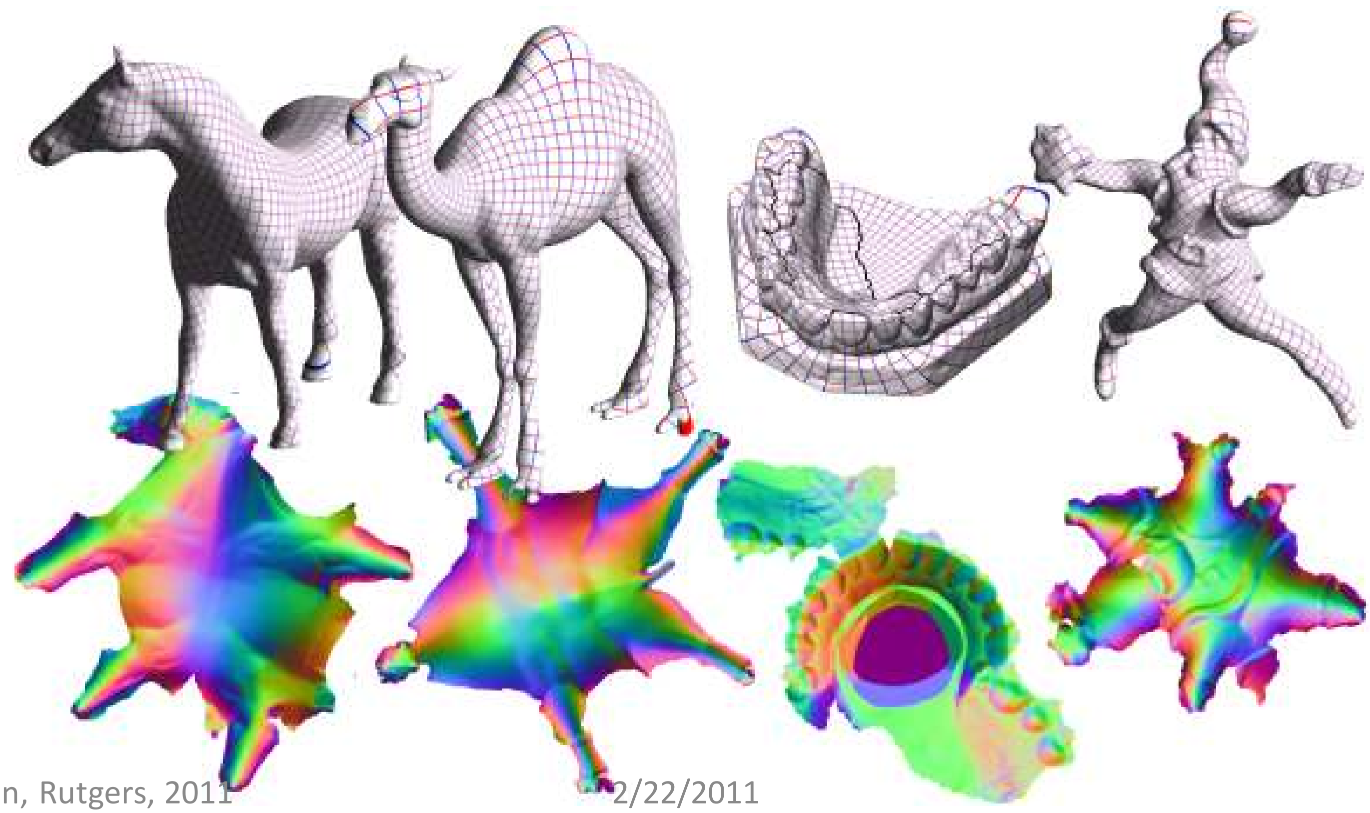
- The matrix of the normal equations is the **cotan Laplacian** (without area weighting)

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.