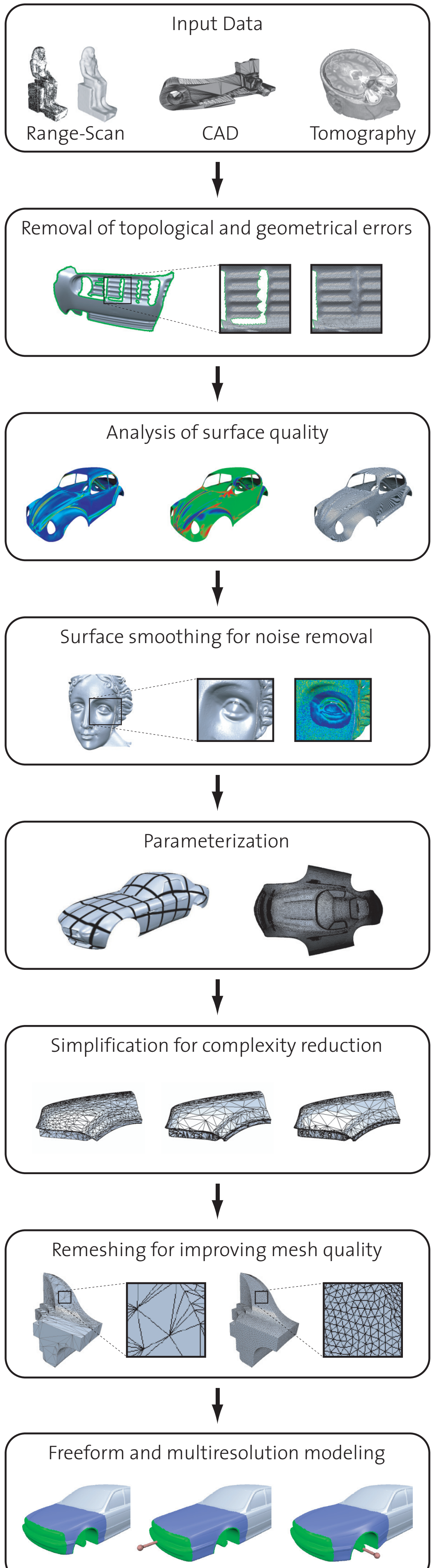
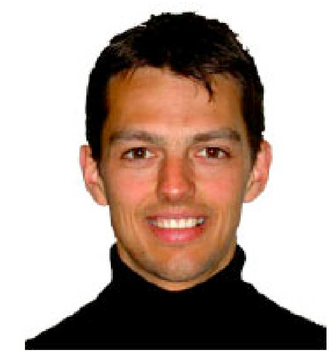


Geometric Modeling Based on Triangle Meshes


Mario Botsch, Mark Pauly
ETH Zurich



Christian Rössl
INRIA Sophia Antipolis


Stephan Bischoff, Leif Kobbelt
RWTH Aachen


Application Areas

- Computer games
- Movie production
- Engineering
- Medical applications
- Architecture
- etc.

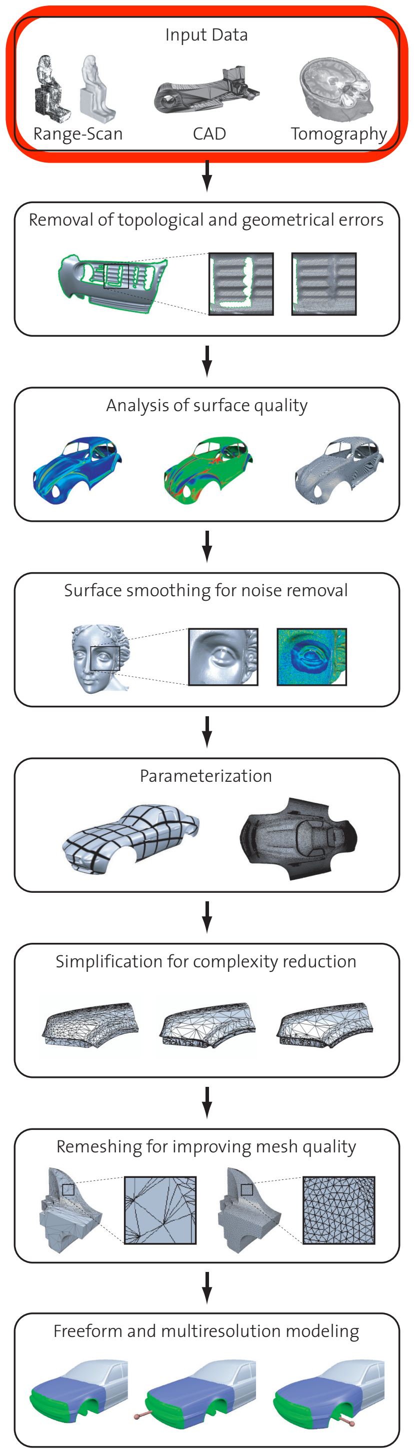
Overview

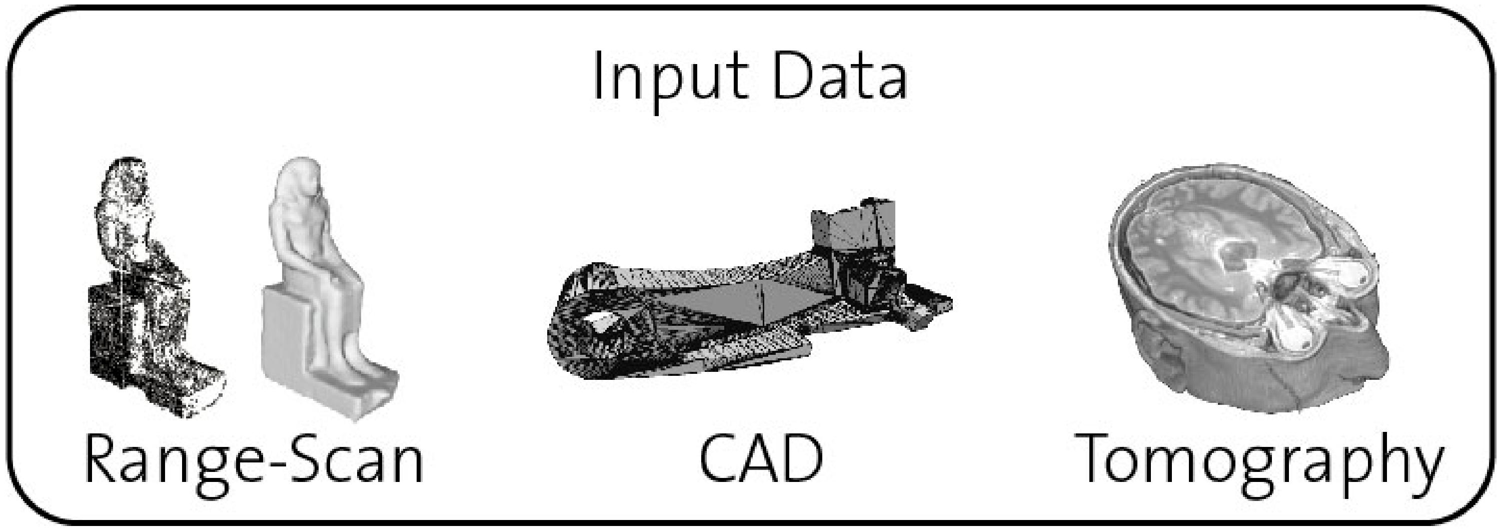
- Present the geometry processing pipeline based on triangle meshes
  - Fundamental concepts & recent developments
  - Show interesting connection between topics
  - Find more details in the course notes
- Provide source code for several examples
  - http://graphics.ethz.ch/~mbotsch
  - Linux, Mac, Windows

Main Questions

- Why are triangle meshes a suitable representation for geometry processing?
- What are the central processing algorithms?
- How can they be implemented efficiently?

Processing Pipeline

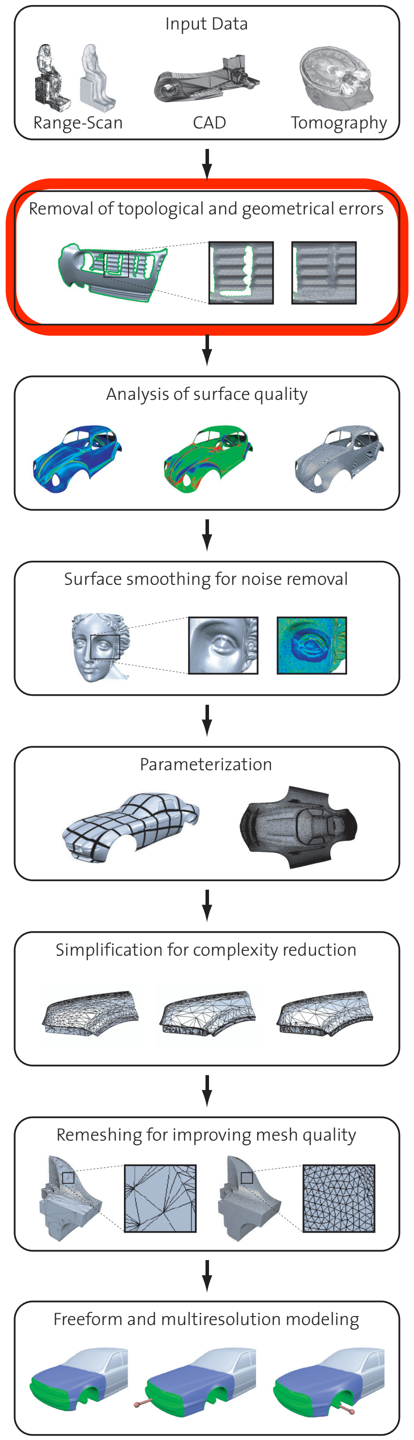


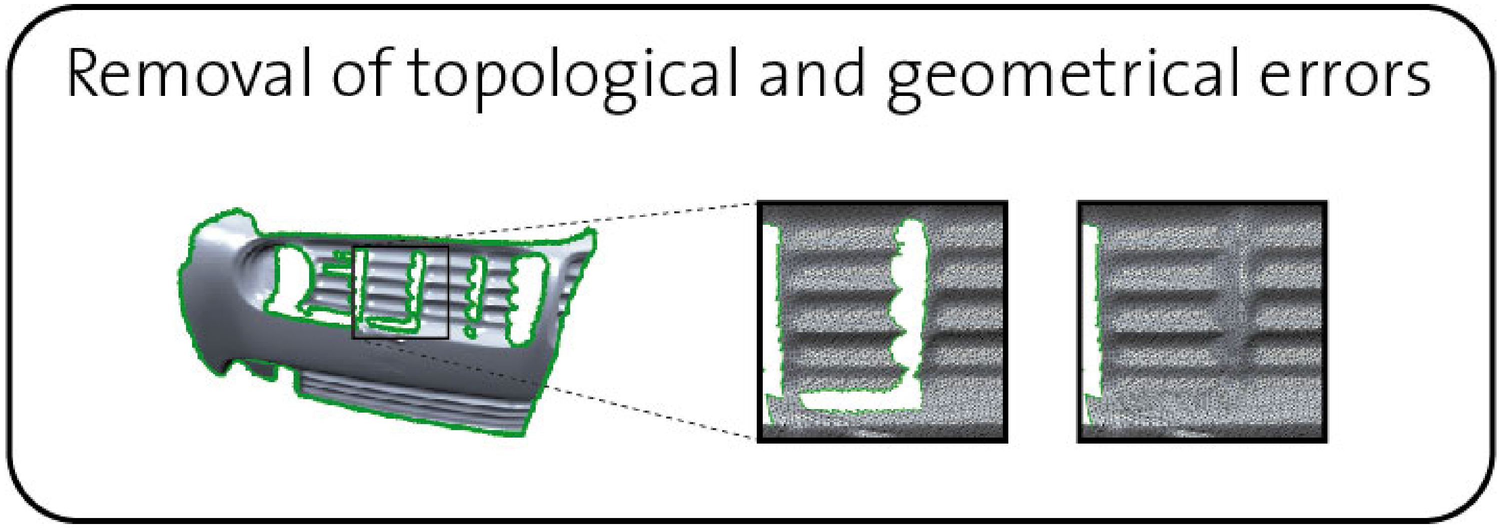


Surface Representations (9:10-9:50)

Mark Pauly

Processing Pipeline

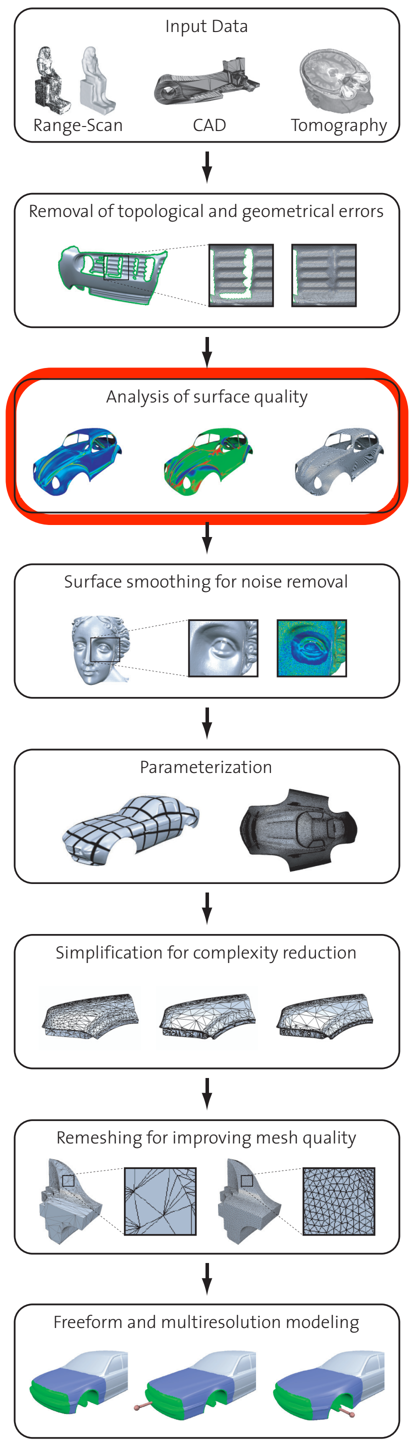


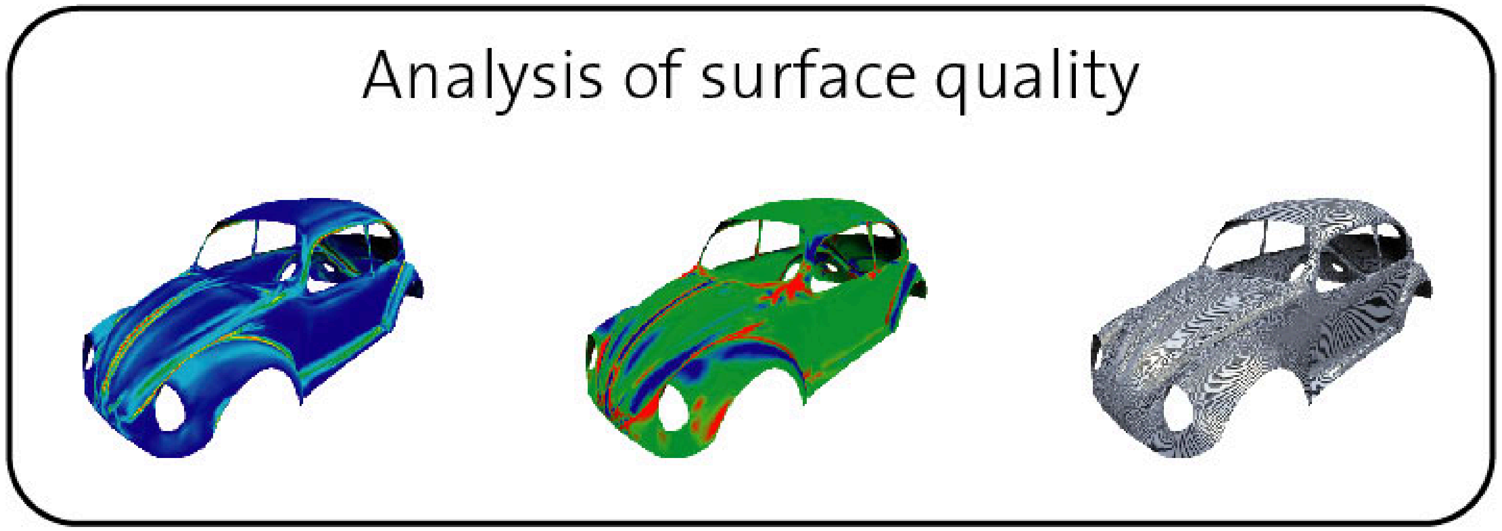


Mesh Repair (9:50-10:30)

Stephan Bischoff

Processing Pipeline

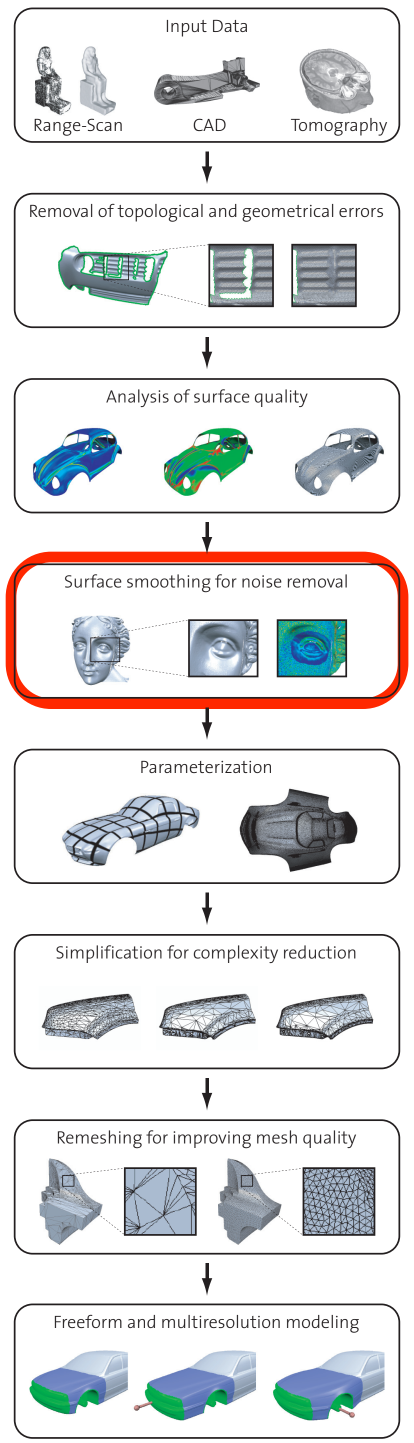


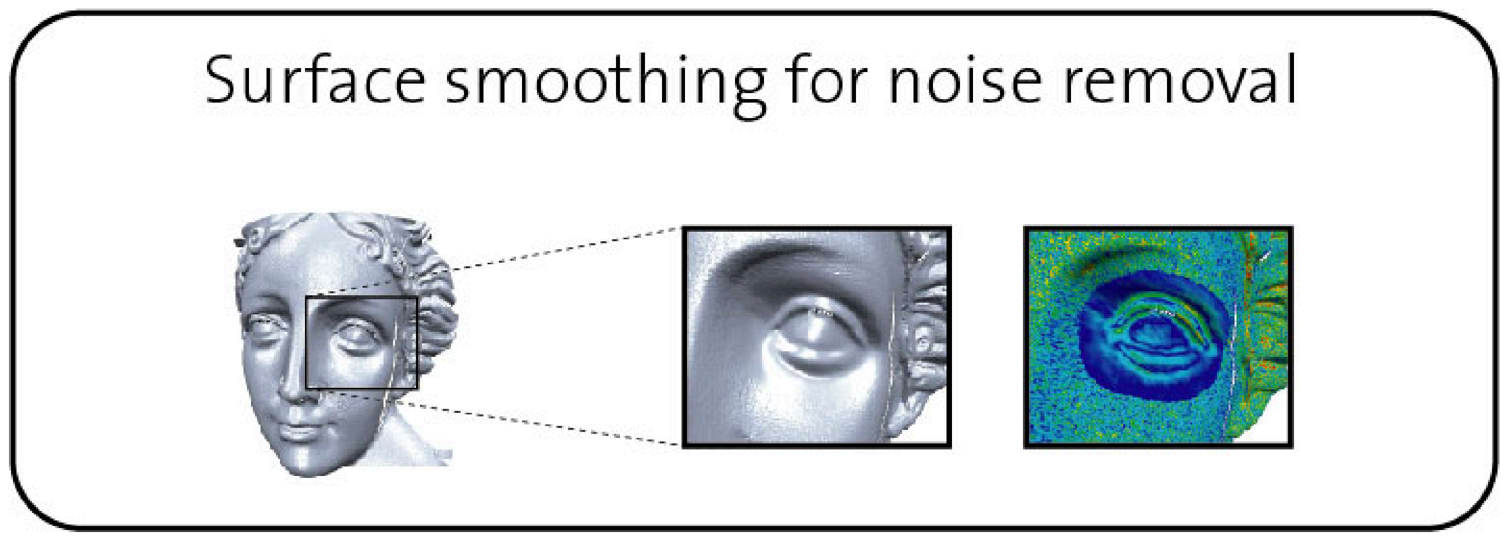


Mesh Quality (11:00-11:30)

Mark Pauly

Processing Pipeline

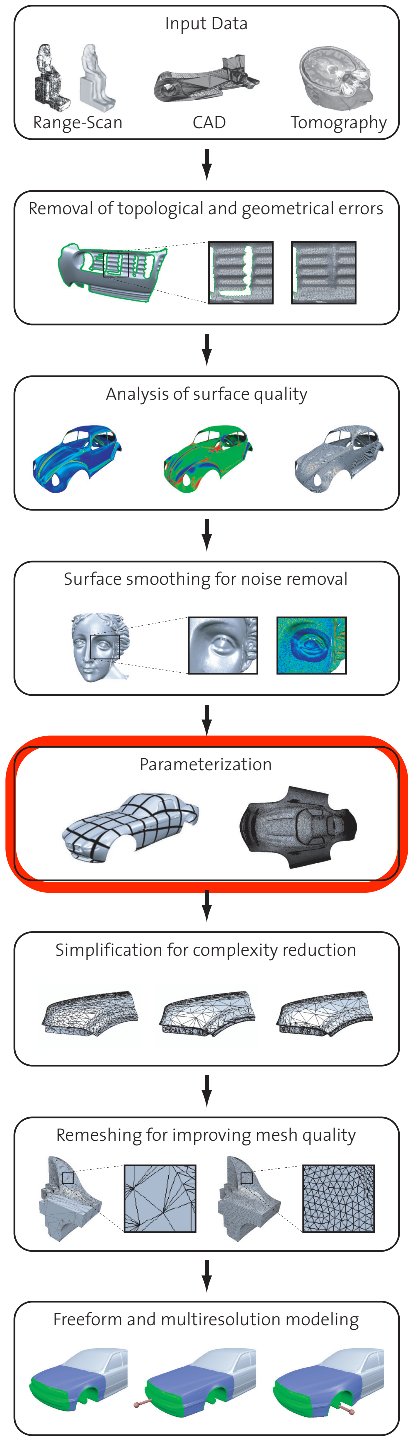


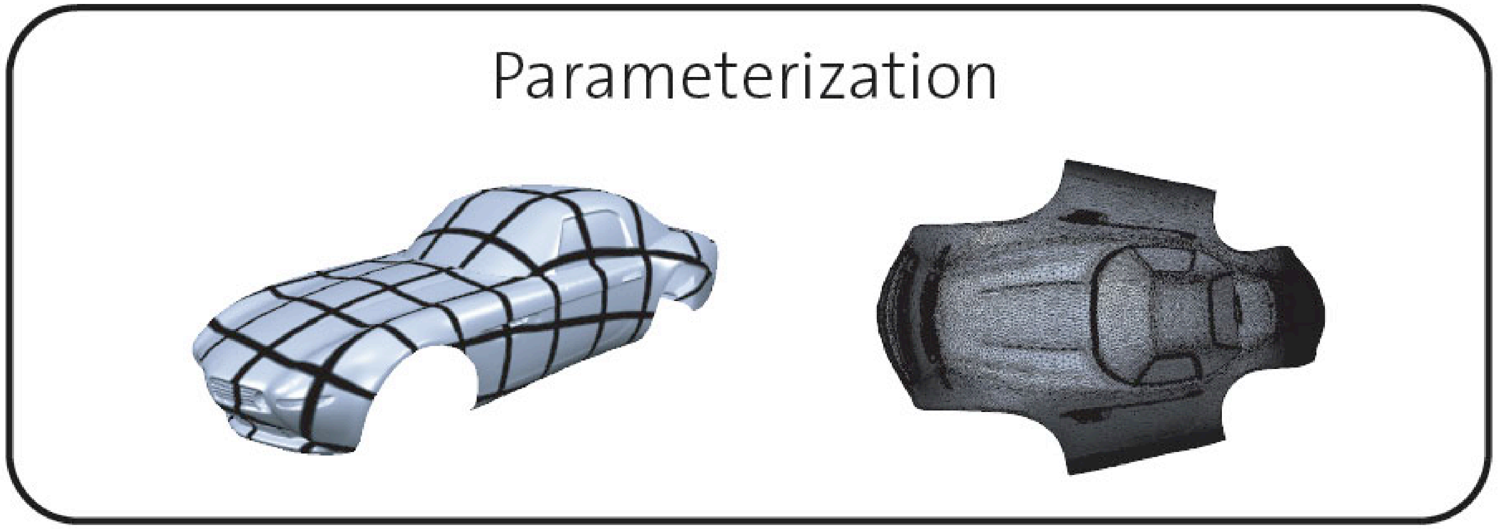


Mesh Smoothing (11:30-12:00)

Christian Rössl

Processing Pipeline

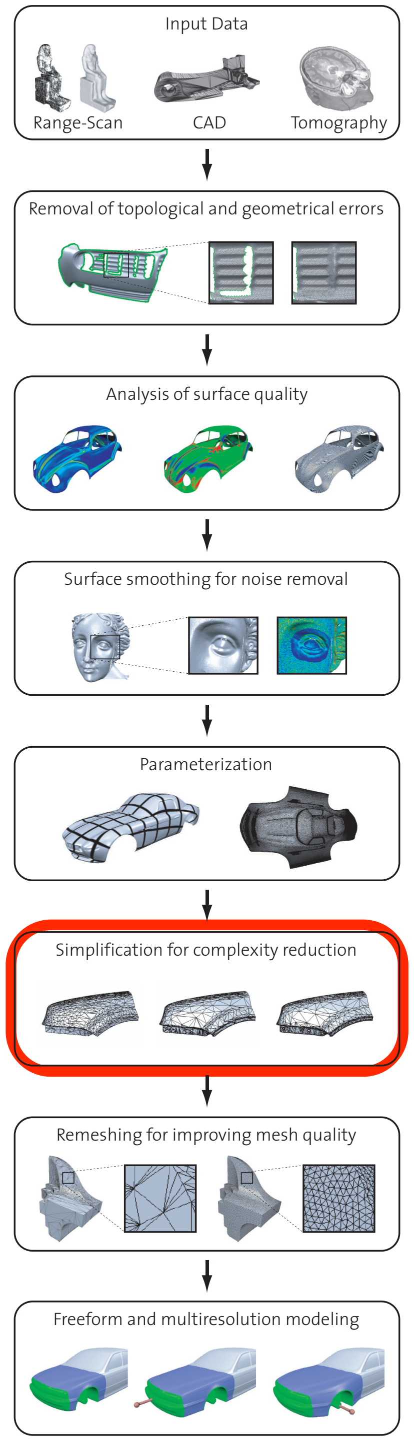


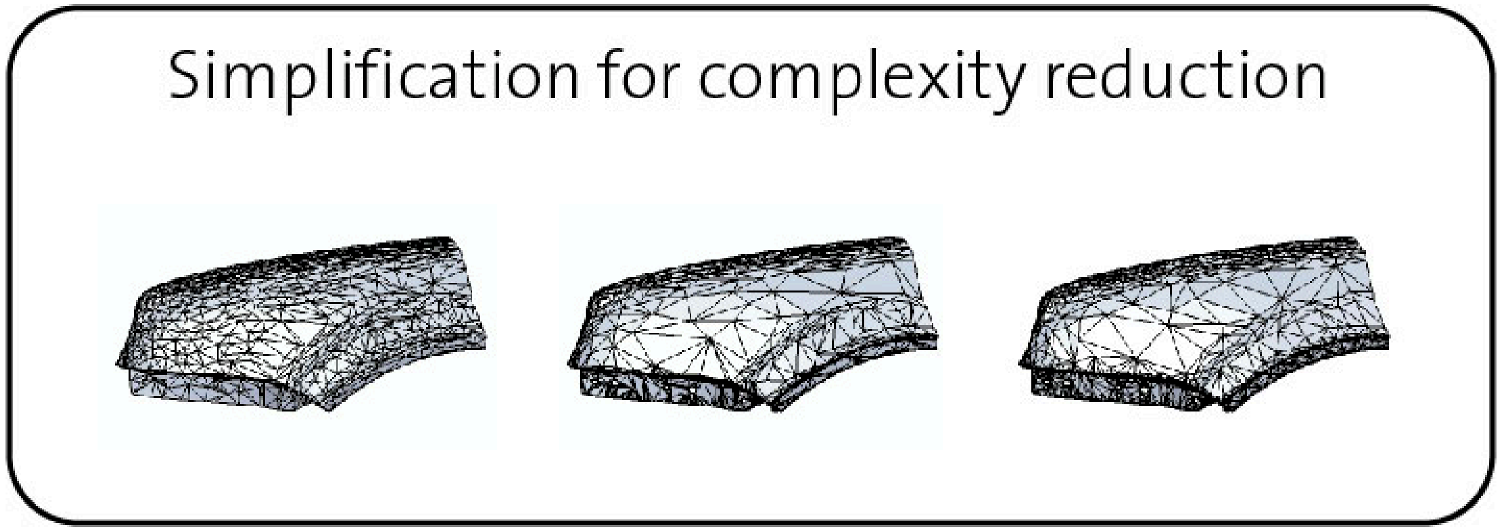


Mesh Parametrization (12:00-12:30)

Christian Rössl

Processing Pipeline

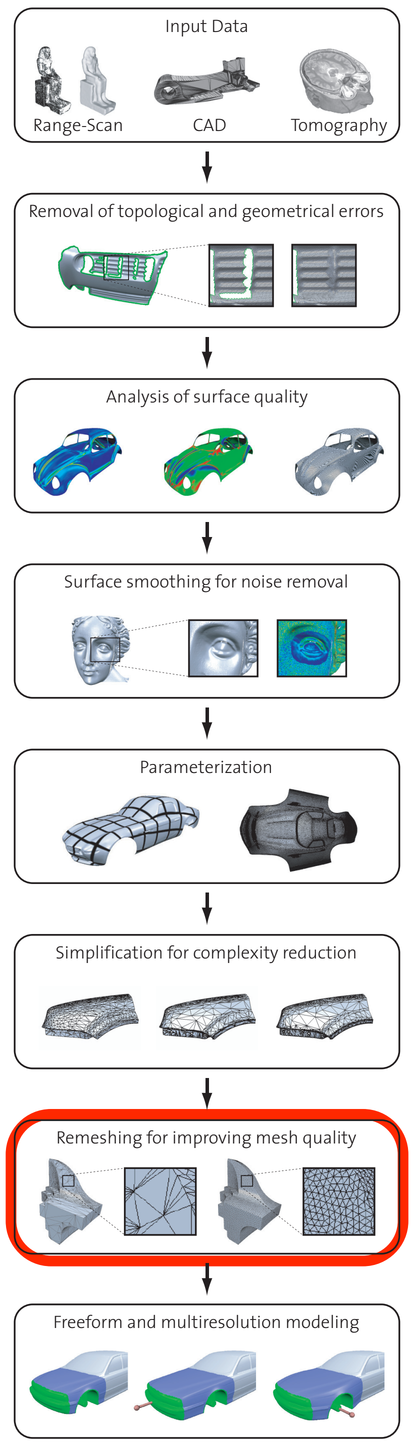


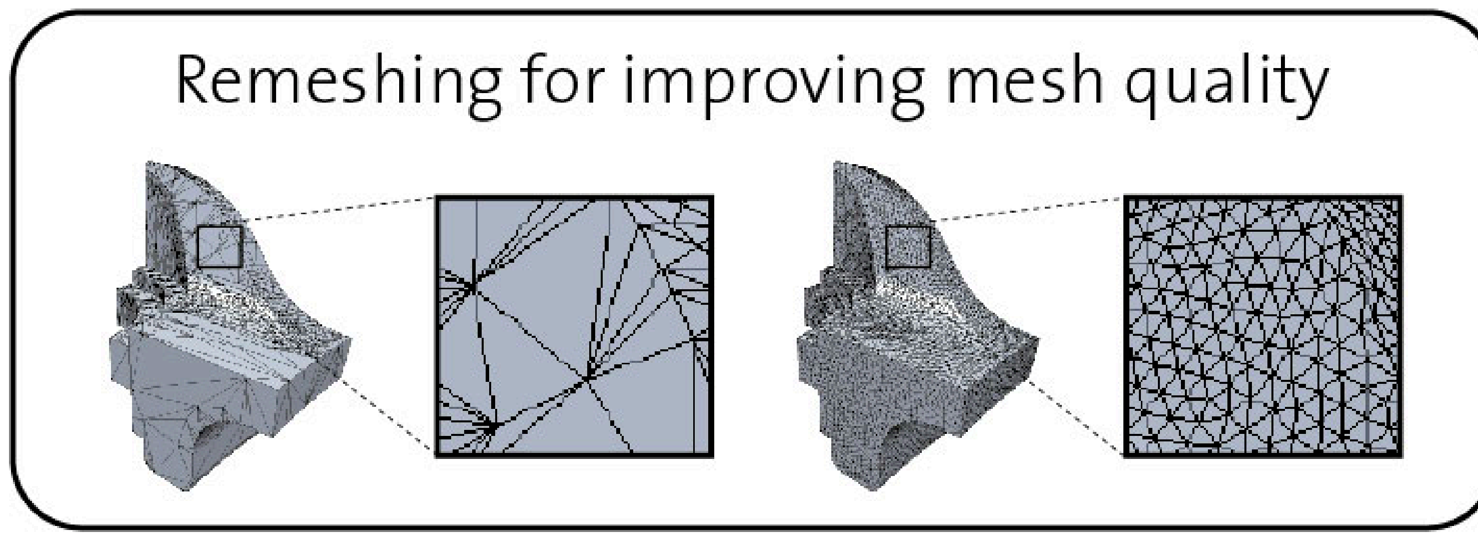


Mesh Decimation (14:00-14:40)

Leif Kobbelt

Processing Pipeline

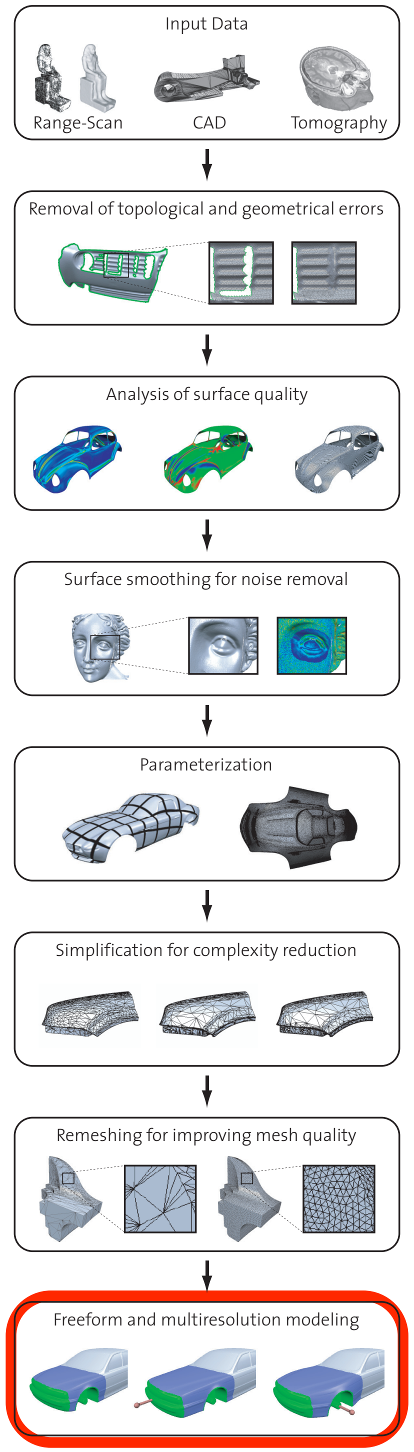


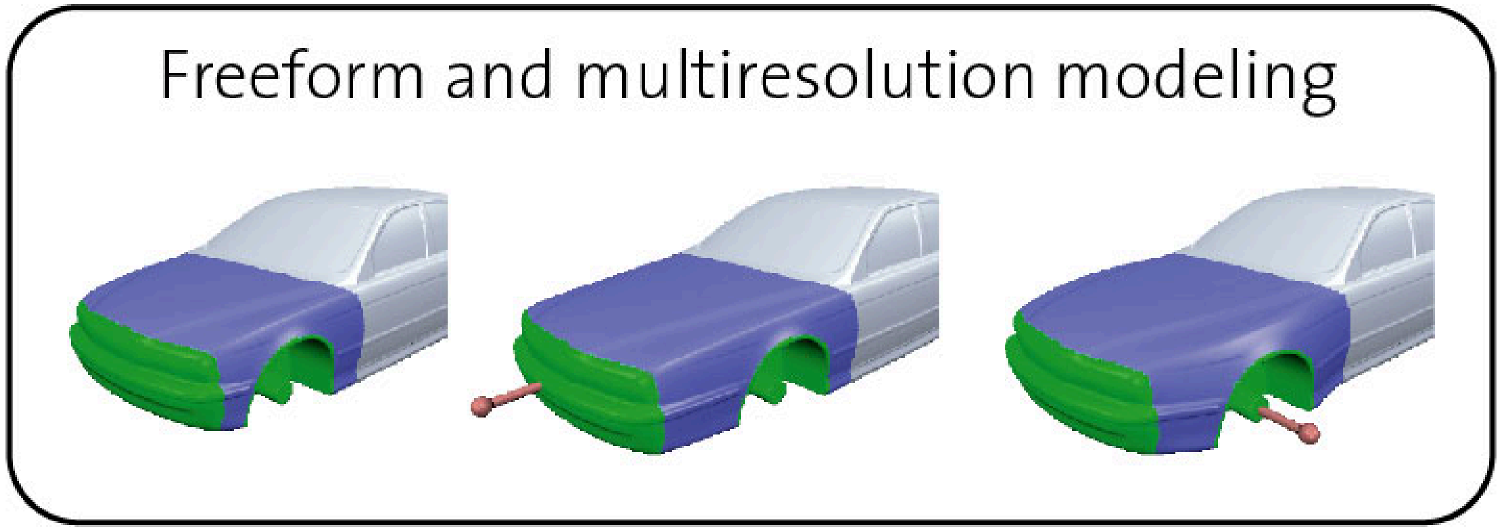


Remeshing (14:40-15:30)

Leif Kobbelt

Processing Pipeline

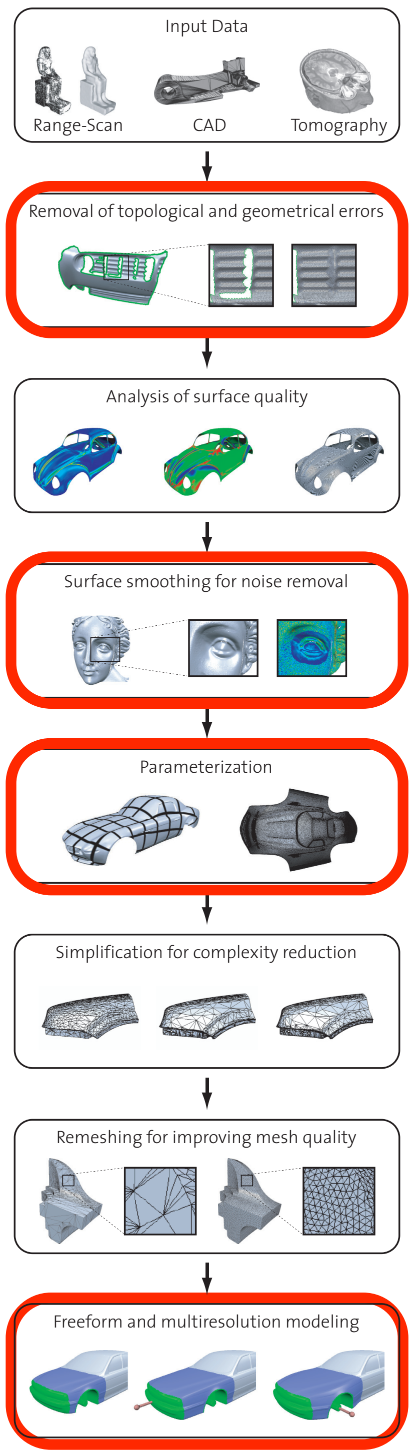


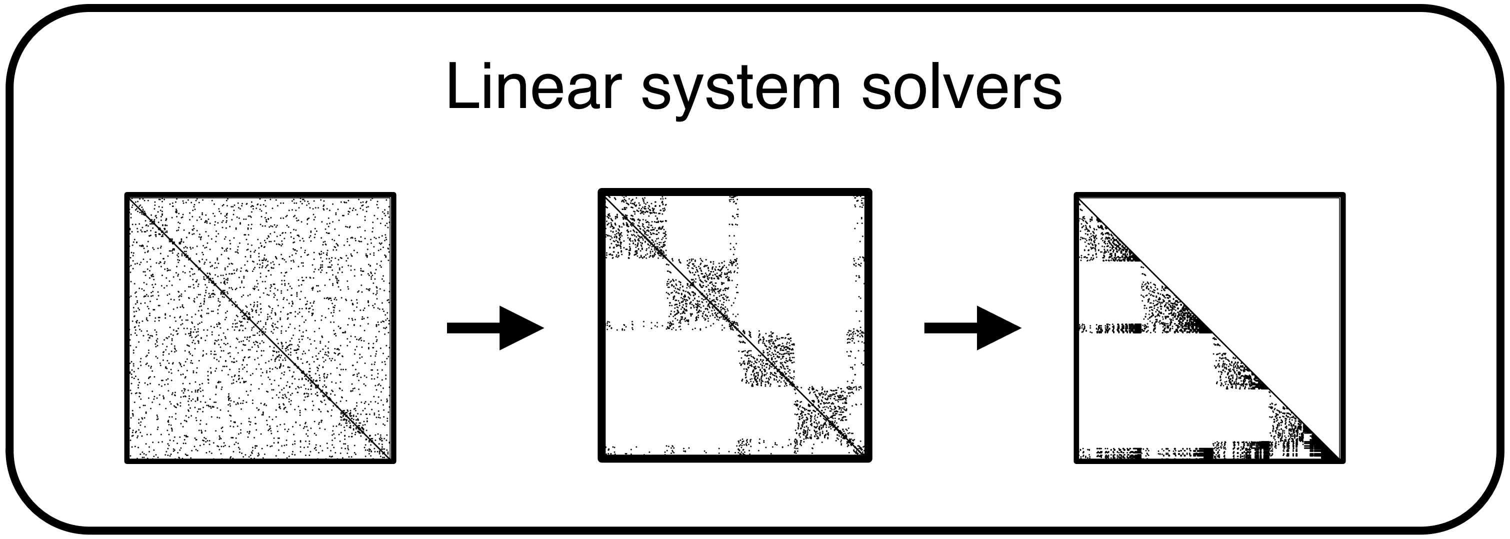


Mesh Editing (16:00-16:45)

Mario Botsch

Processing Pipeline

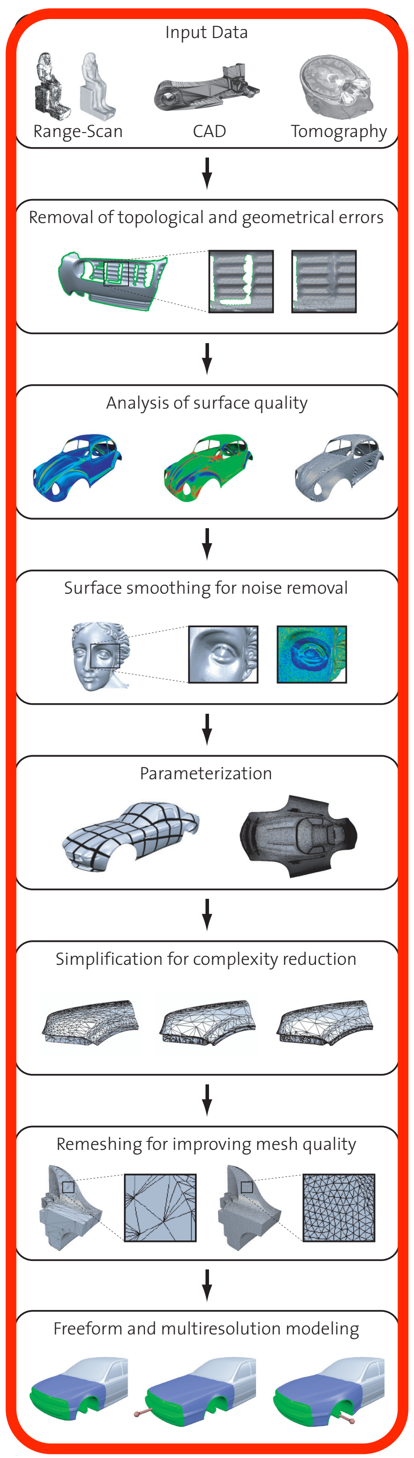


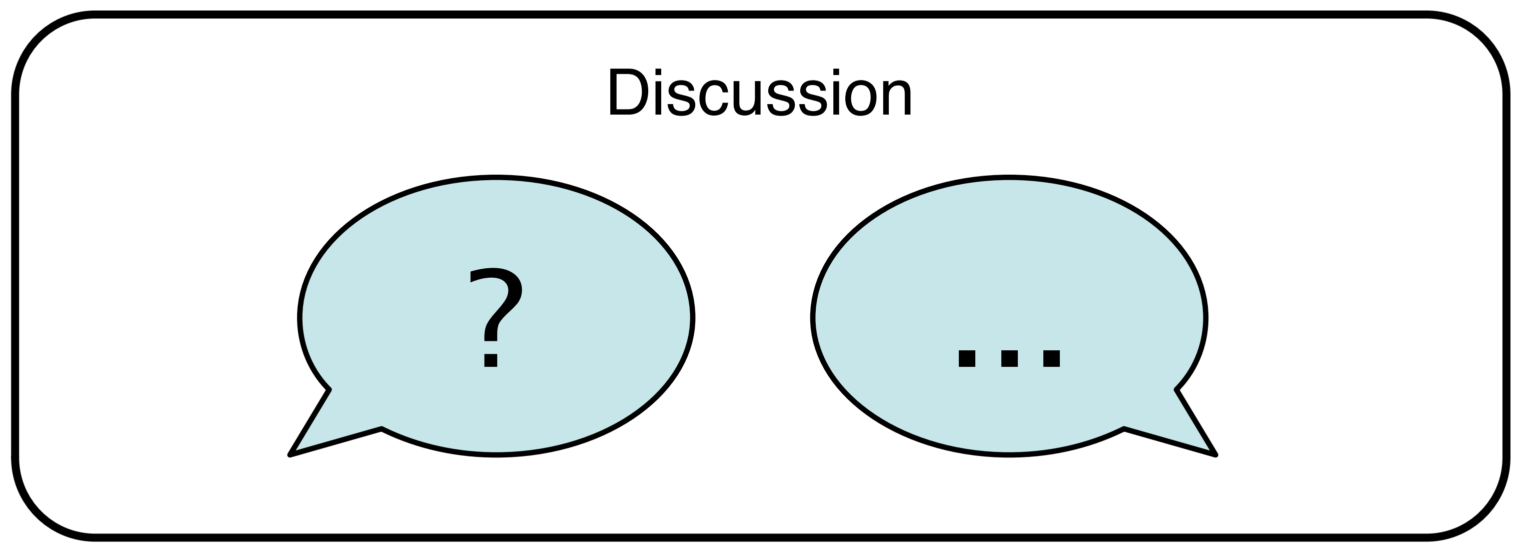


Numerics (16:45-17:15)

Mario Botsch

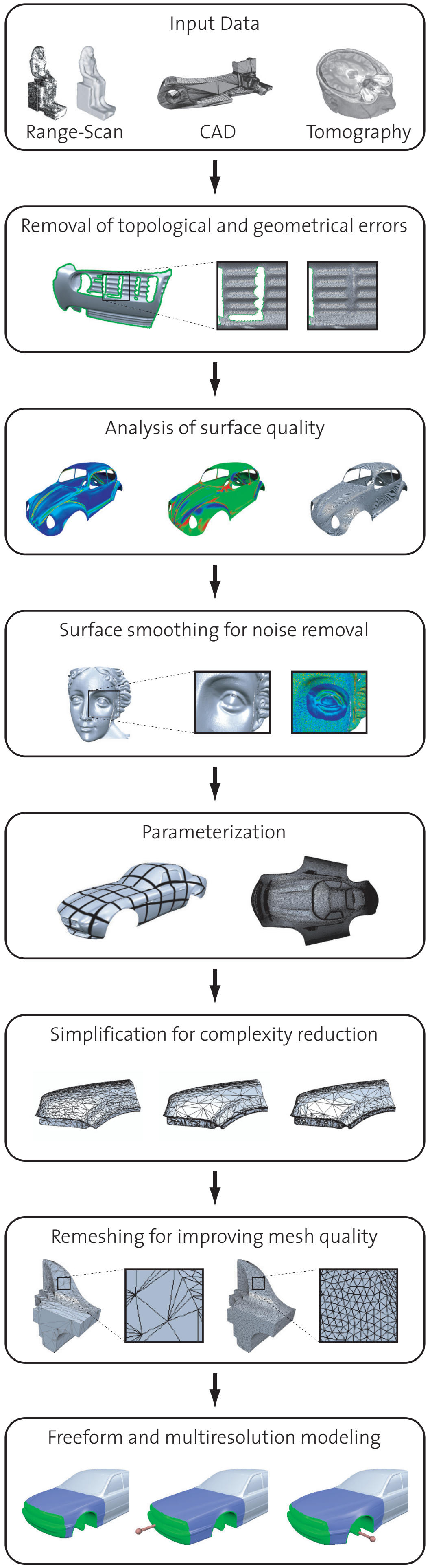
Processing Pipeline





Q & A (17:15-17:30)

All speakers



ETH
APPLIED
GEOMETRY
GROUP

Surface Representations

Mark Pauly

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Outline

- (mathematical) geometry representations
  - parametric vs. implicit
- approximation properties
- types of operations
  - distance queries
  - evaluation
  - modification / deformation
- data structures

Outline

- (mathematical) geometry representations
  - parametric vs. implicit
- approximation properties
- types of operations
  - distance queries
  - evaluation
  - modification / deformation
- data structures

Mathematical Representations

- parametric
  - range of a function
  - surface patch

$$\mathbf{f} : R^2 \rightarrow R^3, \quad \mathcal{S}_\Omega = \mathbf{f}(\Omega)$$

- implicit
  - kernel of a function
  - level set

$$F : R^3 \rightarrow R, \quad \mathcal{S}_c = \{\mathbf{p} : F(\mathbf{p}) = c\}$$

2D-Example: Circle

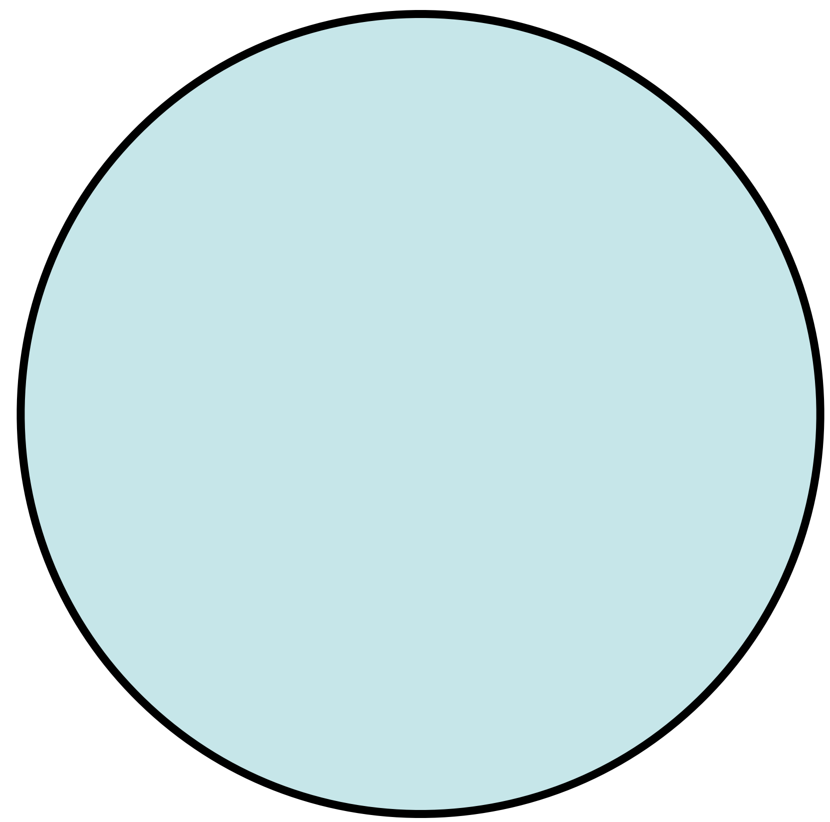
- parametric

$$\mathbf{f} : t \mapsto \left(\begin{array}{c} r\cos(t) \\ r\sin(t) \end{array} \right), \quad \mathcal{S} = \mathbf{f}([0, 2\pi])$$

- implicit

$$F(x, y) = x^2 + y^2 - r^2$$

$$\mathcal{S} = \{(x, y) : F(x, y) = 0\}$$



2D-Example: Island

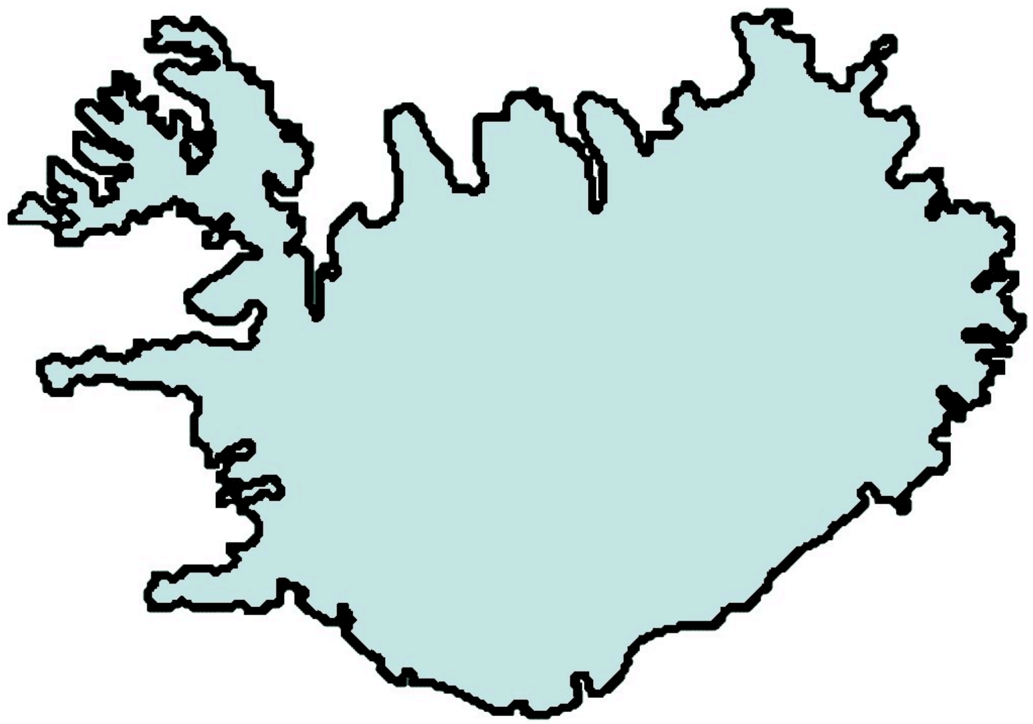
- parametric

$$\mathbf{f} : t \mapsto \begin{pmatrix} ??? \\ ??? \end{pmatrix}, \quad \mathcal{S} = \mathbf{f}([0, 2\pi])$$

- implicit

$$F(x, y) = \quad ???$$

$$\mathcal{S} = \{(x, y) : F(x, y) = 0\}$$



Approximation Quality

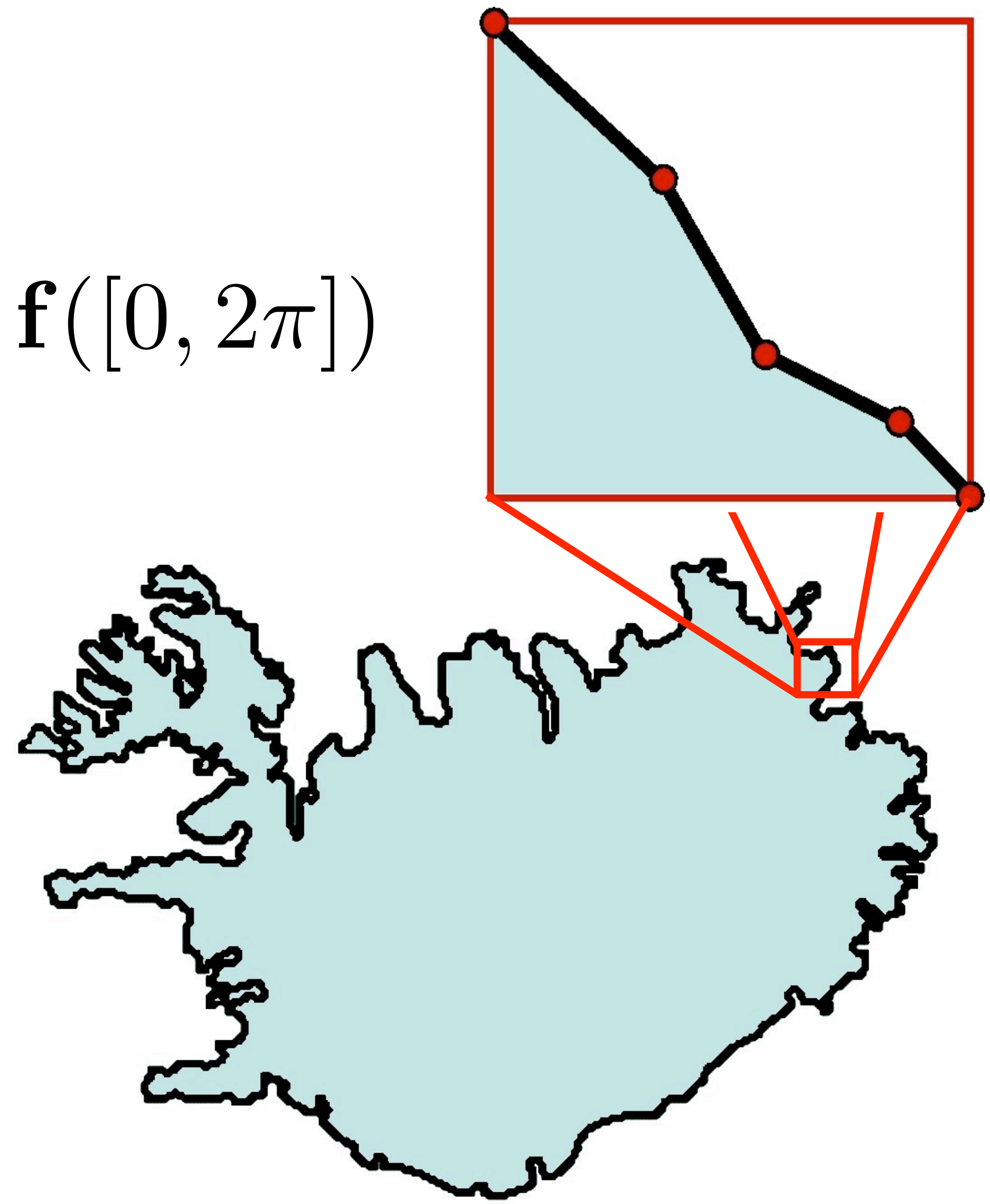
- ***piecewise*** parametric

$$\mathbf{f} : t \mapsto \begin{pmatrix} ??? \\ ??? \end{pmatrix}, \quad \mathcal{S} = \mathbf{f}([0, 2\pi])$$

- piecewise implicit

$$F(x, y) = \quad ???$$

$$\mathcal{S} = \{(x, y) : F(x, y) = 0\}$$



Approximation Quality

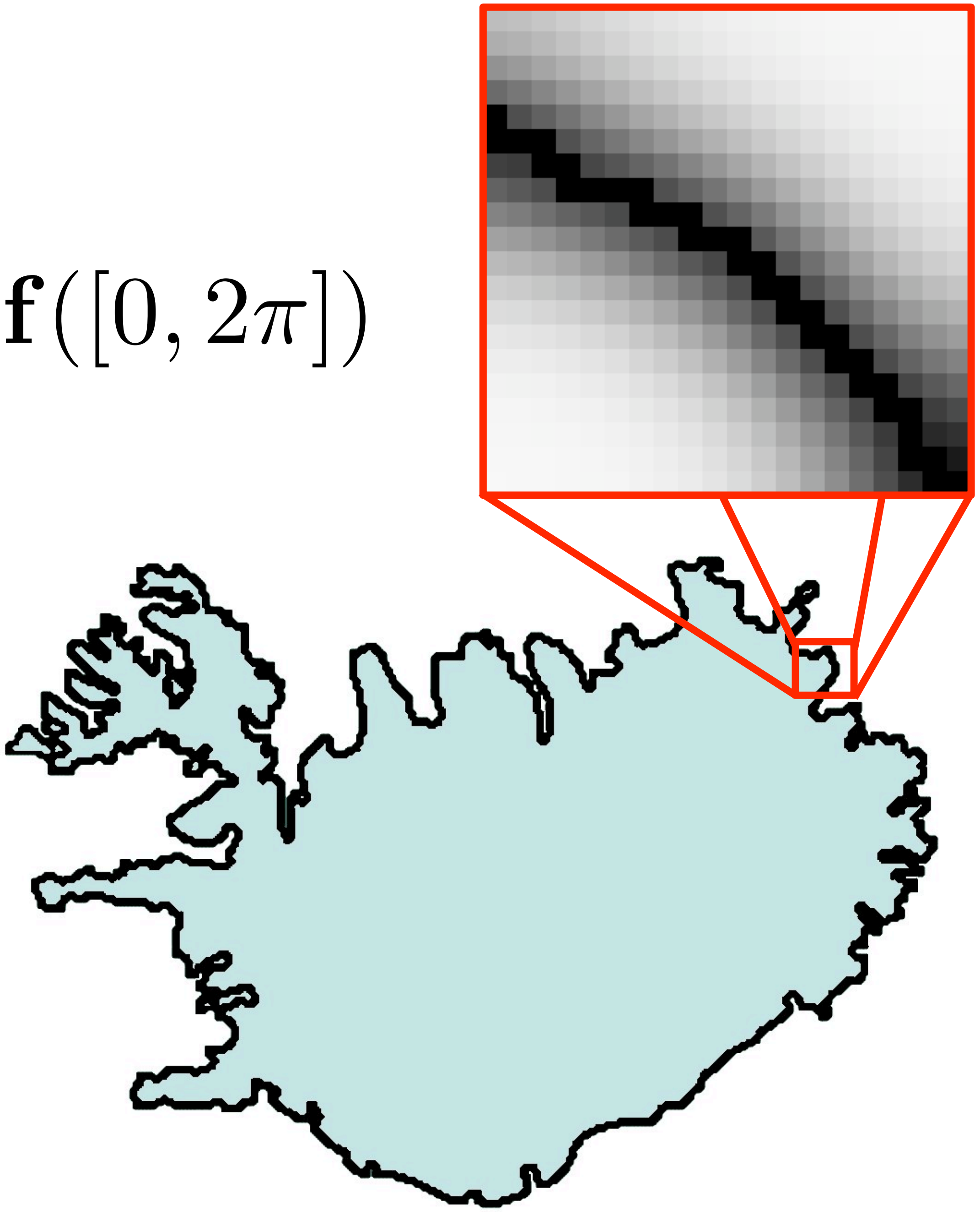
- piecewise parametric

$$\mathbf{f} : t \mapsto \begin{pmatrix} ??? \\ ??? \end{pmatrix}, \quad \mathcal{S} = \mathbf{f}([0, 2\pi])$$

- ***piecewise*** implicit

$$F(x, y) = \quad ???$$

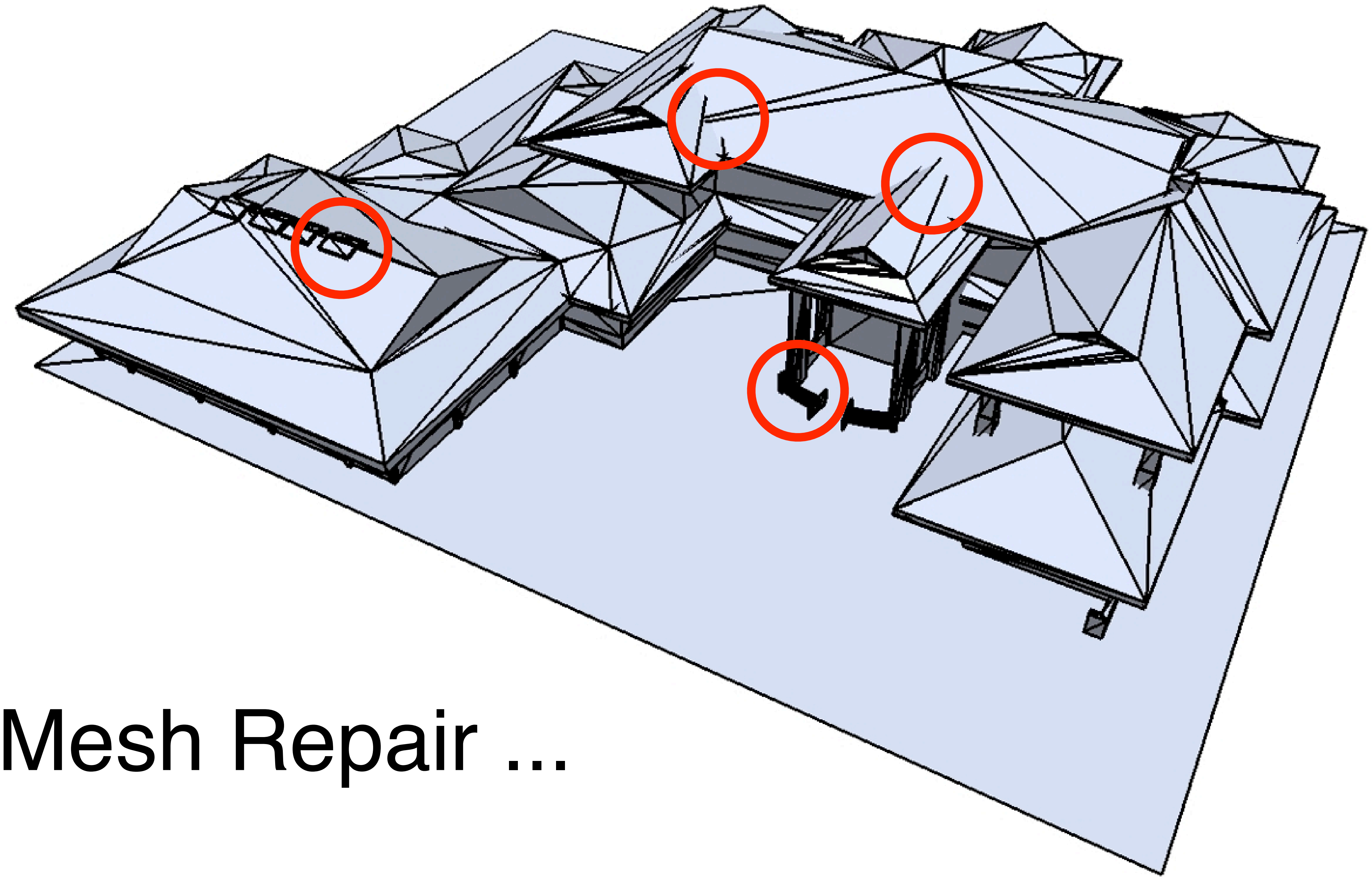
$$\mathcal{S} = \{(x, y) : F(x, y) = 0\}$$



Requirements / Properties

- continuity
  - interpolation / approximation $\mathbf{f}(u_i, v_i) \approx \mathbf{p}_i$
- topological consistency
  - manifold-ness
- smoothness
  - C^0, C^1, C^2, ... C^k
- fairness
  - curvature distribution

Topological Consistency



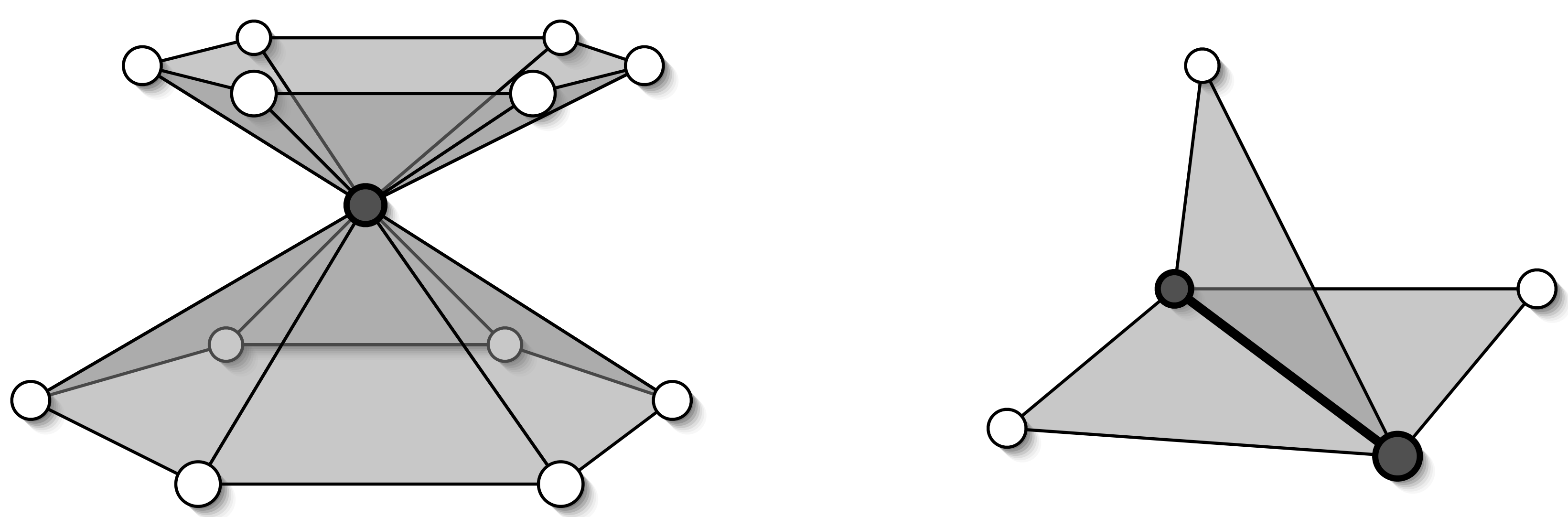
Mesh Repair ...

Closed 2-Manifolds

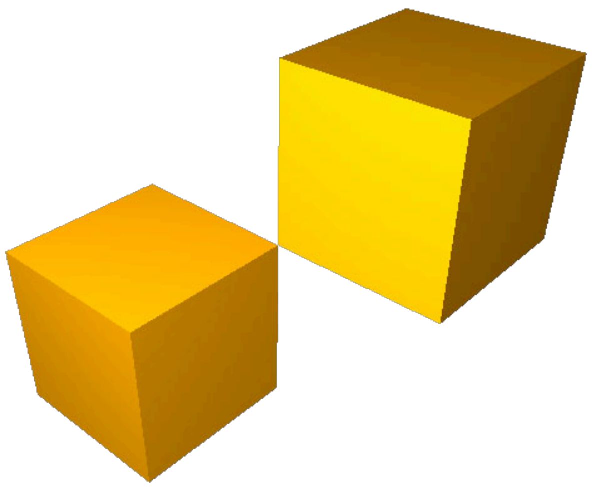
- parametric
  - disk-shaped neighborhoods
  - $\mathbf{f}(D_\varepsilon[u,v]) = D_\delta[\mathbf{f}(u,v)]$
- implicit
  - surface of a "physical" solid
  - $F(x,y,z) = c, \quad \|\nabla F(x,y,z)\| \neq 0$

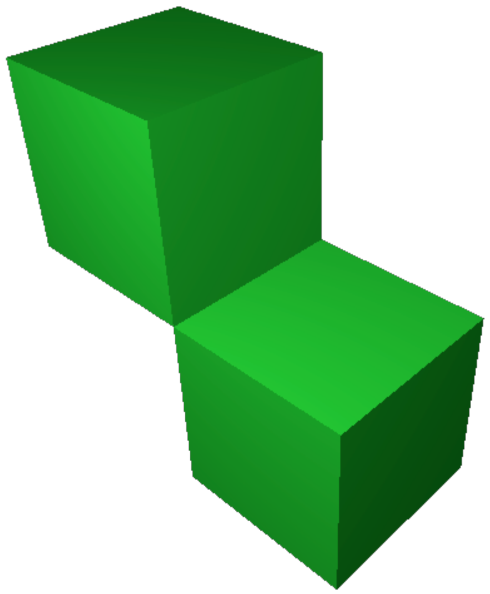
Closed 2-Manifolds

- parametric
  - disk-shaped neighborhoods
  - $\mathbf{f}(D_\varepsilon[u,v]) = D_\delta[\mathbf{f}(u,v)]$



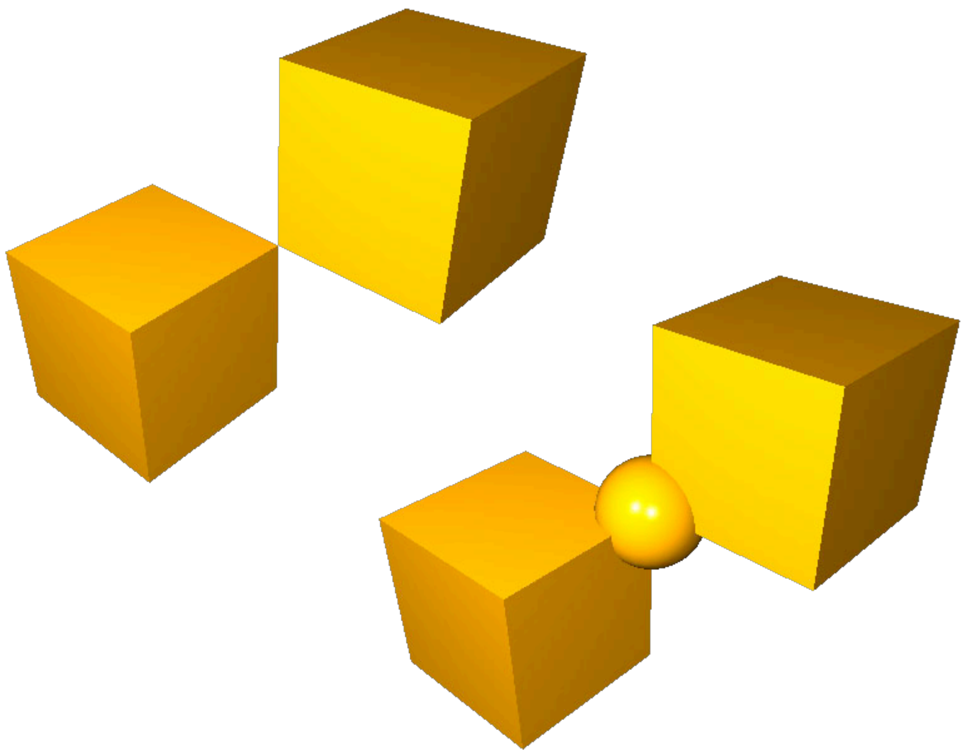
Closed 2-Manifolds

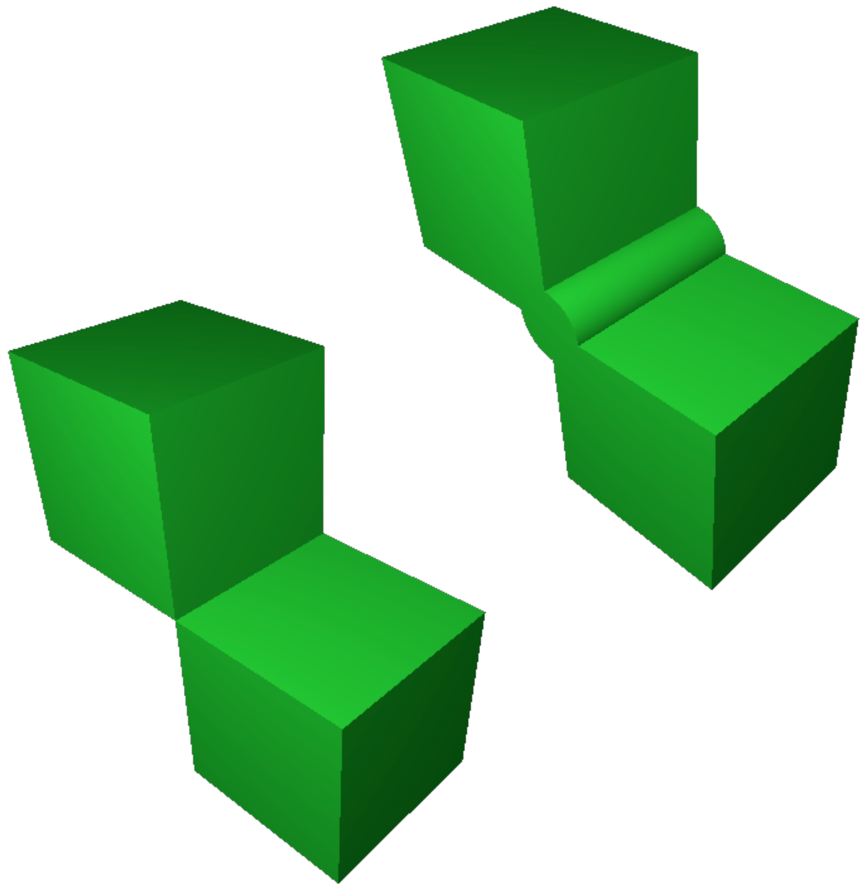




- implicit
  - surface of a “physical” solid
  - $F(x, y, z) = c, \quad \|\nabla F(x, y, z)\| \neq 0$

Closed 2-Manifolds

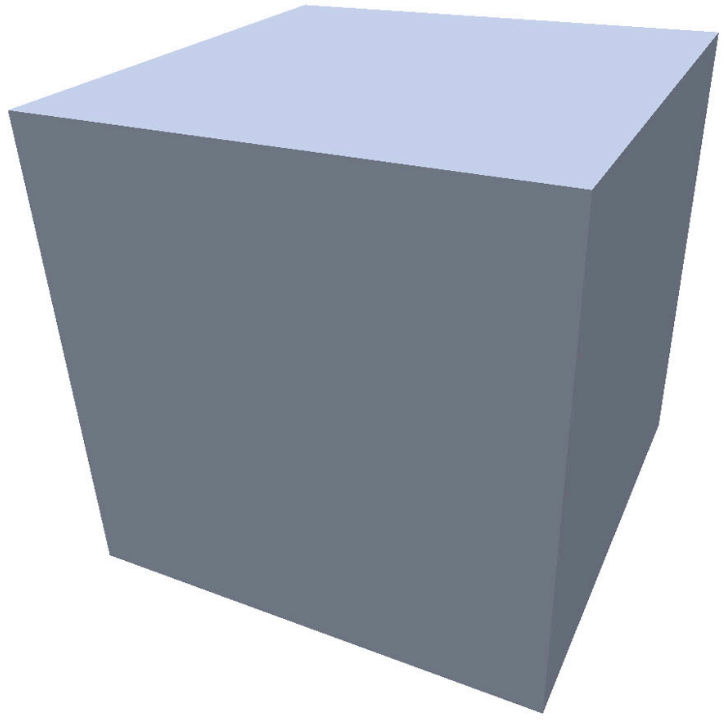


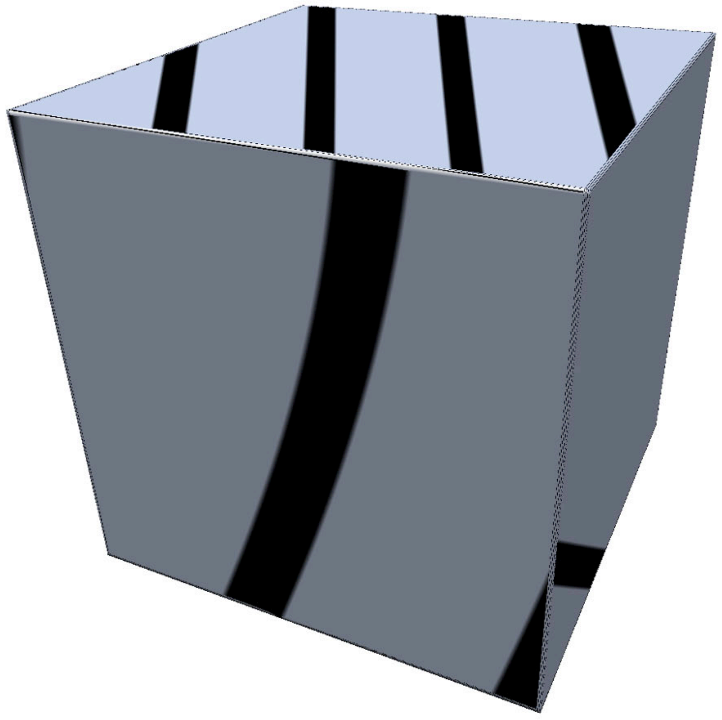


- implicit
  - surface of a “physical” solid
  - $F(x, y, z) = c, \quad \|\nabla F(x, y, z)\| \neq 0$

Smoothness

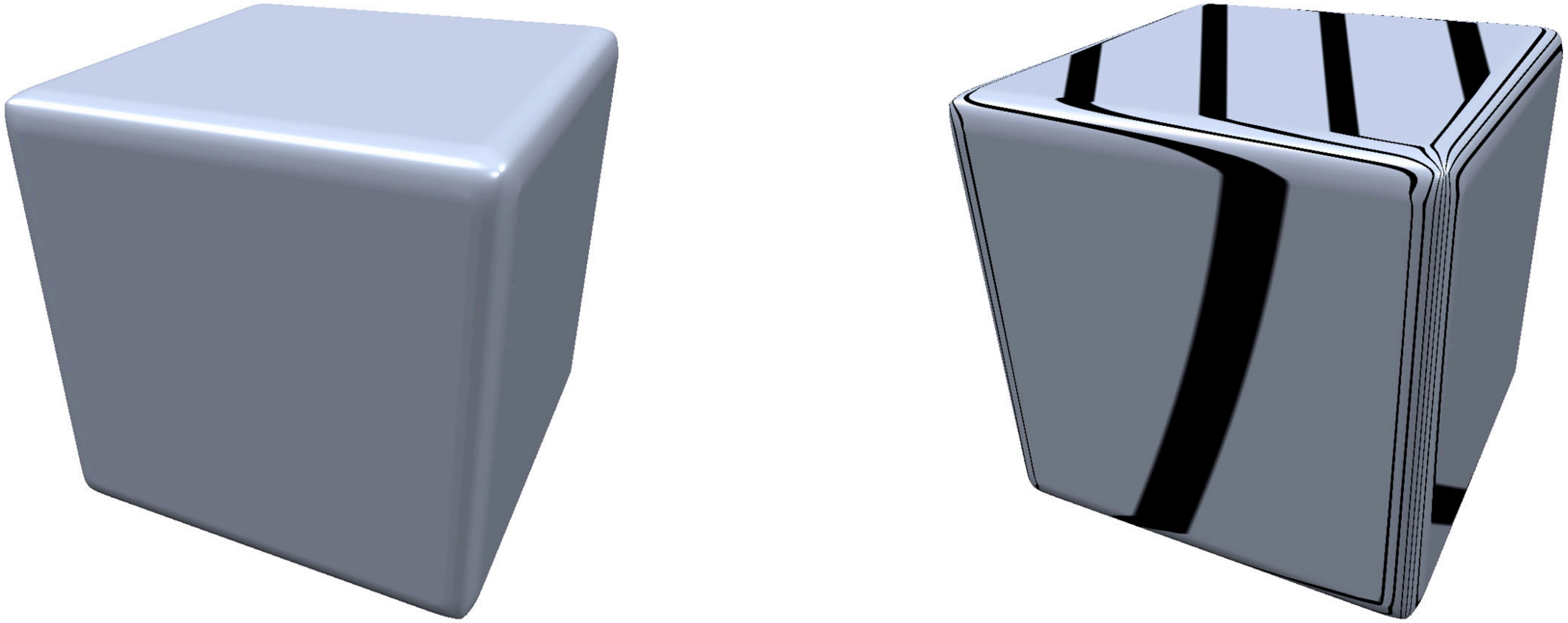
- position continuity : C^0
- tangent continuity : C^1
- curvature continuity : C^2





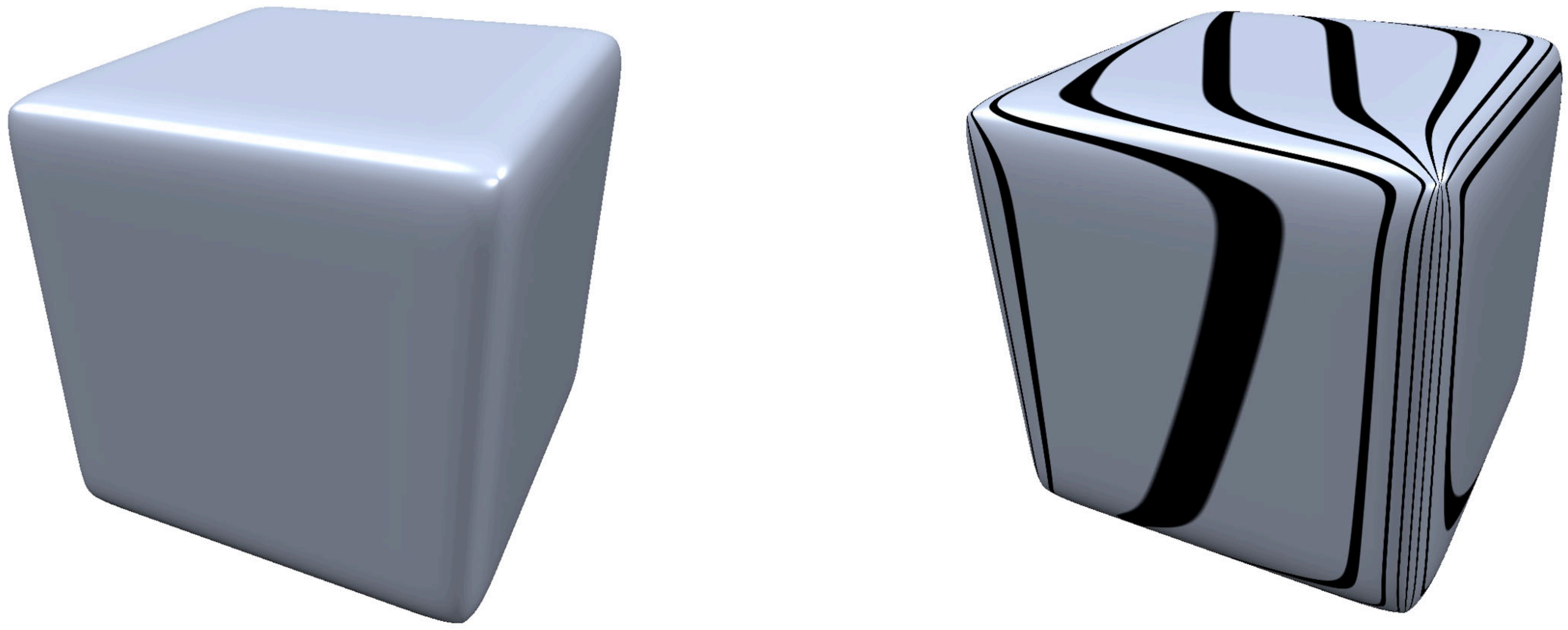
Smoothness

- position continuity : C^0
- tangent continuity : C^1
- curvature continuity : C^2



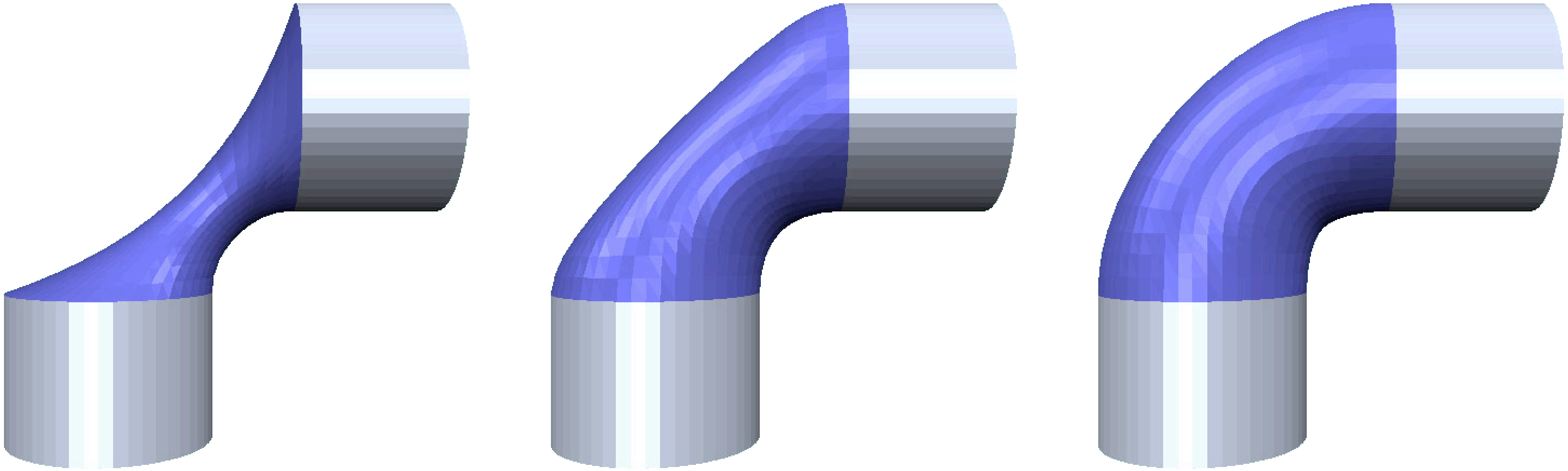
Smoothness

- position continuity : C^0
- tangent continuity : C^1
- curvature continuity : C^2



Fairness

- minimum surface area
- minimum curvature
- minimum curvature variation



Outline

- (mathematical) geometry representations
  - parametric vs. implicit
- approximation properties
- types of operations
  - distance queries
  - evaluation
  - modification / deformation
- data structures

Polynomials

- computable functions

$$\mathbf{p}(t) \;=\; \sum_{i=0}^{p} \mathbf{c}_i \, t^i \;=\; \sum_{i=0}^{p} \mathbf{c}_i' \, \Phi_i(t)$$

- Taylor expansion

$$\mathbf{f}(h) \;=\; \sum_{i=0}^{p} \frac{1}{i!} \, \mathbf{f}^{(i)}(0) \, h^i \;+\; O(h^{p+1})$$

- interpolation error (mean value theorem)

$$\mathbf{p}(t_i) = \mathbf{f}(t_i), \quad t_i \in [0, h]$$

$$\|\mathbf{f}(t) - \mathbf{p}(t)\| \;=\; \frac{1}{(p+1)!} \, \mathbf{f}^{(p+1)}(t^*) \, \prod_{i=0}^{p} (t - t_i) \;=\; O(h^{(p+1)})$$

Polynomials

- computable functions

$$\mathbf{p}(t) = \sum_{i=0}^{p} \mathbf{c}_i \, t^i = \sum_{i=0}^{p} \mathbf{c}'_i \, \Phi_i(t)$$

- Taylor expansion

$$\mathbf{f}(h) = \sum_{i=0}^{p} \frac{1}{i!} \mathbf{f}^{(i)}(0) \, h^i + O(h^{p+1})$$

- interpolation error (mean value theorem)

$$\mathbf{p}(t_i) = \mathbf{f}(t_i), \quad t_i \in [0, h]$$

$$\|\mathbf{f}(t) - \mathbf{p}(t)\| = \frac{1}{(p+1)!} \mathbf{f}^{(p+1)}(t^*) \prod_{i=0}^{p} (t - t_i) = O(h^{(p+1)})$$

Polynomials

- computable functions

$$\mathbf{p}(t) = \sum_{i=0}^{p} \mathbf{c}_i \, t^i = \sum_{i=0}^{p} \mathbf{c}'_i \, \Phi_i(t)$$

- Taylor expansion

$$\mathbf{f}(h) = \sum_{i=0}^{p} \frac{1}{i!} \mathbf{f}^{(i)}(0) \, h^i + O(h^{p+1})$$

- interpolation error (mean value theorem)

$$\mathbf{p}(t_i) = \mathbf{f}(t_i), \quad t_i \in [0, h]$$

$$\|\mathbf{f}(t) - \mathbf{p}(t)\| = \frac{1}{(p+1)!} \mathbf{f}^{(p+1)}(t^*) \prod_{i=0}^{p} (t - t_i) = O(h^{(p+1)})$$

Implicit Polynomials

- interpolation error of the function values

$$\|F(x,y,z) - P(x,y,z)\| \;=\; O(h^{(p+1)})$$

- approximation error of the contour

$$\triangle\mathbf{p} = \lambda\,\nabla F(\mathbf{p}) \qquad \frac{F(\mathbf{p}+\triangle\mathbf{p}) - F(\mathbf{p})}{\|\triangle\mathbf{p}\|} \;\approx\; \|\nabla F(\mathbf{p})\|$$

Implicit Polynomials

- interpolation error of the function values

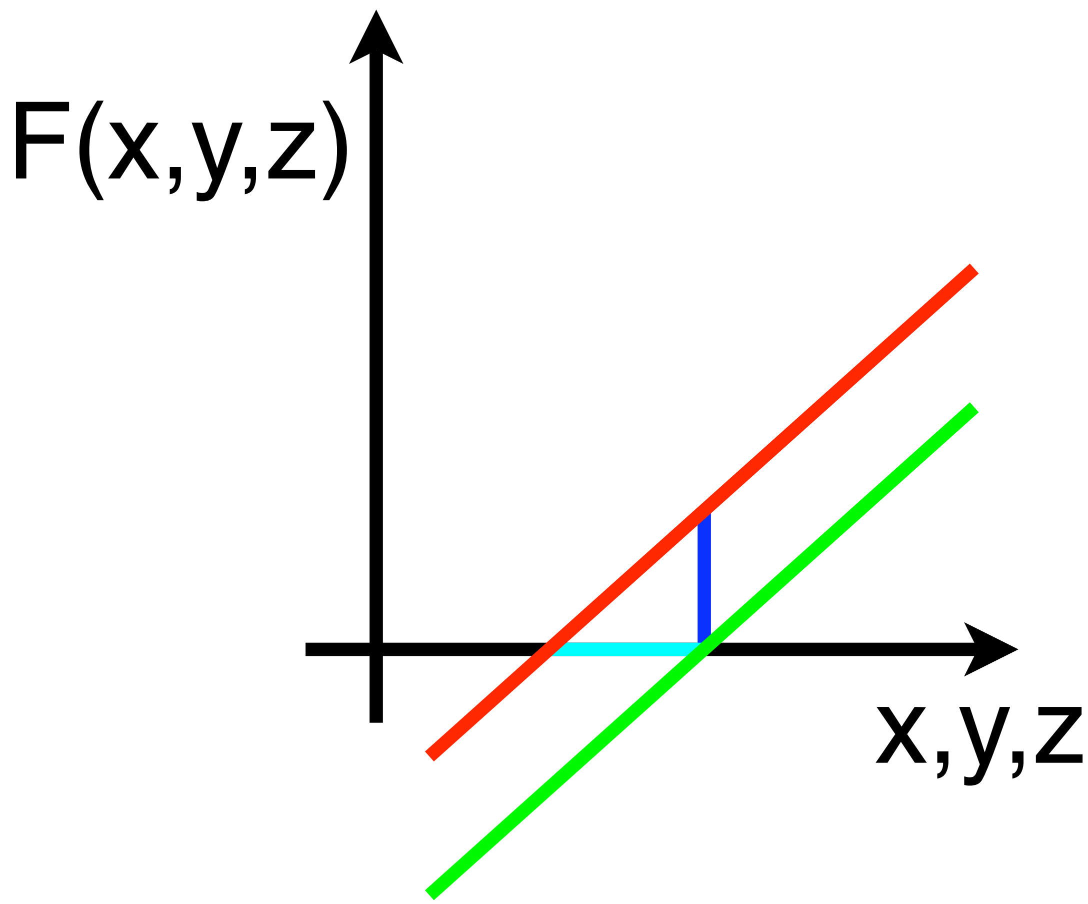
$$\|F(x,y,z) - P(x,y,z)\| \;=\; O(h^{(p+1)})$$

- approximation error of the contour

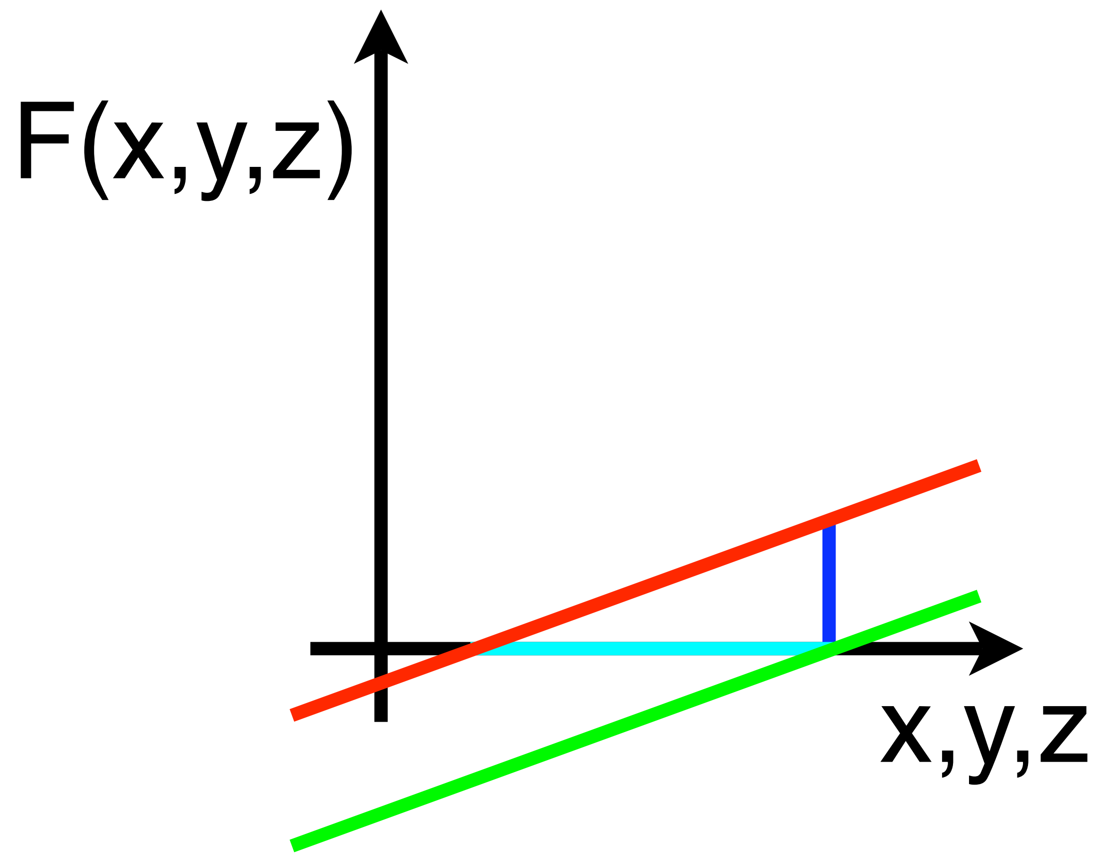
$$\triangle\mathbf{p} = \lambda\,\nabla F(\mathbf{p}) \qquad \|\triangle\mathbf{p}\| \;\approx\; \frac{F(\mathbf{p}+\triangle\mathbf{p}) - F(\mathbf{p})}{\|\nabla F(\mathbf{p})\|}$$

(gradient bounded from below)

Implicit Polynomials



large
gradient



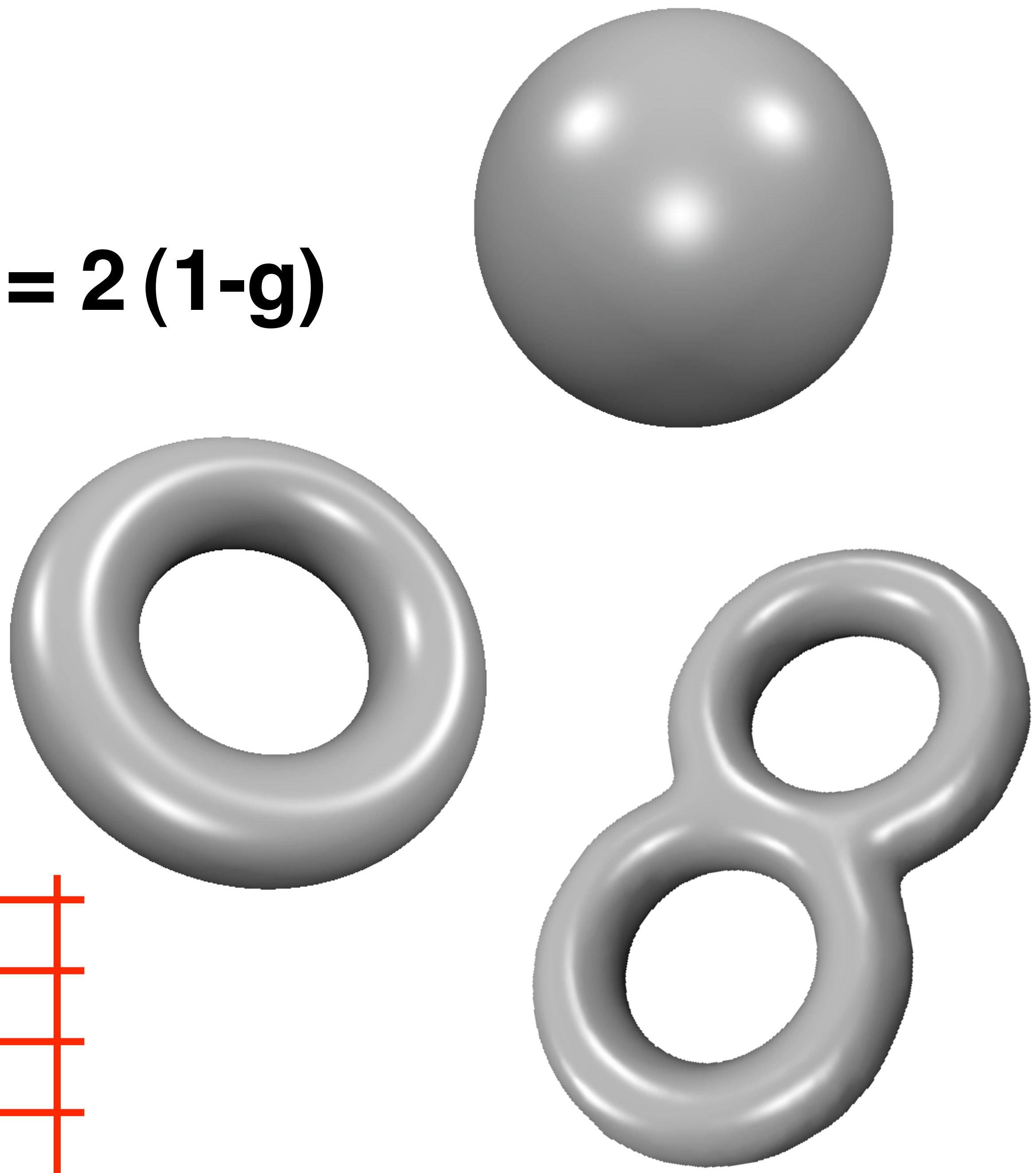
small
gradient

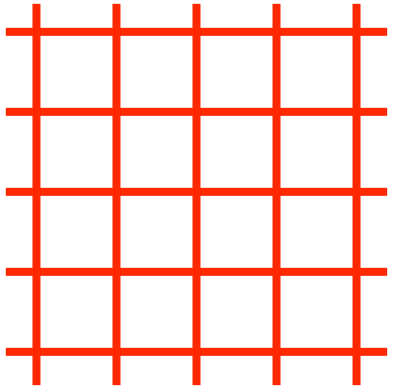
Polynomial Approximation

- approximation error is $\mathbf{O(h^{p+1})}$
- improve approximation quality by
  - increasing **p** ... higher order polynomials
  - decreasing **h** ... smaller / more segments
- issues
  - smoothness of the target data ($\max_t f^{(p+1)}(t)$)
  - smoothness conditions between segments

Regularity

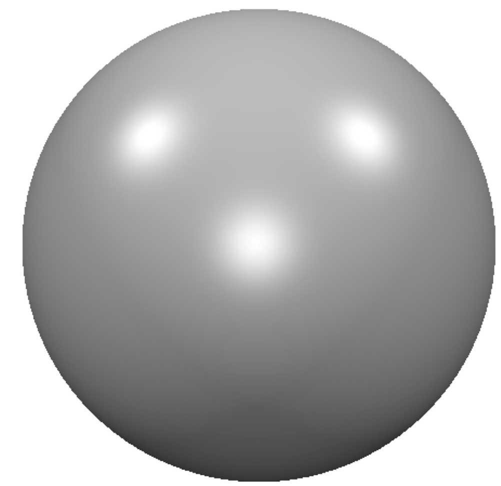
- parametric
  - patches vs. polygons
  - Euler formula: $\mathbf{V - E + F = 2\,(1\text{-}g)}$
  - **quad** meshes
    - $F \approx V$
    - $E \approx 2V$
    - average valence = 4
  - quasi-regular
  - semi-regular

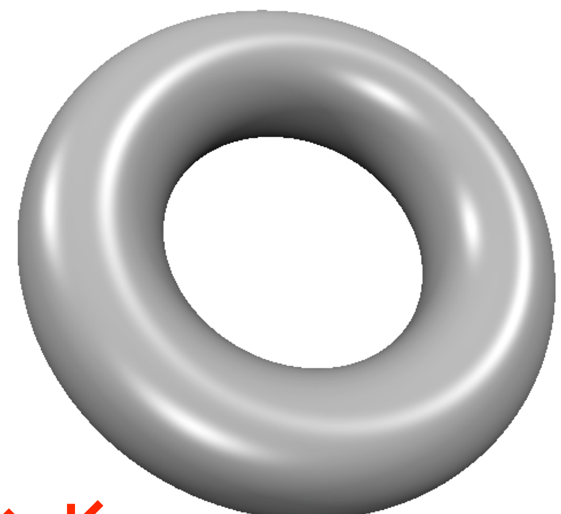


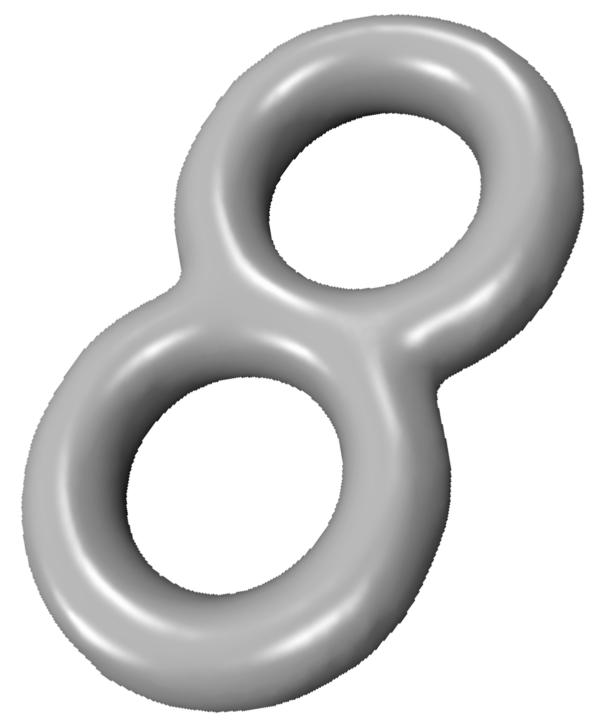


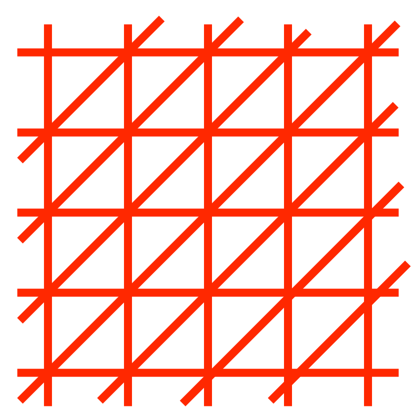
Regularity

- parametric
  - patches vs. polygons
  - Euler formula: **$V - E + F = 2\,(1-g)$**
  - **triangle** meshes
    - $F \approx 2V$
    - $E \approx 3V$
    - average valence = 6
  - quasi-regular
  - semi-regular



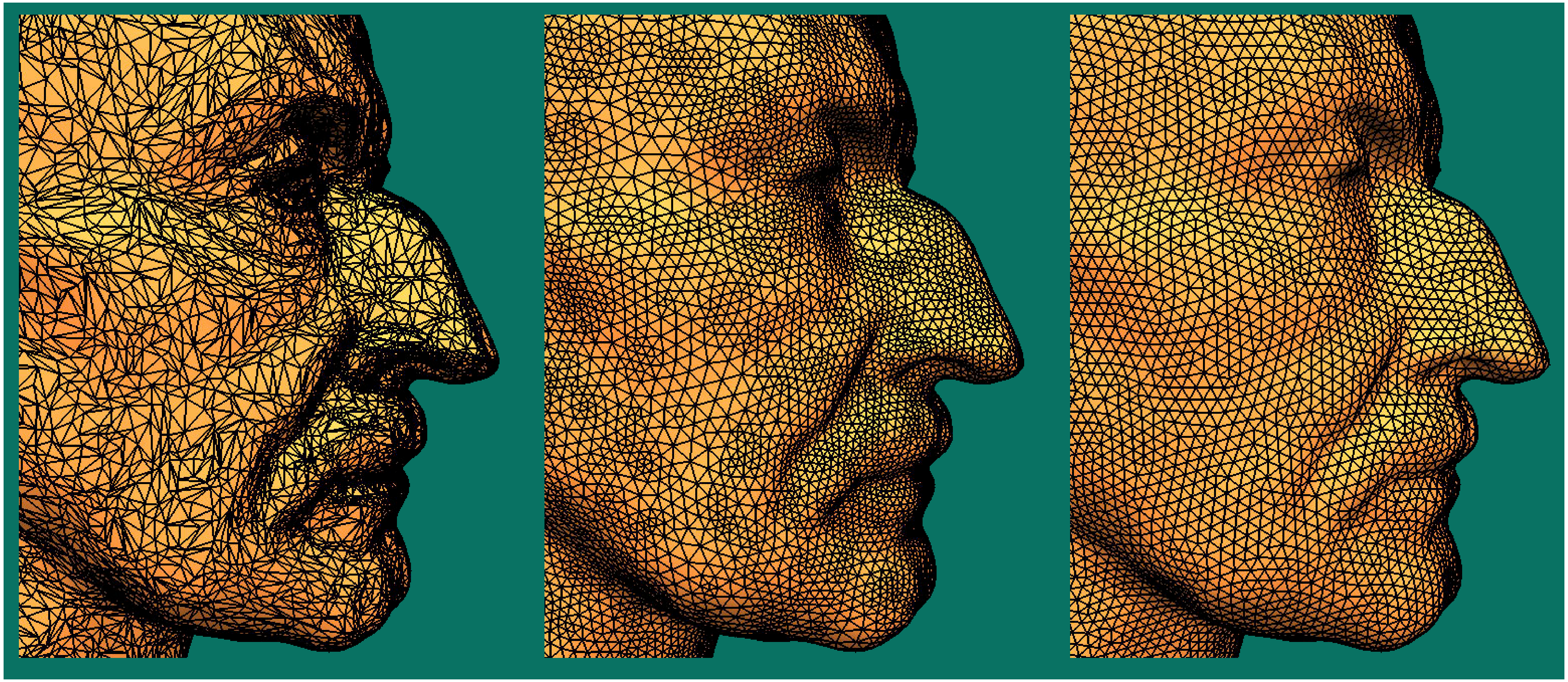






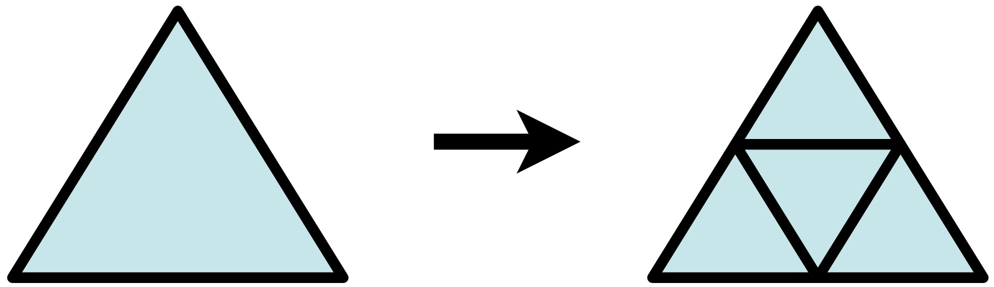
Regularity

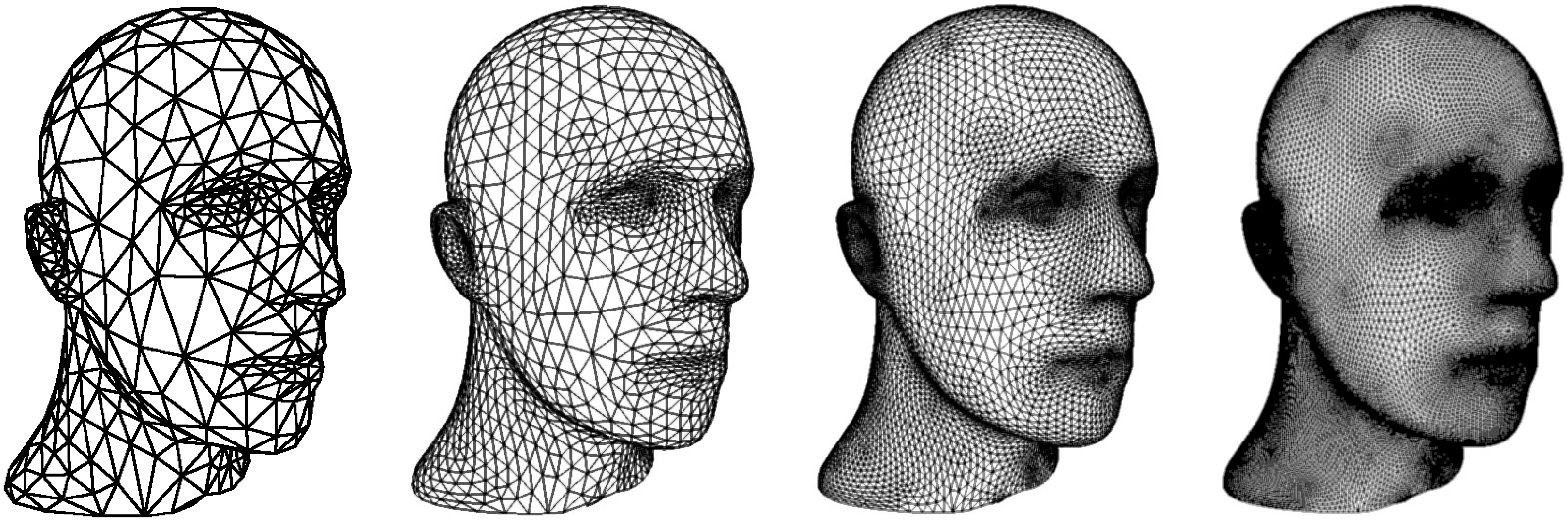
- quasi regular



Regularity

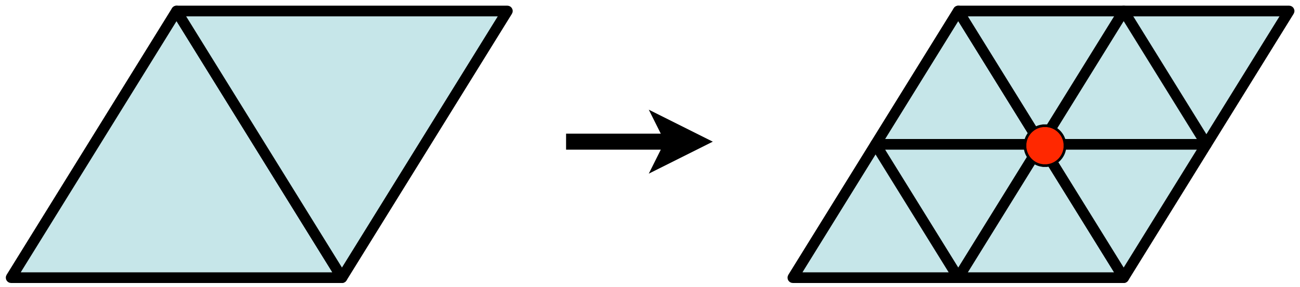
- semi regular

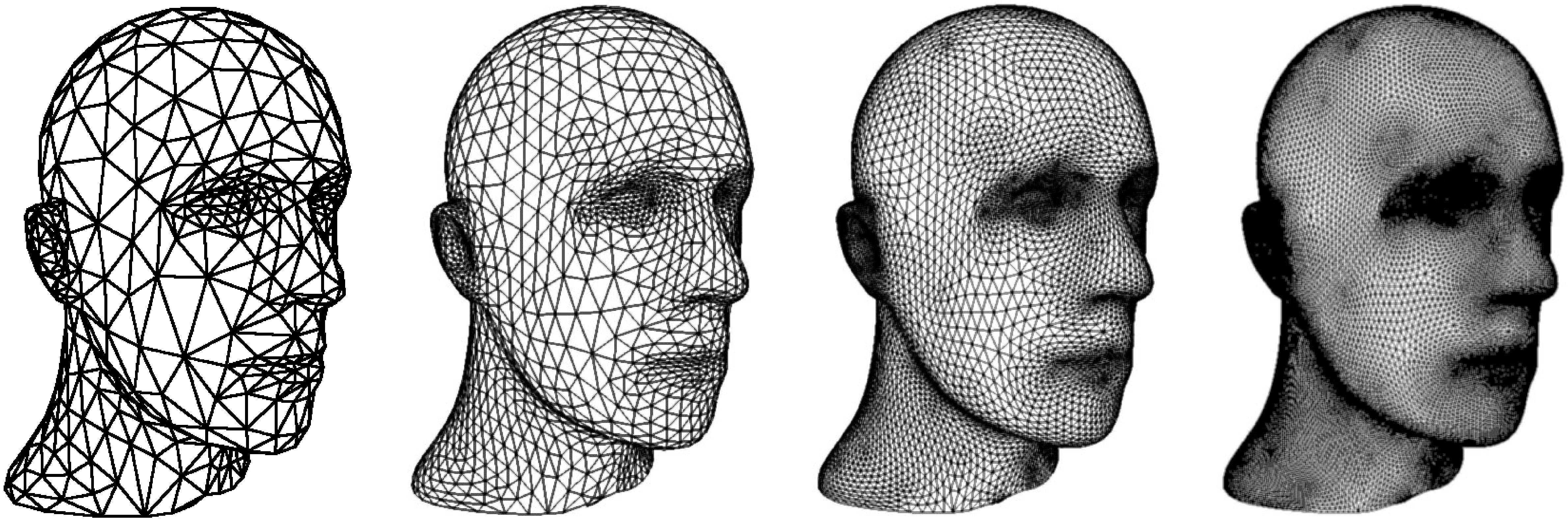




Regularity

- semi regular

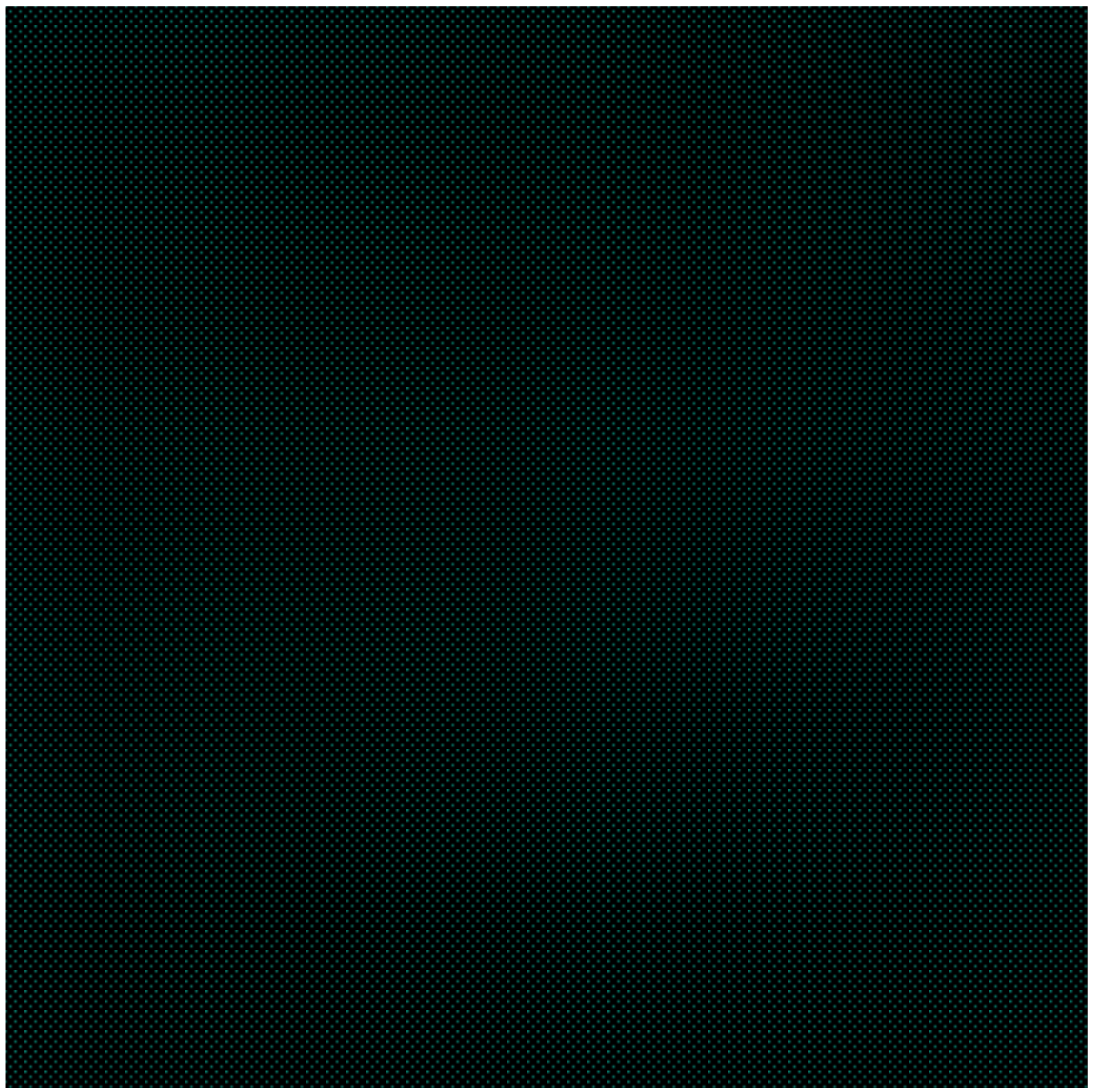




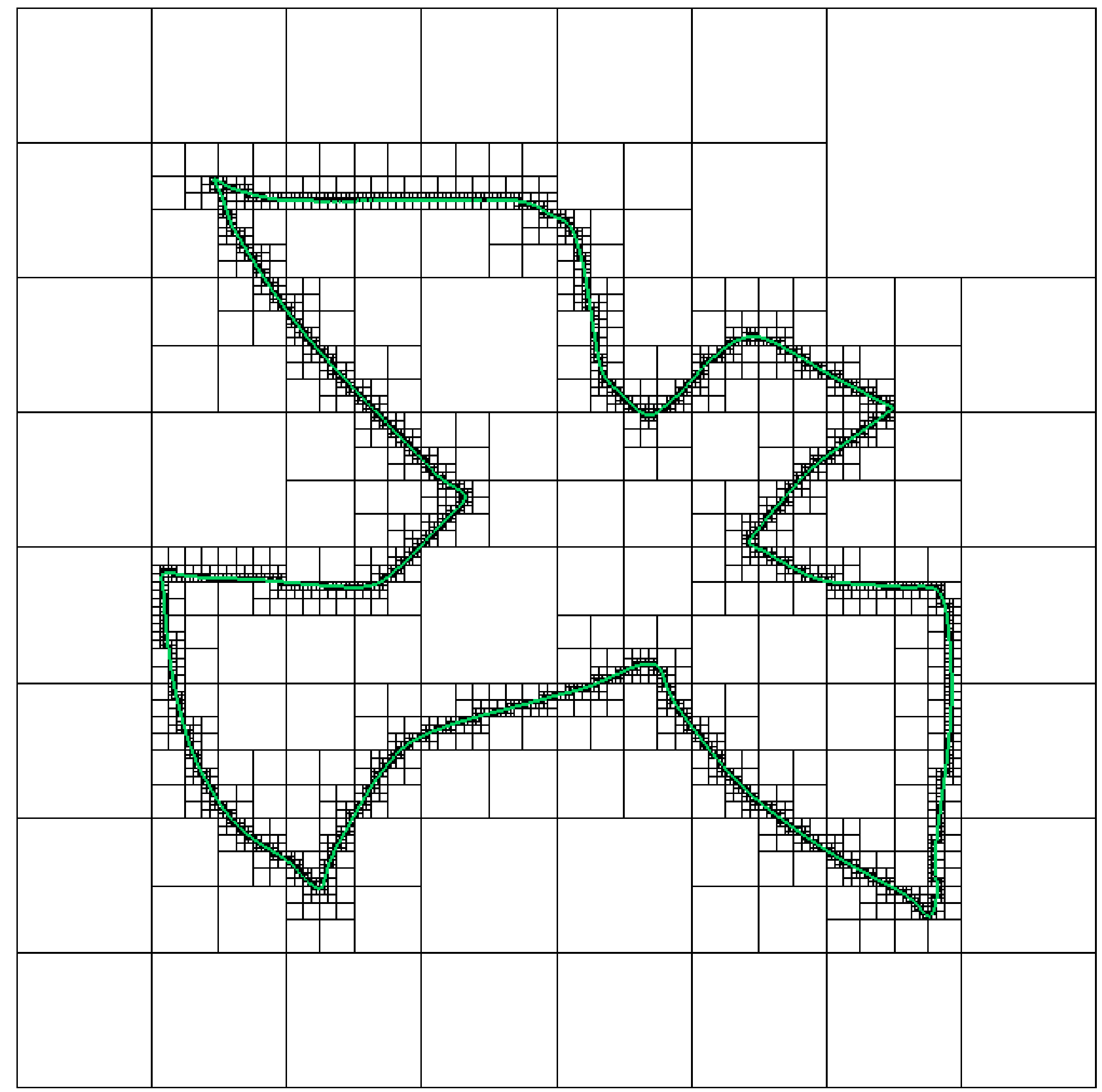
Regularity

- implicit
  - regular voxel grids $O(h^{-3})$
  - three color octrees
    - surface-adaptive refinement $O(h^{-2})$
    - feature-adaptive refinement $O(h^{-1})$
  - irregular hierarchies
    - binary space partition $O(h^{-1})$ (BSP)

3-Color Octree

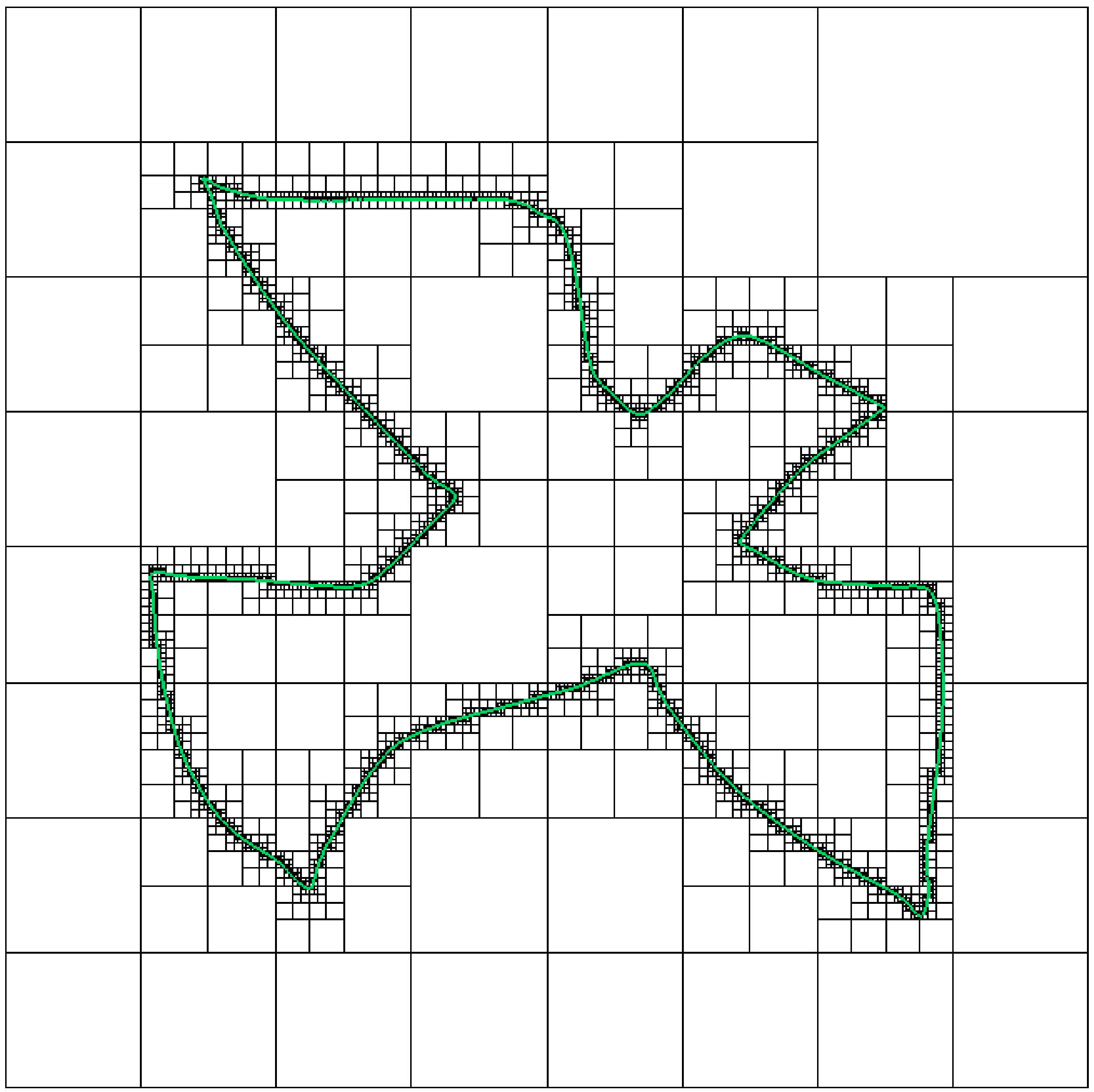


1048576 cells

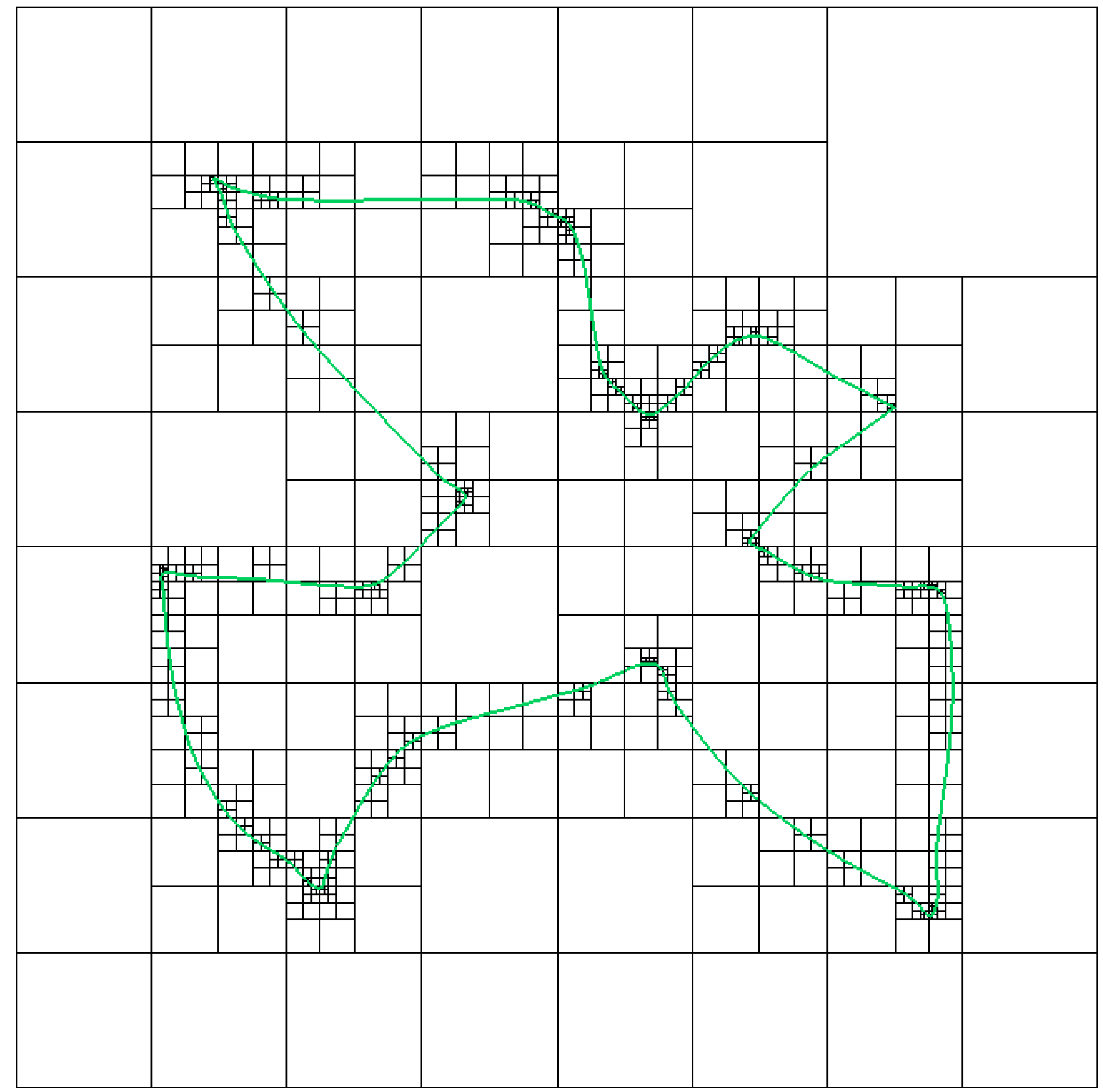


12040 cells

Adaptively Sampled Distance Fields

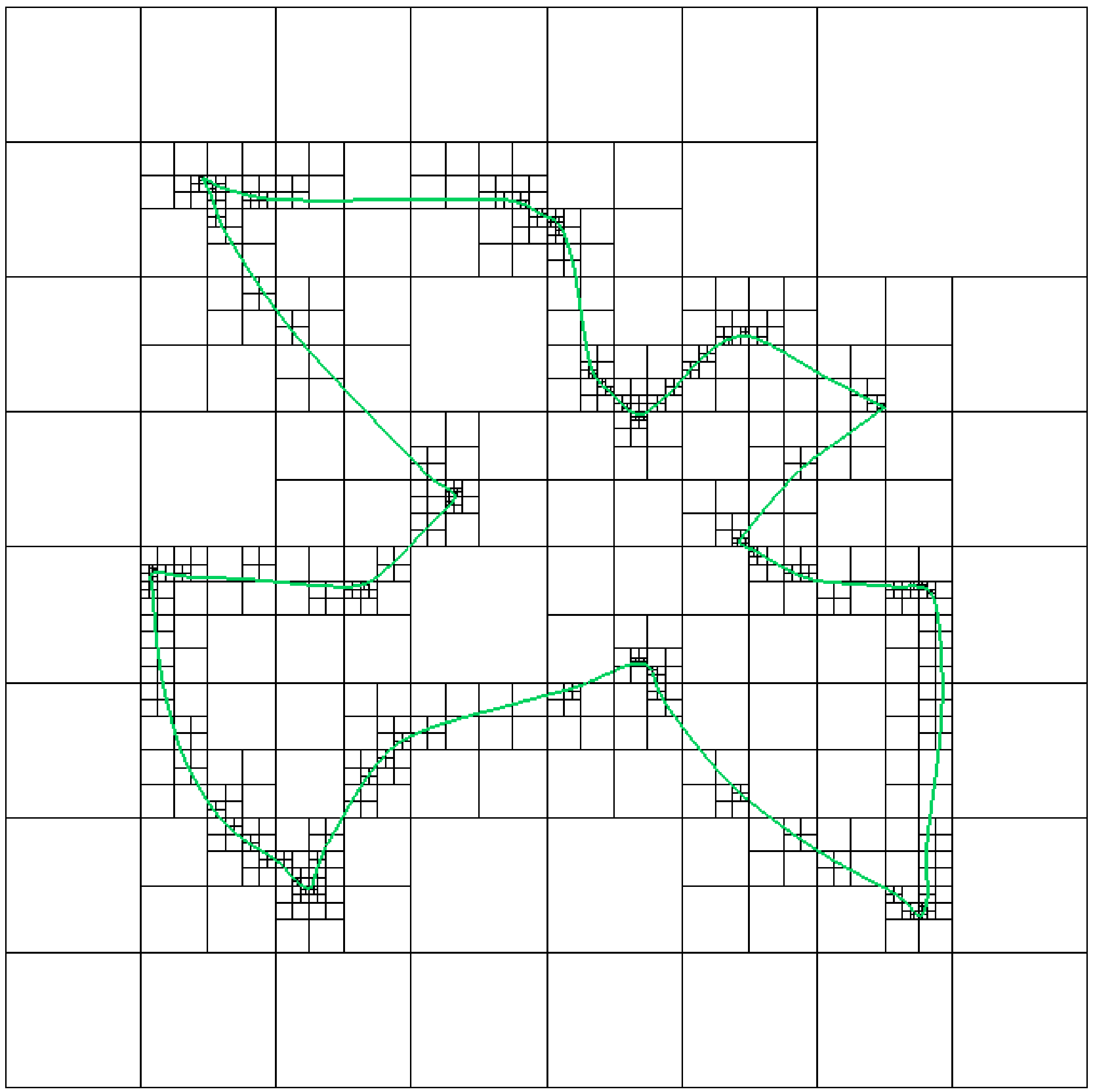


12040 cells

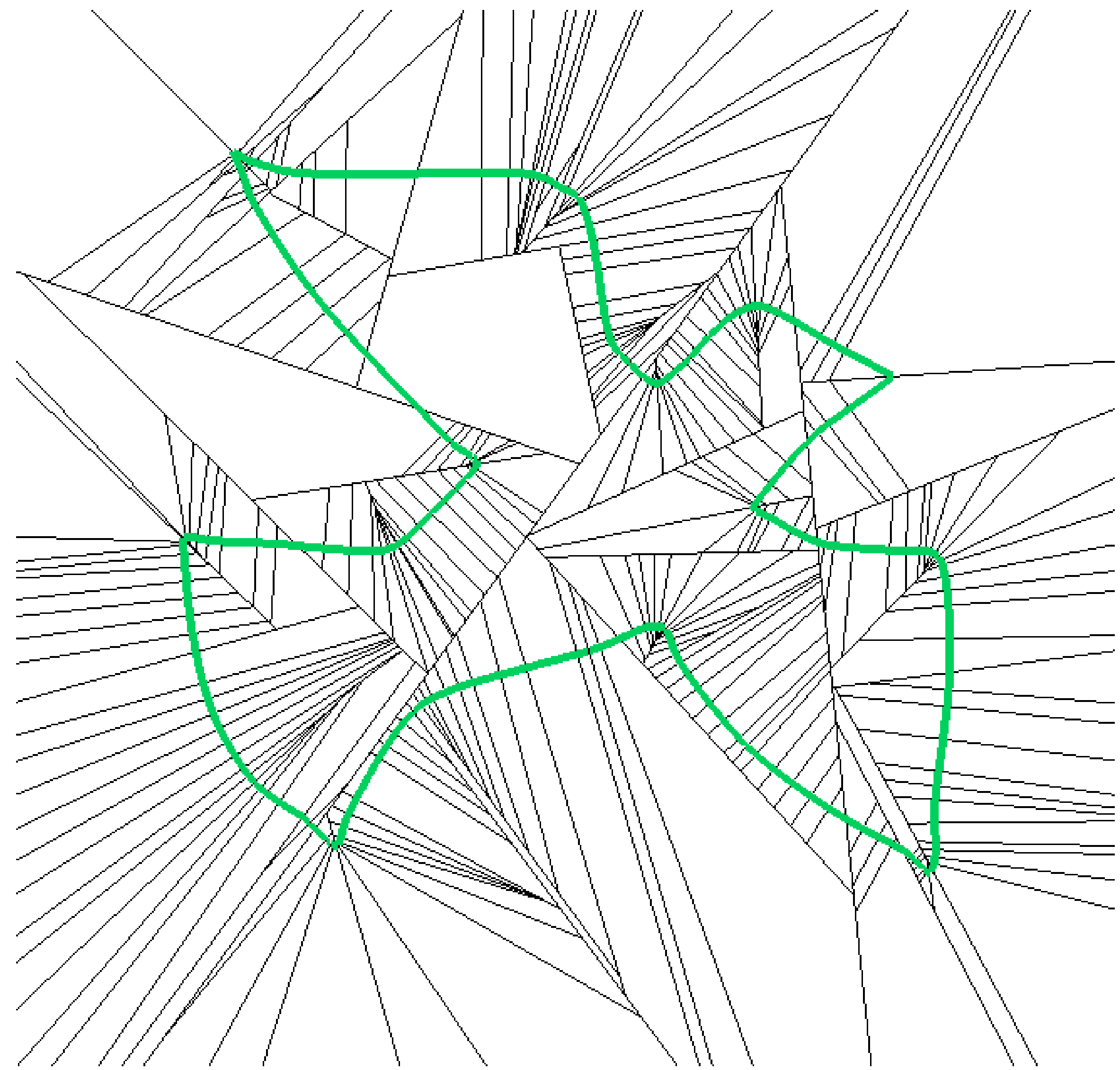


895 cells

Binary Space Partitions



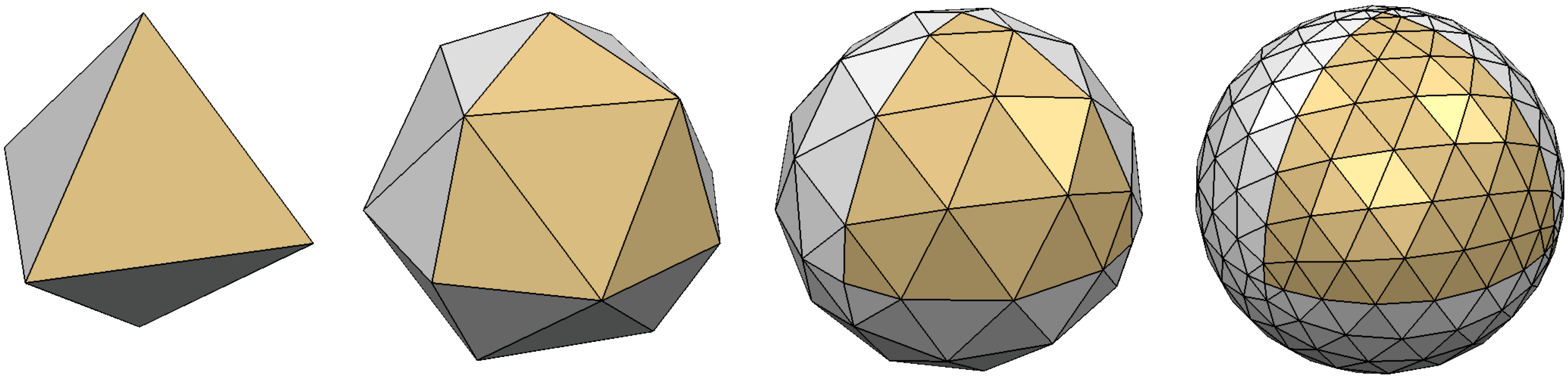
895 cells



254 cells

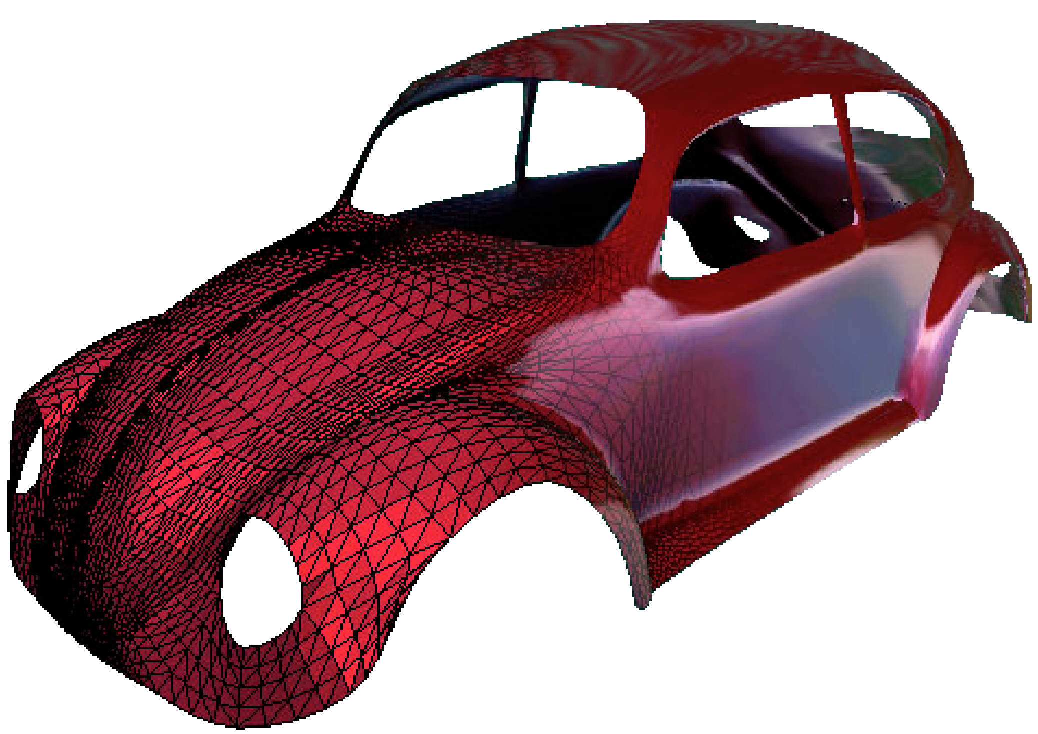
Message of the Day ...

- polygonal meshes are a good compromise
  - approximation $o(h^2)$ ... error * #faces = const.
  - arbitrary topology
  - flexibility for piecewise smooth surfaces
  - flexibility for adaptive refinement



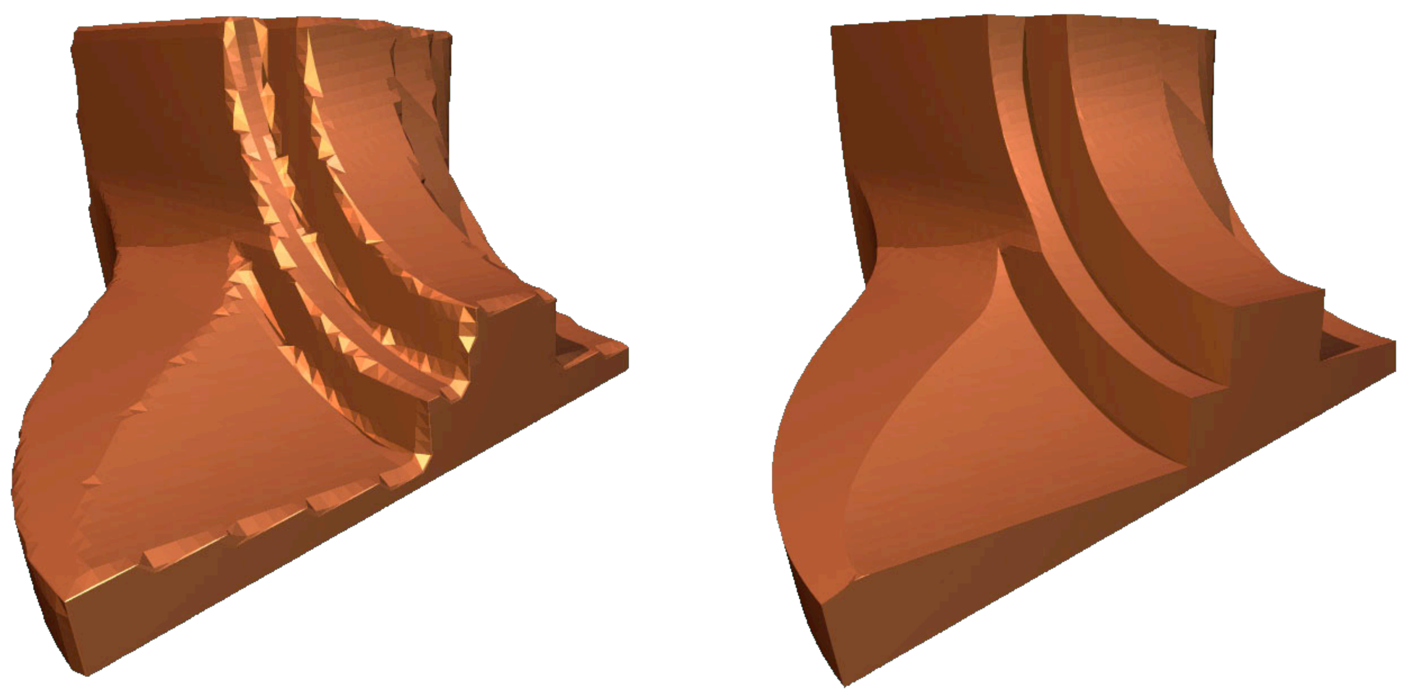
Message of the Day ...

- polygonal meshes are a good compromise
  - approximation $o(h^2)$ ... error * #faces = const.
  - arbitrary topology
  - flexibility for piecewise smooth surfaces
  - flexibility for adaptive refinement



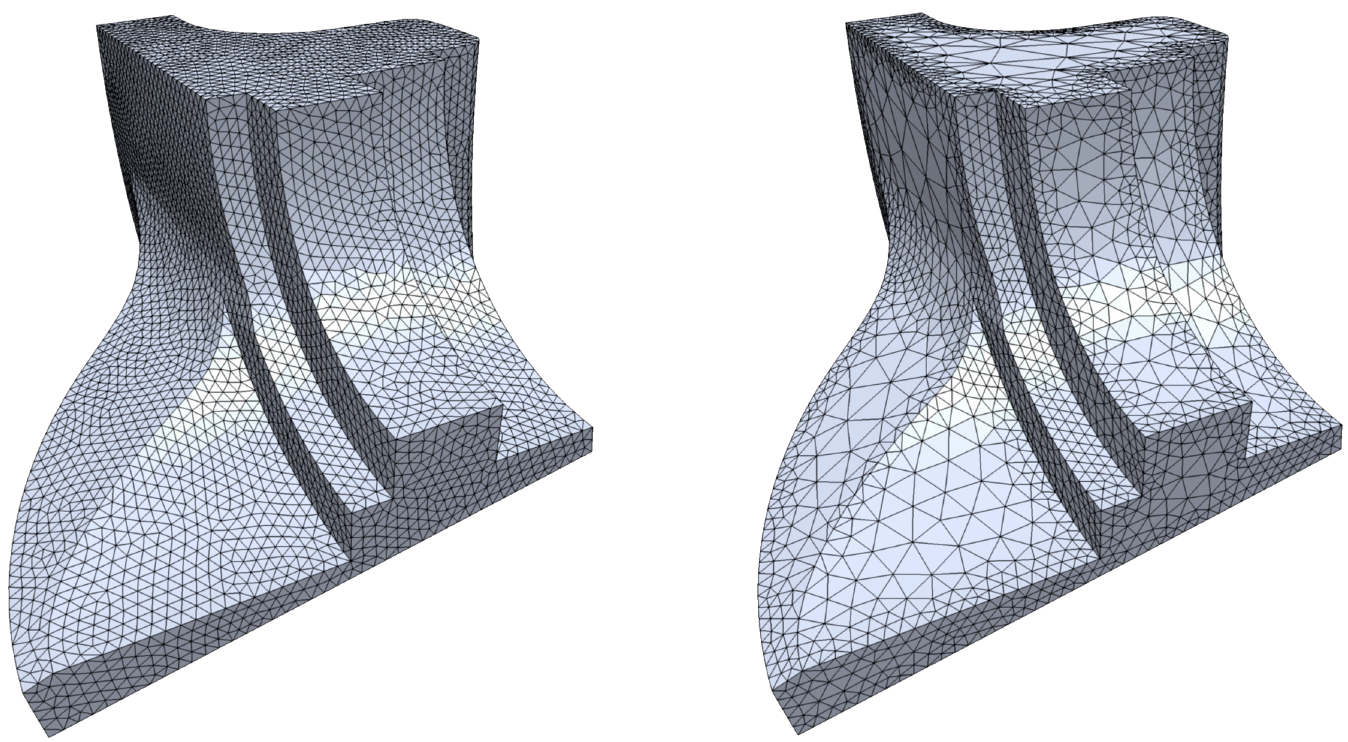
Message of the Day ...

- polygonal meshes are a good compromise
  - approximation $o(h^2)$ ... error * #faces = const.
  - arbitrary topology
  - flexibility for piecewise smooth surfaces
  - flexibility for adaptive refinement



Message of the Day ...

- polygonal meshes are a good compromise
  - approximation $o(h^2)$ ... error * #faces = const.
  - arbitrary topology
  - flexibility for piecewise smooth surfaces
  - flexibility for adaptive refinement



Message of the Day ...

- polygonal meshes are a good compromise
  - approximation $o(h^2)$ ... error * #faces = const.
  - arbitrary topology
  - flexibility for piecewise smooth surfaces
  - flexibility for adaptive refinement
  - efficient rendering

Message of the Day ...

- polygonal meshes are a good compromise
  - approximation $o(h^2)$ ... error * #faces = const.
  - arbitrary topology
  - flexibility for piecewise smooth surfaces
  - flexibility for adaptive refinement
  - efficient rendering

- implicit representation can support efficient access to vertices, faces, ...

Outline

- (mathematical) geometry representations
  - parametric vs. implicit
- approximation properties
- types of operations
  - distance queries
  - evaluation
  - modification / deformation
- data structures

Distance Queries

- parametric
  - find orthogonal base point

$$[\mathbf{p} - \mathbf{f}(u,v)] \times \mathbf{n}(u,v) \; = \; \mathbf{0}$$

  - for triangle meshes
    - use kd-tree or BSP to find closest triangle
    - find base point by Newton iteration
      (use Phong normal field)

Evaluation

- parametric
  - positions $\mathbf{f}(u,v)$
  - normals $\mathbf{n}(u,v) \;=\; \mathbf{f}_u(u,v) \times \mathbf{f}_v(u,v)$
  - curvatures $\mathbf{c}(u,v) \;=\; C\Big(\mathbf{f}_{uu}(u,v), \mathbf{f}_{uv}(u,v), \mathbf{f}_{vv}(u,v)\Big)$
- generalization to triangle meshes
  - positions (barycentric coordinates)

$$(\alpha, \beta) \;\mapsto\; \alpha\,\mathbf{P}_1 + \beta\,\mathbf{P}_2 + (1-\alpha-\beta)\,\mathbf{P}_3$$

Evaluation

- parametric
  - positions $\mathbf{f}(u,v)$
  - normals $\mathbf{n}(u,v) = \mathbf{f}_u(u,v) \times \mathbf{f}_v(u,v)$
  - curvatures $\mathbf{c}(u,v) = C\Big(\mathbf{f}_{uu}(u,v), \mathbf{f}_{uv}(u,v), \mathbf{f}_{vv}(u,v)\Big)$
- generalization to triangle meshes
  - positions (barycentric coordinates)

$$(\alpha, \beta, \gamma) \mapsto \alpha\,\mathbf{P}_1 + \beta\,\mathbf{P}_2 + \gamma\,\mathbf{P}_3$$

$$\alpha + \beta + \gamma = 0$$

Evaluation

- parametric
  - positions $\mathbf{f}(u,v)$
  - normals $\mathbf{n}(u,v) \;=\; \mathbf{f}_u(u,v) \times \mathbf{f}_v(u,v)$
  - curvatures $\mathbf{c}(u,v) \;=\; C\Big(\mathbf{f}_{uu}(u,v), \mathbf{f}_{uv}(u,v), \mathbf{f}_{vv}(u,v)\Big)$
- generalization to triangle meshes
  - positions (barycentric coordinates)

$$\alpha\,\mathbf{u} + \beta\,\mathbf{v} + \gamma\,\mathbf{w} \;\mapsto\; \alpha\,\mathbf{P}_1 + \beta\,\mathbf{P}_2 + \gamma\,\mathbf{P}_3$$

$$\alpha + \beta + \gamma \;=\; 0$$

Evaluation

- parametric
  - positions $\mathbf{f}(u,v)$
  - normals $\mathbf{n}(u,v) \;=\; \mathbf{f}_u(u,v) \times \mathbf{f}_v(u,v)$
  - curvatures $\mathbf{c}(u,v) \;=\; C\Big(\mathbf{f}_{uu}(u,v), \mathbf{f}_{uv}(u,v), \mathbf{f}_{vv}(u,v)\Big)$
- generalization to triangle meshes
  - positions (barycentric coordinates)
  - normals (per face, Phong)

$$\mathbf{N} = (\mathbf{P}_2 - \mathbf{P}_1) \times (\mathbf{P}_3 - \mathbf{P}_1)$$

Evaluation

- parametric
  - positions $\mathbf{f}(u, v)$
  - normals $\mathbf{n}(u, v) = \mathbf{f}_u(u, v) \times \mathbf{f}_v(u, v)$
  - curvatures $\mathbf{c}(u, v) = C\Big(\mathbf{f}_{uu}(u, v), \mathbf{f}_{uv}(u, v), \mathbf{f}_{vv}(u, v)\Big)$
- generalization to triangle meshes
  - positions (barycentric coordinates)
  - normals (per face, Phong)

$$\alpha\, \mathbf{u} + \beta\, \mathbf{v} + \gamma\, \mathbf{w} \;\mapsto\; \alpha\, \mathbf{N}_1 + \beta\, \mathbf{N}_2 + \gamma\, \mathbf{N}_3$$

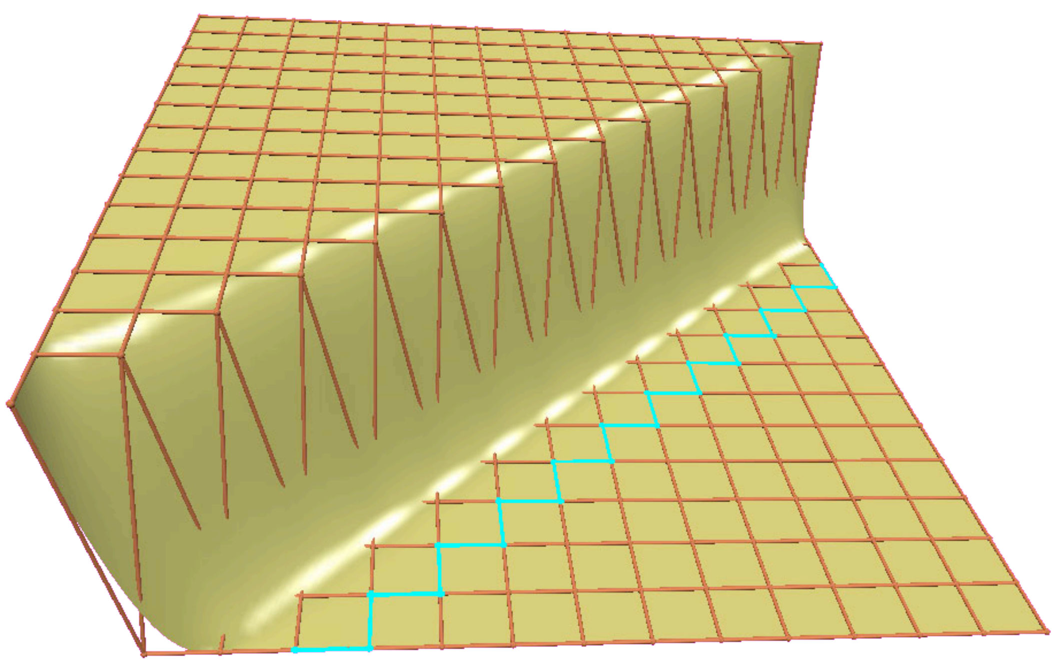
Evaluation

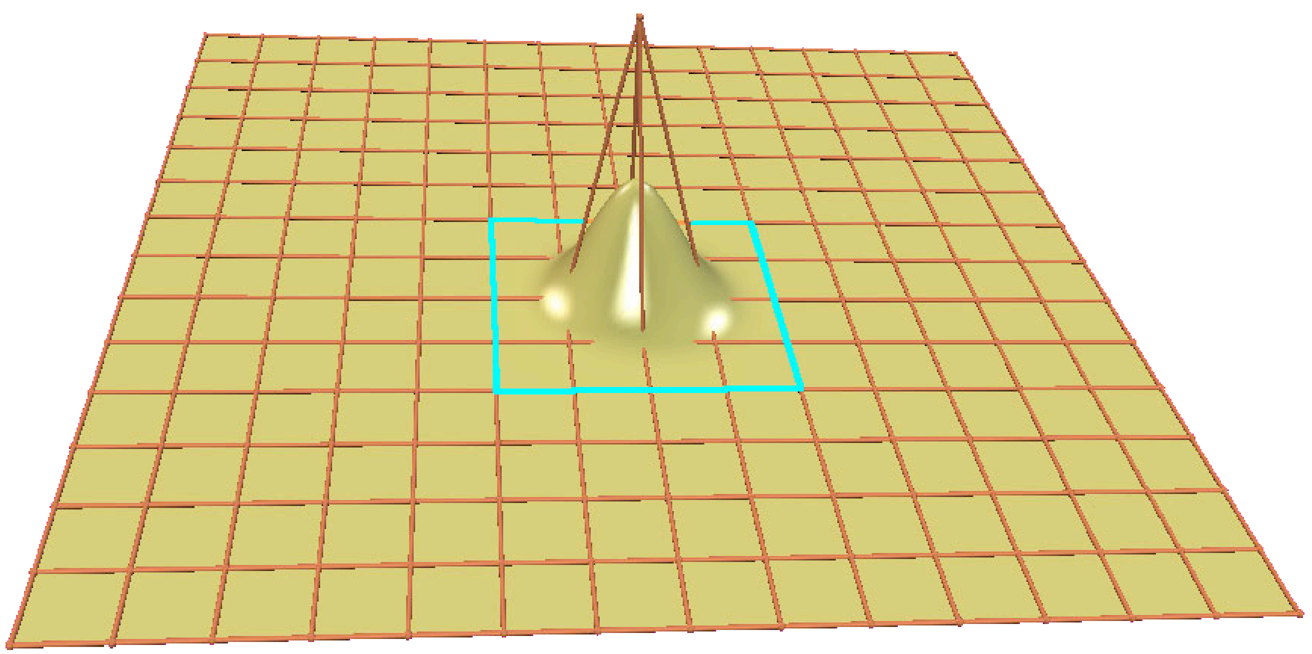
- parametric
  - positions $\mathbf{f}(u,v)$
  - normals $\mathbf{n}(u,v) = \mathbf{f}_u(u,v) \times \mathbf{f}_v(u,v)$
  - curvatures $\mathbf{c}(u,v) = C\Big(\mathbf{f}_{uu}(u,v), \mathbf{f}_{uv}(u,v), \mathbf{f}_{vv}(u,v)\Big)$
- generalization to triangle meshes
  - positions (barycentric coordinates)
  - normals (per face, Phong)
  - curvatures ... later

Modifications

- parameteric
  - control vertices
  - free-form deformation
  - boundary constraint modeling

$$\mathbf{f}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{c}_{ij} N_i^n(u) N_j^m(v)$$

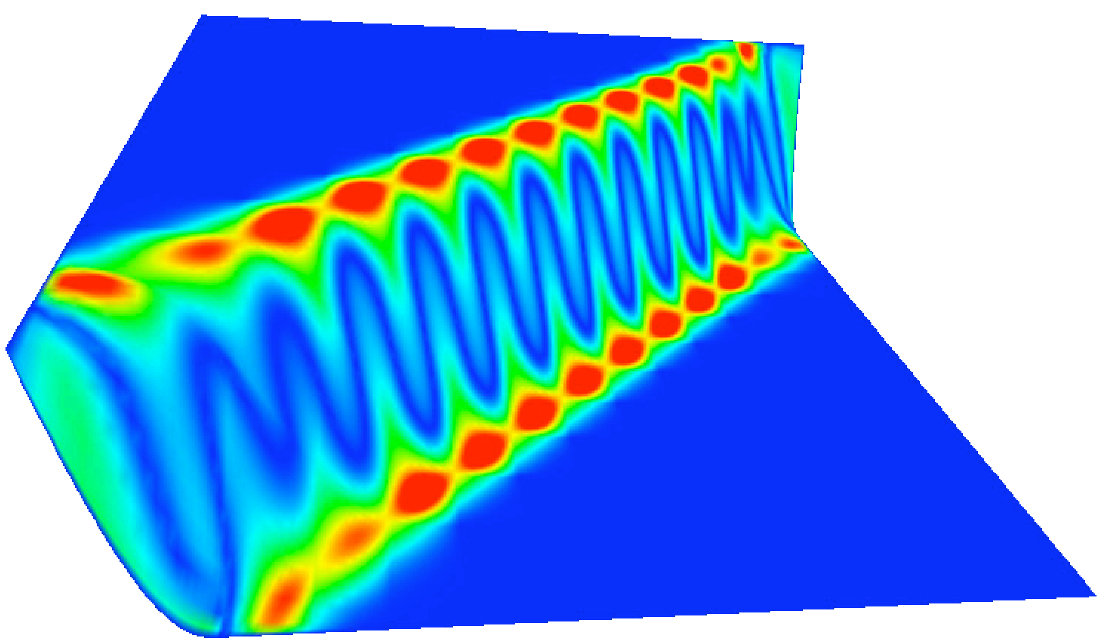


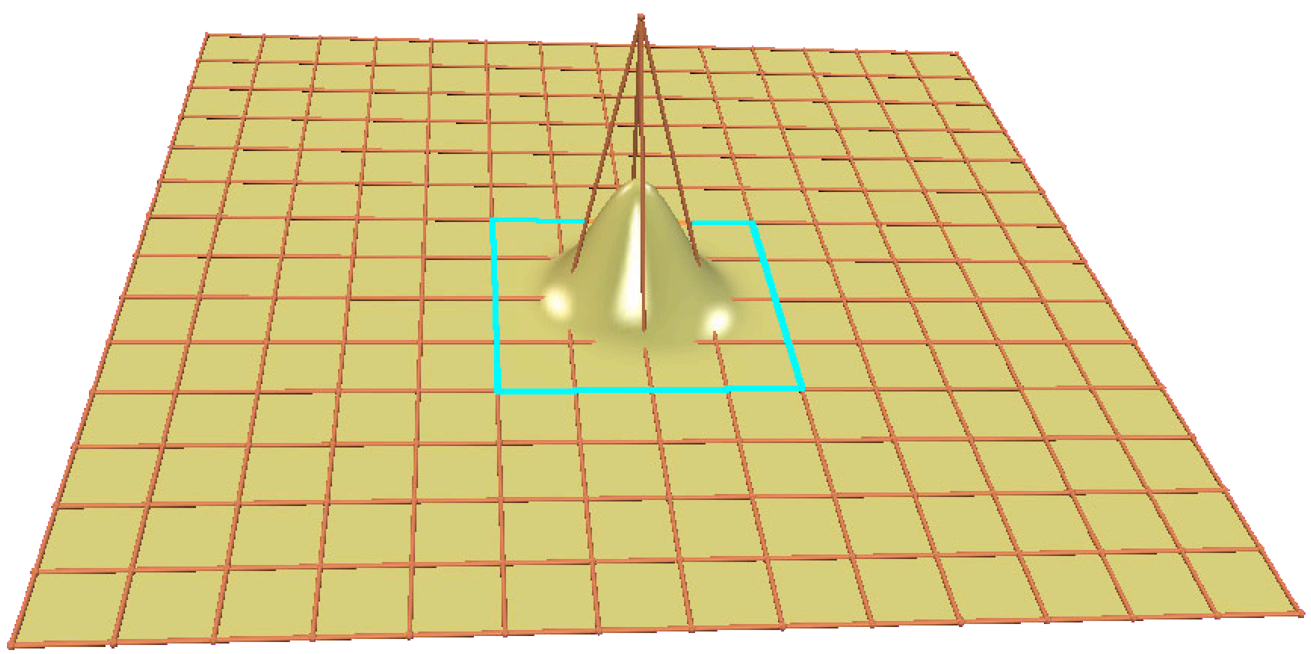


Modifications

- parameteric
  - control vertices
  - free-form deformation
  - boundary constraint modeling

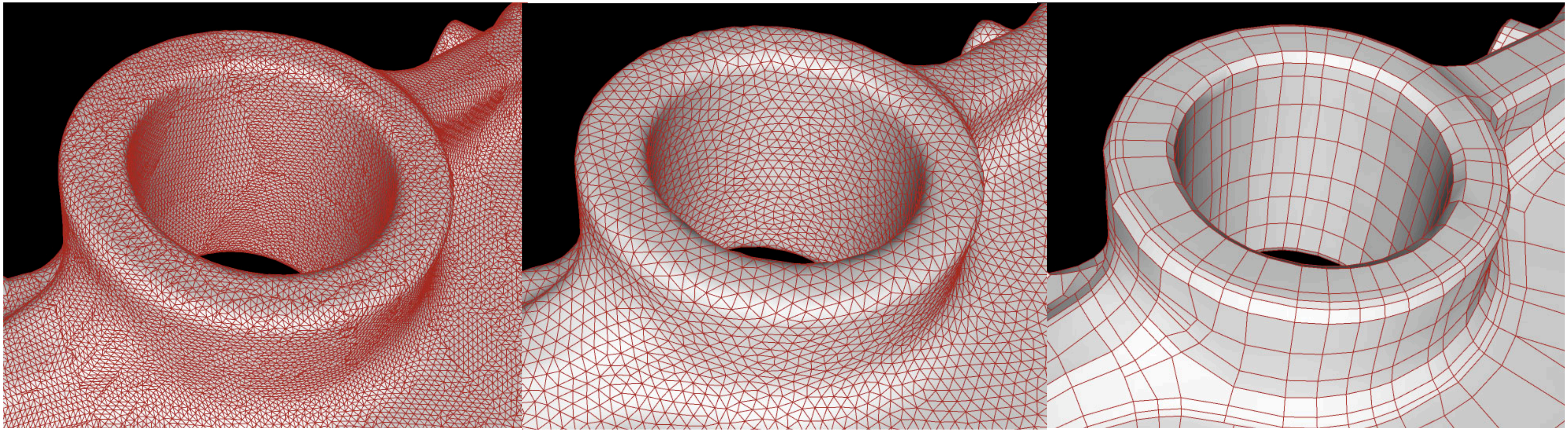
$$\mathbf{f}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} \mathbf{c}_{ij} N_i^n(u)\, N_j^m(v)$$





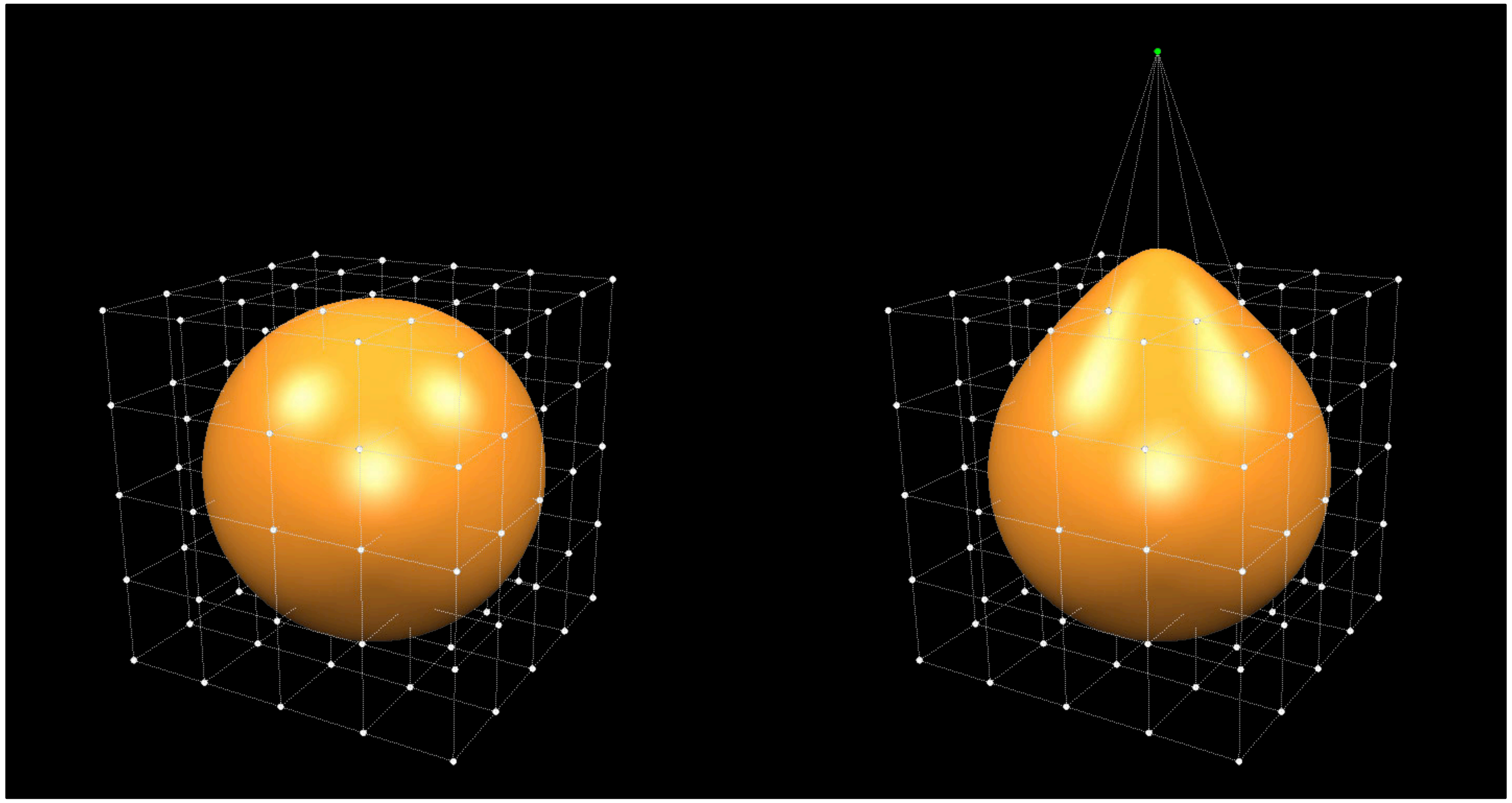
Modifications

- parameteric
  - control vertices
  - free-form deformation
  - boundary constraint modeling



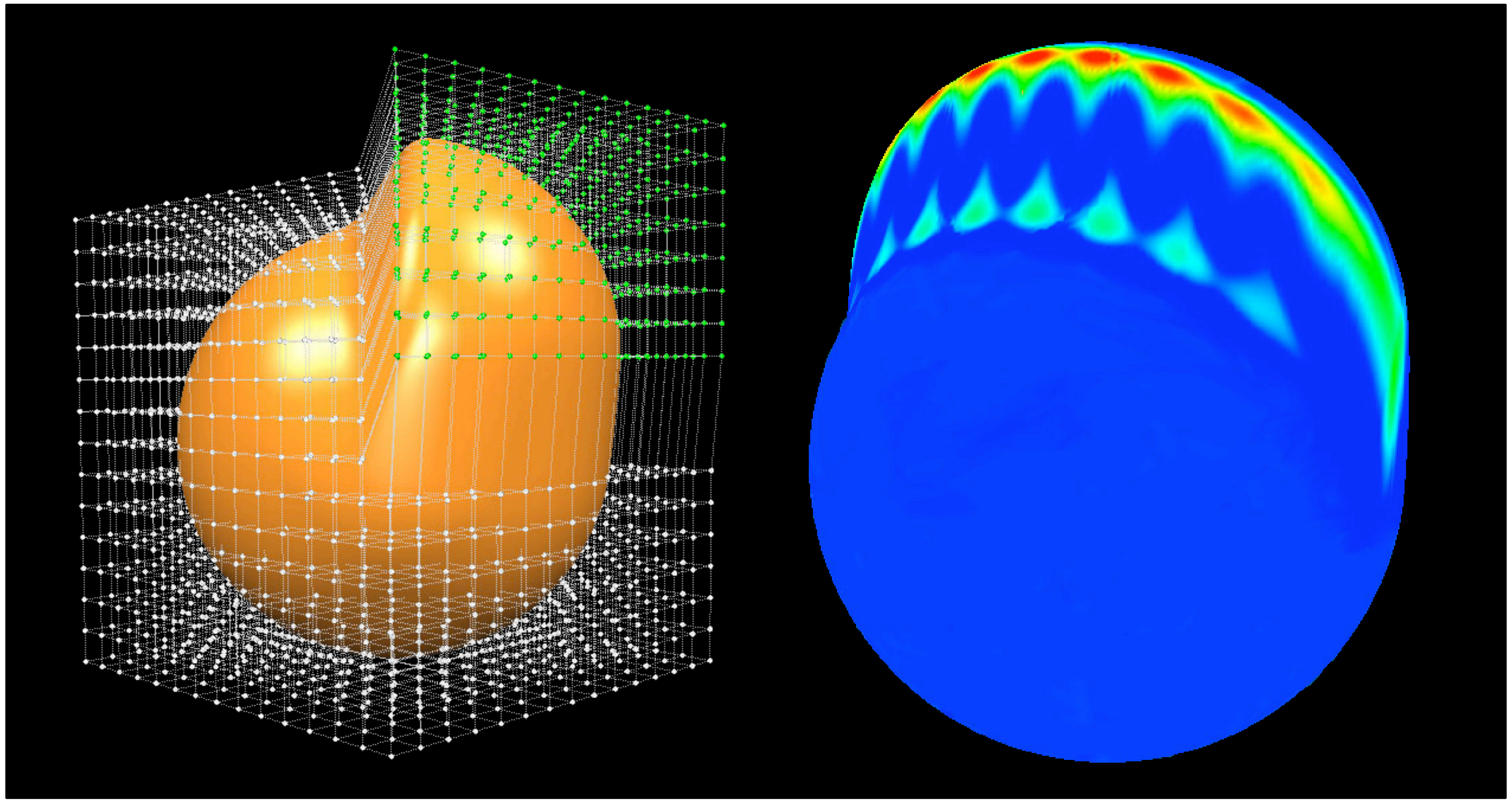
Modifications

- parameteric
  - control vertices
  - free-form deformation
  - boundary constraint modeling



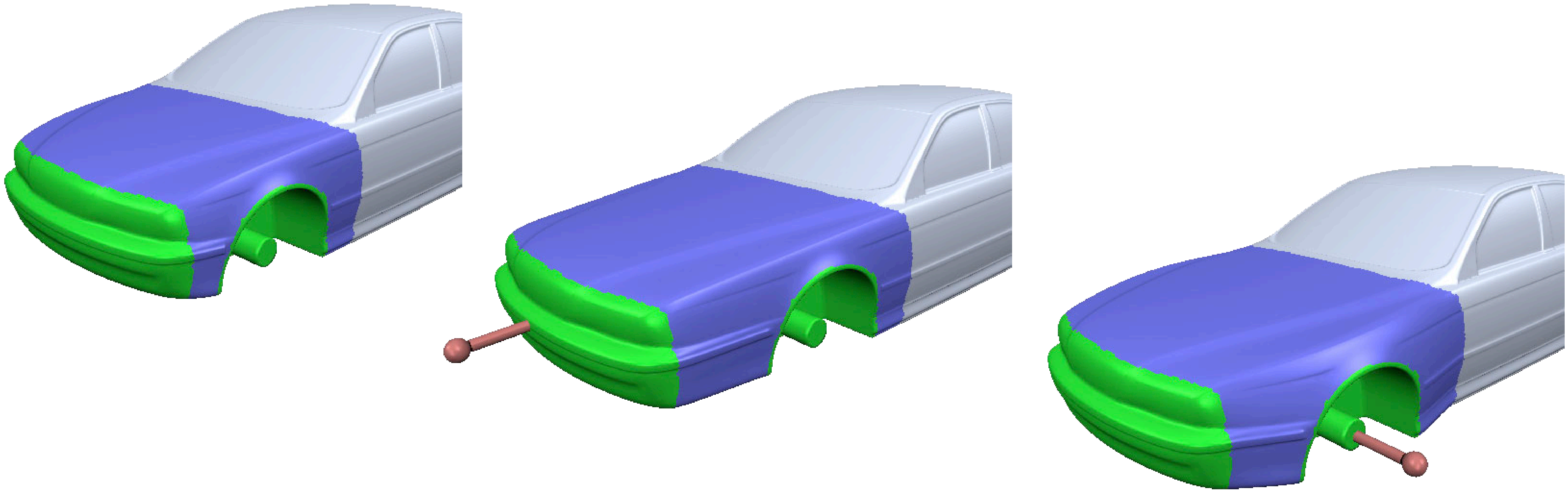
Modifications

- parameteric
  - control vertices
  - free-form deformation
  - boundary constraint modeling



Modifications

- parameteric
  - control vertices
  - free-form deformation
  - boundary constraint modeling



Outline

- (mathematical) geometry representations
  - parametric vs. implicit
- approximation properties
- types of operations
  - distance queries
  - evaluation
  - modification / deformation
- data structures

Mesh Data Structures

- how to store geometry & connectivity?
- compact storage
  – file formats
- efficient algorithms on meshes
  – identify time-critical operations
  – all vertices/edges of a face
  – all incident vertices/edges/faces of a vertex

Face Set (STL)

- face:
  - 3 positions

Triangles		
x_{11} y_{11} z_{11}	x_{12} y_{12} z_{12}	x_{13} y_{13} z_{13}
x_{21} y_{21} z_{21}	x_{22} y_{22} z_{22}	x_{23} y_{23} z_{23}
...	...	...
x_{F1} y_{F1} z_{F1}	x_{F2} y_{F2} z_{F2}	x_{F3} y_{F3} z_{F3}

36 B/f = 72 B/v

no connectivity!

Shared Vertex (OBJ, OFF)

- vertex:
  - position
- face:
  - vertex indices

Vertices
x_1 y_1 z_1
...
x_V y_V z_V

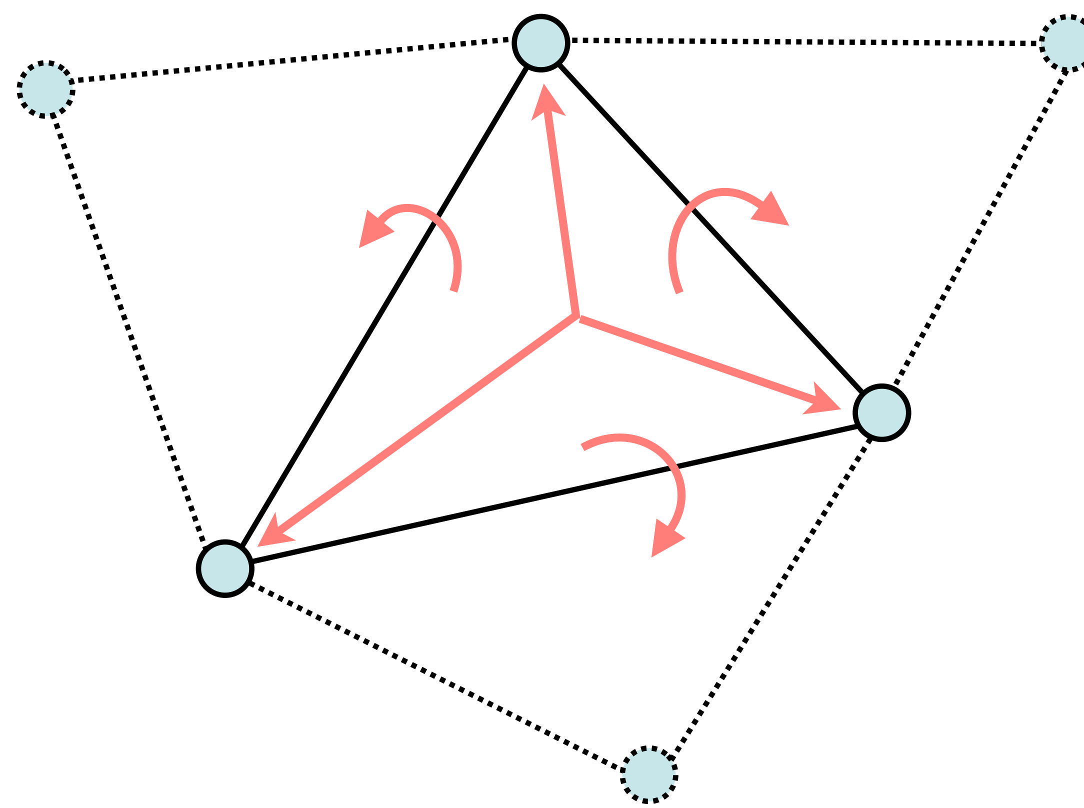
Triangles
v_{11} v_{12} v_{13}
...
...
...
...
v_{F1} v_{F2} v_{F3}

12 B/v + 12 B/f = 36 B/v

no neighborhood info

Face-Based Connectivity

- vertex:
  - position
  - 1 face
- face:
  - 3 vertices
  - 3 face neighbors

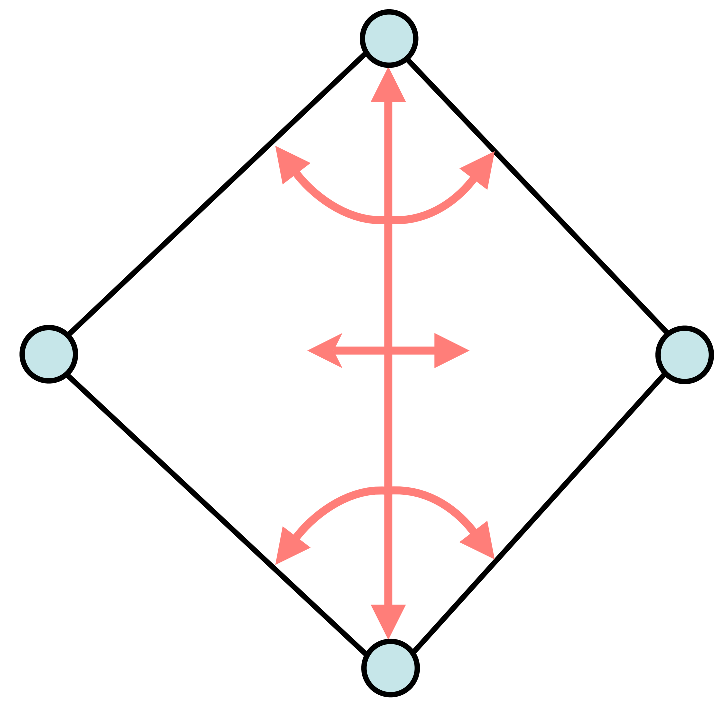


64 B/v

no edges!

Edge-Based Connectivity

- vertex
  - position
  - 1 edge
- edge
  - 2 vertices
  - 2 faces
  - 4 edges
- face
  - 1 edge

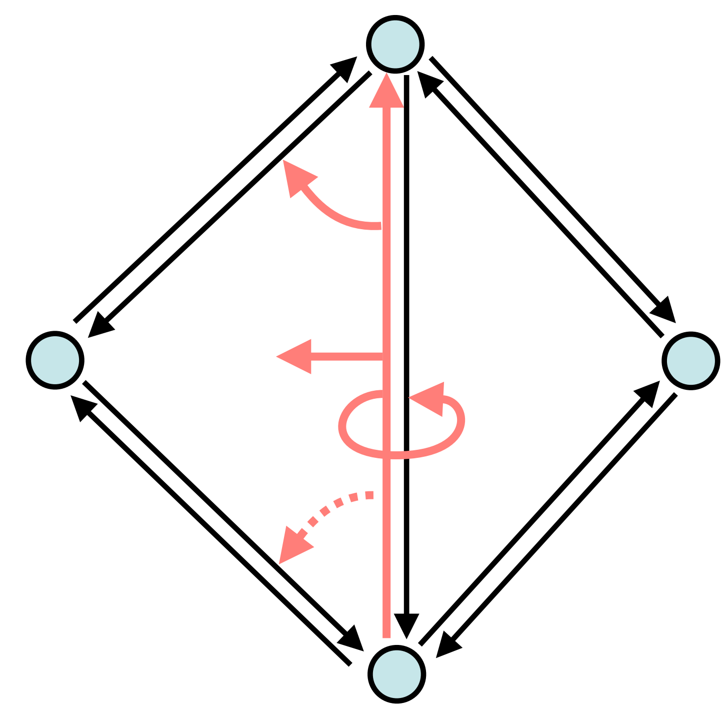


120 B/v

edge orientation?

Halfedge-Based Connectivity

- vertex
  - position
  - 1 halfedge
- halfedge
  - 1 vertex
  - 1 face
  - 1, 2, or 3 halfedges
- face
  - 1 halfedge

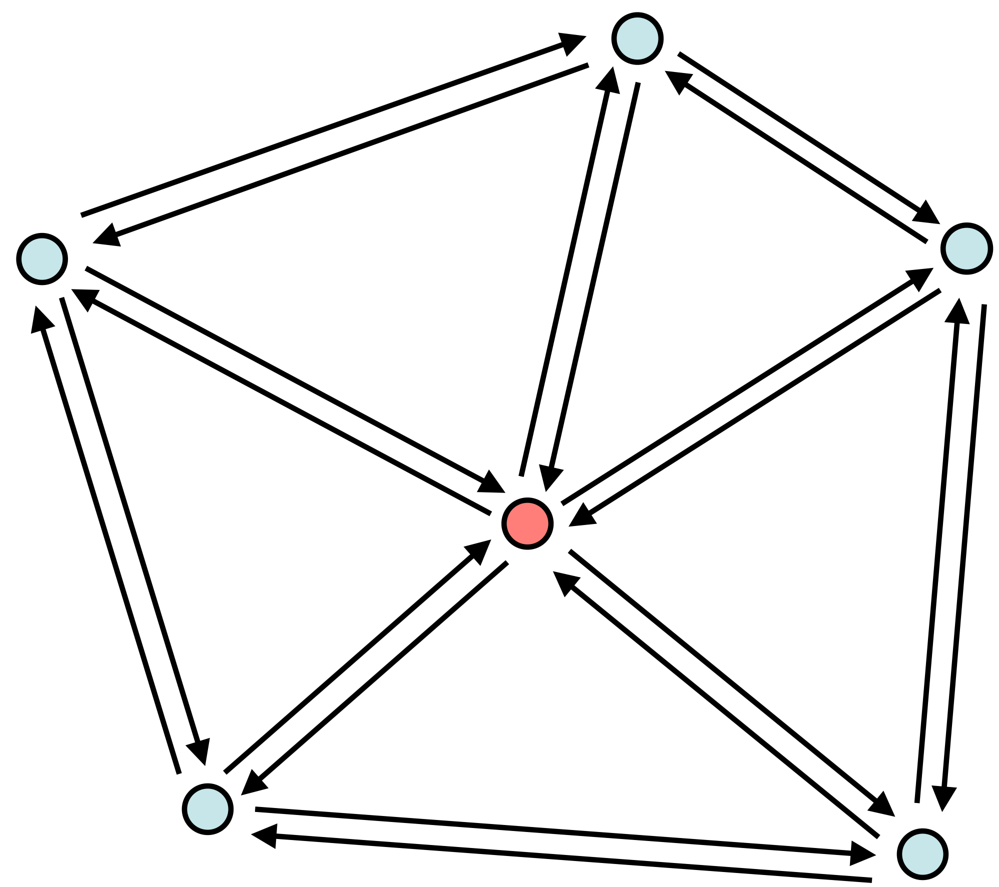


96 to 144 B/v

no case distinctions
during traversal

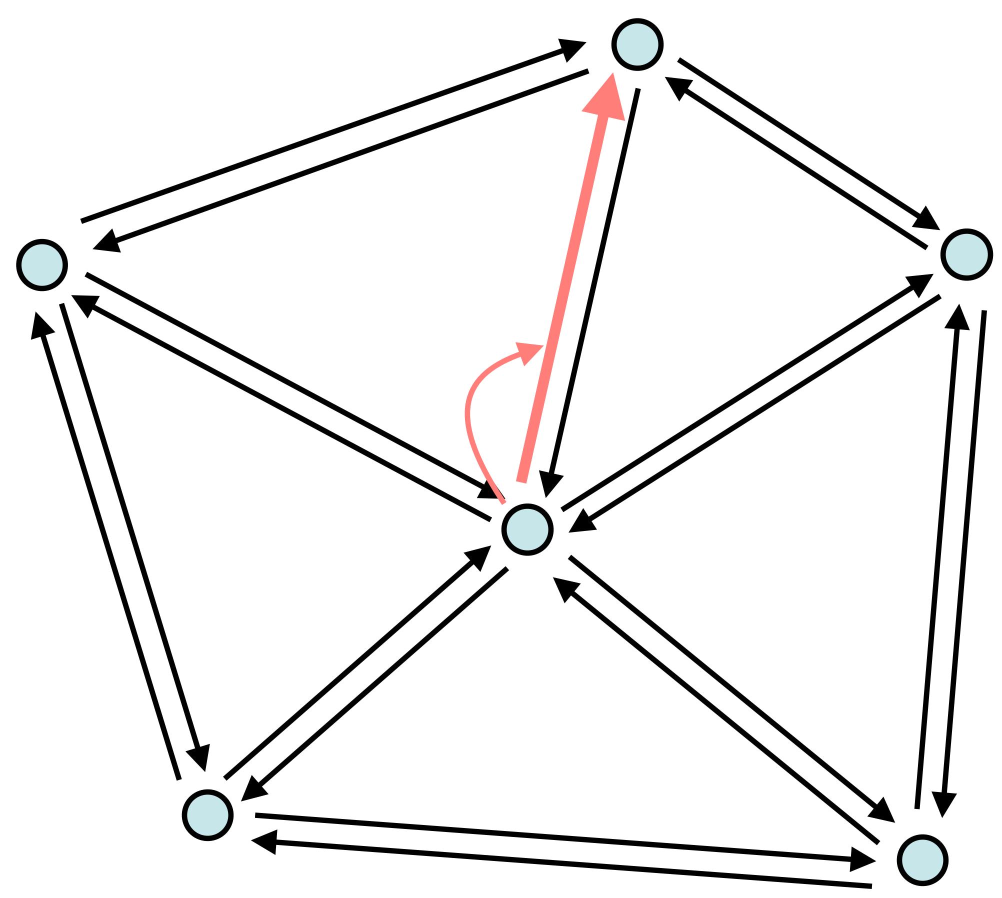
One-Ring Traversal

1. Start at vertex



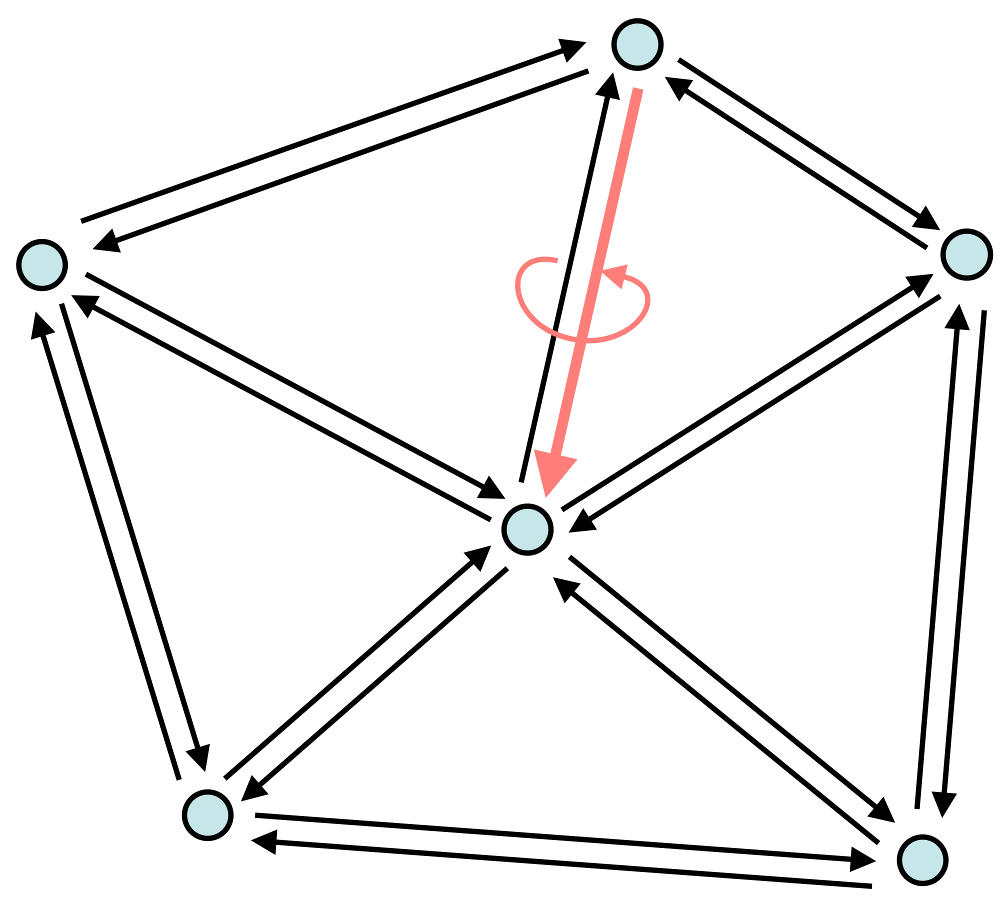
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge



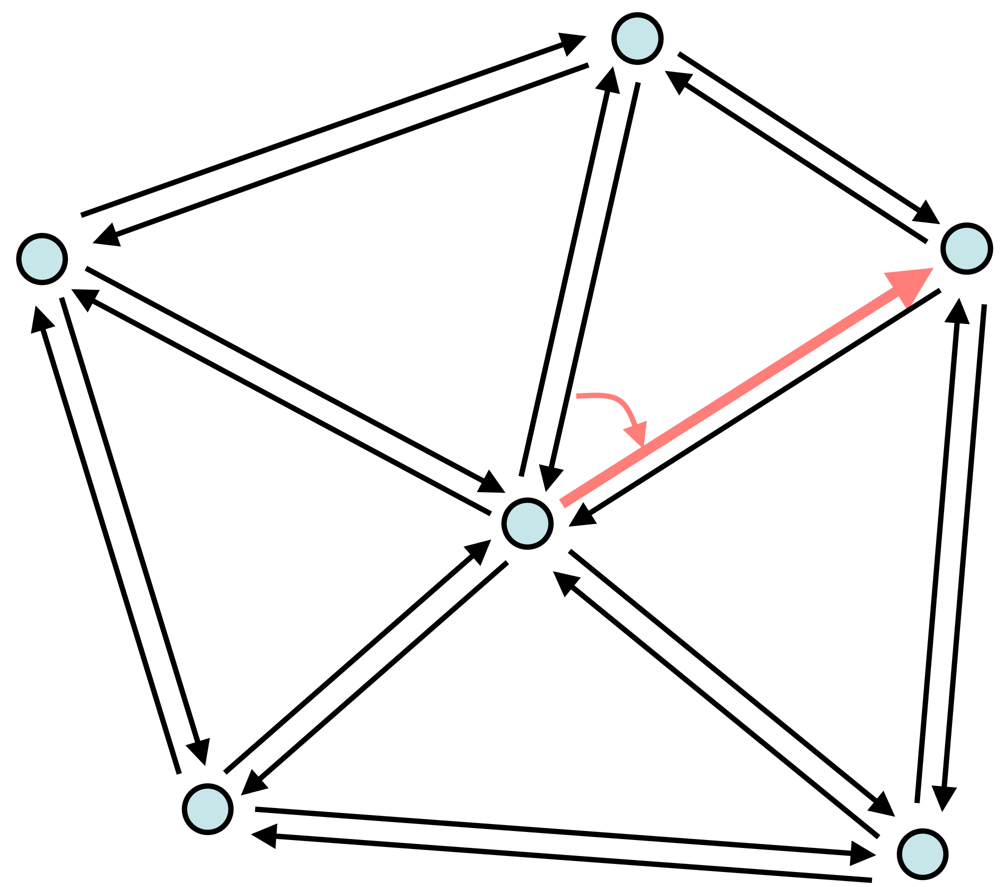
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge



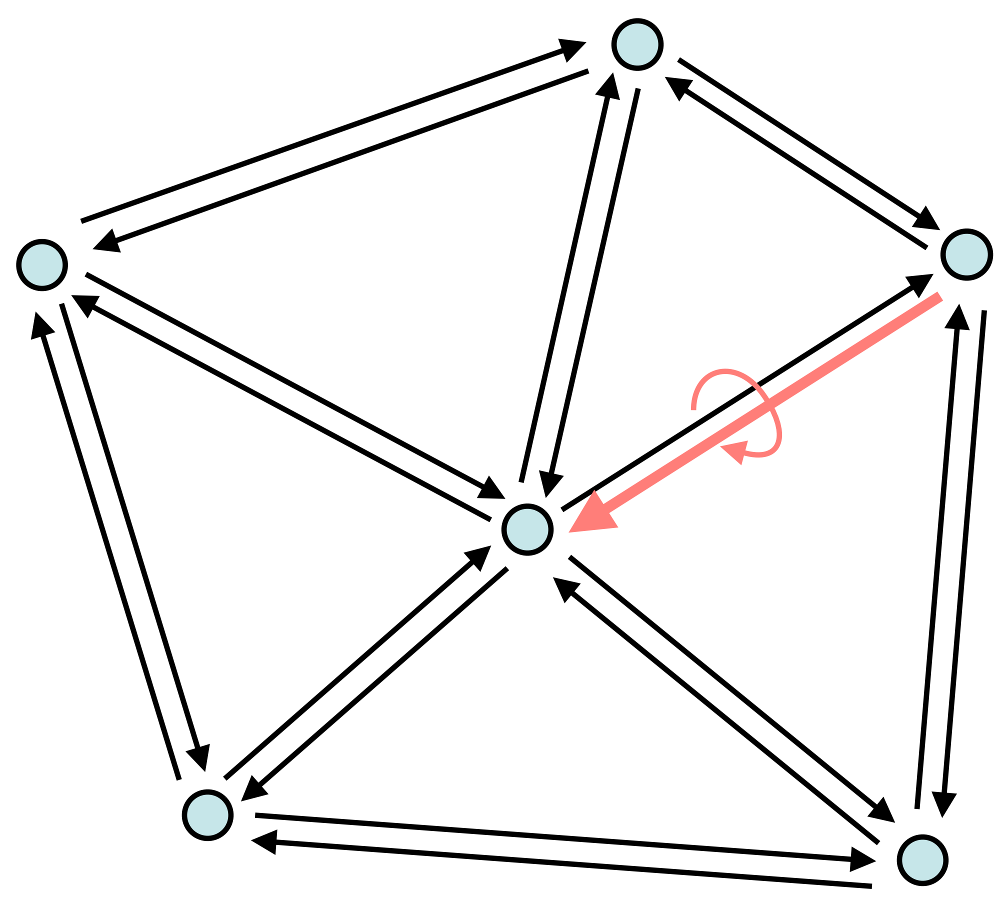
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge



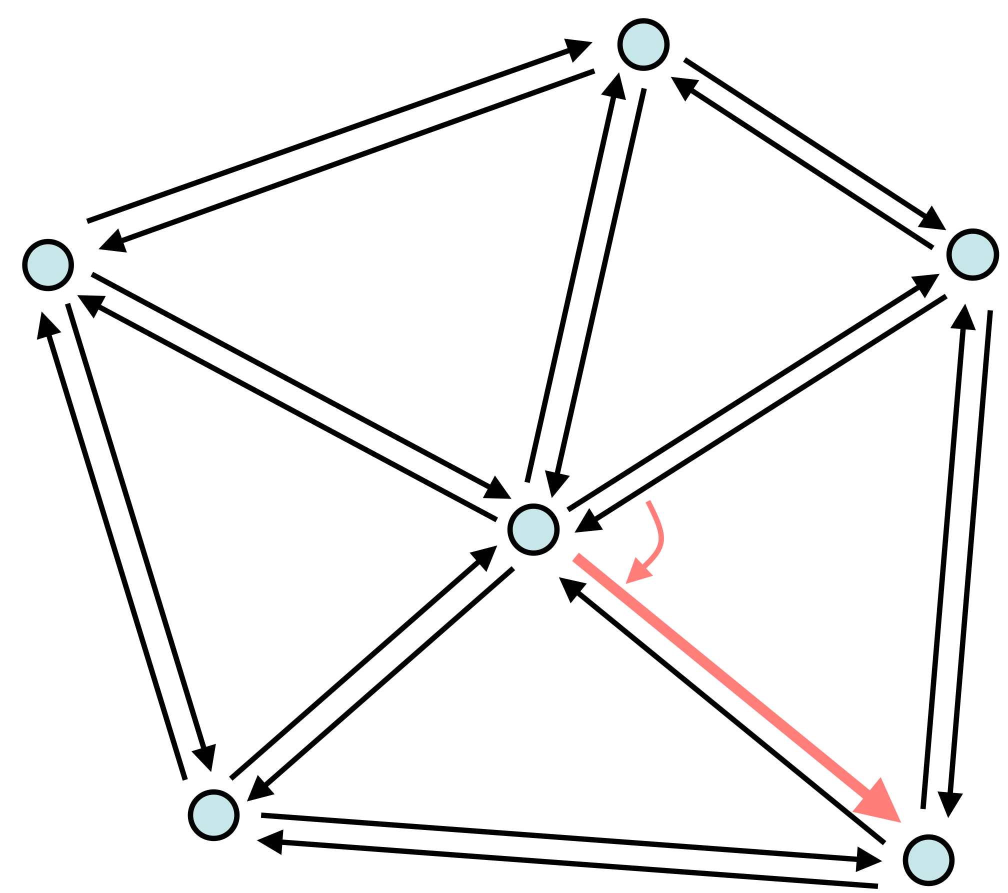
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge
5. Opposite



One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge
5. Opposite
6. Next
7. ...



Halfedge-Based Libraries

- CGAL
  - `www.cgal.org`
  - Computational geometry
  - Free for non-commercial use

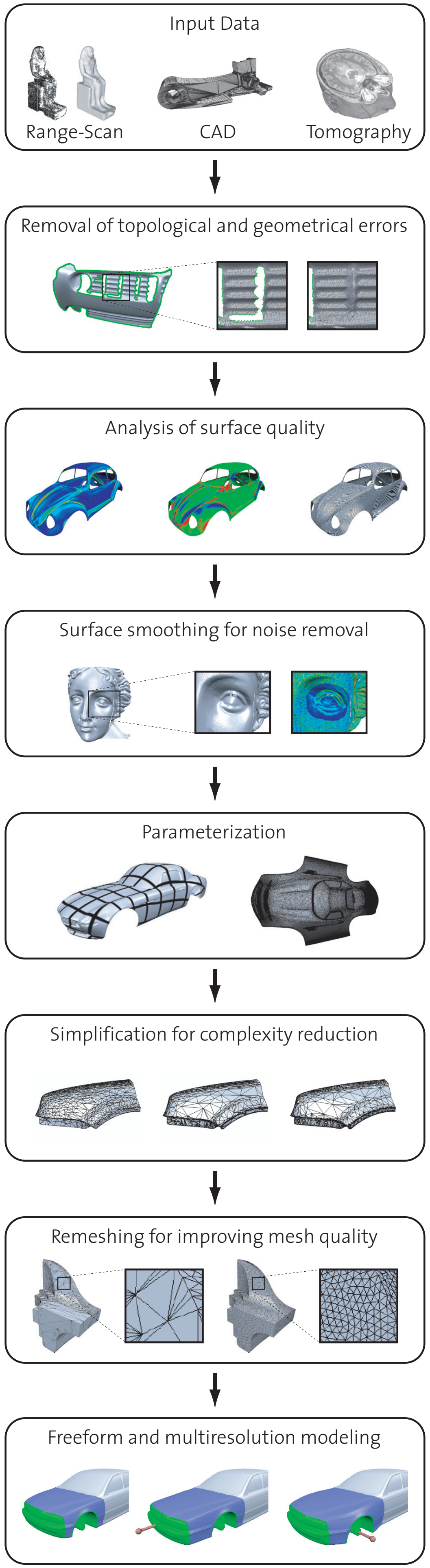
- OpenMesh
  - `www.openmesh.org`
  - Mesh processing
  - Free, LGPL licence

Literature

- Kettner, *Using generic programming for designing a data structure for polyhedral surfaces*, Symp. on Comp. Geom., 1998
- Campagna et al, *Directed Edges - A Scalable Representation for Triangle Meshes*, Journal of Graphics Tools 4(3), 1998
- Botsch et al, *OpenMesh - A generic and efficient polygon mesh data structure*, OpenSG Symp. 2002

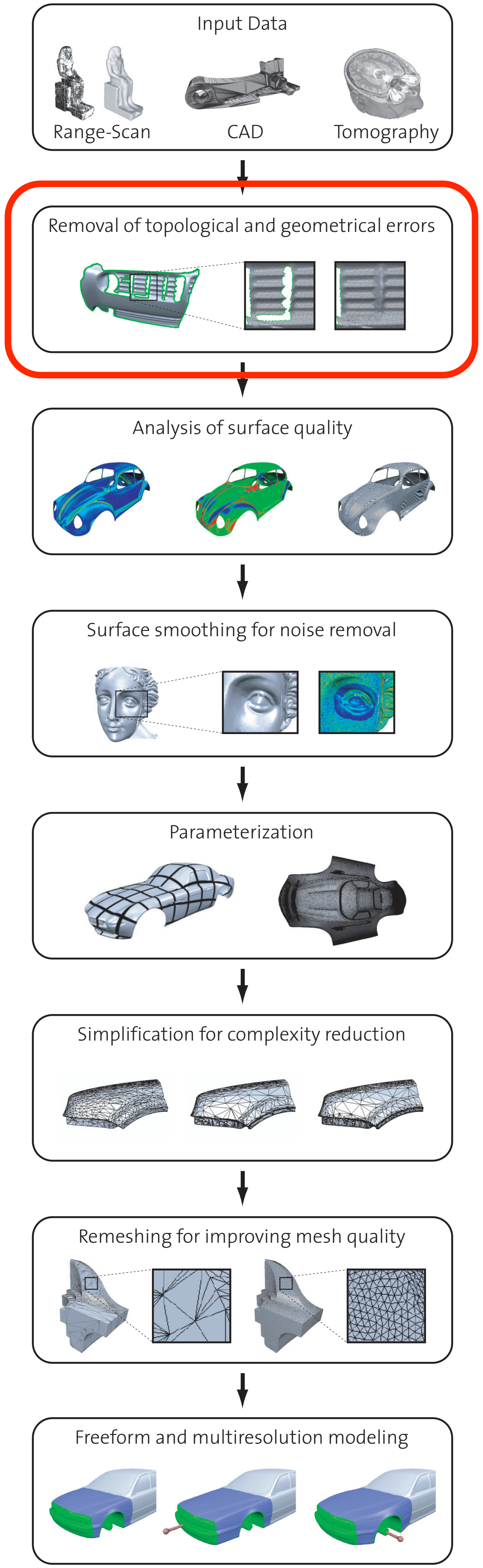
Outline

- (mathematical) geometry representations
  - parametric vs. implicit
- approximation properties
- types of operations
  - distance queries
  - evaluation
  - modification / deformation
- data structures



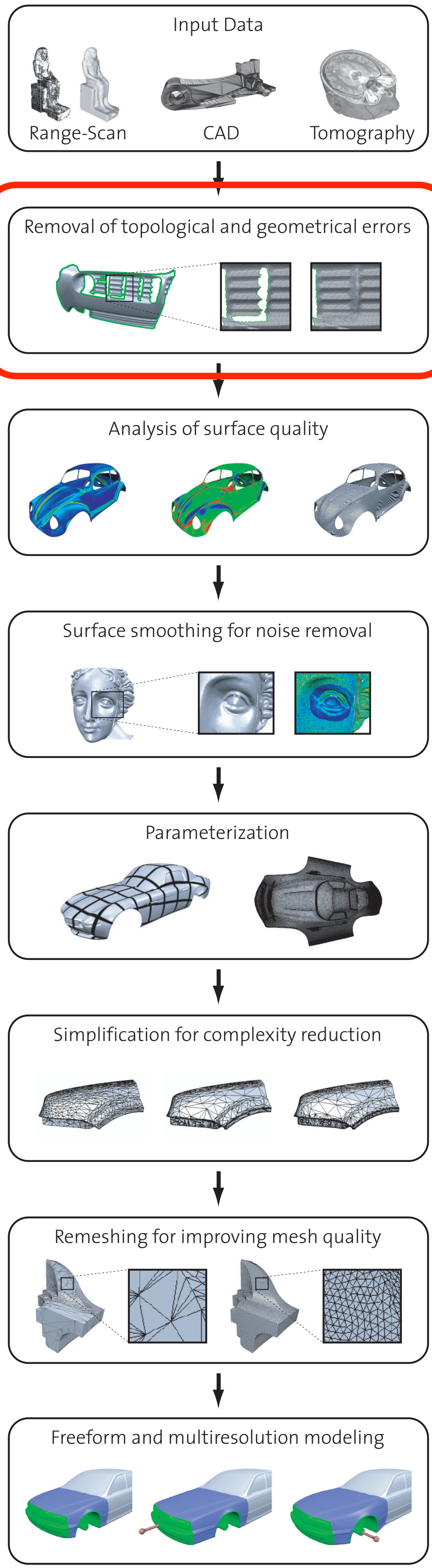
Model Repair

Stephan Bischoff
RWTH Aachen University



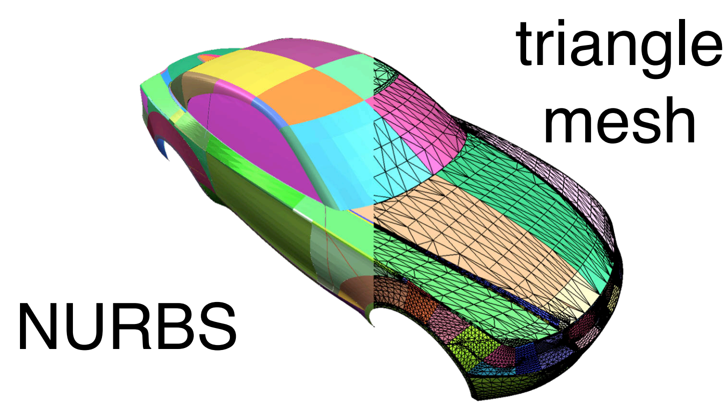
Model Repair

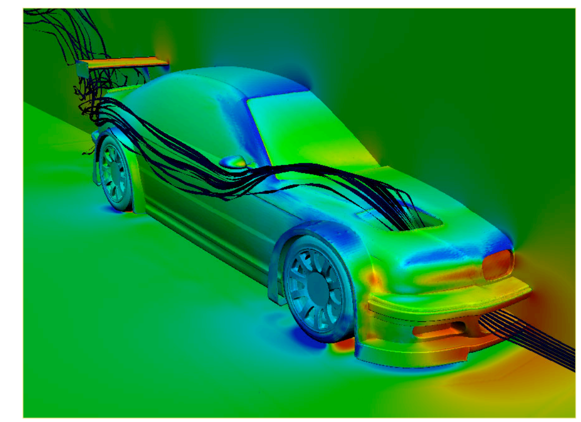
- Model repair is the removal of artifacts from a geometric model such that it becomes suitable for further processing.
- Typically: Produce a nice, manifold triangle mesh
  - with boundary or
  - without boundary (watertight)

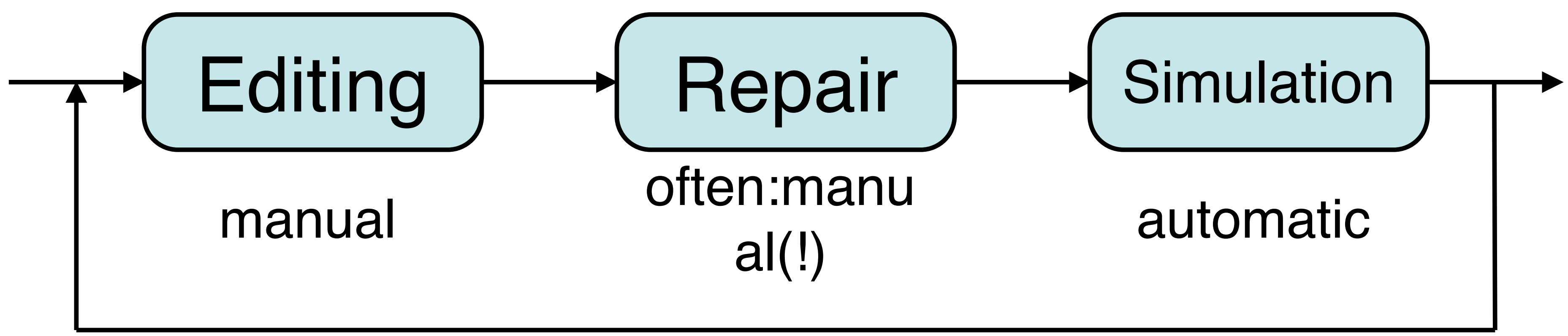


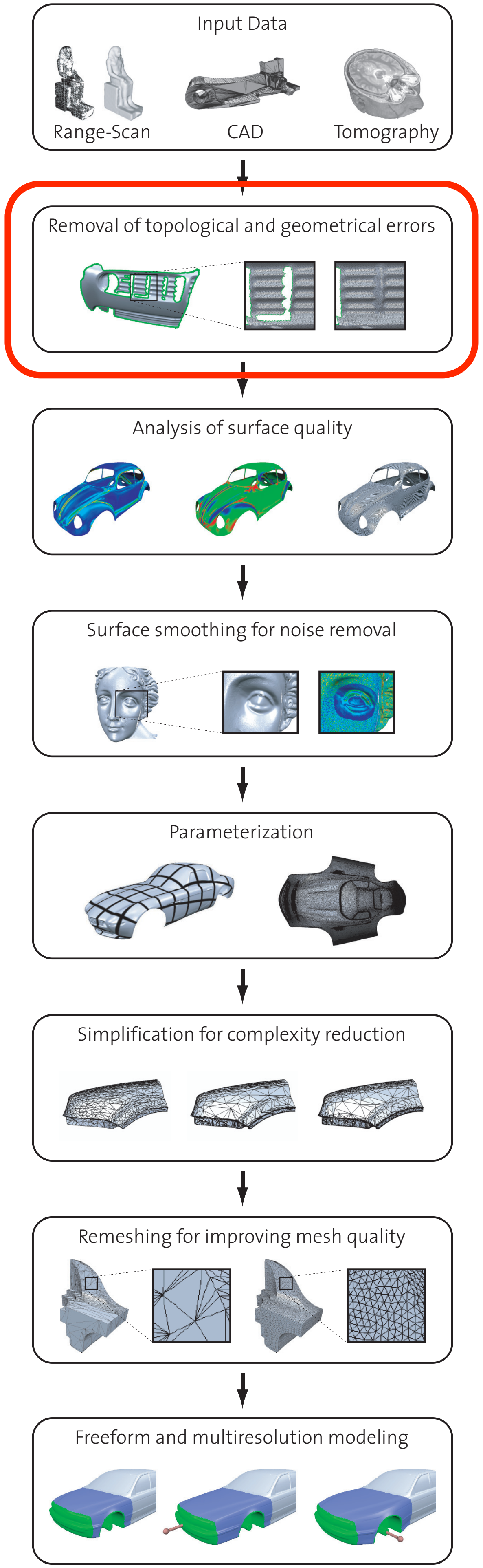
Model Repair

- Impact e.g. in CAD/CAM:







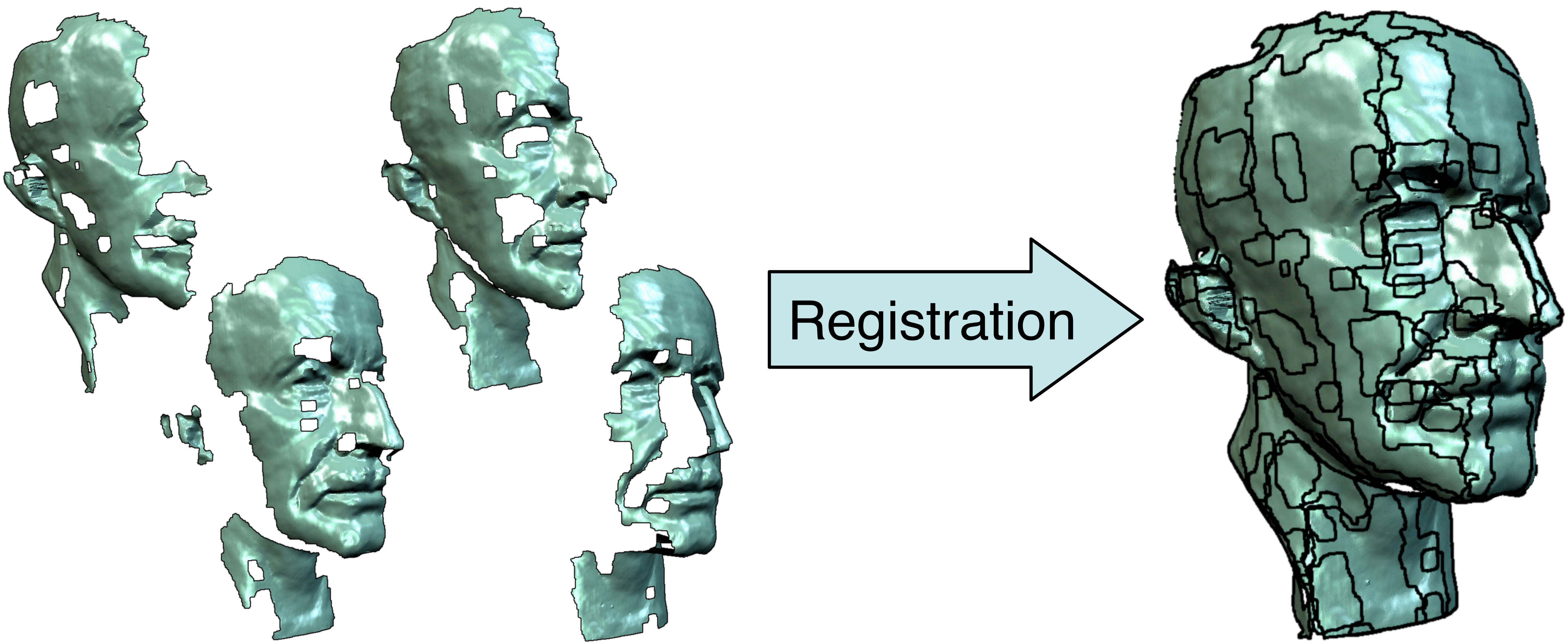


Model Repair

- **Types of input**
- Surface-oriented algorithms
  - Filling holes in meshes [Liepa 2003]
- Volumetric algorithms
  - Simplification and repair of polygonal models using volumetric techniques [Nooruddin and Turk 2003]
  - Automatic restoration of polygon models [Bischoff, Pavic, Kobbelt 2005]
- Conclusion & outlook

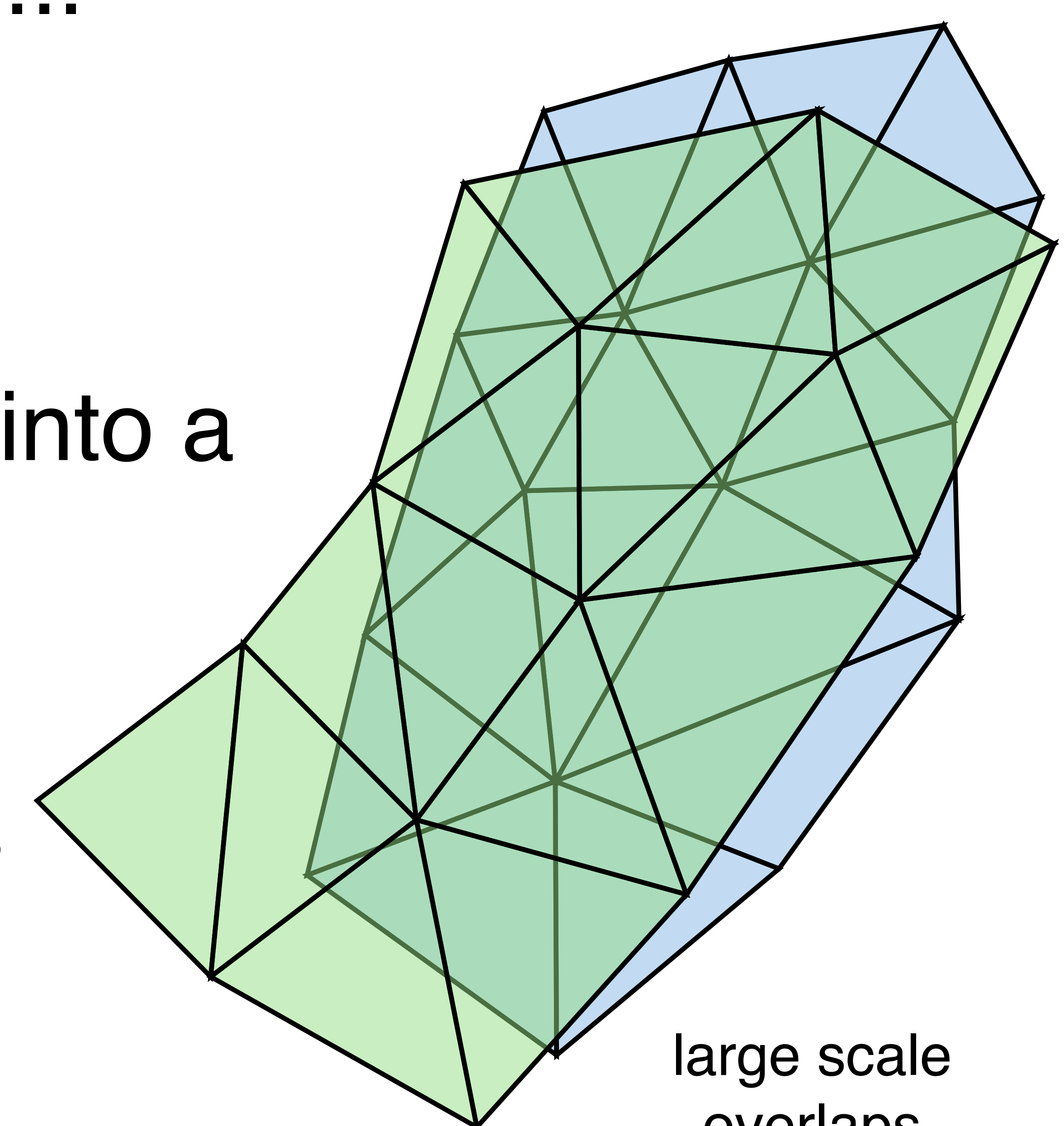
Registered Range Images

- Registered range images are a set of patches that describe different parts of an object.



Registered Range Images

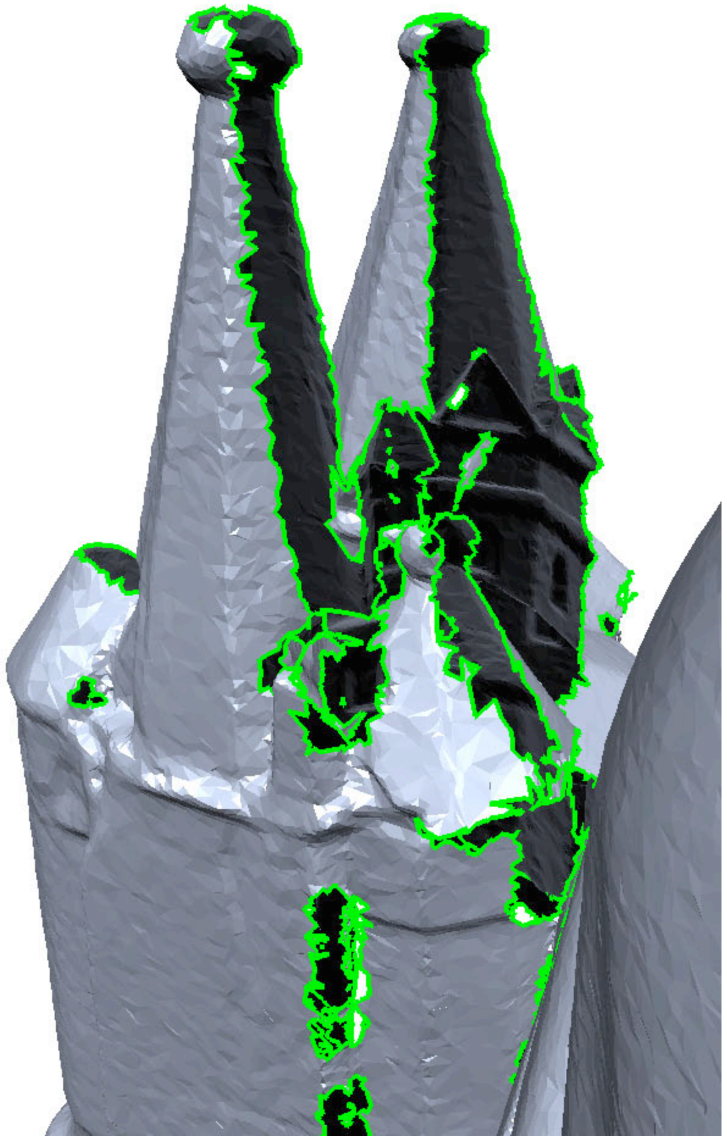
- Large areas of overlap are ...
  - ... good for registration but
  - ... bad for repair
- How to merge the patches into a single mesh?
  - Inconsistent geometry
  - Incompatible connectivities

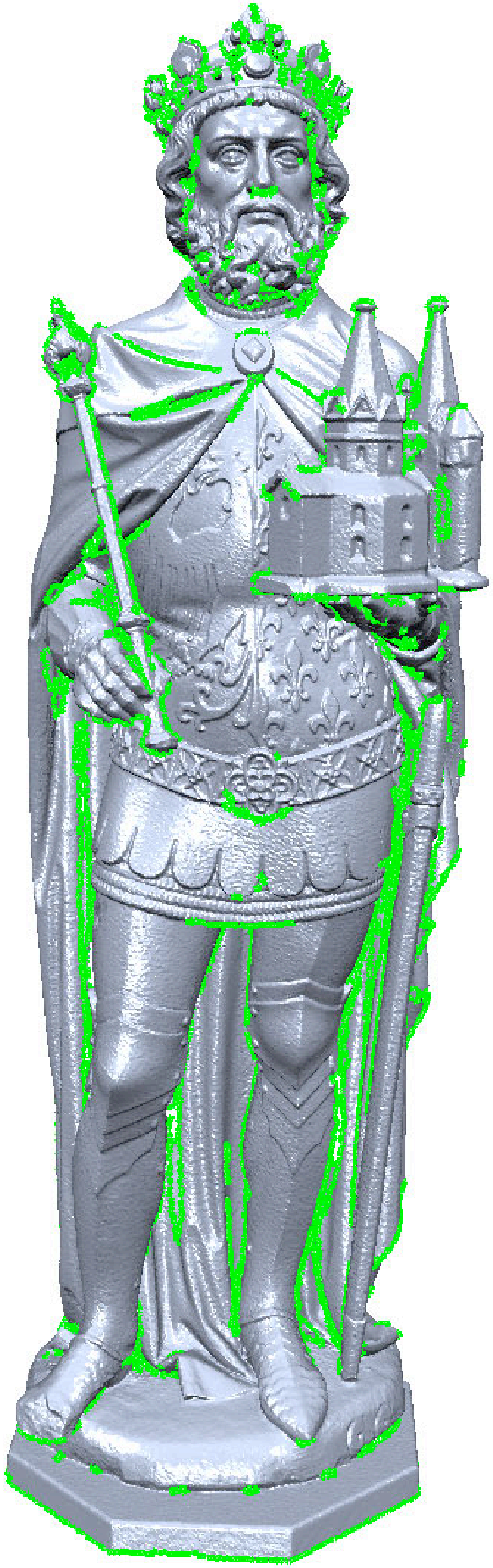


large scale overlaps

Fused Range Images

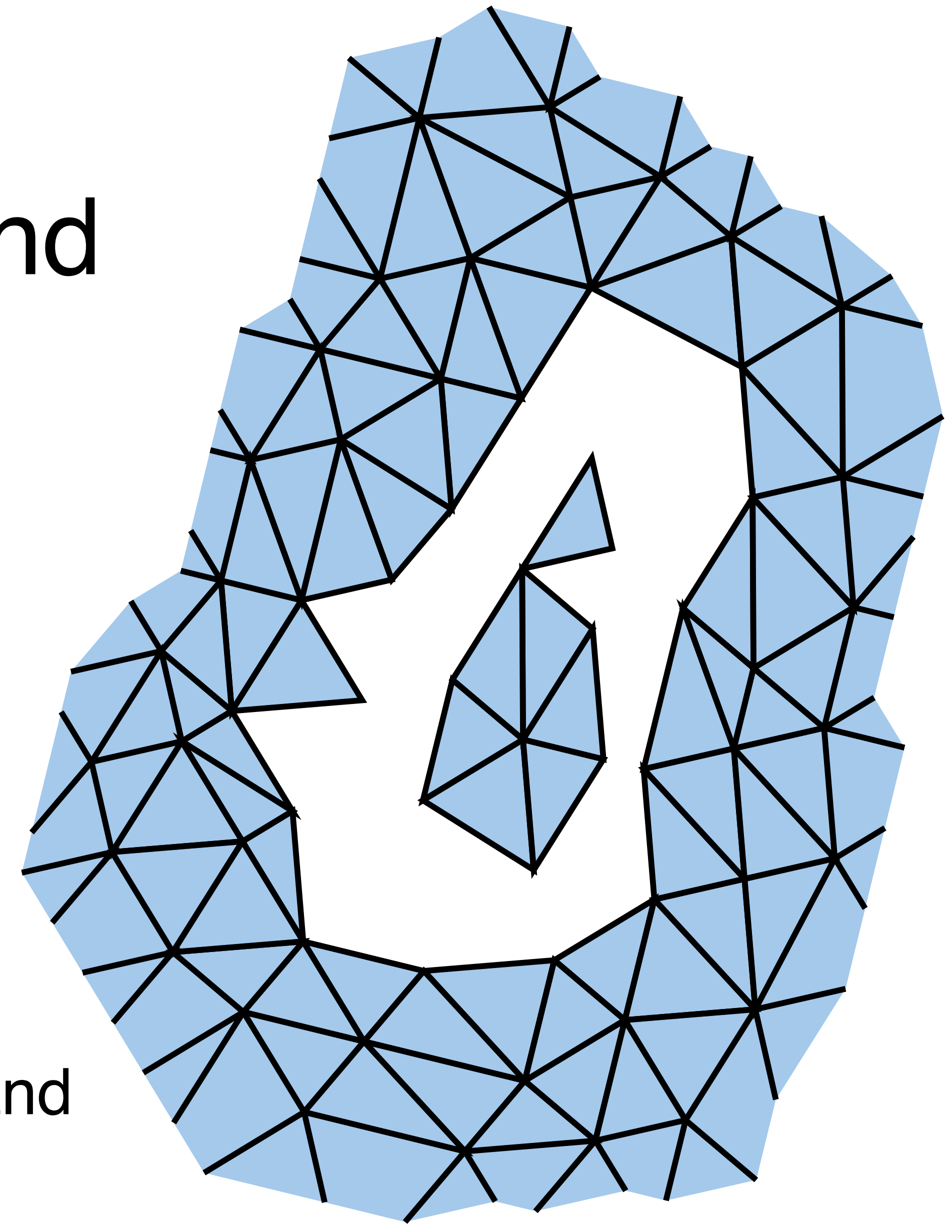
- Fused range images are manifold meshes with holes and isles (i.e. boundaries)





Fused Range Images

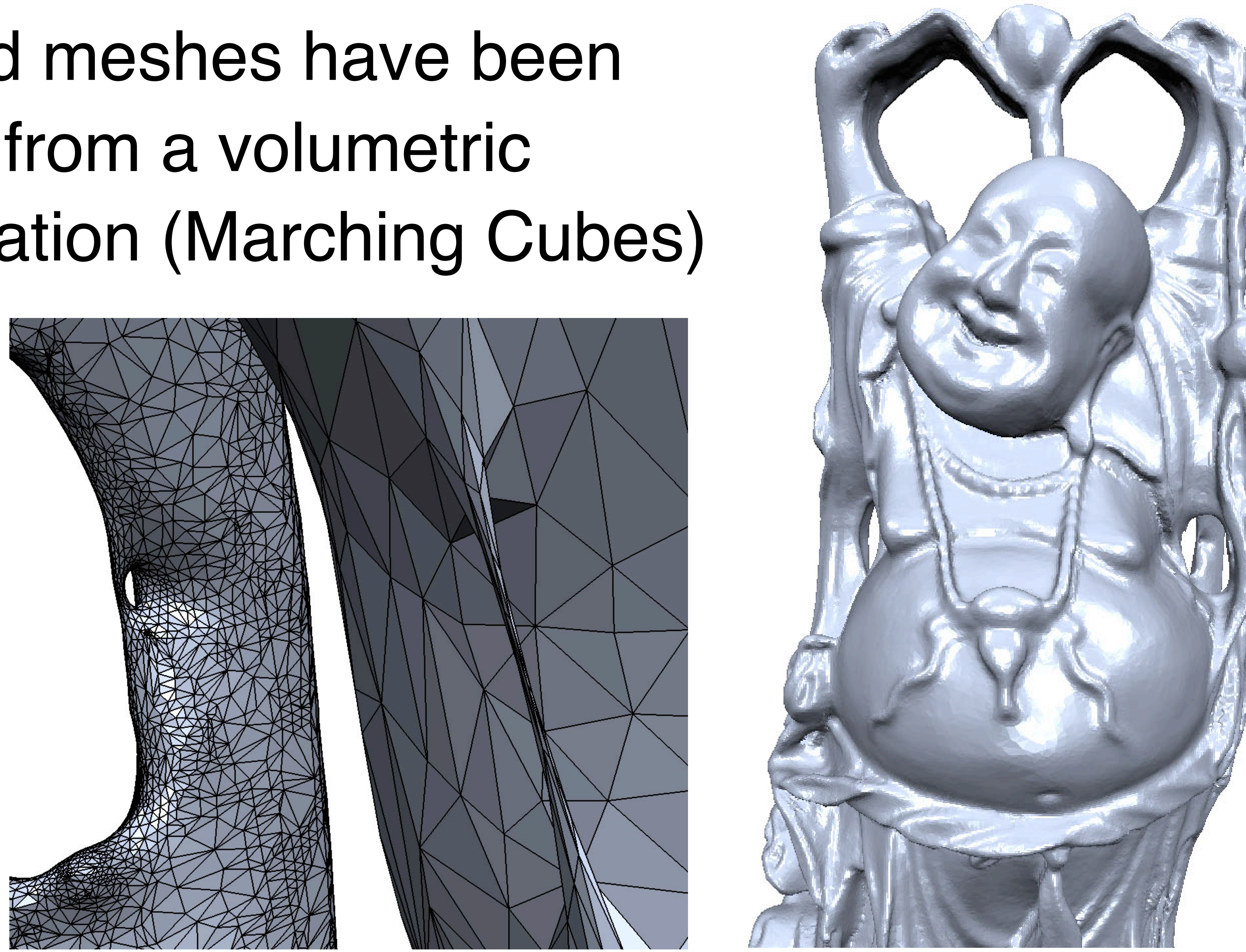
- Holes and isles due to obstructions in the line of sight of the scanner
- Identify corresponding holes and isles
- Fill holes
  - Smoothly
  - Geometry transfer/synthesis
- Avoid intersections



holes and
isles

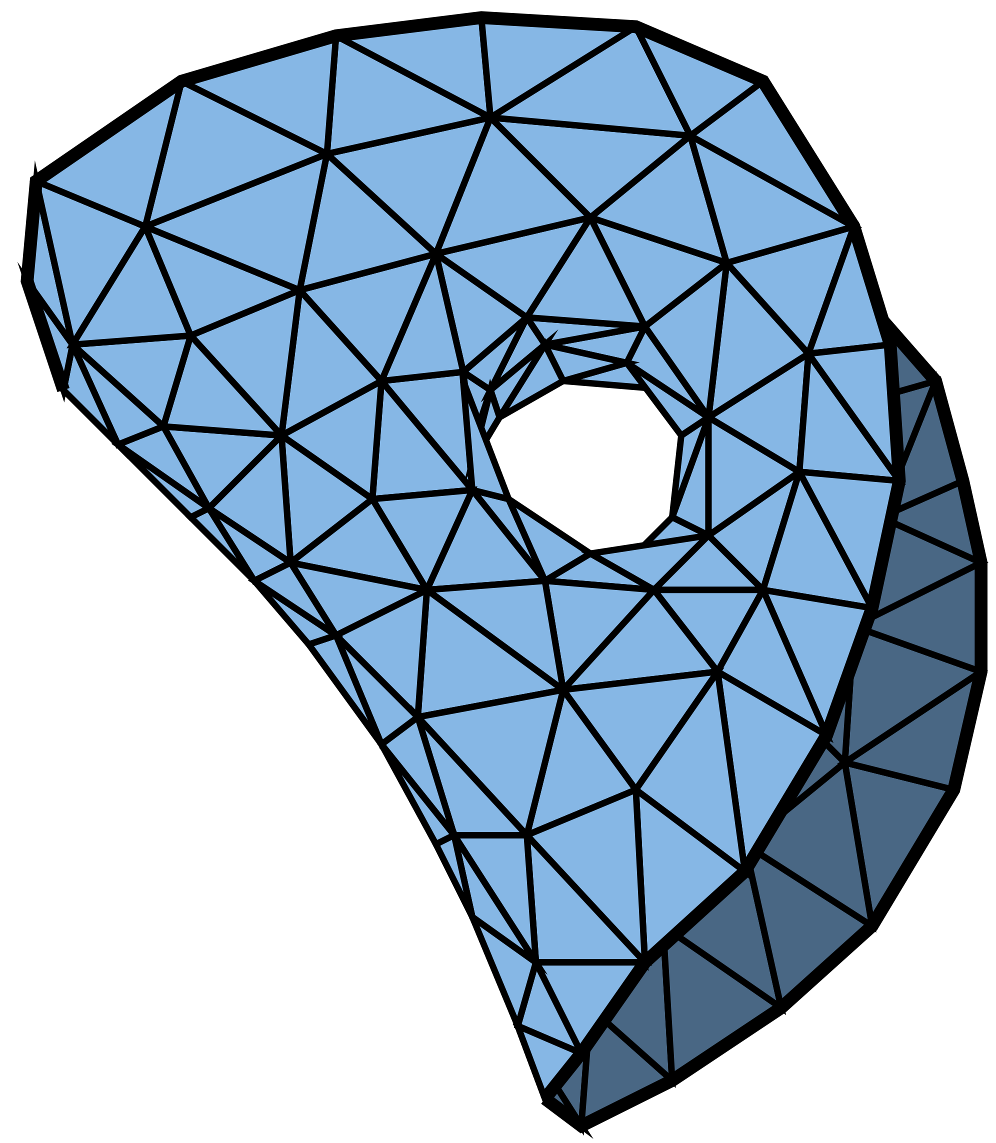
Contoured Meshes

- Contoured meshes have been extracted from a volumetric representation (Marching Cubes)



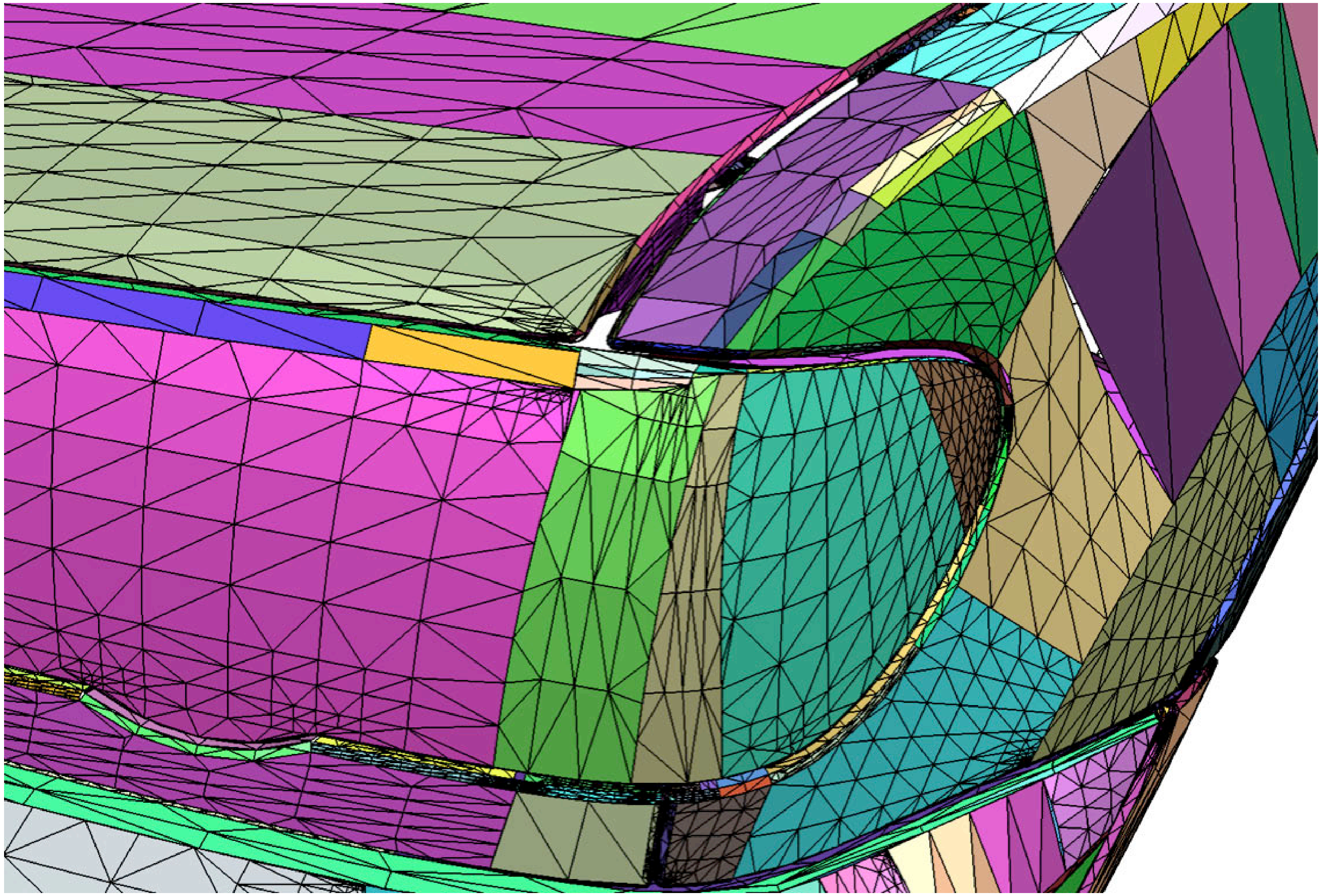
Contoured Meshes

- Contoured meshes are usually manifold, but contain topological noise
  - Handles
  - (Cavities)
  - (Disconnected components)



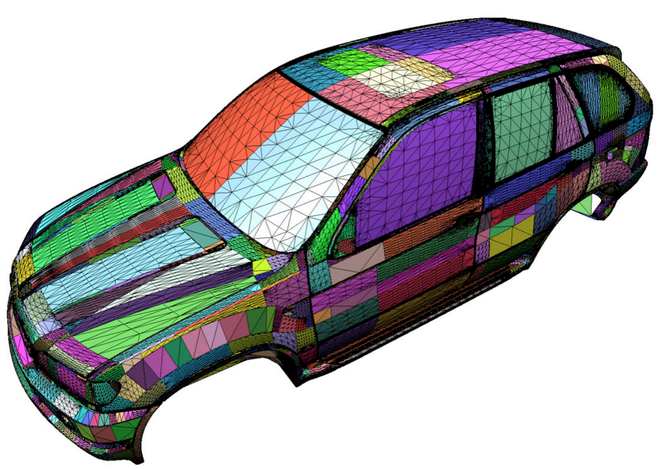
Triangulated NURBS

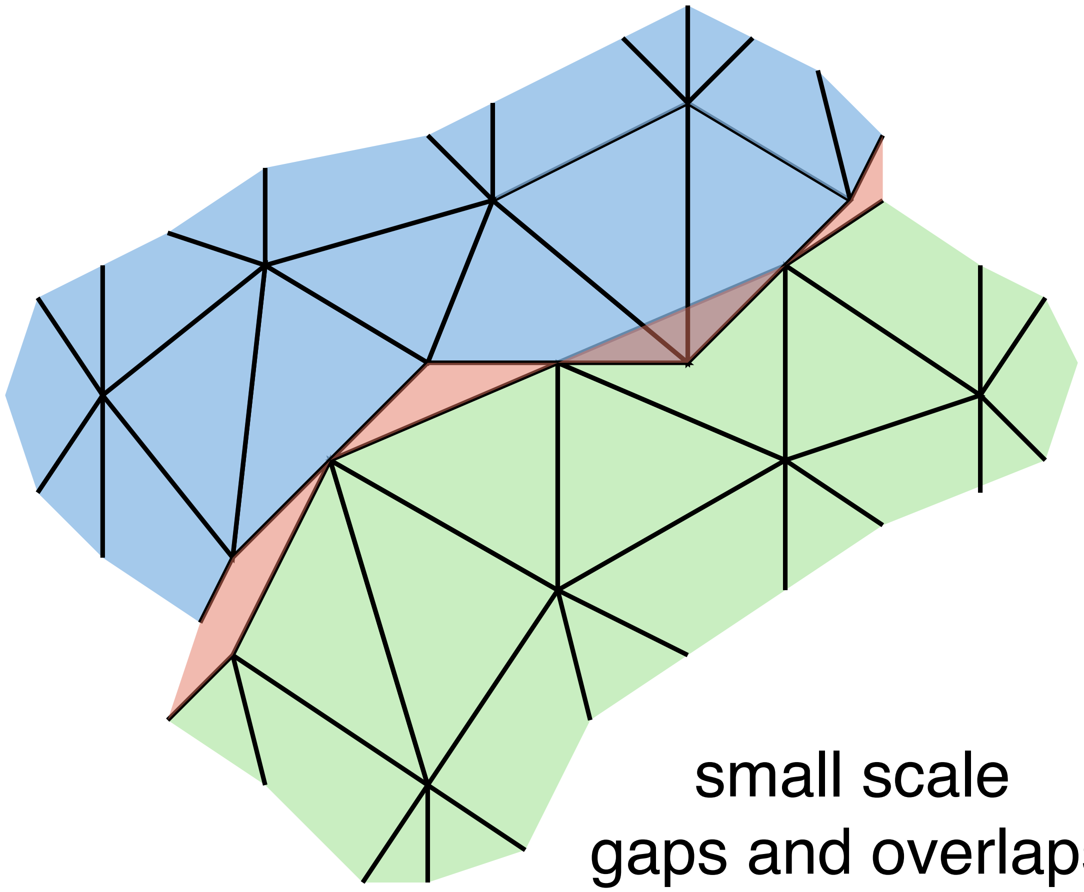
- Set of patches that contain small scale gaps and overlaps



Triangulated NURBS

- Gaps and overlaps are due to triangulating a common patch boundary differently from both sides
- Issues
  - Orientation
  - Structure preservation

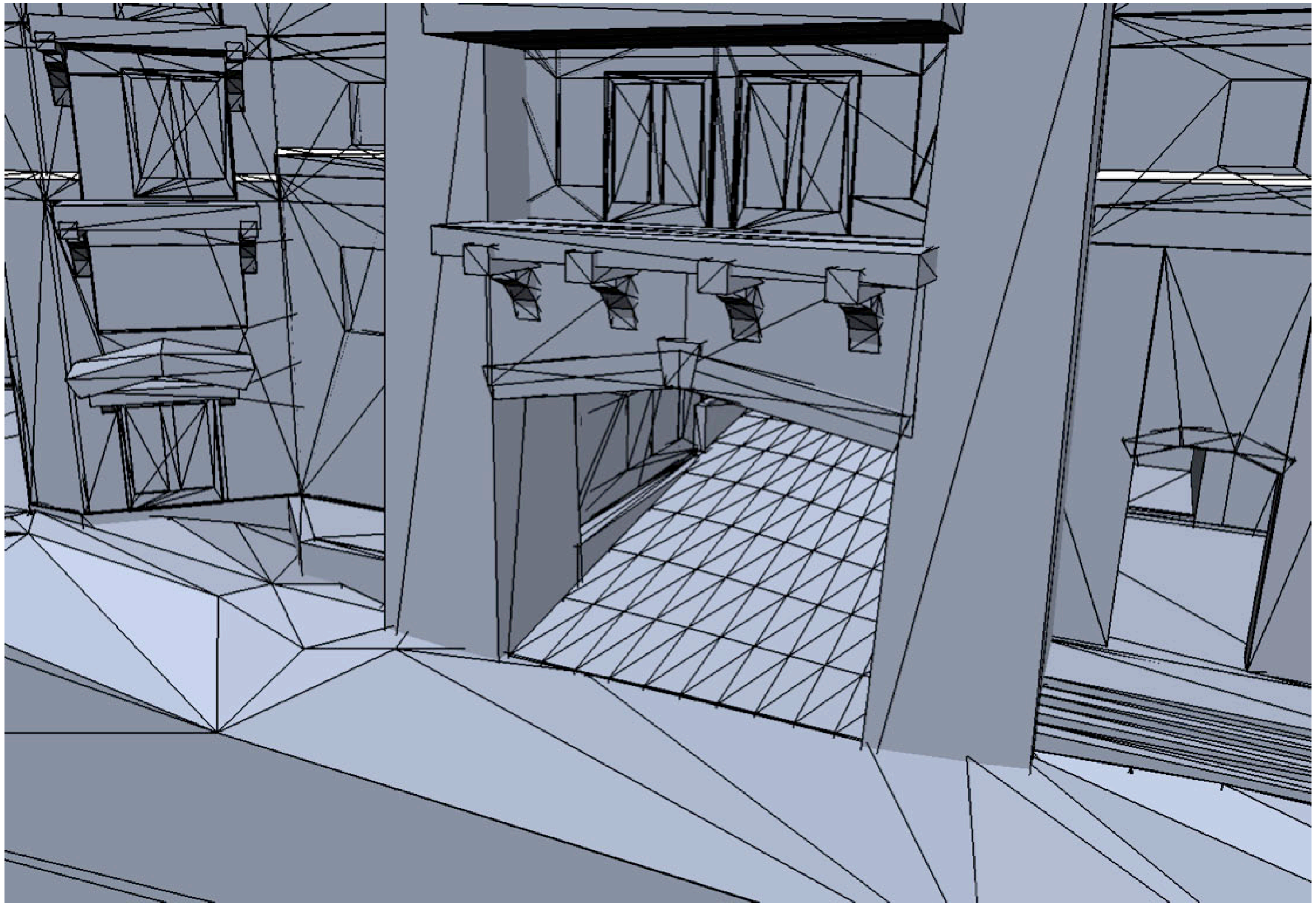




small scale
gaps and overlaps

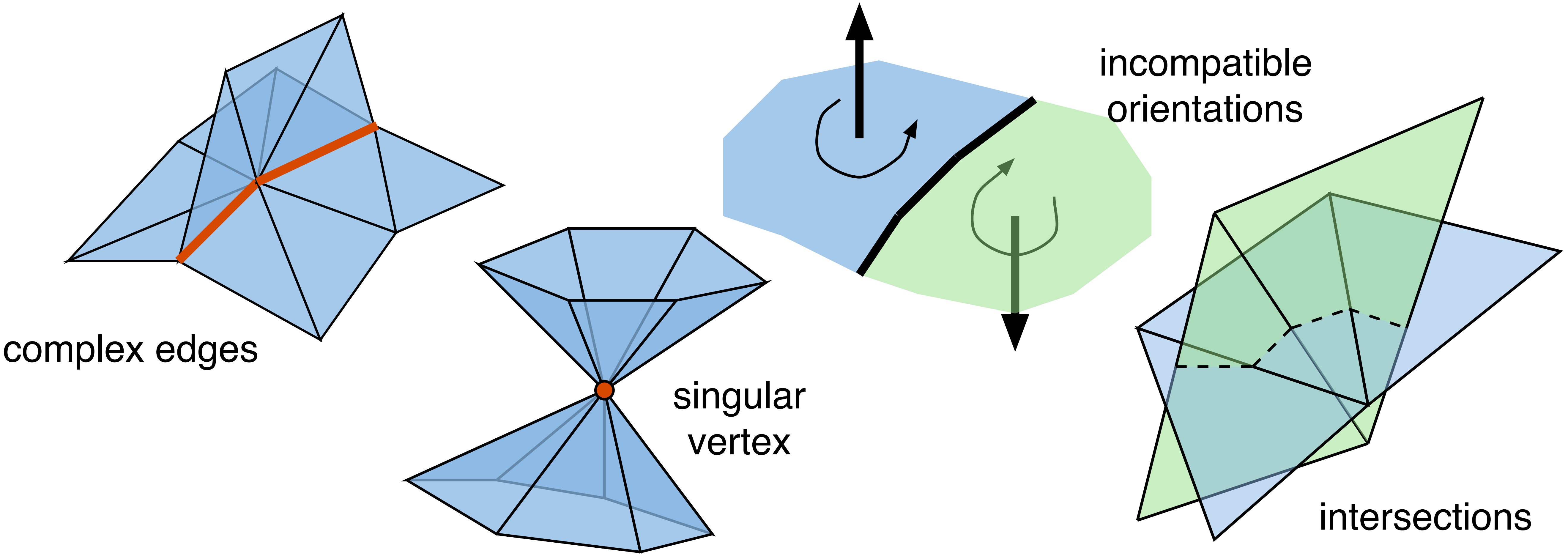
Triangle Soups

- A triangle soup is a set of triangles without connectivity information



Triangle Soups

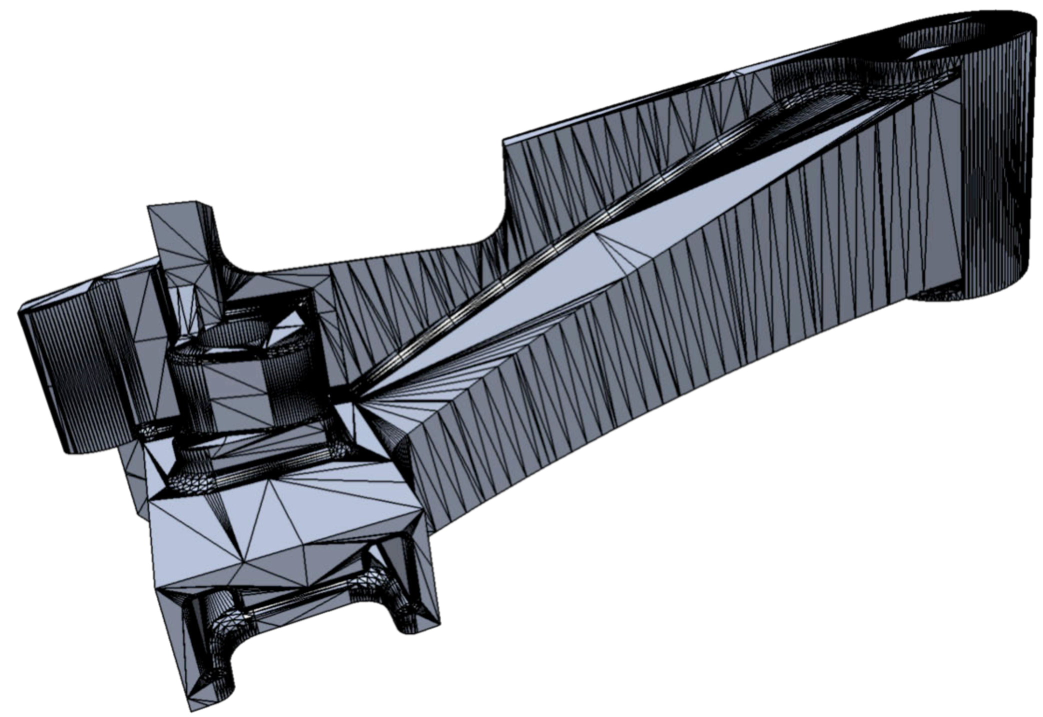
- Ok for visualization but bad for downstream applications that require manifold meshes
- In addition to the artifacts we already covered, ...

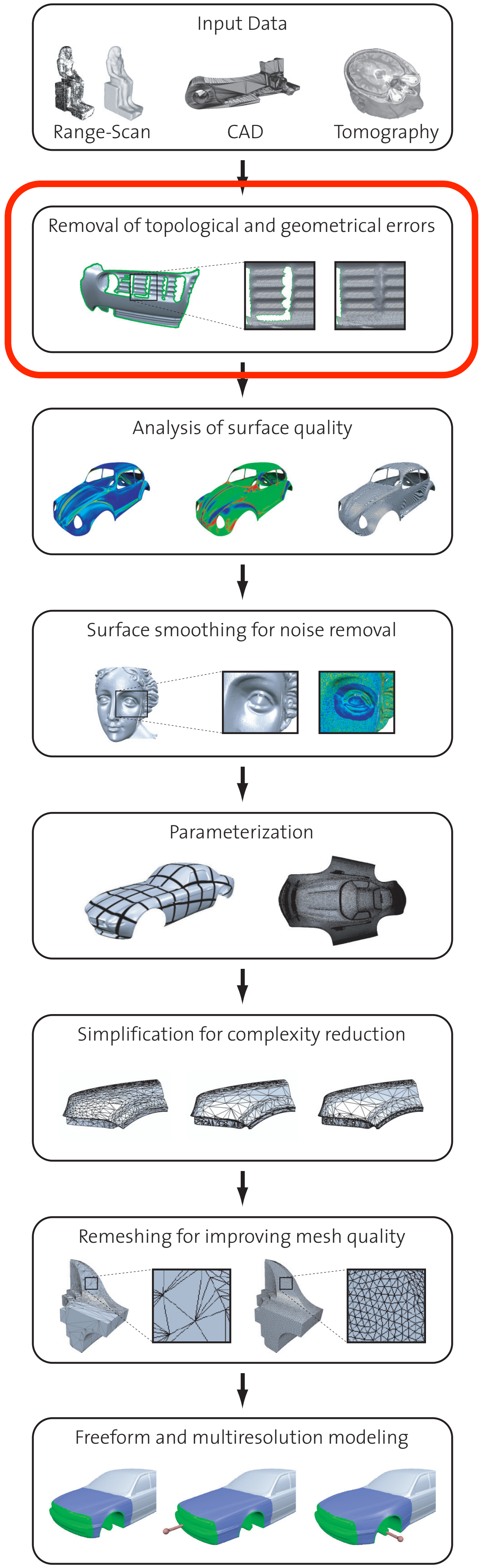


Not Covered ...

- Geometrical noise
  ➞ Smoothing (Christian)

- Badly meshed manifolds
  ➞ Remeshing (Leif)



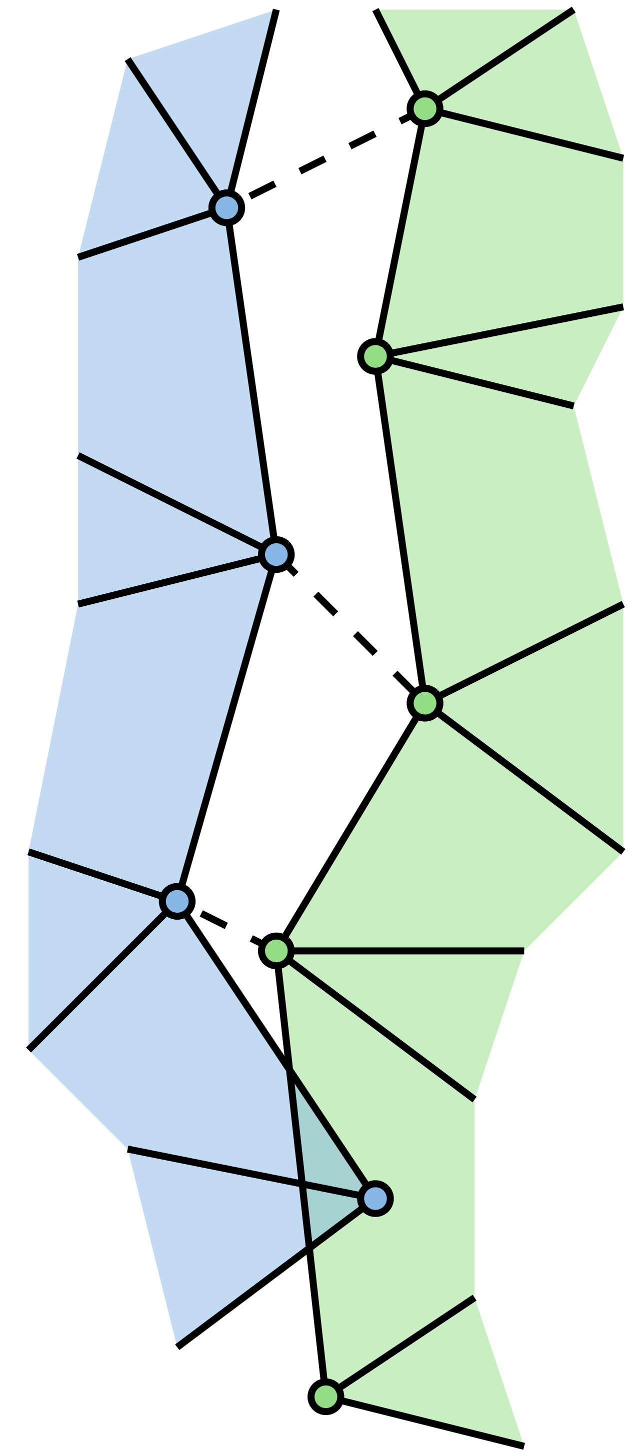


Model Repair

- Types of input
- **Surface-oriented algorithms**
  - Filling holes in meshes [Liepa 2003]
- Volumetric algorithms
  - Simplification and repair of polygonal models using volumetric techniques [Nooruddin and Turk 2003]
  - Automatic restoration of polygon models [Bischoff, Pavic, Kobbelt 2005]
- Conclusion & outlook

Surface-oriented Algorithms

- Surface oriented approaches explicitly identify and resolve artifacts
- Methods
  - Snapping
  - Splitting
  - Stitching
  - ...

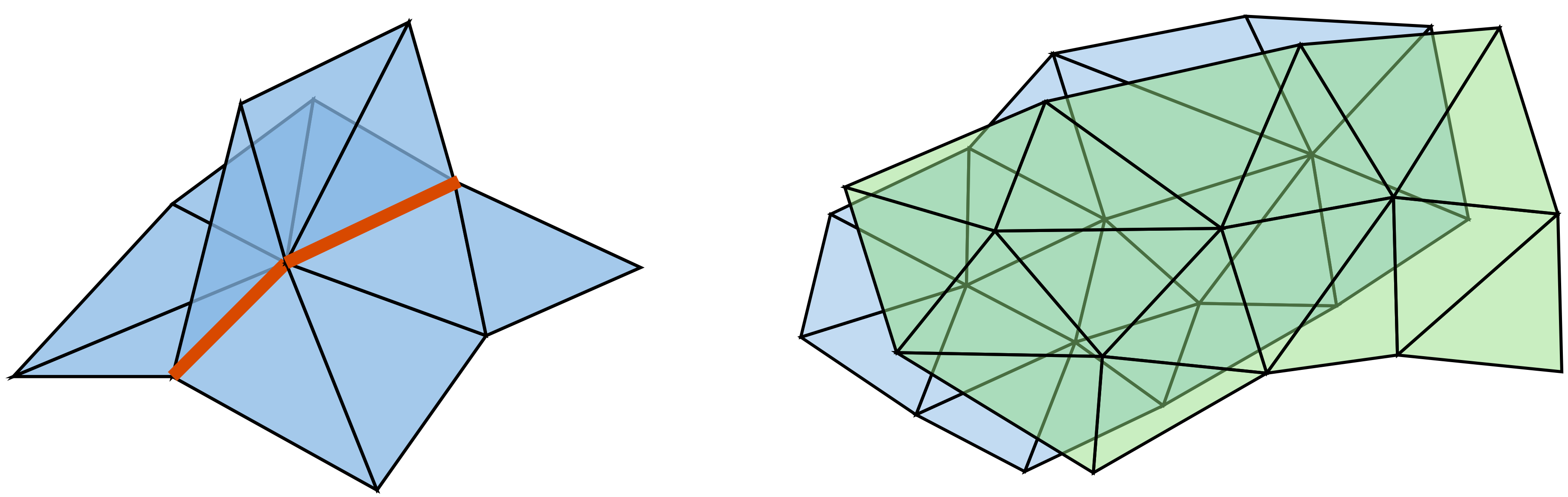


Surface-oriented Algorithms

- Advantages
  - Fast
  - Memory friendly
  - Structure preserving, minimal modification of the input
  - Conceptually easier than volumetric algorithms

Surface-oriented Algorithms

- Problems
  - Not robust
    - Numerical issues ➞ use infinite precision arithmetic
    - Inherent non-robustness



  - No quality guarantees on the output

Example Algorithm

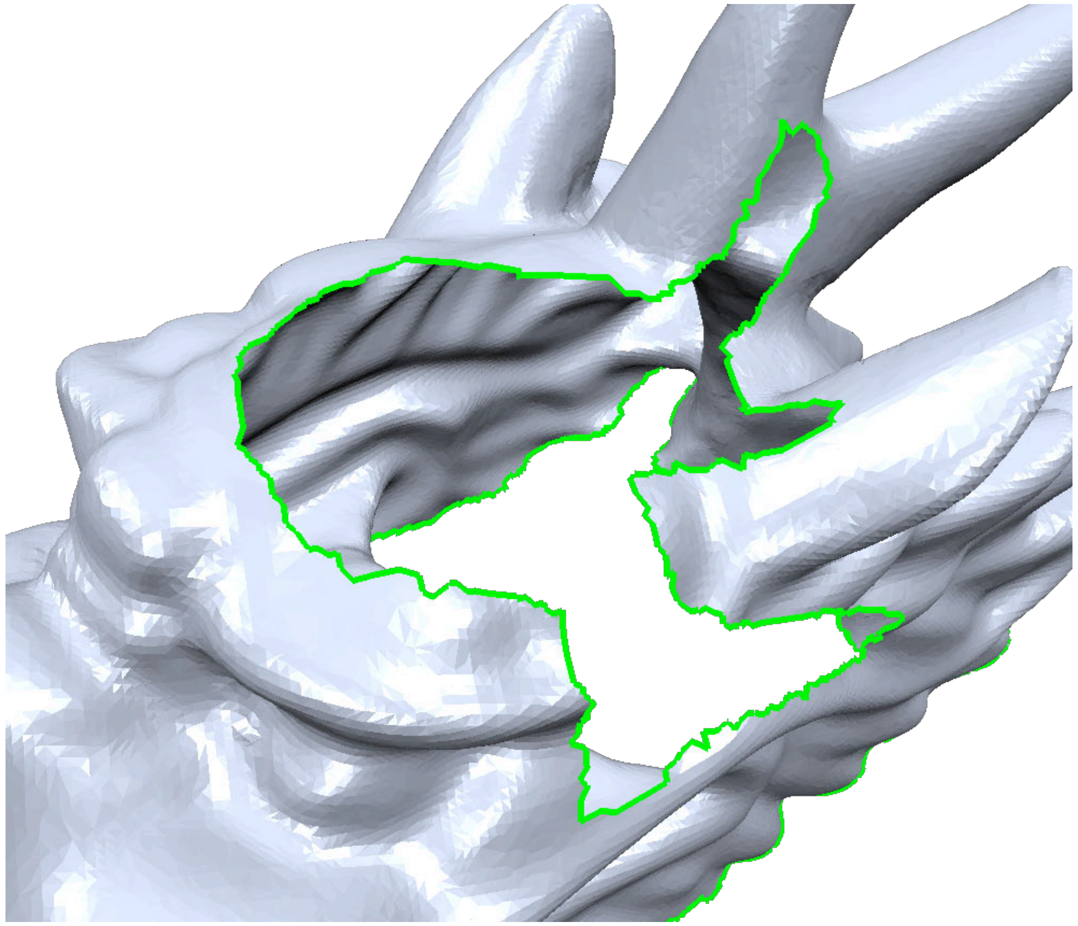
- Algorithm for filling holes

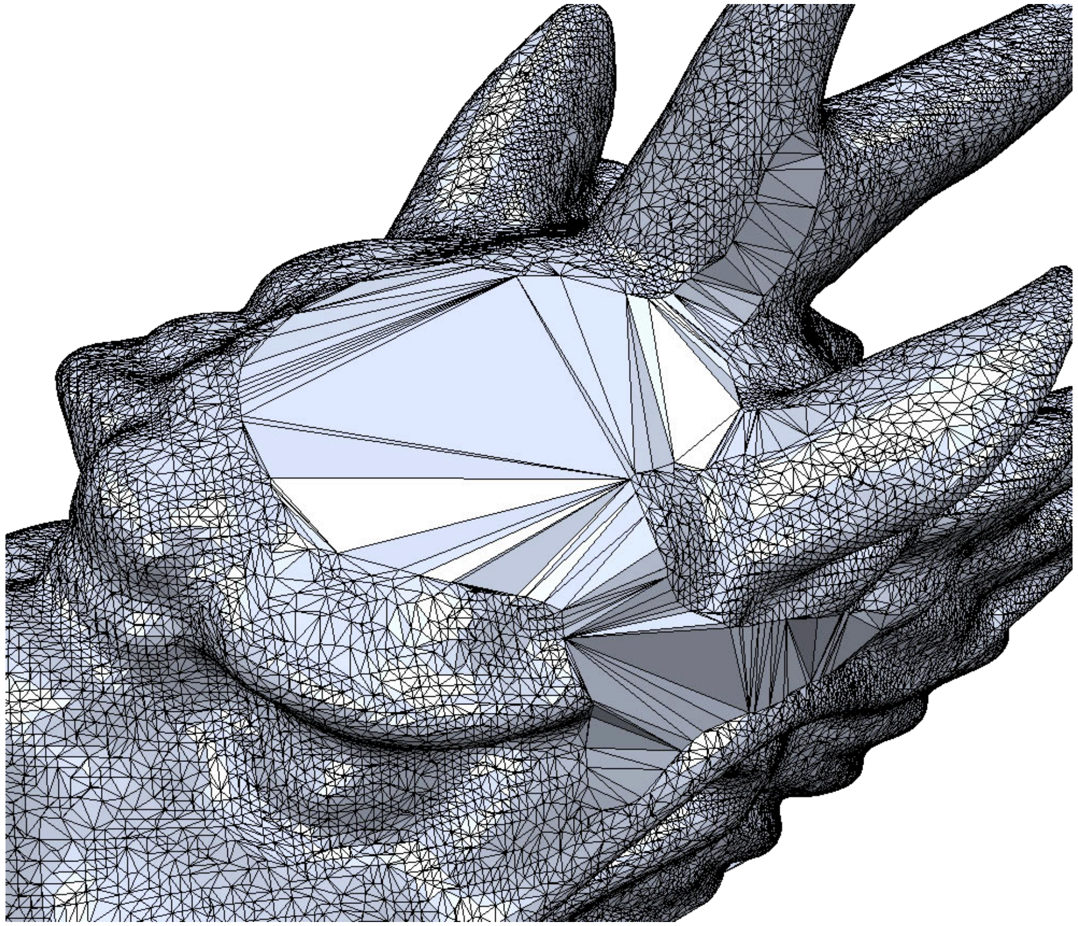
  Peter Liepa
  Filling Holes in Meshes
  In Proc. Symposium on Geometry Processing 2003

- Three stages
  1. Compute a coarse triangulation T to fill the hole
  2. Refine the triangulation, T → T’, to match the vertex densities of the surrounding area
  3. Smooth the triangulation T’ to match the geometry of the surrounding

Filling Holes in Meshes - 1

- Compute a coarse triangulation T

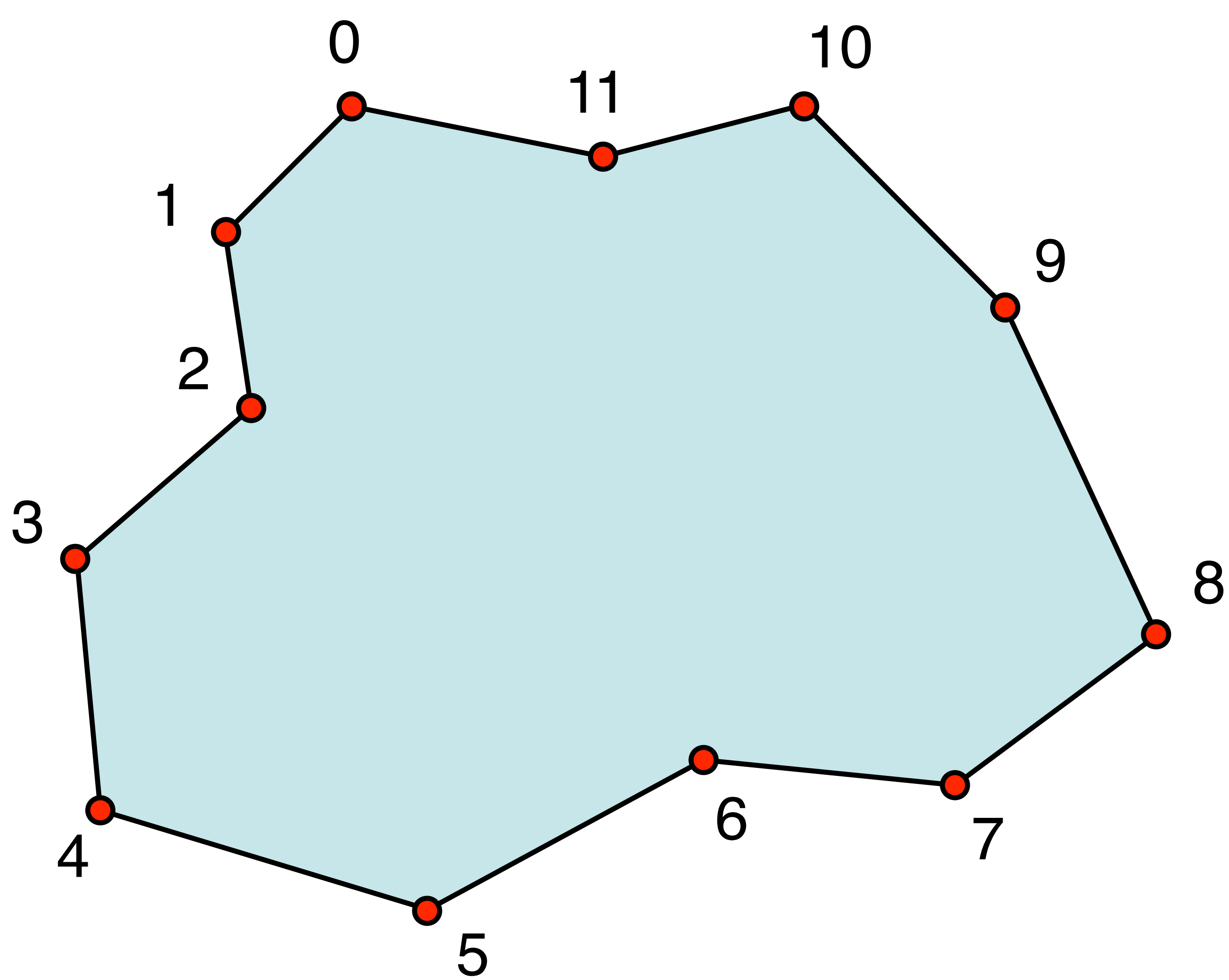




Filling Holes in Meshes - 1

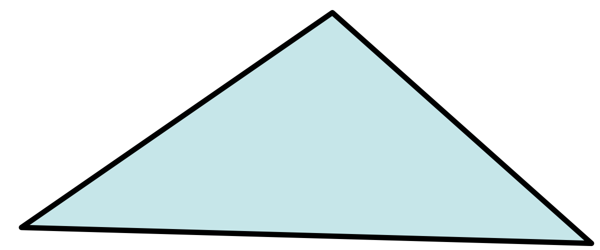
- Compute a triangulation T of minimal weight w(T)

n vertices,
n–2 triangles

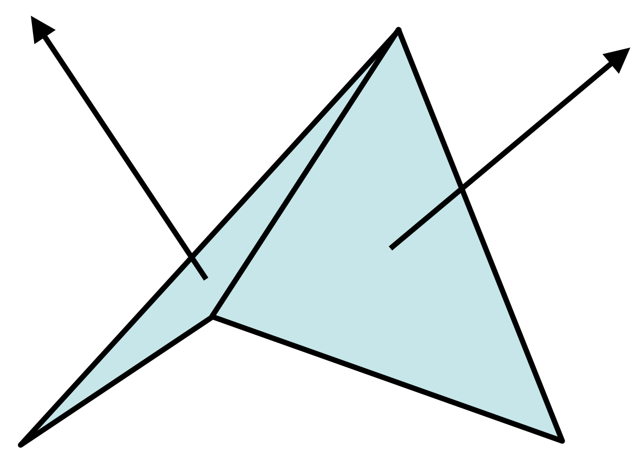


Filling Holes in Meshes - 1

- Weight w(T) is a mixture of
  - area(T) = $\sum_{\Delta \in T} \text{area}(\Delta)$



  - maximum dihedral angle in T

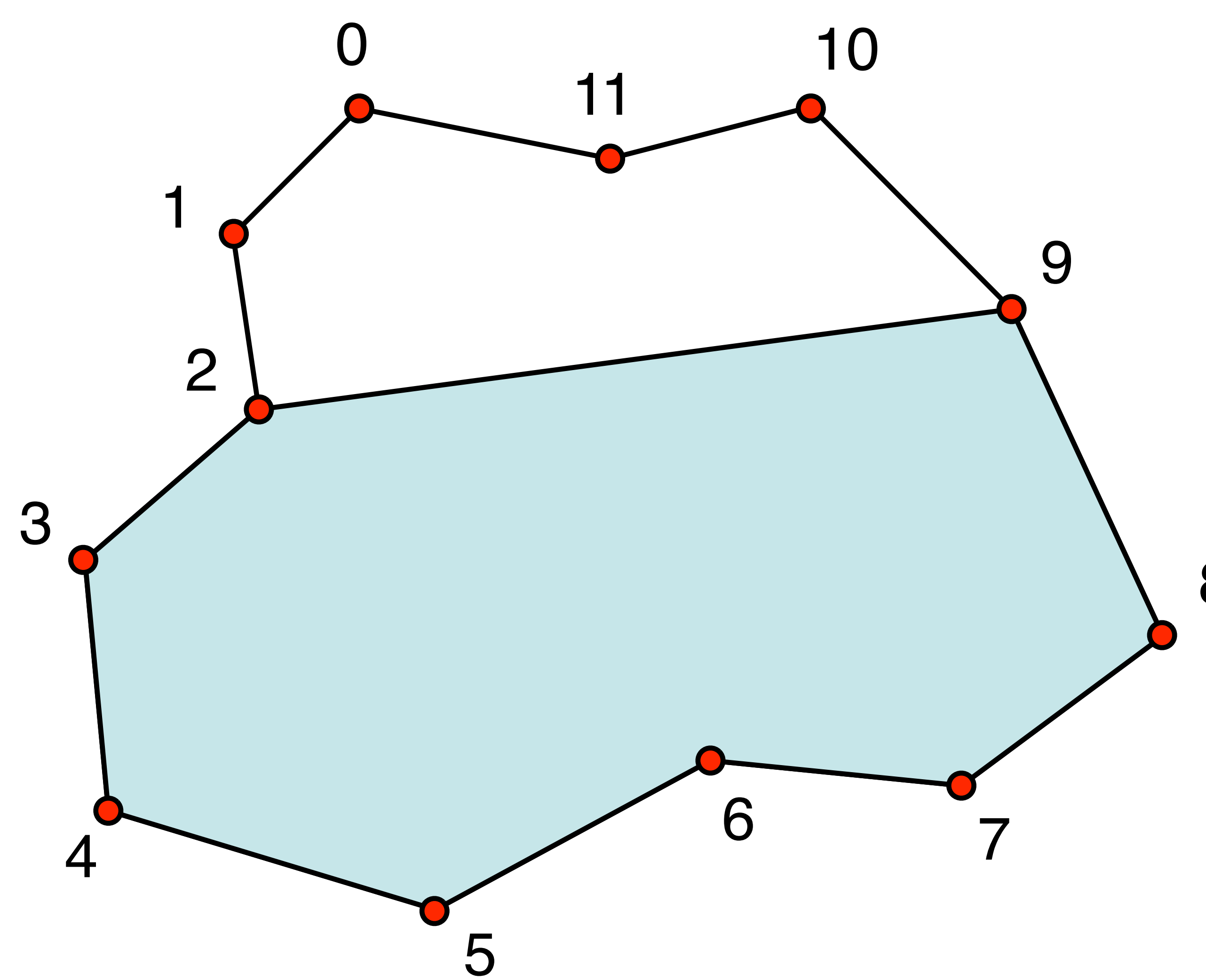


- Thus, we favour triangulations of low area and low normal variation

Filling Holes in Meshes - 1

- Let w[a,c] be the minimal weight that can be achieved in triangulating the polygon a,a+1,...,c

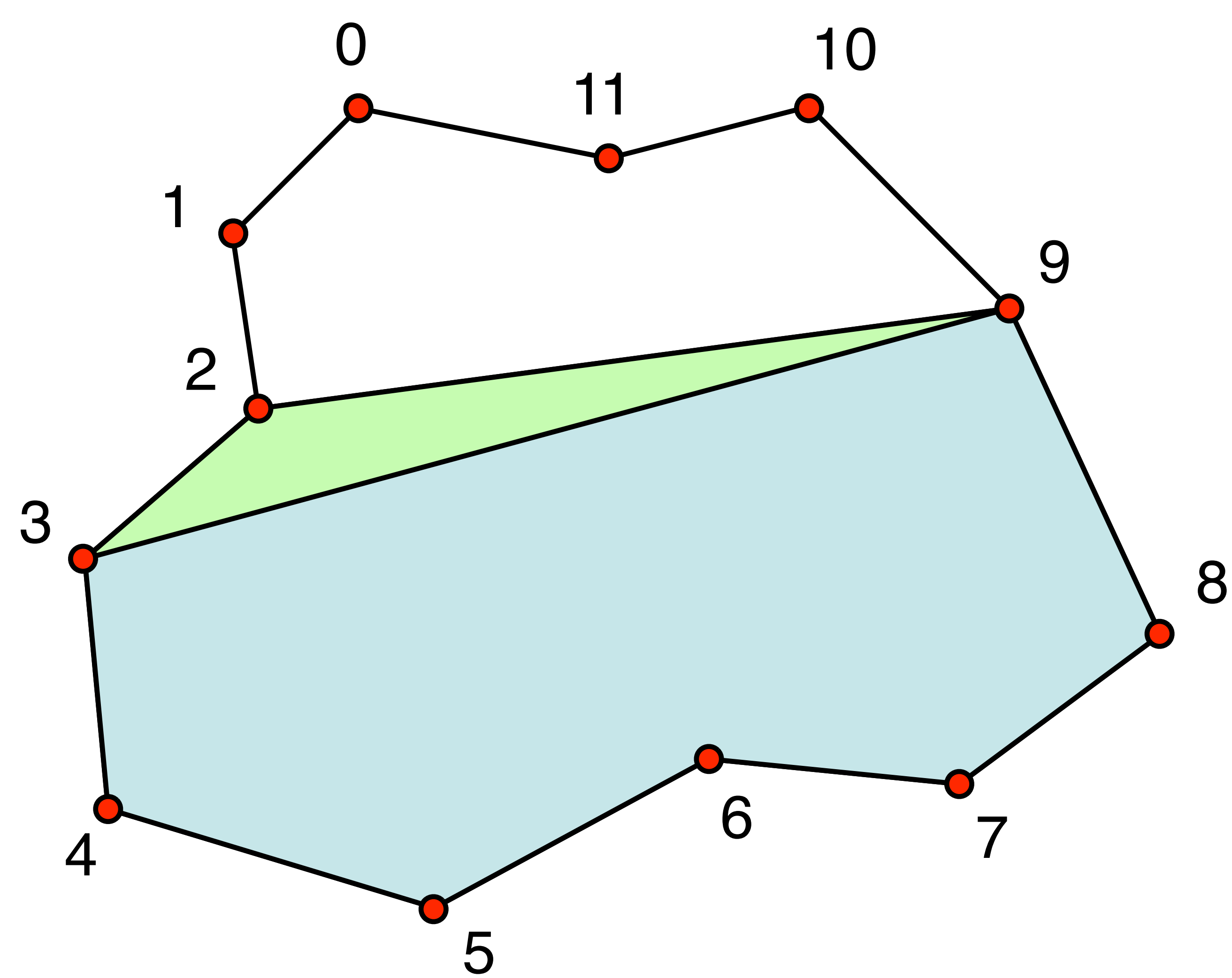
w[2,9] = ?



Filling Holes in Meshes - 1

- Let w[a,c] be the minimal weight that can be achieved in triangulating the polygon a,a+1,...,c

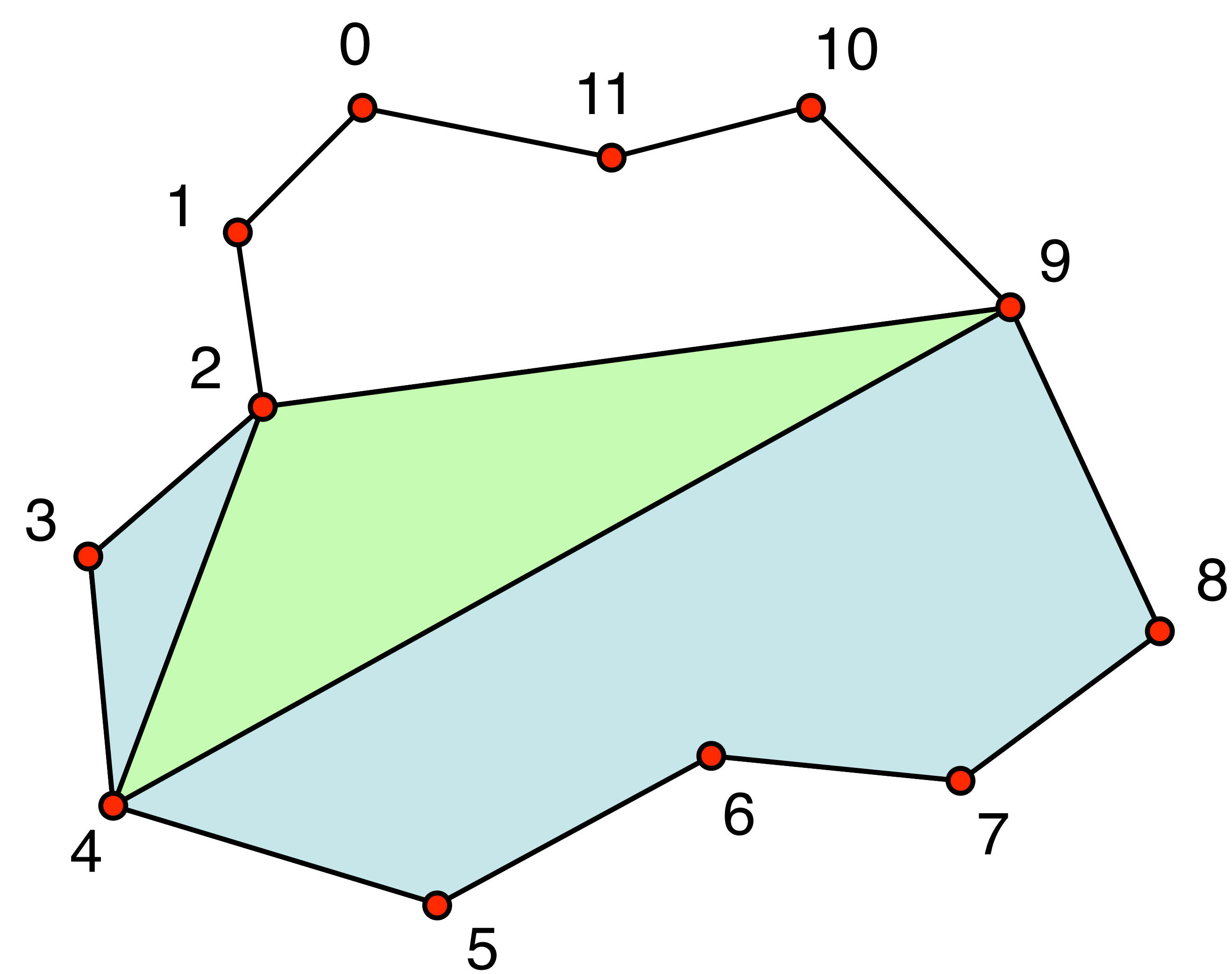
$$w[2,9] = \min(\; w(\Delta(2,3,9)) + w[3,9],$$



Filling Holes in Meshes - 1

- Let w[a,c] be the minimal weight that can be achieved in triangulating the polygon a,a+1,...,c

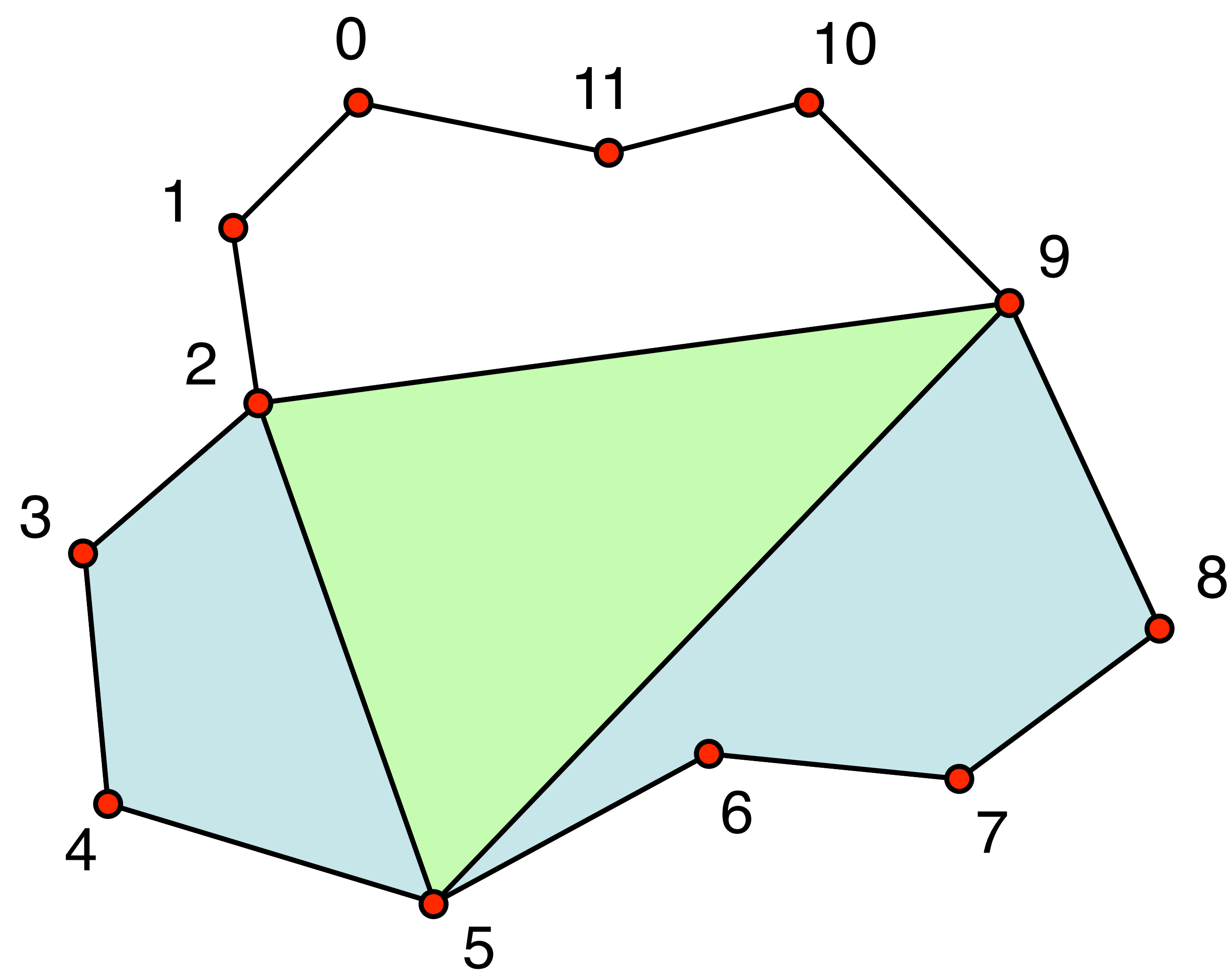
$$
\begin{aligned}
w[2,9] = \min(& \\
& w(\Delta(2,3,9)) + w[3,9], \\
& w[2,4] + w(\Delta(2,4,9)) + w[4,9],
\end{aligned}
$$



Filling Holes in Meshes - 1

- Let w[a,c] be the minimal weight that can be achieved in triangulating the polygon a,a+1,...,c

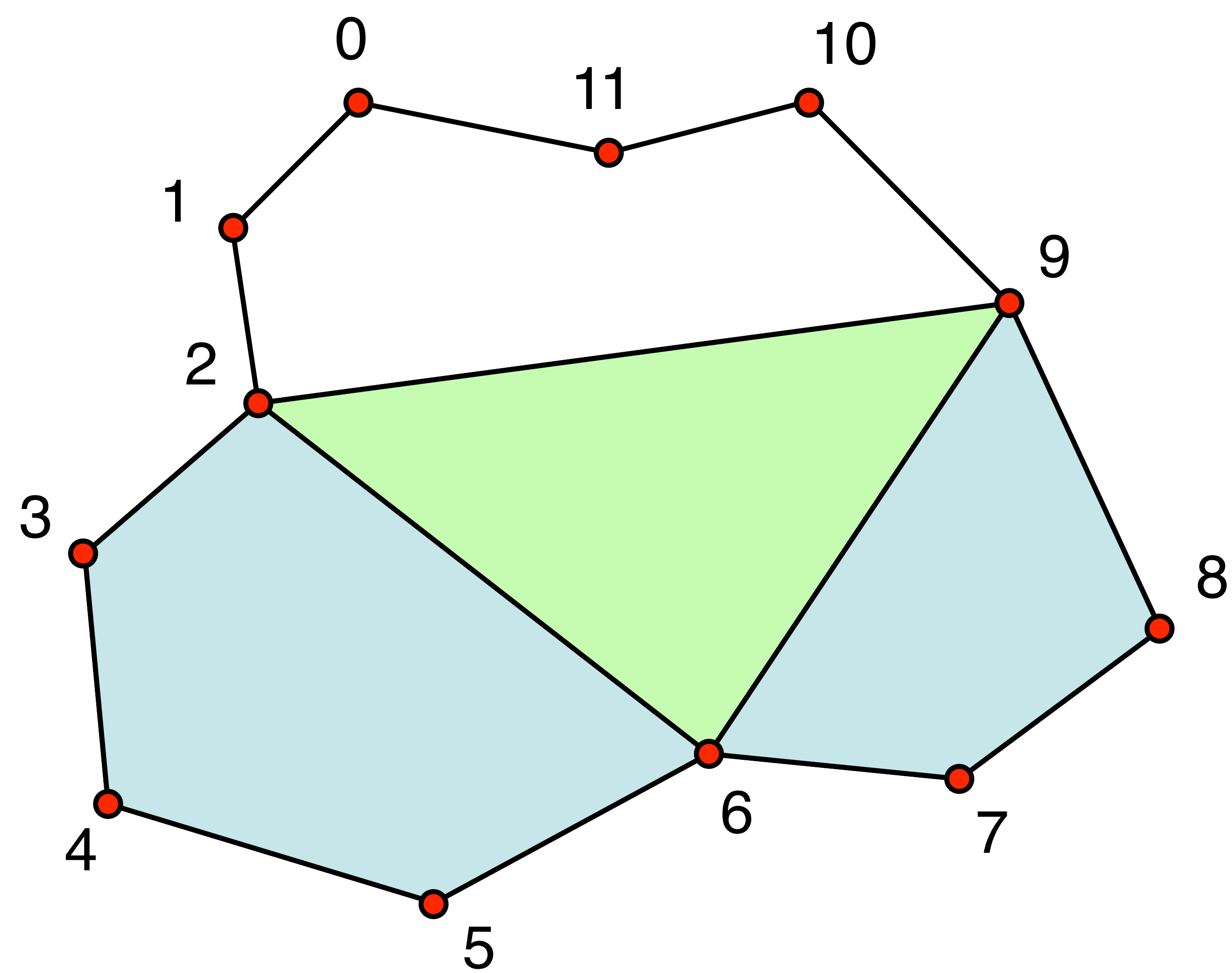
$$
\begin{aligned}
w[2,9] = \min(& \\
& w(\Delta(2,3,9)) + w[3,9], \\
w[2,4] + & w(\Delta(2,4,9)) + w[4,9], \\
w[2,5] + & w(\Delta(2,5,9)) + w[5,9],
\end{aligned}
$$



Filling Holes in Meshes - 1

- Let w[a,c] be the minimal weight that can be achieved in triangulating the polygon a,a+1,...,c

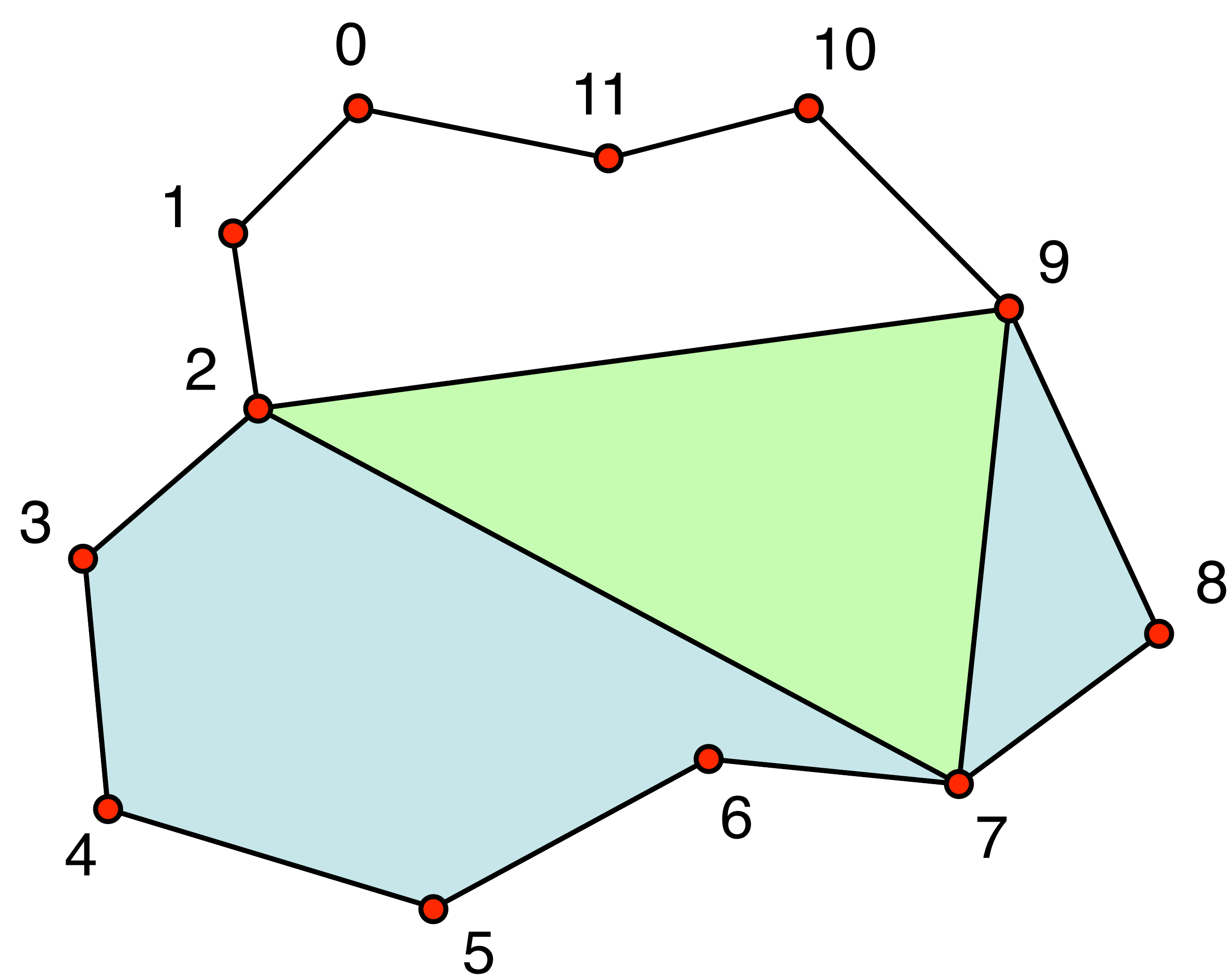
$$
\begin{aligned}
w[2,9] = \min(& \\
& w(\Delta(2,3,9)) + w[3,9], \\
w[2,4] + {} & w(\Delta(2,4,9)) + w[4,9], \\
w[2,5] + {} & w(\Delta(2,5,9)) + w[5,9], \\
w[2,6] + {} & w(\Delta(2,6,9)) + w[6,9],
\end{aligned}
$$



Filling Holes in Meshes - 1

- Let w[a,c] be the minimal weight that can be achieved in triangulating the polygon a,a+1,...,c

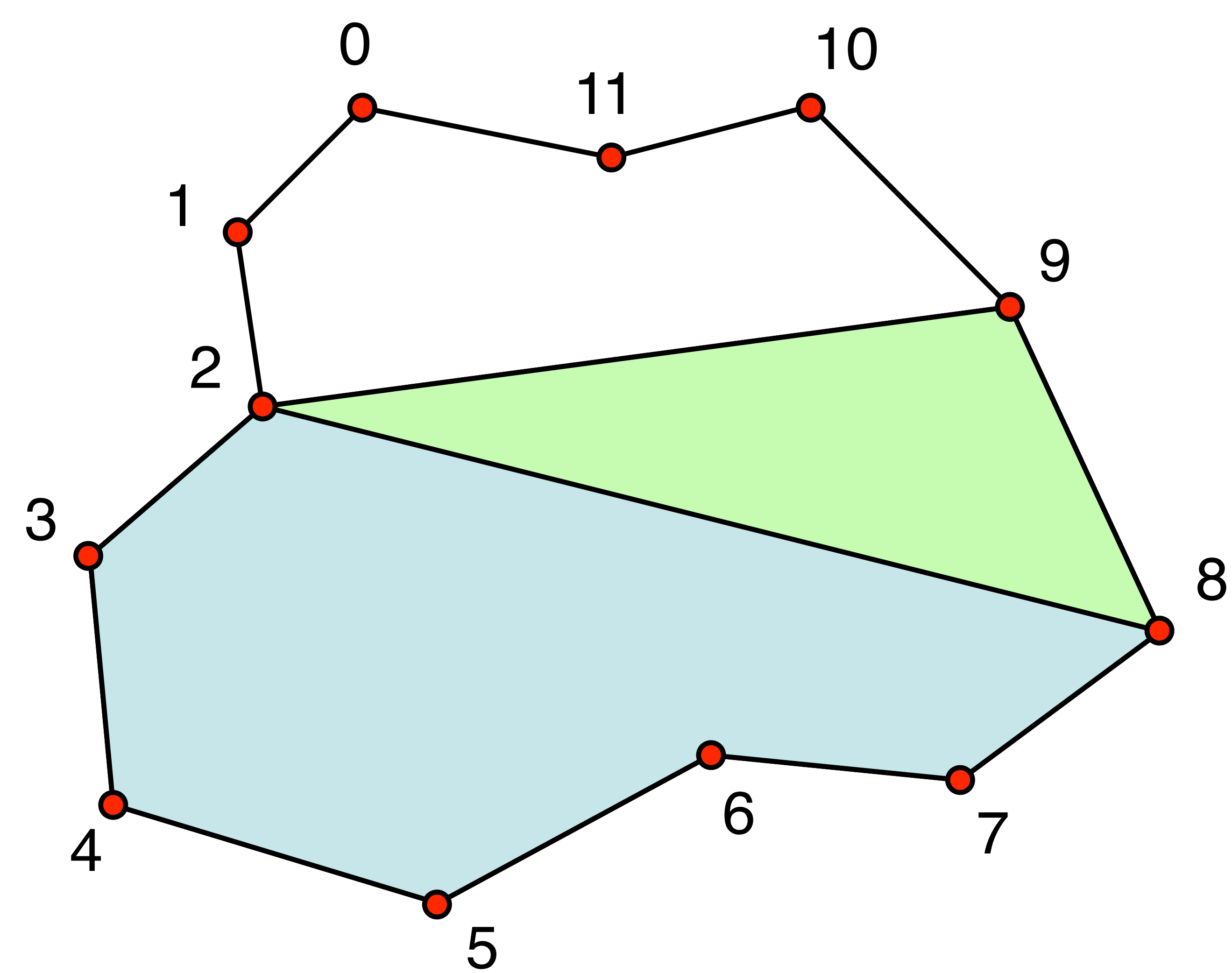
$$
\begin{aligned}
w[2,9] = \min(& \\
& w(\Delta(2,3,9)) + w[3,9], \\
w[2,4] + {} & w(\Delta(2,4,9)) + w[4,9], \\
w[2,5] + {} & w(\Delta(2,5,9)) + w[5,9], \\
w[2,6] + {} & w(\Delta(2,6,9)) + w[6,9], \\
w[2,7] + {} & w(\Delta(2,7,9)) + w[7,9],
\end{aligned}
$$



Filling Holes in Meshes - 1

- Let w[a,c] be the minimal weight that can be achieved in triangulating the polygon a,a+1,...,c

$$
\begin{aligned}
w[2,9] = \min(& \\
& w(\Delta(2,3,9)) + w[3,9], \\
& w[2,4] + w(\Delta(2,4,9)) + w[4,9], \\
& w[2,5] + w(\Delta(2,5,9)) + w[5,9], \\
& w[2,6] + w(\Delta(2,6,9)) + w[6,9], \\
& w[2,7] + w(\Delta(2,7,9)) + w[7,9], \\
& w[2,8] + w(\Delta(2,8,9)) \\
)&
\end{aligned}
$$



Filling Holes in Meshes - 1

- Let w[a,c] be the minimal weight that can be achieved in triangulating the polygon a,a+1,...,c
- Recursion formula

$$w[a,c] = \min_{a<b<c} w[a,b] + w(\Delta(a,b,c)) + w[b,c]$$

$$w[a-1,a+1] = w(\Delta(a-1,a,a+1))$$

- Dynamic programming leads to a $O(n^3)$ algorithm

Filling Holes in Meshes - 2+3

- Refine the triangulation such that its vertex density matches that of the surrounding area

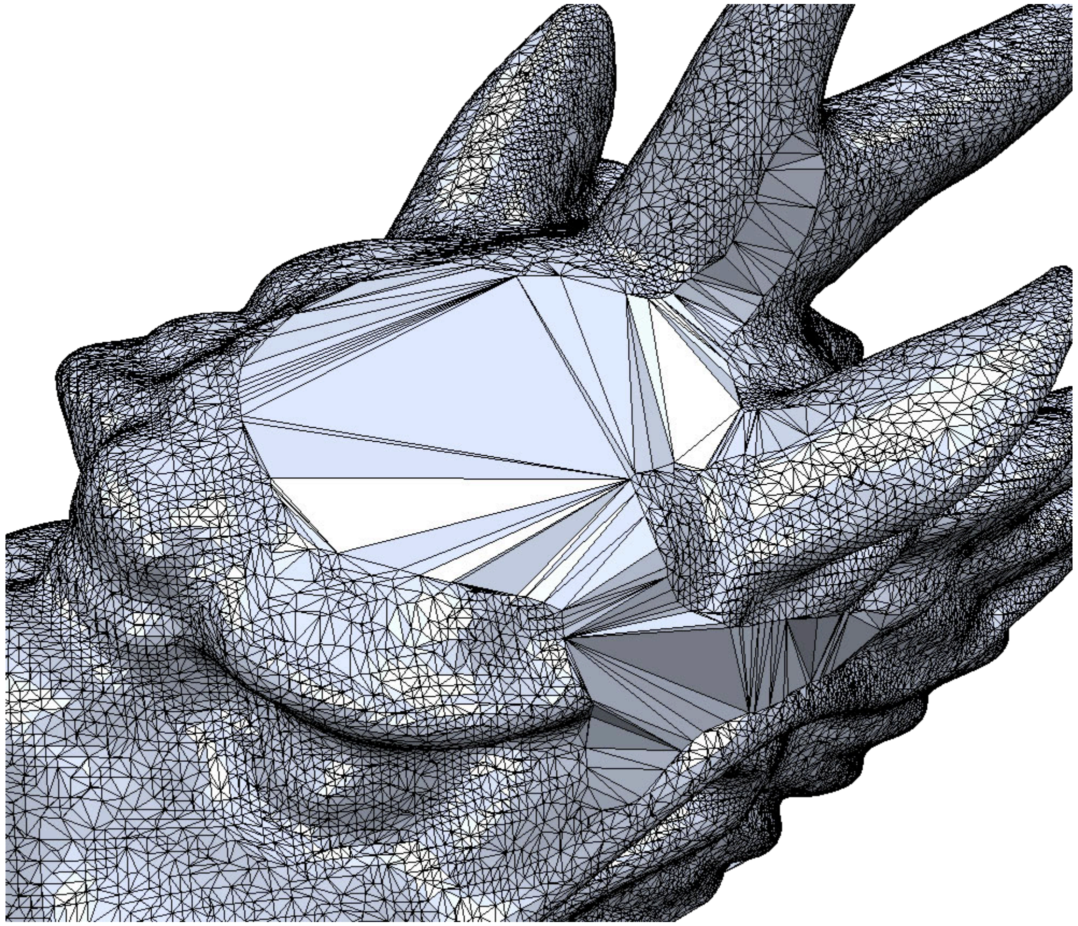
➡ Leif’s talk about remeshing

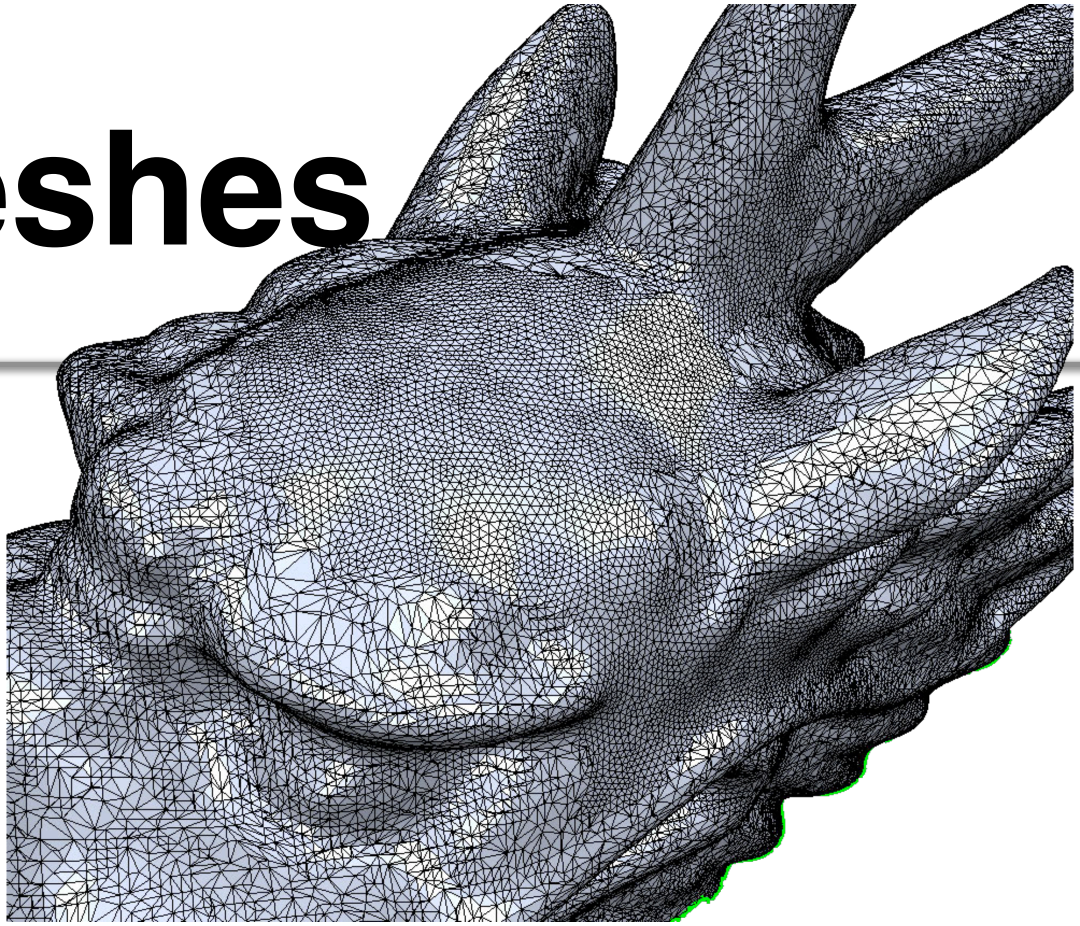
- Smooth the filling such that its geometry matches that of the surrounding area

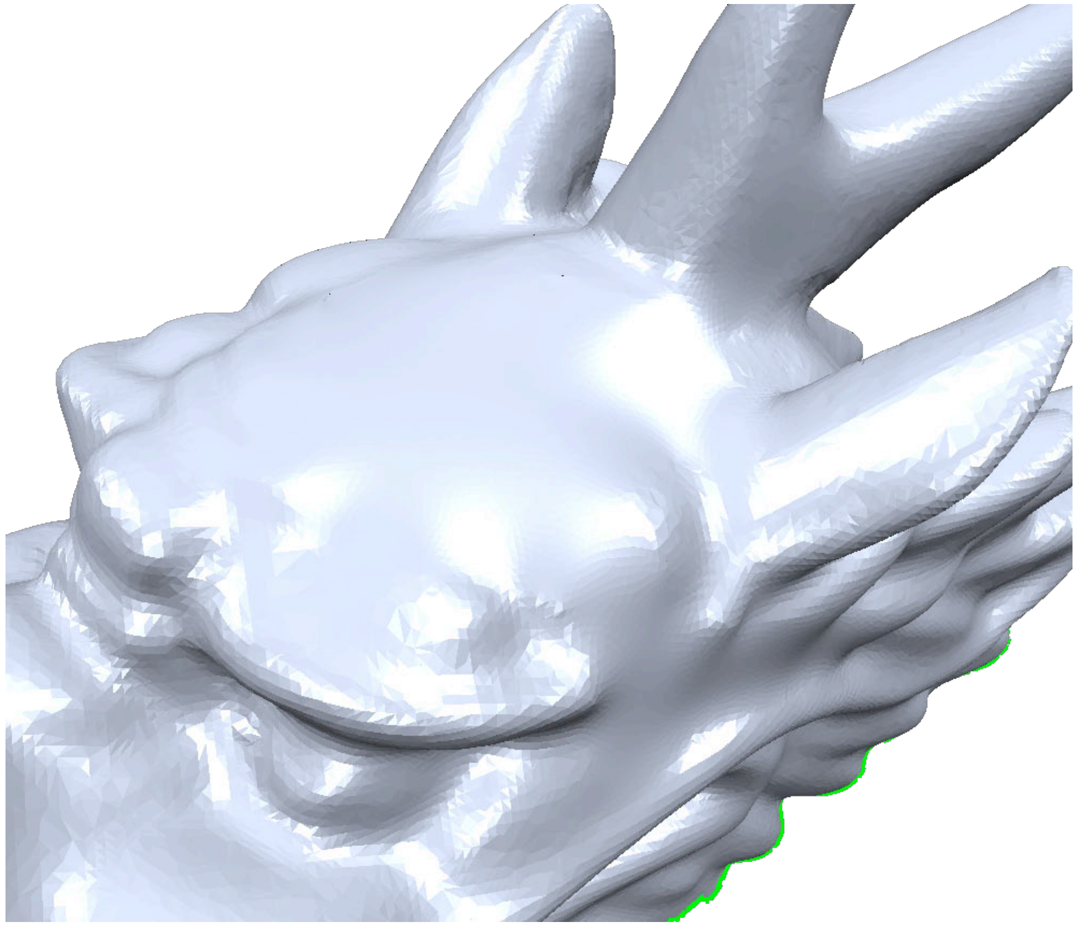
➡ Christian’s talk about mesh smoothing

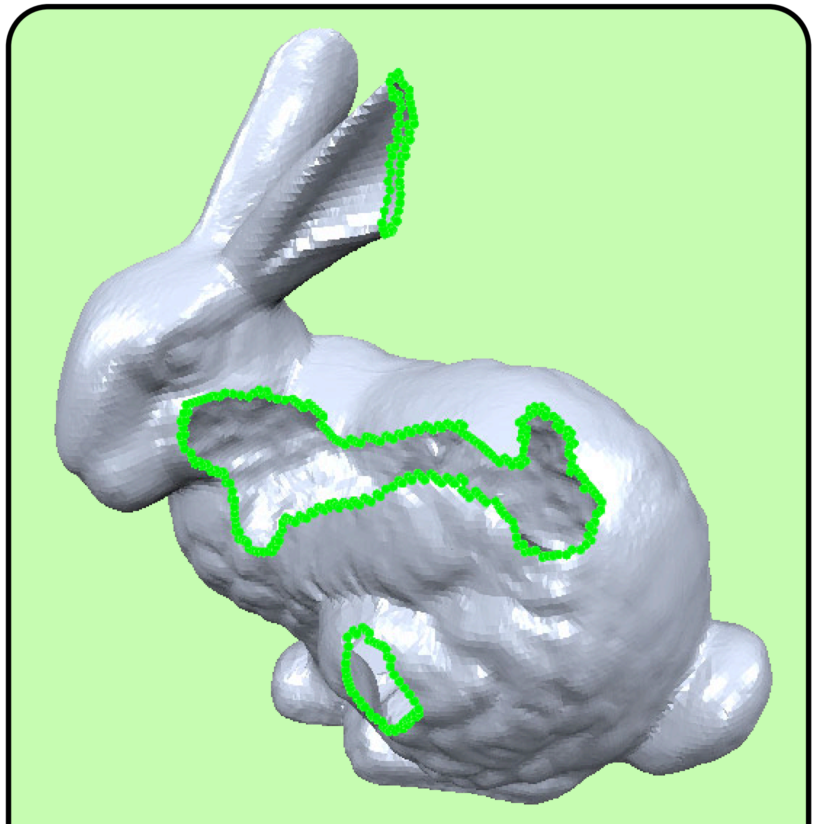
Filling Holes in Meshes

- Refinement and smoothing

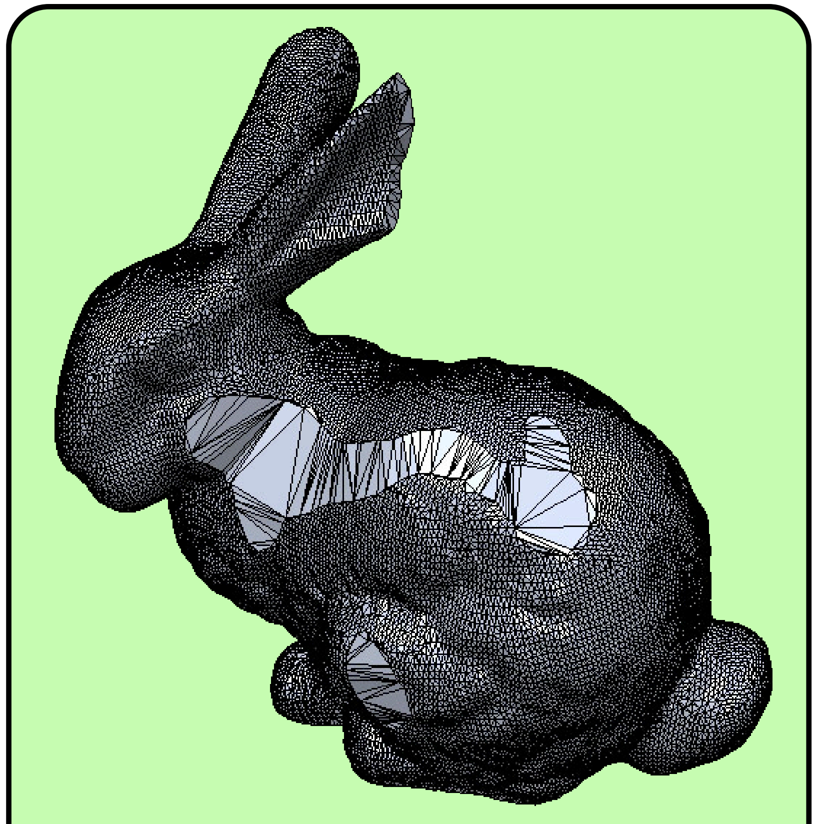




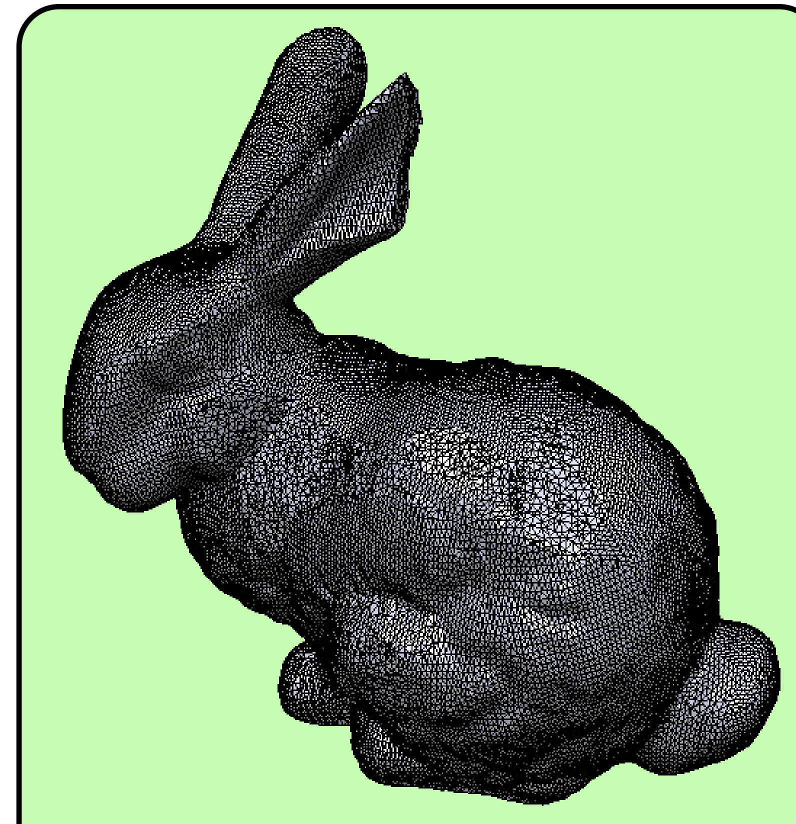




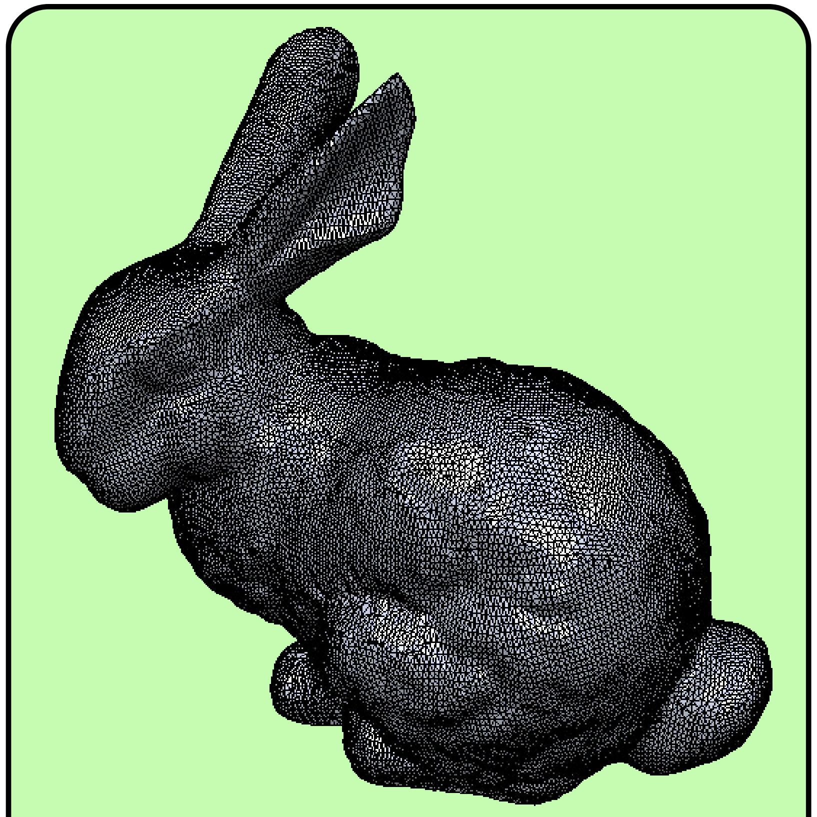
Input model



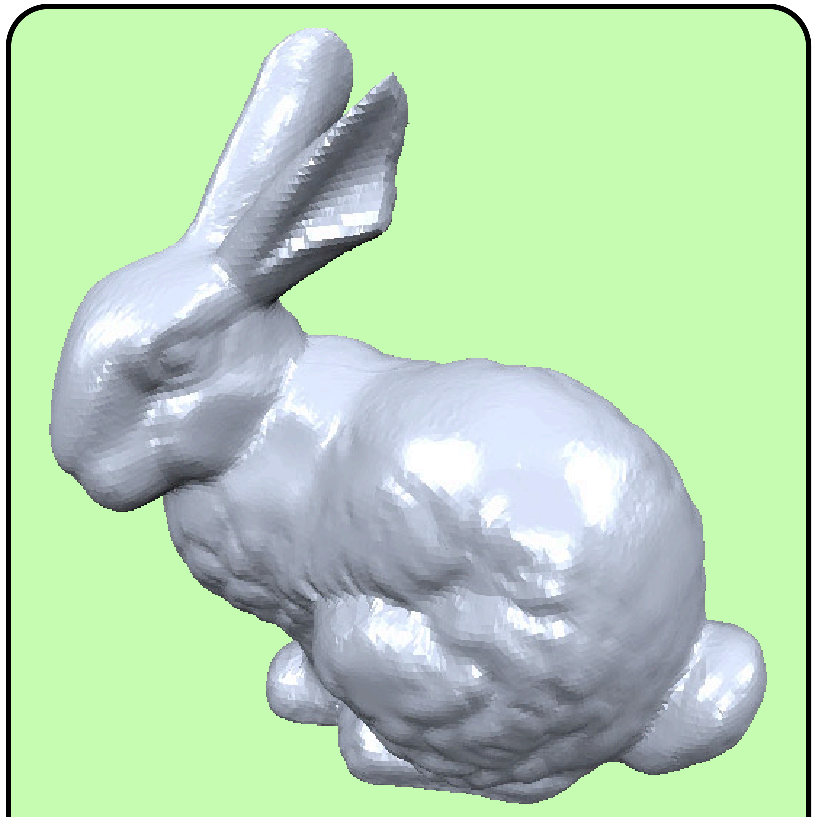
Minimal triangulation



Refined triangulation



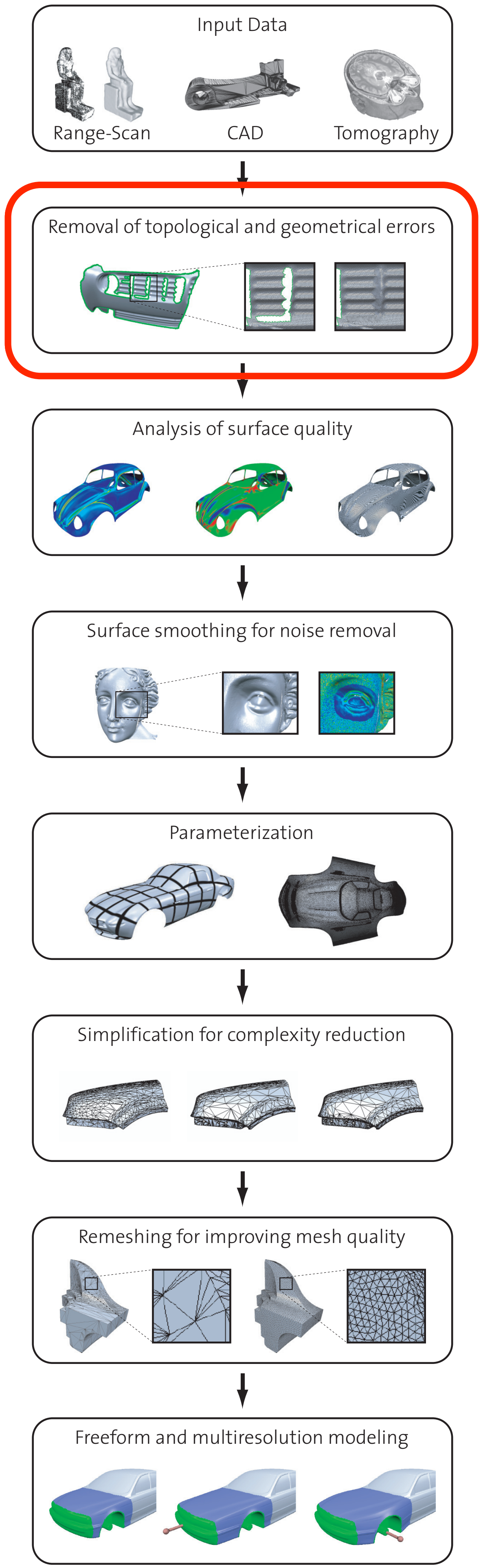
Output model



Output model

Filling Holes in Meshes

- What problems do we encounter?
  - Isles are not incorporated
  - Self-intersections cannot be excluded
  - Ugly fillings if boundary is too distorted
  - Boundary has to be topologically smooth

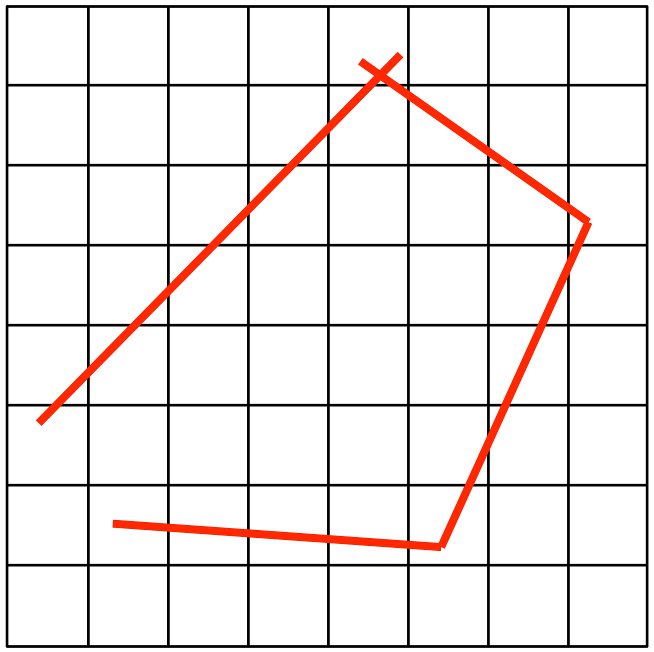


Model Repair

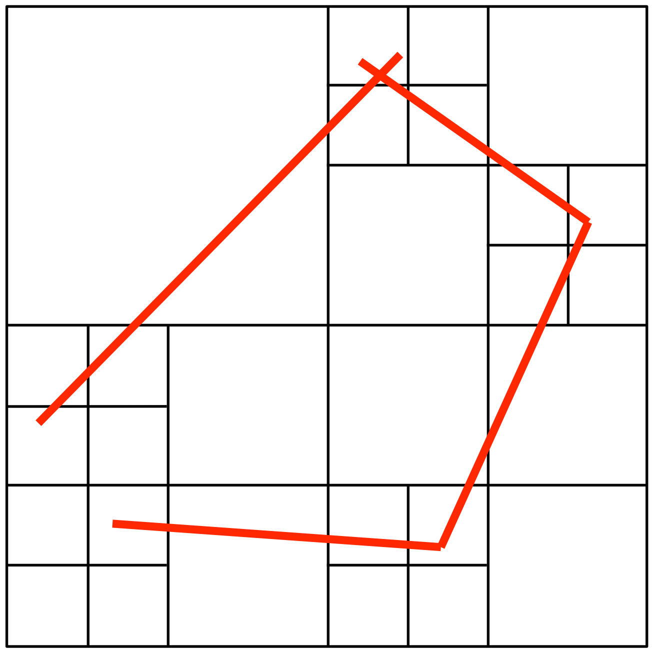
- Types of input
- Surface-oriented algorithms
  - Filling holes in meshes [Liepa 2003]
- **Volumetric algorithms**
  - Simplification and repair of polygonal models using volumetric techniques [Nooruddin and Turk 2003]
  - Automatic restoration of polygon models [Bischoff, Pavic, Kobbelt 2005]
- Conclusion & outlook

Volumetric Algorithms

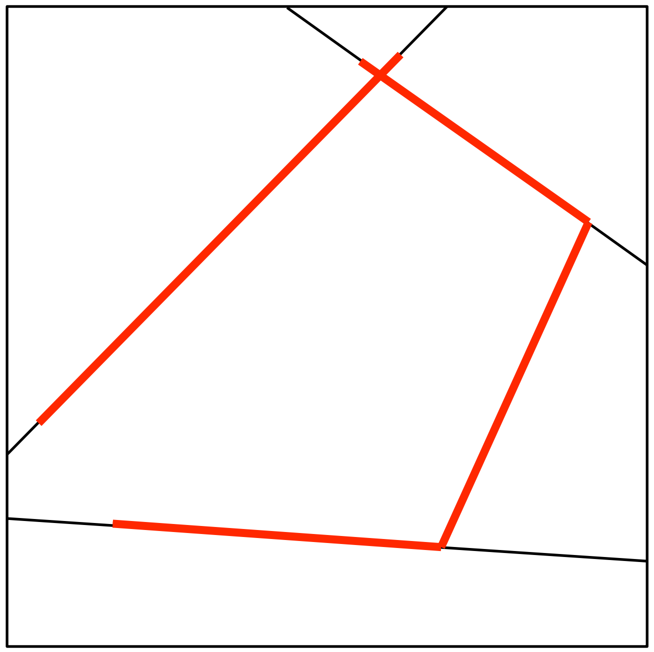
1. Convert the input model into an intermediate volumetric representation ➙ loss of information



Signed distance field



Octree



BSP tree

Volumetric Algorithms

1. Convert the input model into an intermediate volumetric representation ➞ loss of information
2. Discrete volumetric representation ➞ robust processing
   - Morphological operators (dilation, erosion)
   - Smoothing
   - Flood-fill to determine interior/exterior
   - ...

Volumetric Algorithms

1. Convert the input model into an intermediate volumetric representation ➞ loss of information
2. Discrete volumetric representation ➞ robust processing
   - Morphological operators (dilation, erosion)
   - Smoothing
   - Flood-fill to determine interior/exterior
3. Extract a surface from the volume ➞ The surface of a solid object is manifold and watertight!

Volumetric Algorithms

- Advantages
  - Fully automatic
  - Few user parameters
  - Robust
  - Guaranteed manifold output

Volumetric Algorithms

- Issues
  - Slow and memory intensive ➞ adaptive data structures
  - Aliasing and loss of features ➞ feature sensitive reconstruction (EMC, DC, Varadhan et al.)
  - Loss of structure ➞ bad luck
  - Large output ➞ mesh decimation (Mark’s talk)

Example 1

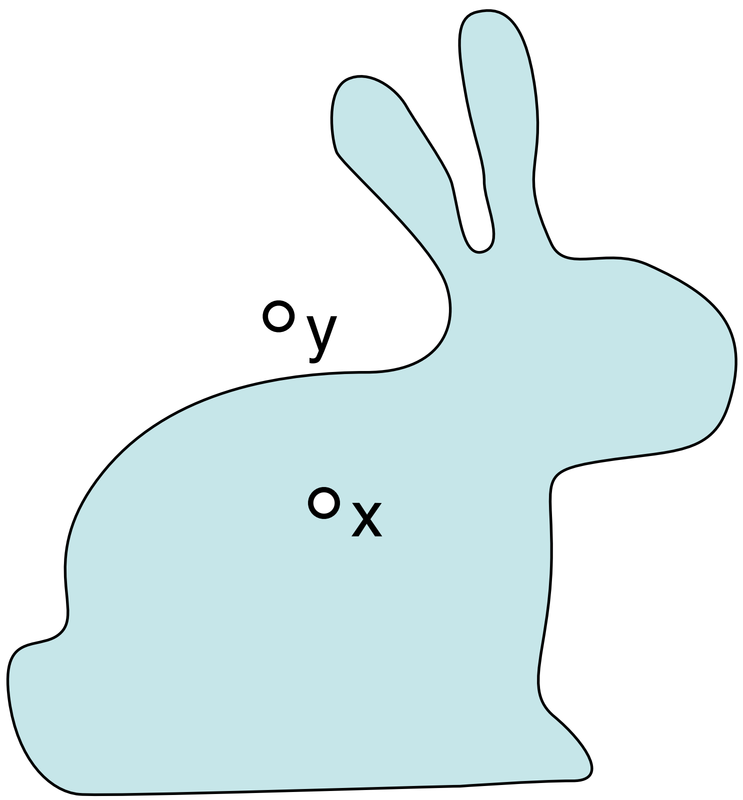
- Example algorithm

  F. S. Nooruddin and G. Turk
  Simplification and Repair of Polygonal Models Using Volumetric Techniques
  IEEE Transactions on Visualization and Computer Graphics 2003

- Issues
  - Classification of sample points x as being inside or outside of the object
  - Sampling the volume
  - Extracting the mesh

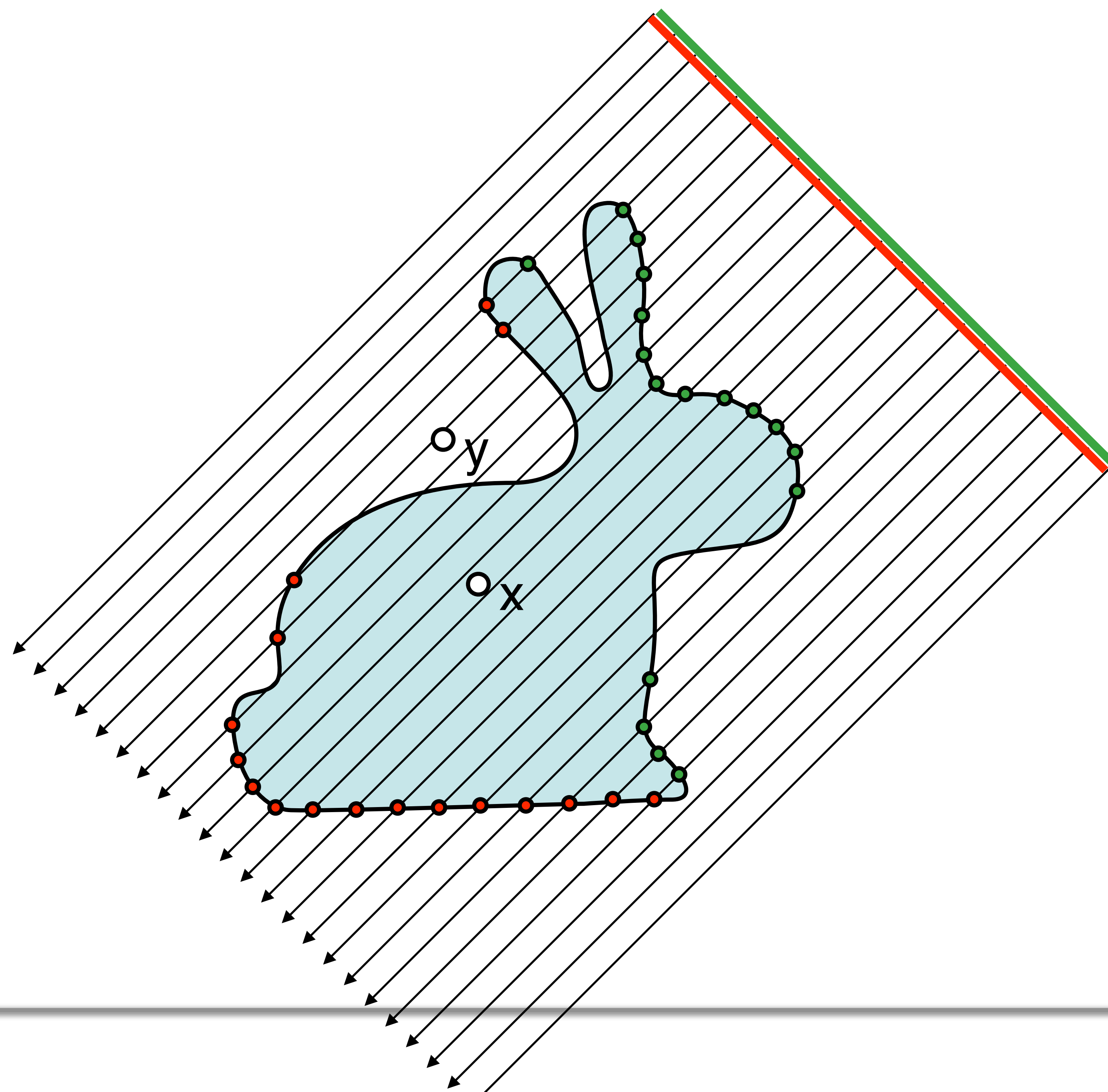
Nooruddin and Turk's Method

- Point classification: Layered depth images (LDI)



Nooruddin and Turk’s Method

- Point classification: Layered depth images (LDI)

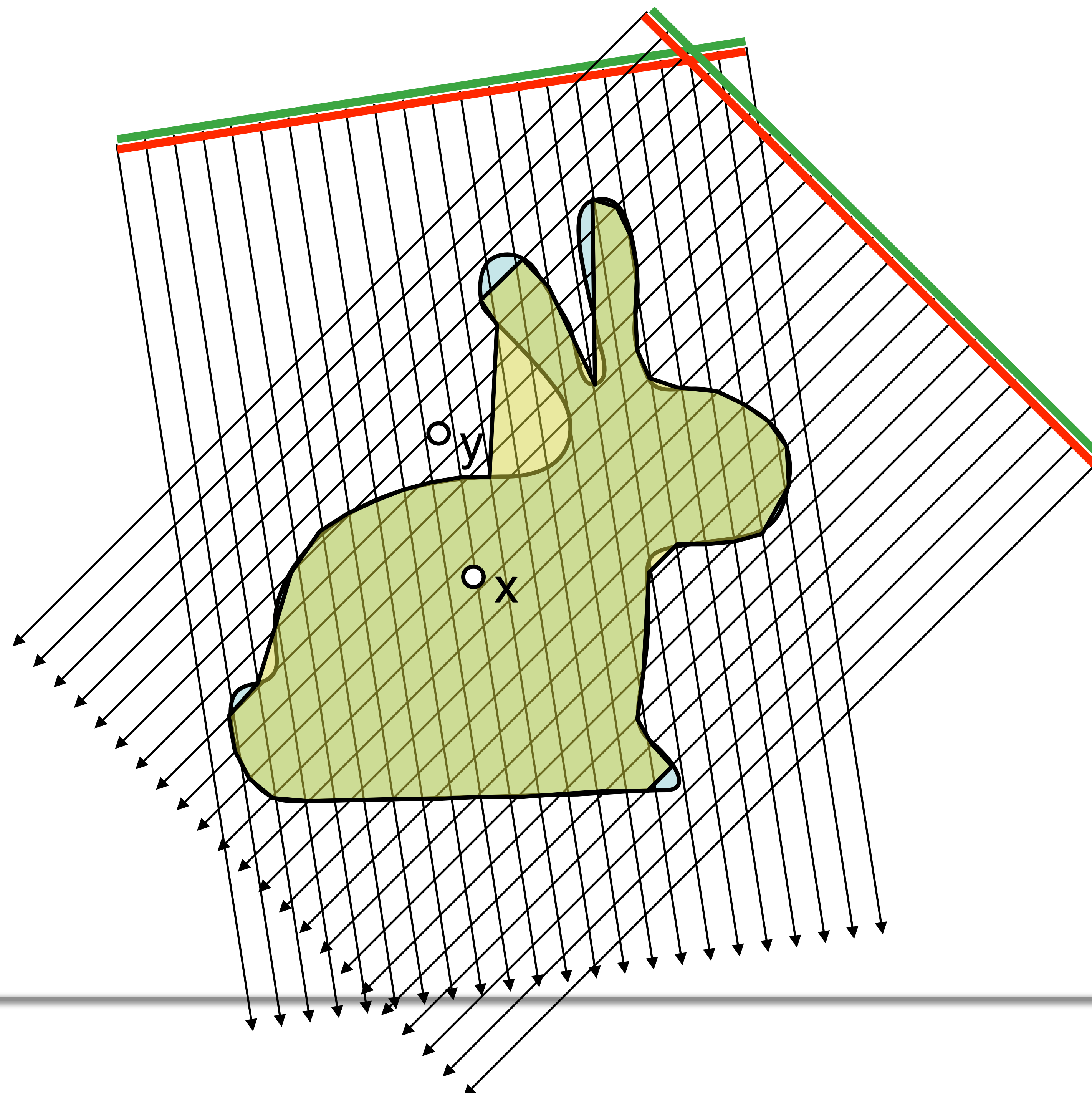


Nooruddin and Turk's Method

- Point classification: Layered depth images (LDI)

Nooruddin and Turk's Method

- Point classification: Layered depth images (LDI)

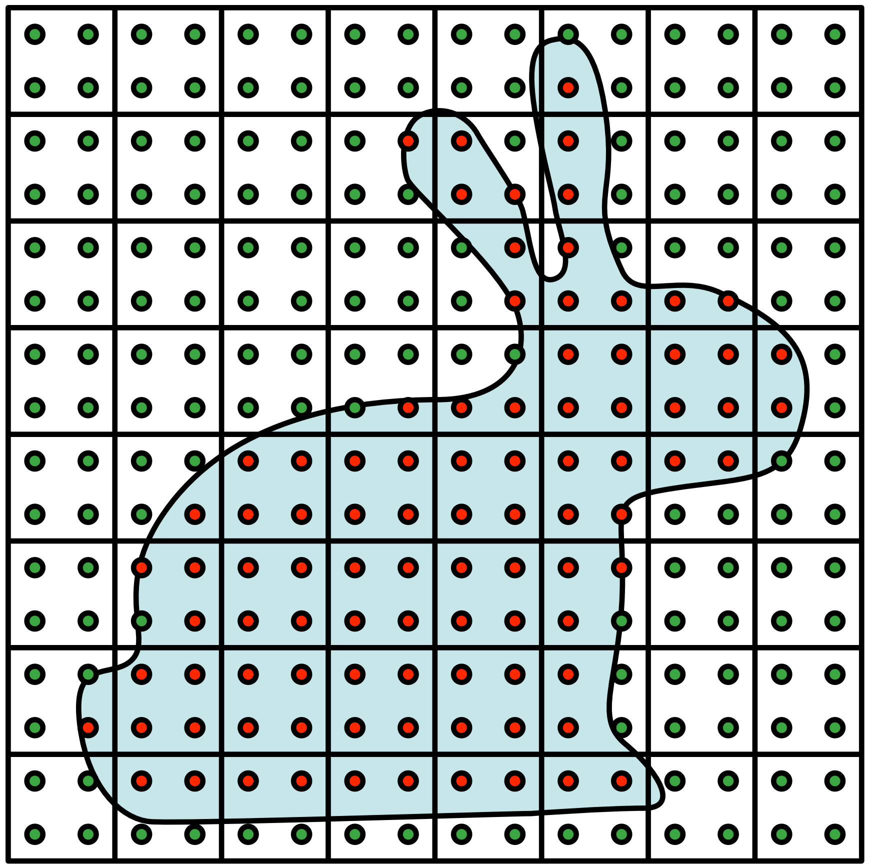


Nooruddin and Turk's Method

- Point classification: Layered depth images (LDI)
    1. Record n layered depth images
    2. Project the query point x into each depth image
    3. If any of the images classifies x as exterior, then x is globally classified as exterior else as interior

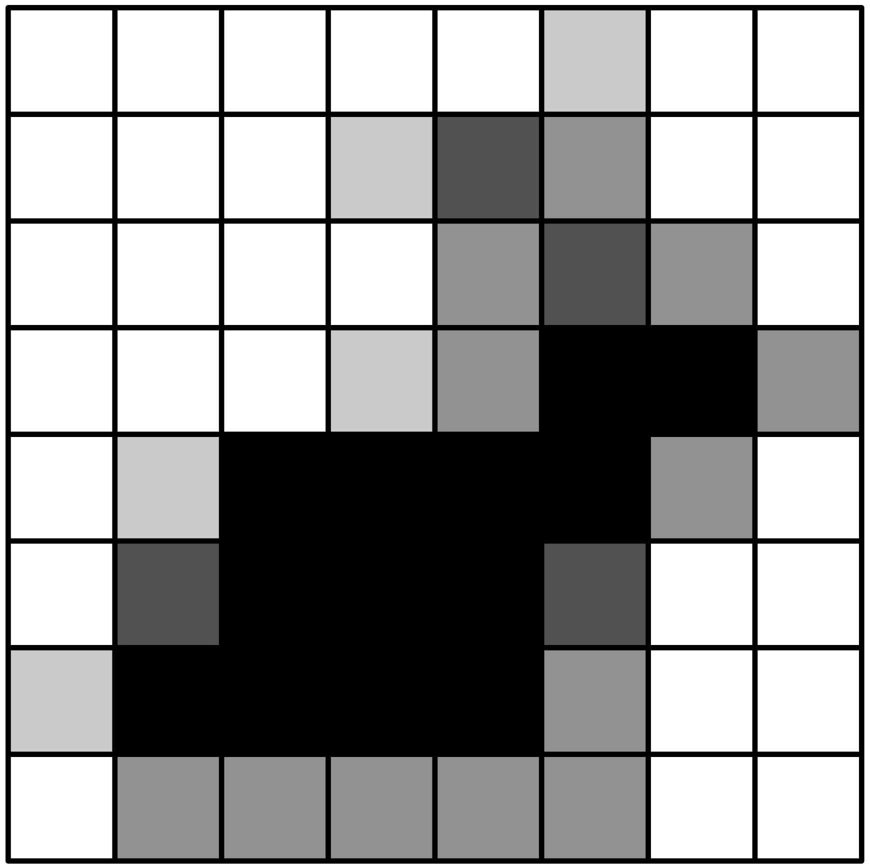
Nooruddin and Turk's Method

- Supersampling
- Filtering
  - Gaussian
  - Morphological filters (dilation, erosion)
    - model simplification
    - reduction of topological noise
- Marching Cubes



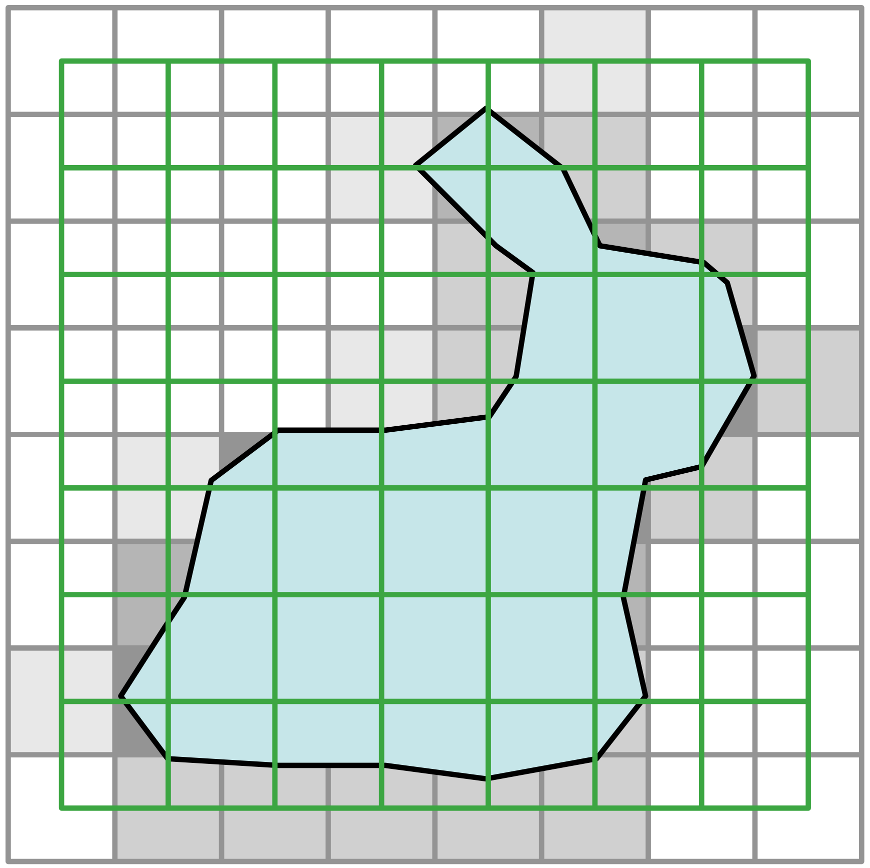
Nooruddin and Turk's Method

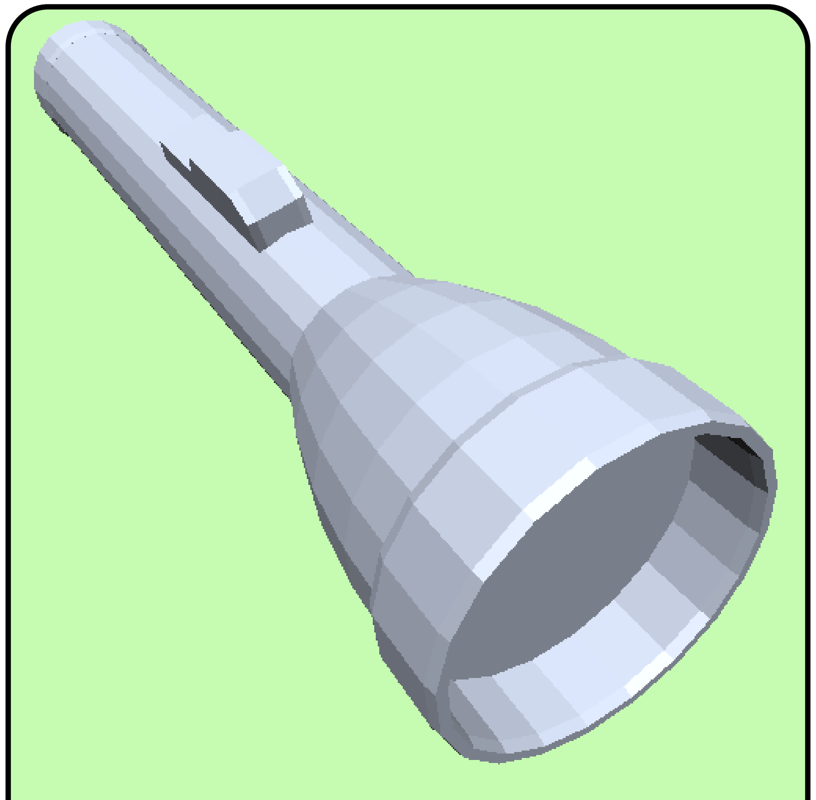
- Supersampling
- Filtering
  - Gaussian
  - Morphological filters (dilation, erosion)
    - model simplification
    - reduction of topological noise
- Marching Cubes

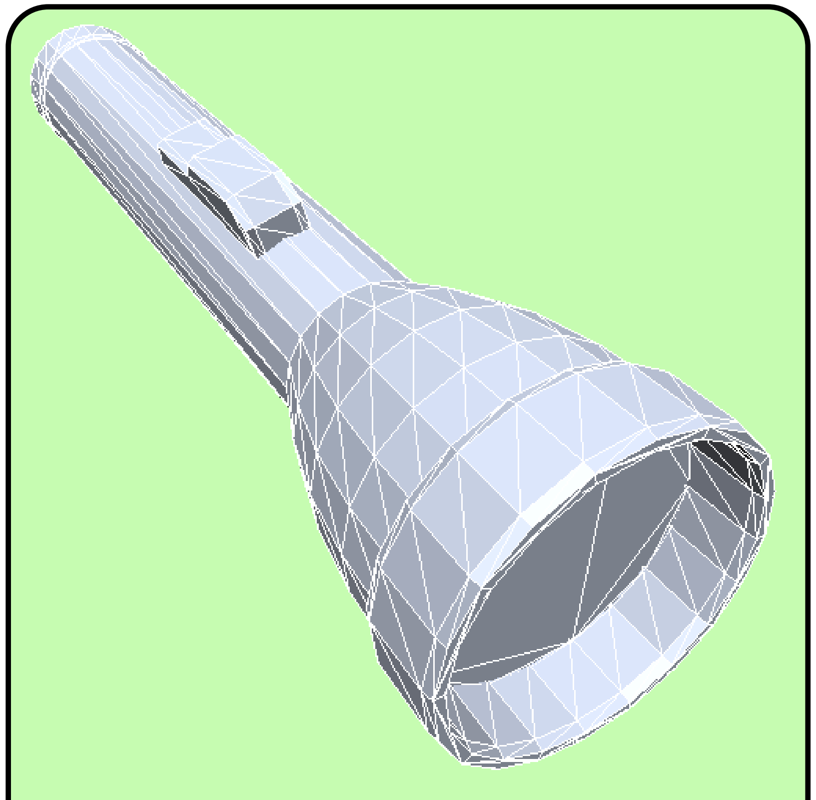


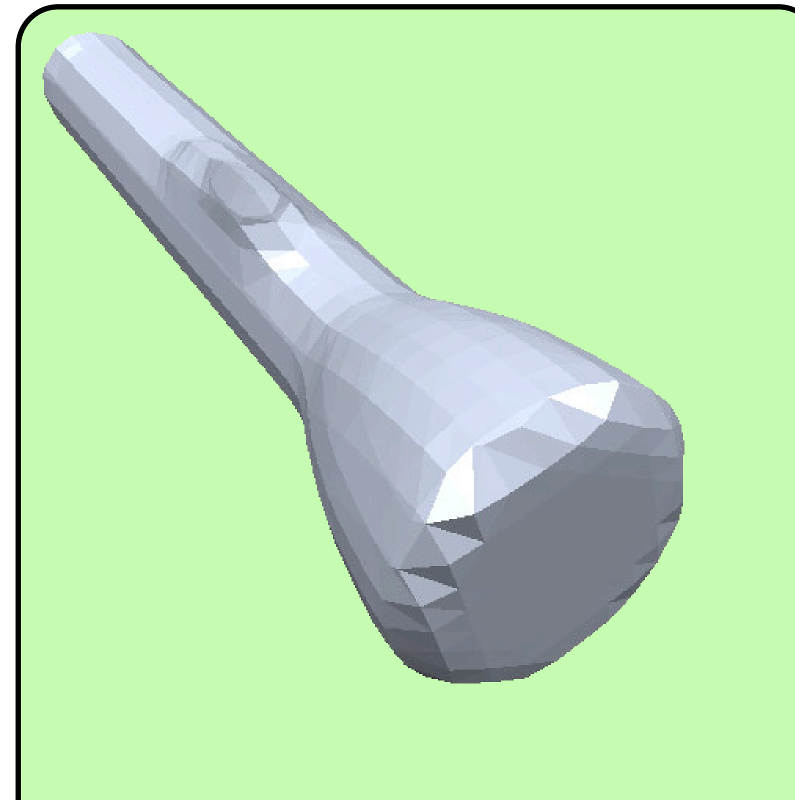
Nooruddin and Turk’s Method

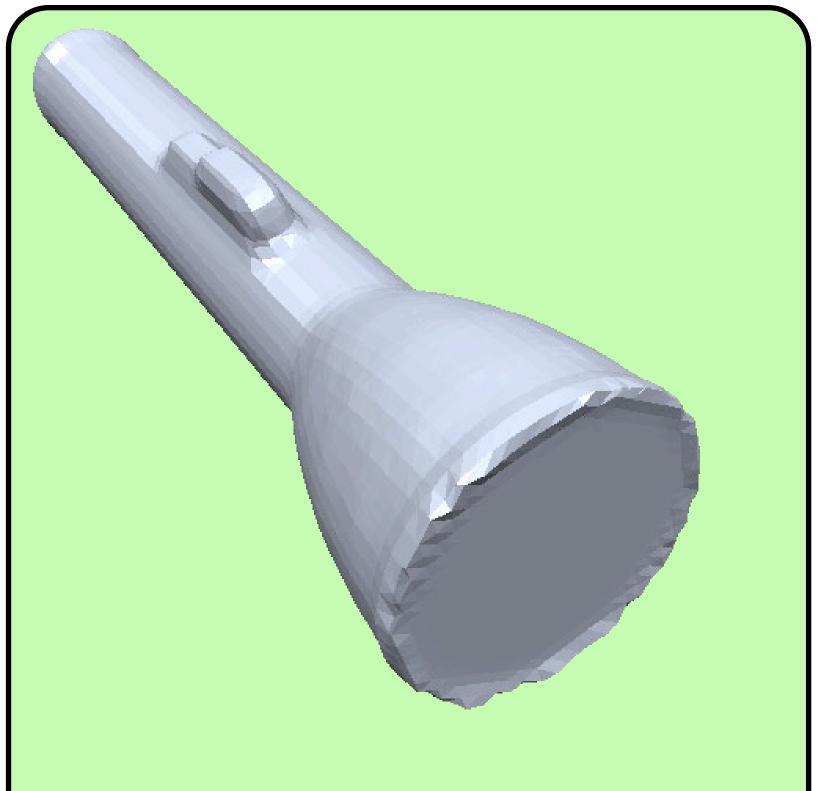
- Supersampling
- Filtering
  - Gaussian
  - Morphological filters (dilation, erosion)
    - model simplification
    - reduction of topological noise
- Marching Cubes

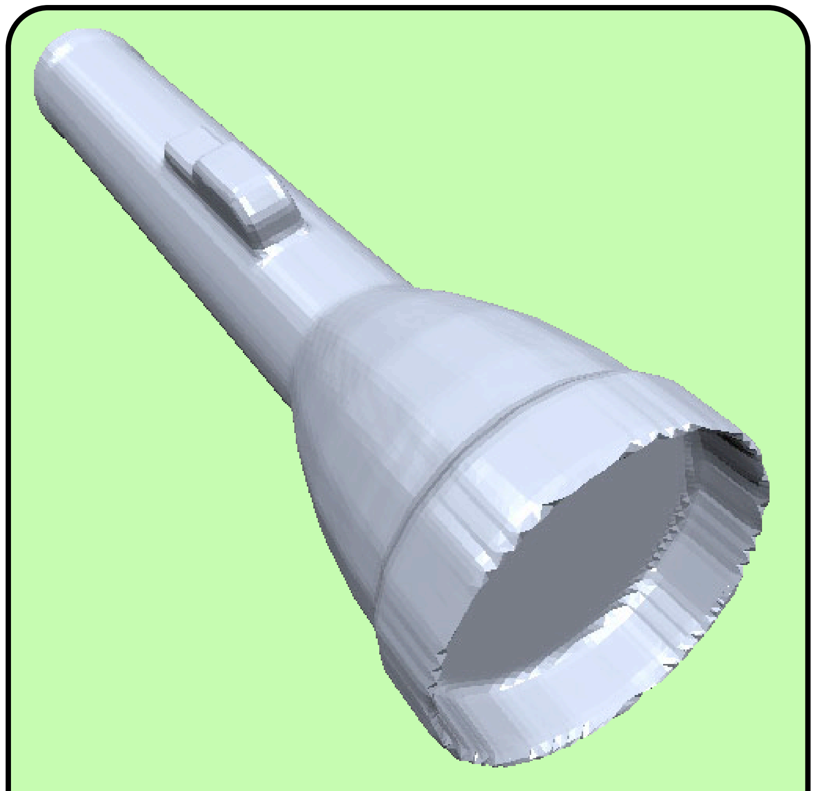


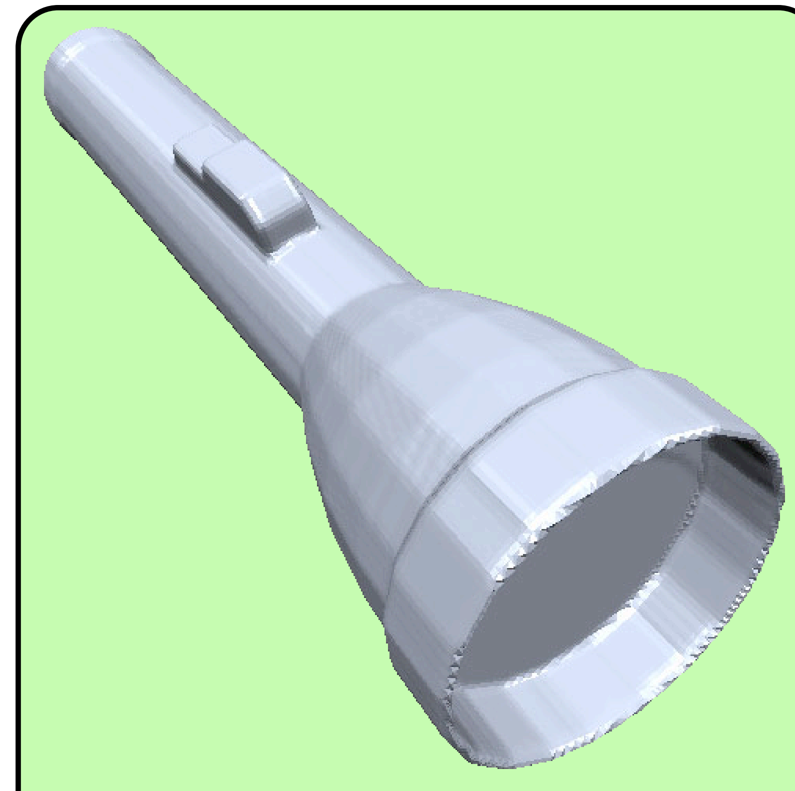

Input model

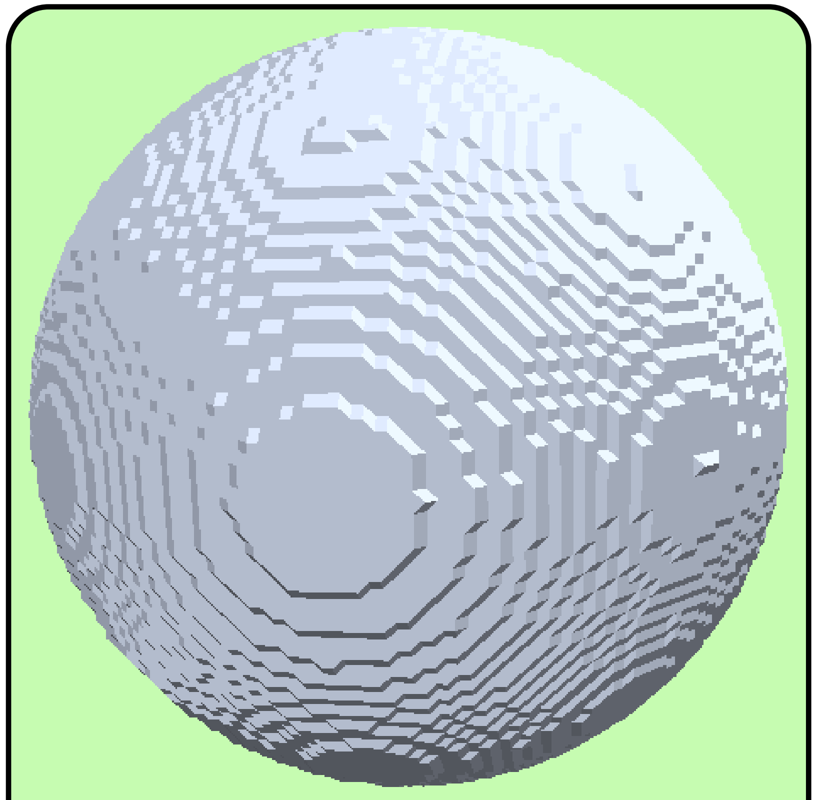

Input model


50×50×50


100×100×100

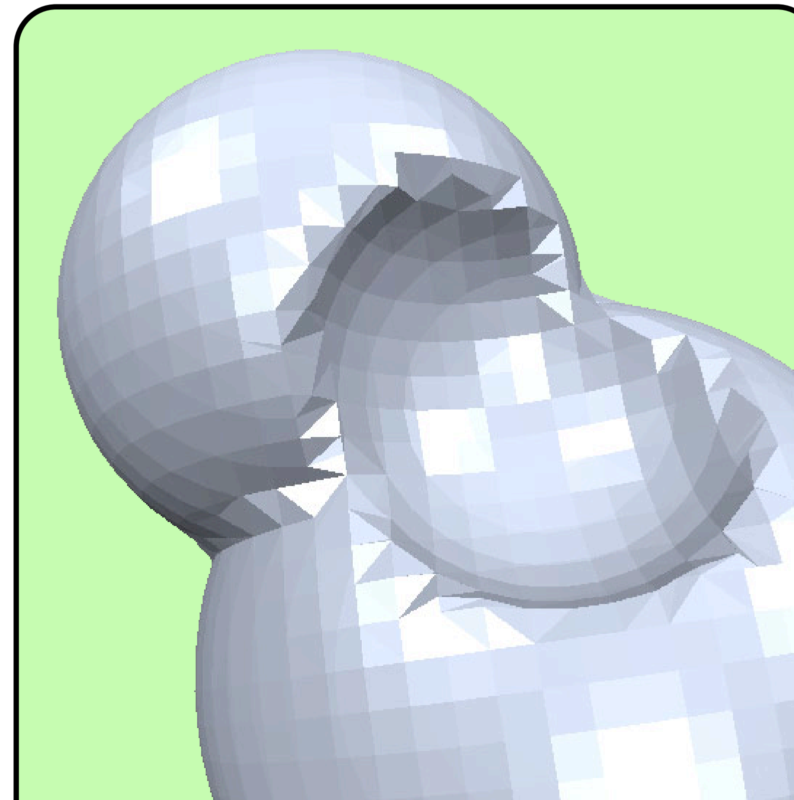

200×200×200


300×300×300

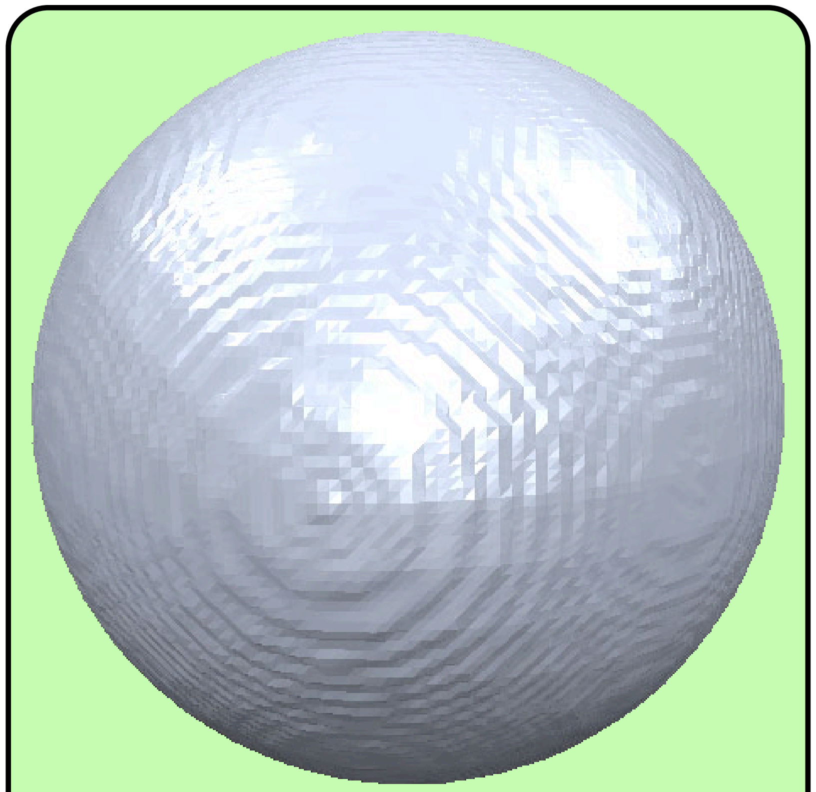


Raw

Smoothing

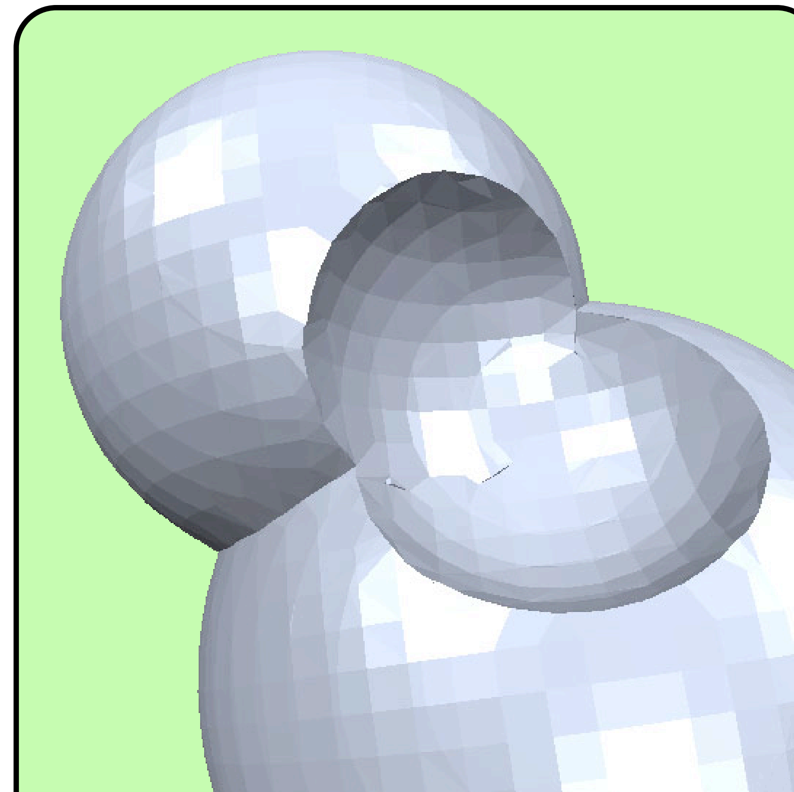


Marching Cubes



Supersampling

Supersampling + smoothing



Extended Marching Cubes

Example 2

- Example algorithm 2

  S. Bischoff, D. Pavic, L. Kobbelt

  Automatic Restoration of Polygon Models

  Transactions on Graphics 2005

Overview

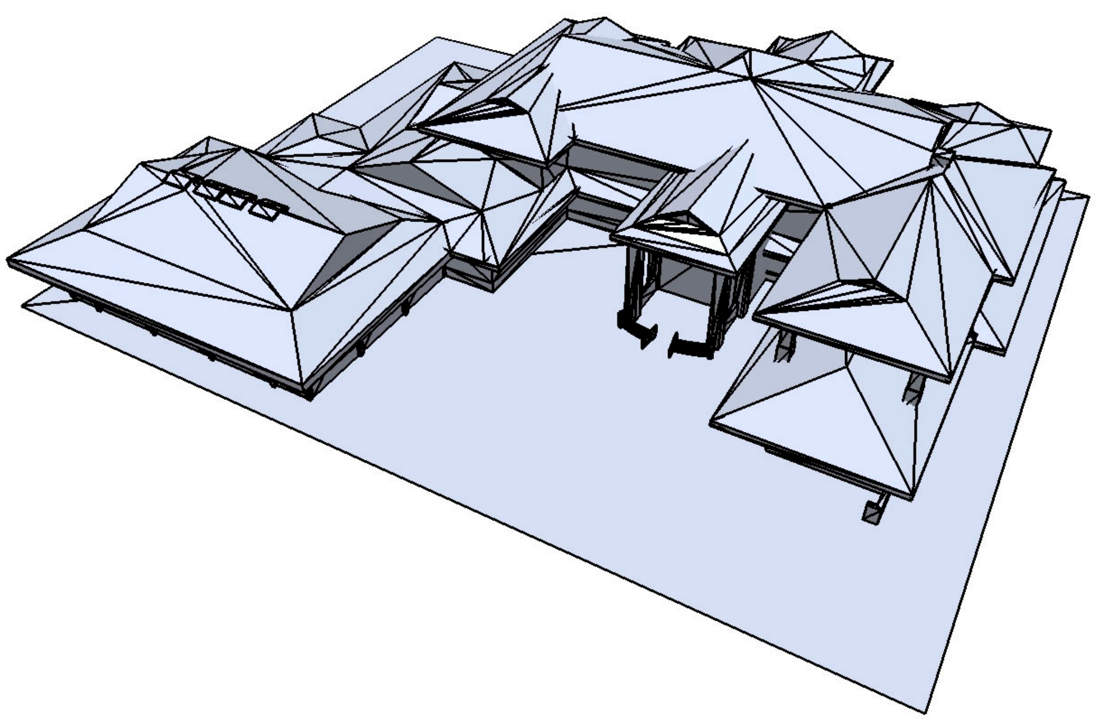
volumetric representation

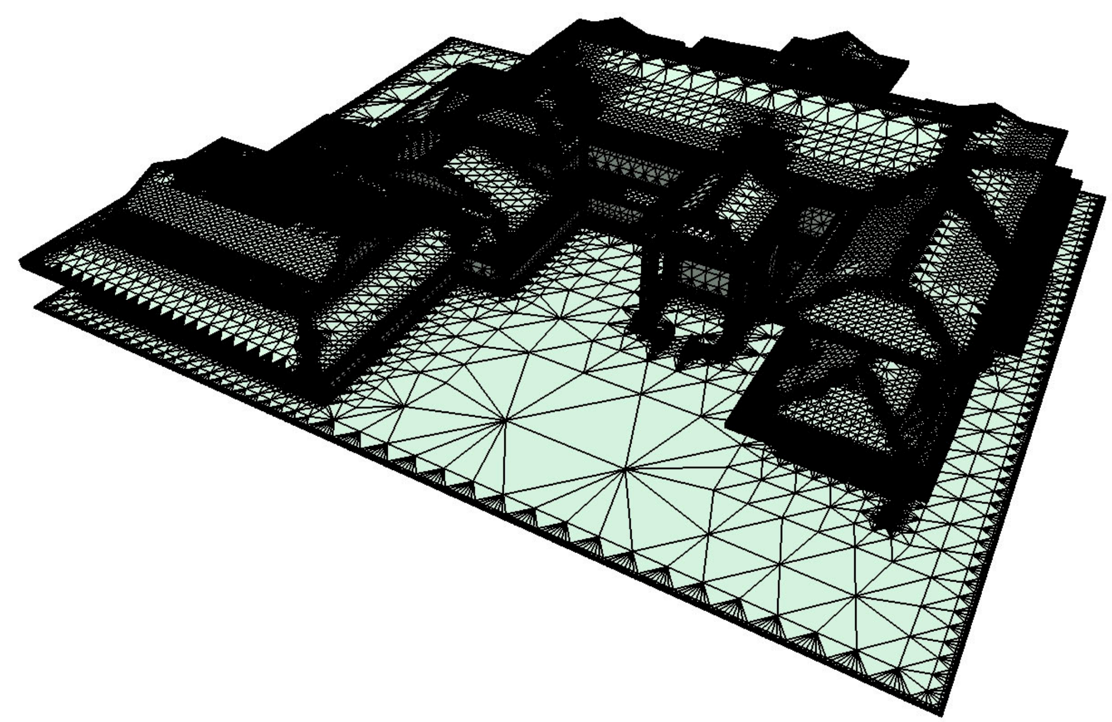
gap filling, removal of interior geometry

volumetric representation

conversion

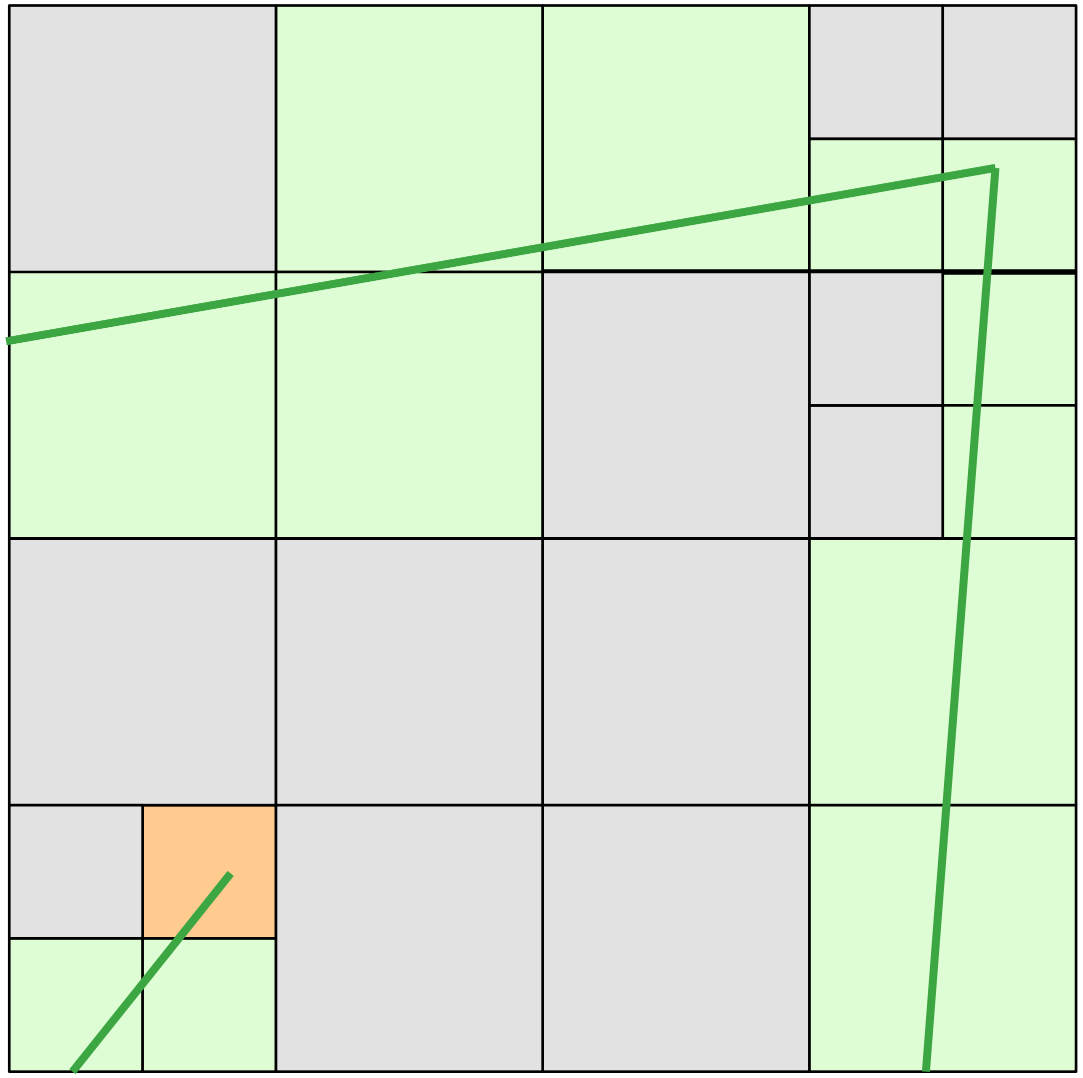
manifold mesh extraction





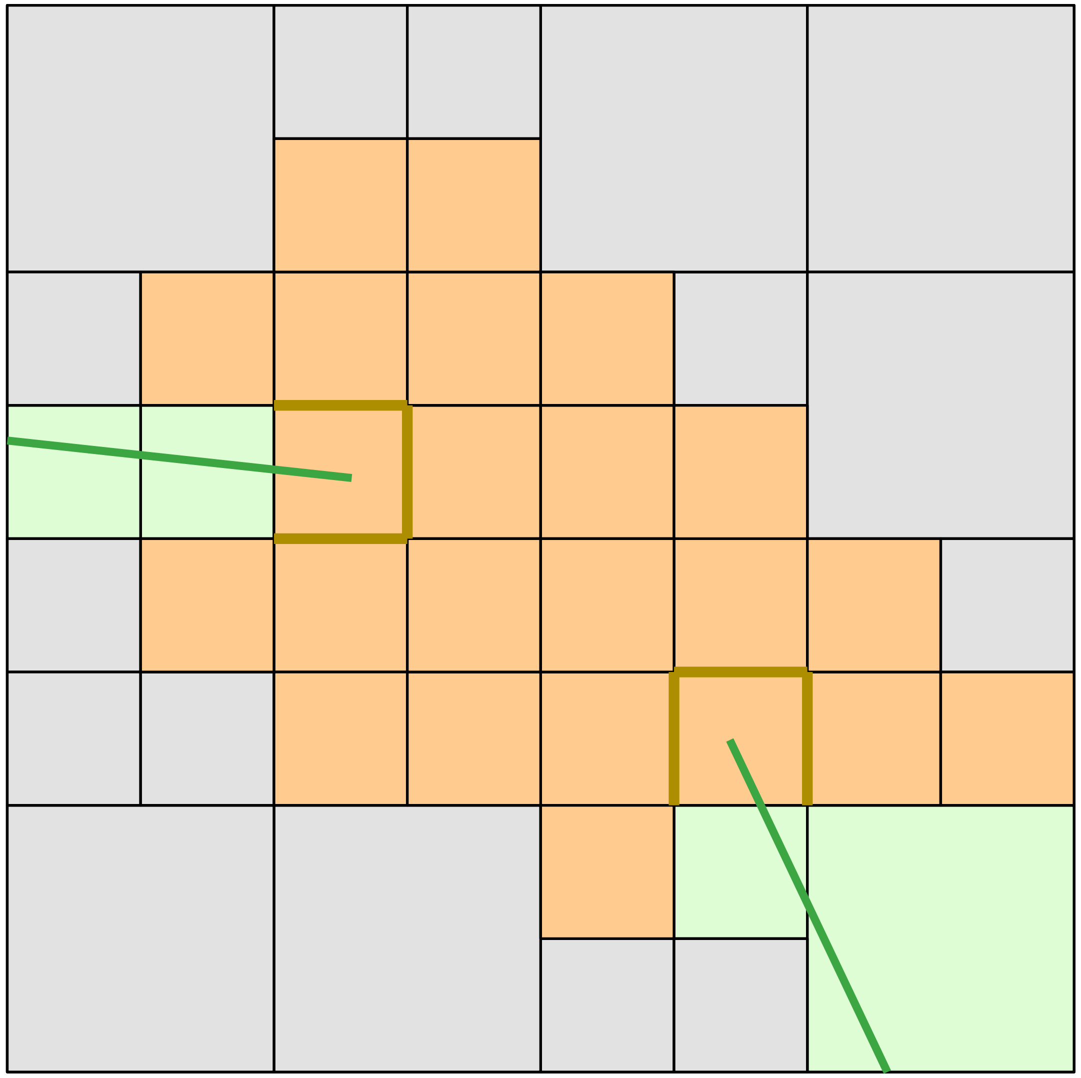
Conversion

- Adaptive octree: Subdivide a cell, if it contains multiple planes or a boundary



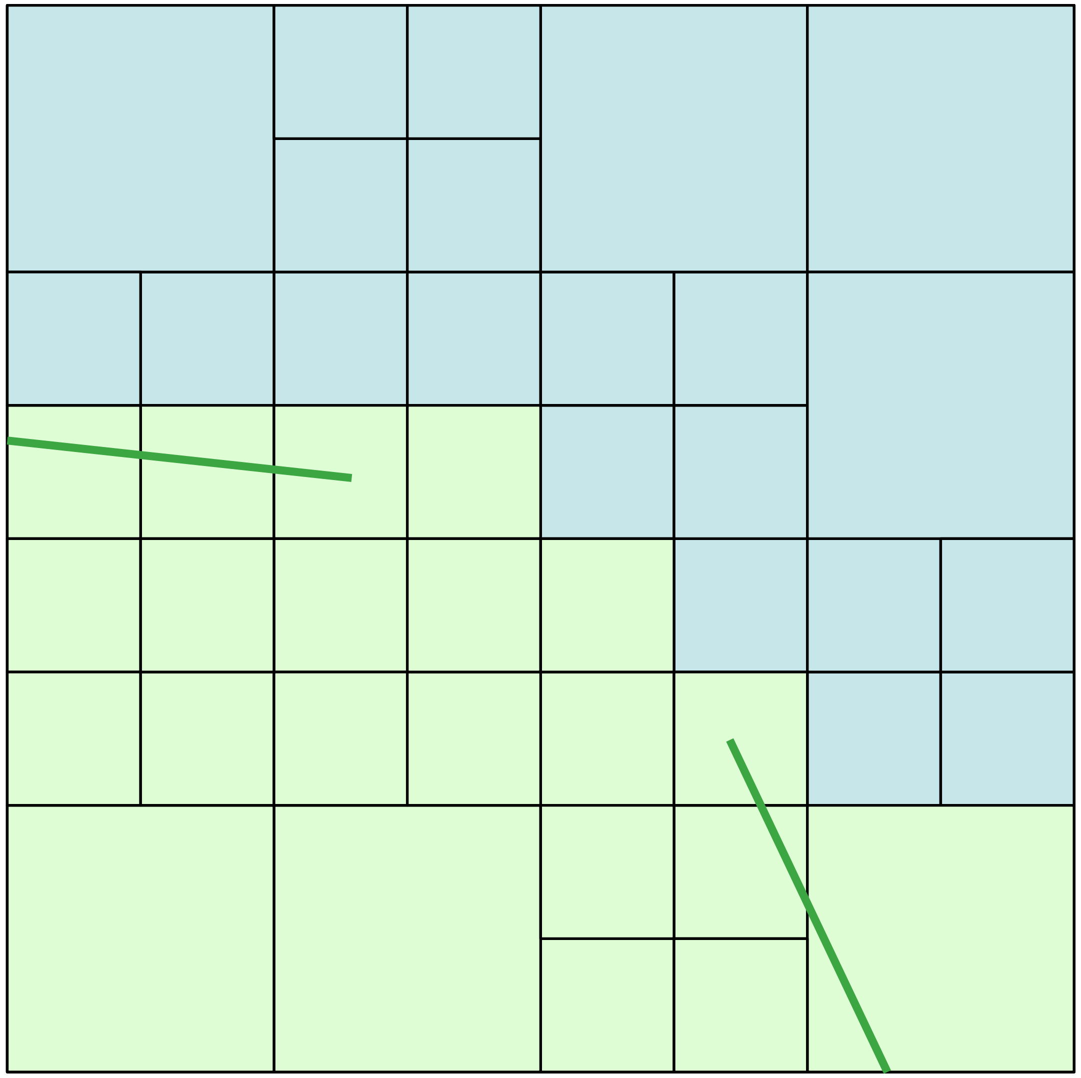
Closing Gaps

- Close gaps by dilating the boundary voxels



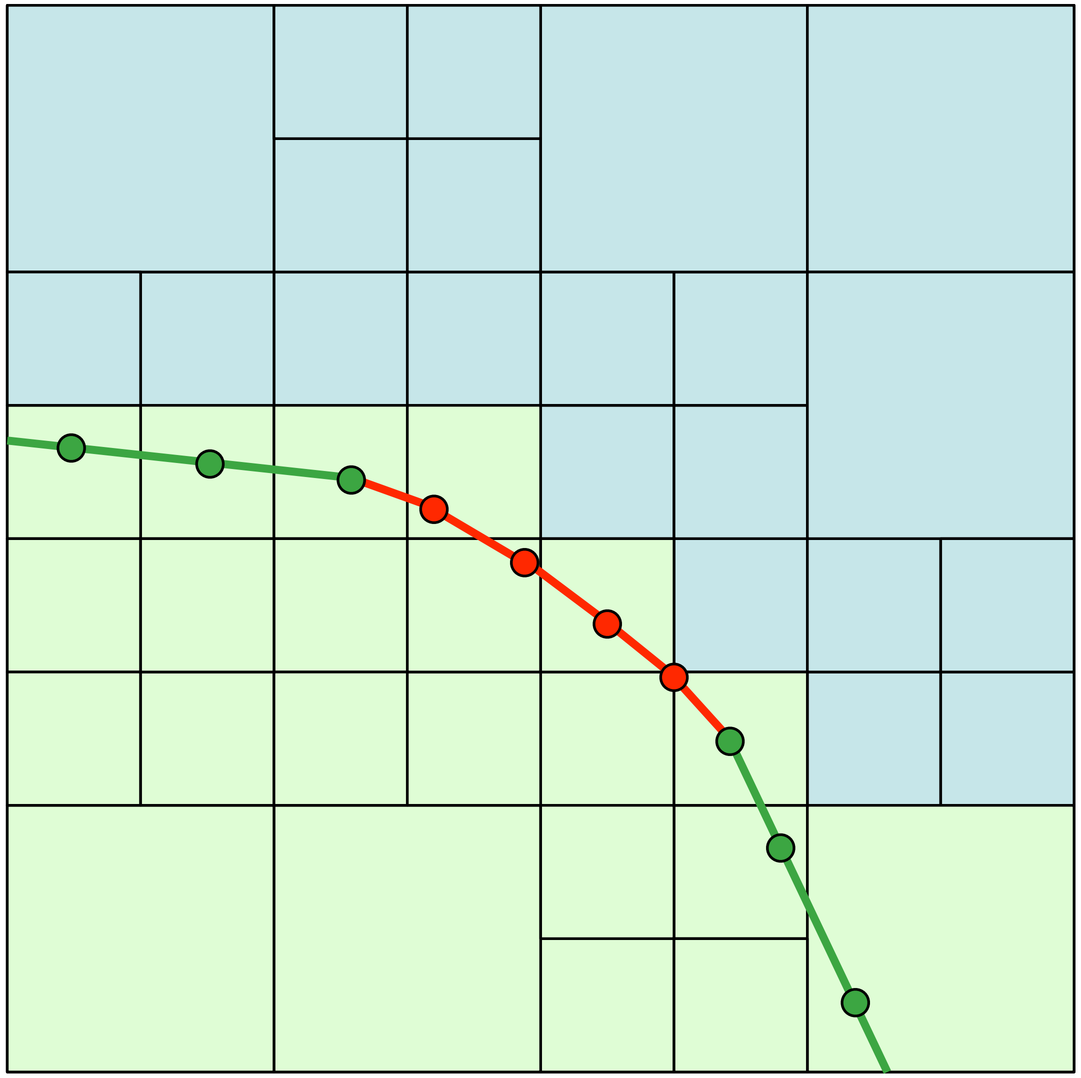
Determine Exterior

- Determine the exterior by flood filling & dilation

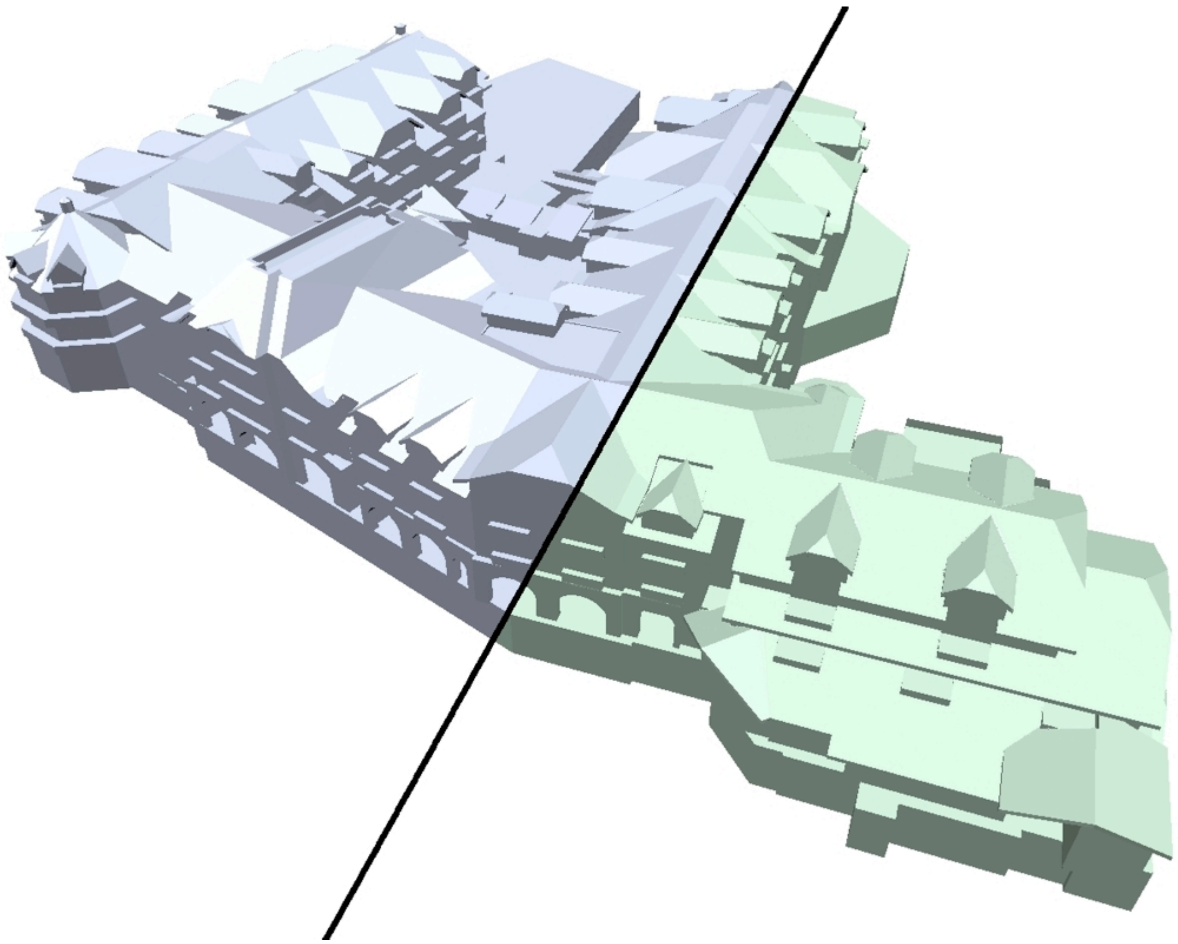


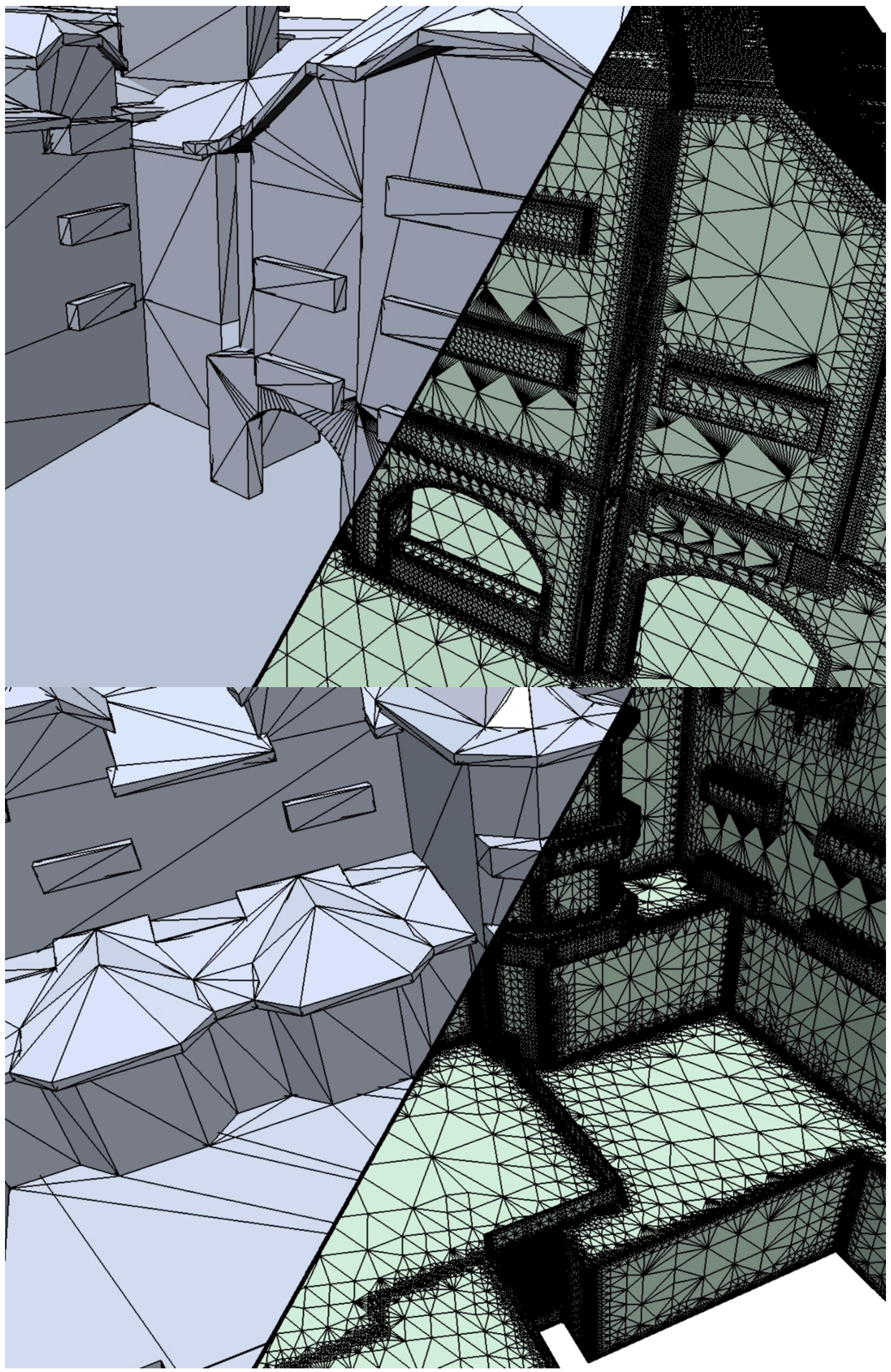
Extract the Surface

- Extract the surface by a variant of Dual Contouring

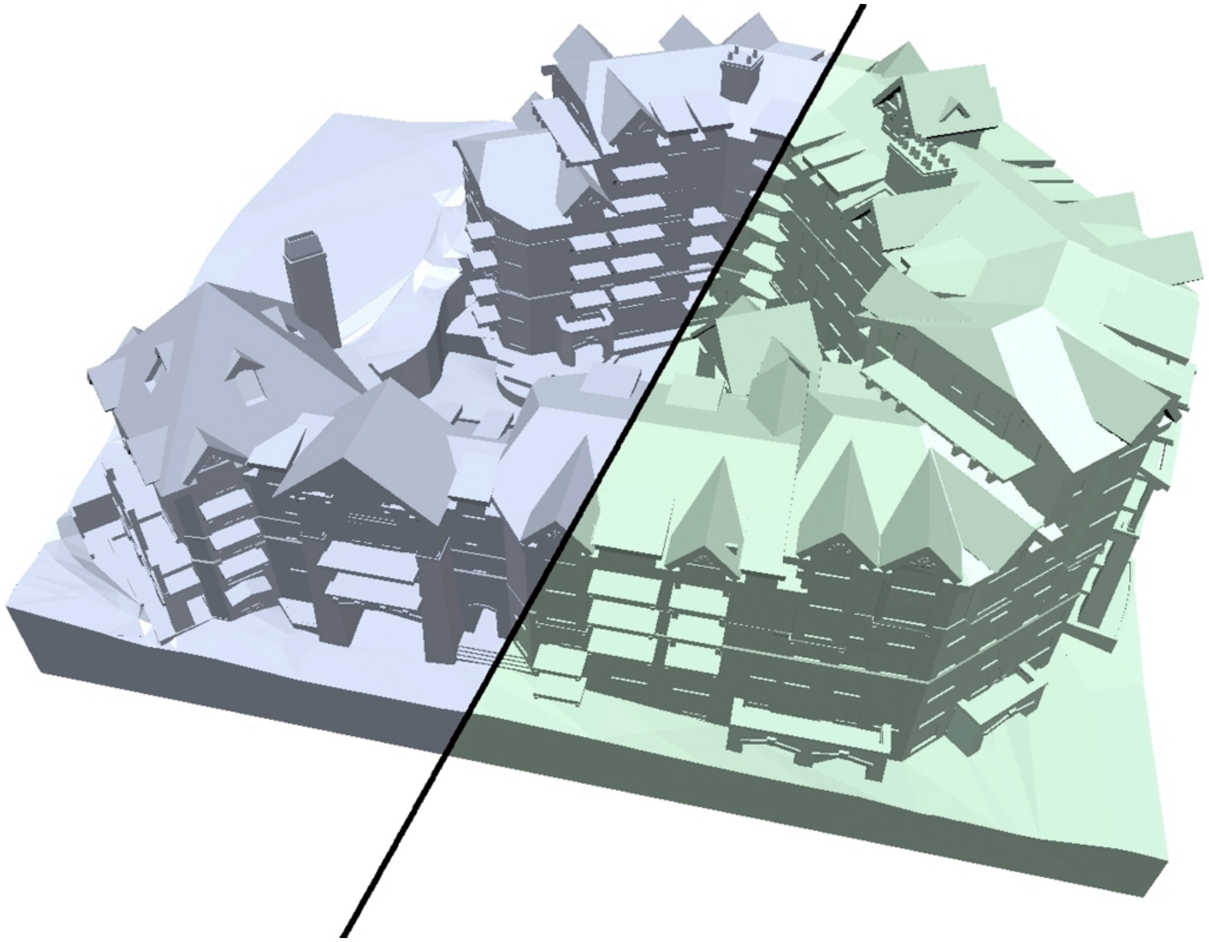


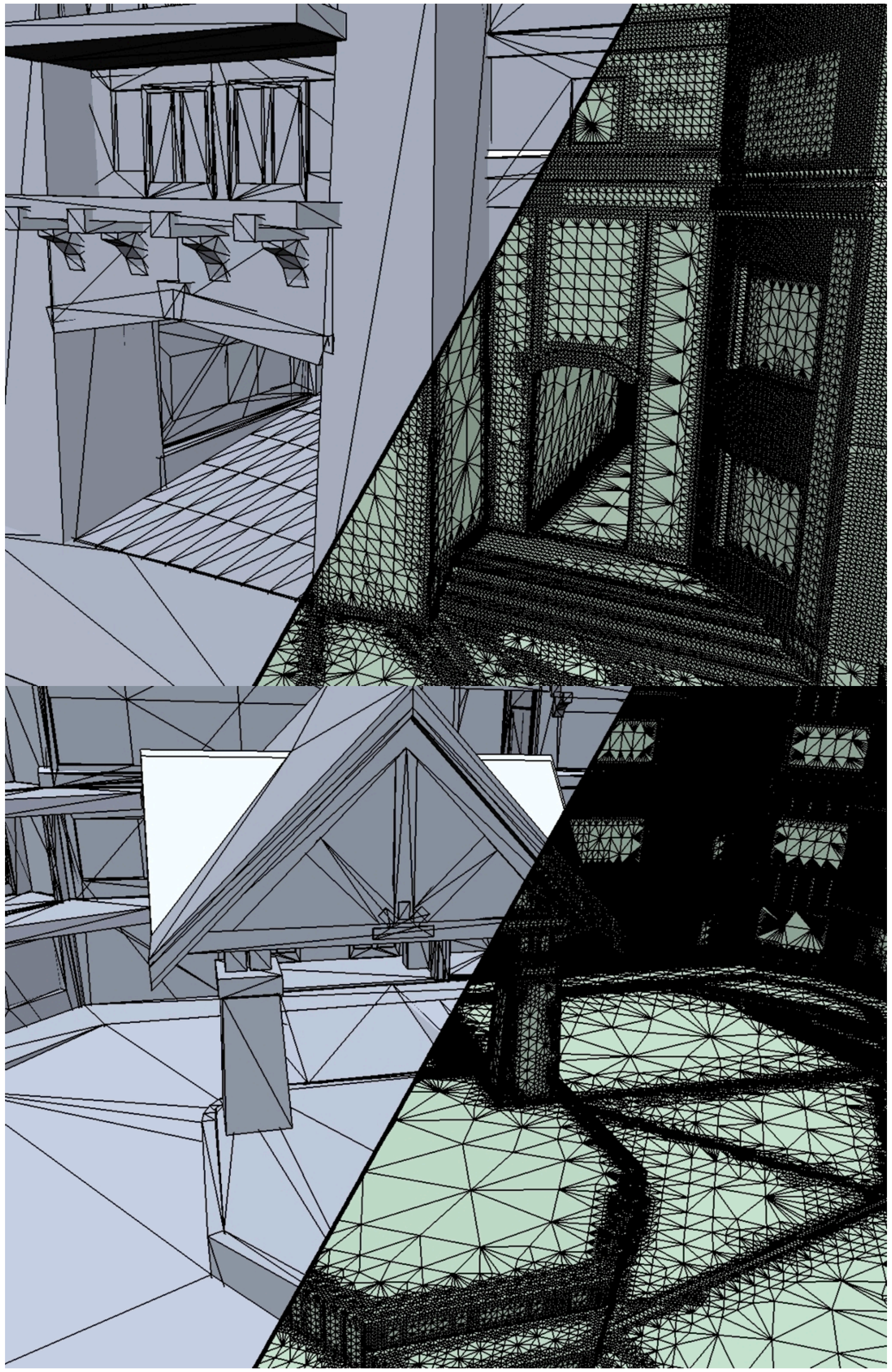
Results



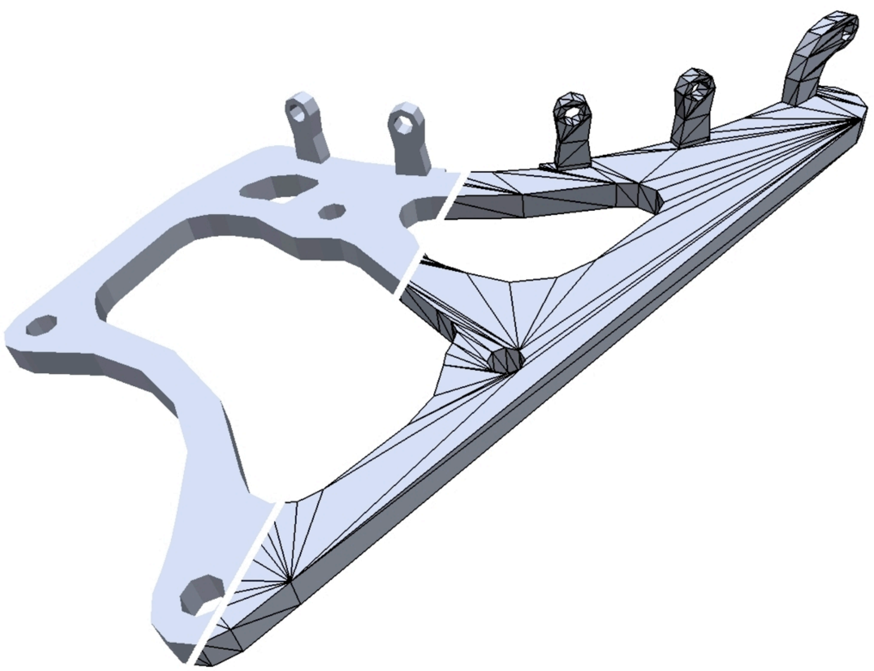


Results

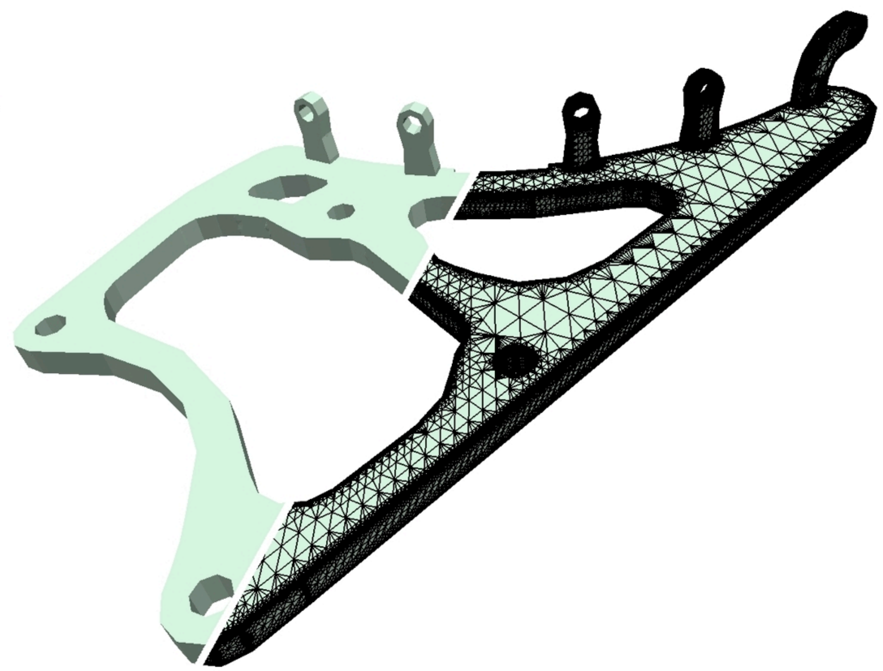




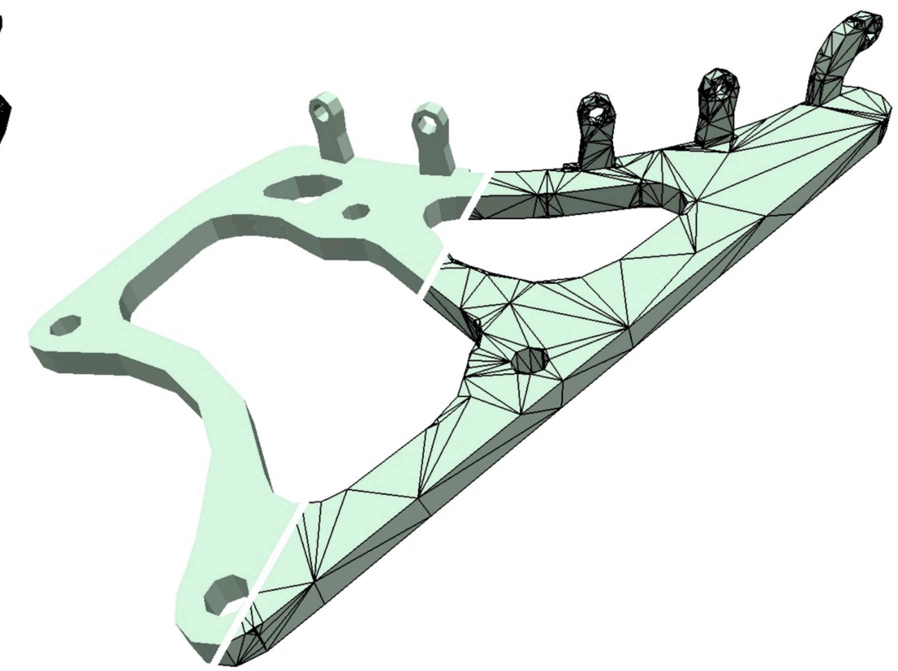
Results



original
1124 triangles

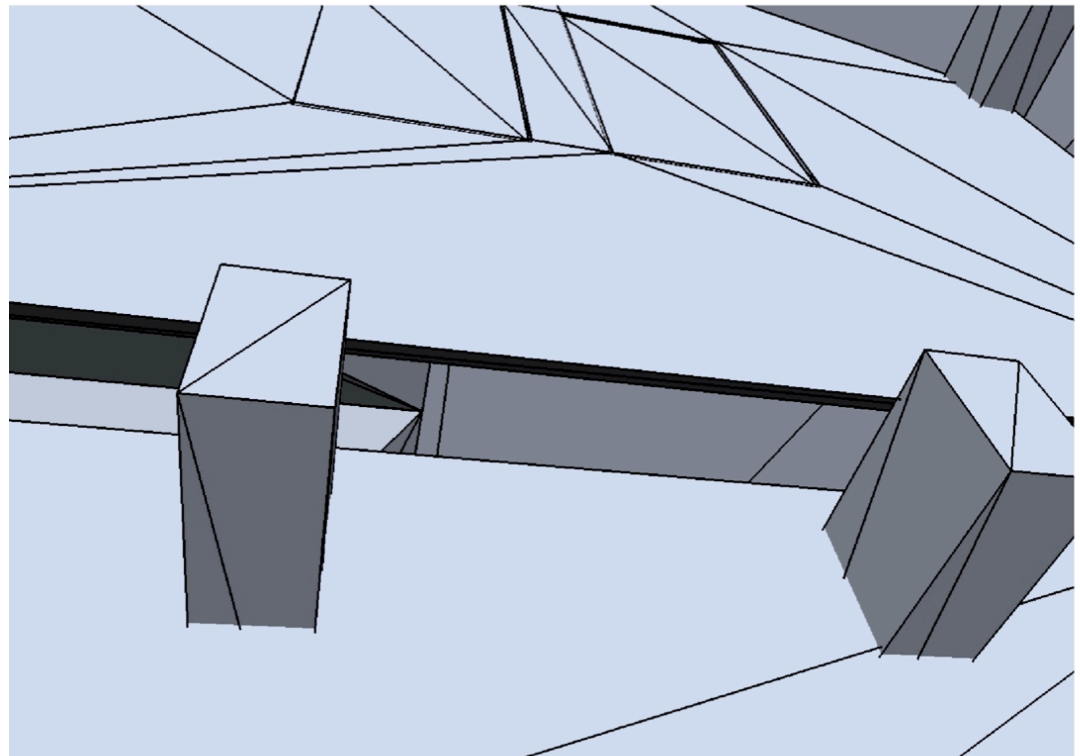


reconstruction
279892 triangles
(at 1000^3)



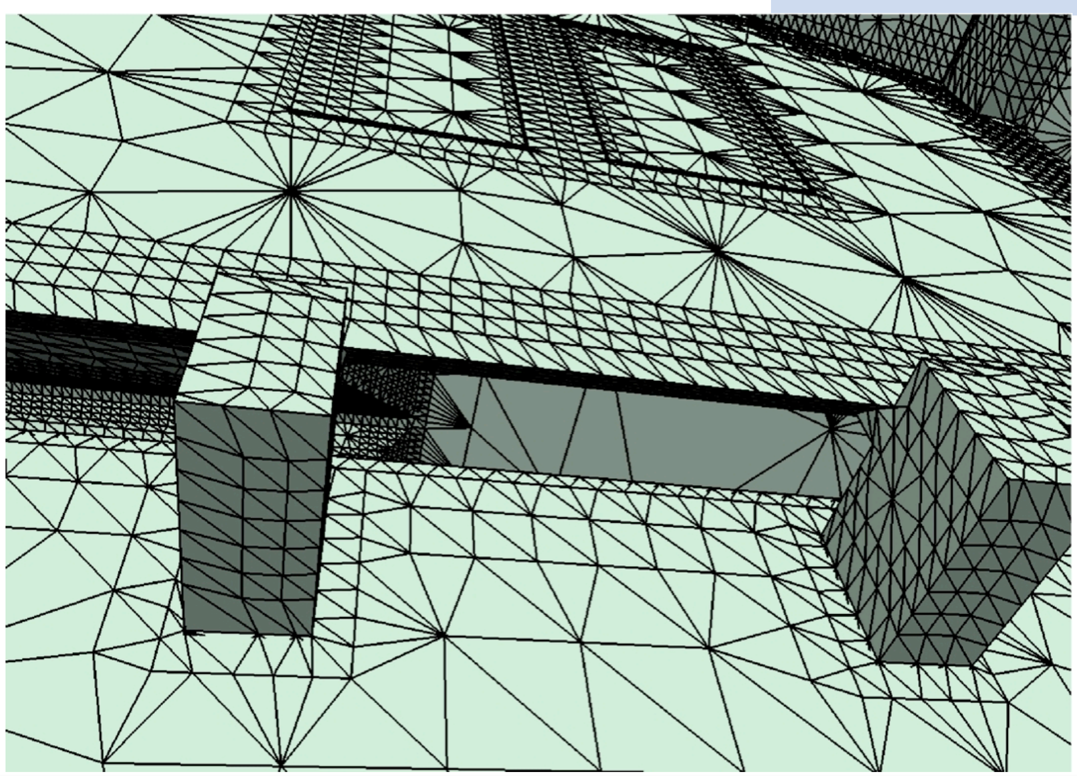
decimated
7018 triangles

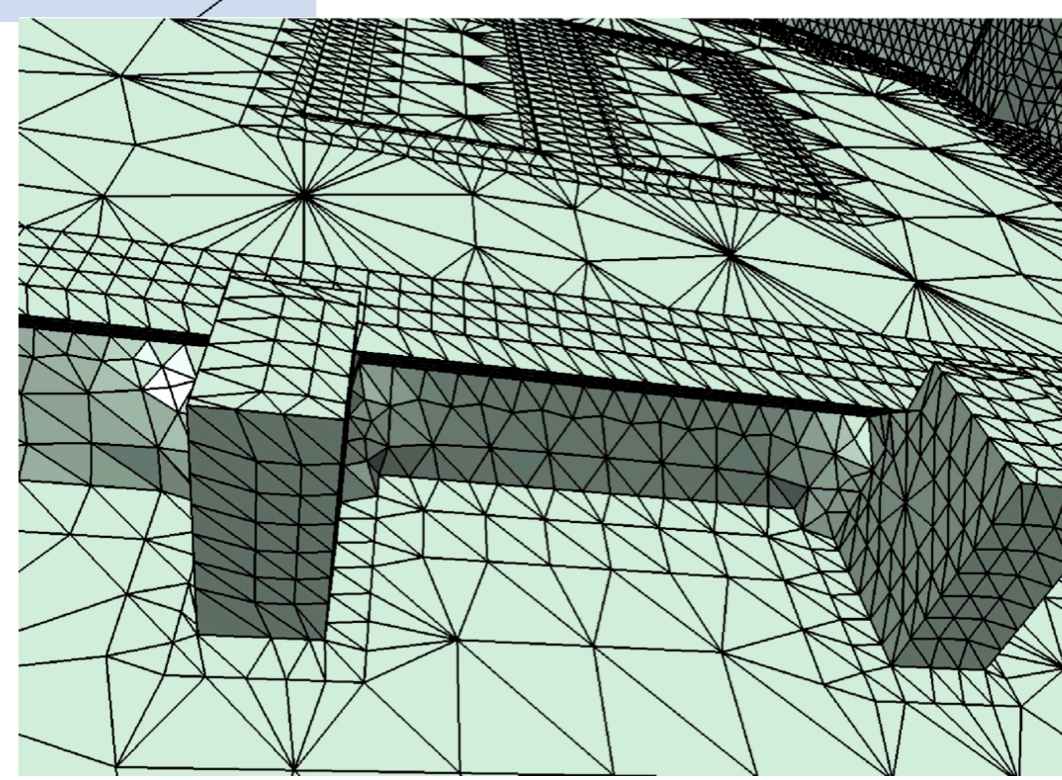
Results



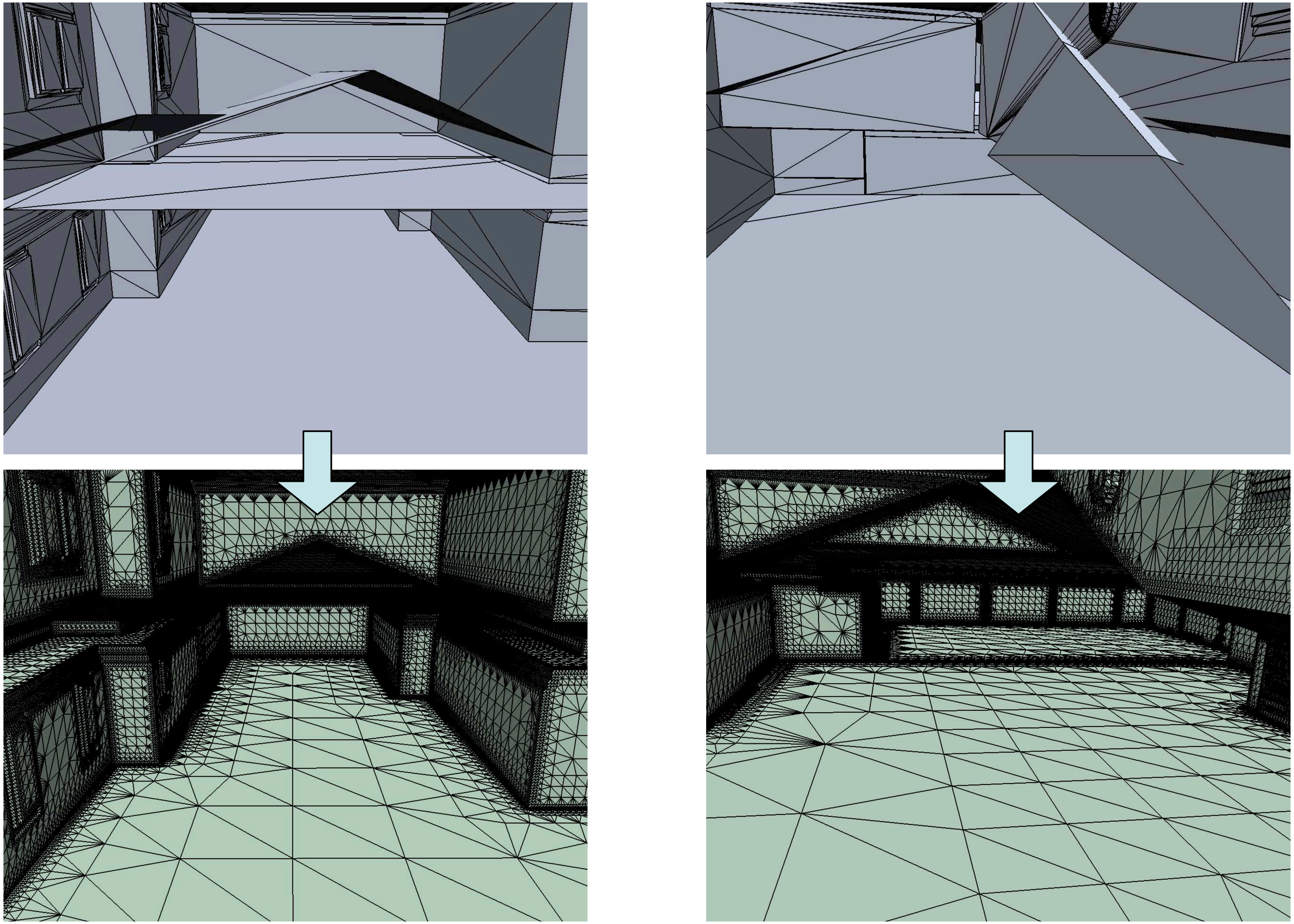
without
gap filling

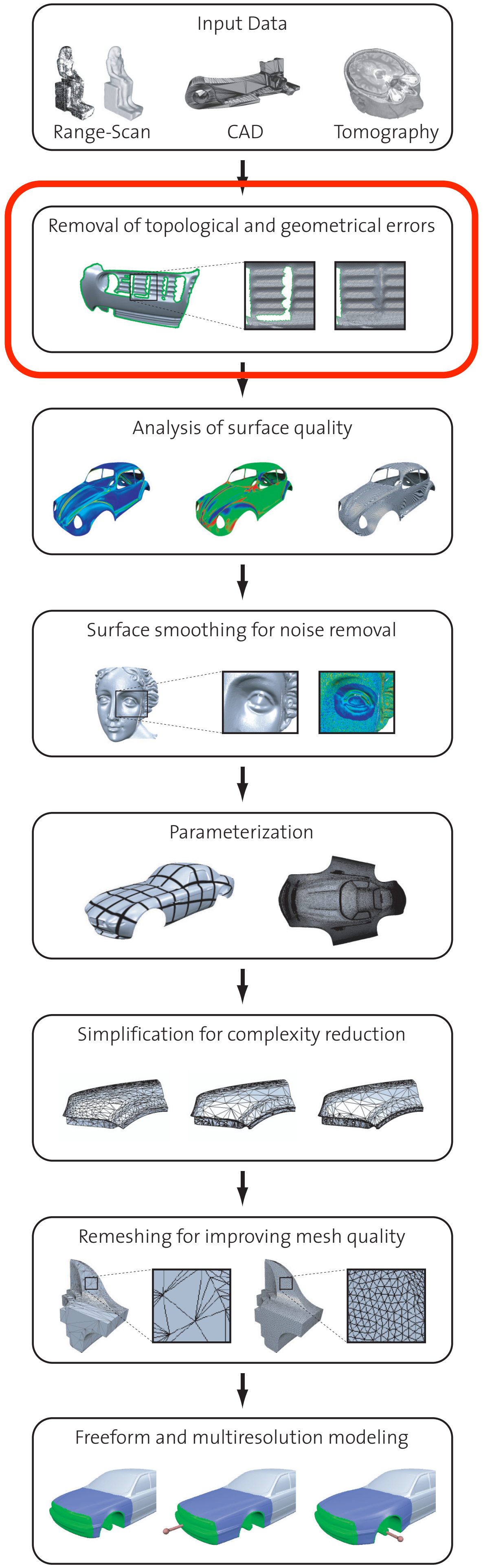
with
gap filling





Results



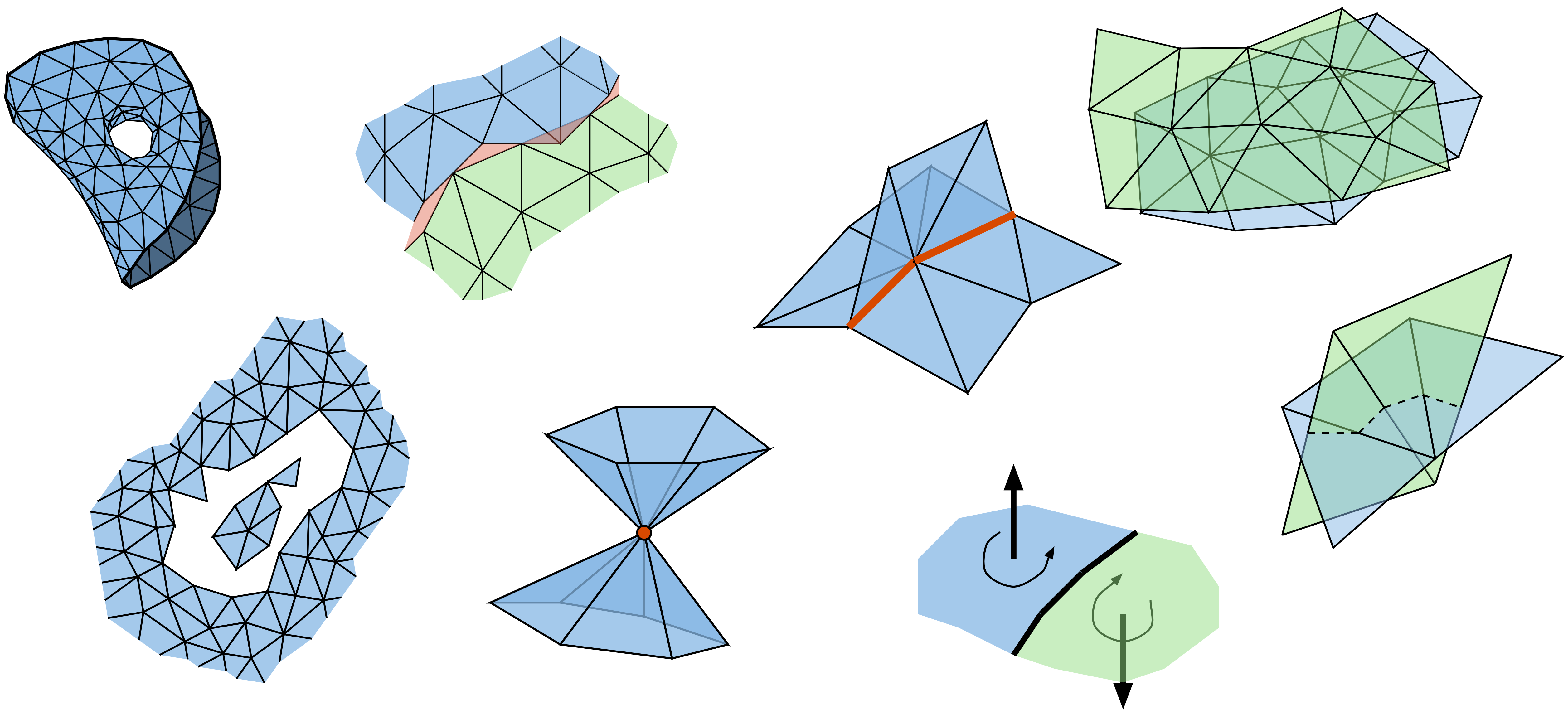


Model Repair

- Types of input
- Surface-oriented algorithms
  - Filling holes in meshes [Liepa 2003]
- Volumetric algorithms
  - Simplification and repair of polygonal models using volumetric techniques [Nooruddin and Turk 2003]
  - Automatic restoration of polygon models [Bischoff, Pavic, Kobbelt 2005]
- **Conclusion & outlook**

Conclusion

- Mesh repair to remove artifacts that arise in various types of input models



Conclusion

- Surface-oriented algorithms ...
  - fast, structure preserving
  - often not robust, need user interaction and cannot give quality guarantees on the output
- Volumetric algorithms ...
  - use an intermediate volumetric representation and thus produce guaranteed watertight meshes
  - suffer from sampling problems (aliasing)

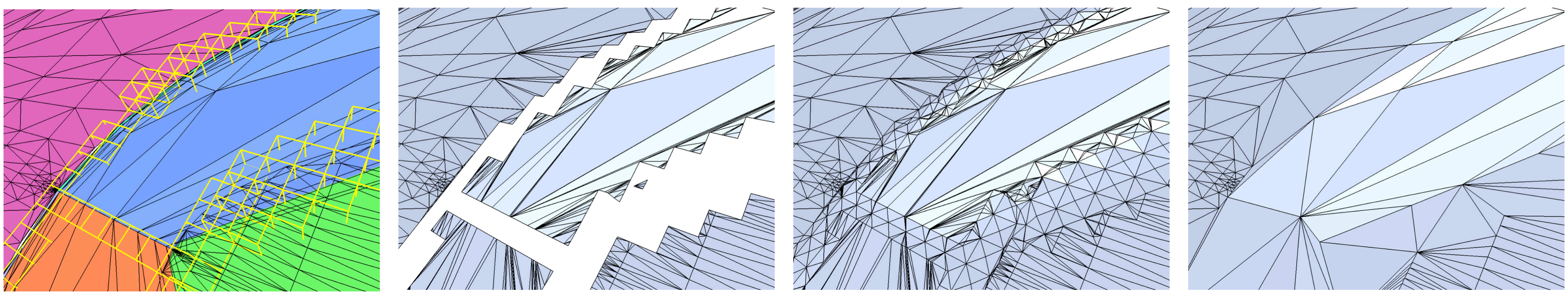
Outlook

- Surface-oriented
- Volumetric

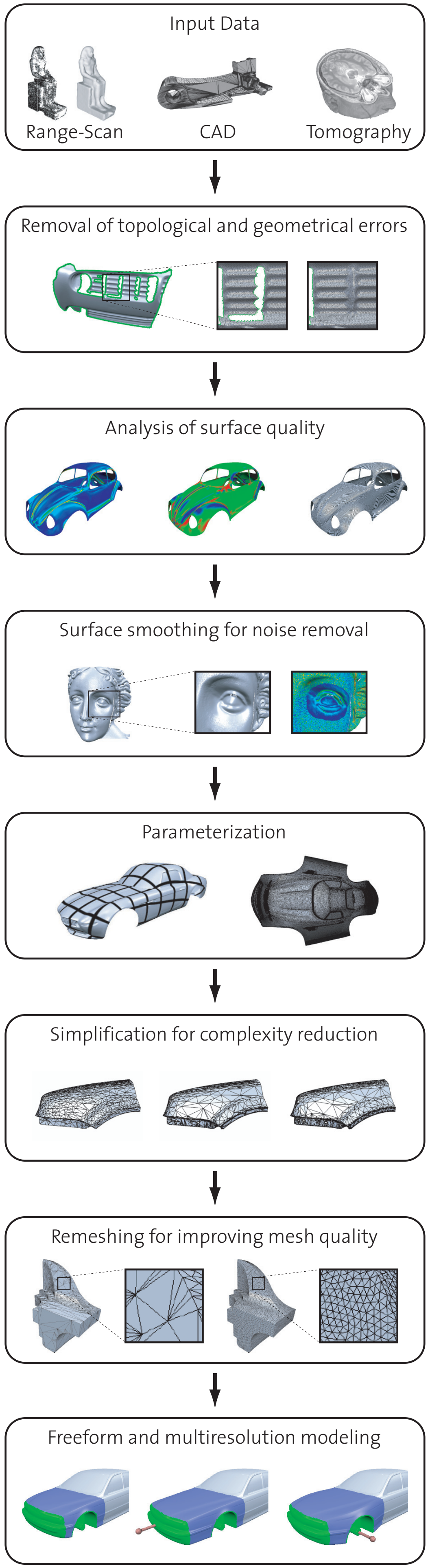
–	Bøhn, Wozny: Automatic CAD Model Repair: Shell-Closure.	1992
–	Mäkelä, Dolenc: Some Efficient Procedures for Correcting Triangulated Models.	1993
–	Turk, Levoy: Zippered Polygon Meshes from Range Images.	1994
–	Barequet, Sharir: Filling Gaps in the Boundary of a Polyhedron.	1995
–	Curless, Levoy: A Volumetric Method for Building Complex Models from Range Images.	1996
–	Barequet, Kumar: Repairing CAD Models.	1997
–	Murali, Funkhouser. Consistent Solid and Boundary Representations.	1997
–	Guéziec, Taubin, Lazarus, Horn: Cutting and Stitching: [...]	2001
–	Guskov, Wood: Topological Noise Removal.	2001
–	Borodin, Novotni, Klein: Progressive Gap Closing for Mesh Repairing.	2002
–	Davis, Marschner, Garr, Levoy: Filling Holes in Complex Surfaces Using Volumetric Diffusion.	2002
–	Liepa: Filling Holes in Meshes.	2003
–	Greß, Klein: Efficient Representation and Extraction of 2-Manifold Isosurfaces Using kd-Trees.	2003
–	Nooruddin, Turk: Simplification and Repair of Polygonal Models Using Volumetric Techniques.	2003
–	Borodin, Zachmann Klein: Consistent Normal Orientation for Polygonal Meshes.	2004
–	Ju: Robust Repair of Polygonal Models.	2004
–	Bischoff, Pavic, Kobbelt: Automatic Restoration of Polygon Models.	2005
–	Podolak, Rusinkiewicz: Atomic Volumes for Mesh Completion.	2005
–	Shen, O'Brien, Shewchuk: Interpolating and Approximating Implicit Surfaces from Polygon Soup.	2005

Outlook

- My own (biased!) opinion: Hybrid algorithms that are ...
  - ... robust and
  - ... structure preserving



- Bischoff, Kobbelt: Structure Preserving CAD Model Repair. 2005



ETH
APPLIED
GEOMETRY
GROUP

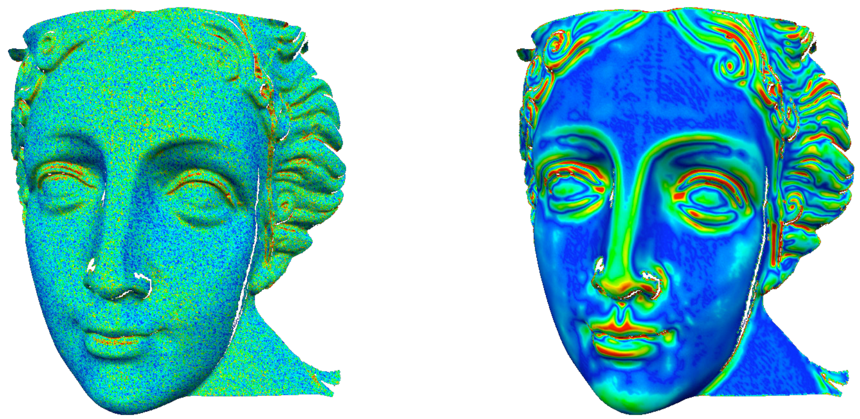
Mesh Quality

Mark Pauly

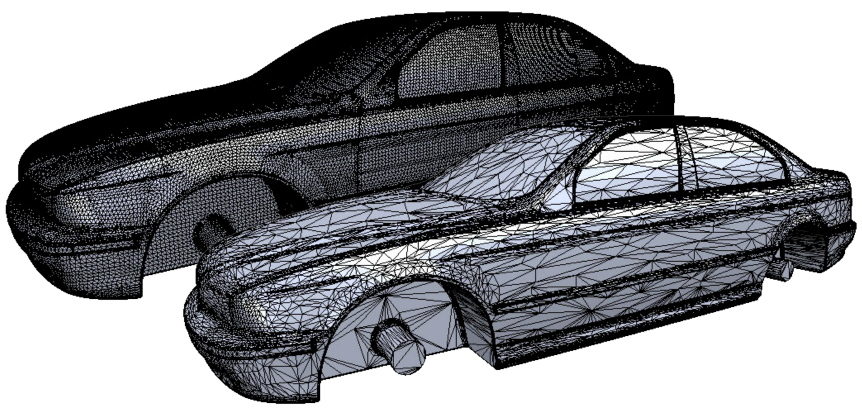
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Mesh Optimization

- Smoothness
  - ➡ Mesh smoothing



- Adaptive tessellation
  - ➡ Mesh decimation



- Triangle shape
  - ➡ Repair, remeshing

Outline

- Differential Geometry
  - Curvature
  - Fundamental Forms
- Laplace-Beltrami Operator
  - Discretizations
- Mesh Quality Criteria
  - Visual inspection

Differential Geometry

- Continuous surface

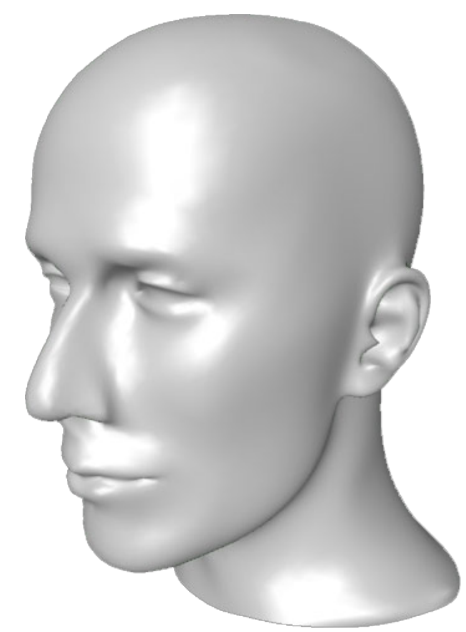
$$\mathbf{x}(u,v) = \left(\begin{array}{c} x(u,v) \\ y(u,v) \\ z(u,v) \end{array} \right), \ (u,v) \in \mathbb{R}^2$$

- Normal vector

$$\mathbf{n} = (\mathbf{x}_u \times \mathbf{x}_v) / \|\mathbf{x}_u \times \mathbf{x}_v\|$$

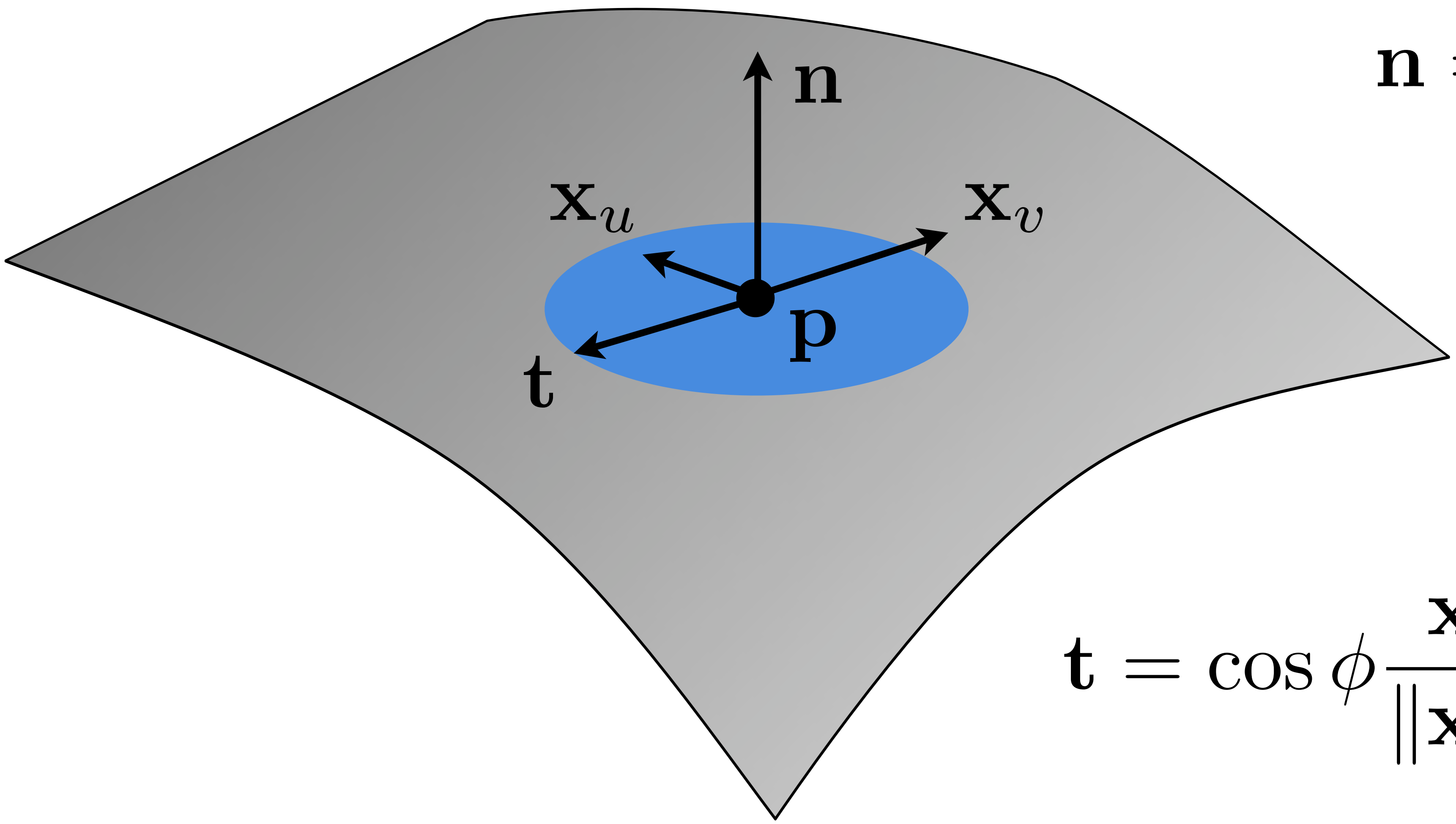
  – assuming regular parameterization, i.e.

$$\mathbf{x}_u \times \mathbf{x}_v \neq \mathbf{0}$$



Differential Geometry

- Normal Curvature

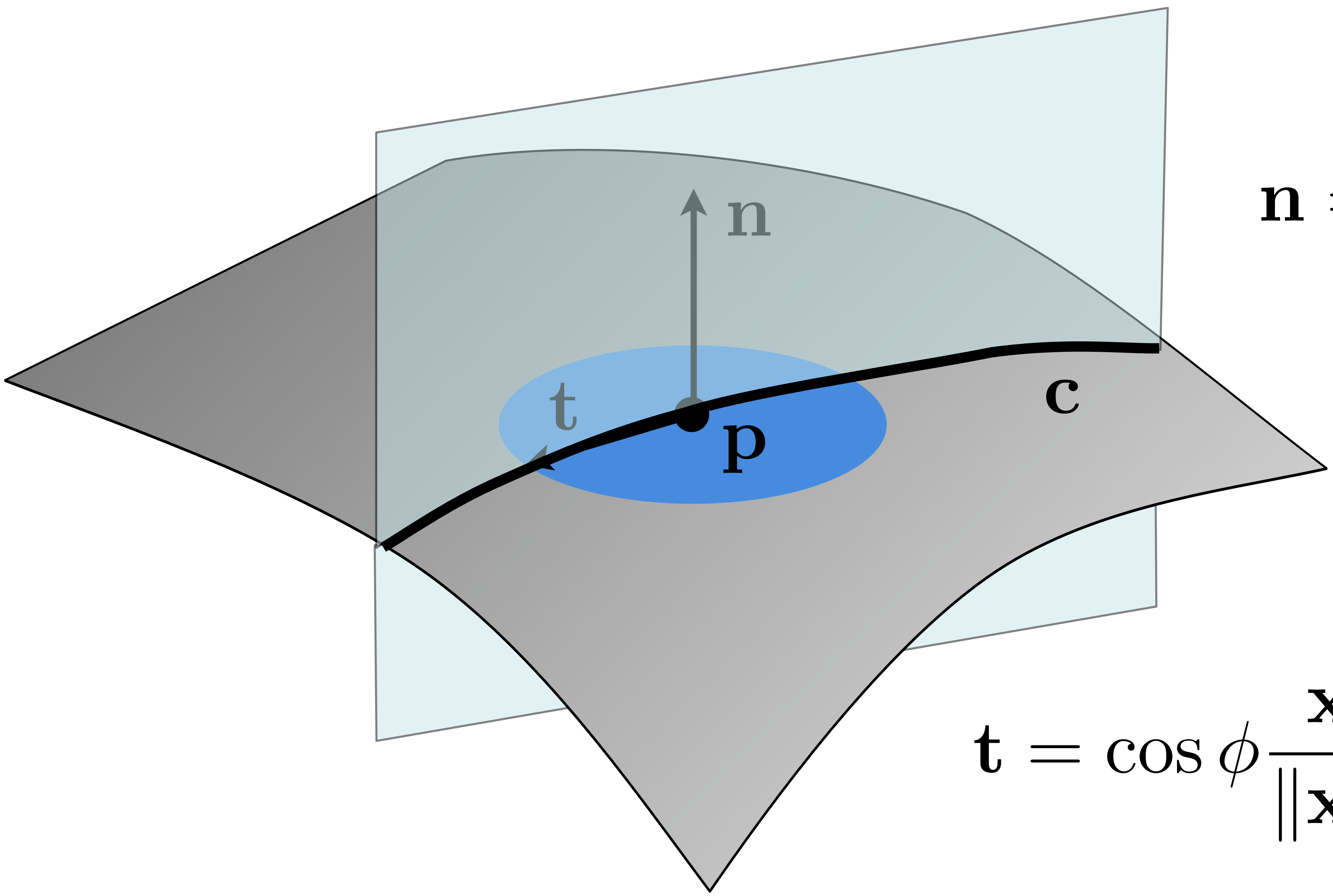


$$\mathbf{n} = \frac{\mathbf{x}_u \times \mathbf{x}_v}{\|\mathbf{x}_u \times \mathbf{x}_v\|}$$

$$\mathbf{t} = \cos\phi \frac{\mathbf{x}_u}{\|\mathbf{x}_u\|} + \sin\phi \frac{\mathbf{x}_v}{\|\mathbf{x}_v\|}$$

Differential Geometry

- Normal Curvature



$$\mathbf{n} = \frac{\mathbf{x}_u \times \mathbf{x}_v}{\|\mathbf{x}_u \times \mathbf{x}_v\|}$$

$$\mathbf{t} = \cos\phi \frac{\mathbf{x}_u}{\|\mathbf{x}_u\|} + \sin\phi \frac{\mathbf{x}_v}{\|\mathbf{x}_v\|}$$

Differential Geometry

- Principal Curvatures
  - maximum curvature $\kappa_1 = \max_{\phi} \kappa_n(\phi)$
  - minimum curvature $\kappa_2 = \min_{\phi} \kappa_n(\phi)$

- Euler Theorem: $\kappa_n(\bar{\mathbf{t}}) = \kappa_n(\phi) = \kappa_1 \cos^2 \phi + \kappa_2 \sin^2 \phi$

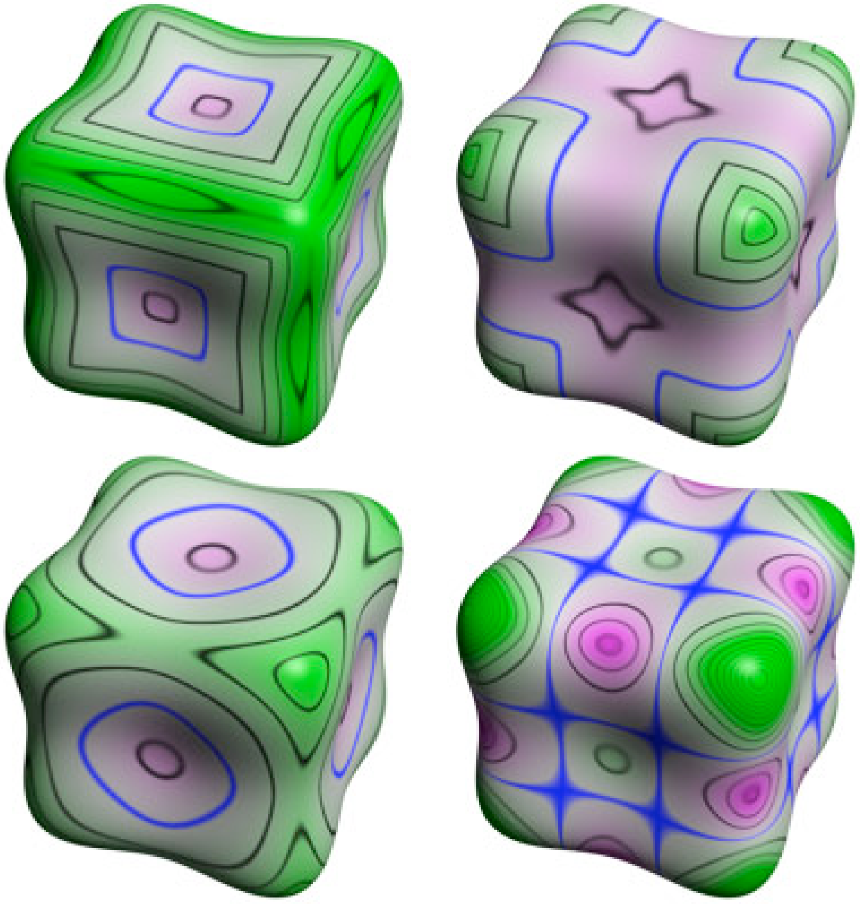
- Mean Curvature $H = \frac{\kappa_1 + \kappa_2}{2} = \frac{1}{2\pi} \int_0^{2\pi} \kappa_n(\phi) d\phi$

- Gaussian Curvature $K = \kappa_1 \cdot \kappa_2$

Differential Geometry

- Curvatures

$$\kappa_1 = \max_{\phi} \kappa_n(\phi)$$



$$\kappa_2 = \min_{\phi} \kappa_n(\phi)$$

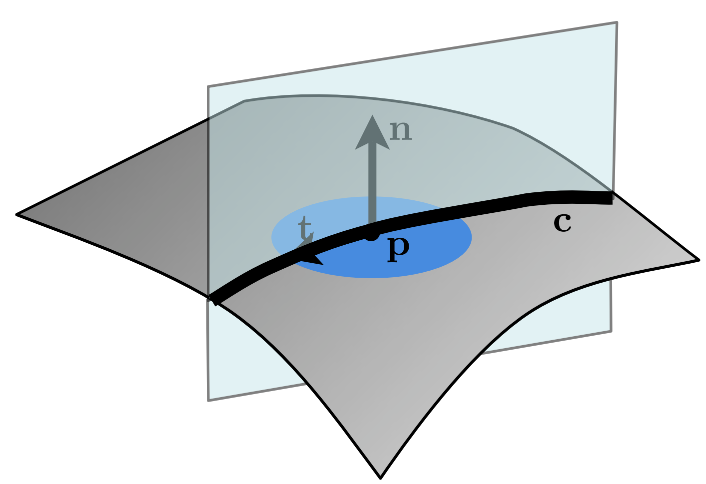
$$H = \frac{1}{2}(\kappa_1 + \kappa_2)$$

$$K = \kappa_1 \cdot \kappa_2$$

Differential Geometry

- Normal curvature is defined as curvature of the normal curve $\mathbf{c} \in \mathbf{x}(u,v)$ at a point $\mathbf{p} \in \mathbf{c}$
- Can be expressed in terms of fundamental forms as

$$\kappa_n(\bar{\mathbf{t}}) = \frac{\bar{\mathbf{t}}^T \mathbf{II}\, \bar{\mathbf{t}}}{\bar{\mathbf{t}}^T \mathbf{I}\, \bar{\mathbf{t}}} = \frac{ea^2 + 2fab + gb^2}{Ea^2 + 2Fab + Gb^2}$$



$$\mathbf{t} = a\mathbf{x}_u + b\mathbf{x}_v$$

Differential Geometry

- First fundamental form

$$\mathbf{I} = \begin{bmatrix} E & F \\ F & G \end{bmatrix} := \begin{bmatrix} \mathbf{x}_u^T \mathbf{x}_u & \mathbf{x}_u^T \mathbf{x}_v \\ \mathbf{x}_u^T \mathbf{x}_v & \mathbf{x}_v^T \mathbf{x}_v \end{bmatrix}$$

- Second fundamental form

$$\mathbf{II} = \begin{bmatrix} e & f \\ f & g \end{bmatrix} := \begin{bmatrix} \mathbf{x}_{uu}^T \mathbf{n} & \mathbf{x}_{uv}^T \mathbf{n} \\ \mathbf{x}_{uv}^T \mathbf{n} & \mathbf{x}_{vv}^T \mathbf{n} \end{bmatrix}$$

Differential Geometry

- $\mathbf{I}$ and $\mathbf{II}$ allow to measure
  - length, angles, area, curvature
  - arc element

$$ds^2 = Edu^2 + 2Fdudv + Gdv^2$$

  - area element

$$dA = \sqrt{EG - F^2}dudv$$

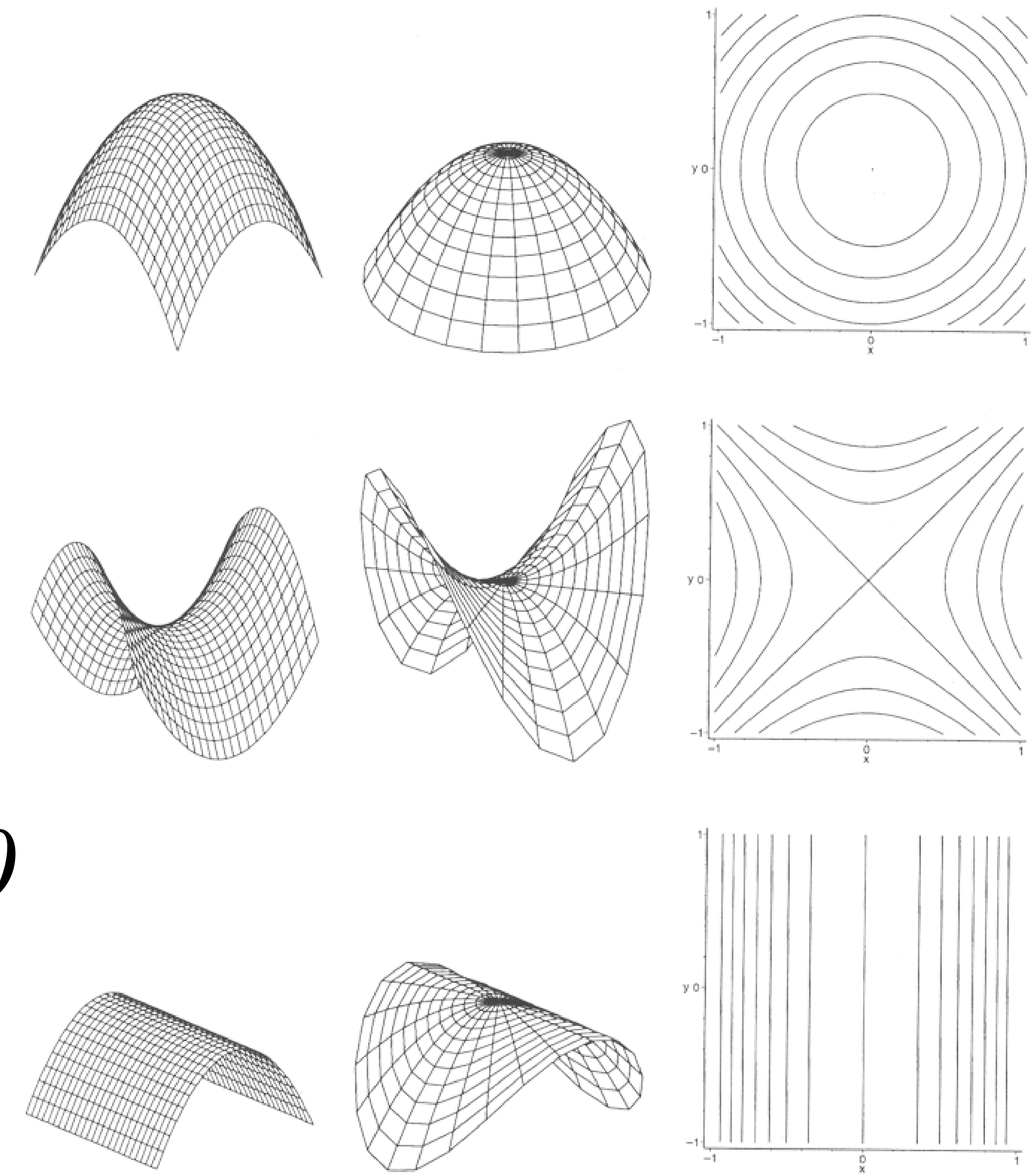
Differential Geometry

- Intrinsic geometry: Properties of the surface that only depend on the first fundamental form
  - length
  - angles
  - Gaussian curvature (Theorema Egregium)

$$K = \lim_{r \to 0} \frac{6\pi r - 3C(r)}{\pi r^3}$$

Differential Geometry

- A point $\mathbf{x}$ on the surface is called
  - *elliptic*, if $K > 0$
  - *parabolic*, if $K = 0$
  - *hyperbolic*, if $K < 0$
  - *umbilical*, if $\kappa_1 = \kappa_2$



- Developable surface $\Leftrightarrow K = 0$

Laplace Operator

Laplace operator

gradient operator

2nd partial derivatives

$$\Delta f = \mathrm{div} \nabla f = \sum_i \frac{\partial^2 f}{\partial x_i^2}$$

function in Euclidean space

divergence operator

Cartesian coordinates

Laplace-Beltrami Operator

- Extension of Laplace to functions on manifolds

Laplace-
Beltrami

gradient
operator

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

function on
manifold $\mathcal{S}$

divergence
operator

Laplace-Beltrami Operator

- Extension of Laplace to functions on manifolds

Laplace-Beltrami

gradient operator

mean curvature

$$\Delta_{\mathcal{S}}\mathbf{x} = \mathrm{div}_{\mathcal{S}}\, \nabla_{\mathcal{S}}\mathbf{x} = -2H\mathbf{n}$$

coordinate function

divergence operator

surface normal

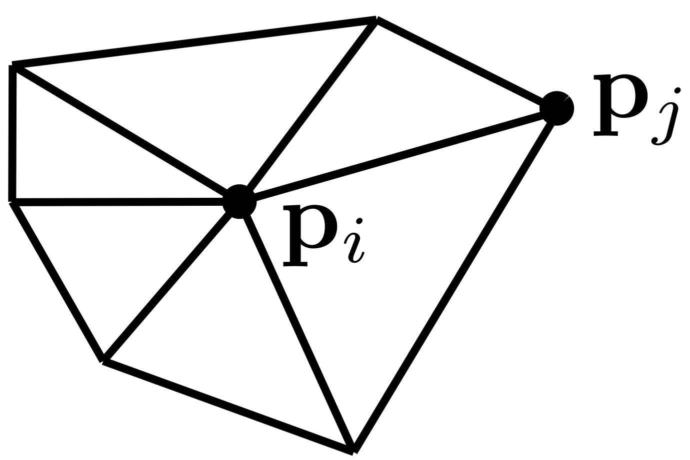
Discrete Differential Operators

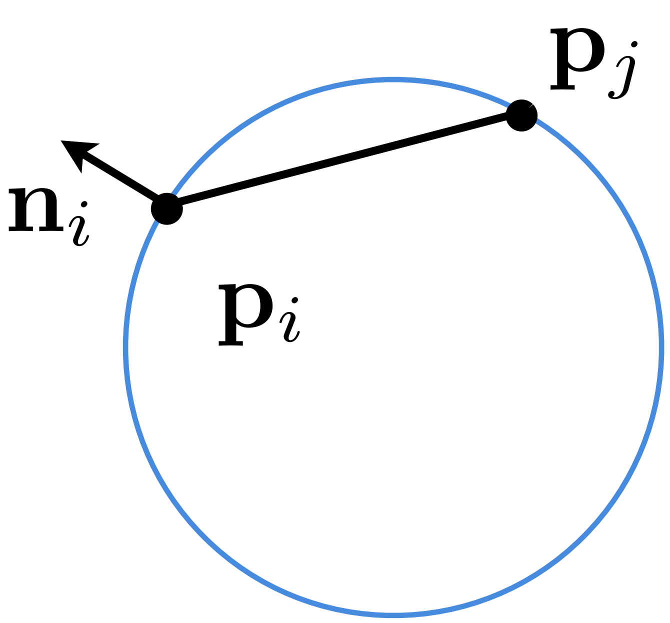
- Assumption: Meshes are piecewise linear approximations of smooth surfaces
- Approach: Approximate differential properties at point x as spatial average over local mesh neighborhood $N(x)$, where typically
  - x = mesh vertex
  - $N(x)$ = n-ring neighborhood or local geodesic ball

Discrete Normal Curvature

- Normal curvature along tangent direction

$$\kappa_{ij} \;=\; 2\frac{(\mathbf{p}_j - \mathbf{p}_i)\mathbf{n}_i}{\|\mathbf{p}_j - \mathbf{p}_i\|^2}$$





Discrete Laplace-Beltrami

- Uniform discretization

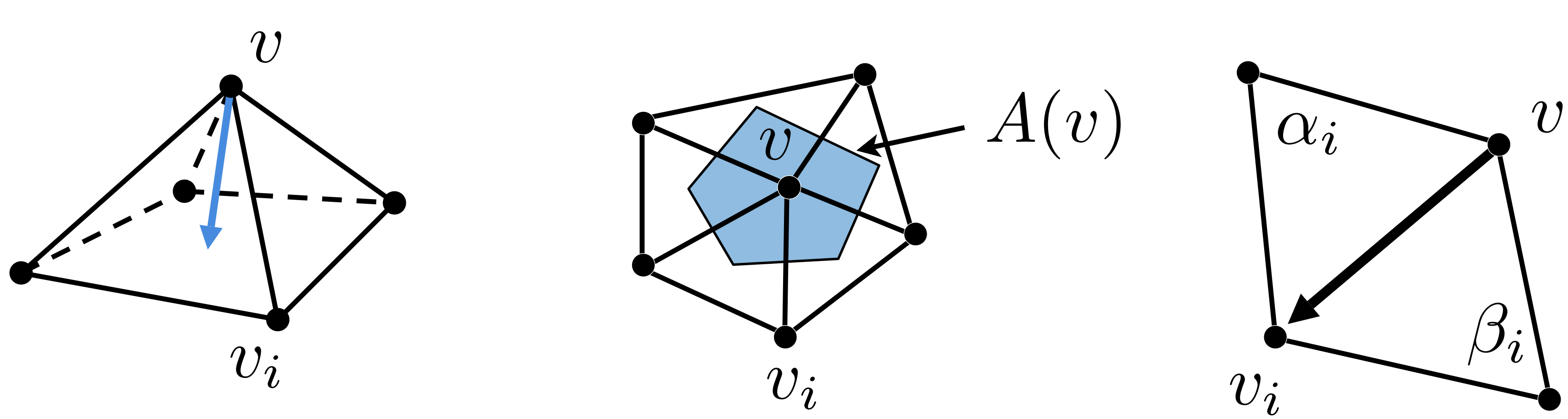
$$\Delta_{uni} f\left(v\right) \ := \ \frac{1}{\left|\mathcal{N}_1\left(v\right)\right|} \sum_{v_i \in \mathcal{N}_1(v)} \left(f\left(v_i\right) - f\left(v\right)\right)$$

  - depends only on connectivity → simple and efficient
  - bad approximation for irregular triangulations

Discrete Laplace-Beltrami

- Cotangent formula

$$\Delta_{\mathcal{S}} f(v) := \frac{2}{A(v)} \sum_{v_i \in \mathcal{N}_1(v)} (\cot \alpha_i + \cot \beta_i) (f(v_i) - f(v))$$



Discrete Laplace-Beltrami

- Cotangent formula

$$\Delta_{\mathcal{S}} f(v) := \frac{2}{A(v)} \sum_{v_i \in \mathcal{N}_1(v)} (\cot \alpha_i + \cot \beta_i) \left(f(v_i) - f(v) \right)$$

- Problems
  – negative weights
  – depends on triangulation

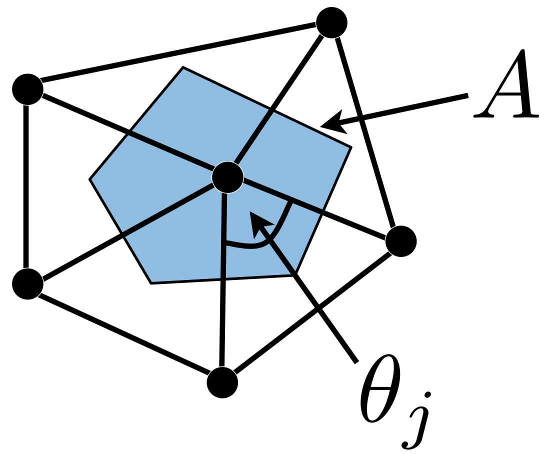
Discrete Curvatures

- Mean curvature

$$H = \|\Delta_{\mathcal{S}} \mathbf{x}\|$$

- Gaussian curvature

$$G = (2\pi - \sum_j \theta_j)/A$$

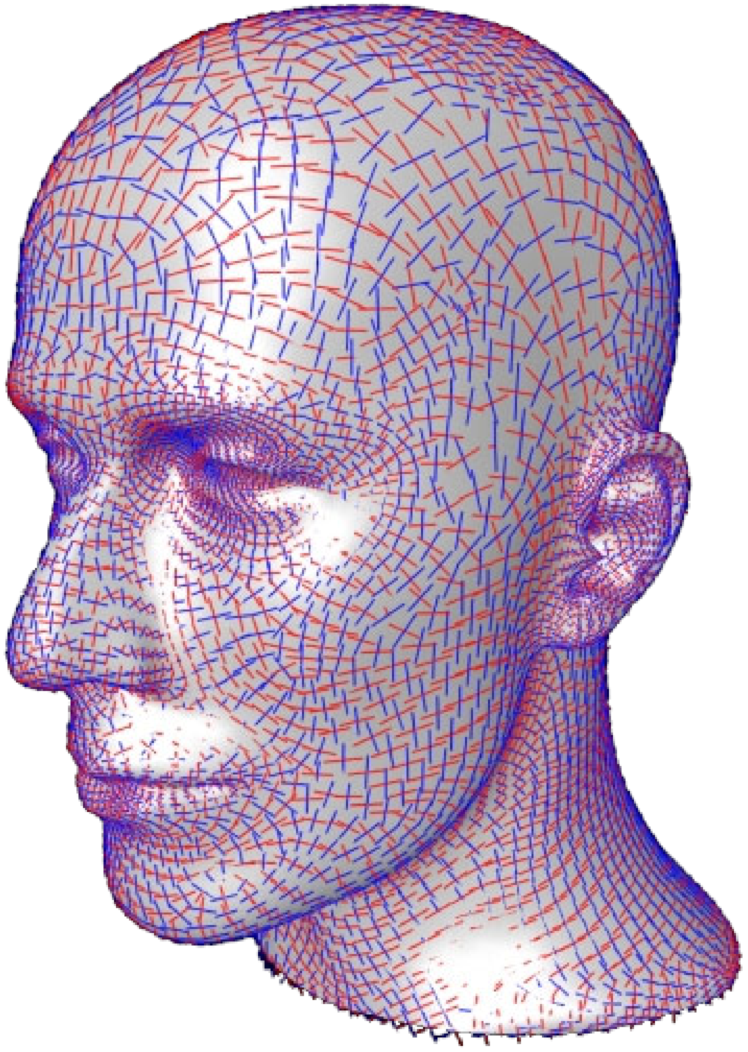


- Principal curvatures

$$\kappa_1 = H + \sqrt{H^2 - G} \qquad \kappa_2 = H - \sqrt{H^2 - G}$$

Links & Literature

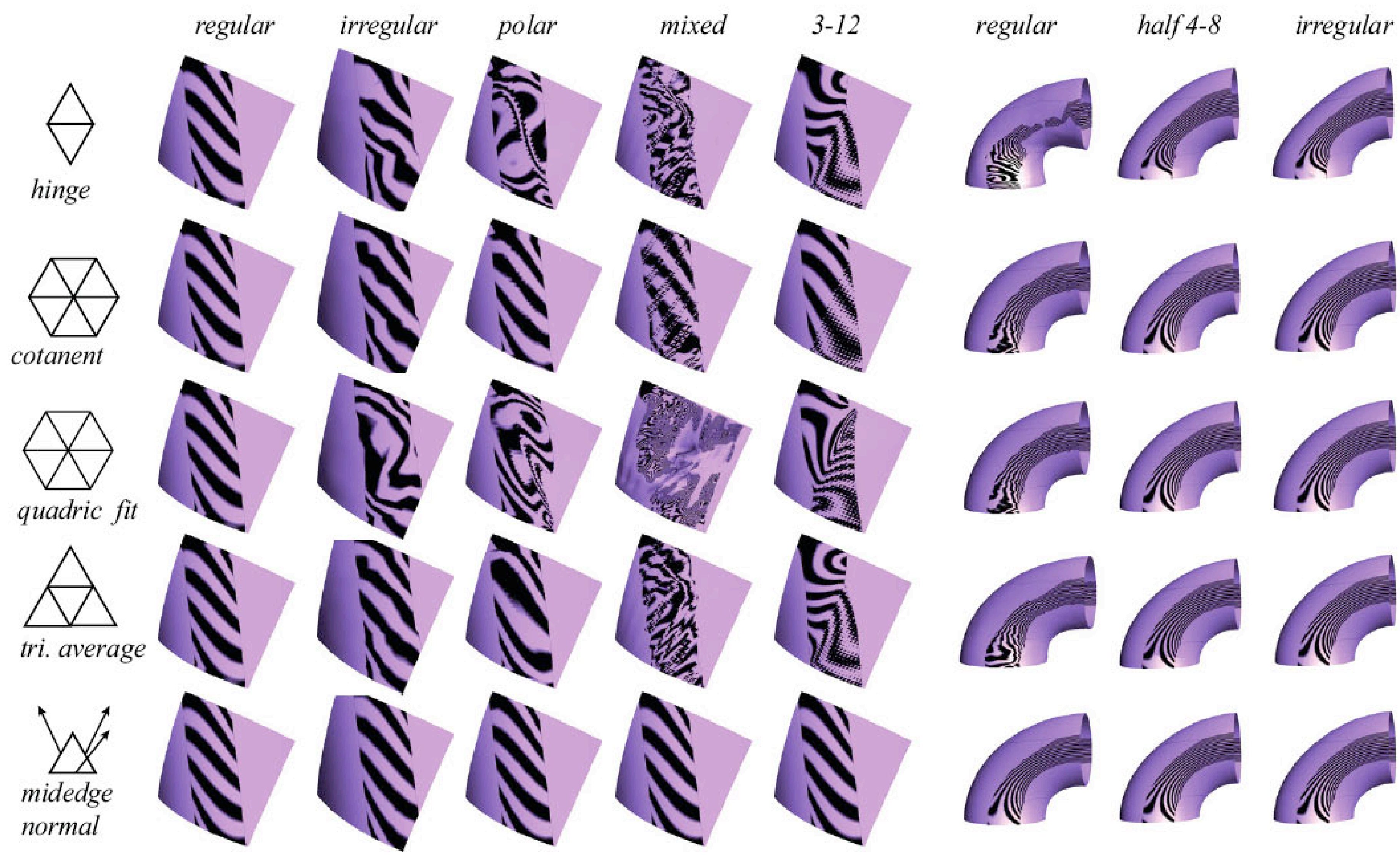
- P. Alliez: *Estimating Curvature Tensors on Triangle Meshes* (source code)
  - http://www-sop.inria.fr/geometrica/team/Pierre.Alliez/demos/curvature/



principal directions

Links & Literature

- Grinspun et al.: *Computing discrete shape operators on general meshes,* Eurographics 2006



Mesh Quality

- Smoothness
  - continuous differentiability of a surface (C^k)
- Fairness
  - aesthetic measure of “well-shapedness”
  - principle of simplest shape
  - fairness measures from physical models

$$\int_{\mathcal{S}} \kappa_1^2 + \kappa_2^2 \, dA$$

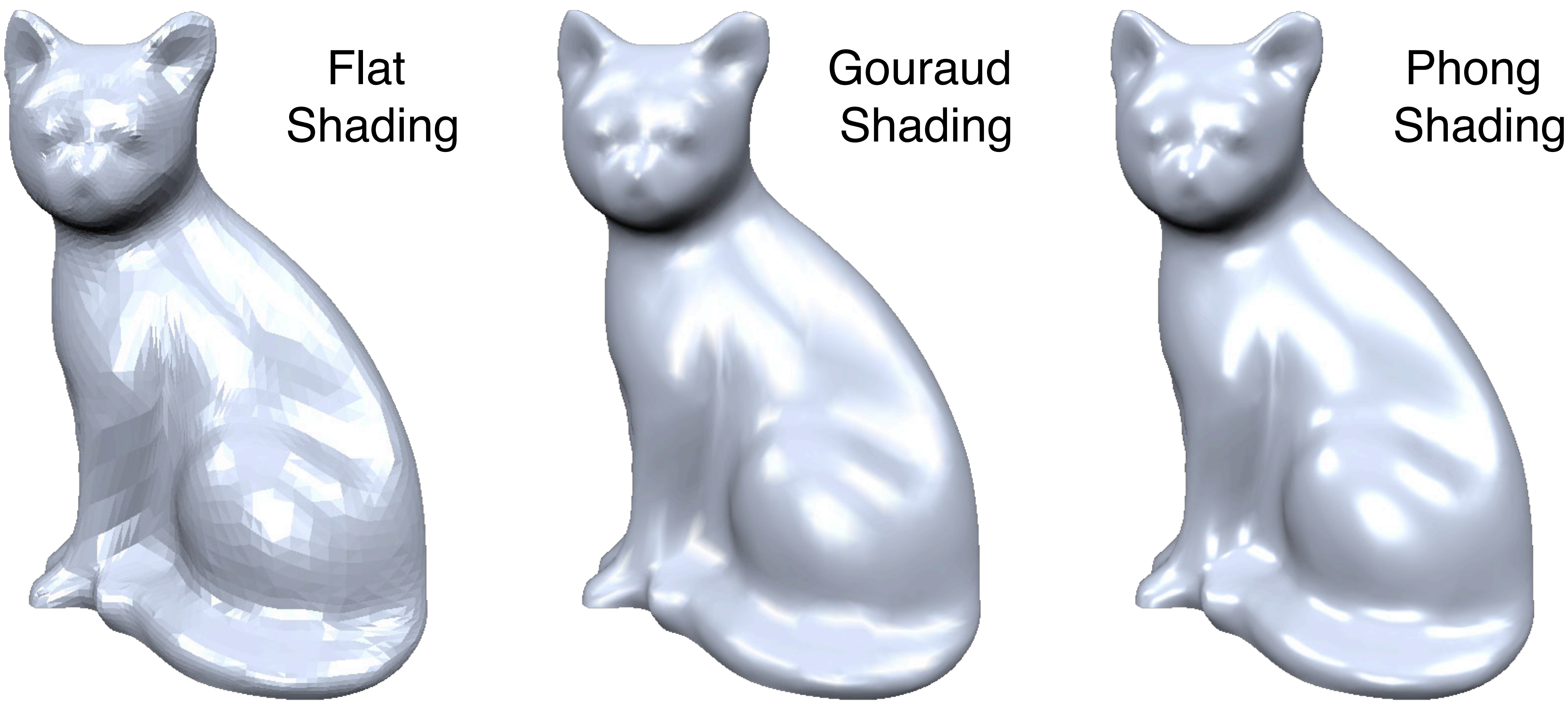
strain energy

$$\int_{\mathcal{S}} \left(\frac{\partial \kappa_1}{\partial \mathbf{t}_1}\right)^2 + \left(\frac{\partial \kappa_2}{\partial \mathbf{t}_2}\right)^2 dA$$

variation of curvature

Mesh Quality

- Visual inspection of “sensitive” attributes
  - Specular shading

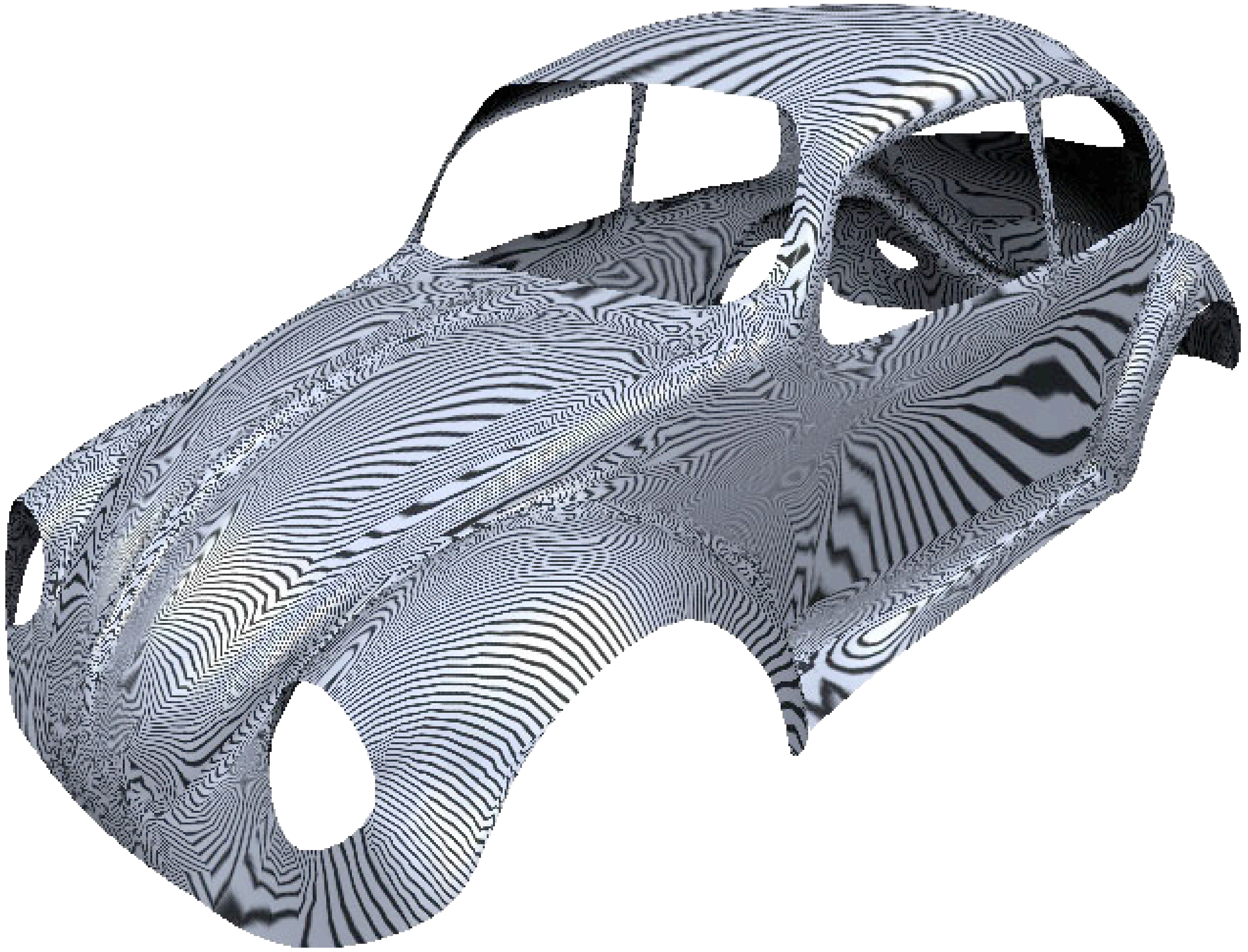


Mesh Quality

- Visual inspection of “sensitive” attributes
  - Specular shading

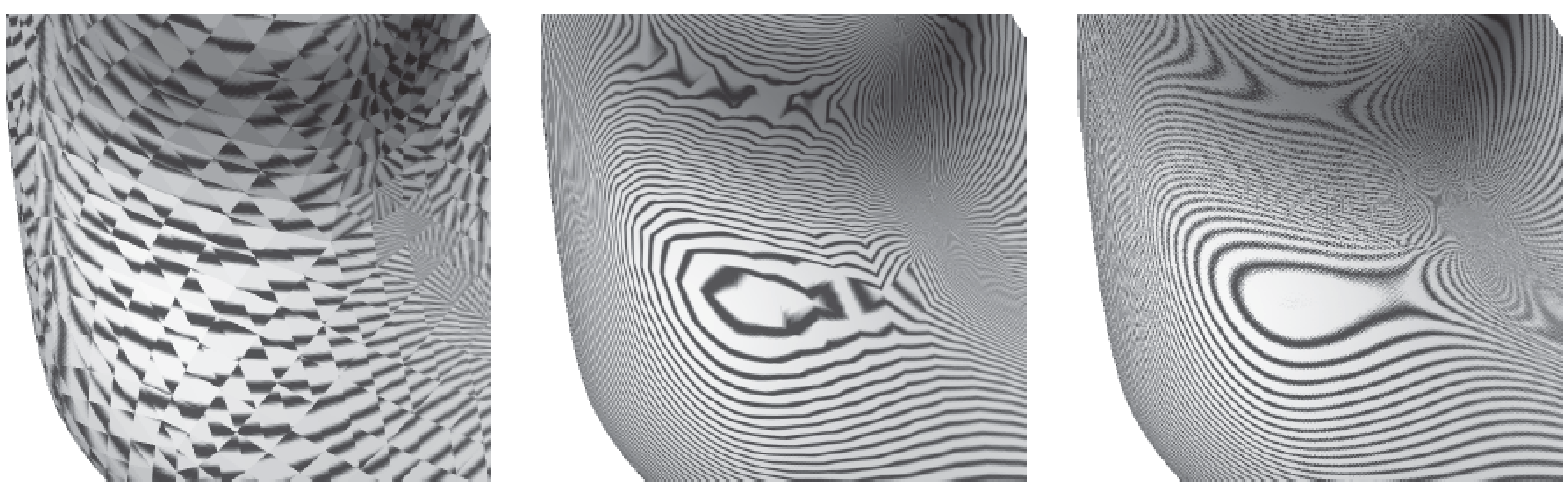
Mesh Quality

- Visual inspection of “sensitive” attributes
  - Specular shading
  - Reflection lines



Mesh Quality

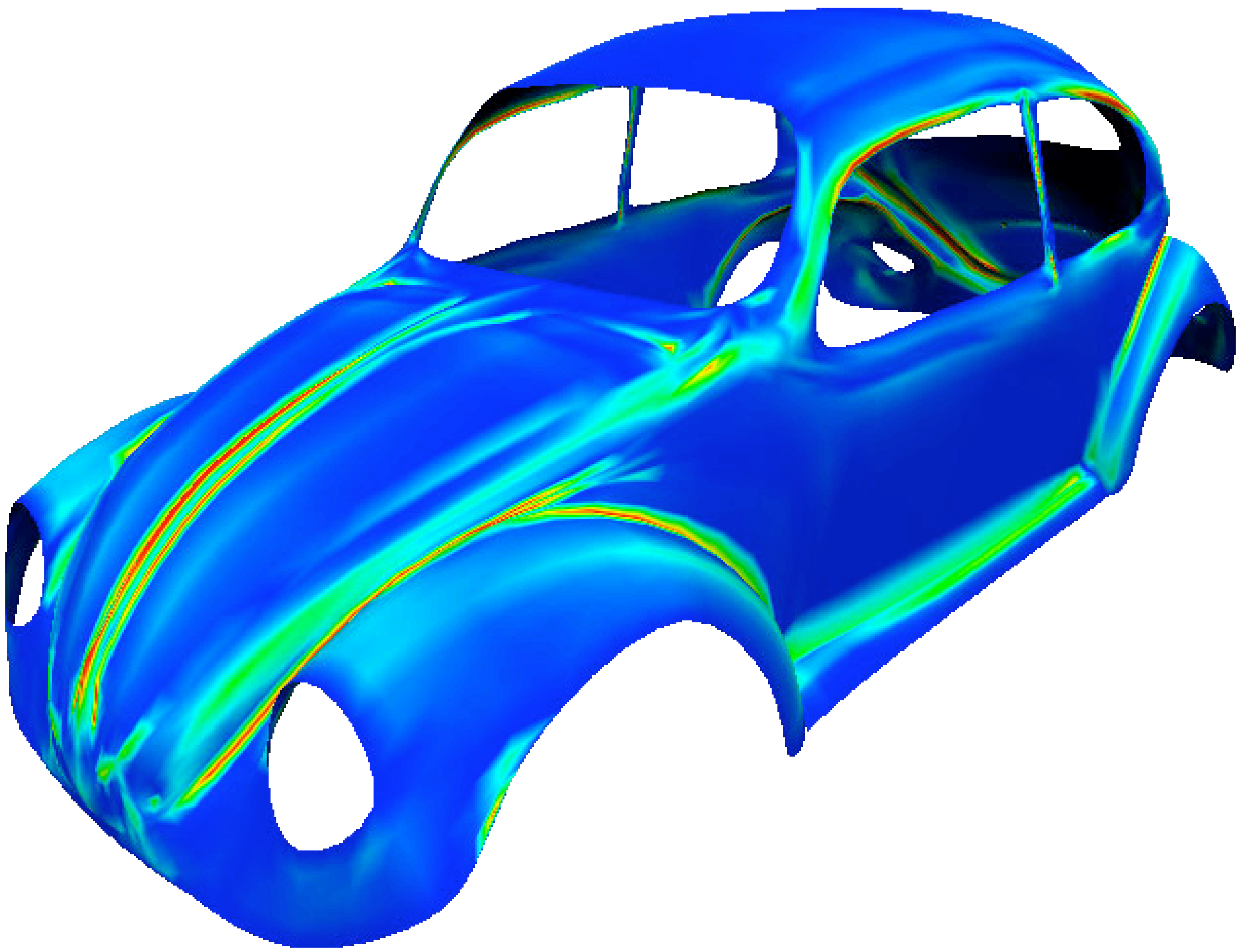
- Visual inspection of “sensitive” attributes
  - Specular shading
  - Reflection lines
    - differentiability one order lower than surface
    - can be efficiently computed using graphics hardware



C^0 C^1 C^2

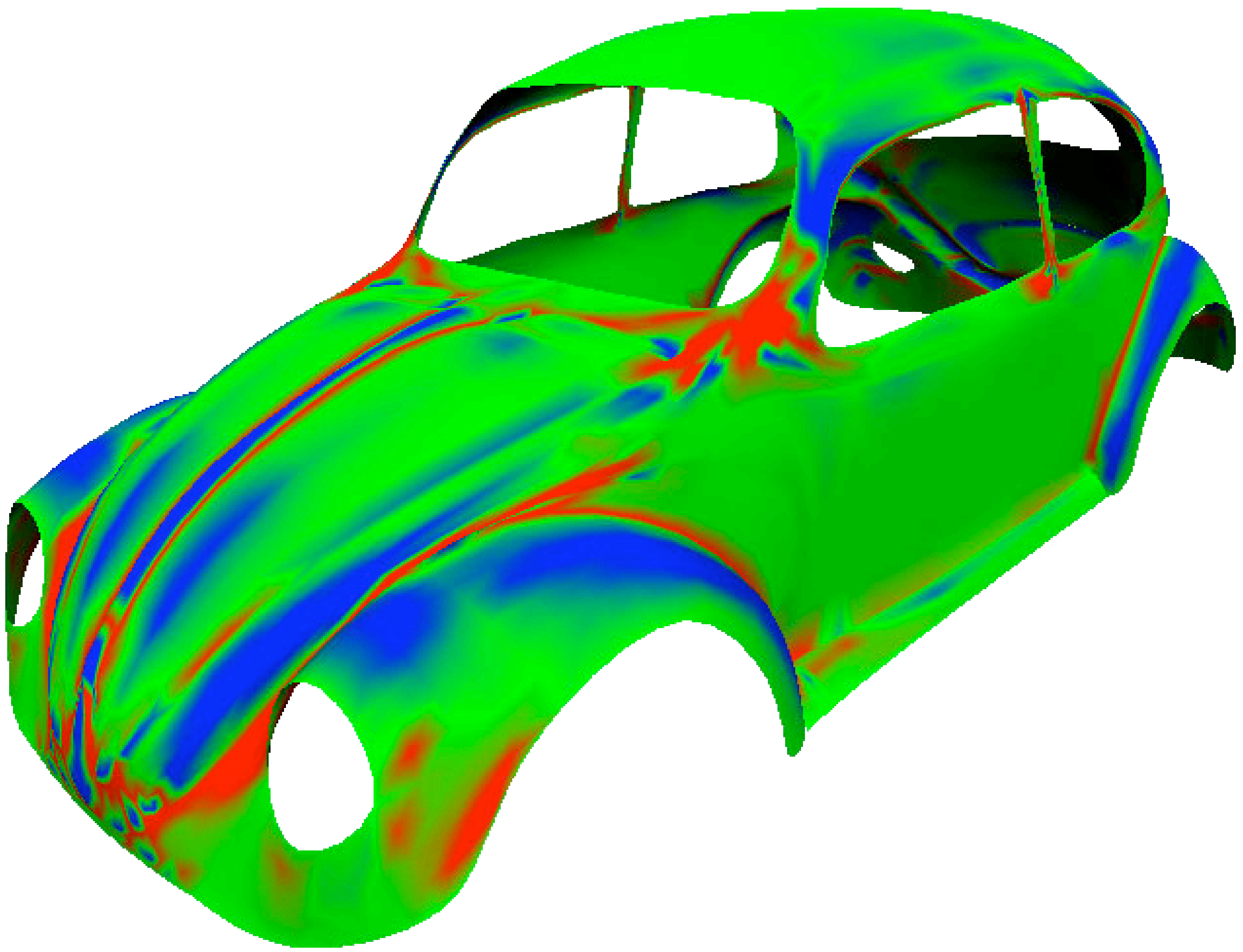
Mesh Quality

- Visual inspection of “sensitive” attributes
  - Specular shading
  - Reflection lines
  - Curvature
    - Mean curvature



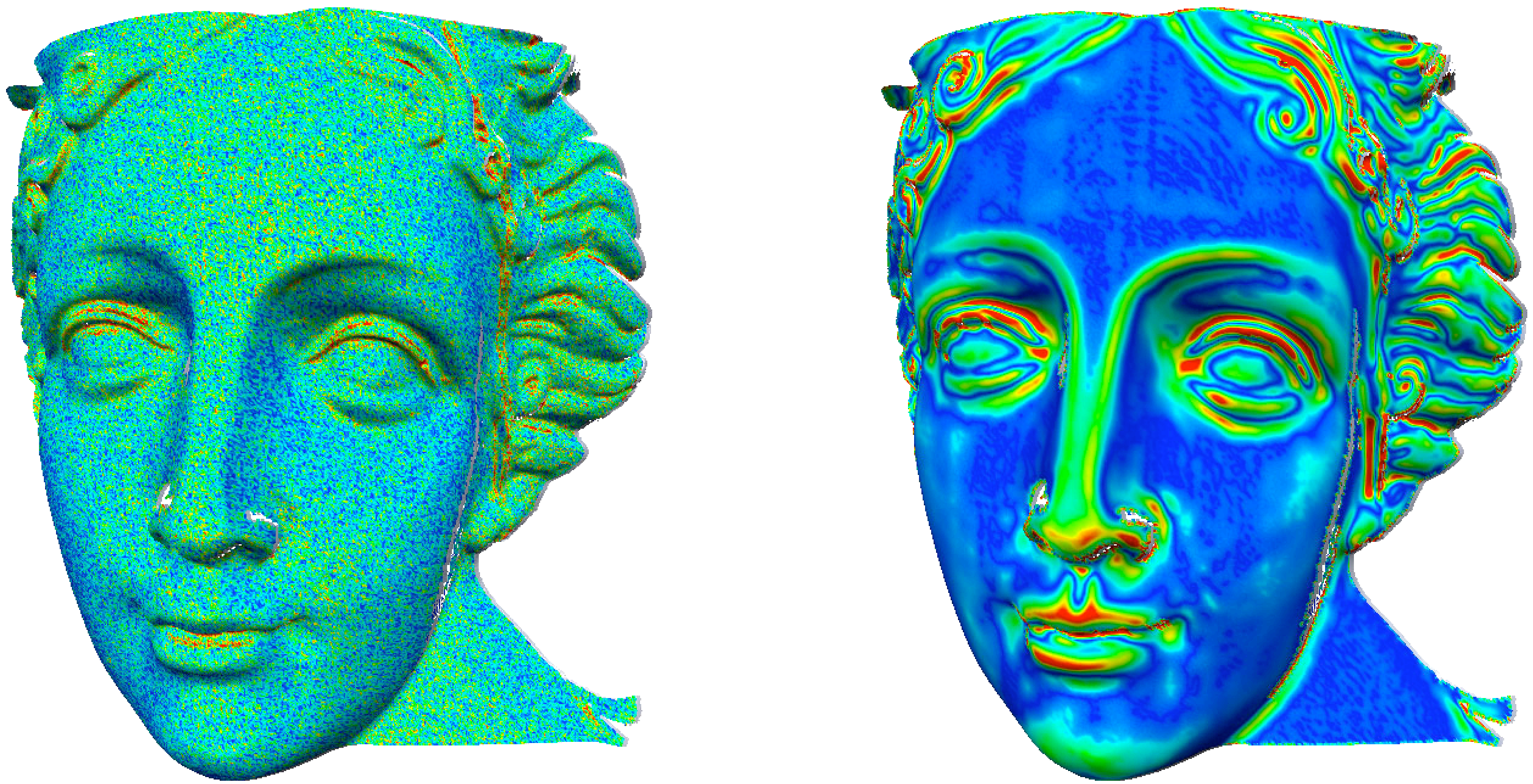
Mesh Quality

- Visual inspection of “sensitive” attributes
  - Specular shading
  - Reflection lines
  - Curvature
    - Mean curvature
    - Gauss curvature



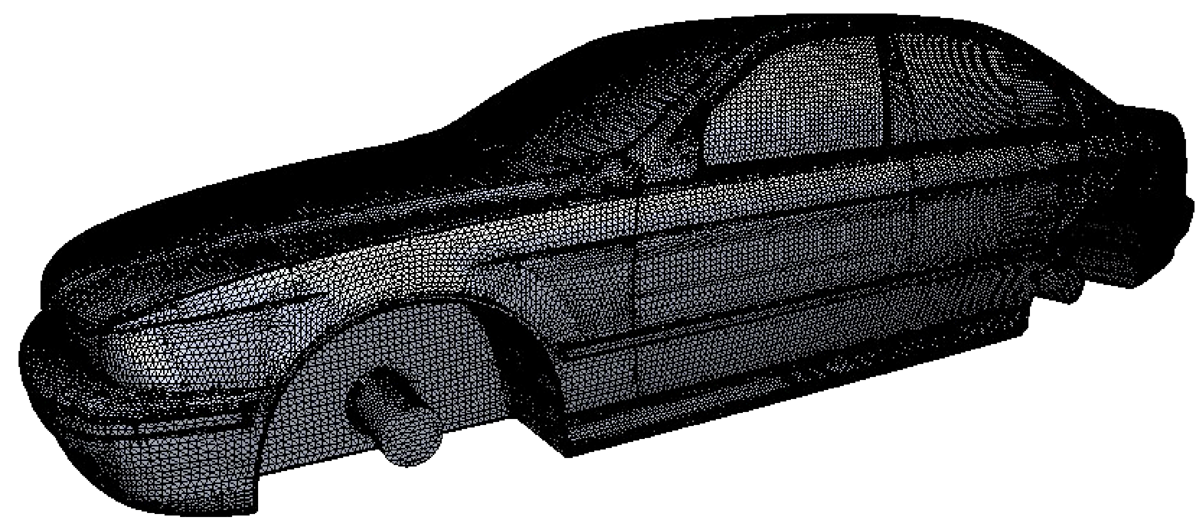
Mesh Quality Criteria

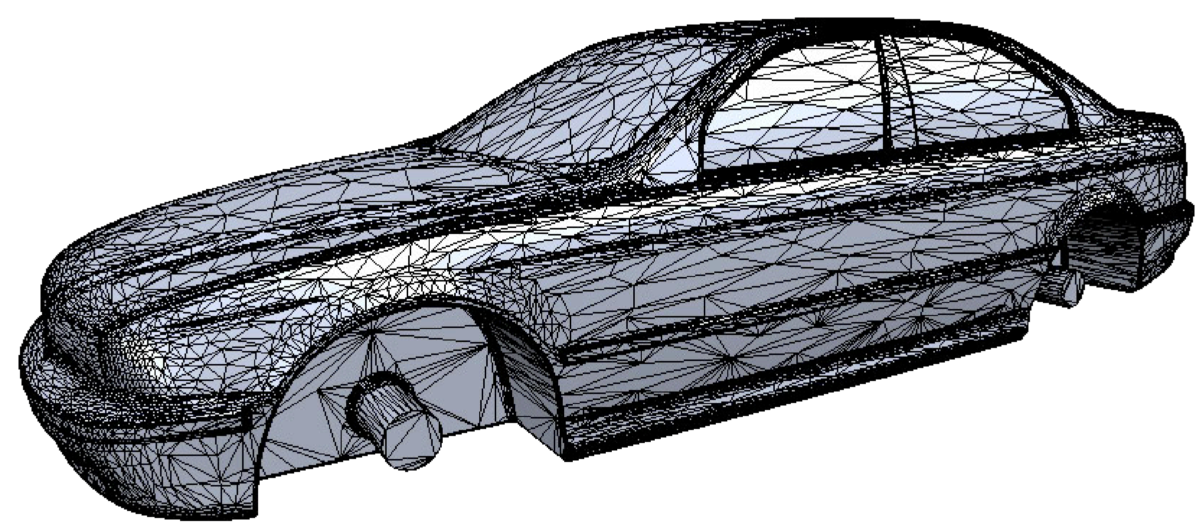
- Smoothness
  - Low geometric noise



Mesh Quality Criteria

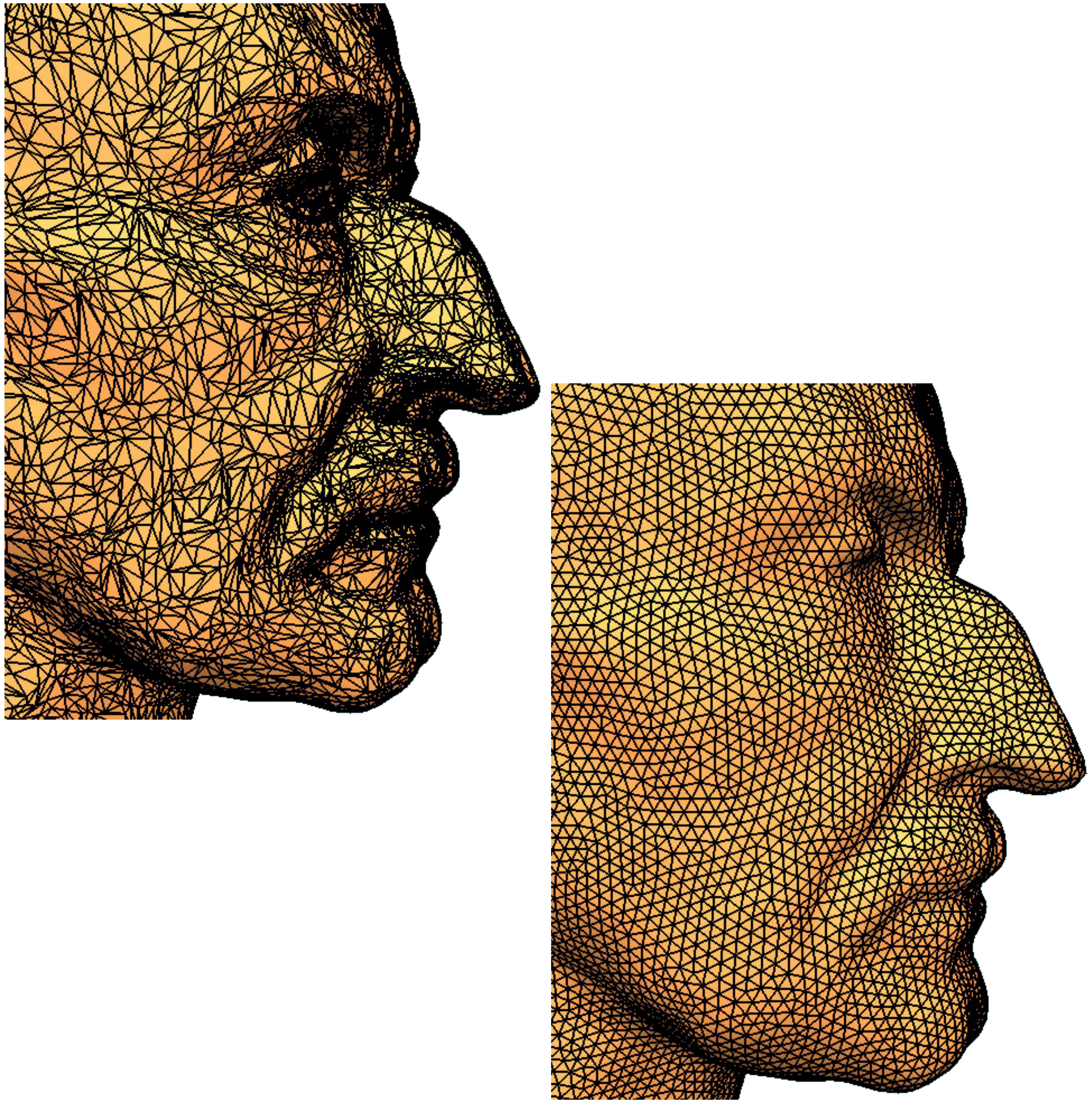
- Smoothness
  - Low geometric noise
- Adaptive tessellation
  - Low complexity





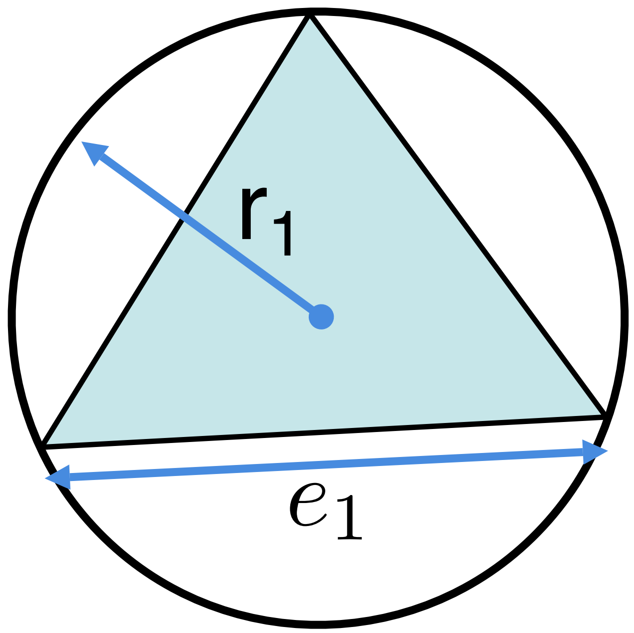
Mesh Quality Criteria

- Smoothness
  - Low geometric noise
- Adaptive tessellation
  - Low complexity
- Triangle shape
  - Numerical robustness

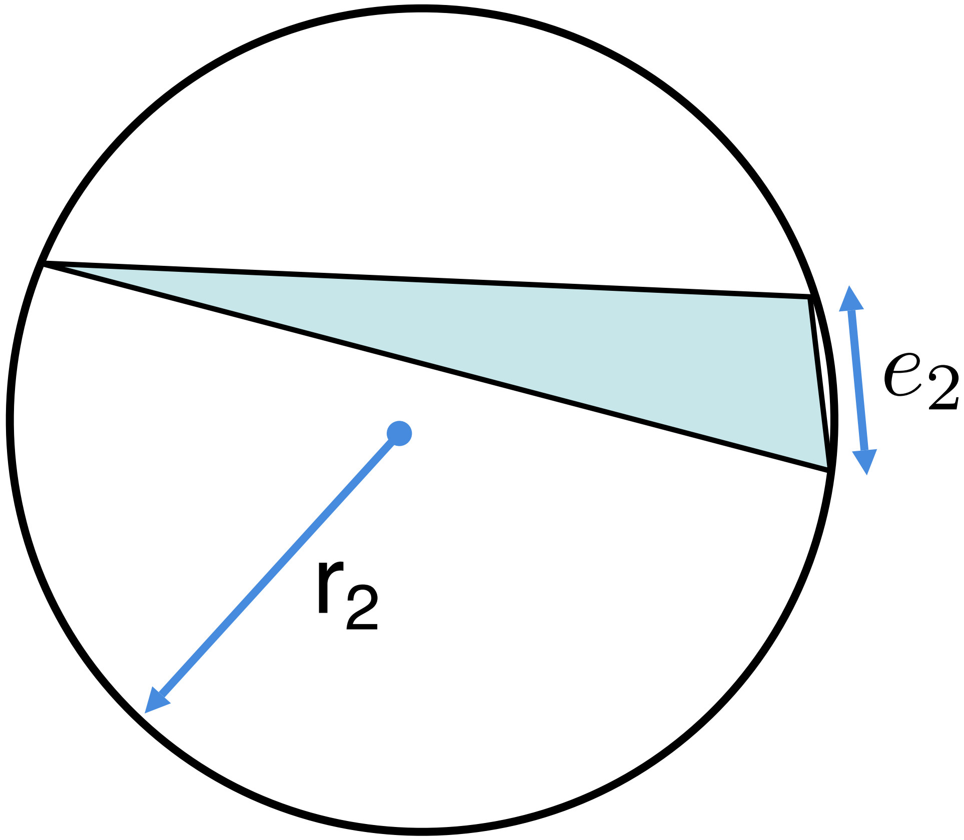


Triangle Shape Analysis

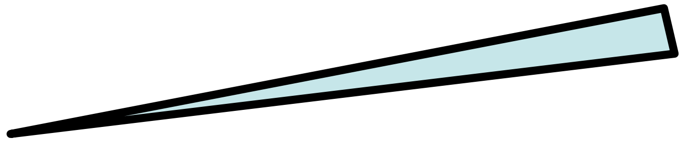
- Circum radius / shortest edge



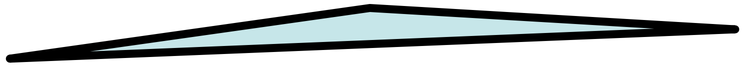
$$\frac{r_1}{e_1} < \frac{r_2}{e_2}$$



- Needles and caps



Needle

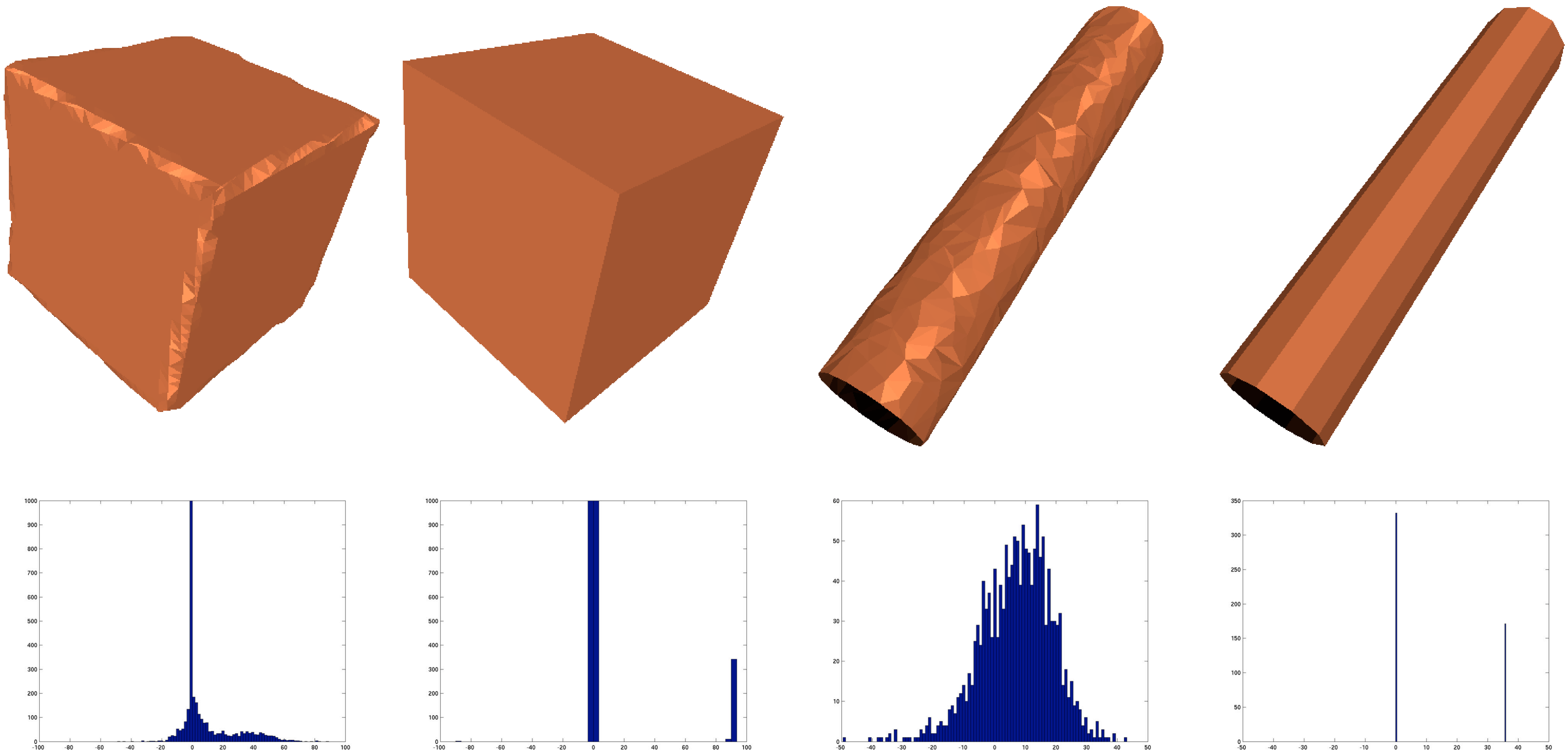


Cap

Mesh Quality Criteria

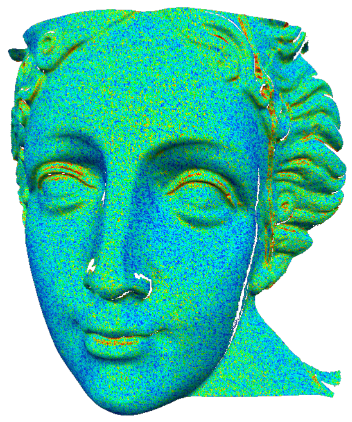
- Smoothness
  - Low geometric noise
- Adaptive tessellation
  - Low complexity
- Triangle shape
  - Numerical robustness
- Feature preservation
  - Low normal noise

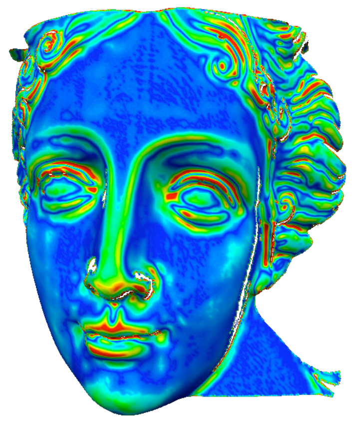
Normal Noise Analysis



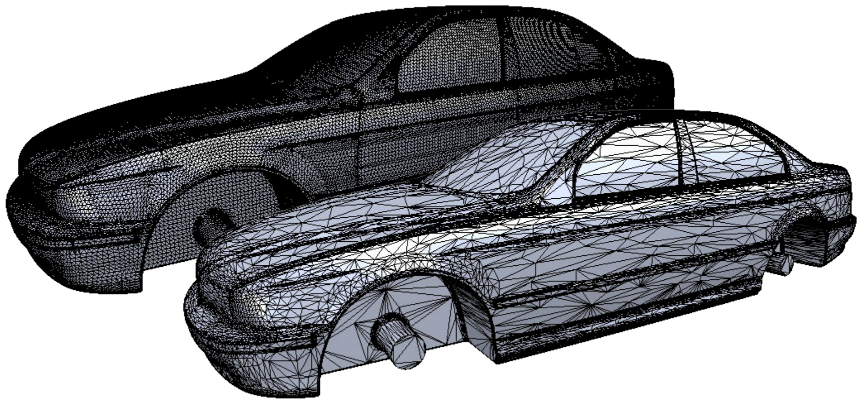
Mesh Optimization

- Smoothness
  - ➡ Mesh smoothing

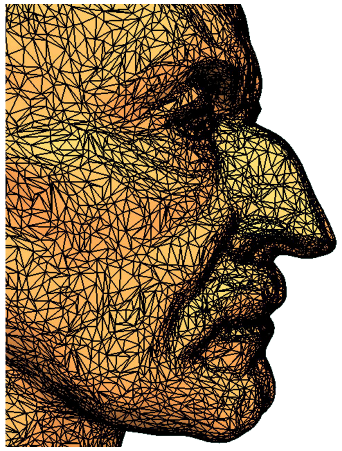


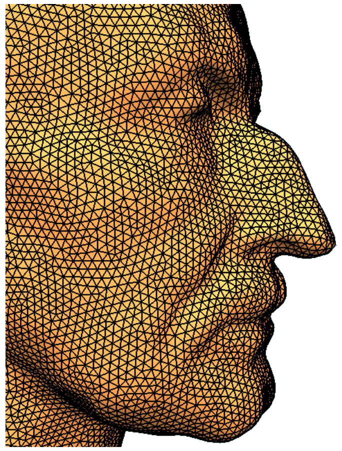


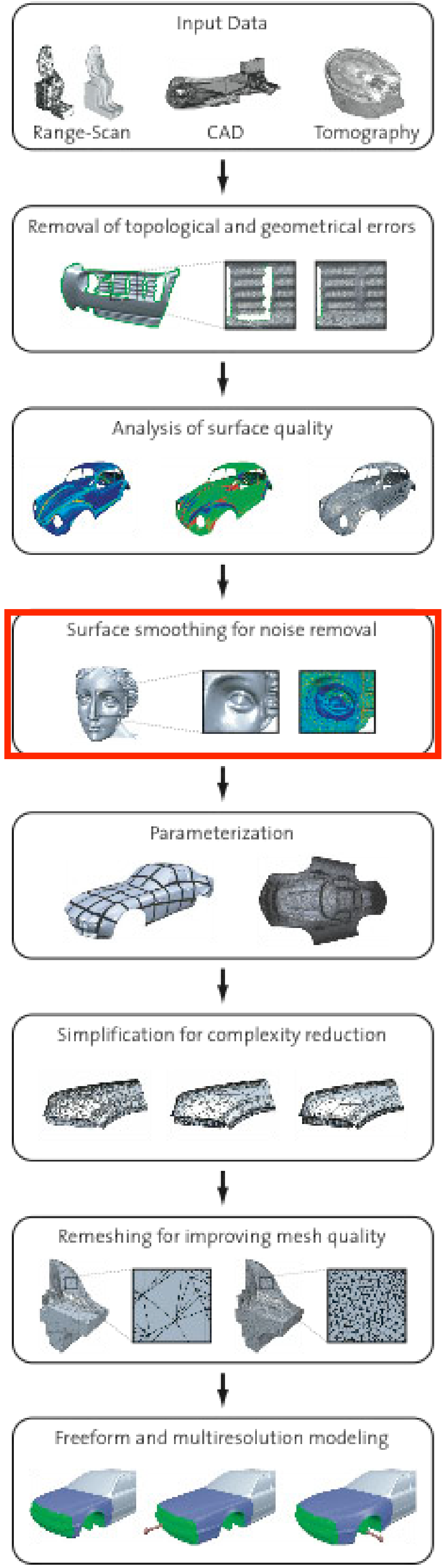
- Adaptive tessellation
  - ➡ Mesh decimation



- Triangle shape
  - ➡ Repair, remeshing







Surface Smoothing

Christian Rössl

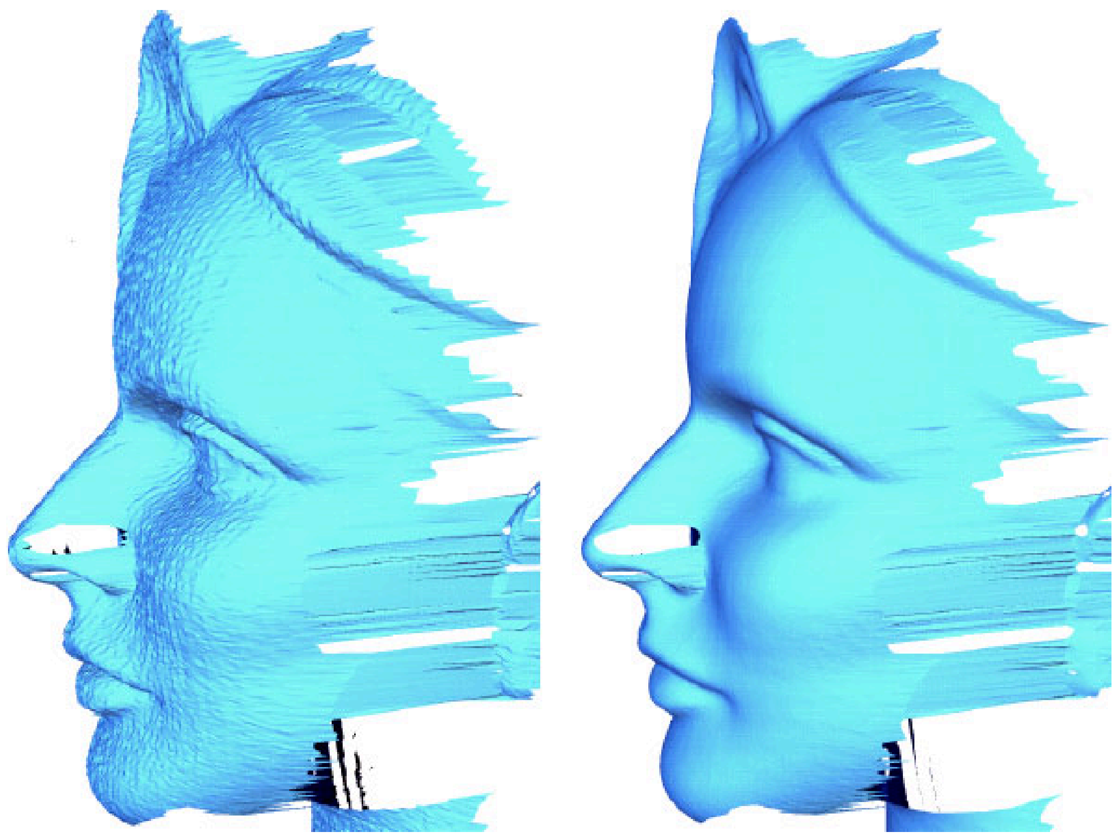
INRIA Sophia-Antipolis

Outline

- Motivation
- Smoothing as *Diffusion*
- Smoothing as *Energy Minimization*
- Alternative Approaches

Motivation

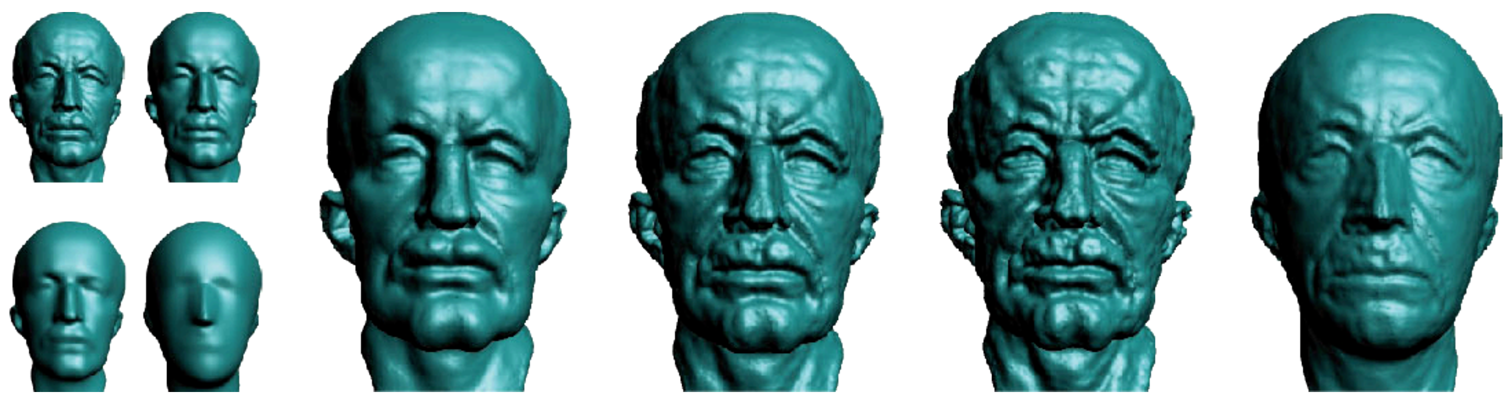
- Filter out high frequency components for noise removal



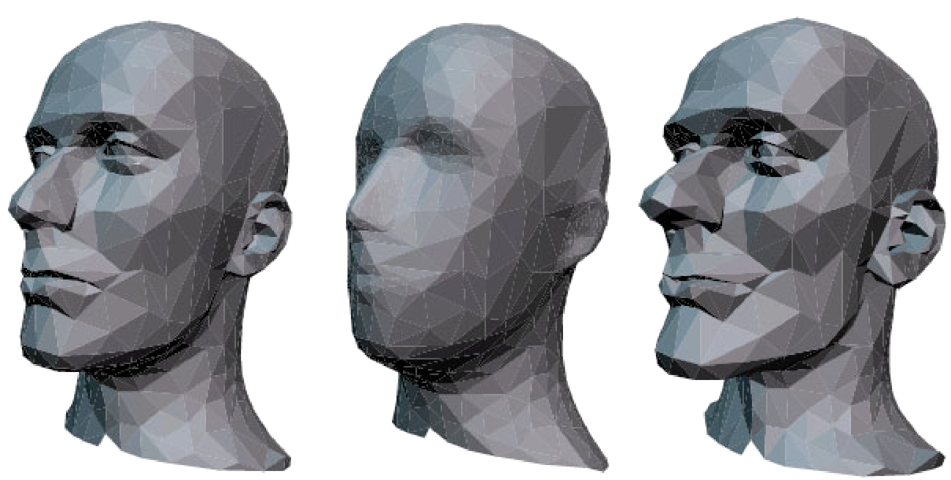
Desbrun, Meyer, Schroeder, Barr: *Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow*, SIGGRAPH 99

Motivation

- Advanced Filtering / Signal Processing



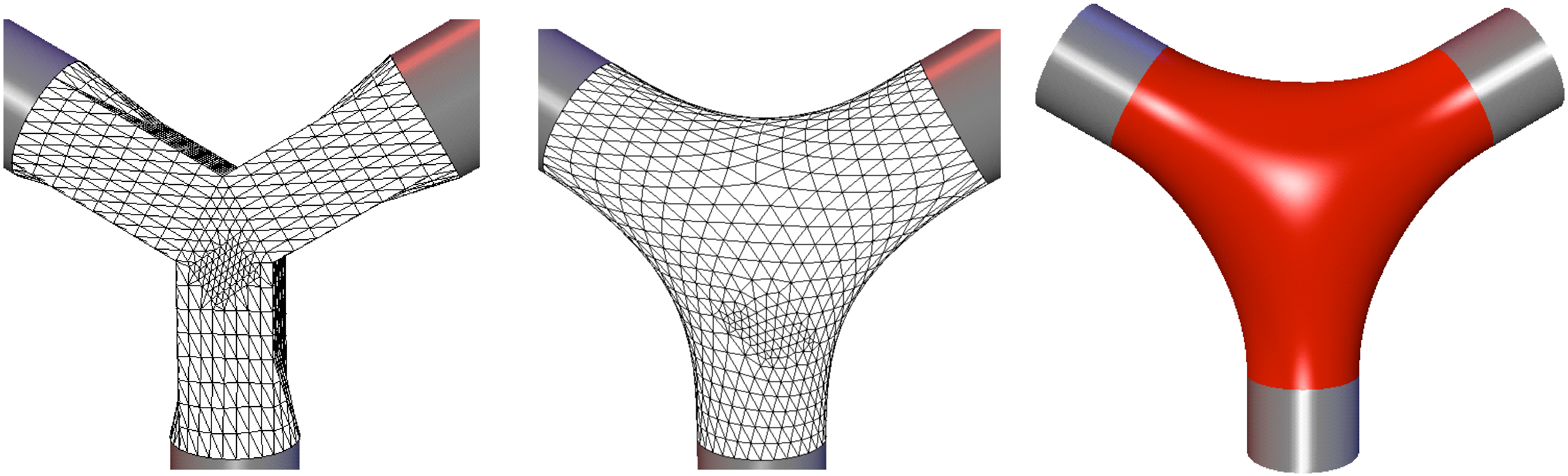
Pauly, Kobbelt, Gross: *Point-Based Multi-Scale Surface Representation*, ACM TOG 2006



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

Motivation

- Fair Surface Design

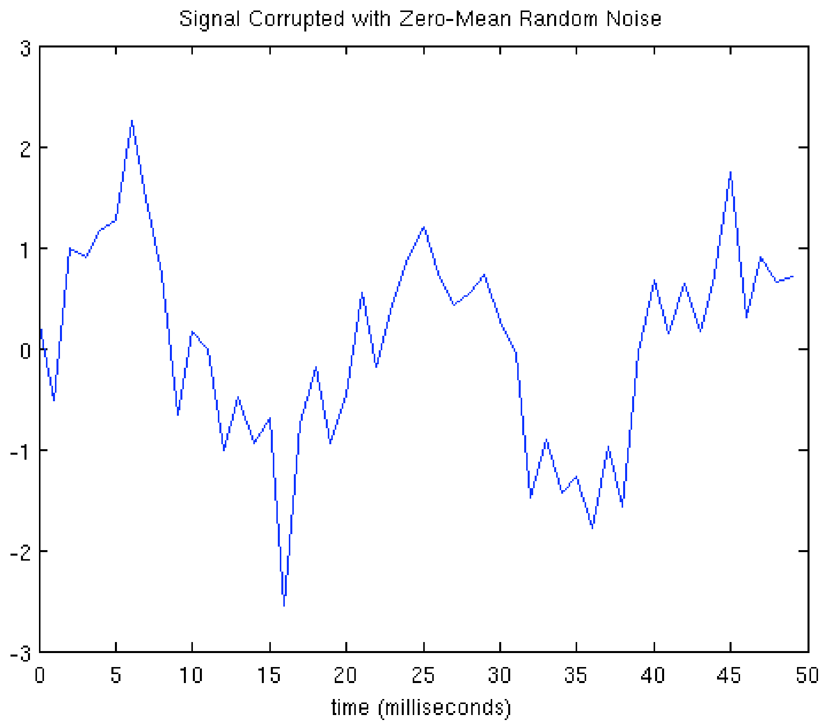


Outline

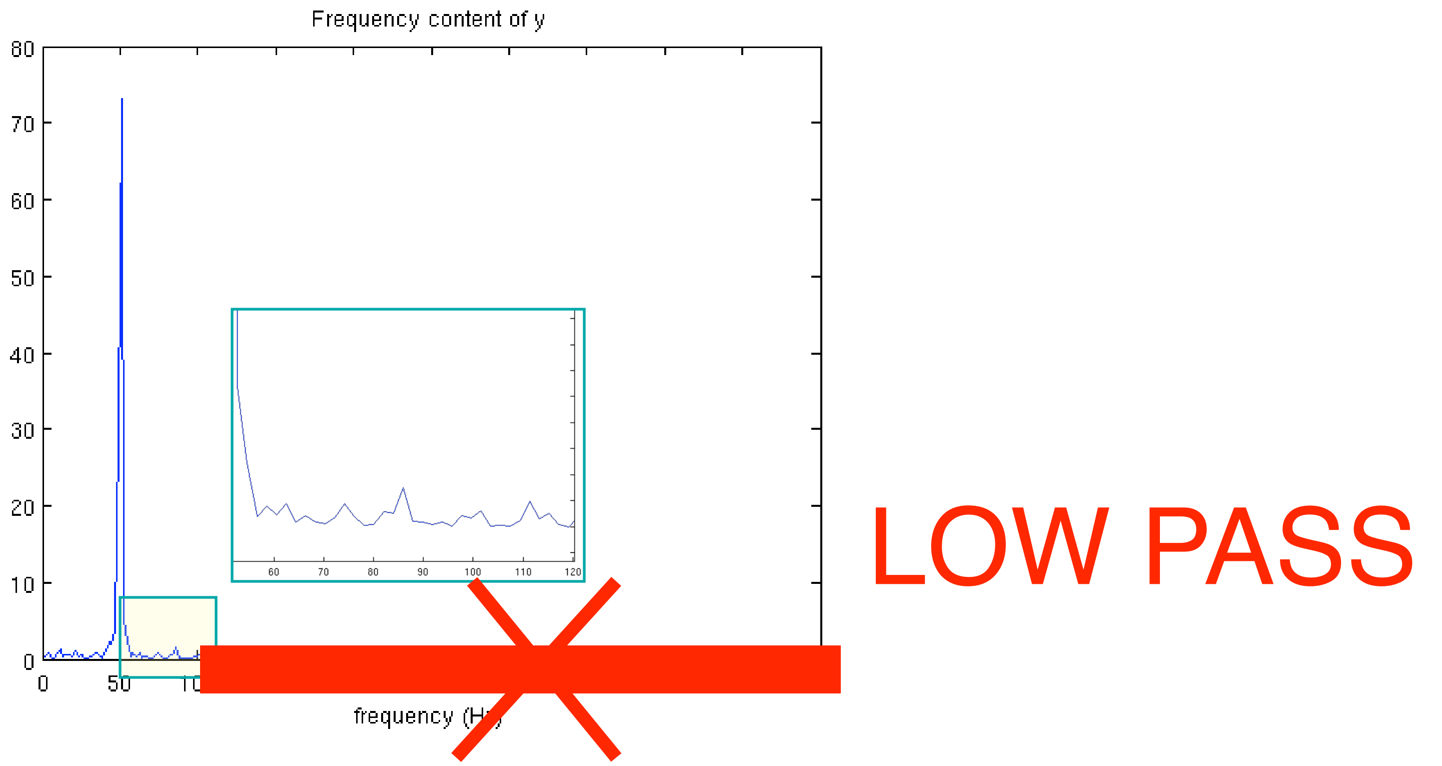
- Motivation
- Smoothing as *Diffusion*
  - Spectral Analysis
  - Laplacian Smoothing
  - Curvature Flow
  - Implementation
- Smoothing as *Energy Minimization*
- Alternative Approaches

Filter Design

- Assume *high frequency* components = *NOISE*
- Low-pass filter



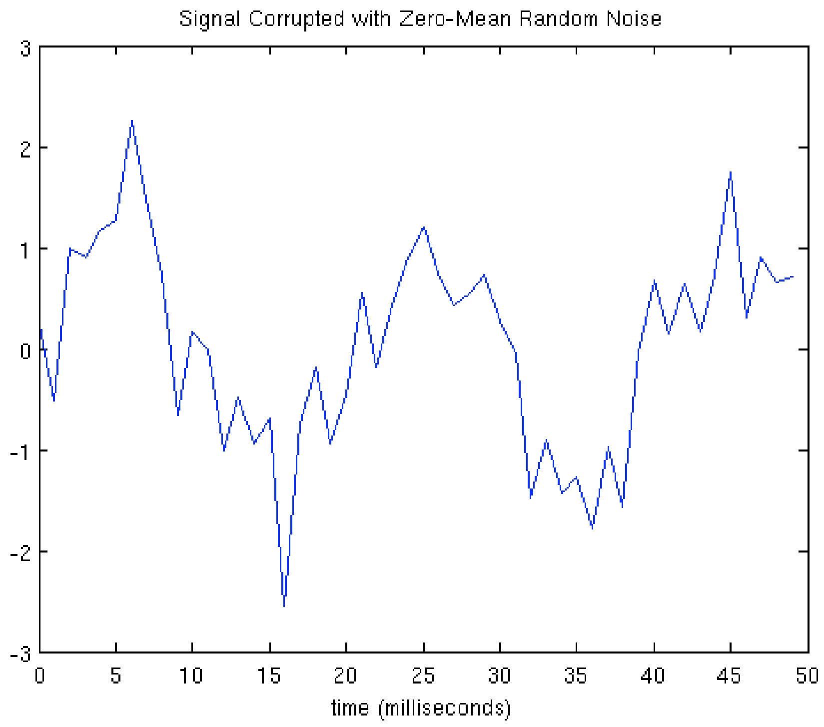
spatial domain

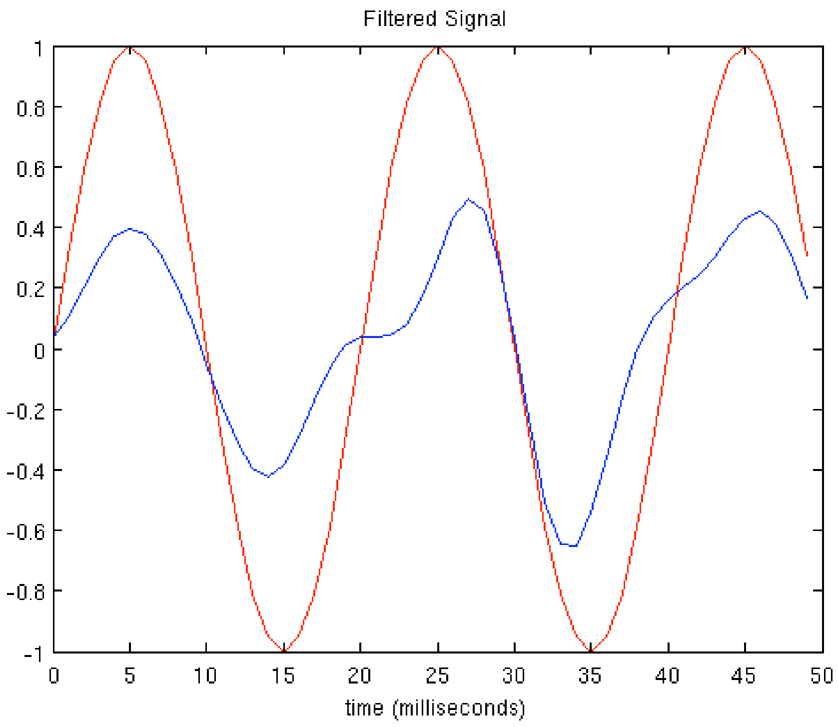


frequency domain

Filter Design

- Assume *high frequency* components = *NOISE*
- Low-pass filter





reconstruction = filtered signal

Filter Design

- Assume *high frequency* components = *NOISE*
- Low-pass filter
  - Damps high frequencies (ideal: cut off)
  - e.g., by convolution with Gaussian *(spatial domain)*
    = multiply with Gaussian *(frequency domain)*
- Fourier Transform

Spectral Analysis and Filter Design

- Univariate: *Fourier Analysis*

$$F(\varphi) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(t) e^{-i\varphi t} dt$$

frequency domain

spatial domain

- Example: *Low-pass filter*
  - Damp (ideally cut off high frequencies)
  - Multiply F with Gaussian (= convolve f with Gaussian)

- Are there "*geometric frequencies*"?

Spectral Analysis and Filter Design

- Univariate: *Fourier Analysis*

$$F(\varphi) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(t) e^{-i\varphi t} dt$$

- Generalization

$$\Delta e^{i\varphi t} = \frac{\partial^2}{\partial t^2} e^{i\varphi t} = -\varphi^2 e^{i\varphi t}$$

  - $e^{i\varphi t}$ are *Eigenfunctions* of the Laplacian
  - *Use them as basis functions for geometry*

Spectral Analysis

- Eigenvalues of Laplacian ≅ *frequencies*
- Low-pass filter ≅
  reconstruction from eigenvectors associated
  with *low* frequencies
- Decomposition in frequency bands is used for
  mesh deformation.
- Too expensive for *direct* use in practice!
  Cannot compute eigenvalues efficiently
- For smoothing apply *diffusion*…
  (similar to convolution vs. multiplication)

Outline

- Motivation
- Smoothing as *Diffusion*
  - Spectral Analysis
  - Laplacian Smoothing
  - Curvature Flow
  - Implementation
- Smoothing as *Energy Minimization*
- Alternative Approaches

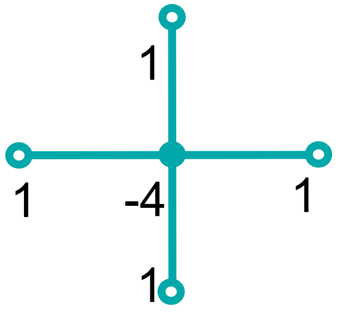
Diffusion

- Diffusion equation

$$\frac{\partial}{\partial t} x = \operatorname{div} \mu \, \nabla x$$

constant scalar

$$\frac{\partial}{\partial t} x = \mu \, \Delta x$$



Laplacian Smoothing

- Discretization of diffusion equation

$$\frac{\partial}{\partial t} p_i = \mu \, \Delta p_i$$

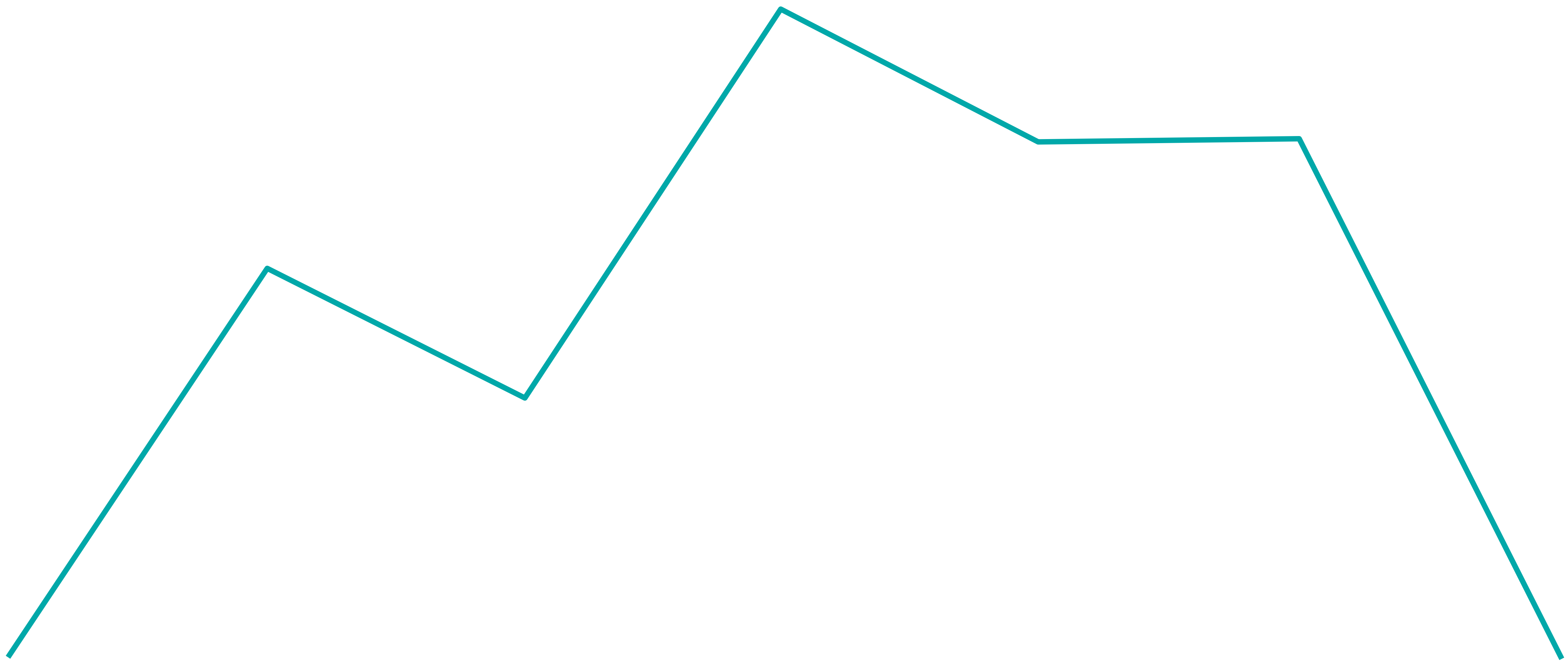
- Leads to simple update rule
  - `Iterate`

$$p_i \leftarrow p_i + \mu \, dt \, \Delta p_i$$

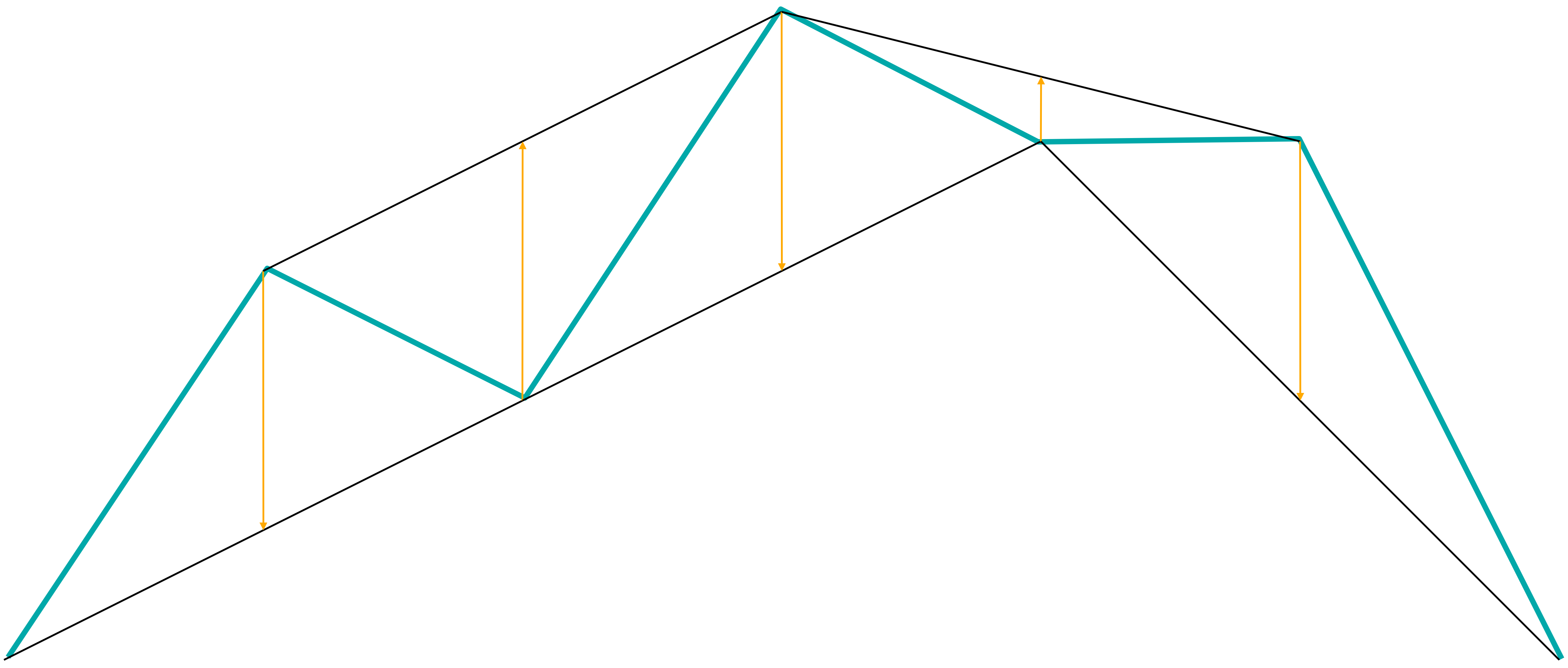
explicit Euler integration

  - `until convergence`

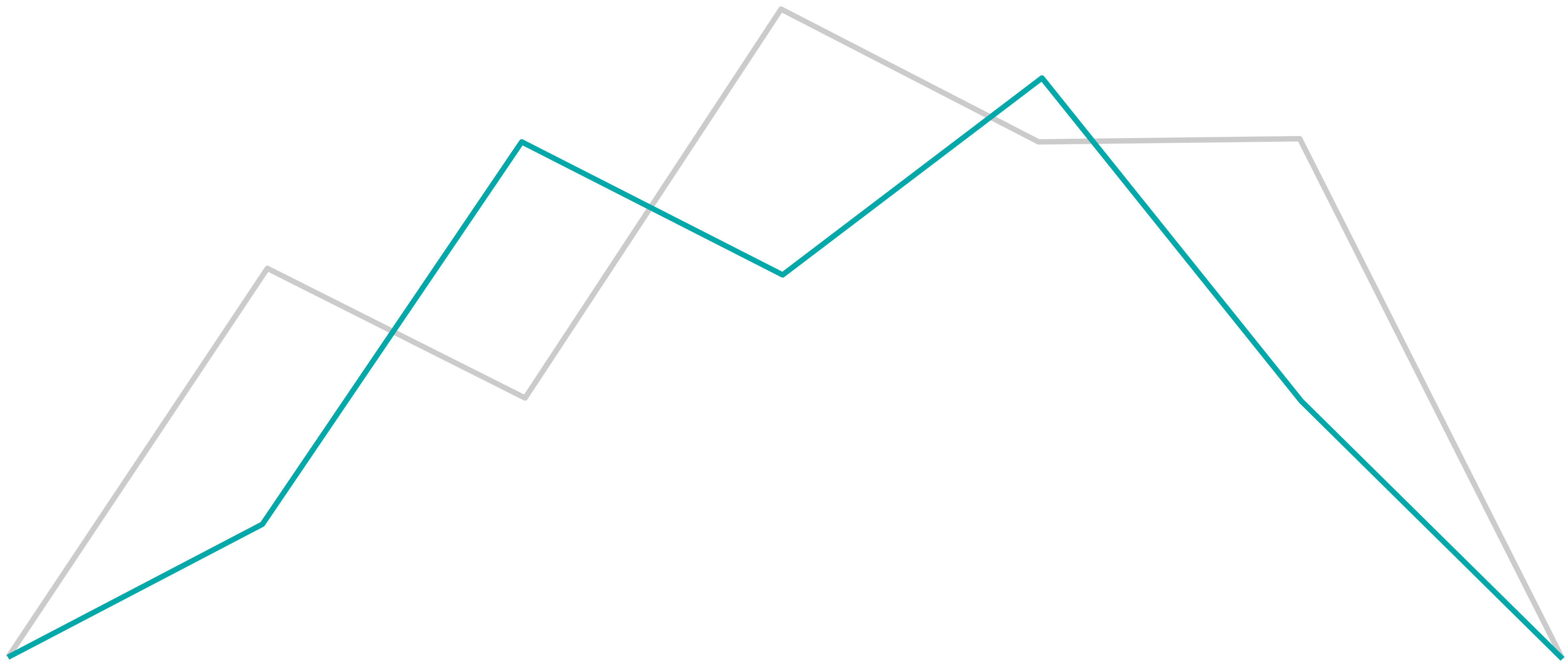
A Simple Example



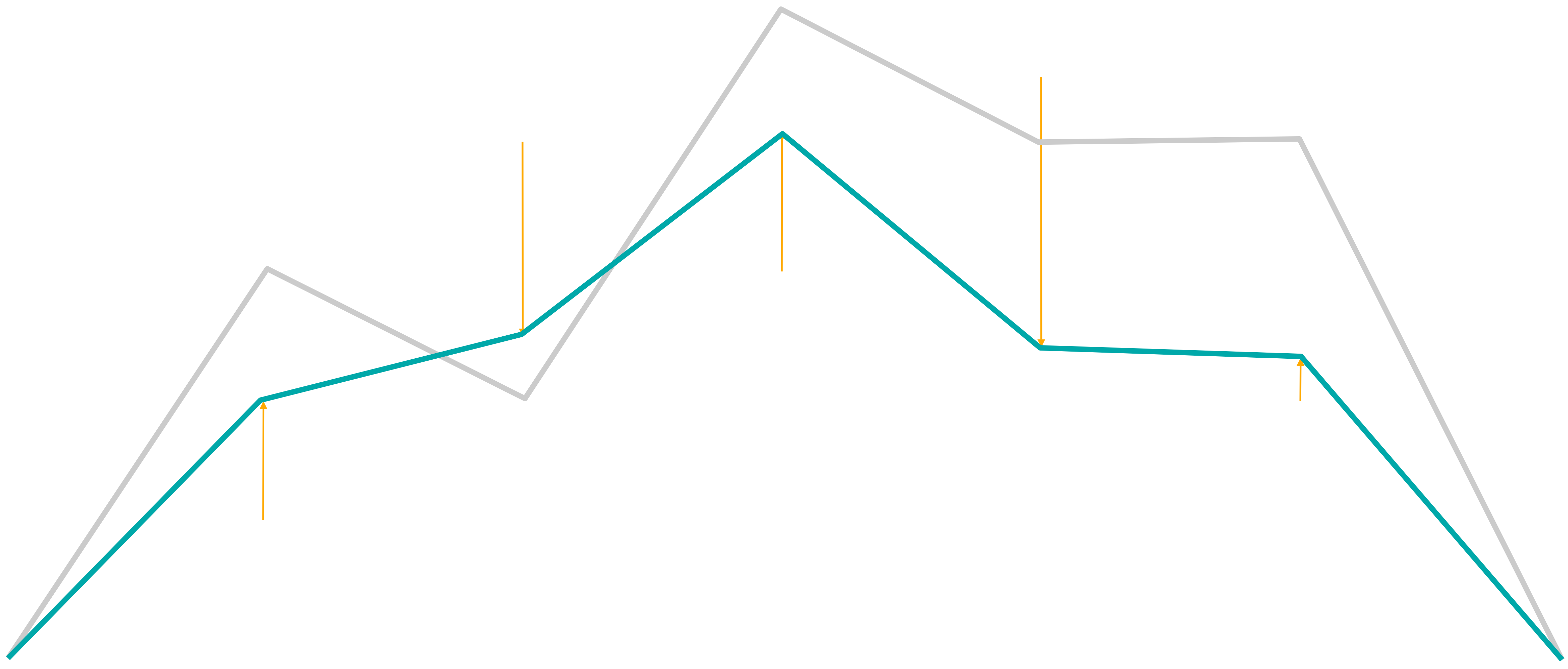
A Simple Example



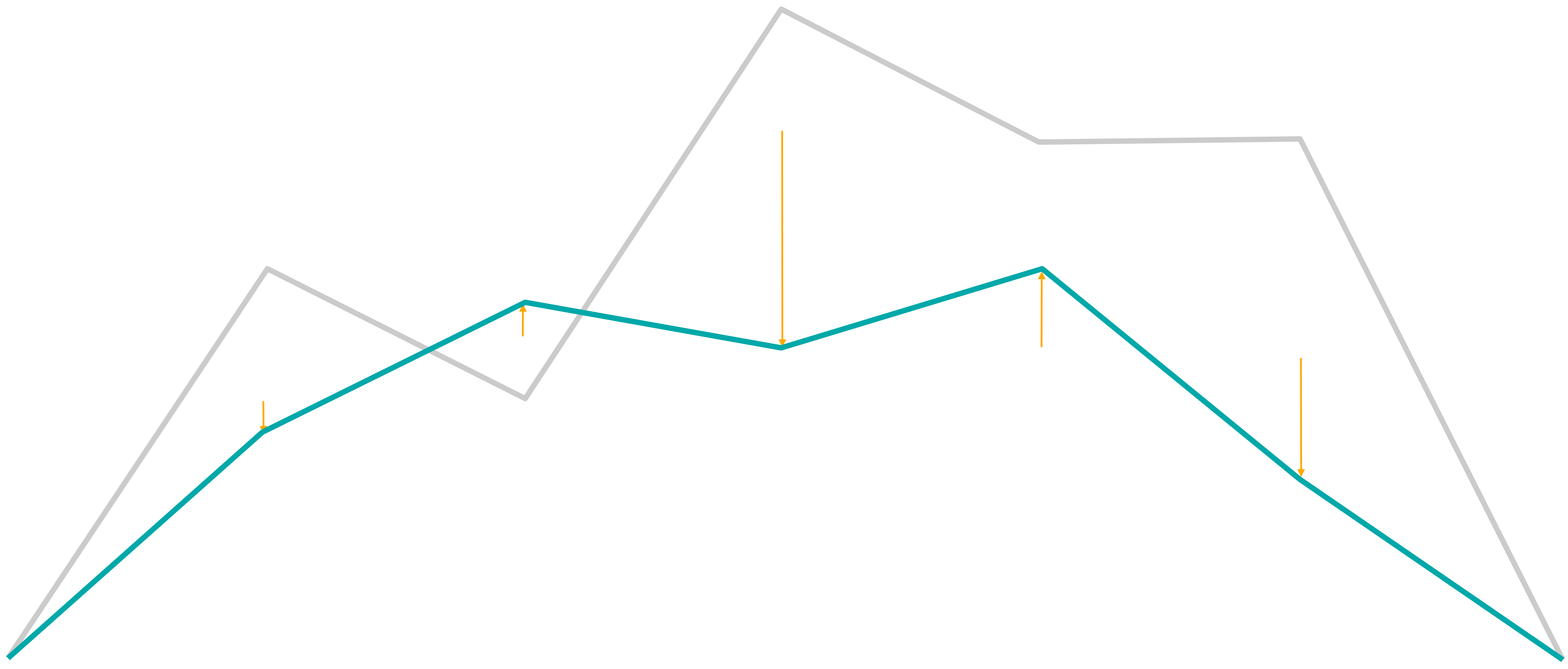
A Simple Example



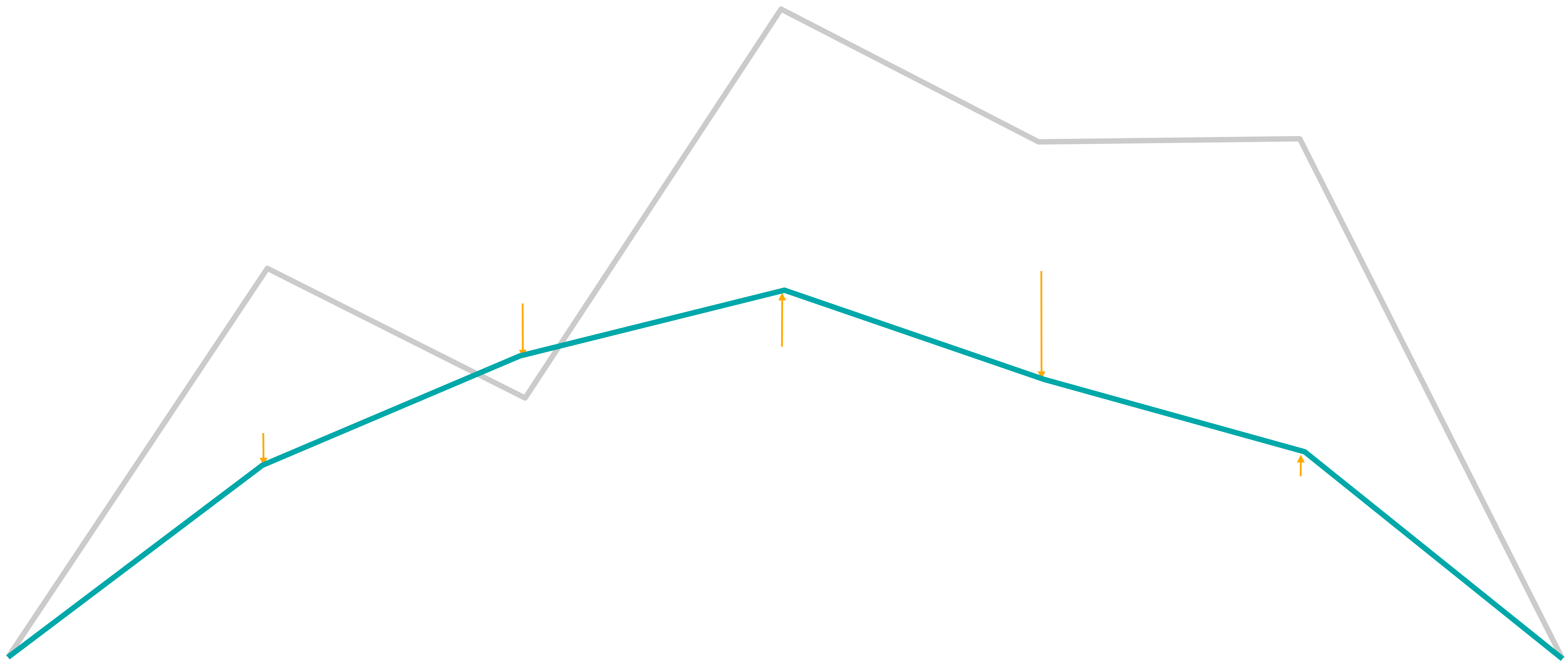
A Simple Example



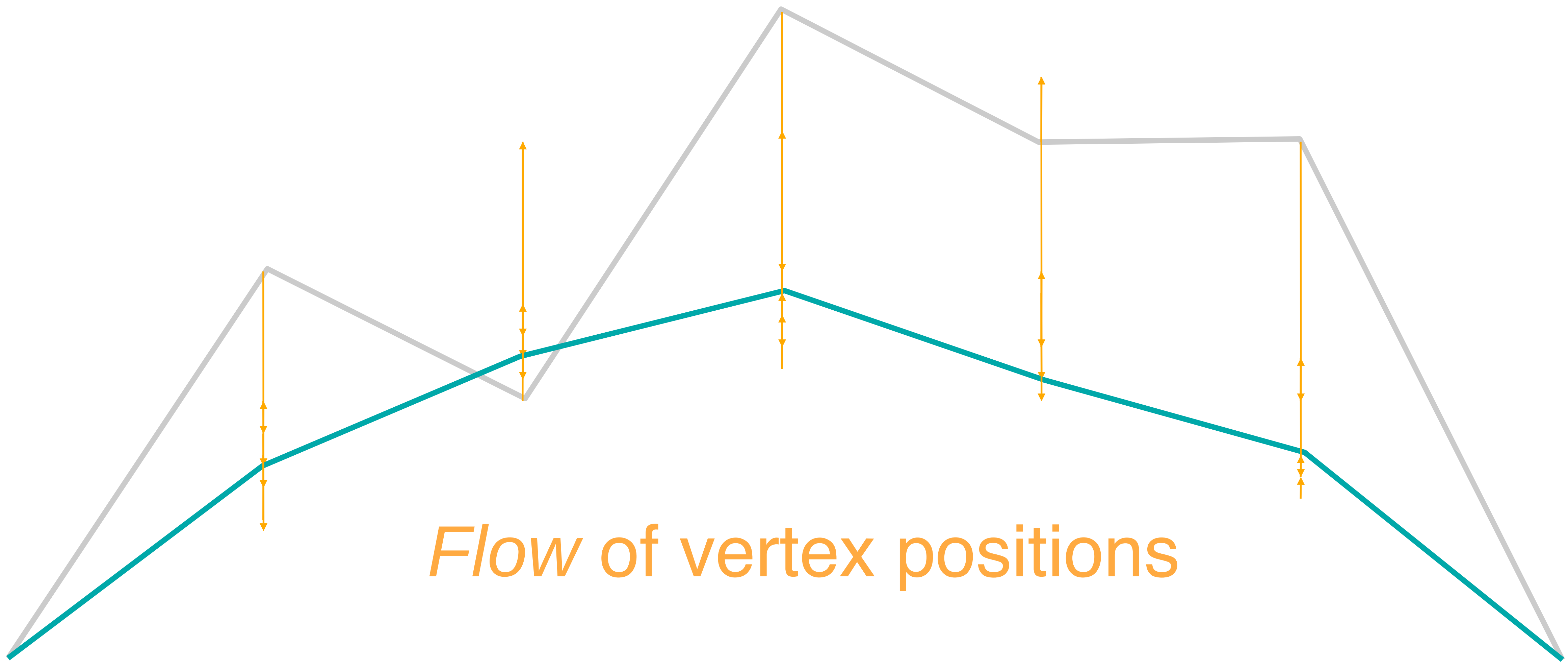
A Simple Example



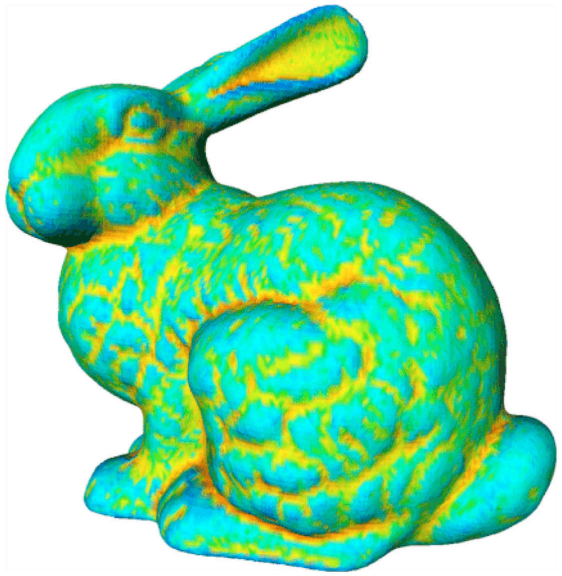
A Simple Example



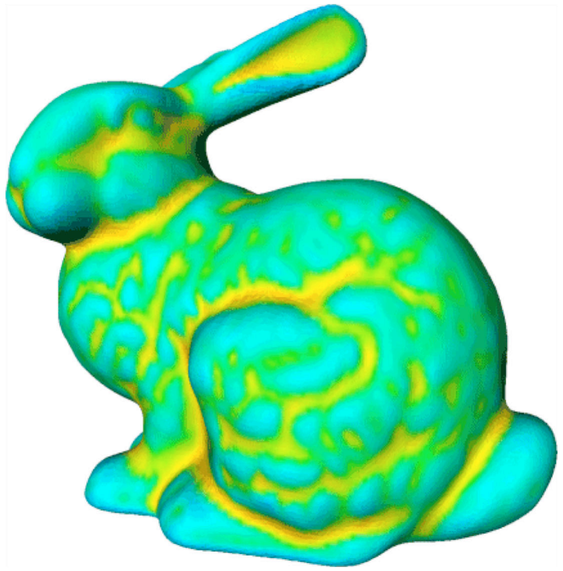
A Simple Example



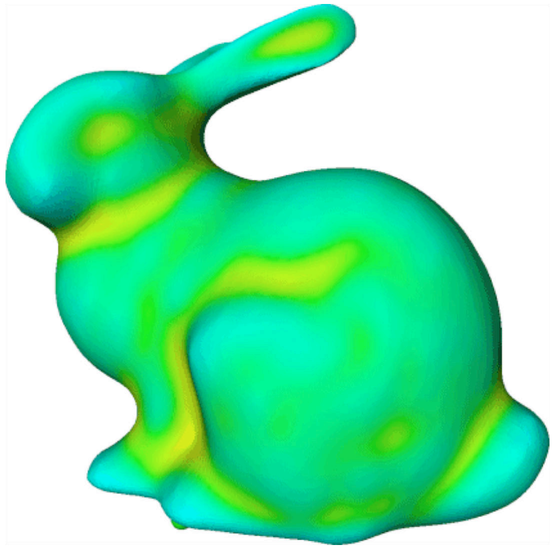
Laplacian Smoothing



0 Iterations



5 Iterations



20 Iterations

Outline

- Motivation
- Smoothing as *Diffusion*
  - Spectral Analysis
  - Laplacian Smoothing
  - Curvature Flow
  - Implementation
- Smoothing as *Energy Minimization*
- Alternative Approaches
  - Anisotropic Smoothing

Curvature Flow

- Curvature is independent of parameterization
- Flow equation

surface normal n

$$\frac{\partial}{\partial t} x = -\mu\, H\, n$$

mean curvature *H*

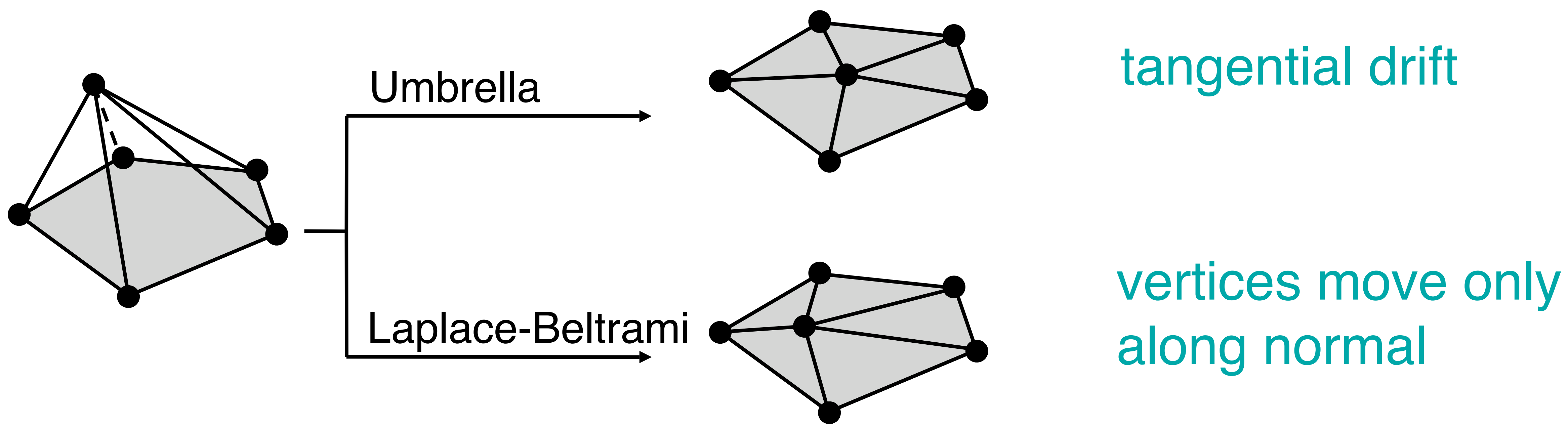
- We have

$$\Delta_S x = -2\, H\, n$$

Laplace-Beltrami operator

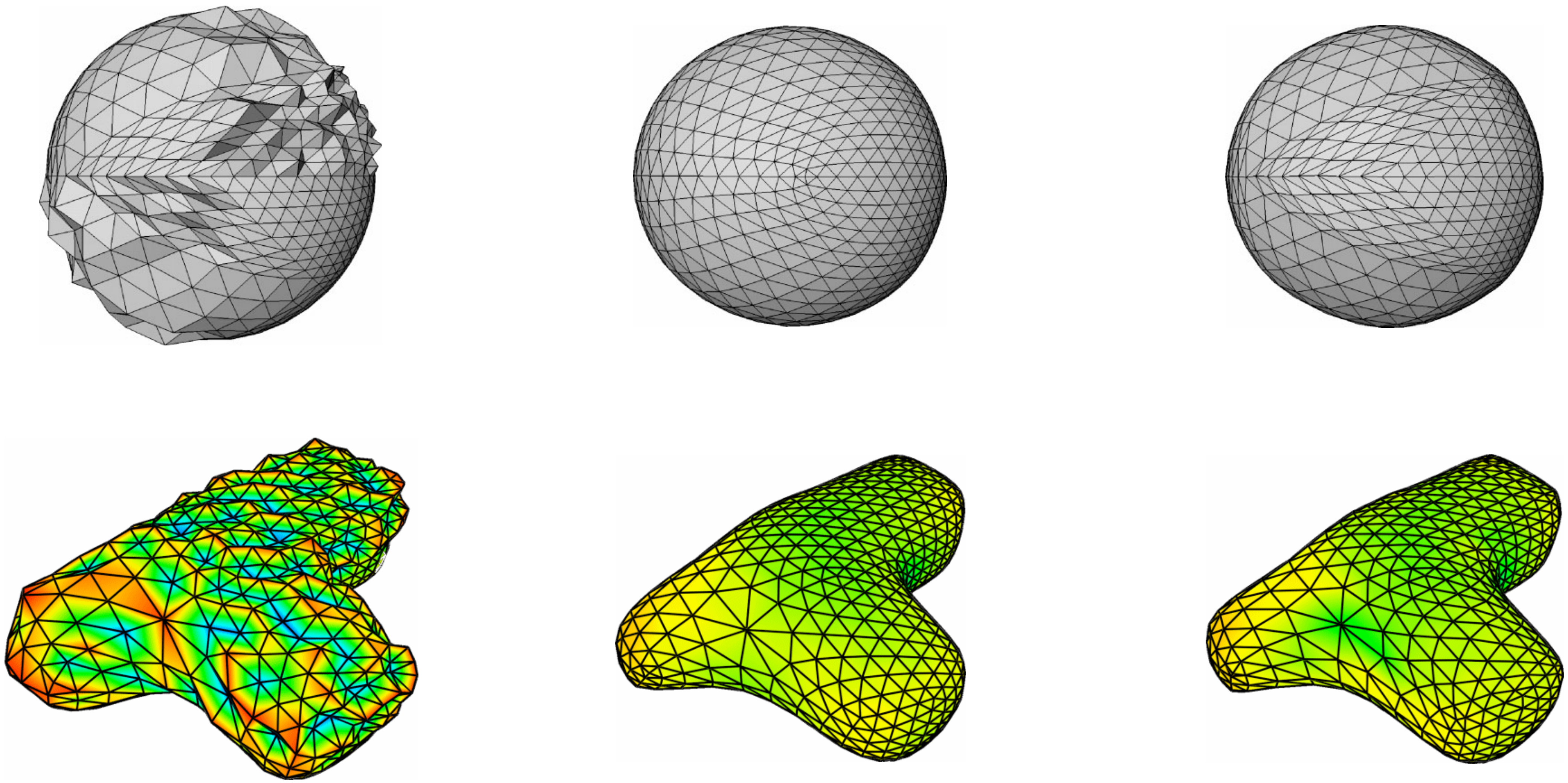
Curvature Flow

- Mean curvature Flow $$\frac{\partial}{\partial t}x = \mu\,\Delta_S x$$
  - Use discrete Laplace-Beltrami operator *(cot weights)*
  - Higher order flows
- Compare to uniform discretization of Laplacian



Comparison

Original Umbrella Laplace-Beltrami



Outline

- Motivation
- Smoothing as *Diffusion*
  - Spectral Analysis
  - Laplacian Smoothing
  - Curvature Flow
  - Implementation
- Smoothing as *Energy Minimization*
- Alternative Approaches

Integration

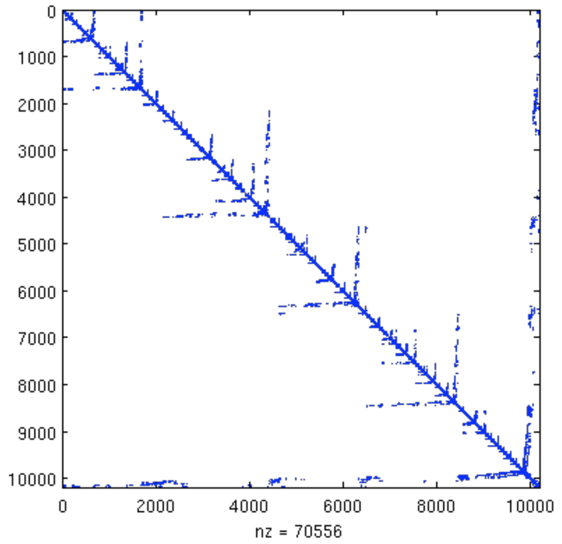
- Find numerical solution of diffusion equation

- *Explicit* integration $p' = (I + \mu\, dt\, L)\, p$

same as before in matrix form

matrix formulation of update rule

$$p' = p + \mu\, dt\, \Delta p$$



Integration

- Find numerical solution of diffusion equation
- *Explicit* integration $p' = (I + \mu\, dt\, L)\, p$

same as before
in matrix form

  - Jacobi / Gauss-Seidel iterations
  - Requires timestep $0 < \mu\, dt < 1$ for stability
- *Implicit* integration $(I - \mu\, dt\, L)\, p' = p$
  - Requires solution of (sparse) linear system
  - Chose $\mu\, dt$ arbitrarily (~ # explicit integration steps)

Outline

- Motivation
- Smoothing as *Diffusion*
- Smoothing as *Energy Minimization*
    - *Membrane energy*
    - *Thin-plate energy*
- Alternative Approaches

Energy Minimization

- Penalize "unaesthetic behavior"
- Measure *fairness*
  - *Principle of the simplest shape*
  - Independent of parameterization (tessellation)
  - Often physical interpretation
- Minimize energy functional
  - Examples: *membrane / thin plate energy*

Energy Minimization

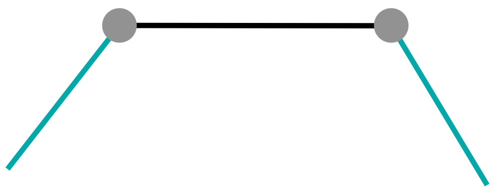
- Membrane Energy

$$f : \Omega \rightarrow \mathbb{R}^3$$

parameterization

$$\int_\Omega f_u^2 + f_v^2 du dv \rightarrow \min$$

+ boundary conditions



  - Euler-Lagrange PDE

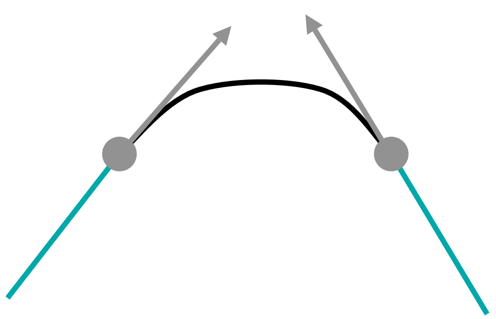
$$\Delta f = f_{uu} + f_{vv} = 0$$

Energy Minimization

- Thin Plate Energy

$$E(S) = \int_S \kappa_1^2 + \kappa_2^2 dS$$

  - No parameter dependence
  - Non-linear functional



  - Find linear approximation…

Energy Minimization

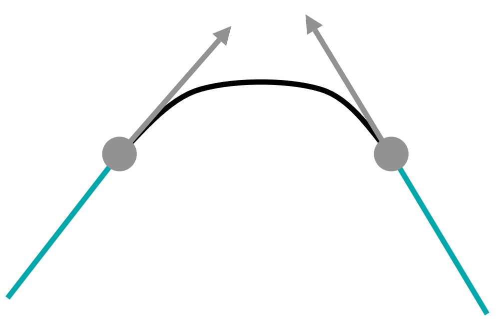
- Thin Plate Energy

$$E(S) = \int_S \kappa_1^2 + \kappa_2^2 dS$$

$$f : \Omega \to \mathbb{R}^3$$

curvatures ~ 2nd order partials

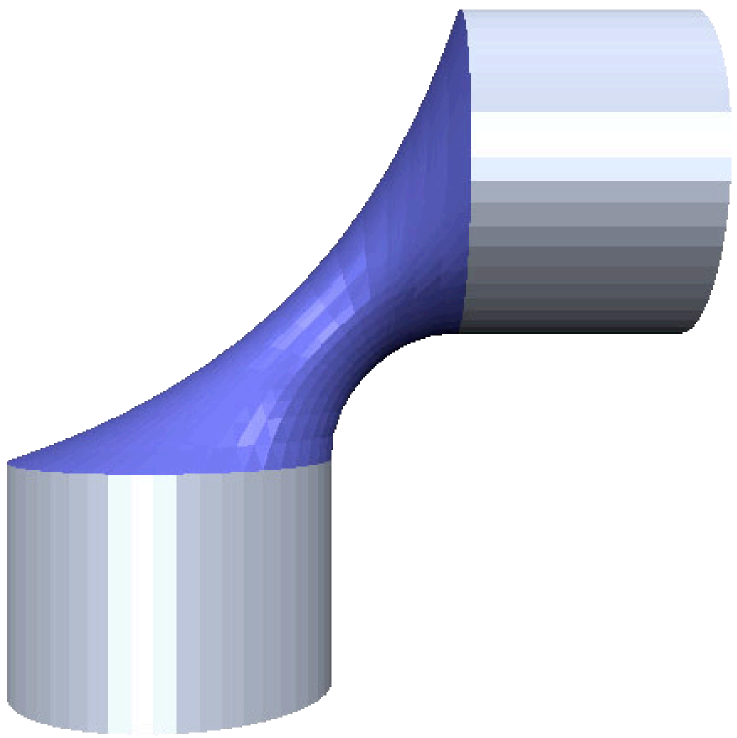
$$\int_\Omega f_{uu}^2 + 2f_{uv}^2 + f_{vv}^2 du dv$$



  - Euler-Lagrange PDE

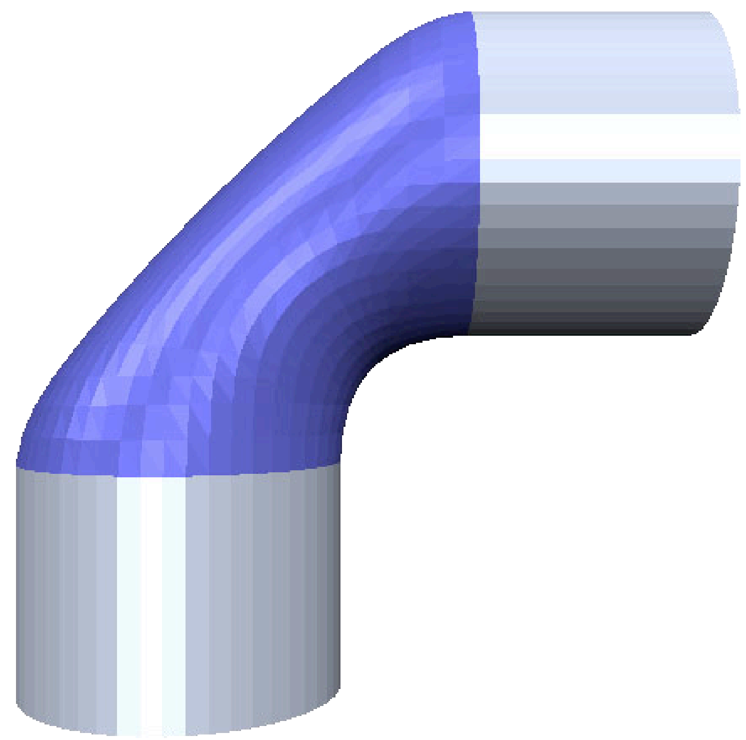
$$\Delta^2 f = f_{uuuu} + 2f_{uuvv} + f_{vvvv} = 0$$

Comparison



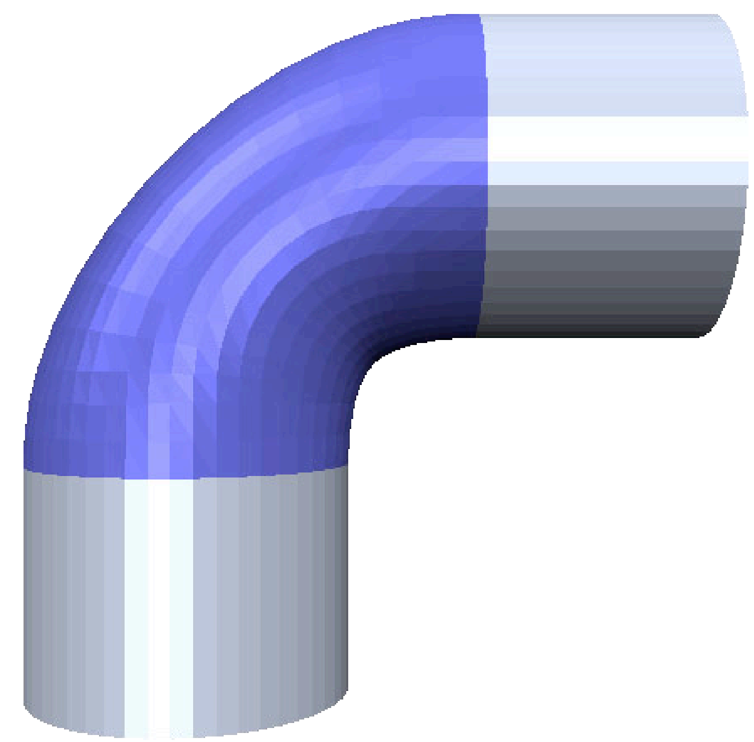
Membrane

$$\Delta_S p = 0$$



Thin Plate

$$\Delta_S^2 p = 0$$



$$\Delta_S^3 p = 0$$

Outline

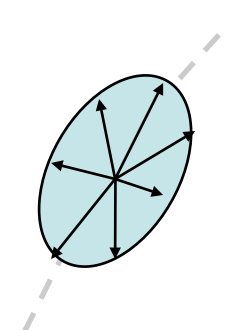
- Motivation
- Smoothing as *Diffusion*
- Smoothing as *Energy Minimization*
- Alternative Approaches

Alternative Approaches

- Anisotropic Diffusion
  - Data-dependent
  - Non-linear

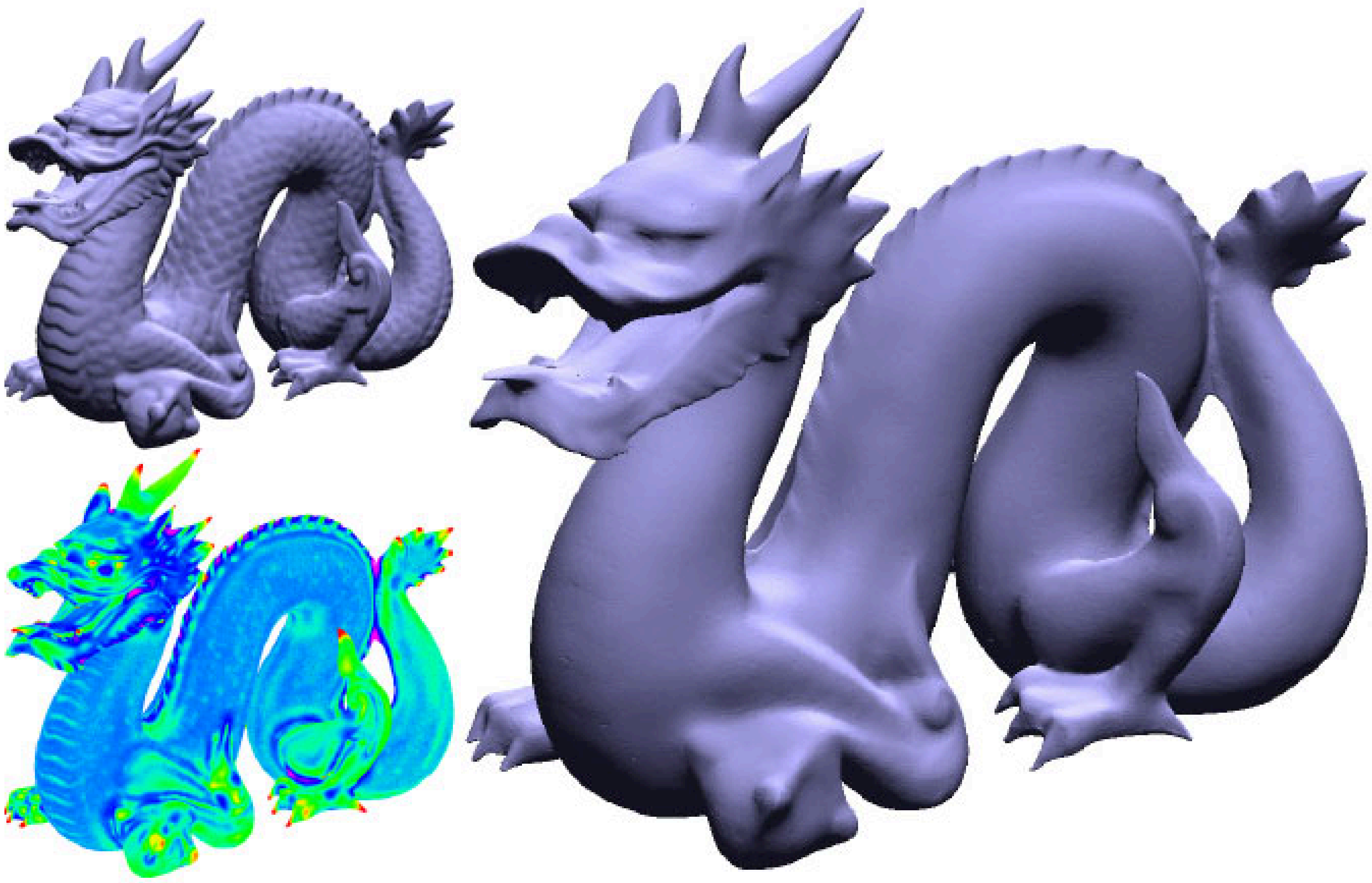
$$\frac{\partial}{\partial t} x = \operatorname{div} D \nabla x$$

diffusion tensor



- Normal filtering
  - Smooth normal field and reconstruct *(mesh editing)*
- Non-linear PDE (e.g., $\Delta_S H = 0$)
  - Avoid parameter dependence for fair surface design
- Bilateral Filtering

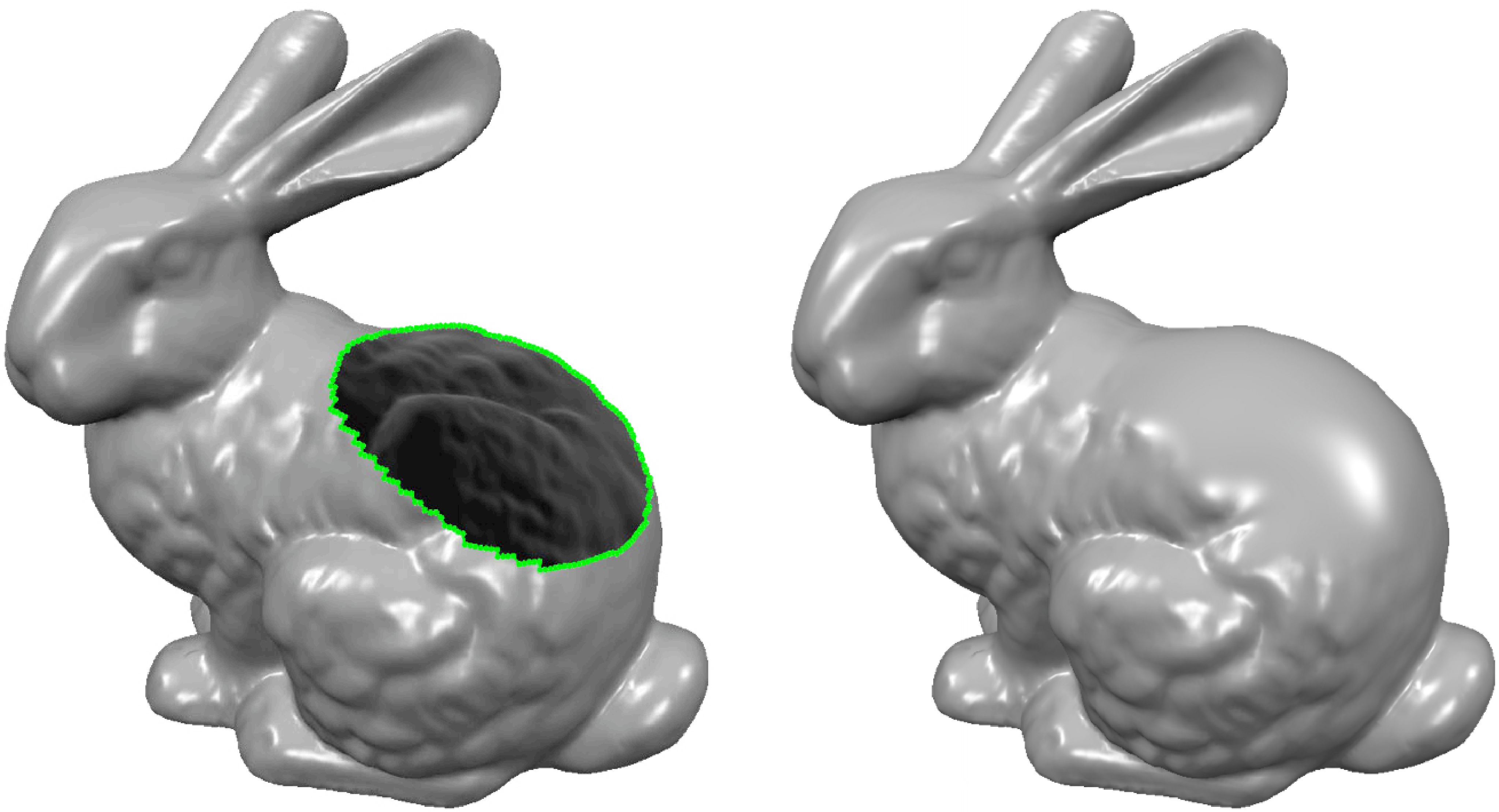
Example of Bilateral Filtering



Jones, Durand, Desbrun: *Non-iterative feature preserving mesh smoothing*, SIGGRAPH 2003

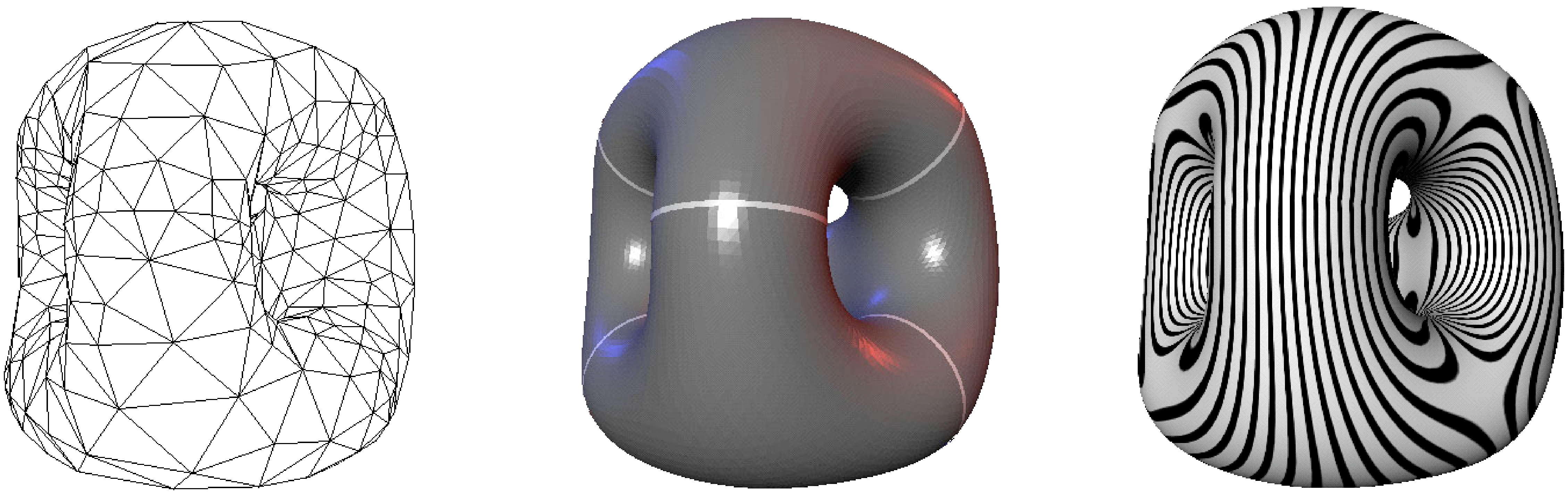
Applications

- Hole-filling



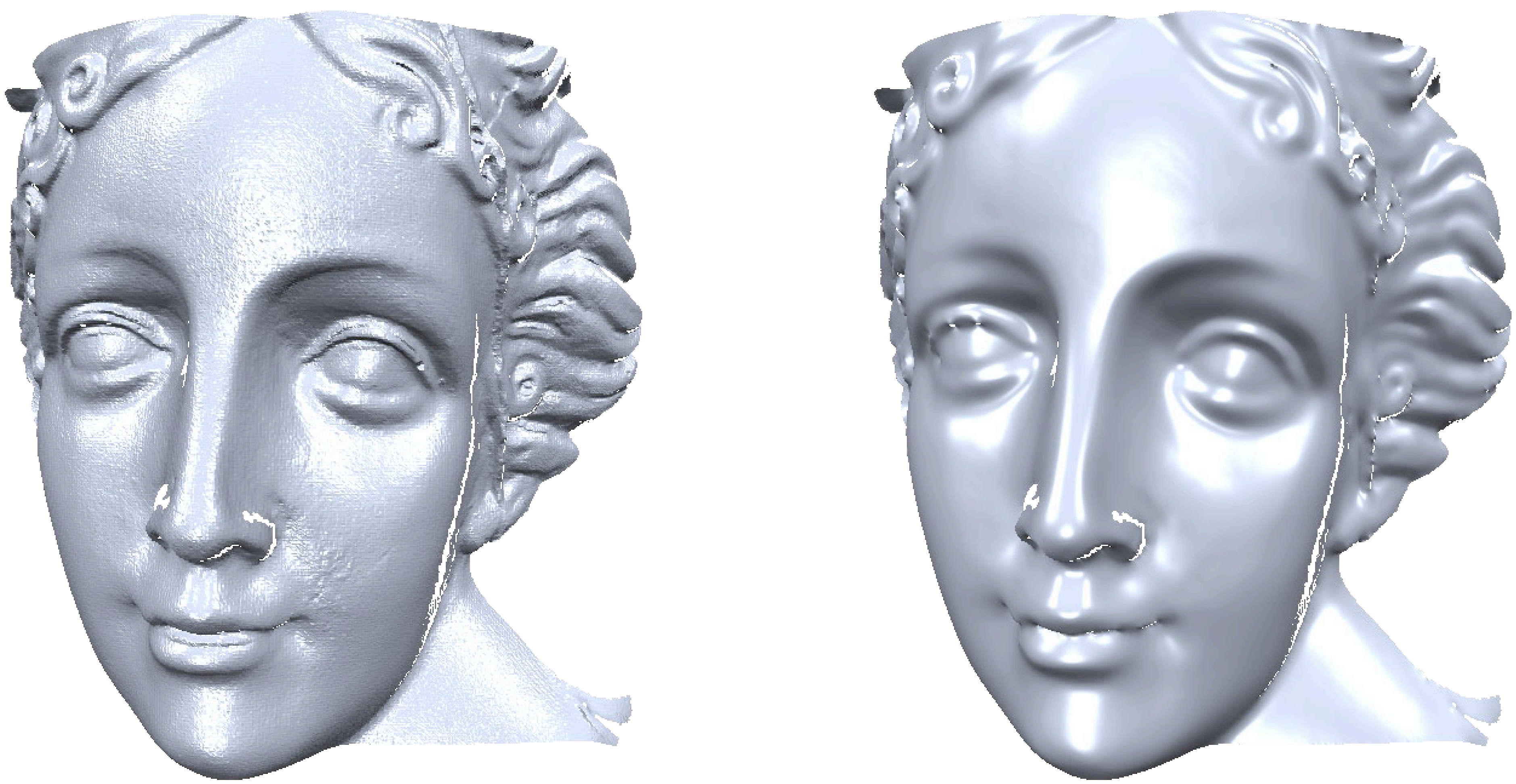
Applications

- Fair surface design



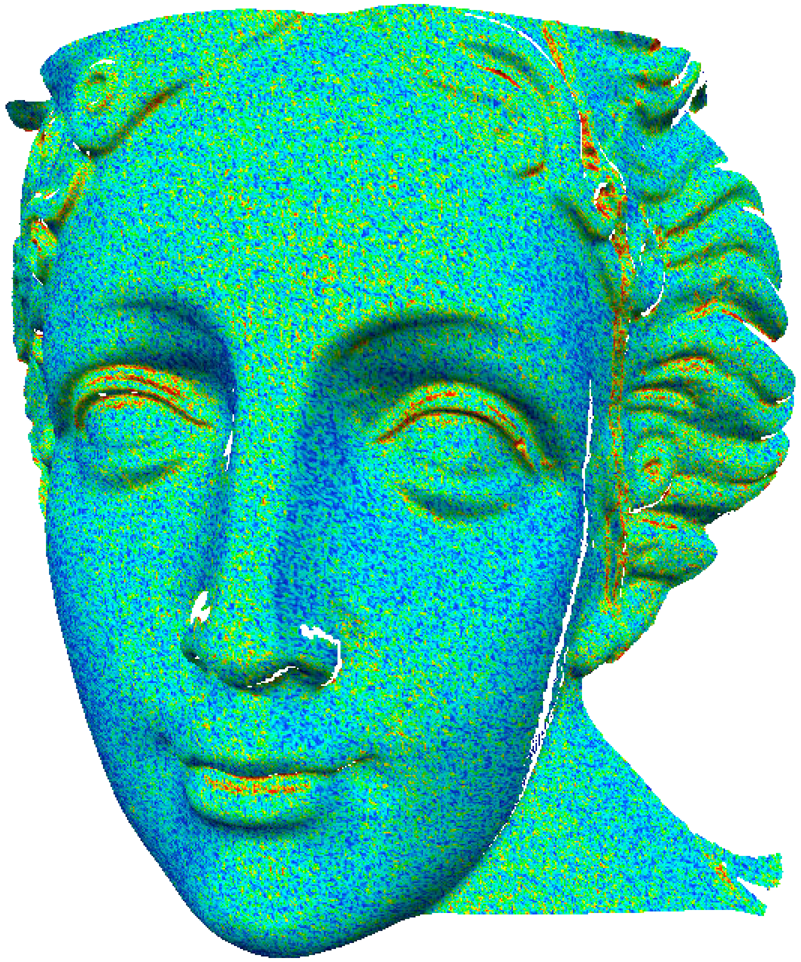
Applications

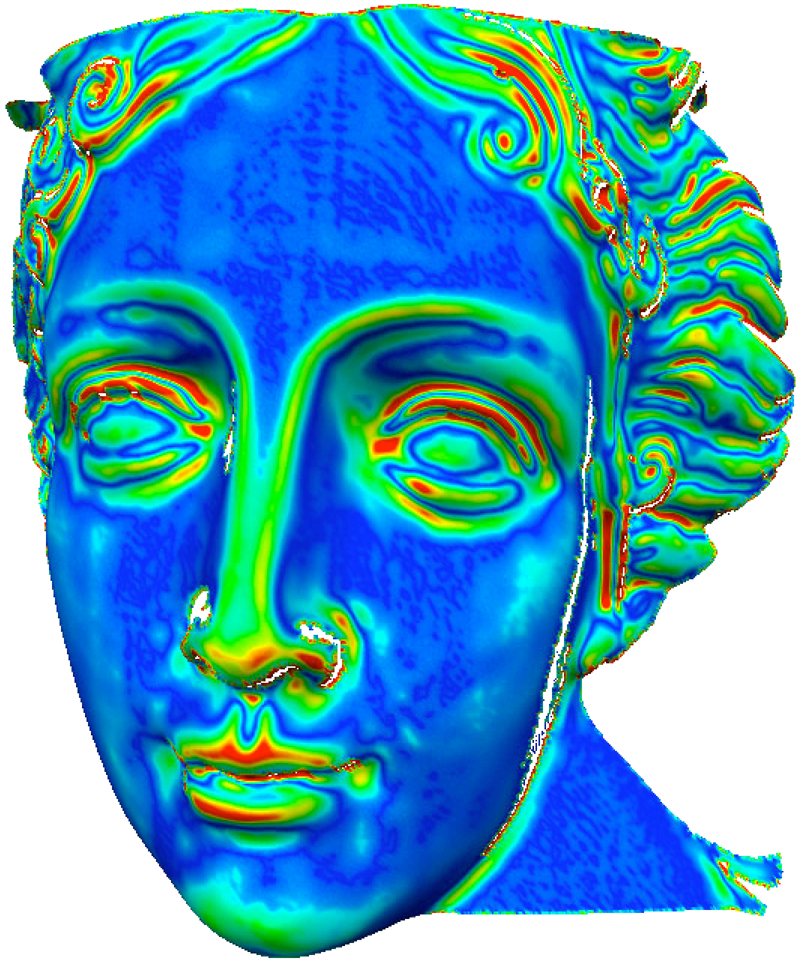
- Noise removal



Applications

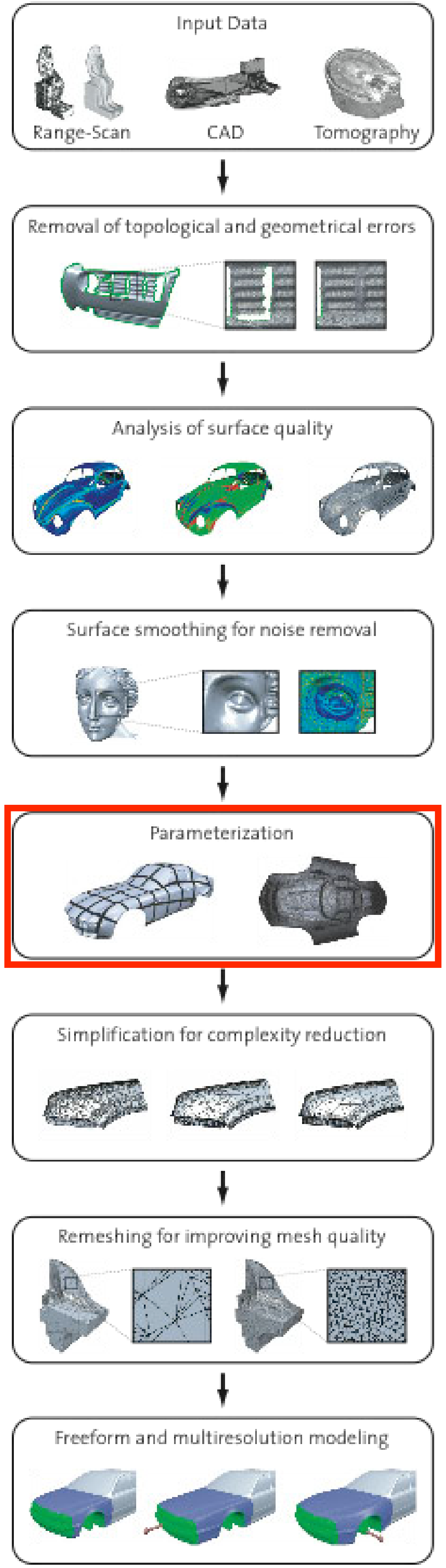
- Noise removal





Literature

- Taubin: *A signal processing approach to fair surface design*, SIGGRAPH 1996
- Desbrun, Meyer, Schroeder, Barr: *Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow*, SIGGRAPH 99
- Botsch, Kobbelt: *An Intuitive Framework for Real-Time Freeform Modeling*, SIGGRAPH 2004
- Fleishman, Drori, Cohen-Or: *Bilateral mesh denoising*, SIGGRAPH 2003
- Jones, Durand, Desbrun: *Non-iterative feature preserving mesh smoothing*, SIGGRAPH 2003



Surface Parameterization

Christian Rössl
INRIA Sophia-Antipolis

Outline

- Motivation
- Objectives and Discrete Mappings
- Angle Preservation
- Reducing Area Distortion
- Alternative Domains

Surface Parameterization

Mercator-Projektion

Mollweide-Projektion

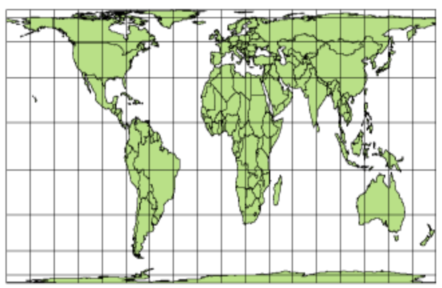
Surface Parameterization

Mollweide-Projektion

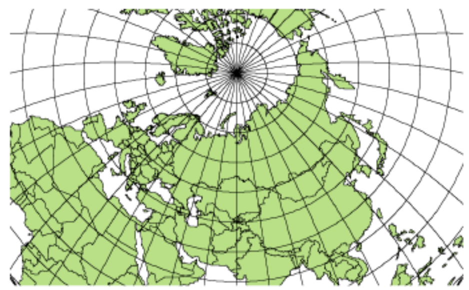
Mercator-Projektion

Zylinderprojektion nach Miller

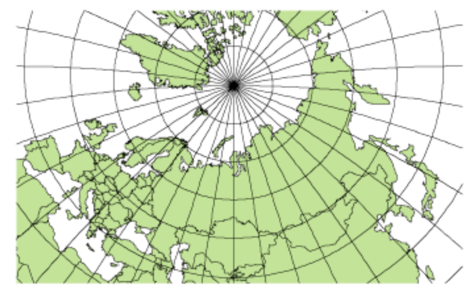
Hammer-Aitoff-Projektion



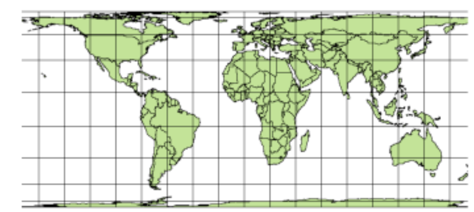
Peters-Projektion



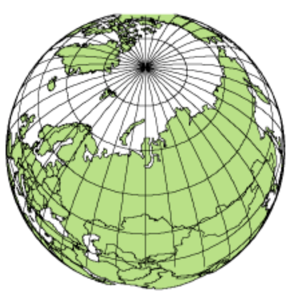
Längentreue Azimuthalprojektion



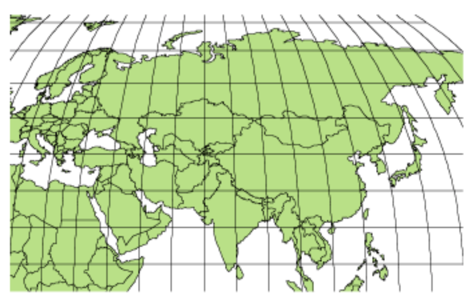
Stereographische Projektion



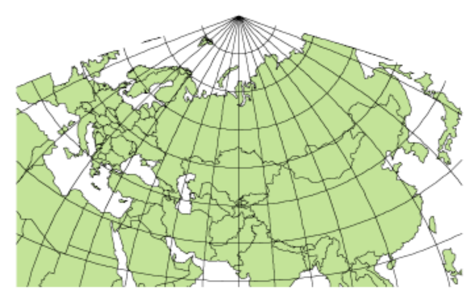
Behrmann-Projektion



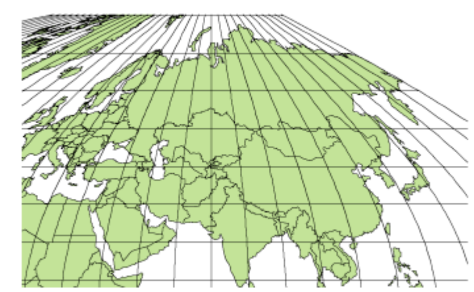
Senkrechte Umgebungsperspektive



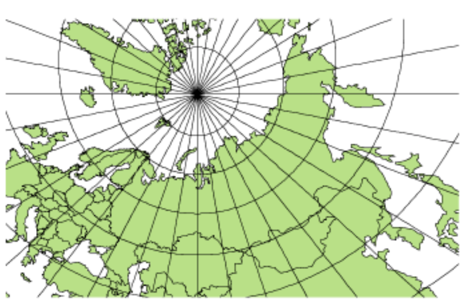
Robinson-Projektion



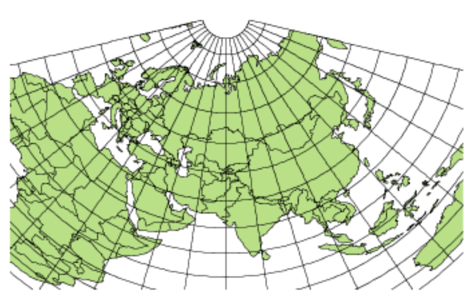
Hotine Oblique Mercator-Projektion



Sinusoidale Projektion

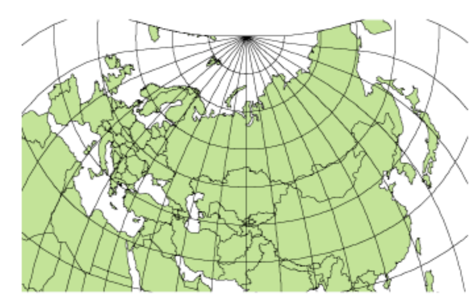


Gnomonische Projektion



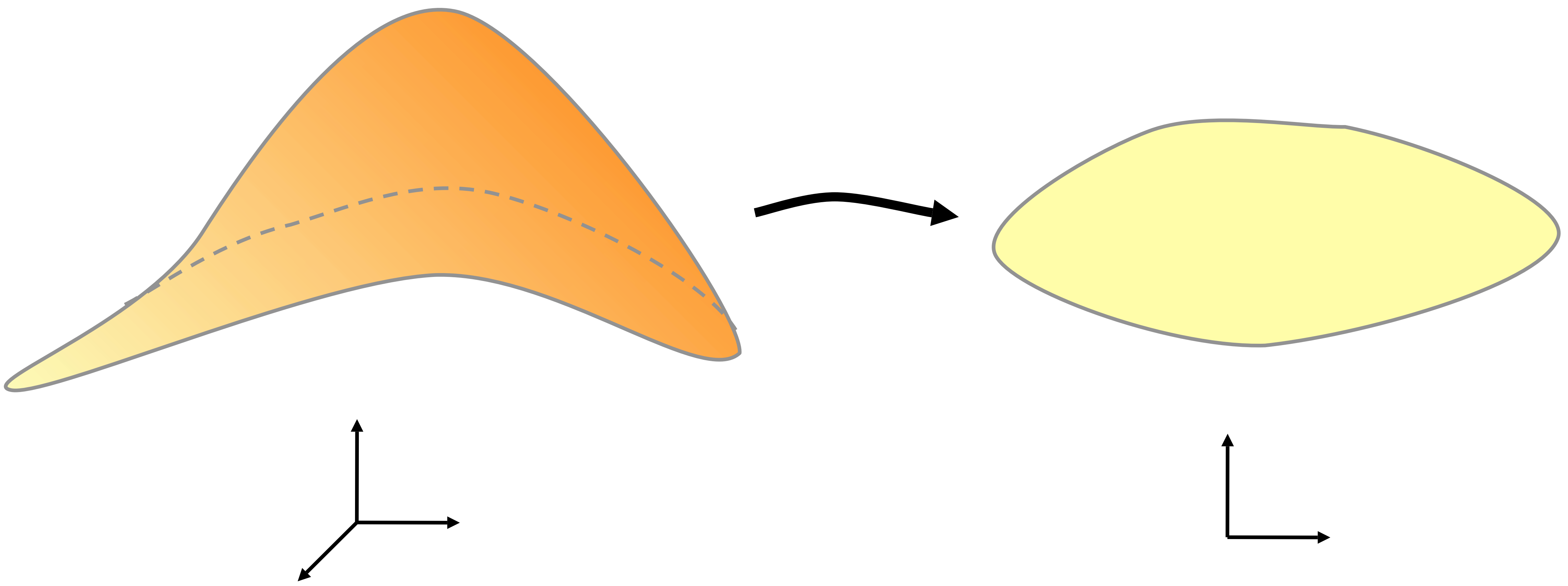
Flächentreue Kegelprojektion

Transverse Mercator-Projektion



Cassini-Soldner-Projektion

Surface Parameterization



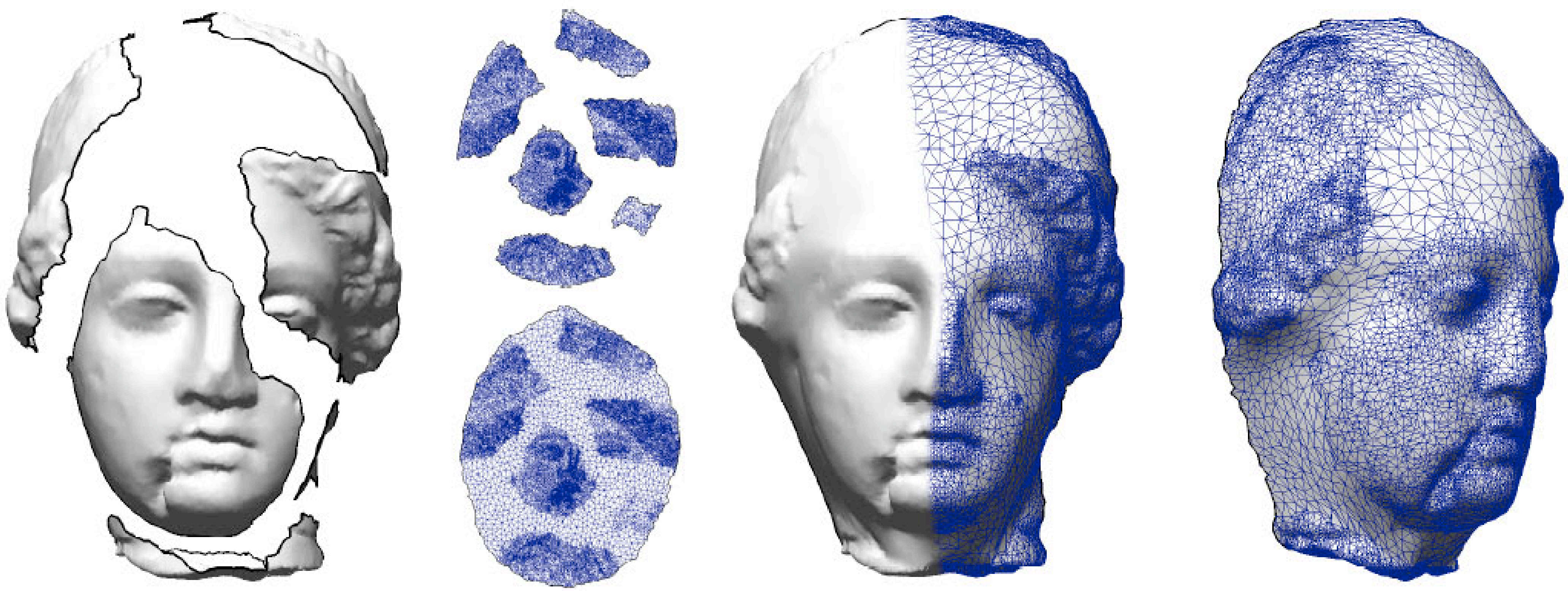
Motivation

- Texture mapping

Lévy, Petitjean, Ray, and Maillot: *Least squares conformal maps for automatic texture atlas generation*, SIGGRAPH 2002

Motivation

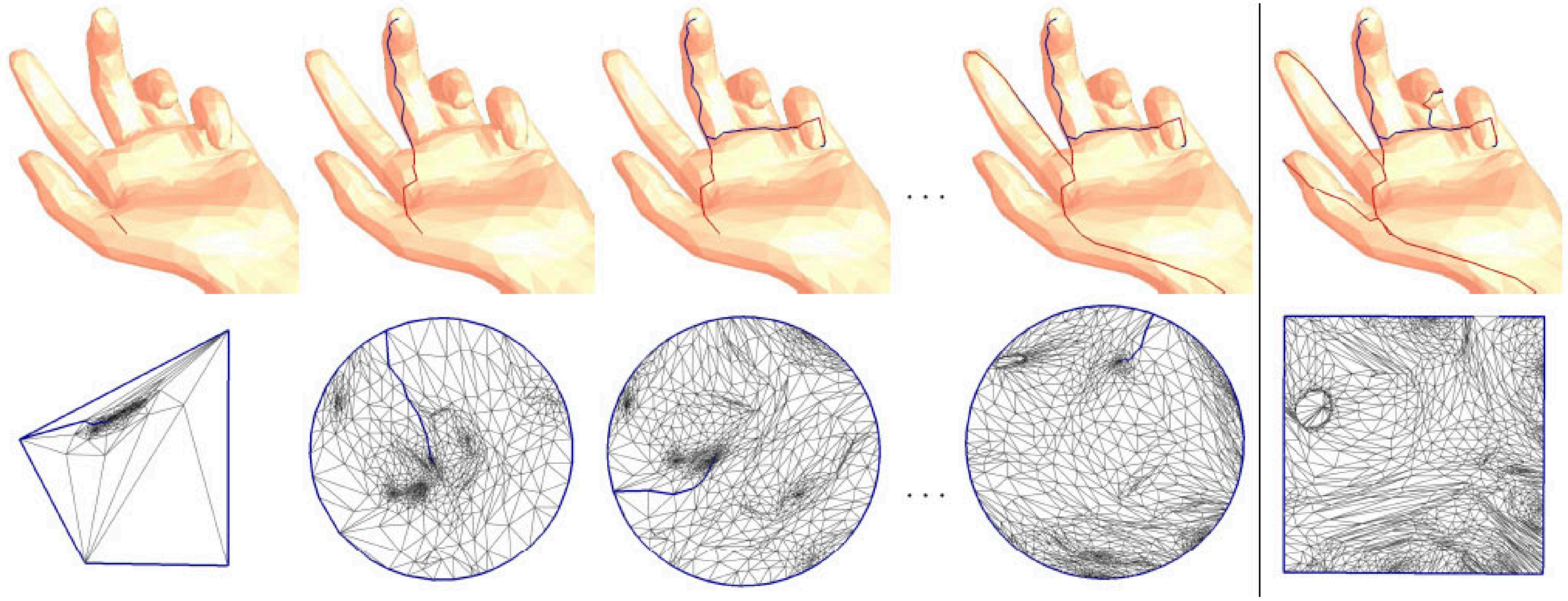
- Many operations are simpler on planar domain

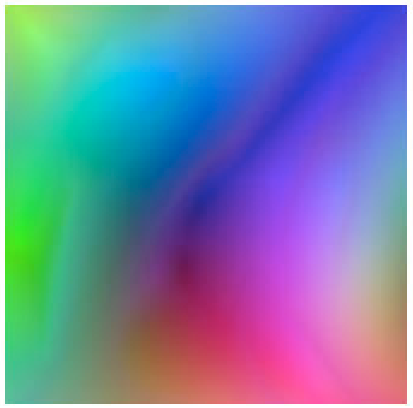


Lévy: *Dual Domain Exrapolation*, SIGGRAPH 2003

Motivation

- Exploit regular structure in domain



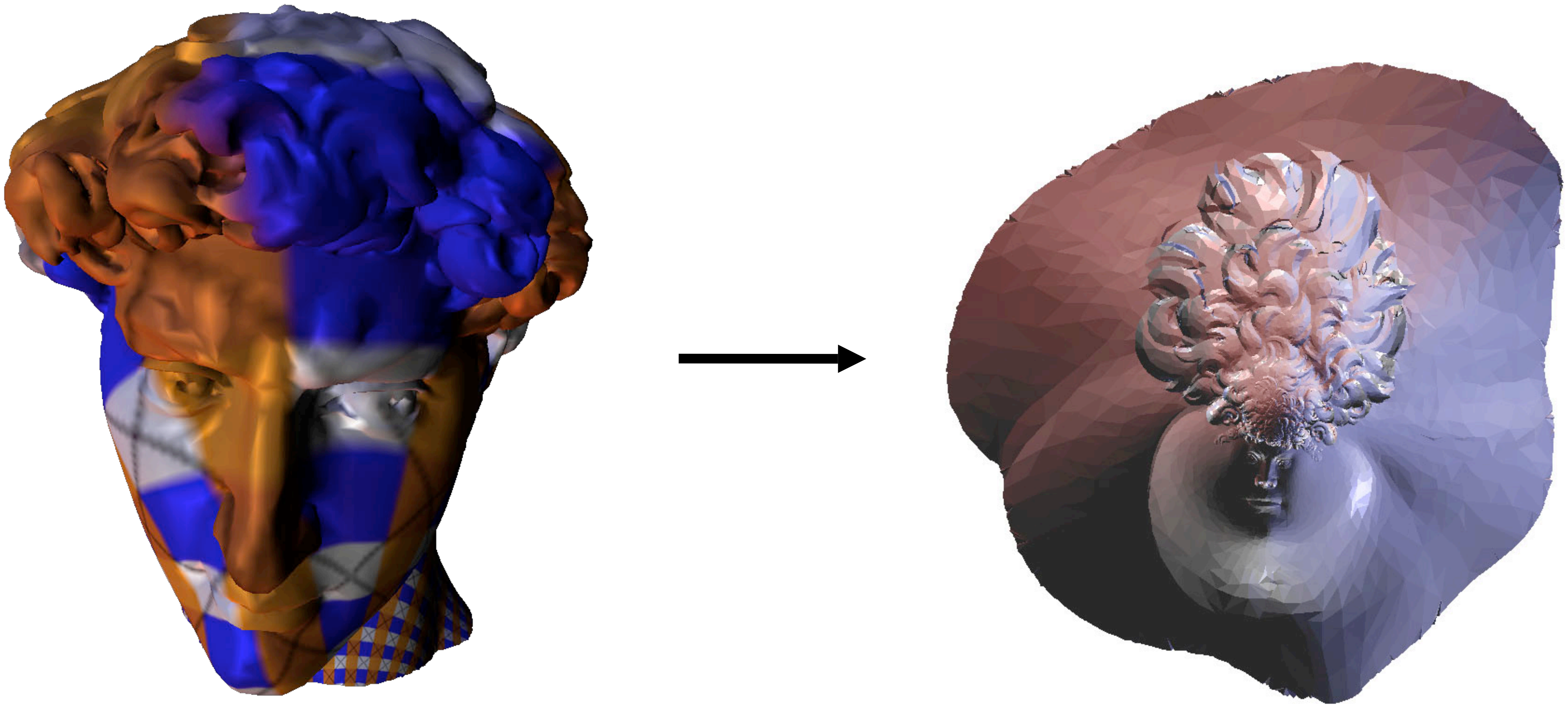


Gu, Gortler, Hoppe: *Geometry Images*, SIGGRAPH 2002

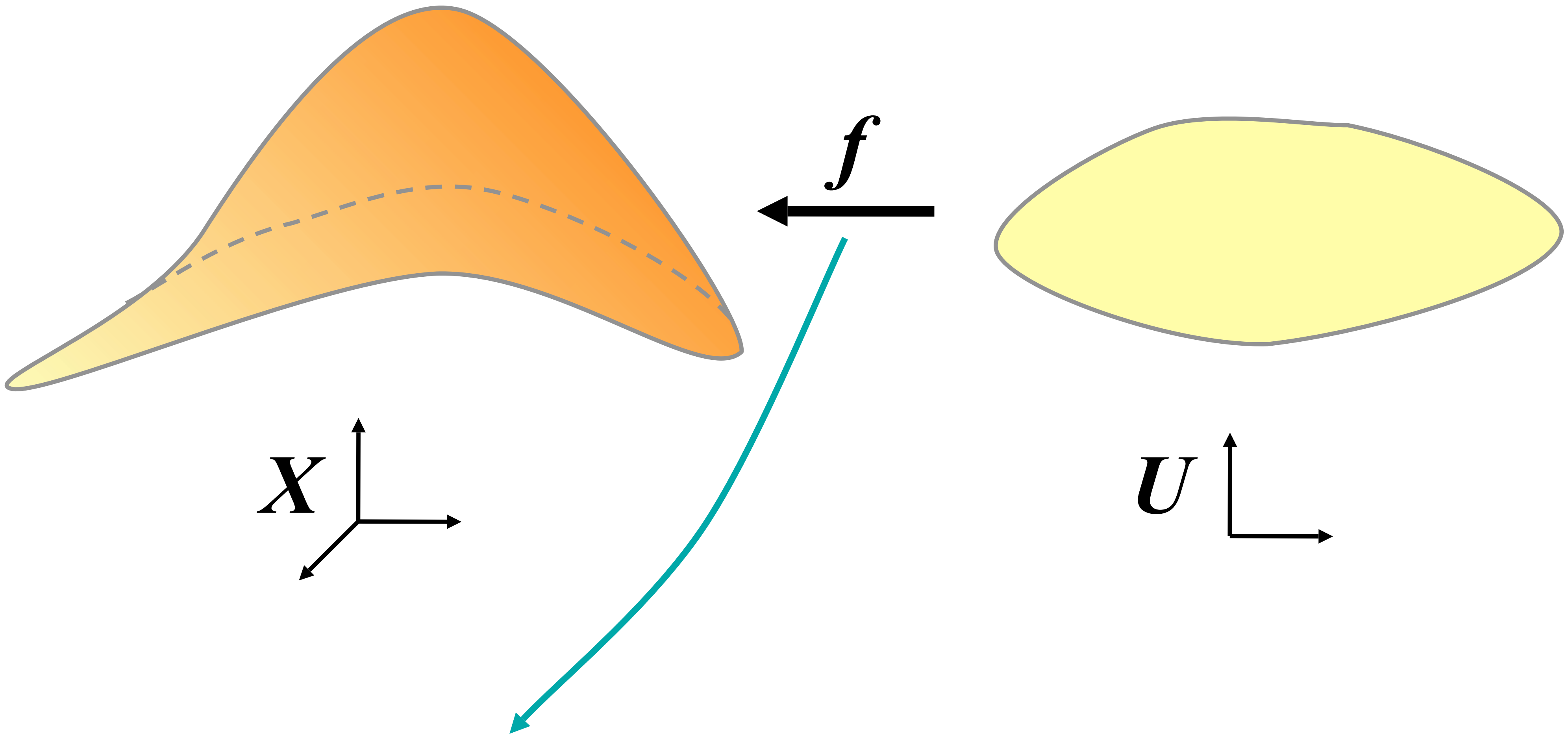
Outline

- Motivation
- Objectives and Discrete Mappings
  - Characterization of mappings
  - Discrete mappings
- Angle Preservation
- Reducing Area Distortion
- Alternative Domains

Surface Parameterization



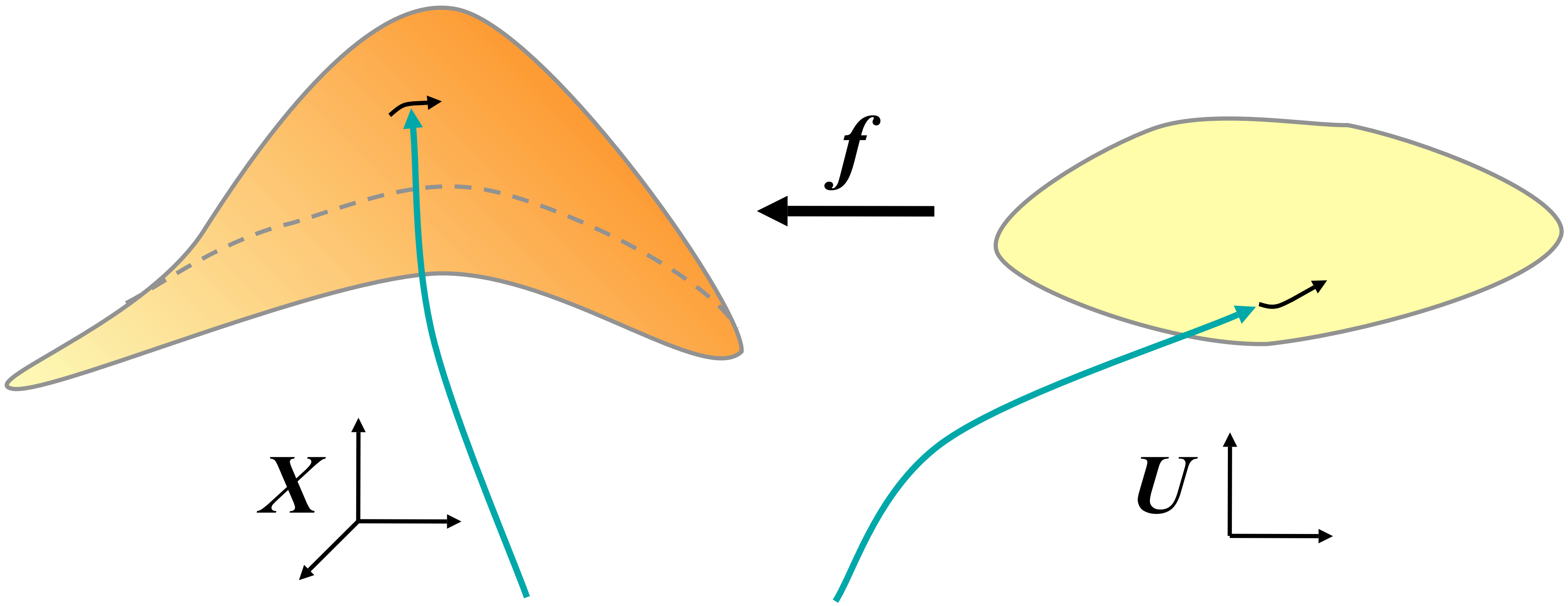
Surface Parameterization



$$f(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} \qquad J = \begin{pmatrix} x_u & x_v \\ y_u & y_v \\ z_u & z_v \end{pmatrix}$$

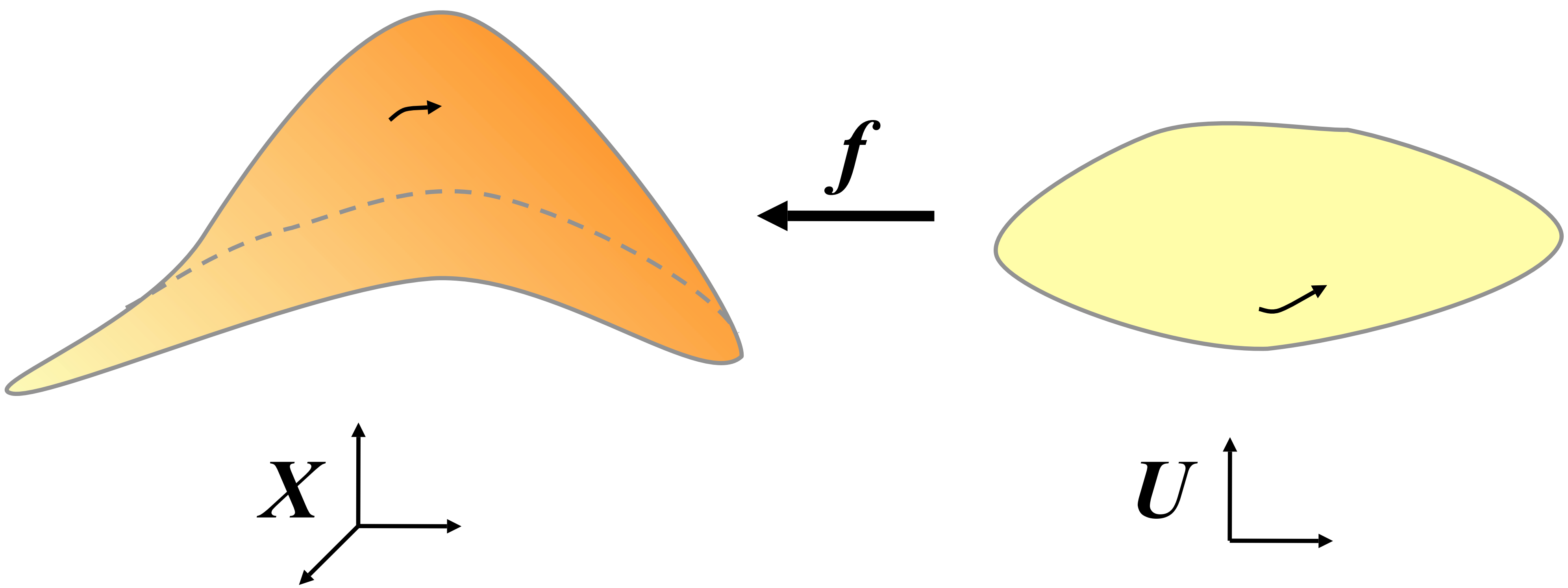
Jacobian

Surface Parameterization



$$dX = J\, dU$$

Surface Parameterization

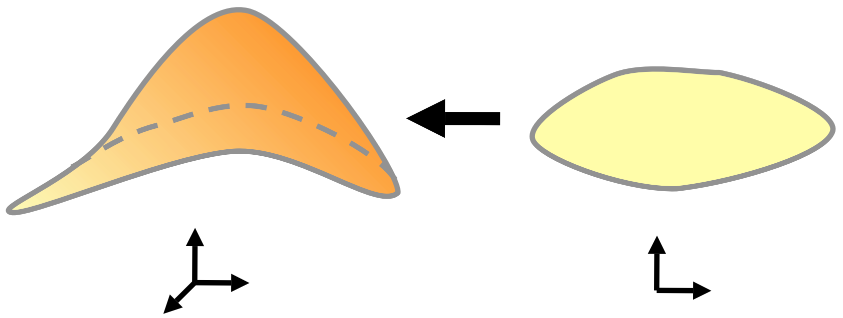


$$d\boldsymbol{X} = \boldsymbol{J}\, d\boldsymbol{U}$$

$$\|d\boldsymbol{X}\|^2 = d\boldsymbol{U}\, \underbrace{\boldsymbol{J}^T\boldsymbol{J}}\, d\boldsymbol{U}$$ First Fundamental Form

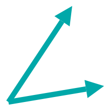
$$\mathbf{I} = \begin{pmatrix} x_u x_u & x_u x_v \\ x_u x_v & x_v x_v \end{pmatrix}$$

Characterization of Mappings



- By first fundamental form $\boldsymbol{I}$
  - Eigenvalues $\lambda_{1,2}$ of $\boldsymbol{I}$
  - Singular values $\sigma_{1,2}$ of $\boldsymbol{J}$ ($\sigma_i^2 = \lambda_i$)

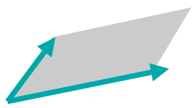
- *Isometric*
  - $\boldsymbol{I} = \boldsymbol{Id}$, $\lambda_1 = \lambda_2 = 1$



- *Conformal*
  - $\boldsymbol{I} = \mu\, \boldsymbol{Id}$, $\lambda_1 / \lambda_2 = 1$

angle preserving

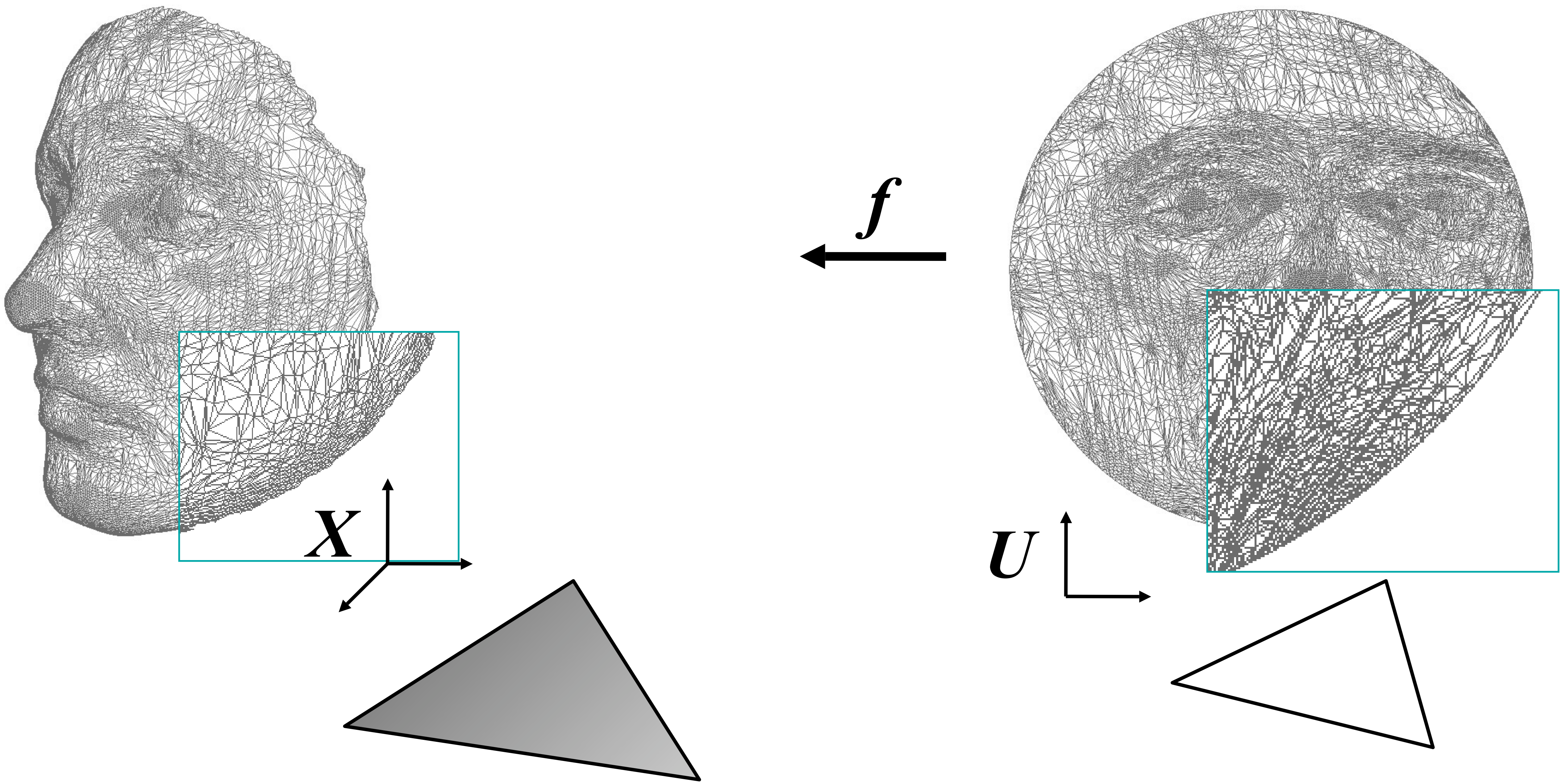
- *Equiareal*
  - $\det \boldsymbol{I} = 1$, $\lambda_1 \lambda_2 = 1$



area preserving

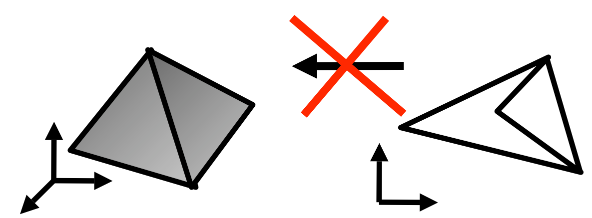
Piecewise Linear Maps

- Mapping = 2D mesh with same connectivity



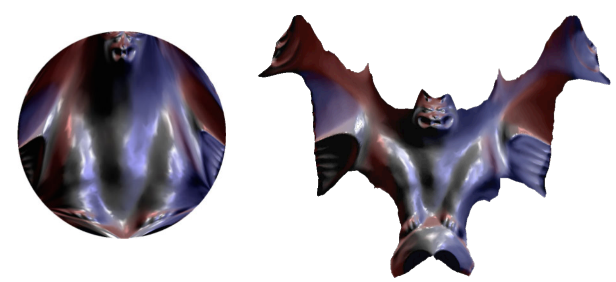
Objectives

- Isometric maps are rare
- Minimize distortion w.r.t. a certain measure
  - Validity (bijective map)



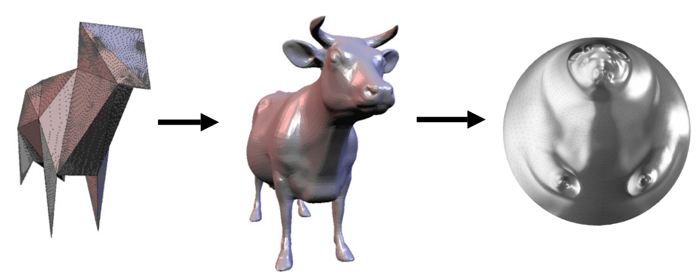
triangle flip

  - Boundary



fixed / free?

  - Domain



e.g.,spherical

  - Numerical solution

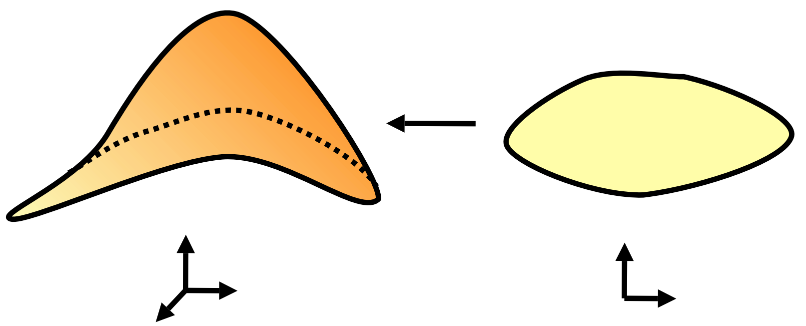
linear / non-linear?

Outline

- Motivation
- Objectives and Discrete Mappings
- Angle Preservation
  - Discrete Harmonic Maps
  - Discrete Conformal Maps
- Reducing Area Distortion
- Alternative Domains

Discrete Harmonic Maps

- f is *harmonic* if $\Delta f = 0$



- Solve Laplace equation

$$\begin{cases} \Delta u = 0 \\ \Delta v = 0 \\ (u, v)_{|\partial\Omega} = (u_0, v_0) \end{cases}$$

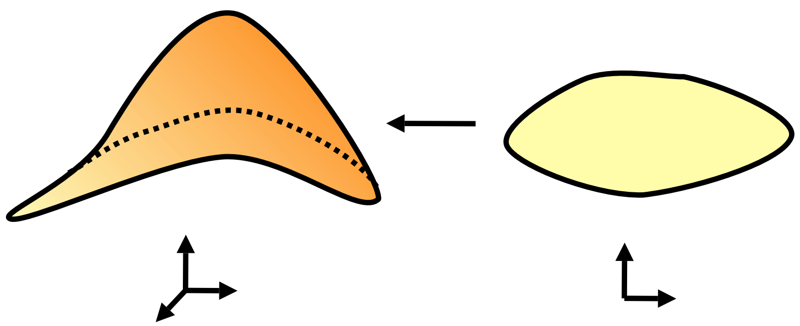
u and v are *harmonic*

Dirichlet boundary conditions

- In 3D: "fix planar boundary and smooth"

Discrete Harmonic Maps

- f *is harmonic* if $\Delta f = 0$
- Solve Laplace equation
- Yields linear system (again)

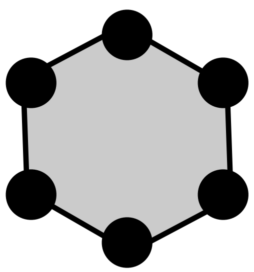


$$L(p_i) = \sum_{j \in N_i} w_{ij}(p_j - p_i) = 0 \qquad \text{vertices } 1 \le i \le n$$

- *Convex combination maps*
  - *Normalization* $$\sum_{j \in N_i} w_{ij} = 1$$
  - *Positivity* $$w_{ij} > 0$$

Convex Combination Maps

- Every (interior) planar vertex is a *convex combination* of its neighbors



- Guarantees *validity* if boundary is mapped to a convex polygon (e.g., rectangle, circle)
- Weights
  - Uniform *(barycentric mapping)*
  - Shape preserving [Floater 1997]
  - Mean Value Coordinates [Floater 2003]
    - Use mean value property of harmonic functions

Reproduction of planar meshes

Outline

- Motivation
- Objectives and Discrete Mappings
- Angle Preservation
  - Discrete Harmonic Maps
  - Discrete Conformal Maps
- Reducing Area Distortion
- Alternative Domains

Conformal Maps

- Planar *conformal mappings* $f(x,y) = \begin{pmatrix} u(x,y) \\ v(x,y) \end{pmatrix}$ satisfy the *Cauchy-Riemann conditions*

$$\frac{\partial u(x,y)}{\partial x} = \frac{\partial v(x,y)}{\partial y} \quad \text{and} \quad \frac{\partial u(x,y)}{\partial y} = -\frac{\partial v(x,y)}{\partial x}$$

Conformal Maps

- Planar *conformal mappings* $f(x,y) = \begin{pmatrix} u(x,y) \\ v(x,y) \end{pmatrix}$ satisfy the *Cauchy-Riemann conditions*

$$u_x = v_y \qquad \text{and} \qquad u_y = -v_x$$

- Differentiating once more by x and y yields

$$u_{xx} = v_{xy} \text{ and } u_{yy} = -v_{xy} \qquad \Rightarrow \qquad u_{xx} + u_{yy} = \Delta u = 0$$

and similar $\Delta v = 0$

- conformal $\Rightarrow$ harmonic

Discrete Conformal Maps

- Planar *conformal mappings* $f(x,y) = \begin{pmatrix} u(x,y) \\ v(x,y) \end{pmatrix}$ satisfy the *Cauchy-Riemann conditions*

$$u_x = v_y \quad \text{and} \quad u_y = -v_x$$

- *In general, there are no conformal mappings for piecewise linear functions!*

Discrete Conformal Maps

- Planar *conformal mappings* $f(x,y) = \begin{pmatrix} u(x,y) \\ v(x,y) \end{pmatrix}$ satisfy the *Cauchy-Riemann conditions*

$$u_x = v_y \qquad \text{and} \qquad u_y = -v_x$$

- Conformal energy *(per triangle T)*

$$E_T = (u_x - v_y)^2 + (u_y + v_x)^2$$

- Minimize

$$\sum_{T \in \mathrm{T}} E_T A_T \rightarrow \min$$

Discrete Conformal Maps

- Least-squares conformal maps [Lévy et al. 2002]

$$\sum_{T \in \mathrm{T}} E_T A_T \rightarrow \min \quad \text{where} \quad E_T = (u_x - v_y)^2 + (u_y + v_x)^2$$

- Satisfy Cauchy-Riemann conditions in *least-squares* sense
- Leads to solution of linear system

- *Alternative formulation leads to same solution…*

Discrete Conformal Maps

- Same solution is obtained for

$$\Delta_S u = 0$$

$$\Delta_S v = 0$$

cotangent weights

$$n \times \nabla u \,|_{\partial\Omega} = c$$

$$n \times \nabla v \,|_{\partial\Omega} = c$$

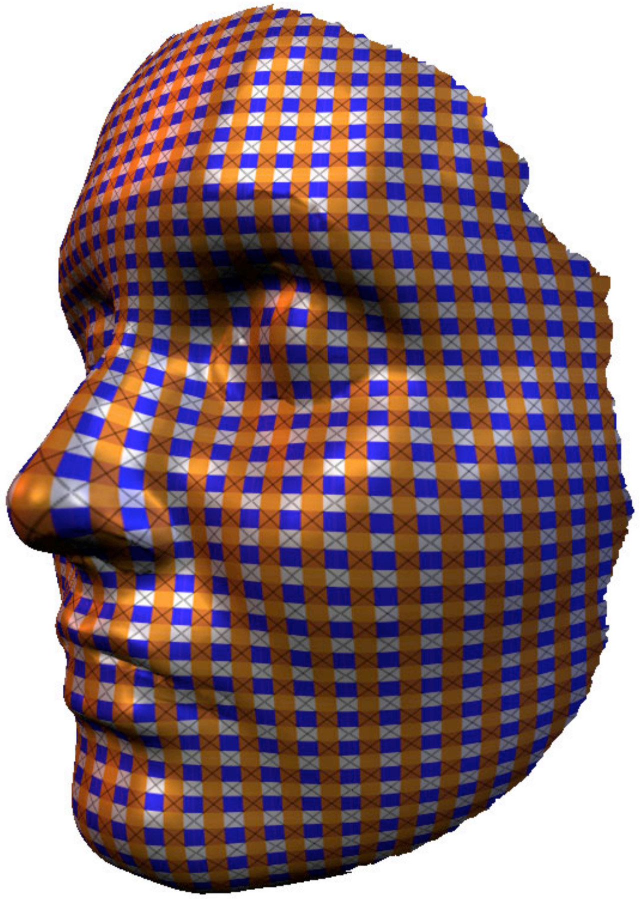
Neumann boundary conditions

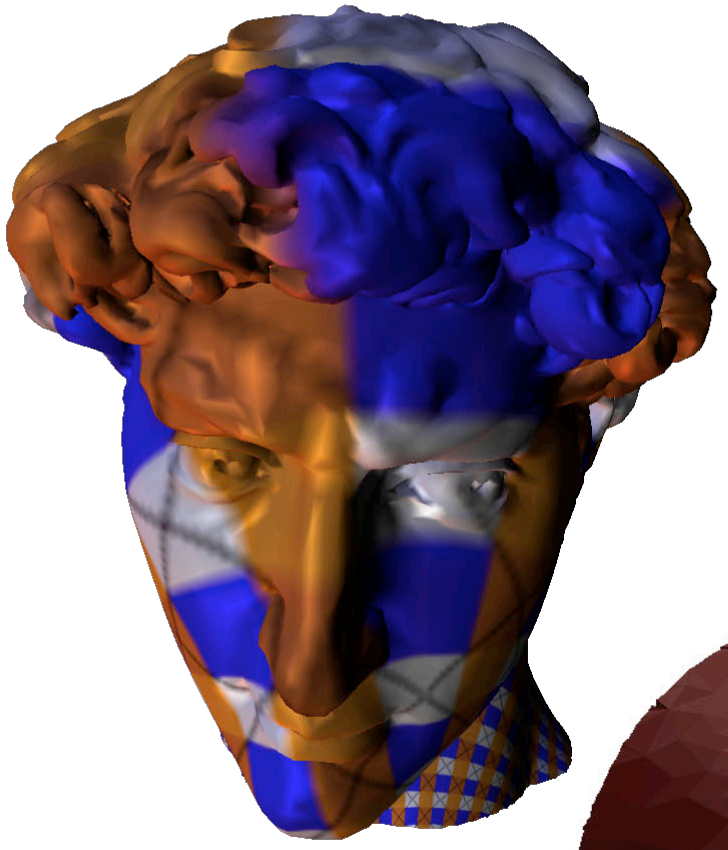
$$(u, v)_{|\partial\Omega_0} = (u_0, v_0)$$

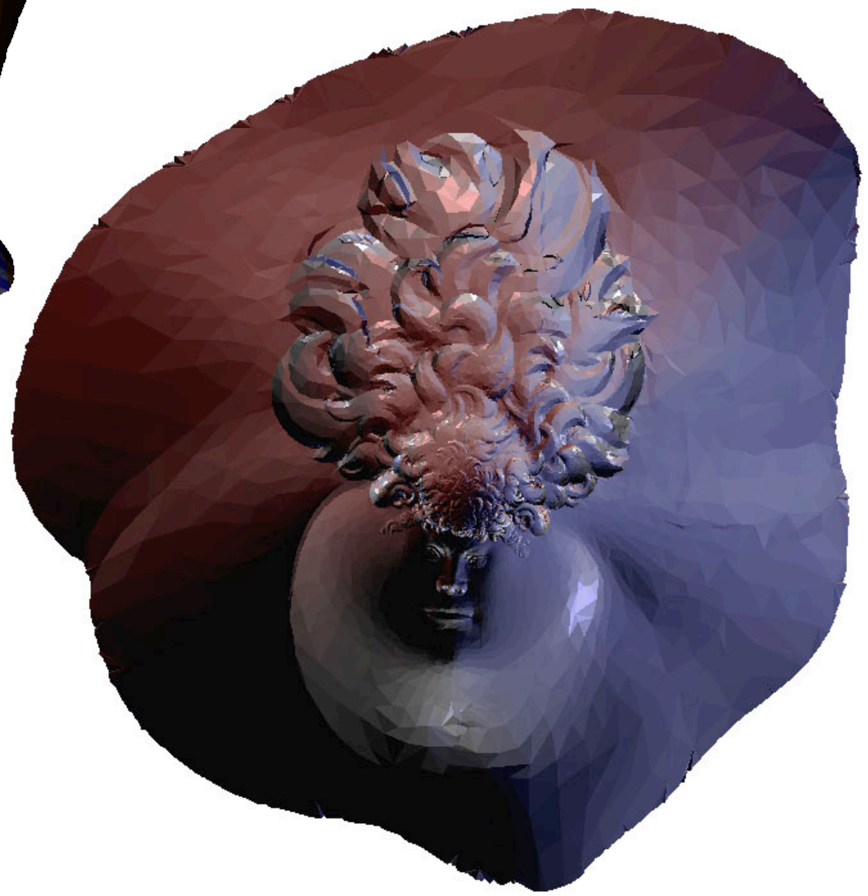
+ fixed vertices

Discrete Conformal Maps [Desbrun et al. 2002]

Discrete Conformal Maps

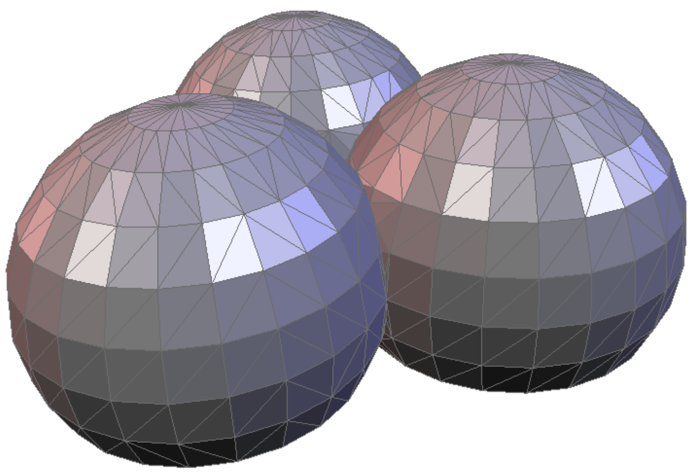


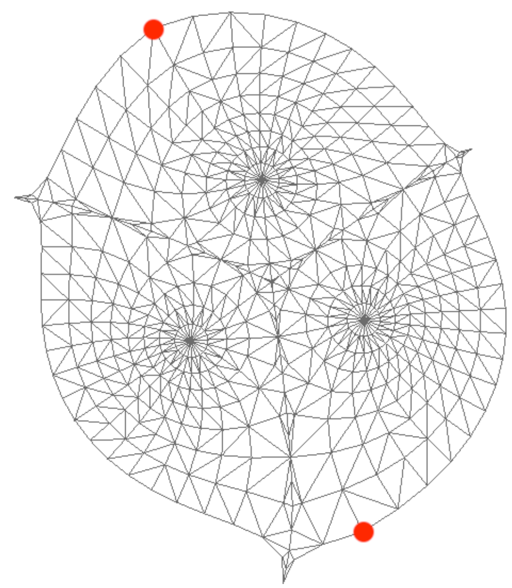


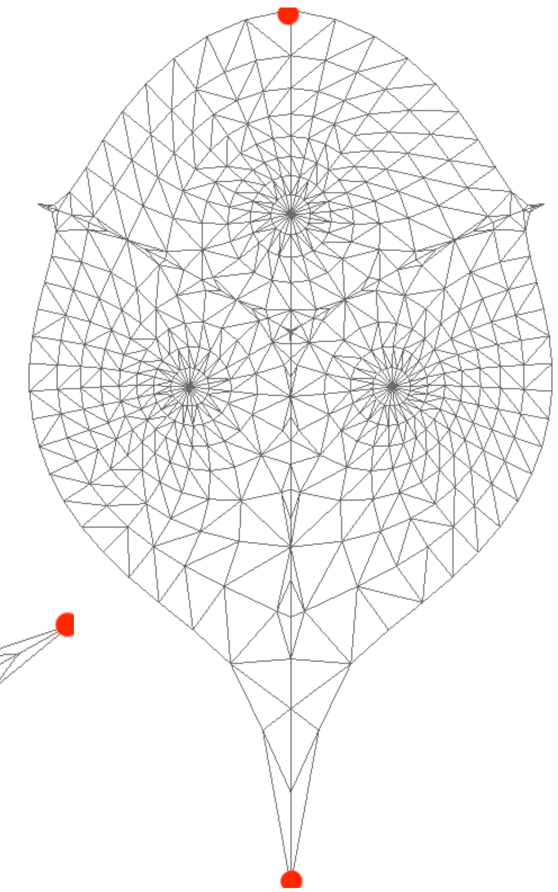


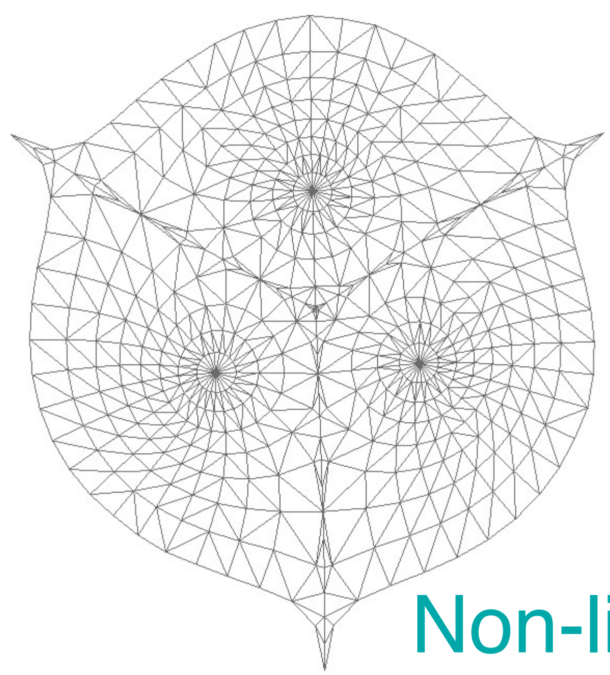
Discrete Conformal Maps

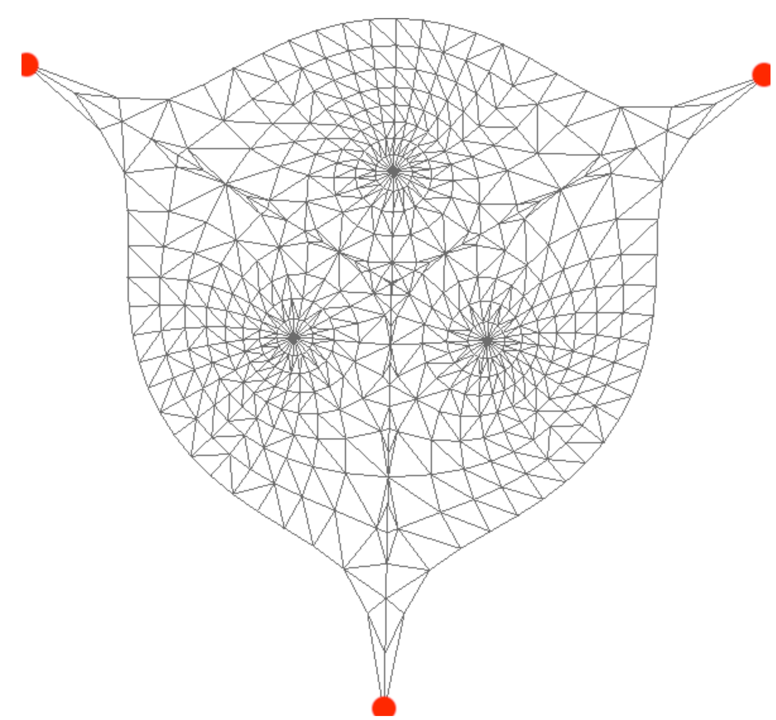
- Free boundary depends on choice of *fixed* vertices (>1)









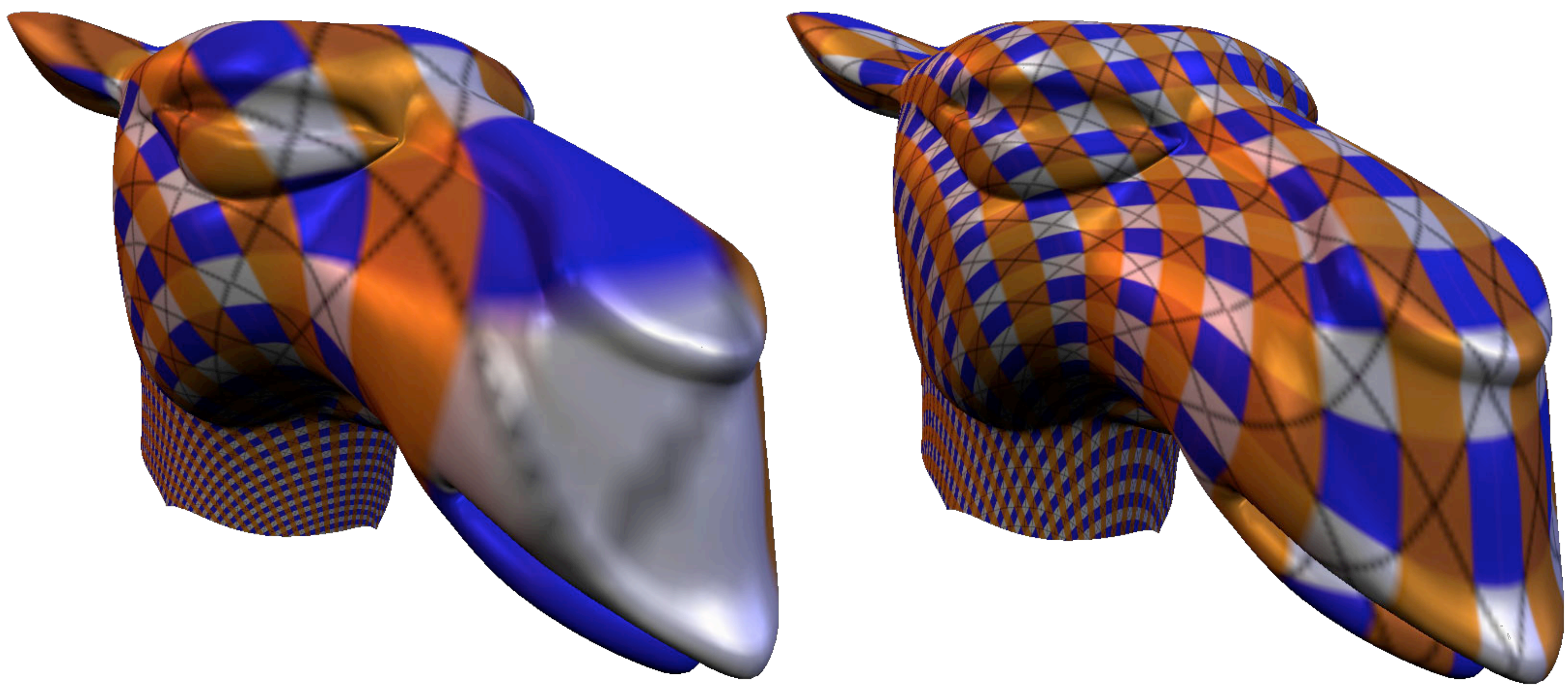


Non-linear ABF

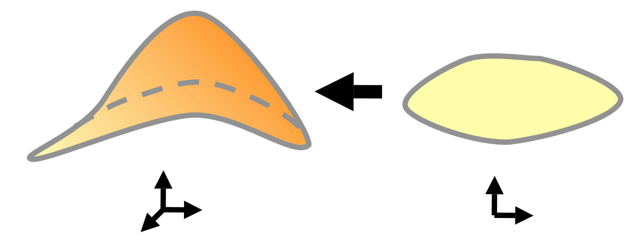
Outline

- Motivation
- Objectives and Discrete Mappings
- Angle Preservation
- Reducing Area Distortion
  - Non-linear optimization
  - Additional cuts
- Alternative Domains

And how about area distortion?



Reducing Area Distortion



- Energy minimization based on
  - MIPS [Hormann & Greiner 2000]

$$\|J\|_F \|J^{-1}\|_F = \frac{\sigma_1}{\sigma_2} + \frac{\sigma_2}{\sigma_1}$$

  - *modification* [Degener et al. 2003]

$$\det J + \frac{1}{\det J} = \sigma_1\sigma_2 + \frac{1}{\sigma_1\sigma_2}$$

  - "Stretch" [Sander et al. 2001]

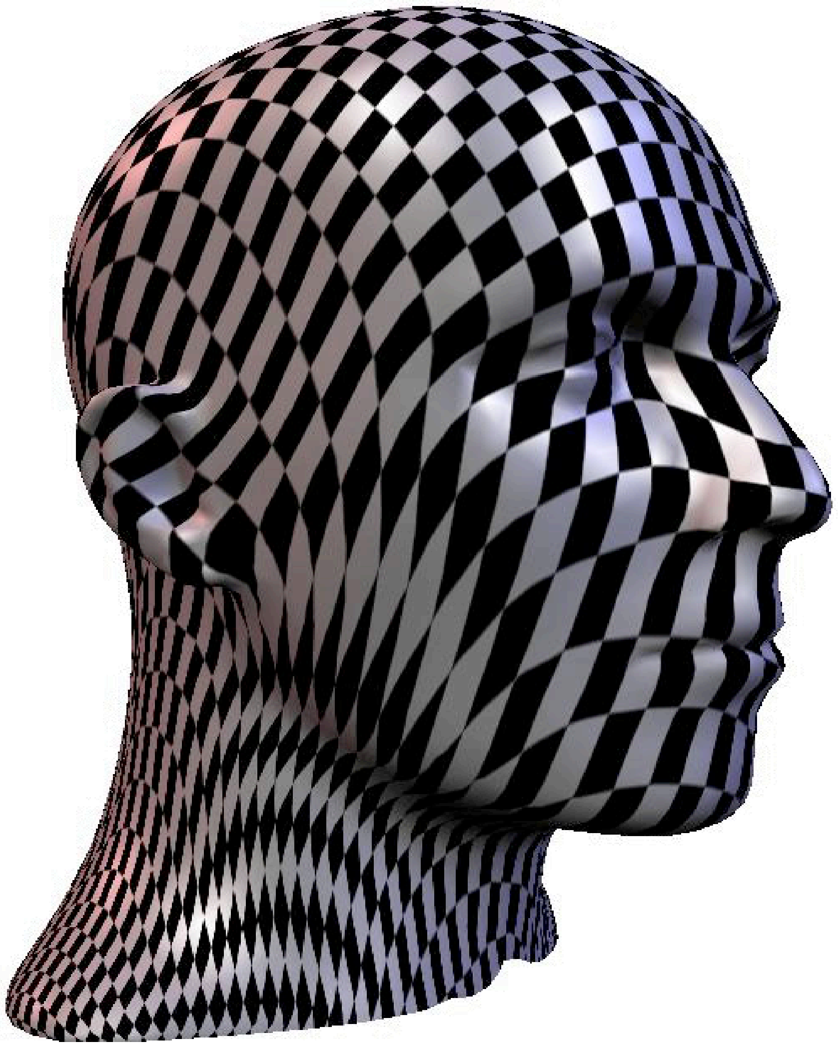
$$\|J\|_F = \sqrt{\sigma_1 + \sigma_2}$$ or $$\|J\|\infty = \sigma_1$$

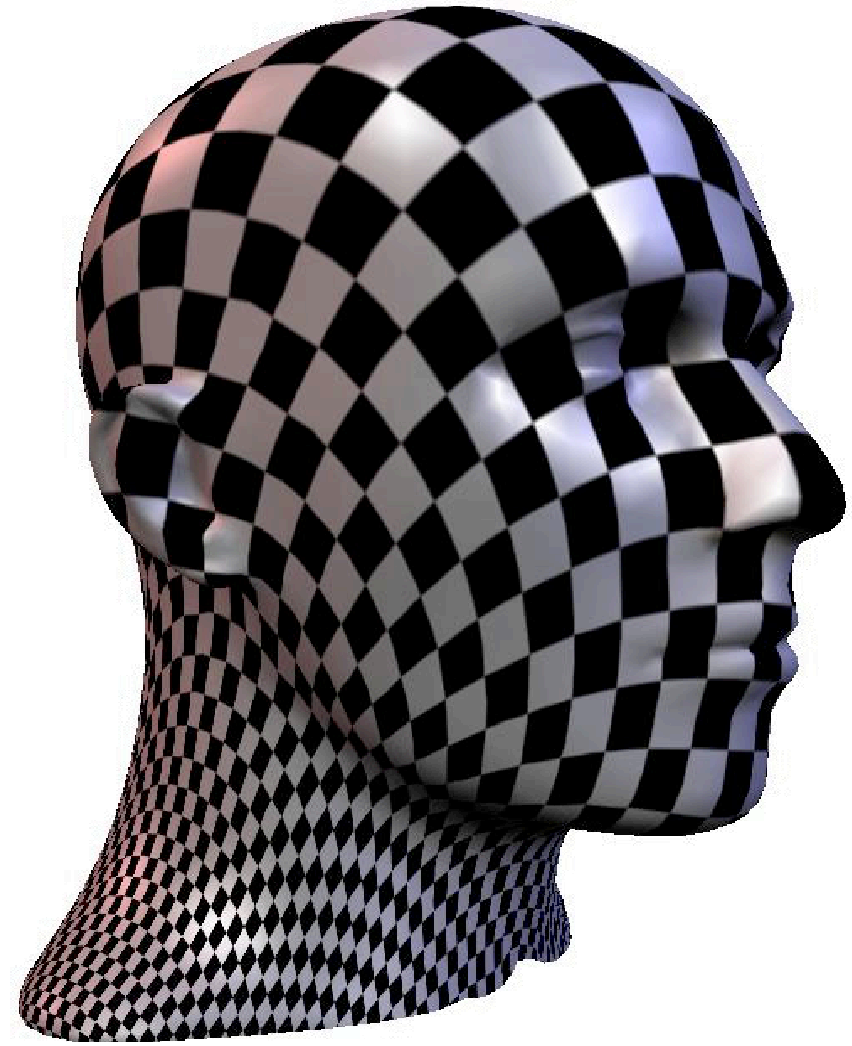
  - *modification* [Sorkine et al. 2002]

$$\max\{\sigma_1, \frac{1}{\sigma_2}\} \,|$$

Examples

"angles and area are competing"





$\sqrt{\sigma_1 + \sigma_2} \rightarrow \min$

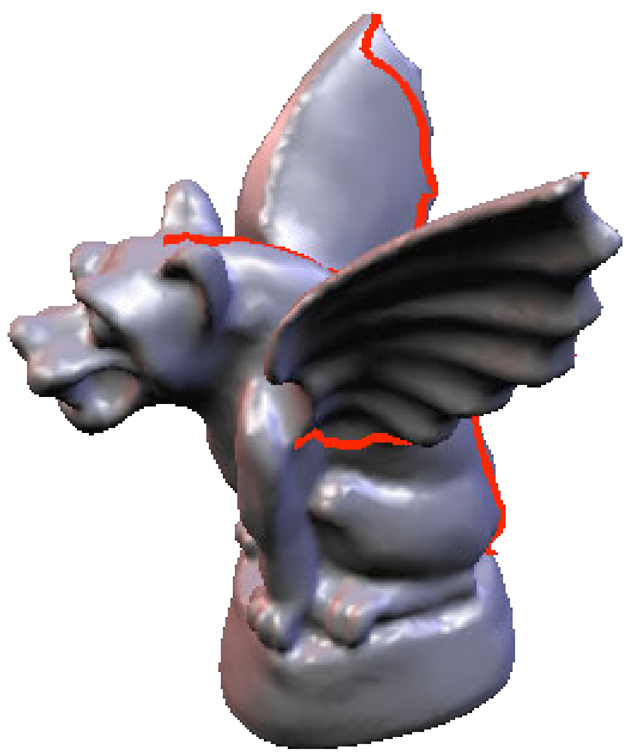
Stretch metric minimization

Using [Yoshizawa et. al 2004]

[Zayer et. al 2005]

Reducing Area Distortion

- Introduce cuts ⇨ area distortion *vs.* continuity
- Cuts are often unavoidable (e.g., open sphere)



Outline

- Motivation
- Objectives and Discrete Mappings
- Angle Preservation
  - Discrete Harmonic Maps
  - Discrete Conformal Maps
- Reducing Area Distortion
- Alternative Domains

Summary

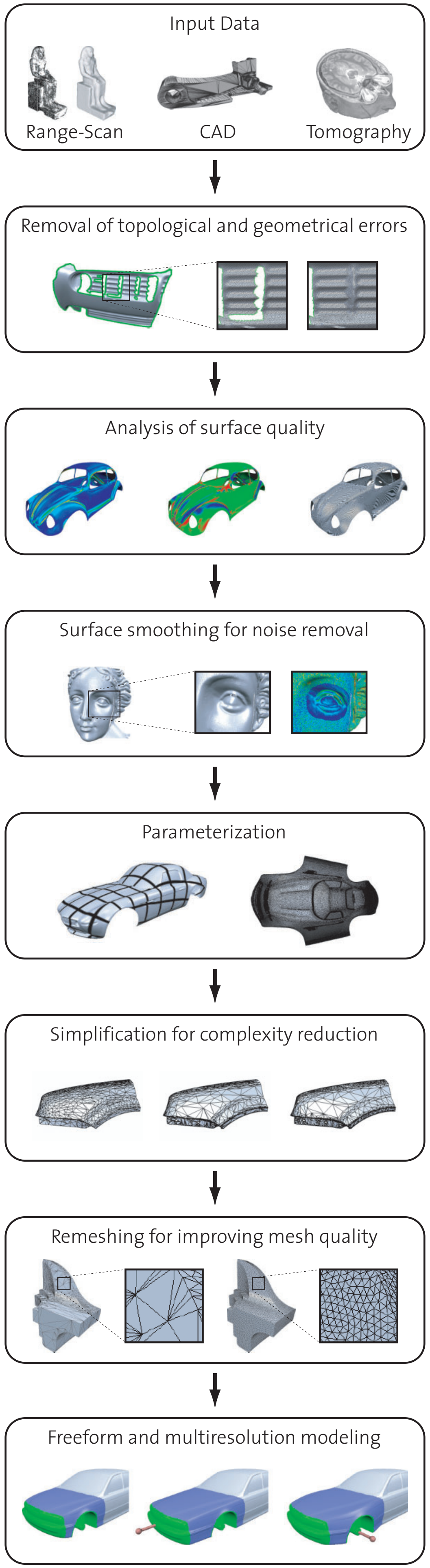
- Isometric mappings are rare
  - Angle preservation vs. area preservation
  - There is no perfect solution.
- Validity
- Boundary
- Linear / non-linear methods
- Domain

Literature

- Floater & Hormann: *Surface parameterization: a tutorial and survey,* Springer, 2005
- Lévy, Petitjean, Ray, and Maillot: *Least squares conformal maps for automatic texture atlas generation*, SIGGRAPH 2002
- Desbrun, Meyer, and Alliez: *Intrinsic parameterizations of surface meshes*, Eurographics 2002
- Sheffer & de Sturler: *Parameterization of faceted surfaces for meshing using angle based flattening*, Engineering with Computers, 2000.

Outline

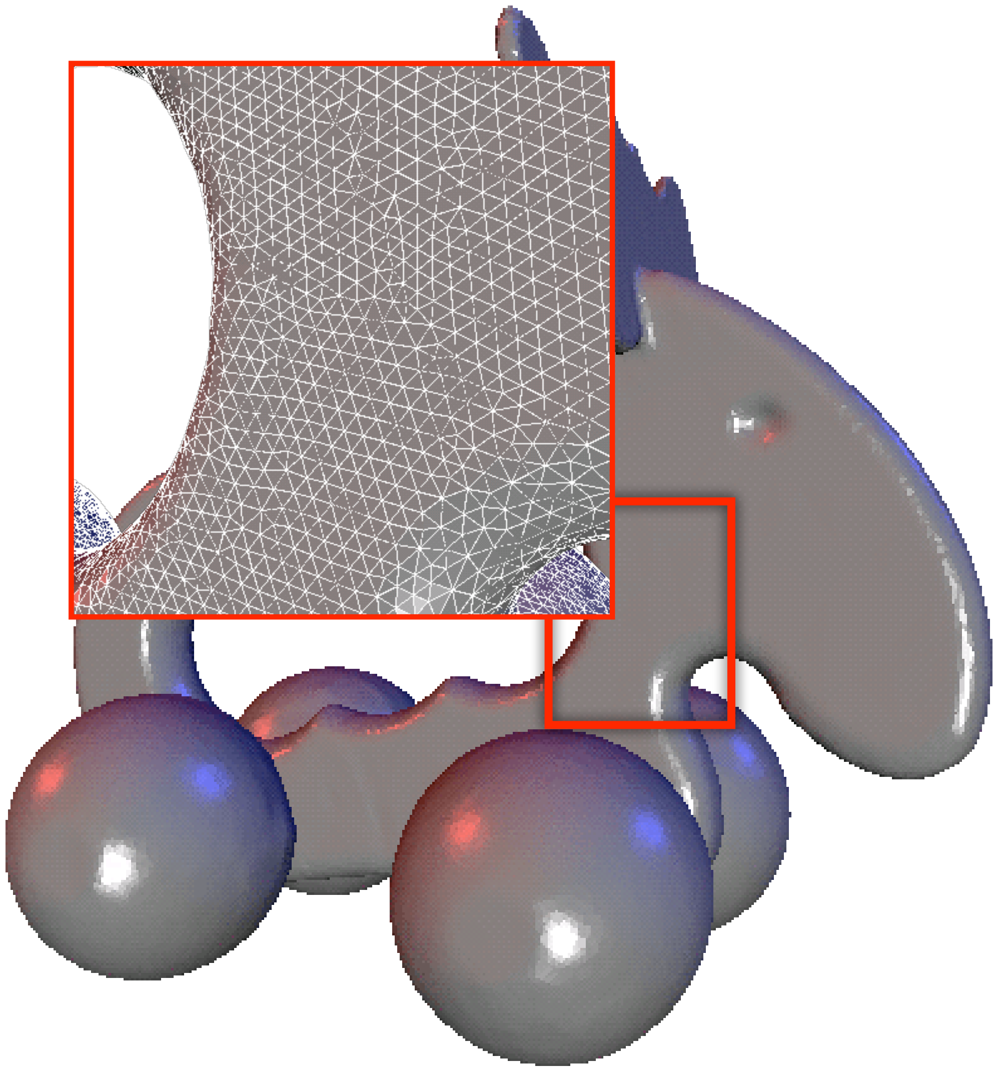
- Motivation
- Objectives and Discrete Mappings
- Angle Preservation
- Reducing Area Distortion
- Alternative Domains



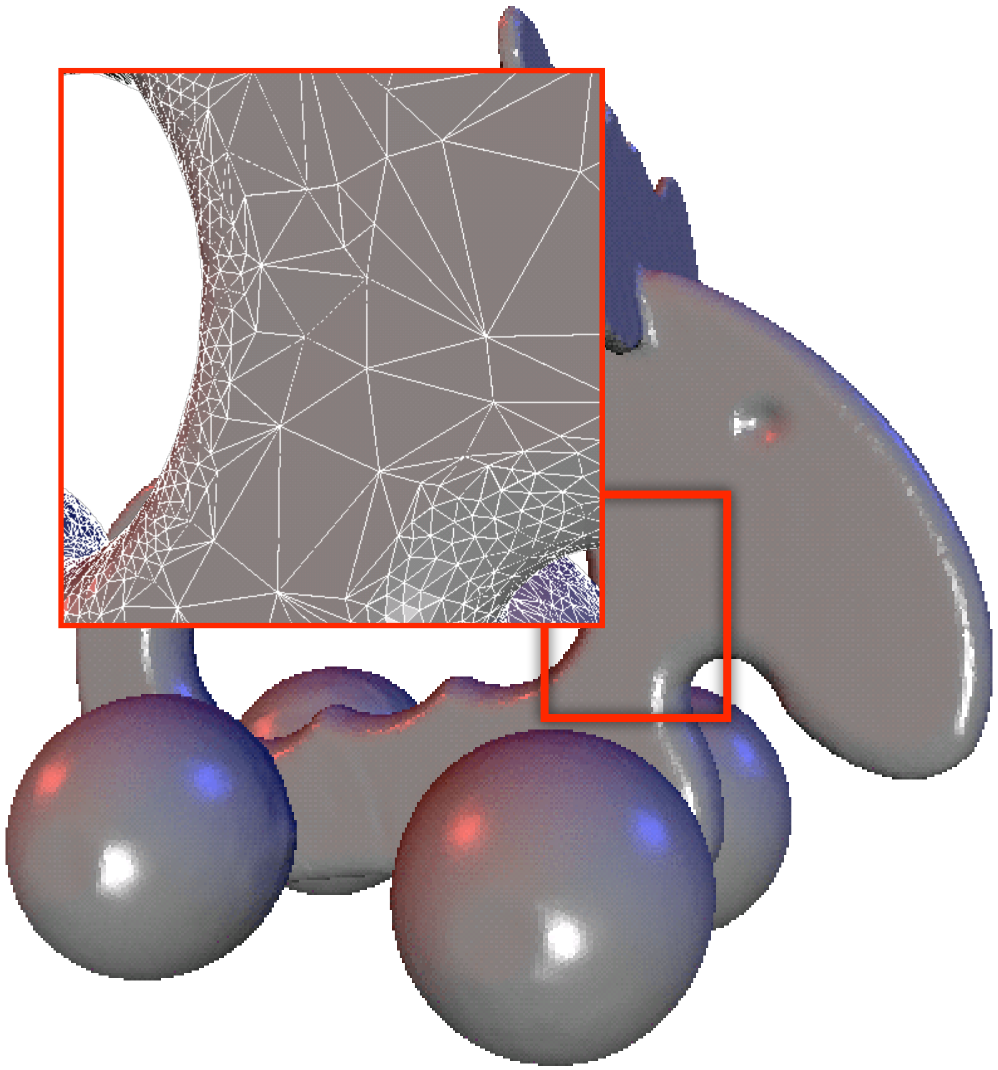
Mesh Decimation

Applications

- Oversampled 3D scan data



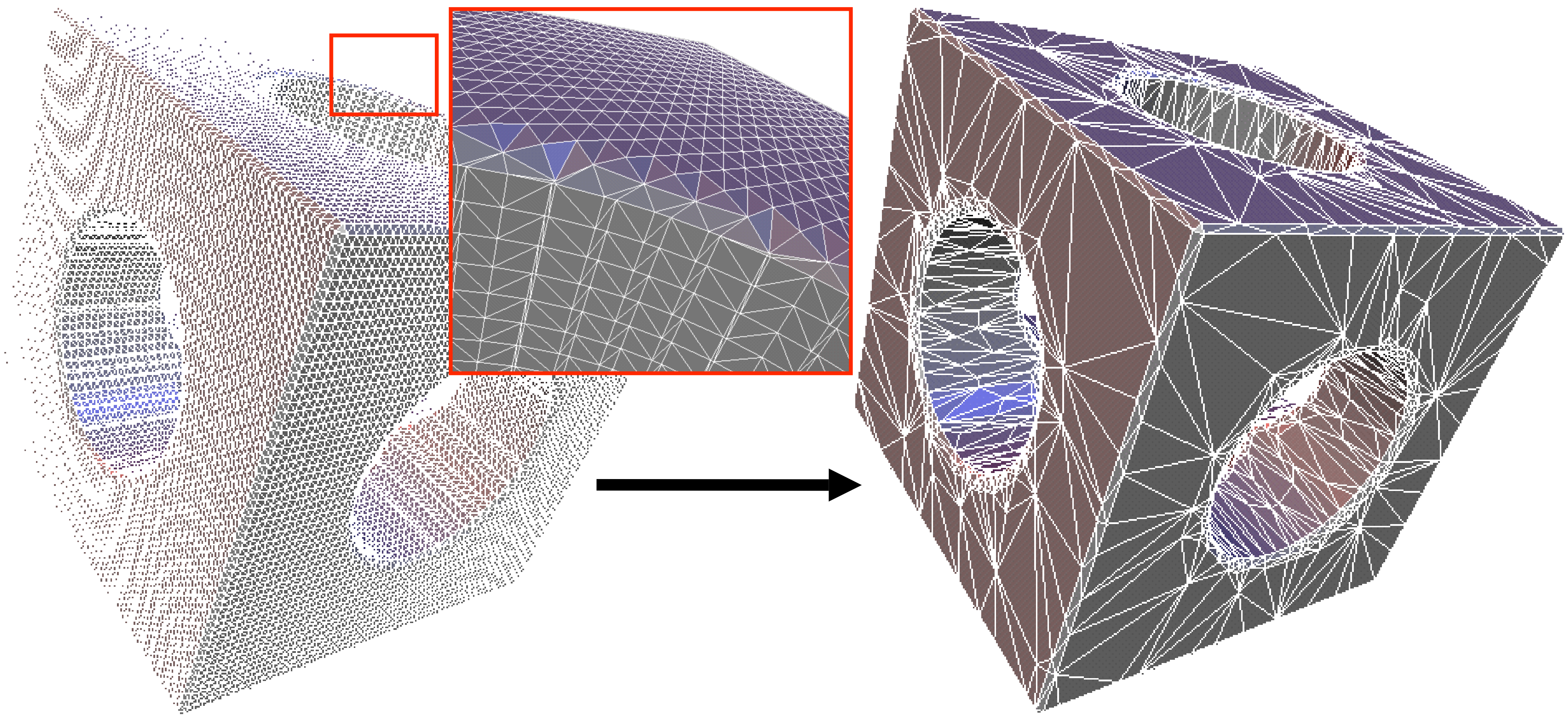
~150k triangles



~80k triangles

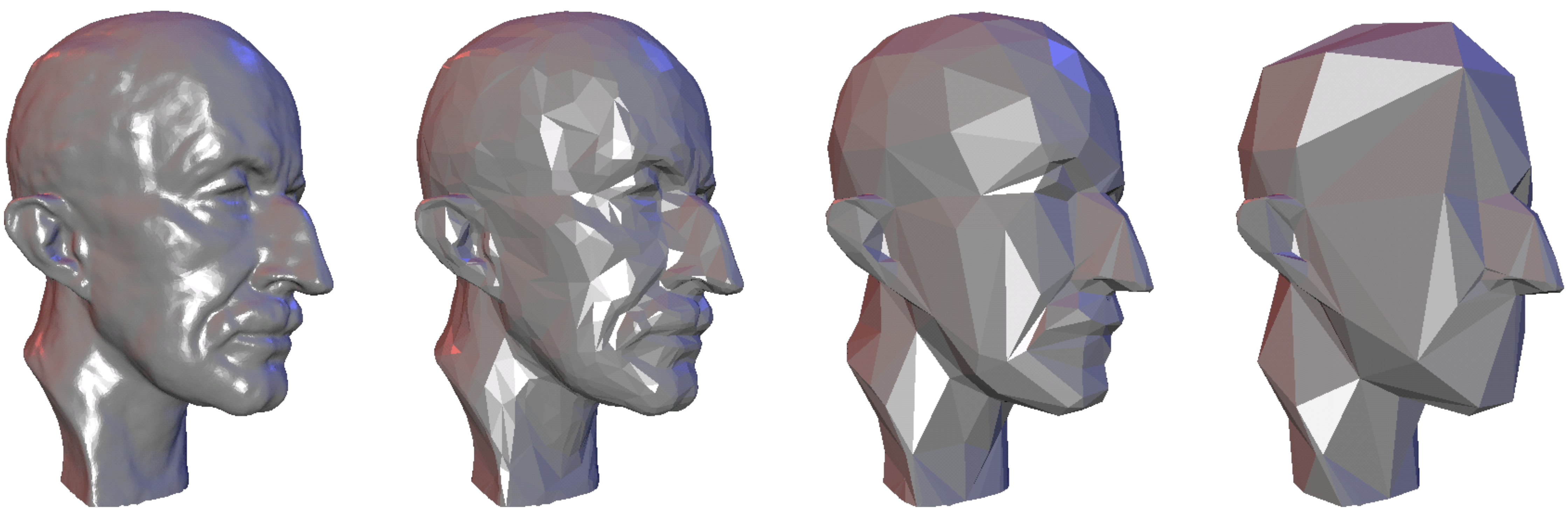
Applications

- Overtessellation: E.g. iso-surface extraction



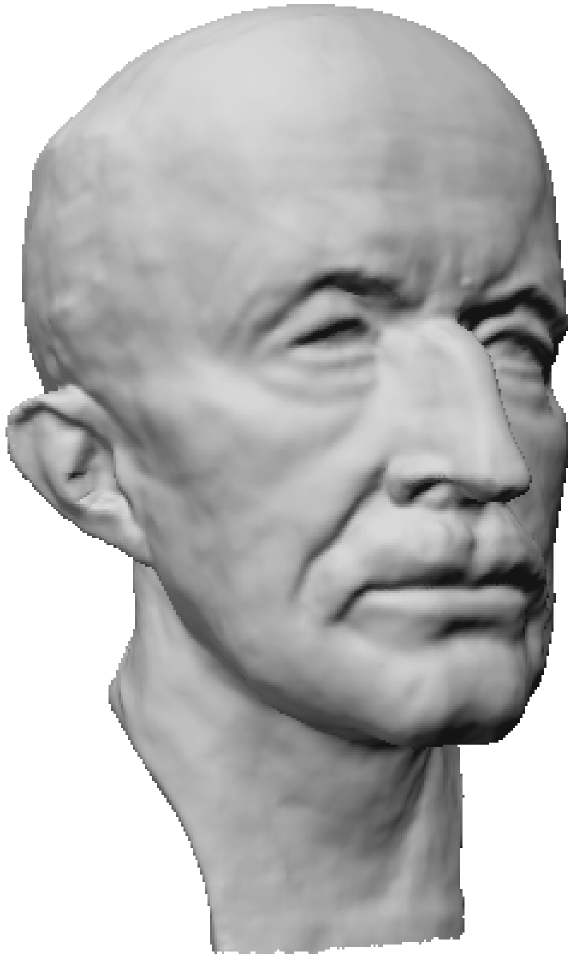
Applications

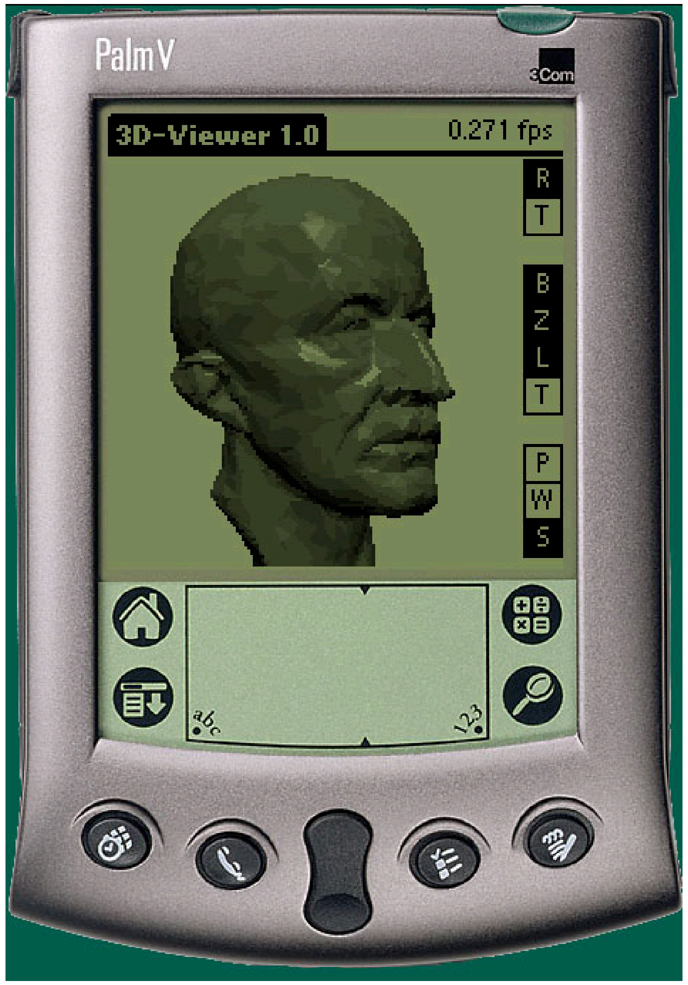
- Multi-resolution hierarchies for
  - efficient geometry processing
  - level-of-detail (LOD) rendering



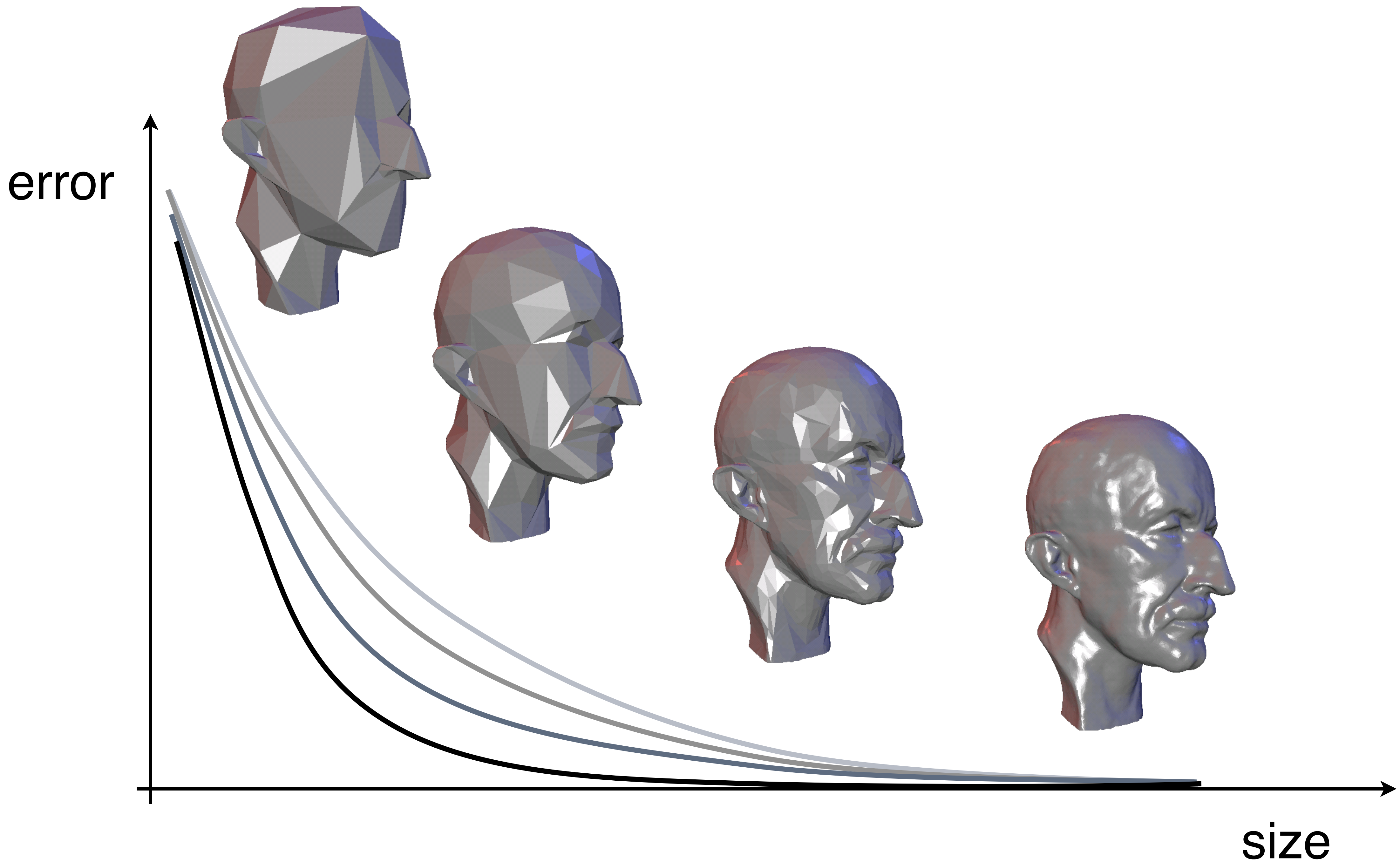
Applications

- Adaptation to hardware capabilities





Size-Quality Tradeoff



Outline

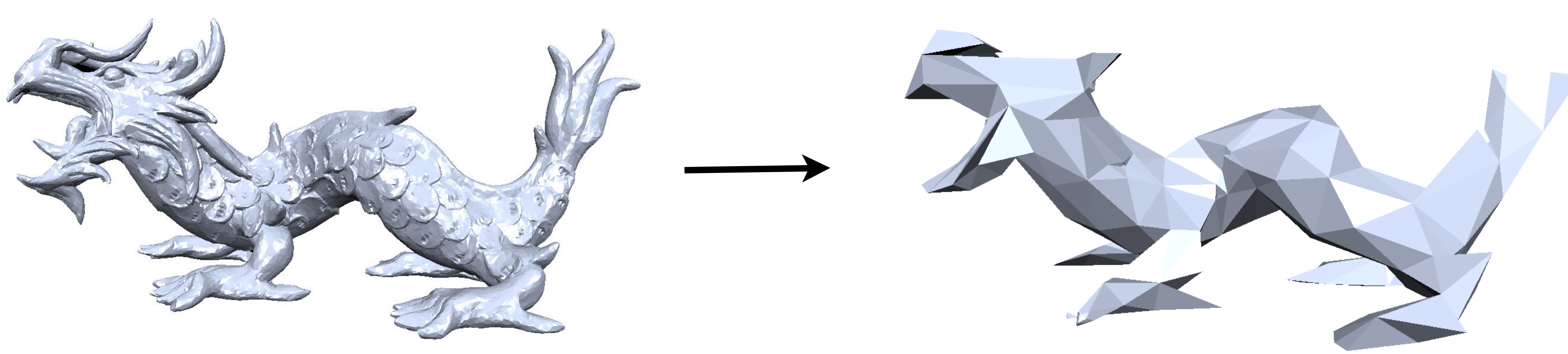
- applications
- problem statement
- mesh decimation schemes
  - vertex clustering
  - incremental decimation
  - out-of-core

Problem Statement

- Given: $\mathcal{M} = (\mathcal{V}, \mathcal{F})$
- Find: $\mathcal{M}' = (\mathcal{V}', \mathcal{F}')$ such that

  1. $|\mathcal{V}'| = n < |\mathcal{V}|$ and $\|\mathcal{M} - \mathcal{M}'\|$ is minimal, or

  2. $\|\mathcal{M} - \mathcal{M}'\| < \epsilon$ and $|\mathcal{V}'|$ is minimal



$\mathcal{M}$ $\qquad\qquad$ $\mathcal{M}'$

Problem Statement

- Given: $\mathcal{M} = (\mathcal{V}, \mathcal{F})$
- Find: $\mathcal{M}' = (\mathcal{V}', \mathcal{F}')$ such that

1. $|\mathcal{V}'| = n < |\mathcal{V}|$ and $\|\mathcal{M} - \mathcal{M}'\|$ is minimal, or
2. $\|\mathcal{M} - \mathcal{M}'\| < \epsilon$ and $|\mathcal{V}'|$ is minimal

combinatorial optimization is NP-hard!

→ find approximate-optimal solution

Problem Statement

- Given: $\mathcal{M} = (\mathcal{V}, \mathcal{F})$
- Find: $\mathcal{M}' = (\mathcal{V}', \mathcal{F}')$ such that

  1. $|\mathcal{V}'| = n < |\mathcal{V}|$ and $\|\mathcal{M} - \mathcal{M}'\|$ is minimal, or

  2. $\|\mathcal{M} - \mathcal{M}'\| < \epsilon$ and $|\mathcal{V}'|$ is minimal

- Take additional fairness criteria into account
  – normal deviation, triangle shape, color etc.

Outline

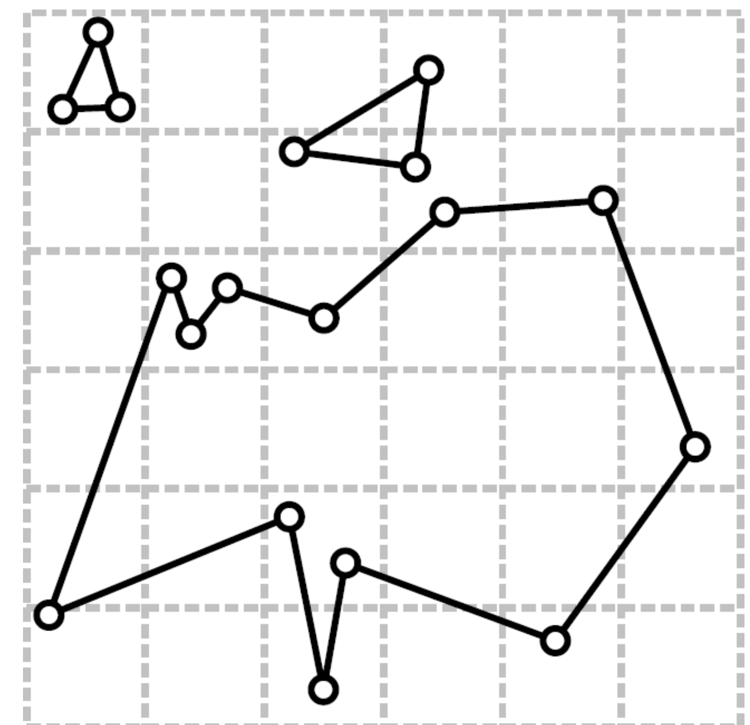
- applications
- problem statement
- mesh decimation schemes
  - **vertex clustering**
  - incremental decimation
  - out-of-core

Vertex Clustering

- cluster generation
- computing a representative
- mesh generation
- topology changes

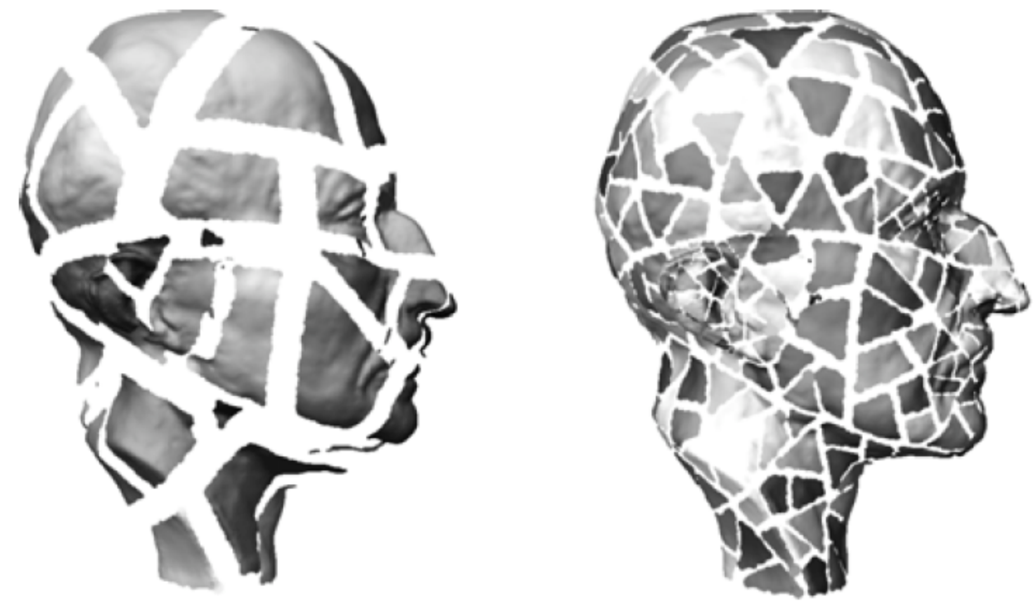
Vertex Clustering

- cluster generation
  - uniform 3D grid
  - map vertices to cluster cells
- computing a representative
- mesh generation
- topology changes



Vertex Clustering

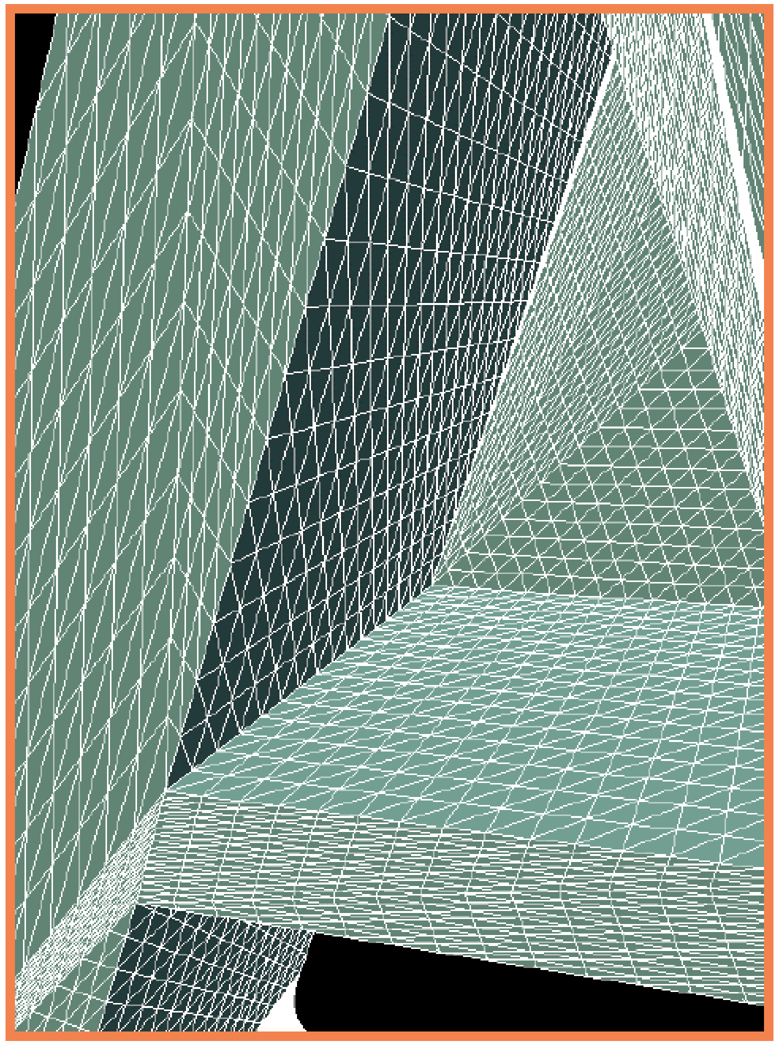
- cluster generation
  - hierarchical approach
  - top-down or bottom-up
- computing a representative
- mesh generation
- topology changes

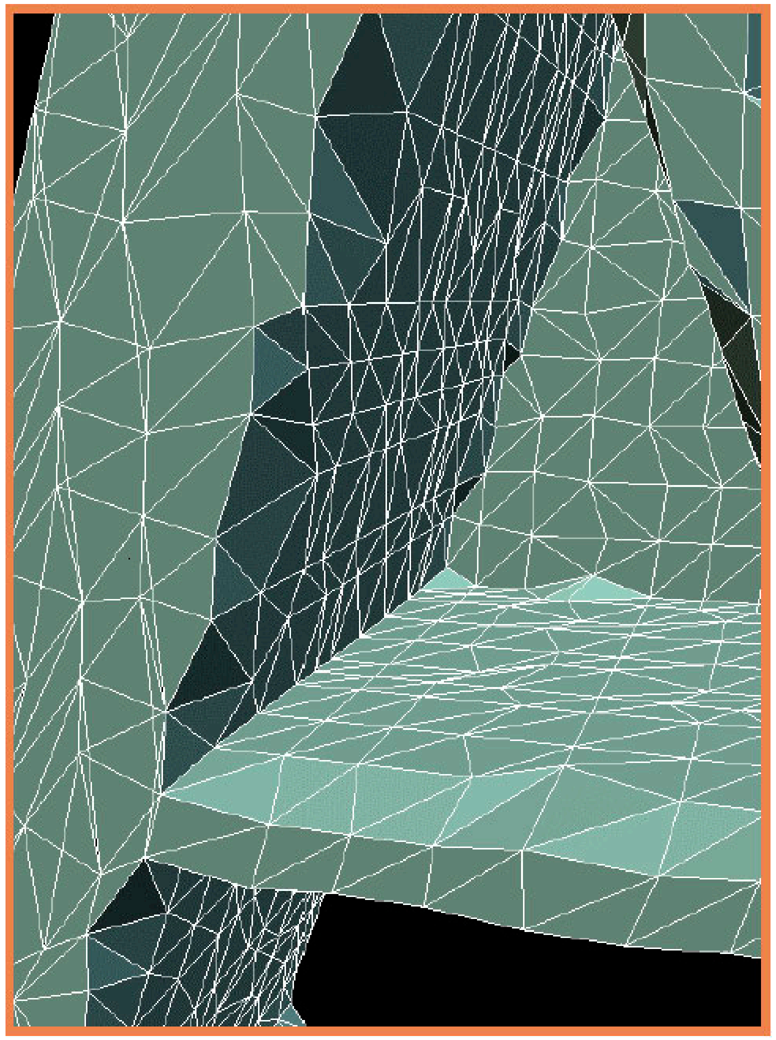


Vertex Clustering

- cluster generation
- computing a representative
  - average/median vertex position
  - error quadrics
- mesh generation
- topology changes

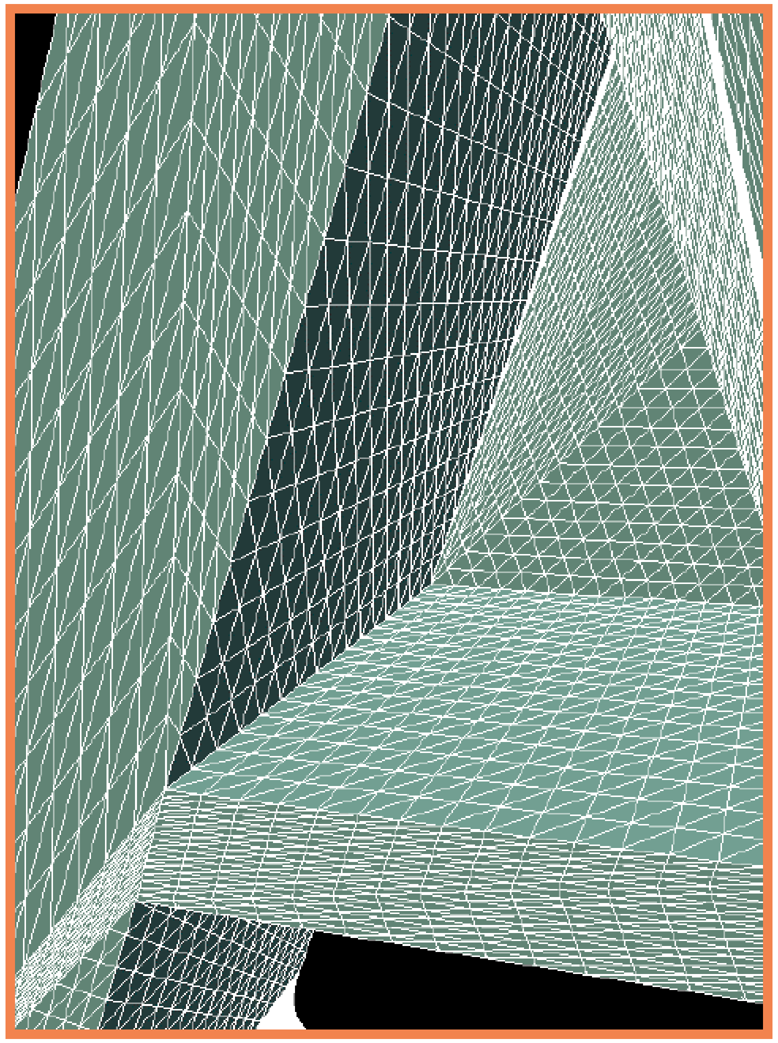
Computing a Representative

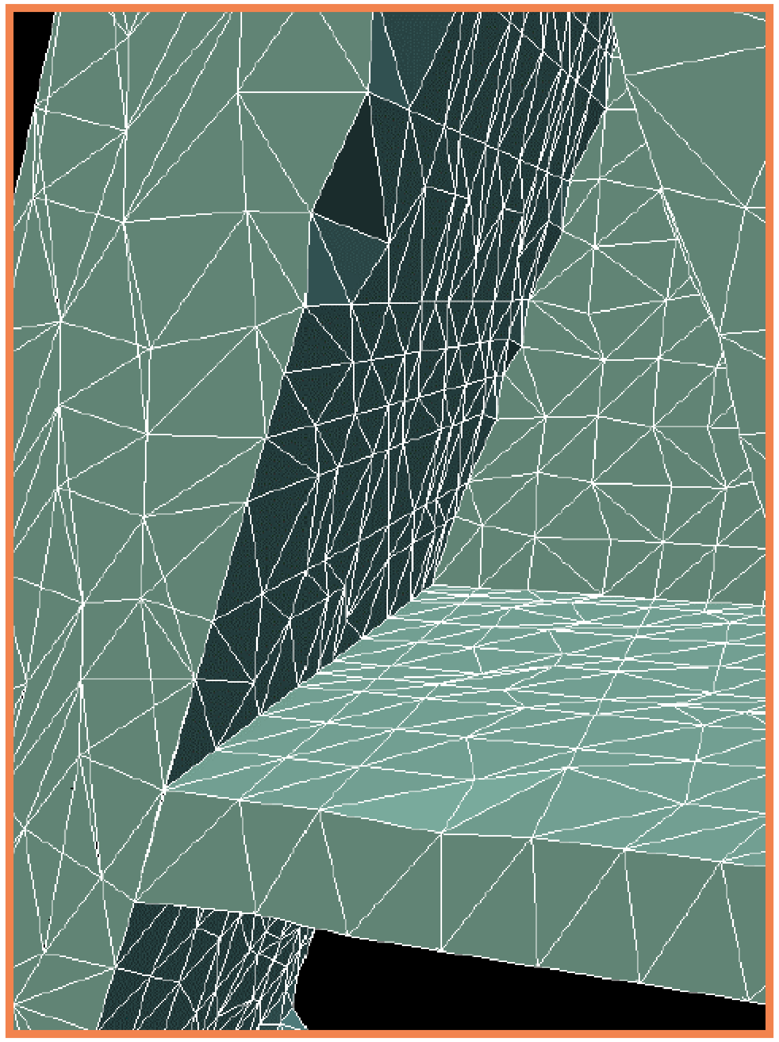




average vertex position → low-pass filter

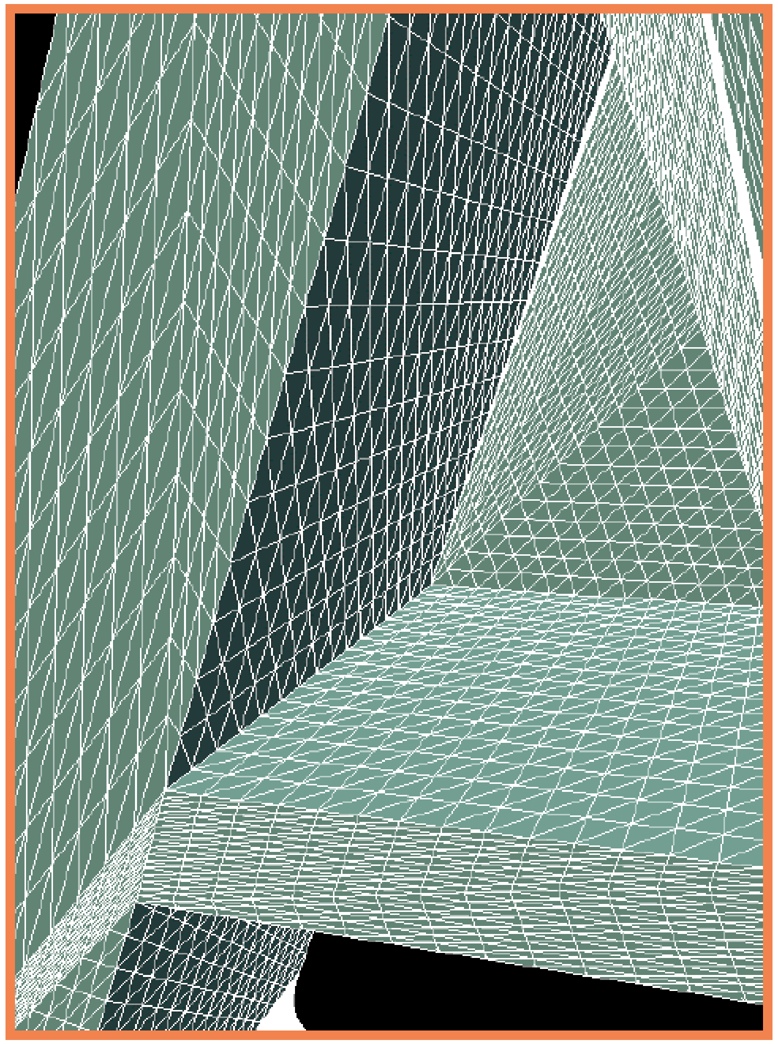
Computing a Representative

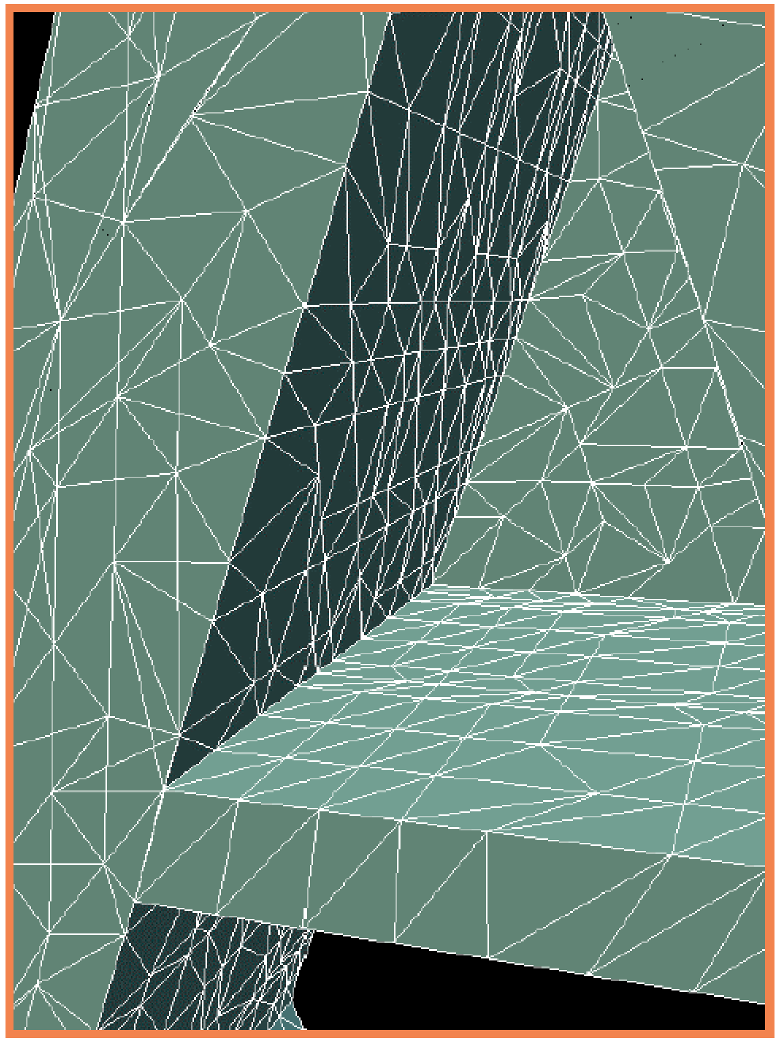




median vertex position → sub-sampling

Computing a Representative





error quadrics

Error Quadrics

- squared distance to plane

$$p = (x, y, z, 1)^T, \quad q = (a, b, c, d)^T$$

$$dist(q,p)^2 \;=\; (q^T p)^2 \;=\; p^T q\, q^T p \;=\; p^T Q_q\, p$$

$$Q_q = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & b^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

Error Quadrics

- sum of squared distances to triangle planes q_i

$$\sum_i dist(q_i, p)^2 = p^T \left(\sum_i Q_{q_i} \right) p$$

- point that minimizes the squared error

$$\begin{bmatrix} & \vdots & \\ a_i & b_i & c_i \\ & \vdots & \end{bmatrix} p* \approx \begin{bmatrix} \vdots \\ -d_i \\ \vdots \end{bmatrix}$$

Error Quadrics

- sum of squared distances to triangle planes q_i

$$\sum_i dist(q_i, p)^2 = p^T \left(\sum_i Q_{q_i} \right) p$$

- point that minimizes the squared error

$$\begin{bmatrix} \vdots \\ n_i^T \\ \vdots \end{bmatrix} p* \approx \begin{bmatrix} \vdots \\ -d_i \\ \vdots \end{bmatrix}$$

Error Quadrics

- sum of squared distances to triangle planes q_i

$$\sum_i dist(q_i, p)^2 \;=\; p^T \left(\sum_i Q_{q_i} \right) p$$

- point that minimizes the squared error

$$\left(\sum_i n_i \, n_i^T \right) p* \;=\; -\sum_i n_i \, d_i$$

Error Quadrics

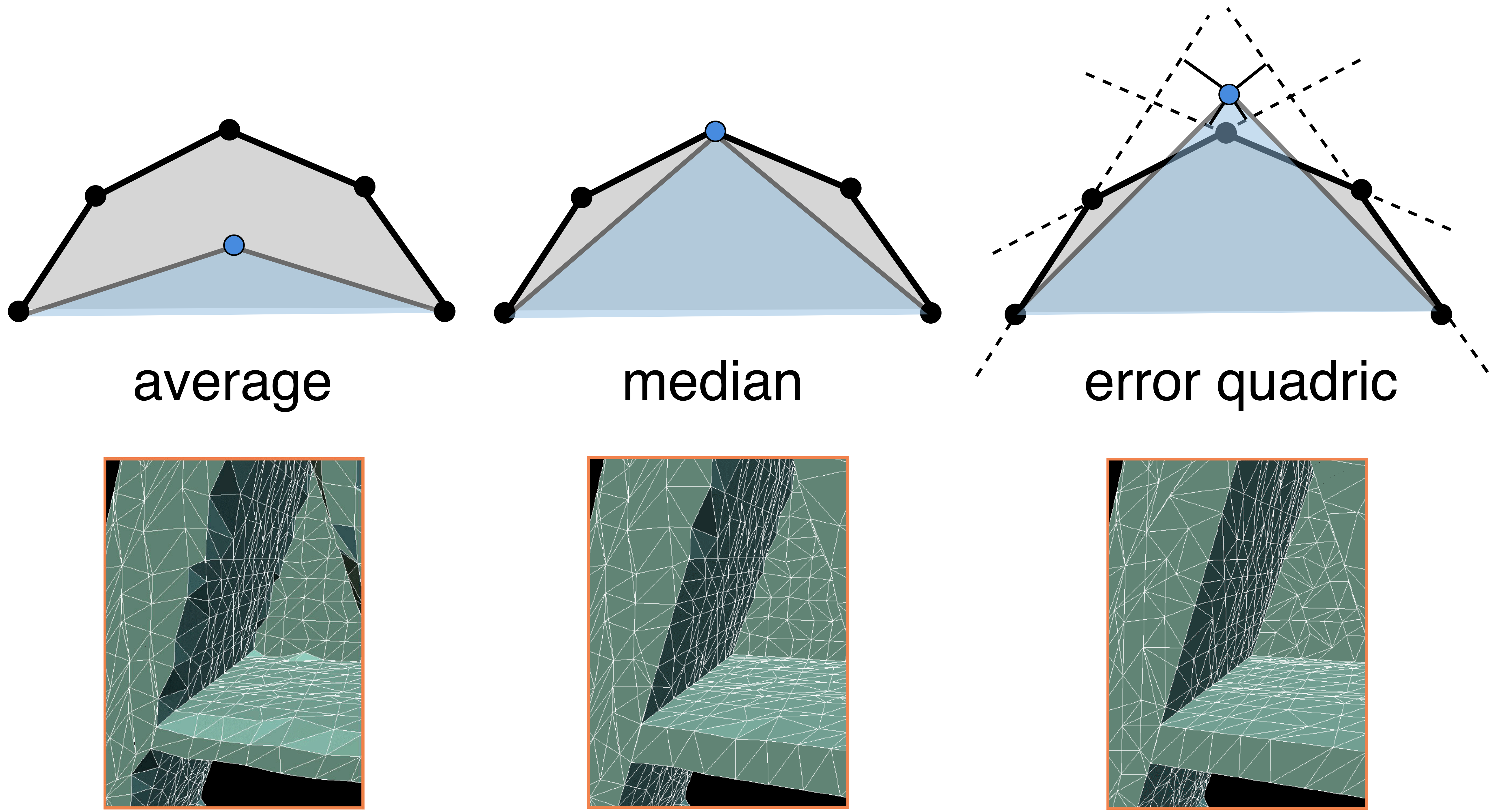
- sum of squared distances to triangle planes q_i

$$\sum_i dist(q_i, p)^2 = p^T \left(\sum_i Q_{q_i} \right) p$$

- point that minimizes the squared error

$$Q = \begin{pmatrix} A & b \\ b^T & c \end{pmatrix} \qquad A p^* = -b$$

Comparison

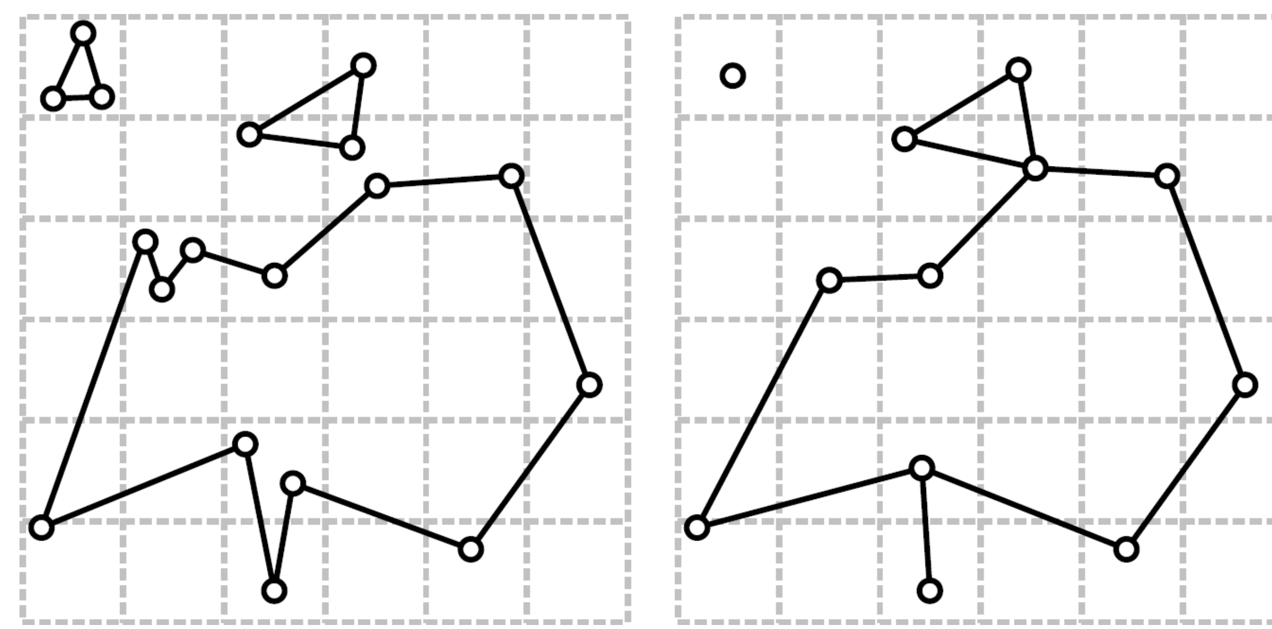


Vertex Clustering

- cluster generation
- computing a representative
- mesh generation
  - clusters $p \Leftrightarrow \{p_0,\ldots,p_n\}$, $q \Leftrightarrow \{q_0,\ldots,q_m\}$
  - connect (p,q) if there was an edge (p_i,q_j)
- topology changes

Vertex Clustering

- cluster generation
- computing a representative
- mesh generation
- topology changes
  - different sheets may pass through one cell
  - not manifold



Outline

- applications
- problem statement
- mesh decimation schemes
  - vertex clustering
  - **incremental decimation**
  - out-of-core

Incremental Decimation

- general setup
- decimation operators
- error metrics
- fairness criteria
- topology changes

General Setup

```
Repeat:
  pick mesh region
  apply decimation operator
Until no further reduction possible
```

Greedy Optimization

```
For each region
  evaluate quality after decimation
  enqueue(quality, region)

Repeat:
  pick best mesh region
  apply decimation operator
  update queue
Until no further reduction possible
```

Global Error Control

```
For each region
  evaluate quality after decimation
  enqeue(quality, region)

Repeat:
  pick best mesh region
  if error < ϵ
     apply decimation operator
     update queue
Until no further reduction possible
```

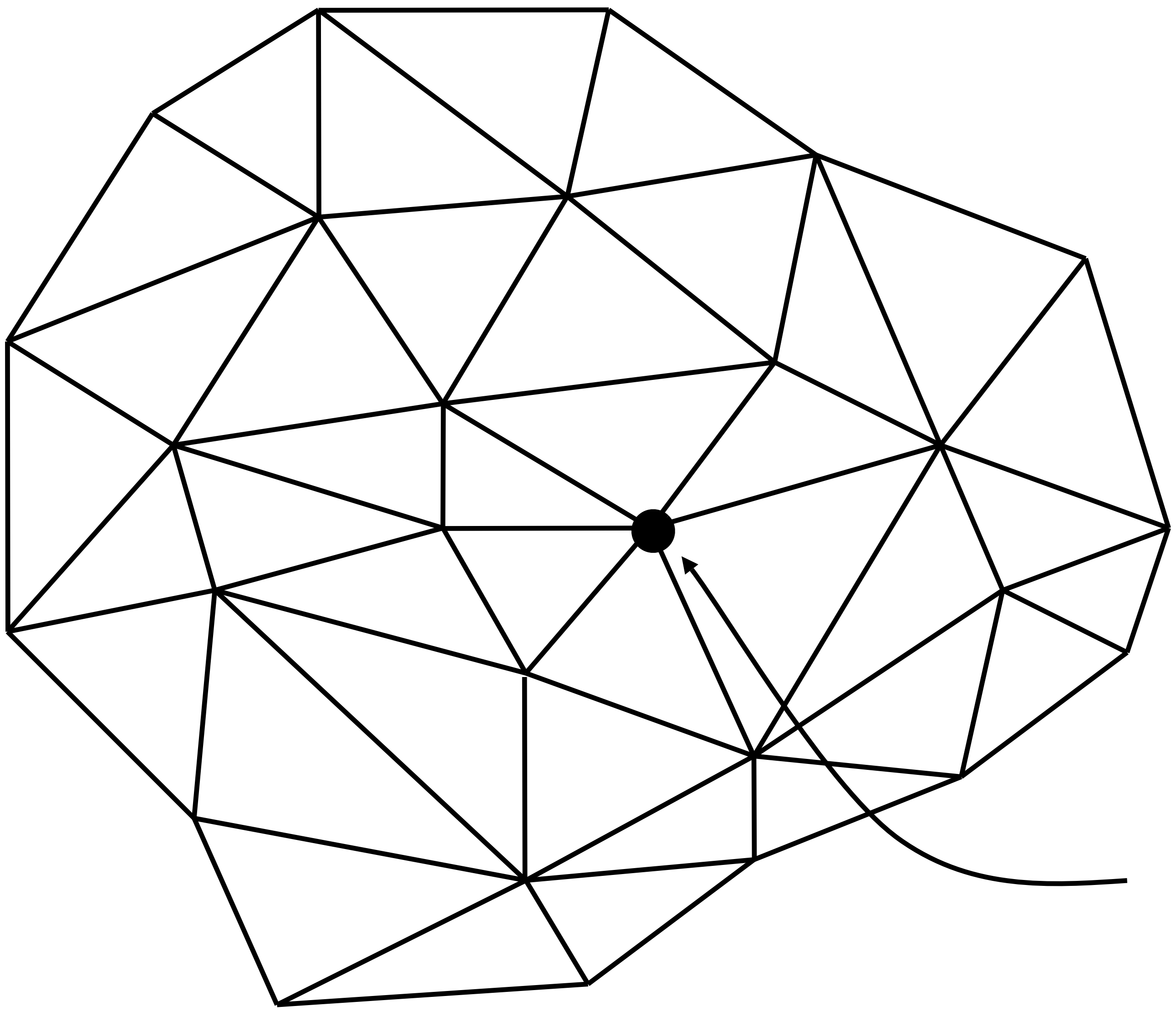
Incremental Decimation

- general setup
- decimation operators
- error metrics
- fairness criteria
- topology changes

Decimation Operators

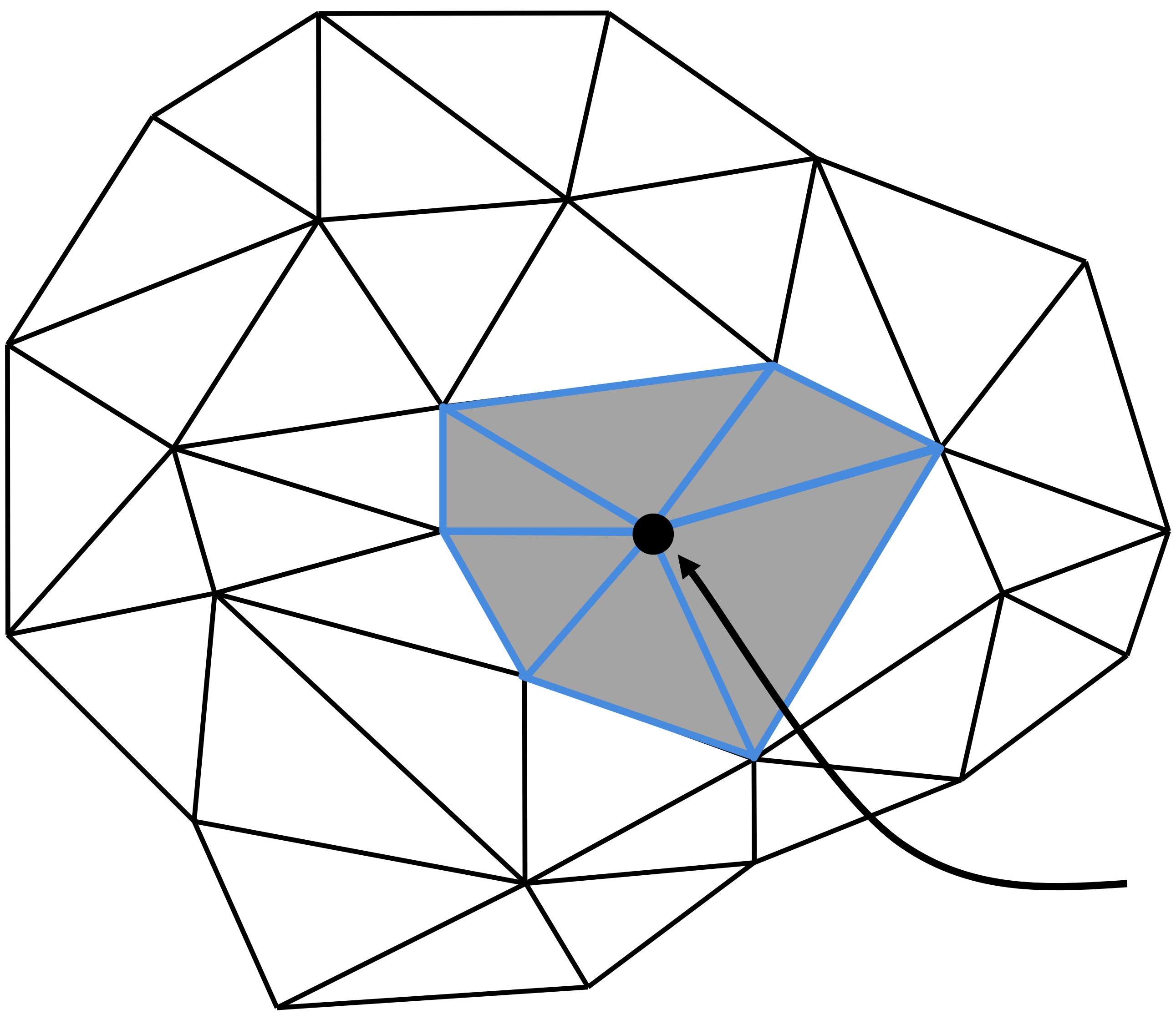
- what is a "region" ?
- what are the DOF for re-triangulation?
- classification
  - topology-changing vs. topology-preserving
  - subsampling vs. filtering
  - inverse operation → progressive meshes

Vertex Removal



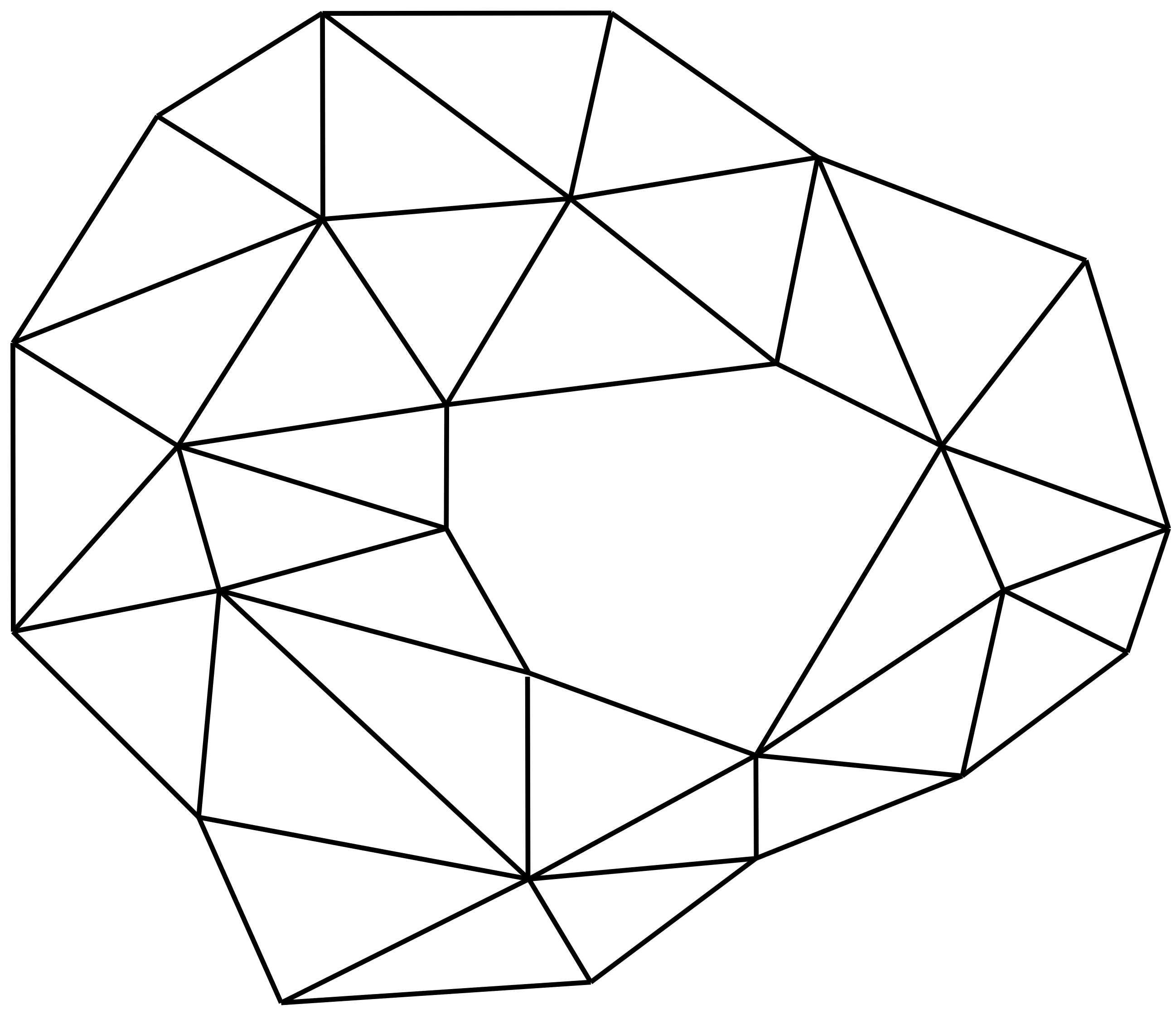
Select a vertex to be eliminated

Vertex Removal



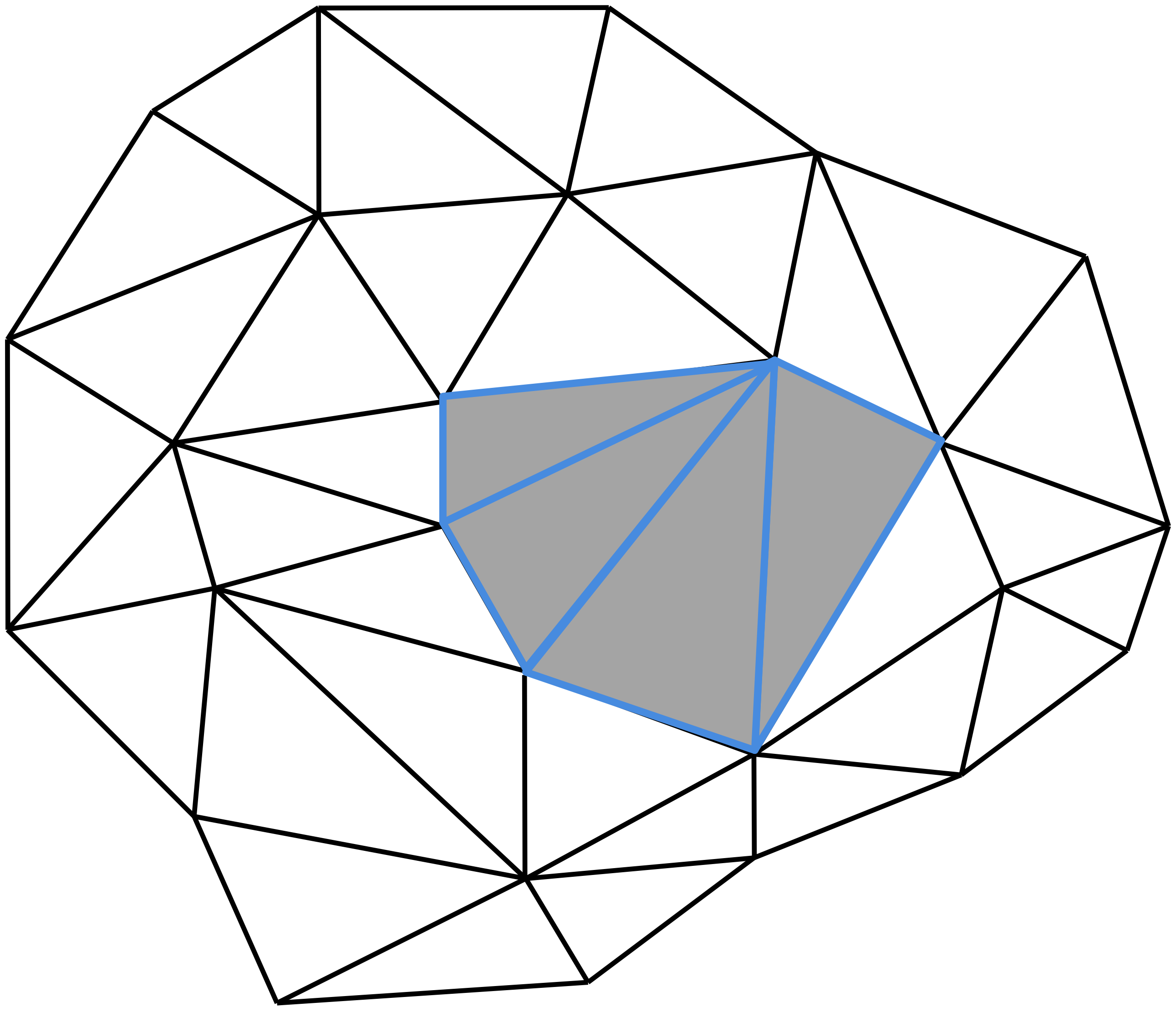
Select all triangles adjacent to this vertex

Vertex Removal



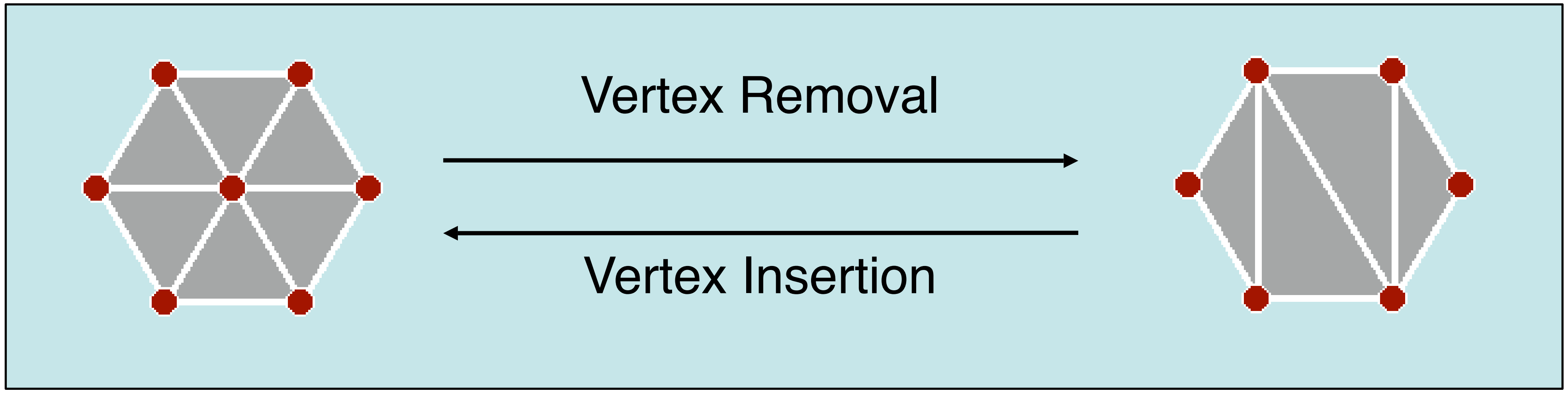
Remove the n
selected triangles,
creating the hole

Vertex Removal



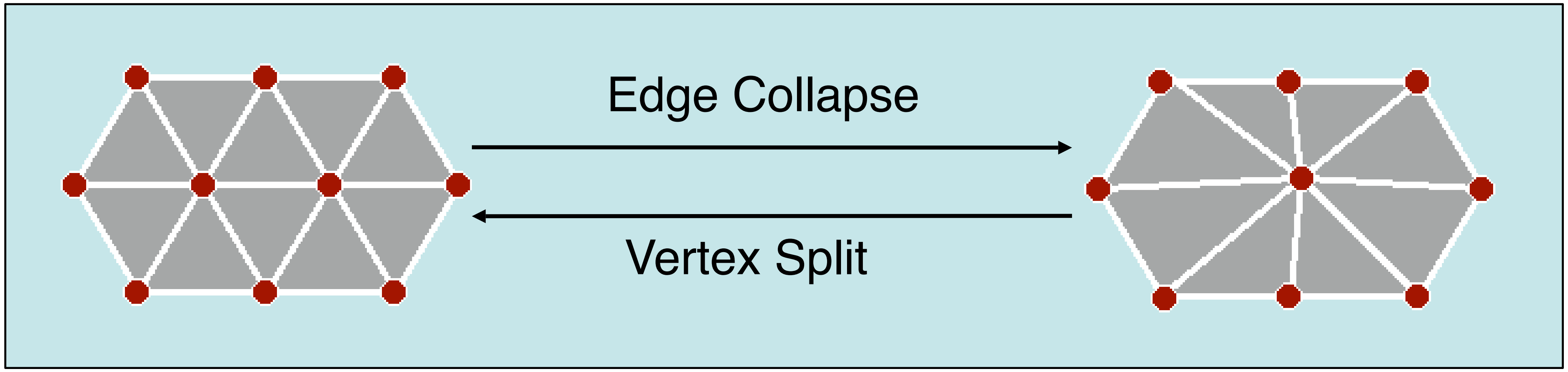
Fill the hole with
(n-2) triangles

Decimation Operators



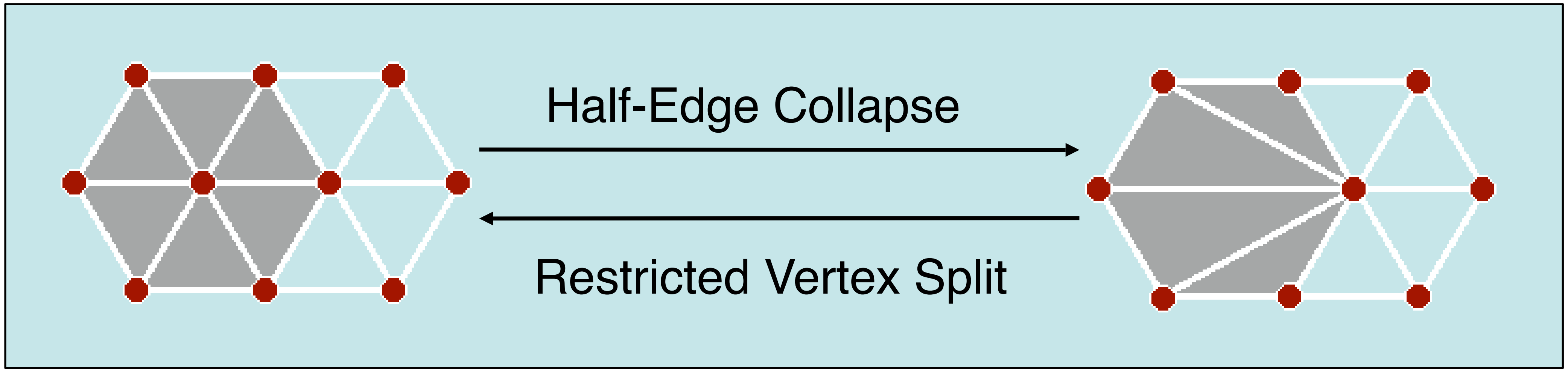
- remove vertex
- re-triangulate hole
  - combinatorial DOFs
  - sub-sampling

Decimation Operators



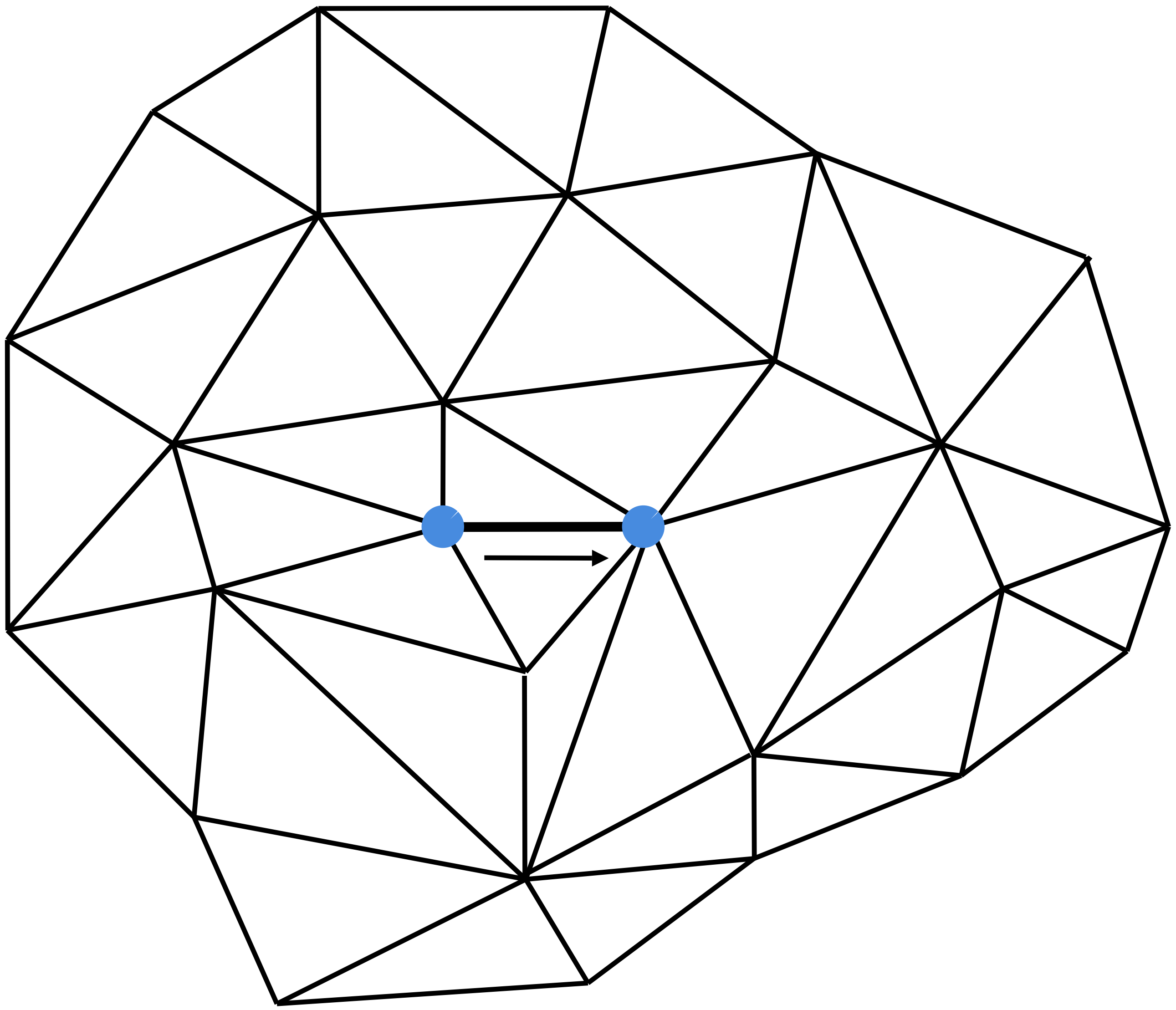
- merge two adjacent triangles
- define new vertex position
  - – continuous DOF
  - – filtering

Decimation Operators

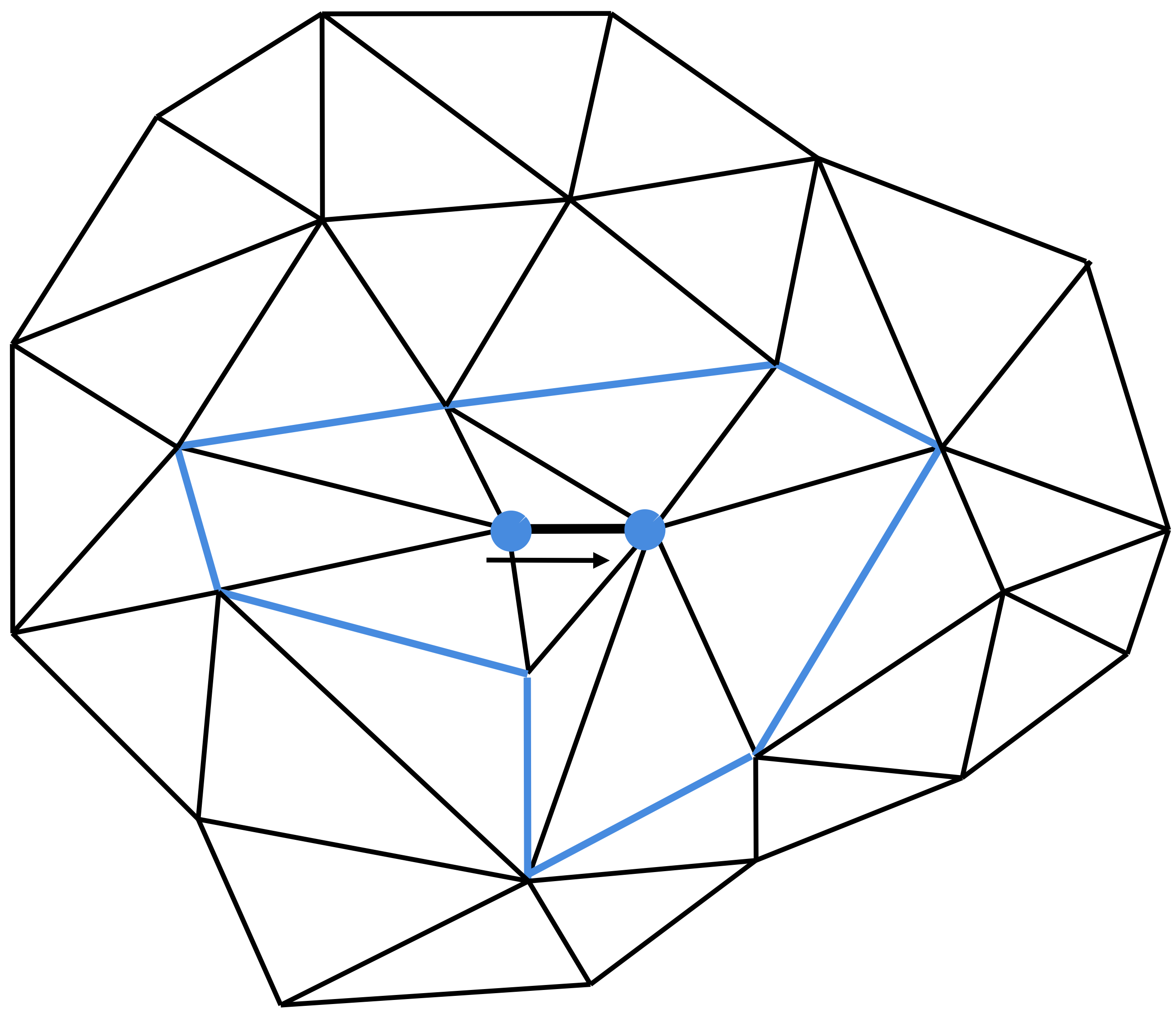


- collapse edge into one of its end points
  - special case of vertex removal
  - special case of edge collapse
- no DOFs
  - one operator per half-edge
  - sub-sampling!

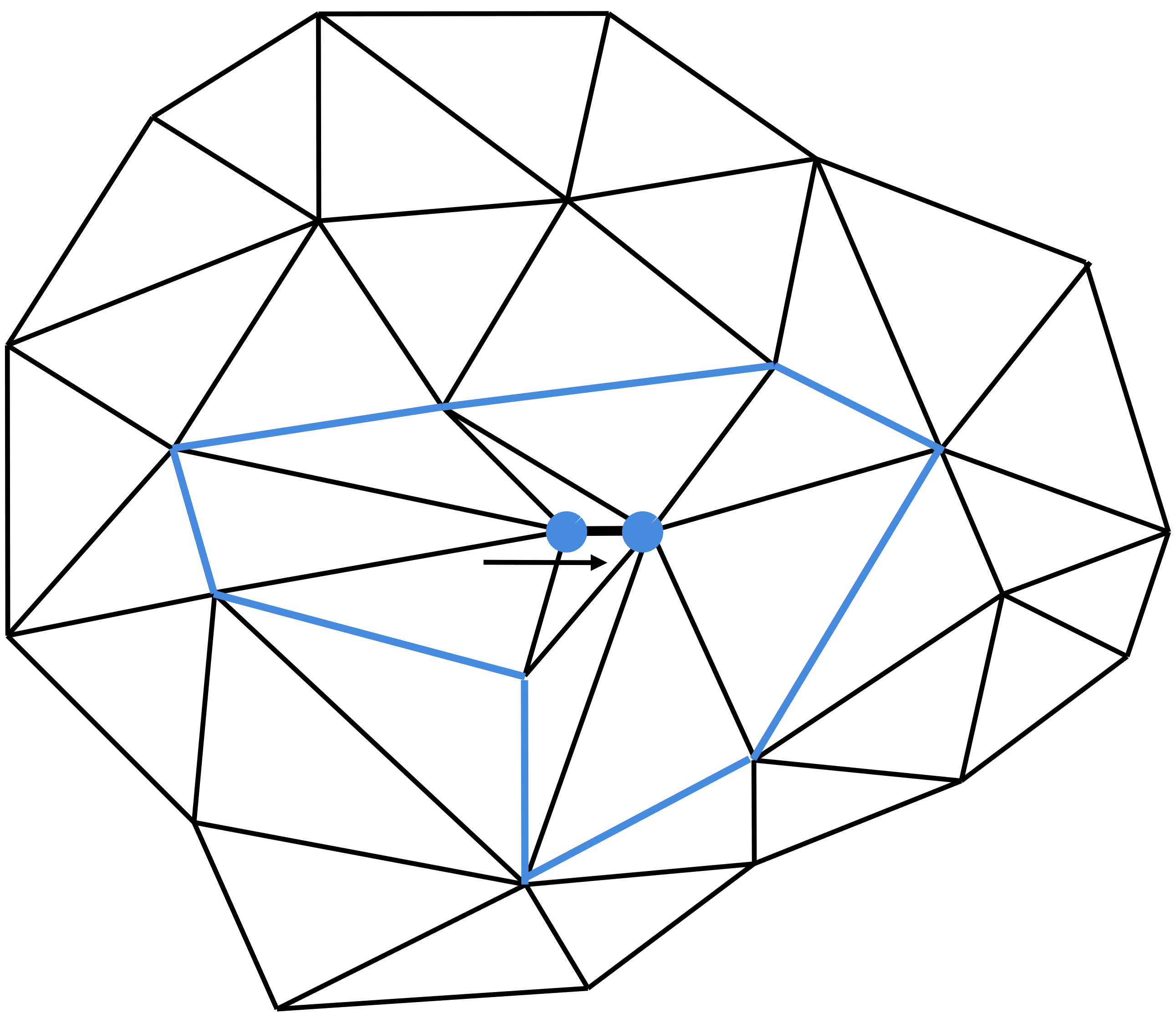
Half-Edge Collapse



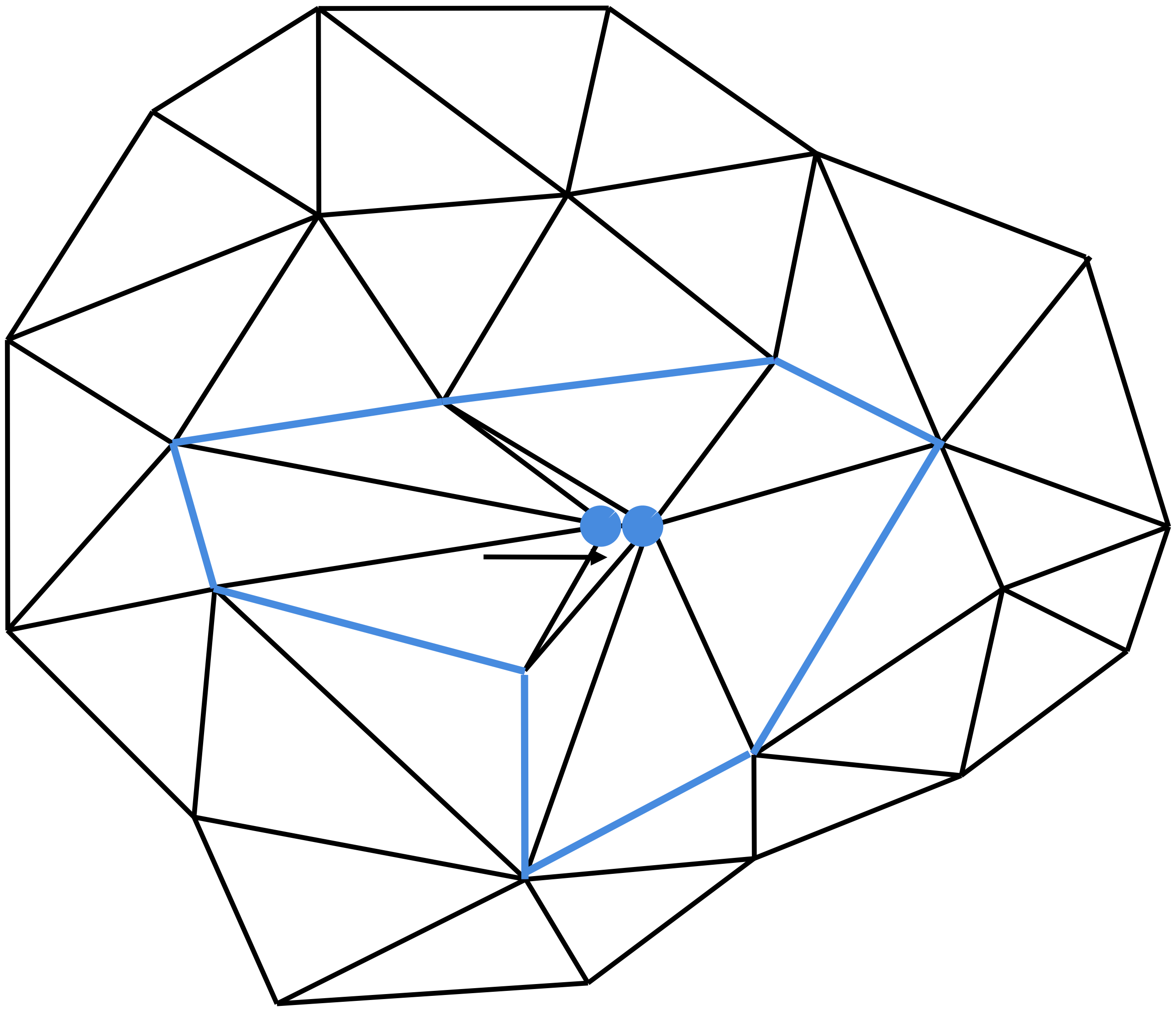
Half-Edge Collapse



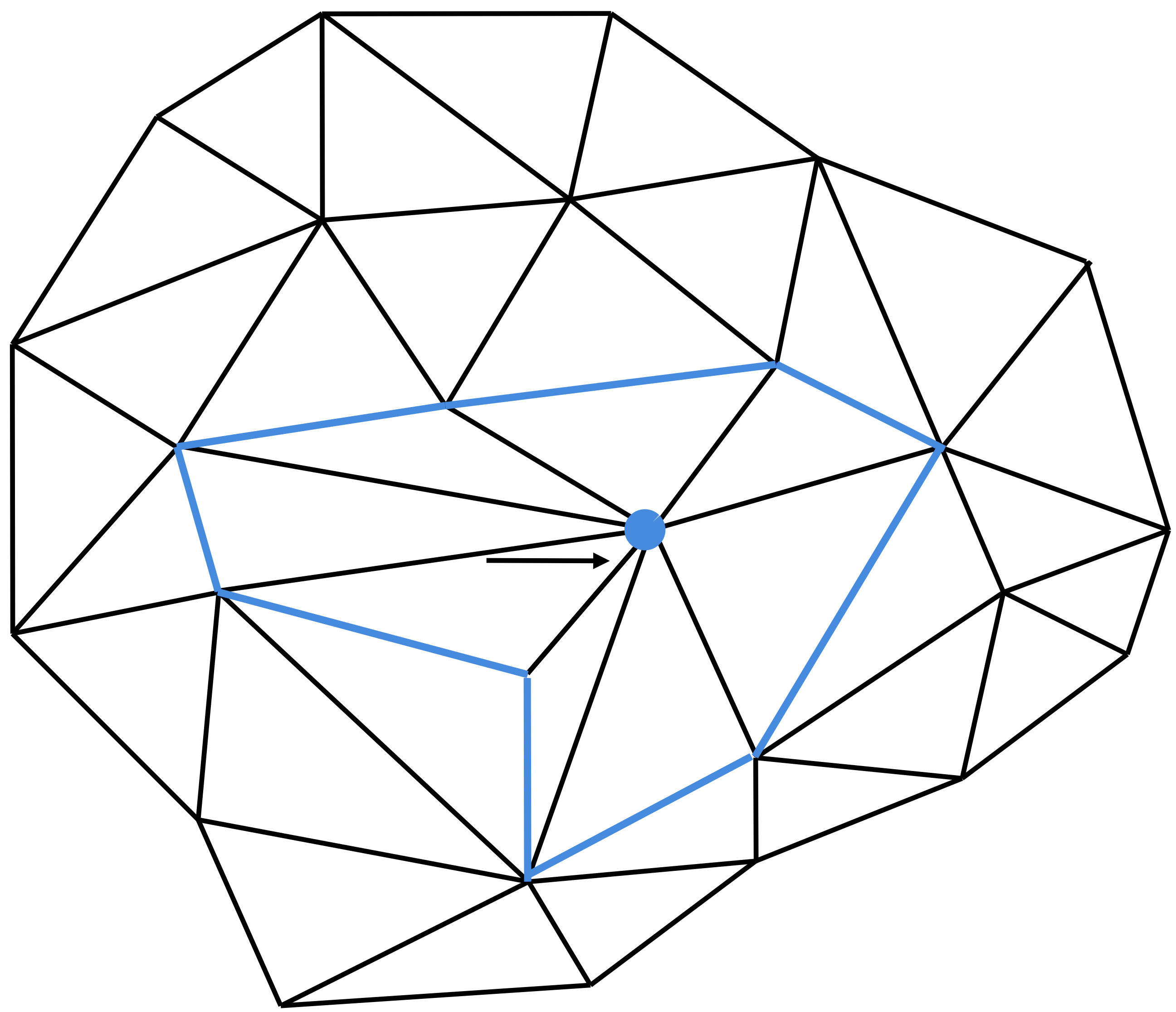
Half-Edge Collapse



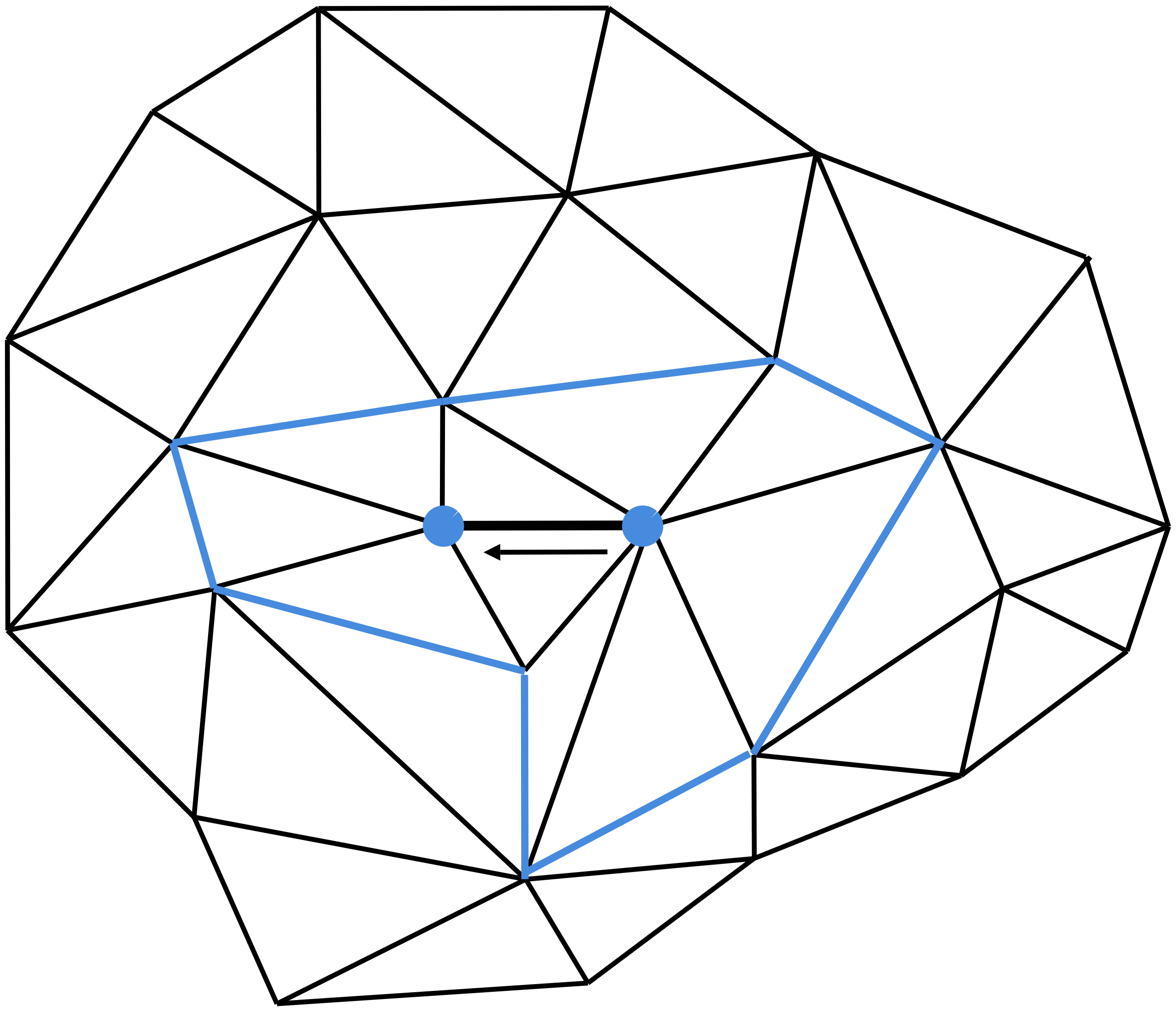
Half-Edge Collapse



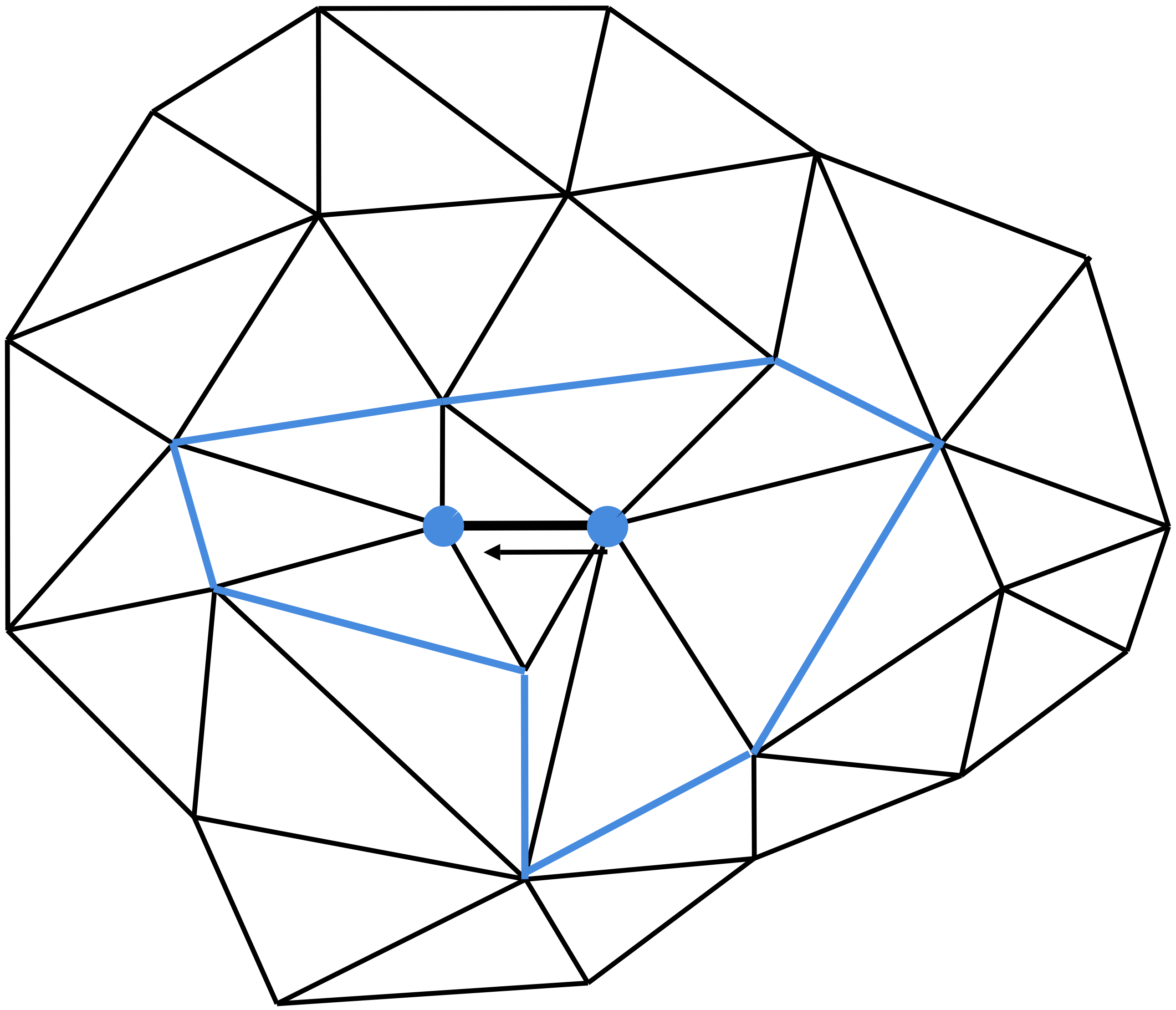
Half-Edge Collapse



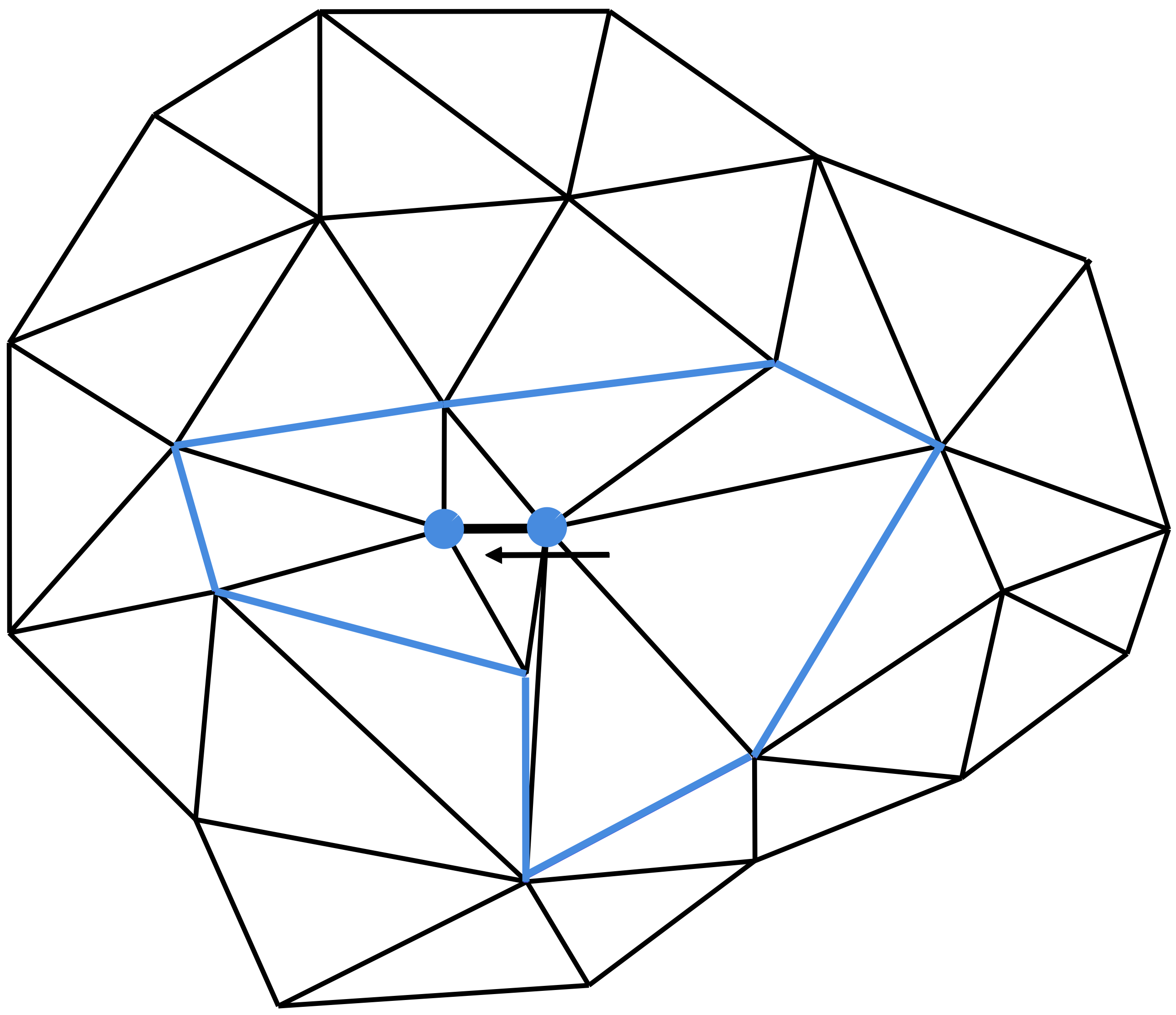
Half-Edge Collapse



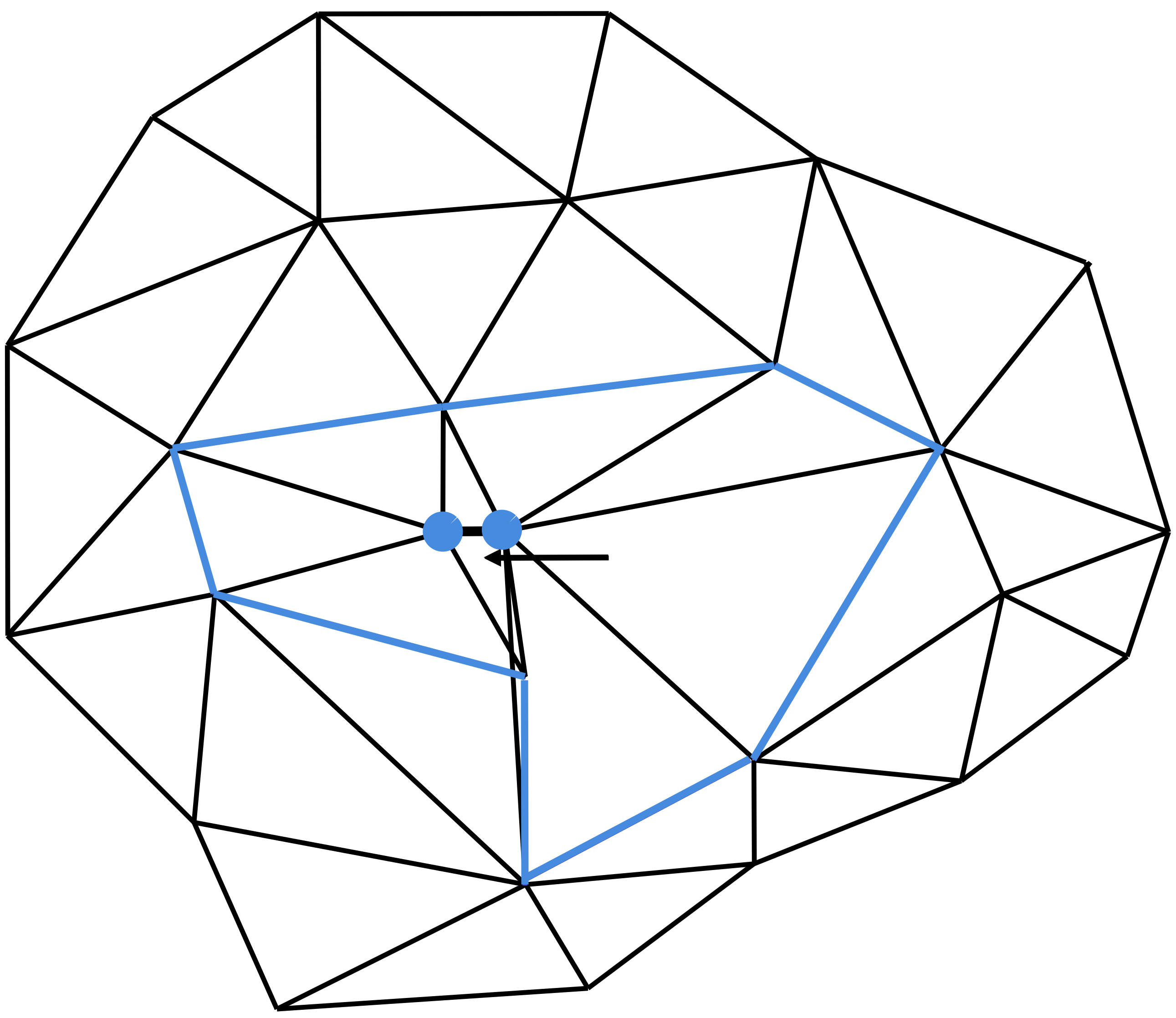
Half-Edge Collapse



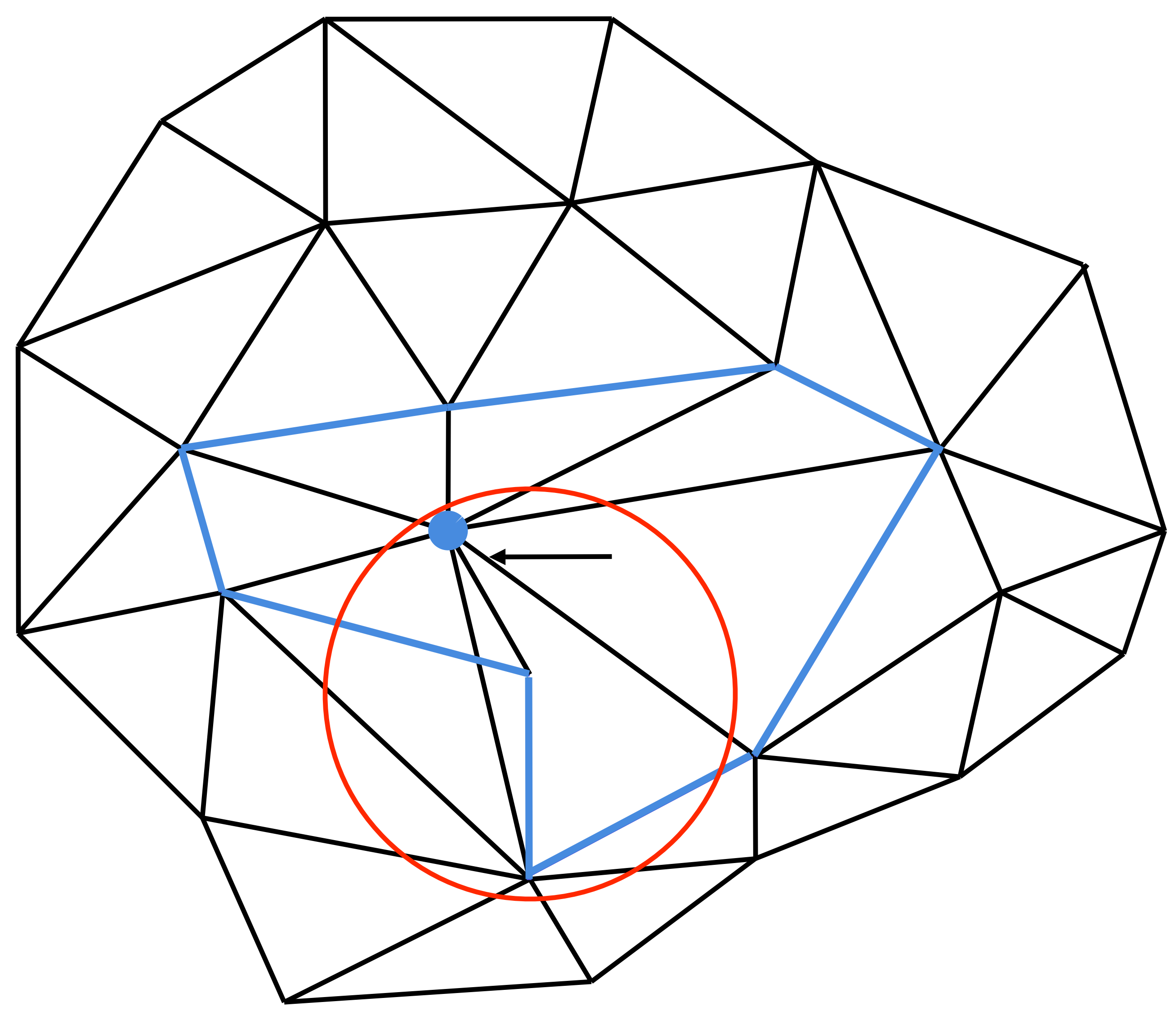
Half-Edge Collapse



Half-Edge Collapse



Half-Edge Collapse

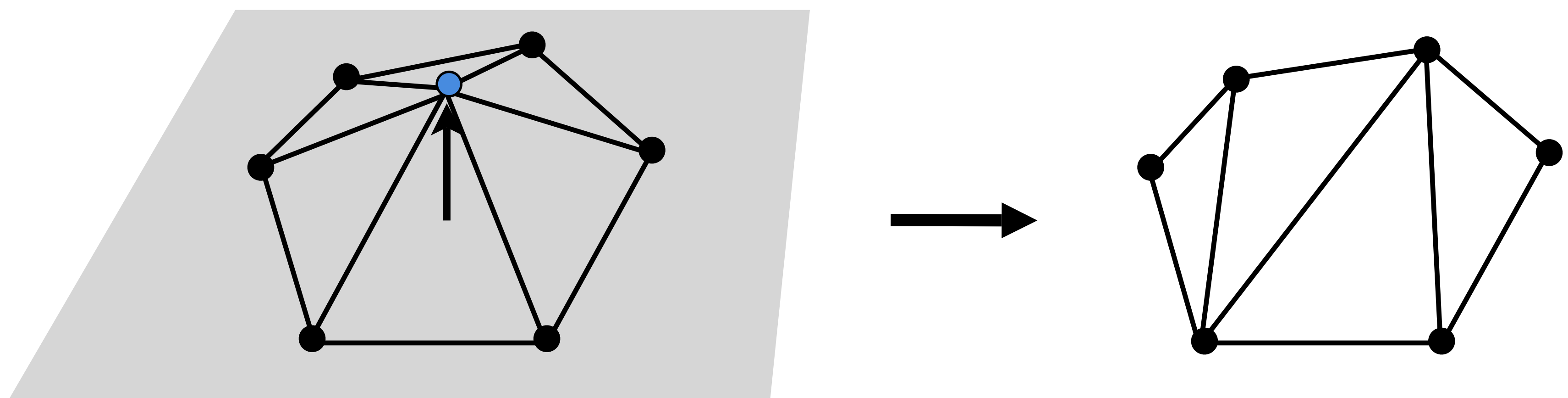


Incremental Decimation

- general setup
- decimation operators
- error metrics
- fairness criteria
- topology changes

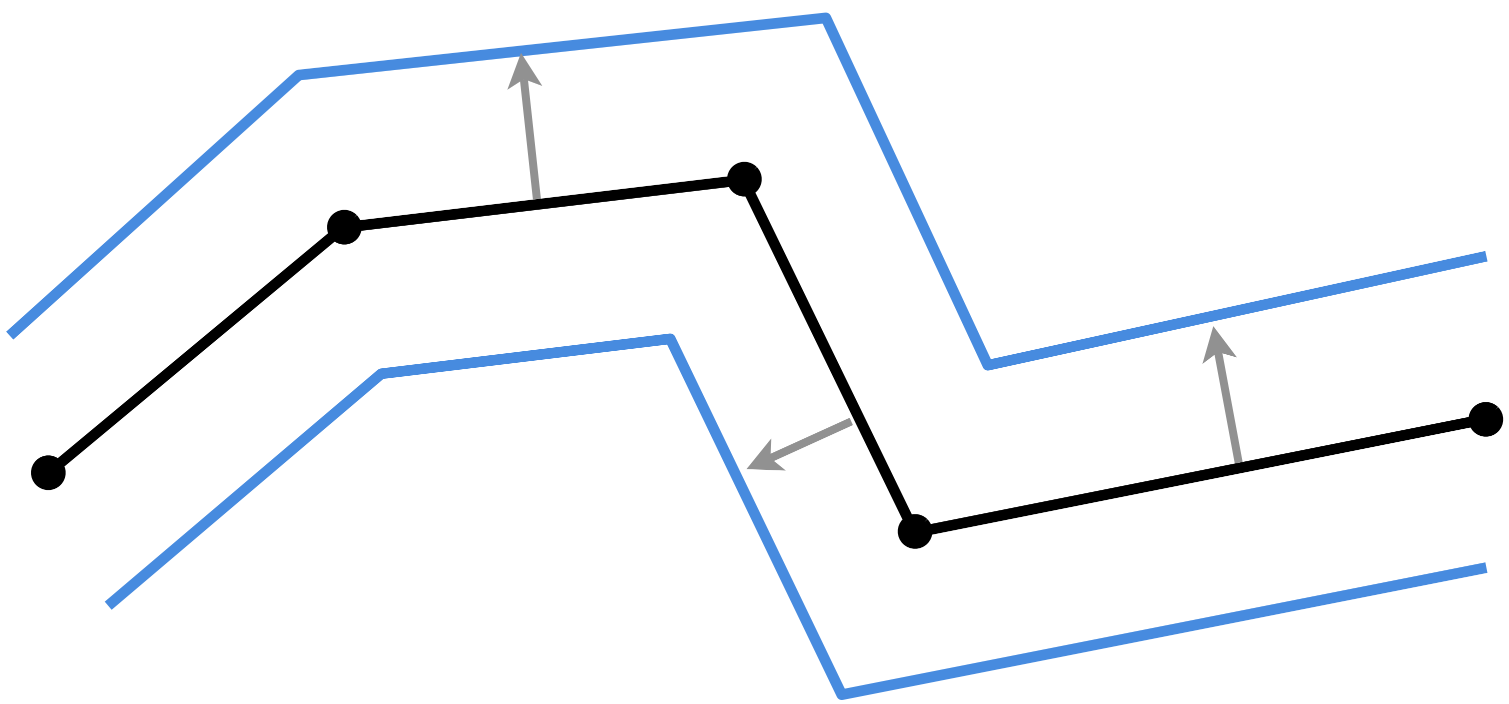
Local Error Metrics

- local distance to mesh [Schroeder et al. 92]
  - compute average plane
  - no comparison to *original* geometry



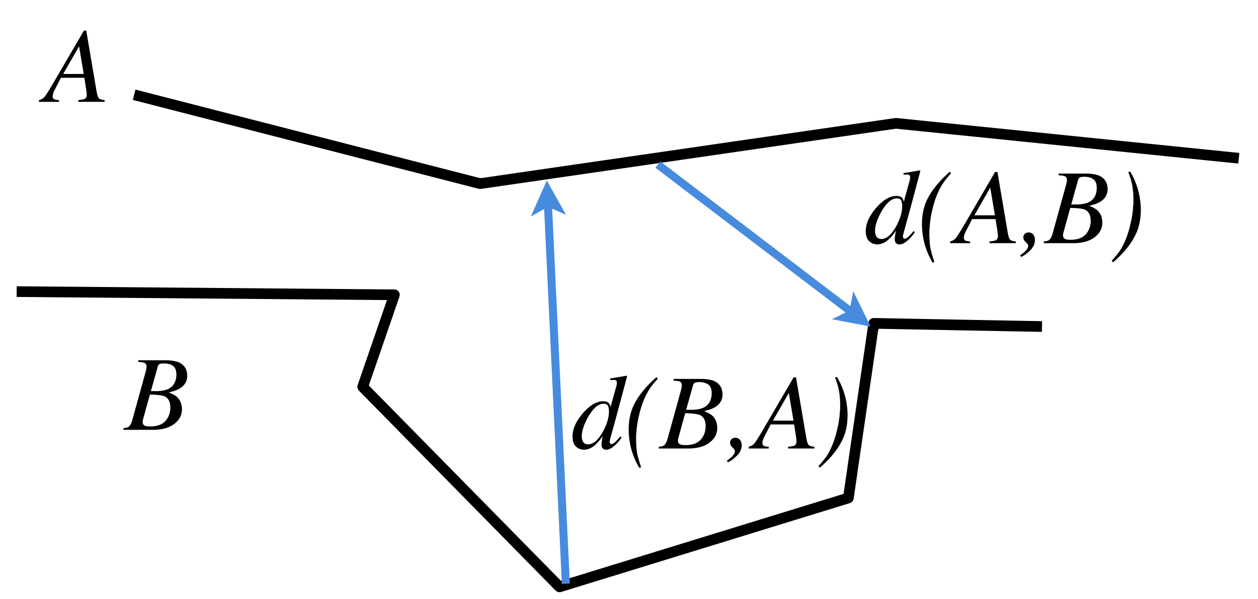
Global Error Metrics

- simplification envelopes [Cohen et al. 96]
  - compute (non-intersecting) offset surfaces
  - simplification guarantees to stay within bounds



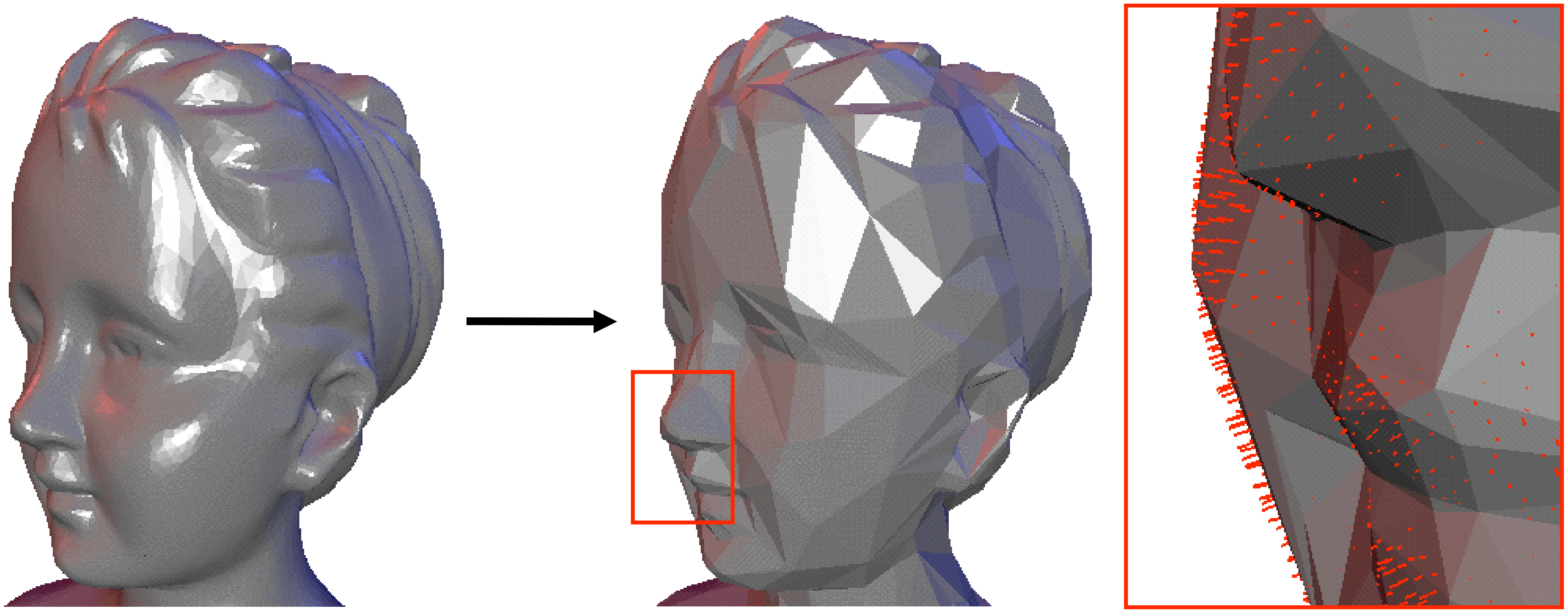
Global Error Metrics

- (two-sided) Hausdorff distance: maximum geometric deviation between two shapes
  - in general $d(A,B) \neq d(B,A)$
  - computationally involved



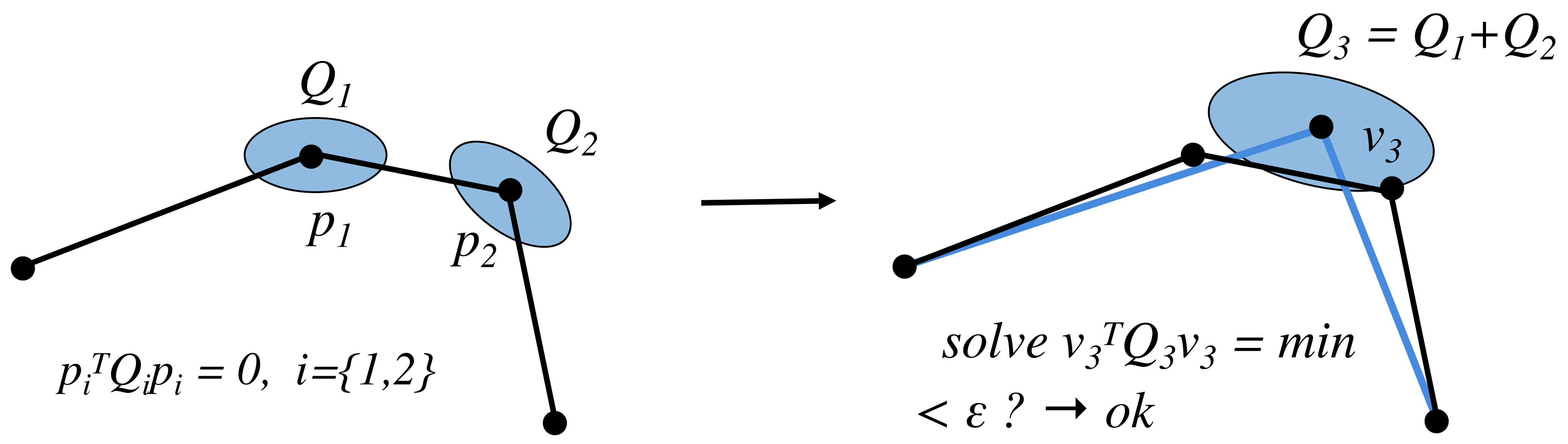
Global Error Metrics

- laser scan data:
  one-sided Hausdorff distance is sufficient
  - > from original vertices to current surface



Global Error Metrics

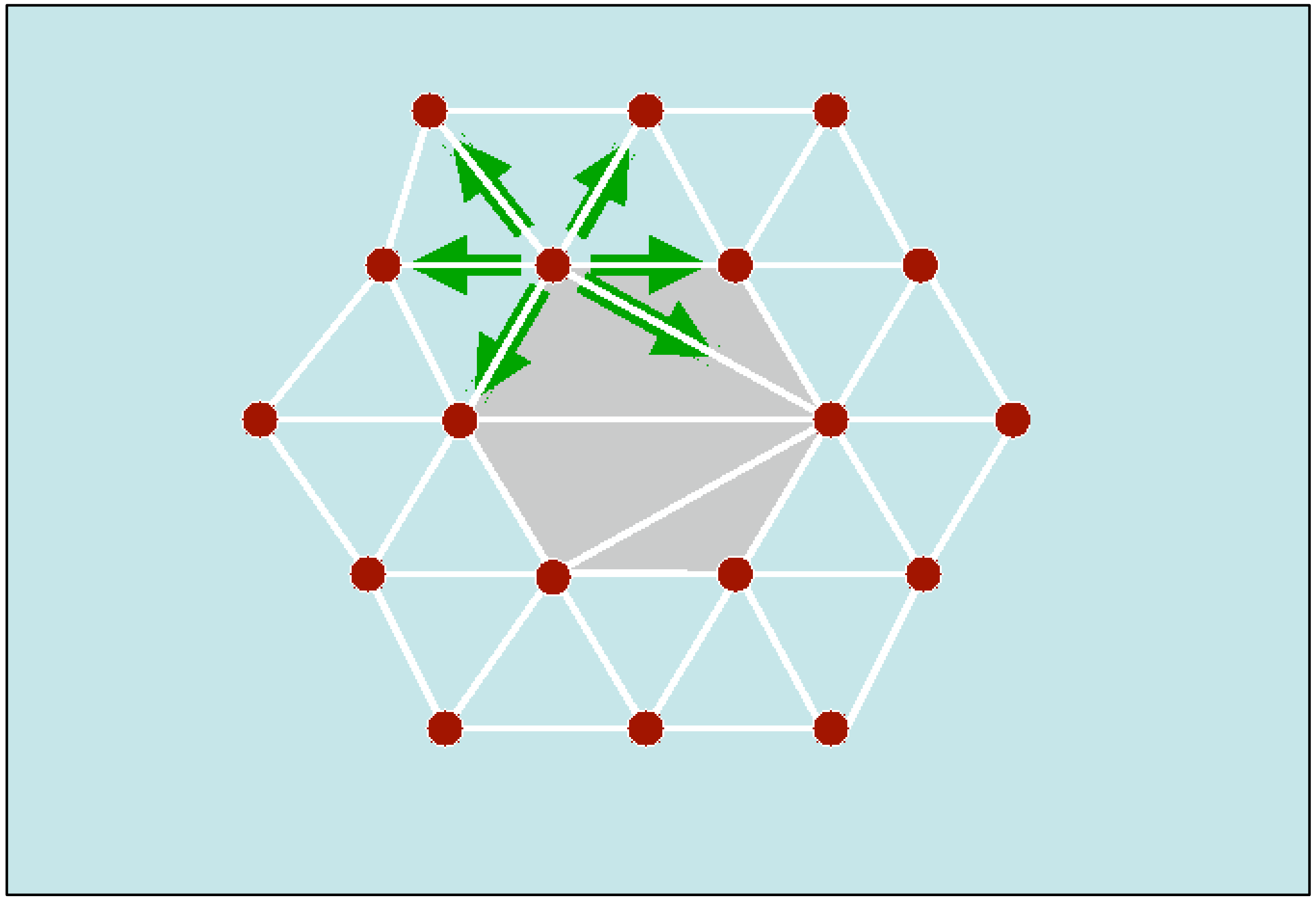
- error quadrics [Garland, Heckbert 97]
  - squared distance to triangle planes at vertices
  - no guaranteed bound on true error



Complexity

- n = number of vertices
- priority queue for half-edges
  - 6n * log (6n)   vs.   n * log(n)
- global error control
  - per vertex O(1+log(n)) $\Rightarrow$ overall O(n log(n))
    (decimate to x % triangles)
  - per vertex O(n+log(n)) $\Rightarrow$ overall O(n^2)
    (decimate to x triangles)

Priority Queue Updating



Incremental Decimation

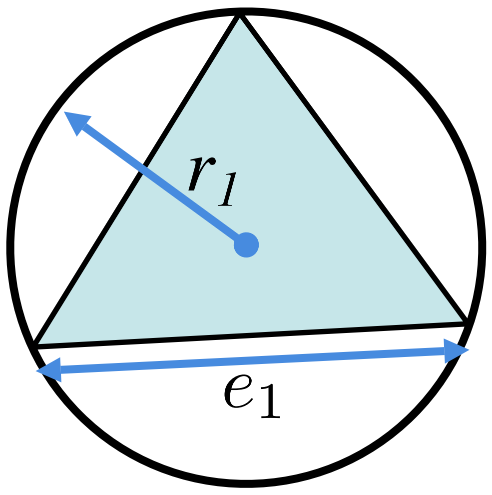
- general setup
- decimation operators
- error metrics
- fairness criteria
- topology changes

Greedy Control

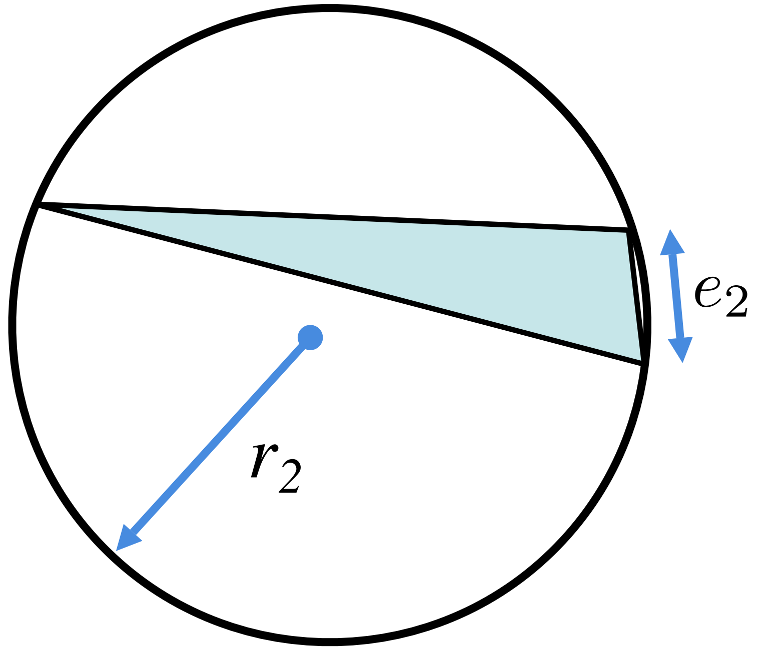
- prescribed approximation tolerance ε
- so far: minimally increase error
- now: use error as binary criterion
- other criteria determine decimation order

Fairness Criteria

- rate quality after decimation
  - triangle shape
  - dihedral angles
  - valence balance
  - color differences
  - ...

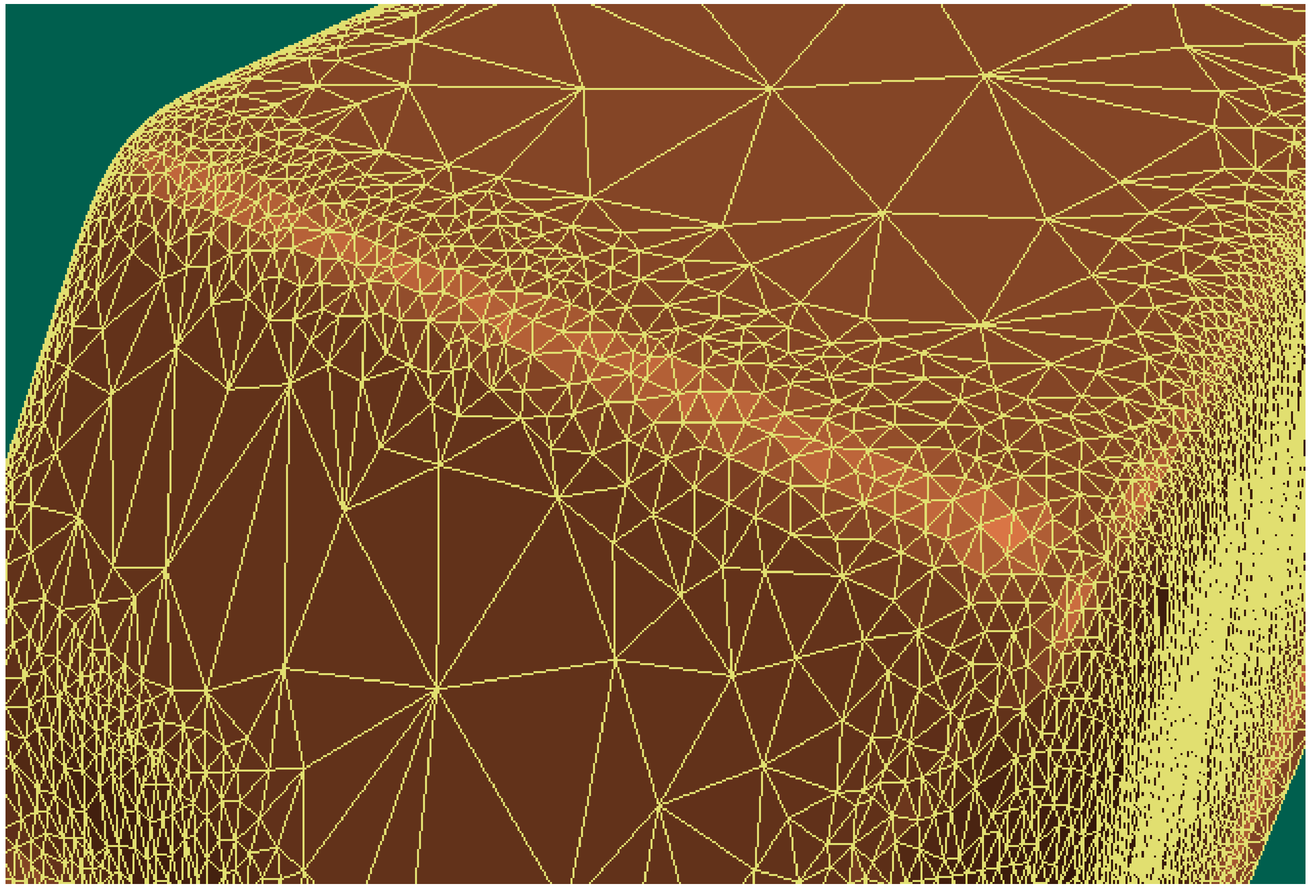


$$\frac{r_1}{e_1} < \frac{r_2}{e_2}$$



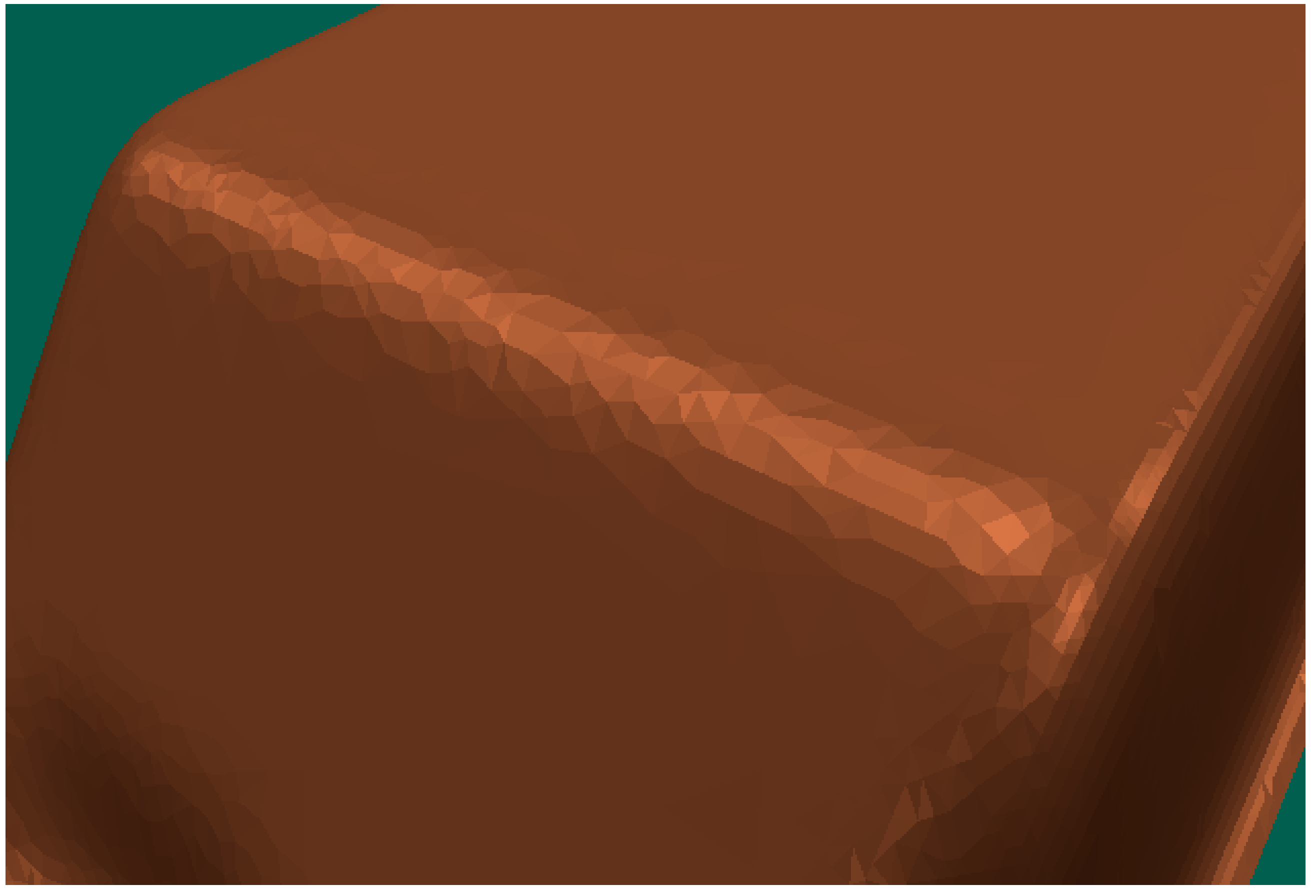
Fairness Criteria

- rate quality after decimation
  - triangle shape
  - dihedral angles
  - valence balance
  - color differences
  - ...



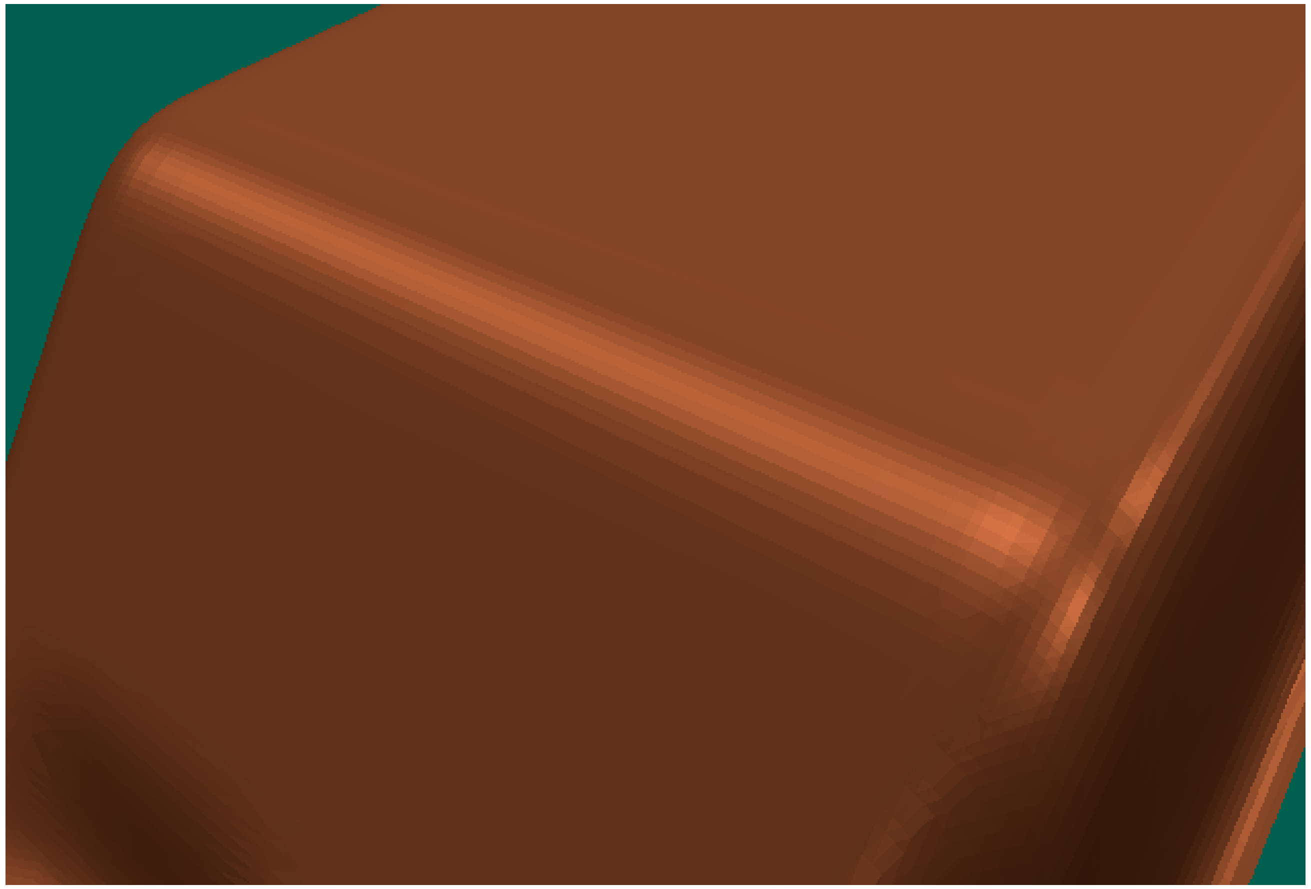
Fairness Criteria

- rate quality after decimation
  - triangle shape
  - dihedral angles
  - valence balance
  - color differences
  - ...



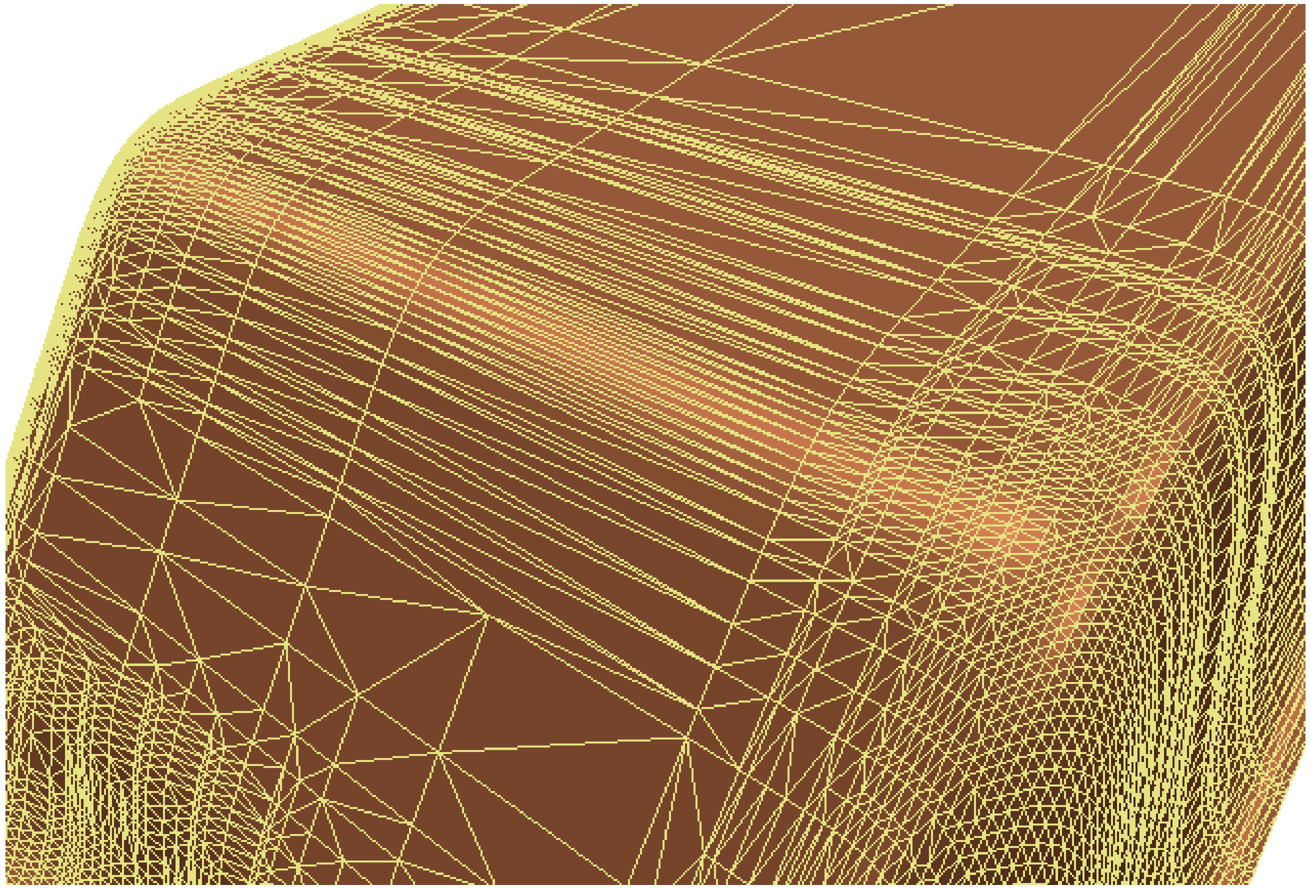
Fairness Criteria

- rate quality after decimation
  - triangle shape
  - dihedral angles
  - valence balance
  - color differences
  - ...



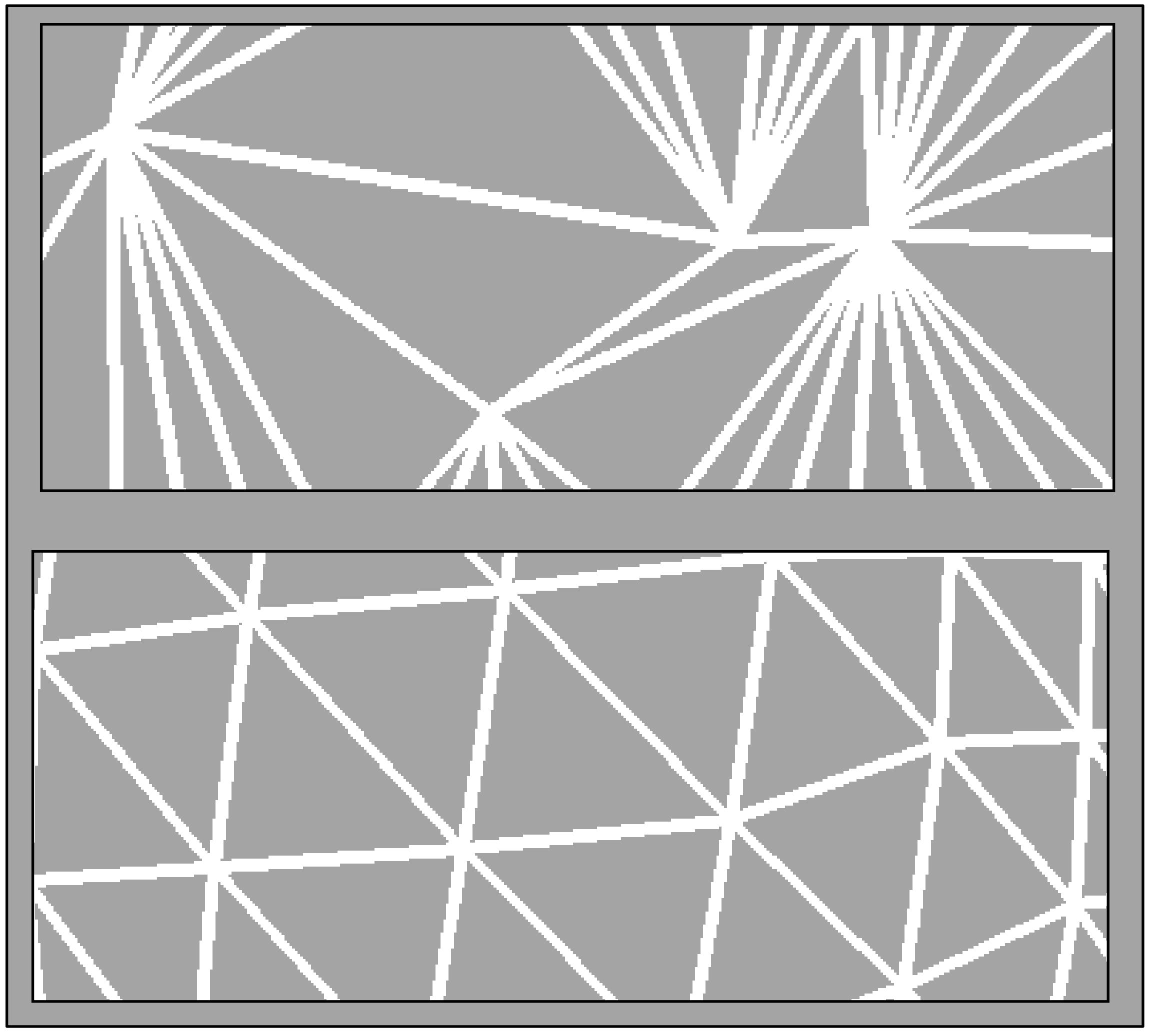
Fairness Criteria

- rate quality after decimation
  - triangle shape
  - dihedral angles
  - valence balance
  - color differences
  - ...



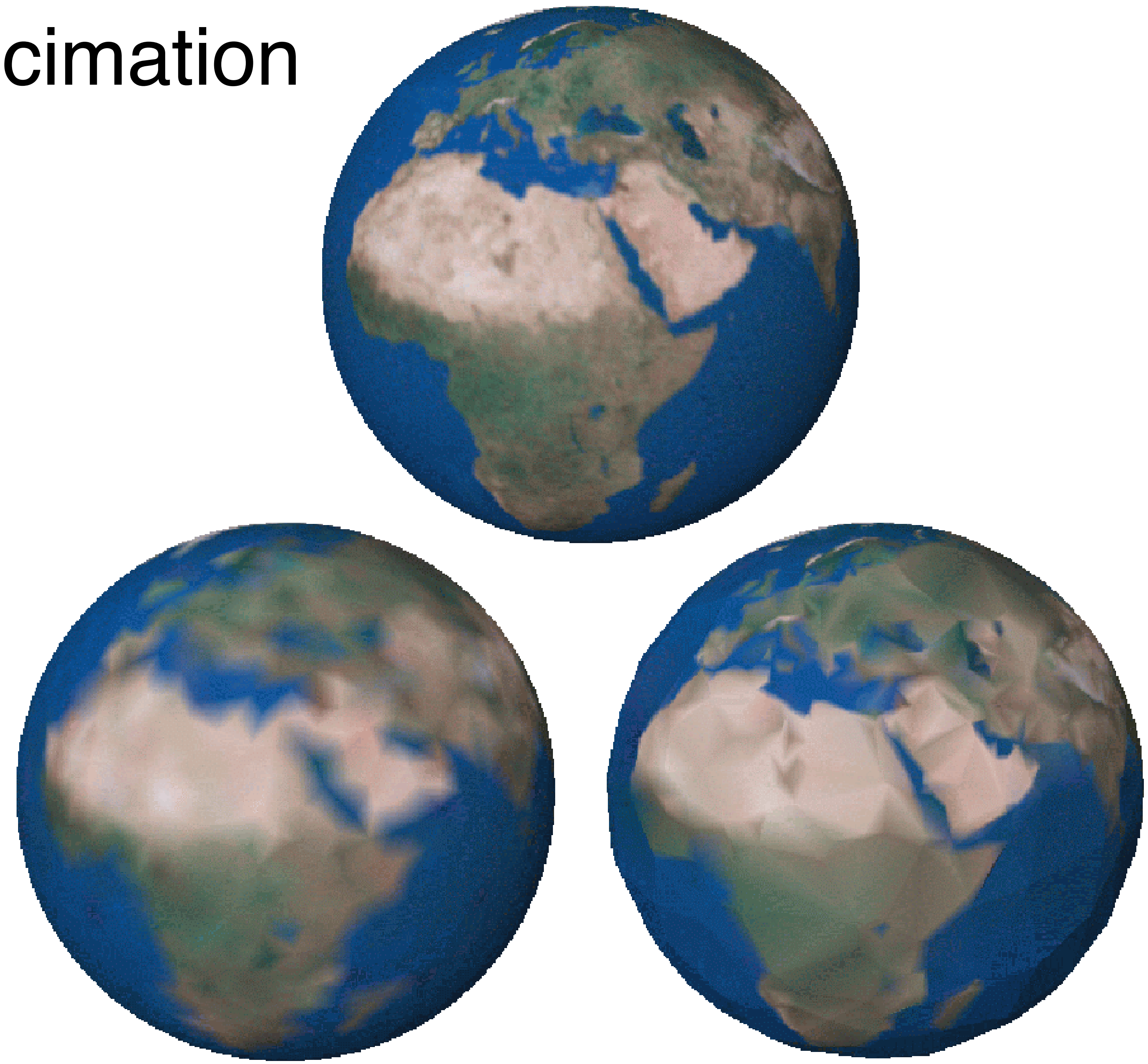
Fairness Criteria

- rate quality after decimation
  - triangle shape
  - dihedral angles
  - valence balance
  - color differences
  - ...



Fairness Criteria

- rate quality after decimation
  - triangle shape
  - dihedral angles
  - valance balance
  - color differences
  - ...

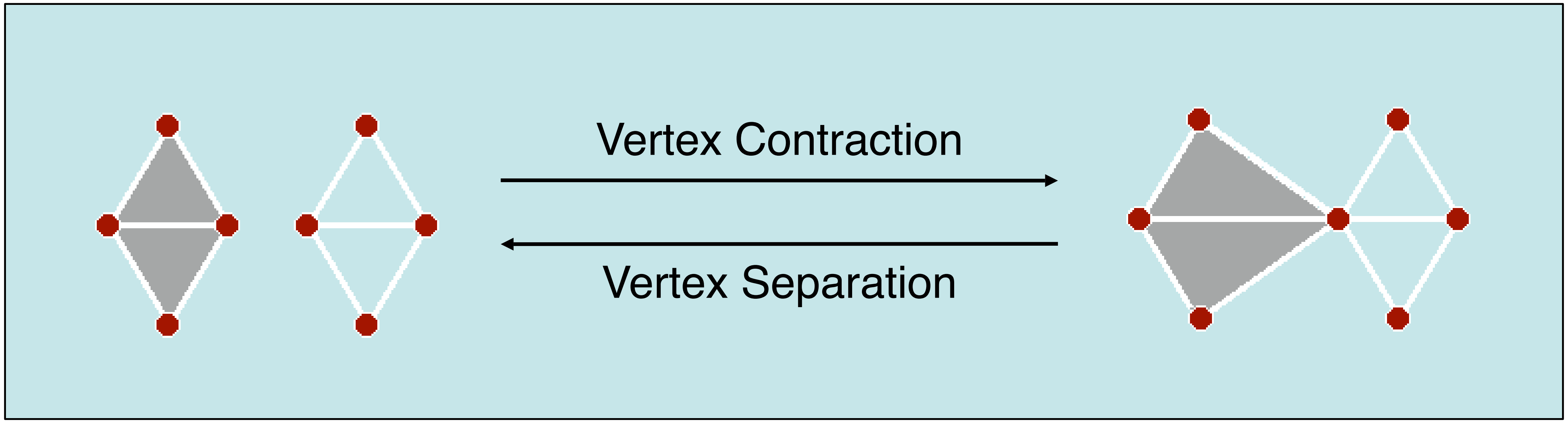


Incremental Decimation

- general setup
- decimation operators
- error metrics
- fairness criteria
- topology changes

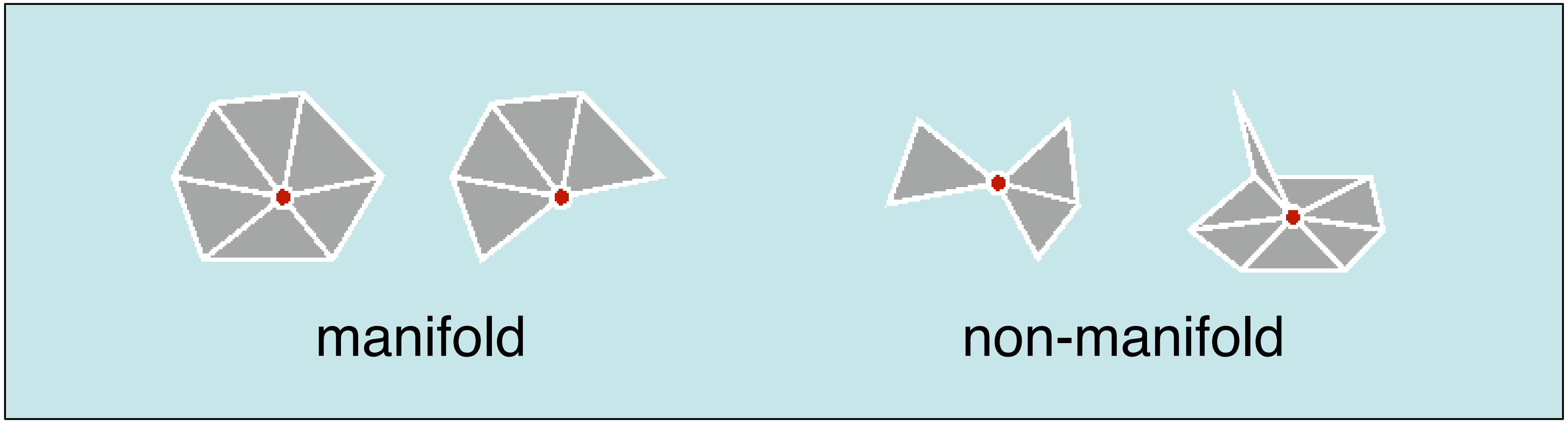
Topology Changes ?

- merge vertices across non-edges
  - changes mesh topology
  - need *spatial neighborhood* information
  - generates *non-manifold* meshes



Topology Changes ?

- merge vertices across non-edges
  - changes mesh topology
  - need *spatial neighborhood* information
  - generates *non-manifold* meshes



Comparison

- vertex clustering
  - fast, but difficult to control target complexity
  - topology changes, non-manifold meshes
  - global error bound, but often far from optimal
- incremental decimation with quadric error metrics
  - good trade-off between mesh quality and speed
  - explicit control over mesh topology
  - restricting normal deviation improves mesh quality

Outline

- applications
- problem statement
- mesh decimation schemes
  - vertex clustering
  - incremental decimation
  - **out-of-core**

Out-of-core Decimation

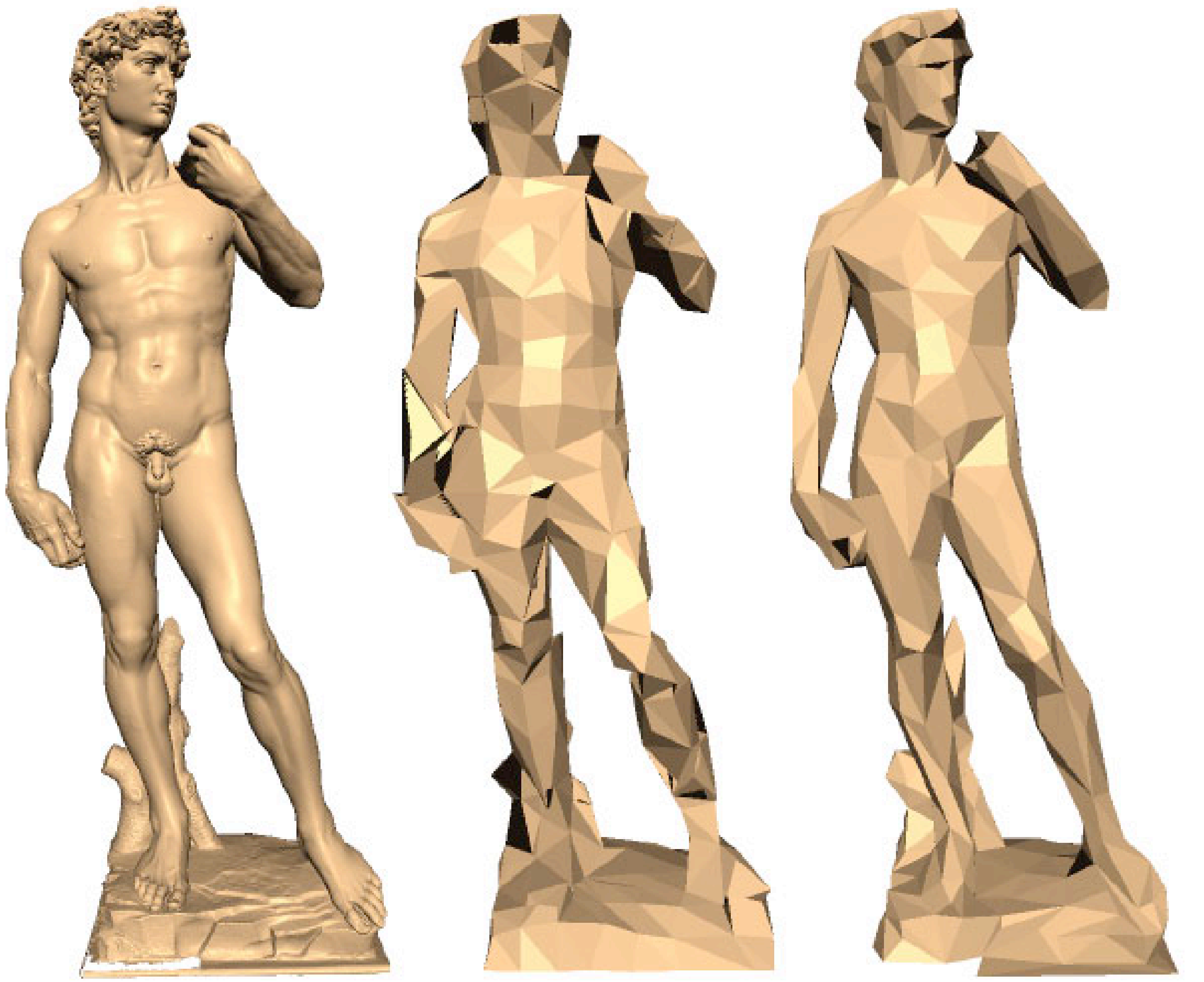
- handle extremely large data sets that do not fit into main memory
- key idea: avoid random access to the mesh data structure during simplification
- examples
  - Garland, Shaffer: *A Multiphase Approach to Efficient Surface Simplification*, IEEE Visualization 2002
  - Wu, Kobbelt: *A Stream Algorithm for the Decimation of Massive Meshes*, Graphics Interface 2003

Multiphase Simplification

1. phase: out-of-core clustering
   - compute accumulated error quadrics and vertex representative for each cell of uniform voxel grid

2. phase: in-core incremental simplification
   - lookup initial quadrics in voxel grid
   - iteratively contract edge of smallest cost

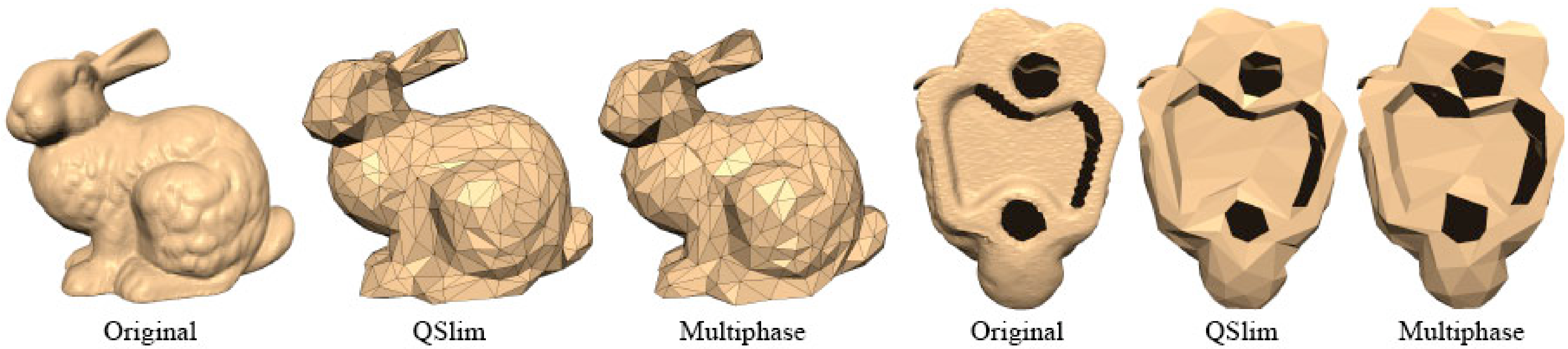
Multiphase Simplification

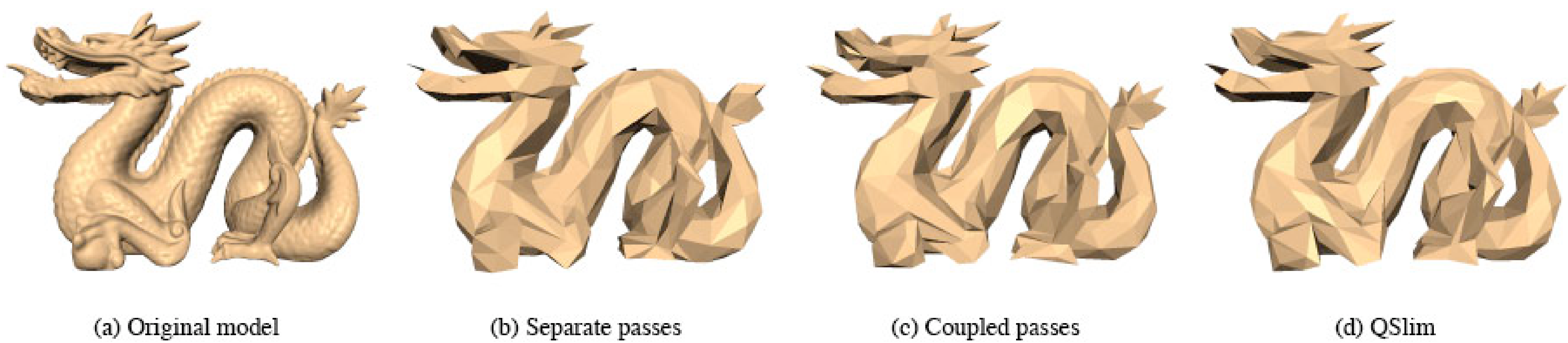


Original (8 million triangles) | Uniform clustering (1157 triangles) | Multiphase (1000 triangles)

Garland, Shaffer: *A Multiphase Approach to Efficient Surface Simplification*, IEEE Visualization 2002

Multiphase Simplification

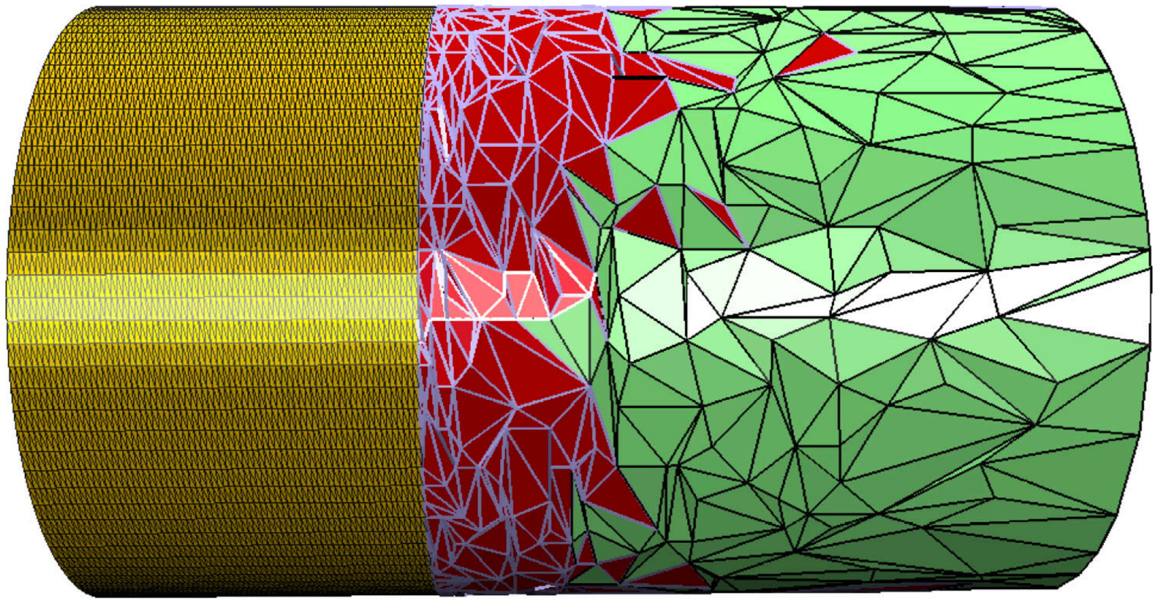




Garland, Shaffer: *A Multiphase Approach to Efficient Surface Simplification*, IEEE Visualization 2002

Out-of-core Decimation

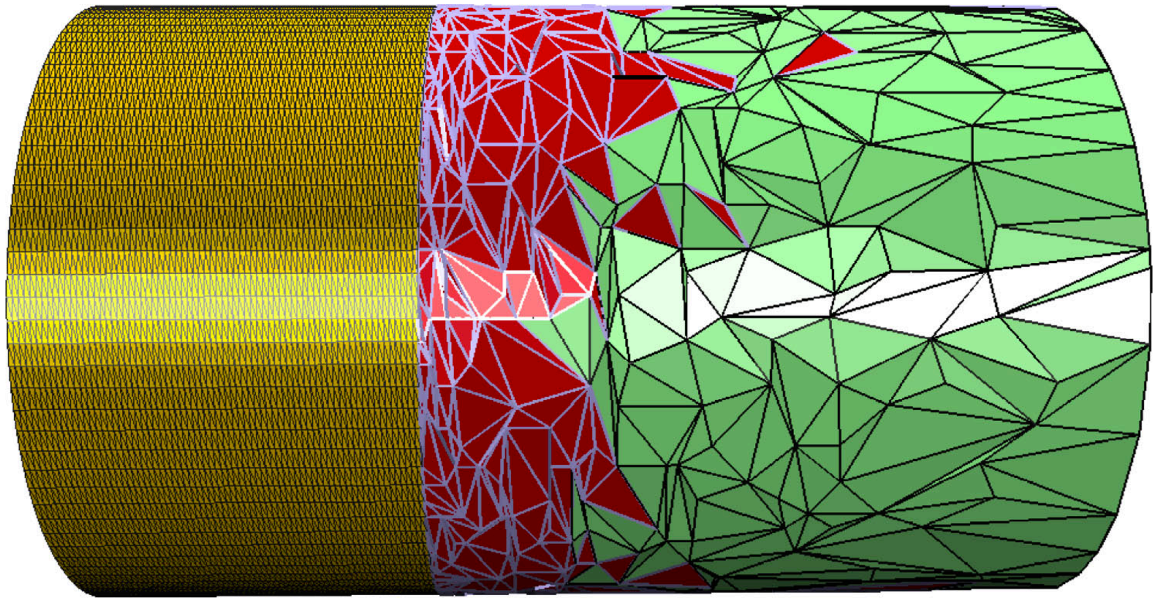
- streaming approach based on edge collapse operations using QEM
- pre-sorted input stream allows fixed-sized active working set



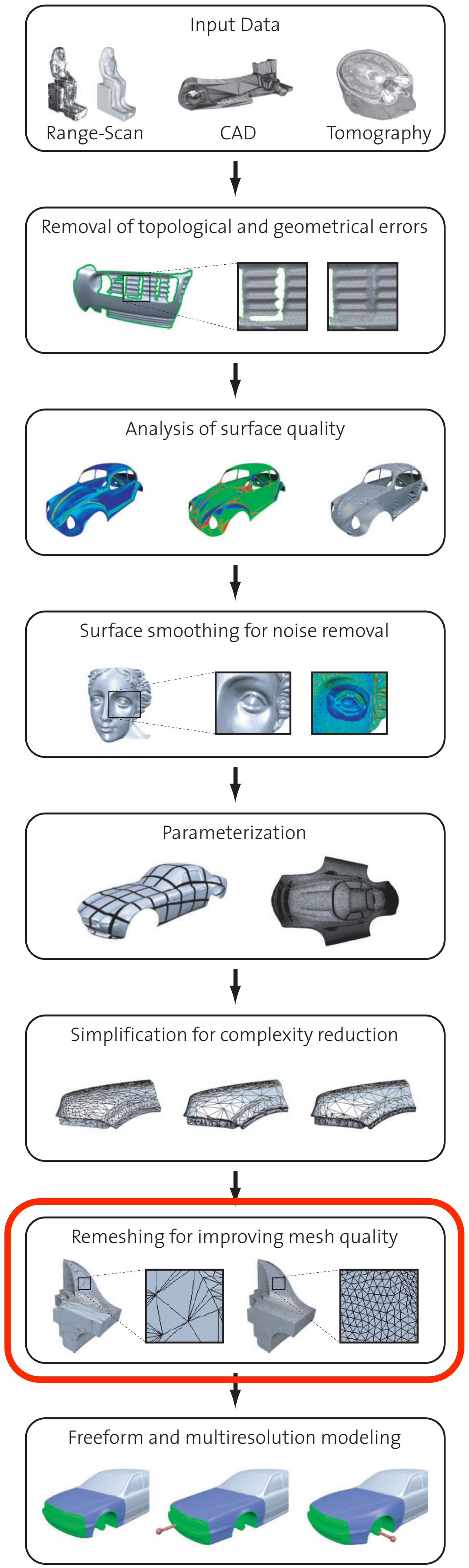
Wu, Kobbelt: *A Stream Algorithm for the Decimation of Massive Meshes*, Graphics Interface 2003

Out-of-core Decimation

- randomized multiple choice optimization avoids global heap data structure
- memory requirements independent from input AND output complexity



Wu, Kobbelt: *A Stream Algorithm for the Decimation of Massive Meshes*, Graphics Interface 2003



Remeshing

Leif Kobbelt

RWTH Aachen

Remeshing Cookbook

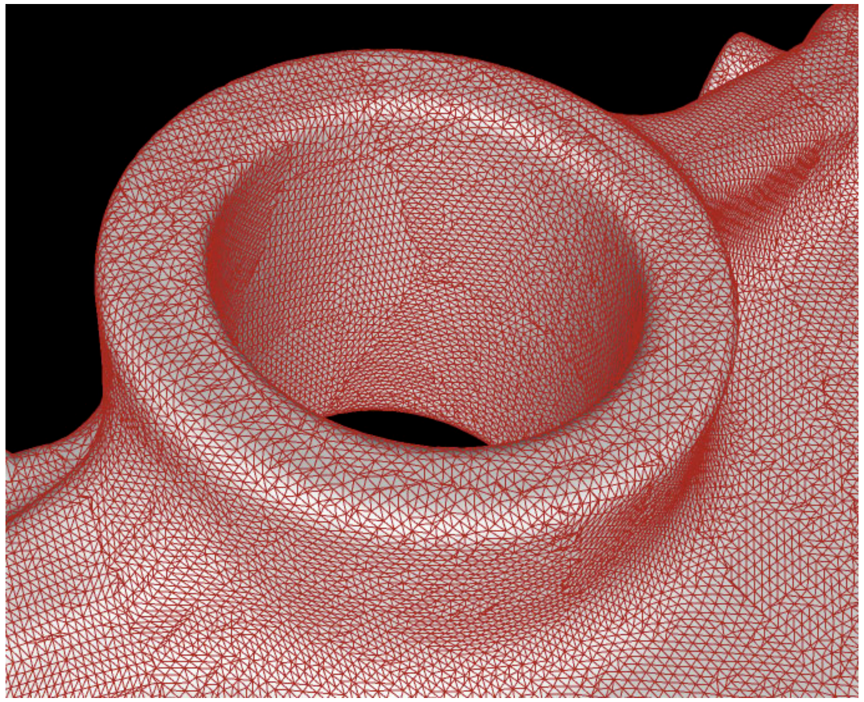
- problem definition
  - input, output
- basic ingredients
  - general requirements
  - types of operations
- a selection of recipes
  - various representative examples of known remeshing schemes

Remeshing Cookbook

- problem definition
  - input, output
- basic ingredients
  - general requirements
  - types of operations
- a selection of recipes
  - various representative examples of known remeshing schemes

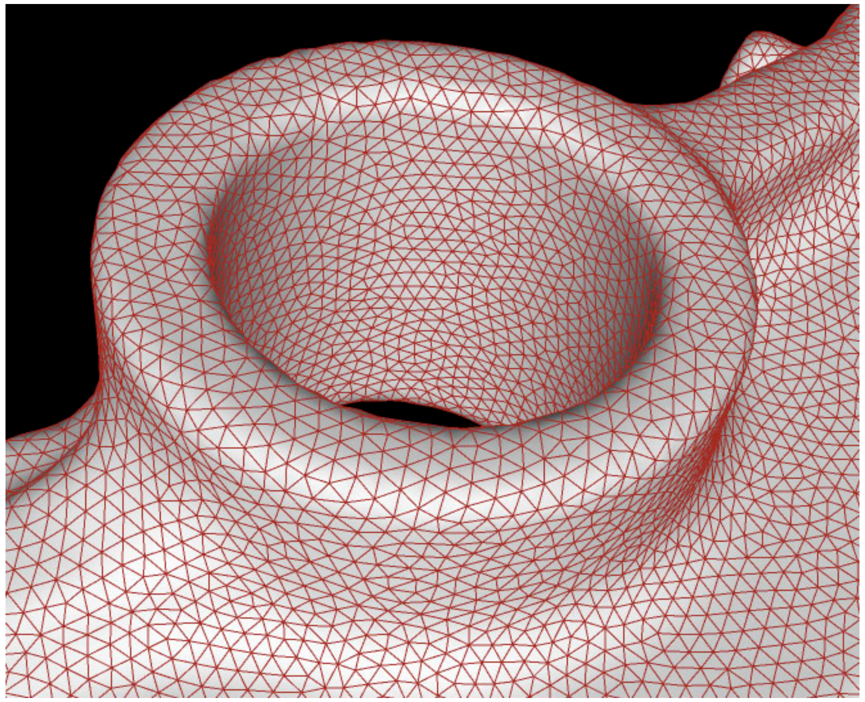
Problem Definition

- input ***M***
  - polygon (triangle) mesh
  - properly defined surface
    - 2-manifold
    - with / without boundary
    - homeomorphic to a disk (?) (pre-segmented)
  - generate new samples
  - access to geodesic neighborhood



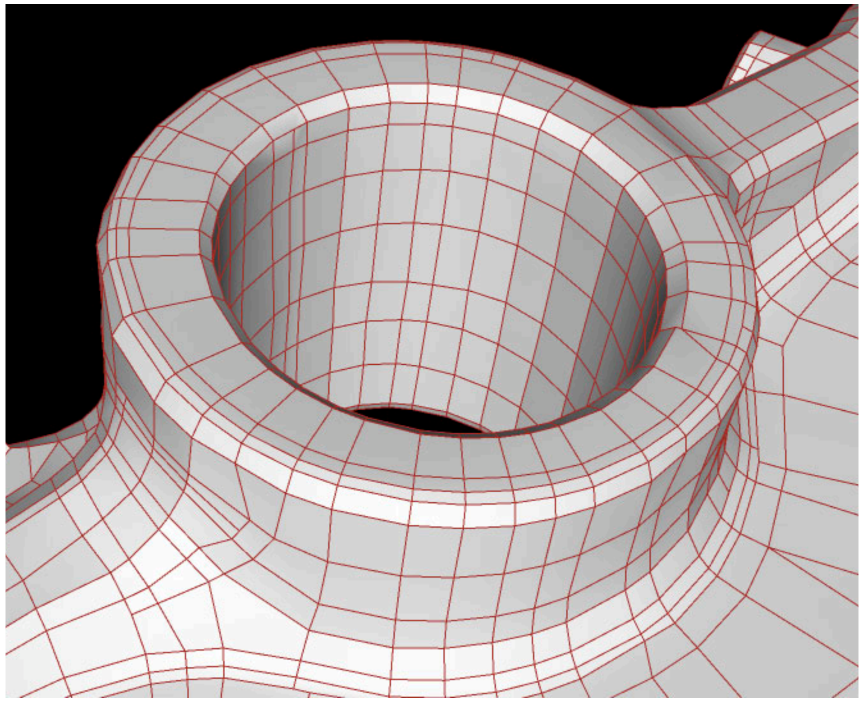
Problem Definition

- output ***R***
  - approximation to the input data ***M***
    - prescribed (Hausdorff) error tolerance **ε**
    - target complexity / target edge length **δ**
  - better vertex distribution
    - uniform vs. adaptive
  - shape of individual faces
  - local and global alignment



Problem Definition

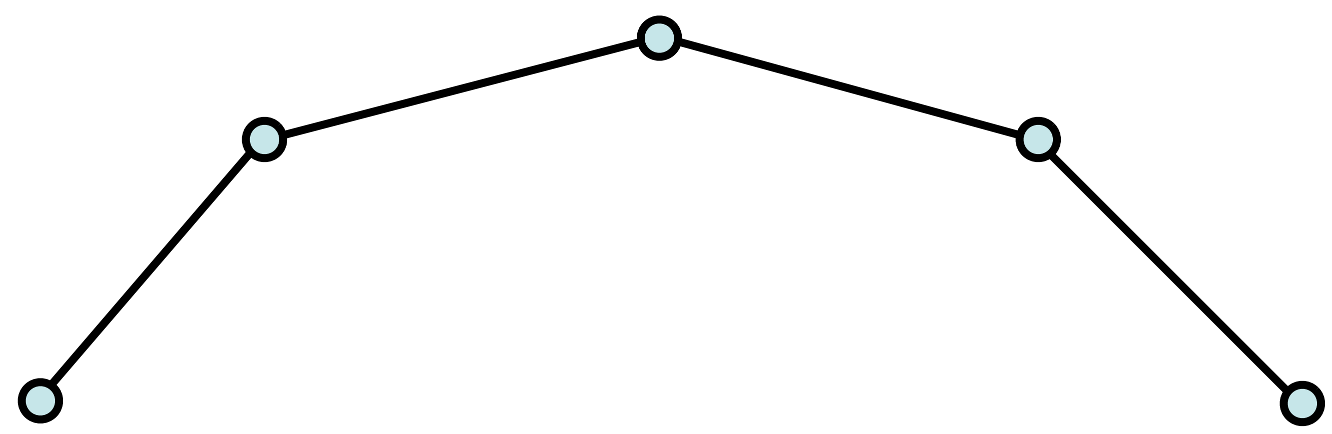
- output ***R***
  - approximation to the input data ***M***
    - prescribed (Hausdorff) error tolerance ε
    - target complexity / target edge length δ
  - better vertex distribution
    - uniform vs. adaptive
  - shape of individual faces
  - local and global alignment



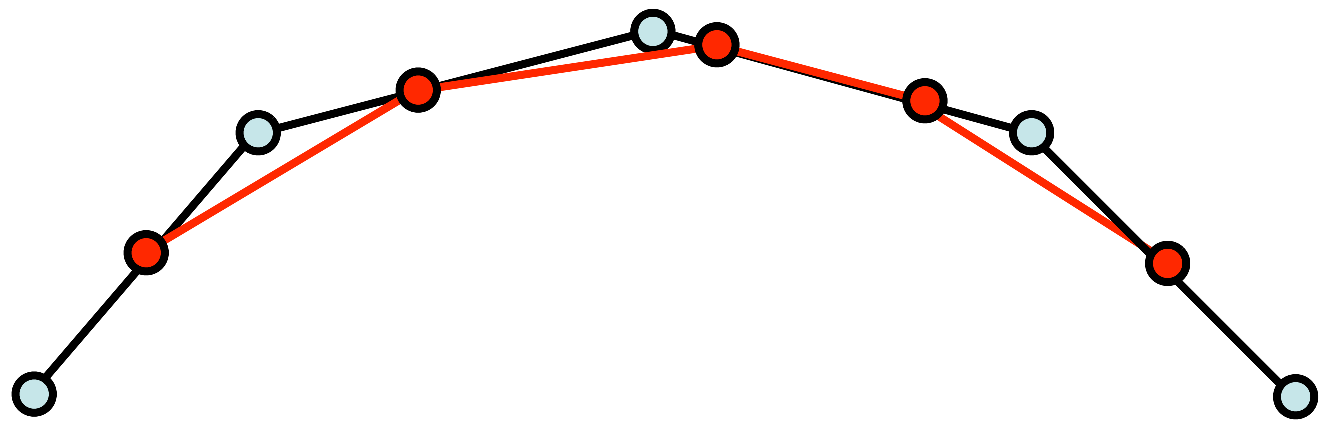
Two Fundamental Approaches

- surface oriented
  - operate directly of the surface
  - treat surface as a set of points / polygons in space
  - efficient for high resolution remeshing (locally flat surface)
- parametrization based
  - map to 2D domain / 2D problem
  - computationally more expensive (?)
  - works even for coarse resolution remeshing (features might be lost)

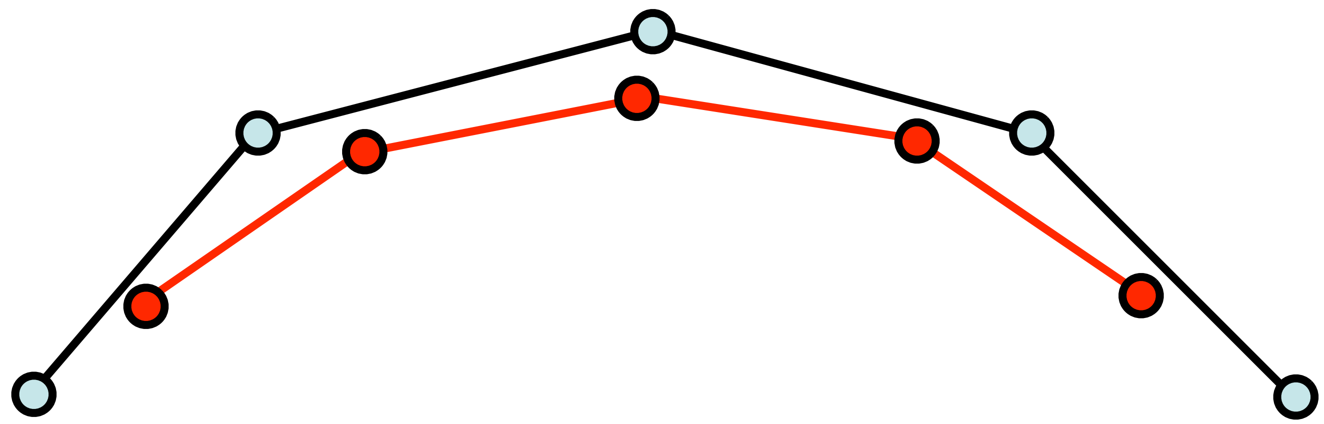
Surface Oriented



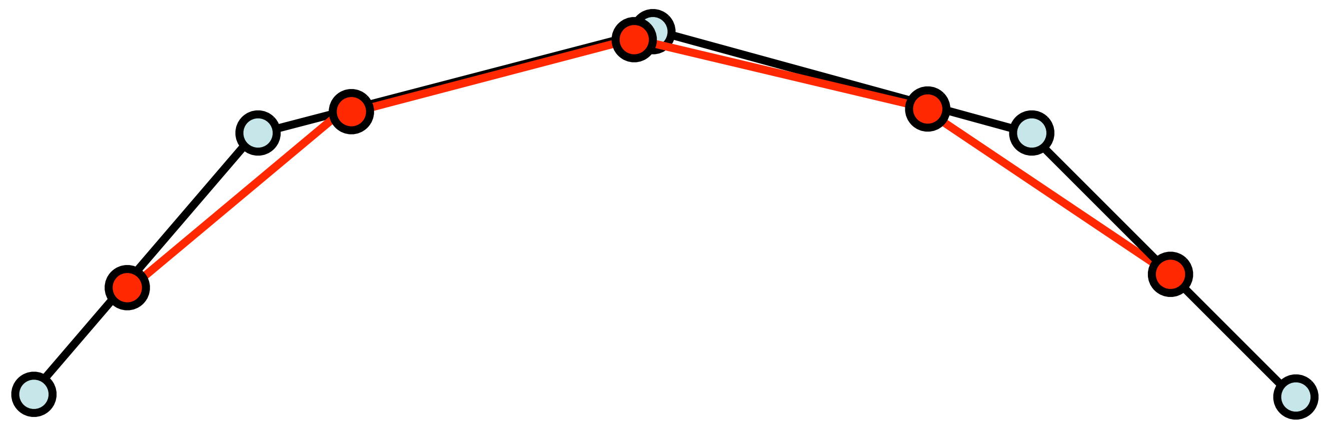
Surface Oriented



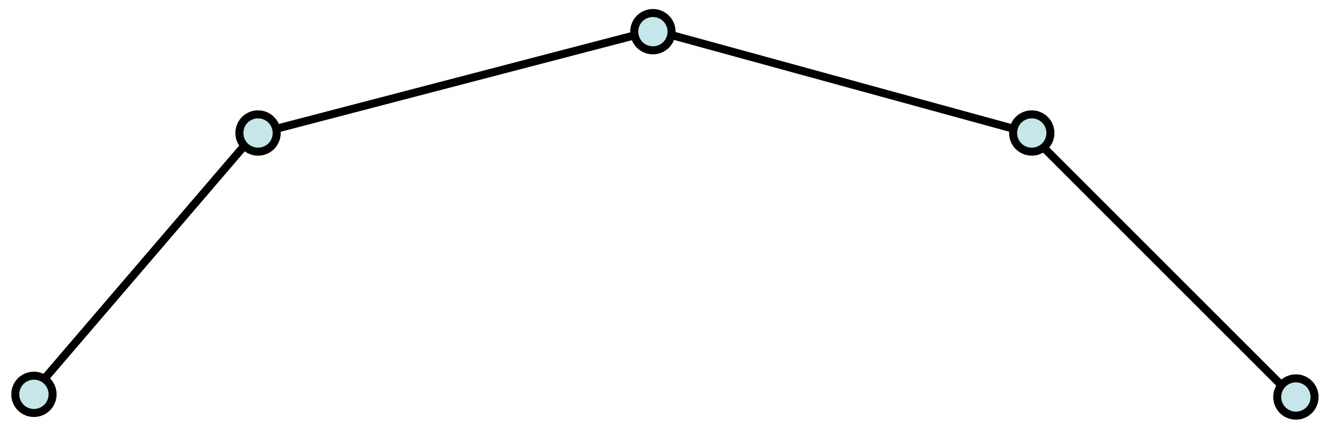
Surface Oriented



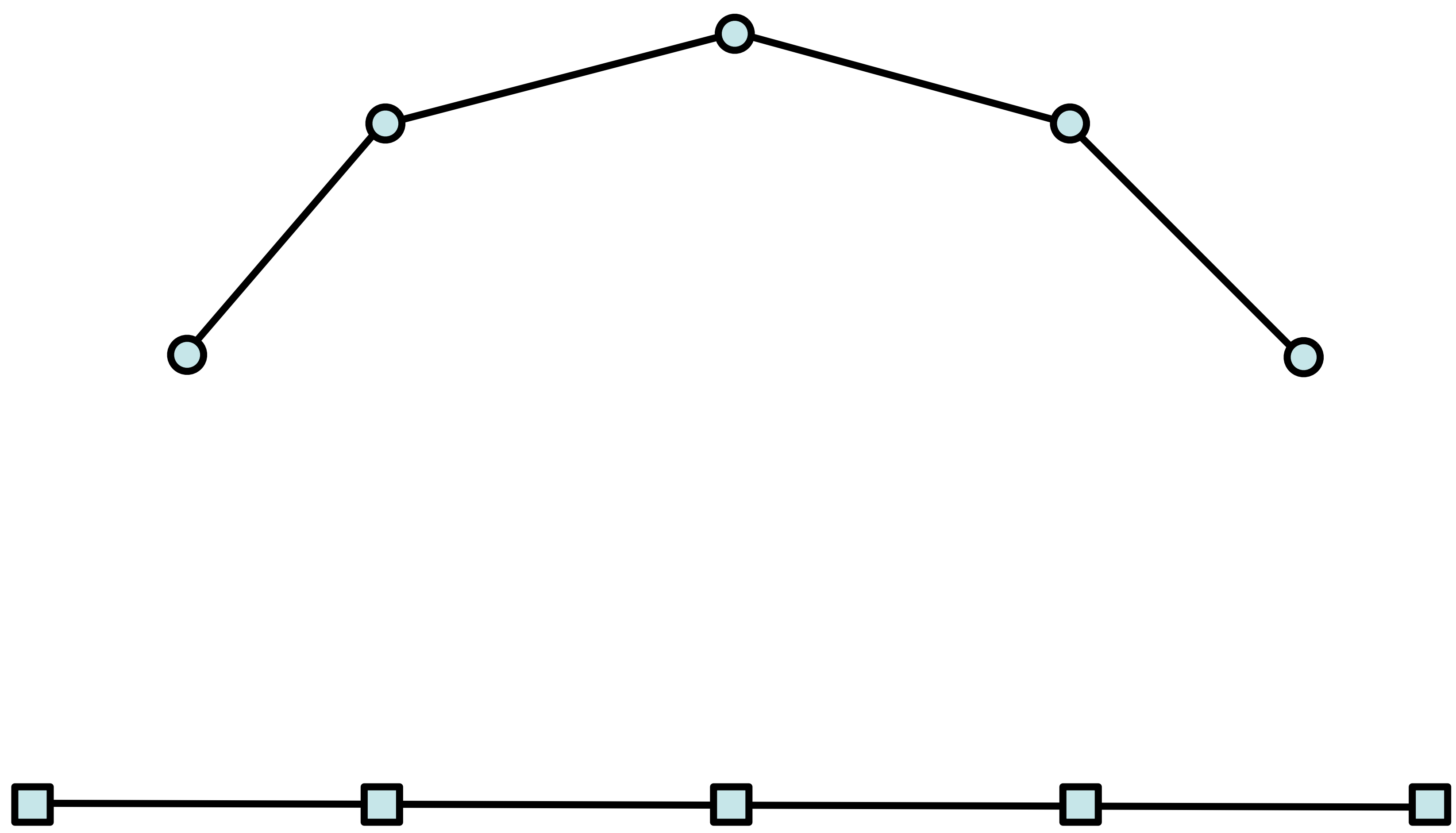
Surface Oriented



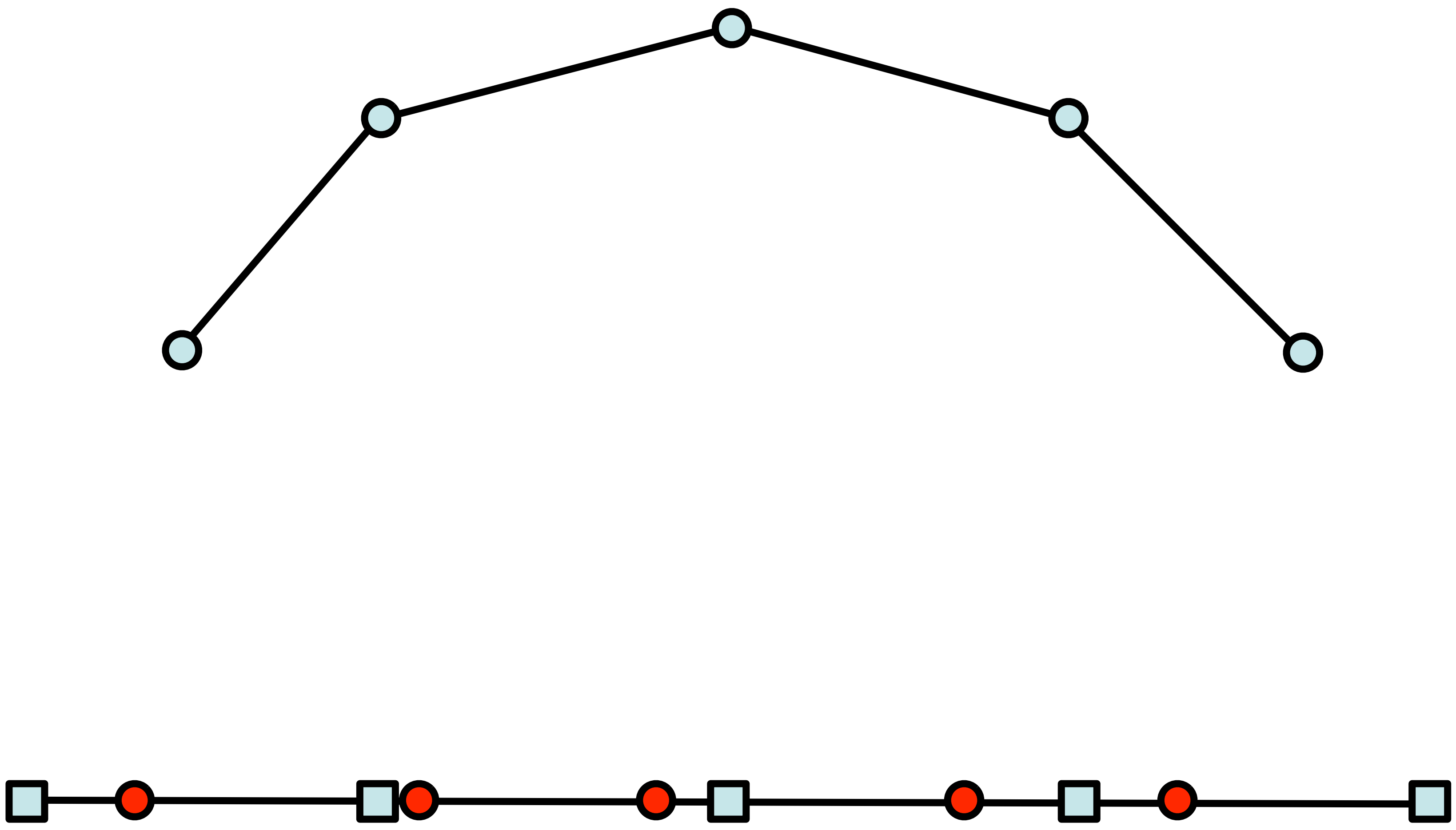
Parametrization Based



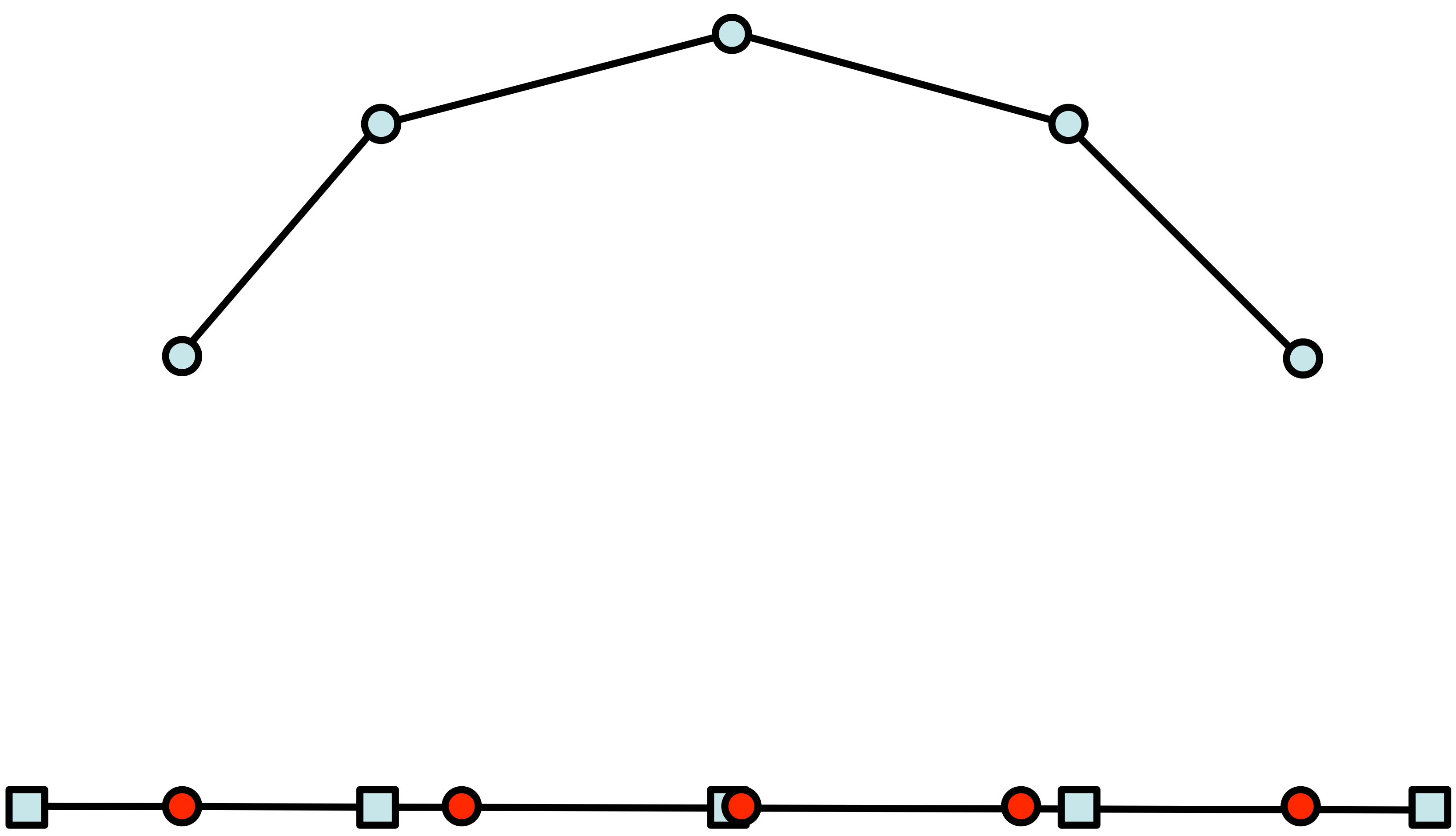
Parametrization Based



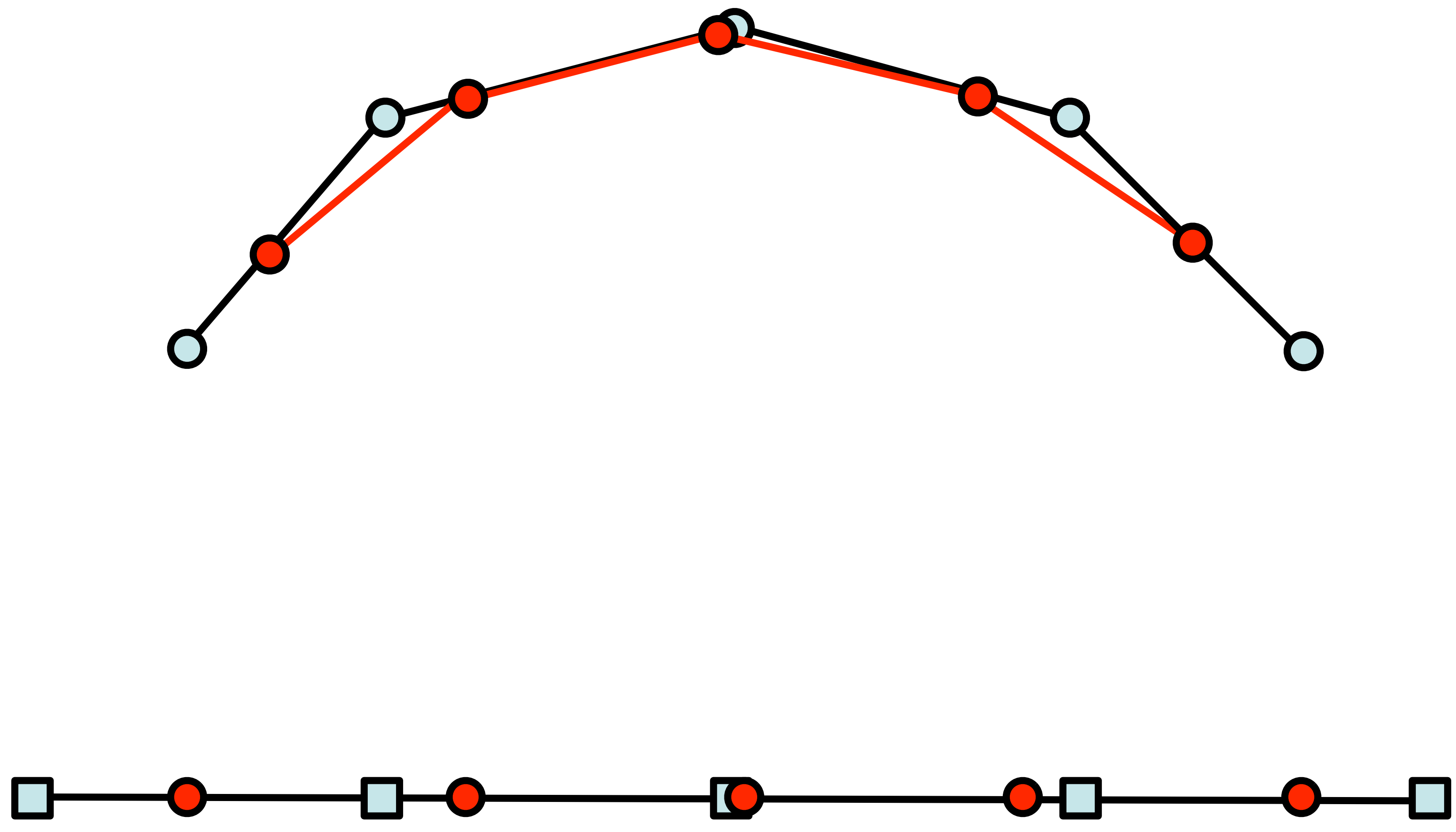
Parametrization Based



Parametrization Based



Parametrization Based



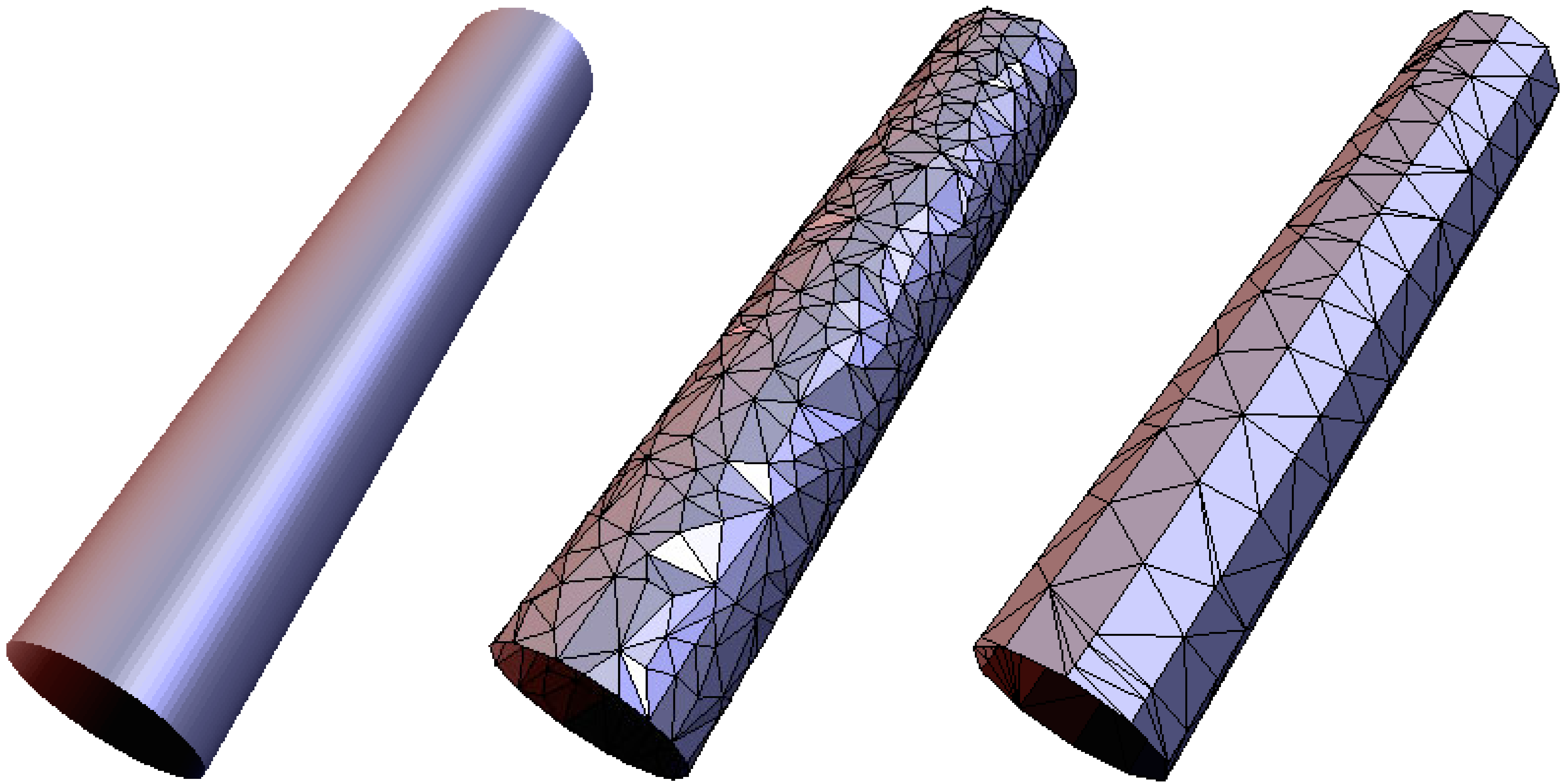
Remeshing Cookbook

- problem definition
  - input, output
- basic ingredients
  - general requirements
  - types of operations
- a selection of recipes
  - various representative examples of known remeshing schemes

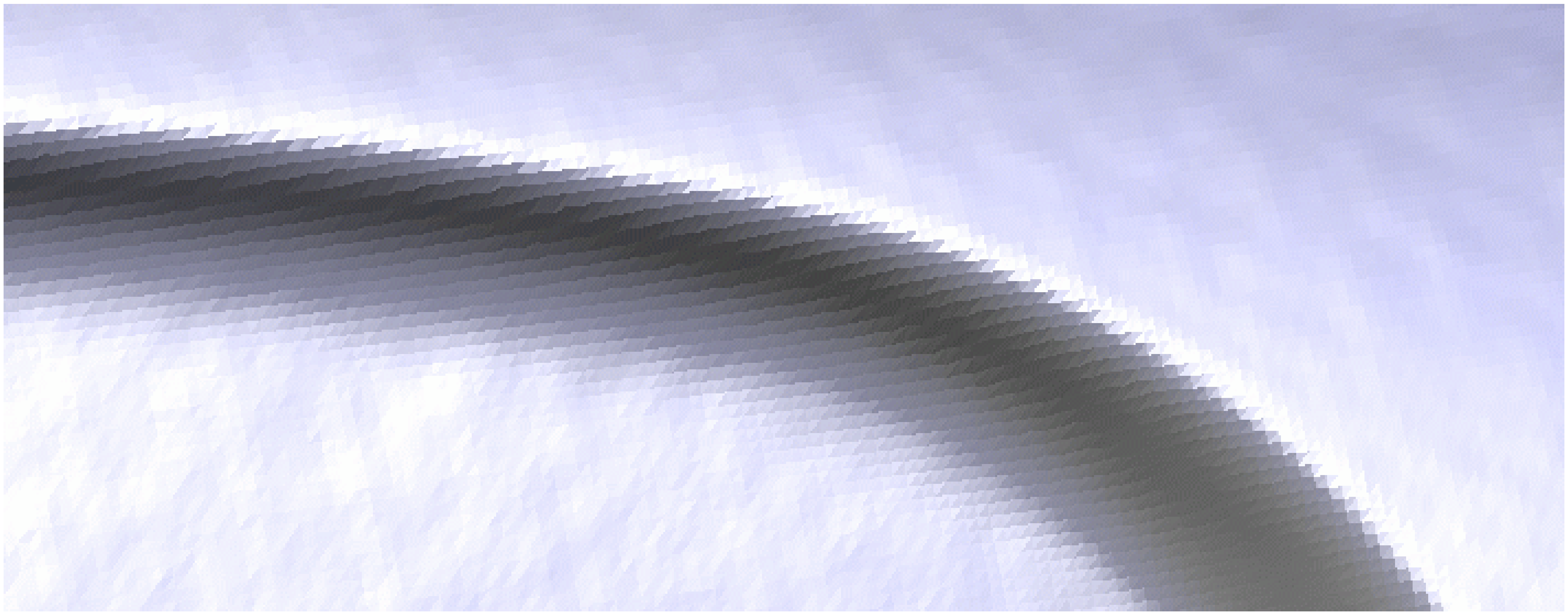
Basic Ingredients

- parametrization
  - global vs. local
- vertex density control
  - uniform vs. adaptive
  - isotropic vs. anisotropic
- local alignment
  - optimal shape approximation
- global alignment
  - feature sensitivity

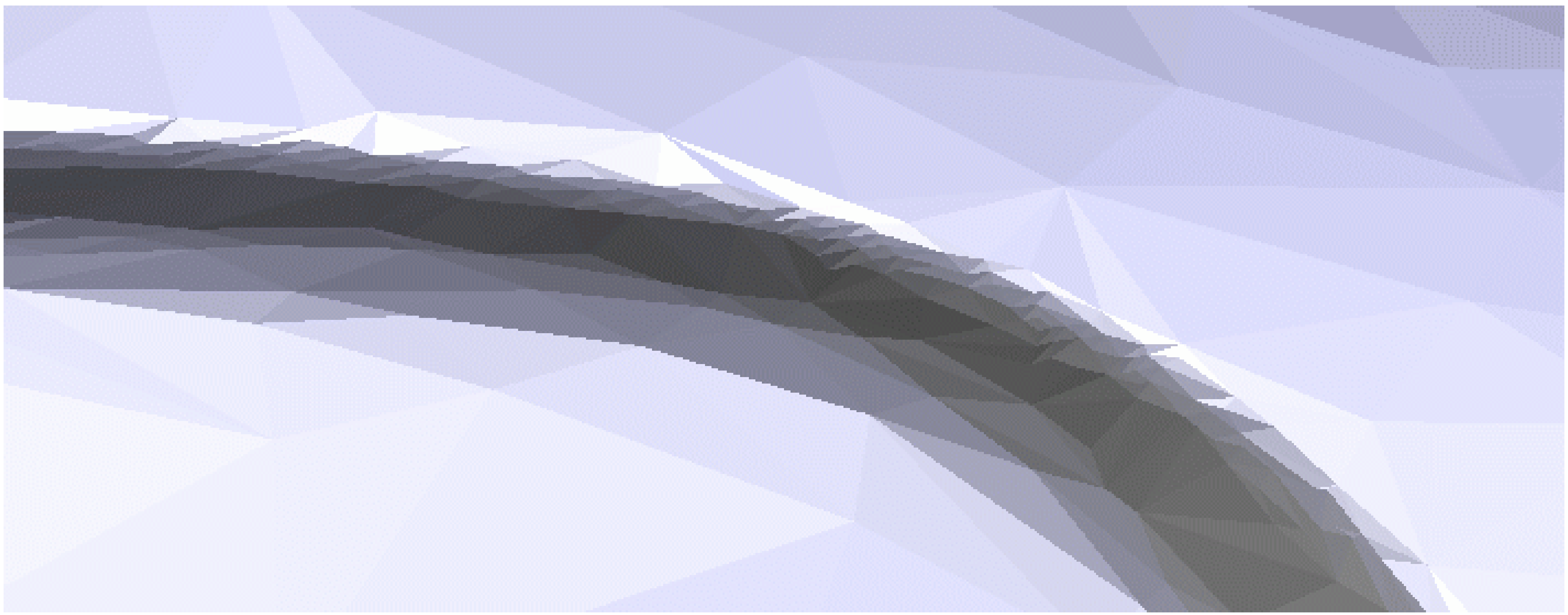
Alignment and Approximation



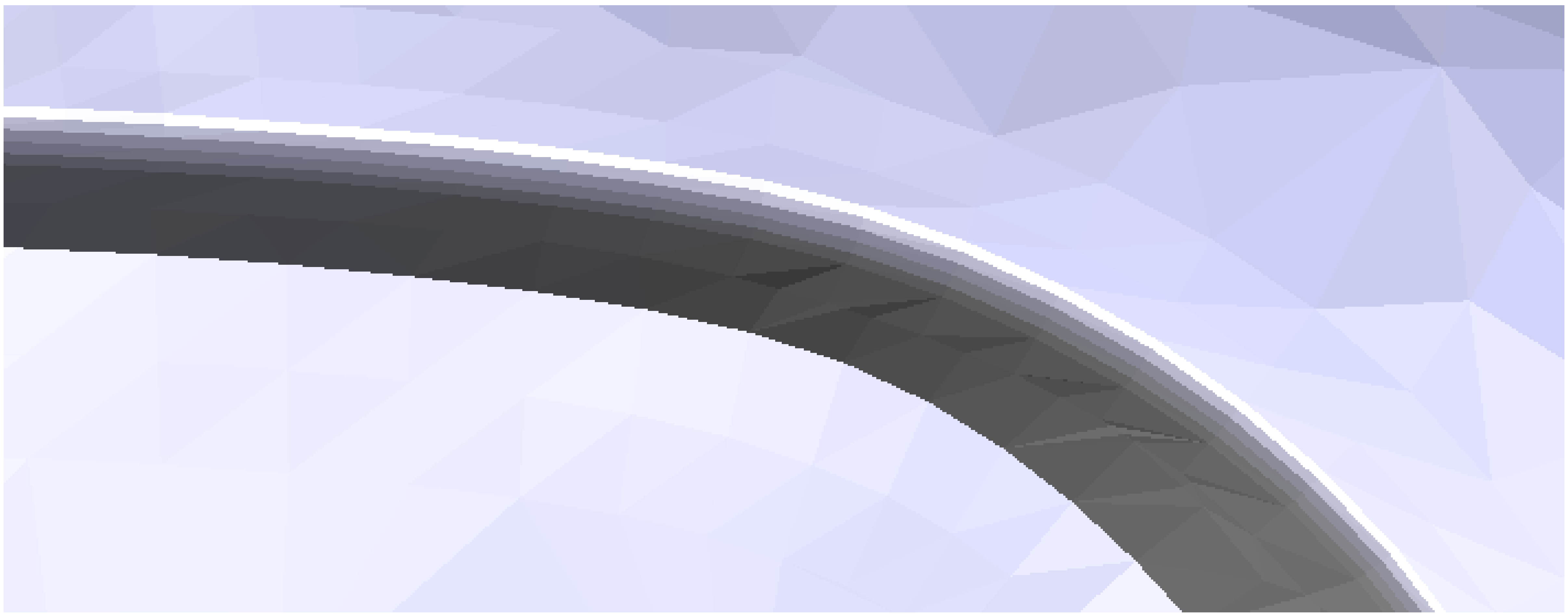
Alignment and Approximation



Alignment and Approximation



Alignment and Approximation

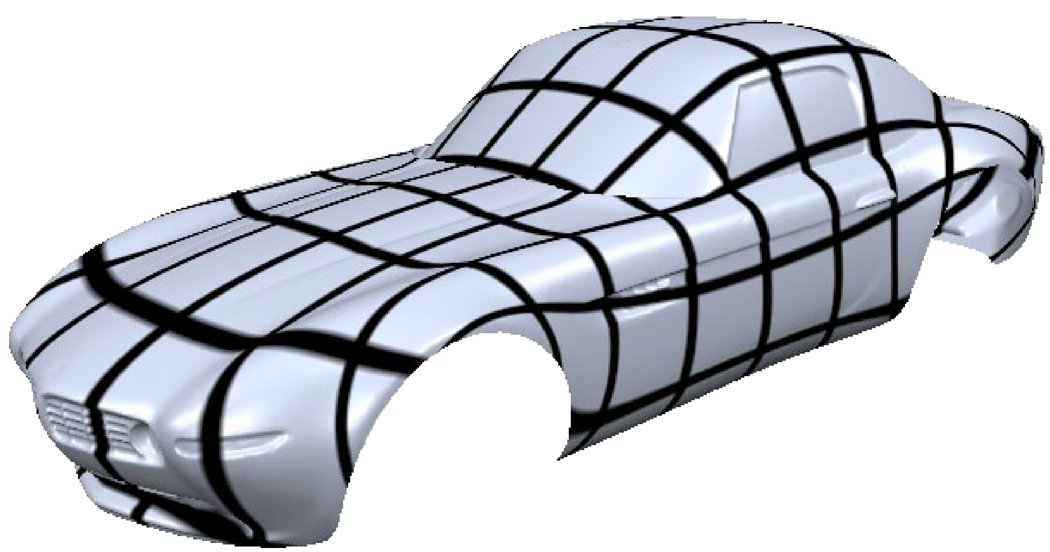


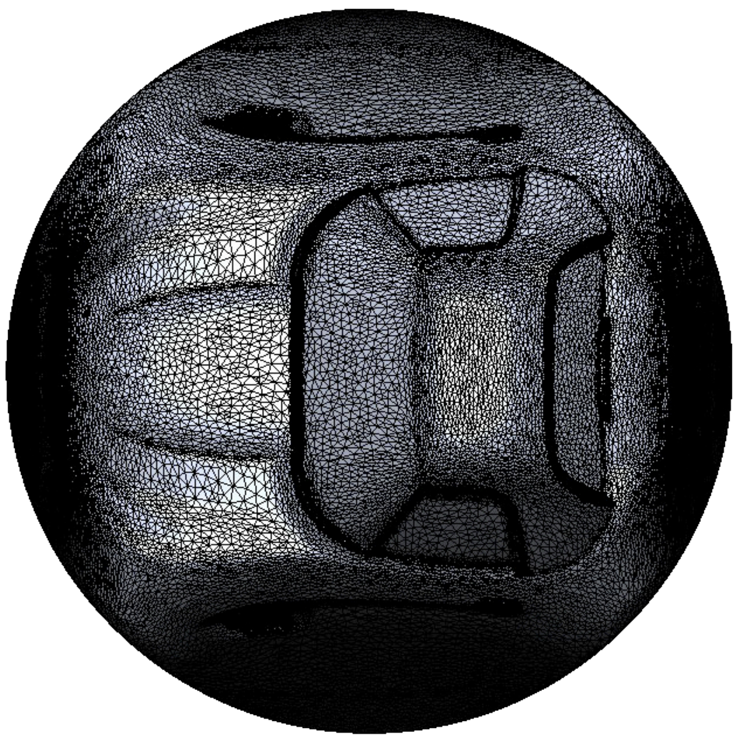
Basic Ingredients

- parametrization
  - global vs. local
- vertex density control
  - uniform vs. adaptive
  - isotropic vs. anisotropic
- local alignment
  - optimal shape approximation
- global alignment
  - feature sensitivity

Mesh Parametrization

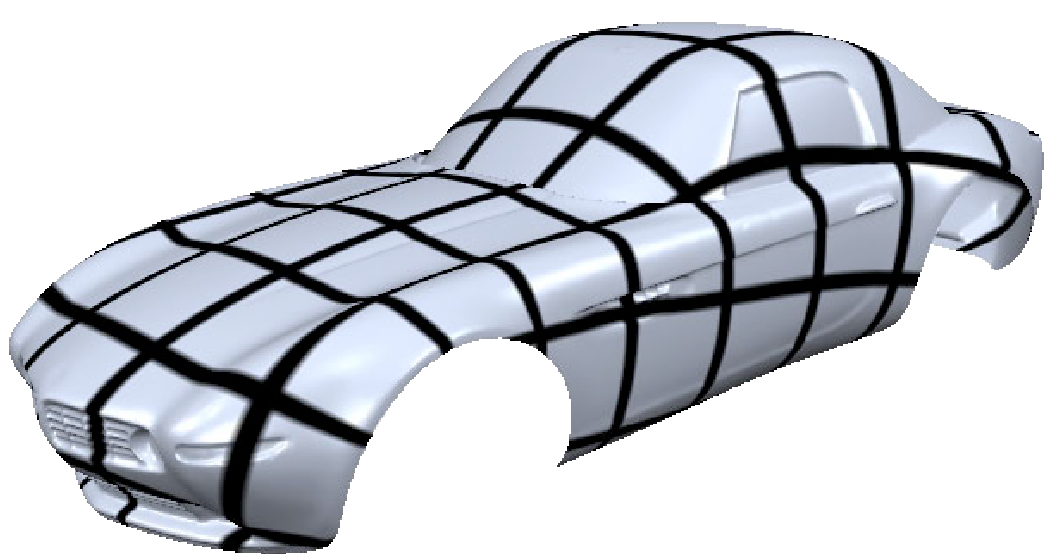
- global parametrization
  - homeomorphic to a disk
  - harmonic maps with fixed boundary conditions
  - least squares conformal maps with free boundaries
  - computationally expensive for large meshes

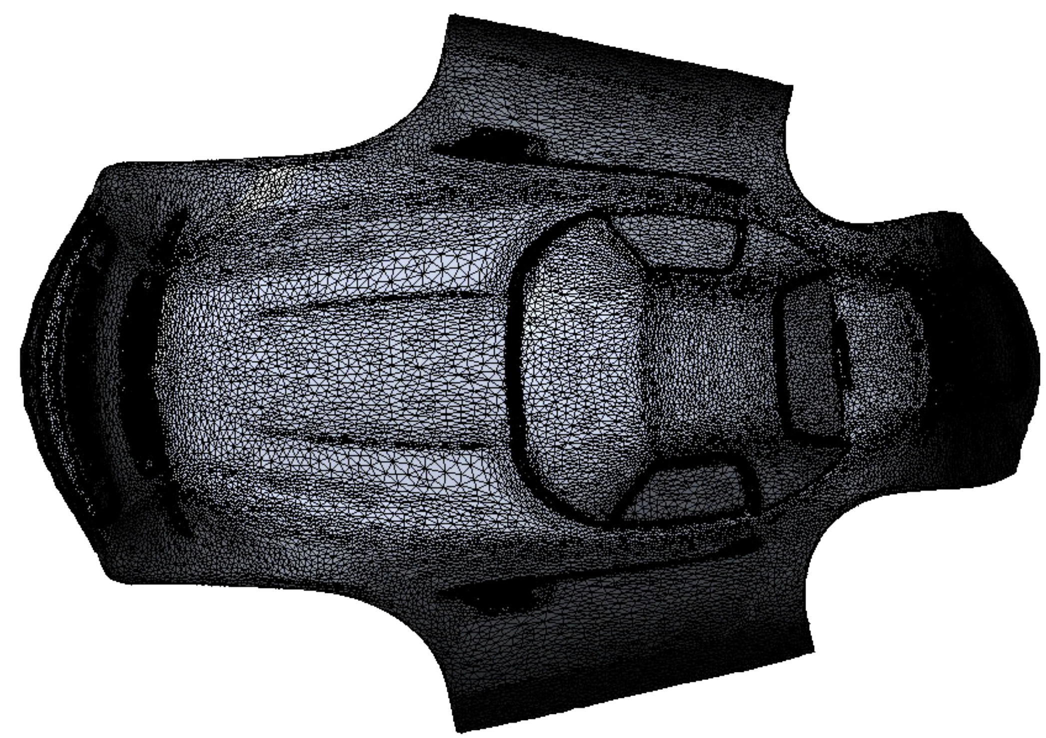




Mesh Parametrization

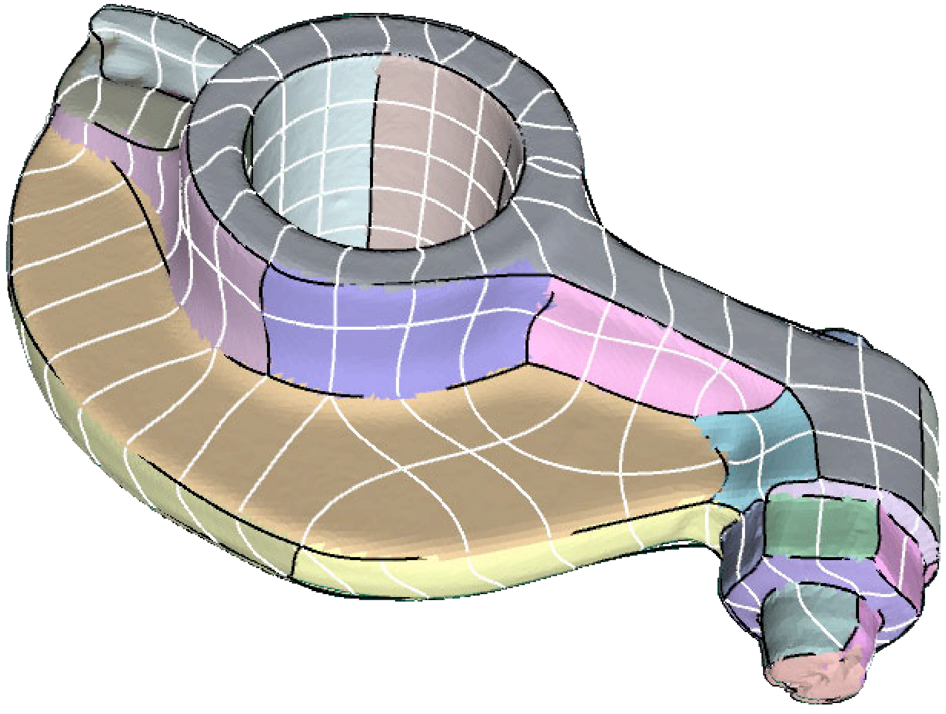
- global parametrization
  - homeomorphic to a disk
  - harmonic maps with fixed boundary conditions
  - least squares conformal maps with free boundaries
  - computationally expensive for large meshes





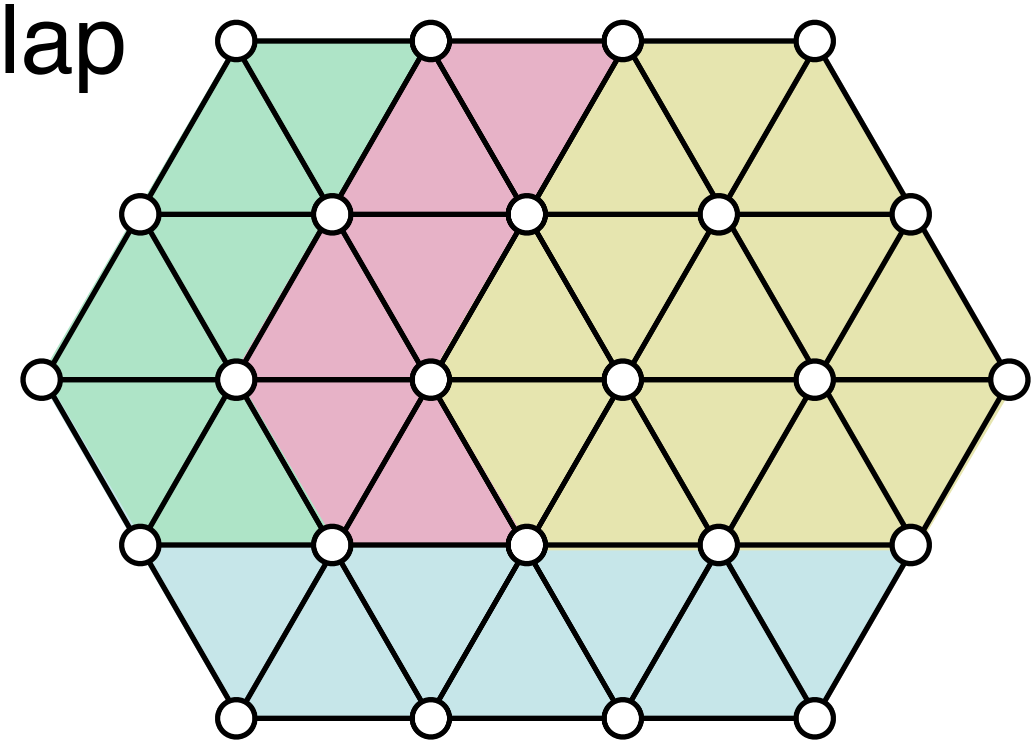
Mesh Parametrization

- global parametrization
- piecewise parametrization
  - pre-segmentation into disjoint patches
  - compatibility conditions at patch boundaries



Mesh Parametrization

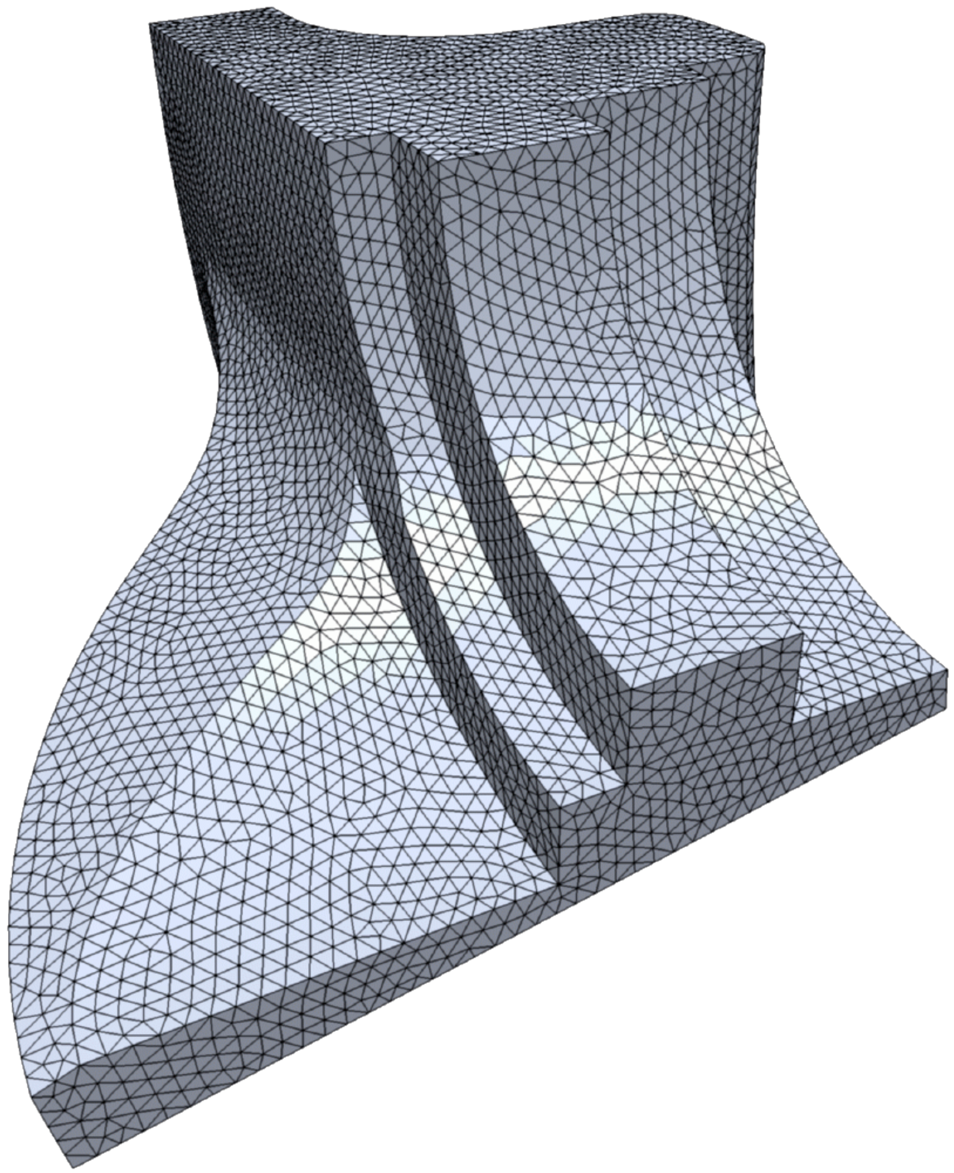
- global parametrization
- piecewise parametrization
- local neighborhood parametrization
  - unfolding a geodesic disk around a vertex / face
  - neighboring regions may overlap
  - efficiency by caching

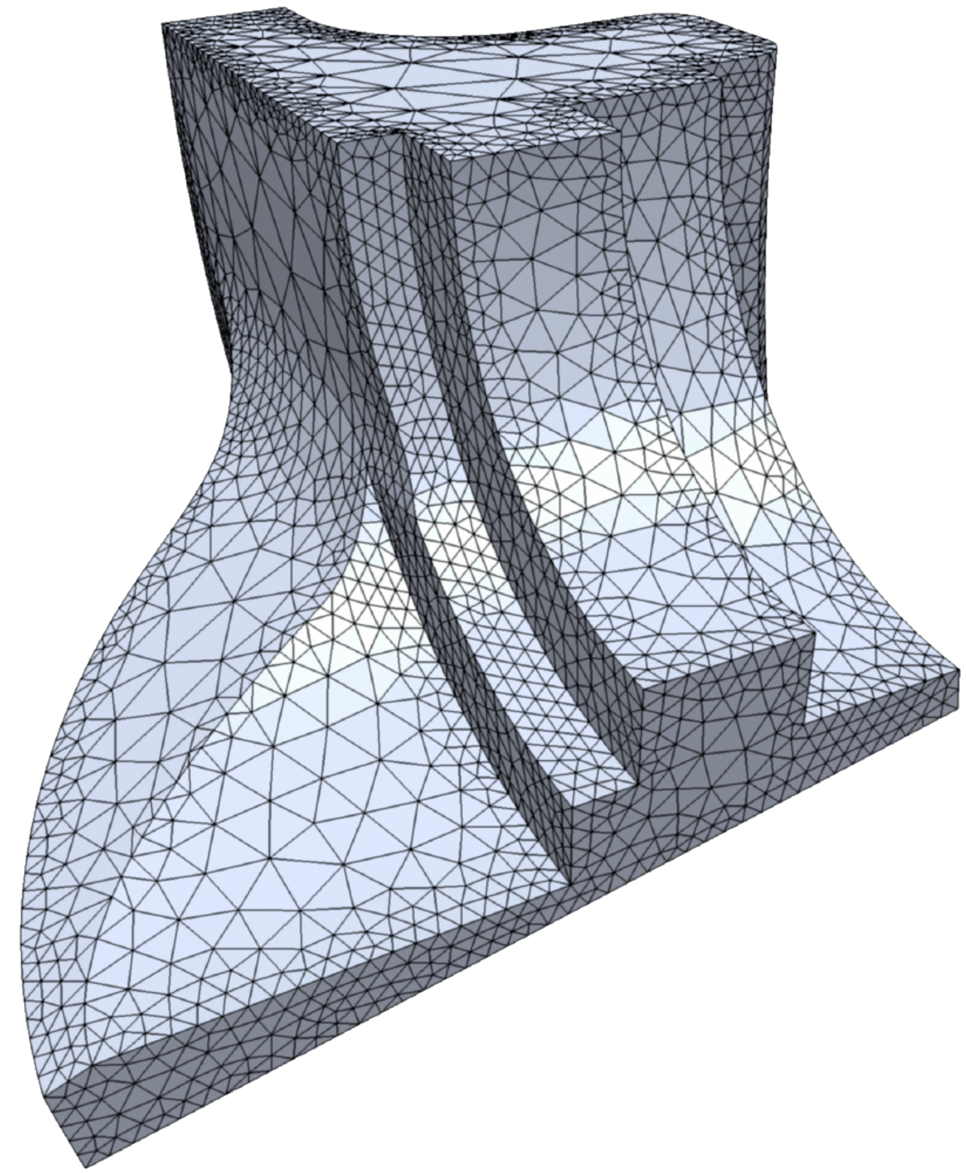


Vertex Density Control

- uniform vs. adaptive
  - curvature-dependent or general sizing map
- isotropic vs. anisotropic
  - 2nd fundamental form
    (error quadrics, shape operator)
- (area weighted) random scatter
- local relaxation
  - particle systems
  - centroidal Voronoi diagrams

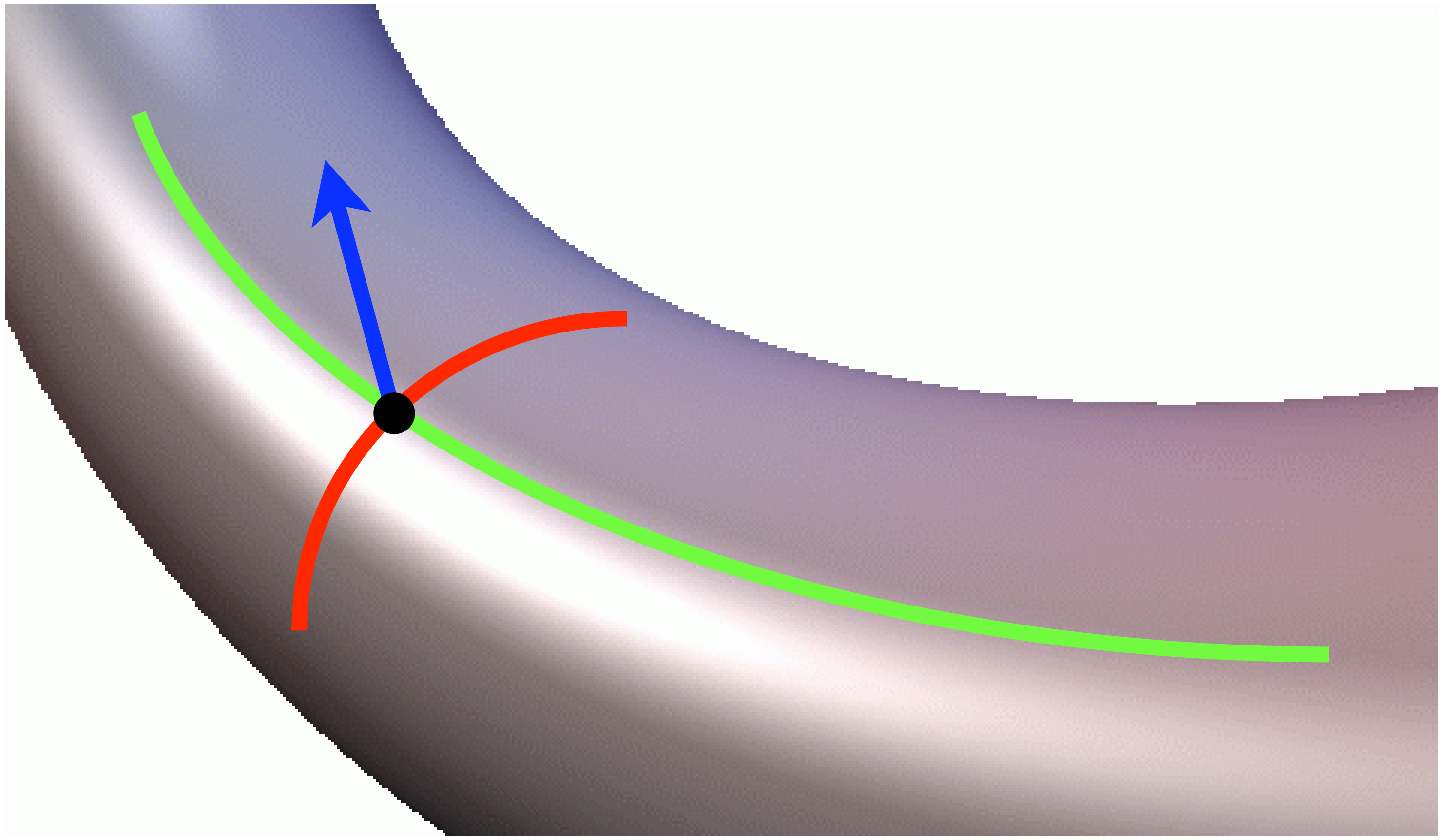
Uniform vs. Adaptive





Anisotropy

- differential geometry
  - 2nd fundamental form defines a local **orthogonal** frame (min- / max-curvature directions and the normal)



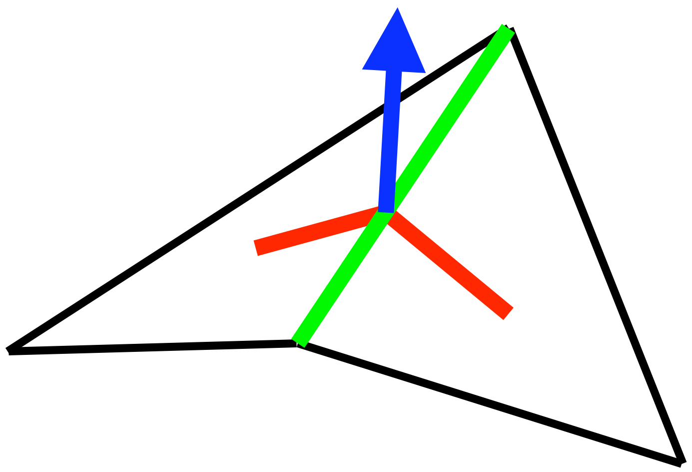
Anisotropy

- differential geometry
  - 2nd fundamental form defines a local **orthogonal** frame (min- / max-curvature directions and the normal)
- discretization
  - eigenbasis of a symmetric matrix
  - shape operator (weighted sum of edge projections)

Shape Operator

- projection to edge: $\mathbf{e}\,\mathbf{e}^T \quad \|\mathbf{e}\| = 1$
  (minimum curvature direction)
- weighted sum of edge-projection operators

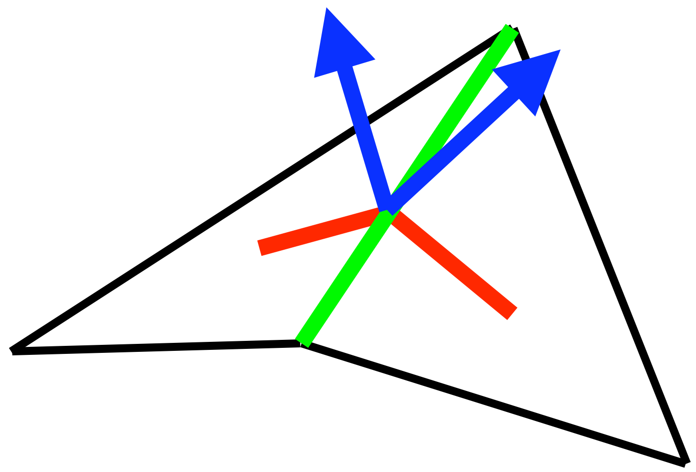
$$\mathcal{S}(\mathbf{p}) = \sum_{\mathbf{e} \in B(\mathbf{p})} \|\mathbf{e} \cap B(\mathbf{p})\| \, \mathbf{e}\,\mathbf{e}^T$$



Shape Operator

- projection to edge: $\mathbf{e}\,\mathbf{e}^T \quad \|\mathbf{e}\| = 1$
  (minimum curvature direction)
- weighted sum of edge-projection operators

$$\mathcal{S}(\mathbf{p}) = \sum_{\mathbf{e}\in B(\mathbf{p})} \beta(\mathbf{e})\,\|\mathbf{e}\cap B(\mathbf{p})\|\,\mathbf{e}\,\mathbf{e}^T$$



Shape Operator

- projection to edge: $\mathbf{e}\,\mathbf{e}^T$
  (minimum curvature direction)
- weighted sum of edge-projection operators

$$\mathcal{S}(\mathbf{p}) \;=\; \sum_{\mathbf{e}\in B(\mathbf{p})} \beta(\mathbf{e})\,\|\mathbf{e}\cap B(\mathbf{p})\|\,\mathbf{e}\,\mathbf{e}^T$$

- eigenvector to largest eigenvalue:
  min-curvature direction
- max-curvature direction: $D_{\max} = D_{\min} \times \mathbf{n}$

Random Scatter

- generate random samples for each triangle
  - n ~ area * density
  - prob = area * density * $\dfrac{\text{total number}}{\text{total area * density}}$
- compensate area distortion when sampling in the parameter domain
  - distortion = 3D area / 2D area
- no anisotropy

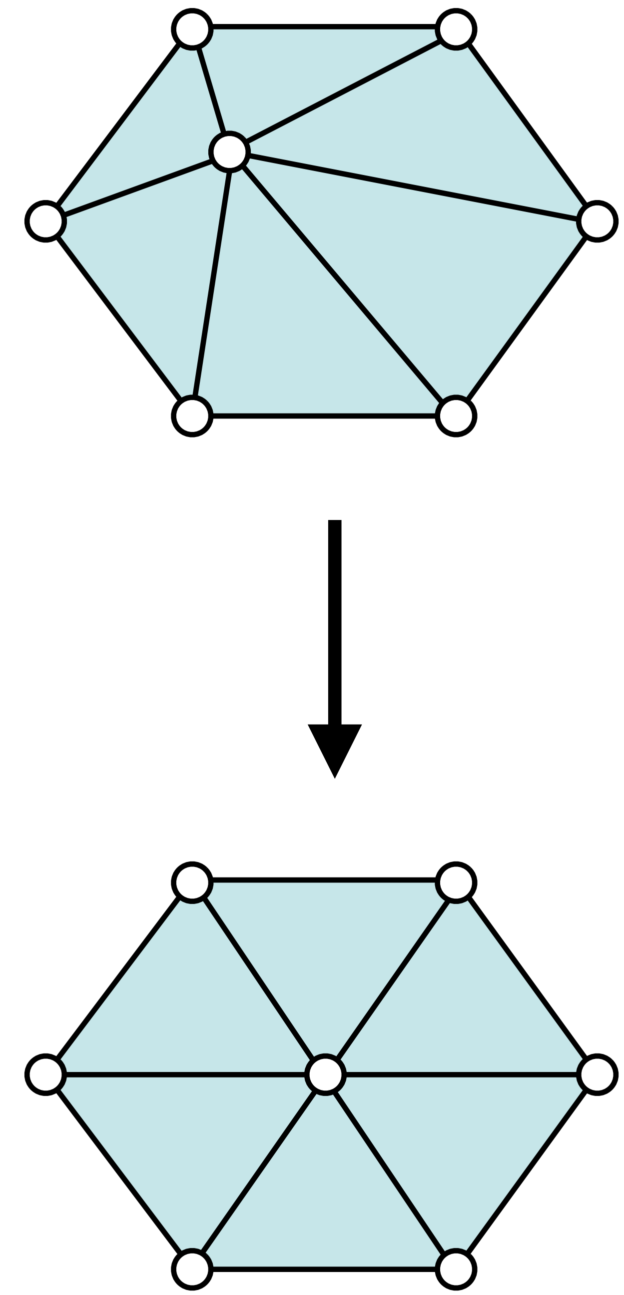
Local Relaxation

- particle systems
  - maximum distance (repelling force)
  - curvature dependent
  - anisotropic forces
- tangential Laplace
  - minimum distance (attracting force)
- centroidal Voronoi diagrams

Tangential Laplace

- local "spring" relaxation
  - uniform Laplacian smoothing
  - barycenter of one-ring neighbors

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



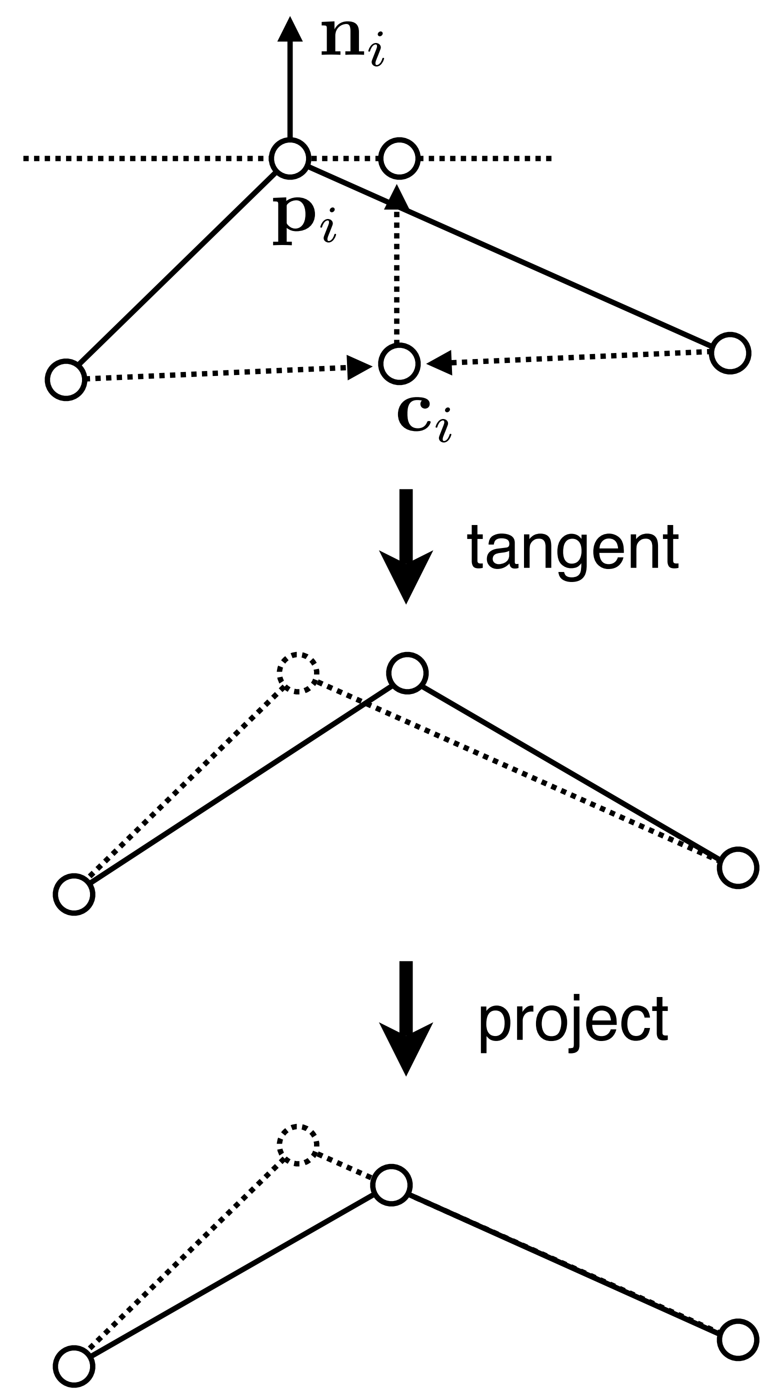
Tangential Laplace

- local “spring” relaxation
  - uniform Laplacian smoothing
  - barycenter of one-ring neighbors

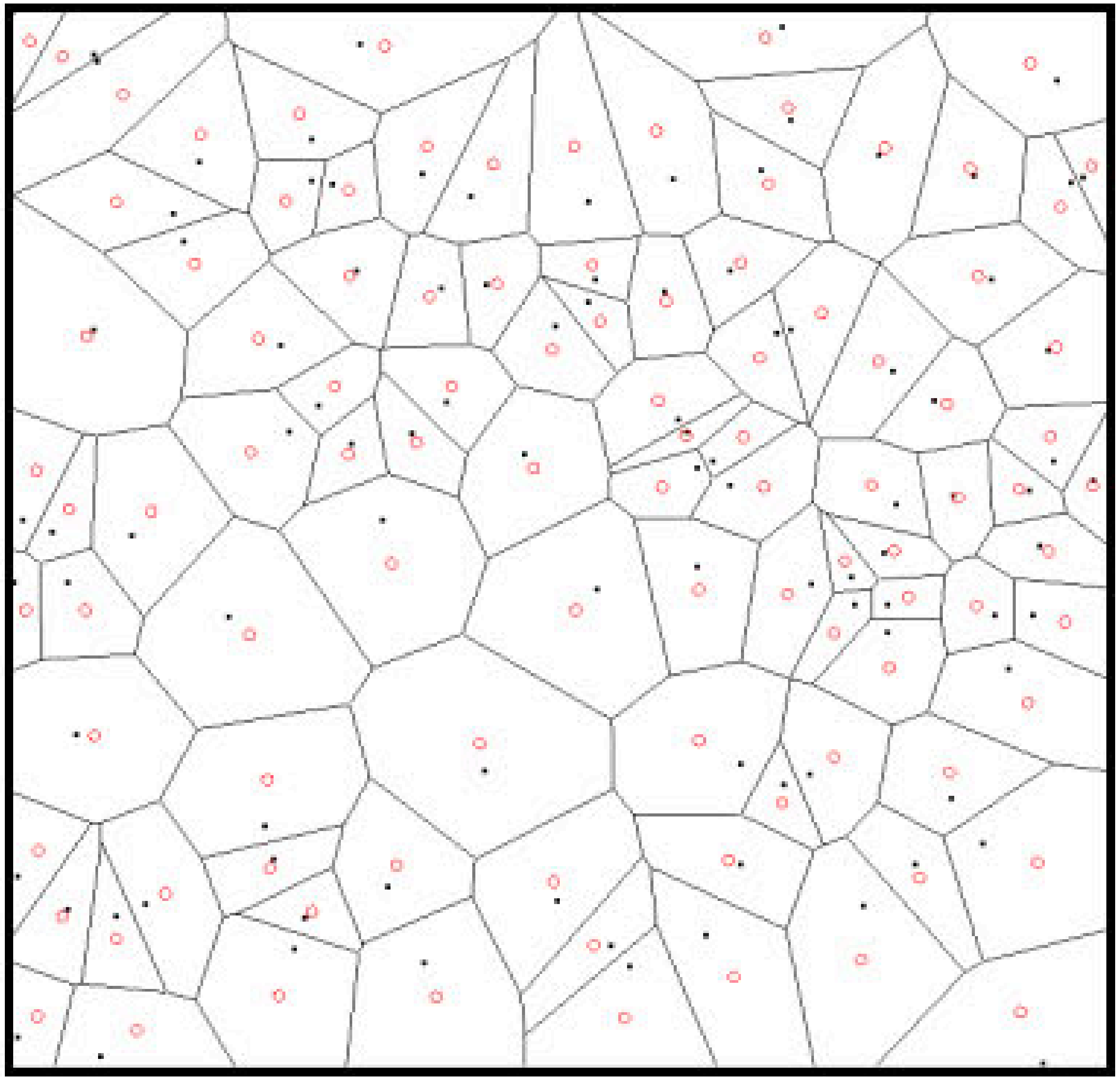
$$\mathbf{c}_i = \frac{1}{\text{valence}\,(v_i)} \sum_{j \in N(v_i)} \mathbf{p}_j$$

- keep vertex on the surface
  - restrict movement to tangent plane

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

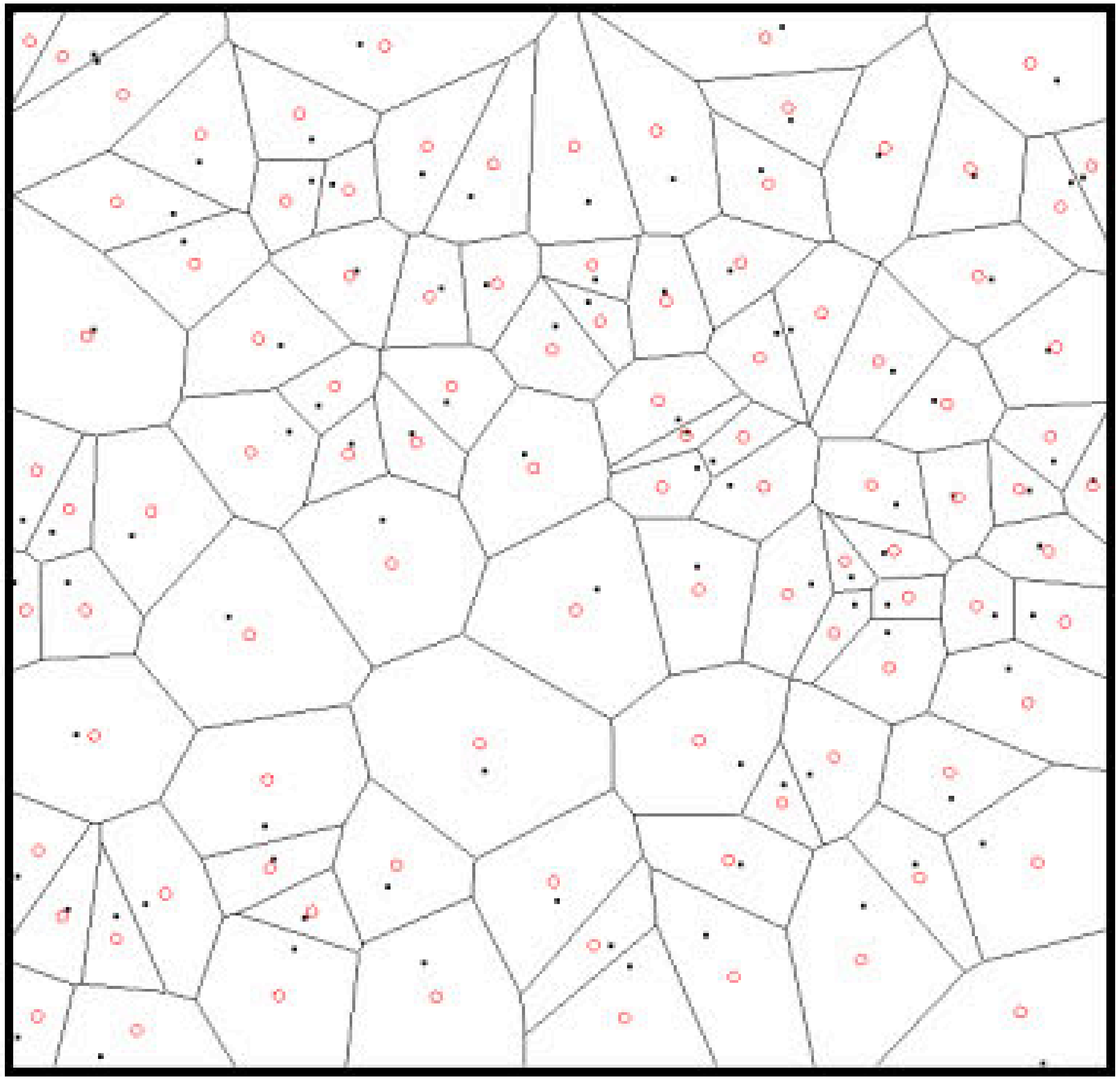


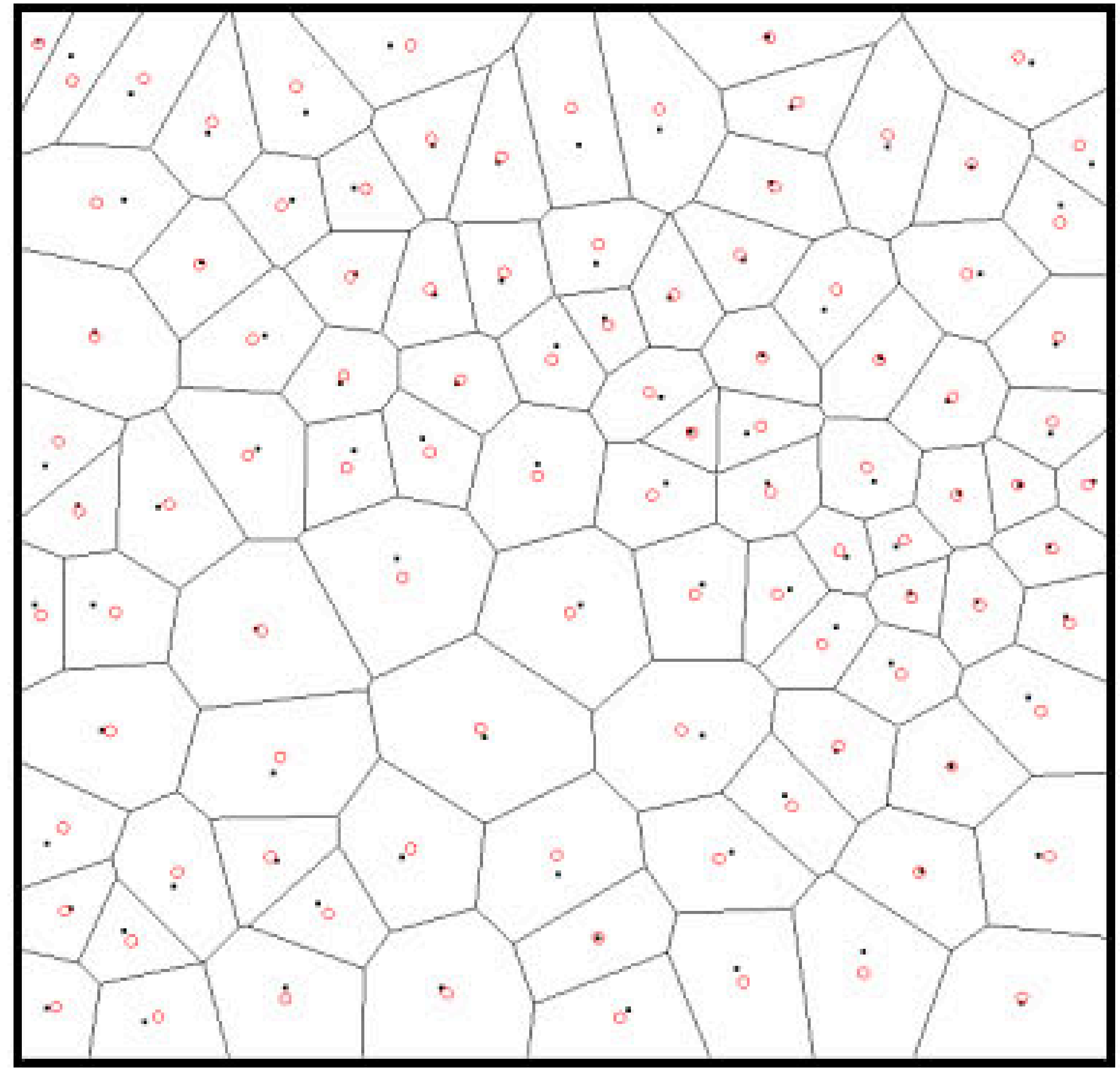
Centroidal Voronoi Diagrams



[Alliez et al. *“Recent Advances in Remeshing of Surfaces”*]

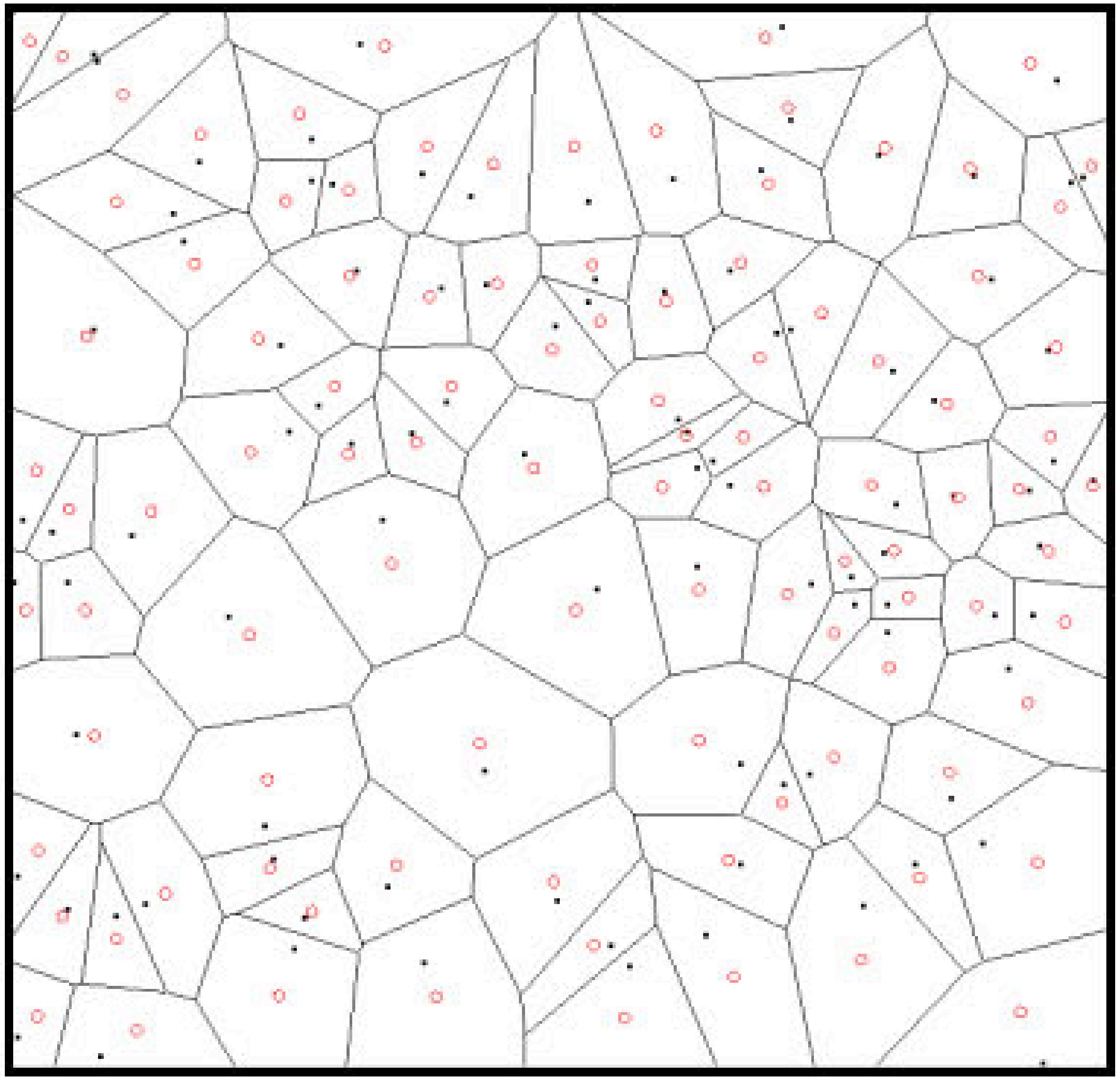
Centroidal Voronoi Diagrams

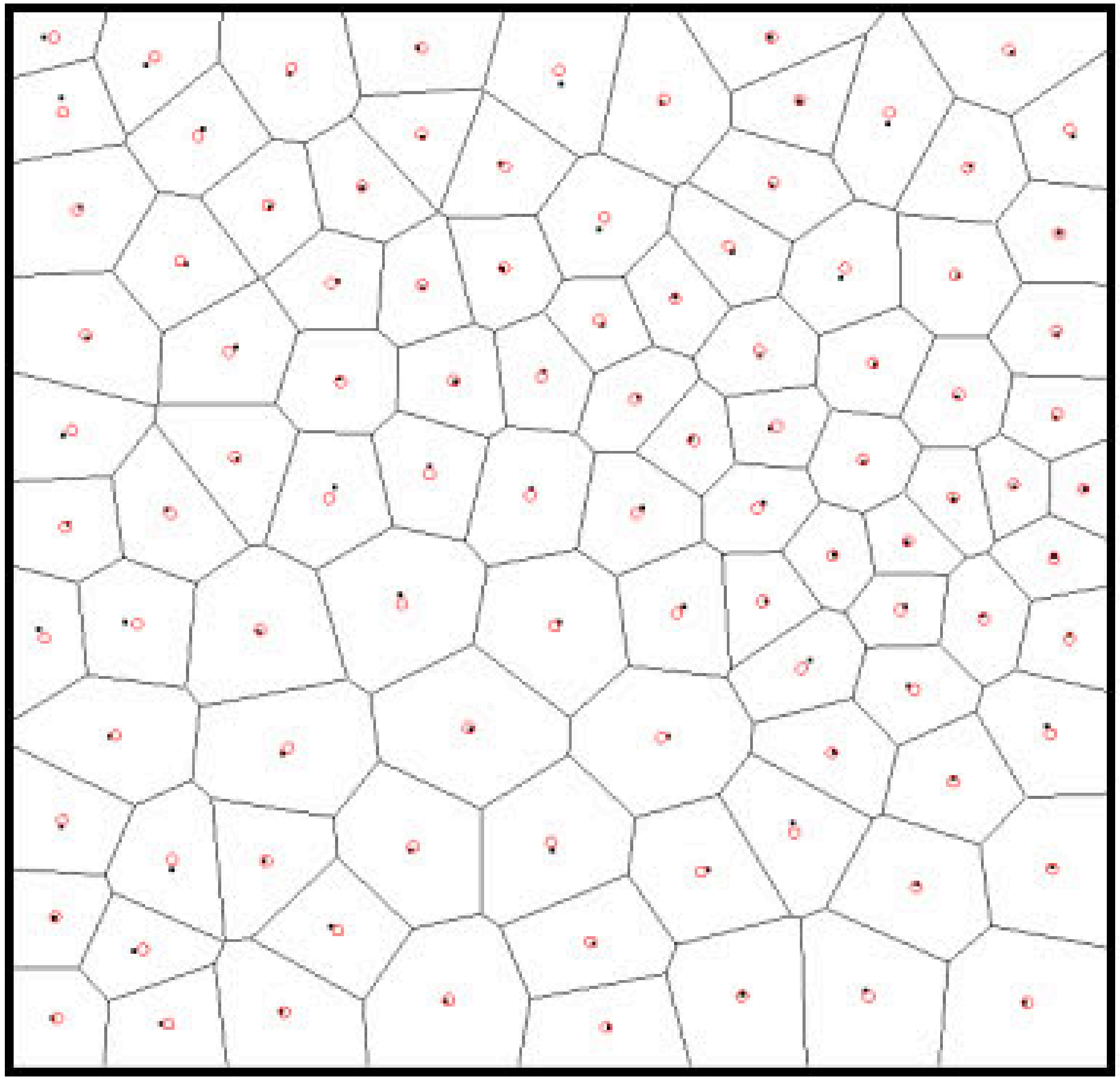




[Alliez et al. *“Recent Advances in Remeshing of Surfaces”*]

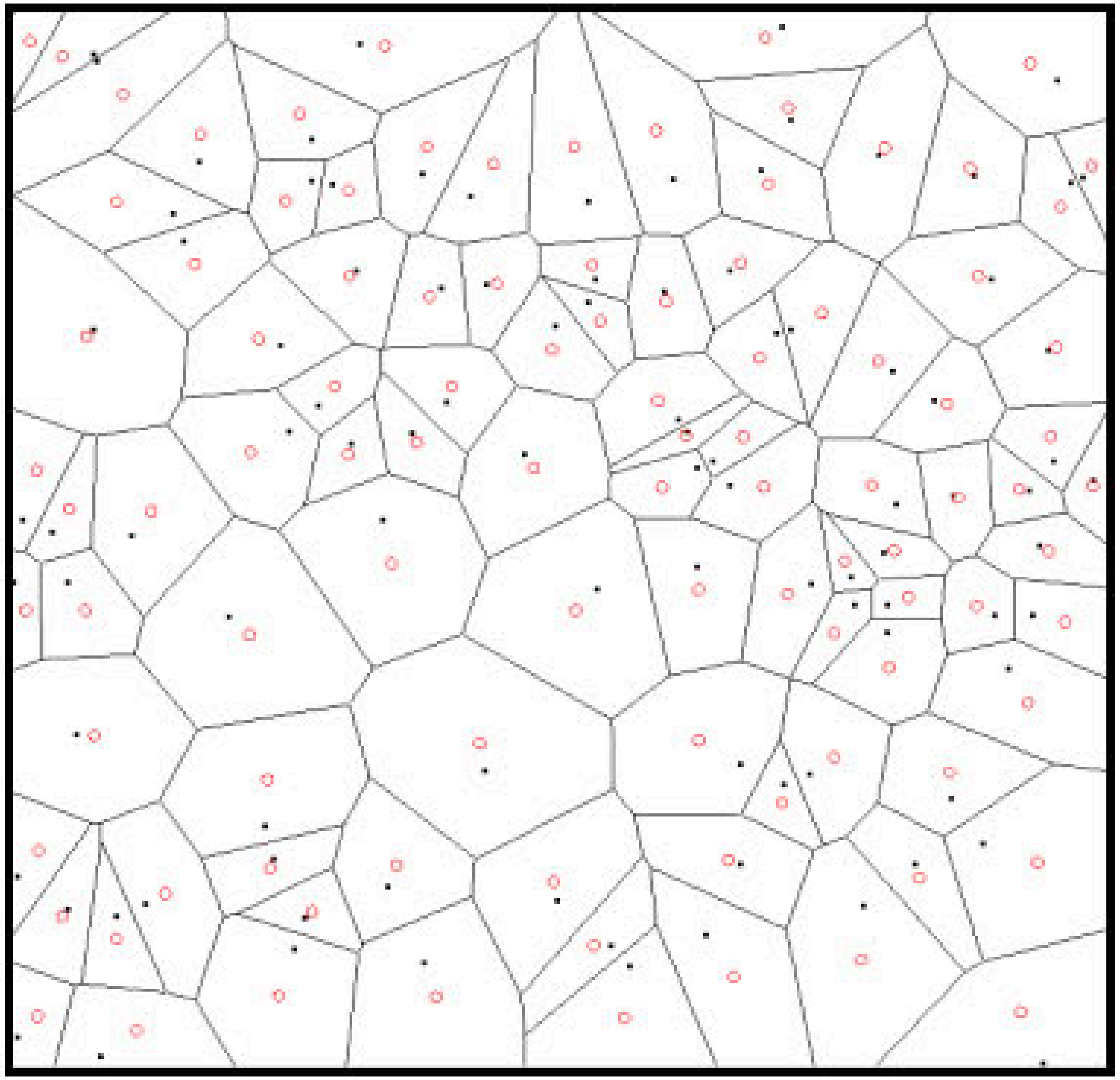
Centroidal Voronoi Diagrams

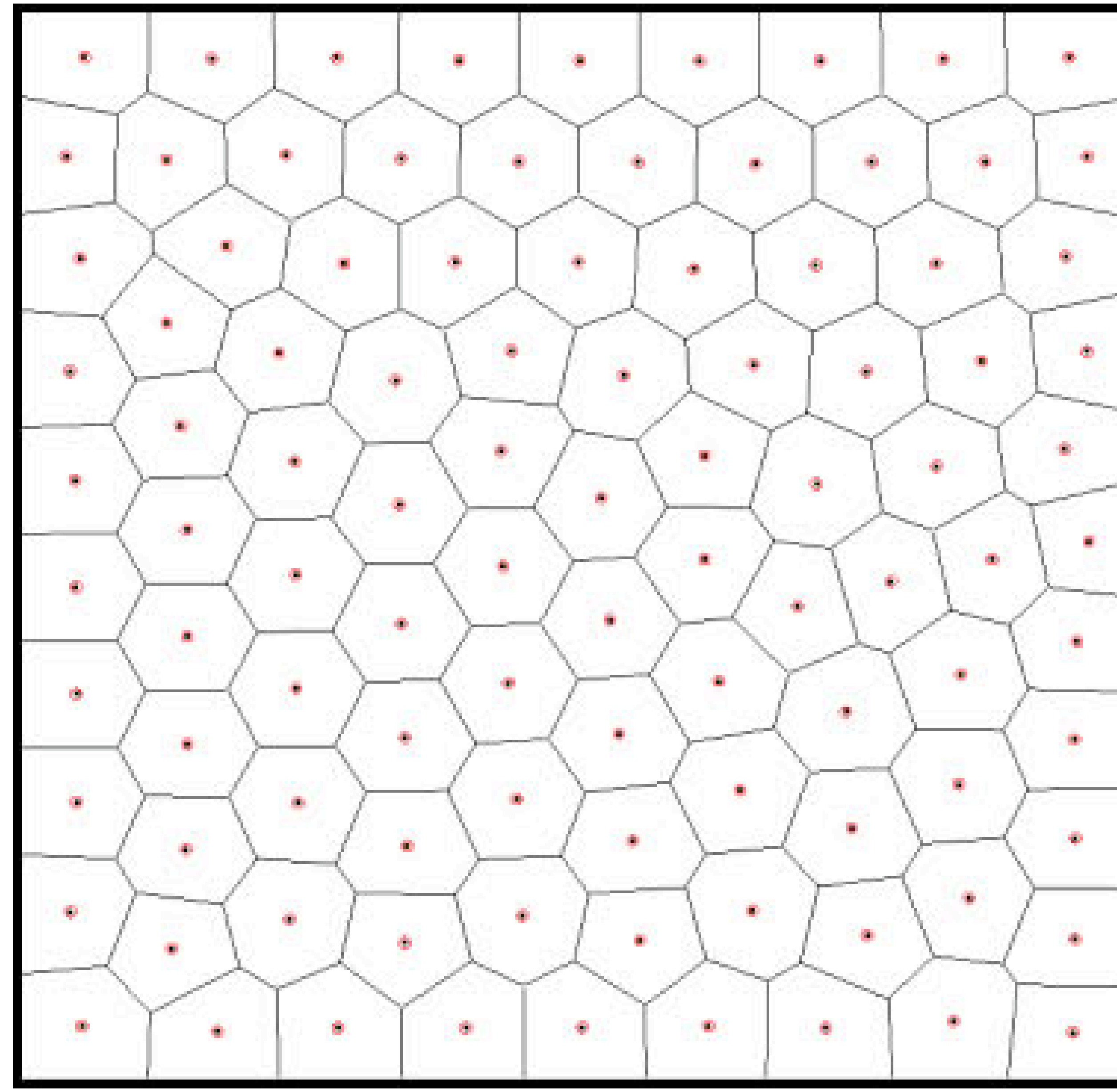




[Alliez et al. *"Recent Advances in Remeshing of Surfaces"*]

Centroidal Voronoi Diagrams





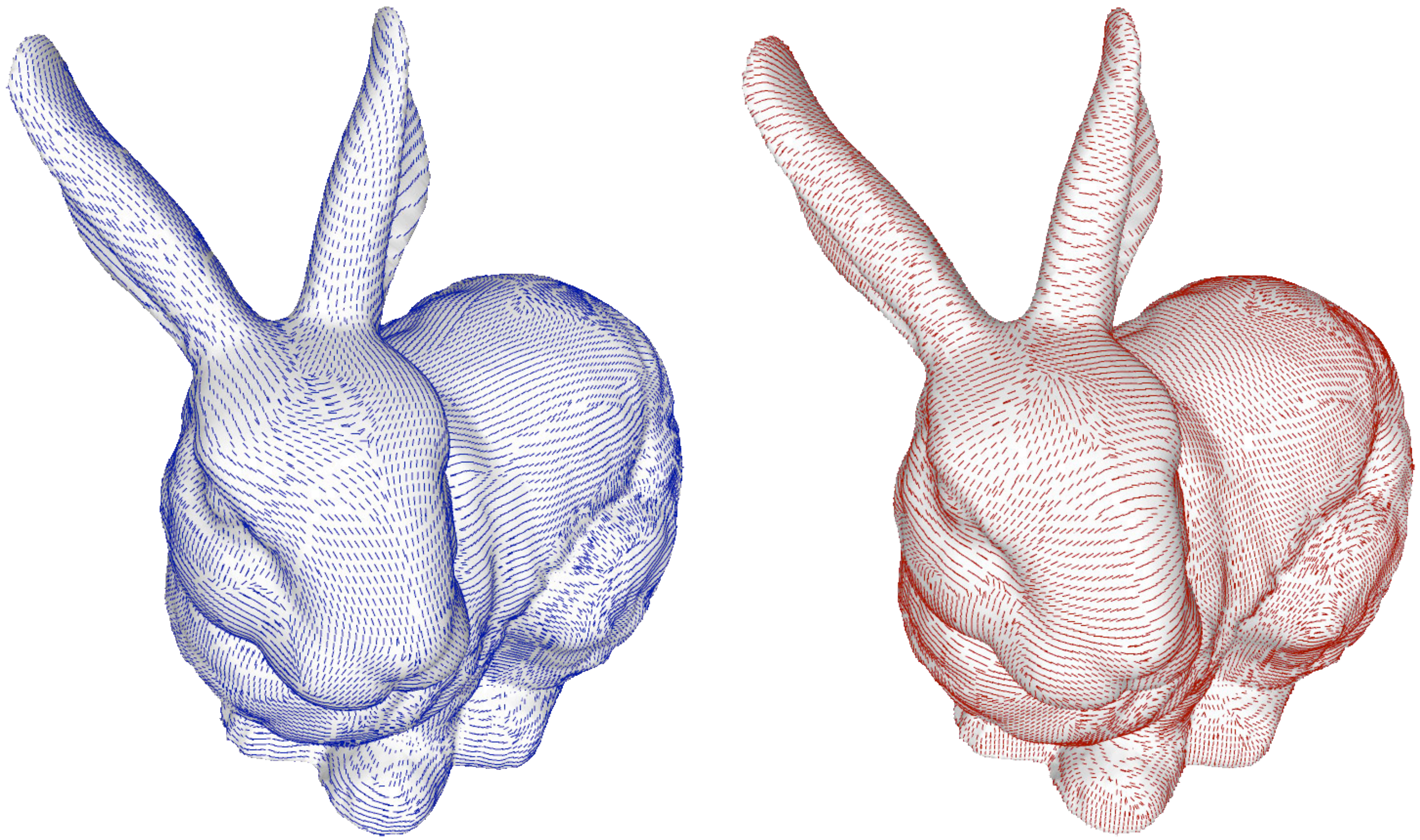
[Alliez et al. *"Recent Advances in Remeshing of Surfaces"*]

Local Alignment

- compute curvature directions fields
  - smoothing filters
  - preserve orthogonality
- trace curvature lines
  - in the parameter domain
  - directly on the polygonal surface
  - face aspect ratio, line density
- vector field integration
  - extract iso-contours

Curvature Directions

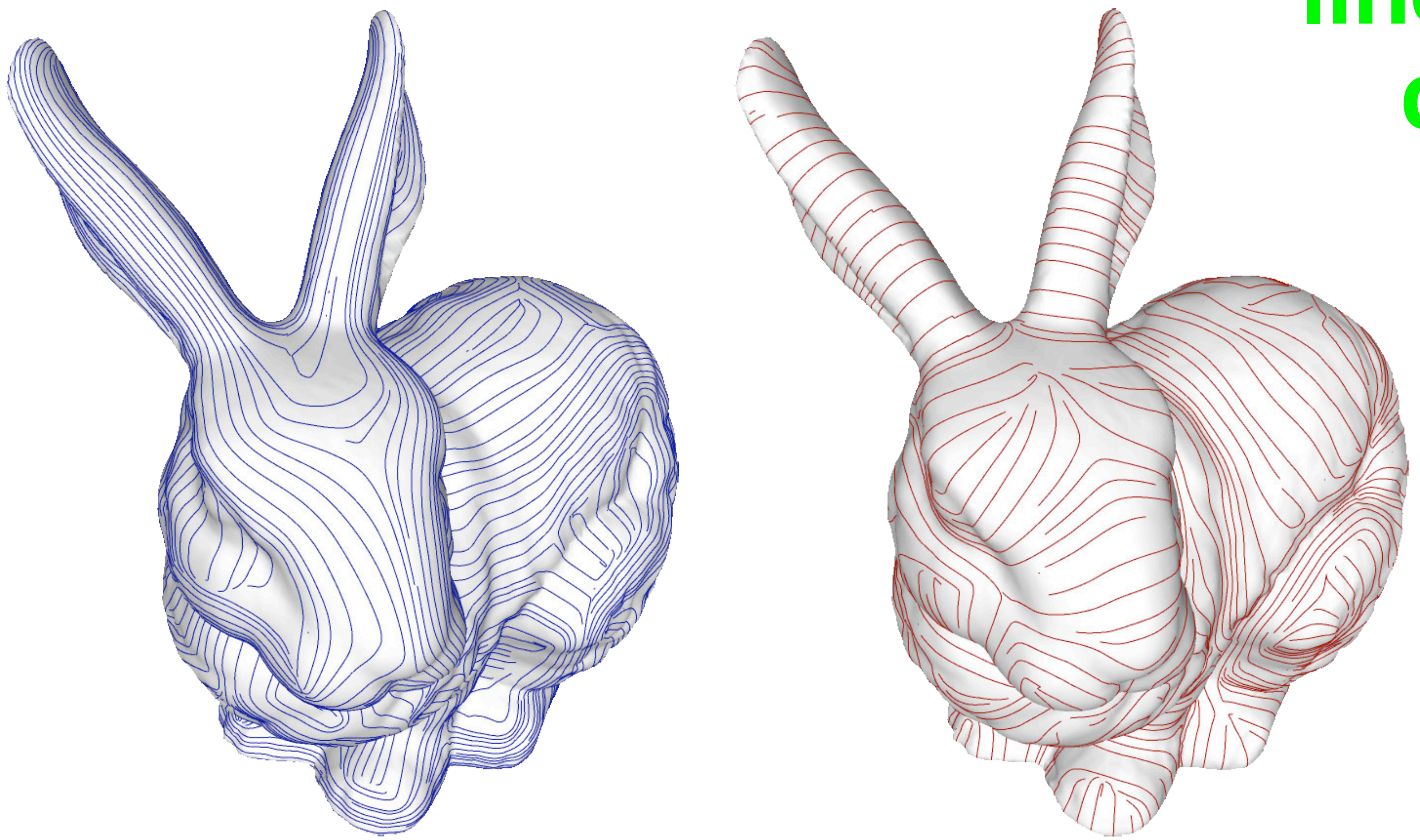
- shape operator
  - tensor averaging preserves orthogonality
  - smoothing *within* the tangent plane



Curvature Directions

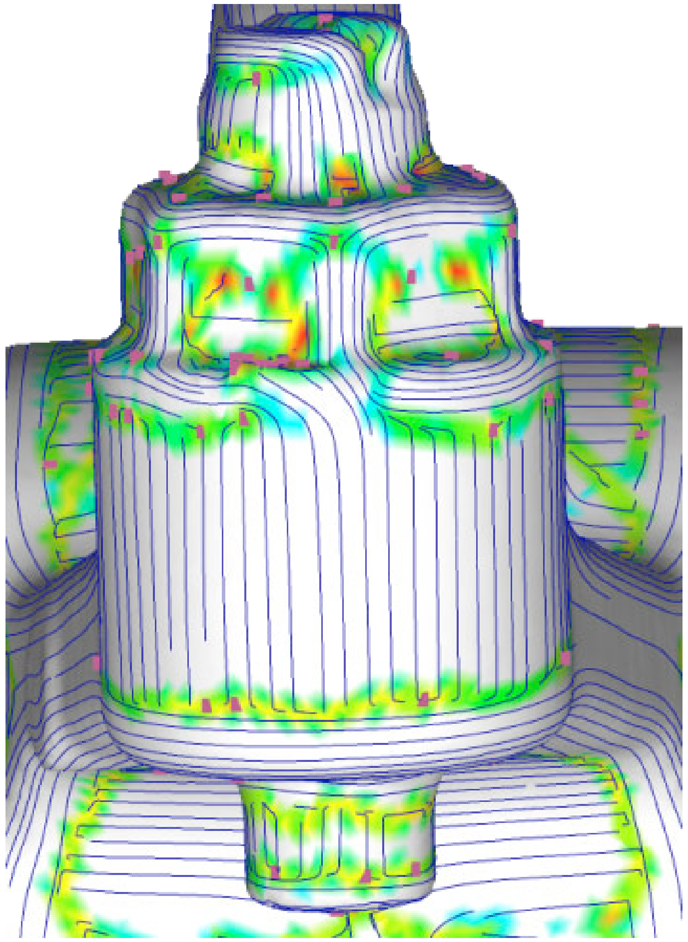
- shape operator
  - tensor averaging preserves orthogonality
  - smoothing *within* the tangent plane

line density control



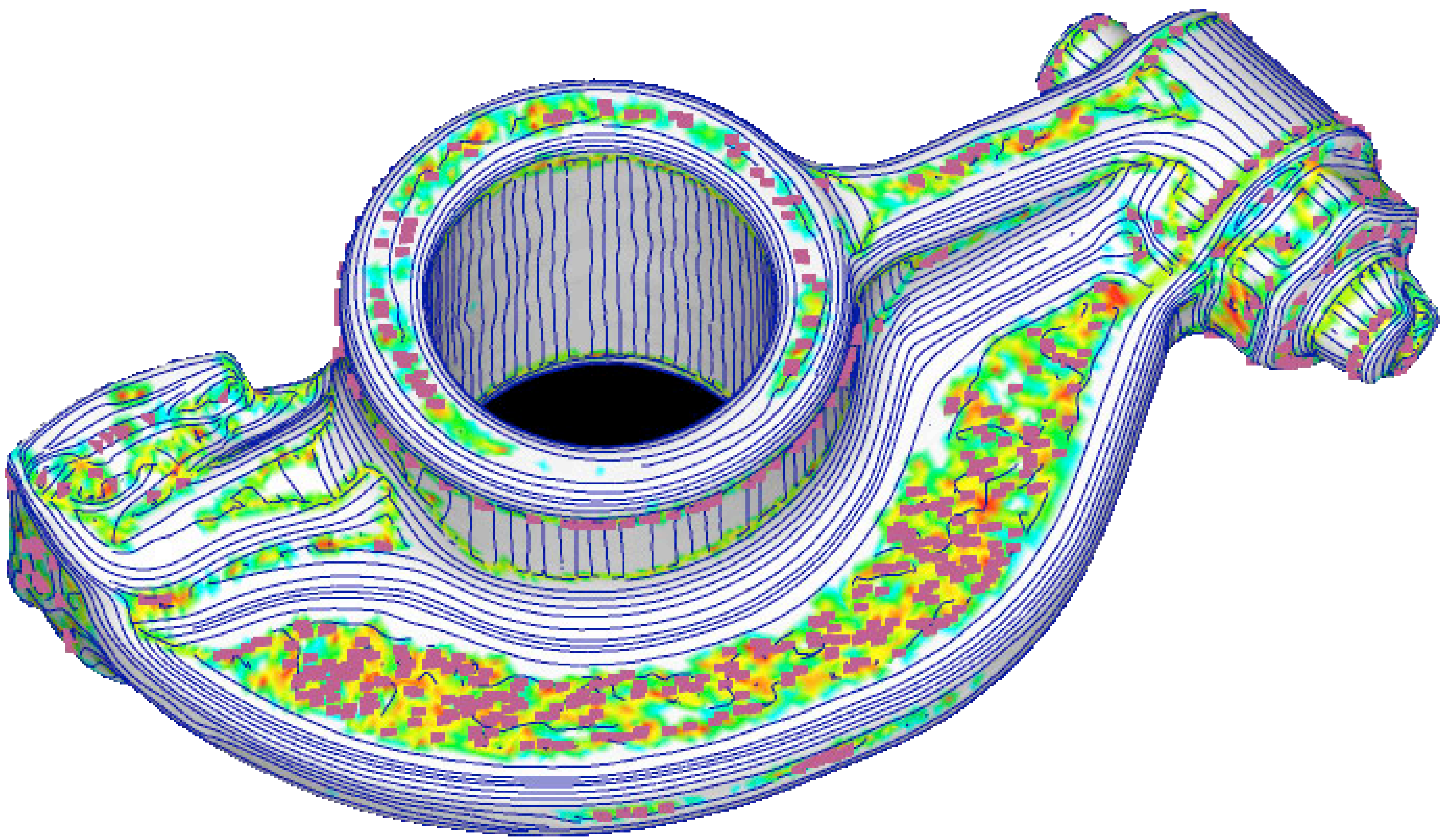
Curvature Directions

- shape operator
  - tensor averaging preserves orthogonality
  - smoothing *within* the tangent plane



Curvature Directions

- shape operator
  - tensor averaging preserves orthogonality
  - smoothing *within* the tangent plane



Curvature Directions

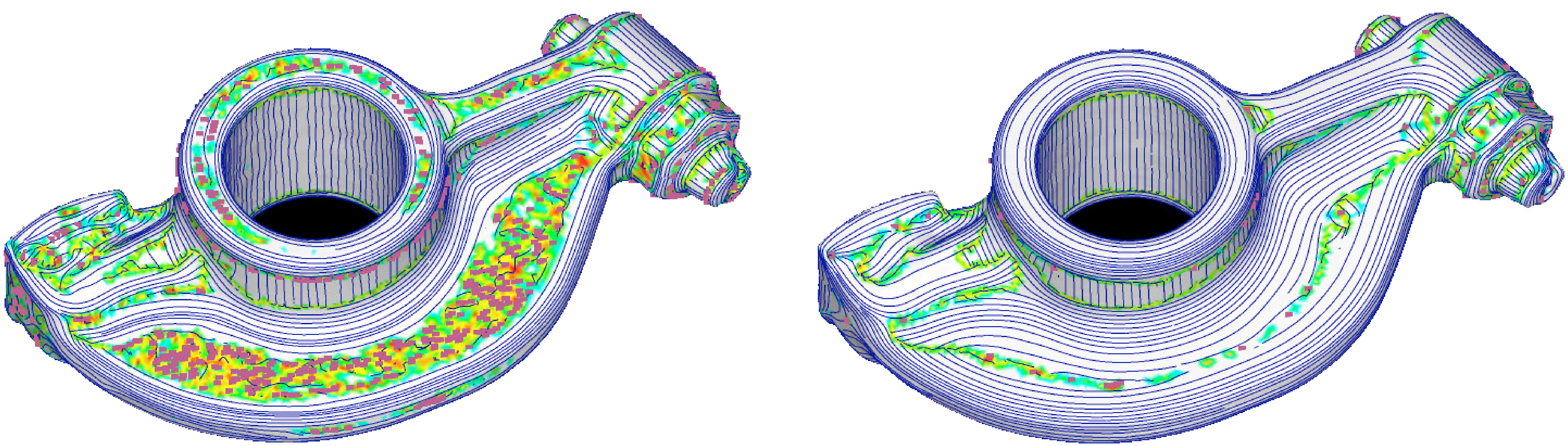
- shape operator
  - tensor averaging preserves orthogonality
  - smoothing *within* the tangent plane
  - propagate *reliable* direction information

$$(\mathcal{C}, \rho) \; = \; \sum_{j \in N(i)} \omega_{ij} \, (\mathcal{P}_j \, , \rho_j \,)$$

$$\mathcal{P}_i \; \leftarrow \; \frac{\rho_i \, \mathcal{P}_i + \rho \, \mathcal{C}}{\rho_i + \rho}$$

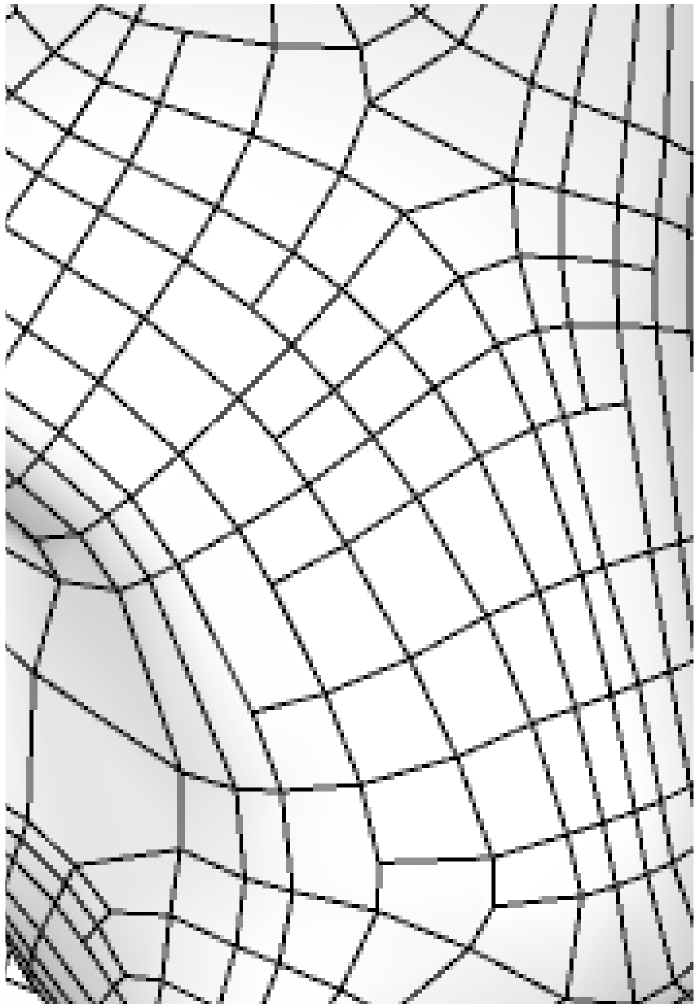
Curvature Directions

- shape operator
  - tensor averaging preserves orthogonality
  - smoothing *within* the tangent plane
  - propagate *reliable* direction information



Vector Field Integration

- curvature lines are traced independently (only line density is controlled, no synchronization)
- curves don't match
- T-vertices are generated
- iso-contours of scalar fields are always closed ...



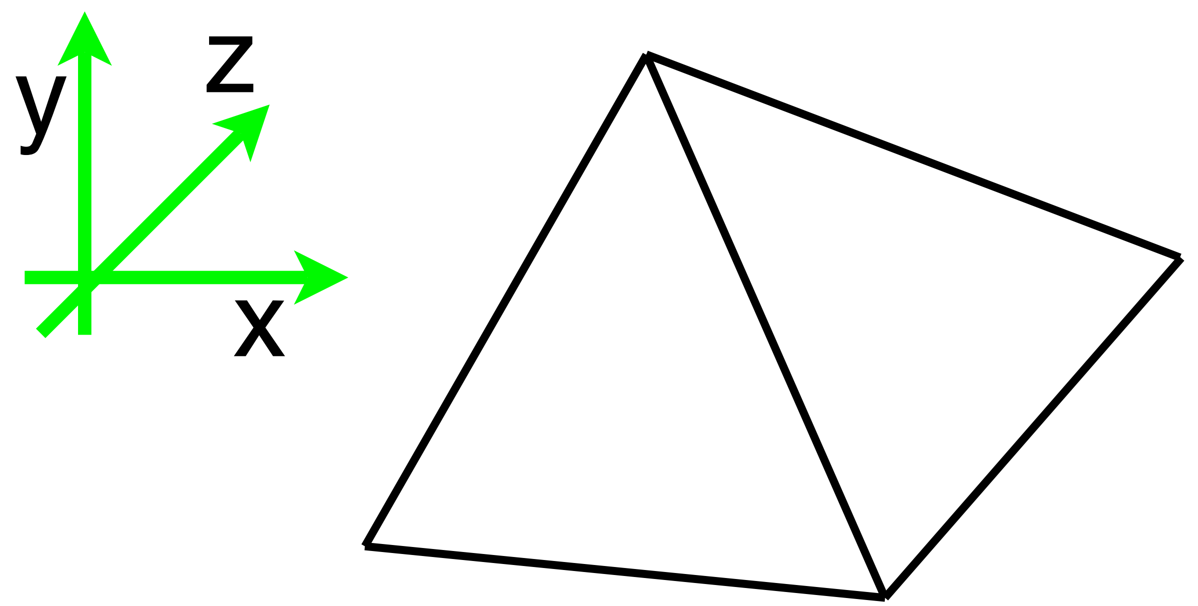
Vector Field Integration

- given curvature direction fields K_{min} and K_{max}
- compute (inverse) parameter functions u and v such that locally
  - $\boxed{\nabla u = \lambda K_{min}}$ and $\boxed{\nabla v = \lambda K_{max}}$ (or vice versa)
- then the iso-contours of u are aligned to K_{max} and the iso-contours of v to K_{min} (or vice versa)
- in general no globally continuous solution possible
  - translational and rotational discontinuities

Vector Field Integration

- "periodic coordinates" [Ray et al.]

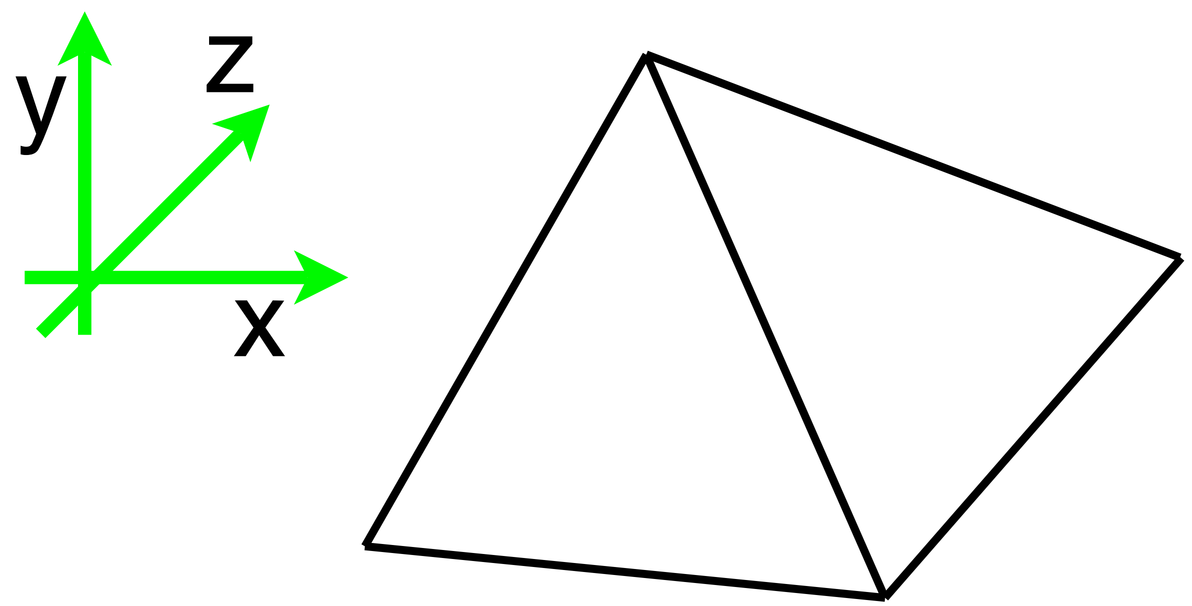
$$\tilde{\mathbf{u}} = (\cos u, \sin u)$$
$$\tilde{\mathbf{v}} = (\cos v, \sin v)$$

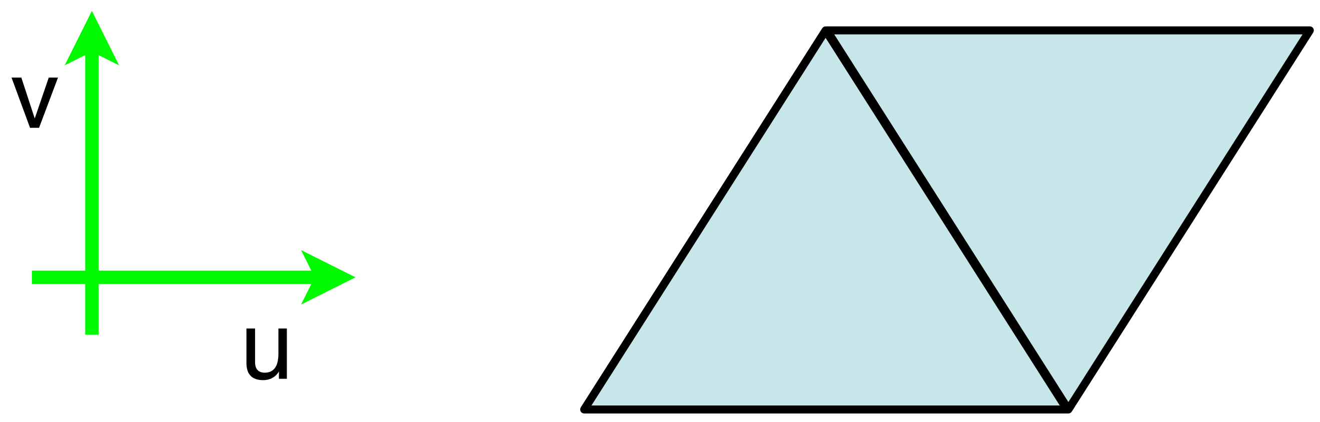


Vector Field Integration

- "periodic coordinates" [Ray et al.]

$$\tilde{\mathbf{u}} = (\cos u, \sin u)$$
$$\tilde{\mathbf{v}} = (\cos v, \sin v)$$

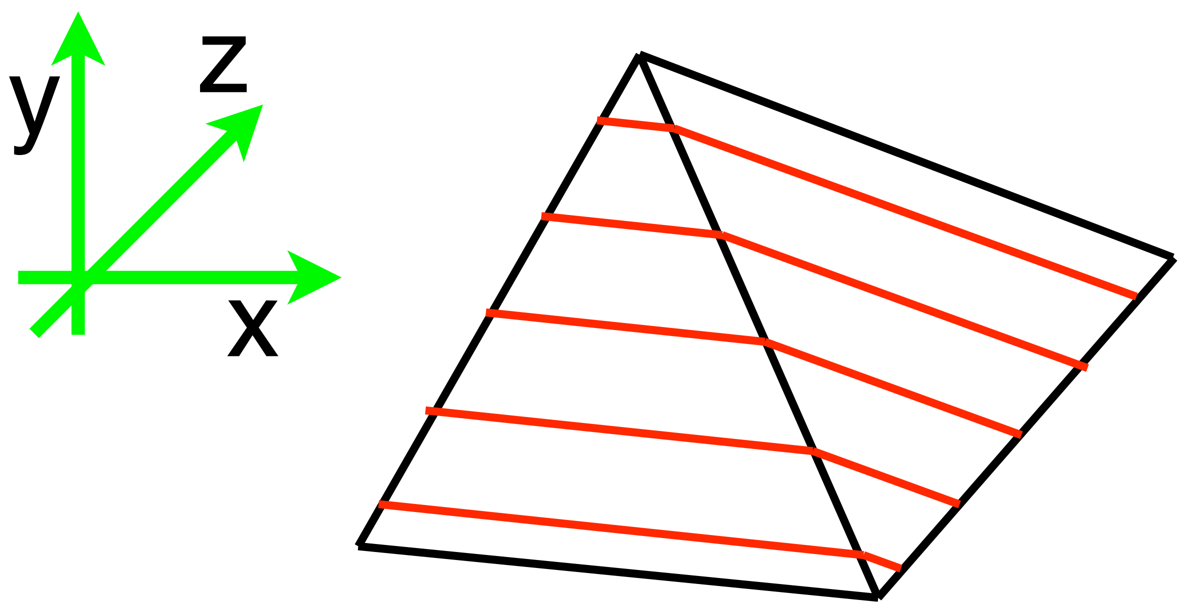


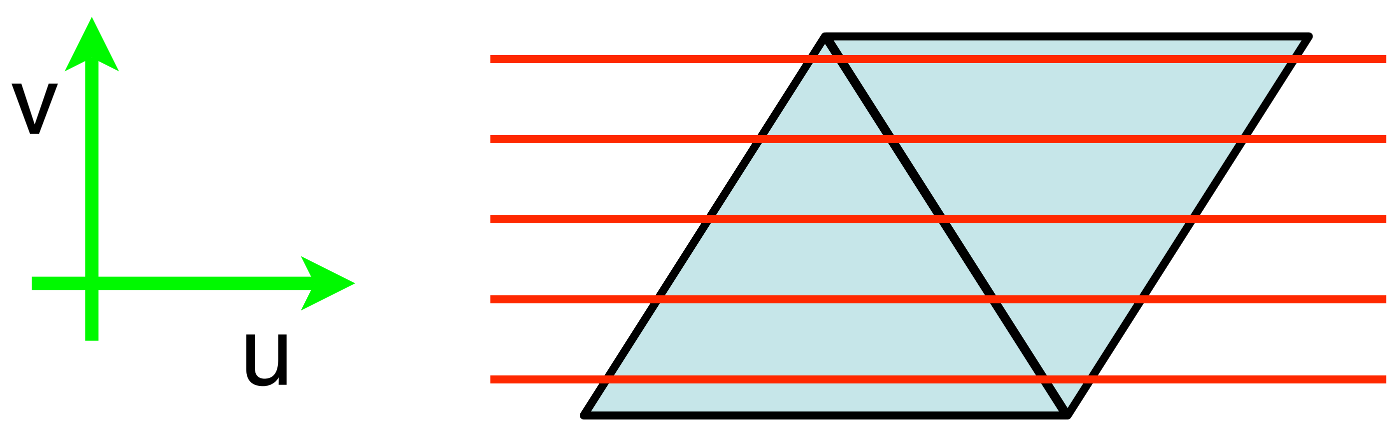


Vector Field Integration

- "periodic coordinates" [Ray et al.]

$$\tilde{\mathbf{u}} = (\cos u, \sin u)$$
$$\tilde{\mathbf{v}} = (\cos v, \sin v)$$

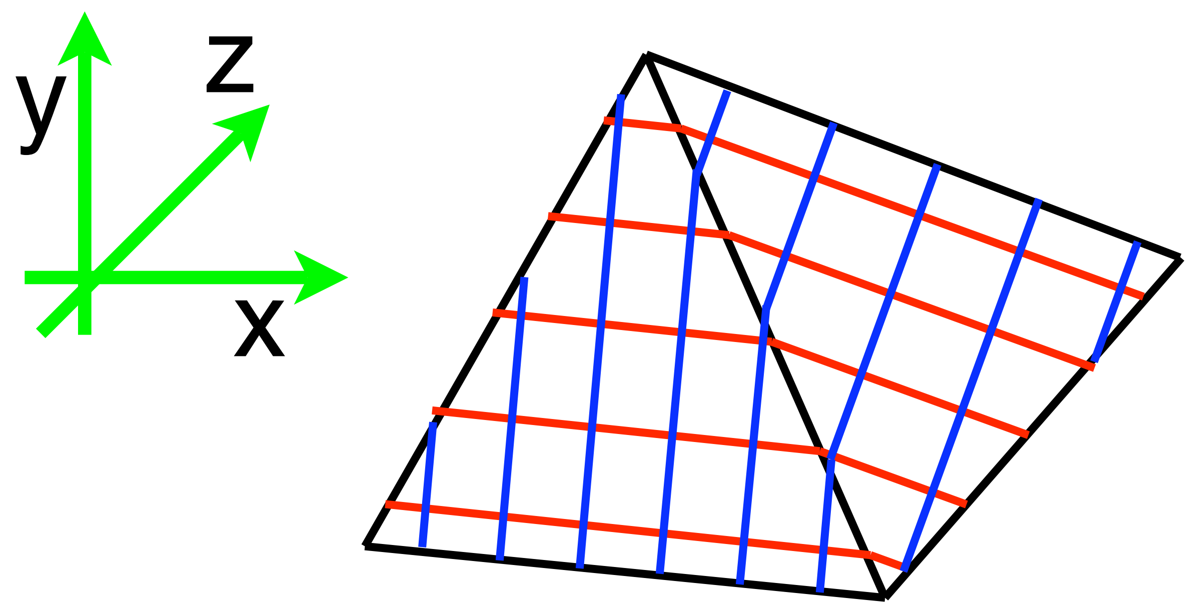


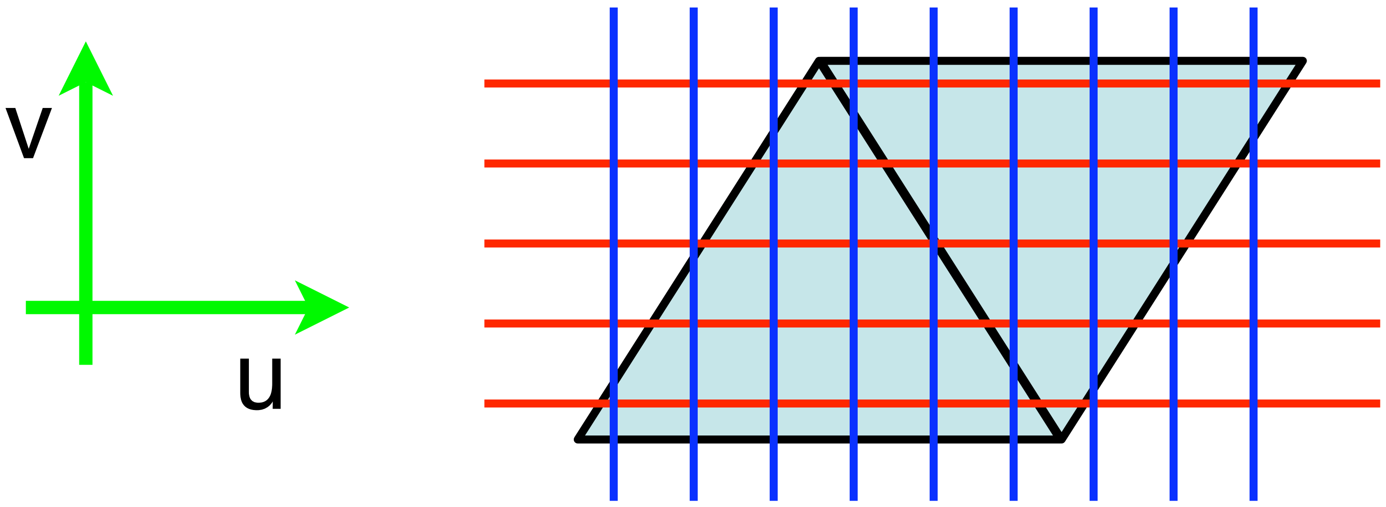


Vector Field Integration

- “periodic coordinates” [Ray et al.]

$$\tilde{\mathbf{u}} = (\cos u, \sin u)$$
$$\tilde{\mathbf{v}} = (\cos v, \sin v)$$





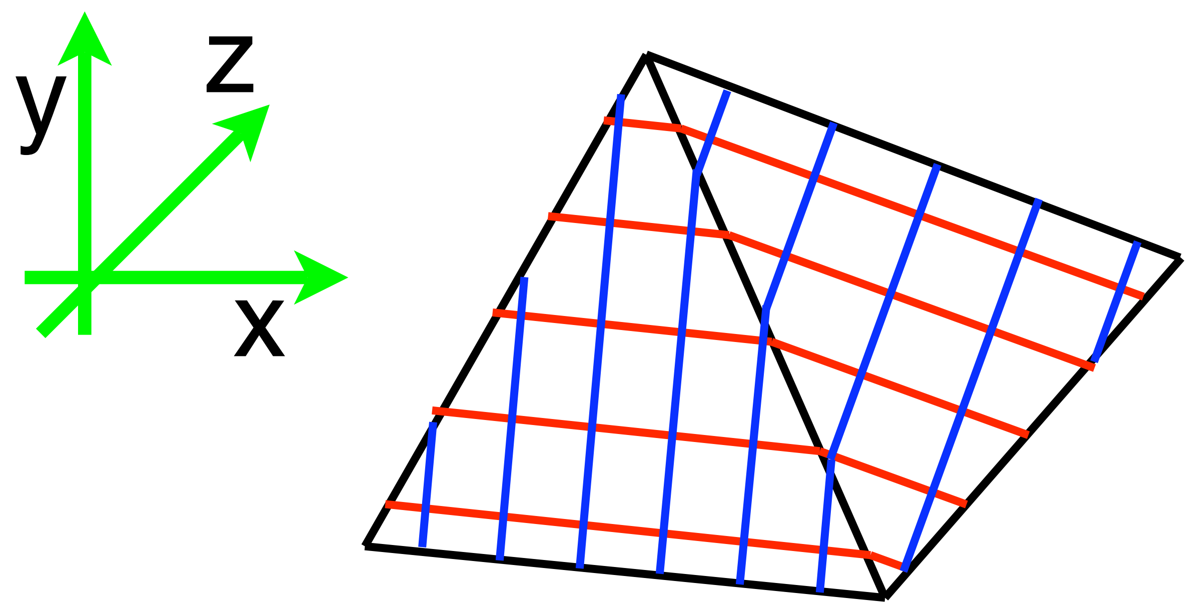
Vector Field Integration

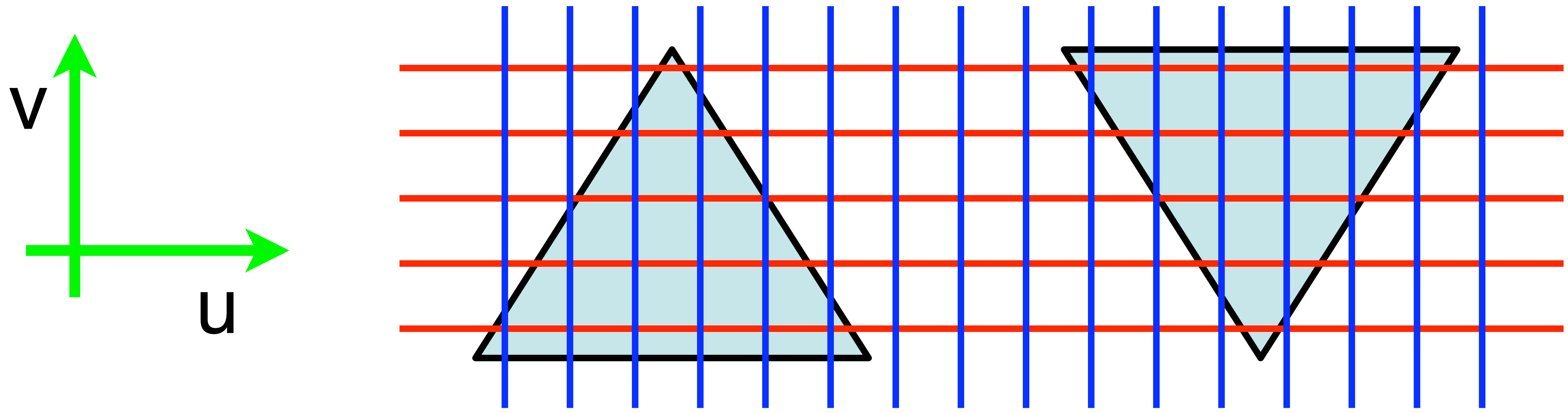
- “periodic coordinates” [Ray et al.]

$$\tilde{\mathbf{u}} = (\cos u, \sin u)$$

$$\tilde{\mathbf{v}} = (\cos v, \sin v)$$

- translations: m π



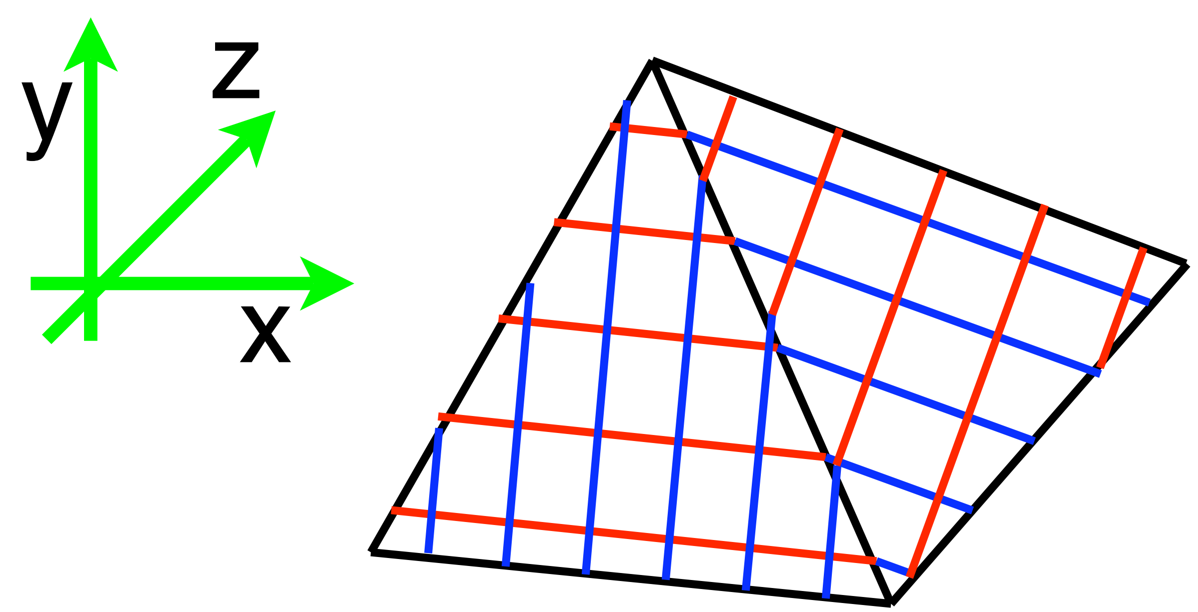


Vector Field Integration

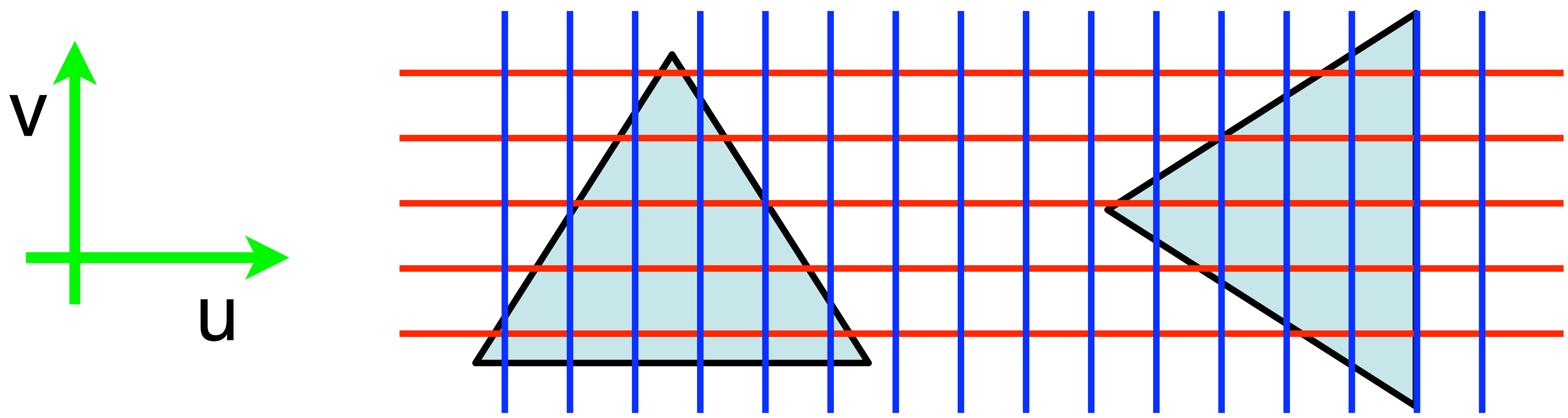
- “periodic coordinates” [Ray et al.]

$$\tilde{\mathbf{u}} = (\cos u, \sin u)$$

$$\tilde{\mathbf{v}} = (\cos v, \sin v)$$



- translations: m π
- rotations: n π/2



Literature

- P. Alliez et al., *“Isotropic Surface Remeshing”*, SMI 2003
- P. Alliez et al., *“Anisotropic polygonal remeshing”*, SIGGRAPH 2003
- M. Botsch, L. Kobbelt, *“A remeshing approach to multiresolution modeling”*, SGP 2004
- V. Surazhsky et al., *“Isotropic Remeshing of Surfaces: A Local Parameterization Approach”*
- Nicolas Ray et al., *“Periodic Global Parametrization”*, ACM ToG 2006

Global Alignment

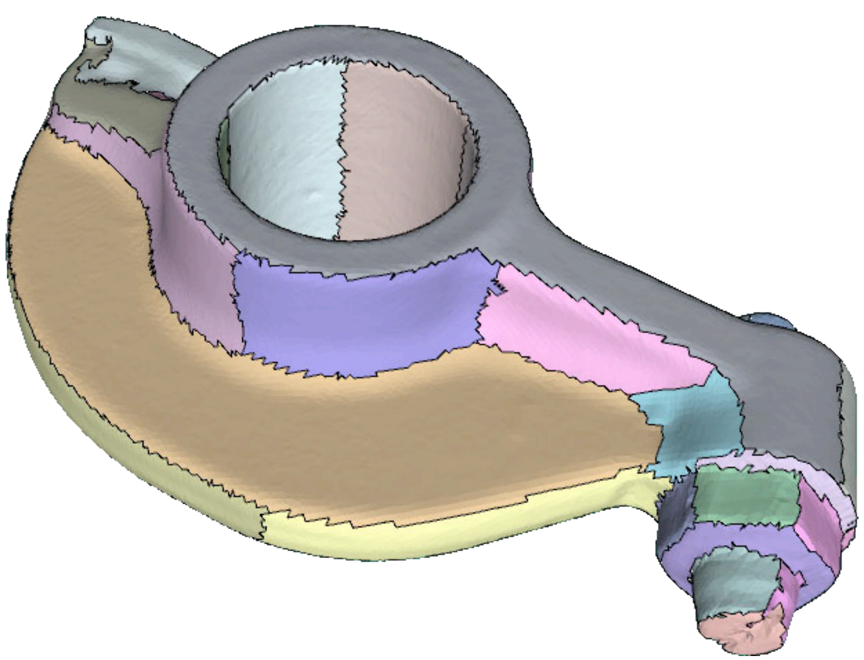
- feature detection
  - thresholding, morphological operations
  - surface snakes
- segmentation
  - region growing
  - clustering
- boundary refinement / optimization
  - graph-cut computation

Feature Detection

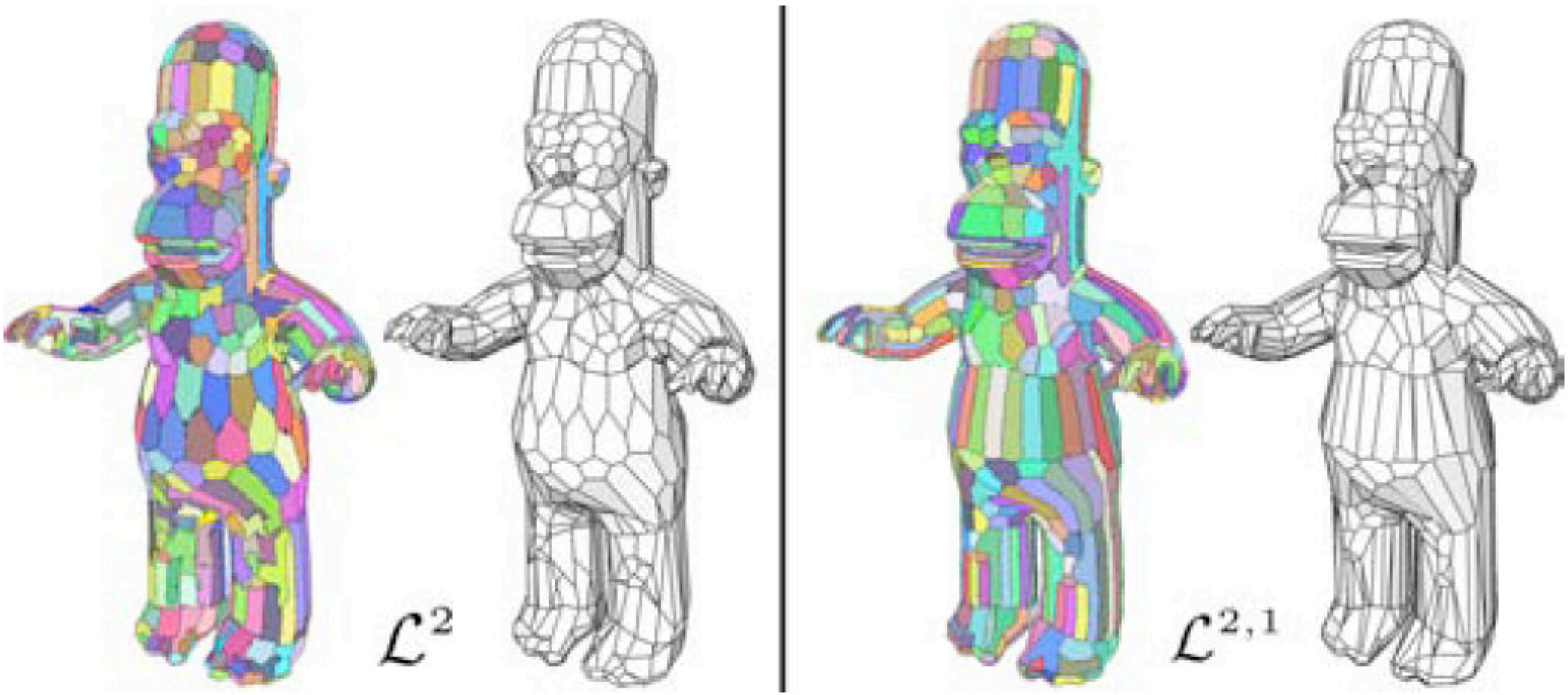
- adapt techniques from image processing
- classify edges by dihedral angle
- topology preserving thinning
  (preserve connected components)
- branch cutting
- snakes on surfaces
  (move polygon towards curvature extrema)

Segmentation

- variational shape approximation
  - select random seeds
  - compute geometry proxies (planes)
  - grow regions / clusters by assigning faces to best matching proxies (L^2 or $L^{2,1}$)
  - iterate:
    - re-compute proxies
    - re-cluster

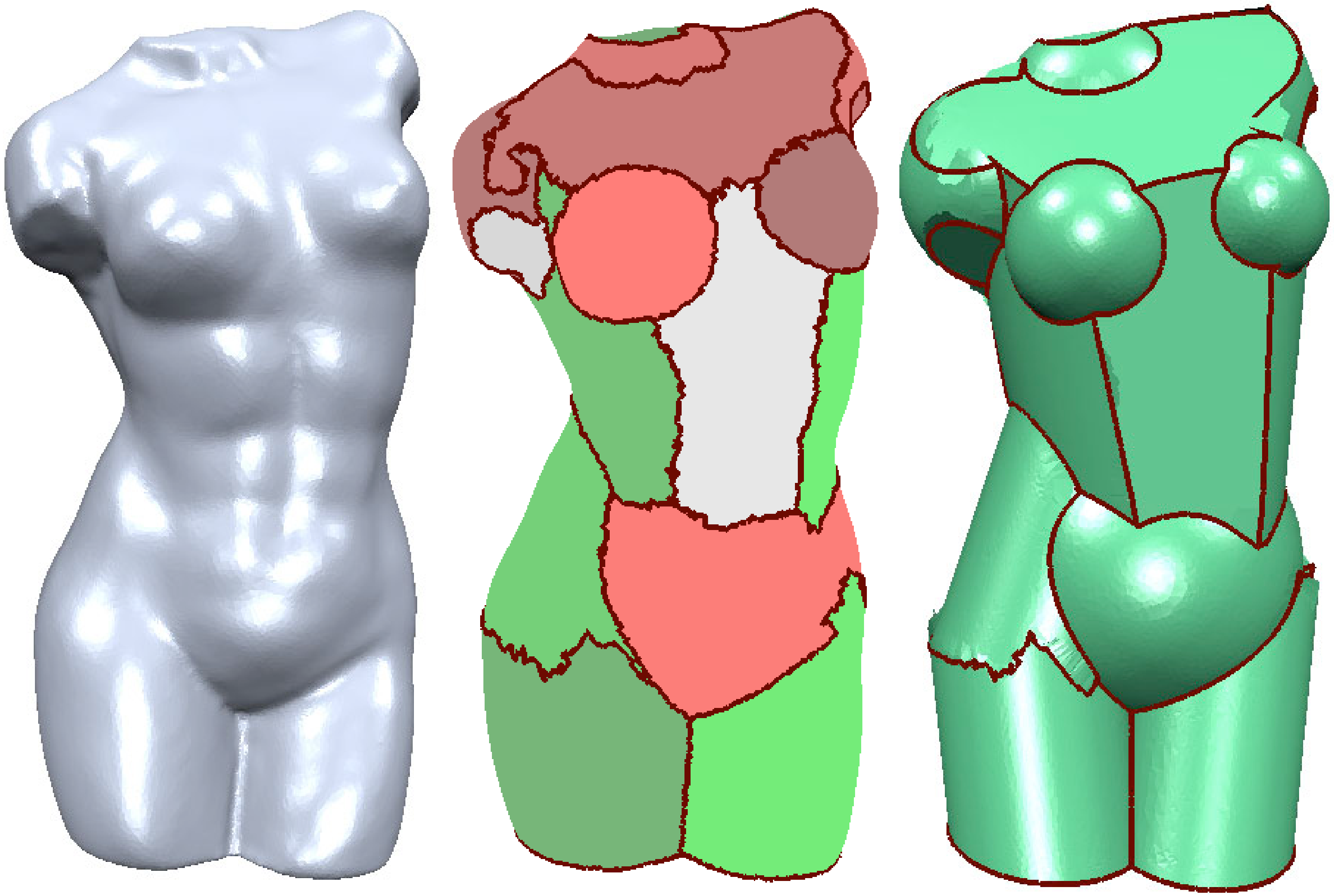


L^2 vs. $L^{2,1}$



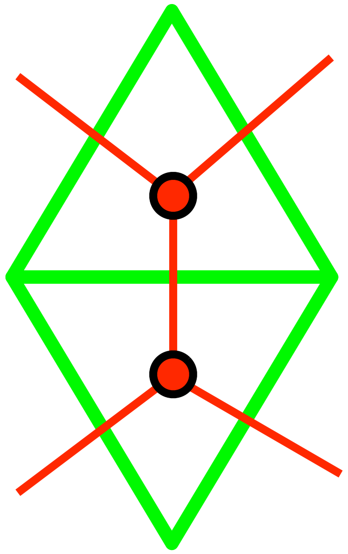
[Cohen-Steiner et al. *"Variational Shape Approximation"*]

Extension to Non-Planar Proxies

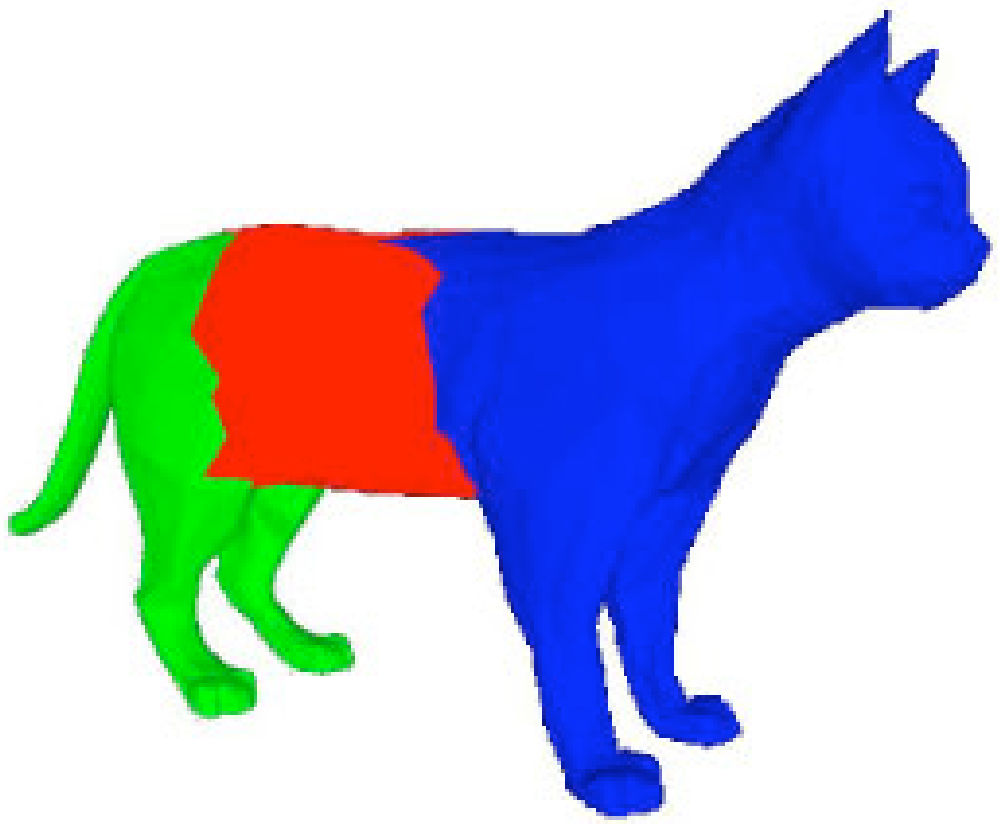


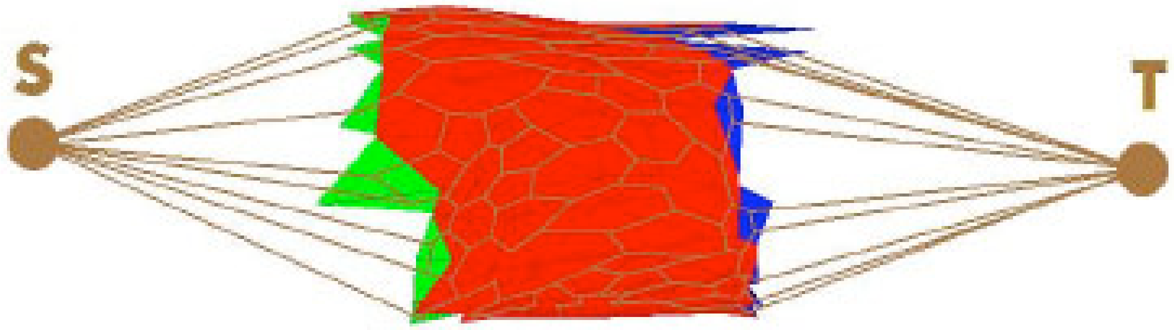
Boundary Refinement

- clustering may oscillate at the segment boundaries
- compute globally optimal boundary polygons by energy minimization
- re-formulation as max-flow / min-cut problem on the dual graph



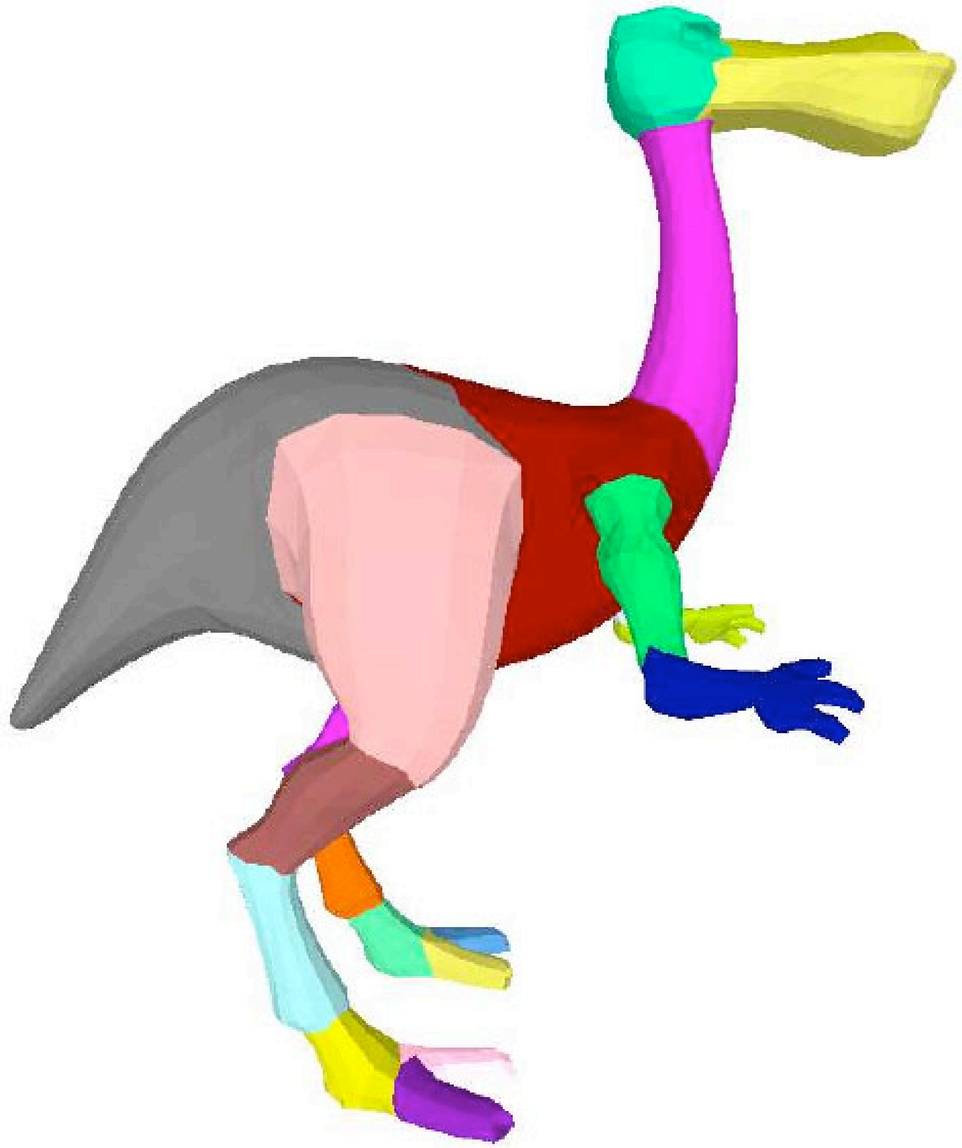
Boundary Refinement





S. Katz, A. Tal, *“Hierarchical Mesh Decomposition using Fuzzy Clustering and Cuts”*

Boundary Refinement



S. Katz, A. Tal, *“Hierarchical Mesh Decomposition using Fuzzy Clustering and Cuts”*

Literature

- S. Katz, A. Tal, *“Hierarchical Mesh Decomposition using Fuzzy Clustering and Cuts”*, SIGGRAPH 2003
- D. Cohen-Steiner et al., *“Variational Shape Approximation”*, SIGGRAPH 2004

Remeshing Cookbook

- problem definition
  - input, output
- basic ingredients
  - general requirements
  - types of operations
- a selection of recipes
  - various representative examples of known remeshing schemes

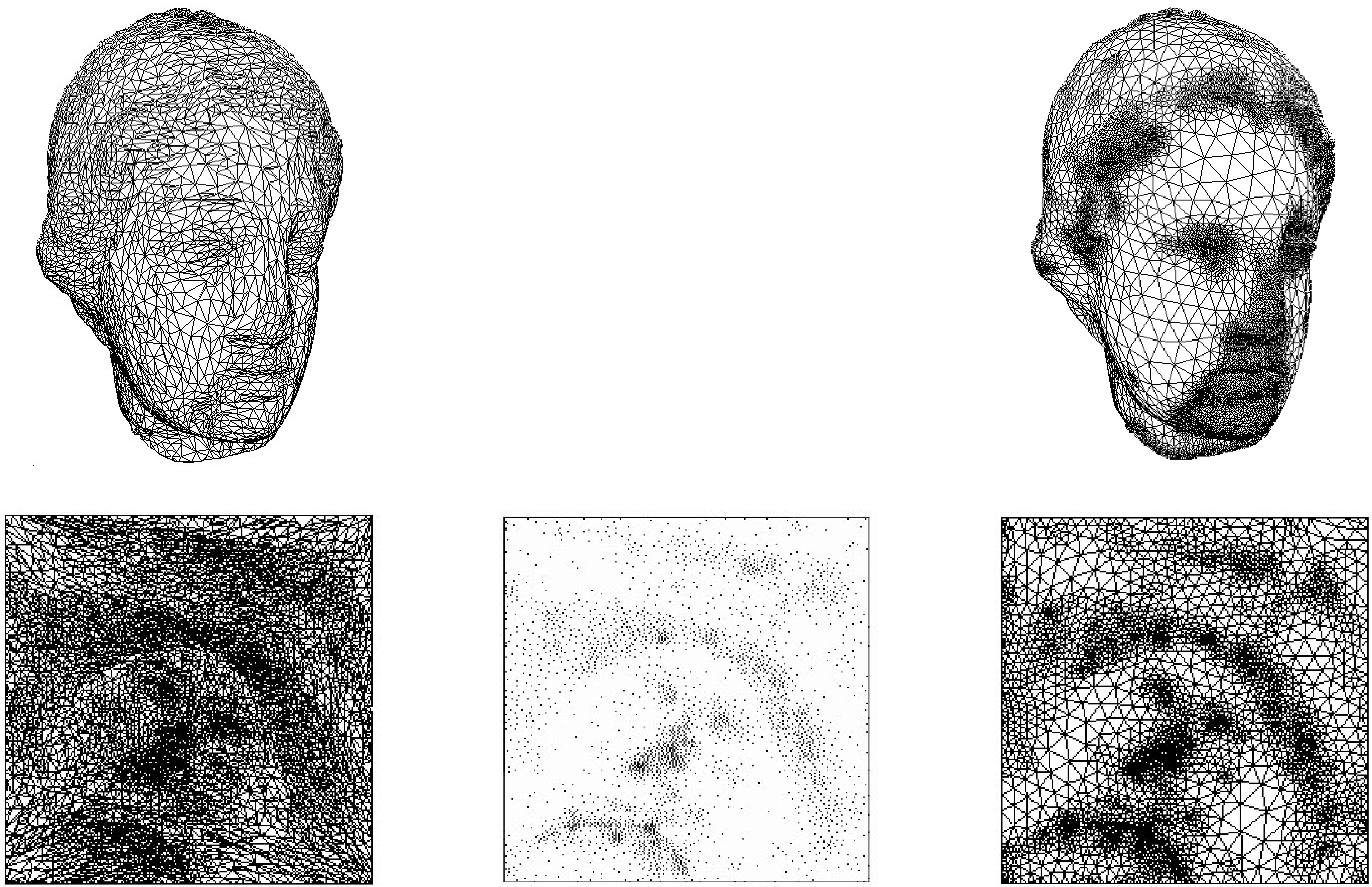
A Selection of Recipes

- realtime remeshing
  (global parametrization)
- iterative mesh optimization
  (local or no parametrization)
- quad-dominant meshing
  (anisotropic, with and without parametrization)
- globally harmonic meshing
  (anisotropic, global parametrization)

Realtime Remeshing

- Alliez et al. *“Interactive Geometry Remeshing”* SIGGRAPH 2003
- compute global parametrization over a rectangle
- define vertex density map in the parameter domain
- generate samples by half-toning / dithering (error diffusion)
- compute 2D Delaunay triangulation

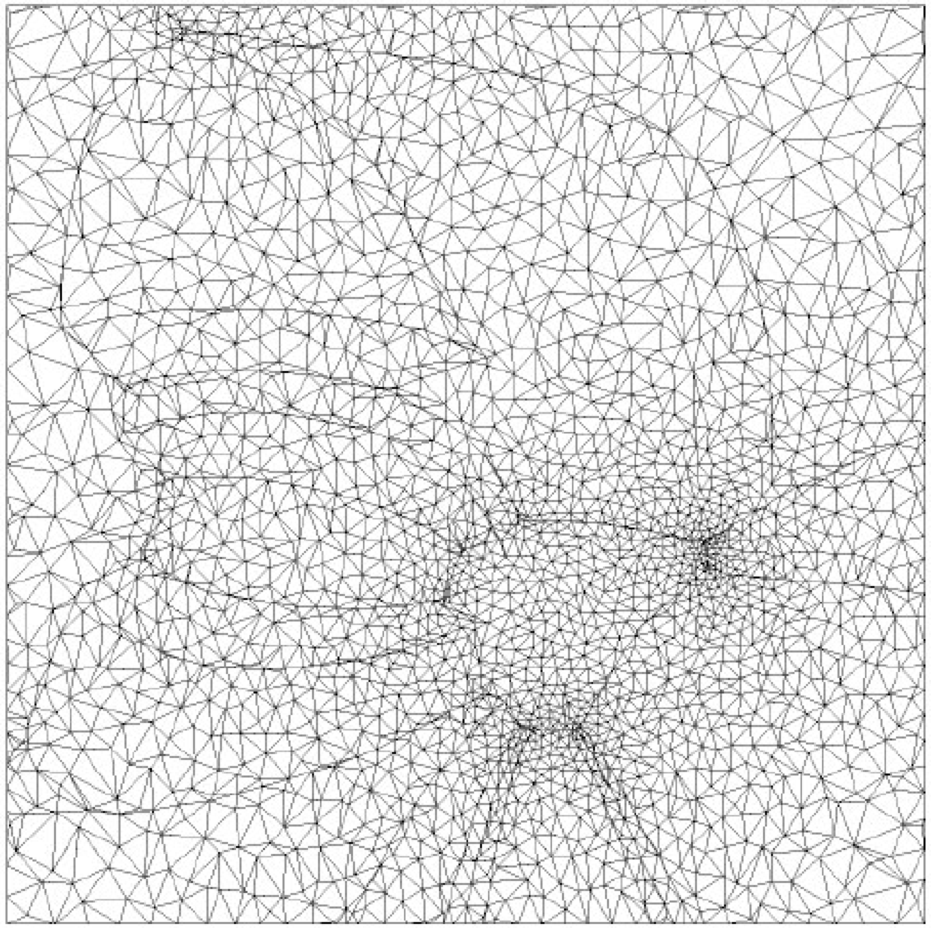
Realtime Remeshing

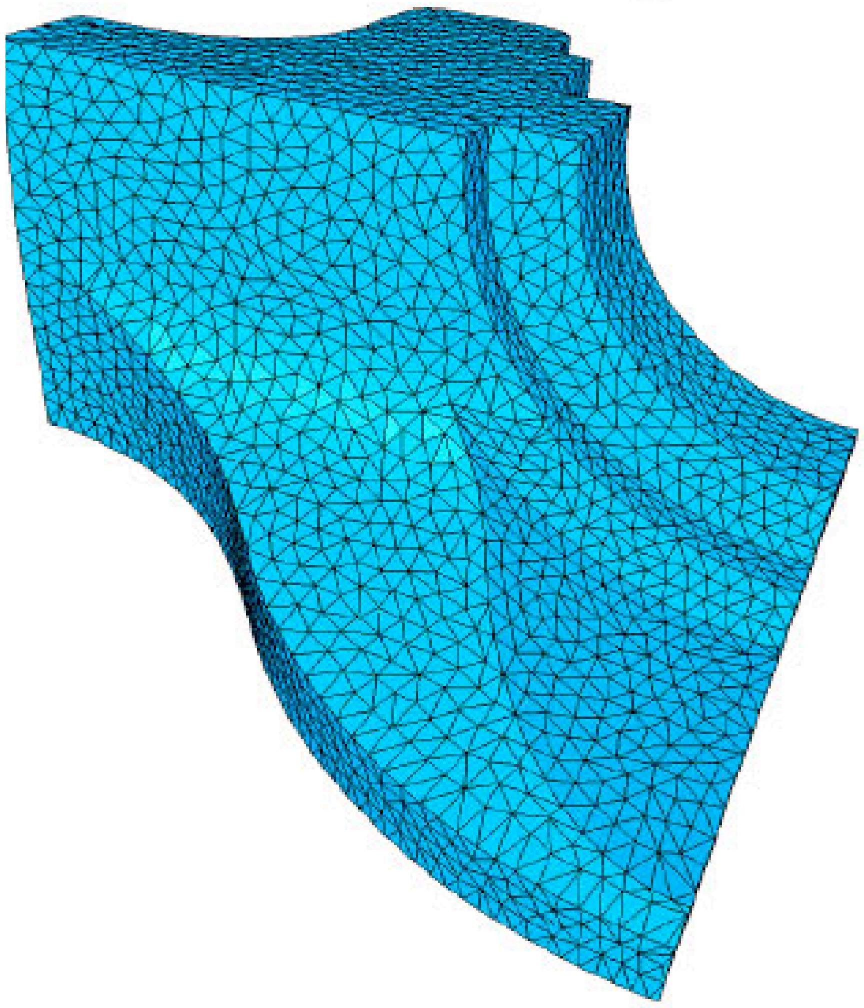


Realtime Remeshing

- **global parametrization** approach
- **vertex density** by half-toning
- no **local alignment** (isotropic)
- **global alignment** possible by constrained Delaunay triangulation

Realtime Remeshing





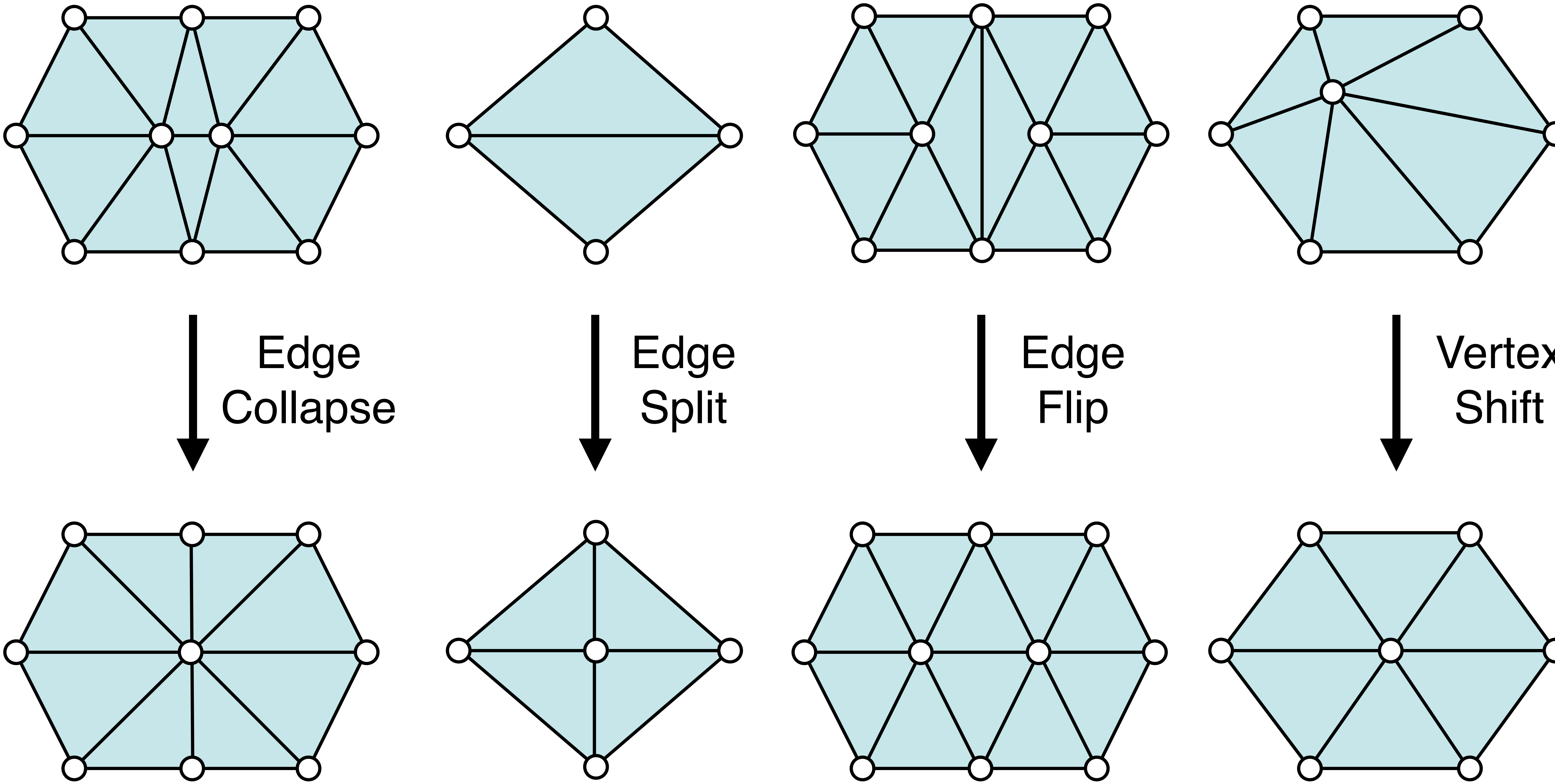
Iterative Mesh Optimization

- (area weighted) random scatter
  or simply start with the given mesh
- improve vertex distribution by
  - particle systems (Turk)
  - area-weighted Laplace smoothing (Surazhsky 1)
  - centroidal Voronoi diagram (Surazhsky 2)
- update mesh connectivity

Iterative Mesh Optimization

- isotropic remeshing prefers ...
  - equal edge length
    - remove too short edges — edge collapses
    - remove too long edges — 2-4 edge split
  - regular valences
    - valence balance — edge flip
  - uniform vertex distribution
    - tangential smoothing — Laplace operator

Local Remeshing Operators



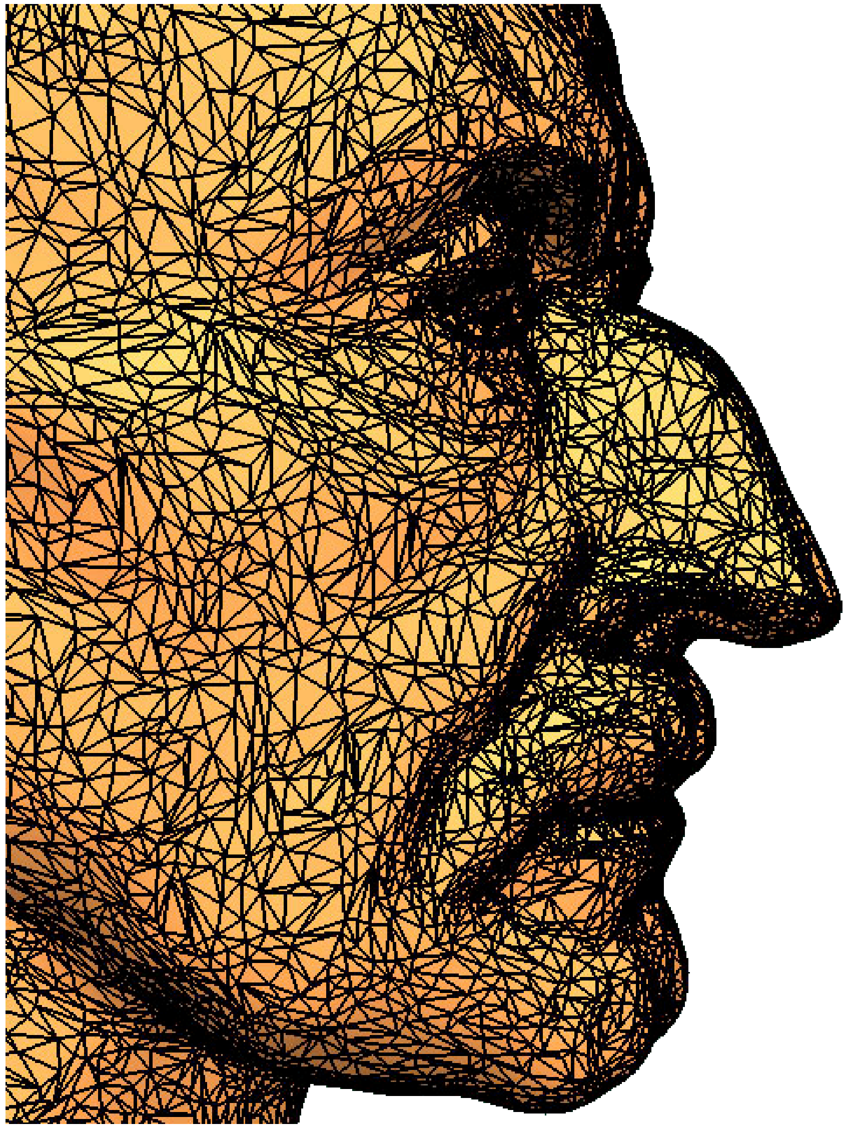
Isotropic Remeshing

Specify target edge length L

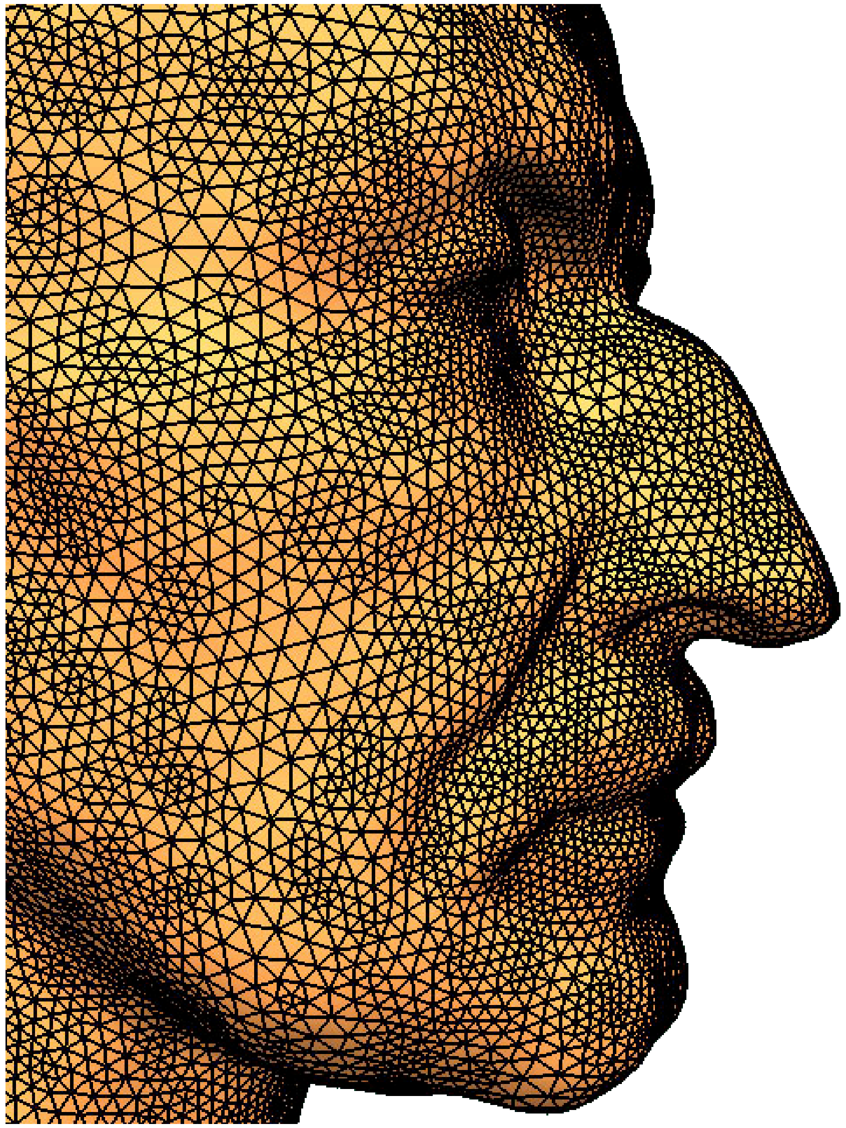
Iterate:

1. Split edges longer than L_{max}
2. Collapse edges shorter than L_{min}
3. Flip edges to get closer to valence 6
4. Vertex shift by tangential relaxation
5. Project vertices onto reference mesh

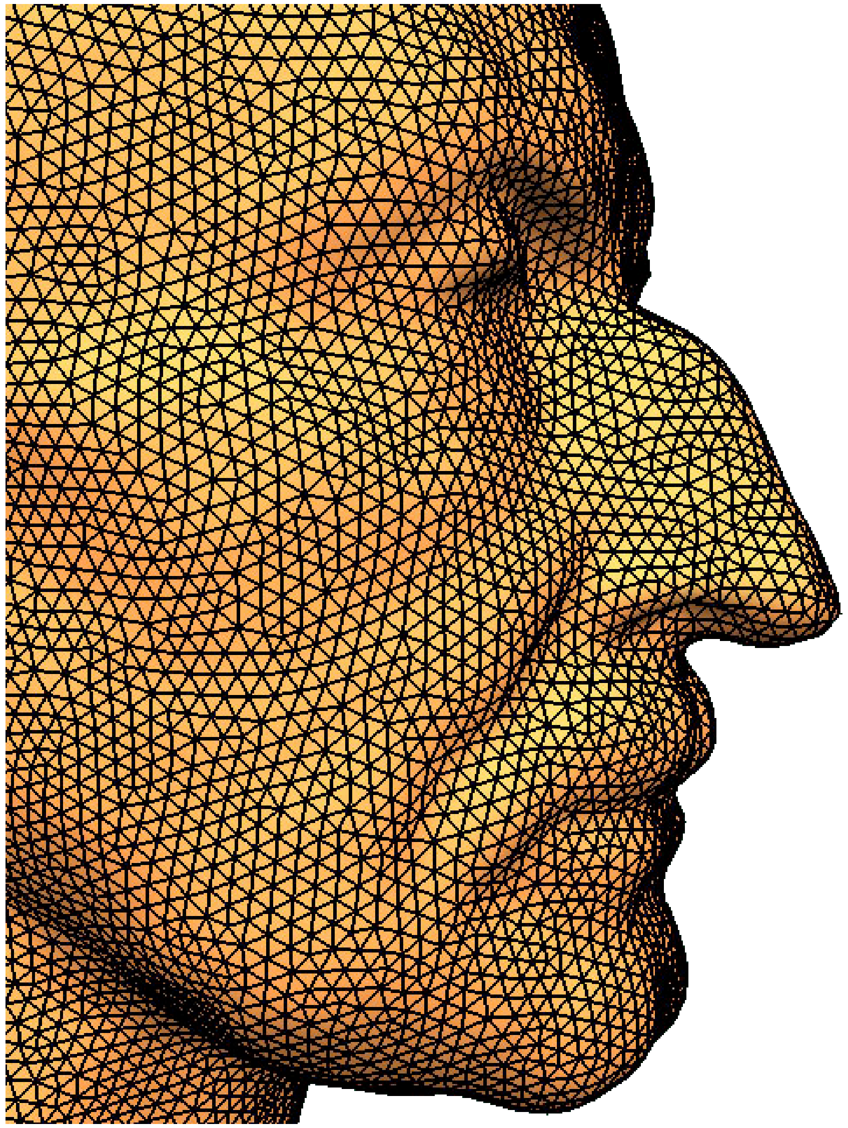
Thresholds L_{min} and L_{max}



Original

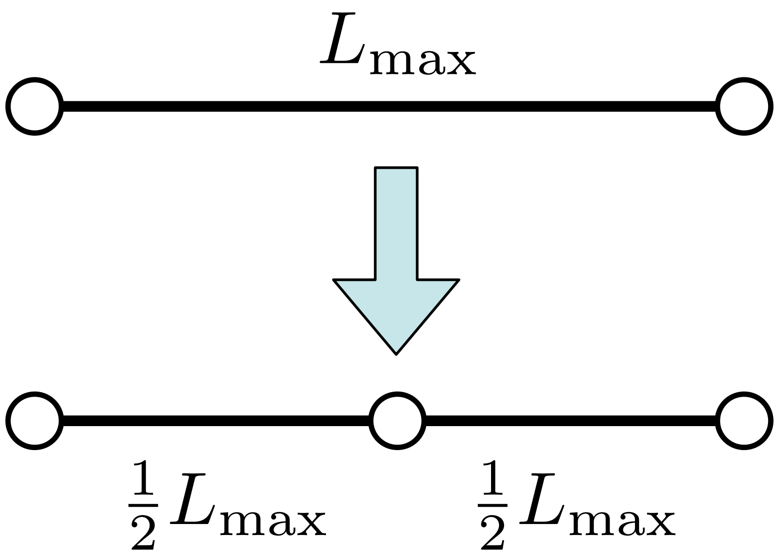


$\left(\frac{1}{2}, 2\right)$

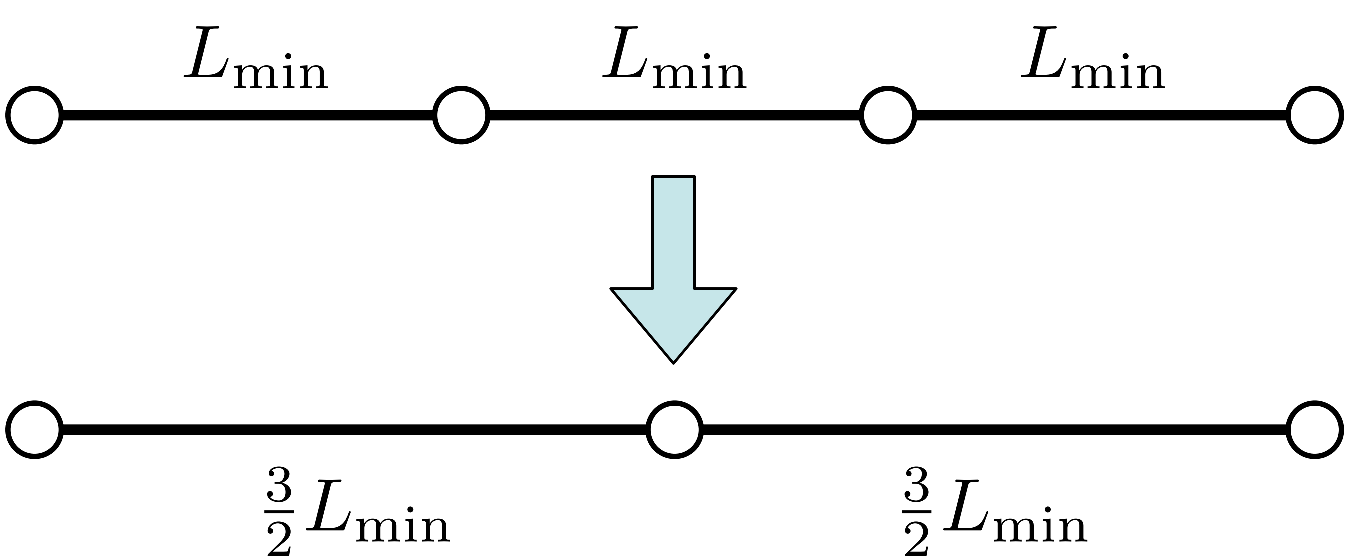


$\left(\frac{4}{5}, \frac{4}{3}\right)$

Edge Collapse / Split



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



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

Area Weighted Tangential Smoothing

- tangential smoothing with area equalization (leads to symmetric Laplace matrix)

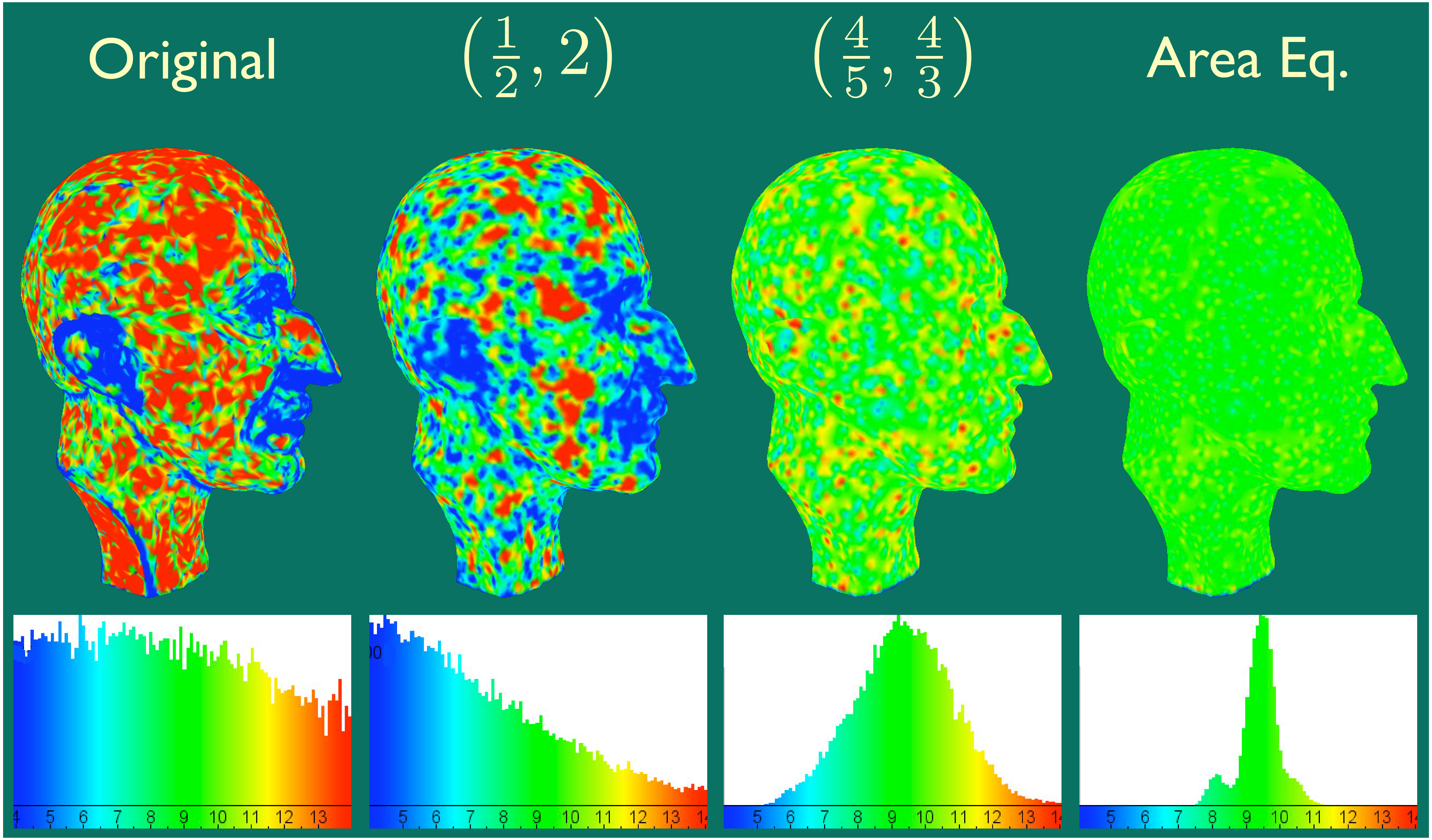
- area-weighted centroid

$$\mathbf{g}_i = \frac{1}{\sum_{\mathbf{q}_i} A(\mathbf{q}_i)} \sum_{\mathbf{q}_i} A(\mathbf{q}_i)\, \mathbf{q}_i$$

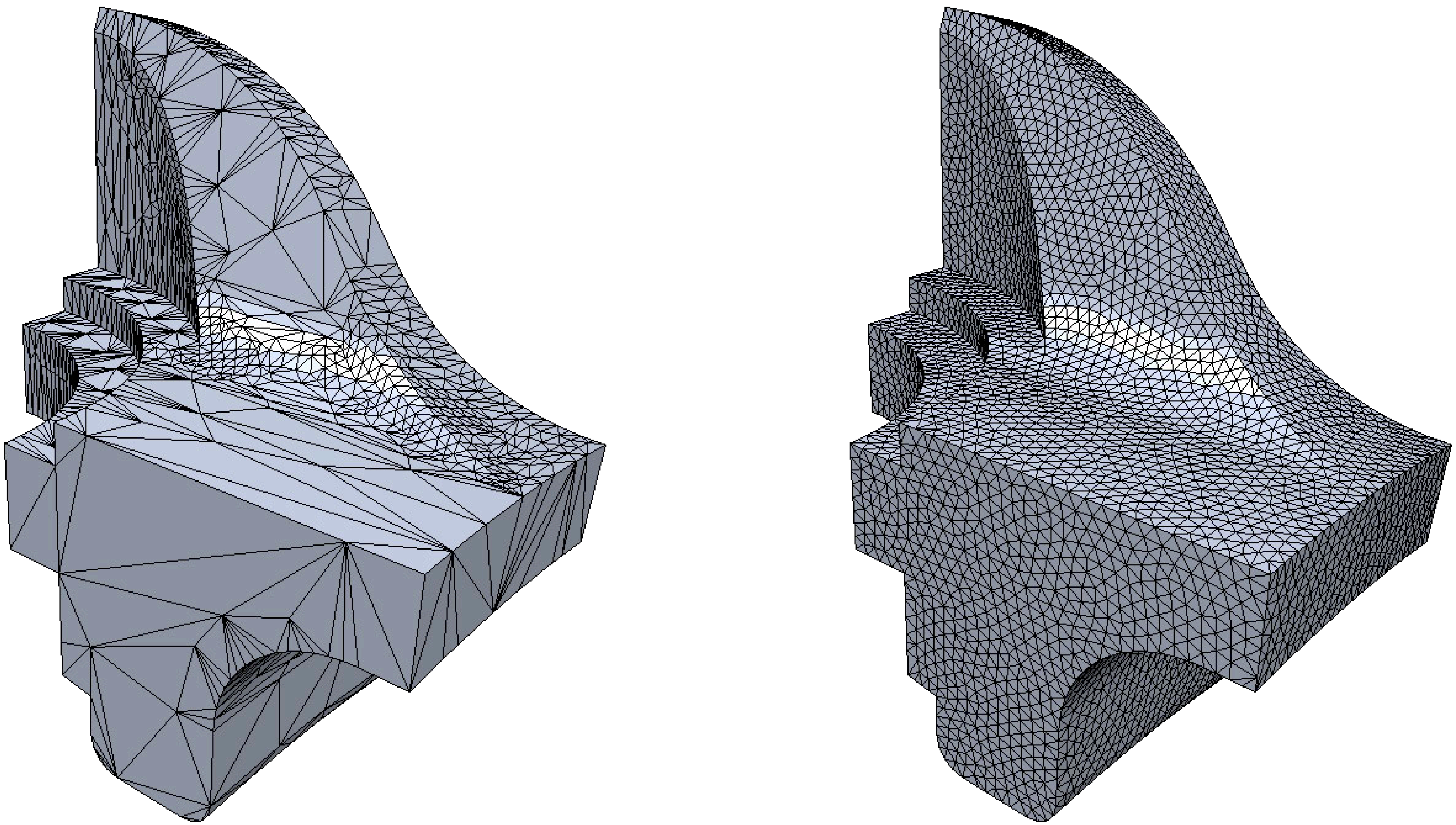
- tangential update

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

Remeshing Results

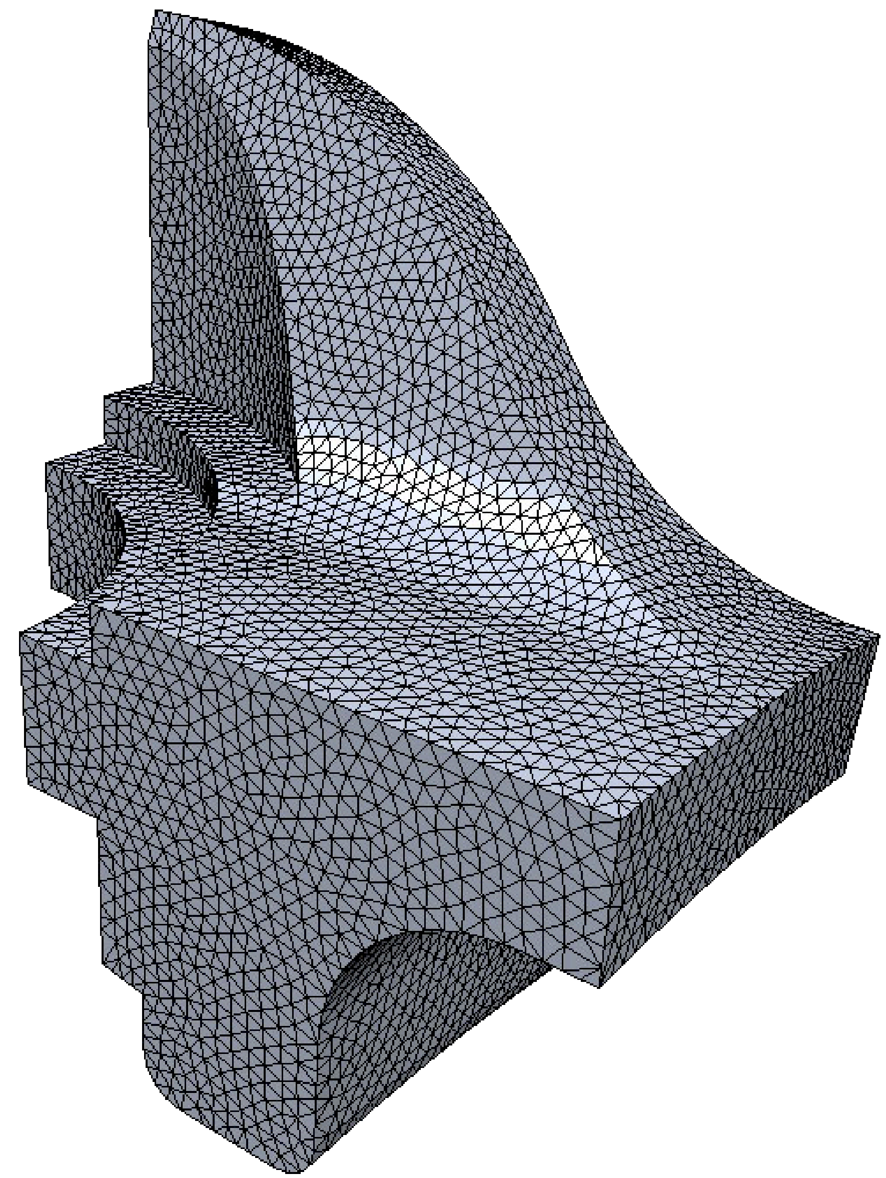


Feature Preservation



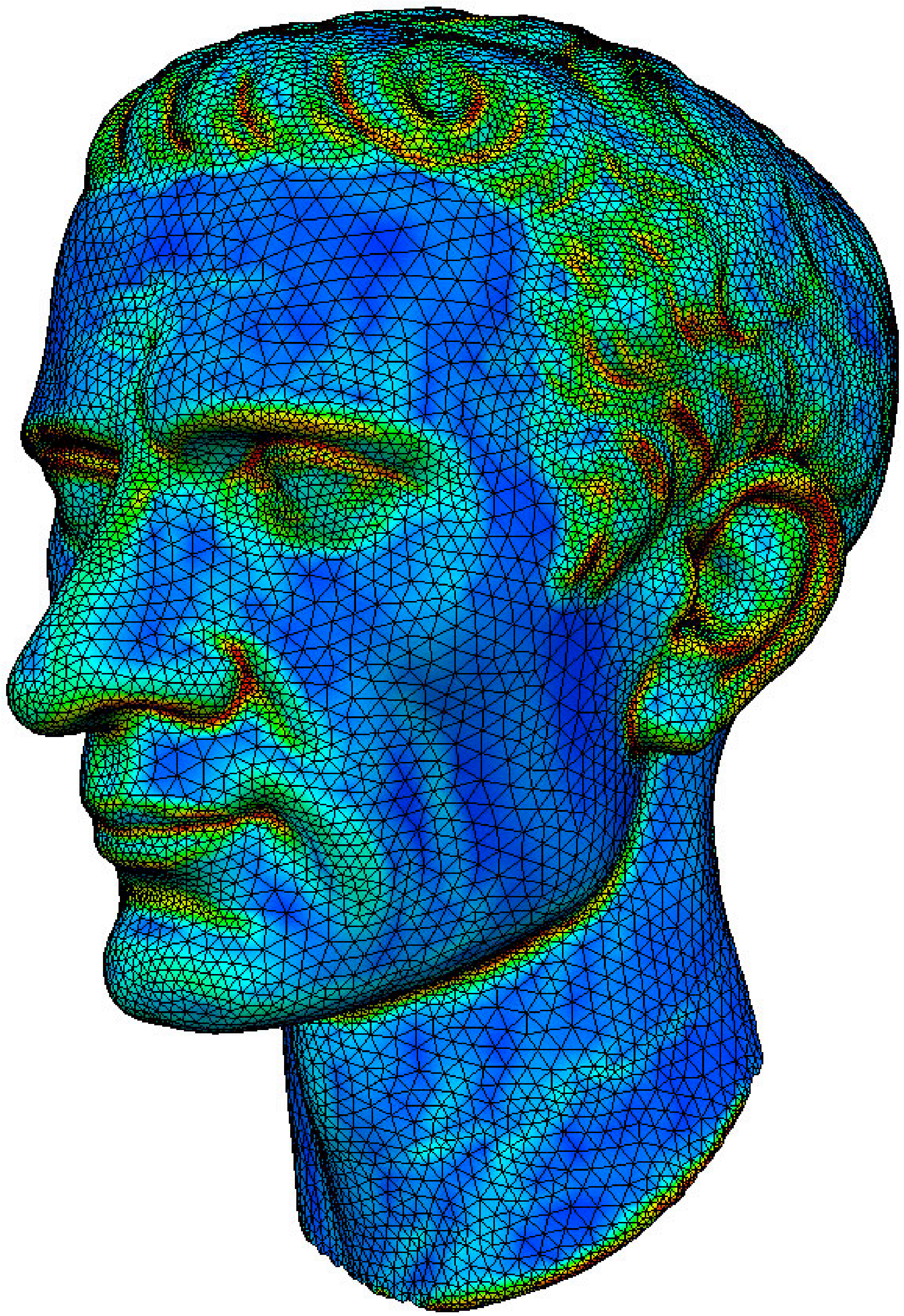
Feature Preservation

- define features
  - sharp edges
  - material boundaries
- adjust local operators
  - don't flip
  - collapse only along features
  - univariate smoothing
  - project to feature curves



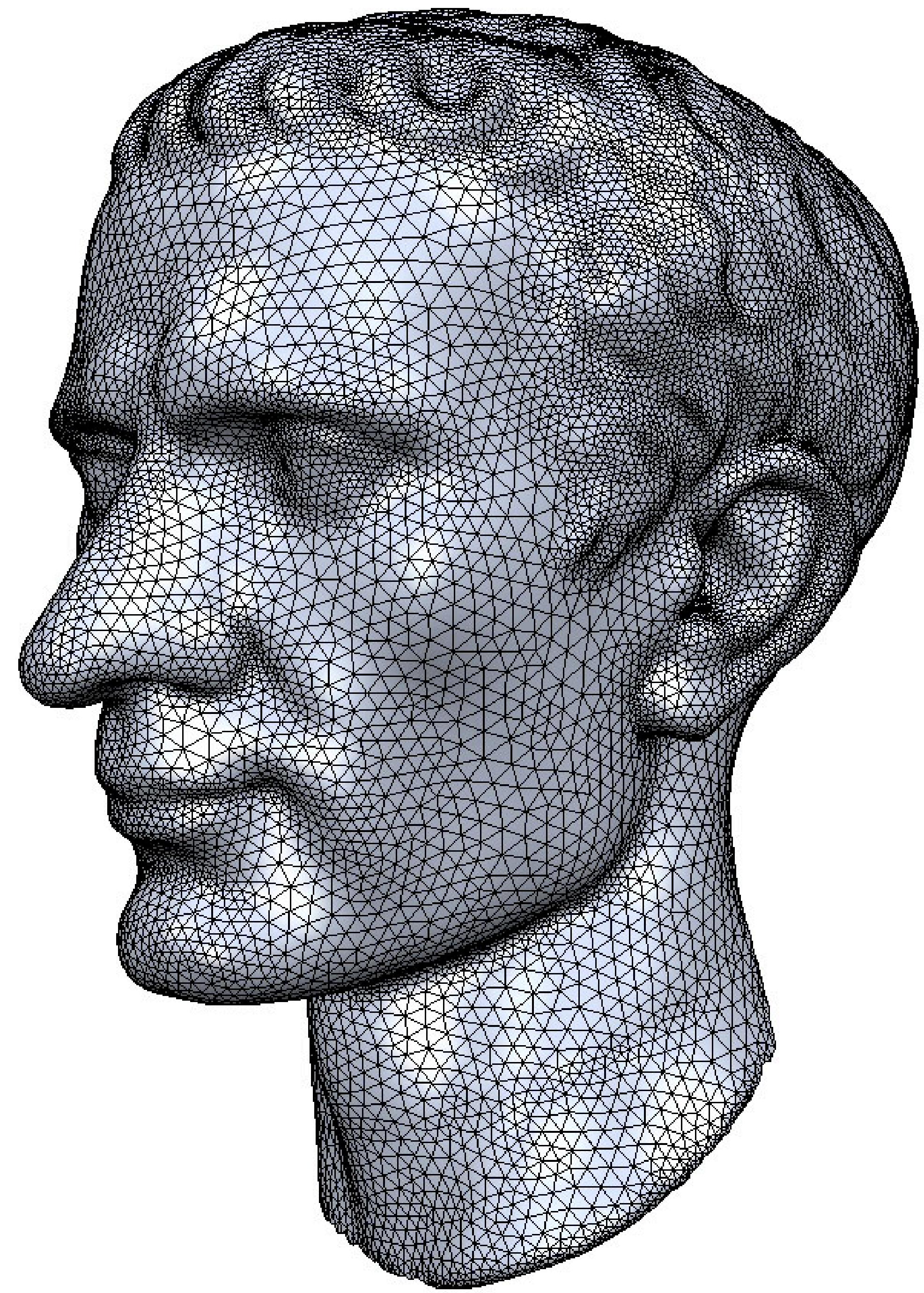
Adaptive Remeshing

- precompute max. curvature on reference mesh
- target edge length locally determined by curvature
- adjust split / collapse criteria



Isotropic Remeshing

- high quality triangulations
  - equilateral triangles
  - valence 6
- extensions
  - feature preservation
  - curvature adaptation
- local operators & projection
  - easy to implement
  - computationally efficient
  - 100K vertices in <5 sec



Iterative Mesh Optimization

- no **parametrization** necessary
- adaptive **vertex distribution** by tangential Laplace and topologcal updates
- no **local orientation** (isotropic meshing)
- **global feature alignment** by restriction of mesh operations

Literature

- Vorsatz et al, *“Dynamic remeshing and applications”*, Solid Modeling 2003
- Surazhsky et al. *“Isotropic Remeshing of Surfaces: a local parametrization approach”*
- Botsch & Kobbelt, *“A remeshing approach to multiresolution modeling”*,
  Symp. on Geometry Processing 2004
- Alliez et al, *“Recent advances in remeshing of surfaces”*, AIM@Shape state of the art report, 2006

Quad Dominant Meshing

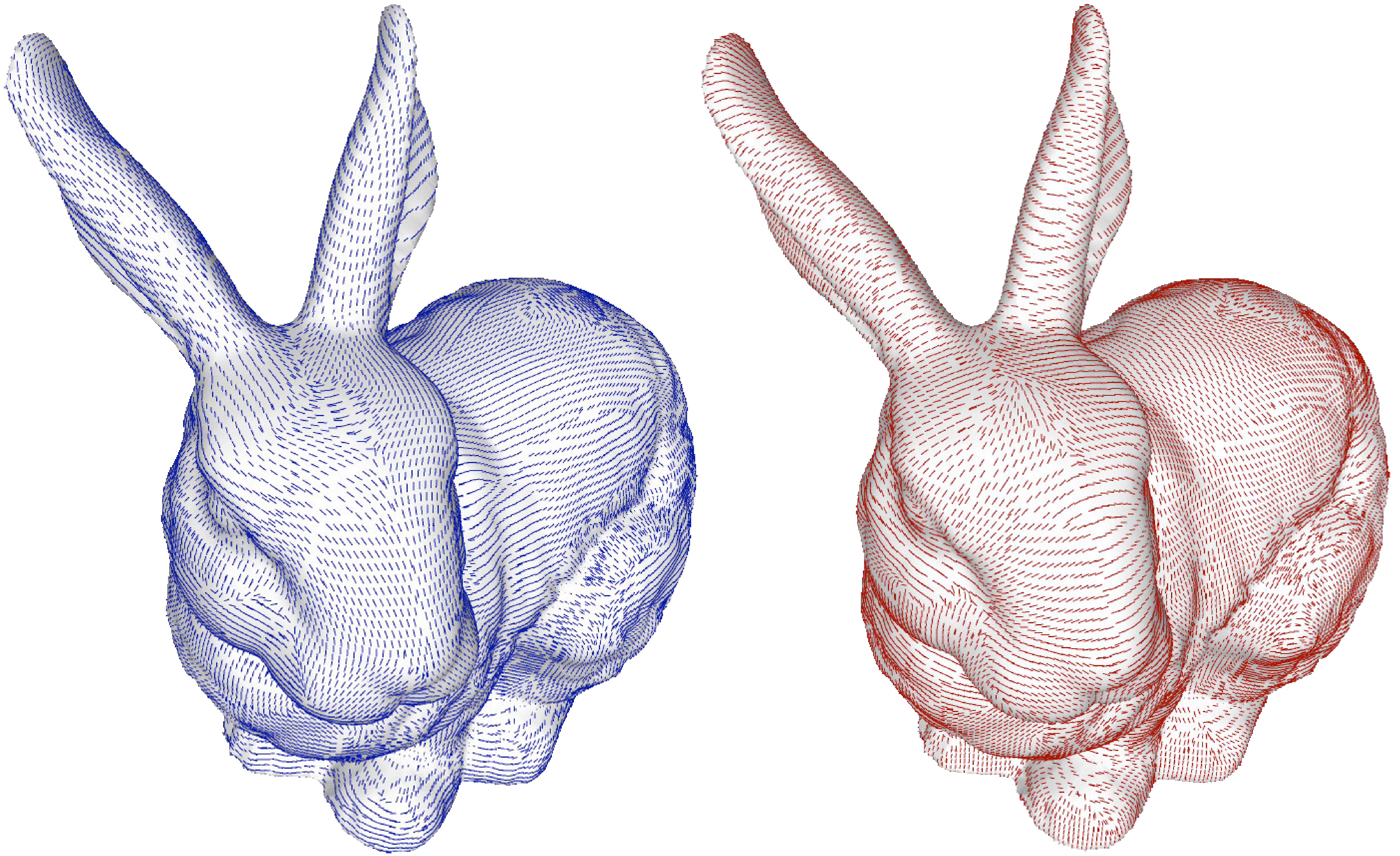
- anisotropic remeshing prefers ...
  - quad faces
  - curvature dependent size and aspect ratio (approximation measure)
  - local orientation (curvature directions, shape operator)
  - global alignment (feature detection and handling)

Quad Dominant Meshing

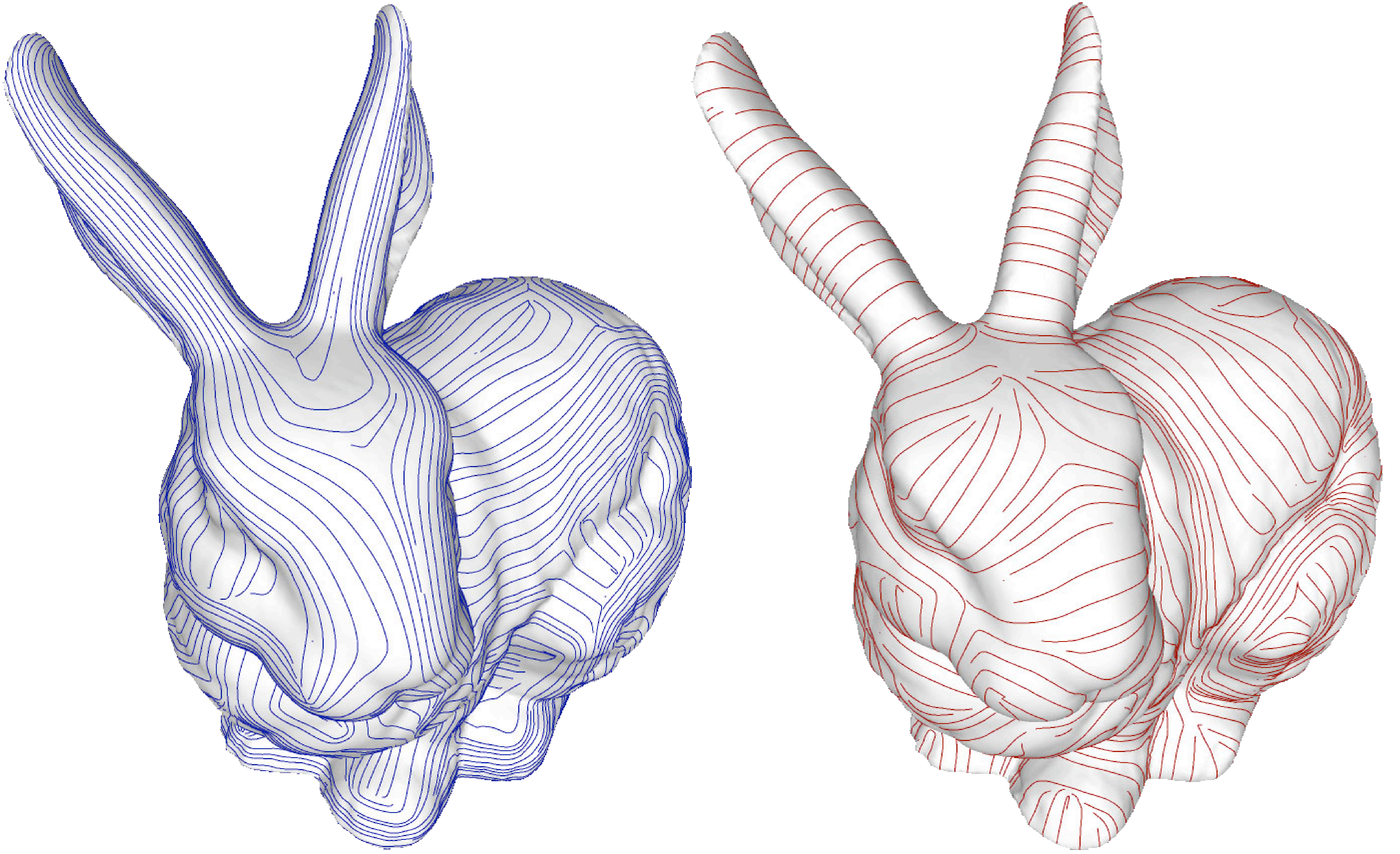
- line density depends on approximation measure
  - L^2 vs $L^{2,1}$
  - L^2 measures geometric deviation
  - $L^{2,1}$ leads to K_{min} / K_{max} aspect ratios

- local orientation by the shape operator
  - K_{min} and K_{max} direction fields
  - direction propagation

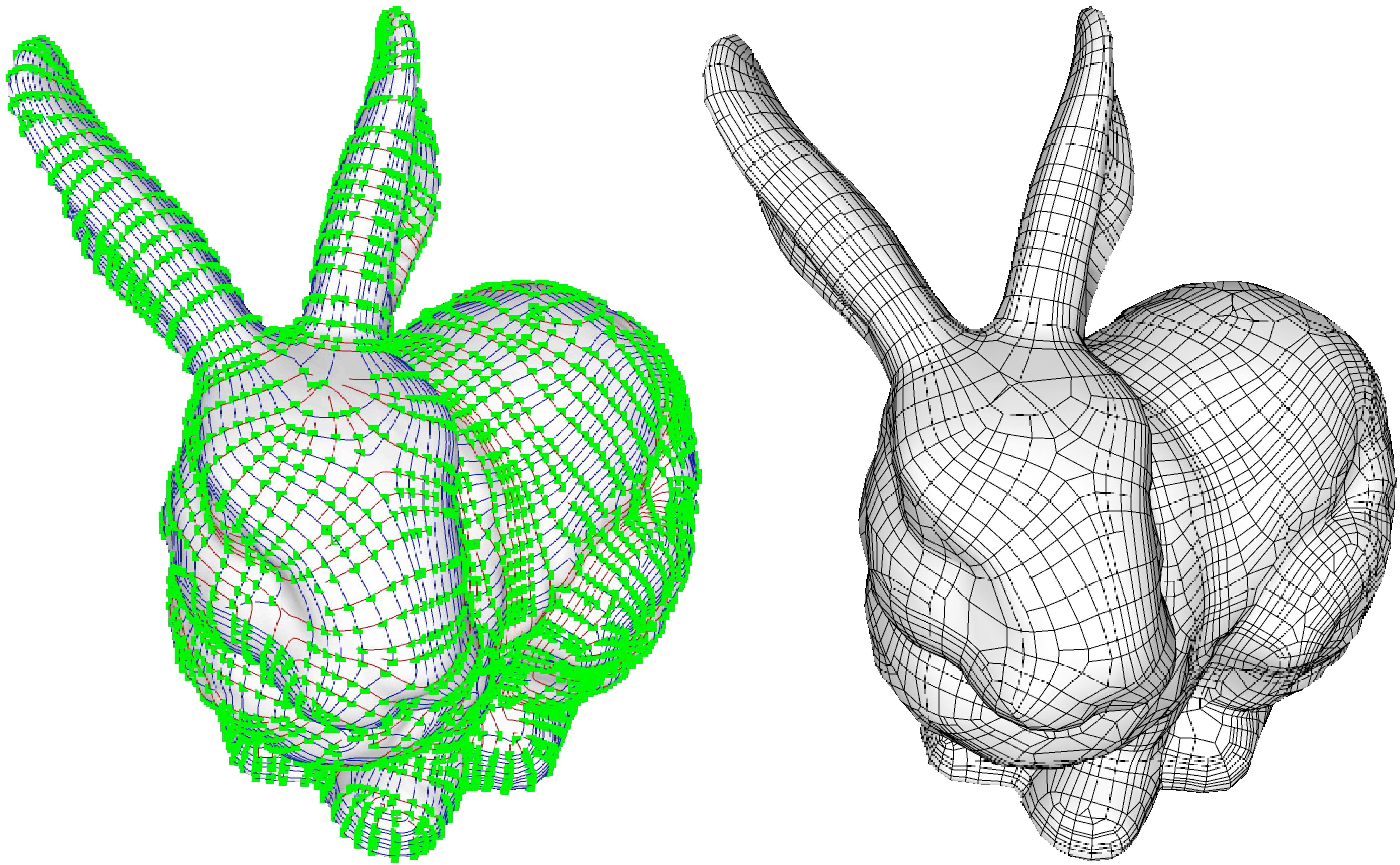
Quad Dominant Meshing



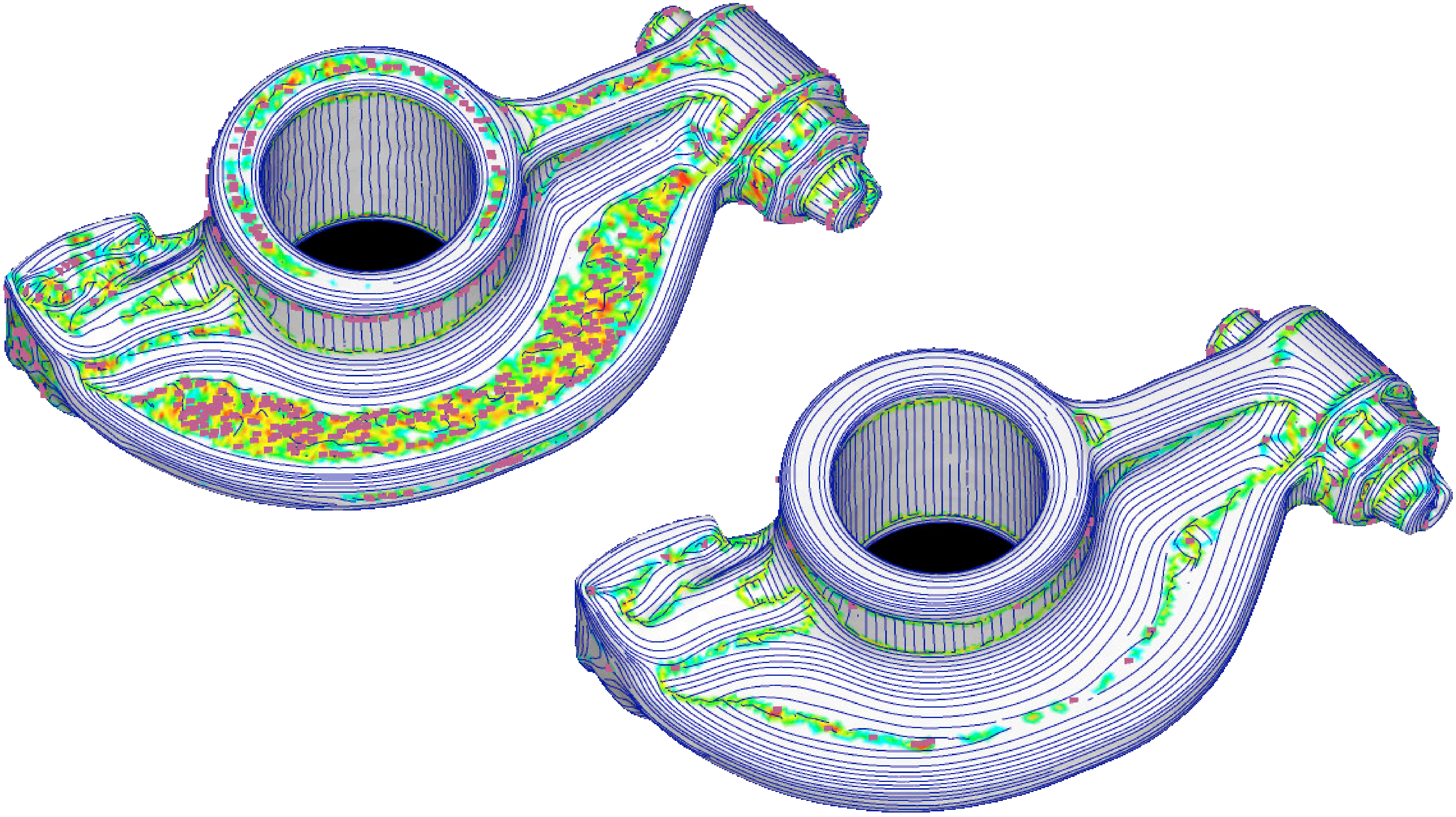
Quad Dominant Meshing



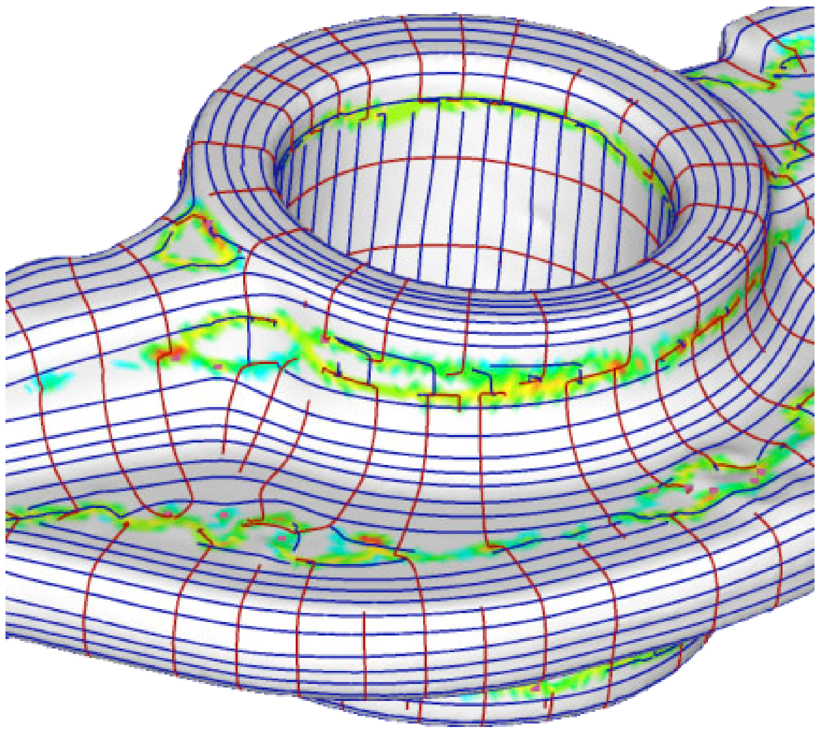
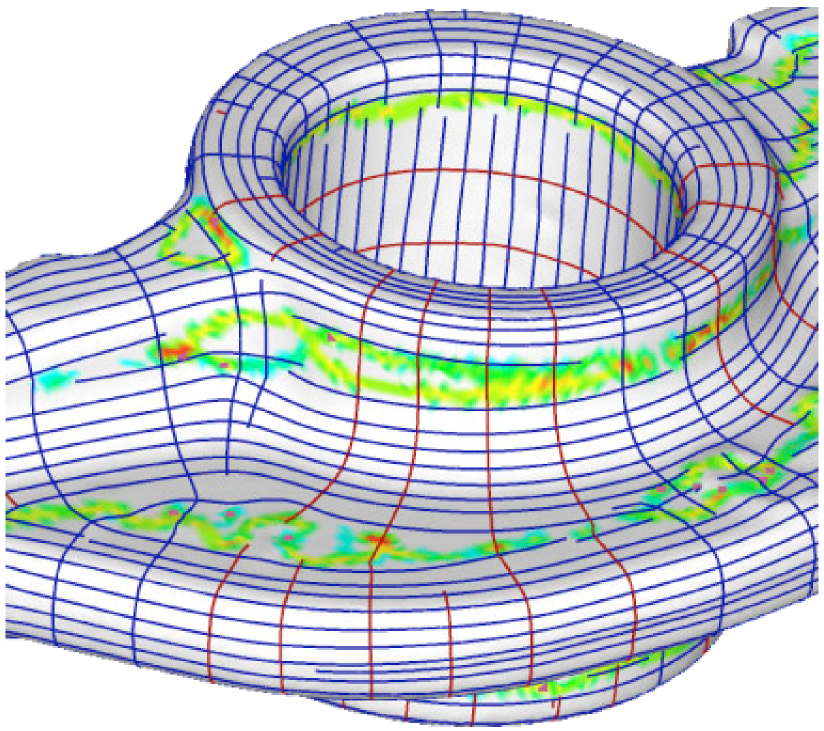
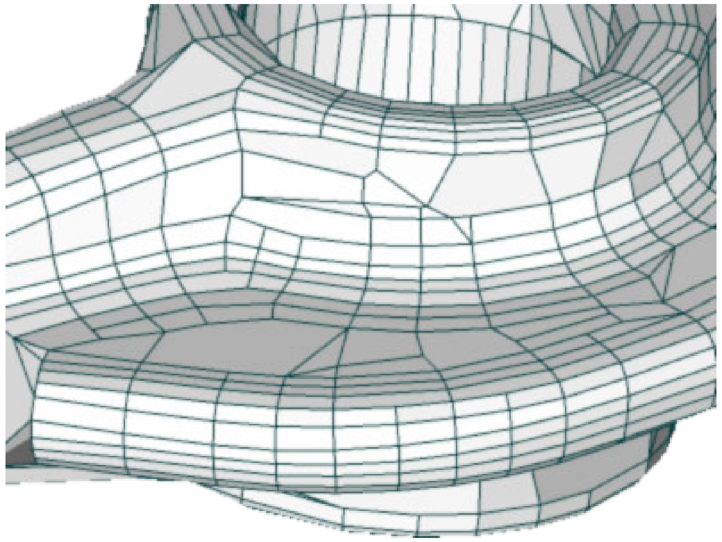
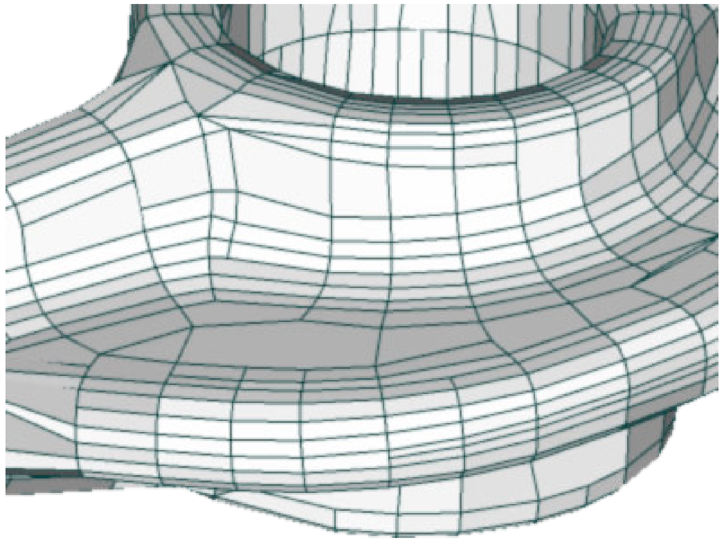
Quad Dominant Meshing



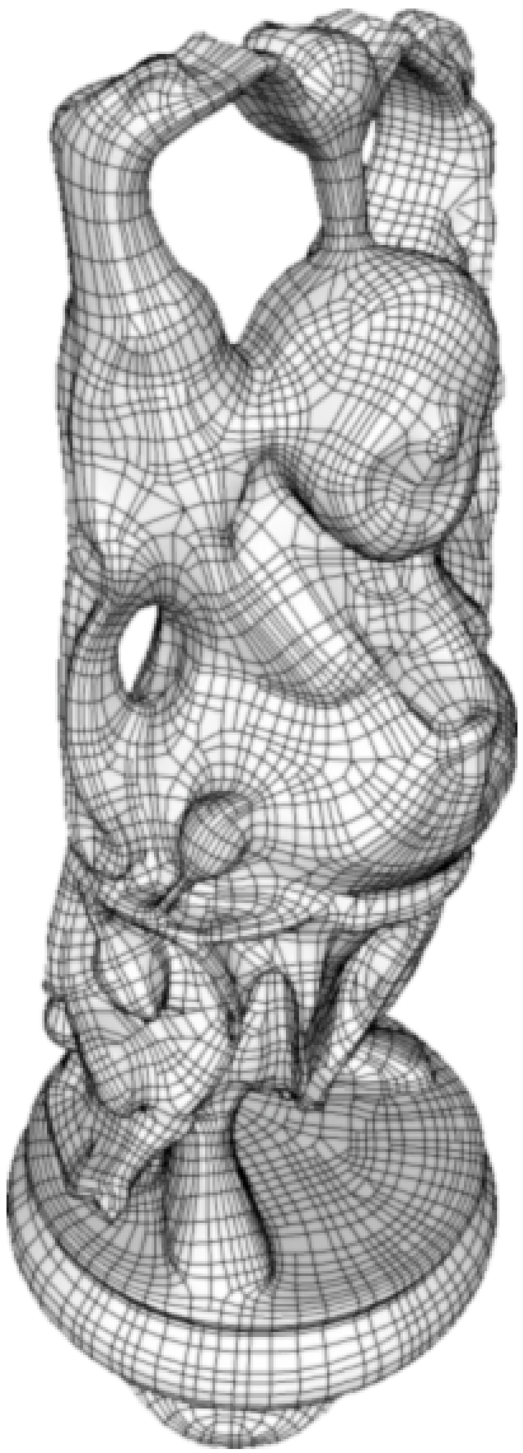
Quad Dominant Meshing

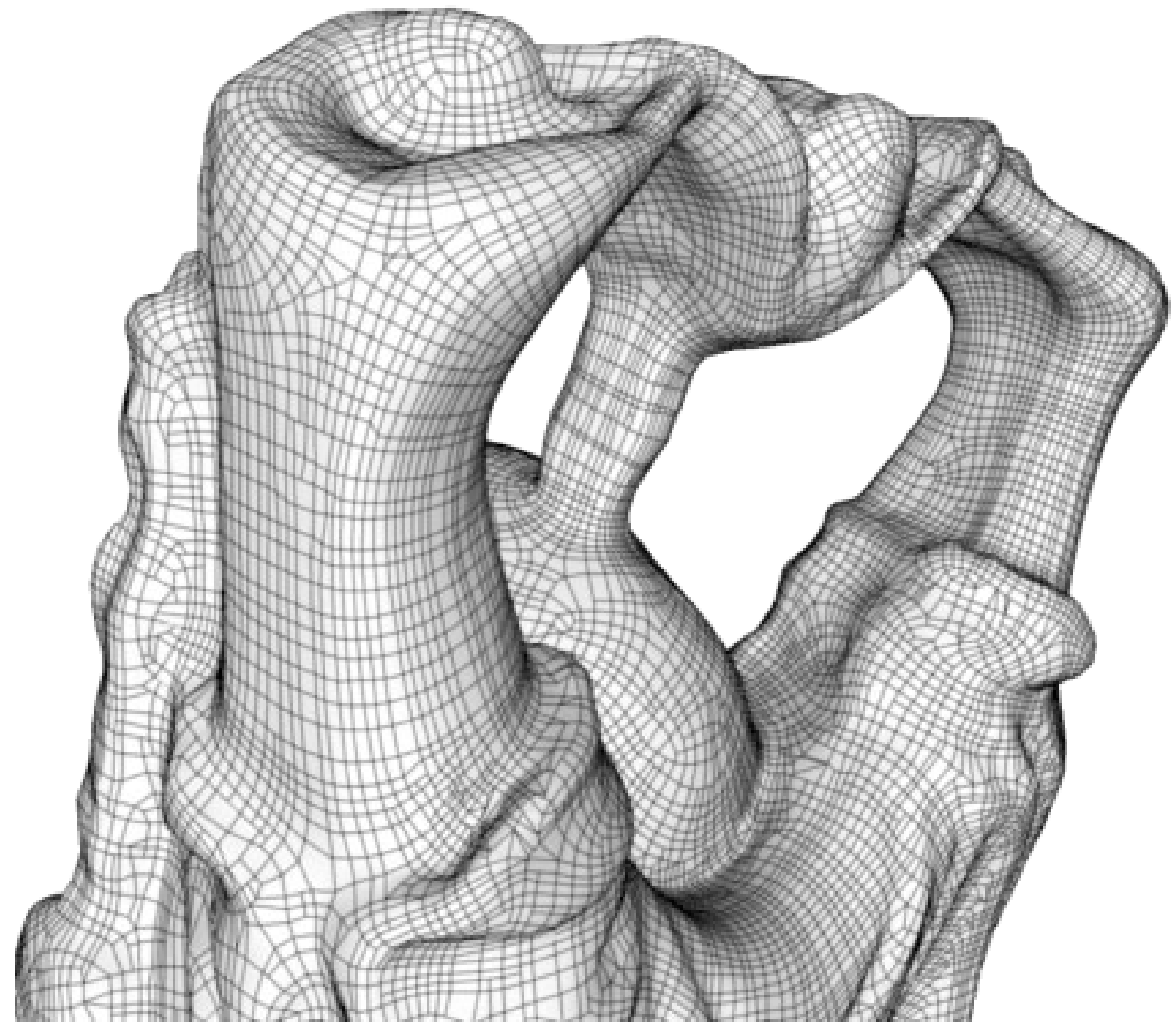


Quad Dominant Meshing

Quad Dominant Meshing

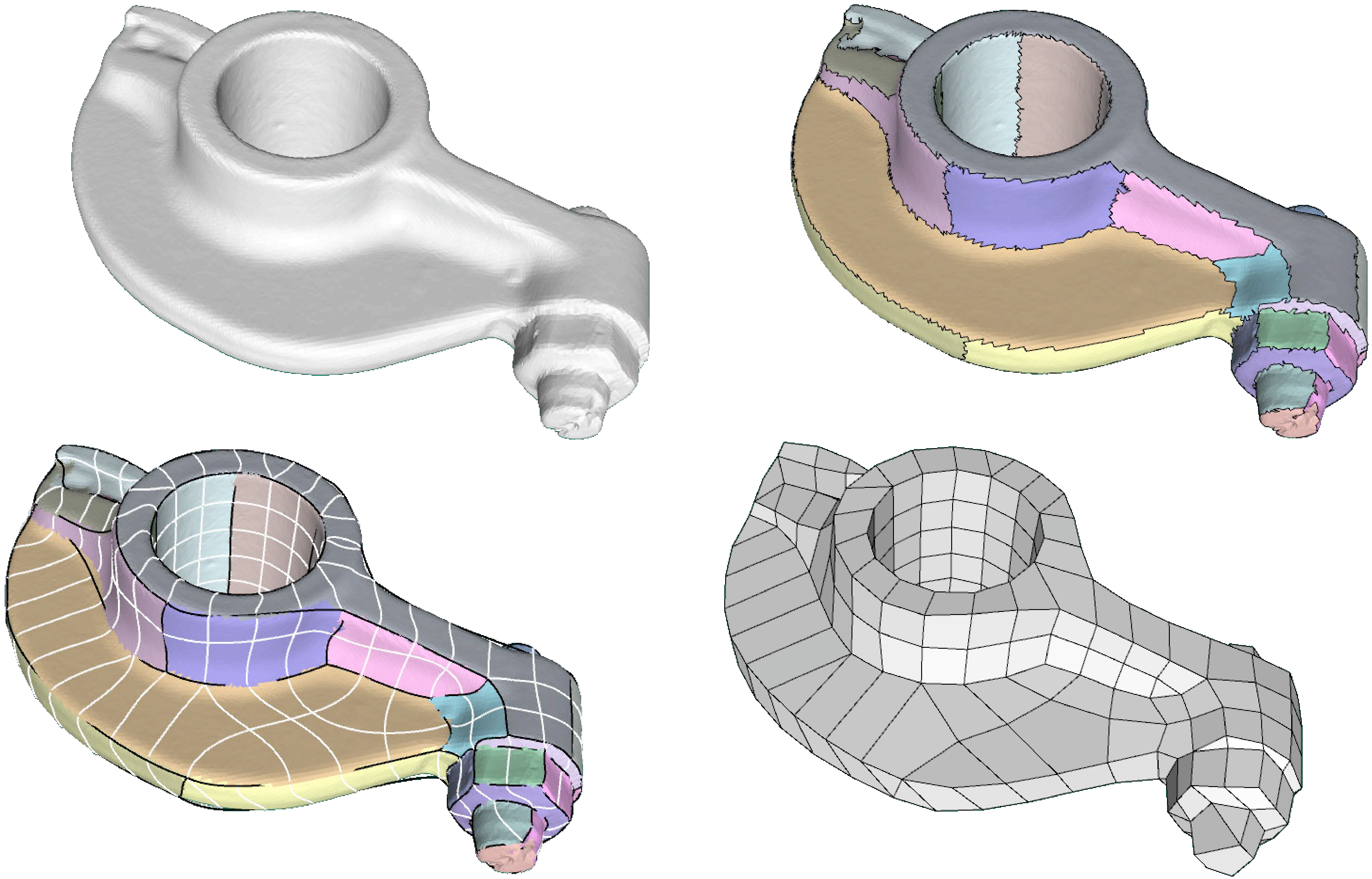




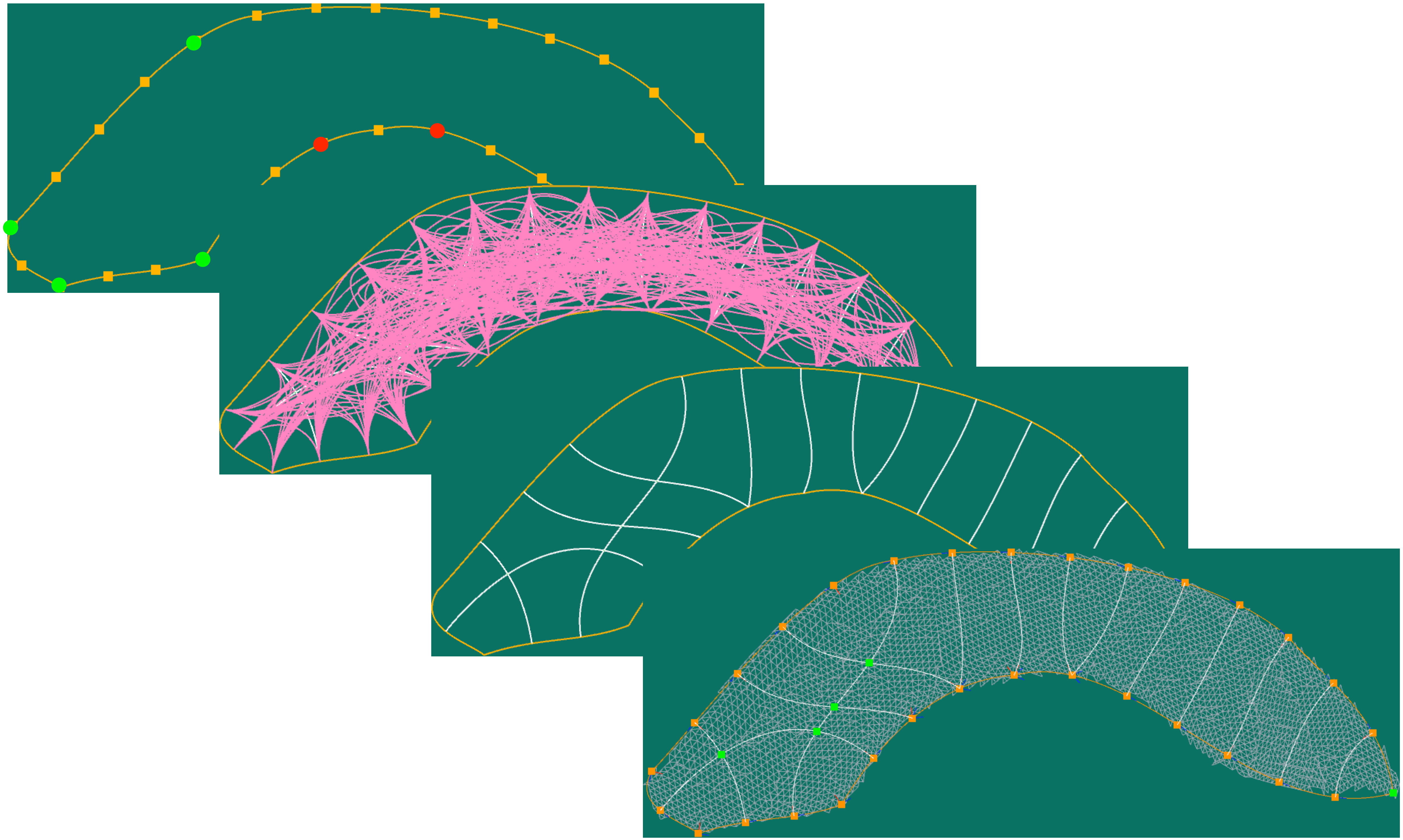
Global Alignment

- marching techniques cannot capture the global structure of the model
- two-step procedure:
  - segmentation (global structure)
  - quad meshing per segment (local shape and alignment)

Global Alignment

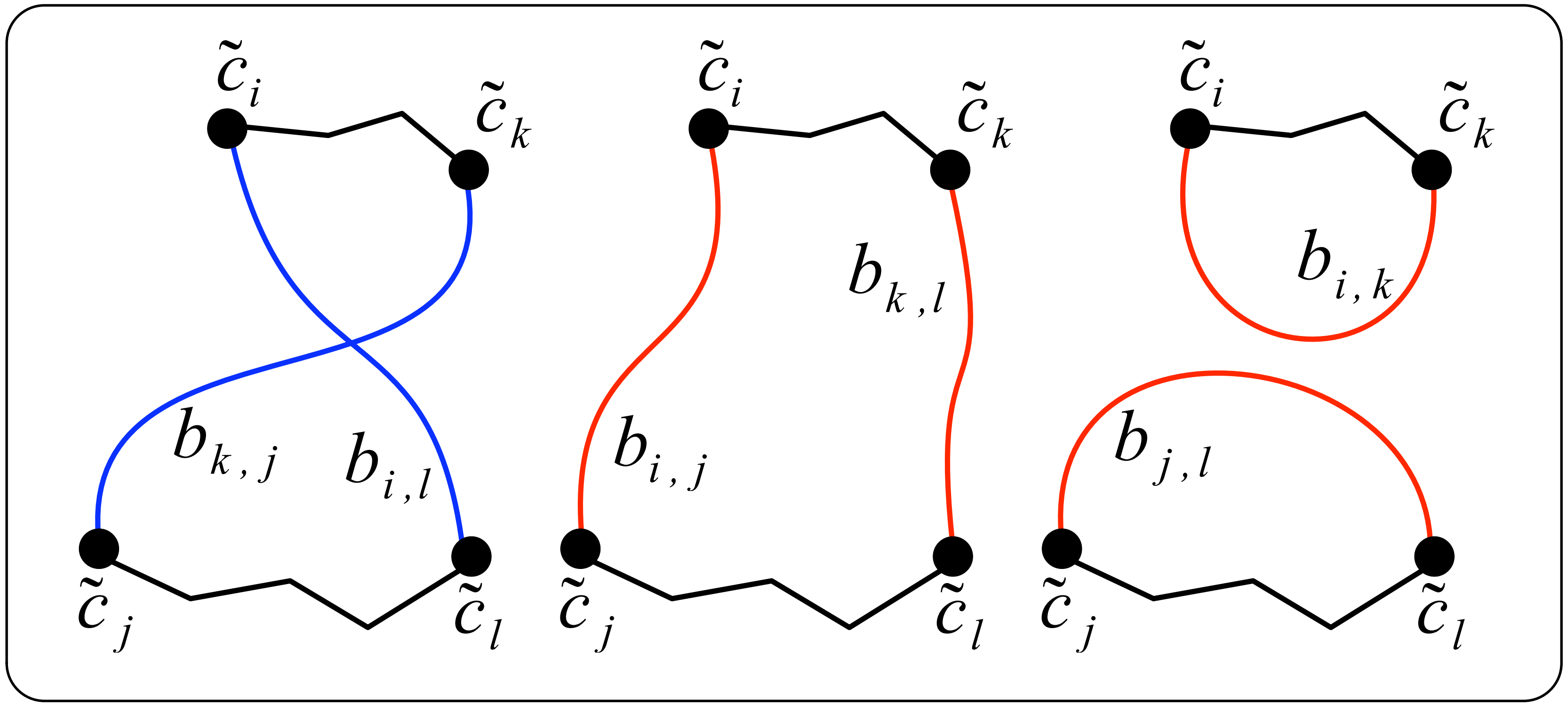


Per-Segment Optimization

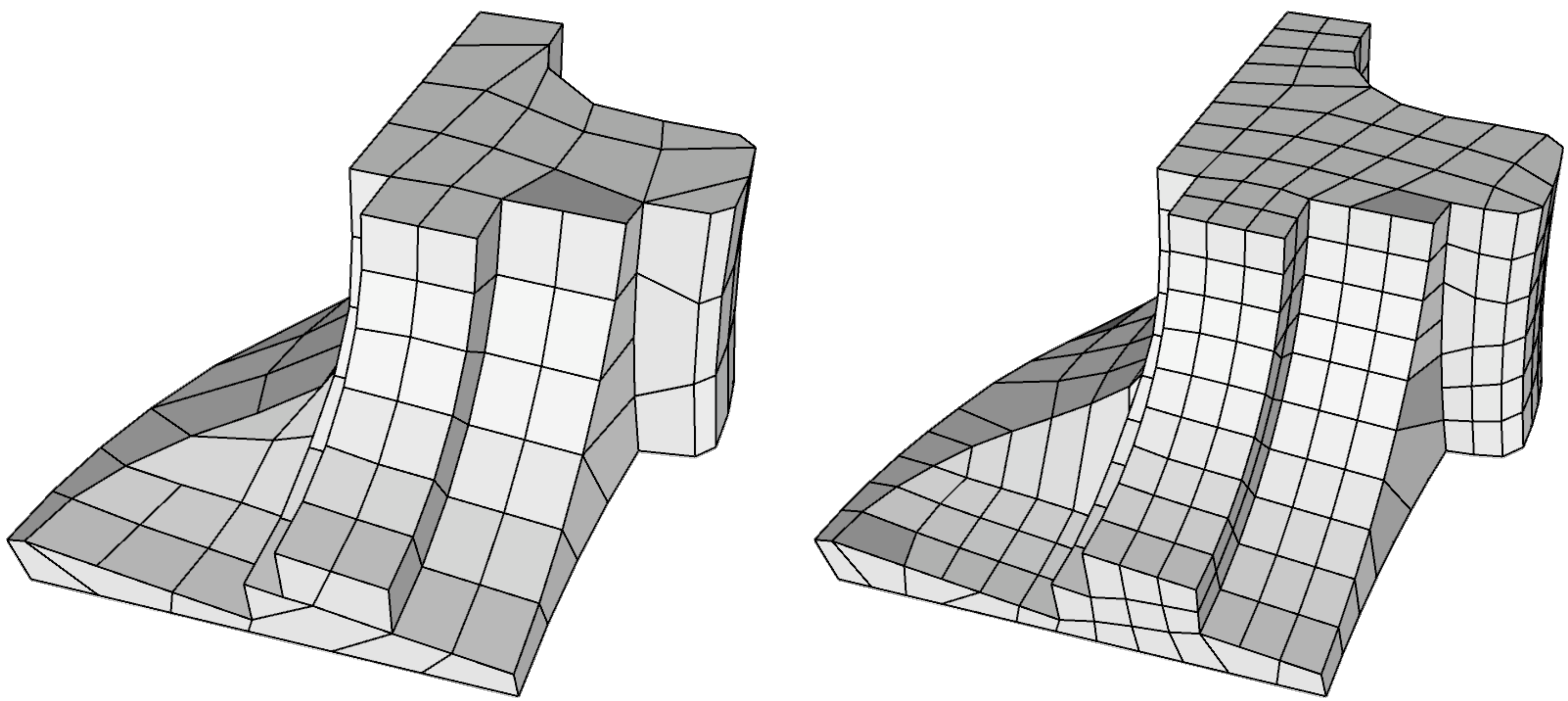


Per-Segment Optimization

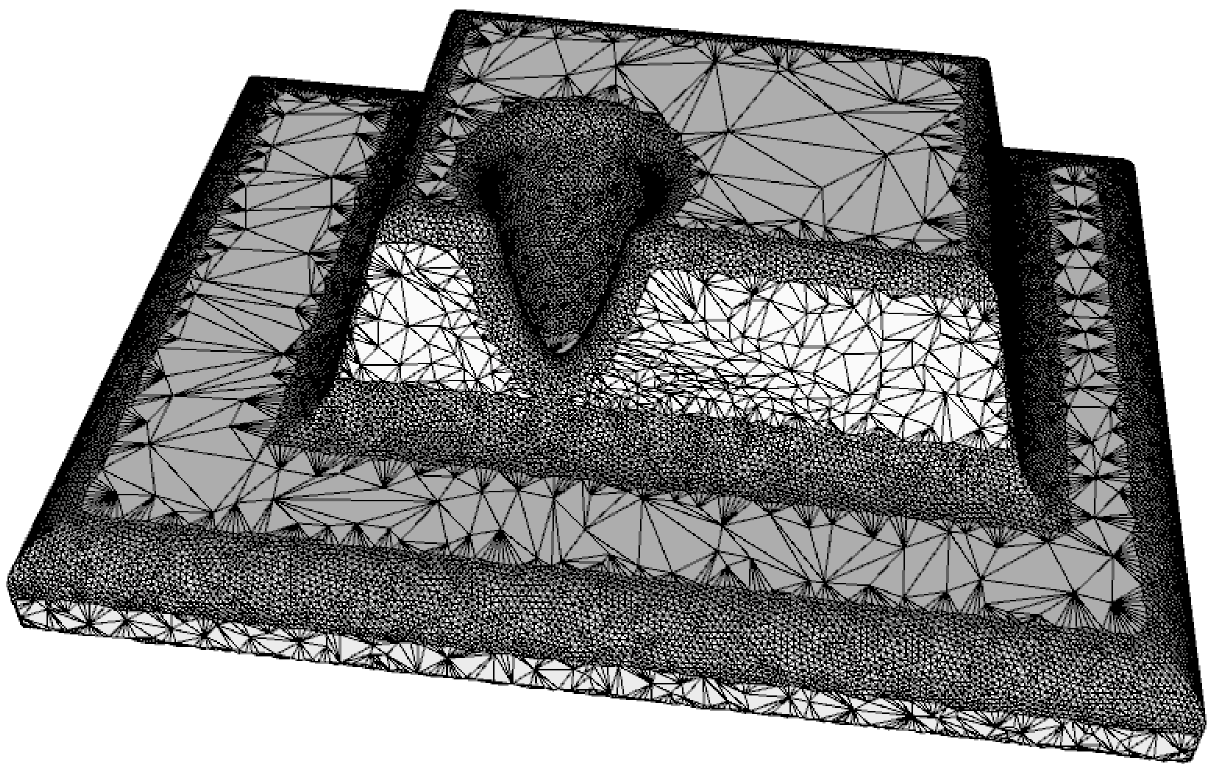
- combinatorial optimization
- energy functional
  - orthogonality at intersections
  - parallelism within faces

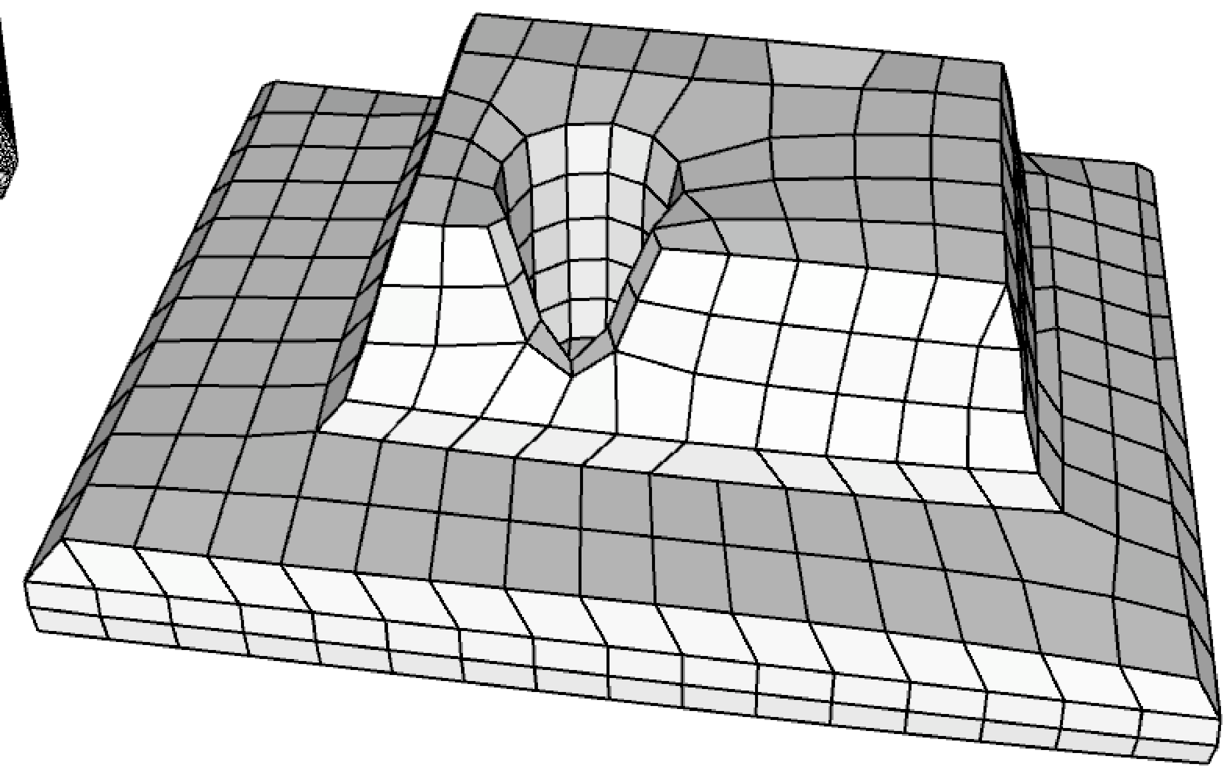


Quad-Meshing Results

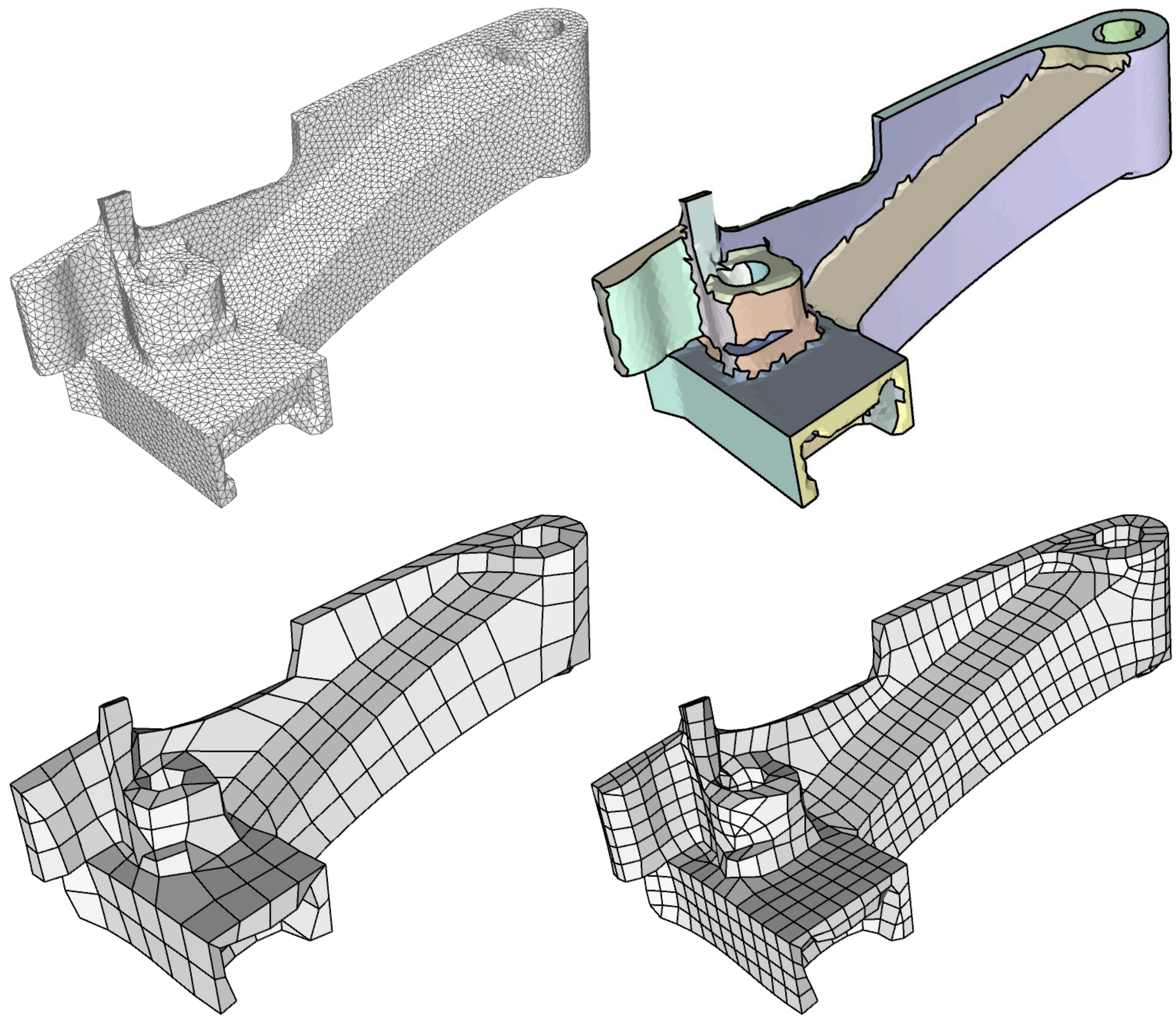


Quad-Meshing Results





Quad-Meshing Results



Quad Dominant Meshing

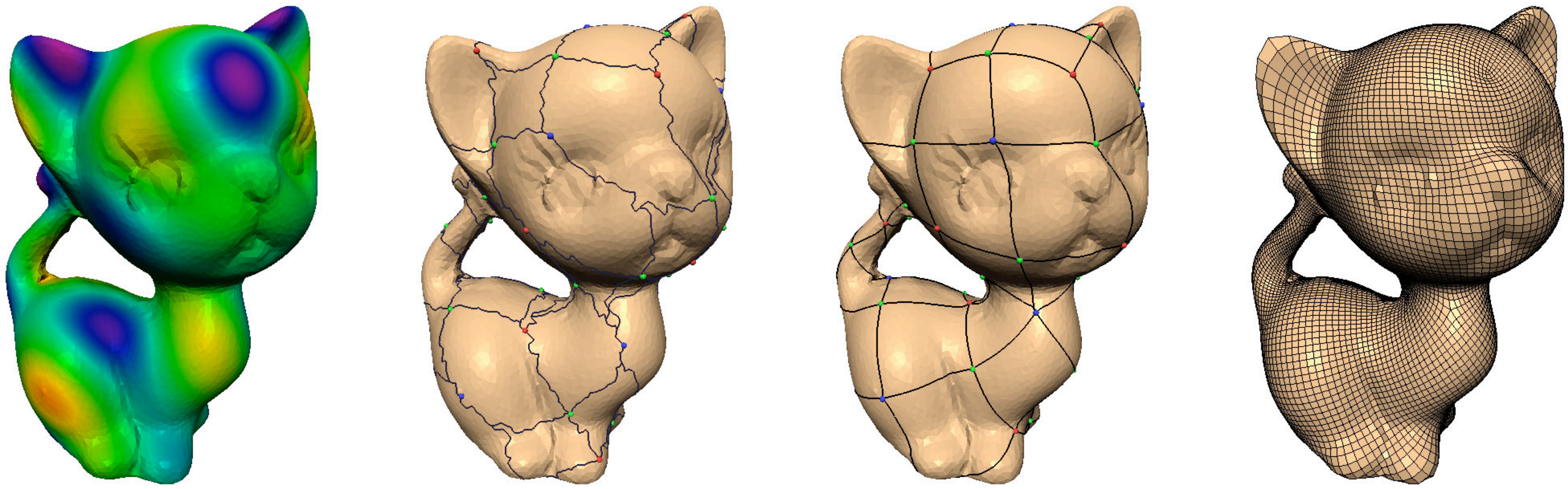
- with or without **parametrization**
- anisotropic **vertex distribution** by controlling the density of curvature lines
- **local alignment** by intersecting curvature lines
- **global alignment** by segmentation

Globally Harmonic Meshing

- generate patch layout
  - quad-dominant
  - quad-only
- compute a harmonic map per patch
  - discontinuities across patch boundaries
- globally smooth parameterization
  - “hide” discontiuunities by transfer functions between patches

Patch Layout Generation

- manually ...
- segmentation based
- using the Laplace eigenmodes
  [S. Dong et al. *"Spectral Surface Quadrangulation"*]



Harmonic Parametrization

- find a 2D parameter $\mathbf{u}_i$ for each 3D vertex $\mathbf{p}_i$
- let

$$U(\mathbf{p}_i) \,=\, \mu_i \sum_j \omega_{i,j} \left(\mathbf{p}_j - \mathbf{p}_i\right)$$

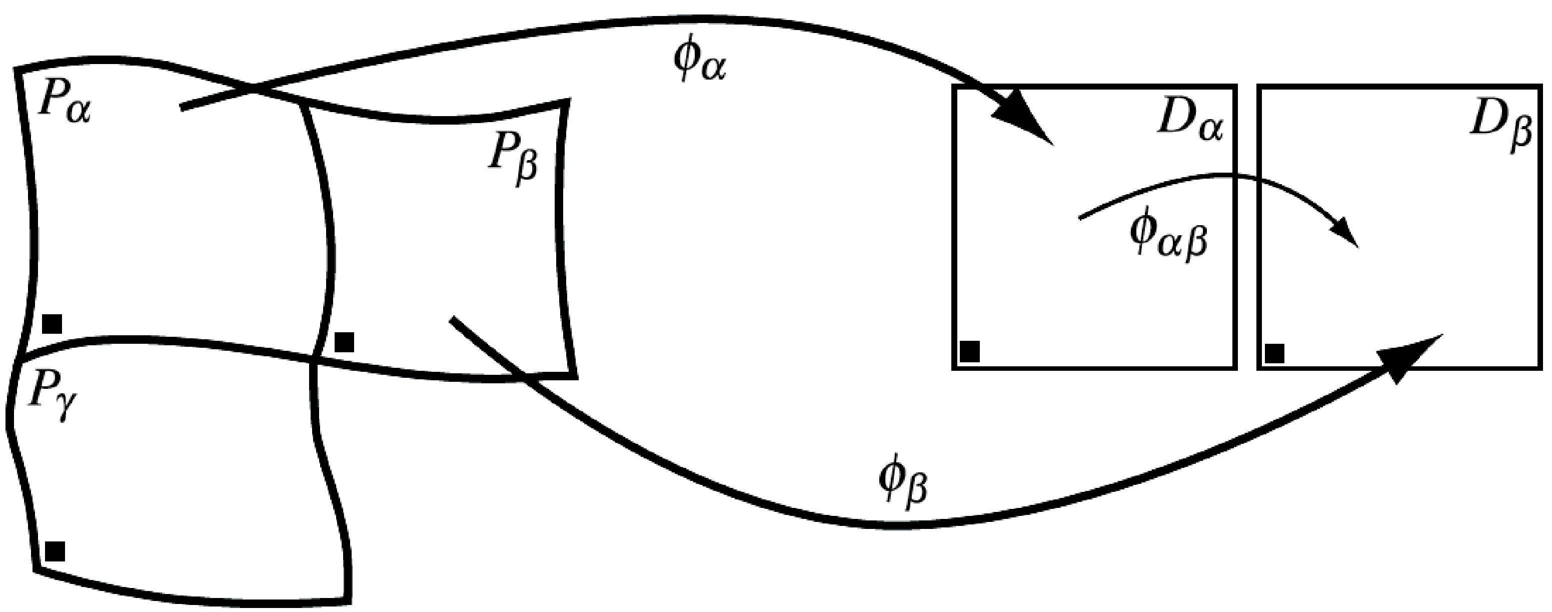
  be the Laplace-Beltrami operator defined on the surface.

- harmonic condition:

$$U(\mathbf{u}_i) \,=\, \mu_i \sum_j \omega_{i,j} \left(\mathbf{u}_j - \mathbf{u}_i\right) \,=\, 0$$

Transition Functions

- inverse parametrization: $\Phi_\alpha : R^3 \rightarrow R^2$
- transition function: $\Phi_{\alpha\beta} : R^2 \rightarrow R^2$



[S. Dong et al. *"Spectral Surface Quadrangulation"*]

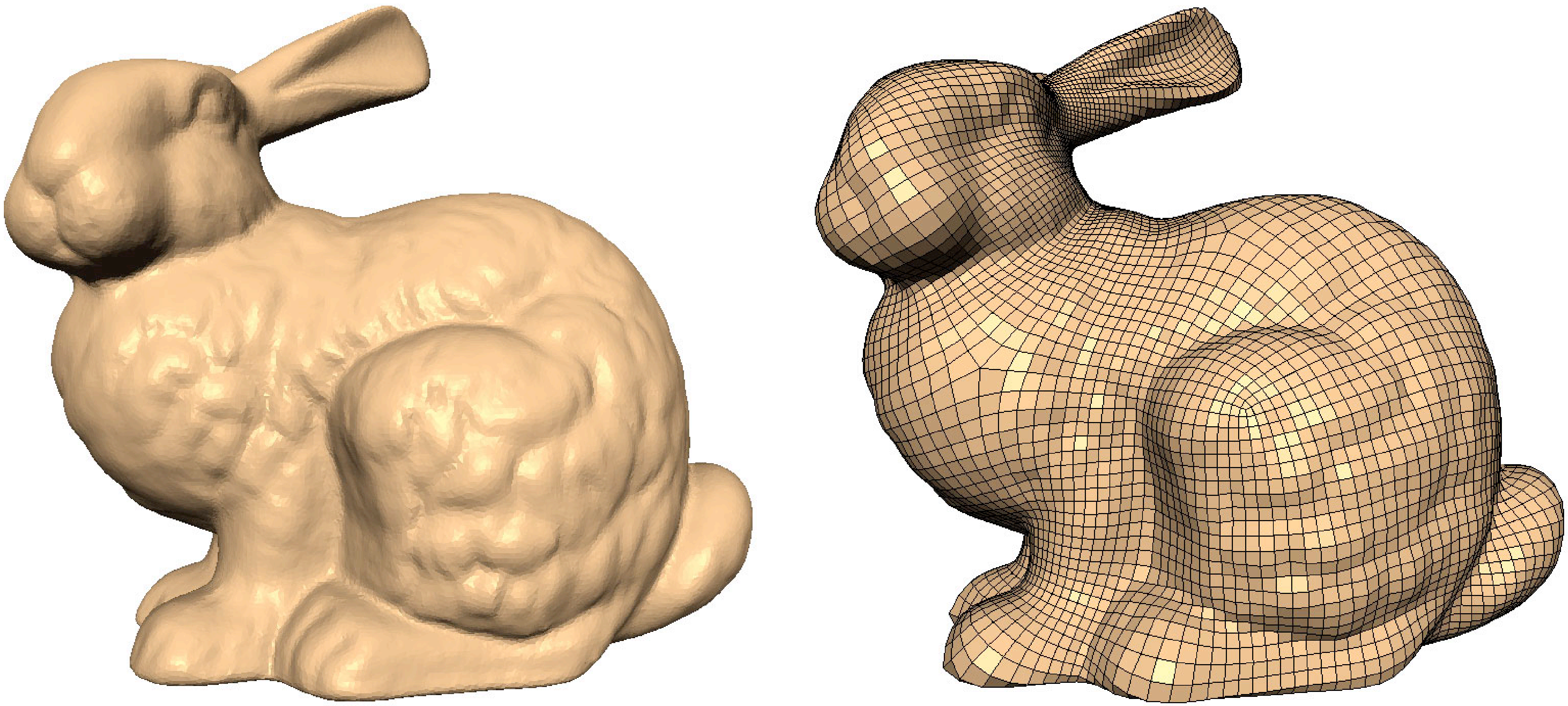
Globally Smooth Parametrization

- harmonic condition

$$U(\mathbf{u}_i^\alpha) \;=\; \mu_i \sum_j \omega_{i,j} \left(\phi_{\beta\alpha}(\mathbf{u}_j^\beta) - \mathbf{u}_i^\alpha\right) \;=\; 0$$

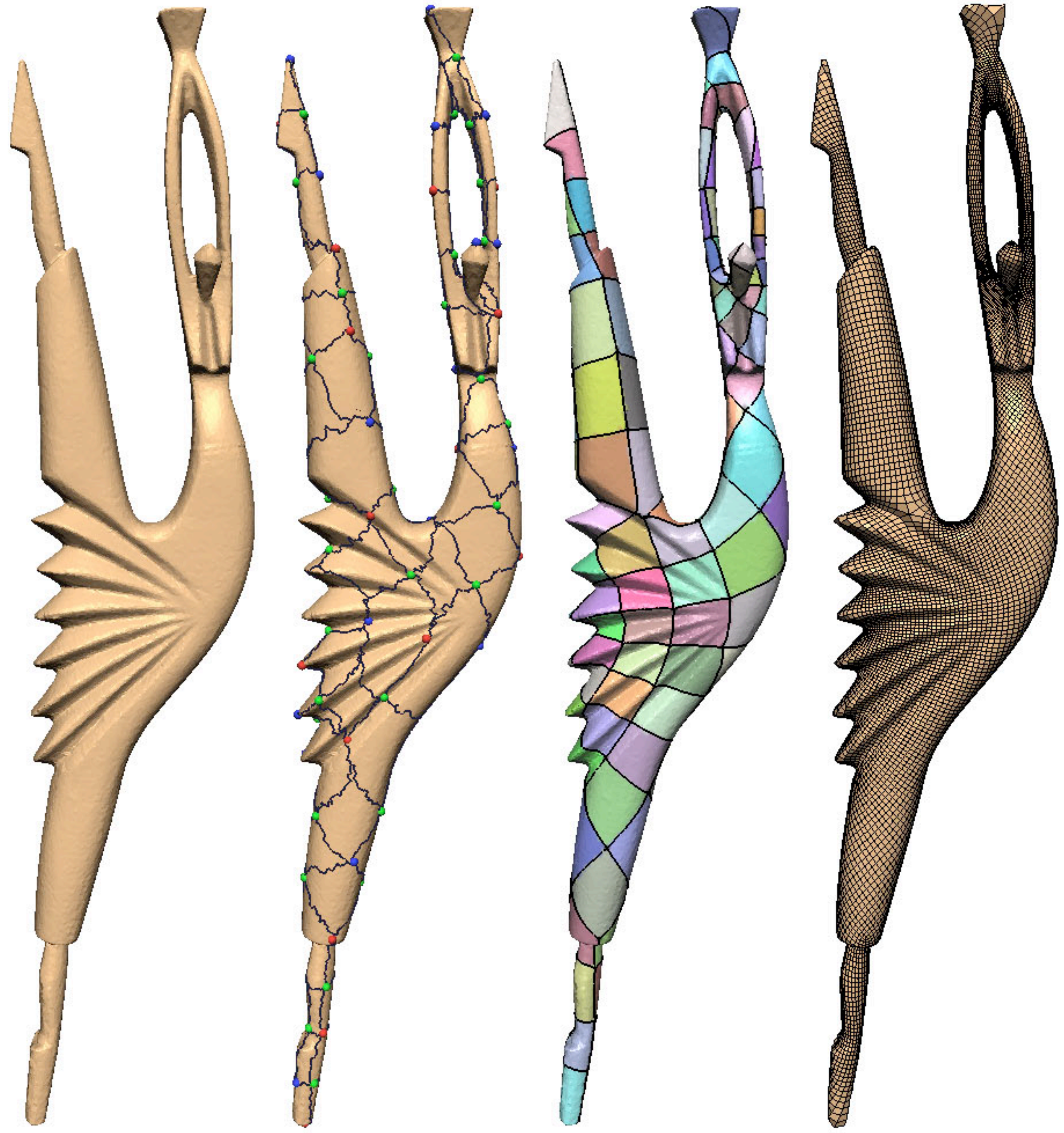
- parameter values for the patch corners are fixed to (0,0), (0,1), (1,0), or (1,1)
- solve sparse linear 2n x 2n system
- iterative update of the patch layout (local parameter values have to lie in the unit square)

Results



[S. Dong et al. *“Spectral Surface Quadrangulation”*]

Results



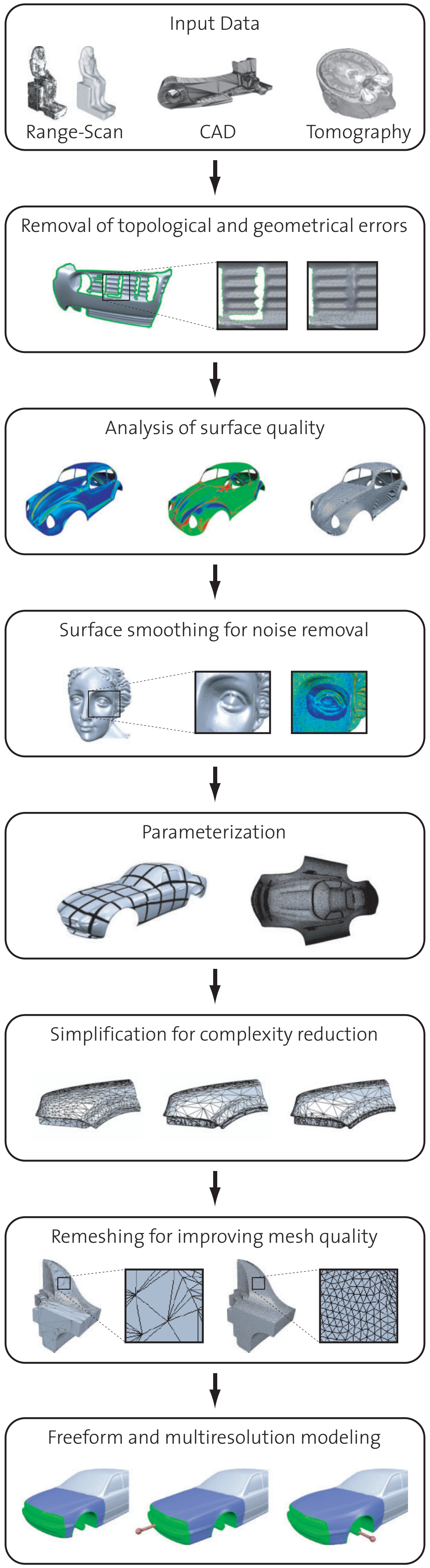
[S. Dong et al. *"Spectral Surface Quadrangulation"*]

Globally Harmonic Meshing

- **global parametrization**
- **vertex distribution** by intersection u- and v-isolines
- no **local alignment** (some alignment induced by the patch layout)
- no **global alignment** (some alignment induced by the patch layout)

Remeshing Cookbook

- problem definition
  - input, output
- basic ingredients
  - general requirements
  - types of operations
- a selection of recipies
  - various representative examples of known remeshing schemes



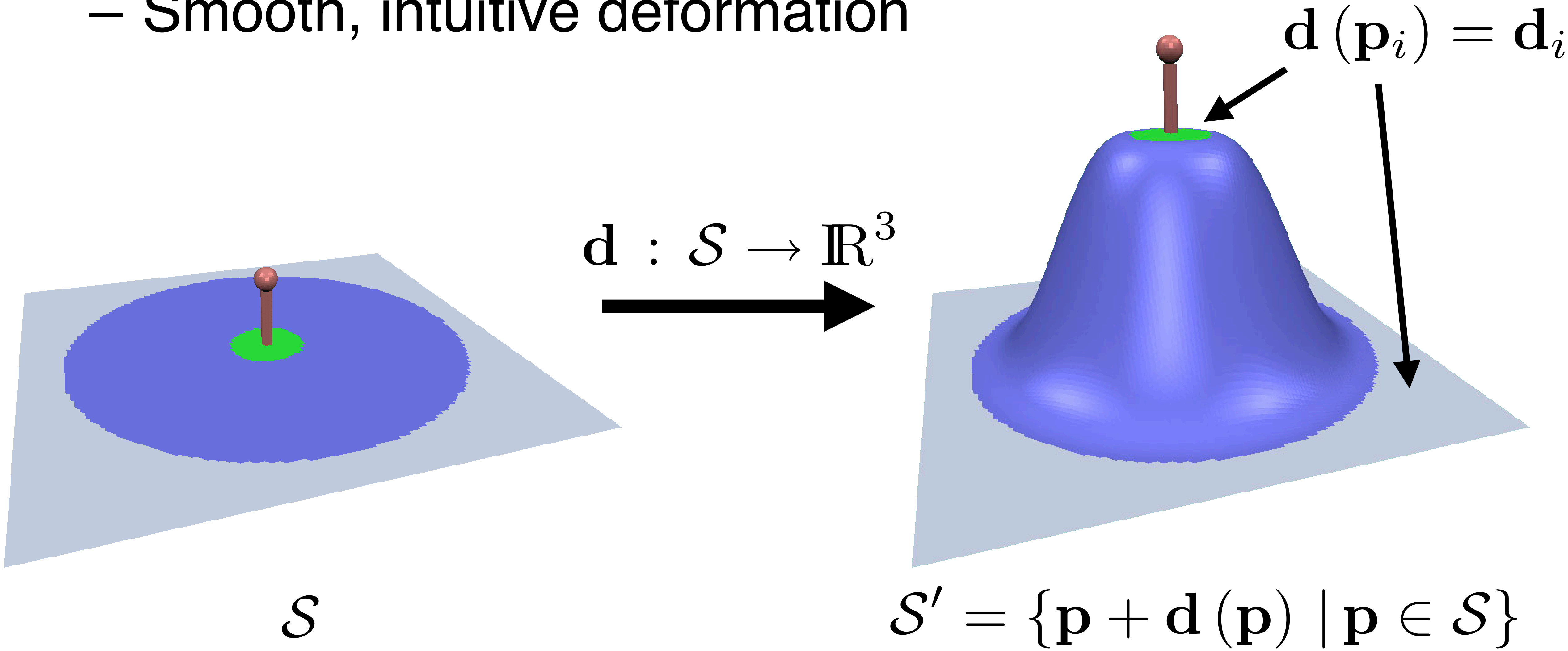
Mesh Editing

Mario Botsch

ETH Zurich

Mesh Editing

- Mesh deformation by displacement function **d**
  - Interpolate prescribed constraints
  - Smooth, intuitive deformation

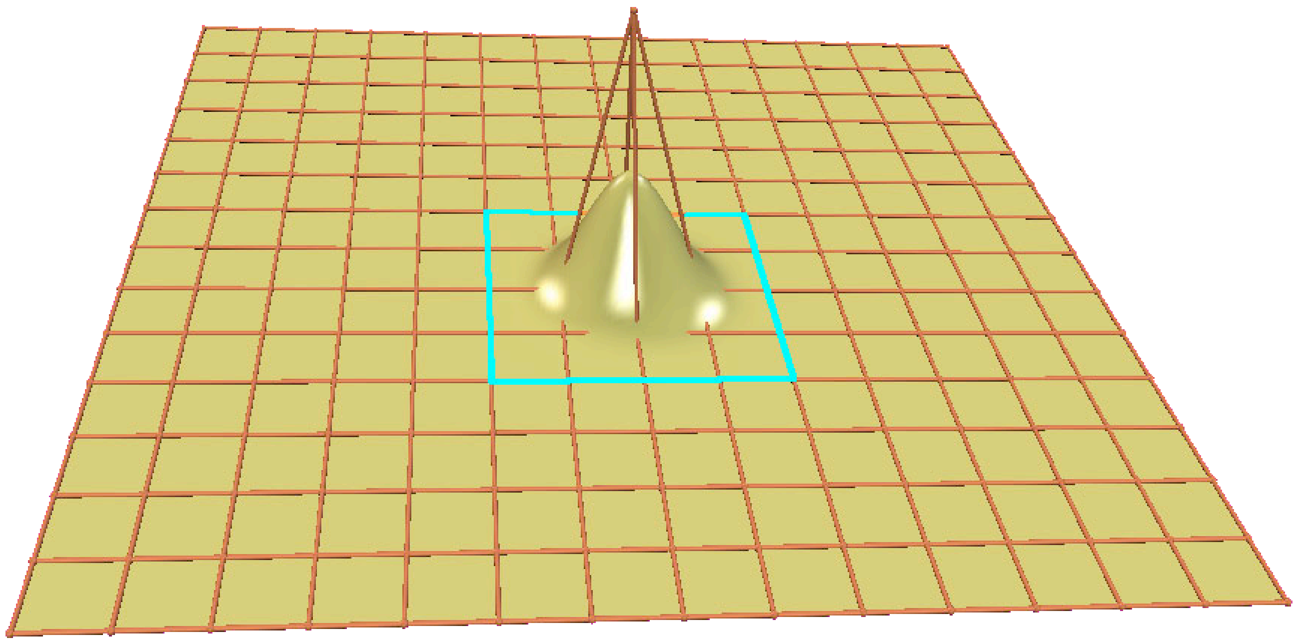


Overview

- **Surface-Based Deformation**
- Space Deformation
- Multiresolution Deformation
- Differential Coordinates
- Comparison

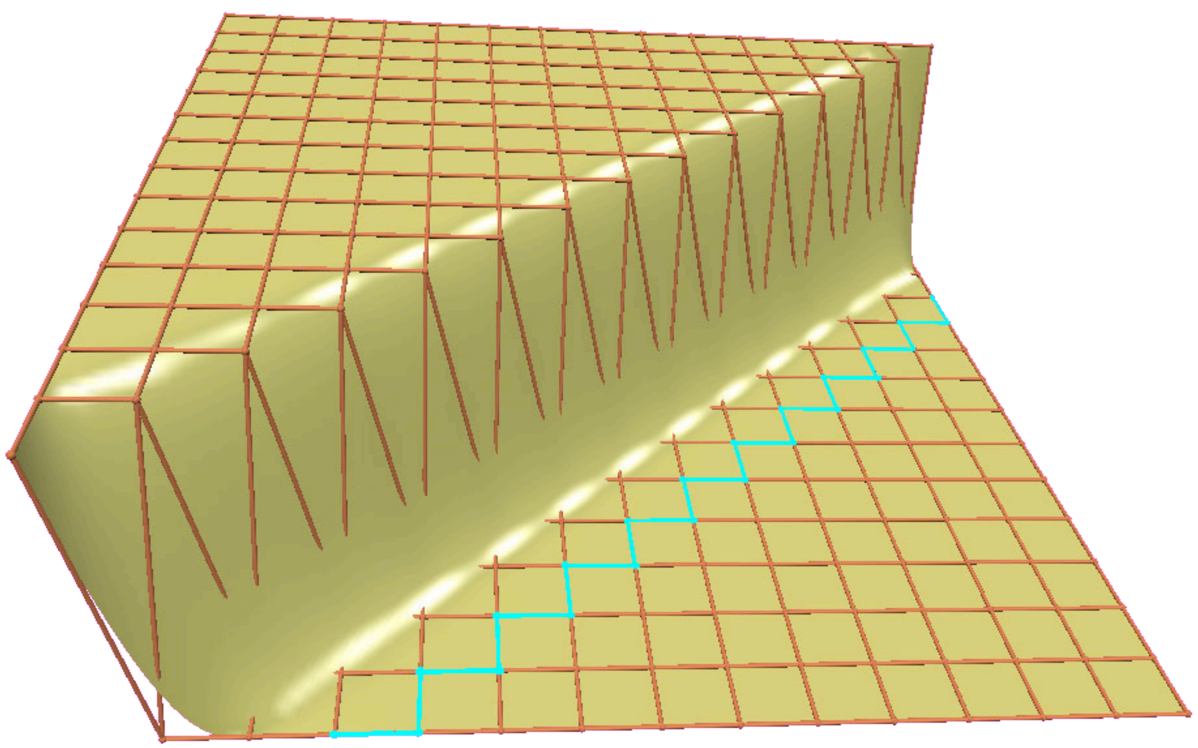
Spline Surfaces

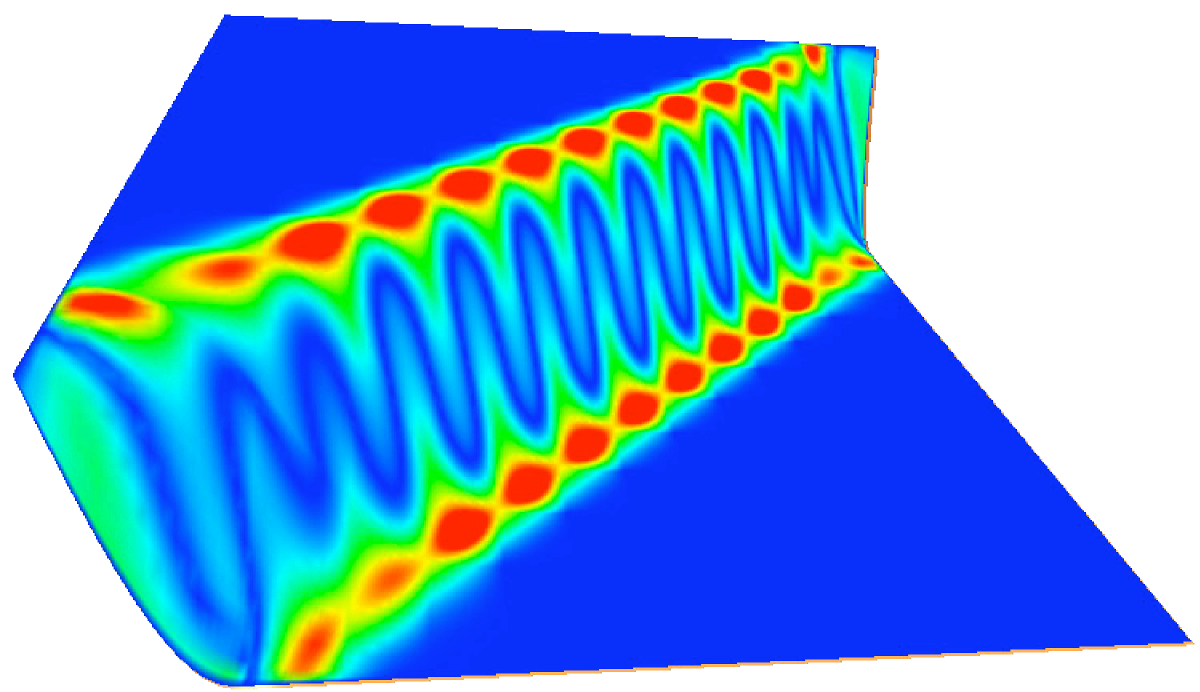
- Basis functions are smooth bumps



Spline Surfaces

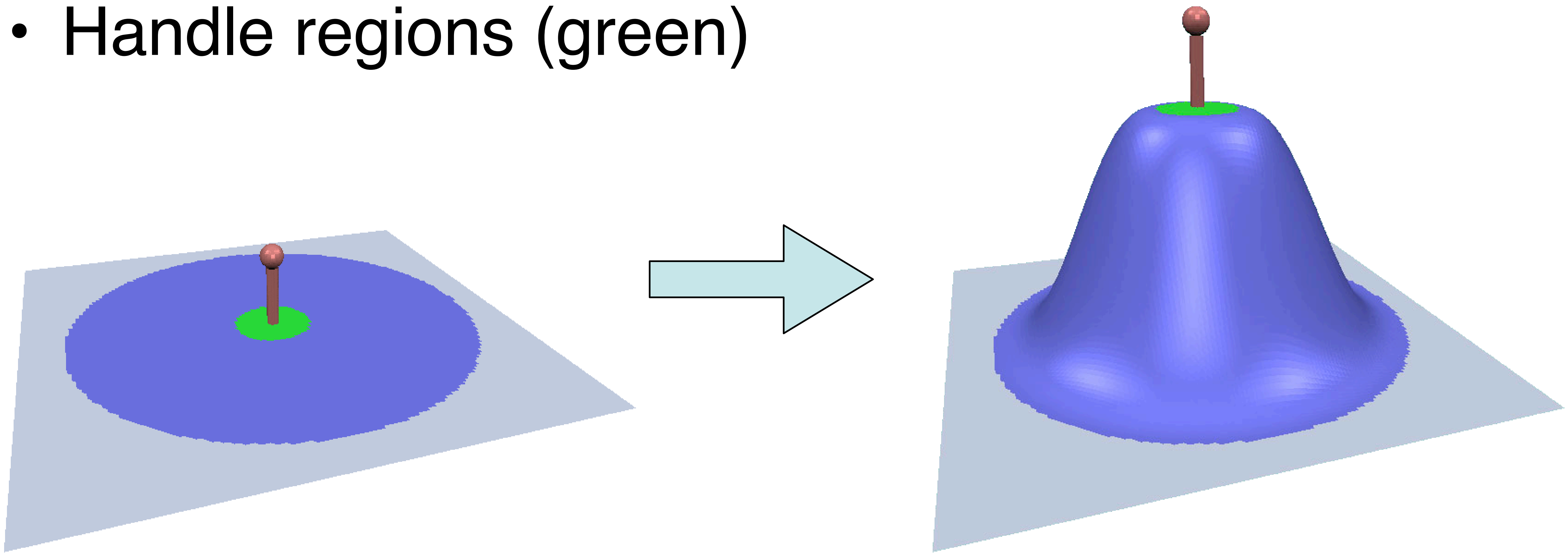
- Basis functions are smooth bumps
  - Fixed support
  - Regular grid





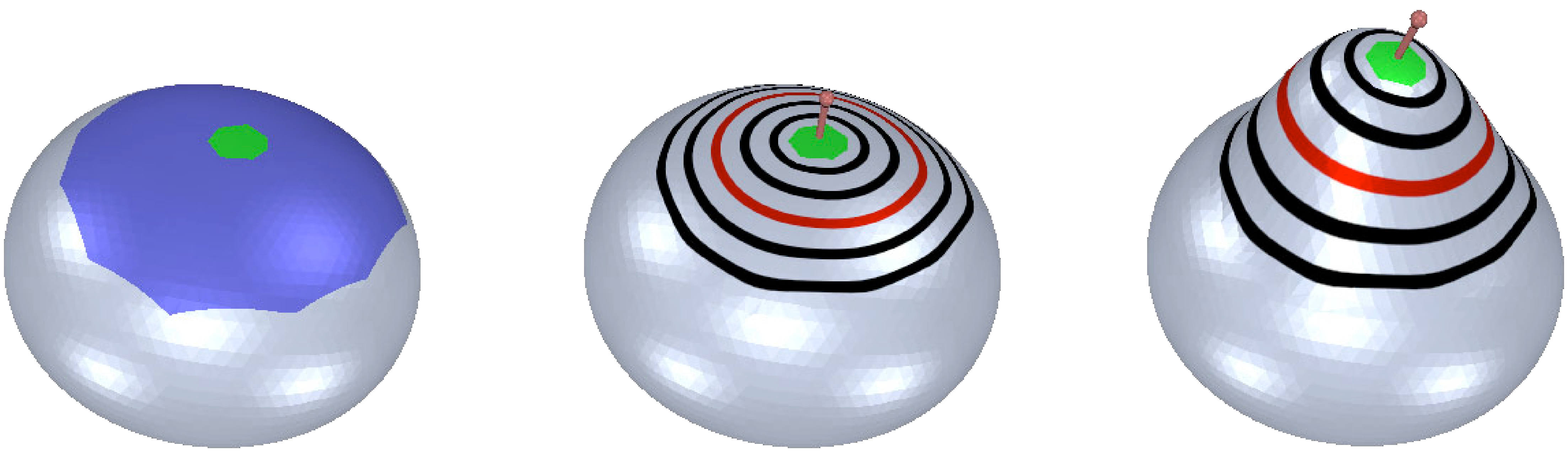
Modeling Metaphor

- Support region (blue)
- Fixed vertices (gray)
- Handle regions (green)

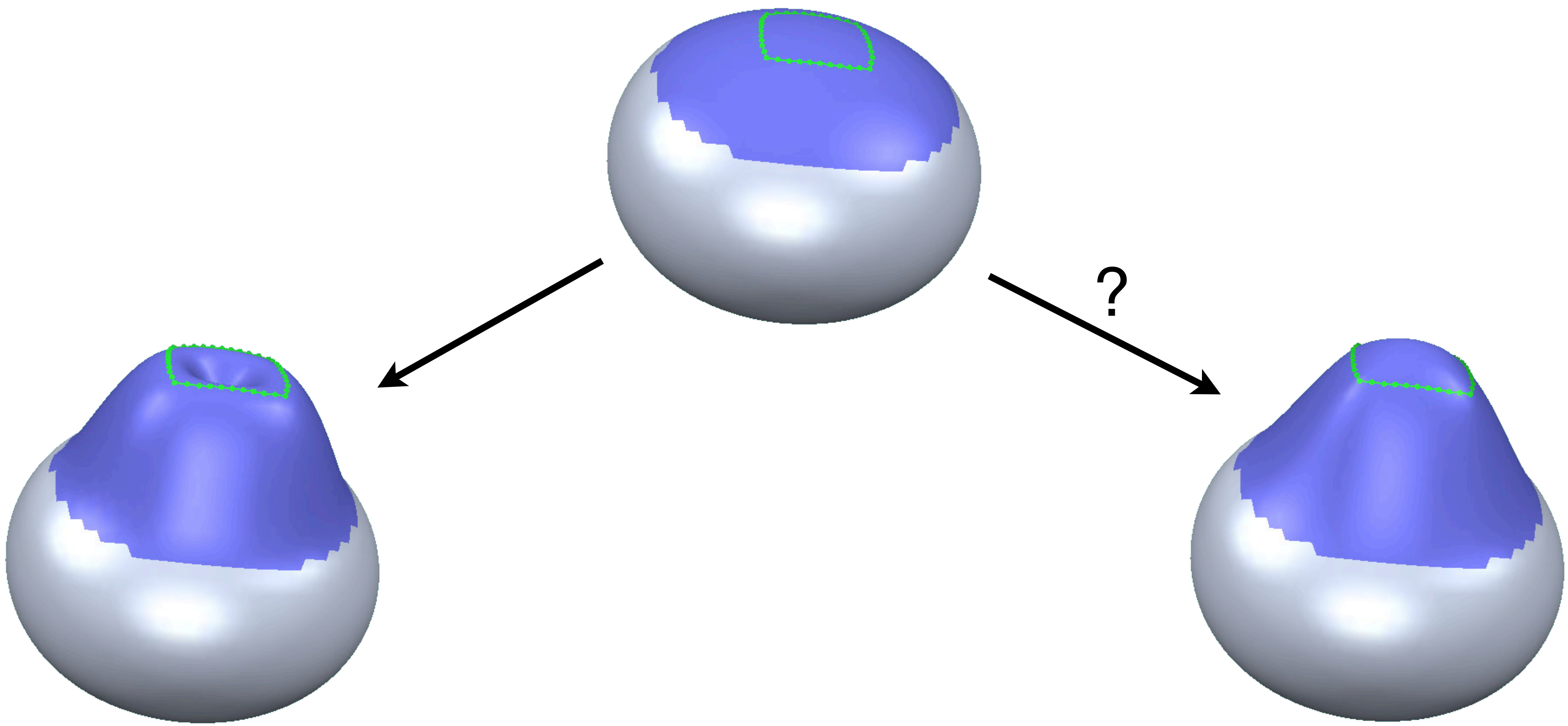


Distance-Based Propagation

1. Construct smooth scalar field $s : \mathcal{S} \rightarrow [0, 1]$
   - s(**p**)=1: Full deformation at handle
   - s(**p**)=0: No deformation for fixed part
   - s(**p**)∈(0,1): Smooth blending inbetween
2. Damp handle transformation with s(**p**)



Distance-Based Propagation



Distance-based propagation

Smooth interpolation

Boundary Constraint Modeling

1. Control: Prescribe *arbitrary* constraints:

$$\mathbf{d}\left(\mathbf{p}_i\right) = \mathbf{d}_i \,, \quad \forall \mathbf{p}_i \in \mathcal{C}$$

2. Fitting: Smoothly interpolate constraints by a displacement function:

$$\mathbf{d} : \mathcal{S} \rightarrow \mathbb{R}^3 \quad \text{with} \quad \mathbf{d}\left(\mathbf{p}_i\right) = \mathbf{d}_i$$

3. Evaluation: Displace all points:

$$\mathbf{p}_i \mapsto \mathbf{p}_i + \mathbf{d}\left(\mathbf{p}_i\right) \quad \forall \mathbf{p}_i \in \mathcal{S}$$

How to interpolate?

- Constrained bending energy minimization

$$\int_{\mathcal{S}} \left\|\mathbf{d}_{uu}\right\|^2 + 2\left\|\mathbf{d}_{uv}\right\|^2 + \left\|\mathbf{d}_{vv}\right\|^2 d\mathcal{S}$$

- Variational calculus, Euler-Lagrange PDE

$$\Delta_{\mathcal{S}}^2 \mathbf{d} \equiv 0 \quad \text{with} \quad \mathbf{d}\left(\mathbf{p}_i\right) = \mathbf{d}_i,\ \forall \mathbf{p}_i \in \mathcal{C}$$

- “Best” deformation which satisfies constraints

Physical Interpretation

- Non-linear stretching & bending energies

$$\int_\Omega k_s \left\|\mathbf{I} - \mathbf{I}'\right\|^2 + k_b \left\|\mathbf{II} - \mathbf{II}'\right\|^2 \; dudv$$

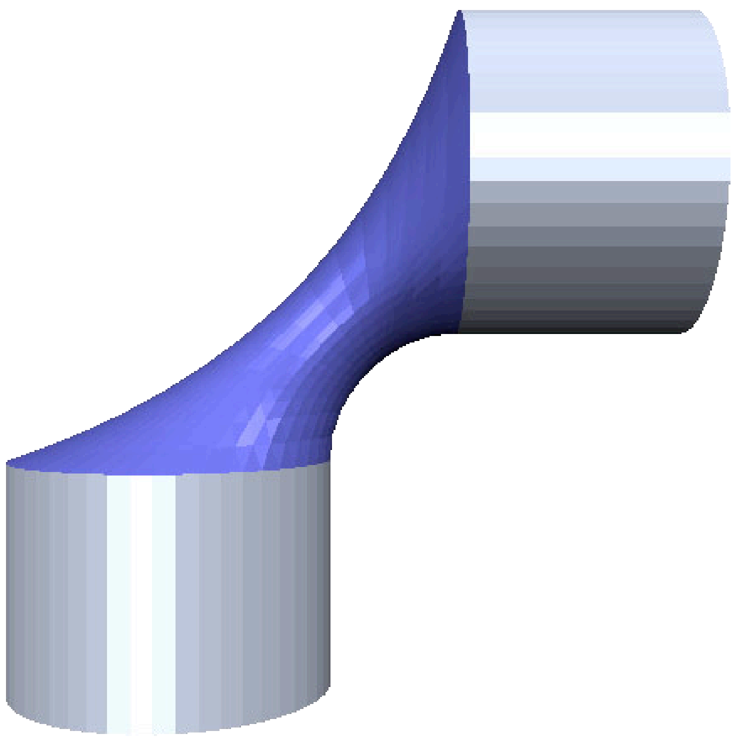
- Linearize energies

$$\int_\Omega k_s \left(\|\mathbf{d}_u\|^2 + \|\mathbf{d}_v\|^2\right) \; + \; k_b \left(\|\mathbf{d}_{uu}\|^2 + 2\,\|\mathbf{d}_{uv}\|^2 + \|\mathbf{d}_{vv}\|^2\right) dudv$$

- Euler-Lagrange PDE

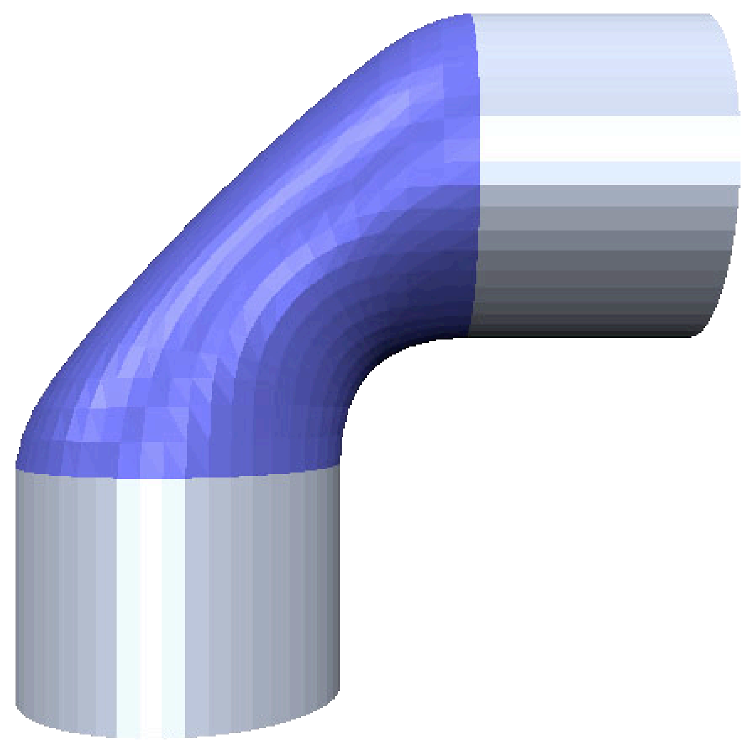
$$k_s\,\Delta\mathbf{d} + k_b\,\Delta^2\mathbf{d} \equiv 0$$

Deformation Energies



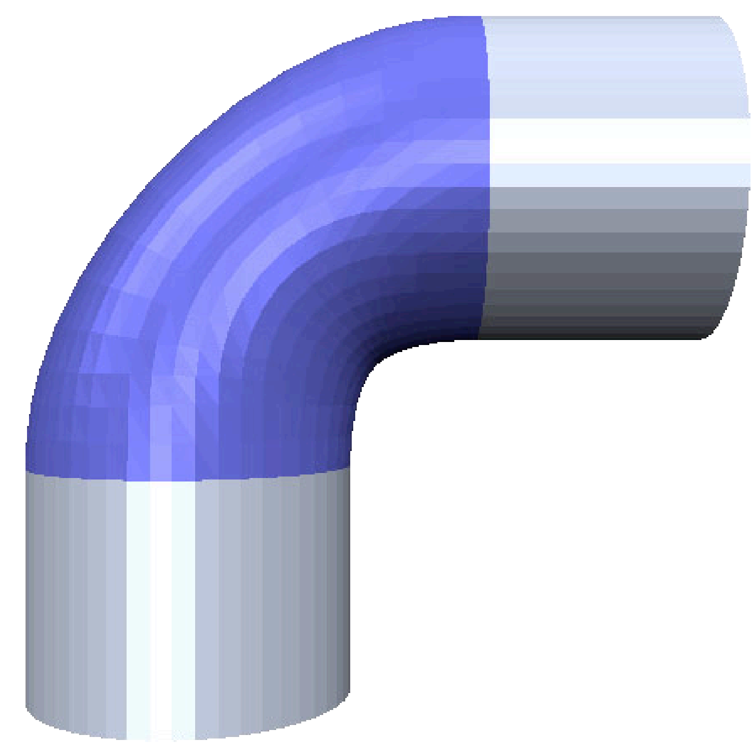
$$\Delta \mathbf{x} \equiv 0$$

(Membrane)



$$\Delta^2 \mathbf{x} \equiv 0$$

(Thin plate)



$$\Delta^3 \mathbf{x} \equiv 0$$

Discretization

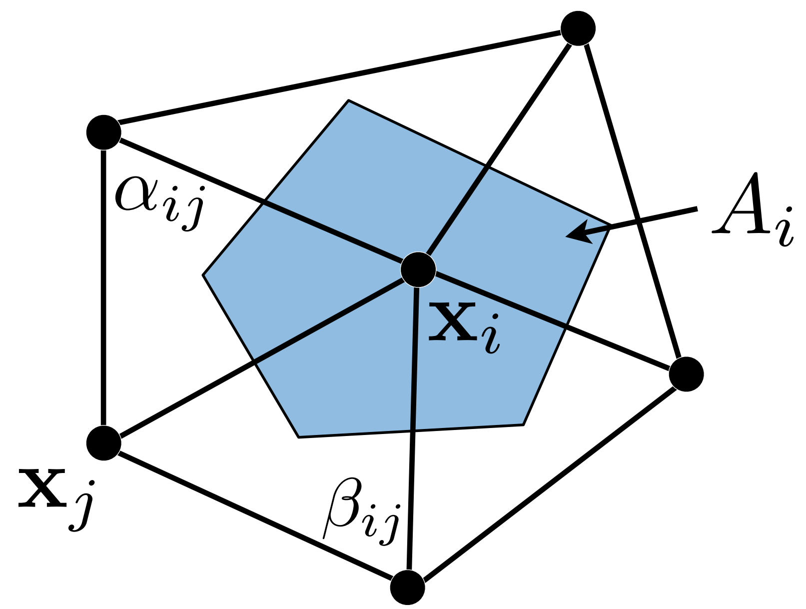
- Euler-Lagrange PDE

$$\Delta_{\mathcal{S}}^{k}\mathbf{d} \equiv \mathbf{0} \quad \text{with} \quad \mathbf{d}\left(\mathbf{p}_i\right) = \mathbf{d}_i, \quad \forall \mathbf{p}_i \in \mathcal{C}$$

- Finite difference Laplace discretization

$$\Delta_{\mathcal{S}}^{k}\mathbf{d}_i = \frac{1}{2A_i} \sum_{j \in \mathcal{N}(i)} \left(\cot\alpha_{ij} + \cot\beta_{ij}\right)\left(\Delta_{\mathcal{S}}^{k-1}\mathbf{d}_j - \Delta_{\mathcal{S}}^{k-1}\mathbf{d}_i\right)$$

$$\Delta_{\mathcal{S}}^{0}\mathbf{d}_i = \mathbf{d}_i$$



Discretization

- Euler-Lagrange PDE

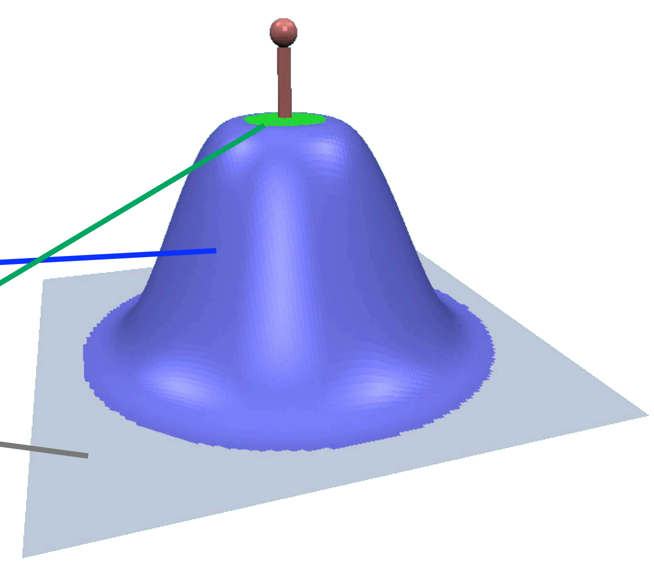
$$\Delta_{\mathcal{S}}^k \mathbf{d} \equiv \mathbf{0} \quad \text{with} \quad \mathbf{d}\left(\mathbf{p}_i\right) = \mathbf{d}_i\,, \quad \forall \mathbf{p}_i \in \mathcal{C}$$

- Finite difference Laplace discretization

$$\Delta_{\mathcal{S}}^k \mathbf{d}_i = \frac{1}{2A_i} \sum_{j \in \mathcal{N}(i)} \left(\cot \alpha_{ij} + \cot \beta_{ij}\right) \left(\Delta_{\mathcal{S}}^{k-1} \mathbf{d}_j - \Delta_{\mathcal{S}}^{k-1} \mathbf{d}_i\right)$$

- Sparse linear system

$$\begin{pmatrix} & \Delta^k & \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \vdots \\ \mathbf{d}_i \\ \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{h}_i' - \mathbf{h}_i \end{pmatrix}$$



Efficient Solution

- Solve linear system each frame
  - sparse, symmetric, pos. definite

$$\Delta^k \begin{pmatrix} \vdots \\ \mathbf{d}_i \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ \mathbf{b}_i \\ \vdots \end{pmatrix}$$

- Only right-hand side changes
  - Use sparse Cholesky factorization (later...)
  - Only back-substitution each frame!

More Efficient Solution

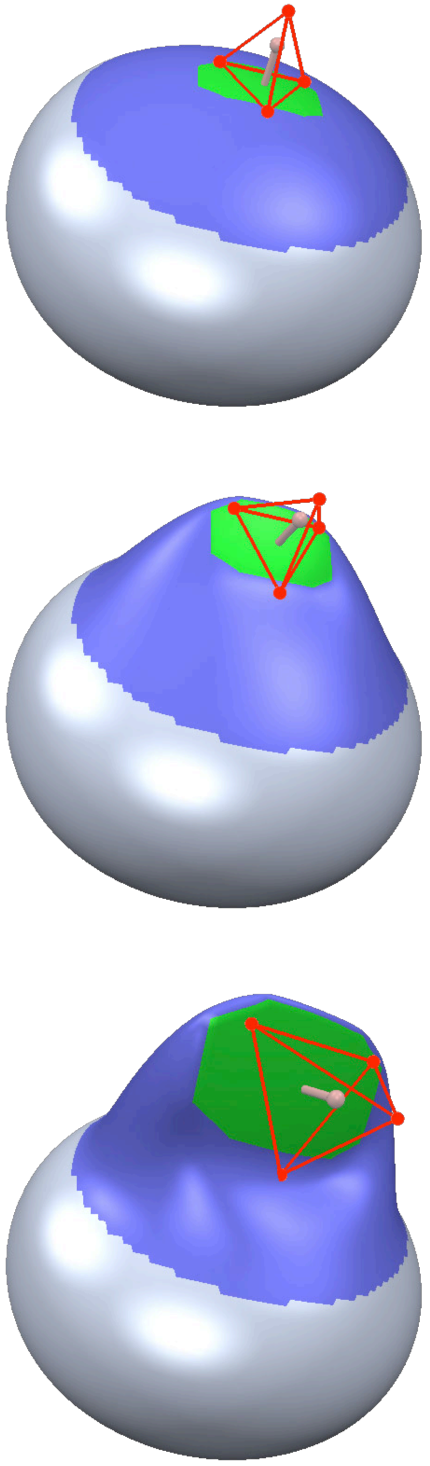
- Handle is transformed affinely only

- Represent handle points wrt. 4 points

$$\left(\ldots, \mathbf{h}_i, \ldots\right) = \mathbf{Q} \left(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}\right)^T$$

- Same for handle displacement

$$\left(\ldots, \delta\mathbf{h}_i, \ldots\right) = \mathbf{Q} \left(\delta\mathbf{a}, \delta\mathbf{b}, \delta\mathbf{c}, \delta\mathbf{d}\right)^T$$



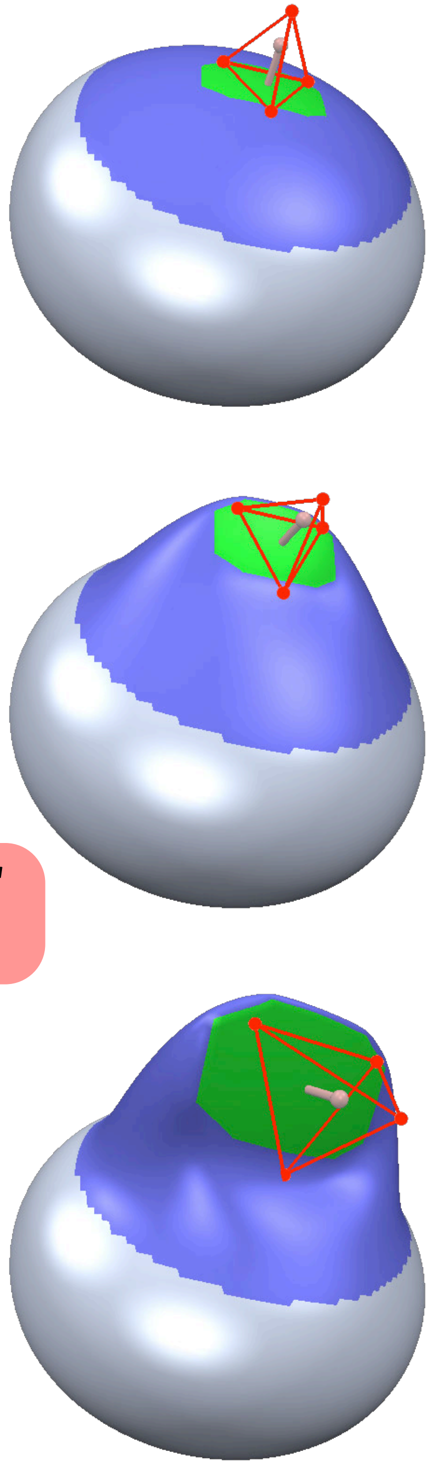
More Efficient Solution

- Precompute basis function matrix **B**

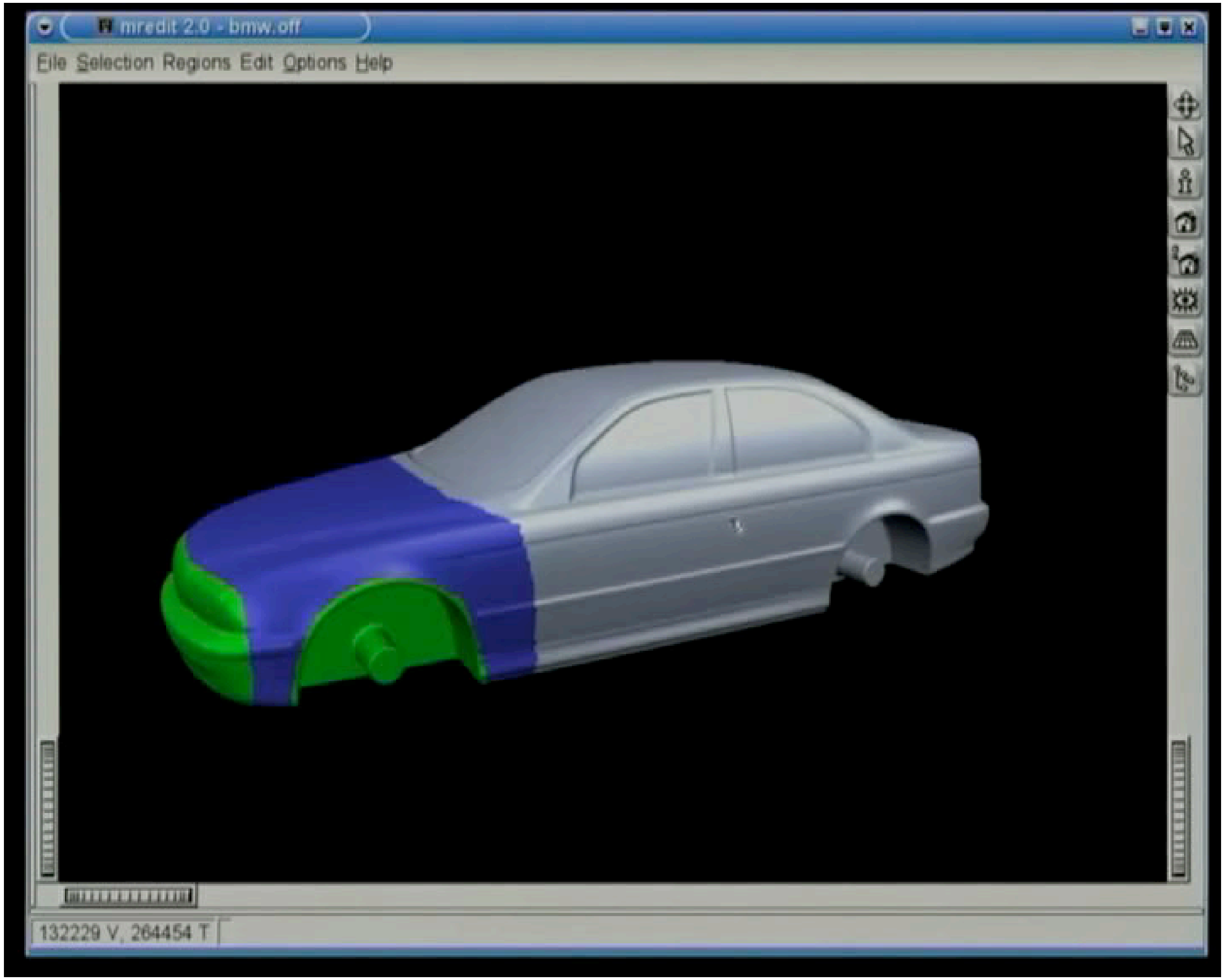
$$\underbrace{\begin{pmatrix} & \Delta^k & \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix}}_{=:\mathbf{M}} \begin{pmatrix} \vdots \\ \mathbf{d}_i \\ \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \delta\mathbf{h}_i \end{pmatrix}$$

$$\begin{pmatrix} \vdots \\ \mathbf{d}_i \\ \vdots \end{pmatrix} = \mathbf{M}^{-1} \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \delta\mathbf{h}_i \end{pmatrix} = \underbrace{\mathbf{M}^{-1} \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ Q \end{pmatrix}}_{=:\mathbf{B}} (\delta\mathbf{a}, \delta\mathbf{b}, \delta\mathbf{c}, \delta\mathbf{d},)^T$$

- **B** has 4 columns $\Rightarrow$ Solve 4 systems



Front Deformation

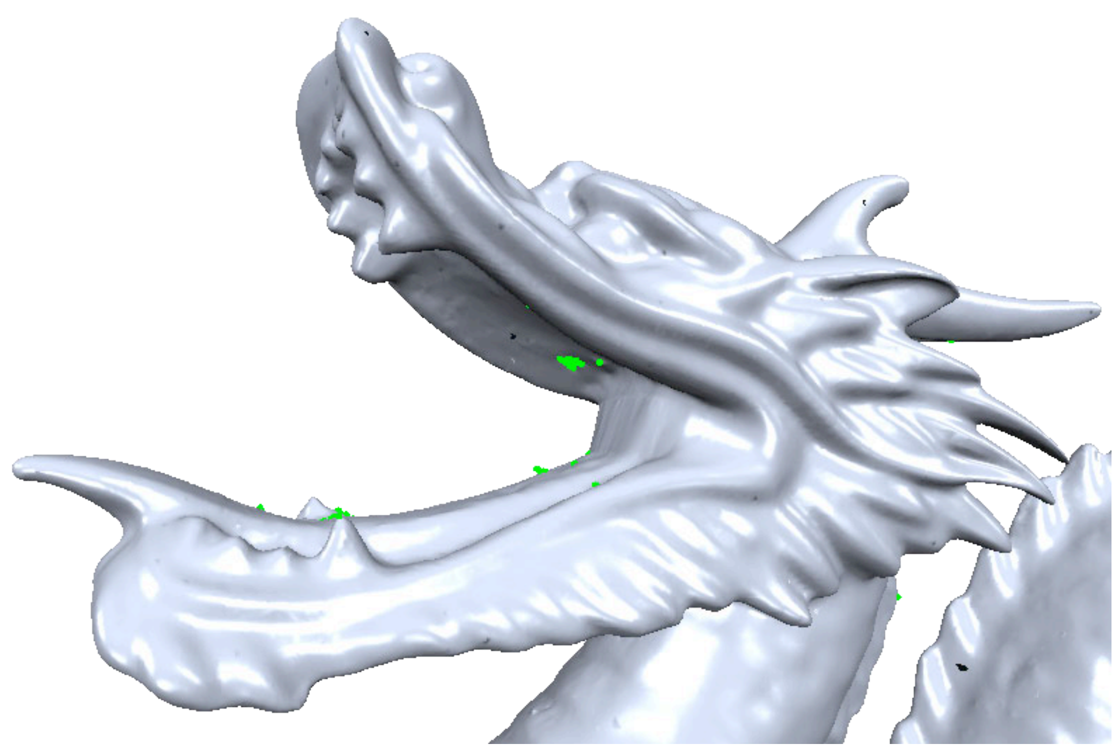


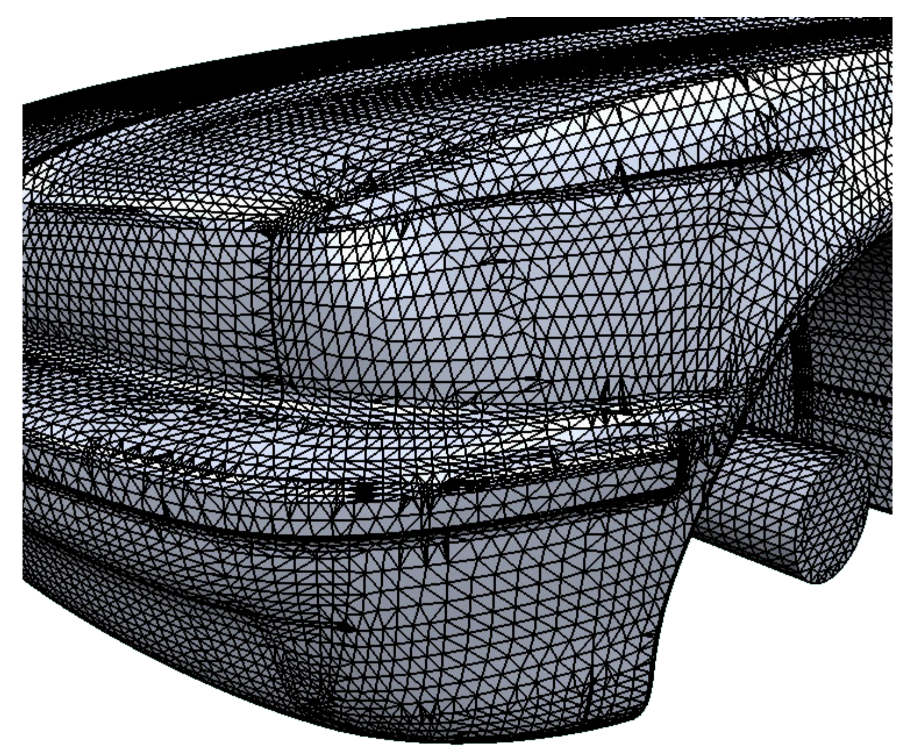
Overview

- Surface-Based Deformation
- **Space Deformation**
- Multiresolution Deformation
- Differential Coordinates
- Comparison

Surface-Based Deformation

- Problems with
  - Highly complex models
  - Topological inconsistencies
  - Geometric degeneracies





Surface-Based Deformation

1. Control: Prescribe *arbitrary* constraints:

$$\mathbf{d}\left(\mathbf{p}_i\right) = \mathbf{d}_i\,, \quad \forall \mathbf{p}_i \in \mathcal{C}$$

2. Fitting: Smoothly interpolate constraints by a displacement function:

$$\mathbf{d} : \mathcal{S} \rightarrow \mathbb{R}^3 \quad \text{with} \quad \mathbf{d}\left(\mathbf{p}_i\right) = \mathbf{d}_i$$

3. Evaluation: Displace all points:

$$\mathbf{p}_i \mapsto \mathbf{p}_i + \mathbf{d}\left(\mathbf{p}_i\right) \quad \forall \mathbf{p}_i \in \mathcal{S}$$

Space Deformation

1. Control: Prescribe *arbitrary* constraints:

$$\mathbf{d}(\mathbf{p}_i) = \mathbf{d}_i\,, \quad \forall \mathbf{p}_i \in \mathcal{C}$$

2. Fitting: Smoothly interpolate constraints by a *trivariate space deformation* function:

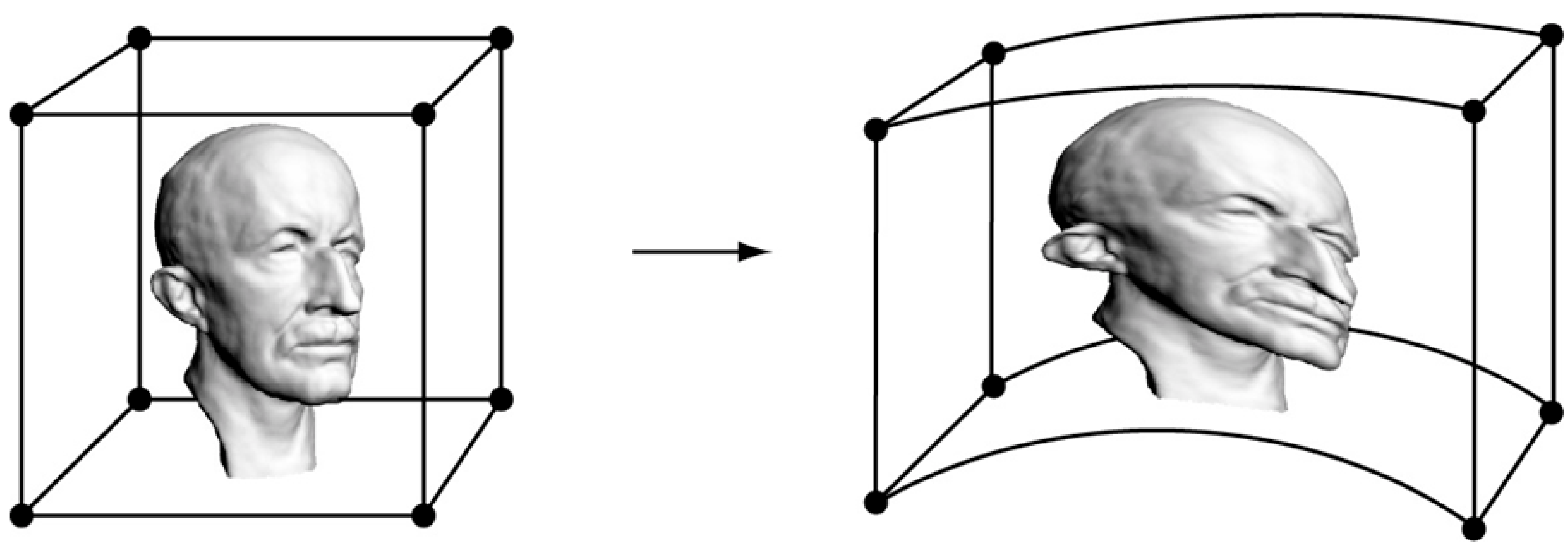
$$\mathbf{d} : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \quad \text{with} \quad \mathbf{d}(\mathbf{p}_i) = \mathbf{d}_i$$

3. Evaluation: Displace all points:

$$\mathbf{p}_i \mapsto \mathbf{p}_i + \mathbf{d}(\mathbf{p}_i) \quad \forall \mathbf{p}_i \in \mathcal{S}$$

Freeform Deformation

- Deform object's bounding box
  - Implicitly deforms embedded objects



Freeform Deformation

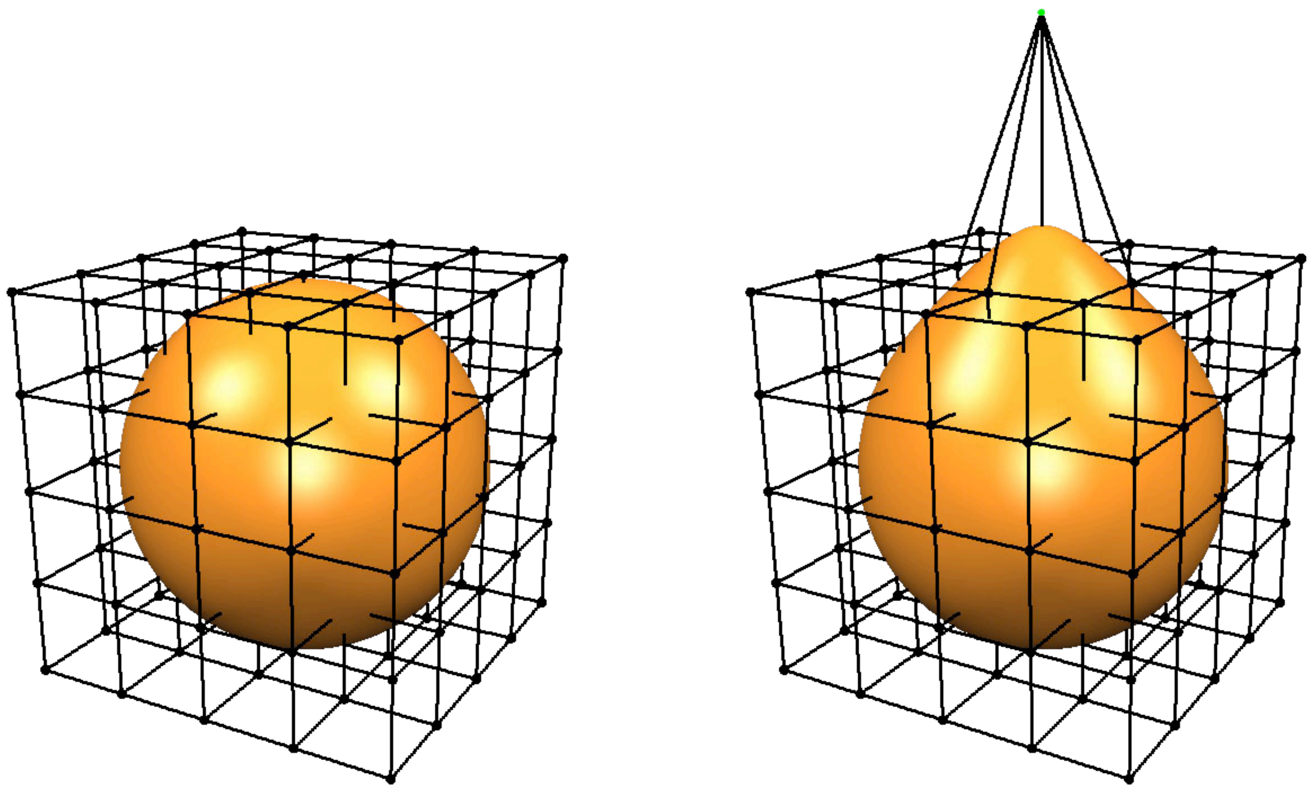
- Deform object's bounding box
  - Implicitly deforms embedded objects

- Tri-variate tensor-product spline

$$\mathbf{d}\left(u, v, w\right) = \sum_{i=0}^{l} \sum_{j=0}^{m} \sum_{k=0}^{n} \mathbf{c}_{ijk} N_i^l(u)\, N_j^m(v)\, N_k^n(w)$$

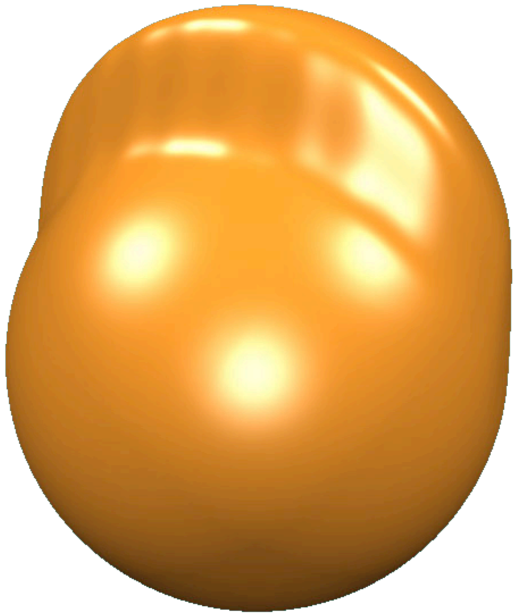
Freeform Deformation

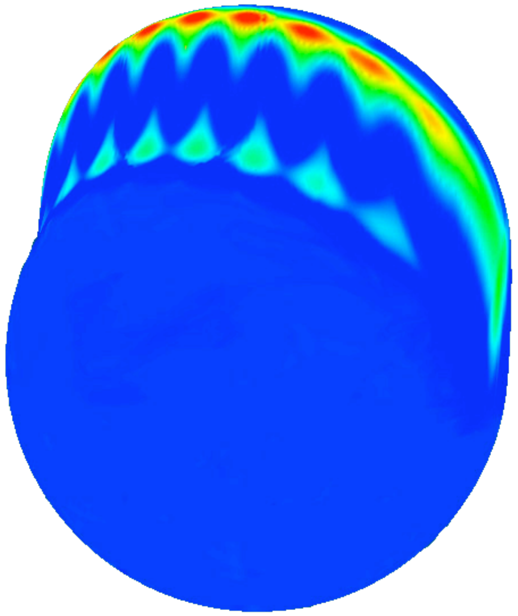
- Deform object's bounding box
  - Implicitly deforms embedded objects
- Tri-variate tensor-product spline



Freeform Deformation

- Deform object's bounding box
  - Implicitly deforms embedded objects
- Tri-variate tensor-product spline
  - Aliasing artifacts
- Interpolate deformation constraints?
  - Only in least squares sense





Radial Basis Functions

- Represent deformation by RBFs

$$\mathbf{d}(\mathbf{x}) = \sum_j \mathbf{w}_j \cdot \varphi\left(\|\mathbf{c}_j - \mathbf{x}\|\right) + \mathbf{p}(\mathbf{x})$$

- Well suited for scattered data interpolation
  – Smooth interpolation
  – Irregularly placed constraints

Which basis function?

- Triharmonic RBF $\varphi(r) = r^3$
  - High fairness, minimizes

$$\int_{\mathbb{R}^3} \|\mathbf{d}_{uuu}\|^2 + \|\mathbf{d}_{vuu}\|^2 + \ldots + \|\mathbf{d}_{www}\|^2 \; dudvdw$$

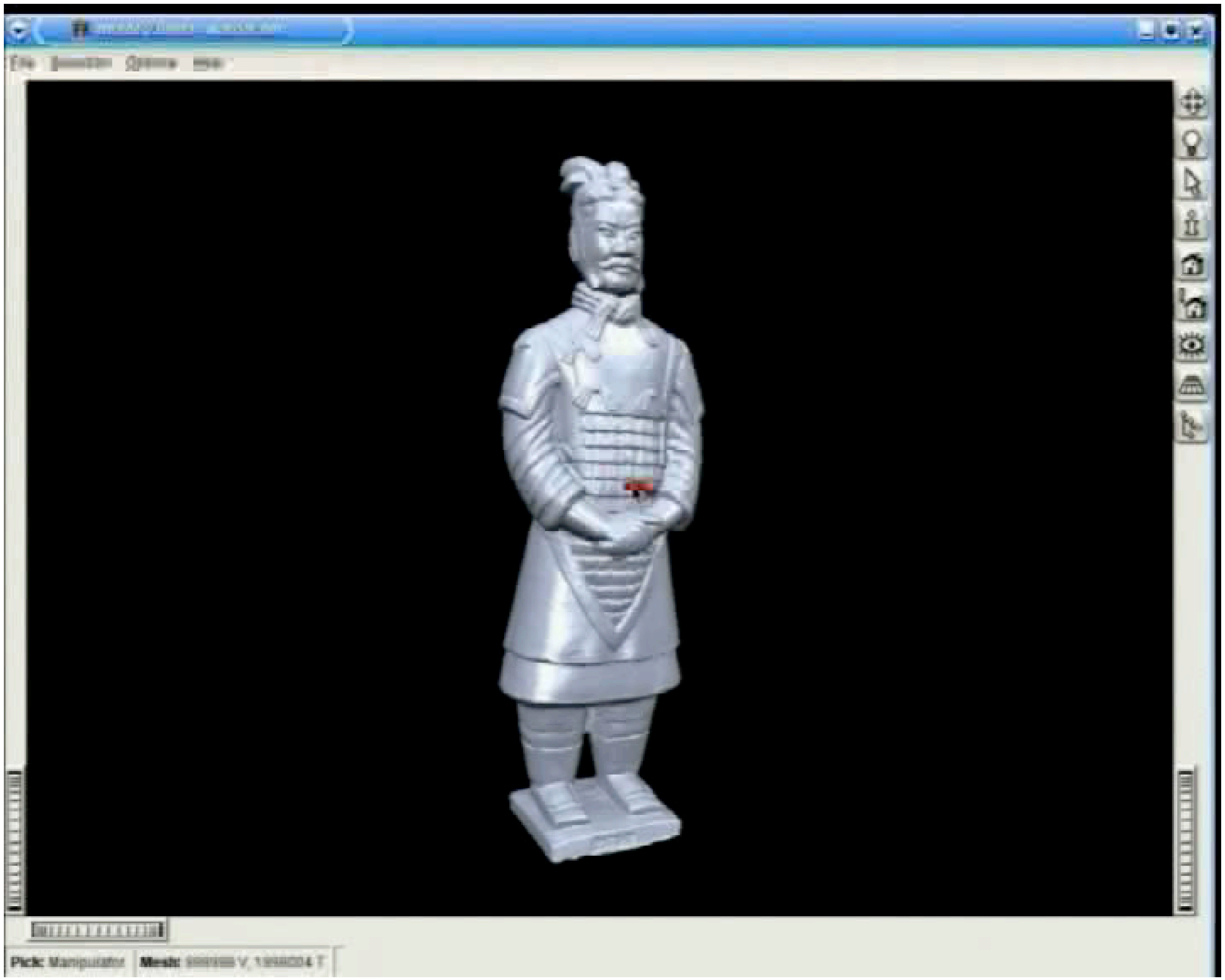
  - C^2 boundary constraints
  - Global support

RBF Deformation [Botsch05]

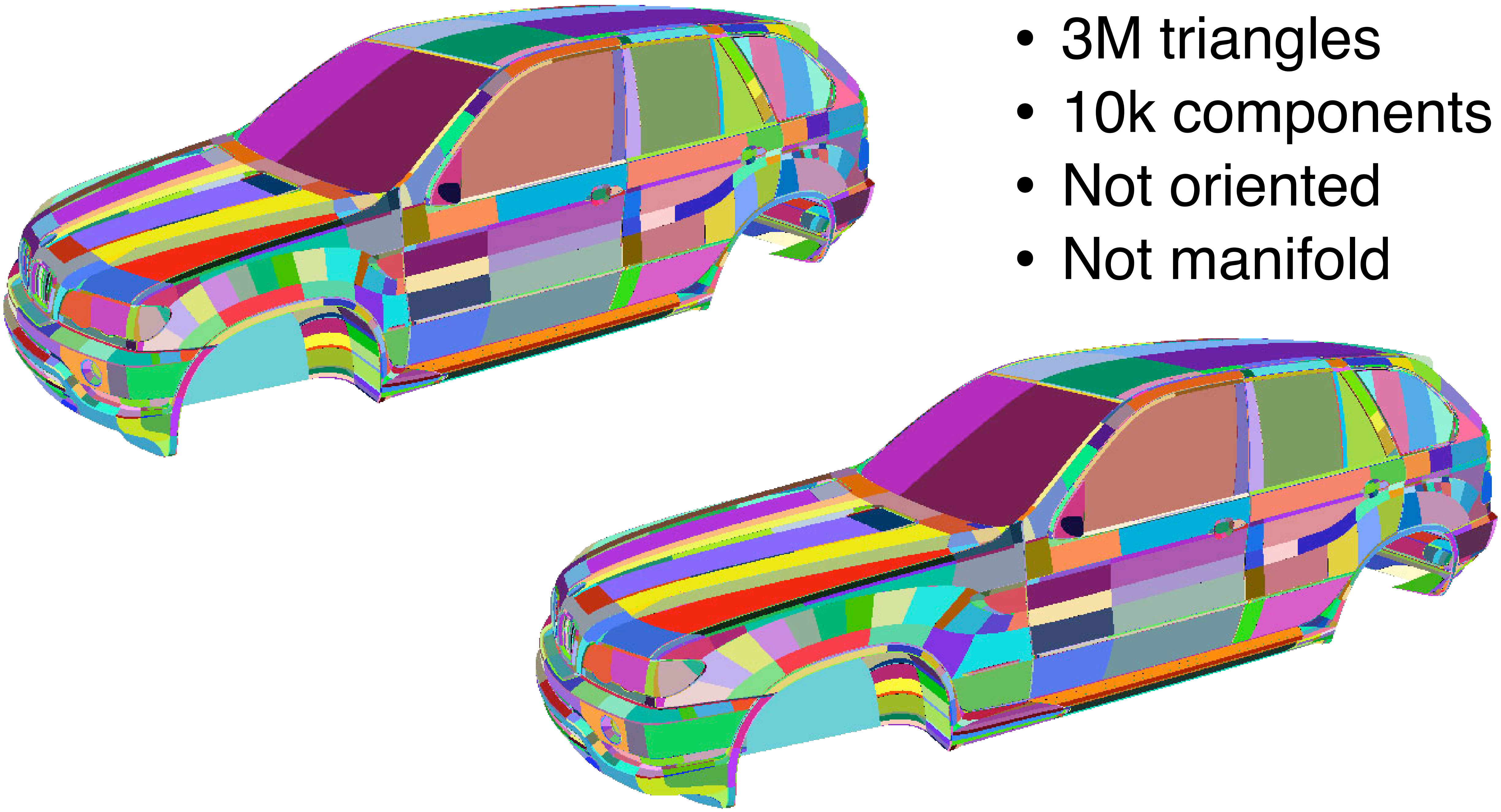
- Fitting
  - Place centers $\mathbf{c}_i$ on constraint points $\mathbf{p}_i$
  - Leads to dense linear system in $\mathbf{w}_i$
  - Incremental least squares solver

- Evaluation
  - Function deforms points $\mathbf{p}_i \mapsto \mathbf{p}_i + \mathbf{d}(\mathbf{p}_i)$
  - Jacobian deforms normals $\mathbf{n}_i \mapsto (\mathbf{I} + \nabla\mathbf{d})^{-T}\,\mathbf{n}_i$
  - Basis function matrices $\mathbf{B},\ \mathbf{B}_x,\ \mathbf{B}_y,\ \mathbf{B}_z$
  - Evalute deformation on graphics card (30M v/s)

Statue: 1M vertices



“Bad Meshes”

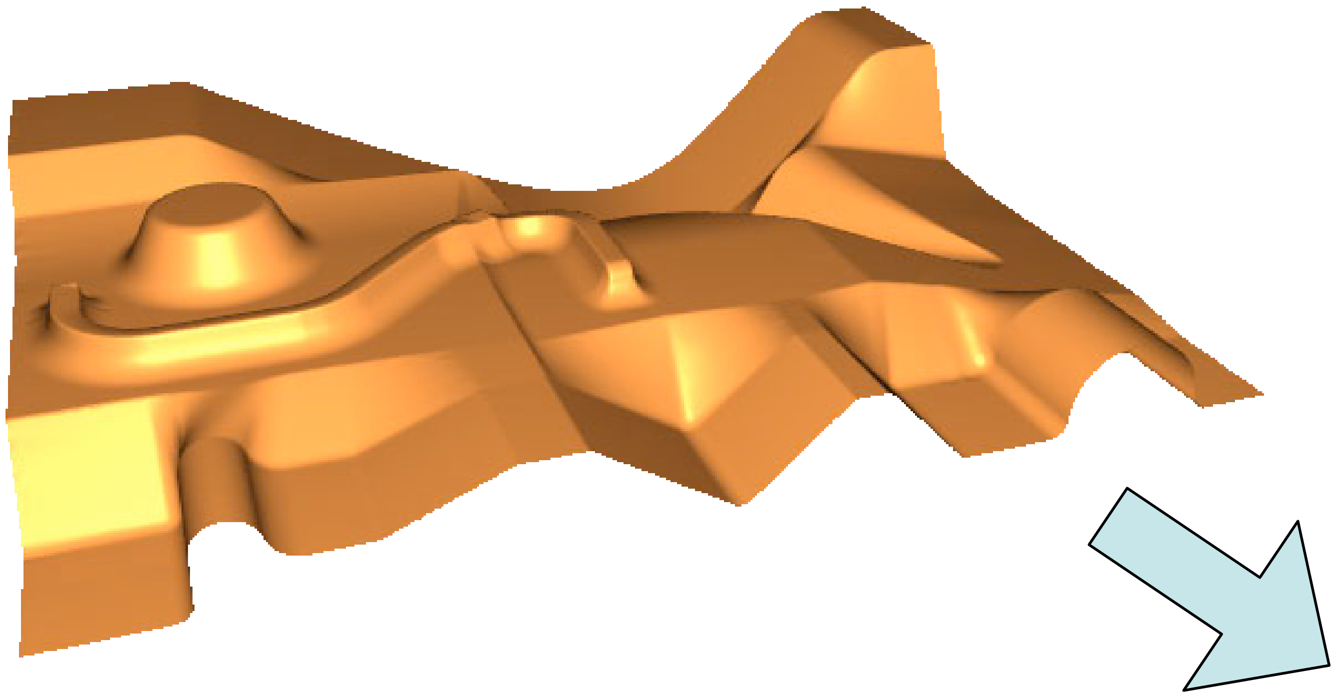


- 3M triangles
- 10k components
- Not oriented
- Not manifold

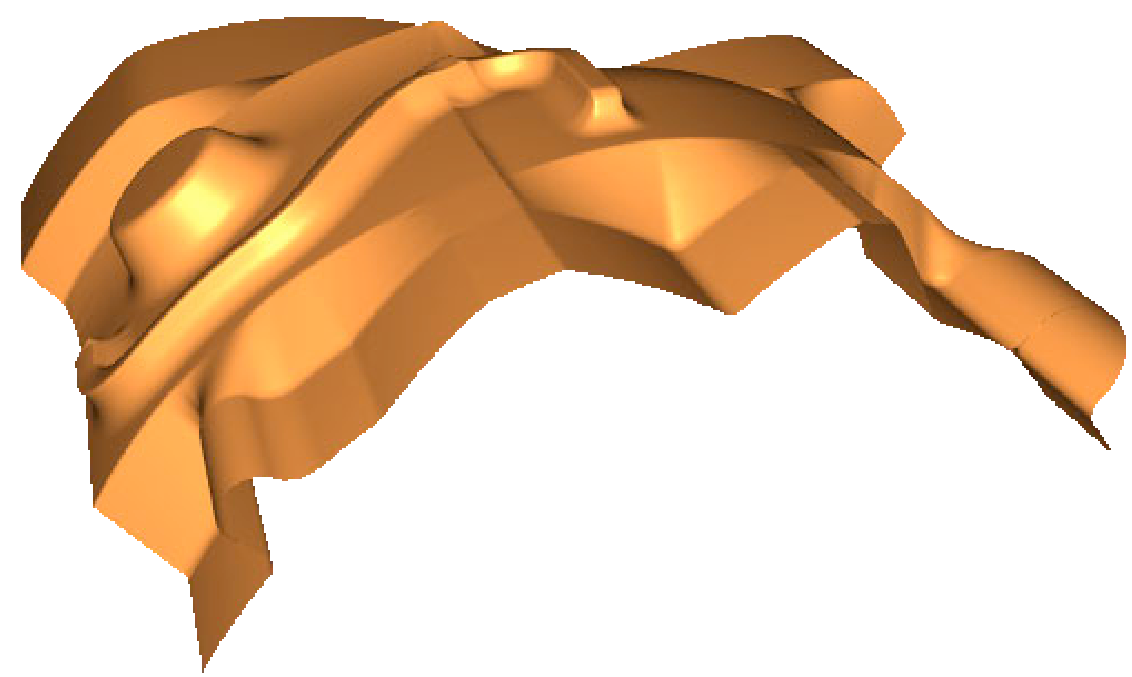
Overview

- Surface-Based Deformation
- Space Deformation
- **Multiresolution Deformation**
- Differential Coordinates
- Comparison

Multiresolution Editing

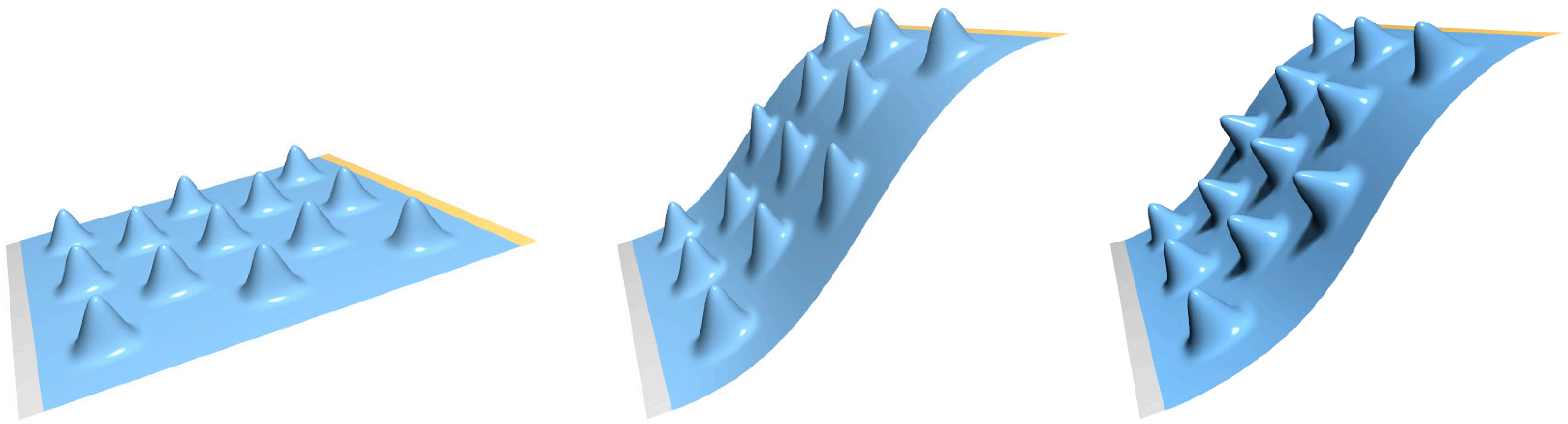


Shape deformation
with intuitive
detail preservation



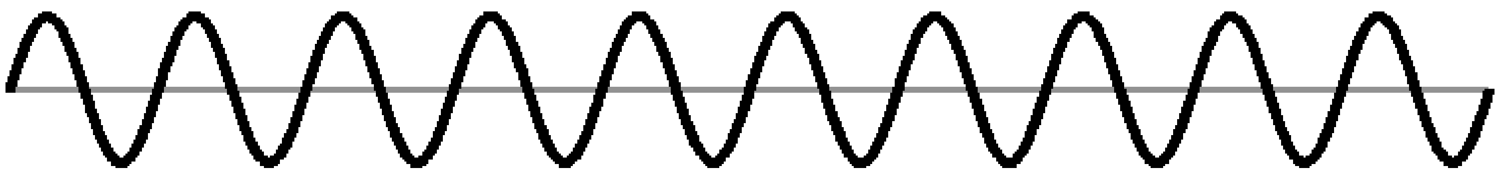
Multiresolution Modeling

- Even pure translations induce local rotations!
  - Inherently non-linear coupling
- Or: linear model + multi-scale decomposition...



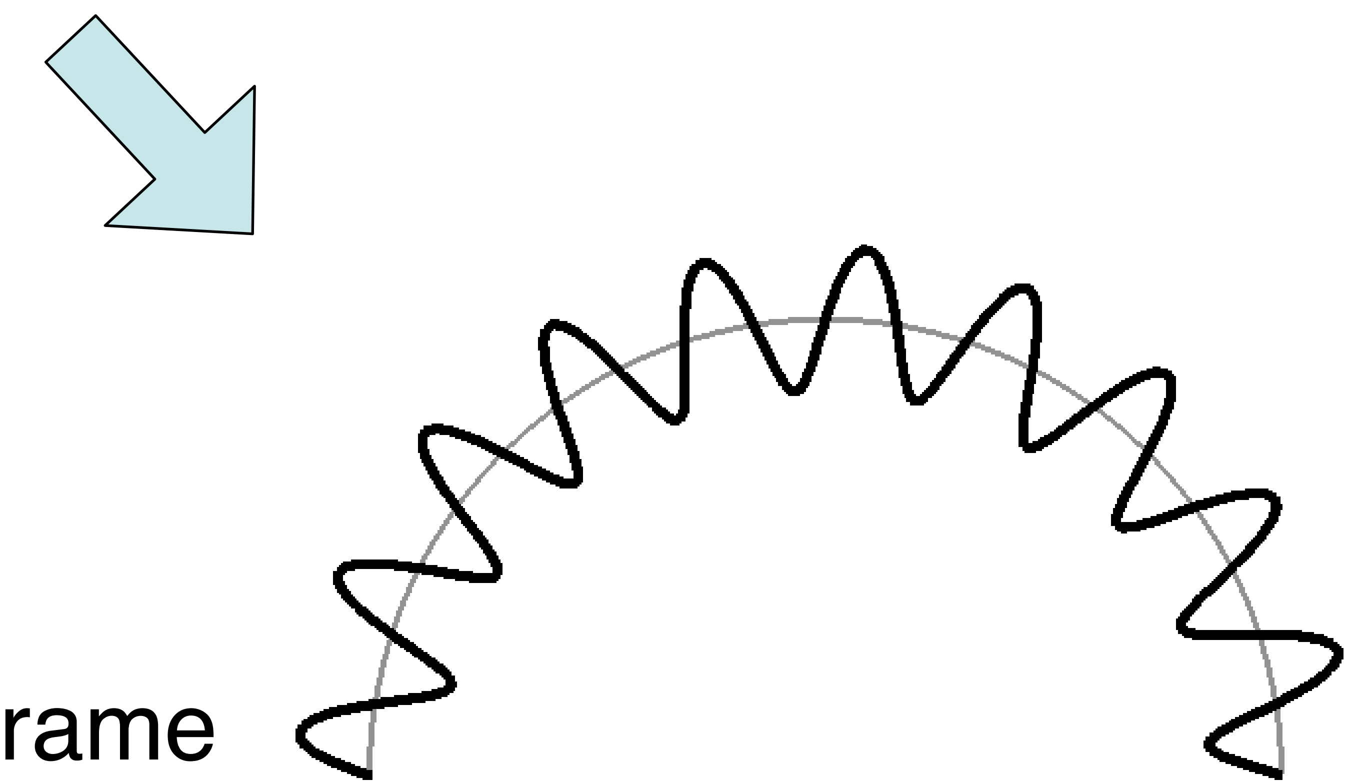
Original | Linear deform. | Non-linear deform.

Multiresolution Editing



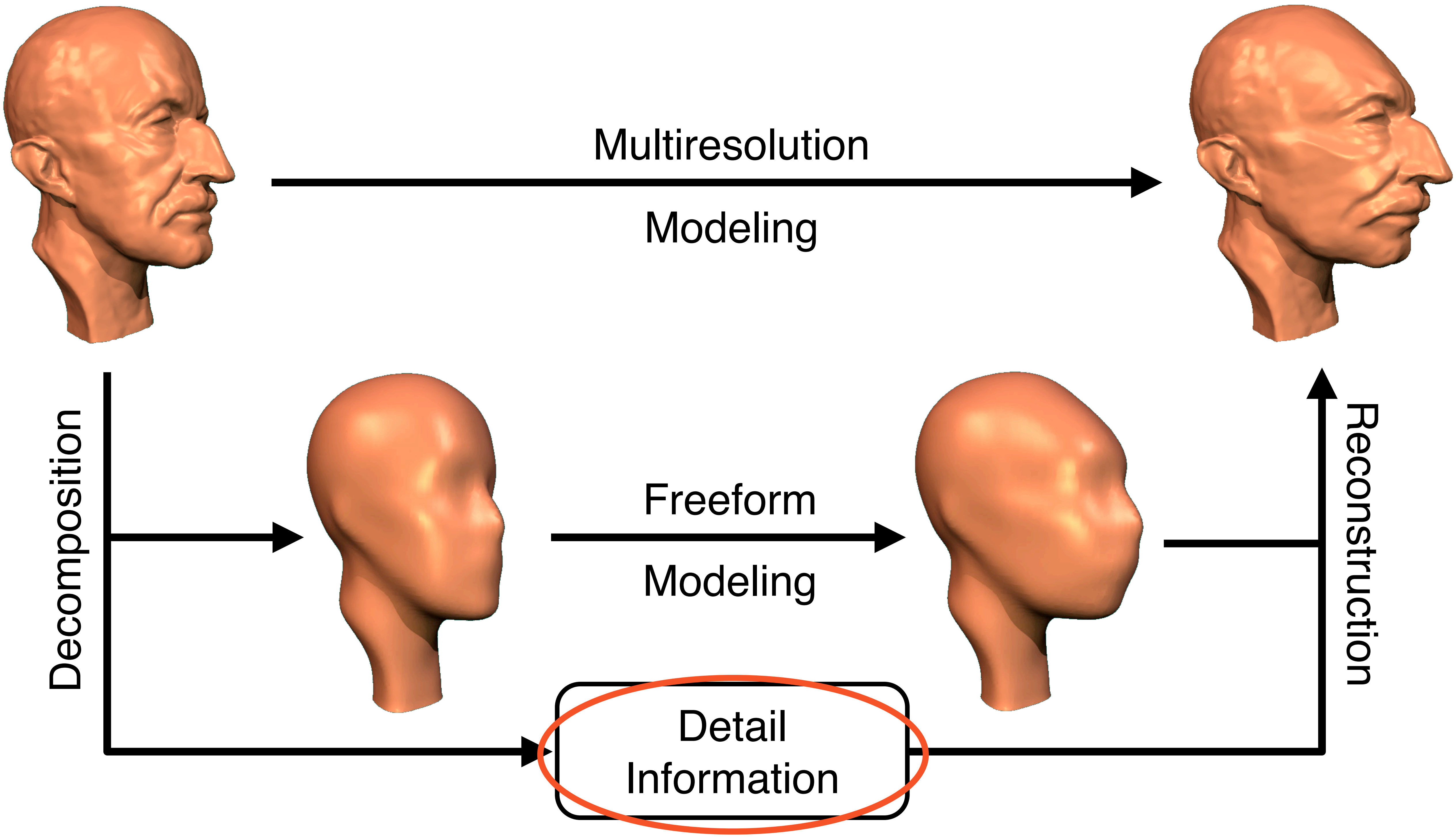
Frequency decomposition

Change low frequencies

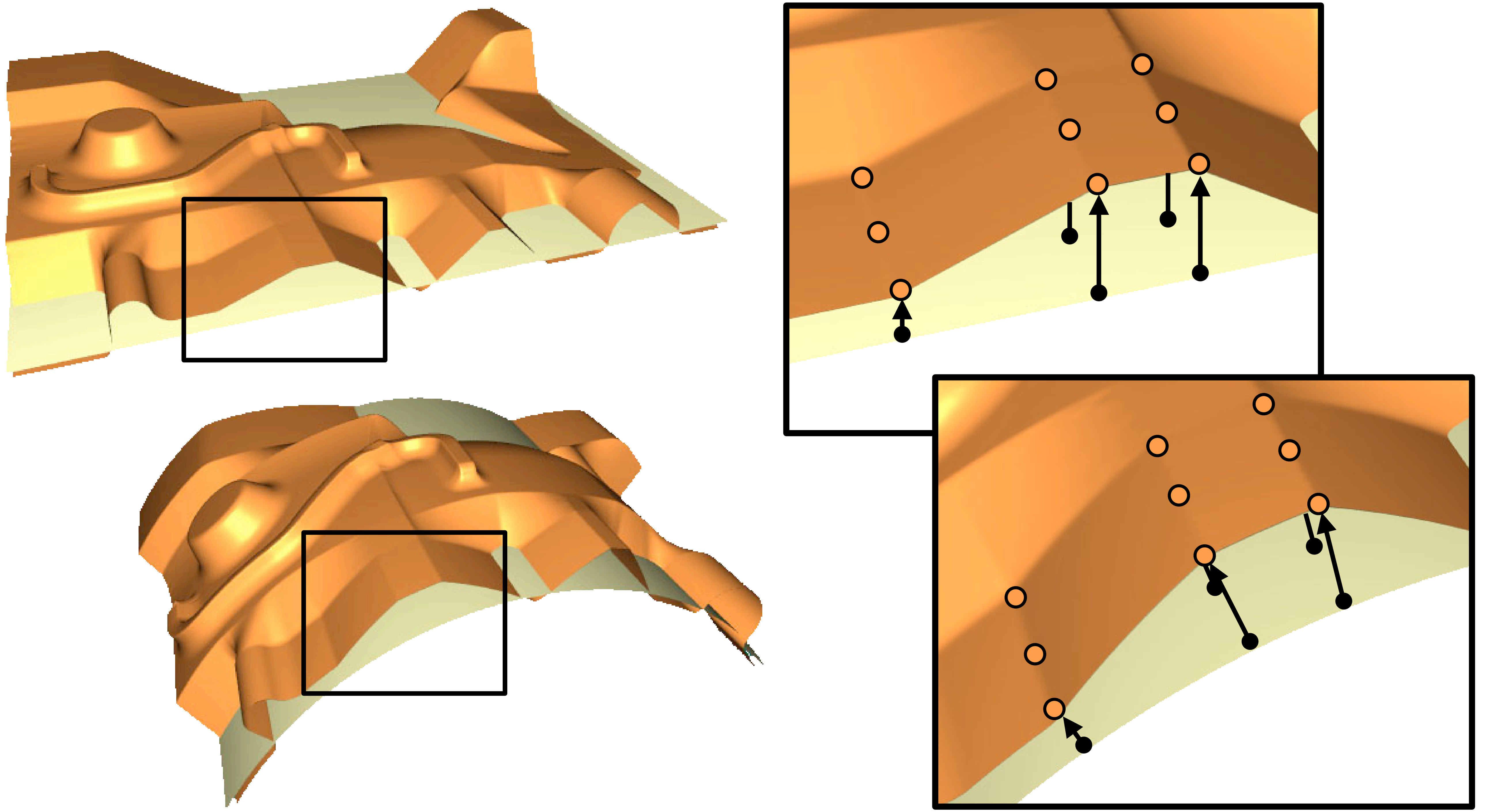


Local frame details

Multiresolution Editing

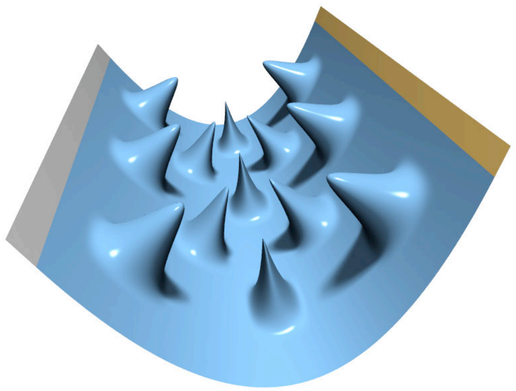


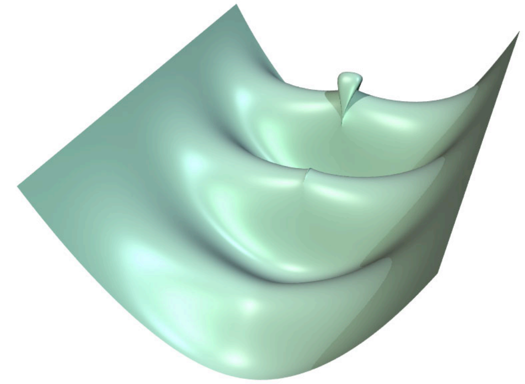
Normal Displacements



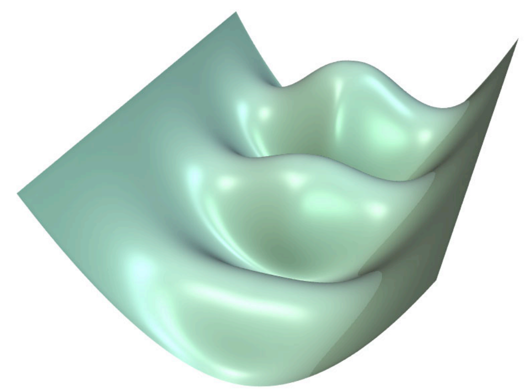
Detail Representations

- Displacement vectors
  - very efficient
  - local self-intersections

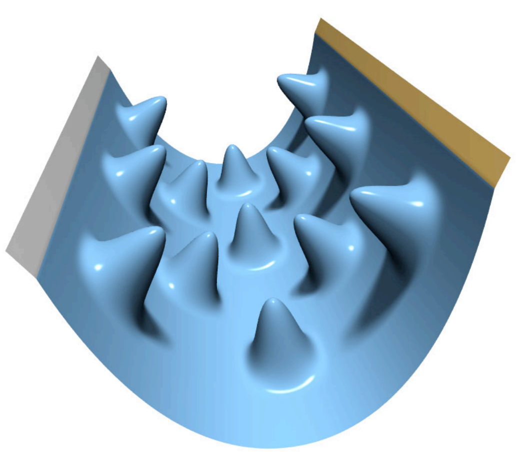




- Displacement volumes
  - avoid self-intersections
  - non-linear method



- Deformation transfer
  - [Botsch et al, VMV 06]
  - inbetween...



Overview

- Surface-Based Deformation
- Space Deformation
- Multiresolution Deformation
- **Differential Coordinates**
- Comparison

Differential Coordinates

- Avoid multiresolution hierarchy because
  - It is difficult for geom. / topol. complex models
  - Might require multiple hierarchy levels

- Change differential instead of spatial coordinates
  - Gradients, Laplacians
  - Find mesh w/ desired differential coordinates

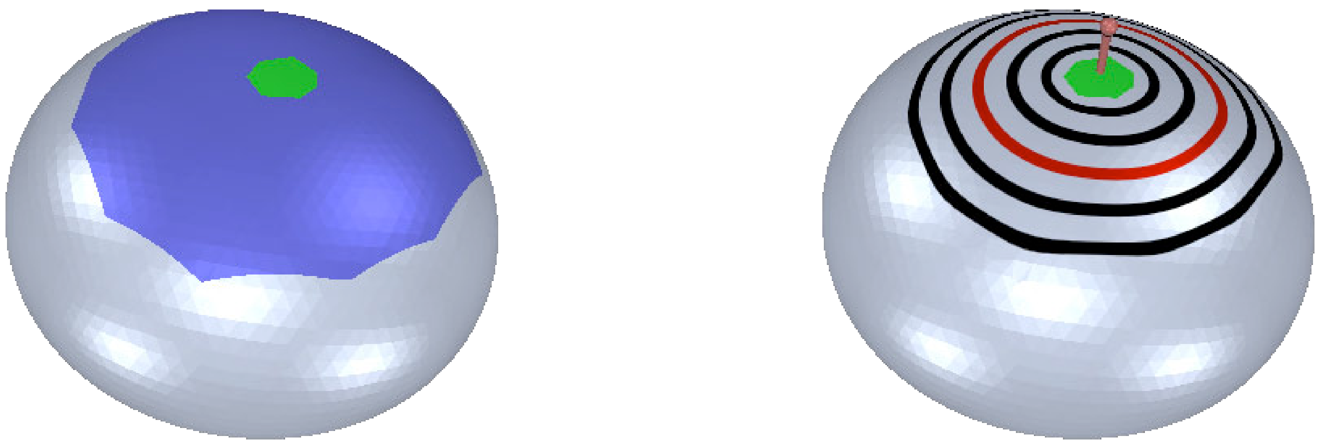
Gradient-Based Editing

- Gradient of coordinate function **p**
  - Constant per triangle $\nabla \mathbf{p}\big|_{f_j} =: \mathbf{G}_j \in \mathbb{R}^{3\times 3}$

$$\begin{pmatrix} \mathbf{G}_1 \\ \vdots \\ \mathbf{G}_F \end{pmatrix} = \underbrace{\mathbf{G}}_{\in \mathbb{R}^{3F\times V}} \cdot \begin{pmatrix} \mathbf{p}_1^T \\ \vdots \\ \mathbf{p}_V^T \end{pmatrix}$$

Gradient-Based Editing

- Manipulate per-face gradients $\mathbf{G}_j \mapsto \mathbf{G}'_j$
  - Gradient of handle deformation
  - Rotation and scale/shear components
  - Distance-based propagation

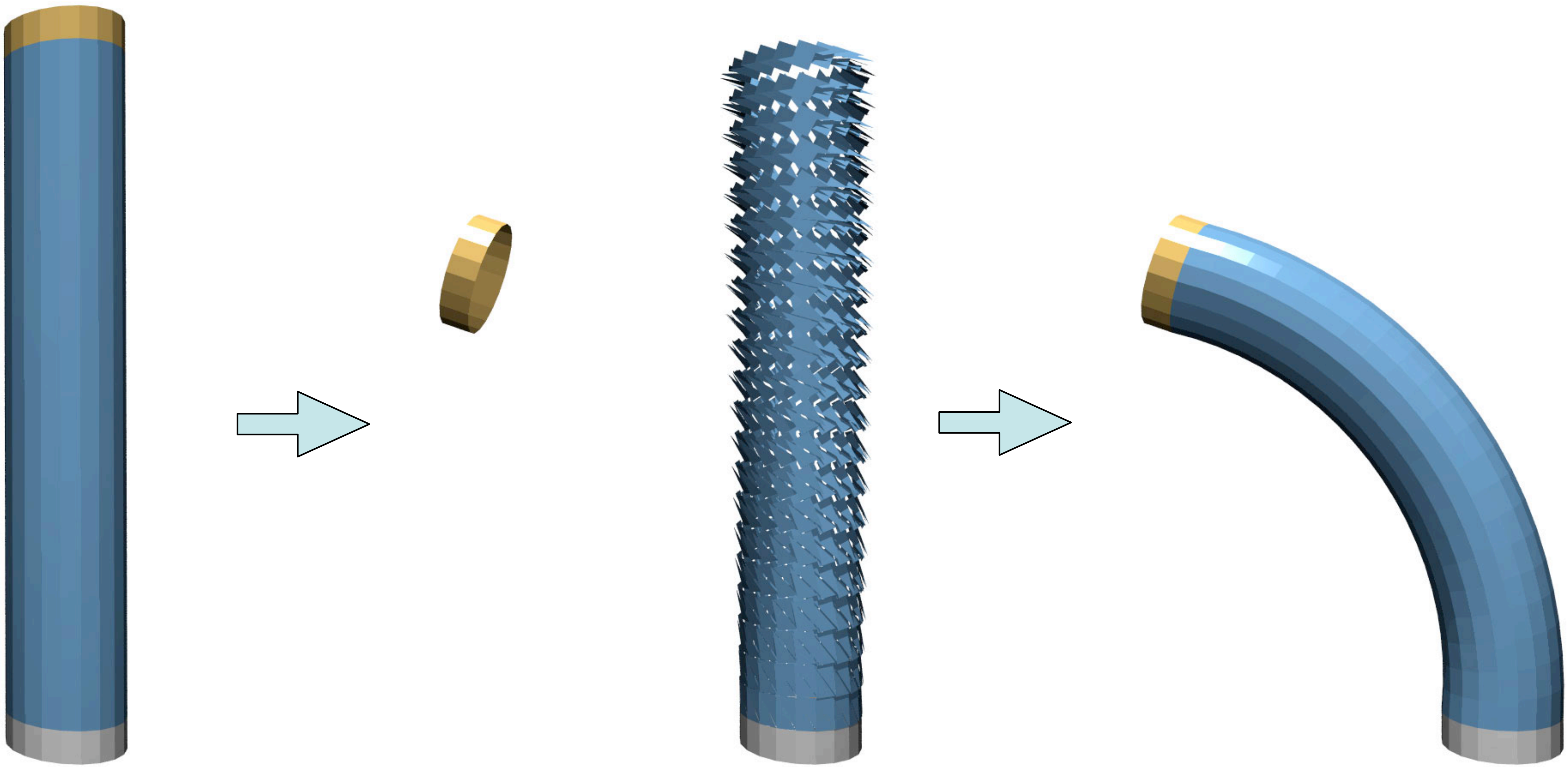


Gradient-Based Editing

- Reconstruct mesh from gradients
  - Overdetermined problem $\mathbf{G} \in \mathbb{R}^{3F \times V}$
  - Weighted least squares system
  - Linear Poisson system

$$\underbrace{\boxed{\mathbf{G}^T\mathbf{D}\mathbf{G}}}_{\mathrm{div}\nabla = \Delta} \cdot \begin{pmatrix} \mathbf{p}_1'^T \\ \vdots \\ \mathbf{p}_V'^T \end{pmatrix} = \underbrace{\boxed{\mathbf{G}^T\mathbf{D}}}_{\mathrm{div}} \cdot \begin{pmatrix} \mathbf{G}_1' \\ \vdots \\ \mathbf{G}_F' \end{pmatrix}$$

Gradient-Based Editing



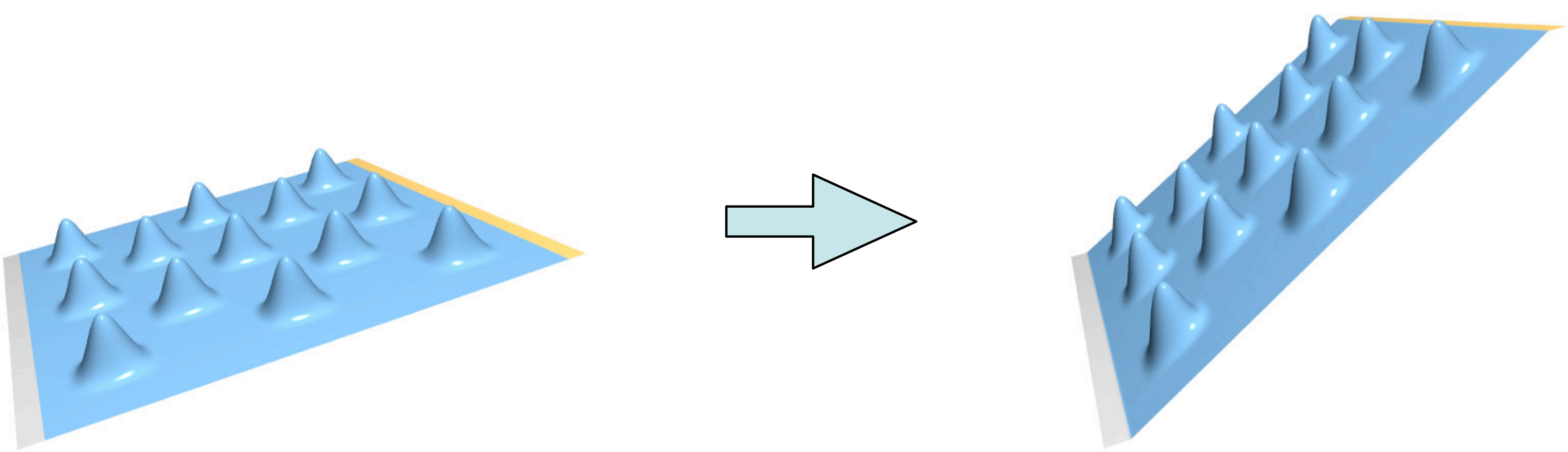
Original

Rotated Gradients

Reconstructed Mesh

Limitations

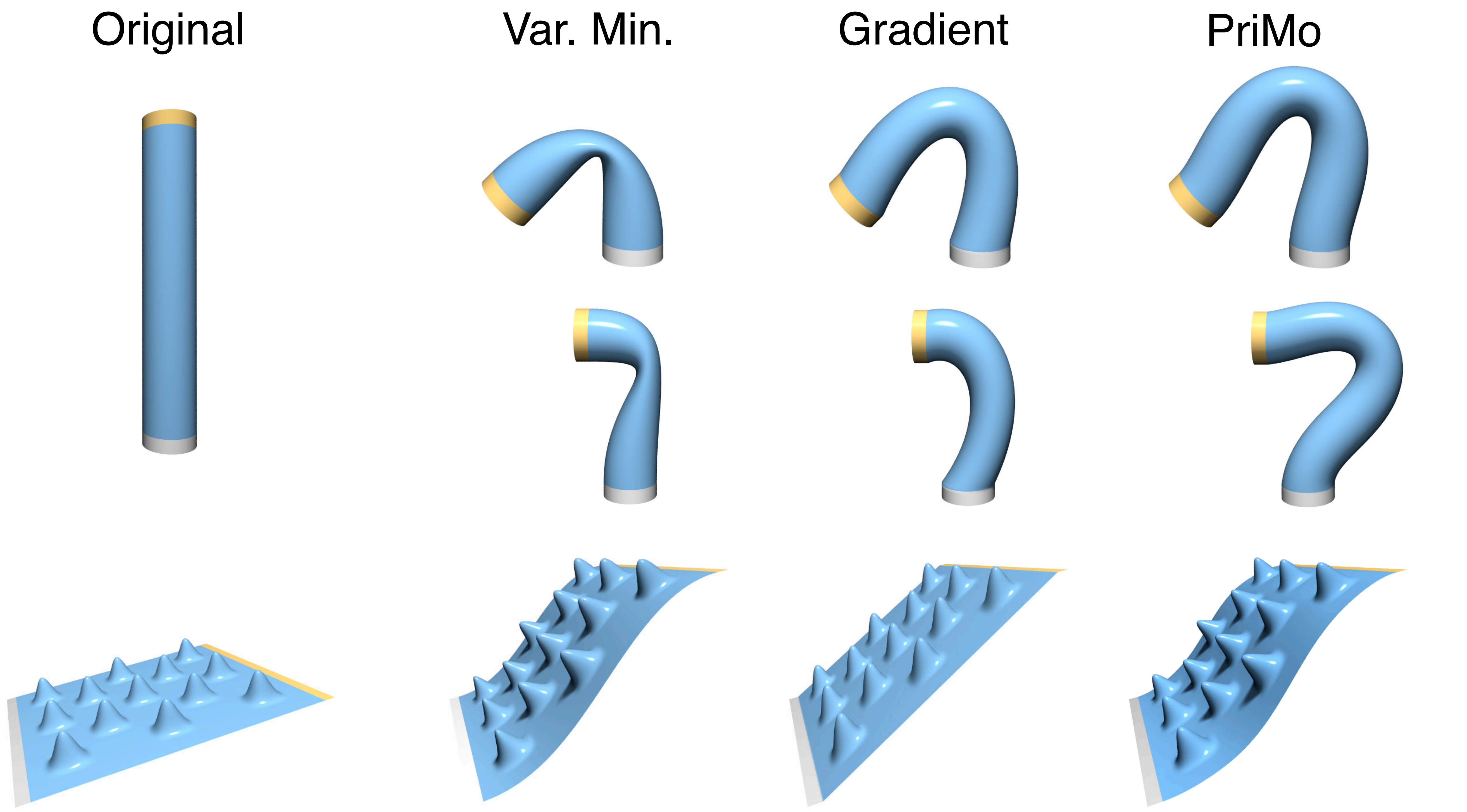
- Differential coordinates work well for rotations
  - Represented by deformation gradient
- Translations don’t change deformation gradient
  - Translations don’t change surface gradient
  - *“Translation insensitivity”*



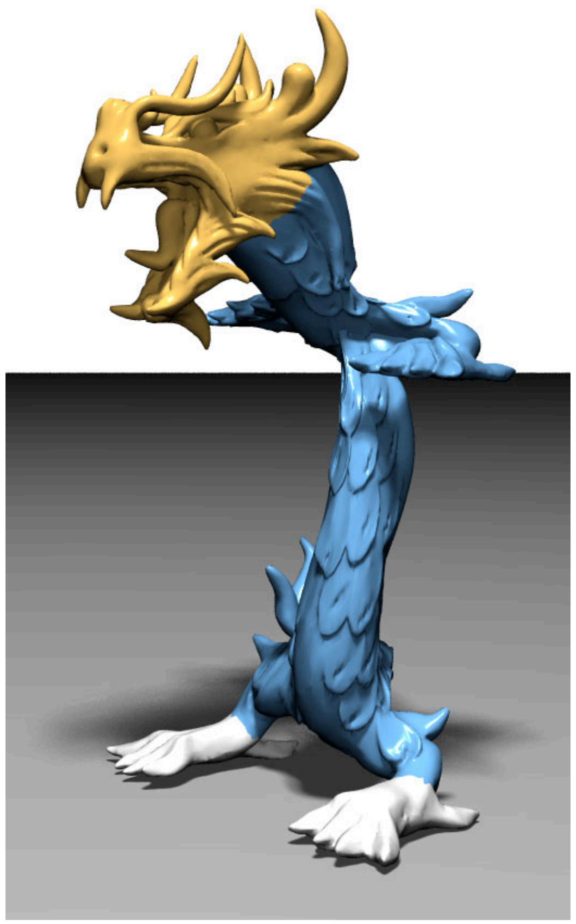
Overview

- Surface-Based Deformation
- Space Deformation
- Multiresolution Deformation
- Differential Coordinates
- **Comparison**

Comparison



Non-Linear Deformation



VarMin

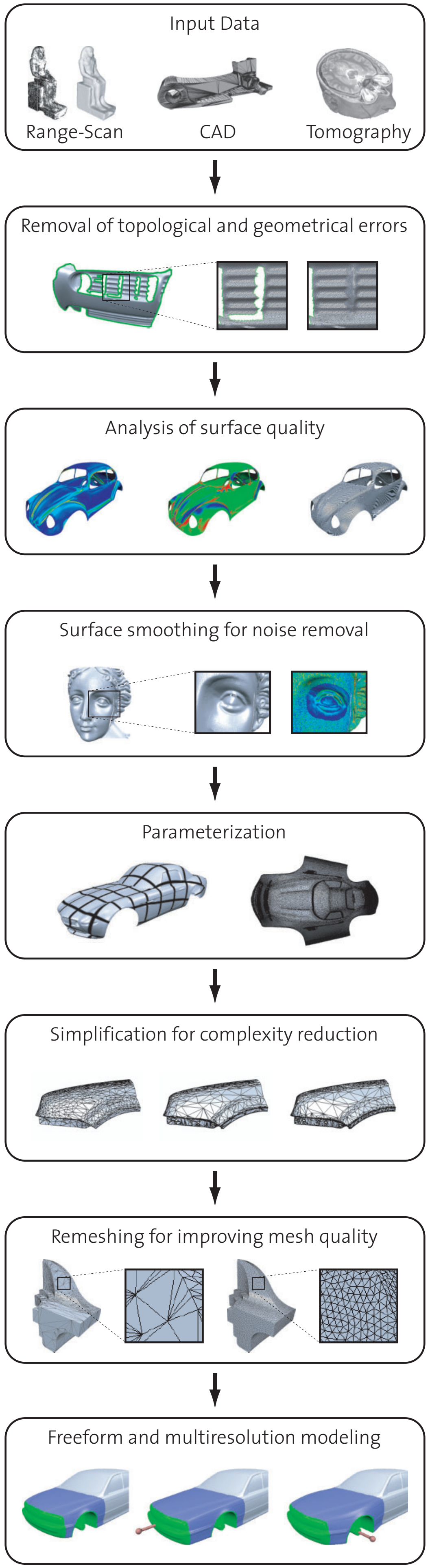
Grad

PriMo

Botsch et al, *“PriMo: Coupled Prisms for Intuitive Surface Modeling”*, SGP 06

Conclusion

- Boundary constraint modeling
  - Smoothness, flexibility, efficiency
  - Need multiresolution framework
- Differential coordinates
  - No multiresolution hierarchy
  - Work well for rotations, problems with translations
- Linear vs. non-linear techniques



Efficient Solvers for (sparse symm. pos. def.) Linear Systems

Mario Botsch

ETH Zurich

Problems in Geometry Processing

- Generic formulation as a PDE
  - Based on partial derivatives
- Discretization for triangle meshes
  - Finite elements / differences
  - Leads to linear systems (typically 10^4 to 10^6 DoFs)
- Partial derivatives are local operators
  - Sparse linear systems

Problems in Geometry Processing

- Most often the PDE can be considered as the Euler-Lagrange equation of an energy minimization problem

- or $\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$ emerges as the normal equation for a least squares problem

➡ Systems are usually symmetric and pos. definite

Problems in Geometry Processing

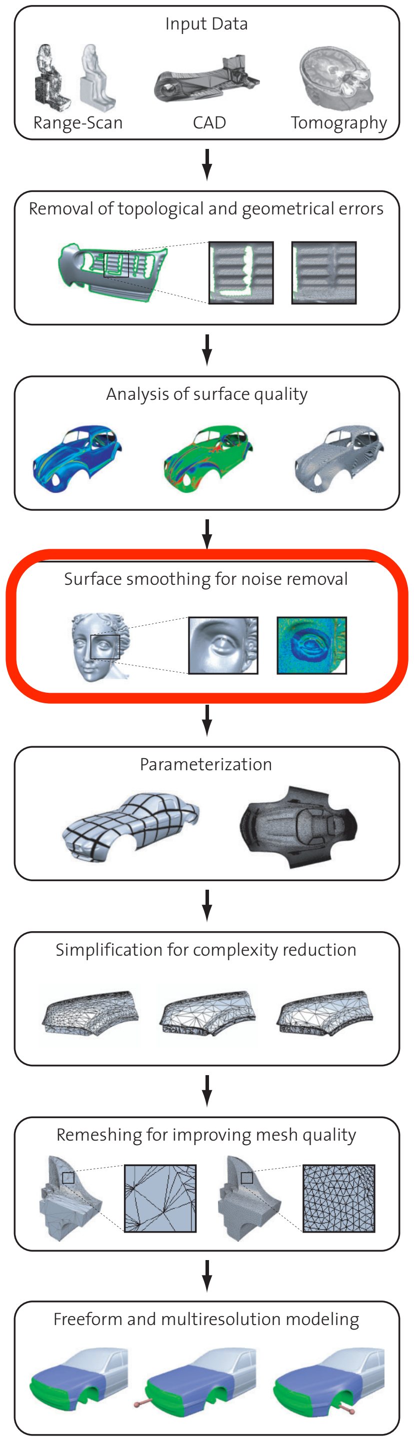
- Linear problems:
  - Solve $\mathbf{Ax} = \mathbf{b}$
- Non-linear problems:
  - Solve sequence of linear systems $\mathbf{A}_k \, \mathbf{x}_k = \mathbf{b}_k$
- Matrix **A** typically is
  - large
  - sparse
  - symmetric positive definite

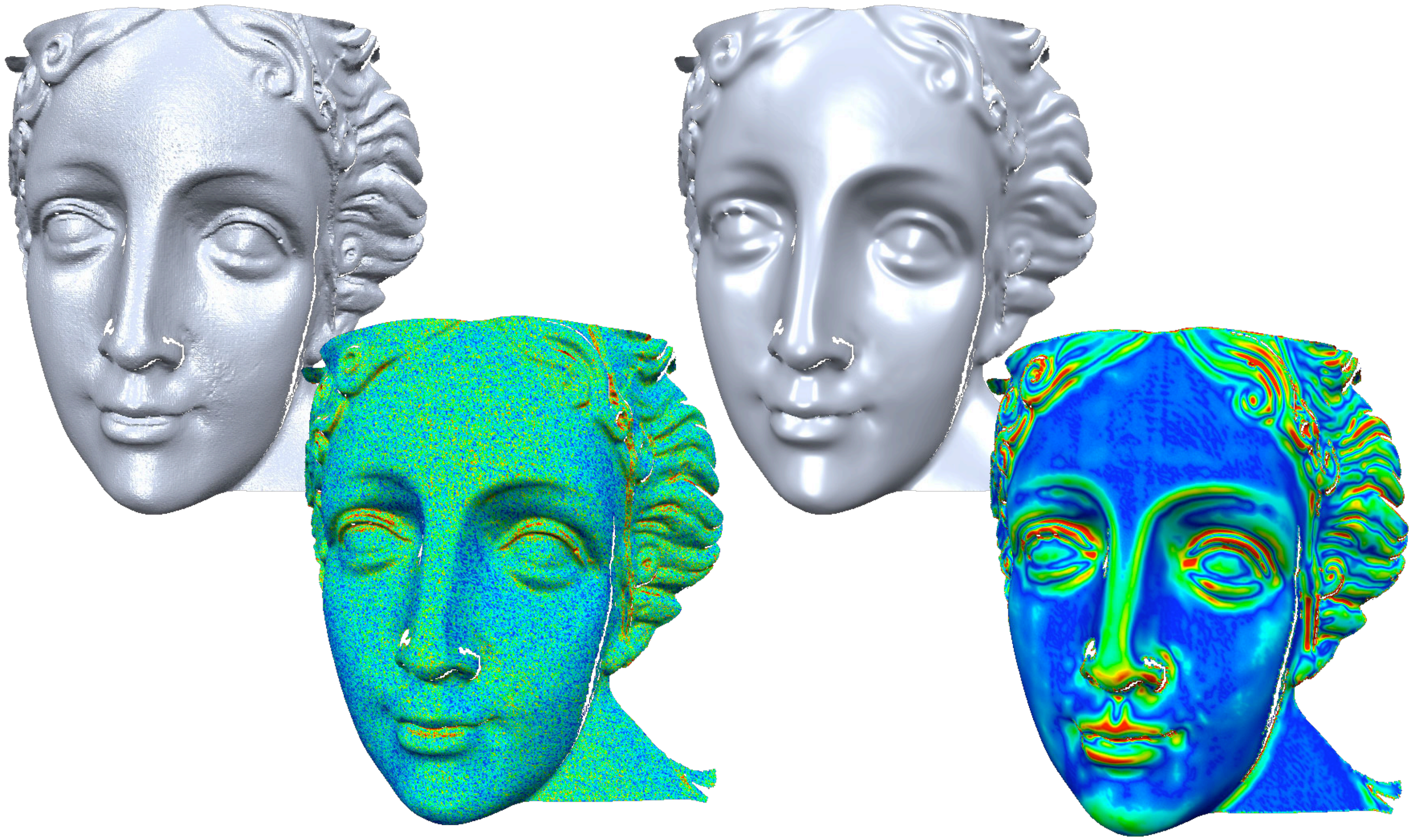
Non-spd systems:
See course notes

Overview

- **Application scenarios**
- Linear system solvers
- Benchmarks

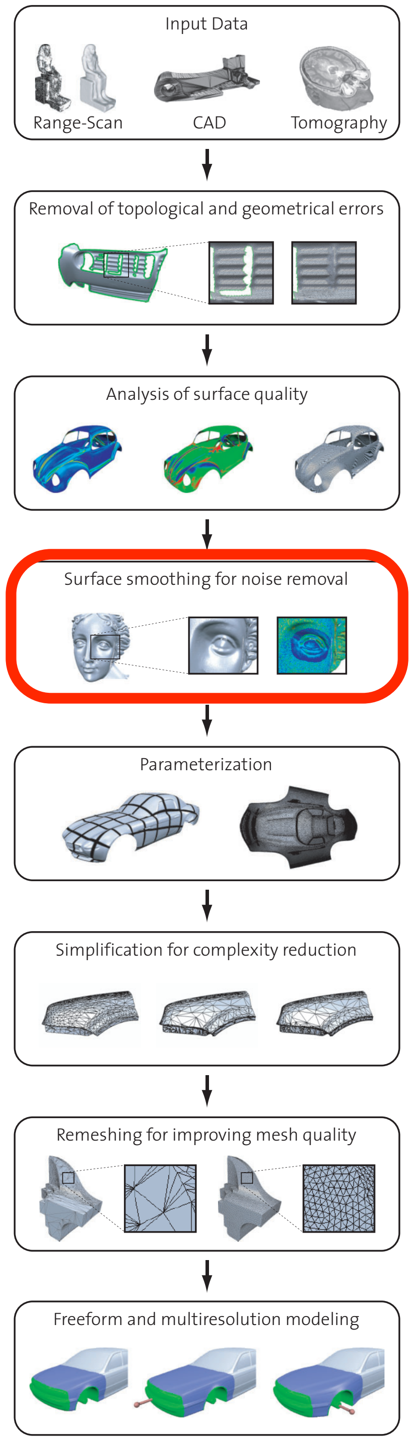
Implicit Fairing

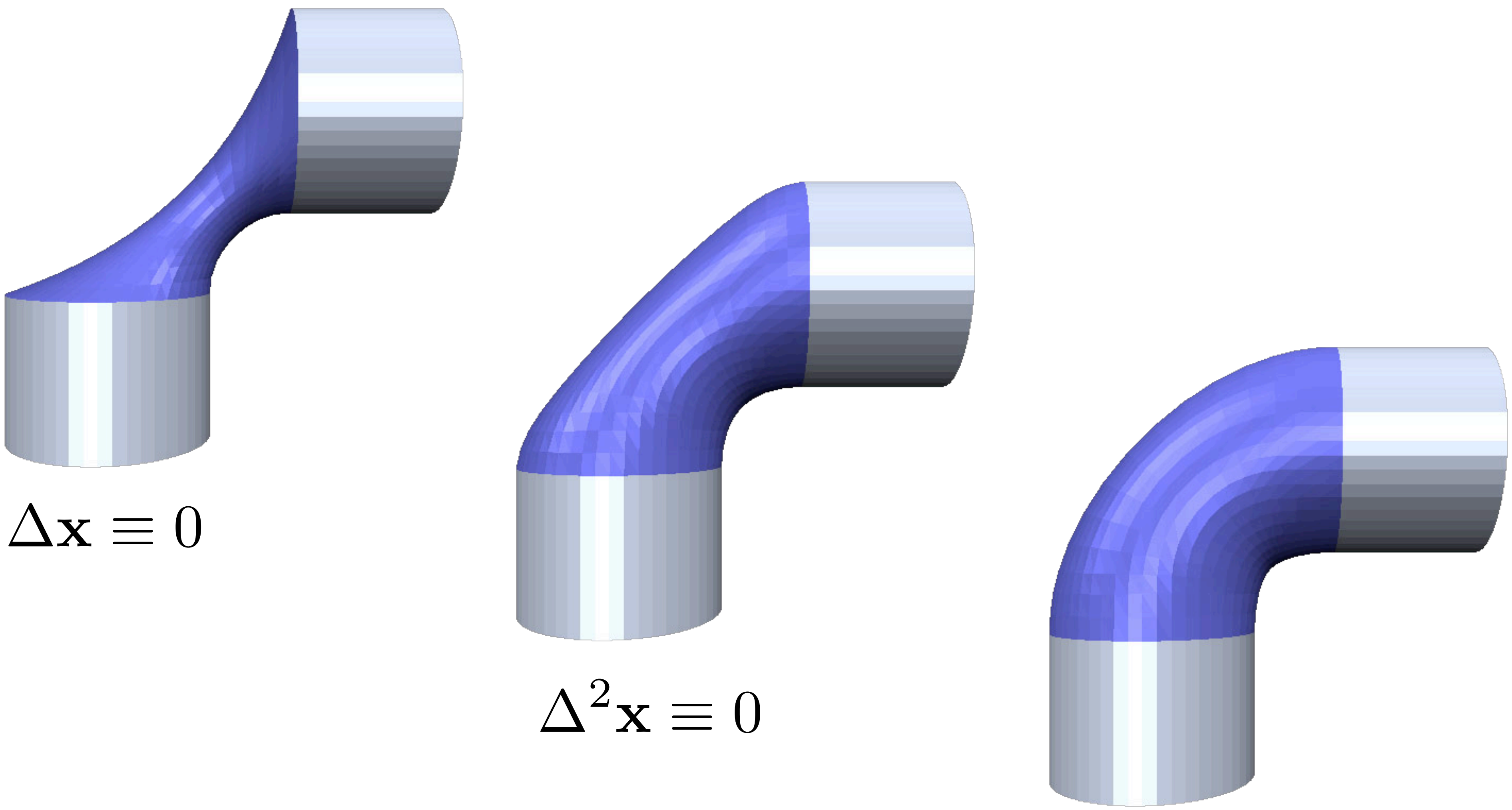




$$\left(\mathbf{I} \pm \Delta_S^k\right) \mathbf{x}_{k+1} = \mathbf{x}_k$$

Variational Energy Minimization



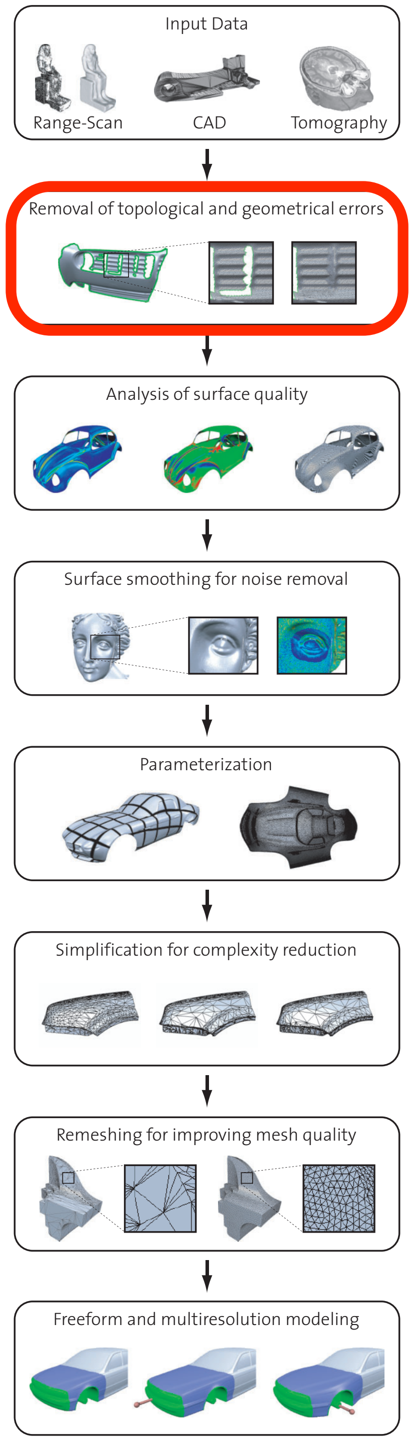


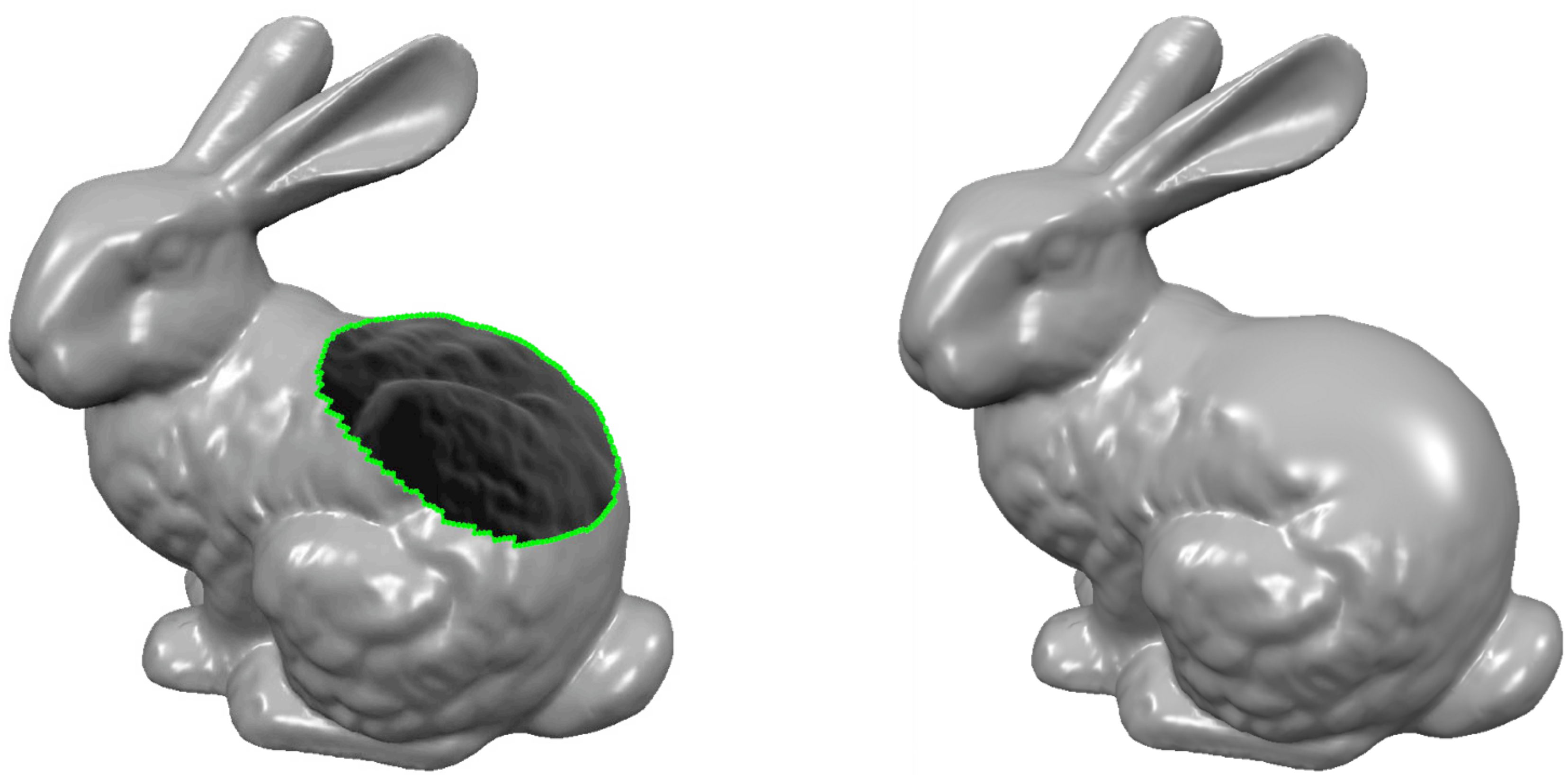
$$\Delta \mathbf{x} \equiv 0$$

$$\Delta^2 \mathbf{x} \equiv 0$$

$$\Delta^3 \mathbf{x} \equiv 0$$

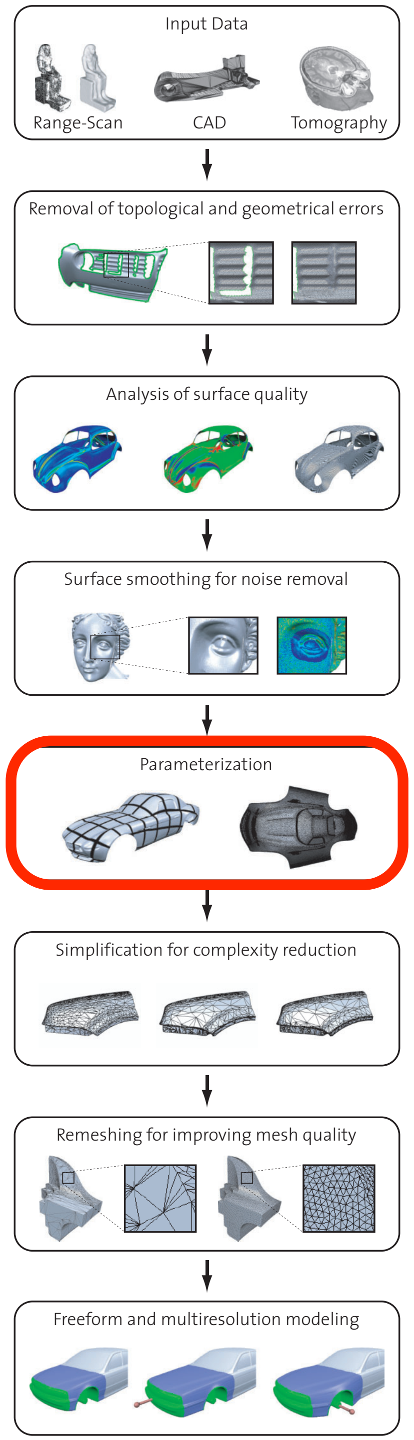
Explicit Hole Filling

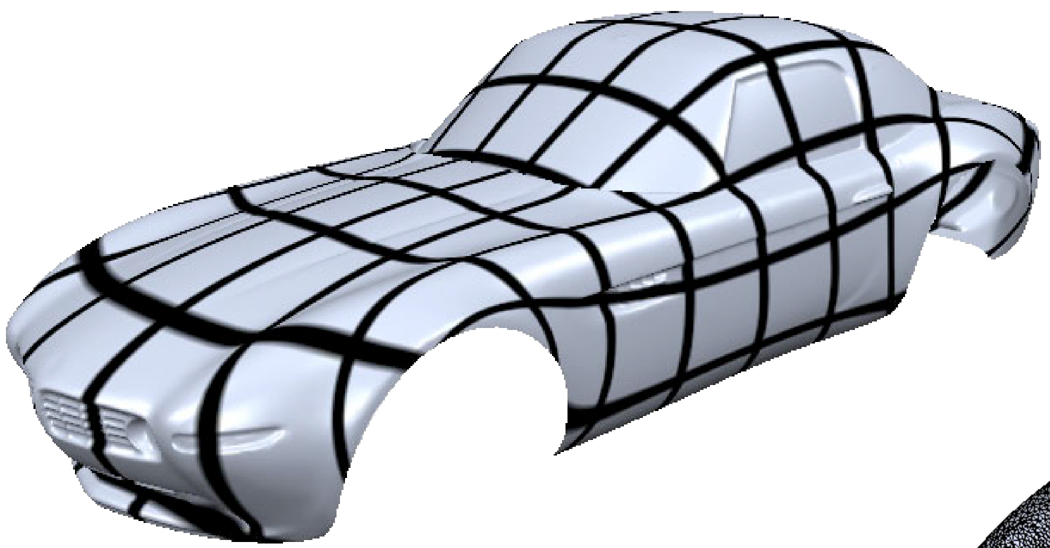




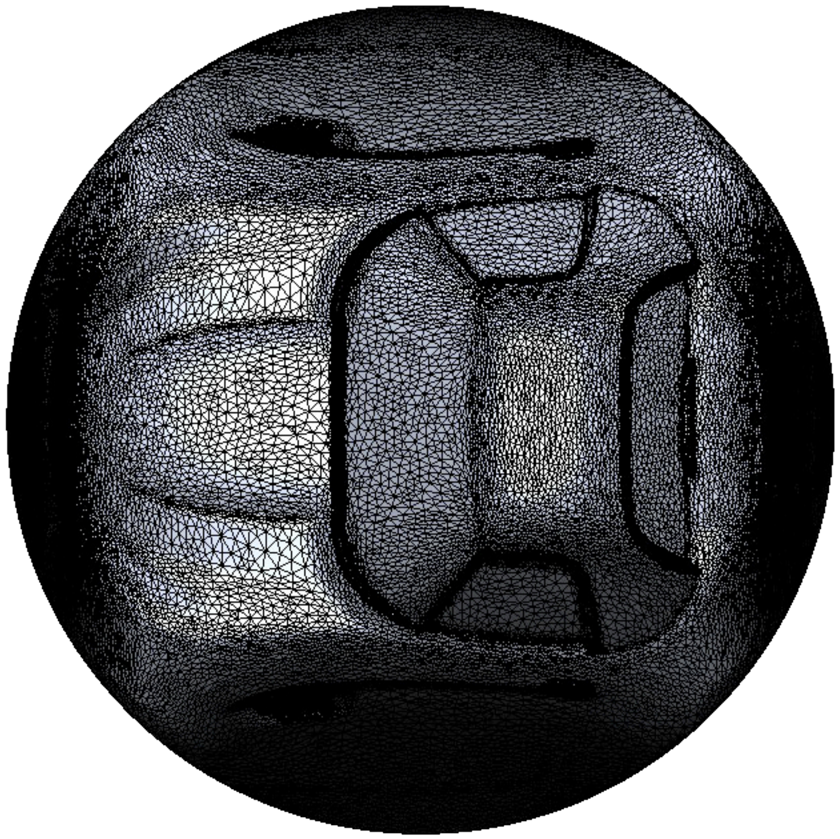
$$\Delta^2 \mathbf{x} = 0$$

Conformal Parameterization

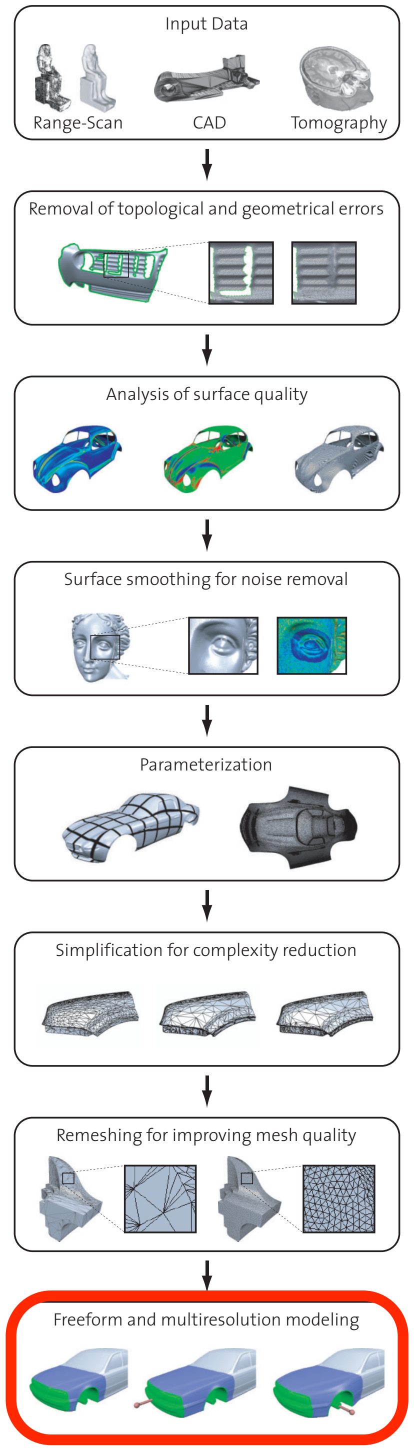


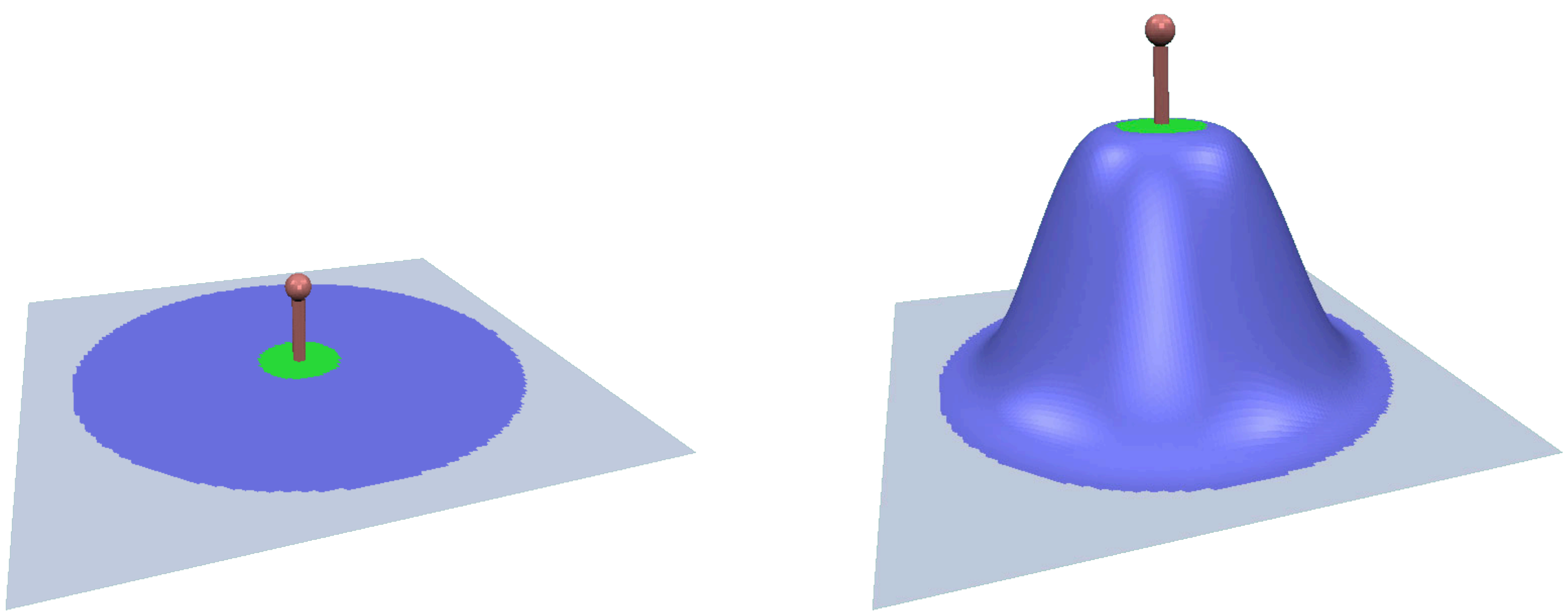


$$\Delta_{\mathcal{S}} \mathbf{u} = 0$$



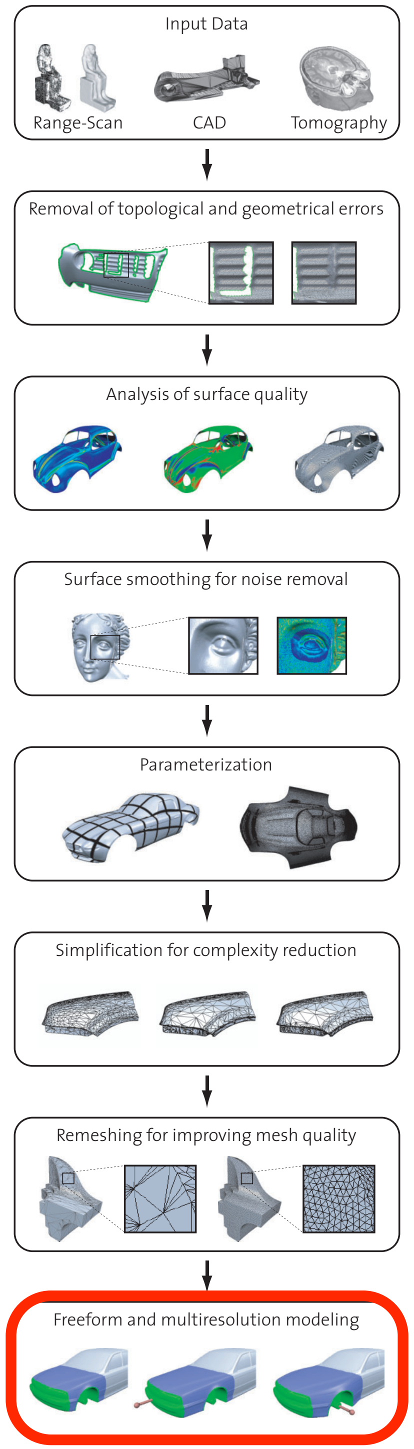
Variational Mesh Editing

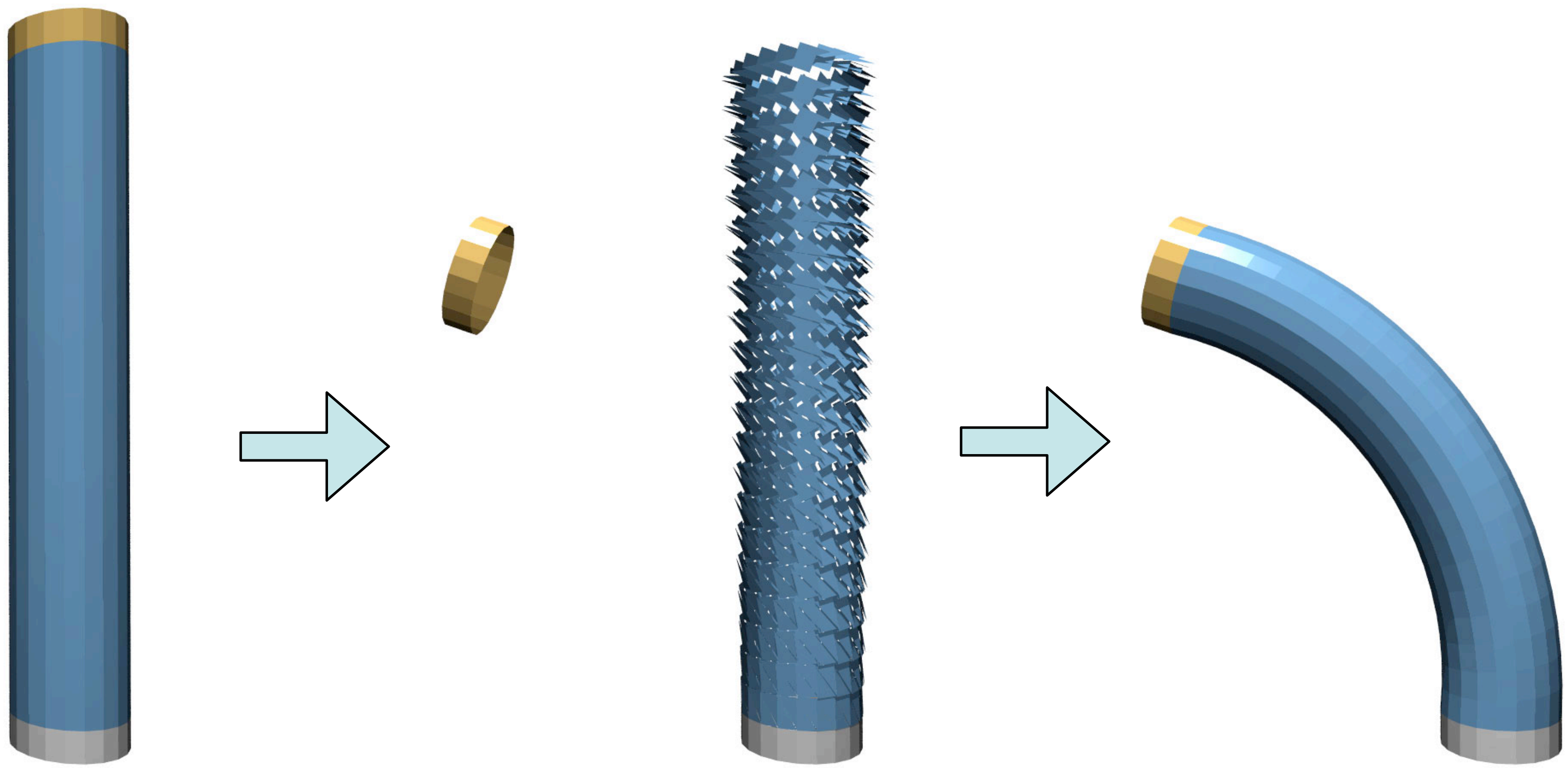




$$\Delta_S^k \mathbf{d} = 0$$

Gradient-Based Editing

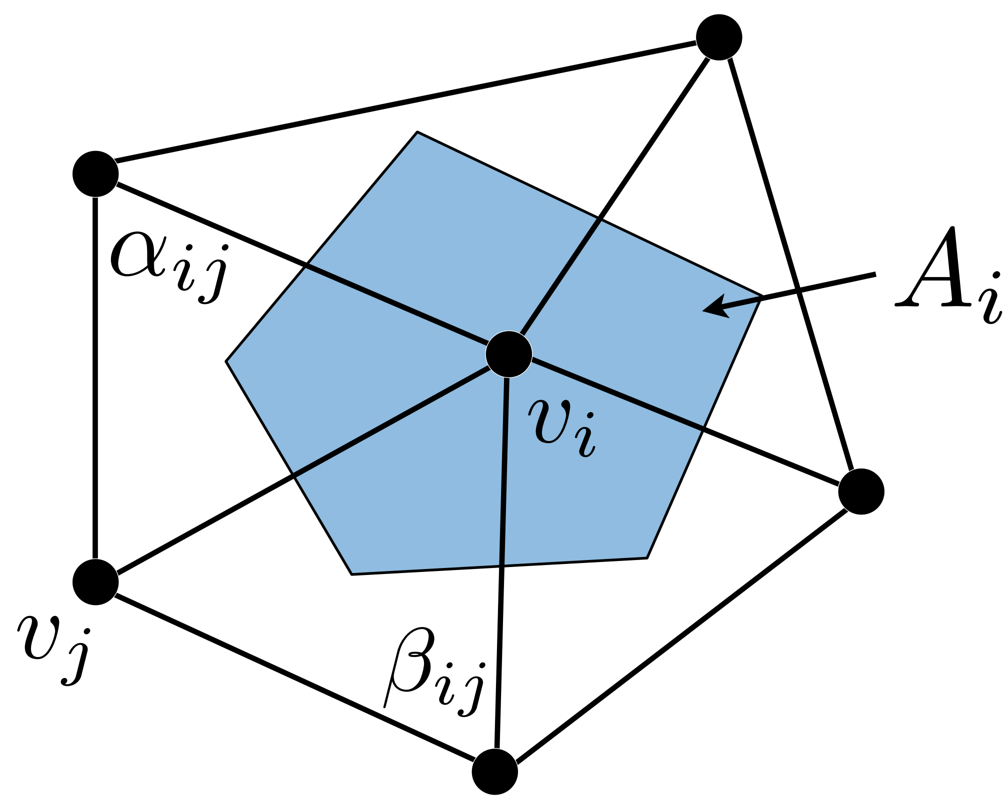




$$\Delta_{\mathcal{S}}\mathbf{p} = \mathrm{div}\,(\mathbf{g})$$

Laplace-Beltrami Discretization

$$\Delta_{\mathcal{S}} f(v) := \frac{2}{A(v)} \sum_{v_i \in \mathcal{N}_1(v)} (\cot\alpha_i + \cot\beta_i)(f(v_i) - f(v))$$



Laplace Matrix

$$\begin{pmatrix} \vdots \\ \Delta_{\mathcal{S}}^k f_i \\ \vdots \end{pmatrix} = (\mathbf{DM})^k \begin{pmatrix} \vdots \\ f_i \\ \vdots \end{pmatrix}$$

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

$$\mathbf{D} = \operatorname{diag}\left(\ldots, \frac{2}{A(v_i)}, \ldots\right)$$

Laplace Matrix

$$\begin{pmatrix} \vdots \\ \Delta_{\mathcal{S}}^k f_i \\ \vdots \end{pmatrix} = (\mathbf{DM})^k \begin{pmatrix} \vdots \\ f_i \\ \vdots \end{pmatrix}$$

- Degree of sparsity: $1 + 3\,(k^2 + k)$
  - k=1 ... 7
  - k=2 ... 19
  - k=3 ... 37

Laplace Matrix

$$\begin{pmatrix} \vdots \\ \Delta_{\mathcal{S}}^k f_i \\ \vdots \end{pmatrix} = (\mathbf{DM})^k \begin{pmatrix} \vdots \\ f_i \\ \vdots \end{pmatrix}$$

- $(\mathbf{DM})^k$ is not symmetric, but $\mathbf{M}(\mathbf{DM})^{k-1}$ is

➡ Instead of $(\mathbf{DM})^k \mathbf{x} = \mathbf{b}$

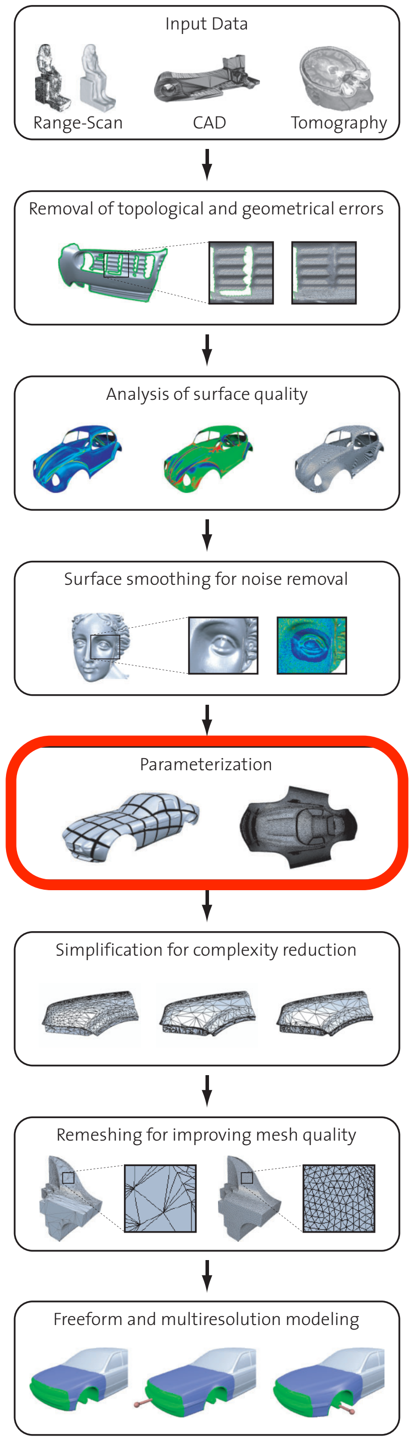
solve $\mathbf{M}(\mathbf{DM})^{k-1} \mathbf{x} = \mathbf{D}^{-1}\mathbf{b}$

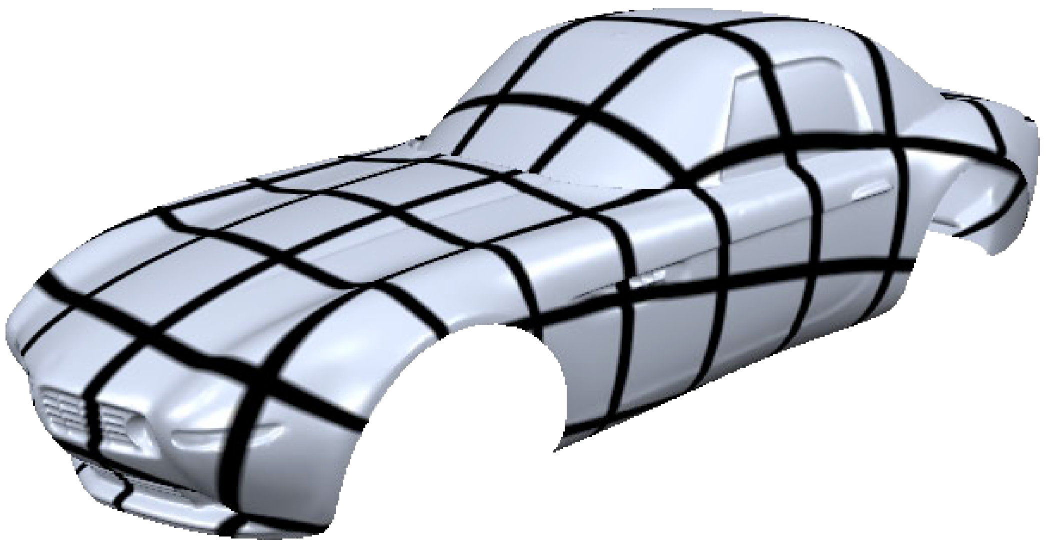
Laplace Matrix

$$\begin{pmatrix} \vdots \\ \Delta_{\mathcal{S}}^k f_i \\ \vdots \end{pmatrix} = (\mathbf{DM})^k \begin{pmatrix} \vdots \\ f_i \\ \vdots \end{pmatrix}$$

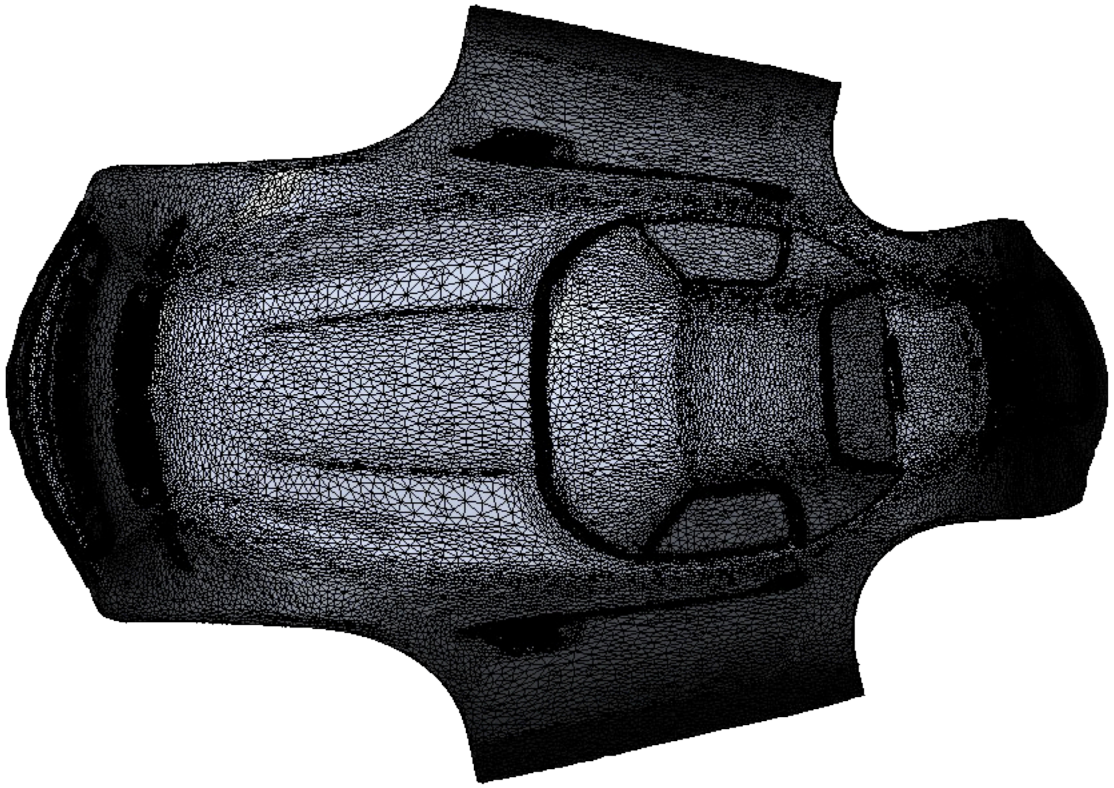
- Positive definiteness
  - Can be derived by variational calculus
  - Energy minimization subject to constraints

Least Squares Conformal Maps

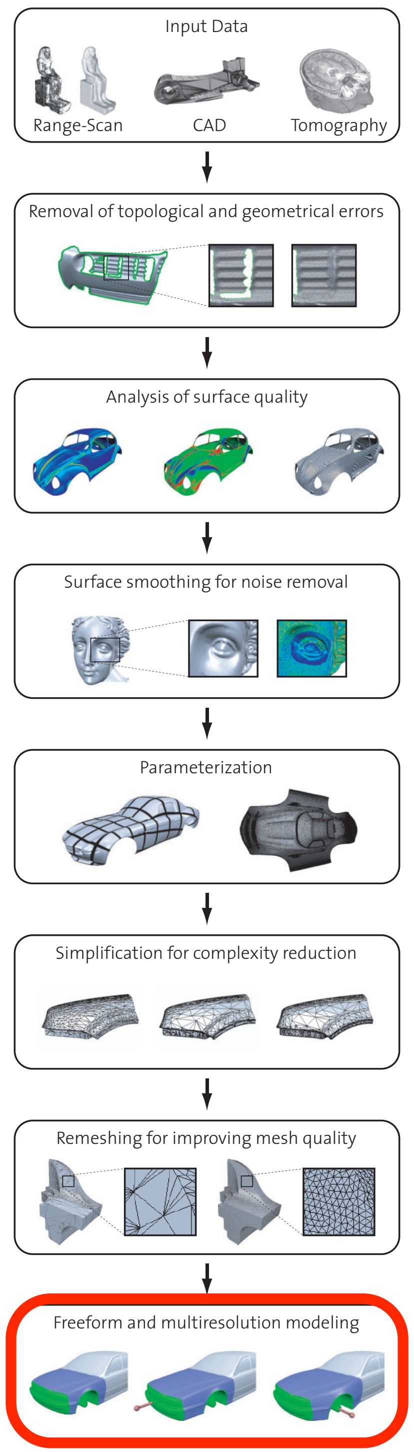




$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$$



Non-Linear Problems



Non-linear minimization (Newton)

$$\mathbf{H}(\mathbf{x})\,\mathbf{h} = -\nabla\mathbf{f}(\mathbf{x})$$

Non-linear least squares (Gauss-Newton)

$$\mathbf{J}(\mathbf{x})^T\,\mathbf{J}(\mathbf{x})\,\mathbf{h} = -\mathbf{J}(\mathbf{x})^T\,\mathbf{f}(\mathbf{x})$$

Overview

- Application scenarios
- **Linear system solvers**
- Benchmarks

Dense Direct Solvers

- Symmetric positive definite (*spd*)
  - Cholesky factorization ($\mathbf{A}=\mathbf{L}\mathbf{L}^T$)
  - Solve systems by back-substitution
  - Numerically stable
- Complexity
  - Factorization $O(n^3)$
  - Back-substitution $O(n^2)$

Iterative Solvers

- Symmetric, positive definite, *sparse*
  - Conjugate gradients
  - Robust, monotone convergence
  - Exact solution after n iterations
- Complexity
  - Each iteration is O(n) (sparse!)
  - Total complexity $O(n^2)$

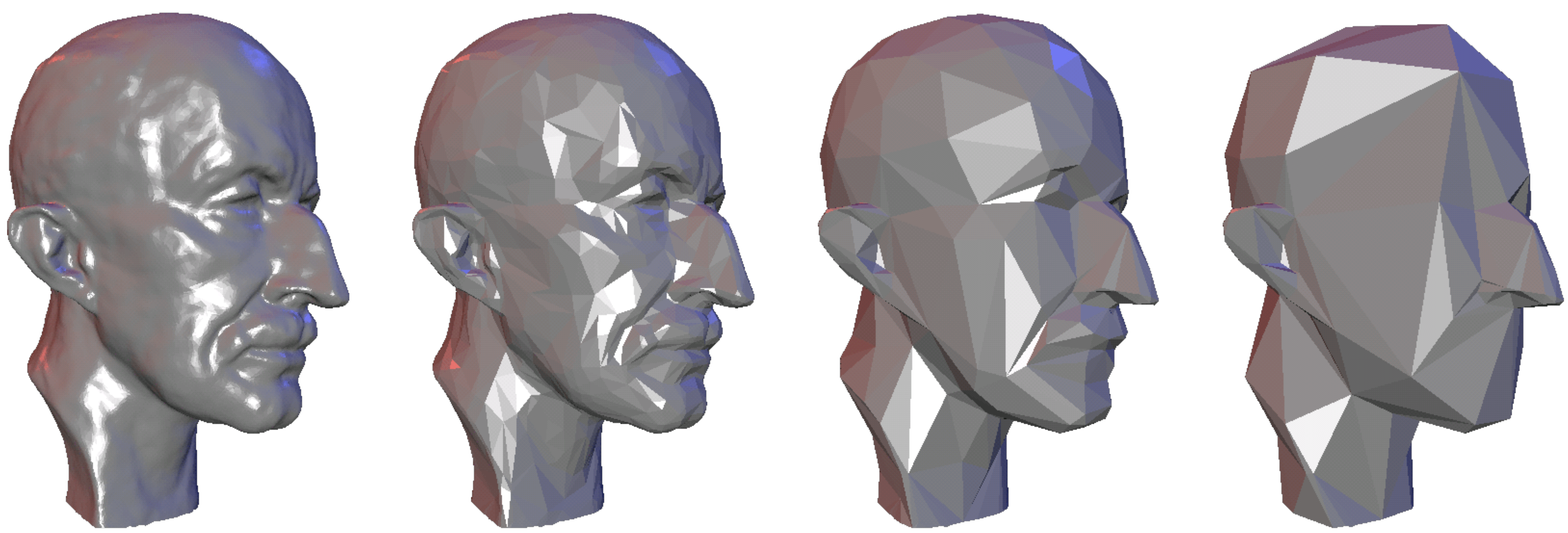
Iterative Solvers

- Numerical convergence rate
  - Depends on matrix condition
  - Preconditioning is mandatory ($\mathbf{A}' = \mathbf{P}\mathbf{A}\mathbf{P}^T$)
  - Problematic for large systems ...

- Iterative solvers are “smoothers”
  - Rapid elimination of high frequency errors
  - Impractically slow convergence for low frequencies

Multigrid Solvers

- Build a hierarchy of meshes
  - Mesh decimation
  - O(log n) levels



Multigrid Solvers

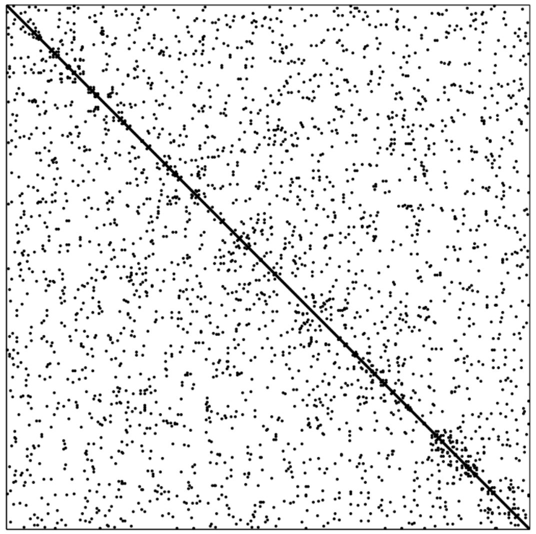
- Apply some pre-smoothing steps on finest level
  - Removes highest error frequencies
- Remaining low frequency error (**r**=**b**-**Ax**)
  - Corresponds to high frequencies on coarser levels
  - Iterate / solve residual system (**Ae**=**r**) on coarse level
- Propagate solution to finer level
  - Followed by post-smoothing steps
- Total O(n) complexity!

Multigrid Solvers

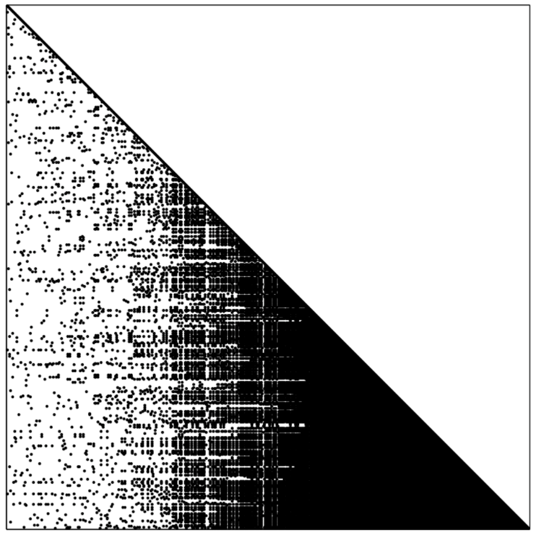
- MG can be quite tricky:
  - How to build an irregular hierarchy ?
  - How many levels ?
  - Special MG pre-conditioners
  - Restriction of system
  - Prolongation of coarse solution
- [Aksoylu et al. 2003], [Shi et al. 2006]

Direct Sparse Solvers

- Dense solvers do not exploit sparsity
  - Matrix factors are dense



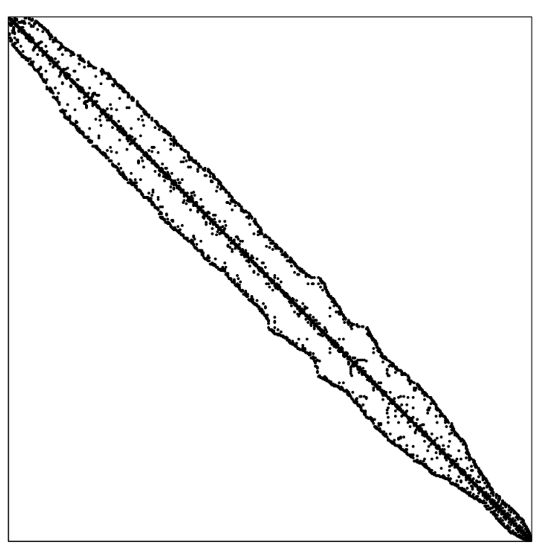
$A=LL^T$

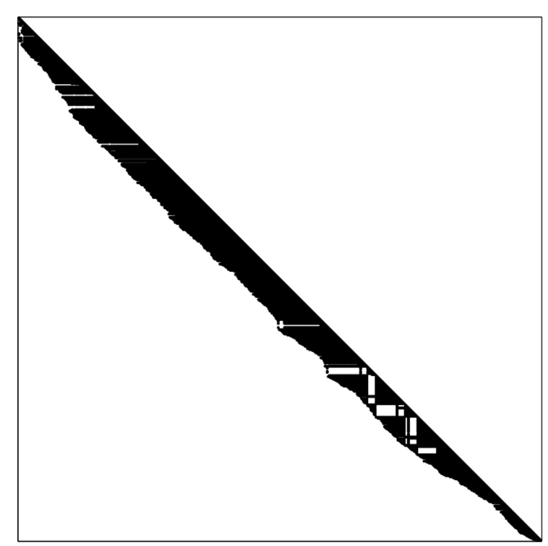


L

Direct Sparse Solvers

- Dense solvers do not exploit sparsity
  - Matrix factors are dense
- Band-limitation can be exploited
  - Bandwidth of factors is that of **A**
  - More precisely: envelope is preserved





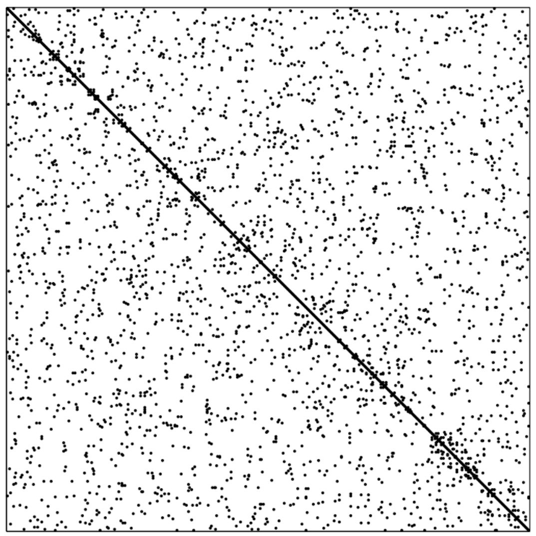
Direct Sparse Solvers

- Dense solvers do not exploit sparsity
  - Matrix factors are dense
- Band-limitation can be exploited
  - Bandwidth of factors is that of **A**
  - More precisely: envelope is preserved
- Complexity
  - Factorization $O(nb^2)$
  - Back-substitution $O(nb)$

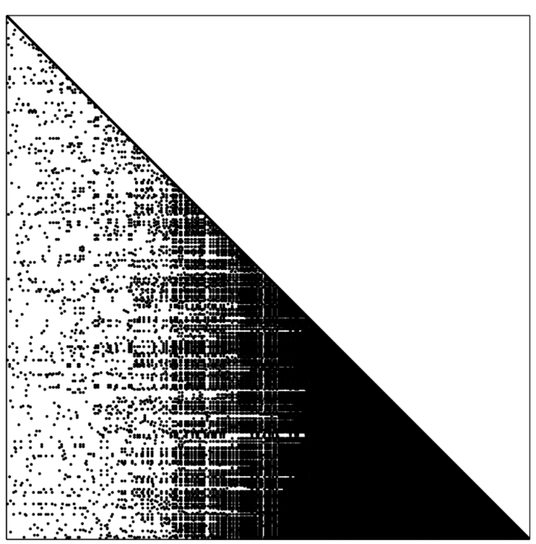
Matrix Re-Ordering

Natural

LL^T



L



36k NZ

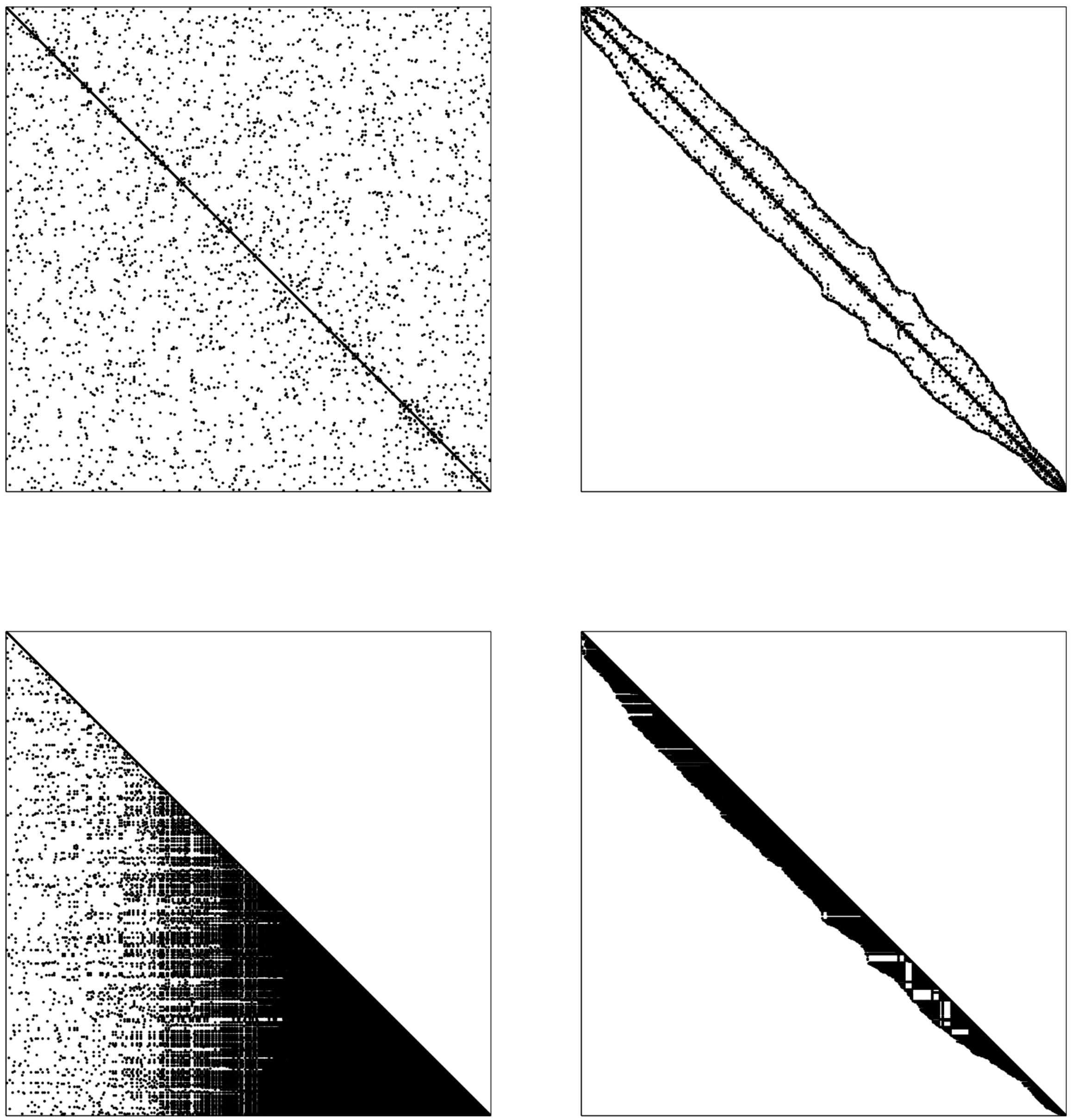
Matrix Re-Ordering

- Find symmetric permutation $\mathbf{A}' = \mathbf{P}^{T}\mathbf{AP}$
- ... which minimizes the band-width:
  - ➡ Cuthill-McKee algorithm

Matrix Re-Ordering

Natural RCMK

LL^T



L

36k NZ 14k NZ

Matrix Re-Ordering

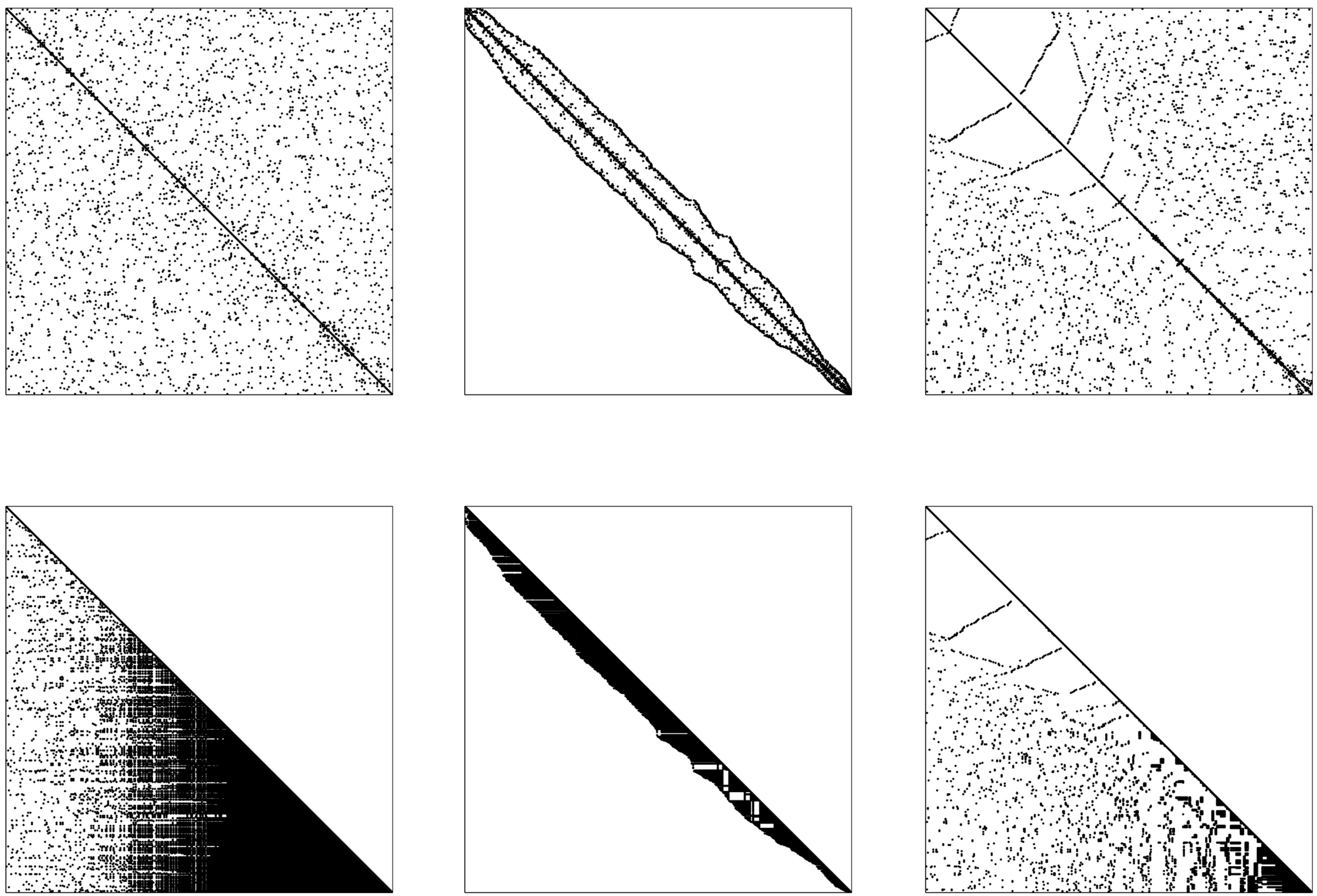
- Find symmetric permutation **A' = P^{T}AP**
- ... which minimizes the band-width:
  - ➡ Cuthill-McKee algorithm
- ... which minimizes the envelope fill-in of **L**:
  - ➡ Minimum Degree algorithm

Matrix Re-Ordering

Natural | RCMK | MD

LL^T

L



36k NZ | 14k NZ | 6.2k NZ

Matrix Re-Ordering

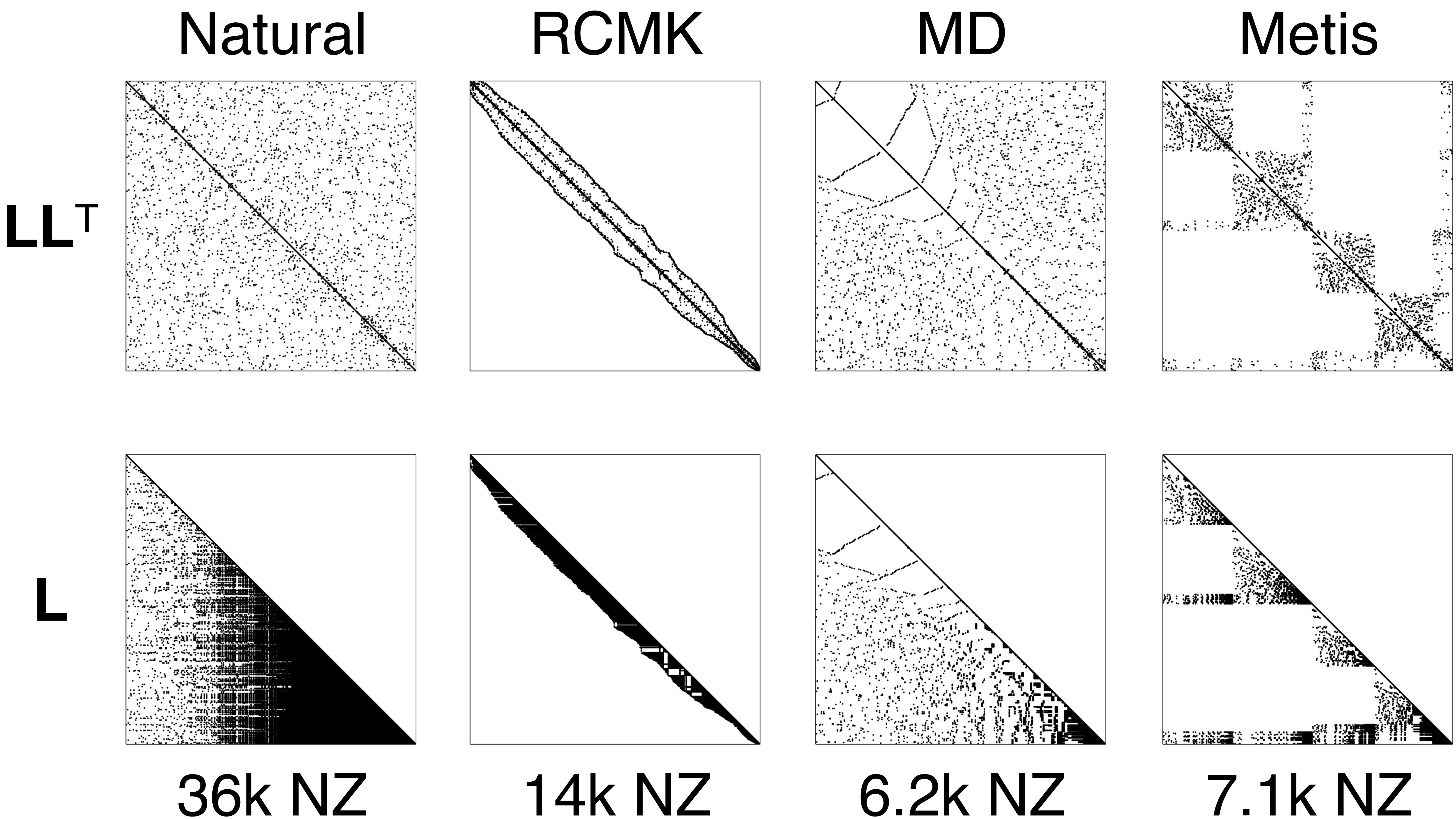
- Find symmetric permutation $\mathbf{A}' = \mathbf{P}^T\mathbf{A}\mathbf{P}$

- ... which minimizes the band-width:
  ➡ Cuthill-McKee algorithm

- ... which minimizes the envelope fill-in of **L**:
  ➡ Minimum Degree algorithm

- ... based on recursive graph partitioning:
  ➡ METIS algorithm

Matrix Re-Ordering



Sparse Cholesky Factorization

- Non-zero structure of **L** can be predicted from the non-zero structure of **A**
  - Build a static data structure in advance
  - *Symbolic factorization*
- Compute numerical entries of **L** based on this data structure
  - Better memory coherence
  - *Numerical factorization*

Sparse Cholesky Solver

1. Matrix re-ordering $\tilde{\mathbf{A}} = \mathbf{P}^T \mathbf{A} \mathbf{P}$
2. Symbolic factorization $\mathbf{L}$
3. Numerical factorization $\tilde{\mathbf{A}} = \mathbf{L}\mathbf{L}^T$
4. Solve system $\mathbf{y} = \mathbf{L}^{-1}\mathbf{P}^T\mathbf{b}, \quad \mathbf{x} = \mathbf{P}\mathbf{L}^{-T}\mathbf{y}$

Sparse Cholesky Solver

Only right hand side changes

1. Matrix re-ordering $\tilde{\mathbf{A}} = \mathbf{P}^T \mathbf{A} \mathbf{P}$
2. Symbolic factorization $\mathbf{L}$
3. Numerical factorization $\tilde{\mathbf{A}} = \mathbf{L}\mathbf{L}^T$
4. Solve system $\mathbf{y} = \mathbf{L}^{-1}\mathbf{P}^T\mathbf{b}, \quad \mathbf{x} = \mathbf{P}\mathbf{L}^{-T}\mathbf{y}$

Sparse Cholesky Solver

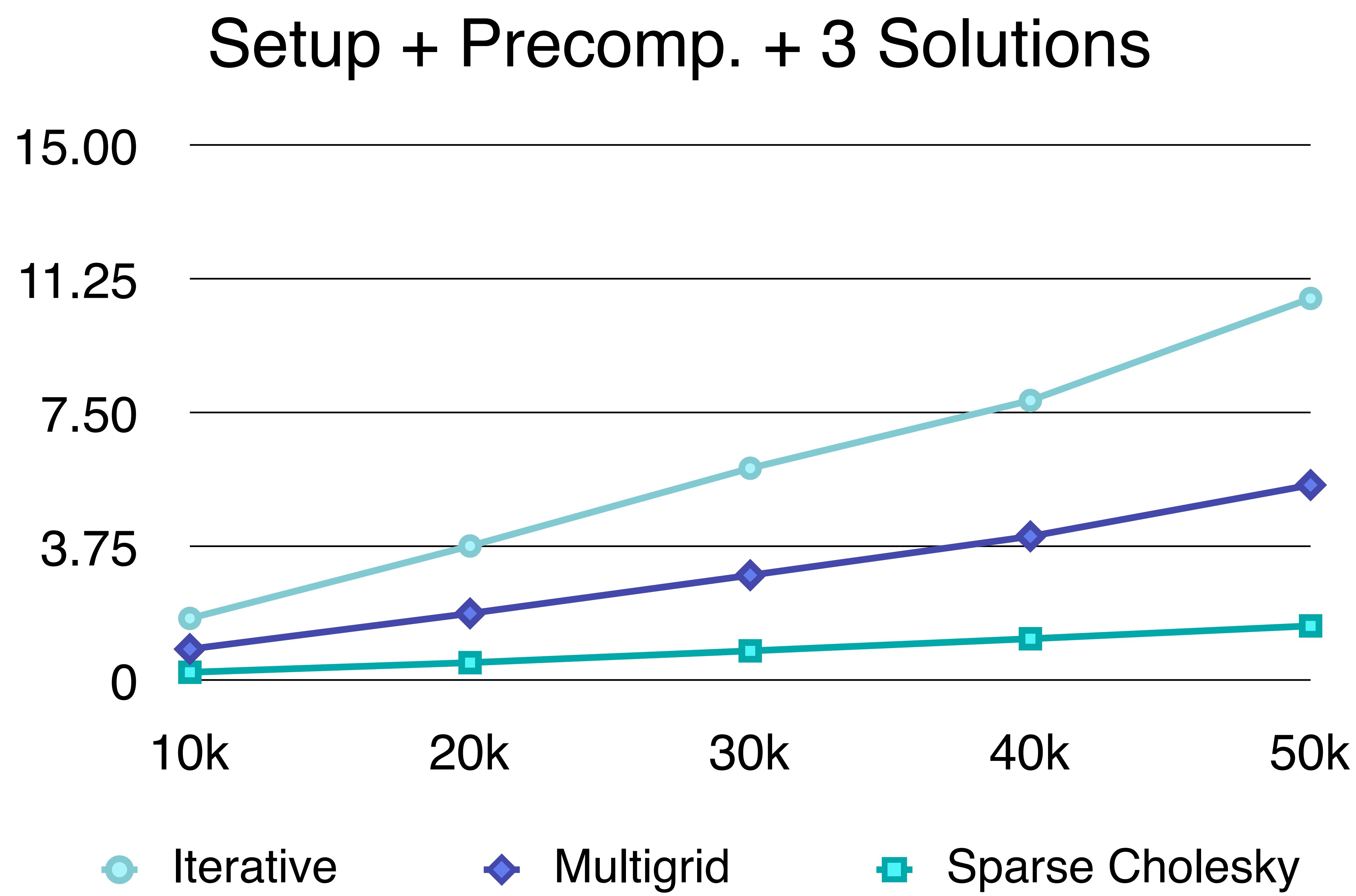
Matrix values change

1. Matrix re-ordering $\tilde{\mathbf{A}} = \mathbf{P}^T \mathbf{A} \mathbf{P}$
2. Symbolic factorization $\mathbf{L}$
3. Numerical factorization $\tilde{\mathbf{A}} = \mathbf{L}\mathbf{L}^T$
4. Solve system $\mathbf{y} = \mathbf{L}^{-1}\mathbf{P}^T\mathbf{b}, \quad \mathbf{x} = \mathbf{P}\mathbf{L}^{-T}\mathbf{y}$

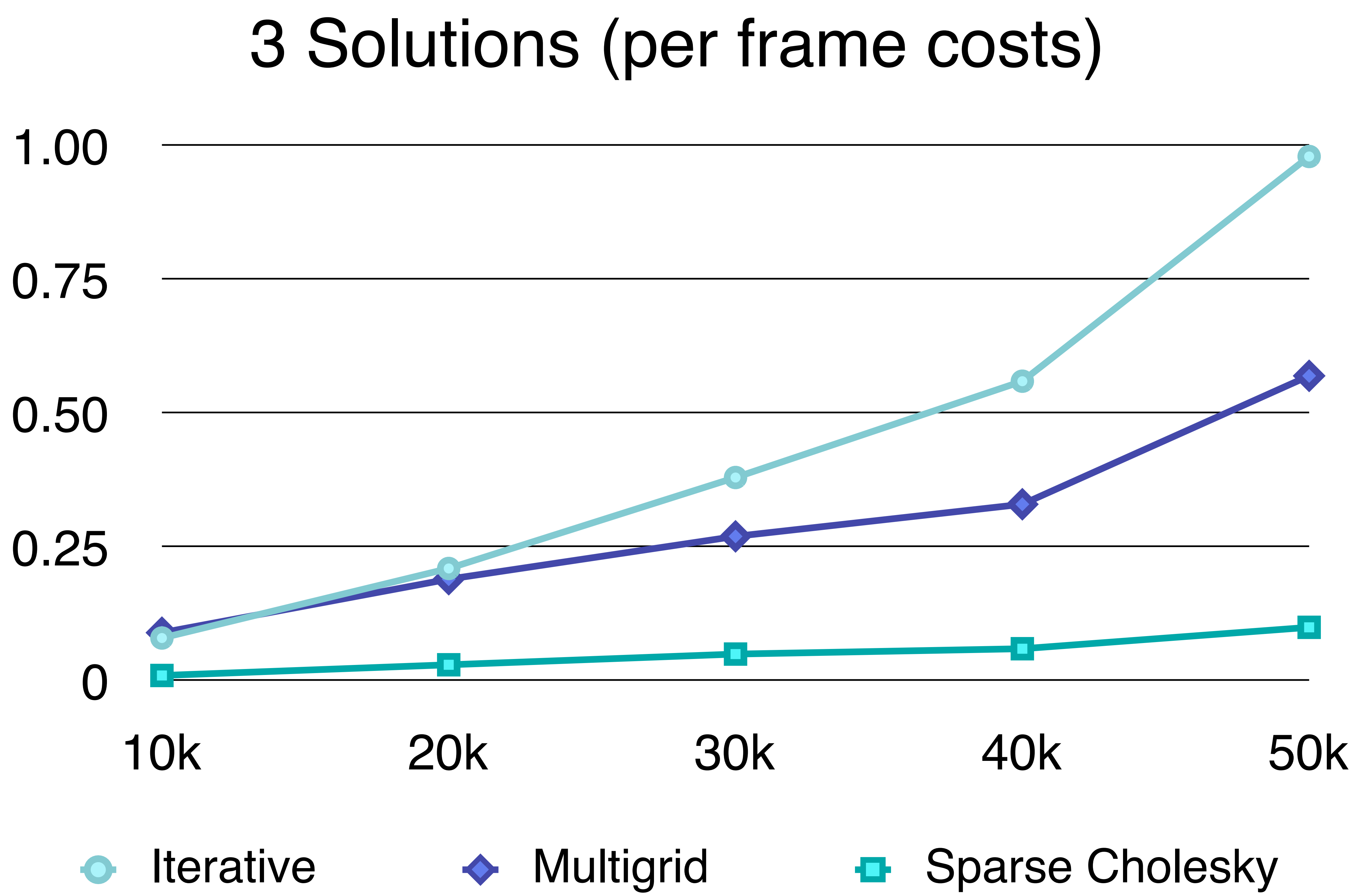
Overview

- Application scenarios
- Linear system solvers
- **Benchmarks**

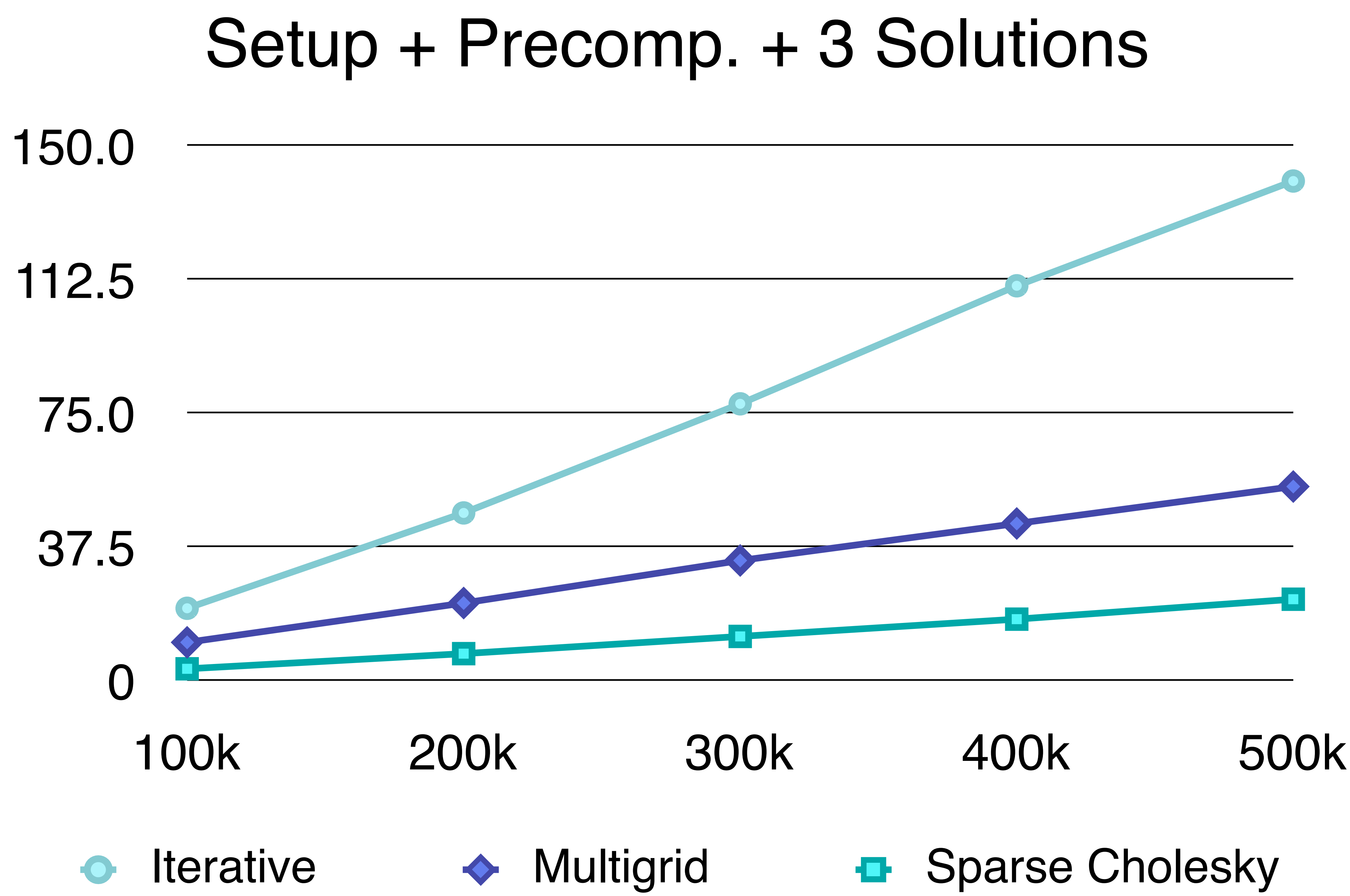
Small Laplace Systems



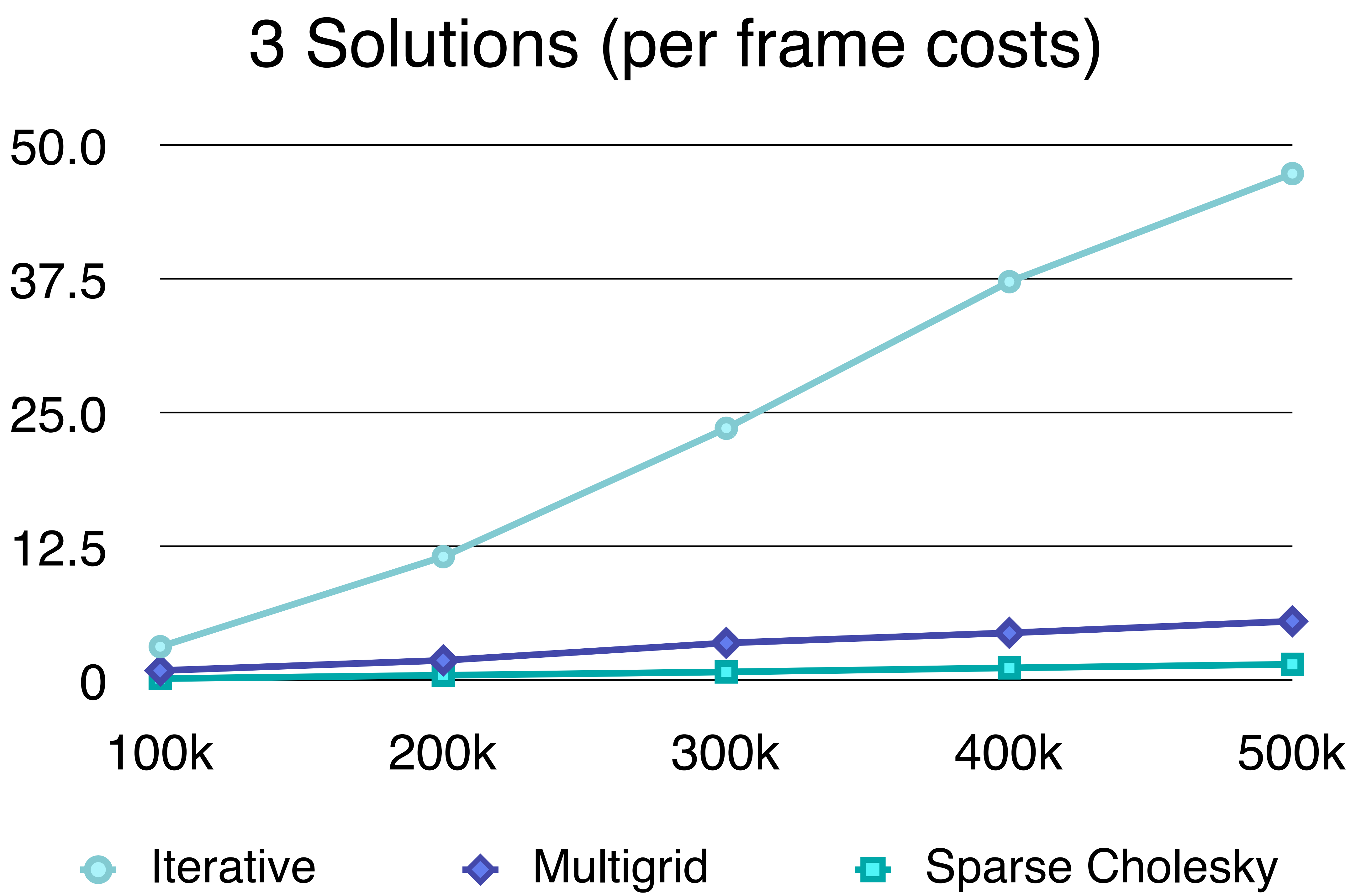
Small Laplace Systems



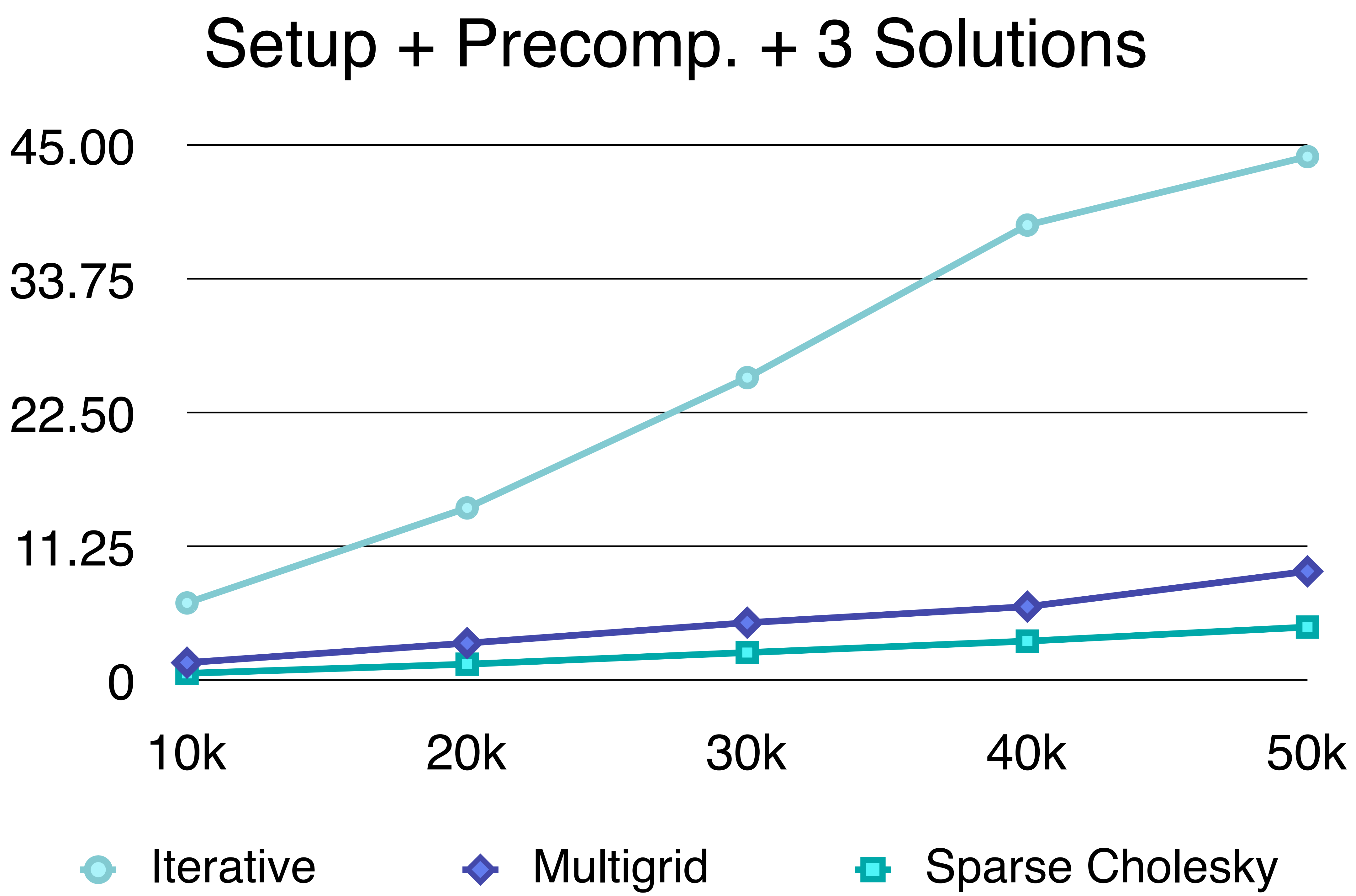
Large Laplace Systems



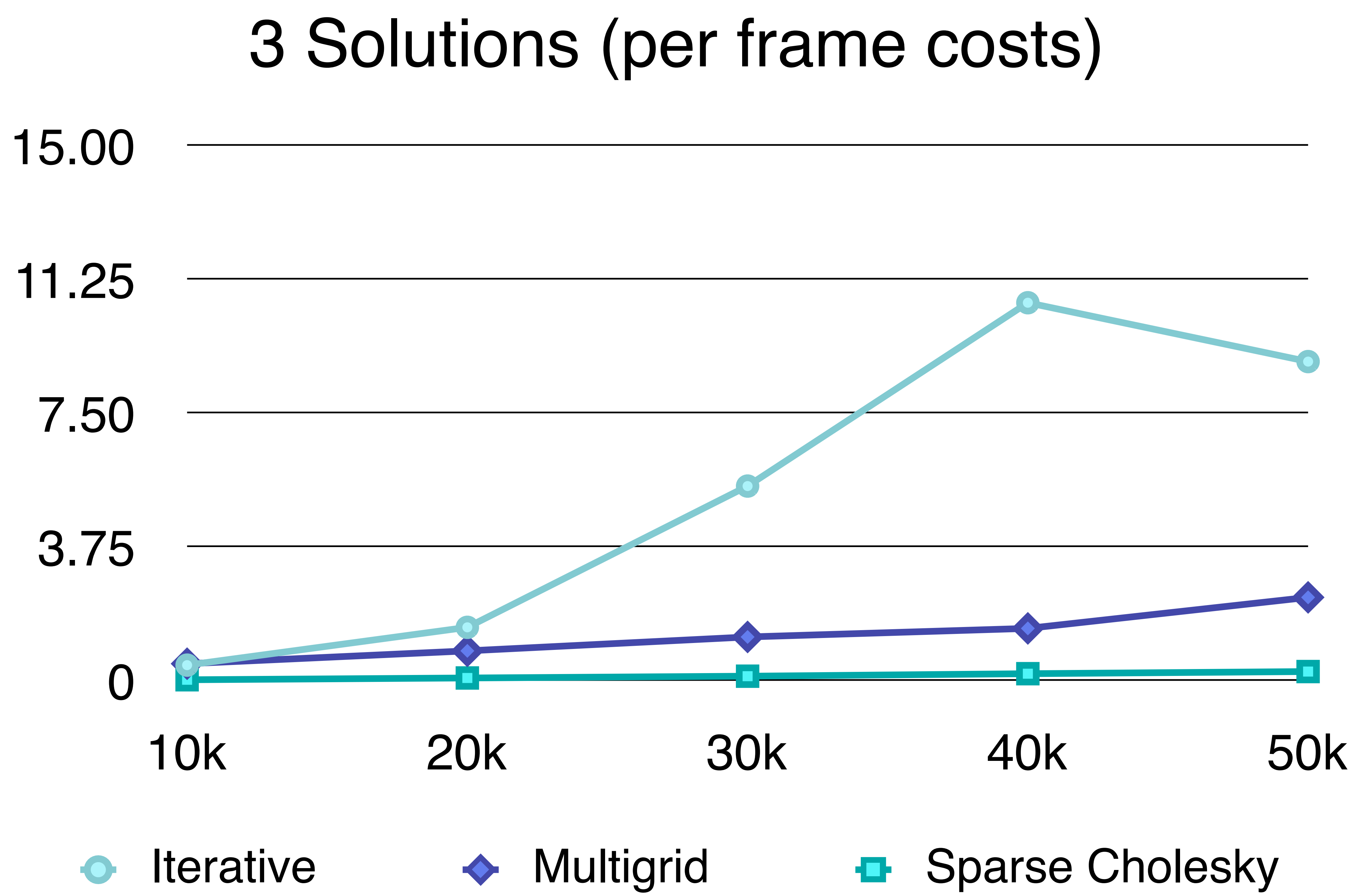
Large Laplace Systems



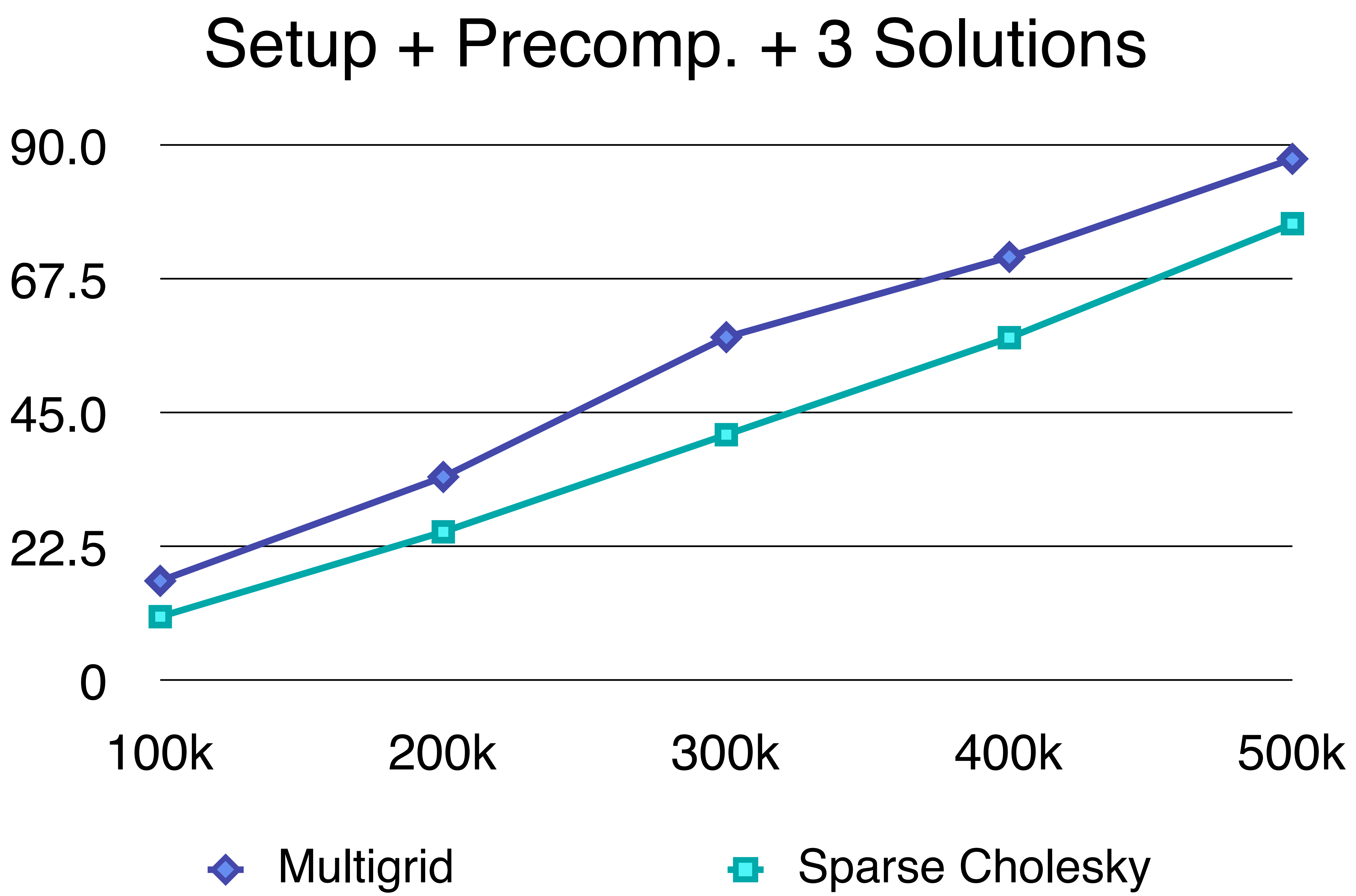
Small Bi-Laplace Systems



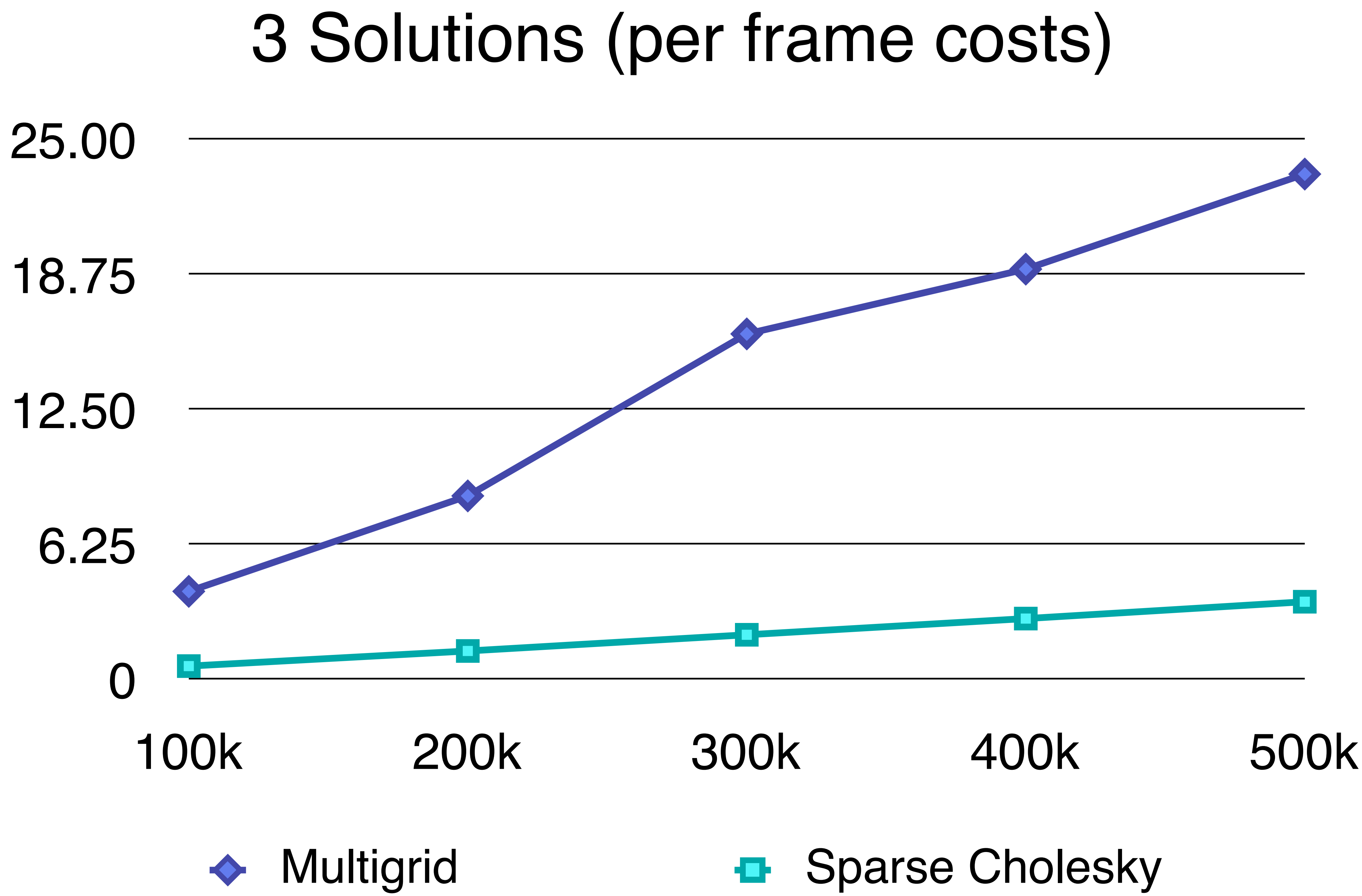
Small Bi-Laplace Systems



Large Bi-Laplace Systems

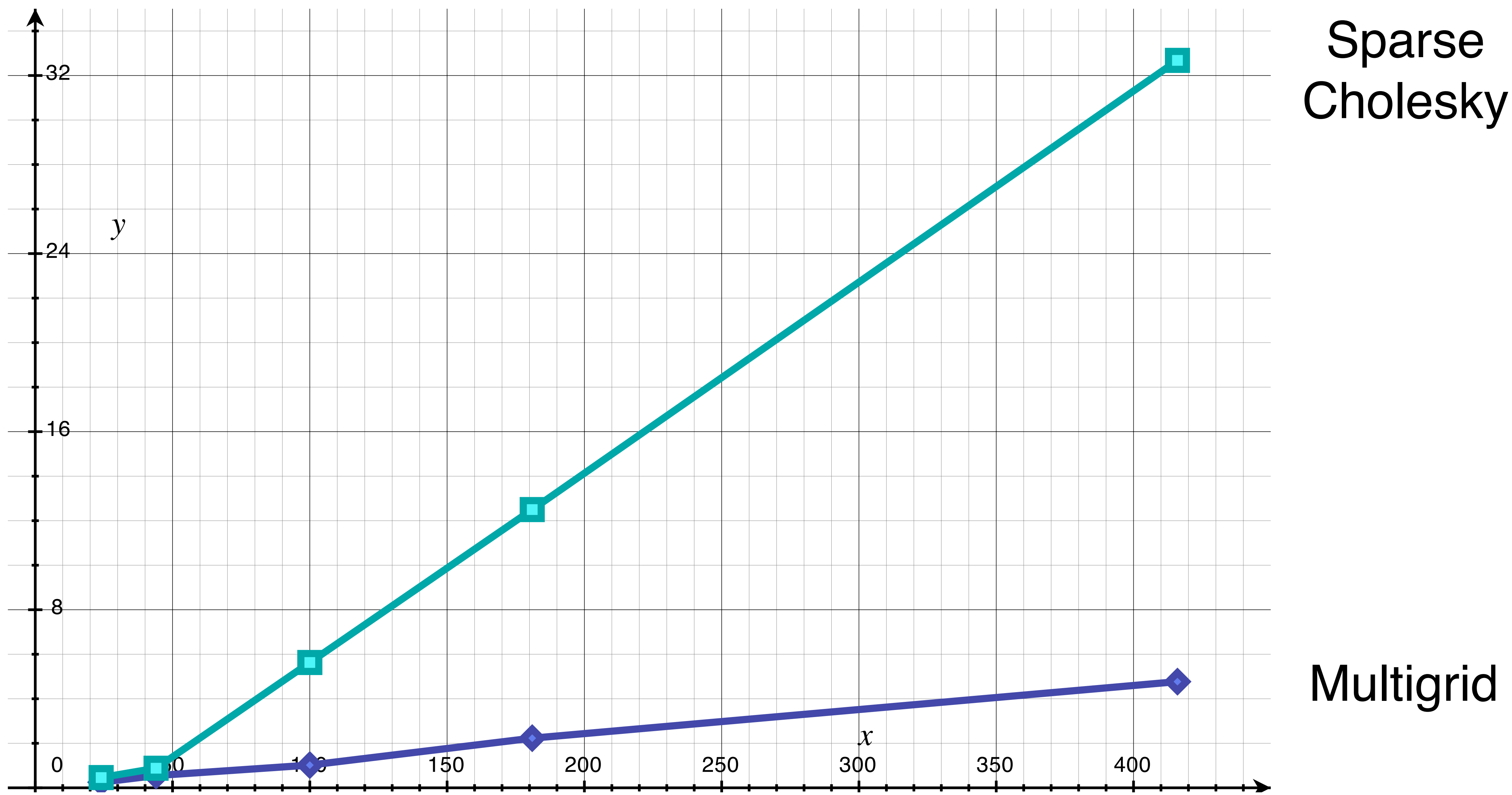


Large Bi-Laplace Systems



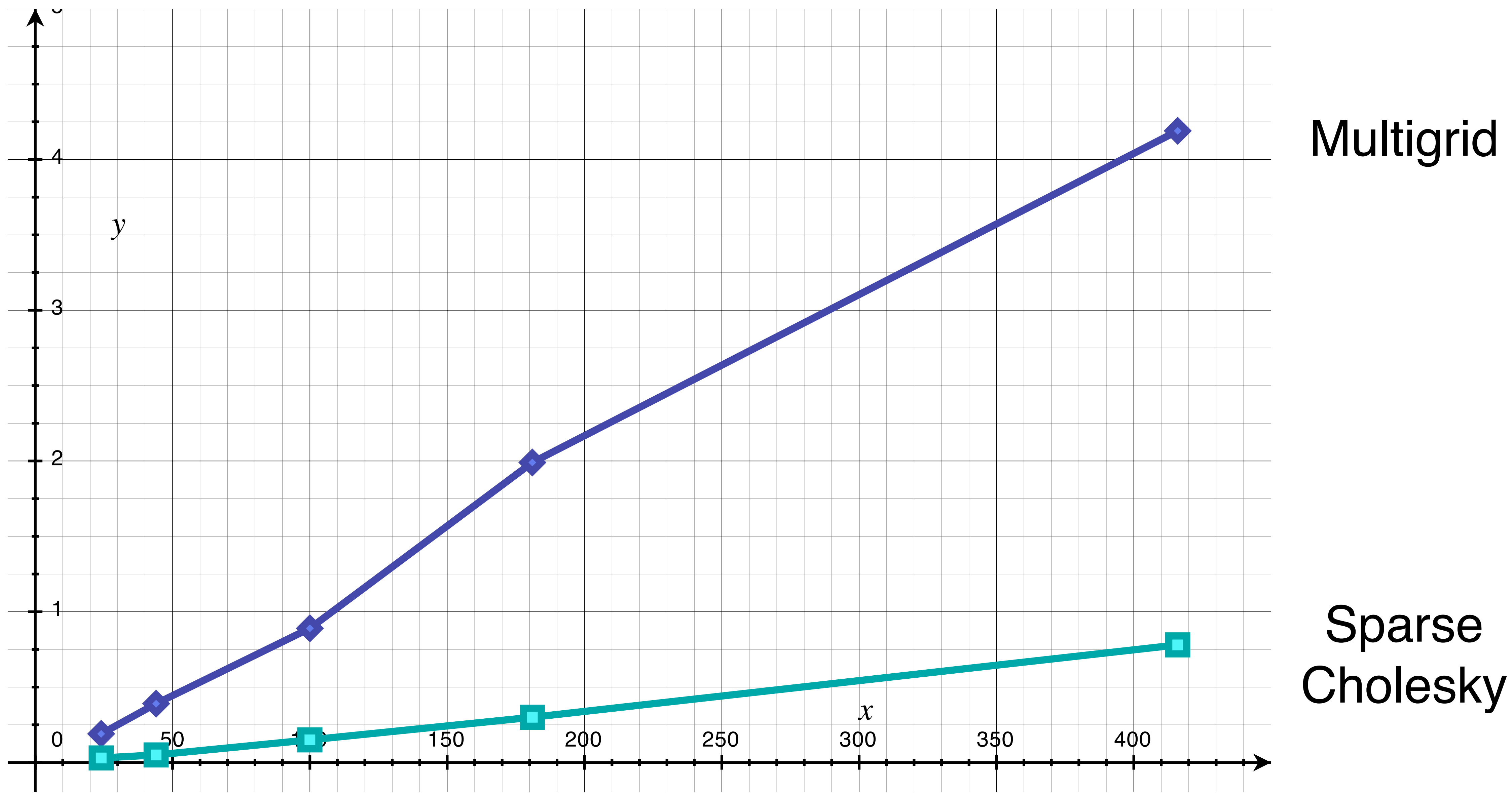
Shi et al., Fast MG Algo

Setup + Precomp. + 3 Solutions



Shi et al., Fast MG Algo

3 Solutions (per frame costs)



Conclusion

- Typical geometry processing problems are
  - large but sparse
  - symmetric positive definite
- Multigrid solvers
  - Require careful implementation
  - Use it if mesh / matrix changes frequently
- Direct sparse solvers
  - Easy to use (black-box)
  - Well suited for multiple rhs, or if only matrix values change