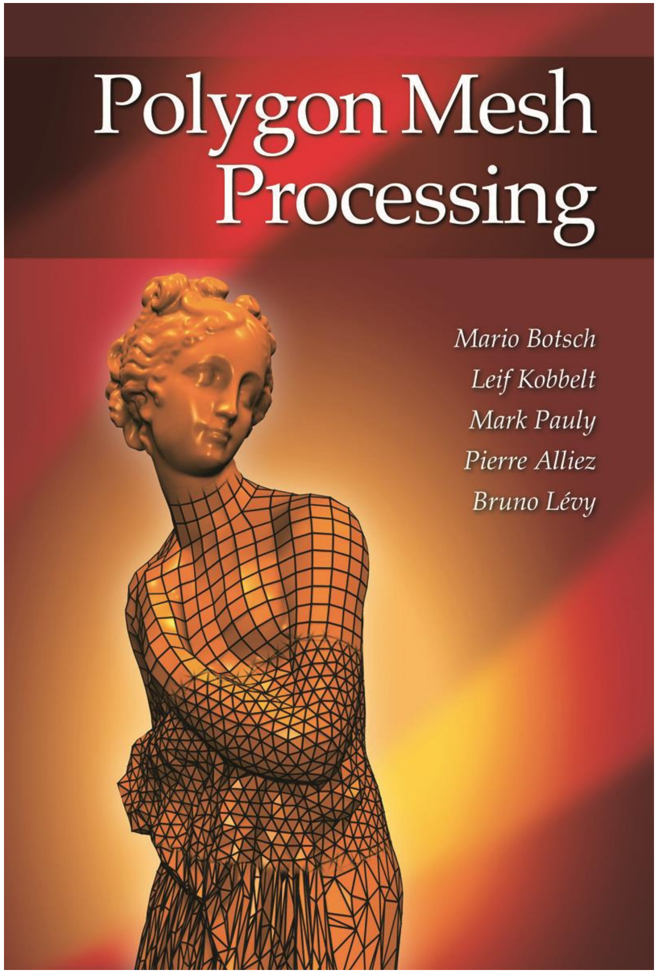
GAMES
Mesh Processing

Qixing Huang
August 26^{th} 2021

Material from



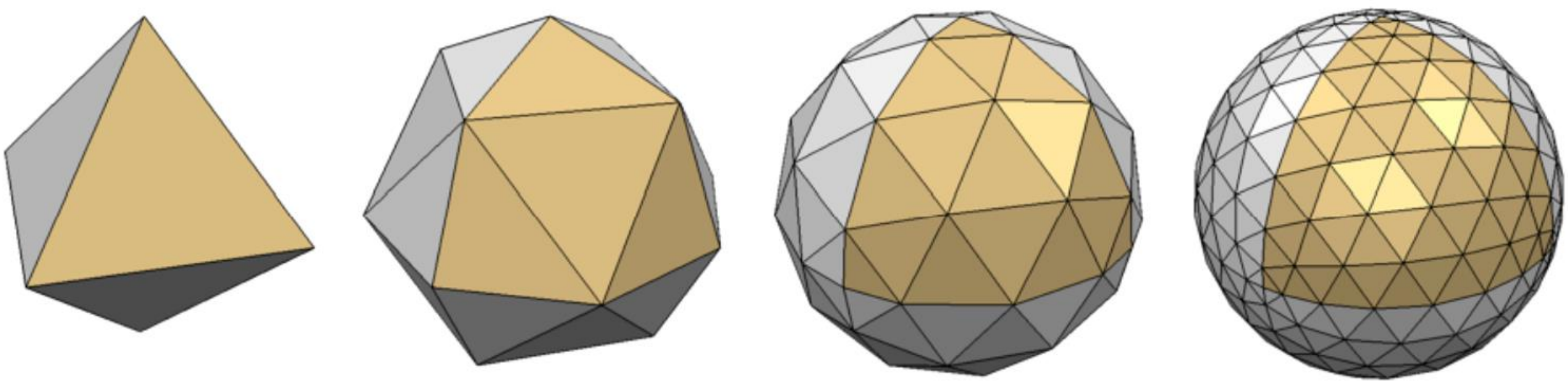
A comprehensive book

Well-written

Lecture slides available

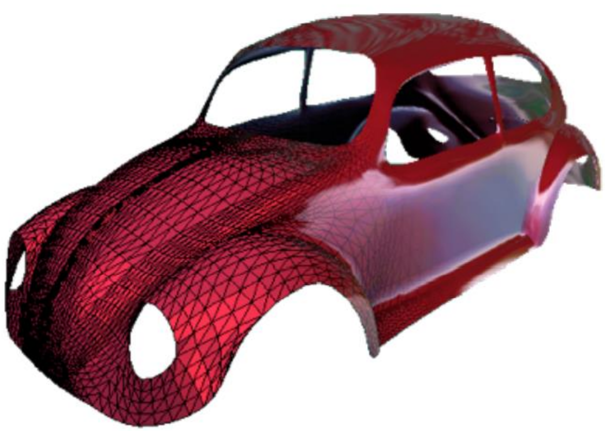
Take-home messages

- Approximation error *#faces = const.
- Arbitrary topology
- Flexibility for piecewise smooth surfaces
- Flexibility for adaptive refinement



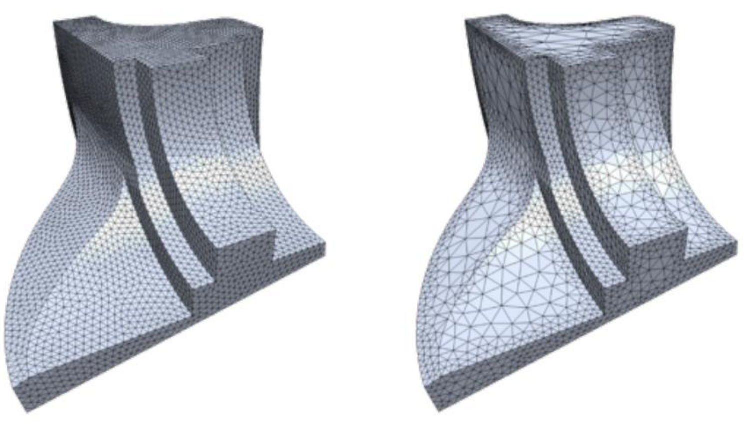
Take-home messages

- Approximation error *#faces = const.
- Arbitrary topology
- Flexibility for piecewise smooth surfaces
- Flexibility for adaptive refinement



Take-home messages

- Approximation error *#faces = const.
- Arbitrary topology
- Flexibility for piecewise smooth surfaces
- Flexibility for adaptive refinement



Take-home messages

- Approximation error *#faces = const.
- Arbitrary topology
- Flexibility for piecewise smooth surfaces
- Flexibility for adaptive refinement

- Implicit representation can support efficient access to vertices, faces…

Typical Operations

Evaluation

- smooth parametric surfaces
  - 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)

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

$$0 \leq \alpha, \quad 0 \leq \beta, \quad \alpha + \beta \leq 1$$

Evaluation

- smooth parametric surfaces
  - 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

- smooth parametric surfaces
  - 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)

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

Distance Queries

- parametric
  - for smooth surfaces: find orthogonal base point

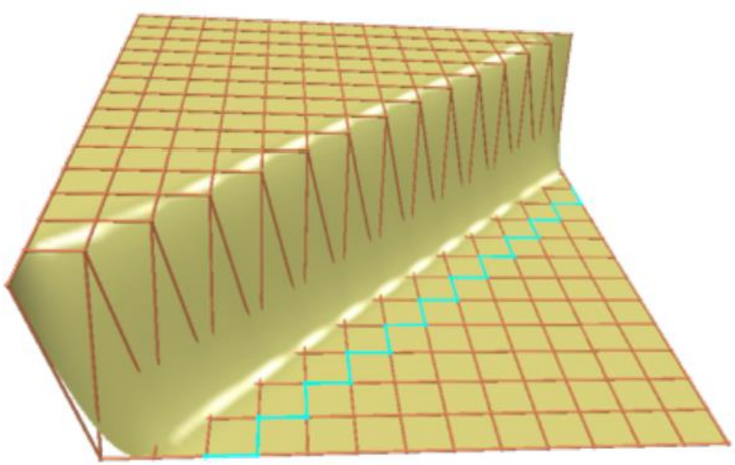
$$[\mathbf{p} - \mathbf{f}(u, v)] \times \mathbf{n}(u, v) \; = \; \mathbf{0}$$

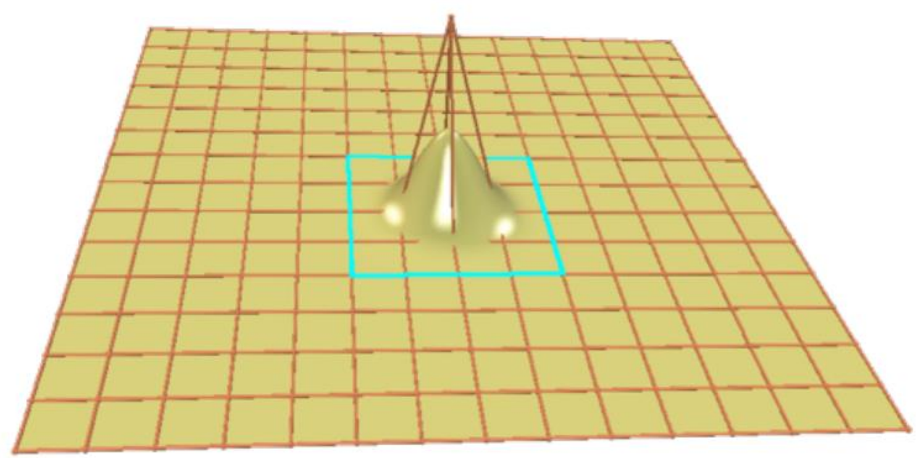
  - for triangle meshes
    - use kd-tree or BSP to find closest triangle
    - find base point by Newton iteration (use Phong normal field)

Modifications

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

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

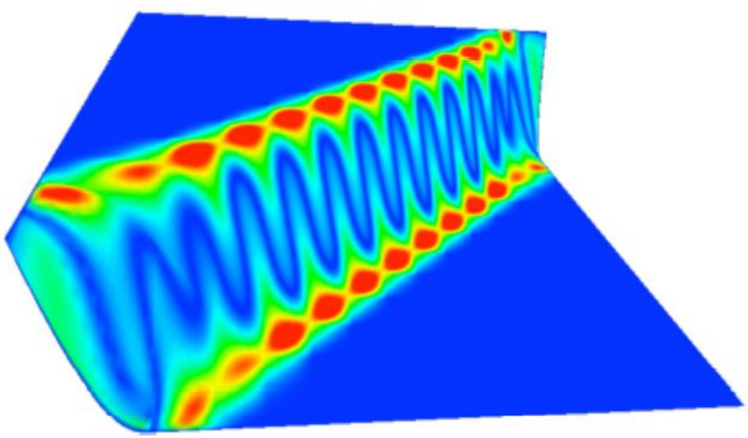


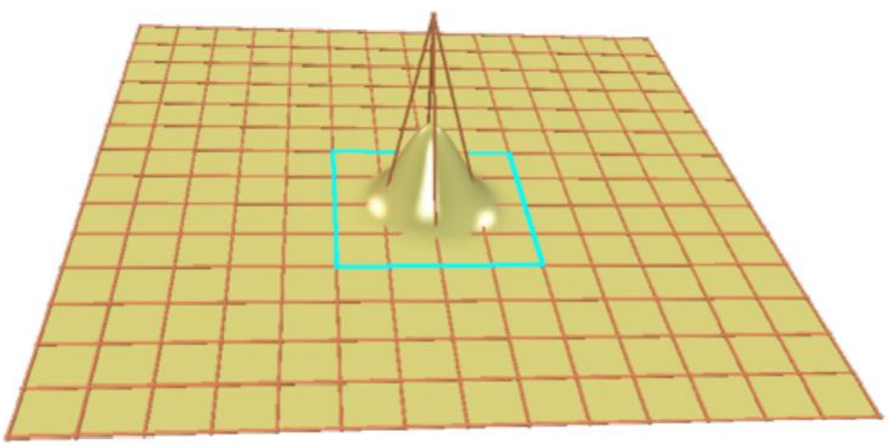


Modifications

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

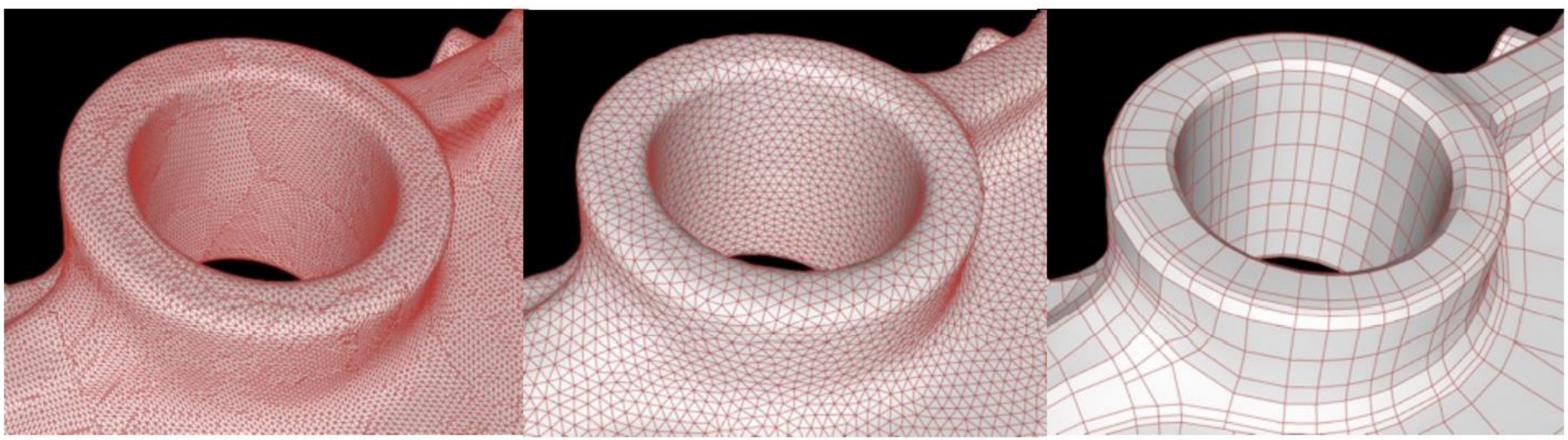
$$\mathbf{f}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{c}_{ij} N_i^n(u) N_j^m(v)$$





Modifications

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



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

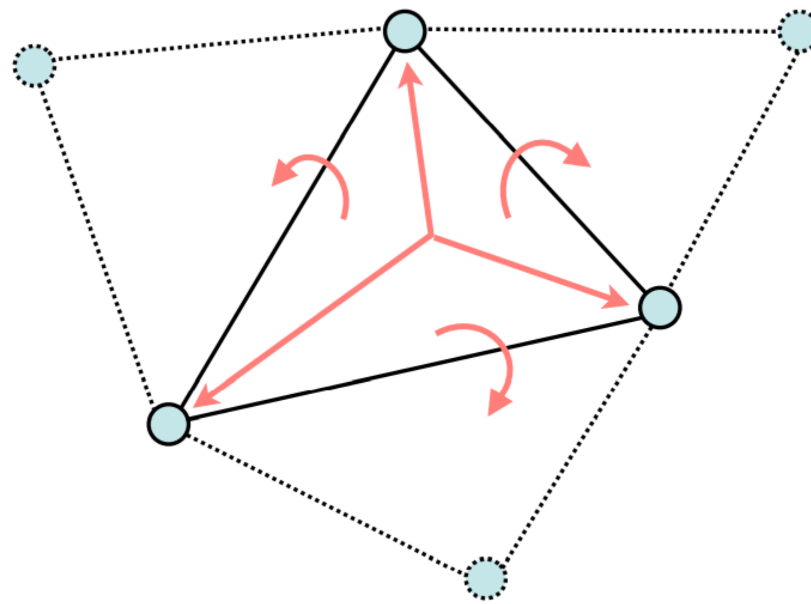
Vertices
x_1 y_1 z_1
...
x_V y_V z_V

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

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

Face-Based Connectivity

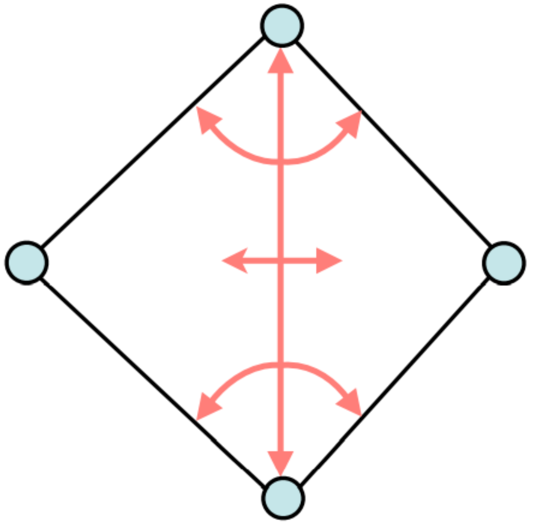
- Vertex
  - Position
  - 1 Face
- Face
  - 3 Vertices
  - 3 Face neighbors



64 B/v

Edge-Based Connectivity

- Vertex
  - Position
  - 1 Edge
- Edge
  - 2 vertices
  - 2 faces
  - 4 edges
- Face
  - 1 edge

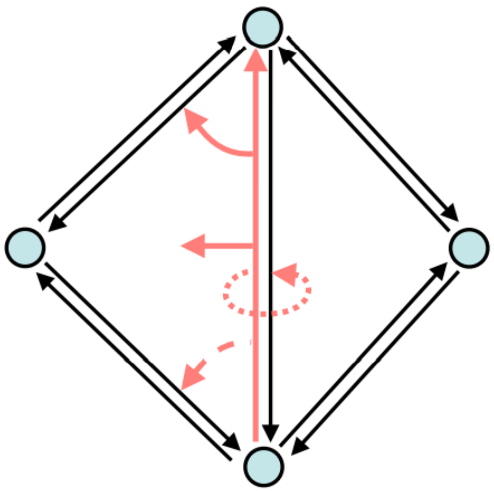


120 B/v

edge orientation?

Halfedge-Based Connectivity

- Vertex
  - Position
  - 1 halfedge
- Halfedge
  - 1 vertex
  - 1 face
  - 1,2,or 3 halfedges
- Face
  - 1 halfedge

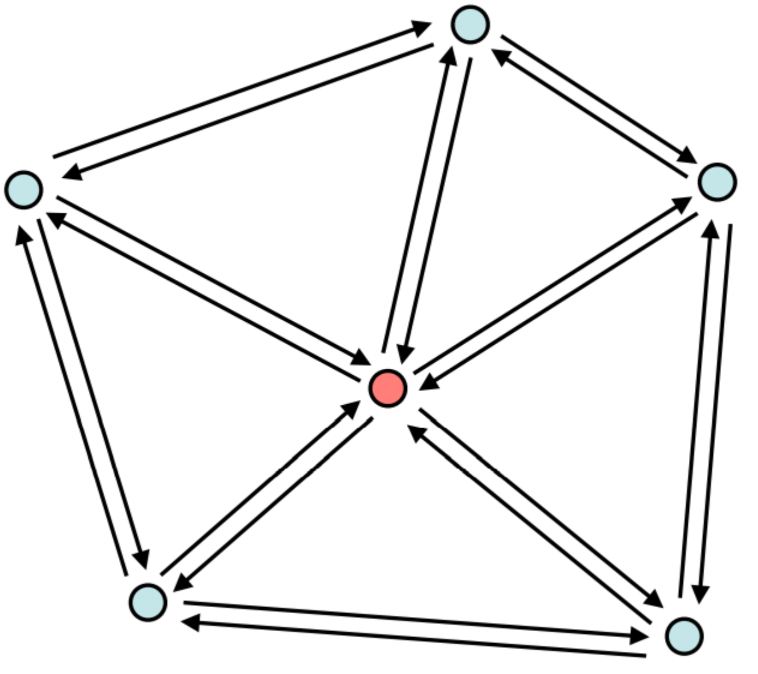


96 to 144 B/v

no case distinctions during traversal

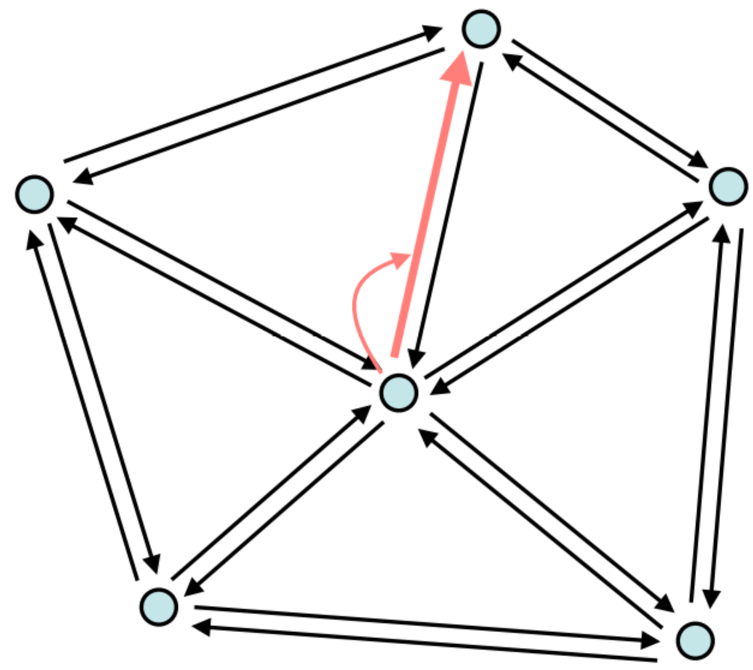
One-Ring Traversal

- Start at vertex



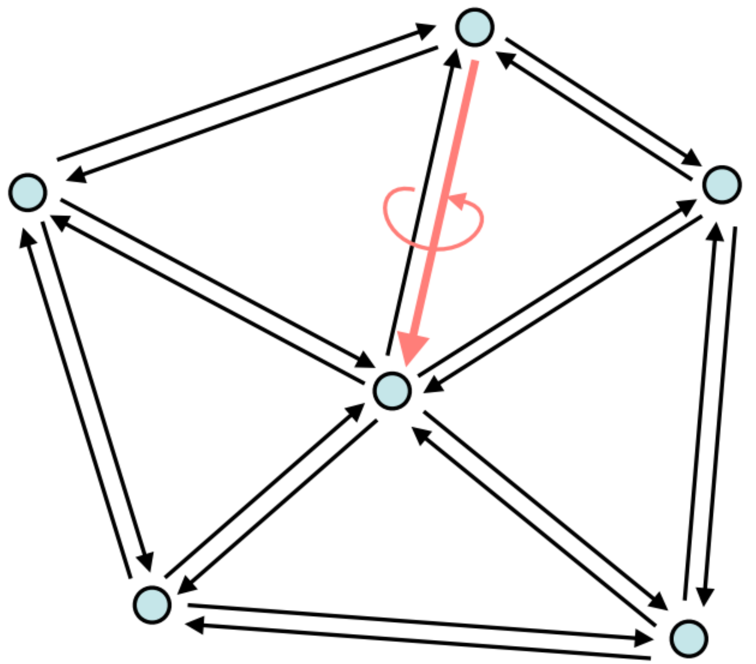
One-Ring Traversal

- Start at vertex
- Outgoing halfedge



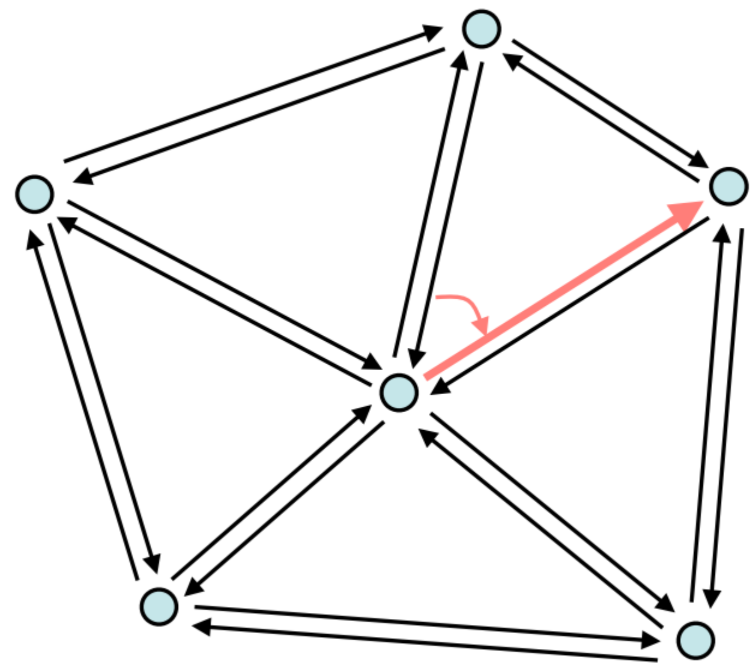
One-Ring Traversal

- Start at vertex
- Outgoing halfedge
- Opposite halfedge



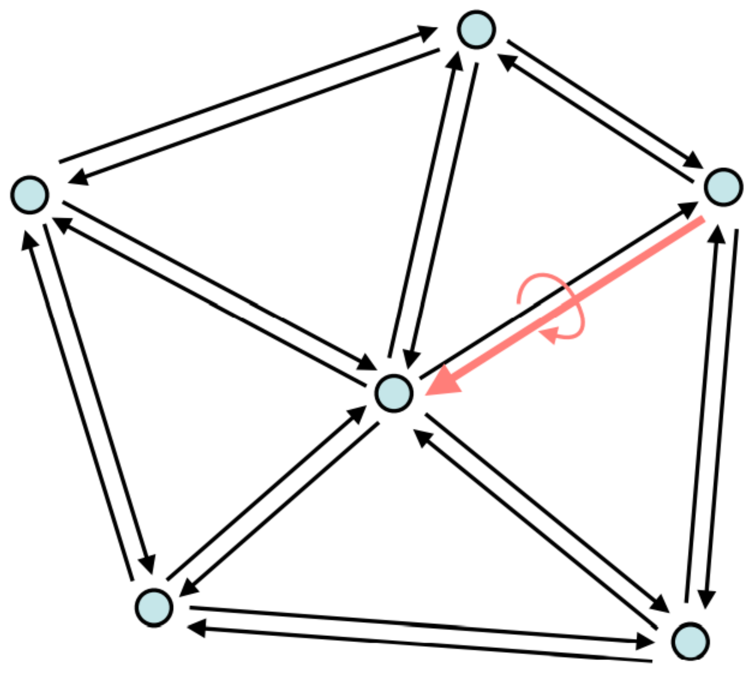
One-Ring Traversal

- Start at vertex
- Outgoing halfedge
- Opposite halfedge
- Next halfedge



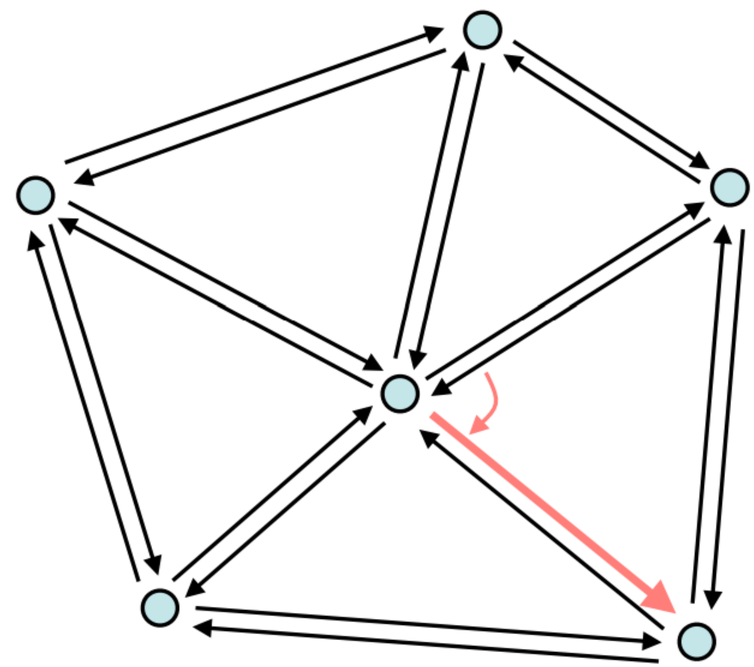
One-Ring Traversal

- Start at vertex
- Outgoing halfedge
- Opposite halfedge
- Next halfedge
- Opposite



One-Ring Traversal

- Start at vertex
- Outgoing halfedge
- Opposite halfedge
- Next halfedge
- Opposite
- Next
- ....



Halfedge-Based Libraries

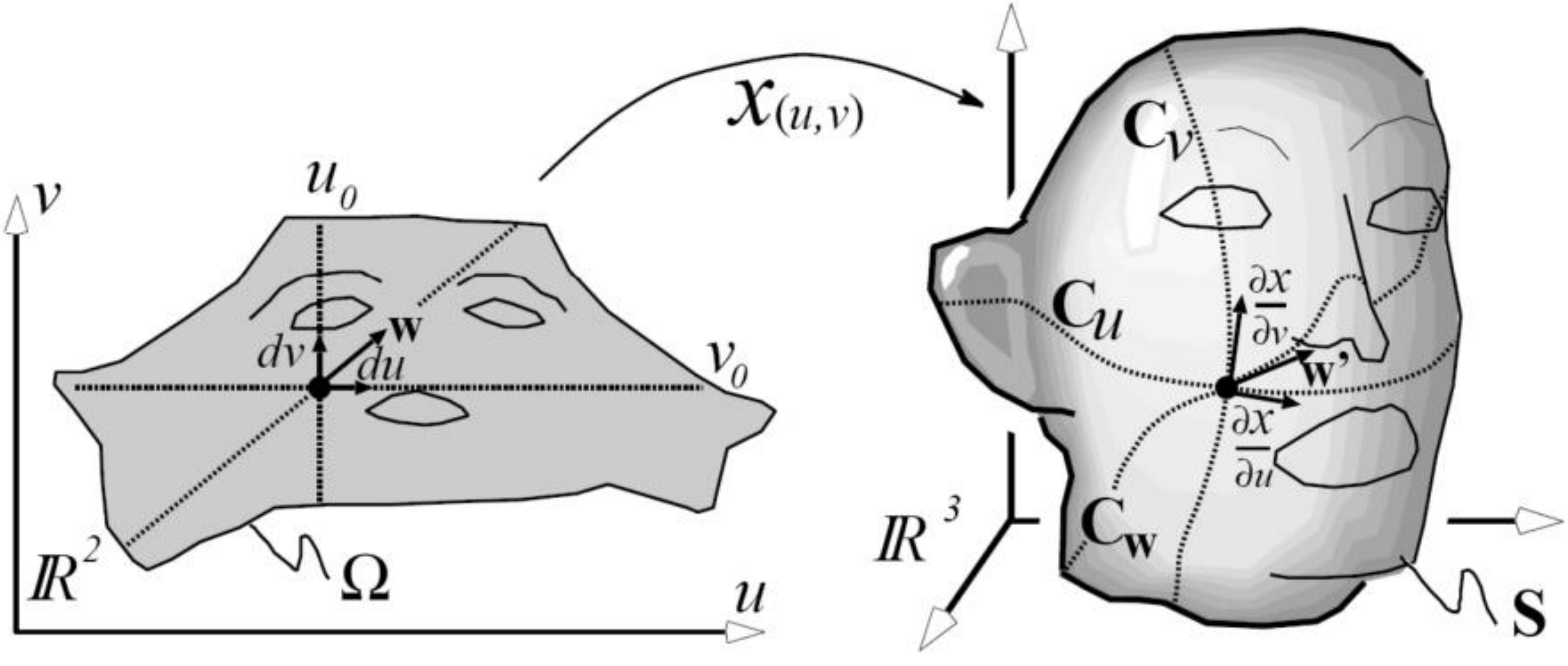
- CGAL
  - www.cgal.org
  - computational geometry
  - free for non-commercial use

- OpenMesh
  - www.openmesh.org
  - Mesh processing
  - free, LGPL license

A Bit Differential Geometry

Differential Geometry: Surfaces

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



Differential Geometry: Surfaces

- Continuous surface

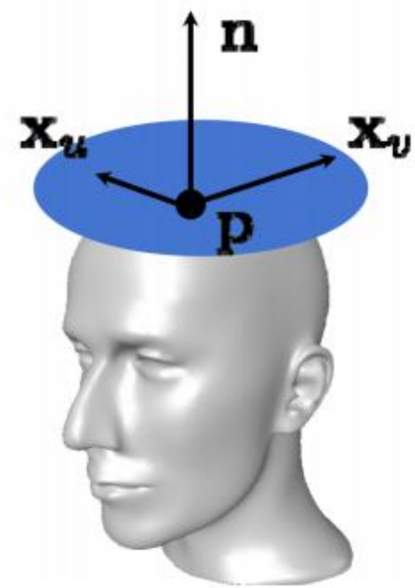
$$\mathbf{x}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}, \ (u,v) \in \mathbb{R}^2$$

- Normal vector

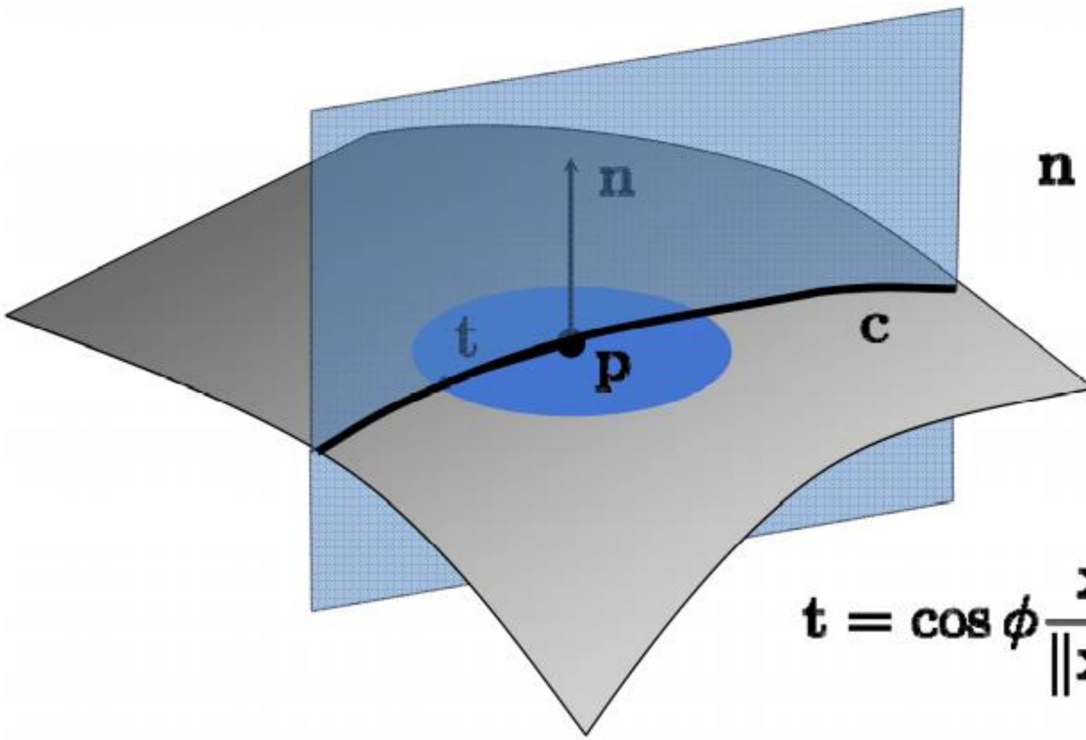
$$\mathbf{n} = (\mathbf{x}_u \times \mathbf{x}_v)/\|\mathbf{x}_u \times \mathbf{x}_v\|$$

  – assuming regular parameterization, i.e.

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



Normal Curvature

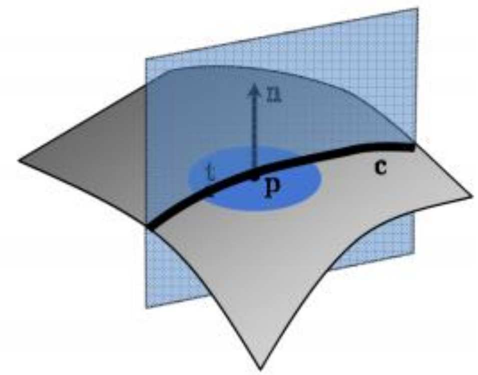


$$\mathbf{n} = \frac{\mathbf{x}_u \times \mathbf{x}_v}{\|\mathbf{x}_u \times \mathbf{x}_v\|}$$

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

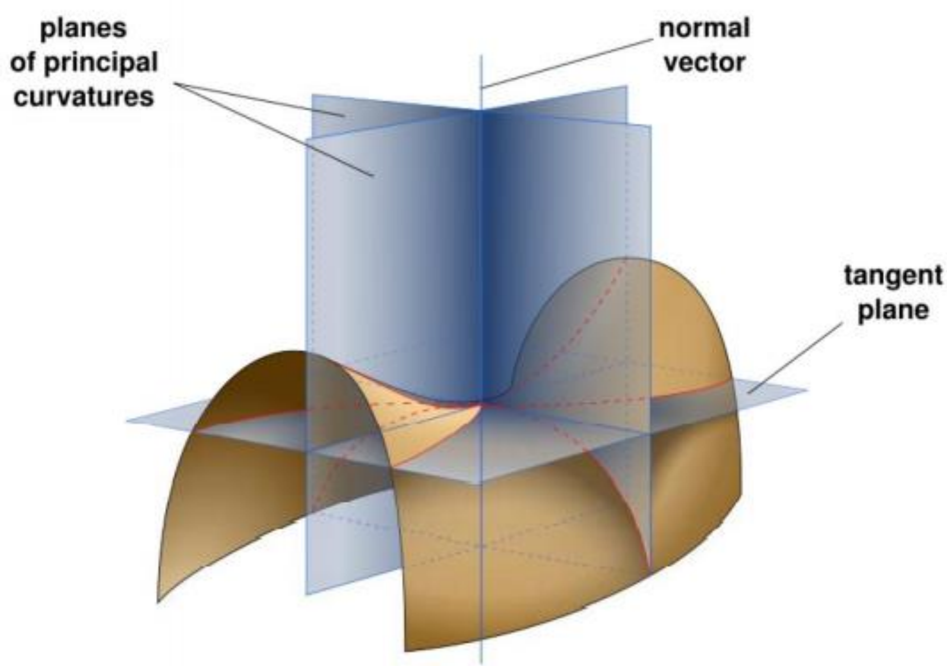
Surface Curvature

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



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

Principal Curvature

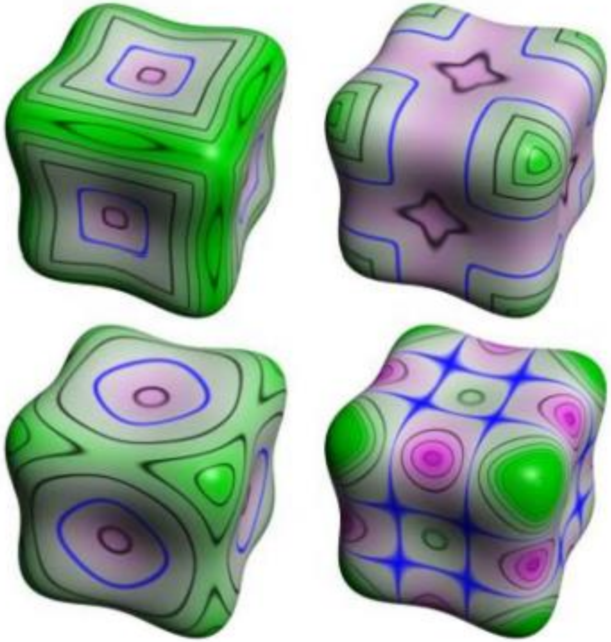


Euler's Theorem: Planes of principal curvature are **orthogonal** and independent of parameterization.

$$\kappa(\theta) = \kappa_1 \cos^2\theta + \kappa_2 \sin^2\theta \qquad \theta = \text{angle with } \kappa_1$$

Curvature

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



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

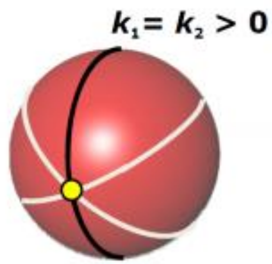
$$H = \frac{1}{2}(\kappa_1 + \kappa_2)$$

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

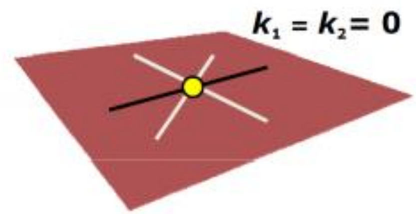
Surface Classification

Isotropic

Equal in all directions



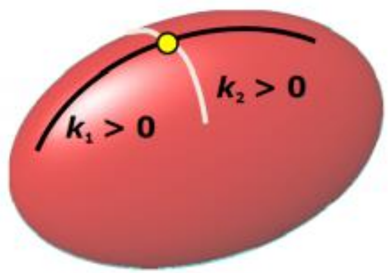
spherical



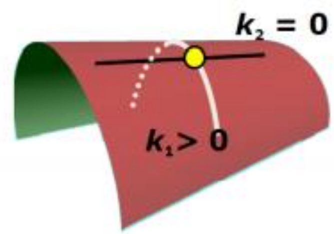
planar

Anisotropic

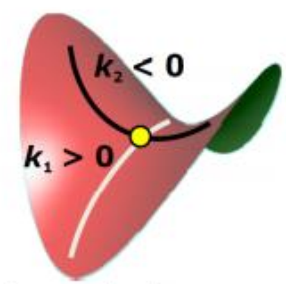
Distinct principal directions



elliptic
$K > 0$

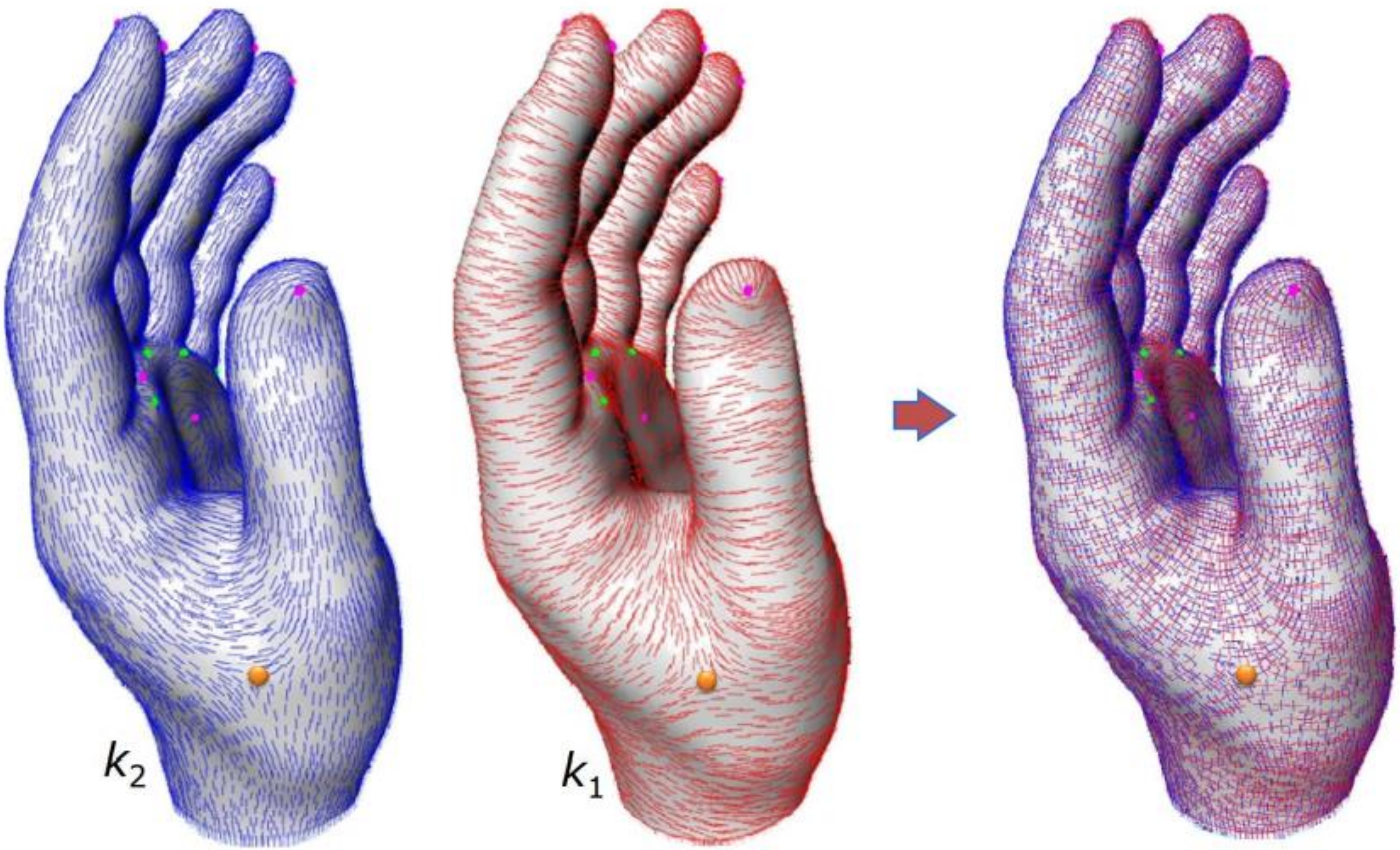


parabolic
$K = 0$
developable



hyperbolic
$K < 0$

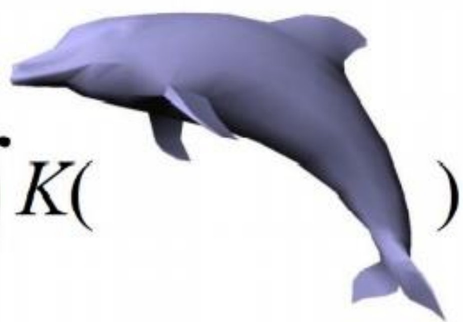
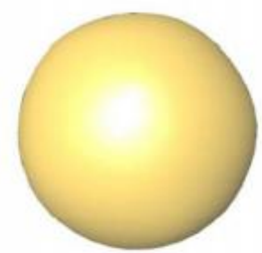
Principal Directions



Gauss-Bonnet Theorem

For ANY closed manifold surface with Euler number χ=2-2g:

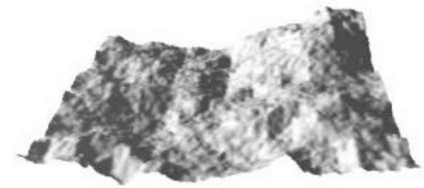
$$\int K = 2\pi\chi$$

$\int K($  $) = \int K($  $) = \int K($  $) = 4\pi$

Smoothing

Laplacian Diffusion

- Diffusion equation



diffusion constant

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

Laplace operator

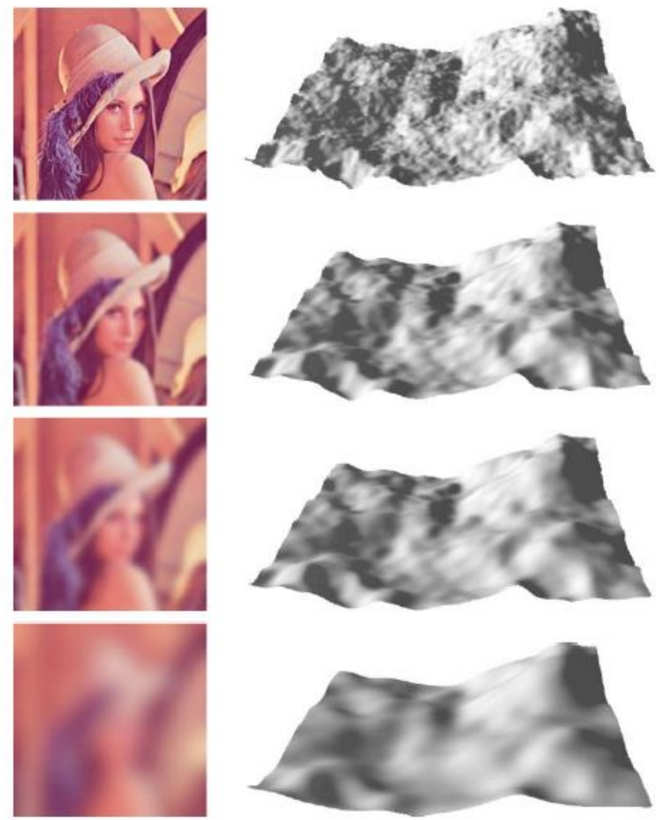
Diffusion

- Diffusion equation

diffusion constant

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

Laplace operator



Laplacian Smoothing

- Discretization of diffusion equation

$$\frac{\partial}{\partial t}\mathbf{p}_i = \mu \Delta \mathbf{p}_i$$

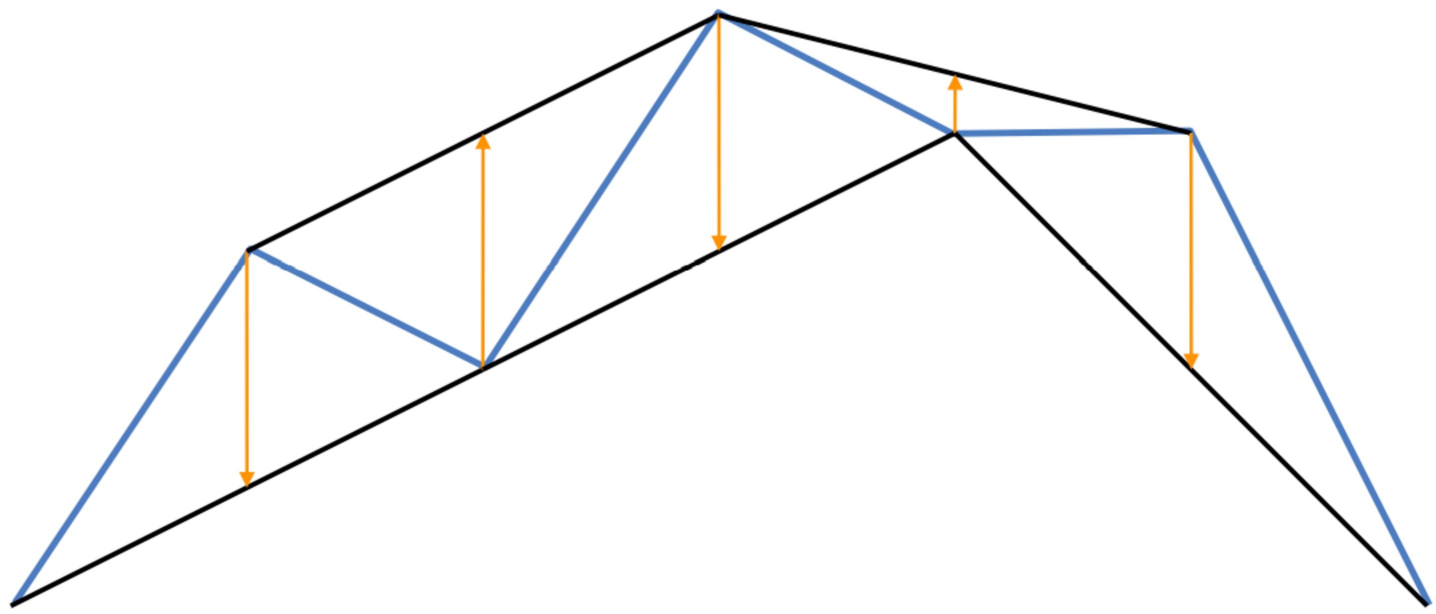
- Leads to simple update rule
  - iterate

$$\mathbf{p}_i' = \mathbf{p}_i + \mu \, dt \, \Delta \mathbf{p}_i$$

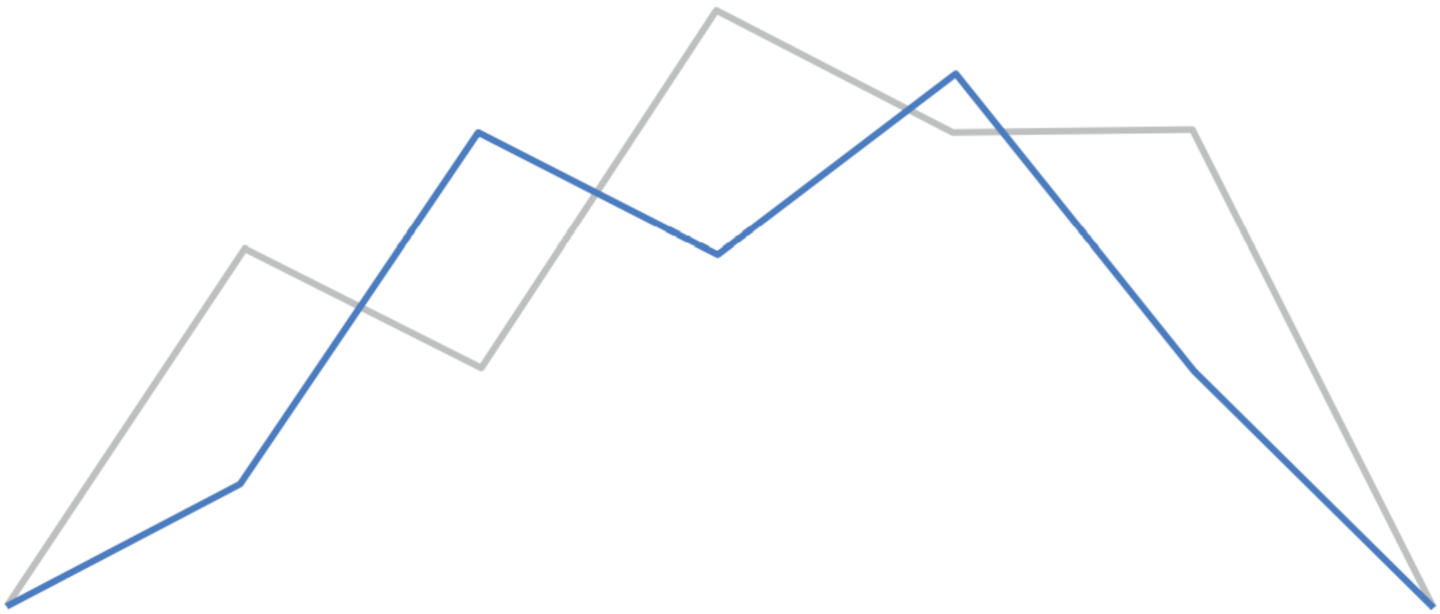
explicit Euler integration

  - until convergence

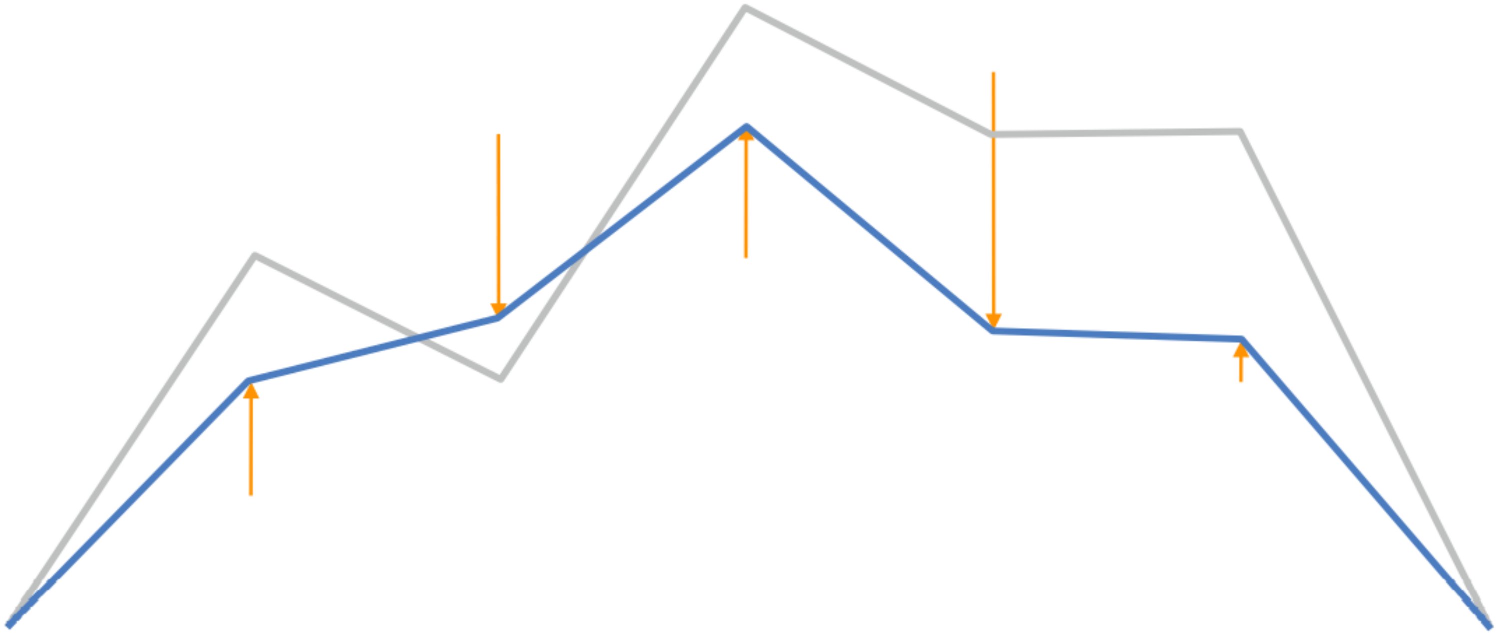
A Simple Example



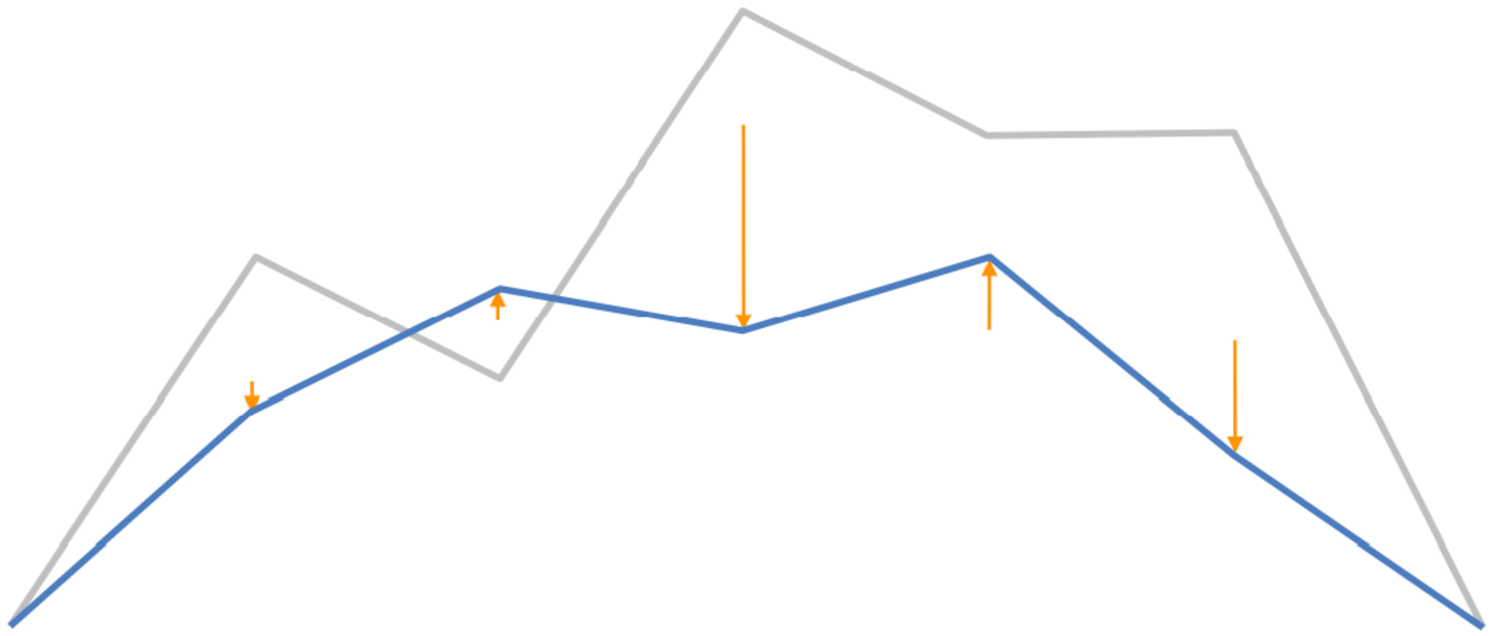
A Simple Example



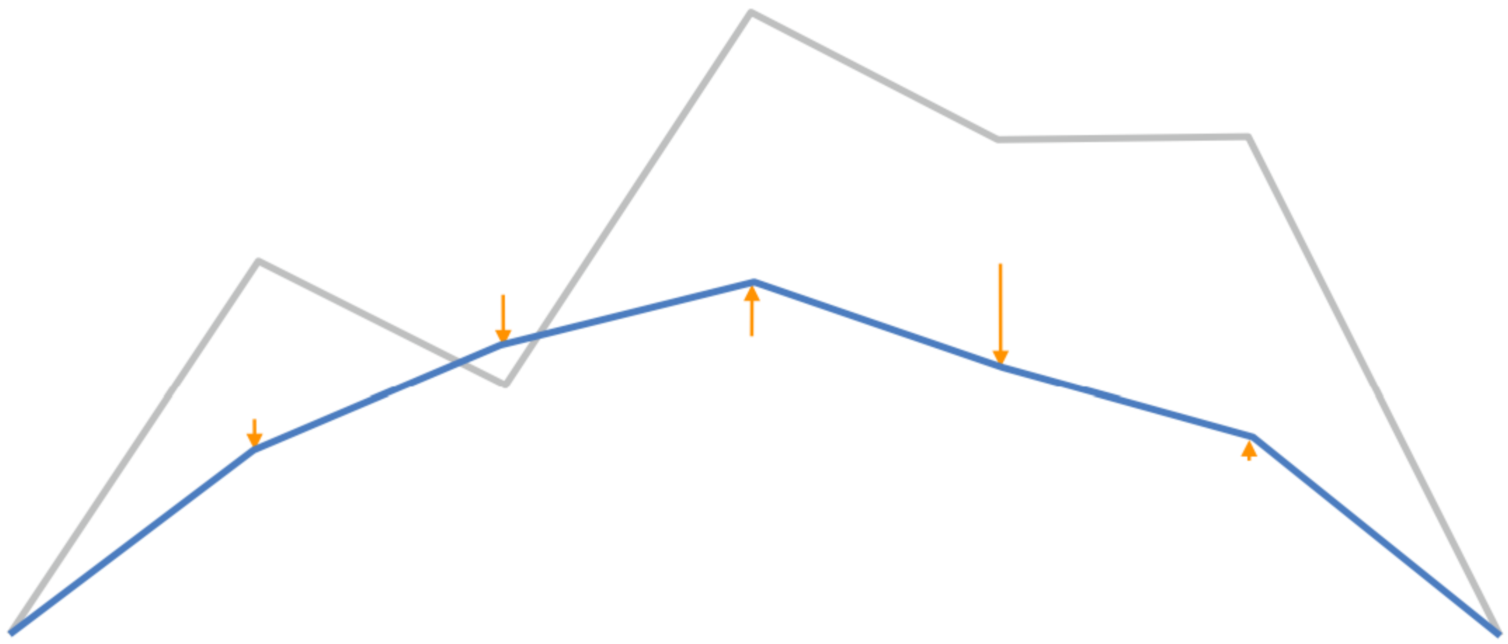
A Simple Example



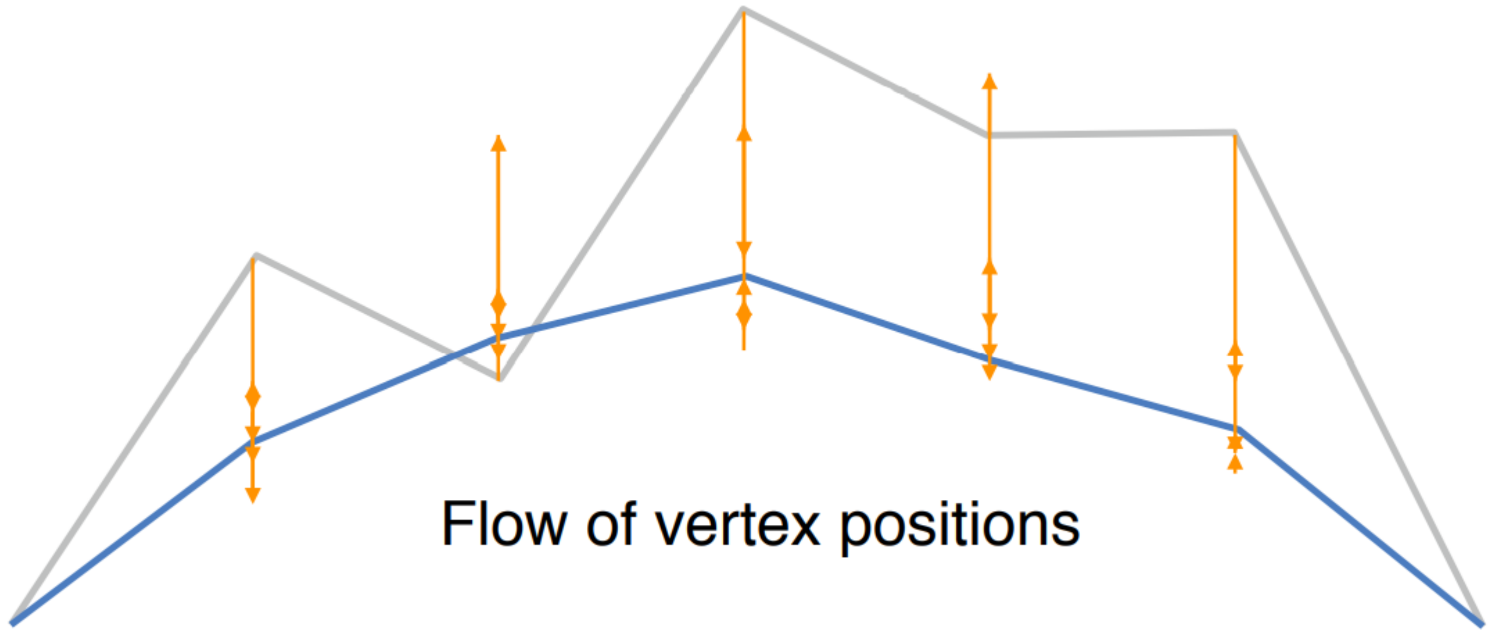
A Simple Example



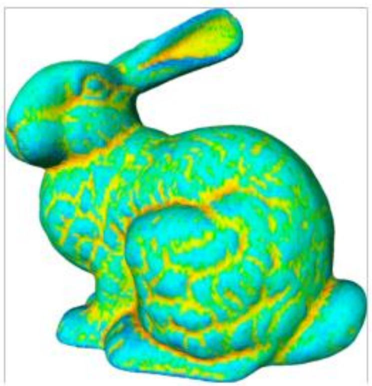
A Simple Example



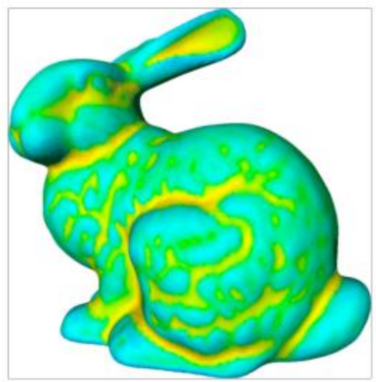
A Simple Example



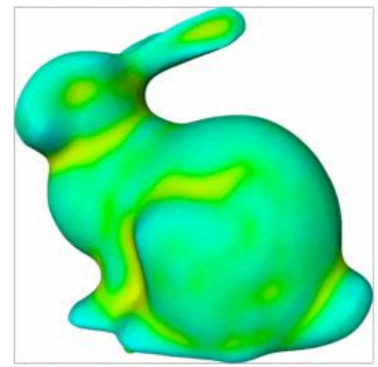
Laplacian Smoothing



0 Iterations



5 Iterations



20 Iterations

Curvature Flow

- Curvature is independent of parameterization
- Flow equation

surface normal

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

mean curvature

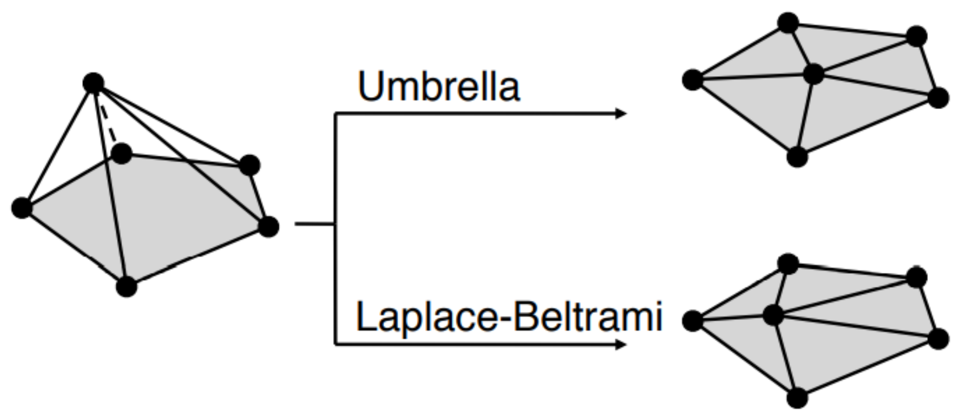
- We have

$$\Delta_S \mathbf{p} = -2H\mathbf{n}$$

Laplace-Beltrami operator

Curvature Flow

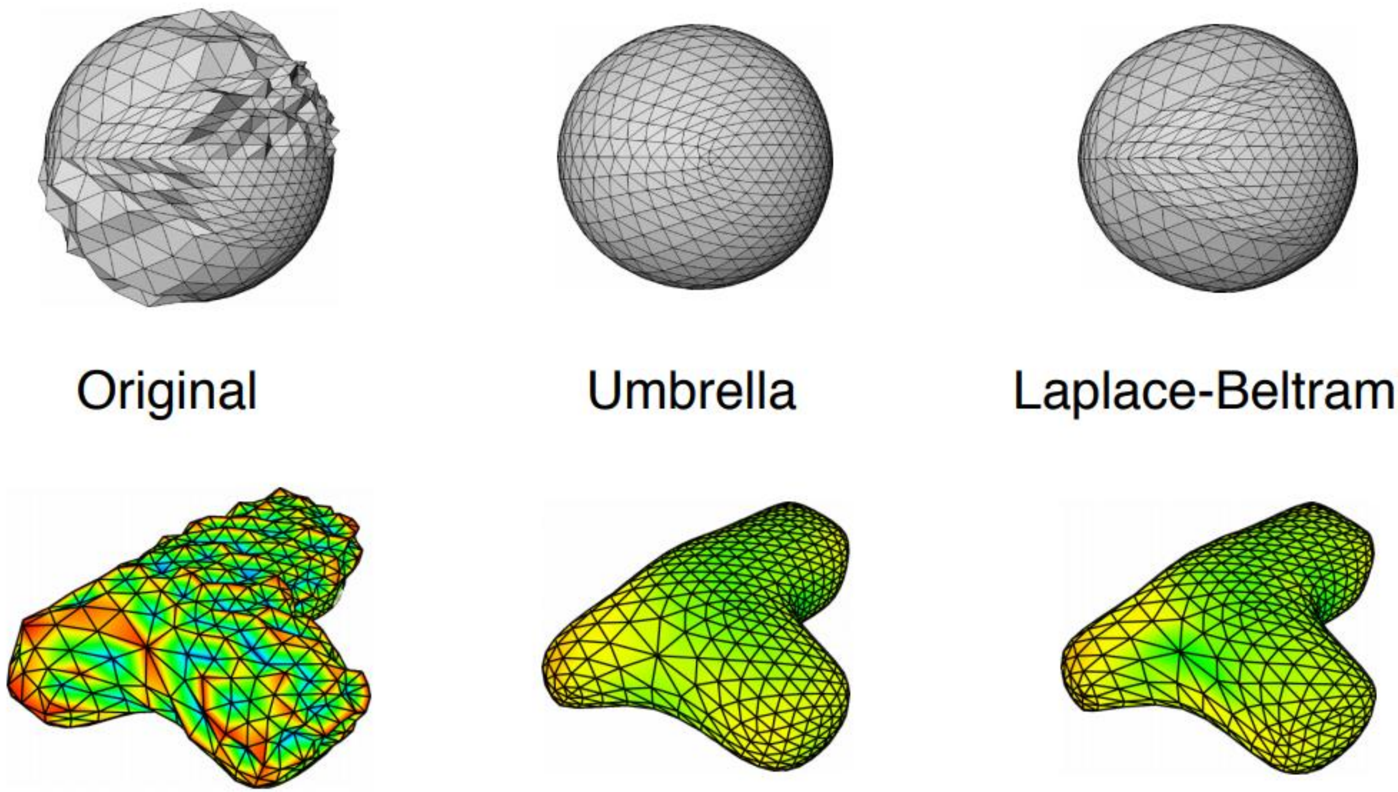
- Mean curvature flow $$\frac{\partial}{\partial t}\mathbf{p} = \mu \Delta_S \mathbf{p}$$
  - use discrete Laplace-Beltrami operator *(cot weights)*
- Compare to uniform discretization of Laplacian



tangential drift

vertices move only along normal

Comparison



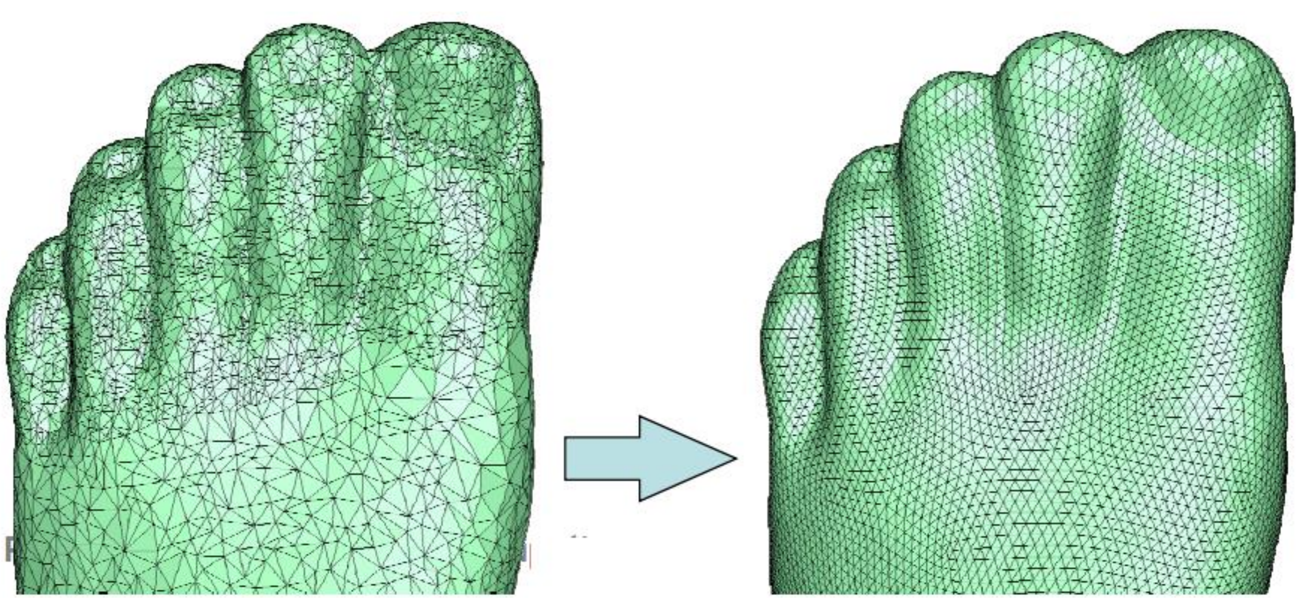
Other approaches

- Smoothing
  - Membrane energy
  - Thin-plate energy
- Alternative approaches
  - Anisotropic diffusion
  - Normal filtering
  - Non-linear PDEs
  - Bilateral filtering

Remeshing

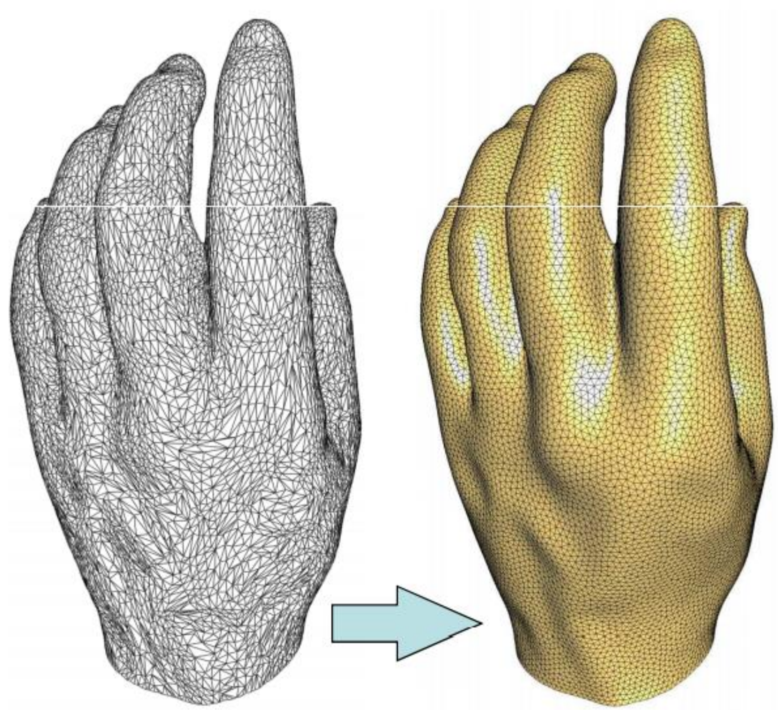
Remeshing

- Altering the mesh geometry and connectivity to improve quality
- Replacing an arbitrarily structured mesh by a structured one



Isotropic Remeshing

- Well-shaped elements
  - for processing & simulation (numerical stability & efficiency)



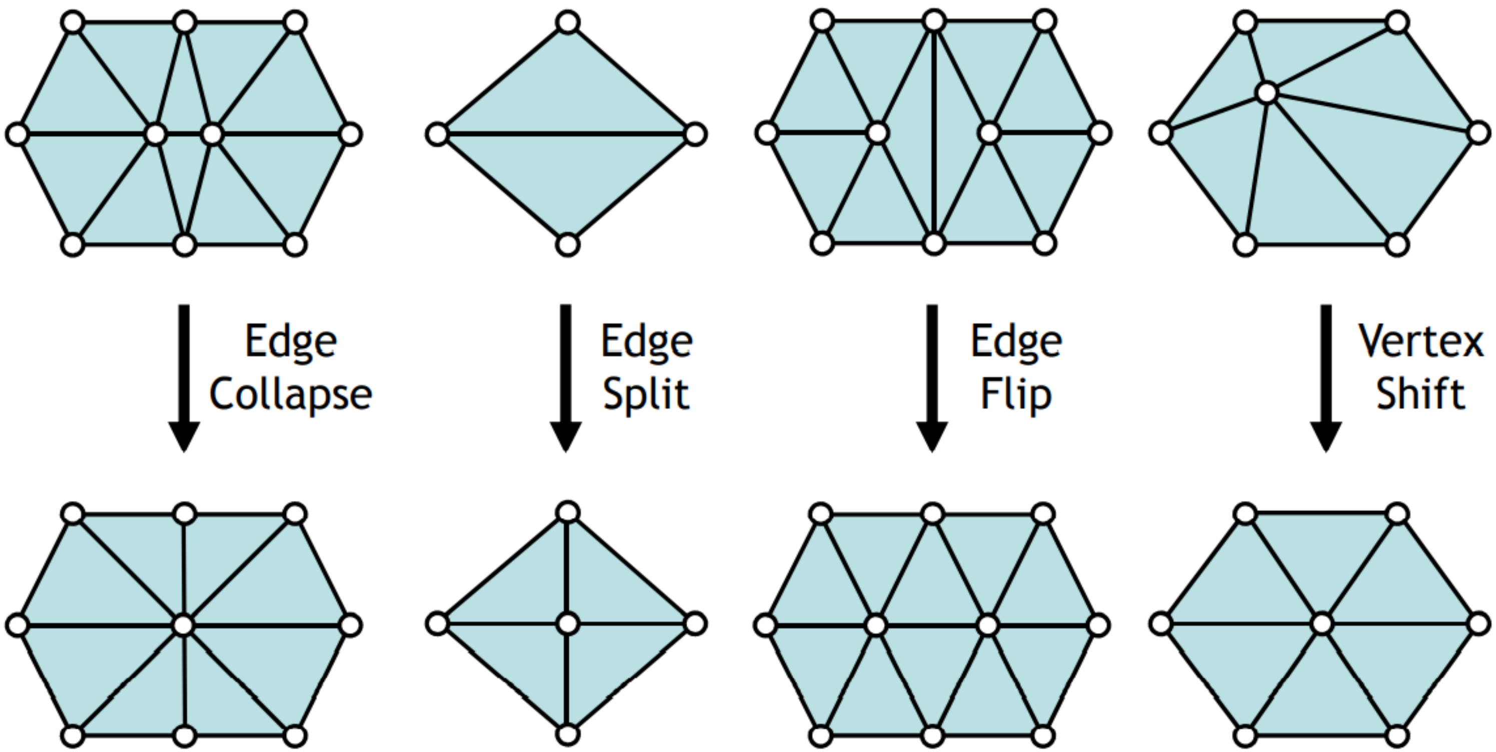
Iterative Mesh Optimization

- (Area weighted) random scatter (area weighted) random scatter or simply start with the given mesh
- Improve vertex distribution
- Update mesh connectivity

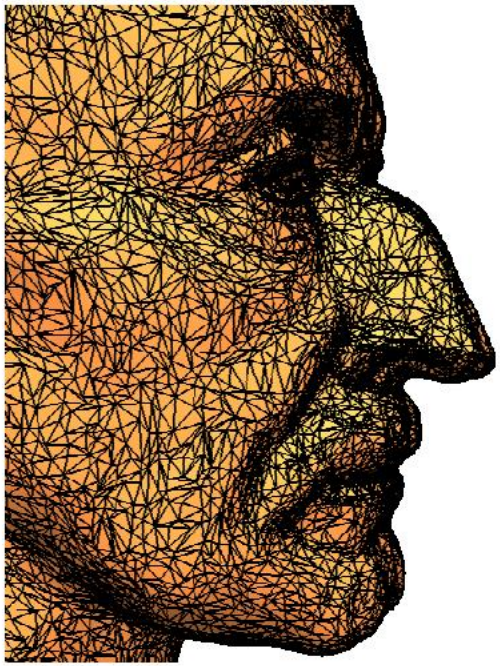
Isotropic remeshing prefers.

- Locally uniform 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 Laplacian operator

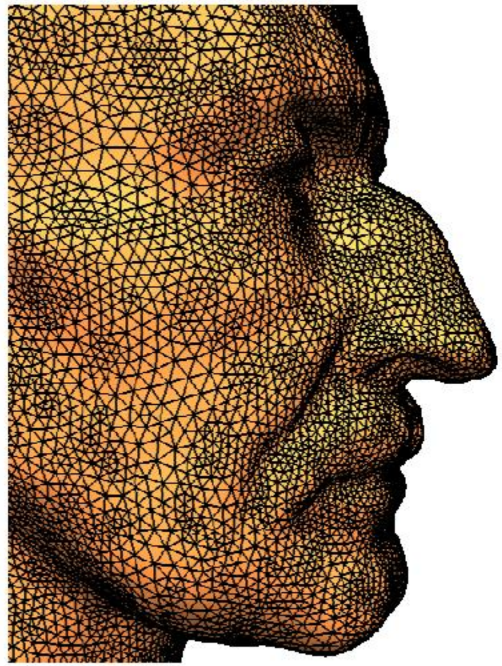
Local Remeshing Operators



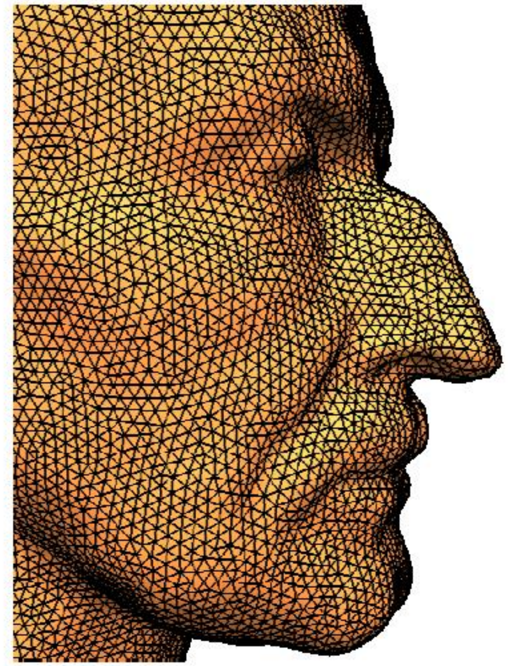
Thresholds L_{min} and L_{max}



Original



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



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

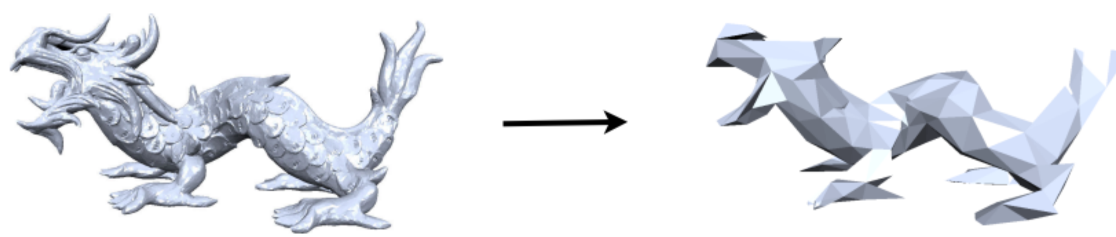
Mesh Simplification & Approximation

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

hard!

→ look for sub-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

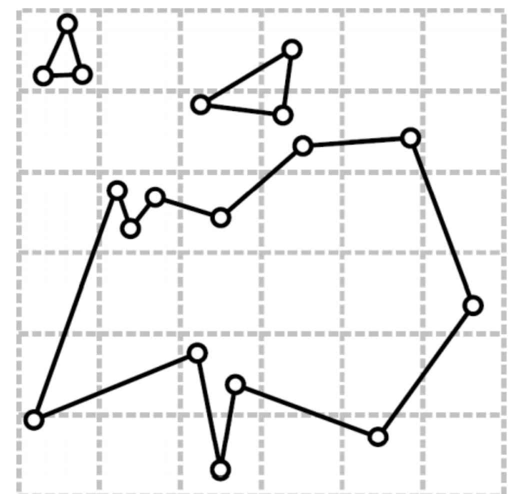
- Respect additional fairness criteria
  - normal deviation, triangle shape, scalar attributes, etc.

Vertex clustering

- Cluster Generation
- Computing a representative
- Mesh generation
- Topology changes

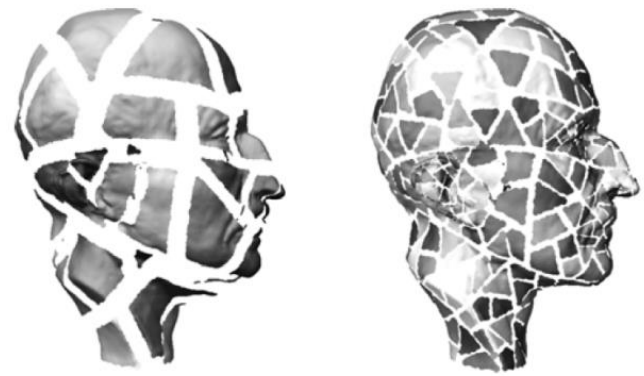
Vertex Clustering

- Cluster Generation
  - Uniform 3D grid
  - Map vertices to cluster cells
- Computing a representative
- Mesh generation
- Topology changes



Vertex Clustering

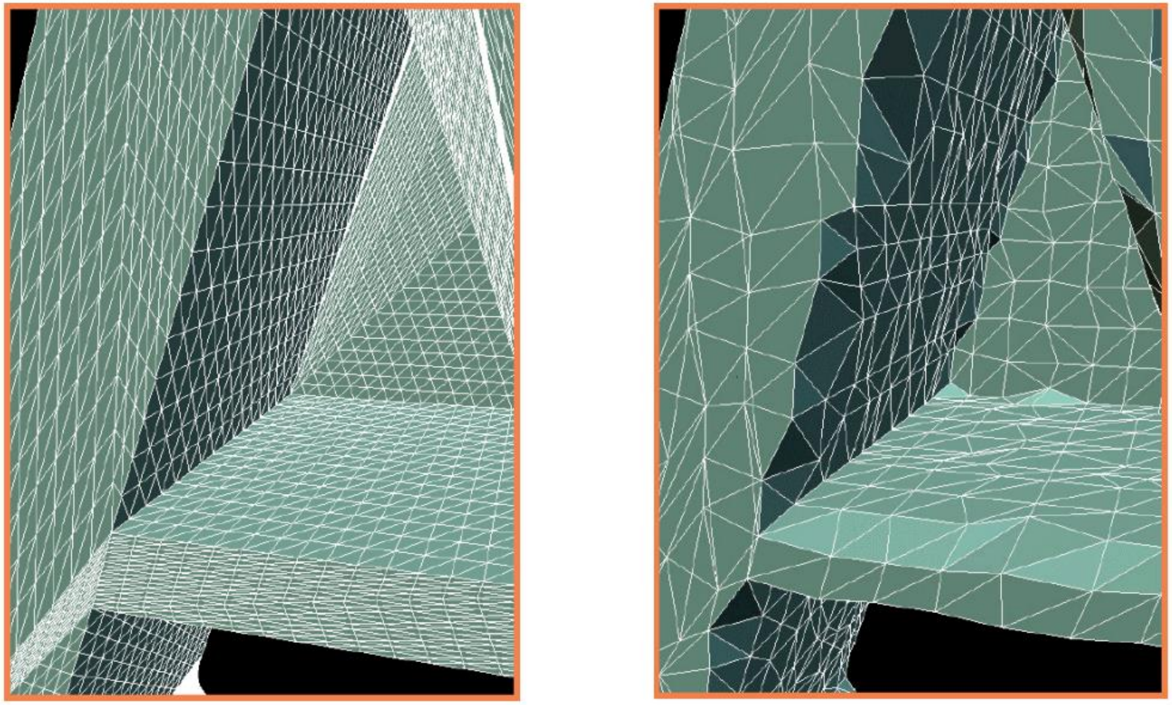
- Cluster Generation
  - Hierarchical approach
  - Top-down or bottom-up
- Computing a representative
- Mesh generation
- Topology changes



Vertex Clustering

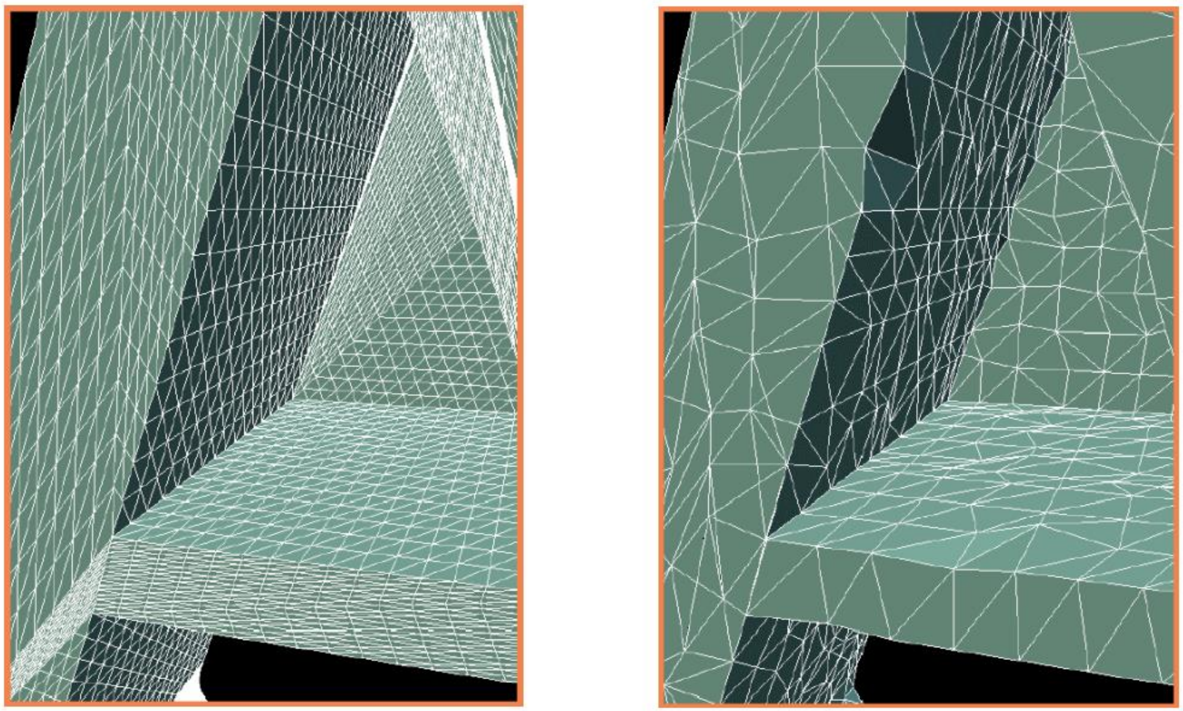
- Cluster Generation
- Computing a representative
  - Average/median vertex position
  - Error quadrics
- Mesh generation
- Topology changes

Computing a Representative



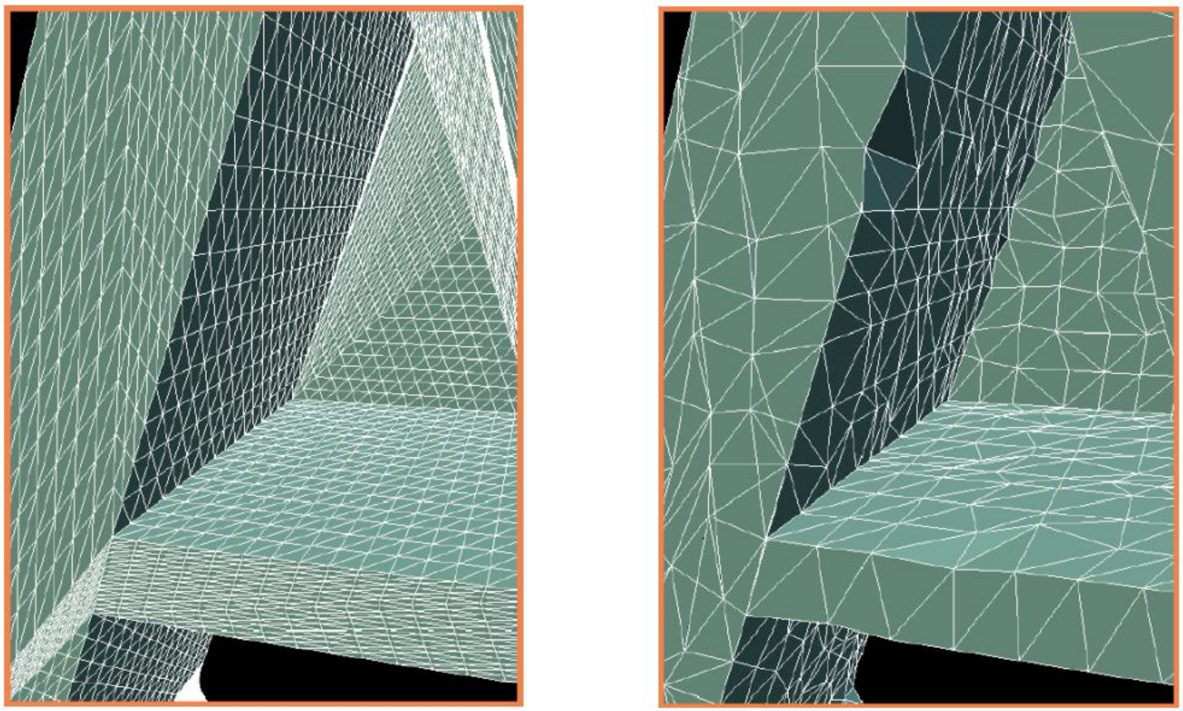
Average vertex position → Low-pass filter

Computing a Representative



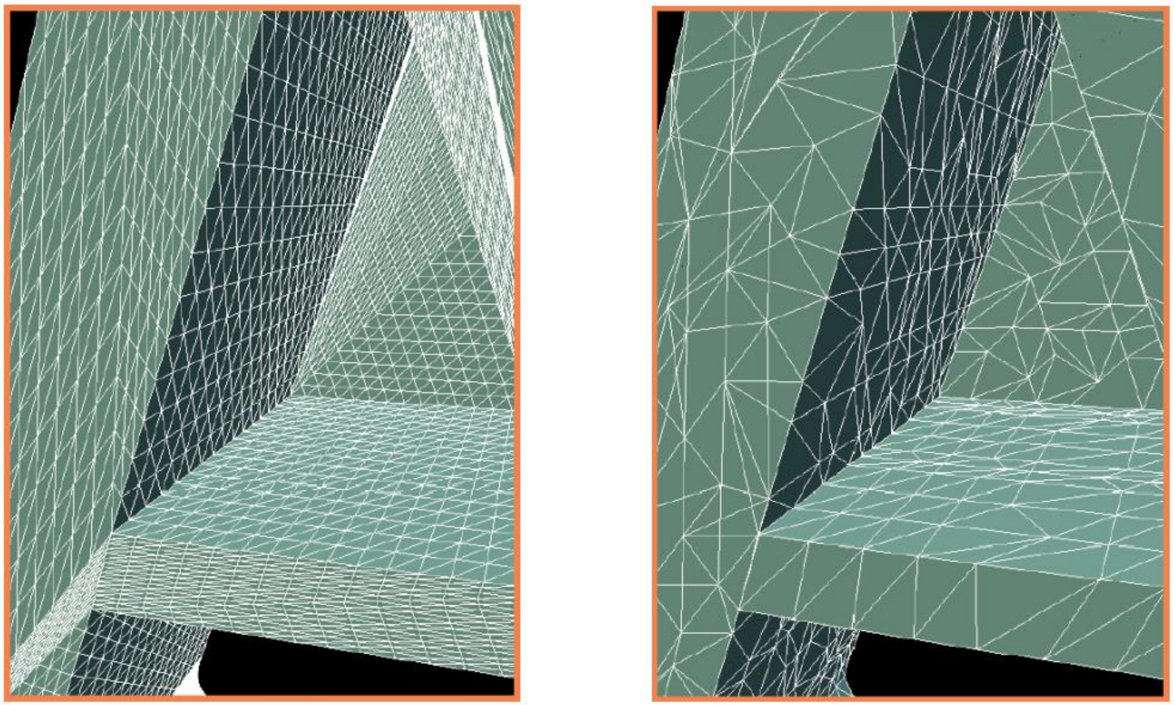
Median vertex position → Sub-sampling

Computing a Representative



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

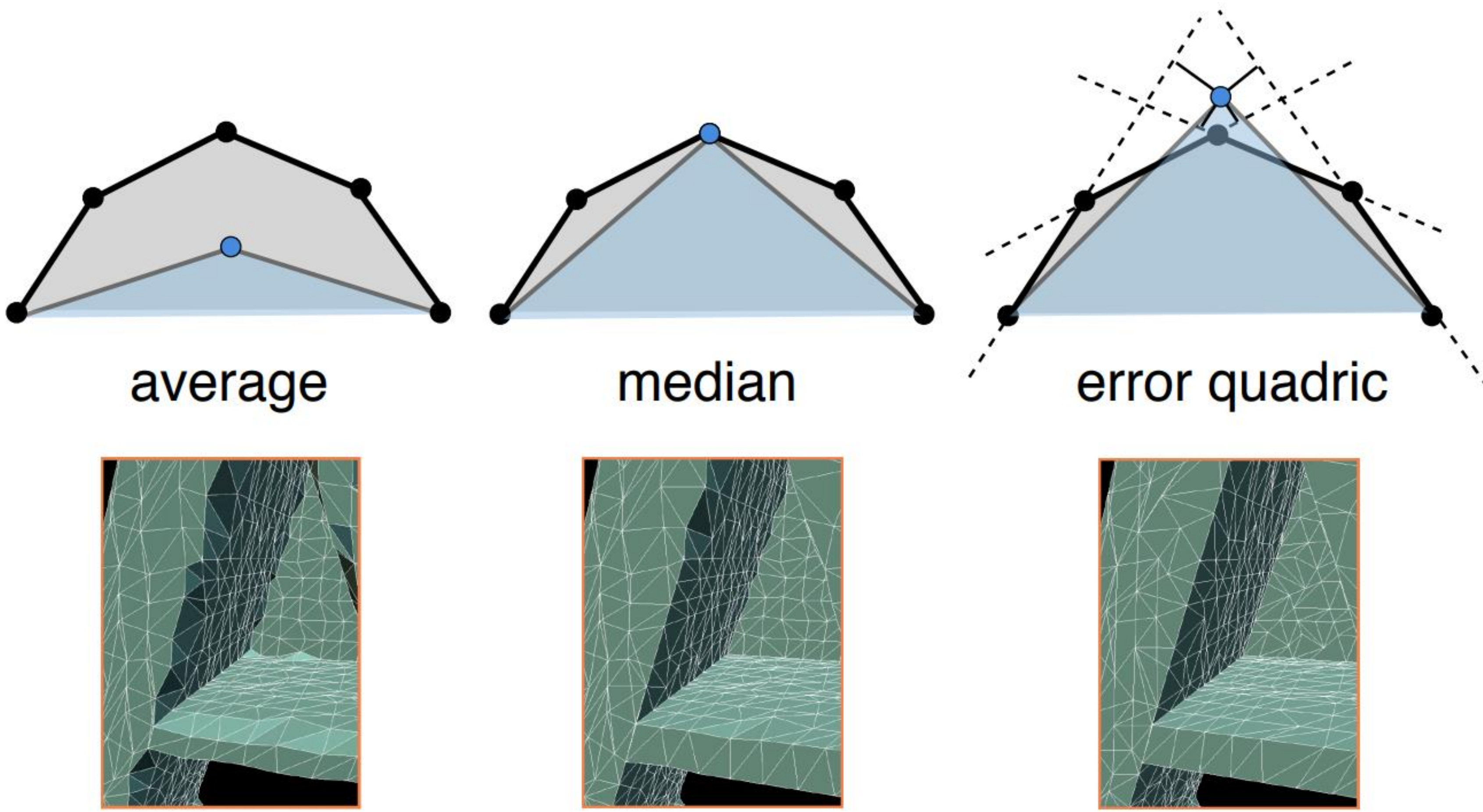
- Sum distances to vertex' planes

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

- Point that minimizes the error

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} p^* = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Comparison

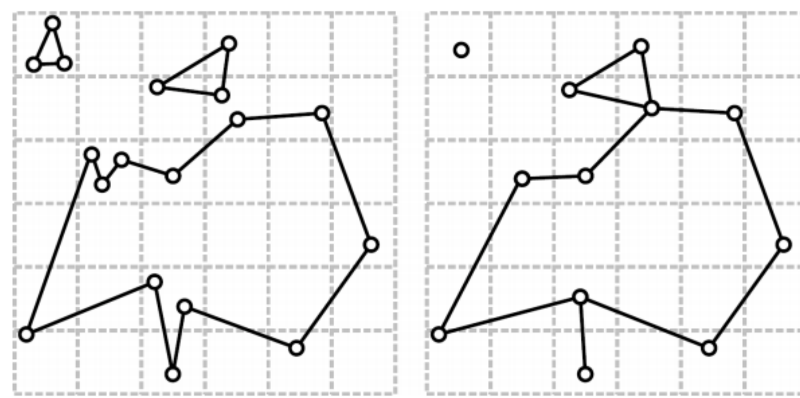


Vertex Clustering

- 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
  - If different sheets pass through one cell
  - Not manifold



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
```

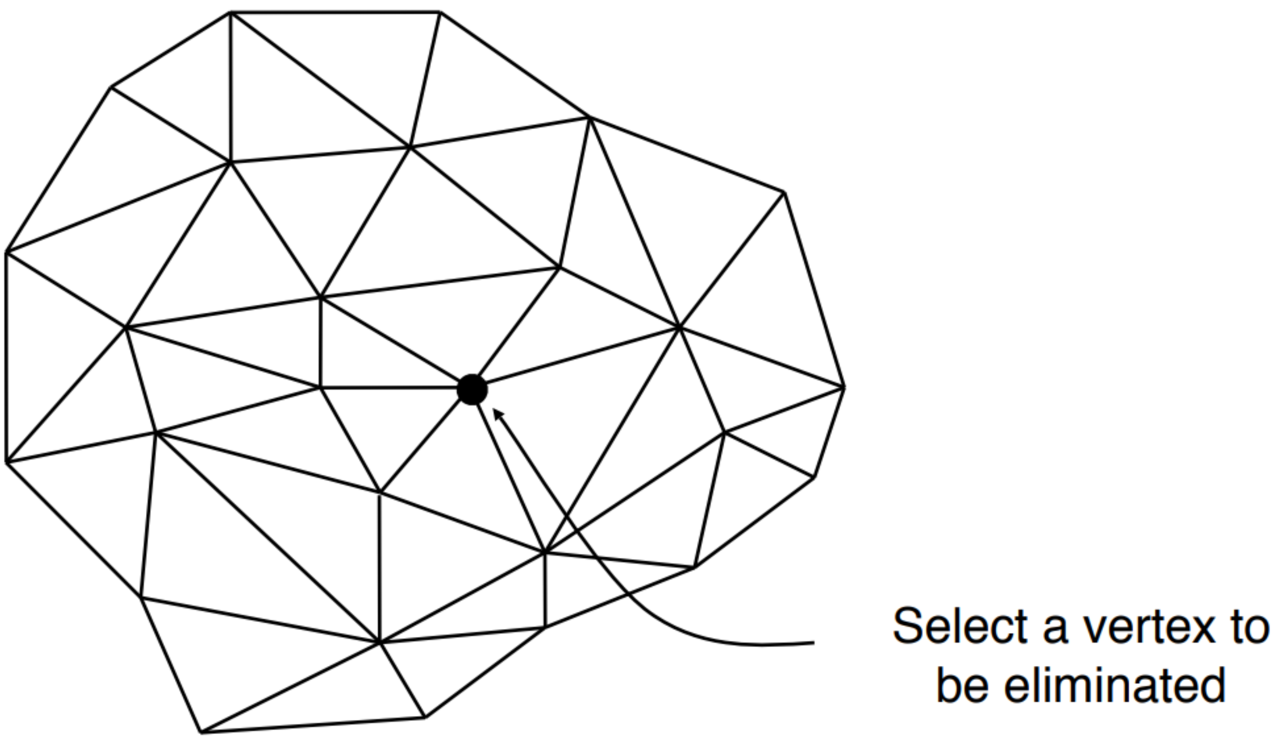
Incremental Decimation

- General Setup
- Decimation operators
- Error metrics
- Fairness criteria
- Topology changes

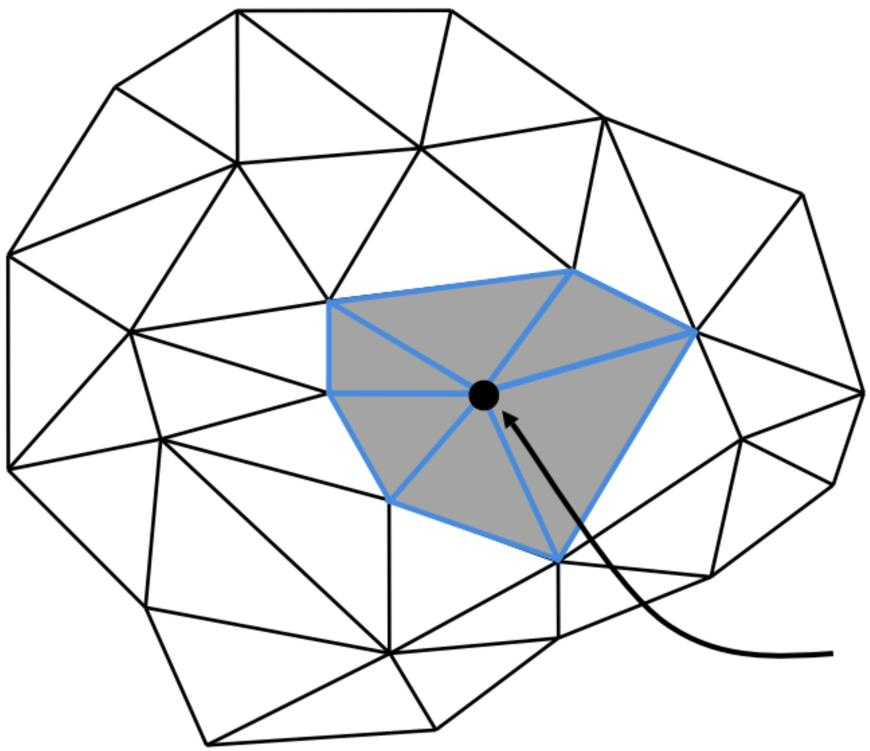
Decimation Operators

- 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

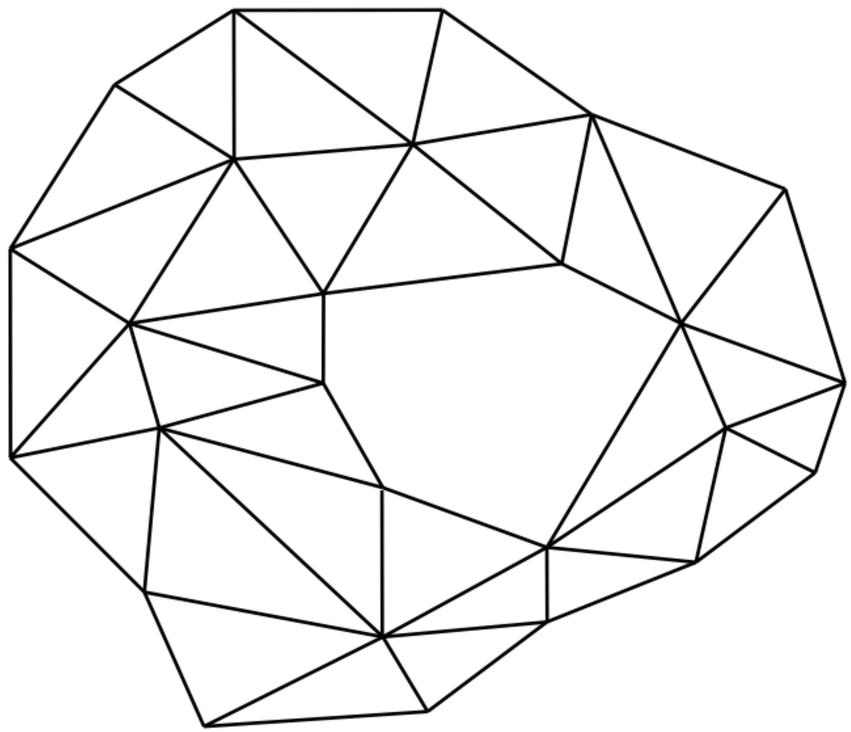


Vertex Removal



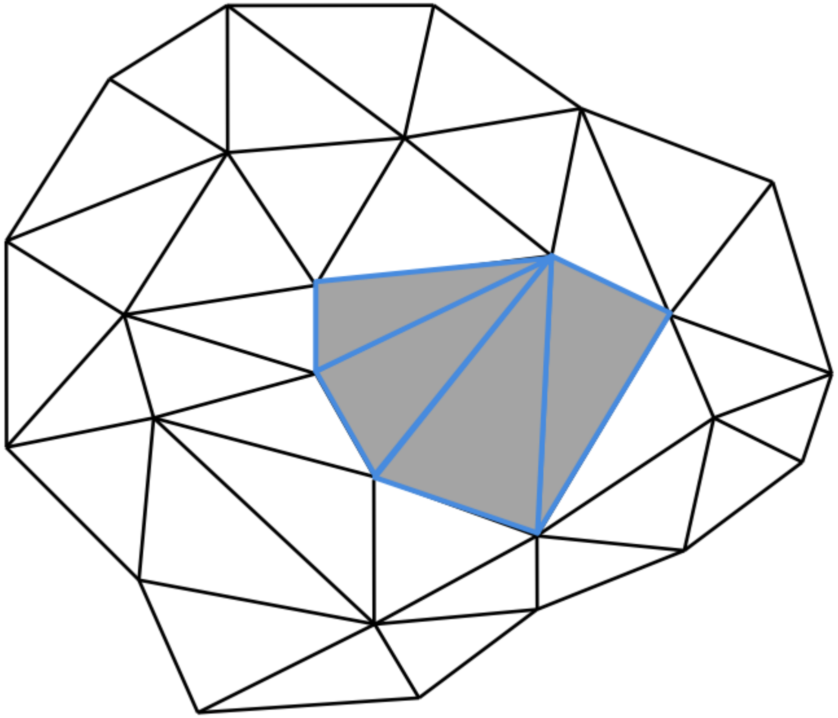
Select all triangles
sharing this vertex

Vertex Removal



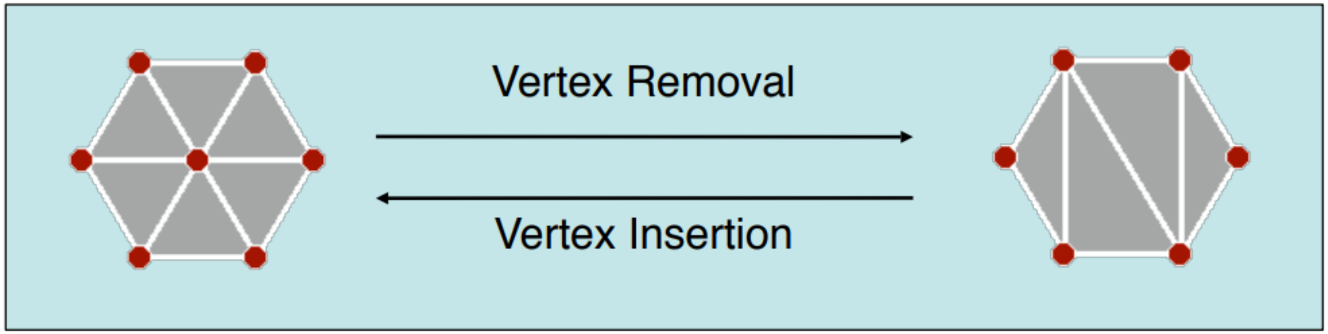
Remove the
selected triangles,
creating the hole

Vertex Removal



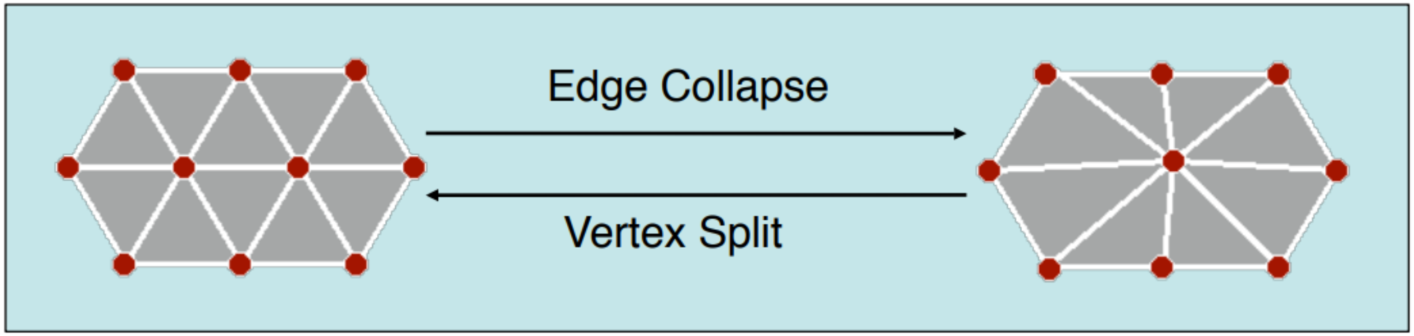
Fill the hole
with triangles

Decimation Operators



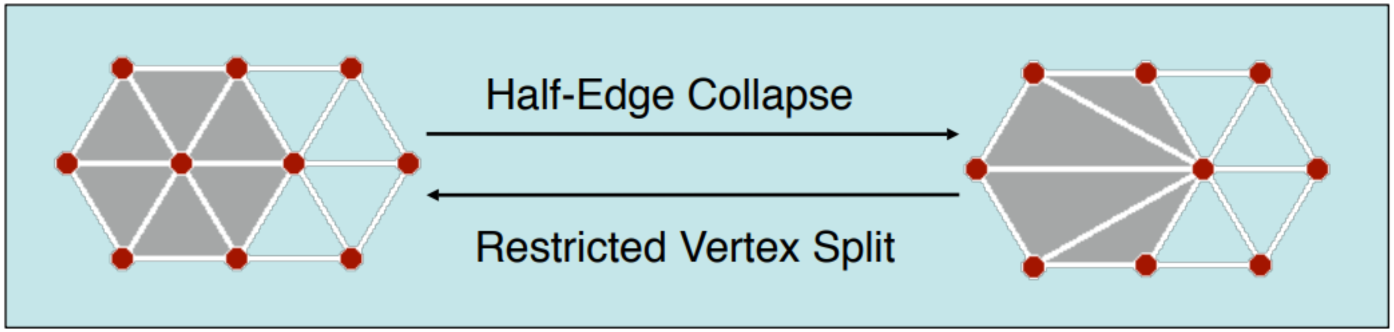
- Remove vertex
- Re-triangulate hole
  - Combinatorial DOFs
  - Sub-sampling

Decimation Operators



- Merge two adjacent triangles
- Define new vertex position
  - Continuous DOF
  - Filtering

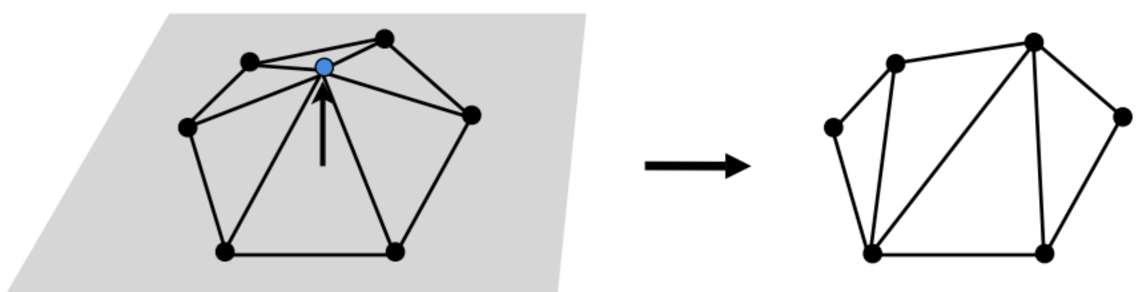
Decimation Operators



- Collapse edge into one end point
  - Special vertex removal
  - Special edge collapse
- No DOFs
  - One operator per half-edge
  - Sub-sampling!

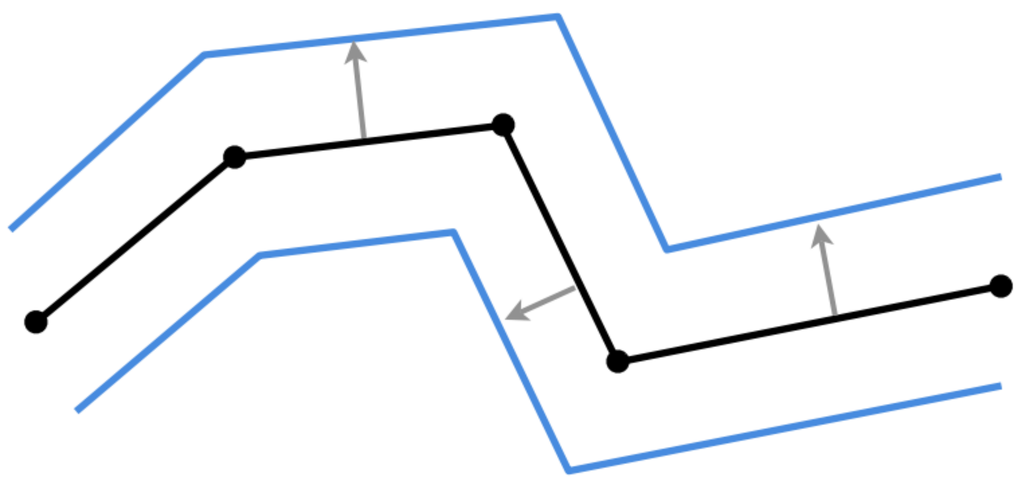
Local Error Metrics

- Local distance to mesh [Schroeder et al. 92]
  - Compute average plane
  - No comparison to *original* geometry



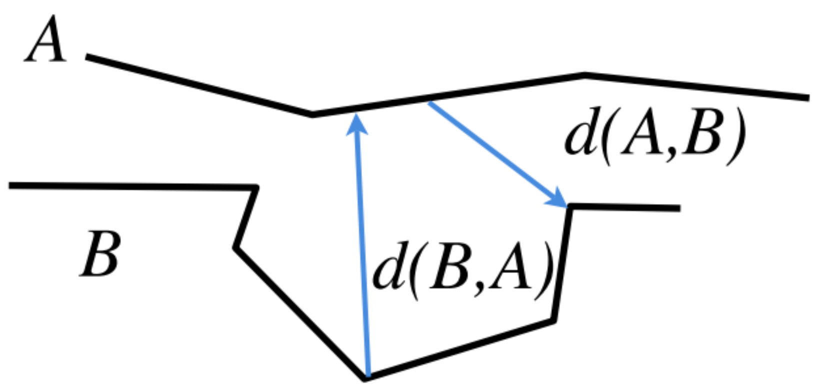
Global Error Metrics

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



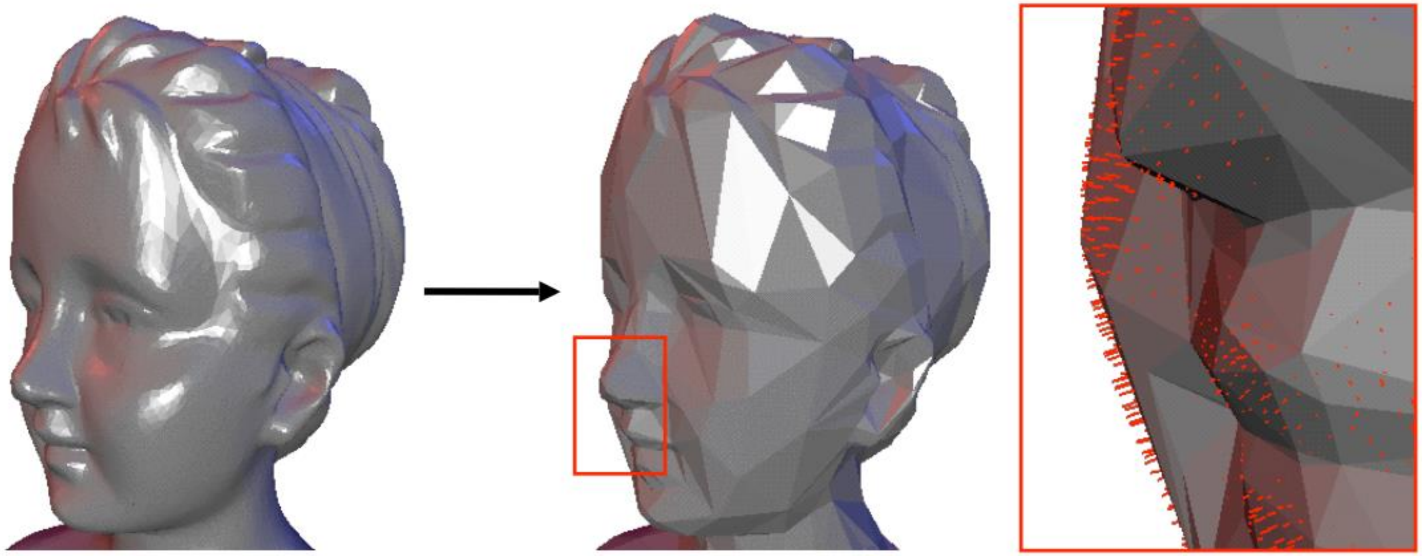
Global Error Metrics

- (Two-sided) Hausdorff distance: Maximum distance between two shapes
  - In general $d(A,B) \neq d(B,A)$
  - Computationally involved



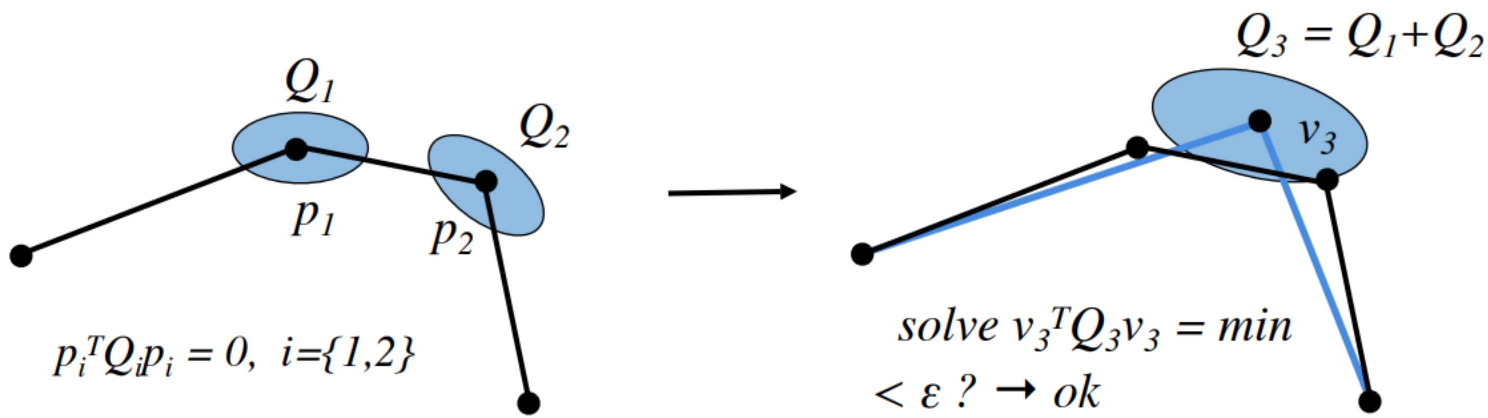
Global Error Metrics

- Scan data: One-sided Hausdorff distance sufficient
  - From original vertices to current surface



Global Error Metrics

- Error quadrics [Garland, Heckbert 97]
  - Squared distance to planes at vertex
  - No bound on true error

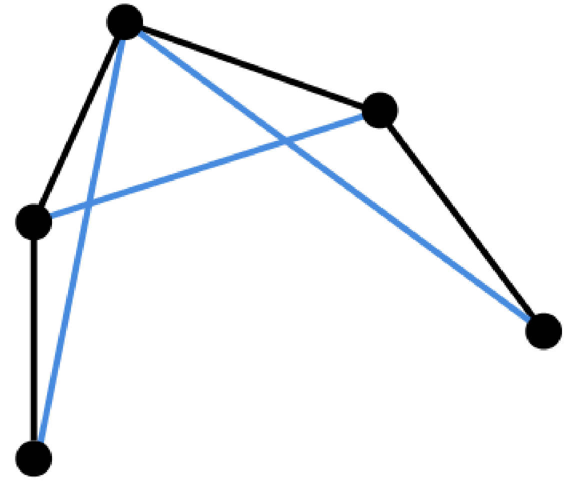


Complexity

- N = number of vertices
- Priority queue for half-edges
  - 6 N * log (6 N)
- Error control
  - Local O(1) -> global O(N)
  - Local O(N) -> global O(N^2)

Fairness Criteria

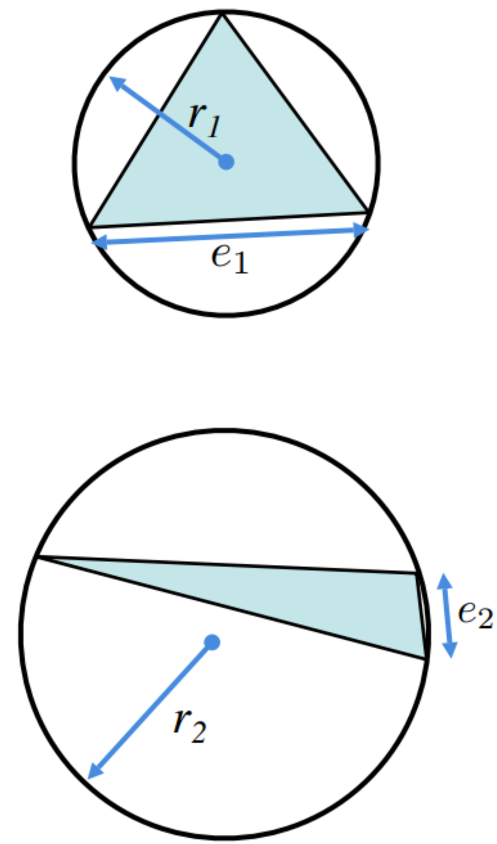
- Rate quality of decimation operation
  - Approximation error
  - Triangle shape
  - Dihedral angles
  - Valence balance
  - Color differences



Fairness Criteria

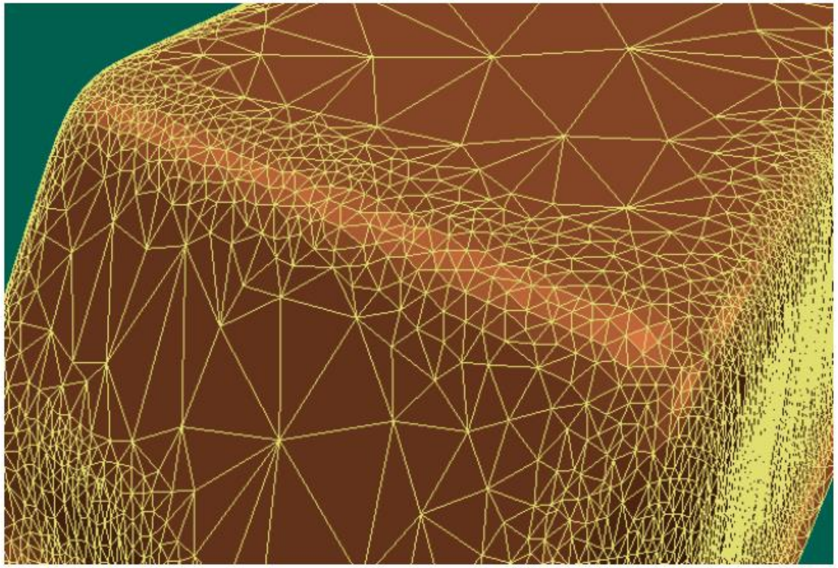
- Rate quality of decimation operation
  - Approximation error
  - Triangle shape
  - Dihedral angles
  - Valence balance
  - Color differences

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



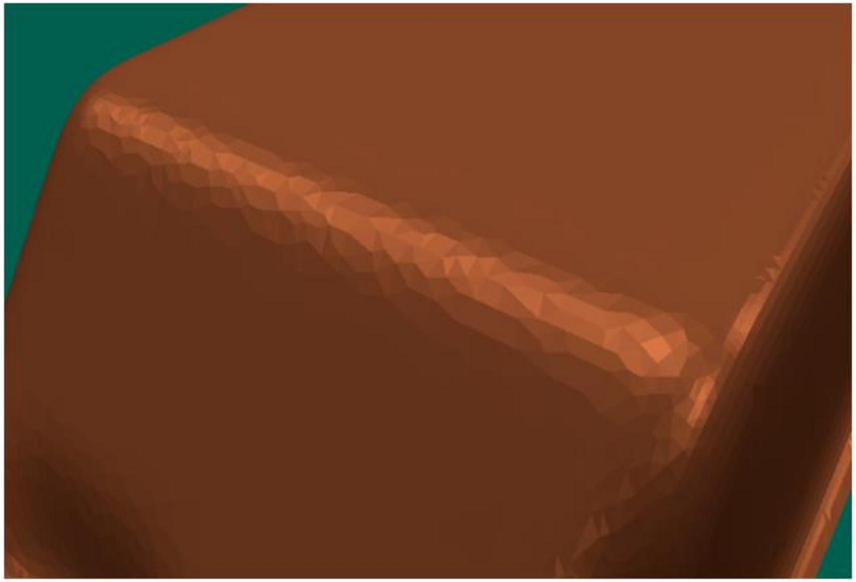
Fairness Criteria

- Rate quality of decimation operation
  - Approximation error
  - Triangle shape
  - Dihedral angles
  - Valence balance
  - Color differences



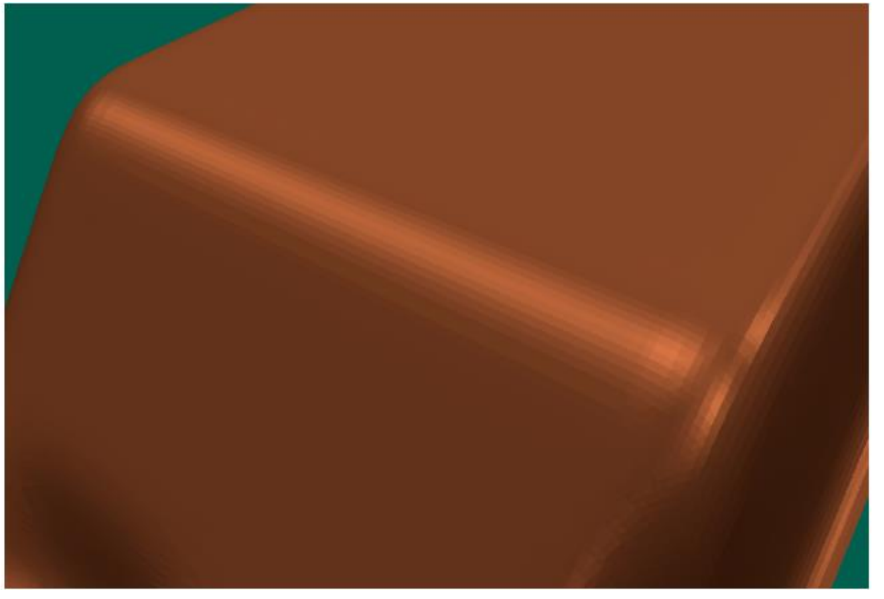
Fairness Criteria

- Rate quality of decimation operation
  - Approximation error
  - Triangle shape
  - Dihedral angles
  - Valence balance
  - Color differences



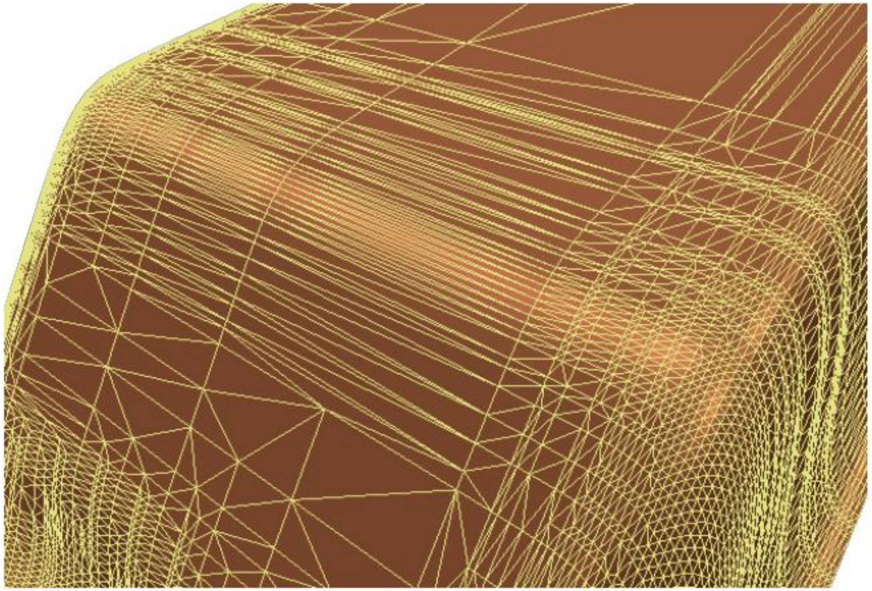
Fairness Criteria

- Rate quality of decimation operation
  - Approximation error
  - Triangle shape
  - Dihedral angles
  - Valence balance
  - Color differences



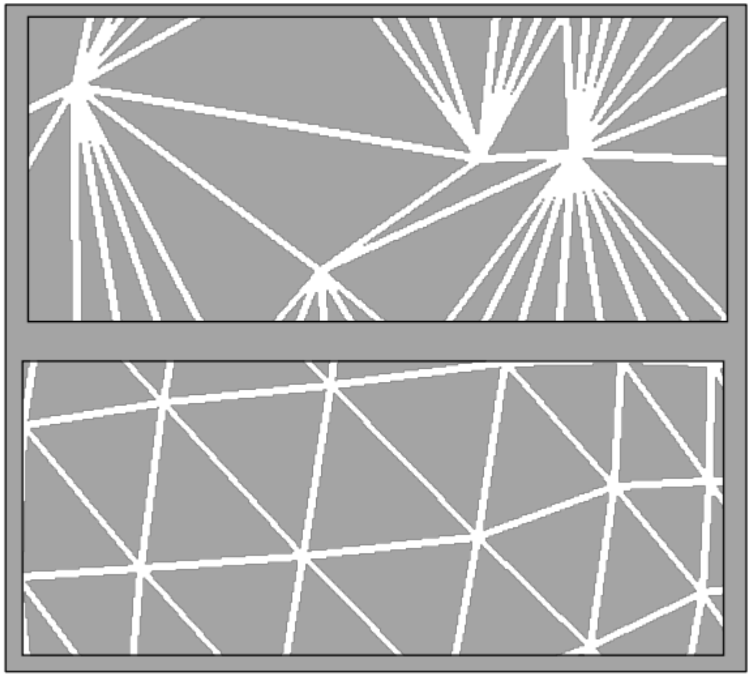
Fairness Criteria

- Rate quality of decimation operation
  - Approximation error
  - Triangle shape
  - Dihedral angles
  - Valence balance
  - Color differences



Fairness Criteria

- Rate quality of decimation operation
  - Approximation error
  - Triangle shape
  - Dihedral angles
  - Valence balance
  - Color differences



Comparison

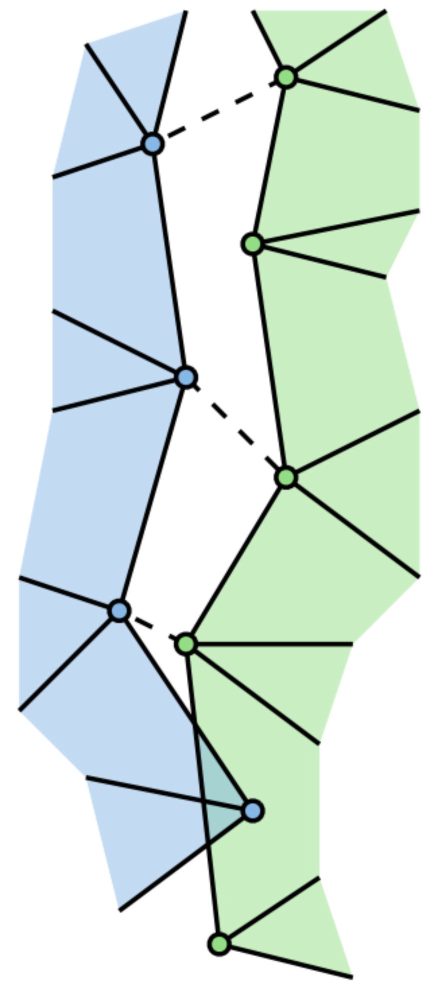
- Vertex clustering
  - fast, but difficult to control simplified mesh
  - topology changes, non-manifold meshes
  - global error bound, but often not close to optimum
- Iterative decimation with quadric error metrics
  - good trade-off between mesh quality and speed
  - explicit control over mesh topology
  - restricting normal deviation improves mesh quality

Mesh Repair

Surface-Oriented Algorithms

- Surface oriented approaches explicitly identify and resolve artifacts

- Methods
  - Snapping
  - Splitting
  - Stitching
  - …

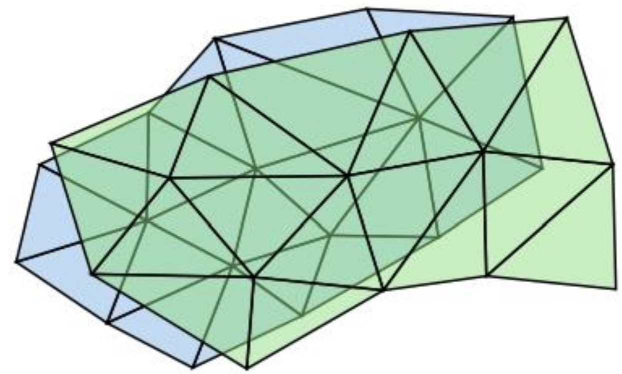


Surface-Oriented Algorithms

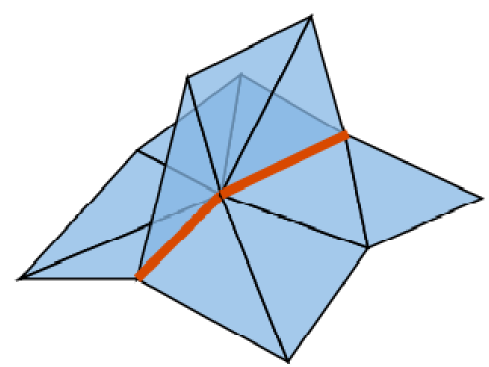
- Advantages
  - Fast
  - Conceptually easy
  - Memory friendly
  - Structure preserving, minimal modification of the input

Surface-Oriented Algorithms

- Problems
  - Not robust
    - Numerical issues
    - Inherent non-robustness

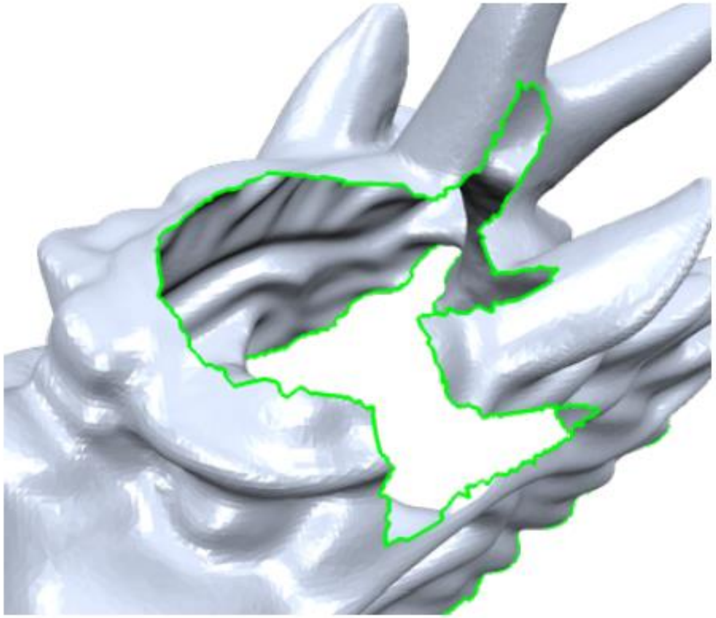


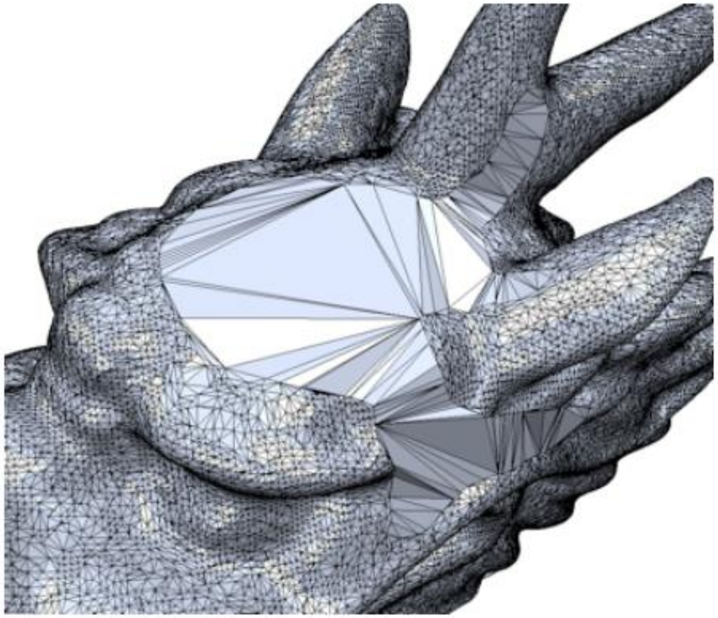
  - No quality guarantees on the output



Filling Holes in Meshes

- compute a coarse triangulation T

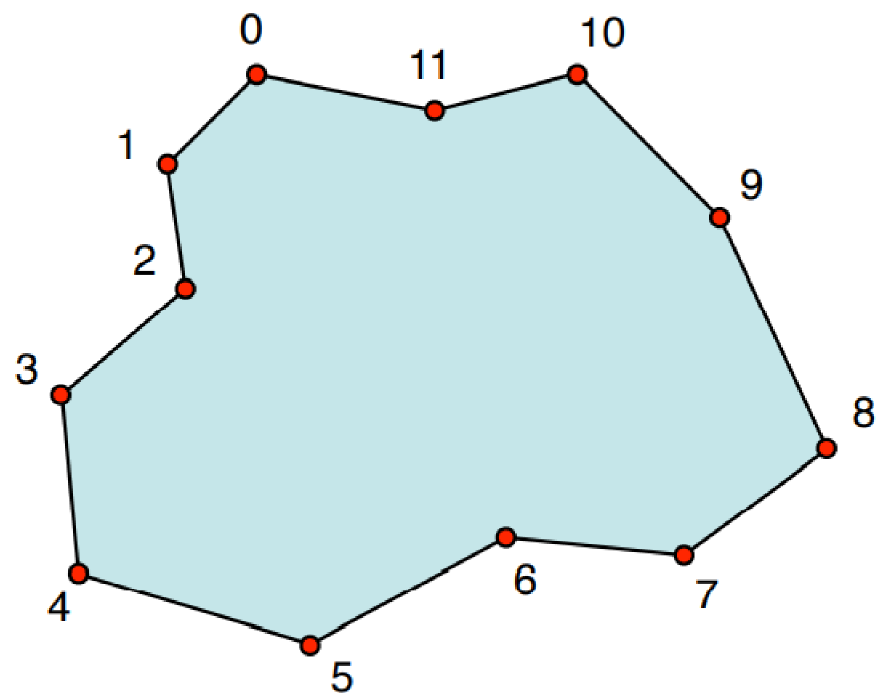




Filling Holes in Meshes

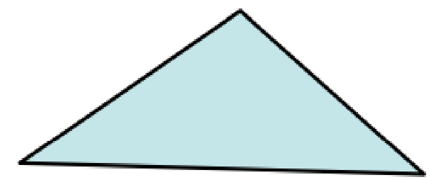
- compute a coarse triangulation T of minimal weight w(T)

n vertices,
n–2 triangles

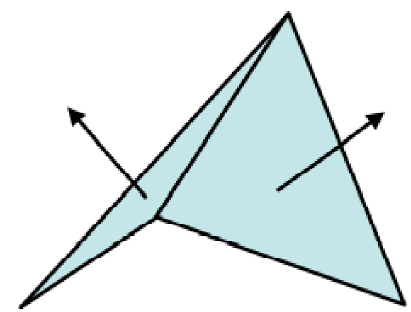


Filling Holes in Meshes

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



  - maximum dihedral angle in T

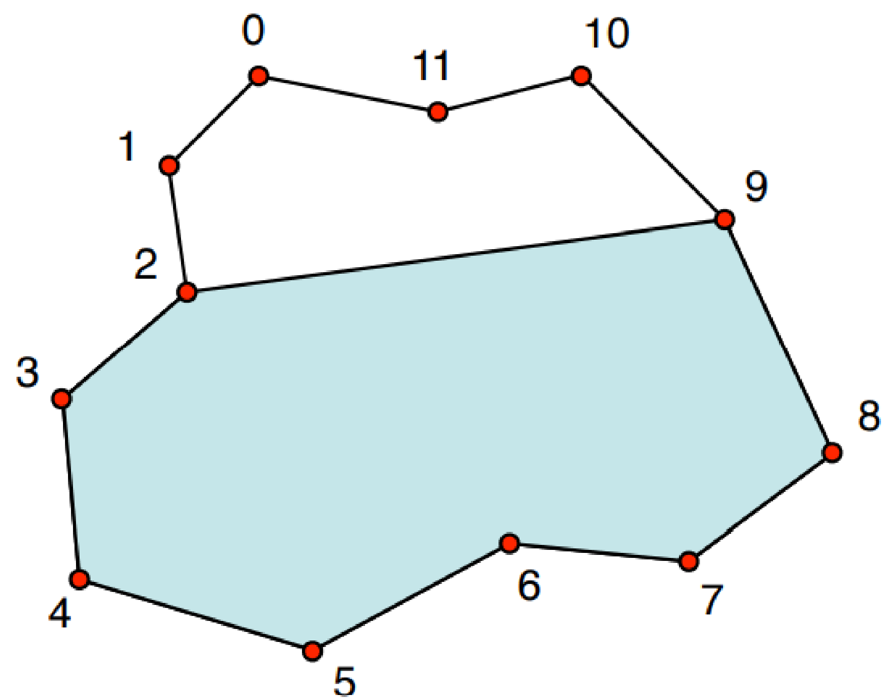


- thus, we favour triangulations of low area and low normal variation

Filling Holes in Meshes

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

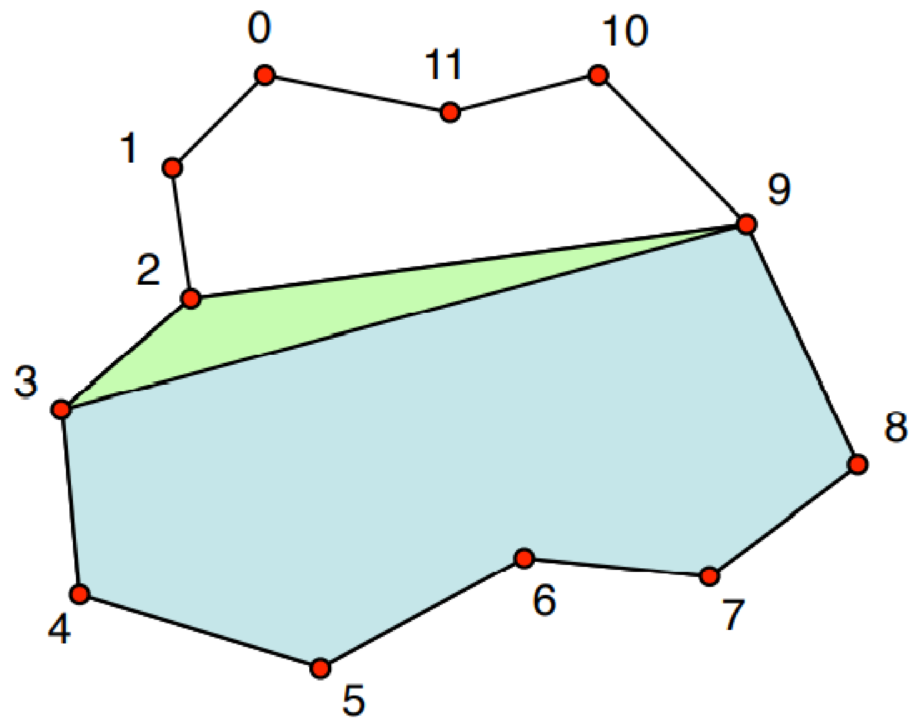
w[2,9] = ?



Filling Holes in Meshes

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

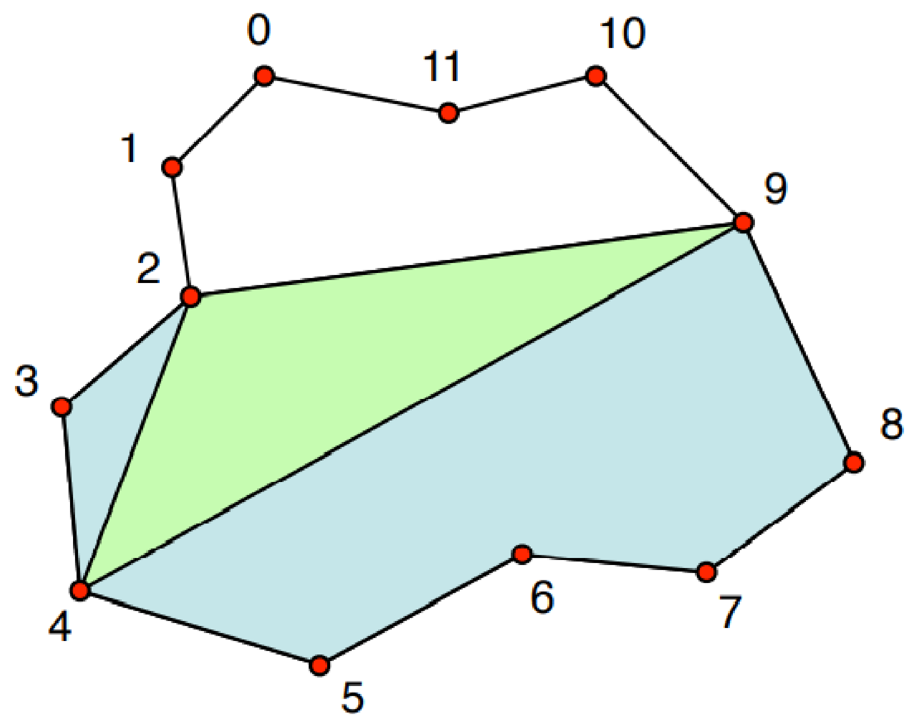
w[2,9] = min(
    w(Δ(2,3,9)) + w[3,9],



Filling Holes in Meshes

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

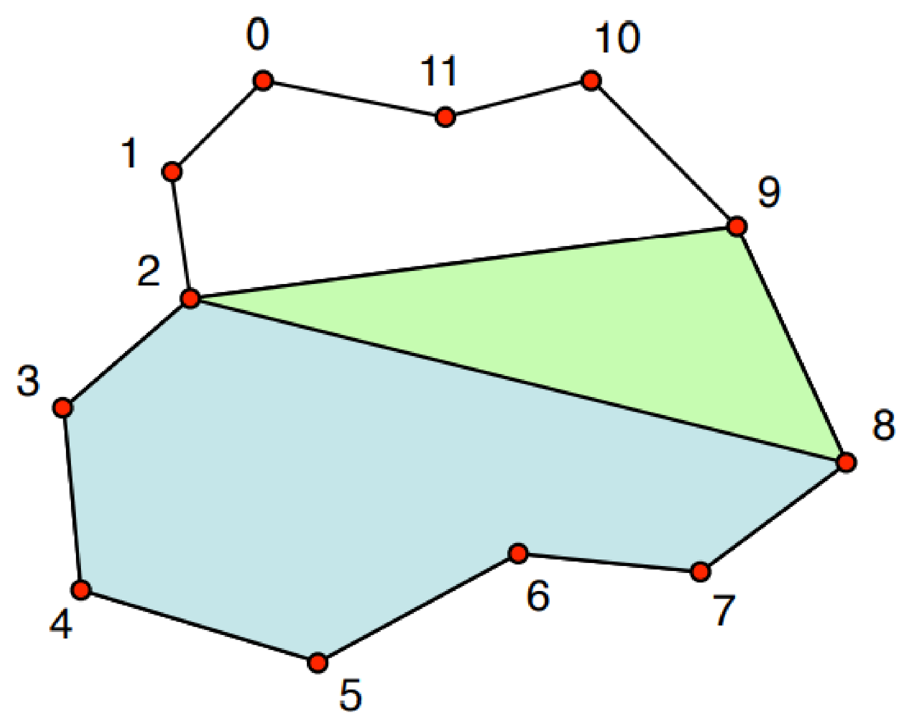
w[2,9] = min(
w(Δ(2,3,9)) + w[3,9],
w[2,4] + w(Δ(2,4,9)) + w[4,9],



Filling Holes in Meshes

- 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

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

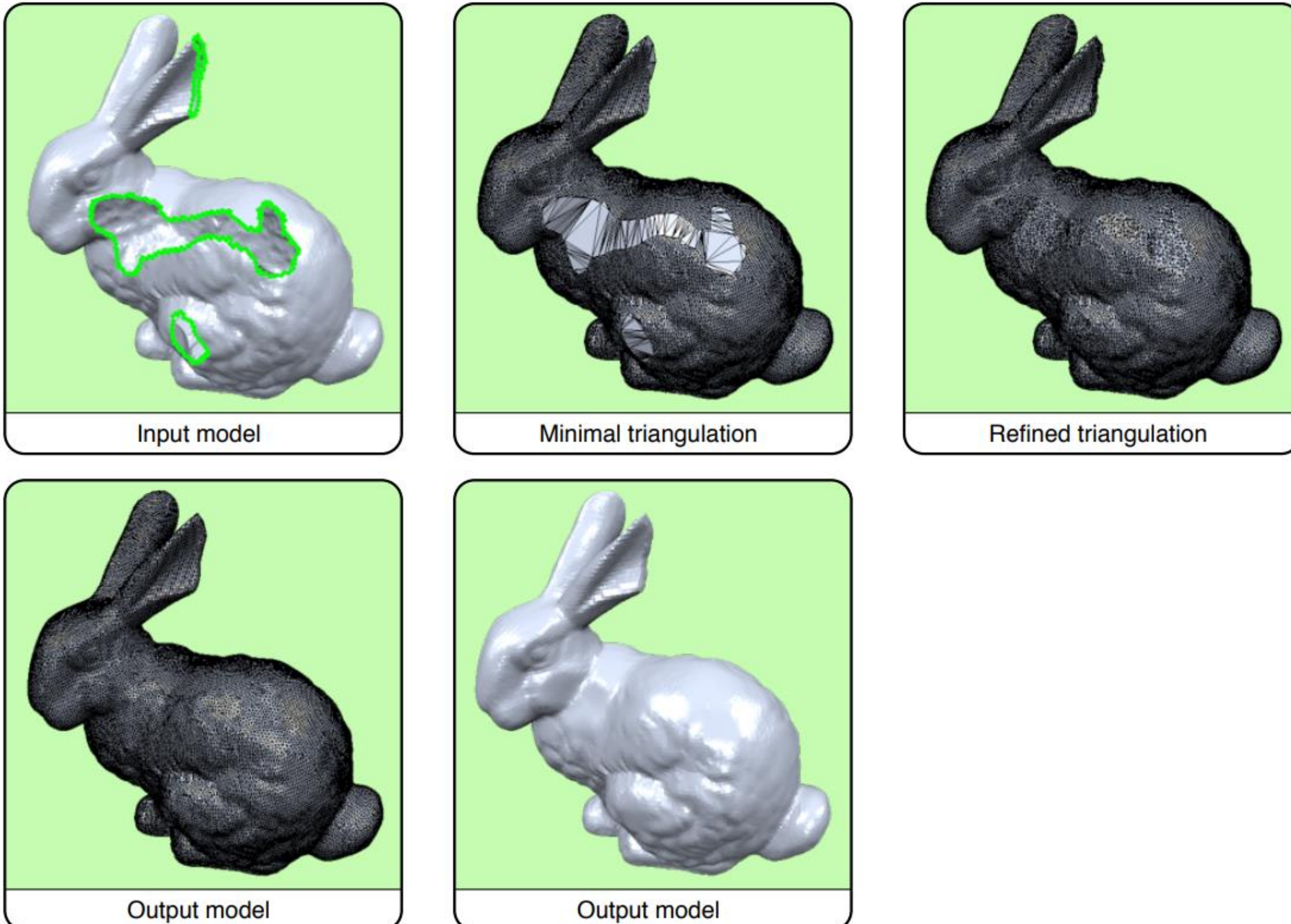
$$w[a,c] = \min_{a<b<c} w[a,b] + w(\Delta(a,b,c)) + w[b,c]$$

$$w[x,x+1] = 0$$

- dynamic programming leads to an $O(n^3)$ algorithm

Additional Steps

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

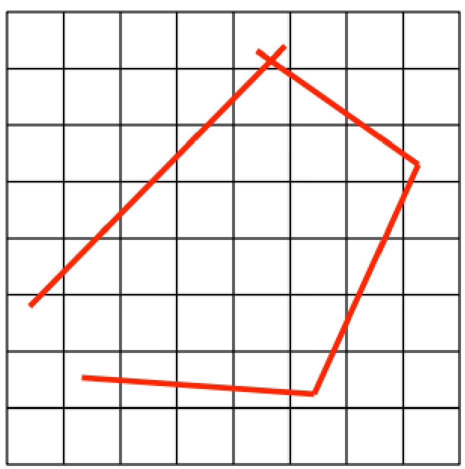

Input model
Minimal triangulation
Refined triangulation
Output model
Output model

Filling Holes in Meshes

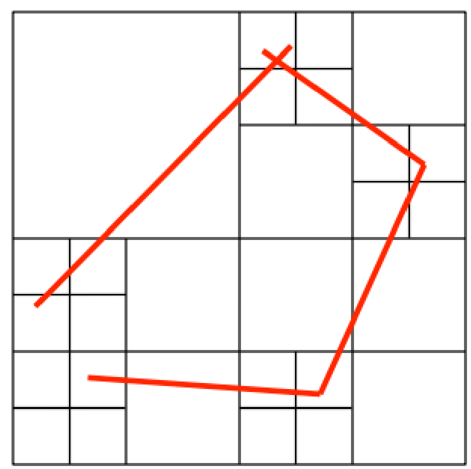
- What problems do we encounter?
  - Islands are not incorporated
  - Self-intersections cannot be excluded
  - Quality depends on boundary distortion

Volumetric Algorithms

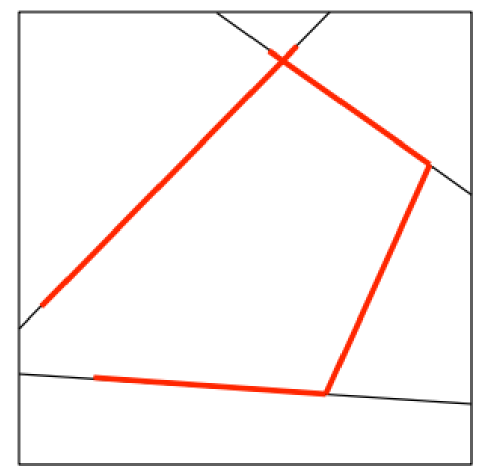
1. convert the input model into an intermediate volumetric representation → loss of information



voxel grid



adaptive octree



BSP tree

Volumetric Algorithms

1. convert the input model into an intermediate volumetric representation → loss of information
2. discrete volumetric representation → robust and reliable 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 and reliable processing
   - morphological operators (dilation, erosion)
   - smoothing
   - flood-fill to determine interior/exterior
3. extract the surface of a solid object from the volume → manifold and watertight

Volumetric Algorithms

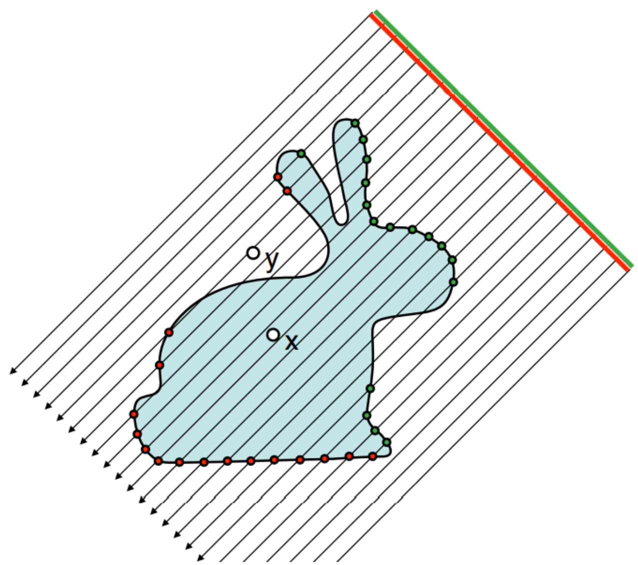
- Advantages
  - fully automatic
  - few (intuitive) user parameters
  - Robust
  - guaranteed manifold output

Volumetric Algorithms

- Problems
  - slow and memory intensive
    - adaptive data structures
  - aliasing and loss of features
    - feature sensitive reconstruction (EMC, DC)
  - loss of mesh structure
    - bad luck (... hybrid approaches)
  - large output
    - mesh decimation

Nooruddin and Turk’s Method

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



Summary

- Learn basic operations under the mesh representation

- Learn how to convert other representations into the mesh representation