

Geometric Processing

Xiao-Ming Fu

USTC, 2020

Me

- ◆ Name: Xiao-Ming Fu, 傅孝明
- ◆ Office: Room 1207, Management and Research Building
- ◆ Email: fluxm@ustc.edu.cn
- ◆ Personal website: <http://staff.ustc.edu.cn/~fluxm/>

Goal

- ◇ Basic knowledge about polygon mesh processing
- ◇ **Basic coding training**

Reference

- ◇ Book: Polygon Mesh Processing
- ◇ Paper: <http://kesen.realtimerendering.com/>
- ◇ Other related papers and books

Score

- ◇ Coding:

- ◇ 14 tasks: $5 \times 14 = 70$

- ◇ Code presentation: 10

- ◇ Quiz: 10

- ◇ Exam: 10

- ◇ Total score: $70 + 10 + 10 + 10 = 100$

Homework

- ◇ <http://staff.ustc.edu.cn/~fuxm/>
- ◇ http://staff.ustc.edu.cn/~fuxm/course/2020_Spring_DGP/index.html
- ◇ Email: fire_fuxm@qq.com
- ◇ Coding: 作业编号_姓名_学号.zip
 - ◇ For example: HW1_Xiaoming_Fu_SA16001062.zip

Representations

Xiao-Ming Fu

Outline

- ◇ Point cloud
- ◇ Signed distance field
- ◇ Implicit function
- ◇ Grid
 - ◇ Pixel, Voxel
 - ◇ Quad-tree, Octree
- ◇ Mesh
 - ◇ Triangle, Tetrahedron
 - ◇ Data structure
 - ◇ Halfedge
 - ◇ File format
 - ◇ Obj, Off

Outline

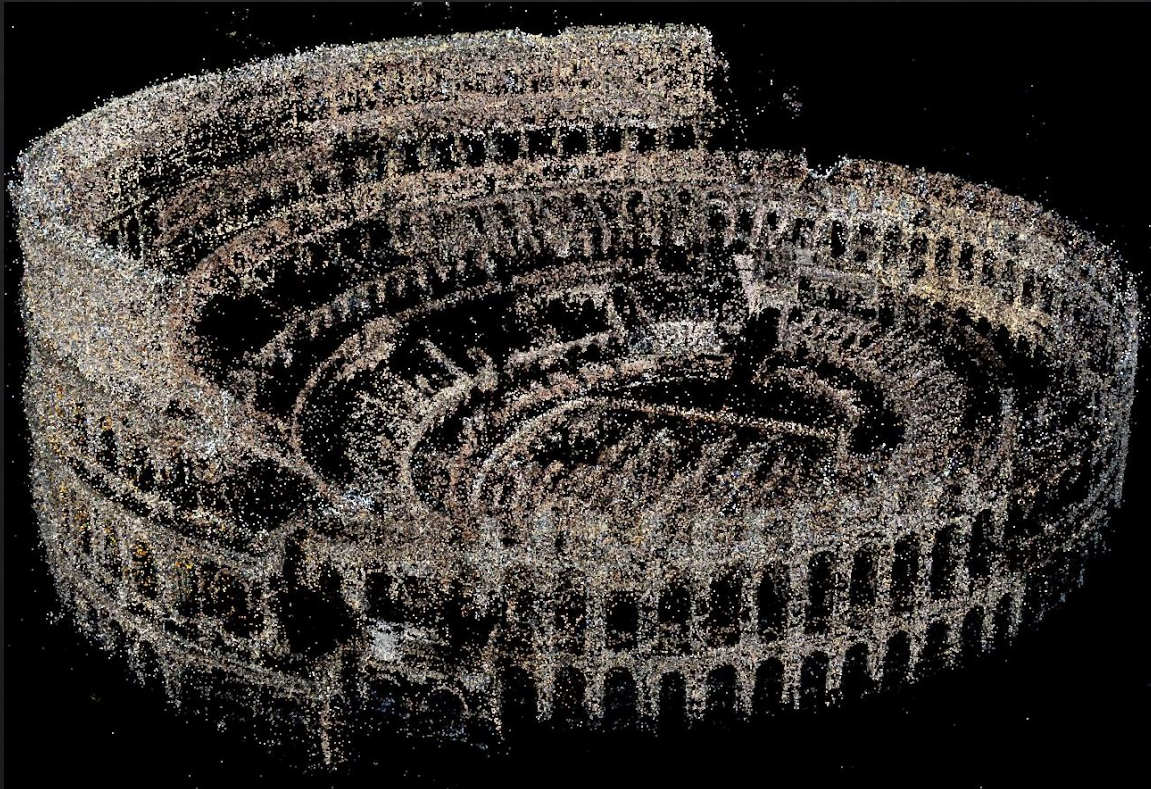
- ◇ Point cloud
- ◇ Signed distance field
- ◇ Implicit function
- ◇ Grid
 - ◇ Pixel, Voxel
 - ◇ Quad-tree, Octree
- ◇ Mesh
 - ◇ Triangle, Tetrahedron
 - ◇ Data structure
 - ◇ Halfedge
 - ◇ File format
 - ◇ Obj, Off

Point cloud

$$\mathcal{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_{N_v}\}, \mathbf{p}_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \in \mathcal{R}^3$$

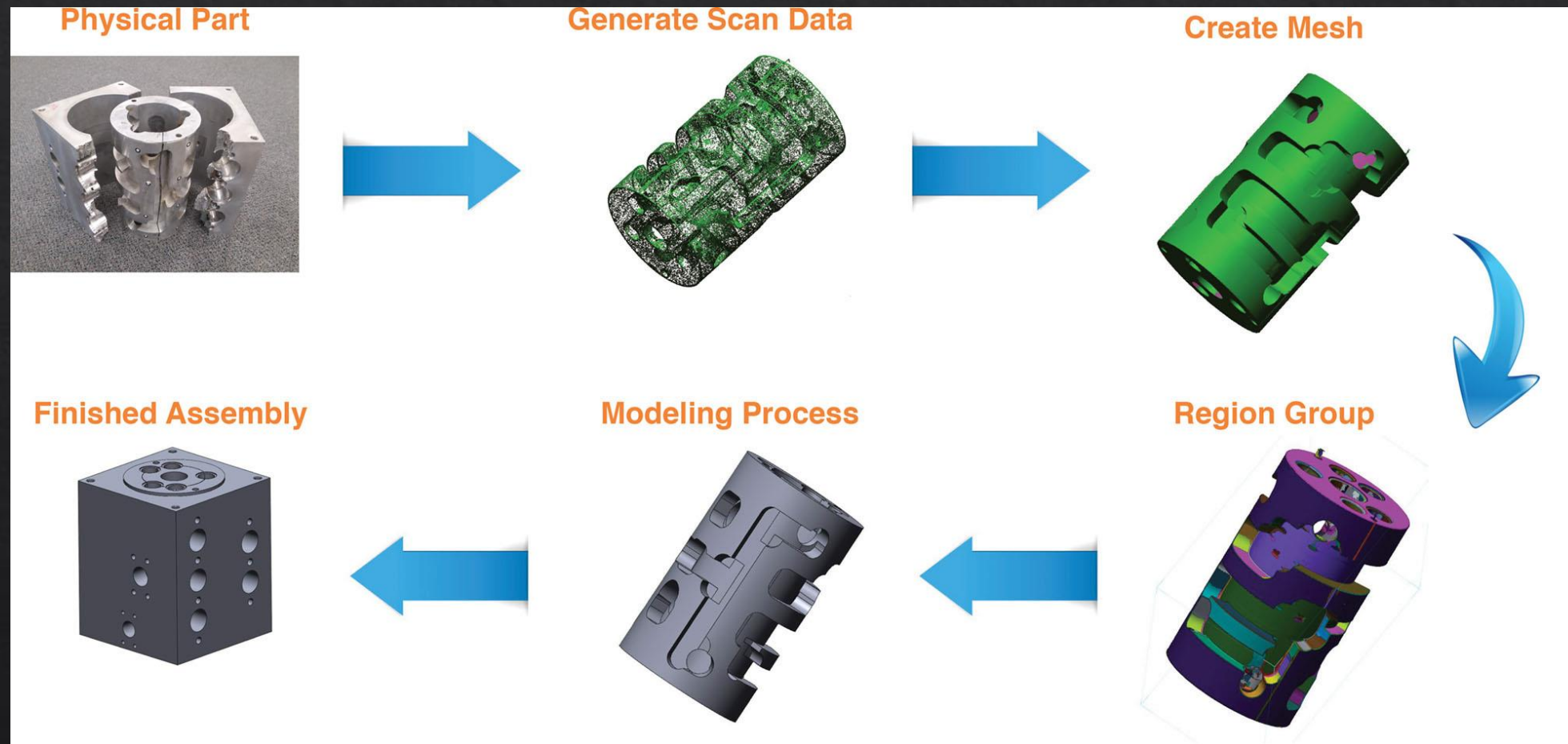
A set of data points in some coordinate system

3D scanners



Applications - scanners

◇ Reverse engineering



Applications - scanners

- ◆ Digital preservation of cultural heritage sites and objects



Depth camera

- ◆ Depth camera: Kinect, RealSense, iPhone X,



Depth camera

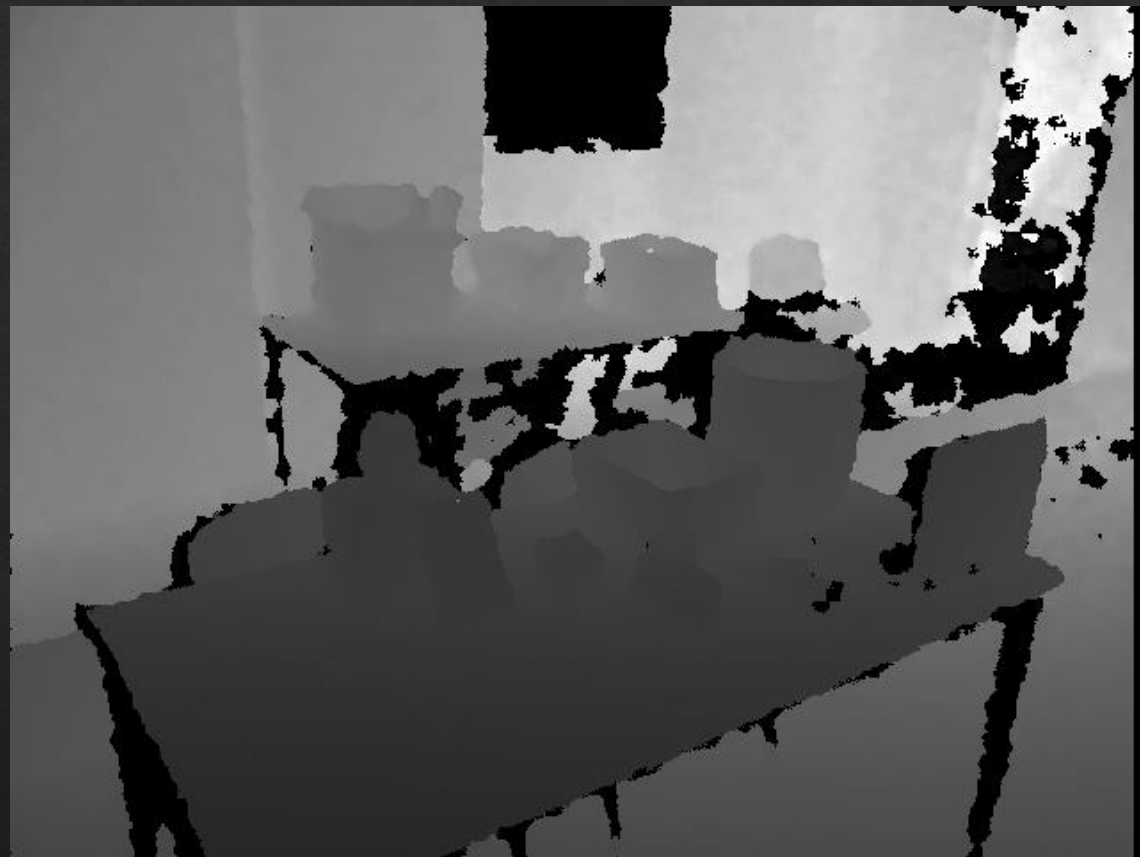
True Depth Camera



Depth Image



Gray image: pixel represents distance from camera.
Nearer is brighter.



Real capture by Kinect.
Nearer is darker.

Outline

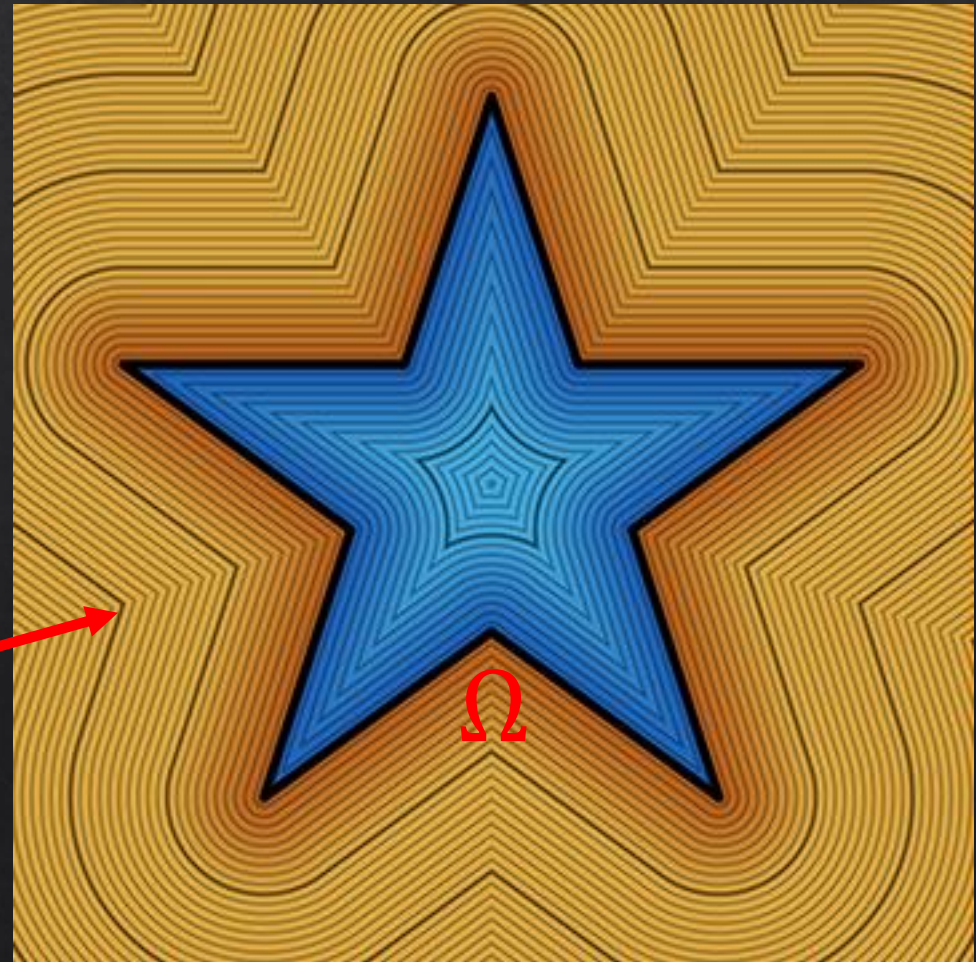
- ◇ Point cloud
- ◇ Signed distance field
- ◇ Implicit function
- ◇ Grid
 - ◇ Pixel, Voxel
 - ◇ Quad-tree, Octree
- ◇ Mesh
 - ◇ Triangle, Tetrahedron
 - ◇ Data structure
 - ◇ Halfedge
 - ◇ File format
 - ◇ Obj, Off

Signed distance field

- ◇ The distance of a given point x from the boundary of Ω :

$$f(x) = \begin{cases} -d(x, \partial\Omega), & \text{if } x \in \Omega \\ d(x, \partial\Omega), & \text{if } x \in \Omega^c \end{cases}$$

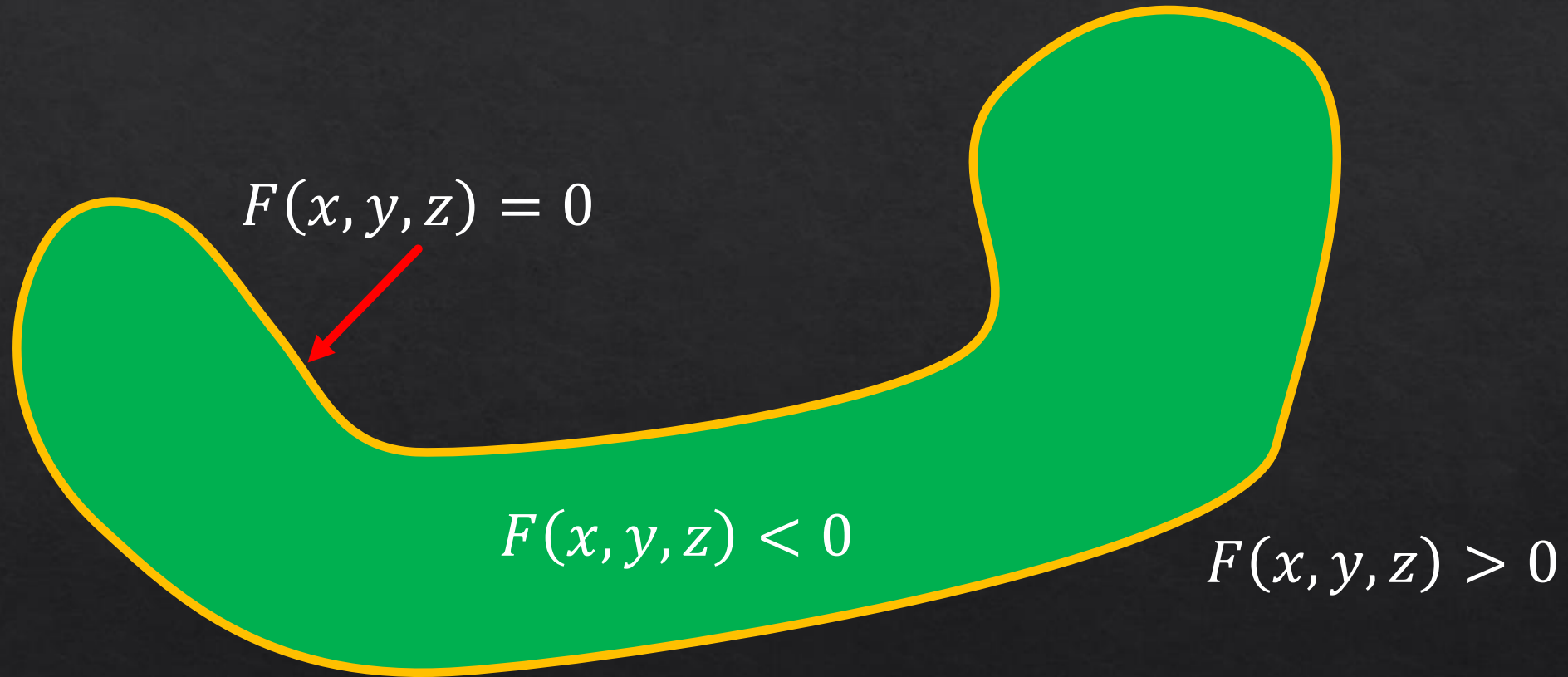
Contour line



Outline

- ◇ Point cloud
- ◇ Signed distance field
- ◇ **Implicit function**
- ◇ Grid
 - ◇ Pixel, Voxel
 - ◇ Quad-tree, Octree
- ◇ Mesh
 - ◇ Triangle, Tetrahedron
 - ◇ Data structure
 - ◇ Halfedge
 - ◇ File format
 - ◇ Obj, Off

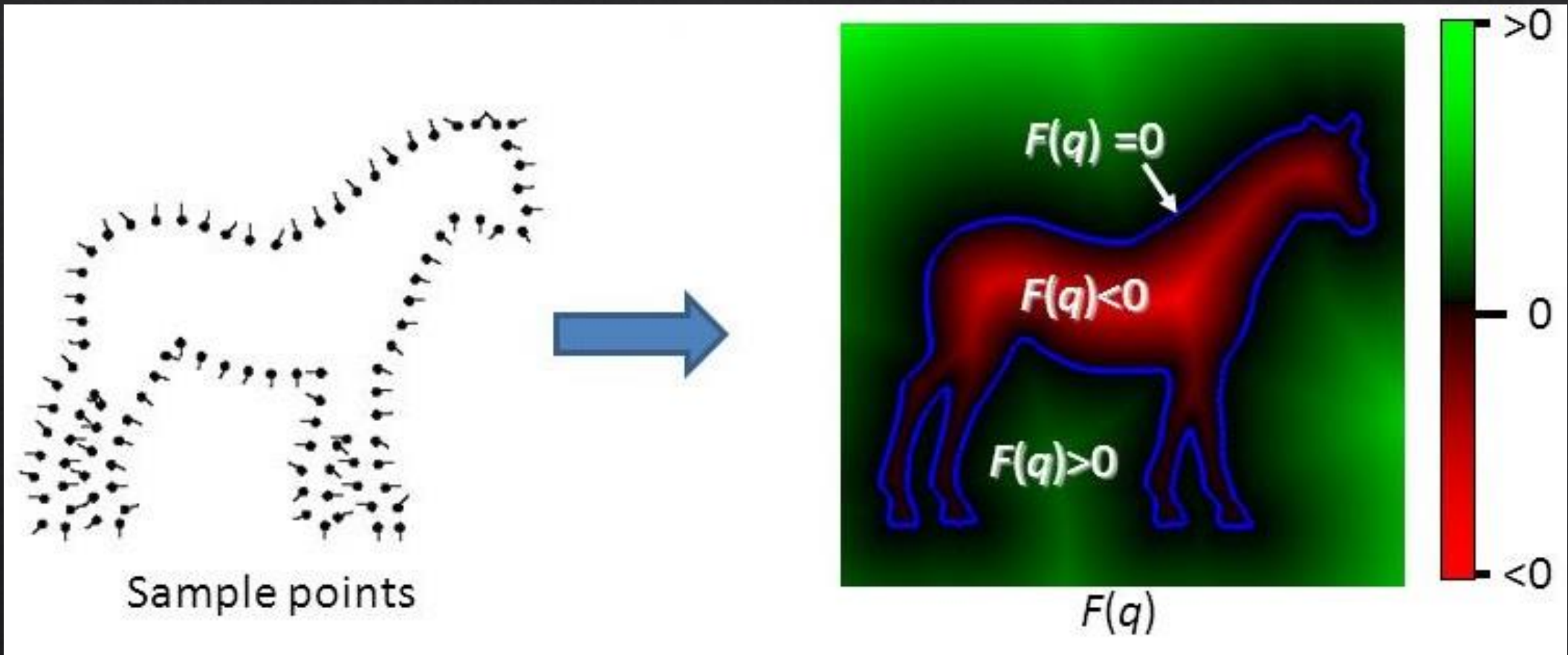
Implicit function



- ◆ Signed distance field is also an implicit function.

Implicit function

◆ Surface reconstruction



Outline

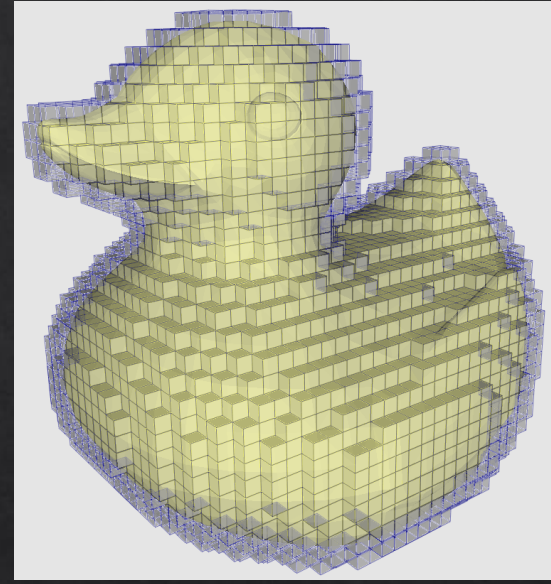
- ◇ Point cloud
- ◇ Signed distance field
- ◇ Implicit function
- ◇ **Grid**
 - ◇ Pixel, Voxel
 - ◇ Quad-tree, Octree
- ◇ Mesh
 - ◇ Triangle, Tetrahedron
 - ◇ Data structure
 - ◇ Halfedge
 - ◇ File format
 - ◇ Obj, Off

Grid

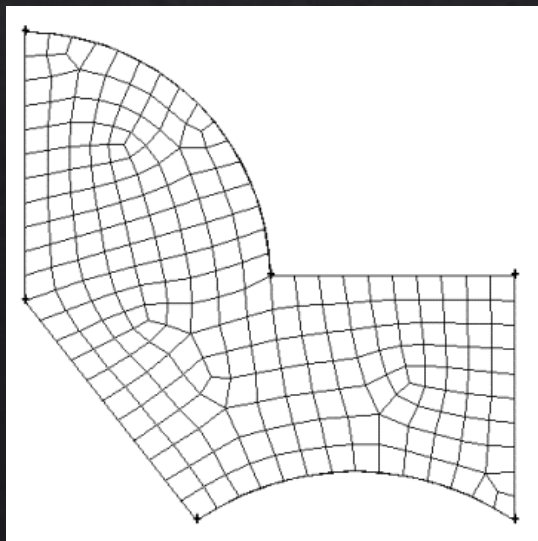
◇ Image



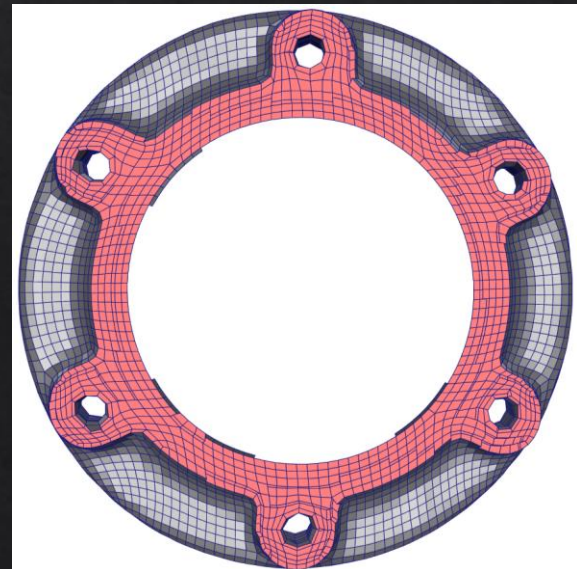
◇ Voxel



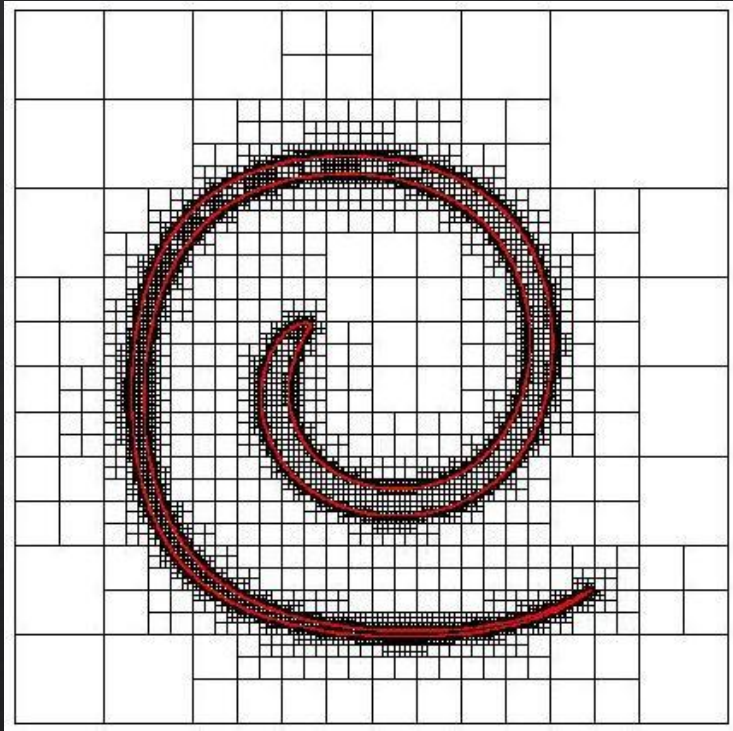
◇ Quad mesh



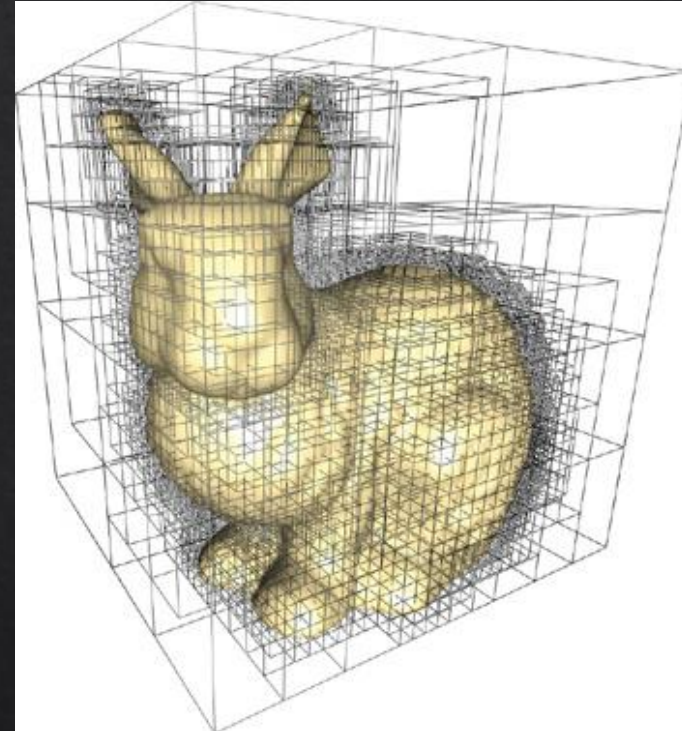
◇ All-Hex mesh



Adaptive grid - hierarchical octree



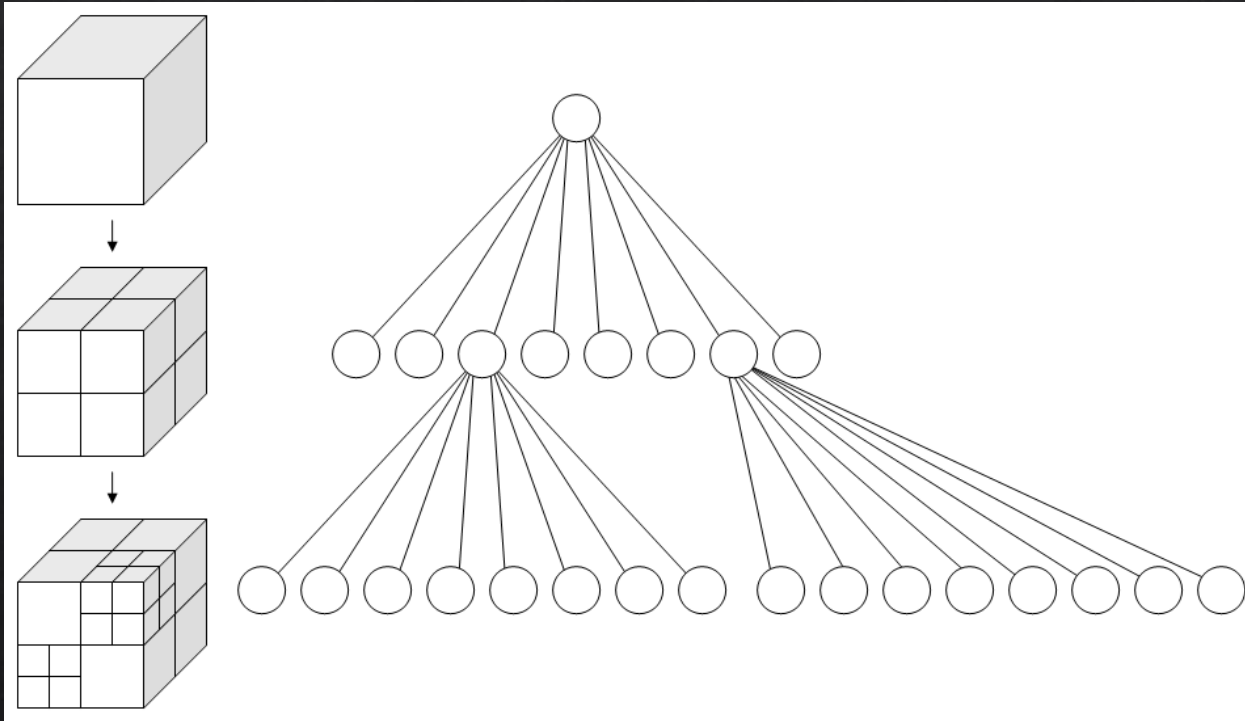
2D case



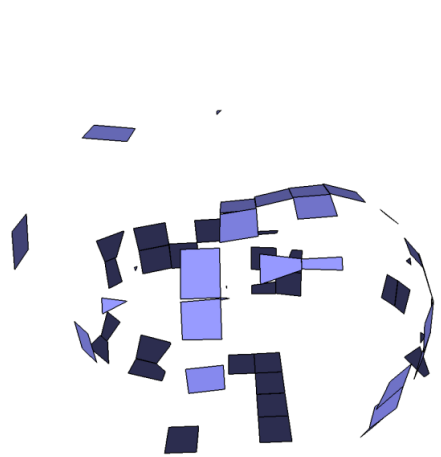
3D case

Partitioning rule

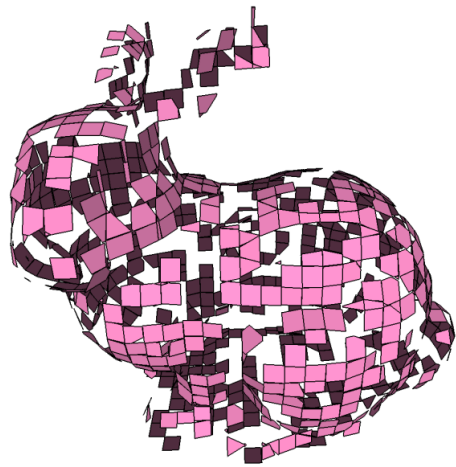
- ◇ The commonly-used octant partitioning depends on:
 - ◇ (1) the **existence** of the shape inside the octant
 - ◇ (2) the partitioning is performed until the **max tree depth** is reached.



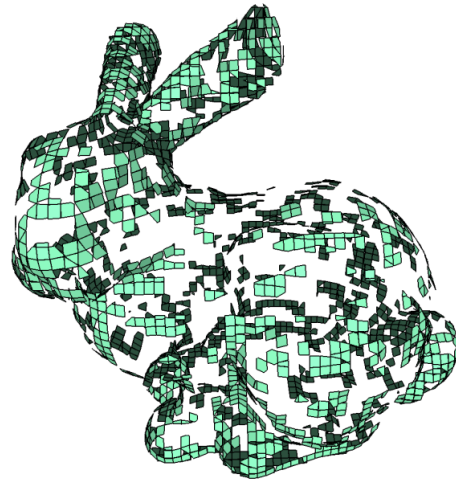
Patch-based octree



(a) 4th-level



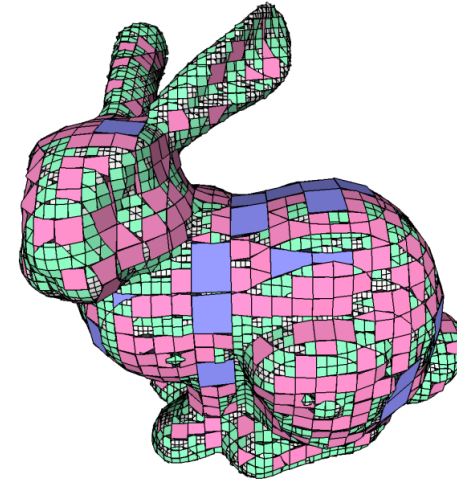
(b) 5th-level



(c) 6th-level



(d) 7th-level

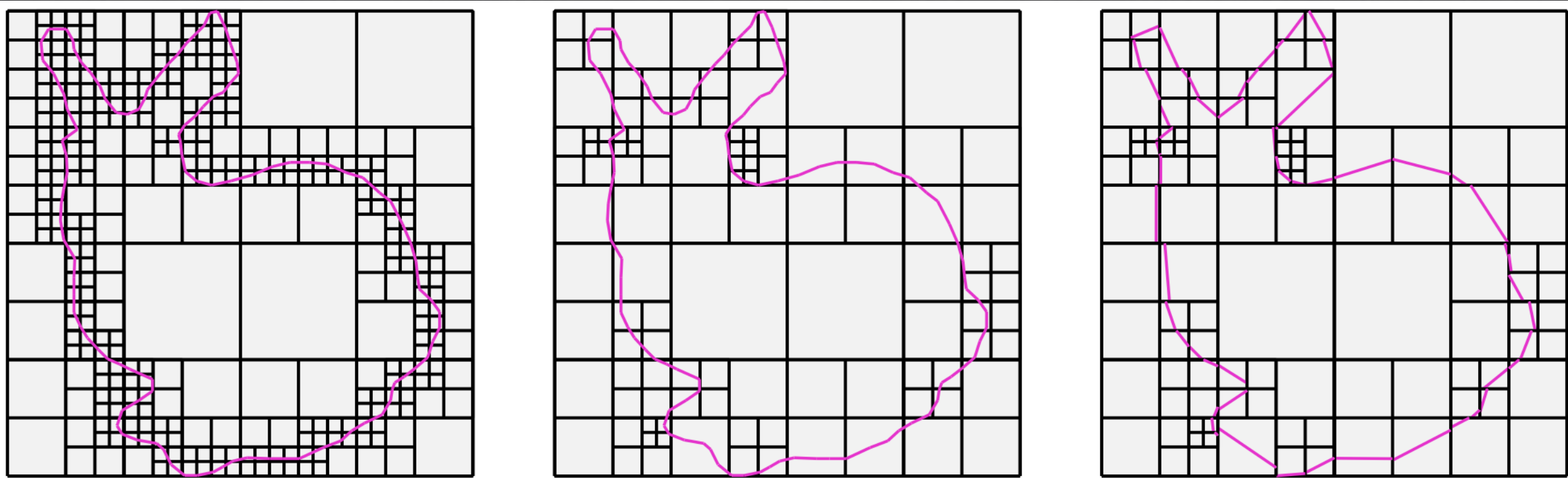


(e) all non-empty leaf nodes

The **partitioning rule** of the octree is:

For any octant O which is not at the max depth level, subdivide it if the local surface S_O restricted by it is not empty and the Hausdorff distance $d_H(S_O, P_O)$ larger than a predefined threshold.

Patch-based quadtree



Paper: Adaptive O-CNN: A Patch-based Deep Representation of 3D Shapes

Outline

- ◇ Point cloud
- ◇ Signed distance field
- ◇ Implicit function
- ◇ Grid
 - ◇ Pixel, Voxel
 - ◇ Quad-tree, Octree
- ◇ **Mesh**
 - ◇ Triangle, Tetrahedron
 - ◇ Data structure
 - ◇ Halfedge
 - ◇ File format
 - ◇ Obj, Off

Triangle Mesh

- ◆ A collection of triangles
 - ◆ without any particular mathematical structure
- ◆ Each triangle: a segment of a piecewise linear surface representation
- ◆ Geometric component
 - ◆ Discrete vertices: $V = \{v_1, \dots, v_{N_V}\}$
- ◆ Topological component
 - ◆ Triangle: $F = \{f_1, \dots, f_{N_F}\}$
 - ◆ Edge: $E = \{e_1, \dots, e_{N_E}\}$

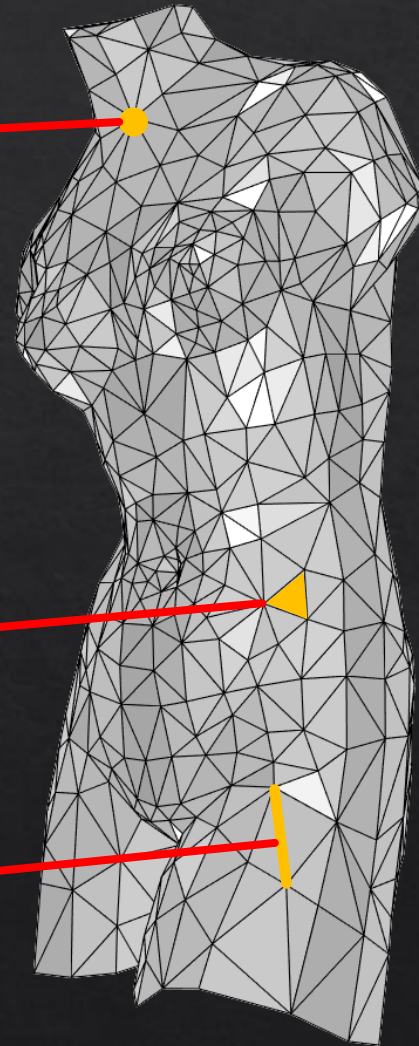
Triangle Mesh

$$\mathbf{p}_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \in \mathcal{R}^3$$

$$\mathcal{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_{N_v}\}$$

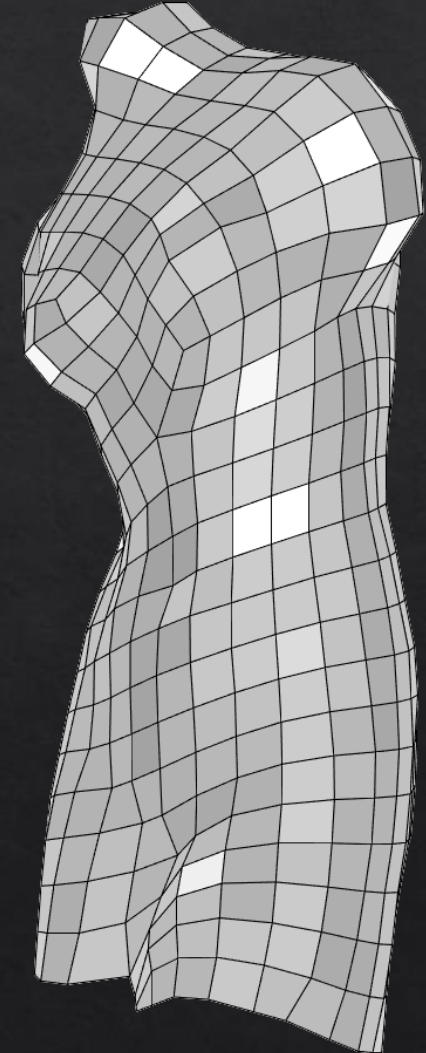
$$f_i: v_{i,1}, v_{i,2}, v_{i,3}$$

$$e_j: v_{j,1}, v_{j,2}$$



triangle mesh

Graph



quad mesh

Homework 1

◇ Shortest path

- ◇ Input: two vertices
- ◇ Output: a edge path connecting the input two vertices with shortest length
- ◇ How about the geodesic path?

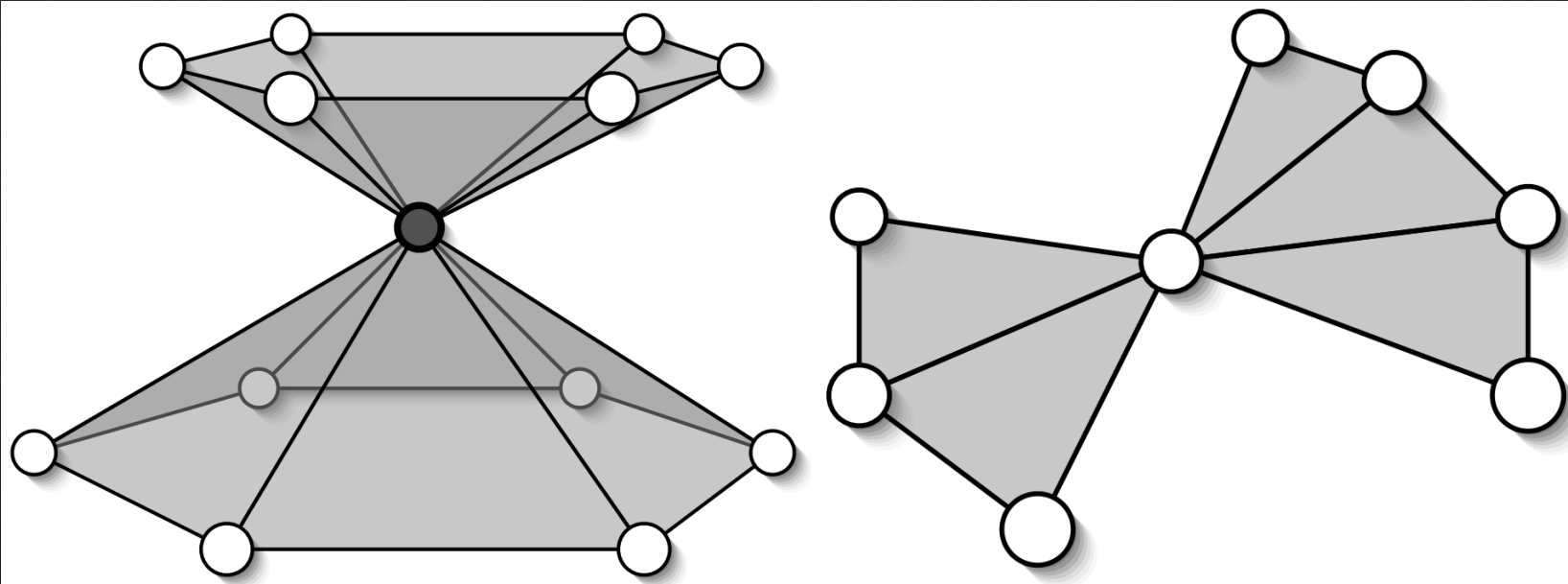
◇ Minimal spanning tree

- ◇ Input: some (>2) vertices
- ◇ Output: a tree passing through all input vertices with minimum length

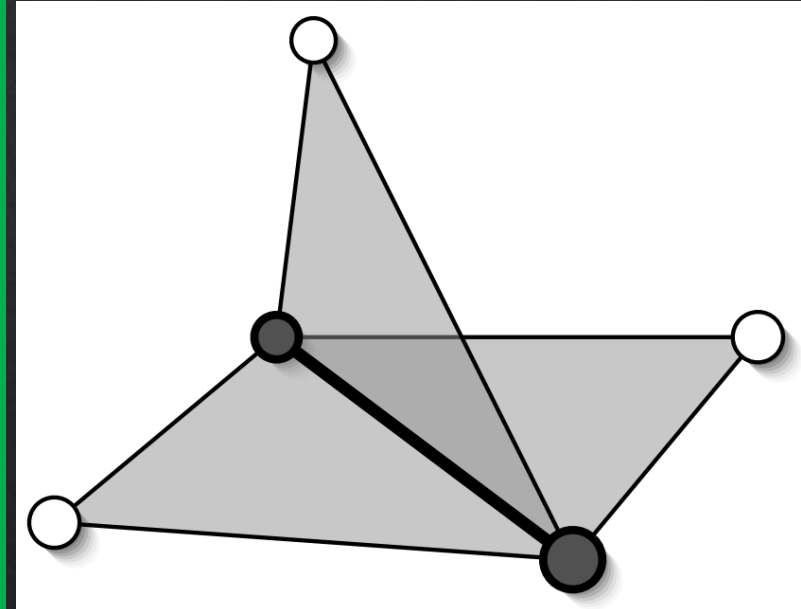
\mathcal{P} : a set of vertices

1. For each pair of terminal points $\mathbf{v}_a \in \mathcal{P}$ and $\mathbf{v}_b \in \mathcal{P}$, compute the shortest path $\mathcal{C}_{a,b}$ between \mathbf{v}_a and \mathbf{v}_b on \mathcal{G} .
2. Construct a complete graph with the node set \mathcal{P} . The weight of each edge $\overline{\mathbf{v}_a \mathbf{v}_b}$ is equal to the length of $\mathcal{C}_{a,b}$.
3. Construct an MST for this shortest distance graph.
4. All mesh edges in the shortest paths that correspond to edges in the MST form an approximate Steiner tree for \mathcal{G} .

2-manifold



Non-manifold vertex is generated by pinching two surface sheets together at that vertex such that the vertex is incident to more than one fan of triangles.

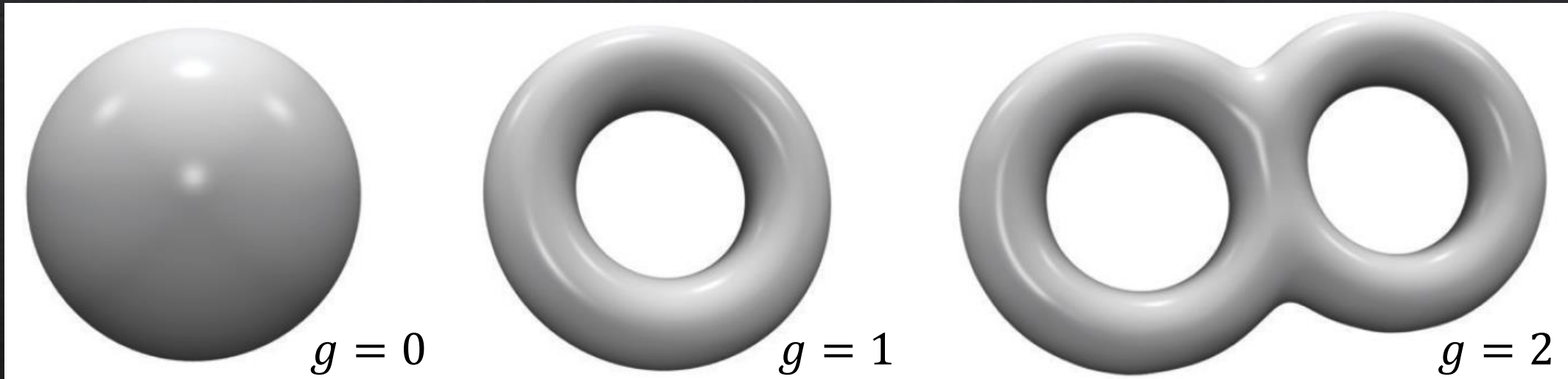


Non-manifold edge has more than two incident triangles.

Euler formula

$$\diamond N_V - N_E + N_F = 2(1 - g)$$

\diamond The numbers of vertices N_V , edges N_E , and faces N_F in a closed and mesh.



$$\diamond N_F \approx 2N_V, N_E \approx 3N_V$$

\diamond 1 face, 1.5 edge $\rightarrow N_E = 1.5N_F$

Barycentric coordinate

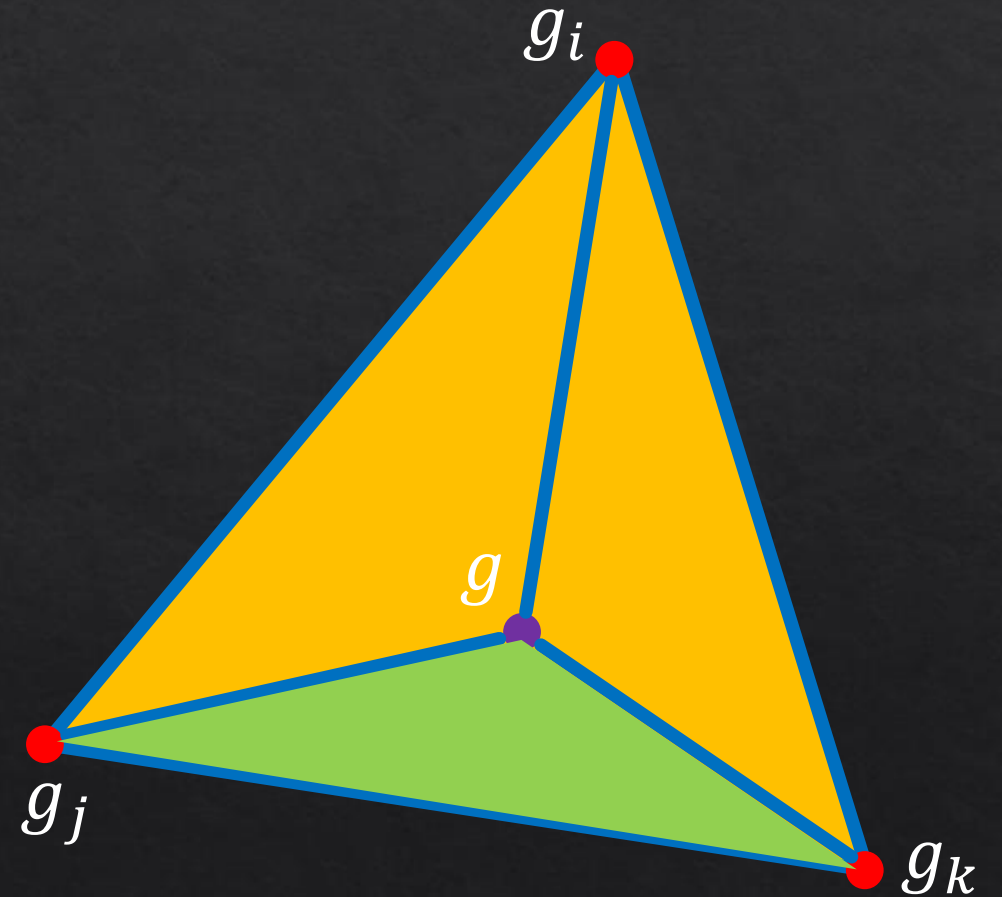
$$g = \alpha g_i + \beta g_j + \gamma g_k$$

$$\alpha + \beta + \gamma = 1,$$

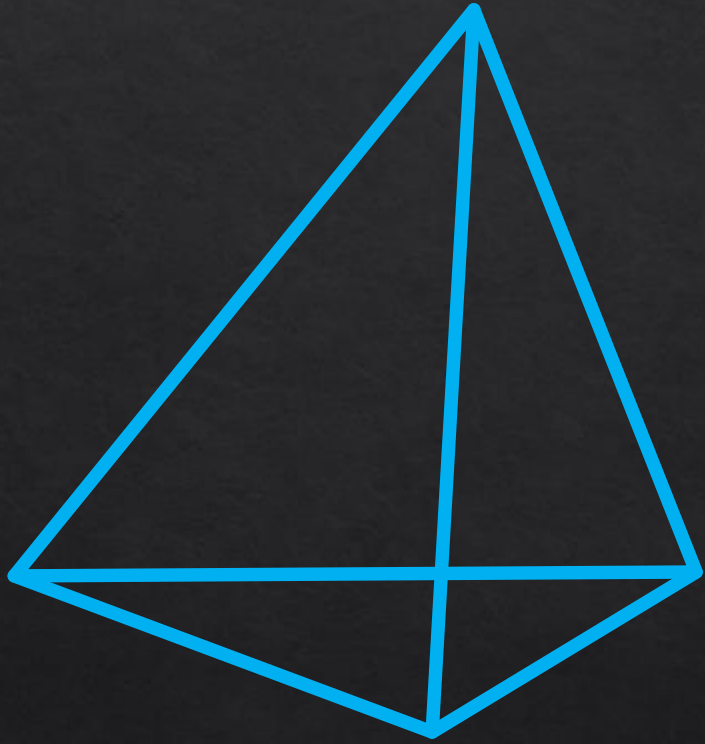
$$\alpha, \beta, \gamma \geq 0.$$

$$\alpha = \frac{s_i}{s_i + s_j + s_k}$$

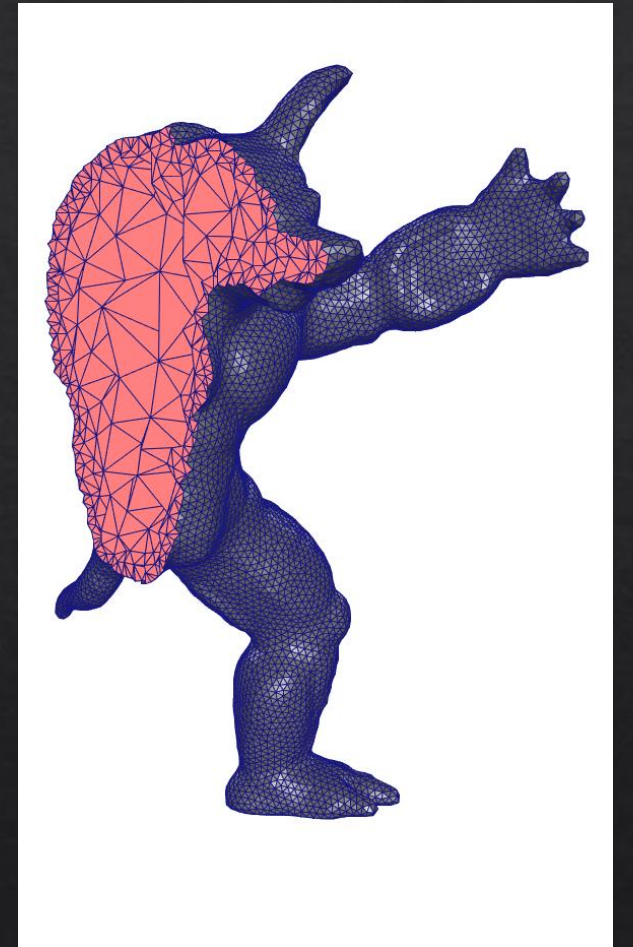
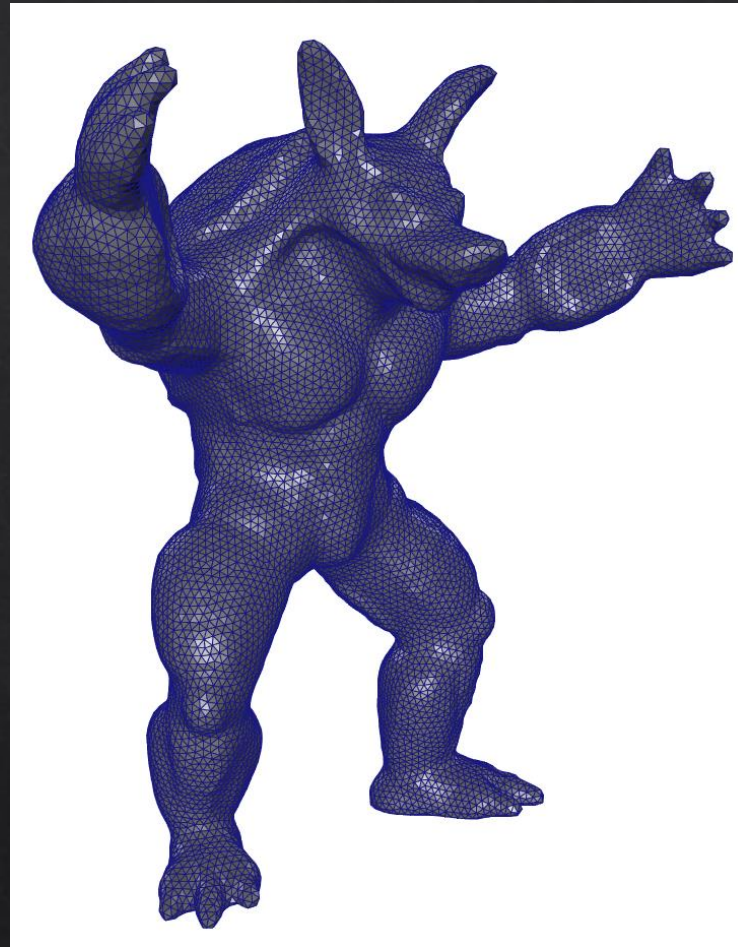
s_i : area of the green triangle



Tetrahedral Mesh



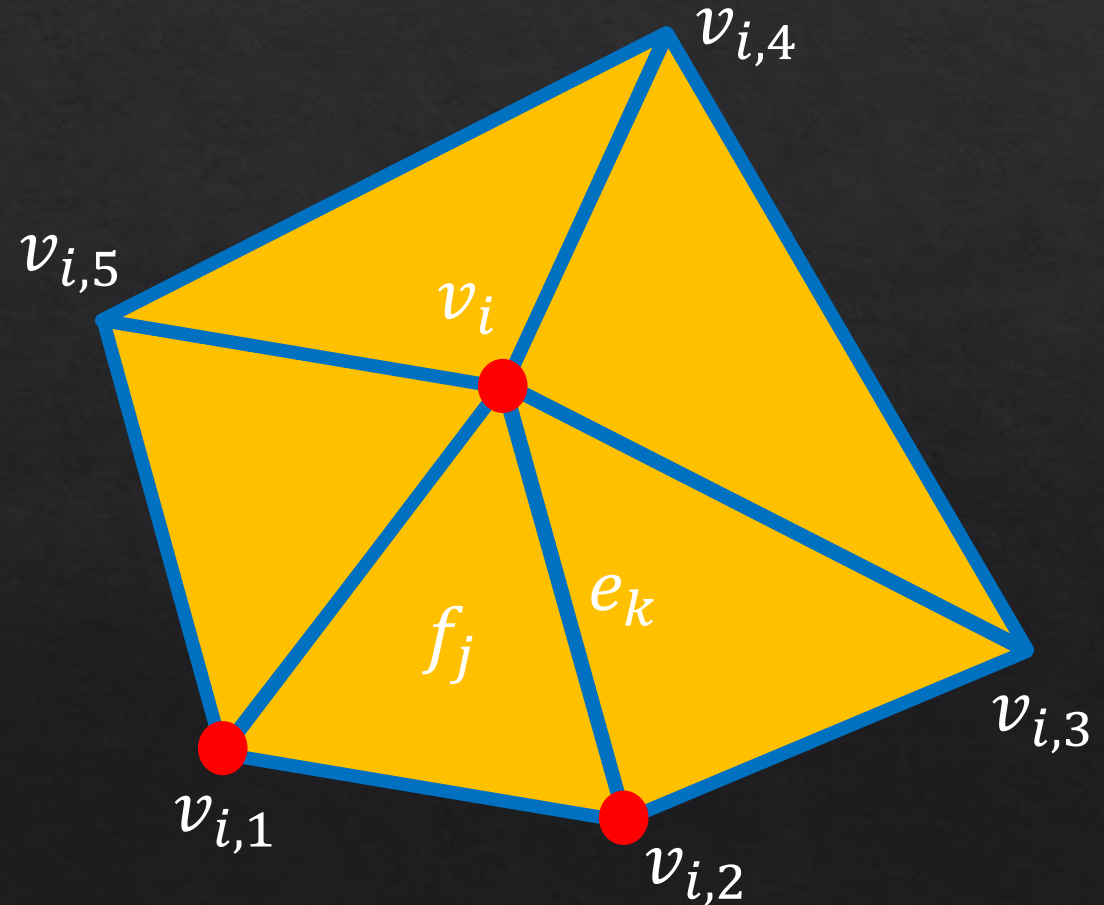
One tetrahedron



A collection of tetrahedrons

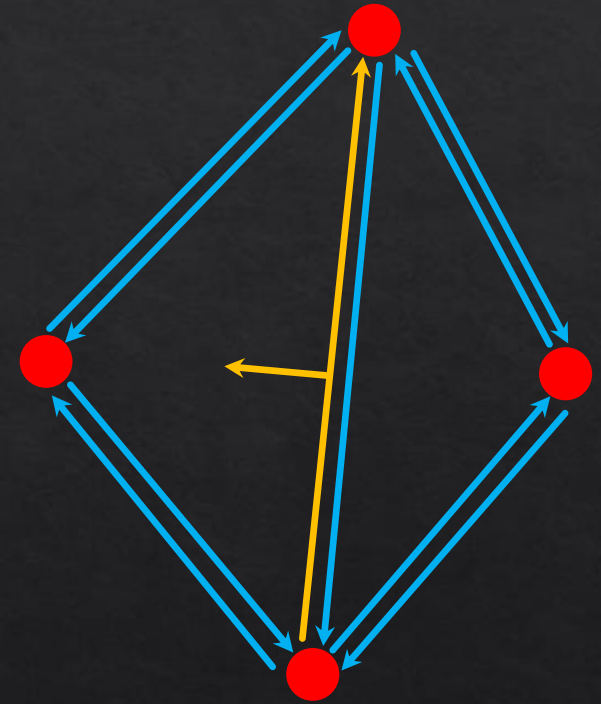
Data structure – requirements

- ◇ Given f_j , find its containing vertices in order.
- ◇ Given v_i , find its one-ring facets in order.
- ◇ Given v_i , find its outgoing edges.
- ◇ Given v_i , find its adjacent vertices.
- ◇ Given e_k , find its connected two facets.
- ◇ Given f_j and e_k , find another facet which connects e_k .
- ◇



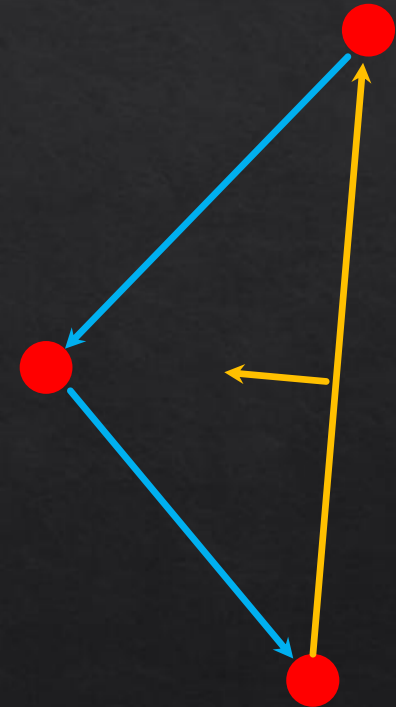
Halfedge

- ◆ Split each edge into two oriented halfedges.
- ◆ Halfedges are oriented consistently in counterclockwise order around each face and along each boundary.
- ◆ One halfedge corresponds one face.
- ◆ Boundary edge: empty face.

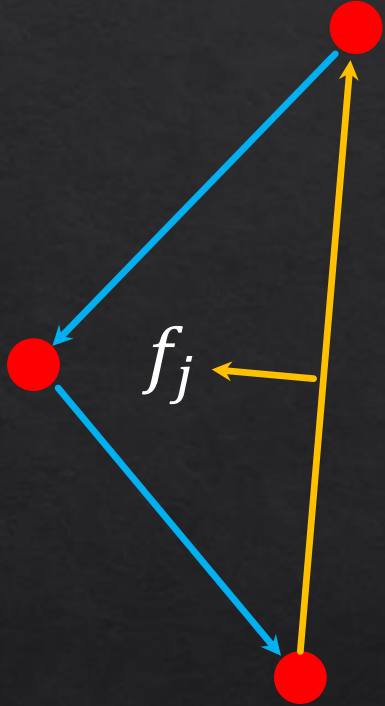


Halfedge

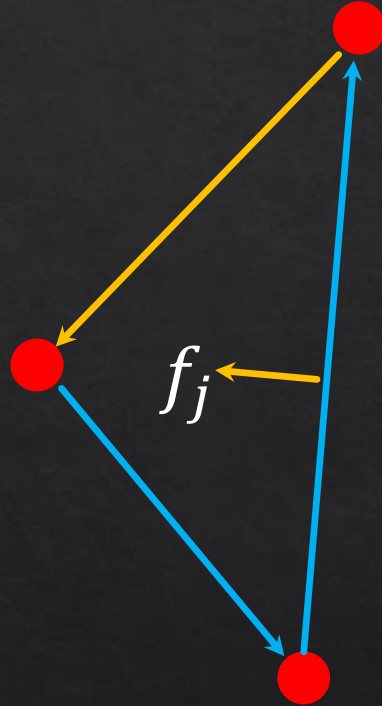
- ◇ Each halfedge:
 - ◇ the vertex it points to,
 - ◇ its adjacent face,
 - ◇ the next halfedge of the face or boundary,
 - ◇ the previous halfedge in the face,
 - ◇ its opposite (or inverse) halfedge.
- ◇ Each vertex:
 - ◇ Position
 - ◇ One outgoing halfedge
- ◇ Each face:
 - ◇ One referenced halfedge



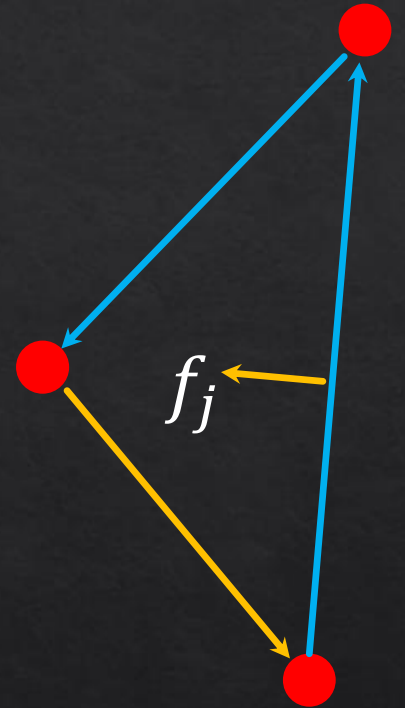
Vertices of one facet



One halfedge of f_j

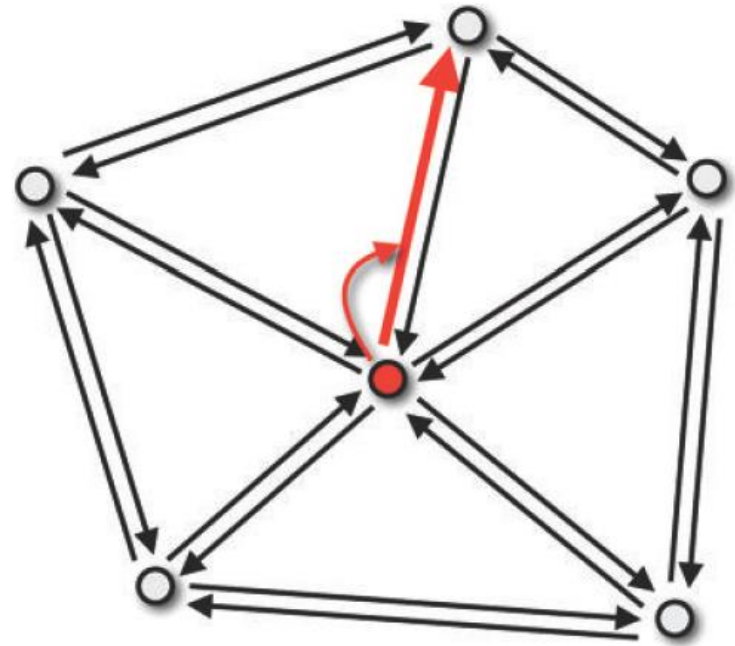


Next halfedge

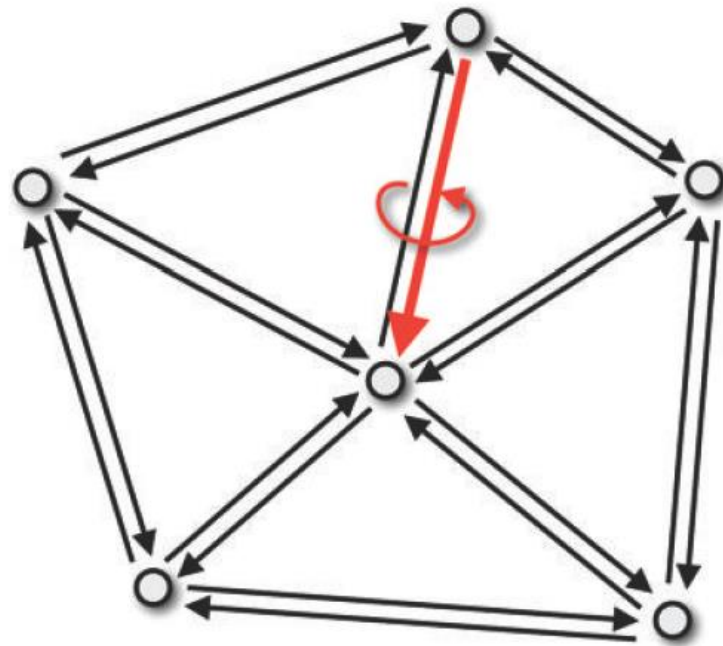


Next halfedge

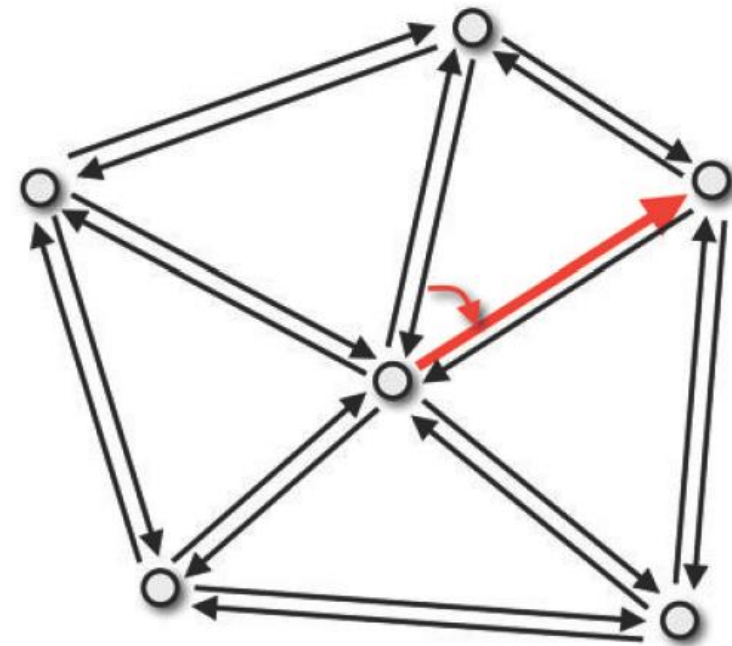
One-ring neighbors



One outgoing halfedge



Opposite halfedge



Next halfedge

Implementation thinking

- ◇ Code: halfedge data implementation
 - ◇ structure
 - ◇ `std::vector`

Outline

- ◇ Point cloud
- ◇ Signed distance field
- ◇ Implicit function
- ◇ Mesh
 - ◇ Triangle, Tetrahedron
- ◇ Grid
 - ◇ Pixel, Voxel
 - ◇ Quad-tree, Octree
- ◇ Data structure
 - ◇ Halfedge
- ◇ **File format**
 - ◇ Obj, Off

File format – obj

https://en.wikipedia.org/wiki/Wavefront_.obj_file

List of geometric vertices, with (x,y,z) coordinates

v 0.123 0.234 0.345

.....

List of texture coordinates, in (u, v) coordinates

vt 0.500 1

.....

List of vertex normals in (x,y,z) form

vn 0.707 0.000 0.707

.....

Polygonal face element (see below)

f 6/4/1 3/5/3 7/6/5 (f v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3) from 1

.....

File format – off

<http://people.sc.fsu.edu/~jburkardt/data/off/off.html>

OFF	#Line 1
vertex_count face_count edge_count	#Line 2
x y z	#One line for each vertex
.....	
n v1 v2 ... Vn	One line for each polygonal face, from 0
.....	

Off - example

OFF

8 6 0

-0.500000 -0.500000 0.500000

0.500000 -0.500000 0.500000

-0.500000 0.500000 0.500000

0.500000 0.500000 0.500000

-0.500000 0.500000 -0.500000

0.500000 0.500000 -0.500000

-0.500000 -0.500000 -0.500000

0.500000 -0.500000 -0.500000

4 0 1 3 2

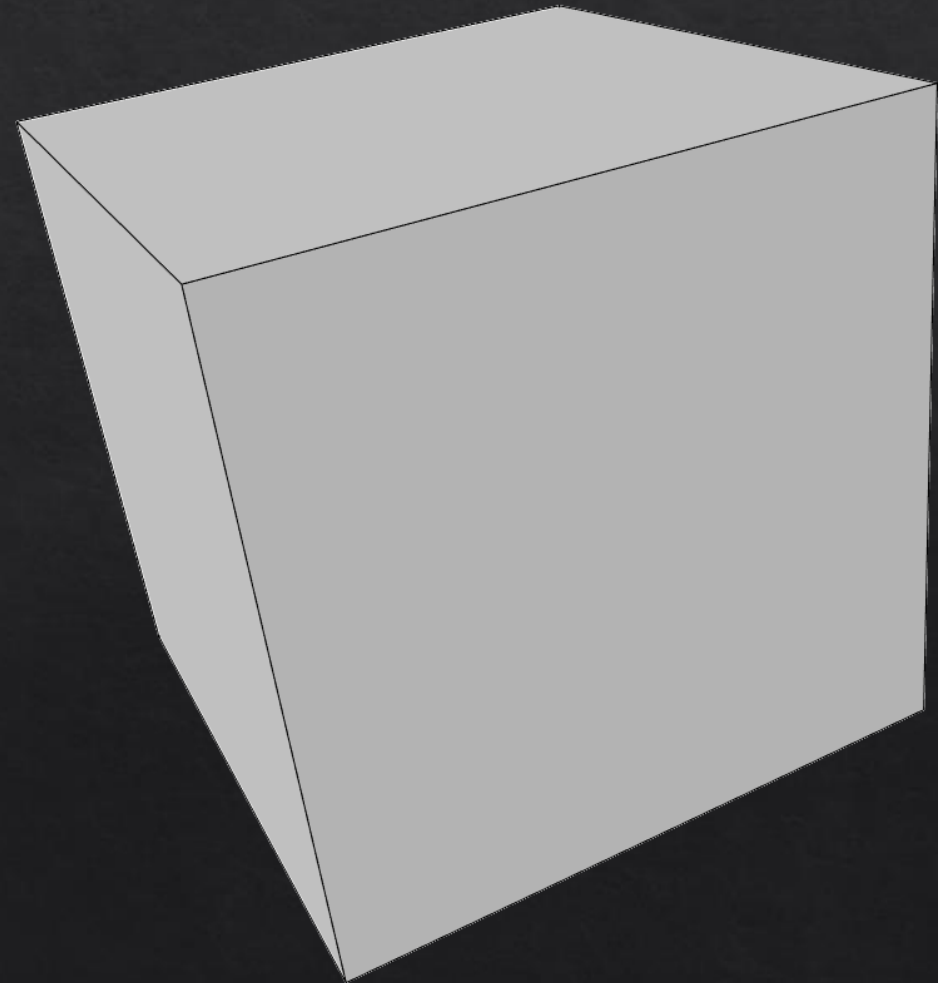
4 2 3 5 4

4 4 5 7 6

4 6 7 1 0

4 1 7 5 3

4 6 0 2 4



Discrete differential geometry

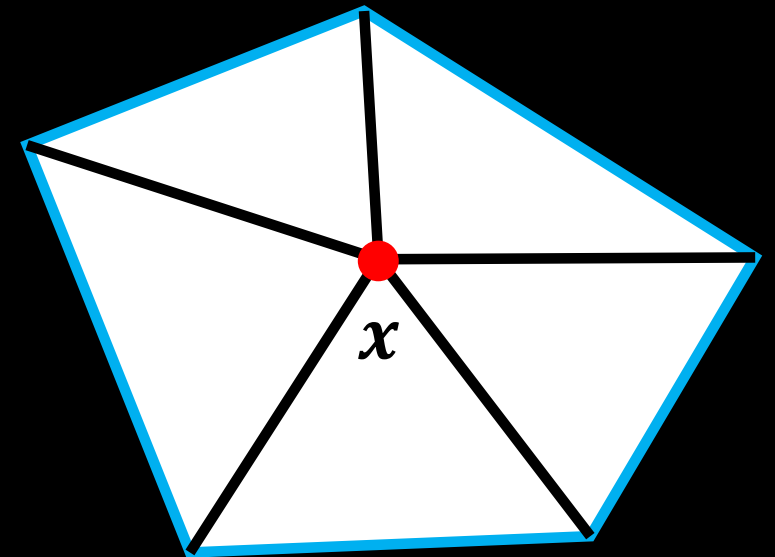
Xiao-Ming Fu

Goal

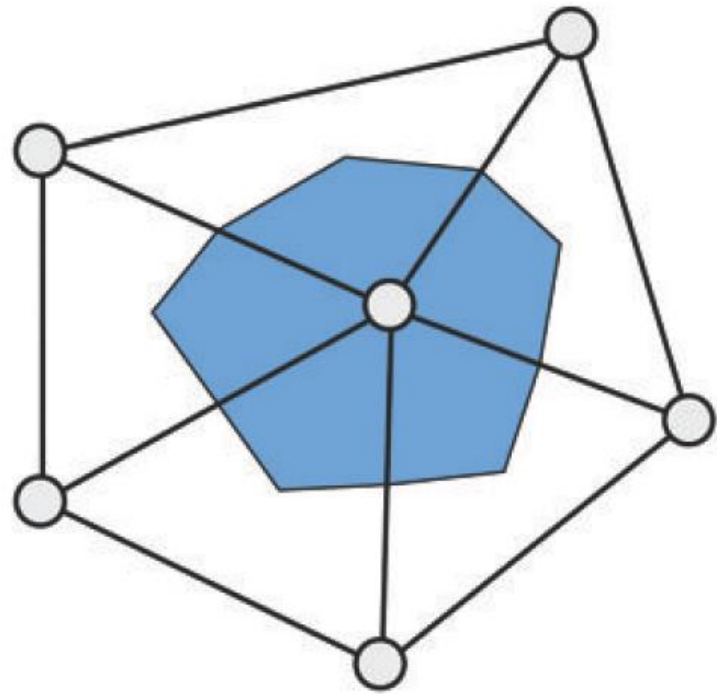
- Compute approximations of the differential properties of this underlying surface directly from the mesh data.
- Local Averaging Region
- Normal Vectors
- Gradients
- Laplace-Beltrami Operator
- Discrete Curvature

Local Averaging Region

- General idea: spatial averages over a local neighborhood $\Omega(\mathbf{x})$ of a point \mathbf{x} .
- \mathbf{x} : one mesh vertex
- $\Omega(\mathbf{x})$: n-ring neighborhoods of mesh vertex or local geodesic balls.
- The size of the $\Omega(\mathbf{x})$: stability and accuracy
 - Large size: smooth
 - Small size: accurate for clean mesh data

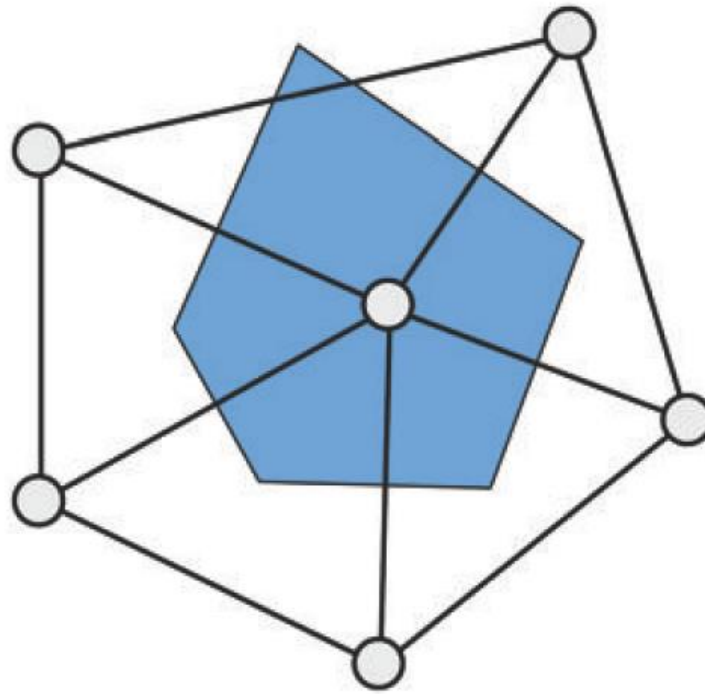


Local Averaging Region



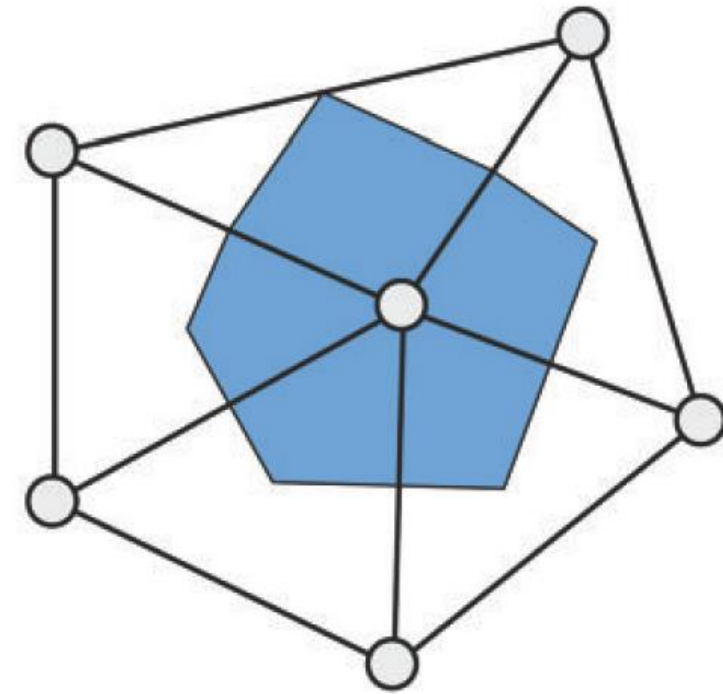
Barycentric cell

triangle barycenters
edge midpoints



Voronoi cell

triangle barycenters
→ triangle circumcenter



Mixed Voronoi cell

circumcenter for obtuse
triangles → edge midpoints

Implementation thinking

- How to compute the area of local average region? For example, barycentric cell.
- One simple idea: for each vertex, compute the area directly.
- Any improvement?
- How about Voronoi cell and mixed Voronoi cell?

Normal Vectors

- Normal vectors for individual triangles are well-defined.
- Vertex normal: spatial averages of normal vectors in a local one-ring neighborhood.

$$\mathbf{n}(v) = \frac{\sum_{T \in \Omega(v)} \alpha_T \mathbf{n}(T)}{\left\| \sum_{T \in \Omega(v)} \alpha_T \mathbf{n}(T) \right\|_2}$$

1. constant weights: $\alpha_T = 1$
2. triangle area: $\alpha_T = \text{area}(T)$
3. incident triangle angles: $\alpha_T = \theta(T)$

Implementation thinking

- How to compute the normal on triangles or vertices?

Barycentric coordinate

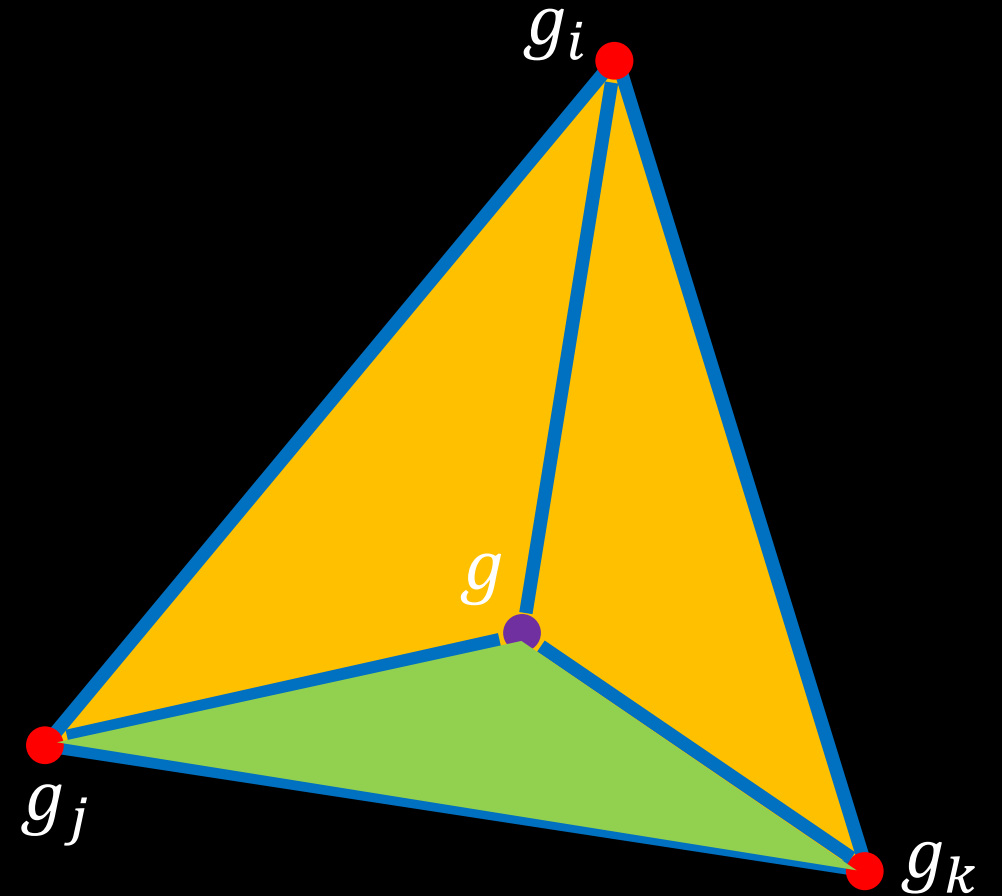
$$g = \alpha g_i + \beta g_j + \gamma g_k$$

$$\alpha + \beta + \gamma = 1,$$

$$\alpha, \beta, \gamma \geq 0.$$

$$\alpha = \frac{s_i}{s_i + s_j + s_k}$$

s_i : area of the green triangle



Gradients

- Given the function value on vertices, compute the gradient on each triangle.

- A piecewise linear function

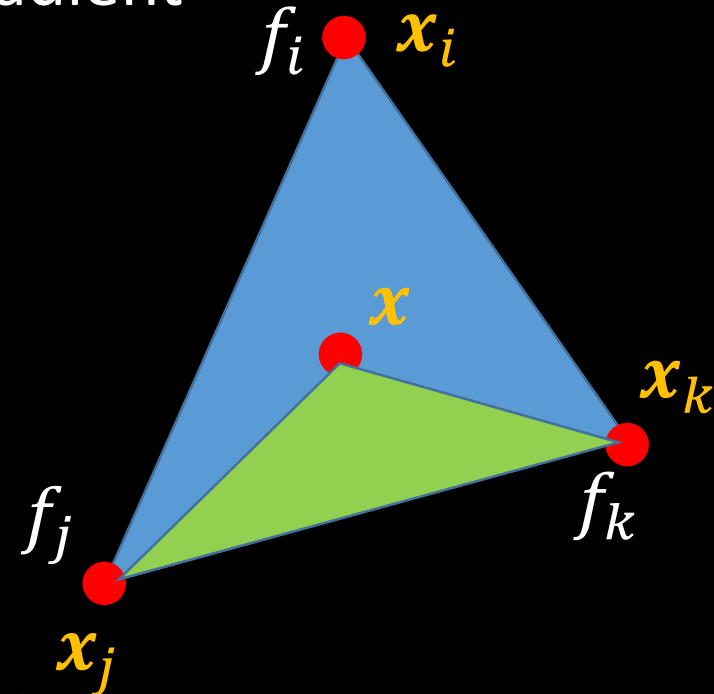
$$f(\mathbf{x}) = \alpha f_i + \beta f_j + \gamma f_k$$

- Gradient:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = f_i \nabla_{\mathbf{x}} \alpha + f_j \nabla_{\mathbf{x}} \beta + f_k \nabla_{\mathbf{x}} \gamma$$

- Because

$$\alpha = \frac{A_i}{A_T} = \frac{\left((\mathbf{x} - \mathbf{x}_j) \cdot \frac{(\mathbf{x}_k - \mathbf{x}_j)^\perp}{\|\mathbf{x}_k - \mathbf{x}_j\|_2} \right) \|\mathbf{x}_k - \mathbf{x}_j\|_2}{2A_T}$$
$$= (\mathbf{x} - \mathbf{x}_j) \cdot (\mathbf{x}_k - \mathbf{x}_j)^\perp / 2A_T$$



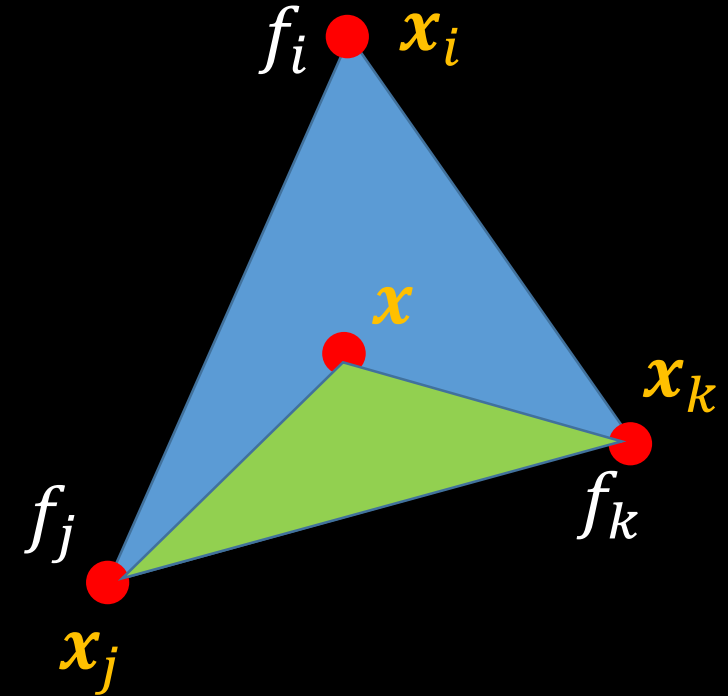
Gradients

- Then

$$\begin{aligned}\nabla_{\mathbf{x}}\alpha &= \frac{(\mathbf{x}_k - \mathbf{x}_j)^\perp}{2A_T} \\ \nabla_{\mathbf{x}}\beta &= \frac{(\mathbf{x}_i - \mathbf{x}_k)^\perp}{2A_T} \\ \nabla_{\mathbf{x}}\gamma &= \frac{(\mathbf{x}_j - \mathbf{x}_i)^\perp}{2A_T}\end{aligned}$$

=>

$$\nabla_{\mathbf{x}}f(\mathbf{x}) = f_i \frac{(\mathbf{x}_k - \mathbf{x}_j)^\perp}{2A_T} + f_j \frac{(\mathbf{x}_i - \mathbf{x}_k)^\perp}{2A_T} + f_k \frac{(\mathbf{x}_j - \mathbf{x}_i)^\perp}{2A_T}$$



Gradients

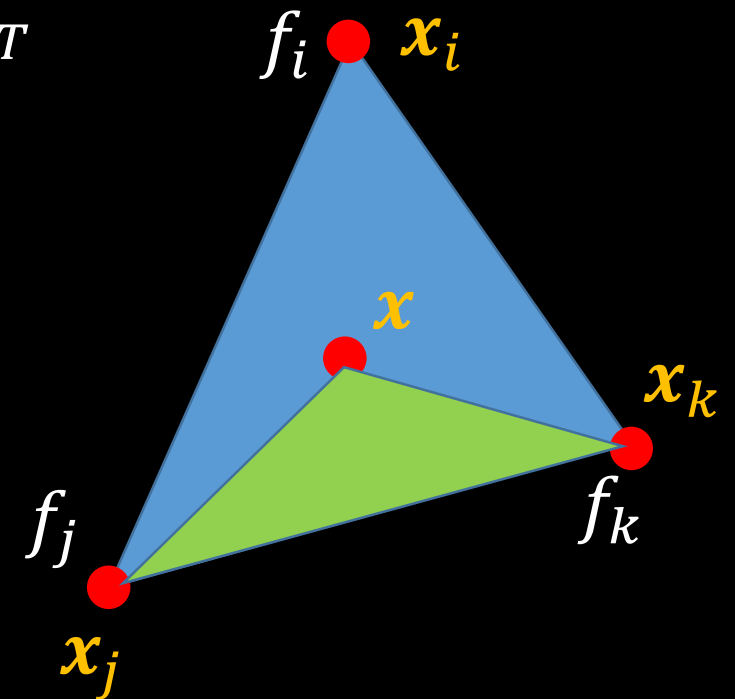
- Because:

$$(\mathbf{x}_k - \mathbf{x}_j)^\perp + (\mathbf{x}_i - \mathbf{x}_k)^\perp + (\mathbf{x}_j - \mathbf{x}_i)^\perp = 0$$

=>

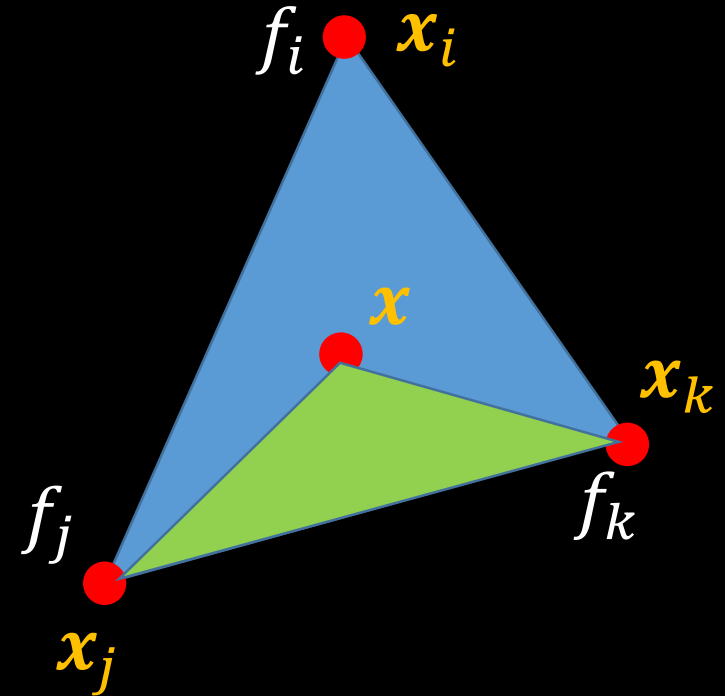
$$\nabla_{\mathbf{x}} f(\mathbf{x}) = (f_j - f_i) \frac{(\mathbf{x}_i - \mathbf{x}_k)^\perp}{2A_T} + (f_k - f_i) \frac{(\mathbf{x}_j - \mathbf{x}_i)^\perp}{2A_T}$$

- Consistent with the formula in the book.



Gradient

- Constant on each facet.
- Different in different facets
 - the signal is C^0
- No definition on vertices.
- If the signal is the positions of the vertices, what does the gradient mean?

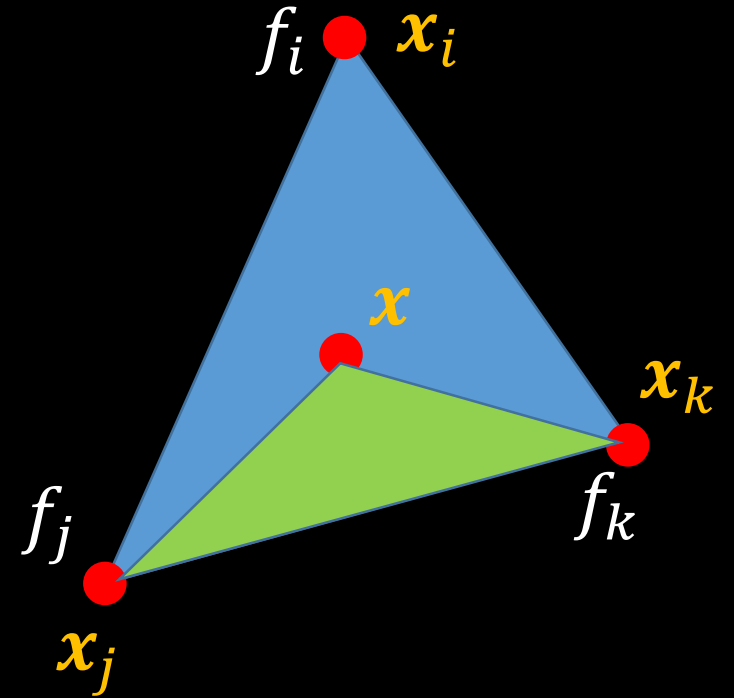


Implementation thinking

- Simple question:

How to compute $(\mathbf{x}_k - \mathbf{x}_j)^\perp$ in 3D?

- How about the gradient in tetrahedral mesh?



Laplace-Beltrami Operator

Paper: Discrete Laplace operators: No free lunch

- $\Delta = -\text{div grad}$ on a smooth surface S
 - **(NULL)**: $\Delta f = 0$ whenever f is constant.
 - **(SYM)** Symmetry: $(\Delta f, g)_{L^2} = (f, \Delta g)_{L^2}$
 - **(LOC)** Local support: for any pair $p \neq q$ of points, $\Delta f(p)$ is independent of $f(q)$.
 - **(LIN)** Linear precision: $\Delta f = 0$ whenever $S \in R^2$ and f is linear.
 - **(MAX)** Maximum principle: harmonic functions have no local maxima at interior points.
 - **(PSD)** Positive semi-definiteness: the Dirichlet energy $E_D(f) = \int_S \|\text{grad } f\|_2^2 dA = (\Delta f, f)_{L^2}$ is non-negative.

Discrete Laplace-Beltrami Operator

Paper: Discrete Laplace operators: No free lunch

- Discrete Laplace operators on triangular surface meshes span the entire spectrum of geometry processing applications:
 - mesh filtering, parameterization, pose transfer, segmentation, reconstruction, re-meshing, compression, simulation, and interpolation via barycentric coordinates.
- Constant gradient on facet \rightarrow zero Laplace value on facet
- Exists on the vertex
- A discrete Laplace operator on vertex-based functions:

$$(\mathbf{L}f)_i = \sum_{j \in \Omega(i)} \omega_{ij} (f_j - f_i)$$

Desired Properties for Discrete Laplace-Beltrami Operator

- Require a discrete Laplacian having properties corresponding to (some subset of) the properties of the continuous Laplace operator:
 - **NULL**
 - $\Delta f = 0$ whenever f is constant.
 - **SYM (SYMMETRY)**
 - Condition: $\omega_{ij} = \omega_{ji}$
 - Real symmetric matrices exhibit real eigenvalues and orthogonal eigenvectors.
 - **LOC (LOCALITY)**
 - Condition: Weights are associated to mesh edges, $\omega_{ij} = 0$ if vertex i and j do not share an edge.
 - Smooth Laplacians govern diffusion processes via $u_t = -\Delta f$.
 - **LIN (LINEAR PRECISION)**
 - $(Lf)_i = 0$ for all interior vertices when the positions of vertices are in the plane.
 - Condition: $0 = (Lx)_i = \sum_j \omega_{ij}(x_i - x_j)$
 - Applications: de-noising, parameterizations, plate bending energies.

Desired Properties for Discrete Laplace-Beltrami Operator

- **POS (POSITIVE WEIGHTS)**

- Condition: $\omega_{ij} > 0$ whenever $i \neq j$.
- A sufficient condition for a discrete maximum principle.
- In diffusion problems, this property assures that flow travels from regions of higher to regions of lower potential.
- Establishes a connection to barycentric coordinates.
- Tutte's embedding theorem: LOCALITY + LINEAR PRECISION + POSITIVE WEIGHTS.

- **PSD (POSITIVE SEMI-DEFINITENESS)**

- Condition: L is symmetric positive semi-definite.
- Discrete Dirichlet energy $E_D(f) = \sum_{i,j} \omega_{ij} (f_i - f_j)^2$.
- SYMMETRY + POSITIVE WEIGHTS \rightarrow POSITIVE SEMI-DEFINITENESS
- POSITIVE SEMI-DEFINITENESS \nrightarrow POSITIVE WEIGHTS

Uniform Laplacian

- $\omega_{ij} = 1$ or $\omega_{ij} = \frac{1}{N_i}$

$$(\mathbf{L}f)_i = \sum_{j \in \Omega(i)} (f_j - f_i) \quad \text{or} \quad (\mathbf{L}f)_i = \frac{1}{N_i} \sum_{j \in \Omega(i)} (f_j - f_i)$$

- Violate property of LINEAR PRECISION

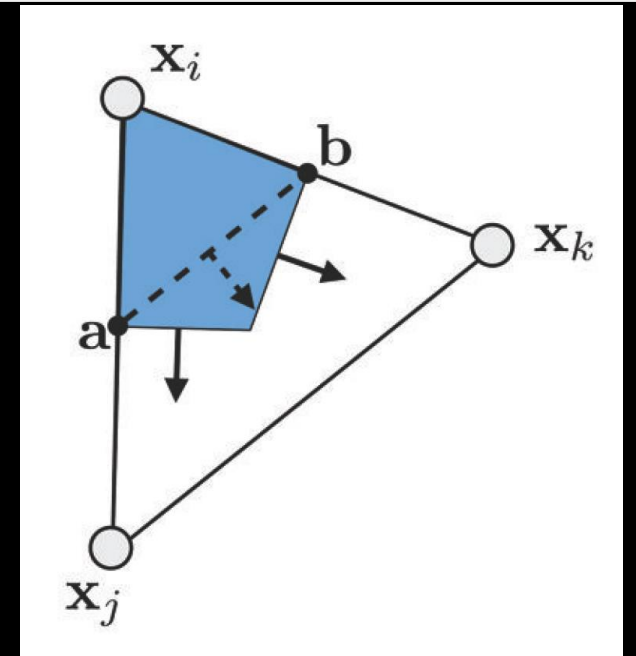
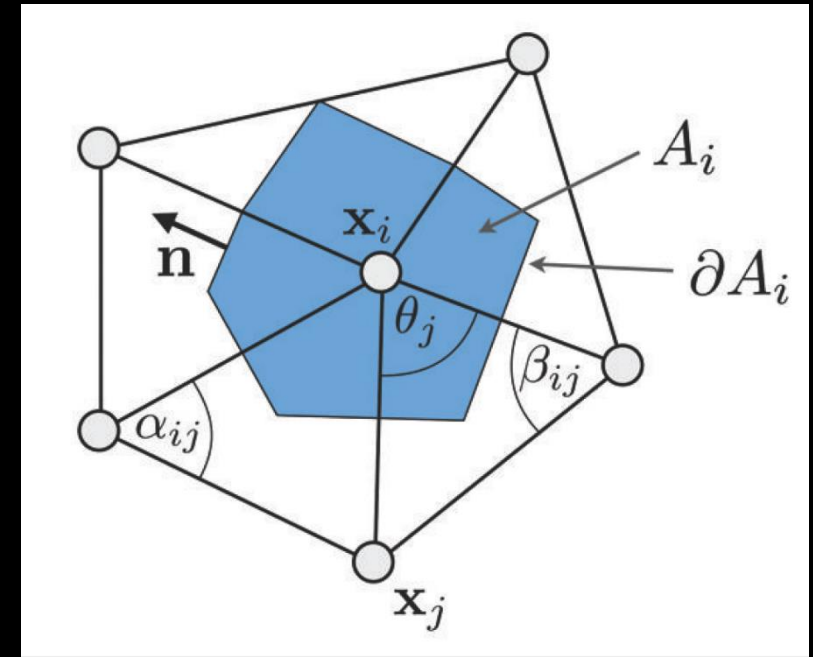
- The definition only depends on the connectivity of the mesh.
- The uniform Laplacian does not adapt at all to the spatial distribution of vertices.

Cotangent Formula

- mixed finite element/finite volume method
 - Assume it constant on each vertex

$$\int_{A_i} \Delta f dA = \int_{A_i} \operatorname{div} \nabla f dA = \int_{\partial A_i} (\nabla f) \cdot \mathbf{n} ds$$

1. A_i is the local averaging domain of vertex i .
2. ∂A_i is the boundary of A_i .
3. \mathbf{n} is the outward pointing unit normal of the boundary.
4. f is the signal defined on mesh.



Cotangent Formula

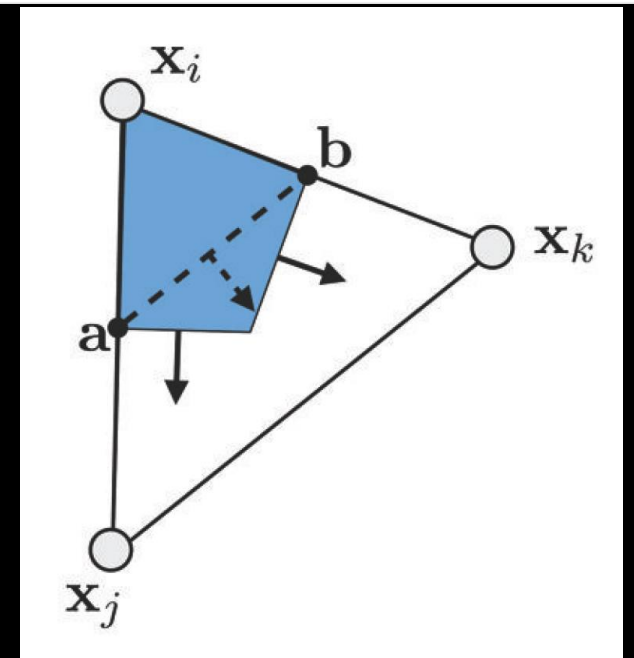
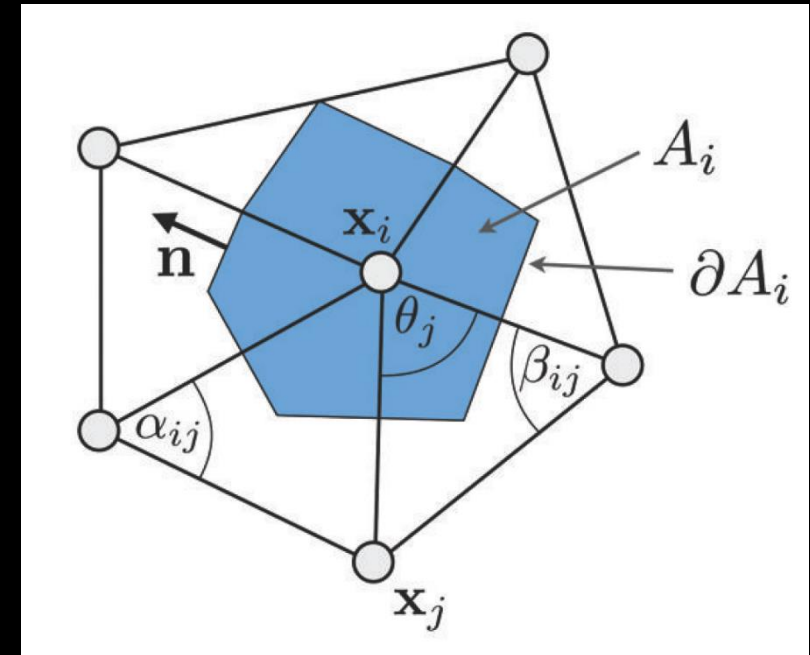
We split this integral by considering the integration separately for each triangle T .

$$\int_{\partial A_i \cap T} \nabla f \cdot \mathbf{n} ds = \nabla f \cdot (\mathbf{a} - \mathbf{b})^\perp = \frac{1}{2} \nabla f \cdot (\mathbf{x}_j - \mathbf{x}_k)^\perp$$

∇f is constant within each triangle.

$$\nabla f = (f_j - f_i) \frac{(\mathbf{x}_i - \mathbf{x}_k)^\perp}{2A_T} + (f_k - f_i) \frac{(\mathbf{x}_j - \mathbf{x}_i)^\perp}{2A_T}$$

$$\int_{\partial A_i \cap T} \nabla f \cdot \mathbf{n} ds = (f_j - f_i) \frac{(\mathbf{x}_i - \mathbf{x}_k)^\perp \cdot (\mathbf{x}_j - \mathbf{x}_k)^\perp}{4A_T} + (f_k - f_i) \frac{(\mathbf{x}_j - \mathbf{x}_i)^\perp \cdot (\mathbf{x}_j - \mathbf{x}_k)^\perp}{4A_T}$$



Cotangent Formula

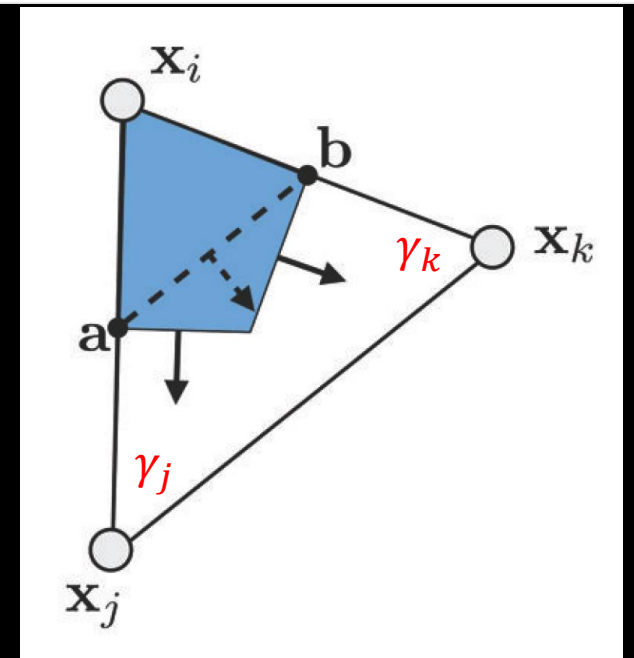
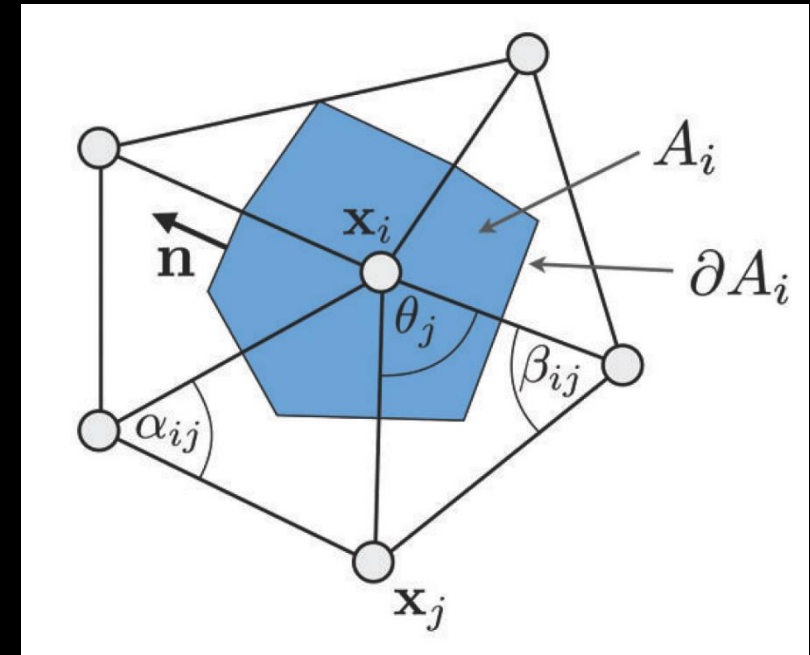
- Because:

$$\begin{aligned} A_T &= \frac{1}{2} \sin \gamma_j \|\mathbf{x}_j - \mathbf{x}_i\| \|\mathbf{x}_j - \mathbf{x}_k\| \\ &= \frac{1}{2} \sin \gamma_k \|\mathbf{x}_i - \mathbf{x}_k\| \|\mathbf{x}_j - \mathbf{x}_k\| \end{aligned}$$

and

$$\cos \gamma_j = \frac{(\mathbf{x}_j - \mathbf{x}_i) \cdot (\mathbf{x}_j - \mathbf{x}_k)}{\|\mathbf{x}_j - \mathbf{x}_i\| \|\mathbf{x}_j - \mathbf{x}_k\|}$$

$$\cos \gamma_k = \frac{(\mathbf{x}_i - \mathbf{x}_k) \cdot (\mathbf{x}_j - \mathbf{x}_k)}{\|\mathbf{x}_i - \mathbf{x}_k\| \|\mathbf{x}_j - \mathbf{x}_k\|}$$



Cotangent Formula

• and

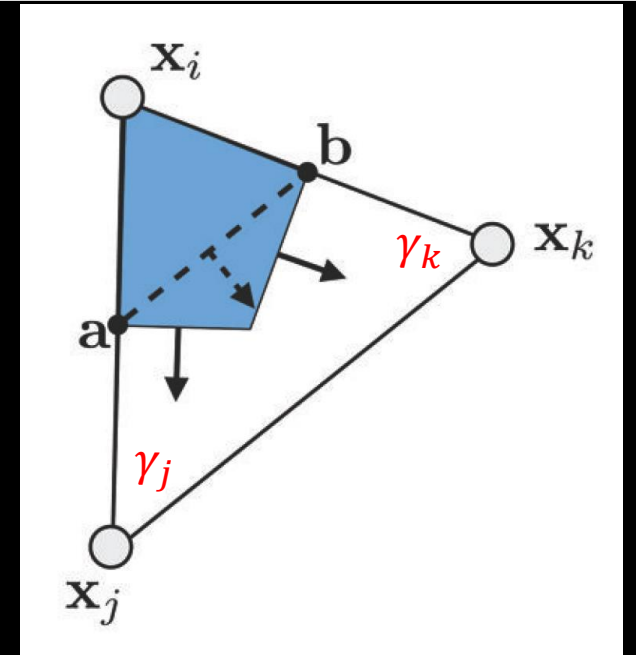
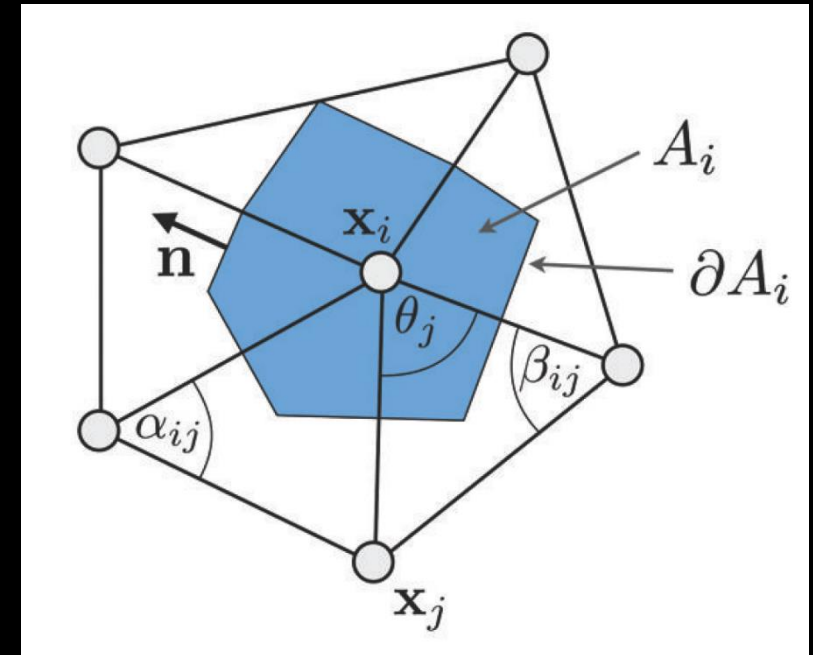
$$(\mathbf{x}_i - \mathbf{x}_k)^\perp \cdot (\mathbf{x}_j - \mathbf{x}_k)^\perp = (\mathbf{x}_i - \mathbf{x}_k) \cdot (\mathbf{x}_j - \mathbf{x}_k)$$

$$(\mathbf{x}_j - \mathbf{x}_i)^\perp \cdot (\mathbf{x}_j - \mathbf{x}_k)^\perp = (\mathbf{x}_j - \mathbf{x}_i) \cdot (\mathbf{x}_j - \mathbf{x}_k)$$

So

$$\int_{\partial A_i \cap T} \nabla f \cdot \mathbf{n} ds$$

$$= \frac{1}{2} \left(\cot \gamma_k (f_j - f_i) + \cot \gamma_j (f_k - f_i) \right)$$



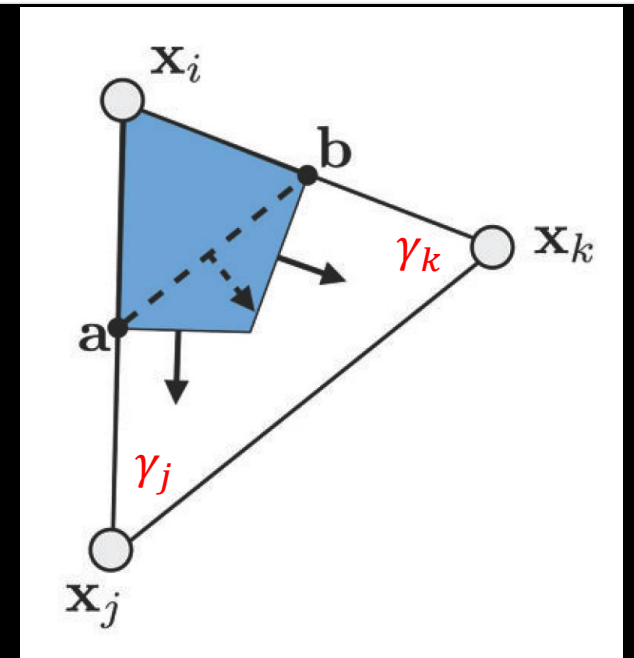
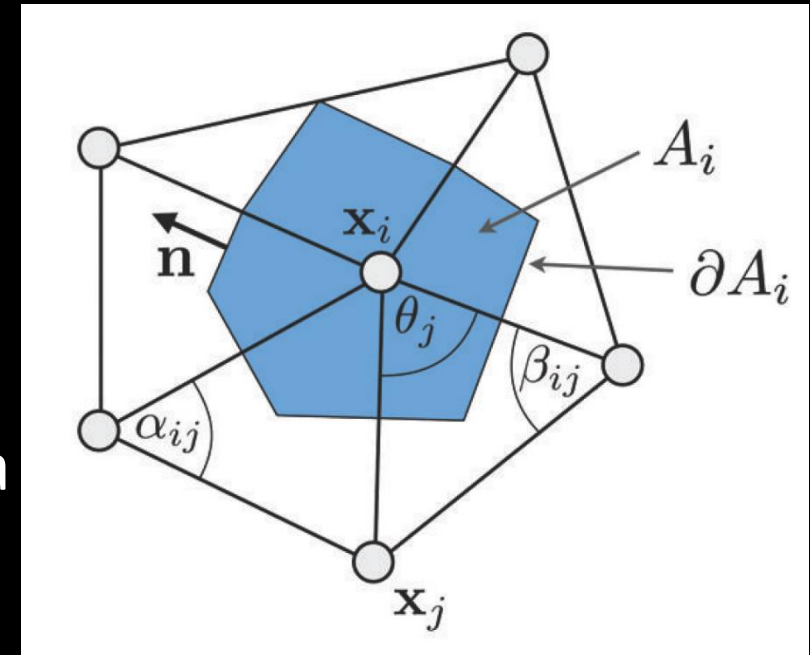
Cotangent Formula

$$\int_{A_i} \Delta f dA = \frac{1}{2} \sum_{j \in \Omega(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(f_j - f_i)$$

Discrete average of the Laplace-Beltrami operator of a function f at vertex v_i is given as:

$$\Delta f(v_i) = \frac{1}{2A_i} \sum_{j \in \Omega(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(f_j - f_i)$$

1. most widely used discretization
2. $(\cot \alpha_{ij} + \cot \beta_{ij})$ become negative if $\alpha_{ij} + \beta_{ij} > \pi$.
Violate the property of POSITIVE WEIGHTS.



No free lunch

- Main result Not all meshes admit Laplacians satisfying properties (SYMMETRY), (LOCALITY), (LINEAR PRECISION), and (POSITIVE WEIGHTS) simultaneously.

Implementation thinking

- How to compute the cotangent formula?
- One simple idea: for each edge, compute the related cot value.
- Any improvement? More efficient?

Discrete Curvature

- When applied to the coordinate function \mathbf{x} , the Laplace-Beltrami operator provides a discrete approximation of the mean curvature normal.

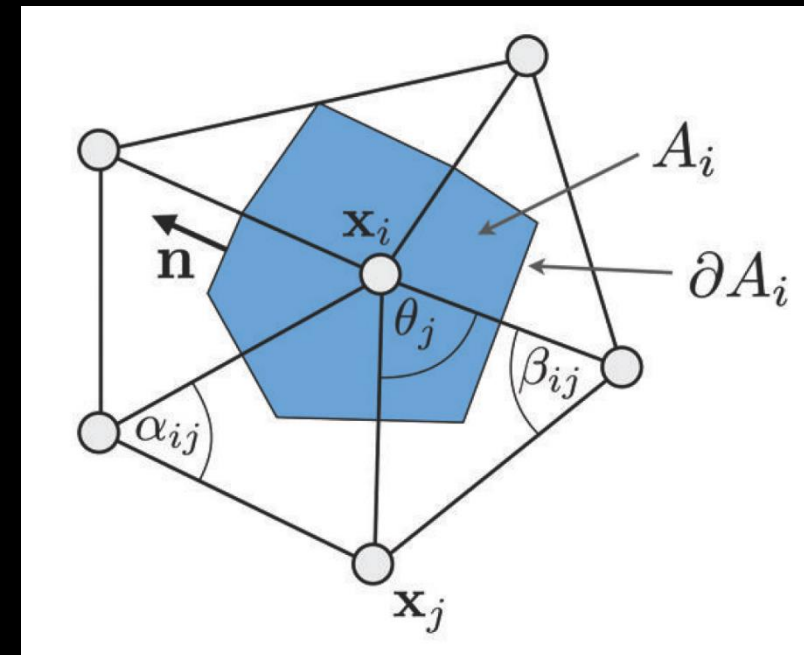
$$\Delta \mathbf{x} = -2H\mathbf{n}$$

absolute discrete mean curvature at vertex i :

$$H_i = \frac{1}{2} \|\Delta \mathbf{x}\|$$

- A discrete operator for Gaussian curvature:

$$K_i = \frac{1}{A_i} \left(2\pi - \sum_{j \in \Omega(i)} \theta_j \right)$$



Implementation thinking

- How to compute the discrete Gaussian curvature?
- One simple idea: for each vertex, compute the related angle.

- Any improvement? More efficient?

Second homework

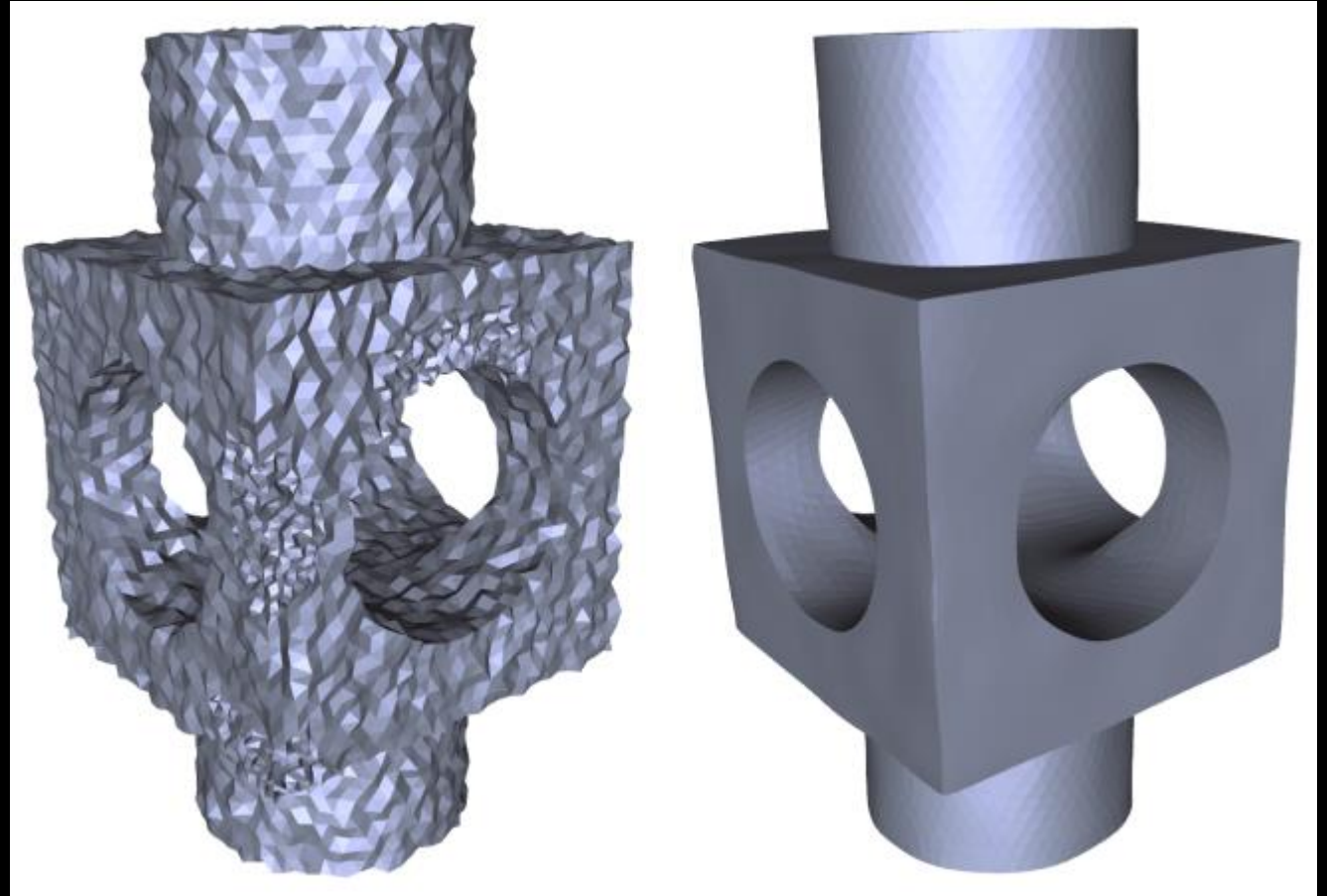
- Color bar
 - Map a value to a color
- Visualize:
 - mean curvature,
 - absolute mean curvature,
 - and Gaussian curvature.

Mesh Smoothing

Xiao-Ming Fu

Denoising

- Removing the noise (the high frequencies) and keeping the overall shape (the low frequencies)
- Physical scanning process
- Feature VS Noise



Smoothing – From wiki

- In statistics and image processing, to smooth a data set is to create **an approximating function** that attempts to capture **important patterns** in the data, while leaving out **noise or other fine-scale structures/rapid phenomena**.
- In smoothing, the data points of a signal are **modified** so individual points (presumably because of noise) are reduced, and points that are lower than the adjacent points are increased leading to a smoother signal.

Outline

- Filter-based methods
- Optimization-based methods
- Data-driven methods

Outline

- Filter-based methods
- Optimization-based methods
- Data-driven methods

Laplacian smoothing

- Diffusion flow: a mathematically well-understood model for the time-dependent process of smoothing a given signal $f(\mathbf{x}, t)$.
 - Heat diffusion, Brownian motion

- Diffusion equation:

$$\frac{\partial f(\mathbf{x}, t)}{\partial t} = \lambda \Delta f(\mathbf{x}, t)$$

1. A second-order linear partial differential equation;
2. Smooth an arbitrary function f on a manifold surface by using Laplace-Beltrami Operator.
3. Discretize the equation both in space and time.

Spatial discretization

- Sample values at the mesh vertices $\mathbf{f}(t) = (f(v_1, t), \dots, f(v_n, t))^T$
- Discrete Laplace-Beltrami using either the uniform or cotangent formula.
- The evolution of the function value of each vertex:

$$\frac{\partial f(v_i, t)}{\partial t} = \lambda \Delta f(x_i, t)$$

Matrix form:

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

Temporal discretization

- Uniform sampling: $(t, t + h, t + 2h, \dots)$
- Explicit Euler integration:

$$f(t + h) = f(t) + h \frac{\partial f(t)}{\partial t} = f(t) + h\lambda \cdot Lf(t)$$

1. Numerically stability: a sufficiently small time step h .

- Implicit Euler integration:

$$\begin{aligned} f(t + h) &= f(t) + h\lambda \cdot Lf(t + h) \\ \Leftrightarrow (I - h\lambda \cdot L)f(t + h) &= f(t) \end{aligned}$$

Laplacian smoothing

- Arbitrary function \implies vertex positions
 - $f \implies (x_1, \dots, x_n)^T$

- Laplacian smoothing:

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + h\lambda \cdot \Delta \mathbf{x}_i$$

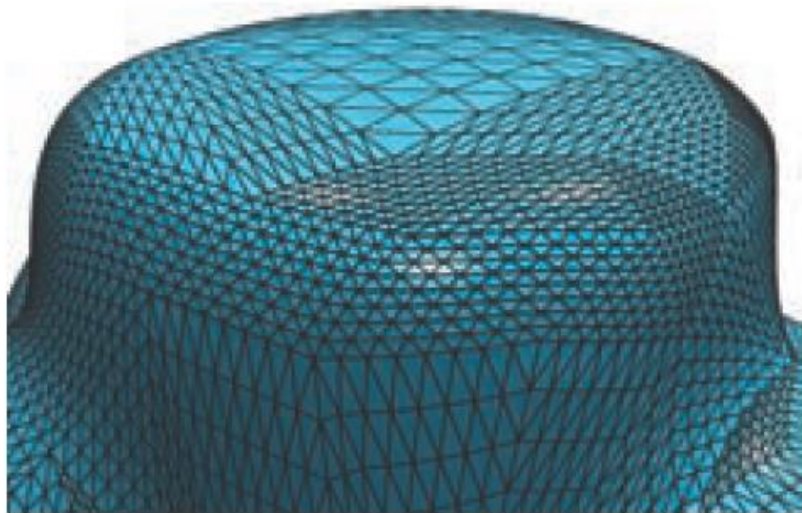
1. $\Delta \mathbf{x} = -2H\mathbf{n} \rightarrow$ vertices move along the normal direction by an amount determined by the mean curvature H .
2. mean curvature flow.



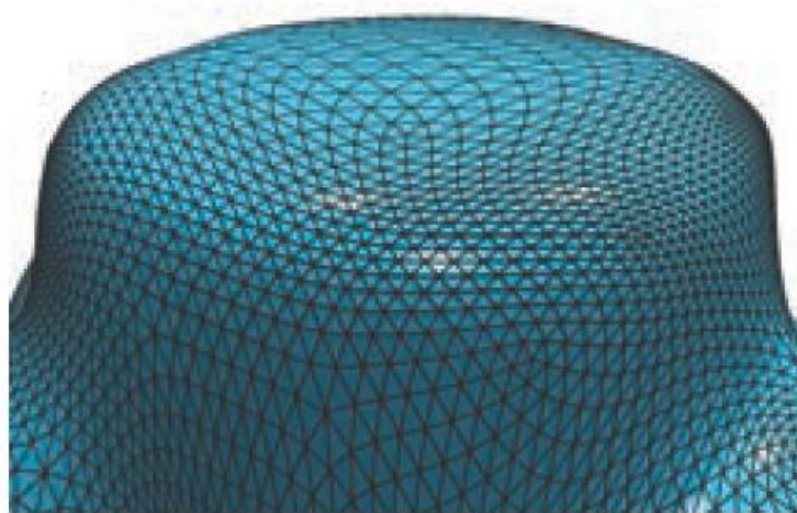
Figure 4.5. Curvature flow smoothing of the bunny mesh (left), showing the result after ten iterations (center) and 100 iterations (right). The color coding shows the mean curvature. (Model courtesy of the Stanford Computer Graphics Laboratory.)

Different Laplace-Beltrami operators

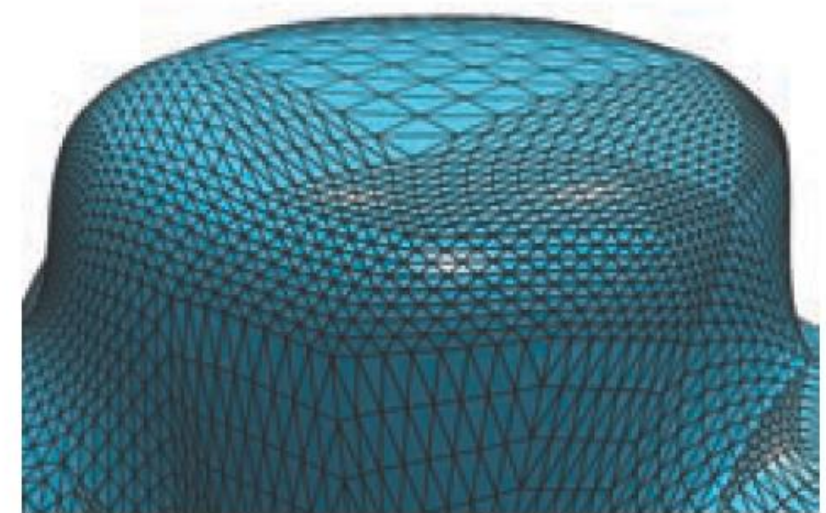
- Cotangent Laplacian.
 - the movement in the normal direction is true.
- Uniform Laplacian
 - move each vertex to the barycenter of its one-ring neighbors.
 - smooths the mesh geometry and a tangential relaxation of the triangulation.



Input



Uniform



Cotangent

Fairing

- Goal: compute shapes that are as smooth as possible
- Steady-states of the flow:
 - $L\mathbf{x} = 0$
 - $L^k\mathbf{x} = 0$
 - ...



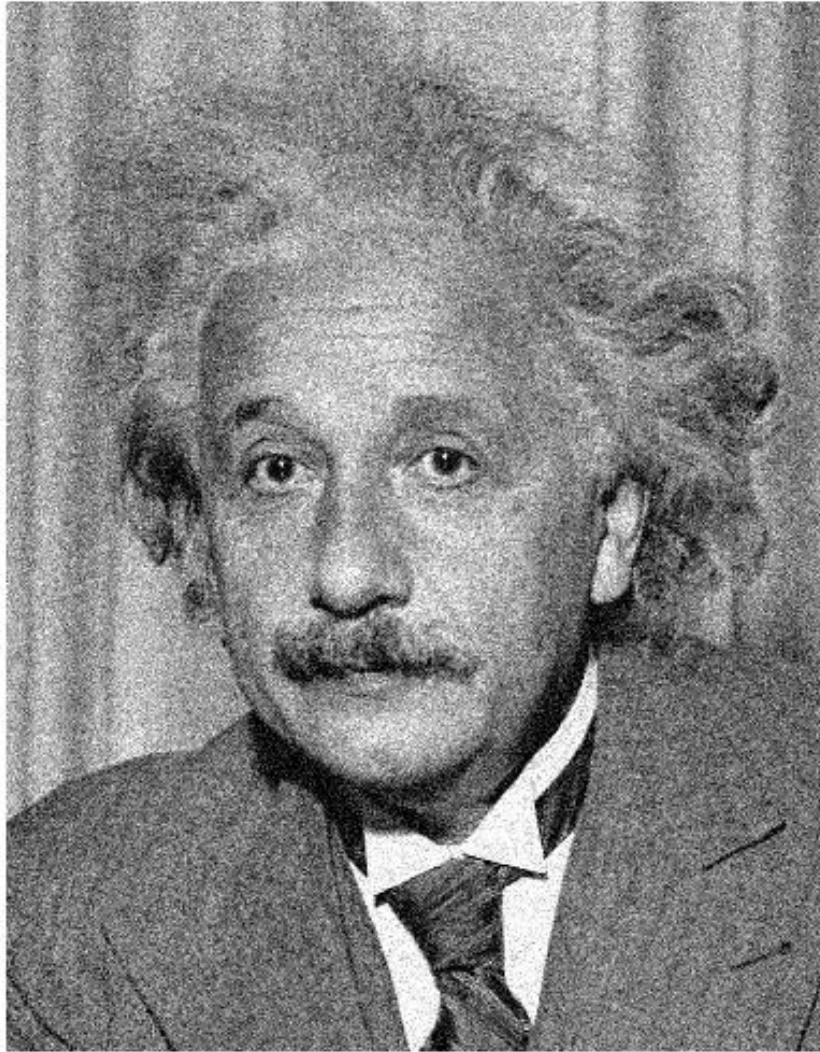
Figure 4.8. The blue region is determined by minimizing a fairness functional: membrane surface ($\Delta\mathbf{x} = 0$, left), thin-plate surface ($\Delta^2\mathbf{x} = 0$, center), and minimum variation surface ($\Delta^3\mathbf{x} = 0$, right). The order k of the Euler-Lagrange equation $\Delta^k\mathbf{x} = 0$ determines the maximum smoothness C^{k-1} at the boundary. (Image taken from [Botsch and Kobbelt 04a]. ©2004 ACM, Inc. Included here by permission.)

Gaussian Image Denoising

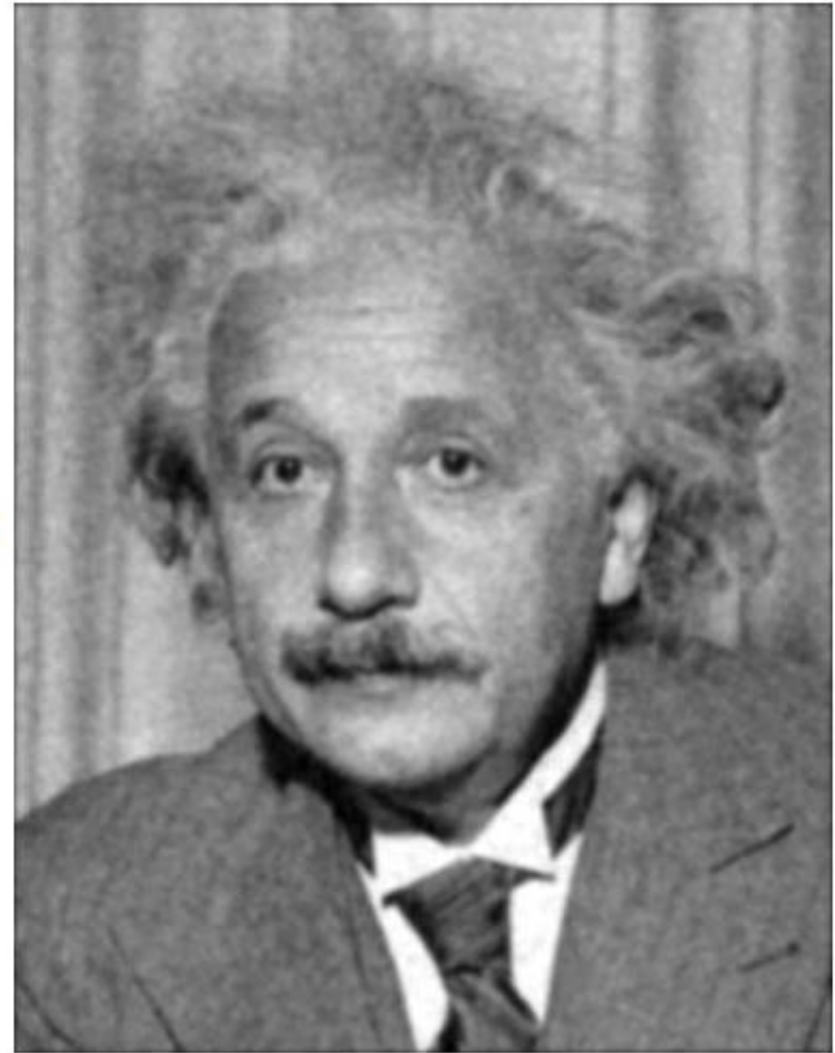
- The Gaussian filter for an image pixel $I(\mathbf{p})$, at coordinate $\mathbf{p} = (x, y)$, is defined as:

$$I(\mathbf{p}) \leftarrow \frac{1}{K_{\mathbf{p}}} \sum_{\mathbf{q} \in \Omega(\mathbf{p})} W_s(\|\mathbf{p} - \mathbf{q}\|) I(\mathbf{q})$$

1. $\Omega(\mathbf{p})$: neighborhood of \mathbf{p} .
2. W_s : position similarity between \mathbf{p} and \mathbf{q} , Gaussian function with standard deviations σ_s
3. $K_{\mathbf{p}}$ is the normalization term, the summation of weights.



Additive Gaussian Noise



Edge is not preserved!!!

Bilateral Image Denoising

- The bilateral filter for an image pixel $I(\mathbf{p})$, at coordinate $\mathbf{p} = (x, y)$, is defined as:

$$I(\mathbf{p}) \leftarrow \frac{1}{K_p} \sum_{\mathbf{q} \in \Omega(\mathbf{p})} W_s(\|\mathbf{p} - \mathbf{q}\|) W_r(\|I(\mathbf{p}) - I(\mathbf{q})\|) I(\mathbf{q})$$

1. $\Omega(\mathbf{p})$: neighborhood of \mathbf{p} .
2. W_s and W_r : monotonically decreasing weighting functions and often be Gaussian functions, with standard deviations σ_s and σ_r .
3. W_s : position similarity between \mathbf{p} and \mathbf{q} .
4. W_r : color similarity between \mathbf{p} and \mathbf{q} .
5. K_p is the normalization term, the summation of weights.

Bilateral Image Denoising

- non-linear
- edge-preserving
- noise-reducing smoothing
- **Extended to mesh case!!!**

Bilateral Mesh Denoising

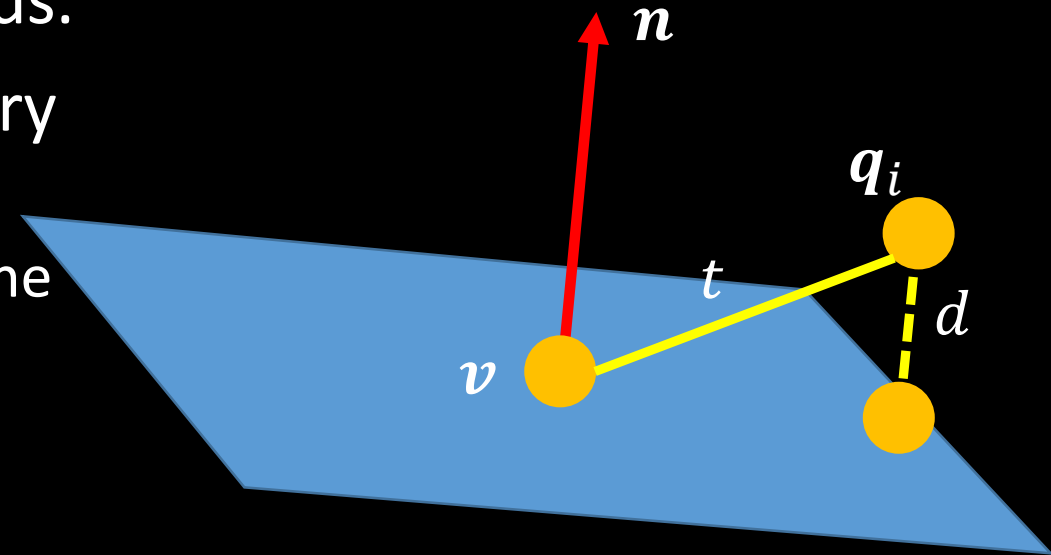
Paper: Bilateral Mesh Denoising

- Vertex positions cannot simply be considered as the signal to be processed.
- Filtering a mesh using local neighborhoods.
- Main idea: local parameter space for every vertex using the tangent plane.
 - The heights of vertices over the tangent plane \Leftrightarrow gray-level values of an image.

- Update v :

$$v^{new} = v + d \cdot n$$

Once n is given, we need to compute the new d to update v .



Pseudo-code

Denoise_Point(Vertex \mathbf{v} , Normal \mathbf{n})

$\{q_i\} = \Omega(\mathbf{v})$, N : number of neighbor vertices, d_{sum} , $K_{\mathbf{v}} = 0$

for $i := 1$ to N

$$t = \|\mathbf{v} - \mathbf{q}_i\|$$

$$d = \langle \mathbf{n}, \mathbf{v} - \mathbf{q}_i \rangle$$

$$w_s = \exp(-t^2 / (2\sigma_s))$$

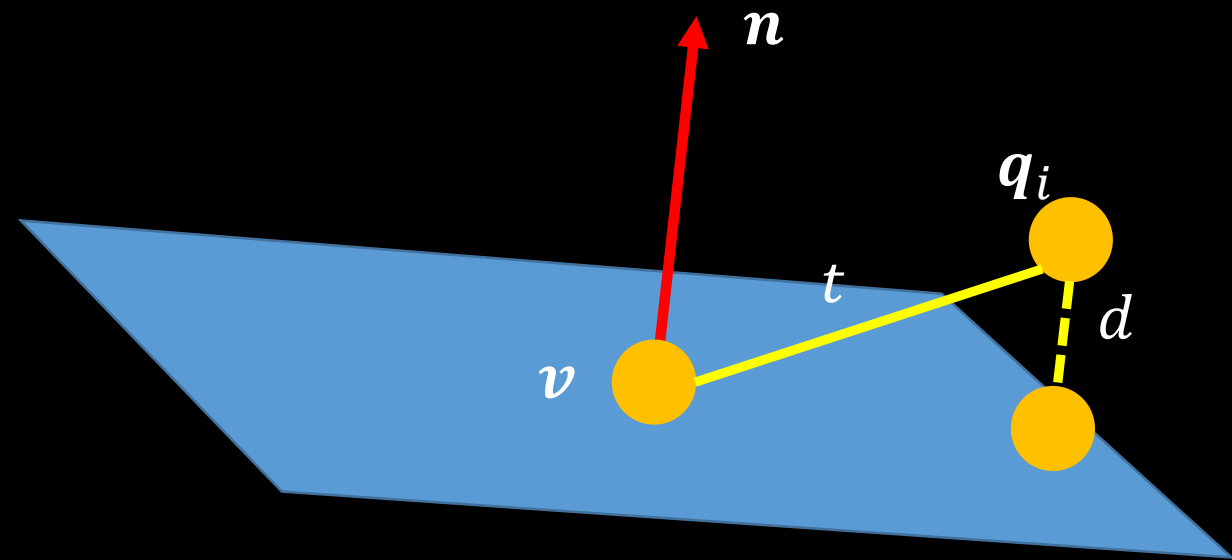
$$w_r = \exp(-d^2 / (2\sigma_r))$$

$$d_{\text{sum}} += w_s \cdot w_r \cdot d$$

$$K_{\mathbf{v}} += w_s \cdot w_r$$

end

return $\mathbf{v}^{\text{new}} = \mathbf{v} + \mathbf{n} \cdot d_{\text{sum}} / K_{\mathbf{v}}$



Detail

- Normal: weighted average of the normals
 - 1-ring neighborhood of the vertex.
 - k-ring neighborhood for extremely noisy data.
 - weight: the area of the triangles.
- Mesh shrinkage: volume preservation technique.
 - Scale the updated mesh to preserve the volume.
 - How to compute the volume?
- Boundary.
- Parameters: σ_s , σ_r , number of iterations.

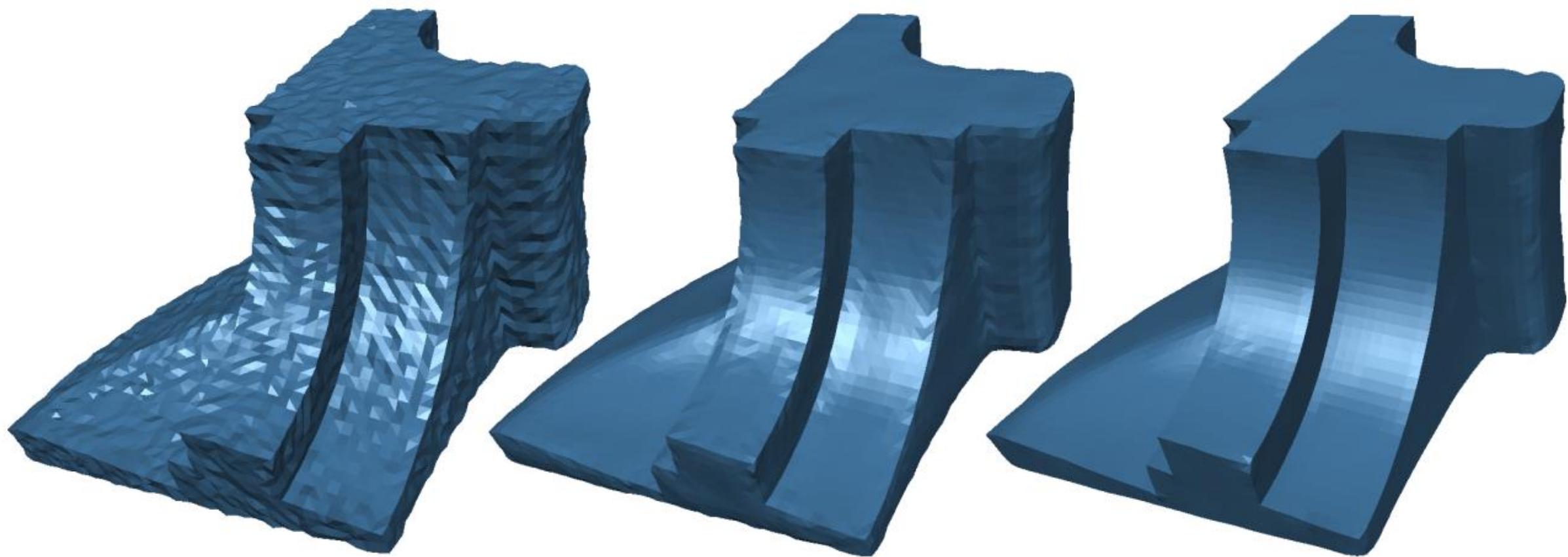


Figure 5: Results of denoising the Fandisk model. On the left is the input noisy model, in the middle is the results of [Jones et al. 2003], and on the right is our result.

Bilateral Normal Filtering

Paper: Bilateral Normal Filtering for Mesh Denoising

- The normals on facets are well-defined.
- Considers normals as a surface signal defined over the original mesh.
- A novel bilateral normal filter that depends on both spatial distance and signal distance.
- Recover vertex positions in global and non-iterative manner.

Bilateral Normal Filtering

$$\mathbf{n}(f_i) \leftarrow \frac{1}{K_p} \sum_{f_j \in \Omega(f_i)} A_j W_s(\|\mathbf{c}_i - \mathbf{c}_j\|) W_r(\|\mathbf{n}(f_i) - \mathbf{n}(f_j)\|) \cdot \mathbf{n}(f_j)$$

1. $\mathbf{n}(f_i)$: the normal of facet f_i .
2. \mathbf{c}_i : the center of facet f_i .
3. $\Omega(f_i)$: the neighbor facets of f_i .
4. $W_s(\|\mathbf{c}_i - \mathbf{c}_j\|)$: spatial distance.
5. $W_r(\|\mathbf{n}(f_i) - \mathbf{n}(f_j)\|)$: normal difference.
6. A_j : the area of facet f_j .

Mesh Denoising

- Given the normal on each facet, determine the vertex positions to match the normal as much as possible.
- Local and Iterative Scheme
 - update the normal field.
 - update the vertex positions.
- Global and Non-Iterative Scheme
 - Energy minimization.

Local and Iterative Scheme

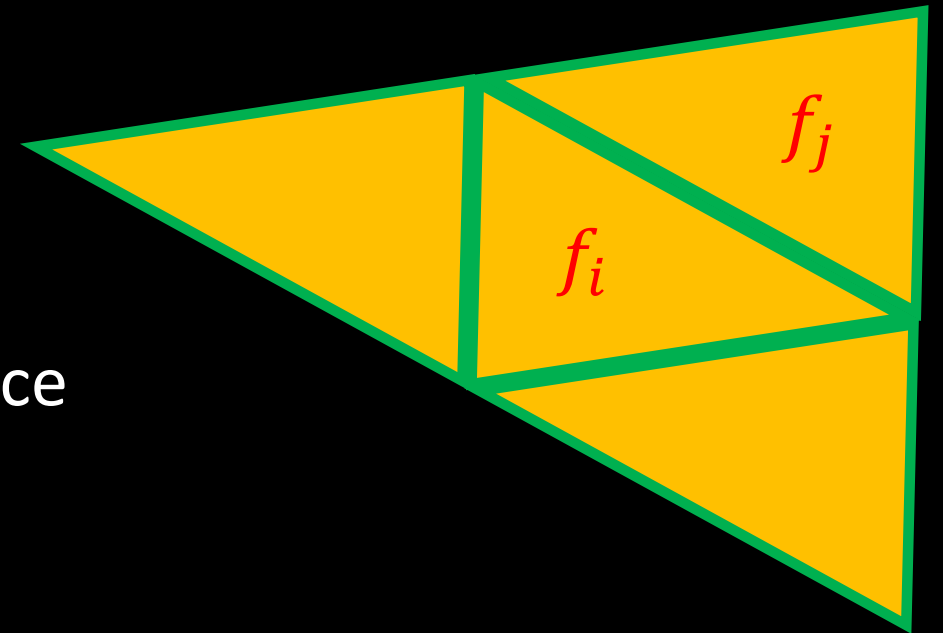
- Normal Updating:

- $\mathbf{n}_i^{t+1} \leftarrow \frac{1}{K_p} \sum_{f_j \in \Omega(f_i)} A_j W_s W_r \cdot \mathbf{n}_j^t$

- Normalize the new normal after each iteration.

- Multiple iterations: increase the influence from a 1-ring neighborhood to a wider region, leading to a smoother mesh.

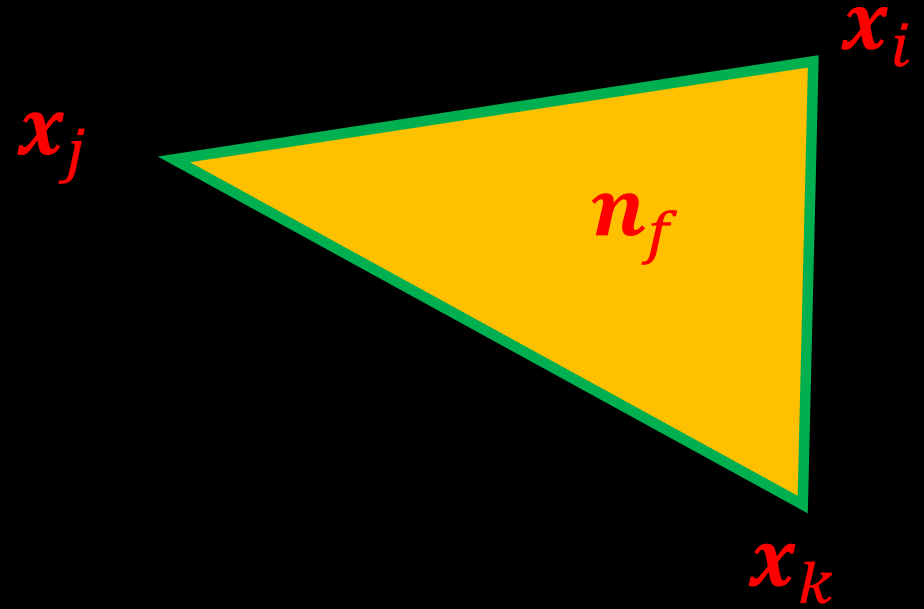
- Iteration number: user controls.



Local and Iterative Scheme

Vertex Updating:

$$\begin{cases} \mathbf{n}_f^T \cdot (\mathbf{x}_j - \mathbf{x}_i) = 0 \\ \mathbf{n}_f^T \cdot (\mathbf{x}_k - \mathbf{x}_j) = 0 \\ \mathbf{n}_f^T \cdot (\mathbf{x}_i - \mathbf{x}_k) = 0 \end{cases}$$



Energy:

$$E = \sum_{f_k} \sum_{i,j \in f_k} \left(\mathbf{n}_k^T \cdot (\mathbf{x}_j - \mathbf{x}_i) \right)^2$$

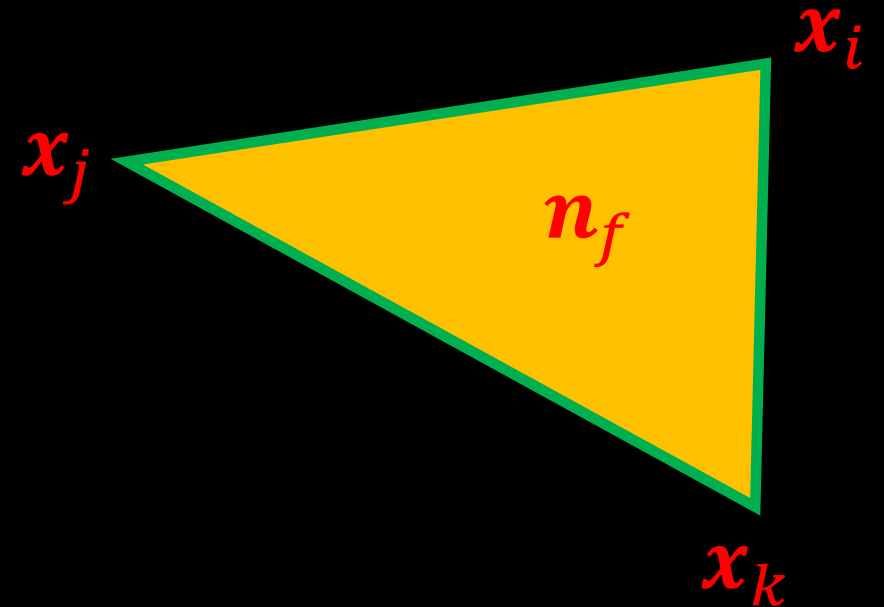
Linear system.

Local and Iterative Scheme

Each time, fix other vertex, update one vertex (Gauss–Seidel iteration). **Why???**

$$\mathbf{x}_i^{new} = \mathbf{x}_i + \frac{1}{N_i} \sum_{f_j \in \Omega(i)} \mathbf{n}_j \cdot (\mathbf{n}_j^T \cdot (\mathbf{c}_j - \mathbf{x}_i))$$

1. No need to determine a suitable step size.
2. Not computationally expensive. No need to solve a linear system.
3. Iteration number: user control.



Global and Non-Iterative Scheme

- Energy minimization:

$$E(\mathbf{n}) = (1 - \lambda)E_s + \lambda E_a$$

$$E_s = \sum_i A_i \|(\mathbf{L}\mathbf{n}^{new})_i\|_2^2$$

$$E_a = \sum_i A_i \|\mathbf{n}_i^{new} - \mathbf{n}_i\|_2^2$$

1. L : Uniform weighting or cotangent weighting.
2. λ : a parameter to balance the E_s and E_a .

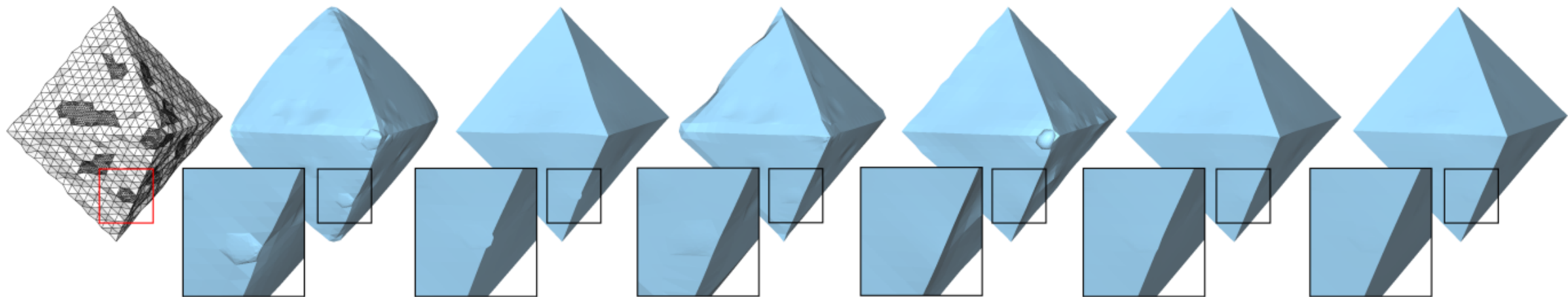


Fig. 1: Our mesh denoising schemes based on bilateral normal filtering produce better results than the state-of-the-art methods at challenging regions with sharp features or irregular surface sampling. From left to right: an input CAD-like model with random subdivision, denoising results with bilateral mesh filtering (vertex-based) [1], unilateral normal filtering [2], probabilistic smoothing [3], prescribed mean curvature flow [4], our local, iterative scheme, and our global, non-iterative scheme. All the meshes in the paper are flat-shaded to show faceting.

Manifold Harmonics

Paper: Spectral Geometry Processing with Manifold Harmonics

- 1D Fourier Transform:

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

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

spatial domain $f(x)$ \Leftrightarrow frequency domain $F(\omega)$

1D Fourier Transform

- An intuitive geometric interpretation of $f(x)$:
 - an element of a certain vector space
 - $\langle f, g \rangle = \int_{-\infty}^{+\infty} f(x) \overline{g(x)} dx$
 - $g(x) = e_{\omega}(x) := e^{2\pi i \omega x} = \cos(2\pi \omega x) + i \cdot \sin(2\pi \omega x)$
- $e_{\omega}(x)$: frequency-related orthogonal basis
 - $f(x) = \int_{-\infty}^{+\infty} \langle f(x), e_{\omega}(x) \rangle e_{\omega}(x) d\omega$
 - It describes how much of the basis function $e_{\omega}(x)$ is contained in $f(x)$.
- Low-pass filter:
 - cutting off all frequencies above a user-defined threshold ω_{max}
 - $\tilde{f}(x) = \int_{-\omega_{max}}^{+\omega_{max}} \langle f(x), e_{\omega}(x) \rangle e_{\omega}(x) d\omega$

Manifold Harmonics

- How to generalize the 1D Fourier Transform to 2-manifold surface?
- **sine and cosine functions \Leftrightarrow eigenfunctions of the Laplace operator**
 - $\Delta e_\omega(x) = -(2\pi\omega)^2 e_\omega(x)$
 - Definition of eigenfunctions of the Laplace operator:
 - $\Delta e_i = \lambda_i e_i$, similar to the eigenvector of a matrix
- Choose eigenfunctions of the Laplace-Beltrami operator on 2-manifold surfaces as **generalized basis functions**.

Manifold Harmonics

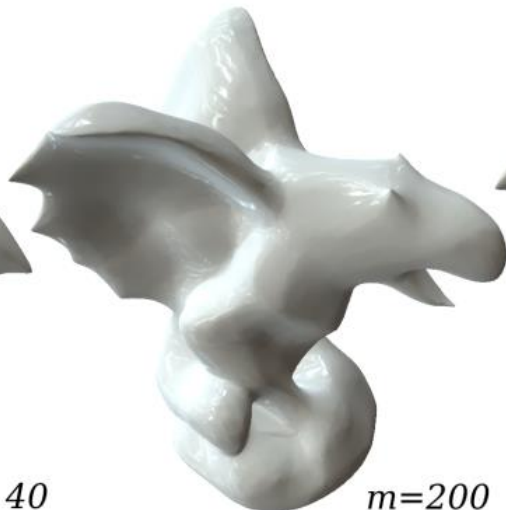
- Discrete Laplace-Beltrami operator: L
 - Symmetry, uniform or cotangent
- **Eigenfunctions become the eigenvectors of L**
 - an eigenvector can be considered a discrete sampling of a continuous eigenfunction on each vertex.
 - natural vibrations: eigenvectors of L
 - natural frequencies: eigenvalues of L
- L : symmetry and positive semi-definite
 - its eigenvectors build an orthogonal basis
 - For each vector $\mathbf{f} = (f_1, \dots, f_n)^T$: $\mathbf{f} = \sum_{i=1}^n \langle \mathbf{f}, \mathbf{e}_i \rangle \mathbf{e}_i$

Manifold Harmonics

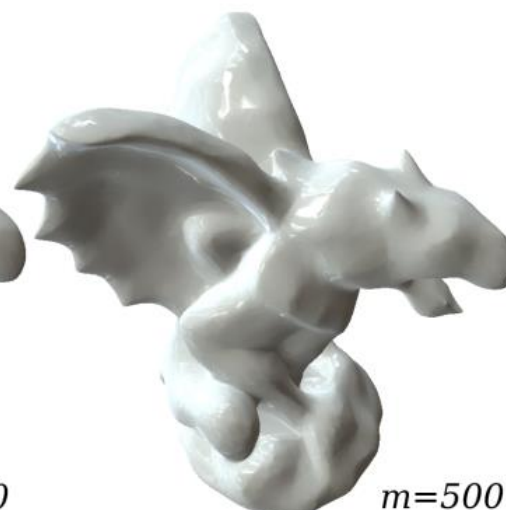
- Low-pass filter:
 - $f = \sum_{i=1}^m \langle f, e_i \rangle e_i$ where $m < n$.
 - Replace f with vertex coordinates.



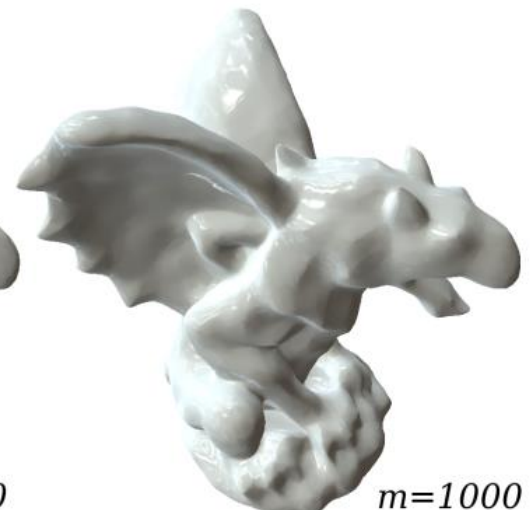
$m = 40$



$m=200$



$m=500$



$m=1000$

Other filters

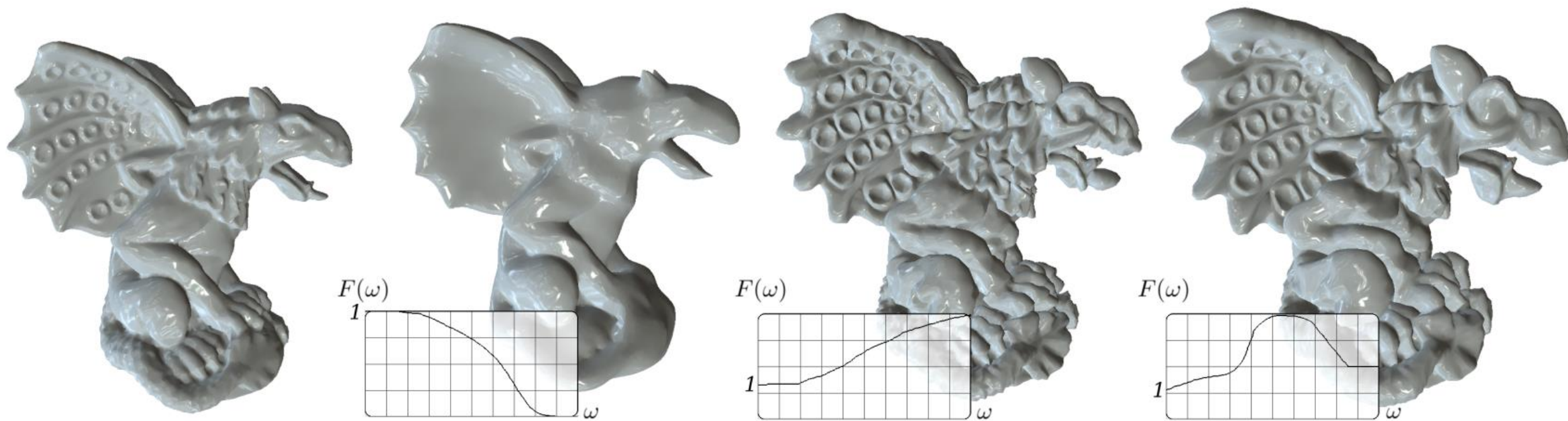


Figure 5: *Low-pass, enhancement and band-exaggeration filters. The filter can be changed by the user, the surface is updated interactively.*

Discussion

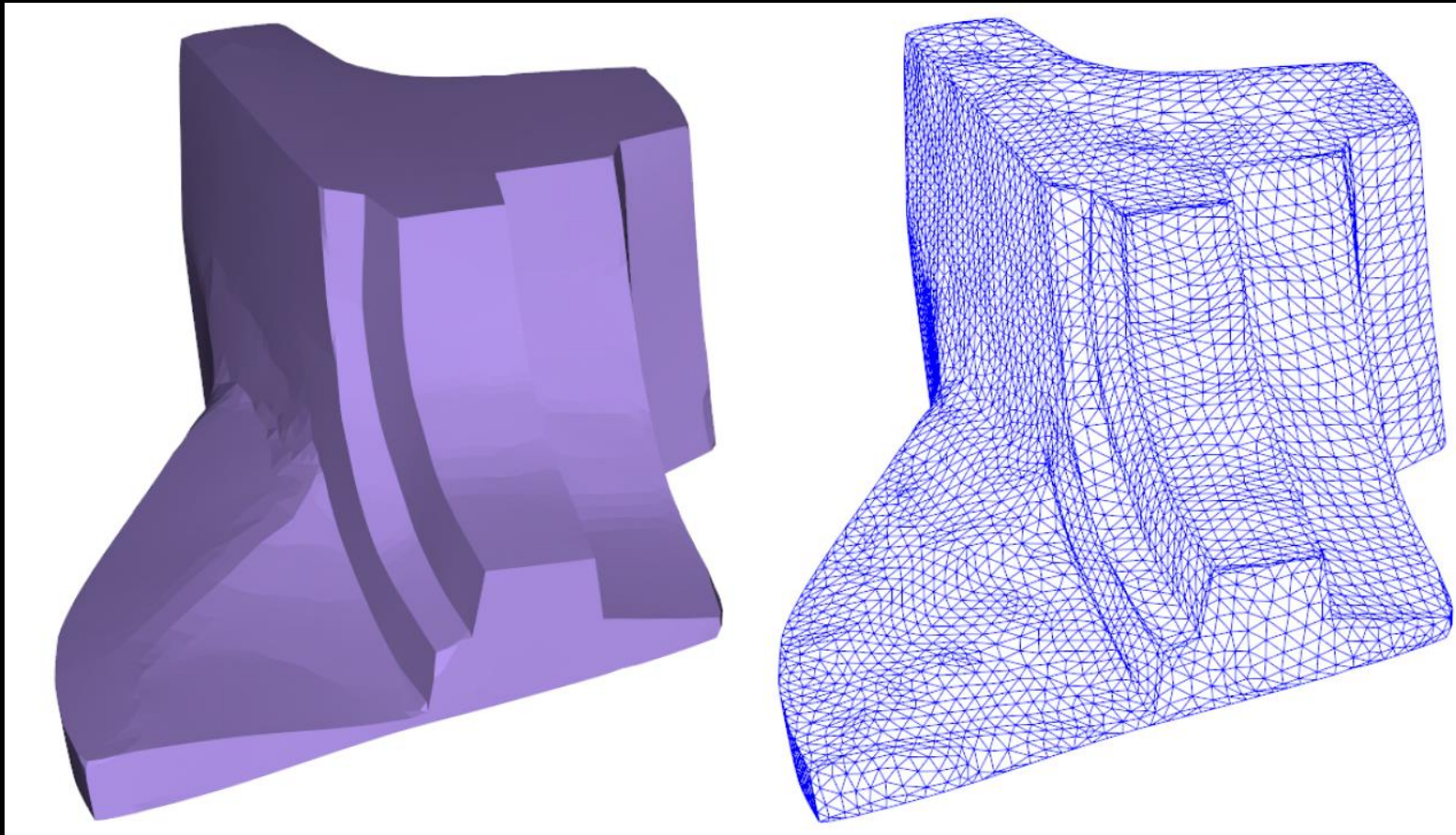
- Computationally expensive
 - eigenvector and eigenvalue
 - Paper: Fast Approximation of Laplace-Beltrami Eigenproblems, SGP 2018
- A very useful representation of triangle mesh:
 - 3D printing
 - Reduced-Order Shape Optimization Using Offset Surfaces, SIGGRAPH 2015
 - Non-Linear Shape Optimization Using Local Subspace Projections, SIGGRAPH 2016
 - Face modeling
 - Use small m to represent the basic shape
 - Use Laplacian coordinate to represent the details
 - ...

Outline

- Filter-based methods
- Optimization-based methods
- Data-driven methods

Prior

- The model consists of flat regions.



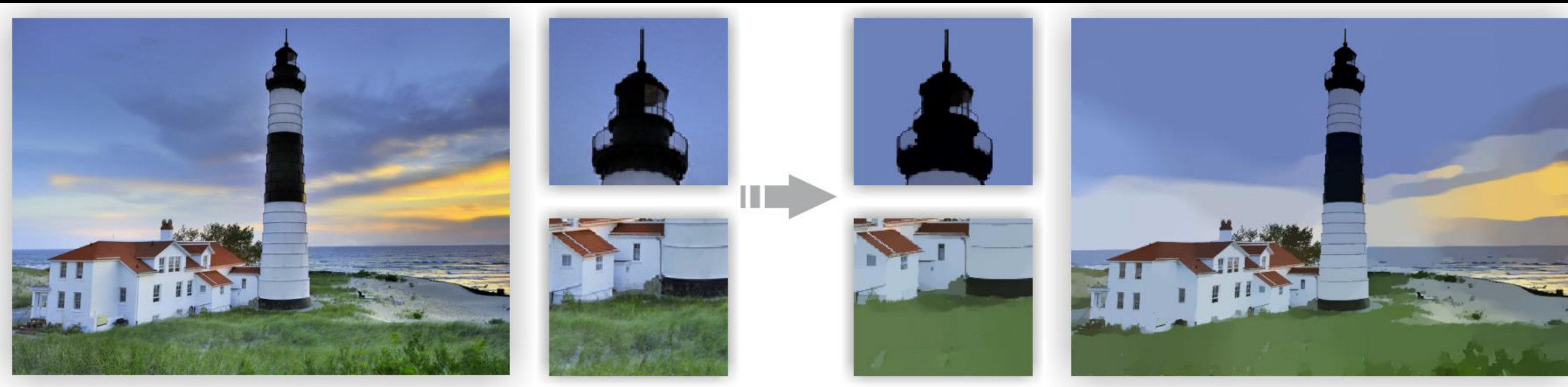
L_0 smoothing

Paper: Mesh denoising via L_0 minimization

- The method maximizes the flat regions of the model and gradually removes noise while preserving sharp features.

L_0 minimization for images

Paper: Image smoothing via L_0 gradient minimization



L_0 minimization for images

Paper: Image smoothing via L_0 gradient minimization

- Energy:

$$|c - c^*|^2 + |\nabla c|_0$$

- c : a vector of pixel colors
- c^* : original image colors
- ∇c : a vector of gradients of these colors
- $|\nabla c|_0$: L_0 norm of ∇c

Difficult to solve!

Optimization method

- Auxiliary variables δ :

$$\min_{c, \delta} |c - c^*|^2 + \beta |\nabla c - \delta|^2 + \lambda |\delta|_0$$

- Alternating optimization:

- 1. Fix c , solve δ – subproblem:

$$\min_{\delta} \beta |\nabla c - \delta|^2 + \lambda |\delta|_0$$

Analytic solution

- 2. Fix δ , solve c – subproblem:

$$\min_{c, \delta} |c - c^*|^2 + \beta |\nabla c - \delta|^2$$

Quadratic

δ – subproblem

$$\delta_i = \begin{cases} 0, & \text{if } \beta(\nabla c_i)^2 \leq \lambda \\ \nabla c_i, & \text{otherwise} \end{cases}$$

1. If $\beta(\nabla c_i)^2 \leq \lambda$, non-zero δ_i yields:

$$\beta(\nabla c_i - \delta_i)^2 + \lambda|\delta_i|_0 = \beta(\nabla c_i - \delta_i)^2 + \lambda \geq \lambda \geq \beta(\nabla c_i)^2$$

When $\delta_i = 0$, $\beta(\nabla c_i - \delta_i)^2 + \lambda|\delta_i|_0 = \beta(\nabla c_i)^2$.

Thus, the minimum is achieved when $\delta_i = 0$.

2. If $\beta(\nabla c_i)^2 > \lambda$,

When $\delta_i = 0$, $\beta(\nabla c_i - \delta_i)^2 + \lambda|\delta_i|_0 = \beta(\nabla c_i)^2 > \lambda$.

When $\delta_i \neq 0$, the minimum is achieved when $\delta_i = \nabla c_i$ and is λ .



(a)

(b)

Figure 13: *Image abstraction and pencil sketching results. Our method removes the least important structures.*

Mesh denoising

- $c \rightarrow$ points p of input triangle mesh
- $\nabla c \rightarrow$ discrete differential operator?
- It is the key.

$$\nabla c$$

- Requirements:
 - $\nabla c = 0$ when the surface is flat
 - It is irrespective of the rotation or translation of the surface.
- Cot Laplacian operator.

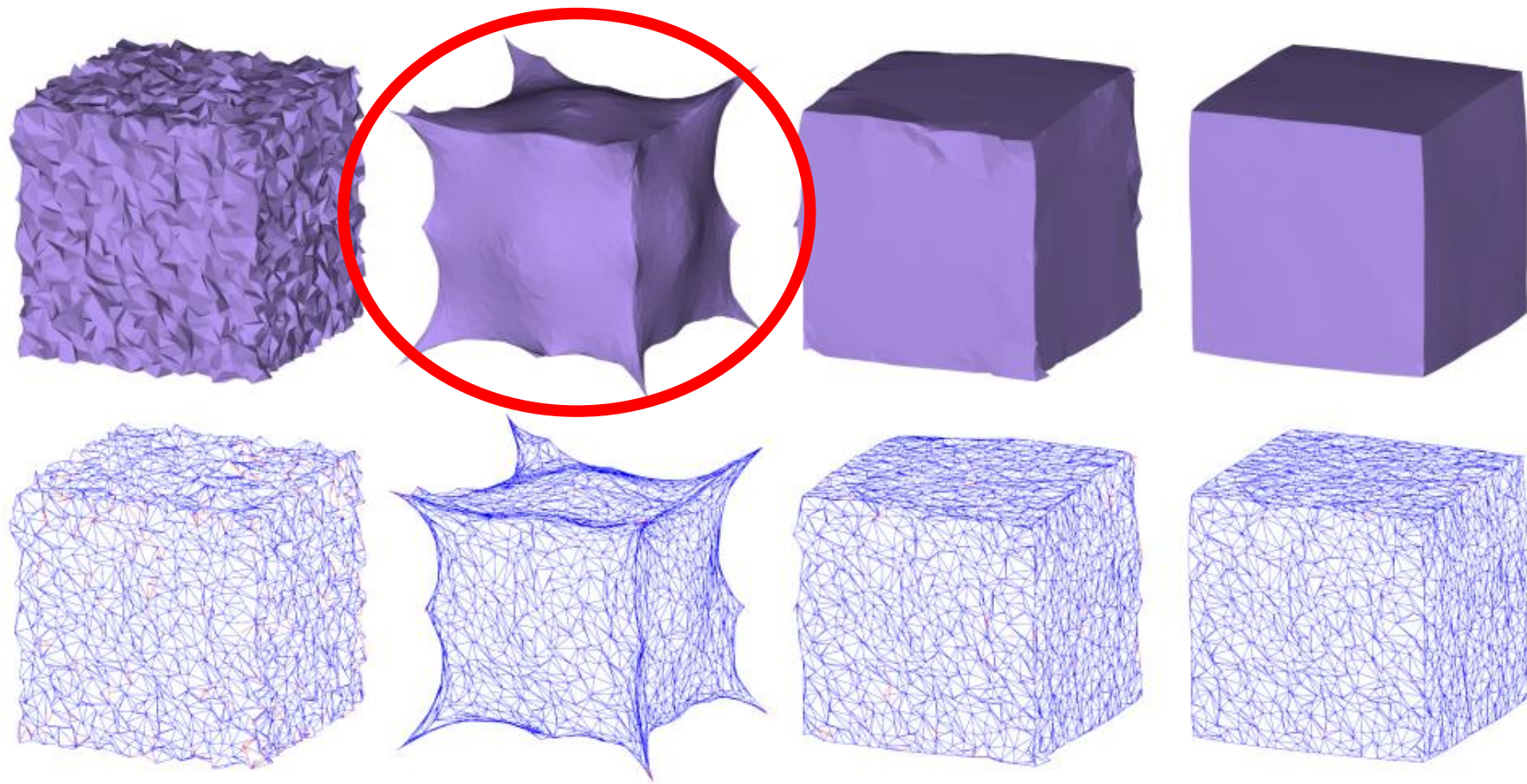
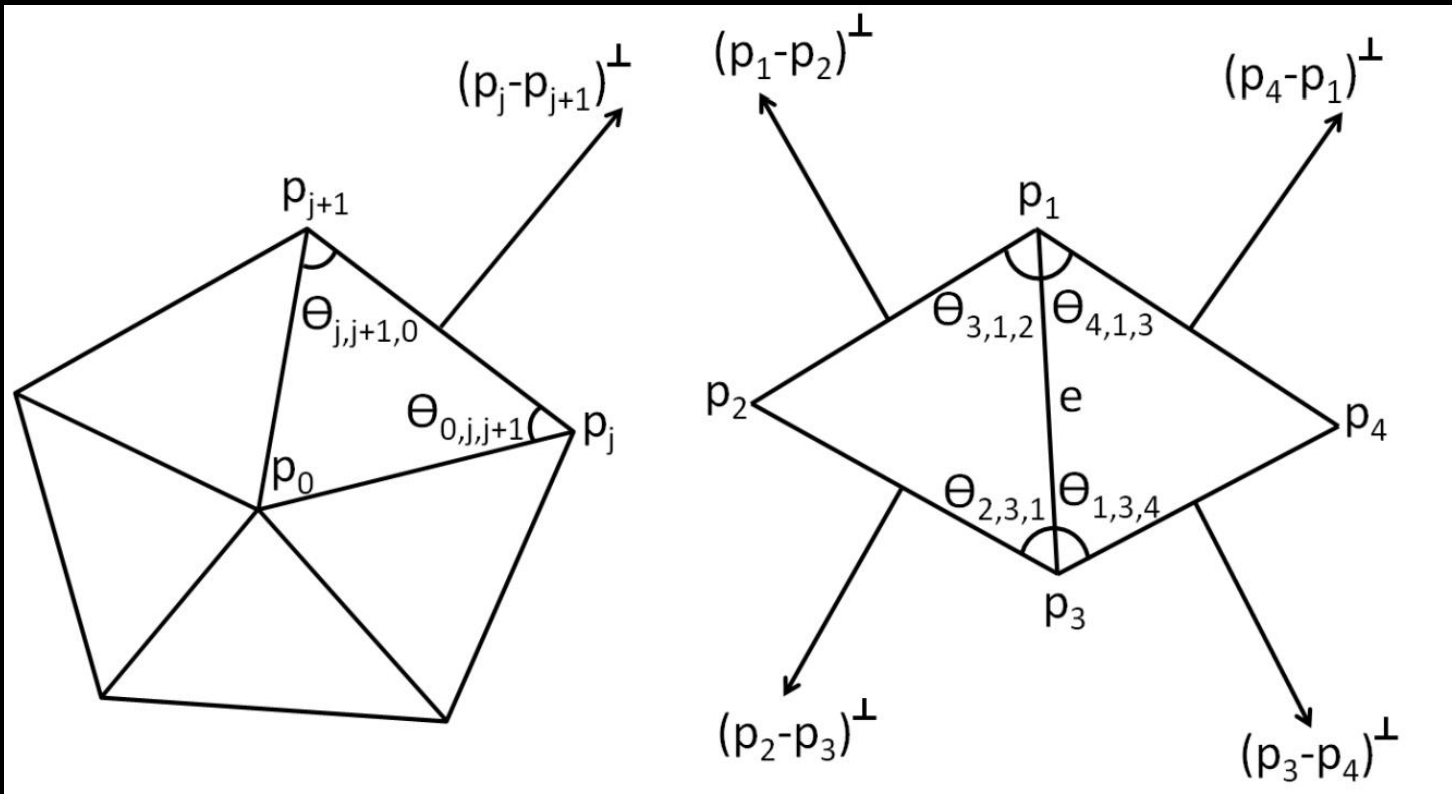


Figure 2: From left to right: noisy input surface with $\sigma = 0.3l_e$, vertex-based cotangent operator, our cotangent edge operator, our area-based edge operator. The bottom row shows wireframes with flipped triangles denoted by red edges. None of these results use regularization.

1. Fail to reproduce sharp feature
2. Shrink the surface

Differential Edge Operator

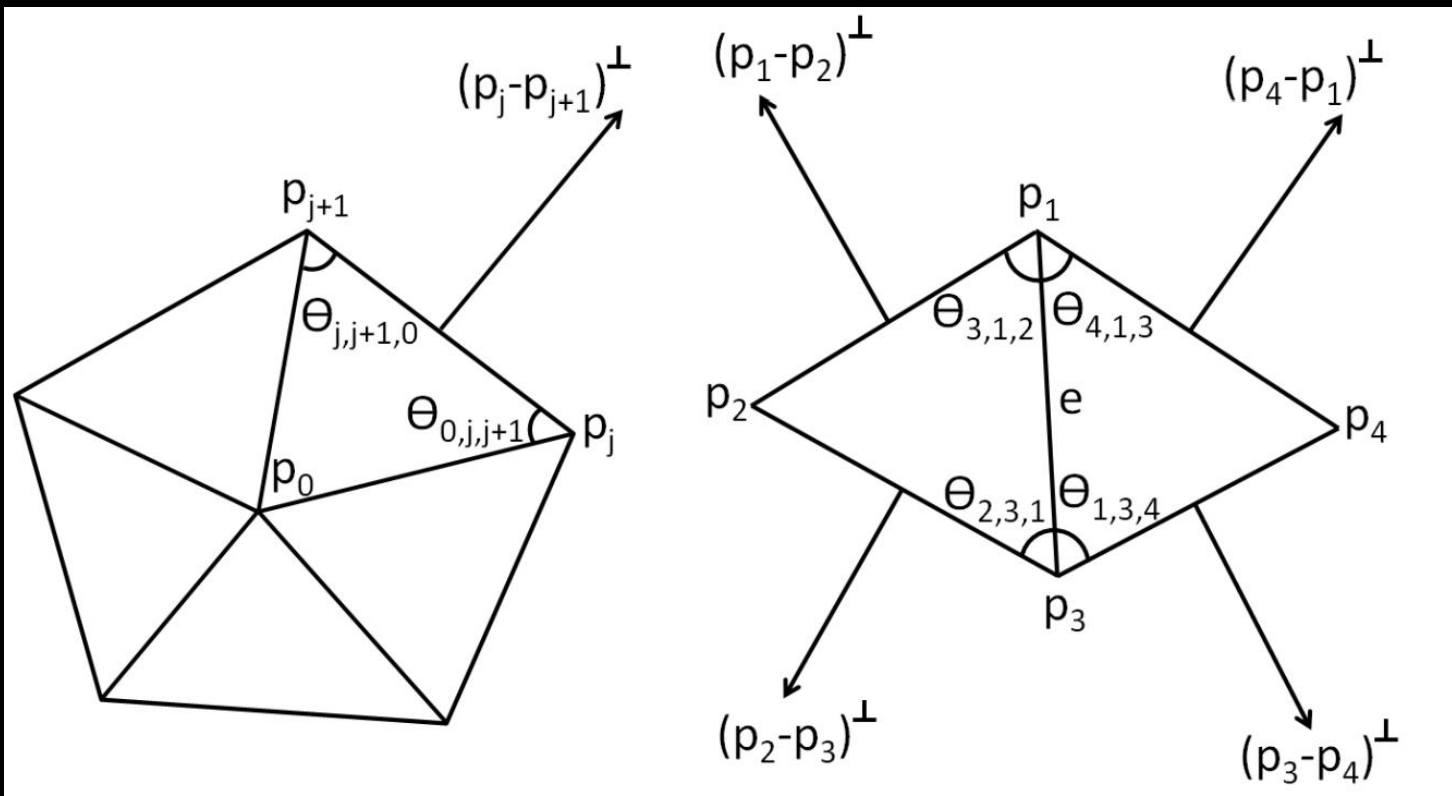
- $(p_j - p_{j+1})^\perp = (p_{j+1} - p_0) \cot \theta_{0,j,j+1} + (p_j - p_0) \cot \theta_{j,j+1,0}$
- Vertex-based cot Laplacian operator: $\sum_{j \in \Omega(p_0)} (p_j - p_{j+1})^\perp$



Differential Edge Operator

- Simialr to vertex version:

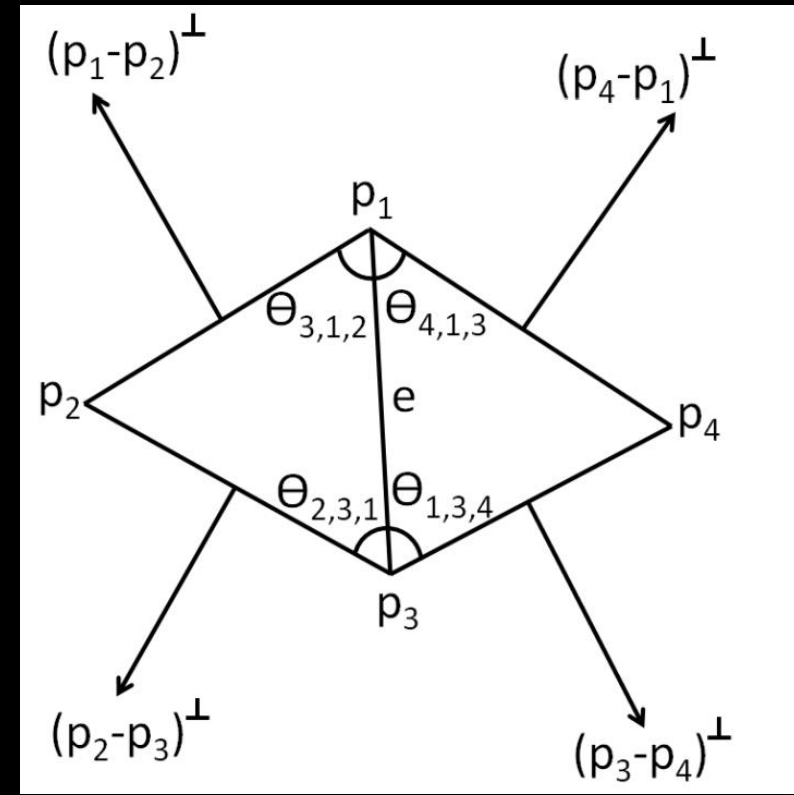
$$D(e) = \sum_{j \in \Omega(e)} (p_j - p_{j+1})^\perp$$

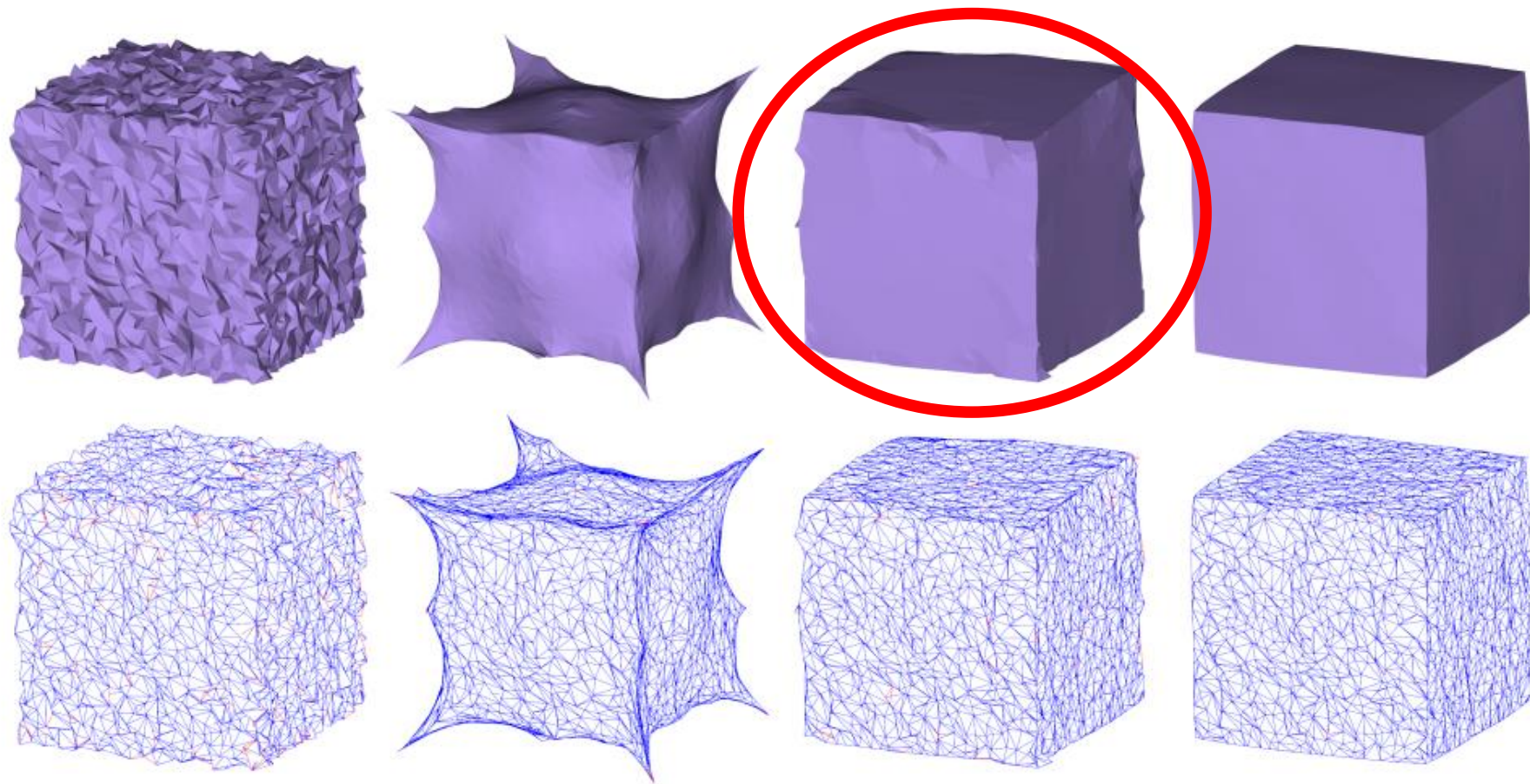


Differential Edge Operator

$$\bullet D(e) = \begin{bmatrix} -\cot \theta_{2,3,1} & -\cot \theta_{1,3,4} \\ \cot \theta_{2,3,1} & +\cot \theta_{3,1,2} \\ -\cot \theta_{3,1,2} & -\cot \theta_{4,1,3} \\ \cot \theta_{1,3,4} & +\cot \theta_{4,1,3} \end{bmatrix}^T \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$$

$$\bullet |D(e)| = 2 \sin\left(\frac{\gamma}{2}\right) |p_3 - p_1|$$





The issue stems from degenerate triangles where the cot weights approach infinity as an angle approaches zero.

Figure 2: From left to right: noisy input surface with $\sigma = 0.3l_e$, vertex-based cotangent operator, our cotangent edge operator, our area-based edge operator. The bottom row shows wireframes with flipped triangles denoted by red edges. None of these results use regularization.

Area-based edge operator

- LINEAR PRECISION:

- $0 = \sum_{j \in \Omega(i)} \omega_{ij} (p_i - p_j) = \sum_{k \in \Omega(i) \cup i} \omega'_{i,j} p_k$

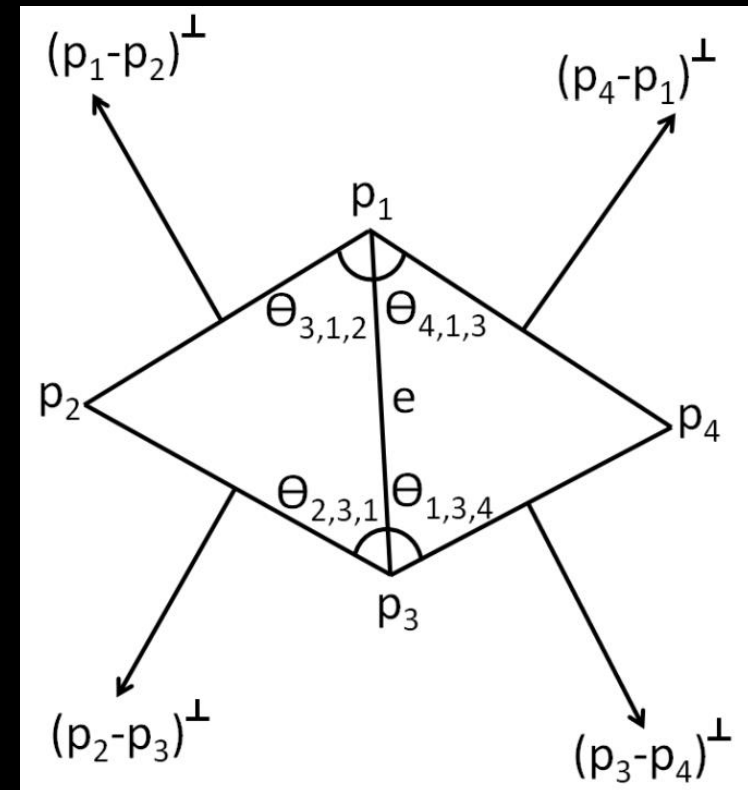
- At the same time: $0 = \sum_{k \in \Omega(i) \cup i} \omega'_{i,j}$

- Similarly: when p_j are planar:

$$0 = \sum_j \omega_j p_j, 0 = \sum_j \omega_j$$

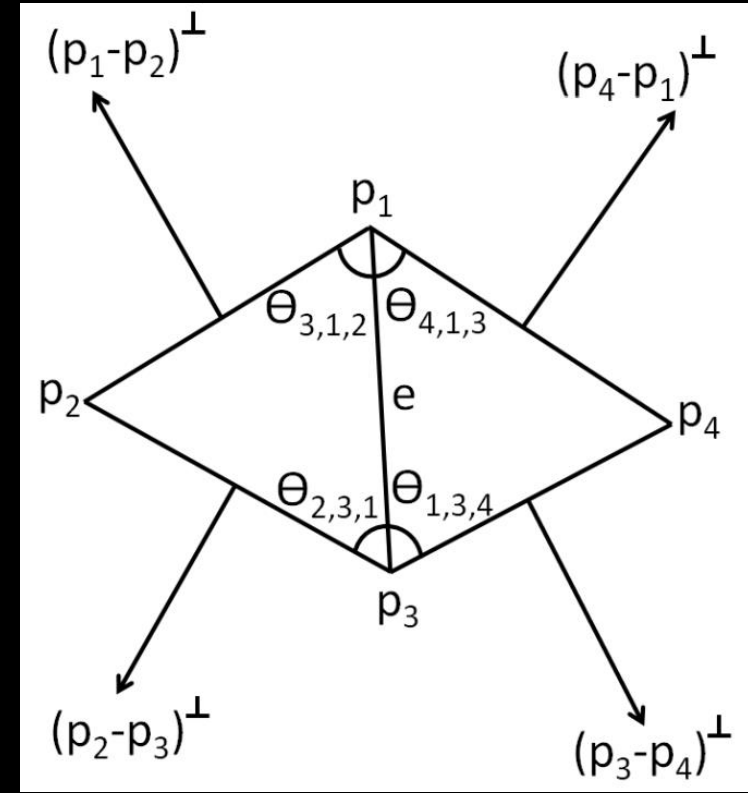
$$\omega_1 = -\Delta_{2,3,4}, \omega_2 = \Delta_{1,3,4},$$

$$\omega_3 = -\Delta_{1,2,4}, \omega_4 = \Delta_{1,2,3}$$



Area-based edge operator

- It is not scale-independent.
- Scaled by $\Delta_{1,3,4} + \Delta_{1,2,3}$
- How to compute $\Delta_{2,3,4}$ and $\Delta_{1,2,4}$?
 - an isometric unfolding of the surface around the shared edge



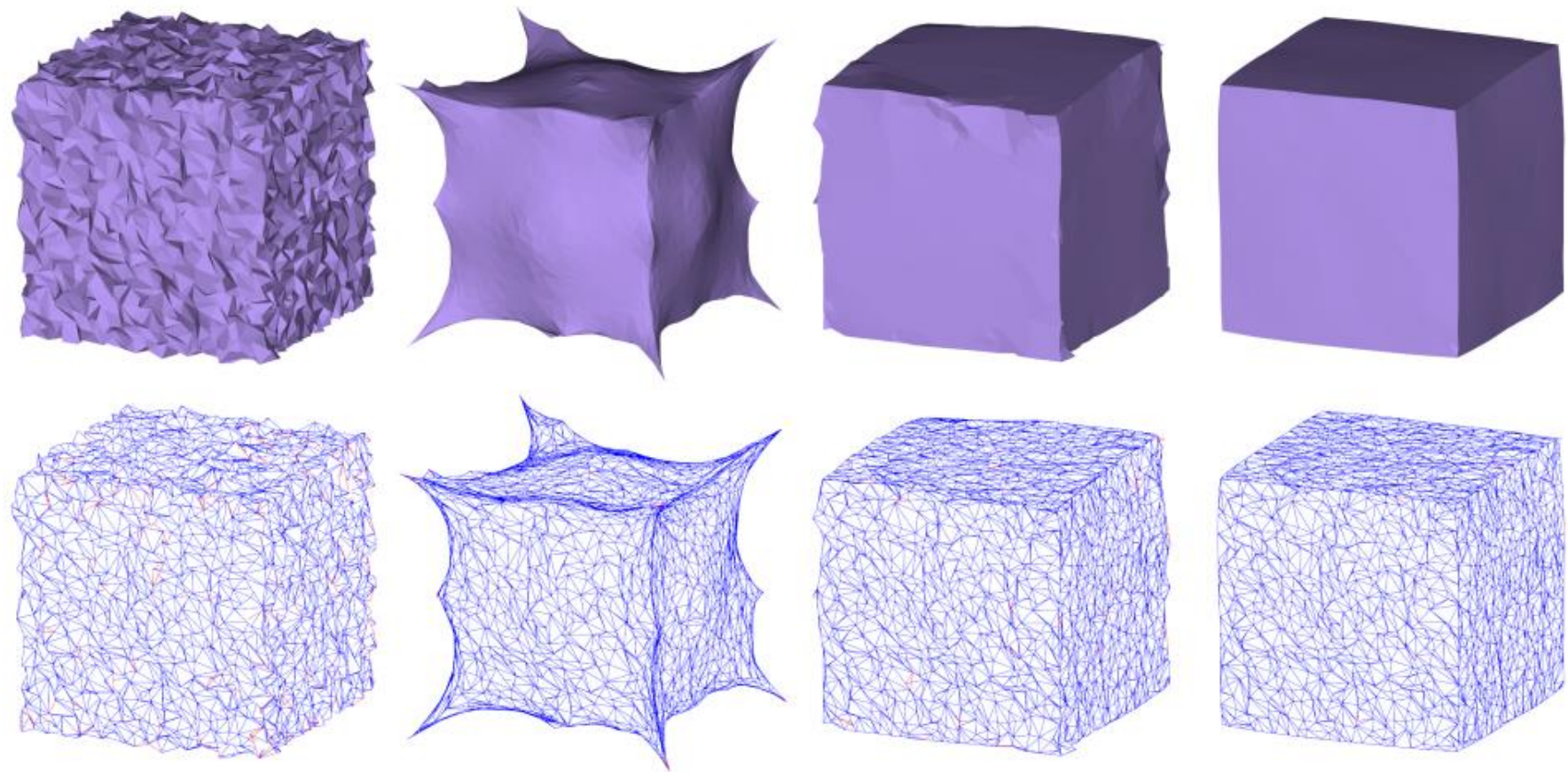


Figure 2: *From left to right: noisy input surface with $\sigma = 0.3l_e$, vertex-based cotangent operator, our cotangent edge operator, our area-based edge operator. The bottom row shows wireframes with flipped triangles denoted by red edges. None of these results use regularization.*

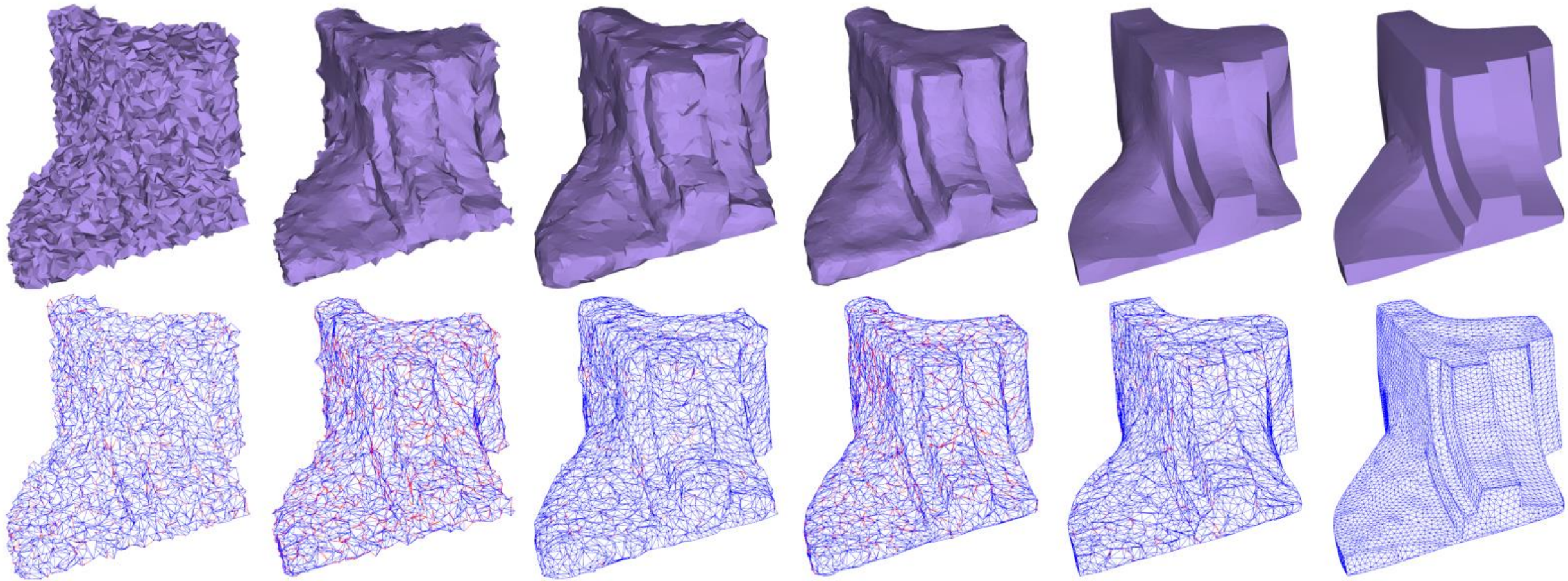


Figure 9: From left to right: the input mesh with large noise in random directions, bilateral filtering [Fleishman et al. 2003], prescribed mean curvature flow [Hildebrandt and Polthier 2004], mean filtering [Yagou et al. 2002], bilateral normal filtering [Zheng et al. 2011], our result. We show the wireframe of each surface below.

L_0 smoothing

Paper: Mesh denoising via L_0 minimization

- Paper:

<http://faculty.cs.tamu.edu/schaefer/research/L0Smoothing.pdf>

- Slides:

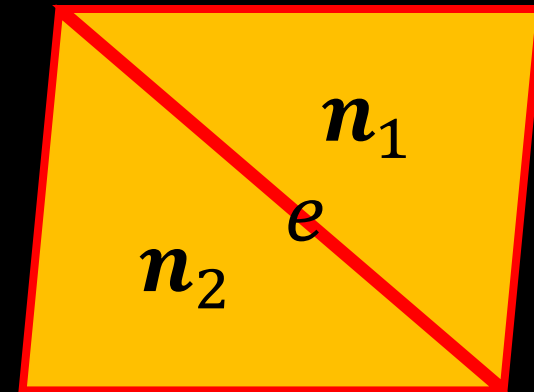
<http://faculty.cs.tamu.edu/schaefer/research/slides/L0Smoothing.pdf>

- Blog: <http://www.cnblogs.com/shushen/p/5113484.html>

Total Variation-based method

Paper: Variational Mesh Denoising using Total Variation and Piecewise Constant Function Space

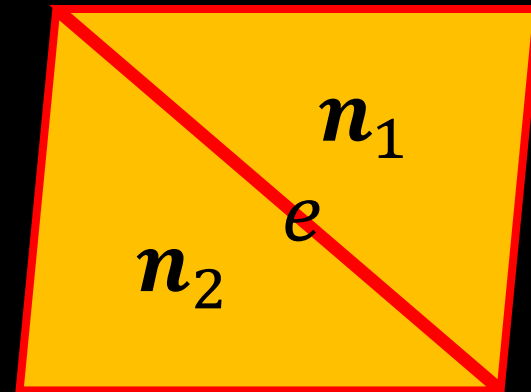
- Replace the vertex positions with the normals.
 - Facet normal filtering
 - Total Variation
 - Vertex updating
 - Iterative updating
- How to remove the noise and preserve the sharp feature?
 - Sharp feature is sparse.
 - Normal difference on edge is sparse.



Total Variation

$$\min E_{TV} + \alpha E_a$$
$$E_{TV} = \sum_e \omega_e \cdot l_e \sqrt{\|\nabla \mathbf{n}_e\|_2^2}$$
$$E_a = \sum_f \|\mathbf{n}_f - \mathbf{n}_f^{in}\|_2^2$$

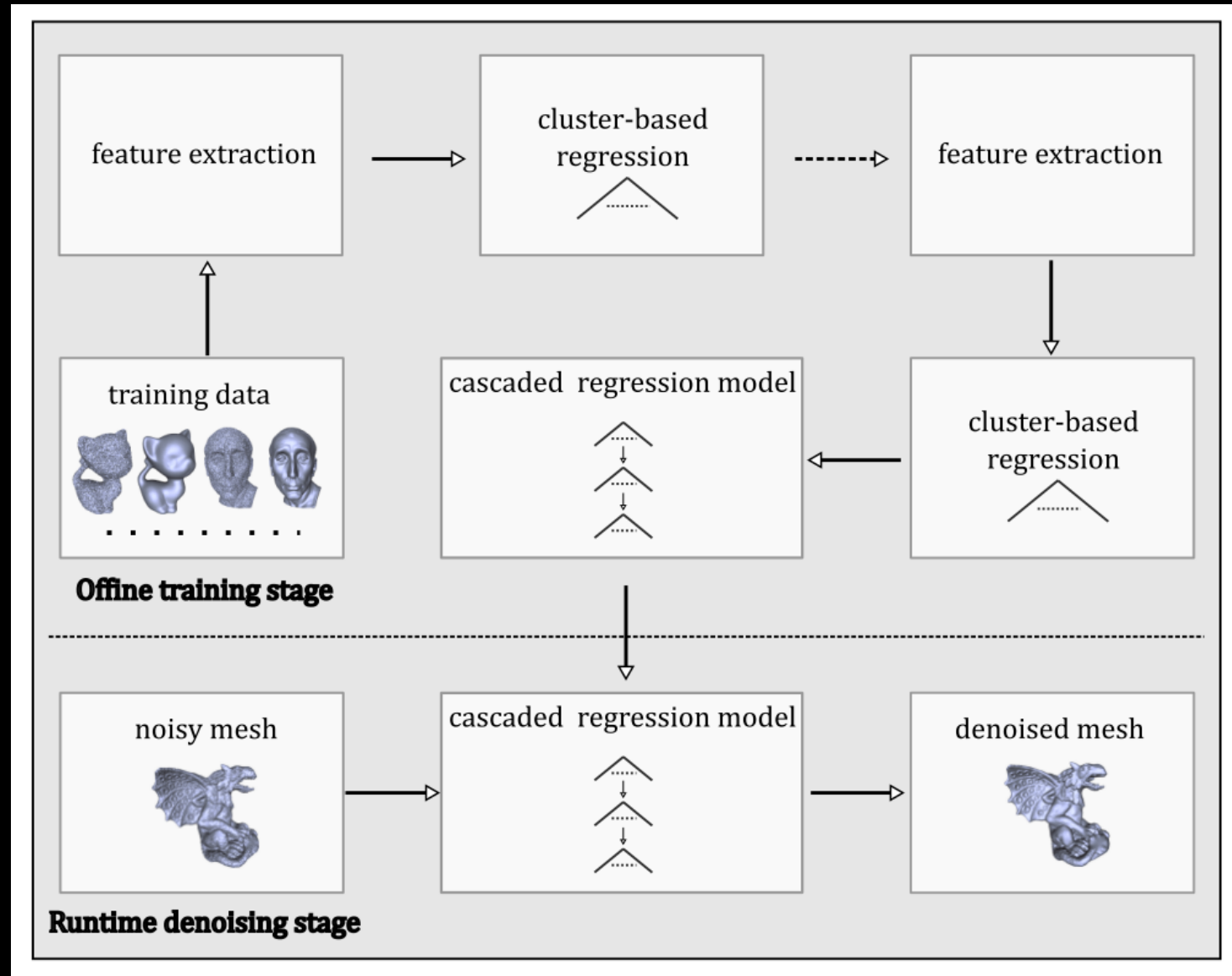
1. $\nabla \mathbf{n}_e = \mathbf{n}_1 - \mathbf{n}_2$
2. $\omega_e = \exp\left(-\|\mathbf{n}_1^{in} - \mathbf{n}_2^{in}\|_2^4\right)$



Outline

- Filter-based methods
- Optimization-based methods
- Data-driven methods

Mesh Denoising via Cascaded Normal Regression



A highly nonlinear function

Overview

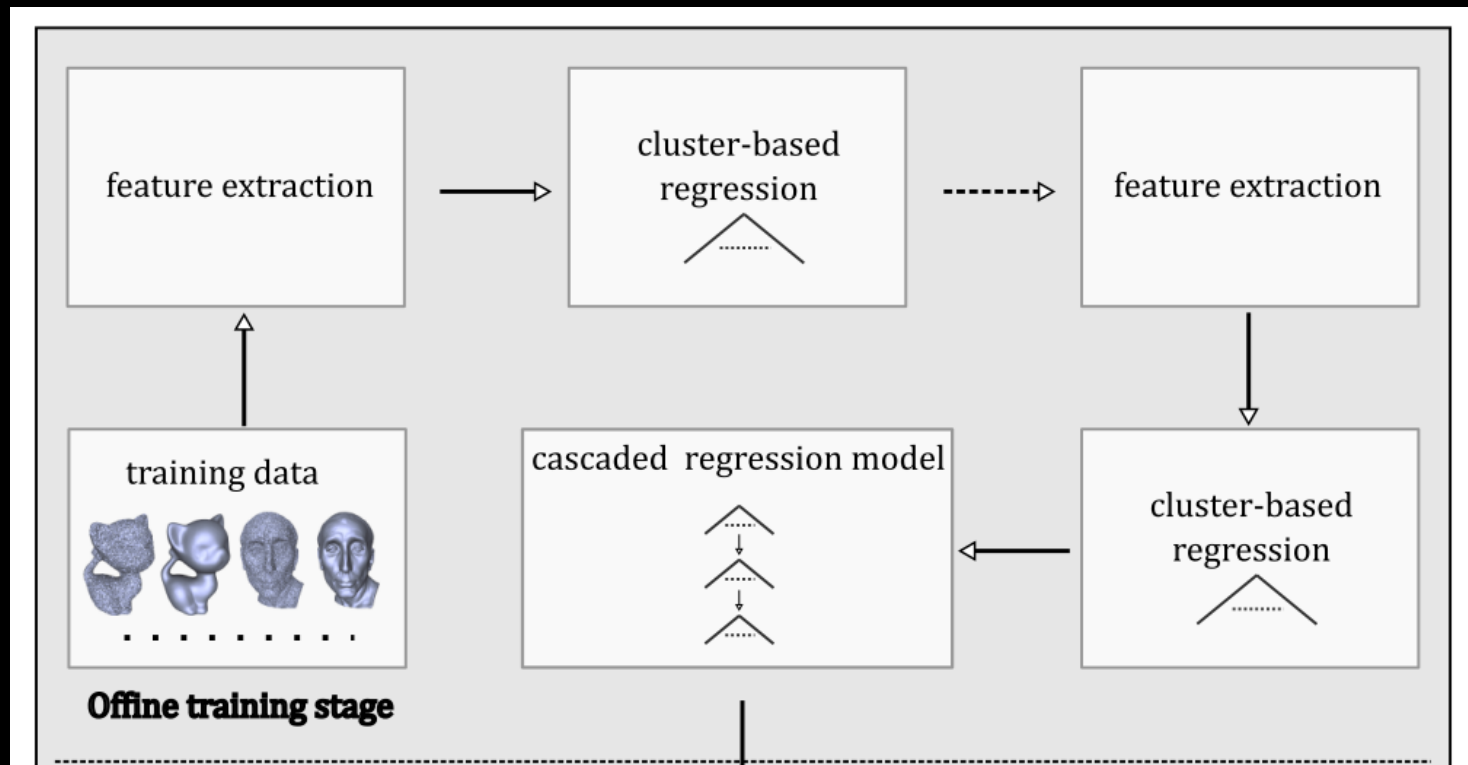
- Goal: learn the relationship between noisy geometry and the ground-truth geometry

$$n_f = \mathcal{F}(\Omega_f)$$

Ω_f : local noisy region

Cascaded Regression

- The output from the current regression function serves as the input of the next regression function.
- Each regression function: a neural network with a single hidden layer



Offline training stage

- A training pair: (S_i, \bar{n}_i)

S_i : filtered facet normal descriptor (FND) of i^{th} facet

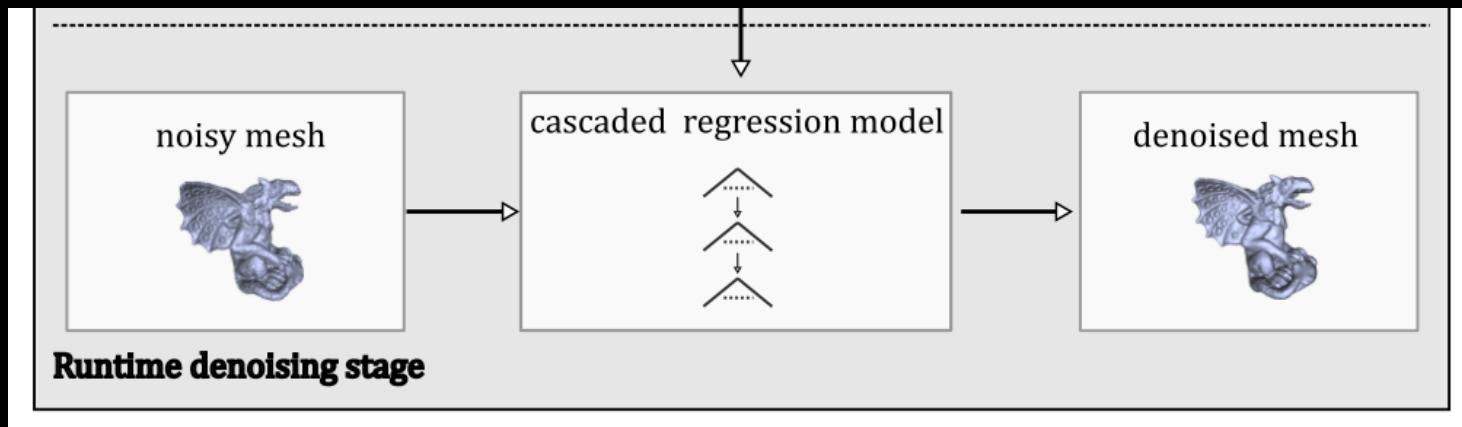
\bar{n}_i : ground-truth facet normal

- Goal: learn the function:

$$\mathcal{F}: S_i \rightarrow \bar{n}_i, \forall i$$

Runtime denoising stage

- Extract FND for each facet
- Apply \mathcal{F} to obtain new normal for each facet
- Recover vertices with known normal



Bilateral Normal Filtering

$$\mathbf{n}_i^{k+1} \leftarrow \frac{1}{K_p} \sum_{f_j} A_j W_s(\|\mathbf{c}_i - \mathbf{c}_j\|) W_r(\|\mathbf{n}_i^k - \mathbf{n}_j^k\|) \cdot \mathbf{n}_j^k$$

Parameters: σ_s, σ_r , iteration number K

- Bilateral filtered facet normal descriptor (B-FND)

S_i

$$:= \left(\mathbf{n}_i^1(\sigma_{s_1}, \sigma_{r_1}), \dots, \mathbf{n}_i^1(\sigma_{s_L}, \sigma_{r_L}), \mathbf{n}_i^2(\sigma_{s_1}, \sigma_{r_1}), \dots, \mathbf{n}_i^2(\sigma_{s_L}, \sigma_{r_L}), \dots \right)$$

Guided bilateral filter (Joint bilateral filter)

$$\mathbf{n}_i^{k+1} \leftarrow \frac{1}{K_p} \sum_{f_j} A_j W_s(\|\mathbf{c}_i - \mathbf{c}_j\|) W_r(\|\mathbf{g}(\mathbf{n}_i^k) - \mathbf{g}(\mathbf{n}_j^k)\|) \cdot \mathbf{n}_j^k$$

In this paper, $\mathbf{g}(\mathbf{n}_i^k) = \frac{1}{K_p} \sum_{f_j} A_j W_s(\|\mathbf{c}_i - \mathbf{c}_j\|) \cdot \mathbf{n}_j^k$ Gaussian normal filter

- Guided filtered facet normal descriptor (G-FND)

S_i^g

$$:= \left(\mathbf{n}_{g,i}^1(\sigma_{s_1}, \sigma_{r_1}), \dots, \mathbf{n}_{g,i}^1(\sigma_{s_L}, \sigma_{r_L}), \right)$$

Training data

- A dataset: $D = \{S_i, \bar{n}_i\}_{i=1}^N$
- First Partition the training data into K_c clusters via a k-means algorithm
- For each cluster D_l : 85% the training set D_{l1} , 15% validation set D_{l2}

Cluster-based regression

- Cost function:

$$E := \sum_{i \in D_{l_1}} \|\Lambda(\Phi_l(S_i)) - \bar{n}_i\|^2 + \lambda E_{reg}$$

E_{reg} : commonly used L2 regularization term of unknown parameters

Φ_l : regression function as a single-hidden layer feed forward network
 N_r hidden nodes

$$\Phi_l(S) = \sum_{k=1}^{N_r} \exp\left(-\|W_{l,k}^T \bar{S} - b_{l,k}\|^2\right) \mathbf{a}_{l,k}$$

\bar{S} : feature standardization version of S

$$W_{l,k}^T \in R^{3LK}, b_{l,k} \in R, \mathbf{a}_{l,k} \in R^3$$

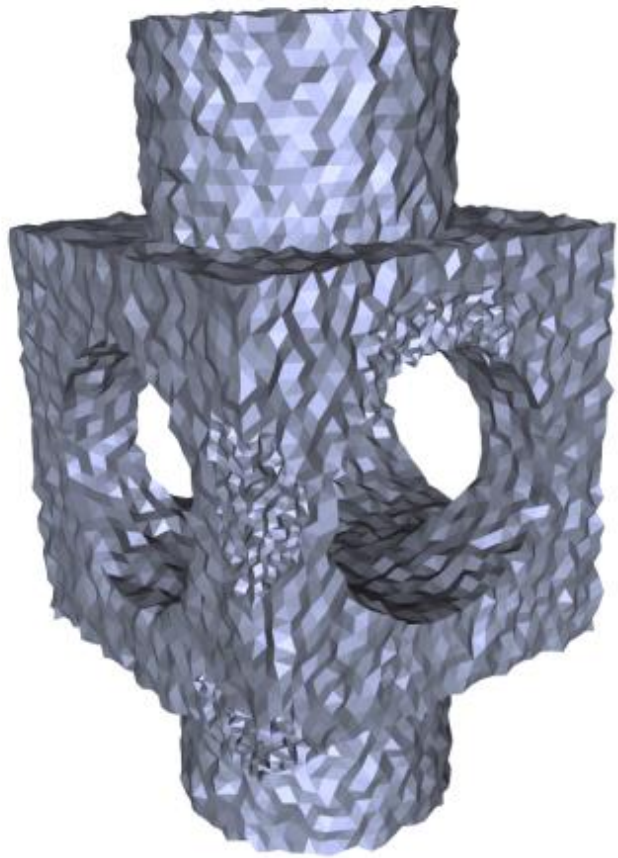
Regression function

$$\mathcal{F}(S) := \Phi_l(S), \text{ if } \|S - c_l\| \leq \|S - c_k\|, \forall k.$$

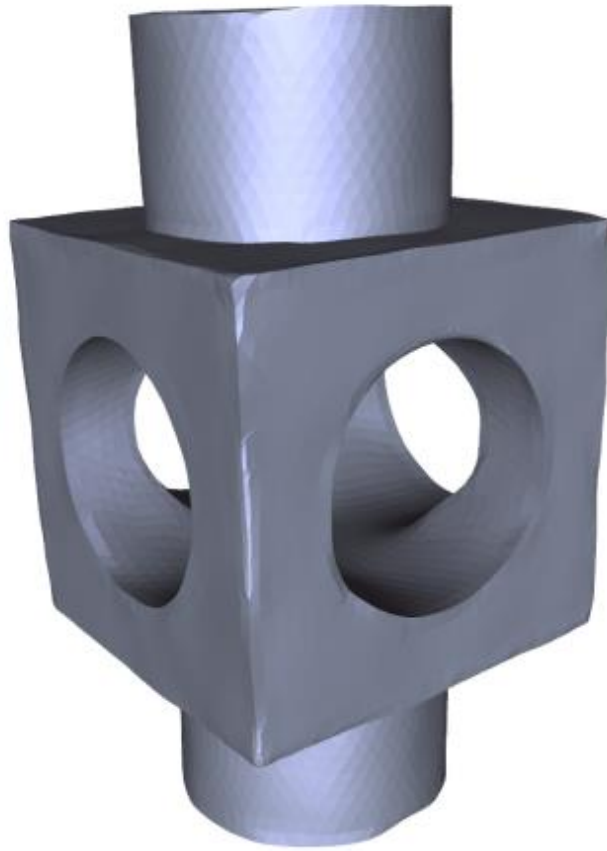
c_l : cluster center of D_l .

Cascaded scheme

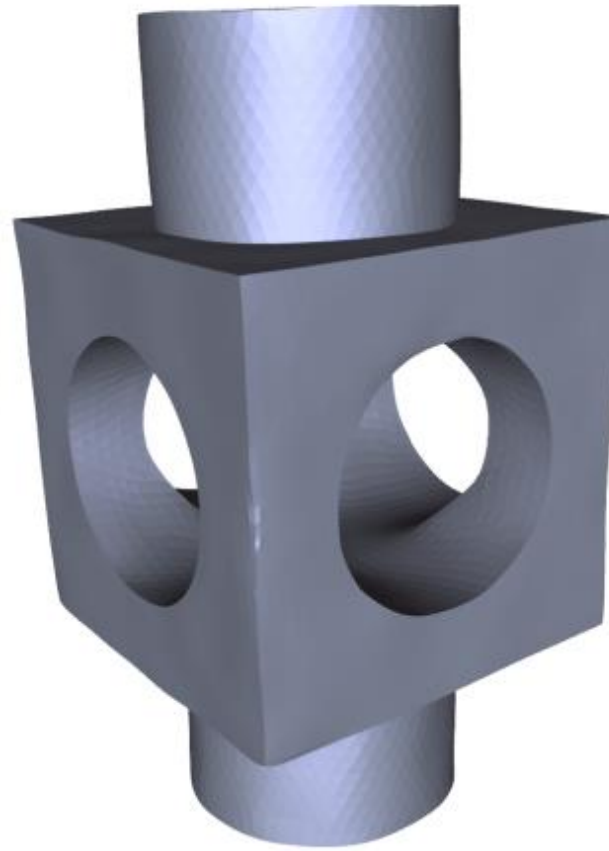
- G-FND in the first regression function



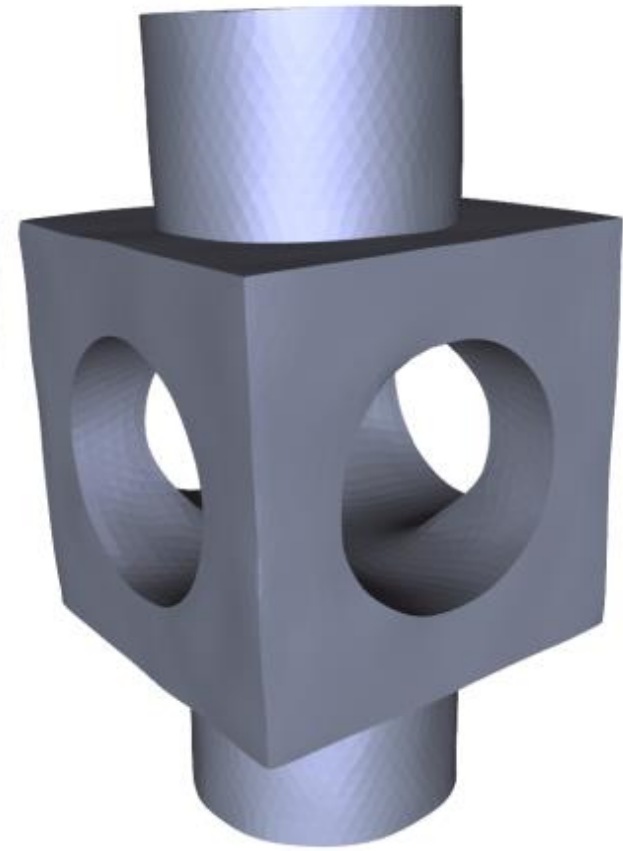
(a) Input



(b) G-FND



(c) B-FND

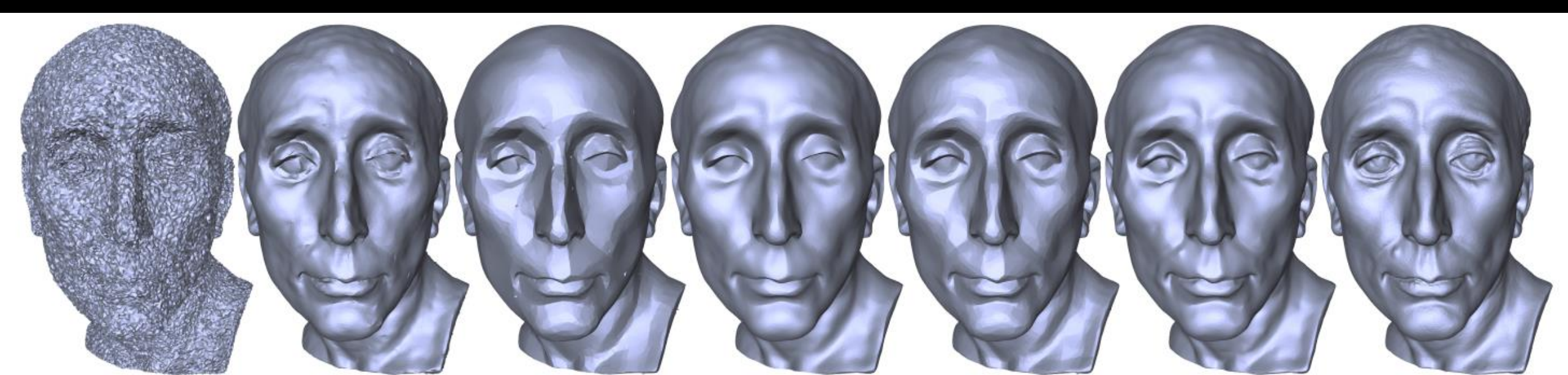


(d) Both

Choice of hyperparameters

- $\sigma_s: \{\bar{l}_e, 2\bar{l}_e\}$, \bar{l}_e is the average edge length.
- $\sigma_r: \{0.1, 0.2, 0.35, 0.5, \infty\}$
- $K = 1$

- 3 cascaded regressions are enough to generate good results.
- $K_c = 4$ after testing.
- $N_r = 20$ after testing.



(a) Noisy input

(b) Bilateral normal

(c) L_0 smoothing

(d) Guided normal

(e) Bayesian

(f) Our method

(g) Ground-truth

Mesh Parameterizations

Xiao-Ming Fu

Outline

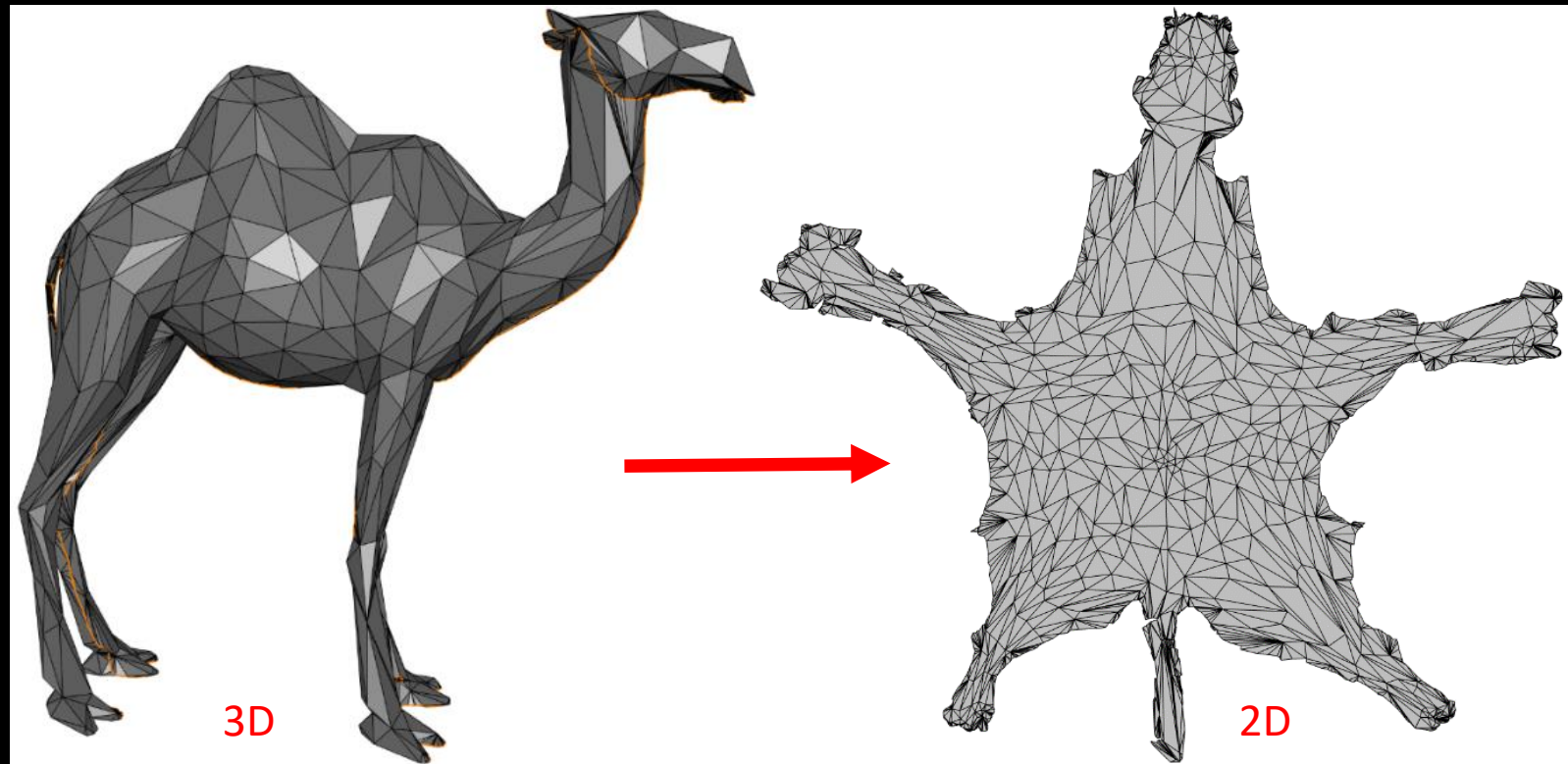
- Definition
- Tutte's barycentric mapping
- Least squares conformal maps(LSCM, ASAP)
- Angle-Based Flattening (ABF)
 - ABF++, LABF
- As-rigid-as-possible (ARAP)
 - Simplex Assembly

Outline

- **Definition**
- Tutte's barycentric mapping
- Least squares conformal maps (LSCM, ASAP)
- Angle-Based Flattening (ABF)
 - ABF++, LABF
- As-rigid-as-possible (ARAP)
 - Simplex Assembly

Definition

- A function that puts input surface in **one-to-one** correspondence with a 2D domain.
- Parameterization of a Triangulated Surface
 - all (u_i, v_i) coordinates associated with each vertex $v_i = (x_i, y_i, z_i)^T$



Goal

- Map attributes
 - Color
 - Normal
 -

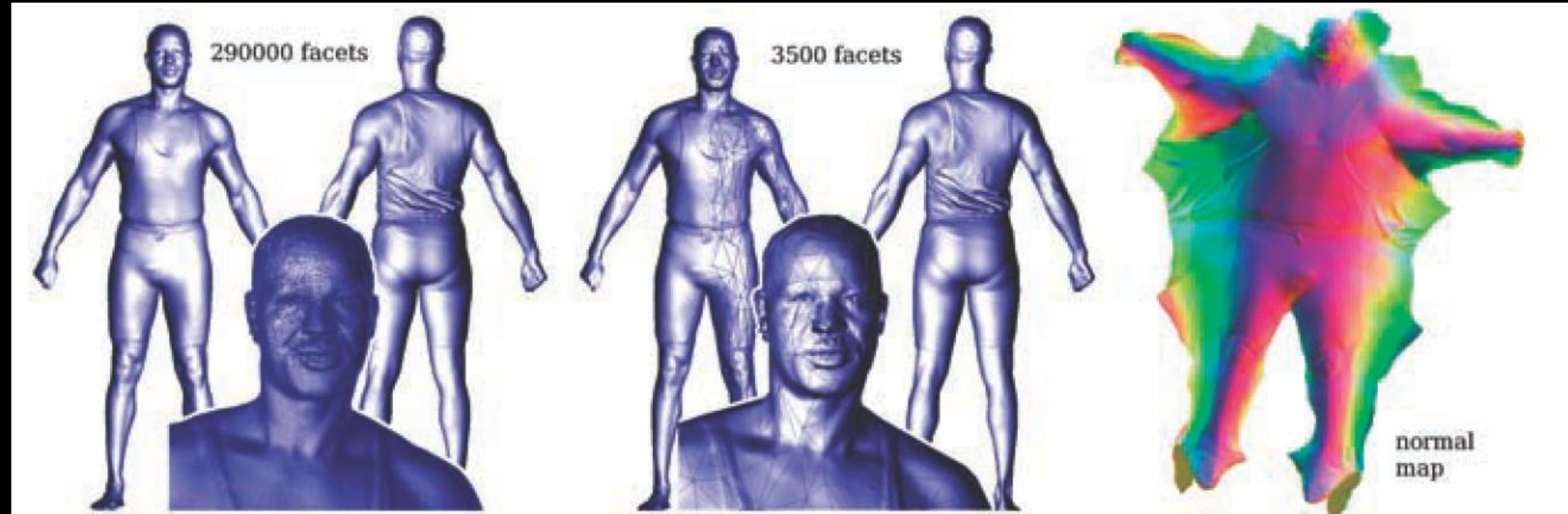
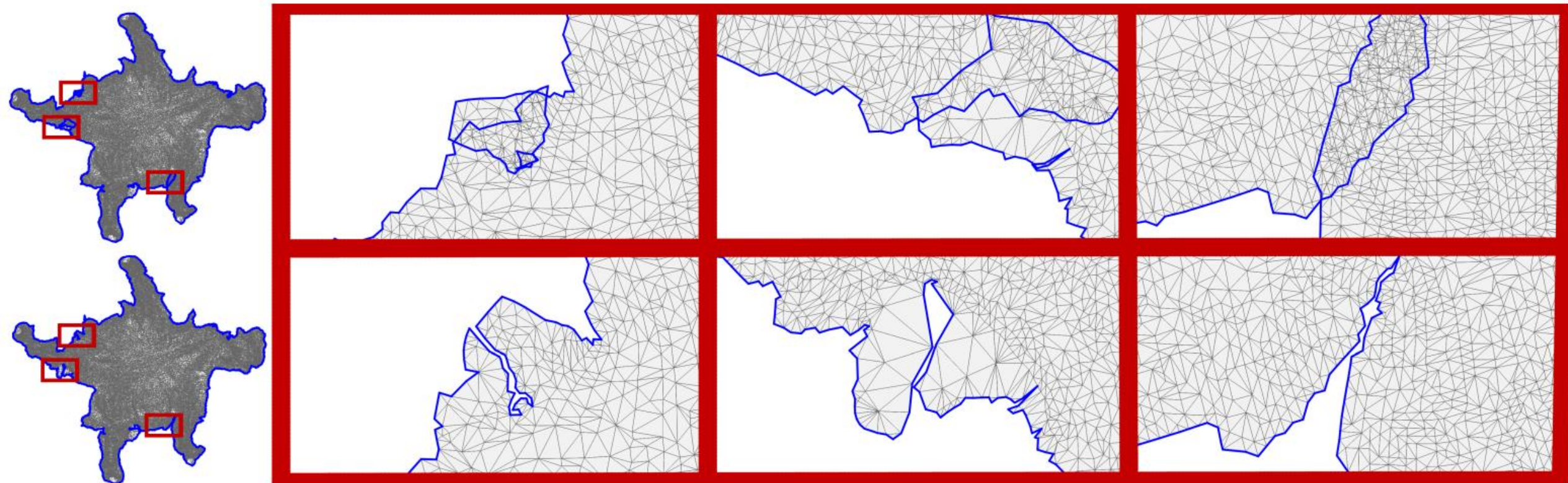


Figure 5.2. Appearance-preserving simplification as another application of parameterization: The initial object (left) is decimated to 1.5% of the original size (center). High-resolution geometric details are encoded in a normal map (right) and mapped to the simplified model, thereby preserving the original appearance. (Model courtesy of Cyberware. Image taken from [Hormann et al. 07]. ©2007 ACM, Inc. Included here by permission.)

Constraints

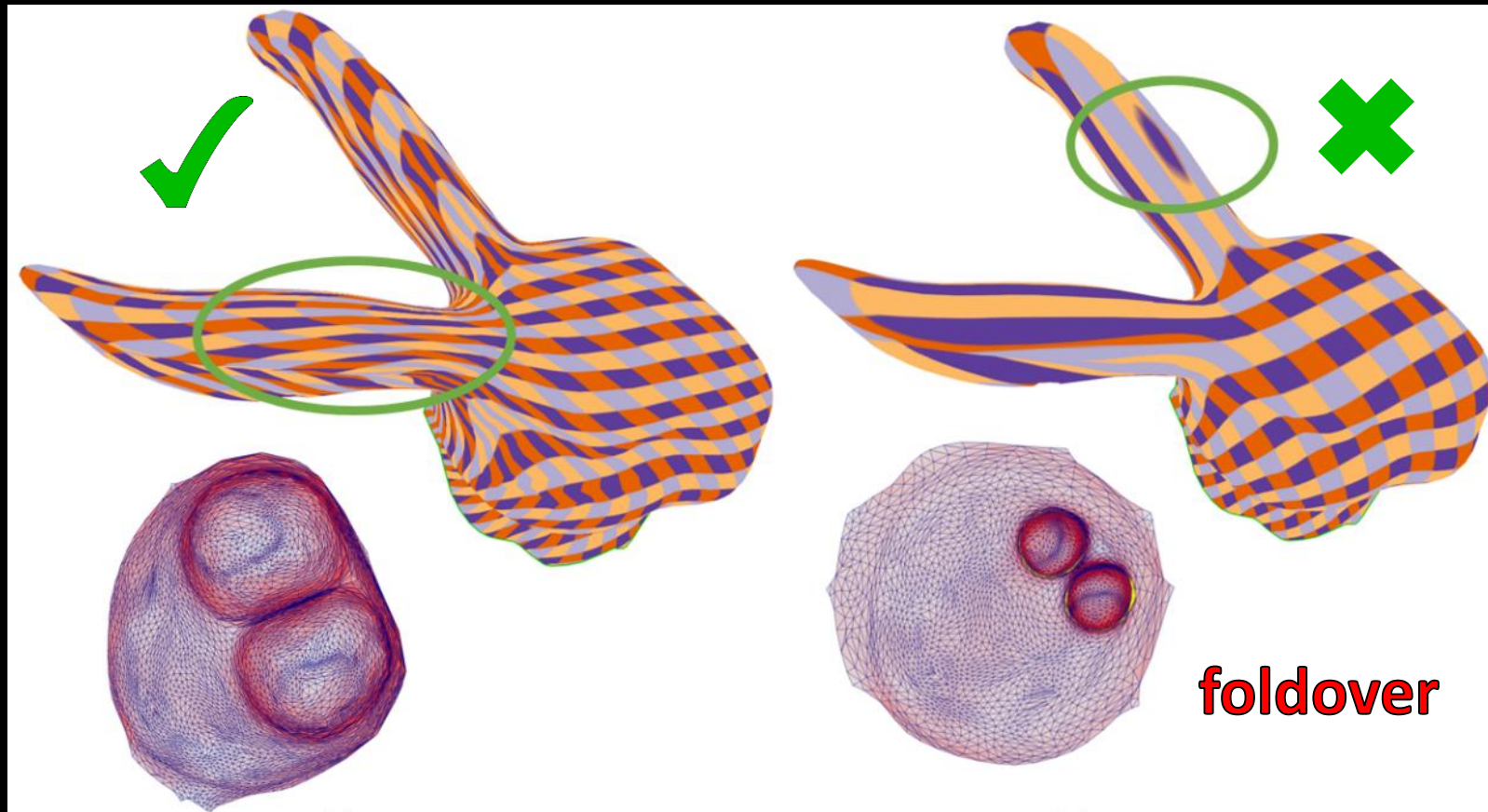
- Bijective

- The image of the surface in parameter space does not self-intersect.
- The intersection of any two triangles in parameter space is either a common edge, a common vertex, or empty.



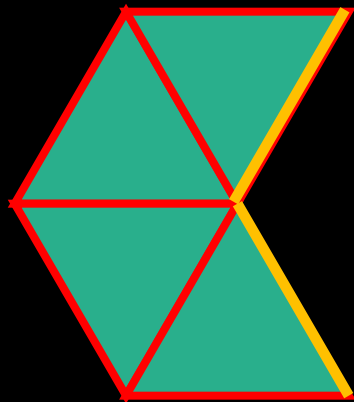
Constraints

- Inversion-free
 - The orientation of each triangle is positive.

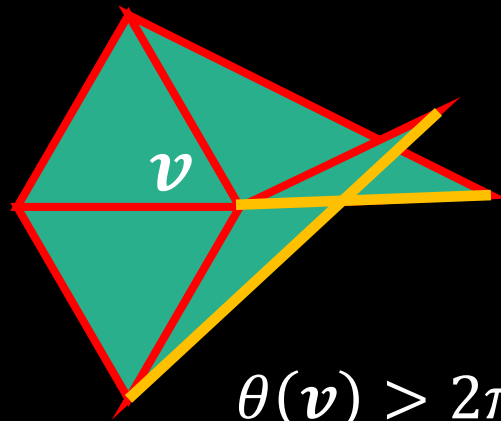


Constraints

- Locally injective
 - The orientation of each triangle is positive $\rightarrow \det J > 0$.
 - For boundary vertex, the mapping is locally bijective $\rightarrow \theta(v) < 2\pi$.



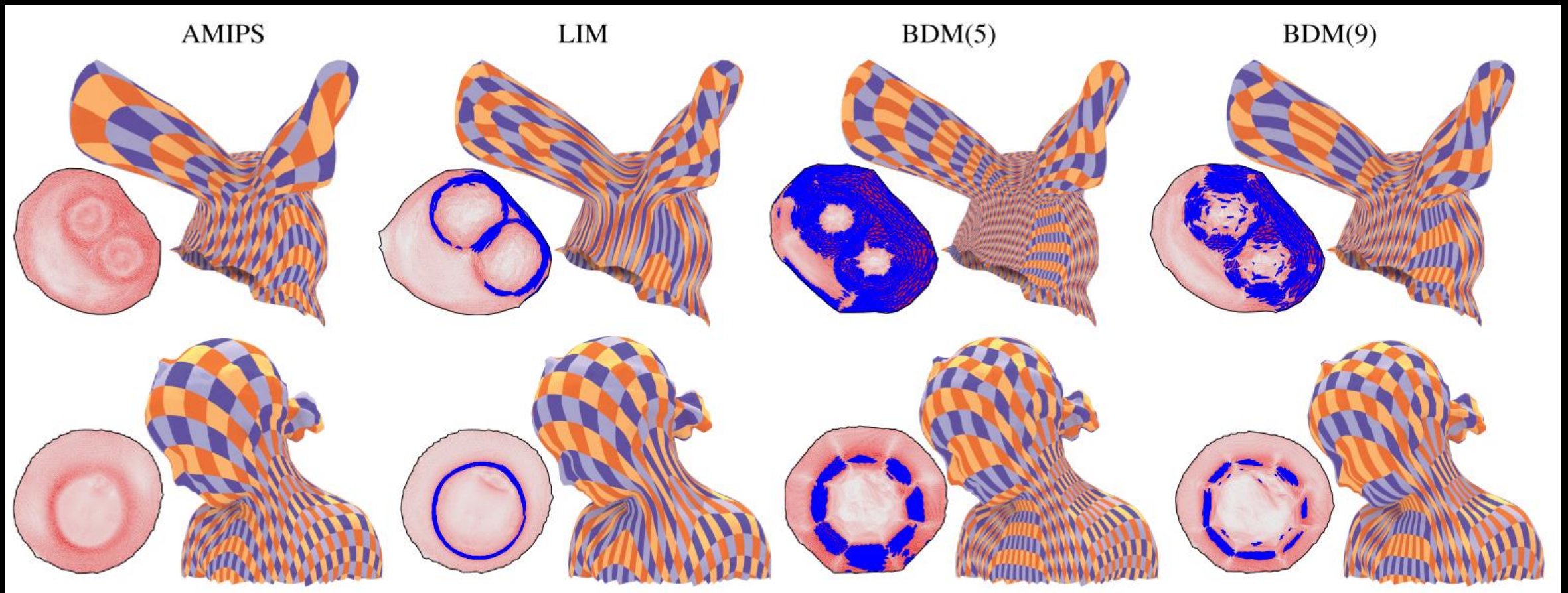
$$\theta(v) < 2\pi$$



$$\theta(v) > 2\pi$$

Constraints

- Low distortion



Outline

- Definition
- **Tutte's barycentric mapping**
- Least squares conformal maps(LSCM, ASAP)
- Angle-Based Flattening (ABF)
 - ABF++, LABF
- As-rigid-as-possible (ARAP)
 - Simplex Assembly

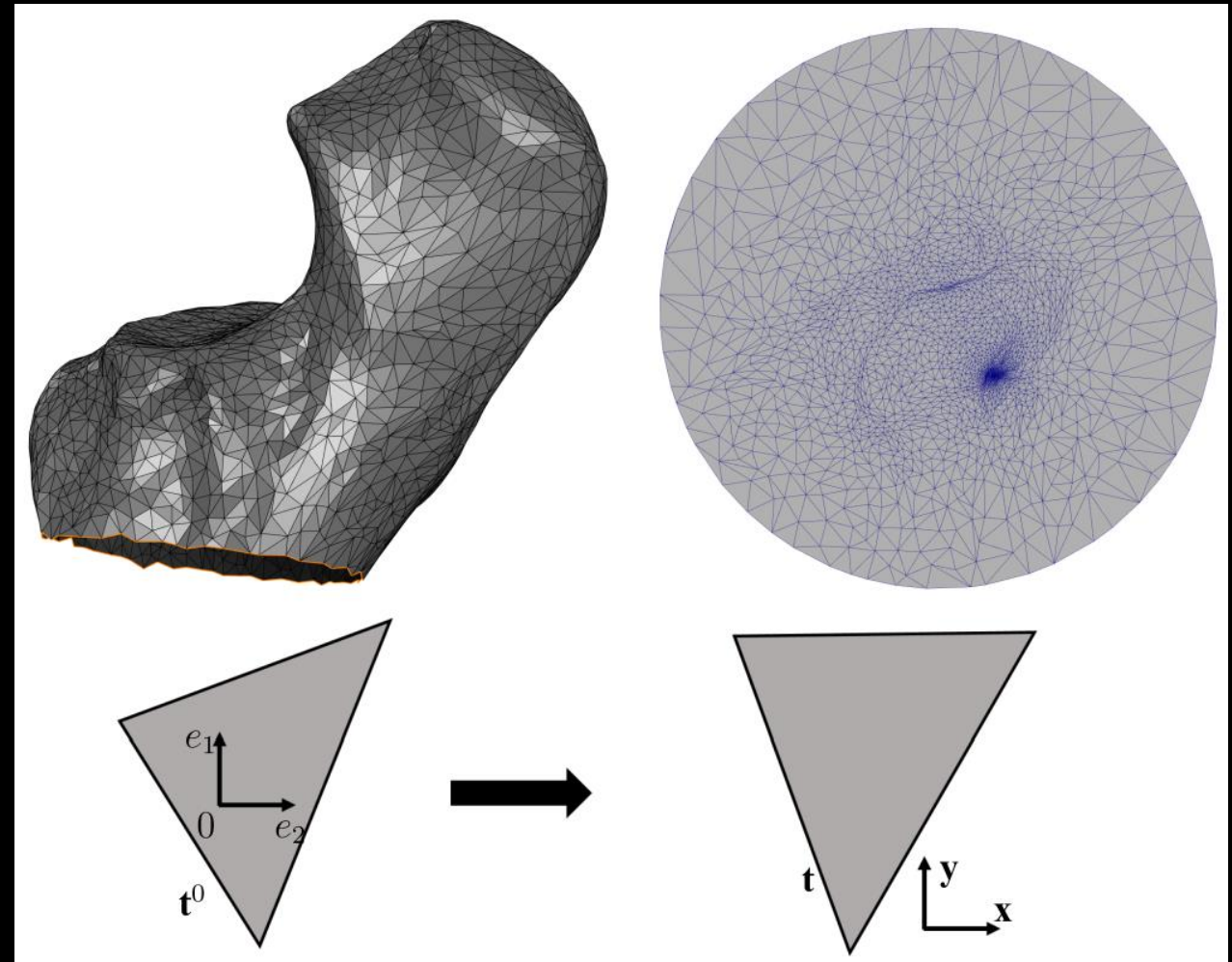
Barycentric Mapping, Tutte's embedding

- One of the most widely used methods.

Given a triangulated surface **homeomorphic to a disk**, if the (u, v) coordinates at the boundary vertices lie on **a convex polygon** in order, and if the coordinates of the internal vertices are **a convex combination** of their neighbors, then the (u, v) coordinates form a valid parameterization (**without self-intersections, bijective**).

Barycentric Mapping

- Homeomorphic to a disk.
- A convex polygon
 - circle, square,.....
- A convex combination
 - $\omega_{ij} > 0$
 - Uniform Laplacian, mean value coordinate
- Solver: linear equation.



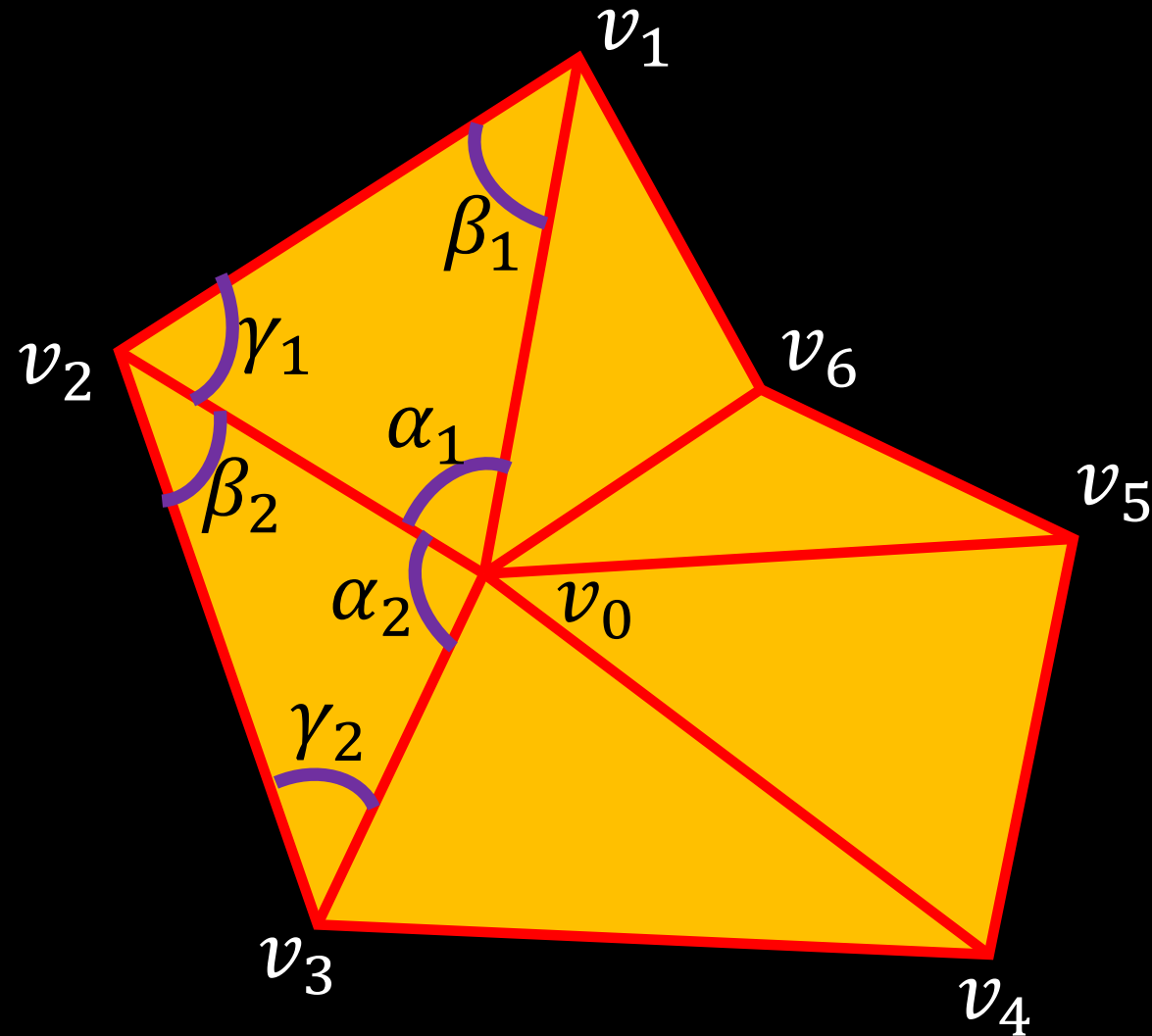
Mean value coordinates

- Our aim is to study sets of weights $\lambda_1, \dots, \lambda_k \geq 0$ such that

$$\sum_{i=1}^k \lambda_i v_i = v_0$$

$$\sum_{i=1}^k \lambda_i = 1$$

v_i is on 2D.



Proposition

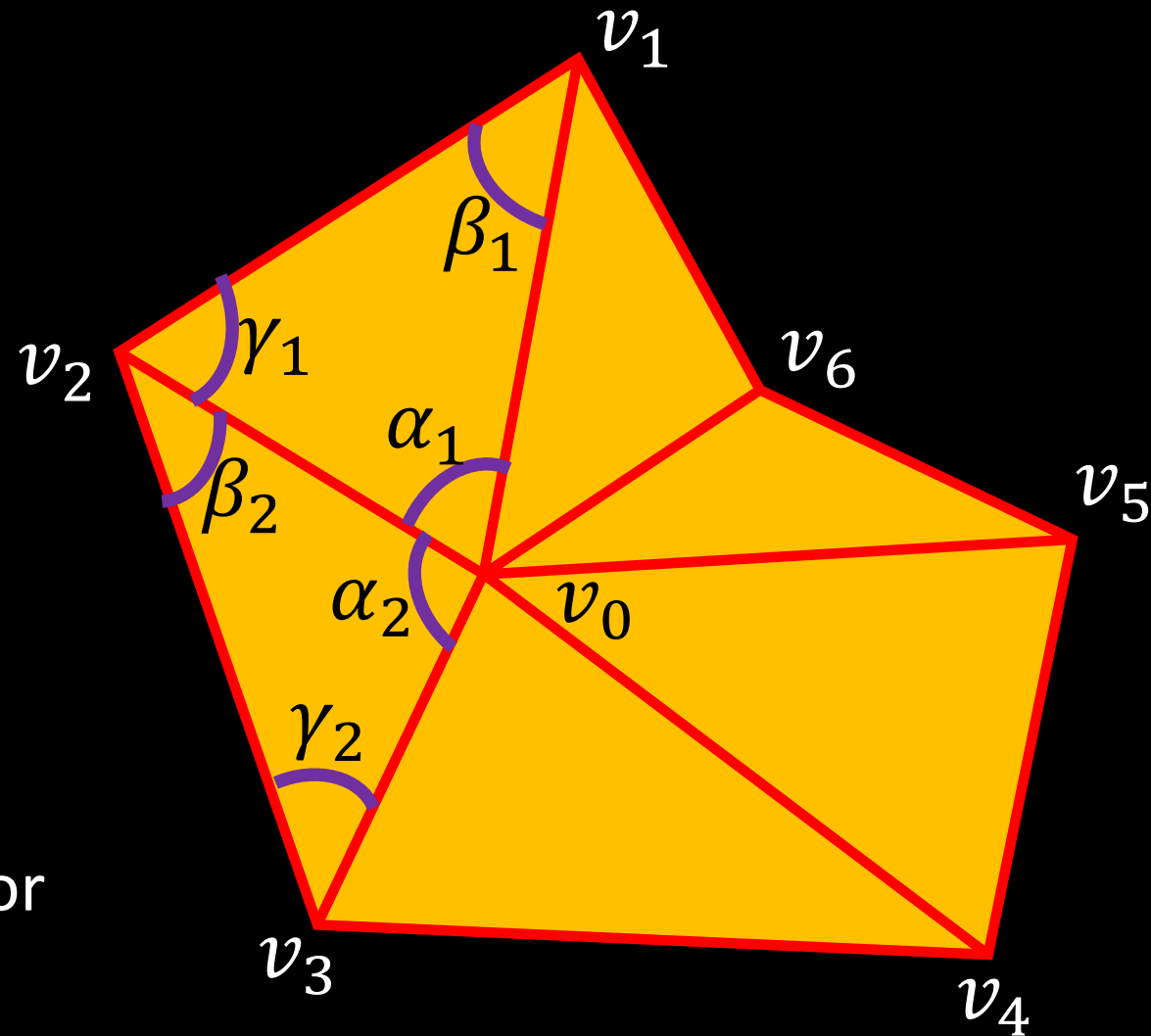
- The weights

$$\lambda_i = \frac{\omega_i}{\sum_{i=1}^k \omega_i},$$
$$\omega_i = \frac{\tan \frac{\alpha_{i-1}}{2} + \tan \frac{\alpha_i}{2}}{\|v_i - v_0\|}$$

are the valid weights.

Proof: substitution. ???

Come from the mean value theorem for harmonic functions. ???



Mean value coordinates

- The input mesh is a spatial one.
 - $v_i \in R^3$
 - the mean value coordinates can be applied directly.
 - compute the coordinates directly from the spatial angle.

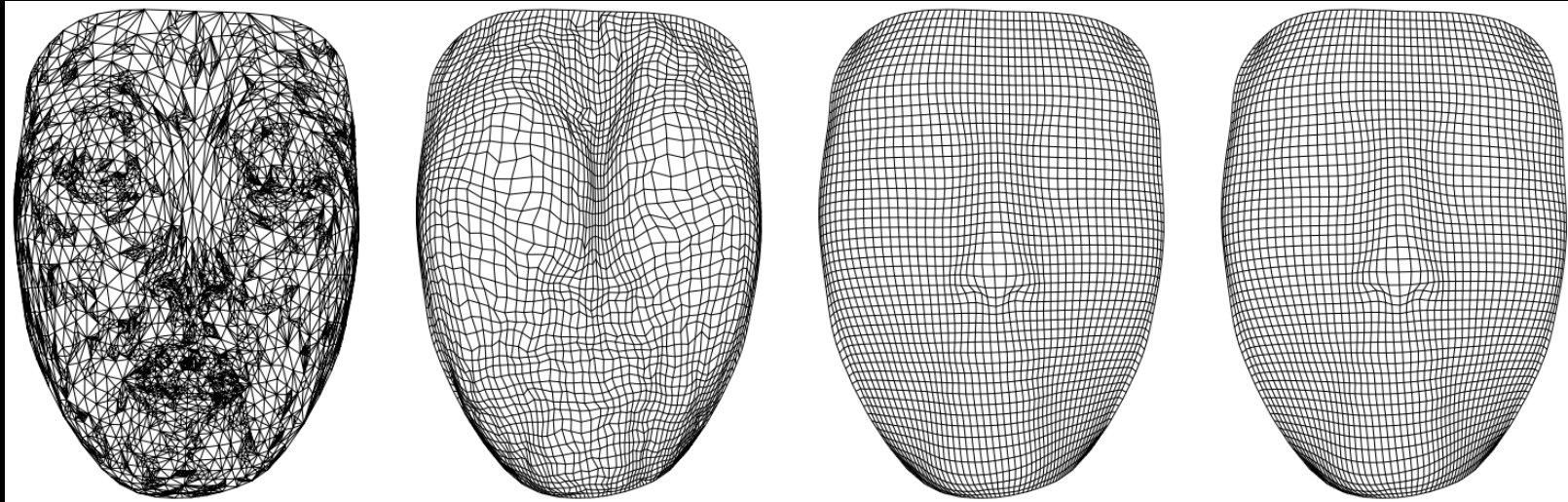


Figure 3. Comparisons from left to right:
(3a) Triangulation, (3b) Tutte, (3c) shape-preserving, (3d) mean value

Outline

- Definition
- Tutte's barycentric mapping
- **Least squares conformal maps (LSCM, ASAP)**
- Angle-Based Flattening (ABF)
 - ABF++, LABF
- As-rigid-as-possible (ARAP)
 - Simplex Assembly

Conformal mapping

- Conformal mappings locally correspond to similarities

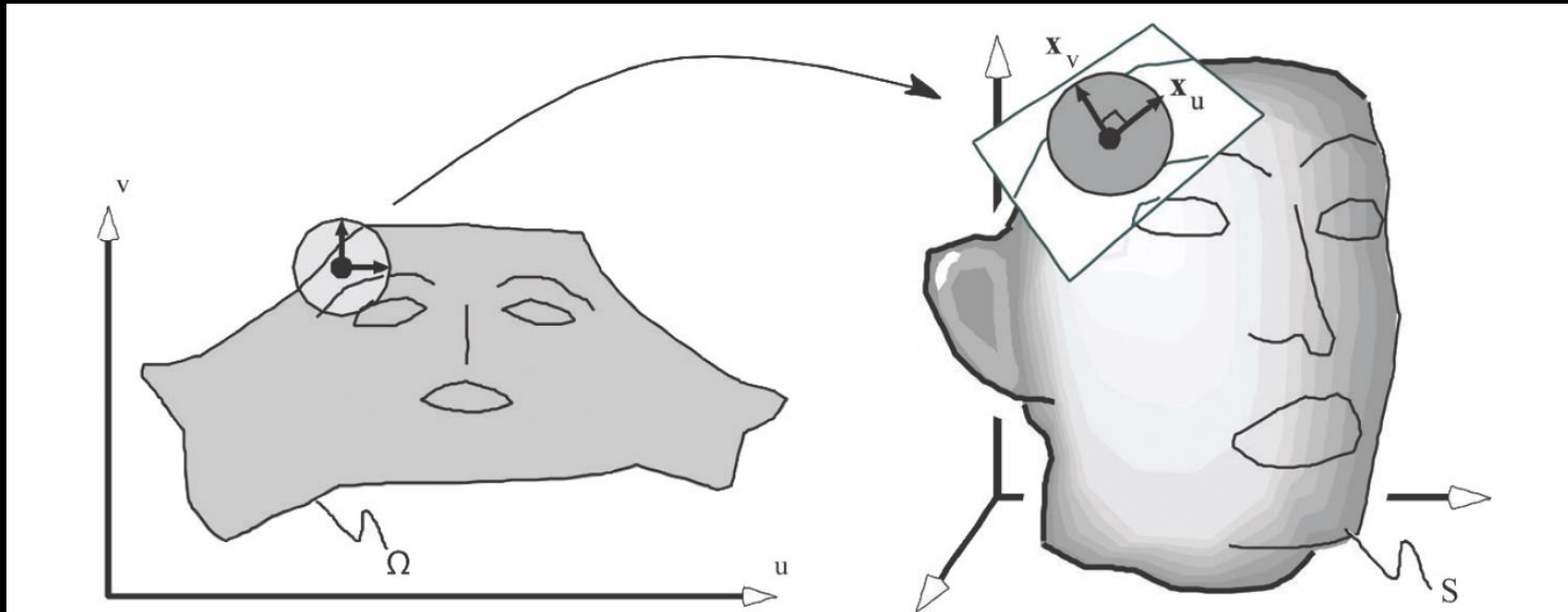
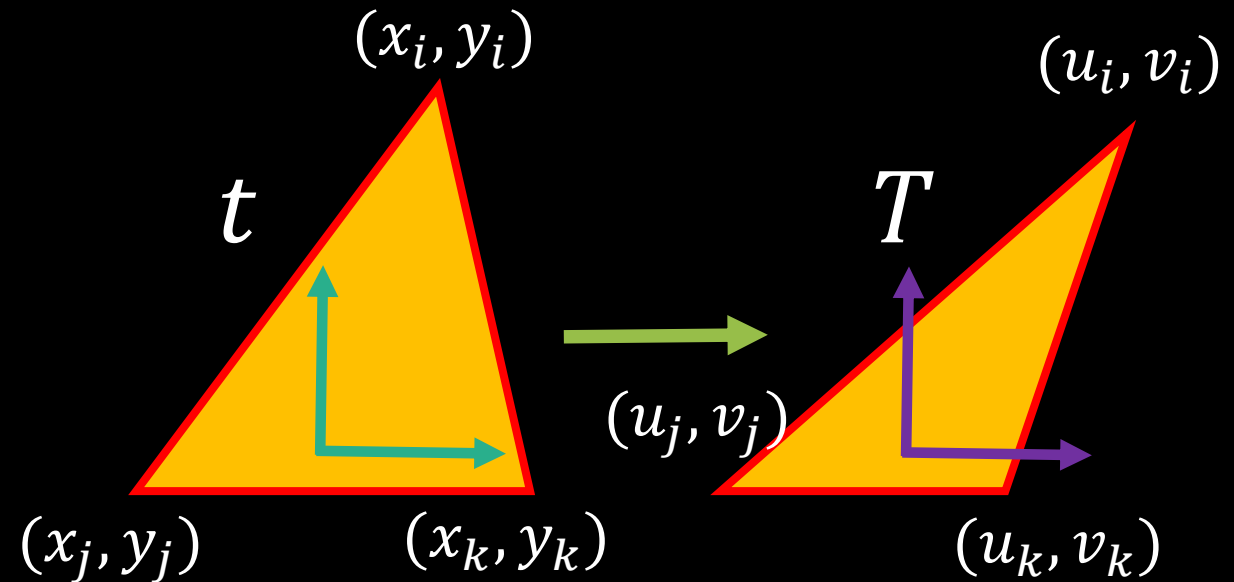


Figure 5.8. A conformal parameterization transforms a small circle into a small circle, i.e., it is locally a similarity transform. (Image taken from [Hormann et al. 07]. ©2007 ACM, Inc. Included here by permission.)

Mapping

- Build a local coordinate system on input triangle t .
- The mapping is piecewise linear.
- J_t is 2×2 .



$$\begin{pmatrix} u_j - u_i & u_k - u_i \\ v_j - v_i & v_k - v_i \end{pmatrix} \begin{pmatrix} x_j - x_i & x_k - x_i \\ y_j - y_i & y_k - y_i \end{pmatrix}^{-1}$$

$$f_t(\mathbf{x}) = J_t \mathbf{x} + \mathbf{b}_t$$

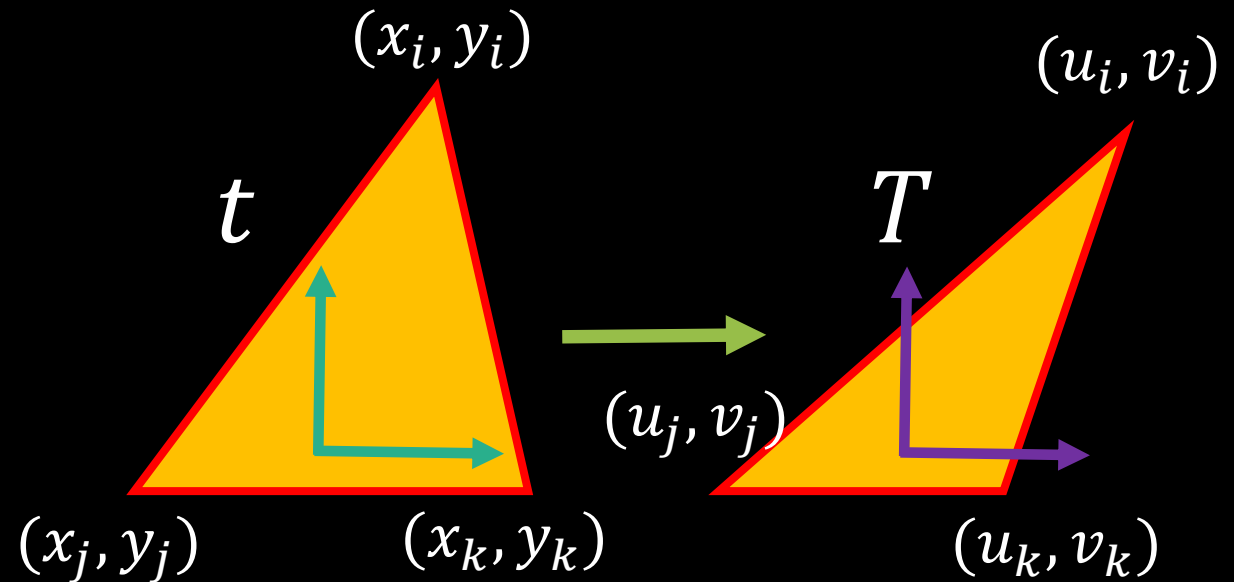
Mapping

- J_t is the Jacobian of $f_t(\mathbf{x})$.

$$J_t = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{pmatrix}$$

$$\begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} \\ \frac{\partial v}{\partial y} \end{pmatrix} = \nabla u$$

$$= \frac{1}{2A_t} \begin{pmatrix} y_j - y_k & y_k - y_i & y_i - y_j \\ x_k - x_j & x_i - x_k & x_j - x_i \end{pmatrix} \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix}$$



$$f_t(\mathbf{x}) = J_t \mathbf{x} + \mathbf{b}_t$$

Similar transform

- 2D case: for one triangle t

- $J_t = \begin{pmatrix} a & -b \\ b & a \end{pmatrix} = s \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$

- $\Rightarrow \begin{cases} \frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \\ \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x} \end{cases}$

- Cauchy-Riemann Equations.

$$J_t = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{pmatrix}$$

Least squares conformal maps(LSCM, ASAP)

- Energy

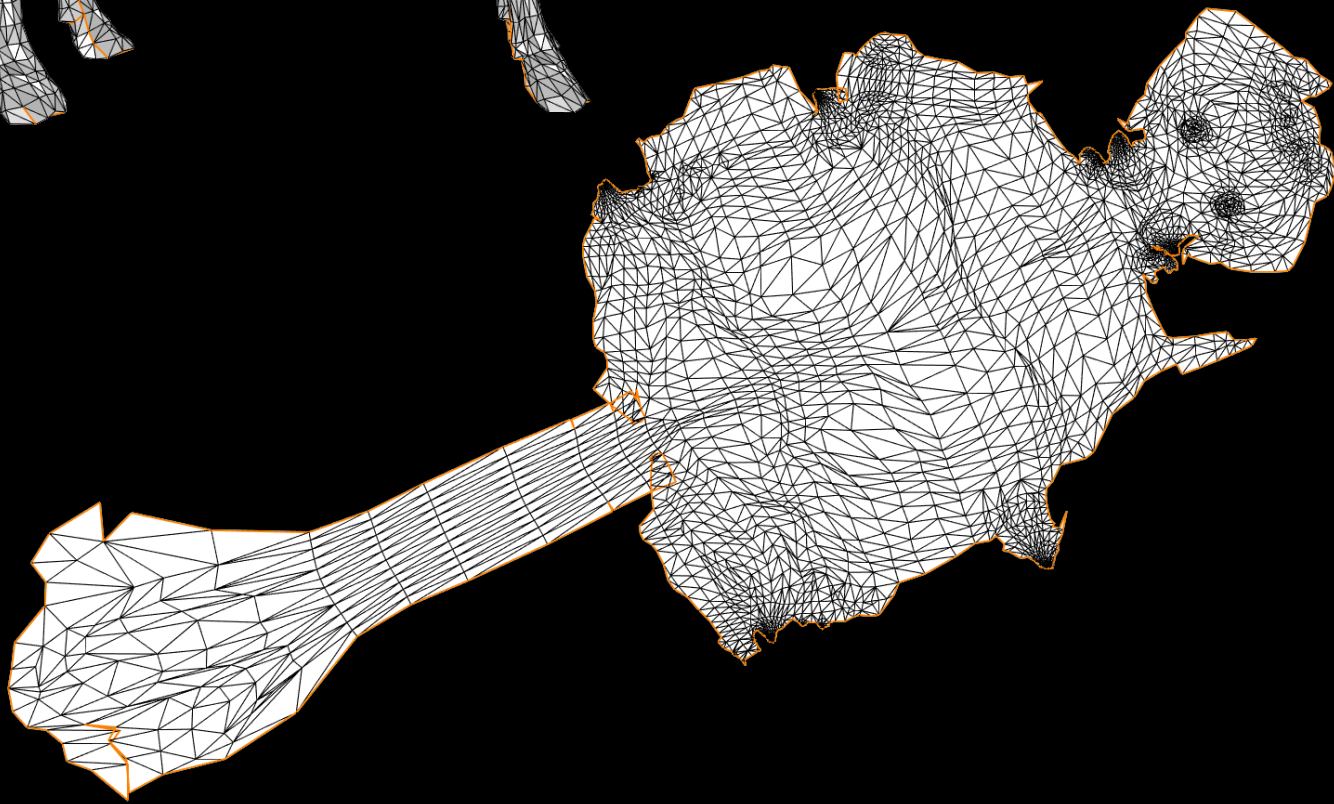
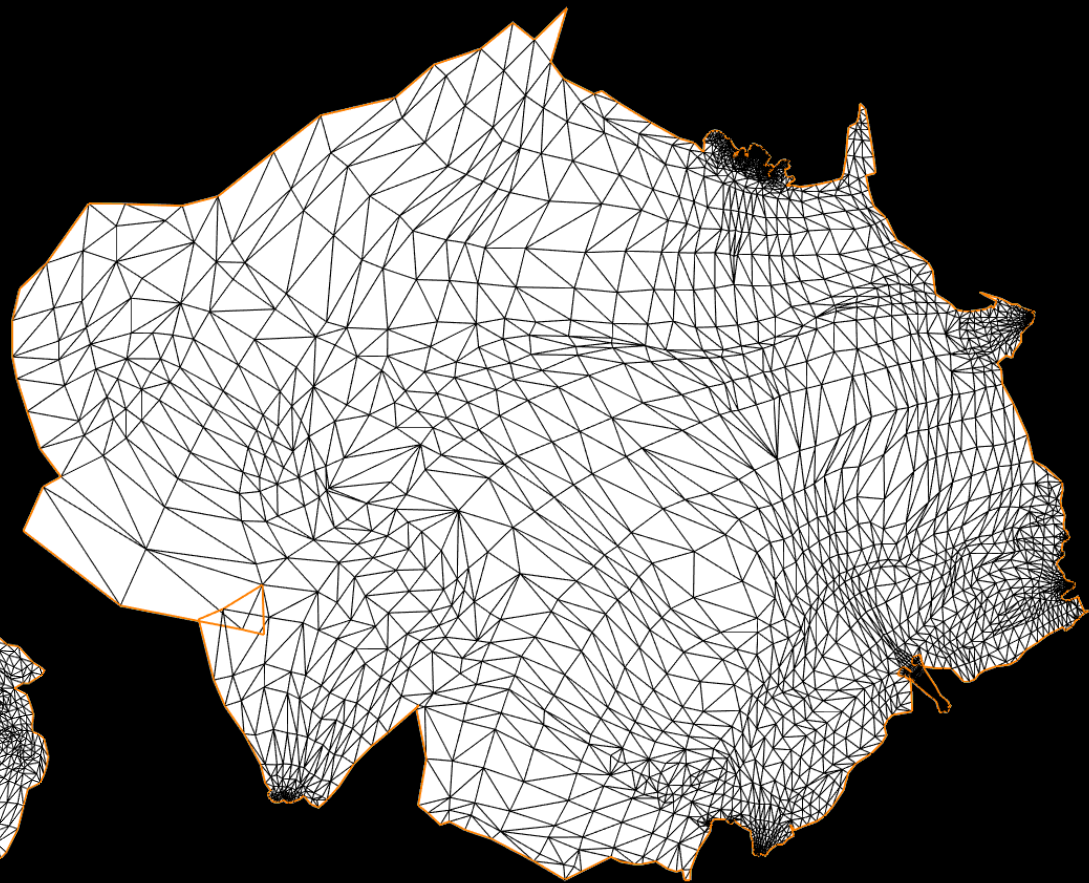
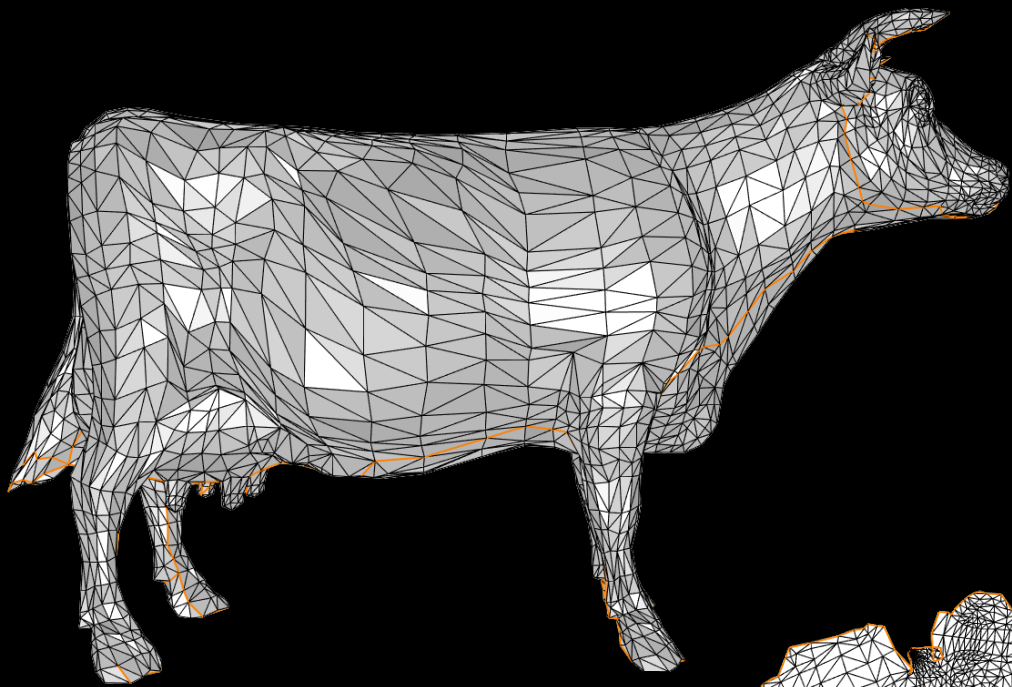
- $E_{LSCM} = \sum_t A_t \left(\left(\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} \right)^2 + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right)^2 \right)$

- measure non-conformality

- It is invariant with respect to arbitrary translations and rotations.

- E_{LSCM} does not have a unique minimizer.

- Fixing at least two vertices. Significantly affect the results.



Outline

- Definition
- Tutte's barycentric mapping
- Least squares conformal maps(LSCM, ASAP)
- **Angle-Based Flattening (ABF)**
 - ABF++, LABF
- As-rigid-as-possible (ARAP)
 - Simplex Assembly

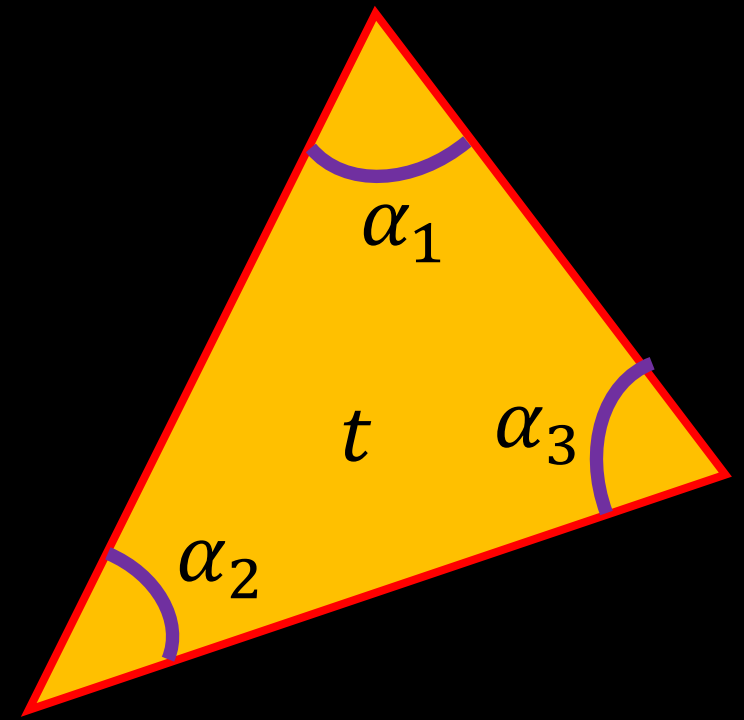
Angle-Based Flattening (ABF)

- Key observation: the parameter space is a 2D triangulation, uniquely defined by all the angles at the corners of the triangles.
 - Find angles instead of (u_i, v_i) coordinates.
 - Use angles to reconstruct the resulting parameterization.
- Optimization goal:

$$E_{ABF} = \sum_t \sum_{i=1}^3 \omega_i^t (\alpha_i^t - \beta_i^t)^2$$

β_i^t : Optimal angles for α_i^t .

$$\omega_i^t = (\beta_i^t)^{-2}.$$



$$\beta_i^t = \begin{cases} \frac{\tilde{\beta}_i^t \cdot 2\pi}{\sum_i \tilde{\beta}_i^t}, & \text{Interior vertex} \\ \tilde{\beta}_i^t, & \text{Boundary vertex} \end{cases}$$

Constraints

- Positive resulting angles:

$$\alpha_i^t > 0$$

- The three triangle angles have to sum to π :

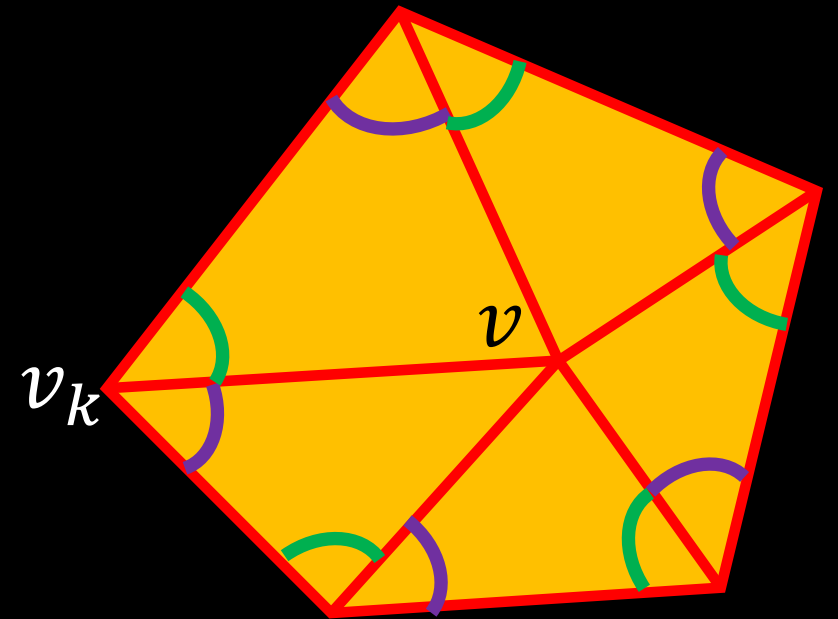
$$\alpha_1^t + \alpha_2^t + \alpha_3^t = \pi$$

- For each internal vertex the incident angles have to sum to 2π :

$$\sum_{t \in \Omega(v)} \alpha_k^t = 2\pi$$

- Reconstruction constraints:

$$\prod_{t \in \Omega(v)} \sin \alpha_{k \oplus 1}^t = \prod_{t \in \Omega(v)} \sin \alpha_{k \ominus 1}^t \quad ???$$



Linear ABF

- Reconstruction constraints are nonlinear and hard to solve.
- Initial estimation + estimation error

- $\alpha_i^t = \gamma_i^t + e_i^t$

$$\log \left(\prod_{t \in \Omega(v)} \sin \alpha_{k \oplus 1}^t \right) = \log \left(\prod_{t \in \Omega(v)} \sin \alpha_{k \ominus 1}^t \right)$$

$$\sum_{t \in \Omega(v)} \log(\sin \alpha_{k \oplus 1}^t) = \sum_{t \in \Omega(v)} \log(\sin \alpha_{k \ominus 1}^t)$$

- Taylor expansion:

$$\begin{aligned} \log(\sin \alpha_{k \oplus 1}^t) &= \log(\sin \gamma_{k \oplus 1}^t + e_{k \oplus 1}^t) \\ &= \log(\sin \gamma_{k \oplus 1}^t) + e_{k \oplus 1}^t \cot \gamma_{k \oplus 1}^t + \dots \end{aligned}$$

It is linear with estimation error.

Solver

- Set $\gamma_i^t = \beta_i^t$
- Problem:

$$\min_e E_{ABF} = \sum_t \sum_{i=1}^3 \omega_i^t (e_i^t)^2$$

subject to $Ae = b$

\Rightarrow

$$\begin{pmatrix} D & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} e \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ b \end{pmatrix}$$

\Rightarrow

$$e = D^{-1}A^T(AD^{-1}A^T)^{-1}b ???$$

Reconstruct parameterization

- Greed method.
 - constructs the triangles one by one using a depth-first traversal.
- Least squares method.
 - an angle based least squares formulation which solves a set of linear equations relating angles to coordinates.

Greed method

- Choose a mesh edge $e^1 = (v_a^1, v_b^1)$.
- Project v_a^1 to $(0,0,0)$ and v_b^1 to $(\|e^1\|, 0,0)$.
- Push e^1 on the stack S .
- While S not empty, pop an edge $e = (v_a, v_b)$. For each face $f_i = (v_a, v_b, v_c)$ containing e :
 - If f_i is marked as **set**, continue.
 - If v_c is not projected, compute its position based on v_a, v_b and the face angles of f_i .
 - Mark f_i as **set**, push edge (v_b, v_c) and (v_a, v_c) on the stack.
- Accumulate numerical error.

Least squares method

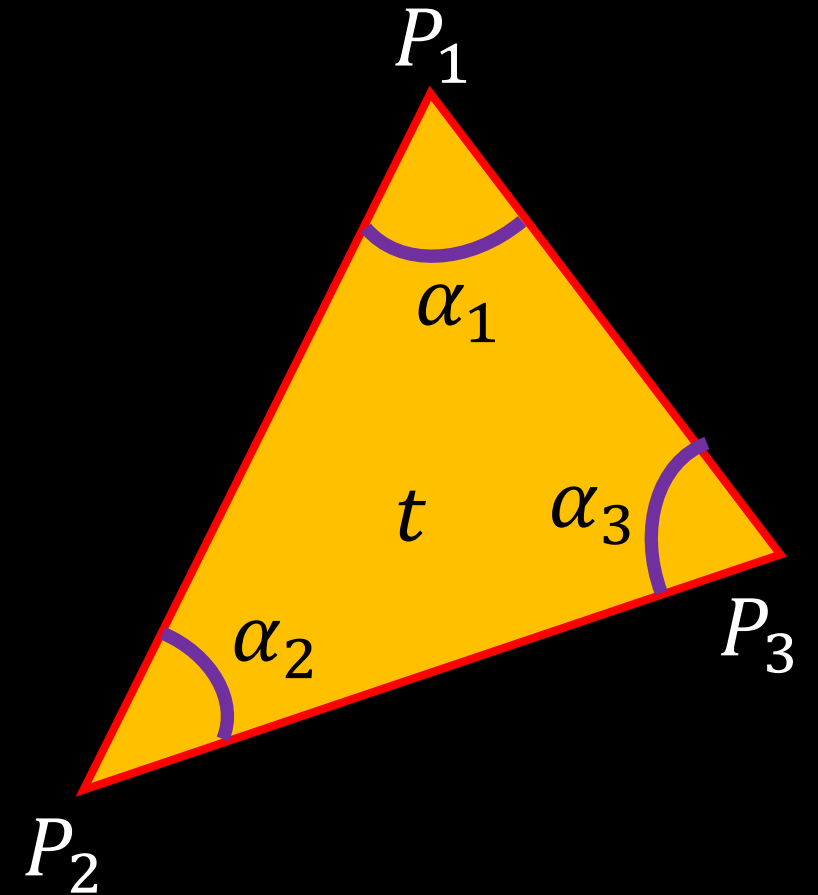
- The ratio of triangle edge lengths $\|\overrightarrow{P_1P_3}\|$ and $\|\overrightarrow{P_1P_2}\|$ is

$$\frac{\|\overrightarrow{P_1P_3}\|}{\|\overrightarrow{P_1P_2}\|} = \frac{\sin \alpha_2}{\sin \alpha_3}$$

\Rightarrow

$$\overrightarrow{P_1P_3} = \frac{\sin \alpha_2}{\sin \alpha_3} \begin{pmatrix} \cos \alpha_1 & -\sin \alpha_1 \\ \sin \alpha_1 & \cos \alpha_1 \end{pmatrix} \overrightarrow{P_1P_2}$$

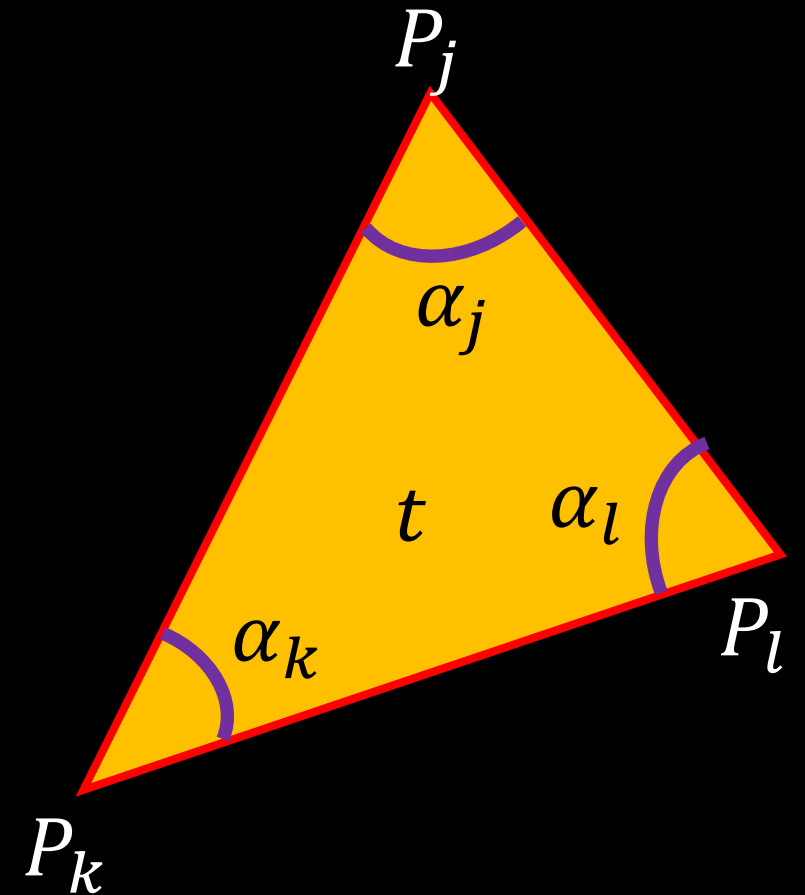
- Thus for each triangle, given the position of two vertices and the angles, the position of the third vertex can be uniquely derived.
 - greedy method.



Least squares method

$$\forall t = (j, k, l), \quad M^t (P_k - P_j) + P_j - P_l = 0$$
$$M^t = \frac{\sin \alpha_k}{\sin \alpha_l} \begin{pmatrix} \cos \alpha_j & -\sin \alpha_j \\ \sin \alpha_j & \cos \alpha_j \end{pmatrix}$$

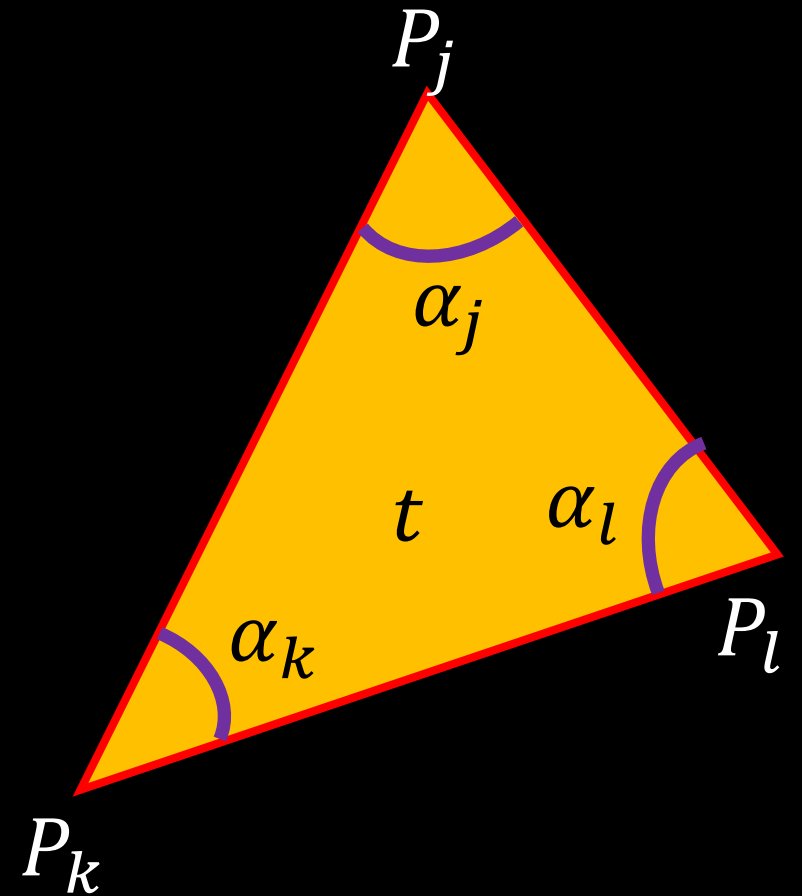
1. Two equations per triangle for the x and y coordinates of the vertices.
2. The angles of a planar triangulation define it uniquely up to **rigid transformation** and **global scaling**.
 - Introduce four constraints which eliminate these degrees of freedom.
 - Fix two vertices sharing a common edge.

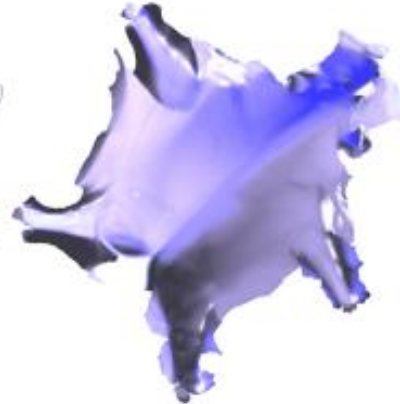


Least squares method

- Choose one edge $e^1 = (v_a^1, v_b^1)$.
- Project v_a^1 to $(0,0,0)$ and v_b^1 to $(\|e^1\|, 0,0)$.
- Solve following energy to compute positions of other vertices:

$$E = \sum_t \left\| M^t (P_k - P_j) + P_j - P_l \right\|^2$$





Outline

- Definition
- Tutte's barycentric mapping
- Least squares conformal maps(LSCM, ASAP)
- Angle-Based Flattening (ABF)
 - ABF++, LABF
- **As-rigid-as-possible (ARAP)**
 - Simplex Assembly

As-rigid-as-possible method

Paper: A Local/Global Approach to Mesh Parameterization

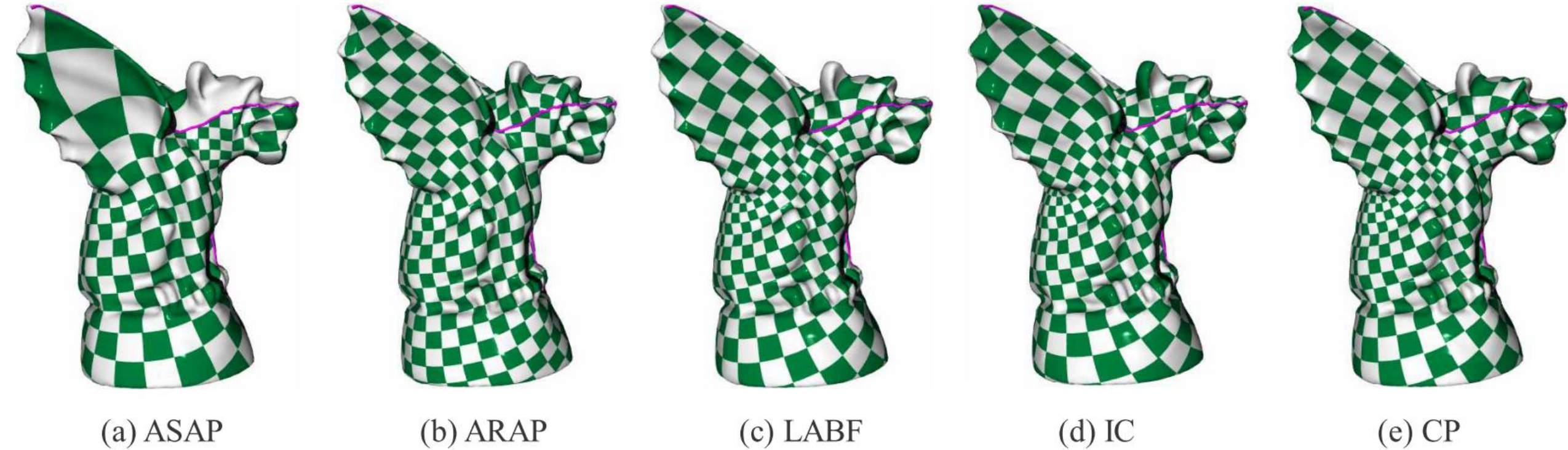
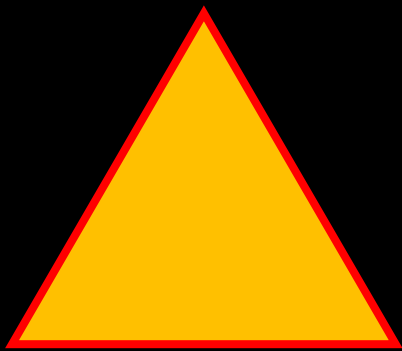


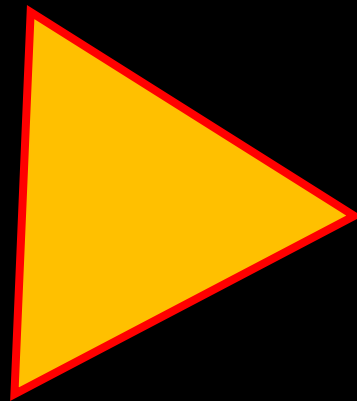
Figure 1: Parameterization of the Gargoyle model using (a) our As-Similar-As-Possible (ASAP) procedure, (b) As-Rigid-As-Possible (ARAP) procedure, (c) Linear ABF [ZLS07], (d) inverse curvature approach [YKL*08], and (e) curvature prescription approach [BCGB08]. The pink lines are the seams of the closed mesh when cut to a disk.

Distortion type

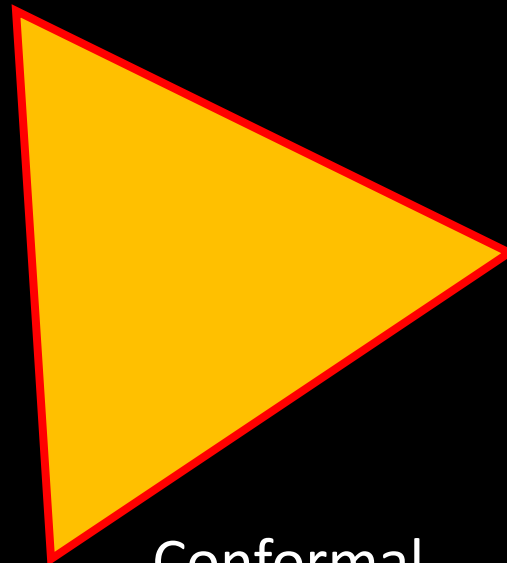
- Three common distortion types:
 - Isometric mapping: rotation + translation
 - Conformal mapping: similarity + translation
 - Area-preserving mapping: area-preserving + translation
 - Conformal + Area-preserving \Leftrightarrow Isometric



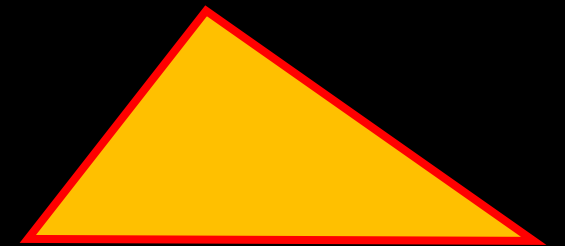
source



Isometric



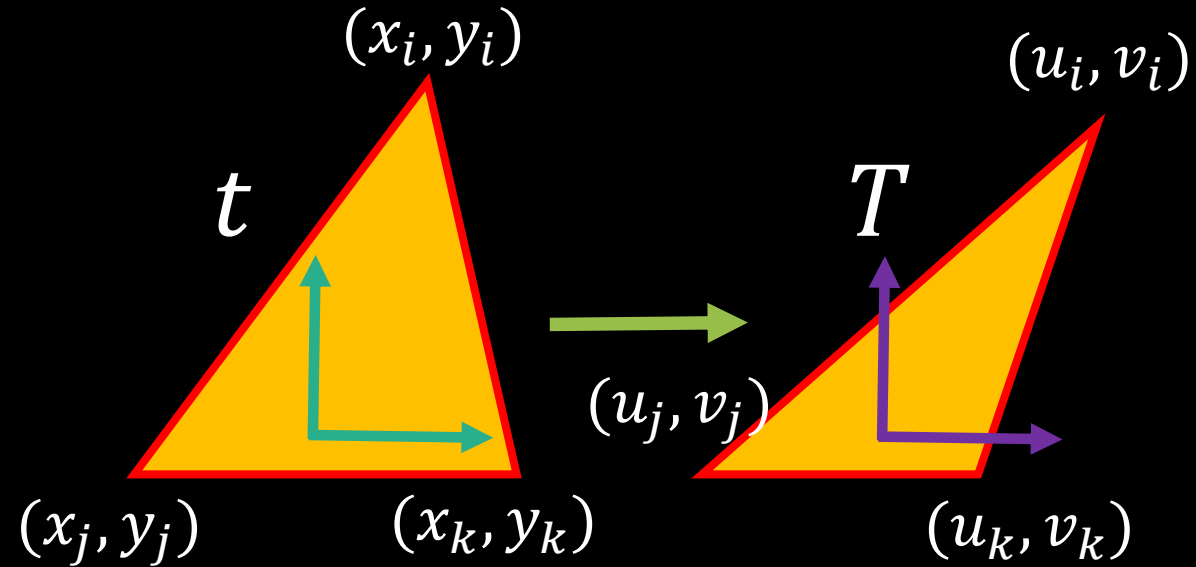
Conformal



Area-preserving

Singular values

- Isometric mapping
 - $J_t \Rightarrow$ rotation matrix
 - $\sigma_1 = \sigma_2 = 1$
- Conformal mapping
 - $J_t \Rightarrow$ similar matrix
 - $\sigma_1 = \sigma_2$
- Area-preserving mapping
 - $\det J_t = 1$
 - $\sigma_1 \sigma_2 = 1$



$$f_t(\mathbf{x}) = J_t \mathbf{x} + \mathbf{b}_t$$

$$J_t = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{pmatrix}$$

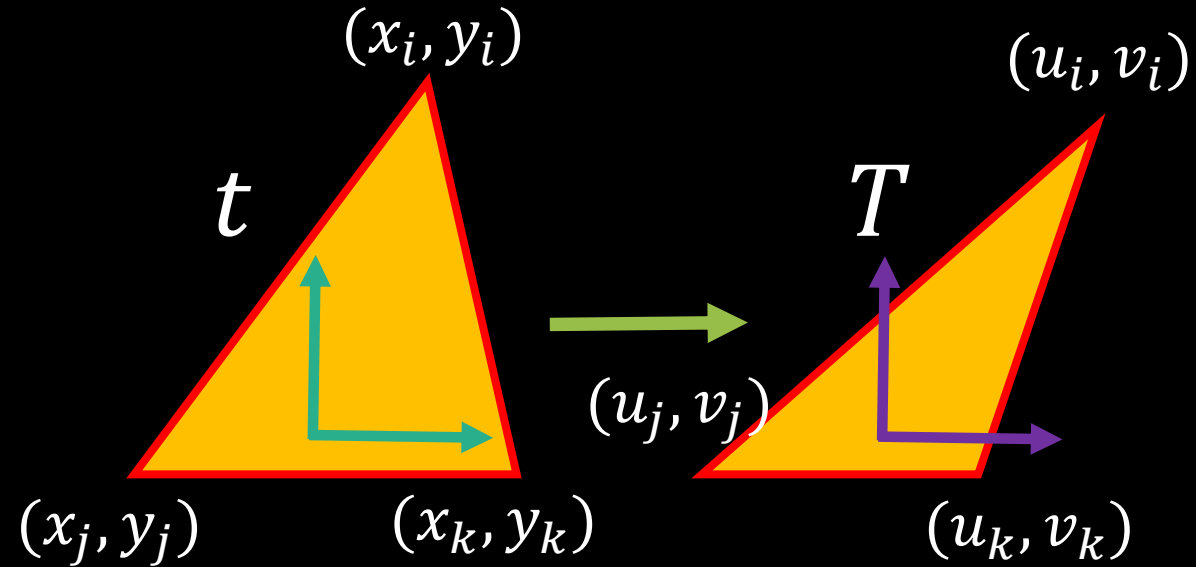
σ_1, σ_2 are the two singular values of J_t .

Goal

$$E(u, L) = \sum_t A_t \|J_t - L_t\|_F^2$$

L_t : target transformation

- Isometric mapping: rotation matrix
- Conformal mapping: similar matrix
- Variables:
 - 2D parameterization coordinate
 - Target transformation
- **How to optimize?**



$$f_t(\mathbf{x}) = J_t \mathbf{x} + \mathbf{b}_t$$

$$J_t = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{pmatrix}$$

σ_1, σ_2 are the two singular values of J_t .

General Local/Global Approach

- Alternatively optimization

- Local step:

- Fix 2D parameterization coordinates, optimize target transformations.

- Global step:

- Fix target transformations, optimize 2D parameterization coordinates.

- **Global step:**

- Quadratic energy

- Linear system

- Eigen

$$E(u, L) = \sum_t A_t \|J_t - L_t\|_F^2$$

Local step: Procrustes analysis

- Approximate one 2×2 matrix J_t as best we can by another 2×2 matrix L_t .
- $d(J_t, L_t) = \|J_t - L_t\|_F^2 = \text{trace}((J_t - L_t)^T (J_t - L_t))$
- Minimize $d(J_t, L_t)$ through Singular Value Decomposition (SVD)
 - $J_t = U\Sigma V^T$, $\Sigma = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}$
 - Signed SVD: U and V are rotation matrix, σ_2 maybe negative
 - Best rotation: UV^T
 - Best similar matrix: $U \begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix} V^T$, $s = \frac{\sigma_1 + \sigma_2}{2}$

Local/Global Approach summary

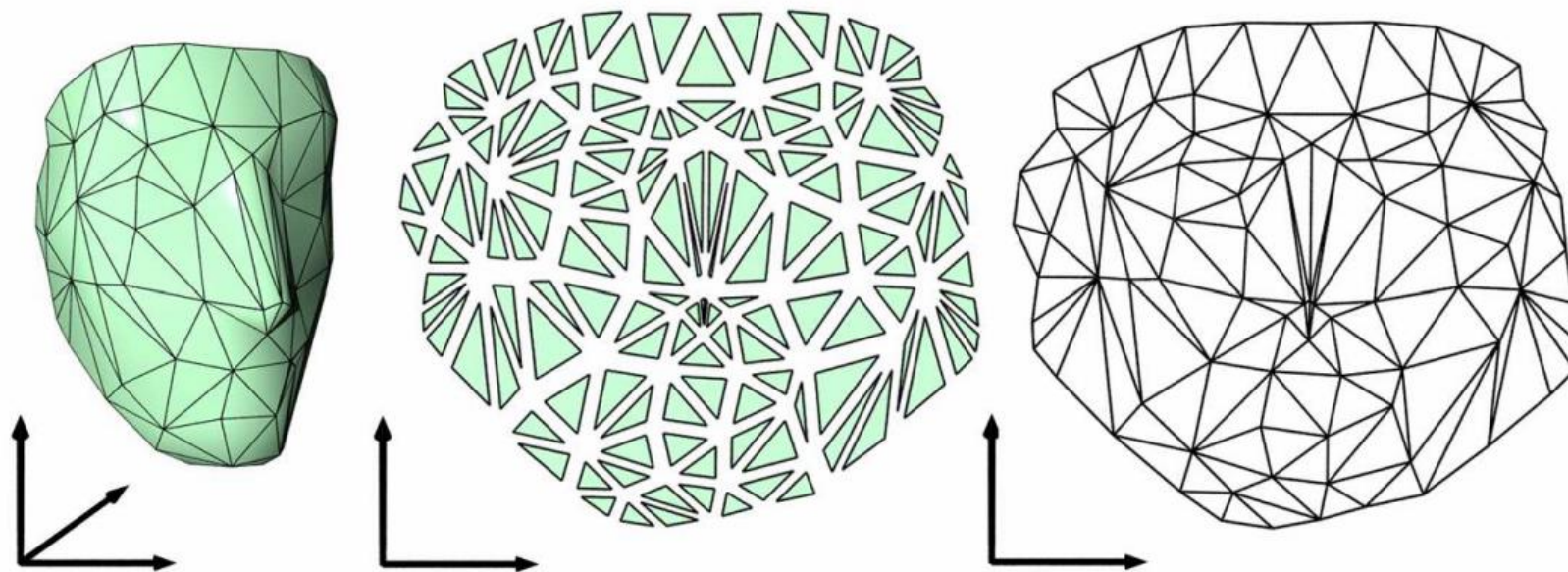


Figure 2: *Parameterizing a mesh by aligning locally flattened triangles. (Left) Original 3D mesh; (middle) flattened triangles; (right) 2D parameterization.*

Connection to singular values

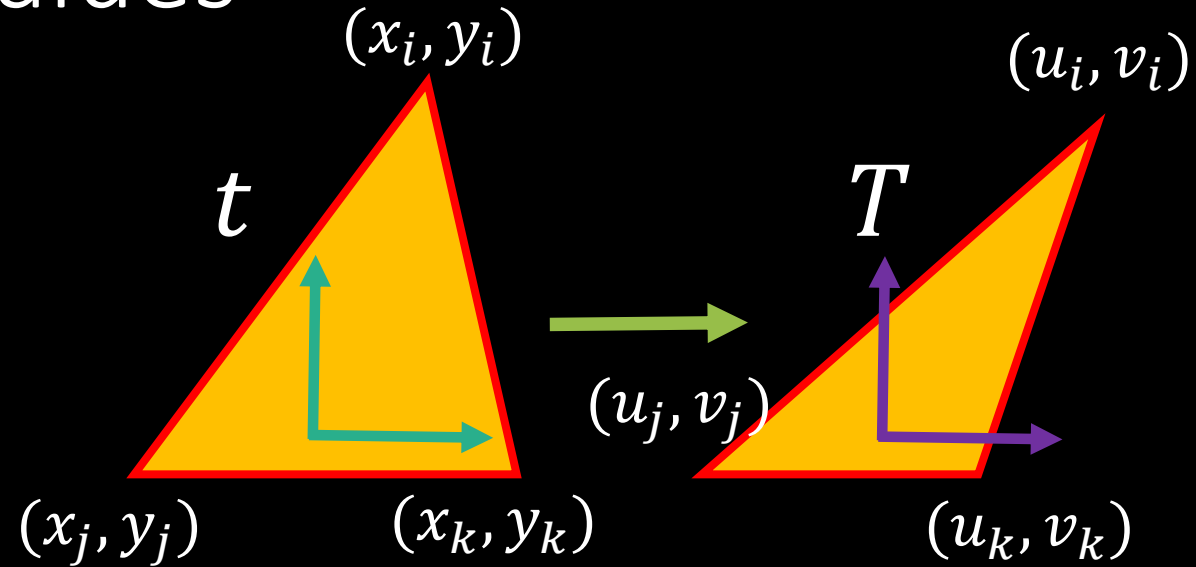
- Conformal

$$E(u) = \sum_t A_t (\sigma_t^1 - \sigma_t^2)^2$$

- Isometric

$$E(u) = \sum_t A_t \left((\sigma_t^1 - 1)^2 + (\sigma_t^2 - 1)^2 \right)$$

???????



$$f_t(\mathbf{x}) = J_t \mathbf{x} + \mathbf{b}_t$$

σ_t^1, σ_t^2 are the two singular values of J_t .

$$E(u, L) = \sum_t A_t \|J_t - L_t\|_F^2$$

Outline

- Definition
- Tutte's barycentric mapping
- Least squares conformal maps(LSCM, ASAP)
- Angle-Based Flattening (ABF)
 - ABF++, LABF
- As-rigid-as-possible (ARAP)
 - **Simplex Assembly**

Information

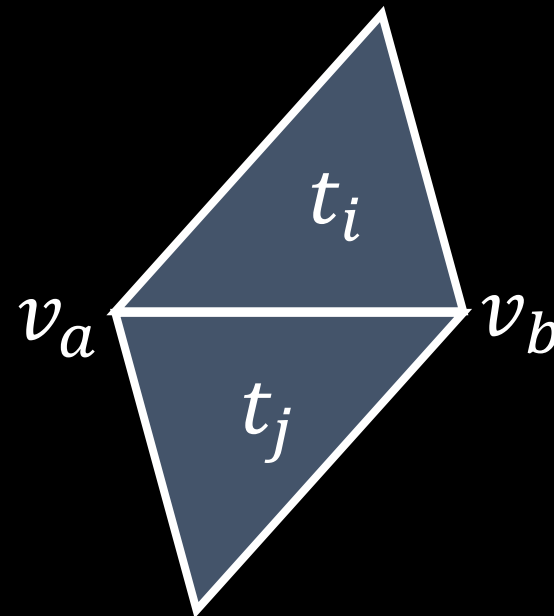
- **Computing Inversion-Free Mappings by Simplex Assembly**
- ***ACM Transactions on Graphics(SIGGRAPH Asia) 35(6), 2016.***
- <http://staff.ustc.edu.cn/~fuxm/projects/SimplexAssembly/index.html>

Affine transformation

Key observation: the parameter space is a 2D triangulation, uniquely defined by all the **AFFINE TRANSFORMATIONS** on the triangles.

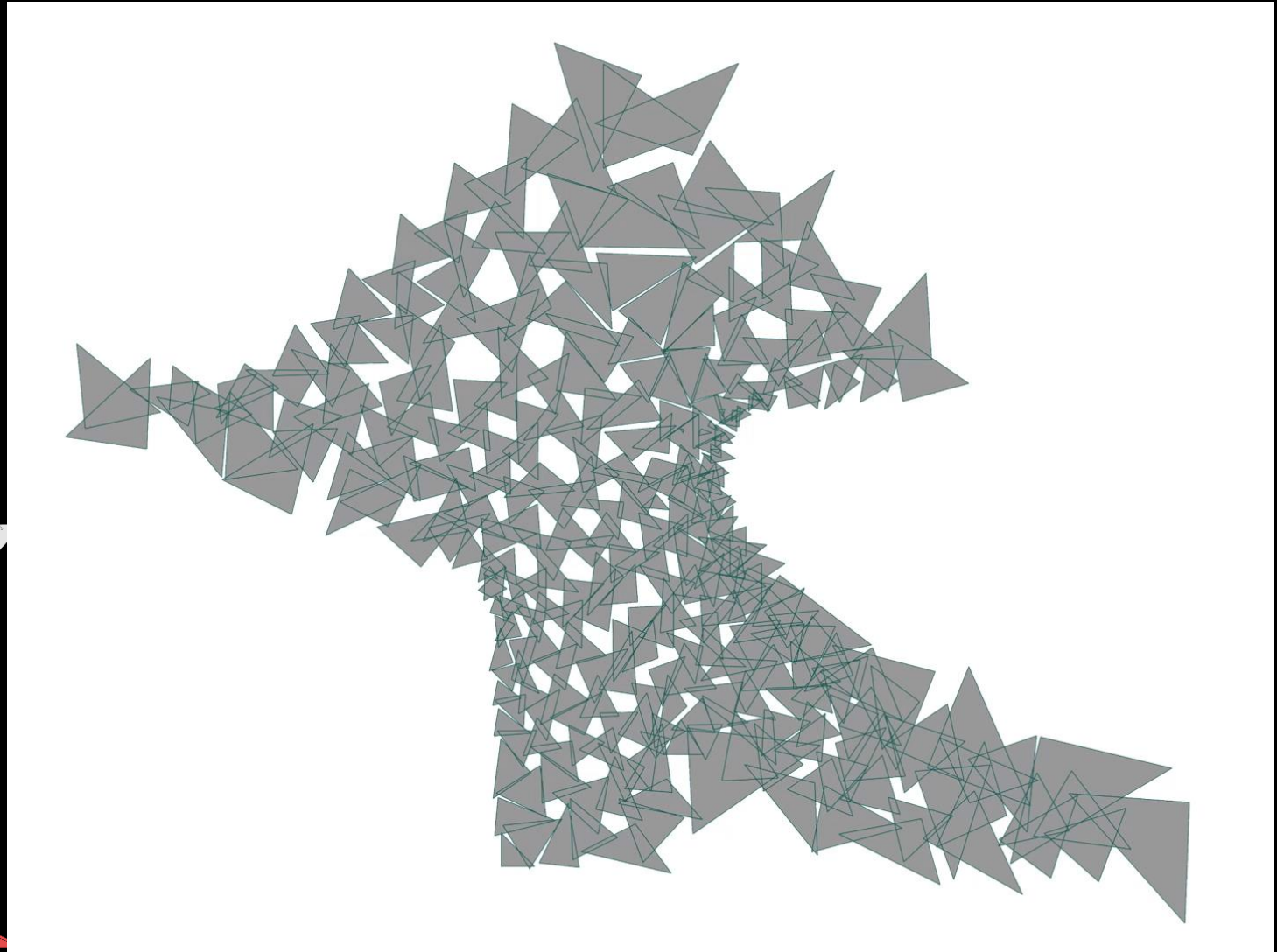
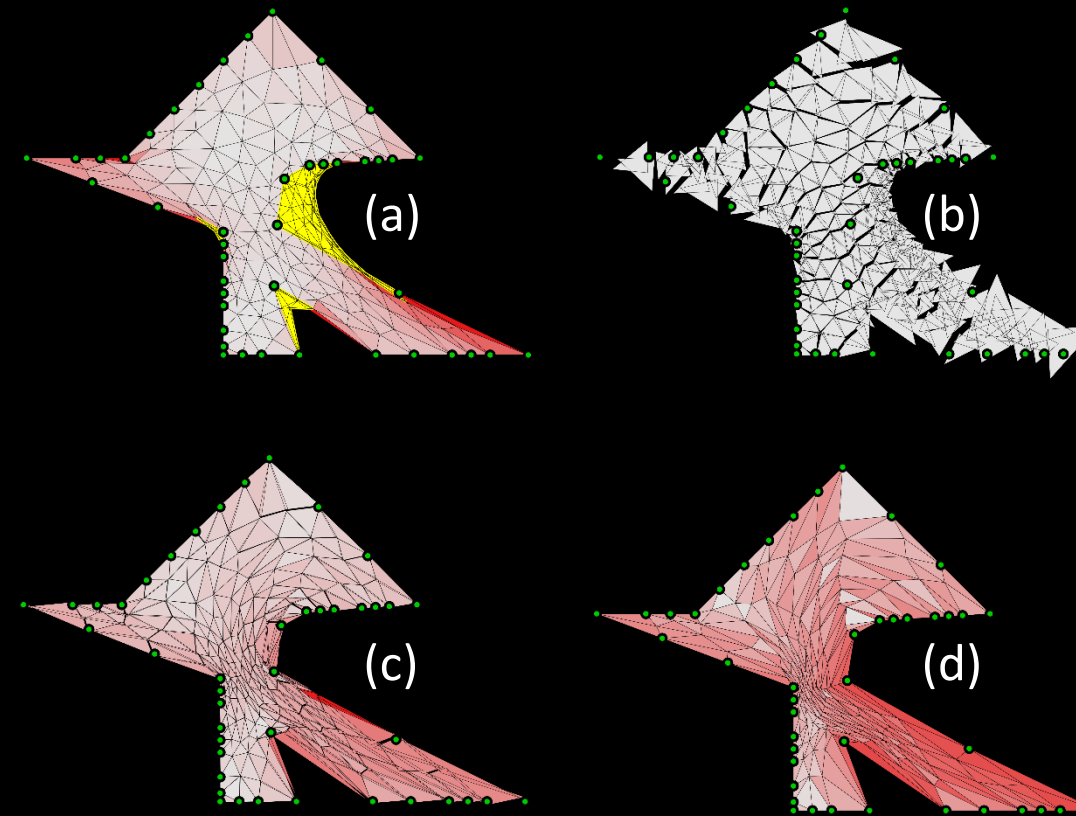
Edge assembly constraints:

$$A_i(v_a - v_b) = A_j(v_a - v_b)$$



Key idea

- disassembly + assembly
 - Treat affine transformation as variables
 - Unconstrained optimization



Distortion control

$$\text{Conformal: } d_i^c = \begin{cases} \frac{1}{2} \|A_i\|_F \|A_i^{-1}\|_F & , d = 2 \\ \frac{1}{8} \left(\|A_i\|_F^2 \|A_i^{-1}\|_F^2 - 1 \right) & , d = 3 \end{cases}$$

$$\text{Volumetric: } d_i^{vol} = \frac{1}{2} \left(\det(A_i) + \frac{1}{\det(A_i)} \right)$$

$$\text{Isometric: } d_i^{iso} = 0.5 \cdot (d_i^c + d_i^{vol})$$

Barrier function on distortion:

1. The type of distortion and distortion bound K are given:

$$E_C^* = \sum_{i=1}^N \frac{e^{s \cdot d_i^*}}{K - d_i^*}$$

2. The type of distortion is not specified or distortion bound $K = \infty$:

$$E_C^* = \sum_{i=1}^N e^{s \cdot d_i^*}$$

Unconstrained optimization problem

Disassembly: project initial A_i^0 into feasible space.

Assembly: unconstrained optimization.

$E_{assembly}$: summation of squares of edge, assembly constraints.

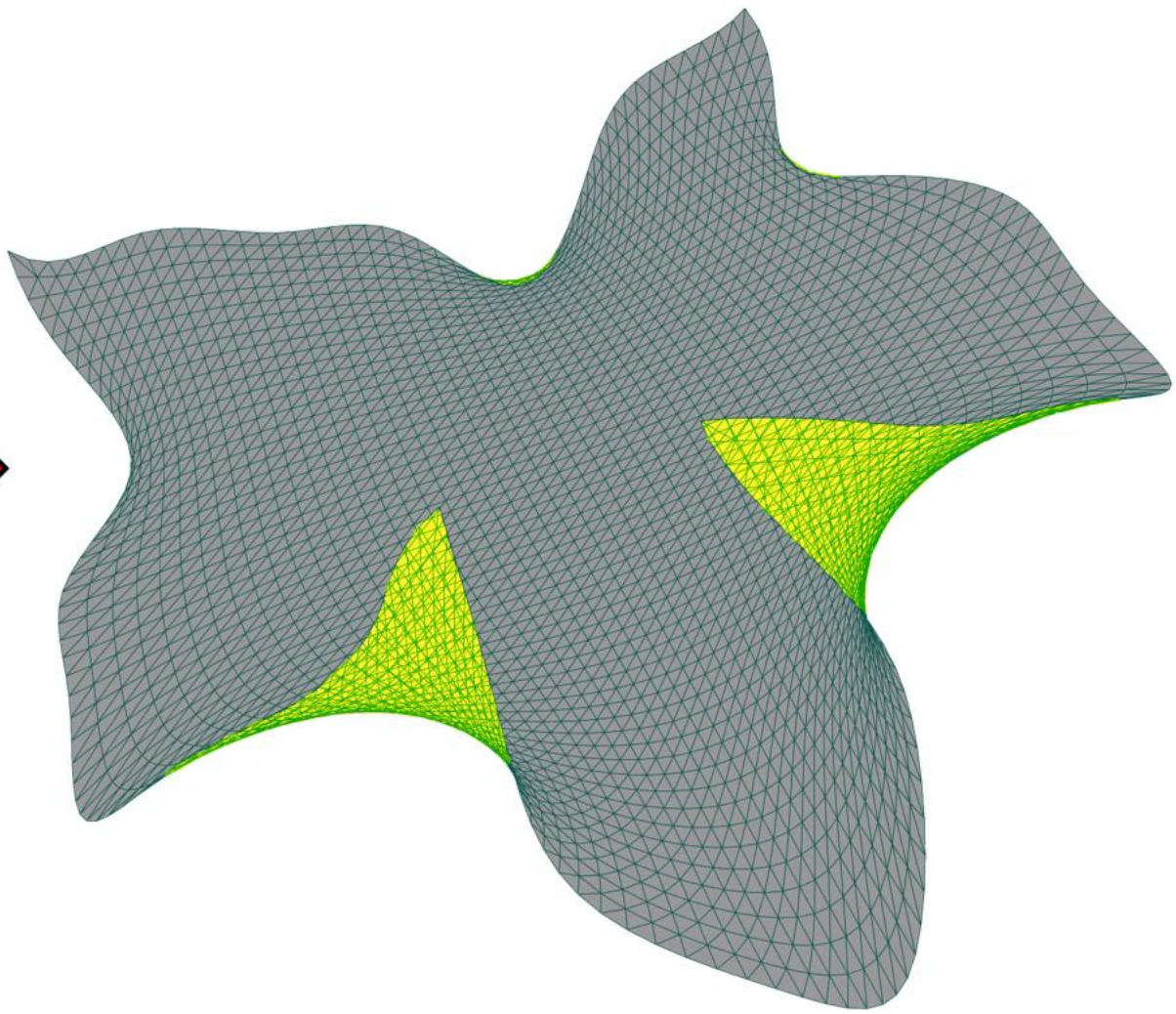
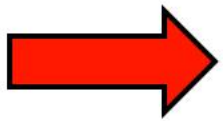
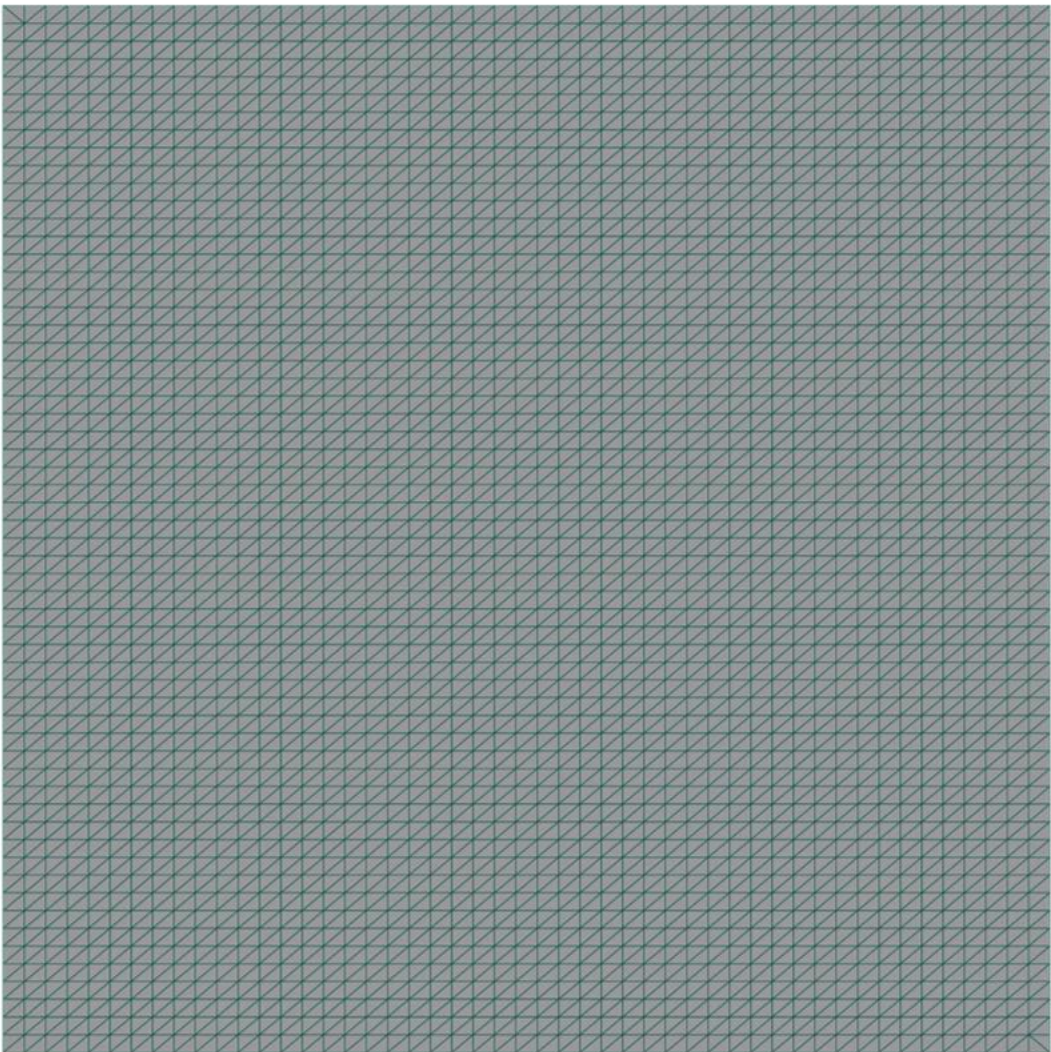
$$\min_{\substack{A_1, \dots, A_N \\ T_1, \dots, T_N}} \lambda E_{assembly} + E_C + \mu E_m$$

E_C : Barrier function on distortion

E_m : users' designed energy

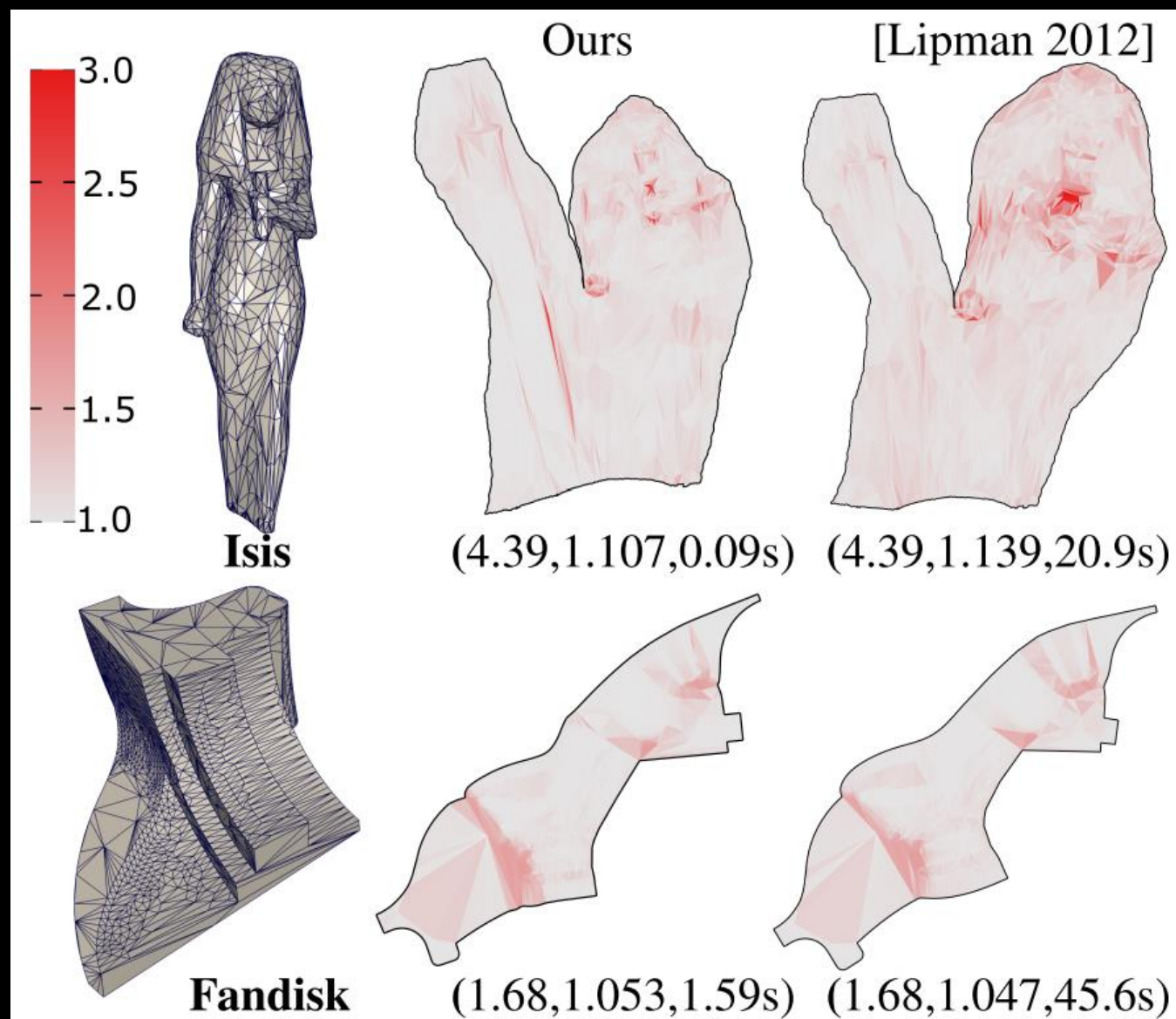
$$\lambda_{k+1} = \min \left(\lambda_{\min} \cdot \max \left(\frac{E_{C,k} + \mu E_{m,k}}{E_{assembly,k}}, 1 \right), \lambda_{\max} \right)$$

1. $E_{assembly}$ dominates the energy, approach zero;
2. λ_{\max} : avoid large distortion.



Optimal bound

- Use the current maximal distortion as the bound for the next round of minimization.



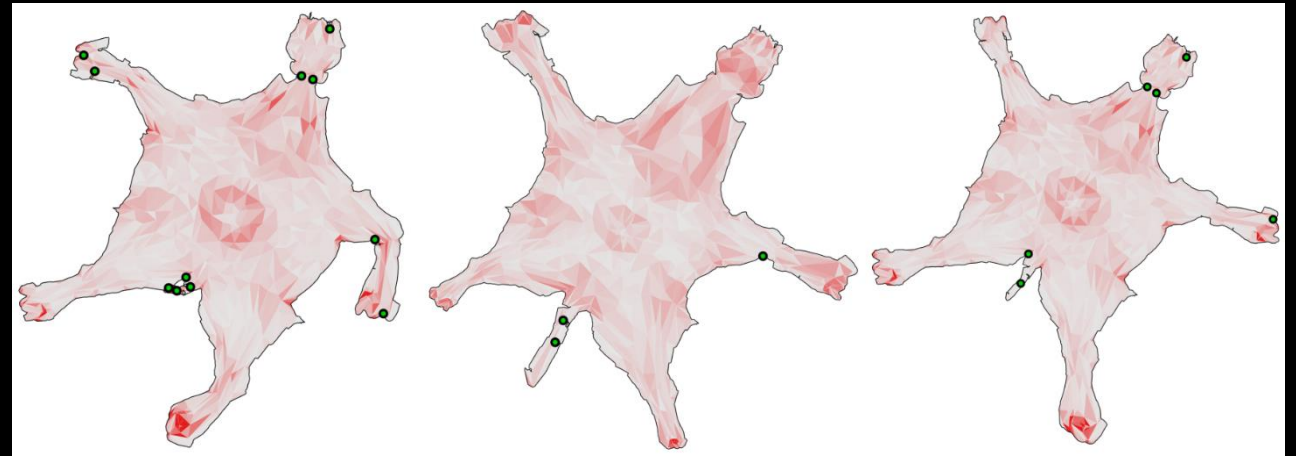
Locally injective mapping

- Requirements for locally injective mapping on triangle mesh:

- 1. inversion-free;
- 2. the sum of triangle angles θ_v around boundary vertex v is less than 2π .

- A barrier term:

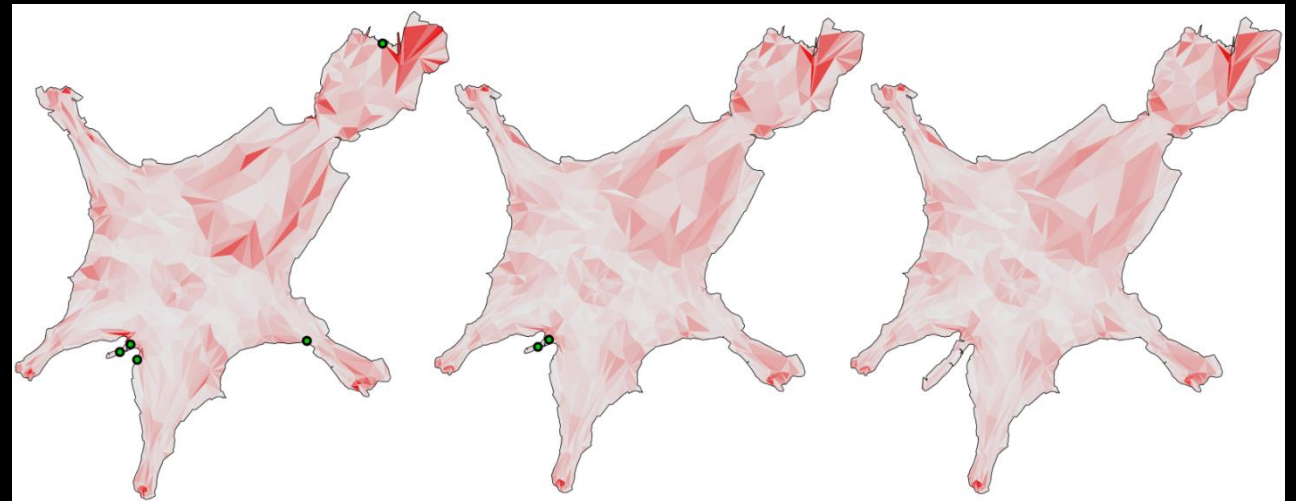
$$E_\theta = \sum_{v \in \partial M} \frac{1}{2\pi - \theta_v}$$



[Lipman 2012]

[Fu et al. 2012]

[Schuller et al. 2013]



[Kovalsky et al. 2012]

Ours without E_θ

Ours with E_θ

Barycentric Coordinates

Xiao-Ming Fu

Outlines

- Introduction
- Barycentric coordinates on convex polygons
- Inverse bilinear coordinates
- Mean value coordinates
- Harmonic Coordinates
- A general construction

Outlines

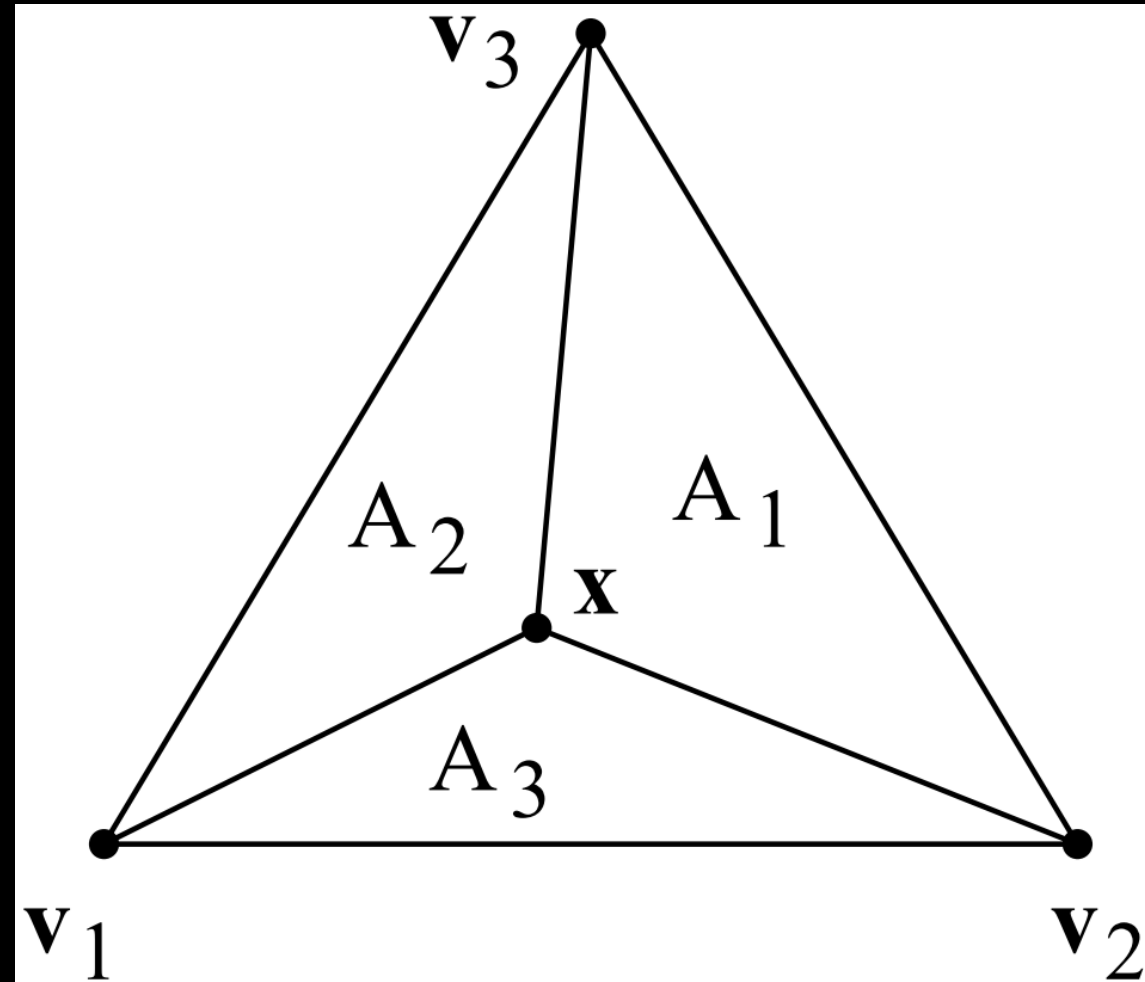
- **Introduction**
- Barycentric coordinates on convex polygons
- Inverse bilinear coordinates
- Mean value coordinates
- Harmonic Coordinates
- A general construction

Barycentric coordinates on triangles

$$x = \phi_1 v_1 + \phi_2 v_2 + \phi_3 v_3,$$

where $\phi_i = \frac{A_i}{A}$.

- Tetrahedron with four sub-tetrahedral.
- Any simplex.



Applications

- Function interpolation
- Function composite
- Defining Bernstein-Bézier polynomials over simplices
- ...

Outlines

- Introduction
- Barycentric coordinates on convex polygons
- Inverse bilinear coordinates
- Mean value coordinates
- Harmonic Coordinates
- A general construction

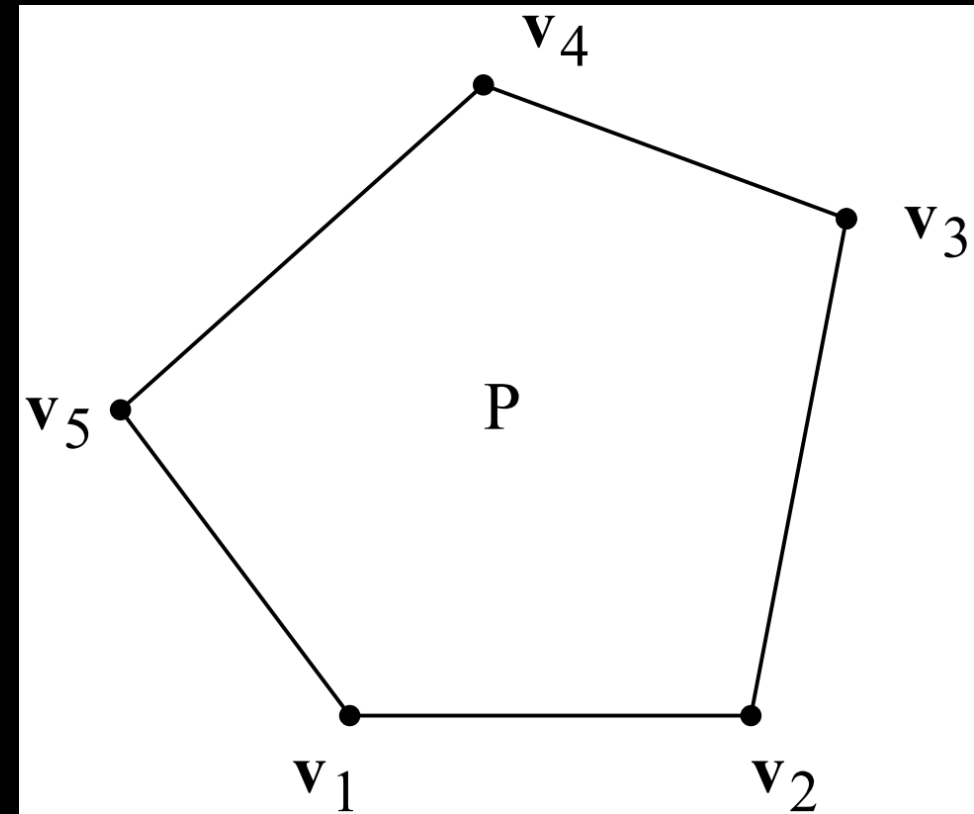
Generalized barycentric coordinates

- Let $P \subset R^2$ be a convex polygon, viewed as an open set, with vertices $v_1, v_2, \dots, v_n, n \geq 3$, in some anticlockwise ordering.

- Any functions $\phi_i: P \rightarrow R, i = 1, \dots, n$, will be called **generalized barycentric coordinates** if $\forall x \in P, \phi_i(x) \geq 0, i = 1, \dots, n$, and

$$\sum_{i=1}^n \phi_i(x) = 1, \sum_{i=1}^n \phi_i(x)v_i = x$$

- ϕ_i : from any point in polygon P to R



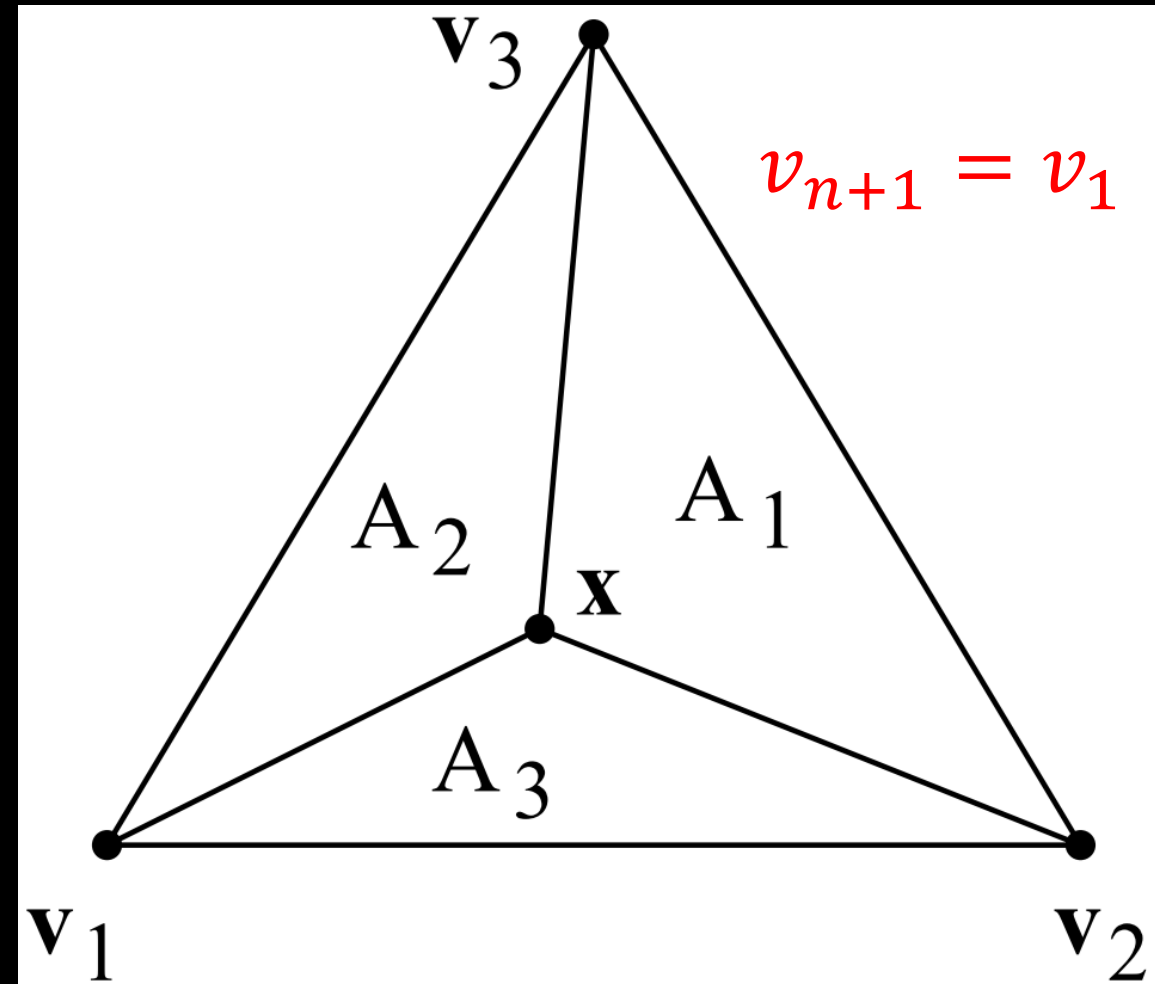
Triangular barycentric coordinates

$$\phi_i(x) = \frac{A(x, v_{i+1}, v_{i+2})}{A(v_1, v_2, v_3)}$$

Note: $A(p_1, p_2, p_3)$ is the signed area of the triangle with vertices $p_k = (x_k, y_k)$, $k = 1, 2, 3$,

$$A(x_1, x_2, x_3) := \frac{1}{2} \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{vmatrix}$$

When $n \geq 4$, it is not unique.



Some basic properties/requirements

- The functions ϕ_i have a unique continuous extension to ∂P , the boundary of P .
- **Lagrange property:** $\phi_i(v_j) = \delta_{ij}$
- **Piecewise linearity** on ∂P
 - $\phi_i((1 - \mu)v_j + \mu v_{j+1}) = (1 - \mu)\phi_i(v_j) + \mu\phi_i(v_{j+1}), \mu \in [0, 1]$.
- **Interpolation**
 - If $g(x) = \sum_{i=1}^n \phi_i(x)f(v_i), x \in P$, then $g(v_i) = f(v_i)$. We call g a barycentric interpolant to f .
- **Linear precision:** if f is linear then $g = f$.

Some basic properties

- $l_i \leq \phi_i \leq L_i$ where $L_i, l_i: P \rightarrow R$ are the continuous, **piecewise linear functions** over the partitions of P satisfying $L_i(v_j) = l_i(v_j) = \delta_{ij}$. L_i is the least upper bound on ϕ_i and l_i the greatest lower bound.

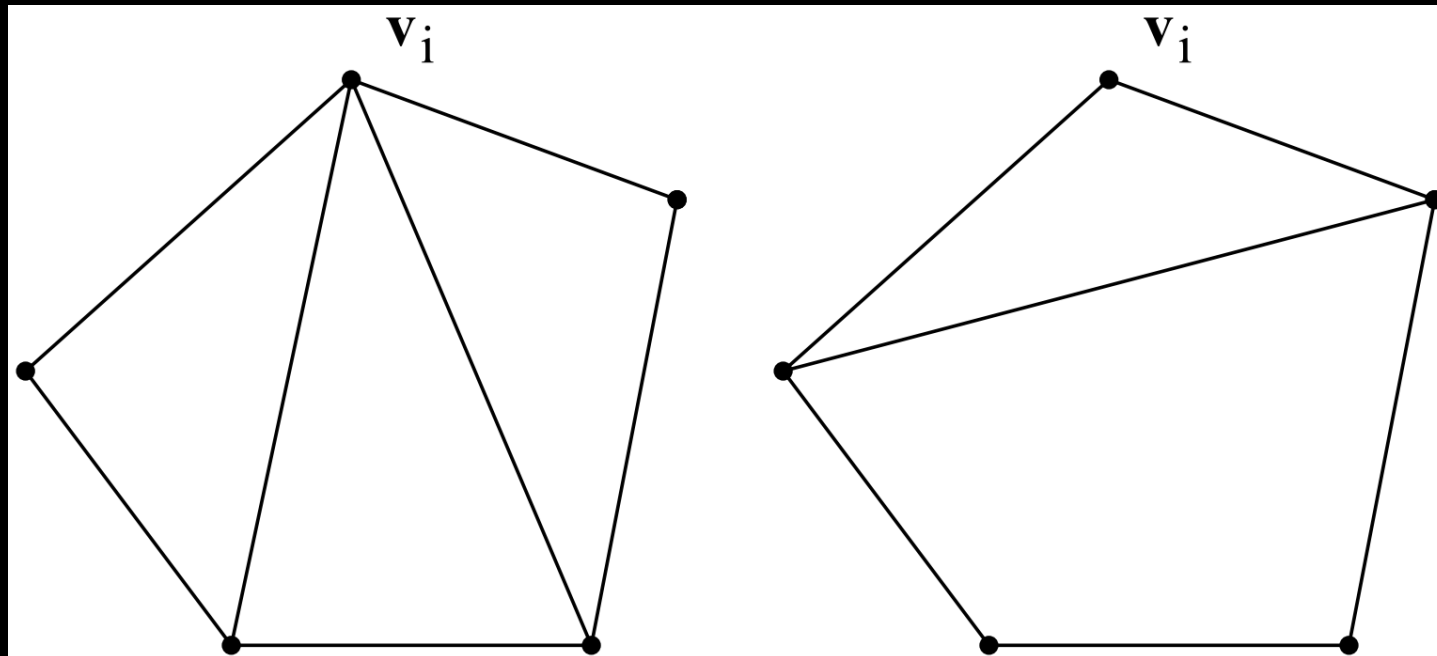


Figure 2.3. Partitions for L_i and l_i .

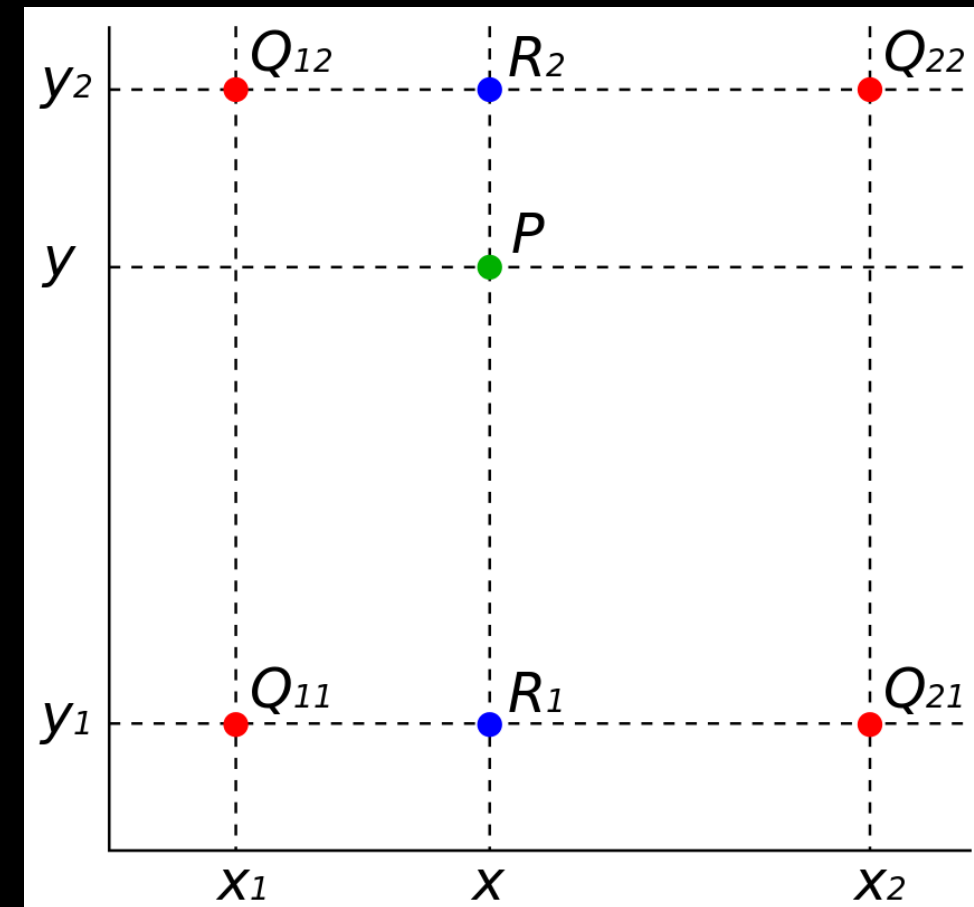
Outlines

- Introduction
- Barycentric coordinates on convex polygons
- **Inverse bilinear coordinates**
- Mean value coordinates
- Harmonic Coordinates
- A general construction

Bilinear interpolation

https://en.wikipedia.org/wiki/Bilinear_interpolation

- Suppose that we want to find the value of the unknown function f at the point (x, y) .
- It is assumed that we know the value of f at the four points $Q_{11} = (x_1, y_1)$, $Q_{12} = (x_1, y_2)$, $Q_{21} = (x_2, y_1)$, $Q_{22} = (x_2, y_2)$.
- Bilinear interpolation: The key idea is to perform linear interpolation **first** in one direction, and **then again** in the other direction.



Bilinear interpolation

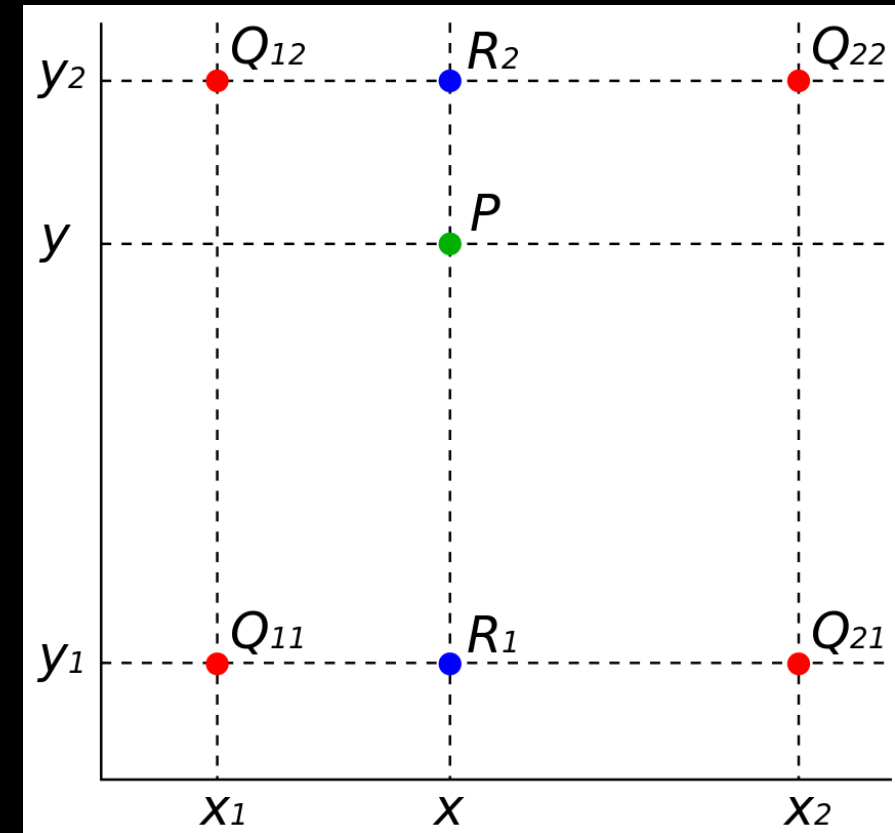
https://en.wikipedia.org/wiki/Bilinear_interpolation

x-direction

$$f(x, y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$
$$f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$

y-direction

$$f(x, y) = \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2)$$
$$= \frac{(y_2 - y)(x_2 - x_1)}{(y_2 - y_1)(x_2 - x_1)} f(Q_{11}) + \frac{(y_2 - y)(x - x_1)}{(y_2 - y_1)(x - x_1)} f(Q_{21})$$
$$+ \frac{(y - y_1)(x_2 - x_1)}{(y_2 - y_1)(x_2 - x_1)} f(Q_{12}) + \frac{(y - y_1)(x - x_1)}{(y_2 - y_1)(x - x_1)} f(Q_{22})$$

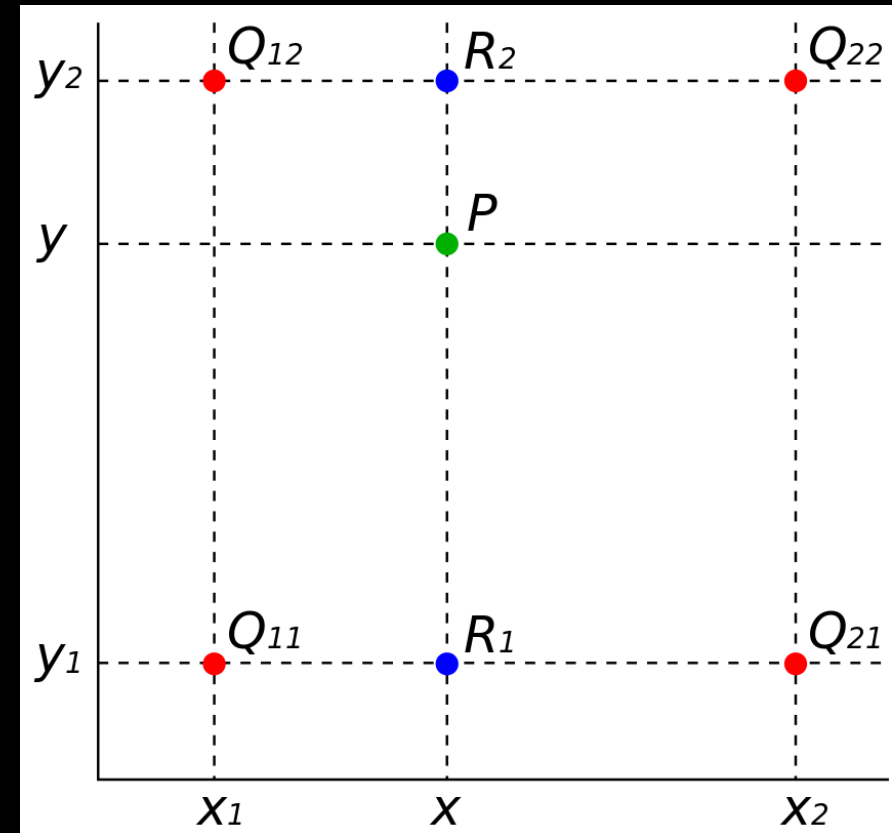


Unit square

- Suppose $x_1 = y_1 = 0, x_2 = y_2 = 1$

$f(x, y)$

$$= (1 - x)(1 - y) \cdot f(0,0) + x(1 - y) \cdot f(1,0) \\ + (1 - x)y \cdot f(0,1) + xy \cdot f(1,1)$$



Convex quadrilaterals

- View P as the image of a bilinear map from the unit square $[0,1] \times [0,1]$.

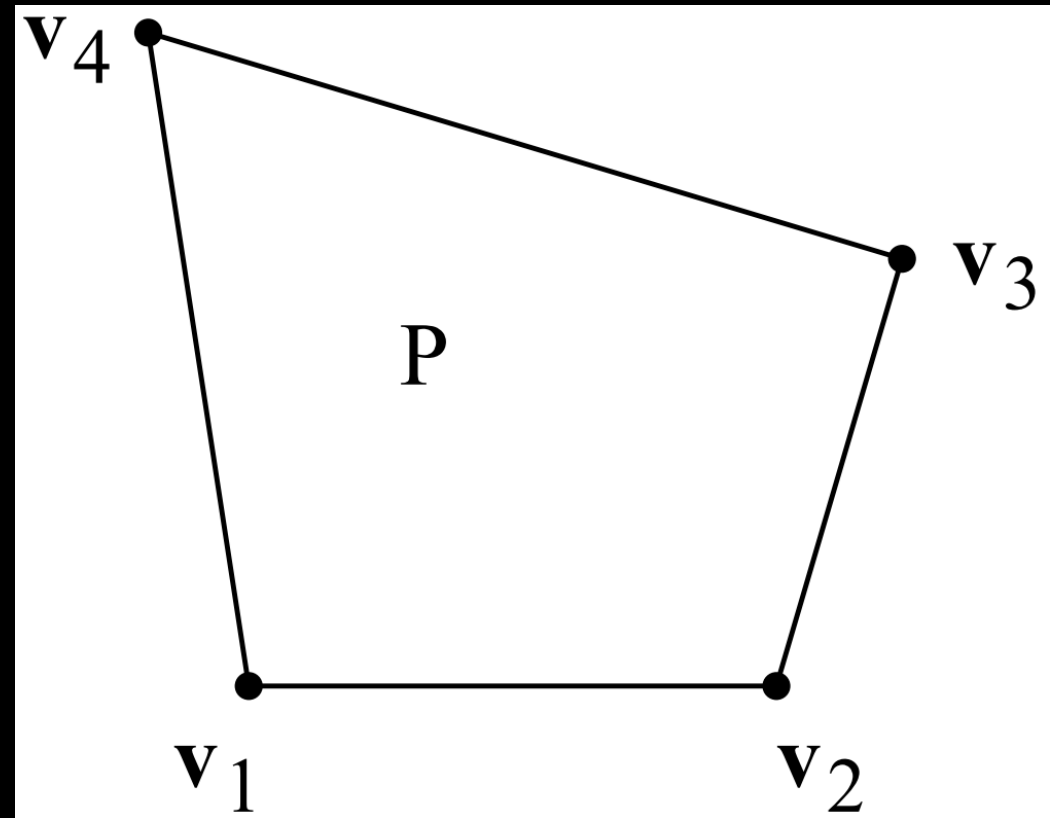
- For each $x \in P$, there exist unique $\lambda, \mu \in (0,1)$ such that

$$(1 - \lambda)(1 - \mu)v_1 + \lambda(1 - \mu)v_2 + \lambda\mu v_3 + (1 - \lambda)\mu v_4 = x$$

and so the four functions

$$\phi_1(x) = (1 - \lambda)(1 - \mu), \phi_2(x) = \lambda(1 - \mu), \phi_3(x) = \lambda\mu, \phi_4(x) = (1 - \lambda)\mu$$

are barycentric coordinates for x .



Inverse of the bilinear map

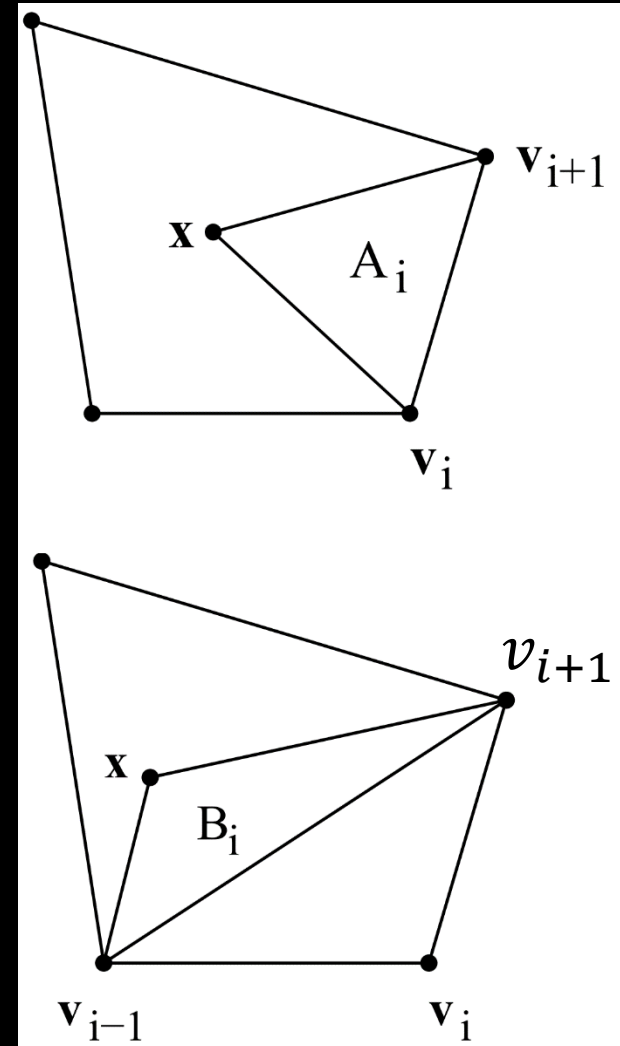
- $A_i(x) = A(x, v_i, v_{i+1}), B_i(x) = A(x, v_{i-1}, v_{i+1})$
- Theorem

$$(\mu, 1 - \lambda, 1 - \mu, \lambda) = \left(\frac{2A_i}{E_i} \right)_{i=1,2,3,4}$$

where $E_i = 2A_i - B_i - B_{i+1} + \sqrt{D}$ and
 $D = B_1^2 + B_2^2 + 2A_1A_3 + 2A_2A_4$.

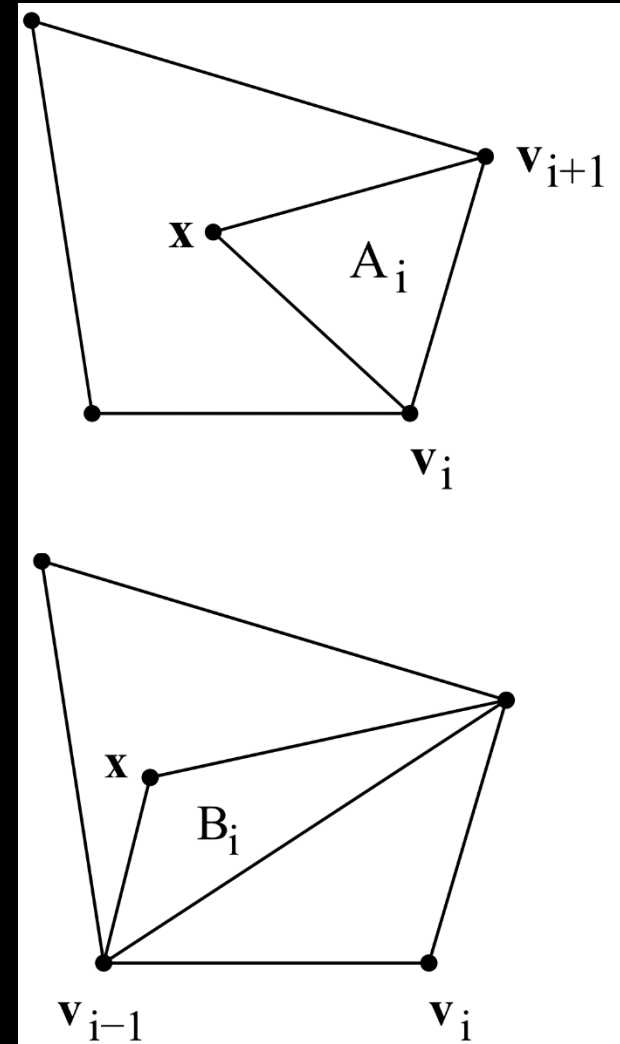
Therefore

$$\phi_i = \frac{4A_{i+1}A_{i+2}}{E_{i+1}E_{i+2}}$$



Inverse of the bilinear map

- Proof process, i.e., **computational process**:
- Known: $(1 - \lambda)(1 - \mu)v_1 + \lambda(1 - \mu)v_2 + \lambda\mu v_3 + (1 - \lambda)\mu v_4 = x$ and the convex quad P
- We want to solve λ, μ .



Proof. It is sufficient to show that

$$\mu = \frac{2A_1}{E_1} = \frac{2A_1}{2A_1 - B_1 - B_2 + \sqrt{D}}$$

as the derivation of the other three terms in (3.2) is similar. Defining the four vectors $\mathbf{d}_i = \mathbf{v}_i - \mathbf{x}$, $i = 1, 2, 3, 4$, (3.1) can be expressed as

$$(1 - \lambda)(1 - \mu)\mathbf{d}_1 + \lambda(1 - \mu)\mathbf{d}_2 + \lambda\mu\mathbf{d}_3 + (1 - \lambda)\mu\mathbf{d}_4 = 0.$$

Next, divide the equation by $\lambda\mu$, and defining

$$\alpha := \frac{1 - \lambda}{\lambda}, \quad \beta := \frac{1 - \mu}{\mu},$$

the equation becomes

$$\alpha\beta\mathbf{d}_1 + \beta\mathbf{d}_2 + \mathbf{d}_3 + \alpha\mathbf{d}_4 = 0.$$

By writing this as

$$\alpha(\beta\mathbf{d}_1 + \mathbf{d}_4) + (\beta\mathbf{d}_2 + \mathbf{d}_3) = 0,$$

and taking the cross product of it with $\beta\mathbf{d}_1 + \mathbf{d}_4$ eliminates α :

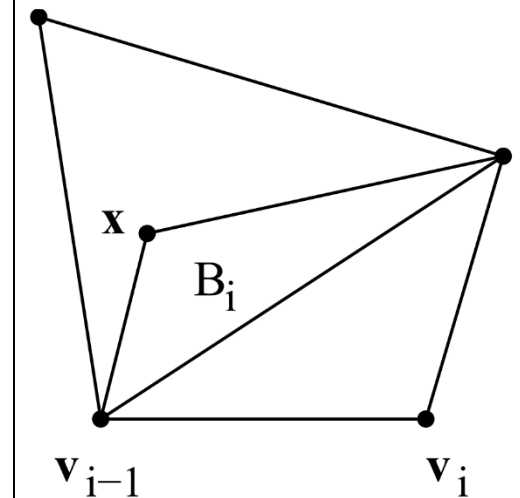
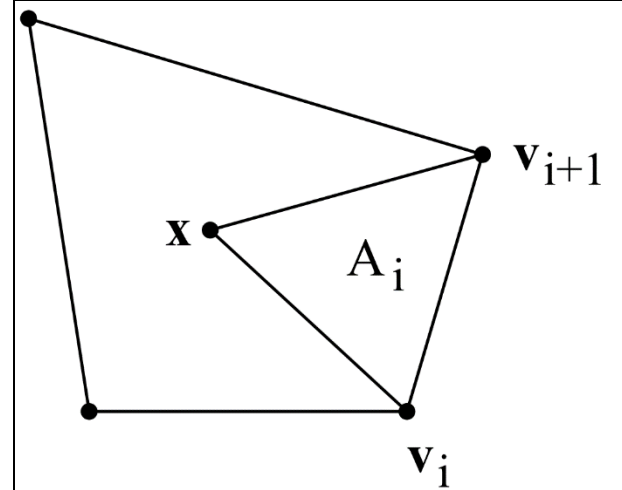
$$(\beta\mathbf{d}_1 + \mathbf{d}_4) \times (\beta\mathbf{d}_2 + \mathbf{d}_3) = 0.$$

(Here, $\mathbf{a} \times \mathbf{b} = (a_1, a_2) \times (b_1, b_2) := a_1b_2 - a_2b_1$.) This is a quadratic equation in β , which, in terms of the A_i and B_i , is

$$A_1\beta^2 + (B_1 + B_2)\beta - A_3 = 0.$$

The discriminant is

$$D = (B_1 + B_2)^2 + 4A_1A_3 > 0,$$



and so the equation has real roots, and β is the positive one,

$$\beta = \frac{-B_1 - B_2 + \sqrt{D}}{2A_1}.$$

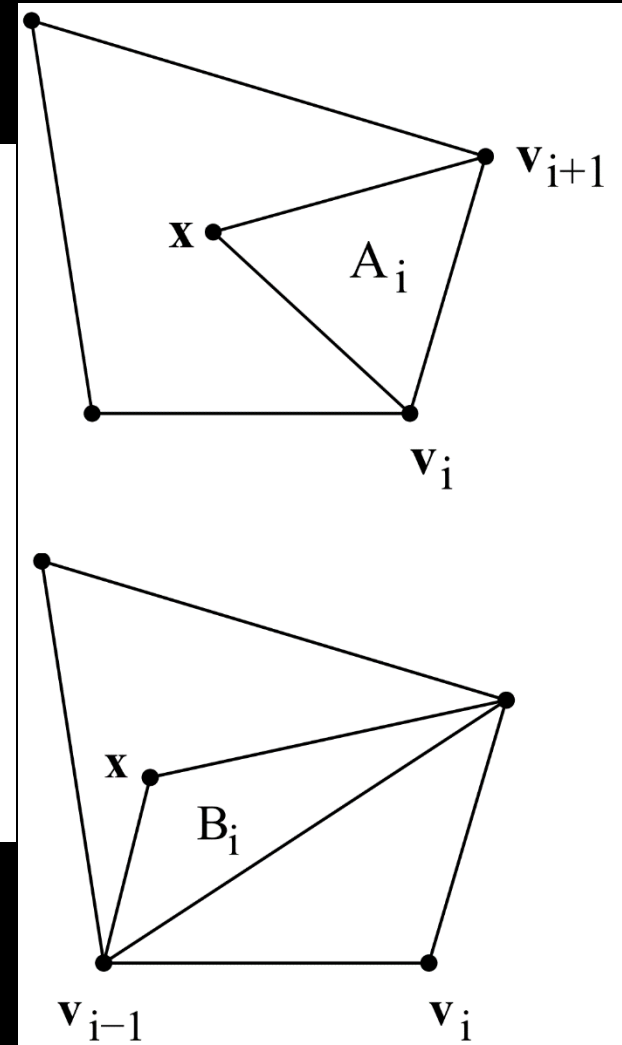
Next observe that

$$B_1 B_2 = A_2 A_4 - A_1 A_3,$$

which follows from taking the cross product of \mathbf{d}_4 with the well known equation

$$(\mathbf{d}_1 \times \mathbf{d}_2)\mathbf{d}_3 + (\mathbf{d}_2 \times \mathbf{d}_3)\mathbf{d}_1 + (\mathbf{d}_3 \times \mathbf{d}_1)\mathbf{d}_2 = 0.$$

From this we find that D can be expressed as (3.3). From β we now obtain $\mu = 1/(1 + \beta)$. \square



Outlines

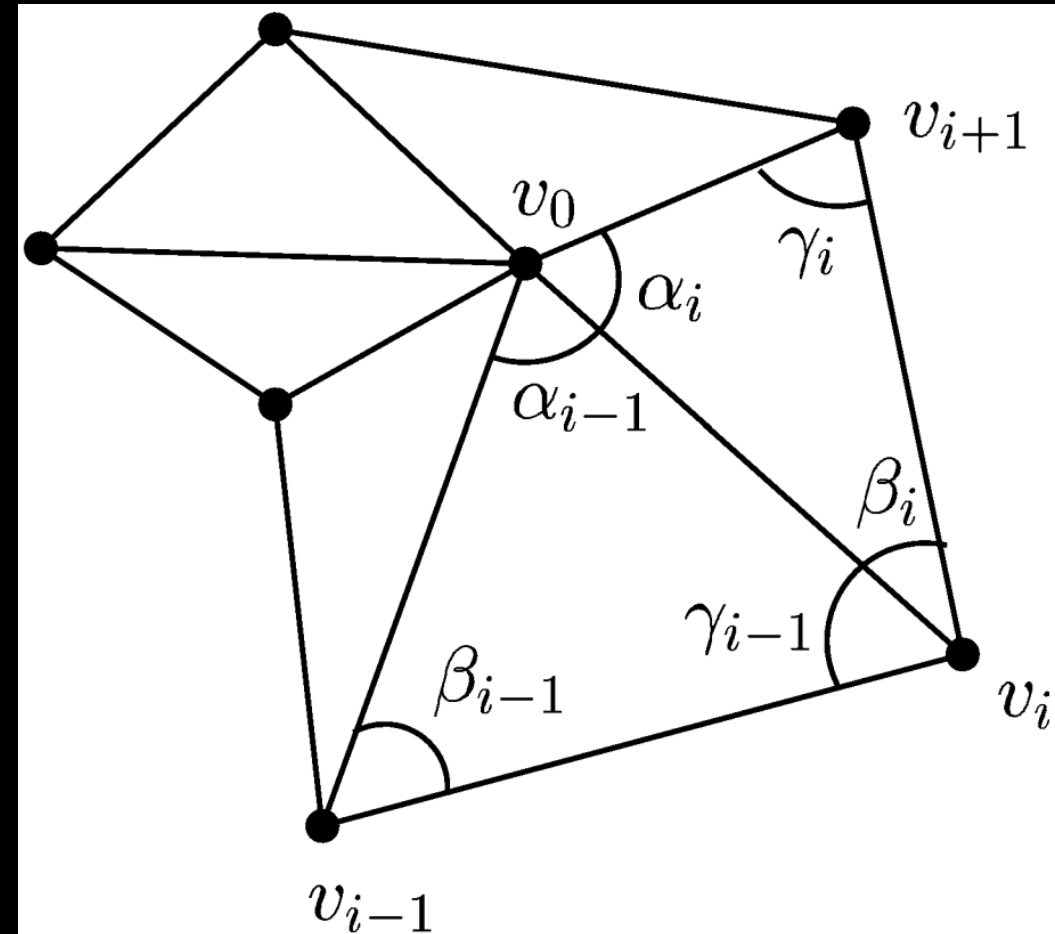
- Introduction
- Barycentric coordinates on convex polygons
- Inverse bilinear coordinates
- **Mean value coordinates**
- Harmonic Coordinates
- A general construction

Mean value coordinates (MVC)

The weights

$$\phi_i = \frac{\omega_i}{\sum_{j=1}^n \omega_j},$$
$$\omega_i = \frac{\tan\left(\frac{\alpha_{i-1}}{2}\right) + \tan\left(\frac{\alpha_i}{2}\right)}{\|v_i - v_0\|}$$

are coordinates for v_0 with respect to v_1, \dots, v_n .



Three requirements

Since $0 < \alpha_i < \pi$, $\phi_i(x) \geq 0$

$\sum_{i=1}^n \phi_i(x) = 1$, by definition

$$\sum_{i=1}^n \phi_i(x) v_i = v_0 \iff \sum_{i=1}^n \phi_i(x) (v_i - v_0) = 0$$

$$\phi_i = \frac{\omega_i}{\sum_{j=1}^n \omega_j},$$
$$\omega_i = \frac{\tan\left(\frac{\alpha_{i-1}}{2}\right) + \tan\left(\frac{\alpha_i}{2}\right)}{\|v_i - v_0\|}$$

Three requirements

- Proof.

$$v_i = v_0 + r_i(\cos \theta_i, \sin \theta_i)$$

Then we have

$$\frac{v_i - v_0}{\|v_i - v_0\|} = (\cos \theta_i, \sin \theta_i)$$

$$\alpha_i = \theta_{i+1} - \theta_i$$

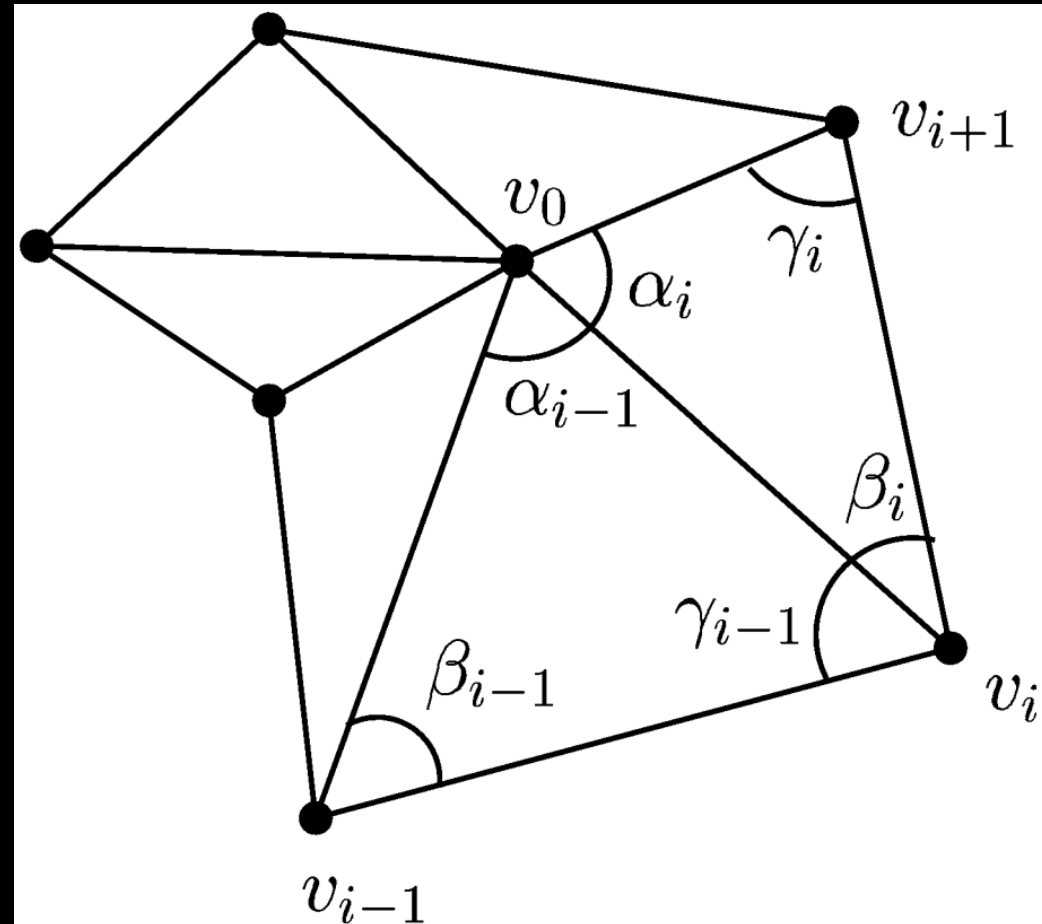
Then,

$$\sum_{i=1}^n \phi_i(x)(v_i - v_0)$$

$$= \sum_{i=1}^n \left(\tan\left(\frac{\alpha_{i-1}}{2}\right) + \tan\left(\frac{\alpha_i}{2}\right) \right) (\cos \theta_i, \sin \theta_i)$$

$$\phi_i = \frac{\omega_i}{\sum_{j=1}^n \omega_j},$$

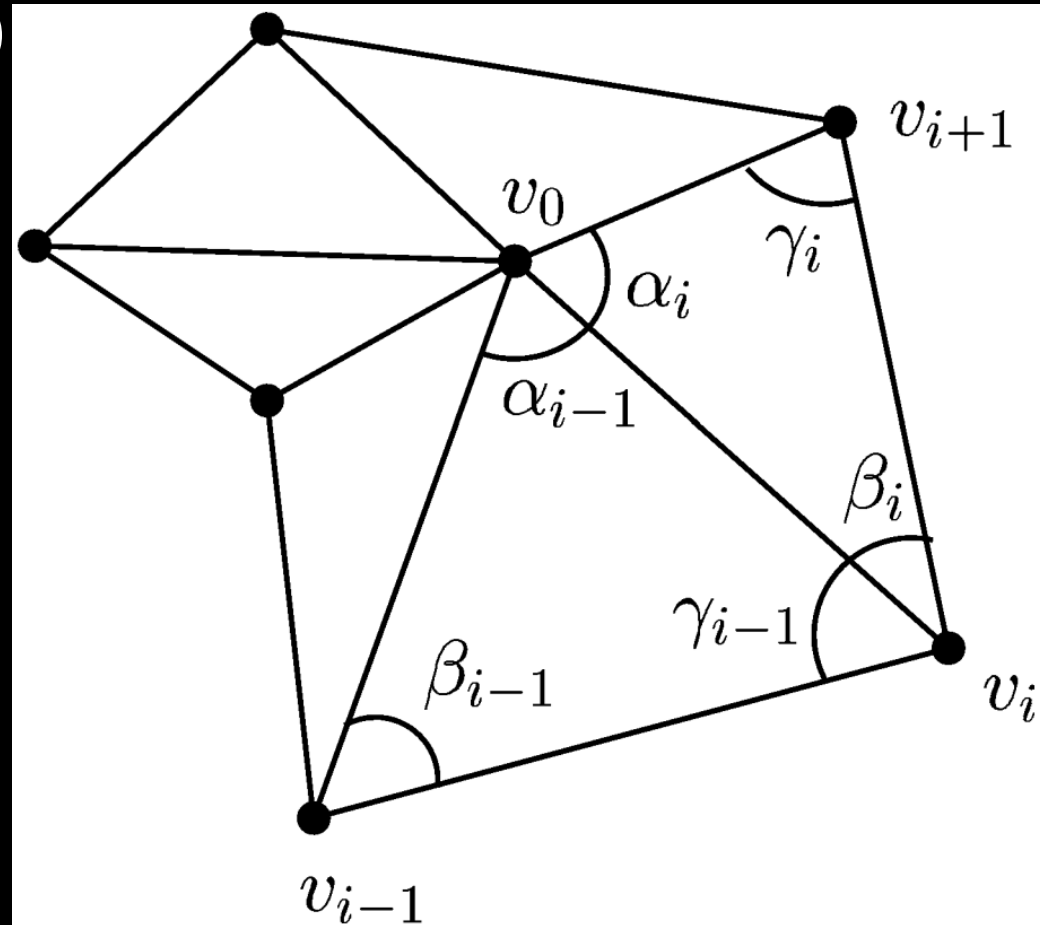
$$\omega_i = \frac{\tan\left(\frac{\alpha_{i-1}}{2}\right) + \tan\left(\frac{\alpha_i}{2}\right)}{\|v_i - v_0\|}$$



Proof

$$\sum_{i=1}^n \left(\tan\left(\frac{\alpha_{i-1}}{2}\right) + \tan\left(\frac{\alpha_i}{2}\right) \right) (\cos \theta_i, \sin \theta_i)$$
$$= \sum_{i=1}^n \tan\left(\frac{\alpha_i}{2}\right) ((\cos \theta_i, \sin \theta_i))$$

$$\phi_i = \frac{\omega_i}{\sum_{j=1}^n \omega_j},$$
$$\omega_i = \frac{\tan\left(\frac{\alpha_{i-1}}{2}\right) + \tan\left(\frac{\alpha_i}{2}\right)}{\|v_i - v_0\|}$$

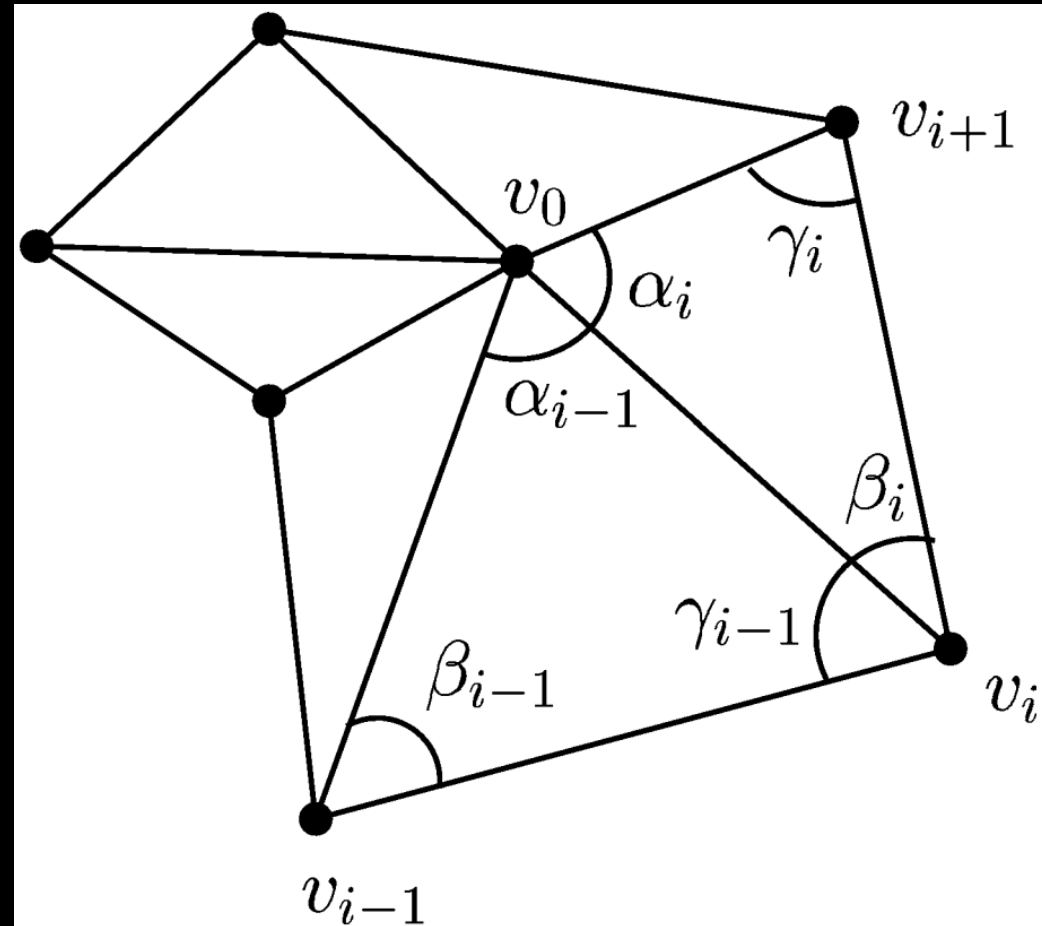


Proof

Since

$$\begin{aligned}
 \cos \theta &= \frac{\cos \theta \sin(\theta_{i+1} - \theta_i)}{\sin(\theta_{i+1} - \theta_i)} \\
 &= \frac{\cos \theta \sin(\theta_{i+1}) \cos \theta_i - \cos \theta \sin(\theta_i) \cos \theta_{i+1}}{\sin(\alpha_i)} \\
 &= \frac{\cos \theta \sin(\theta_{i+1}) \cos \theta_i - \sin \theta \cos \theta_i \cos \theta_{i+1}}{\sin(\alpha_i)} \\
 &\quad + \frac{\sin \theta \cos \theta_i \cos \theta_{i+1} - \cos \theta \sin(\theta_i) \cos \theta_{i+1}}{\sin(\alpha_i)} \\
 &= \frac{\sin(\theta_{i+1} - \theta) \cos \theta_i}{\sin(\alpha_i)} + \frac{\sin(\theta - \theta_i) \cos \theta_{i+1}}{\sin(\alpha_i)}
 \end{aligned}$$

$$\begin{aligned}
 \phi_i &= \frac{\omega_i}{\sum_{j=1}^n \omega_j}, \\
 \omega_i &= \frac{\tan\left(\frac{\alpha_{i-1}}{2}\right) + \tan\left(\frac{\alpha_i}{2}\right)}{\|v_i - v_0\|}
 \end{aligned}$$



Proof

Similarly,

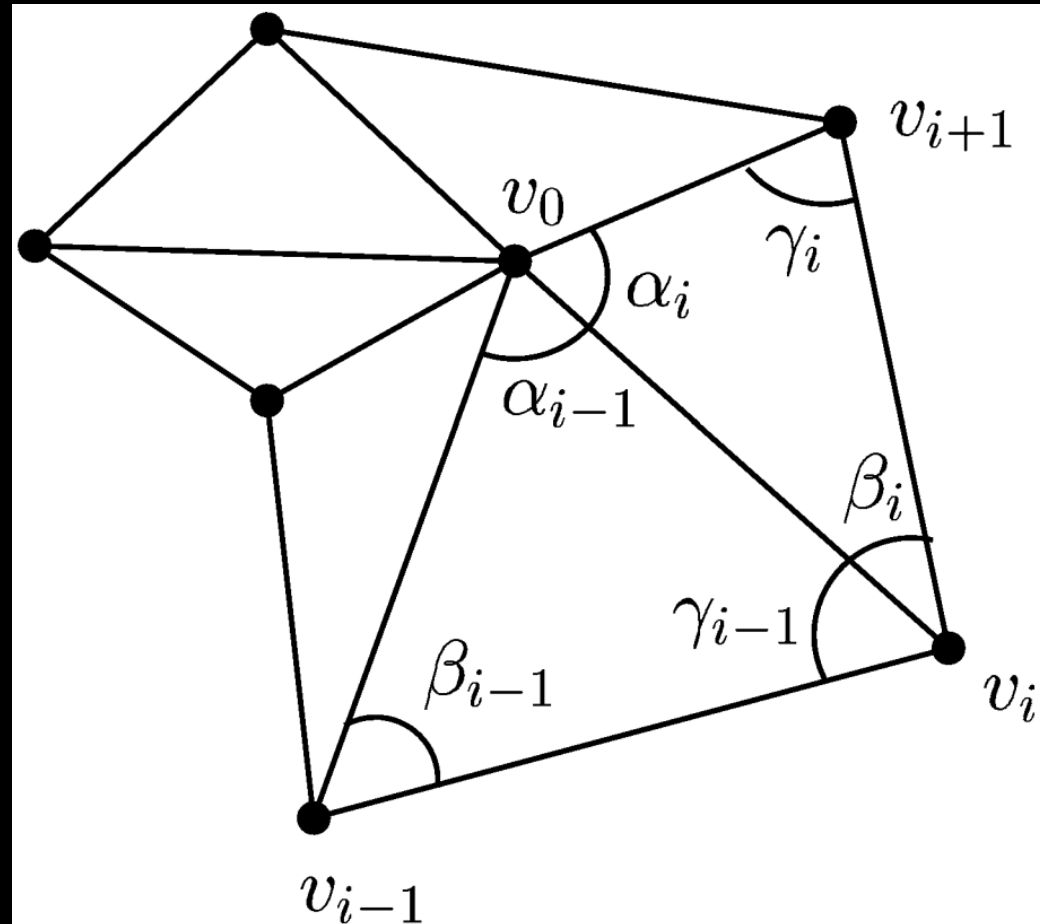
$$\sin \theta = \frac{\sin(\theta_{i+1} - \theta) \sin \theta_i}{\sin(\alpha_i)} + \frac{\sin(\theta - \theta_i) \sin \theta_{i+1}}{\sin(\alpha_i)}$$

As we know

$$\begin{aligned} 0 &= \int_0^\pi (\cos \theta, \sin \theta) d\theta = \sum_{i=1}^n \int_{\theta_i}^{\theta_{i+1}} (\cos \theta, \sin \theta) d\theta \\ &= \sum_{i=1}^n \int_{\theta_i}^{\theta_{i+1}} \frac{\sin(\theta_{i+1} - \theta)}{\sin(\alpha_i)} (\cos \theta_i, \sin \theta_i) \\ &\quad + \frac{\sin(\theta - \theta_i)}{\sin(\alpha_i)} (\cos \theta_{i+1}, \sin \theta_{i+1}) d\theta \end{aligned}$$

$$\phi_i = \frac{\omega_i}{\sum_{j=1}^n \omega_j},$$

$$\omega_i = \frac{\tan\left(\frac{\alpha_{i-1}}{2}\right) + \tan\left(\frac{\alpha_i}{2}\right)}{\|v_i - v_0\|}$$



Proof

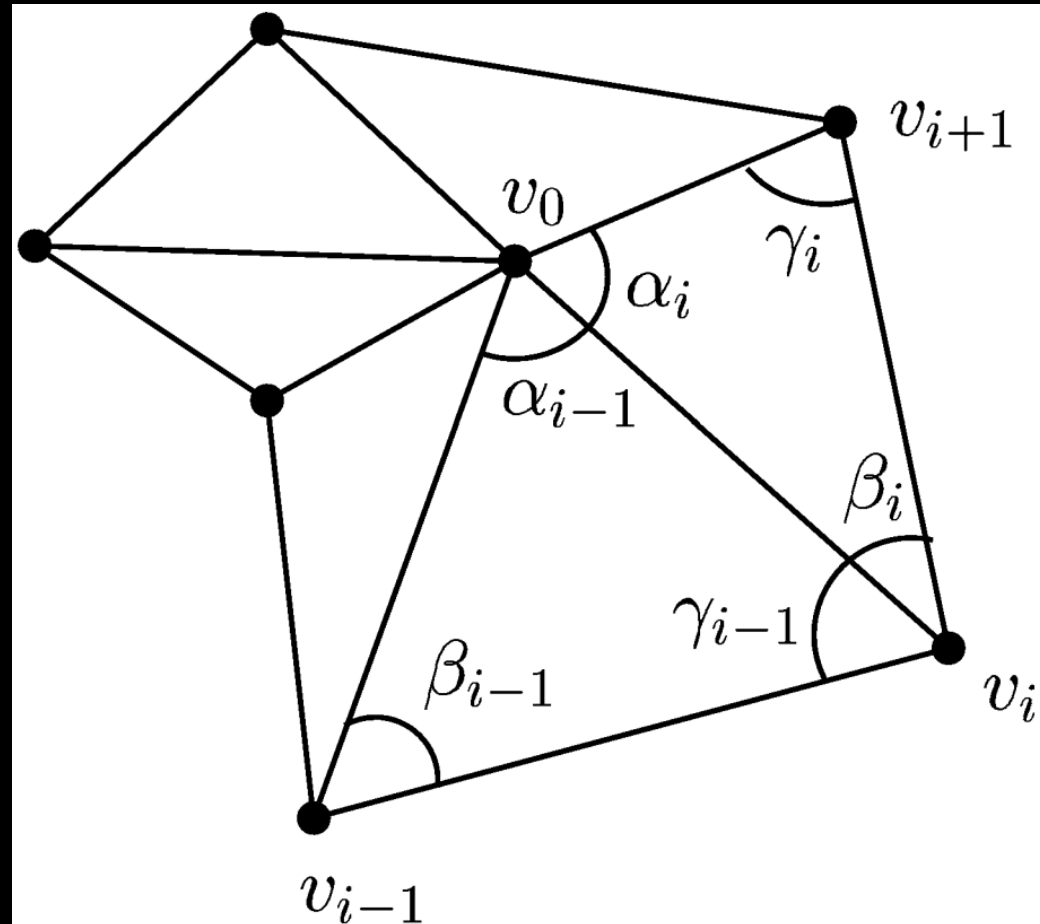
Since

$$\int_{\theta_i}^{\theta_{i+1}} \frac{\sin(\theta_{i+1} - \theta)}{\sin(\alpha_i)} d\theta = \int_{\theta_i}^{\theta_{i+1}} \frac{\sin(\theta - \theta_i)}{\sin(\alpha_i)} d\theta$$
$$= \frac{1 - \cos \alpha_i}{\sin \alpha_i} = \tan \frac{\alpha_i}{2}$$

Thus

$$\sum_{i=1}^n \tan \left(\frac{\alpha_i}{2} \right) ((\cos \theta_i, \sin \theta_i))$$

$$\phi_i = \frac{\omega_i}{\sum_{j=1}^n \omega_j},$$
$$\omega_i = \frac{\tan \left(\frac{\alpha_{i-1}}{2} \right) + \tan \left(\frac{\alpha_i}{2} \right)}{\|v_i - v_0\|}$$



Motivation of MVC

- The motivation behind the coordinates was an attempt to **approximate harmonic maps** by piecewise linear maps over triangulations, in such a way that injectivity is preserved.
 - $u_{xx} + u_{yy} = 0$

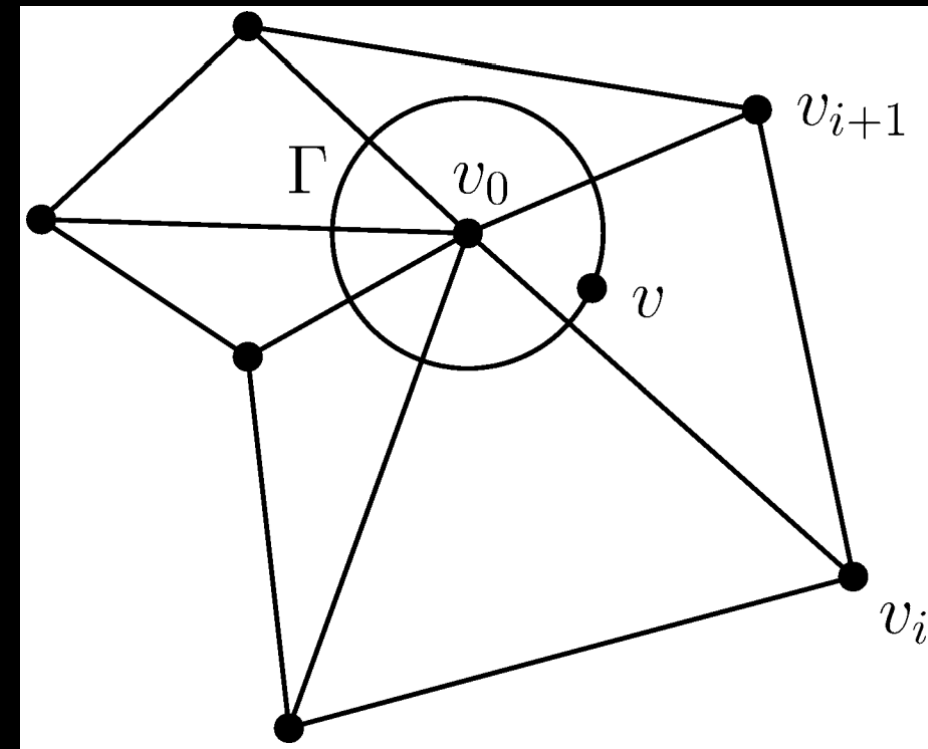
Motivation of MVC

- Suppose we want to approximate the solution u with respect to Dirichlet boundary conditions, $u|_{\partial\Omega} = u_0$, by a piecewise linear function u_T over some triangulation T of Ω .

- $\int_{\Omega} |\nabla u_T|^2 dx$
- boundary conditions
- a sparse linear system
- $u_T(v_0) = \sum_{i=1}^n \phi_i u_T(v_i)$
- Tutte's embedding

- Mean value theorem:

$$u(v_0) = \frac{1}{2\pi r} \int_{\Gamma} u(v) ds$$



Motivation of MVC

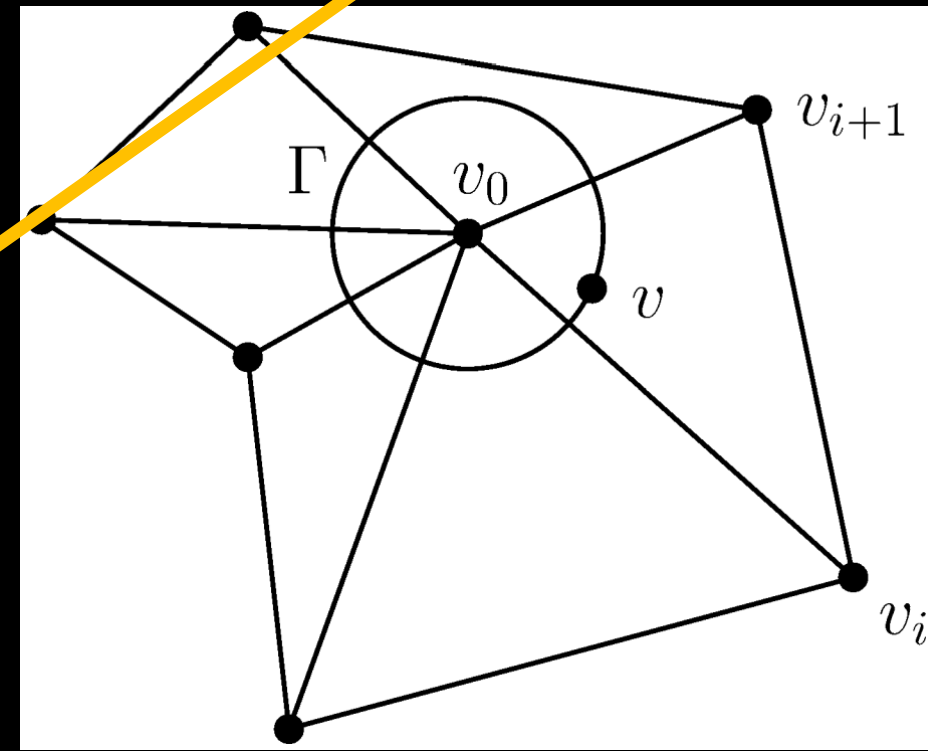
- Thus, we want to find

$$u_T(v_0) = \frac{1}{2\pi r} \int_{\Gamma} u_T(v) ds$$

for r sufficiently small that the disc $B(v_0, r)$ is entirely contained in the union of the triangles containing v_0 .

If above condition is satisfied $\rightarrow u_T(v_0) = \sum_{i=1}^n \phi_i u_T(v_i)$ where ϕ_i is, independent of the choice of r .

$$\phi_i = \frac{\omega_i}{\sum_{j=1}^n \omega_j},$$
$$\omega_i = \frac{\tan\left(\frac{\alpha_{i-1}}{2}\right) + \tan\left(\frac{\alpha_i}{2}\right)}{\|v_i - v_0\|}$$



Motivation of MVC

- Lemma: if $f: T_i \rightarrow R$ is any **linear** function then

$$\int_{\Gamma_i} f(v) ds = r \alpha_i f(v_0) + r^2 \tan\left(\frac{\alpha_i}{2}\right) \left(\frac{f(v_i) - f(v_0)}{\|v_i - v_0\|} + \frac{f(v_{i+1}) - f(v_0)}{\|v_{i+1} - v_0\|} \right)$$

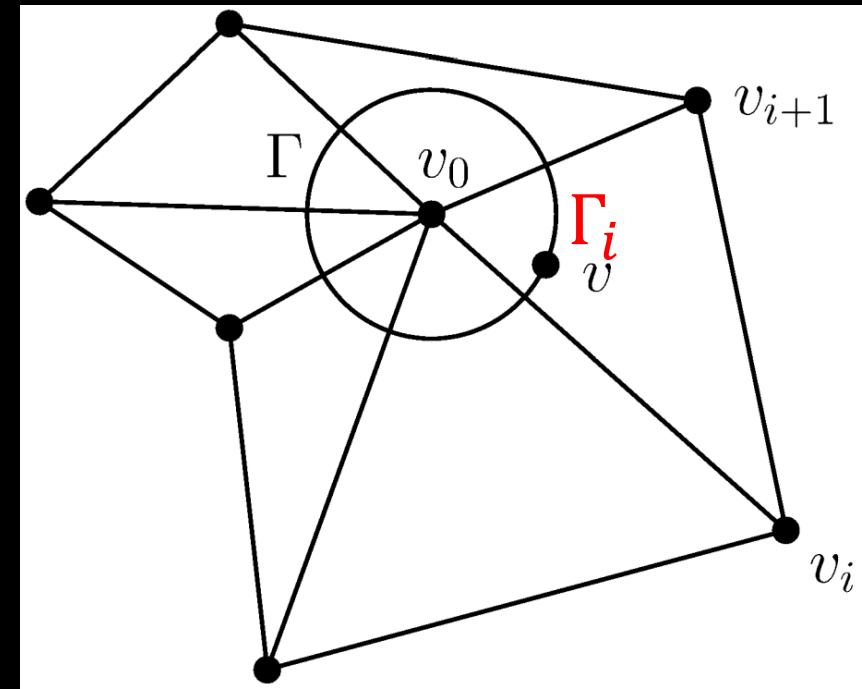
- Proof: $\forall v \in \Gamma_i, v = v_0 + r(\cos\theta, \sin\theta)$

and $v_j = v_0 + r_j(\cos\theta_j, \sin\theta_j)$.

Then, $\int_{\Gamma_i} f(v) ds = r \int_{\theta_i}^{\theta_{i+1}} f(v) d\theta$

Since f is linear, and using barycentric coordinates

we have: $f(v) = f(v_0) + \lambda_1(f(v_i) - f(v_0)) + \lambda_2(f(v_{i+1}) - f(v_0))$



Motivation of MVC

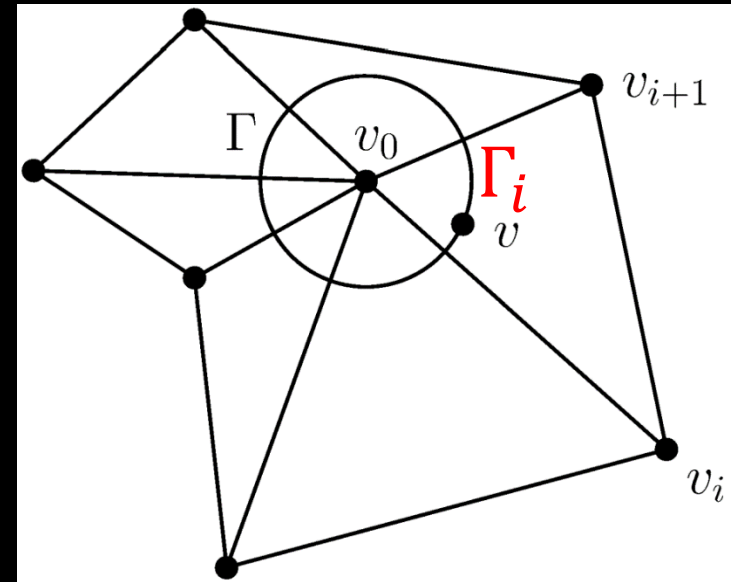
- Proposition: Suppose the piecewise linear function $u_T : \Omega \rightarrow R$ satisfies the local mean value theorem, i.e., for each interior vertex v_0 , it satisfies $u_T(v_0) = \frac{1}{2\pi r} \int_{\Gamma} u_T(v) ds$ for some $r > 0$ suitably small. Then $u_T(v_0)$ is given by the convex combination in $u_T(v_0) = \sum_{i=1}^n \phi_i u_T(v_i)$ with the weights ϕ_i is

$$\phi_i = \frac{\omega_i}{\sum_{j=1}^n \omega_j},$$
$$\omega_i = \frac{\tan\left(\frac{\alpha_{i-1}}{2}\right) + \tan\left(\frac{\alpha_i}{2}\right)}{\|v_i - v_0\|}$$

Proof of Proposition

$$u_{\mathcal{T}}(v_0) = \frac{1}{2\pi r} \int_{\Gamma} u_{\mathcal{T}}(v) ds = \frac{1}{2\pi r} \sum_{i=1}^n \int_{\Gamma_i} u_{\mathcal{T}}(v) ds$$

$$= \frac{1}{2\pi r} \sum_{i=1}^n \left(r \alpha_i u_{\mathcal{T}}(v_0) \right)$$

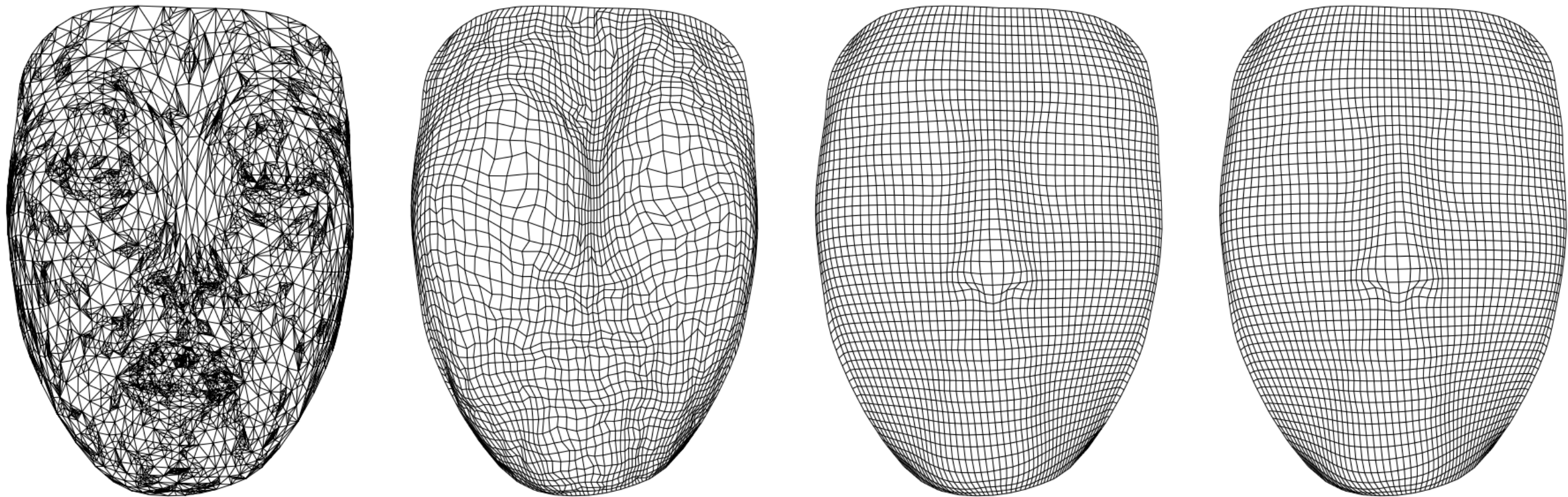


Applications of MVC

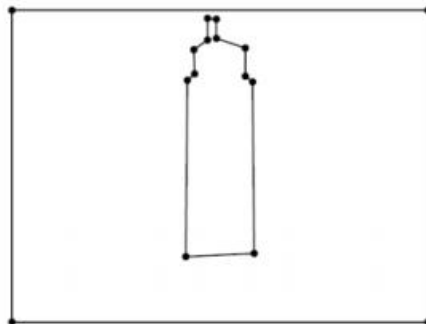
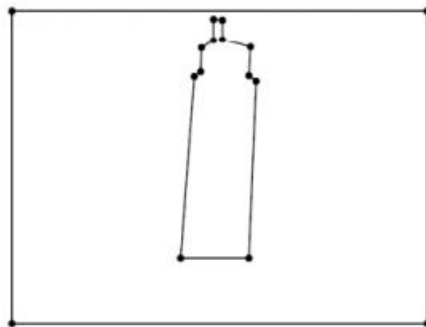
- Parameterization
 - Mean Value Coordinate
- Deformation
 - Mean Value Coordinates for Closed Triangular Meshes
- Poisson image editing
 - Coordinates for Instant Image Cloning
- Diffusion curves/surfaces
 - Volumetric Modeling with Diffusion Surfaces

Parameterization

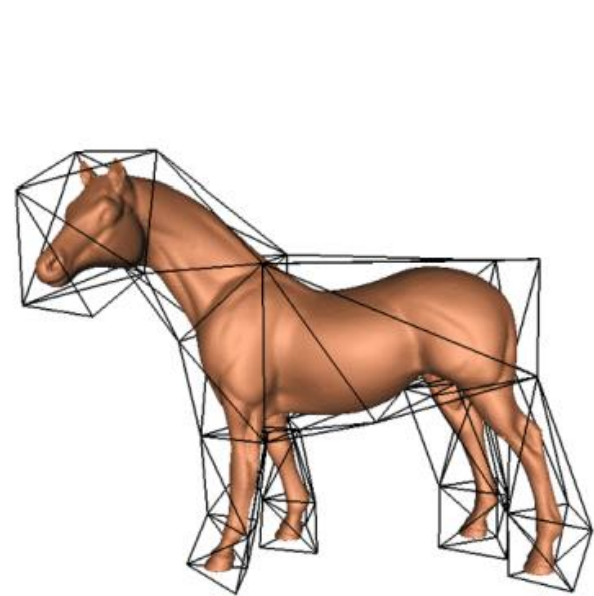
- Compute the coordinates ϕ_i directly from the vertices $v_0, \dots, v_k \in R^3$



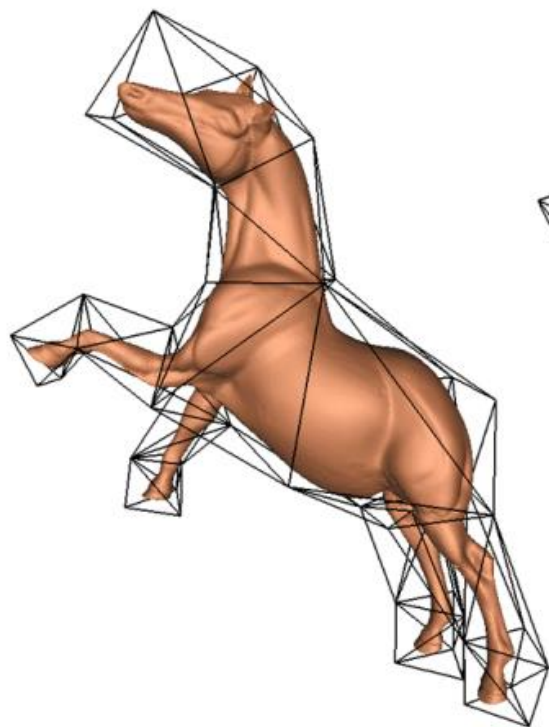
Deformation



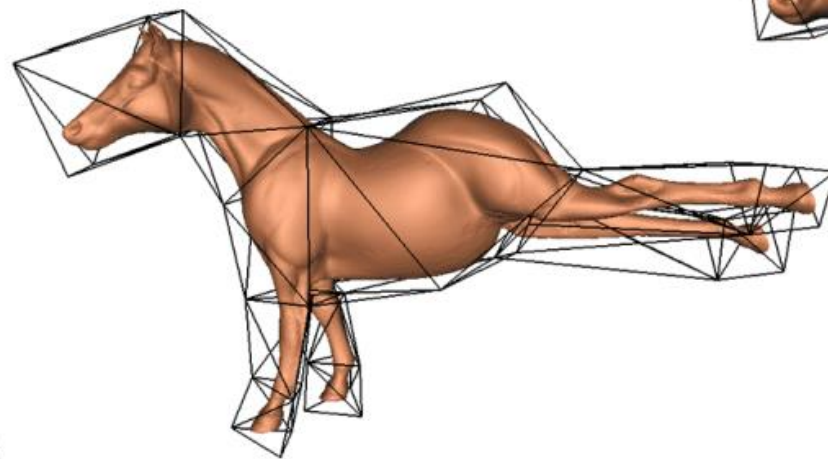
Deformation



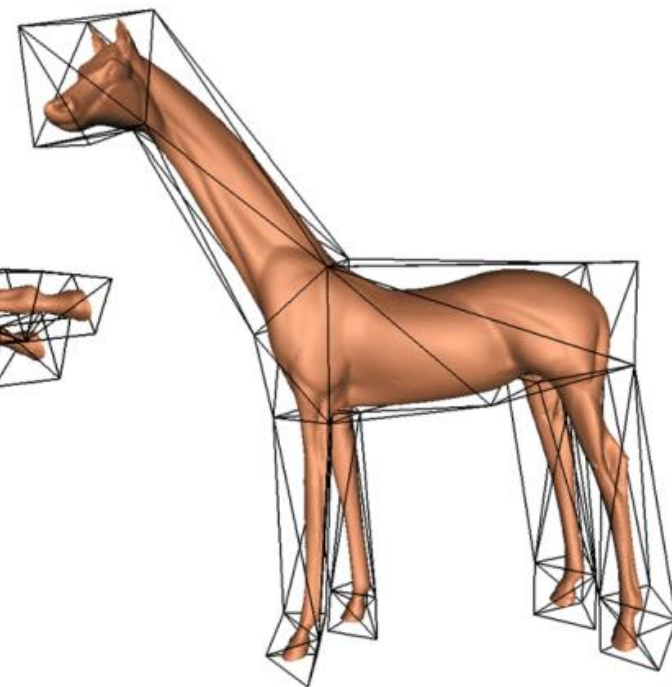
(a)



(b)



(c)

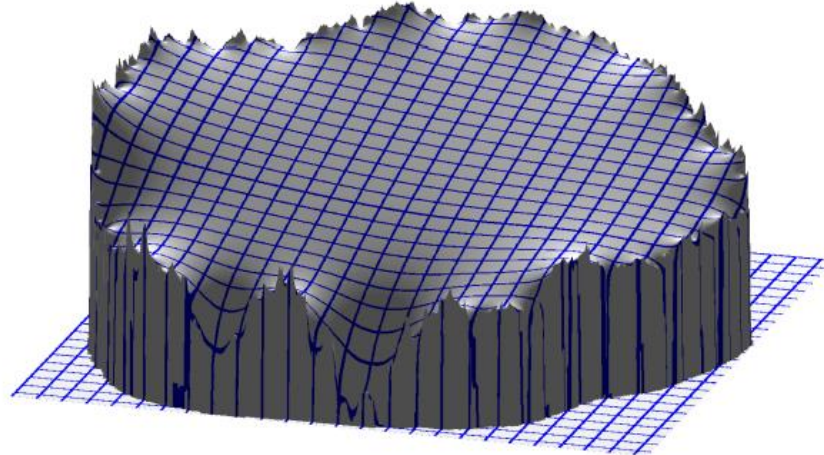


(d)

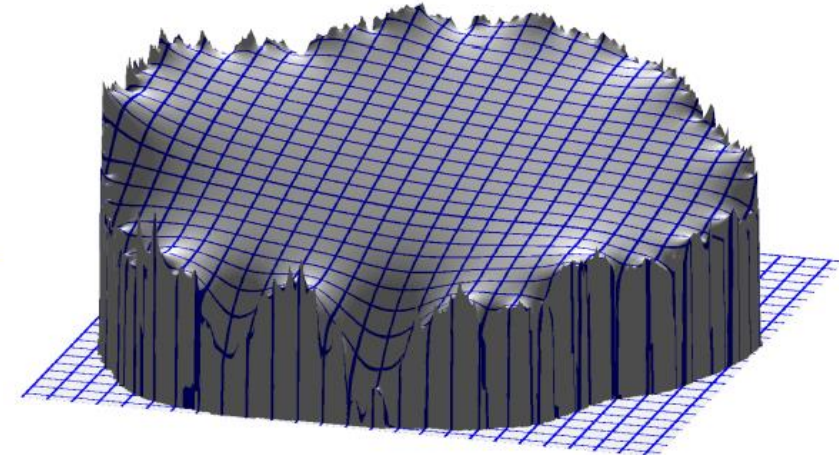
Poisson image editing



(a) Source patch



(b) Laplace membrane



(c) Mean-value membrane



(d) Target image



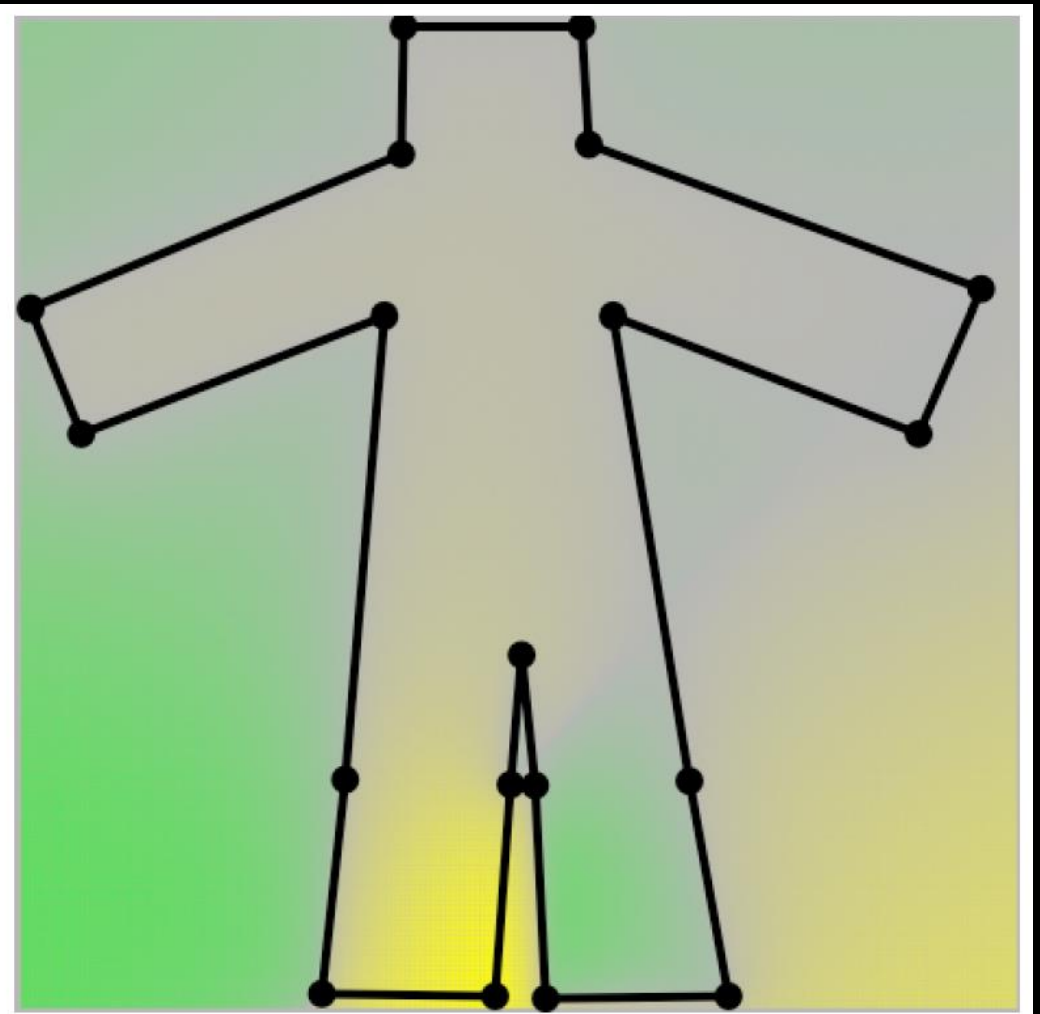
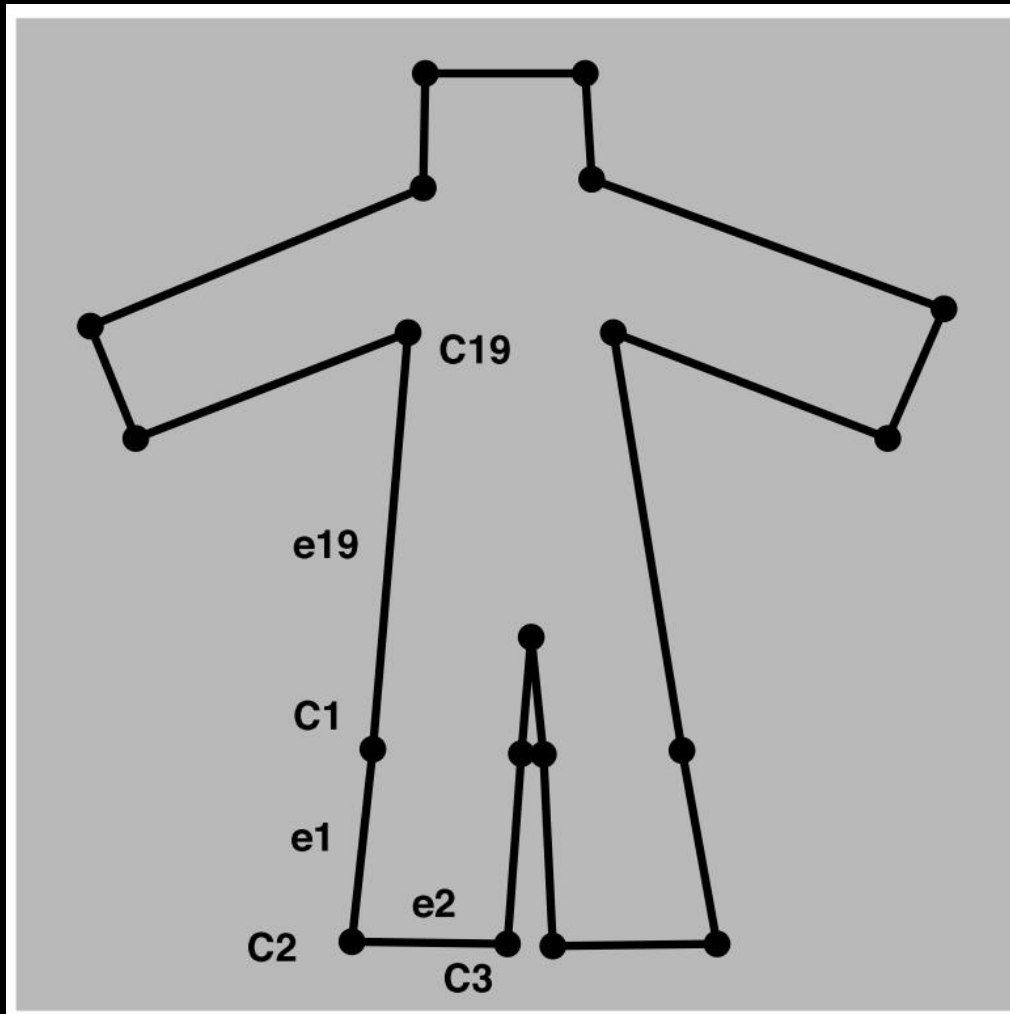
(e) Poisson cloning



(f) Mean-value cloning

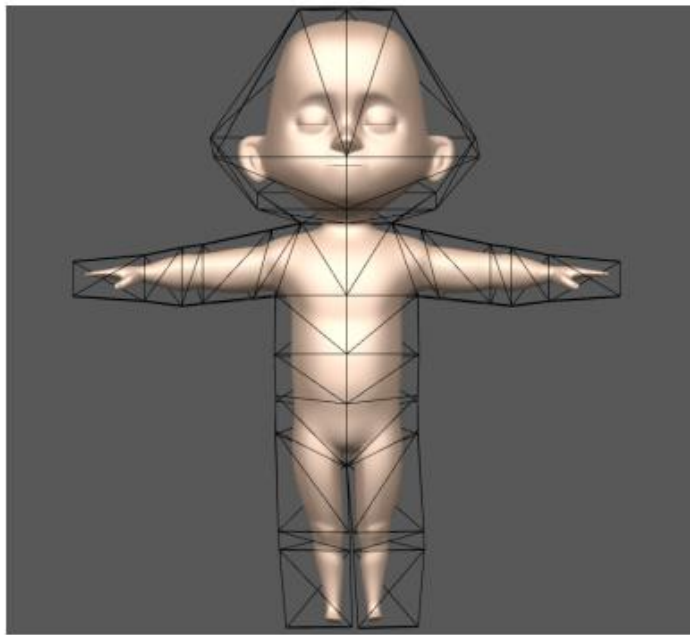
Concave Polygon

Yellow indicates positive values
Green indicates negative values

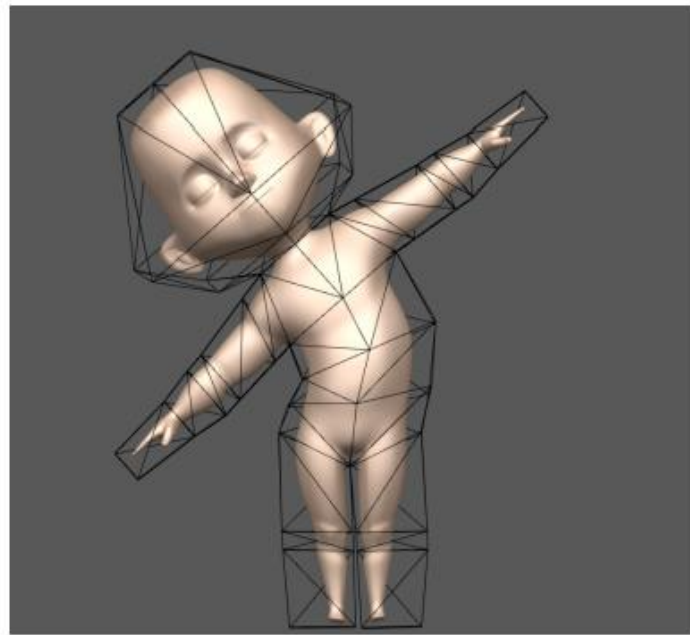


MVC doesn't have

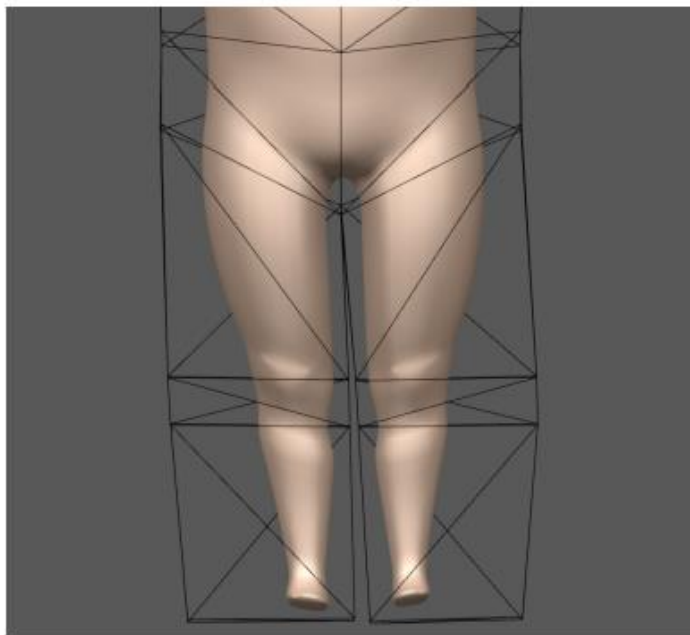
- Non-negativity
 - All weights are positive
- Interior locality
 - Interior locality holds, if, in addition to non-negativity, the coordinate functions have no interior extrema.



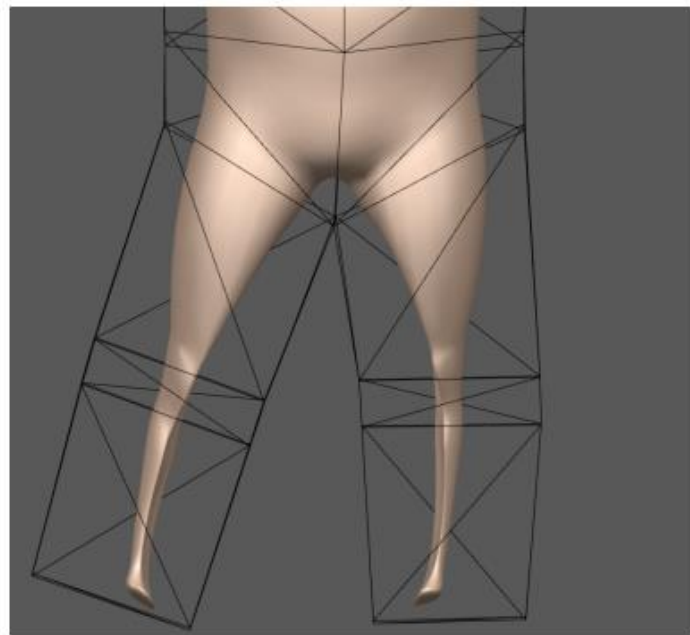
(a)



(b)



(d)



(e)

Outlines

- Introduction
- Barycentric coordinates on convex polygons
- Inverse bilinear coordinates
- Mean value coordinates
- **Harmonic Coordinates**
- A general construction

Harmonic Coordinates

$$\begin{aligned} \nabla^2 \phi_i(x) &= 0, \forall x \in P \\ \text{s.t. } \phi_i(\partial P) &= h_i(\partial P) \end{aligned}$$

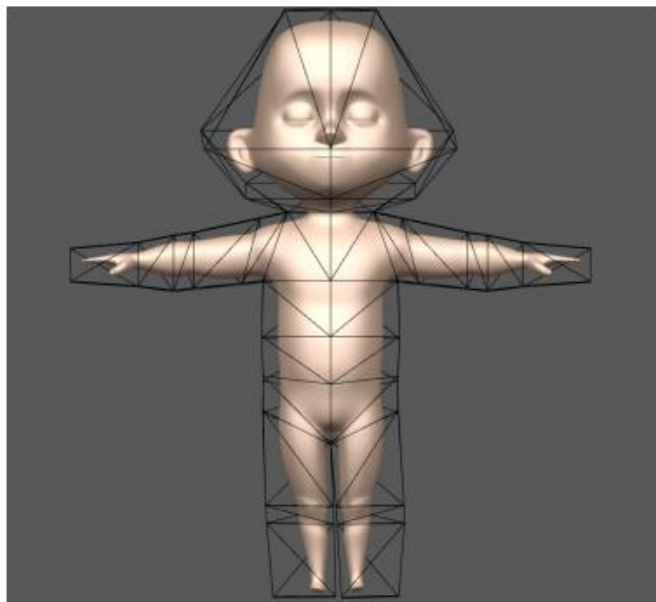
$h_i(\partial P)$: the (univariate) piecewise linear function such that $h_i(v_j) = \delta_{i,j}$.

- Non-negativity: harmonic functions achieve their extrema at their boundaries.
- Interior locality: follows from non-negativity and the fact that harmonic functions possess no interior extrema.

Numerical solution

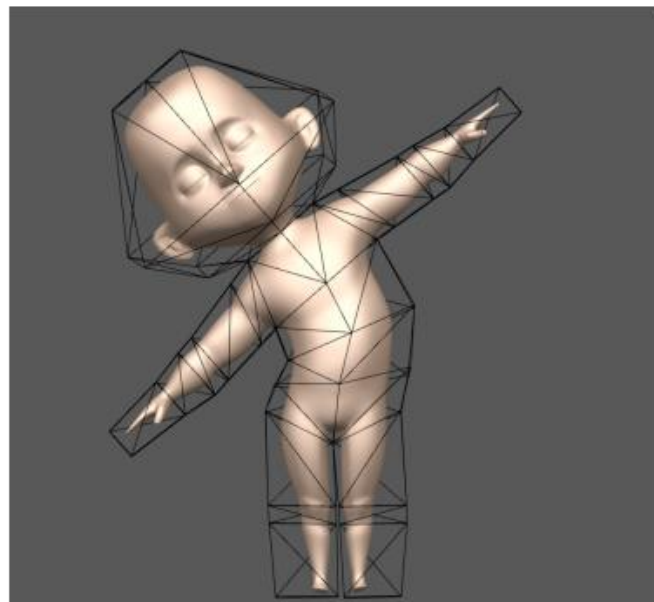
- 1. Allocate a regular grid of cells that is large enough to enclose the cage.
- 2. Laplacian smooth: For each INTERIOR cell, replace the value of the cell with the average of the value of its neighbors. This Laplacian smoothing step is performed iteratively until the termination criterion is reached.
- A simple hierarchical finite difference solver
 - By first solving the problem at a lower resolution, better starting points for the iteration can be obtained.

Bind



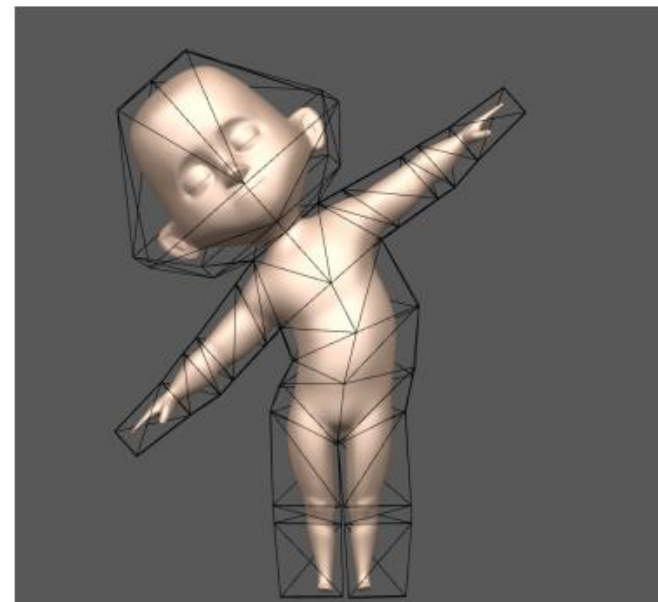
(a)

Mean Value

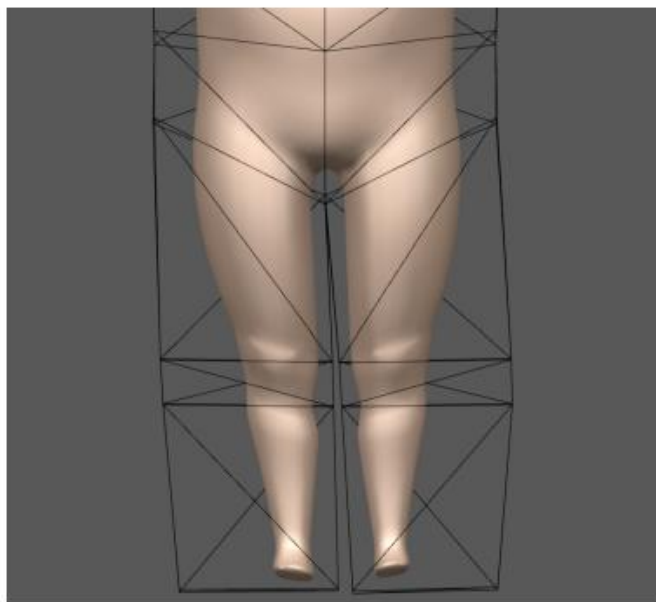


(b)

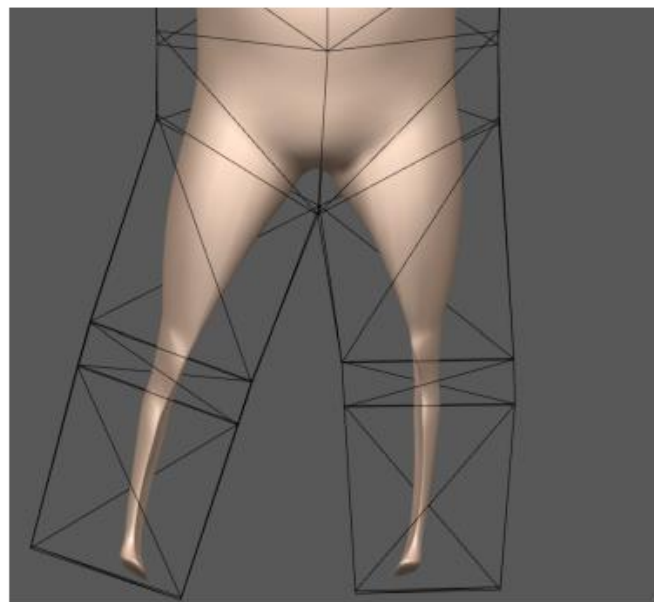
Harmonic



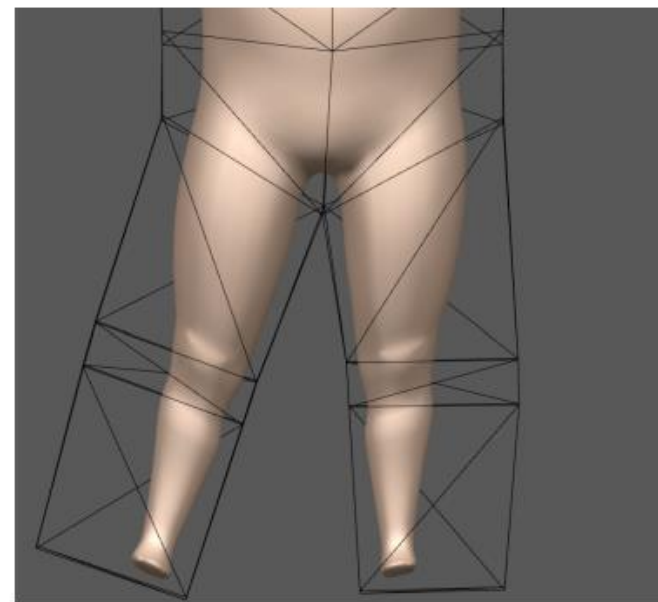
(c)



(d)



(e)



(f)

More papers

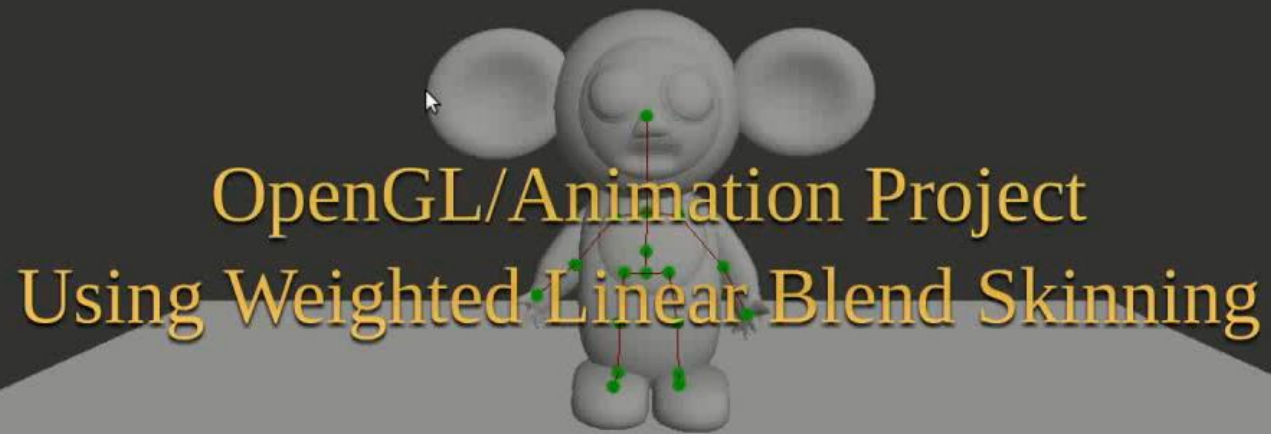
- Green Coordinates, 2008
- Complex Barycentric Coordinates with Applications to Planar Shape Deformation, 2009
- A Complex View of Barycentric Mappings, 2011
- Poisson Coordinates, 2013
- Cubic Mean Value Coordinates, 2013
-

Outlines

- Introduction
- Barycentric coordinates on convex polygons
- Inverse bilinear coordinates
- Mean value coordinates
- Harmonic Coordinates
- **A general construction**

Linear blend skinning

- Skeleton-subspace deformation



Linear blend skinning - input data

- Rest pose shape
 - Represented as a polygon mesh
 - The mesh connectivity is assumed to be constant, i.e., only vertex positions will change during deformations.
 - Rest-pose vertices: $v_1, \dots, v_n \in R^3$
- Bone transformations
 - A list of matrices
 - Spatial transformations aligning the rest pose of bone i with its current (animated) pose.
- **Skinning weights**
 - For vertex v_i , we have weights $w_{i,1}, \dots, w_{i,m} \in R$.
 - Each weight $w_{i,j}$ describes the amount of influence of **bone j on vertex i** .

Linear blend skinning - Bone transformations

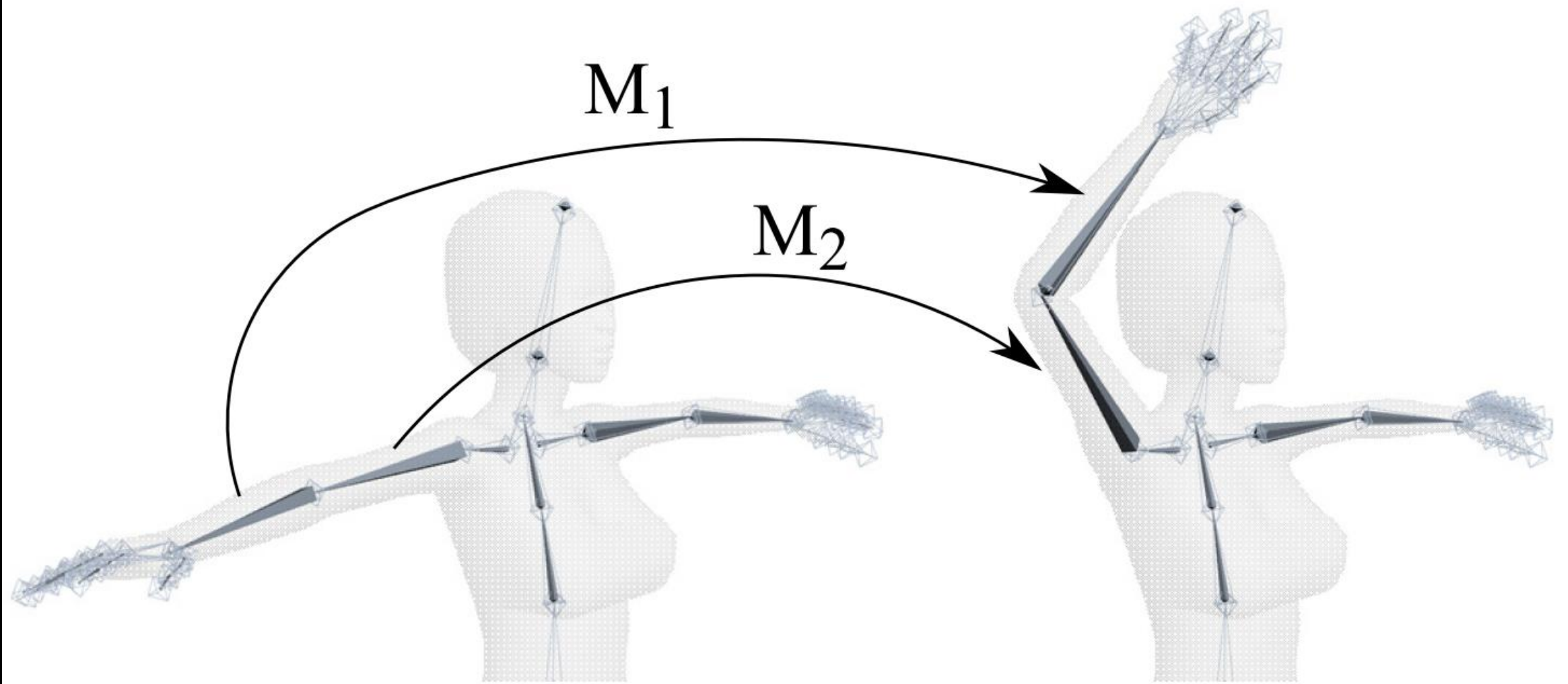


Figure 1: Bone transformations (lower and upper arm bones) for one example deformed pose.

Linear blend skinning - Skinning weights

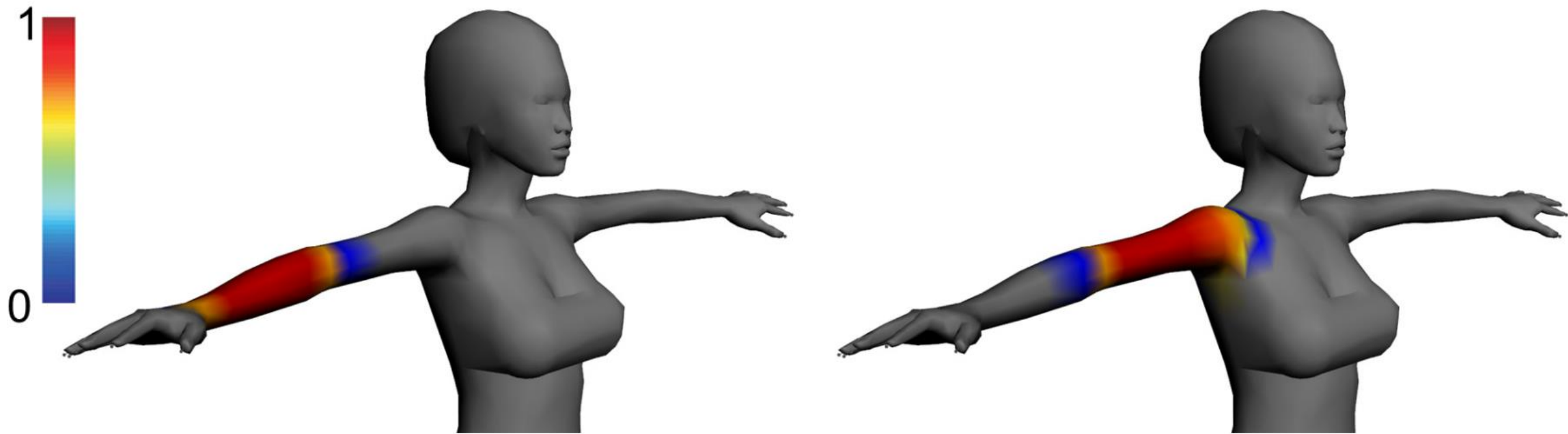


Figure 2: Influence weights corresponding to lower and upper arm bones.

Deformed vertex positions

$$v_i^{new} = \sum_{j=1}^m w_{i,j} T_j v_i = \left(\sum_{j=1}^m w_{i,j} T_j \right) v_i$$

The latter form highlights the fact that the rest pose vertex v_i is transformed by a linear combination (blend) of bone transformation matrices T_j .

Recap of properties

- Interpolation (**Lagrange property**)
- Smoothness
- Non-negativity ($\phi_i(x) \geq 0$)
- Interior locality
- Linear reproduction ($\sum_{i=1}^n \phi_i(x)v_i = x$)
- Affine-invariance = Partition of unity ($\sum_{i=1}^n \phi_i(x) = 1$)

Some papers

- Bounded Biharmonic Weights for Real-Time Deformation, 2011
- Local Barycentric Coordinates, 2014
- Linear Subspace Design for Real-Time Shape Deformation, 2015

Bounded Biharmonic Weights for Real-Time Deformation

- Real-time performance is critical for both interactive design and interactive animation.
- Among all deformation methods, linear blending and its variants dominate practical usage thanks to their speed
 - each point on the object is transformed by a **linear combination** of a small number of affine transformations.
- Real-time object deformations would be easier with support for all handle types: **points, skeletons, and cages**.
 - Goal: smooth and intuitive deformation

Various handles

Points
Bones
Cages

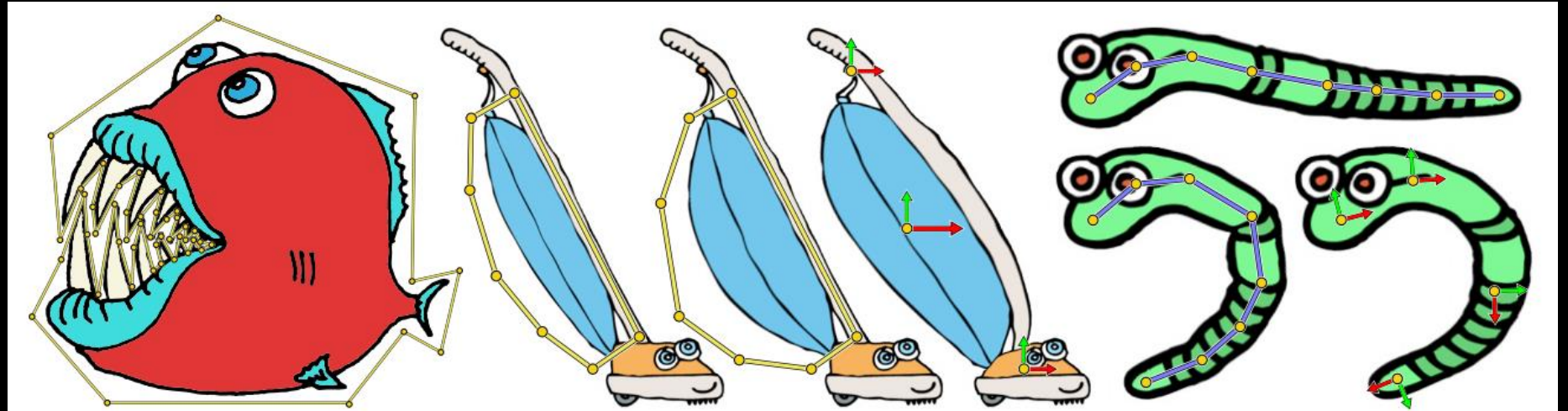
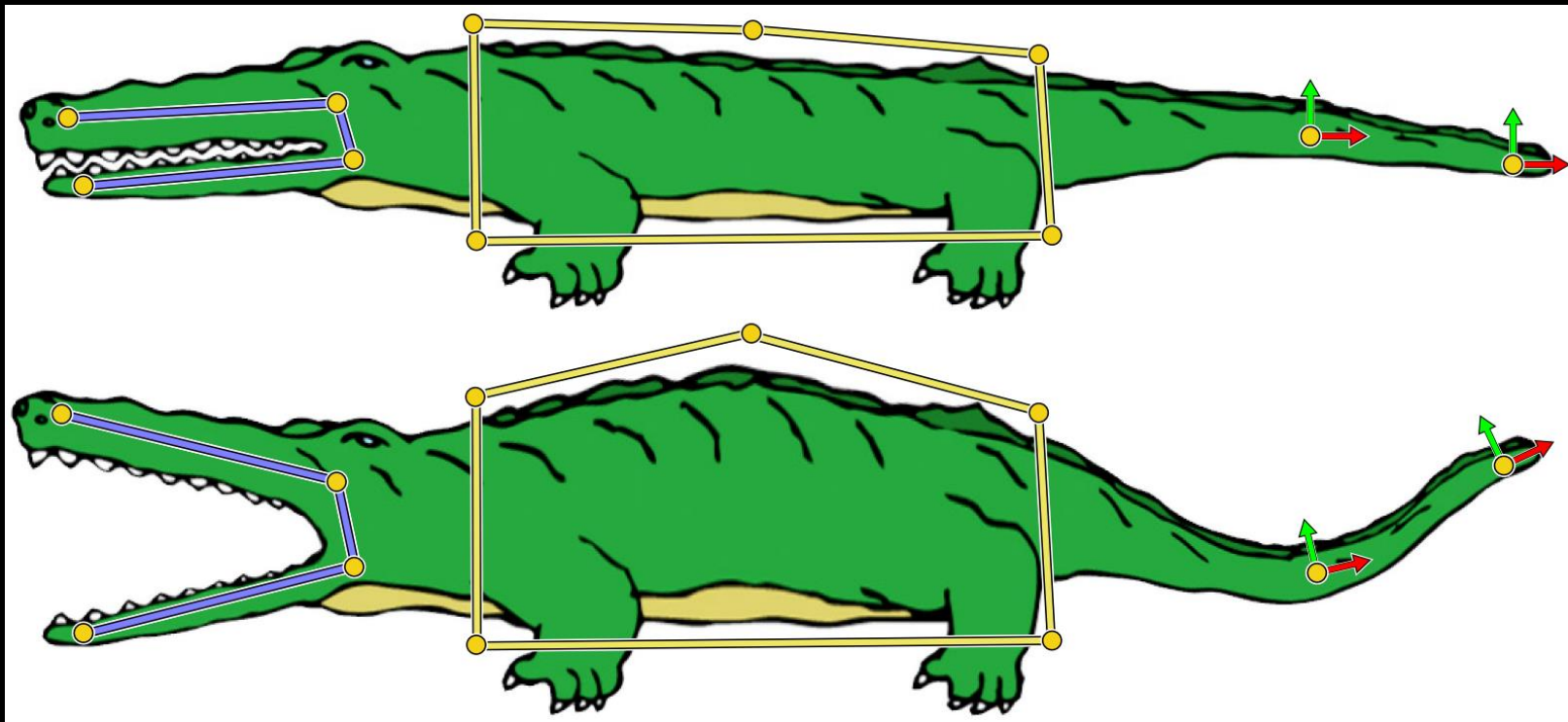


Figure 2: *Left to right: Although cages allow flexible control, setting up a closed cage can be both tedious and unintuitive: the Pi-ranha's jaws require weaving around the teeth. In the case of the Vacuum, points can provide crude scaling effects, while cages provide precise scaling articulation. Point handles can provide loose and smooth control, while achieving the same effect with a skeleton results in an overly complex armature.*

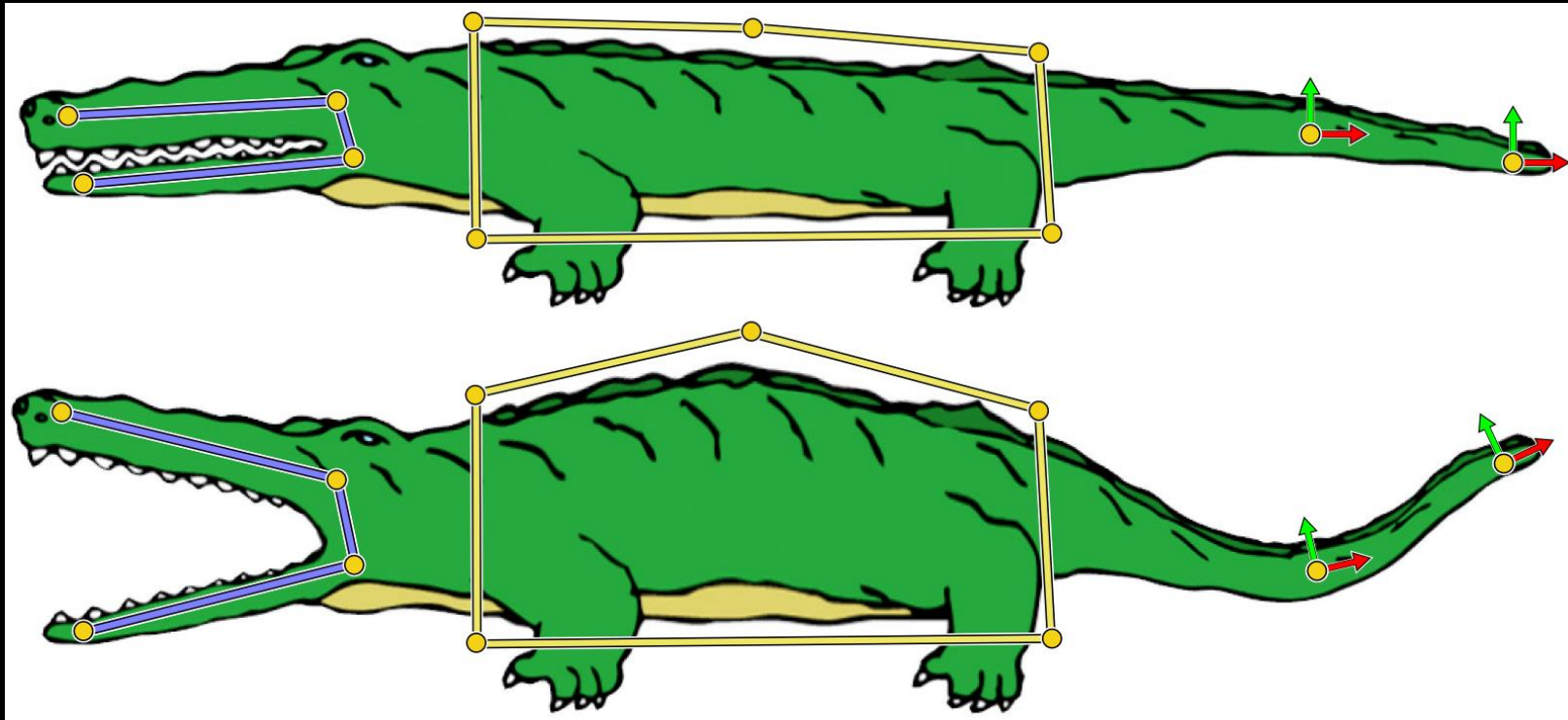
Handles - Points

- Points are quick to place and easy to manipulate.
- They specify local deformation properties (position, rotation and scaling) that smoothly propagate onto nearby areas of the object.



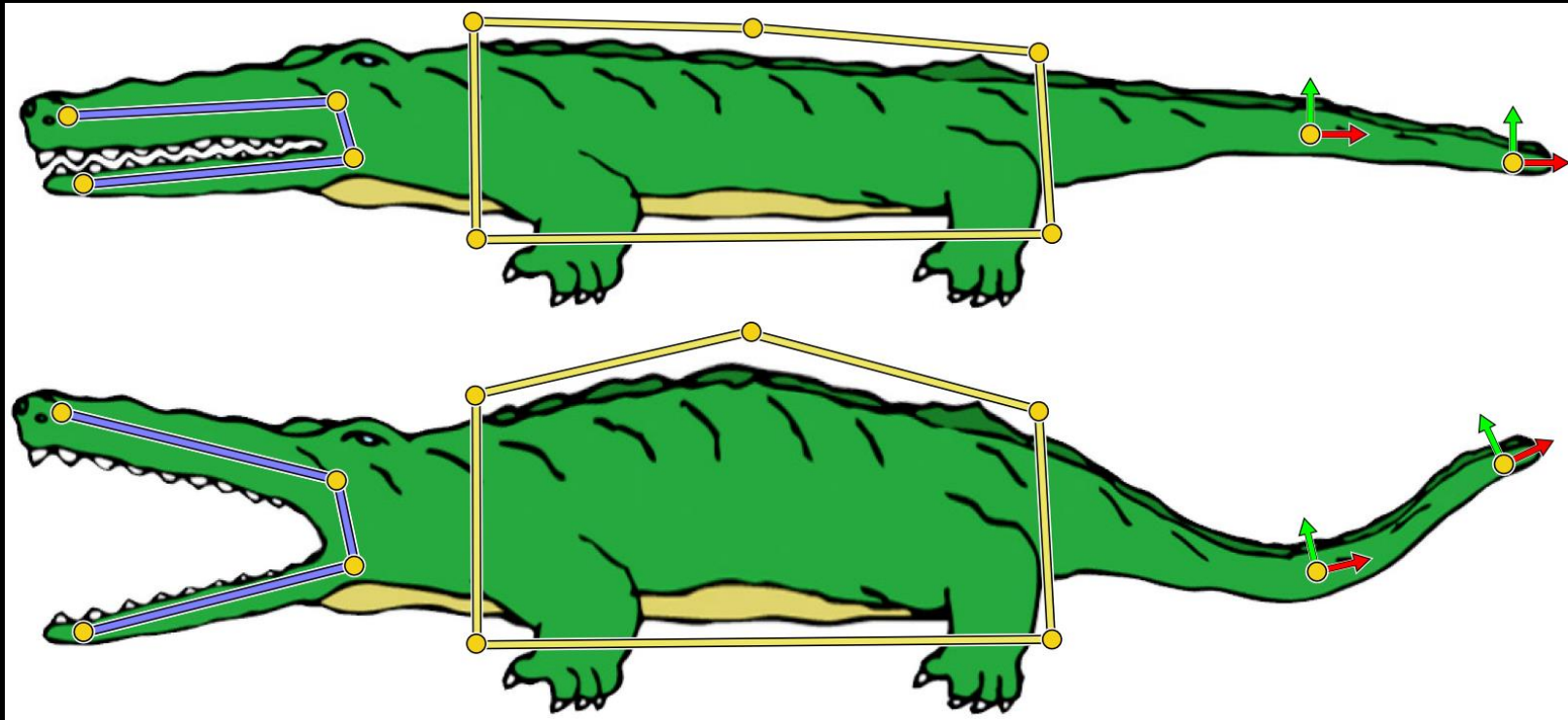
Handles - Bones

- Bones make some directions stiffer than others.
- If a region between two points appears too supple, bones can transform it into a rigid limb.



Handles - Cages

- Cages allow **influencing a significant portion** of the object at once, making it easier to control bulging and thinning in regions of interest.



Bounded biharmonic weights

$$\arg \min_{w_j, j=1, \dots, m} \sum_{j=1}^m \frac{1}{2} \int_{\Omega} \|\Delta w_j\|^2 dV \quad (2)$$

$$\text{subject to: } w_j|_{H_k} = \delta_{jk} \quad (3)$$

$$w_j|_F \text{ is linear} \quad \forall F \in \mathcal{F}_c \quad (4)$$

$$\sum_{j=1}^m w_j(\mathbf{p}) = 1 \quad \forall \mathbf{p} \in \Omega \quad (5)$$

$$0 \leq w_j(\mathbf{p}) \leq 1, \quad j = 1, \dots, m, \quad \forall \mathbf{p} \in \Omega, \quad (6)$$

Properties

- Smoothness ($\Delta^2 w_j = 0$)
 - The bounded biharmonic weights are C^1 at the handles and C^∞ everywhere else, provided that the posed boundary conditions are smooth.
- Non-negativity
- Shape-awareness: bi-Laplacian operator
- Partition of unity
- Locality and sparsity: just observation
- No local maxima: experimentally observed
- No Linear reproduction ($\sum_{i=1}^n \phi_i(x) v_i = x$)

Properties

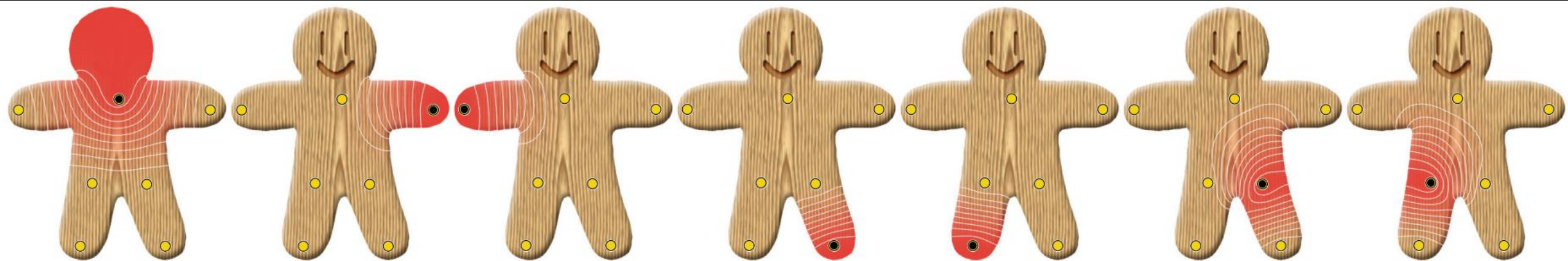


Figure 3: *Bounded biharmonic weights are smooth and local: the blending weight intensity for each handle is shown in red with white isolines. Each handle has the maximum effect on its immediate region and its influence disappears in distant parts of the object.*

Properties

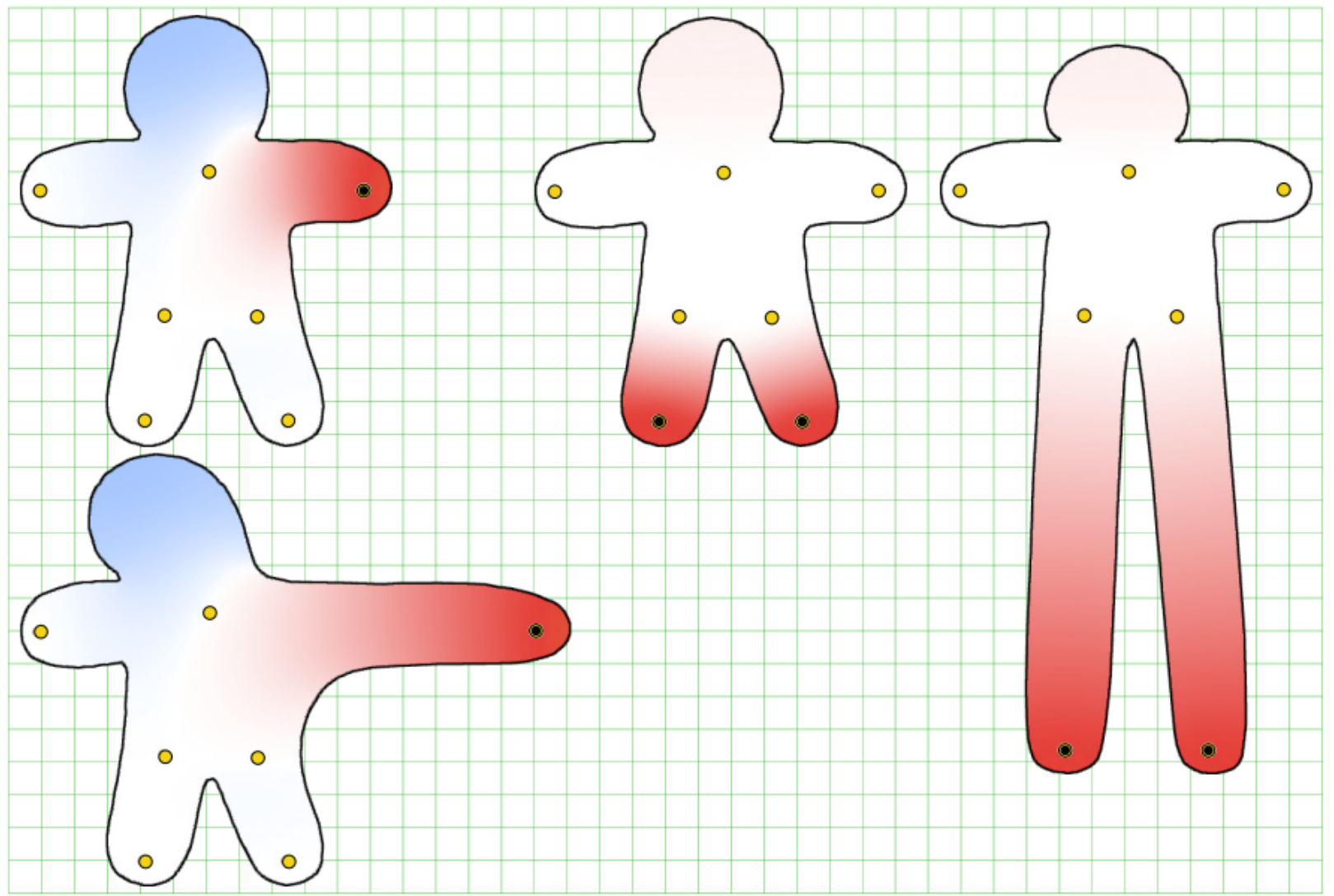


Figure 4: *Weights like unconstrained biharmonic functions that have negative weights (left) and extraneous local maxima (right) lead to undesirable and unintuitive behavior. Notice the shrinking of the head on the right.*

Bounded Biharmonic Weights for Real-Time Deformation

Alec Jacobson¹

Ilya Baran²

Jovan Popović³

Olga Sorkine^{1,4}

¹New York University

²Disney Research, Zurich

³Adobe Systems, Inc.

⁴ETH Zurich

This video contains narration

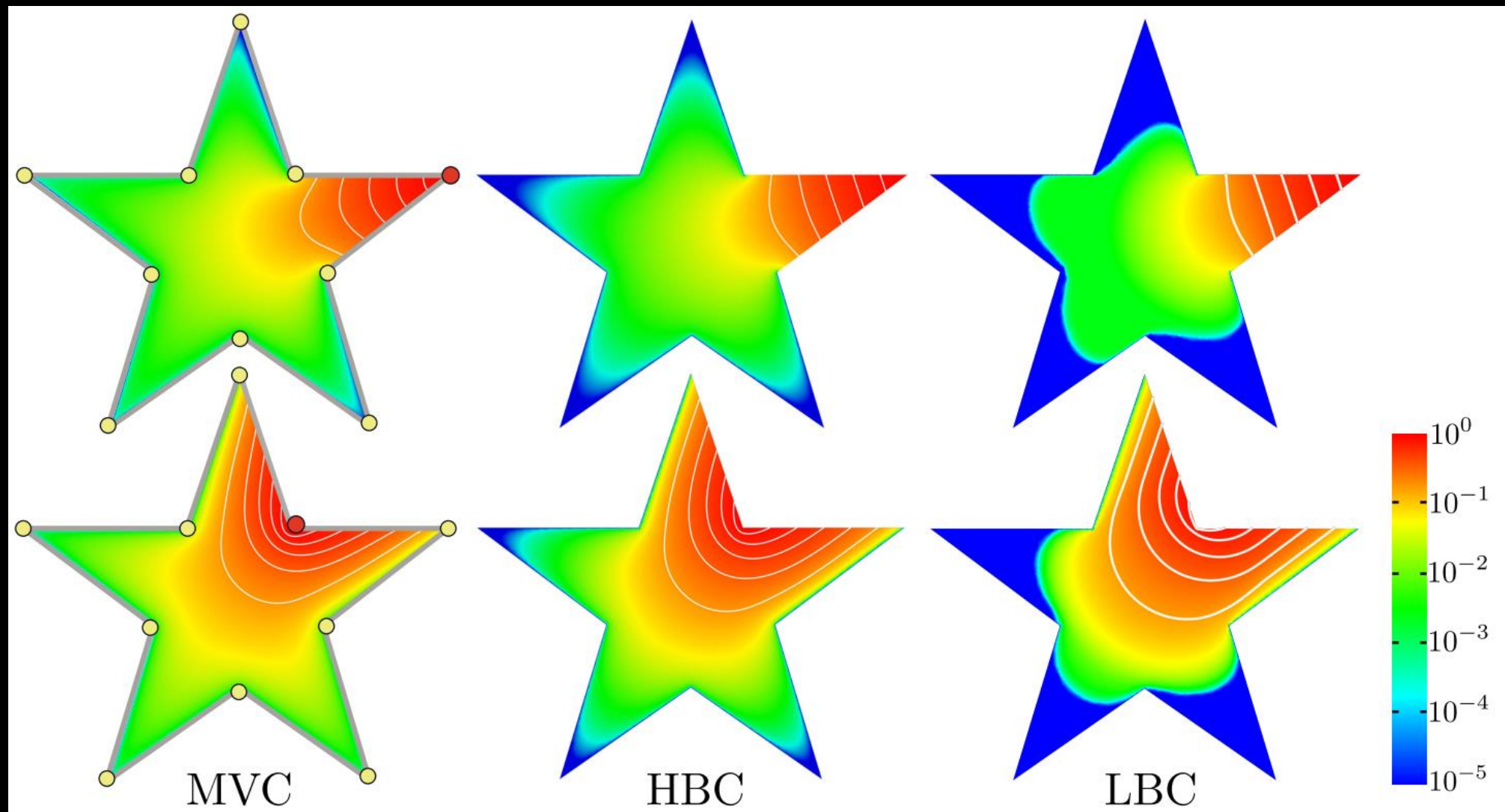
Local Barycentric Coordinates

- A local change in the value at a single control point will create a global change by propagation into the whole domain.
- Global nature
 - The first one is the lack of locality and control over a deformation.
 - The second drawback is scalability.
 - Most practical applications store barycentric coordinates using one scalar value per control point for every vertex of the target domain.

Formulation

$$\begin{aligned} \min_{w_1, \dots, w_n} \quad & \sum_{i=1}^n \int_{\Omega} |\nabla w_i| \\ \text{s.t.} \quad & \sum_{i=1}^n w_i(\mathbf{x}) \mathbf{c}_i = \mathbf{x}, \quad \sum_{i=1}^n w_i = 1, \quad w_i \geq 0, \quad \forall \mathbf{x} \in \Omega, \\ & w_i(\mathbf{c}_j) = \delta_{ij} \quad \forall i, j, \\ & w_i \text{ is linear on cage edges and faces } \forall i. \end{aligned} \tag{5}$$

Locality



Local extrema

- TV measures oscillation, and hence its minimization inhibits local extremal values.

Demo

Linear Subspace Design for Real-Time Shape Deformation

- Linear reproduction
 - Cot weights of Laplacian satisfy.
 - ?

Demo

Simplification

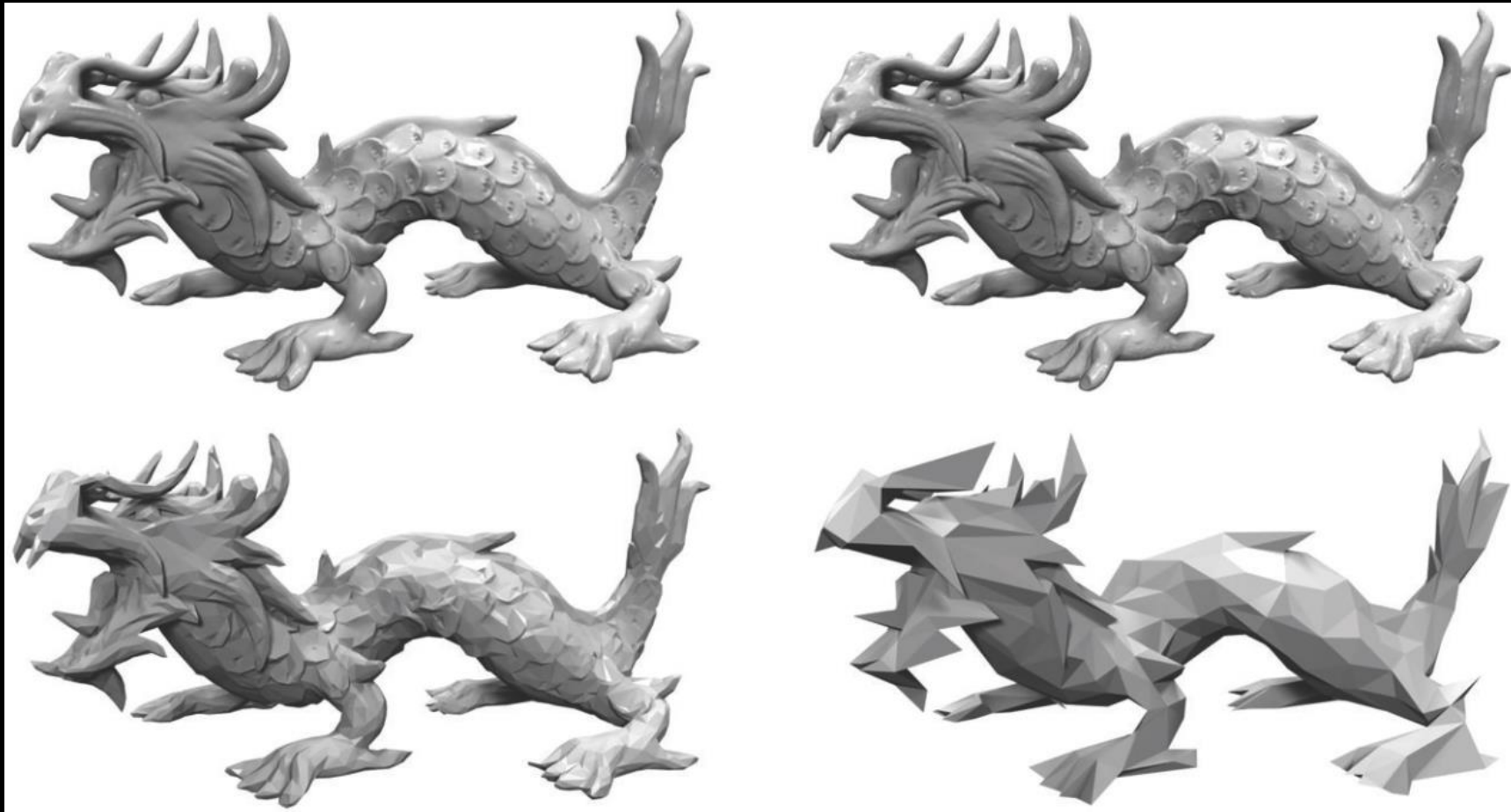
Xiao-Ming Fu

Outlines

- Definition
- Local operations
- Quadric error metric
- Variational shape approximation

Simplification and approximation

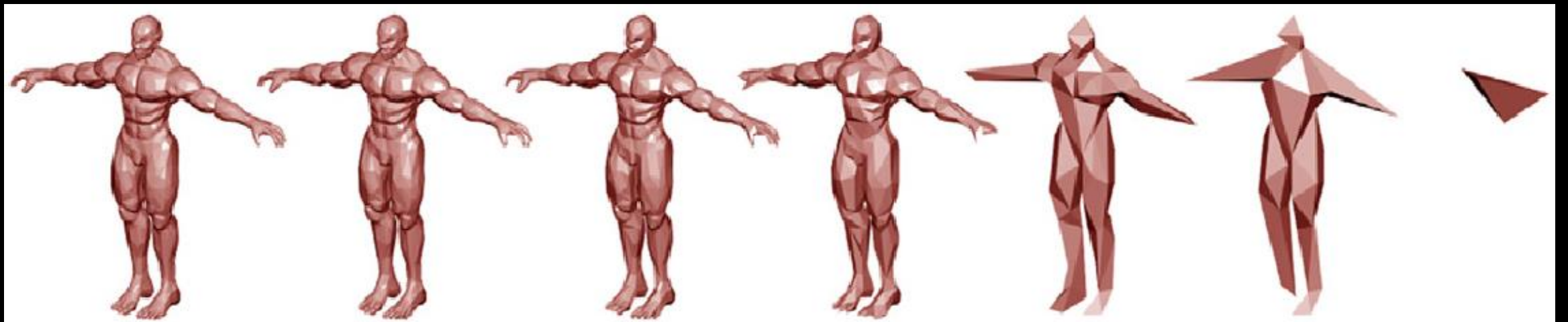
- Transform a given polygonal mesh into another mesh with **fewer** faces, edges, and vertices.



Simplification and approximation

- Transform a given polygonal mesh into another mesh with **fewer** faces, edges, and vertices.
- The simplification or approximation procedure is usually controlled by user-defined **quality criteria**.

Curvature-preserved criteria



2053

1500

1000

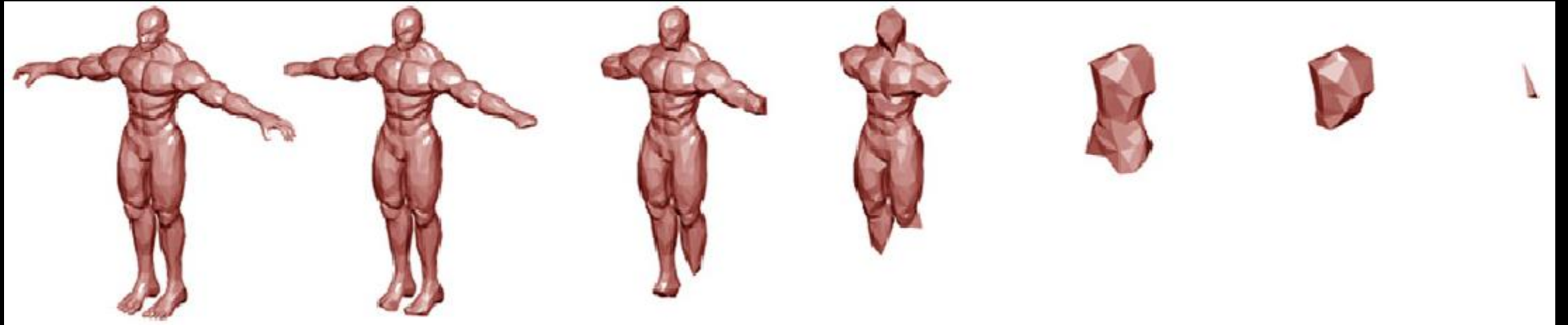
500

100

50

4

Curvature-removed criteria



2053

1500

1000

500

100

50

4

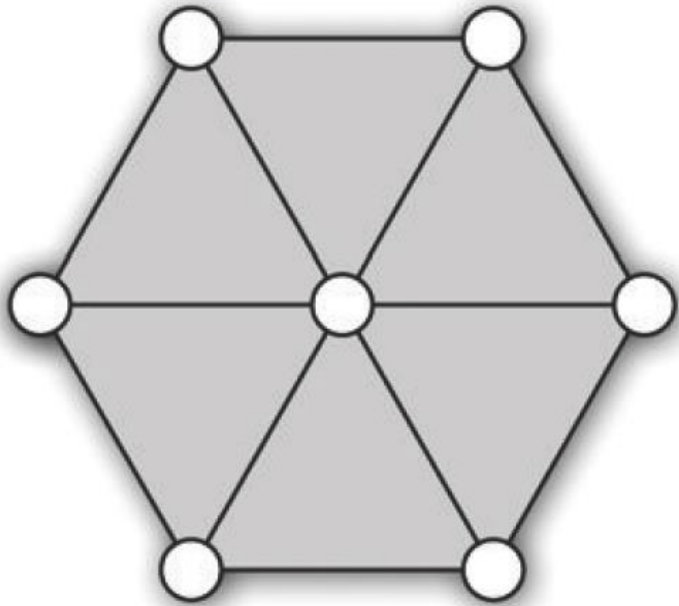
Simplification applications

- Adjust the complexity of a geometric data set
- Since many decimation schemes work iteratively, i.e., they decimate a mesh by removing one vertex at a time, they usually can be **inverted**.
 - Hierarchical method

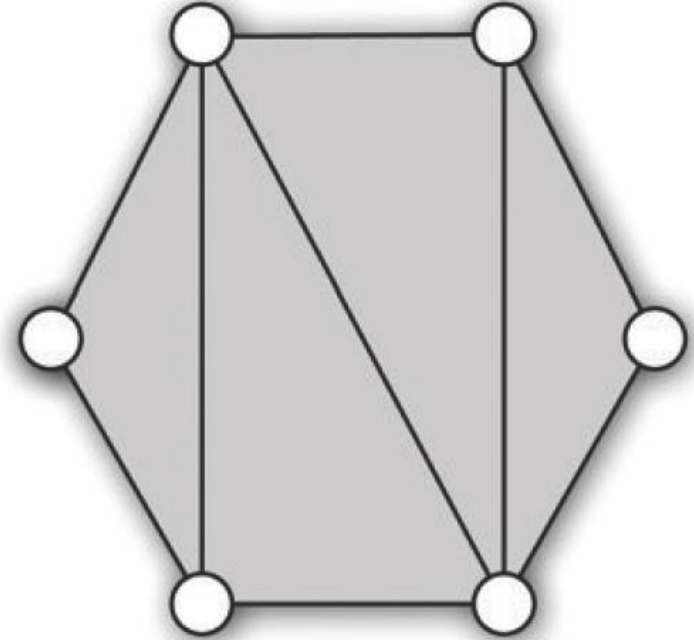
Outlines

- Definition
- **Local operations**
- Quadric error metric
- Variational shape approximation

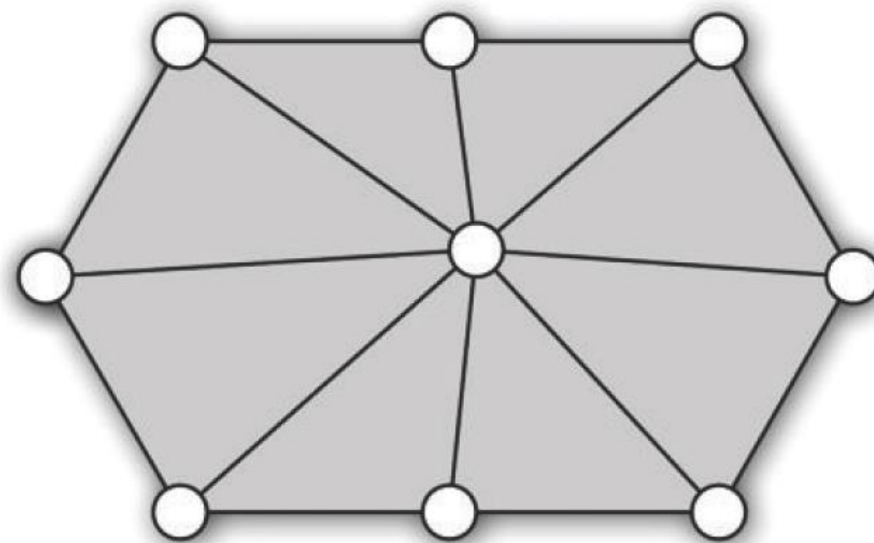
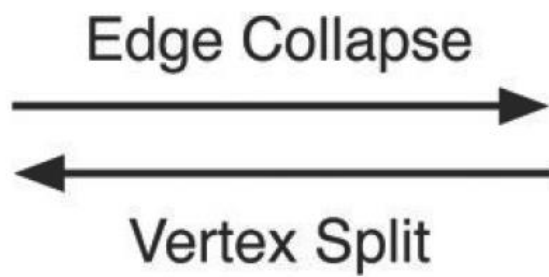
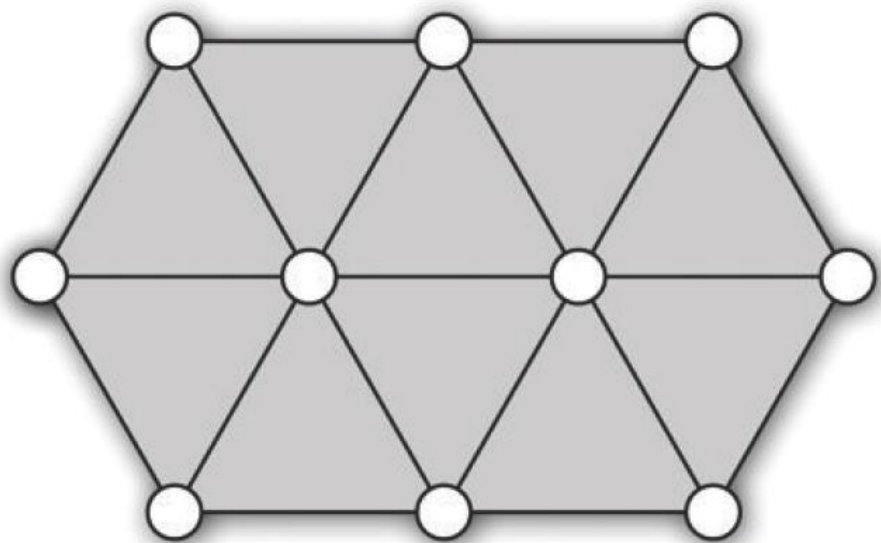
Vertex removal



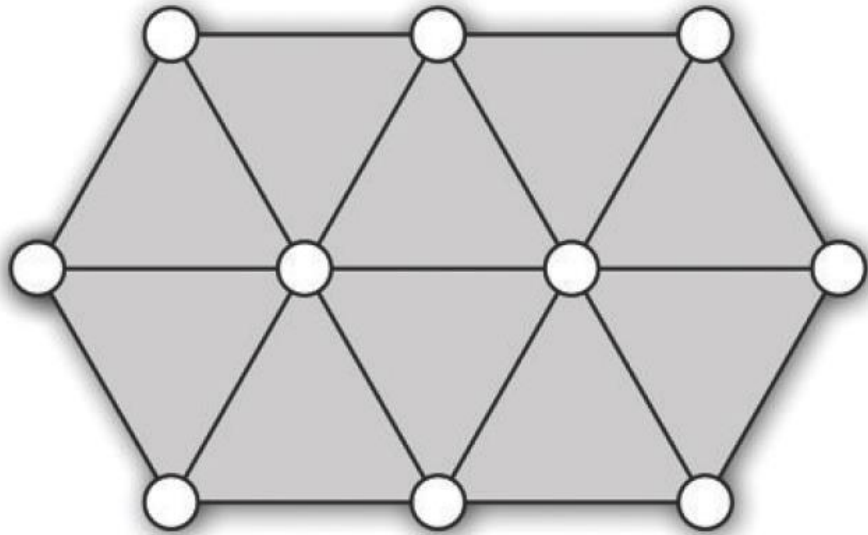
Vertex Removal
→
←
Vertex Insertion



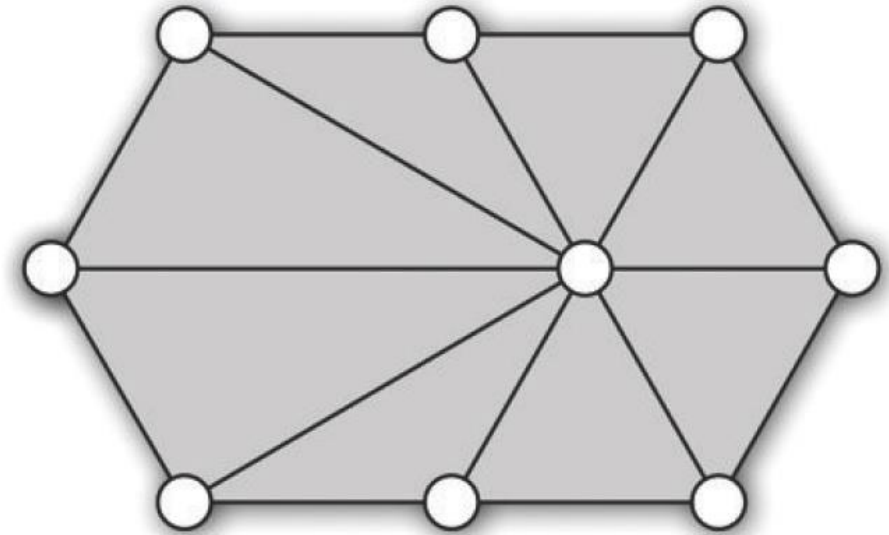
Edge collapse



Half-edge collapse



Halfedge Collapse
→
←
Restricted Vertex Split



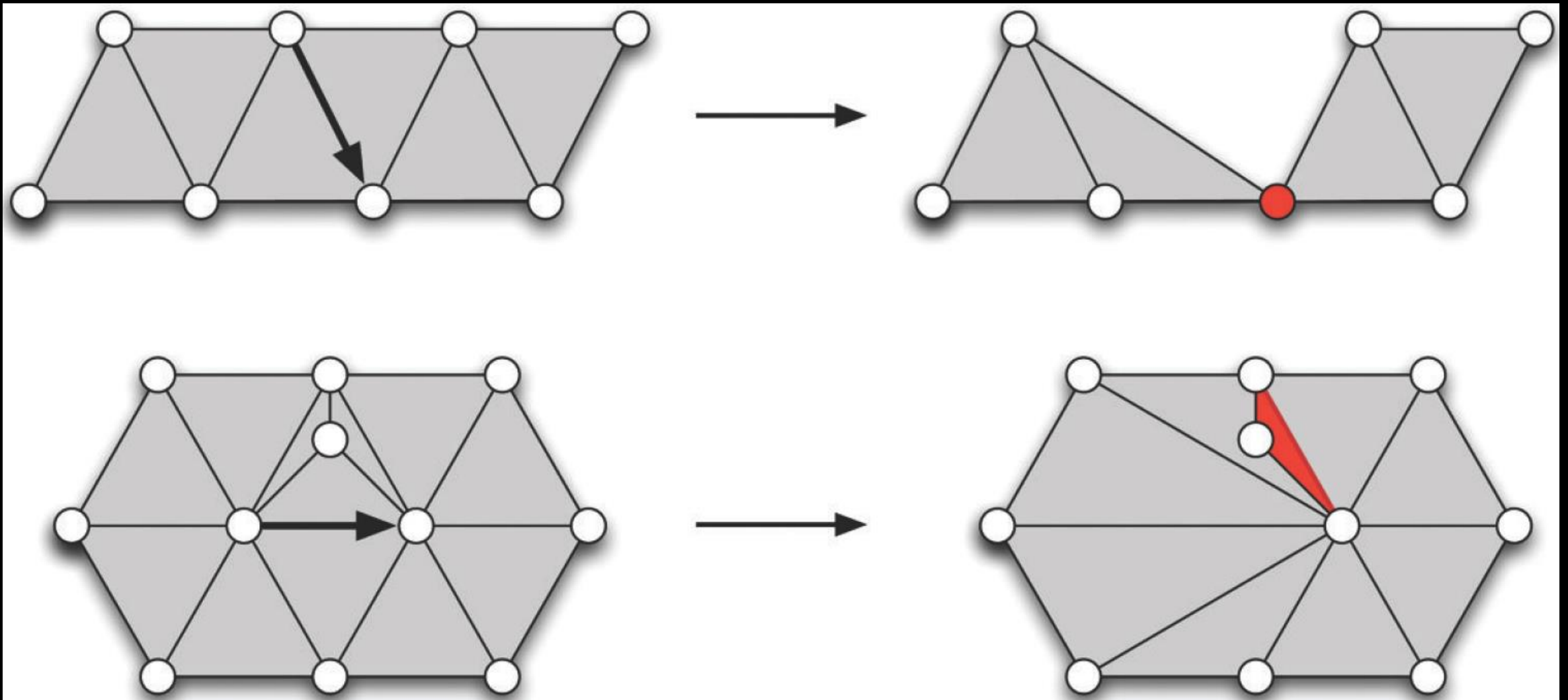
After collapse: $n(E) - 3$, $n(V) - 1$, $n(F) - 2$.

According to Euler's formula: $2 - 2m = n(V) + n(F) - n(E)$.

Half-edge collapsing would not change the genus of a mesh.

OpenMesh: `collapse()`, `is_collapse_ok()`.

Topologically illegal (half-)edge collapses



Outlines

- Definition
- Local operations
- **Quadratic error metric**
- Variational shape approximation

Incremental algorithms

- Removing one vertex at a time
- The iterative decimation procedure can take **arbitrary user-defined criteria** into account, according to **which the next removal operation is chosen**.

Quadratic error metric (QEM)

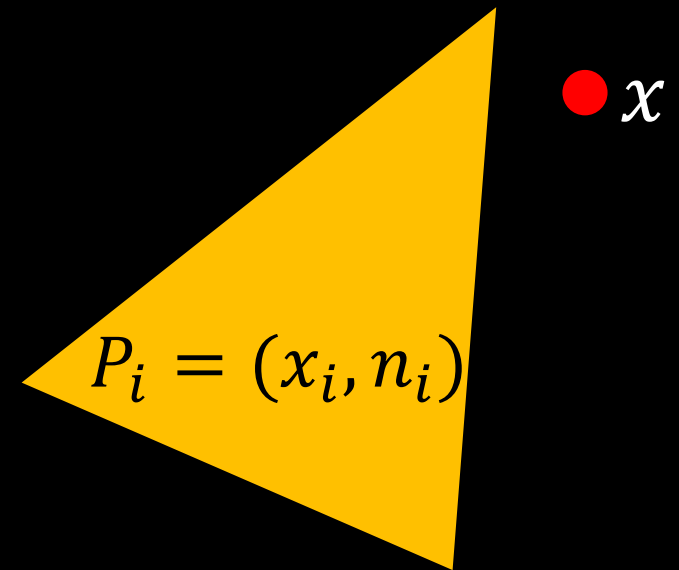
- The squared distance of a point x from the plane P_i :

$$d(x, P_i) = (n_i^T x - d_i)^2$$
$$d_i = n_i^T x_i$$

Denote $\bar{x} = (x, 1)$ and $\bar{n}_i = (n_i, -d_i)$.

Then:

$$d(x, P_i) = (\bar{n}_i^T \bar{x})^2 = \bar{x}^T \bar{n}_i \bar{n}_i^T \bar{x} =: \bar{x}^T Q_i \bar{x}$$

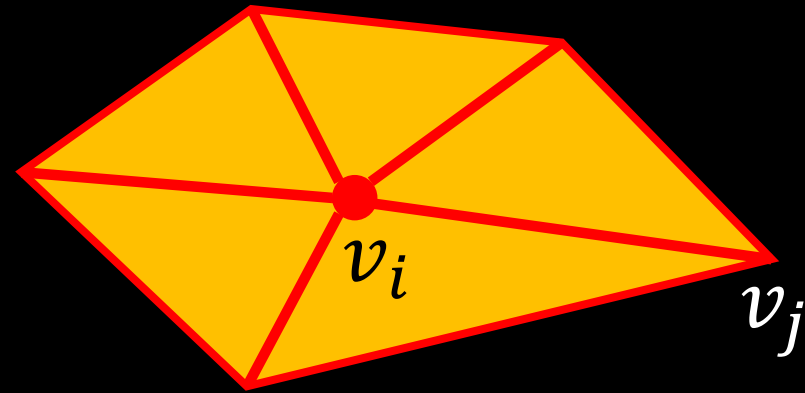


Quadratic error Matrix

Quadratic error Matrix Q

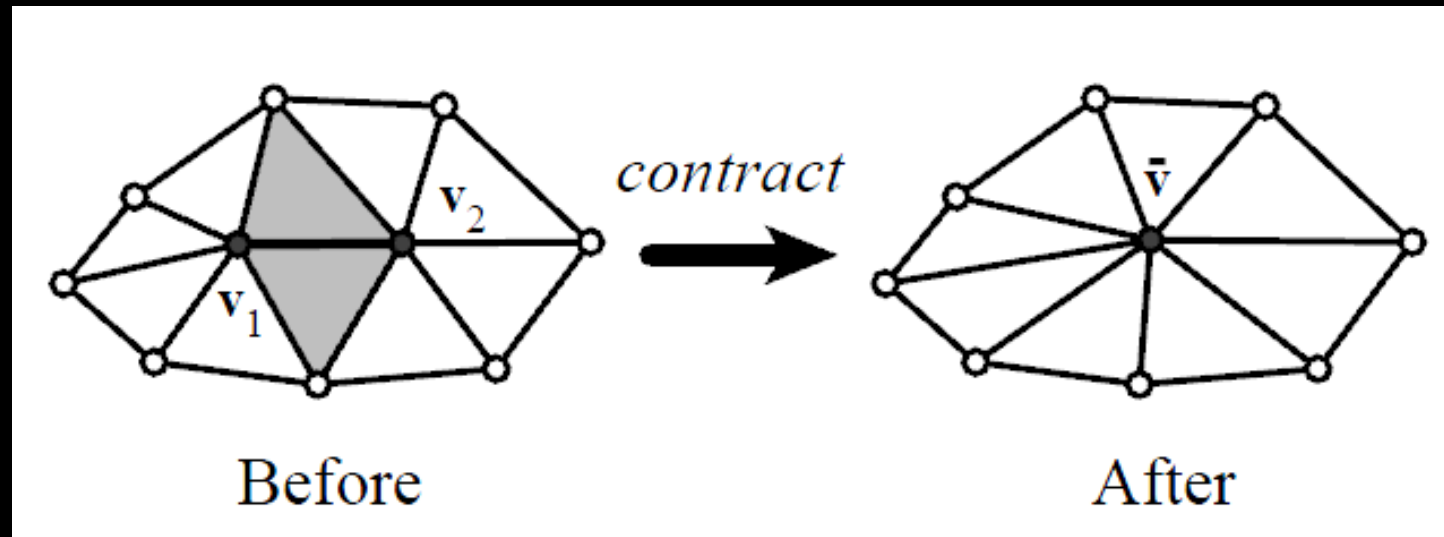
- On vertices

$$Q_i^v = \sum_{j \in \Omega(i)} Q_j$$



- On edge

$$Q^e = Q_1^v + Q_2^v$$



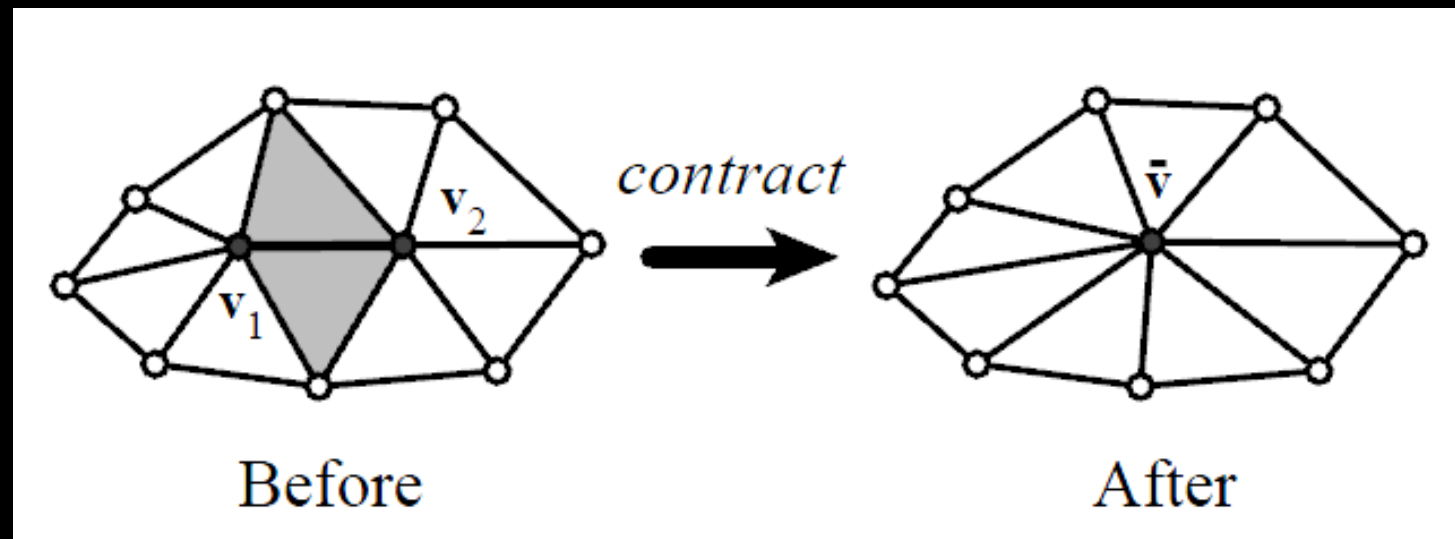
QEM error

- QEM error On edge:

$$\bar{v} = \arg \min_v v^T Q^e v$$

Note: Q^e may not a full rank matrix

- Q on \bar{v} is just Q^e .



QEM Algorithm

Input: a mesh

Output: a simplified mesh

Initialization:

Compute the Q^e matrices for all the edges.

Compute the optimal contraction target \bar{v} for each edge.

While $N_v > n$ and $Cost_{min} < t$

The error $\bar{v}^T Q^e \bar{v}$ becomes the **cost** of the edge.

Place all the edges in a priority queue keyed on cost with minimum **cost** edge at the top.

Remove the edge of the least cost from the heap, collapse this edge, and update the costs of all edges involving.

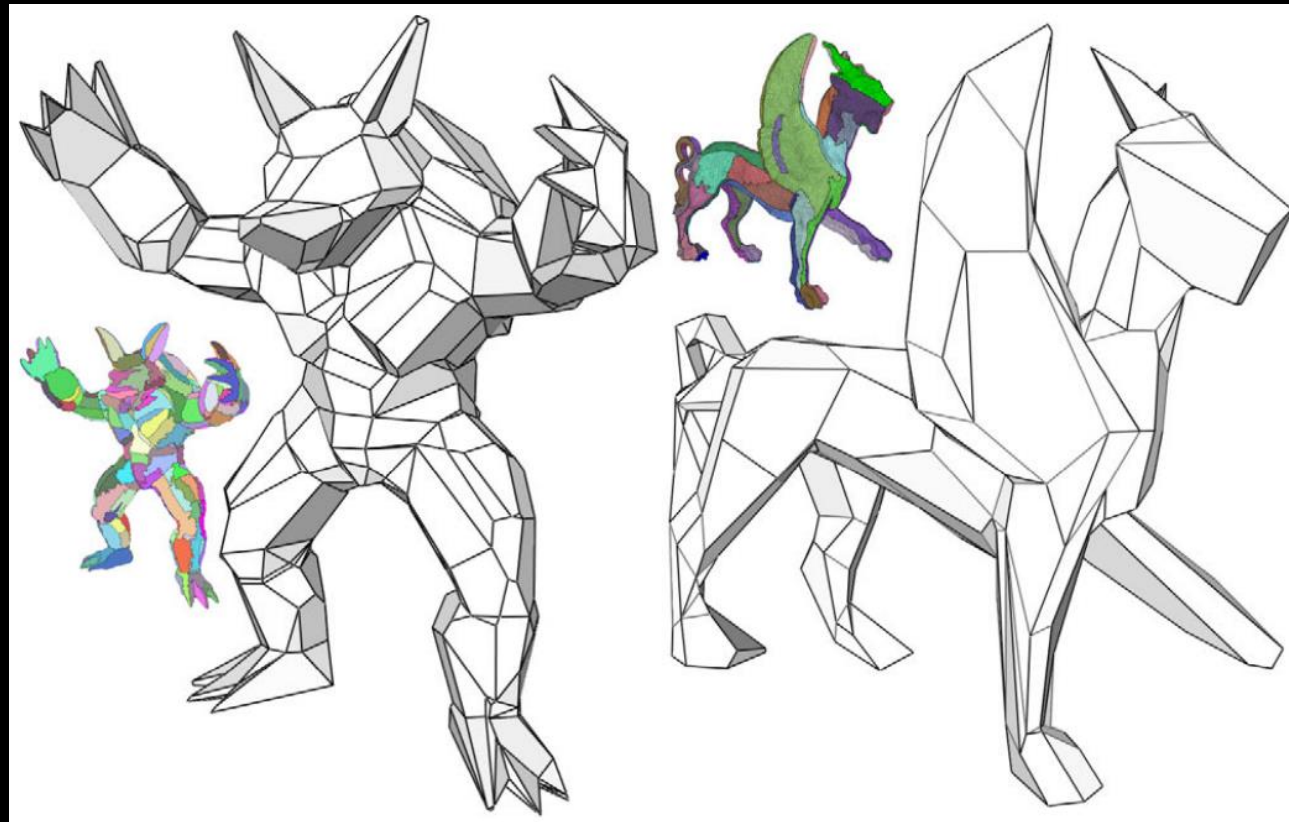
End;

Outlines

- Definition
- Local operations
- Quadric error metric
- **Variational shape approximation**

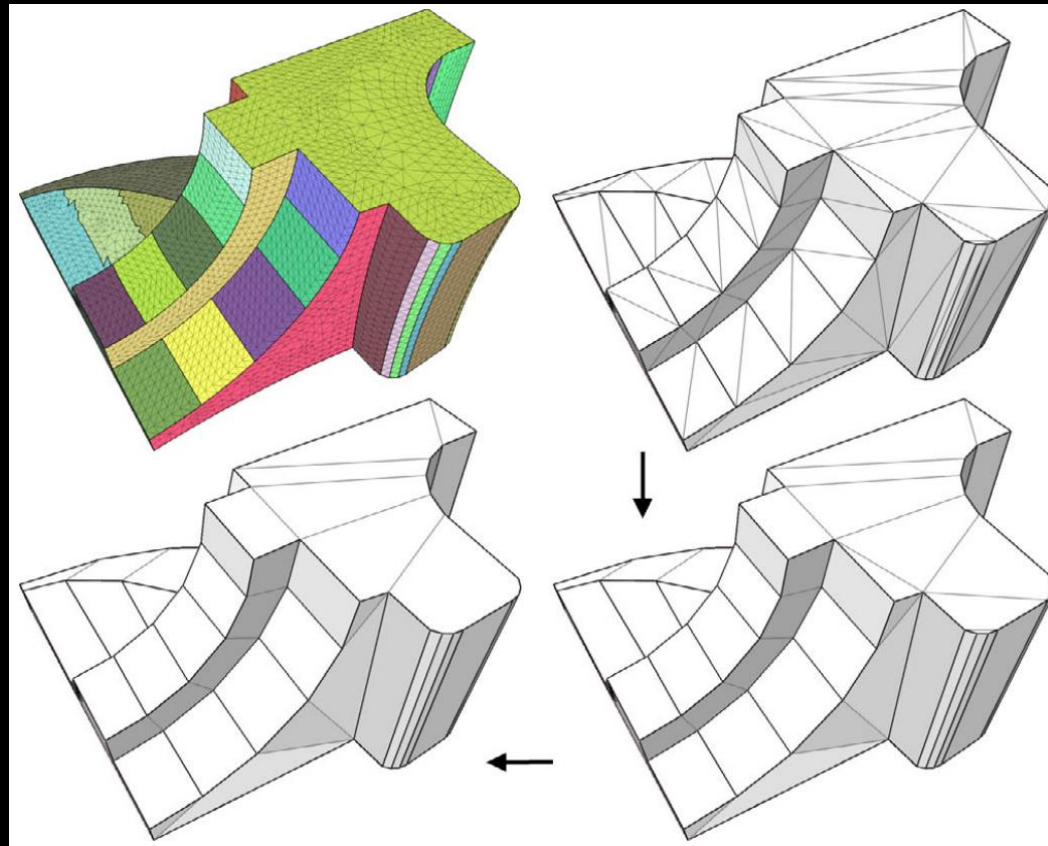
Variational shape approximation (VSA)

- VSA is highly sensitive to features and symmetries and produces anisotropic meshes of high approximation quality.



Variational shape approximation (VSA)

- The input shape is approximated by a set of proxies.
 - A plane in space through the point x_i with normal direction n_i .



Region representation

$$R_1 \cup \dots \cup R_k = M$$

M : a triangle mesh

$R = \{R_1, \dots, R_k\}$: a partition of M into k regions.

Proxies: $P = \{P_1, \dots, P_k\}$, $P_i = (x_i, n_i)$

Distance metrics between R_i and P_i

- The squared orthogonal distance of x from the plane P_i .

$$L^2(R_i, P_i) = \int_{x \in R_i} (n_i^T x - n_i x_i)^2 dA$$

- A measure of the normal field:

$$L^{2,1}(R_i, P_i) = \int_{x \in R_i} \|n(x) - n_i\|^2 dA$$

Goal of VSA

- Given a number k and an error metric E (L^2 or $L^{2,1}$), find a set $R = \{R_1, \dots, R_k\}$ of regions and a set $P = \{P_1, \dots, P_k\}$ of proxies such that the global distortion

$$E(R, P) = \sum_{i=1}^k E(R_i, P_i)$$

is minimized.

Lloyd's clustering algorithm

- The algorithm iteratively alternates between a **geometry partitioning** phase and a **proxy fitting** phase.
- Geometry partitioning phase
 - a set of regions that best fit a **given** set of proxies
- Proxy fitting phase
 - the partitioning is kept **fixed** and the proxies are adjusted
- Initialization
 - randomly picking k triangles as R
 - The planes of k triangles are used to initialize P

Geometry partitioning phase

- **Modifies** the set R of regions to achieve a **lower approximation error** while keeping the proxies P **fixed**.

partition($\mathcal{R} = \{\mathcal{R}_1, \dots, \mathcal{R}_k\}, \mathcal{P} = \{P_1, \dots, P_k\}$)

// find the seed triangles and initialize the priority queue

queue = \emptyset

for $i = 1$ to k do

 select the triangle $t \in \mathcal{R}_i$ that minimizes $E(t, P_i)$

$\mathcal{R}_i = \{t\}$

 set t to conquered

 for all neighbors r of t do

 insert (r, P_i) into queue

// grow the regions

while the queue is not empty do

 get (t, P_i) from the queue that minimizes $E(t, P_i)$

 if t is not conquered then

 set t to conquered

$\mathcal{R}_i = \mathcal{R}_i \cup \{t\}$

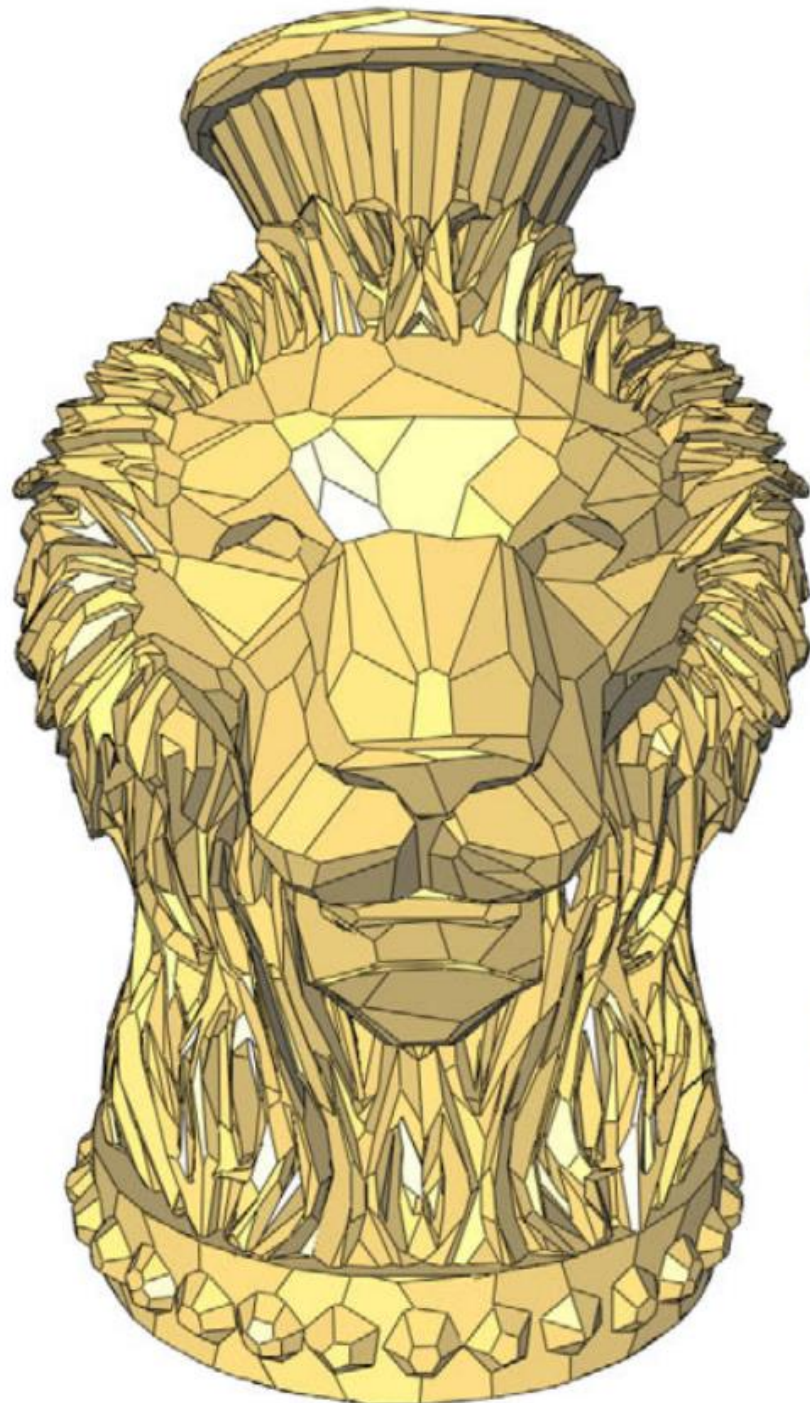
 for all neighbors r of t do

 if r is not conquered then

 insert (r, P_i) into queue

Proxy fitting phase

- The partition R is kept fixed, the proxies P_i are adjusted in order to minimize approximation error.
- L^2 metric
 - The best proxy is the least-squares fitting plane.
- $L^{2,1}$ metric
 - The proxy normal n_i is just the area-weighted average of the triangle normals.



Deformation

Xiao-Ming Fu

Outline

- Definition
- Transformation Propagation
- Multi-Scale Deformation
- Differential Coordinates
- Deformation transfer
- As-Rigid-As-Possible surface deformation
- Freeform Deformation
 - Meshless mapping
- Volumetric Deformation
 - Tetrahedral mapping

Outline

- **Definition**
- Transformation Propagation
- Multi-Scale Deformation
- Differential Coordinates
- Deformation transfer
- As-Rigid-As-Possible surface deformation
- Freeform Deformation
 - Meshless mapping
- Volumetric Deformation
 - Tetrahedral mapping

Definition

- The deformation of a given surface S into the desired surface S'
 - a displacement $d(p)$ on each vertex $p \in S$
 - $S' = \{p + d(p) | p \in S\}$
- The user controls the deformation by
 - prescribing displacements \bar{d}_i for a set of vertices $p_i \in H \subset S$.
 - constraining certain parts F stay fixed.
 - **handles**
- **The main question: determine the displacements for vertices in $S \setminus (H \cup F)$.**

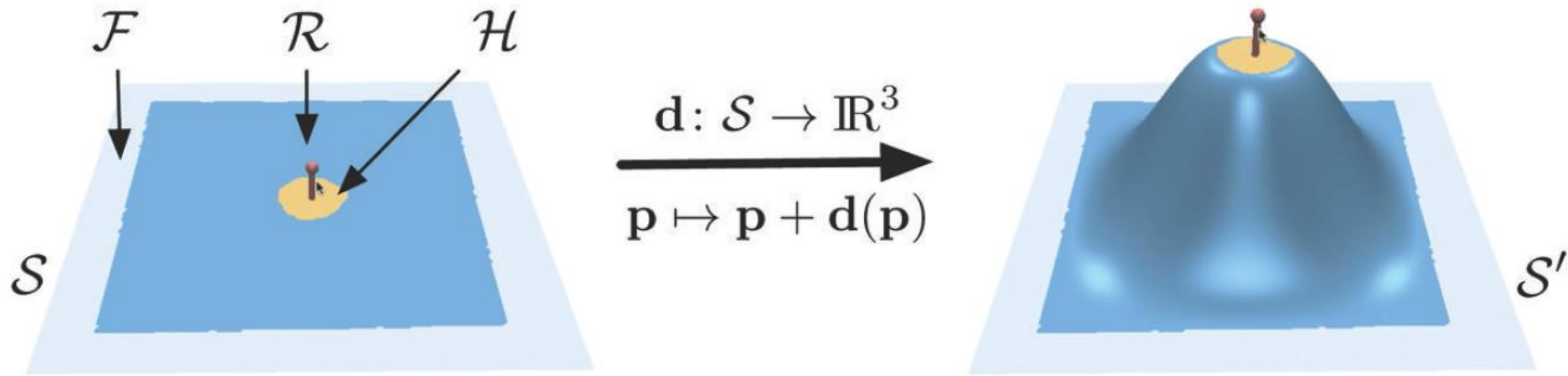


Figure 9.1. A given surface \mathcal{S} is deformed into \mathcal{S}' by a displacement function $\mathbf{d}(\mathbf{p})$. The user controls the deformation by moving a handle region \mathcal{H} (yellow) and keeping the region \mathcal{F} (gray) fixed. The unconstrained deformation region \mathcal{R} (blue) should deform in an intuitive, physically-plausible manner.

Two classes of shape deformations

- Surface-based deformations
 - The displacement is defined on each vertex
 - A high degree of control, since each vertex can be constrained individually.
 - The robustness and efficiency of the involved computations are strongly affected by the mesh complexity and the triangle quality of the original surface S .
- Space deformations
 - Displacement is defined on each point in the space.
 - Smooth.

Outline

- Definition
- Transformation Propagation
- Multi-Scale Deformation
- Differential Coordinates
- Deformation transfer
- As-Rigid-As-Possible surface deformation
- Freeform Deformation
 - Meshless mapping
- Volumetric Deformation
 - Tetrahedral mapping

Transformation Propagation

- Propagating the user-defined handle transformation:
 - ✓ 1. specify the support region of the deformation
 - ✓ 2. specify a handle region within the support region
 - ✓ 3. the handle is transformed using some modeling interface
 - ✓ 4. propagate the transformation of handle and damp within the support region
 - ✓ leading to a **smooth blending** between the transformed handle and the fixed region



Figure 9.2. After specifying the blue support region and the green handle region (left), a smooth scalar field is constructed that is 1 at the handle and 0 outside the support (center). Its isolines are visualized in black and red, where red is the $\frac{1}{2}$ -isoline. This scalar field is used to propagate and damp the handle's transformation within the support region (right). (Image taken from [Botsch et al. 06b]. ©2006 ACM, Inc. Included here by permission.)

Smooth blend

- Controlled by a scalar field:
 - 1 is at the handle;
 - 0 is at the fixed region;
 - smoothly blends between 1 and 0 within the support region.
- One method:
 - $d_F(p)$: **distance** from p to the fixed region
 - $d_H(p)$: **distance** from p to the handle

$$s(p) = \frac{d_F(p)}{d_F(p) + d_H(p)}$$

Harmonic field

$$\Delta s(p_i) = 0 \quad \forall p_i \in R$$

$$s(p_i) = 1 \quad \forall p_i \in H$$

$$s(p_i) = 0 \quad \forall p_i \in F$$

Simple to implement

Discussion

- simple and efficient to compute
- distance-based propagation of transformations will typically not result in the geometrically most intuitive solution.

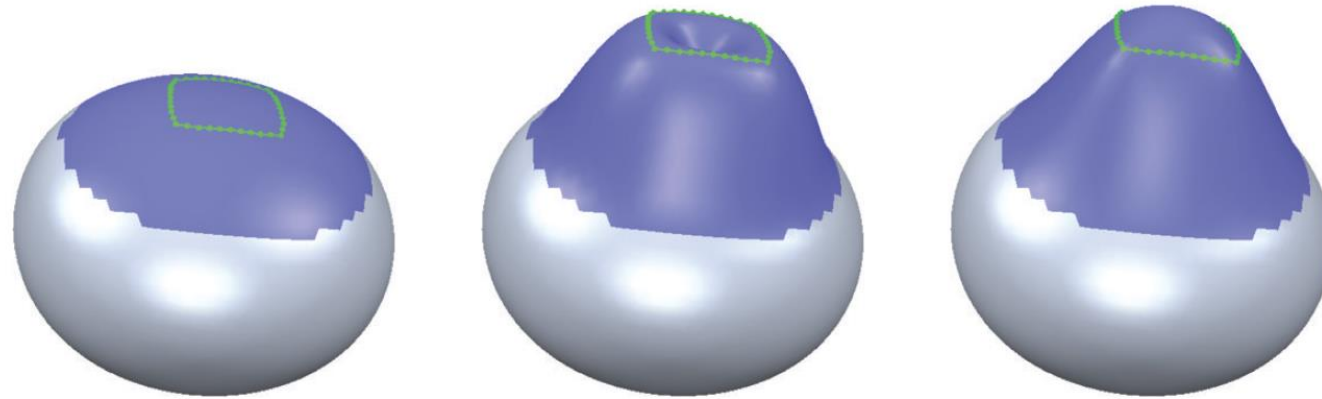


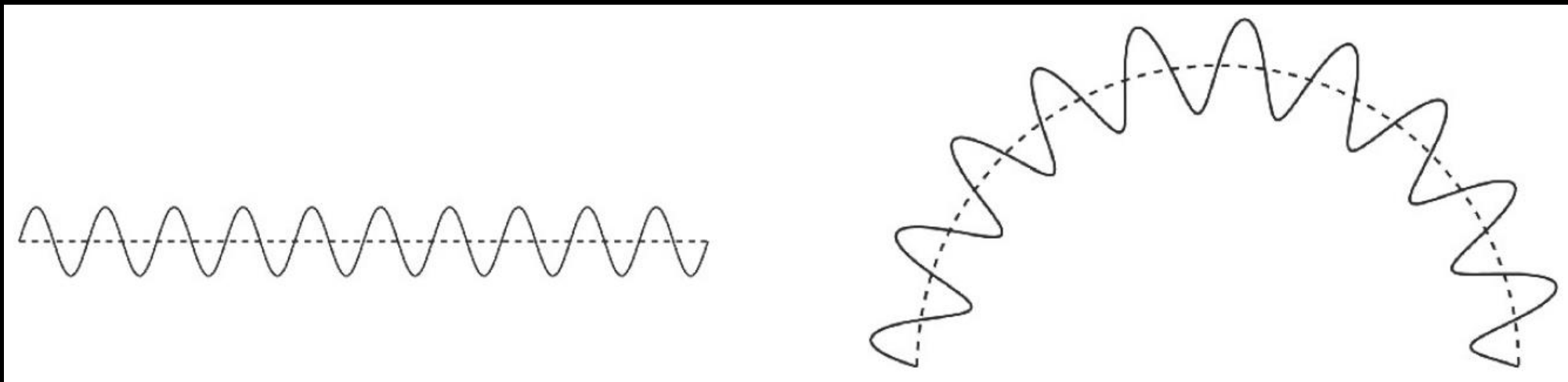
Figure 9.3. A sphere is deformed by lifting a closed handle polygon (left). Propagating this translation based on geodesic distance causes a dent in the interior of the handle polygon (center). A more intuitive solution can be achieved by minimizing physically-motivated deformation energies (right). (Image taken from [Botsch 05].)

Outline

- Definition
- Transformation Propagation
- **Multi-Scale Deformation**
- Differential Coordinates
- Deformation transfer
- As-Rigid-As-Possible surface deformation
- Freeform Deformation
 - Meshless mapping
- Volumetric Deformation
 - Tetrahedral mapping

Multi-scale deformations

- Main idea: decompose the object into two frequency bands using the **smoothing and fairing techniques**.
 - the low frequencies correspond to the smooth global shape;
 - the high frequencies correspond to the fine-scale details.
- Goal: deform the low frequencies (global shape) while preserving the high-frequency details



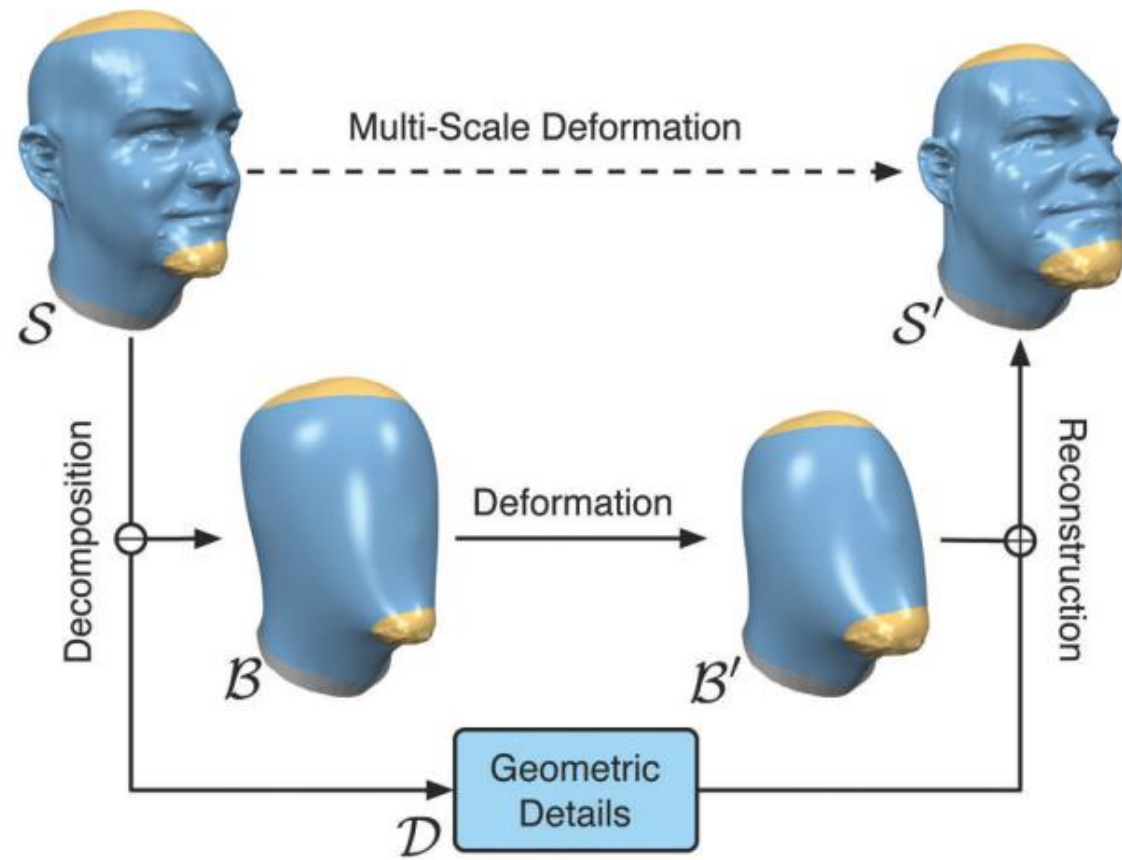


Figure 9.7. A general multi-scale editing framework consists of three main operators: the decomposition operator, which separates the low and high frequencies; the editing operator, which deforms the low frequency components; and the reconstruction operator, which adds the details back onto the modified base surface. Since the lower part of this scheme is hidden in the multi-scale kernel, only the multi-scale edit in the top row is visible to the designer. (Image taken from [Botsch and Sorkine 08]. ©2008 IEEE. Model courtesy of Cyberware.)

Pipeline

- First a low-frequency representation of the given surface S is computed by removing the high frequencies, yielding a smooth base surface B . The geometric detail $D = S \ominus B$.
- Deform the B to B'
- Adding the geometric details onto B' : $S' = B' \oplus D$

- \ominus : decomposition
- \oplus : reconstruction
- mesh smoothing or fairing

Representation for the geometric detail

- The straightforward representation: a vector-valued displacement function
 - associates a displacement vector to each point on the base surface.
 - per-vertex displacement vectors
 - $p_i = b_i + h_i, p_i \in S, b_i \in B, h_i \in R^3$

- Encoded in local frame

$$h_i = \alpha_i n_i + \beta_i t_{i,1} + \gamma_i t_{i,2}$$

n_i : normal

$t_{i,1}, t_{i,2}$: two tangent vectors

Encoded in local frame

- When the base surface S is deformed to S'

$$h'_i = \alpha_i n'_i + \beta_i t'_{i,1} + \gamma_i t'_{i,2}$$

$$p'_i = b'_i + h'_i$$

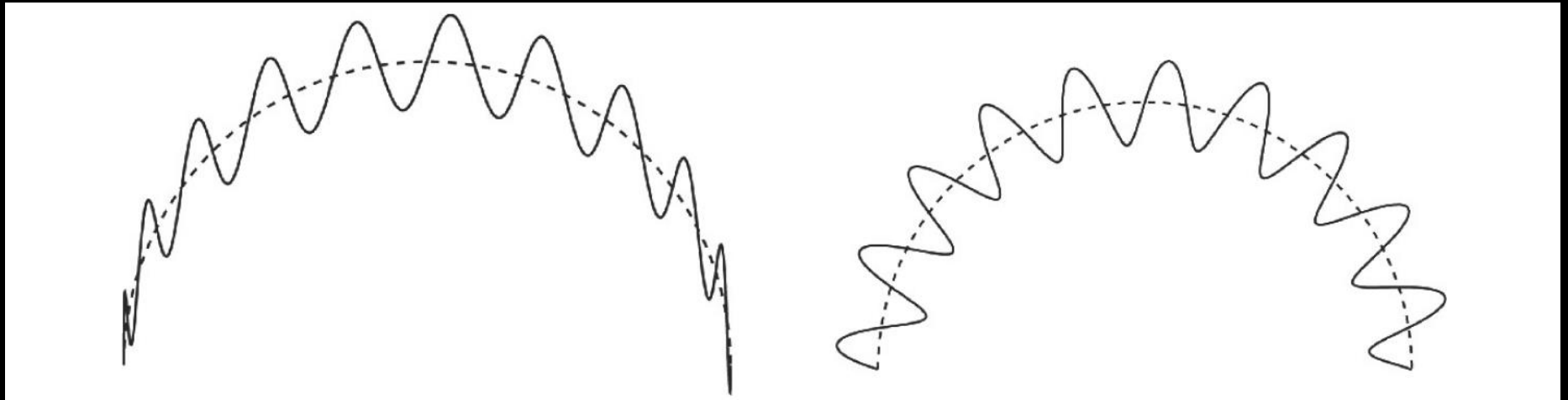


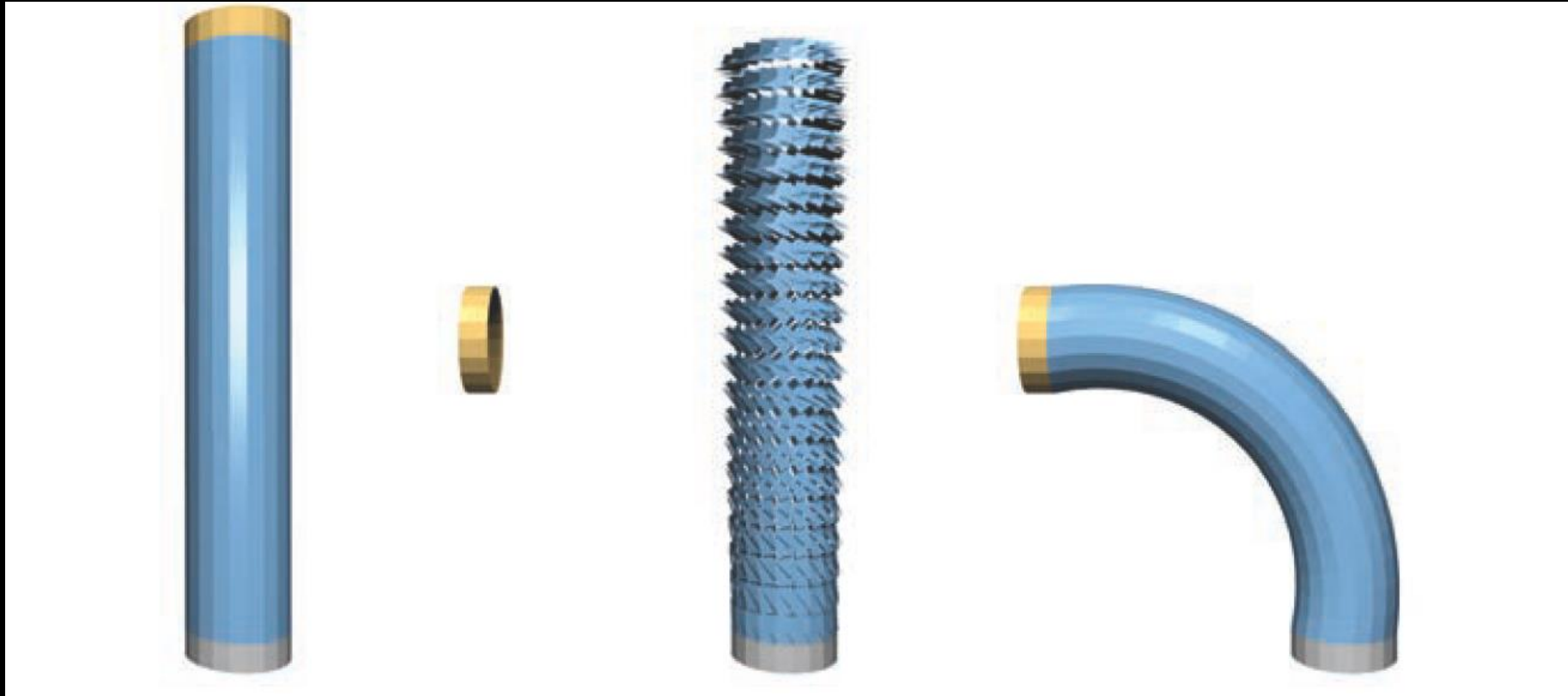
Figure 9.8. Representing the displacements with regard to the global coordinate system does not lead to the desired result (left). The geometrically intuitive solution is achieved by storing the details with regard to local frames that rotate according to the local tangent plane's rotation of \mathcal{B} (right). (Image taken from [Botsch 05].)

Outline

- Definition
- Transformation Propagation
- Multi-Scale Deformation
- **Differential Coordinates**
- Deformation transfer
- As-Rigid-As-Possible surface deformation
- Freeform Deformation
 - Meshless mapping
- Volumetric Deformation
 - Tetrahedral mapping

Gradient-Based Deformation

- deform the surface by
 - 1. manipulating the original surface gradients
 - 2. finding the deformed surface that matches the target gradient field in the least-squares sense



Gradient-Based Deformation

- Gradient of the coordinate function on facet f_i

$$\nabla p_i = \begin{bmatrix} \nabla p_{x,i} \\ \nabla p_{y,i} \\ \nabla p_{z,i} \end{bmatrix} =: J_i \in R^{3 \times 3}$$

The rows of J_i are just the gradients of the x -, y - and z -coordinates.

- Manipulation: $J'_i = M_i J_i$
 - M_i : local rotation/scale/shear, (discussed later)
 - breaking up the mesh. (similar to ARAP parameterization)

Find new vertex positions p'_i $J'_i = M_i J_i$

- Goal: the gradient of p'_i are as close as possible to J'_i

p'_i : new vertex position

A_i : the area of facet f_i

$\nabla p'_i$: the gradient is defined on the original surface (**just replace the function value**)

Solving Laplace equation for x, y, z .

Poisson equation.

Laplacian-Based Deformation

- manipulate **per-vertex Laplacians** instead of per-face gradients
- 1. compute initial **Laplace coordinates** $\delta_i = \Delta(p_i)$
- 2. manipulate them to $\delta'_i = M_i \delta_i$, (**discussed later**)
- 3. find new coordinates p'_i that match the target Laplacian coordinates

$$E = \sum_i A_i \|\Delta(p'_i) - \delta'_i\|_F^2$$

A_i : local average area for vertex i .

bi-Laplacian system

Uniform Laplace or cot Laplace

The cot weight $\Delta(p'_i)$ is same to $\Delta(p_i)$.

Local Transformations M_i for face

- Propagation of deformation gradients.
- The user manipulates the handle by prescribing an affine transformation

$$T(x) = Mx + t$$

M : gradient of $T(x)$

- propagate this matrix over the deformable region and damp it using the smooth scalar field.
- Rotations should be interpolated differently than scalings.

Propagation of deformation gradients

- Polar decomposition:

$$\begin{aligned}M &= R \cdot S \\R &= UV^T \\S &= V\Sigma V^T\end{aligned}$$

Where

$$M = U\Sigma V^T$$

R : rotation; S : scaling

rotation and scaling components are then interpolated separately

Propagation of deformation gradients

- Rotation: quaternion interpolation R_i
- Scaling: linear interpolation $S_i = (1 - s_i)S + s_i \cdot Id$
- $M_i = R_i \cdot S_i$

- Discussion:
 - works very well for rotations
 - insensitive to handle translations

Local Transformations M_i for vertex

- Implicit optimization: **simultaneously optimize** for both the new vertex positions p'_i and the local rotations M_i .

$$E = \sum_i A_i \|\Delta(p'_i) - M_i \delta_i\|_F^2$$

- To avoid a nonlinear optimization and rigid transformation is desired
 - linearized similarity transformations, skew-symmetric matrices

$$M_i = \begin{bmatrix} s_i & -h_{i,z} & h_{i,y} \\ h_{i,z} & s_i & -h_{i,x} \\ -h_{i,y} & h_{i,x} & s_i \end{bmatrix}$$

Local Transformations M_i for vertex

- To determine s_i , $h_{i,x}$, $h_{i,y}$ and $h_{i,z}$:

$$M_i(p_i - p_j) = p'_i - p'_j, \forall j \in \Omega(i)$$

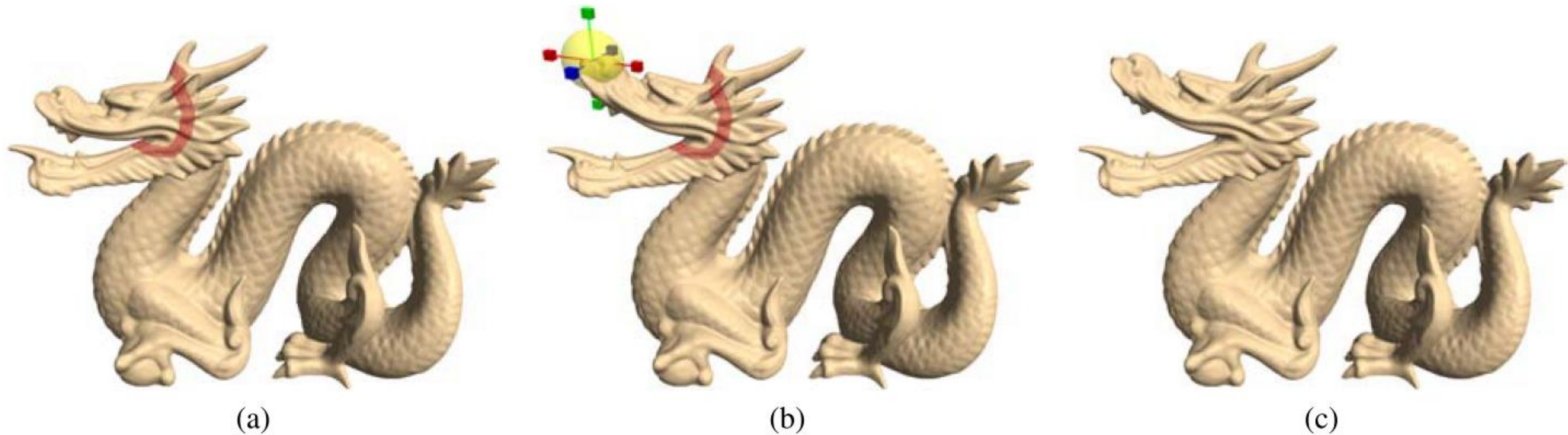
Then:

M_i is a linear combinations of p'_i .

$E = \sum_i A_i \|\Delta(p'_i) - M_i \delta_i\|_F^2$ becomes a quadratic energy.

Linear least-squares problem, which can be solved efficiently.

The linearized transformations lead to artifacts in the case of large rotations.



(a)

(b)

(c)

Figure 1: *The editing process. (a) The user selects the region of interest – the upper lip of the dragon, bounded by the belt of stationary anchors (in red). (b) The chosen handle (enclosed by the yellow sphere) is manipulated by the user: translated and rotated. (c) The editing result.*

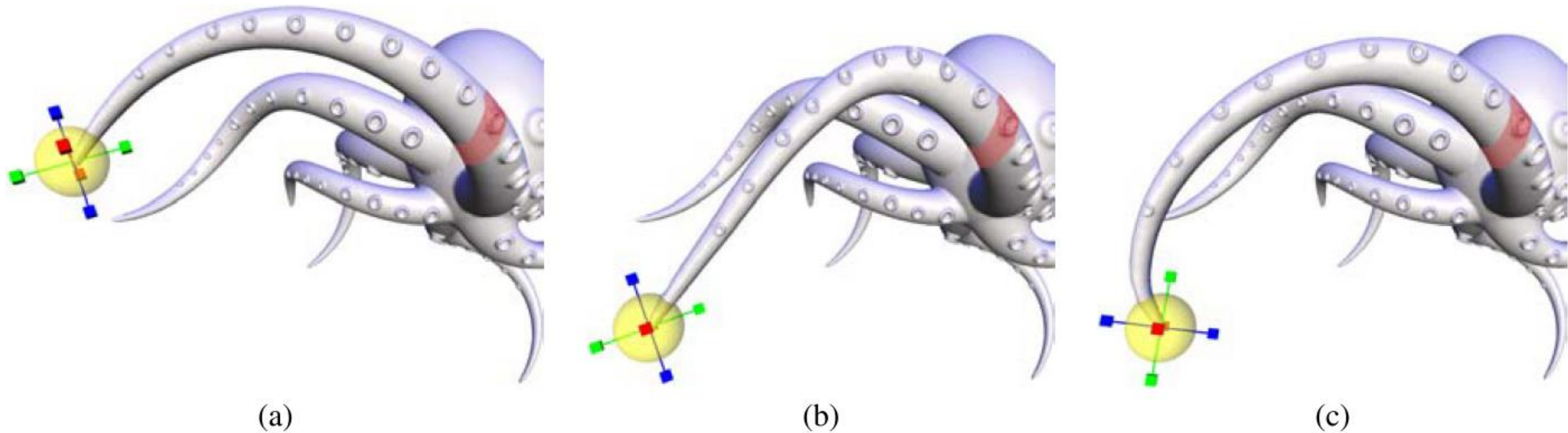


Figure 2: *Different handle manipulations. (a) The region of interest (arm), bounded by the belt of stationary anchors, and the handle. (b) Translation of the handle. (c) Subsequent handle rotation. Note that the detail is preserved in all the manipulations.*

Outline

- Definition
- Transformation Propagation
- Multi-Scale Deformation
- Differential Coordinates
- **Deformation transfer**
- As-Rigid-As-Possible surface deformation
- Freeform Deformation
 - Meshless mapping
- Volumetric Deformation
 - Tetrahedral mapping

Deformation transfer

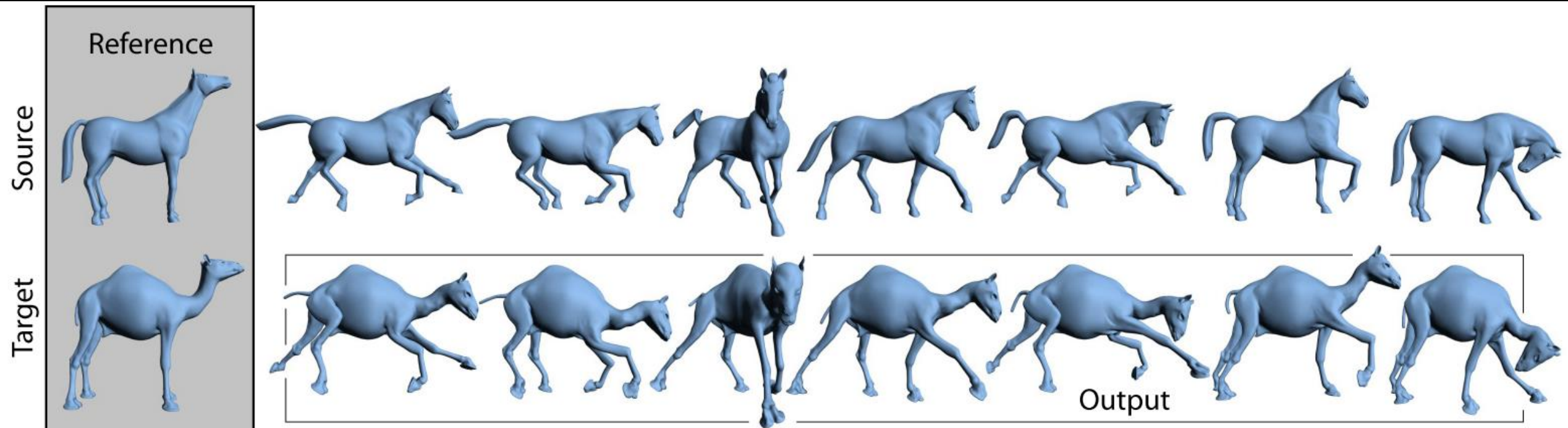
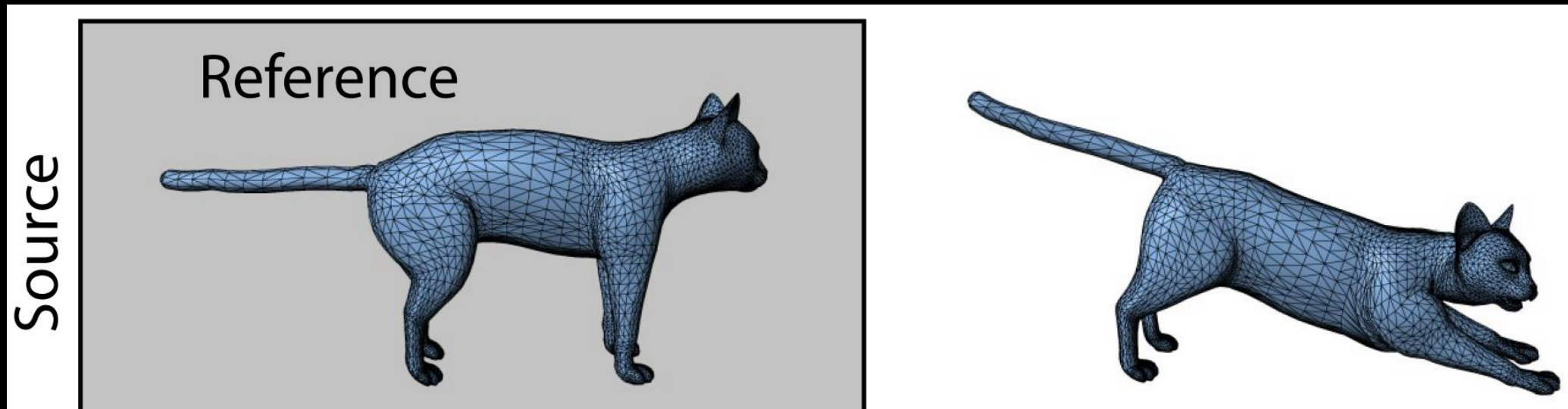


Figure 1: Deformation transfer copies the deformations exhibited by a source mesh onto a different target mesh. In this example, deformations of the reference horse mesh are transferred to the reference camel, generating seven new camel poses. Both gross skeletal changes as well as more subtle skin deformations are successfully reproduced.

Deformation transfer

- The goal of deformation transfer: transfer the **change** in shape exhibited by the source deformation onto the target.
- Input: **source deformation**
 - a collection of **affine transformations** tabulated for each triangle of the source mesh.



- The three vertices of a triangle before and after deformation **do not fully determine the affine transformation**.

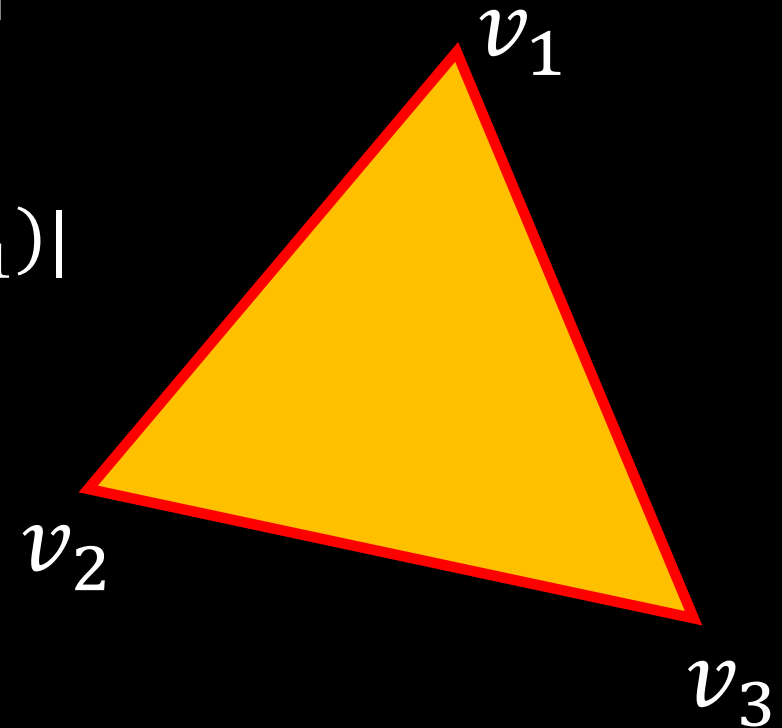
Affine transformation

- $v_i \in$ undeformed, $\tilde{v}_i \in$ deformed
- add a fourth vertex in the direction perpendicular to the triangle.

$$v_4 = v_1 + n$$

$$n = (v_2 - v_1) \times (v_3 - v_1) / |(v_2 - v_1) \times (v_3 - v_1)|$$

1. an analogous computation for \tilde{v}_4
2. How to compute the Affine transformation?



Transfer

- Transfer the **source transformations** via the correspondence map to the target.

$$E = \sum_i \|S_j - T_j\|_F^2$$

1. Require one-to-one correspondence between source and target model.
2. Least squares.

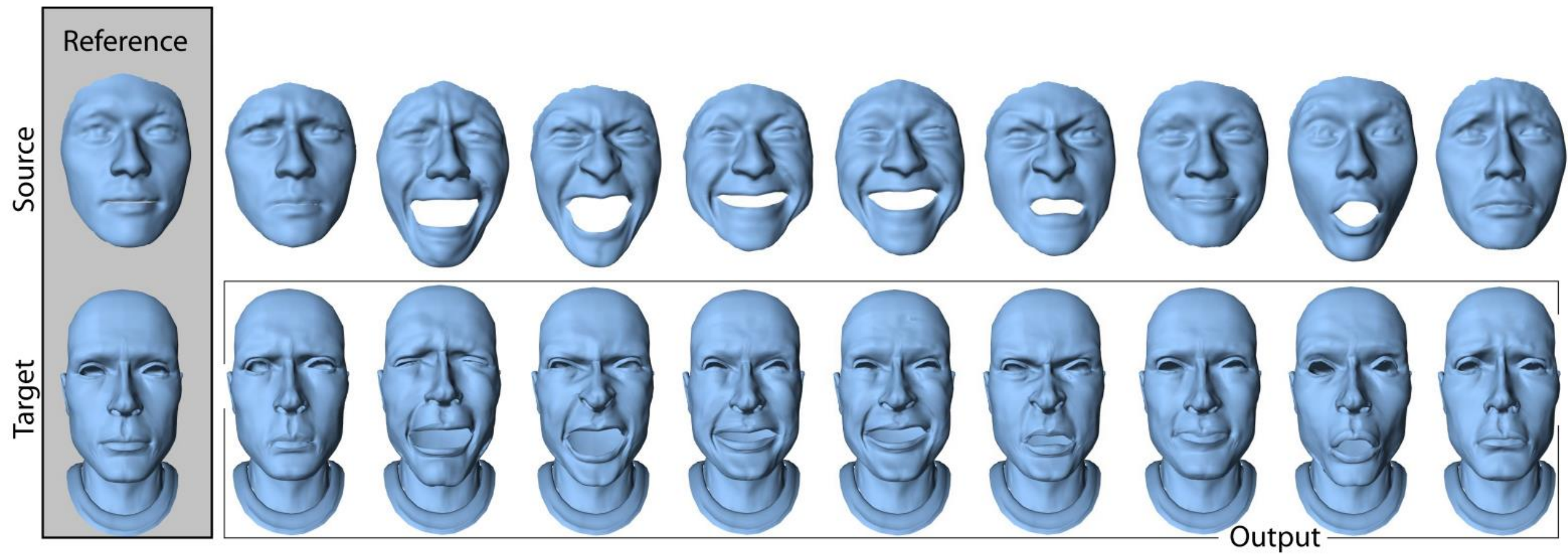


Figure 7: Scanned facial expressions cloned onto a digital character.

Outline

- Definition
- Transformation Propagation
- Multi-Scale Deformation
- Differential Coordinates
- Deformation transfer
- **As-Rigid-As-Possible surface deformation**
- Freeform Deformation
 - Meshless mapping
- Volumetric Deformation
 - Tetrahedral mapping

As-Rigid-As-Possible Surface Modeling

- Goal: preserve shape meaning that an object is only rotated or translated, but not scaled or sheared.
 - small parts of the shape should change as rigidly as possible
- Energy:

$$E = \sum_{i=1}^{N_v} w_i \sum_{j \in \Omega(i)} w_{ij} \left\| (p'_i - p'_j) - R_i(p_i - p_j) \right\|^2$$

w_i, w_{ij} : fixed cell and edge weights.

w_{ij} : cot weight; w_i : local average area

Variables: R_i and p'_i

Local step

$$E = \sum_{i=1}^{N_p} w_i \sum_{j \in \Omega(i)} w_{ij} \|(p'_i - p'_j) - R_i(p_i - p_j)\|^2$$

- Given p'_i , compute R_i

$$E_i = \sum_{j \in \Omega(i)} w_{ij} \|(p'_i - p'_j) - R_i(p_i - p_j)\|^2$$

Set $e'_{ij} = p'_i - p'_j$, $e_{ij} = p_i - p_j$,

$$\begin{aligned} E_i &= \sum_{j \in \Omega(i)} w_{ij} (e'_{ij} - R_i e_{ij})^T (e'_{ij} - R_i e_{ij}) \\ &= \sum_{j \in \Omega(i)} w_{ij} (e'^T_{ij} e'_{ij} - 2e'^T_{ij} R_i e_{ij} + e^T_{ij} e_{ij}) \end{aligned}$$

Local step

$$E = \sum_{i=1}^{N_p} w_i \sum_{j \in \Omega(i)} w_{ij} \|(p'_i - p'_j) - R_i(p_i - p_j)\|^2$$

$$\arg \min_{R_i} \sum_{j \in \Omega(i)} w_{ij} (e_{ij}^T e_{ij} - 2e_{ij}^T R_i e_{ij} + e_{ij}^T e_{ij})$$

$$= \arg \max_{R_i} \sum_{j \in \Omega(i)} w_{ij} 2e_{ij}^T R_i e_{ij} = \arg \max_{R_i} \text{Tr} \left(\sum_{j \in \Omega(i)} w_{ij} R_i e_{ij} e_{ij}^T \right)$$

$$= \arg \max_{R_i} \text{Tr} \left(\sum_{j \in \Omega(i)} w_{ij} R_i e_{ij} e_{ij}^T \right) = \arg \max_{R_i} \text{Tr} \left(R_i \sum_{j \in \Omega(i)} w_{ij} e_{ij} e_{ij}^T \right)$$

Local step

$$E = \sum_{i=1}^{N_p} w_i \sum_{j \in \Omega(i)} w_{ij} \|(p'_i - p'_j) - R_i(p_i - p_j)\|^2$$

Set $S_i = \sum_{j \in \Omega(i)} w_{ij} e_{ij} e_{ij}^T$ and $S_i = U_i \Sigma_i V_i^T$.

If M is a positive-symmetric-definite matrix then for any orthogonal R , $Tr(M) > Tr(RM)$.

The rotation matrix R_i maximizing $Tr(R_i S_i)$ is obtained when $R_i S_i$ is symmetric positive semi-definite.

$$\Rightarrow R_i = V_i U_i^T.$$

Global step

$$E = \sum_{i=1}^{N_v} w_i \sum_{j \in \Omega(i)} w_{ij} \|(p'_i - p'_j) - R_i(p_i - p_j)\|^2$$

- Given R_i , compute p'_i
- Linear squares, easy to solve
- Initial deformation
 - 1. Previous frame (for interactive manipulation)
 - 2. Naive Laplacian editing
 - ...

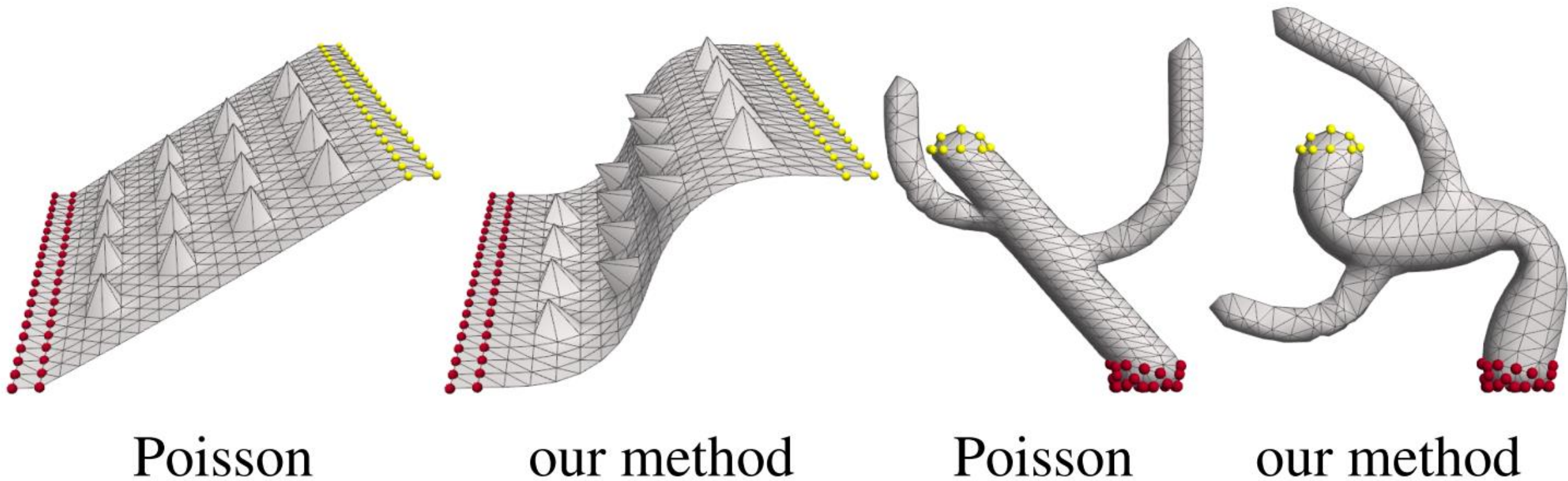


Figure 5: Comparison with Poisson mesh editing. The original models appear in Figures 2 and 7. The yellow handle was only translated; this poses a problem for rotation-propagation methods such as [YZX*04, ZRKS05, LSLCO05].

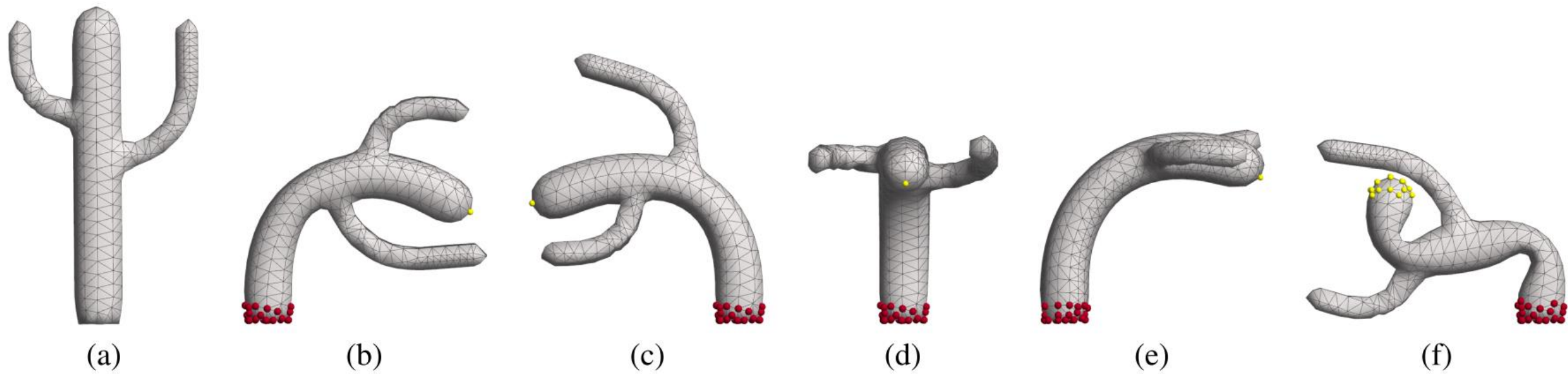


Figure 7: *Bending the Cactus. (a) is the original model; yellow handles are translated to yield the results (b-f). (d) and (e) show side and front views of forward bending, respectively. Note that in (b-e) a single vertex at the tip of the Cactus serves as the handle, and the bending is the result of translating that vertex, no rotation constraints are given.*

Outline

- Definition
- Transformation Propagation
- Multi-Scale Deformation
- Differential Coordinates
- Deformation transfer
- As-Rigid-As-Possible surface deformation
- **Freeform Deformation**
 - Meshless mapping
- Volumetric Deformation
 - Tetrahedral mapping

Space deformations

- Deform the ambient space and thus implicitly deform the embedded objects.

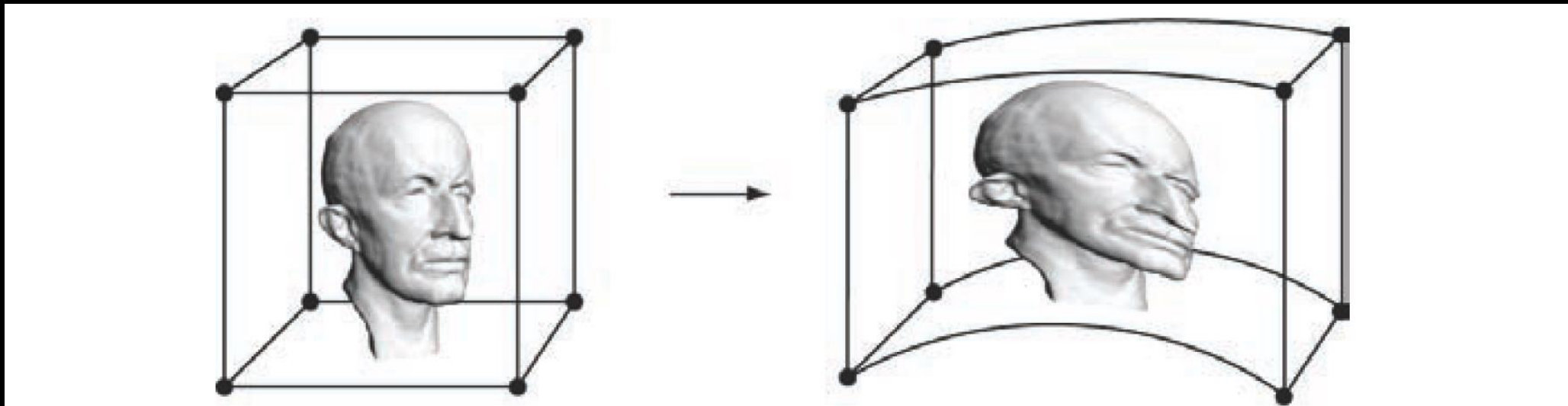


Figure 9.12. Space deformations warp the embedding space around an object and thus implicitly deform the object. (Image taken from [Botsch et al. 06b]. ©2006 ACM, Inc. Included here by permission.)

Lattice-Based Freeform Deformation

- Freeform deformation represents the space deformation by a trivariate tensor-product spline function

$$d(u, v, w) = \sum_i \sum_j \sum_k \delta c_{ijk} N_i(u) N_j(v) N_k(w)$$

1. N_i are B-spline basis functions
2. $\delta c_{ijk} = c'_{ijk} - c_{ijk}$ displacements of the control points c_{ijk}
3. Original vertex p_i satisfying

$$p_i = \sum_i \sum_j \sum_k c_{ijk} N_i(u) N_j(v) N_k(w)$$

New vertex $p'_i = p_i + d(u, v, w) = \sum_i \sum_j \sum_k c'_{ijk} N_i(u) N_j(v) N_k(w)$

Deformation

- A handle-based interface for direct manipulation.
- Input a set of displacement constraints: \bar{d}_i for $H \cup F = \{p_1, \dots, p_m\}$.
- Least squares:

$$E = \sum_{l=1}^m \left| \bar{d}_i - \sum_i \sum_j \sum_k \delta c_{ijk} N_i(u) N_j(v) N_k(w) \right|^2$$

After getting c'_{ijk} , the deformed surface is determined.

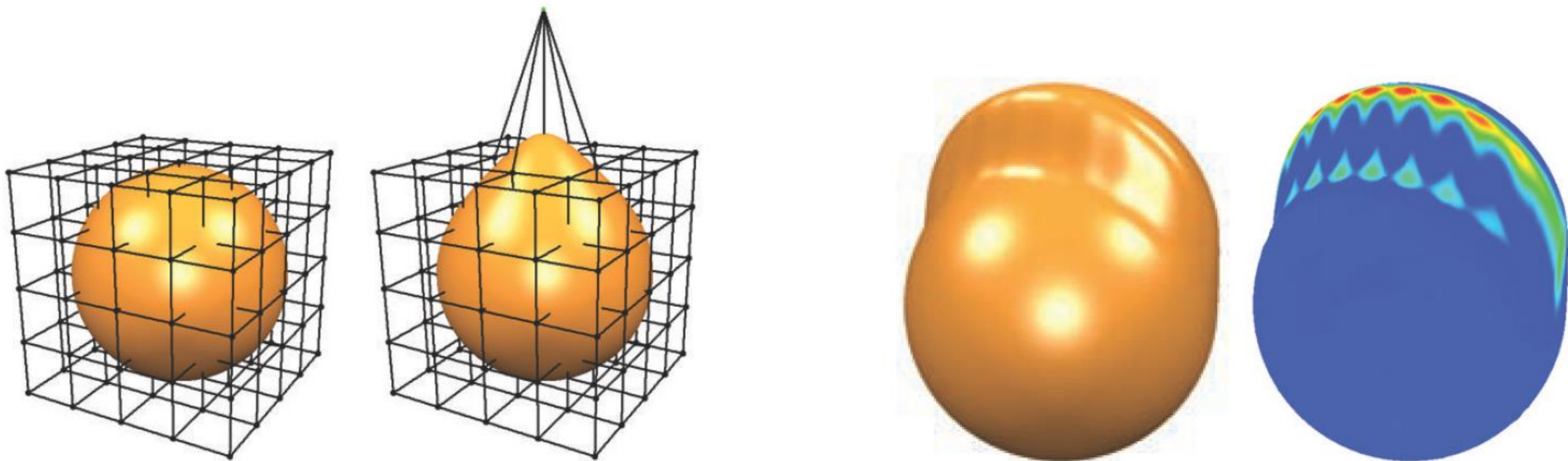


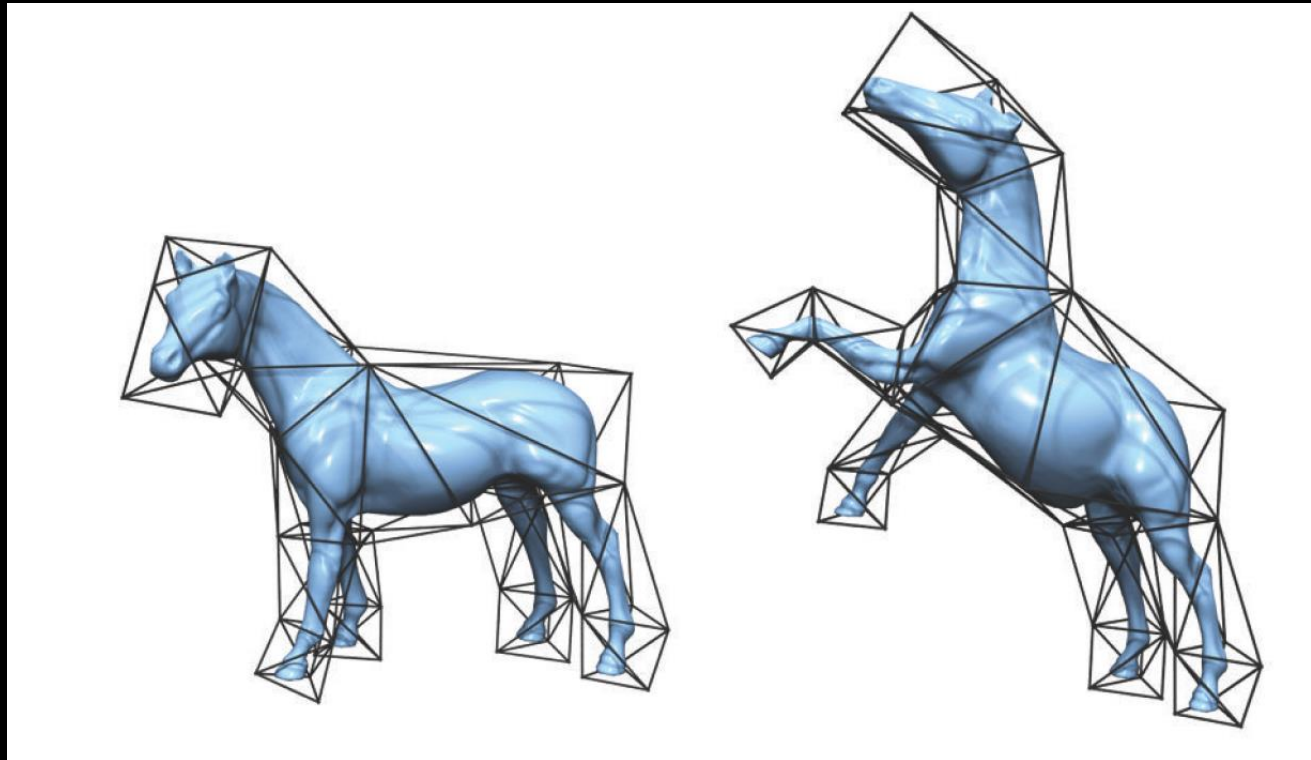
Figure 9.13. In the FFD approach a 3D control grid is used to specify a volumetric displacement function (left). The regular placement of grid basis functions can lead to alias artifacts in the deformed surface (right). (Image taken from [Botsch 05].)

Discussion

- Two drawbacks:
 - Displacement constraints cannot be satisfied exactly.
 - The placement of basis functions on a regular grid.
- How to support concave region?

Cage-Based Freeform Deformation

- A generalization of the lattice-based freeform deformation
- This cage typically is a coarse, arbitrary triangle mesh enclosing the object to be modified.



Deformation

- The vertices p_i of the original mesh S :

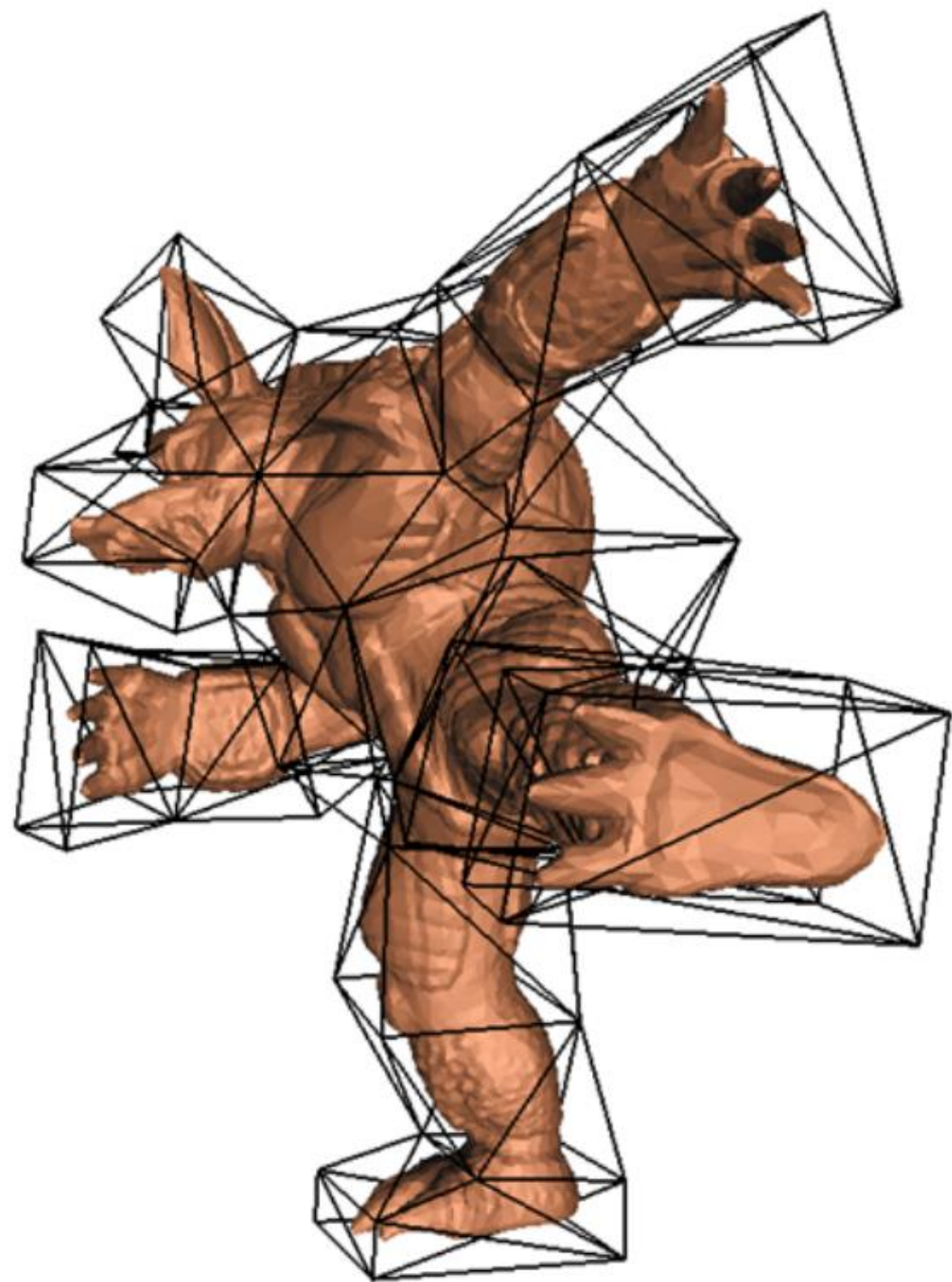
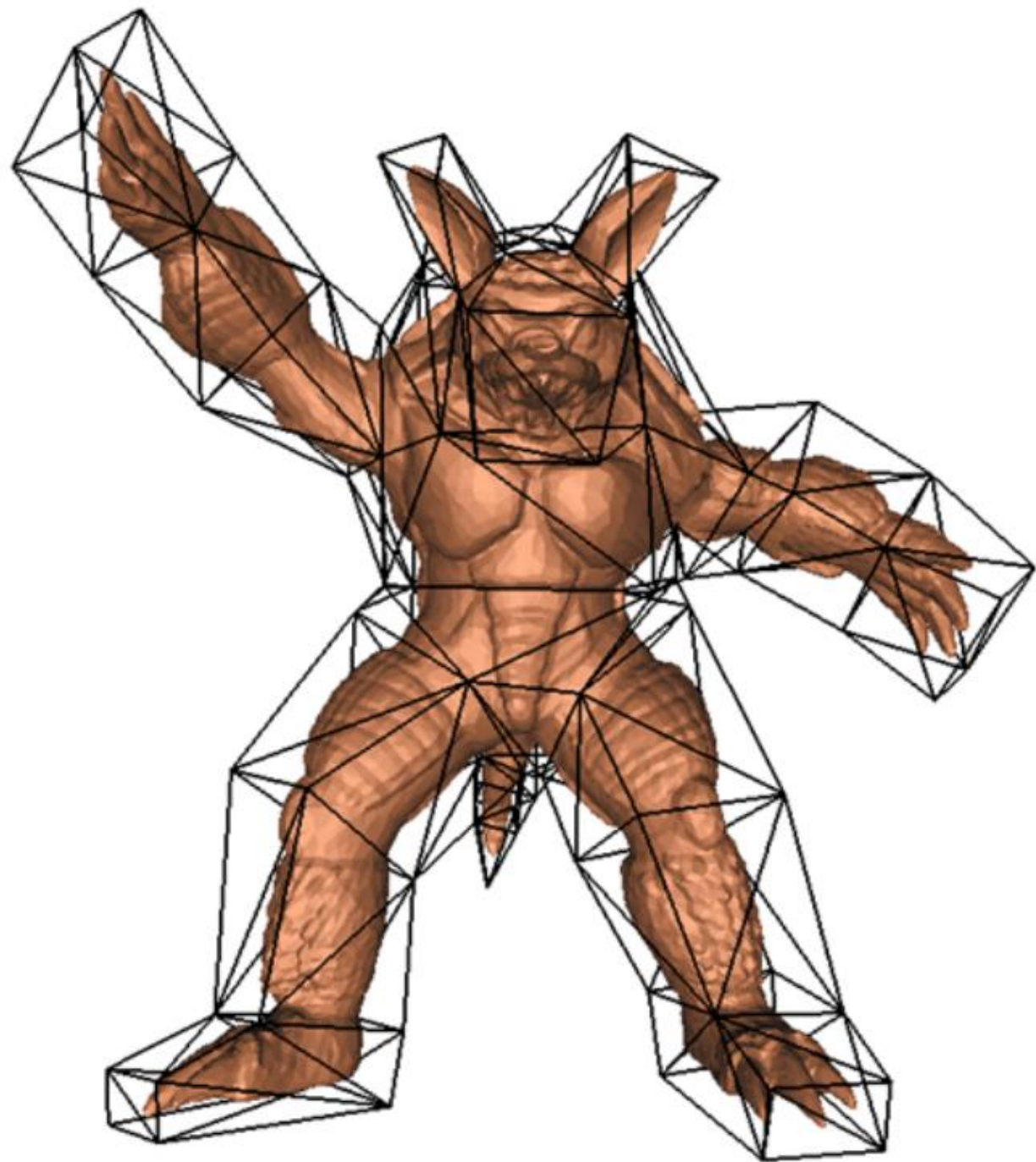
$$p_i = \sum_{l=1}^n c_l \varphi_l(p_i)$$

n : the vertex number of cage mesh

$\varphi_l(p_i)$: generalized barycentric coordinates

- Deform by manipulating the cage vertices $c_l \mapsto c_l + \delta c_l$, displacement:

$$d(p_i) = \sum_{l=1}^n \delta c_l \varphi_l(p_i)$$



Outline

- Definition
- Transformation Propagation
- Multi-Scale Deformation
- Differential Coordinates
- Deformation transfer
- As-Rigid-As-Possible surface deformation
- Freeform Deformation
 - Meshless mapping
- **Volumetric Deformation**
 - Tetrahedral mapping

How to implement ARAP
tetrahedral deformation?

Mappings

Xiao-Ming Fu

Outlines

- Introduction
- Maintenance-based methods
- Bounded distortion methods
- Representation-based method

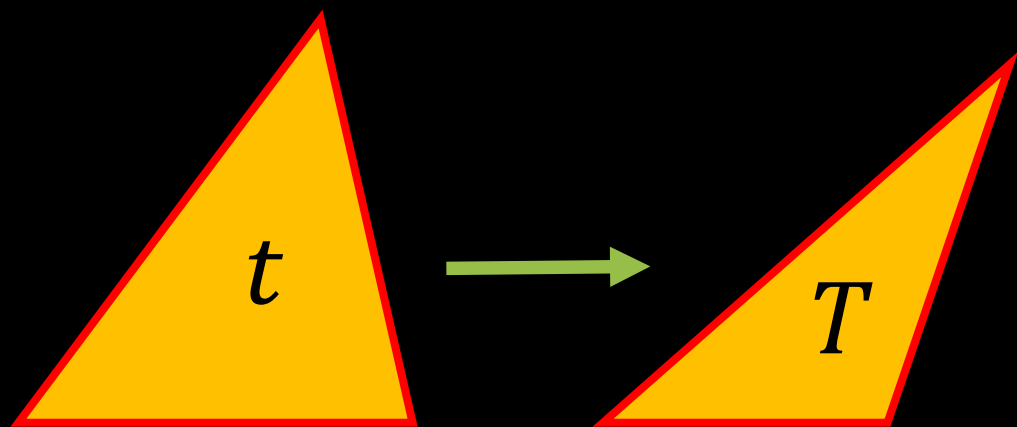
Outlines

- Introduction
- Maintenance-based methods
- Bounded distortion methods
- Representation-based method

Mappings

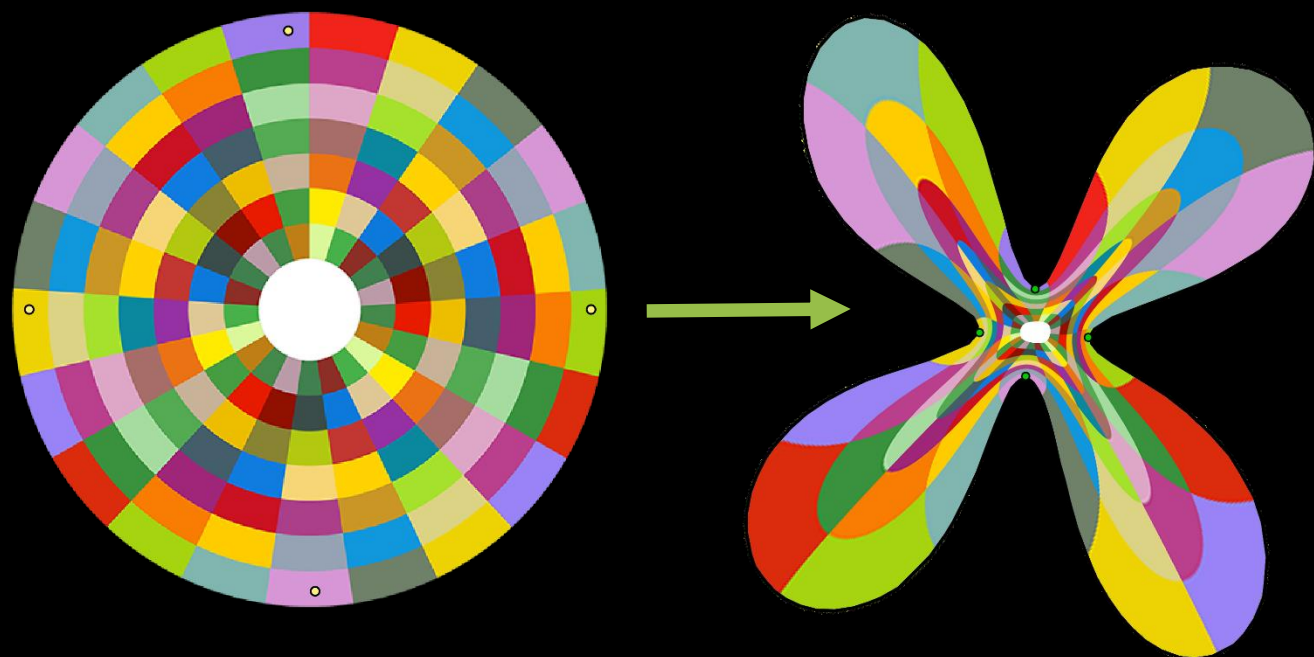
$$f: \Omega \subset \mathbb{R}^d \rightarrow \mathbb{R}^d$$

- mesh-based mapping



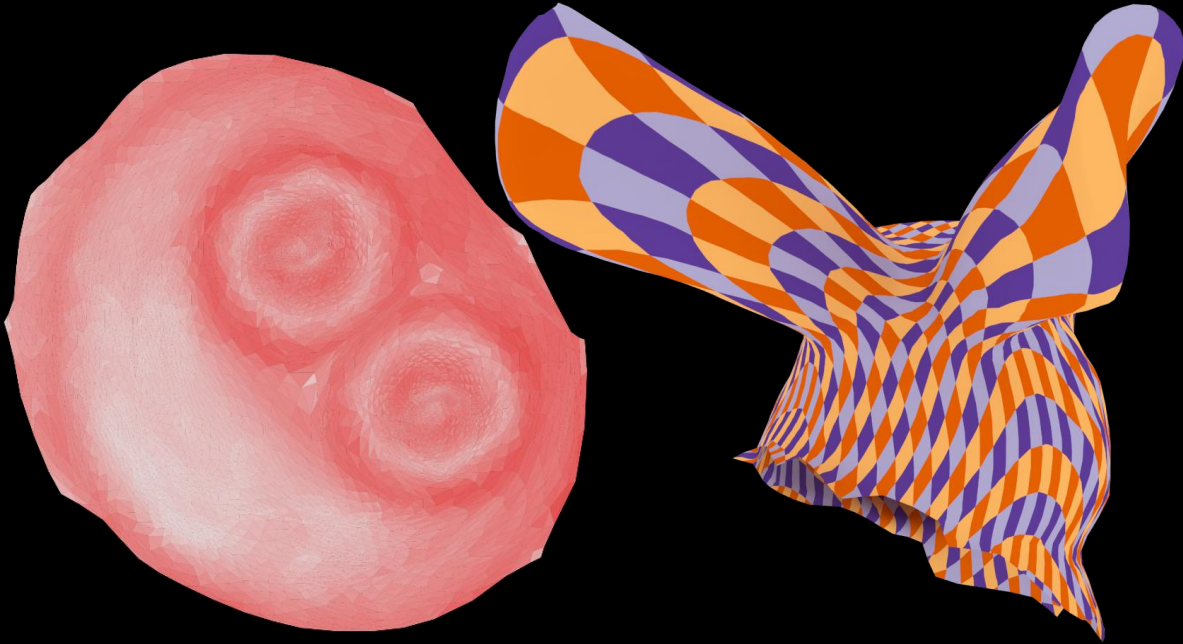
$$f_t(\mathbf{x}) = J_t \mathbf{x} + \mathbf{b}_t$$

- meshless mapping

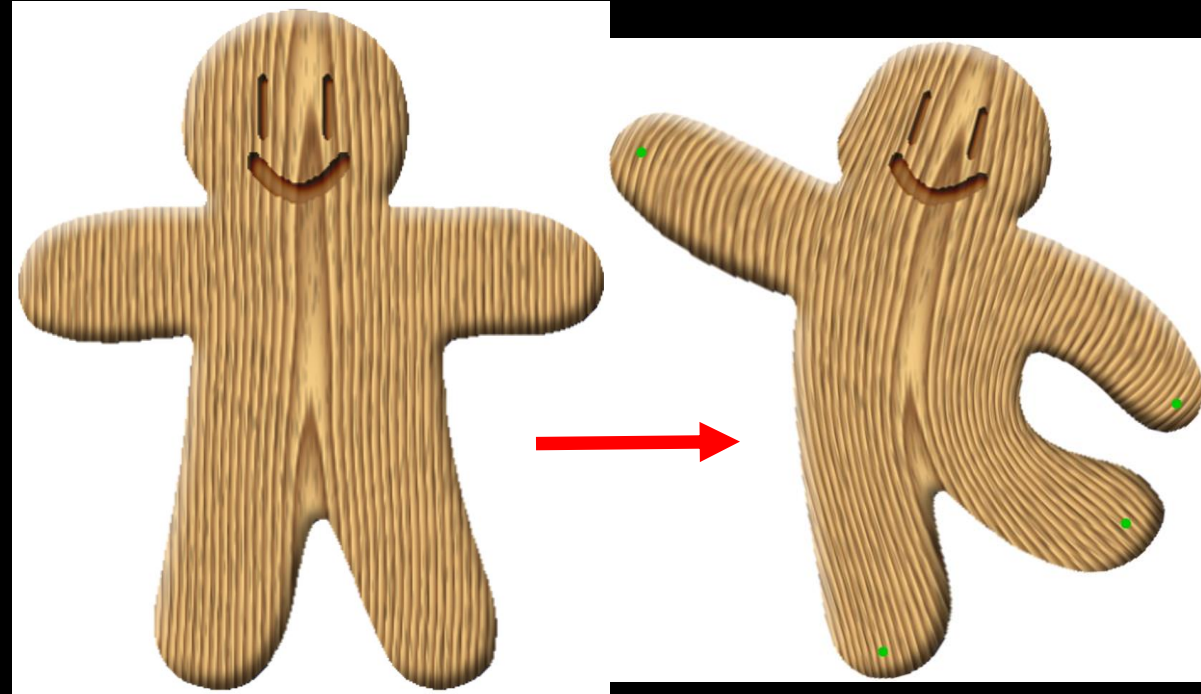


$$f(\mathbf{x}) = \mathbf{x} + \sum_{i=1}^m \mathbf{c}_i B_i(\mathbf{x})$$

Applications

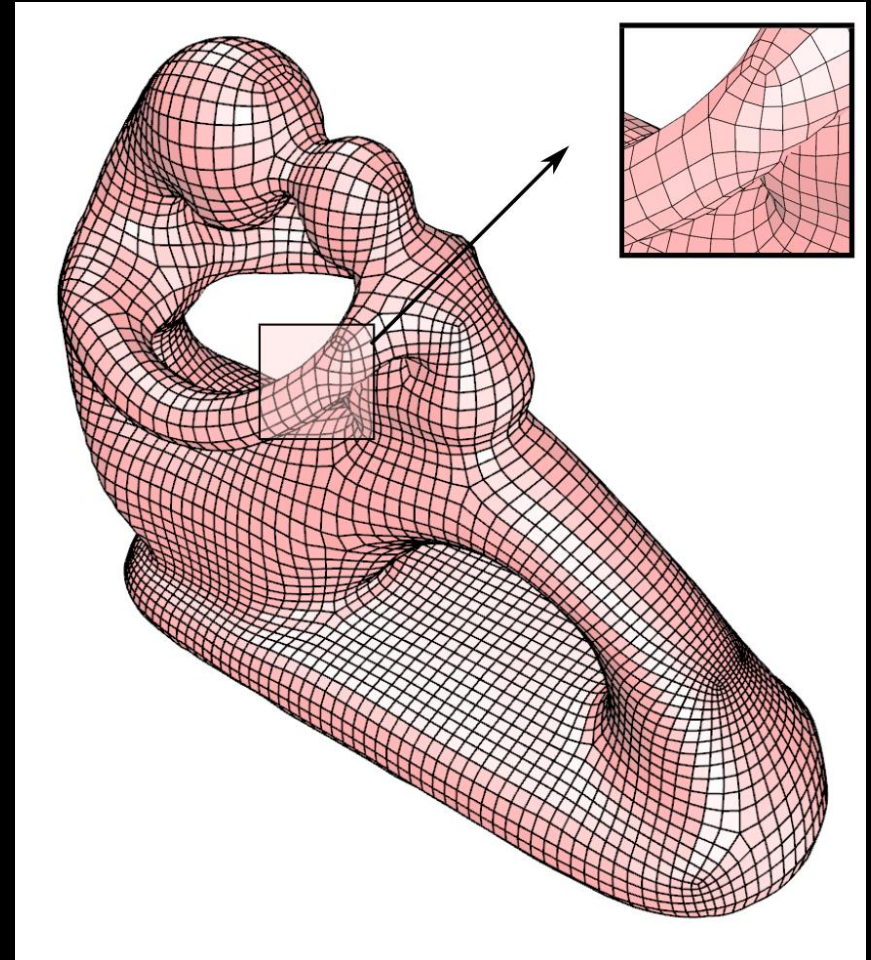
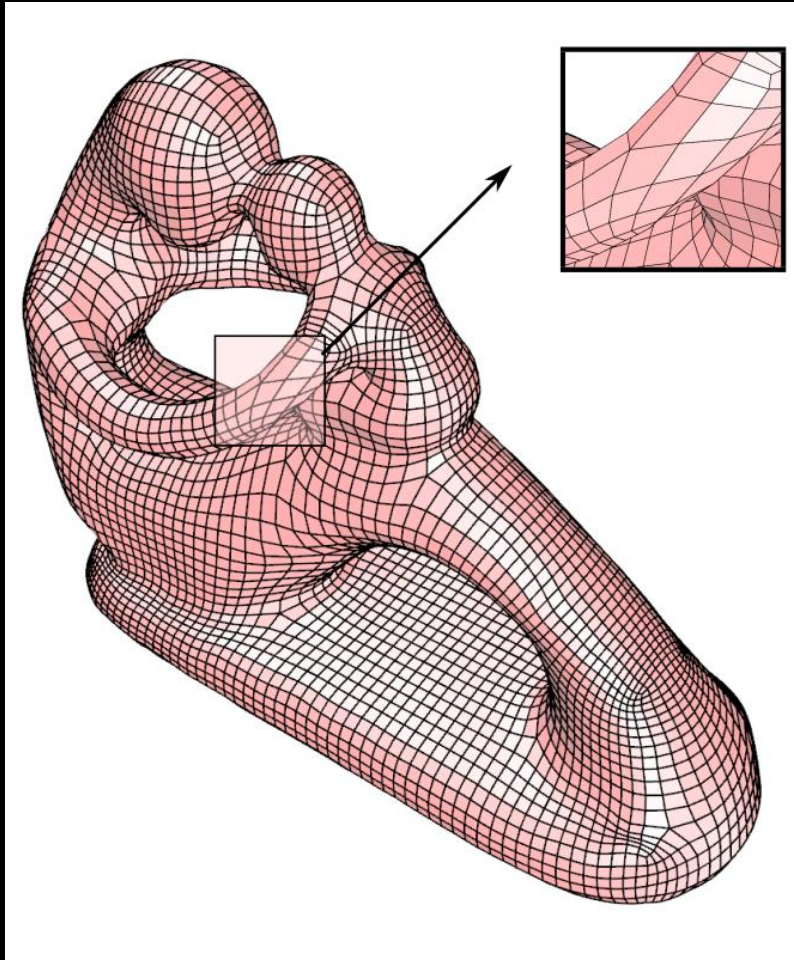


Parameterization



Deformation

Application - Mesh Improvement



[Gregson et al. 2011]

Improved results

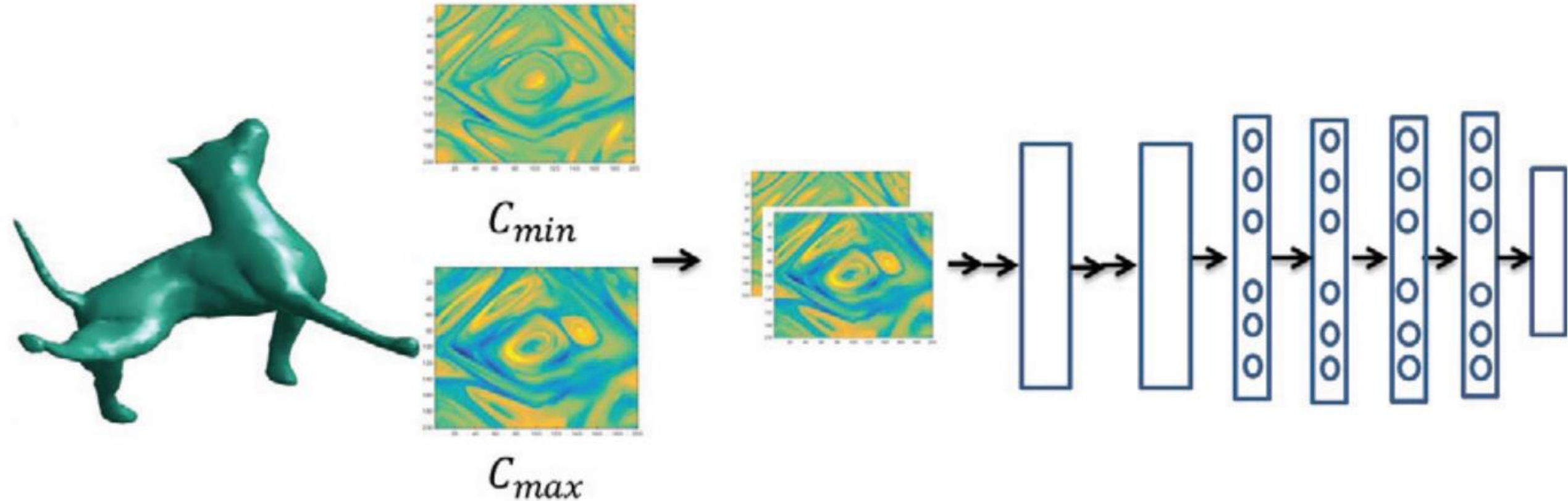
Application – Compatible remeshing



All meshes share a common connectivity.

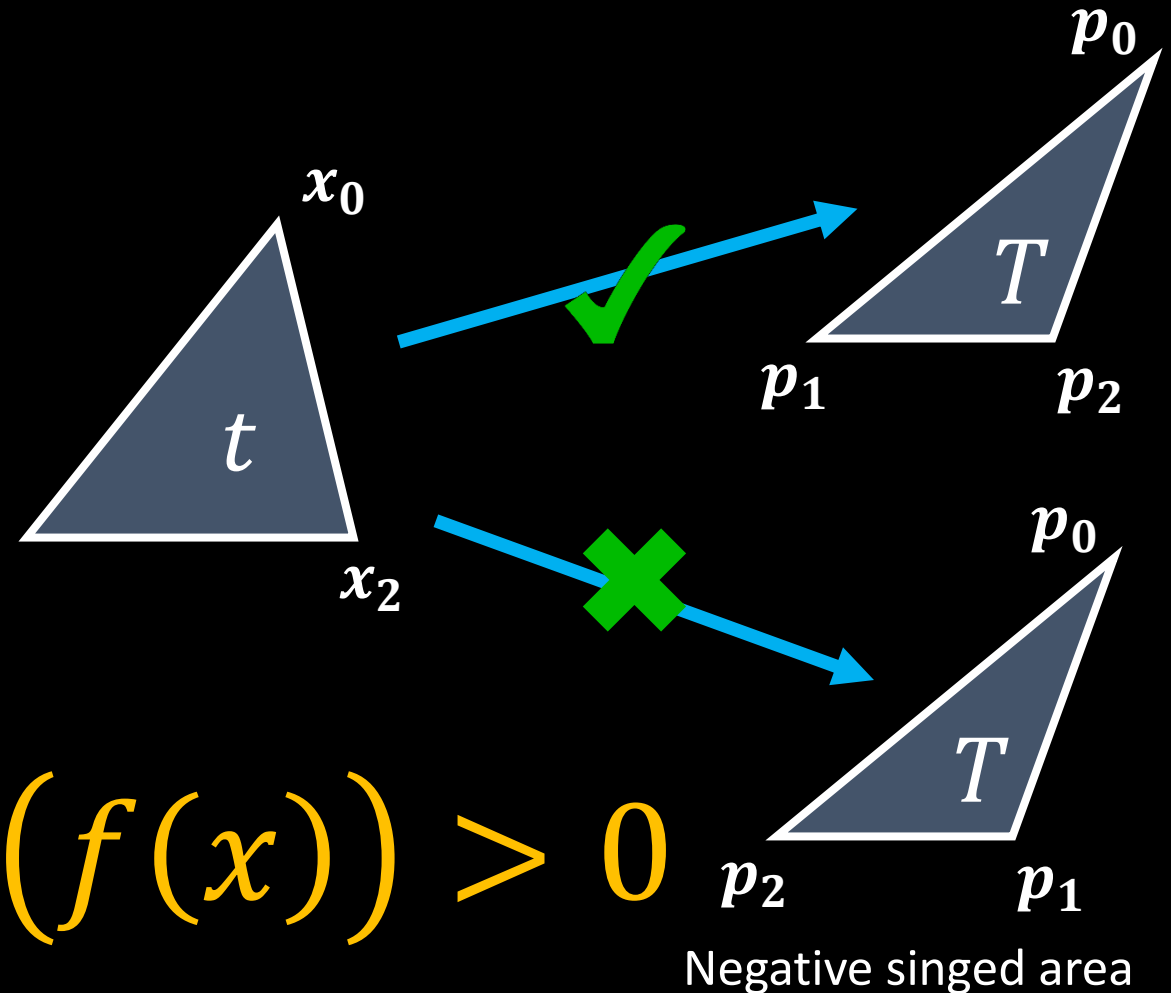
Application - Learning

3D shape \longrightarrow Geometry Image \longrightarrow Convolutional Neural Net



Basic requirements

- Foldover-free:
 - No realistic material can be compressed to zero or even negative volume.
 - Flipped elements correspond to physically impossible deformation.
 - Inverted elements lead to invalidity for following applications, for example, remeshing.
 -



Basic goal – low distortion

- Distortion

- Rotation: rigid transformation

- ✓ Isometric = conformal + equiareal

- ✓ $\delta^{iso} = \max\{\sigma_{max}, \frac{1}{\sigma_{min}}\}$

- Similar transformation

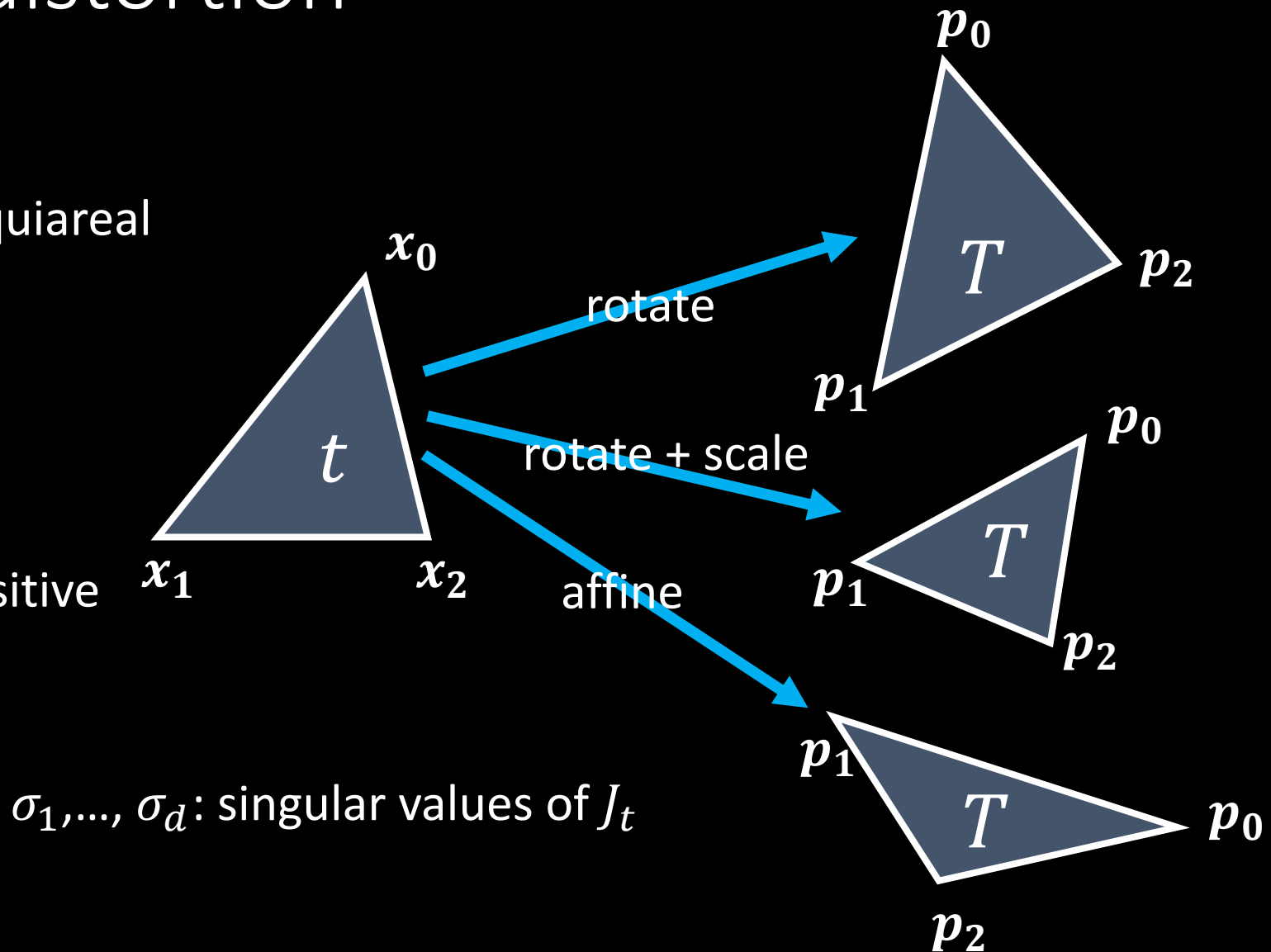
- ✓ Conformal

- ✓ $\delta^{con} = \sigma_{max}/\sigma_{min}$

- Affine transformation with positive determinant

- Our goal

- As Rigid As Possible
 - As similar as Possible



Common distortion metrics $D(f)$

- Common conformal distortion

- LSCM: $\sum_t A_t (\sigma_1 - \sigma_2)^2$

- MIPS: $\sum_t \frac{\sigma_1}{\sigma_2} + \frac{\sigma_2}{\sigma_1}$

- Common isometric distortion

- ARAP: $\sum_t A_t ((\sigma_1 - 1)^2 + (\sigma_2 - 1)^2)$

- AMIPS: $\sum_t \left(\left(\frac{\sigma_1}{\sigma_2} + \frac{\sigma_2}{\sigma_1} \right) + \left(\frac{1}{\sigma_2 \sigma_1} + \sigma_2 \sigma_1 \right) \right)$

- Symmetric Dirichlet: $\sum_t A_t (\sigma_1^2 + \sigma_1^{-2} + \sigma_2^2 + \sigma_2^{-2})$

Formulation

$$\begin{aligned} \min_f D(f) \\ \text{s. t. } \det J(f(x)) > 0, \forall x \in M \\ S(f) \leq 0 \end{aligned}$$

$S(f) \leq 0$: specific constraints for applications

$D(f)$: distortion metric

M : input mesh or domain

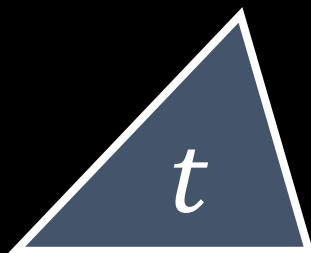
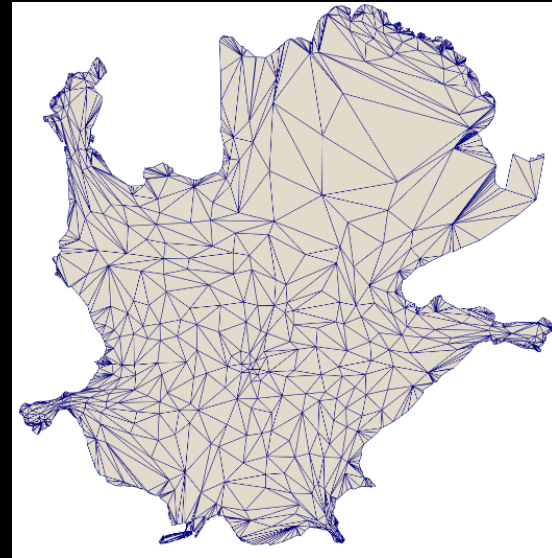
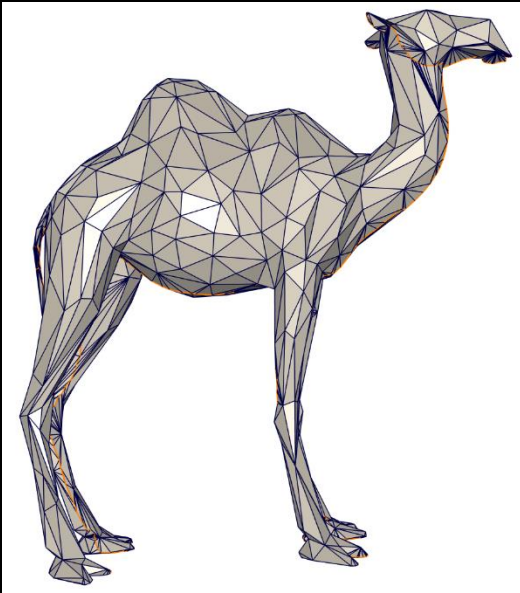
Outlines

- Introduction
- Maintenance-based methods
 - MIPS: An efficient global parametrization method
- Bounded distortion methods
- Representation-based method

Most-Isometric Parameterization (MIPS)

[Hormann and Greiner 2000]

- Mapping: a triangle mesh \rightarrow 2D parameterization region



$$f_t(\mathbf{x}) = J_t \mathbf{x} + \mathbf{b}_t$$

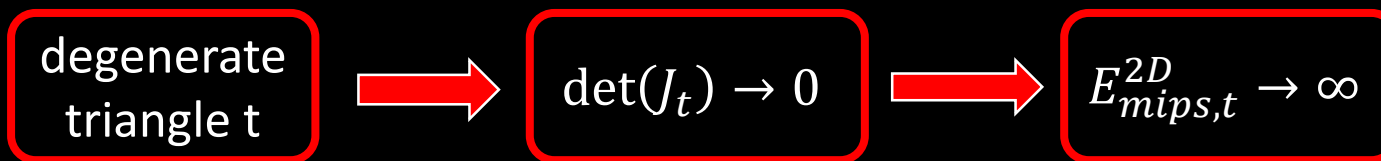


MIPS energy

- MIPS energy on triangle t

$$E_{mips,t}^{2D} = \frac{\sigma_1}{\sigma_2} + \frac{\sigma_2}{\sigma_1} = \|J_t\|_F \|J_t^{-1}\|_F = \frac{\text{trace}(J_t^T J_t)}{\det(J_t)}$$

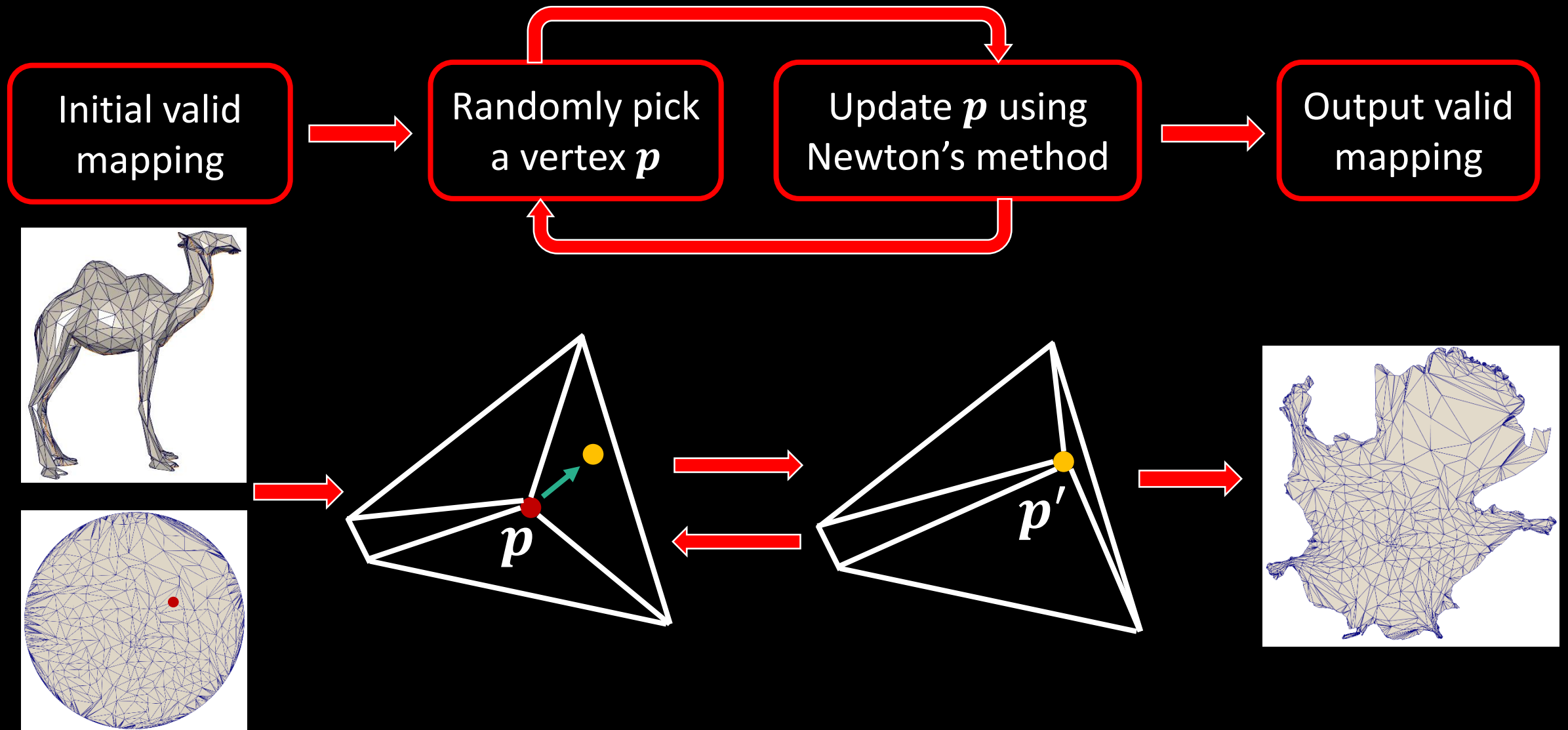
penalize degenerate triangles



a conformal energy

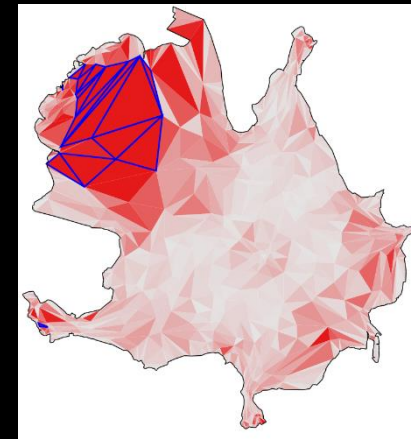
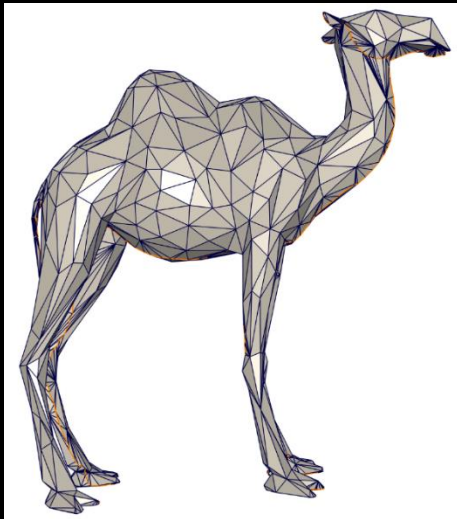
Optimal value when $\sigma_1 = \sigma_2$

MIPS optimization – $\min \sum_t E_{mips,t}^{2D}$



MIPS discussion

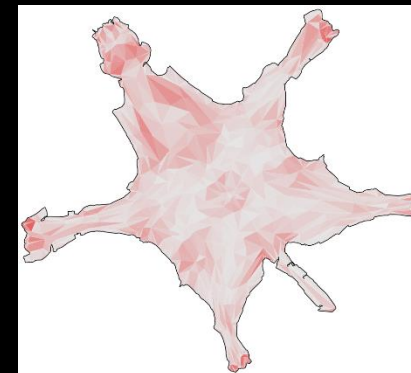
- Advantage: penalize degenerate triangles
- Disadvantages:
 - only for 2D conformal mapping
 - easily be trapped by local minimum
 - no strong penalization on maximal distortion



MIPS:

$$\delta_{max}^{con} = 15.72$$

Time: 7.09s



AMIPS:

$$\delta_{max}^{con} = 3.96$$

Time: 1.68s

Maintenance-based methods

- 1. An initial mapping that satisfies the constraints.
- 2. Reduce the distortion as much as possible while **not violating** the constraints.

- Parameterizations:
 - Initialization: Tutte's embedding
 - distortion metrics
 - **Solvers**

Complex solvers

- AMIPS: Computing locally injective mappings by advanced MIPS (2015)
- AQP: Accelerated Quadratic Proxy for Geometric Optimization (2016)
- SLIM: Scalable locally injective mappings (2017)
- CM: Geometric optimization via composite majorization (2017)
- AKVF: Isometry-Aware Preconditioning for Mesh Parameterization (2017)
-

Outlines

- Introduction
- Maintenance-based methods
- Bounded distortion methods
 - Bounded distortion mapping spaces for triangular meshes
- Representation-based method

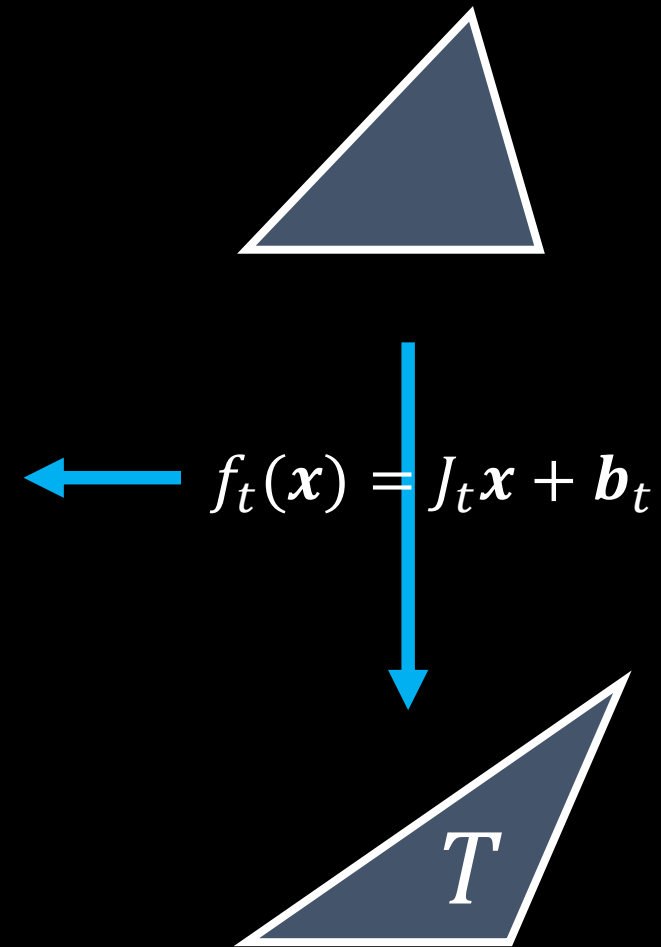
Bounded distortion mapping [Lipman 2012]

- Goal: explicitly bound the conformal distortion

Constraints: $\delta_t^{con} < K, \det(J_t) > 0$

non-linear and non-convex

$$J_t = \begin{pmatrix} a_t & b_t \\ c_t & d_t \end{pmatrix}$$
$$\mathbf{b}_t = \begin{pmatrix} b_{t,x} \\ b_{t,y} \end{pmatrix}$$



Rewrite the constraints

$$J_t = \begin{pmatrix} a_t + c_t & d_t - b_t \\ d_t + b_t & a_t - c_t \end{pmatrix} \quad \sigma_{max} = \sqrt{a_t^2 + b_t^2} + \sqrt{c_t^2 + d_t^2}$$

$$\sigma_{min} = \left| \sqrt{a_t^2 + b_t^2} - \sqrt{c_t^2 + d_t^2} \right|$$

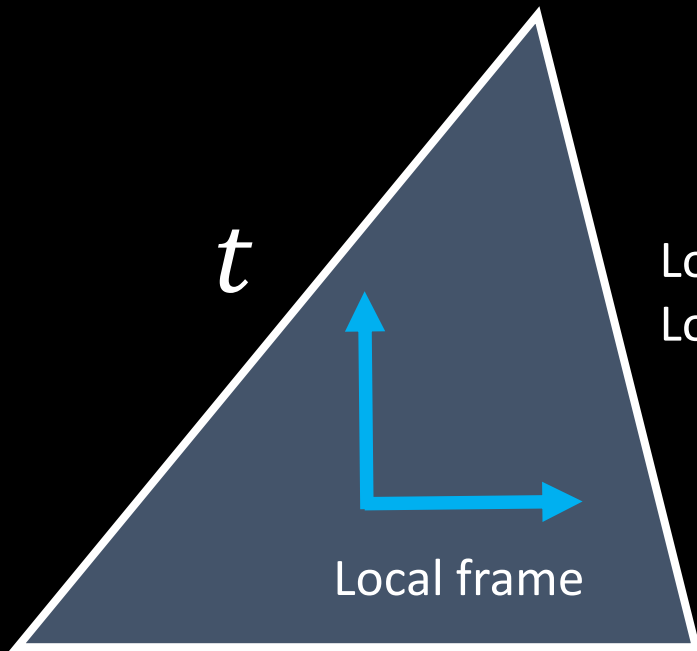
$$\delta_t^{con} < K, \det(J_t) > 0$$

$$\det(J_t) > 0 \longrightarrow \sqrt{c_t^2 + d_t^2} < \sqrt{a_t^2 + b_t^2} \quad \left. \begin{array}{l} r_t \leq \sqrt{a_t^2 + b_t^2} \quad \text{Non-convex} \\ \sqrt{c_t^2 + d_t^2} \leq \frac{K-1}{K+1} r_t \quad \text{Convex} \end{array} \right\}$$

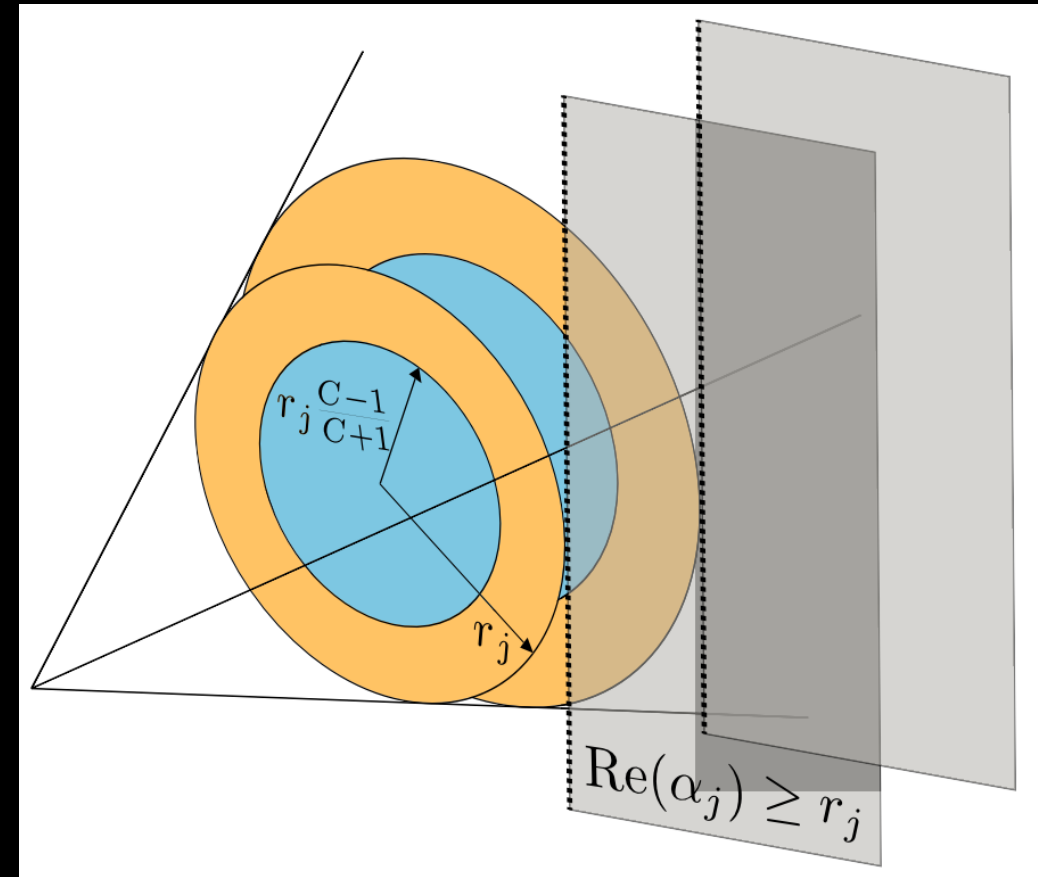
$$\delta_t^{con} < K \longrightarrow \sqrt{c_t^2 + d_t^2} \leq \frac{K-1}{K+1} \sqrt{a_t^2 + b_t^2} \quad \left. \begin{array}{l} r_t > 0 \quad \text{Convex} \end{array} \right\}$$

maximal convex subset

$$r_t \leq \sqrt{a_t^2 + b_t^2} \longrightarrow r_t \leq a_t \quad \text{Convex}$$



Local frame changes, a_t changes.
Local frame is also a variable.



$$\alpha_j = a_j + i \cdot b_j$$

Optimization

- Objective function:

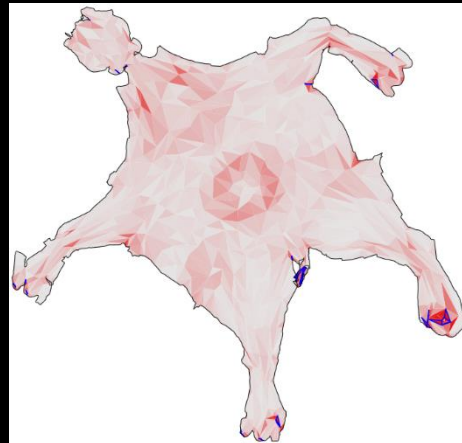
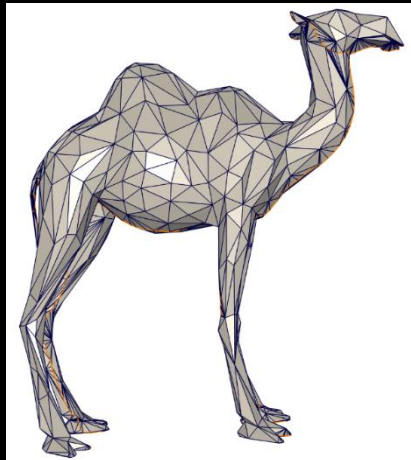
- LSCM: $E = \sum_t Area(t) \cdot (c_t^2 + d_t^2)$

- ARAP: $E = \sum_t Area(t) \cdot \left((a_t - 1)^2 + b_t^2 + c_t^2 + d_t^2 \right)$

$$J_t = \begin{pmatrix} a_t + c_t & d_t - b_t \\ d_t + b_t & a_t - c_t \end{pmatrix}$$

- Optimization:

- Fix the local frame on each triangle: Second-Order Cone Programming (SOCP);
 - Update local frame to let $b_t = 0$.



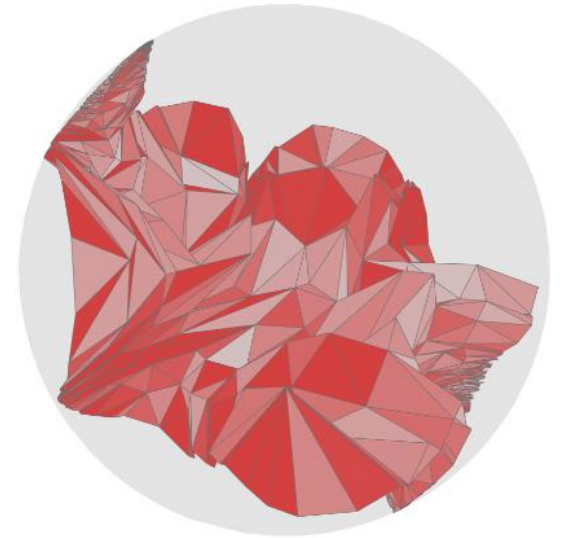
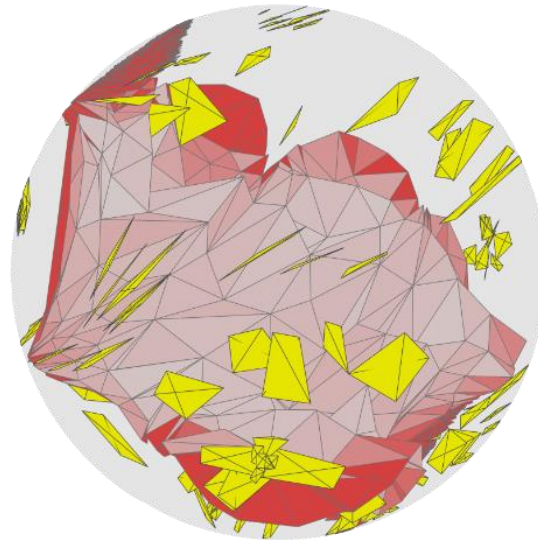
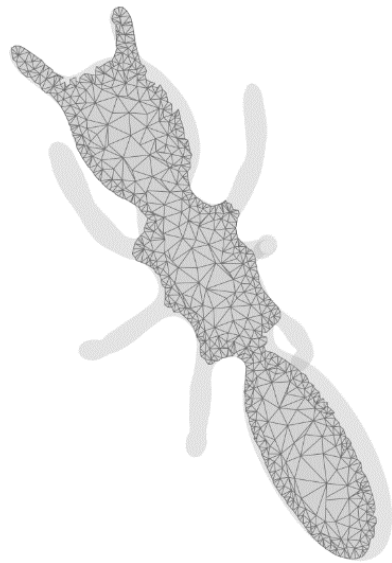
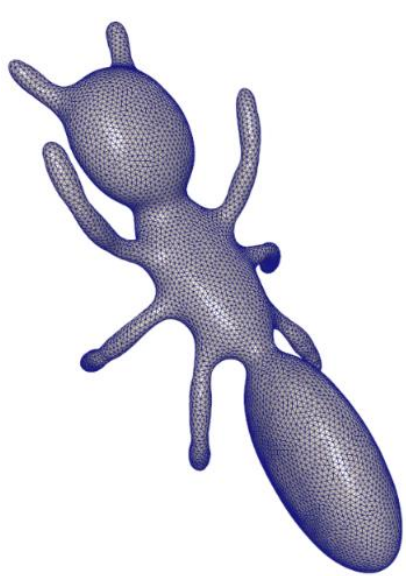
$\delta_{max}^{con} = 26.98$

Time: 4.03s

1. How to choose K?
2. The speed is slow.

Local/global formulation

- Practical Foldover-Free Volumetric Mapping Construction (PG 2019)



Source tetrahedral mesh

Initial
volumetric map

Foldover-free
volumetric map

Input

Output

- ◆ Signed singular value decomposition

$$J_i(\mathbf{u}) = U_i S_i V_i^T, S_i = \text{diag}(\sigma_{i,1}, \sigma_{i,2}, \sigma_{i,3})$$

$$\sigma_{i,1} \geq \sigma_{i,2} \geq |\sigma_{i,3}|.$$

- ◆ Foldover-free constraints

$$\det J_i(\mathbf{u}) > 0, i = 1, \dots, N \Leftrightarrow \sigma_{i,3} > 0$$

- ◆ Conformal distortion

$$\tau(J_i(\mathbf{u})) = \sigma_{i,1}/\sigma_{i,3}$$

- ◆ Bounded conformal distortion constraints

$$1 \leq \tau(J_i(\mathbf{u})) \leq K$$

Foldover-free constraints

$$\det J_i(\mathbf{u}) > 0$$

$$\sigma_{i,3} > 0, \tau(J_i) = \sigma_{i,1}/\sigma_{i,3}$$

$$\sigma_{i,1} \geq \sigma_{i,2} \geq |\sigma_{i,3}|$$

$$K = \max_{i=1,\dots,N} \tau(J_i)$$

Bounded conformal distortion constraints

$$1 \leq \tau(J_i(\mathbf{u}))$$

$$\tau(J_i(\mathbf{u})) \leq K$$

$$\tau(J_i) \geq 1, \sigma_{i,3} > 0, \sigma_{i,3} > 0$$

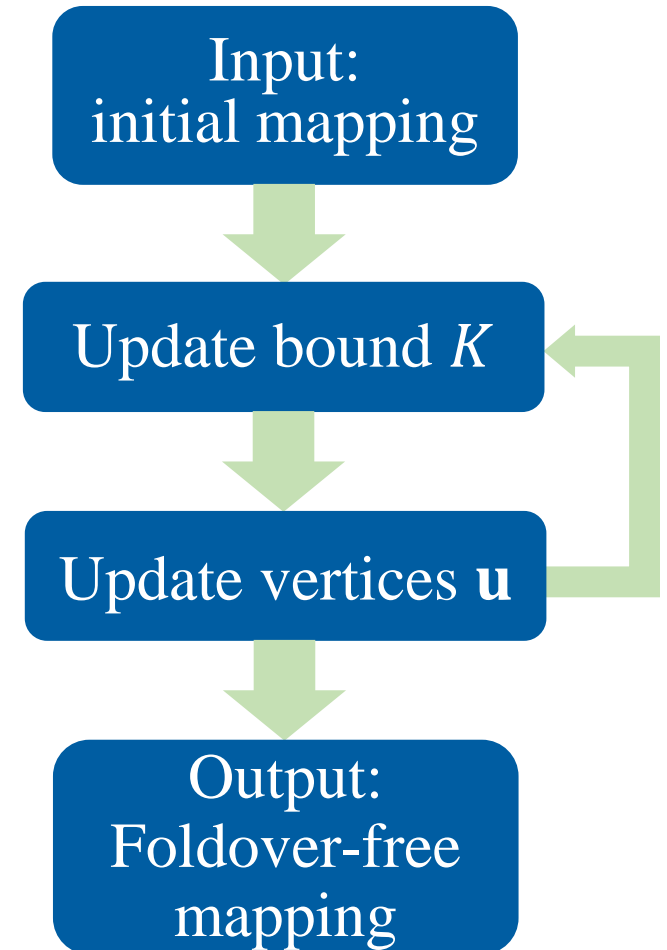


It is difficult to satisfy the constraints!

$$1 \leq \tau(J_i(\mathbf{u})) \leq K$$

Alternatively solving K and \mathbf{u}

- Update K : generate a conformal distortion bound;
- Update \mathbf{u} : project the mapping into the bounded distortion space;
- If there are foldovers, go to Step 1;



- Monotone projection

$$\min_{\mathbf{u}} E_d = \sum_{i=1, \dots, N} \|J_i(\mathbf{u}) - H_i\|_F^2,$$

$$s. t. \quad H_i \in \mathcal{H}_i, i = 1, \dots, N,$$

$$A\mathbf{u} = b.$$

$\mathcal{H}_i = \{H_i | 1 \leq \tau(H_i) \leq K\}$: bounded conformal distortion space.

Local-global solver

- Local-global solver

Local step

Fix \mathbf{u} and J_i , solve H_i

$$\min_{\mathbf{u}} E_d = \sum_{i=1, \dots, N} \|J_i(\mathbf{u}) - H_i\|_F^2,$$

$$s. t. \quad H_i \in \mathcal{H}_i, i = 1, \dots, N,$$

Global step

Fix H_i , solve \mathbf{u}

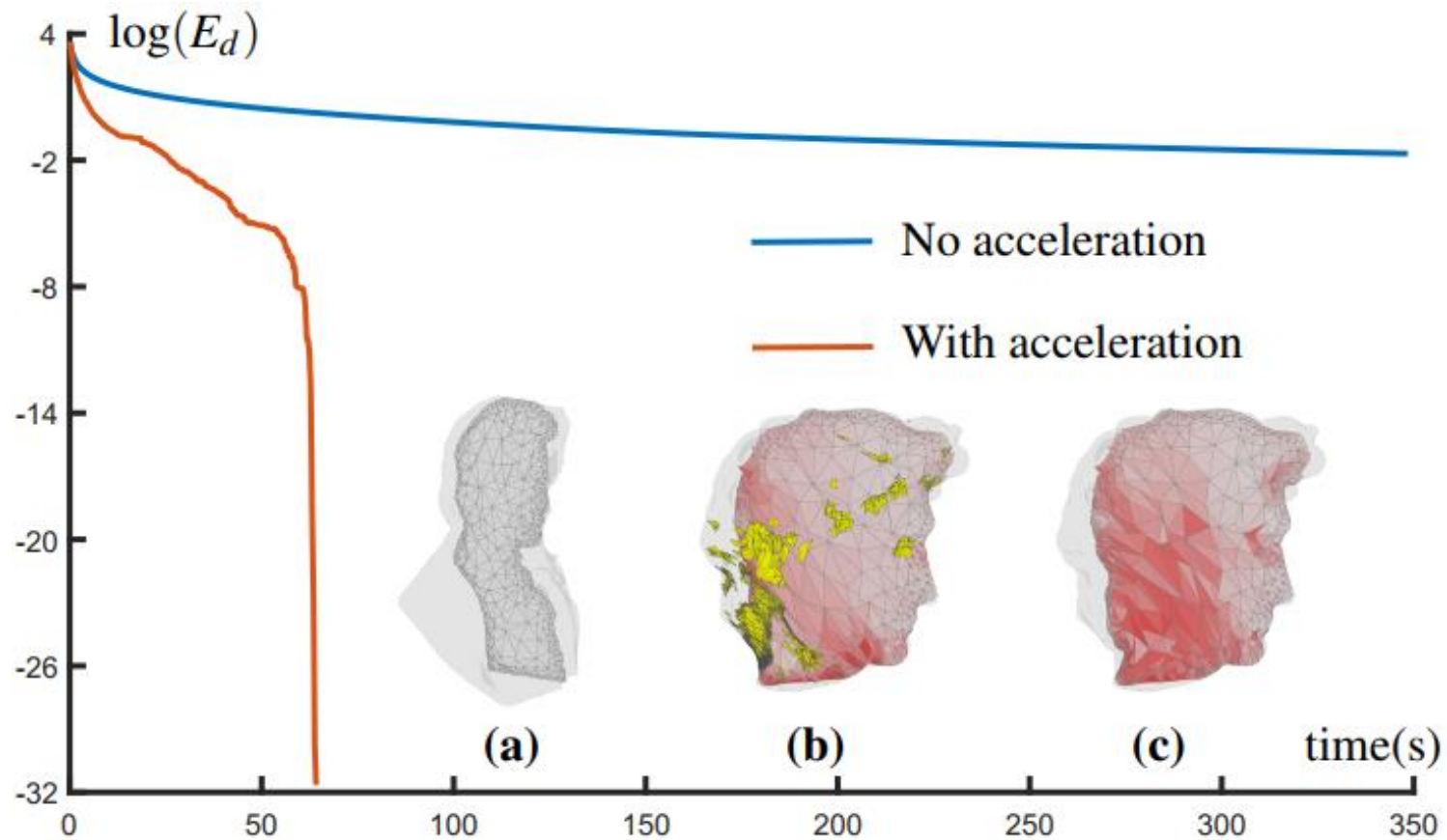
$$\min_{\mathbf{u}} E_d = \sum_{i=1, \dots, N} \|J_i(\mathbf{u}) - H_i\|_F^2,$$

$$s. t. \quad A\mathbf{u} = b$$

Very slow convergence...

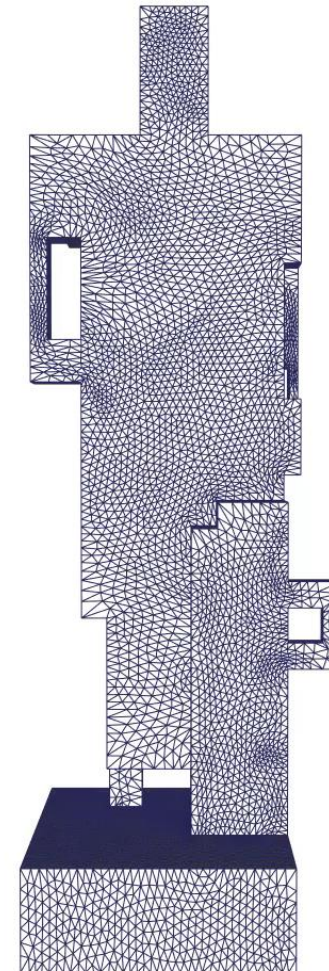
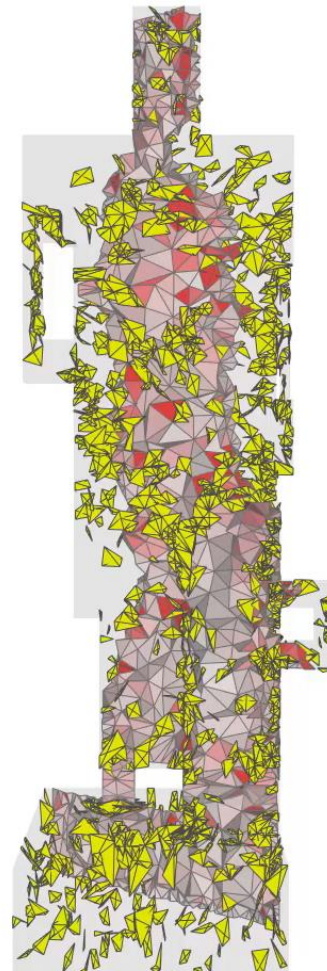
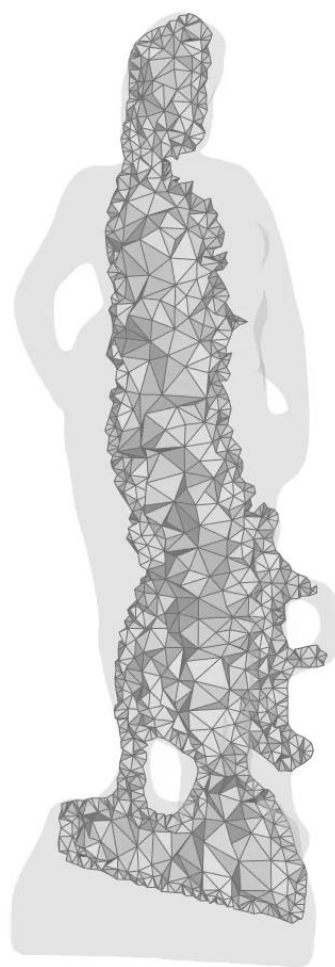
Update vertices \mathbf{u}

- Anderson acceleration method [Peng et al. 2018]



Why update bound K ?

Projection cannot
eliminate all foldovers



round:1 iter:0 foldovers:3274

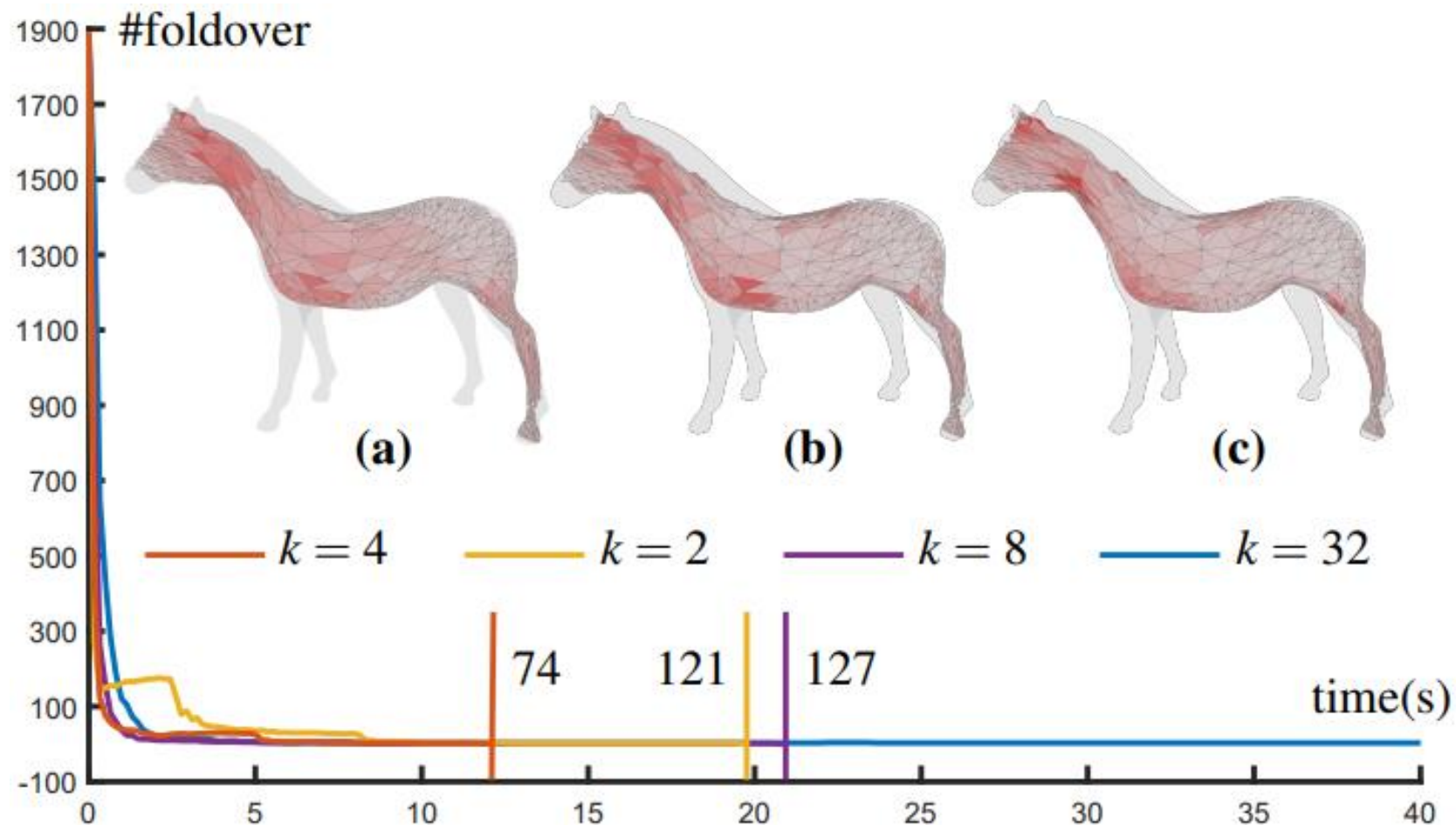
Update bound K

Bound generation

$$K^{new} = \beta K$$

$$\beta = 2$$

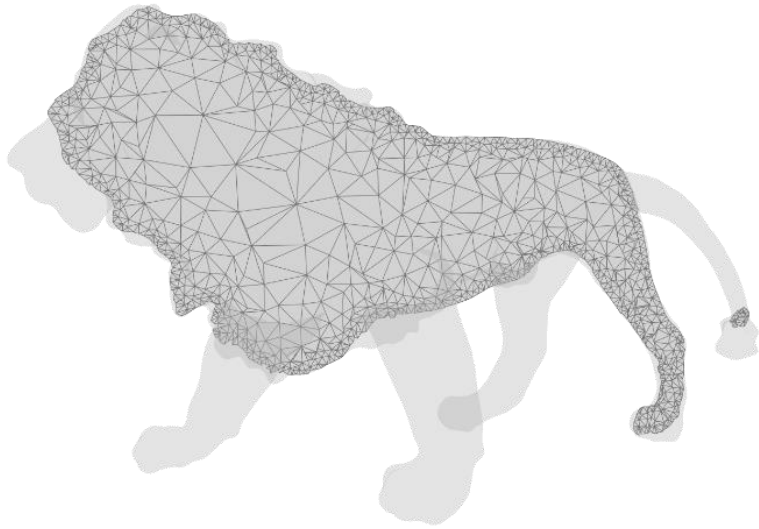
initialize $K = 4$



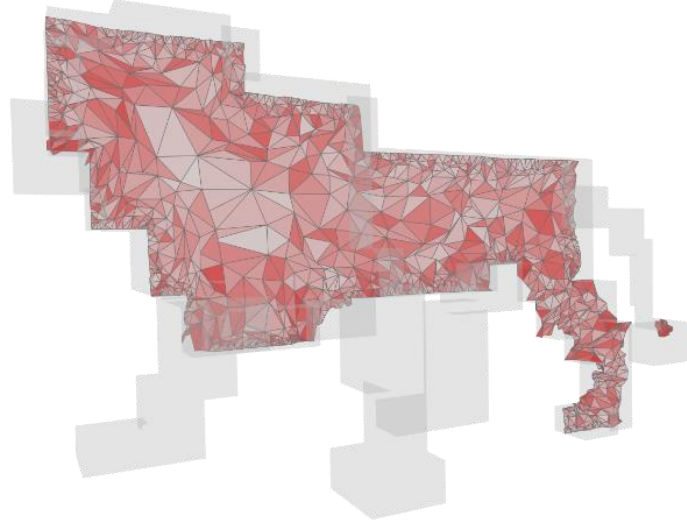
- Apply a maintenance-based method

Before post-optimization

After post-optimization



Source tetrahedral mesh



Average / maximum
conformal distortion:

2.72 / 107.10



Average / maximum
conformal distortion:

2.08 / 22.61

Elimination Process :

alternates in each round:

- generate conformal distortion bound
- try to project the mapping into the bounded distortion space

Outlines

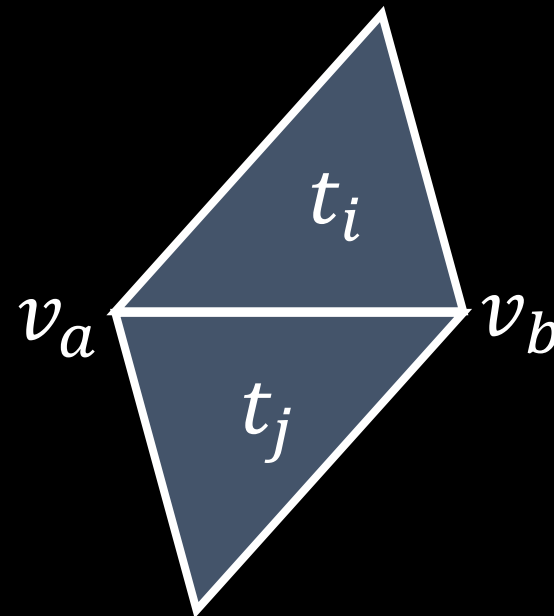
- Introduction
- Maintenance-based methods
- Bounded distortion methods
- Representation-based method
 - Computing inversion-free mappings by simplex assembly

Affine transformation

Key observation: the parameter space is a 2D triangulation, uniquely defined by all the **AFFINE TRANSFORMATIONS** on the triangles.

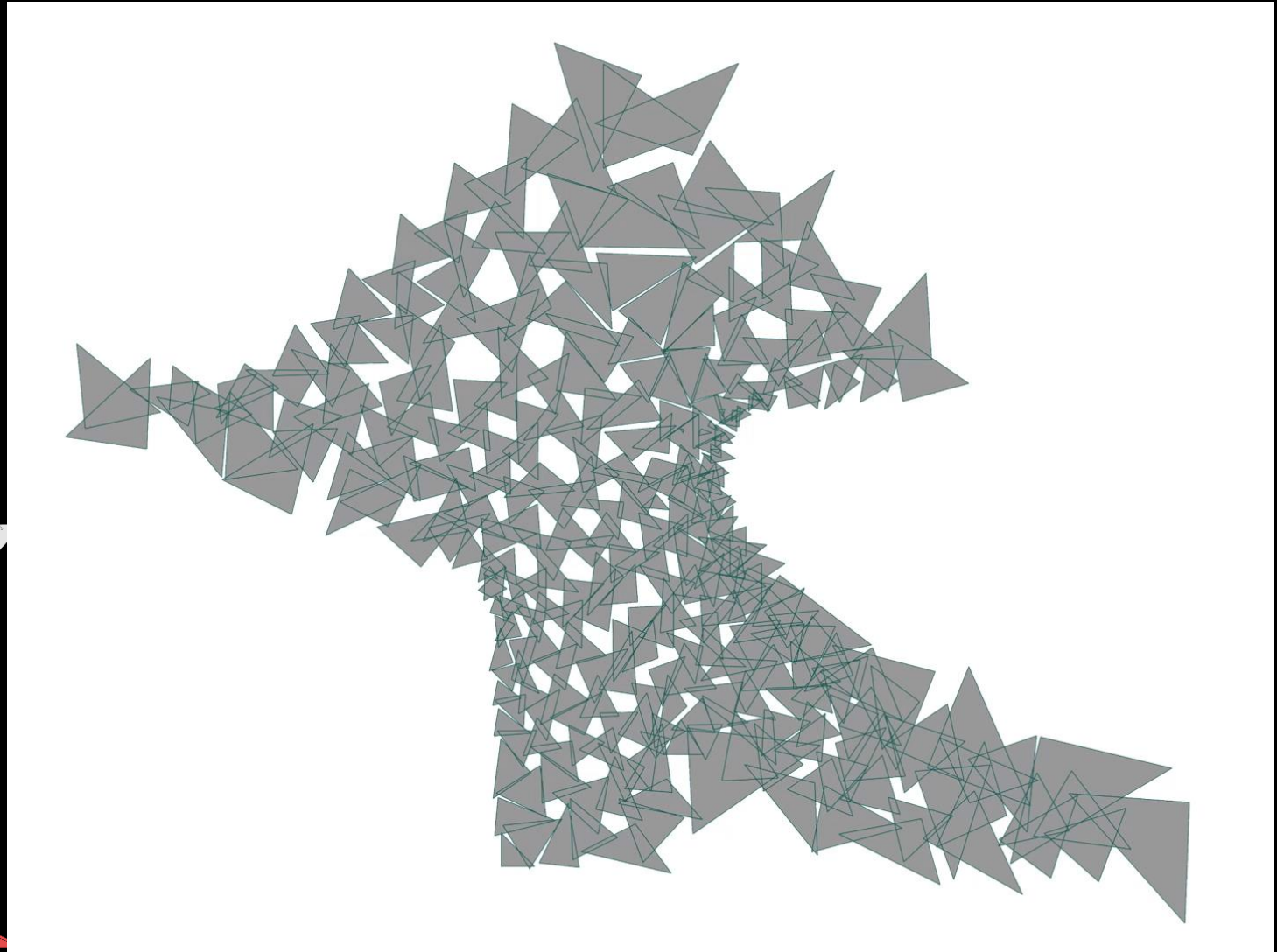
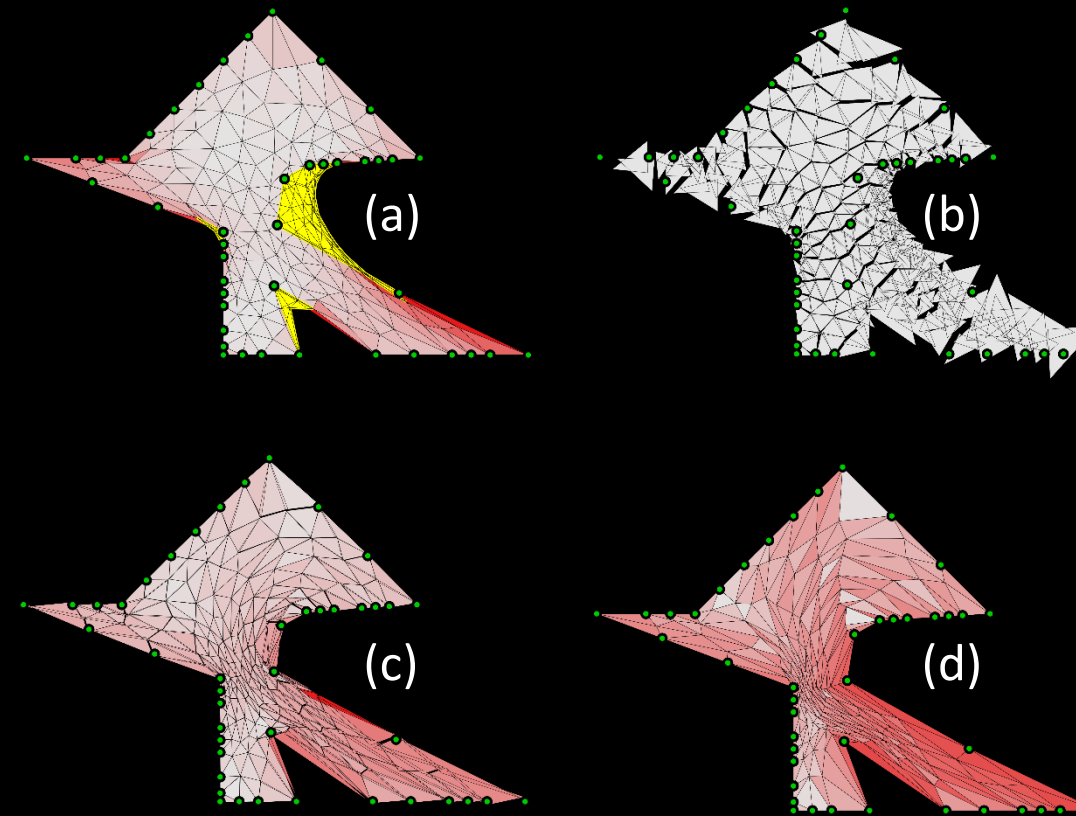
Edge assembly constraints:

$$A_i(v_a - v_b) = A_j(v_a - v_b)$$



Key idea

- disassembly + assembly
 - Treat affine transformation as variables
 - Unconstrained optimization



Unconstrained optimization problem

Disassembly: project initial A_i^0 into feasible space.

Assembly: unconstrained optimization.

$E_{assembly}$: summation of squares of edge, assembly constraints.

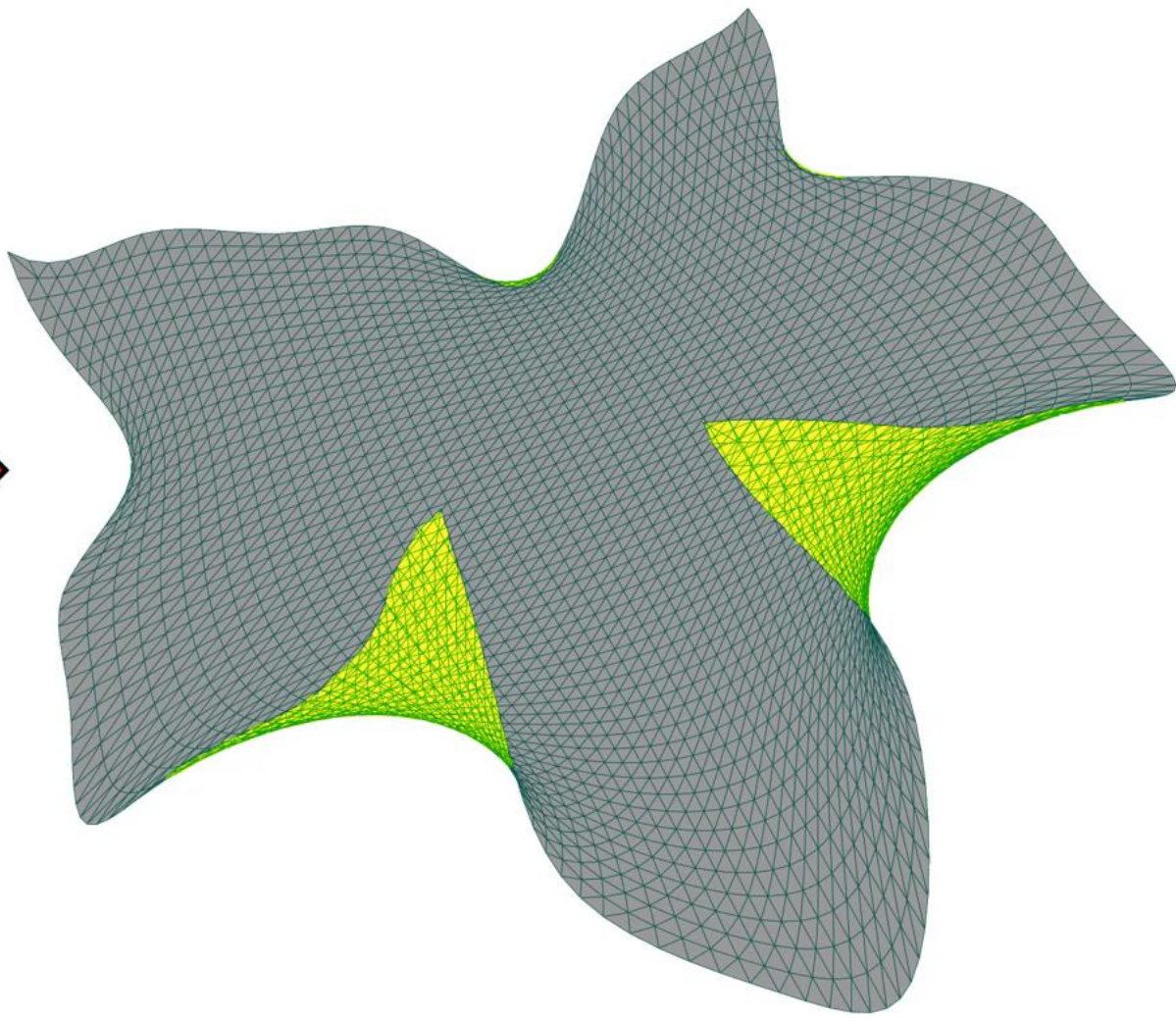
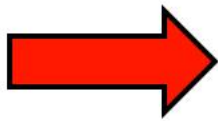
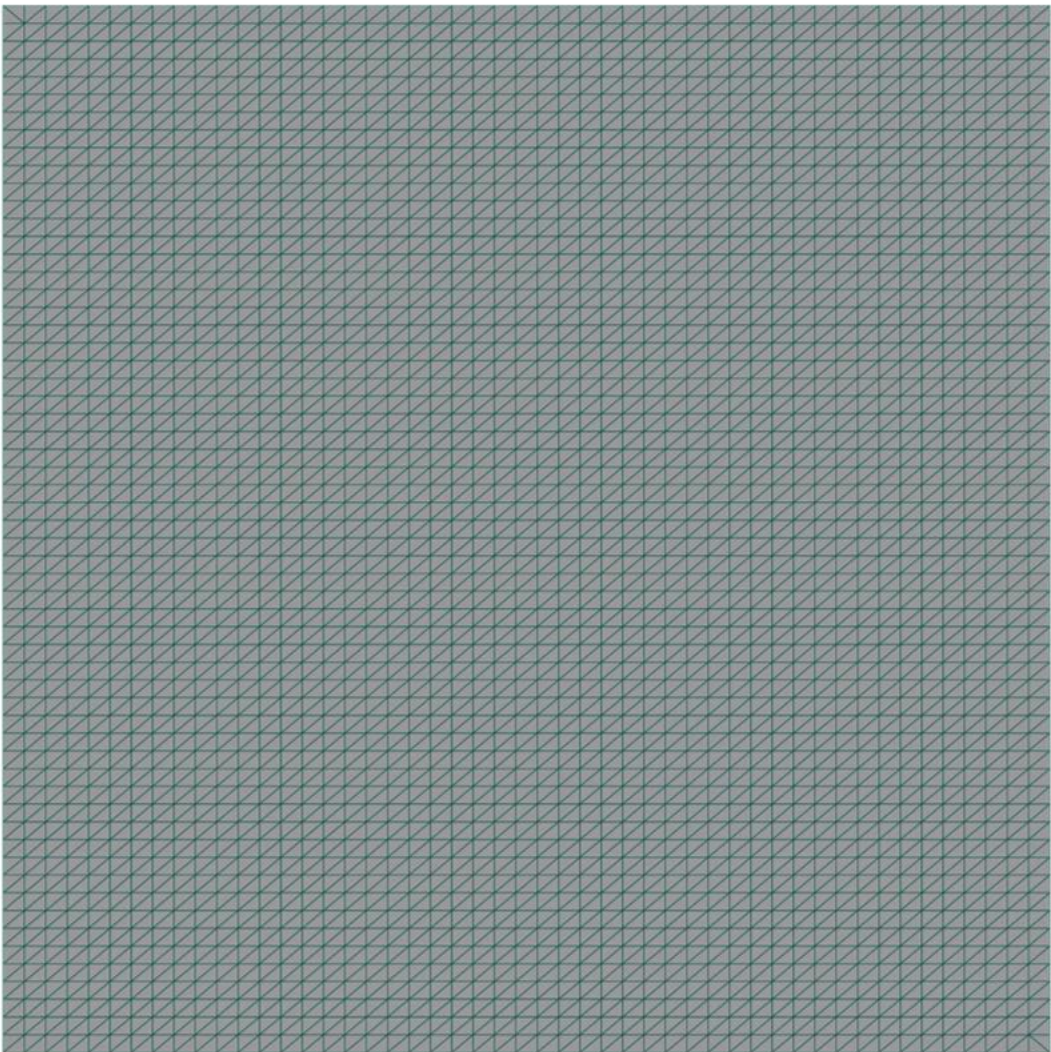
$$\min_{\substack{A_1, \dots, A_N \\ T_1, \dots, T_N}} \lambda E_{assembly} + E_C + \mu E_m$$

E_C : Barrier function on distortion

E_m : users' designed energy

$$\lambda_{k+1} = \min \left(\lambda_{\min} \cdot \max \left(\frac{E_{C,k} + \mu E_{m,k}}{E_{assembly,k}}, 1 \right), \lambda_{\max} \right)$$

1. $E_{assembly}$ dominates the energy, approach zero;
2. λ_{\max} : avoid large distortion.



Spherical Parametrizations

Xiao-Ming Fu

Outlines

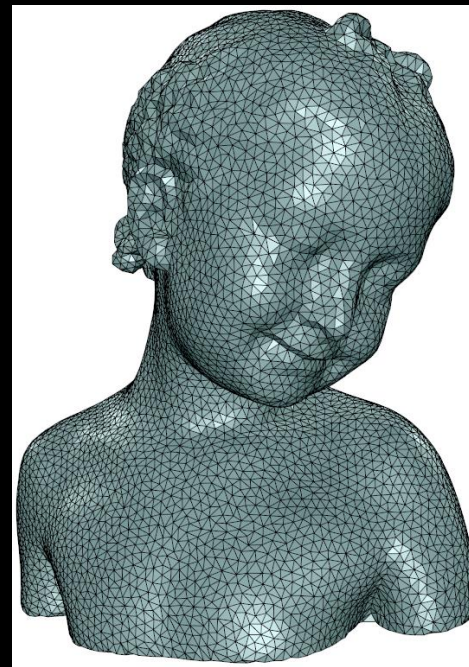
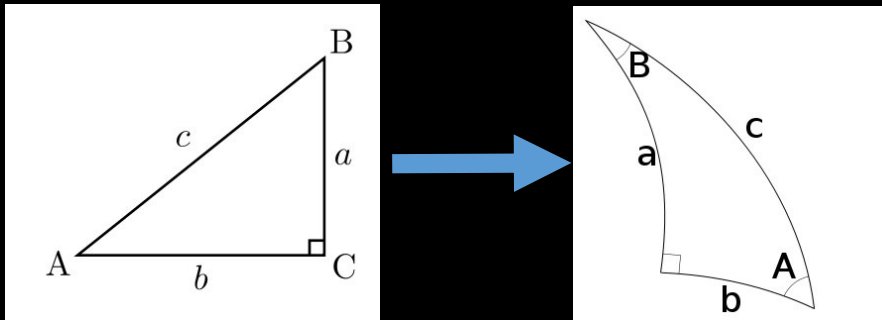
- Definition & Applications
- Hierarchical method
 - Paper: Advanced Hierarchical Spherical Parameterizations
- Two hemispheres
- Curvature flow

Outlines

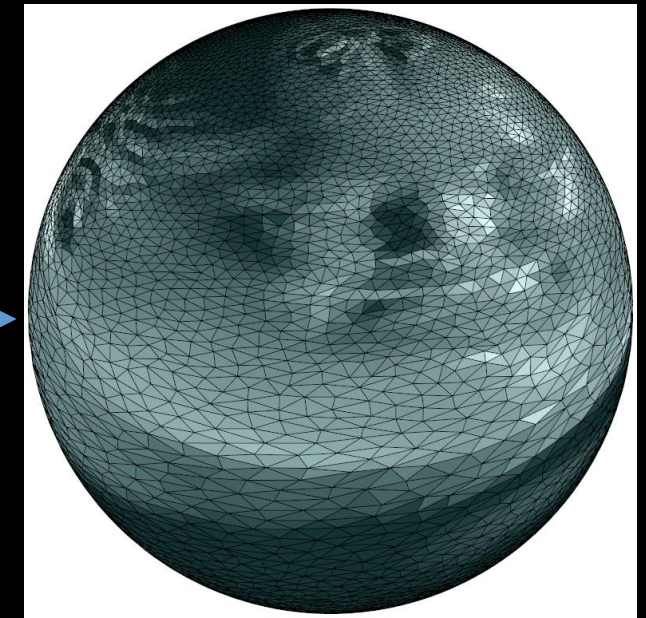
- **Definition & Applications**
- Hierarchical method
 - Paper: Advanced Hierarchical Spherical Parameterizations
- Two hemispheres
- Curvature flow

Spherical Parametrizations

- Homeomorphic mapping between a genus-0 closed surface to a sphere
- If the surface is represented by a triangle mesh, each triangle is projected to a spherical triangle



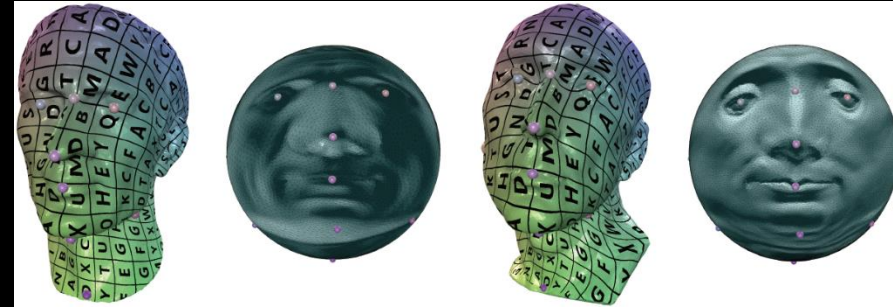
Genus-0 surface



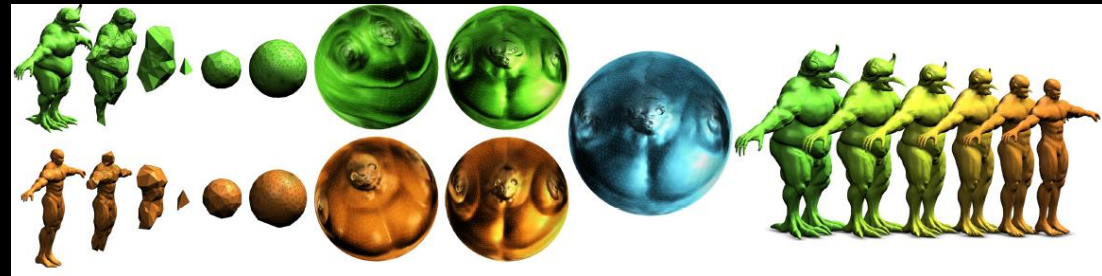
Sphere

Applications

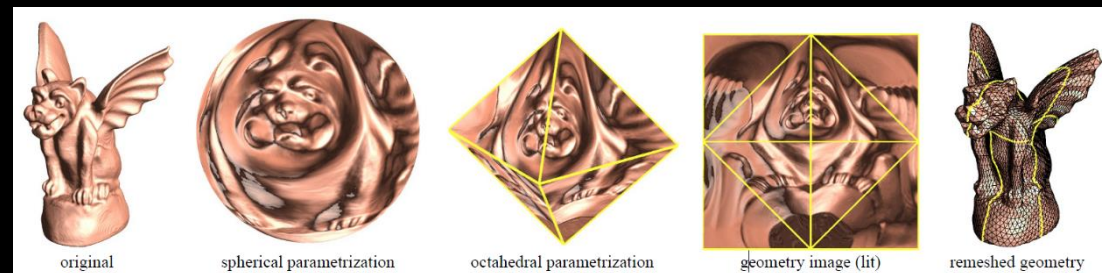
- Correspondence



- Morphing



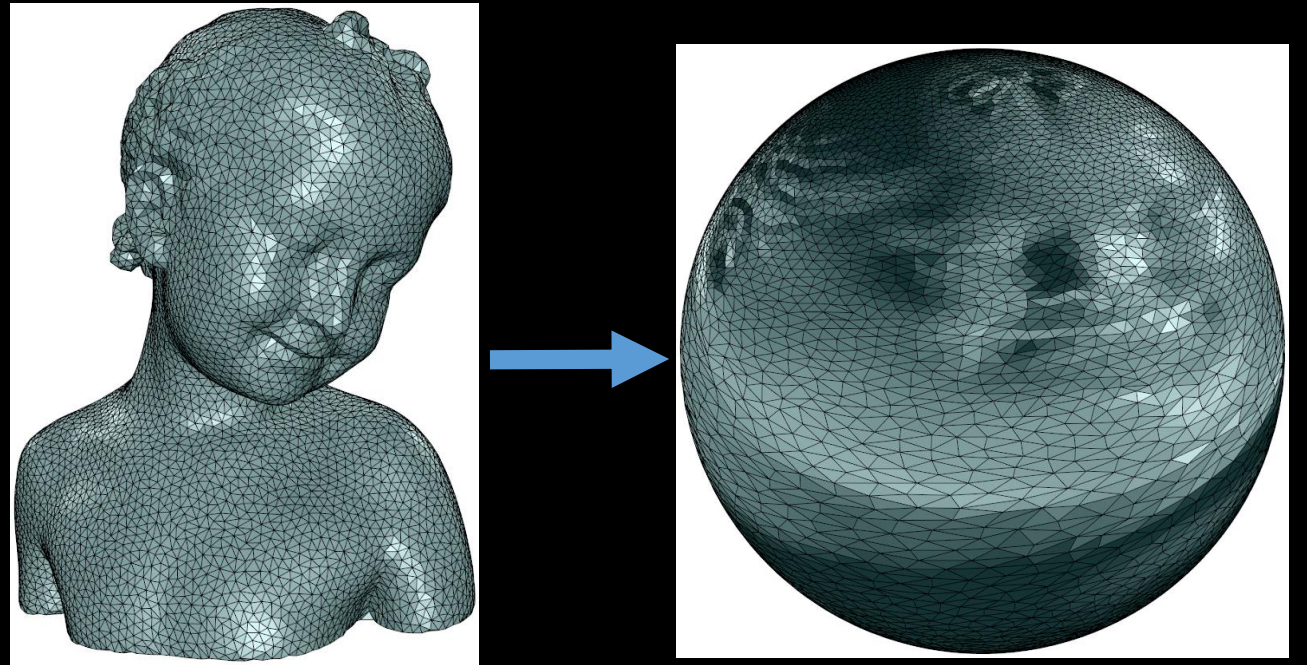
- Remeshing



•

Constraints

- Spherical constraints
- Bijective constraints
 - No foldover
- Low distortion



$$x^2 + y^2 + z^2 = r^2$$

Non-linear, non-convex

Challenge

- No Tutte's embedding method
- Non-linear, non-convex optimization problem.

Very challenging!!!

Outlines

- Definition & Applications
- Hierarchical method
 - Paper: Advanced Hierarchical Spherical Parameterizations
- Two hemispheres
- Curvature flow

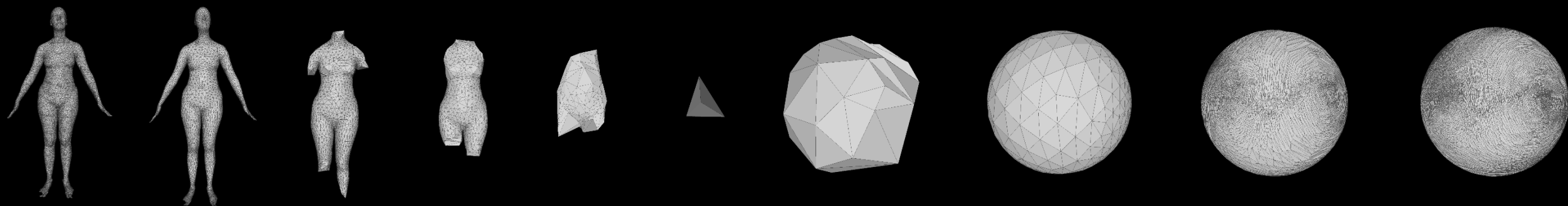
Hierarchical scheme pipeline

Input:
A triangle mesh

Decimation

Refinement

Output:
A triangular sphere



15477

5869

2000

1000

500

4

50

500

8000

15477

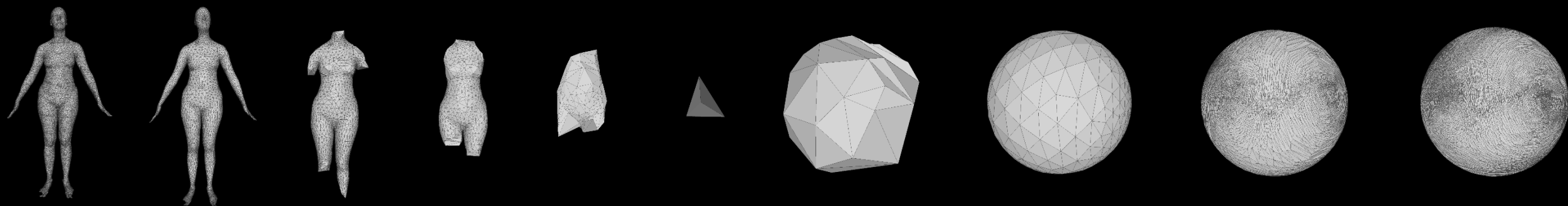
Hierarchical scheme pipeline

Input:
A triangle mesh

Decimation

Refinement

Output:
A triangular sphere



15477

5869

2000

1000

500

4

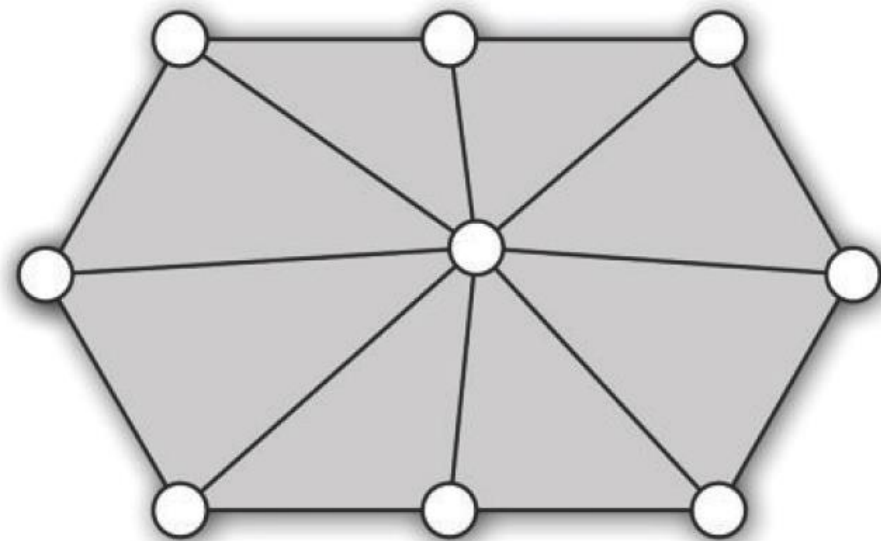
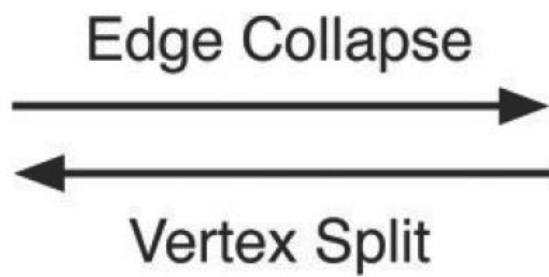
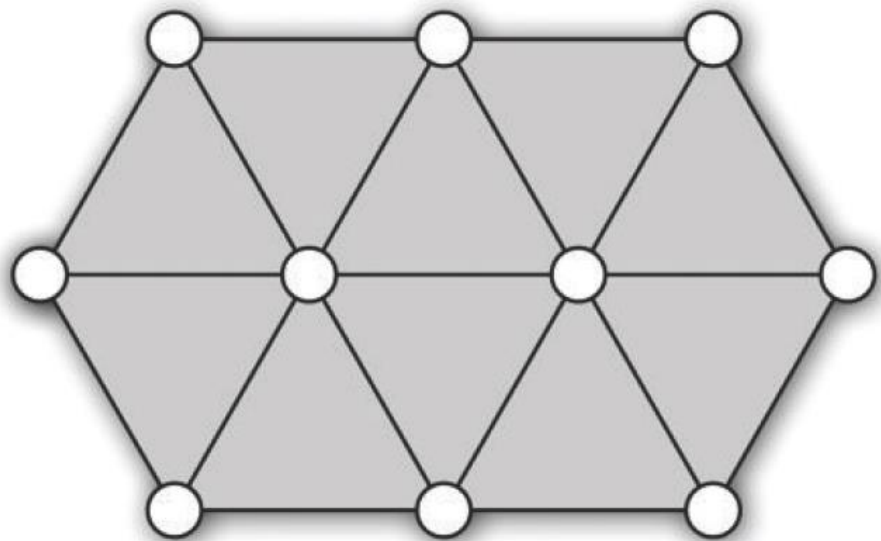
50

500

8000

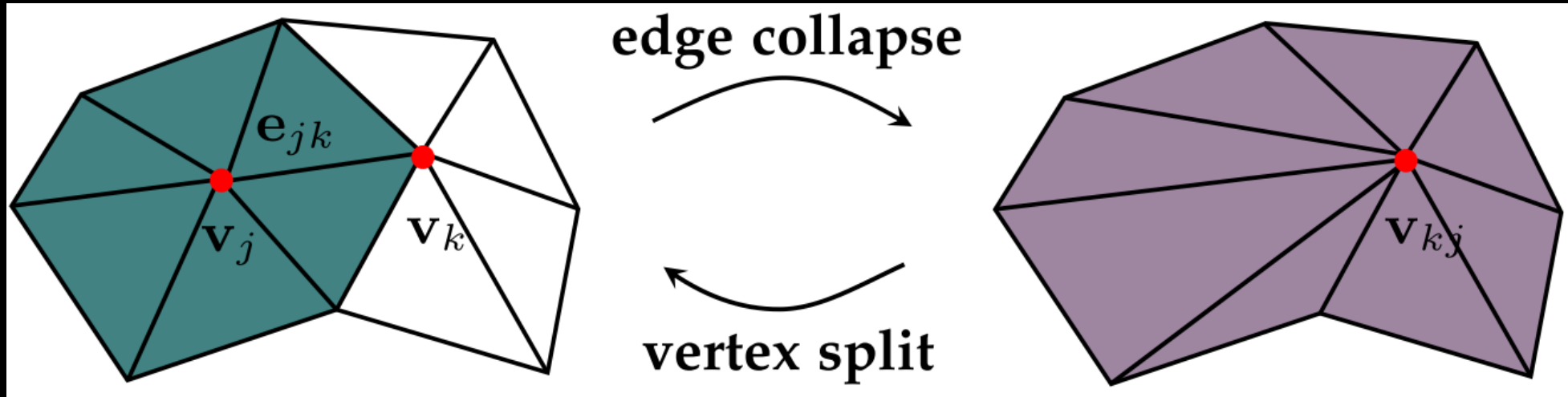
15477

Decimation



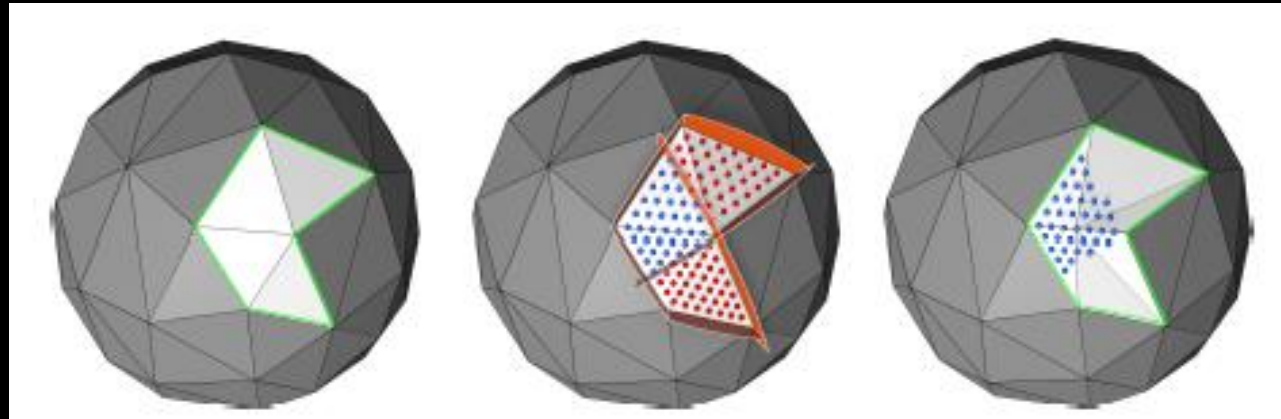
Curvature error metric (CEM)

- $E_{jk}^C = \frac{g(V_j)}{d_e(V_{kj}) \cdot \rho(V_{kj})}$
 - $g(V_j)$ is the Gaussian curvature at V_j
 - $d_e(V_{kj}) = e^{(d(V_{kj})-6)^2}$, $d(V_{kj})$ is the valence of V_{kj}
 - $\rho(V_{kj}) = \sum_{f \in \Omega(V_{kj})} \frac{c_r(f)}{2 \times ir(f)}$



Refinement

- Insert a new vertex on sphere

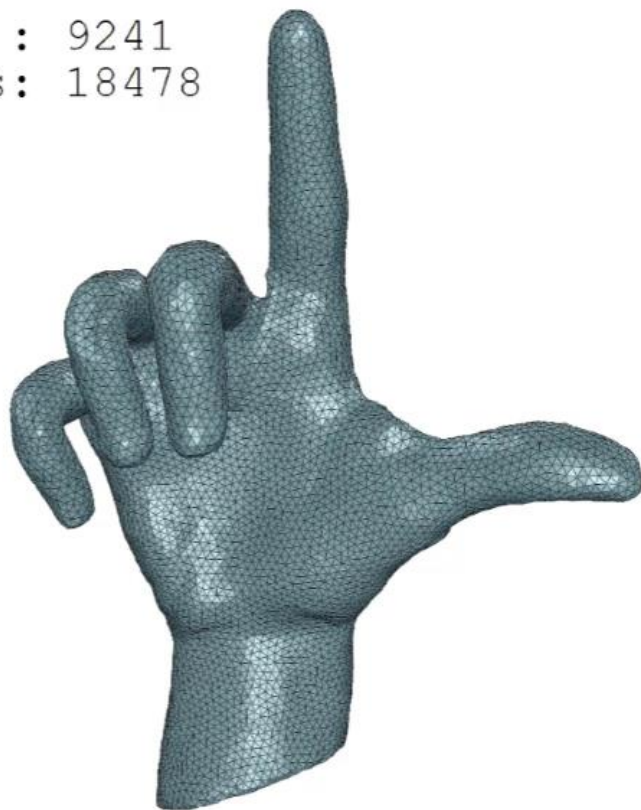


Such sphere kernel is non-empty !

Paper: Advanced Hierarchical Spherical Parameterizations

Hand Decimation

```
#vertices : 9241  
#triangles: 18478
```

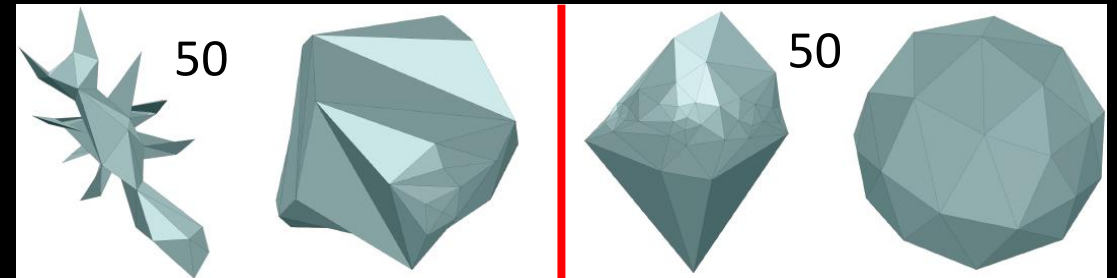


Flat-to-extrusive decimation strategy

Once an approximate surface has been spherically parameterized, the details of the input surface can be refined easily.



The shape contains highly curved areas at the beginning of refinement, it is hard to make the vertices evenly.

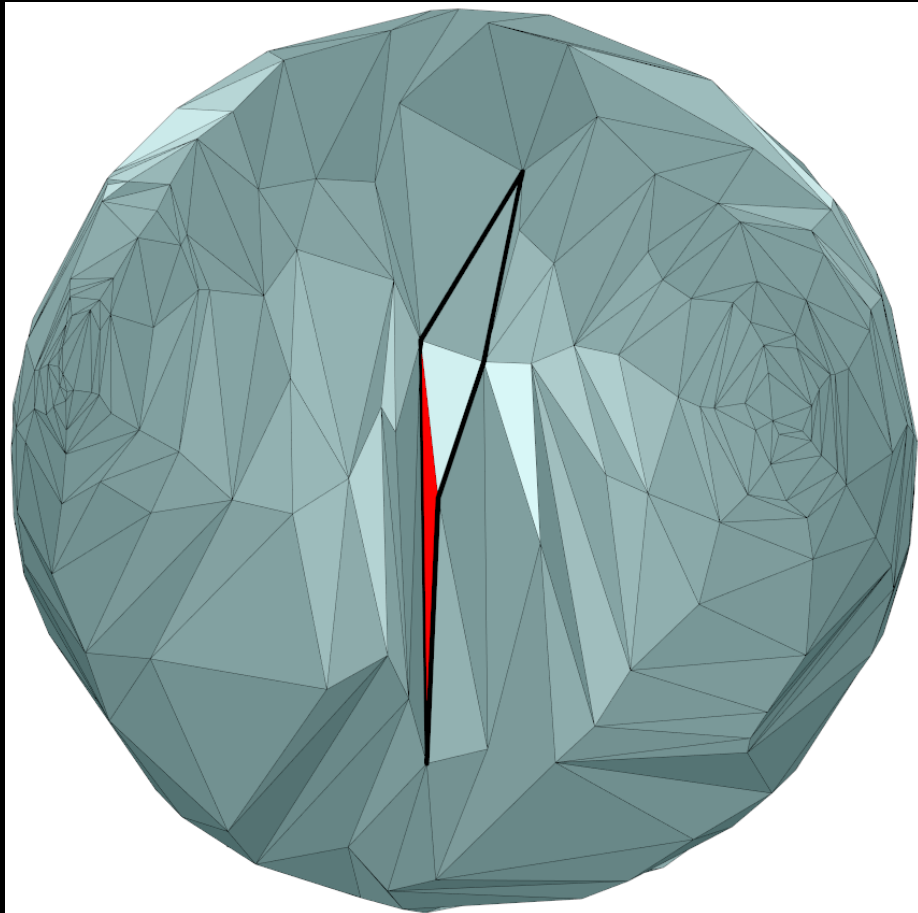


A flat-to-extrusive mesh decimation scheme, first simplifies the **flat** regions by QEM and then the **extrusive** regions by CEM.



Depression illustration

- Long and thin triangles (**red**) may block the next vertex insertion.



Make the vertices evenly distribute over the sphere as much as possible.

It is easier to insert the later vertices.

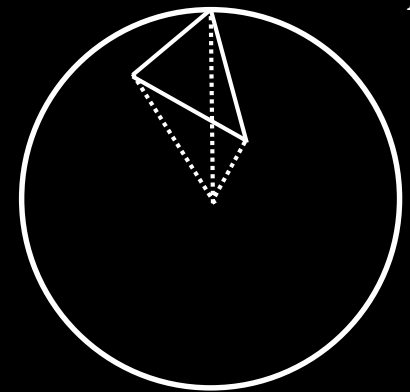
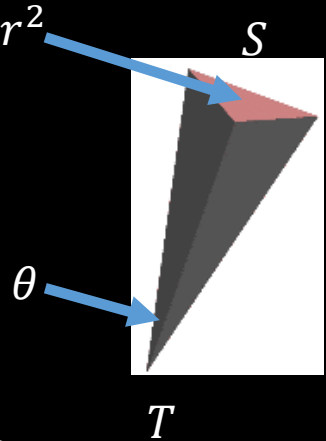
Flexible group mesh refinement

- Group insertion
 - Distortion control: iteratively insert vertices until the maximal distortion exceeds a threshold.
 - Global optimization: reduce distortion.
- Former methods optimize the vertices after inserting fixed number of vertices.
 - [Praun and Hoppe 2003], [Peng et al.2016], [Wan et al. 2012]
- This group scheme is much more robust and efficient.

Global optimization

- We want to make the vertices evenly distribute on the sphere
 - The ideal tetrahedron T
 - S is a equilateral triangle
 - The area of spherical triangle S^t is decided by the current face number
 - Use *spherical excess* to compute dihedral angle
 - Use *Cosing Law* to compute the angle θ
 - Optimizing the tetrahedral mesh
 - Tetrahedron: formed by mesh triangles and coordinate origin
 - Inexact block coordinate descent
- Only the topology information of origin mesh is used during the global optimization.

$$n(f) * Area(S^t) = 4\pi r^2$$



Global optimization energy

- 3D AMIPS energy:

$$E_i^{con} = \frac{1}{8} (\|J_i\|_F^2 \cdot \|J_i^{-1}\|_F^2 - 1)$$

$$E_i^{vol} = \frac{1}{2} (\det J_i + (\det J_i)^{-1})$$

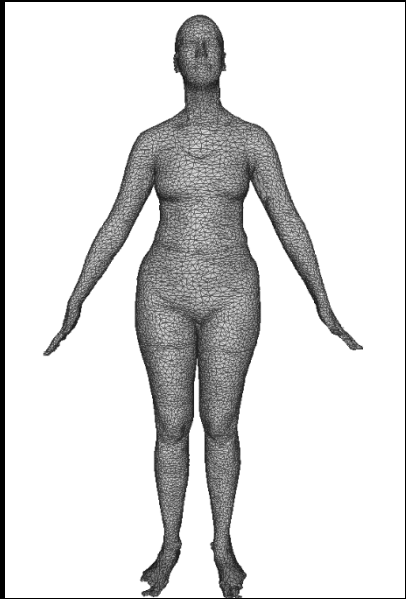
$$E_i^{iso} = \frac{1}{2} (E_i^{con} + E_i^{vol})$$

$$E_i^* = \exp\left((E_i^{iso})^s\right)$$

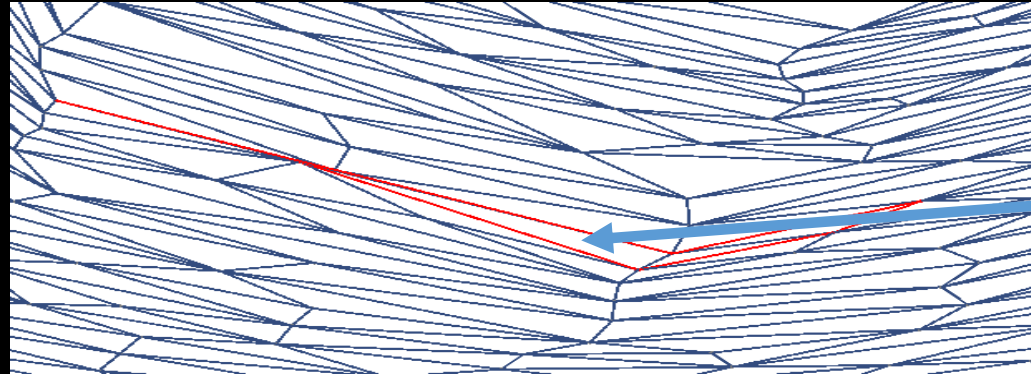
- $s = \frac{\ln \tau}{\ln E_{\max}^m}, \quad (E_i^{iso})^s < \tau$

- Volume-based energy significantly improve the robustness of our refinement process.

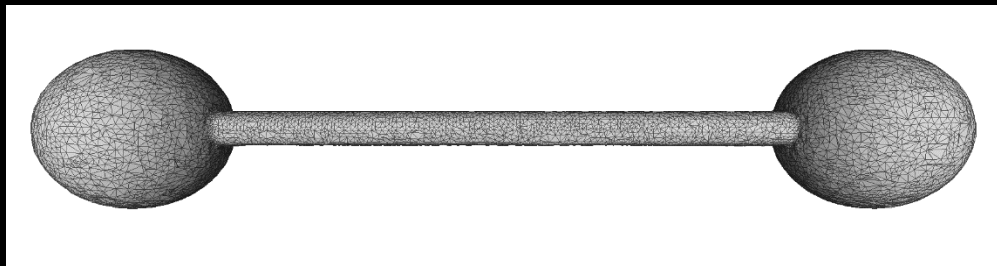
Triangle-based failure cases



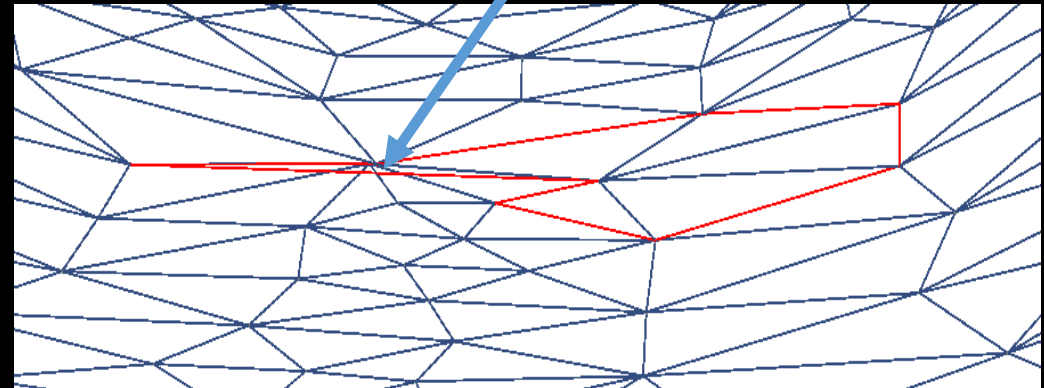
15477



Narrow and long triangle
make it hard to insert
new vertex



87554



Post optimization

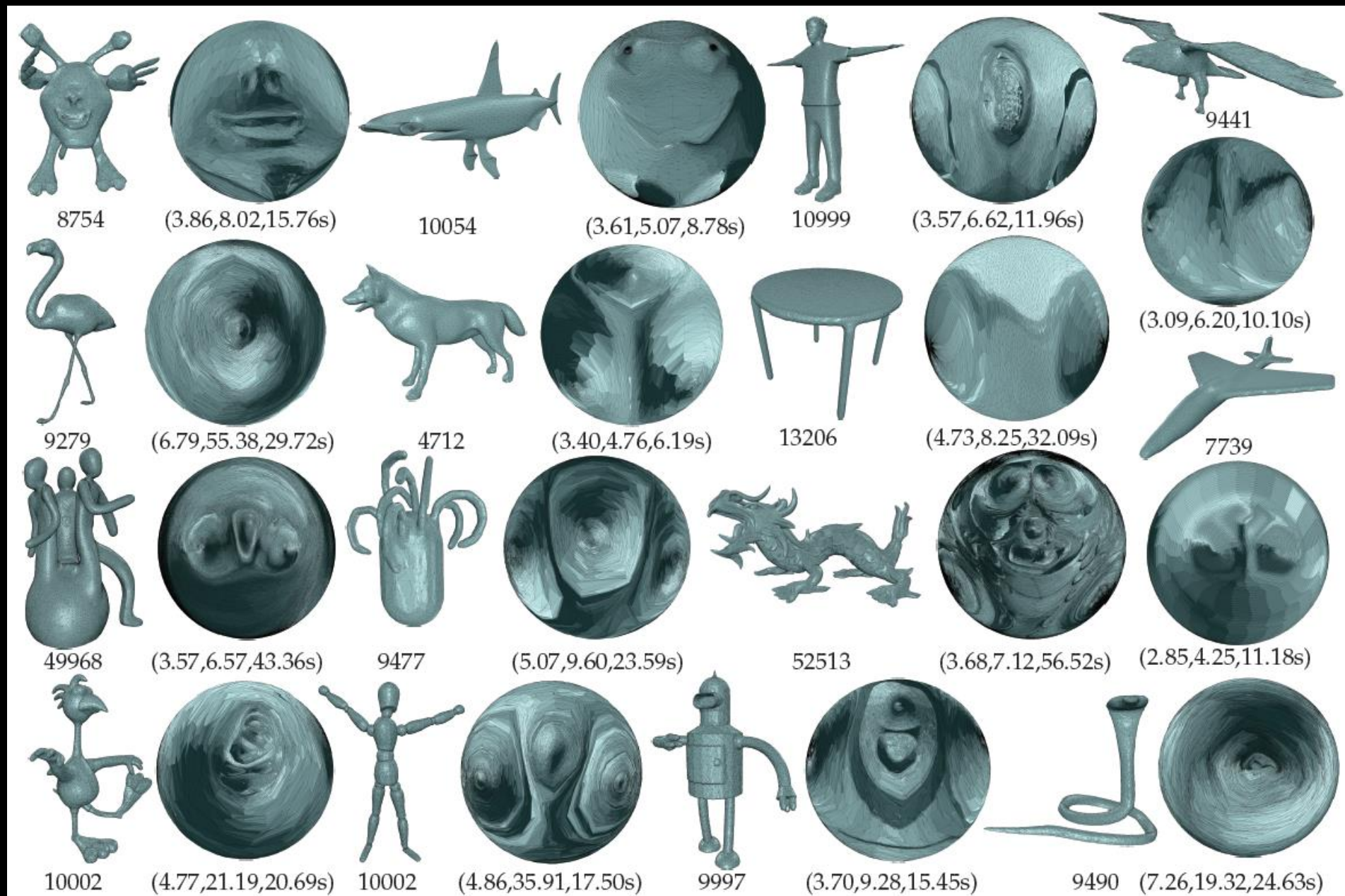
- Rigidly transfer each triangle t_i of original mesh to \tilde{t}_i on the sphere
- Form tetrahedron \tilde{T}_i by the origin and three vertices of \tilde{t}_i
- Use AMIPS energy to optimize T_i toward \tilde{T}_i



Before optimization

After optimization

Results



Outlines

- Definition & Applications
- Hierarchical method
 - Paper: Advanced Hierarchical Spherical Parameterizations
- **Two hemispheres**
- Curvature flow

Pipeline

1. Partition M into two balanced sub-meshes.
2. Embed each sub-mesh in a planar disk using the barycentric method with weights w .
3. Combine the embeddings of the two submeshes into one planar embedding using Moebius inversion.
4. Use inverse stereo projection to obtain a spherical embedding.

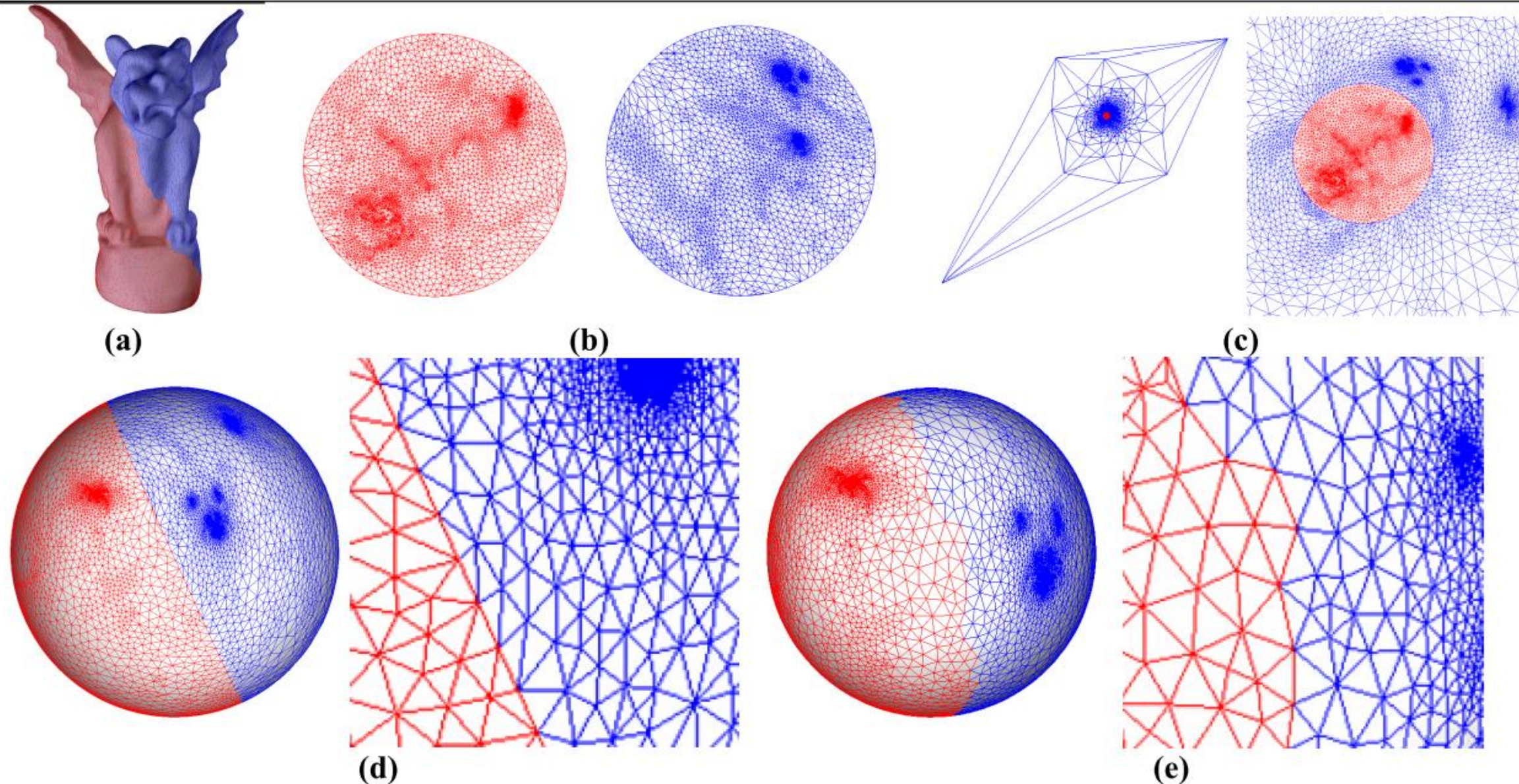


Figure 1. Stages in generating a spherical embedding for the gargoyle model (using uniform weights): (a) Partition into two sub-meshes using MeTiS. (b) Planar parameterization of the sub-meshes. (c) Combined planar embedding (with zoom); (d) Initial spherical parameterization generated by inverse stereo projection (with zoom) (f) final result after projected Gauss-Seidel and local Newton iterations (with zoom).

Improvements

- How to cut?
 - MeTiS graph partitioning package: obtain a balanced minimal vertex separator of the mesh graph.
- How to map the planar parameterizations onto the sphere?
 - Moebius inversion $f(z) = 1/\text{conj}(z)$
 - This maps the interior of the unit disk to its exterior.
 - Mapped to the unit sphere using the inverse stereo projection.

$$P(u, v) = \frac{1}{1 + u^2 + v^2} (2u, 2v, 1 - u^2 - v^2)$$

More papers

- Connectivity Shapes
- Spherical Parameterization Balancing Angle and Area Distortions, TVCG 2017

Outlines

- Definition & Applications
- Hierarchical method
 - Paper: Advanced Hierarchical Spherical Parameterizations
- Two hemispheres
- **Curvature flow**

Calabi flow

- Calabi energy: it is squared difference between current curvature vector and target curvature.

Mean curvature flow

Surface Mapping

Xiao-Ming Fu

Outlines

- Definition
- Application
- Algorithms
 - Common base domain
 - Parameterization-based method
 -

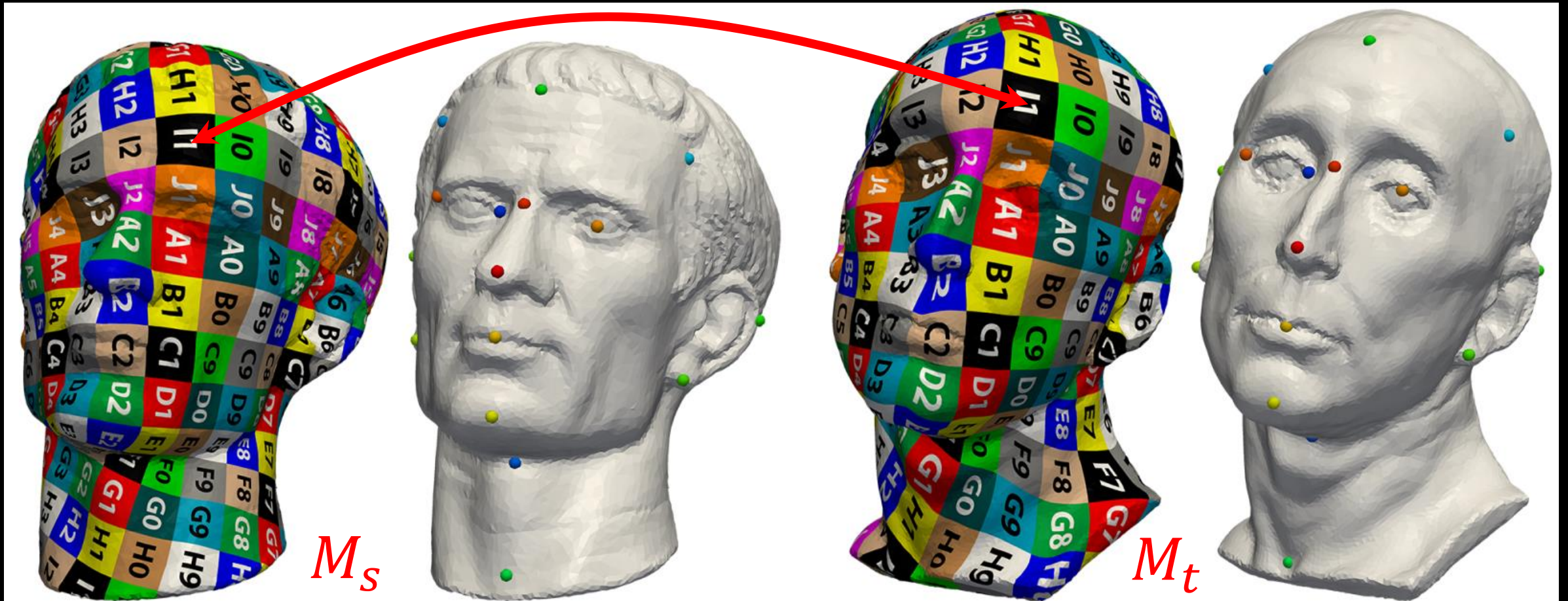
Outlines

- Definition
- Application
- Algorithms
 - Common base domain
 - Parameterization-based method
 -

Surface Mapping

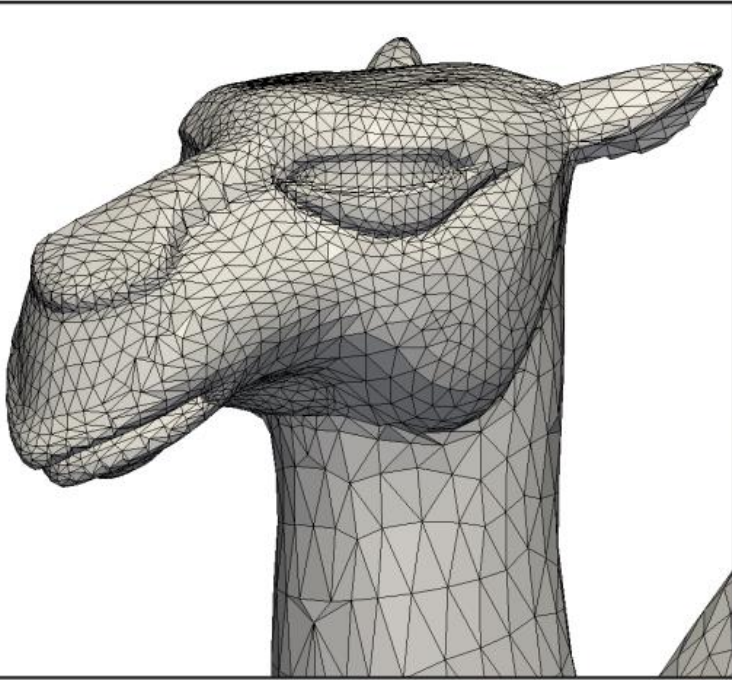
Inter-surface mapping, Cross parameterization

- A one-to-one mapping f between the two surfaces M_s and M_t

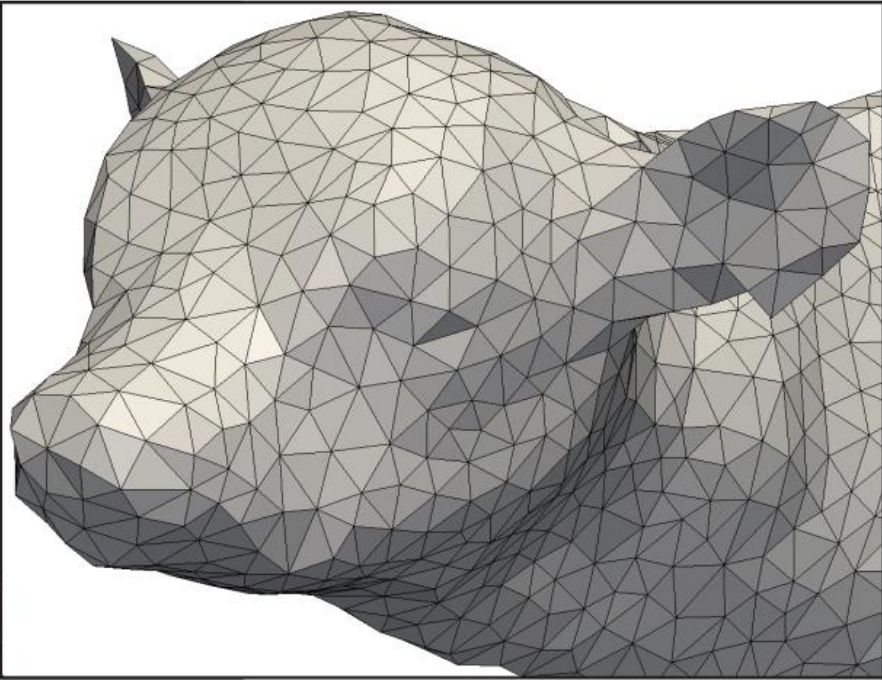


Compatible meshes

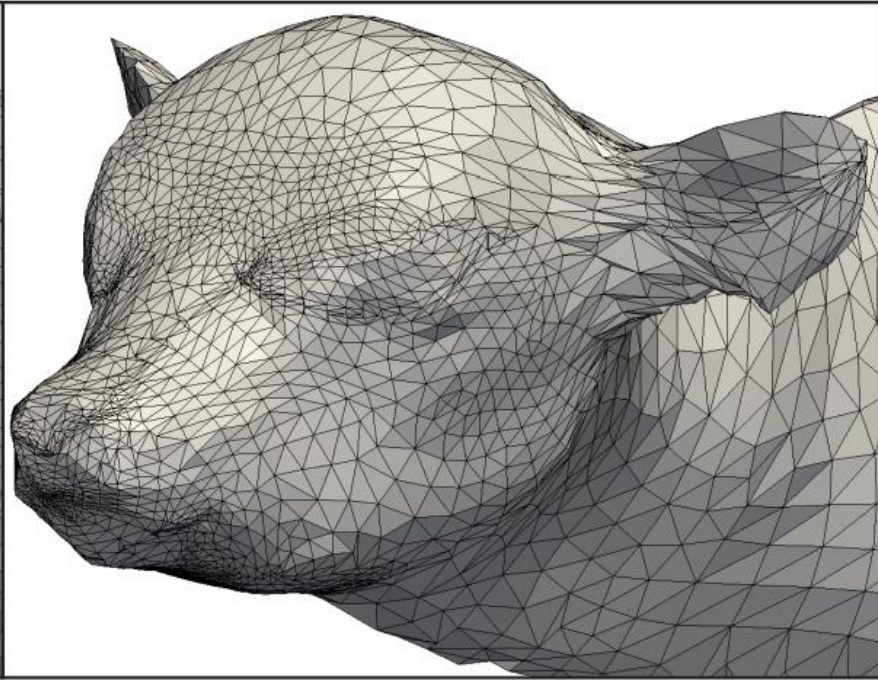
- Meshes with identical connectivity



M_s



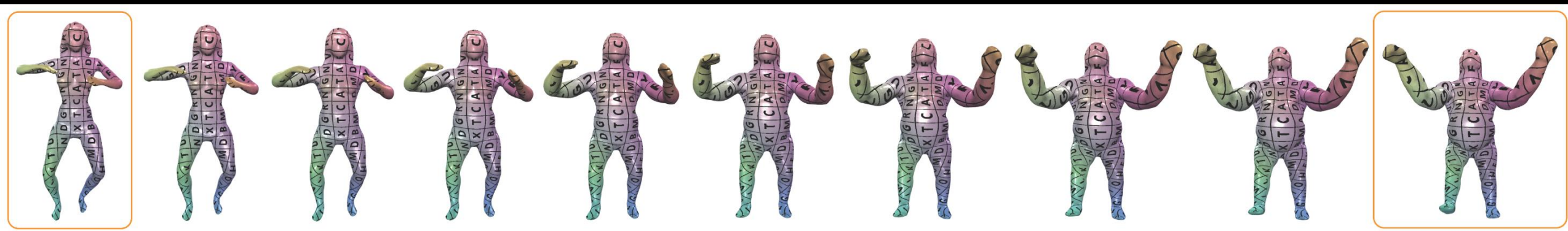
M_t



$\widehat{M}_t = f(M_s) \approx M_t$

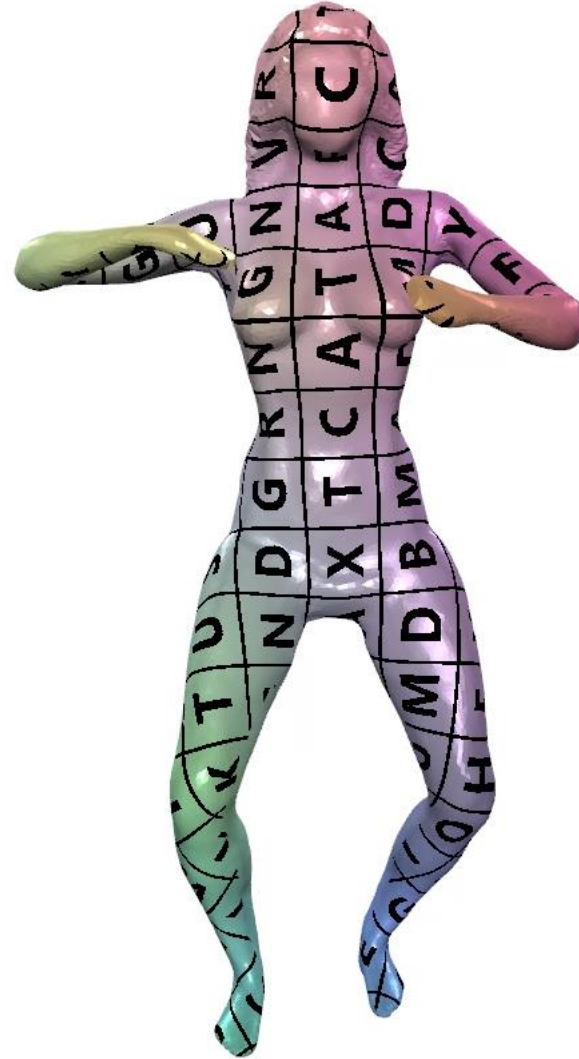
Applications

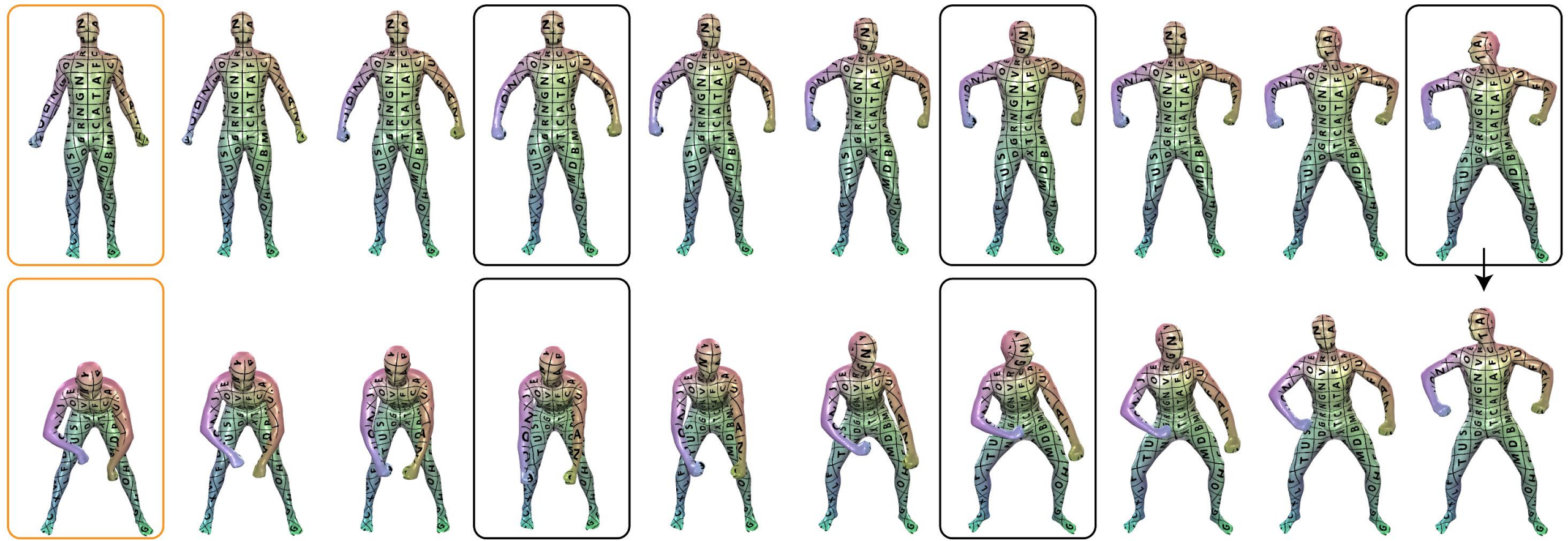
- **Morphing**
- Attribute transfer
-

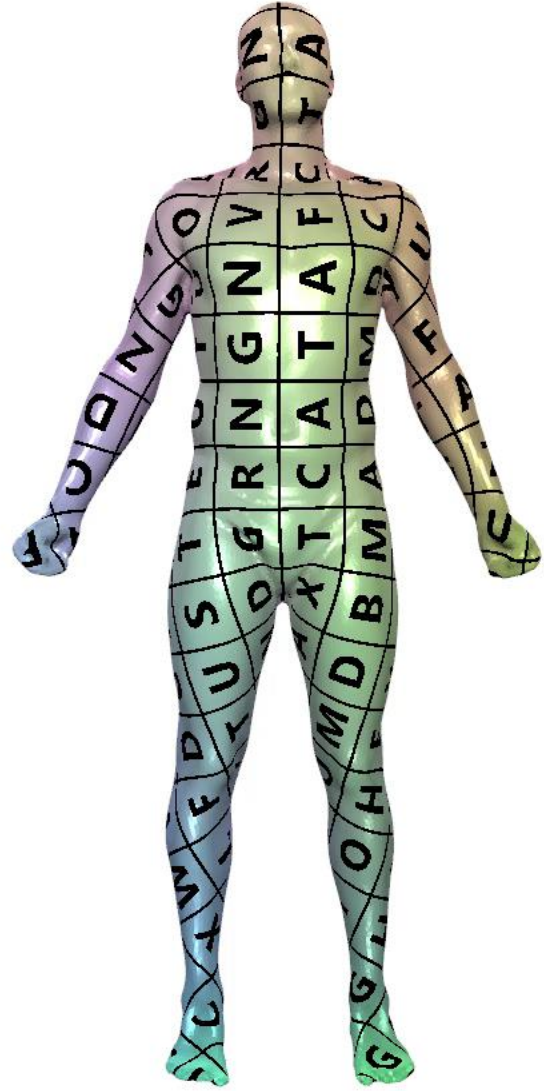


Applications

- **Morphing**
- Attribute transfer
-

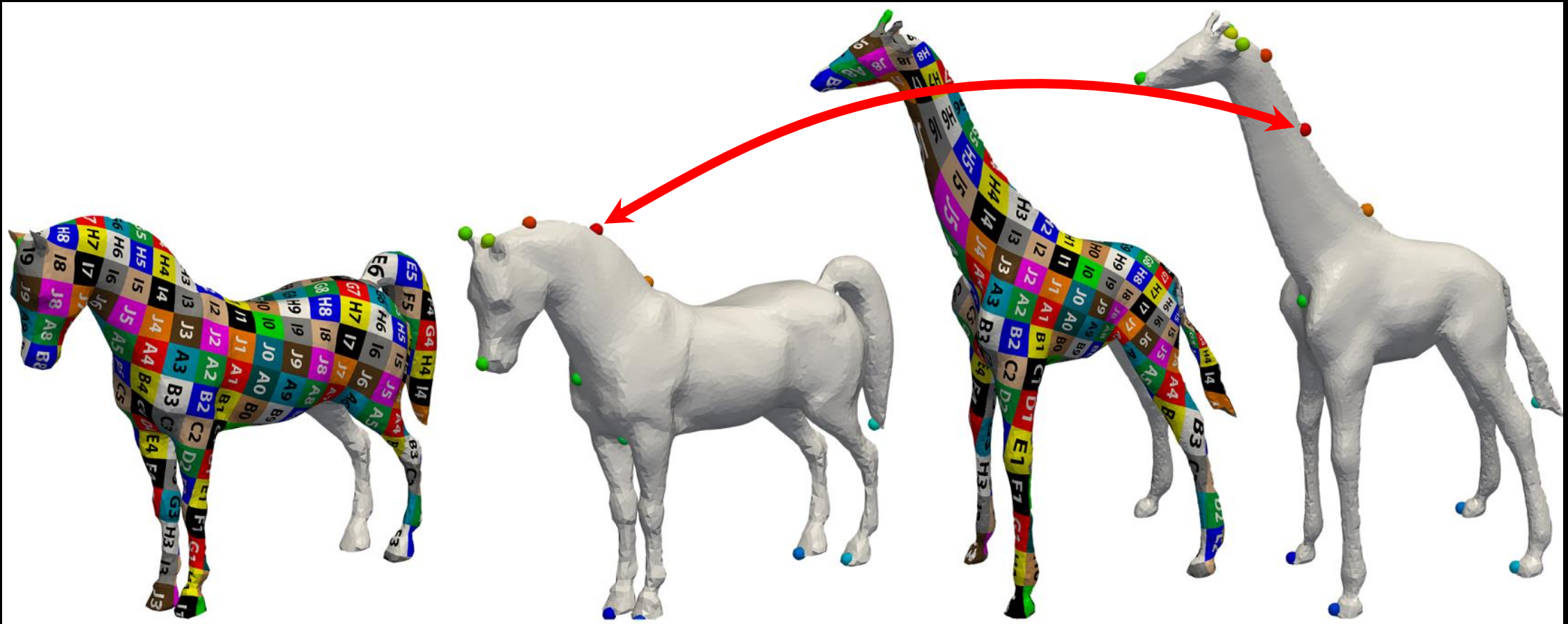






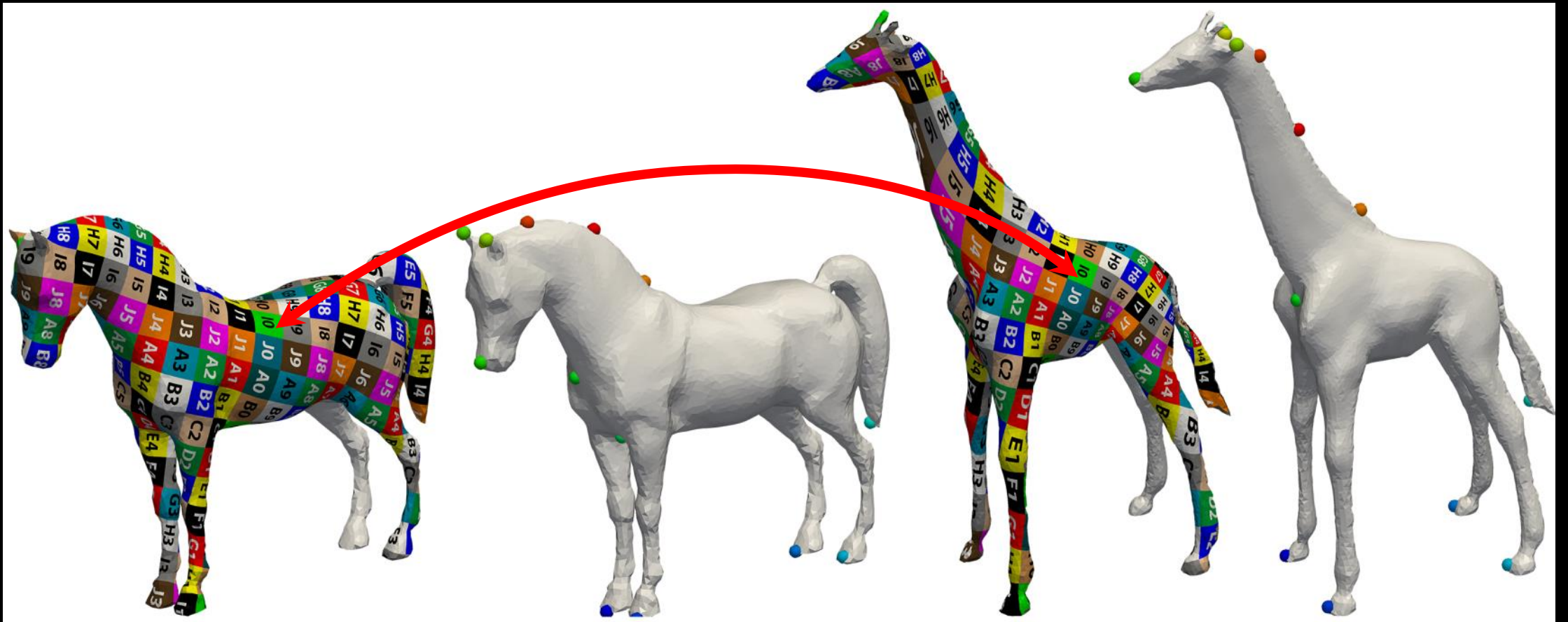
Inputs

- Two (n) models and some corresponding landmarks



Goal

- Bijection and low distortion



Outlines

- Definition
- Application
- Algorithms
 - **Common base domain**
 - **Cross-Parameterization and Compatible Remeshing of 3D Models**
 - Parameterization-based method
 -

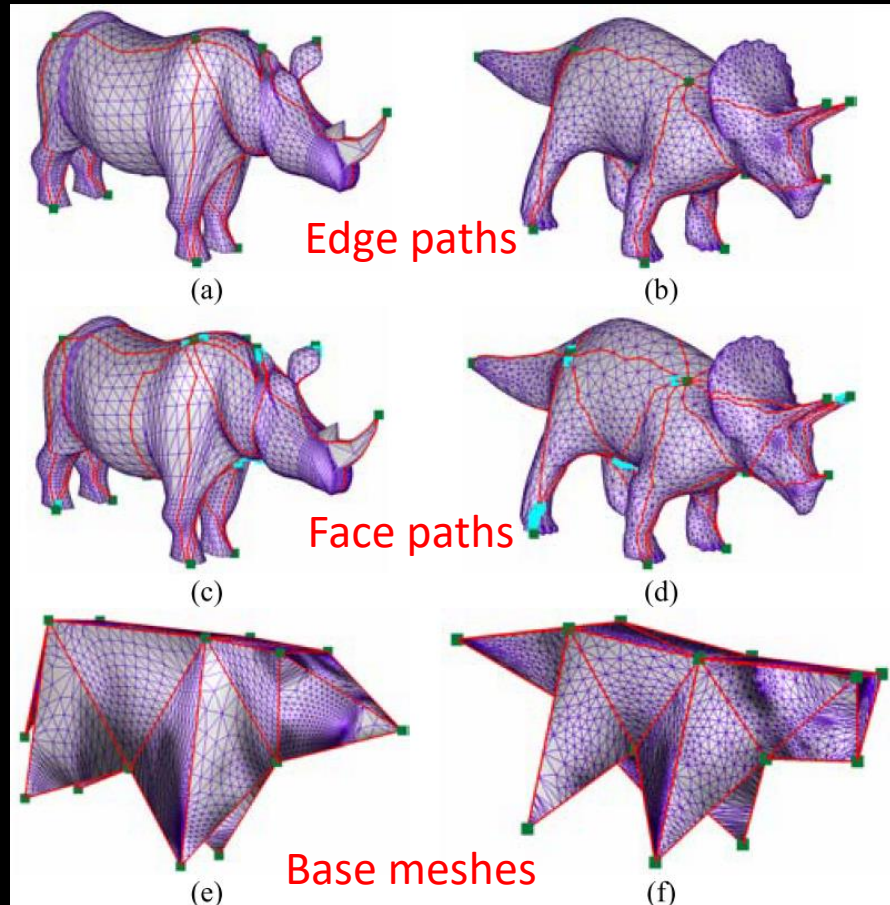
Algorithm stages

- Construct a common base domain
 - Topologically identical triangular layouts of the two meshes.
- Compute a low distortion cross-parameterization
 - Each patch is mapped to the corresponding base mesh triangle.
- Compatibly remeshes the input models using the parameterization

Common base domain

Topologically identical triangular layouts

- Incrementally adding pairs of matching edge paths between feature vertices.



Algorithm *PathMatch*

$M_s' = M_s$

$M_t' = M_t$

Compute the shortest paths s^{ij} for each pair of vertices in V_s

Compute the shortest paths t^{ij} for each pair of vertices in V_t

$ST = \emptyset$

foreach s^{ij}

$ST \leftarrow \langle s^{ij}, t^{ij} \rangle$ /* pairs of matching paths */

while $ST \neq \emptyset$

$\langle s, t \rangle = ST.RemoveShortest()$

a pair of paths with the smallest length sum

if *NonBlocking*(s, t)

Add s to P_s ; Add t to P_t

Remove all interior vertices of s from M_s'

Remove all interior vertices of t from M_t'

Update(ST, s, t)

end

end

end

Cross-Parameterization

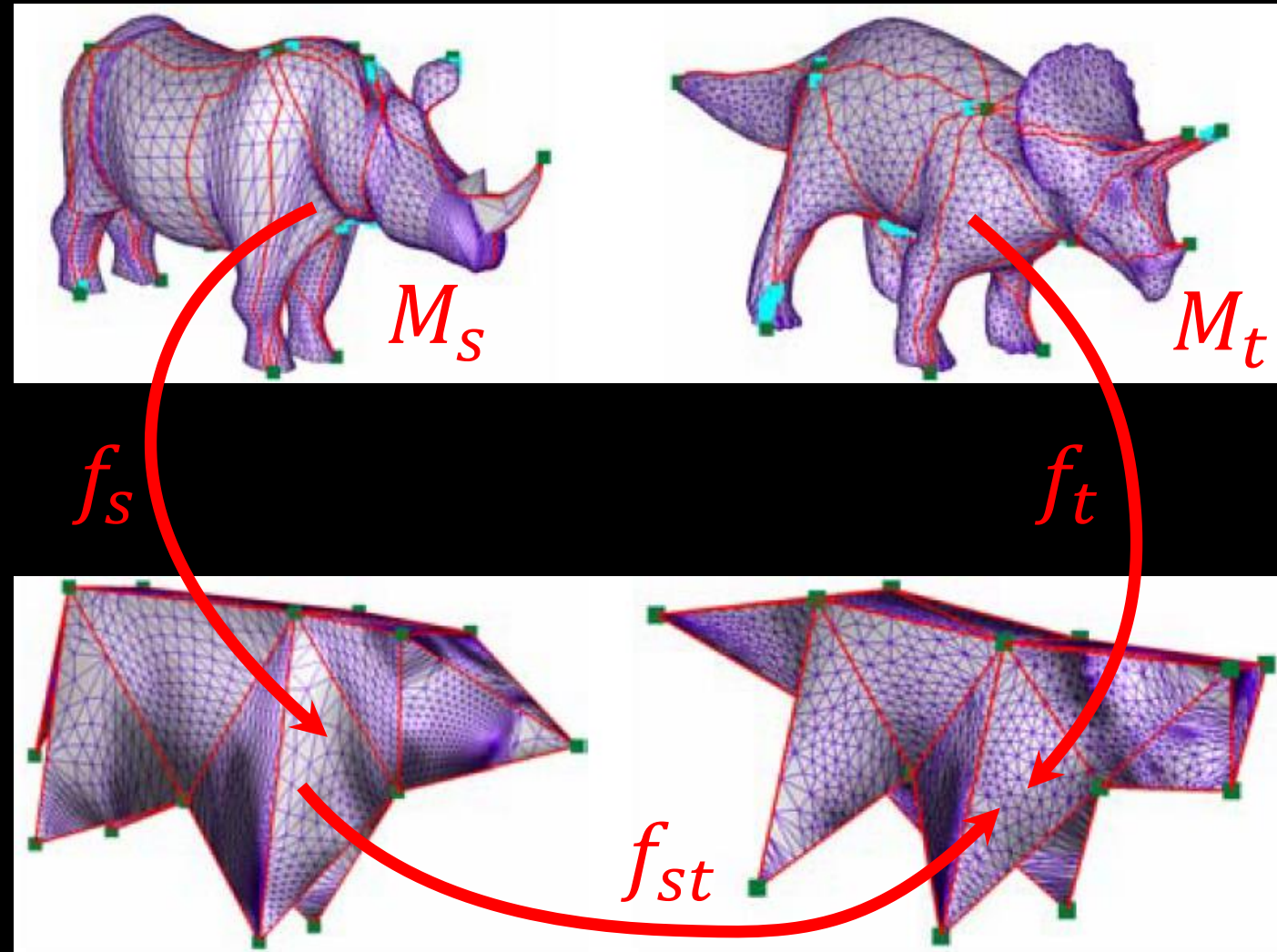
- Tutte's embedding:

Given a triangulated surface **homeomorphic to a disk**, if the (u, v) coordinates at the boundary vertices lie on **a convex polygon** in order, and if the coordinates of the internal vertices are **a convex combination** of their neighbors, then the (u, v) coordinates form a valid parameterization (**without self-intersections, bijective**).

- Each patch is a triangle, i.e., it is a convex boundary.
 - Bijection guarantee.

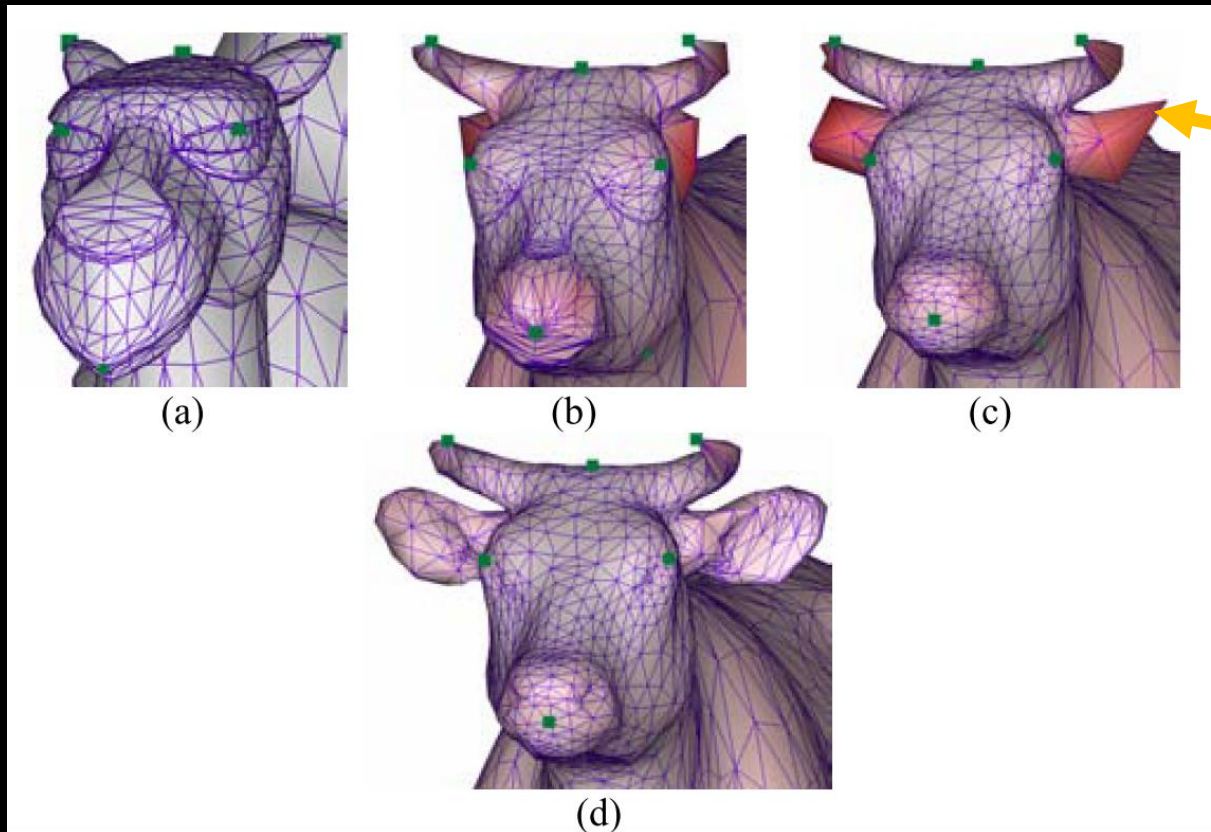
Cross-Parameterization

- $f = f_t^{-1} \circ f_{st} \circ f_s$



Compatible Remeshing

- First remeshes the target model with the connectivity of the source mesh
- Perform smoothing and refinement



High approximation error

(b) Initial projection

(c) After smoothing

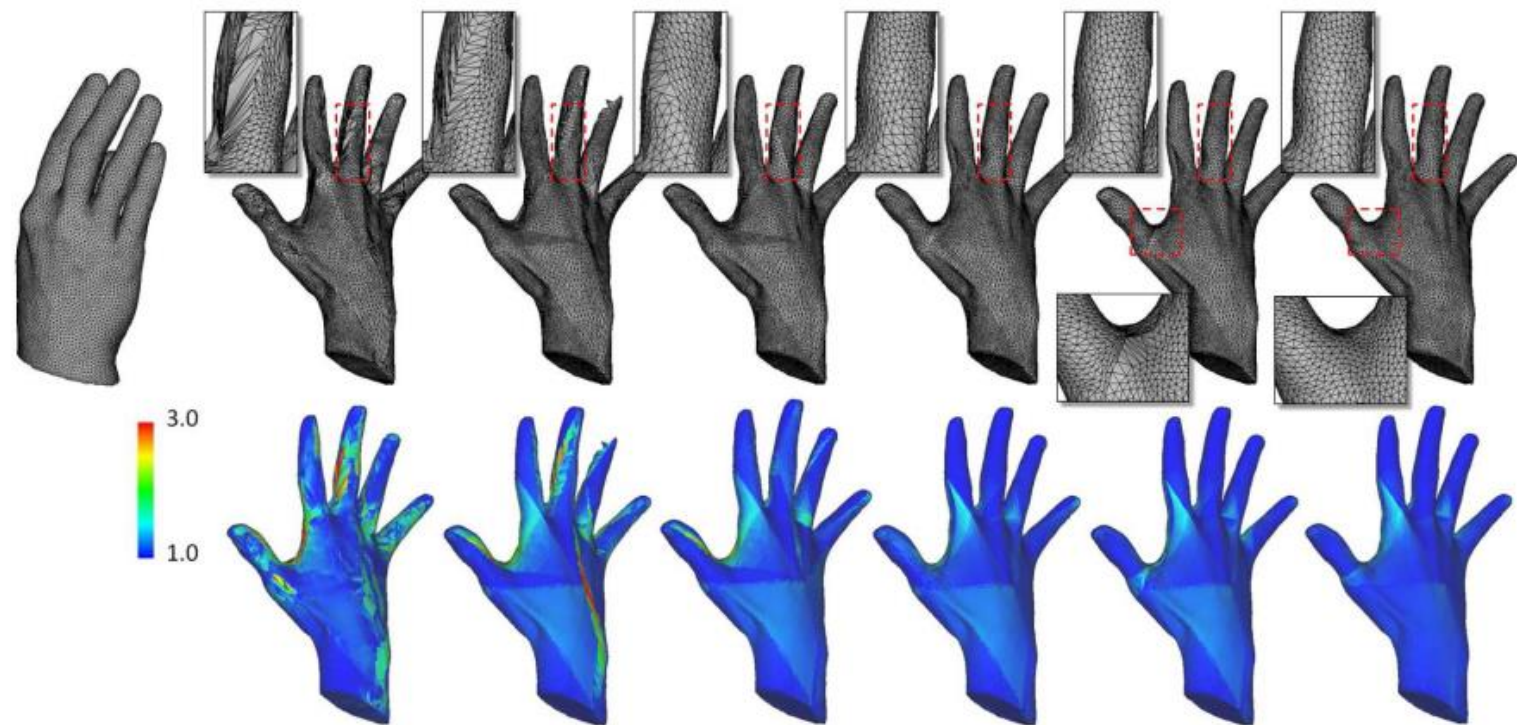
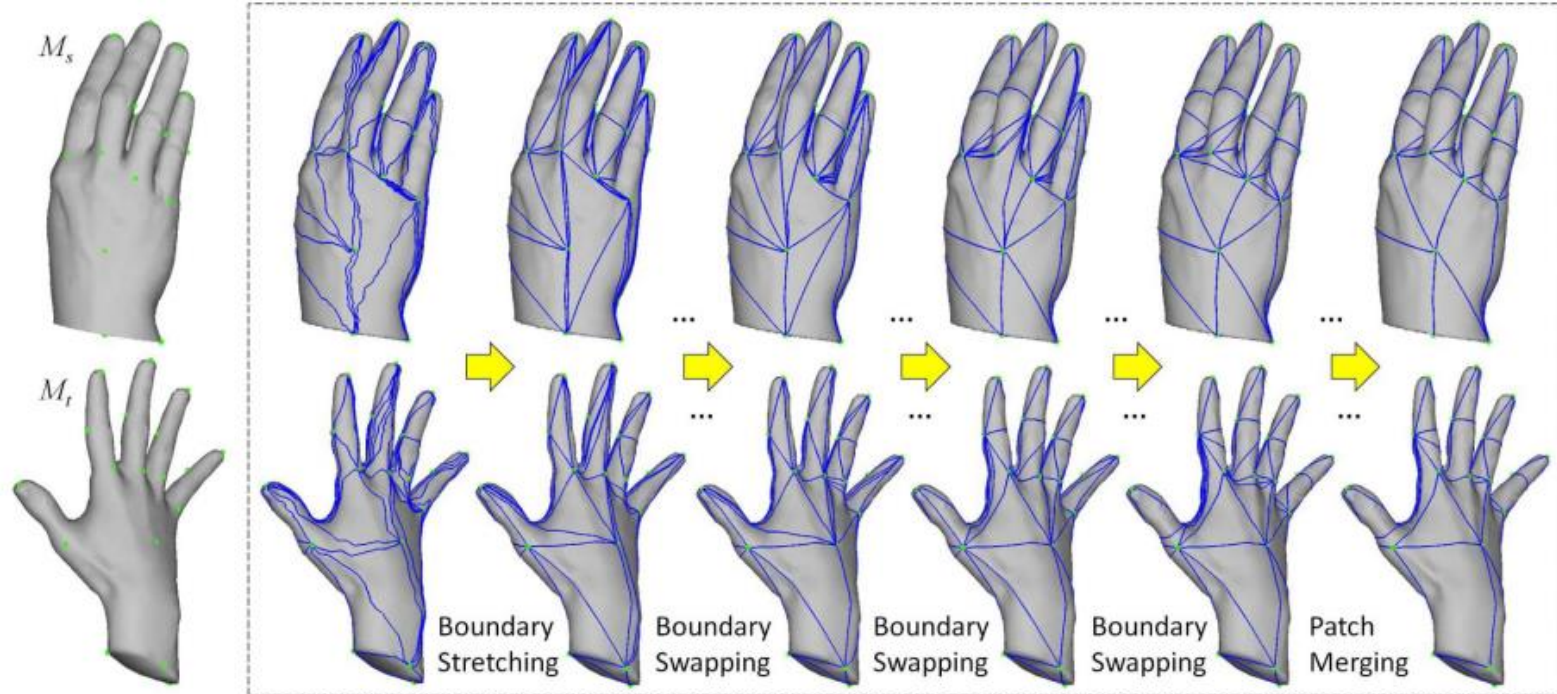
(d) Smoothing and refinement

Disadvantages

- The construction of common base domain is non-trivial.
- The distortion of surface mappings is not optimized directly.

Efficient Optimization of Common Base Domains for Cross Parameterization *2012*

- Initial Base Domain Construction (previous method)
- Boundary Stretching
 - curve stretching operator is to convert a curve into a geodesic curve locally
- Boundary Swapping
 - Similar to edge flip
- Patch Merging
 - helps reduce the distortion

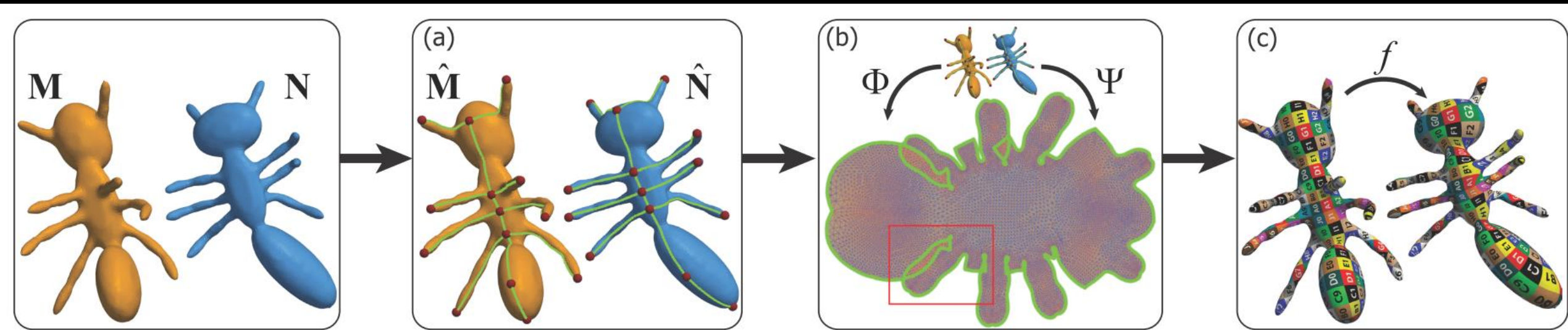


Outlines

- Definition
- Application
- Algorithms
 - Common base domain
 - Parameterization-based method
 -

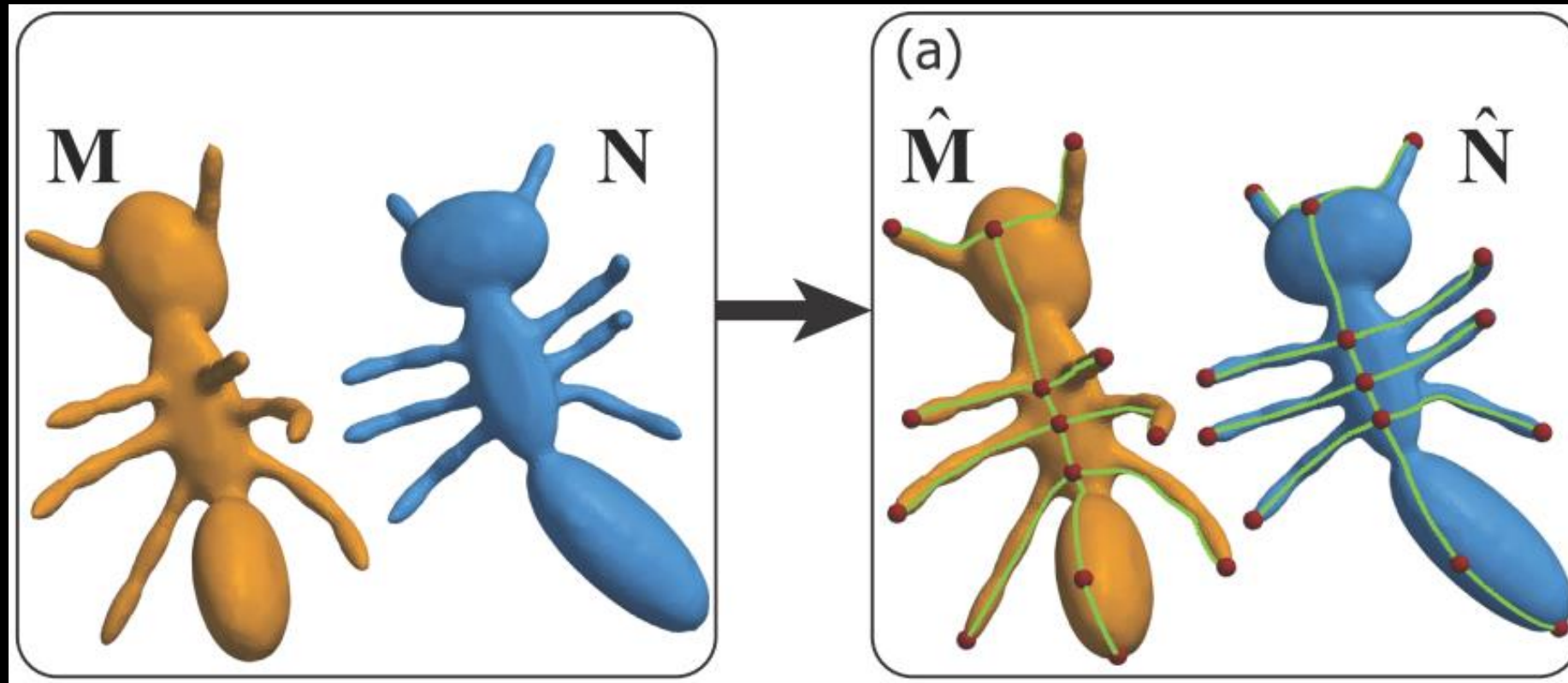
Algorithm steps

- (a) Cutting to disk topology.
- (b) Computing the joint flattenings Φ , Ψ .
- (c) Bijection Lifting.



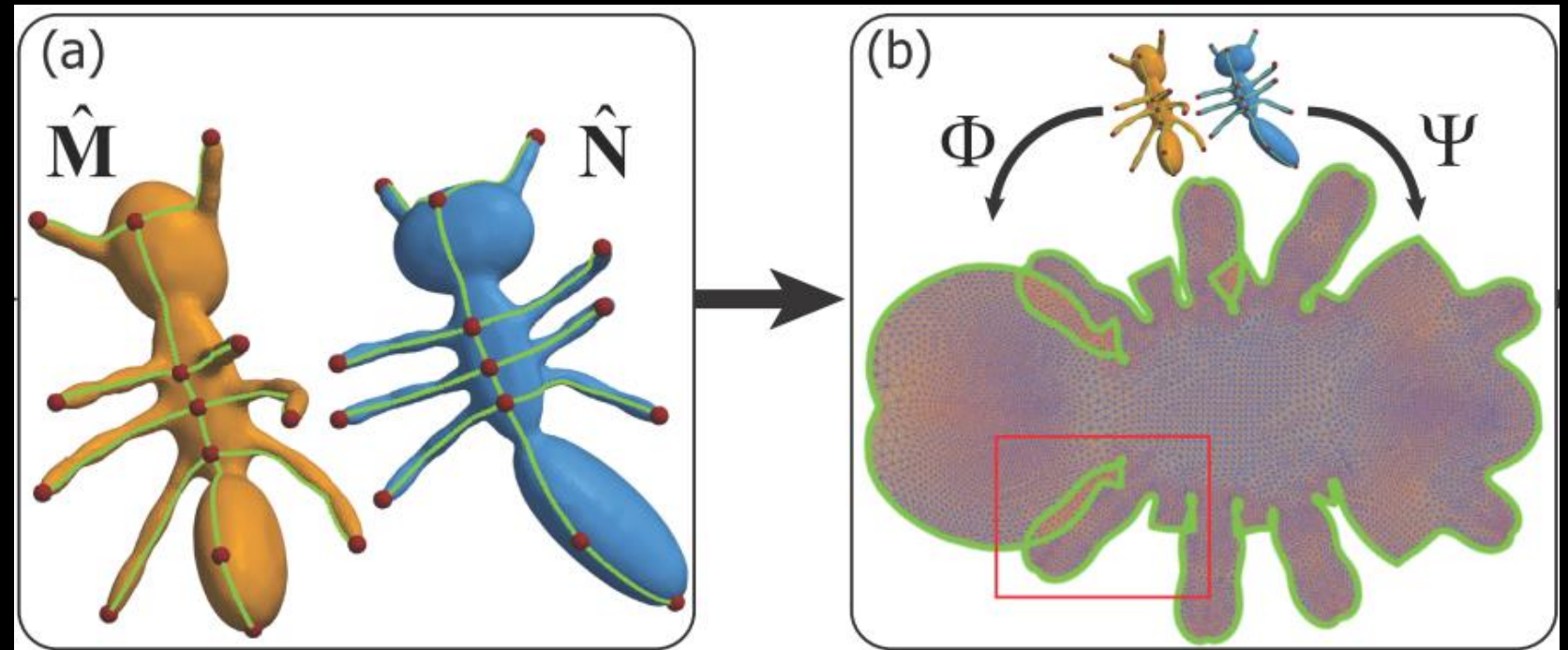
Cutting paths

- Bijective correspondence
 - Shortest path
 - Minimal spanning tree



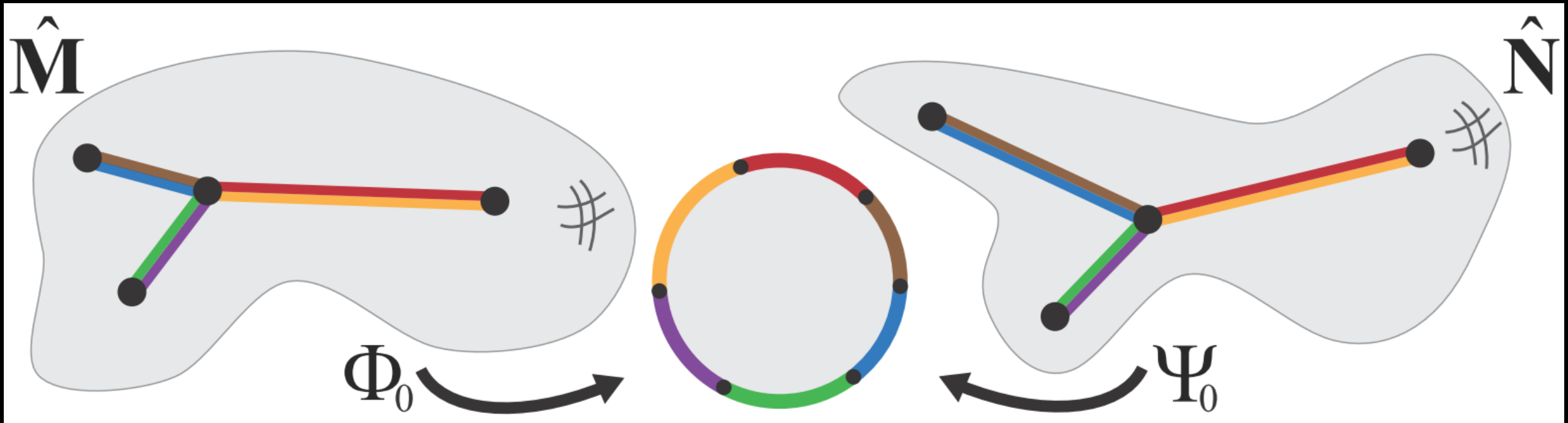
Computing Φ, Ψ

- Constraint
 - **Common boundary condition**
 - Locally injective
- Solvers:
 - Former methods



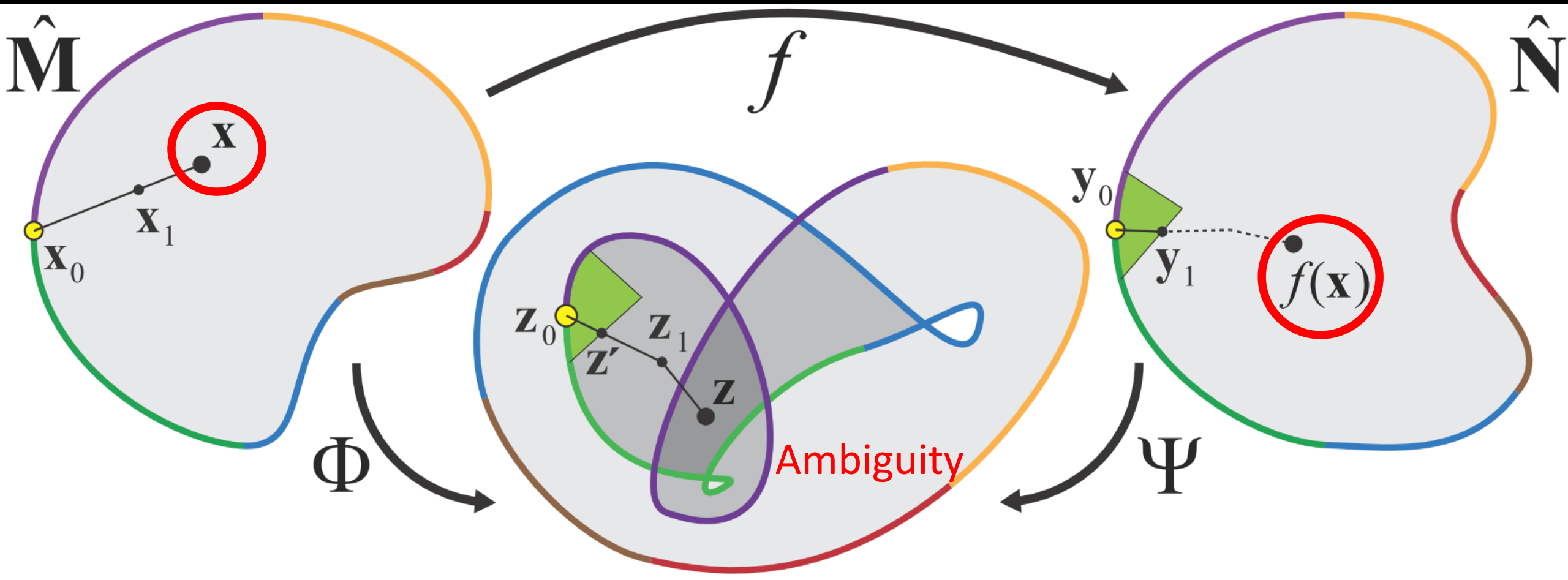
Bijection Lifting

- Bijective parameterizations



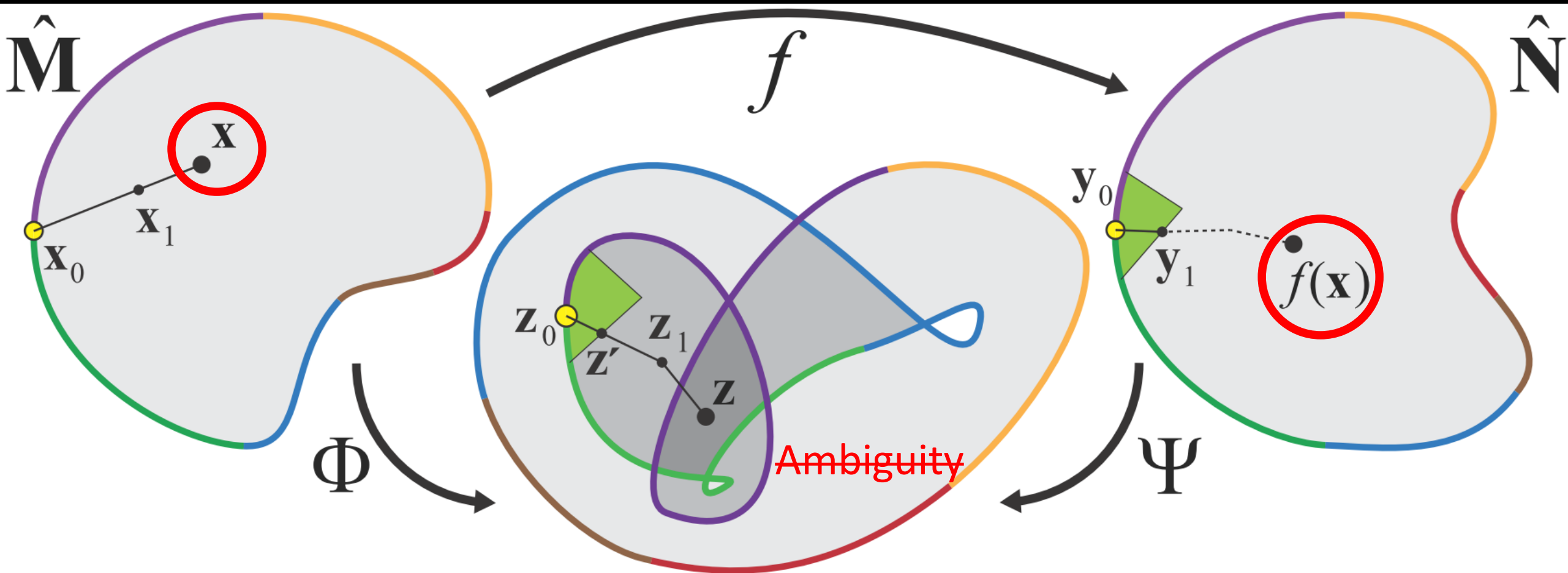
Bijection Lifting

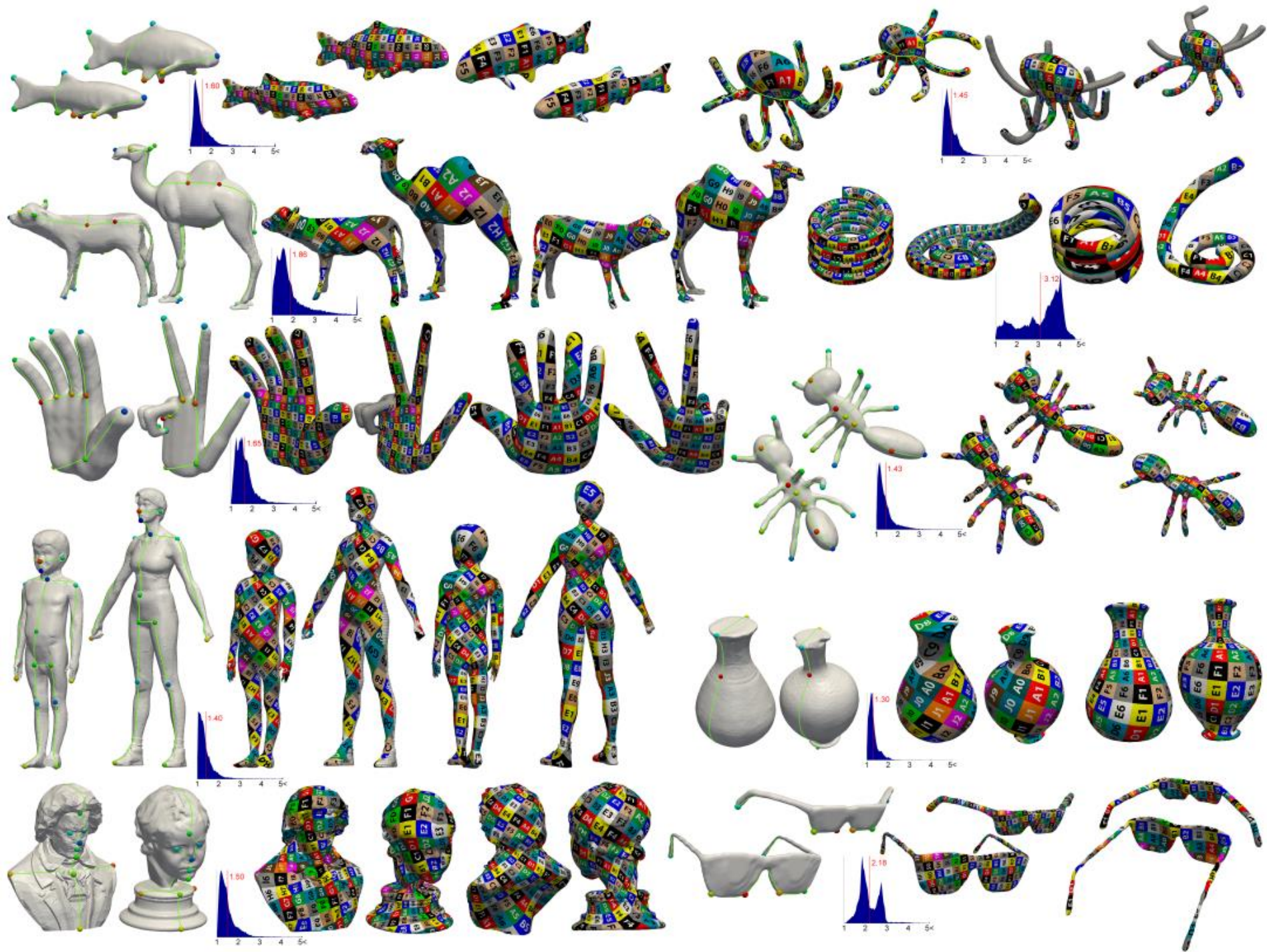
- Only locally injective constrains



Bijection Lifting

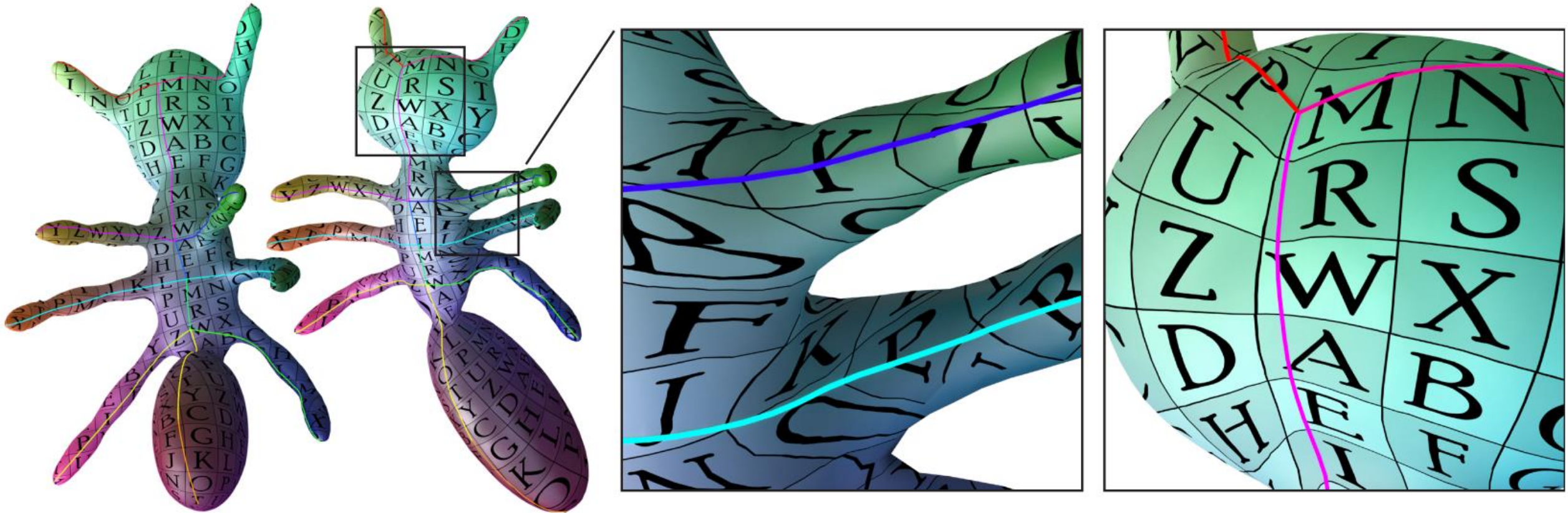
- Only locally injective constrains





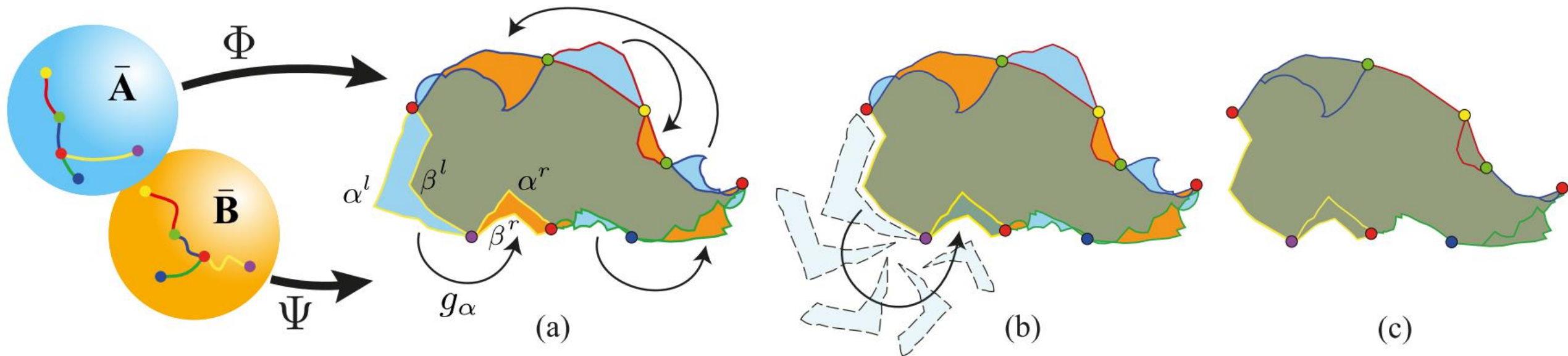
Disadvantages

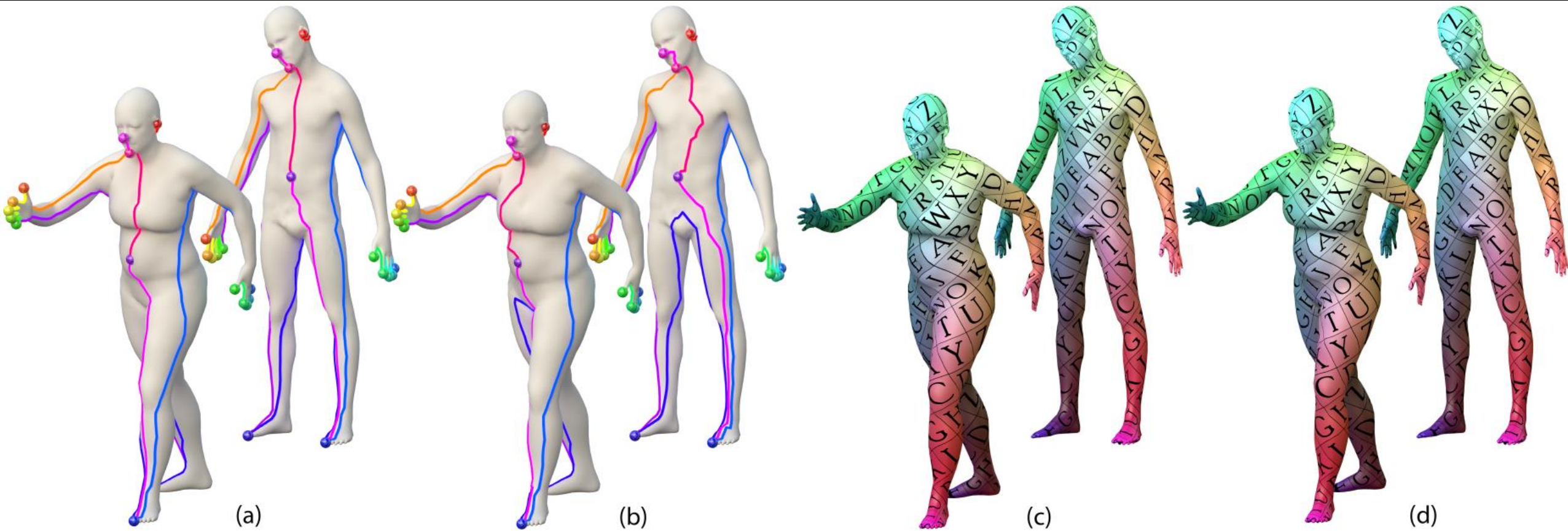
- Cut-dependent



Seamless Surface Mappings

SIGGRAPH 2015





Cut-independent

More methods

- Inter-Surface Mapping, 2004
- Functional Maps: A Flexible Representation of Maps Between Shapes, 2012
- Hyperbolic Orbifold Tutte Embeddings, 2016
- Variance-Minimizing Transport Plans for Inter-surface Mapping, 2017
-

Morphing

Xiao-Ming Fu

Outlines

- Definition
- Angle, length, area, volume, and curvature
 - Example-Driven Deformations Based on Discrete Shells
- Affine transformation
 - As-Rigid-As-Possible Shape Interpolation
- Data-driven morphing
 - A Data-Driven Approach to Realistic Shape Morphing
 - Data-Driven Shape Interpolation and Morphing Editing

Outlines

- **Definition**
- Angle, length, area, volume, and curvature
 - Example-Driven Deformations Based on Discrete Shells
- Affine transformation
 - As-Rigid-As-Possible Shape Interpolation
- Data-driven morphing
 - A Data-Driven Approach to Realistic Shape Morphing
 - Data-Driven Shape Interpolation and Morphing Editing

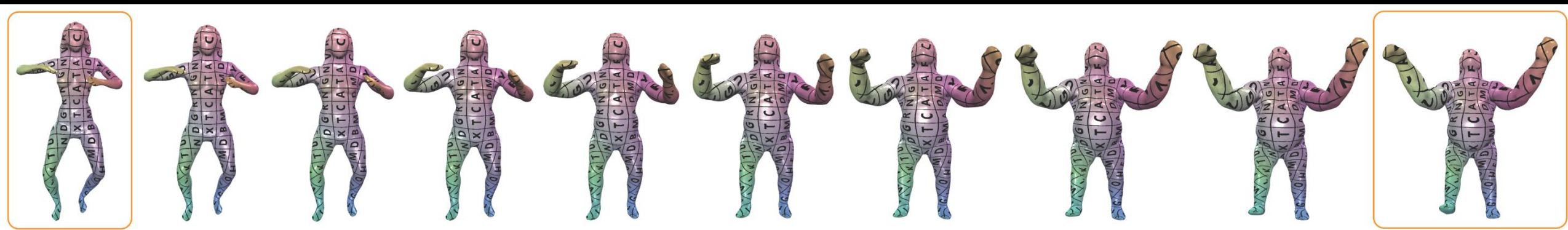
Definition

- Morphing is a special effect in motion pictures and animations that changes (or morphs) one image or shape into another through a seamless transition.



Definition

- Problem: Given M^0 , M^1 , and t , how to compute the shape M^t ?
- $t \in [0,1]$, interpolation
- $t \notin [0,1]$, extrapolation



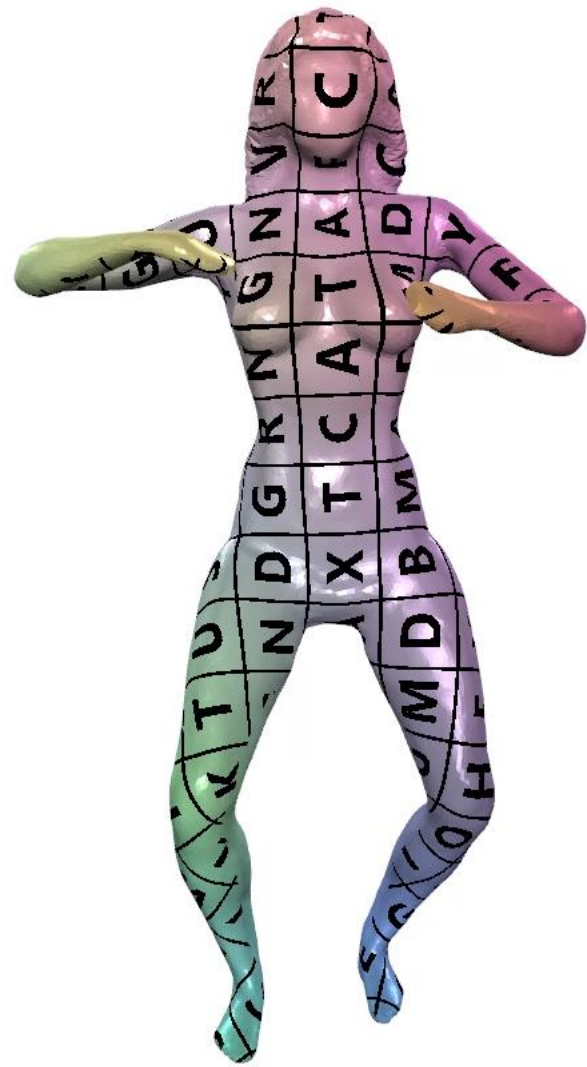
M^0

M^t

M^1

Source

Target



Requirements

- **Look naturally and intuitively**
- Symmetry
- Smooth vertex paths
- Bounded distortion / low distortion
- Foldover-free
- Large deformation
-

Some methods

- First interpolate some values/metrics, then reconstruct the shape.
- **Angle, length, area, volume, and curvature**
 - Example-Driven Deformations Based on Discrete Shells
- **Affine transformation**
 - As-Rigid-As-Possible Shape Interpolation
- **Data-driven morphing**
 - A Data-Driven Approach to Realistic Shape Morphing
 - Data-Driven Shape Interpolation and Morphing Editing

Outlines

- Definition
- Angle, length, area, volume, and curvature
 - Example-Driven Deformations Based on Discrete Shells
- Affine transformation
 - As-Rigid-As-Possible Shape Interpolation
- Data-driven morphing
 - A Data-Driven Approach to Realistic Shape Morphing
 - Data-Driven Shape Interpolation and Morphing Editing
-

Interpolation

Angle, length, and volume

$$\begin{aligned}l_e^t &= (1 - t)l_e^0 + tl_e^1 \\ \theta_e^t &= (1 - t)\theta_e^0 + t\theta_e^1 \\ V^t &= (1 - t)V^0 + tV^1\end{aligned}$$

l_e : edge length

θ_e : dihedral angles

V : volume

$$V = \frac{1}{6} \sum_{f_{i,j,k}} (\mathbf{x}_i \times \mathbf{x}_j) \cdot \mathbf{x}_k$$

Reconstruction

- A mesh with prescribed edge lengths and dihedral angles does not exist.

$$E_l = \frac{1}{2} \sum (l_e - l_e^t)^2$$
$$E_a = \frac{1}{2} \sum^e (\theta_e - \theta_e^t)^2$$
$$E_v = \frac{1}{2} (V - V_e^t)^2$$
$$E = \lambda E_l + \mu E_b + \nu E_v$$

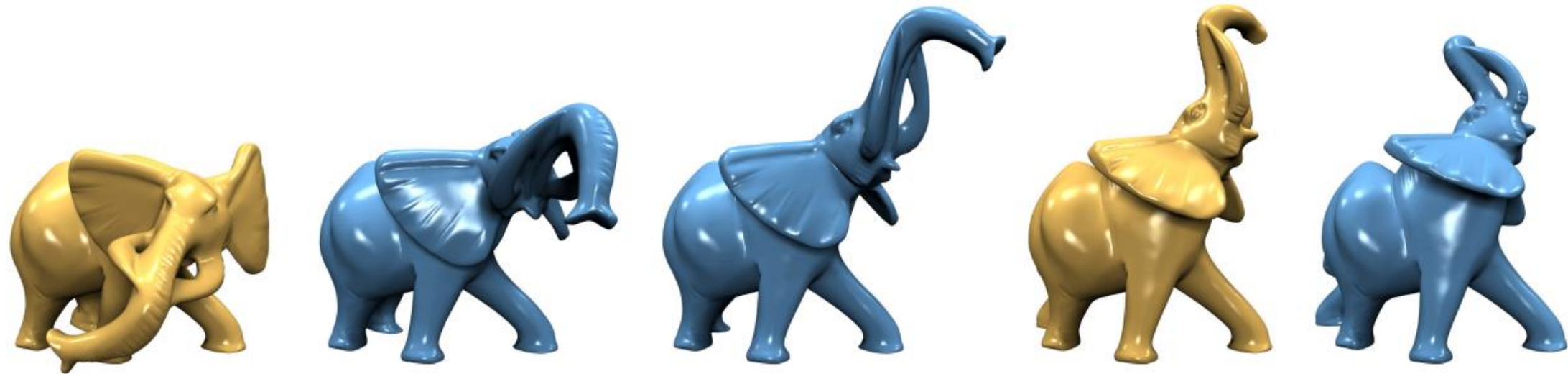


Figure 5: *Interpolation and extrapolation of the yellow example poses. The blending weights are 0, 0.35, 0.65, 1.0, and 1.25.*

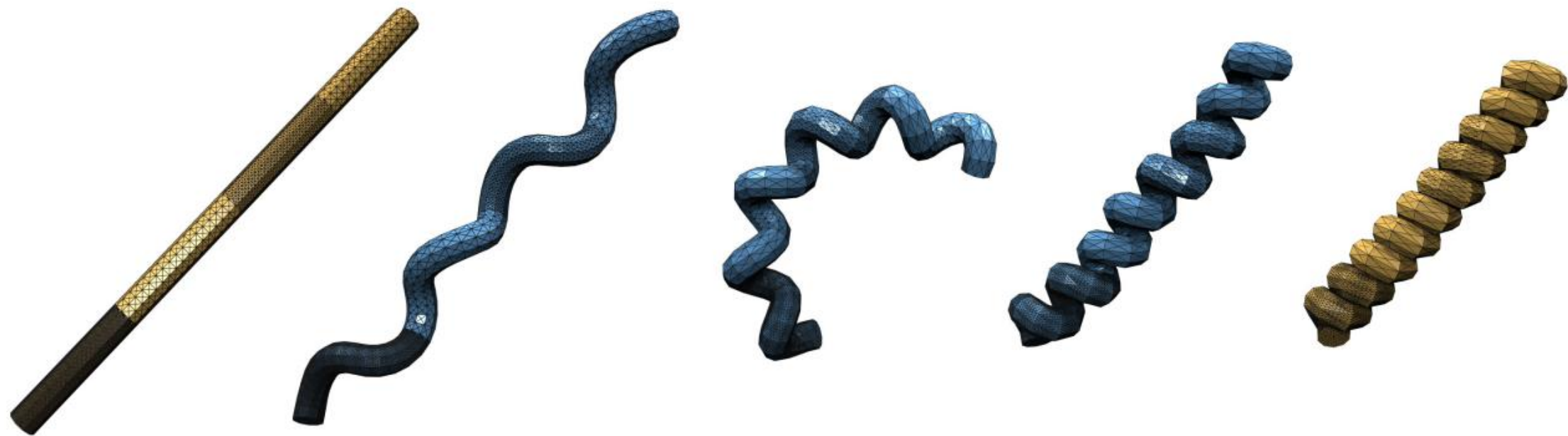


Figure 6: *Interpolation of an adaptively meshed and strongly twisted helix with blending weights 0, 0.25, 0.5, 0.75, 1.0.*

Outlines

- Definition
- Angle, length, area, volume, and curvature
 - Example-Driven Deformations Based on Discrete Shells
- **Affine transformation**
 - **As-Rigid-As-Possible Shape Interpolation**
- Data-driven morphing
 - A Data-Driven Approach to Realistic Shape Morphing
 - Data-Driven Shape Interpolation and Morphing Editing

Interpolation

- How to define $A(t)$ reasonably?
- Simplest solution:

$$A(t) = (1 - t)I + tA$$

- More elaborate approaches:
 - Singular value decomposition

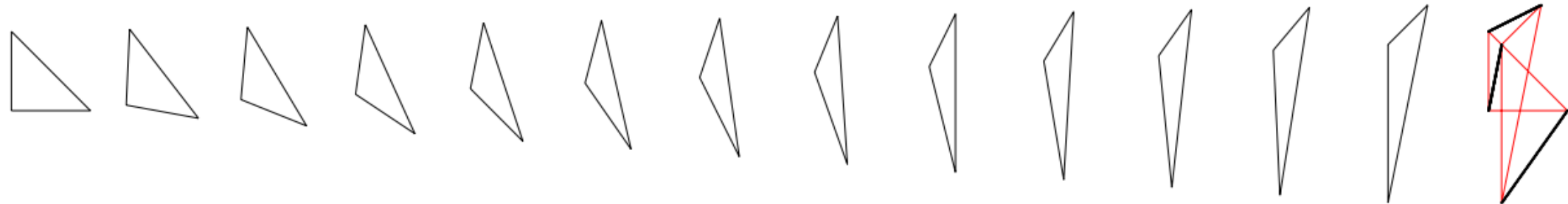
$$A = U\Sigma V^T$$

$$A(t) = U(t)((1 - t)I + t\Sigma)V^T(t)$$

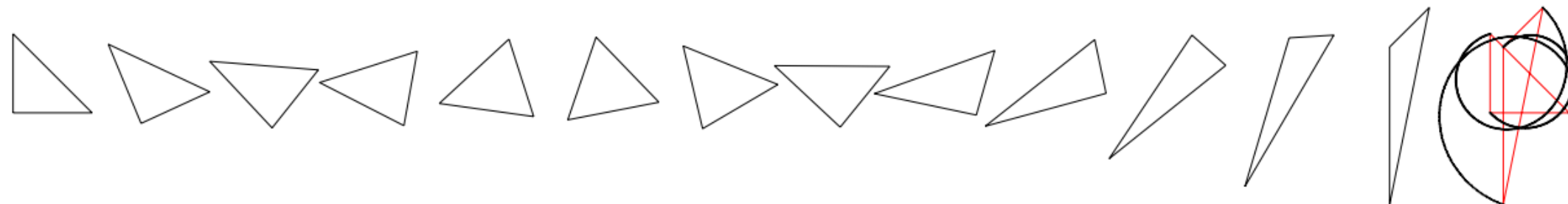
- Polar decomposition

$$A = U\Sigma V^T = UV^T V\Sigma V^T = RS$$

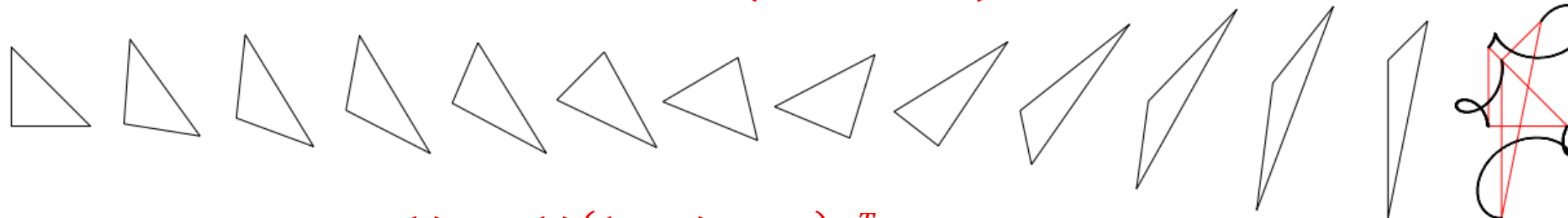
$$A(t) = R(t)((1 - t)I + tS)$$



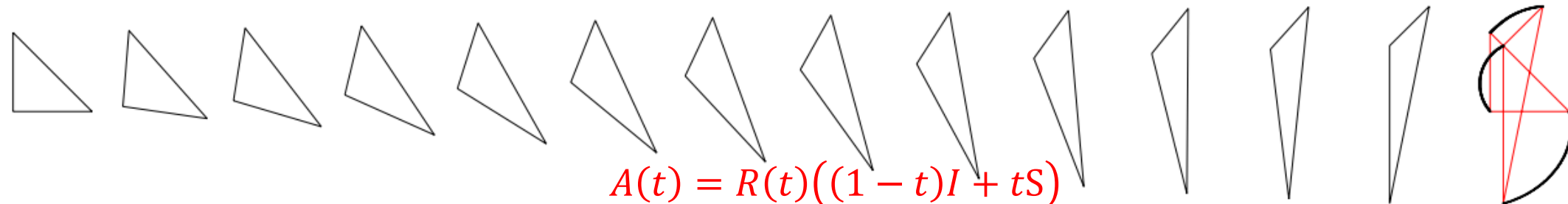
$$A(t) = (1 - t)I + tA$$



$$A(t) = U(t)((1 - t)I + t\Sigma)V^T(t)$$



$$A(t) = U(t)((1 - t)I + t\Sigma)V^T(t) \text{ with subtracting } 2\pi$$



$$A(t) = R(t)((1 - t)I + tS)$$

Reconstruction

- Least squares:

$$E = \sum_f \|J - A(t)\|_F^2$$



Figure 12: Morph between photographs of an elephant and a giraffe.

Outlines

- Definition
- Angle, length, area, volume, and curvature
 - Example-Driven Deformations Based on Discrete Shells
- Affine transformation
 - As-Rigid-As-Possible Shape Interpolation
- **Data-driven morphing**
 - A Data-Driven Approach to Realistic Shape Morphing
 - Data-Driven Shape Interpolation and Morphing Editing

Data-driven approach

- Problem:
 - Input: a database with various models belonging to the same category and containing identical connectivity
 - Given source and target models, how to utilize the database to generate the morphing?



Two stages

- Offline stage
 - Analyze the model database to form **local shape spaces** that better characterize the plausible distribution of models in the category.
- Online stage
 - When the source and target models are given, we **find reference models** in the local shape spaces and **use them to guide the as-rigid-as-possible shape morphing**.

More details

- Offline stage
 - Define distance between pairs of models
- Online stage
 - Find a minimal distance path connecting the source and target models
 - In-between reference models, do as-rigid-as-possible shape interpolation.

Distance Measure

$$\bar{d}(M_i, M_j) = \sqrt{\frac{\sum_{k=1}^n \|v_k^i - v_k^j\|^2}{n}}$$

v_k^i : the k^{th} vertex of the i^{th} model (M_i).

n : the vertex number of the model

Pre-alignment: align models in a database using rigid transforms with the known correspondences.

Morphing

- Path Optimization
 - Shortest path (see more complex algorithm in the paper)
- Interpolation

$$E = \sum_{k=1}^{N_R} w_k(t) E_k$$

N_R ← The number of models on the generated path.

$$E_k = \sum_{i=1}^n \left(\sum_{j \in \Omega(i)} w_{ij} \left\| (\hat{v}^i - \hat{v}^j) - R_k^i (v_k^i - v_k^j) \right\|^2 + \gamma \left\| \hat{v}^i - v_k^i \right\|^2 \right)$$

$w_k(t)$: $\exp(-\varepsilon|t - t_k|)$ where $t_k = \frac{k-1}{N_R-1}$

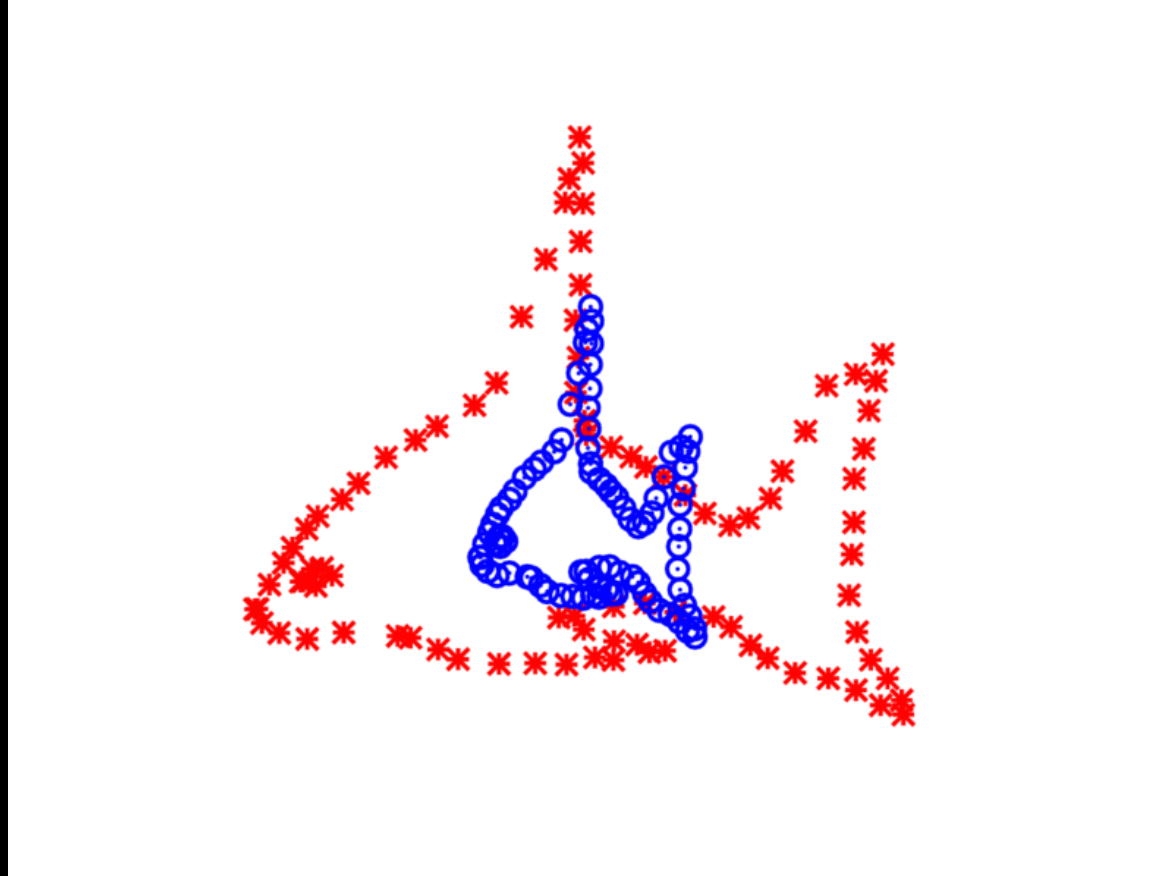
Solver: Local/global

Point set registration

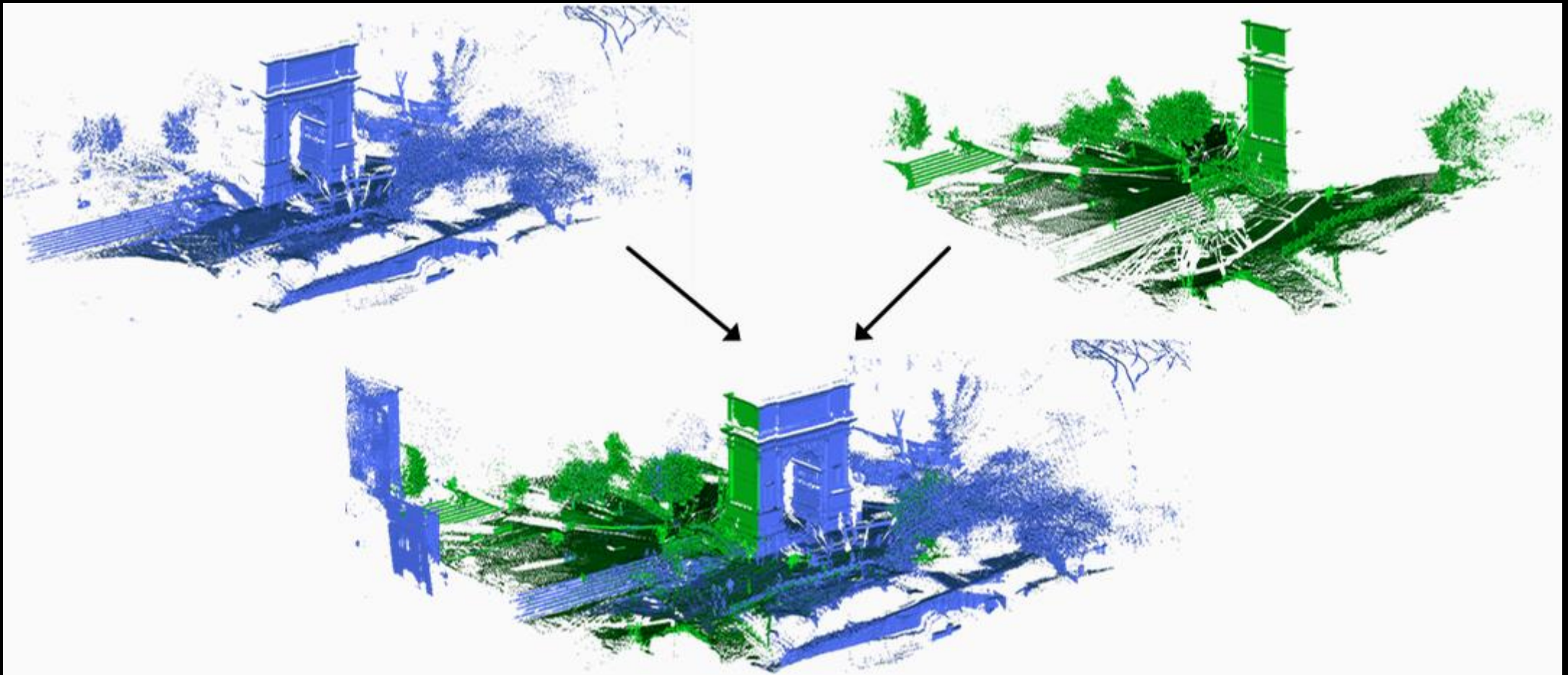
Xiao-Ming Fu

Point set registration

- The process of finding a **spatial transformation** that **aligns two point sets**.



- The purpose of finding such a transformation includes **merging multiple data sets into a globally consistent model**, and **mapping a new measurement** to a known data set to identify features or to estimate its pose.



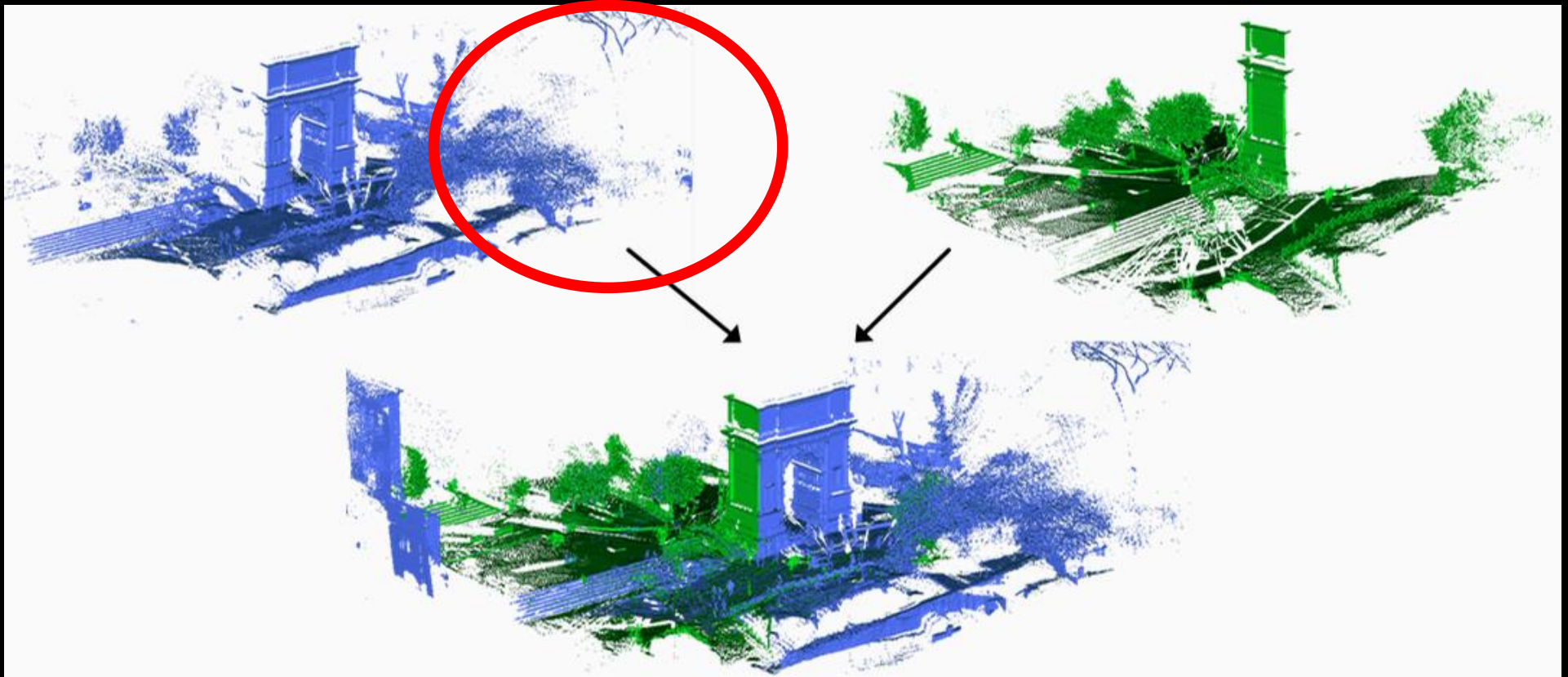
Problem

- Input: two finite size point sets $\{P, Q\}$, which contain M and N points.
- Output: a transformation to be applied to the **moving** “model” point set P such that the **difference** between P and the **static** “scene” set Q is **minimized**.
- The mapping may consist of a rigid or non-rigid transformation.
 - Rigid registration: translation and rotation
 - Non-rigid registration: affine transformations or any nonlinear transformation

For example: Spline

Challenges

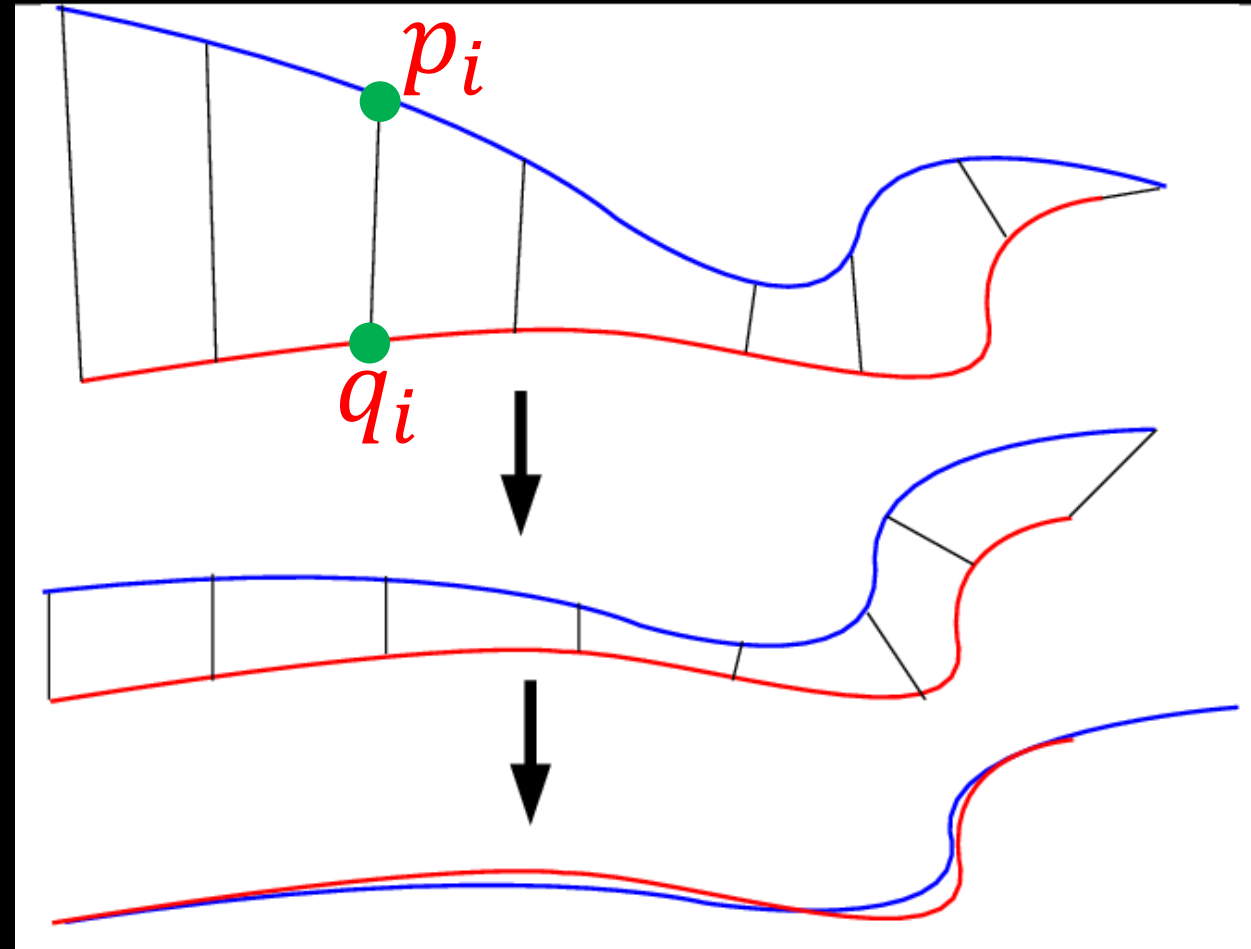
- No correspondences.
- Noisy point cloud.



Iterative closest point (ICP)

https://en.wikipedia.org/wiki/Iterative_closest_point

- 1. $\forall p_i \in P$, match the closest point in Q , denoted as q_i
- 2. Estimate the rigid transformation that aligns the corresponding points as much as possible.
- 3. Iterate above two steps.



Estimation of rigid transformation

- Error:

$$E(P, Q) = \sum_{(p_i, q_i)} \|p_i - q_i\|_2^2$$

Compute rotation R and translation t :

$$E(P, Q) = \sum_{(p_i, q_i)} \|Rp_i + t - q_i\|_2^2$$

Analytical solution

- Define $\mu_p = \frac{1}{n} \sum_{i=1}^n p_i$, $\mu_q = \frac{1}{n} \sum_{i=1}^n q_i$

$$E(P, Q) = \sum_{i=1}^n \|Rp_i + t - q_i\|^2$$

$$= \sum_{i=1}^n \|Rp_i + t - q_i + R\mu_p - \mu_q - R\mu_p + \mu_q\|^2$$

$$= \sum_{i=1}^n \|R(p_i - \mu_p) - (q_i - \mu_q) + t + R\mu_p - \mu_q\|^2$$

Analytical solution

$$\begin{aligned} E(P_n Q) &= \sum_{i=1}^n \|R(p_i - \mu_p) - (q_i - \mu_q)\|^2 + \|t + R\mu_p - \mu_q\|^2 \\ &\quad + 2(t + R\mu_p - \mu_q)^T (R(p_i - \mu_p) - (q_i - \mu_q)) \end{aligned}$$

Since:

$$\begin{aligned} &\sum_{i=1}^n 2(t + R\mu_p - \mu_q)^T (R(p_i - \mu_p) - (q_i - \mu_q)) \\ &= 2(t + R\mu_p - \mu_q)^T \sum_{i=1}^n (R(p_i - \mu_p) - (q_i - \mu_q)) \\ &= 2(t + R\mu_p - \mu_q)^T \left(\sum_{i=1}^n R(p_i - \mu_p) - \sum_{i=1}^n (q_i - \mu_q) \right) = 0 \end{aligned}$$

Analytical solution

$$E(P, Q) = \sum_{i=1}^n \|R(p_i - \mu_p) - (q_i - \mu_q)\|^2 + \|t + R\mu_p - \mu_q\|^2$$

No matter what R is got, set $t = -R\mu_p + \mu_q$.

Thus,

$$\begin{aligned} E(P, Q) &= \sum_{i=1}^n \|R(p_i - \mu_p) - (q_i - \mu_q)\|^2 \\ &= \sum_{i=1}^n (p_i - \mu_p)^T \mathbf{R}^T \mathbf{R} (p_i - \mu_p) + \| (q_i - \mu_q) \|^2 - 2(q_i - \mu_q)^T R(p_i - \mu_p) \\ &= \sum_{i=1}^n (p_i - \mu_p)^T (p_i - \mu_p) + \| (q_i - \mu_q) \|^2 - 2(q_i - \mu_q)^T R(p_i - \mu_p) \end{aligned}$$

Analytical solution

$$\begin{aligned} & \arg \min_R E(P, Q) \\ &= \arg \min_R \sum_{i=1}^n \left((p_i - \mu_p)^T (p_i - \mu_p) + \| (q_i - \mu_q) \|^2 - 2(q_i - \mu_q)^T R(p_i - \mu_p) \right) \end{aligned}$$

Analytical solution

- If M is a positive-symmetric-definite matrix then for any orthogonal R , $tr(M) > tr(RM)$.
- Proof: Set $M = AA^T$

$$tr(RM) = tr(RAA^T) = tr(A^T RA) = \sum a_i^T (Ra_i)$$

Schwarz inequality: $a_i^T (Ra_i) \leq \sqrt{a_i^T a_i (a_i R^T R a_i)} = a_i^T a_i = tr(M)$

Analytical solution

• Denote $H = \sum_{i=1}^n \left((p_i - \mu_p)(q_i - \mu_q)^T \right) = U\Sigma V^T$.

Solve $\arg \max_R tr(2RH)$.

Set $X = VU^T$,

Then, $XH = V\Sigma V^T$

For any orthonormal matrix B ,

$$tr(XH) \geq tr(BXH)$$

Thus,

$$VU^T = X = \arg \max_R tr(2RH)$$

Atlas generation

Xiao-Ming Fu

Outlines

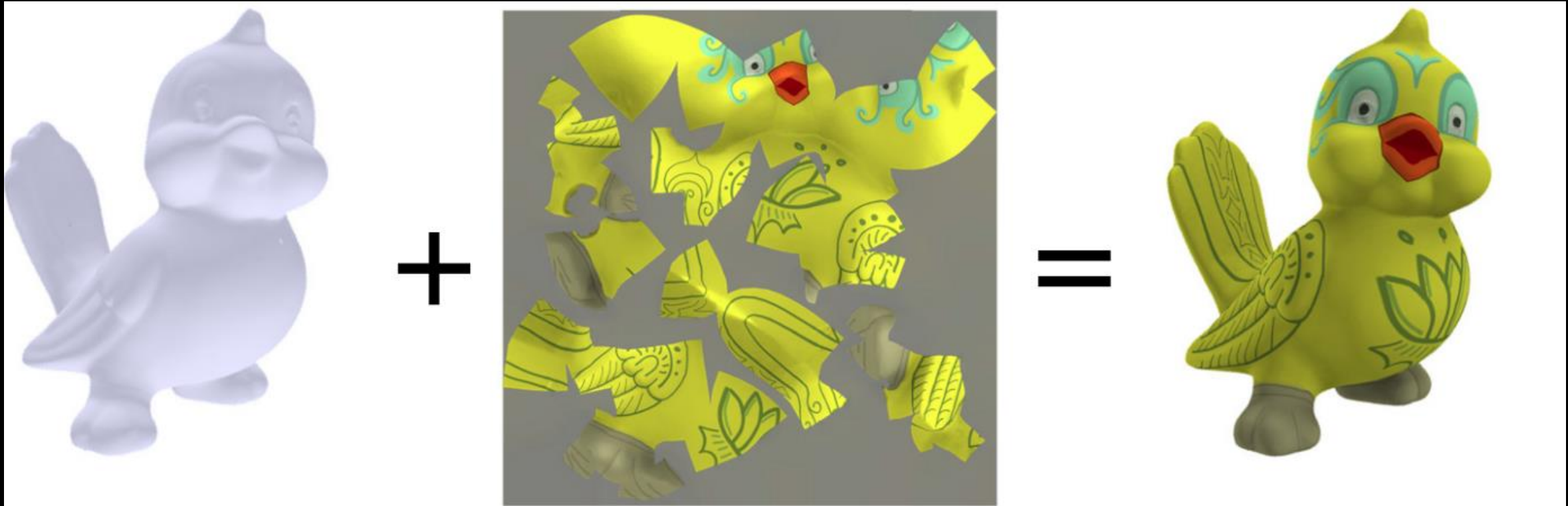
- Definition
- Mesh cutting
- Chart parameterization
 - Bijective, low distortion
- Chart packing

Outlines

- Definition
- Mesh cutting
- Chart parameterization
 - Bijective, low distortion
- Chart packing

Texture Mapping

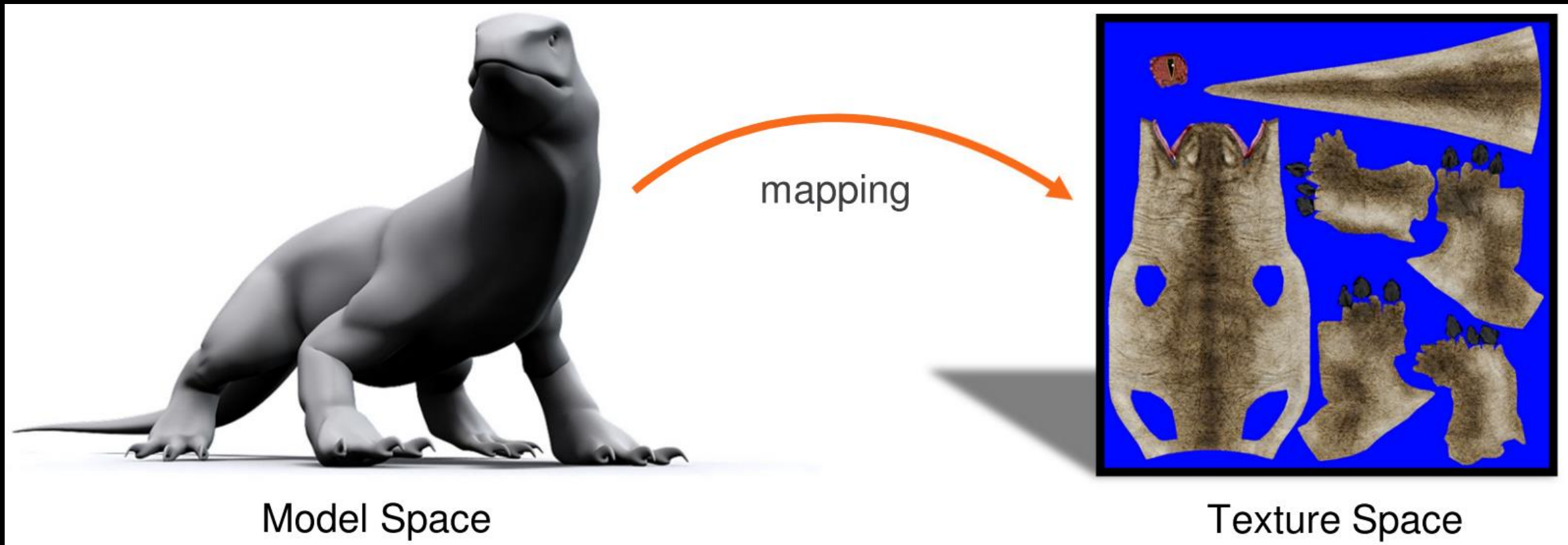
- Texture mapping is a method for defining high frequency detail, surface texture, or color information on a computer-generated graphic or 3D model.





Atlas

- Requires defining a **mapping** from the model space to the texture space.

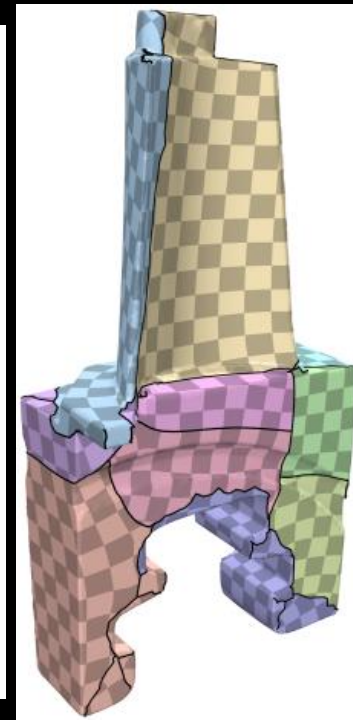
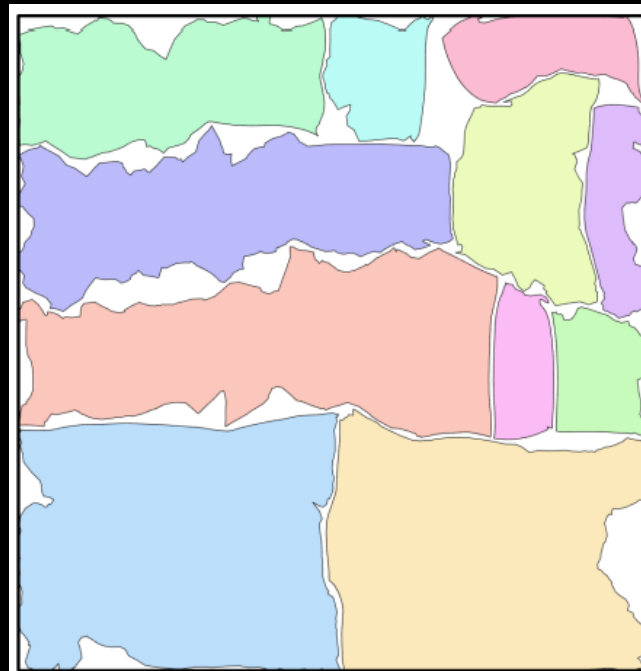
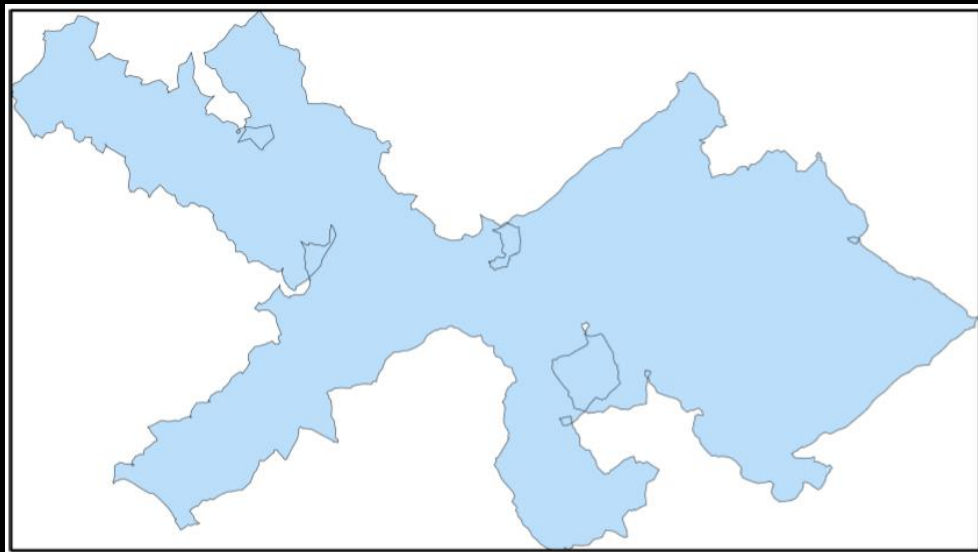
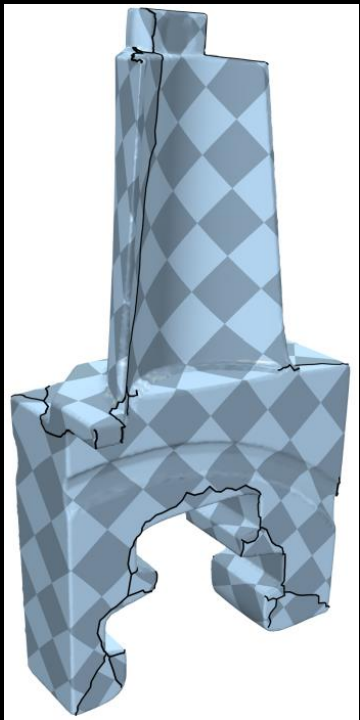


Generation process

Mesh Cutting

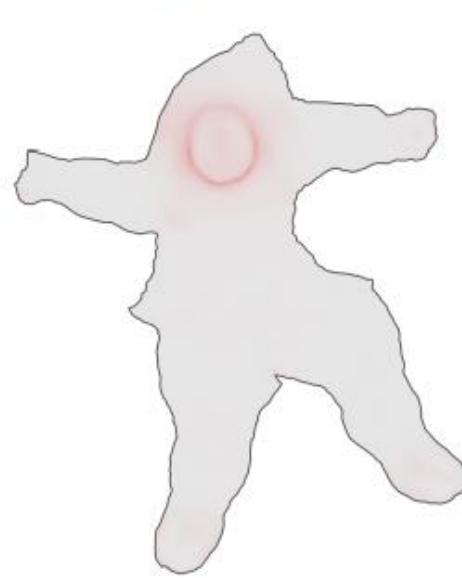
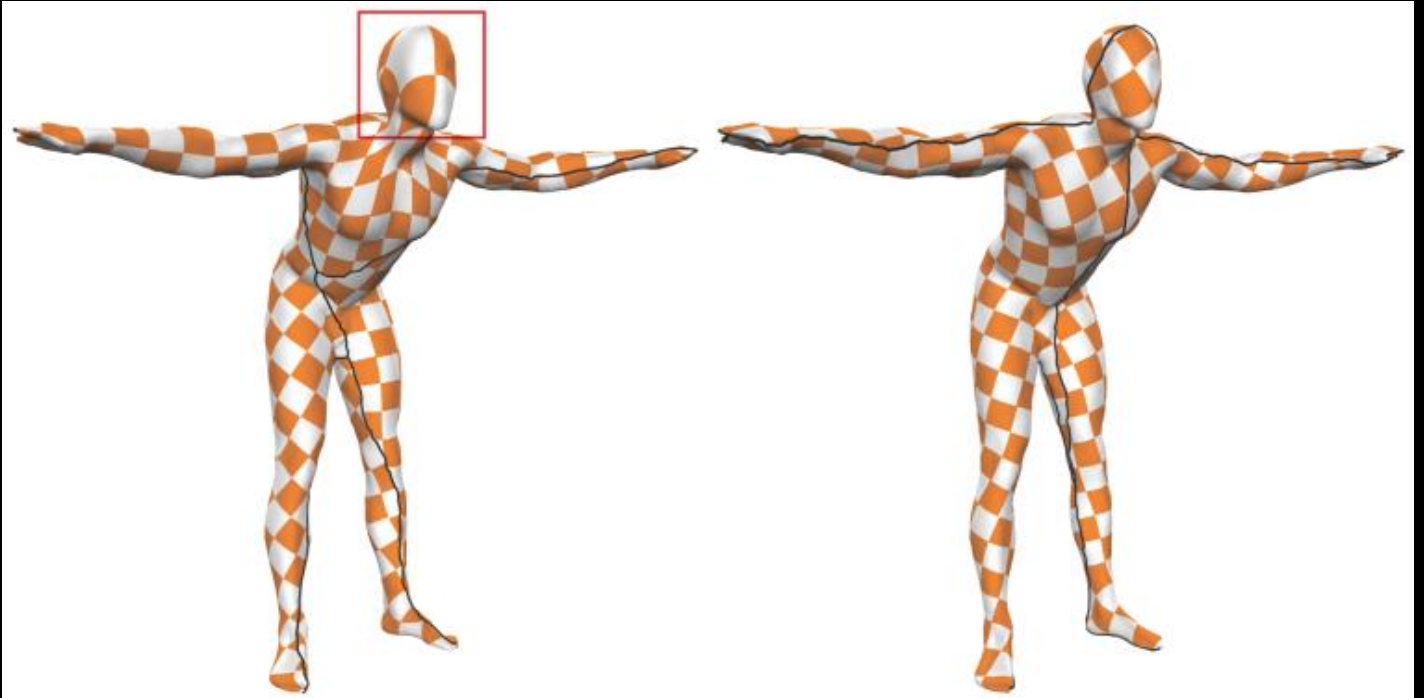
Parameterizations

Packing

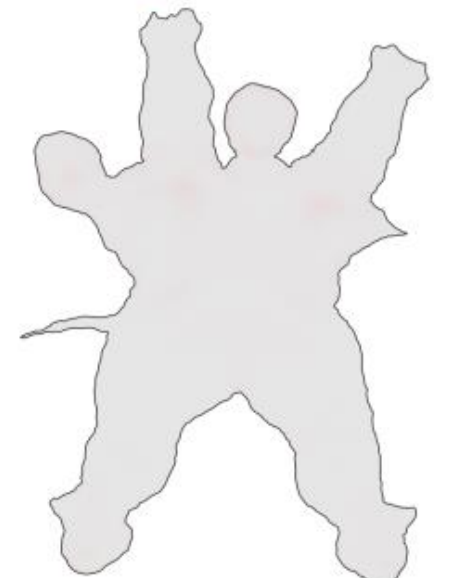


Mesh Cutting

- Low distortion
- As short as possible length



(a) (4.07/1.28/0.37)

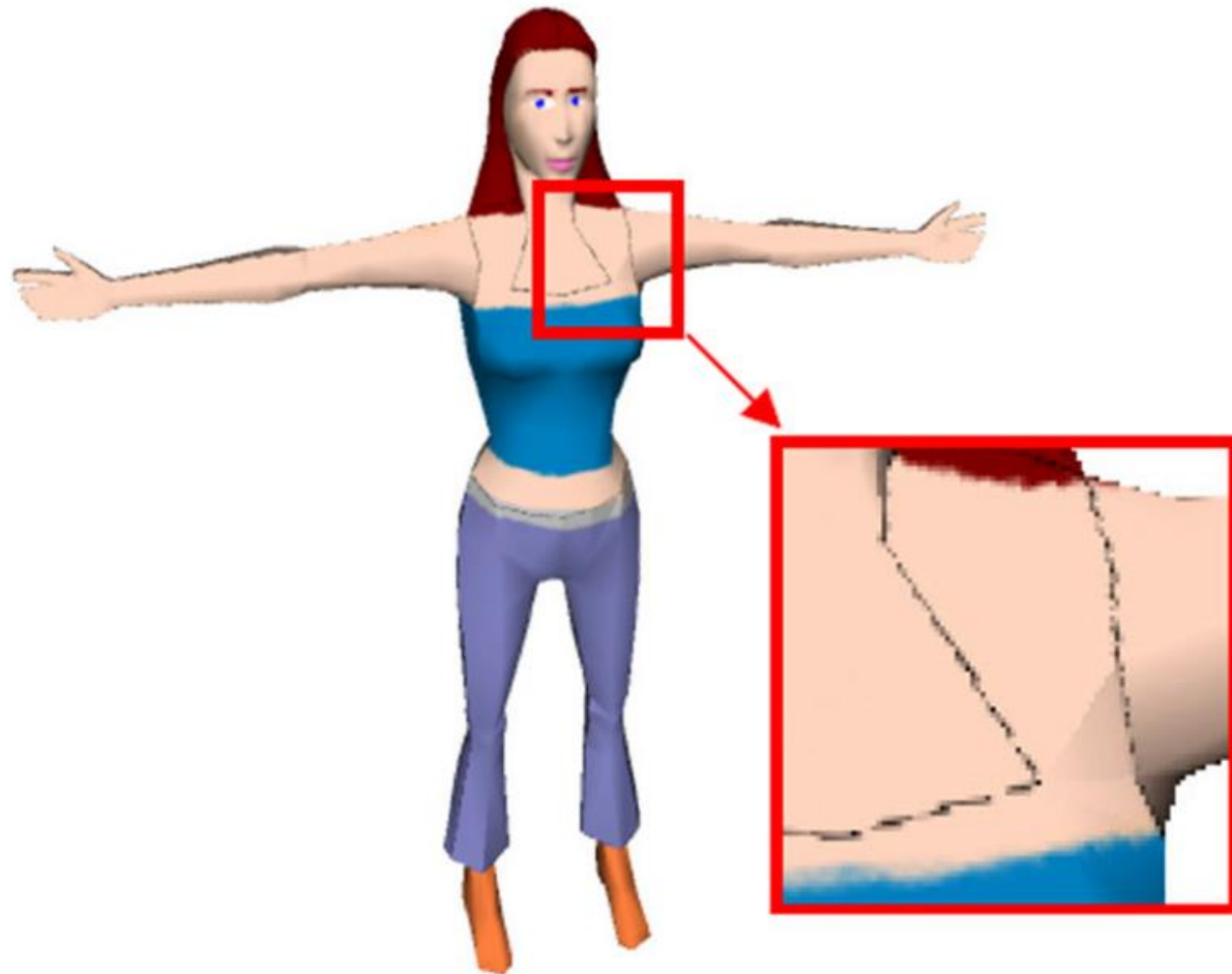


(b) (4.78/1.13/0.12)

Seams introduce filtering artifacts

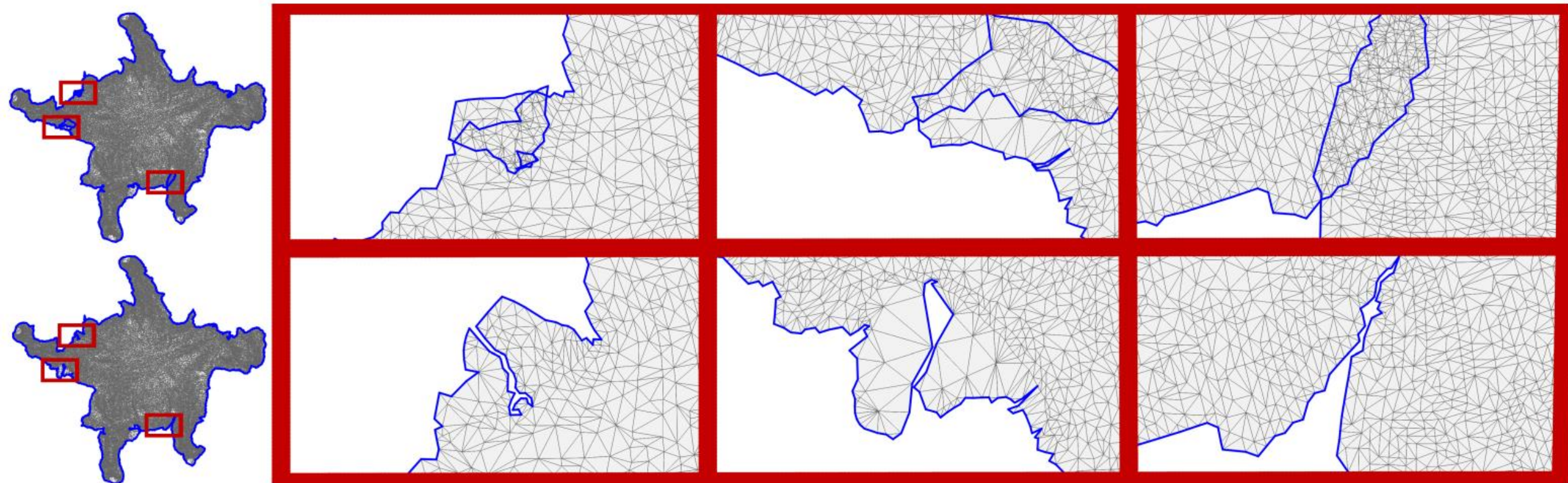


High-resolution texture



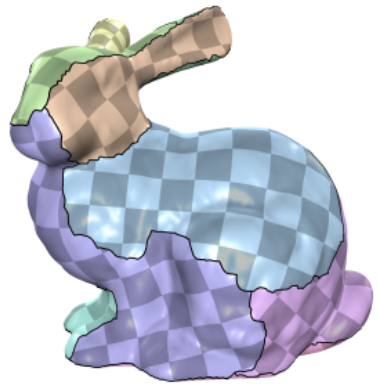
Parameterizations

- Bijective
- Low isometric distortion

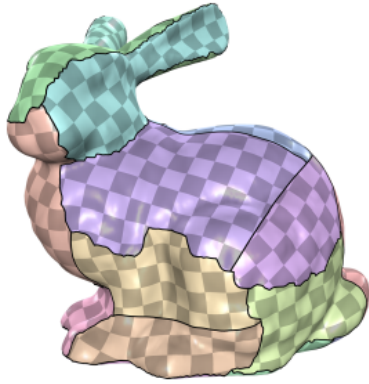
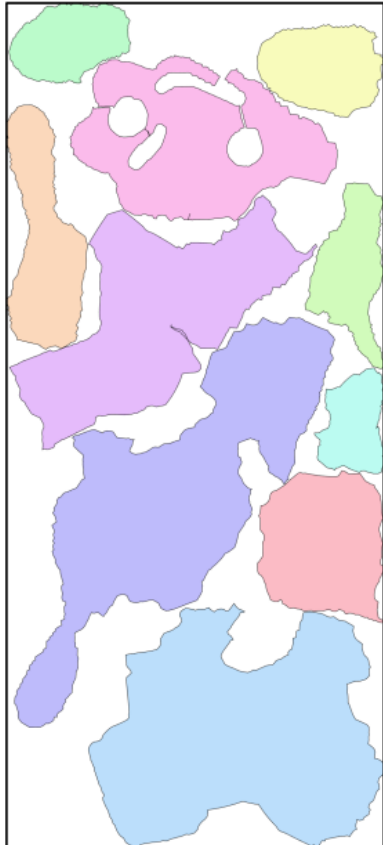


Packing

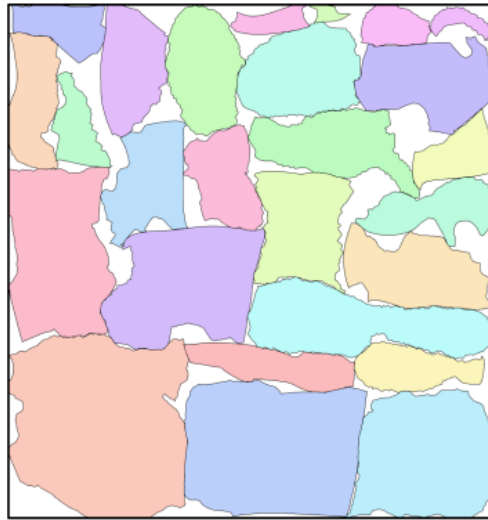
- High packing efficiency



PE = 66.0%
BL = 16.30
CN = 10

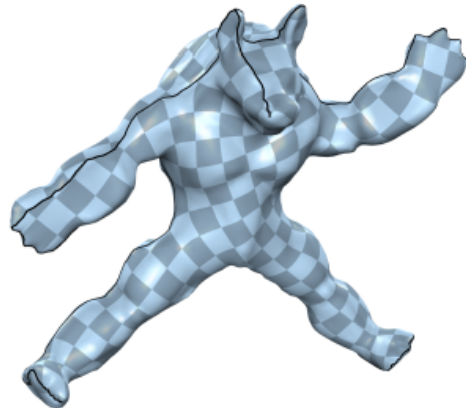
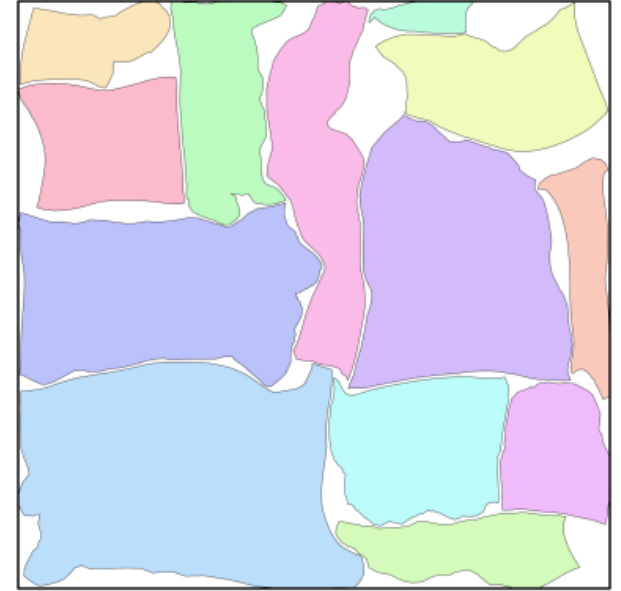
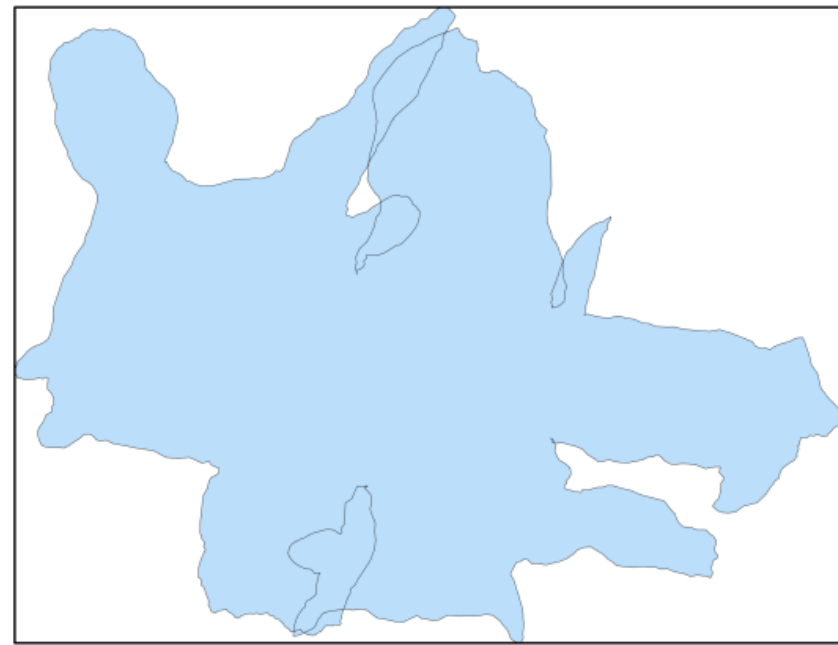


PE = 85.8%
BL = 21.24
CN = 26



Packing

- High packing efficiency



BL = 6.34
 $E_d = 1.040$

Input

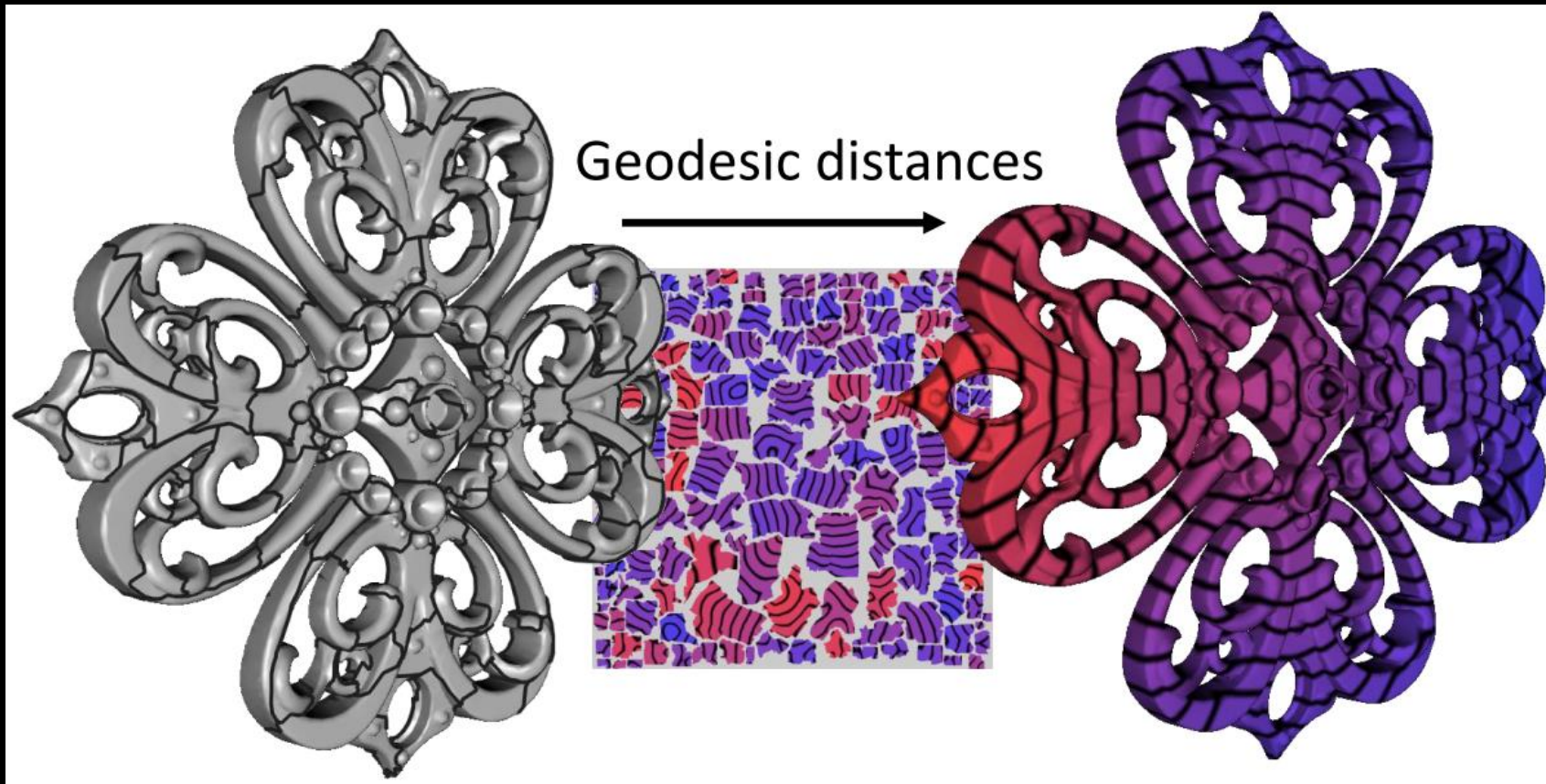


PE = 85.0%
BL = 11.45
CN = 13
 $E_d = 1.030$

Result

Applications

- Signal storage
- Geometric processing



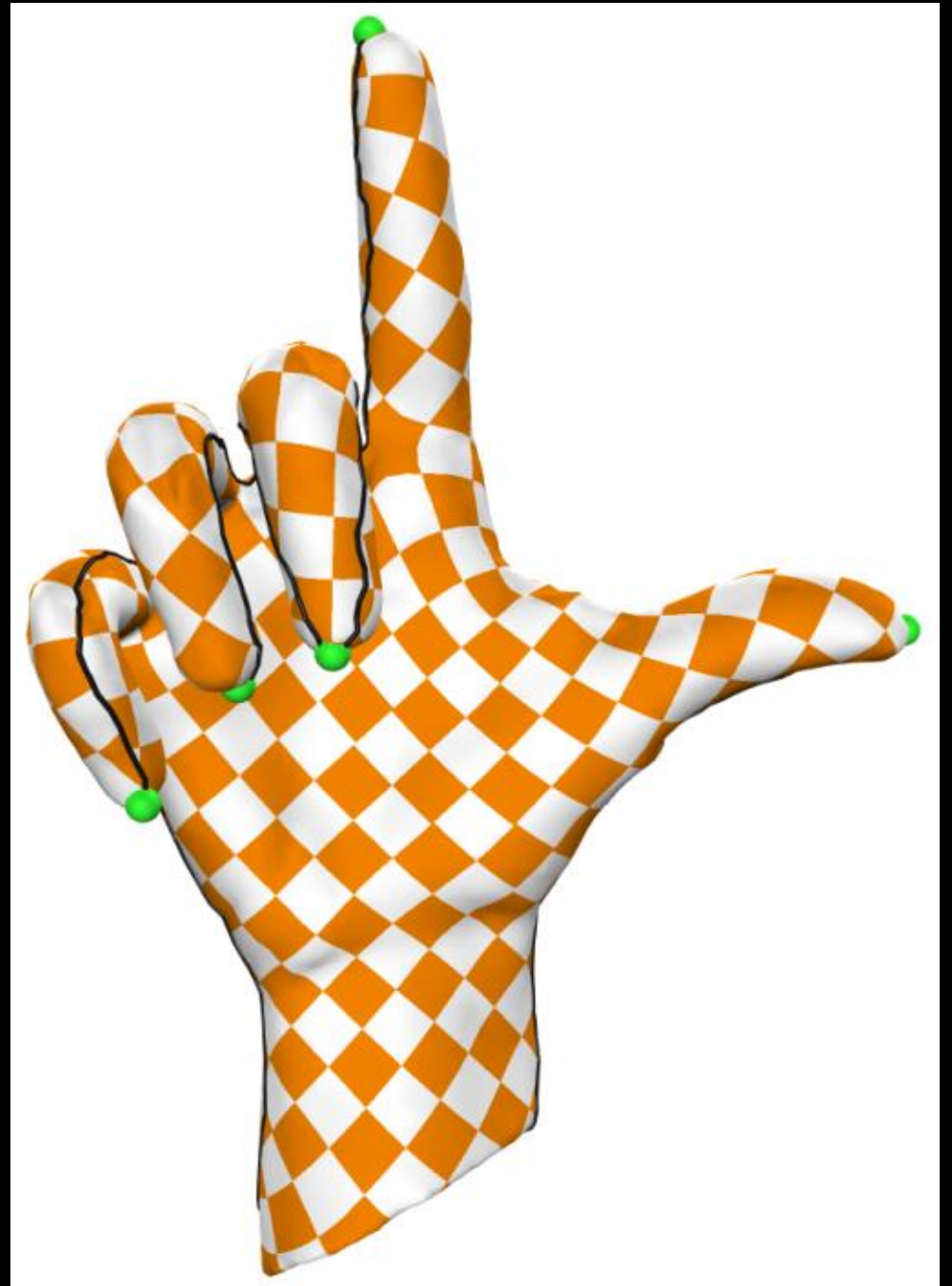
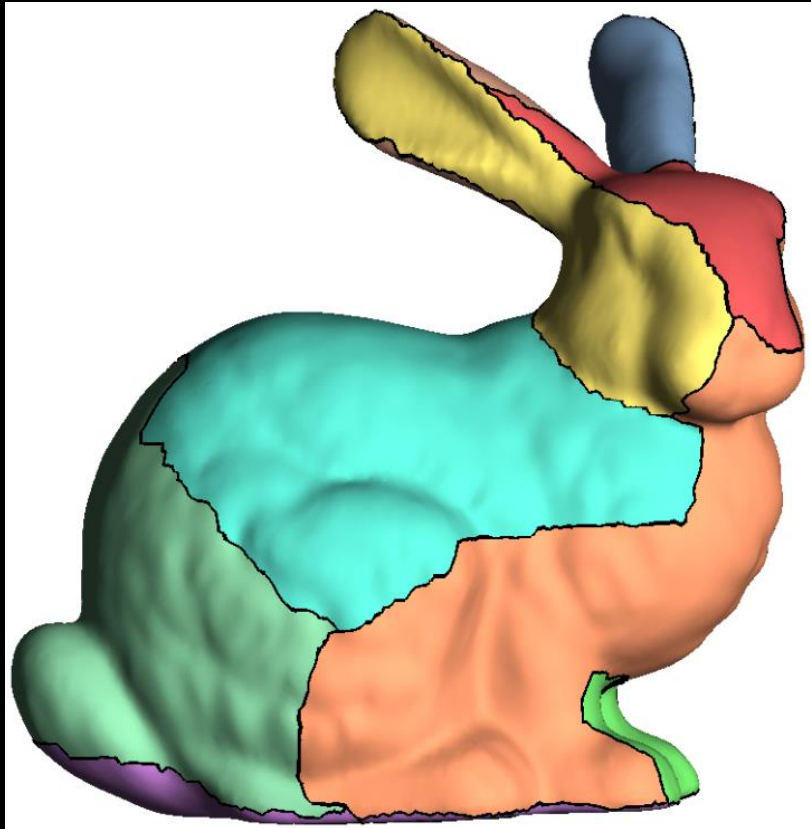
**Gradient-Domain Processing
within a Texture Atlas**

Outlines

- Definition
- **Mesh cutting**
- Chart parameterization
 - Bijective, low distortion
- Chart packing

Mesh cutting

- Points \rightarrow Paths
- Segmentation

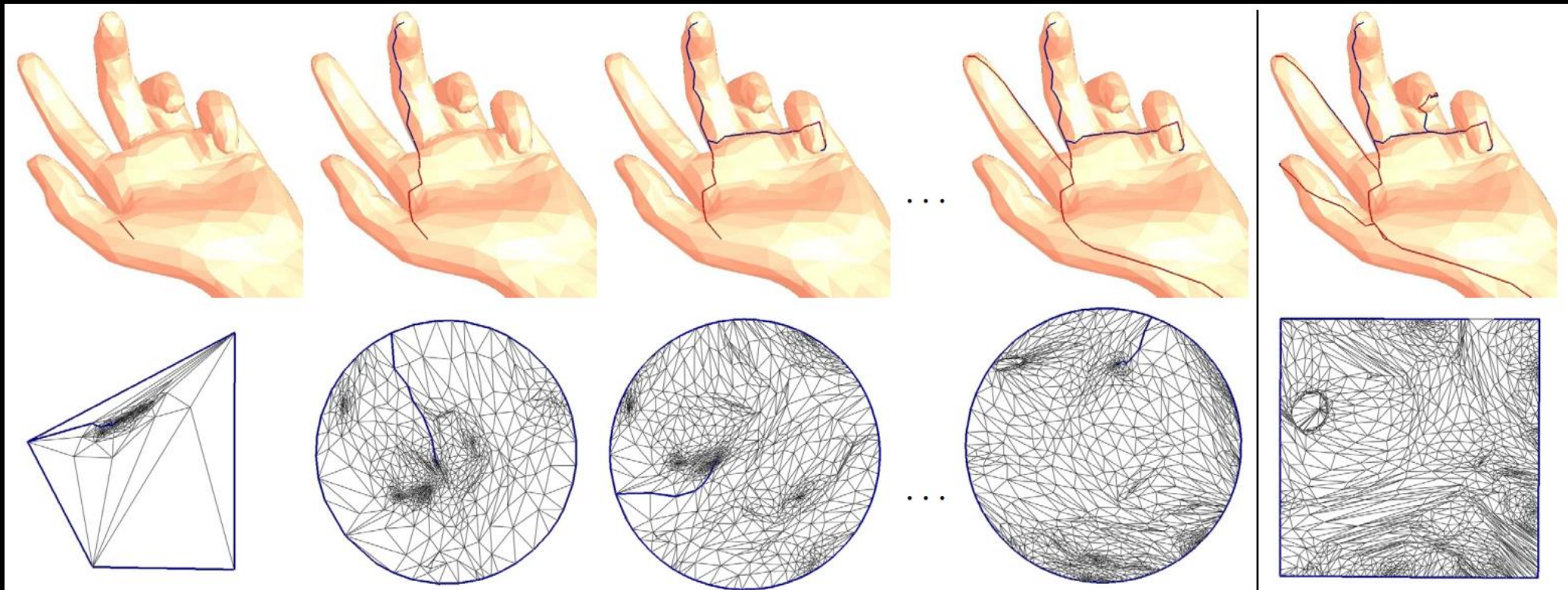


Distortion points

Geometry Images, SIGGRAPH 2002

- Iterative method

- Parameterize the mesh to the plane.
- Add the point of greatest isometric distortion.



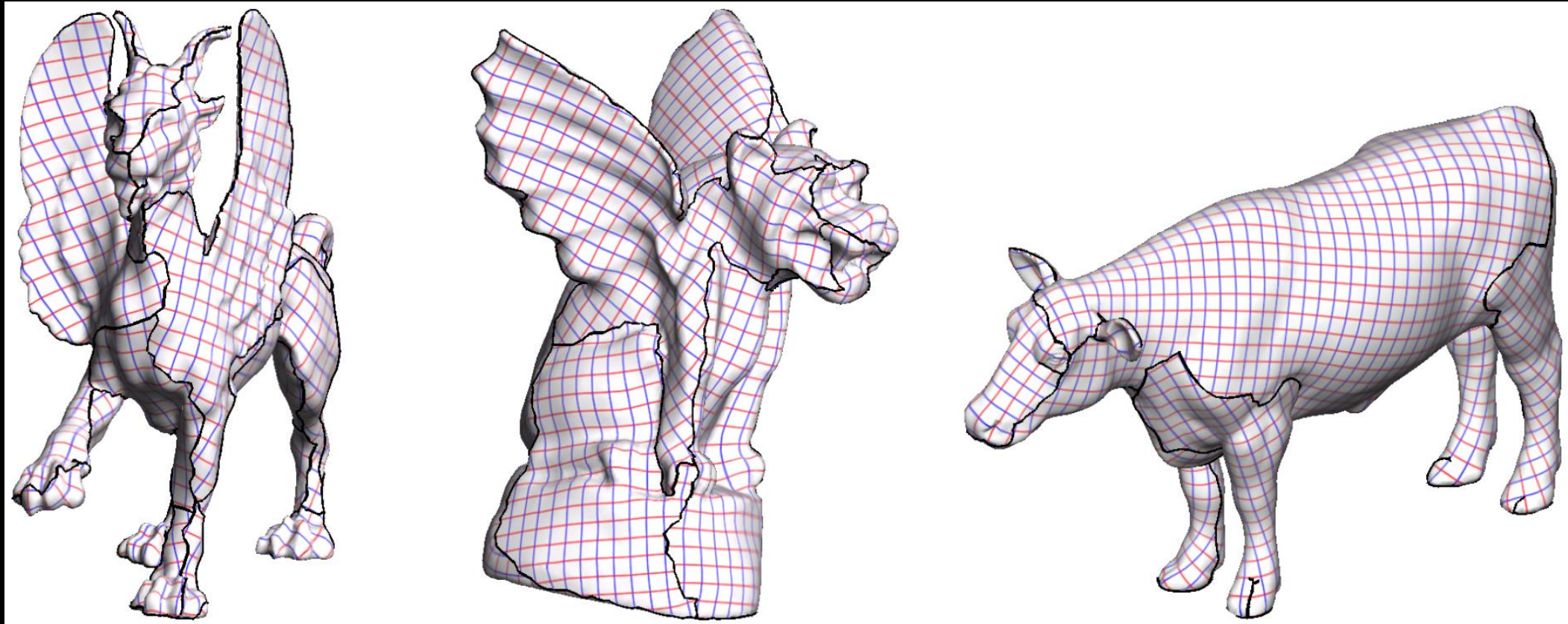
More methods

- Spanning tree seams for reducing parameterization distortion of triangulated surfaces, 2002
- Sphere-based cut construction for planar parameterizations, 2018

Segmentation

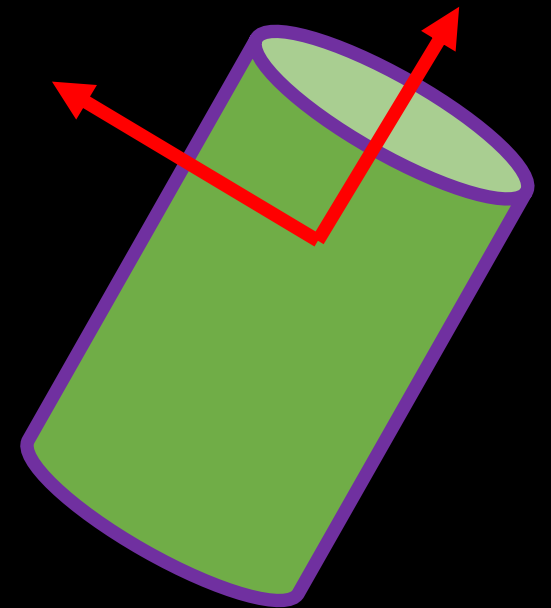
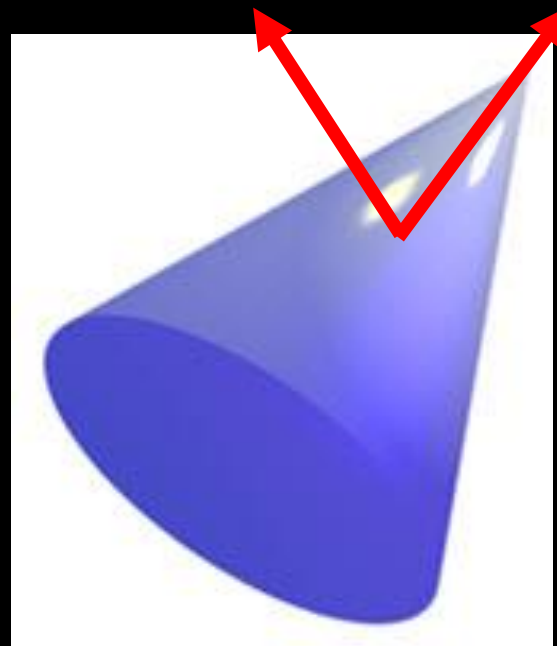
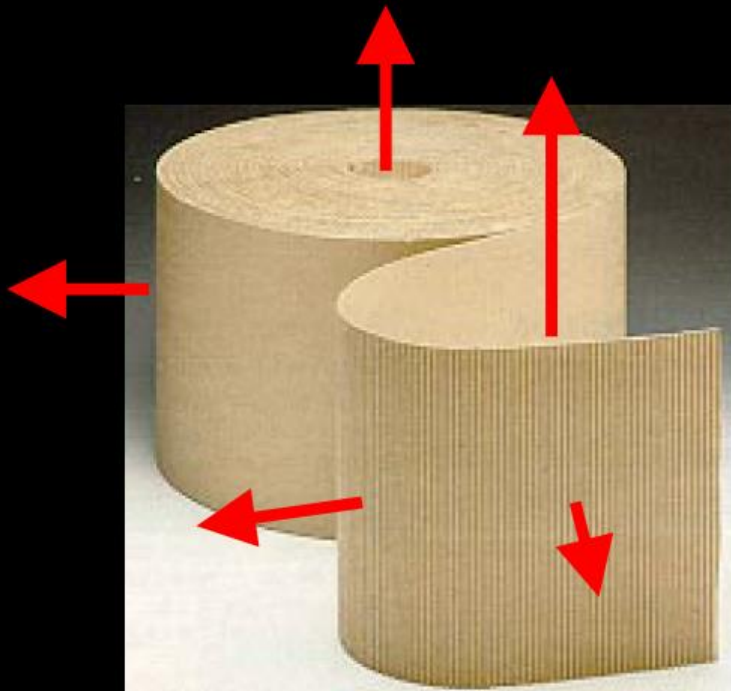
D-Charts: Quasi-Developable Mesh Segmentation, EG 2015

- Goal: mesh segmentation into compact charts that unfold with minimal distortion



Proxy

- Developable surfaces of constant slope
- Constant angle between surface normal and axis
- Proxy: $\langle axis, angle \rangle, \langle N_c, \theta_c \rangle$



Fitting error

- Measures how well triangle fits a chart

$$F(C, t) = (N_c \cdot n_t - \cos\theta_c)^2$$

- Combine with compactness

$$C(C, t) = \frac{\pi D(S_c, t)^2}{A_c}$$

✓ S_c is the seed triangle of the given chart

✓ $D(S_c, t)$ is the length of the shortest path (inside the chart) between the two triangles

✓ A_c is the area of chart C

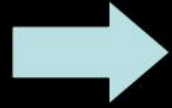
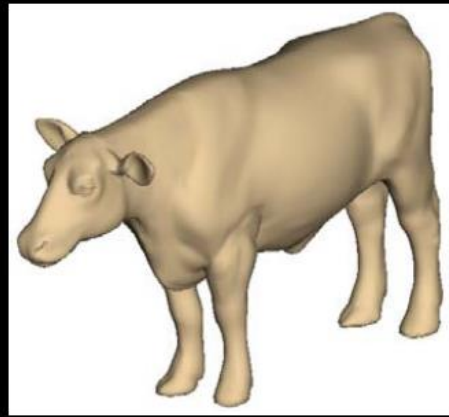
- Cost function

$$Cost(C, t) = A_t F(C, t)^\alpha C(C, t)^\beta$$

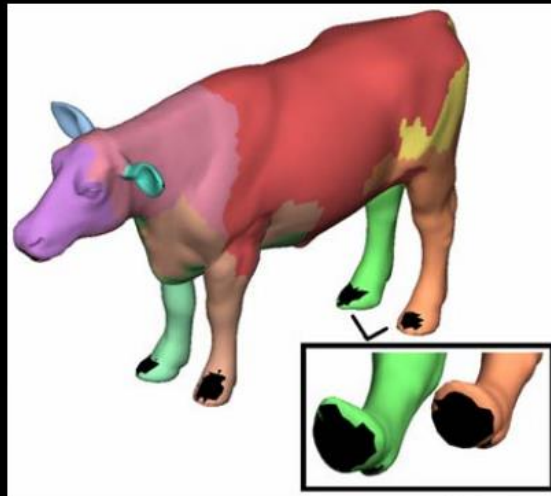
Segmentation method

- Lloyd algorithm
 - 1. Select random triangles to act as seeds
 - 2. Grow charts around seeds using a greedy approach
 - 3. Find new proxy for each chart
 - 4. Repeat from step 2 until convergence
- K-means
- CVT

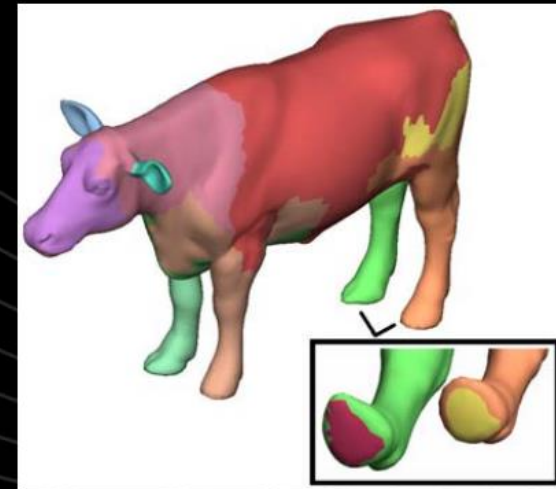
Algorithm overview



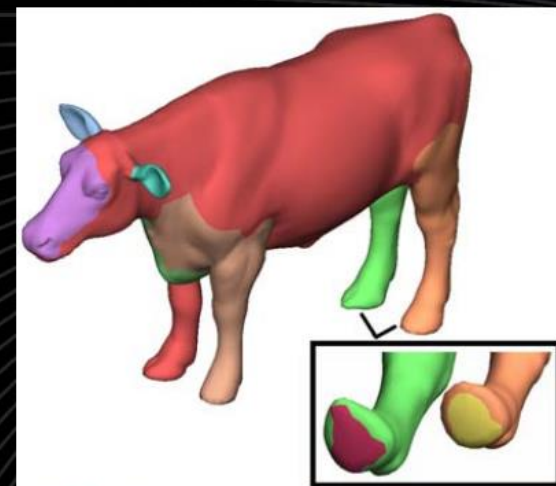
Bounded
Lloyd
iterations



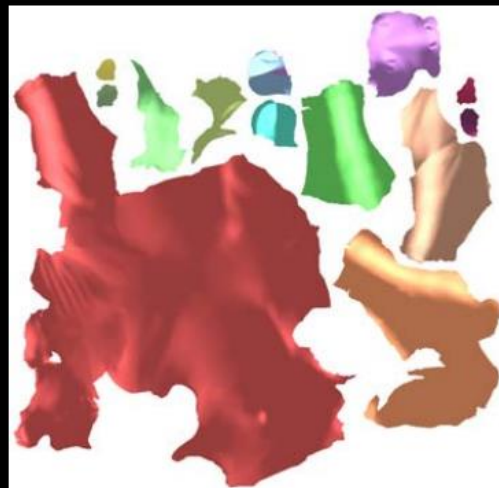
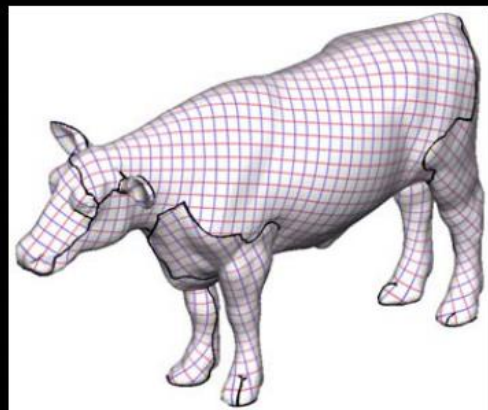
Hole
Filling



Merging

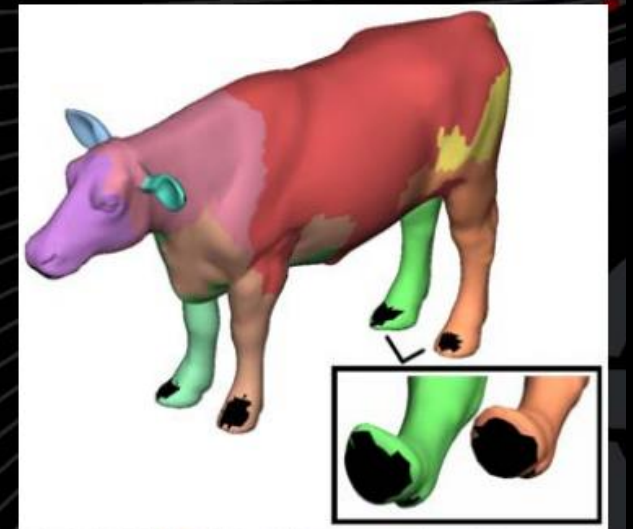
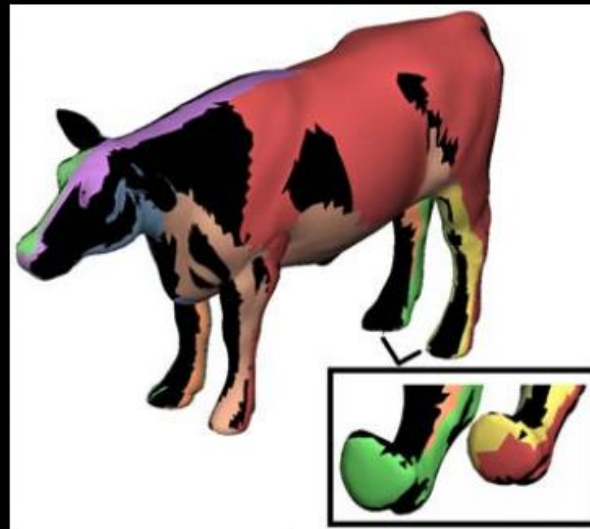
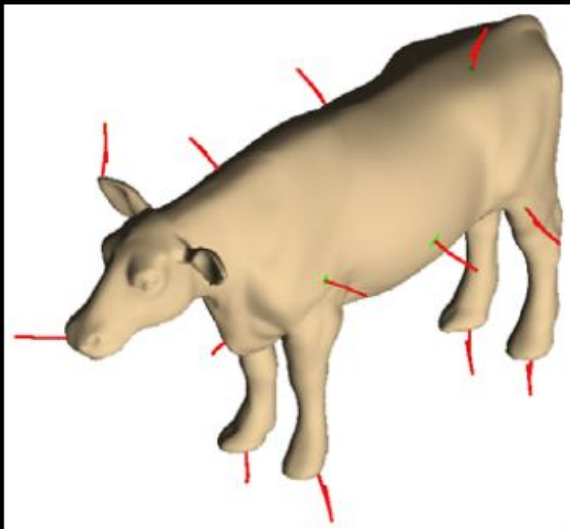


Post-Processing
&
Parameterization



Bounded Lloyd iterations

- Initialization
 - Random / Furthest point seeds
 - Compute initial proxy
- Bounded Growing/Reseeding iterations
- Termination



Bounded Lloyd iterations – Growing

- Use greedy approach
 - Prioritize by $Cost(C, t)$
- Bound Fitting Error
 - Guarantee (nearly) developable charts
 - $F(C, t) < F_{max}$

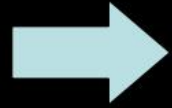
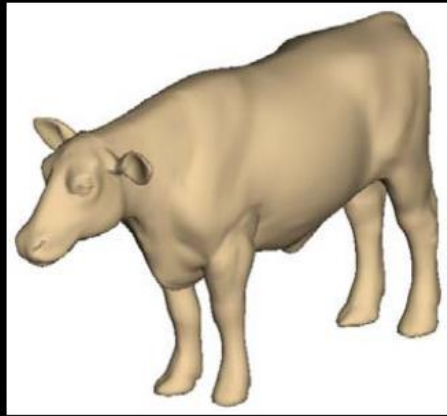
Bound Lloyd iterations – Reseeding

- Find new proxy

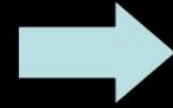
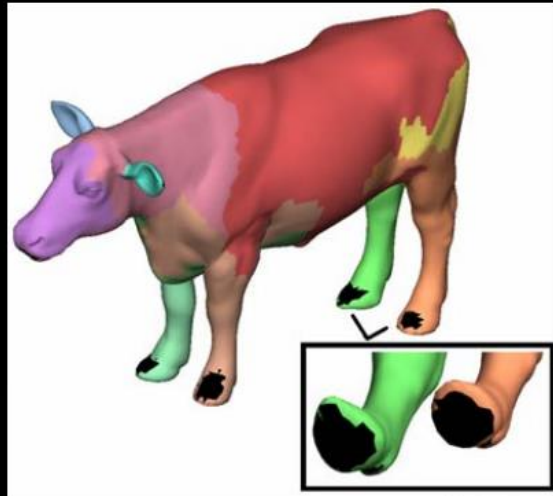
$$\min_{N_c, \theta_c} \frac{1}{A_c} \sum_{t \in C} A_t F(C, t) \quad s.t. \quad ||N_c|| = 1$$

- Find new seed
 - Minimal Fitting Error
 - Close to center of chart
 - To find such seeds, we examine the first k triangles in the chart with minimal fitting error ($k = 10$ in all our examples), and then select the one closest to the center of the chart.

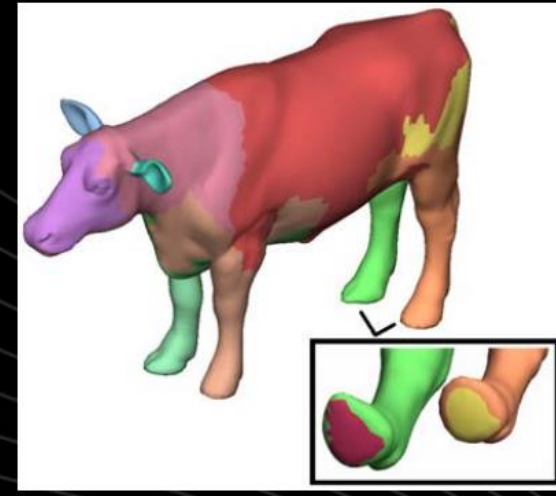
Algorithm overview



Bounded
Lloyd
iterations



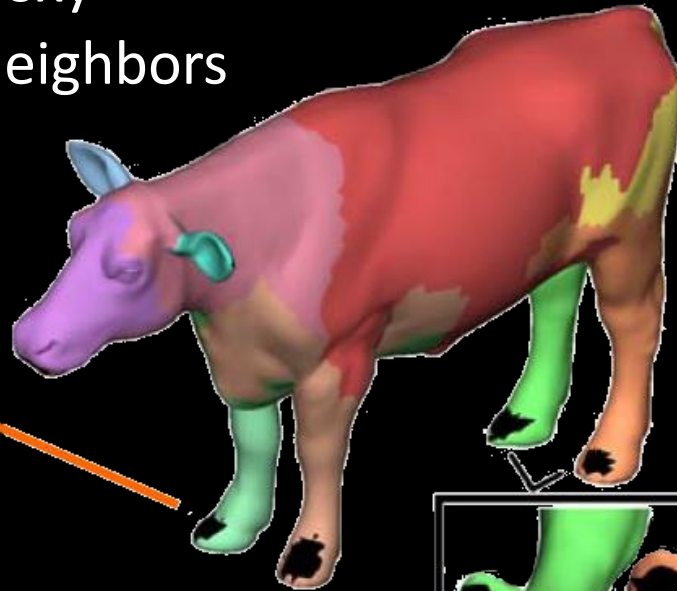
Hole
Filling



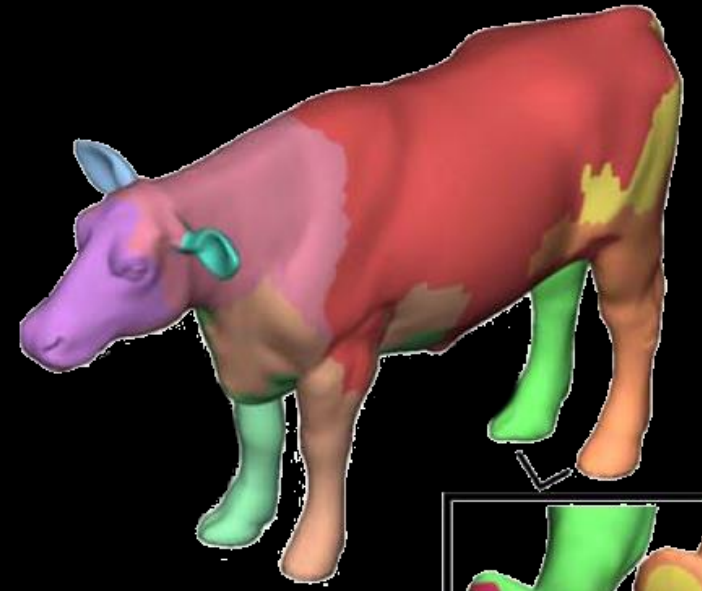
Hole filling

- Bound on Fitting Error
 - Unclassified triangles
- Fill holes
 - Large holes → New proxy
 - Small holes → Grow neighbors

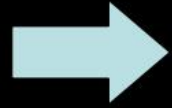
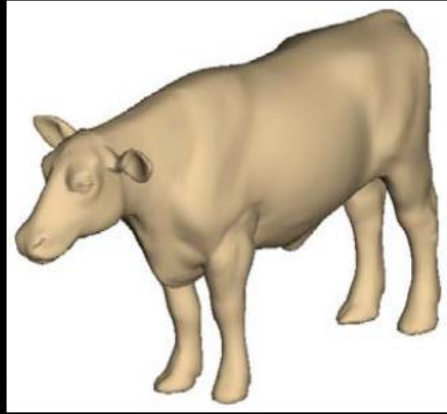
Small



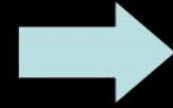
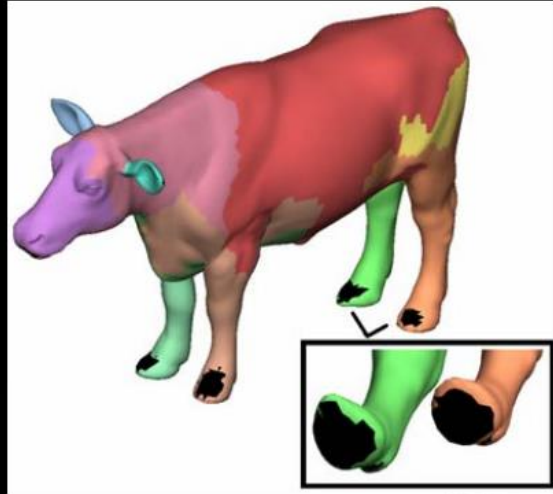
Large



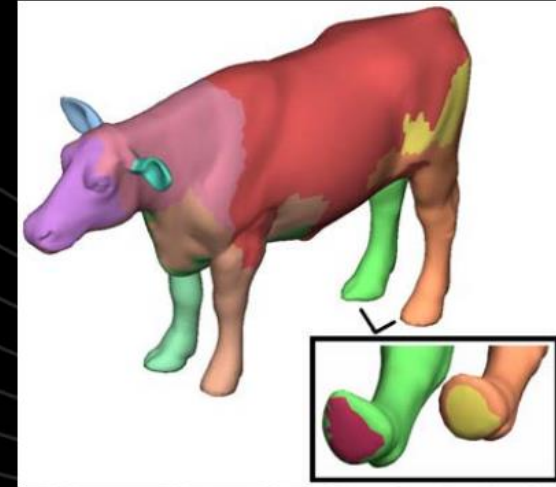
Algorithm overview



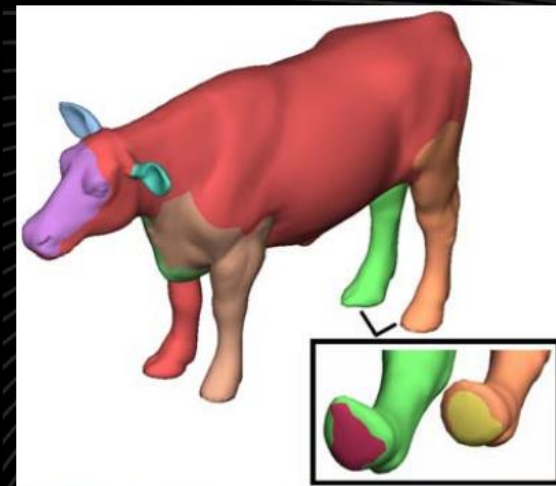
Bounded
Lloyd
iterations



Hole
Filling

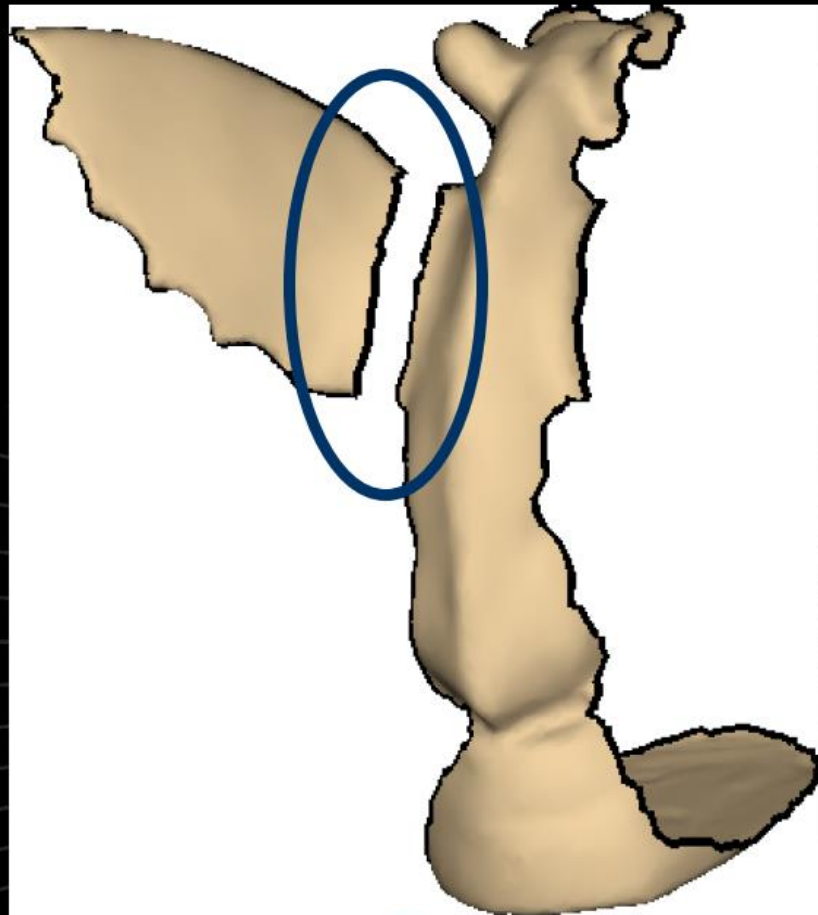


Merging

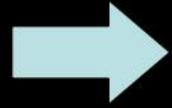
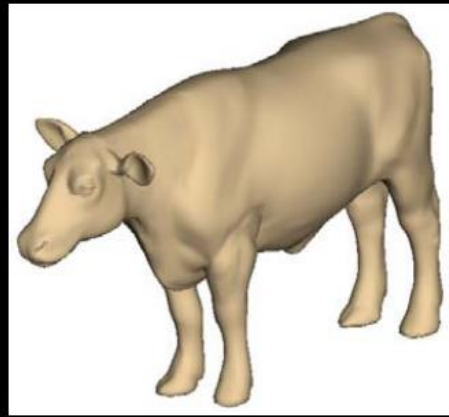


Merging

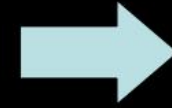
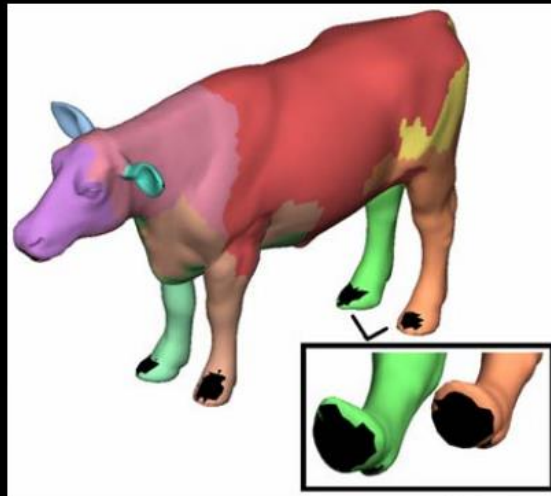
- Broaden set of captured developable surfaces
- Reduce number of charts



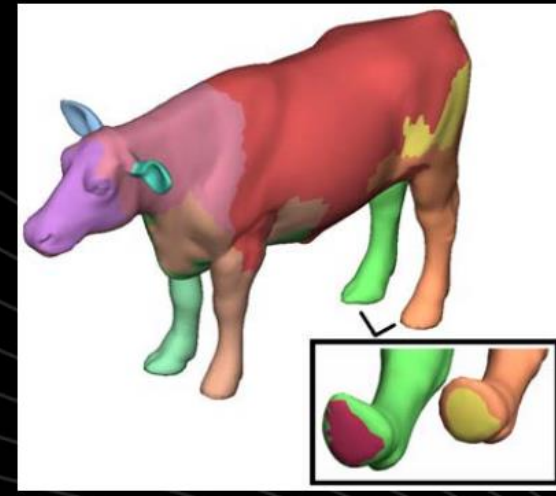
Algorithm overview



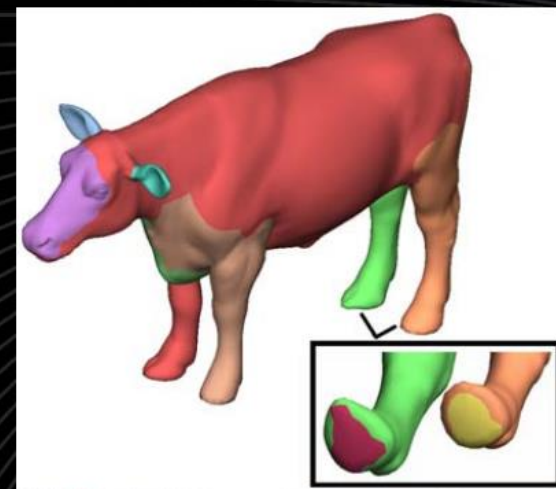
Bounded
Lloyd
iterations



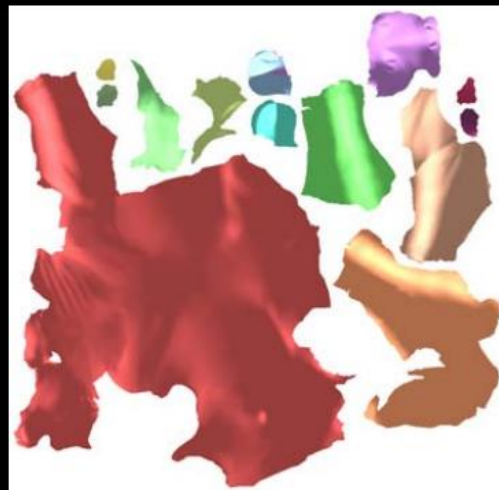
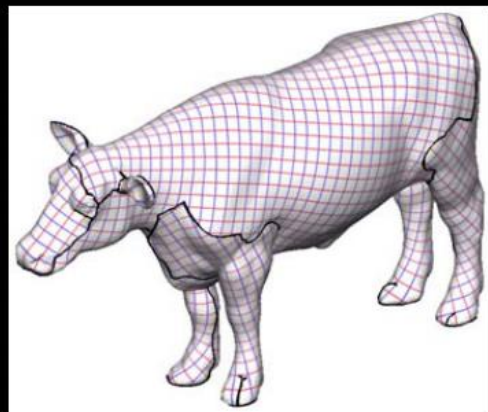
Hole
Filling



Merging



Post-Processing
&
Parameterization



Post processing

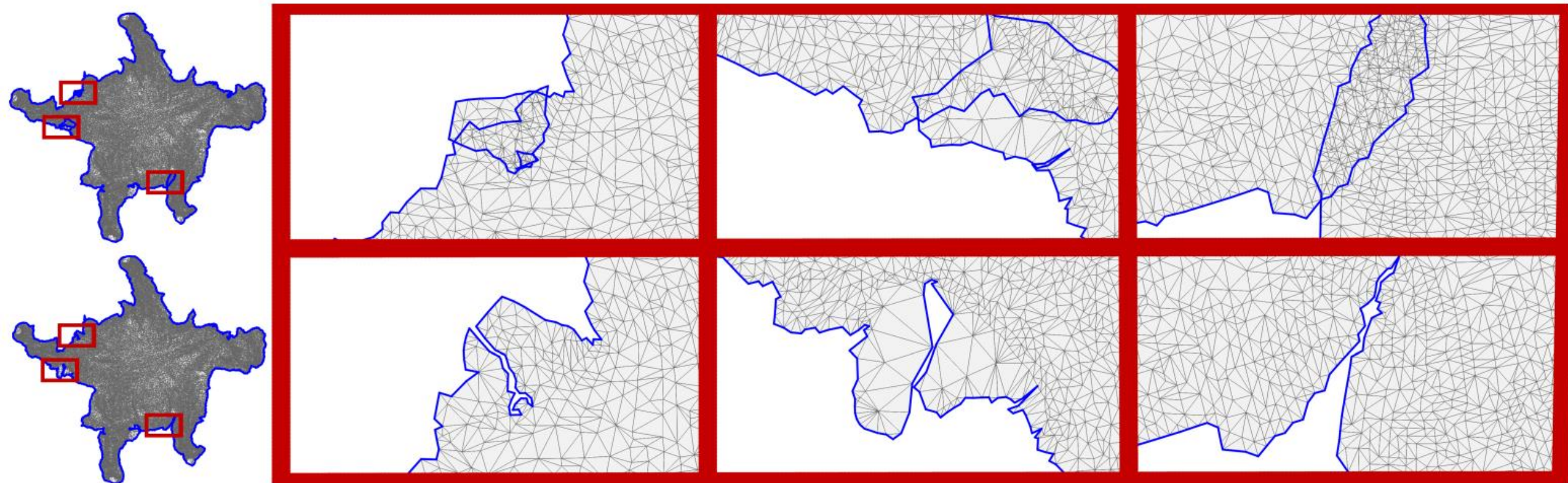
- Straighten boundaries
- Darts/Gussets relax stress
 - Add seams toward high error regions
- Verify disc topology
- Parameterization

Outlines

- Definition
- Mesh cutting
- **Chart parameterization**
 - Bijective, low distortion
- Chart packing

Bijection

- Preserving orientations
- No intersections of boundary



Barrier

Bijective Parameterization with Free Boundaries, SIGGRAPH 2015

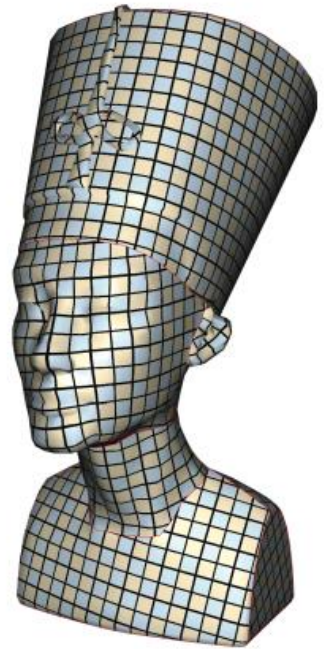
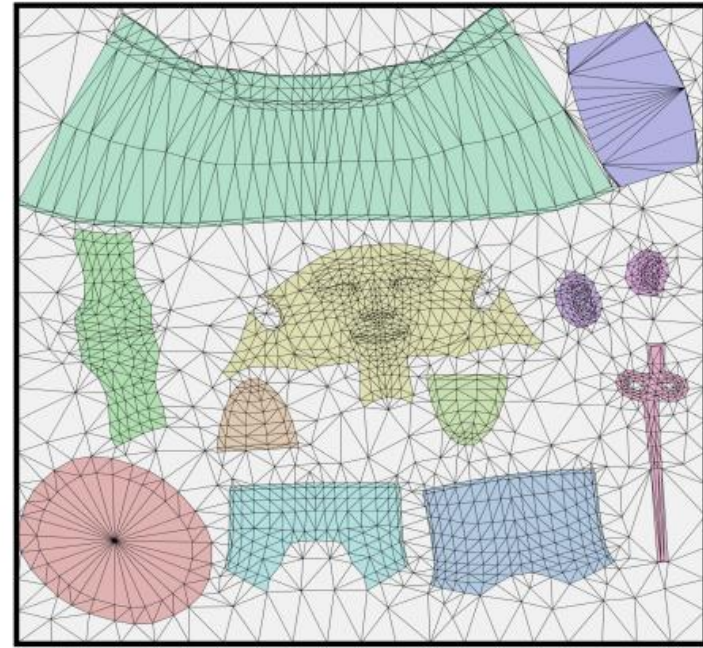
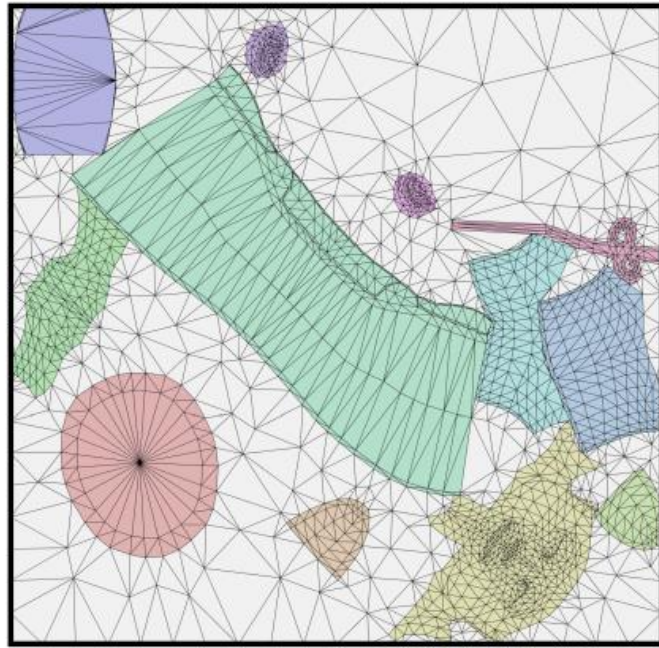
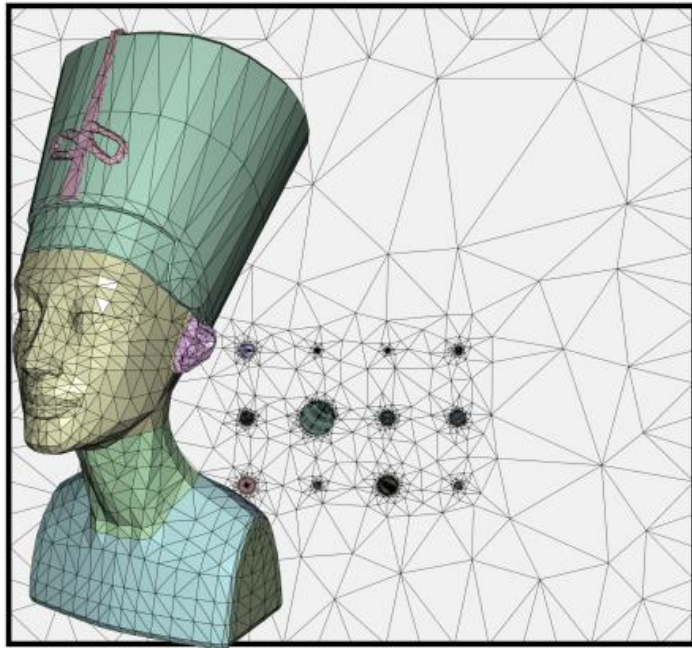
- For each boundary edge with vertices U_1, U_2 , we associate a barrier function:

$$\max\left(0, \frac{\varepsilon}{\text{dist}(U_1, U_2, U_i)} - 1\right)^2$$

where, $\text{dist}(U_1, U_2, U_i)$ measures the distance from a boundary point $U_{i \neq 1,2}$ to the edge (U_1, U_2) .

Scaffold

Simplicial Complex Augmentation Framework for Bijective Maps, SIGGRAPH Asia 2017



Outlines

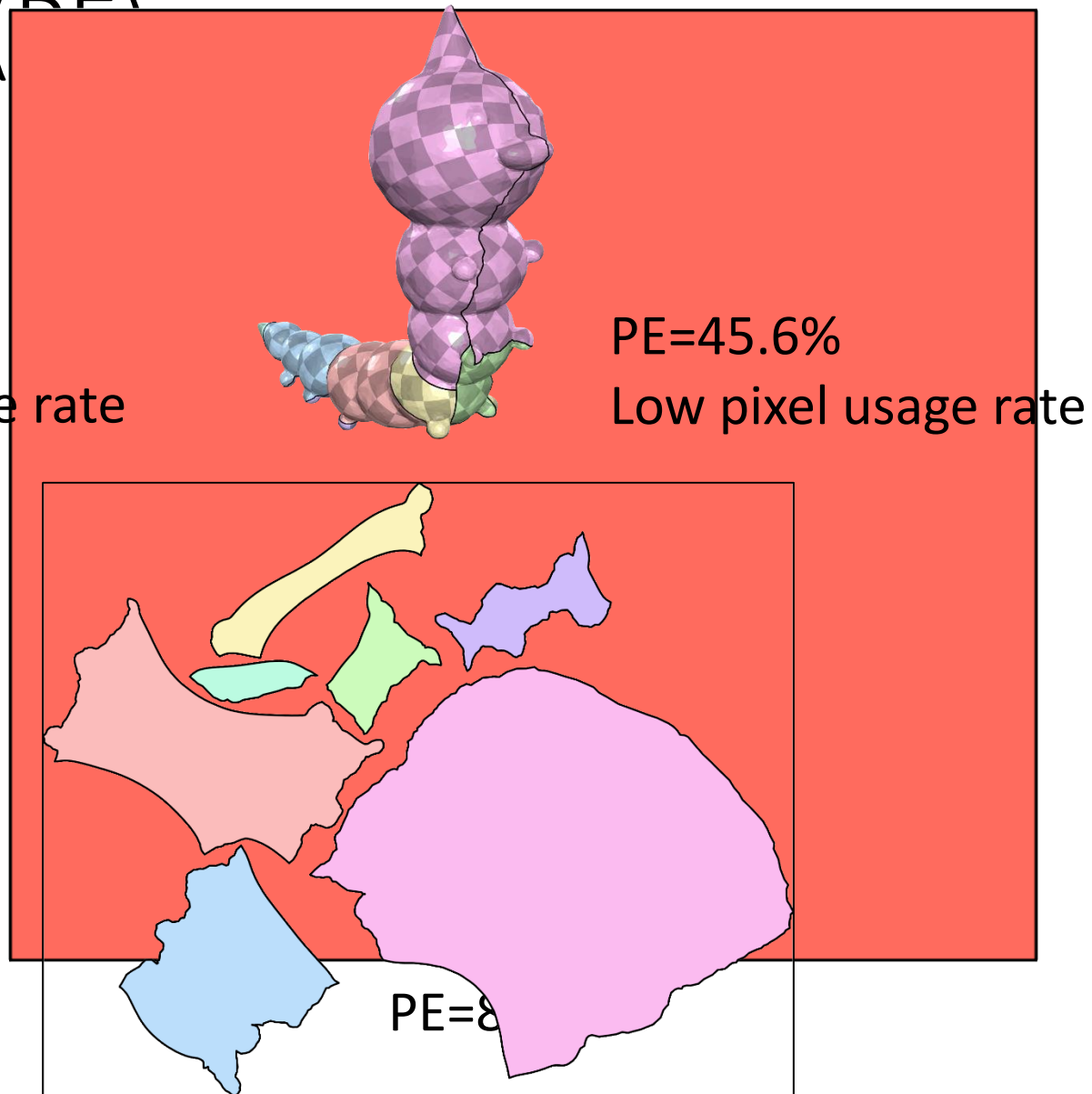
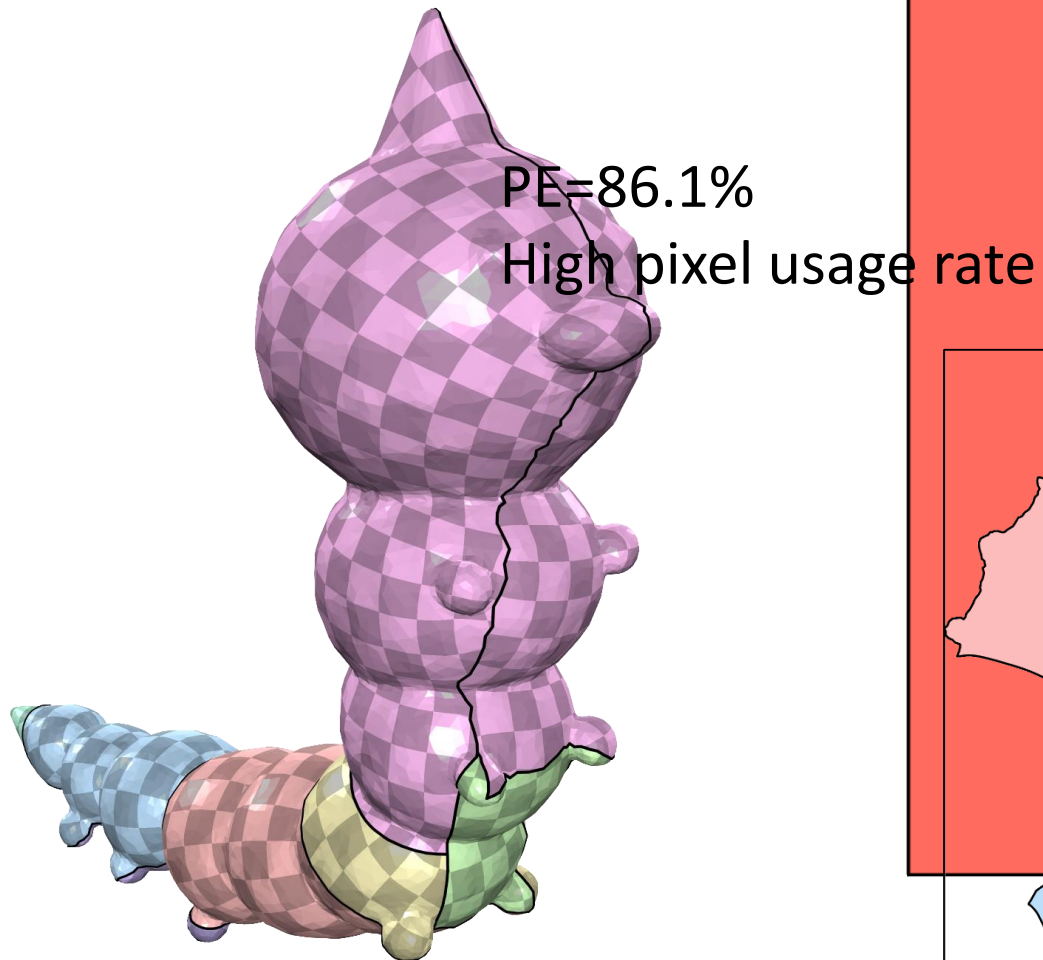
- Definition
- Mesh cutting
- Chart parameterization
 - Bijective, low distortion
- Chart packing

Atlas Refinement with Bounded Packing Efficiency

Hao-Yu Liu, Xiao-Ming Fu, Chunyang Ye, Shuangming Chai, Ligang Liu

ACM Transactions on Graphics (SIGGRAPH) 38(4), 2019.

Packing Efficiency (PE)



Packing Efficiency (PE)

Maximizing atlas packing efficiency is NP-hard!

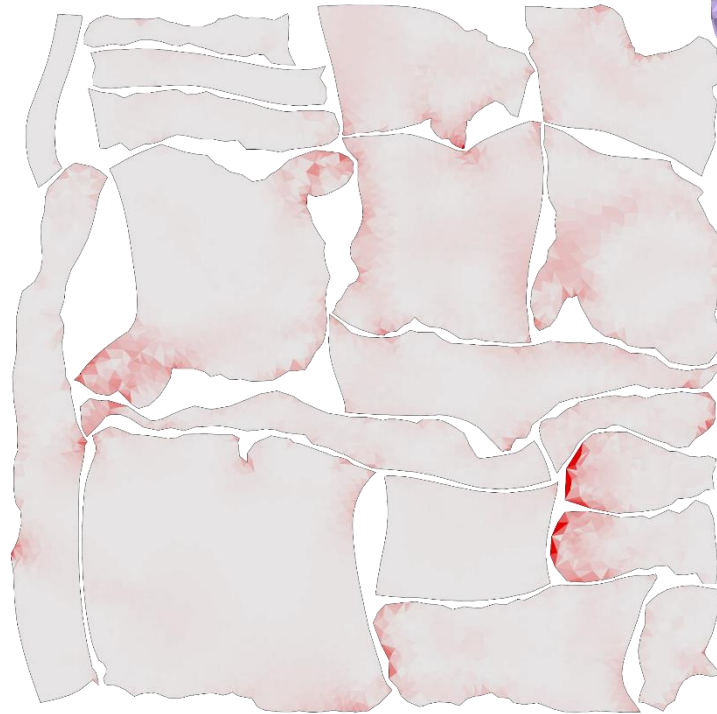
[Garey and Johnson 1979; Milenkovic 1999]

Other Requirements

- Low distortion



High Distortion



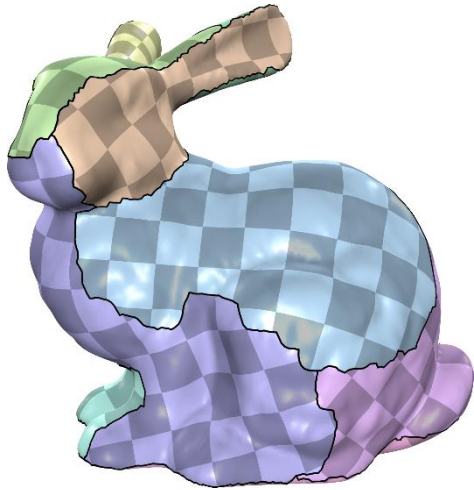
Low Distortion

Other Requirements

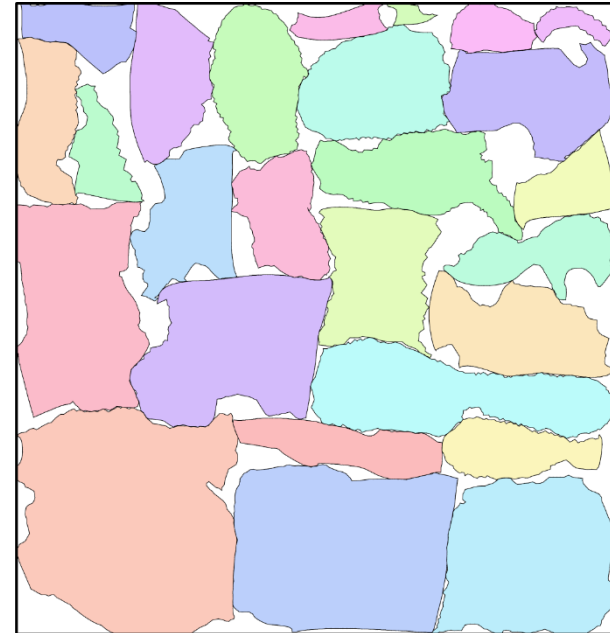
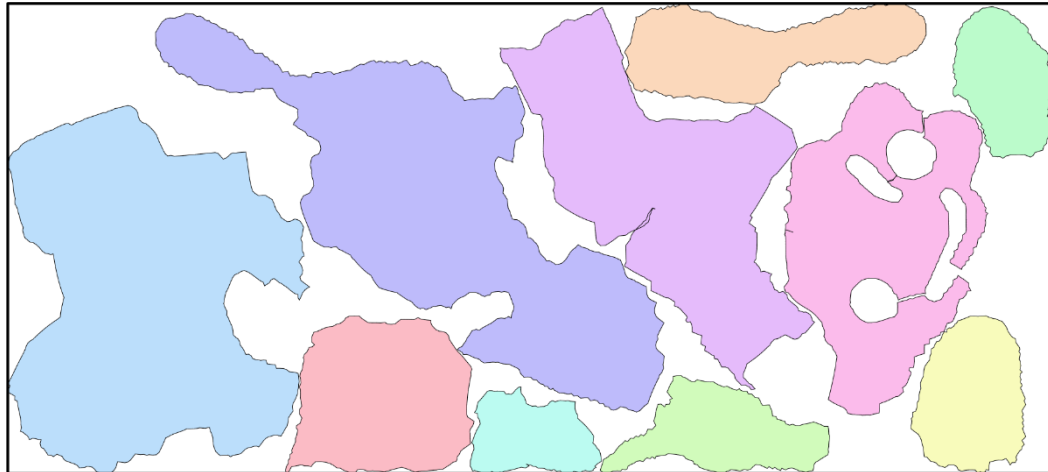
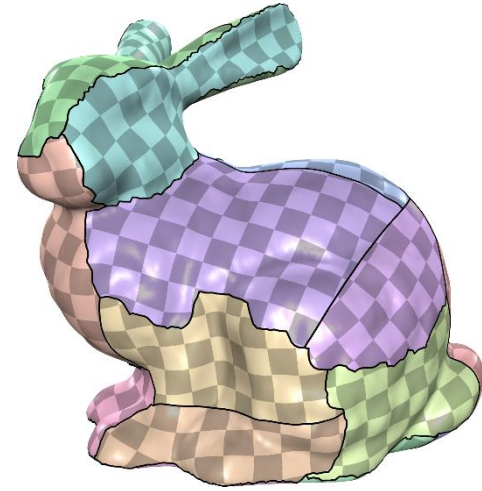
- Low distortion
 - [Golla et al. 2018; Liu et al. 2018; Shtengel et al. 2017; Zhu et al. 2018]
- Consistent orientation
 - [Floater 2003; Tutte 1963; Claici et al. 2017; Hormann and Greiner 2000; Rabinovich et al. 2017; Schüller et al. 2013]
- Overlap free
 - [Jiang et al. 2017; Smith and Schaefer 2015]
- Low boundary length
 - [Li et al. 2018; Poranne et al. 2017; Sorkine et al. 2002]

These methods do not consider PE!

Atlas Refinement



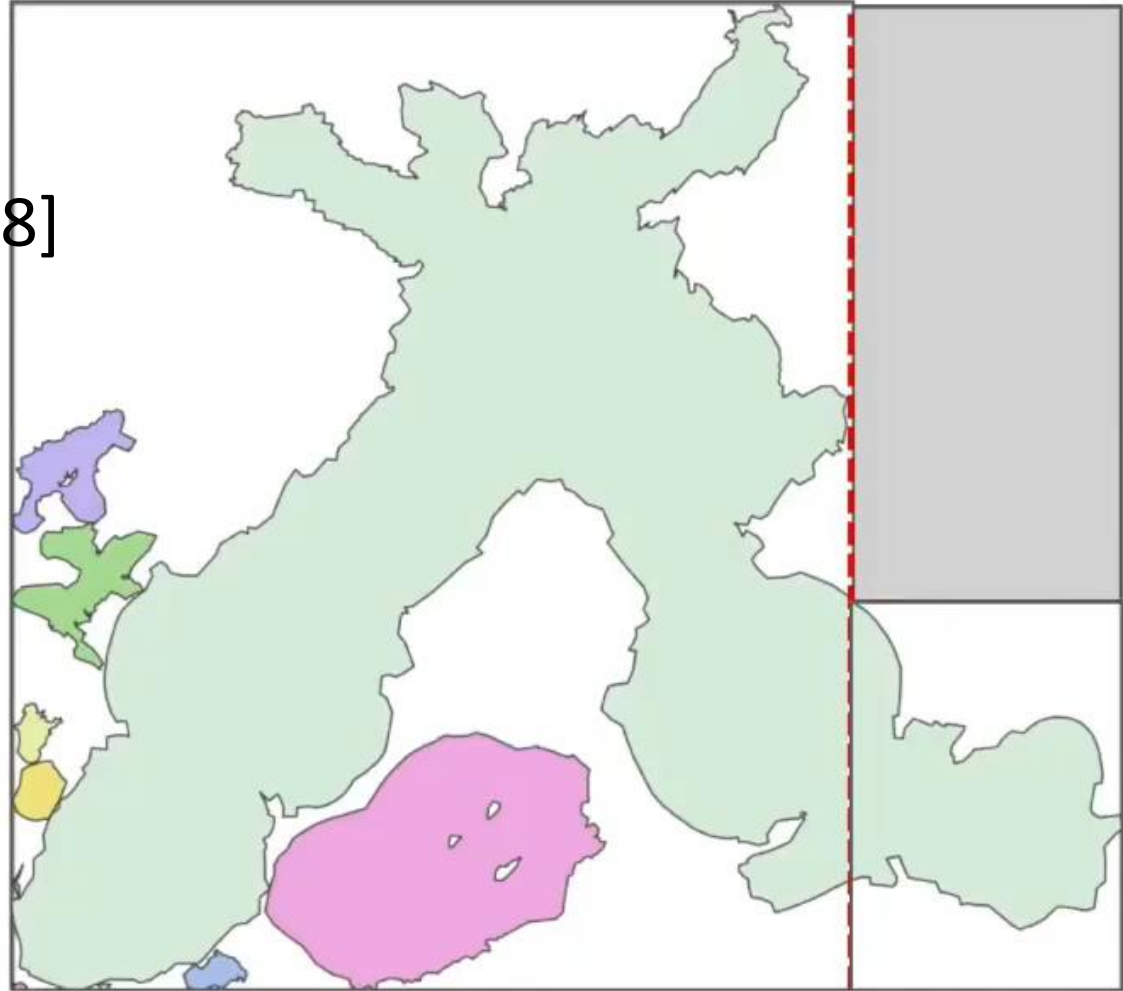
Input



No overlap
High PE

Previous Work

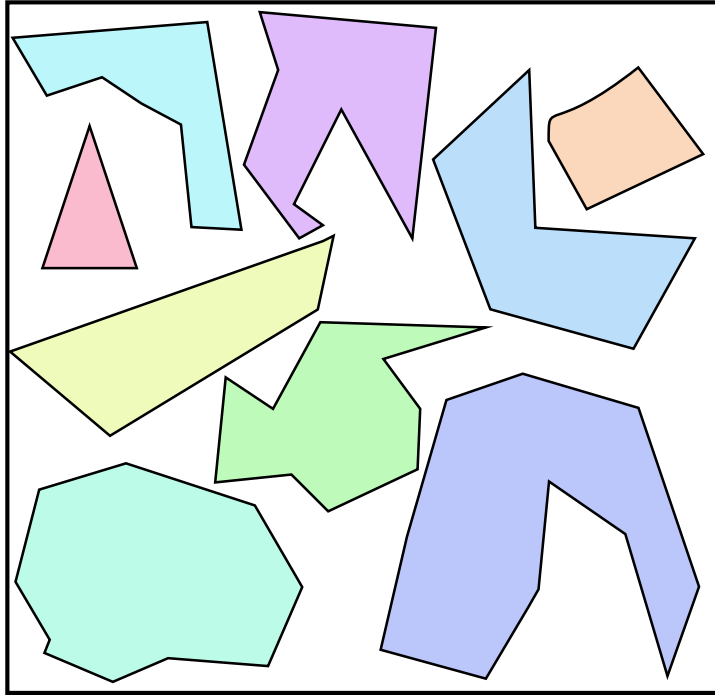
Box Cutter [Limper et al. 2018]



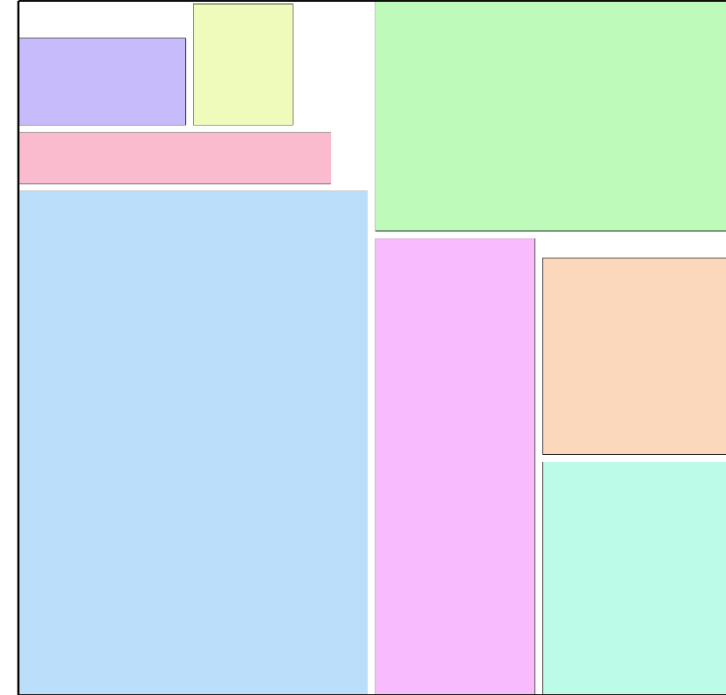
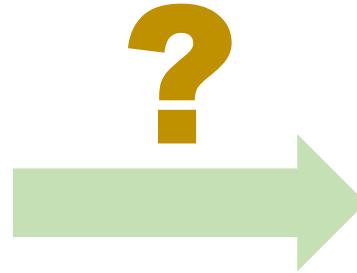
No guarantee for a high PE result!

Motivation

Packing Problems

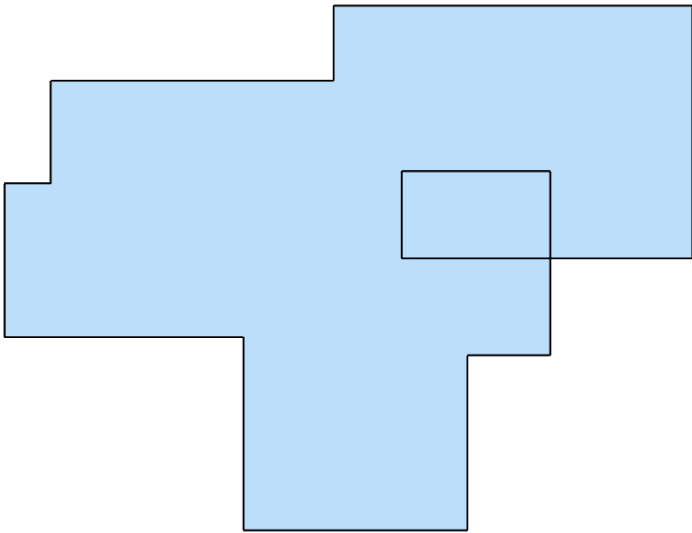


Irregular shapes
Hard to achieve high PE

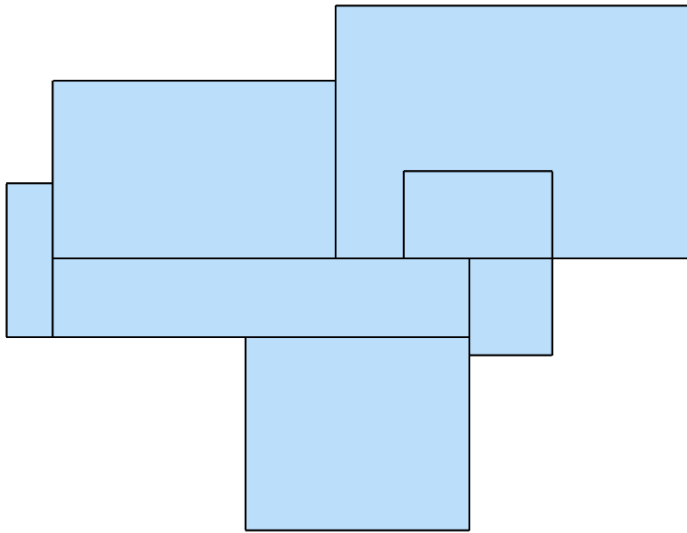


Rectangles
Simple to achieve high PE
Widely used in practice

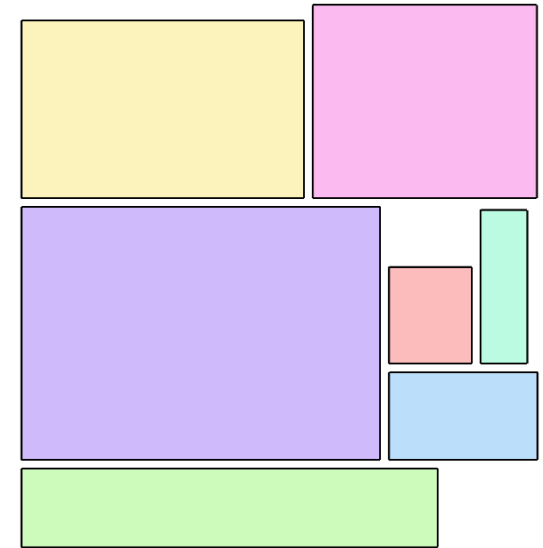
Axis-Aligned Structure



Axis-aligned structure

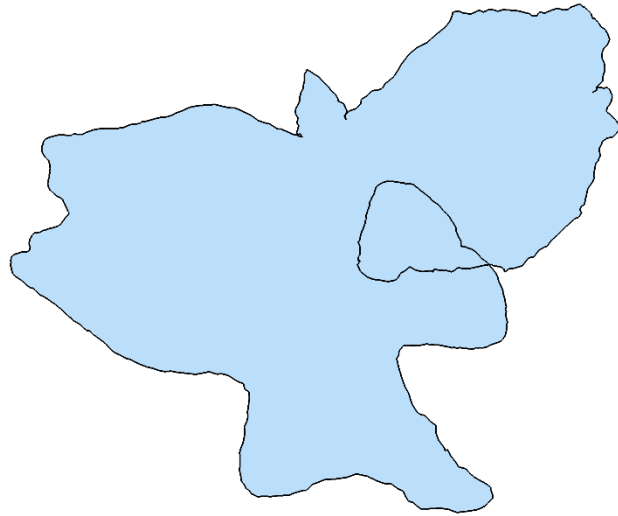


Rectangle decomposition



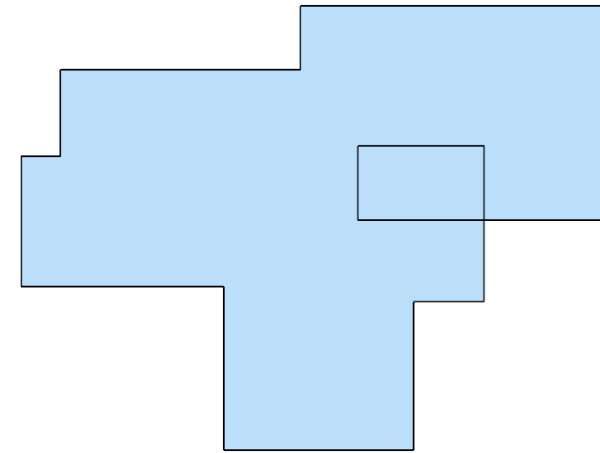
High PE (87.6%)!

General Cases



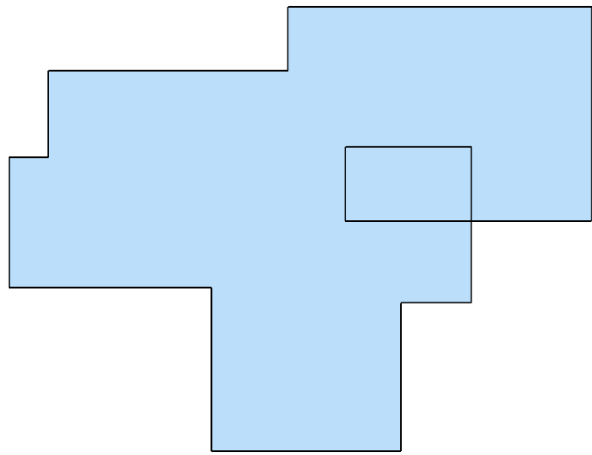
Not axis-aligned

Axis-aligned deformation

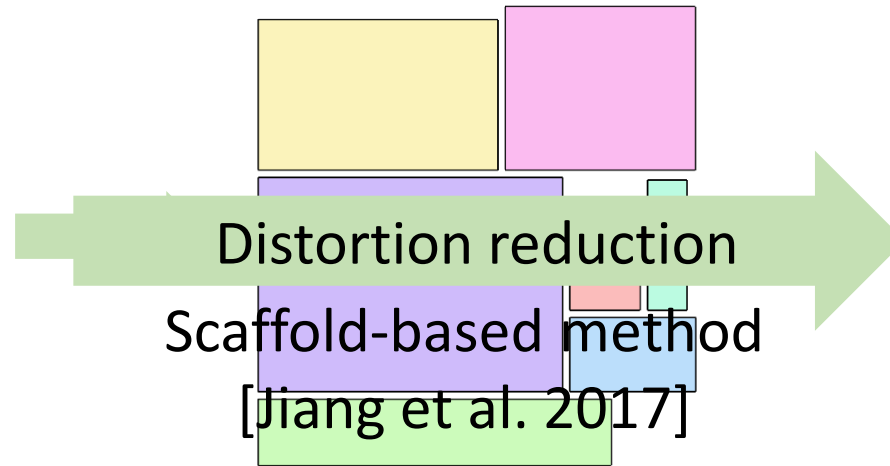


Axis-aligned
Higher distortion

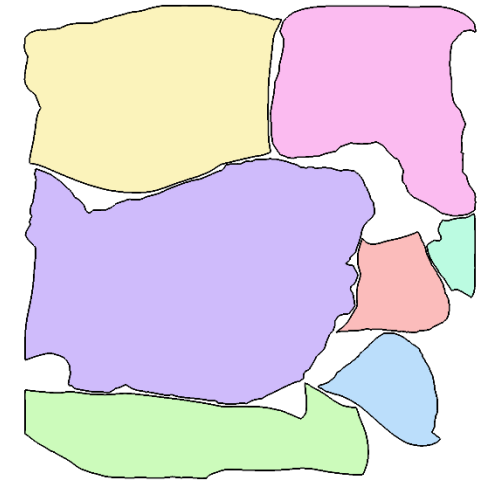
Distortion Reduction



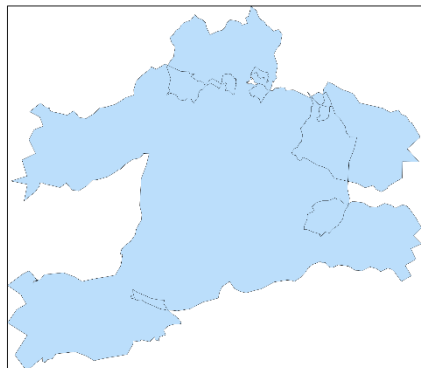
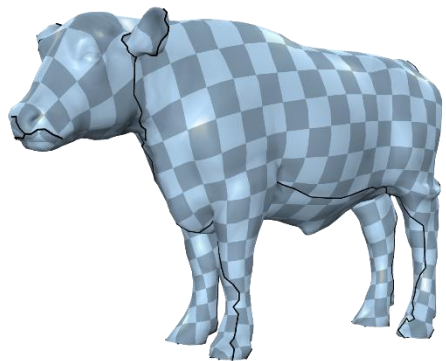
Axis-aligned
High distortion



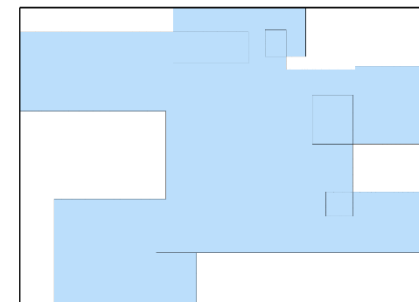
No overlap & High PE
High distortion



No overlap & High PE
Low distortion
Bounded PE

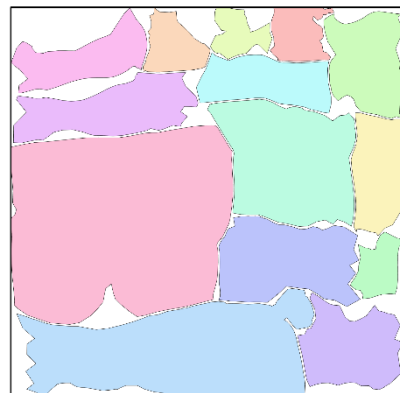
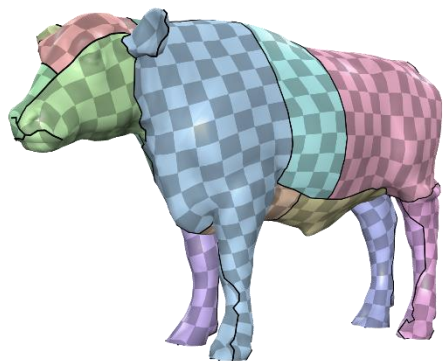


Axis-aligned deformation

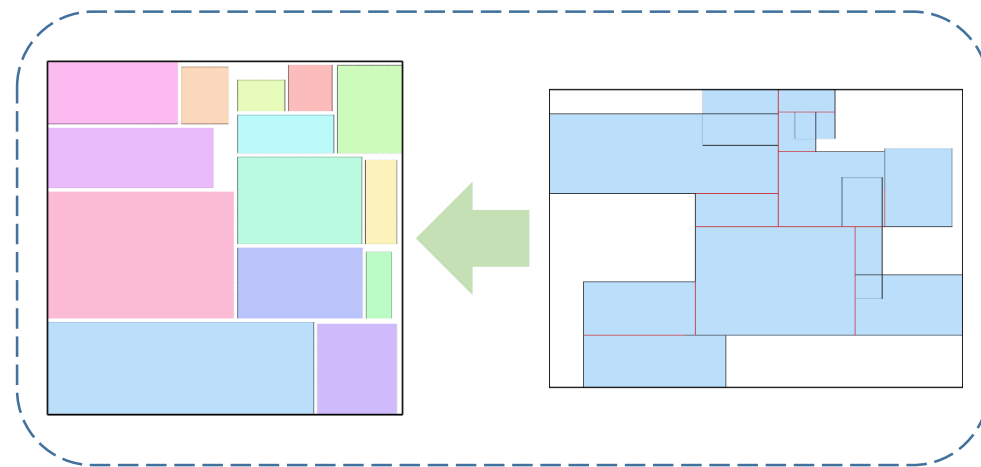


Pipeline

Rectangle
decomposition
and packing

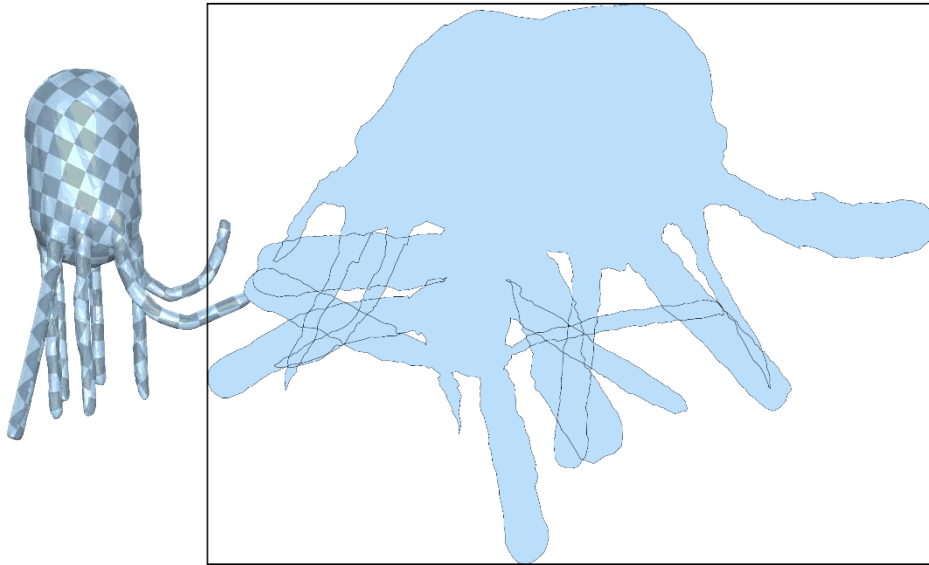


Distortion reduction

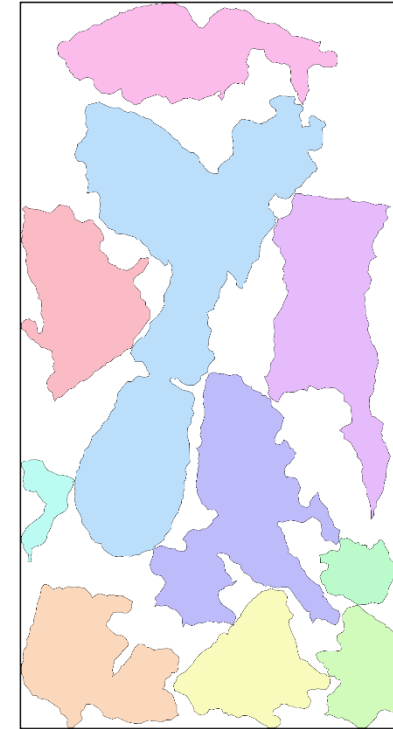
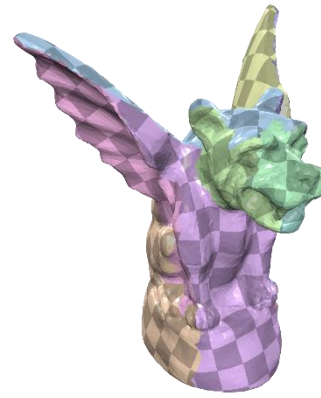


Axis-Aligned Deformation

- Input

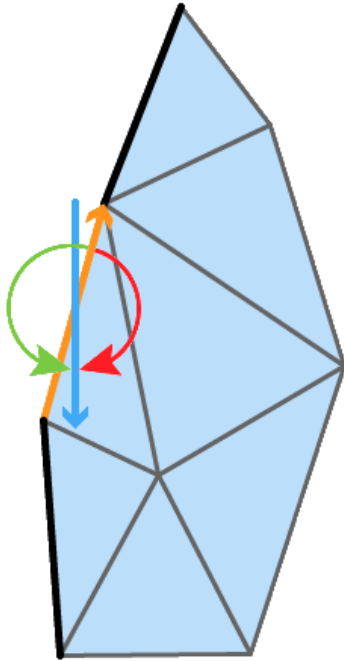


Single chart
With overlap



10 charts
Without overlap

Axis-Aligned Deformation

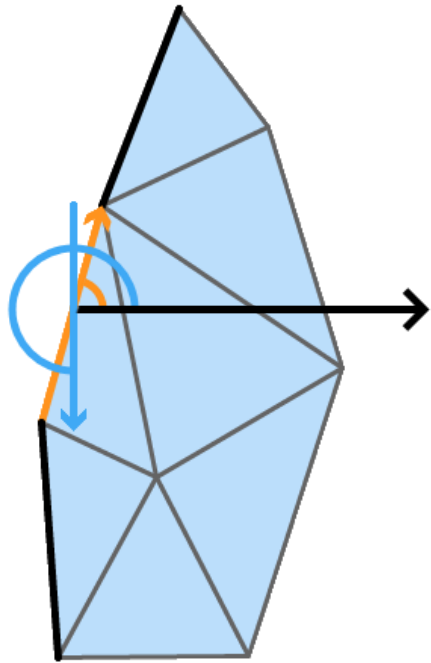


Direction vector
Ambiguous rotating directions

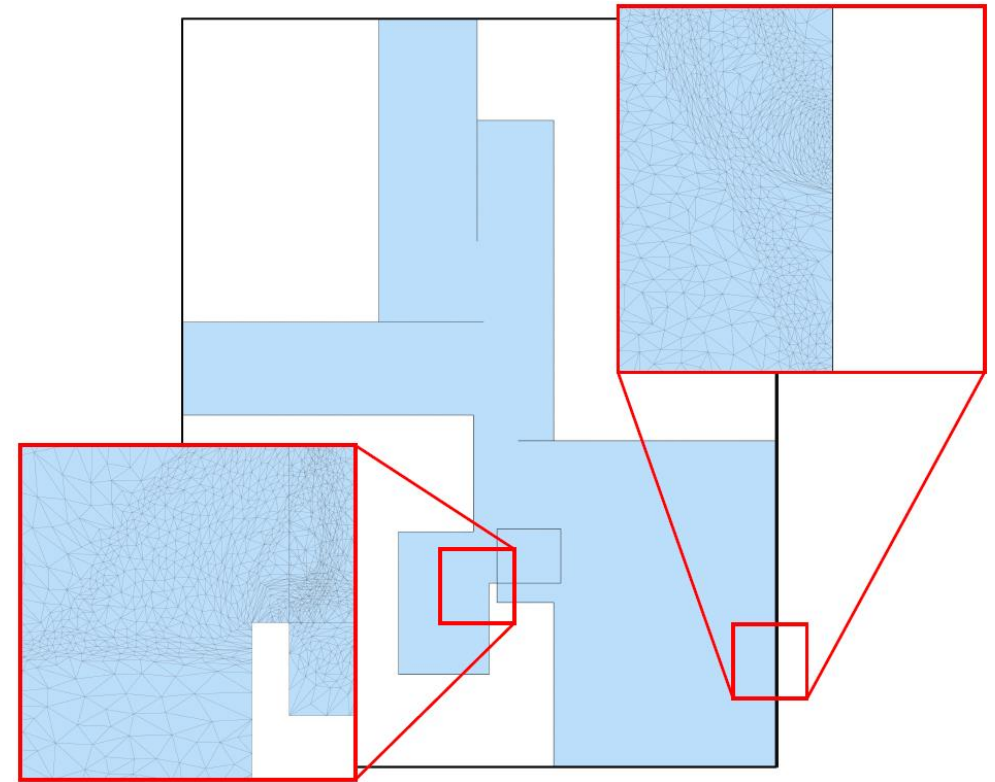


Fail!

Axis-Aligned Deformation

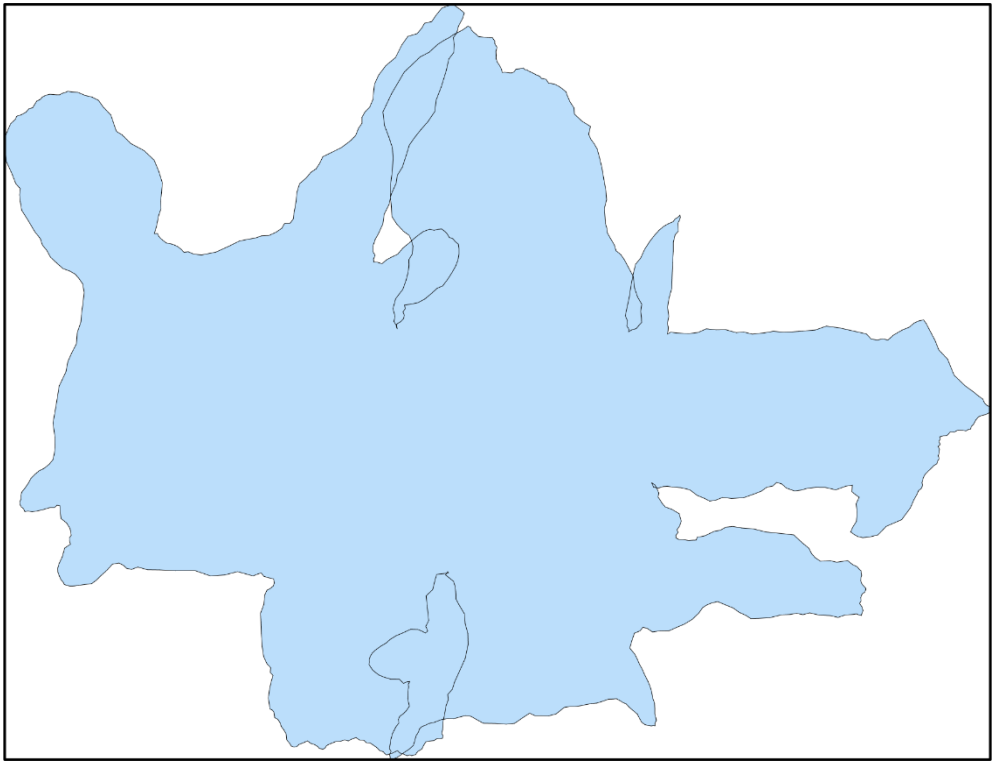


Polar angle
Clear rotating direction

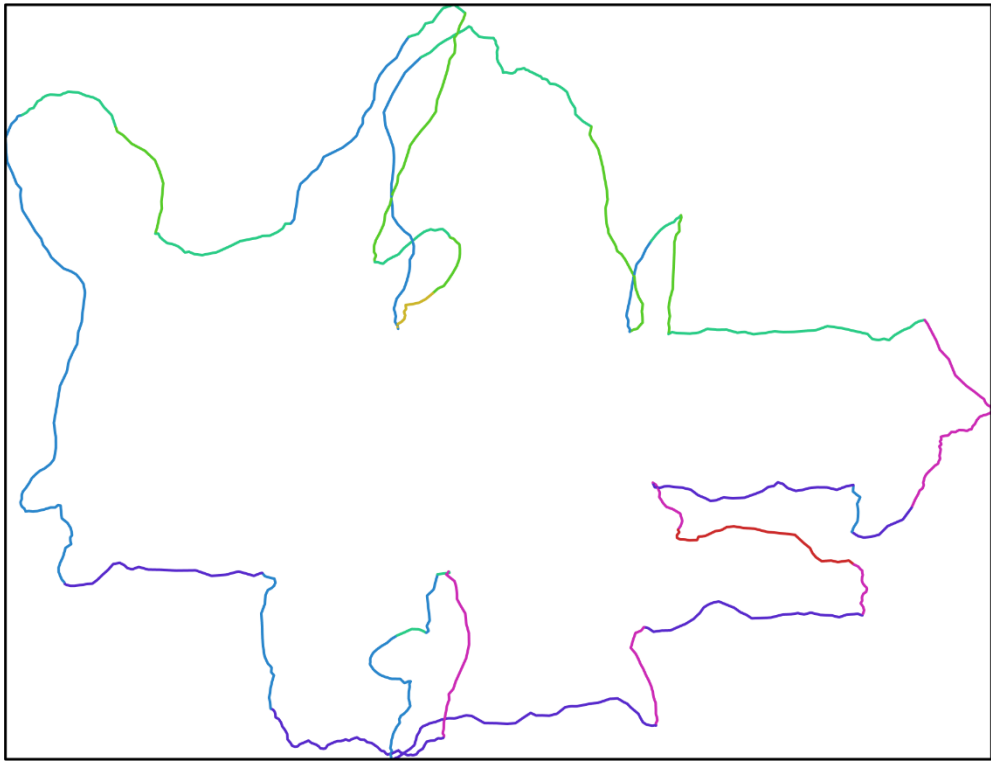


Success!

Axis-Aligned Deformation



Input



Target polar angle

Axis-Aligned Deformation

- Energy of boundary alignment

$$E_{\text{edge}}(\mathbf{b}_i) = \overset{\text{Rotate polar angle}}{\frac{1}{2}(1-\gamma)\left(\theta_i - \frac{\pi}{2}\Theta_i\right)^2} + \overset{\text{Keep length}}{\frac{1}{2}\gamma\left(\frac{l_i}{l_i^0} - 1\right)^2}$$
$$E_{\text{align}}(\mathbf{c}) = \sum_{i=1}^{N_b} \frac{l_i^0}{l^0} E_{\text{edge}}(\mathbf{b}_i)$$

Axis-Aligned Deformation

- Energy of isometric distortion (symmetric Dirichlet)

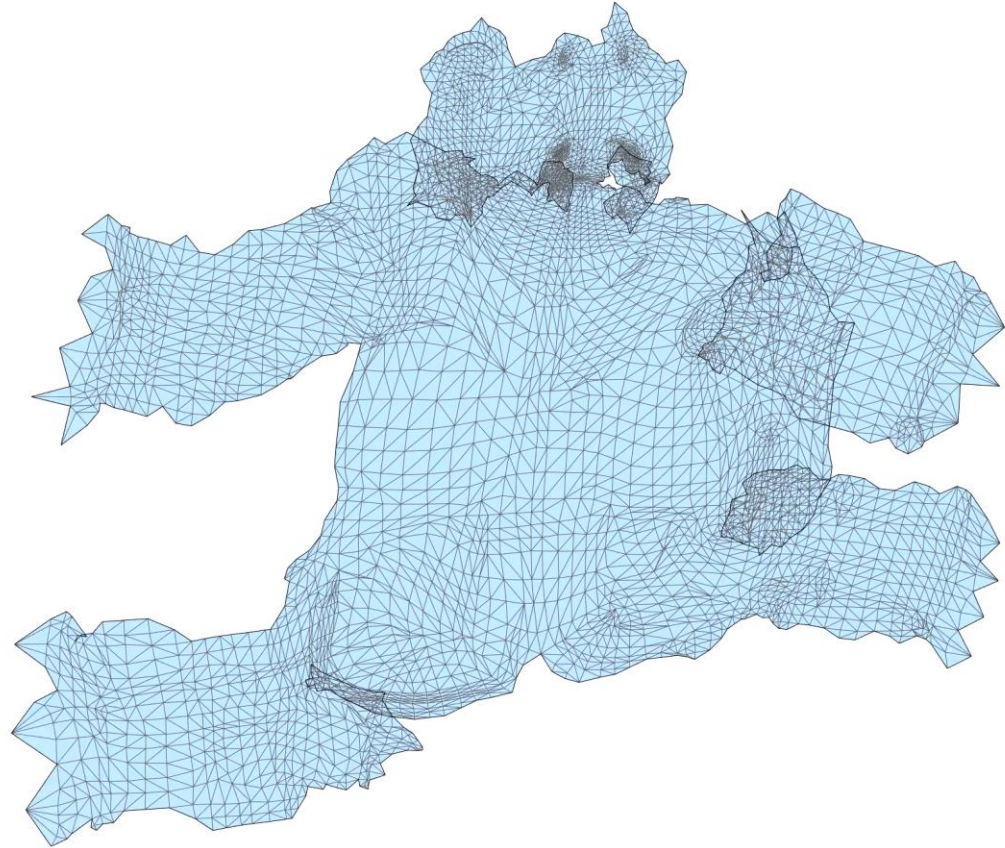
$$E_d(c) = \frac{1}{4} \sum_{f_i \in FC} \frac{\text{Area}(f_i)}{\text{Area}(M^c)} (\|J_i\|_F^2 + \|J_i^{-1}\|_F^2)$$

Keep low distortion and orientation consistency.

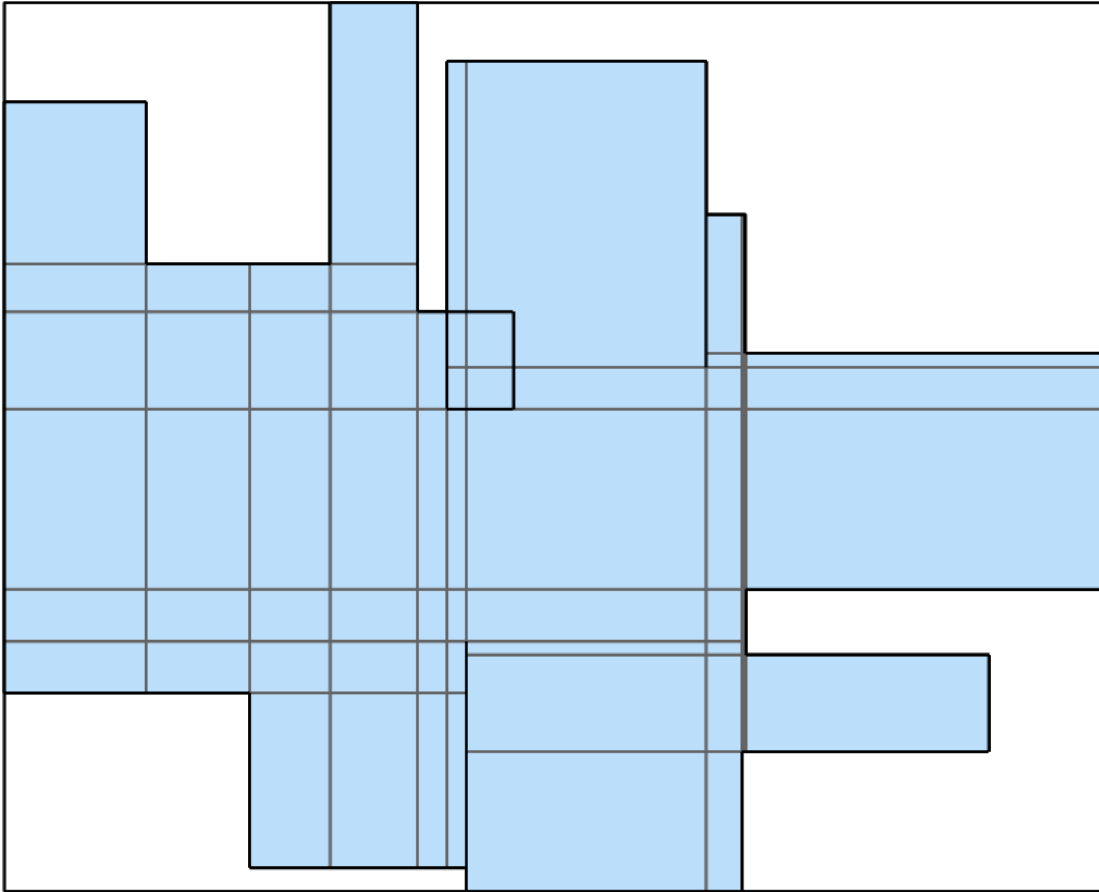
Axis-Aligned Deformation

0.2X Playback

$$\begin{aligned} \min_{\mathbf{c}} \quad & E_d(\mathbf{c}) + \lambda E_{\text{align}}(\mathbf{c}) \\ \text{s.t.} \quad & \det J_i > 0, \forall i \end{aligned}$$



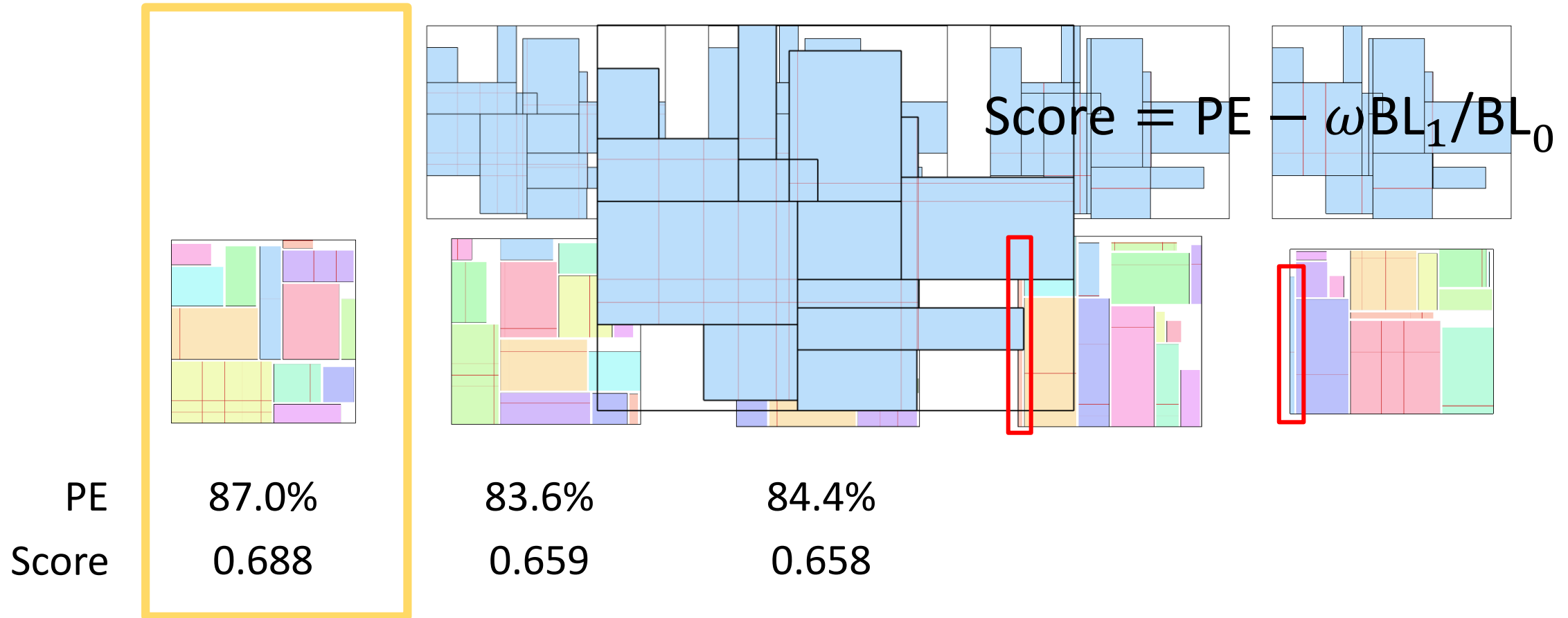
Rectangle Decomposition and Packing



The faces are all rectangles.
But the number is too many.

Rectangle Decomposition and Packing

- Motorcycle graph algorithm



Distortion Reduction

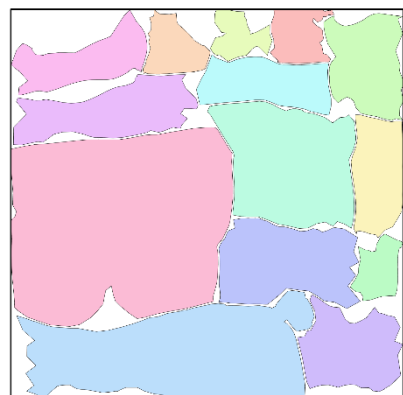
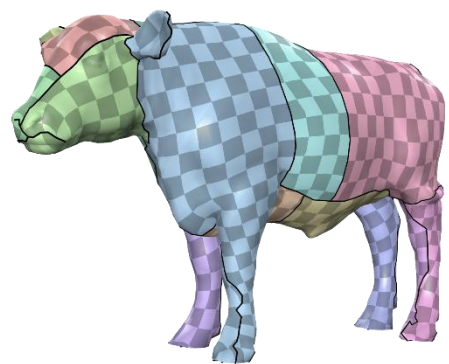
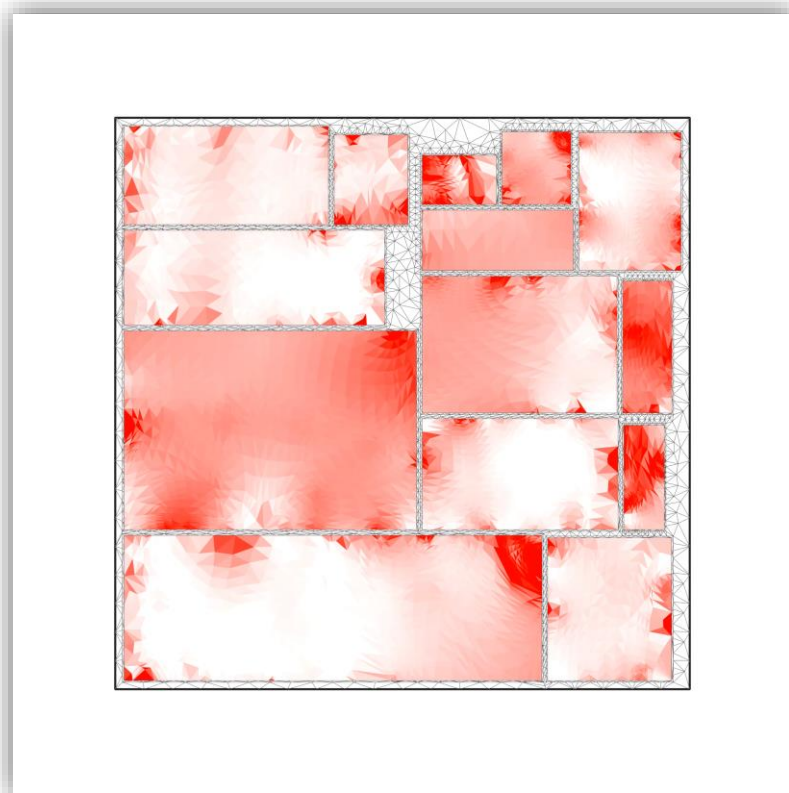
$$\min_{\mathcal{C}} E_{\text{reduction}} = E_{\text{d}}(\mathcal{C}) + E_{\text{PE}}(\mathcal{C})$$

s.t. Φ is **bijjective**

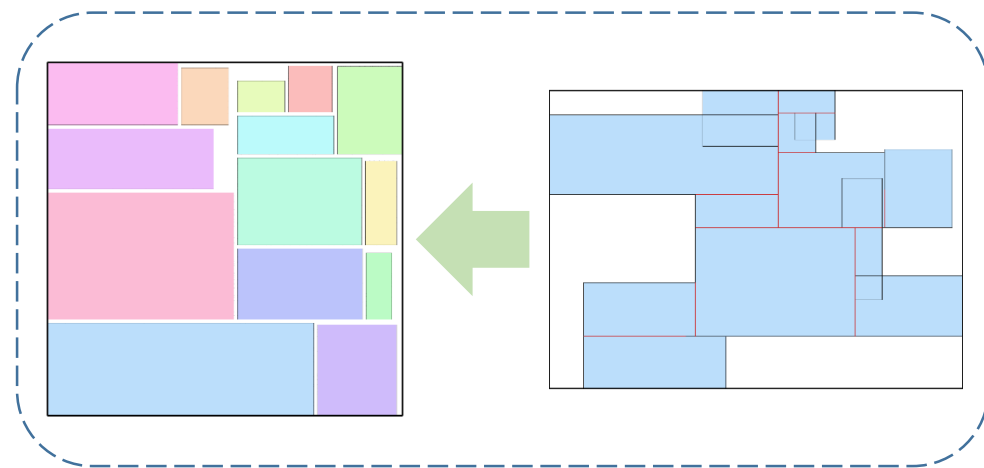
Isometric energy

Barrier function of PE bound

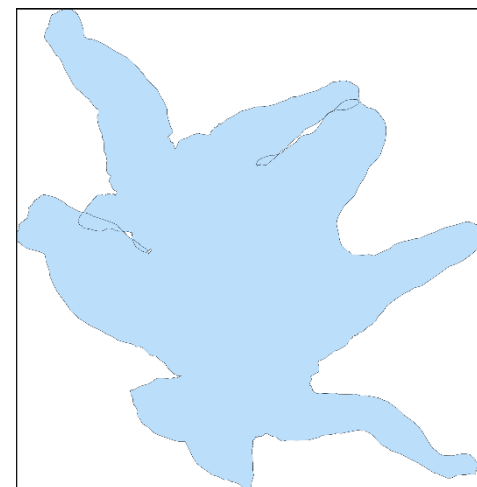
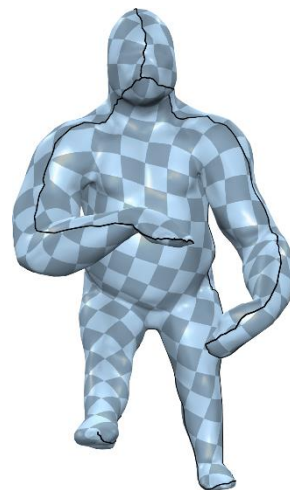
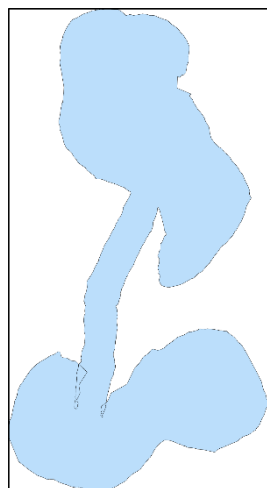
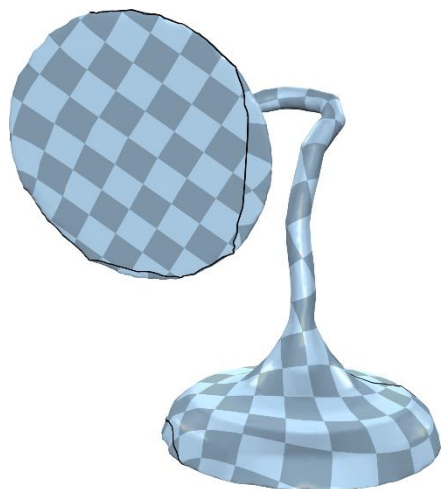
Scaffold-based method
[Jiang et al. 2017]



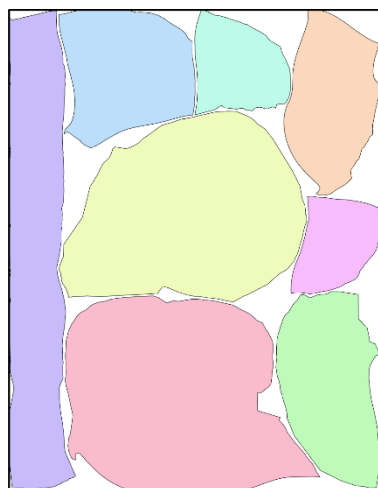
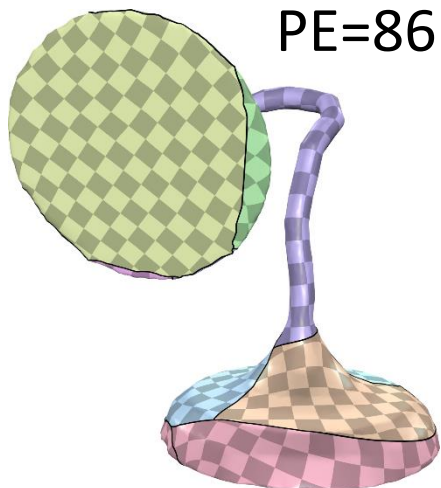
Distortion reduction



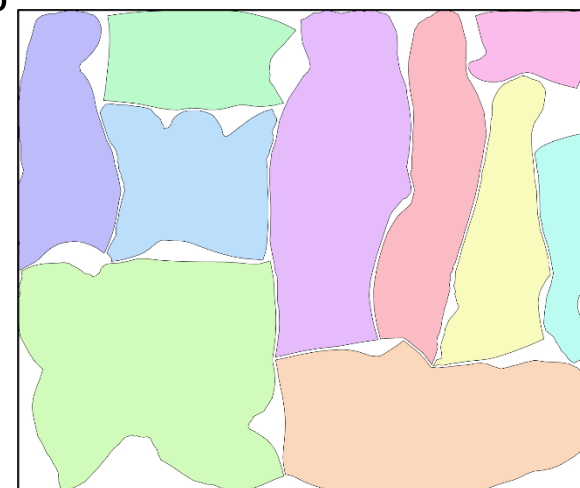
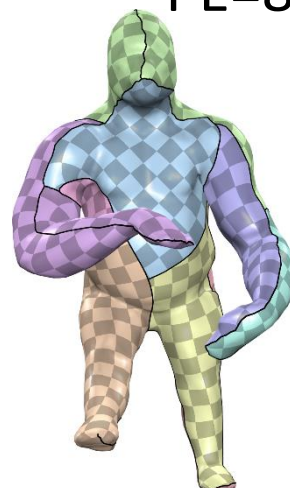
Benchmark (5,588)



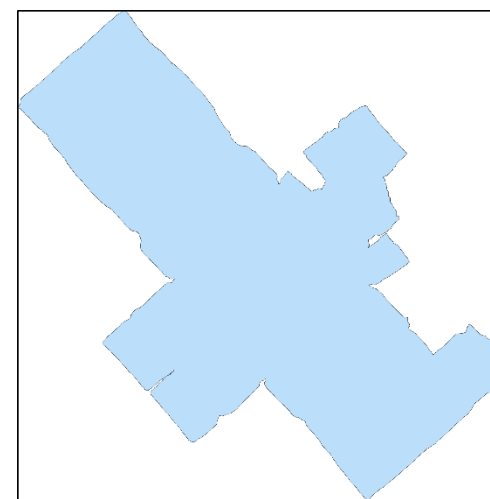
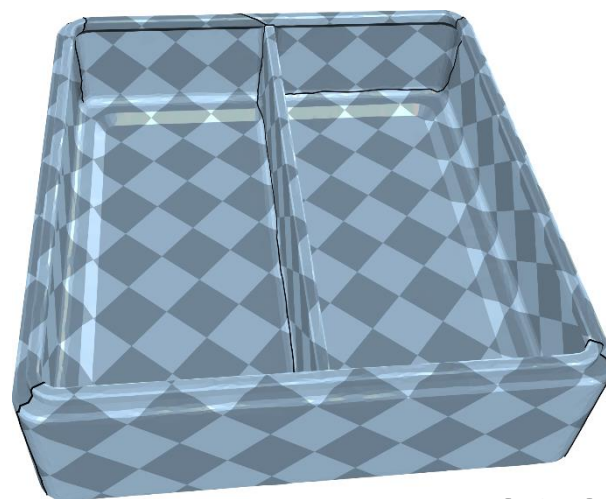
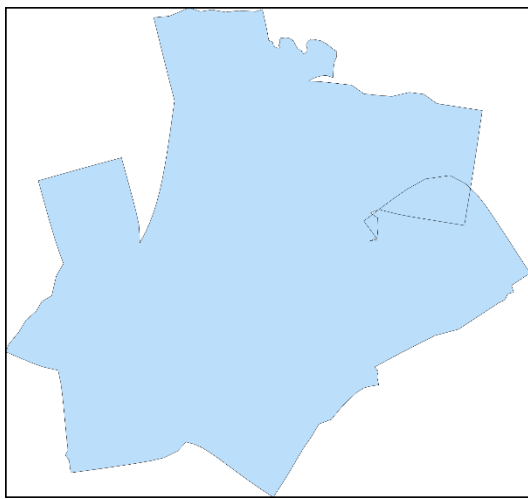
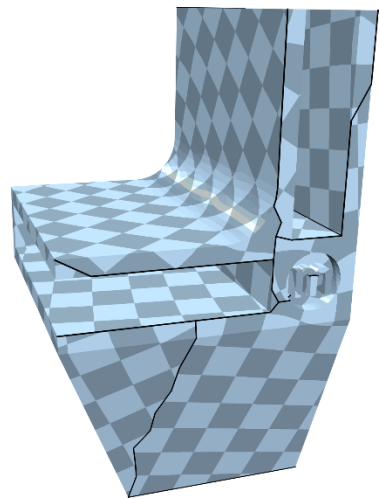
PE=86.2%



PE=86.7%

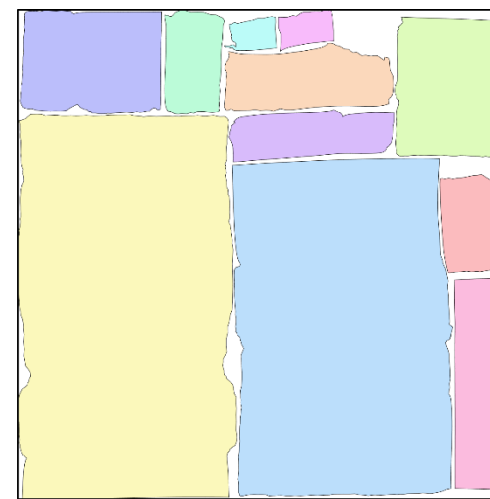
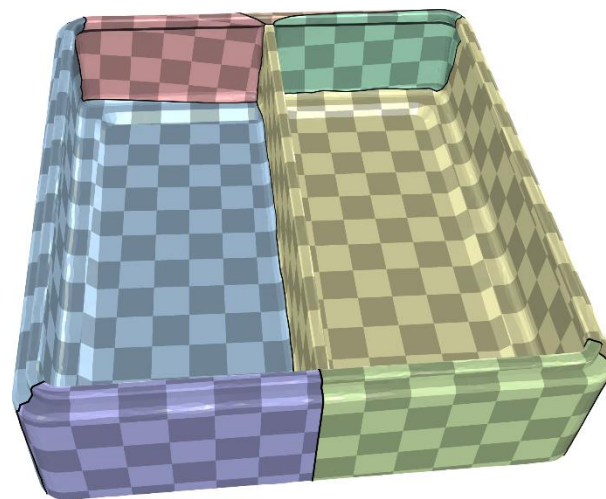
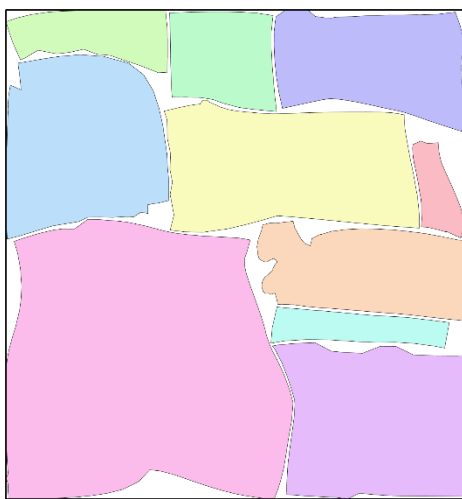
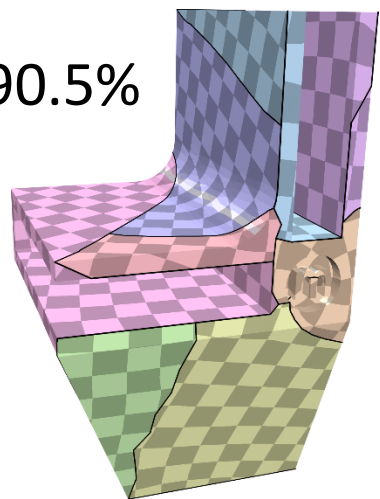


Benchmark (5,588)



PE=91.0%

PE=90.5%



PolyAtlas: Atlas Refinement with Bounded Packing Efficiency

Submitted to ACM SIGGRAPH 2019

ID: 339

PolyCube

Xiao-Ming Fu

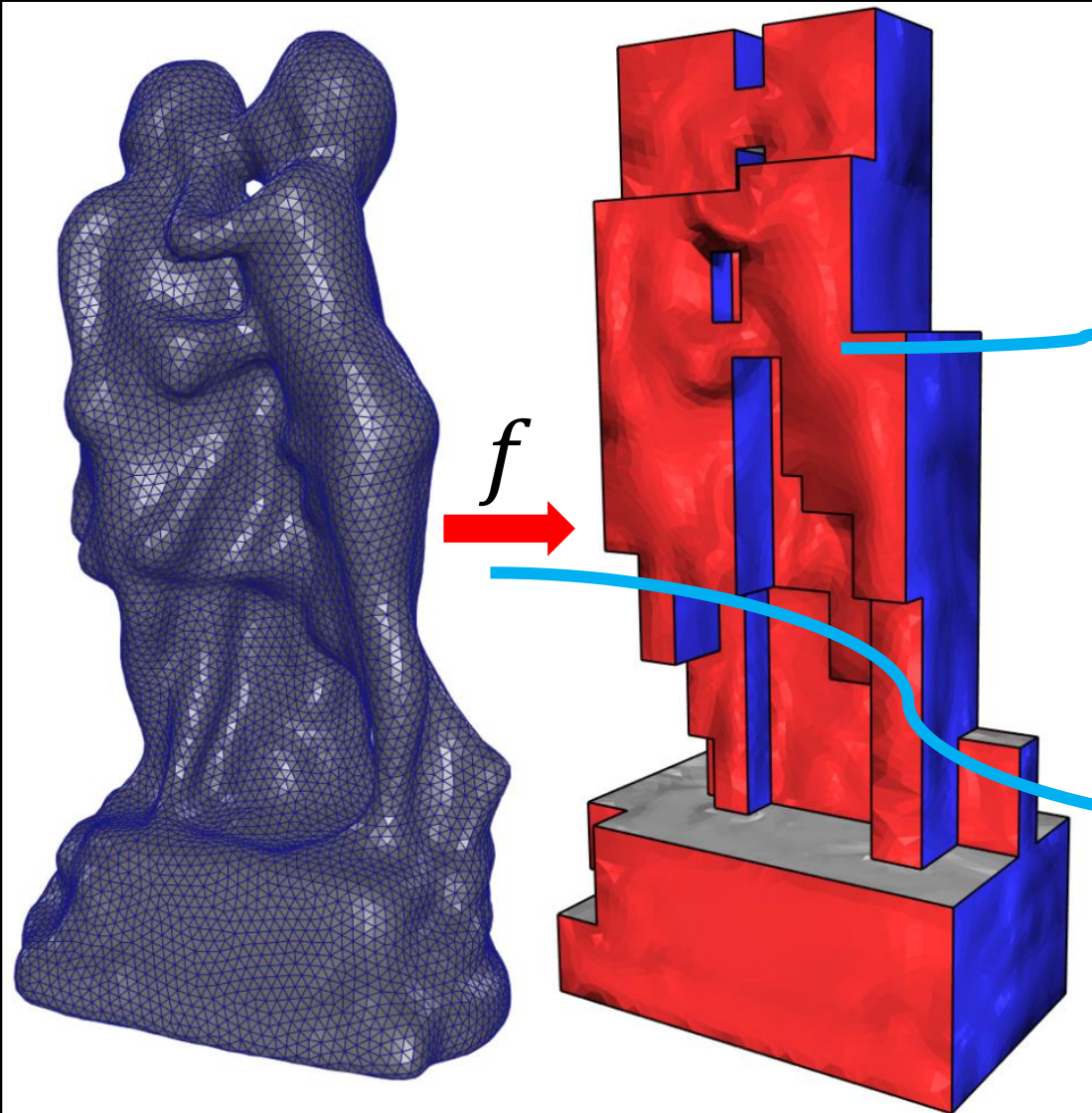
Outlines

- Definition
- Deformation-based method
 - All-Hex Mesh Generation via Volumetric PolyCube Deformation
- Voxel-based method
 - Optimizing PolyCube domain construction for hexahedral remeshing
- Cluster-based method
 - PolyCut: Monotone Graph-Cuts for PolyCube Base-Complex Construction
- Generalized PolyCube

Outlines

- **Definition**
- Deformation-based method
 - All-Hex Mesh Generation via Volumetric PolyCube Deformation
- Voxel-based method
 - Optimizing PolyCube domain construction for hexahedral remeshing
- Cluster-based method
 - PolyCut: Monotone Graph-Cuts for PolyCube Base-Complex Construction
- Generalized PolyCube

PolyCube



Tetrahedral Mesh

PolyCube

PolyCube:

1. Compact representations for closed complex shapes
2. Boundary normal aligns to the axes.
3. Axes: $(\pm 1, 0, 0)^T$, $(0, \pm 1, 0)^T$, $(0, 0, \pm 1)^T$.

PolyCube-Map f :

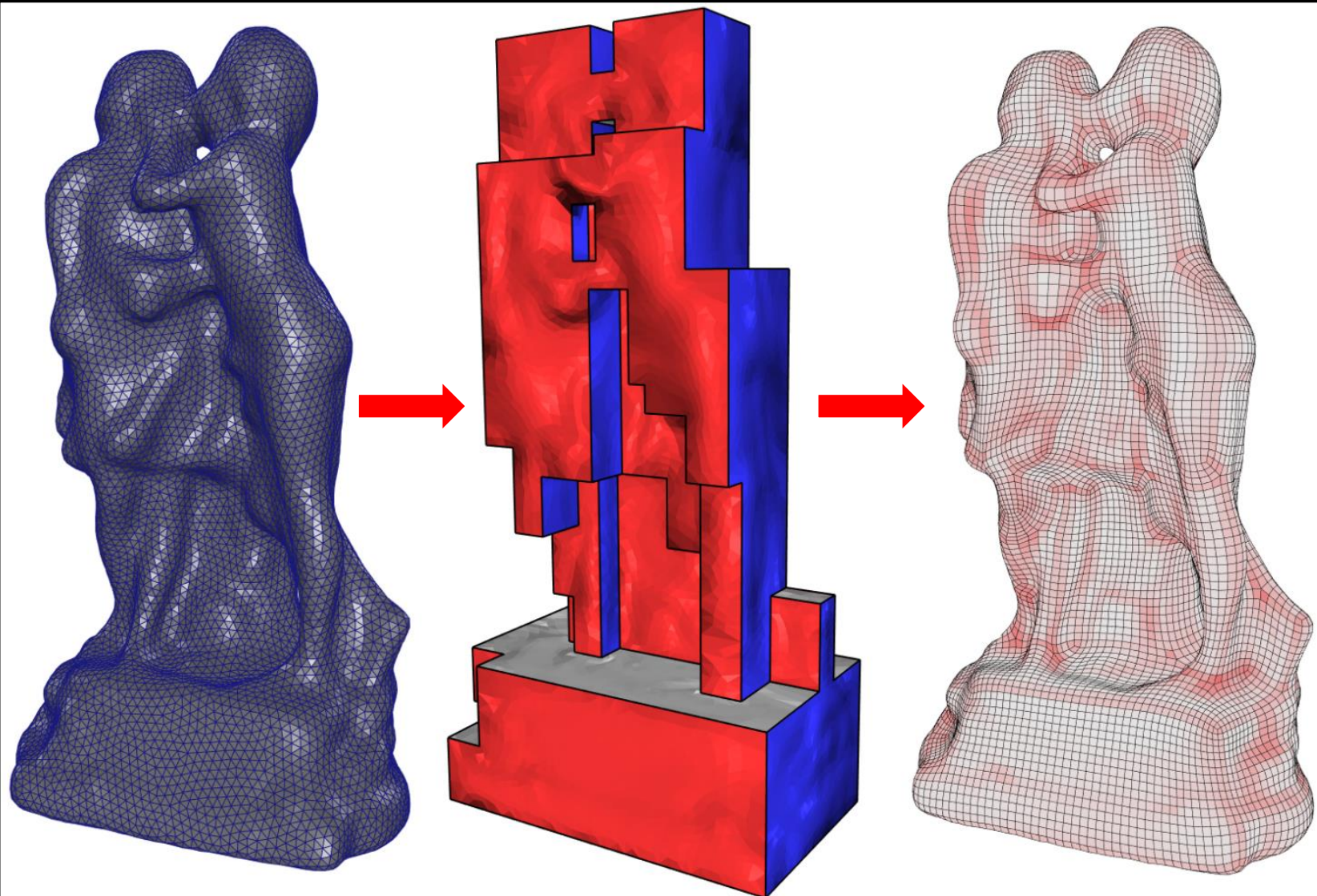
1. A mesh-based map.
2. Foldover-free and low distortion.

Application – All-hex meshing

Tetrahedral Mesh

PolyCube

All-Hex Mesh



Applications based on PolyCube:

1. All-Hex Mesh generation.
2. Texture Mapping [Tarini et al. 2004].
3. GPU-based subdivision [Xia et al. 2011].

.....

Application – Seamless texture mapping

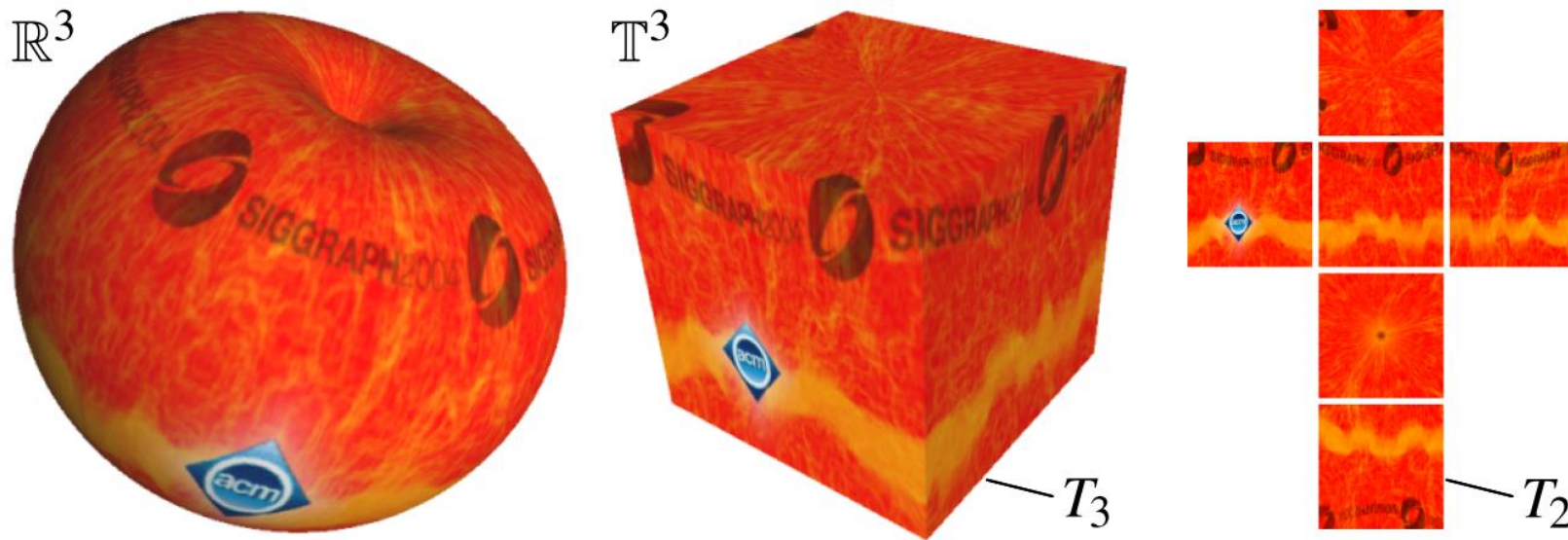


Figure 1: Cube maps can be used to seamlessly texture map an apple (left). In this case, the 3D texture domain T_3 is the surface of a single cube that is immersed in the 3D texture space \mathbb{T}^3 (middle) and corresponds to a 2D texture domain T_2 that consists of six square images (right).

Applications based on PolyCube:
1. All-Hex Mesh generation.
2. Texture Mapping [Tarini et al. 2004].
3. GPU-based subdivision [Xia et al. 2011].

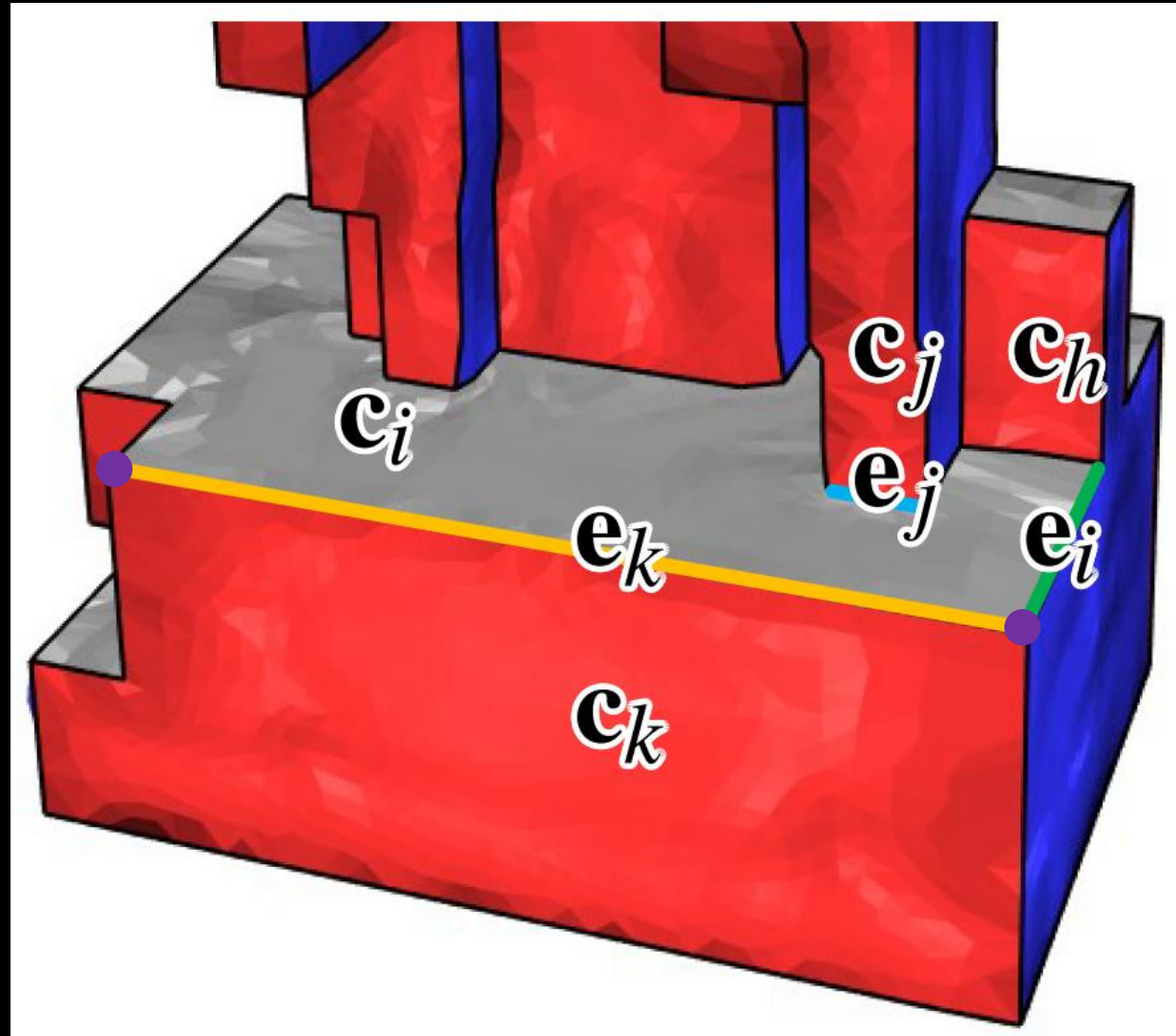
.....

PolyCube facet, edge, and vertex

PolyCube **facet**:
share the same label

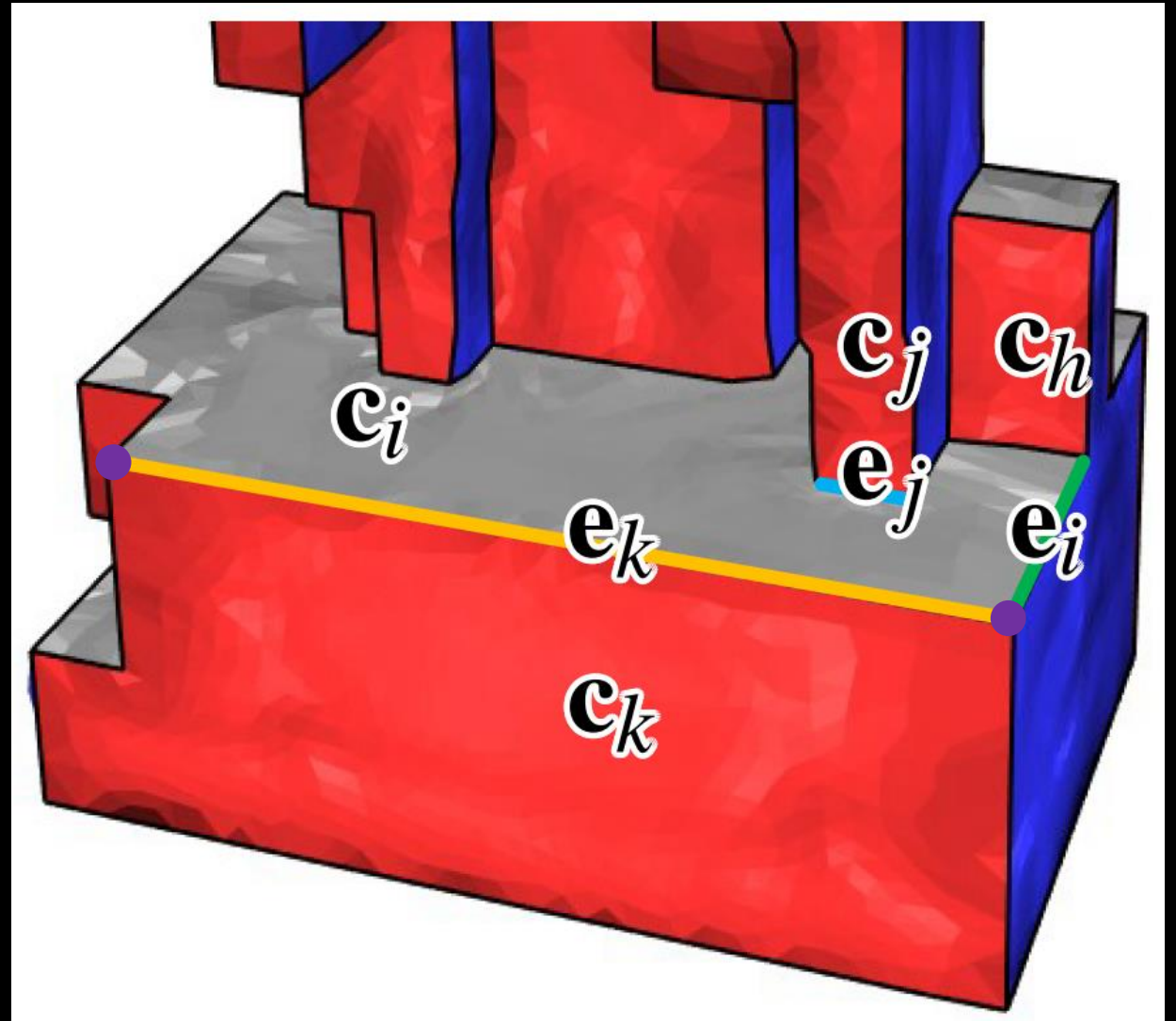
PolyCube **edge**:
The edges between facets

PolyCube **vertex**:
sharing by at least three charts



Sufficient topological conditions

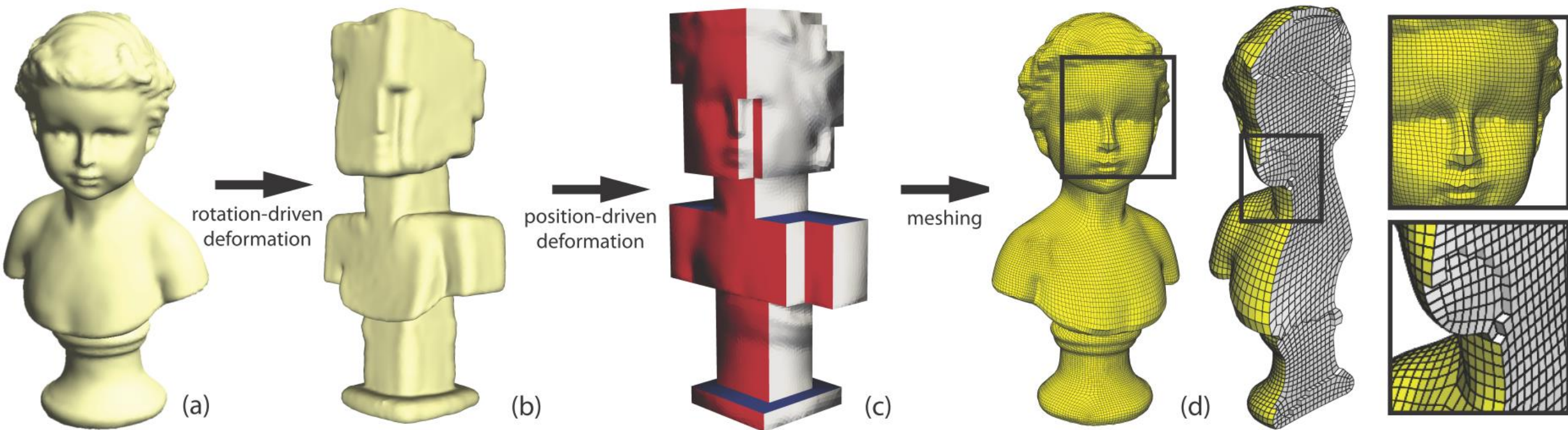
- Any PolyCube facet should have **at least four** neighboring Poly-Cube facets.
- Any two neighboring PolyCube facets should not have **opposite** labels such as $+X$ and $-X$.
- The valence of each PolyCube vertex is **three**.



Outlines

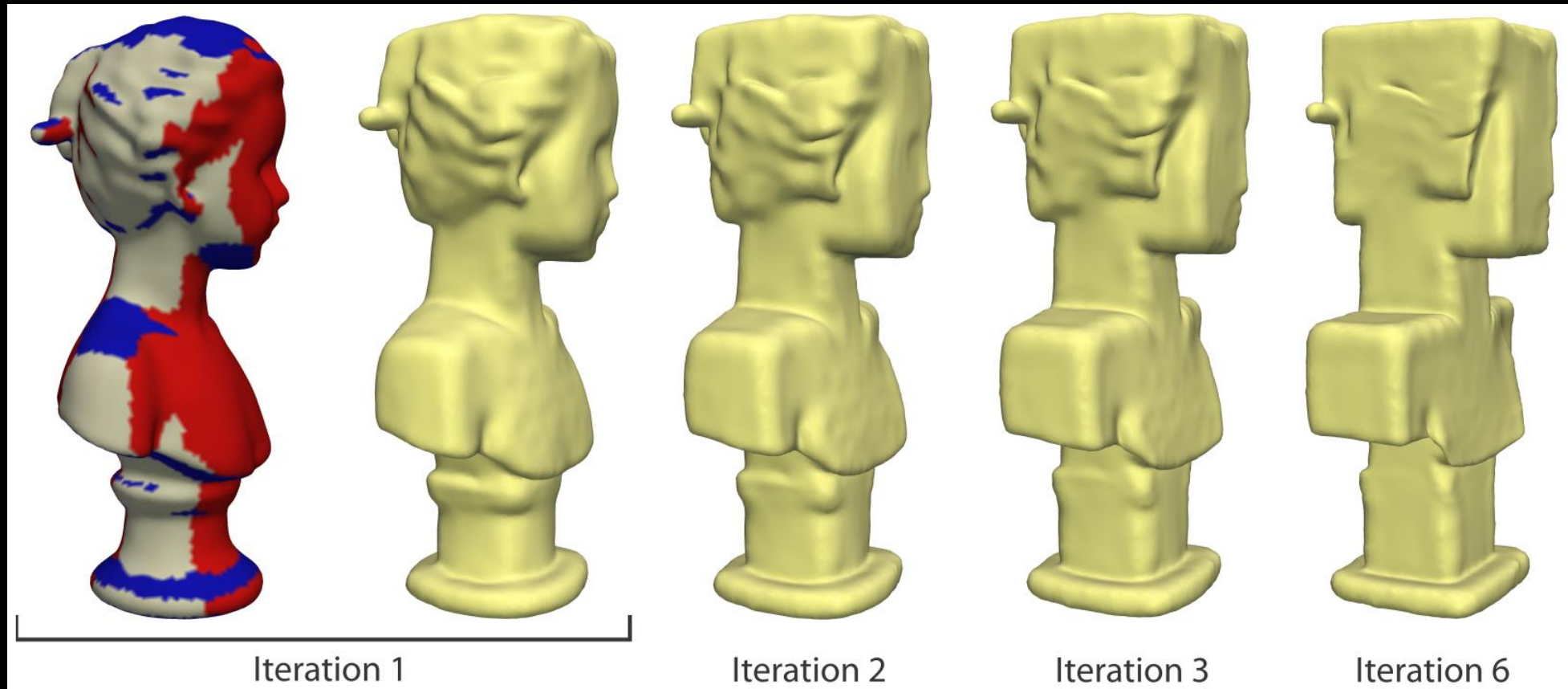
- Definition
- Deformation-based method
 - All-Hex Mesh Generation via Volumetric PolyCube Deformation
- Voxel-based method
 - Optimizing PolyCube domain construction for hexahedral remeshing
- Cluster-based method
 - PolyCut: Monotone Graph-Cuts for PolyCube Base-Complex Construction
- Generalized PolyCube

Pipeline



Rotation-driven deformation

- Goal: gradually **aligns** the model's surface normals with one of the six global axes, **preserving shape** as much as possible.



Rotation-driven deformation

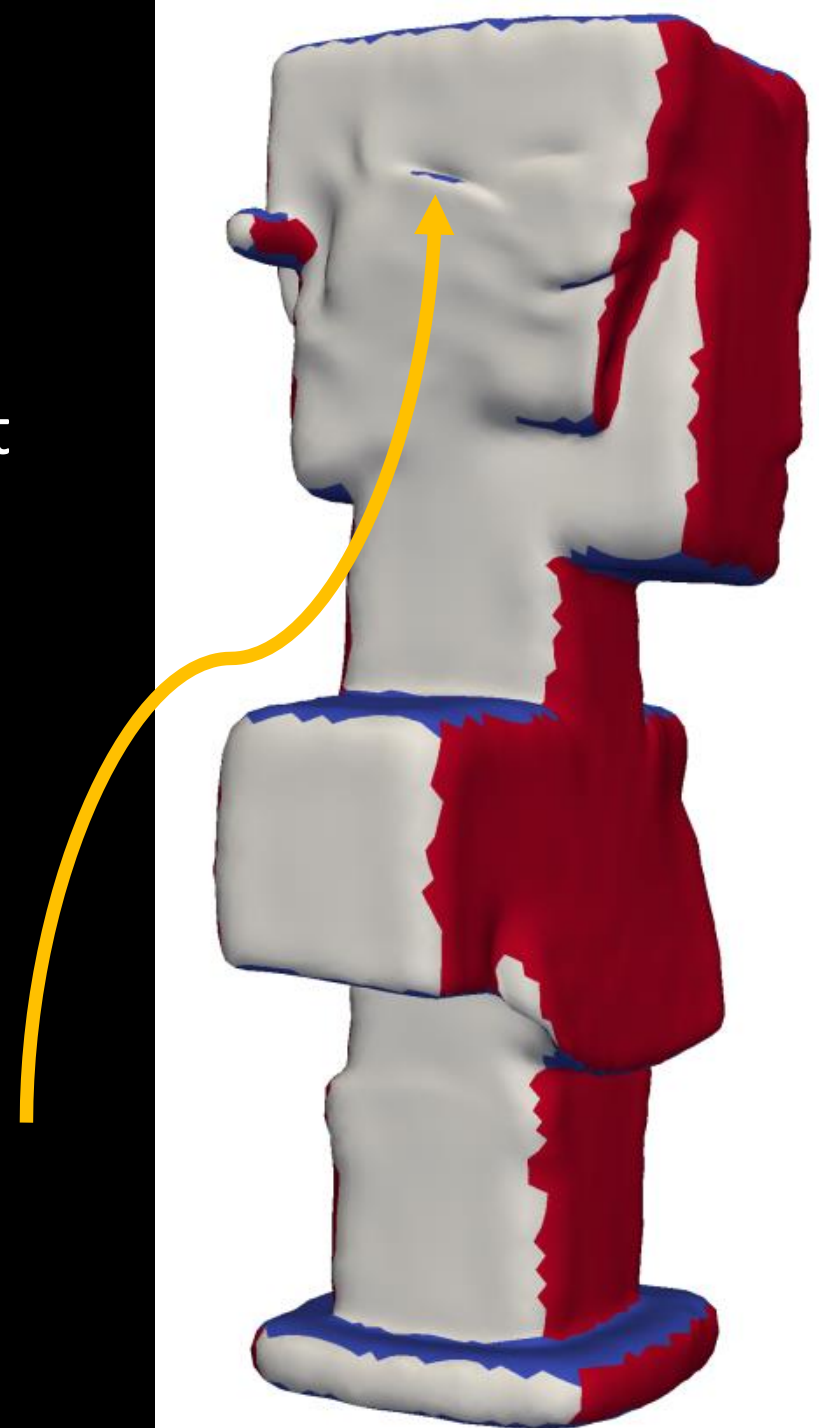
- As-Rigid-As-Possible deformation $E = \sum_{i=1}^{N_v} w_i \sum_{j \in \Omega(i)} w_{ij} \|(p'_i - p'_j) - R_i(p_i - p_j)\|^2$
 - No local step
 - Rotations are determined by axis-alignment constraints.
- Steps:
 - For every surface vertex (except those on sharp features), the minimal rotation necessary to align each surface vertex normal with one of $\pm X, \pm Y, \pm Z$.
 - quaternion
 - Smoothly propagate to feature and interior vertices.
 - Laplace equation per quaternion component
 - Solve E .
 - Least squares.

Labeling

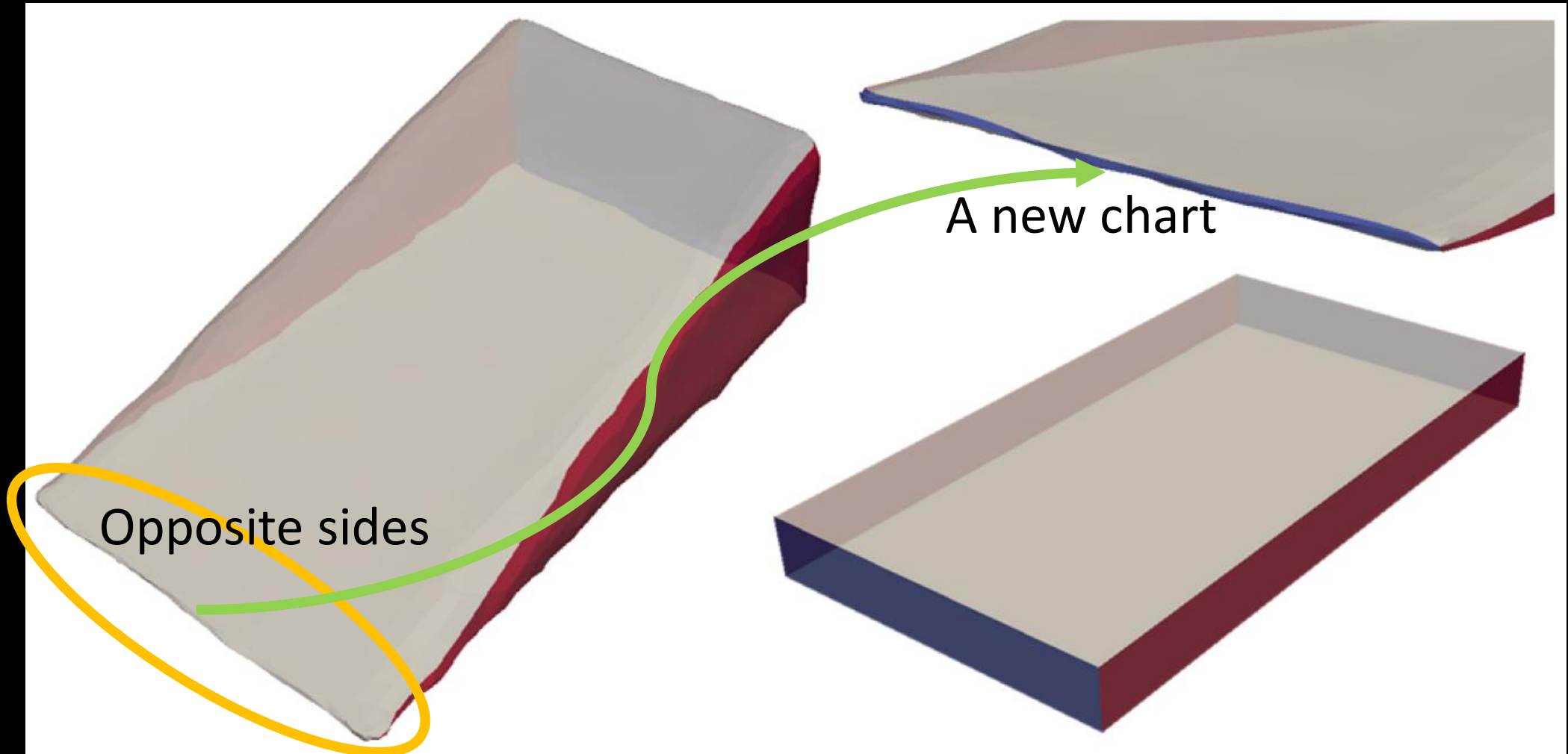
- 1. Label surface triangles according to the closest axis
- 2. Group similarly labeled triangles into charts.
- 3. Straighten chart boundaries.



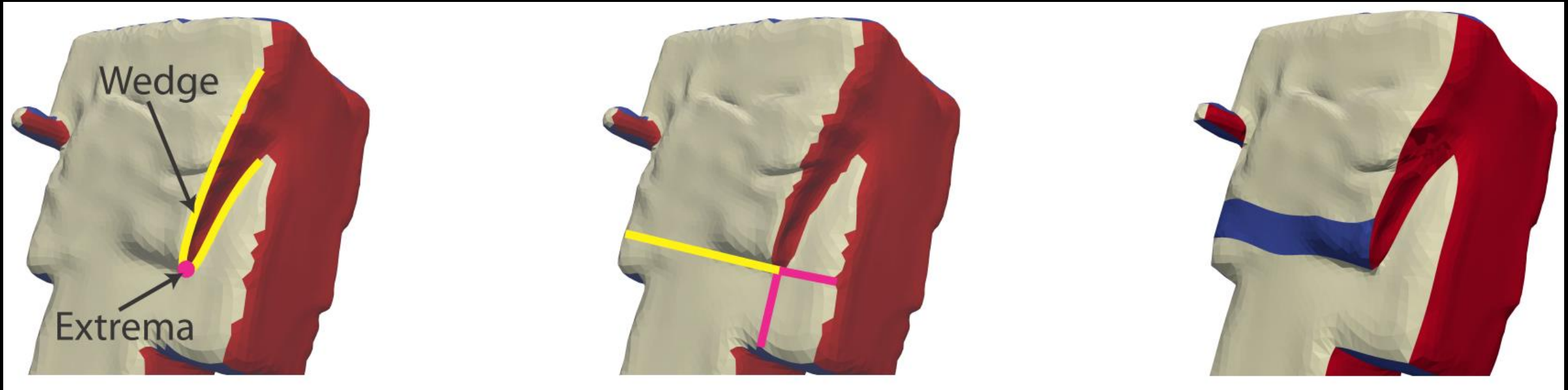
- 4. remove small, spurious charts bounded by at most two edges



Multi-orientation chart



Highly non-planar chart



Detect extrema along the chart boundary

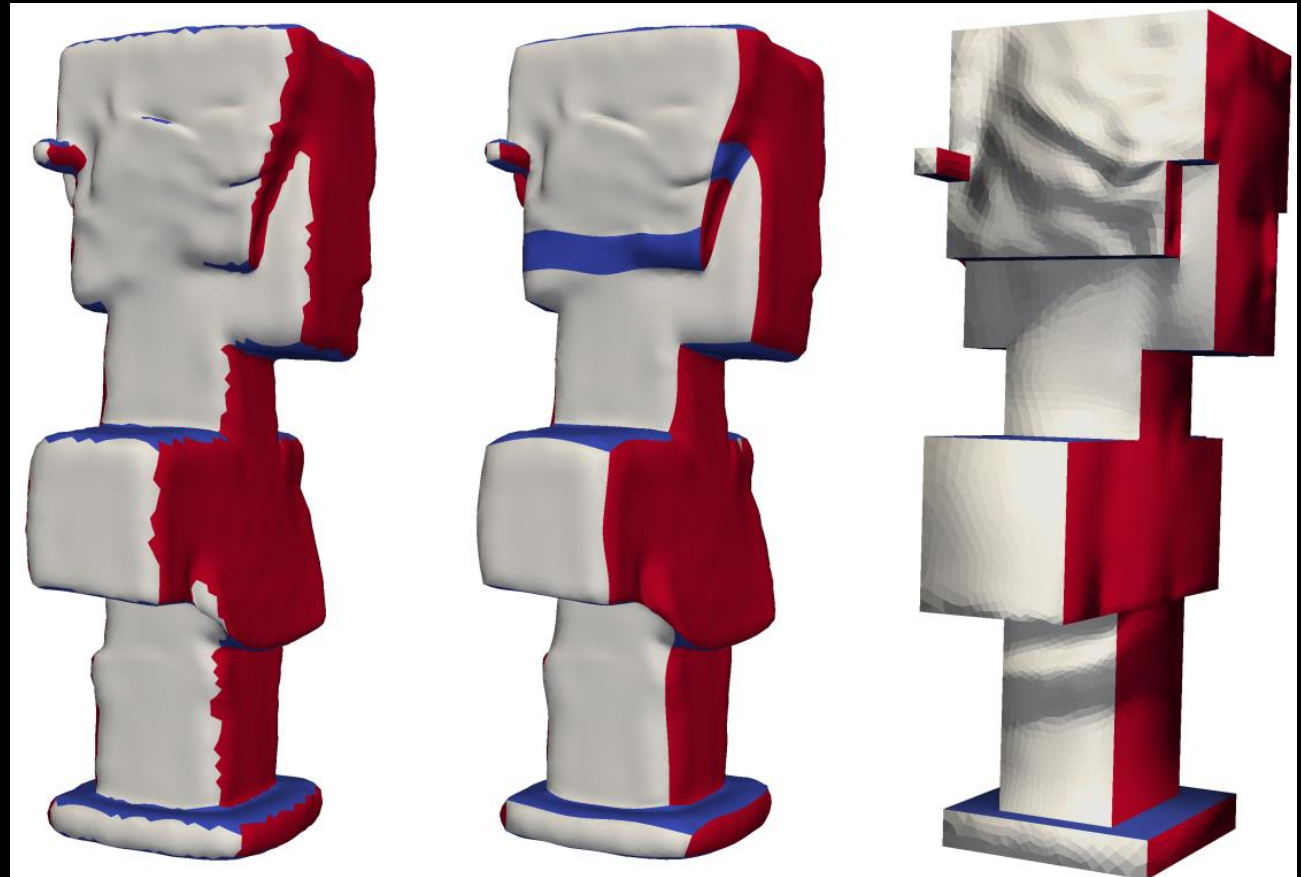
Three possible axis-aligned cut options.

Valid cuts are defined as those that would not introduce new charts with **three or fewer** neighbors.

Position-driven deformation

- constrain each chart to an axis-aligned plane.
 - the chart coordinate

$$E = \sum_{i=1}^{N_v} w_i \sum_{j \in \Omega(i)} w_{ij} \|(p'_i - p'_j) - R_i(p_i - p_j)\|^2$$



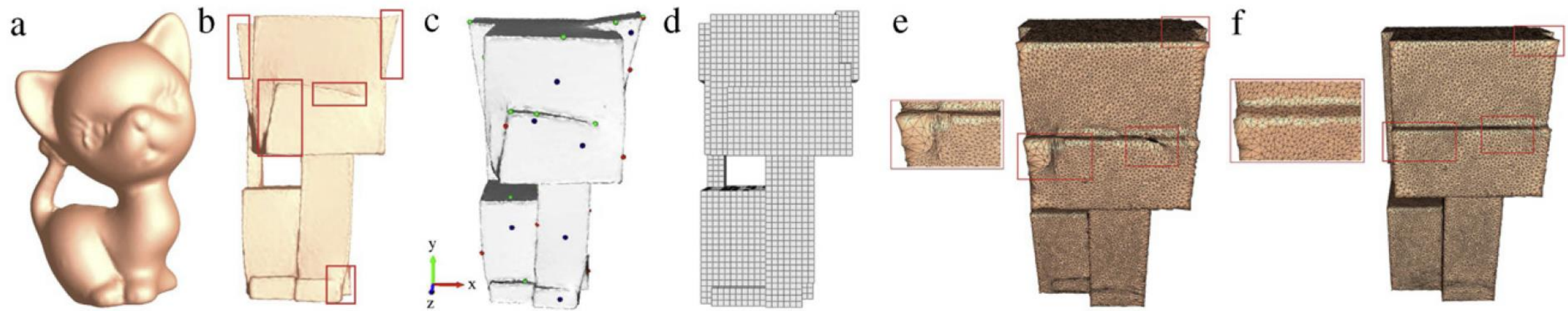
More papers:

- L_1 -based Construction of Polycube Maps from Complex Shapes (2014)
- Efficient Volumetric PolyCube-Map Construction (2016)

Outlines

- Definition
- Deformation-based method
 - All-Hex Mesh Generation via Volumetric PolyCube Deformation
- **Voxel-based method**
 - **Optimizing PolyCube domain construction for hexahedral remeshing**
- Cluster-based method
 - PolyCut: Monotone Graph-Cuts for PolyCube Base-Complex Construction
- Generalized PolyCube

Pipeline

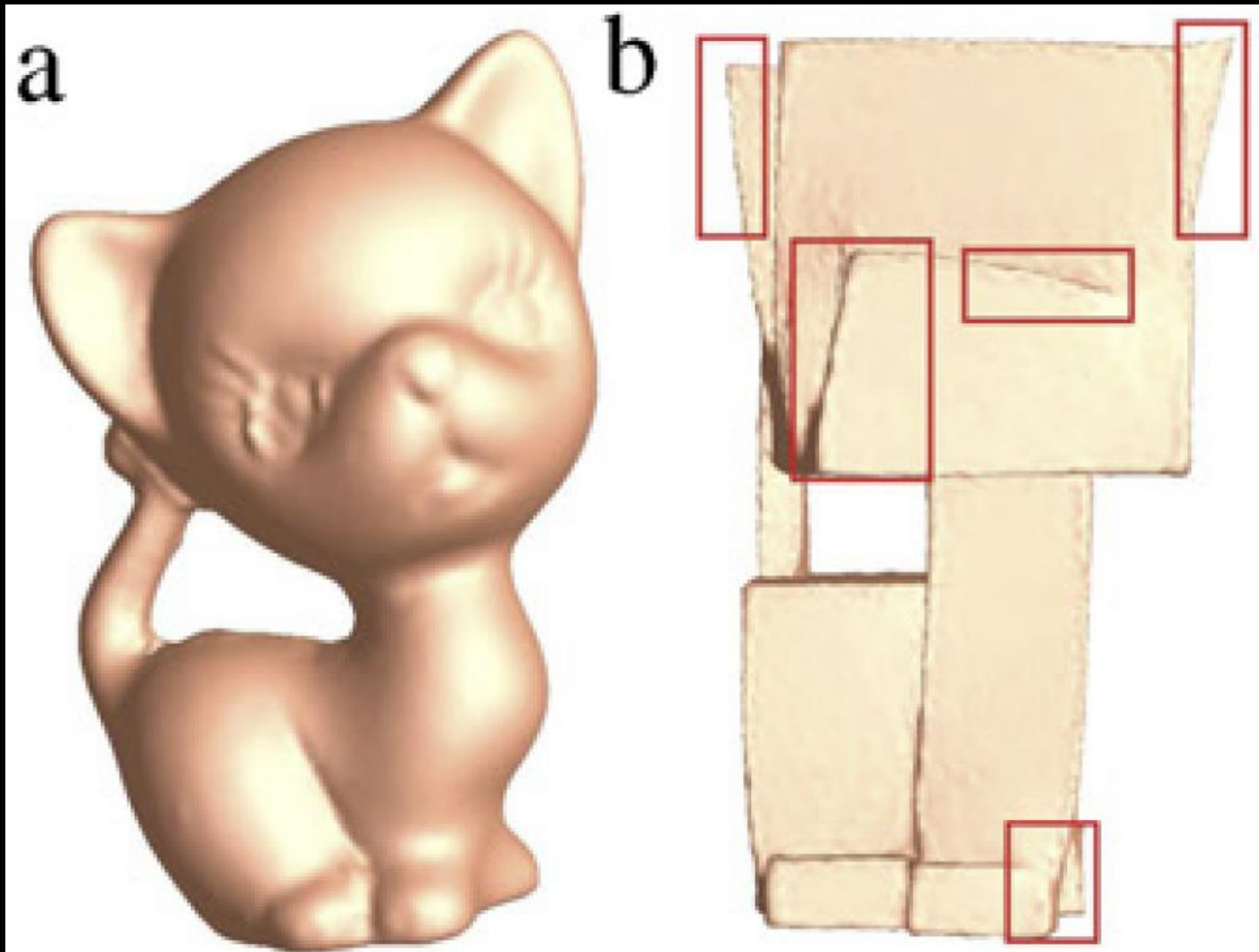


1. Pre-deformation

2. PolyCube construction
and optimization

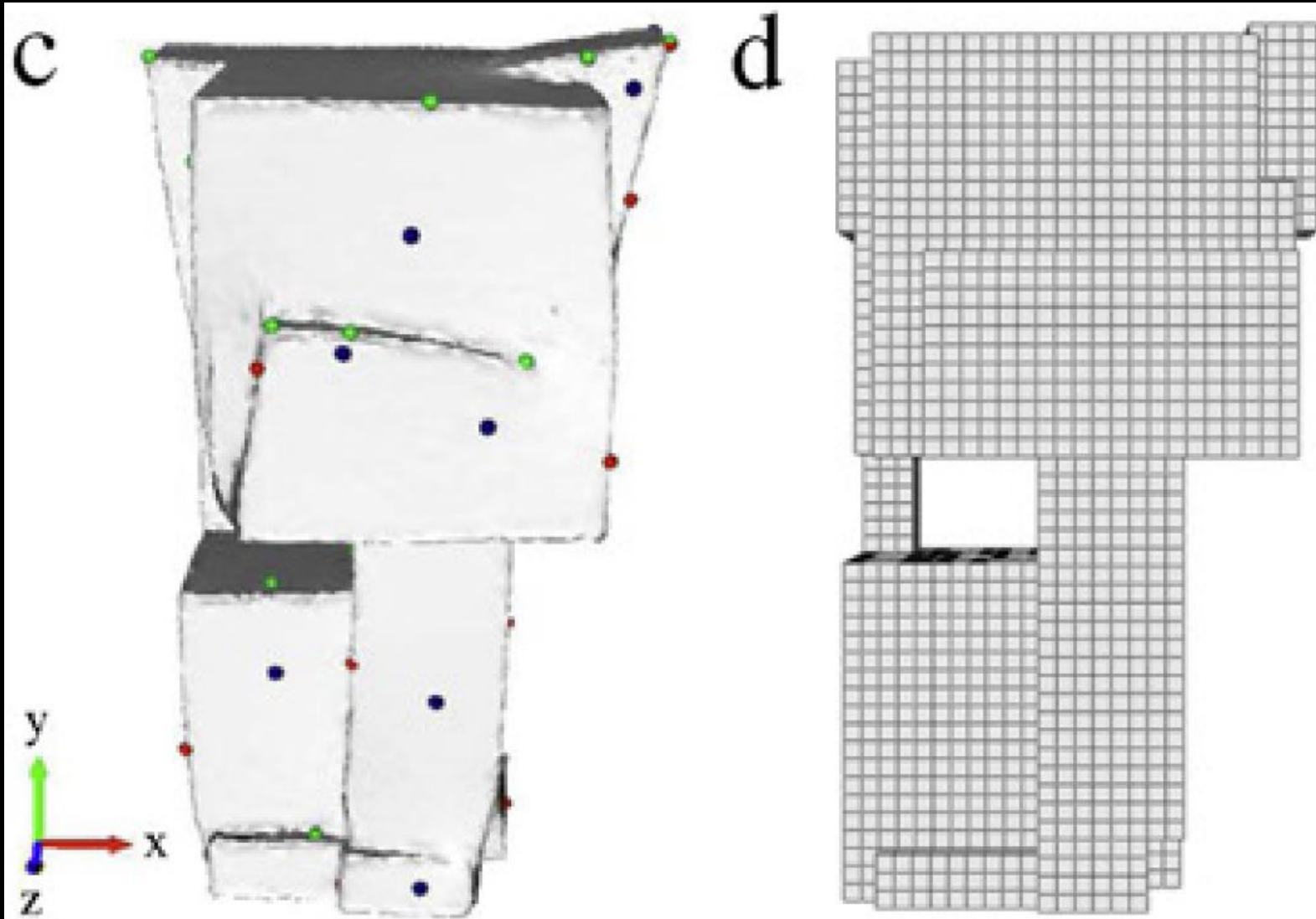
3. Mapping computation

Wedge regions



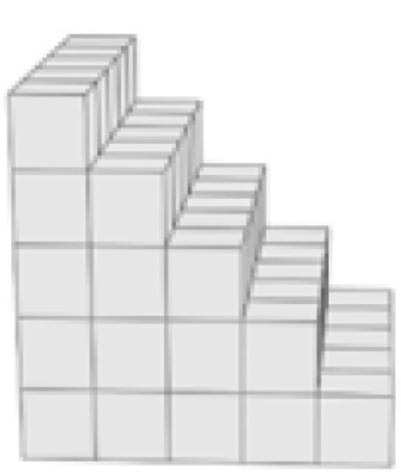
Wedge regions are hard to avoid

Voxelization

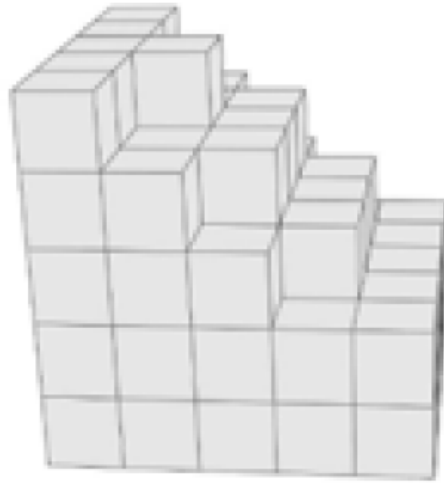


Length of cube

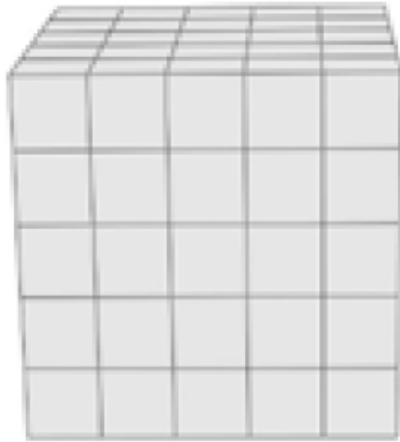
Optimization



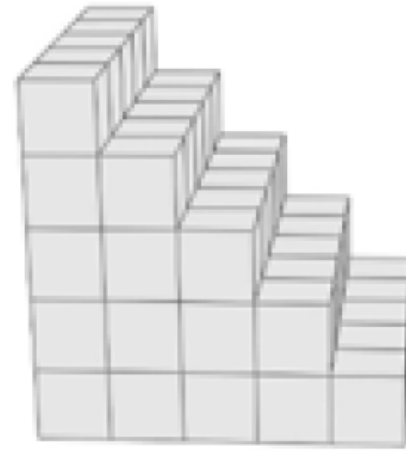
Pseudo-Polycube
 Q



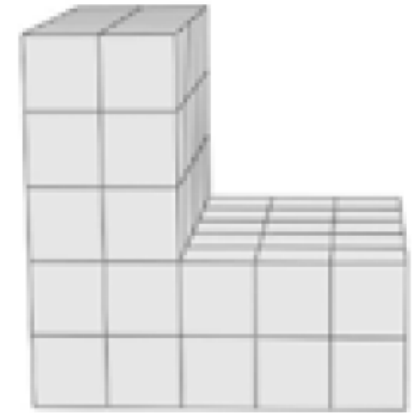
Initial polycube
 \tilde{P}



Opt. polycube
with E_c



Opt. polycube
with E_g

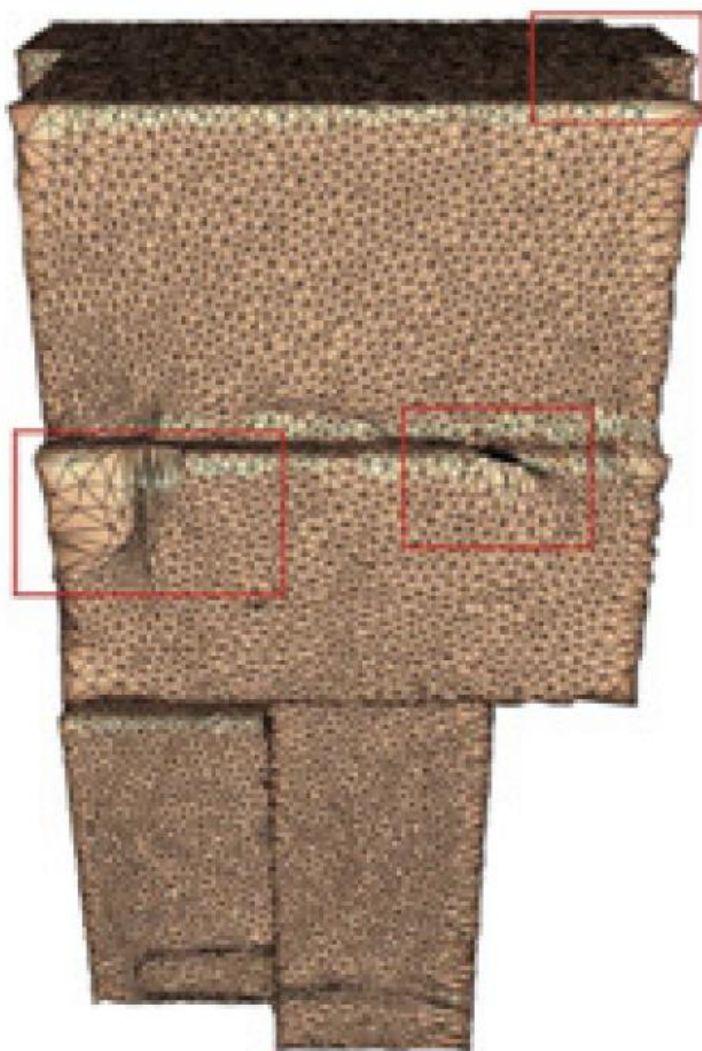


Opt. polycube
with $E_c + 20E_g$

Domain simplicity $E_c + \alpha$ Geometric deviation E_g
Morphological operations

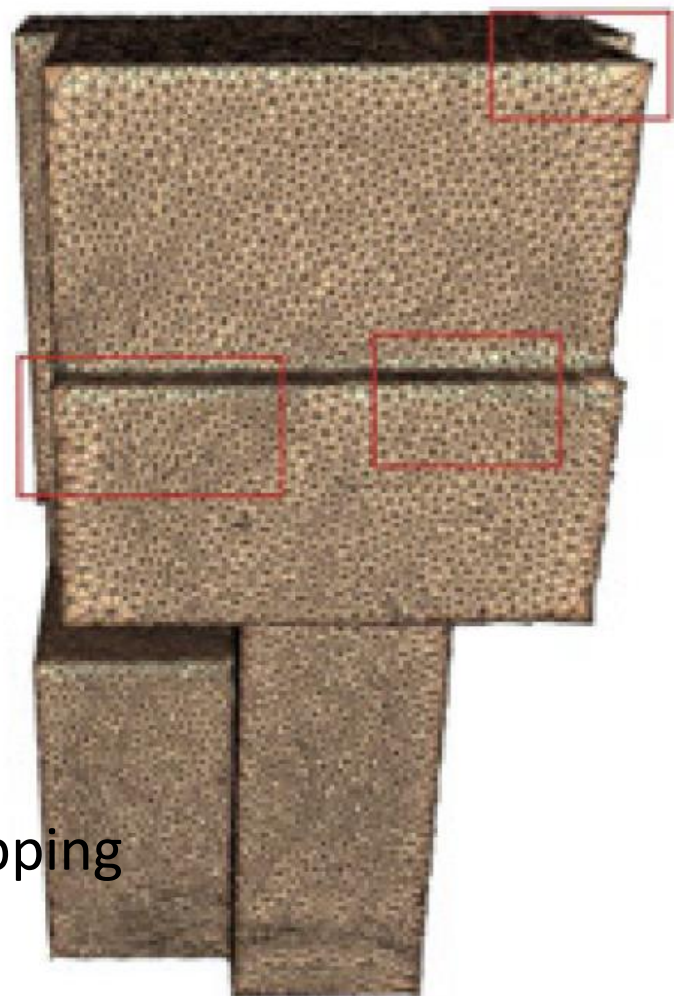
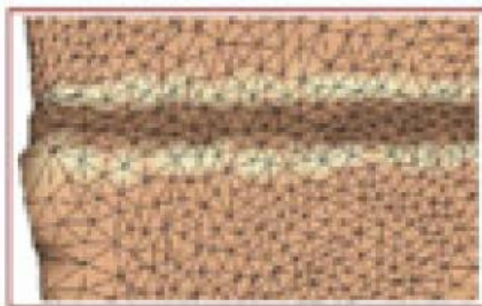
PolyCube volumetric parameterization

e



Projection

f



Fixed boundary mapping

More papers based on construction

- Computing Surface PolyCube-Maps by Constrained Voxelization (PG 2019)

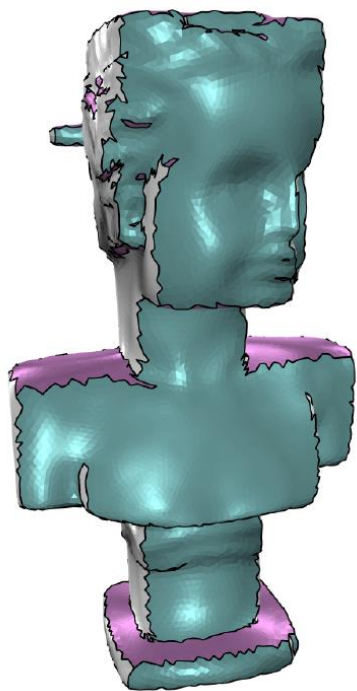
Input:

A source mesh &
a pre-axis-aligned shape

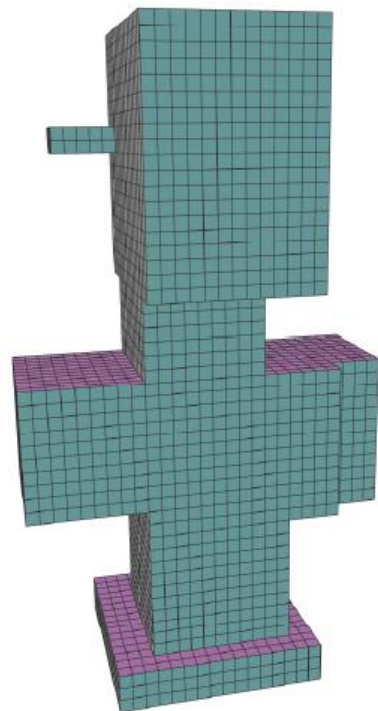
Algorithm workflow

Output:

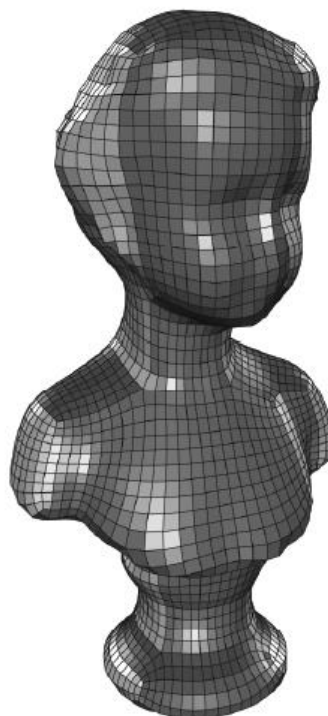
PolyCube-map



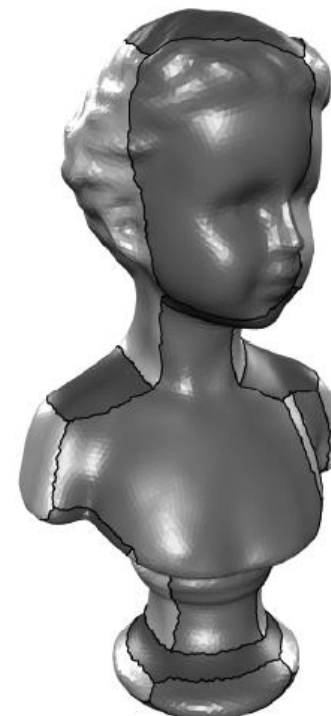
Pre-axis-aligned
shape A



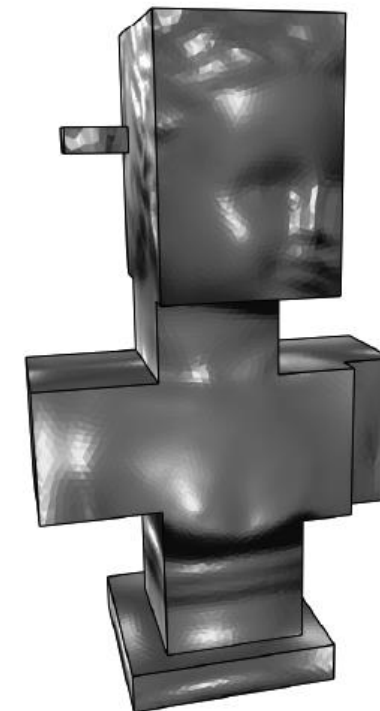
Constructed
PolyCube C



Optimized
quad mesh Q



Segmentation S



PolyCube-map f

I. Constrained voxelization

II. Computing surface PolyCube-Map

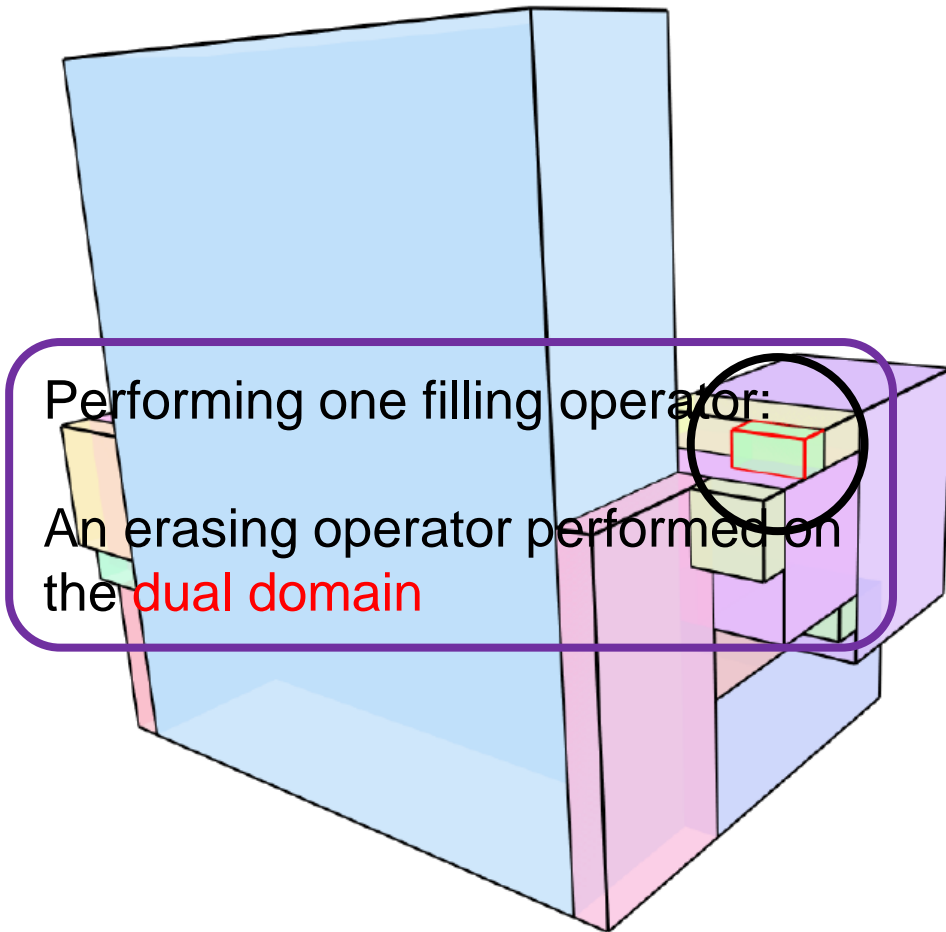
Constrained voxelization: Formulation

$\min_{\mathcal{C}} \rightarrow N(\mathcal{C})$ The number of corners of the PolyCube

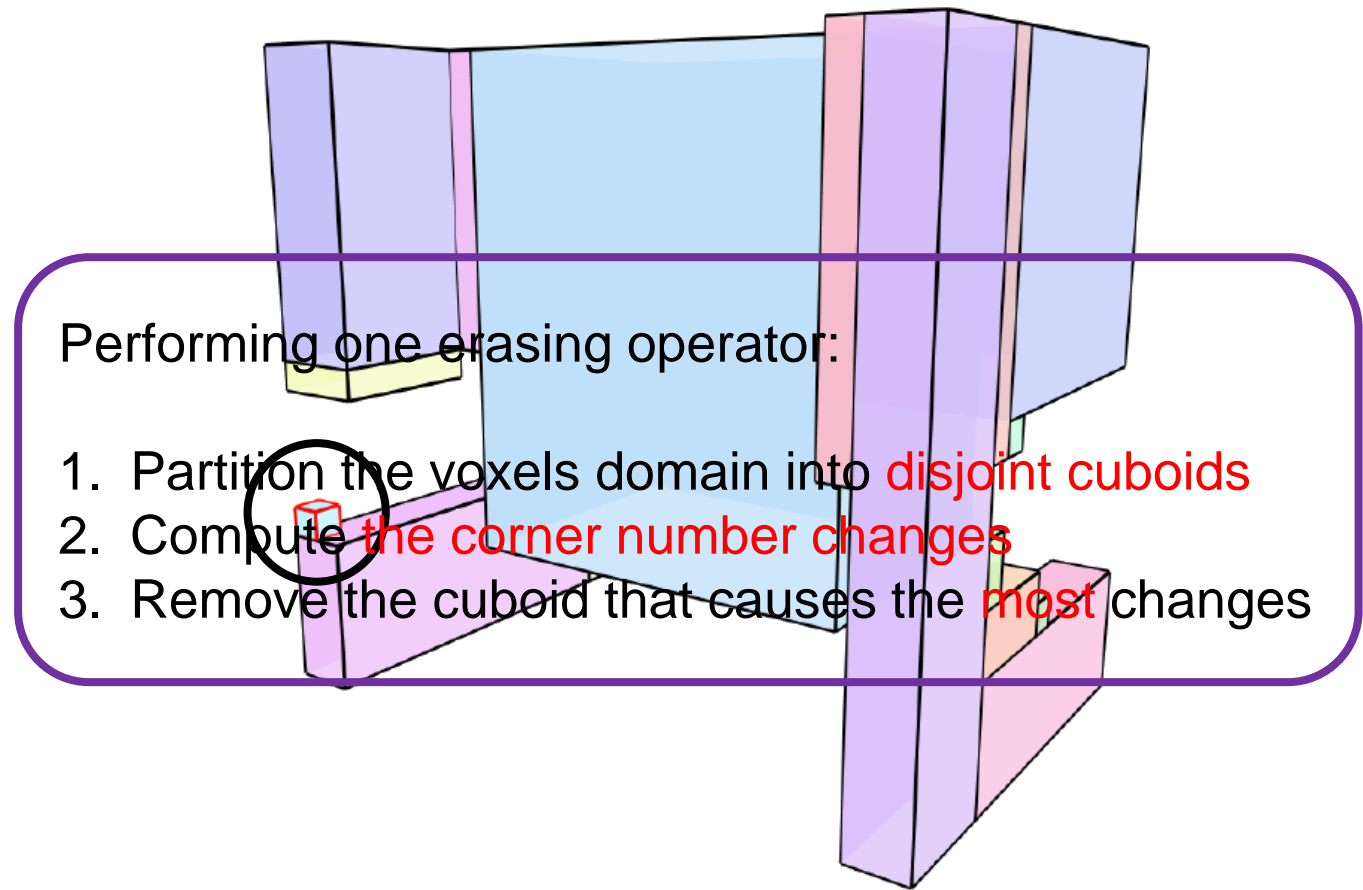
s.t. $d_h(\mathcal{C}, \mathcal{A}) \leq K_{err}$ The K_{err} -bounded constraint

$\mathcal{C} \in \mathcal{T}$ The topological constraint

Two operators

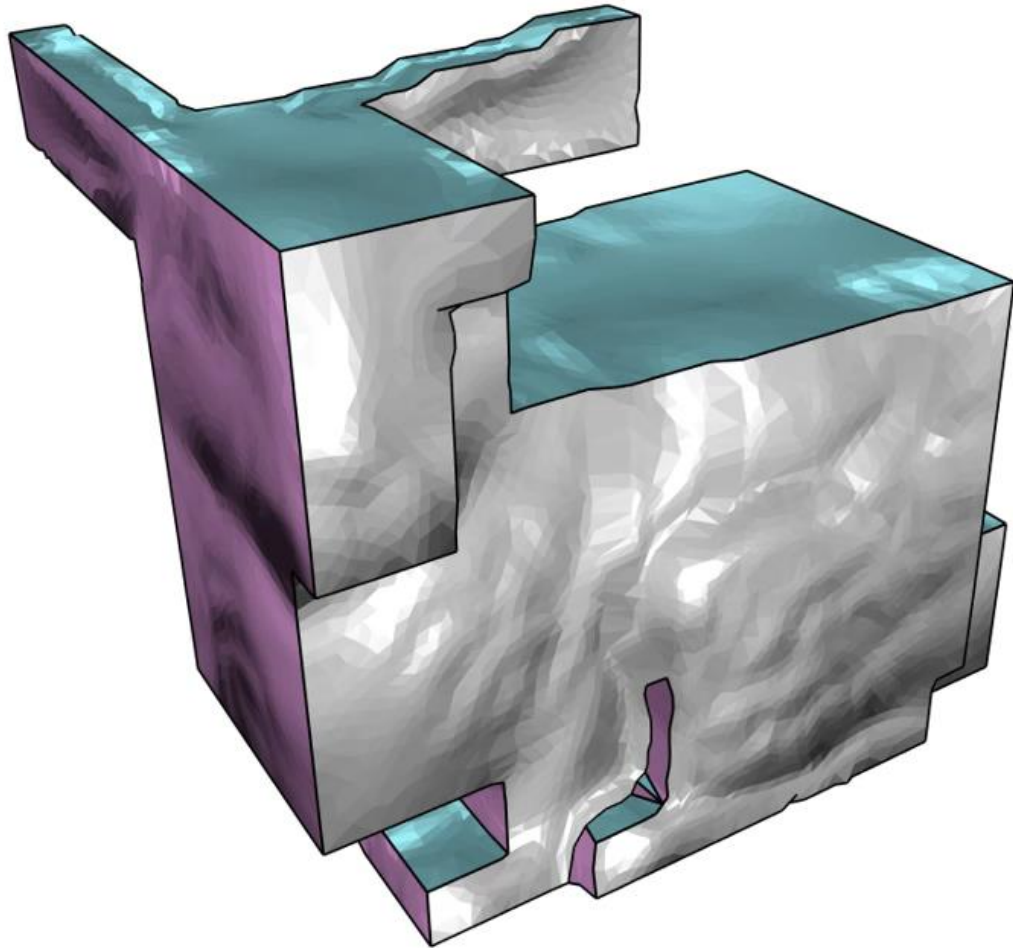


Erasing operator



Filling operator

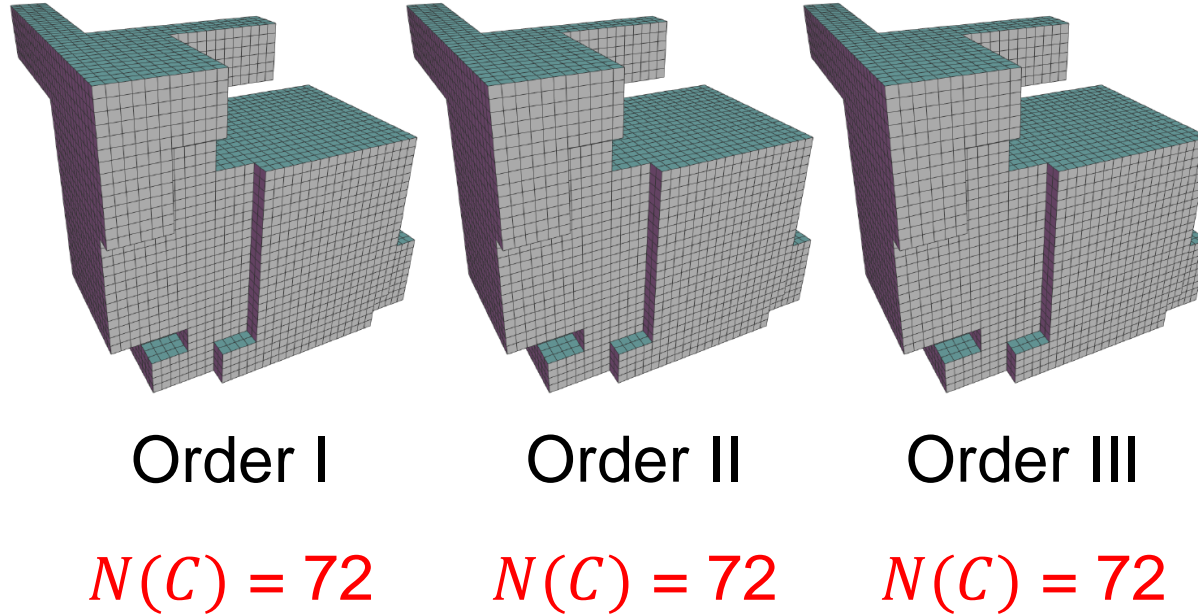
Erasing-and-filling Strategy



Input

1. Generate the **initialization**, $k = 1$
2. Perform **one erasing operator** without violating the constraints
3. Perform **one filling operator** without violating the constraints
4. $k = k + 1$, go to step 2

Erasing-and-filling Strategy



Our developed erasing and filling operators are **effective** and **robust**

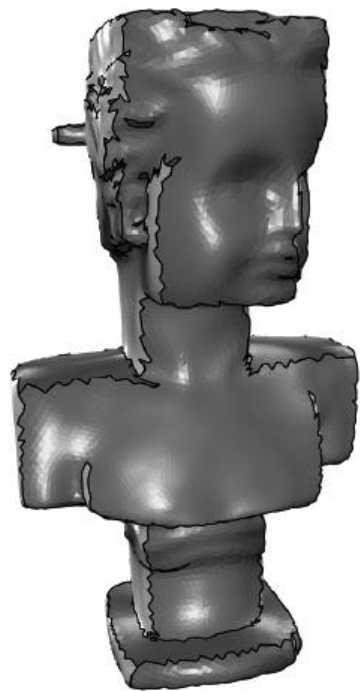
Input:

A source mesh &
a pre-axis-aligned shape

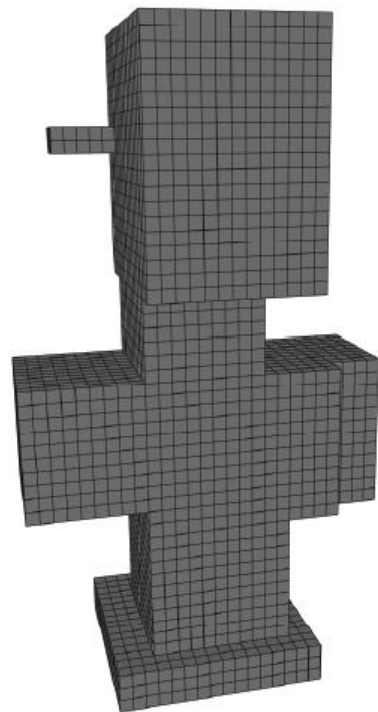
Algorithm workflow

Output:

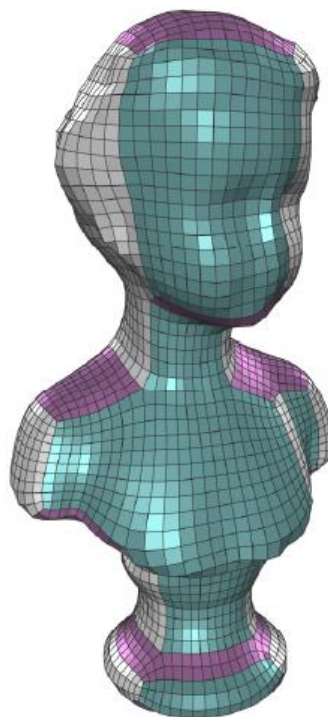
PolyCube-map



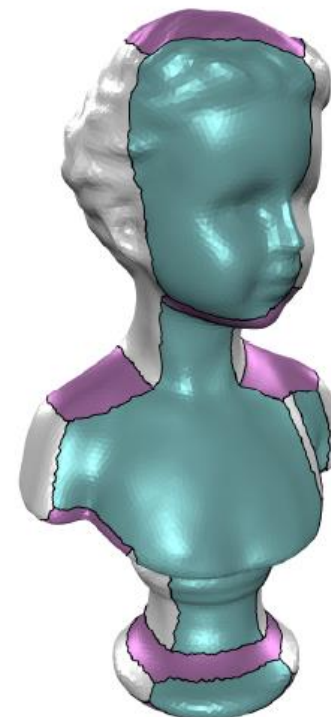
Pre-axis-aligned
shape A



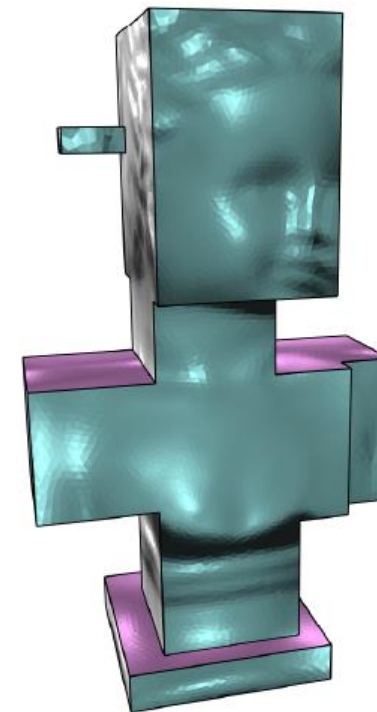
Constructed
PolyCube C



Optimized
quad mesh Q



Segmentation S



PolyCube-map f

I. Constrained voxelization

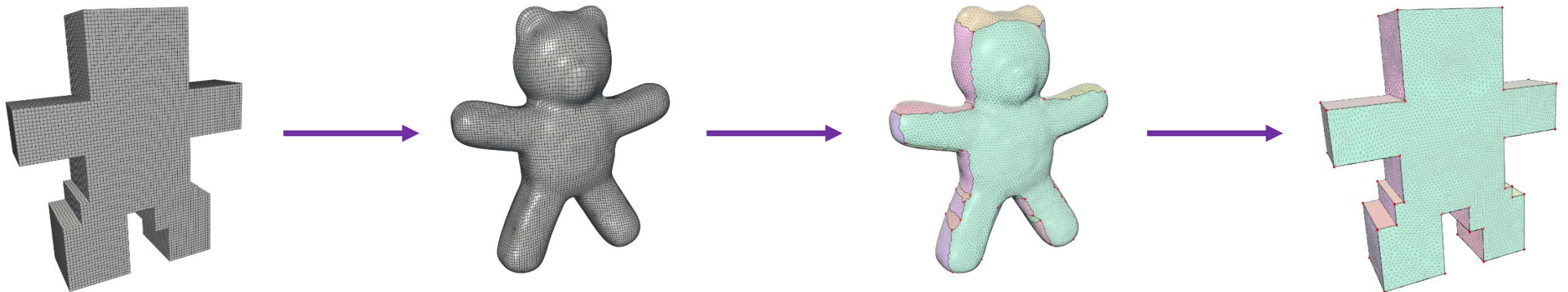
II. Computing surface PolyCube-Map

PolyCube-Maps computation

1. Optimize a new quad mesh

2. Segment the triangular surface into a set of submeshes

3. Map each submesh onto one PolyCube chart



Quad mesh optimization

Requirements

1. Preserve the given shape with the **same** connectivity of the Polycube

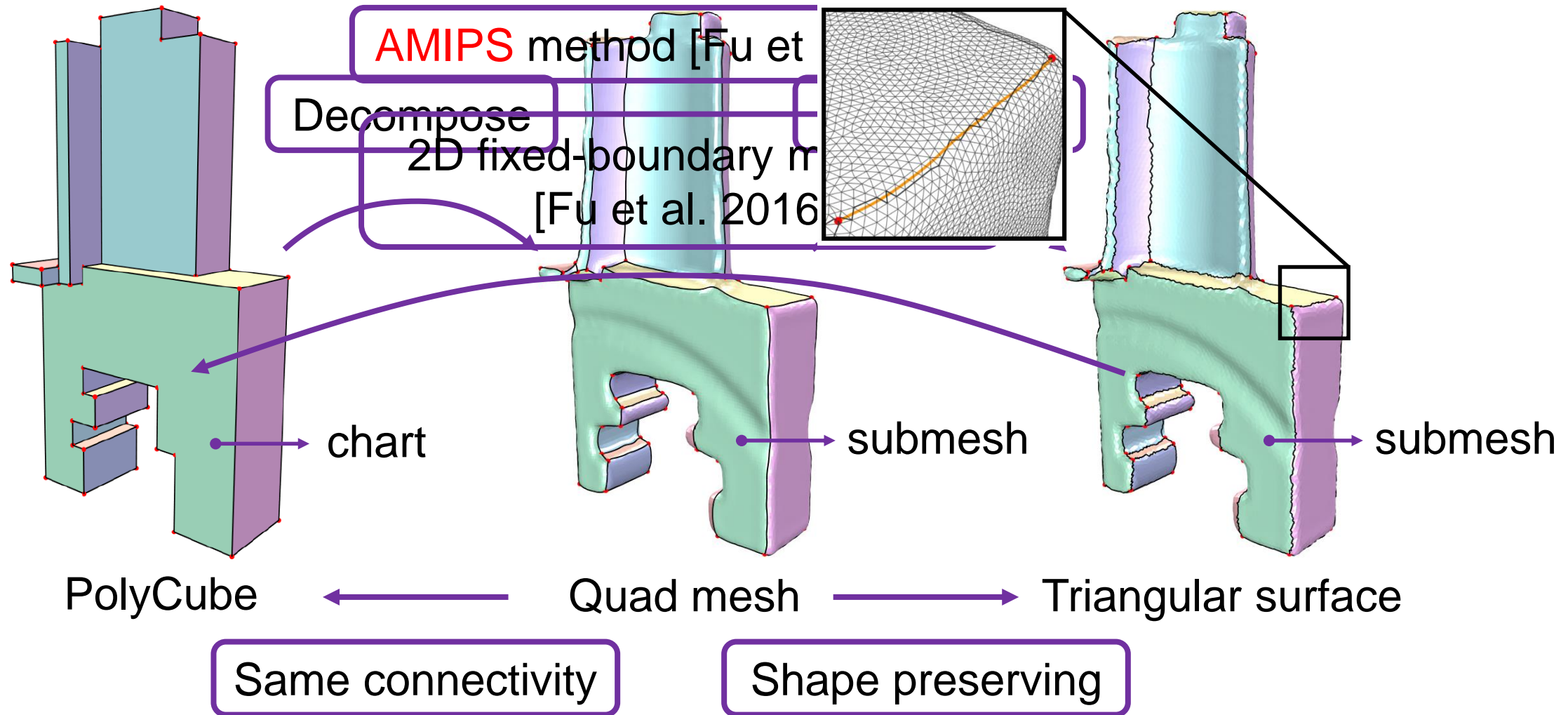
2. All edge lengths are equal to a **constant**

3. All interior angles are in an **interval** $[\frac{\pi}{2} - \theta, \frac{\pi}{2} + \theta]$

4. Almost **no flipped** quads


Anderson acceleration [Peng et al. 2018]

Segmentation and map computation

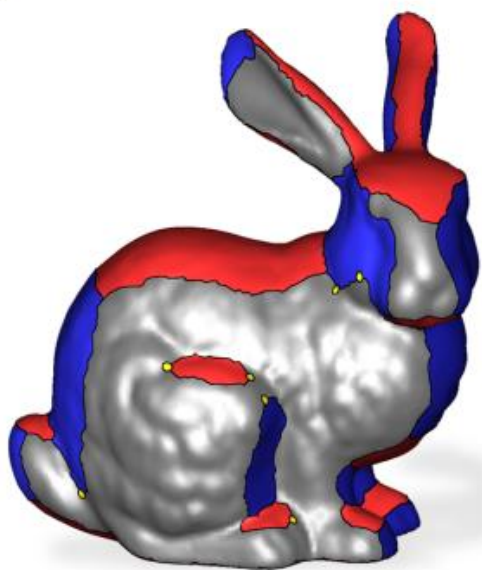


Outlines

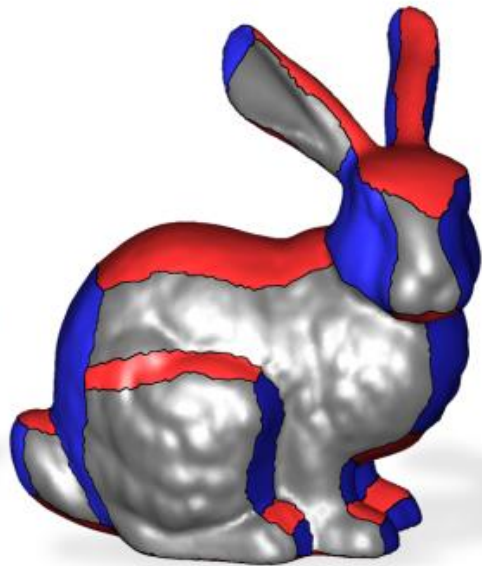
- Definition
- Deformation-based method
 - All-Hex Mesh Generation via Volumetric PolyCube Deformation
- Voxel-based method
 - Optimizing PolyCube domain construction for hexahedral remeshing
- **Cluster-based method**
 - **PolyCut: Monotone Graph-Cuts for PolyCube Base-Complex Construction**
- Generalized PolyCube

Pipeline

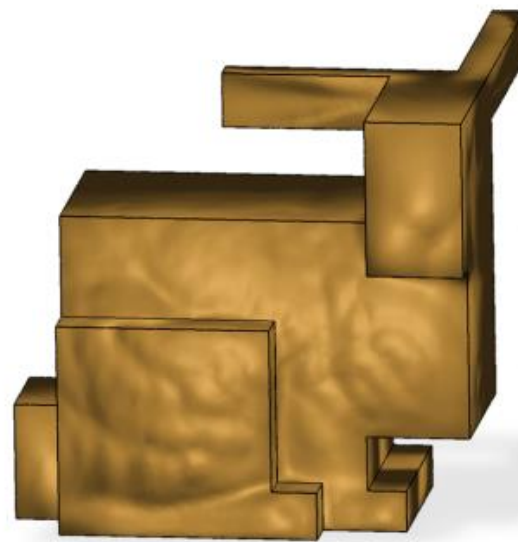
PolyCut Segmentation



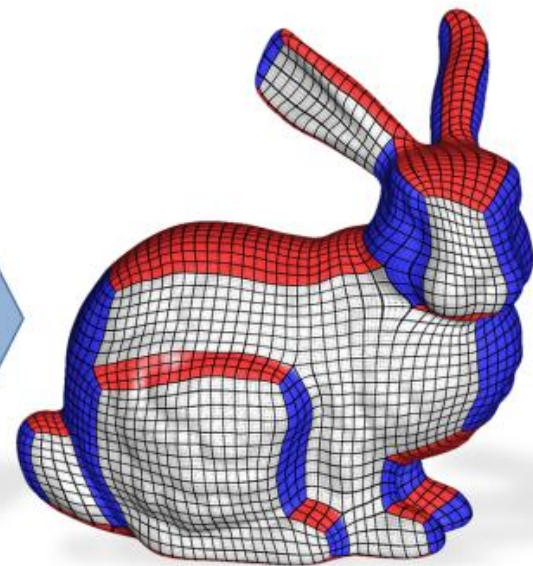
a. Initial Labelling



b. Discrete Optimization



c. PolyCube



d. Parameterization

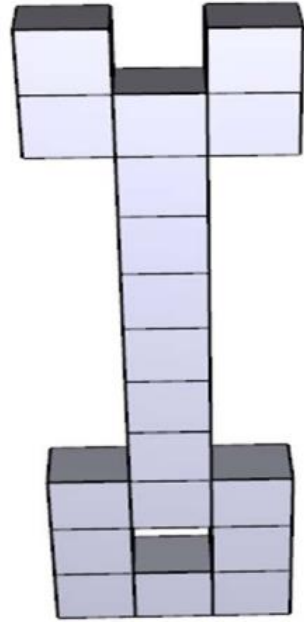
Outlines

- Definition
- Deformation-based method
 - All-Hex Mesh Generation via Volumetric PolyCube Deformation
- Voxel-based method
 - Optimizing PolyCube domain construction for hexahedral remeshing
- Cluster-based method
 - PolyCut: Monotone Graph-Cuts for PolyCube Base-Complex Construction
- Generalized PolyCube

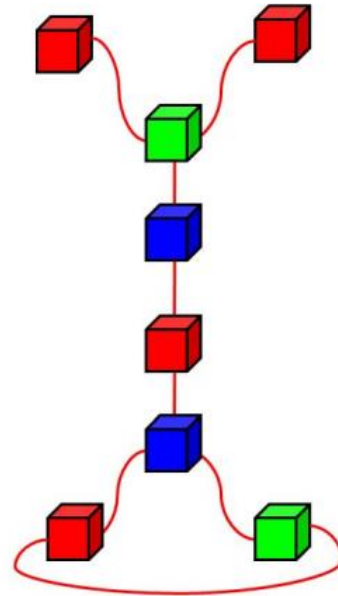
Definitions – Thinking from topology



(a)



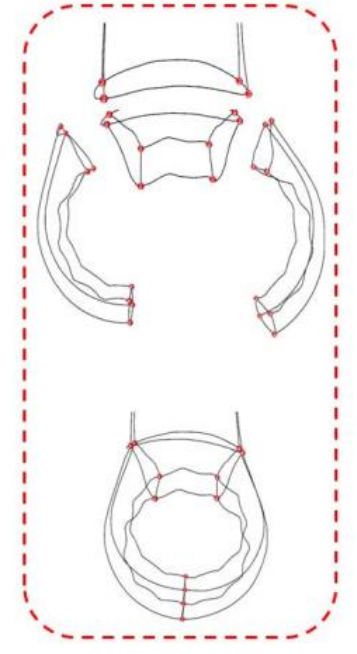
(b)



(c)



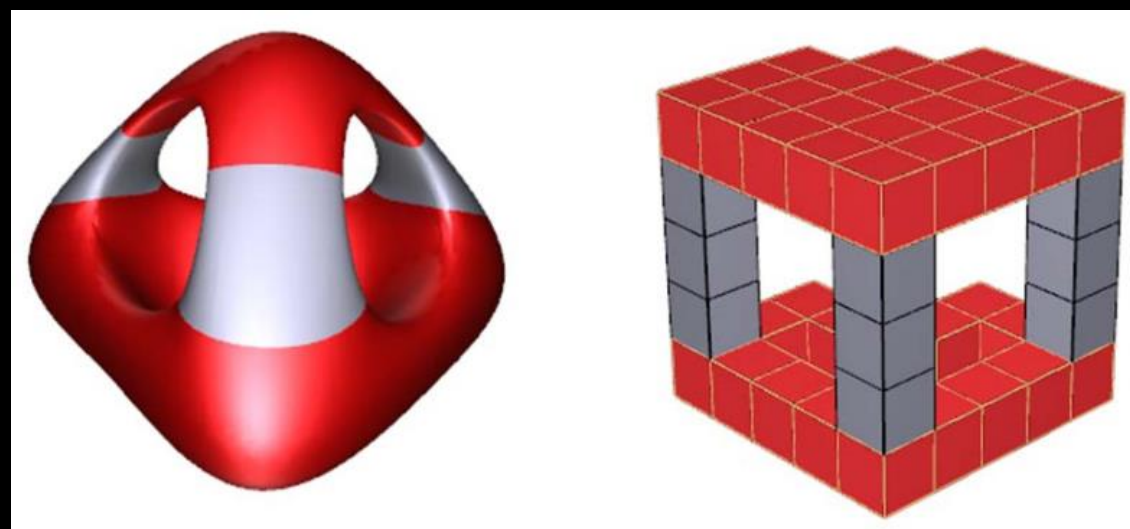
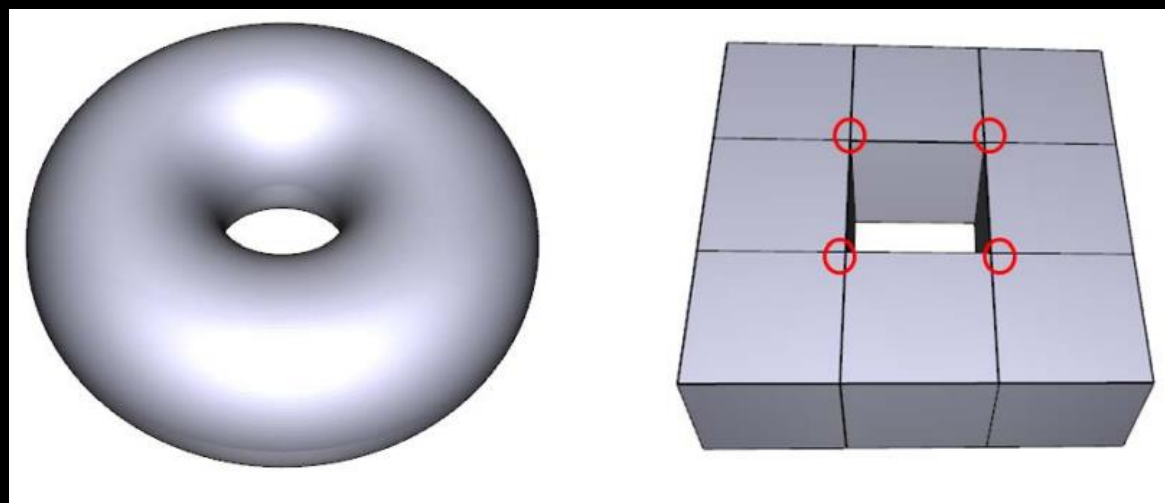
(d)



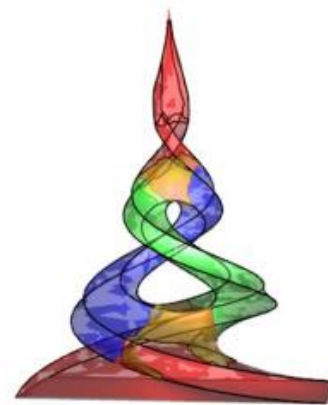
(e)

Generalized poly-cubes: (a) The wrench model; (b) The conventional poly-cube (CPC); (c) The generalized poly-cube (GPC) as a topological graph; (d-e) The cuboid edges are overlaid onto the model to visualize the GPC global structure.

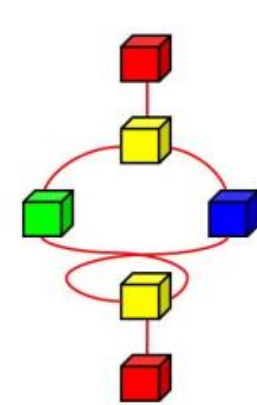
More examples



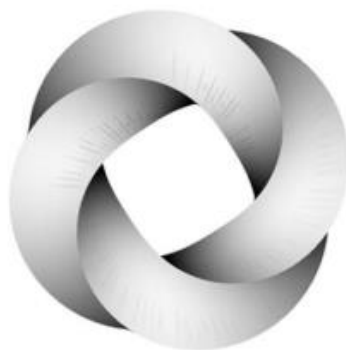
(a)



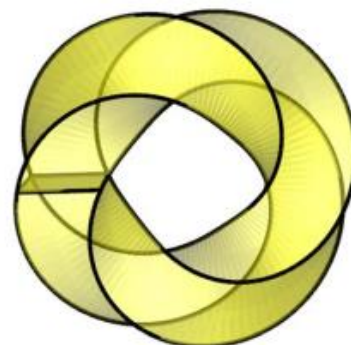
(b)



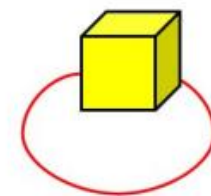
(c)



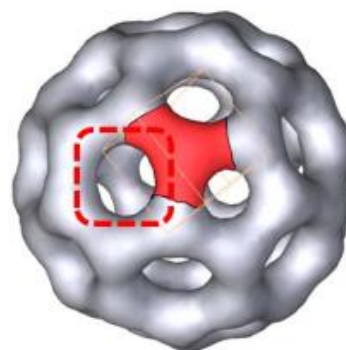
(d)



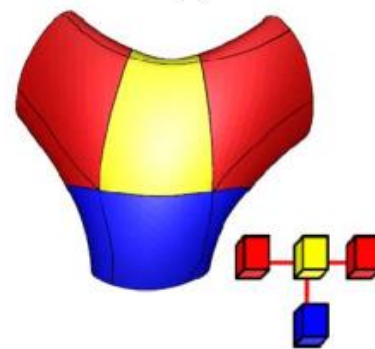
(e)



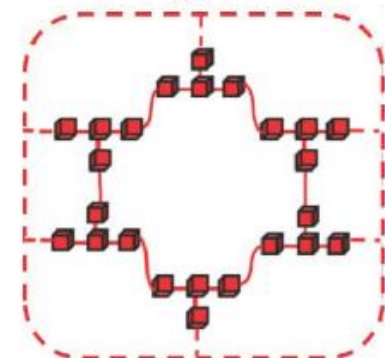
(f)



(g)



(h)

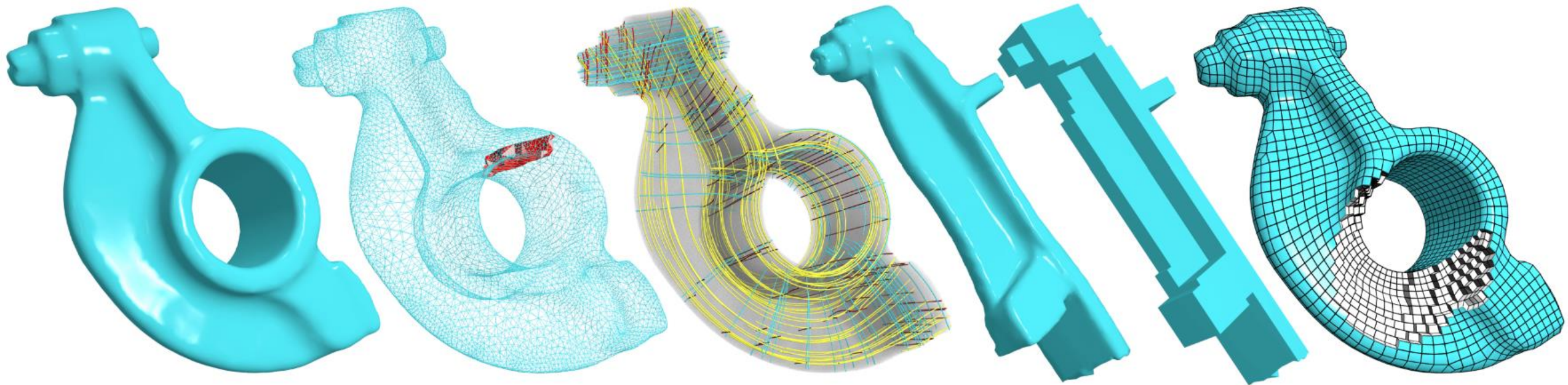


(i)

Difference

- Conventional PolyCube:
 - A shape composes of **axis-aligned unit cubes** that abut with each other.
 - **Unit cubes** as the building block.
 - All cubes are glued together and embedded in the 3D space.
- Generalized PolyCube:
 - A shape composes of a set of cuboids glued together **topologically**.
 - Topological simplicity and elegance.

Paper: All-Hex Meshing using Closed-Form Induced Polycube



Cut faces

Frame field

Deformed
cut mesh

PolyCube

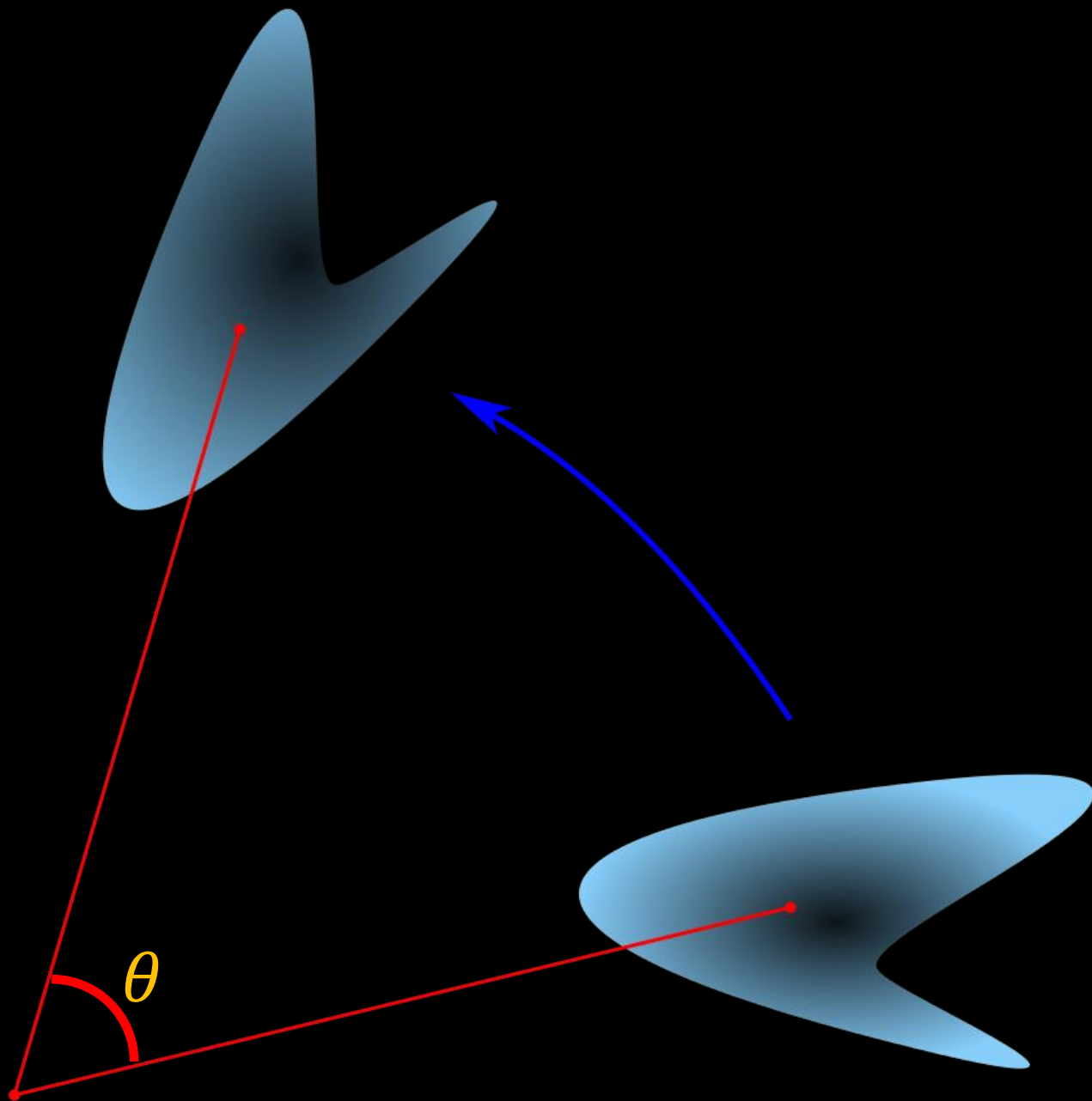
Rotation

<https://en.wikipedia.org/wiki/Rotation>

- A **rotation** is a circular movement of an object around a center (or point) of rotation.
- A three-dimensional object can always be rotated around an infinite number of imaginary lines called *rotation axes*.
- A rotation is a **rigid** body movement.

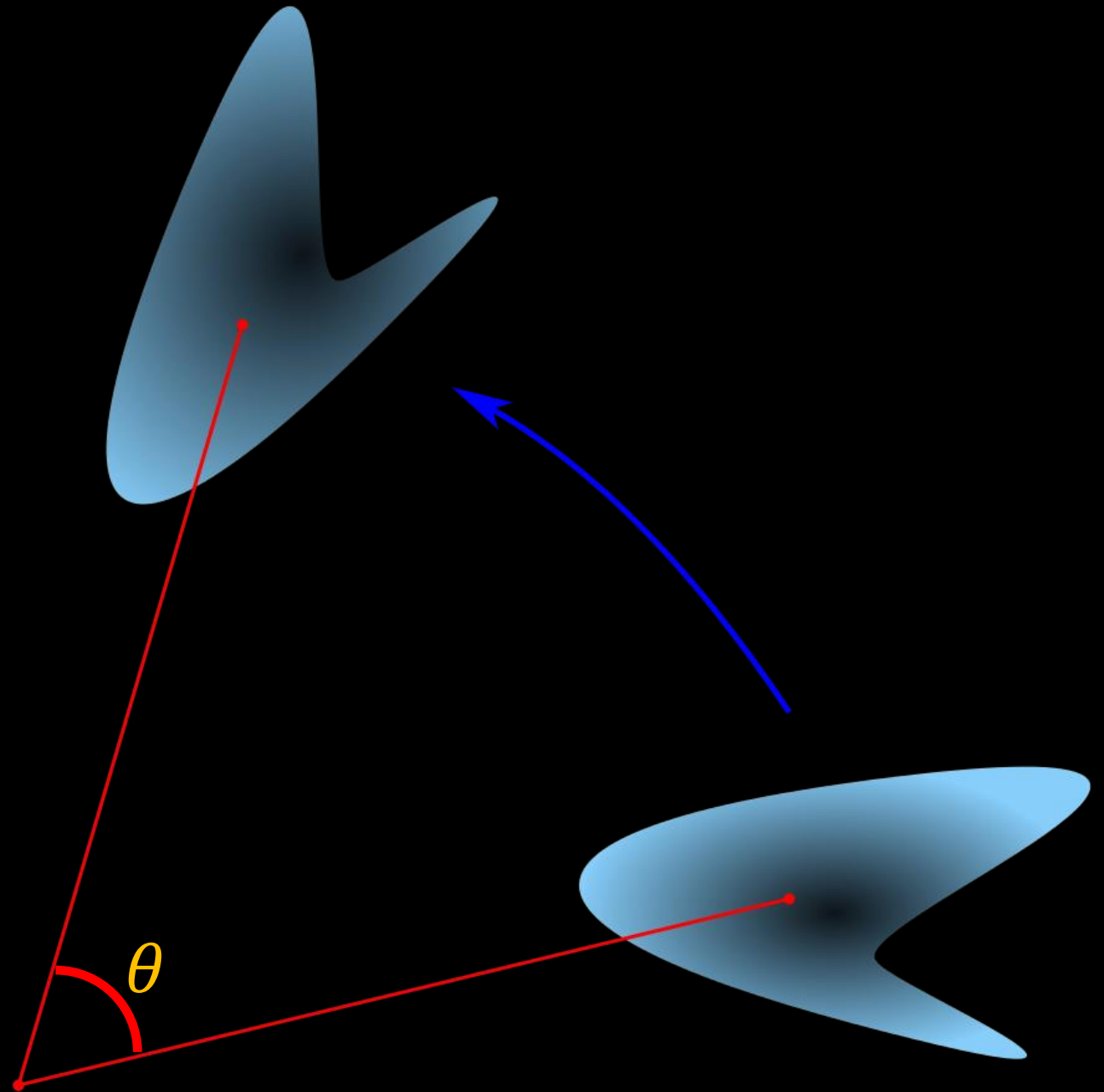
2D rotation

$$\bullet R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$



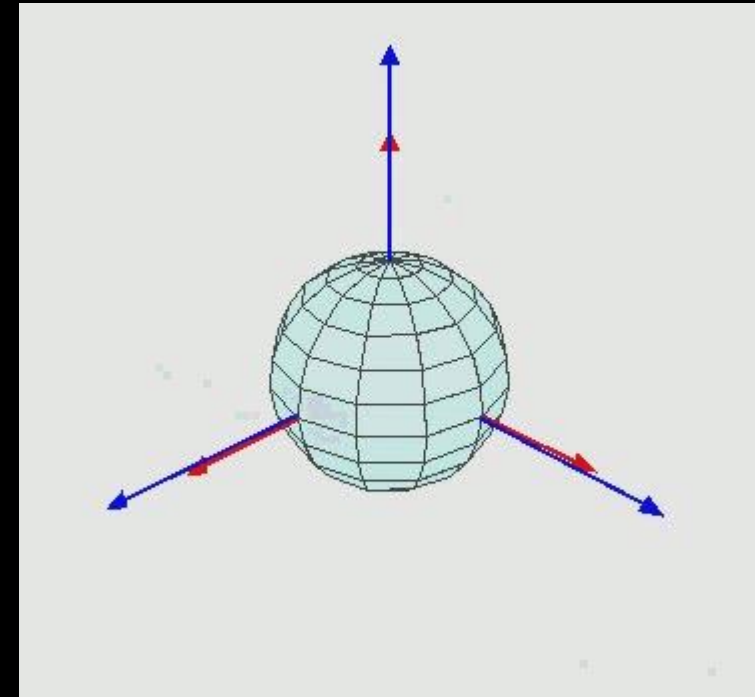
Complex view

- $z' = e^{i\theta} z$
- z' has the same length as z .
- z' is rotated by θ degrees.



Euler angles

- The Euler angles are **three** angles introduced by Leonhard Euler to describe the orientation of a rigid body with respect to a fixed coordinate system.
- Any orientation can be achieved by composing three elemental rotations, i.e. **rotations about the axes of a coordinate system**. Euler angles can be defined by three of these rotations.



Quaternion

<https://en.wikipedia.org/wiki/Quaternion>

- Form: $q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$
 - a, b, c, d are real numbers
 - $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are the fundamental *quaternion units*
 - $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$
- Componentwise addition
 - $(a_1 + b_1\mathbf{i} + c_1\mathbf{j} + d_1\mathbf{k}) + (a_2 + b_2\mathbf{i} + c_2\mathbf{j} + d_2\mathbf{k}) = (a_1 + a_2) + (b_1 +$

Quaternion

- $ij = k, ji = -k; jk = i, kj = -i; ki = j, ik = -j.$

- Multiplication (*Hamilton product*)

$$\begin{aligned} & (a_1 + b_1\mathbf{i} + c_1\mathbf{j} + d_1\mathbf{k})(a_2 + b_2\mathbf{i} + c_2\mathbf{j} + d_2\mathbf{k}) \\ &= a_1(a_2 + b_2\mathbf{i} + c_2\mathbf{j} + d_2\mathbf{k}) + b_1\mathbf{i}(a_2 + b_2\mathbf{i} + c_2\mathbf{j} + d_2\mathbf{k}) \\ &+ c_1\mathbf{j}(a_2 + b_2\mathbf{i} + c_2\mathbf{j} + d_2\mathbf{k}) + d_1\mathbf{k}(a_2 + b_2\mathbf{i} + c_2\mathbf{j} + d_2\mathbf{k}) \\ &= a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2 + (a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2)\mathbf{i} \\ &+ (a_1c_2 - b_1d_2 + c_1a_2 + d_1b_2)\mathbf{j} + (a_1d_2 + b_1c_2 - c_1b_2 + d_1a_2)\mathbf{k} \end{aligned}$$

noncommutative

Conjugation, the norm, and reciprocal

- Conjugation:

- $q^* = a - bi - cj - dk$

- Norm:

- $\|q\| = \sqrt{qq^*} = \sqrt{a^2 + b^2 + c^2 + d^2}$

- Reciprocal:

- $q^{-1} = \frac{q^*}{\|q\|^2}$

Unit quaternion

- Because the vector part of a quaternion is a vector in R^3 , the geometry of R^3 is reflected in the algebraic structure of the quaternions.
- A single rotation by a given angle θ about a fixed axis $u = xi + yj + zk$
- An **extension of Euler's formula**:
 - $q = e^{\frac{\theta}{2}(xi+yj+zk)} = \cos\frac{\theta}{2} + (xi + yj + zk) \sin\frac{\theta}{2}$
- The desired rotation can be applied to an ordinary vector $p = p_x i + p_y j + p_z k$
 - $p' = qpq^{-1}$
 - $q^{-1} = q^* = \cos\frac{\theta}{2} - (xi + yj + zk) \sin\frac{\theta}{2}$

Unit quaternion \Leftrightarrow Rotation Matrix

$$q = w + xi + yj + zk \Leftrightarrow M$$

- Unit quaternion \Rightarrow Rotation Matrix

$$M = \begin{pmatrix} 1 - 2y^2 - 2z^2 & 2xy + 2zw & 2xz - 2yw \\ 2xy - 2zw & 1 - 2x^2 - 2z^2 & 2yz + 2xw \\ 2xz + 2yw & 2yz - 2xw & 1 - 2x^2 - 2y^2 \end{pmatrix}$$

- Unit quaternion \Leftarrow Rotation Matrix

$$\text{Suppose } M = \begin{pmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{pmatrix}$$

$$m_{00} + m_{11} + m_{22} = 3 - 4(x^2 + y^2 + z^2) = 3 - 4(1 - w^2) = -1 + 4w^2 \text{ (ambiguity)}$$

$$m_{01} - m_{10} = 4zw; m_{20} - m_{02} = 4yw; m_{12} - m_{21} = 4xw$$

Directional Field

Xiao-Ming Fu

Outlines

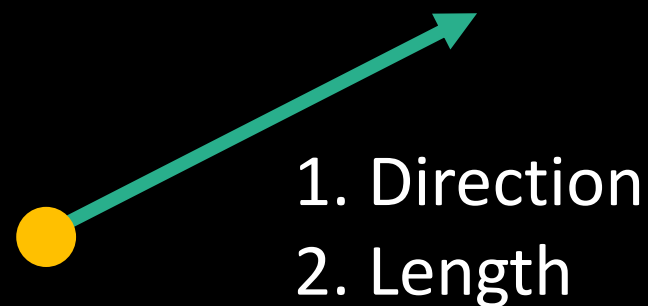
- Introduction
- Discretization
- Representation
- Objectives and Constraints

Outlines

- Introduction
- Discretization
- Representation
- Objectives and Constraints

Definition



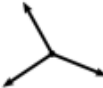






- Spatially-varying **directional information**, assigned to **each point** on a given domain.
 - A field on a domain is the assignment of a directional to each point in the domain.
- Magnitude + direction



Multi-valued field

- **Multiple** directions per point with some notion of symmetry
- A set of directions or vectors at every point.
 - Rotationally-symmetric direction fields (*RoSy fields*)
 - $N = 1, 2, 4, 6$
 - Four directions with $\pi/2$ RoSy
 - Two independent pairs of directions with π RoSy within each pair.

- *Vector fields*
- *Direction fields*
- *Line fields*
- *Cross fields*
- *Frame fields*
- *RoSy fields*
- *N-symmetry fields*
- *PolyVector fields*
- *Tensor fields*

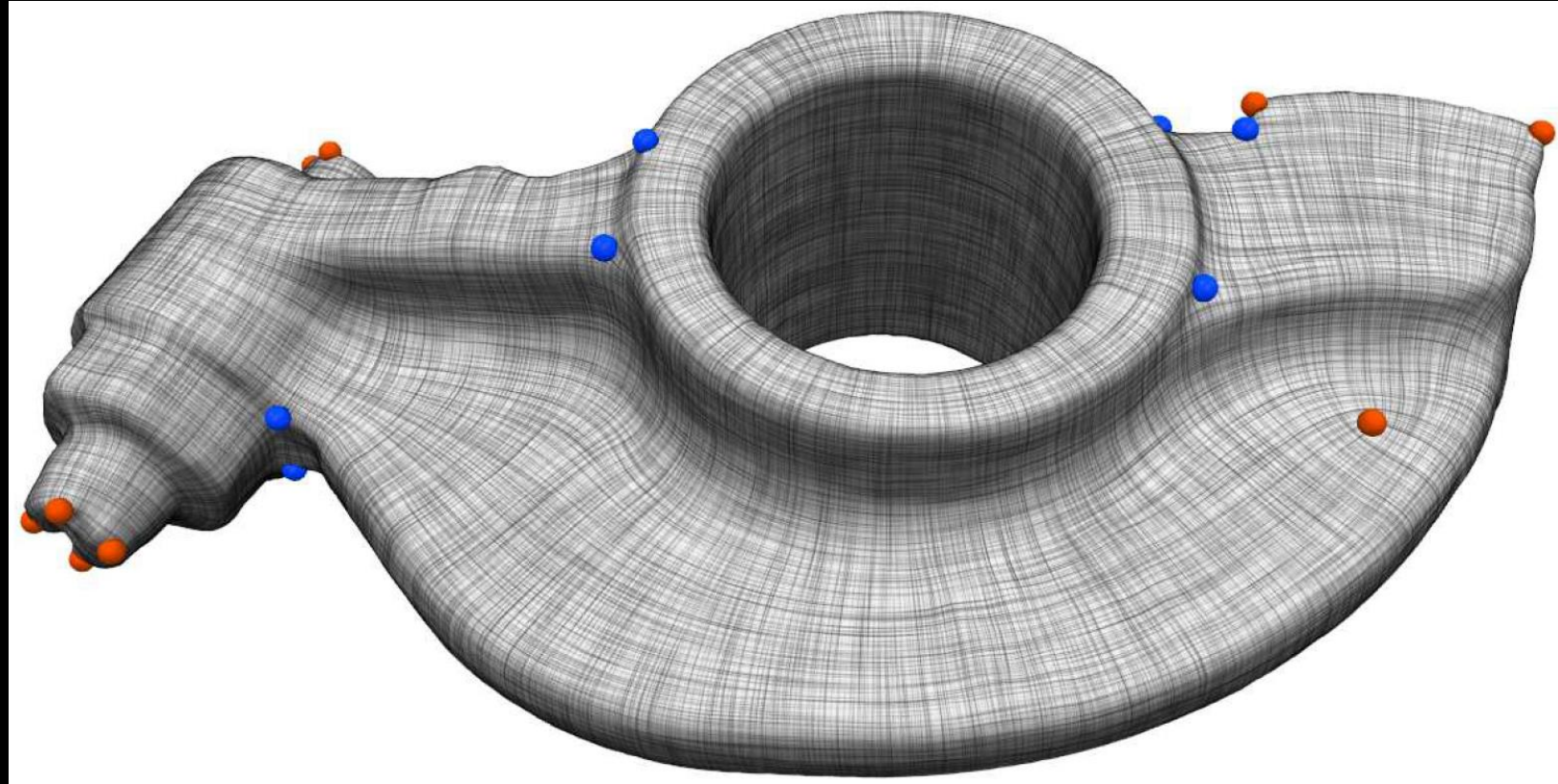
	1-vector field	One vector, classical “vector field”
	2-direction field	Two directions with π symmetry, “line field”, “2-RoSy field”
	1^3 -vector field	Three independent vectors, “3-polyvector field”
	4-vector field	Four vectors with $\pi/2$ symmetry, “non-unit cross field”
	4-direction field	Four directions with $\pi/2$ symmetry, “unit cross field”, “4-RoSy field”
	2^2 -vector field	Two pairs of vectors with π symmetry each, “frame field”
	2^2 -direction field	Two pairs of directions with π symmetry each, “non-ortho. cross field”
	6-direction field	Six directions with $\pi/3$ symmetry, “6-RoSy”
	2^3 -vector field	Three pairs of vectors with π symmetry each

Some concrete examples

- Principal directions of a shape
- Stress or strain tensors
- The gradient of a scalar field
- The advection field of a flow
- Diffusion data from MRI

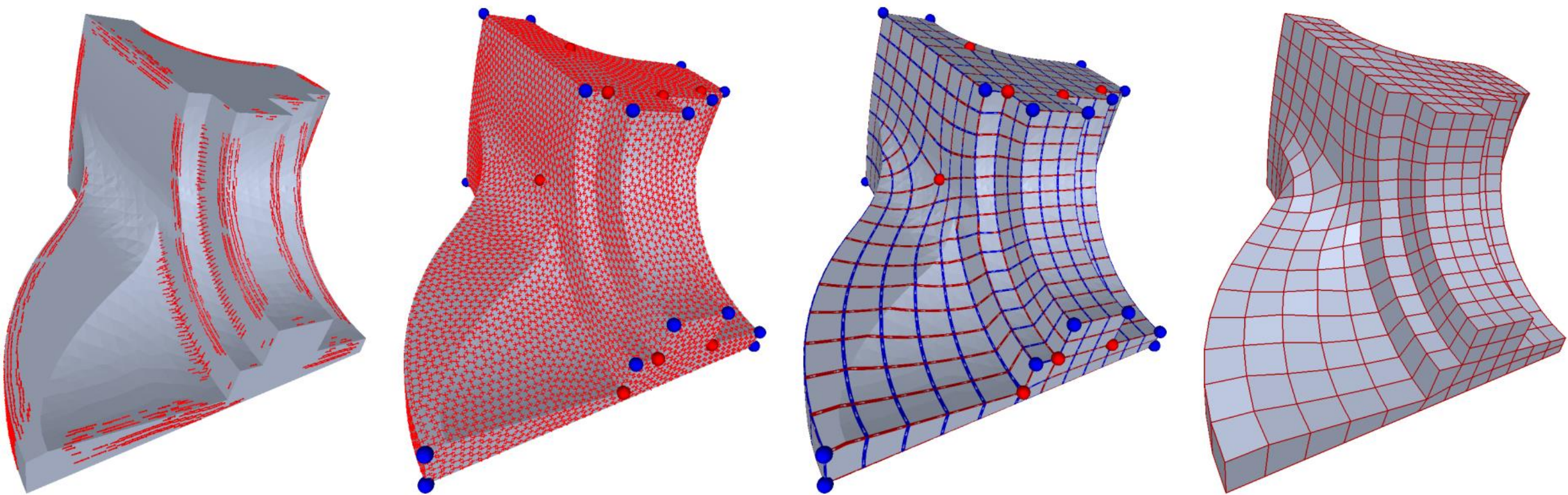
Synthesis or design

- User constraints
- Alignment conditions
- Fairness objectives
- Physical realizations



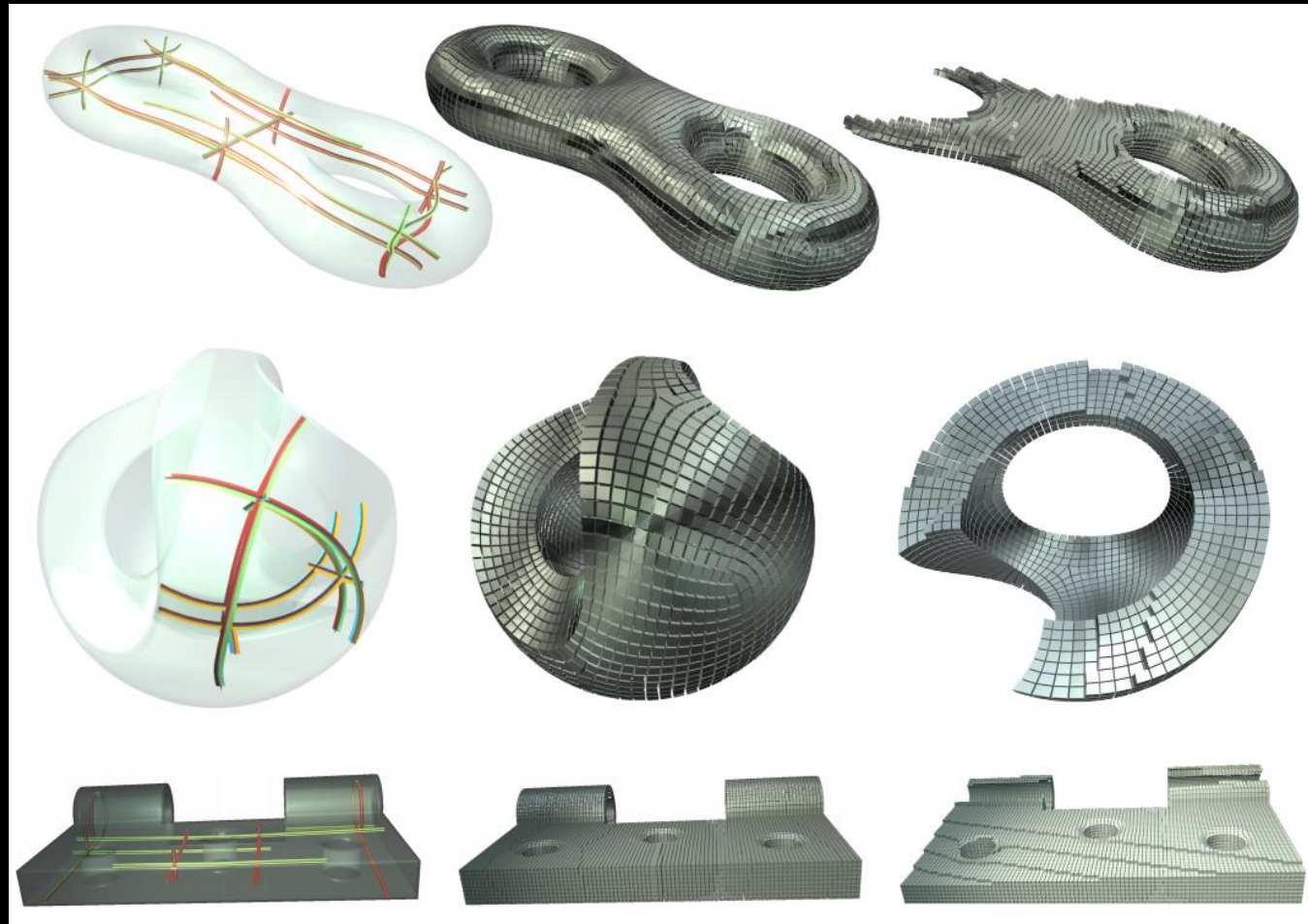
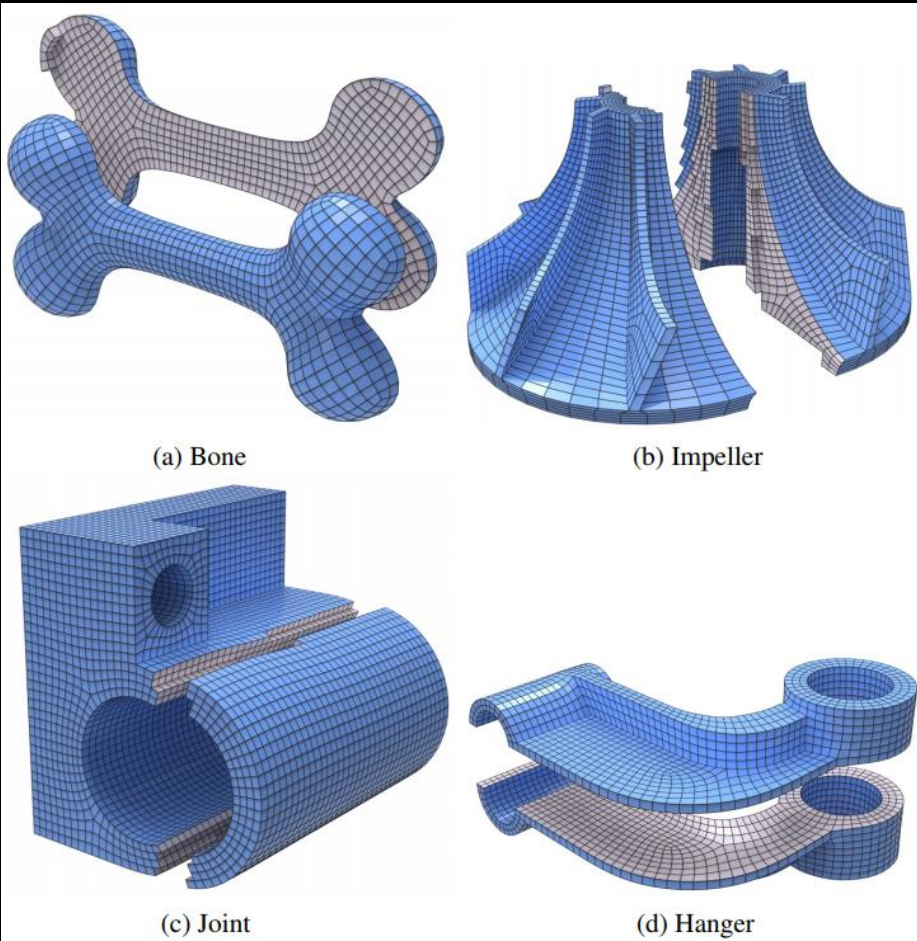
Applications

- Mesh Generation



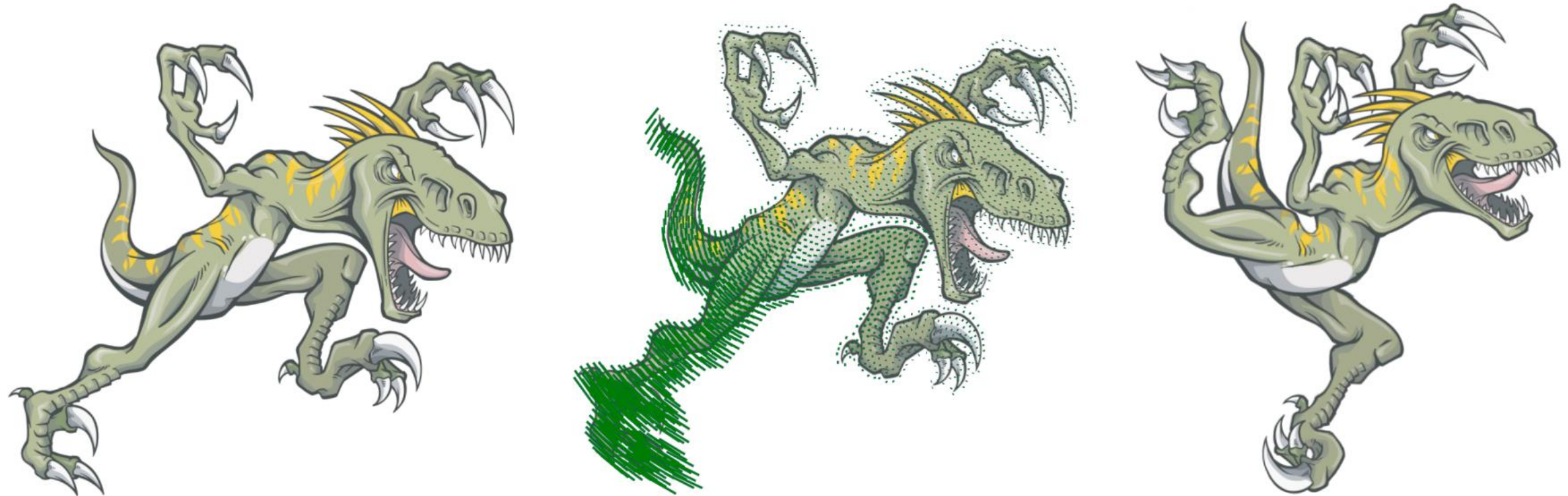
Applications

- All-hex meshing



Deformation

- Deformations which are as isometric as possible can be generated using approximate Killing vector fields.



Texture Mapping and Synthesis

- Seamless texture

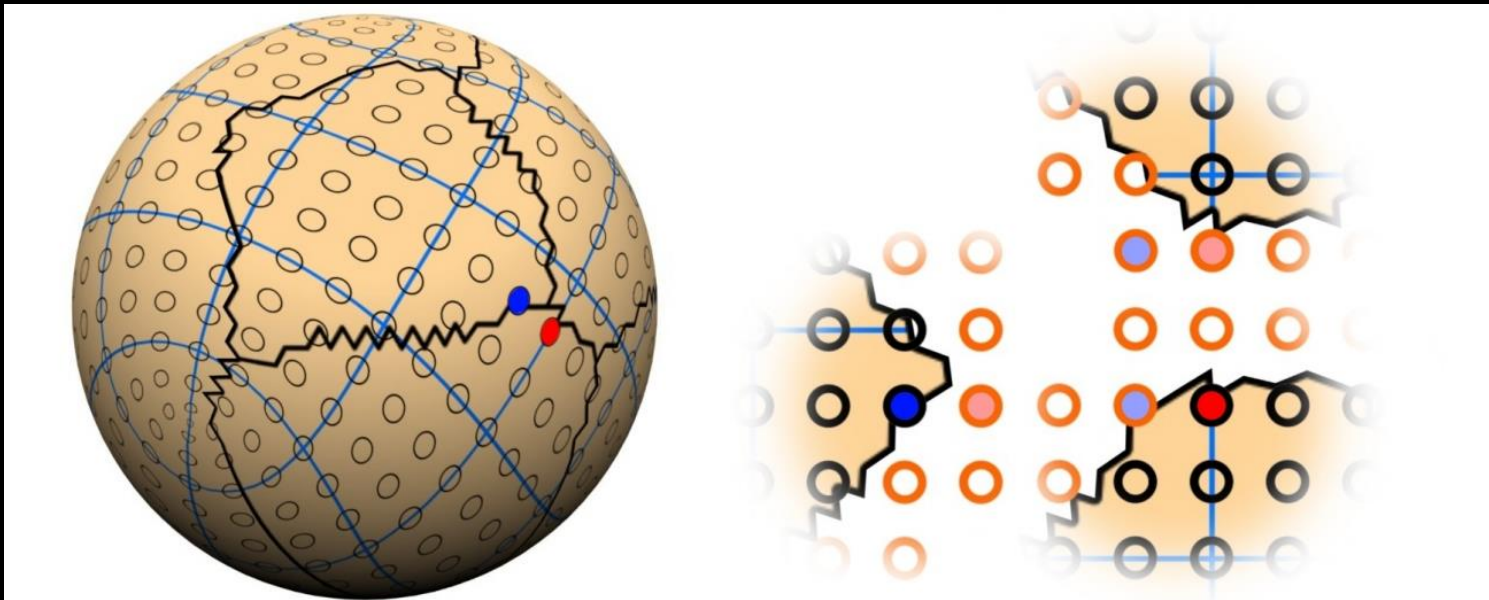
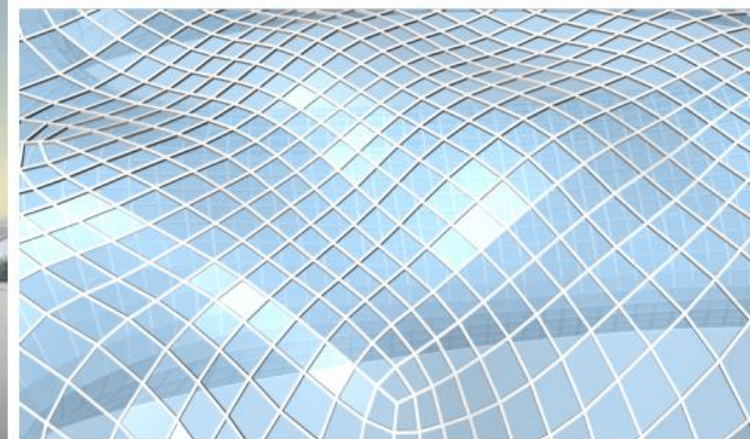
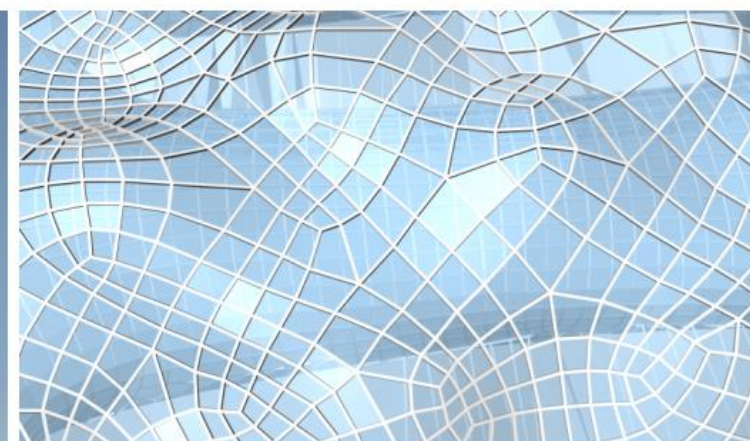
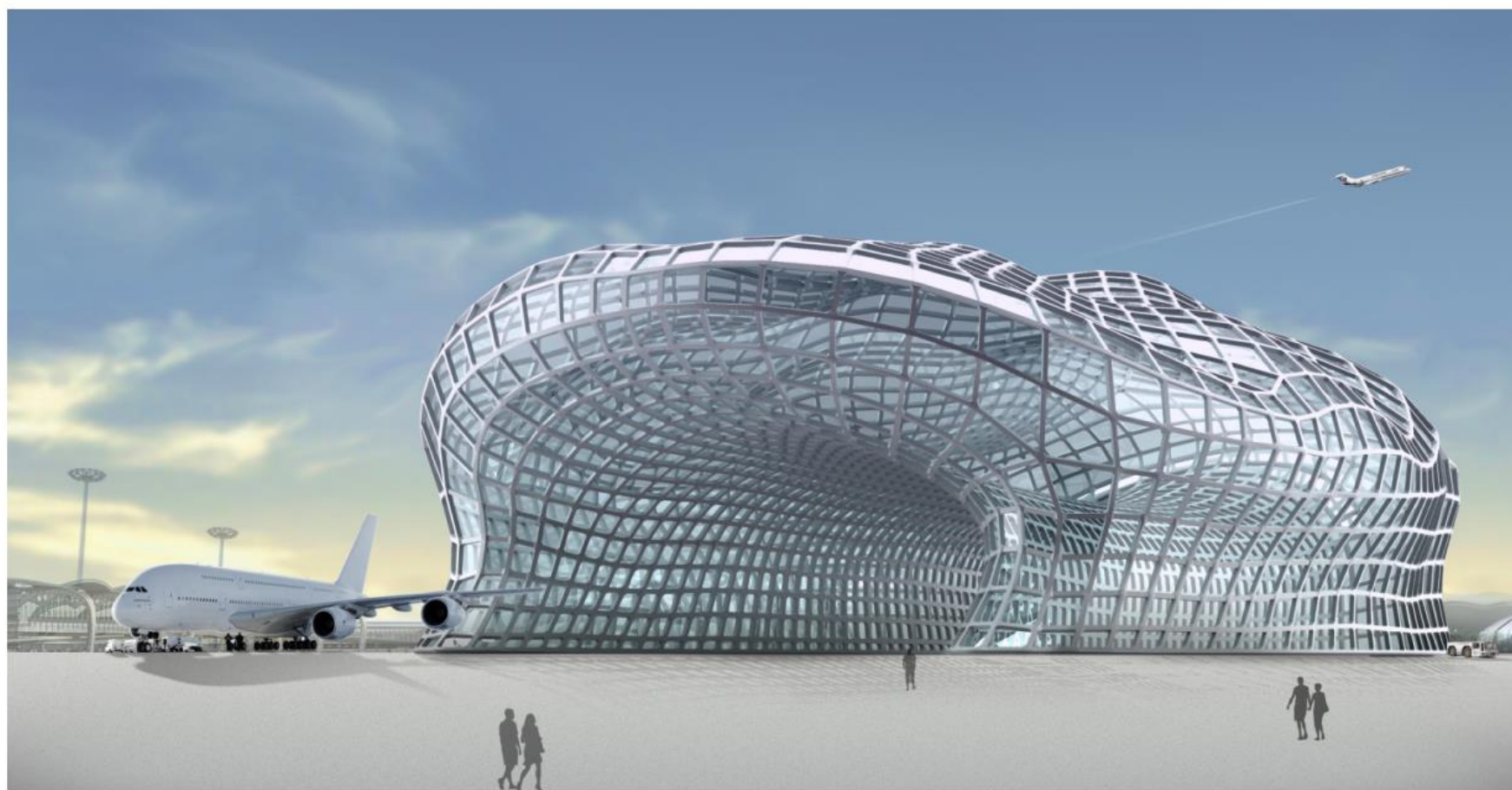


Figure 1: *Neighboring texels on the surface (left, red/blue dots) are not neighbors in the atlas (right). To make the seam invisible, texels have to align across chart boundaries and their colors have to be duplicated (light colored texels).*

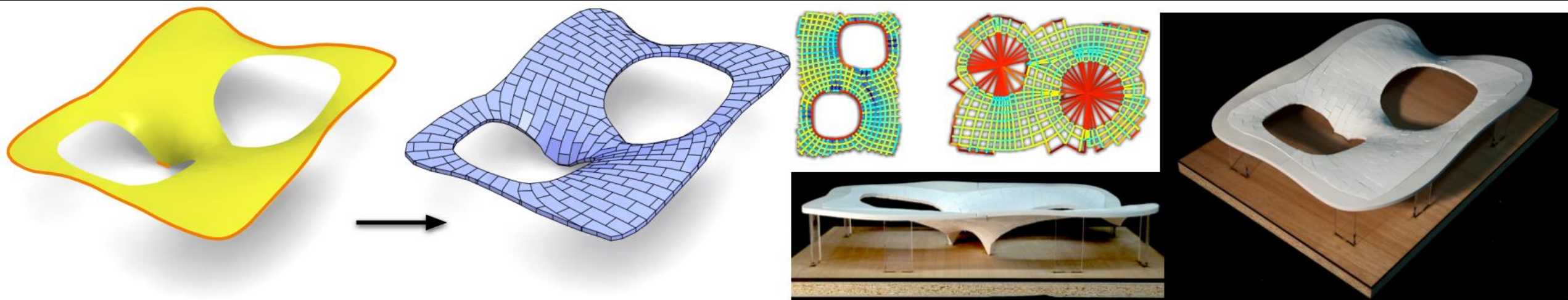
Architectural Geometry

- Conjugate directions



Architectural Geometry

- Self-Supporting Structures



Outlines

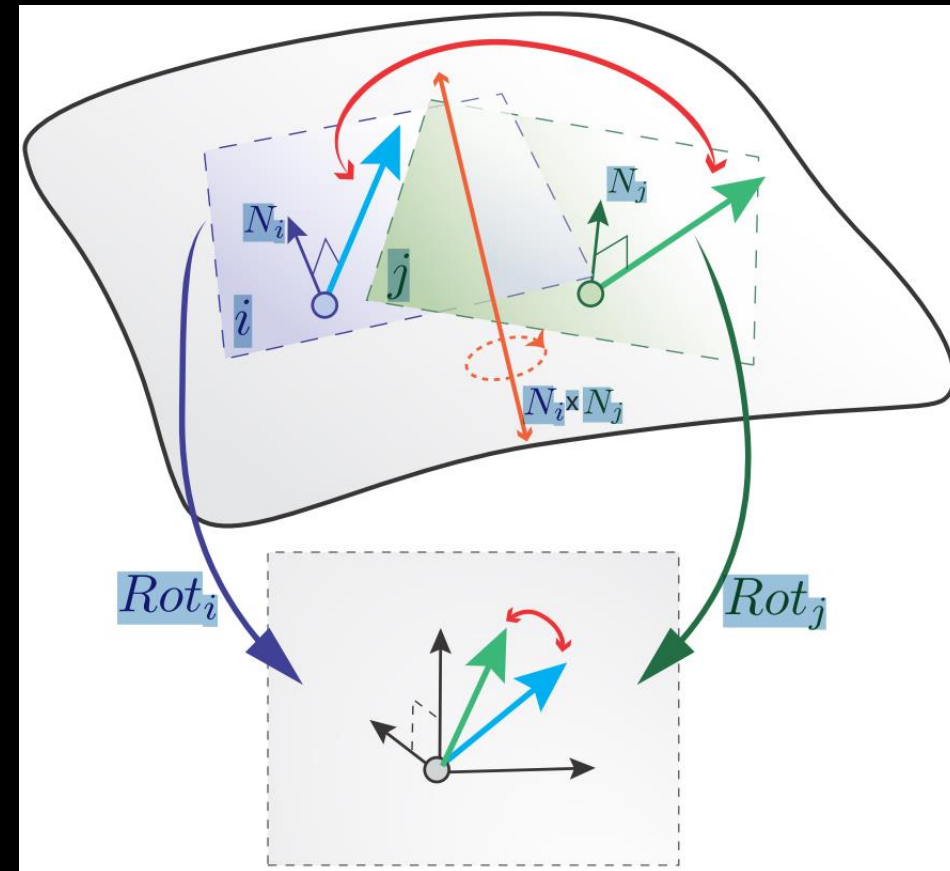
- Introduction
- **Discretization**
- Representation
- Objectives and Constraints

Tangent Spaces

- The tangent spaces of the a triangle mesh can be located on the faces, edges, or vertices of a triangle mesh.
- One way to construct a tangent space at a point is to assign a **surface normal vector** to the point.
 - Face: normal vector
 - Vertex and edge: local average of the adjacent triangle normal vectors
- Local coordinate system
 - Two orthogonal tangent vectors

Discrete Connections

- Given two adjacent tangent spaces i and j , we need a notion of **connection** between them in order to **compare** two directional objects that are defined on them.
- A straightforward discretization of the Levi-Civita connection is made by “flattening” the two adjacent tangent planes.
- X_{ij} : angle difference between corresponding axes



Vector Field Topology – continuous

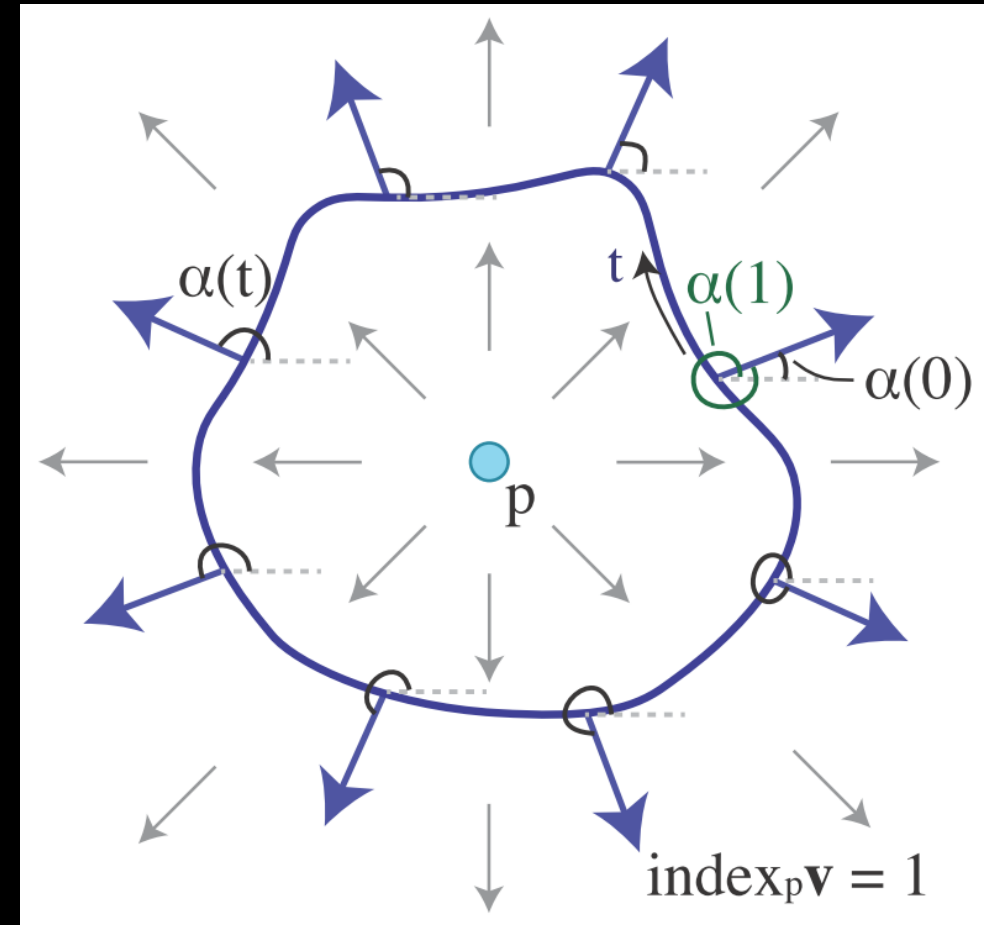
- A vector field has a singularity at a point p if **it vanishes or is not defined at this point.**
- 2D case:
 - Parameterized curve: $c: [0,1] \rightarrow R^2$
 - A smooth angle function: $\alpha: [0,1] \rightarrow R$
 - Vector field:

$$\mathbf{v}(c(t)) = \|\mathbf{v}(c(t))\| \begin{pmatrix} \cos(\alpha(t)) \\ \sin(\alpha(t)) \end{pmatrix}$$

- Define the index (an integer) of the singularity at p :

$$\text{index}_p = \frac{1}{2\pi} (\alpha(1) - \alpha(0))$$

Note: Since α is smooth, the difference $\alpha(1) - \alpha(0)$ is unique and it is a multiple of 2π .

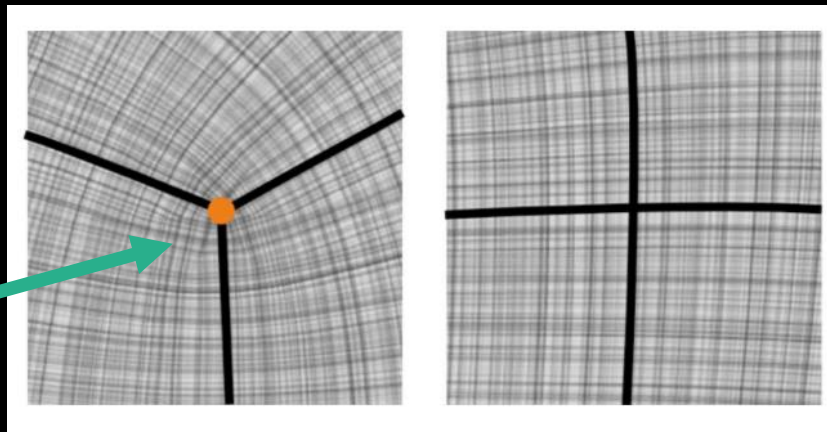


Singularity

$$\text{index}_p = \frac{1}{2\pi} (\alpha(1) - \alpha(0))$$

- The index measures **the number of times** the vectors along the curve c rotate counterclockwise, while traversing the curve once.
- It is common to consider only points p with index $\text{index}_p \neq 0$ as singular.
 - It vanishes or is not defined at this point.

Singularity



Singularity

- The definition does not directly extend to surfaces, because there is **no global coordinate system** (the tangent bundle is not trivial).
- Calculate the index at a point p of a vector field v on a surface M by using an arbitrary **chart** around p .
- A chart for a topological space M (also called a coordinate chart, coordinate patch, coordinate map, or local frame) is a **homeomorphism** from an open subset of M to an open subset of a **Euclidean** space.

Singularity

- Poincaré–Hopf theorem: the sum of all the indices of a vector field equals $2 - 2g$ for a surface without boundary.
- For N -vector fields, the index is a multiple of $1/N$. Some examples:

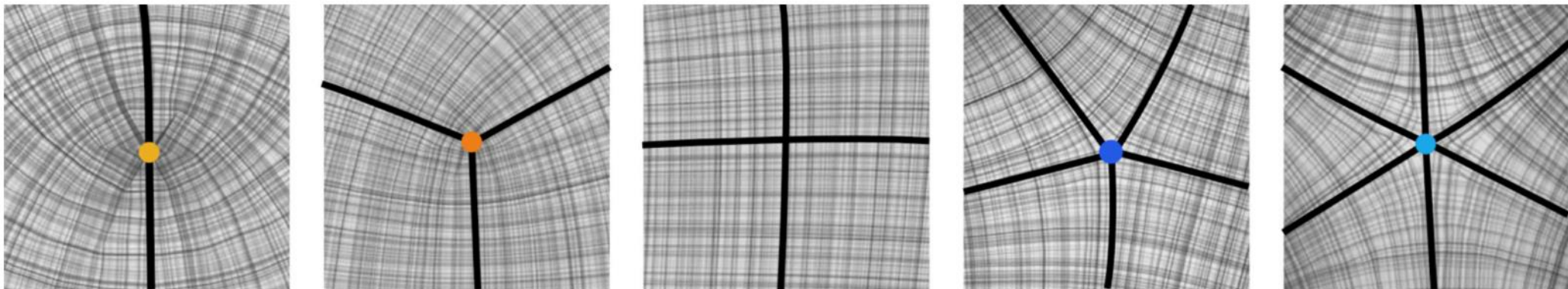
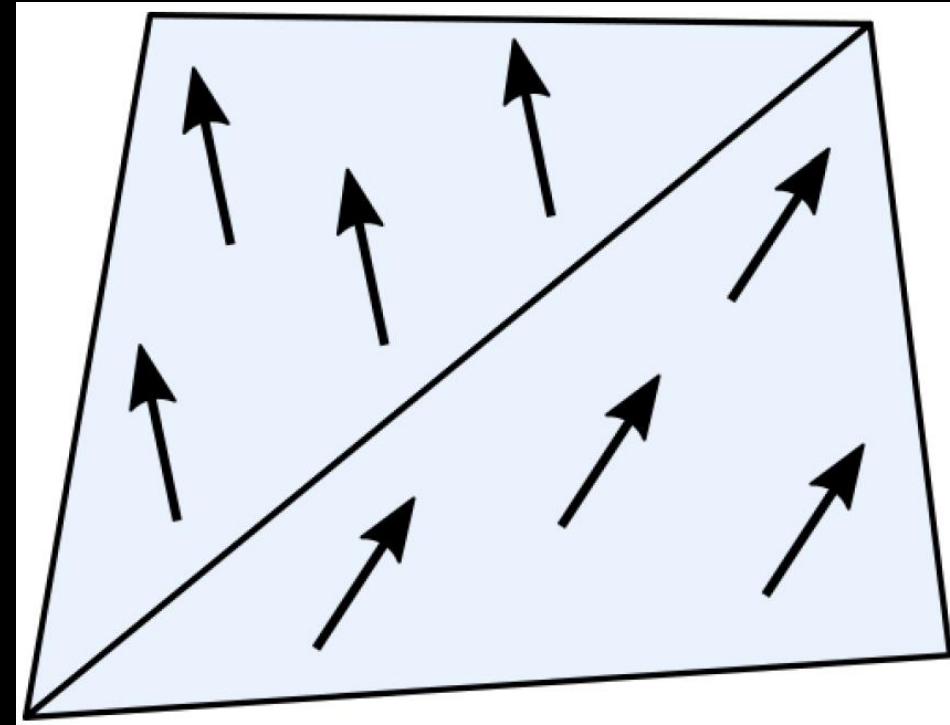


Figure 2: Singularities of index $\frac{1}{2}$, $\frac{1}{4}$, 0 (non-singular), $-\frac{1}{4}$, $-\frac{1}{2}$ in a 4-vector field. The black curves are the so-called separatrices – integral curves (cf. Section 10.3) of the field intersecting the singularity.

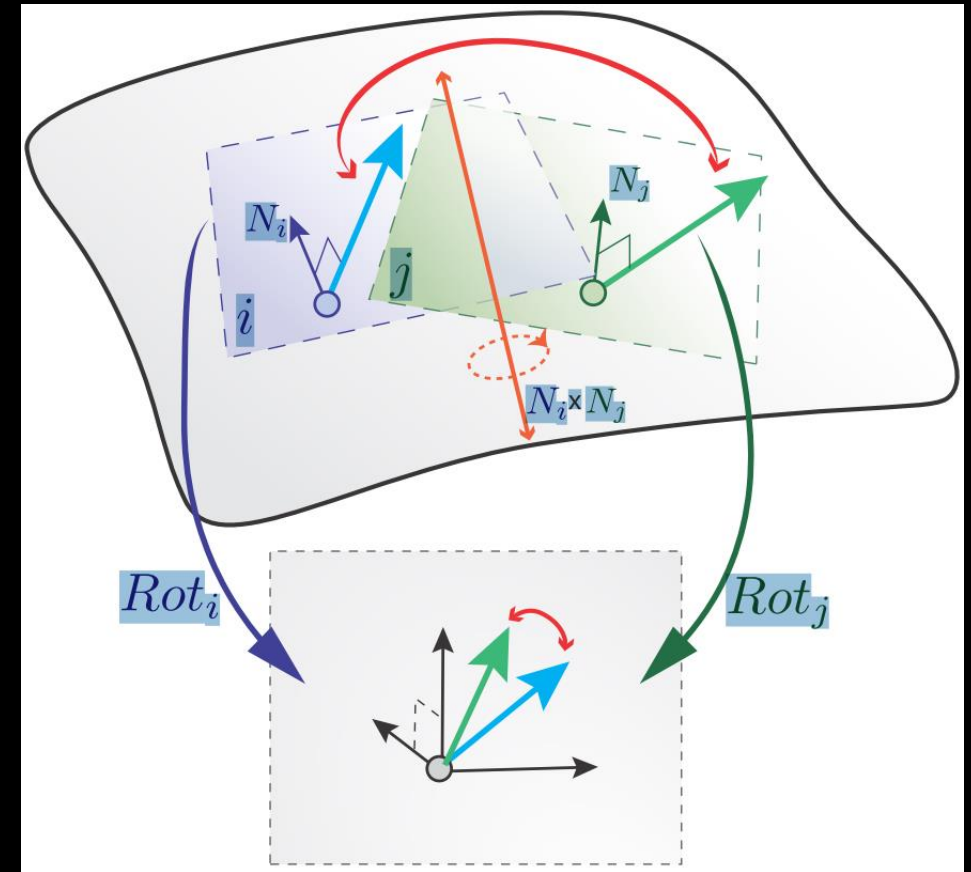
Discrete Field Topology

- Piecewise constant face-based **1-direction** field
 - Constant on face
 - Discontinuous on edges
- Extension from continuous setting
 - Define rotation between adjacent triangles to define angle difference
 - It is intuitive to assume that the field undergoes a **rotation** $\delta_{ij} = \frac{\pi}{4}$ clockwise.
 - $\delta_{ij} = \frac{\pi}{4} + 2\pi k$ would be a valid assumption.



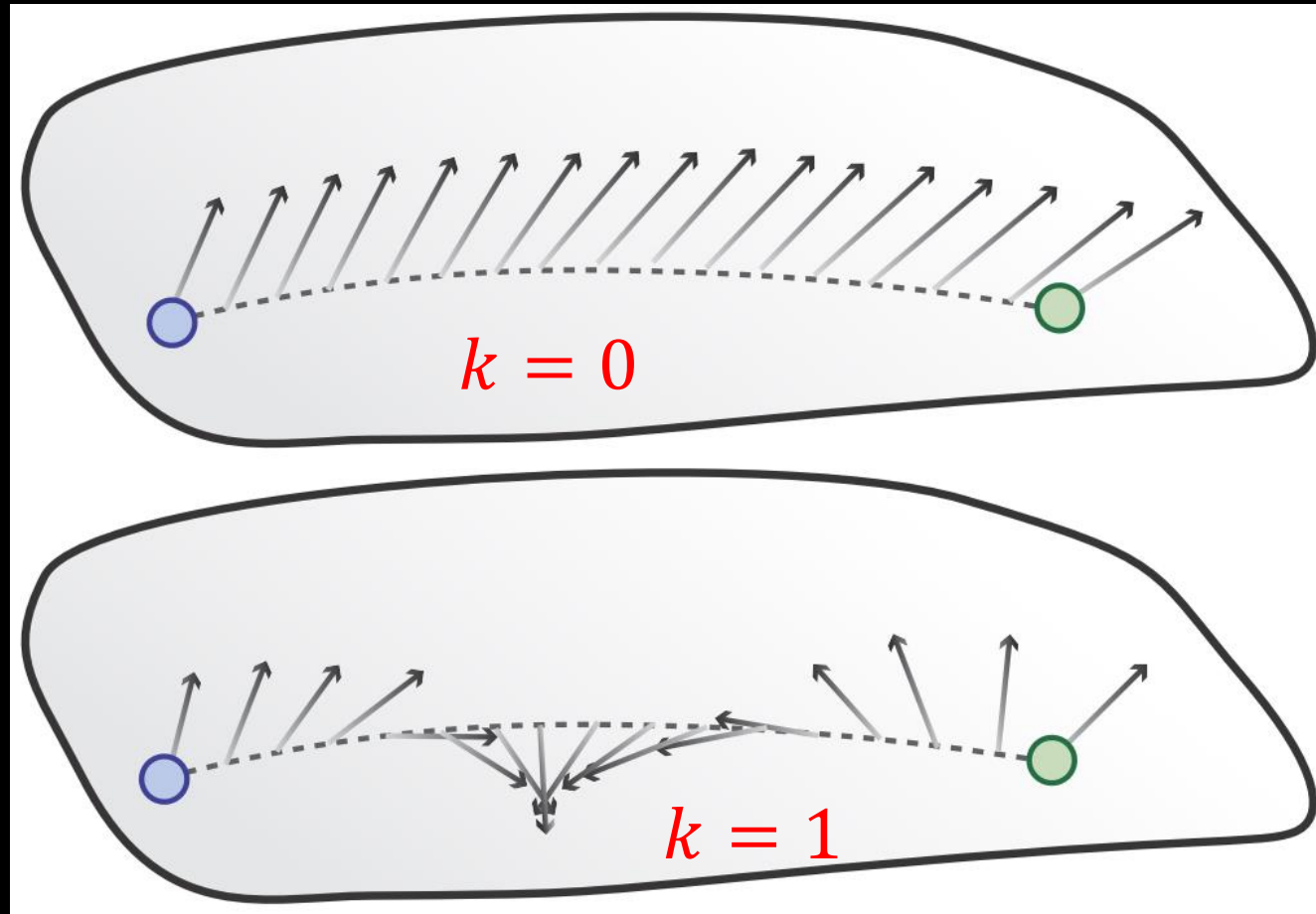
Rotation

- Principal rotation
 - $\delta_{ij} \in [-\pi, \pi)$
- Summarize all rotation angles on edges that are incident to the vertex
 - $\text{index}_p = \frac{1}{2\pi} \sum \delta_{ij}$



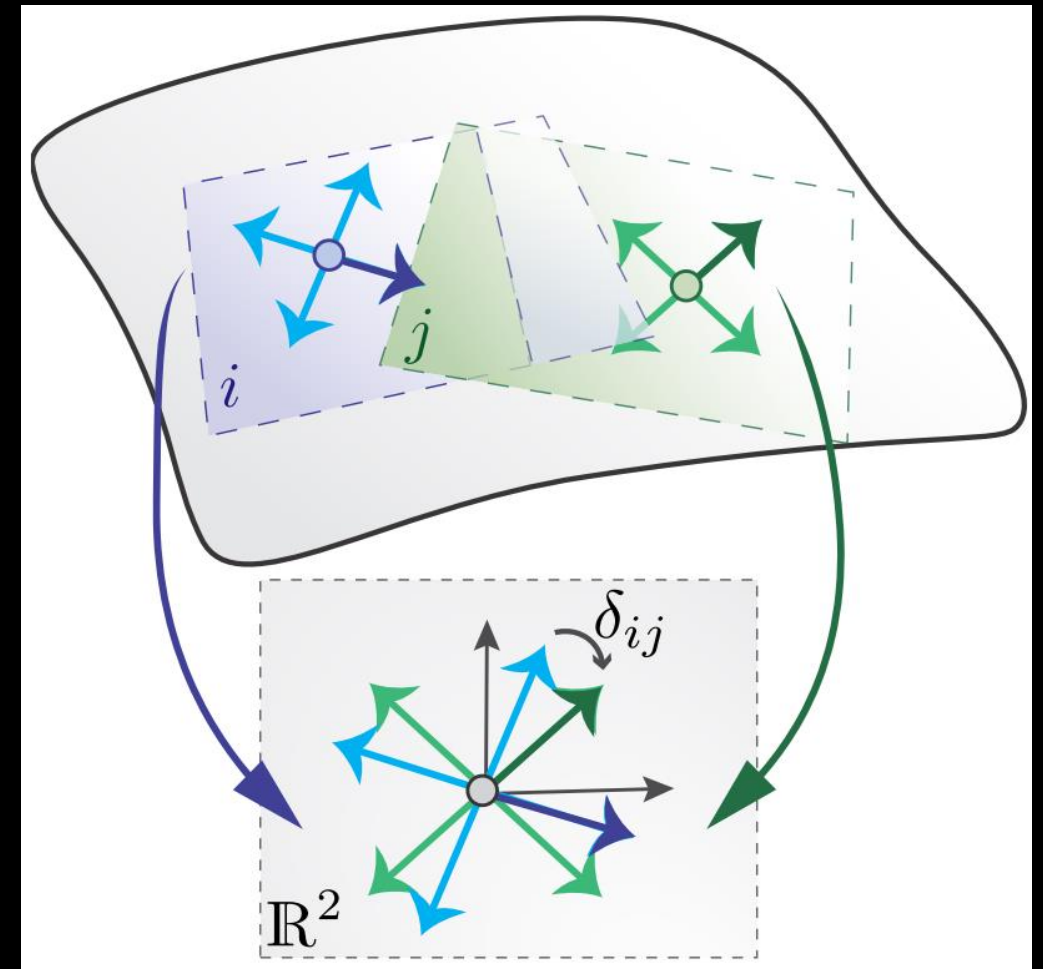
Period Jumps

- Non-principal rotation:
 - $\delta_{ij} + 2\pi k$
 - k full period rotations
- The values of k are denoted as *period jumps*.



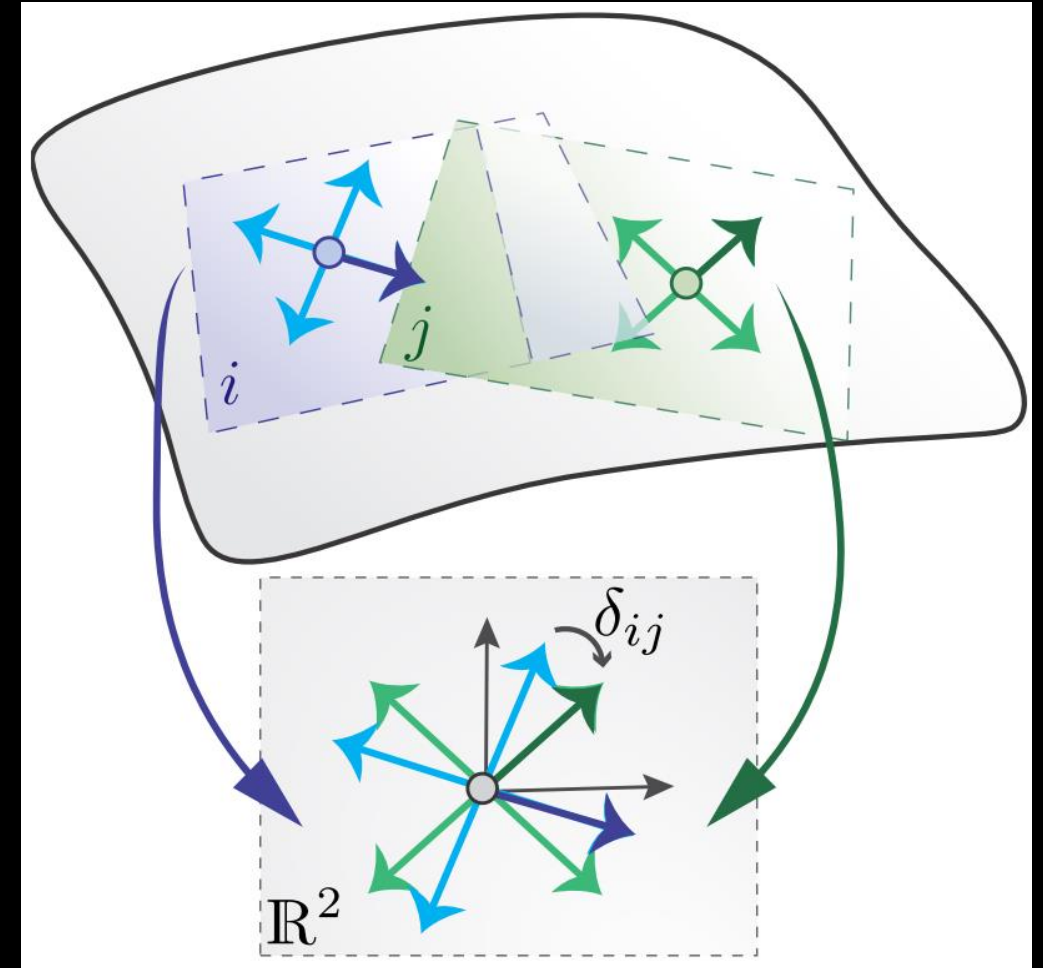
Matching - multi-valued field

- $N > 1$ directionals per tangent space
- An additional degree of freedom:
 - The **correspondence** between the individual directionals in tangent space i to those in the adjacent tangent space j .
- A **matching** between two N -sets of directional is a **bijjective** map f between them (or their indices).
 - It preserves order: $f(u_r) = v_s \Leftrightarrow f(u_{r+1}) = v_{s+1}$



Effort

- Based on a matching f , the notions of rotation and principal rotation can be generalized to multi-valued fields.
- δ_{ij}^r : rotation between u_r and $f(u_r)$
- **Effort** of the matching f : $Y_{ij} = \sum_{r=1}^N \delta_{ij}^r$
- Symmetric N -directional field
 - $\delta_{ij} = \delta_{ij}^r$ for every r
- The efforts of different (order-preserving) matchings differ by 2π .

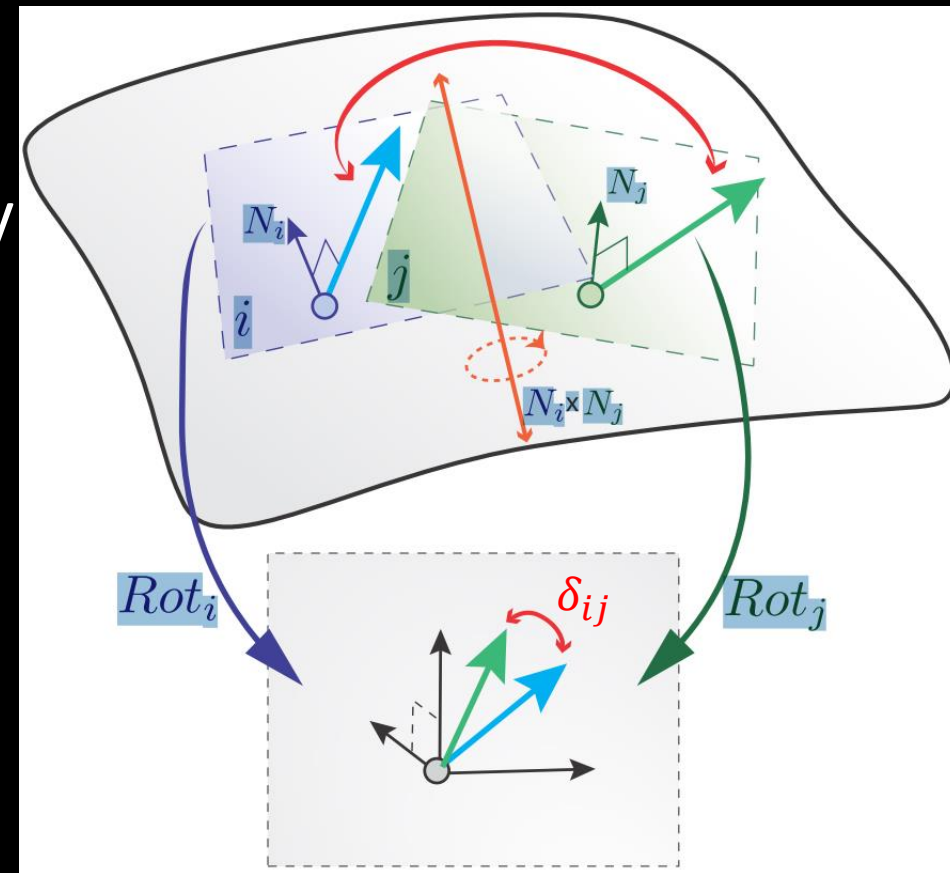


Outlines

- Introduction
- Discretization
- **Representation**
- Objectives and Constraints

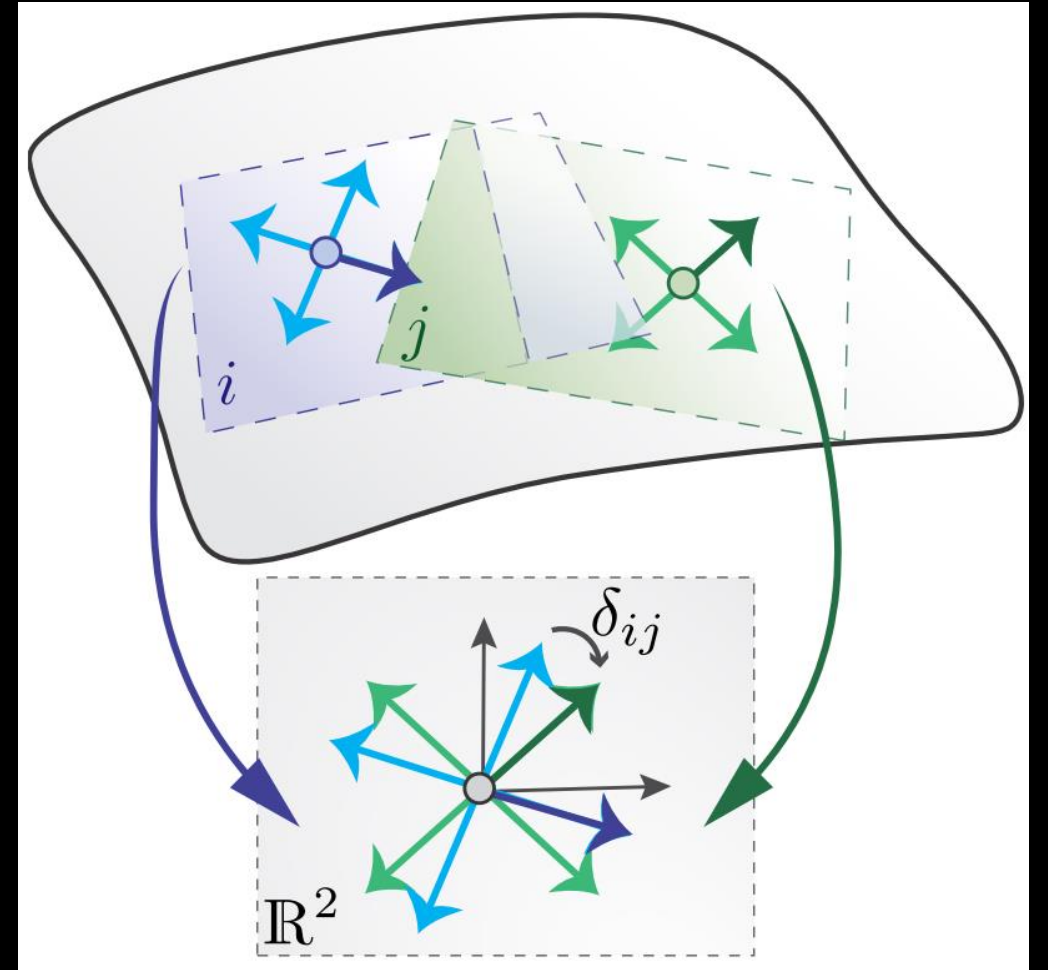
Angle-Based

- Local thonormal frame $\{e_1, e_2\}$ on each tangent space
- 1-direction (unit vector) fields can be concisely described within each tangent space by a **signed angle φ** that is relative to e_1 .
- Rotation angle:
 - $\delta_{ij} = \phi_j - (\phi_i + X_{ij} + 2\pi k_{ij})$
 - X_{ij} : the change of bases e between the flattened tangent spaces i and j
 - k_{ij} : period jump



Angle-Based N symmetry directions

- A single φ representing the set of N symmetry directions.
 - $\{\phi + l \cdot 2\pi/N | l \in \{0, \dots, N - 1\}\}$
- Period jump to be an integer multiple of $\frac{1}{N}$.
- Rotation angle:
 - $\delta_{ij} = \phi_j - (\phi_i + X_{ij} + \frac{2\pi}{N} k_{ij})$



Pros and cons

- Advantage

- Directions, as well as possible period jumps, are represented explicitly.
- A linear expression of the rotation angle.

- Disadvantage

- The use of integer variables, which leads to discrete optimization problems.

Cartesian and Complex

- A vector v in a two-dimensional tangent space can be represented using **Cartesian** coordinates (from R^2) in the local coordinate system $\{e_1, e_2\}$, or equivalently as **complex** numbers (from C).

- Connection to angle-based representation

$$v = \begin{pmatrix} \cos \phi \\ \sin \phi \end{pmatrix} = e^{i\phi}$$

- The change of bases from one tangent space to another

$$\begin{pmatrix} \cos X_{ij} & -\sin X_{ij} \\ \sin X_{ij} & \cos X_{ij} \end{pmatrix} \text{ or } e^{iX_{ij}}$$

N -directional fields

- By multiplying the argument of the trigonometric functions, or taking the complex exponential to the power of N

$$v^N = \begin{pmatrix} \cos N\phi \\ \sin N\phi \end{pmatrix} = e^{iN\phi}$$

- $e^{iN\phi_l} = e^{iN\phi}$

- $\phi_l = \phi + l \cdot \frac{2\pi}{N}, \forall l \in \{0, \dots, N-1\}$

- v^N becomes a 1-directional field.

- X_{ij} becomes NX_{ij} :

$$\begin{pmatrix} \cos NX_{ij} & -\sin NX_{ij} \\ \sin NX_{ij} & \cos NX_{ij} \end{pmatrix} \text{ or } e^{iNX_{ij}}$$

Complex Polynomials

- Analogously, every N -vector set $\{u_1, \dots, u_N\}$, in the complex form $u_i \in \mathbb{C}$, can be uniquely identified as **the roots of a complex polynomial** $p(z) = (z - u_1) \dots (z - u_N)$.
- Writing p in monomial form, $p(z) = \sum_i c_n z^n$, **the coefficient set $\{c_n\}$** is thus an order-invariant representative of a 1 N -vector.
 - N -PolyVector
 - A generation of former representation.

Comparison

- Comparing between PolyVectors on adjacent tangent spaces amounts to **comparing polynomial coefficient**.
- Every coefficient c_n contains multiplications of $N - n$ roots.

Tensors

- Real-valued 2×2 matrices in local coordinates

$$T = \begin{pmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{pmatrix}$$

- Symmetric Tensors

- an eigen-decomposition $T = U\Lambda U^T$
- $\Lambda = \text{diag}(\lambda_1, \lambda_2)$, two real eigenvalues
- $U = [u_1, u_2]$, two (orthogonal) eigenvectors with $\|u_i\| = 1$
- Since eigenvectors are only determined up to sign, a rank-2 tensor field can in fact be interpreted as two orthogonal **2-direction fields** $\pm u_i$.

Outlines

- Introduction
- Discretization
- Representation
- Objectives and Constraints

Objectives & Constraints

- Different applications have different requirements.
 - various objectives can be used for vector field optimization

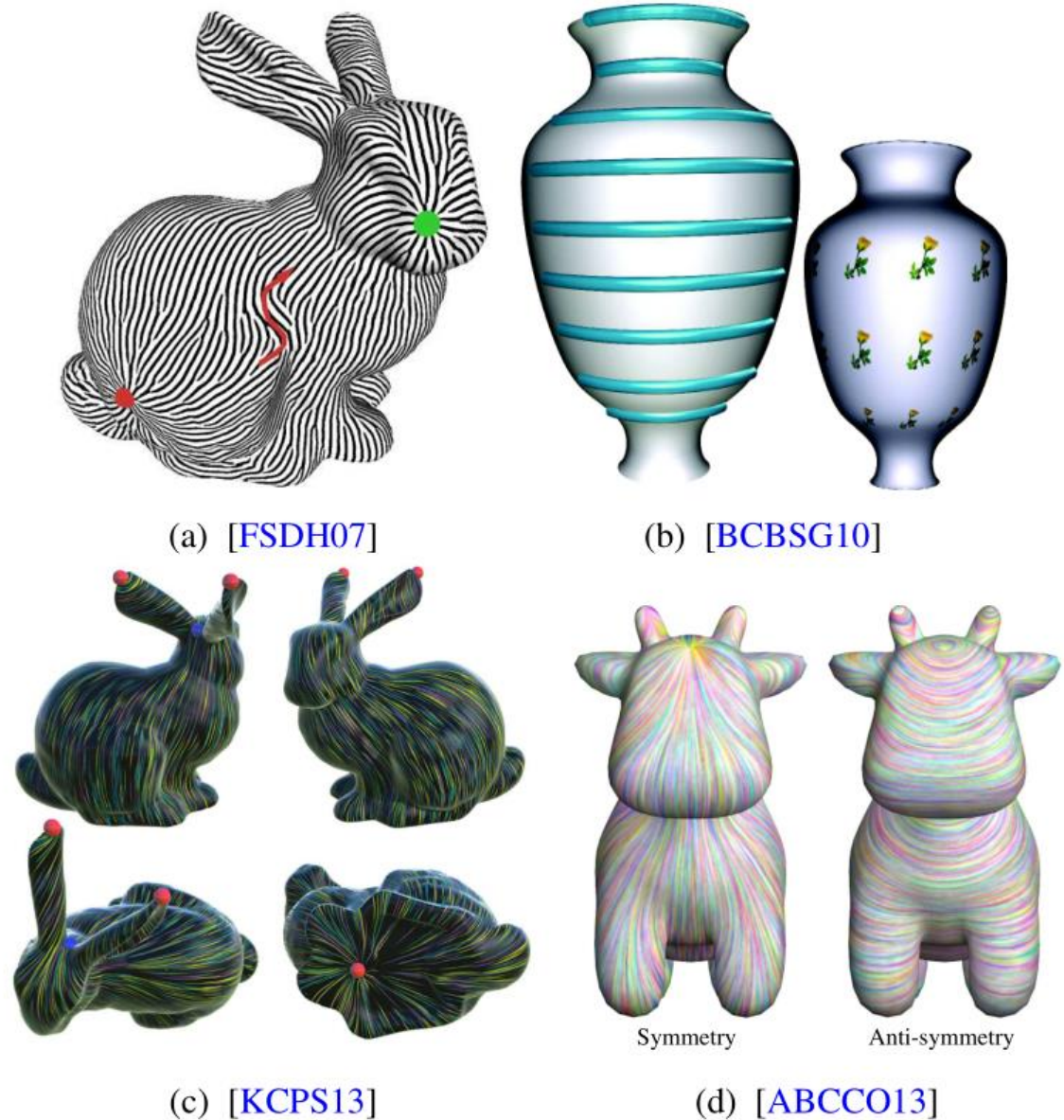


Figure 5: Various objectives used for vector field optimization - (a) alignment to constraints, (b) Killing energy for isometric on-surface deformations, (c) smoothness (Dirichlet energy) (d) commutativity with the symmetry/antisymmetry self maps.

Objectives

- Fairness
 - measuring how variable, or rather, non-similar, the field is between adjacent tangent spaces.
- Parallelity
- Orthogonality
- Minimization of curl
-

Parallelity – as-parallel-as-possible

- Parallel
 - The direction in one tangent space is obtained via parallel transport from the directions in adjacent tangent spaces.

- As-parallel-as-possible goal:

$$E_{fair-N} = \frac{N}{2} \sum_e w_e (\delta_e)^2$$

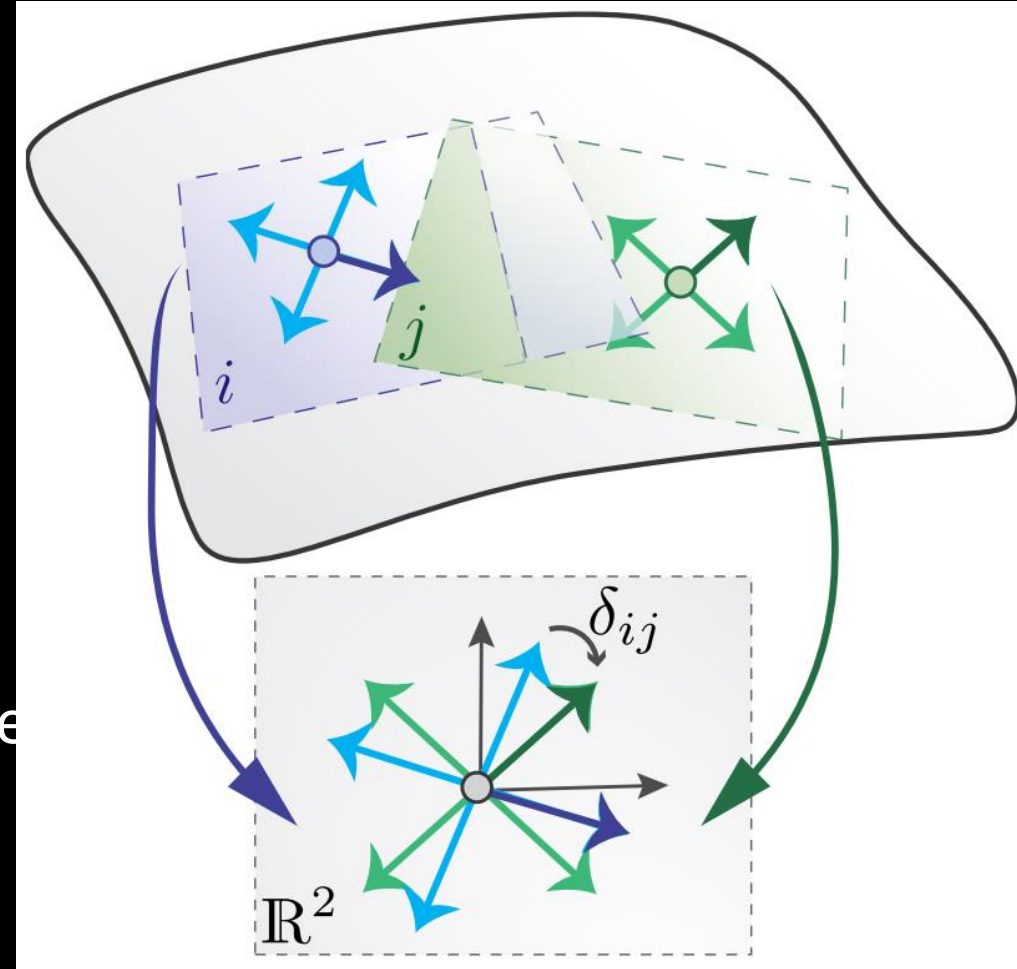
An example of cross field

$$E_{fair-4} = \frac{4}{2} \sum_e w_e \left(\phi_j - \left(\phi_i + X_{ij} + \frac{2\pi}{N} k_{ij} \right) \right)^2$$

Variables: ϕ_i and k_{ij} (integers)

Greedy solver:

1. Treat k_{ij} as floating number, minimize E_{fair-4}
2. Round the variable which causes the smallest absolute error if we round it to the nearest integer
3. Repeat above two steps.

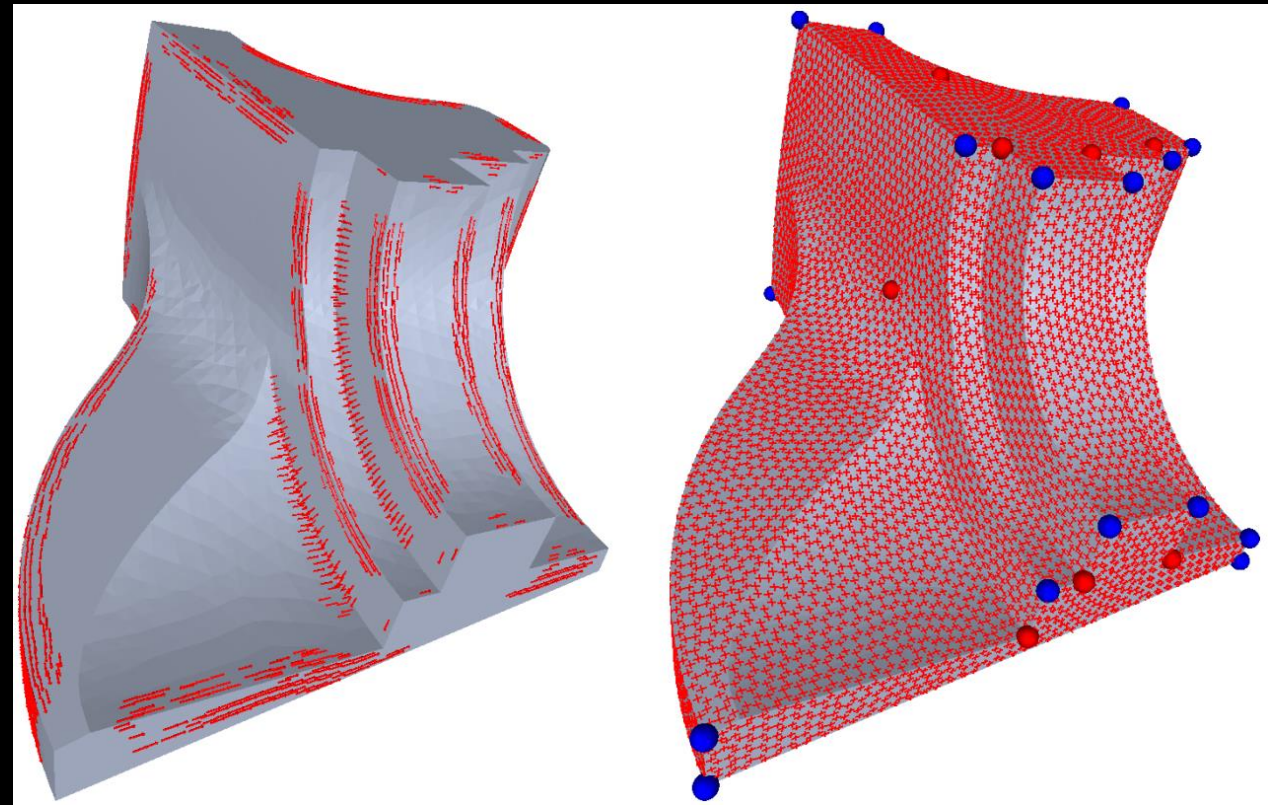


Constraints

- Alignment
 - fit certain prescribed directions
 - principal curvature
 - strokes given by an artist on the surface
 - boundary curves
 - feature lines
 - Soft data term
 - Least square
 - Hard constraint

Constraints

- Alignment
 - fit certain prescribed directions
 - principal curvature
 - strokes given by an artist on the surface
 - boundary curves
 - feature lines
 - Soft data term
 - Least square
 - Hard constraint



Constraints

- Symmetry:
 - If the **surface has bilateral symmetry**, it is advantageous if the designed directional fields adhere to **the same symmetry**, allowing field-guided applications to **preserve the symmetry as well**.

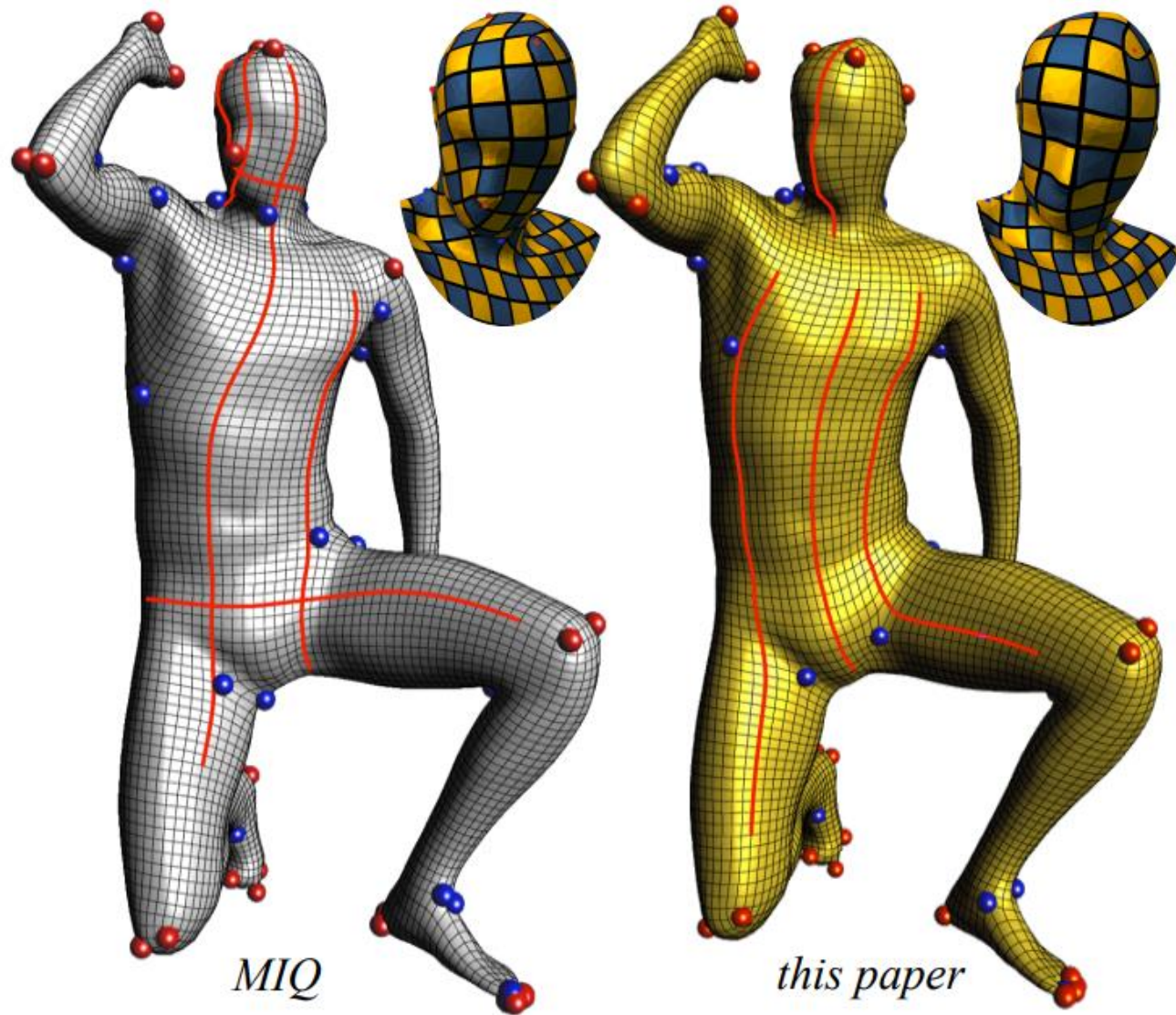


Figure 1: Field-aligned parametrization of the knelt human model using the symmetry field construction method developed in this paper, and using the MIQ technique of Bommes et. al.[2009]. Red/blue bullets represent field singularities with positive/negative index. Red lines trace flows of the cross field.

Constraints

- Surface mapping:
 - Given multiple shapes with a **correspondence** between them, we could require that the directional fields **commute with the correspondence**, effectively designing directional fields jointly on multiple shapes.

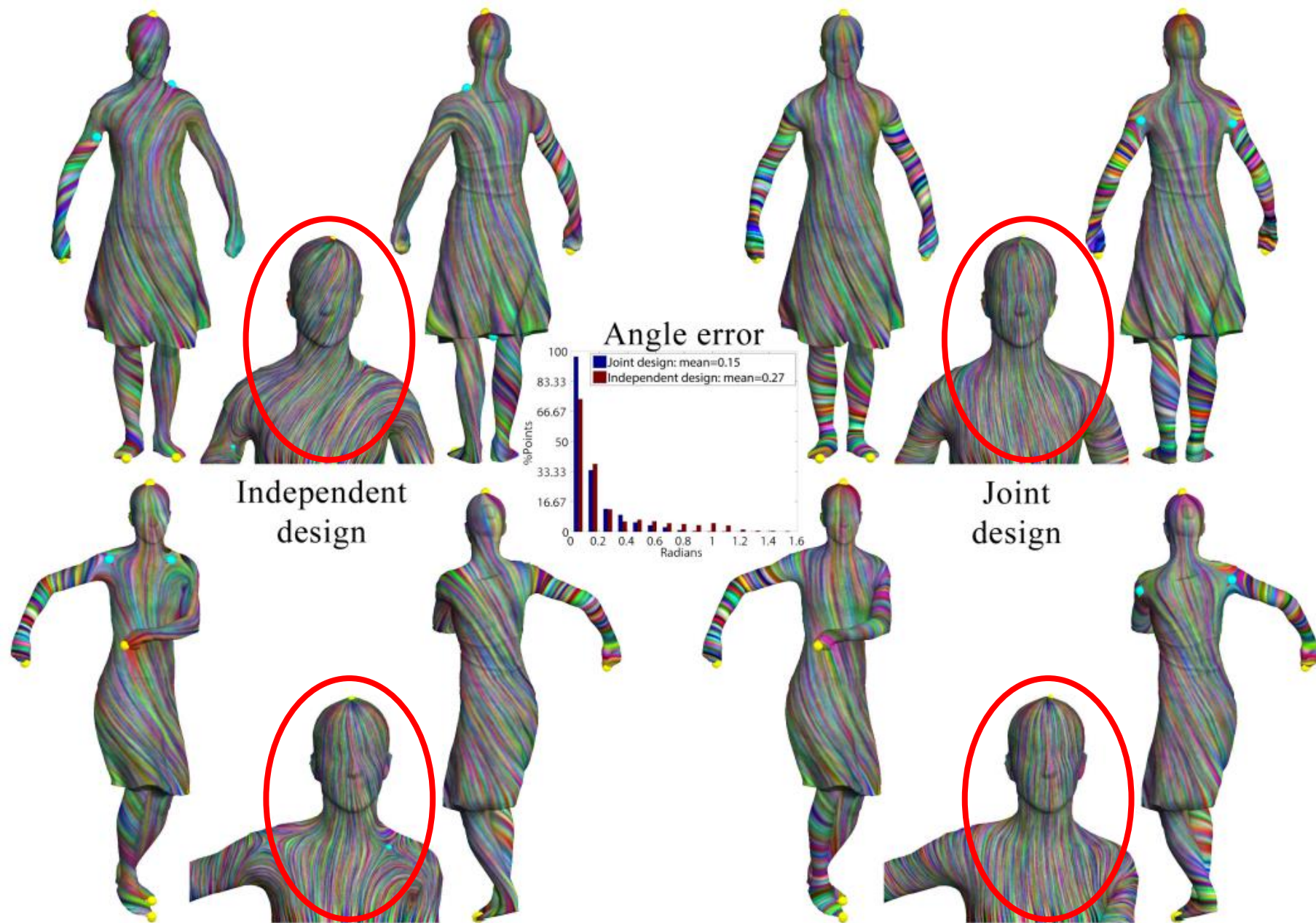


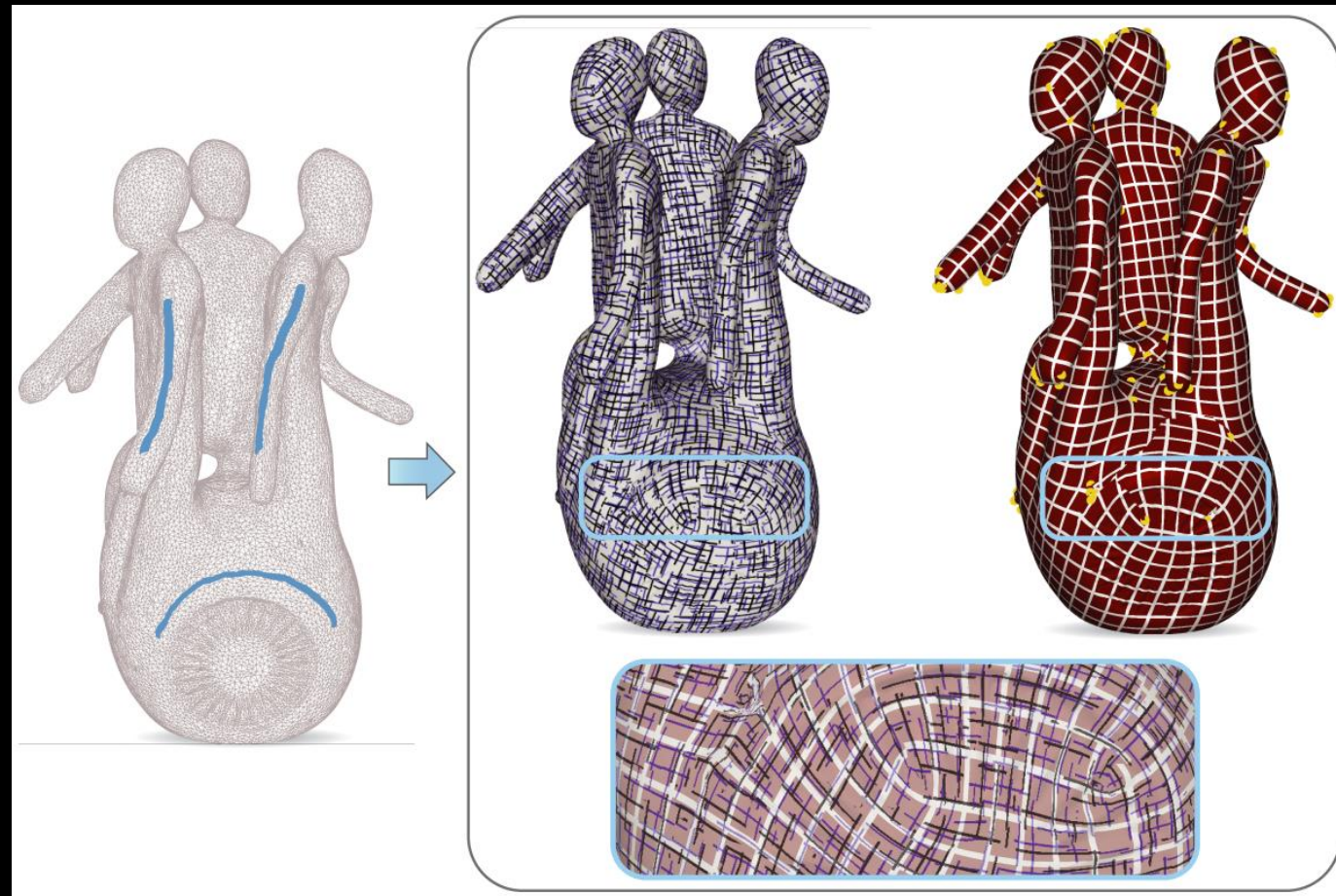
Figure 12: (left) Independent design on two shapes which are in correspondence does not yield a consistent vector field, even if compatible constraints are used. (right) Solving jointly using our framework yields consistent vector fields (note the corresponding locations of the singularities on the back of the shape). See the text for details.

Integrable field

- In most of these applications, vector fields are computed to serve as a guiding basis for the construction of global parameterizations.
- Parameterization coordinates:
 - Defined on vertices
 - Two scalar variables
- Gradients of parameterization coordinates are two separate vector fields.
 - Integrable
 - Curl-free

Integrable field

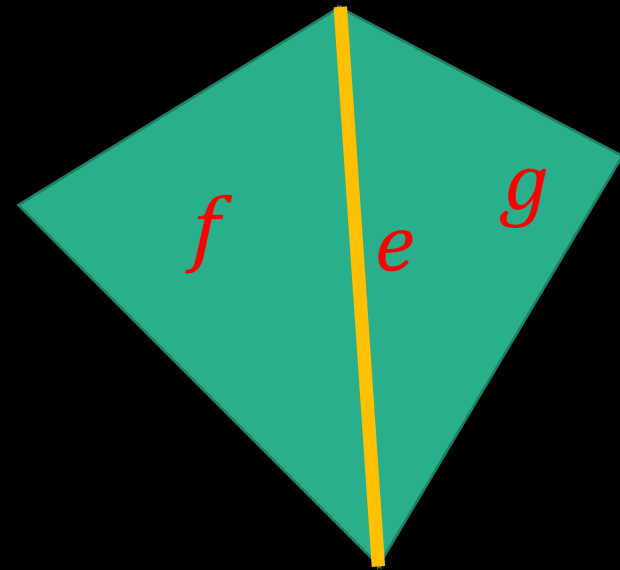
- Thinking from a opposite way:
 - If the vector fields are curl-free, it can be integrated to be parameterization coordinates.
 - Minimize the difference between the tangent field and the gradient of the function in the **least-squares** sense.
 - Thus, if the curl-free field is foldover-free and with low distortion, the parameterization is also foldover-free and with low distortion.



Integrability

- Scalar function $h: M \rightarrow R$
 $\langle \nabla h_f, e \rangle = \langle \nabla h_g, e \rangle$
- It trivially follows that $\langle \nabla h_f, e \rangle - \langle \nabla h_g, e \rangle$ is zero for any function h .
- Discrete curl for any vector field α :
 - $\langle \alpha_f, e \rangle - \langle \alpha_g, e \rangle$
 - α is curl-free if and only if $\langle \alpha_f, e \rangle = \langle \alpha_g, e \rangle$.

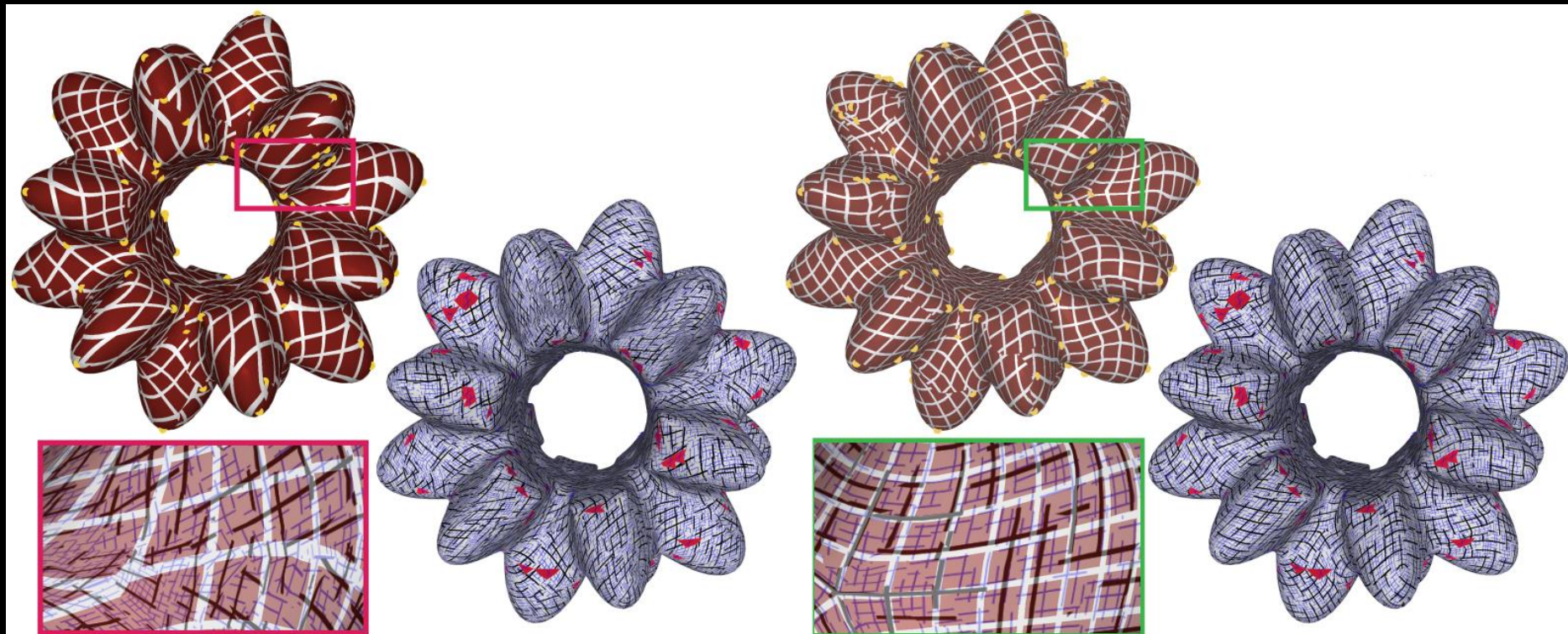
Circulation: $\oint_L v \cdot dr$



Vector field design

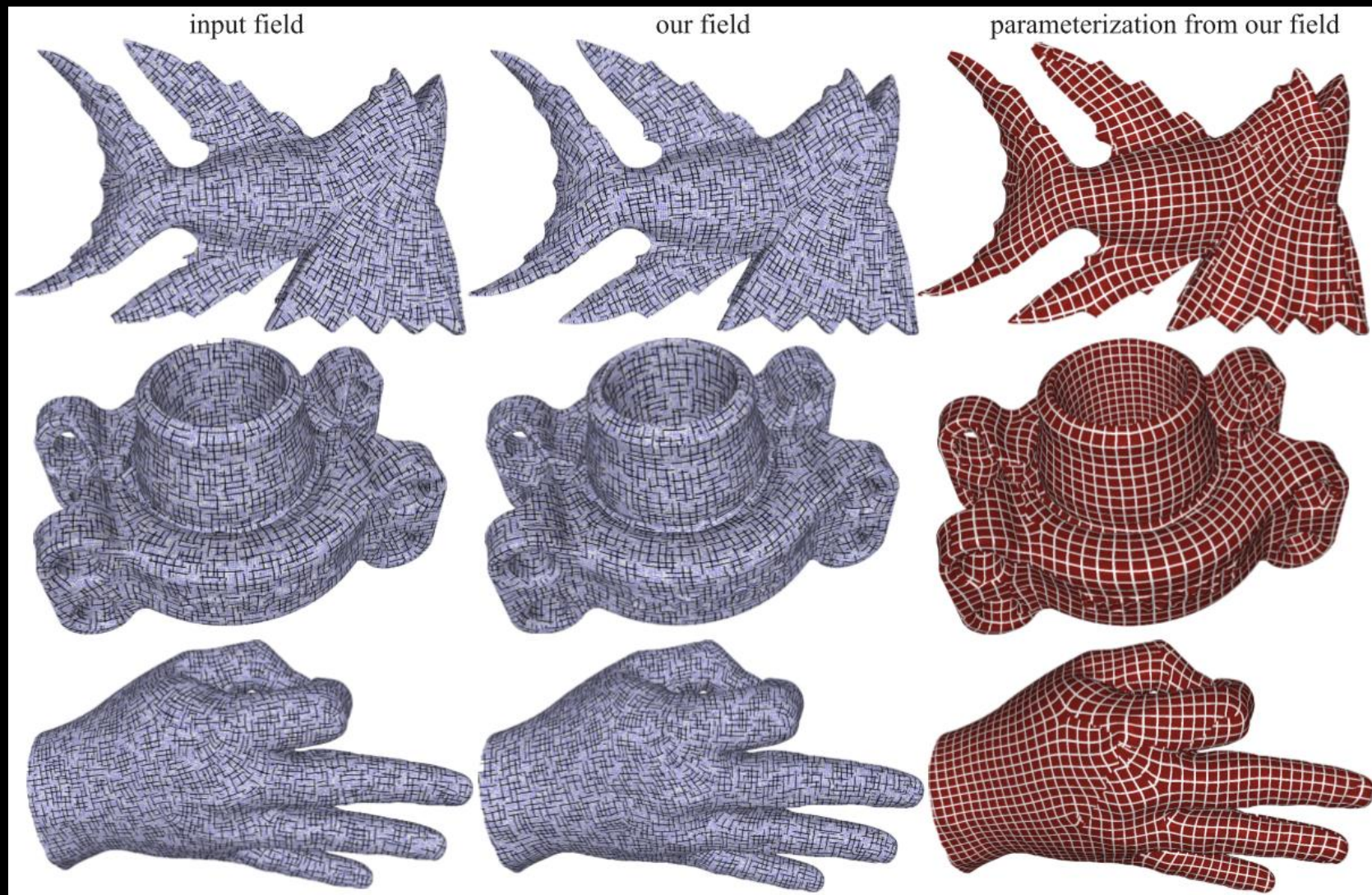
$$J = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix}, \alpha = \left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \right), \beta = \left(\frac{\partial v}{\partial x}, \frac{\partial v}{\partial y} \right)$$

- Objective:
 - Fairness
- Constraints:
 - Curl-free
 - Foldover-free
 - Low distortion



Poisson integration

$$\min_h \sum_f \|\nabla h_f - \alpha_f\|_2^2$$



Extension

- Another question:
 - **Given a cross field, how to modify it to be curl-free?**
- Paper: *Computing inversion-free mappings by simplex assembly*

$$E = E_C + \lambda E_{field}$$

$$E_{field} = \sum_e \left(\langle \alpha_f, e \rangle - \langle \alpha_g, e \rangle \right)^2 + \left(\langle \beta_f, e \rangle - \langle \beta_g, e \rangle \right)^2$$

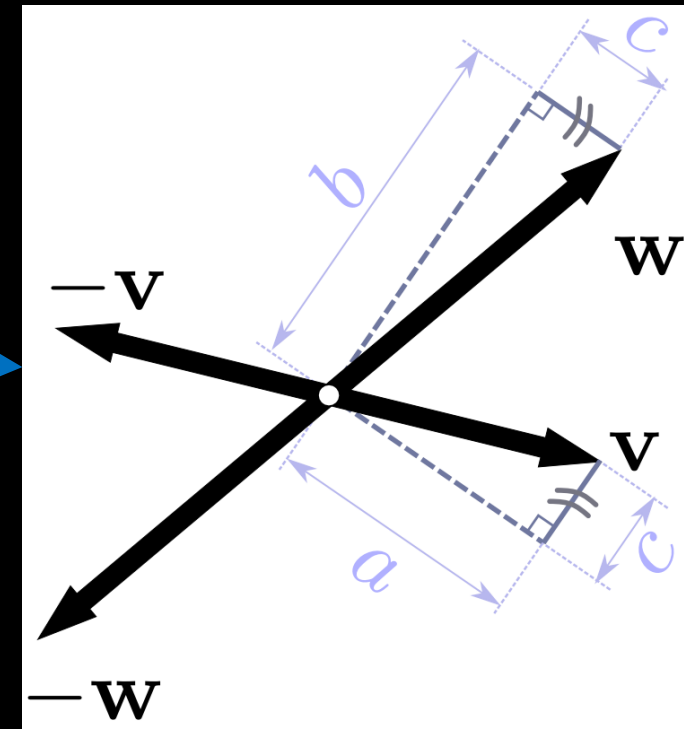
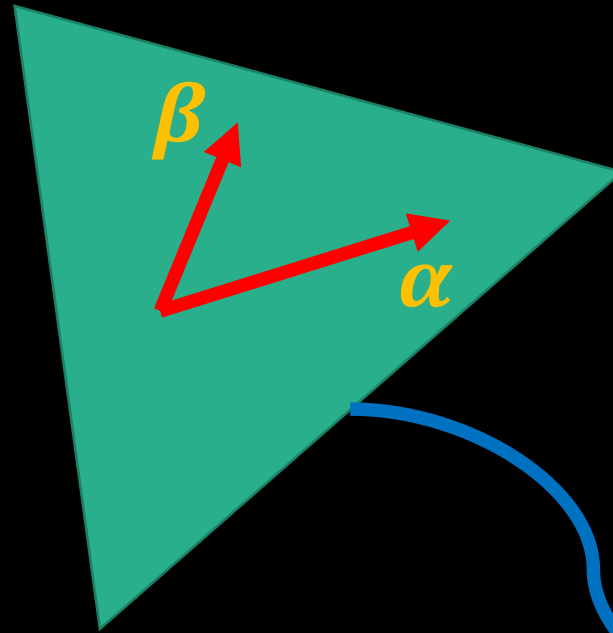
E_C : distortion energy

Increase λ to make E_{field} approach zero

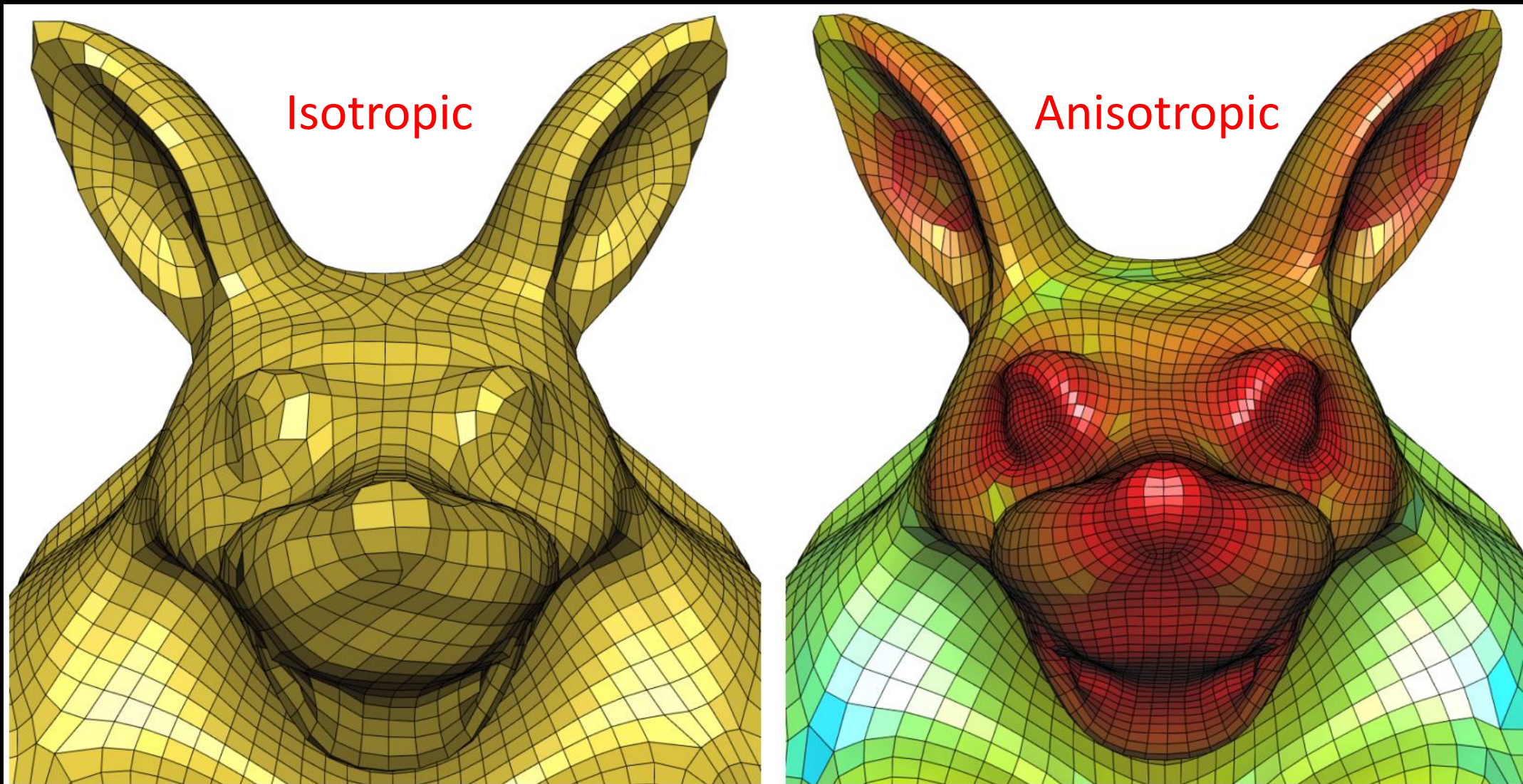
Properties of resulting field

- Non-Orthogonal
- Different lengths

- Frame field

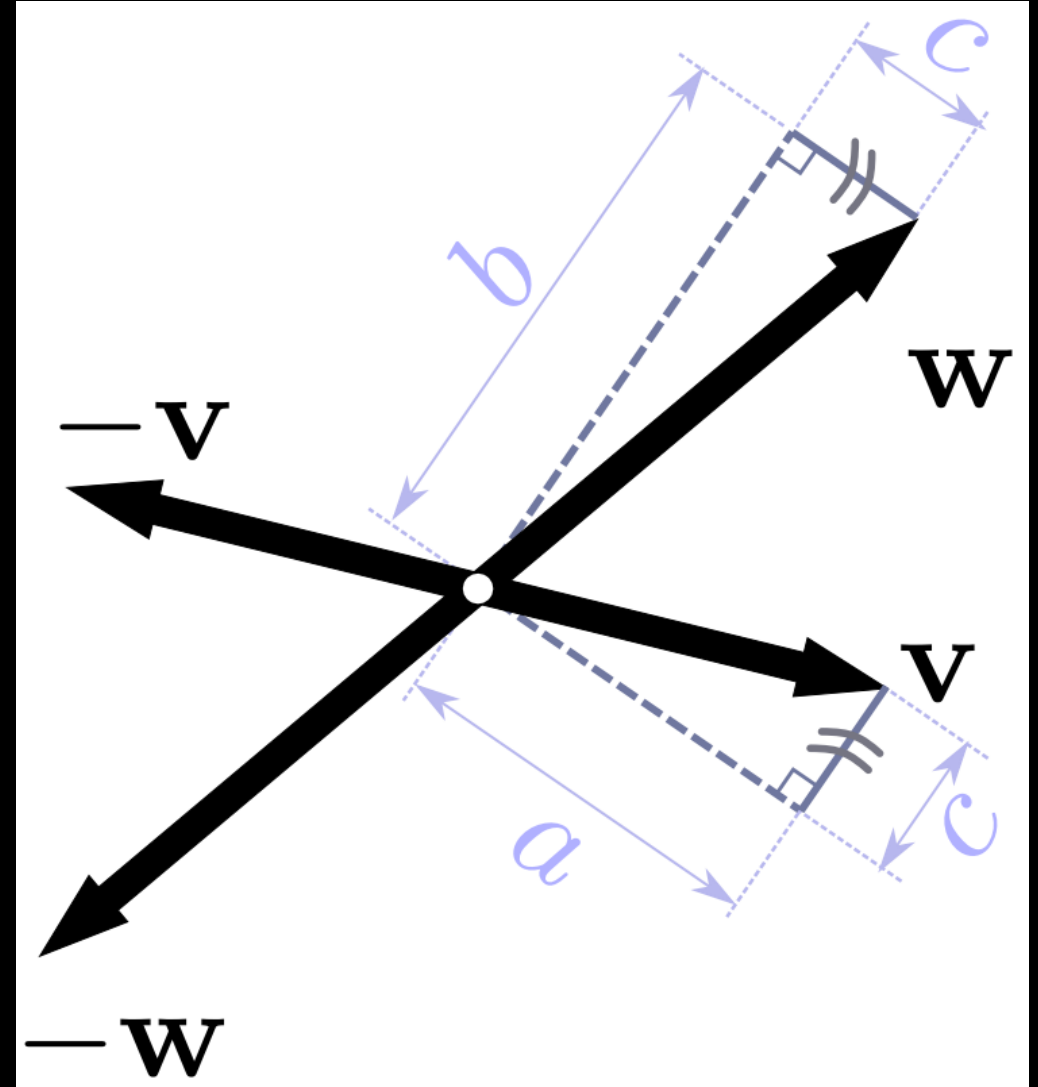


Frame Fields



Frame Fields

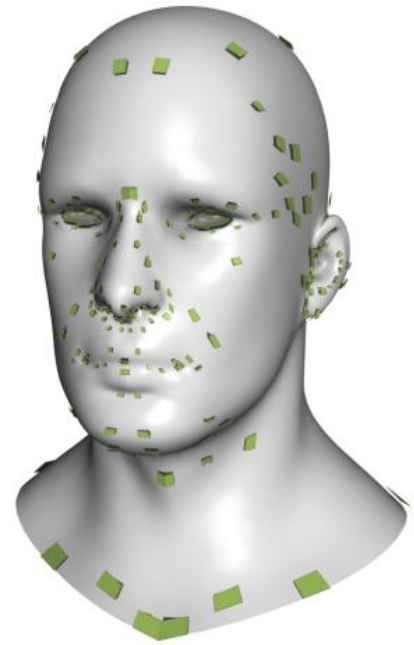
- Cross filed (4-RoSy field)
 - $X = \langle \mathbf{u}, \mathbf{u}^\perp, -\mathbf{u}, -\mathbf{u}^\perp \rangle$
- Frame field (2^2 vector field)
 - $F = \langle \mathbf{v}, \mathbf{w}, -\mathbf{v}, -\mathbf{w} \rangle$
- Canonical decomposition
 - $F = WX$ (polar decomposition)
 - W : symmetric positive definite matrix



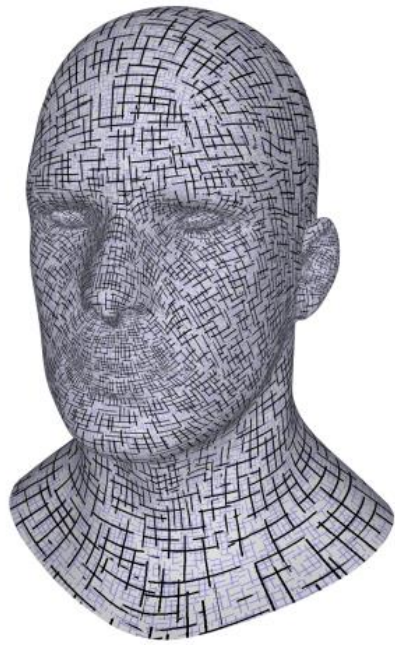
Paper: *Frame Fields: Anisotropic and Non-Orthogonal Cross Fields*

- A frame field is said to be continuous/smooth if both X and W are continuous/smooth.
- Synthesis of frame field:
 - Separately design X and W
 - X : formerly method, e.g., $E_{fair-4} = \frac{4}{2} \sum_e w_e \left(\phi_j - \left(\phi_i + X_{ij} + \frac{2\pi}{N} k_{ij} \right) \right)^2$
 - W : Laplacian smoothing, guarantee that the resulting W are SPD.

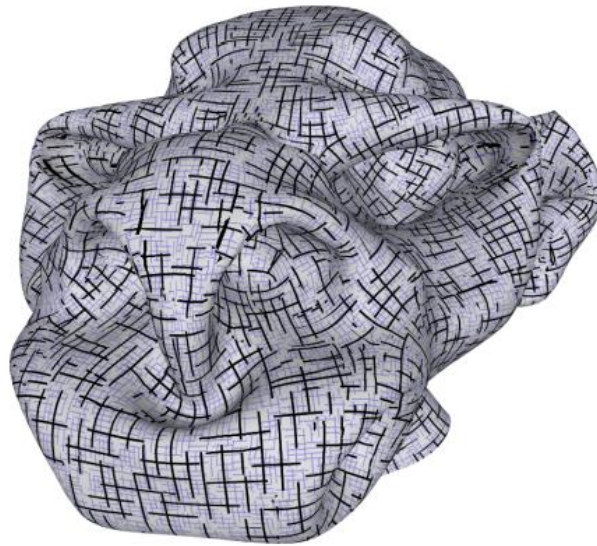
An example



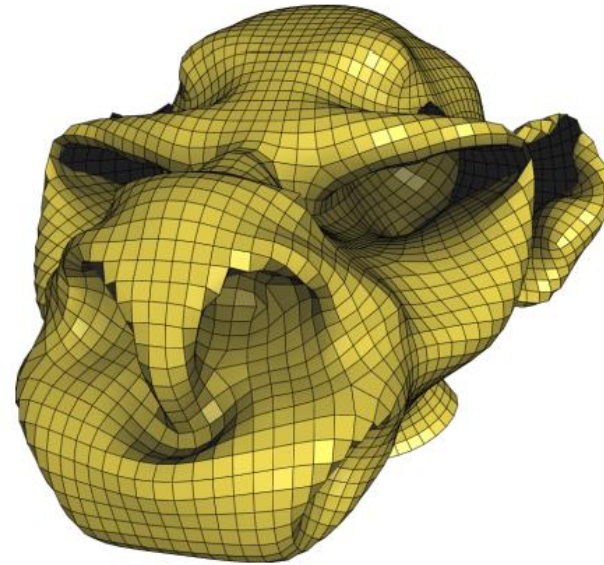
Constraints



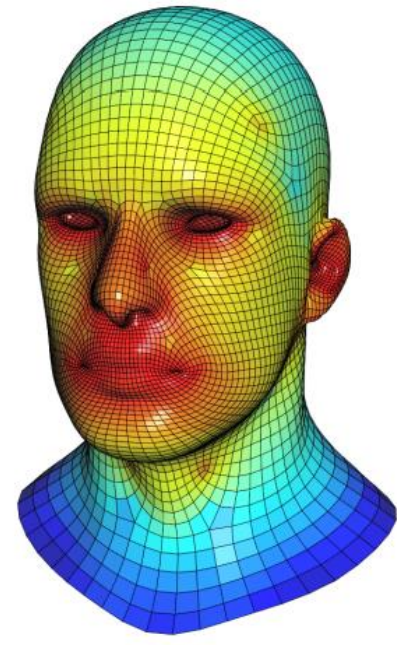
Frame field



Deformation



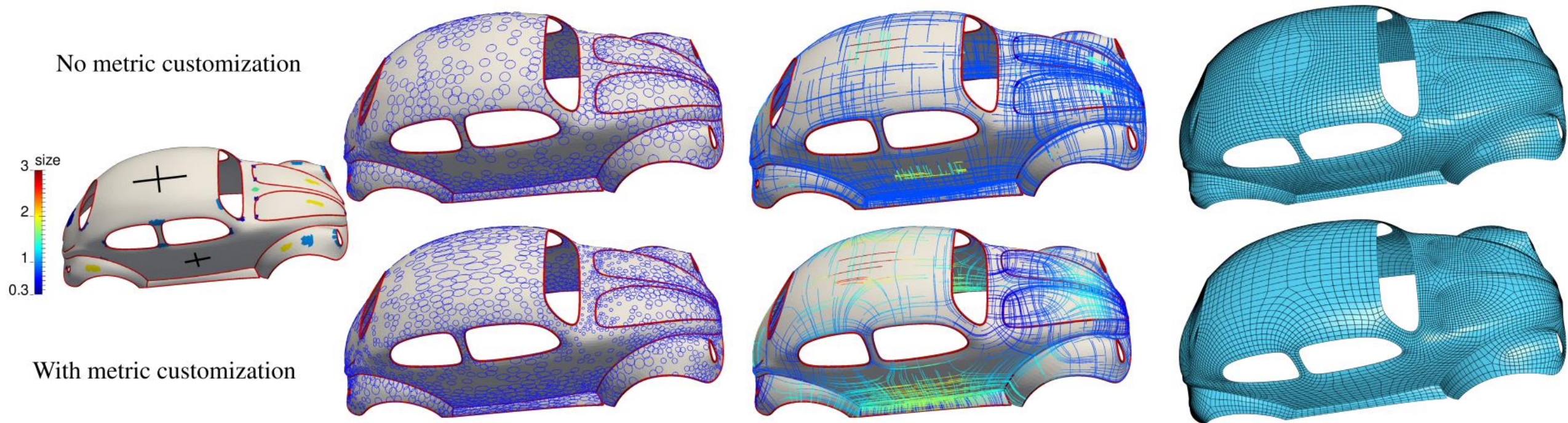
Isotropic



Anisotropic

Paper: *Frame Field Generation through Metric Customization*

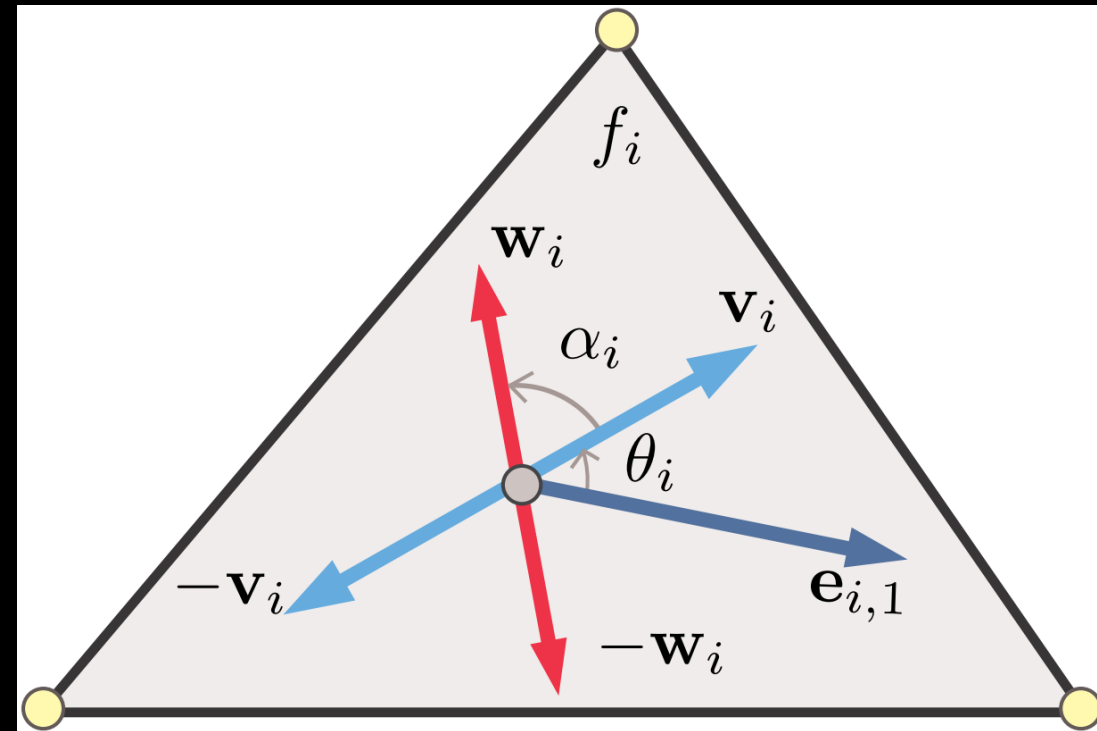
- Generic frame fields (with arbitrary anisotropy, orientation, and sizing) can be regarded as cross fields in a **specific Riemannian metric**.
 - First compute a **discrete metric** on the input surface.



2^2 directional field

Paper: General Planar Quadrilateral Mesh Design Using Conjugate Direction Field

- $F = \langle \mathbf{v}, \mathbf{w}, -\mathbf{v}, -\mathbf{w} \rangle$
 - $\|\mathbf{v}\| = \|\mathbf{w}\|$
- Equivalent classes using permutation
 - $\langle \mathbf{v}, \mathbf{w}, -\mathbf{v}, -\mathbf{w} \rangle$
 - $\langle \mathbf{w}, -\mathbf{v}, -\mathbf{w}, \mathbf{v} \rangle$
 - $\langle -\mathbf{v}, -\mathbf{w}, \mathbf{v}, \mathbf{w} \rangle$
 - $\langle -\mathbf{w}, \mathbf{v}, \mathbf{w}, -\mathbf{v} \rangle$
- Signed-permutation matrix group G
 - $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$



Smoothness of 2^2 directional field

- Transformation between adjacent faces

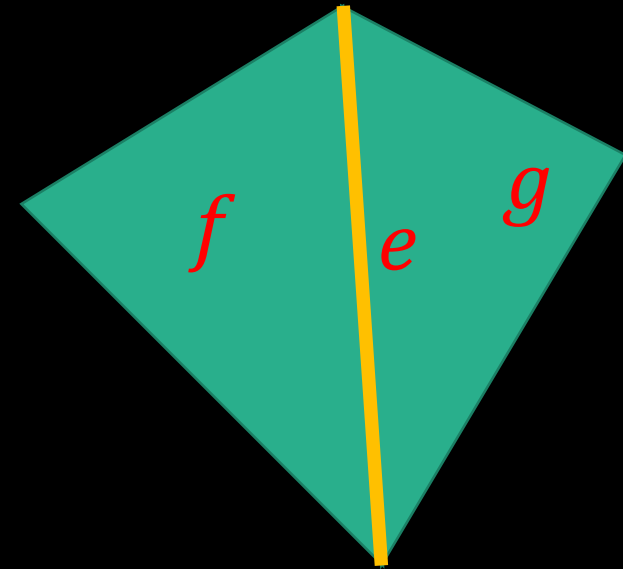
$$[v_f | w_f] = [v_g | w_g] P_{fg}$$

- Smoothness measure

- closeness from P_{fg} to G

- $$E_{fg} = \sum_i (H(P_{fg}[\cdot, i]) + H(P_{fg}[i, \cdot])) + \sum_i \left((P_{fg}[\cdot, i]^2 - 1)^2 + (P_{fg}[i, \cdot]^2 - 1)^2 \right) + (\det P_{fg} - 1)^2$$

- $$H(\eta) = \eta_x^2 \eta_y^2 + \eta_y^2 \eta_z^2 + \eta_z^2 \eta_x^2$$



Example

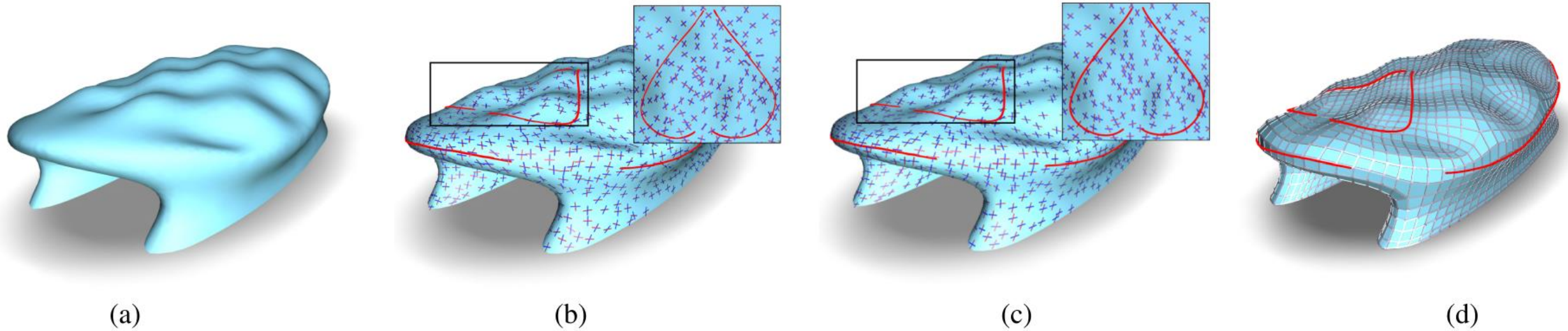


Figure 3: CDF design on an airport terminal model. (a) The original model. (b) An initial CDF from user-specified strokes (red lines). (c) The optimized CDF. (d) The resulting PQ mesh.

Extension to 3D field

Paper: All-Hex Meshing using Singularity-Restricted Field

- $F = \{U, V, W\}$
 - Right-handed: $U \times V \cdot W > 0$
- **24** equivalent classes since the permutations form the chiral cubical symmetry group G .
 - First vector has **six** options
 - Second one has **four** options
 - Third one only has **one** options
 - $6 \times 4 \times 1 = 24$
- Smoothness measure:
 - closeness from P_{fg} to G
 - Similar to former method

Efficiency

- Former methods:
 - Large-scale sparse linear system or nonlinear energy
 - Expensive: too many variables
 - Global view
- Is there any other ways?
 - **Reducing the variable number**
 - **Starting from local view**

Paper: *Instant Field-Aligned Meshes*

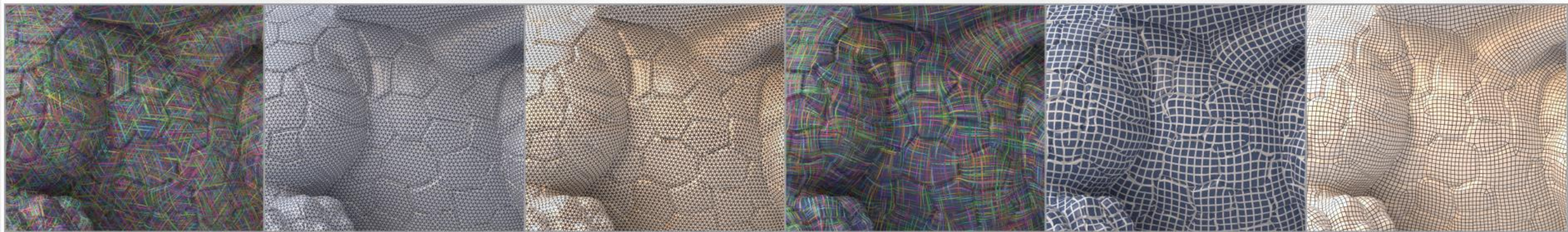


Figure 1: Remeshing a scanned dragon with 13 million vertices into feature-aligned isotropic triangle and quad meshes with $\sim 80k$ vertices. From left to right, for both cases: visualizations of the orientation field, position field, and the output mesh (computed in 71.1 and 67.2 seconds, respectively). For the quad case, we optimize for a quad-dominant mesh at quarter resolution and subdivide once to obtain a pure quad mesh.

Key techniques

- Gauss-Seidel method

$$v_i \leftarrow \frac{1}{\sum_{j \in \Omega(i)} w_{ij}} \sum_{j \in \Omega(i)} w_{ij} v_j$$

- Multiresolution hierarchy
 - improve convergence
 - allow the algorithm to move out of local minima

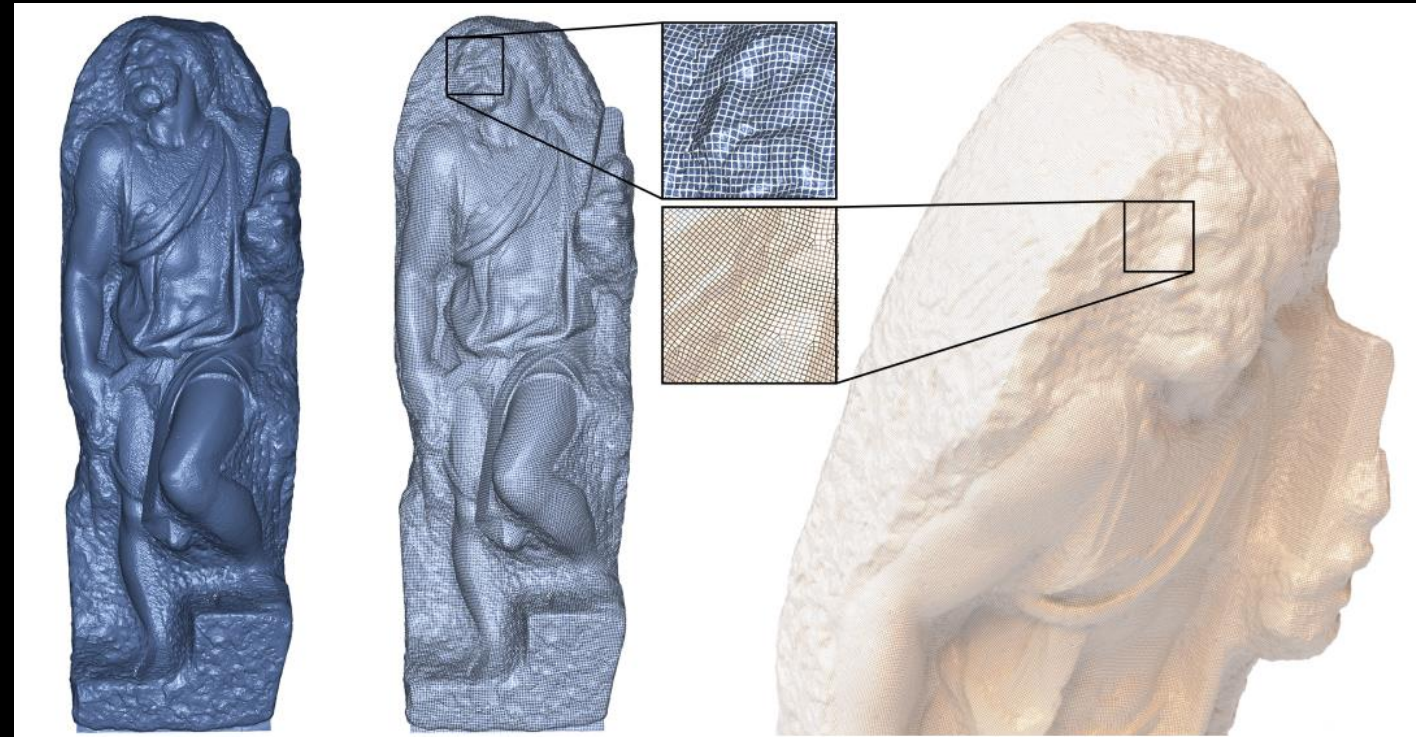


Figure 24: *Our method scales to extremely large datasets, such as the 372M triangle St. Matthew statue acquired by the Digital Michelangelo project [Levoy et al. 2000]. The middle column shows a visualization of the position field, and the right is the final quadrilateral mesh. The entire process takes 9 minutes and 18 seconds.*

Delaunay Triangulations

Xiao-Ming Fu

Outlines

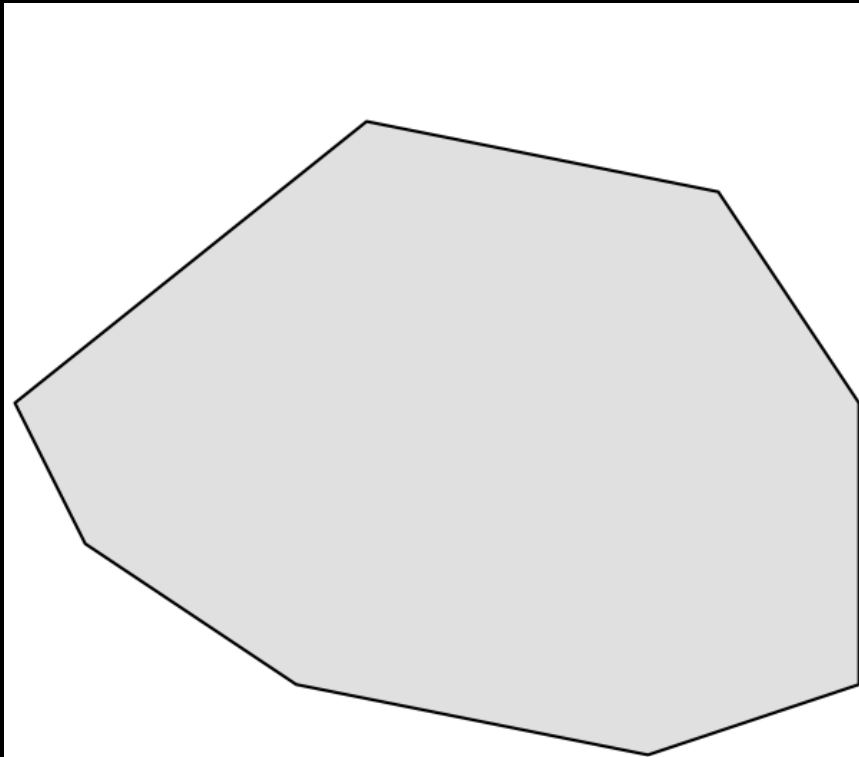
- Introduction
 - Convex hull
 - Triangulation
 - Delaunay triangulation
 - The Lawson Flip algorithm
- Properties
 - Empty Circle
 - Maximize the minimum angle
- Optimal Delaunay triangulation

Outlines

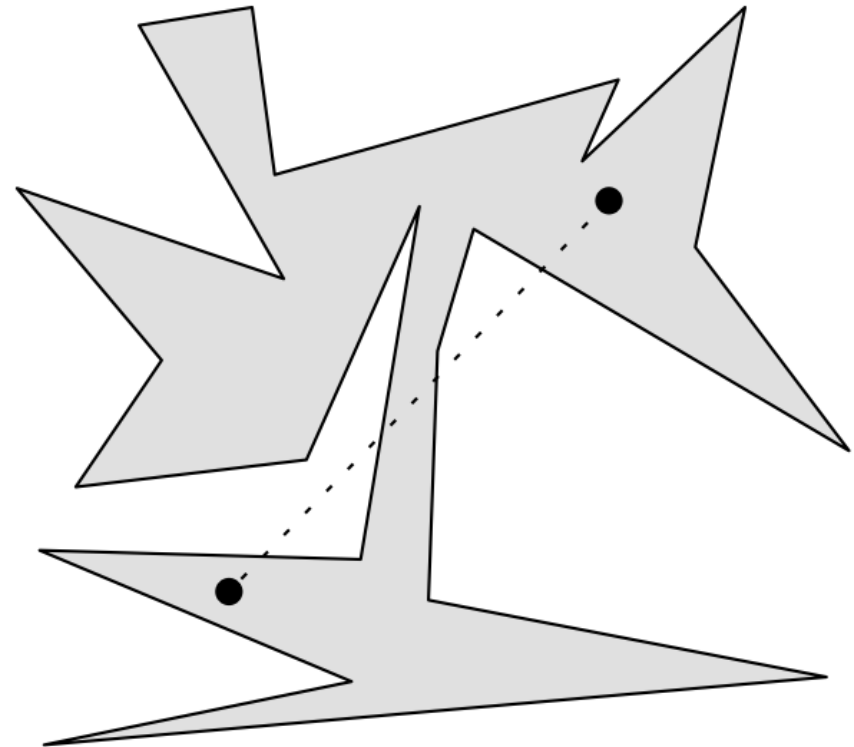
- **Introduction**
 - Convex hull
 - Triangulation
 - Delaunay triangulation
 - The Lawson Flip algorithm
- Properties
 - Empty Circle
 - Maximize the minimum angle
- Optimal Delaunay triangulation

Convex polygon

- One can walk between any two vertices along a straight line without ever leaving the polygon.



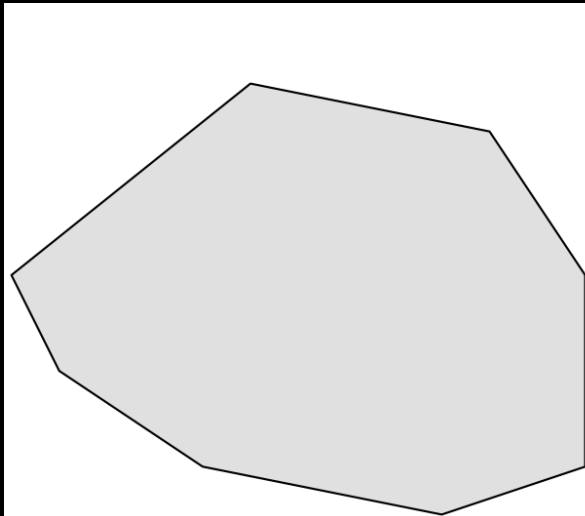
(a) A convex polygon.



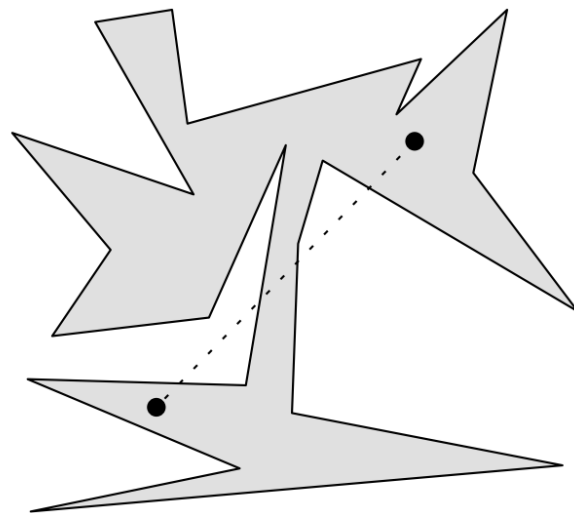
(b) A non-convex polygon

Convex polygon

- A set $P \in R^d$ is *convex* if $\overline{pq} \in P, \forall p, q \in P$.
- An alternatively equivalent way to phrase convexity:
 - For every line $l \in R^d$, the intersection $l \cap P$ is connected



(a) A convex polygon.



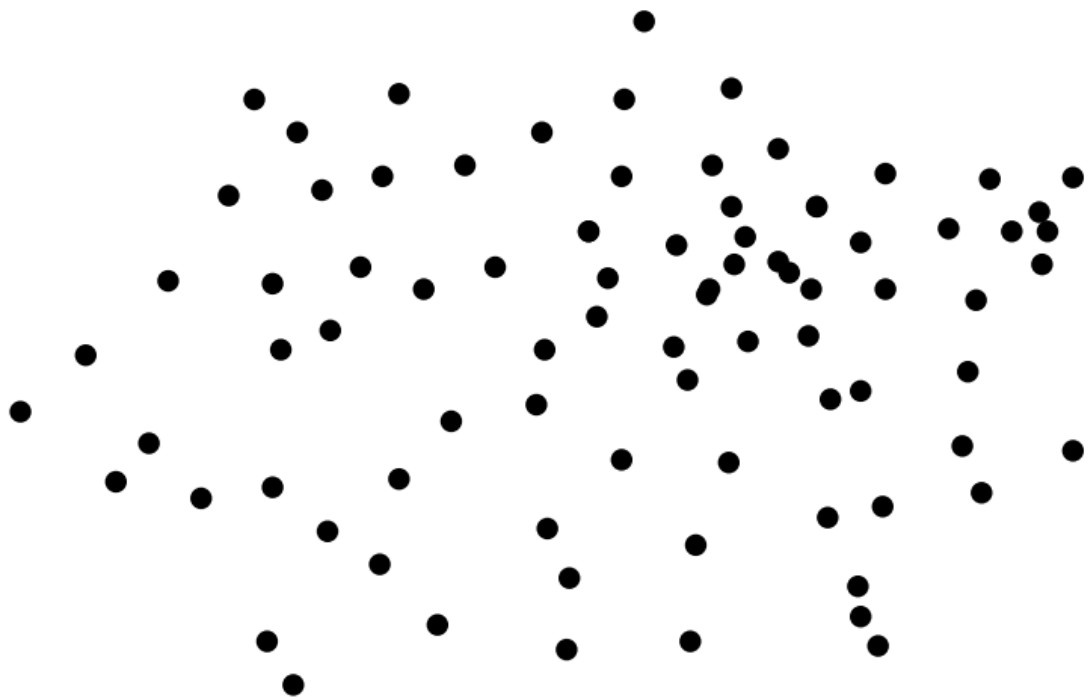
(b) A non-convex polygon

- For any family $\{P_i\}$ of convex sets, the intersection $\bigcap_i P_i$ is convex.

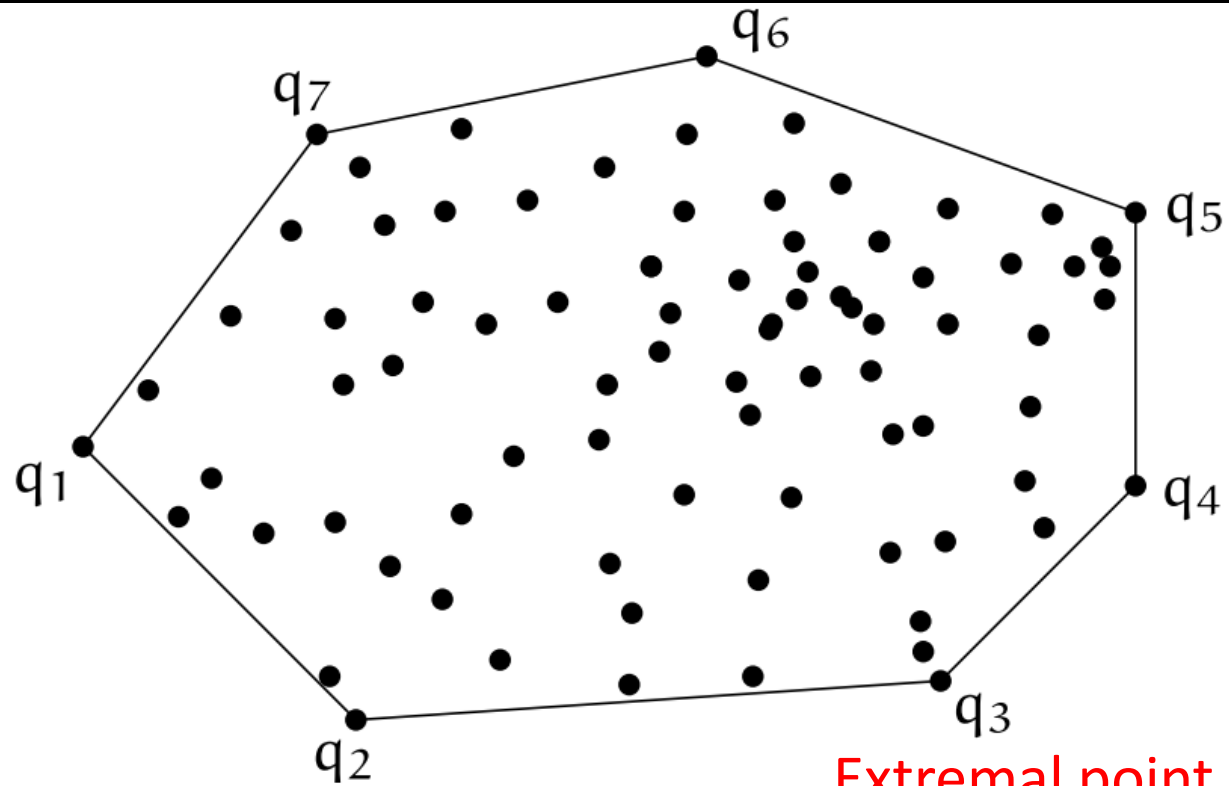
Convex hull

- The convex hull of a finite point set $P \in R^d$ forms a convex polytope, denoted as $\text{conv}(P)$.
- Each $p \in P$ for which $p \notin \text{conv}(P \setminus \{p\})$ is called a vertex of $\text{conv}(P)$.
- A vertex of $\text{conv}(P)$ is also called an *extremal point* of P .
- A convex polytope in R^2 is called a convex polygon.

An example of $\text{conv}(P)$



(a) Input.



(b) Output.

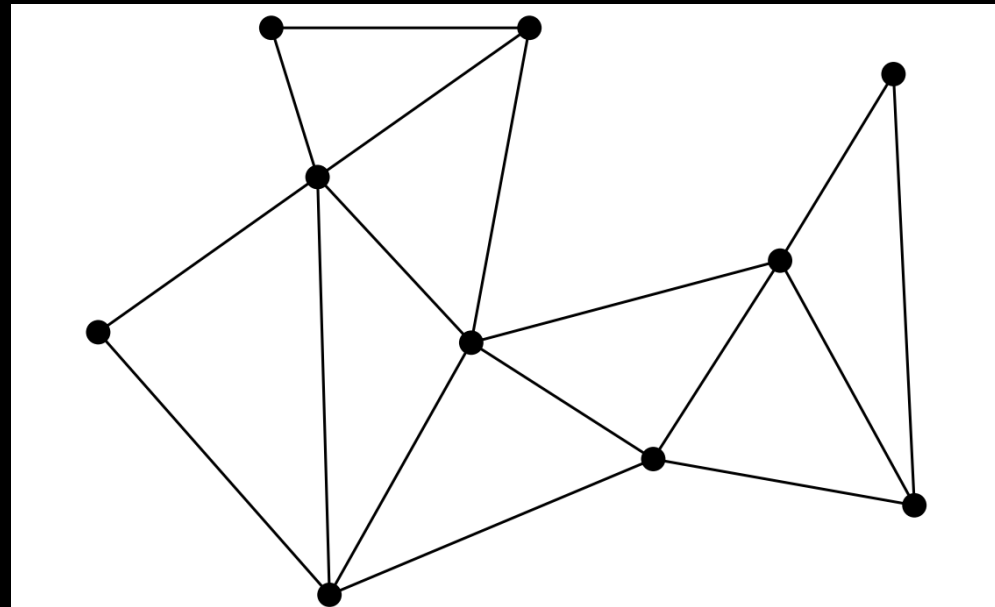
Extremal point

Trivial algorithms of Convex hull

- Carathéodory's Theorem
 - Test for every point $p \in P$ whether there are $q, r, s \in P \setminus \{p\}$ such that p is inside the triangle with vertices q, r , and s .
 - Runtime $O(n^4)$.
- The Separation Theorem:
 - Test for every pair $(p, q) \in P^2$ whether all points from $P \setminus \{p, q\}$ are to the left of the directed line through p and q (or on the line segment \overline{pq}).
 - Runtime $O(n^3)$.

Triangulation of polygon

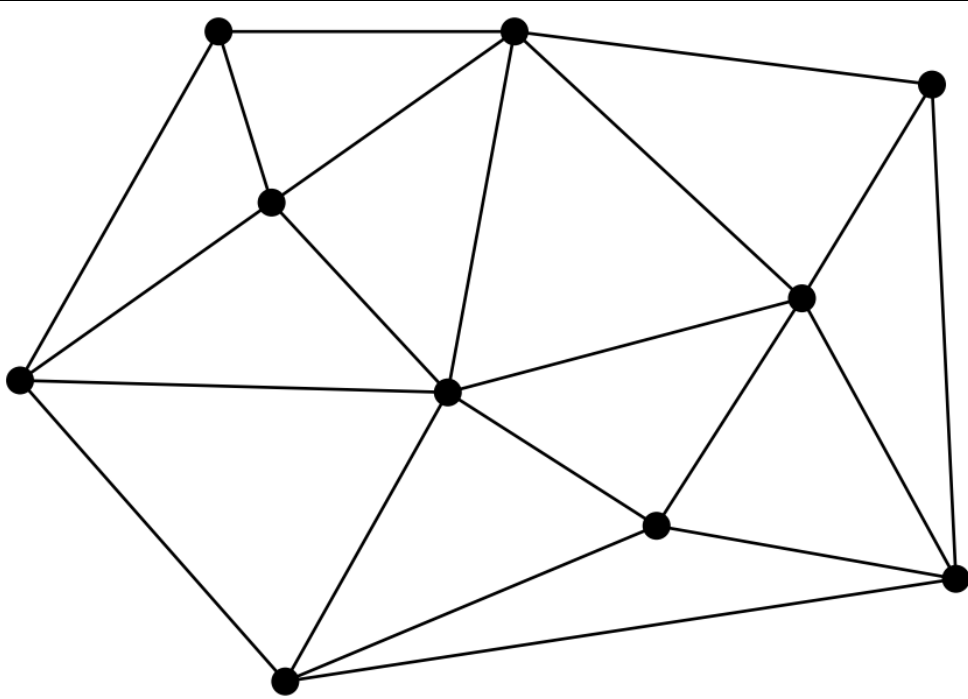
- A triangulation nicely **partitions a polygon into triangles**, which allows, for instance, to easily compute the area or a guarding of the polygon.
- Another typical application scenario is to use a triangulation T for interpolation.



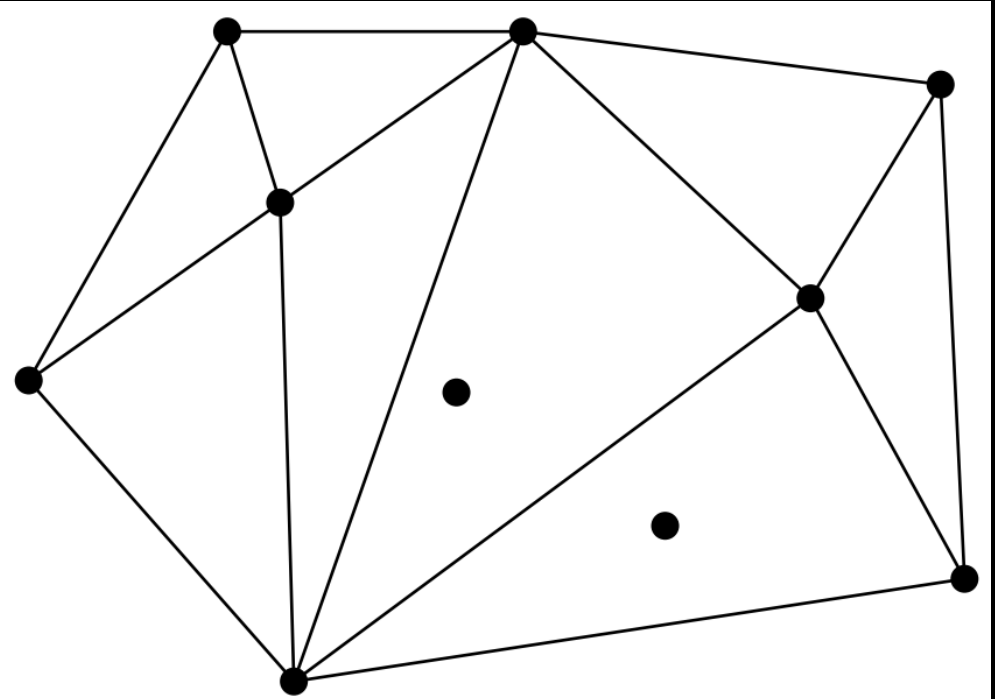
(a) Simple polygon triangulation.

Triangulation of a point set

- A triangulation should then partition the **convex hull** while **respecting the points** in the interior.



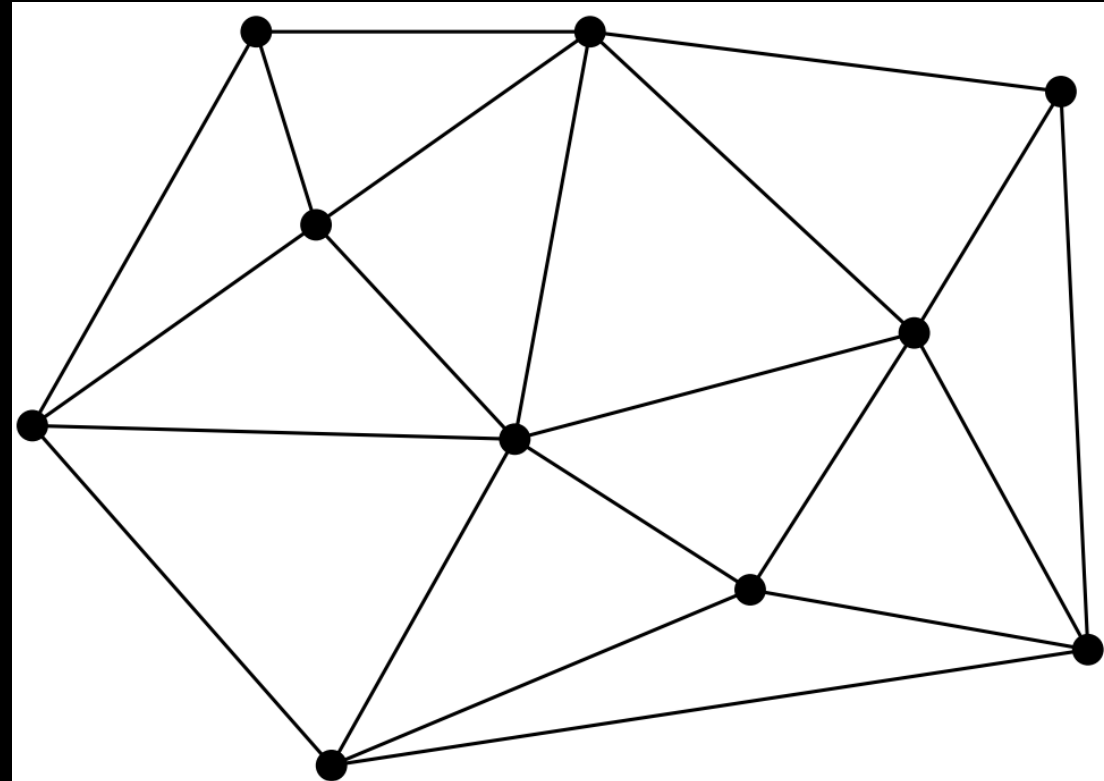
(b) Point set triangulation.



(c) Not a triangulation.

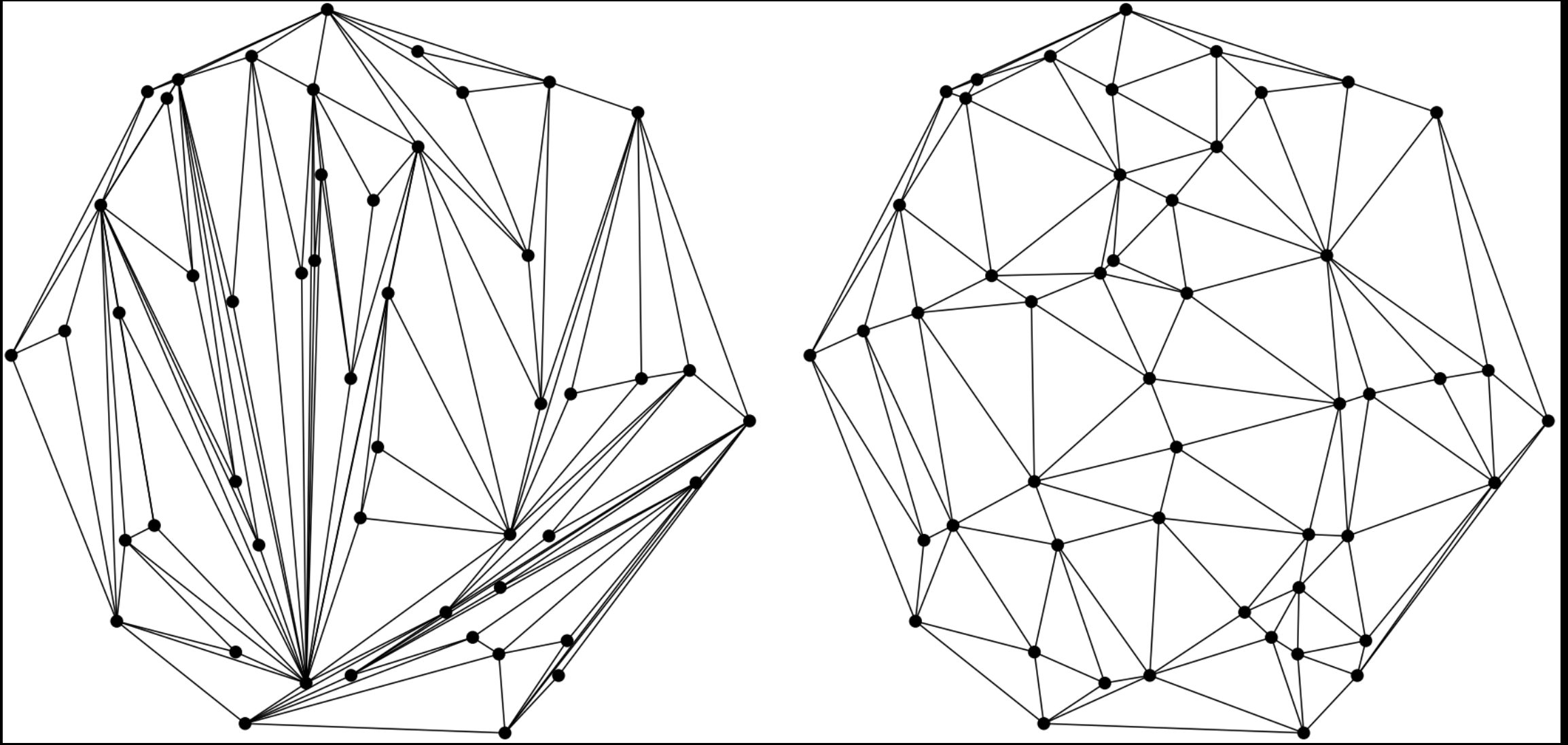
Definition

- A triangulation of a finite point set $P \subset \mathbb{R}^2$ is a collection \mathcal{T} of triangles, such that:
 - (1) $\text{conv}(P) = \bigcup_{T \in \mathcal{T}} T$
 - (2) $P = \bigcup_{T \in \mathcal{T}} V(T)$
 - (3) For every distinct pair $T, U \in \mathcal{T}$, the intersection $T \cap U$ is either a common vertex, or a common edge, or empty.



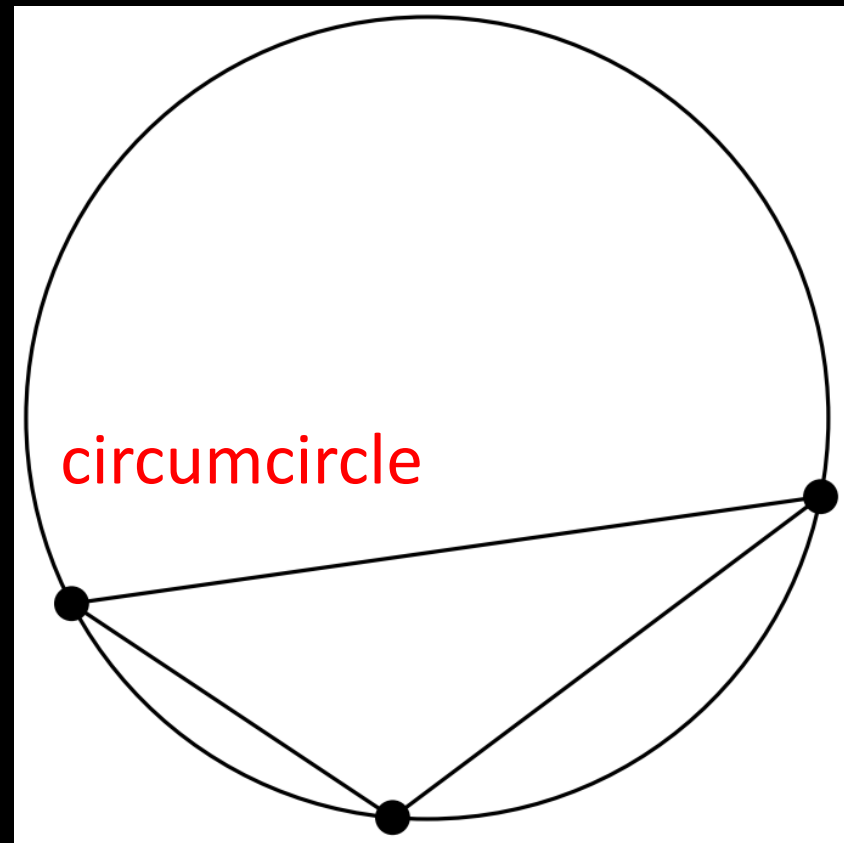
(b) Point set triangulation.

Various triangulations

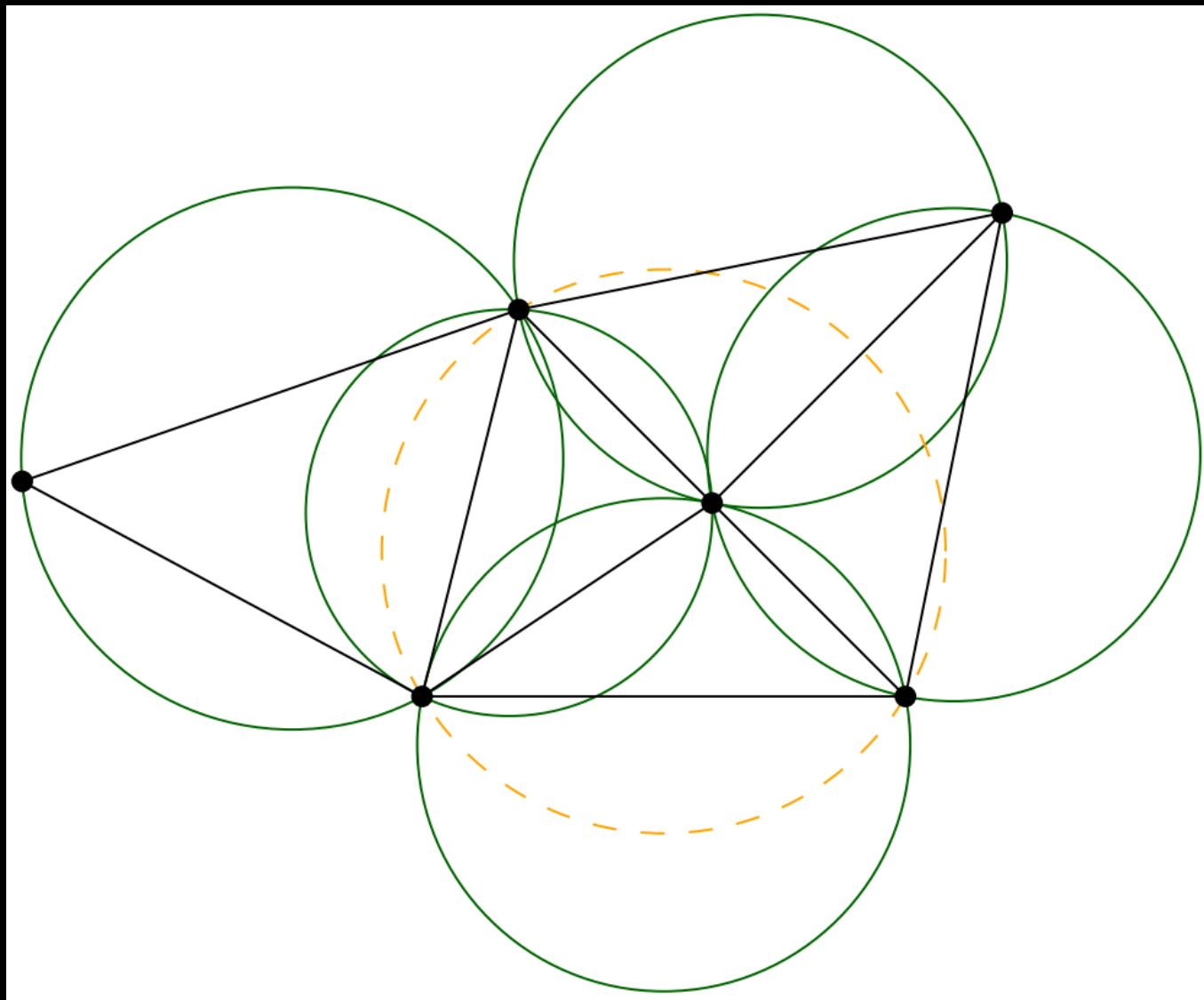


Delaunay triangulation

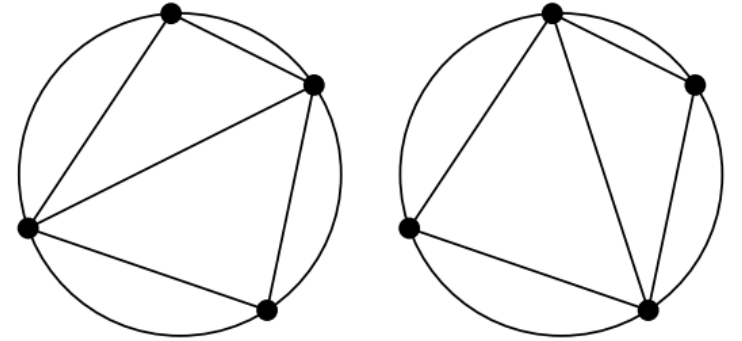
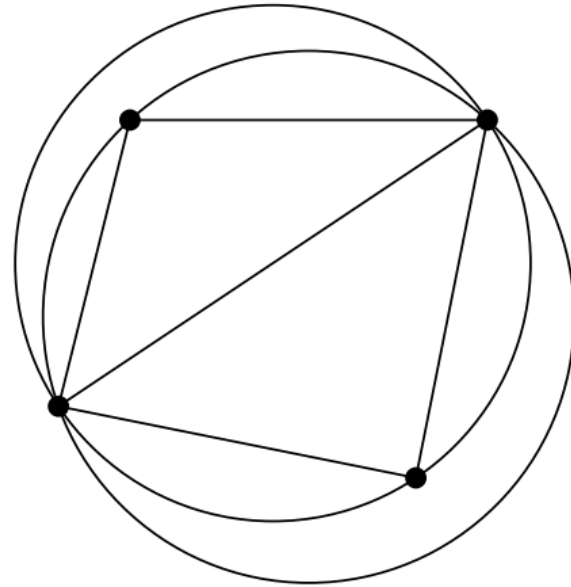
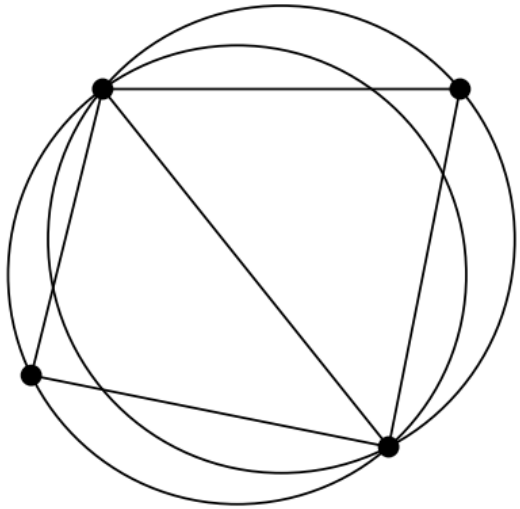
- Definition: For a given set P of discrete points in a plane is a triangulation $DT(P)$ such that no point in P is inside the circumcircle of any triangle in $DT(P)$.
- Empty Circle property



Empty Circle



Four points in convex position



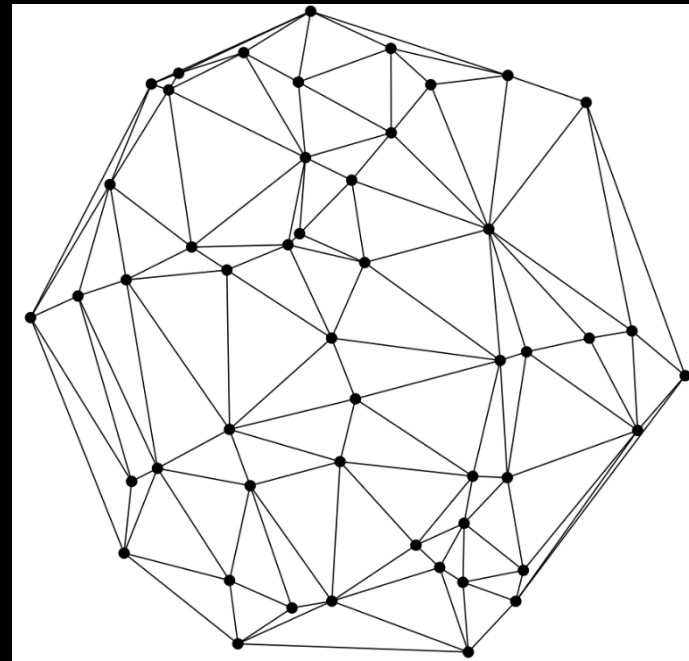
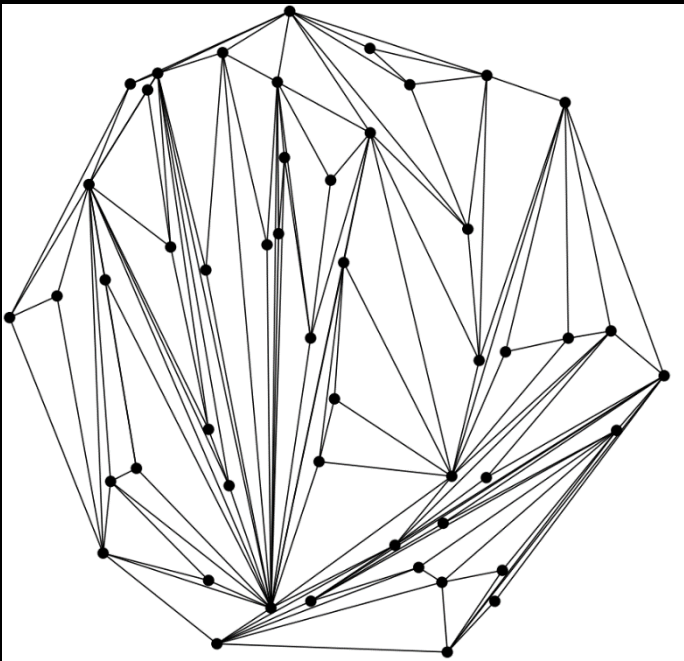
(a) Delaunay triangulation.

(b) Non-Delaunay triangulation.

(c) Two Delaunay triangulations.

The Lawson Flip algorithm

- (1) Compute some triangulation of P
- (2) While there exists a subtriangulation of four points in convex position that is not Delaunay, replace this subtriangulation by the other triangulation of the four points.



Theorem

Let $P \subseteq R^2$ be a set of n points, equipped with some triangulation \mathcal{T} . The Lawson flip algorithm terminates after at most $\binom{n}{2} = O(n^2)$ flips, and the resulting triangulation D is a Delaunay triangulation of P .

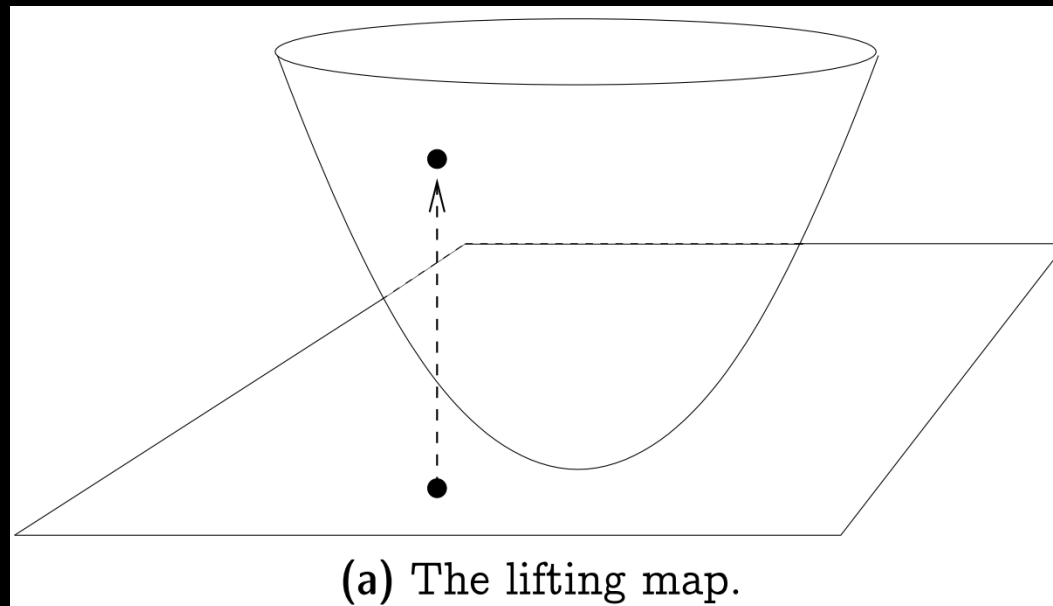
Two-step proof:

1. The program described above always terminates.
2. The algorithm does what it claims to do, namely the result is a Delaunay triangulation.

The Lifting Map

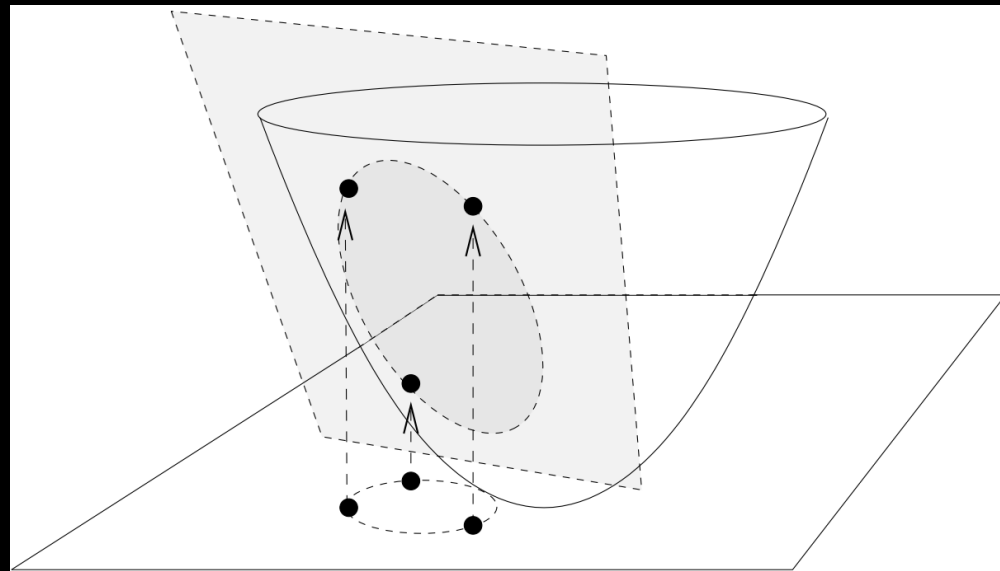
- Given a point $p = (x, y) \in R^2$, its lifting $l(p)$ is the point $l(p) = (x, y, x^2 + y^2) \in R^3$

Geometrically, l “lifts” the point vertically up until it lies on the unit paraboloid:



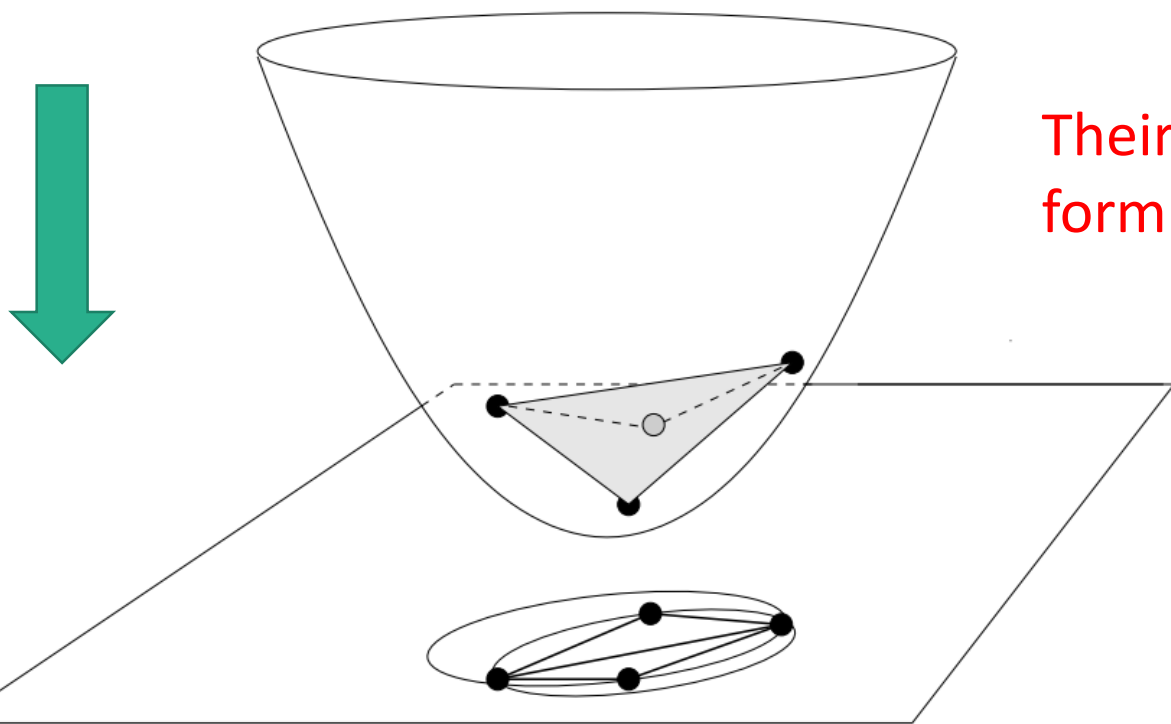
Important property of the lifting map

- Lemma: Let $C \subseteq R^2$ be a circle of positive radius. The “lifted circle” $l(C) = \{l(p) | p \in C\}$ is contained in **a unique plane** $h(C) \subseteq R^3$.
- Moreover, a point $p \in R^2$ is strictly **inside** (outside, respectively) of C if and only if the lifted point $l(p)$ is strictly **below** (above, respectively) $h(C)$.

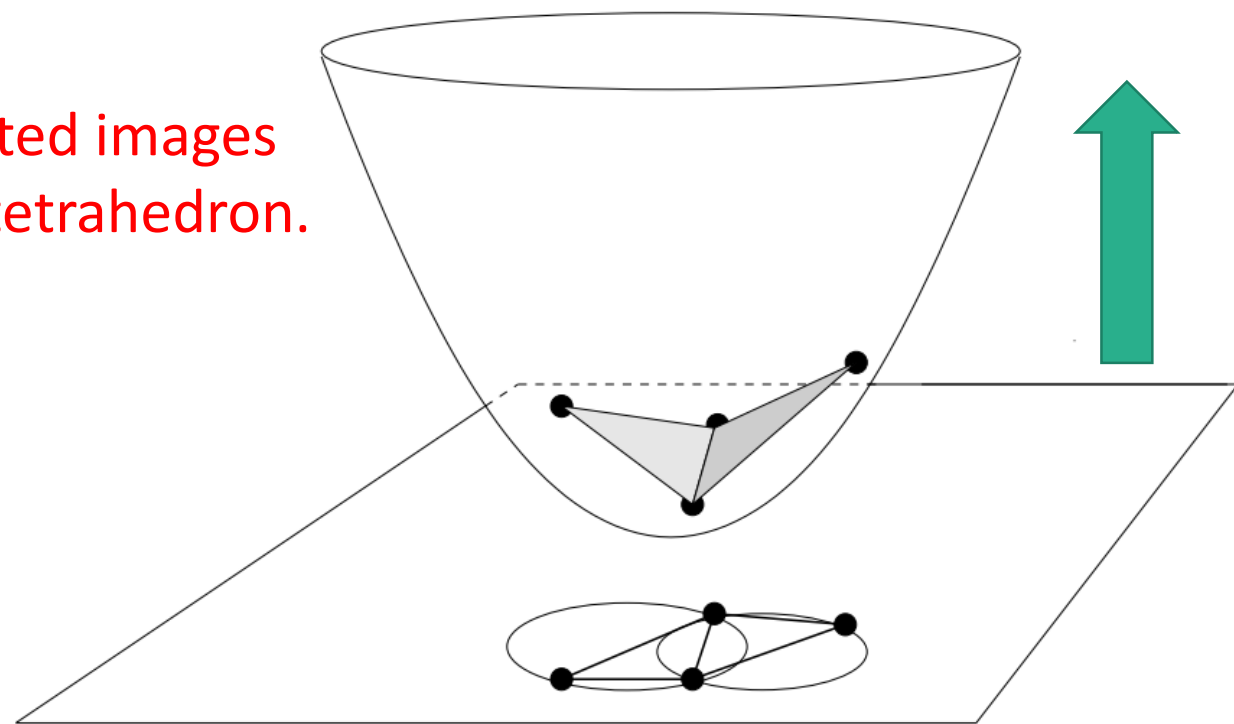


(b) Points on/inside/outside a circle are lifted to points on/below/above a plane.

(1) Termination



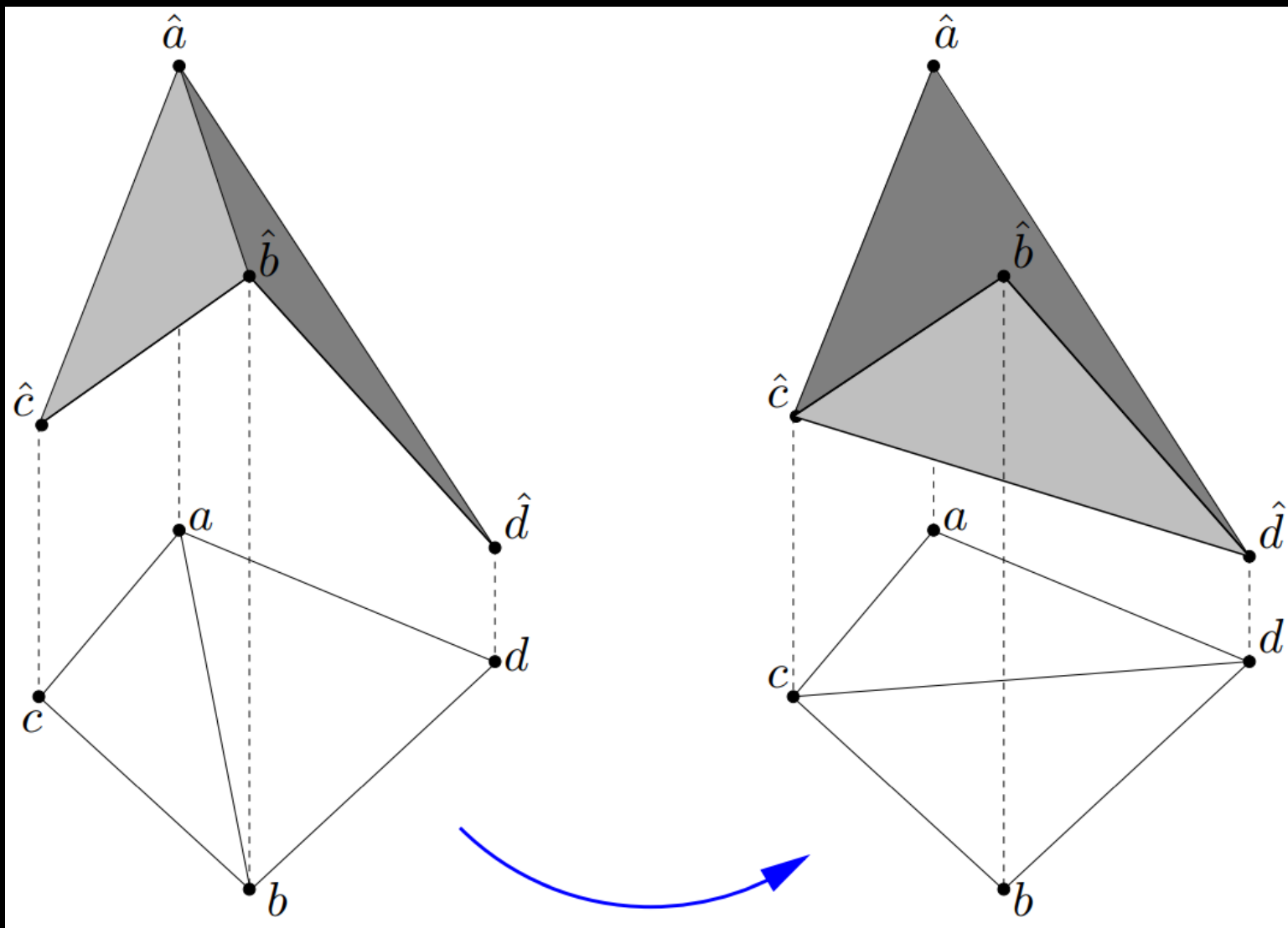
Their lifted images
form a tetrahedron.



(a) Before the flip: the top two triangles of the tetrahedron and the corresponding non-Delaunay triangulation in the plane.

(b) After the flip: the bottom two triangles of the tetrahedron and the corresponding Delaunay triangulation in the plane.

(1) Termination



(1) Termination

- A Lawson flip can therefore be interpreted as an operation that replaces the **top two triangles** of a tetrahedron by the **bottom two ones**.
- If we consider the lifted image of the current triangulation, we therefore have a surface in R^3 whose pointwise height can only decrease through Lawson flips.
- In particular, once an edge has been flipped, this edge will be strictly above the resulting surface and can therefore never be flipped a second time. Since n points can span at most $\binom{n}{2}$ edges, the bound on the number of flips follows.

(2) Correctness

- Locally Delaunay: Let Δ, Δ' be two adjacent triangles in the triangulation D that results from the Lawson flip algorithm. Then the circumcircle of Δ does not have any vertex of Δ' in its interior, and vice versa.
- Locally Delaunay \Leftrightarrow Globally Delaunay:
 - contradiction

Locally Delaunay \Leftrightarrow Globally Delaunay

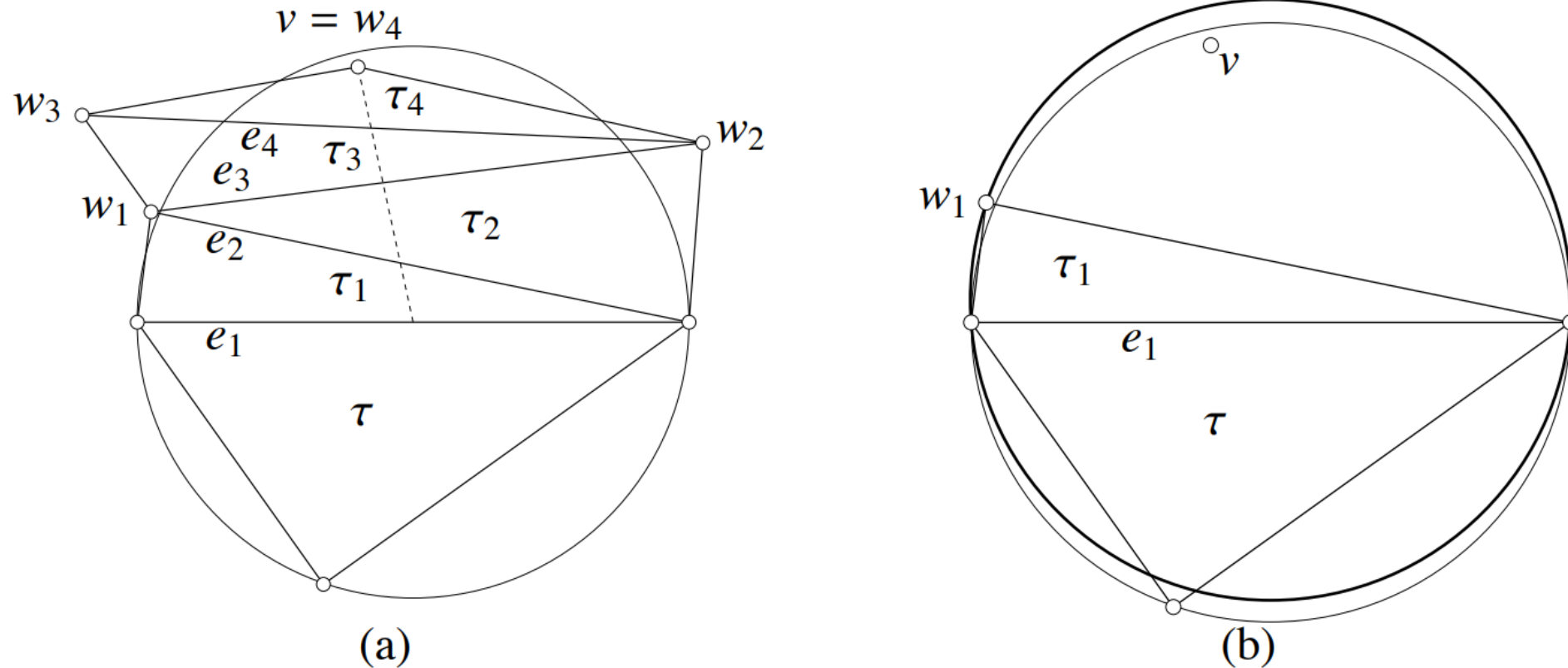
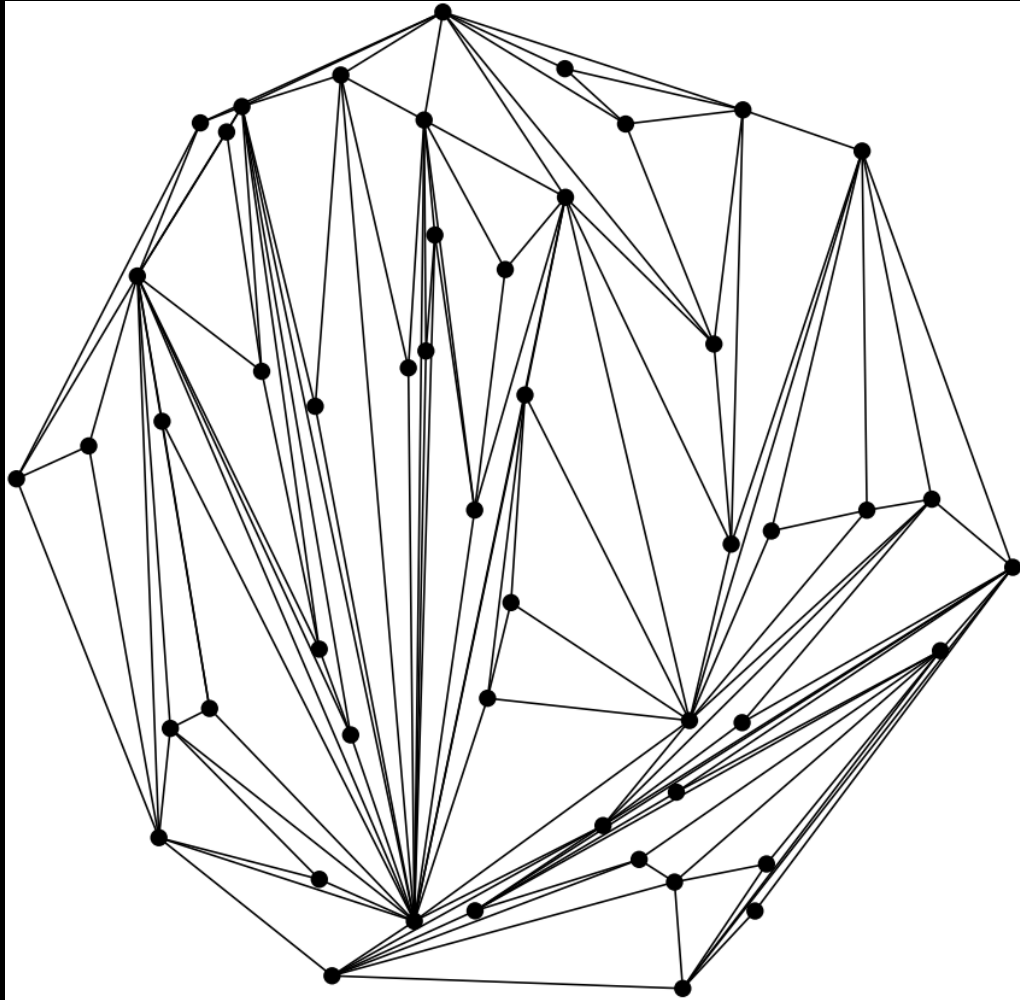


Figure 2.7: (a) Because τ 's open circumdisk contains v , some edge between v and τ is not locally Delaunay. (b) Because v lies above e_1 and in τ 's open circumdisk, and because w_1 lies outside τ 's open circumdisk, v must lie in τ_1 's open circumdisk.

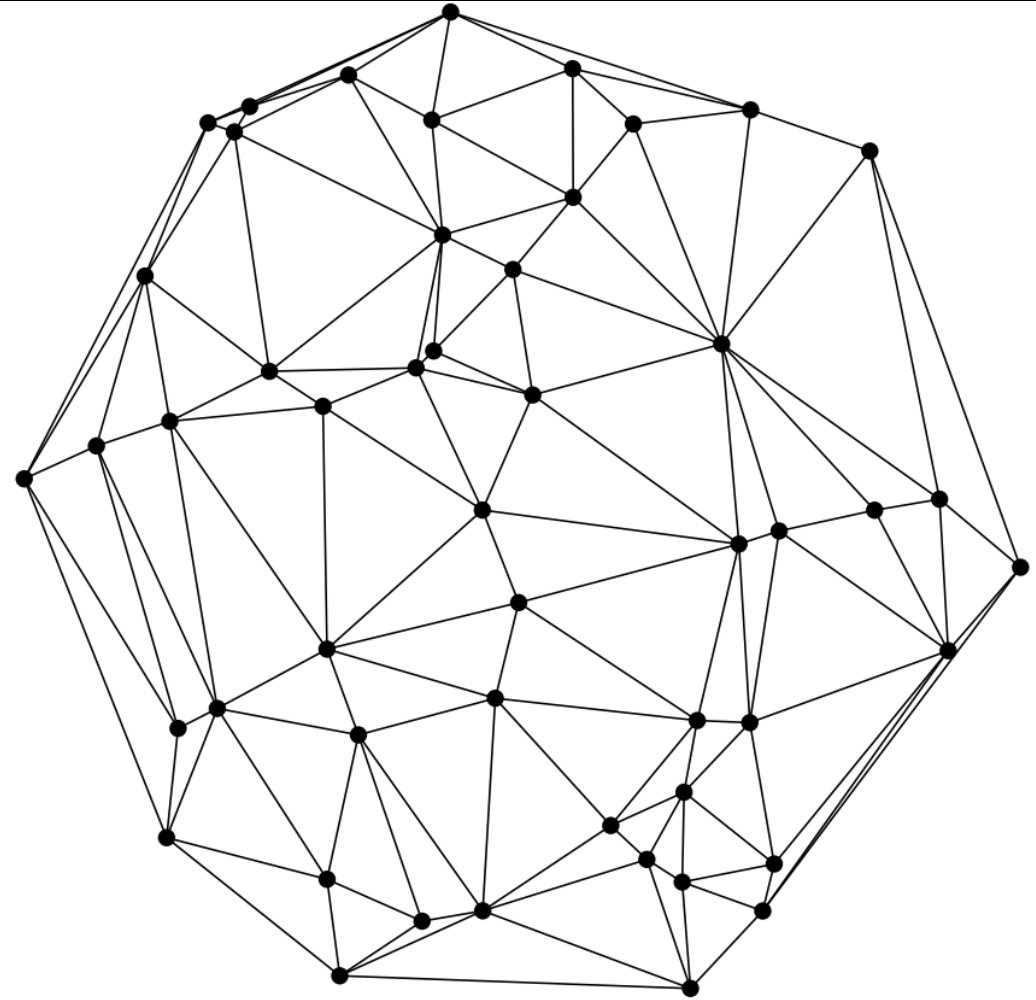
Outlines

- Introduction
 - Convex hull
 - Triangulation
 - Delaunay triangulation
 - The Lawson Flip algorithm
- **Properties**
 - Empty Circle
 - Maximize the minimum angle
- Optimal Delaunay triangulation

Maximize the minimum angle



Long and skinny triangles



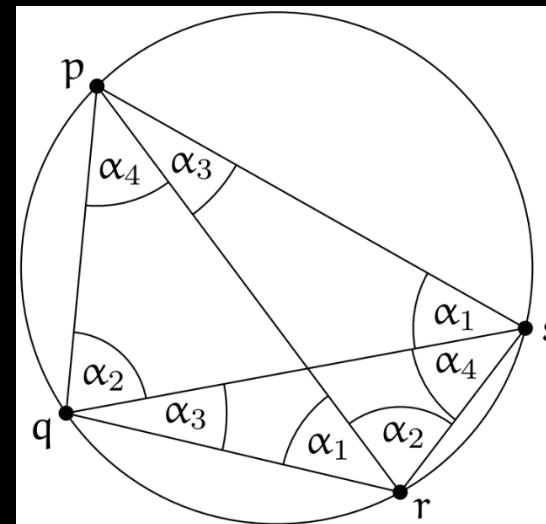
Much closer to an equilateral triangle

Maximize the minimum angle

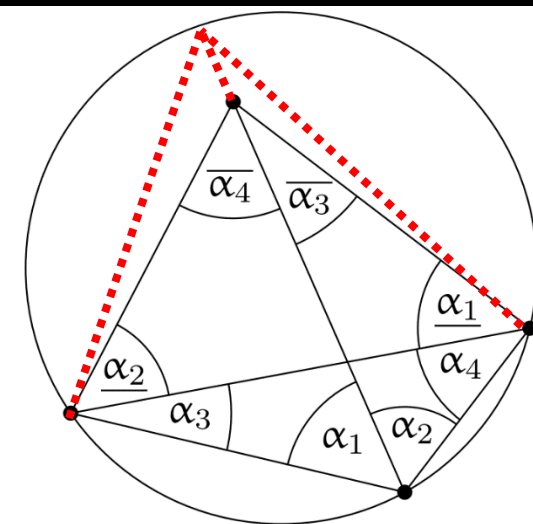
- Indeed, we will show that Delaunay triangulations maximize the smallest angle among all triangulations of a given point set.
- Note that this does not imply that there are no long and skinny triangles in a Delaunay triangulation.
- But if there is a long and skinny triangle in a Delaunay triangulation, then there is an at least as long and skinny triangle in **every** triangulation of the point set.

Maximize the minimum angle

- A flip replaces six interior angles by six other interior angles, and we will actually show that the smallest of the six angles strictly increases under the flip.
 - Before the flip:
 - $\alpha_1 + \alpha_2, \alpha_3, \alpha_4, \underline{\alpha_1}, \underline{\alpha_2}, \overline{\alpha_3} + \overline{\alpha_4}$
 - After the flip:
 - $\alpha_1, \alpha_2, \overline{\alpha_3}, \overline{\alpha_4}, \underline{\alpha_1} + \alpha_4, \underline{\alpha_2} + \alpha_3$
 - $\alpha_1 > \underline{\alpha_1}, \alpha_2 > \underline{\alpha_2}, \overline{\alpha_3} > \alpha_3, \overline{\alpha_4} > \alpha_4$
 - $\underline{\alpha_1} + \alpha_4 > \alpha_4, \underline{\alpha_2} + \alpha_3 > \alpha_3$



(a) Four cocircular points and the induced eight angles.

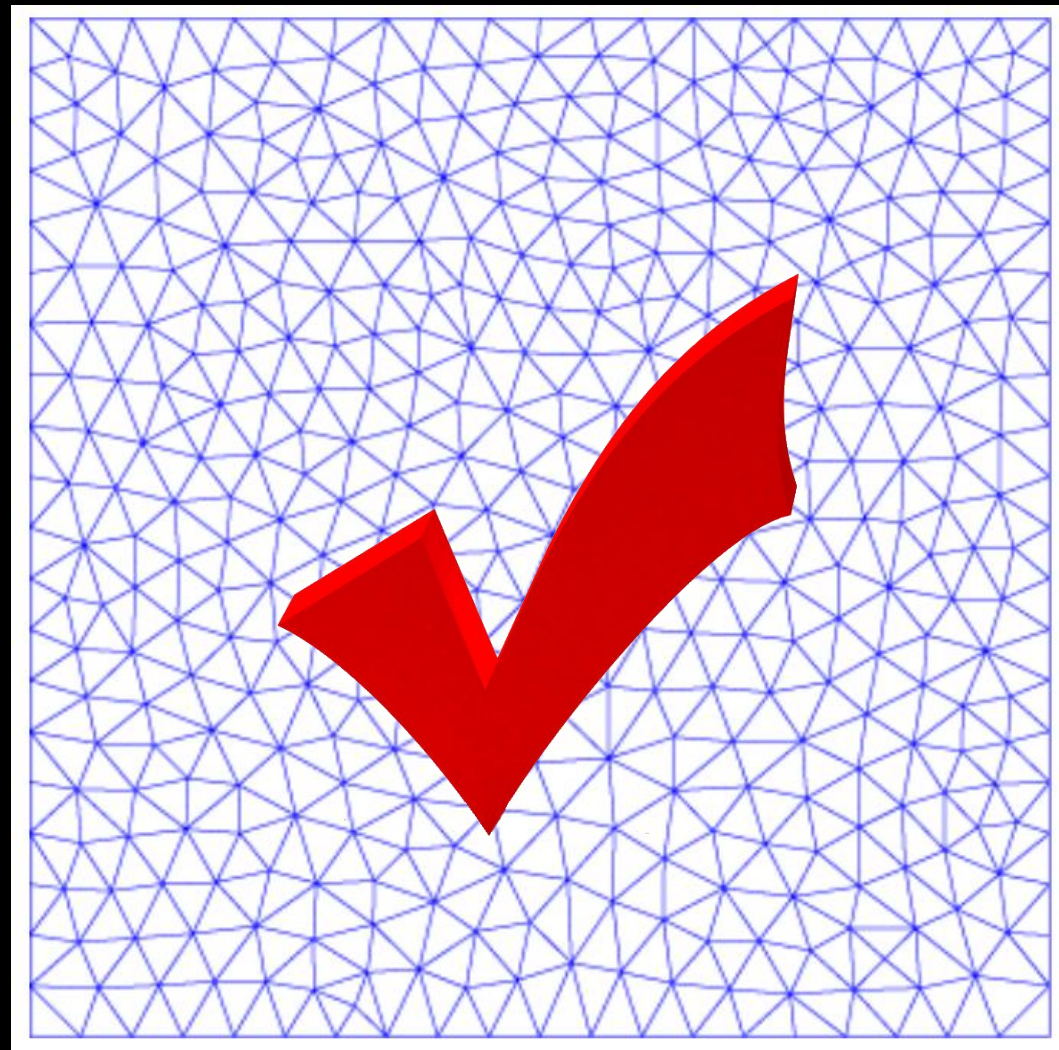
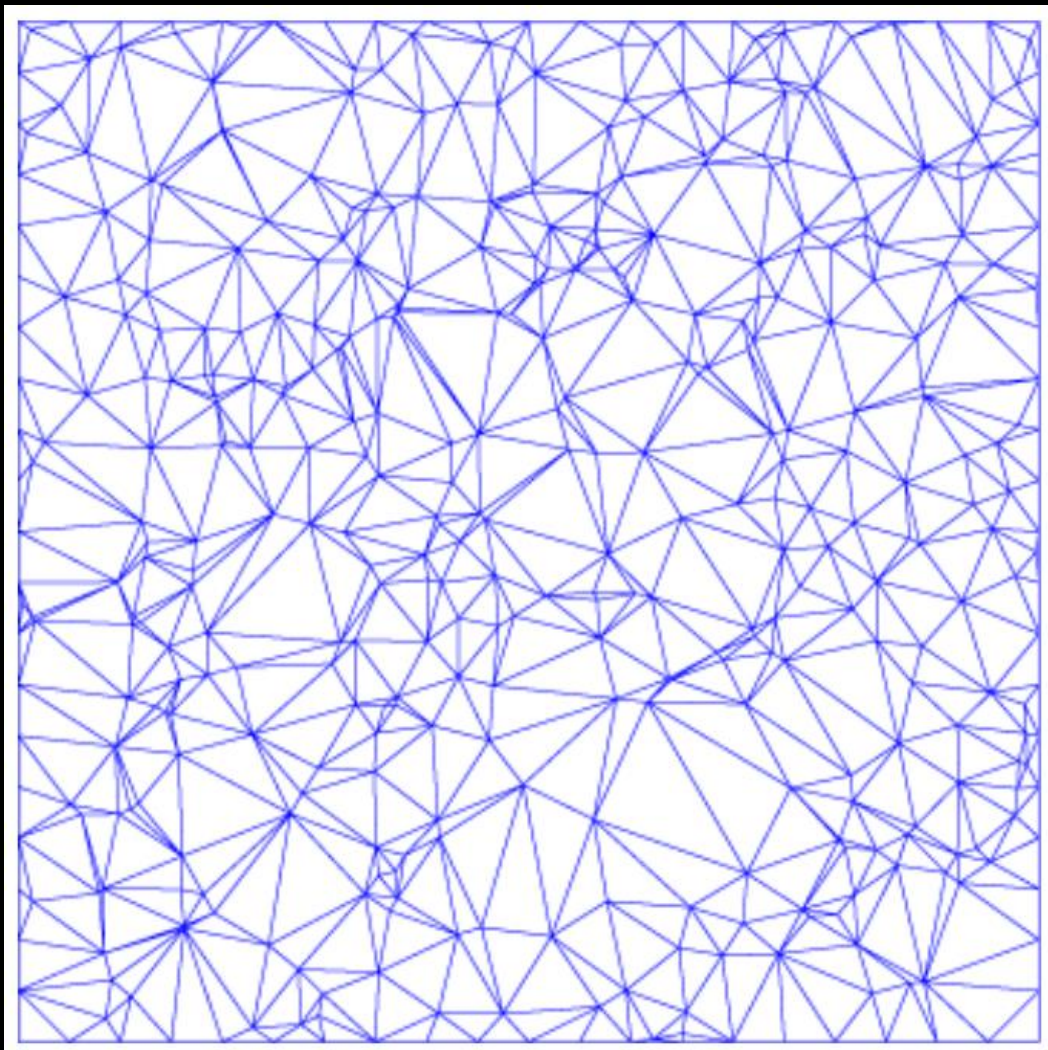


(b) The situation before a flip.

Outlines

- Introduction
 - Convex hull
 - Triangulation
 - Delaunay triangulation
 - The Lawson Flip algorithm
- Properties
 - Empty Circle
 - Maximize the minimum angle
 - Euclidean Minimum Spanning Tree
- **Optimal Delaunay triangulation**

Optimal Delaunay triangulation



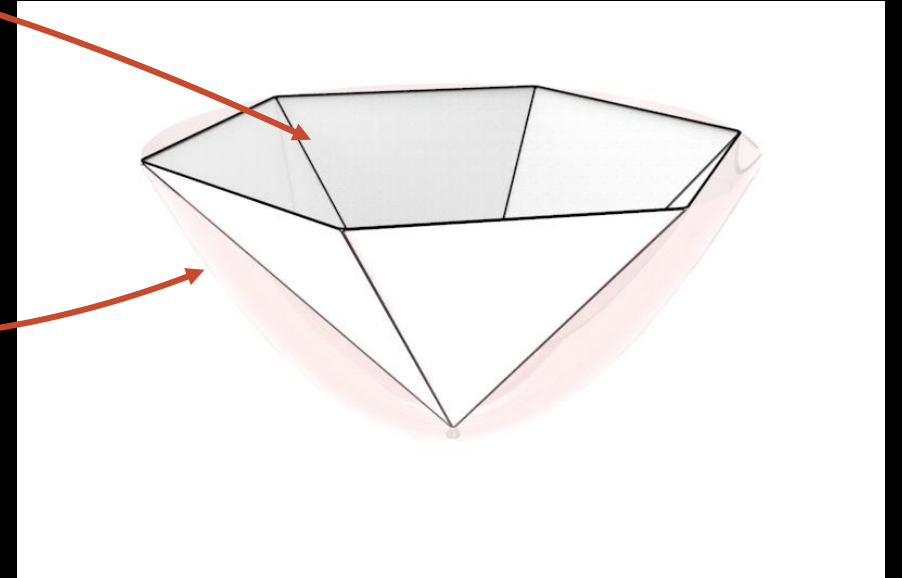
Thinking from surface approximation

$$E = \sum_{T \in \mathcal{T}} \int_T |\hat{u}(x) - u(x)| dx$$

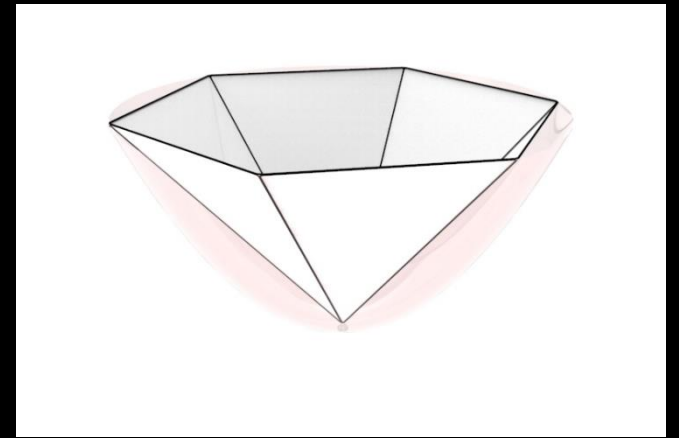
$$u(x): z = x^2 + y^2$$

$\hat{u}(x)$: piecewise linear interpolation of u

\mathcal{T} : a triangulation



Fix positions of vertices, Delaunay triangulation is optimal.



Update of vertices' positions

- Fix the triangulation, update the vertices.

$$E = \sum_{T \in \mathcal{T}} \int_T |\hat{u}(x) - u(x)| dx = \sum_{T \in \mathcal{T}} \int_T \hat{u}(x) dx + C$$

$$= \sum_{T \in \mathcal{T}} \frac{|T|}{3} (u(p_i) + u(p_j) + u(p_k)) + C$$

$$\nabla E_{p_i} = \sum_{T \in \Omega(i)} \frac{\nabla |T|}{3} (u(p_i) + u(p_j) + u(p_k)) + \frac{|\Omega|}{3} \nabla u(p_i) = 0$$

Because $\sum_{T \in \Omega(i)} \frac{\nabla |T|}{3} u(p_i) = 0$

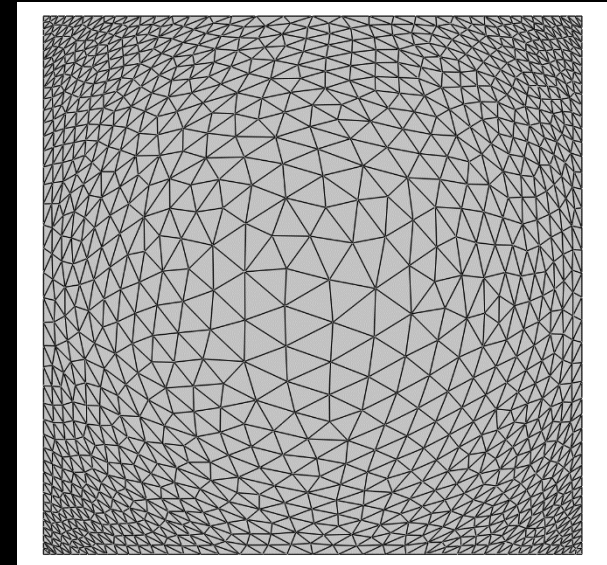
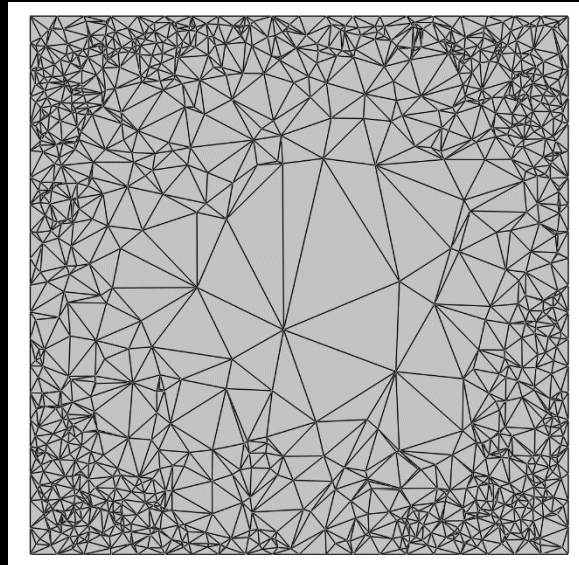
$$\nabla u(p_i) = -\frac{1}{|\Omega|} \sum_{T \in \Omega(i)} \frac{\nabla |T|}{3} (u(p_j) + u(p_k))$$

Optimal Delaunay triangulation

- Alternately iterate:
 - Update triangulation
 - Update vertices
- Extension to any convex function $u(x)$:
 - Delaunay triangulation \rightarrow regular triangulation

$$u(x, y) = e^{\frac{(x^2+y^2)}{10}}$$

$$\Omega = [-5, 5]^2$$



Voronoi Diagram

Xiao-Ming Fu

Outlines

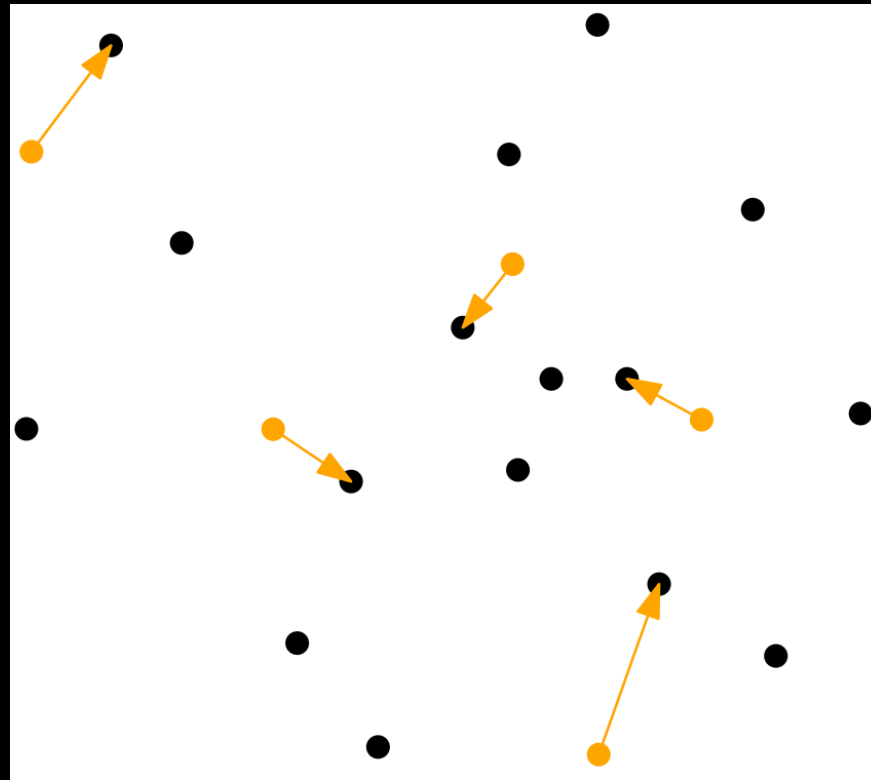
- Introduction
 - Post Office Problem
 - Voronoi Diagram
- Duality: Delaunay triangulation
- Centroidal Voronoi tessellations (CVT)
 - Definition
 - Applications
 - Algorithms

Outlines

- **Introduction**
 - Post Office Problem
 - Voronoi Diagram
- Duality: Delaunay triangulation
- Centroidal Voronoi tessellations (CVT)
 - Definition
 - Applications
 - Algorithms

Post Office Problem

- Suppose there are n post offices p_1, \dots, p_n in a city.
- Someone who is located at a position q within the city would like to know which post office is **closest** to him.

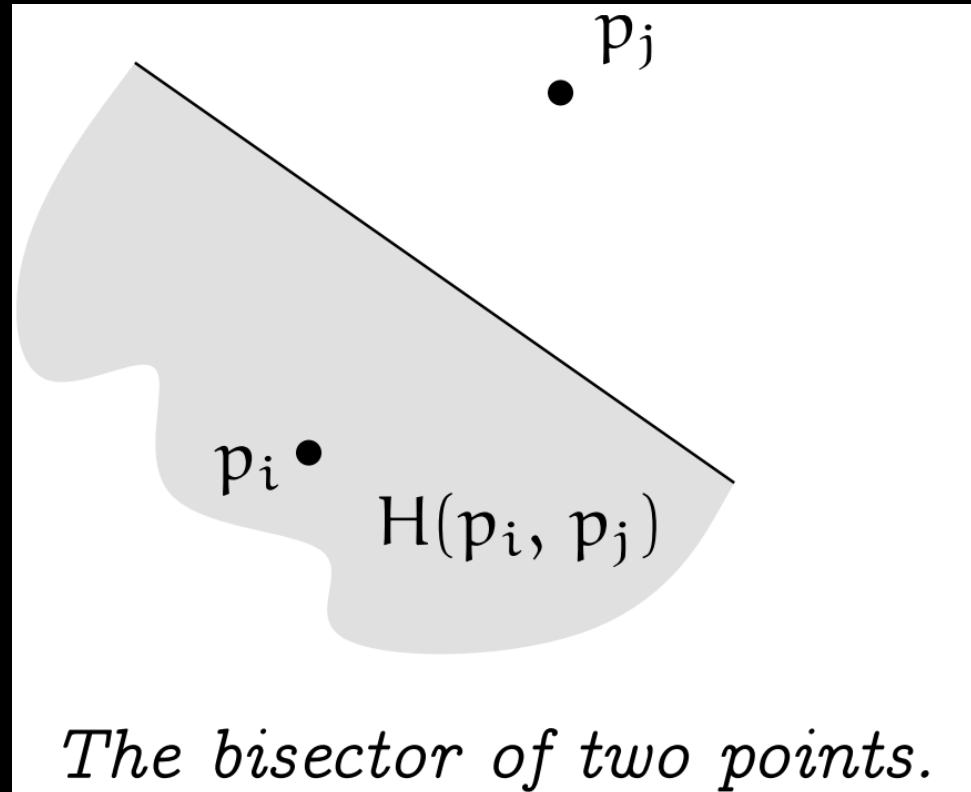


Post Office Problem

- Do not think from the queries.
- Our long term goal is to come up with a **data structure** on top of P that allows to answer any possible query efficiently.
- Basic idea:
 - Partition the query space into regions on which is the answer is the same.
 - In our case, this amounts to partition the plane into regions such that for all points within a region the same point from P is closest.

Two post offices

- Proposition
 - For any two distinct points in R^d , the **bisector** is a hyperplane, that is, in R^2 it is a line.



Voronoi cell

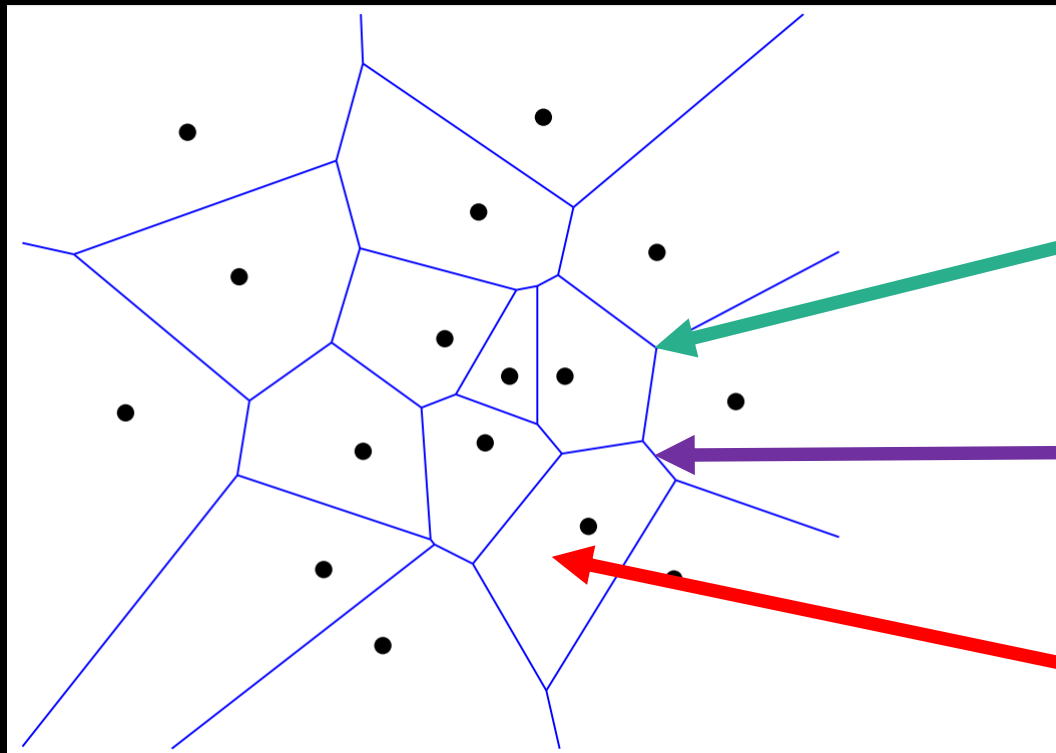
- Given a set $P = \{p_1, \dots, p_n\}$ of points in R^2 , for $p_i \in P$ denote the Voronoi cell $VP(i)$ of p_i by

$$VP(i) := \{q \in R^2 \mid \|q - p_i\| \leq \|q - p\|, \forall p \in P\}$$

1. $VP(i) = \bigcap_{j \neq i} H(p_i, p_j)$
2. $VP(i)$ is non-empty and convex.
3. Observe that every point of the plane lies in some Voronoi cell but no point lies in the interior of two Voronoi cells. Therefore these cells form a **subdivision** of the plane.

Voronoi Diagram

- The Voronoi Diagram $VD(P)$ of a set $P = \{p_1, \dots, p_n\}$ of points in R^2 is the subdivision of the plane induced by the Voronoi cells $VP(i)$, for $i = 1, \dots, n$.



$VV(P)$: the set of vertices

$VE(P)$: the set of edges

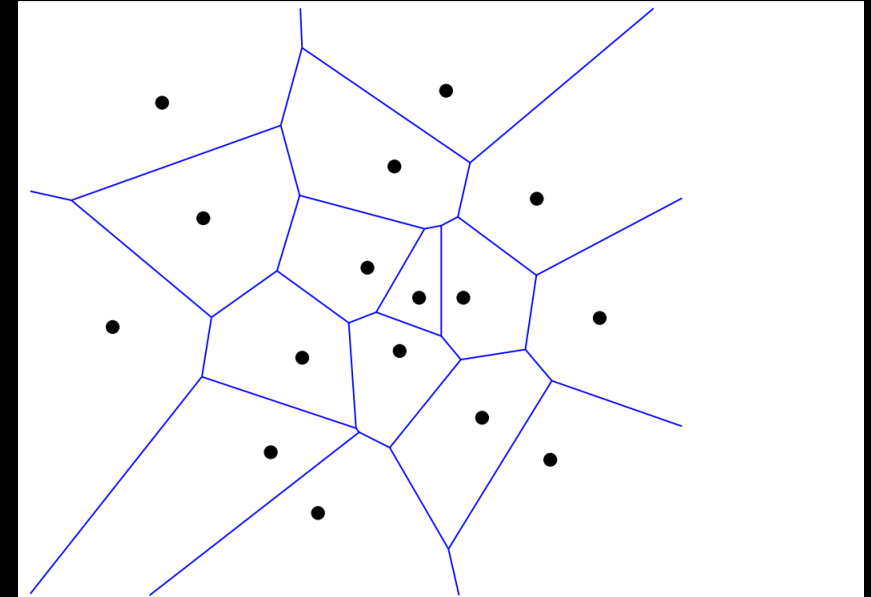
$VR(P)$: the set of regions

Example: The Voronoi diagram of a point set.

Lemma 1

- For every vertex $v \in VV(P)$ the following statements hold.
 - 1) v is the common intersection of at least three edges from $VE(P)$;
 - 2) v is incident to at least three regions from $VR(P)$;

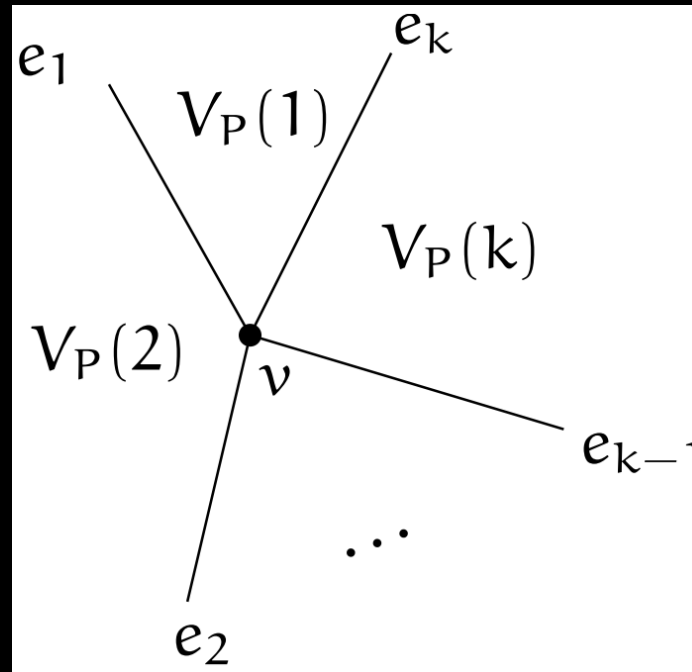
Proof: As all Voronoi cells are convex, each interior angle is less than π , thus $k \geq 3$ of them must be incident to v .



Example: The Voronoi diagram of a point set.

Lemma 1

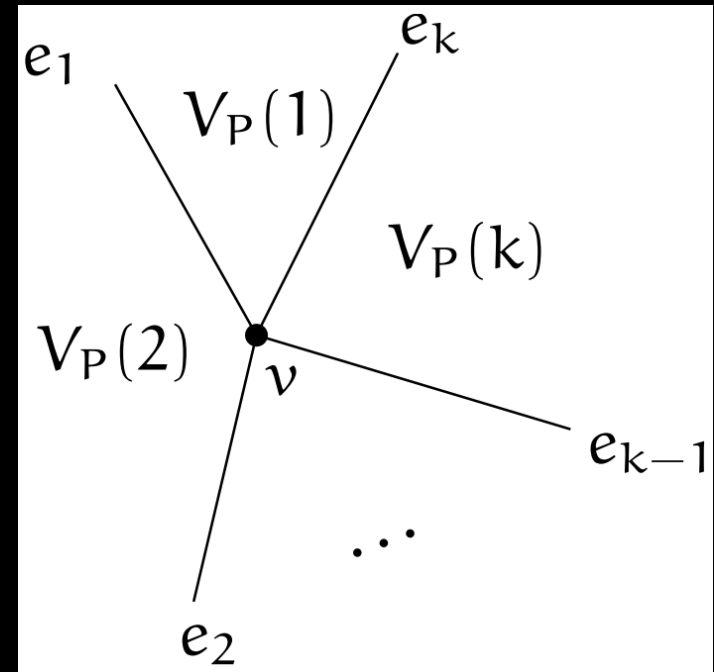
- For every vertex $v \in VV(P)$ the following statements hold.
 - 1) v is the common intersection of at least three edges from $VE(P)$;
 - 2) v is incident to at least three regions from $VR(P)$;
 - 3) v is the center of a circle $C(v)$ through at least three points **from P** ;



Lemma 1

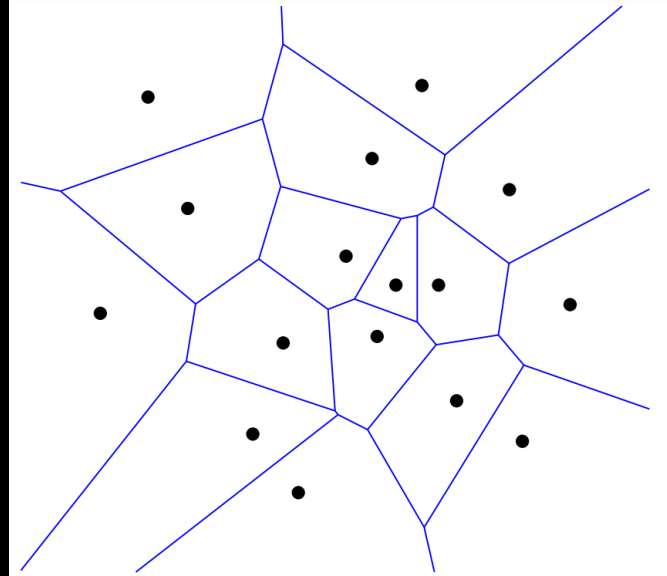
- For every vertex $v \in VV(P)$ the following statements hold.
 - 1) v is the common intersection of at least three edges from $VE(P)$;
 - 2) v is incident to at least three regions from $VR(P)$;
 - 3) v is the center of a circle $C(v)$ through at least three points from P ;
 - 4) $C(v)^\circ \cap P = \emptyset$. $C(v)^\circ$: The interior of $C(v)$.

Suppose there exists a point $p_l \in C(v)^\circ$.
Then the vertex v is closer to p_l than it is to any of p_1, \dots, p_k , in contradiction to the fact that v is contained in all of $VP(1), \dots, VP(k)$.

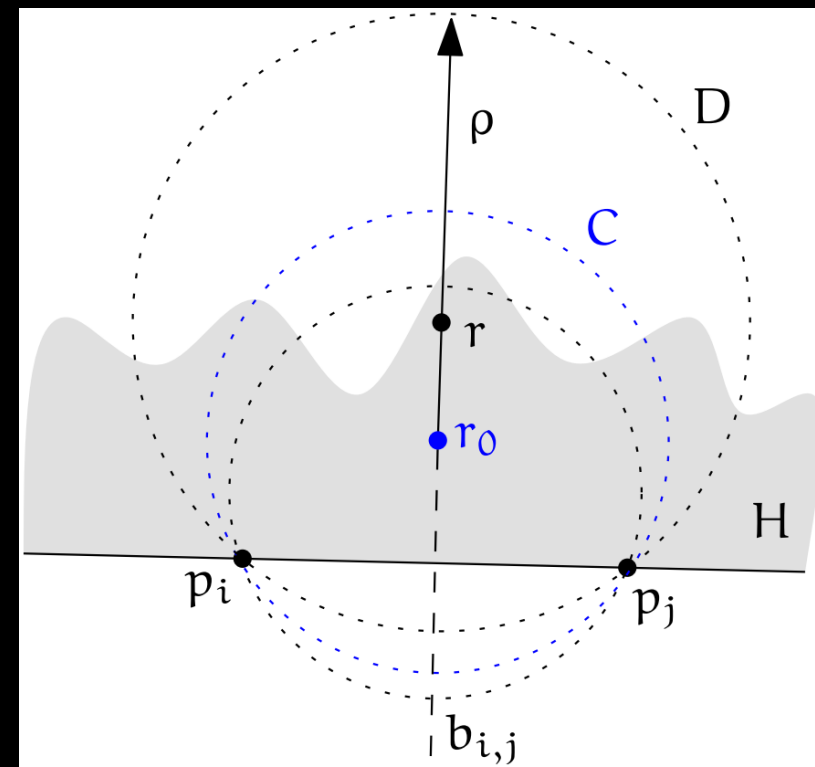


Lemma 2

- There is an unbounded Voronoi edge bounding $VP(i)$ and $VP(j) \iff \overline{p_i p_j} \cap P = \{p_i, p_j\}$ and $\overline{p_i p_j} \in \partial \text{conv}(P)$, where the latter denotes the boundary of the convex hull of P .
- Proof: There is an unbounded Voronoi edge bounding $VP(i)$ and $VP(j) \iff$ there is a ray $\rho \subset b_{i,j}$ such that $\|r - p_k\| > \|r - p_i\| (= \|r - p_j\|), \forall r \in \rho$ and $p_k \in P \setminus \{p_i, p_j\}$. **Equivalently**, there is a ray $\rho \subset b_{i,j}$ such that for every point $r \in \rho$ the circle $C \in D$ centered at r does not contain any point from P in its interior.



Example: The Voronoi diagram of a point set.

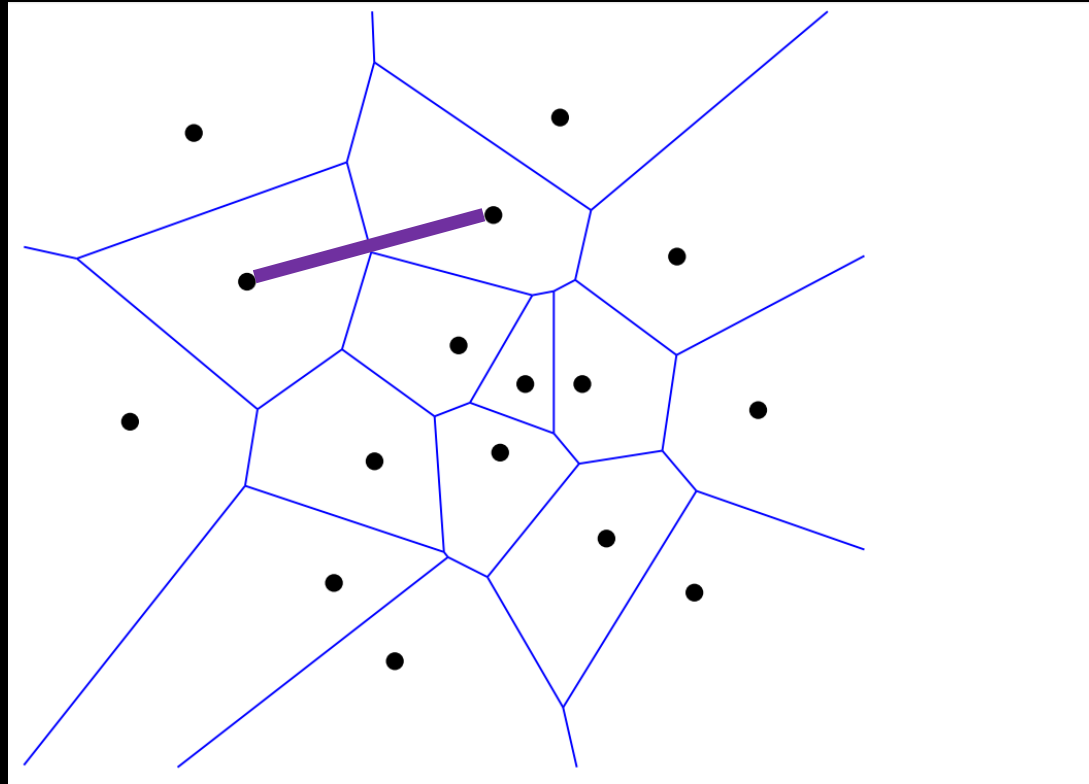


Outlines

- Introduction
 - Post Office Problem
 - Voronoi Diagram
- Duality: Delaunay triangulation
- Centroidal Voronoi tessellations (CVT)
 - Definition
 - Applications
 - Algorithms

Duality

- A **straight-line dual** of a plane graph G is a graph G' defined as follows: **choose a point for each face** of G and connect any two such points by a straight edge, if the corresponding faces share an edge of G .

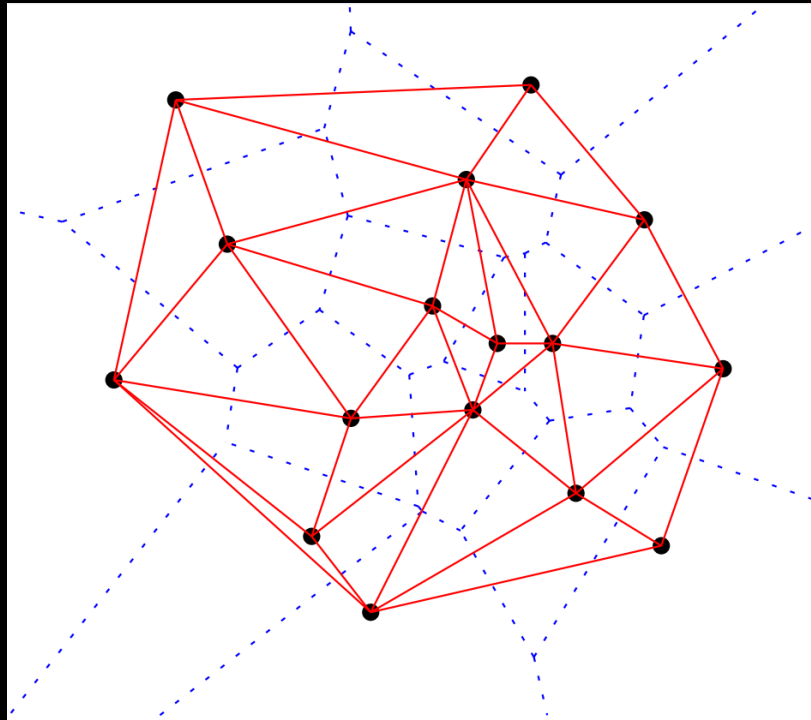


Delaunay triangulation

- Theorem: The **straight-line dual** of $VD(P)$ for a set $P \subset R^2$ of $n > 3$ points in general position (no three points from P are collinear and no four points from P are cocircular) is a triangulation: the unique Delaunay triangulation of P .

Proof: \implies

1. convex hull
2. Triangles
3. Empty circle property



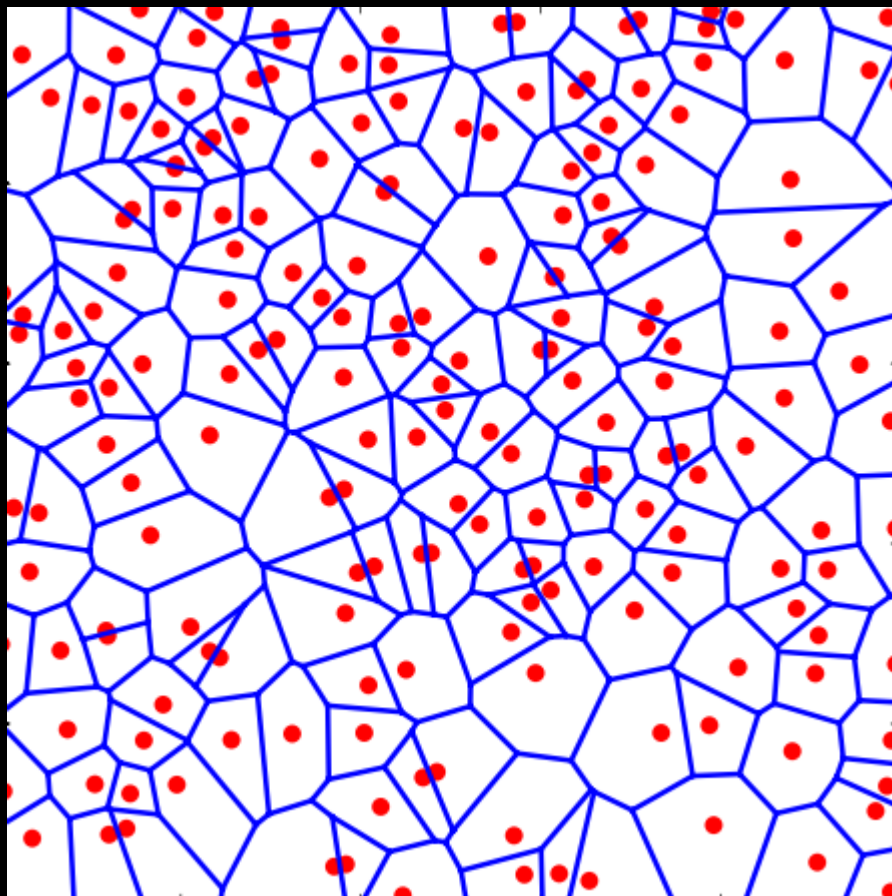
Proof: \impliedby

1. Circumcenter is selected for each face.
2. Empty circle property.

Outlines

- Introduction
 - Post Office Problem
 - Voronoi Diagram
- Duality: Delaunay triangulation
- Centroidal Voronoi tessellations (CVT)
 - Definition
 - Applications
 - Algorithms

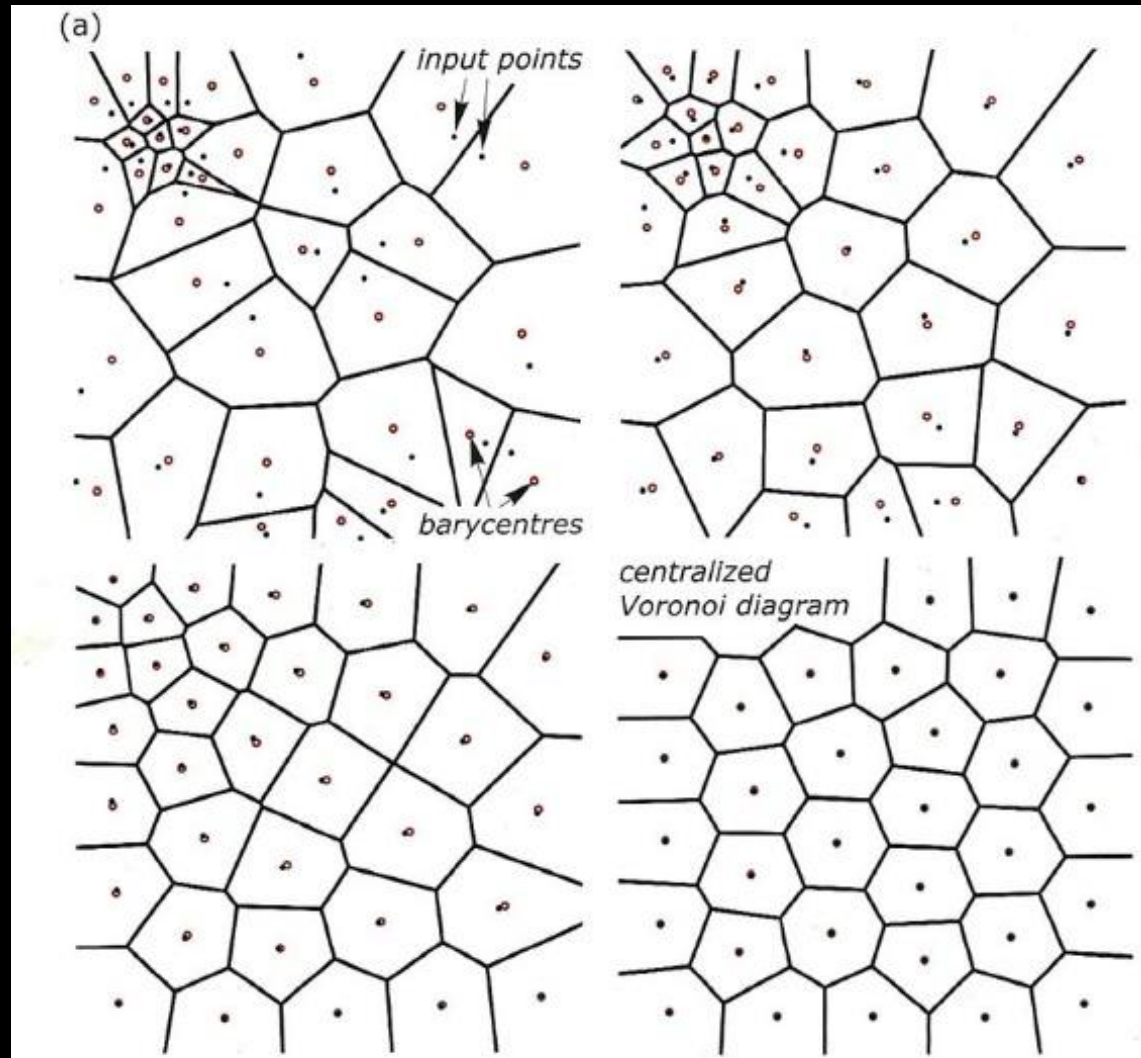
Problem



Update vertices



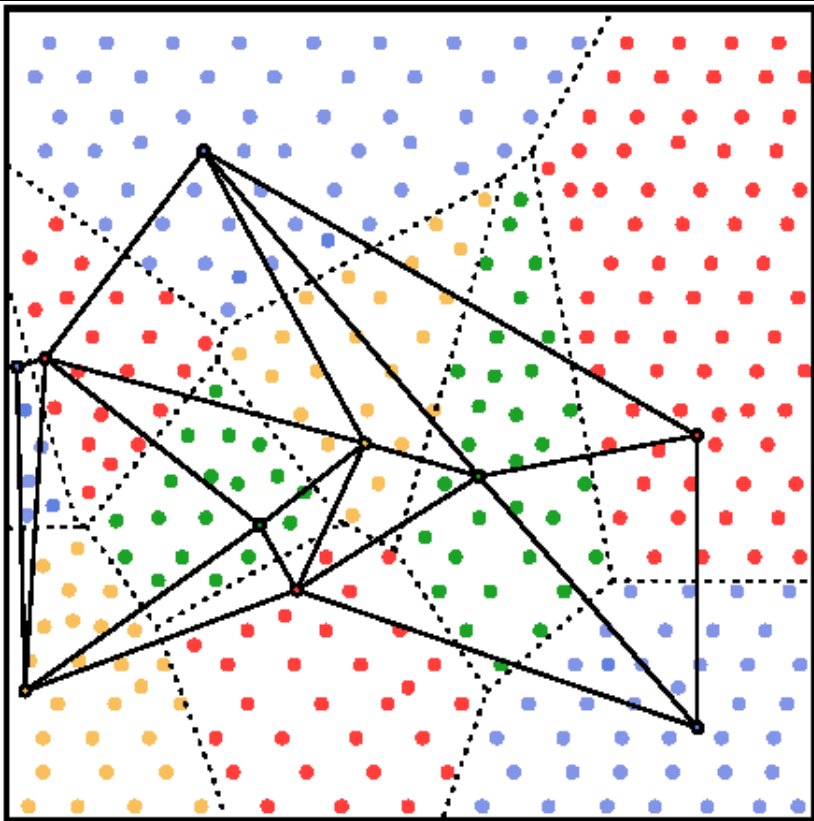
Definition – CVT



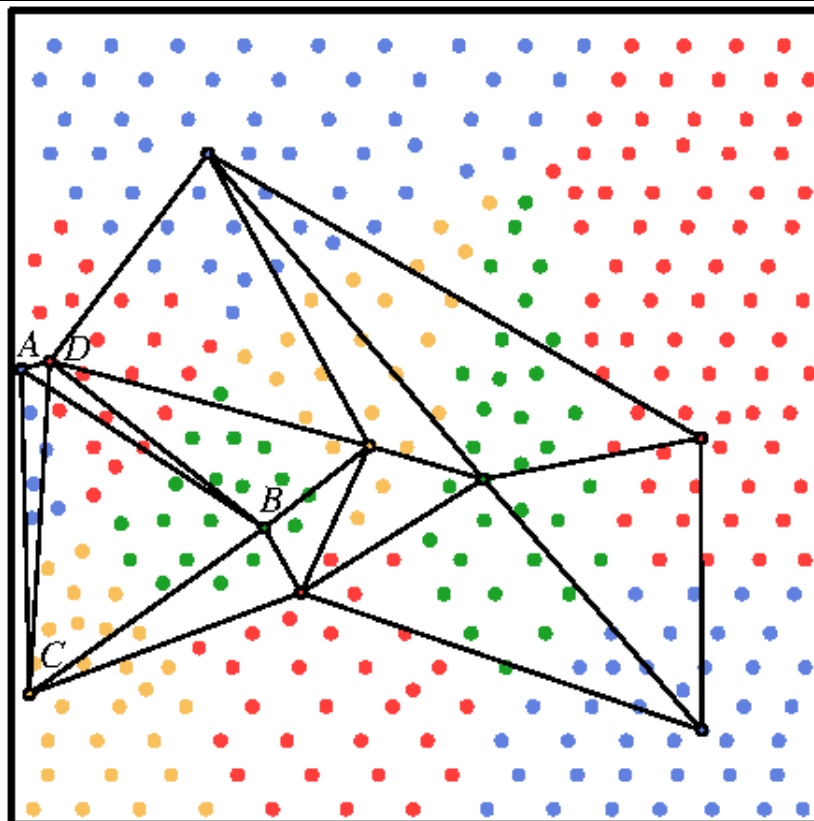
A class of Voronoi tessellations where each site **coincides** with the centroid (i.e., center of mass) of its Voronoi region.

$$c_i = \frac{\int_{V_i} x \rho(x) dx}{\int_{V_i} \rho(x) dx}$$

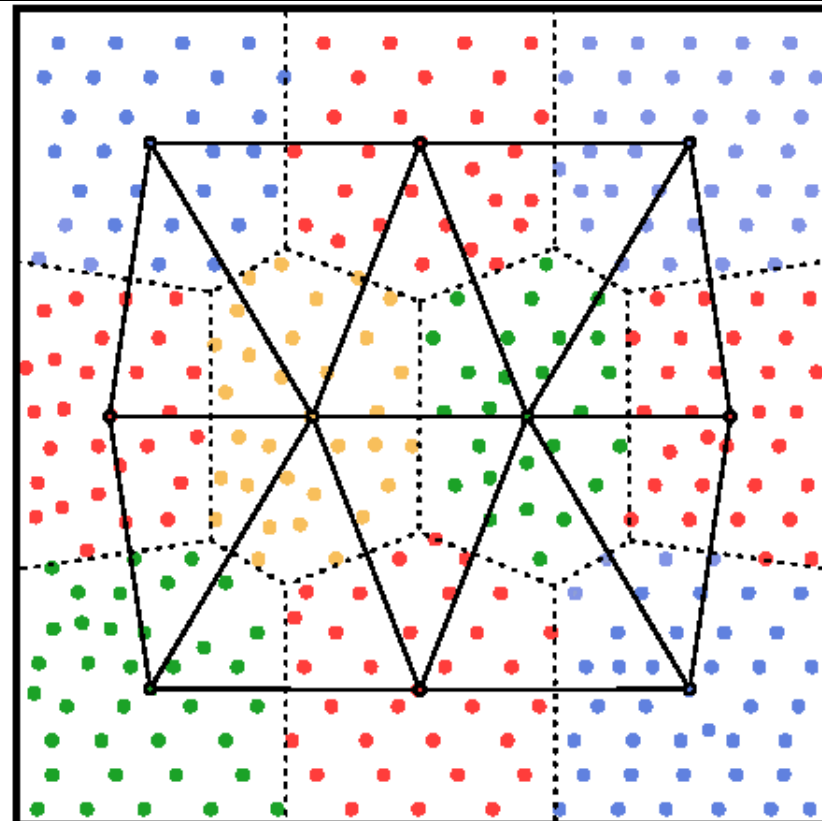
Applications – Remeshing



(a)



(b)



(c)

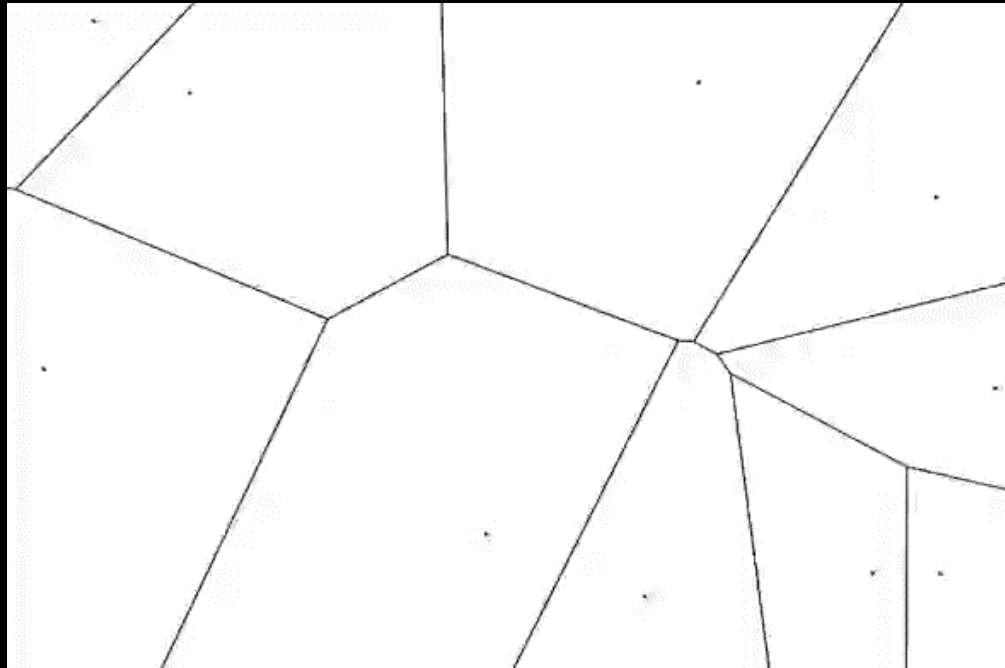
Energy function

$$E(p_1, \dots, p_n, V_1, \dots, V_n) = \sum_{i=1}^n \int_{V_i} \|x - p_i\|^2 dx$$

1. For a fixed set of sites $P = \{p_1, \dots, p_n\}$, the energy function is minimized if $\{V_1, \dots, V_n\}$ is a Voronoi tessellation.
2. For the fixed regions, the p_i are the mass centroids c_i of their corresponding regions V_i .

Lloyd iteration

- 1. Construct the Voronoi tessellation corresponding to the sites p_i .
- 2. Compute the centroids c_i of the Voronoi regions V_i and move the sites p_i to their respective centroids c_i .
- 3. Repeat steps 1 and 2 until satisfactory convergence is achieved.



Remeshing

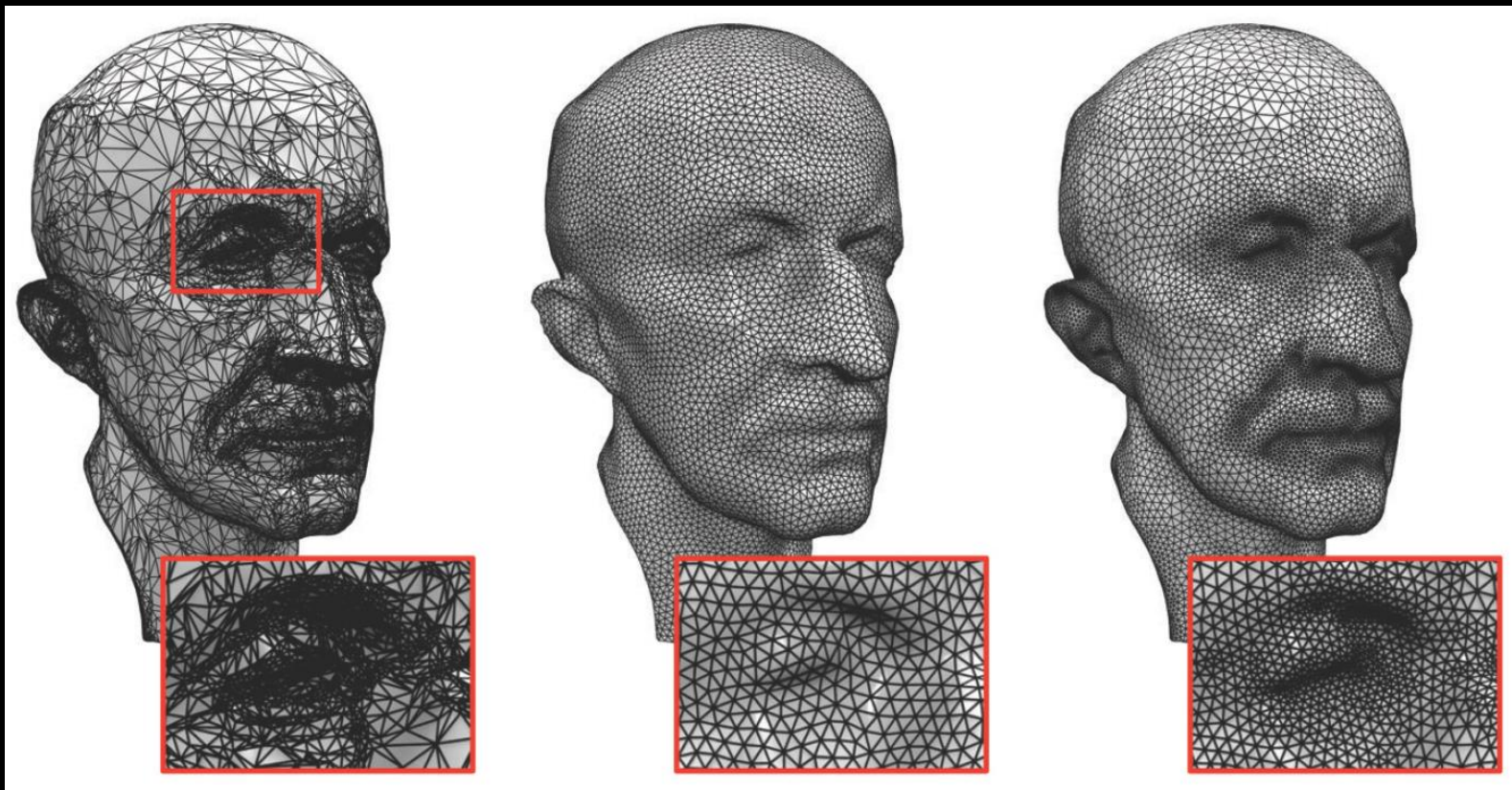
Xiao-Ming Fu

Outlines

- Isotropic remeshing
 - Error-bounded method
 - Error-Bounded and Feature Preserving Surface Remeshing with Minimal Angle Improvement
 - Improve small and large angles
 - Optimal Delaunay Triangulation (ODT)
 - Centroidal Patch Triangulation (CPT)
- Anisotropic remeshing
 - Introduction
 - ODT with general convex function
 - Local convex triangulation
 - Partial-based method
 - High-dim embedding

Remeshing

- Given a 3D mesh, compute another mesh, whose elements satisfy some quality requirements, while **approximating** the input acceptably.



Remeshing

- A key technique for mesh quality improvement
- Goal
 - 1. **Reduce the complexity of an input surface mesh**
 - subject to certain quality criteria
 - 2. **Improve the quality of a mesh**
 - Different applications imply different quality criteria and requirements.
- **Given a 3D mesh, compute another mesh, whose elements satisfy some quality requirements, while **approximating** the input acceptably.**

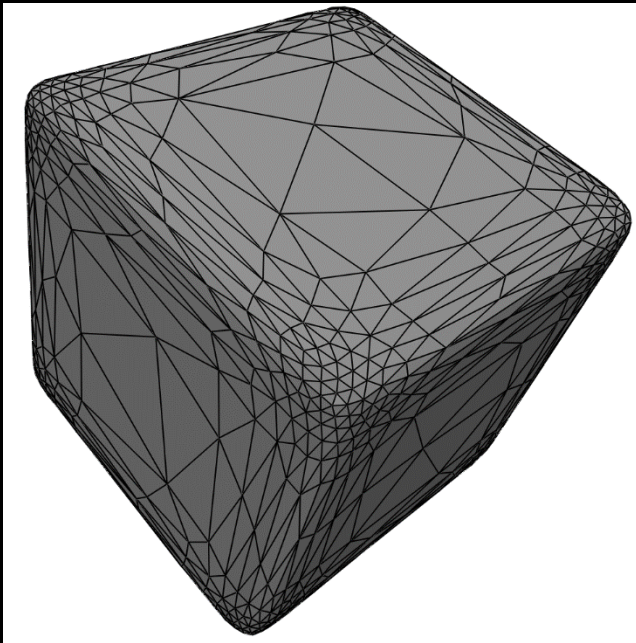
Discussion

- Input: a manifold triangle mesh or part of it.
- Mesh quality
 - Sampling density
 - Regularity
 - Size
 - Orientation
 - Alignment
 - Shape of the mesh elements.
 - Non-topological issues (mesh repair)

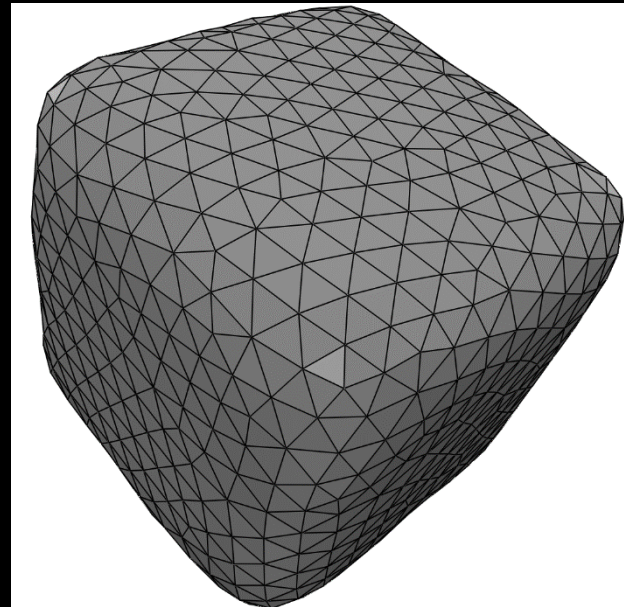
Local Structure

- Element shape
 - Isometric
 - Anisotropic

anisotropic

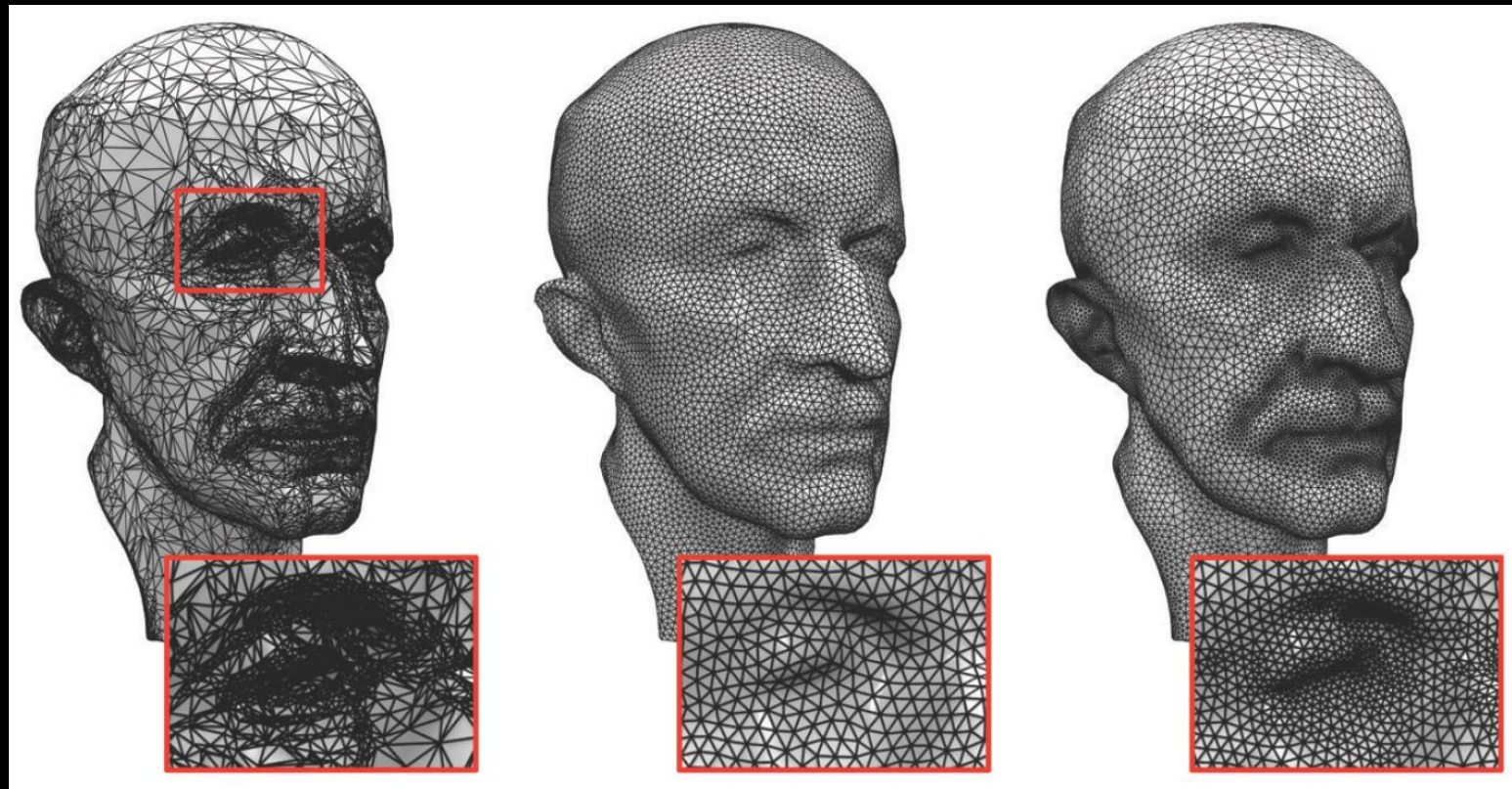


isotropic



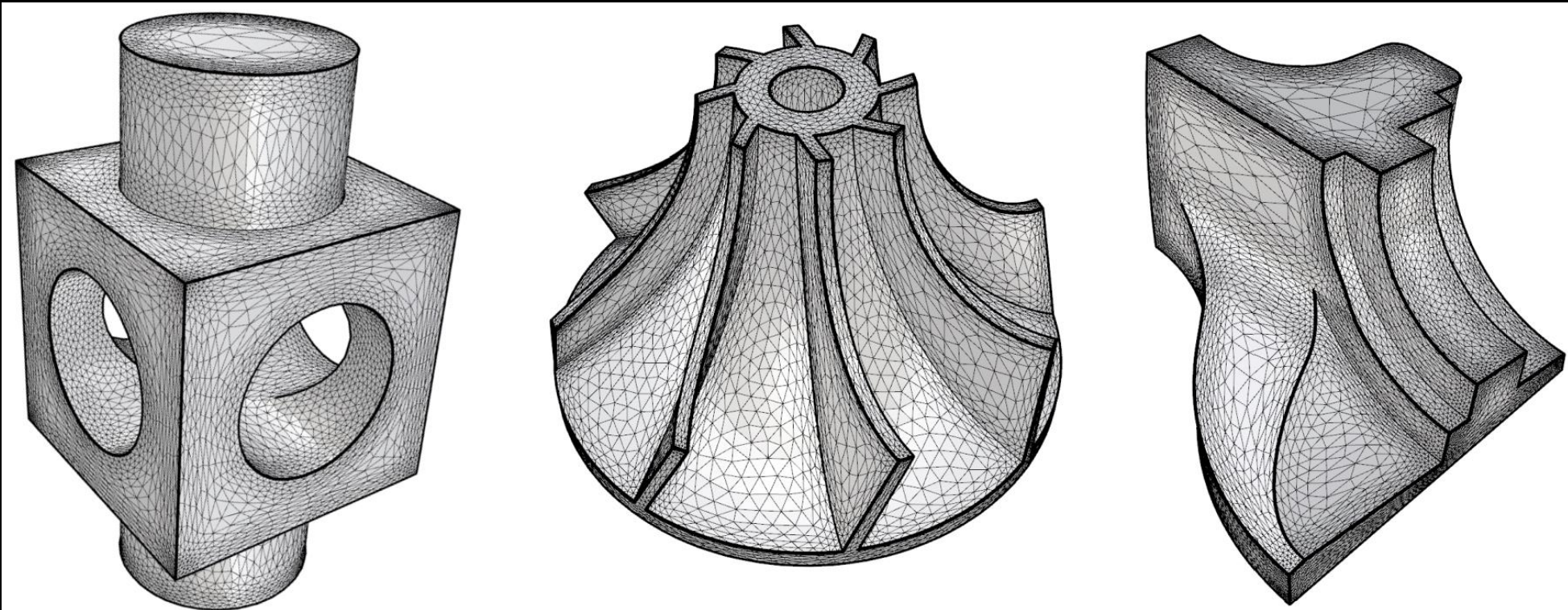
Local Structure

- Element density
 - uniform VS. nonuniform or adaptive



Local Structure

- Element alignment and orientation
 - elements should align to sharp features
 - orientation of anisotropic elements



Global structure

- Vertex
 - Regular
 - Valence = 6 for triangle mesh
 - Valence = 4 for quad mesh
 - Irregular (singular)
- Global
 - Irregular
 - Semiregular
 - regular subdivision of a coarse initial mesh
 - Highly regular
 - most vertices are regular
 - Regular
 - all vertices are regular

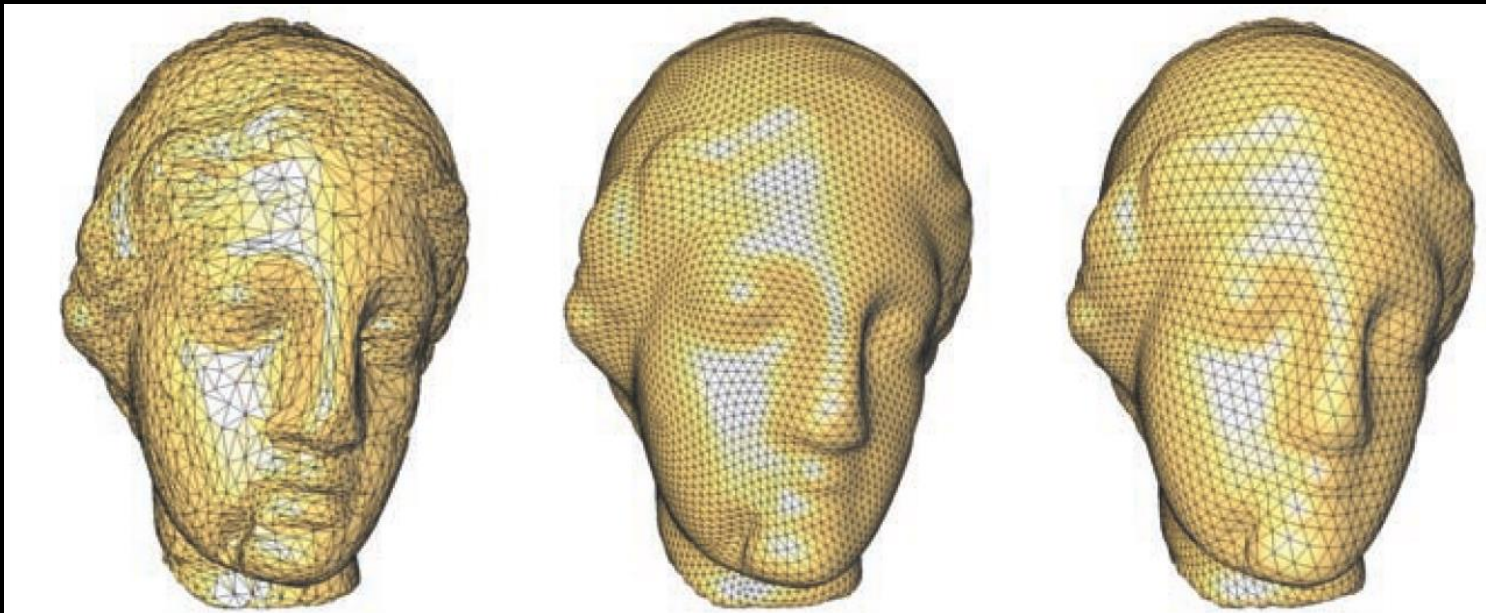


Figure 6.2. Meshes: Irregular (left), semiregular (center), and regular (right). (Model courtesy of Cyberware.)

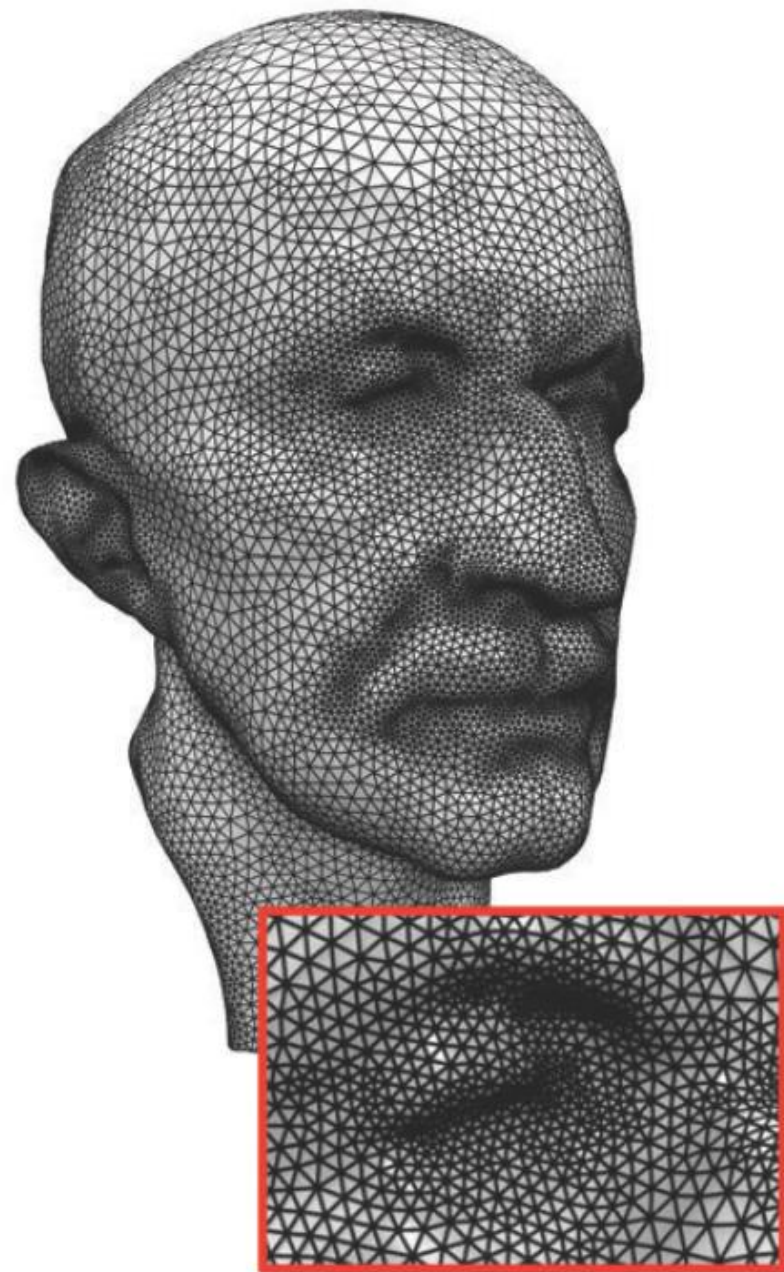
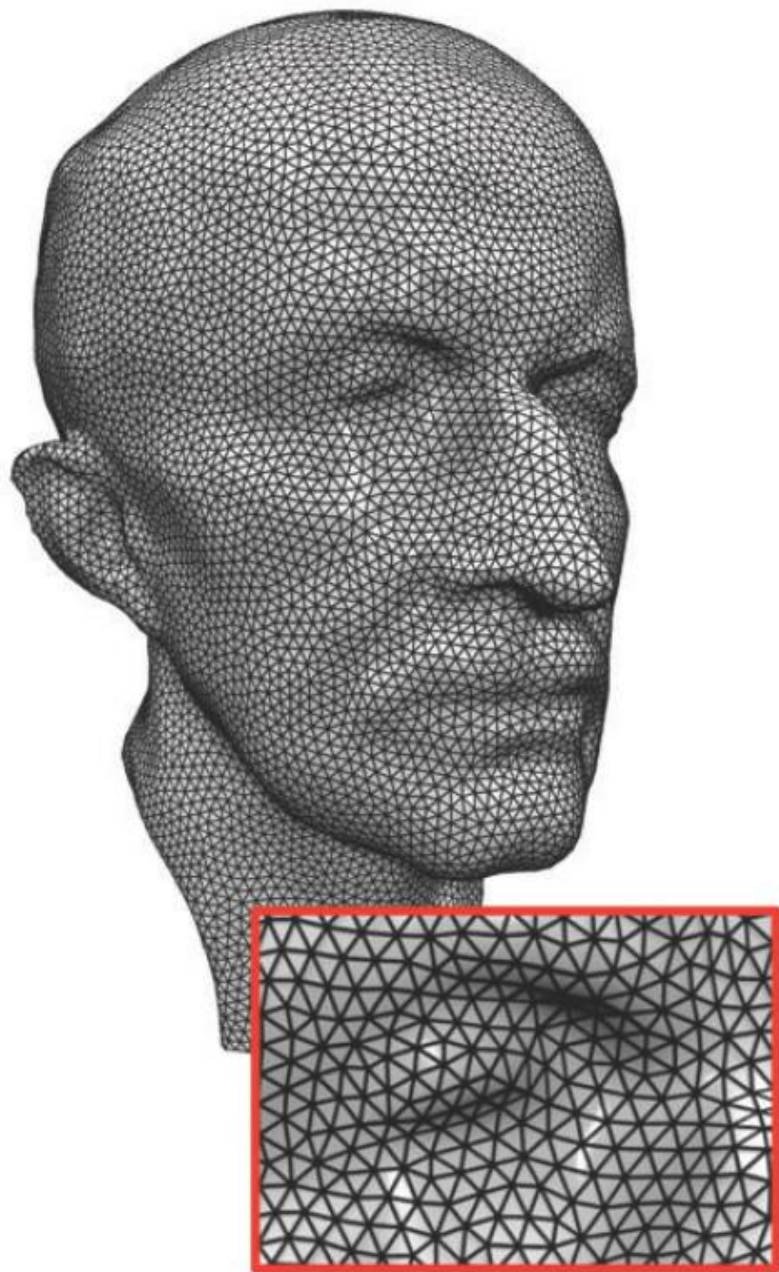
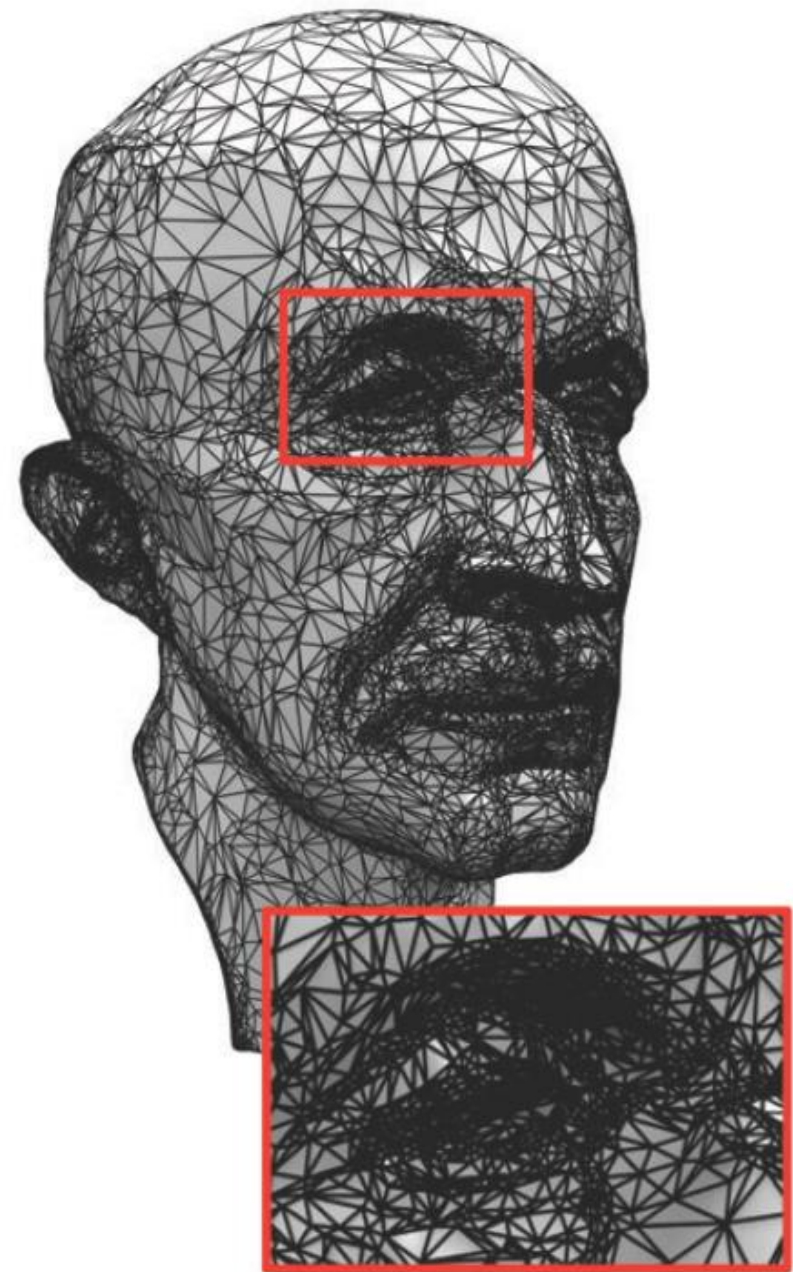
Outlines

- **Isotropic remeshing**

- Error-bounded method
 - Error-Bounded and Feature Preserving Surface Remeshing with Minimal Angle Improvement
- Improve small and large angles
- Optimal Delaunay Triangulation (ODT)
- Centroidal Patch Triangulation (CPT)

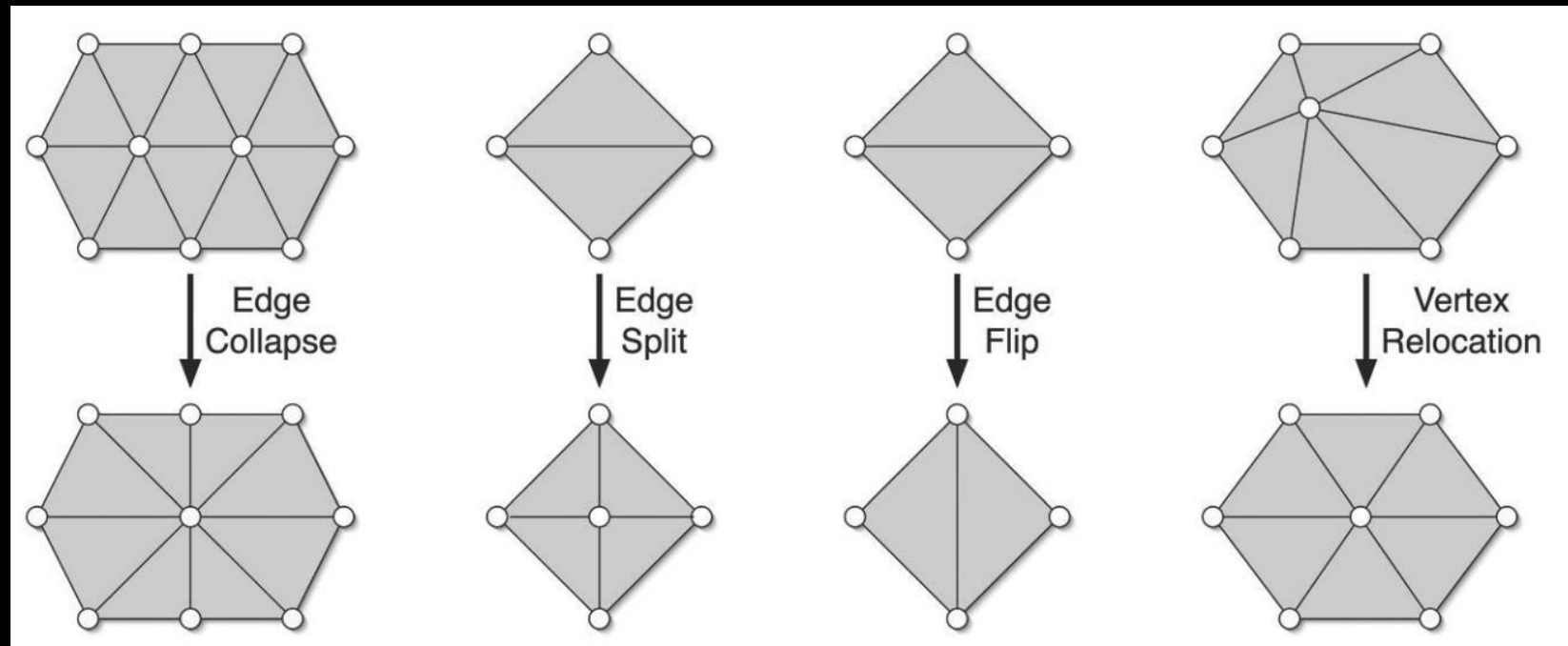
- **Anisotropic remeshing**

- Introduction
- ODT with general convex function
- Local convex triangulation
- Partial-based method
- High-dim embedding



Incremental Remeshing

- Input: triangle mesh and a target edge length
- Method
 - **Split long edges**
 - **Collapse short edges**
 - **Relocate vertices**



Pseudo-code

Remesh(target edge length)

$Low_e = \frac{4}{5} * \text{target edge length}$

$High_e = \frac{4}{3} * \text{target edge length}$

for i = 0 to 10 do

 split long edges(high_e)

 collapse short edges(low_e, high_e)

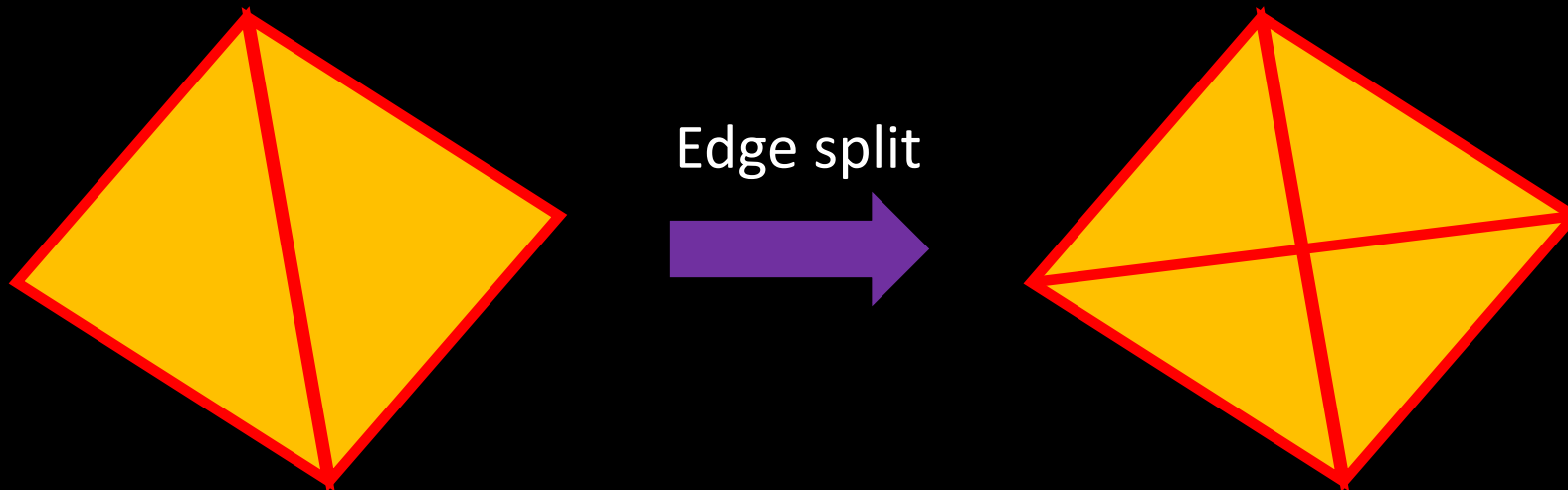
 equalize valences()

 tangential relaxation()

 project to surface()

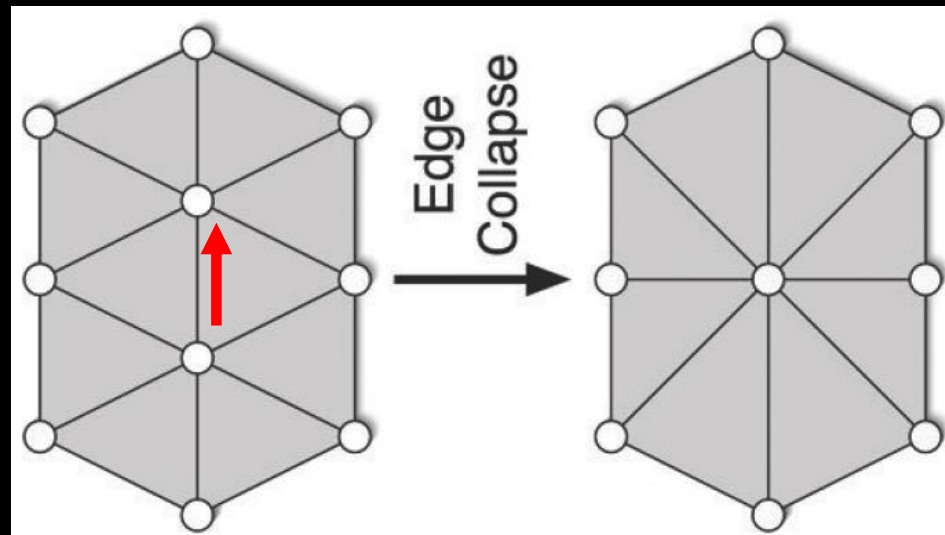
Discussion

- Proper threshold $\frac{4}{5}$ and $\frac{4}{3}$ are essential.
- split long edges(high_e)
 - visits all edges of the current mesh
 - If an edge is longer than the given threshold high_e, the edge is split at its midpoint and the two adjacent triangles are bisected



Discussion

- collapse short edges(low_e , $high_e$)
 - collapses and thus removes all edges that are shorter than a threshold low_e .
- Issue
 - by collapsing along chains of short edges, the algorithm may create new edges that are arbitrarily long.
 - This issue is resolved by testing before each collapse whether the collapse would produce an edge that is longer than $high$.
 - If so, the collapse is not executed.



Discussion

- equalize valences()
 - equalizes the vertex valences by flipping edges.
 - The algorithm tentatively flips each edge and checks whether the deviation to the target valences decreases.
- The tangential relaxation()
 - an iterative smoothing filter to the mesh
 - the vertex movement has to be constrained to the vertex tangent plane in order to stabilize the following projection operator.

Discussion

- The tangential relaxation(): uniform Laplacian weights

$$q = \frac{1}{N_p} \sum_{p_j \in \Omega(p)} p_j$$

projecting q onto p 's tangent plane:

$$p' = q + nn^t(p - q)$$

- **Project to surface()**

- maps the vertices back to the surface.
- **CGAL: AABB tree**

Adaptive edge length

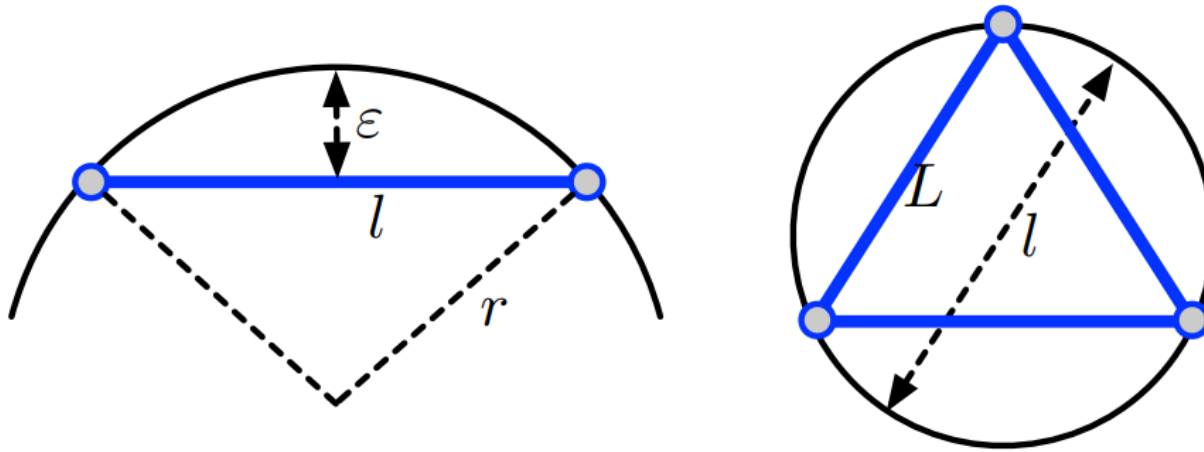


Figure 1: Left: How to determine the edge length l for approximating a circular arc up to an error tolerance ε . Right: For surfaces the edge length l has to be scaled to yield the target edge length L for an equilateral triangle. In the surface case, the left and right figures can be considered a cross-section or a top view of each other.

- $l = 2\sqrt{2r\varepsilon - \varepsilon^2}$
 - $L(x_i) = \sqrt{\frac{6\varepsilon}{k_i} - 3\varepsilon^2}$
- k_i : maximum absolute curvature at x_i

Discussion

- Feature preservation.
 - feature edges and vertices have already been marked in the input model.
 - 1. Corner vertices with **more than two or exactly one incident feature edge** have to be preserved and are excluded from all topological and geometric operations.
 - 2. Feature vertices may only be collapsed **along their incident feature edges**.
 - 3. Splitting a feature edge creates two new feature edges and a feature vertex.
 - 4. **Feature edges are never flipped.**
 - 5. Tangential smoothing of feature vertices is restricted to univariate smoothing along the corresponding feature lines. **(How to do???)**

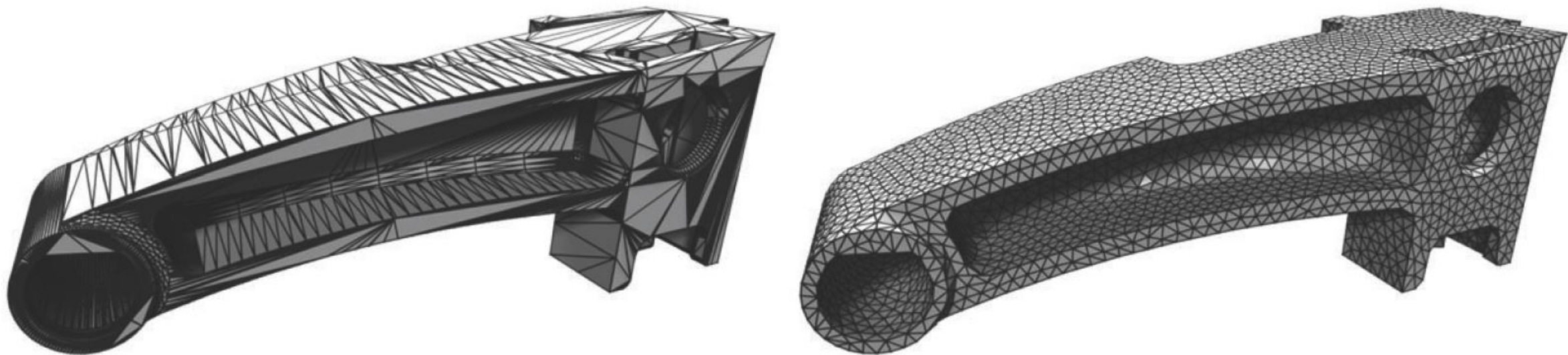
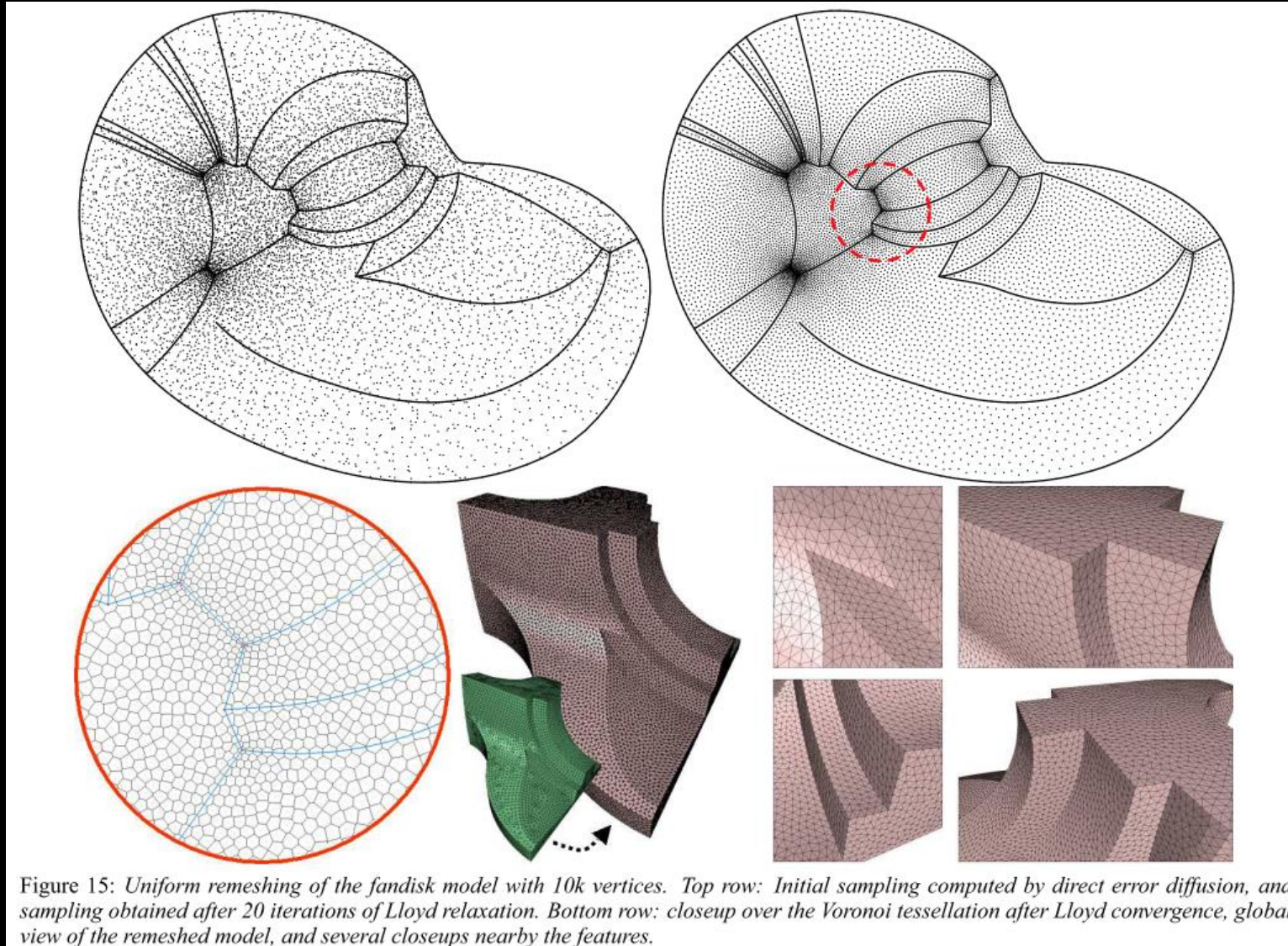


Figure 6.13. Isotropic, feature-sensitive remeshing (right) of a CAD model (left).

Parameterization-based method

- All remeshing algorithms compute point locations on or near the original surface.
- Global parameterization.
 - The input model is globally parameterized onto a 2D domain.
 - Sample points can then be easily distributed and relocated in the 2D domain.
 - Be lifted to three dimensions
 - Disadvantages:
 - Parametric distortion
 - Discontinuities when the mesh needs to be cut into a topological disk.

Global parameterization



Closed surface

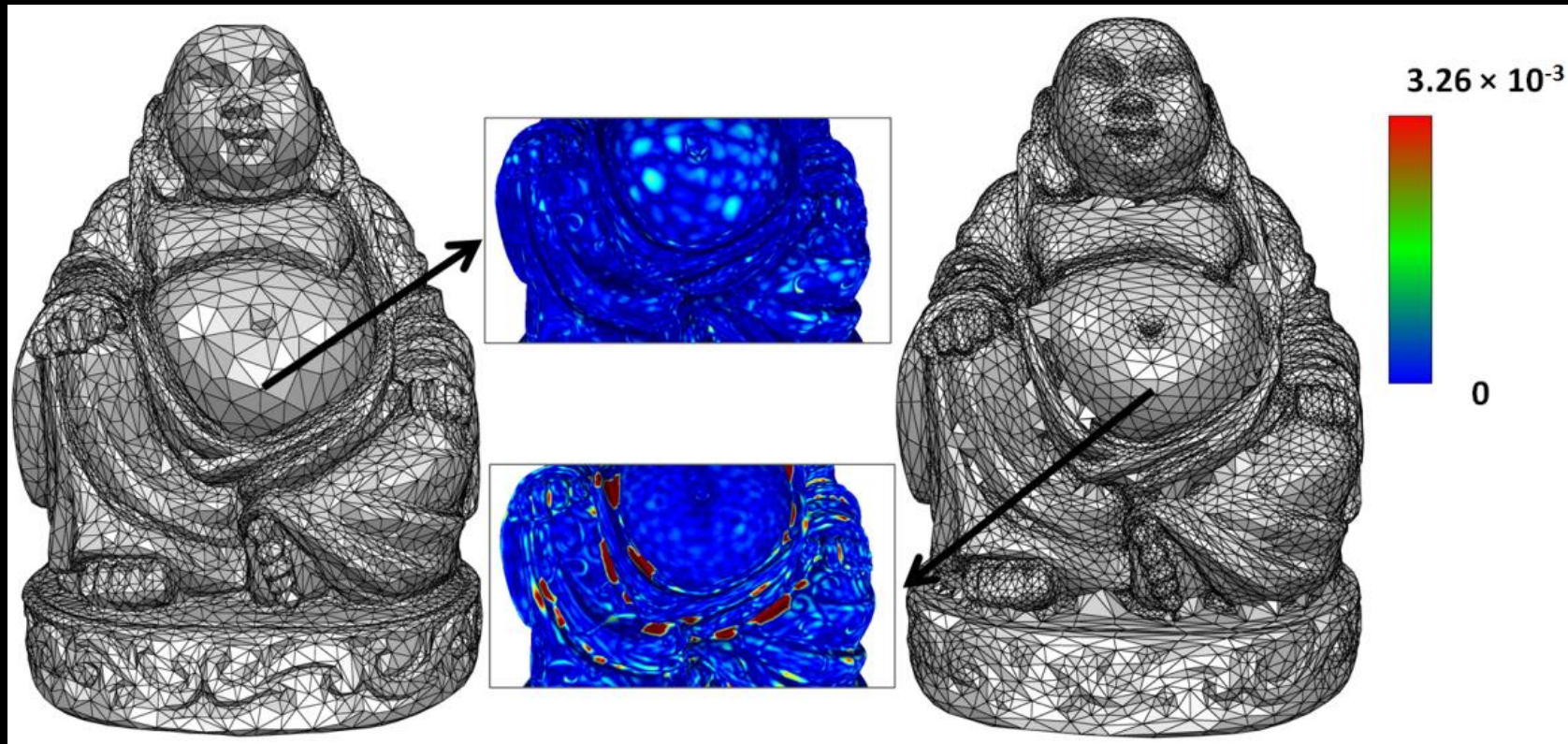
- Parameterization-based method requires cut paths.
- How to generate the cut paths:
 - Distortion, D-Charts
 - Visit at least twice

Outlines

- Isotropic remeshing
 - **Error-bounded method**
 - Error-Bounded and Feature Preserving Surface Remeshing with Minimal Angle Improvement
 - Improve small and large angles
 - Optimal Delaunay Triangulation (ODT)
 - Centroidal Patch Triangulation (CPT)
- Anisotropic remeshing
 - Introduction
 - ODT with general convex function
 - Local convex triangulation
 - Partial-based method
 - High-dim embedding

Approximation

- Definition:
 - Given a 3D mesh, compute another mesh, whose elements satisfy some quality requirements, while **approximating** the input acceptably.



Hausdorff distance

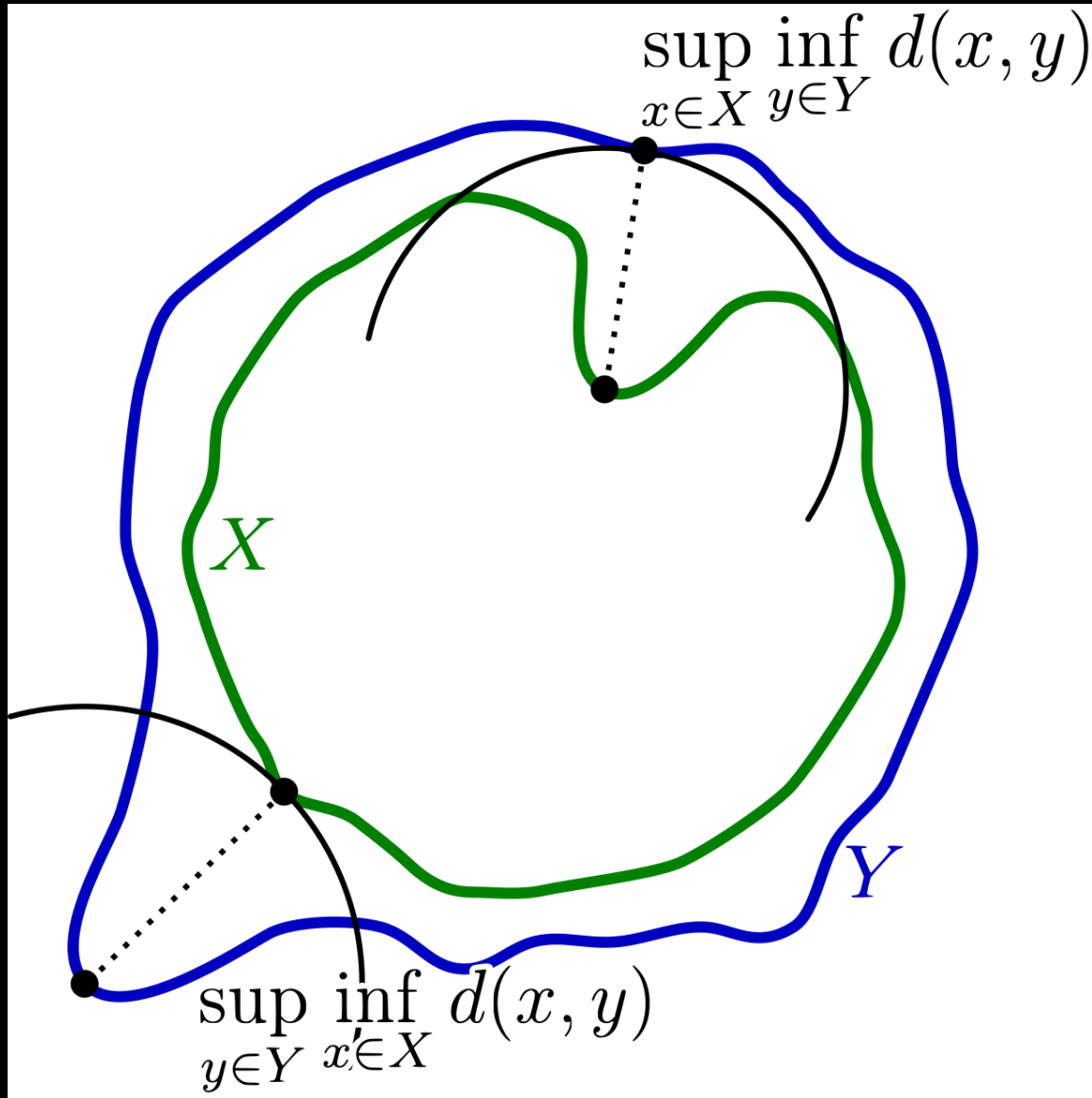
https://en.wikipedia.org/wiki/Hausdorff_distance

- **Hausdorff distance** measures **how far** two subsets of a metric space are from each other.
- Let X and Y be two non-empty subsets of a metric space (M, d) . We define their Hausdorff distance $d_H(X, Y)$ by

$$d_H(X, Y) = \max \left\{ \sup_{x \in X} \inf_{y \in Y} d(x, y), \sup_{y \in Y} \inf_{x \in X} d(x, y) \right\}$$

where sup represents the **supremum** and inf the **infimum**.

Hausdorff distance



In general:

$$\sup_{x \in X} \inf_{y \in Y} d(x, y) \neq \sup_{y \in Y} \inf_{x \in X} d(x, y)$$

Hausdorff distance on triangular mesh

- Hausdorff distance between M_1 and M_2 :

$$d_H(M_1, M_2) = \max\{d_h(M_1, M_2), d_h(M_2, M_1)\}$$

where

$$d_h(S, T) = \max_{p \in S} \min_{q \in T} d(p, q)$$

Note: $d(p, T) = \min_{q \in T} d(p, q)$ is the distance from p to T .

The distance of a point p to a surface T is defined as the shortest distance between p and **any point** of T .

Approximating $d_H(M_1, M_2)$

- Assume that M_1 is sampled by a point set $S_1 \subset M_1$ and M_2 is sampled by a point set $S_2 \subset M_2$

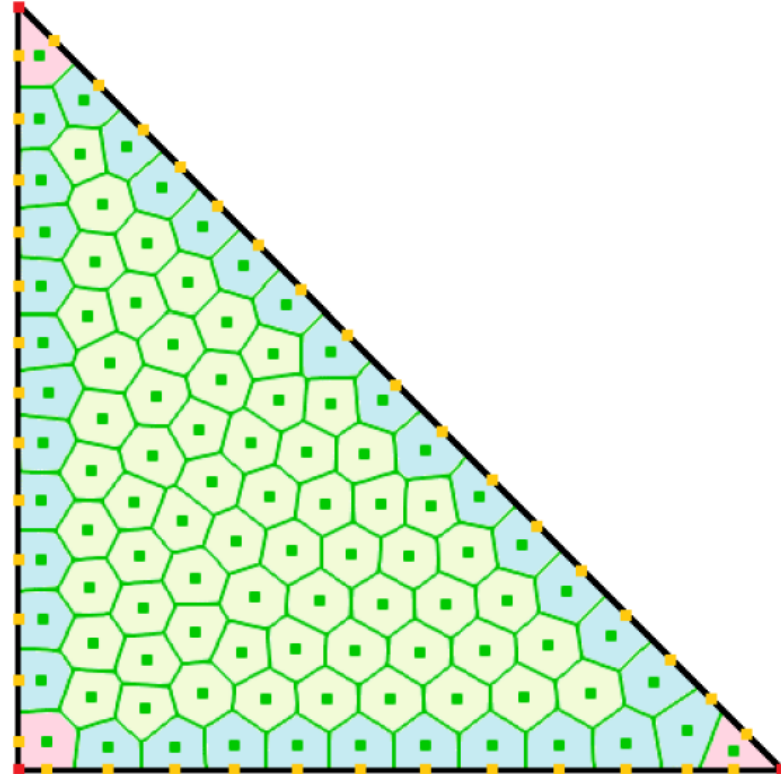
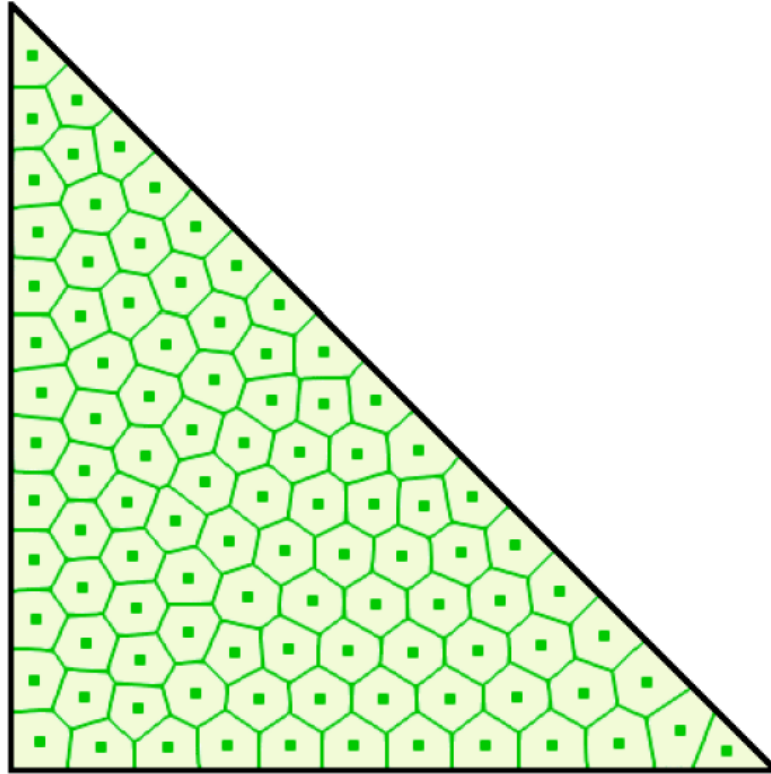
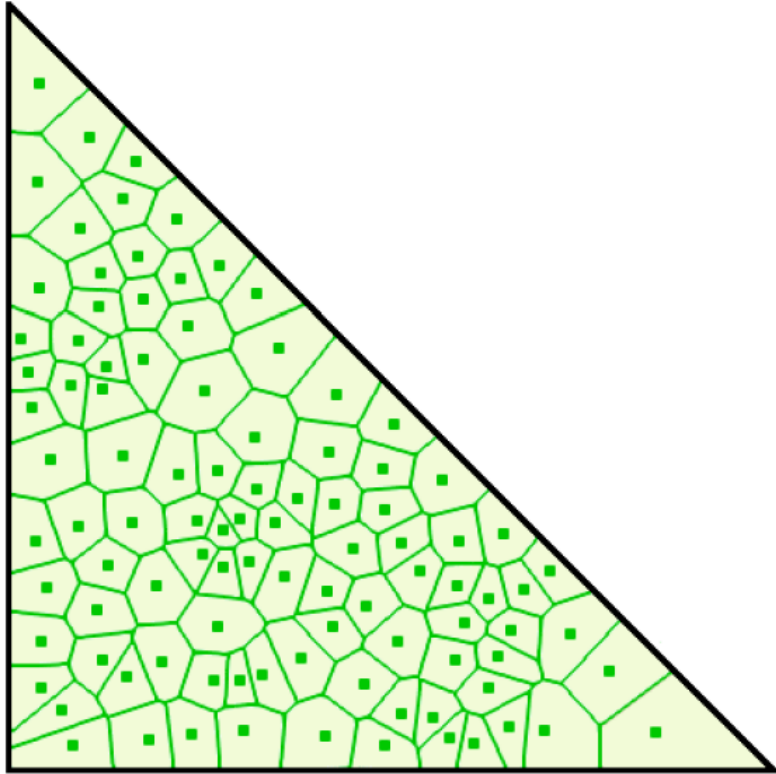
- $d_h(M_1, M_2)$ can be approximated by

$$d_h(M_1, M_2) \approx \max_{a \in S_1} d(a, M_2)$$

- $\Rightarrow d_H(M_1, M_2) = \max\left\{\max_{a \in S_1} d(a, M_2), \max_{b \in S_2} d(b, M_1)\right\}$

- It provides significantly higher accuracy than a point cloud distance $d_H(S_1, S_2)$.

Sampling



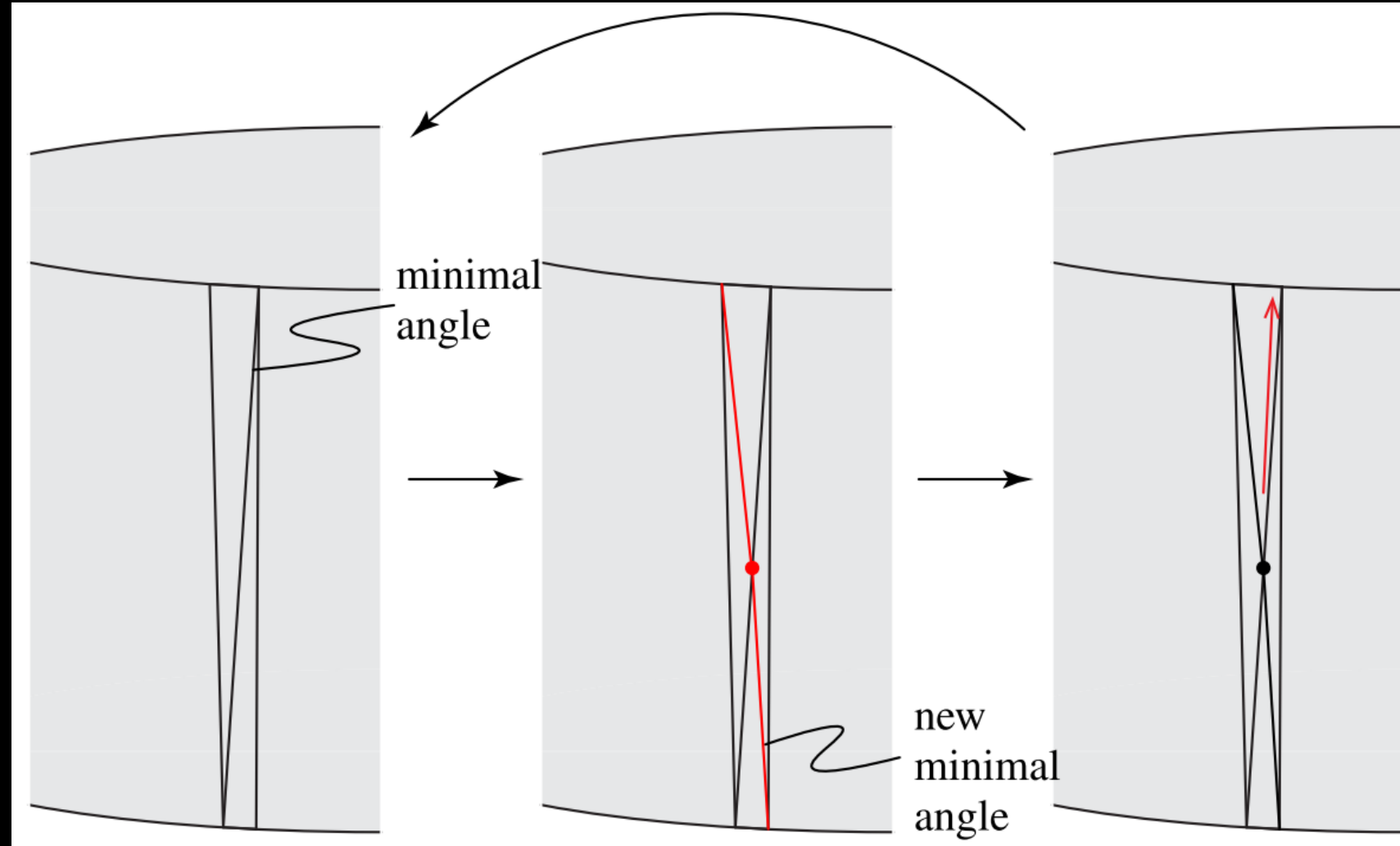
Stratified sampling process

Error-bounded method

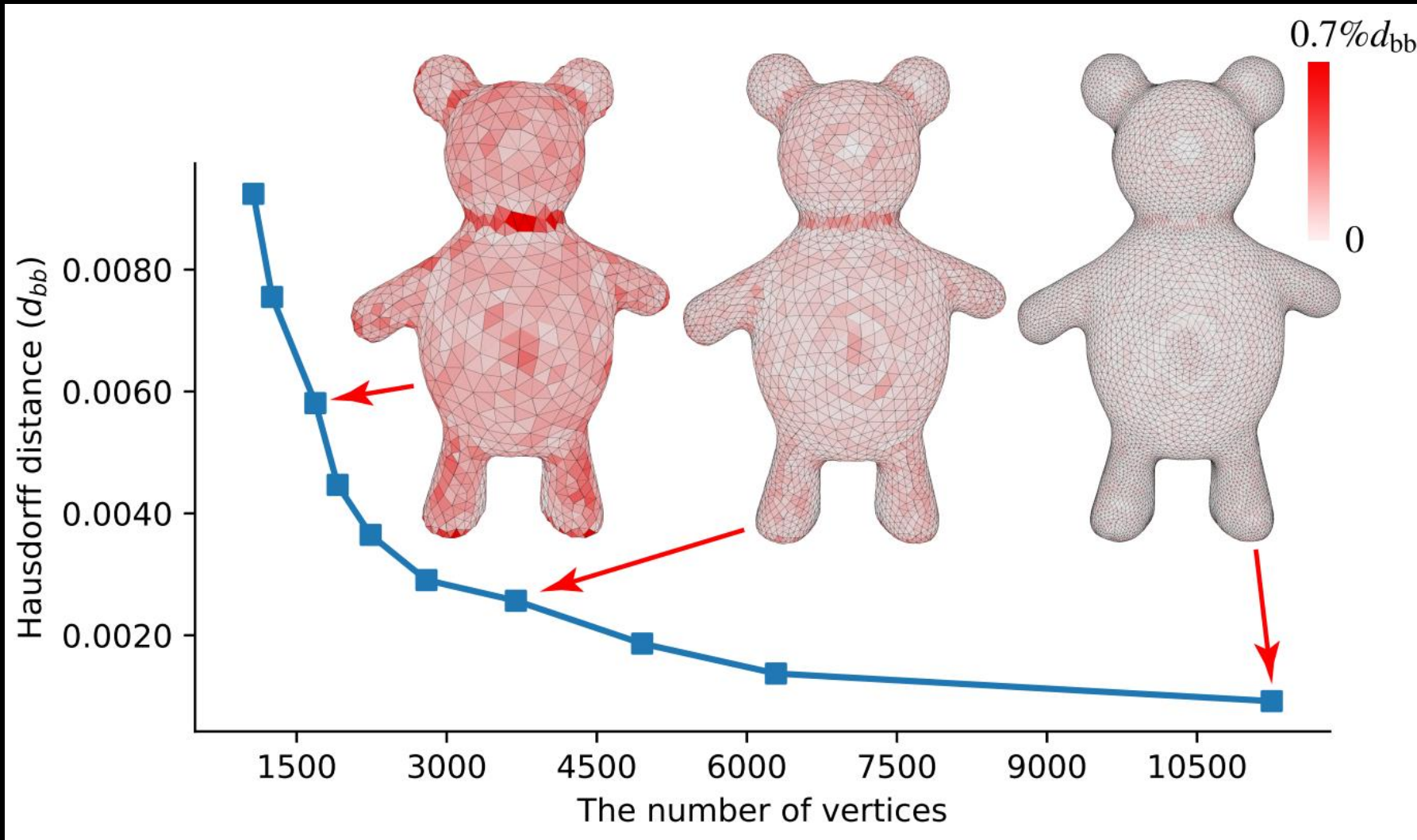
- A local operator is only executed if it respects the approximation error bound.
 - never leaving the feasible set of meshes with approximation error
- A local operator
 - Edge collapse
 - Edge split
 - Edge flip
 - Smoothing / Relocation
 -

Main limitations

- 1. Time consuming
- 2. Infinite loop



Thinking from a different view

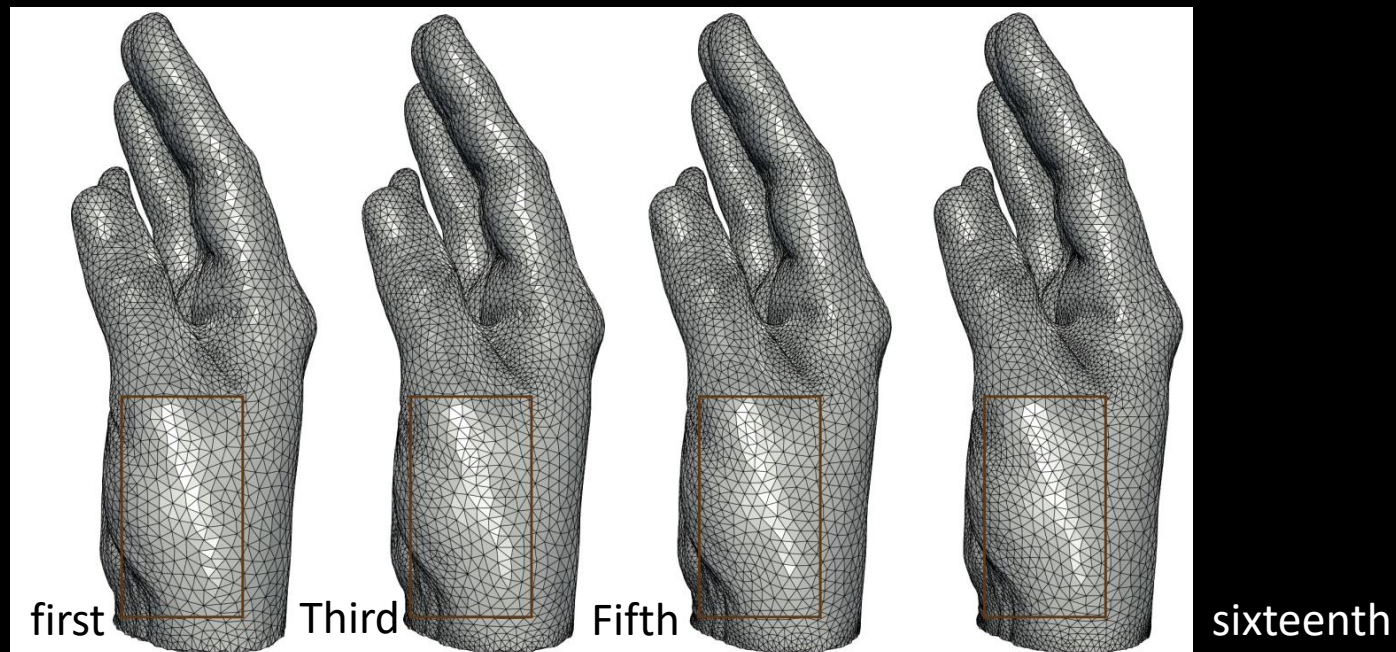


Thinking from a different view

- A key observation: more uniformly distributed vertices are used for remeshing, the Hausdorff distance between the remeshed and input meshes is usually easily bounded.
- During the remeshing process, intermediate meshes are allowed to violate the error-bounded constraint, and attempt to reduce the Hausdorff distance by adding vertices into the mesh.

Algorithm

- 1. Initialize a target edge length field $L(x)$
- 2. Perform edge-based remeshing using $L(x)$
- If $d_h(M_1, M_2) \leq \delta$, stop the algorithm; otherwise, adaptively adjust $L(x)$, and then go to step (2).



Outlines

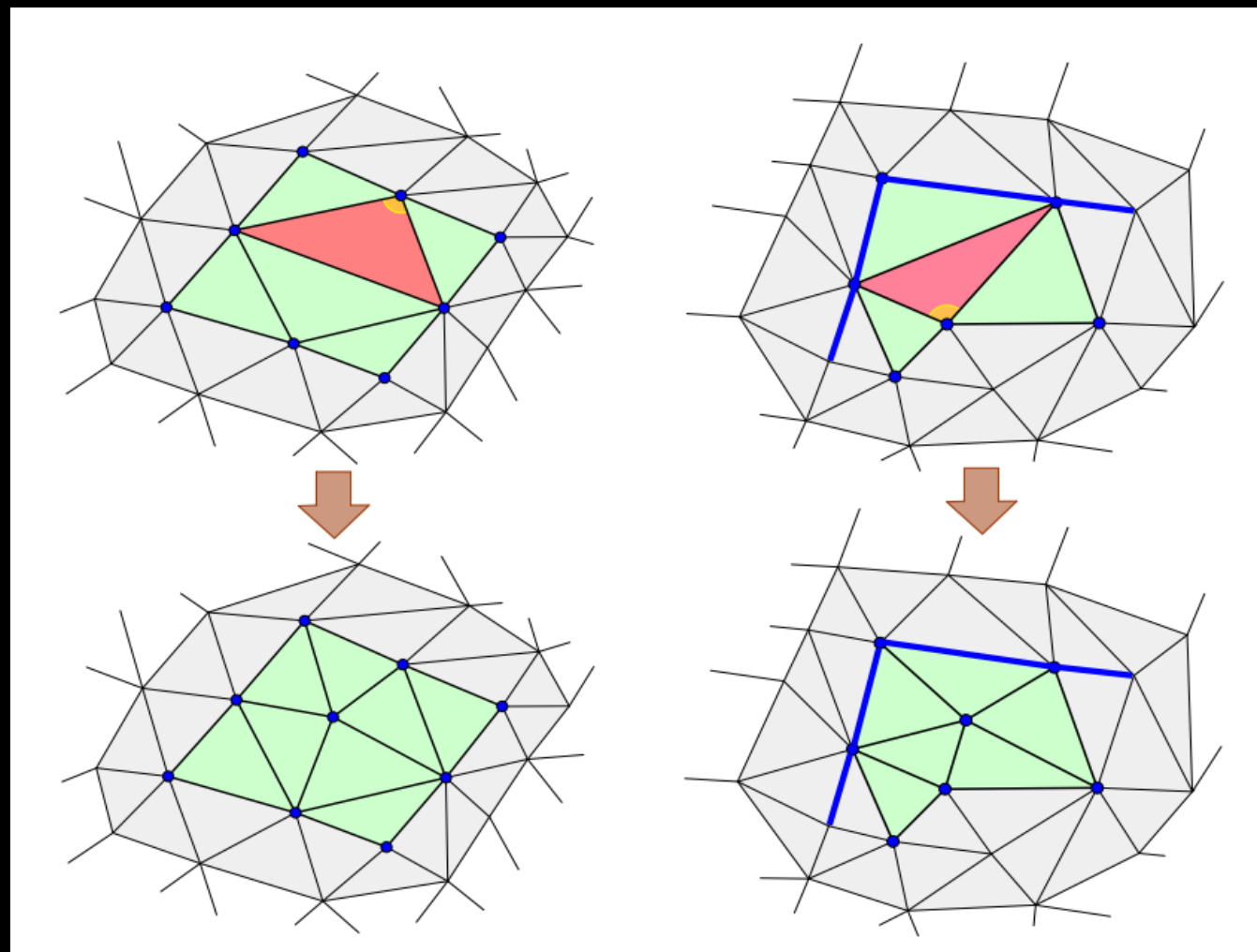
- Isotropic remeshing
 - Error-bounded method
 - Error-Bounded and Feature Preserving Surface Remeshing with Minimal Angle Improvement
 - Improve small and large angles
 - Optimal Delaunay Triangulation (ODT)
 - Centroidal Patch Triangulation (CPT)
- Anisotropic remeshing
 - Introduction
 - ODT with general convex function
 - Local convex triangulation
 - Partial-based method
 - High-dim embedding

Angles

Isotropic Surface Remeshing without Large and Small Angles

- All the triangles with small or large angles outside the desired bounds $[\beta_{\min}, \beta_{\max}]$ are processed.
- Large angle removal: edge splitting
- Small angle improvement: edge collapsing

Large angle removal



1. Split
2. Flip



Outlines

- Isotropic remeshing
 - Error-bounded method
 - Error-Bounded and Feature Preserving Surface Remeshing with Minimal Angle Improvement
 - Improve small and large angles
 - Optimal Delaunay Triangulation (ODT)
 - Centroidal Patch Triangulation (CPT)
- Anisotropic remeshing
 - Introduction
 - ODT with general convex function
 - Local convex triangulation
 - Partial-based method
 - High-dim embedding

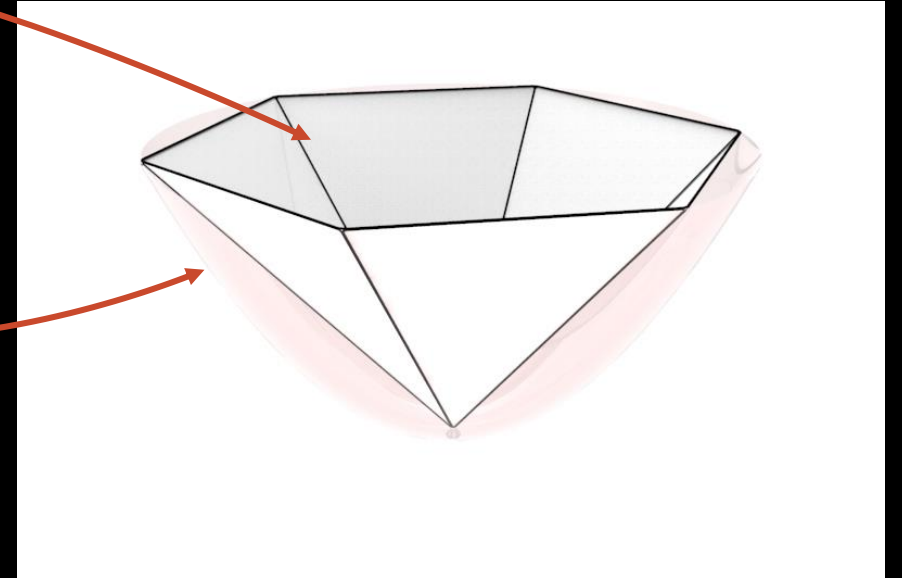
Optimal Delaunay Triangulation (ODT)

$$E = \sum_{T \in \mathcal{T}} \int_T |\hat{u}(x) - u(x)| dx$$

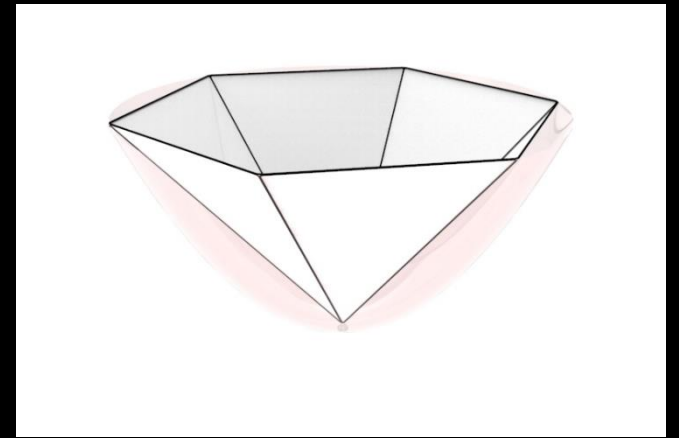
$$u(x): z = x^2 + y^2$$

$\hat{u}(x)$: piecewise linear interpolation of u

\mathcal{T} : a triangulation



Fix positions of vertices, Delaunay triangulation is optimal.



Update of vertices' positions

- Fix the triangulation, update the vertices.

$$E = \sum_{T \in \mathcal{T}} \int_T |\hat{u}(x) - u(x)| dx = \sum_{T \in \mathcal{T}} \int_T \hat{u}(x) dx + C$$

$$= \sum_{T \in \mathcal{T}} \frac{|T|}{3} (u(p_i) + u(p_j) + u(p_k)) + C$$

$$\nabla E_{p_i} = \sum_{T \in \Omega(i)} \frac{\nabla |T|}{3} (u(p_i) + u(p_j) + u(p_k)) + \frac{|\Omega|}{3} \nabla u(p_i) = 0$$

Because $\sum_{T \in \Omega(i)} \frac{\nabla |T|}{3} u(p_i) = 0$

$$\nabla u(p_i) = -\frac{1}{|\Omega|} \sum_{T \in \Omega(i)} \frac{\nabla |T|}{3} (u(p_j) + u(p_k))$$

Optimal position

- Since $u = x^2 + y^2$

$$p_i^* = -\frac{1}{|\Omega|} \sum_{T \in \Omega(i)} \frac{\nabla |T|}{6} (\|p_j\|^2 + \|p_k\|^2)$$

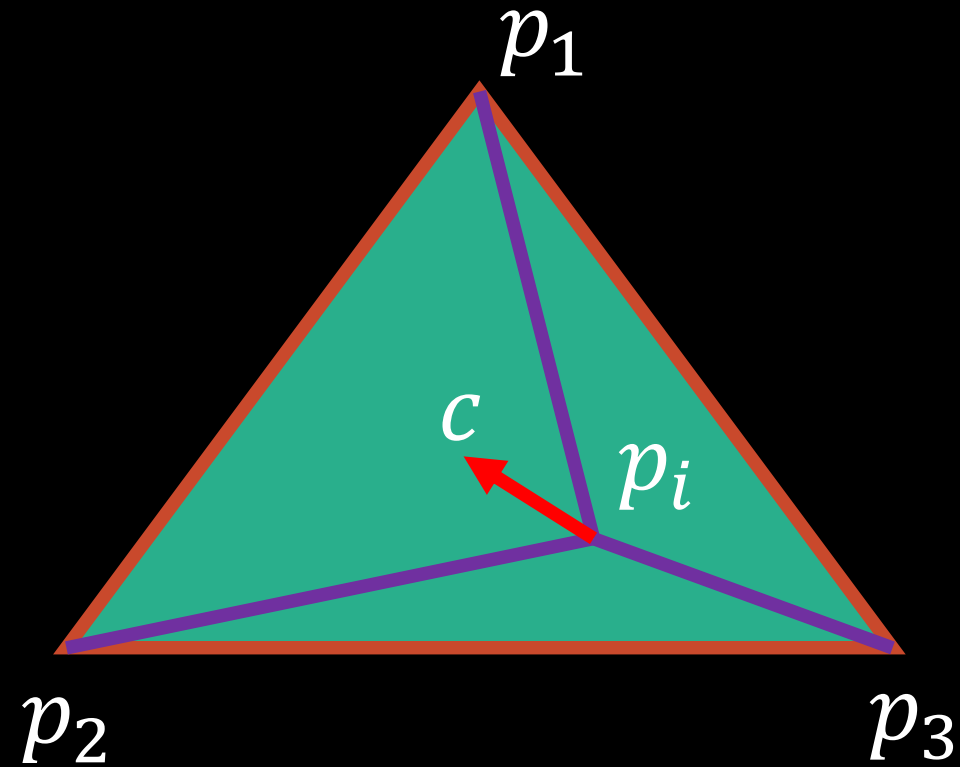
If $u = x^2 + y^2 = \|x\|^2 \rightarrow u = \|x - p_i\|^2$, it does not change the interpolation error, leading to the same optimal position.

$$p_i^* = p_i - \frac{1}{|\Omega|} \sum_{T \in \Omega(i)} \frac{\nabla |T|}{6} (\|p_j - p_i\|^2 + \|p_k - p_i\|^2)$$

Weighted circumcenter

- Since $\sum_{T \in \Omega(i)} \frac{\nabla|T|}{6} = 0$, if $\|p_j - p_i\|^2$ are equal, then $p_i^* = p_i$.
- For the right case, the optimal position of p_i is the circumcenter c of $\Delta p_1 p_2 p_3$.

$$\begin{aligned}
 & c \\
 & = p_i \\
 & - \frac{1}{|\Omega|} \left(\frac{\nabla|\Delta p_1 p_i p_3|}{6} (\|p_1 - p_i\|^2 + \|p_3 - p_i\|^2) \right. \\
 & \left. + \frac{\nabla|\Delta p_1 p_2 p_i|}{6} (\|p_1 - p_i\|^2 + \|p_2 - p_i\|^2) \right)
 \end{aligned}$$



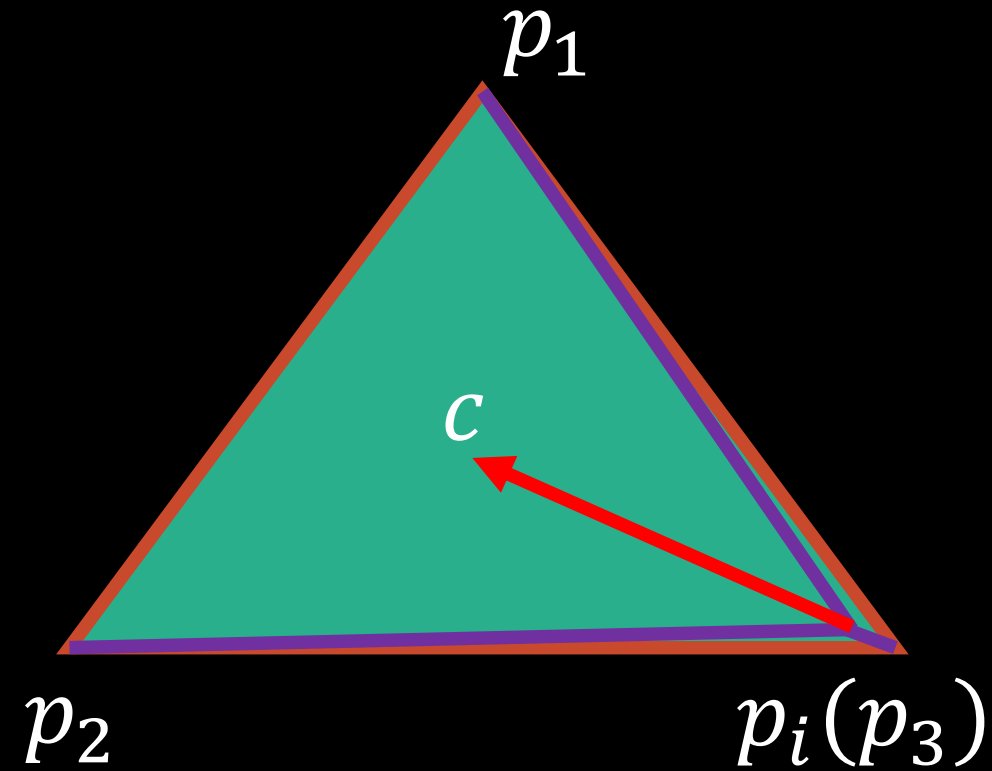
Weighted circumcenter

- A special case: $p_i = p_3$

c

$= p_3$

$$= \frac{1}{|\Delta p_1 p_2 p_3|} \left(\frac{\nabla |\Delta p_1 p_i p_3|}{6} (\|p_1 - p_3\|^2) \right. \\ \left. + \frac{\nabla |\Delta p_1 p_2 p_3|}{6} (\|p_1 - p_3\|^2 + \|p_2 - p_3\|^2) \right)$$



Weighted circumcenter

- Taking the one-ring of p_3

$$\begin{aligned} & \sum_{T_j \in \Omega(p_3)} |T_j| c_j \\ &= \sum_{T_j \in \Omega(p_3)} |T_j| p_3 \\ &- \sum_{T_j \in \Omega(p_3)} \left(\frac{\nabla |\Delta p_1 p_i p_3|}{6} (\|p_1 - p_3\|^2) + \frac{\nabla |\Delta p_1 p_2 p_3|}{6} (\|p_1 - p_3\|^2 + \|p_2 - p_3\|^2) \right) \end{aligned}$$

Centroidal Voronoi tessellations (CVT)

$$E = \sum_{i=1}^n \int_{V_i} \|x - p_i\|^2 dx \quad \text{CVT}$$

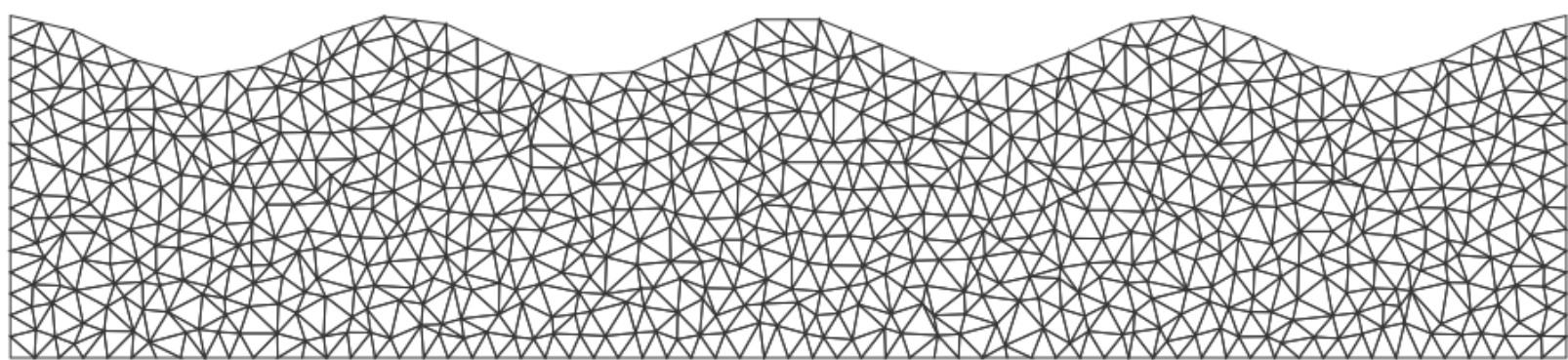


$$E = \sum_{i=1}^n \int_{\Omega(p_i)} \|x - p_i\|^2 dx \quad \text{CPT}$$

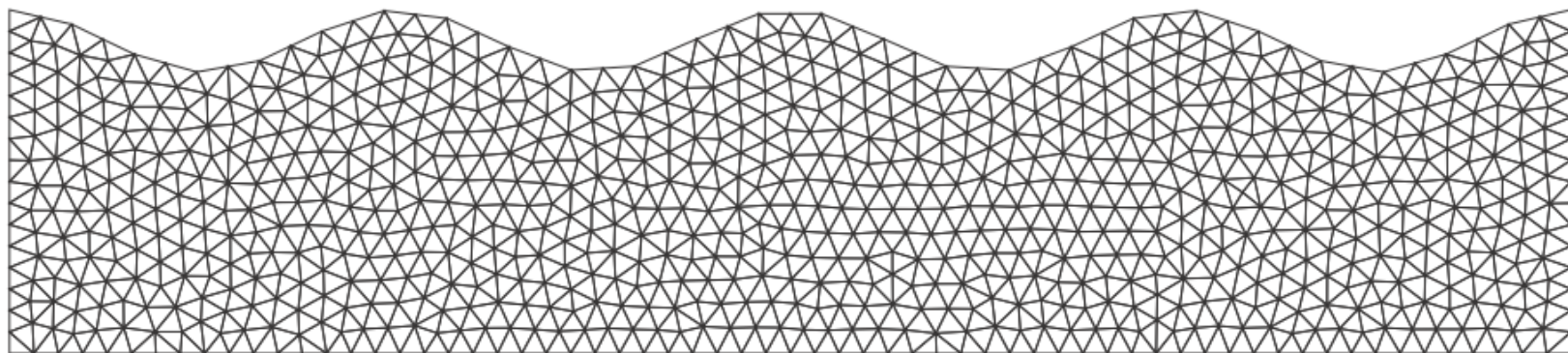
Centroidal Patch Triangulation (CPT)

- Delaunay triangulation
- Moving p_i to the centroid c_i of the corresponding patch.

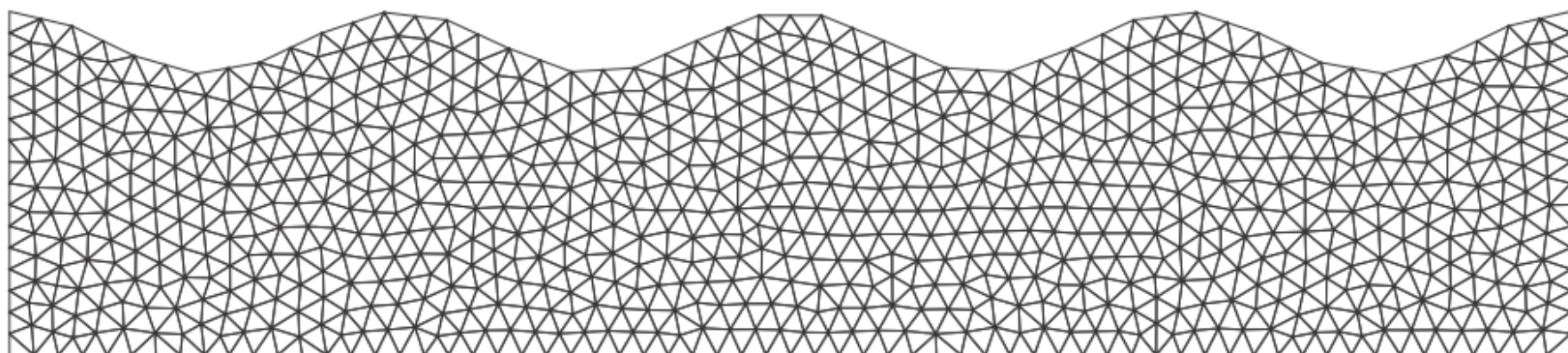
$$E = \sum_{i=1}^n \int_{\Omega(p_i)} \|x - p_i\|^2 dx$$



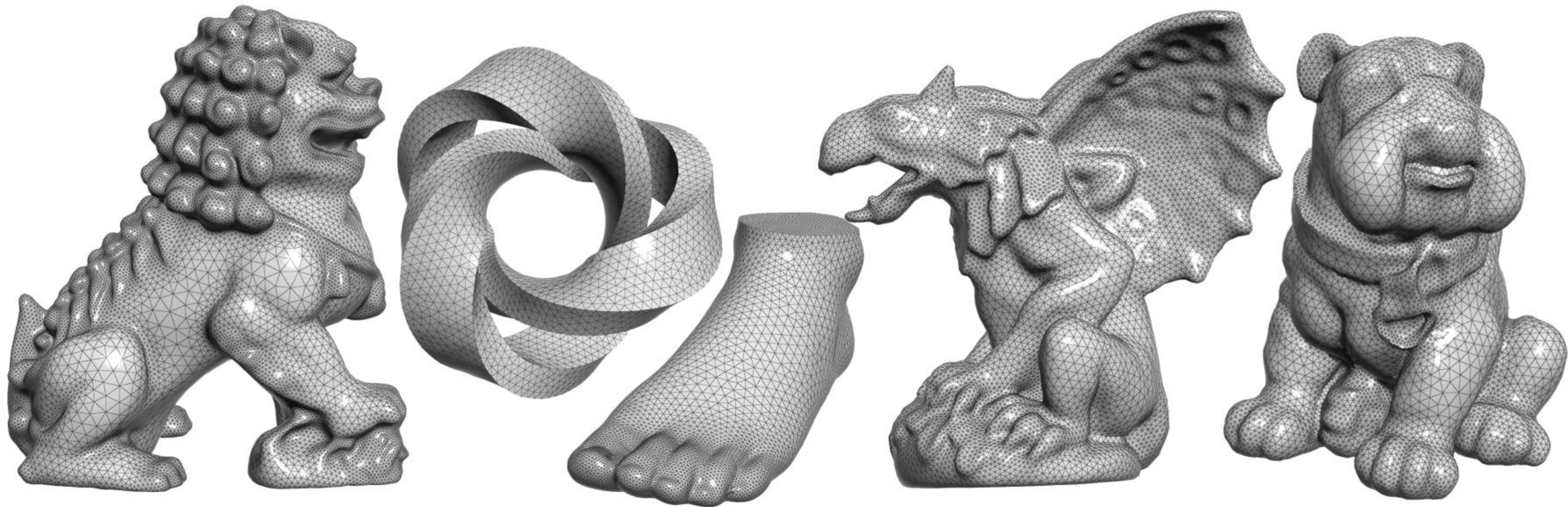
(a) Original mesh: $\min(q) = 0.141$, $\text{mean}(q) = 0.875$, $\text{uniformity} = 15.62\%$



(b) ODT smoothing: $\min(q) = 0.709$, $\text{mean}(q) = 0.964$, $\text{uniformity} = 5.16\%$



(c) CPT smoothing: $\min(q) = 0.708$, $\text{mean}(q) = 0.967$, $\text{uniformity} = 6.15\%$



Simulated annealing method with the operation “perturb-optimize”

Outlines

- Isotropic remeshing
 - Error-bounded method
 - Error-Bounded and Feature Preserving Surface Remeshing with Minimal Angle Improvement
 - Improve small and large angles
 - Optimal Delaunay Triangulation (ODT)
 - Centroidal Patch Triangulation (CPT)
- Anisotropic remeshing
 - Introduction
 - ODT with general convex function
 - Local convex triangulation
 - Partial-based method
 - High-dim embedding

Metric

- A metric on a set X is a function (called the distance function or simply distance)

$$d: X \times X \rightarrow [0, \infty)$$

where $[0, \infty)$ is the set of non-negative real numbers.

- For all $x, y, z \in X$, the following conditions are satisfied:
 - ✓ Non-negativity or separation axiom
 - ✓ $d(x, y) \geq 0$
 - ✓ Identity of indiscernibles
 - ✓ $d(x, y) = 0 \Leftrightarrow x = y$
 - ✓ Symmetry
 - ✓ $d(x, y) = d(y, x)$
 - ✓ Subadditivity or triangle inequality
 - ✓ $d(x, z) \leq d(x, y) + d(y, z)$

Metric

- Conditions 1 and 2 together define a **positive-definite** function.
- The first condition is implied by the others.
- In practice, the metric can be represented by a **positive-definite** symmetric $m \times m$ matrix $M(x)$.
 - $M(x) = Q(x)^T Q(x)$.
- Given a $M(x)$, its decomposition to $Q(x)$ is non-unique.

Length

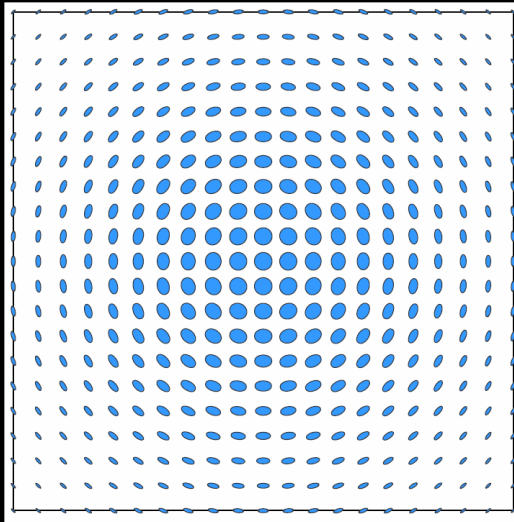
- Given the metric field $M(x)$ and an open curve $C \subset \Omega$, the length of C is defined as the integration of the length of tangent vector along the curve C with metric $M(x)$
- The anisotropic distance $d_M(x, y)$ between two points x and y can be defined as the length of the (possibly non-unique) shortest curve (**assuming line segment**) that connects x and y .

$$\int_0^1 \sqrt{(x - y)^T M(tx + (1 - t)y)(x - y)} dt$$

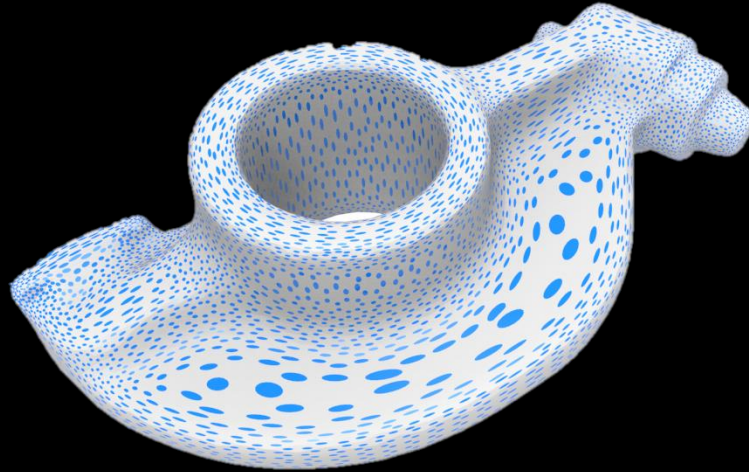
Anisotropic remeshing

- Input:
 - Domain: $\Omega \in \mathbb{R}^d$
 - Metric field: $\mathbf{M}(\mathbf{x})$, $\mathbf{x} \in \Omega$
 - $d \times d$ positive-definite matrix
- Isotropic remeshing
 - All edge lengths are as equal as possible.
- Anisotropic remeshing
 - All edge lengths **with metric** are as equal as possible.

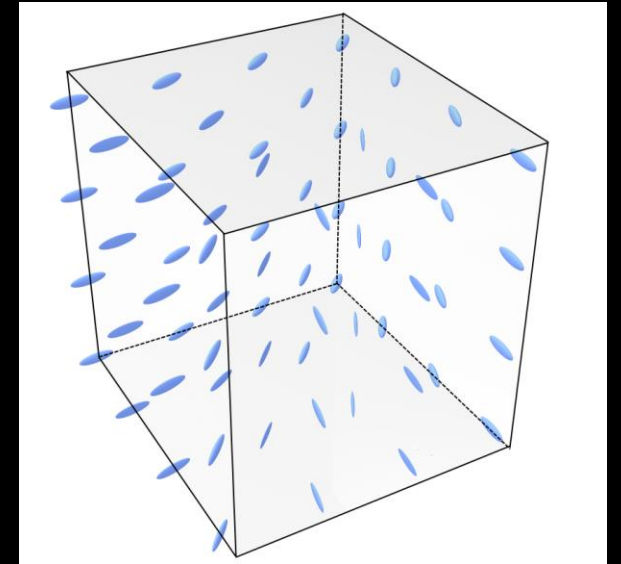
Input examples



$\Omega = 2\text{D square,}$
 $M(p) = \text{Hessian of given } u$



$\Omega = 3\text{D surface,}$
 $M(p) = \text{mesh curvature}$



$\Omega = 3\text{D cube,}$
 $M(p) = \text{given tensor field}$

Anisotropic remeshing

- Eigen-decomposition: $M(x) = U(x)\Lambda U(x)^T$
- Transformation $\Phi = \Lambda^{1/2}U(x)^T$
- The quality metrics are measured in the transformed space.



transforms simplex to isotropic space

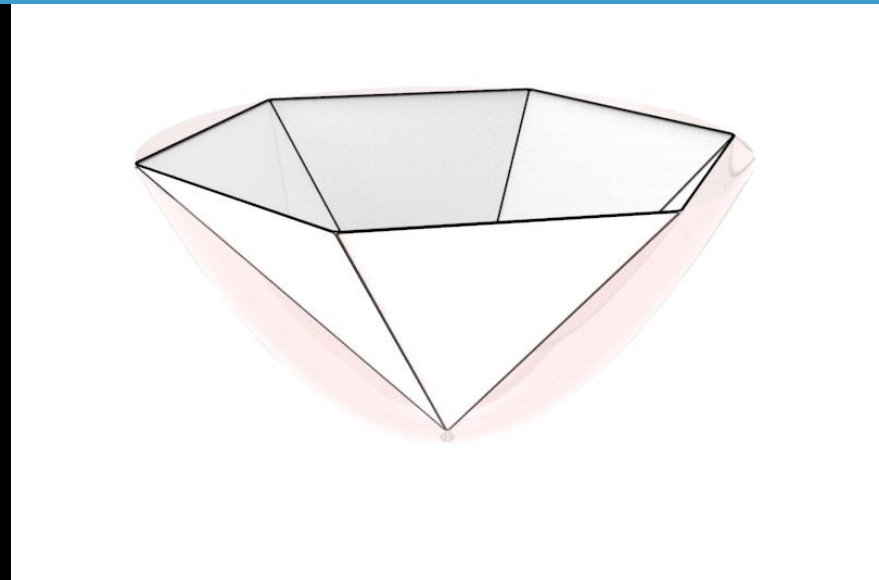
ODT properties [Chen 2004, Liu et al. 2013, Desbrun et al. 2013]

$$\min_T E_{ODT} \triangleq \sum_{\tau \in T} \int_{\tau} |\hat{u}(\mathbf{x}) - u(\mathbf{x})| d\mathbf{x}$$

- $u(\mathbf{x}) = \frac{1}{2} \mathbf{x}^2 \Rightarrow$
 $T_{ODT} =$ Delaunay triangulation

- $u(\mathbf{x})$ convex \Rightarrow
 $T_{ODT} =$ regular triangulation

power weight: $w(\mathbf{x}) = \mathbf{x}^2 - 2 u(\mathbf{x})$



ODT method

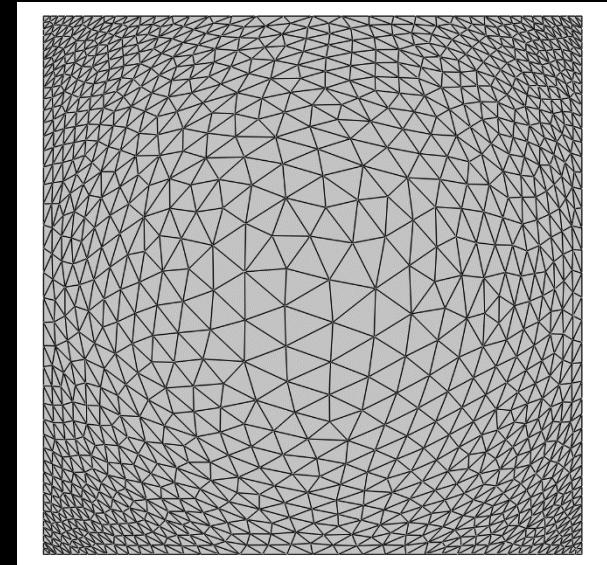
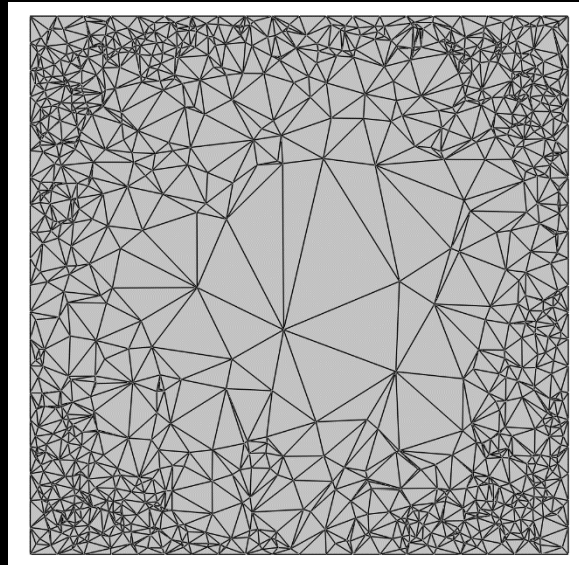
power weight:

$$w(\mathbf{x}) = \mathbf{x}^2 - 2u(\mathbf{x})$$

Iterative mesh optimization:

- update vertices: *move each vertex to its power-weighted circumcenter*
- update connectivity: *compute constrained regular triangulation*

$$u(x, y) = e^{\frac{(x^2+y^2)}{10}}$$
$$\Omega = [-5, 5]^2$$



Thinking from optimization

$$\min_T E_{ODT} \triangleq \sum_{\tau \in T} \int_{\tau} |\hat{u}(\mathbf{x}) - u(\mathbf{x})| d\mathbf{x}$$

1. Update connectivity
2. Update vertex positions

ODT limitations

- M must be Hessian of a global u
- u must be convex

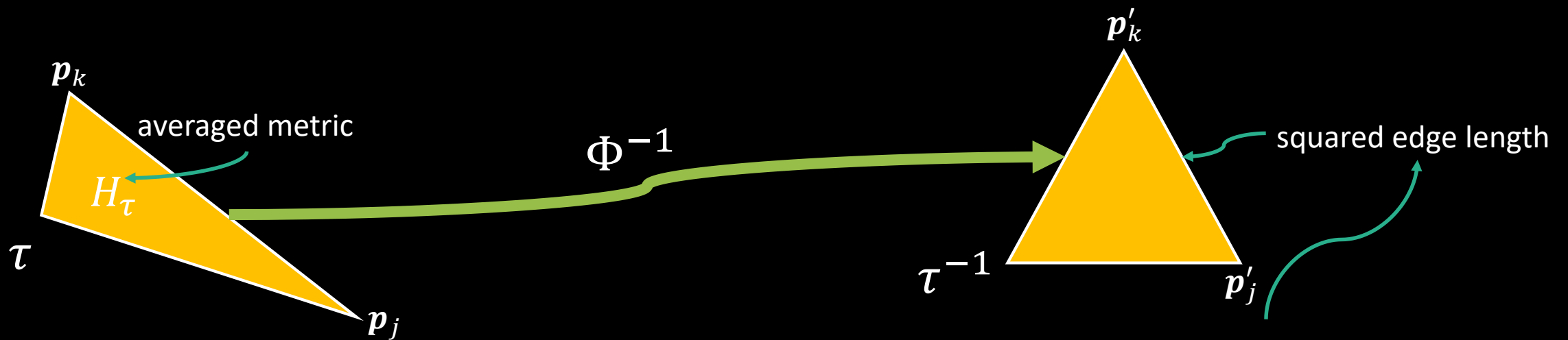
⇒ neither true for general M !

Locally Convex Triangulation (LCT)

Anisotropic Simplicial Meshing Using Local Convex Functions

Anisotropy approximated **locally** by per-simplex convex function:

$$u(\mathbf{x}) \rightarrow u_\tau(\mathbf{x}).$$



$$u_\tau(\mathbf{x}) \triangleq \frac{1}{2} \mathbf{x}^T H_\tau \mathbf{x} \longrightarrow E_\tau \triangleq \int_\tau |\hat{u}_\tau - u_\tau| d\mathbf{x} \longrightarrow E_\tau = \frac{|\tau| \sum_{j < k} (\mathbf{p}_j - \mathbf{p}_k)^T H_\tau (\mathbf{p}_j - \mathbf{p}_k)}{2(d+1)(d+2)}$$

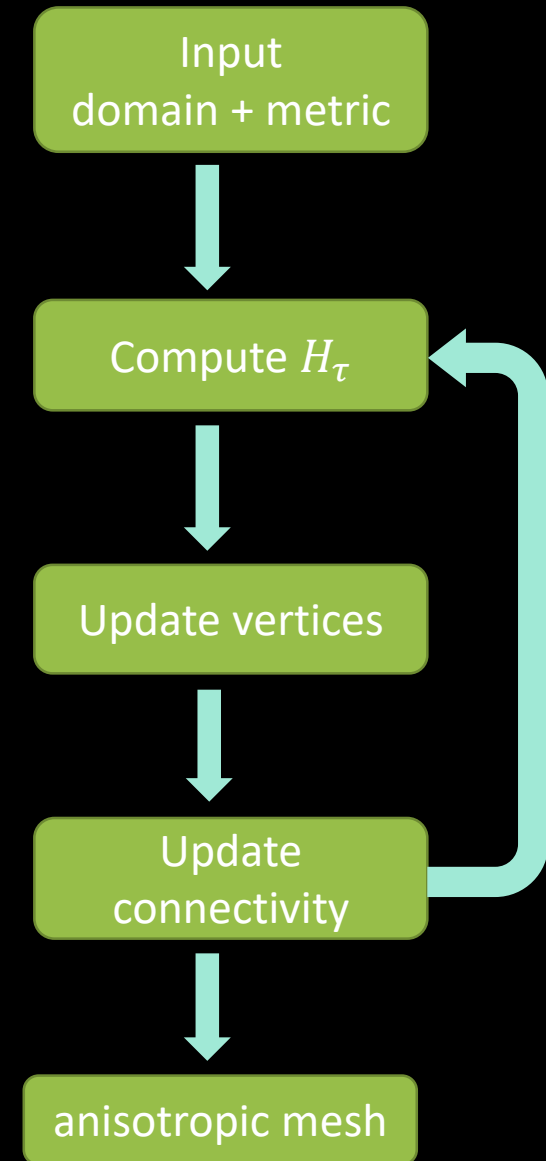
LCT Optimization

$$\min_T E_{LCT} \triangleq \sum_{\tau \in T} E_\tau = \sum_{\tau \in T} \int_{\tau} |\hat{u}_\tau(\mathbf{x}) - u_\tau(\mathbf{x})| d\mathbf{x}$$

Step 1: compute H_τ on each simplex

Step 2: fix connectivity, update vertices

Step 3: fix vertices, update connectivity

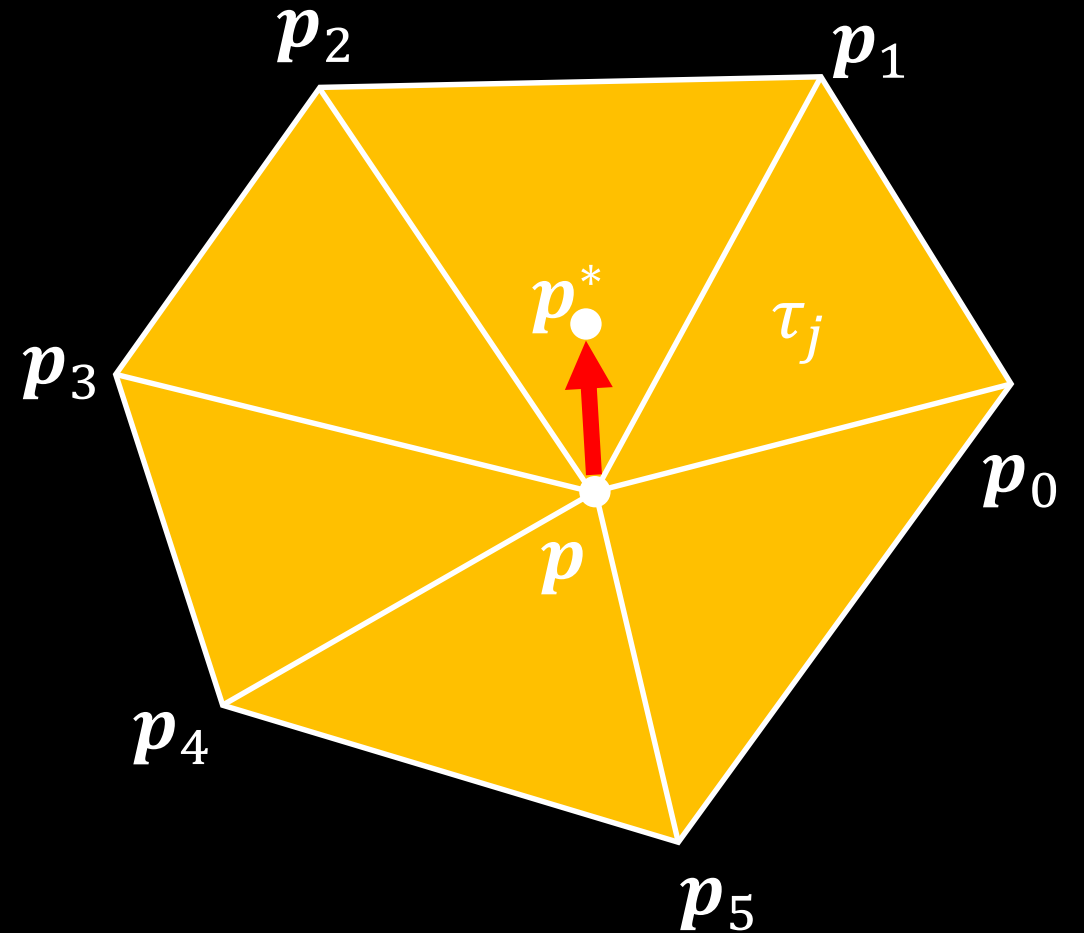


Vertex update

Update each vertex \mathbf{p} in sequence:

- one-ring $\tau_j \in \Omega(\mathbf{p})$
- energy sum $E_{\mathbf{p}} = \sum_j E_{\tau_j}$
- gradient $\mathbf{g} = \partial E_{\mathbf{p}} / \partial \mathbf{p}$
- Hessian $\mathbf{h} = \text{PD}(\partial^2 E_{\mathbf{p}} / \partial \mathbf{p}^2)$
- $\mathbf{p}^* := \mathbf{p} - \alpha \mathbf{h}^{-1} \mathbf{g}$

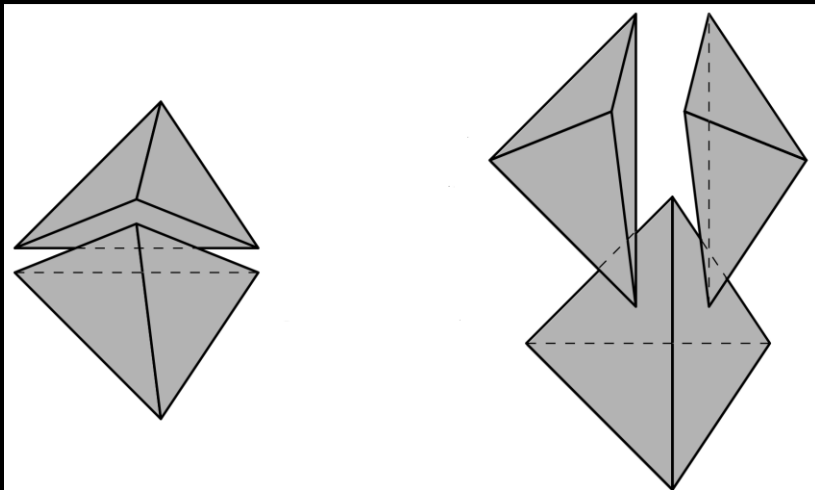
restrict boundary/feature vertices



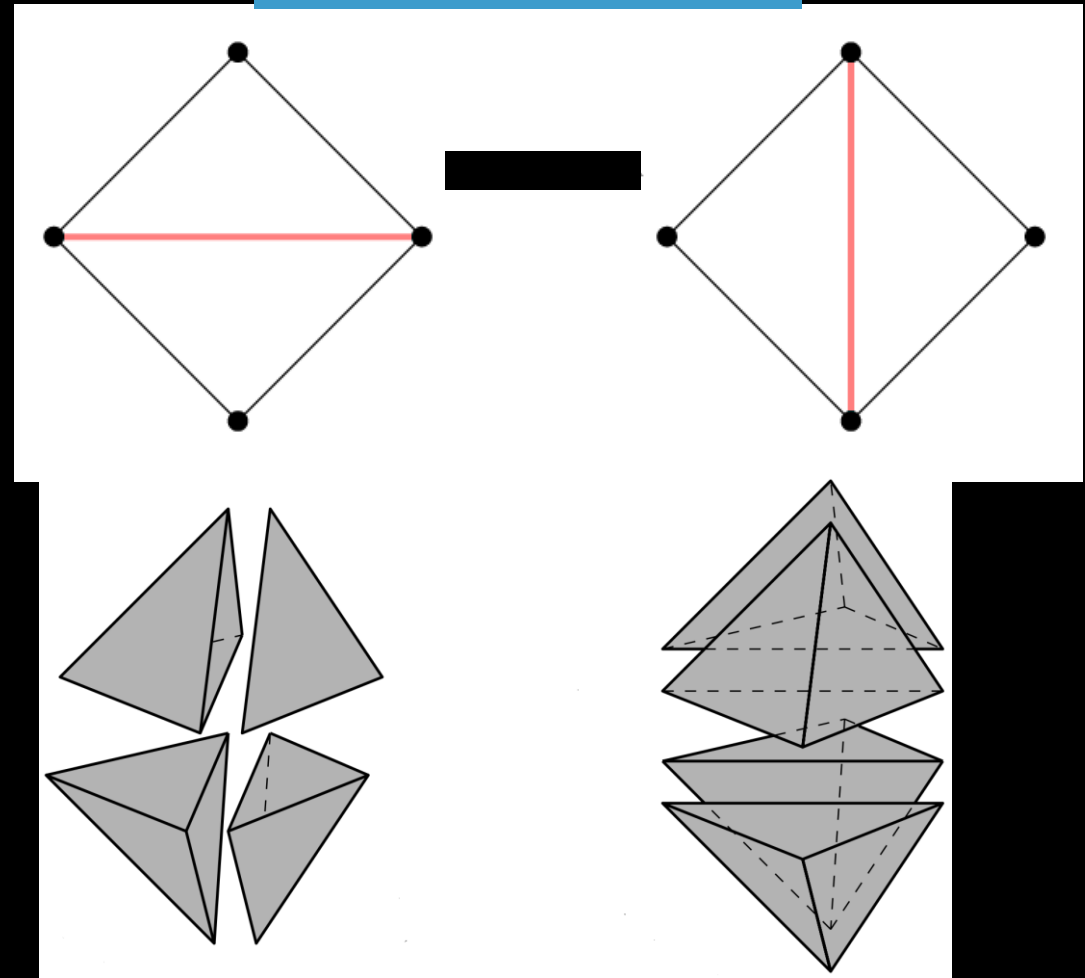
Connectivity update

triangle mesh \Rightarrow edge flip

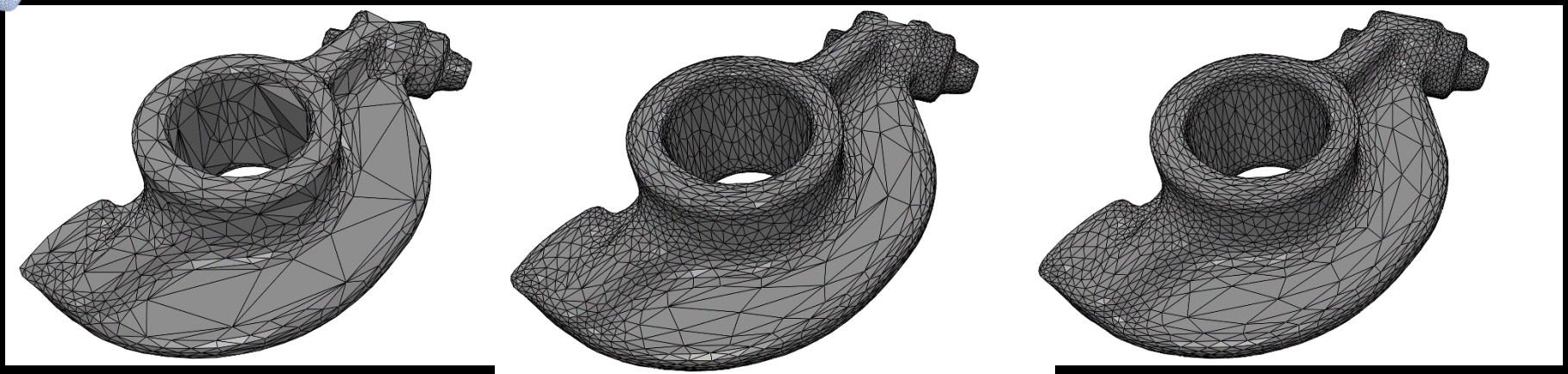
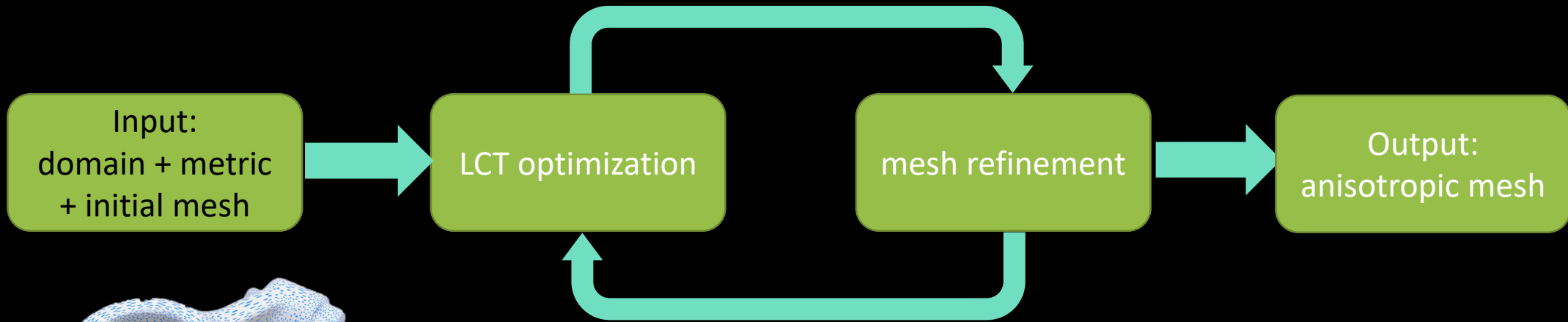
tetrahedral mesh \Rightarrow 2-3,
3-2, 4-4, 2-2 flips



$$E_{\text{before}} > E_{\text{after}}$$



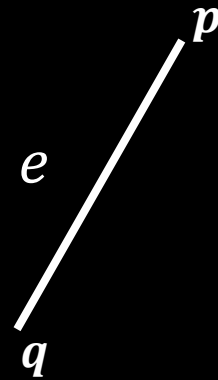
Anisotropic meshing pipeline



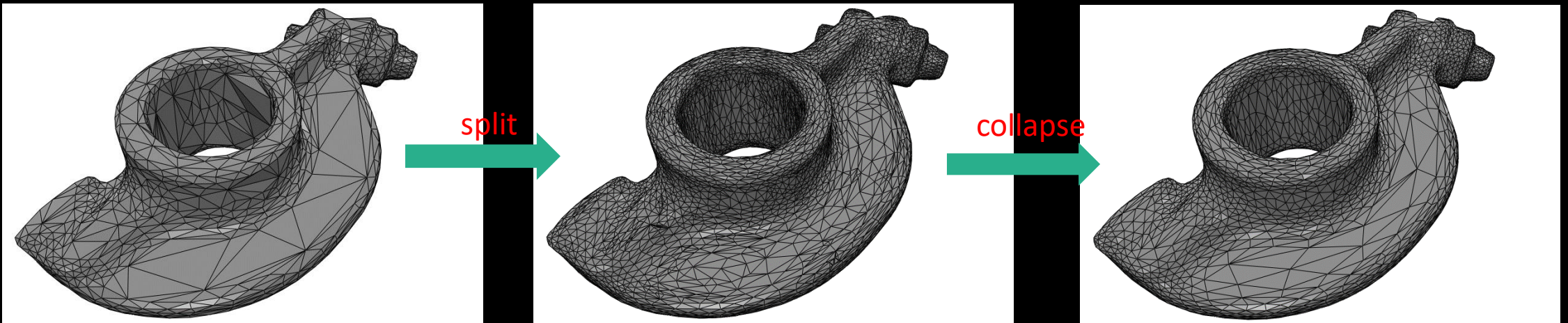
Edge refinement

$$L/\beta \leq |e^{-1}| \leq \beta L$$

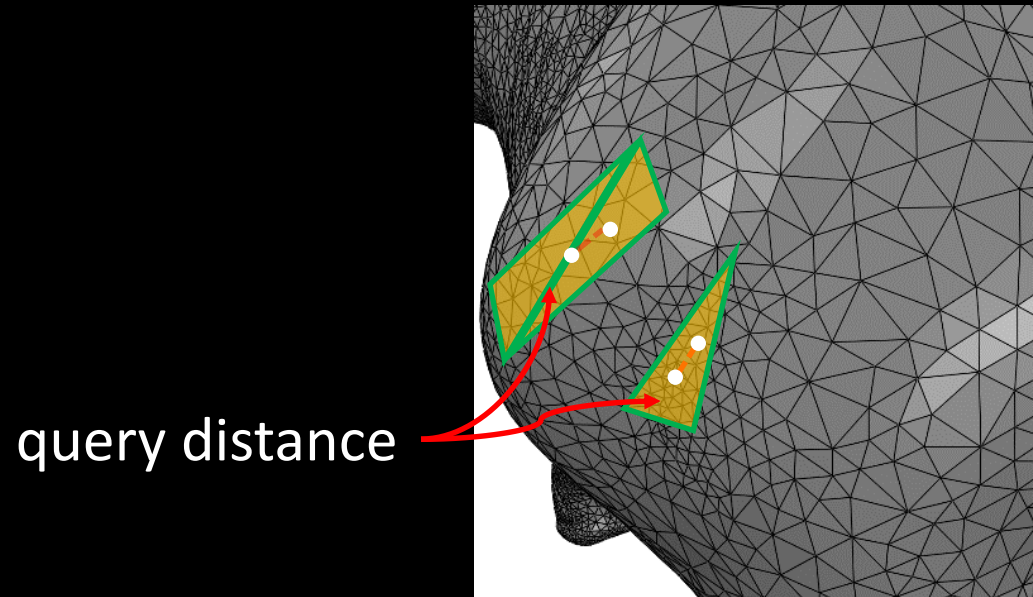
- split if $|e^{-1}| > \beta L$
- collapse if $|e^{-1}| < L/\beta$



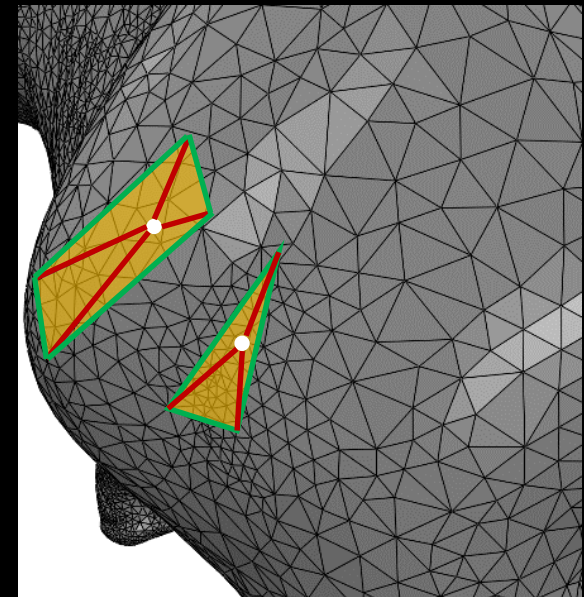
$$|e^{-1}| \approx \sqrt{(\mathbf{p} - \mathbf{q})^T \frac{\mathbf{M}(\mathbf{p}) + \mathbf{M}(\mathbf{q})}{2} (\mathbf{p} - \mathbf{q})}$$



Geometric refinement

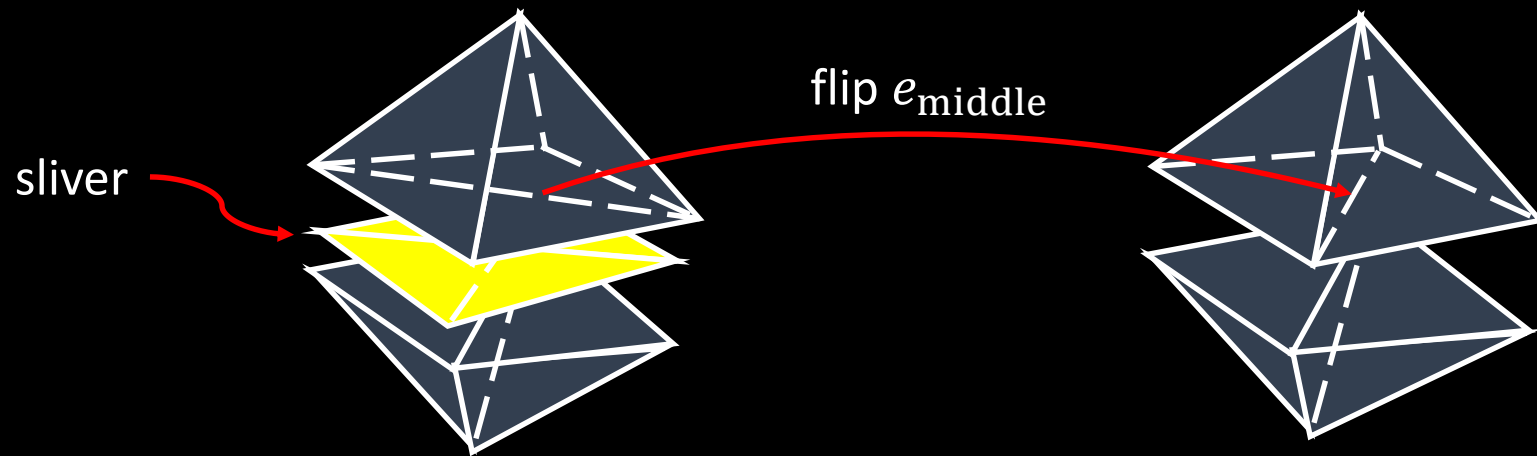


project & split

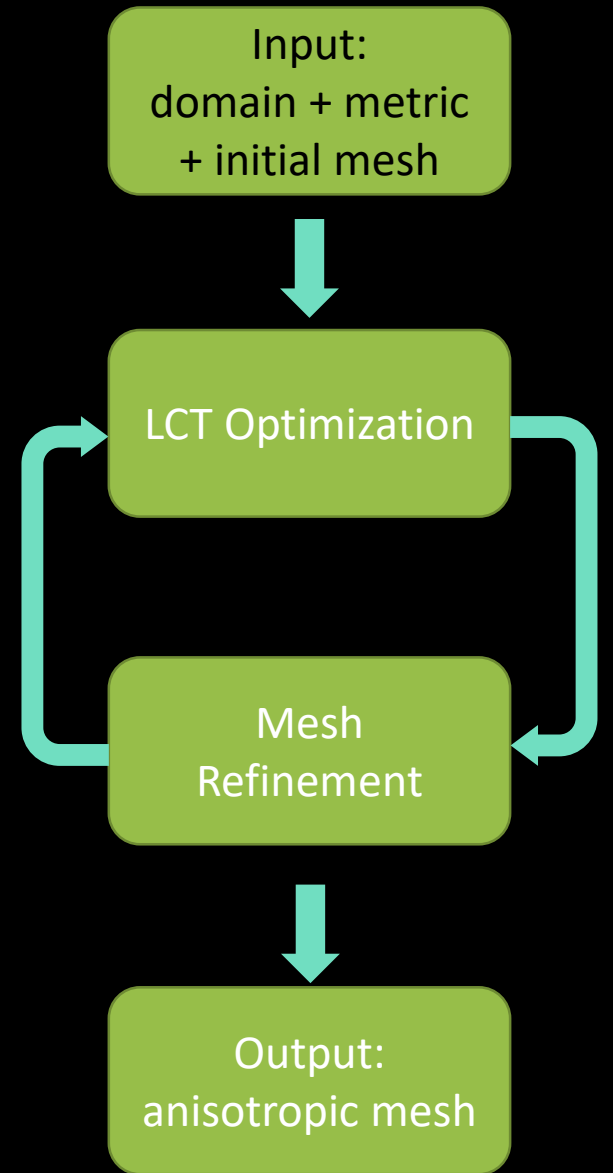
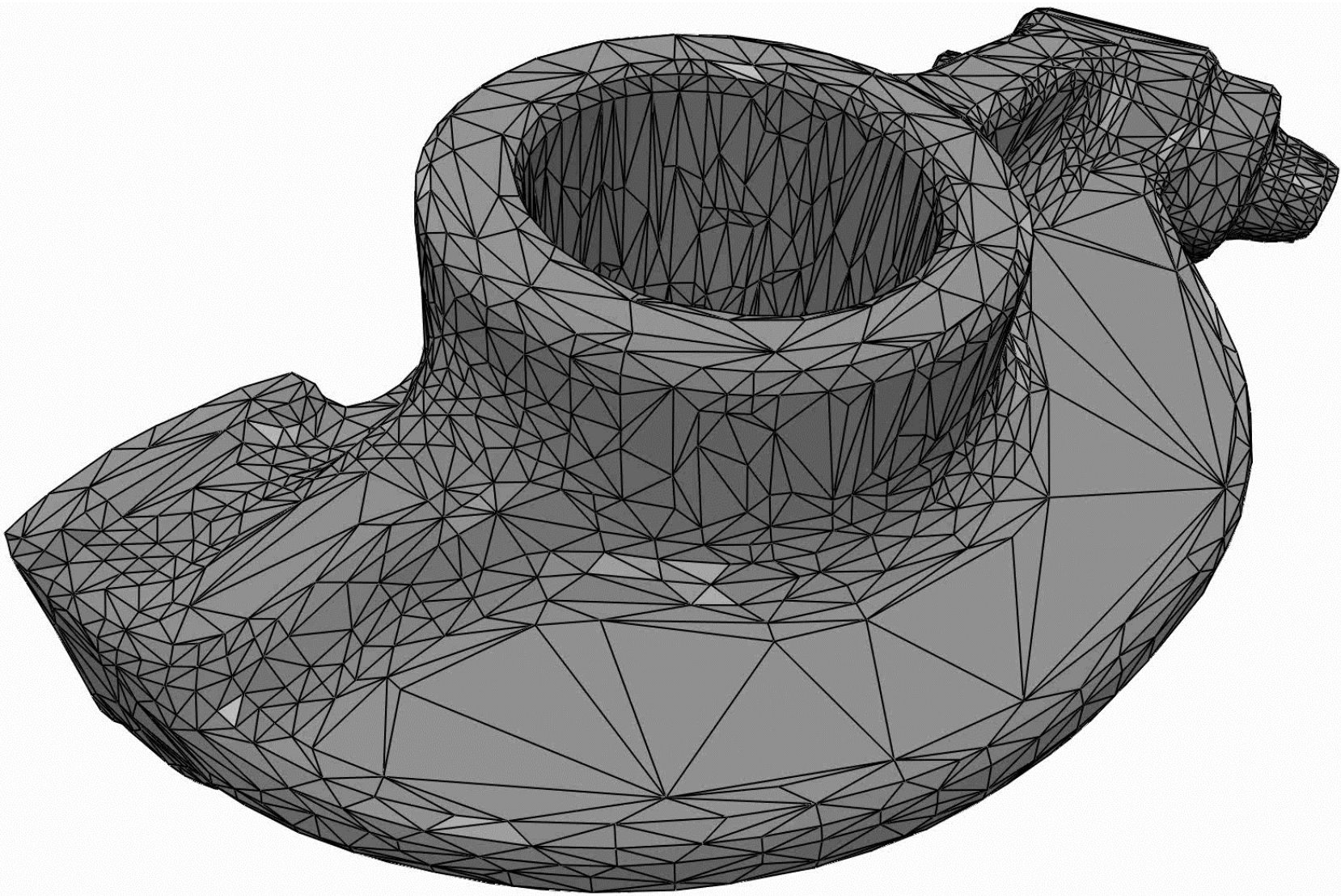


Sliver elimination (tetrahedral mesh)

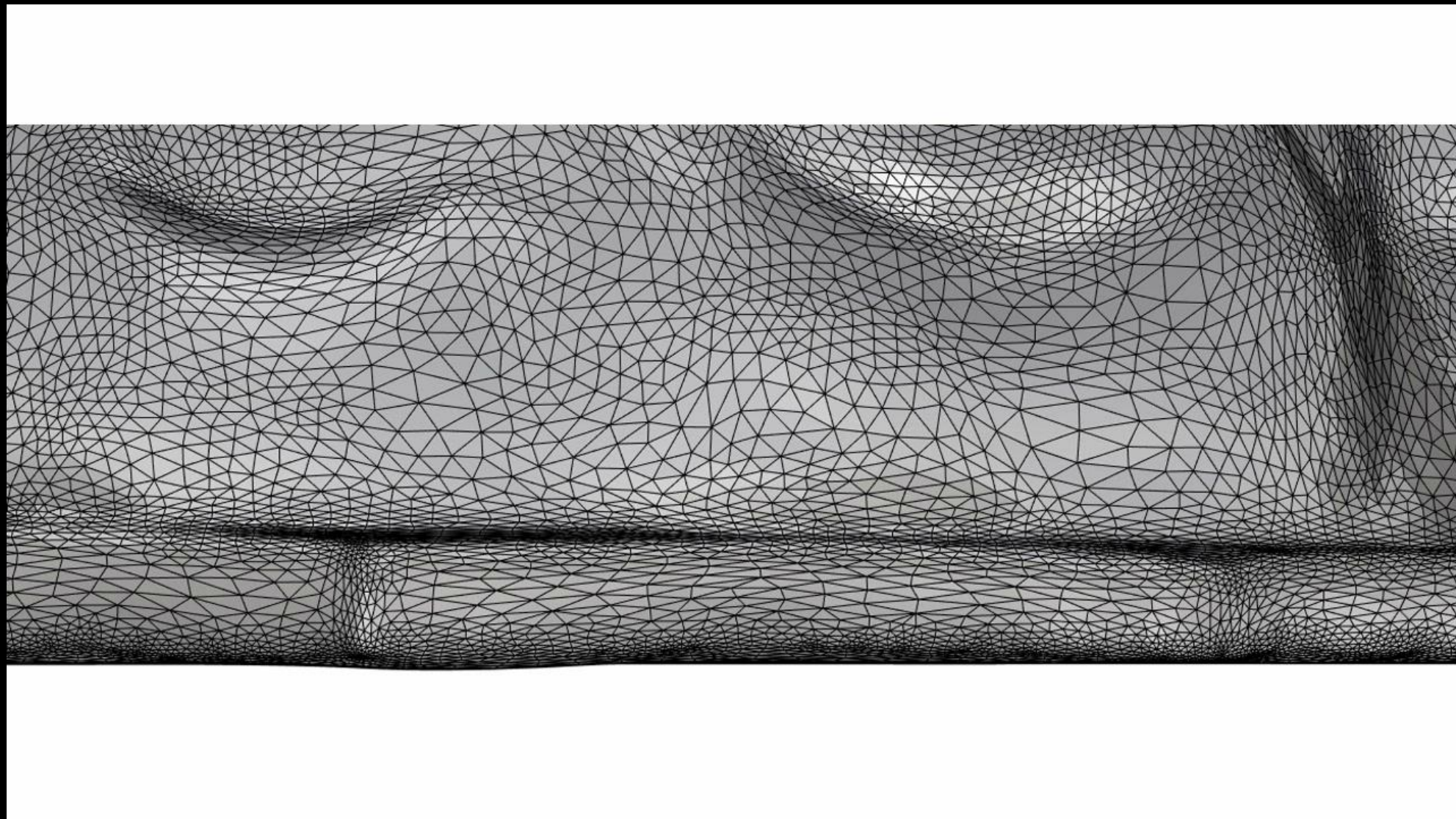
- perform 5-4 flips
- perturb vertices and perform flips to eliminate small dihedral angles
[Tournois et al. 2009]



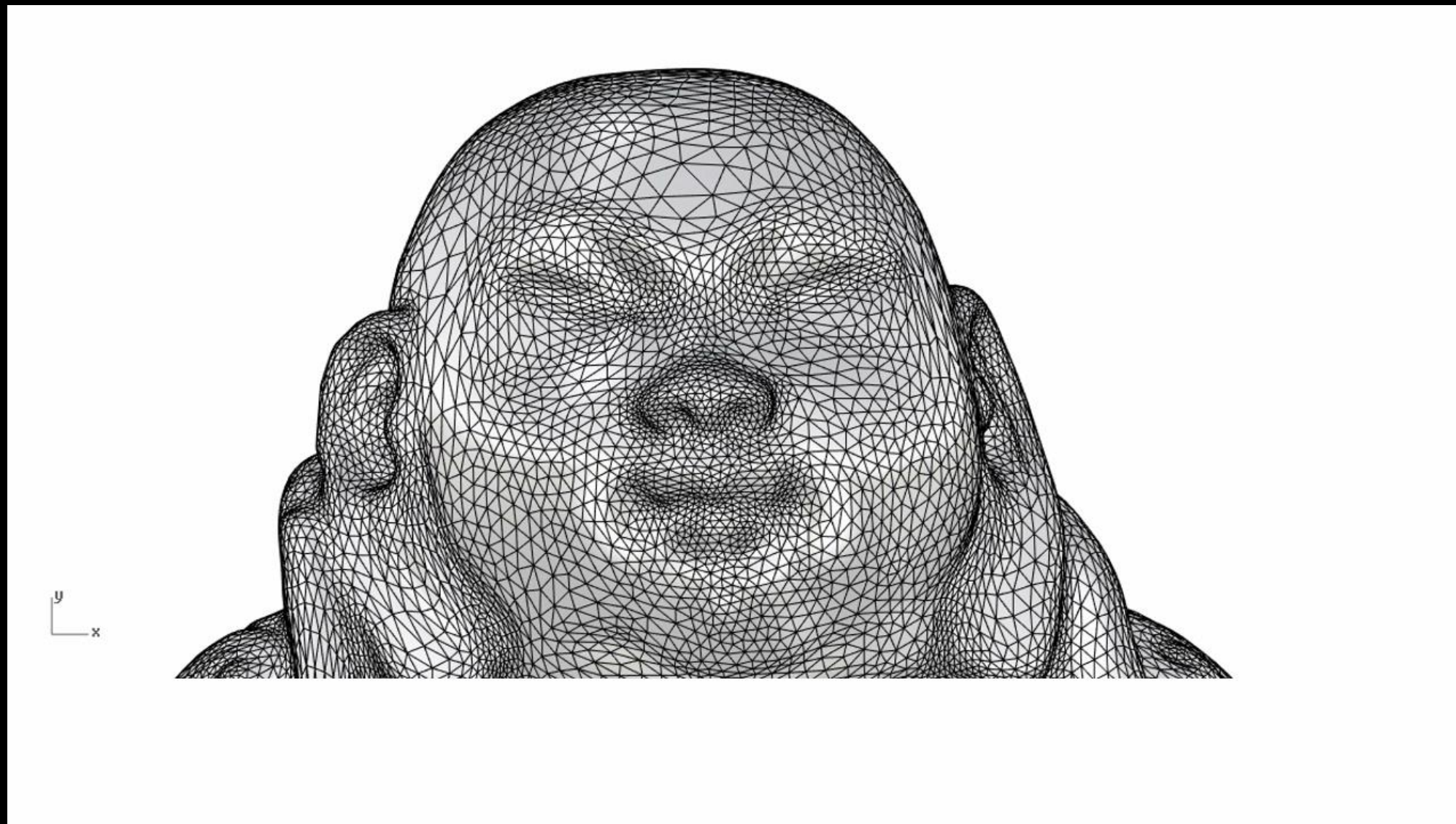
Anisotropic meshing pipeline



Lucy



Buddha



Partial-based method

Particle-Based Anisotropic Surface Meshing

- Considering each vertex as a particle, the potential energy between the particles determines the inter-particle forces. When the forces applied on each particle become **equilibrium**, the particles reach the optimal balanced state with **uniform distribution**.

- Energy:

$$E = \sum_{i=1}^n \sum_{j \neq i}^n E_{ij}$$
$$E_{ij} = \exp \left(- \left(x_i - x_j \right)^T M_{ij} \left(x_i - x_j \right) / 4\sigma^2 \right)$$

Data: a surface Ω with metric \mathbf{M} , and the desired number of vertices n

Result: an anisotropic sampling \mathbf{X} of Ω

Initialize particle locations \mathbf{X} ;

while *stopping condition not satisfied* **do**

Update the ANN data structure for the current sampling \mathbf{X} ;

for *each particle* i **do**

Get particle i 's neighbor $N(i)$ from ANN;

for *each particle* $j \in N(i)$ **do**

Compute \bar{E}^{ij} using Eq. (12);

Compute $\tilde{\mathbf{F}}^{ij}$ using Eq. (21);

end

Sum the total force $\tilde{\mathbf{F}}^i$ using Eq. (22);

Project $\tilde{\mathbf{F}}^i$ to the surface tangent using Eq. (27);

end

Sum the total energy \bar{E} in Eq. (13);

Run L-BFGS with \bar{E} and $\{\tilde{\mathbf{F}}^i\}$, to get updated locations \mathbf{X} ;

Project \mathbf{X} onto the surface;

end

Algorithm 1: Anisotropic Particle Optimization with Metric \mathbf{M} .

$$\frac{Q_{ij}(\mathbf{x}_i - \mathbf{x}_j)}{2\sigma^2} \exp\left(-\frac{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M}_{ij} (\mathbf{x}_i - \mathbf{x}_j)}{4\sigma^2}\right)$$

Particle-Based Anisotropic Surface Meshing

Zichun Zhong¹ Xiaohu Guo¹ Wenping Wang² Bruno Lévy³
Feng Sun² Yang Liu⁴ Weihua Mao⁵

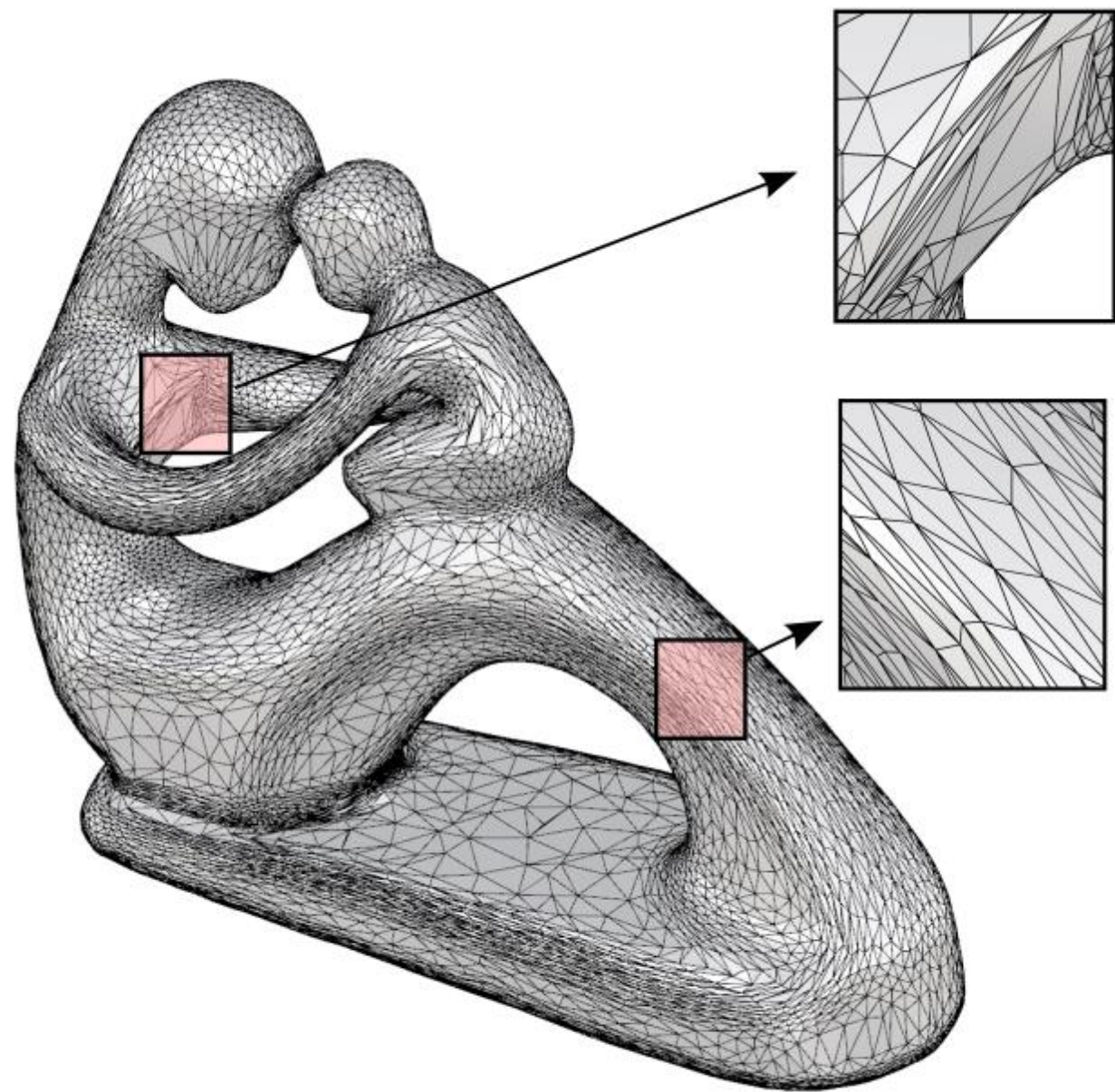
1 The University of Texas at Dallas

2 The University of Hong Kong

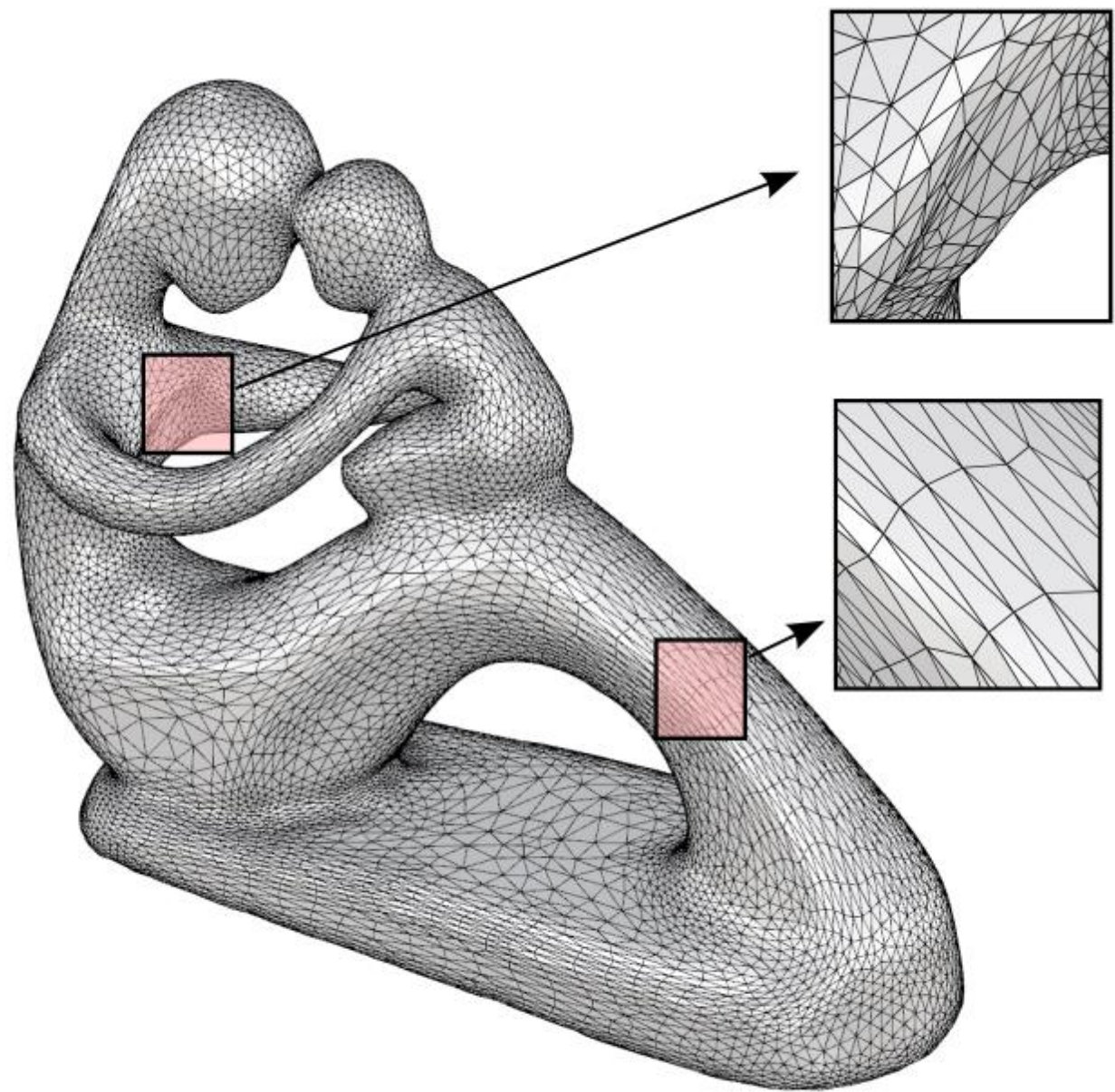
3 INRIA Nancy - Grand Est

4 NVIDIA Corporation

5 UT Southwestern Medical Center at Dallas



particle



LCT

Repairing

Xiao-Ming Fu

Outlines

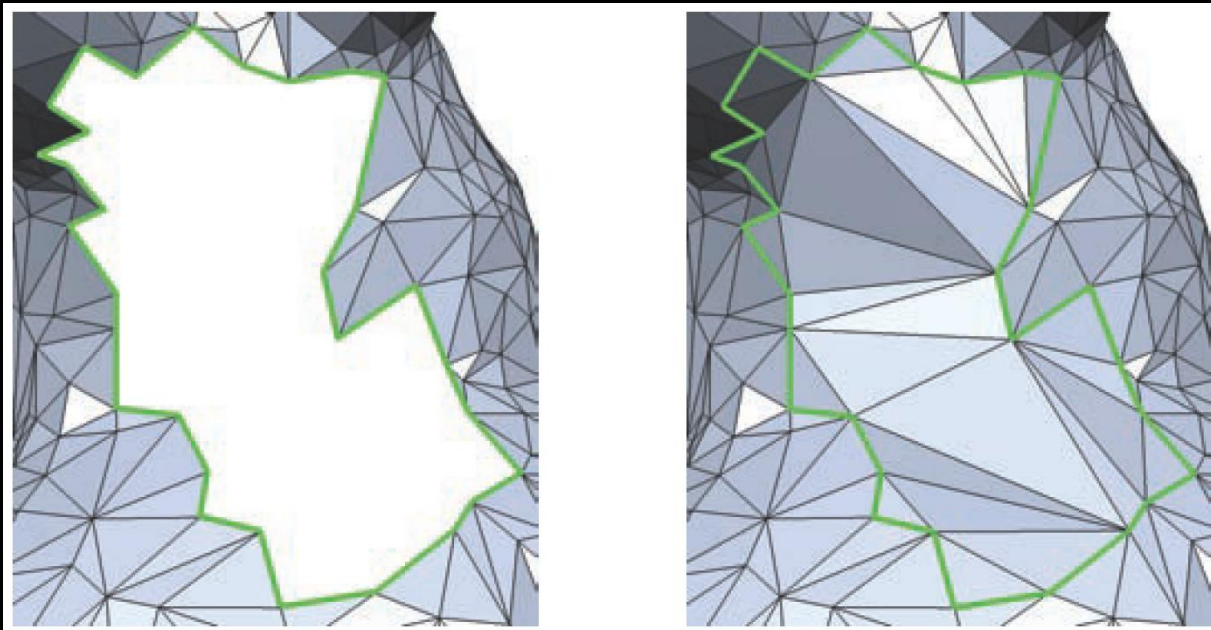
- Definitions
- Defects and flaws
- Upstream and Downstream applications
- Types of input
- Approaches

Outlines

- **Definitions**
- Defects and flaws
- Upstream and Downstream applications
- Types of input
- Approaches

Problem Statement

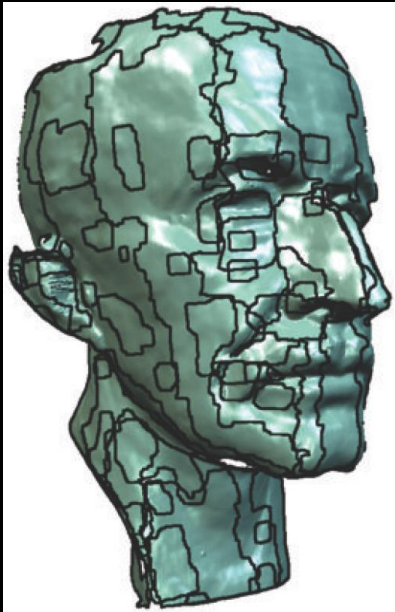
- Model repair is the process of removing **artifacts** from a **geometric model** in order to generate an output model **suitable for further processing** by downstream applications that require certain quality guarantees for their input.



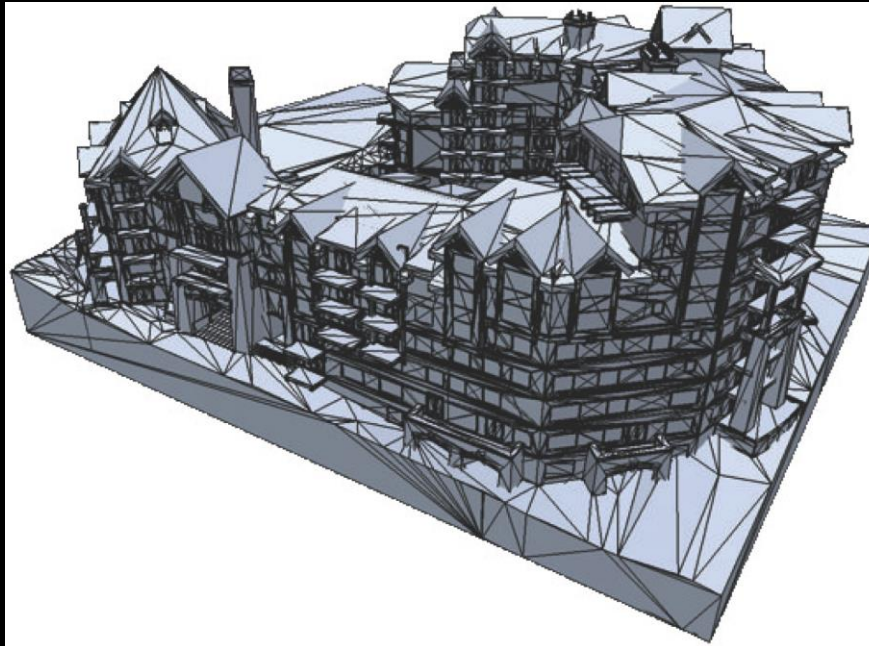
Hole filling

Application dependent

- Depends on the particular application scenario:
 - what kind of “models” are considered,
 - what exactly constitutes an “artifact,”
 - what is meant by “suitable for further processing”



Registered range scans from scanners



Triangle soups
from CAD models

One application

- The design cycle encountered in automotive CAD/CAM.
- Models are typically manually designed in CAD systems that use **trimmed NURBS surfaces** as the underlying data structure for representing freeform surface geometry.
- However, numerical fluid simulations for shape analysis and optimization cannot handle such NURBS patches directly but rather **need a watertight, manifold triangle mesh** as input.
- Thus, there is a need for an intermediate stage that **converts the NURBS model into a triangle mesh**.
- Unfortunately, this conversion process is prone to producing **meshing artifacts** that cannot be handled by simulation packages.
- Thus, the converted model **has to be repaired**—usually in **a tedious manual post-process**, which often takes longer than the simulation itself.

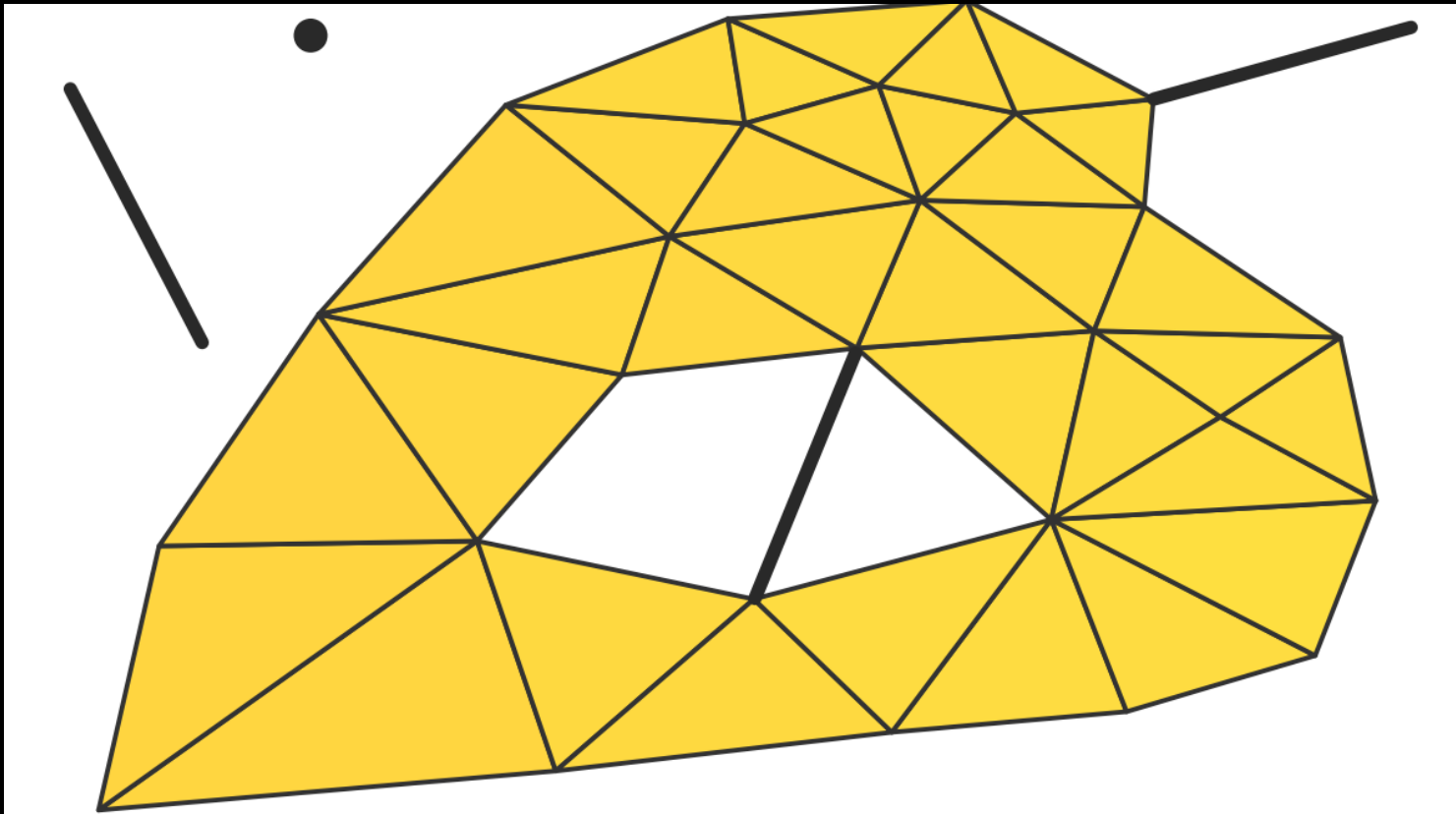
Repairing Guidelines

- What is the upstream application? (**trimmed NURBS surfaces**)
 - Determines characteristics and defects of input
- What is the downstream application? (**manifold triangle meshes for FEM**)
 - Determines requirements on output
- Based on this information,
 - is it necessary to repair the input?
- If repairing is necessary,
 - is there an algorithm that does it directly?
- If direct repair is not possible,
 - can several algorithms be used in sequence?
- If not,
 - there is a gap in the state-of-the-art.

Outlines

- Definitions
- **Defects and flaws**
- Upstream and Downstream applications
- Types of input
- Approaches

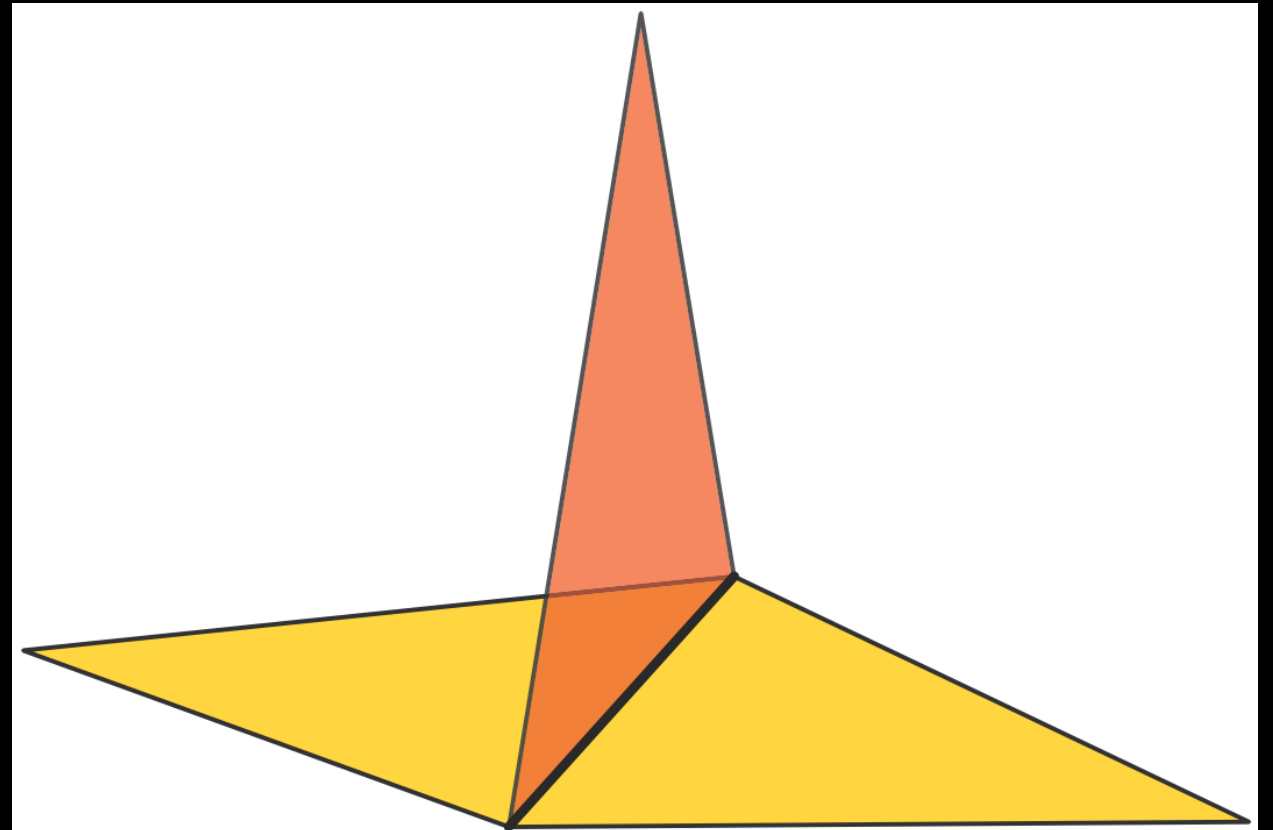
Isolated Vertices and Dangling Edges



Isolated & Dangling Elements

Singular Edges

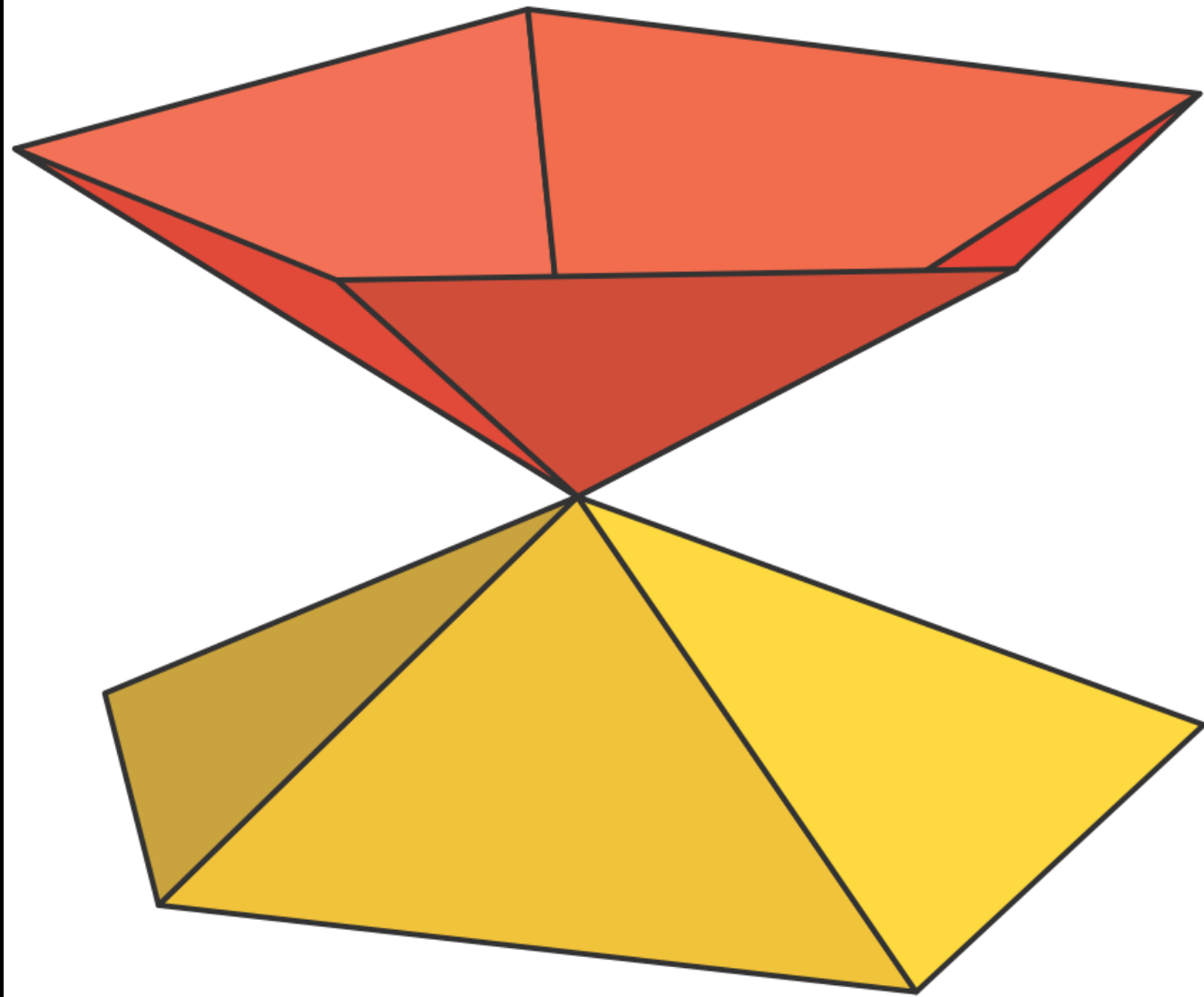
- When more than two polygons share a common edge, then such an edge is said to be singular, complex, or nonmanifold.
- Detection
 - count the number of incident triangles



Singular Edge

Singular Vertices

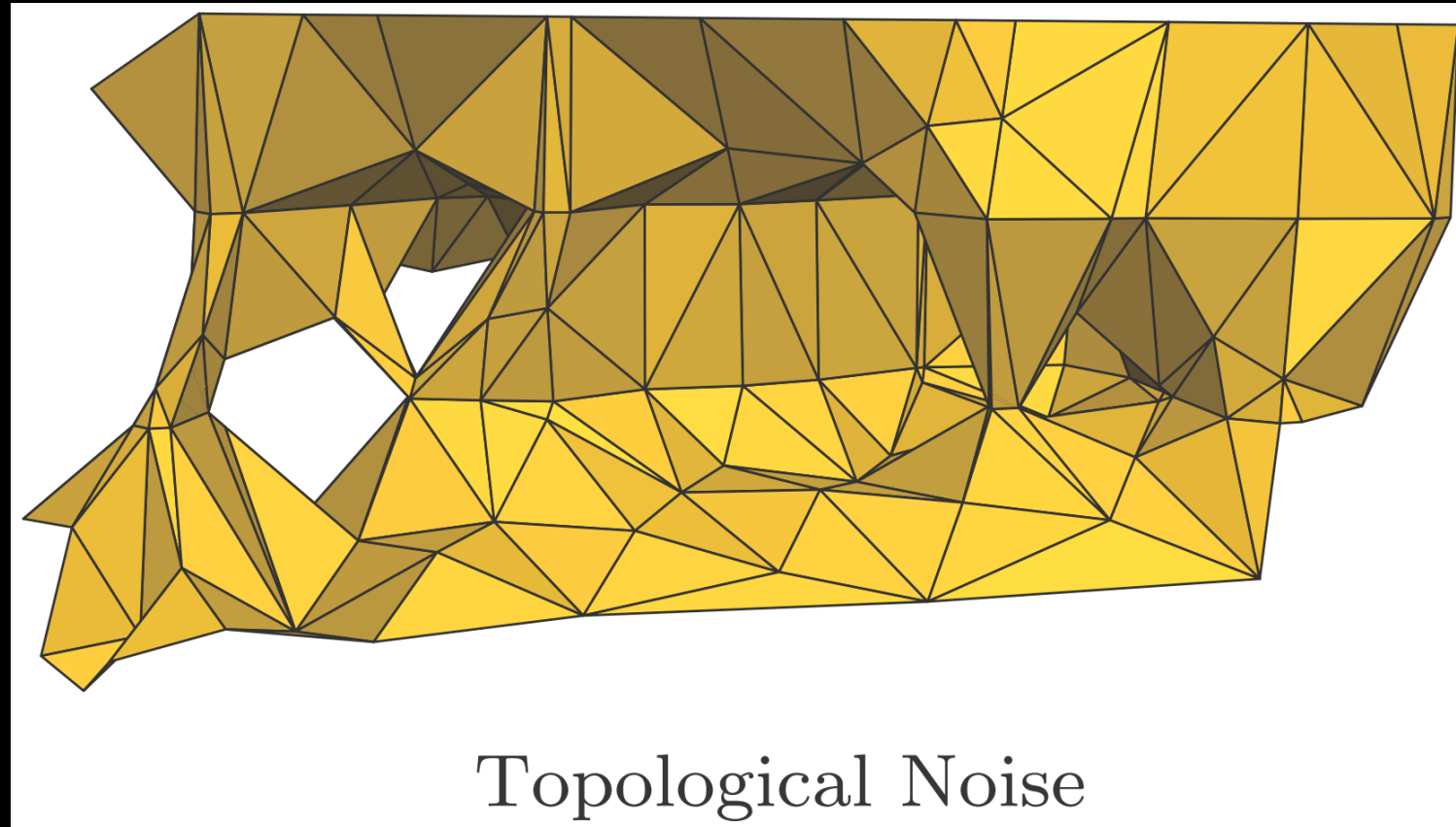
- When a vertex is not manifold in the topology of the abstract simplicial complex, it is called a combinatorially singular vertex.
- Detection
 - count the number of connected components in the neighborhood



Singular Vertex

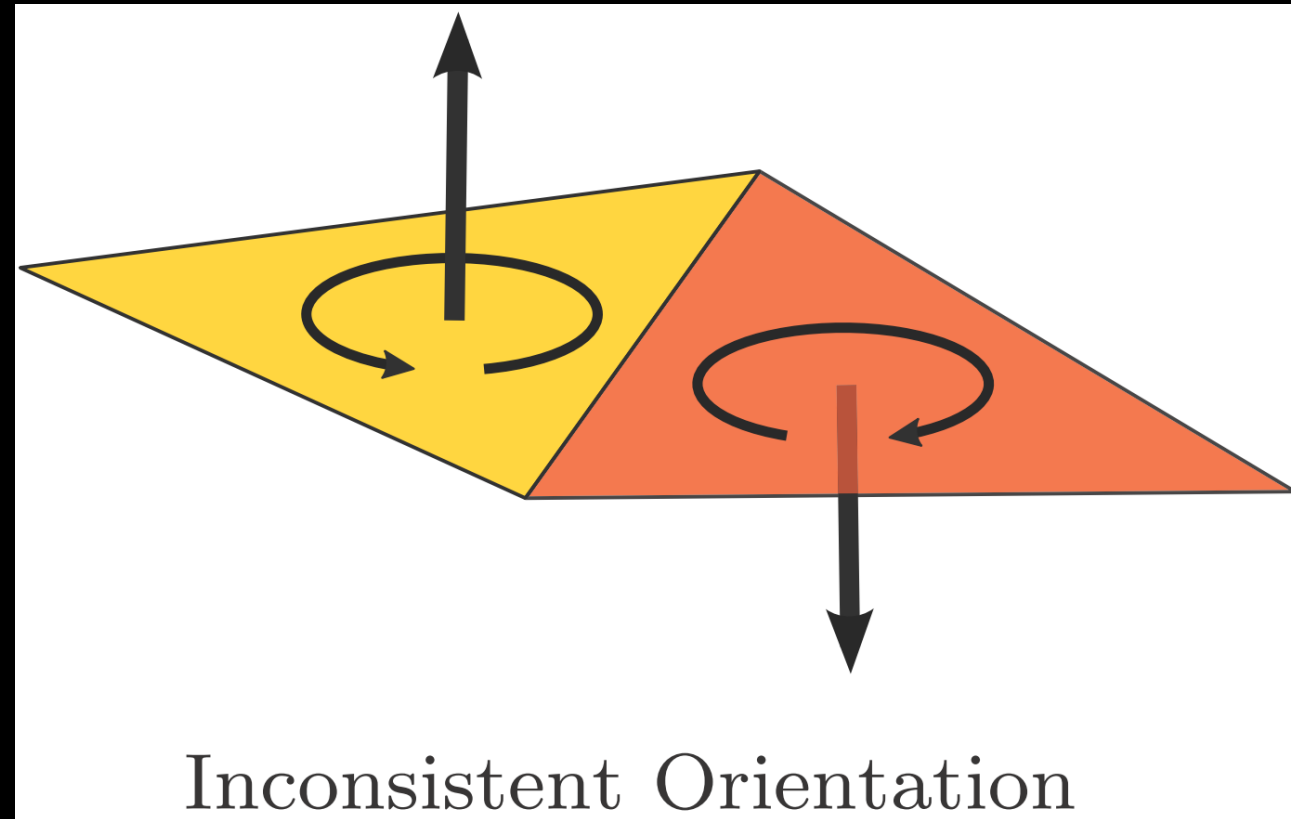
Topological Noise

- Often, in these processes **tiny handles or tunnels**, which were not present in the original object, are introduced in the constructed digital model due to **aliasing effects or noise** in the discrete underlying data.



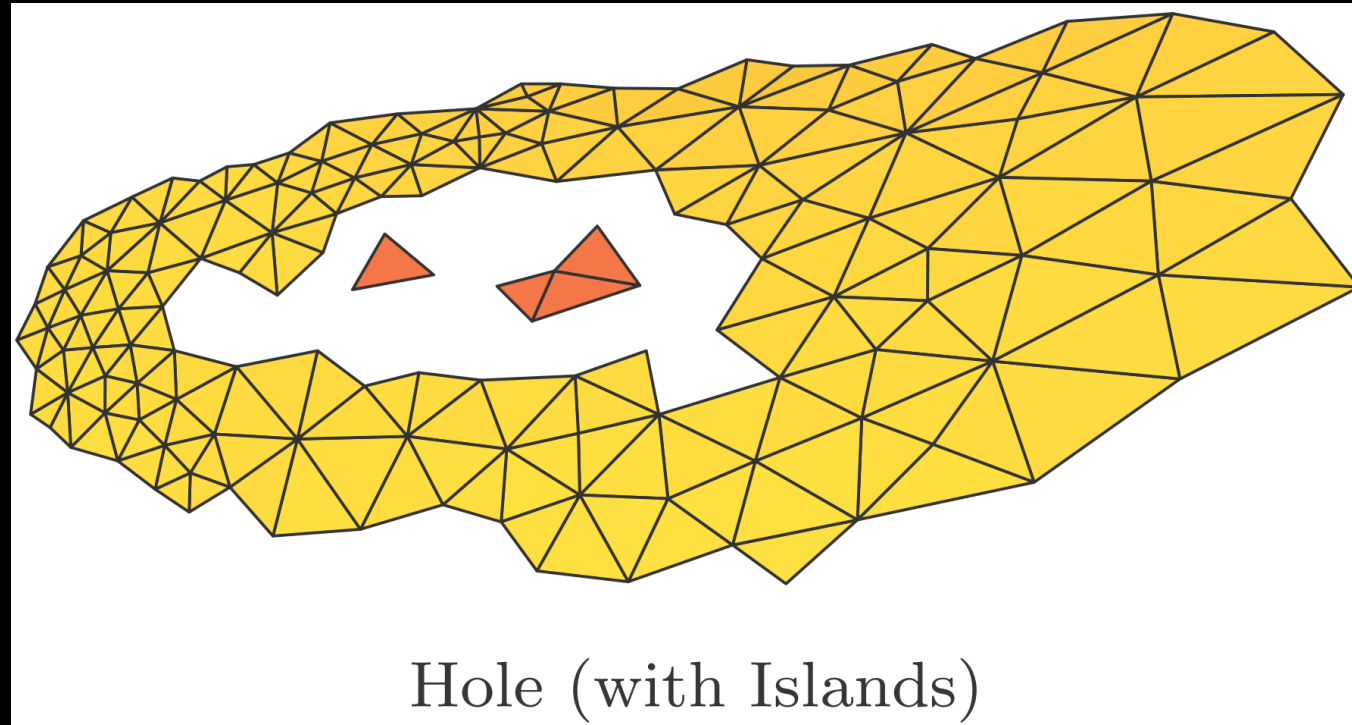
Orientation

- Polygons in an indexed face set are represented through sequences of vertex indices.
- This is typically achieved by selecting a **seed** face and by **propagating** the orientation to neighboring faces.
- Nevertheless, some configurations are intrinsically not orientable.



Surface Holes

- When digitizing a real-world object through standard **laser range scanners**, it is usual to encounter **occluded parts** which cannot be captured because the laser beam is **shadowed** by other parts of the object.
- A hole is an undesirably missing piece of surface within a triangulated patch.

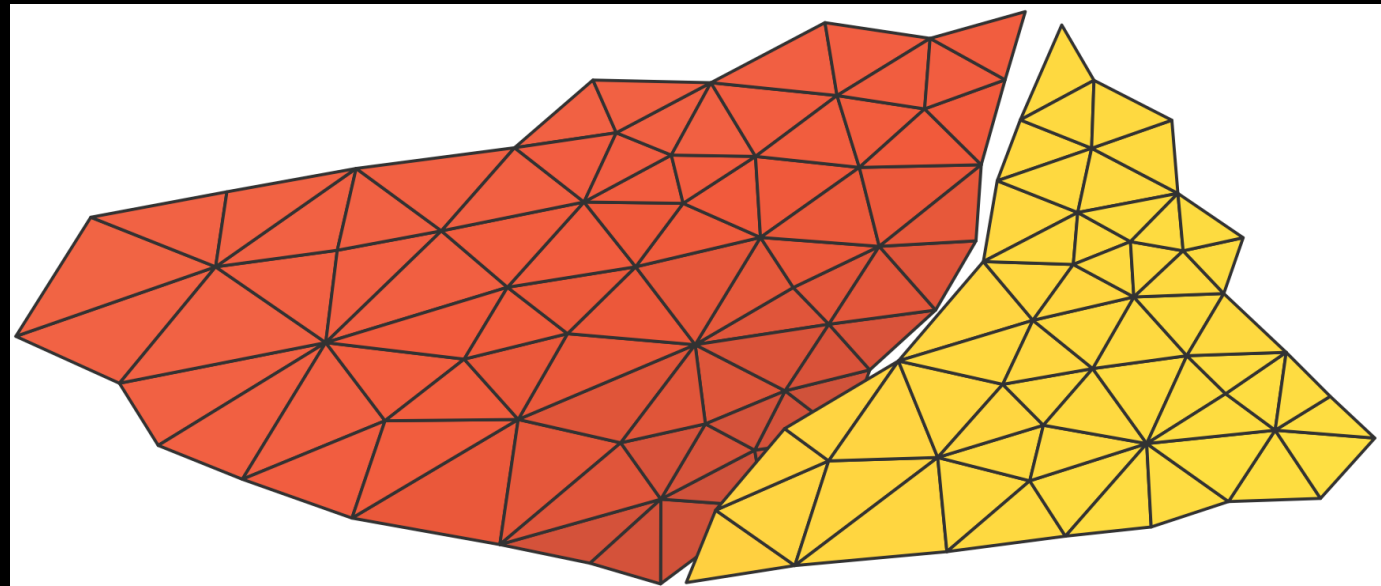


Surfaces holes

- The boundary of a hole normally consists of one or more closed edge loops.
- Holes might represent larger areas of missing data.
 - Challenge of conceiving a plausible geometry to fill the holes
- May contain so-called islands.

Gaps

- When designing a surface through standard CAD systems, the various tessellated patches are typically slightly displaced in a way that—though the intention of the designer was to construct a continuous surface—**adjacent patches are separated by undesired gaps.**



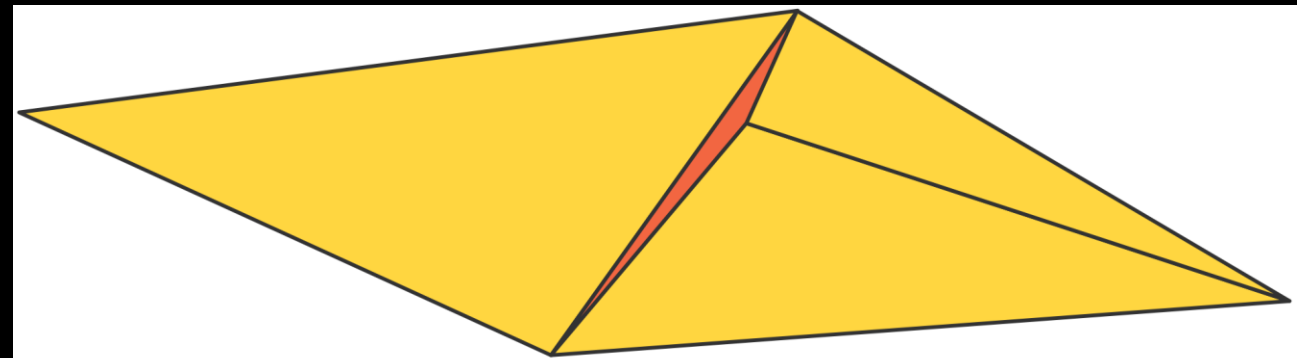
Gap (with partial Overlap)

Gaps

- A gap is defined as the **empty region** between two triangulated surface patches that should be continuously connected but are not due to the gap.
- The boundary of a gap, indeed, is typically made of two (or more) disconnected chains of edges.
- Quite narrow.

Degenerate Elements

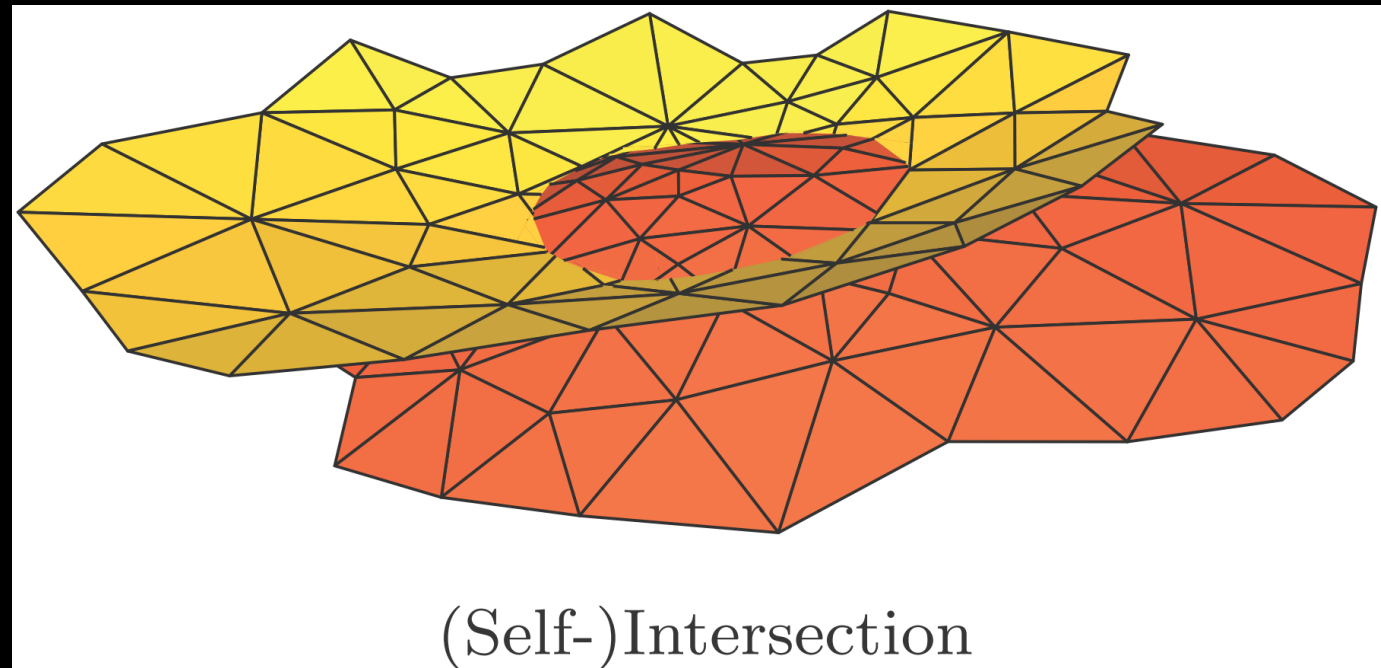
- Degenerate triangles are triangles with zero area.
- These elements **are the source of several problems for numerous applications**, since many useful entities cannot be computed on such triangles (normal vectors, circumscribing circles, barycentric coordinates, etc).



(Near) Degeneracy

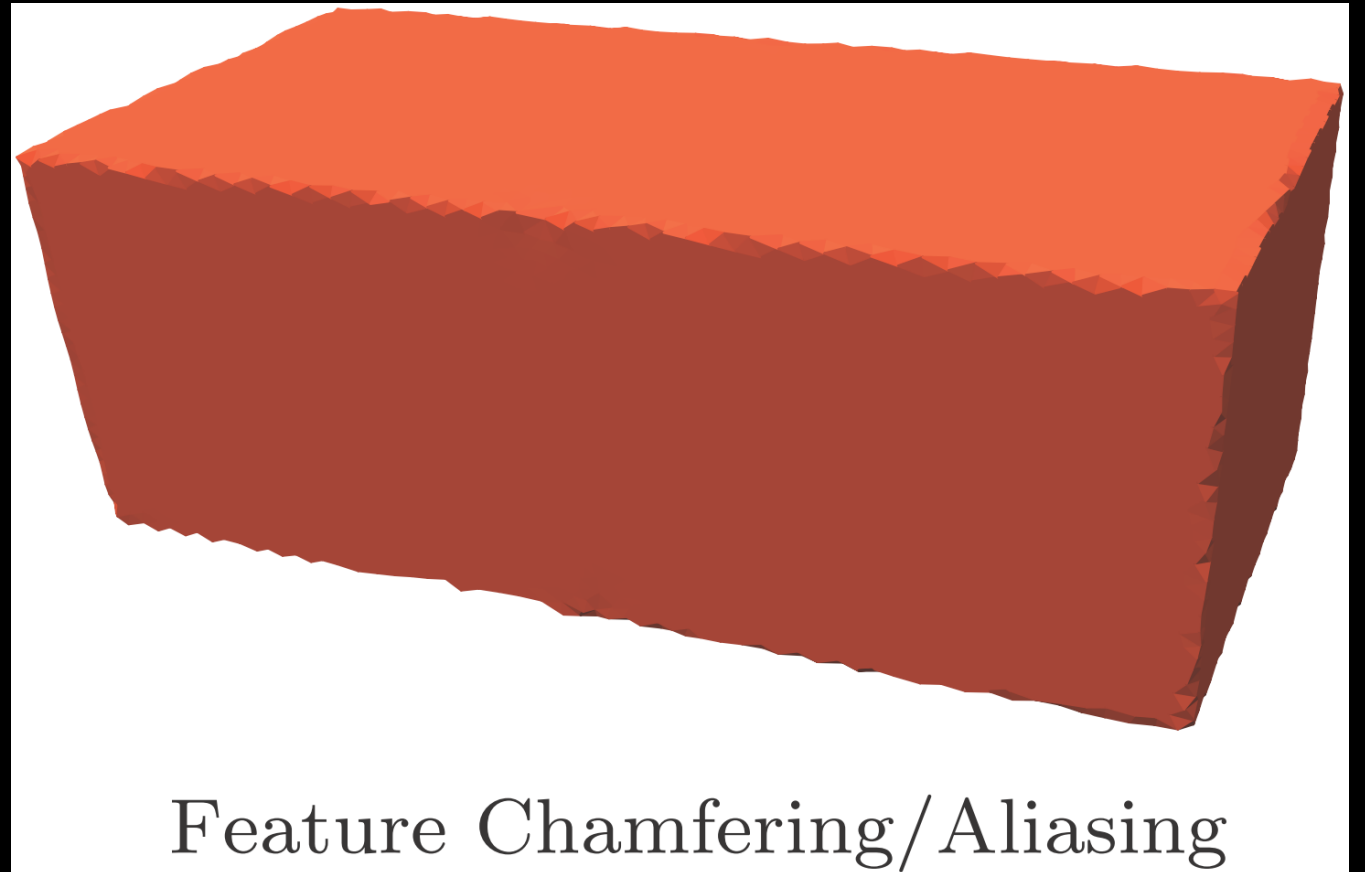
Self-Intersections

- Self-intersecting meshes are typically generated
 - by tessellation of multipatch CAD models,
 - by deformation of mesh models,
 - by composing models out of multiple parts without care,
 - or when merging patches reconstructed from partial scans of a 3D object.
- Due to the ambiguities, there is no common strategy to tackle this problem.



Sharp Feature Chamfering

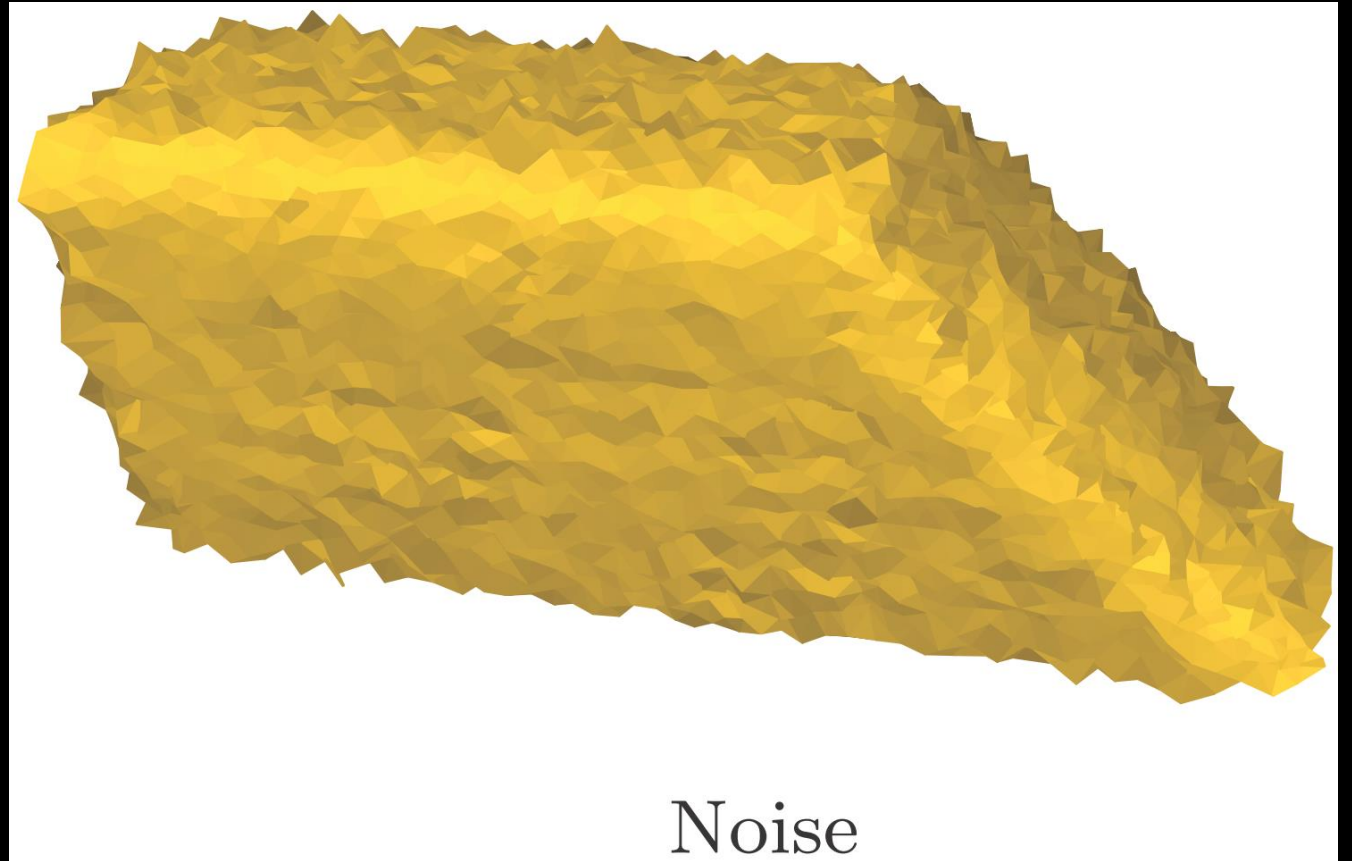
- The sharp edges and corners of the original shape are removed by the sampling process and replaced by irregularly triangulated chamfers.
- Having such well-defined sharp features has clear advantages for both visualization and reverse engineering.



Feature Chamfering/aliasing

Data Noise

- Every digitization tool has a finite precision.
- Thus, the acquired raw data of the sampled model contains additive noise from various sources.
- A main challenge is to remove the noise while preserving the main morphology of the underlying sampled surface.



Outlines

- Definitions
- Defects and flaws
- **Upstream and Downstream applications**
- Types of input
- Approaches

Upstream applications

- Determines characteristics and defects of input
- The origin of defects in a mesh: Nature and Approach
- Nature of the data modeled
 - (physical) real-world data
 - (virtual) concepts
- Approach employed to convert such data into polygon meshes

Nature

- If a model is designed, the basic concept is typically an **abstraction**.
- Downstream applications may face problems such as **nonmanifoldness, gaps, and intersections**.
- These defects are either caused by **inaccuracies** in modeling or produced by **description processes** that are often based on surface representations although solids are meant to be created.

Nature

- if the model is digitized, problems are mostly in the measured data.
- May include **noise, holes, chamfered features, and topological noise**
- Due to **limitations of the measurement process** employed for digitization.

Approach

- Such abstraction/data is converted into a polygon mesh (if not originally designed in polygonal form).
- The conversion itself can be the source of further flaws that depend on the specific approach used.
 - For example, a CAD model, gaps and intersections might arise due to the necessarily occurring **deviation of each triangulated patch from the original curved surface**.
 - Depending on the quality of the tessellation algorithm also **(near-)degenerate polygons** might be created.

Downstream applications

- Determines requirements on output
- Visualization
 - only the existence of **significant holes** is generally deemed unacceptable; all other types of defects can often be neglected.
 - To achieve pleasing renderings of a certain visual quality, however, also **noise, gaps, and chamfered features** can be adverse.

Downstream applications

- Modeling

- Connected surfaces **without degeneracies** are usually required.
- Intersections are often acceptable in the case of surface-based methods.
- Singularities and topological noise do not cause problems for some methods, others require or prefer clean manifold meshes.

- Rapid prototyping

- The mesh model naturally needs to be convertible to a solid model, that is, it has to **well-define an interior and exterior volume**
- So the mesh definitely has to be **closed** and **free of intersections and singular non-manifold configurations** that would prevent an unambiguous volume classification.

Downstream applications

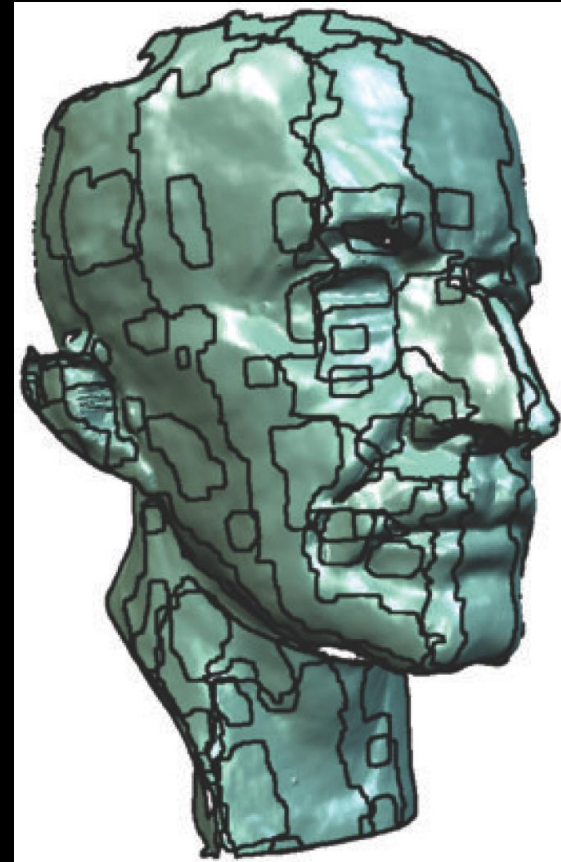
- Geometry processing
 - The input mesh is additionally required to **be free of degeneracies and noise** in order to allow for the computation of element properties and discrete differential quantities in a reasonable way.
 - Aliasing effects **like topological noise and chamfered features** negatively affect and disturb several of these methods.
- Simulation (FEM) of real-world phenomena on digital models
 - The **highest (all)** requirements on the model's quality in order to be able to achieve reliable results.

Outlines

- Definitions
- Defects and flaws
- Upstream and Downstream applications
- **Types of input**
- Approaches

Registered range scans

- A set of patches (usually triangle meshes) that represent **overlapping** parts of the surface S of a scanned object.
- The main geometric problem in this setup is the potentially **very large overlap** of the scans.
 - a point x on S is often described by multiple patches
- Each patch has its **own connectivity** that is usually not compatible to the connectivity of the other patches.



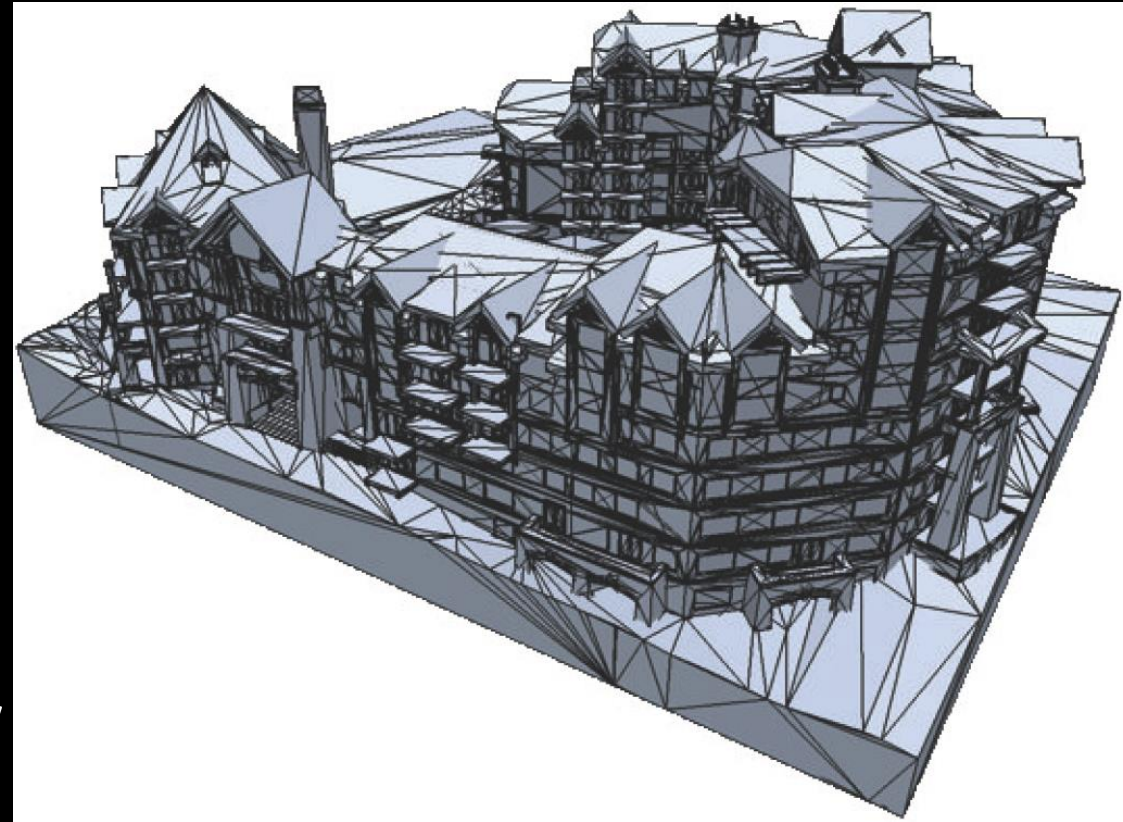
Fused range scans

- Manifold meshes with boundaries (i.e., gaps, holes, and islands).
- Either these artifacts are due to **obstructions** in the line of sight of the scanner
- Or they result from **bad surface properties** of the scanned model, such as transparency or glossiness.



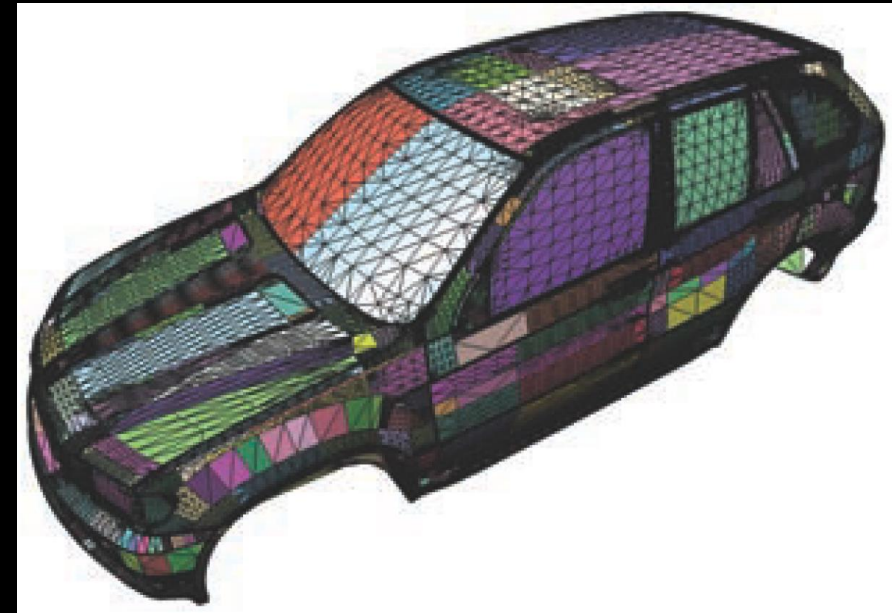
Triangle soups

- Mere sets of triangles with little or no connectivity information.
- They most often arise in CAD models
 - manually created in a boundary representation where users typically assemble predefined elements (taken from a library) without bothering about consistency constraints.
- Due to the manual layout, these models typically are made of only a few thousands triangles, but they may contain all kinds of artifacts.



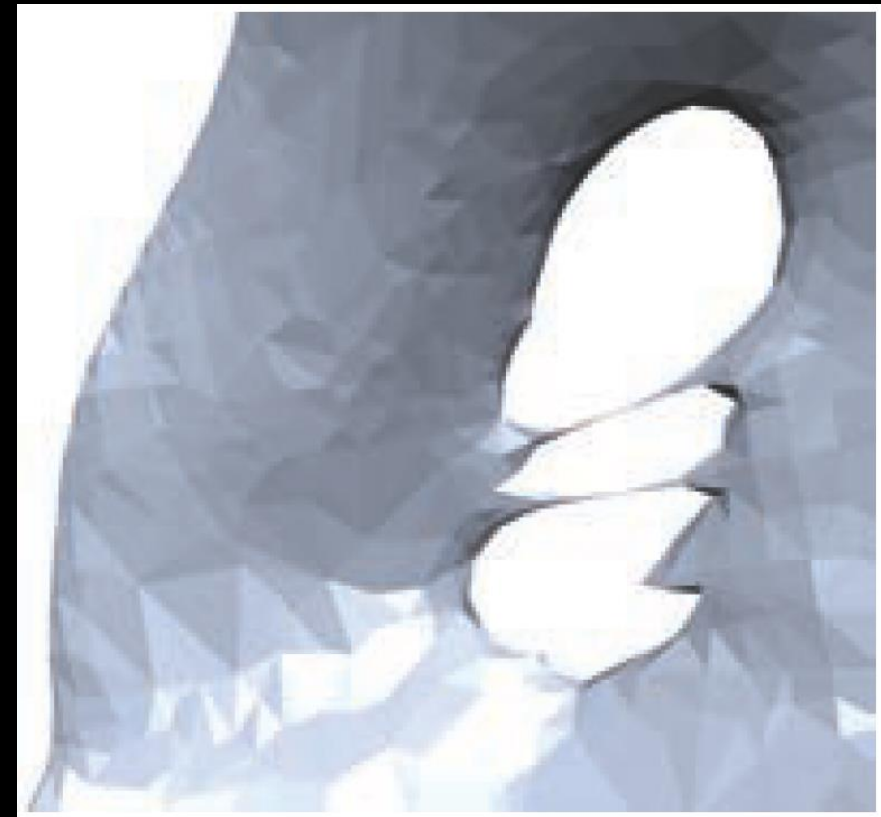
Triangulated NURBS patches

- A set of **connected triangle mesh patches** that contain gaps and small overlaps along the boundaries of the patches.
 - intersecting patches and inconsistent normal orientations.
- These artifacts arise when triangulating two or more trimmed NURBS patches **that join at a common boundary curve**.
- Usually, each patch is triangulated separately; thus the common boundary is sampled differently from each side.



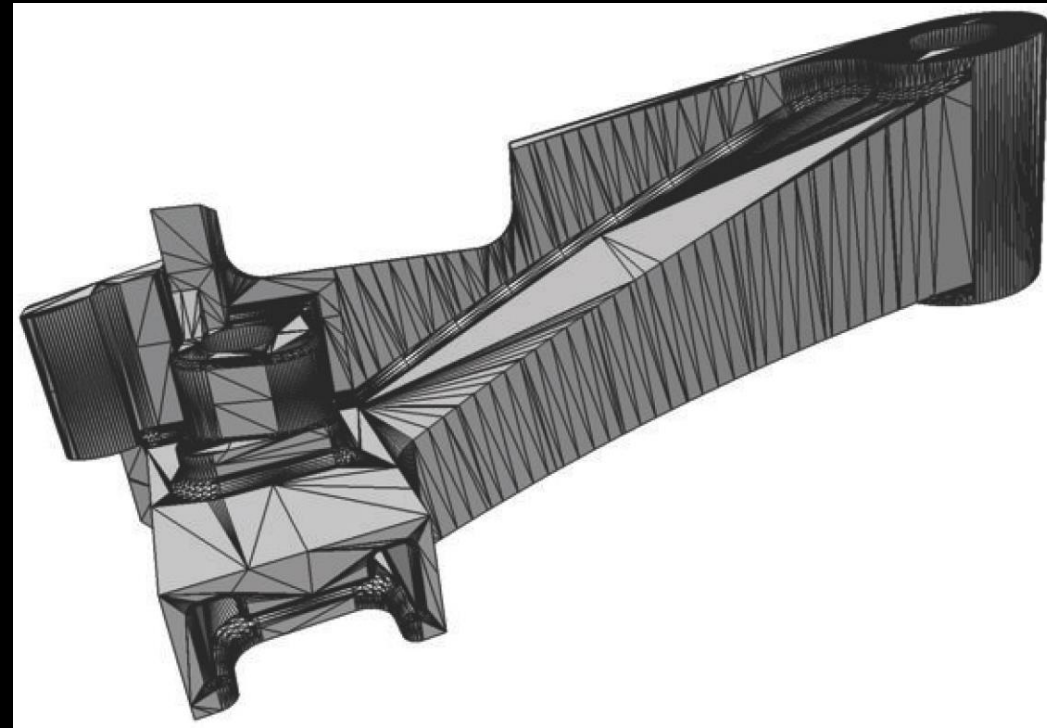
Contoured meshes

- Meshes have been extracted from a **volumetric dataset** by Marching Cubes, Dual Contouring, or other polygon mesh extraction algorithms.
 - signed distance field
- These meshes often contain other **topological artifacts**, such as small spurious handles.
- Due to the **finite resolution** of the underlying grid, voxels are often classified incorrectly, leading to the so-called **partial volume effect**.



Badly meshed manifolds

- Degenerate elements such as **triangles with zero area**, **caps** (one inner angle close to π), **needles** (one edge length close to zero), and **triangle flips** (normal jump between adjacent faces close to π).
- From the tessellation of CAD models
- Output of Marching Cubes
 - in particular if they are enhanced by feature-preserving techniques
- The degenerate shapes of the elements prevent further processing and lead to instabilities in numerical simulations.



Outlines

- Definitions
- Defects and flaws
- Upstream and Downstream applications
- Types of input
- **Approaches**

Surface-oriented algorithms

- **operate directly on the input data** and try to explicitly identify and resolve artifacts on the surface.
- only **minimally perturb the input model** and are able to preserve the polygonal mesh structure in areas that are not in the direct vicinity of artifacts.
- **Gaps** could be removed by snapping boundary elements (vertices and edges) onto each other or by stitching triangle strips in between the gap.
- **Holes** can be closed by filling in a triangulated patch that is optimal with respect to some surface quality functional.
- **Intersections** could be located and resolved by explicitly splitting edges and triangles.

Surface-oriented algorithms (downside)

- To guarantee a valid output, surface-oriented repair algorithms usually require that **the input model already satisfy certain quality requirements**
 - Often enough these requirements cannot be guaranteed nor even be checked automatically, so these algorithms are **rarely fully automatic** but instead need user interaction and manual post-processing.
- Due to numerical inaccuracies, certain types of artifacts (like intersections or large overlaps) cannot be resolved robustly.
- Other artifacts, like gaps between **two separate solids** that are geometrically close to each other, **cannot even be identified**.

Consistent Normal Orientation

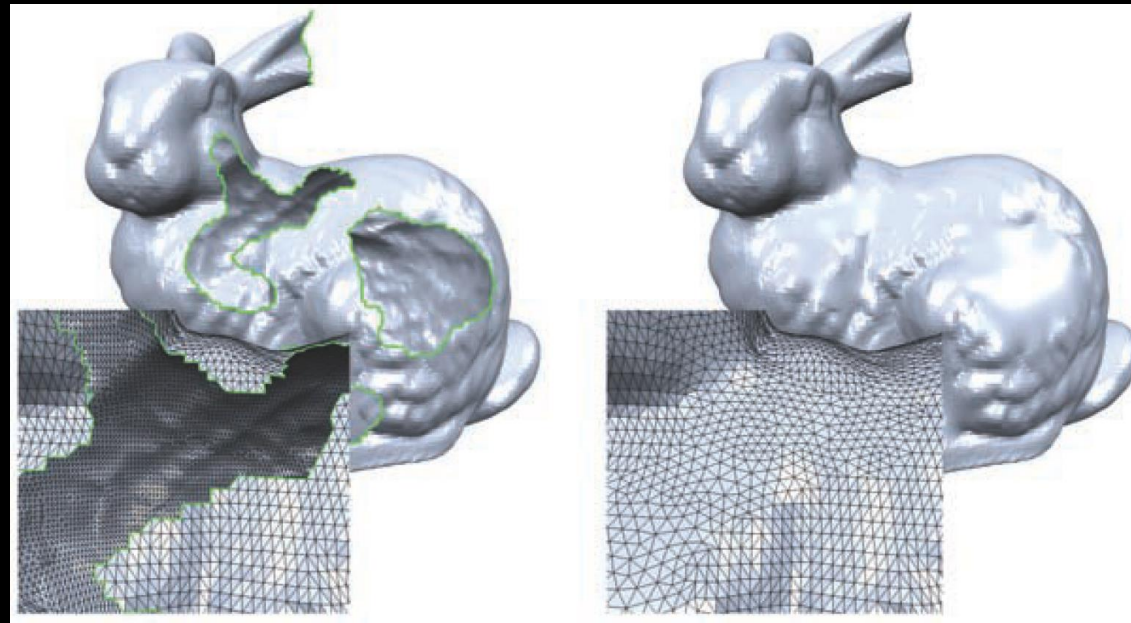
- Consistently orienting the normals of an input model is part of most surface-oriented repair algorithms
- Can even improve the performance of volumetric algorithms.
- Usually the orientation of the normals is propagated along a minimum spanning tree between neighboring patches.

Surface-Based Hole Filling

- Describe an algorithm for computing a smooth triangulation of a hole.
- **First**, the holes are **identified** and **filled** by a coarse triangulation.
- These patches are **then refined** such that their vertex densities and average edge lengths match those of the mesh surrounding the holes.
- Finally, the patch is **smoothed** so as to blend with the geometry of the surrounding mesh.

Surface-Based Hole Filling

- This algorithm reliably fills holes in models with **smooth** patches.
- The density of the vertices matches that of the surrounding surface.
- does not check or avoid geometric self-intersections
- does not detect or incorporate islands into the filling patch



Conversion to Manifolds

- All complex edges and singular vertices are identified by counting the number of adjacent faces.
- The input is then cut along these complex edges into separate manifold patches.
- Finally, pairs of matching edges (i.e., edges that have geometrically the same endpoints) are identified and merged, if possible, in a topologically consistent manner.
- This, however, is done efficiently and robustly.

Gap Closing

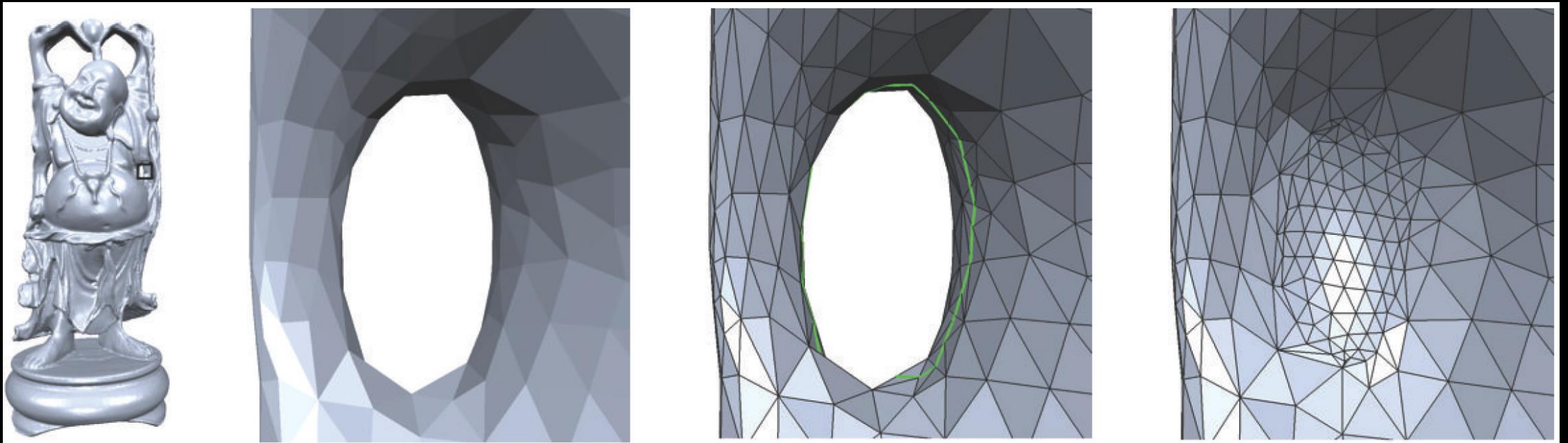
- Typical for triangulated NURBS models.
- For each pair of boundary edges, the area between the two edges normalized by the edge lengths is computed.
- This score measures the geometric error that would be introduced by merging the two edges.
- Pairs of boundary edges are then iteratively merged in order of increasing score.

Gap Closing

- Usually easy to implement
- If the input data is **well behaved** and the **user parameters** are chosen in accordance with the error that was accepted during triangulation, they manage to produce satisfying results.
- However, there are **no guarantees** on the quality of the output.
- Allows the user to override the decisions towards the expected result.

Topology Simplification

- Detects and resolves all **handles** up to a given size ε in a manifold triangle mesh.
- Handles are removed by cutting the input along a non-separating closed path and sealing the two resulting holes by triangle patches



Topology Simplification

- Detection
 - Dijkstra's algorithm on the dual graph from a seed triangle
 - When two different loops touch along a common, a handle is detected
 - To detect all handles of the input mesh, one has to perform the region growing for every triangle.
- Downside
 - cannot guarantee that no geometric self-intersections are created after a handle is removed.

Volumetric algorithms

- Convert the input model into an intermediate volumetric representation from which the output model is then extracted.
 - fully automatic and produce guaranteed watertight models
- A volumetric representation can be any kind of partitioning of the embedding space into cells such that each cell can be classified as being **inside**, **outside**, or **intersected** by the surface.
- Volumetric representations: **regular Cartesian grids, adaptive octrees, kd-trees, BSP-trees, and Delaunay triangulations.**
- Do not allow for artifacts like intersections, holes, gaps, overlaps, or inconsistent normal orientations.
- Often also guarantee the absence of complex edges and singular vertices
- Spurious handles, however, might still be present.

Volumetric algorithms (downside)

- The conversion to and from a volume leads to **a resampling of the model**
 - Introduces **aliasing artifacts** and **loss of model features**
 - destroys any structure that might have been present in the **connectivity** of the input model.
- The number of triangles in the output of a volumetric algorithm is **usually much higher** than that of the input model
 - thus has to be decimated in a post-processing step.
- The quality of the output **triangles often degrades and has to be improved afterwards.**
- Volumetric representations are quite **memory-intensive** so it is hard to run them at very high resolutions.

Volumetric Repair on Adaptive Grids

- The algorithm first **creates an adaptive octree representation** of the input model where each cell stores the triangles intersecting with it.
 - Cells that are not yet on maximum depth are recursively split if they either contain a boundary edge or **if the triangles within the cell deviate too much from a common regression plane**.
 - From these triangles **a feature-sensitive sample point** can be computed for each cell.
- Then, a sequence of **morphological** operations is applied to the octree to **determine the topology of the model**.
- Finally, the connectivity and geometry of the reconstruction are derived from **the octree structure and samples**, respectively.
 - A Dual Contouring

Volumetric Repair on Adaptive Grids

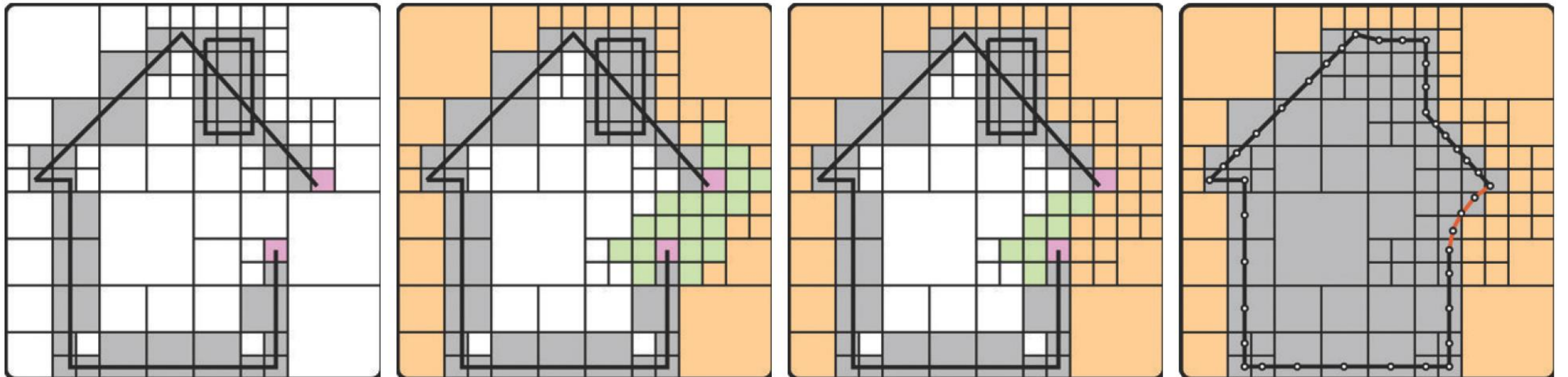


Figure 8.7. From left to right: Adaptive octree (boundary cells are marked red). Dilated boundary (green) and outside component (orange). Outside component dilated back into the boundary cells. Final reconstruction. (Image taken from [Botsch et al. 06b]. ©2006 ACM, Inc. Included here by permission.)

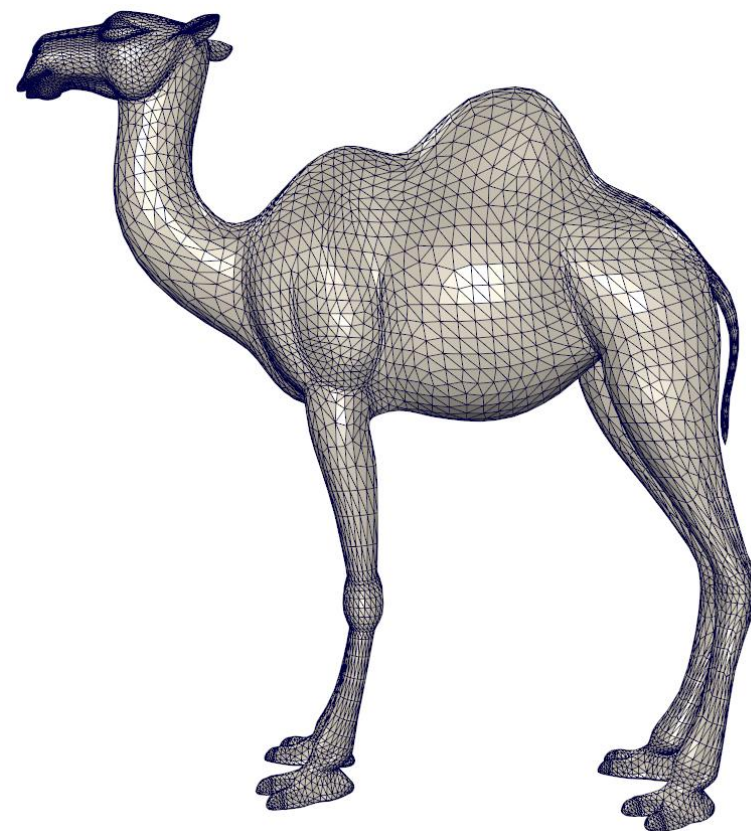
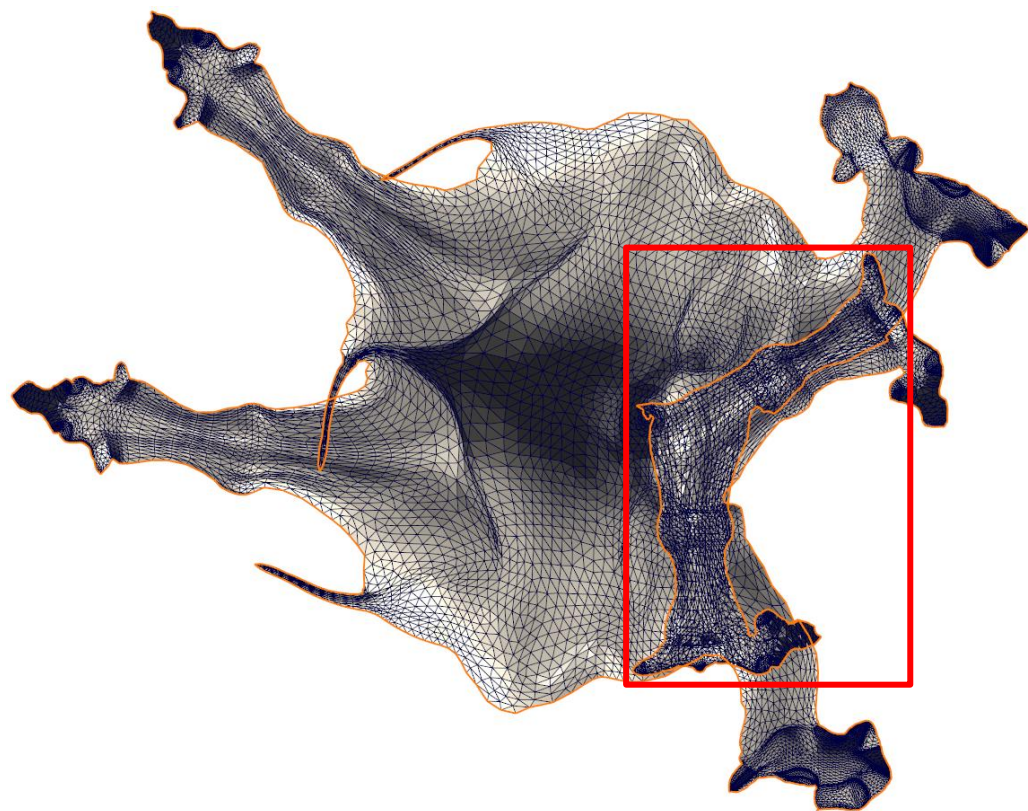
Bijjective Mappings

Jian-Ping Su

May 22, 2020

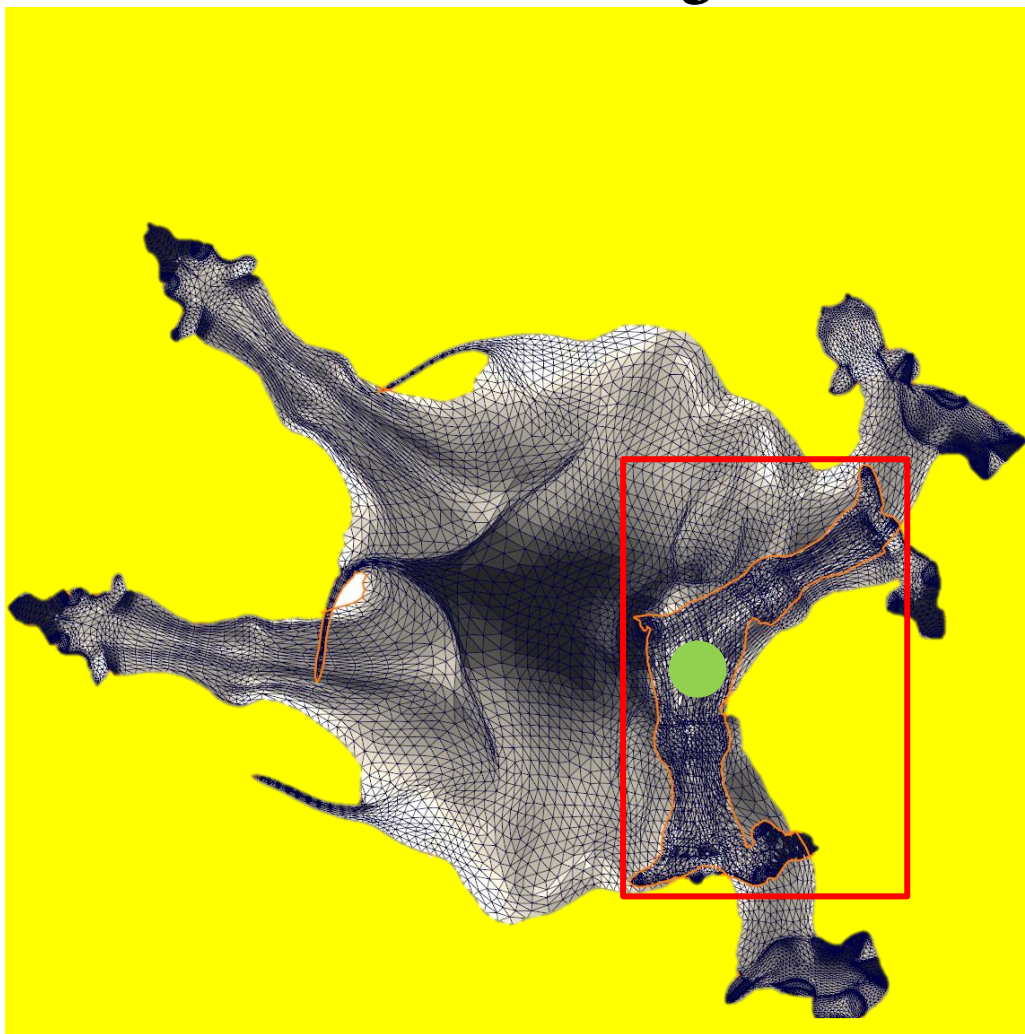
USTC 2020 Spring Digital Geometry Processing

Bijjective Mappings



Injective

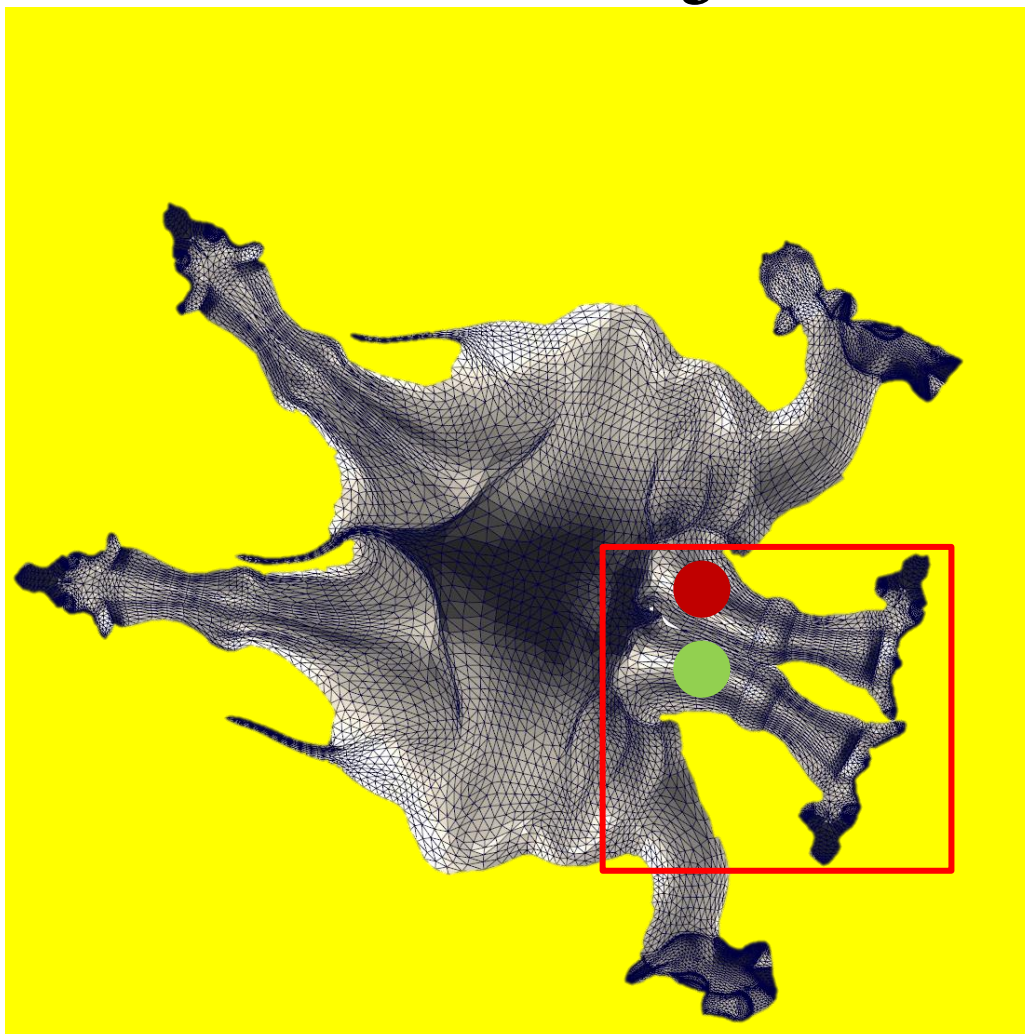
Bijjective Mappings



Injective

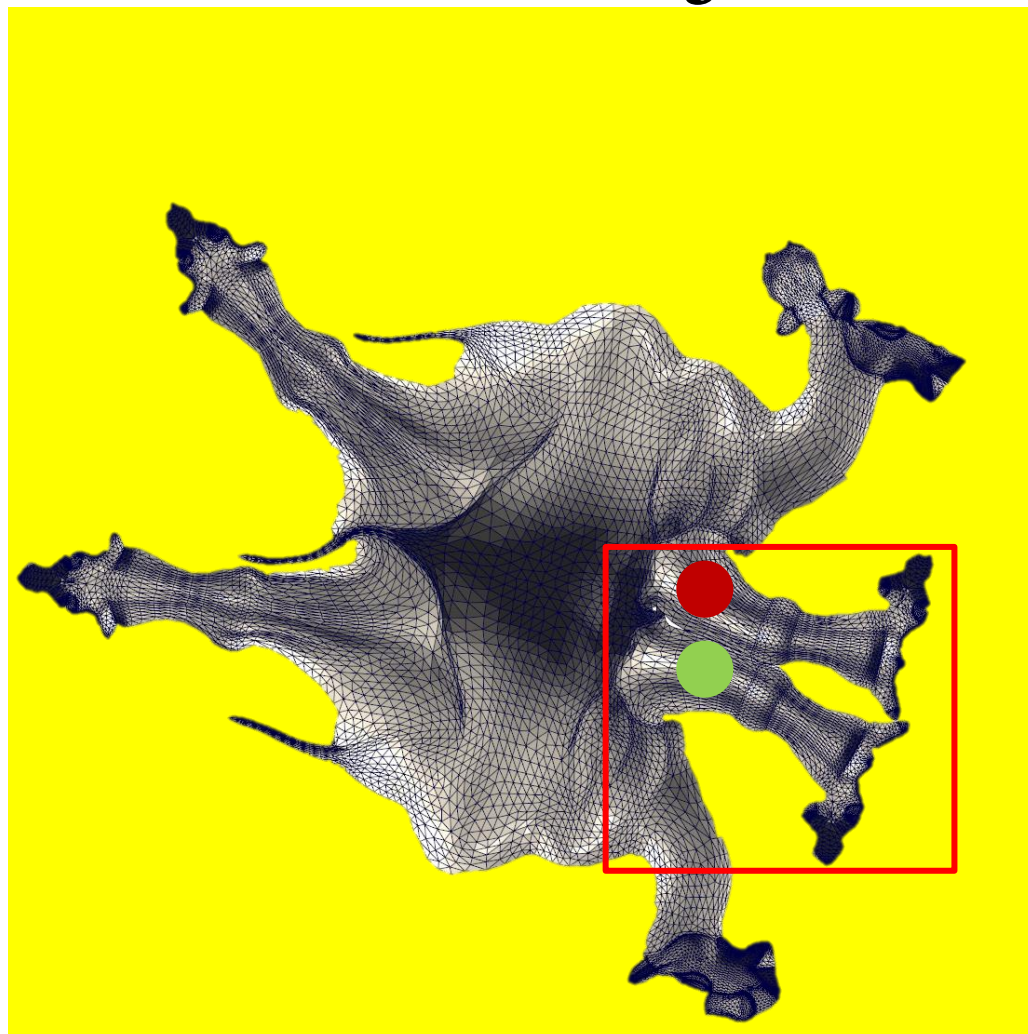


Bijjective Mappings



Bijjective

Bijective Mappings



Bijective

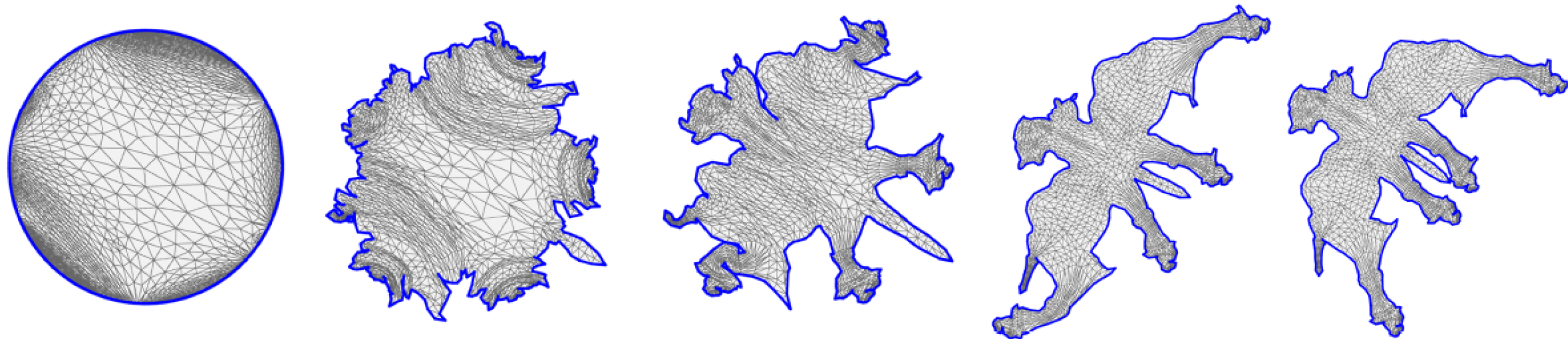
Existing work

Paper List

- Bijective Parameterization with Free Boundaries.
- Simplicial Complex Augmentation Framework for Bijective Maps.
- Efficient Bijective Parameterizations

Bijective Parameterization with Free Boundaries

Jason Smith, Scott Schaefer

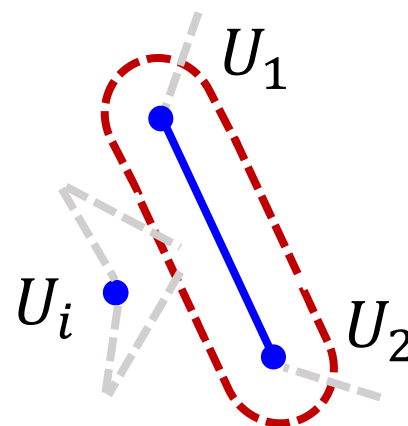
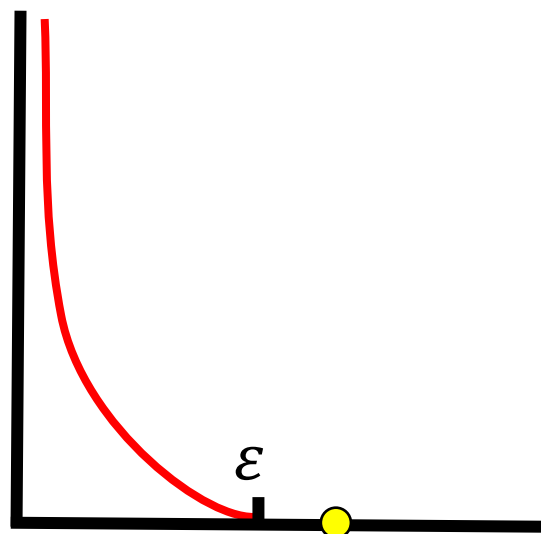


Barrier function

distortion barrier

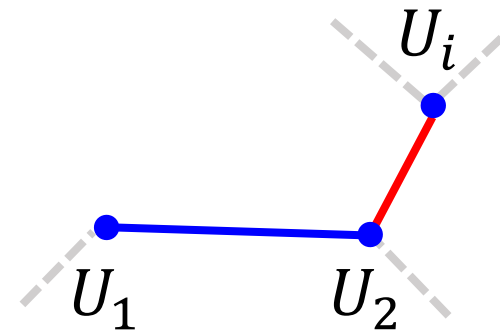
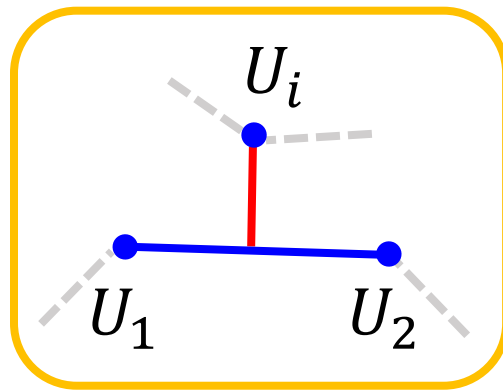
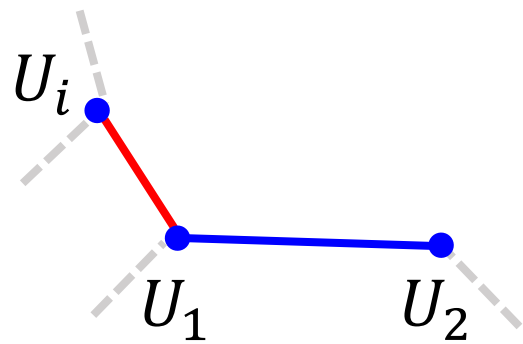
$$\min E_D + E_B$$

$$E_B = \max\left(0, \frac{\varepsilon}{\text{dist}(U_1, U_2, U_i)} - 1\right)^2$$



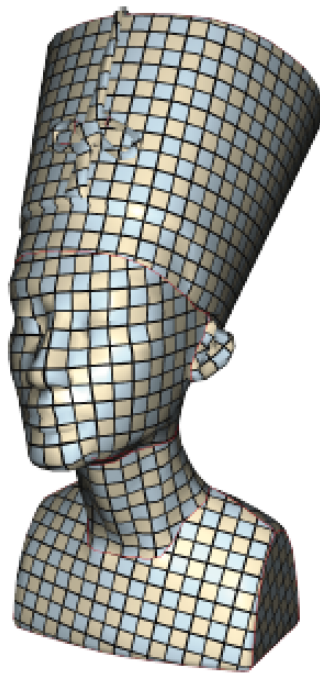
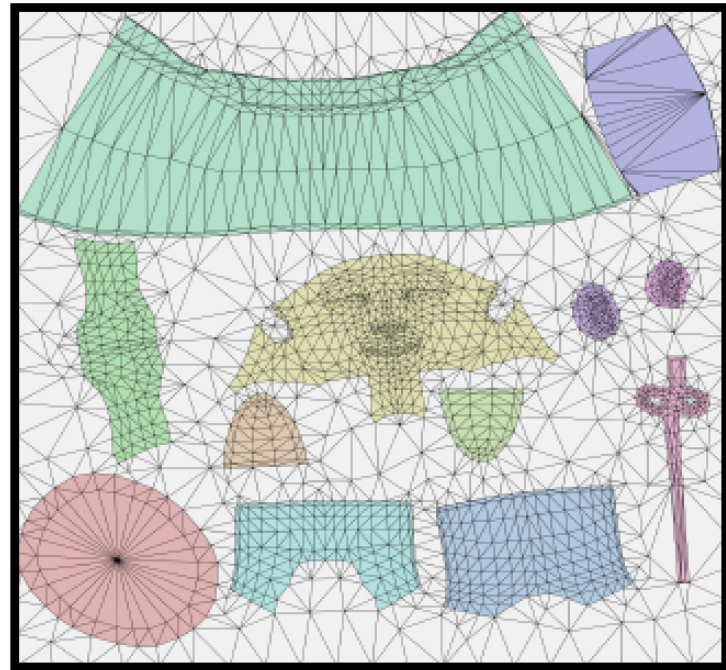
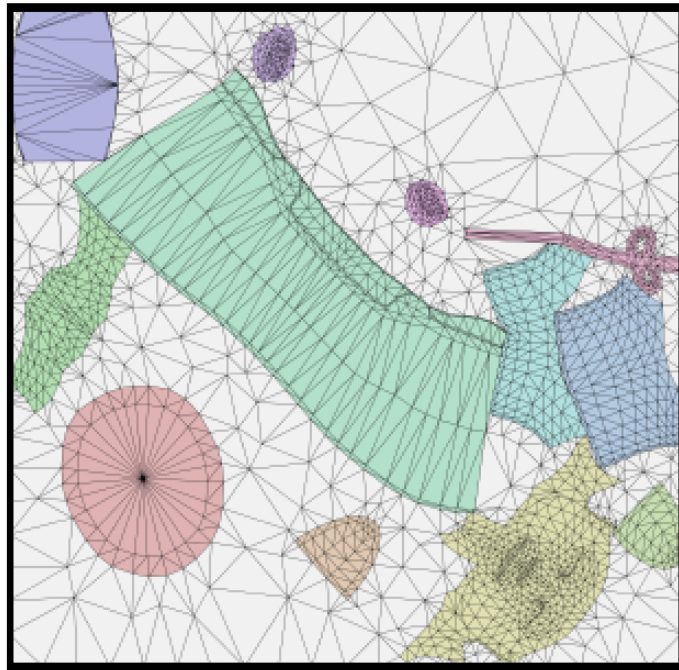
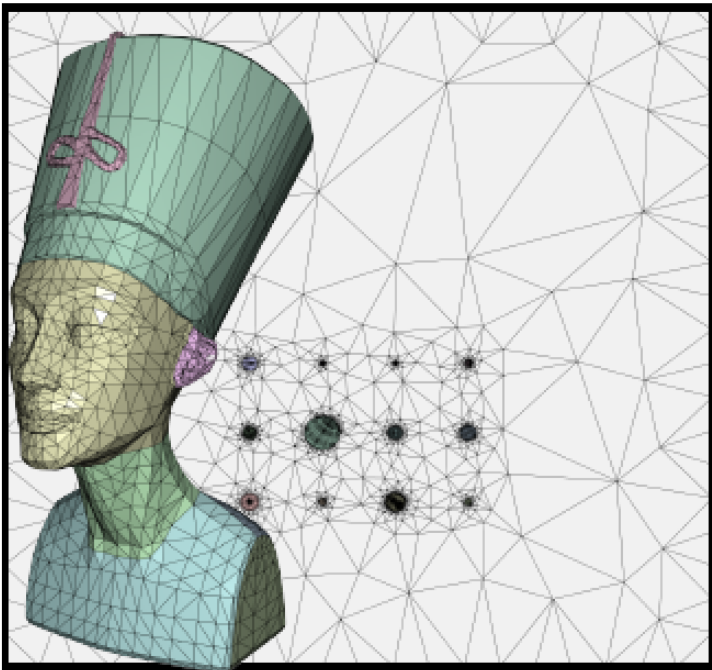
Definition of distance

- Distance is not C^2 .
- Hessian of barrier function is difficult to compute.
- Convex-concave decomposition is not easy.



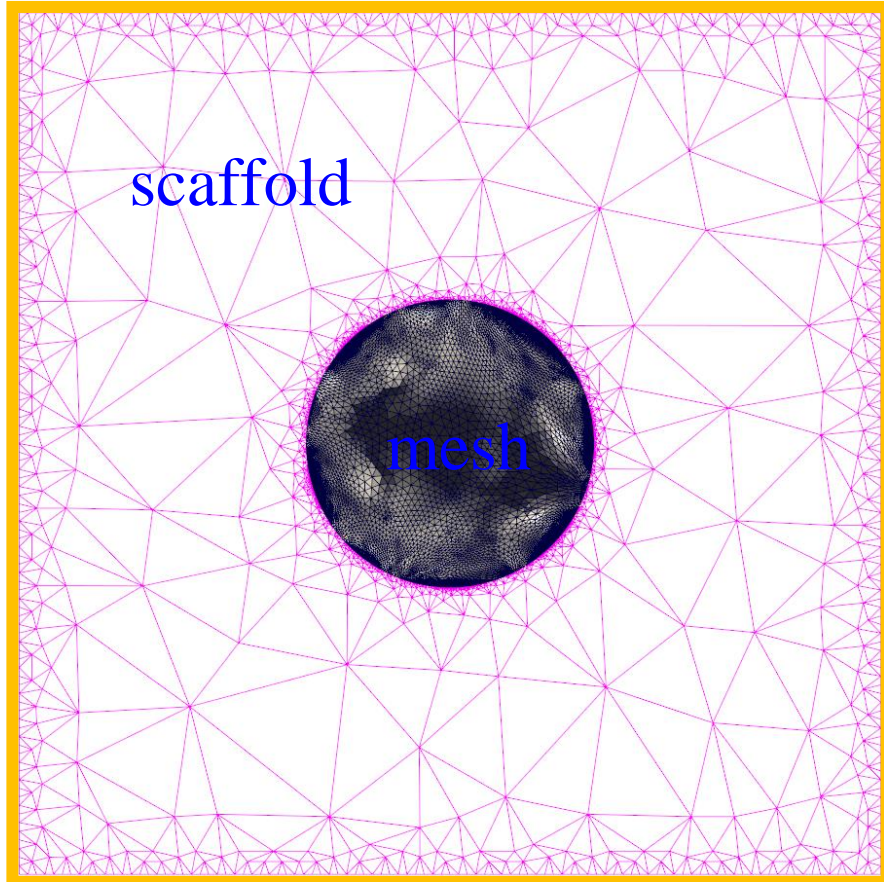
Simplicial Complex Augmentation Framework for Bijective Maps

Zhongshi Jiang, Scott Schaefer, Daniele Panozzo

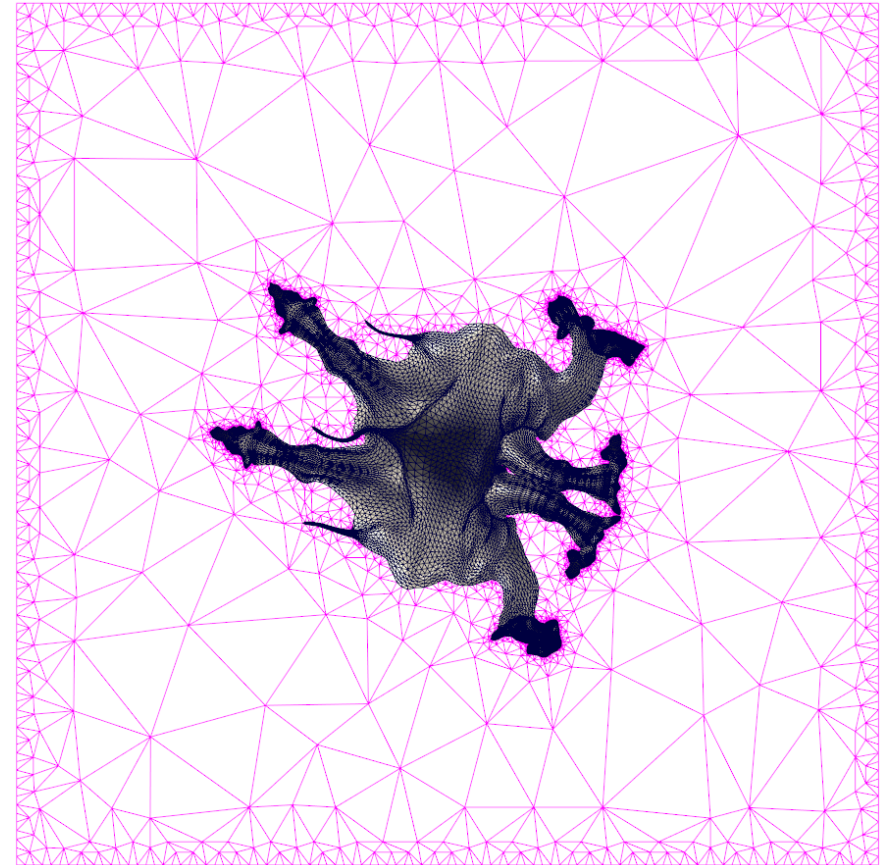


Scaffold structure

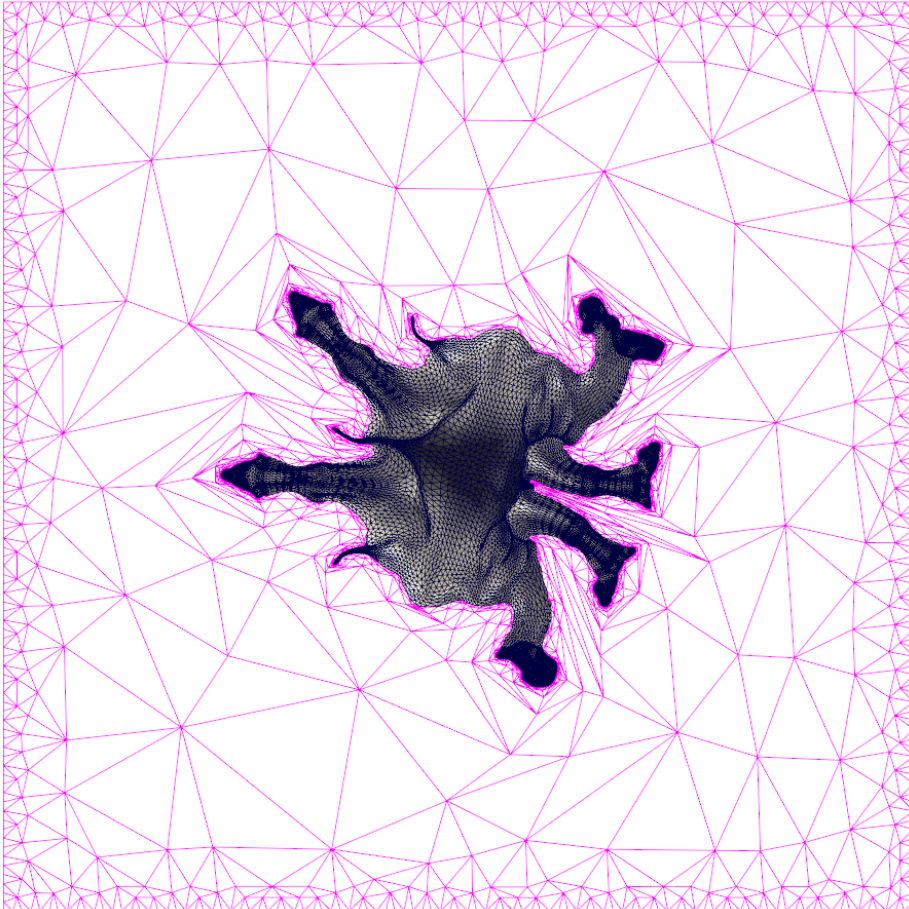
$$\min E_M + E_S$$



Fix boundary
Locally injective



Remesh



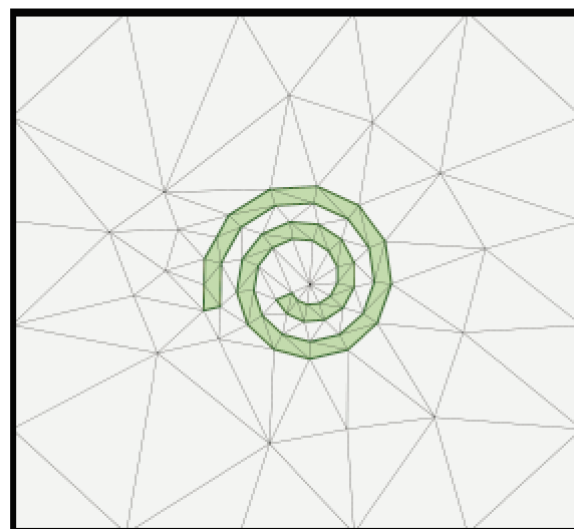
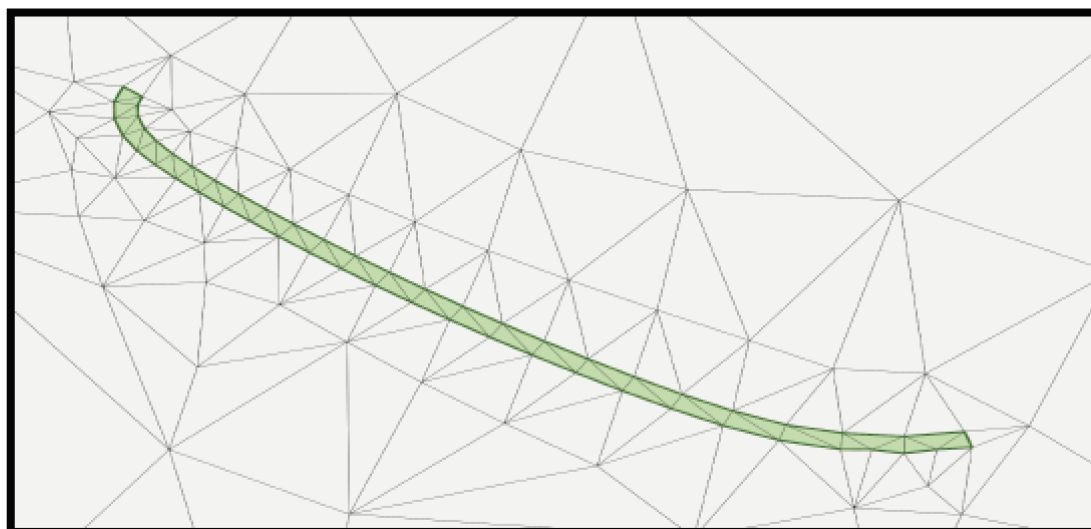
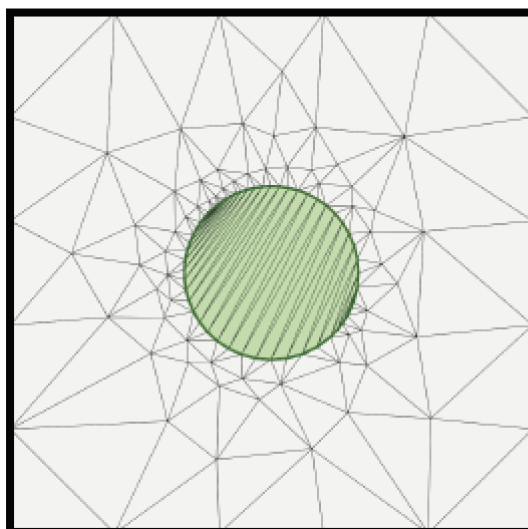
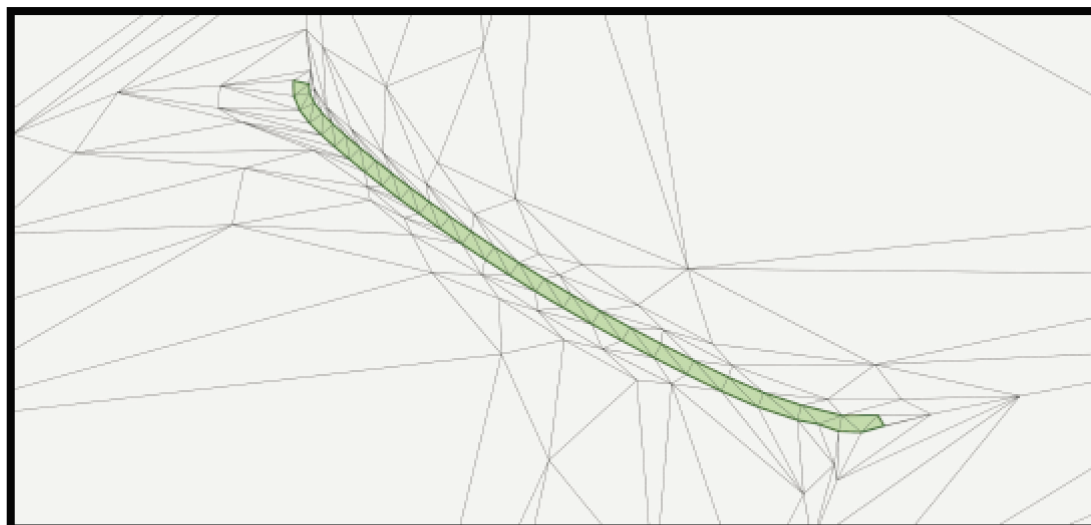
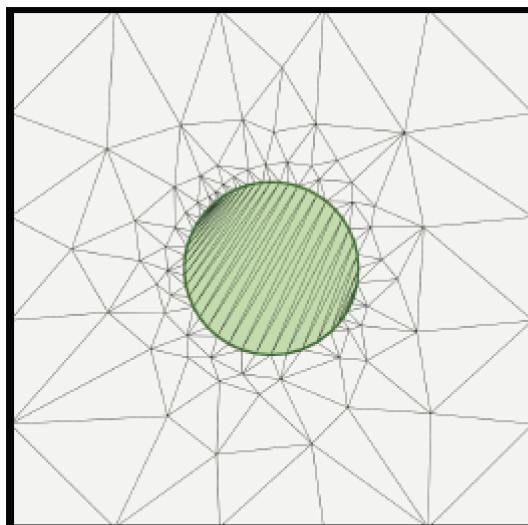
$$\min E_M + \boxed{E_S}$$

Hessian's sparse structure changes

Solving sparse equations

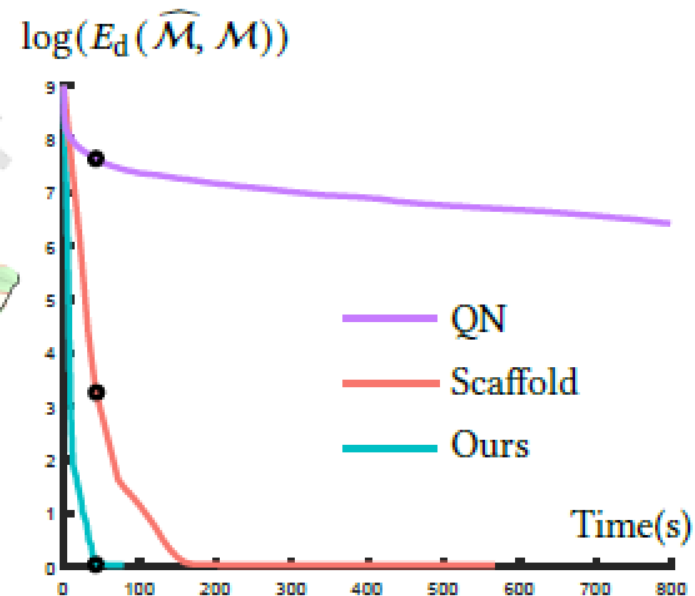
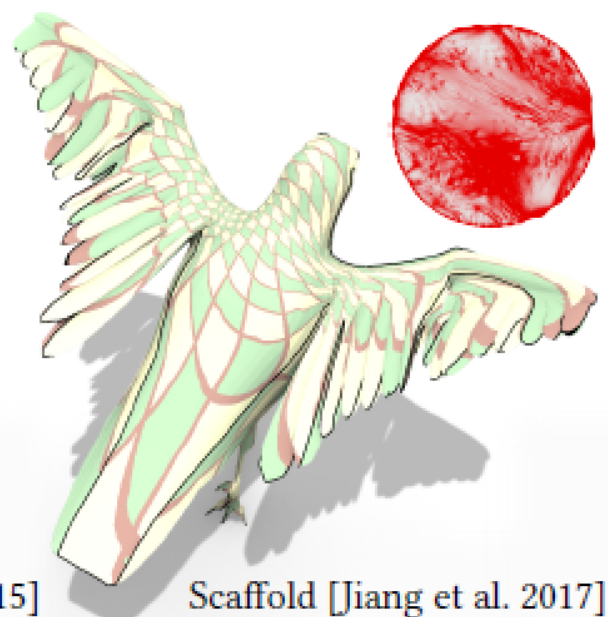
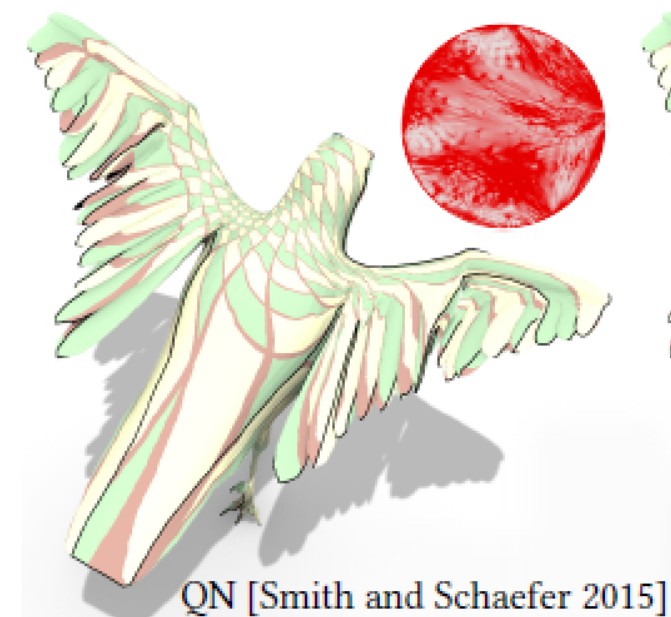
- Symbolic phase (nonzero structure) (0.071)
- Numerical phase (value) (0.012)
- Solve phase (value)

Remesh



Efficient Bijective Parameterizations

Jian-Ping Su , Chunyang Ye, Ligang Liu, Xiao-Ming Fu

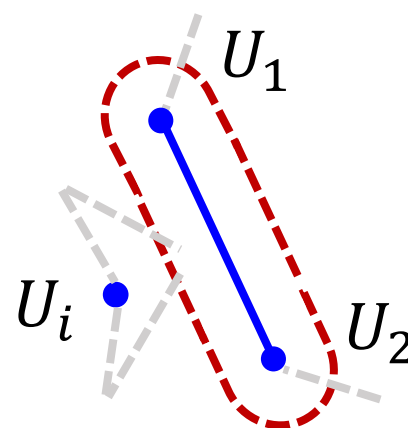
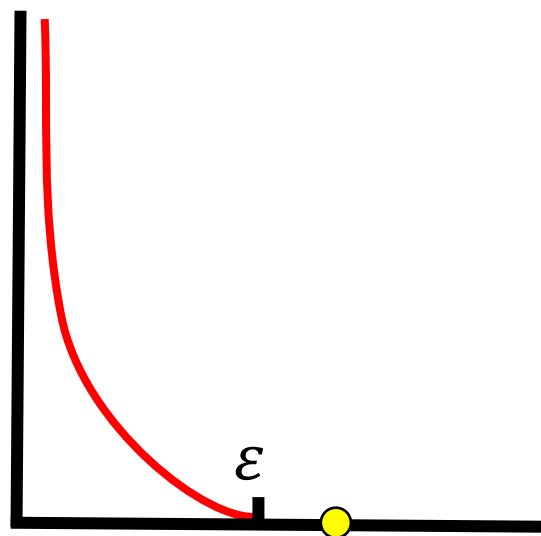


Fixed nonzero structure

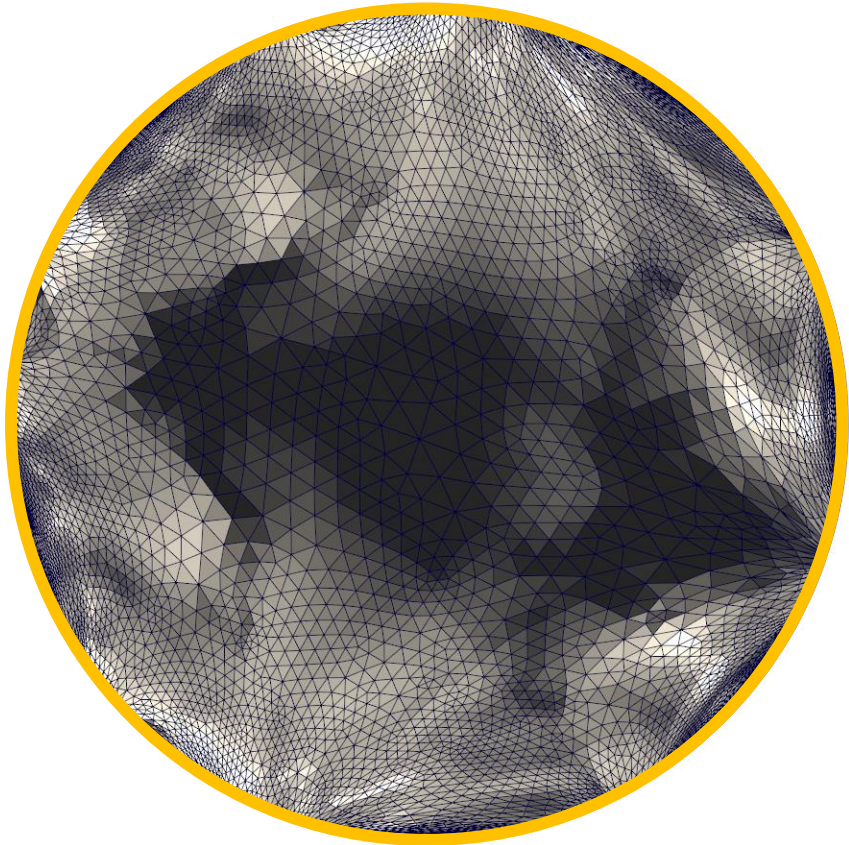
distortion barrier

$$\min E_D + E_B$$

$$E_B = \max\left(0, \frac{\varepsilon}{\text{dist}(U_1, U_2, U_i)} - 1\right)^2$$



High density of matrix

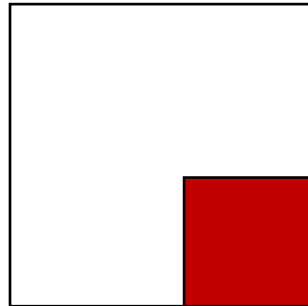


each boundary edge and any boundary vertex

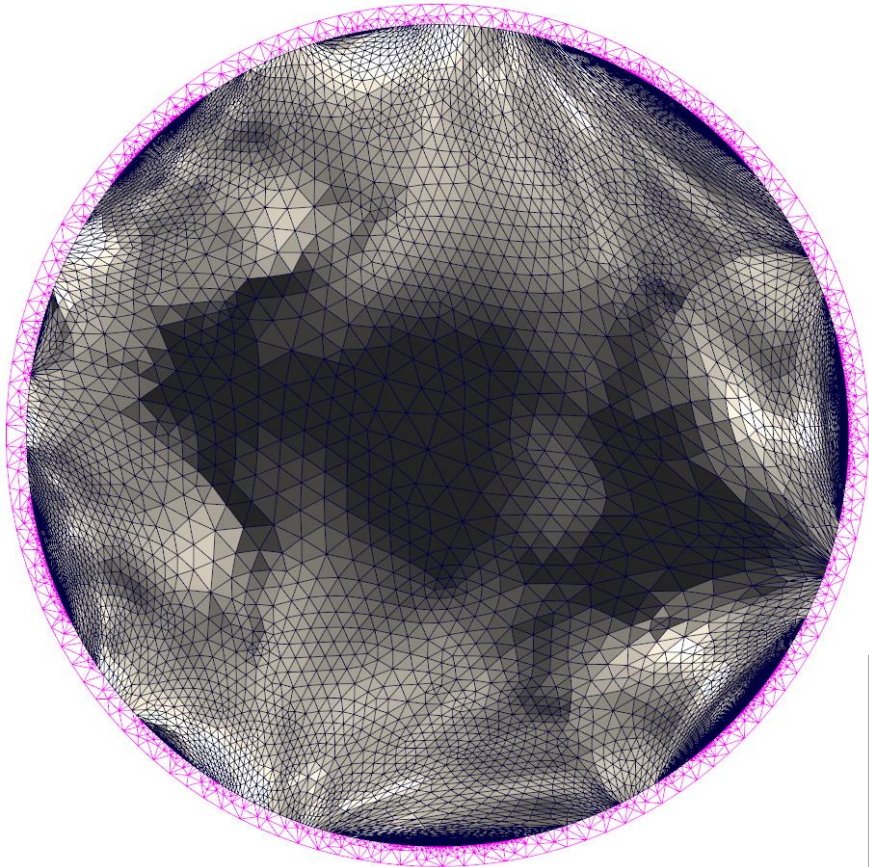
Hessian's sparse structure is fixed

B_N : the number of boundary vertices

Hessian is dense



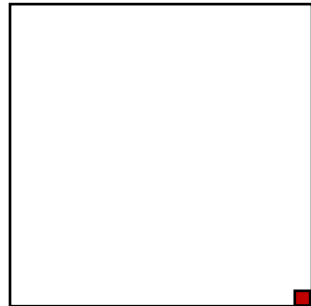
Coarse shell



C_N : the number of cage boundary vertices

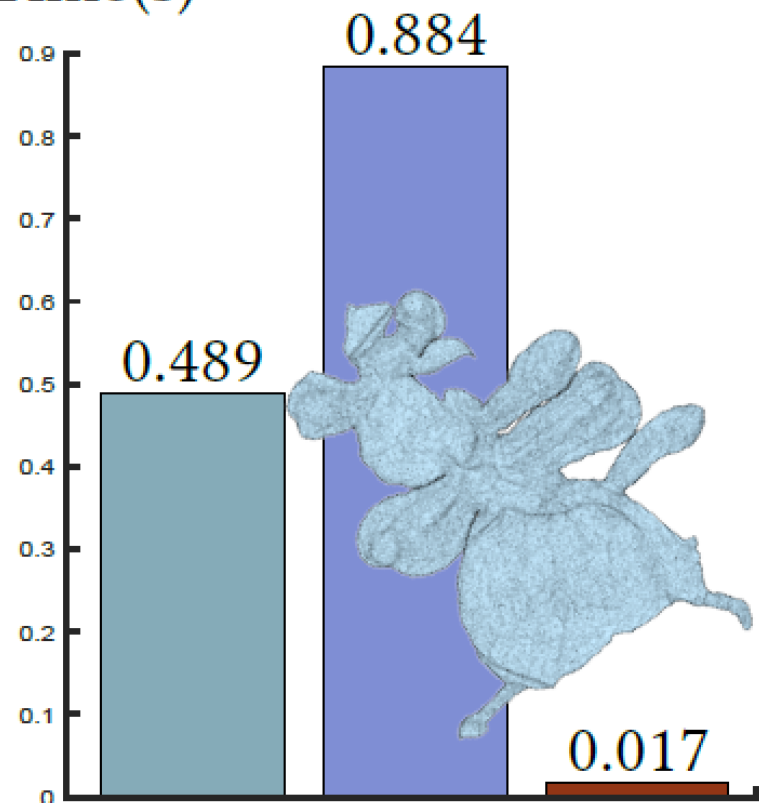
$$C_N \ll B_N$$

Hessian is sparse



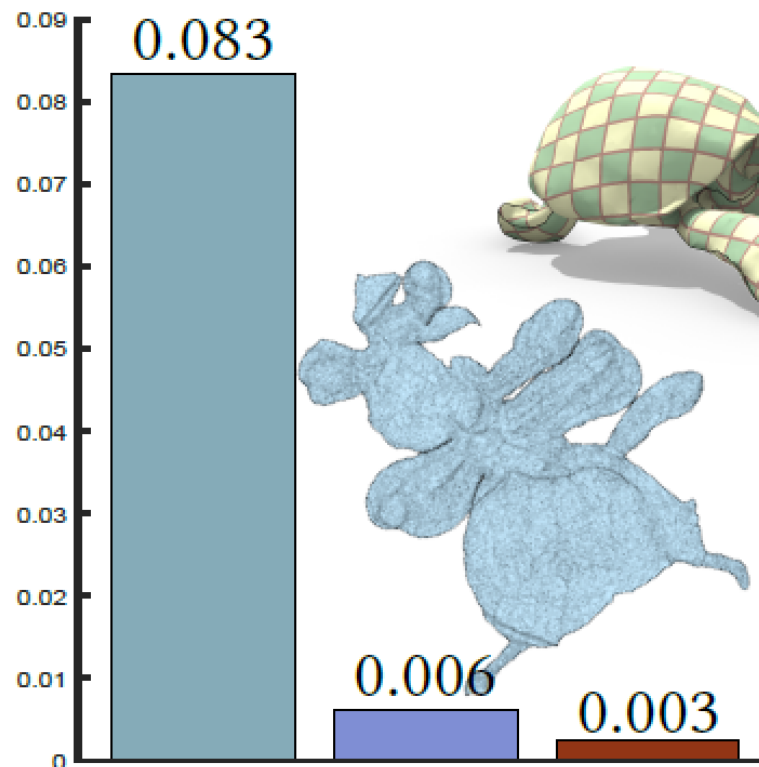
Coarse shell

Time(s)



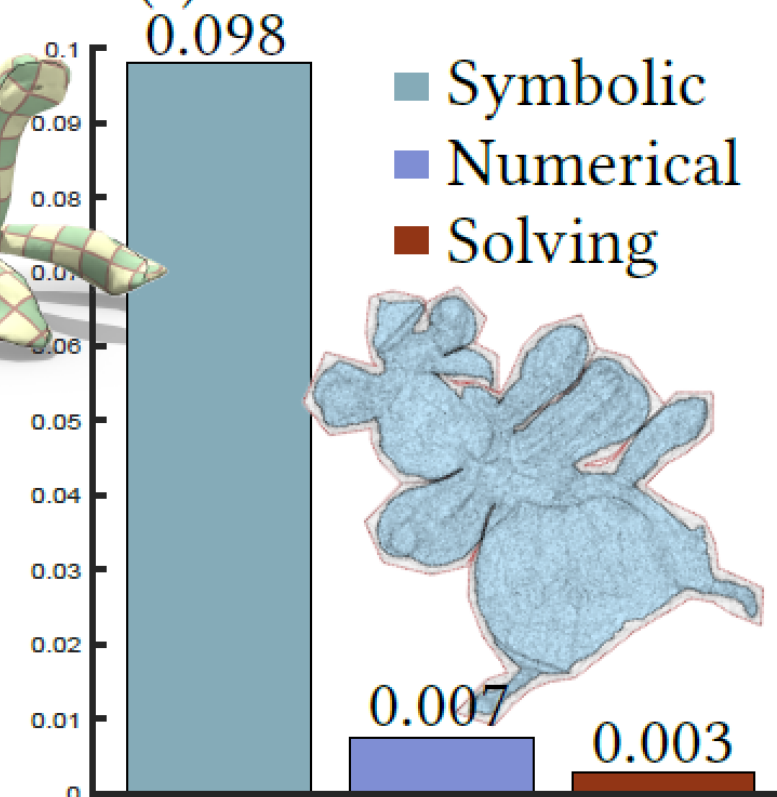
(a) (1.035, 25.937s)

Time(s)



(b) (1.035, 4.547s)

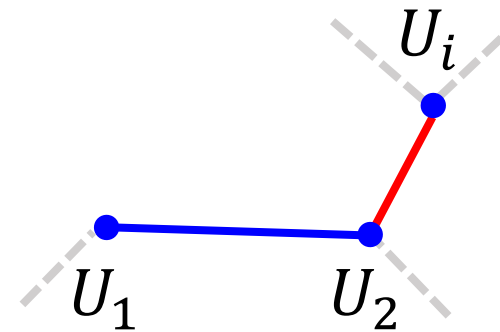
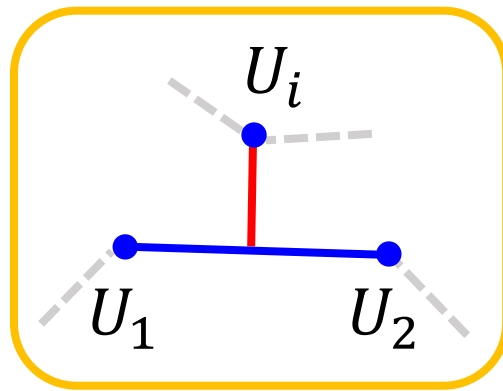
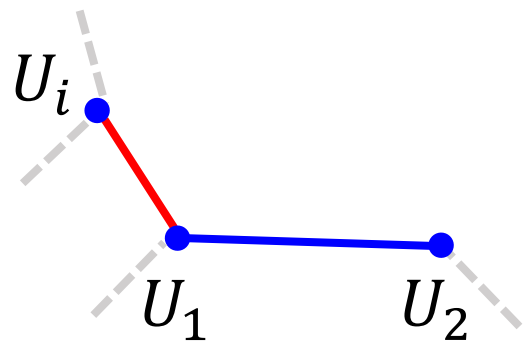
Time(s)



(c) (1.035, 0.509s)

Distance in [Smith et al. 2015]

- Distance is not C^2 .
- Hessian of barrier function is difficult to compute.
- Convex-concave decomposition is not easy.



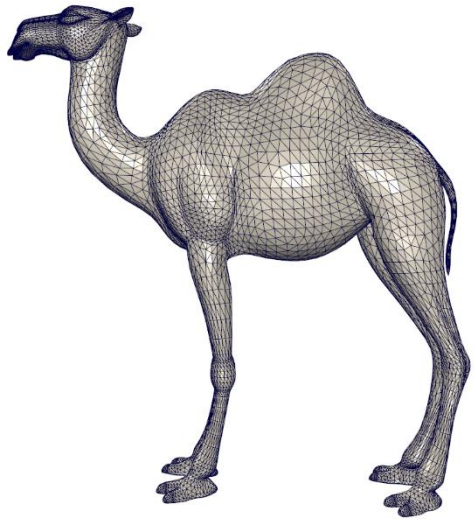
Distance based on triangle inequality

- Infinitely differentiable.
- Analytical second order approximation.

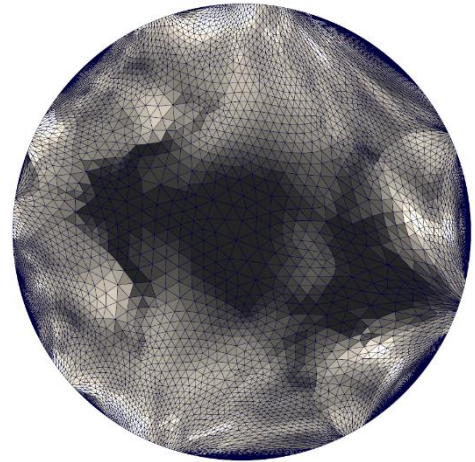
$$dist(U_1, U_2, U_i) = \overset{\text{convex}}{\|U_1 U_i\| + \|U_2 U_i\|} \overset{\text{concave}}{- \|U_1 U_2\|}$$

$$E_B = \max \left(0, \frac{\varepsilon}{dist(U_1, U_2, U_i)} - 1 \right)^2$$

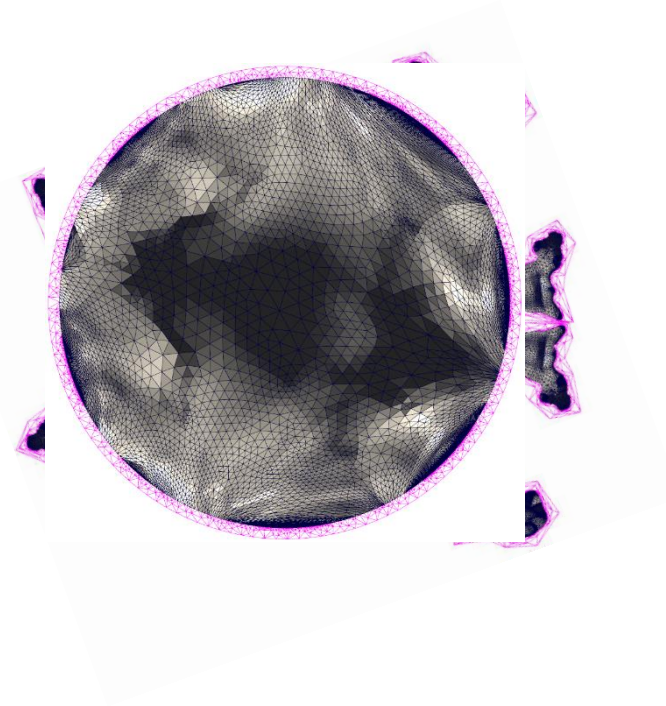
$$\text{convex } f = \left(\frac{\varepsilon}{g} - 1 \right)^2, g = dist(U_1, U_2, U_i)$$



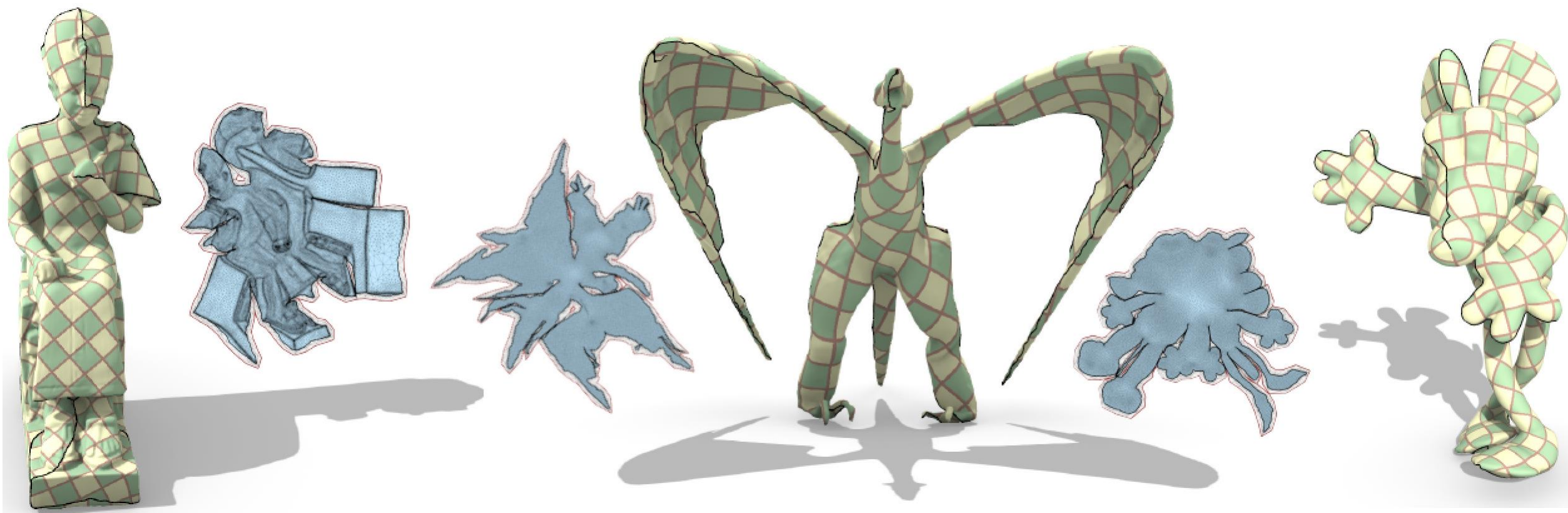
Tutte
initialization



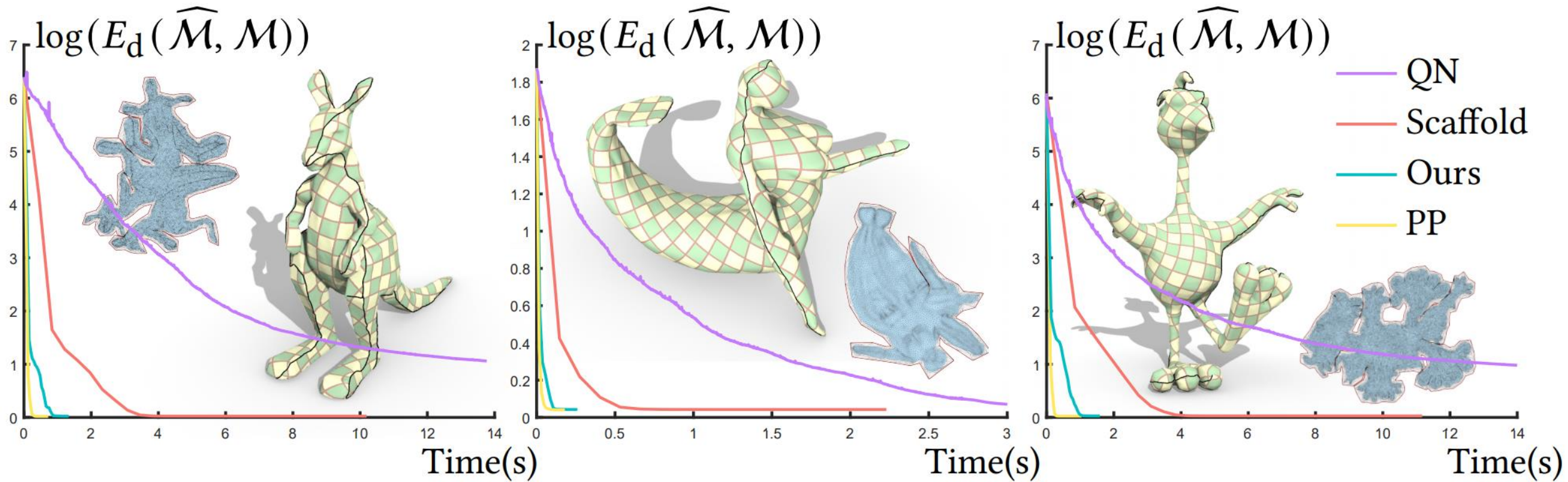
Shell



Result

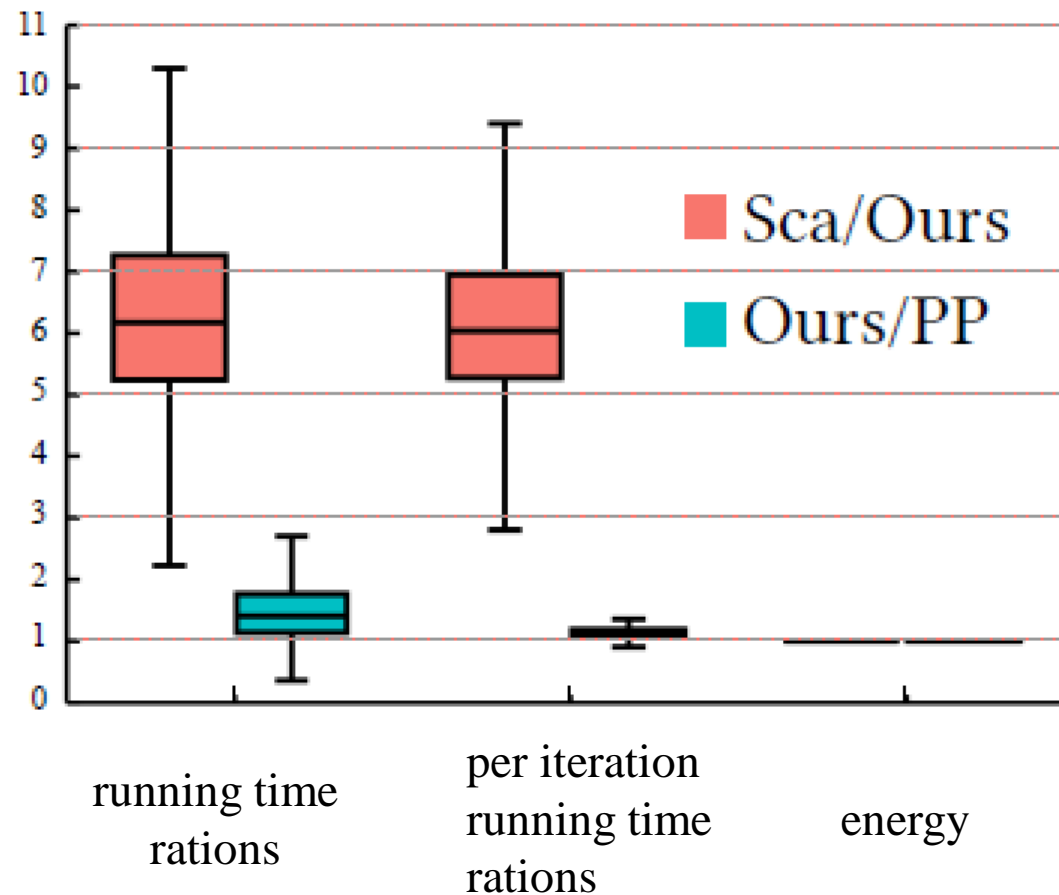


Result



Result

	QN	Sca	PP	Ours
Order	1 st +	2 nd	2 nd	2 nd
Symbolic	-	×	✓	✓
Bijection	✓	✓	×	✓



Geometric Optimization

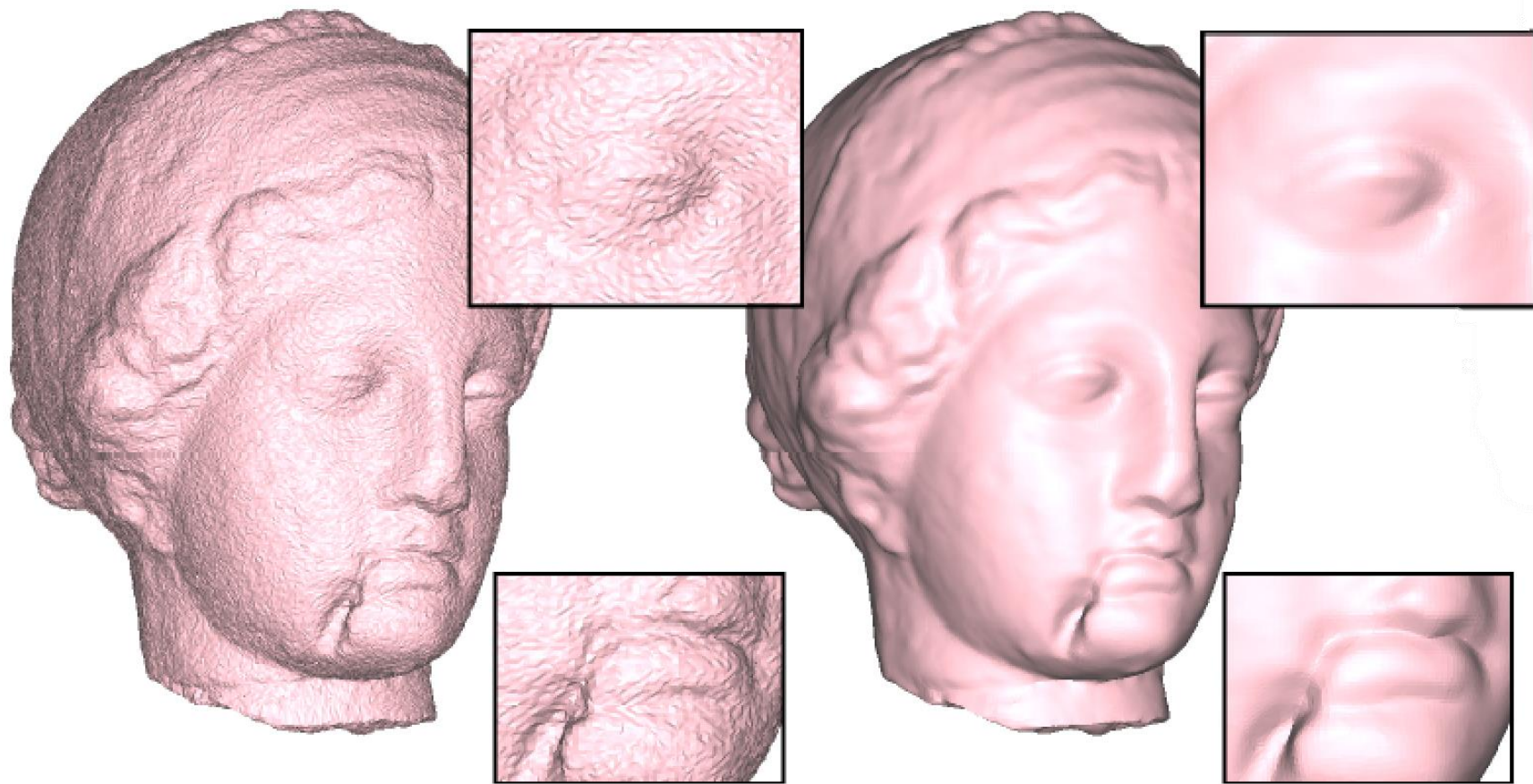
Jian-Ping Su

May 19, 2020

USTC 2020 Spring Digital Geometry Processing

Geometric Optimization

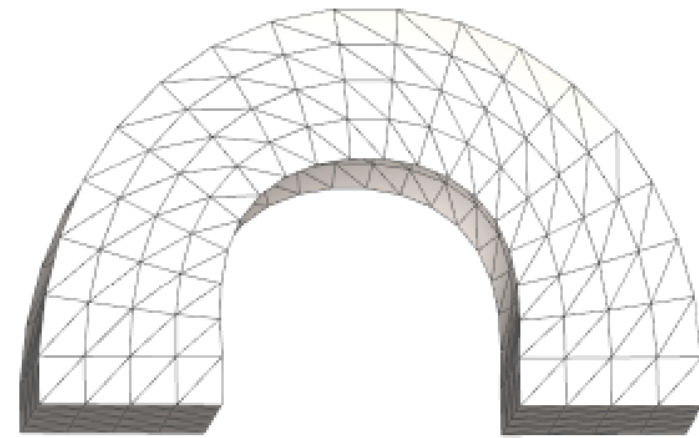
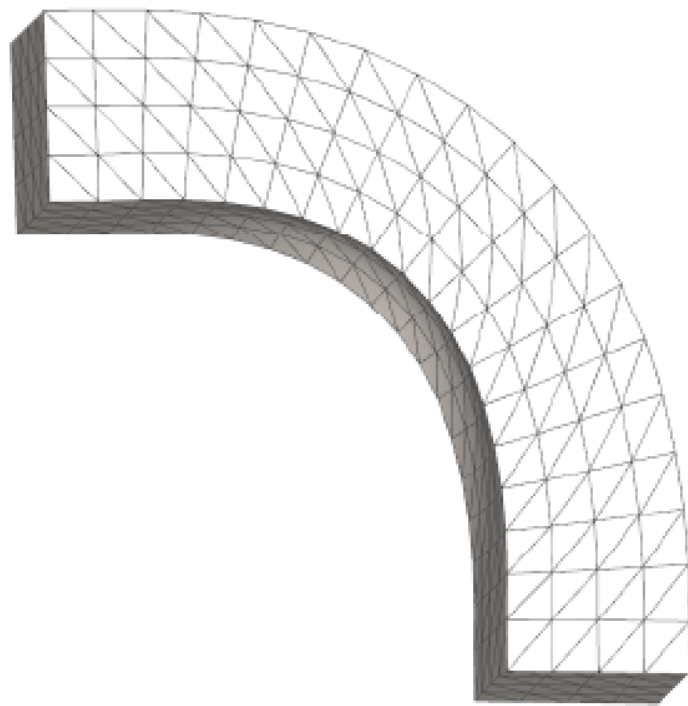
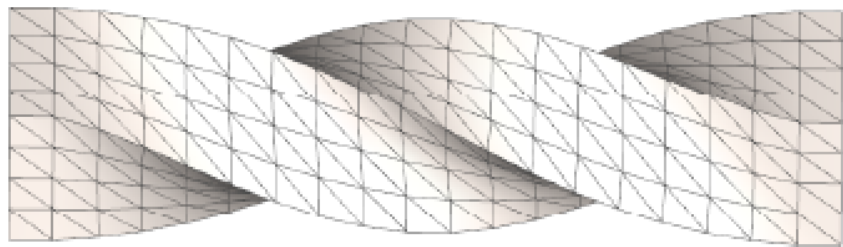
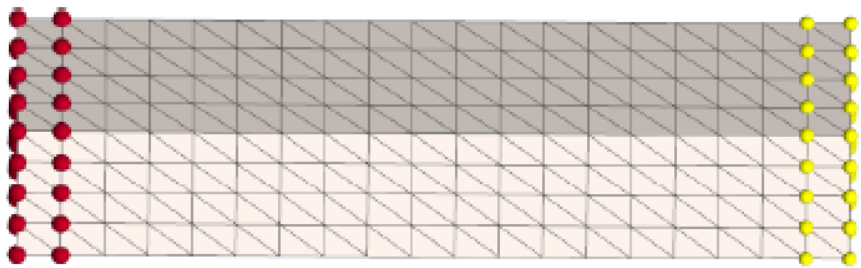
(i) Mesh denoising



Geometric Optimization

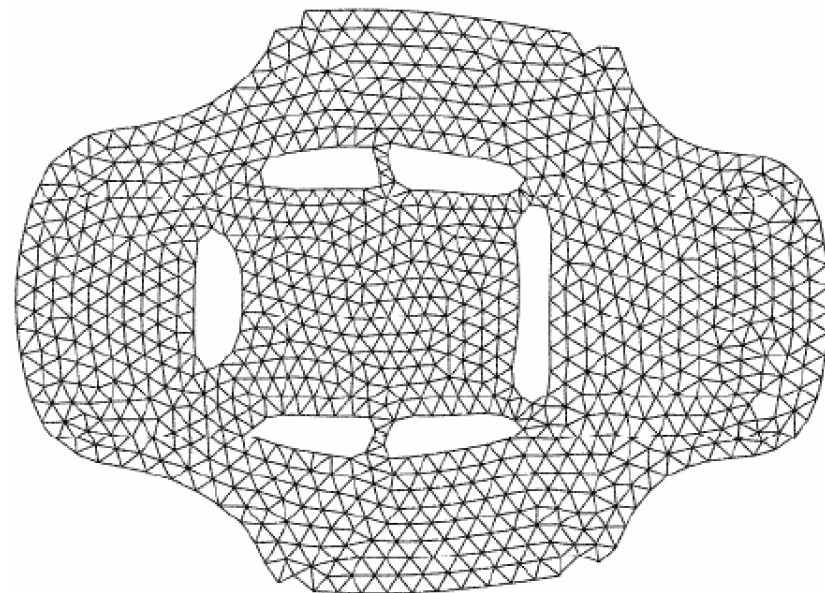
(i) Mesh denoising

(ii) Mesh Deformation



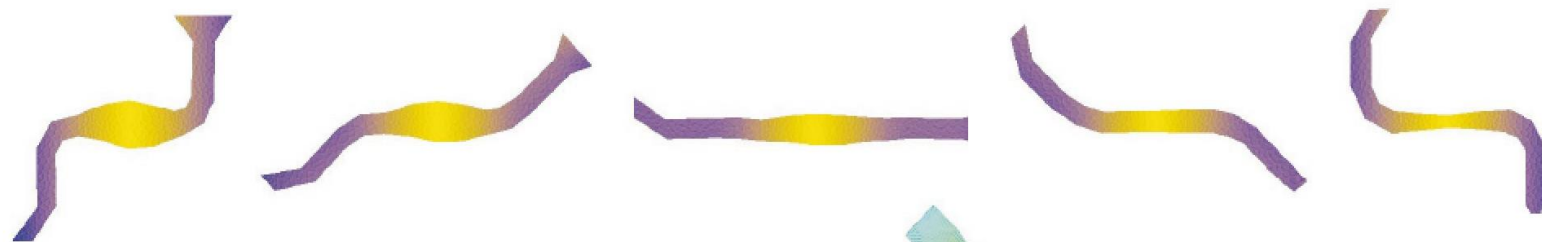
Geometric Optimization

- (i) Mesh denoising
- (ii) Mesh Deformation
- (iii) Mesh parameterization**

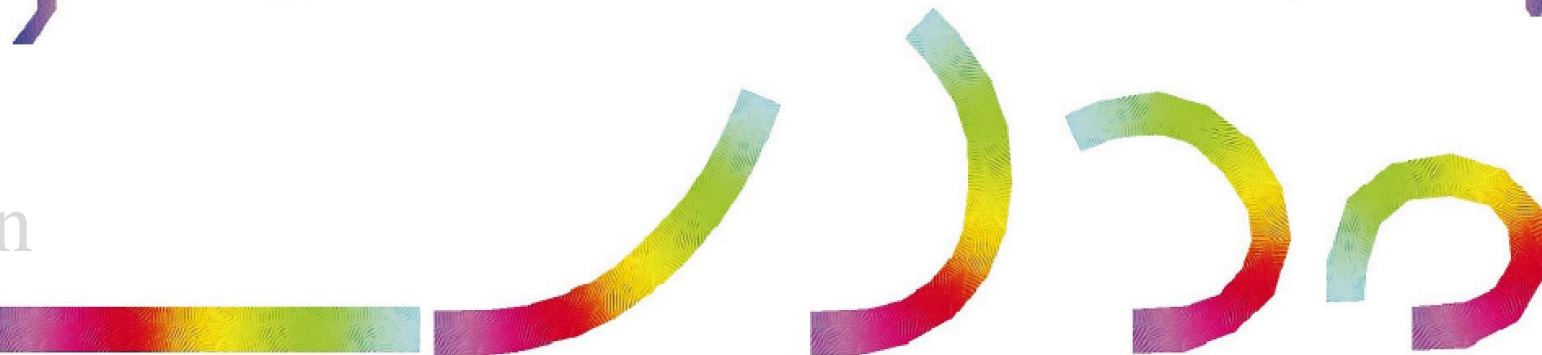


Geometric Optimization

(i) Mesh denoising



(ii) Mesh Deformation



(iii) Mesh parameterization



(iv) Mesh Interpolation

Geometric Optimization

(i) Mesh denoising

(ii) Mesh Deformation

(iii) Mesh parameterization

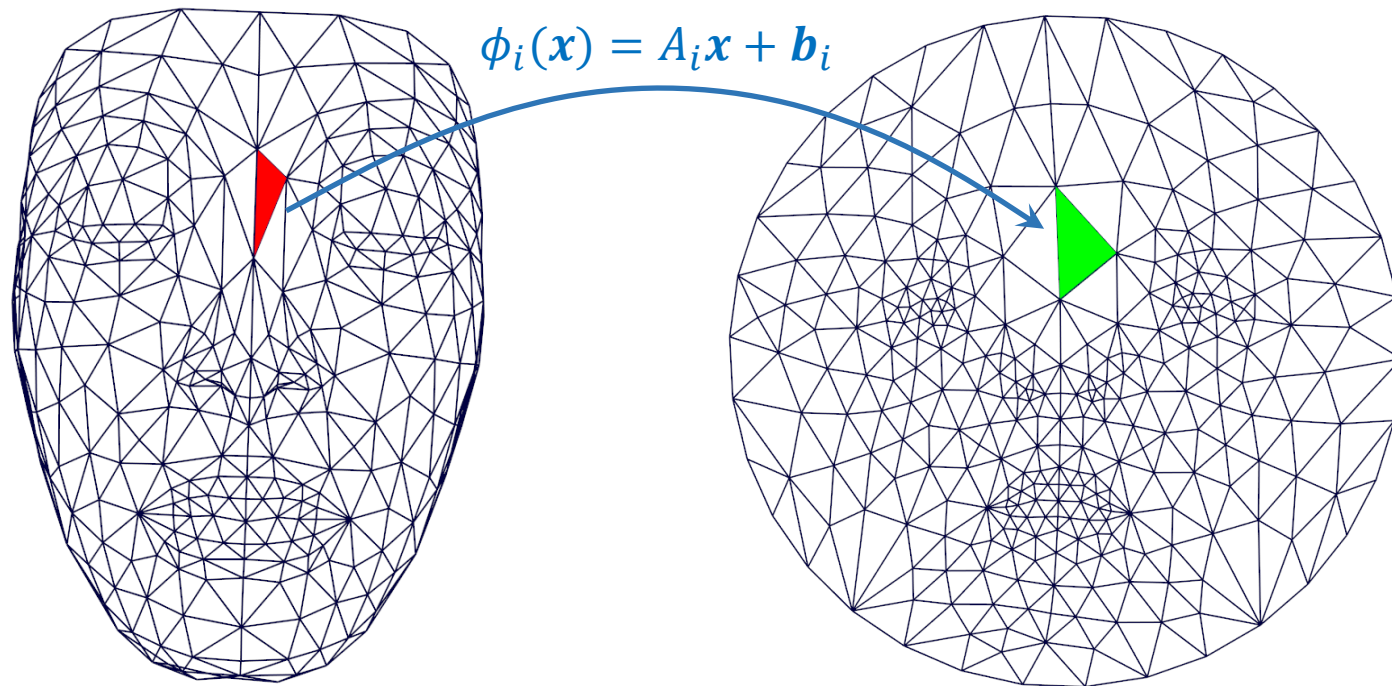
(iv) Mesh Interpolation

(v) Mesh Simplification

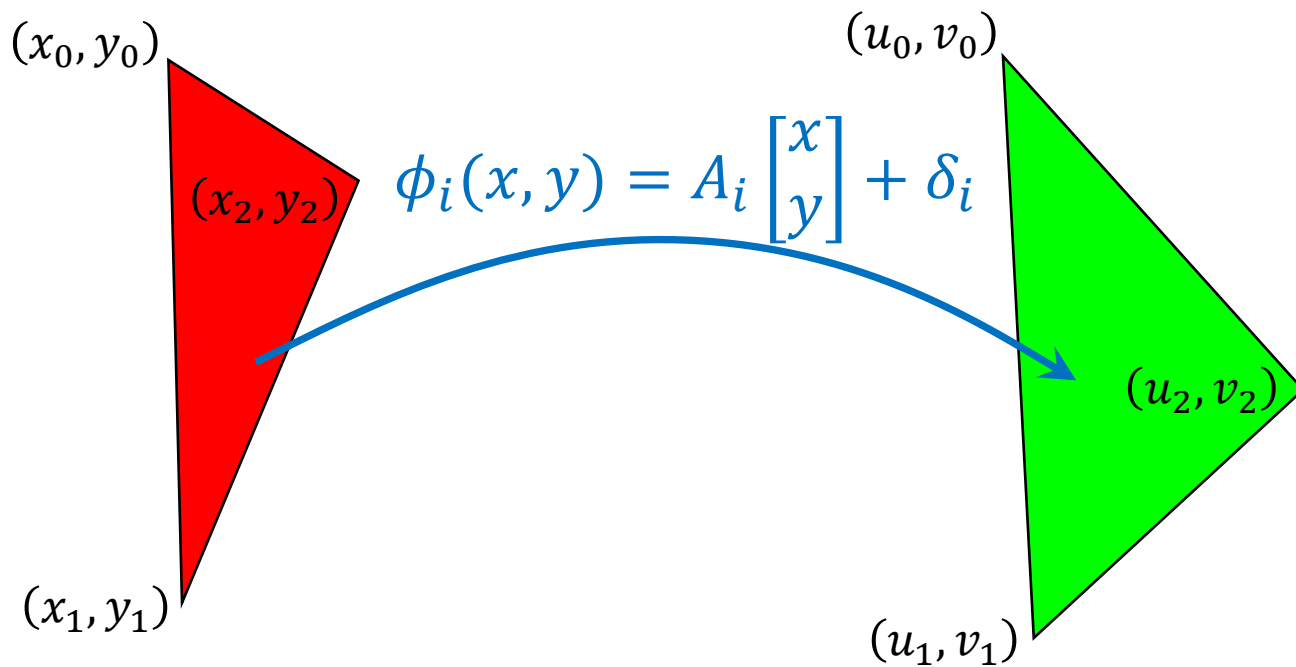


Preliminary

Jacobian Matrix



Jacobian Matrix

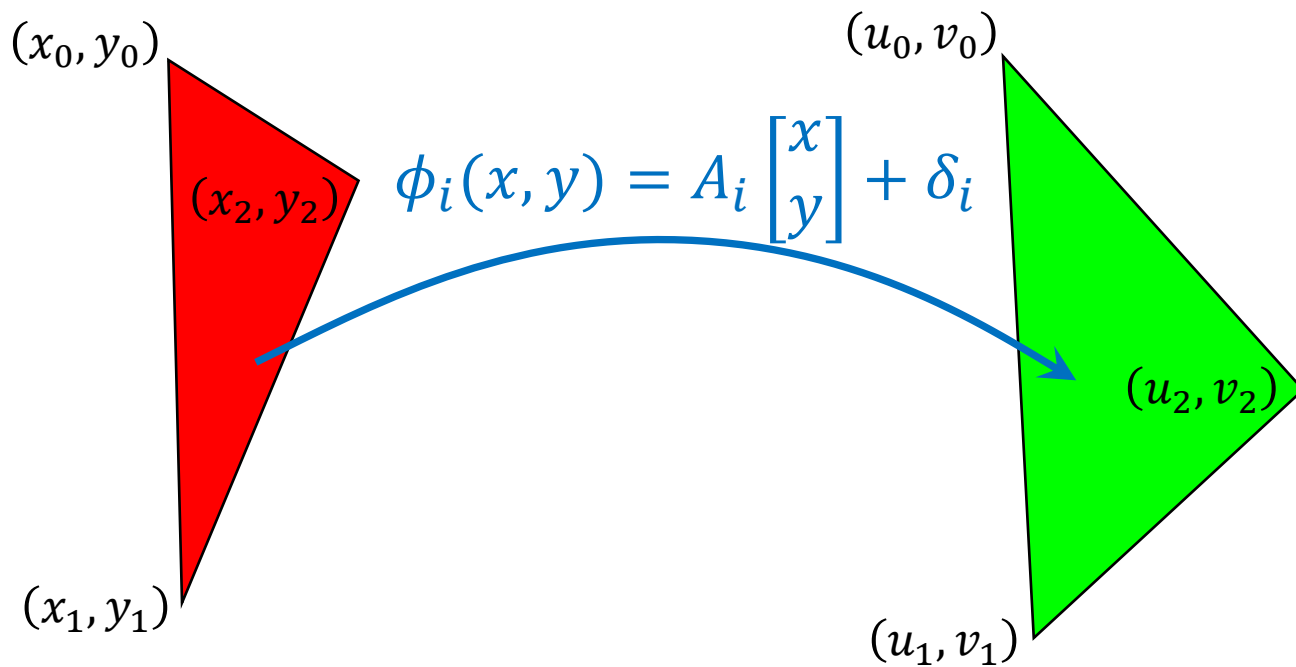


$$\begin{bmatrix} u \\ v \end{bmatrix} = A_i \begin{bmatrix} x \\ y \end{bmatrix} + \delta_i$$

$$J_i = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} = A_i$$

$$A_i = \begin{bmatrix} u_1 - u_0 & u_2 - u_0 \\ v_1 - v_0 & v_2 - v_0 \end{bmatrix} \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix}^{-1}$$

Jacobian Matrix



$$\begin{bmatrix} u \\ v \end{bmatrix} = A_i \begin{bmatrix} x \\ y \end{bmatrix} + \delta_i$$

$$J_i = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} = A_i$$

$$J_i = \begin{bmatrix} u_1 - u_0 & u_2 - u_0 \\ v_1 - v_0 & v_2 - v_0 \end{bmatrix} \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix}^{-1}$$

Jacobian Matrix

$$J_i = \begin{bmatrix} u_1 - u_0 & u_2 - u_0 \\ v_1 - v_0 & v_2 - v_0 \end{bmatrix} \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix}^{-1}$$

$$J_i^T = \begin{bmatrix} x_1 - x_0 & y_1 - y_0 \\ x_2 - x_0 & y_2 - y_0 \end{bmatrix}^{-1} \begin{bmatrix} u_1 - u_0 & v_1 - v_0 \\ u_2 - u_0 & v_2 - v_0 \end{bmatrix}$$

$$J_i^T = \frac{1}{2|t_j|} \begin{bmatrix} y_2 - y_0 & y_0 - y_1 \\ x_0 - x_2 & x_1 - x_0 \end{bmatrix} \begin{bmatrix} u_1 - u_0 & v_1 - v_0 \\ u_2 - u_0 & v_2 - v_0 \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \frac{1}{2|t_i|} \begin{bmatrix} y_1 - y_2 & y_2 - y_0 & y_0 - y_1 \\ x_2 - x_1 & x_0 - x_2 & x_1 - x_0 \\ y_1 - y_2 & y_2 - y_0 & y_0 - y_1 \\ x_2 - x_1 & x_0 - x_2 & x_1 - x_0 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ v_0 \\ v_1 \\ v_2 \end{bmatrix}$$

Singular Value

The arithmetic square root of non-negative eigenvalues of $J^T J$

$$J^T J = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} a^2 + c^2 & ab + cd \\ ab + cd & b^2 + d^2 \end{bmatrix}$$

$$a' = \frac{a+d}{2}, c' = \frac{a-d}{2}, b' = \frac{c-b}{2}, d' = \frac{c+b}{2}$$

$$\Sigma = \sqrt{a'^2 + b'^2} + \sqrt{c'^2 + d'^2}, \quad \sigma = \left| \sqrt{a'^2 + b'^2} - \sqrt{c'^2 + d'^2} \right|$$

$$\|J\|_F^2 = a^2 + b^2 + c^2 + d^2 = \text{tr}(J^T J) = \Sigma^2 + \sigma^2$$

Distortion Measure

- Dirichlet

$$f_{\text{DRCH}}(\mathbf{x}) = \sum_i \|J_i\|_F^2 |t_i| = \sum_i (\Sigma_i^2 + \sigma_i^2) |t_i|$$

- ARAP

$$f_{\text{ARAP}}(\mathbf{x}) = \sum_i \|J_i - R(J_i(\mathbf{x}))\|_F^2 |t_i| = \sum_i ((\Sigma_i - 1)^2 + (\sigma_i - 1)^2) |t_i|$$

- Symmetric Dirichlet

$$f_{\text{ISO}}(\mathbf{x}) = \sum_i (\|J_i\|_F^2 + \|J_i\|_F^{-2}) |t_i| = \sum_i (\Sigma_i^2 + \Sigma_i^{-2} + \sigma_i^2 + \sigma_i^{-2}) |t_i|$$

- Conformal

$$f_{\text{CONF}}(\mathbf{x}) = \sum_i \left(\frac{\Sigma_i}{\sigma_i} \right)^2 |t_i|$$

Laplace Matrix

Laplace Matrix is Hessian of $f_{\text{DRCH}}(\mathbf{x})$

$$f_{\text{DRCH}}(\mathbf{x}) = \sum_i \|J_i\|_F^2 |t_i| = \sum_i (\Sigma_i^2 + \sigma_i^2) |t_i|$$

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \frac{1}{2|t_i|} \begin{bmatrix} y_1 - y_2 & y_2 - y_0 & y_0 - y_1 \\ x_2 - x_1 & x_0 - x_2 & x_1 - x_0 \\ y_1 - y_2 & y_2 - y_0 & y_0 - y_1 \\ x_2 - x_1 & x_0 - x_2 & x_1 - x_0 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ v_0 \\ v_1 \\ v_2 \end{bmatrix}$$

$$J = T\mathbf{x} \quad f_{\text{DRCH}}(\mathbf{x}) = \mathbf{x}^T T^T T \mathbf{x} \quad L = T^T T$$

Problem

$$\min_{\mathbf{x}} f(\mathbf{x}) = \sum_i |t_i| \mathcal{D}(J_i(\mathbf{x}))$$

1. Initial point: $\mathbf{x}_0, n = 0$

2. Descent direction: $\min_p f(\mathbf{x}_n + p)$

3. Step size: $\min_{\alpha} f(\mathbf{x}_n + \alpha p)$

4. Update: $\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha p, n = n + 1$

Method

$$\min_p f(\mathbf{x}_n + p)$$

$$f(\mathbf{x}_n + p) \approx f(\mathbf{x}_n) + \nabla f(\mathbf{x}_n)^T p + \frac{1}{2} p^T H p$$

$$H p = -\nabla f(\mathbf{x}_n)$$

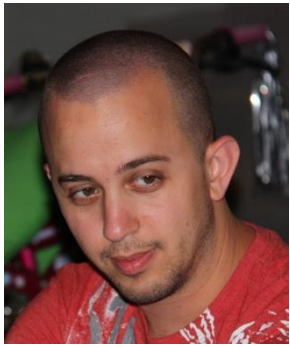
- **First-order methods** build descent steps by preconditioning the gradient with a **fixed proxy matrix**, which often suffer from slower convergence as lacking of higher-order information.
- **Newton-type methods** uses the energy Hessian, $\nabla^2 f(\mathbf{x})$, to form a proxy matrix, which can achieve the most rapid convergence but require the costly assembly, factorization and backsolve of new linear systems per iterate.

First-order method

Paper List

- Accelerated Quadratic Proxy for Geometric Optimization
- Scalable Locally Injective Mappings
- Blended Cured Quasi-Newton for Distortion Optimization

Accelerated Quadratic Proxy for Geometric Optimization



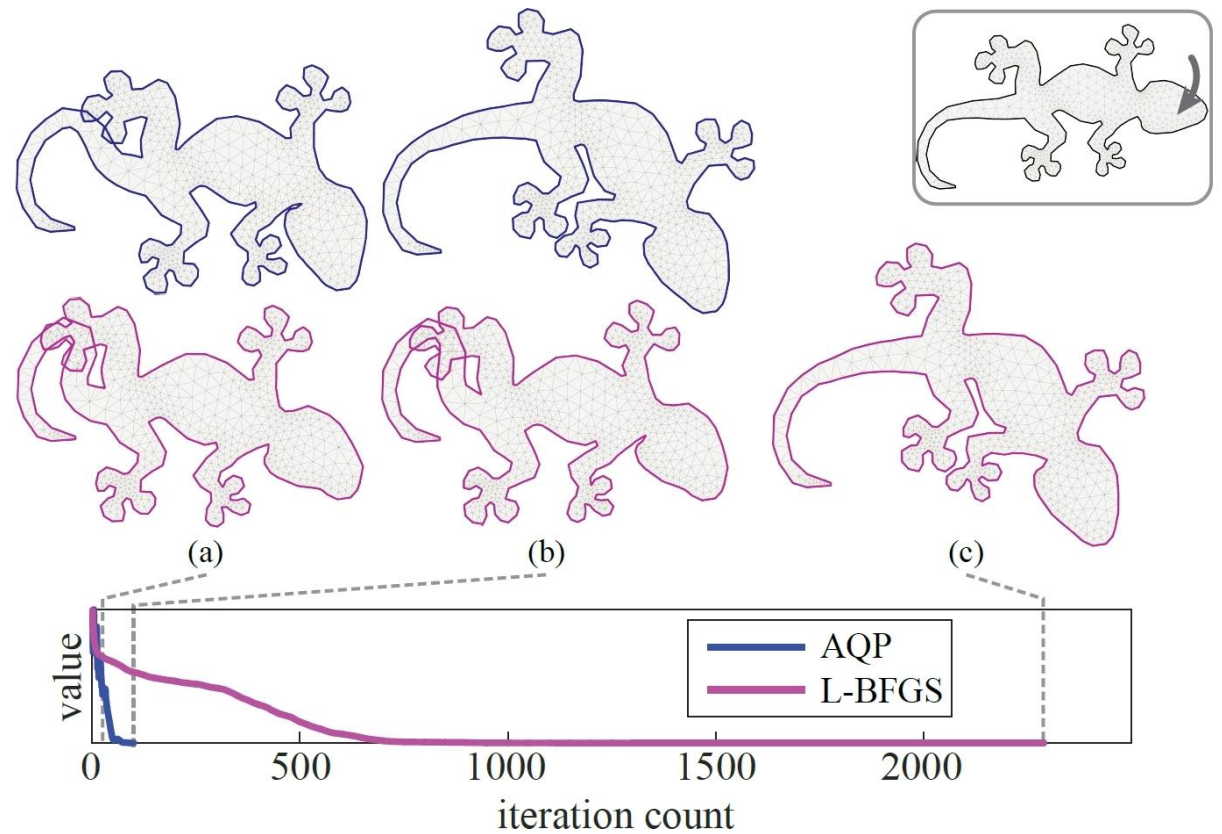
Shahar Kovalsky



Meirav Galun



Yaron Lipman



Motivation

ill-conditioning dominated by a Laplacian-like term in energy.

locally approximating the energy with a function whose Hessian is **Laplacian**.

$$f(\mathbf{x}) = h(\mathbf{x}) + g(\mathbf{x})$$

$$h(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T H \mathbf{x}$$

$$f(x_{n-1} + p) = h(x_{n-1} + p) + g(x_{n-1} + p)$$

$$f(x_{n-1} + p) \approx h(x_{n-1} + p) + g(x_{n-1}) + \nabla g(x_{n-1})^T p$$

$$H(x_{n-1} + p) + \nabla g(x_{n-1}) = 0$$

$$Hp = -Hx_{n-1} - \nabla g(x_{n-1}) = -\nabla f(x_{n-1})$$

Motivation

correctly balancing the information from its **two last iterations**.

$$y_n = (1 + \theta)x_{n-1} - \theta x_{n-2}$$

$$f(x) = h(x) + g(x)$$

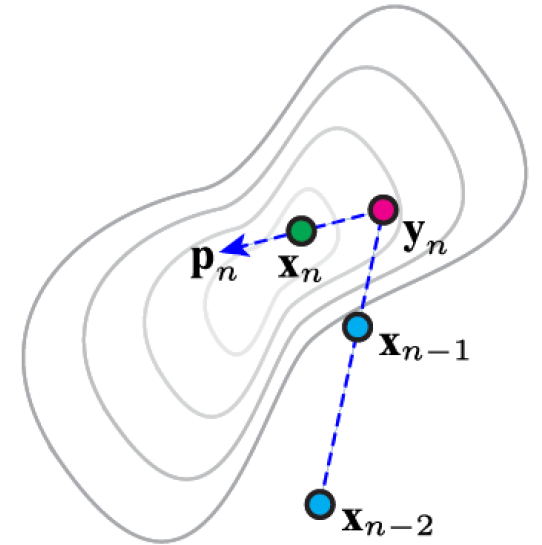
$$h(x) = \frac{1}{2}x^T Hx$$

$$f(y_n + p) = h(y_n + p) + g(y_n + p)$$

$$f(y_n + p) \approx h(y_n + p) + g(y_n) + \nabla g(y_n)^T p$$

$$H(y_n + p) + \nabla g(y_n) = 0$$

$$Hp = -Hy_n - \nabla g(y_n) = -\nabla f(y_n)$$



Method

$$f(\mathbf{x}) = h(\mathbf{x}) + g(\mathbf{x})$$

$$h(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T H \mathbf{x}$$

$$f(\mathbf{y}_n + \mathbf{p}) = h(\mathbf{y}_n + \mathbf{p}) + g(\mathbf{y}_n + \mathbf{p})$$

$$f(\mathbf{y}_n + \mathbf{p}) \approx h(\mathbf{y}_n + \mathbf{p}) + g(\mathbf{y}_n) + \nabla g(\mathbf{y}_n)^T \mathbf{p}$$

$$H(\mathbf{y}_n + \mathbf{p}) + \nabla g(\mathbf{y}_n) = 0$$

$$H\mathbf{p} = -H\mathbf{y}_n - \nabla g(\mathbf{y}_n) = -\nabla f(\mathbf{y}_n)$$

Algorithm 1: Accelerated Quadratic Proxy (AQP)

Data: feasible initialization \mathbf{x} ; parameter η

$\mathbf{x}_{-1} = \mathbf{x}_0 = \mathbf{x}$;

$\theta = \frac{1 - \sqrt{1/\eta}}{1 + \sqrt{1/\eta}}$;

while *not converged* **do**

/* Acceleration */

$\mathbf{y}_n = (1 + \theta)\mathbf{x}_{n-1} - \theta\mathbf{x}_{n-2}$;

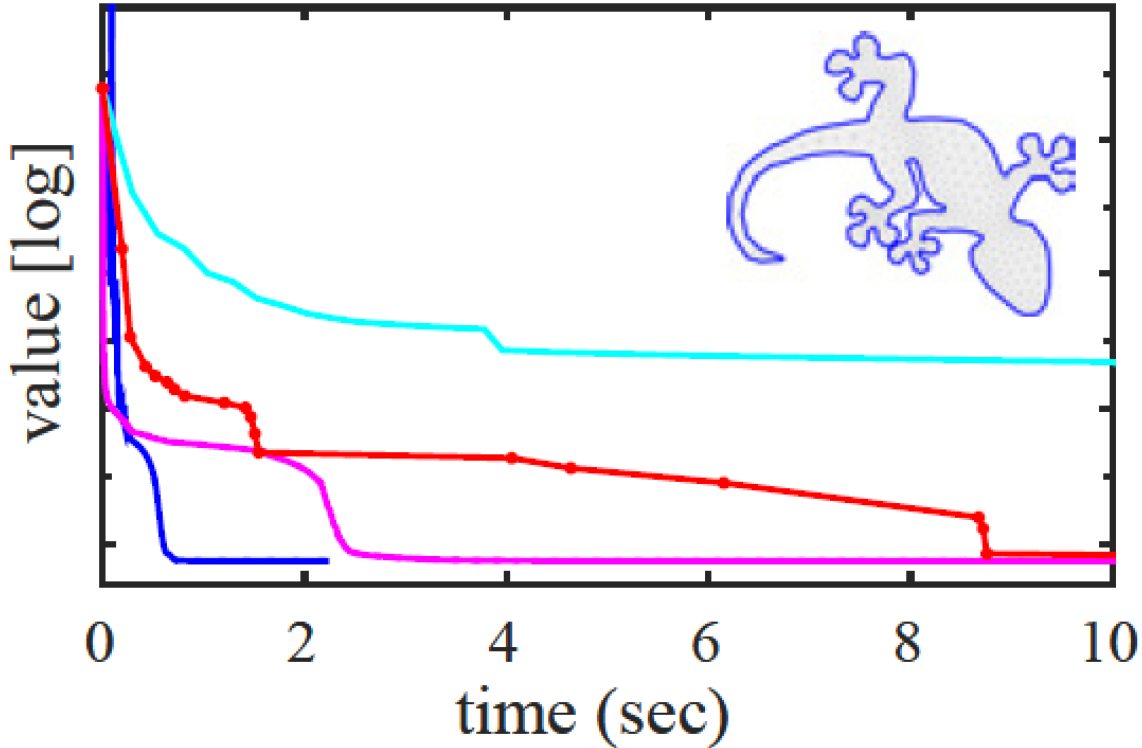
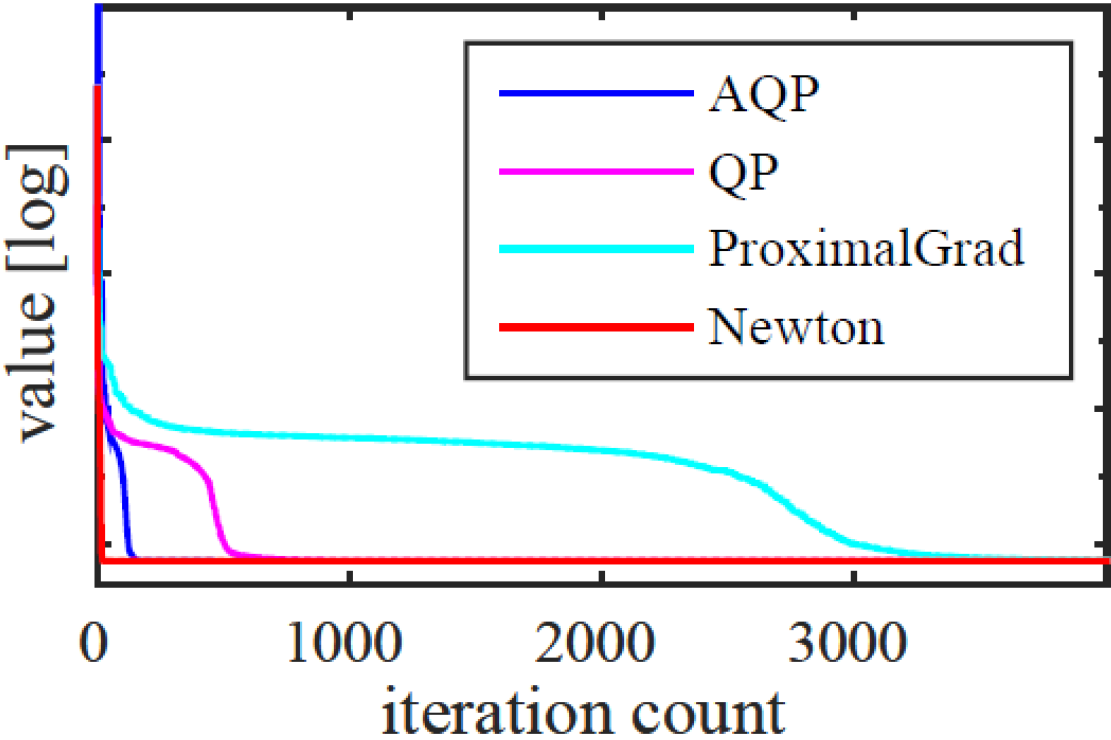
/* Quadratic proxy minimization */

$\mathbf{p}_n = \arg \min_{\mathbf{p}} h(\mathbf{y}_n + \mathbf{p}) + g(\mathbf{y}_n) + \nabla g(\mathbf{y}_n)^T \mathbf{p}$
s.t. $A\mathbf{p} = 0$

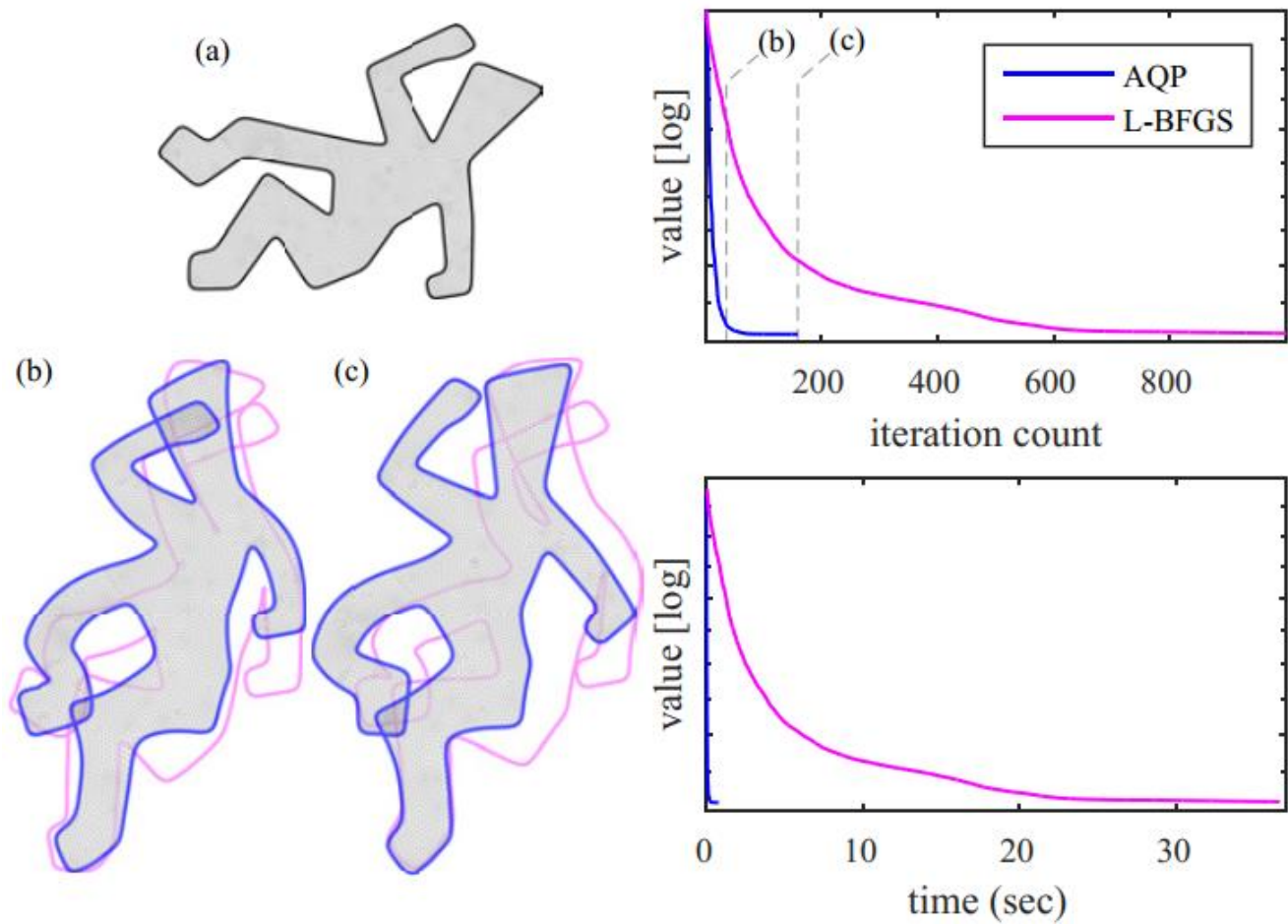
/* Line search */

$\mathbf{x}_n = \text{linesearch}_{0 < t \leq 1} f(\mathbf{y}_n + t\mathbf{p}_n)$

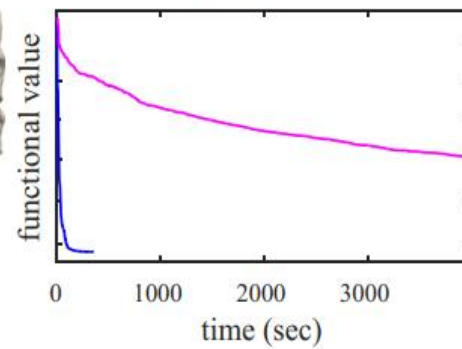
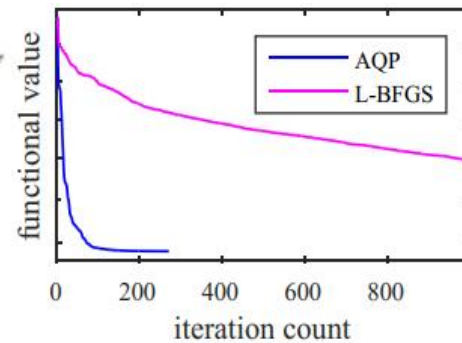
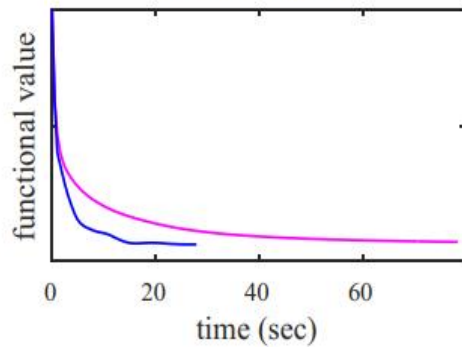
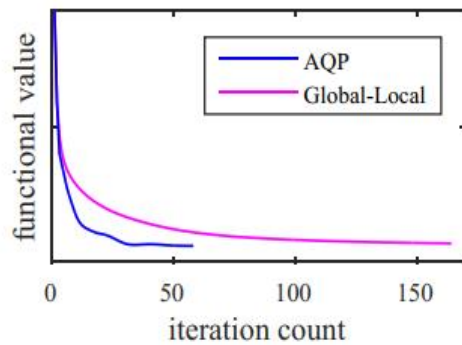
Results



Results



Results



Limitation

$$f(\mathbf{x}) = h(\mathbf{x}) + g(\mathbf{x})$$

$$h(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T H \mathbf{x}$$

$$f_{\text{ARAP}}(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T H \mathbf{x} - \sum_i \|J_i\|_* |t_i| + c_0$$

efficiency of Laplacian
approximation for an
arbitrary energy

$$f_{\text{ISO}}(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T H \mathbf{x} + \sum_i \|J_i\|_F^{-2} |t_i|$$

$$f_{\text{CONF}}(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T H \mathbf{x} + \sum_i \|J_i\|_F^2 \left(\frac{1}{\sigma_i^2} - 1 \right) |t_i|$$

Video

Scalable Locally Injective Mappings



Michael Rabinovich



Roi Poranne

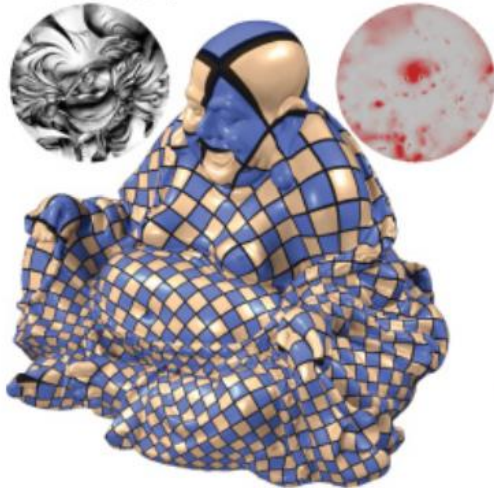


Daniele Panozzo

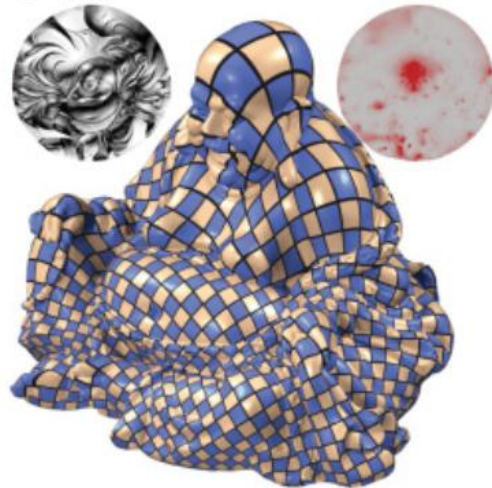


Olga Sorkine-Hornung

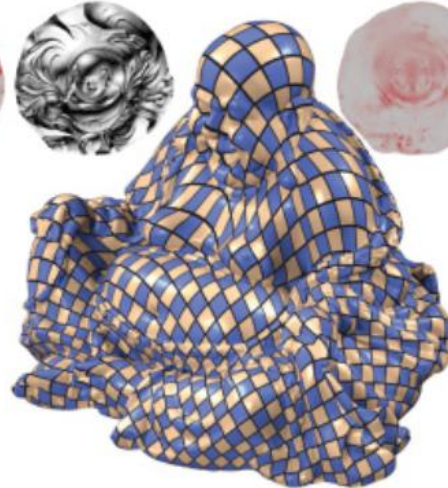
Initialization



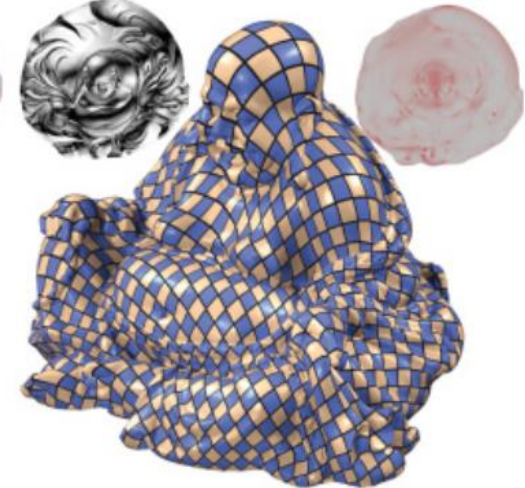
Iteration 1



Iteration 2

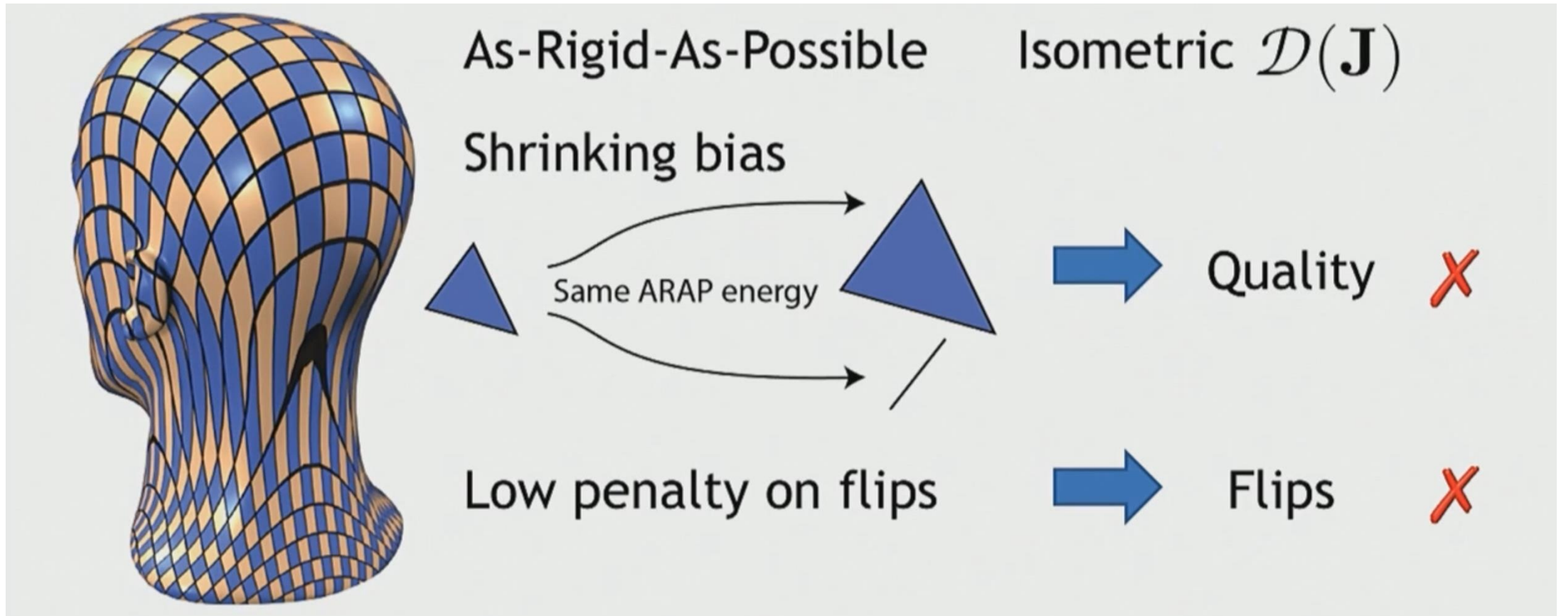


Iteration 20



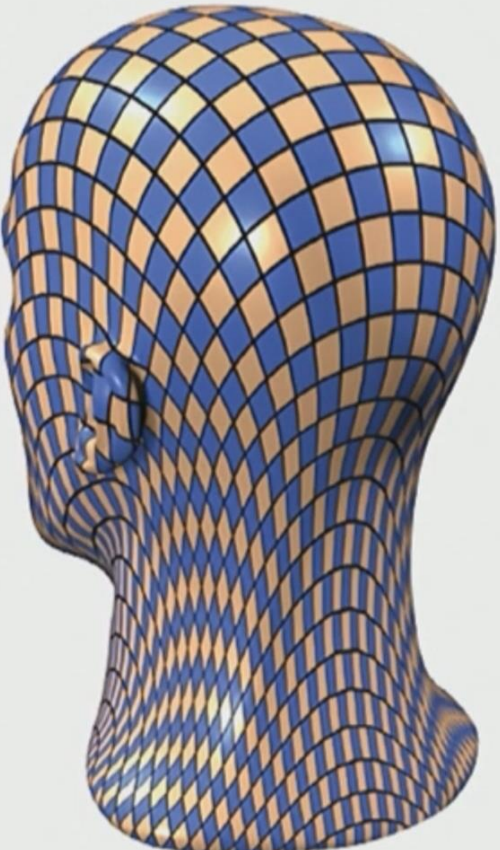
Motivation

$$f_{\text{ARAP}}(\mathbf{x}) = \sum_i ((\Sigma_i - 1)^2 + (\sigma_i - 1)^2) |t_i|$$



Motivation

$$f_{\text{ISO}}(\mathbf{x}) = \sum_i (\Sigma_i^2 + \Sigma_i^{-2} + \sigma_i^2 + \sigma_i^{-2}) |t_i|$$



Isometric $\mathcal{D}(\mathbf{J})$

Scaling = Shrinking	➔	Quality ✓
Infinite on flips	➔	No flips ✓

Motivation

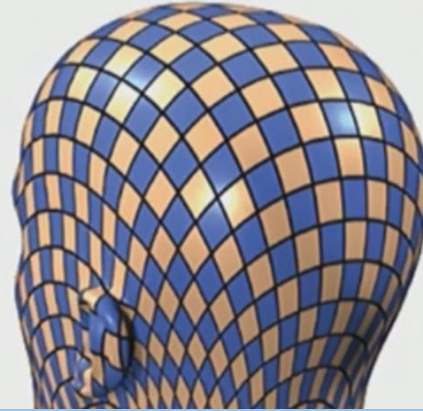
$$f_{\text{ARAP}}(\mathbf{x}) = \sum_i ((\Sigma_i - 1)^2 + (\sigma_i - 1)^2) |t_i|$$

$$f_{\text{ISO}}(\mathbf{x}) = \sum_i (\Sigma_i^2 + \Sigma_i^{-2} + \sigma_i^2 + \sigma_i^{-2}) |t_i|$$

Local/Global
[Liu et al. 2008]

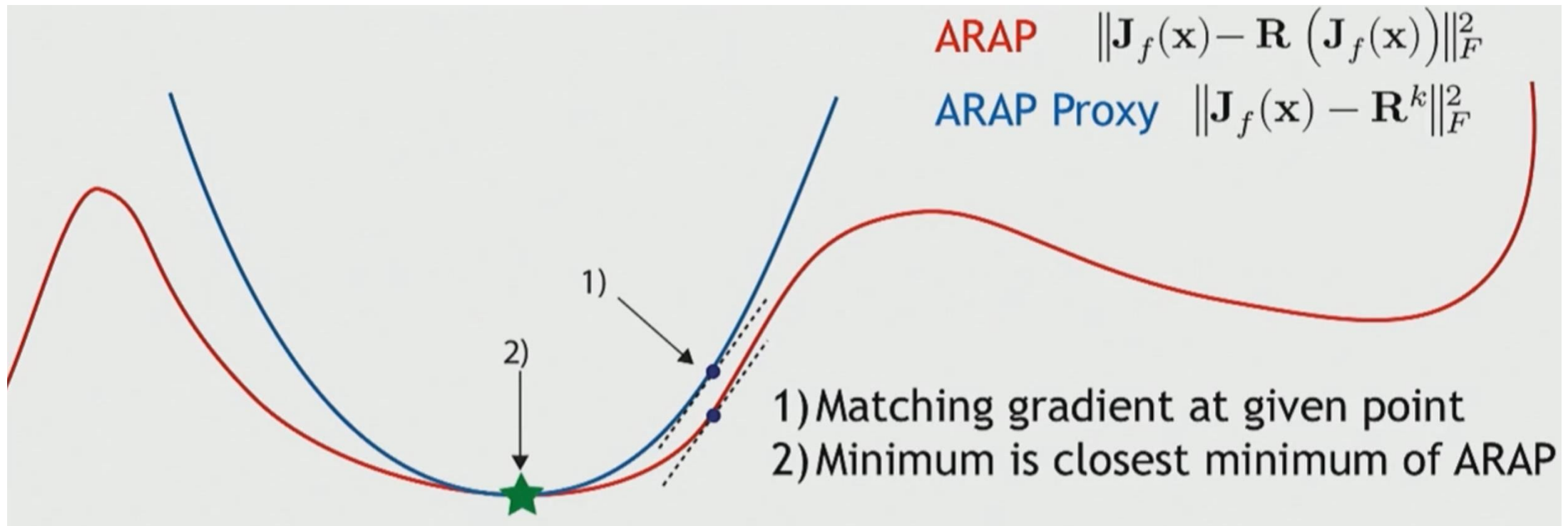
Symmetric Dirichlet
[Smith and Schaefer 2015]

Scalable ✓
Quality ✗



Scalable ✗
Quality ✓

Generalize Local/Global



ARAP Proxy $P^{R_i^n}(J) = \|J - R_i^n\|_F^2 \longrightarrow \mathcal{D}(J) = \|J - R(J)\|_F^2$ **ARAP**

• Majorizer:

$$P^{R_i^n}(J) \geq \mathcal{D}(J) \quad \forall J$$

• Matching gradient:

$$\nabla_J P^{R_i^n}(J_i^n) = \nabla_J \mathcal{D}(J_i^n)$$

• Closest minimizer:

$$\min_J P^{R_i^n}(J) = \text{Proj}(J_i^n)$$

Generalize Local/Global

• Majorizer: $P^{R_i^n}(J) \geq \mathcal{D}(J) \quad \forall J$

• Matching gradient: $\nabla_J P^{R_i^n}(J_i^n) = \nabla_J \mathcal{D}(J_i^n)$

• Closest minimizer: $\min_J P^{R_i^n}(J) = \text{Proj}(J_i^n)$

$$J_i^n = J_i(\mathbf{x}_{n-1}) = USV^T, R_i^n = R(J_i^n) = UV^T$$

$$P_w^R(J) = \|W(J - R)\|_F^2 \quad R = R_i^n$$

$$\nabla_J \|W(J - R)\|_F^2 = \nabla_J \mathcal{D}(J)$$

$$\nabla_J \text{tr}(W^T W (J - R)(J - R)^T) = (W^T W + W W^T)(J - R) = \nabla_J \mathcal{D}(J)$$

$$\mathcal{D}(J) = \|J - R(J)\|_F^2$$

$$\nabla_J \mathcal{D}(J) = 2(J - R)$$

$$W = I$$

$$W = \left(\frac{1}{2} \nabla_J \mathcal{D}(J) (J - R)^{-1} \right)^{1/2}$$

Generalize Local/Global

$$W = \left(\frac{1}{2} \nabla_J \mathcal{D}(J) (J - R)^{-1} \right)^{1/2}$$

$$J = USV^T \quad S = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}$$

$$\mathcal{D}(J) = \mathcal{D}(S) \quad \nabla_J \mathcal{D}(J) = U \nabla_S \mathcal{D}(S) V^T$$

$$(J - R)^{-1} = (USV^T - UV^T)^{-1} = (U(S - I)V^T)^{-1} = V(S - I)^{-1}U^T$$

$$W = \left(\frac{1}{2} U \nabla_S \mathcal{D}(S) V^T V (S - I)^{-1} U^T \right)^{1/2}$$

$$W = U \left(\frac{1}{2} \nabla_S \mathcal{D}(S) (S - I)^{-1} \right)^{1/2} U^T = US_W U^T$$

Generalize Local/Global

$$W = U \left(\frac{1}{2} \nabla_S \mathcal{D}(S) (S - I)^{-1} \right)^{1/2} U^T = U S_W U^T$$

$$\mathcal{D}(J) = \|J\|_F^2 + \|J\|_F^{-2} = (\sigma_1^2 + \sigma_1^{-2} + \sigma_2^2 + \sigma_2^{-2}) \quad S = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}$$

$$\nabla_S \mathcal{D}(S) = \begin{bmatrix} 2(\sigma_1 + \sigma_1^{-3}) & 0 \\ 0 & 2(\sigma_2 + \sigma_2^{-3}) \end{bmatrix}$$

$$S_W = \begin{bmatrix} \sqrt{\frac{(\sigma_1 + \sigma_1^{-3})}{\sigma_1 - 1}} & 0 \\ 0 & \sqrt{\frac{(\sigma_2 + \sigma_2^{-3})}{\sigma_2 - 1}} \end{bmatrix}$$

Generalize Local/Global

Iterate

Local $\mathbf{R}^k = \mathbf{R} \left(\mathbf{J}_f(\mathbf{x}^k) \right)$

Compute weights

Global $\tilde{\mathbf{x}}^{k+1} = \arg \min_{\mathbf{x}} \sum_f \left\| W_f^k \left(\mathbf{J}_f(\mathbf{x}) - \mathbf{R}^k \right) \right\|_F^2$

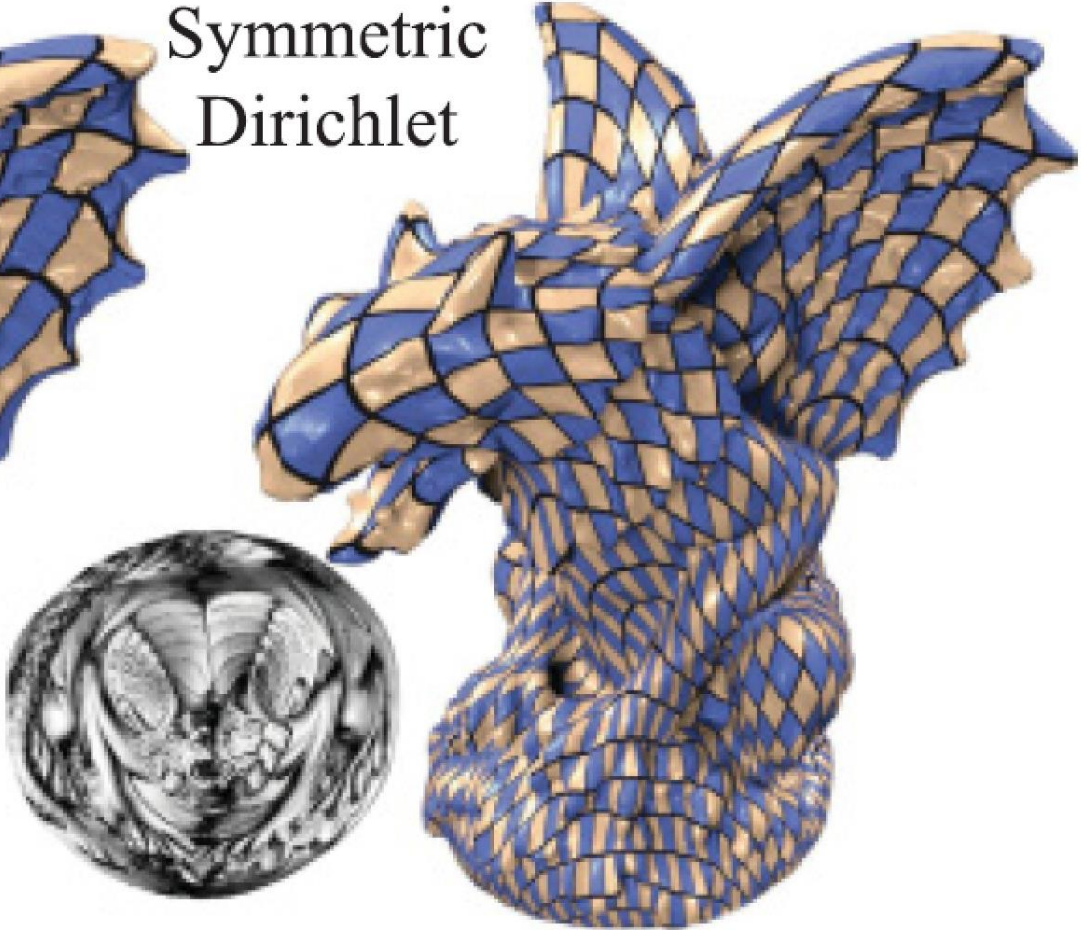
Line search $\mathbf{d} = \tilde{\mathbf{x}}^{k+1} - \mathbf{x}^k$ [Smith and Schaefer 2015]

Result

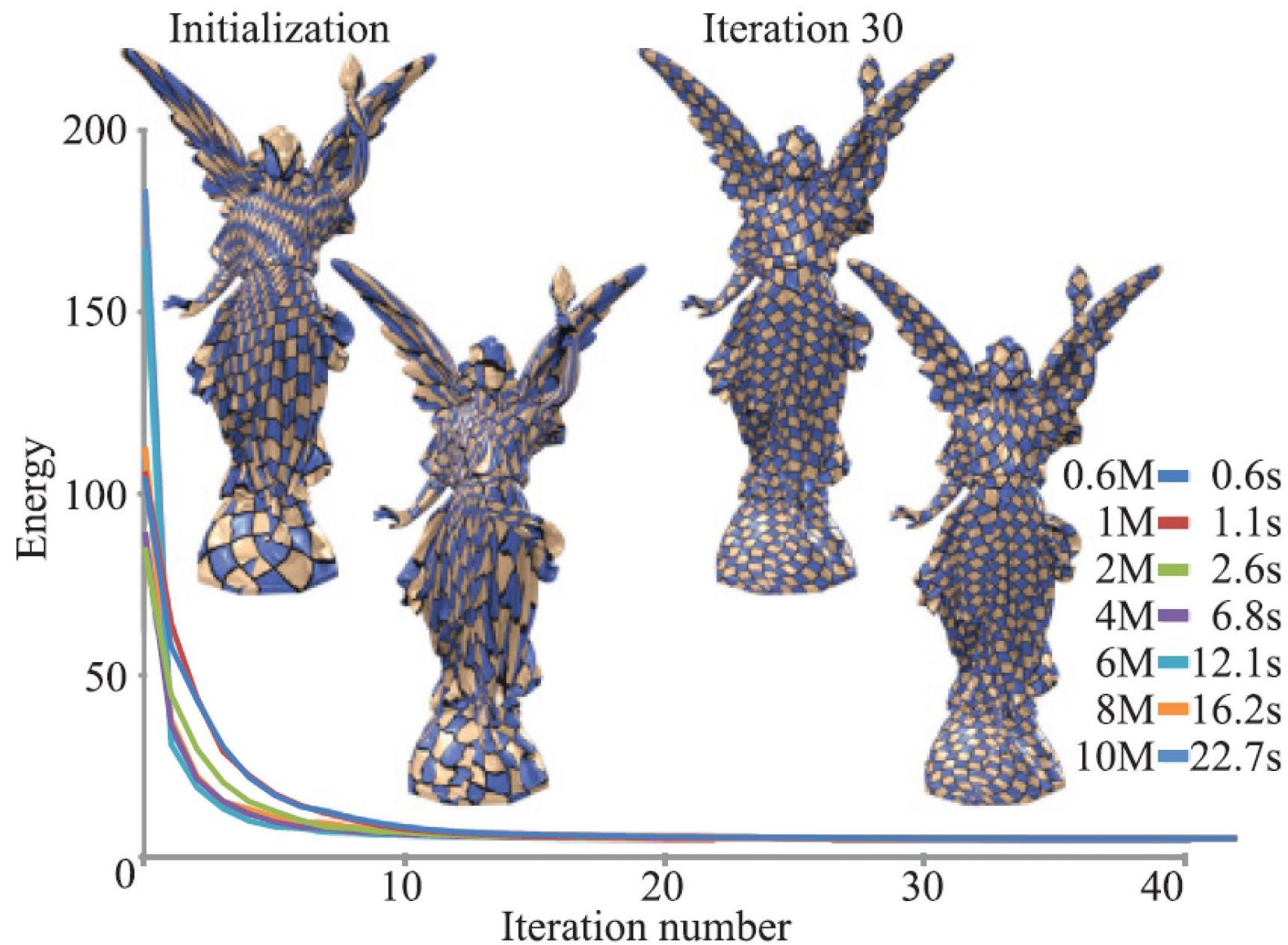
ARAP



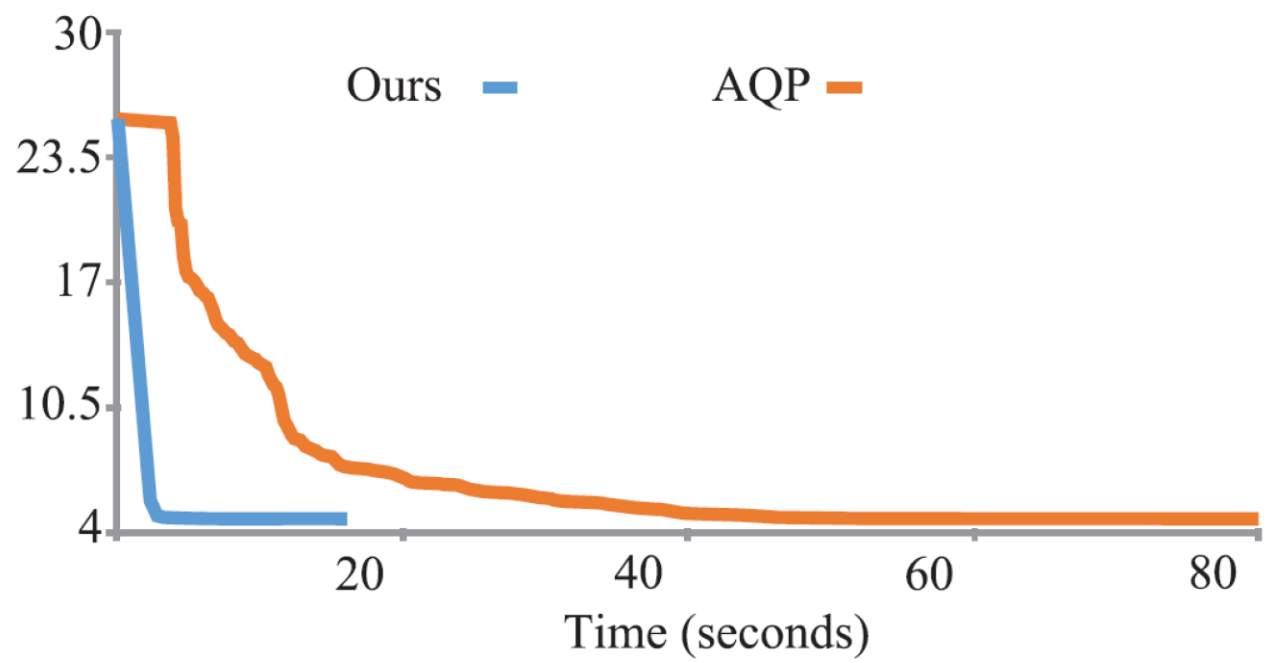
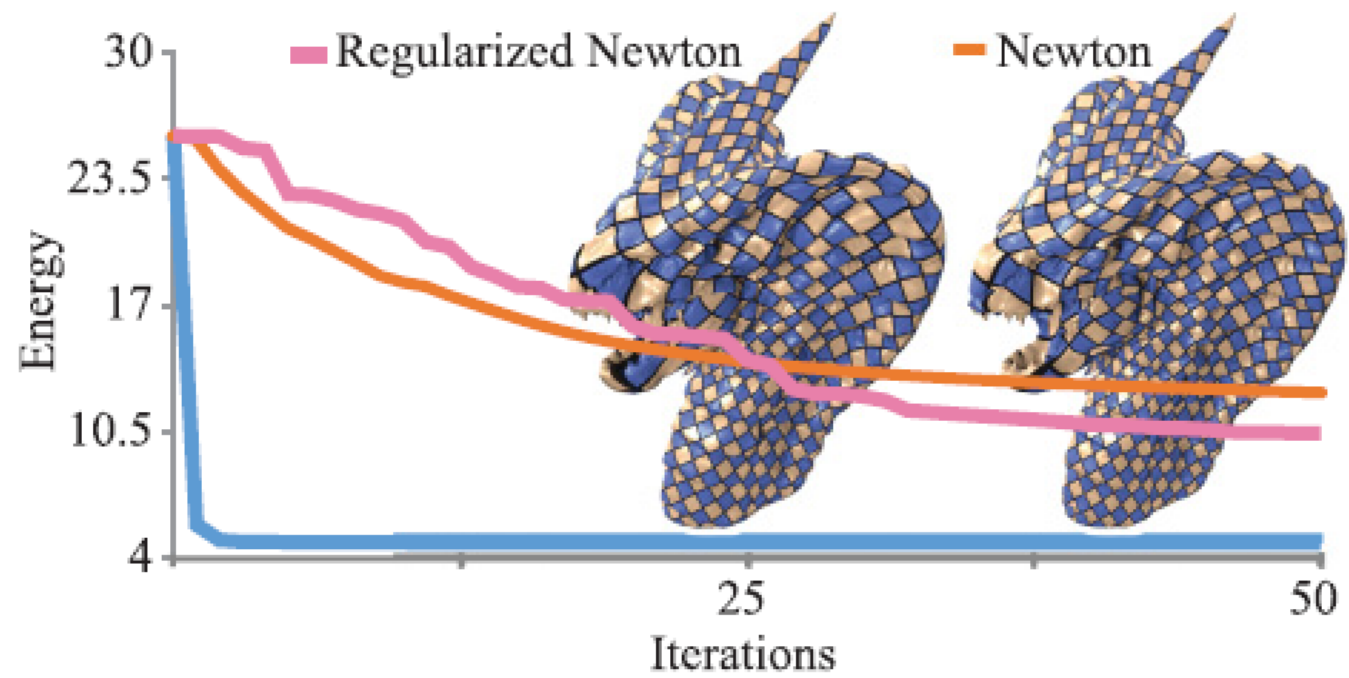
Symmetric
Dirichlet



Result



Result



Result

Blended Cured Quasi-Newton for Distortion Optimization



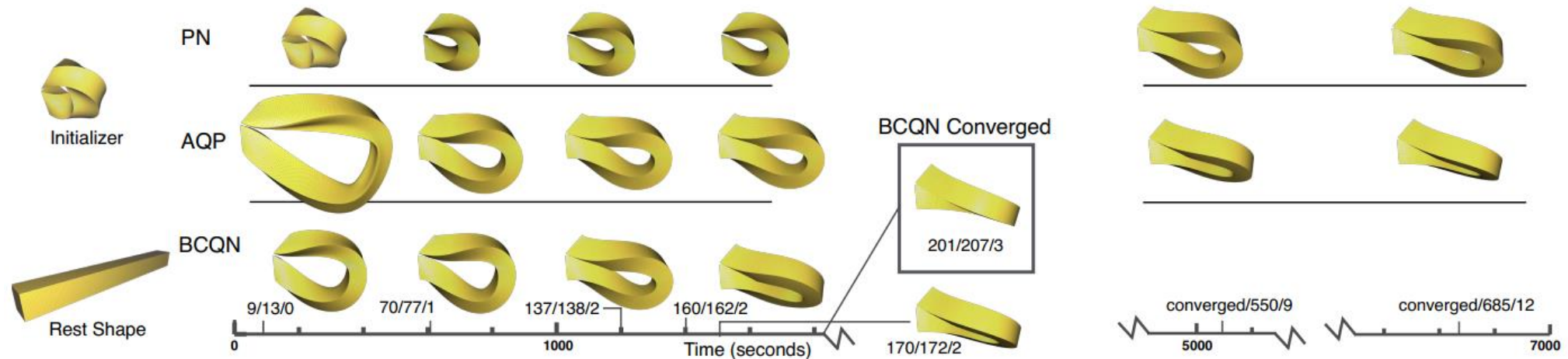
YuFeng Zhu



Robert Bridson



Danny Kaufman



Motivation

$$\min_p f(\mathbf{x}_n + p)$$

$$f(\mathbf{x}_n + p) \approx f(\mathbf{x}_n) + \nabla f(\mathbf{x}_n)^T p + \frac{1}{2} p^T H p$$

$$H p = -\nabla f(\mathbf{x}_n)$$

- **First-order methods** build descent steps by preconditioning the gradient with a **fixed proxy matrix**, which **often suffer from slower convergence as lacking of higher-order information**.
- **Newton-type methods** uses the energy Hessian, $\nabla^2 f(\mathbf{x})$, to form a proxy matrix, which can achieve the most rapid convergence but require the costly **assembly, factorization and backsolve of new linear systems per iterate**.

Quasi-Newton Methods

$$f(\mathbf{x}) \approx f(\mathbf{x}_{n+1}) + \nabla f(\mathbf{x}_{n+1})^T (\mathbf{x} - \mathbf{x}_{n+1}) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_{n+1})^T H_{n+1} (\mathbf{x} - \mathbf{x}_{n+1})$$

$$\nabla f(\mathbf{x}_{n+1}) + H_{n+1} (\mathbf{x}_n - \mathbf{x}_{n+1}) = \nabla f(\mathbf{x}_n)$$

$$H_{n+1} (\mathbf{x}_{n+1} - \mathbf{x}_n) = \nabla f(\mathbf{x}_{n+1}) - \nabla f(\mathbf{x}_n)$$

secant equation

$$H_{n+1} s_n = y_n \quad s_n = D_{n+1} y_n$$

$$p_{n+1} = -D_{n+1} \nabla f(\mathbf{x}_{n+1})$$

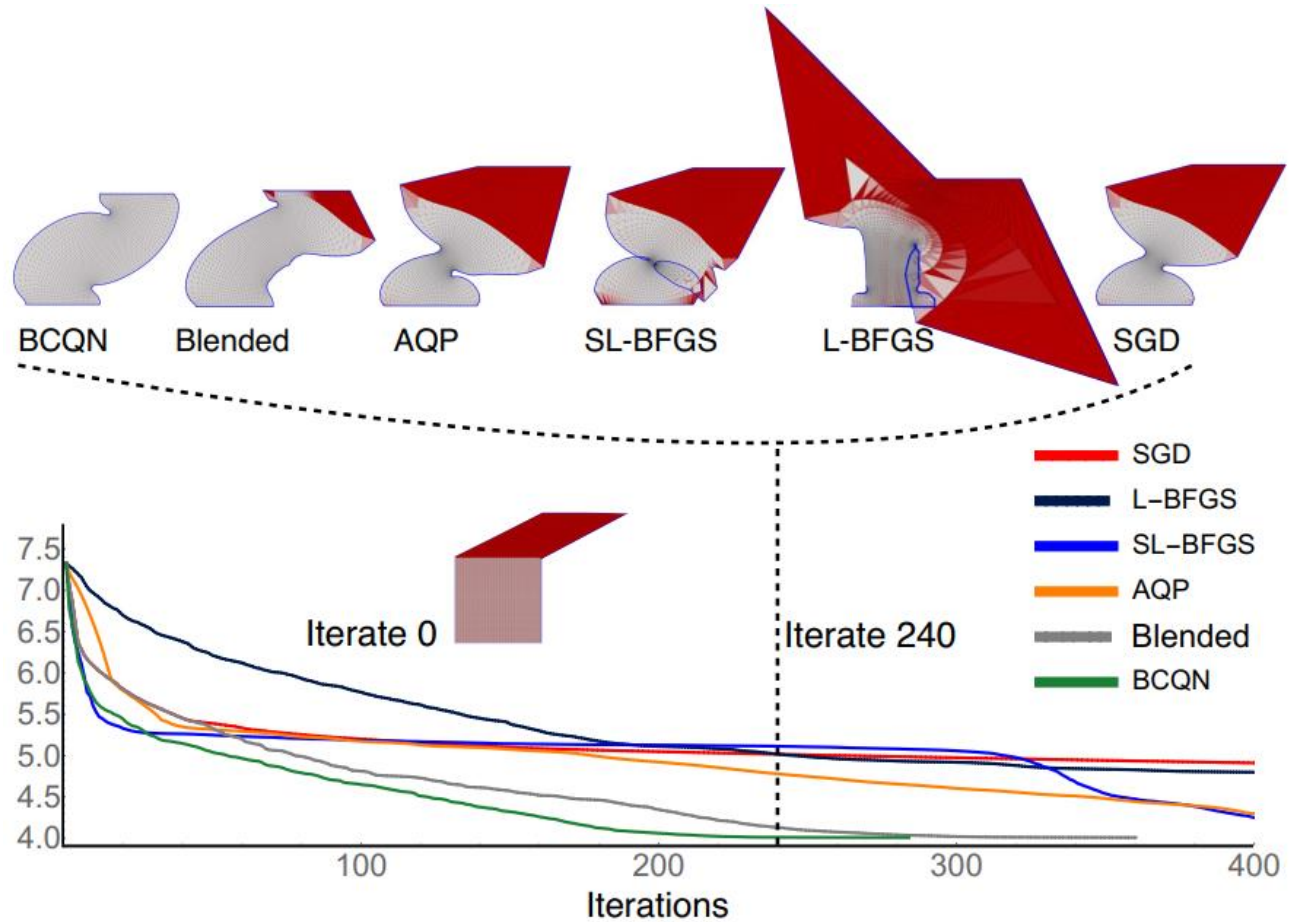
$$D_{n+1} = QN(\mathbf{z}, D_n) = V(\mathbf{z})^T D_n V(\mathbf{z}) + \frac{s_n s_n^T}{s_n^T \mathbf{z}}, \quad V(\mathbf{z}) = I - \frac{\mathbf{z} s_n^T}{s_n^T \mathbf{z}} \quad \mathbf{z}: \text{The difference in gradients}$$

$$s_n = QN(\mathbf{z}, D_n) \mathbf{z}$$

$$D_{n+1} = QN(y_n, D_n)$$

Quasi-Newton Methods

Quasi-Newton methods lie in between two methods. They employ sequential gradients to update approximations of the system Hessian, per descent iterate. However, the secant approximation can implicitly create a dense proxy, unlike the sparse true Hessian, direct and incorrect coupling distant vertices.



Blended Quasi-Newton

- The difference in gradients would be the better behaved Ls than y , which may introduce spurious coupling or have badly scaled entries near distorted triangles

$$D_{n+1} = QN(Ls_n, D_n)$$

- However, to achieve the superlinear convergence BFGS offers, near solutions we wish to come closer to satisfying the secant equation, and so aim to move towards using y instead.

$$z_n = (1 - \beta_n)y_n + \beta_n Ls_n$$

$$\beta_n = \min_{\beta \in [0,1]} \|y_n - \beta Ls_n\|^2, \quad \beta_n = \text{proj}_{[0,1]} \left(\frac{y_n^T Ls_n}{\|Ls_n\|^2} \right)$$

$$D_{n+1} = QN(z_n, D_n)$$

Result

Second-order method

Paper List

- Geometric Optimization via Composite Majorization
- Piecewise Linear Mapping Optimization Based on The Complex View
- Progressive Parameterizations

Geometric Optimization via Composite Majorization



Anna Shtengel



Roi Poranne



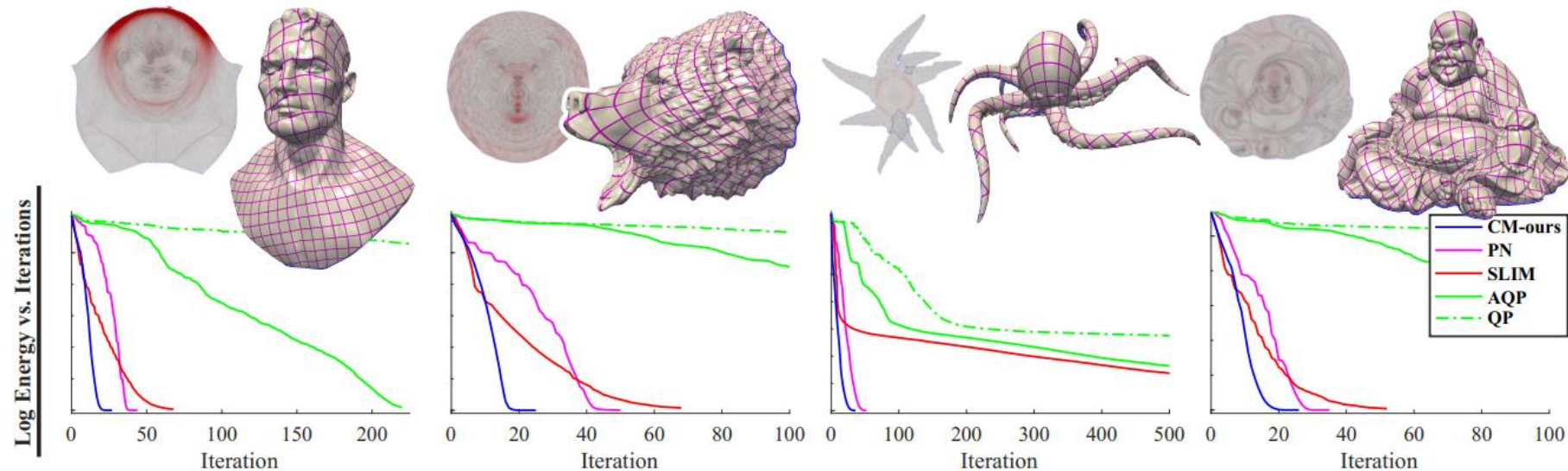
Olga Sorkine-Hornung



Shahar Kovalsky



Yaron Lipman



Motivation

$$\min_{\mathbf{x}} f(\mathbf{x}) = \sum_i |t_i| \mathcal{D}(J_i(\mathbf{x}))$$

1. Initial point: $\mathbf{x}_0, n = 0$

2. Descent direction: $Hp = -\nabla f(x_n)$

3. Step size: $\min_{\alpha} f(x_n + \alpha p)$

4. Update: $\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha p, n = n + 1$

analytic and simple to evaluate
convex approximate Hessian H

Method

$$f(\mathbf{x}) = \sum_i h_i(\mathbf{g}_i(\mathbf{x})) = \sum_i (h_i \circ \mathbf{g}_i)(\mathbf{x})$$

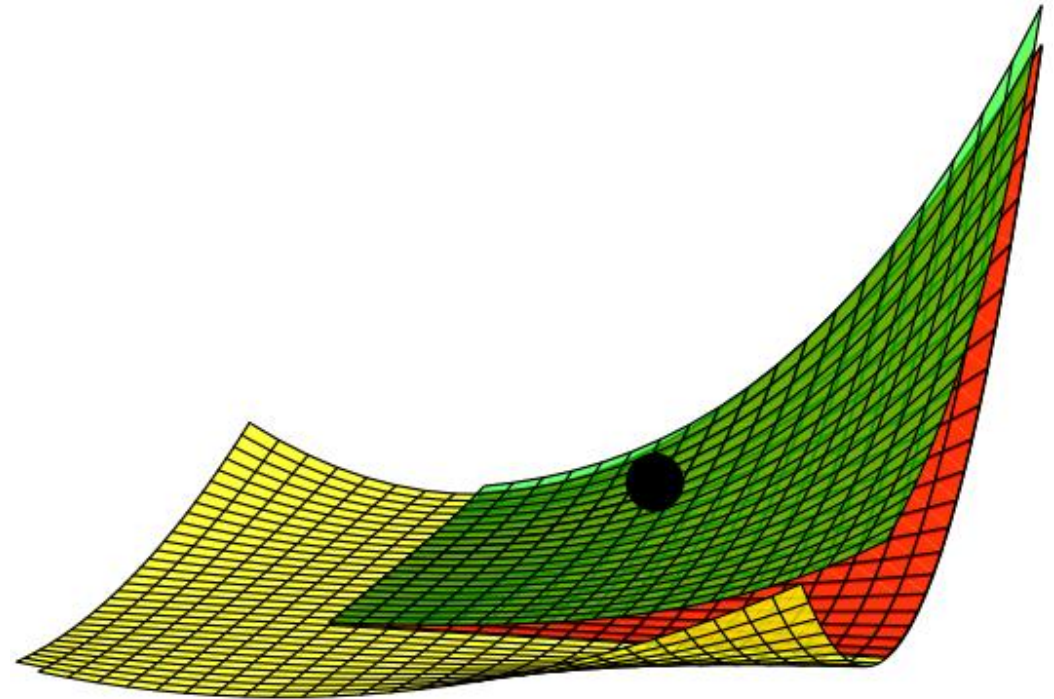
$$h_i : R^k \rightarrow R, \quad h_i = h_i^+ + h_i^-$$

$$\mathbf{g}_i : R^n \rightarrow R^k, \quad \mathbf{g}_i = \mathbf{g}_i^+ + \mathbf{g}_i^-$$

$$f(\mathbf{x}_n + \mathbf{p}) \approx f(\mathbf{x}_n) + \nabla f(\mathbf{x}_n)^T \mathbf{p} + \frac{1}{2} \mathbf{p}^T H \mathbf{p}$$

\bar{f} is a local convex majorizer of f

$$H = \nabla^2 \bar{f}$$



Method

convex majorizer

$$\bar{r}(\mathbf{x}; \mathbf{x}_0) = r^+(\mathbf{x}) + r^-(\mathbf{x}_0) + \nabla r^-(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0)$$

concave minorizer

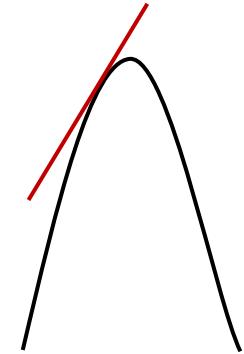
$$\underline{r}(\mathbf{x}; \mathbf{x}_0) = r^-(\mathbf{x}) + r^+(\mathbf{x}_0) + \nabla r^+(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0)$$

$$f(\mathbf{x}) = h(\mathbf{g}(\mathbf{x})) = h(g_1(\mathbf{x}), \dots, g_k(\mathbf{x}))$$

$$\mathbf{u}_0 = \mathbf{g}(\mathbf{x}_0) \quad s_j(\mathbf{u}) = \text{sign} \left(\frac{\partial \bar{h}}{\partial u_j}(\mathbf{u}; \mathbf{u}_0) \right)$$

$$[g_j](\mathbf{x}; \mathbf{x}_0) = \begin{cases} \bar{g}_j(\mathbf{x}; \mathbf{x}_0) & s_j(\mathbf{u}_0) > 0 \\ \underline{g}_j(\mathbf{x}; \mathbf{x}_0) & s_j(\mathbf{u}_0) < 0 \end{cases}$$

$$\bar{f}(\mathbf{x}; \mathbf{x}_0) = \bar{h}([g](\mathbf{x}; \mathbf{x}_0); \mathbf{u}_0)$$



Method

$$\bar{f}(\mathbf{x}; \mathbf{x}_0) = \bar{h}([\mathbf{g}](\mathbf{x}; \mathbf{x}_0); \mathbf{u}_0)$$

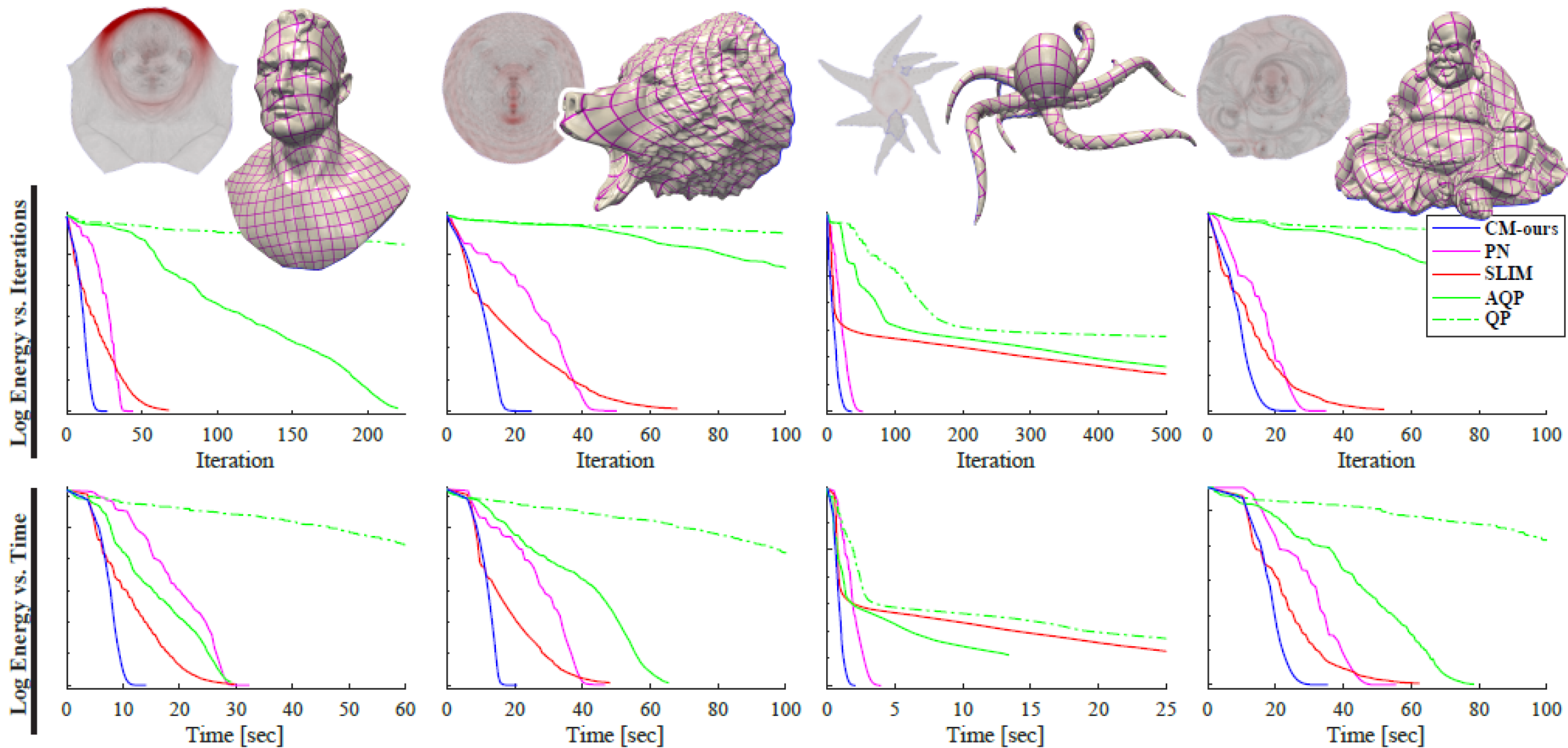
\bar{f} is convex majorier

$$\nabla_{\mathbf{x}}^2 \bar{f}(\mathbf{x}; \mathbf{x}_0) = \frac{\partial [\mathbf{g}]^T}{\partial \mathbf{x}} \nabla^2 h + \frac{\partial [\mathbf{g}]}{\partial \mathbf{x}} + \sum_j \frac{\partial \bar{h}}{\partial u_j} \begin{cases} \nabla^2 g_j^+ & s_j(\mathbf{u}_0) > 0 \\ \nabla^2 g_j^- & s_j(\mathbf{u}_0) < 0 \end{cases}$$

$$\bar{f}(\mathbf{x}; \mathbf{x}_0) = \bar{h}([\mathbf{g}](\mathbf{x}; \mathbf{x}_0); \mathbf{u}_0) \geq \bar{h}(\mathbf{g}(\mathbf{x}); \mathbf{u}_0) \geq h(\mathbf{g}(\mathbf{x}))$$

$$H = \nabla^2 \bar{f}(\mathbf{x}; \mathbf{x}_0) \Big|_{\mathbf{x}=\mathbf{x}_0}$$

Result



Result

Piecewise linear mapping optimization based on the complex view



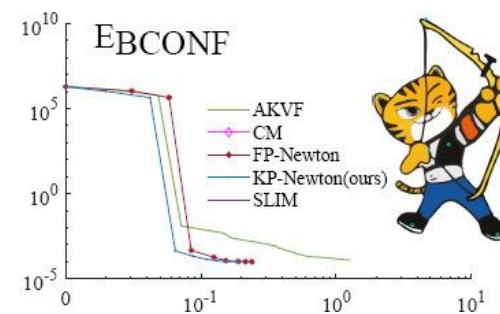
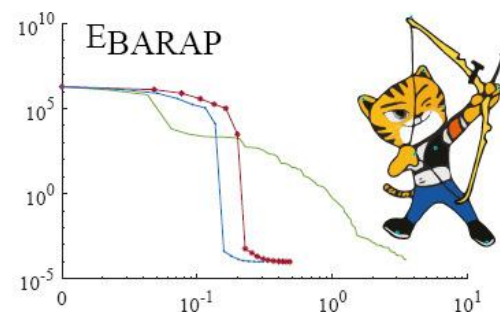
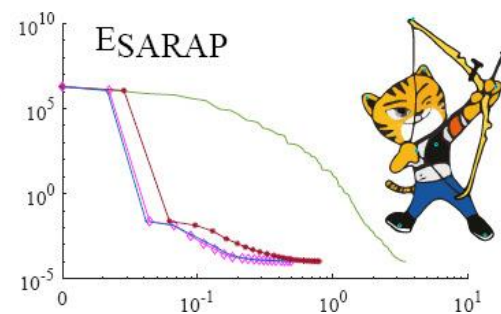
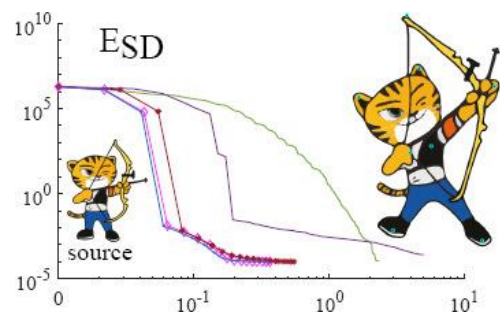
Björn Golla



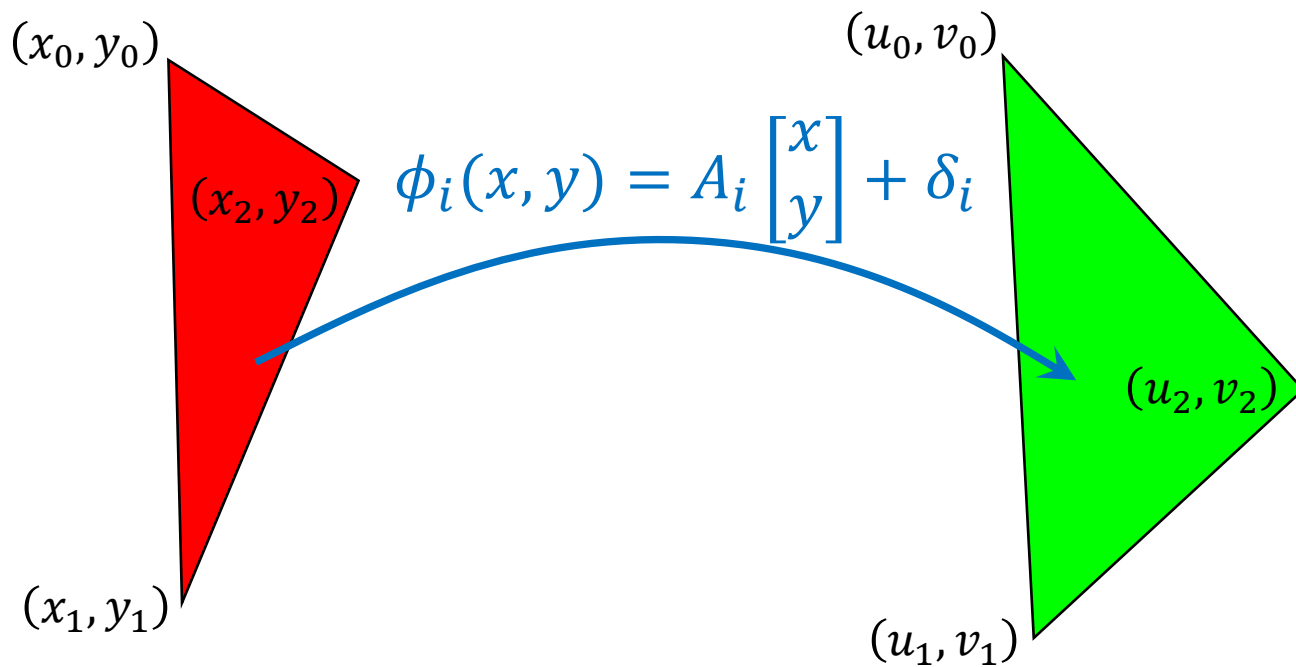
Hans-Peter Seidel



Renjie Chen



Real view

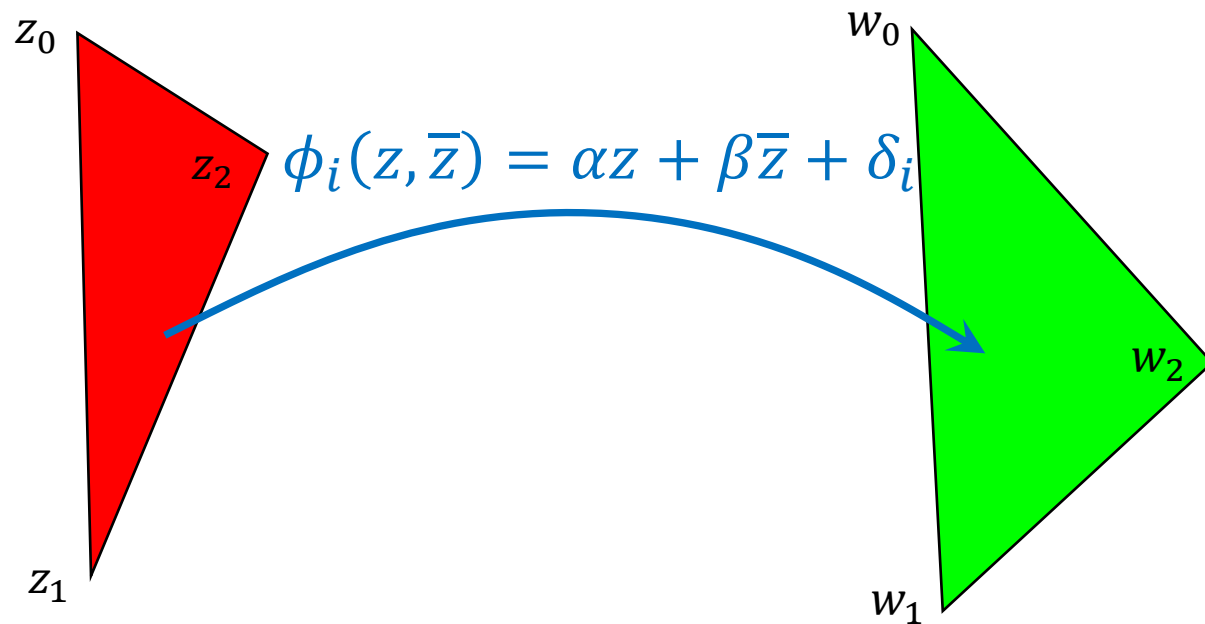


$$\begin{bmatrix} u \\ v \end{bmatrix} = A_i \begin{bmatrix} x \\ y \end{bmatrix} + c_i$$

$$J_i = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} = A_i$$

$$J_i = \begin{bmatrix} u_1 - u_0 & u_2 - u_0 \\ v_1 - v_0 & v_2 - v_0 \end{bmatrix} \begin{bmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{bmatrix}^{-1}$$

Complex view



$$z = x + iy, \quad \bar{z} = x - iy$$

$$x = \frac{z + \bar{z}}{2}, \quad y = \frac{z - \bar{z}}{2i}$$

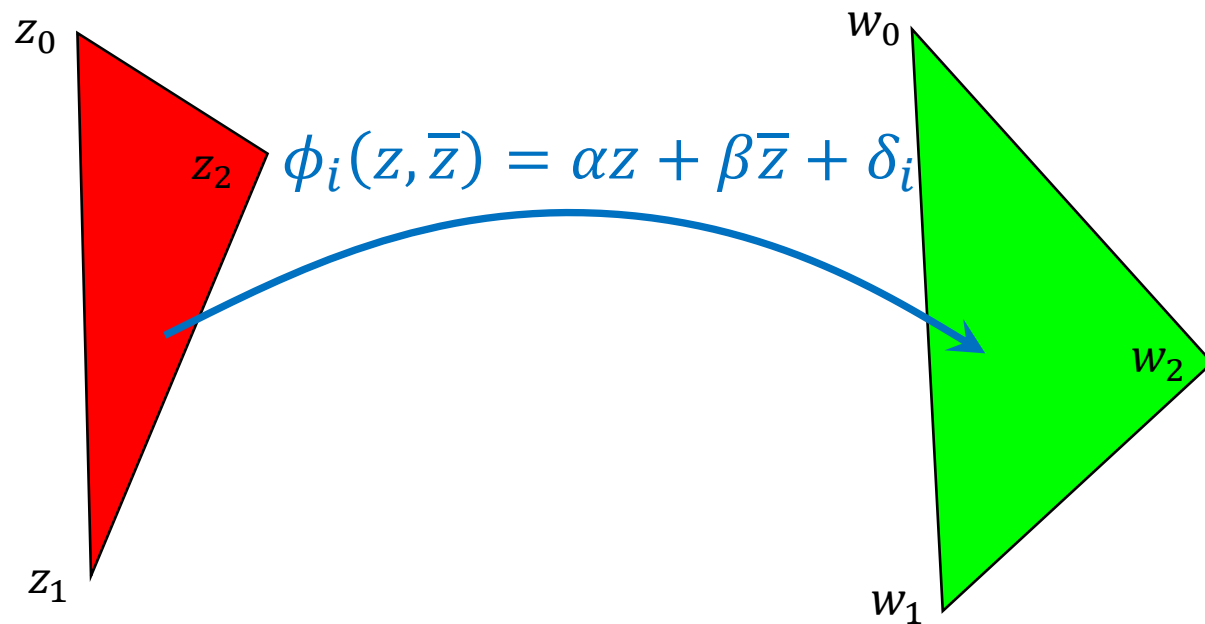
$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (ax + by) + i(cx + dy)$$

$$\left(\frac{a+d}{2} + i \frac{c-b}{2} \right) z + \left(\frac{a-d}{2} + i \frac{c+b}{2} \right) \bar{z}$$

$$\alpha = \frac{a+d}{2} + i \frac{c-b}{2}, \quad \beta = \frac{a-d}{2} + i \frac{c+b}{2}$$

$$\Sigma = |\alpha| + |\beta|, \quad \sigma = \left| |\alpha| - |\beta| \right|$$

Complex view



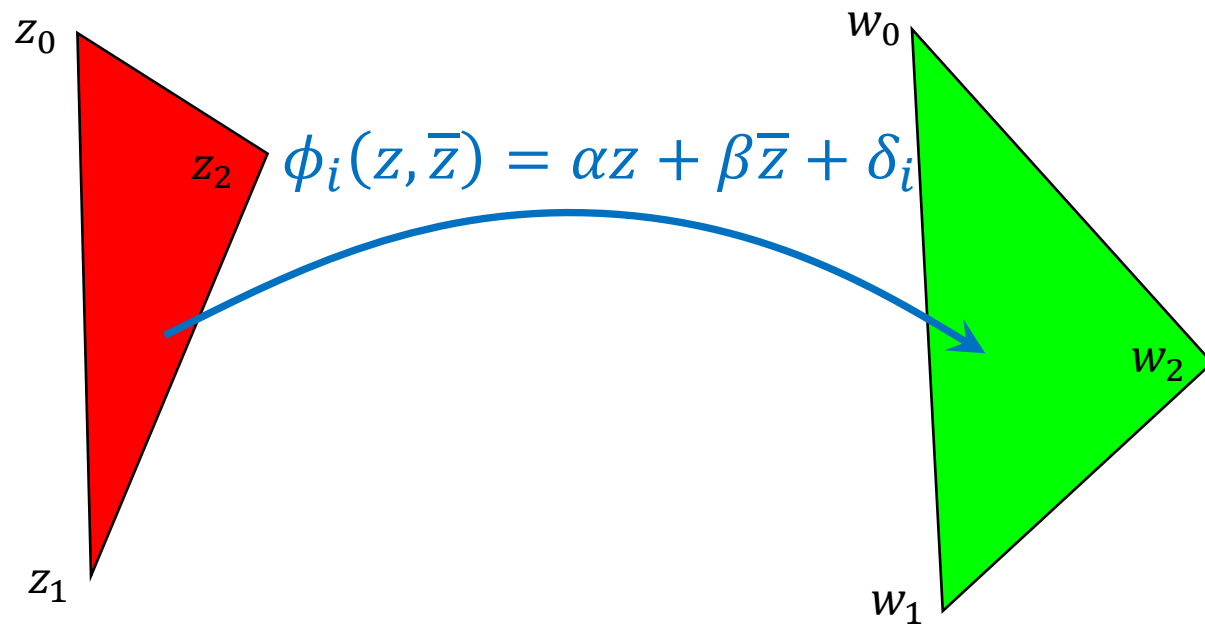
$$\begin{bmatrix} z_0 & \bar{z}_0 & 1 \\ z_1 & \bar{z}_1 & 1 \\ z_2 & \bar{z}_2 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \delta \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} z_1 - z_0 & \bar{z}_1 - \bar{z}_0 \\ z_2 - z_0 & \bar{z}_2 - \bar{z}_0 \end{bmatrix}^{-1} \begin{bmatrix} w_1 - w_0 \\ w_2 - w_0 \end{bmatrix}$$

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \frac{i}{4|t_i|} \begin{bmatrix} \bar{z}_2 - \bar{z}_0 & \bar{z}_0 - \bar{z}_1 \\ z_0 - z_2 & z_1 - z_0 \end{bmatrix} \begin{bmatrix} w_1 - w_0 \\ w_2 - w_0 \end{bmatrix}$$

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \frac{i}{4|t_i|} \begin{bmatrix} \bar{z}_1 - \bar{z}_2 & \bar{z}_2 - \bar{z}_0 & \bar{z}_0 - \bar{z}_1 \\ z_2 - z_1 & z_0 - z_2 & z_1 - z_0 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

Complex view



$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \frac{i}{4|t_i|} \begin{bmatrix} \bar{z}_1 - \bar{z}_2 & \bar{z}_2 - \bar{z}_0 & \bar{z}_0 - \bar{z}_1 \\ z_2 - z_1 & z_0 - z_2 & z_1 - z_0 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

$$e^0 = z_1 - z_2, \quad e^1 = z_2 - z_0, \quad e^2 = z_0 - z_1$$

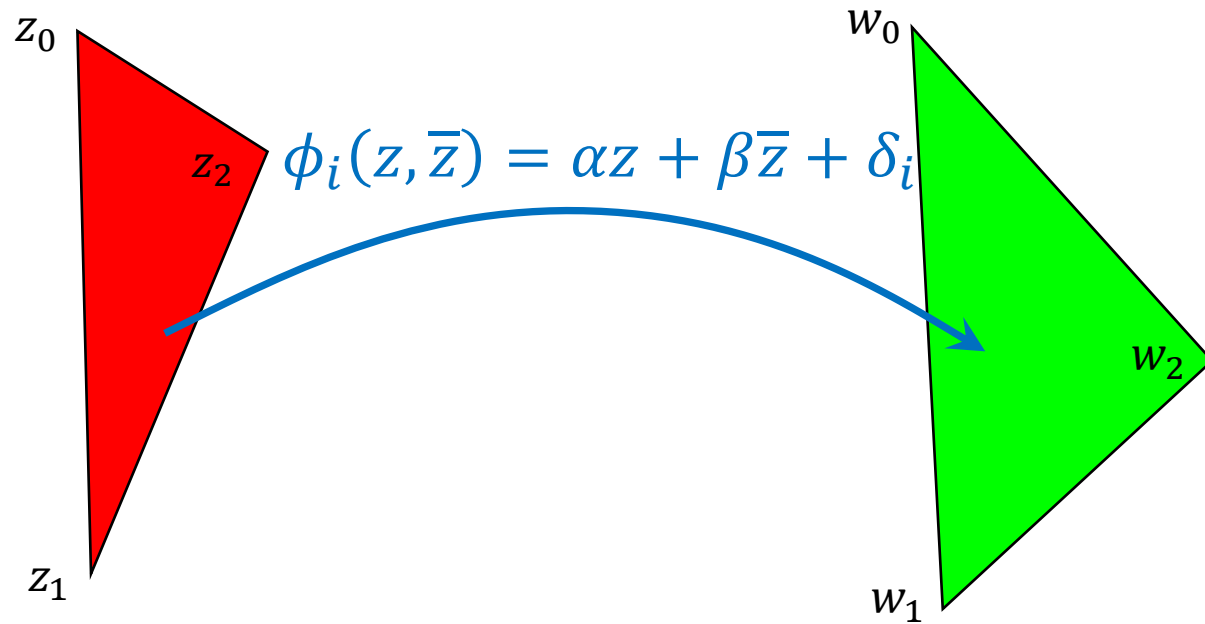
$$\alpha = \frac{i}{4|t_i|} \overline{[e^0 \quad e^1 \quad e^2]W}$$

$$\beta = -\frac{i}{4|t_i|} [e^0 \quad e^1 \quad e^2]W$$

$$D = \frac{i}{4|t_i|} \overline{[e^0 \quad e^1 \quad e^2]}$$

$$\alpha = DW, \quad \beta = \bar{D}W$$

Complex view



$$\alpha = DW, \quad \beta = \bar{D}W$$

$$\alpha = \begin{bmatrix} \operatorname{Re}(\alpha) \\ \operatorname{Im}(\alpha) \end{bmatrix}_{2 \times 1}, \quad W = \begin{bmatrix} \operatorname{Re}(W) \\ \operatorname{Im}(W) \end{bmatrix}_{6 \times 1}$$

$$D = \begin{bmatrix} \operatorname{Re}(D) & -\operatorname{Im}(D) \\ \operatorname{Im}(D) & \operatorname{Re}(D) \end{bmatrix}_{2 \times 6}$$

Method

$$\Sigma = |\alpha| + |\beta|, \quad \sigma = |\alpha| - |\beta|$$

$$r = |\alpha|^2, \quad s = |\beta|^2$$

$$\alpha = DW, \quad \beta = \bar{D}W$$

$$E = (\Sigma_i^2 + \sigma_i^2 + \Sigma_i^{-2} + \sigma_i^{-2})$$

$$= (\Sigma_i^2 + \sigma_i^2) \left(1 + \frac{1}{\Sigma_i^2 \sigma_i^2} \right)$$

$$= (r + s)(1 + (r - s)^{-2})$$

$$\xi_1 = \nabla_r E, \quad \xi_2 = \nabla_s E$$

$$\eta_1 = \nabla_r^2 E, \quad \eta_2 = \nabla_s^2 E, \quad \eta_3 = \nabla_r \nabla_s E$$

$$H_{6 \times 6} = \nabla^2 E = M^T K M$$

$$M_{4 \times 6} = \begin{bmatrix} D \\ \bar{D} \end{bmatrix}, \quad K = \begin{bmatrix} 2\xi_1 I + 4\eta_1 \alpha \alpha^T & 4\eta_3 \alpha \beta^T \\ 4\eta_3 \beta \alpha^T & \xi_2 I + 4\eta_2 \beta \beta^T \end{bmatrix}$$

Method

$$H_{6 \times 6} = \nabla^2 E = M^T K M$$

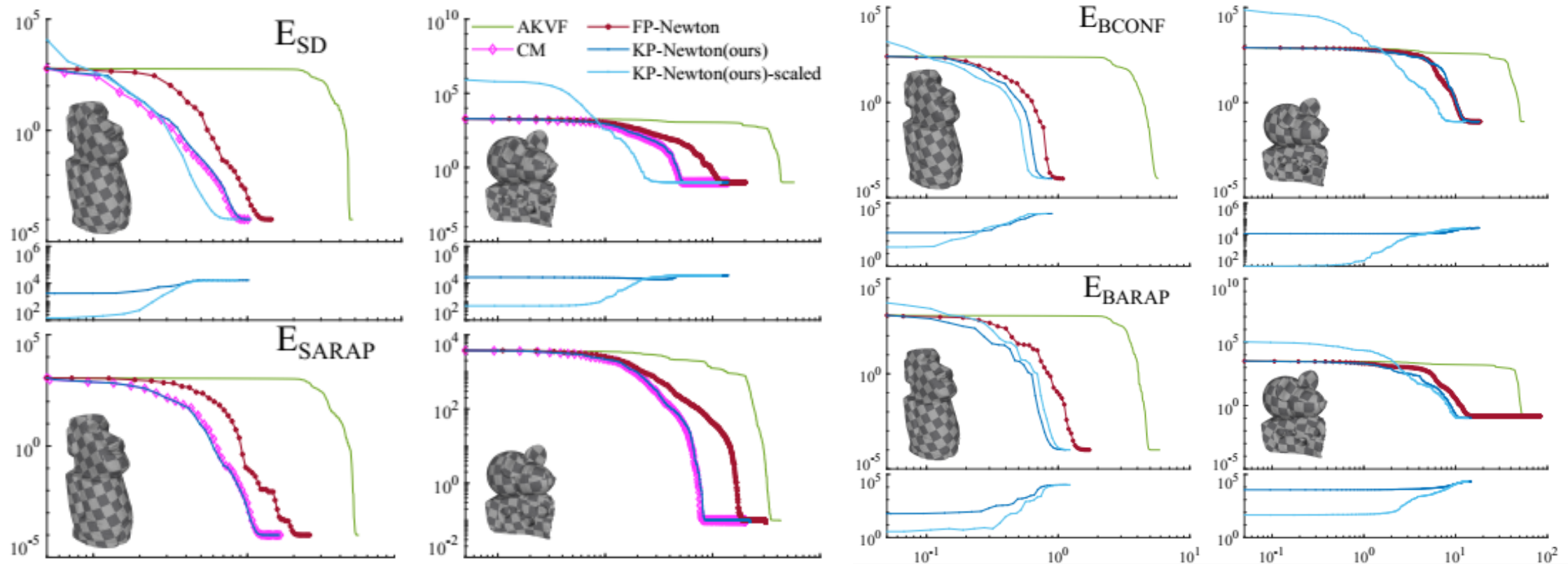
$$M_{4 \times 6} = R_{4 \times 4} Q_{4 \times 6}$$

R is lower triangular matrix, Q is orthonormal

$$H = Q^T (R^T K R) Q$$

The 6×6 PSD projection of H is therefore equivalent to the 4×4 PSD projection of $R^T K R$, since Q is orthonormal.

Result

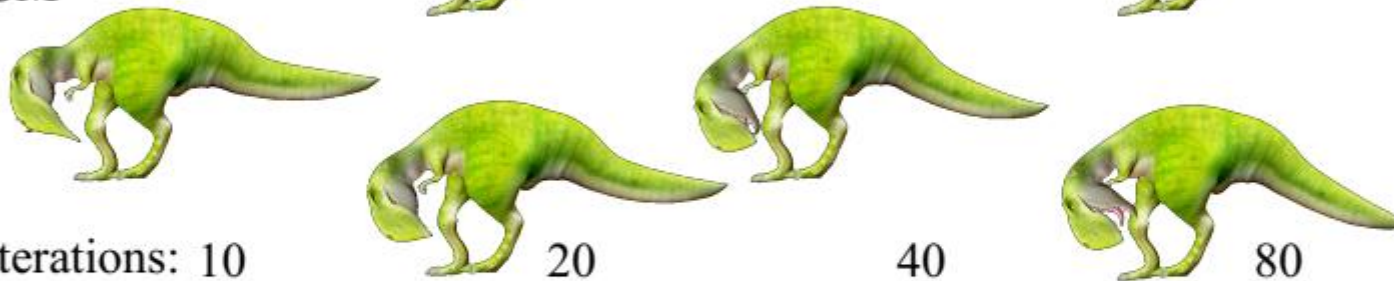


Result

KP-Newton



CM

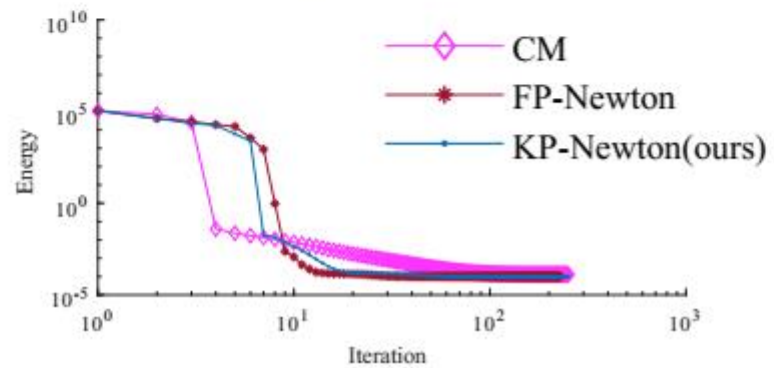


Iterations: 10

20

40

80



source

Progressive Parameterizations



Ligang Liu



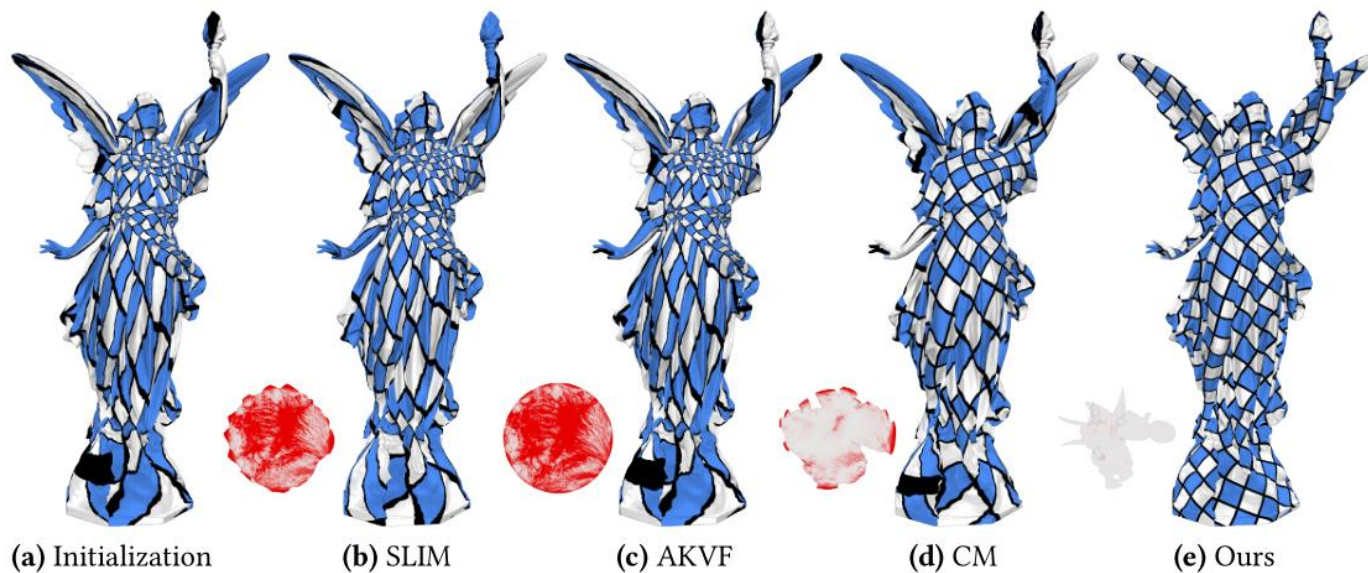
Chunyang Ye



Ruiqi Ni



Xiao-Ming Fu



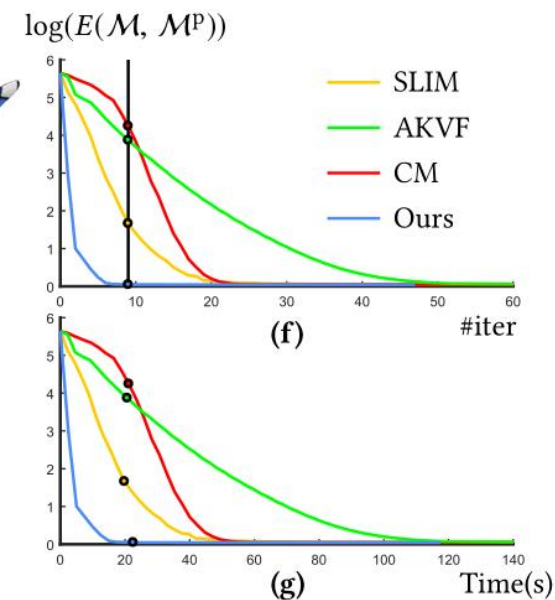
(a) Initialization

(b) SLIM

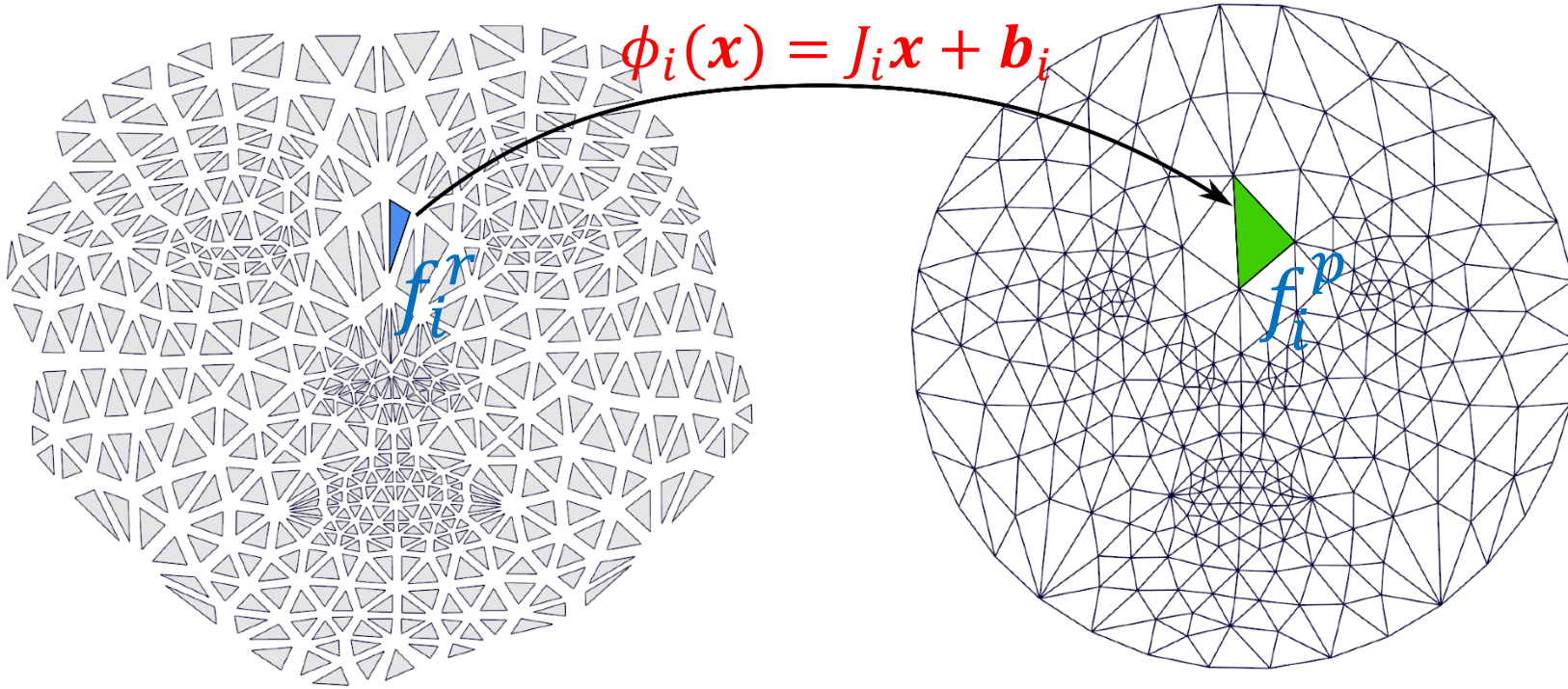
(c) AKVF

(d) CM

(e) Ours



Motivation

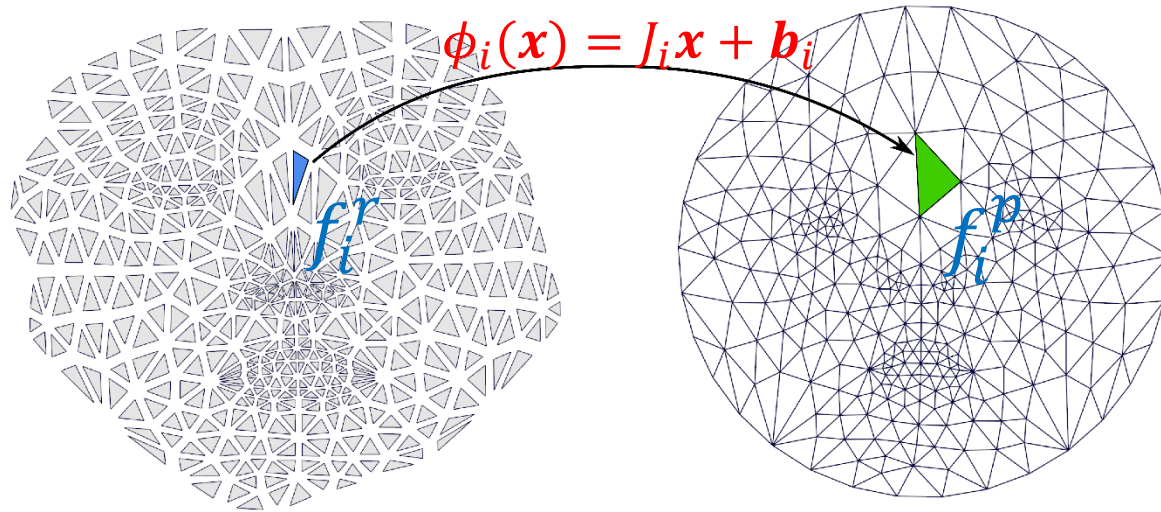


Reference M^r : A set of individual triangles

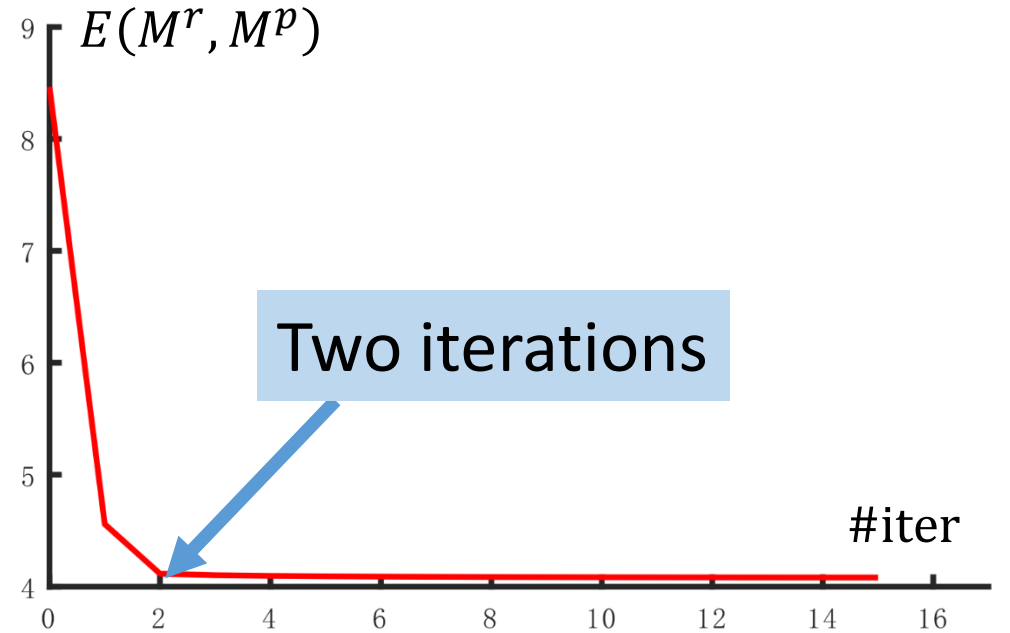
Parameterized mesh M^p

Existing methods choose *the triangles f_i of input mesh M* as reference triangles. The energy is *numerically difficult to optimize*, leading to numerous iterations and high computational cost.

Motivation



If $D(f_i^r, f_i^p) \leq K, \forall i$, only a few iterations in the optimization of $E(M^r, M^p)$ are necessary.



Goal: find a triangle between f_i and f_i^p as the reference f_i^r that satisfies $D(f_i^r, f_i^p) \leq K$.

New reference triangles

- Exponential function :

$$J_i(t) = U_i \text{diag}(\sigma_i^t, \tau_i^t) V_i^T$$

where $J_i = U_i \text{diag}(\sigma_i, \tau_i) V_i^T$

- Bounded distortion:

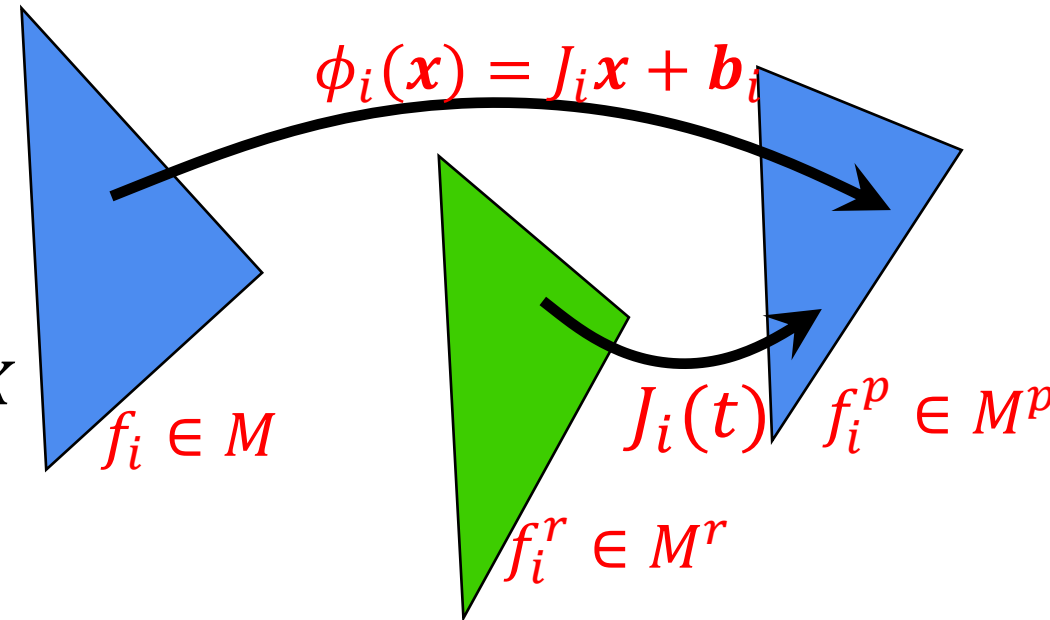
$$D(f_i^r, f_i^p) = \frac{1}{4} (\sigma_i^{2t} + \sigma_i^{-2t} + \tau_i^{2t} + \tau_i^{-2t}) \leq K$$

It is strictly increasing *w.r.t* t .

- Maximize the guidance of reference triangle:

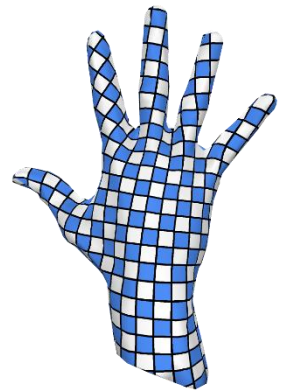
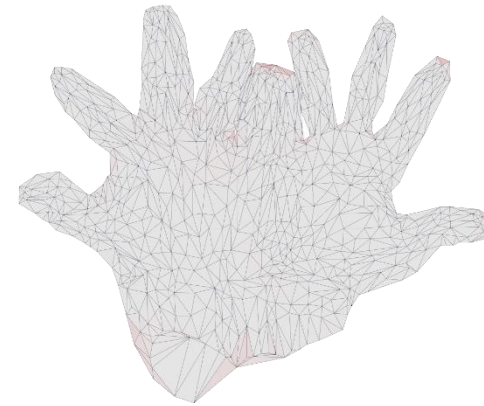
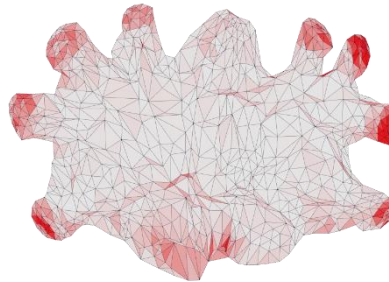
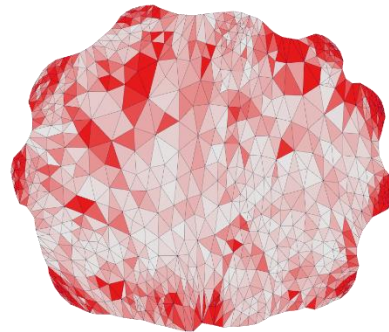
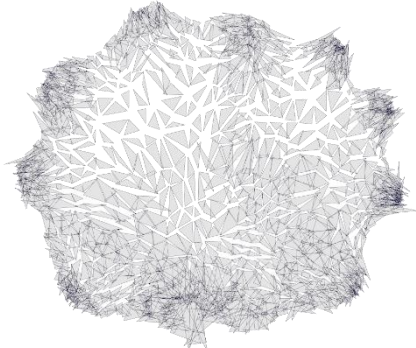
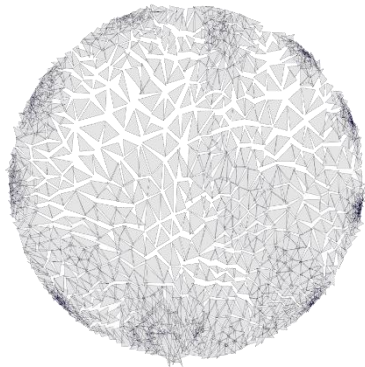
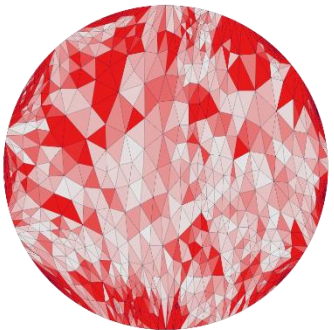
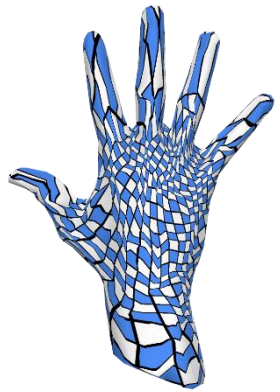
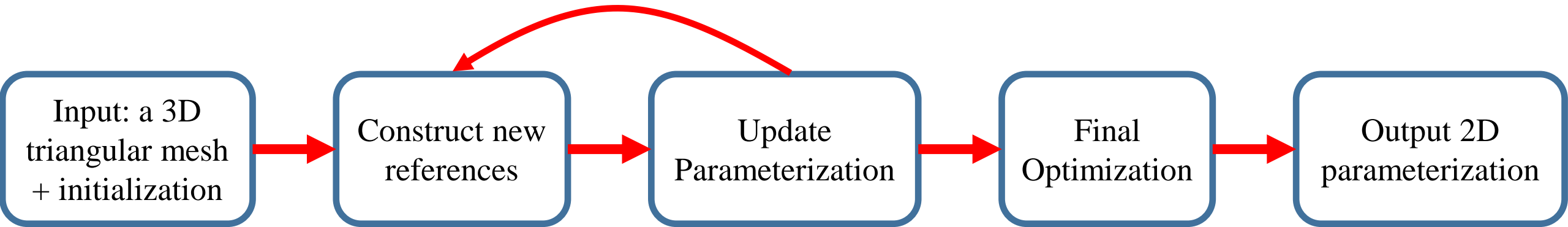
$$\frac{1}{4} (\sigma_i^{2t_i} + \sigma_i^{-2t_i} + \tau_i^{2t_i} + \tau_i^{-2t_i}) = K$$

Newton-Raphson method



Construction of new reference

Our algorithm – Progressive parameterization



Hybrid solver

- SLIM [Rabinovich et al. 2017]
 - A reweighting scheme
 - Pros: effectively penalize the maximum distortion
 - Cons: a poor convergence rate
- CM [Shtengel et al. 2017]
 - Pros: converge quickly
 - Cons: cannot reduce large distortion quickly
- Hybrid
 - First perform SLIM solver
 - Then use the CM solver

Result