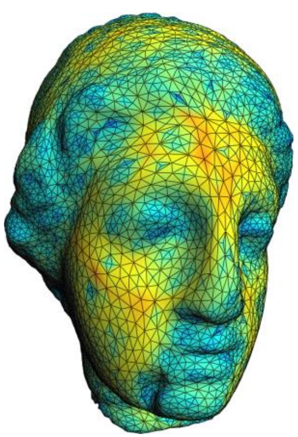
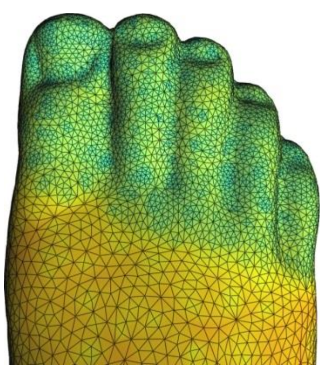
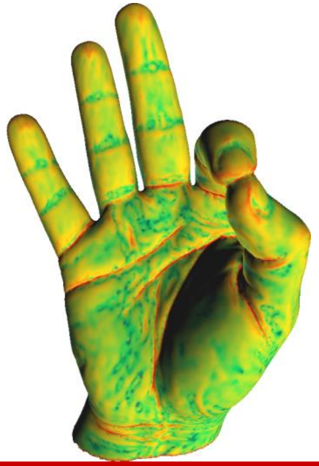
Polygonal Mesh

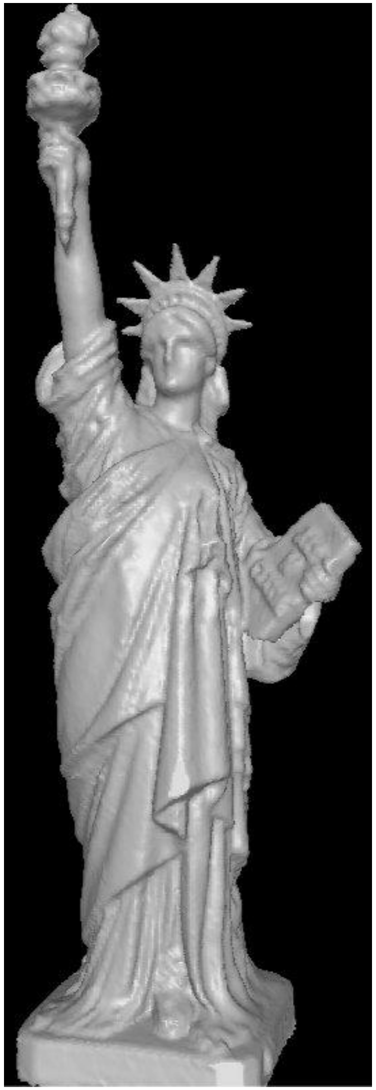
- Representation and properties
- Data Structure
- Simplification, compression, LOD
- Parametrization, Fairing

Polygonal Mesh

78K vertices

160K triangle



Polygonal meshes are simply collections of polygons, or “faces,” that together form the “skin” of an object. They have become a standard way of representing a broad class of shapes in graphics.

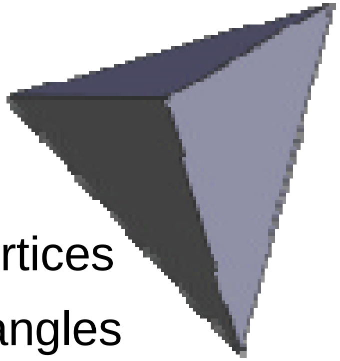
ACCURACY

vs.

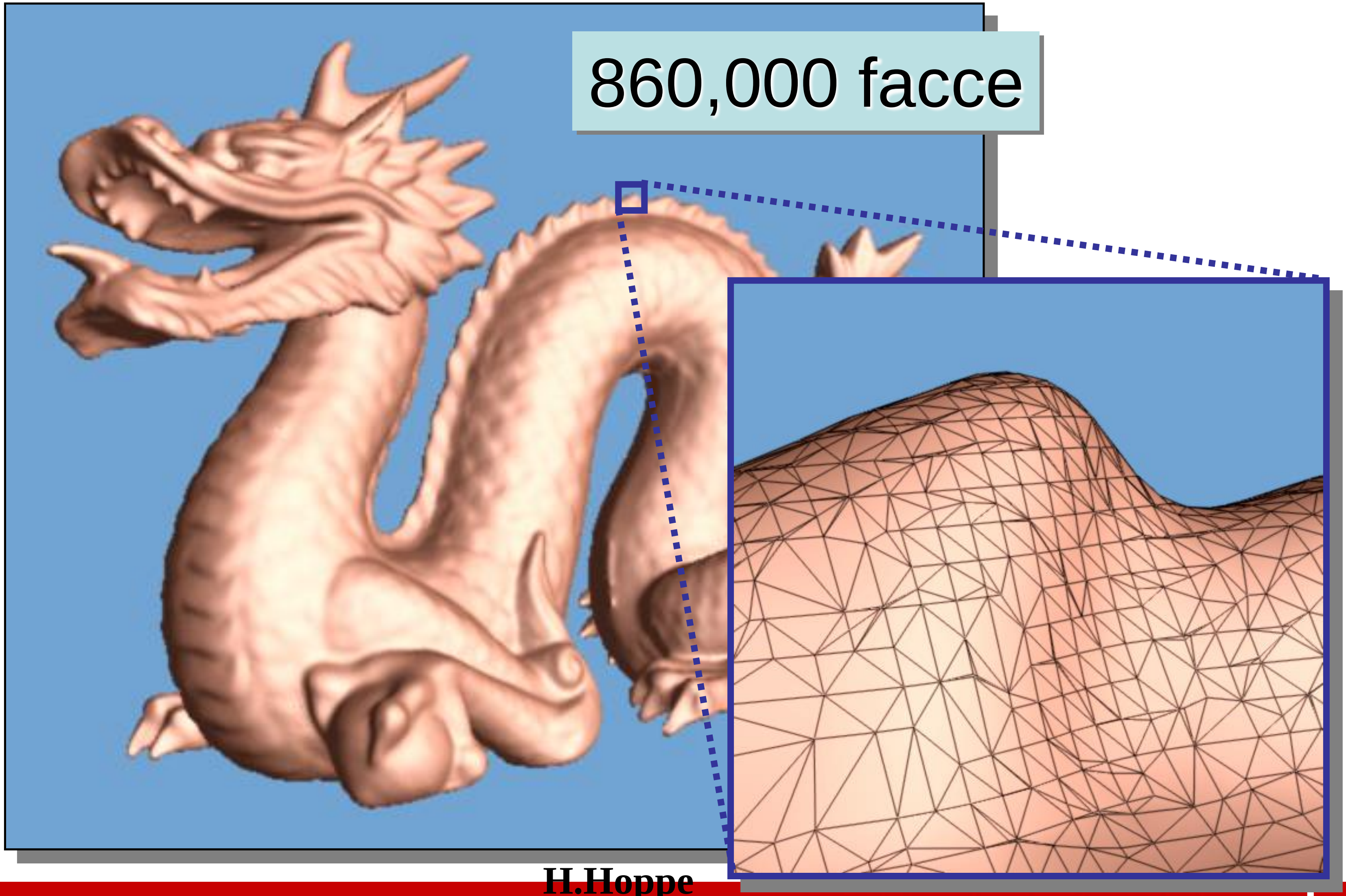
SPEED

4 vertices

4 triangles



Complex meshes in CG..



H.Hoppe

Complex meshes in art…

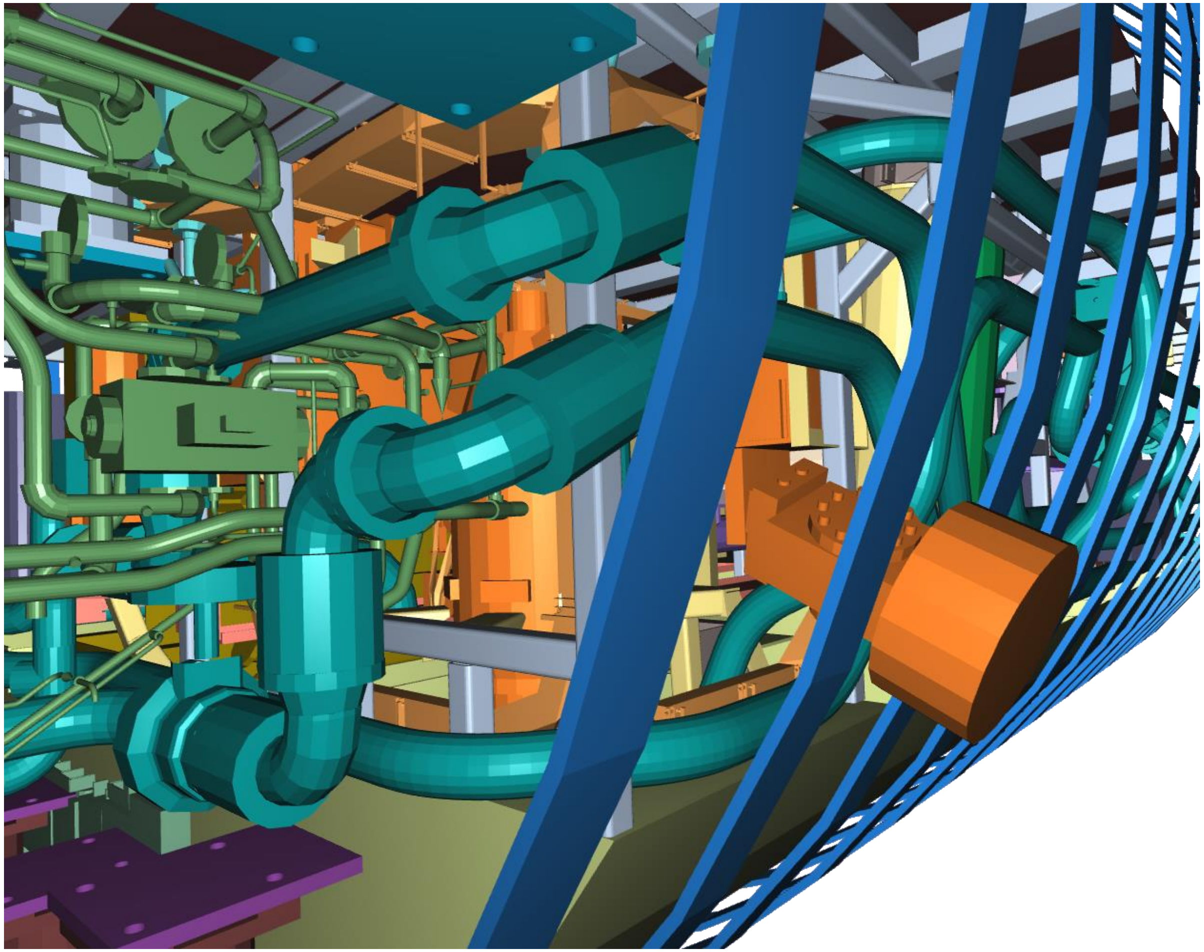
[Digital Michelangelo Project, 2000]

2,000,000,000 faces

Objectives:
- rendering
- storage
- transmission
- scalability

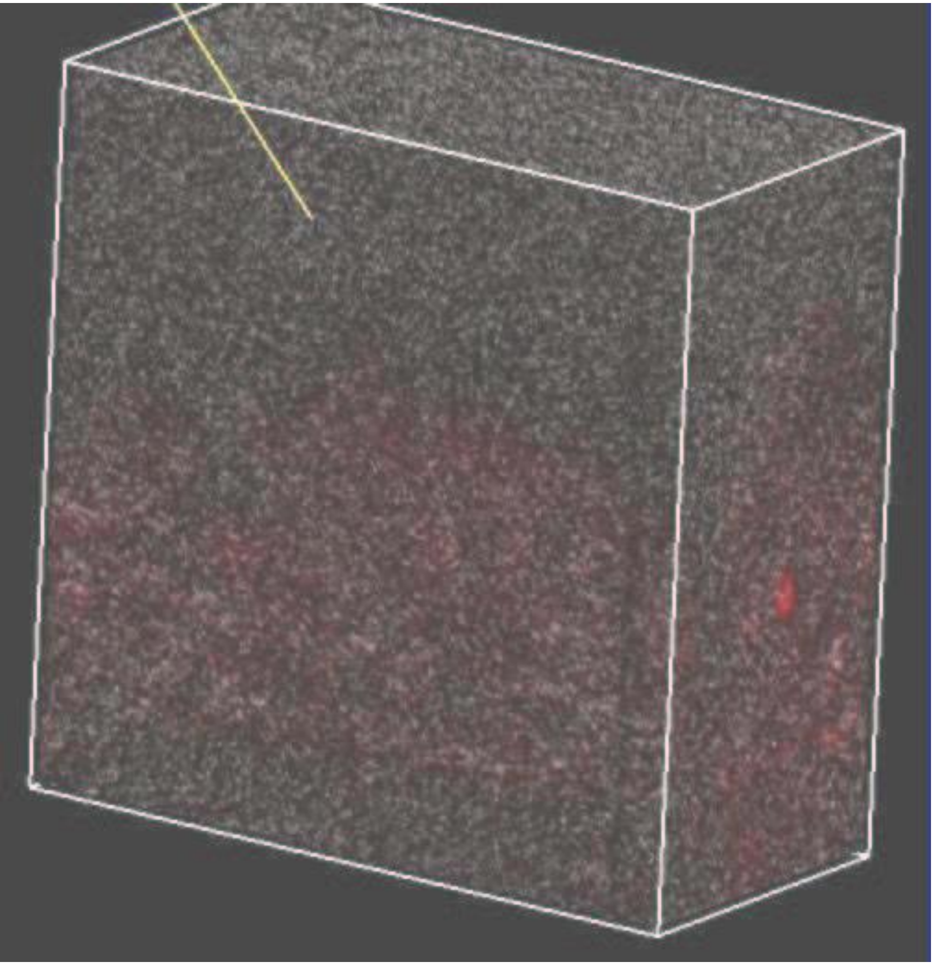
Complex meshes in CAD

- Submarine Auxiliary Machine Room
- 500,000 polygons

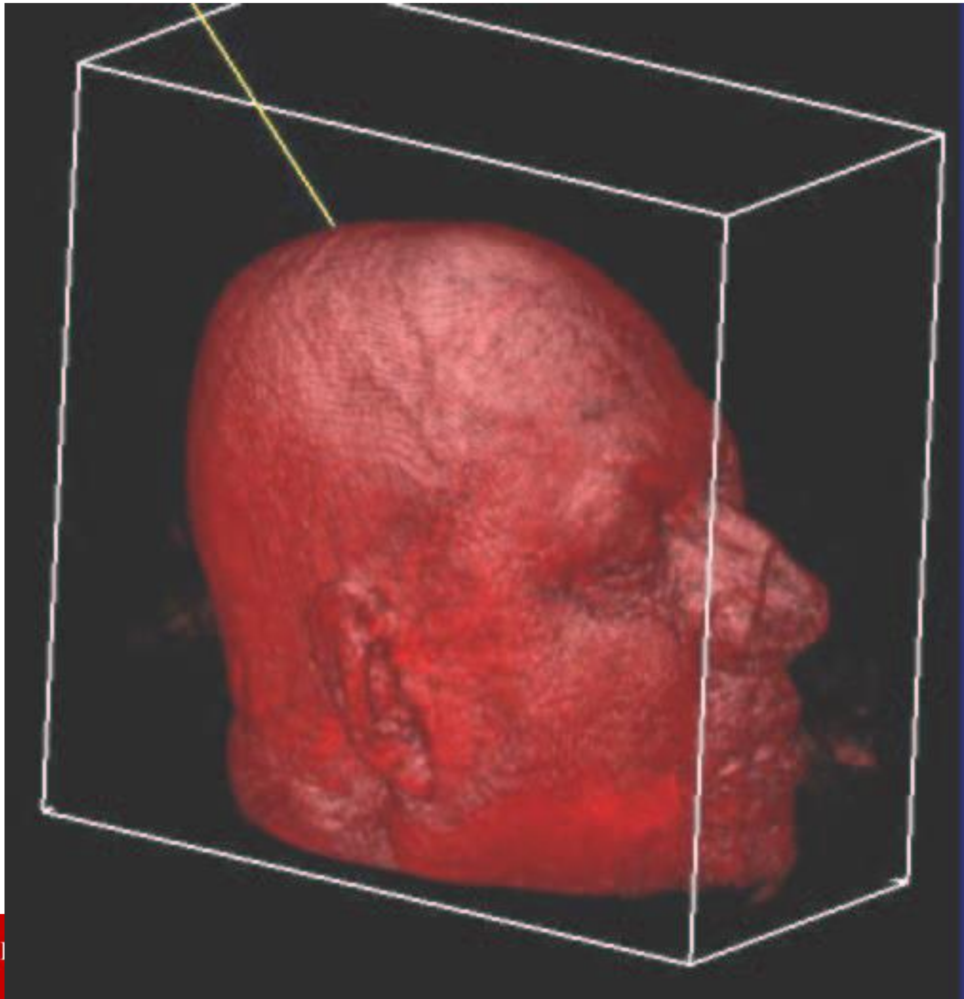


Courtesy General Dynamics, Electric Boat Div.

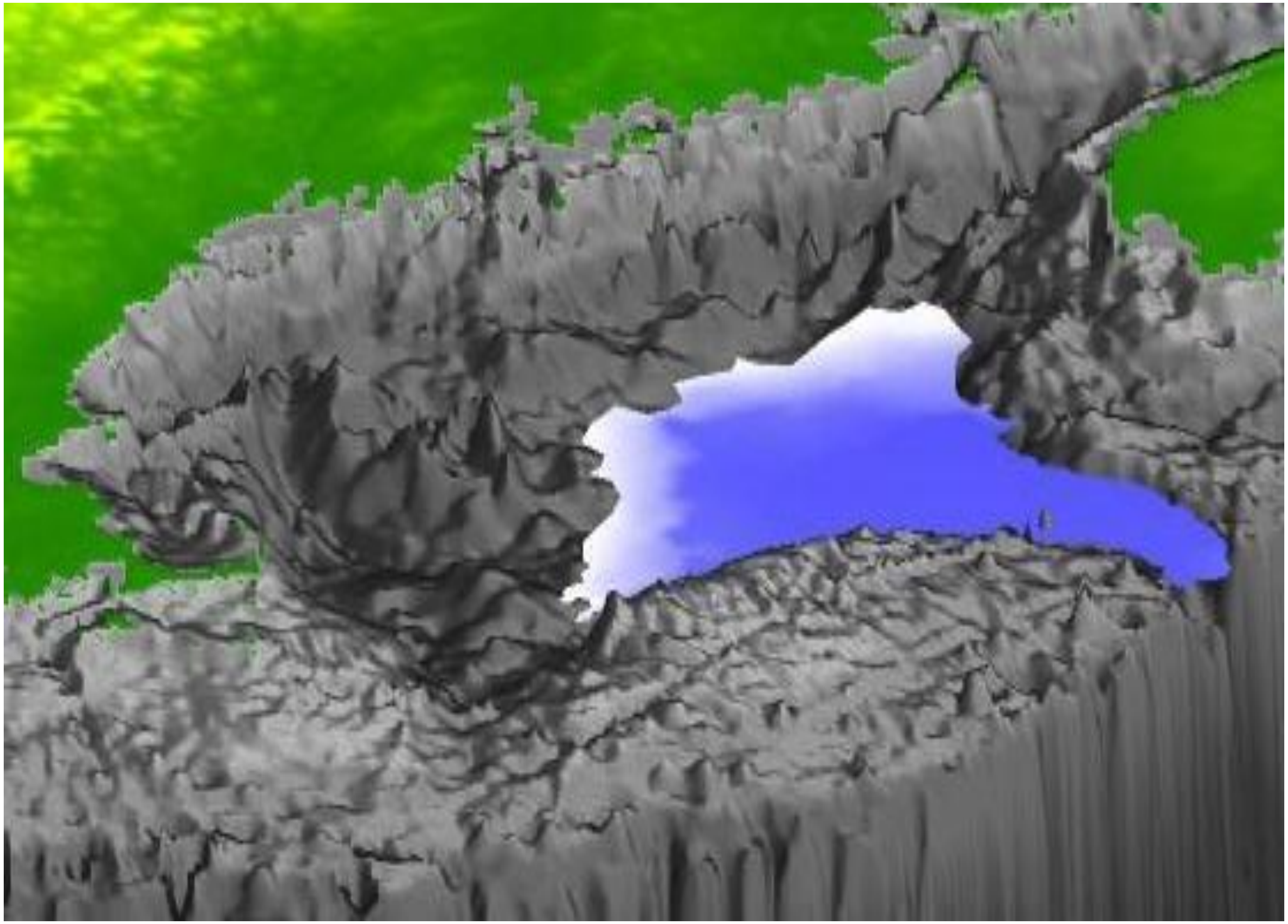
Meshes in medical imaging...



Visible-body: 512x512x1734 punti

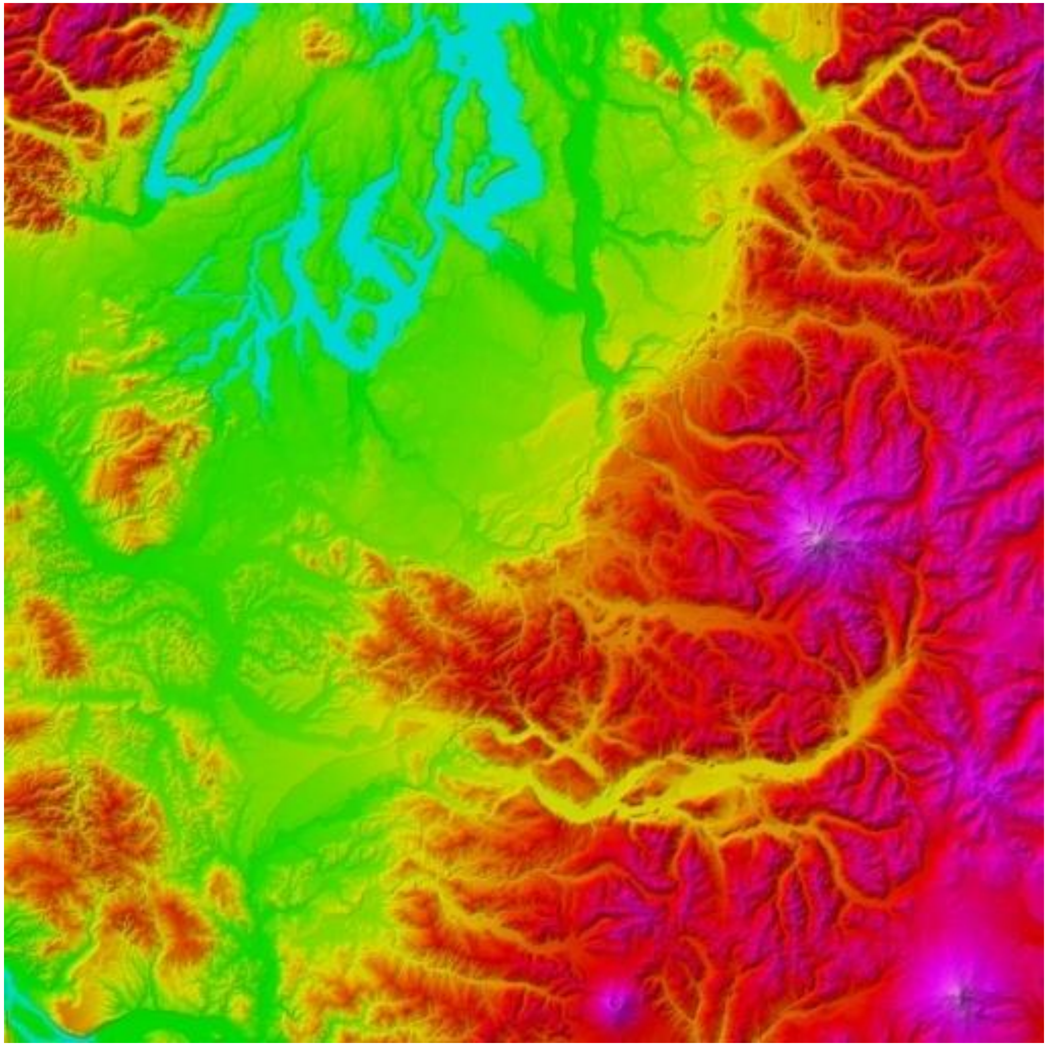


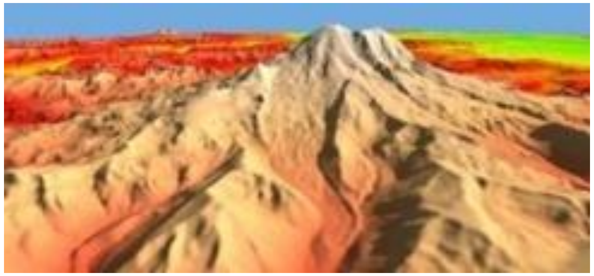
Complex meshes for terrain



topography

Territorial Data: special meshes





16K x 16K vertices
~537 milion of triangles

View-dependent,
Geometric model +
bump (texture) mapping

Height field

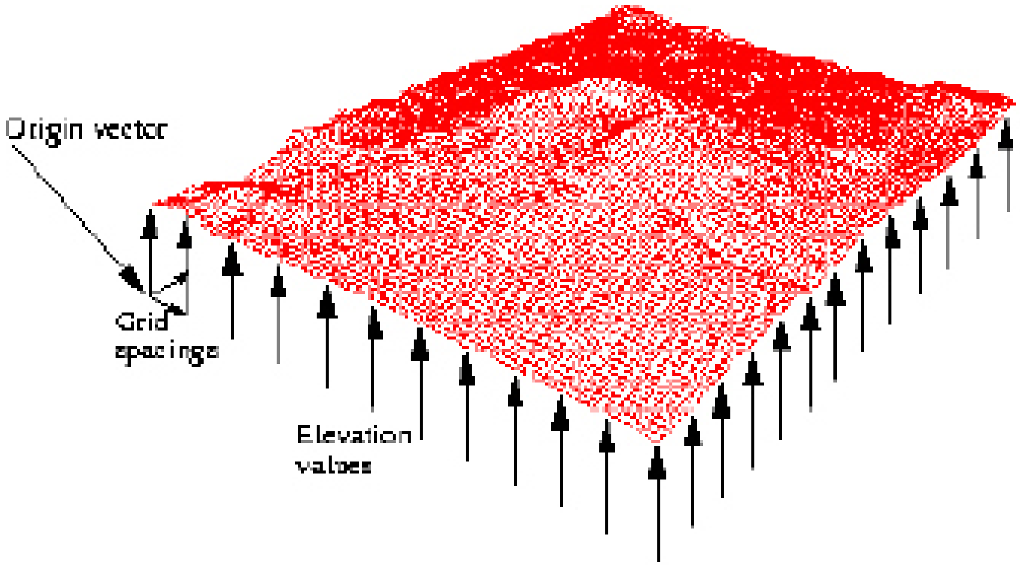
Visualize the explicit function:
$z = f(x,y)$

A height field is defined on a regular grid of quotes

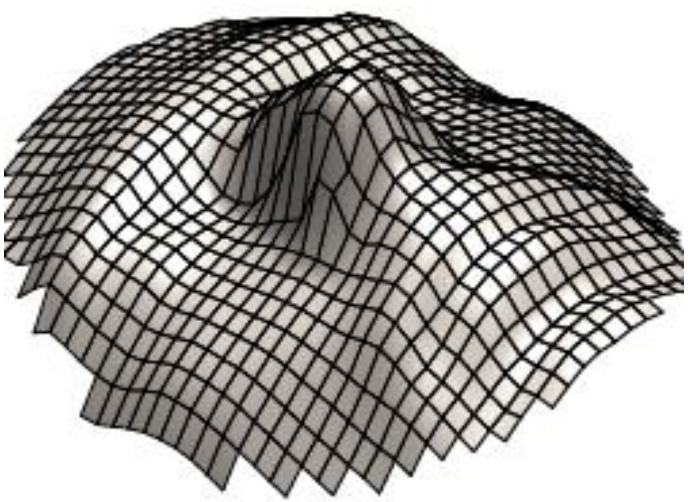
$h : [0, N - 1]^2 \rightarrow \mathbf{R}$,

where N is huge

Store the height
as an image (i.e. Format gif)



Geometry images



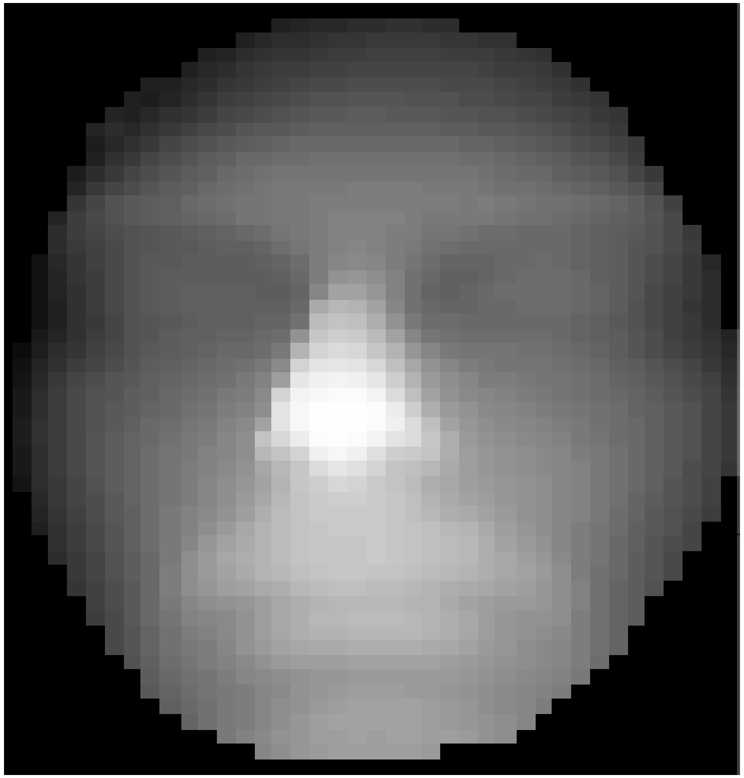
Surface

$$\left(u, v, z(u, v)\right)$$

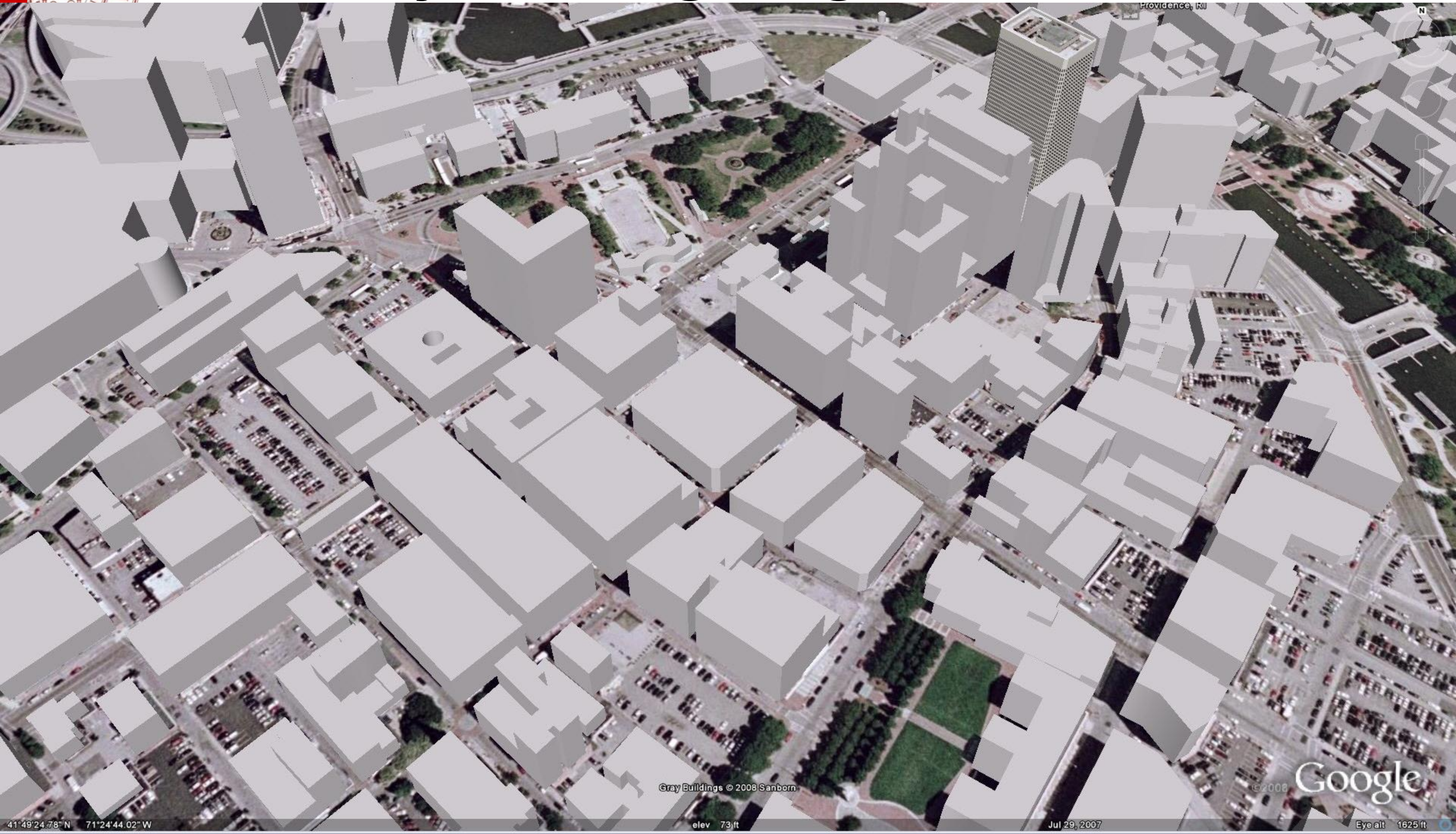
$$(u_{ij}, v_{ij}) = (i\Delta u,\ j\Delta v)$$

$$Z_{ij} = z(u_{ij}, v_{ij})$$

Geometry image



.obj from google earth

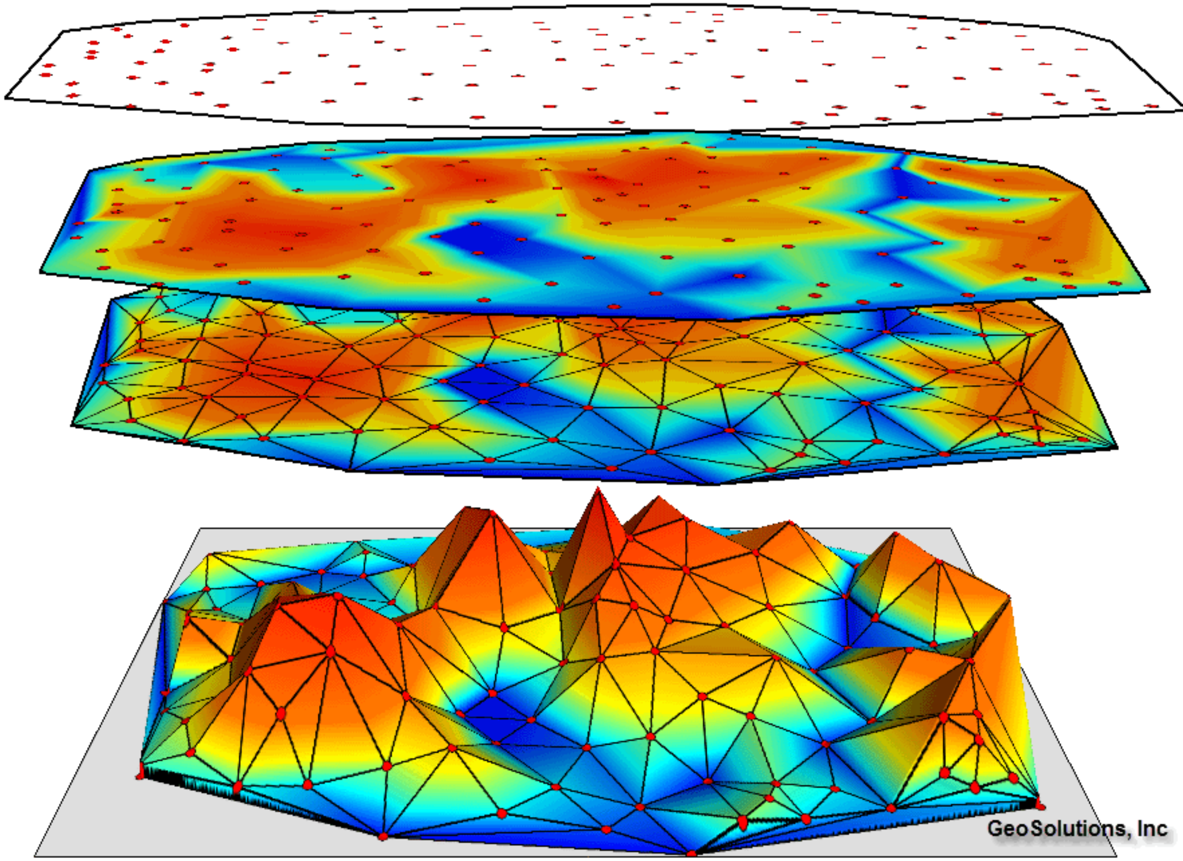


Height field as a TIN

Alternatively, a height field is stored as

triangular irregular networks (TIN) or mesh: a vector-based representation of a surface

use an optimal number of polygons to approximate a surface to a given level of detail and accuracy

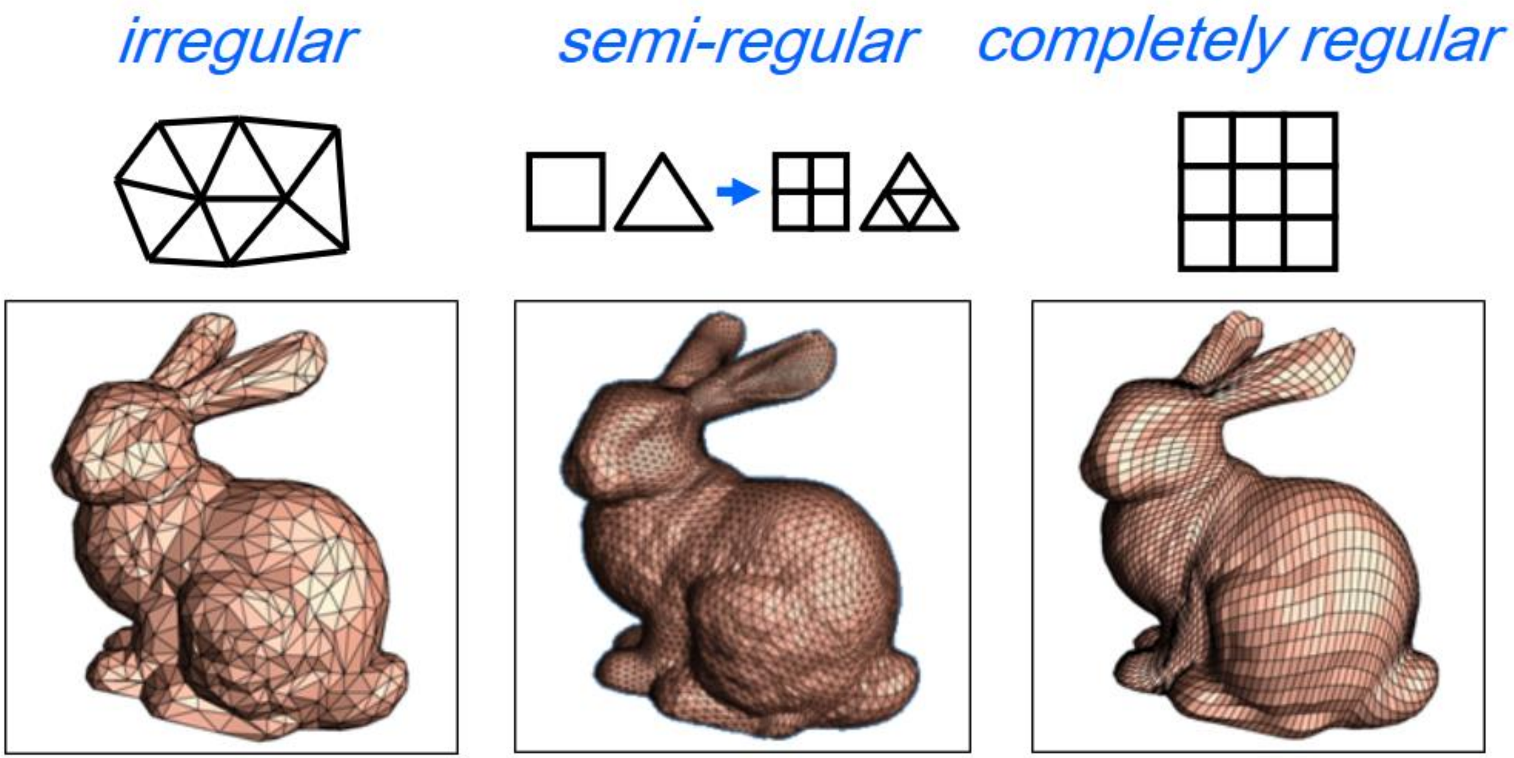


Mesh generation

- **Mesh from points:**
  triangulation or tetraedralization
  - Sampling by a digitizer
  - Reconstruction from multiple views
  - 3D Scanner
  - Territorial models
- **Mesh from surfaces**: tessellation (for surface rendering)
- **Mesh from data volumes**: polygonalization
  - Volumetric data (isosurfacing: extract a surface of constant value through the volume)

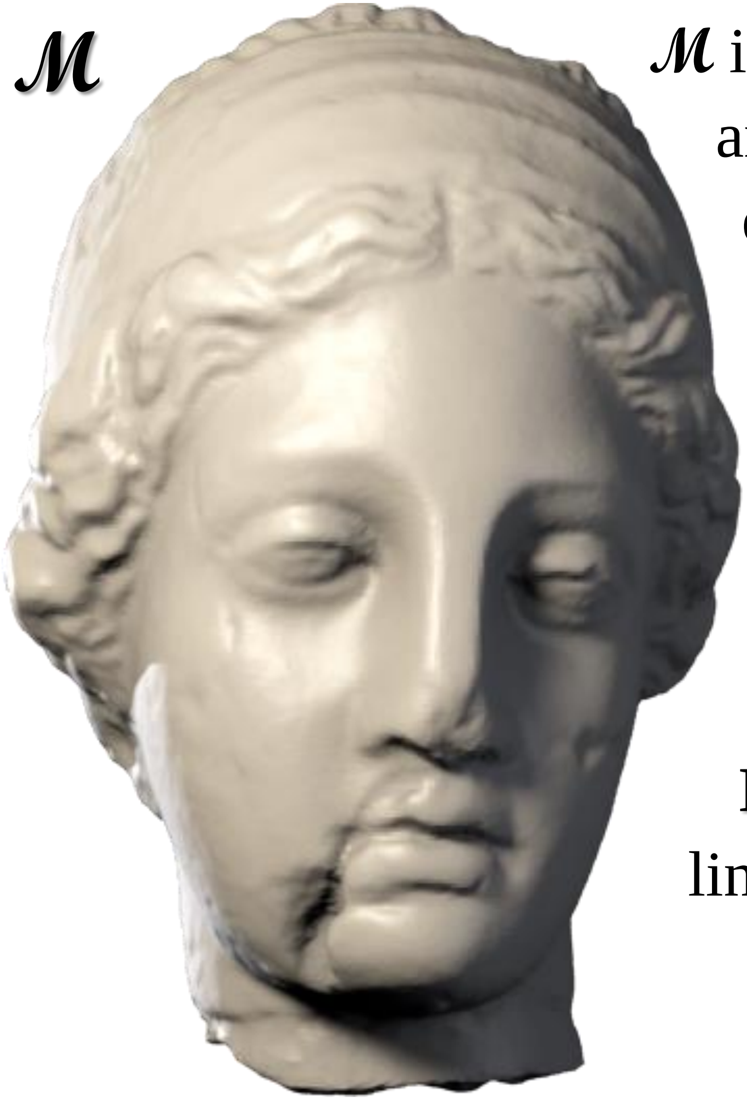
Mesh representation

- **Structured (regular) mesh** : all internal vertices are surrounded by a costant number of elements.
- **Semi-regular mesh** is obtained by regular subdivision of an irregular mesh: all the vertices are regular except for a small number of extraordinary vertices



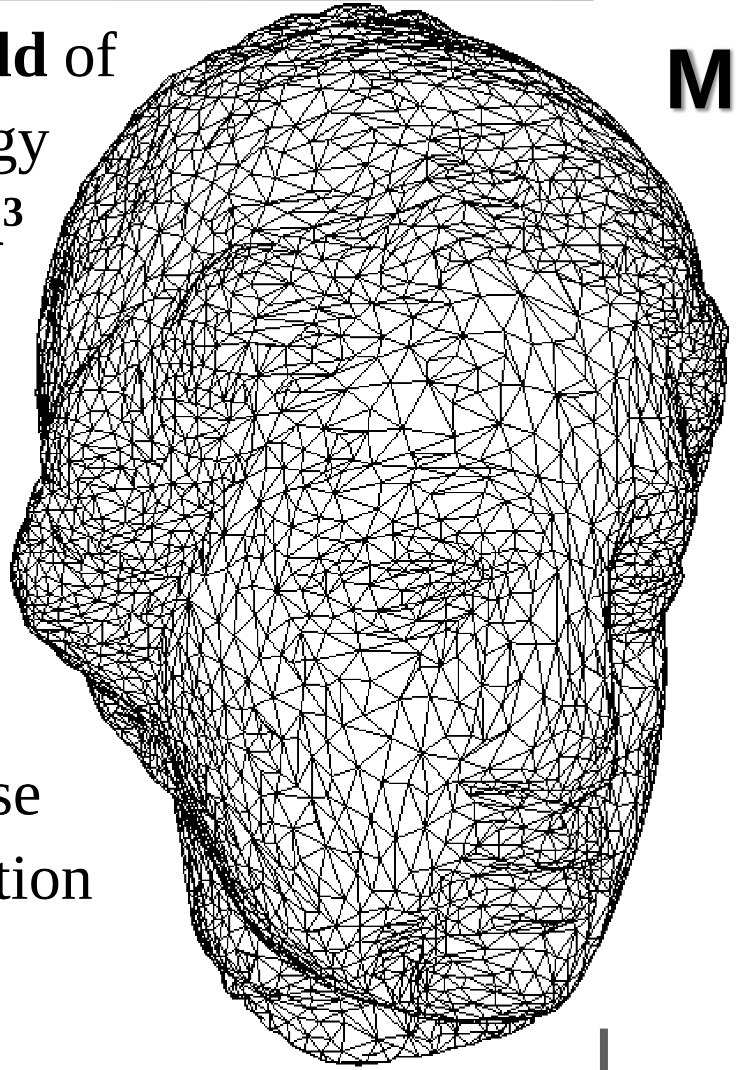
Mesh & $\mathcal{M}$anifold

$\mathcal{M}$



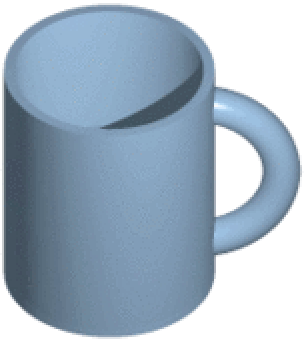
$\mathcal{M}$ is a 2D-**manifold** of arbitrary topology embedded in $\mathbf{R}^3$

M is a piecewise linear approximation of $\mathcal{M}$

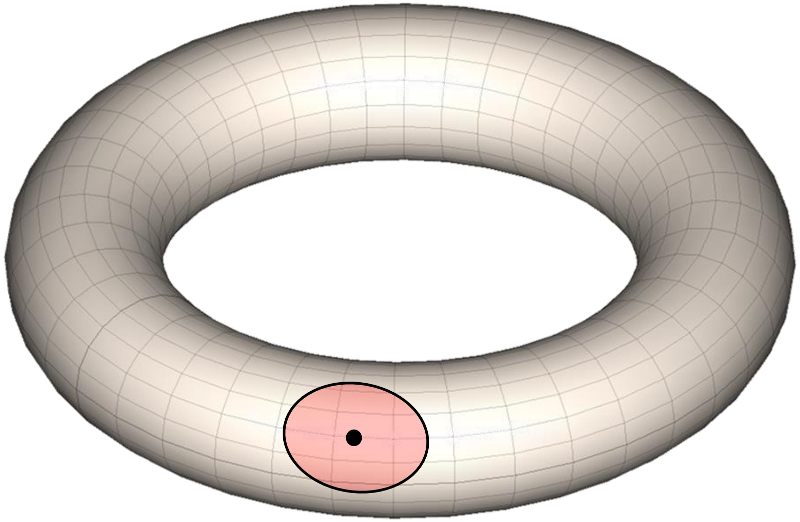


M

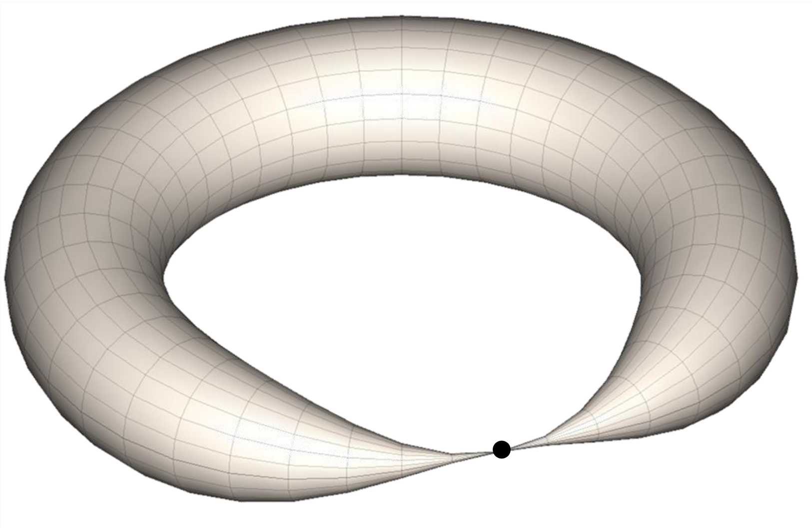
Manifold



A topological space in which every point has a neighborhood homeomorphic to $\mathbb{R}^n$(topological disc) is called an *n*-dimensional (or ***n*-**) **manifold**



2-manifold

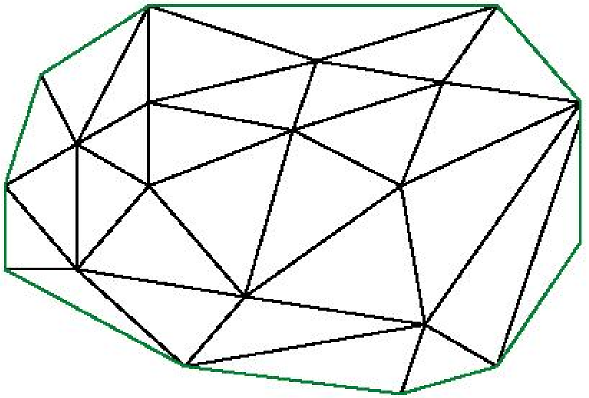


Not a manifold

Earth is an example of a 2-manifold

Triangle Mesh

A triangle mesh M consists of a **geometric** and a **topological** component, where the latter can be represented by a graph structure of the form (V,E,F) consisting of

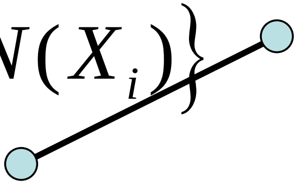


Vertices

$$V = \{1, .., N_v\}$$

Edges

$$E = \{(i,j) \in V \times V : X_j \in N(X_i)\}$$



between two vertices

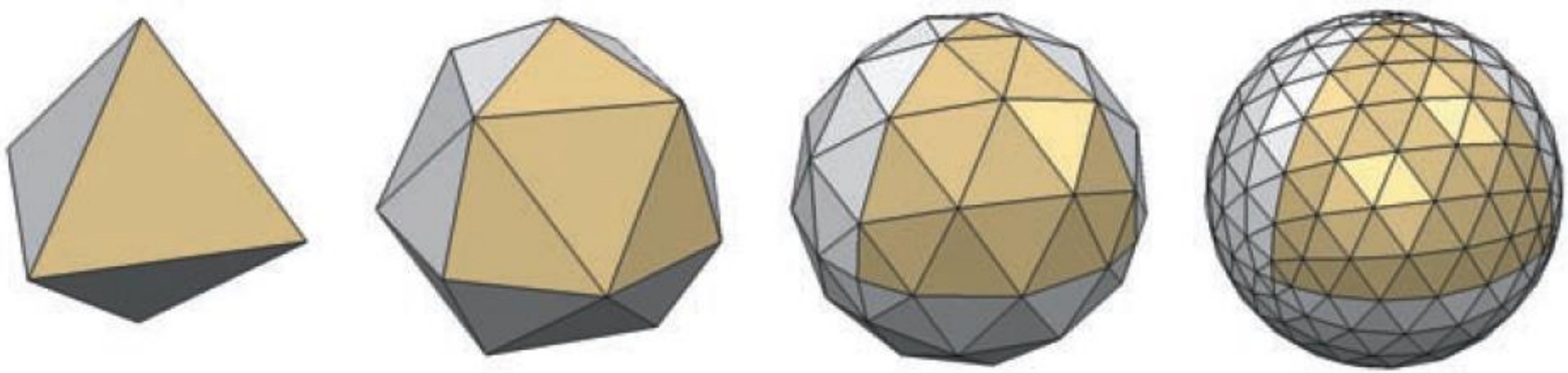
Faces

$$F = \{(i,j,k) \in V \times V \times V : (i,j),(i,k),(k,j) \in E\}$$

between edges

Approximation Quality

- If a sufficiently smooth surface is approximated by a piecewise linear function, the approximation error is of the order $O(h^2)$, with h denoting the maximum edge length.



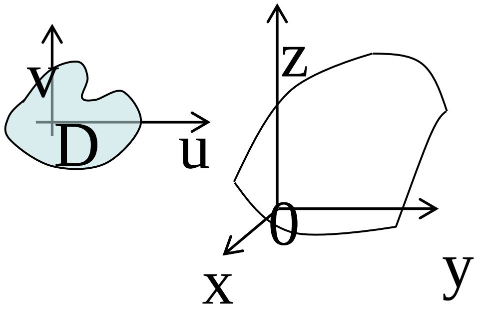
…..the error is reduced by a factor of about 1/4 ….

[Botsch et al.]

- The actual magnitude of the approximation error depends on the curvature of the underlying smooth surface.

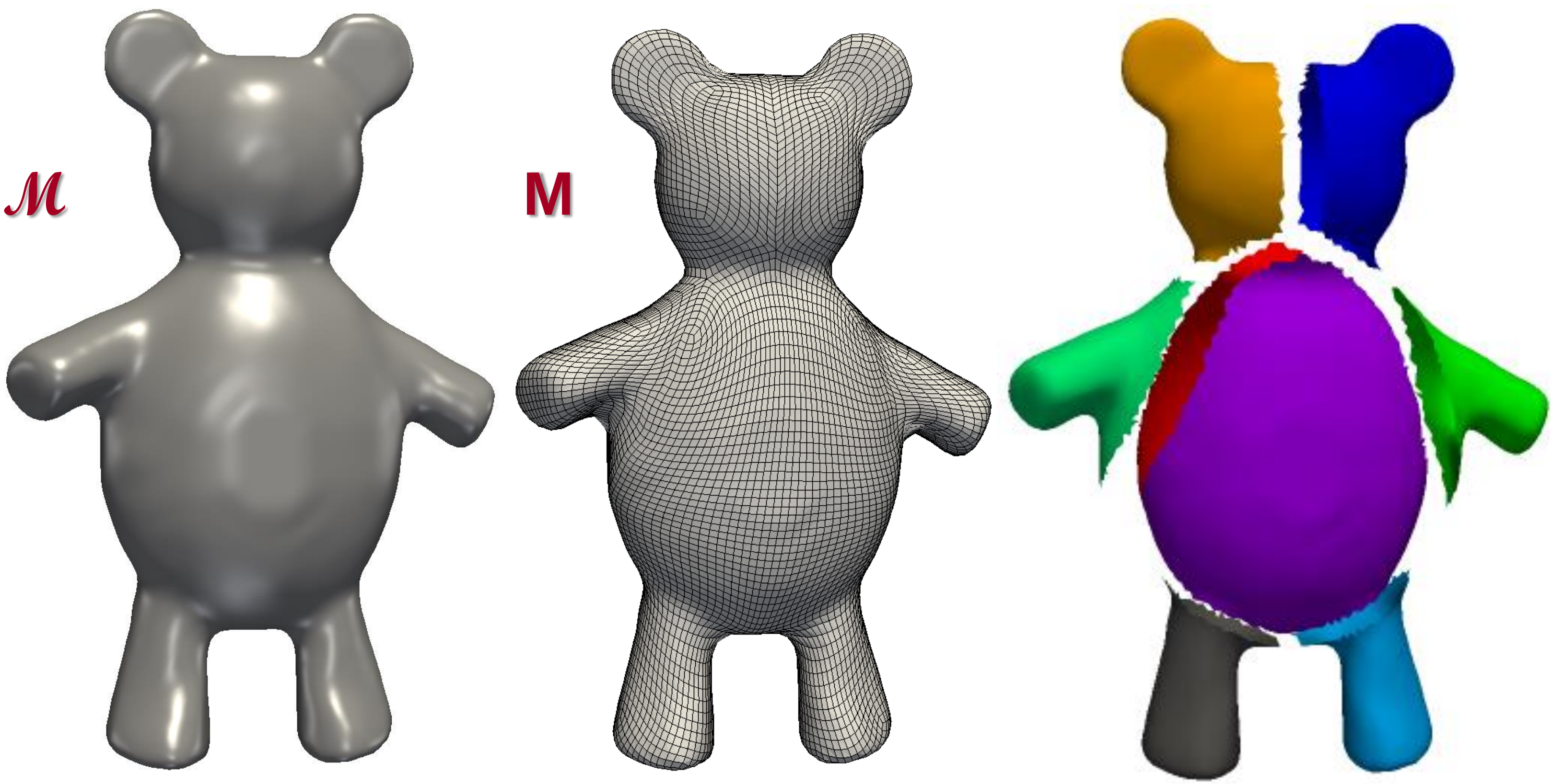
2D-manifold (Embedded surface)

- Boundaries of tangible physical objects are two-dimensional manifolds.
- They reside in (are embedded into, are subspaces of) the ambient three-dimensional Euclidean space.
- Two-dimensional manifolds are also called embedded surfaces (or simply surfaces).



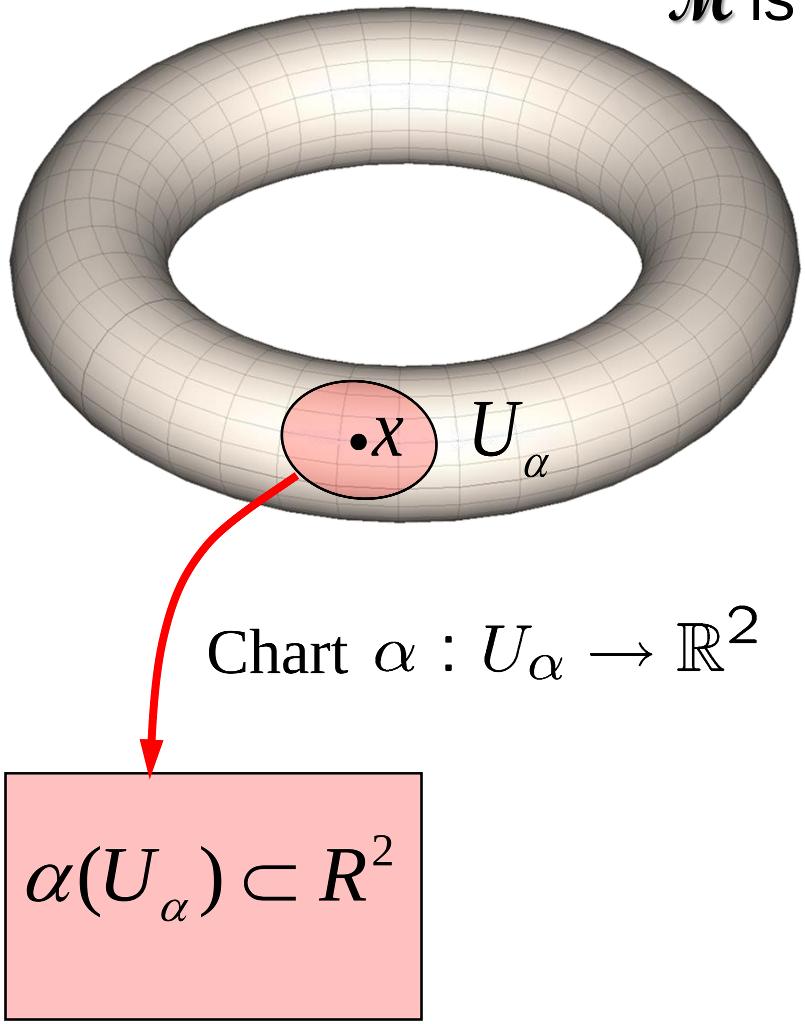
- Can often be described by the map $S : D \subset R^2 \rightarrow R^3$
  - $D \subset R^2$ is a **parameter domain**.
  - the map $S(u,v) = \left(x(u,v) \quad y(u,v) \quad z(u,v) \right)^T$ is a global **parameterization** (embedding) of the manifold.
- Smooth global parameterization does not always exist or is easy to find.

- The description of most manifolds requires more than one parametric domain (more than one patch).
- A single patch is adequate for only the simplest manifolds.



Charts and atlases

$\mathcal{M}$ is a 2D manifold of arbitrary topology embedded in R^3.



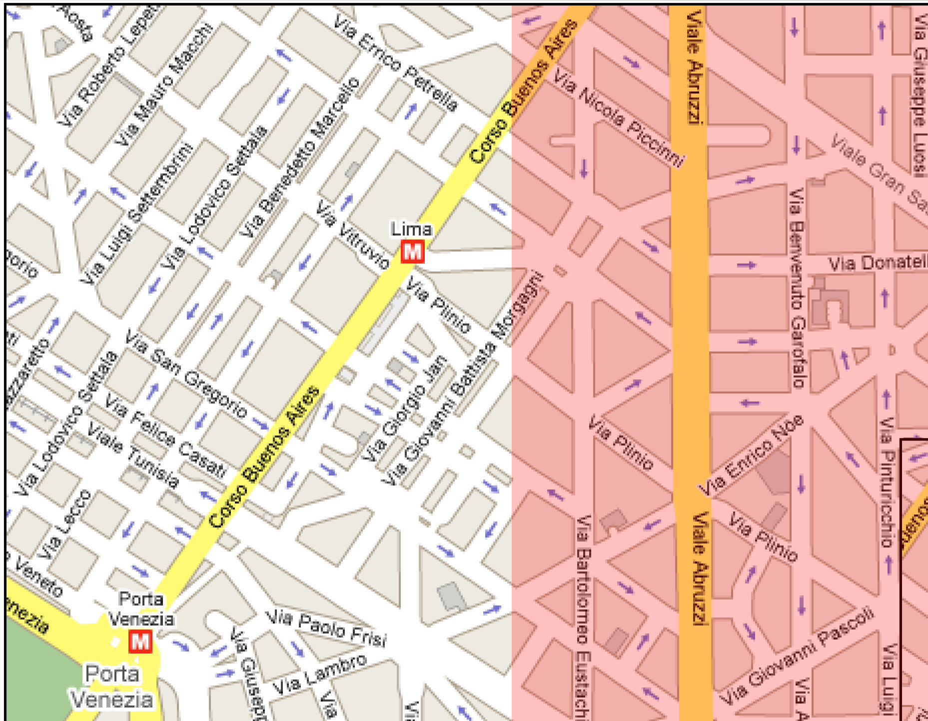
A homeomorphism $\alpha : U_\alpha \to R^2$ from a neighborhood $U_\alpha \subset \mathcal{M}$ to R^2 is called a **chart**

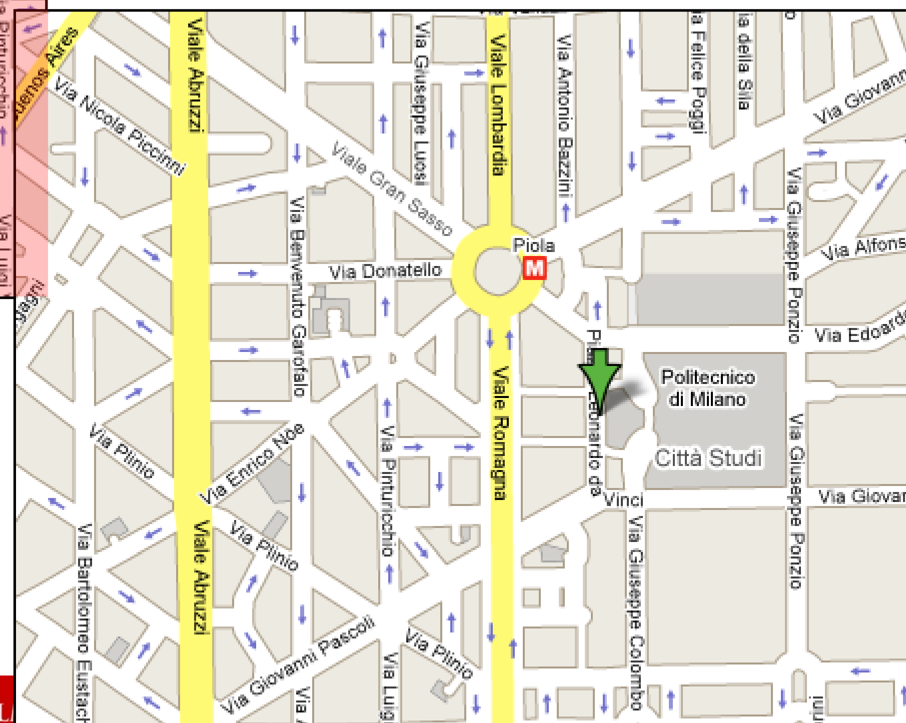
It is not generally possible to describe a manifold with just one chart

A collection of charts $(U_\alpha, \alpha)_{\alpha \in A}$ whose domains cover the manifold is called an **atlas**

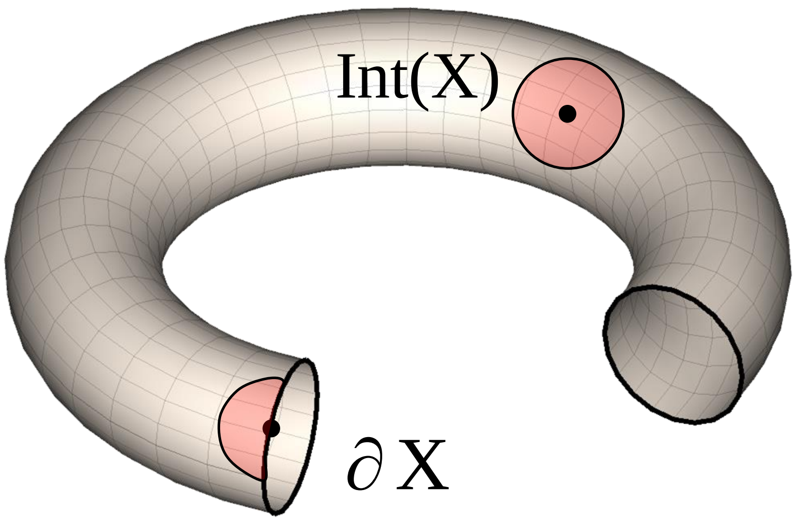
$$\bigcup U_\alpha = \mathcal{M}$$

Charts and atlases





Manifolds with boundary



A topological space in which every point has an open neighborhood homeomorphic to either

- topological disc $\mathbb{R}^n$; or
- topological half-disc $[0, \infty) \times \mathbb{R}^{n-1}$

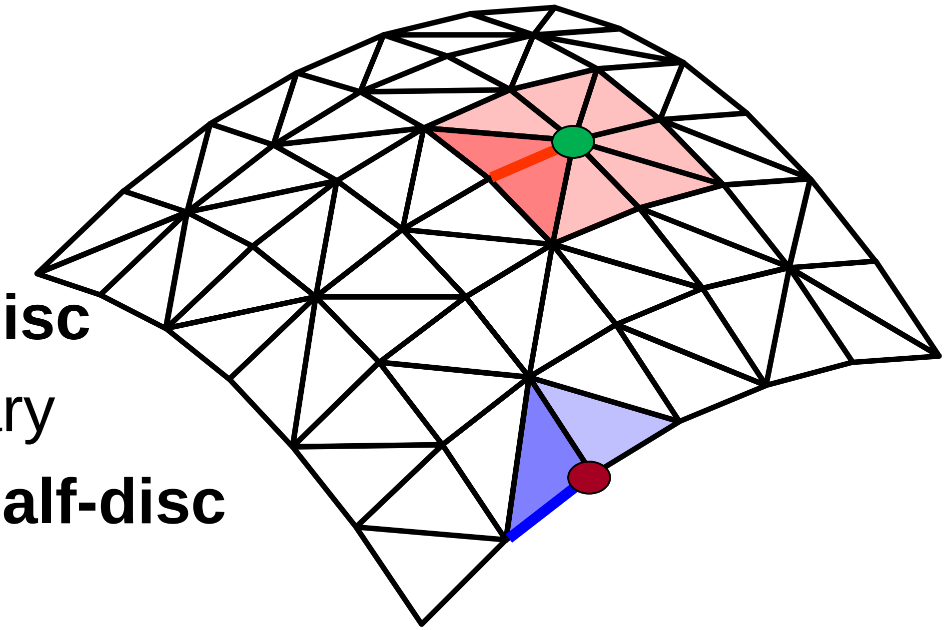
is called a manifold with boundary

Points with disc-like neighborhood are called **interior,** denoted by $\mathrm{Int}(X)$

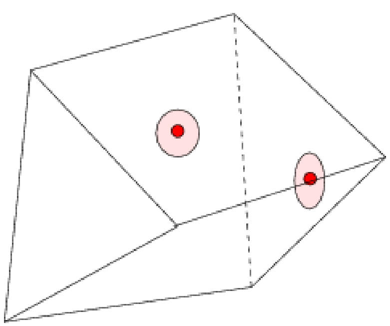
Points with half-disc-like neighborhood are called **boundary**, denoted by ∂X

Manifold meshes

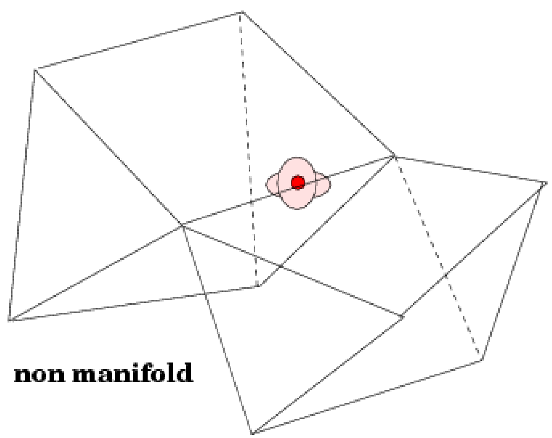
A mesh is a 2-**manifold if**



- Neighborhood of each interior vertex is homeomorphic to a **disc**
- Neighborhood of each boundary vertex is homeomorphic to a **half-disc**



2D manifold



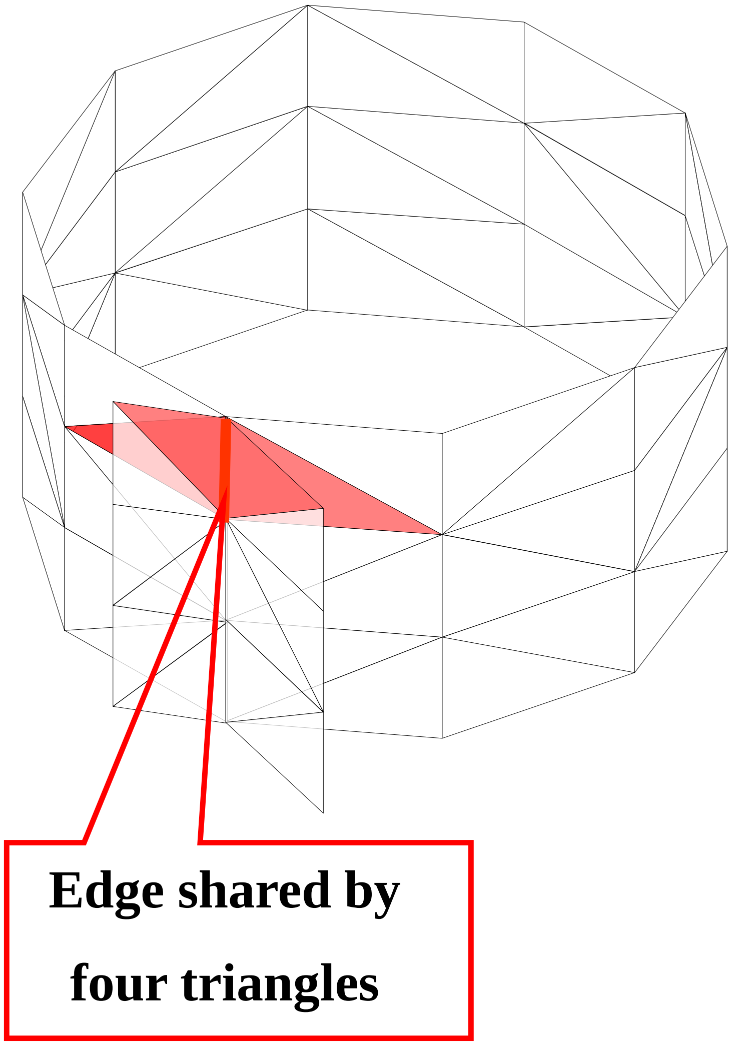
non manifold

Two kinds of vertices allowed:

Internal

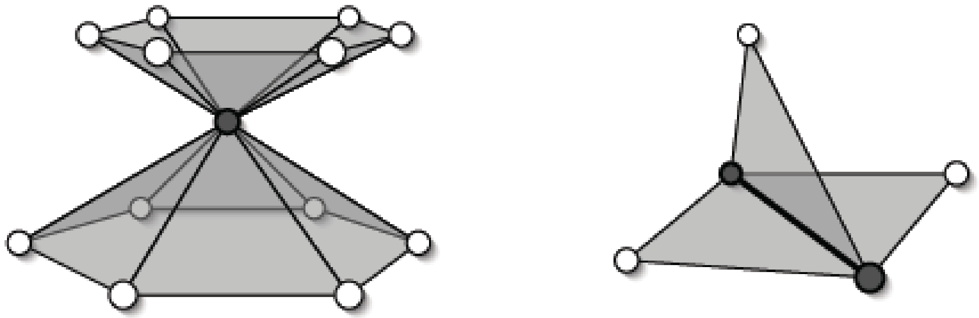
Boundary

Non-manifold meshes



A triangle mesh is a 2-manifold if it contains neither **non-manifold edges** nor **non-manifold vertices** nor **self-intersections**

- **non-manifold edge** has more than two incident triangles
- **non-manifold vertex** is incident to more than one fan of triangles



Representing meshes

- Discrete surface representation
- The mesh is a piece-wise planar approximation obtained by gluing the polygonal faces together,

Connectivity (Topological) data

The mesh is a purely **topological** object and does not contain any geometric properties

Geometric data

The **geometric realization** of the mesh is defined by specifying the coordinates of the vertices $X_i \in R^3$ for all $i \in V$

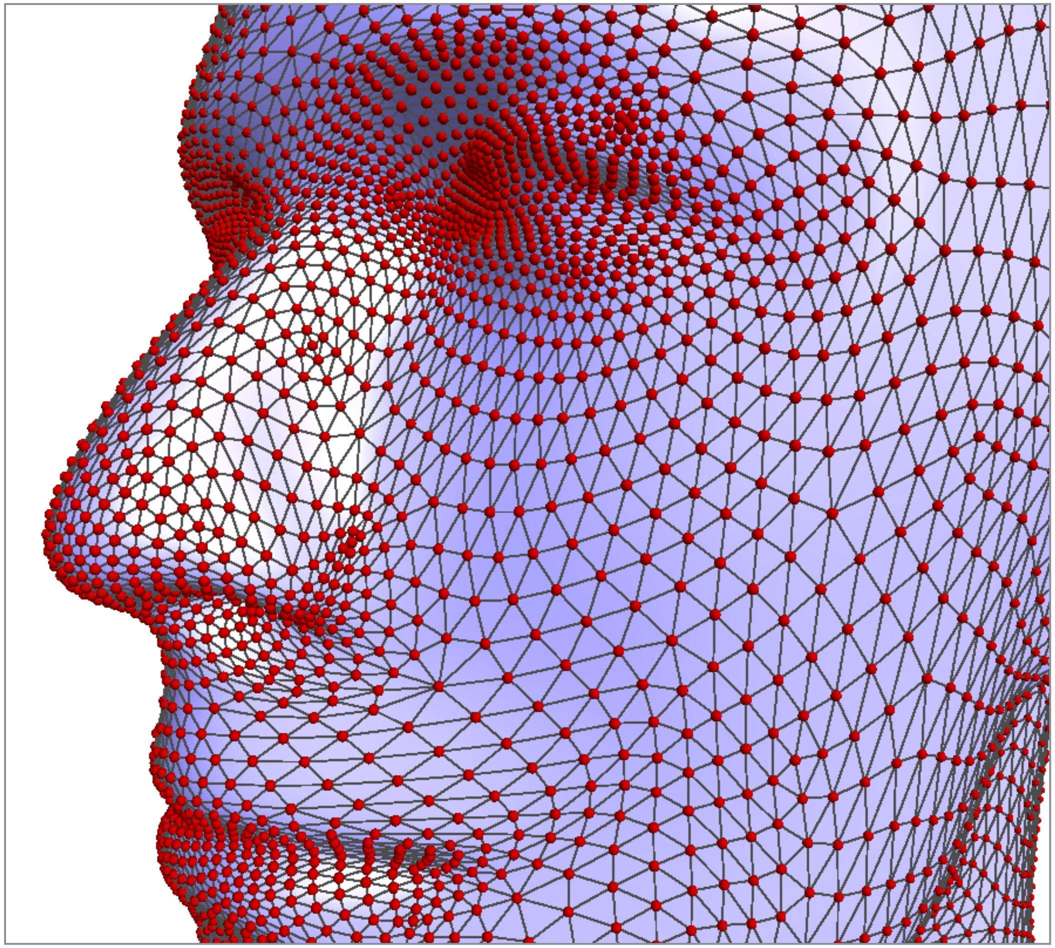
Orientation data

Representing meshes

- Geometry:
  - Vertex Coords

    (x_1, y_1, z_1)
    (x_2, y_2, z_2)
    . . .
    (x_n, y_n, z_n)
- Orientation
  - List of normals
- Connectivity
  - List of triangles

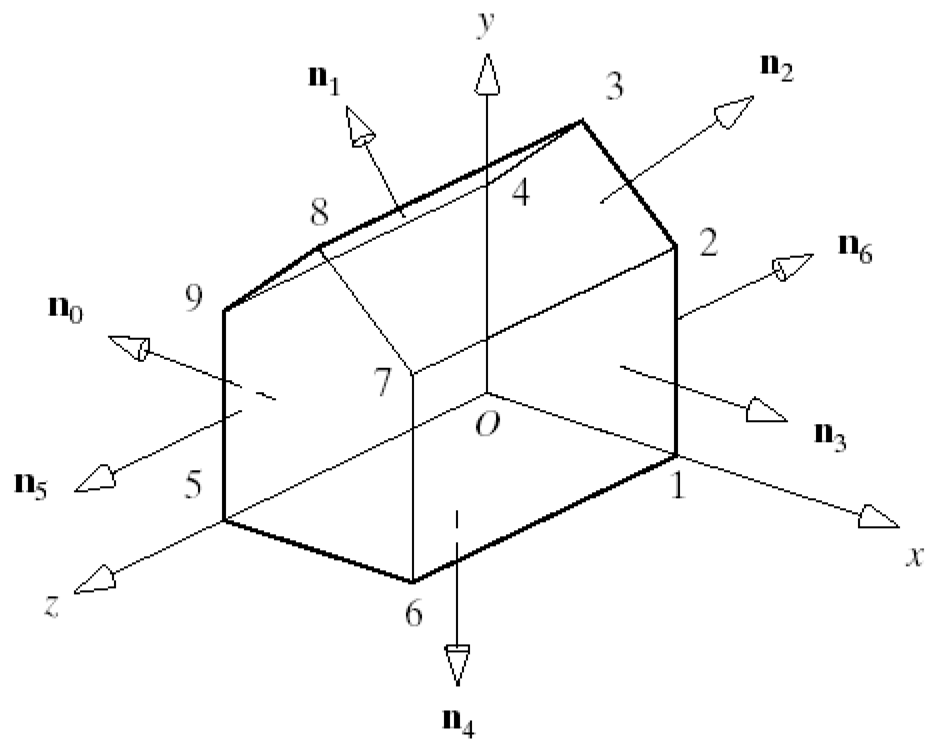
    (i_1, j_1, k_1)
    (i_2, j_2, k_2)
    . . .
    (i_m, j_m, k_m)



Example

normal	n_x	n_y	n_z
0	−1	0	0
1	−0.707	0.707	0
2	0.707	0.707	0
3	1	0	0
4	0	−1	0
5	0	0	1
6	0	0	−1

vertex	x	y	z
0	0	0	0
1	1	0	0
2	1	1	0
3	0.5	1.5	0
4	0	1	0
5	0	0	1
6	1	0	1
7	1	1	1
8	0.5	1.5	1
9	0	1	1



face	vertices	associated normal
0 (left)	0,5,9,4	0,0,0,0
1 (roof left)	3,4,9,8	1,1,1,1
2 (roof right)	2,3,8,7	2,2,2,2
3 (right)	1,2,7,6	3,3,3,3
4 (bottom)	0,1,6,5	4,4,4,4
5 (front)	5,6,7,8,9	5,5,5,5,5
6 (back)	0,4,3,2,1	6,6,6,6,6

Normal Vectors

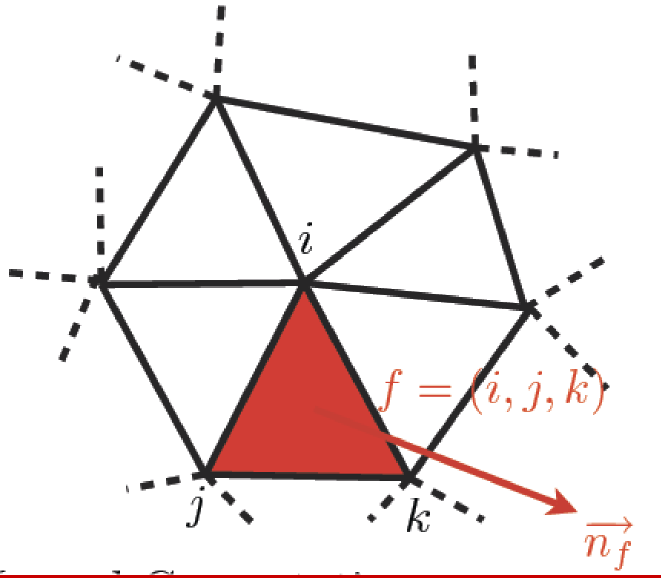
- Local connectivity

Valence of a vertex: number of incident edges at a vertex

Vertex 1-ring $N(i) = \{ j \in V : (i, j) \in E \} \subset V$

Face 1-ring $N(f) = \{(i, j, k) \in F : j, k \in V\} \subset T$

- Normal Computation



Face Normal

$$\forall f = (i, j, k) \in T, \quad n_f = \frac{(X_j - X_i) \times (X_k - X_i)}{\left\|(X_j - X_i) \times (X_k - X_i)\right\|}$$

Vertex Normal

$$\forall i \in V, \quad n_i = \frac{\overline{n_i}}{\left\|\overline{n_i}\right\|} \quad average \;\; \overline{n_i} = \frac{1}{|N(f)|} \sum_{f \in N(f)} n_f$$

Vertex Normal

Max-Nelson's Method

Take into account the edge lengths

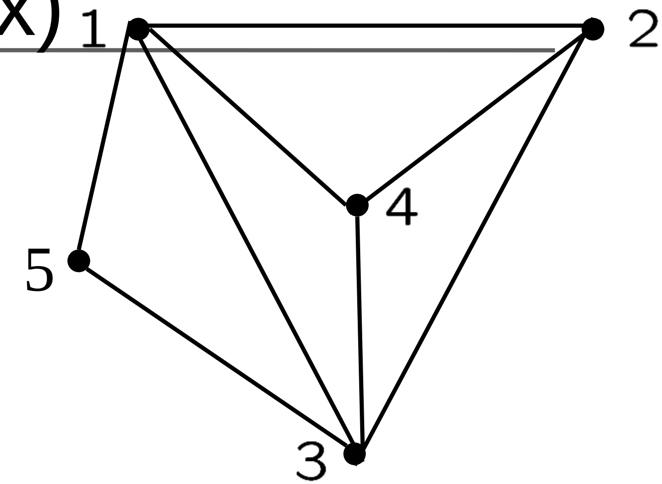
– N(i) number of edges that share the vertex
– (i+1) mod N(i) next edge after i
– Compute the normal vector of vertex i:

$$\forall i \in V, \quad n_i = \frac{1}{|N(f)|} \sum_{j \in N(i)} \frac{e_j \times e_{j+1}}{\|e_j\|^2 \|e_{j+1}\|^2}$$

$$e_j = X_j - X_i$$

Connectivity matrix

(Discrete Laplacian matrix)



Connectivity graph can be represented as a matrix L with dimension N_v x N_v

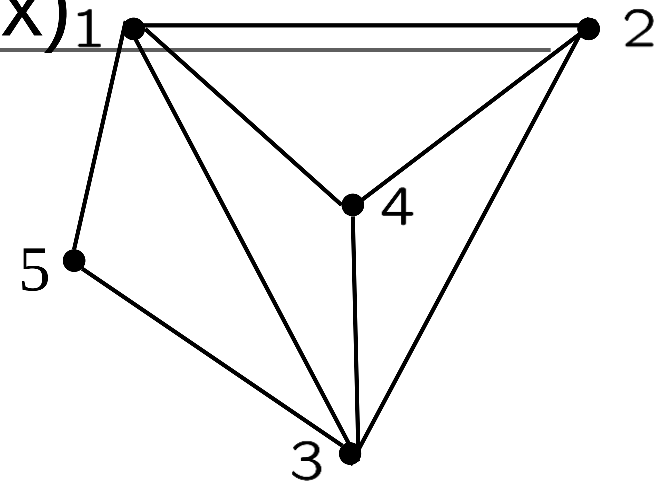
$$L_{ij} = \begin{cases} -1 & i = j \\ \lambda_{ij} & (i,j) \in E \quad \text{(neighbors)} \\ 0 & \text{otherwise} \end{cases}$$

$$\boldsymbol{L} = \begin{pmatrix} -1 & \lambda_{1,2} & \lambda_{1,3} & \lambda_{1,4} & \lambda_{1,5} \\ \lambda_{2,1} & -1 & \lambda_{2,3} & \lambda_{2,4} & 0 \\ \lambda_{3,1} & \lambda_{3,2} & -1 & \lambda_{3,4} & \lambda_{3,5} \\ \lambda_{4,1} & \lambda_{4,2} & \lambda_{4,3} & -1 & 0 \\ \lambda_{5,1} & 0 & \lambda_{5,3} & 0 & -1 \end{pmatrix}$$

choice of weights λ_{ij}

Connectivity matrix

(Discrete Laplacian matrix)



$$L_{ij} = \begin{cases} -1 & i = j \\ \lambda_{ij} & (i, j) \in E \quad \text{(neighbors)} \\ 0 & \text{otherwise} \end{cases}$$

choice of weights λ_{ij} satisfying:

$$\lambda_{i,i} = -\sum_{j \neq i} \lambda_{i,j},$$

$$\lambda_{i,j} > 0 \quad , \sum_{j \in N(i)} \lambda_{i,j} = 1$$

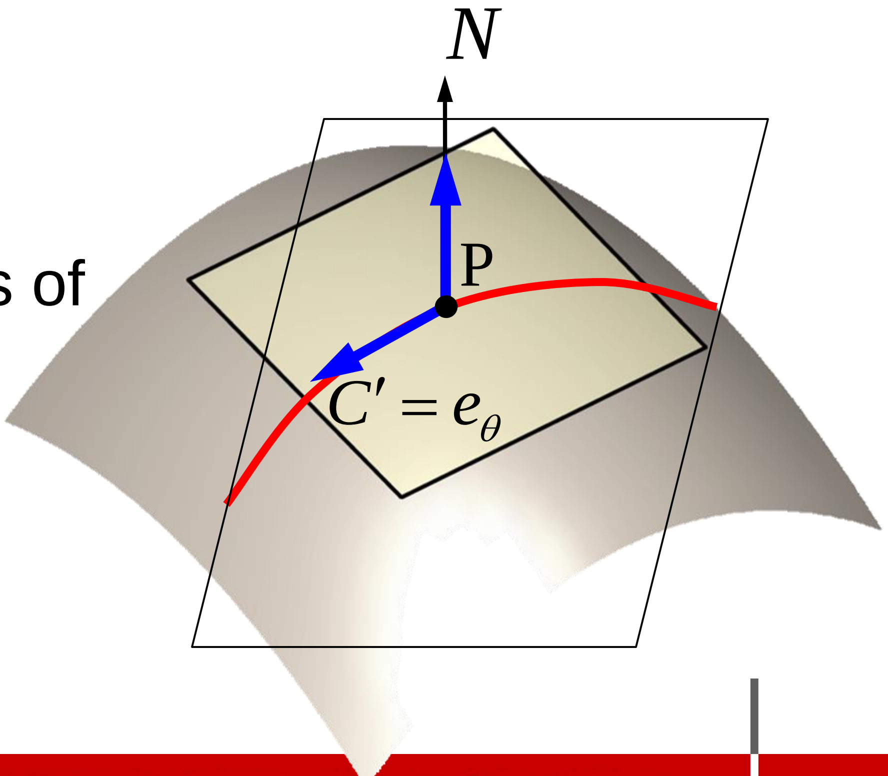
$$\boldsymbol{L} = \begin{pmatrix} -1 & \lambda_{1,2} & \lambda_{1,3} & \lambda_{1,4} & \lambda_{1,5} \\ \lambda_{2,1} & -1 & \lambda_{2,3} & \lambda_{2,4} & 0 \\ \lambda_{3,1} & \lambda_{3,2} & -1 & \lambda_{3,4} & \lambda_{3,5} \\ \lambda_{4,1} & \lambda_{4,2} & \lambda_{4,3} & -1 & 0 \\ \lambda_{5,1} & 0 & \lambda_{5,3} & 0 & -1 \end{pmatrix}$$

Curvature on surface

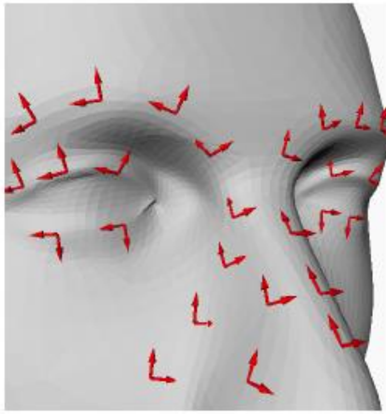
- For a given point P each direction e_Θ in the tangent plane T_pS defines a curve C (normal section) as the intersection between the plane containing N, e_Θ and the surface S

- Curves passing in different directions have different values of **normal curvatures**

$$\kappa_N(\theta) = \langle N, C'' \rangle$$



Principal curvatures

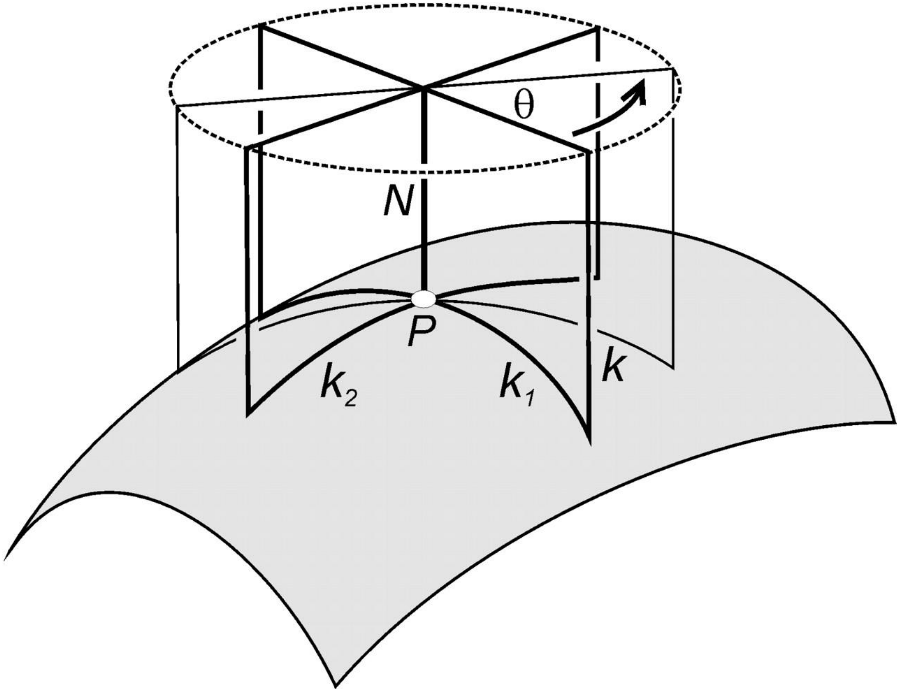


For each direction $e_\theta \in T_pS$, a curve C may have a different normal curvature $\kappa_N(\theta) = \langle N, C'' \rangle$

- A point $p \in S$ has multiple curvatures!
- **Principal curvatures**

$$k_1 = \min_{e_\theta \in T_pS} \kappa_N(\theta), \quad k_2 = \max_{e_\theta \in T_pS} \kappa_N(\theta)$$

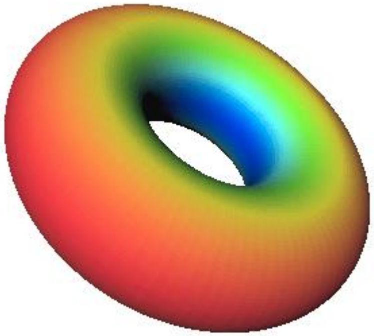
- **Principal directions**

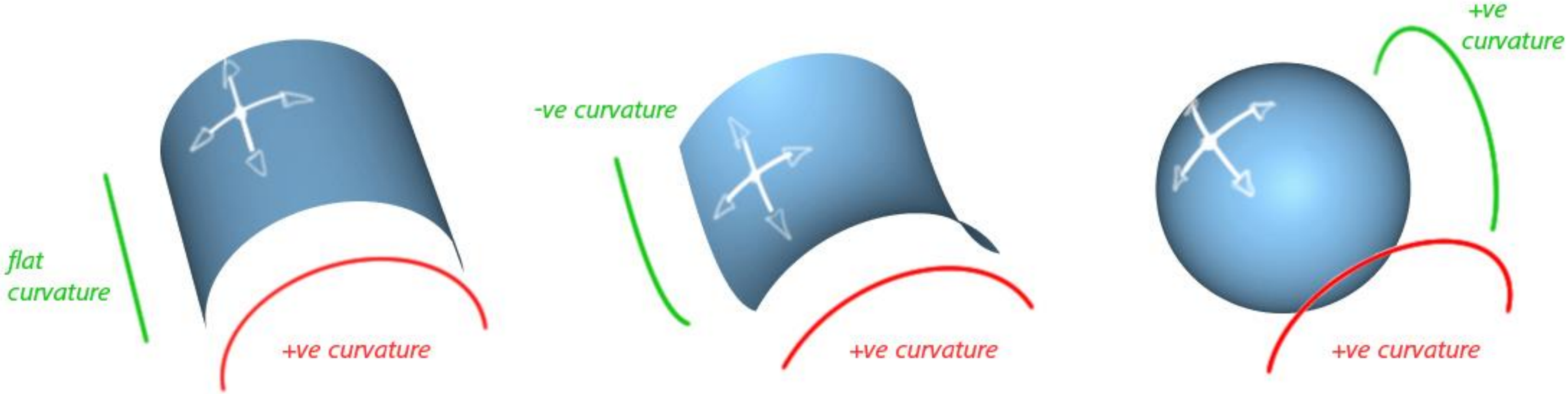


$$e_1 = \arg\min_{e_\theta \in T_pS} \kappa_N(\theta), \quad e_2 = \arg\max_{e_\theta \in T_pS} \kappa_N(\theta)$$

Mean and Gaussian curvatures

- Mean curvature $H = \frac{1}{2}(\kappa_1 + \kappa_2)$
- Gaussian curvature $K = \kappa_1 \kappa_2$





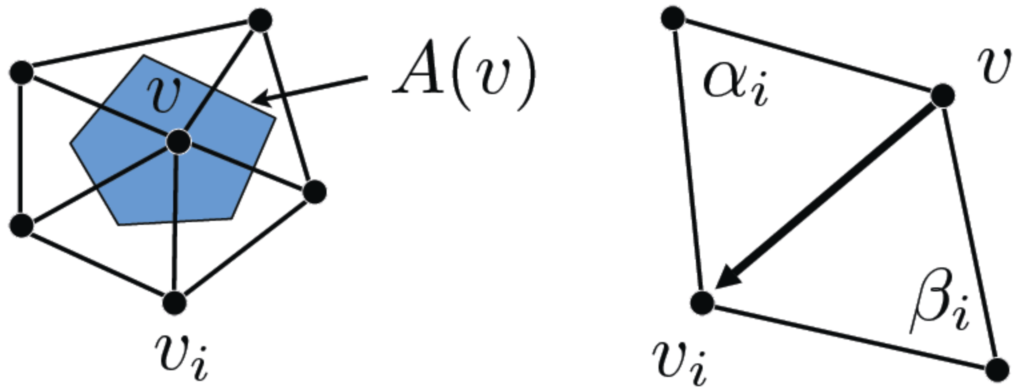
parabolic point $K = 0$

hyperbolic point $K < 0$

elliptic point $K > 0$

K determines if a surface is locally saddle (-) or locally convex (+)

Discrete Mean Curvature



- ***Mean Curvature vector:***

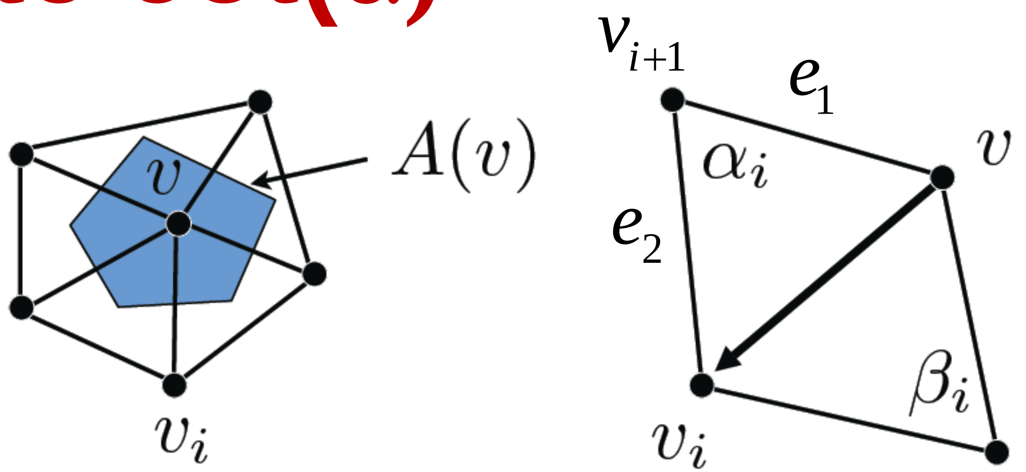
$$\vec{H}(v) = H(v)\vec{N}(v) = \frac{-\Delta(v)}{2} = \frac{1}{2A_v}\sum_{j\in N(v)}(\cot\alpha_j + \cot\beta_j)(v - v_j)$$

  – $\vec{N}(v)$ unit surface normal $\quad \vec{N}(v) = \dfrac{\vec{H}(v)}{\left\|\vec{H}(v)\right\|}$

  – (Scalar) Mean Curvature at v:

$$H(v) = \frac{1}{2A_v}\sum_{j\in N(v)}(\cot\alpha_j + \cot\beta_j)\left\|v - v_j\right\|$$

Compute cot(α)



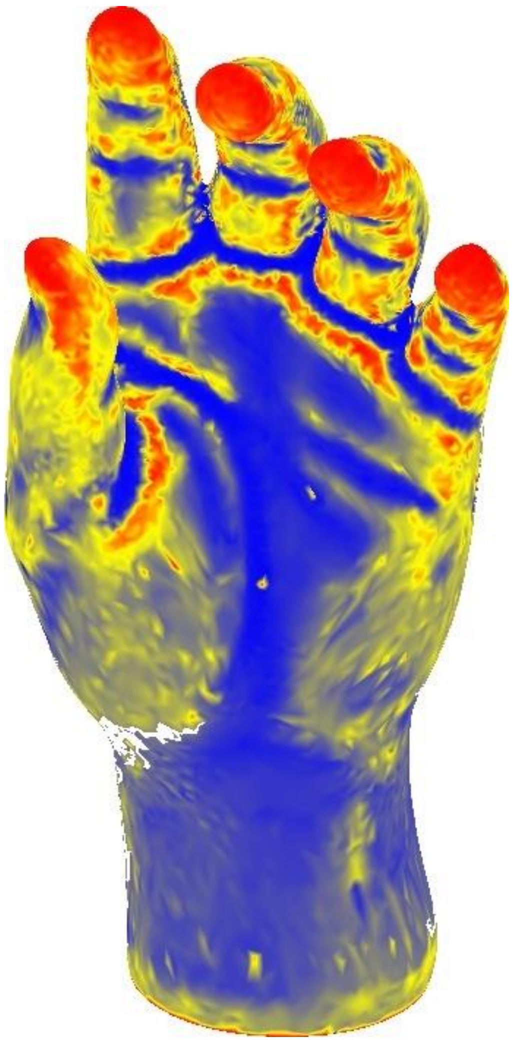
$$\vec{H}(v) = \frac{1}{2A_v} \sum_{j \in N(v)} (\cot \alpha_j + \cot \beta_j)(v - v_j)$$

$$\cot \alpha = \frac{e_1 \cdot e_2}{\sqrt{\|e_1\|^2 \|e_2\|^2 - (e_1 \cdot e_2)^2}}$$

$$e_1 = v - v_{i+1}, \quad e_2 = v_i - v_{i+1}$$

$$Theorem: \quad (\cot(\alpha_{ij}) + \cot(\beta_{ij})) > 0 \quad \leftrightarrow \quad \alpha_{ij} + \beta_{ij} < \pi$$

Discrete Mean Curvature



low curvature

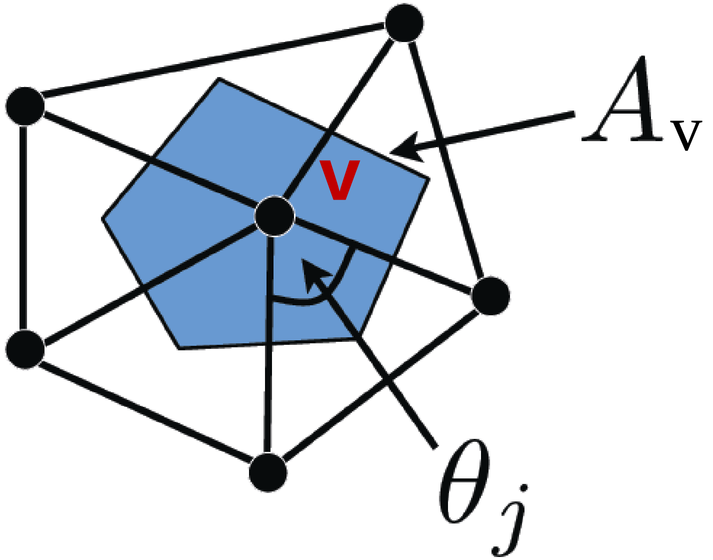
high curvature

False colors visualization

Discrete Gaussian Curvature

Gaussian Curvature at v:

$$K_v = \frac{1}{A_v}\left(2\pi - \sum_{j \in N(v)} \theta_j\right)$$



where θ_j are the incident internal angles at v

- Measure of the distance of a surface (mesh) to a plane
- Depends on angles and lengths
- Zeros curvature in flat areas

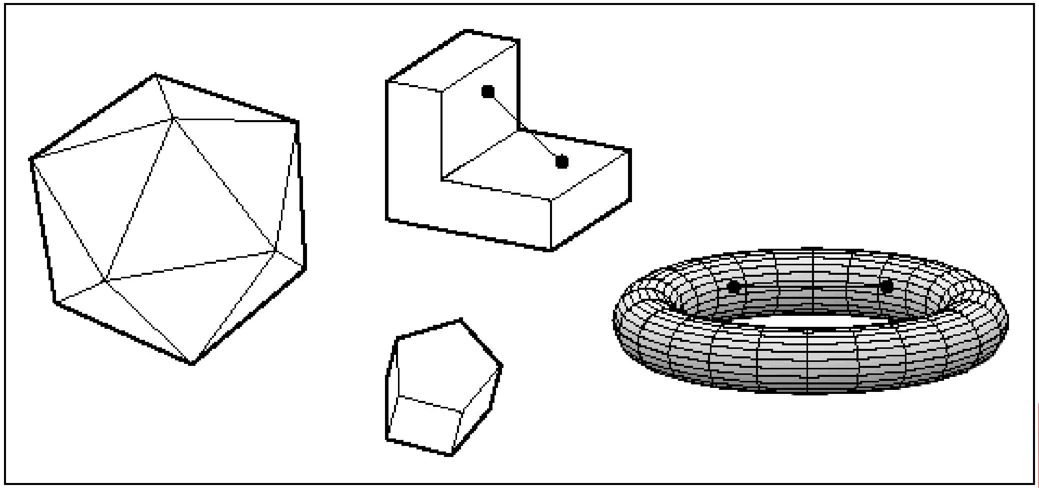
Principal curvatures

$$\kappa_1 = H + \sqrt{H^2 - K}$$

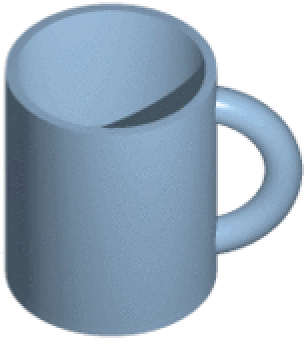
$$\kappa_2 = H - \sqrt{H^2 - K}$$

Properties of a Mesh

- **Solidity** : a mesh represents a solid object if its faces together enclose a positive and finite amount of space.
- **Connectedness** : A mesh is **connected** if an unbroken path along polygon edges exists between any two vertices.
- **Simplicity** : A mesh is **simple** if the object it represents is solid and has no holes through it ; that is, the object can be deformed into a sphere without tearing. Genus = 0
- **Convexity** : The mesh represents a **convex** object if the line connecting any two points within the object lies wholly inside the object.



Surface Genus



In TOPOLOGY, the **genus of a surface is defined as the biggest number of simple, close curves that can be drawn on the surface without splitting it into two non-connected parts**

For orientable surfaces, **the genus counts the number of "handles or holes" of an object**


sphere of genus 0


torus of genus 1


double torus of genus 2

Euler's formula for a mesh without boundaries

Euler's formula provides a fundamental relationship between the number of faces, edges, and vertices for polyhedral in a closed and connected (but otherwise unstructured) mesh.

For a mesh that is not simple:

$$\chi = |V| - |E| + |T| = 2(1-g)$$

is the **Euler Characteristic** and **g** is the GENUS through the polyhedron.

For a simple, solid, connected mesh:

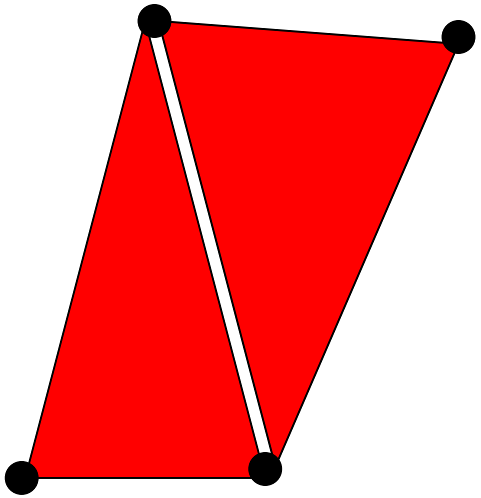
$$\chi = |V| - |E| + |T| = 2$$

$$\chi(cube)??$$

Euler's formula for mesh with Boundaries

$$\chi = |V| - |E| + |T| + |B| = 2(1 - g)$$

|B| # of boundaries

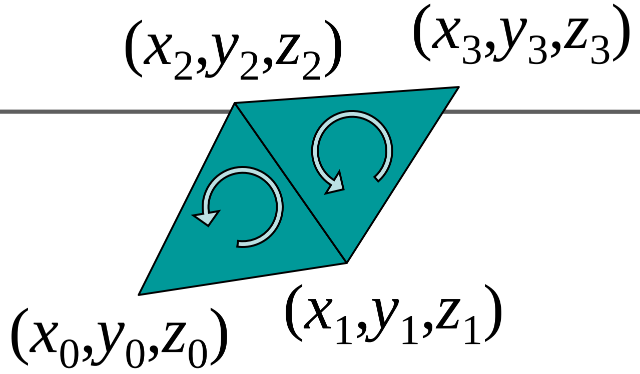


Example:

Genus = 1-(4-5+2+1)/2

= 0

.obj Mesh format file



- Popular file format
  - VRML, Wavefront, etc.
- Ordered list of vertices
  - Prefaced by “v” (Wavefront)
  - Spatial coordinates x,y,z
  - Index given by order
- List of polygons
  - Prefaced by “f” (Wavefront)
  - Ordered list of vertex indices
  - Length = # of sides
  - Orientation given by order

v x_0 y_0 z_0

v x_1 y_1 z_1

v x_2 y_2 z_2

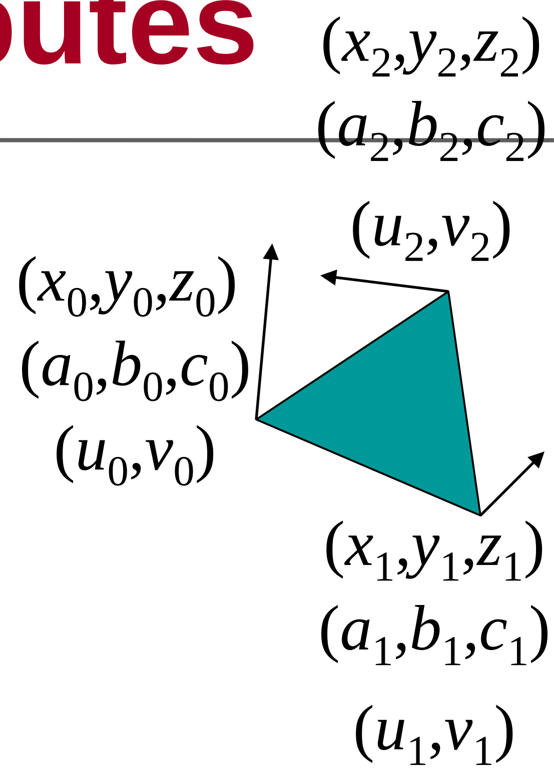
v x_3 y_3 z_3

f 0 1 2

f 1 3 2

Other Attributes

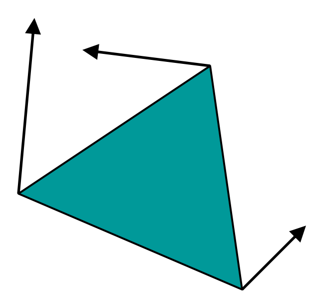
- Vertex normals
  - Prefixed w/ “vn” (Wavefront)
  - Contains x,y,z of normal
  - Not necessarily unit length
  - Not necessarily in vertex order
  - Indexed as with vertices
- Texture coordinates
  - Prefixed with “vt” (Wavefront)
  - Not necessarily in vertex order
  - Contains u,v surface parameters
- Faces
  - Uses “/” to separate indices
  - Vertex “/” normal “/” texture
  - Normal and texture optional
  - Can eliminate normal with “//”



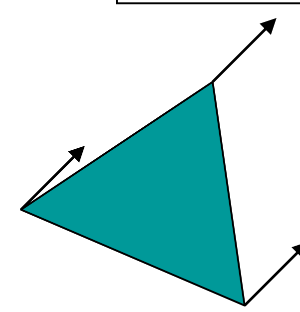
v x_0 y_0 z_0
v x_1 y_1 z_1
v x_2 y_2 z_2

vn a_0 b_0 c_0
vn a_1 b_1 c_1
vn a_2 b_2 c_2

vt u_0 v_0
vt u_1 v_1
vt u_2 v_2



f 0/0/0 1/1/1 2/2/2

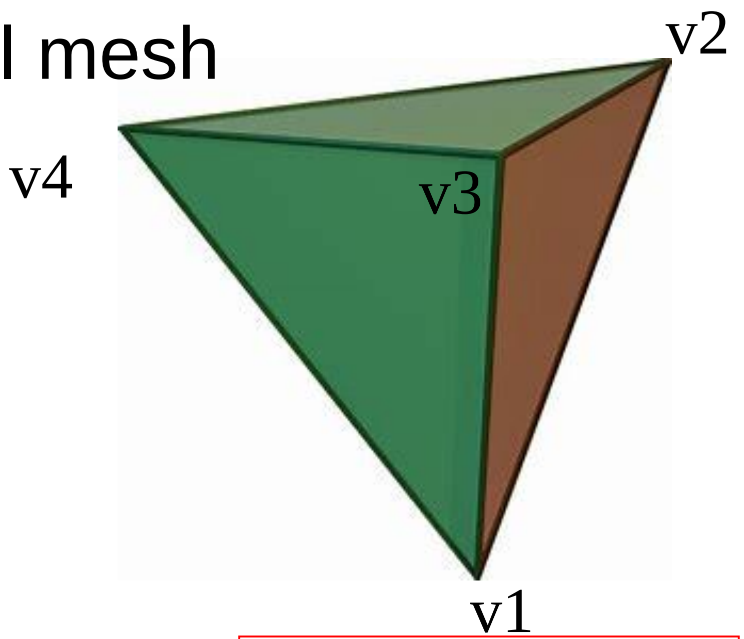


f 0/0/0 1/0/1 2/0/2

Format file .obj (Wavefront)

```
# .obj formato file:  tetraedral mesh
v 1.0 0.0 0.0
v 0.0 1.0 0.0
v 0.0 0.0 1.0
v 0.0 0.0 0.0
f 2 4 3
f 4 2 1
f 1 2 3
f 1 3 4
```

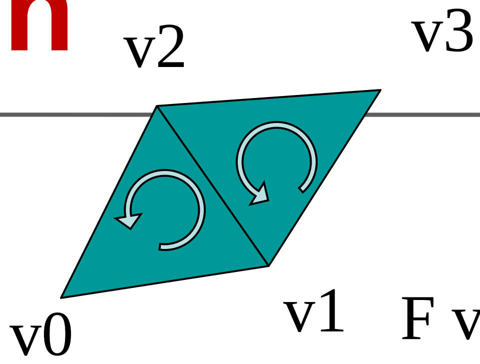
TRI/QUAD Face



v x,y,z	vertex
f v1 v2 v3	face
# comment	

Some applications support vertex colors, by putting red, green and blue values after x y and z.

Mesh: orientation

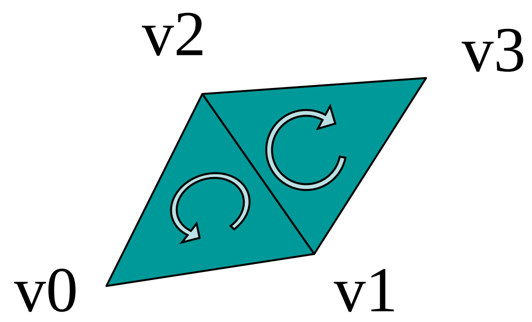


OK

F v0 v1 v2

F v1 v3 v2

- Given by the order of the vertices in a face
- Every poly has counterclockwise outline



Mesh Orientation checking

- If the orientation for an edge is the same for both polys sharing it, then **flip the poly outline**

KO

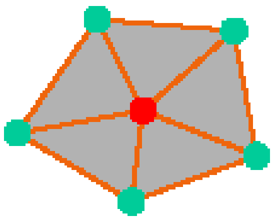
F v0 v1 v2

F v0 v2 v1

Definition

1-ring neighborhood   N(i)

Faces/vertices adjacent at vertex i



Valence of a vertex

Number of incident edges at a vertex

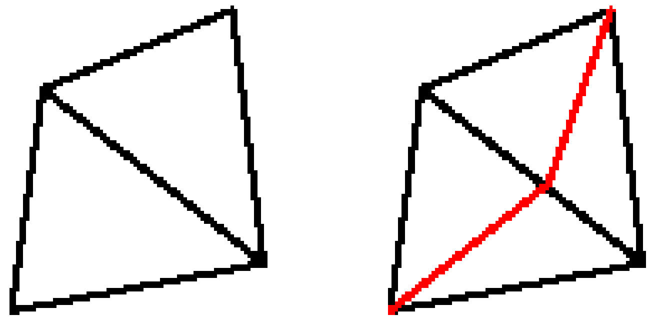
Common Mesh Operations

- *Direct Access to the elements*
- *Ordered Access to the elements* – From an initial elements walk through the mesh elements by adjacent elements (i.e. along edges, faces, …)
- *Topological Relations* – given a face, find out its edges and its vertices. Given a vertex which is the set of incidents elements.

Additional operations

- **Edge Split**

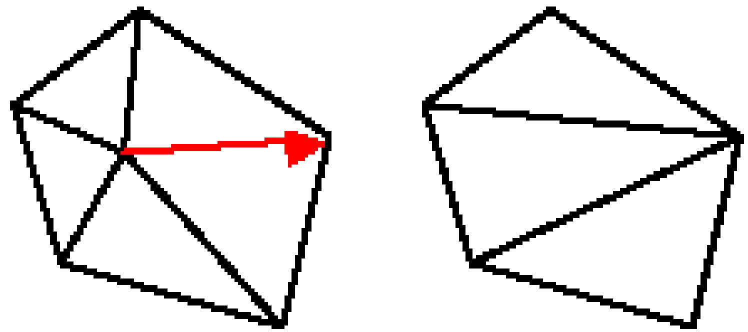
  Adds a vertex to get four triangles



- **Edge collapse**

  Removes an edge

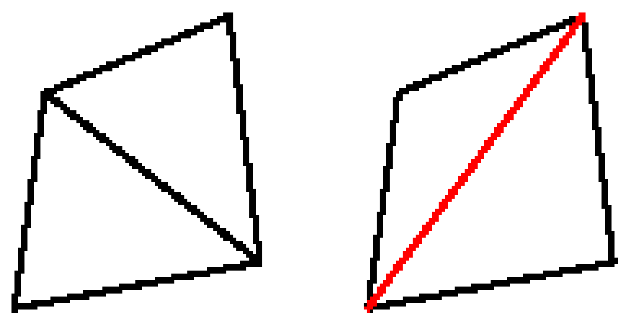
  Removes a vertex



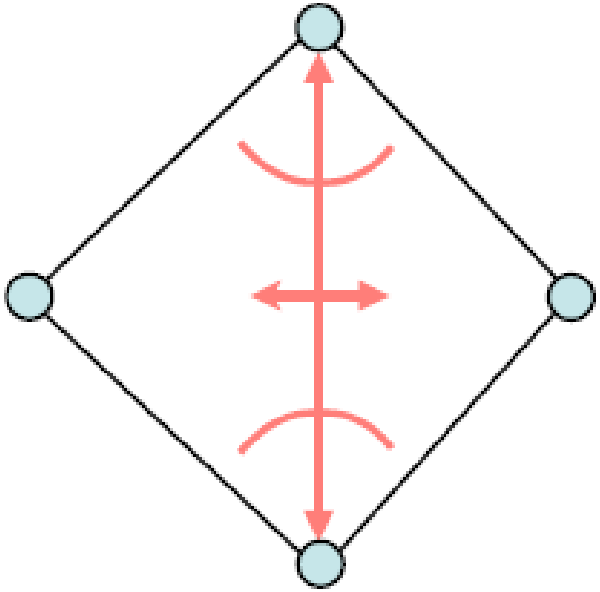
- **Edge Flip**

  Flip an edge

  – beware of valence three

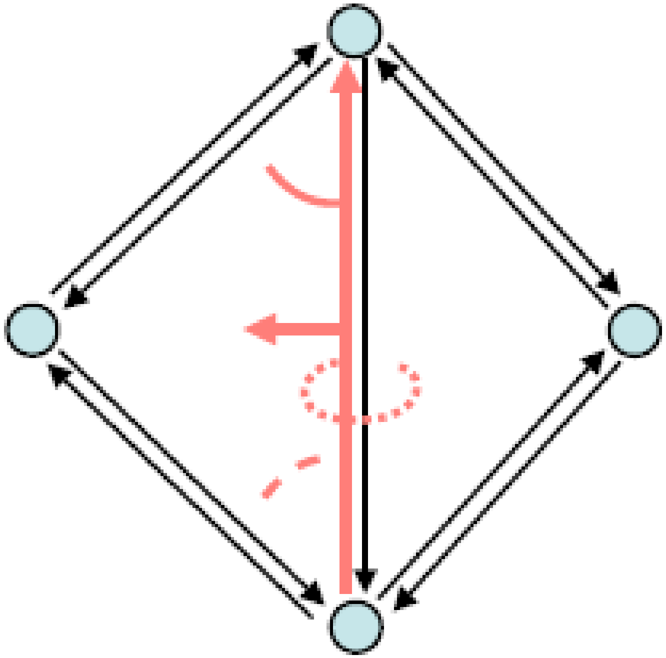


Winged-Edge data structure



- vertex
  – 1 pointer to one edge
- edge pointers to
  – 2 endpoint vertices
  – 2 faces that share edge
  – 4 edges emanating from its endpoints
- face
  – 1 pointer to one edge

Halfedge data structure



- vertex
  - 1 pointer to halfedge
- halfedge pointers to
  - 1 incident vertex
  - 1 incident face
  - 1, 2, or 3 halfedges (prec.,next.,opposite)
- face
  - 1 halfedge (random)

Geometry Mesh Processing

Geometry Processing Pipeline

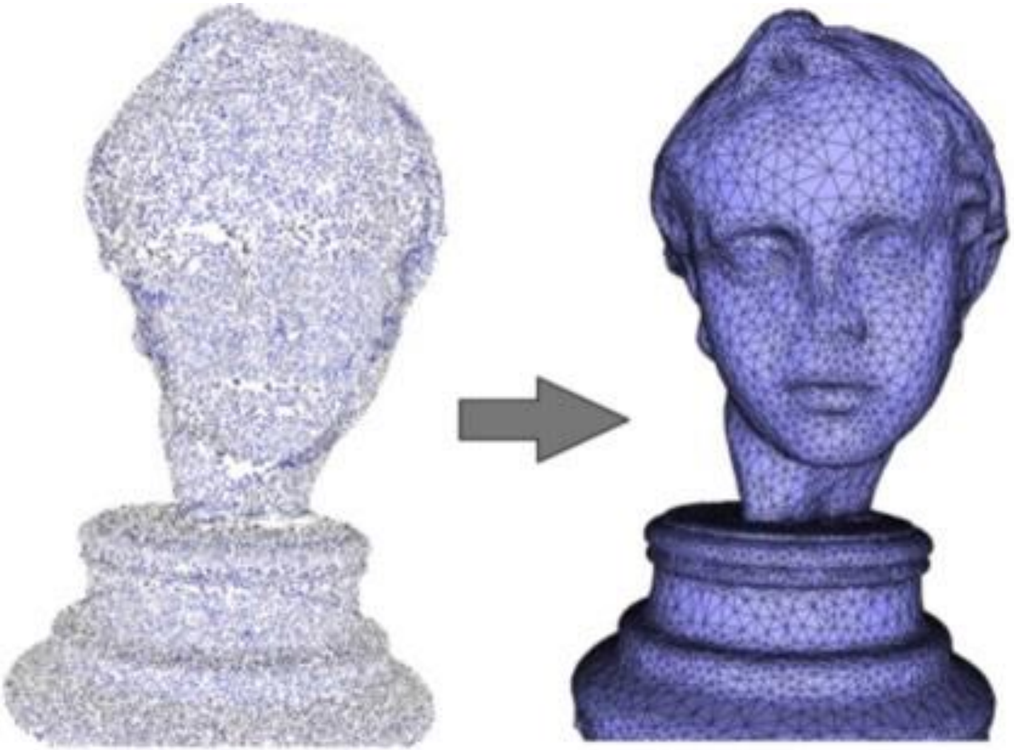
scan

process

print

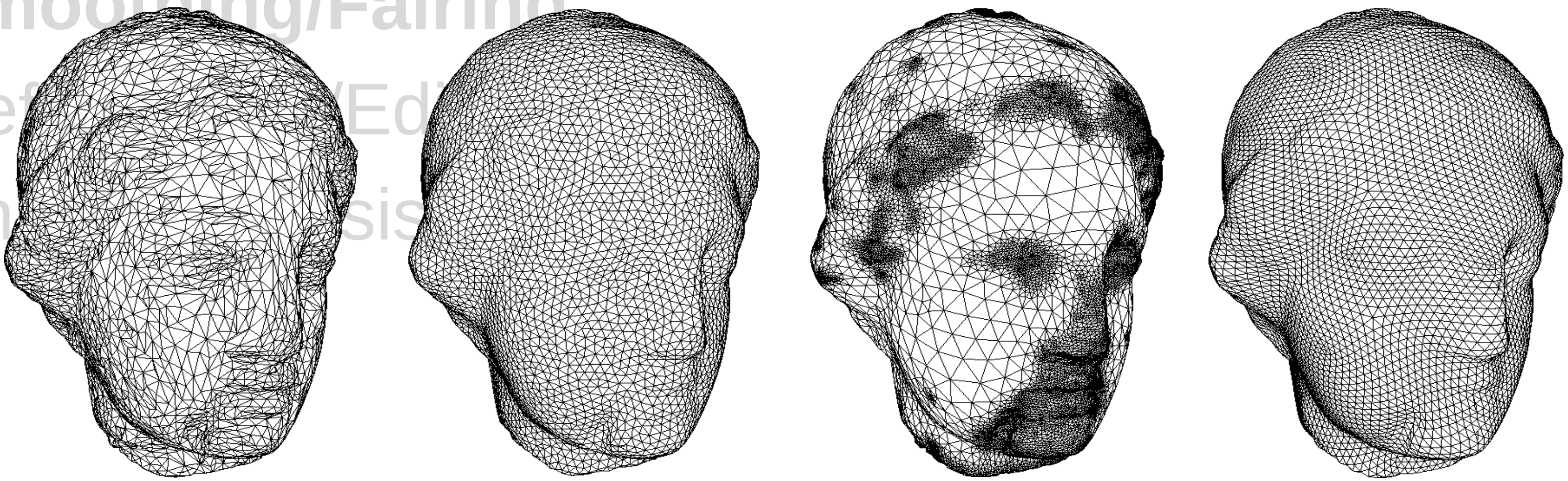
Outline

- **Reconstruction**
- Simplification
- **Parameterization**
- Remeshing
- **Smoothing/Fairing**
- Deformation/Editing
- Shape Analysis



Outline

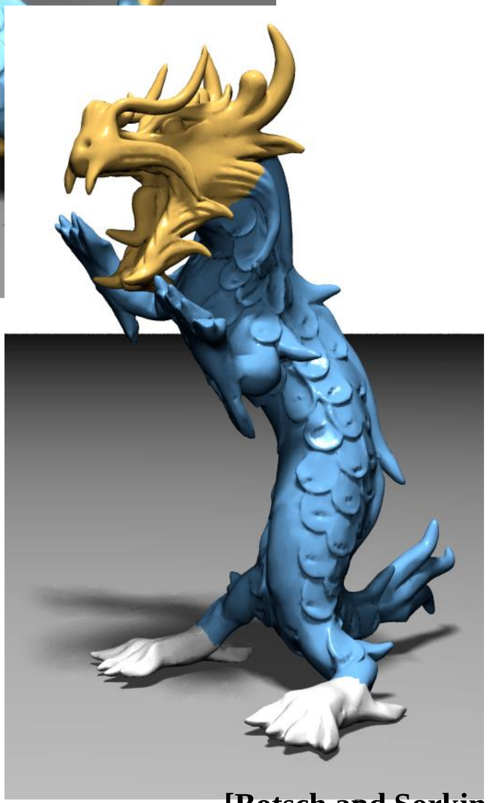
- Reconstruction
- Simplification
- **Parameterization**
- **Remeshing**
- **Smoothing/Fairing**
- Def[illegible]/Edi[illegible]
- Sh[illegible]sis



Original irregular | **uniform** | **adapted** | **semi-regular**

Outline

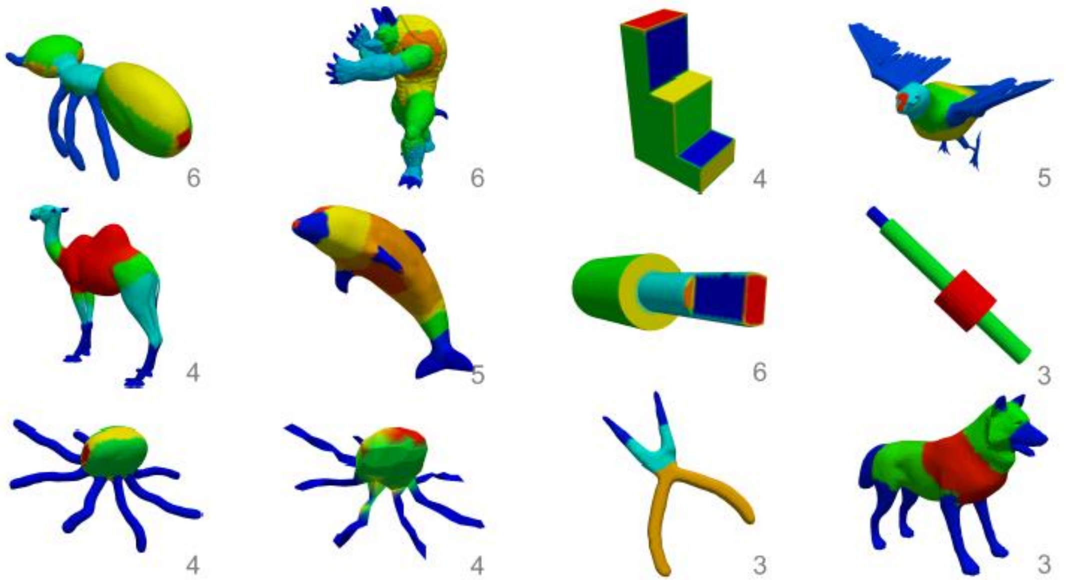
- Reconstrction
- Simplification
- **Parameterizatio**
- Remeshing
- **Smoothing/Fairi**
- **Deformation/Editing**
- Shape Ana



[Botsch and Sorkine]

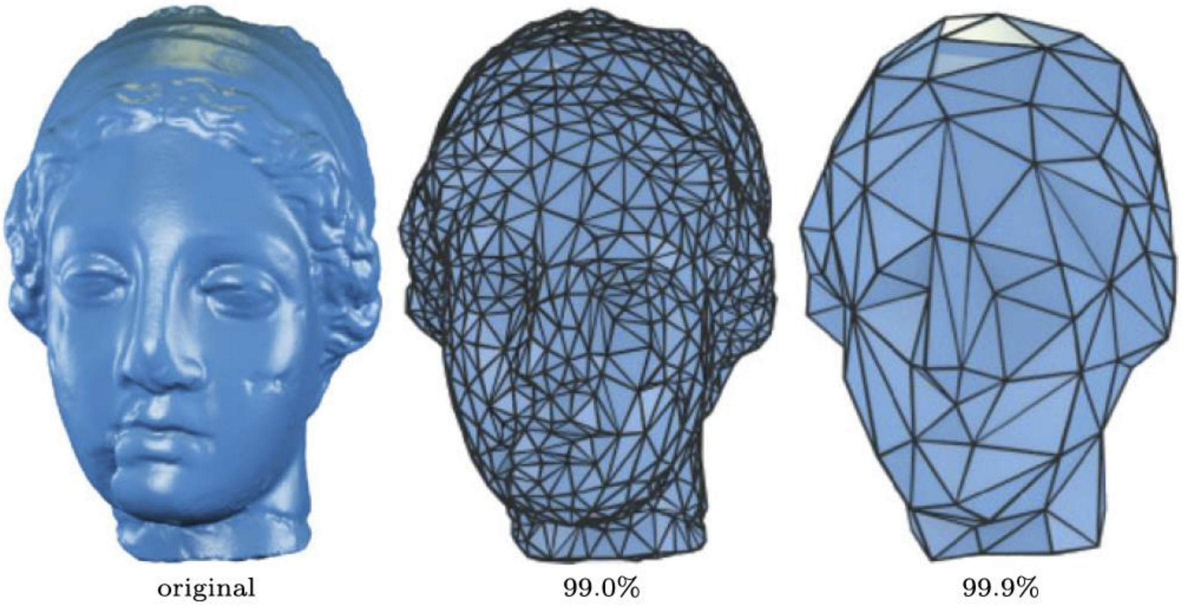
Outline

- Reconstrction
- Simplification
- **Parameterization**
- Remeshing
- **Smoothing/Fairing**
- Deformation/Editing
- **Shape Analysis**
  - Identify/understand important semantic features
  - segmentation, correspondence, symmetry detection, ...



Outline

- Reconstruction
- **Simplification**
- **Parameterization**
- Remeshing
- **Smoothing/Fa**
- Deformation/Ec



Mesh Optimization

PROBLEM:

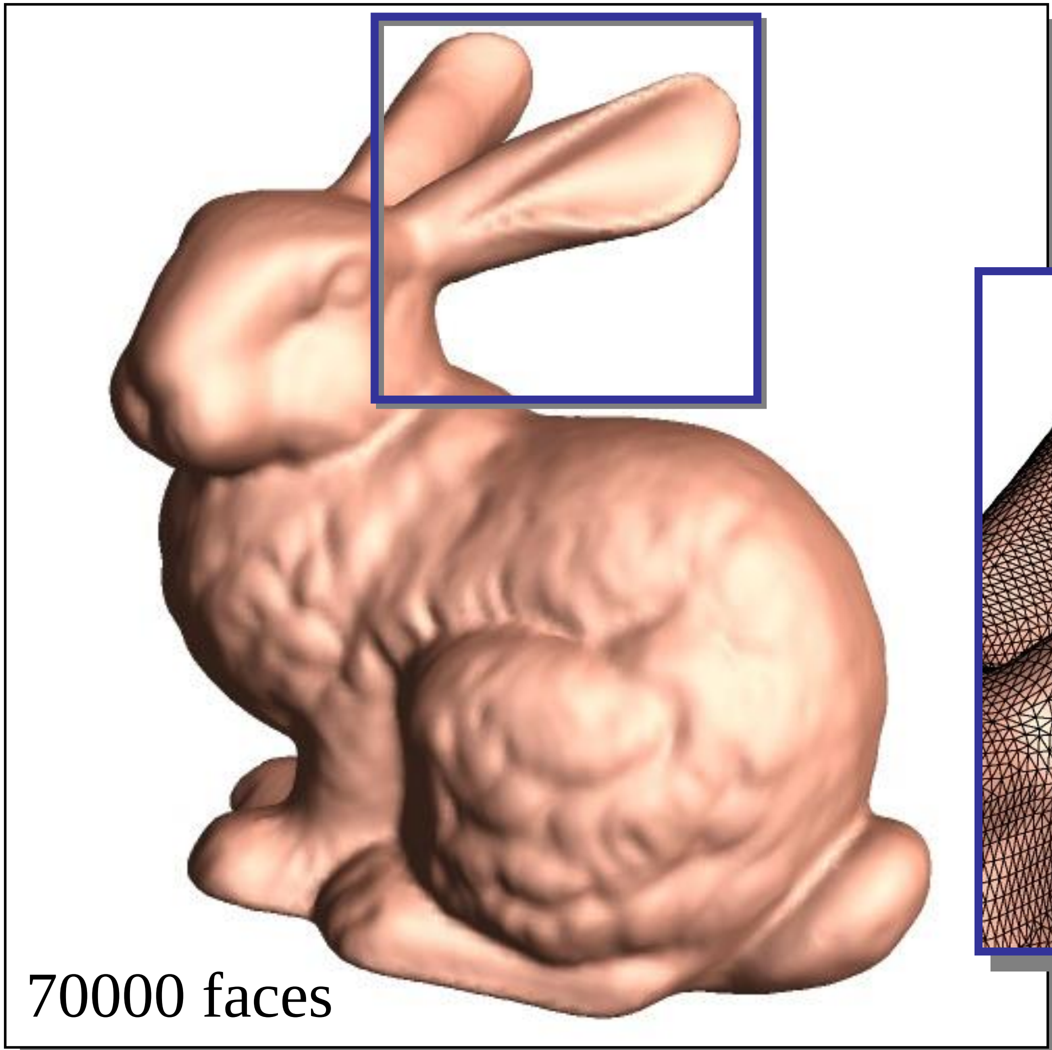
the number of required polys to efficiently represent a complex object is huge and redundant.

SOLUTION: optimize the mesh preserving its topology

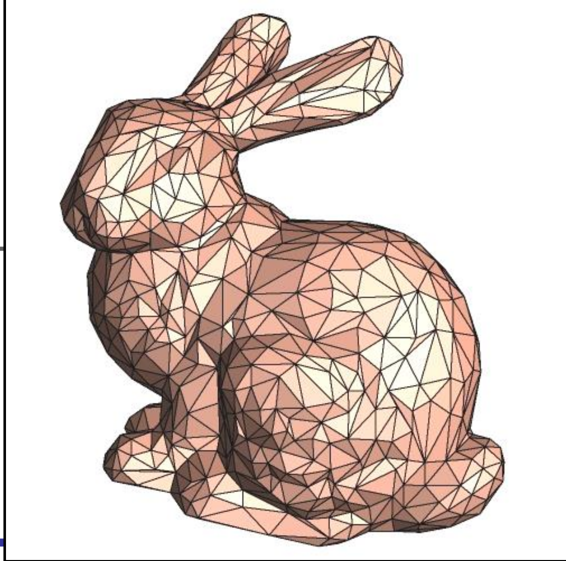
- Produce approximations with fewer triangles
  - should be as similar as possible to original
  - want computationally efficient process
- Speed up Rendering
- Less storage
- Easier processing

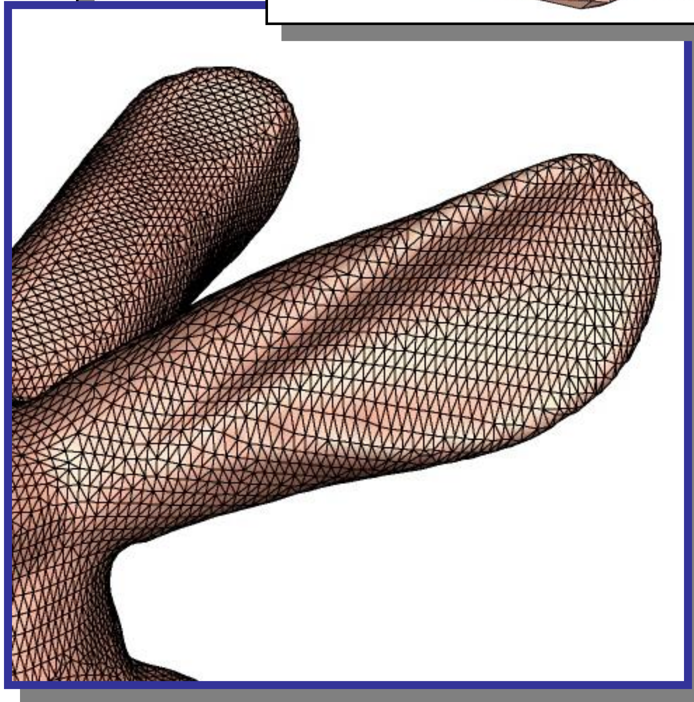
Applications

Oversampled 3D scan data



70000 faces





H.Hoppe

Applications

Level of detail hierarchies

Vertices: 81457

Faces:162910

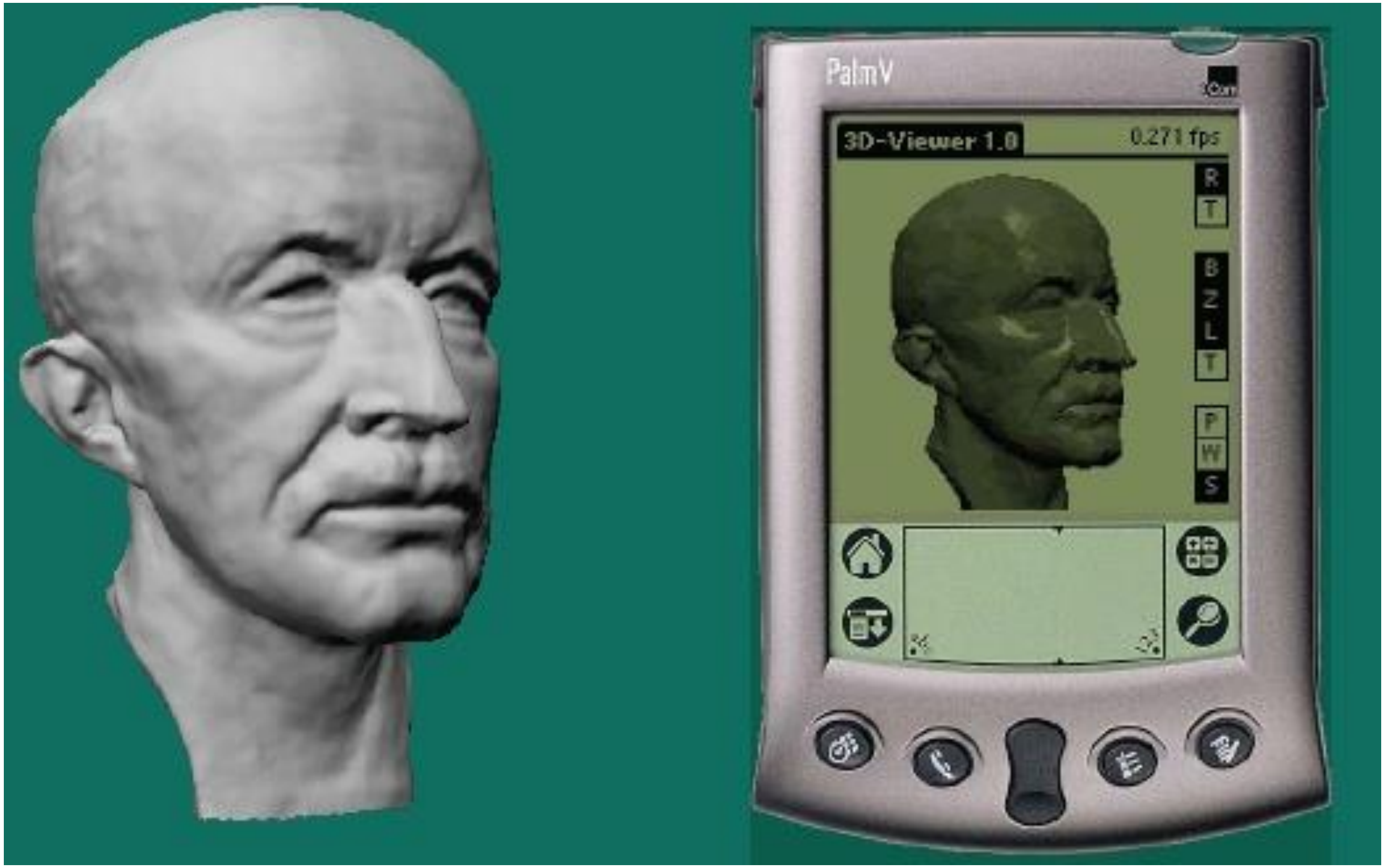
Vertices 60K

Vertices:17K

Vertices: 400

Applications

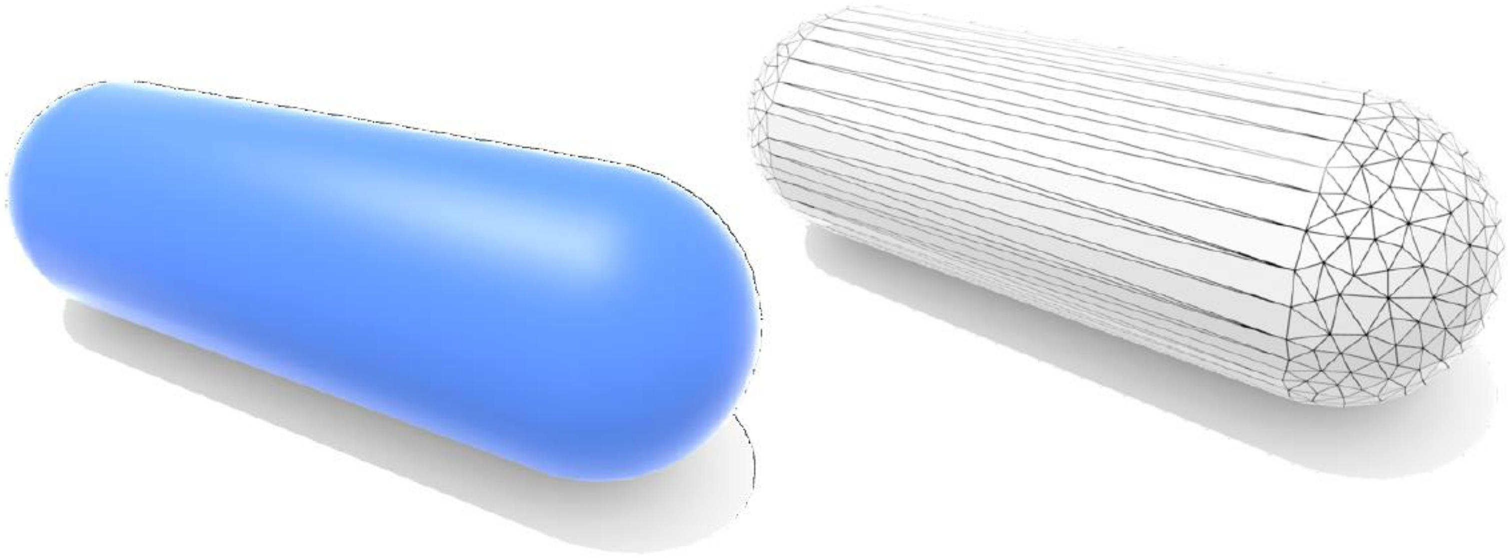
Adaptation to hardware capabilities



Kobbelt,Botsch,2008

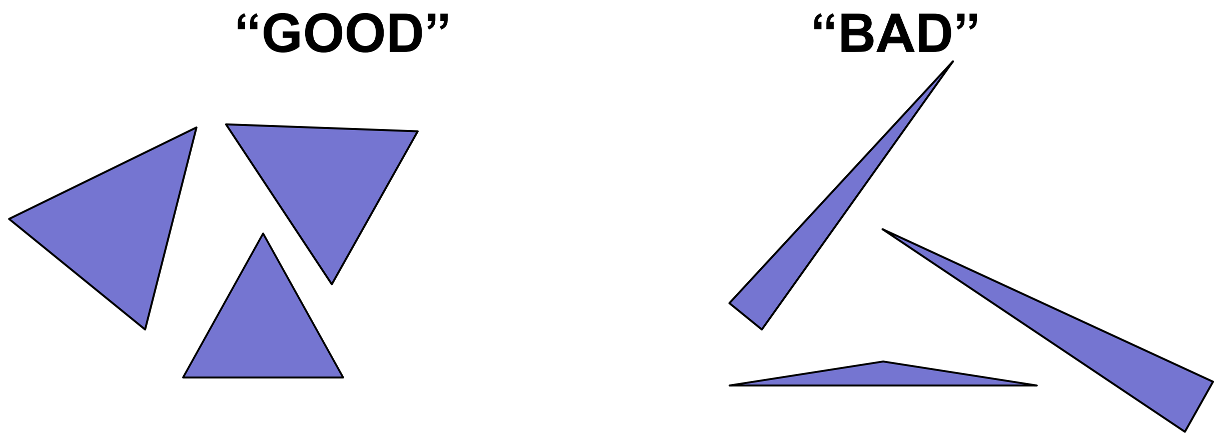
What makes a "good" mesh?

- Good approximation of original shape!
- Keep only elements that contribute information about shape
- Add additional information where, e.g., curvature is large



What else makes a "good" triangle mesh?

- Another rule: *triangle shape*



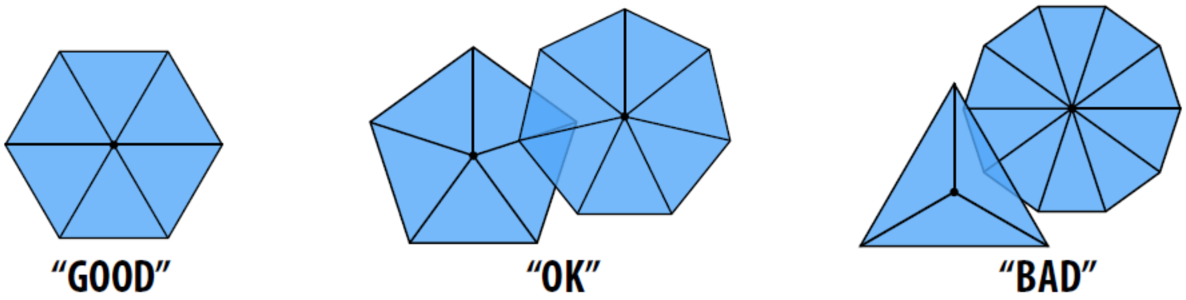
- E.g., all angles close to 60 degrees
- More sophisticated condition: *Delaunay*
- Can help w/ numerical accuracy/stability
- Tradeoffs w/ good geometric approximation*

***See Shewchuk, "What is a Good Linear Element"**

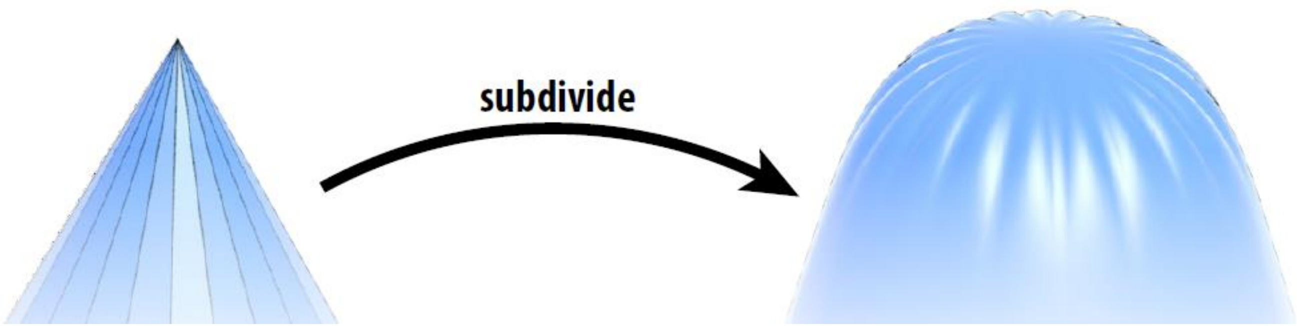
What else constitutes a good mesh?

- Another rule: regular vertex degree

E.g., valence 6 for triangle meshes (equilateral)



- Why? Better polygon shape, important for (e.g.) subdivision:



- FACT: Can't have perfect valence everywhere!

Mesh Optimization

PROBLEM STATEMENT:

Given: 3D model M = (V, E,F)

Point samples V={Pi}, Mesh connectivity F={Tj}

Find: 3D model M' = (V', F') such that

– # V' << #V

– ||M – M'|| < ε

Most common kinds of optimization methods:

1) Vertex clustering decimation

2) Incremental Decimation

Incremental mesh Decimation (data reduction)

ALGORITHM:

Input M (original model)

For each geometric item (edge/face)

- rank all geometric item with some cost metric
- sort for increasing cost

Repeat

- contract minimum cost geometric item by decimation
- update geometric item list of costs

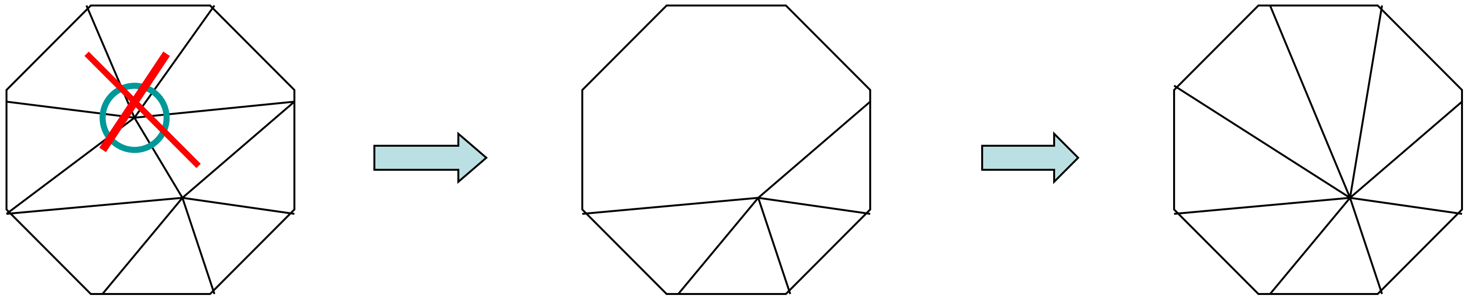
until (no further reduction possible)

Output M’ (simplified model)

Decimation Operators: vertex remove

Remove of the vertex

- Remove associated triangles
- Triangulate the hole



Decimation Operators: edge collapse

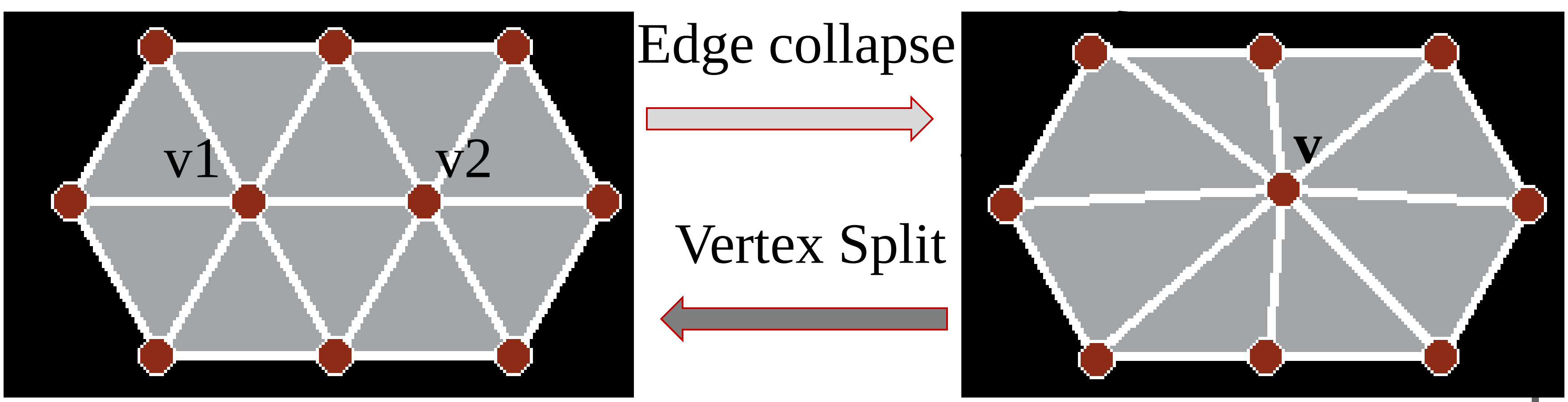
Edge collapse: (v1,v2) ⟹ v (based on appropriate rule *like average*)

removes:

- 1 vertex
- 3 edges
- 2 triangles

Vertex split:

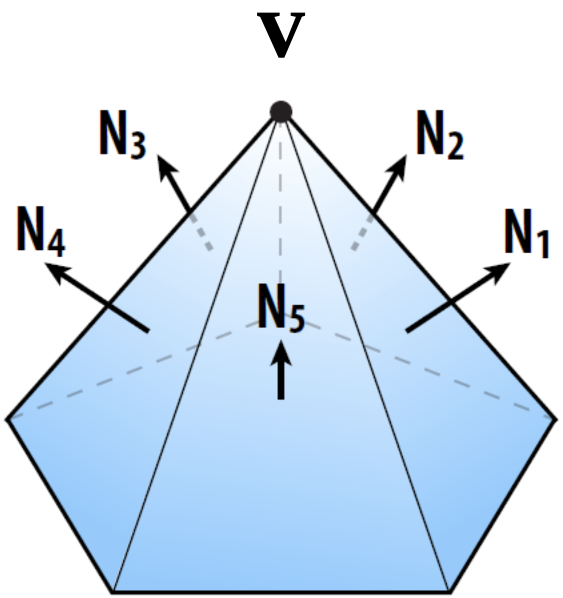
Dual of edge collapse, adds 2 triangles

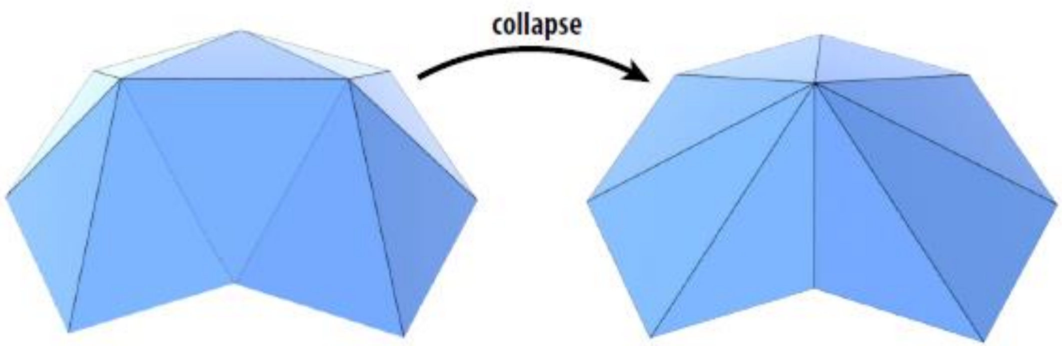


Quadric Error Metric

Garland & Heckbert, SIGGRAPH 97 (DirectX)

- Based on point-to-plane distance
- The set of planes at a vertex is initialized to be the planes of the triangles that meet at that vertex.





Quadric Error Metric

- The error of the vertex v with respect to the set of planes is the sum of squared distances from vertex to planes:

$$\Delta_v = \sum_p Dist(v, p)^2$$

$$v = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \qquad p = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$$

***Signed* distance to plane:**

$$Dist(v, p) = ax + by + cz + d = p^T v$$

Quadric Error Metric

$$\Delta_v = \sum_p (p^T \mathrm{v})^2$$

$$= \sum_p v^T p p^T v$$

$$= v^T \sum_p Q^{(p)} v$$

$$= v^T Q v$$

Error metric rewritten as a quadratic form:

symmetric 4x4 matrix Q

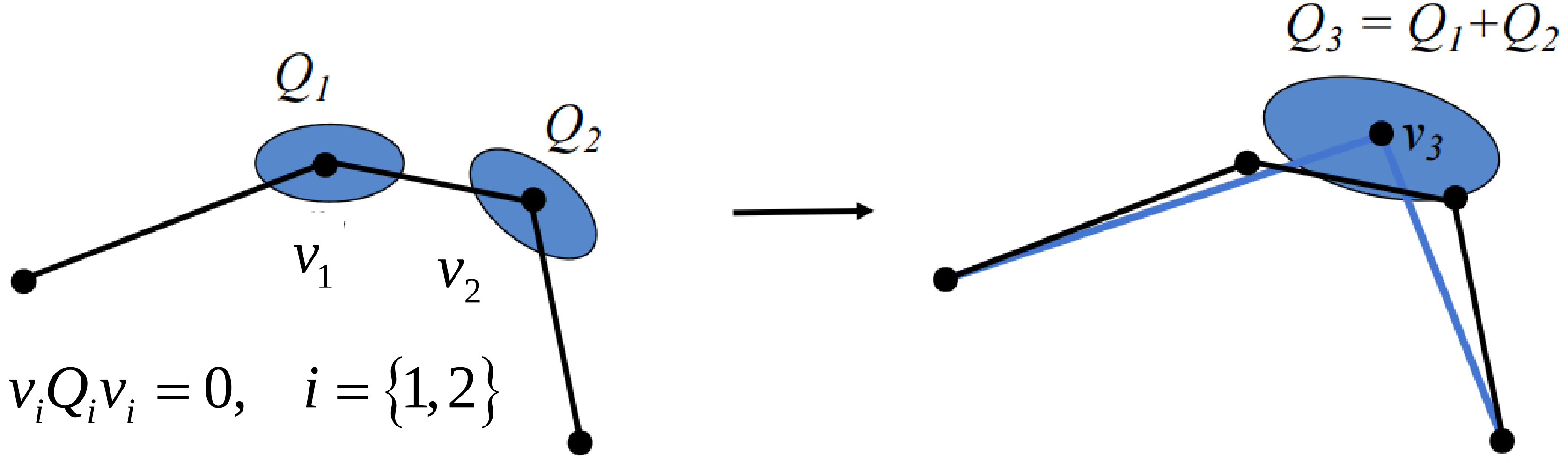
multiplied twice by a vector

$$Q^{(p)} = \begin{pmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{pmatrix}$$

- Compute $Q^{(p)}$ for each triangle (distance to plane p)
- Set Q at each vertex to sum of Qs from incident triangles

Using Quadrics

- Approximate error of edge collapses
  - Each vertex v has associated quadric Q
  - Error of collapsing v_1 and v_2 to v_3 is
    $v_3^T Q_1 v_3 + v_3^T Q_2 v_3$
  - Quadric for new vertex v_3 is $Q_3 = Q_1 + Q_2$



Using Quadrics

Find optimal location v_3 after collapse:

$$Q_3 = \begin{pmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{11} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ q_{14} & q_{24} & q_{34} & q_{44} \end{pmatrix}$$

$$\min_{v_3} v_3^T Q_3 v_3 : \quad \frac{\partial}{\partial x} = \frac{\partial}{\partial y} = \frac{\partial}{\partial z} = 0$$

- How do we find a critical point (min/max/saddle)?
- Set derivative to zero!
- matrix Q is *positive-definite* $\rightarrow$ *min*

Using Quadrics

Find optimal location v_3 after collapse:

$$Q_3 v_3 = \begin{pmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{11} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} v_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$v_3 = \begin{pmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{11} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Incremental mesh Decimation (with Quadric Error Metric)

ALGORITHM: INPUT: the original model

- Compute the **Q** matrices for all the initial vertices

For each edge

- Compute the optimal contraction target **v** for each edge (**v**1,**v**2). The error $\mathbf{v}^T(\mathbf{Q}1+\mathbf{Q}2)\,\mathbf{v}$ of this target vertex becomes the ***cost* of collapsing that edge**.

Sort for increasing cost

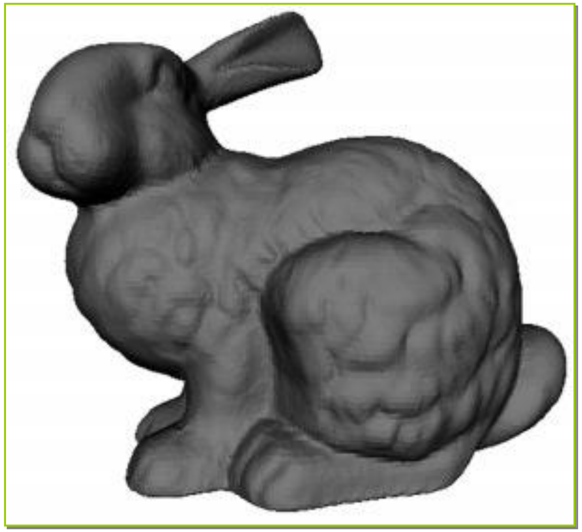
Repeat

- Edge collapse the minimum cost edge (vi,vj) **(decimation)** to get new vertex v
- add Qi and Qj to get quadric Qv at v
- update cost of edges touching v

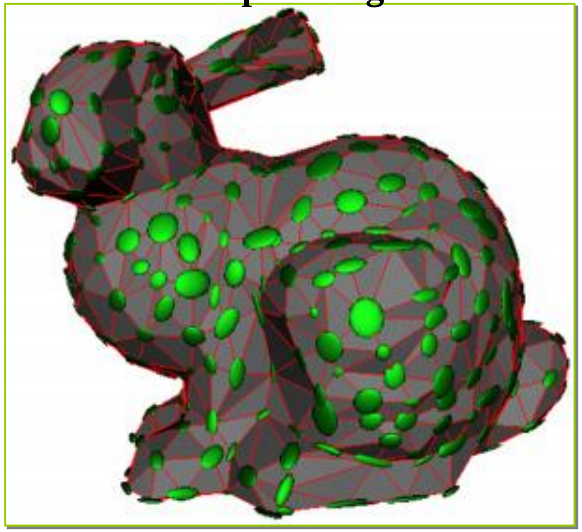
until (no further reduction possible)

Results: Quadric Visualization

- **Ellipsoids: iso-error surfaces**
- **Smaller ellipsoid = greater error**

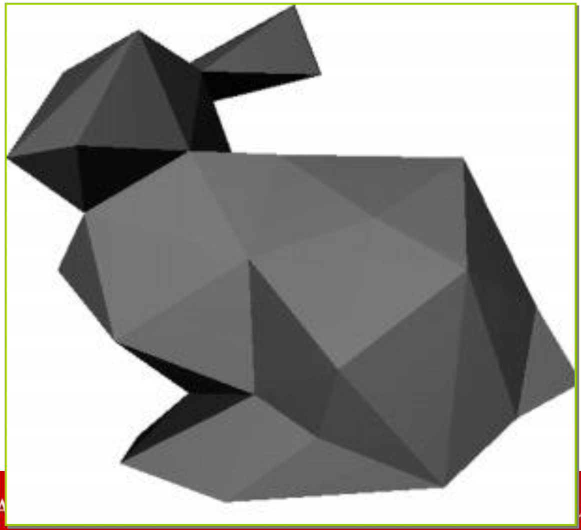


Original



Quadrics

1k tris



100 tris

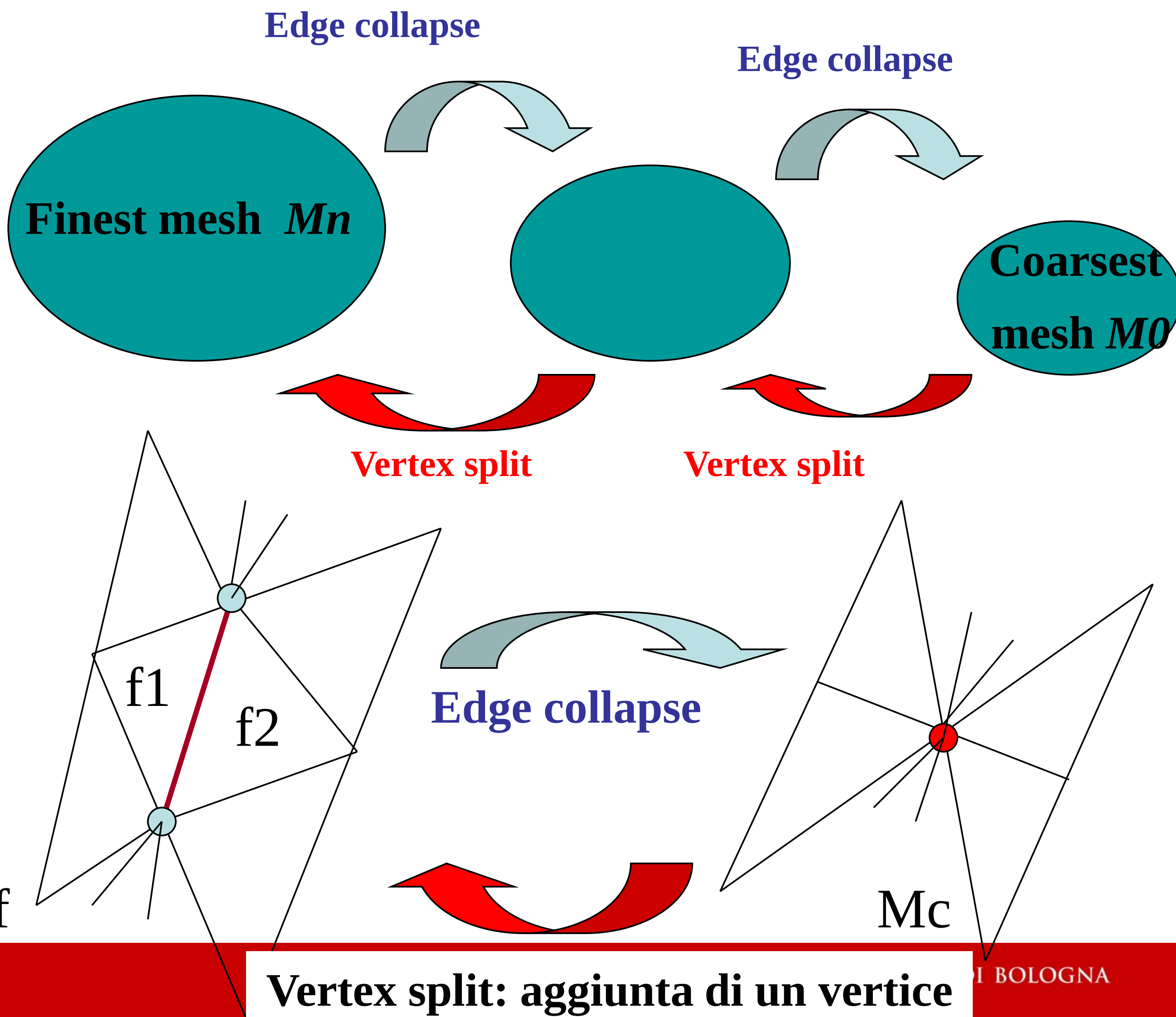
Progressive Mesh Algorithm (Hoppe 1996)

Reversible :

store

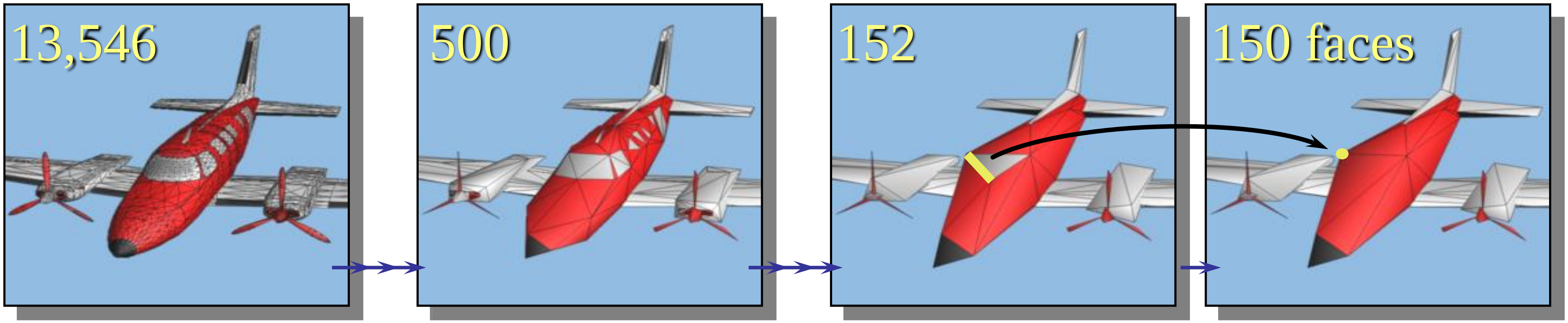
edge collapses

in a ordered list



Vertex split: aggiunta di un vertice

Progressive Mesh (Hoppe)



$$M^n \xrightarrow{ecol_{n-1}} \cdots\; M^{175} \;\cdots \xrightarrow{ecol_i} M^1 \xrightarrow{ecol_0} M^0$$

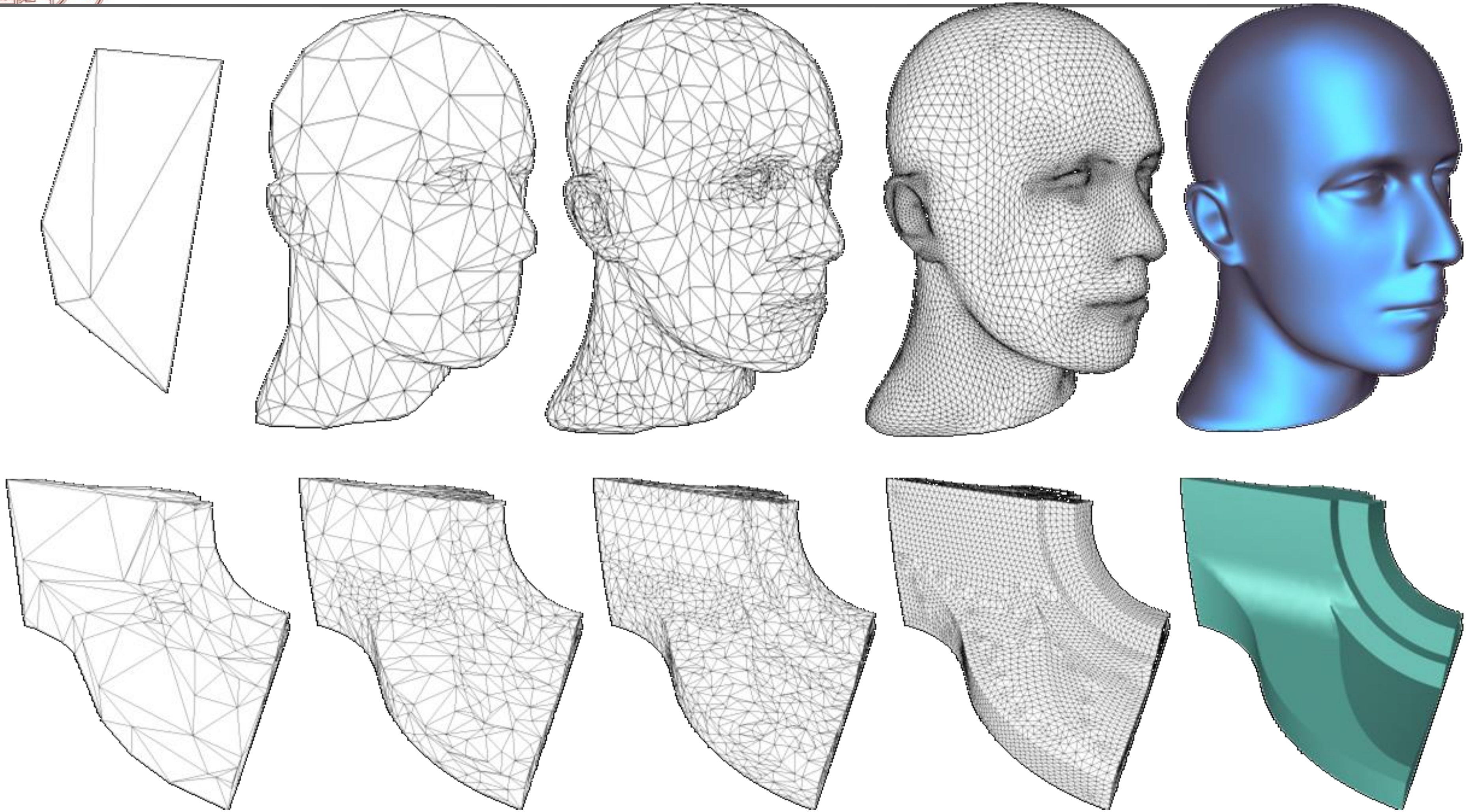
- Store edge collapses in a ordered list (decreasing edge collapsing (ecol) cost)
- Iteratively process a decimation of minimum cost and recompute the list of costs for neighbors
- An edge collapse is ok only if it does not change the mesh topology

Applications

Mesh simplification applied to ..

- ❑ LOD Approximation
- ❑ Progressive Transmission
- ❑ Mesh compression
- ❑ Selective Refinement

LOD Approximation



Create meshes at different **Level of Detail (LOD)**

LOD Approximation

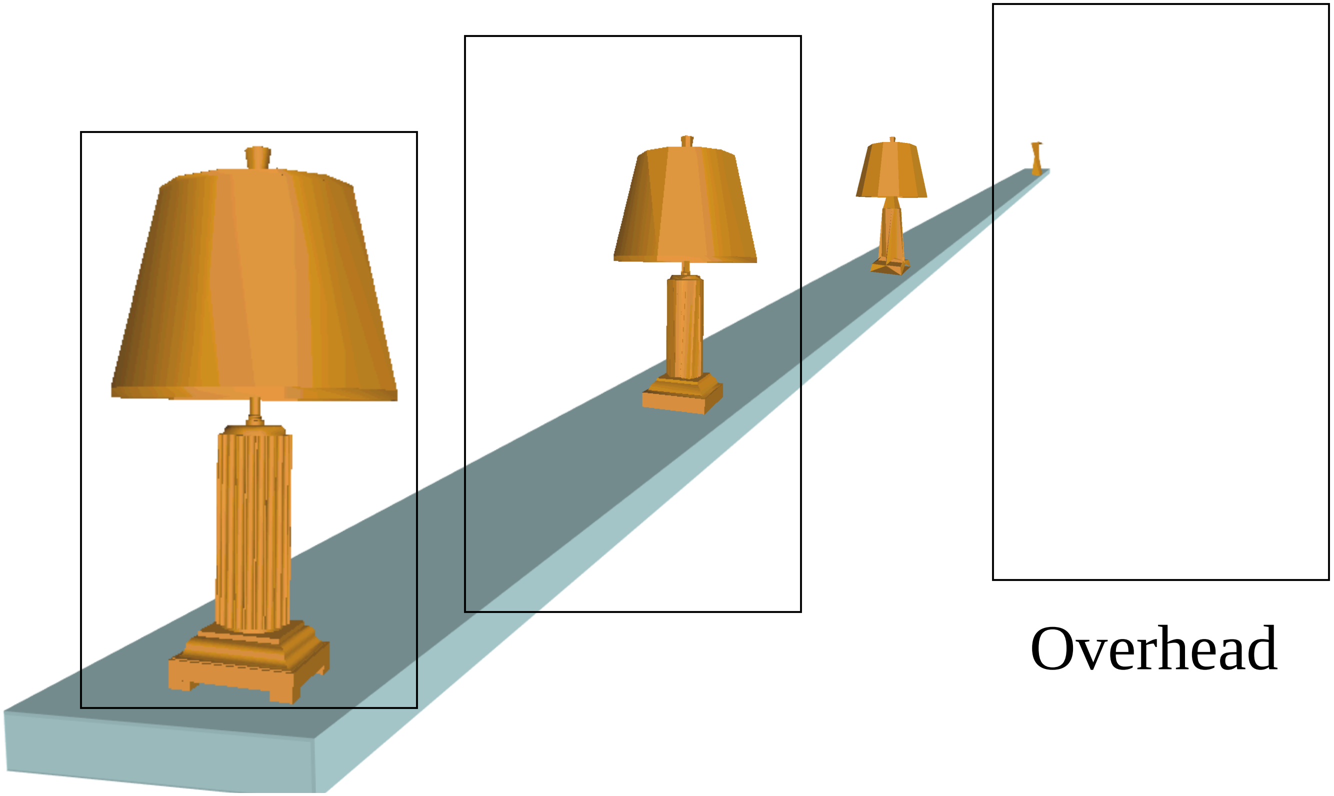
Need Multiresolution Models

- Context dictates required detail
  - LOD should vary with context

10,108 polys 1,383 polys 474 polys 46 polys

Courtesy IBM

need high detail near the viewer
need less detail far away



Example: obj with 10000T viewer near the object, if the object is far away (covers10x5 pixels) use a simplified model with 100 T

LOD

- **LOD of parametric surfaces:**

  Bézier patches, spline and subdivision have a natural LOD on discrete parametric domain.Different LODs are given by the resolution of the tessellation

- **LOD of meshes:**

  ALGORITHM:

  - Generation: Given a model, build a set of approximations can be produced by any simplification system
  - Selection: at run time, simply select which to render
  - Switching: Inter-frame switching from a LOD to the next

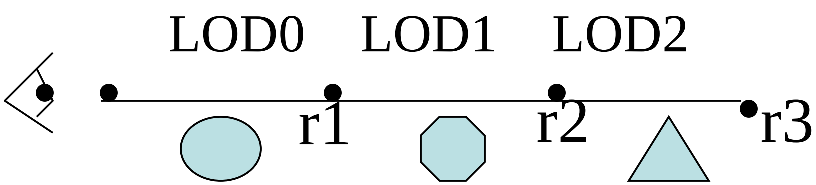
LOD switching

- **Discrete geometry LOD**
  - Switch from a LOD at frame i to the next LOD at frame i+1
  - Causes *popping*

- **Blend LOD**
  - Transition from LOD 1 to LOD2 in FB. If LOD1 is the current LOD then draw LOD1 in z-buffer and color buffer with alpha =1 (opacity)
  - Draw LOD2 in FB with alpha =0 (trasparency)
  - **LOD2 appears incrementing alpha from 0 to 1**
  - When LOD2 is visualized (alpha =1) LOD1 start to disappear (alpha decreases to 0)
  - Inbetween both LODs are rendered overlapped
  - No popping, it works in hardware

LOD selection

Evaluate a criterium based on point of view and object position, choose a suitable LOD



- **Range-based**
  - Distance from point of view
  - Each LOD is associated with a distance range

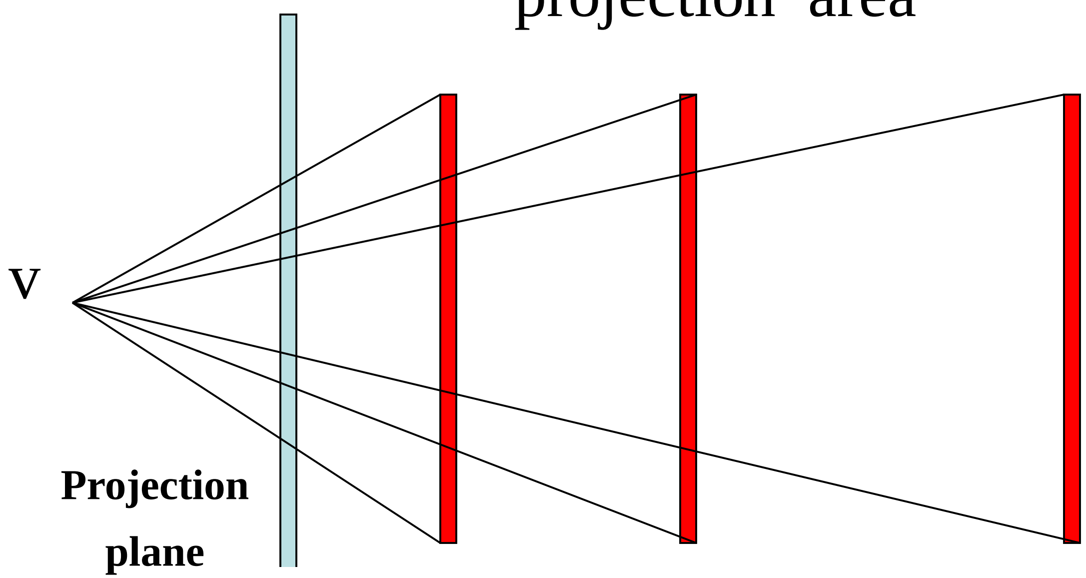
    (LOD0 more detailed associated with [0,r1])

LOD selection

- **Projected area based**
  - –Projected Area from bounding volume of the object (eg.sphere)
  - –Sphere center c radius r, camera at v, direction vector d,
  - –n distance camera-near plane

$$p = \frac{nr}{d \cdot (c - v)}$$ Radius projected sphere

projection area^2



Polygon at double distance

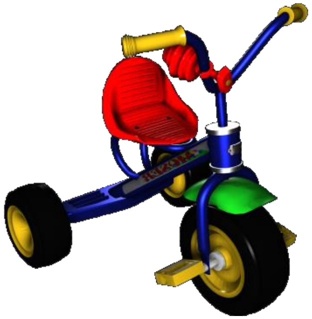
La dimensione è dimezzata al raddoppiare della distanza

3D Data (Mesh) Compression

Different needs:

Lossy…

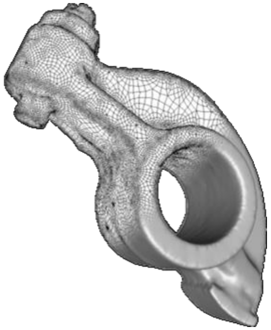
Games



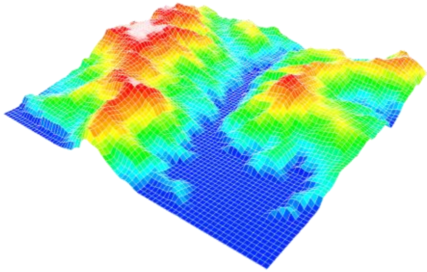
Virtual malls

o Lossless?

Medical

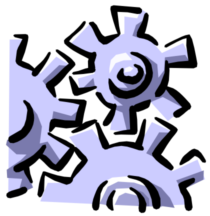


Engineering



Topography

Mesh compression



...01101101100010...

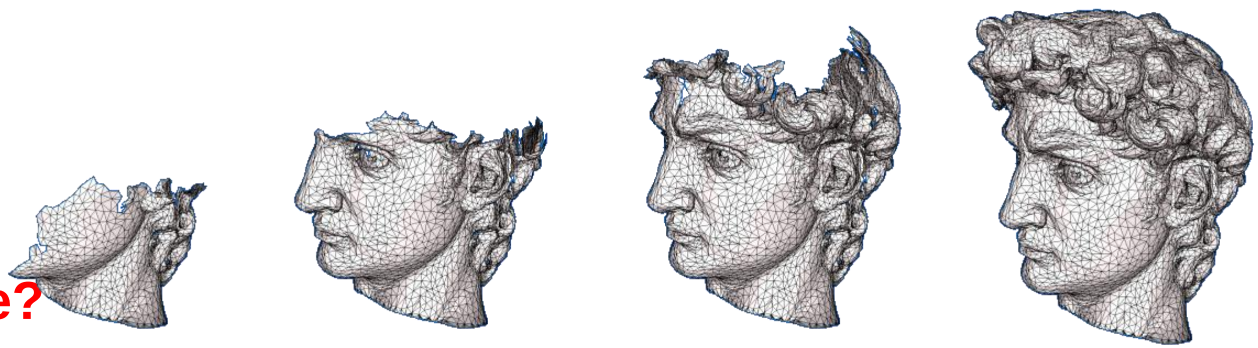
M0 (Original) + vertex split records

Not only the number of polys is reduced, also the storage for each LOD model is compressed.

Progressive transmission

The user waits for the entire transmission time in order to visualize the entire object,

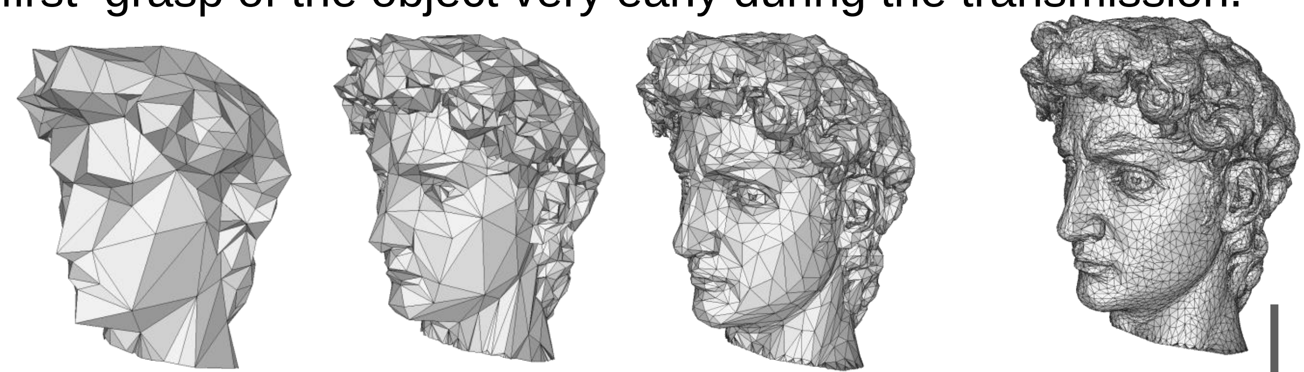
Single-rate…



or Progressive?

Transmission

The user sees a first grasp of the object very early during the transmission.



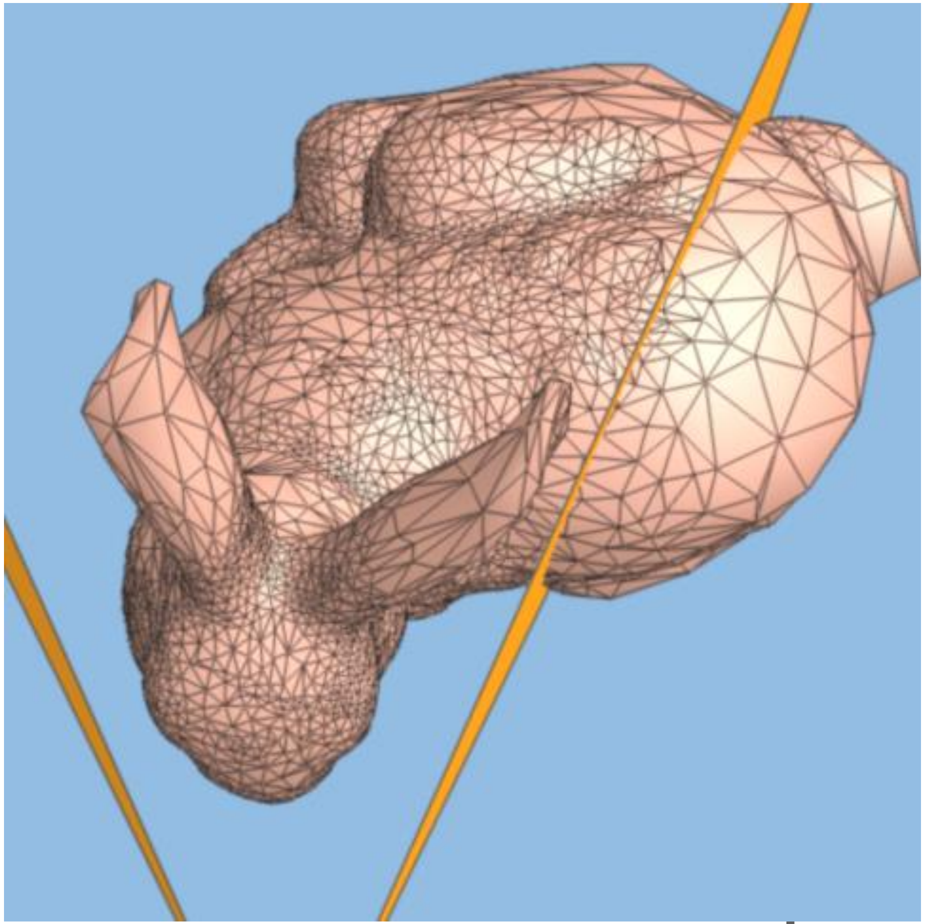
Coarse mesh +Database edge-collapse (compresses)

APPLY REVERSIBLE DECIMATIONS

Selective Refinement

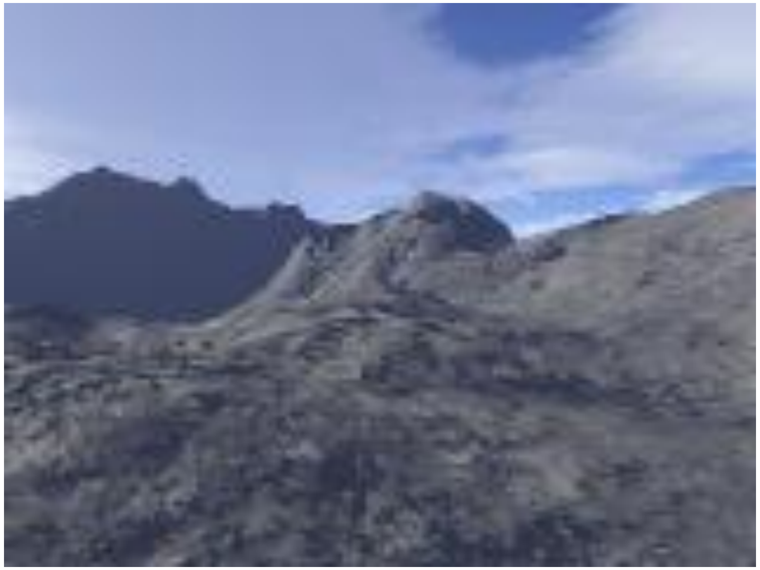
We may need varying LOD over surface:

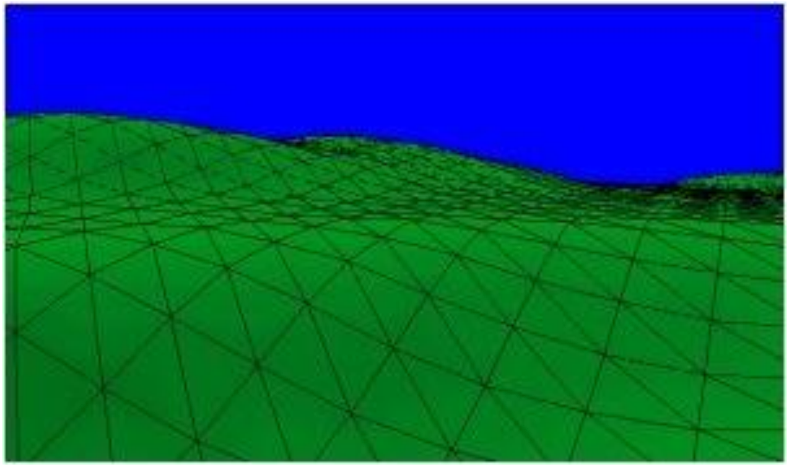
- large surface, oblique view (eg. on terrain)
  - high detail near the viewer
  - less detail far away
- single LOD will be inappropriate
  - either excessively detailed in the distance (wasteful)
  - or insufficiently detailed near viewer (visual artifacts)

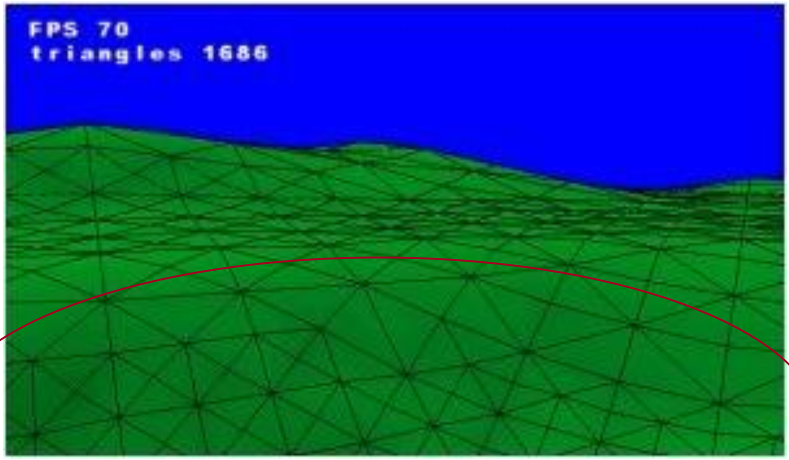


Selective Refinement for territorial models

- Find a base mesh
- Subdivide recursively the base mesh to approximate the original mesh



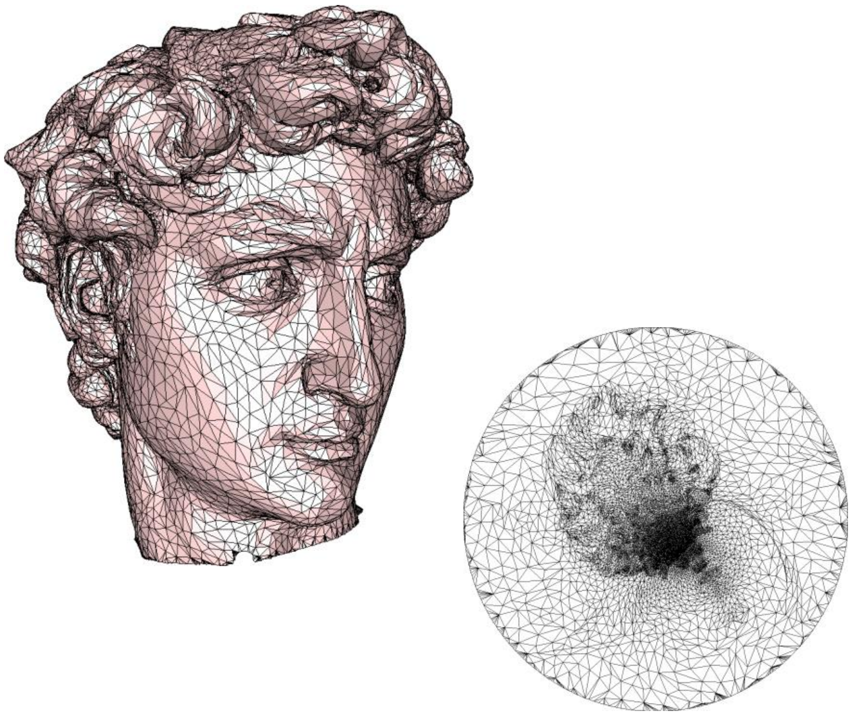




The closer part of the terrain is rendered to higher resolution

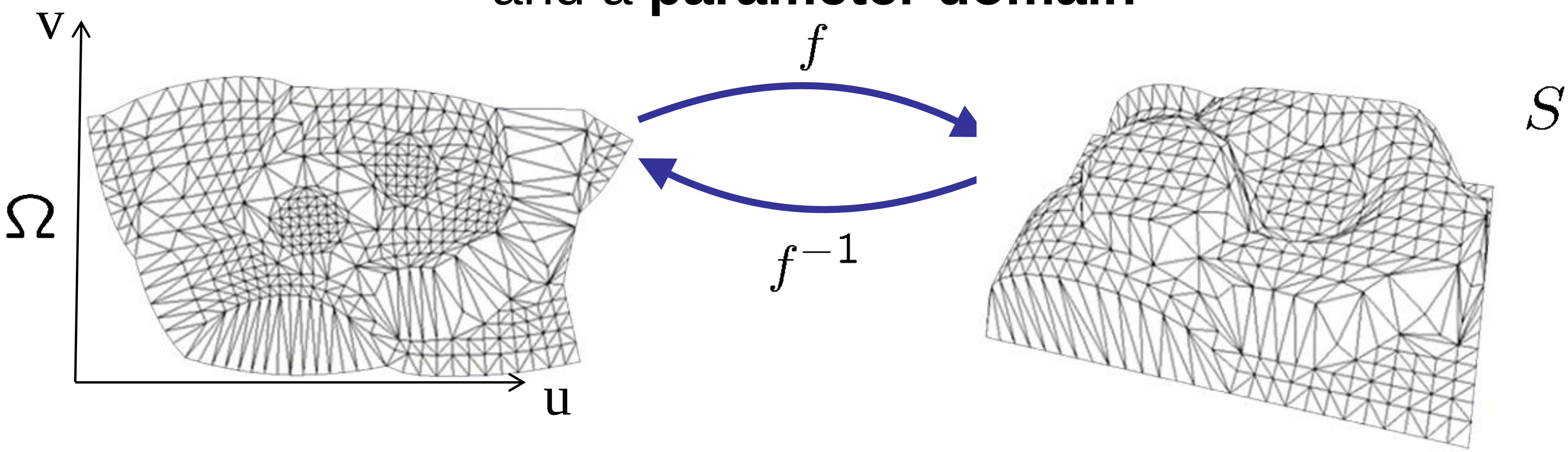
Outline

- Reconstruction
- Simplification
- **Parameterization**
- Remeshing
- **Smoothing/Fairing**
- Deformation/Editing
- Shape Analysis



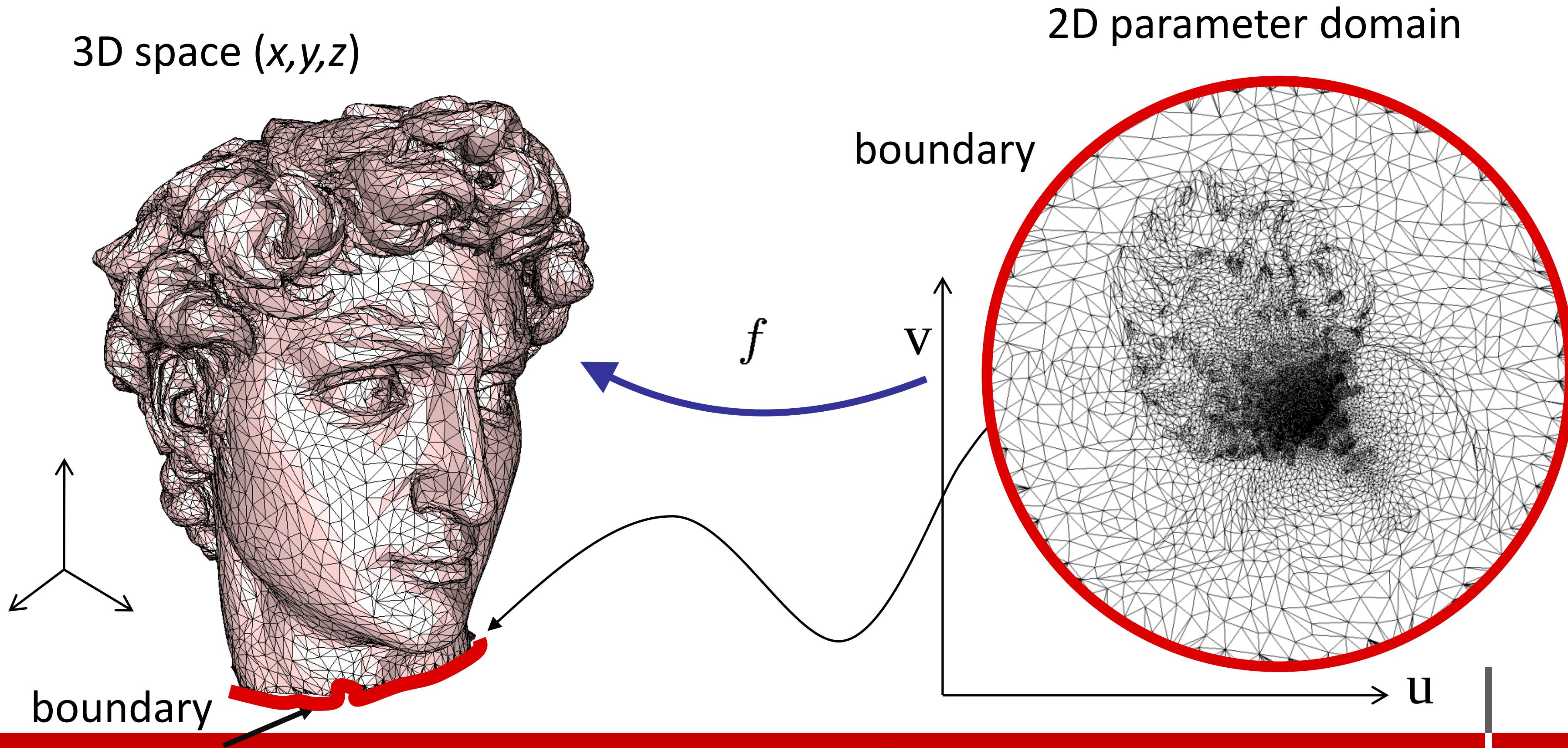
Parameterization

A **parameterization** is a **bijective mapping** between a **surface** and a **parameter domain**

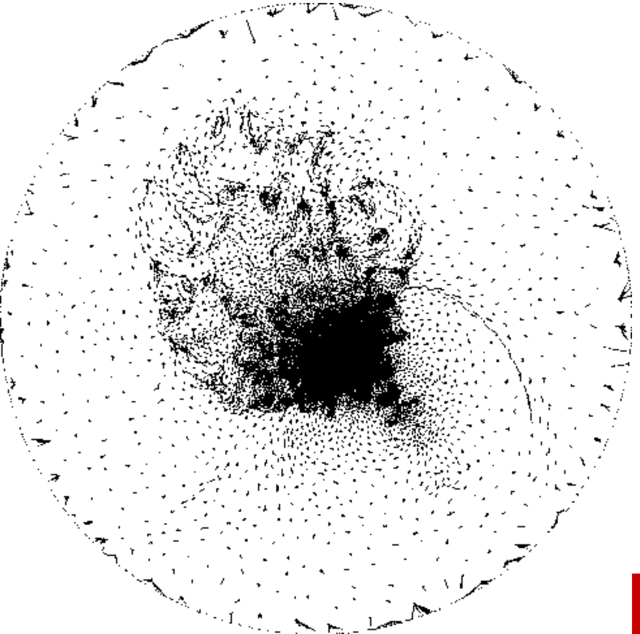


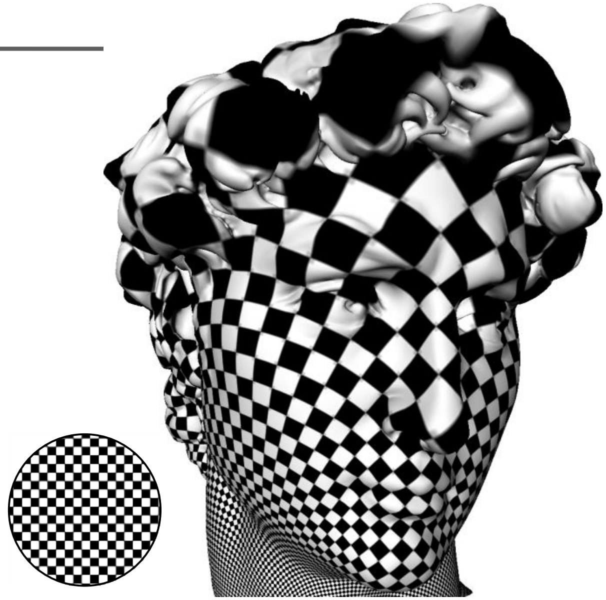
- **surface** $S \subset \mathbb{R}^3$
- **parameter domain** $\Omega \subset \mathbb{R}^2$
- **mapping** $f : \Omega \to S$ and $f^{-1} : S \to \Omega$

Mesh Parameterization

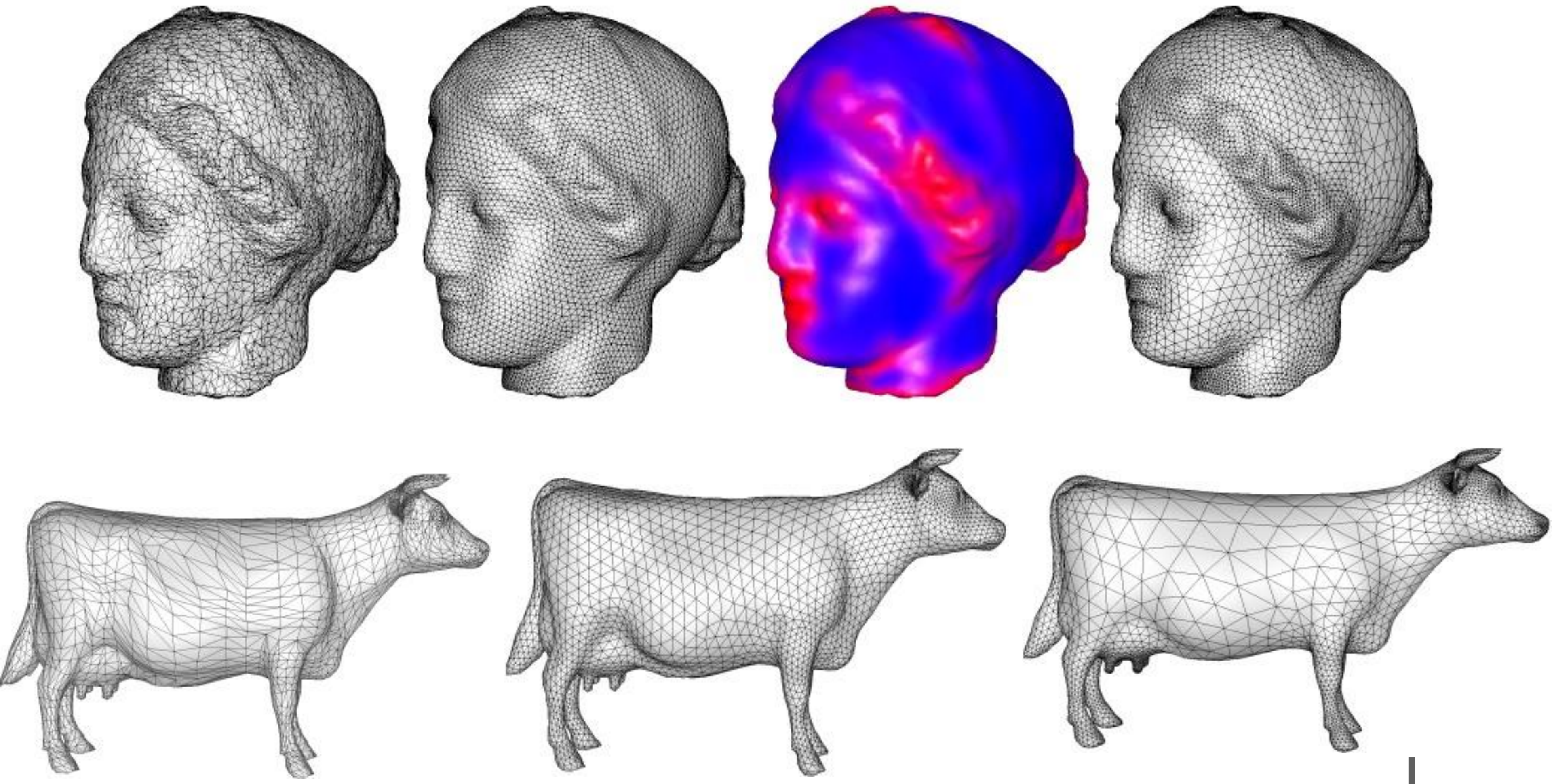


Applications - Texture mapping



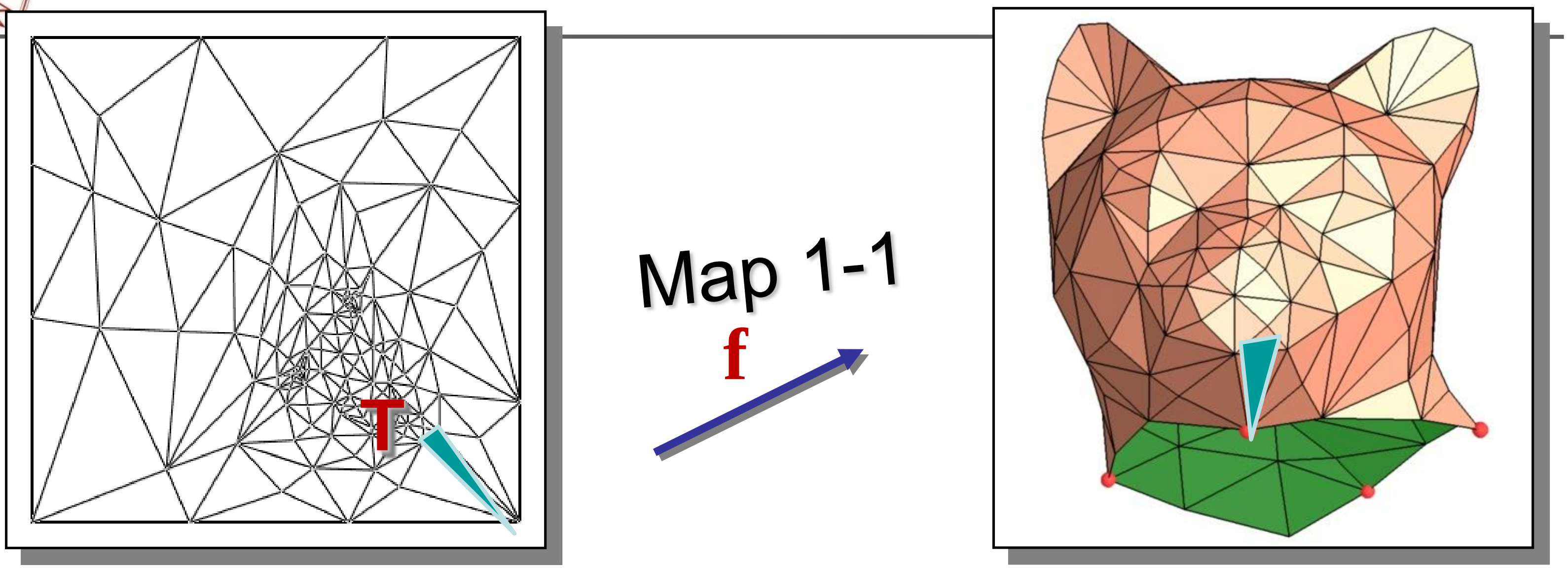


Applications - Remeshing

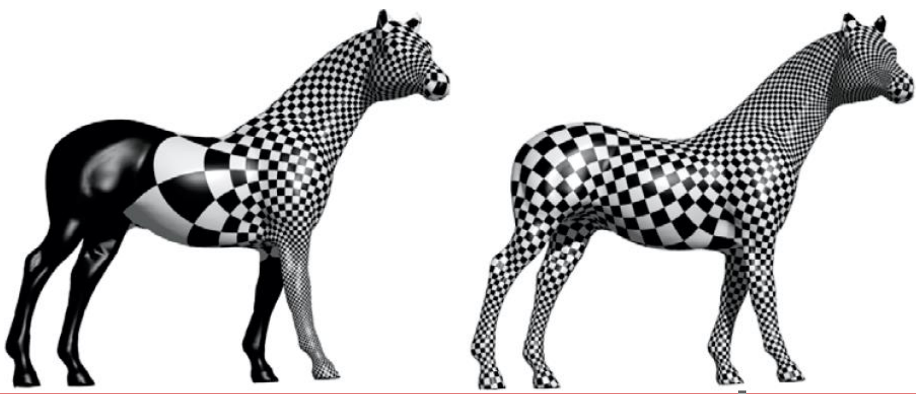


Pierre Alliez

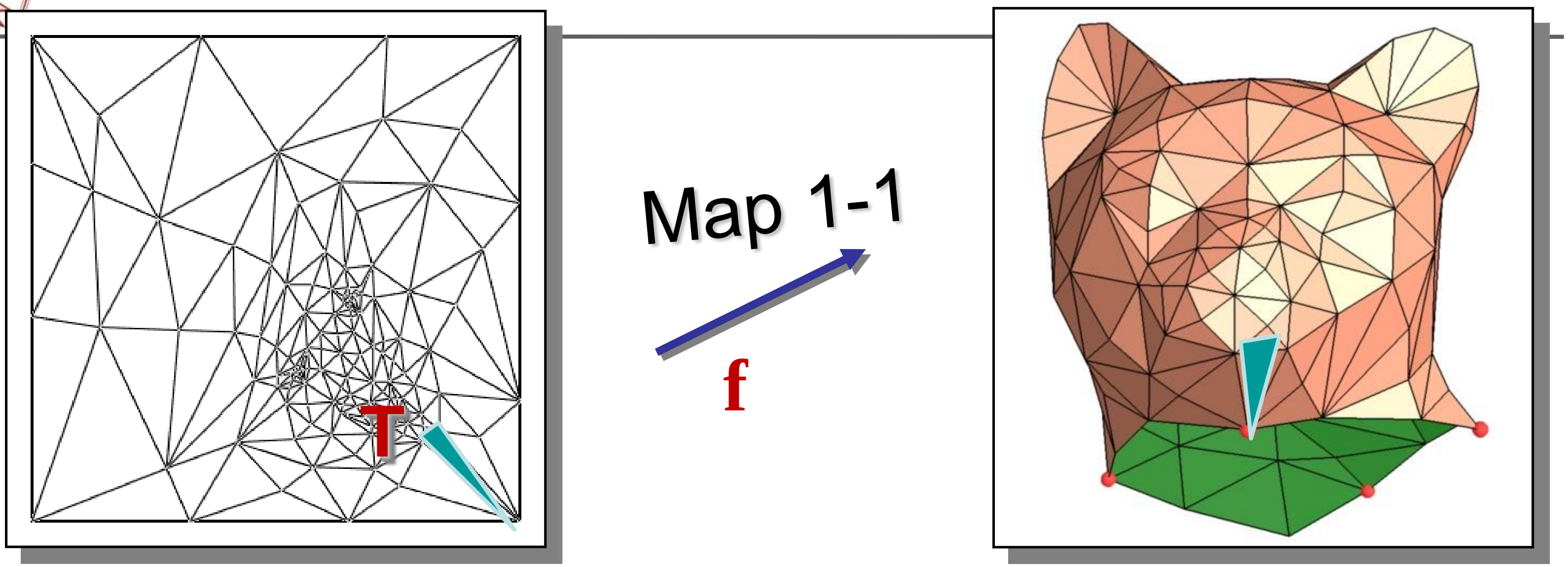
Desirable Properties



- Bijective function f (1-1 and onto): No triangles fold over.
- Minimal "distortion"
  - Preserve 3D angles (conformal)
  - Preserve 3D lengths (isometric)
  - Preserve 3D areas (equiareal)
  - No "stretch"
- Efficiently computable



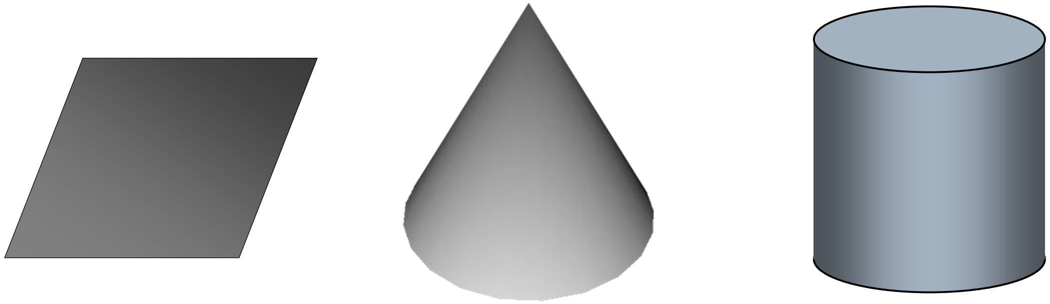
Definitions



- *f is* ***isometric (length preserving), if the length of*** any arc on *S is the same as that of its pre-image* on *Ω.*
- *f is* ***conformal (angle preserving), if the angle of*** intersection of every pair of intersecting arcs on *S* is the same as that of the corresponding preimages* on *Ω.*
- *f is* ***equiareal (area preserving) if every part of Ω*** is mapped onto a part of *S* with the same area*

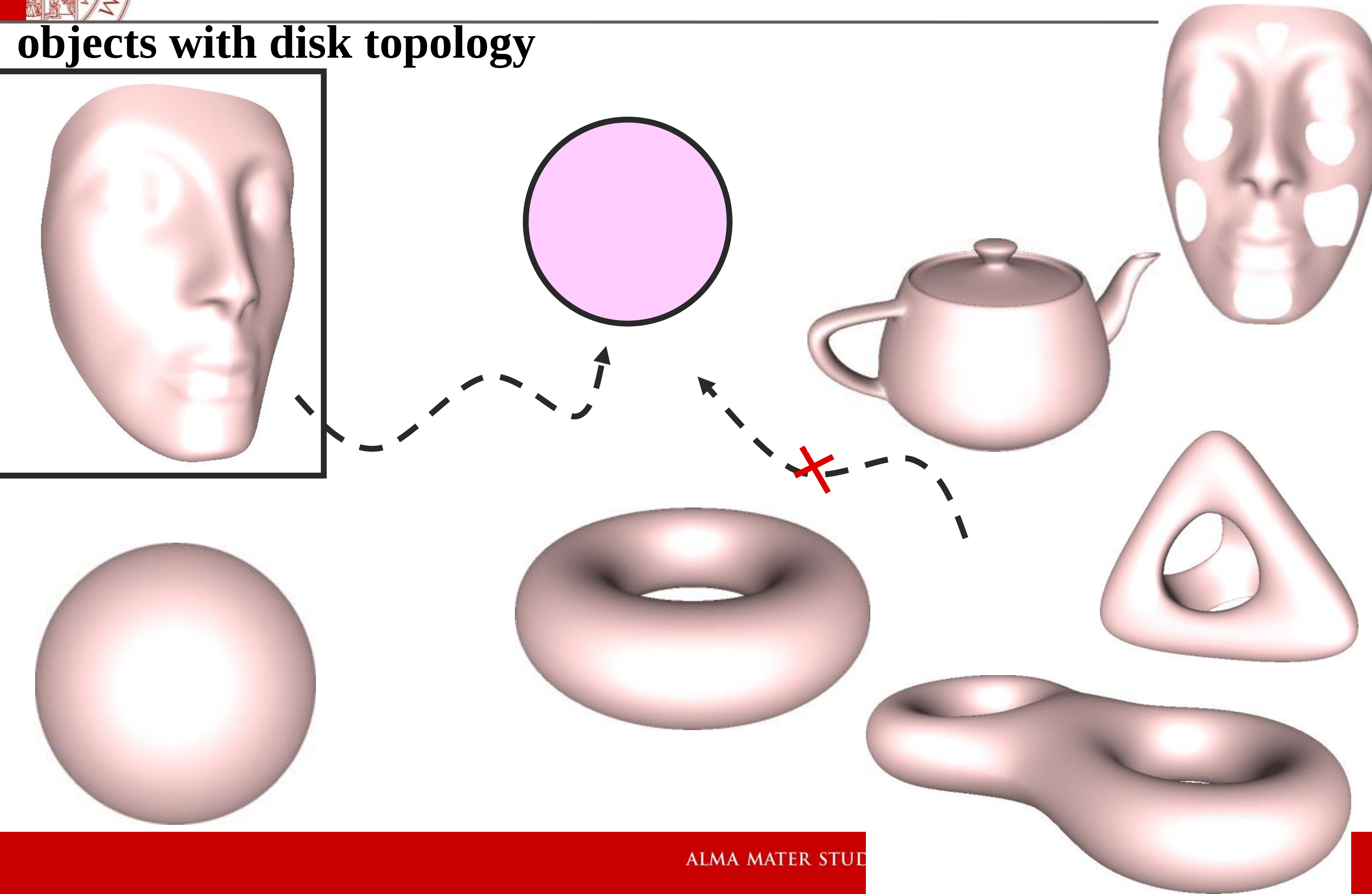
Distortion is (almost) Inevitable

- **Theorema Egregium** (C. F. Gauß)
  “*A general surface cannot be parameterized without distortion.*”
- no distortion = conformal + equiareal = **isometric**
- requires surface to be **developable**
  – planes
  – cones
  – cylinders



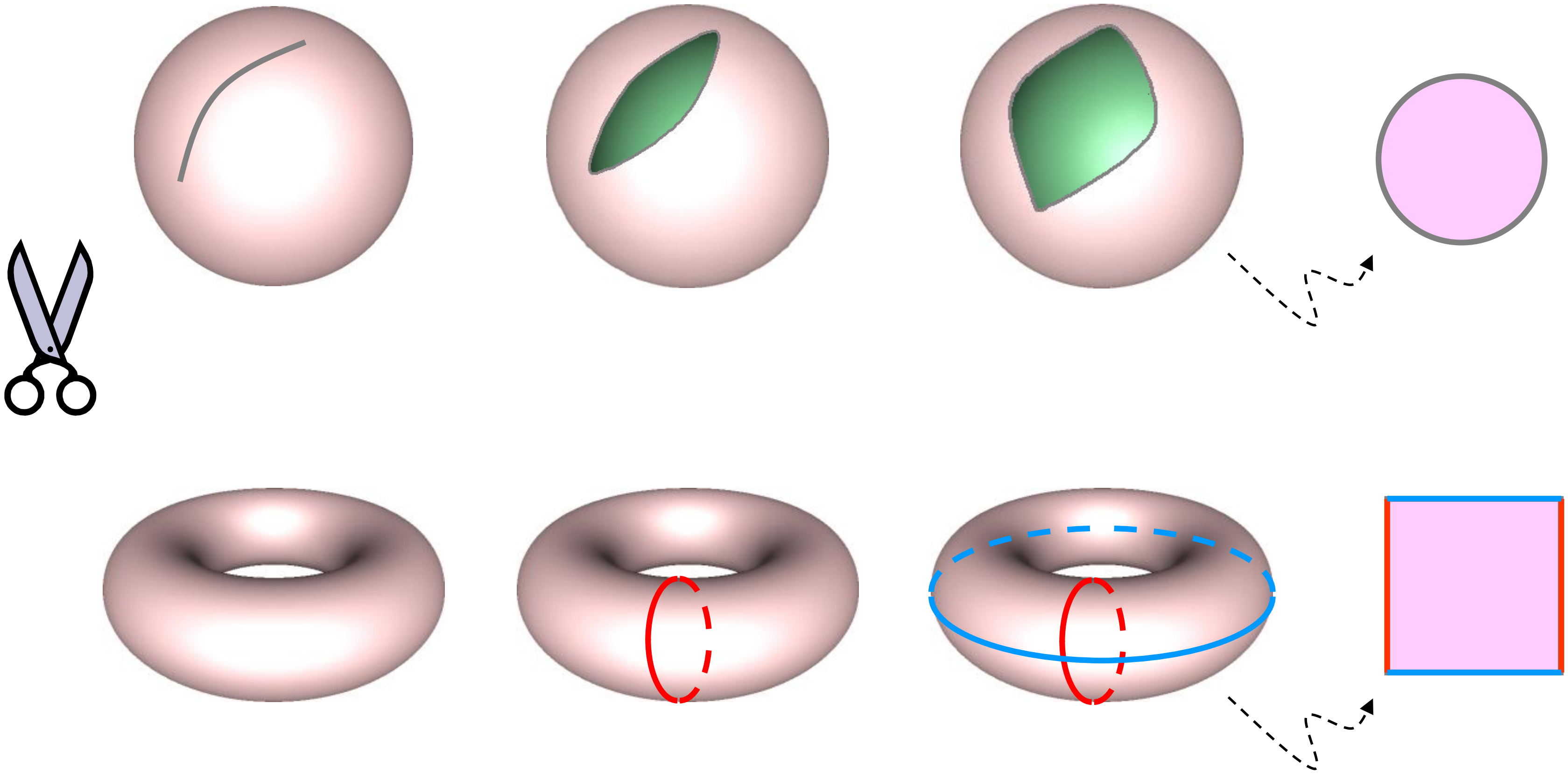
Genus=0, with boundary

objects with disk topology

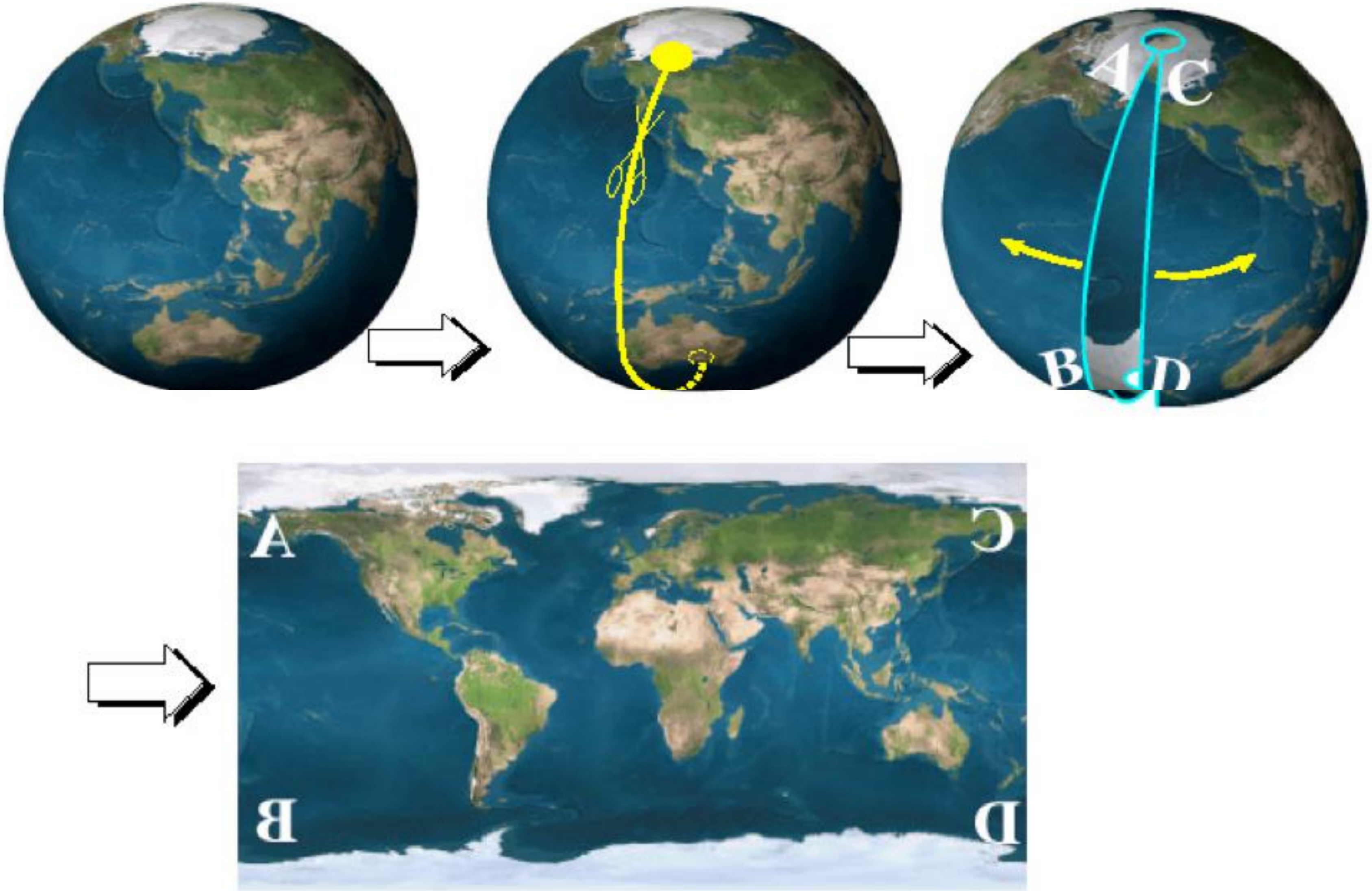


Cutting

- Introduce cuts on the mesh



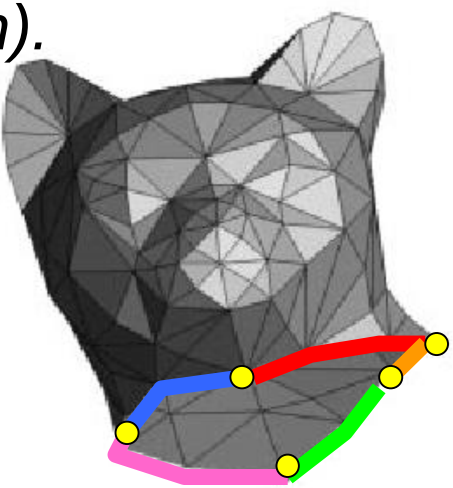
UNFOLDING the word



Convex combination maps
(Tutte-Floater)

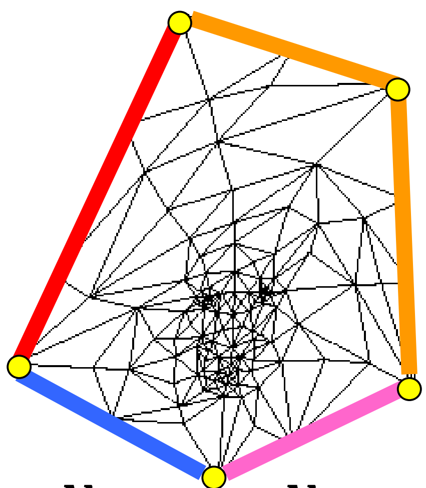
Barycentric mapping theorem (Tutte)

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



$\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$;

inner vertices



$\boldsymbol{v}_{n+1}, \ldots, \boldsymbol{v}_N$

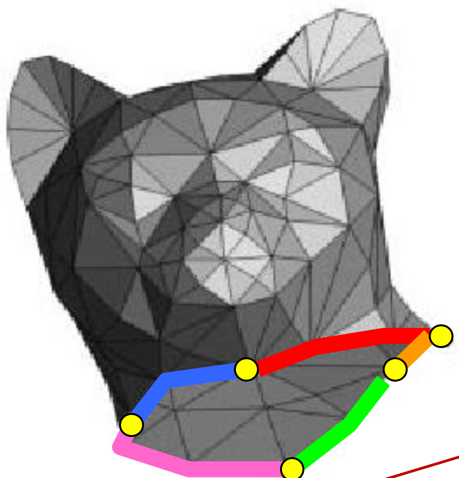
boundary vertices

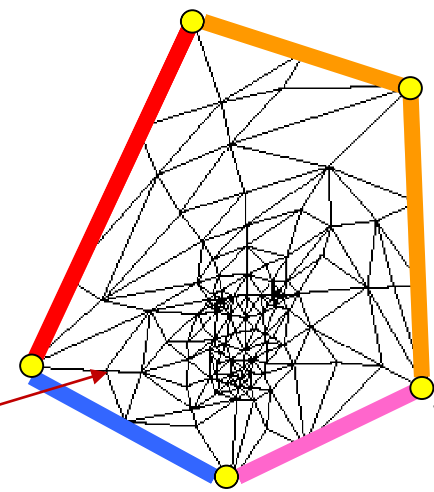
Eck, DeRose, Duchamp, Hoppe, Lounsbery, Stuetzle: *Multiresolution Analysis of Arbitrary Meshes, SIGGRAPH, 1995*

Triangle Mesh Parameterization

Convex combination maps (Tutte-Floater)

- Works for meshes equivalent to a disk
- Algorithm:
  - First, we map the 3D boundary to a 2D convex polygon
  - Then we compute the inner vertices positions





$\boldsymbol{x}_i = (\boldsymbol{v}_i, \boldsymbol{u}_i)$ coordinates of parameter points

$\{ (\boldsymbol{v}_1, \boldsymbol{u}_1), (\boldsymbol{v}_2, \boldsymbol{u}_2), \ldots, (\boldsymbol{v}_n, \boldsymbol{u}_n)\}$

(V,U) inner vertices

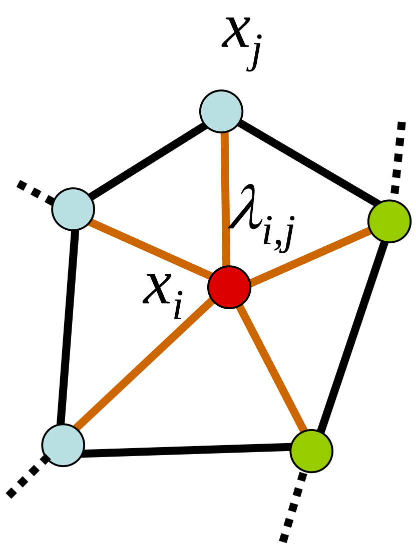
$\{ (\boldsymbol{v}_{n+1}, \boldsymbol{u}_{n+1}), \ldots, (\boldsymbol{v}_N, \boldsymbol{u}_N)\}$

(V_B, U_B) boundary vertices

Compute the inner vertices

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

$$\boxed{x_i = \sum_{j \in N(i)} \lambda_{i,j} x_j, \quad i = 1, 2, \ldots, n}$$



- with **weights**

$$\begin{cases} 0 & \text{if } (i, j) \text{ are not neighbors} \\ \lambda_{i,j} > 0 & \text{if } (i, j) \in E \text{ (neighbors)} \\ \lambda_{i,i} = -\sum_{j \neq i} \lambda_{i,j}, & \\ \sum_{j \in N(i)} \lambda_{i,j} = 1 & \end{cases}$$

Linear systems of equations

$$x_i - \sum_{j \in N(i)} \lambda_{i,j} x_j = 0, \quad i = 1, 2, \ldots, n$$

$$\underbrace{x_i - \sum_{j \in N(i) \setminus B} \lambda_{i,j} x_j}_{\text{unknown parameter points}} = \underbrace{\sum_{k \in N(i) \cap B} \lambda_{i,k} x_k}_{\text{fixed } (r_i)}, \quad i = 1, 2, \ldots, n$$

solve the linear system :

$$\underbrace{\begin{pmatrix} -1 & & \lambda_{1,j_1} & & \lambda_{1,j_{d1}} \\ & -1 & & & \\ & & -1 & & \\ & \lambda_{4,j_1} & & \ddots & \\ & & \lambda_{n,j_5} & & -1 \end{pmatrix}}_{L} \begin{pmatrix} x_1 \\ x_2 \\ \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} -r_1 \\ -r_2 \\ \\ \\ -r_n \end{pmatrix}$$

Linear system of equations

$$x_i = (u_i, v_i) \Rightarrow$$

$$LU = U_B \quad and \quad LV = V_B$$

- **solve** system **twice**
- matrix L is
  - sparse
  - diagonally dominant
  - nonsingular
- A unique solution always exists
- Important: the solution is legal (bijective)
- The system is sparse, thus fast numerical solution is possible
- Numerical problems (because the vertices in the middle might get very dense…)

Choice of Weights

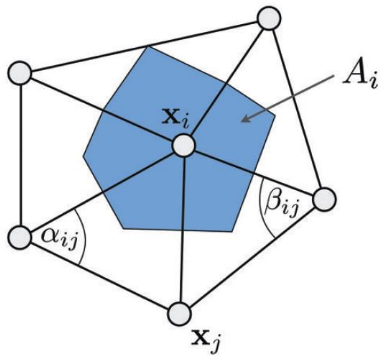
Discrete Laplacian

The weights are given by

$$\boxed{L = (\lambda_{i,j}), \qquad \lambda_{i,i} = -1}$$

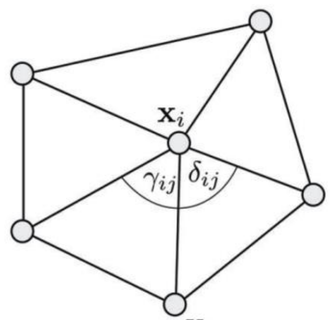
- Uniform /Barycentric (Tutte) $\quad \overline{\lambda_{i,j}} = \dfrac{1}{d_i}$ $\quad$ d_i valence of vertex i

- Discrete harmonic



$$\overline{\lambda_{i,j}} = \frac{1}{2A_i}(\cot\alpha_{i,j} + \cot\beta_{i,j})$$

- Mean value

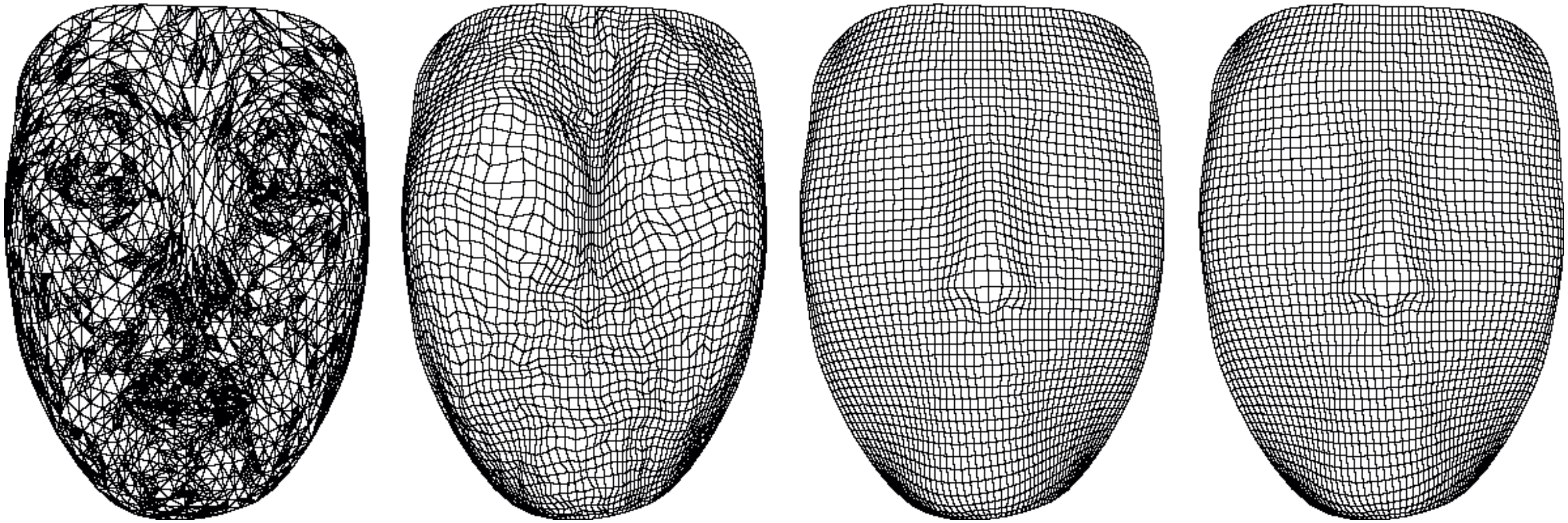


$$\overline{\lambda_{i,j}} = \frac{1}{\left\| X_i - X_j \right\|}\left(\tan\left(\frac{\delta_{i,j}}{2}\right) + \tan\left(\frac{\gamma_{i,j}}{2}\right)\right)$$

normalization $\quad \lambda_{i,j} = \dfrac{\overline{\lambda_{i,j}}}{\sum_{k \in N(i)} \overline{\lambda_{i,k}}}$

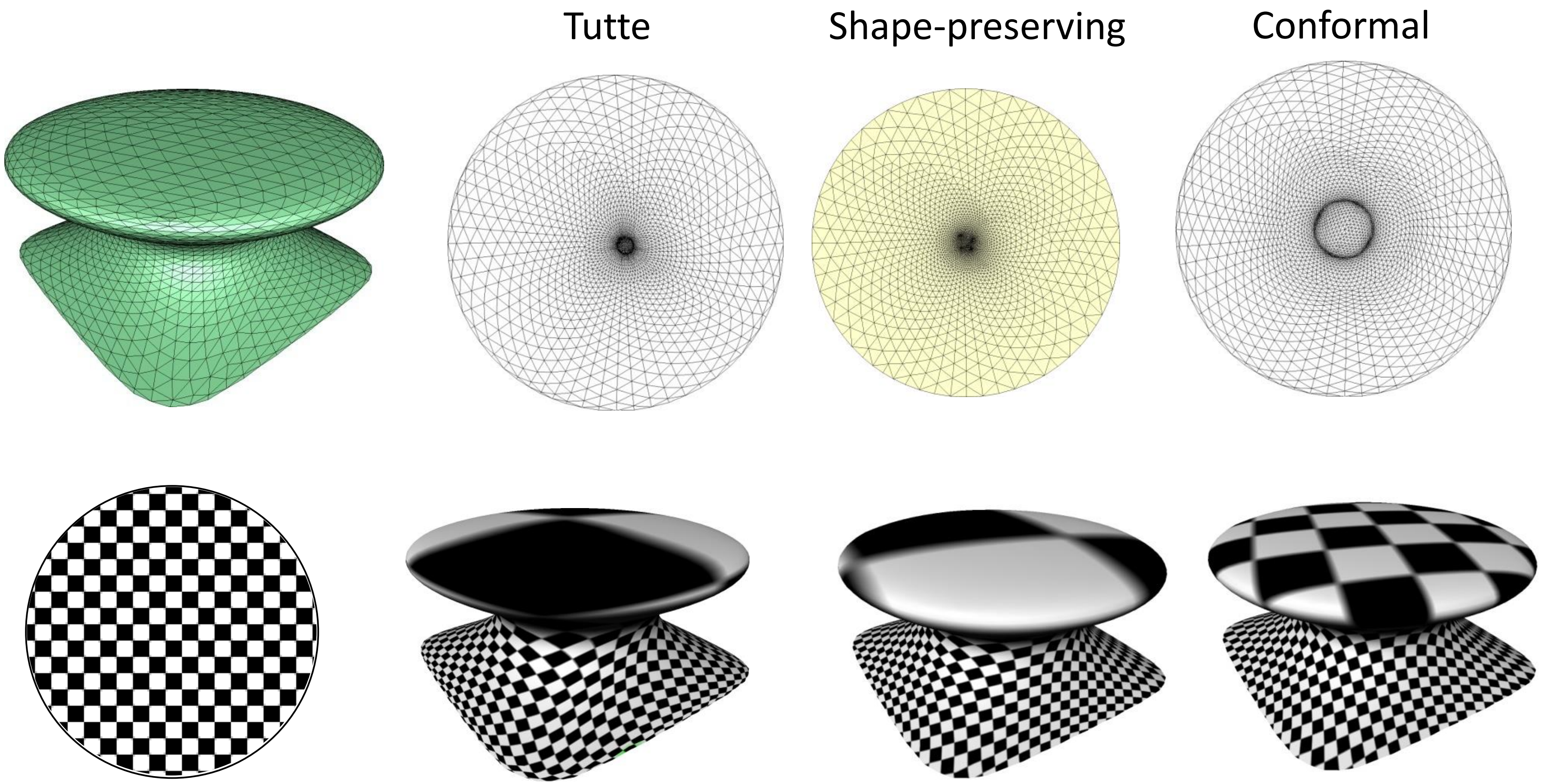
Convex Combination Maps

- Comparison



Original mesh | Barycentric | Discrete Harmonic | mean value

Texture mapping applications



Texture map

Stretch minimization

(S. Yoshizawa , 2004)

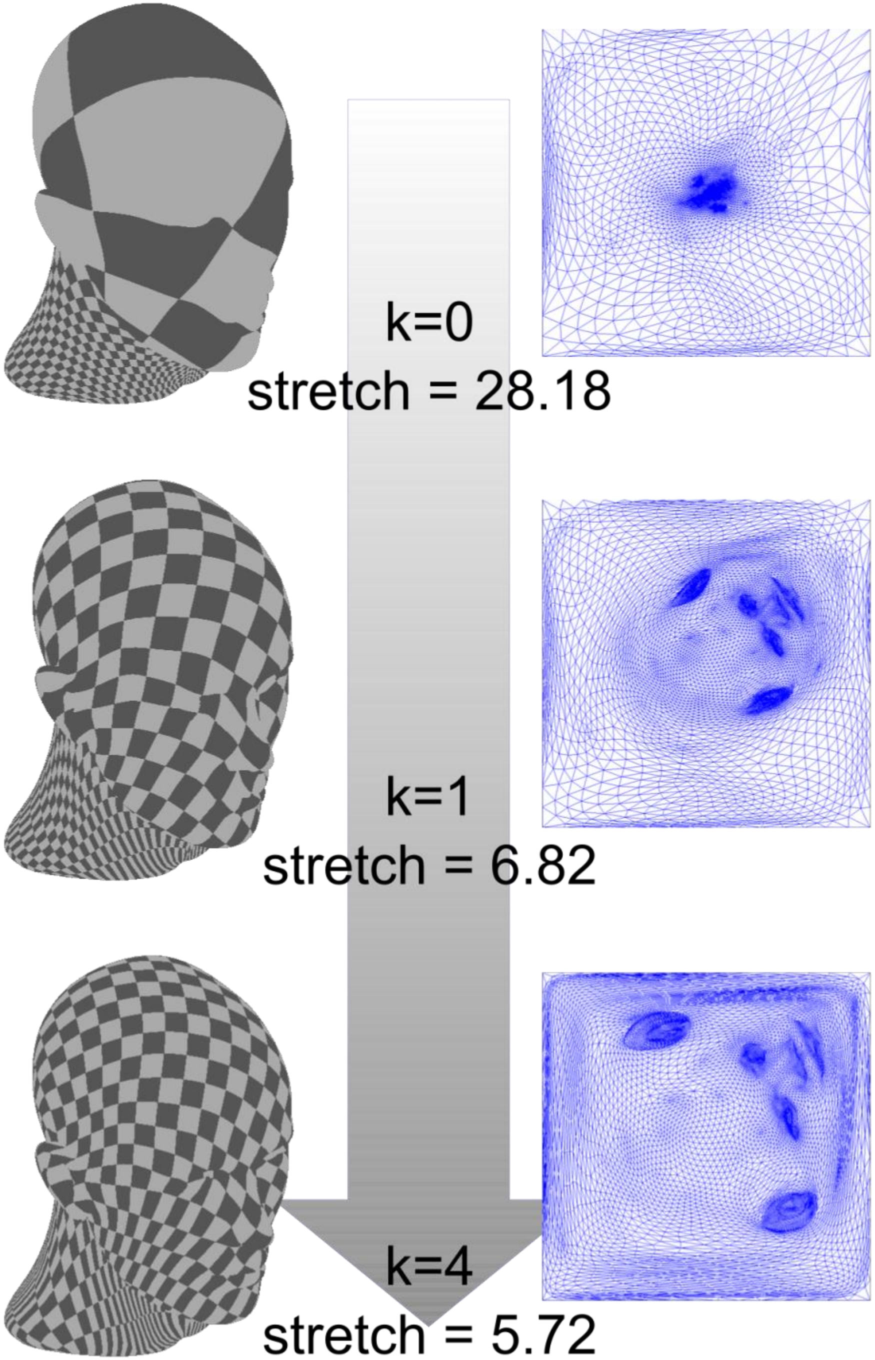
Iterative algorithm to correct the weights

- **Initialize:**
  convex map
- **STEP *k*:**
  - Update $\lambda_{i,j}$ w.r.t. stretch σ_j in X_j

$$\lambda_{i,j}^{k} = \frac{\overline{\lambda_{i,j}}}{\sum_{n \in N(i)} \overline{\lambda_{i,n}}} \qquad \overline{\lambda_{i,j}} = \frac{\lambda_{i,j}^{k-1}}{\sigma_j}$$

  - Solve the new LS

- **Stopping** when the global stretch energy stops decreases

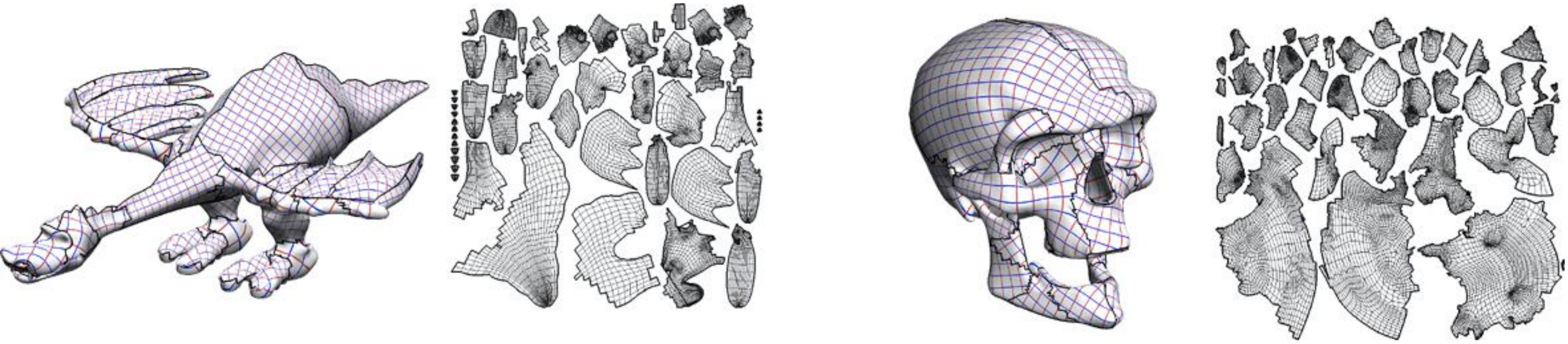


Texture Atlas Generation

Split model into number of patches (atlas)

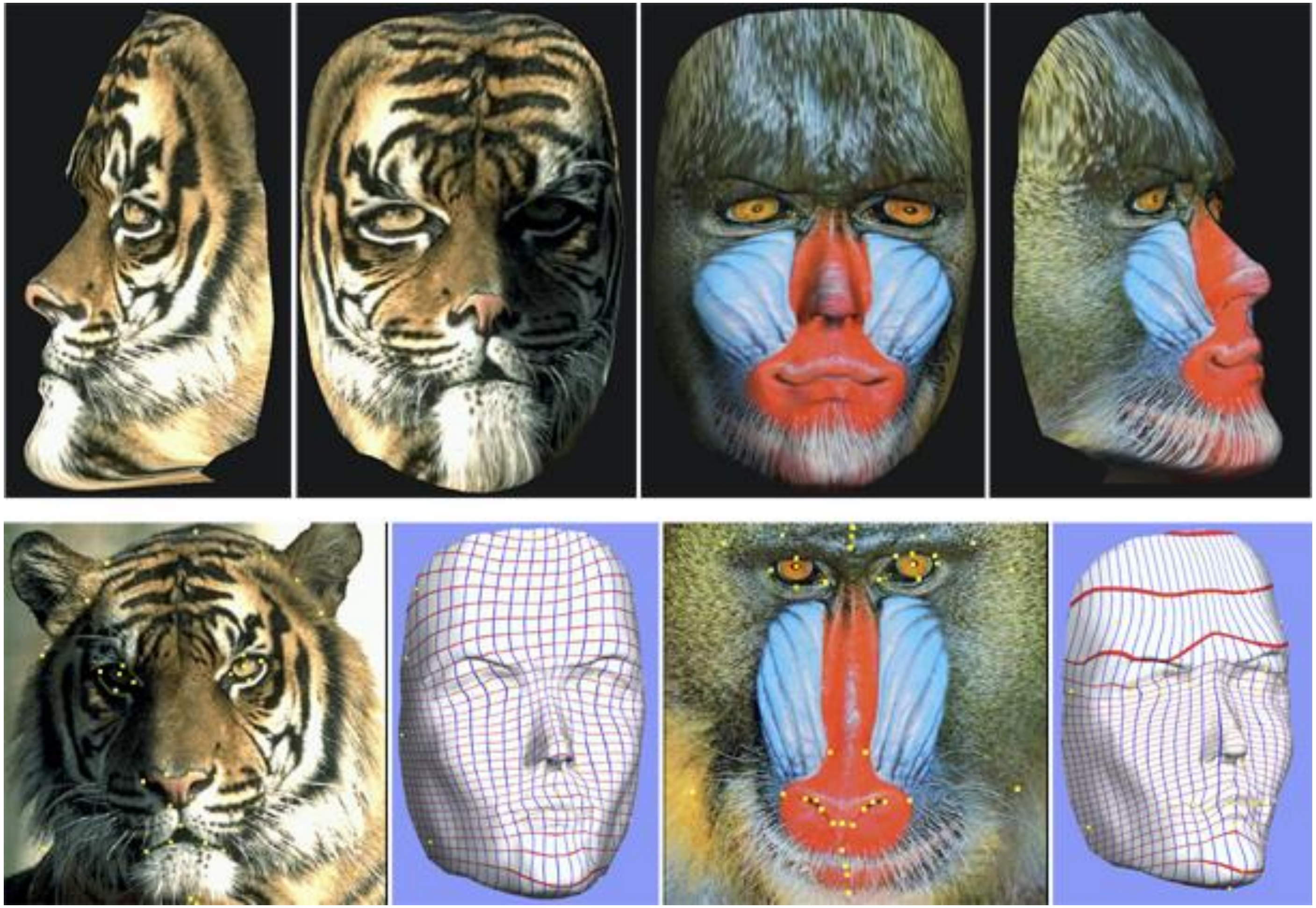
– because higher genus models cannot be mapped onto plane and/or

– because distortion will be too high eventually



Levy, Petitjean, Ray, Maillot: *Least Squares Conformal Maps for Automatic Texture Atlas Generation, SIGGRAPH, 2002*

Constrained Parameterizations



Levy: *Constraint Texture Mapping, SIGGRAPH, 2001*

Outline

- Recnstruction
- Simplification
- **Parameterization**
- Remeshing
- **Smoothing/Fairing**
- Deformation/Editing
- Shape Analysis

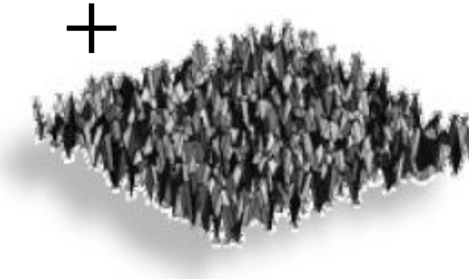
Surface fairing

$$X^0 = \overline{X} + \vec{E}$$

$$\overline{X} = (x, y, z)$$

$=$

$+$



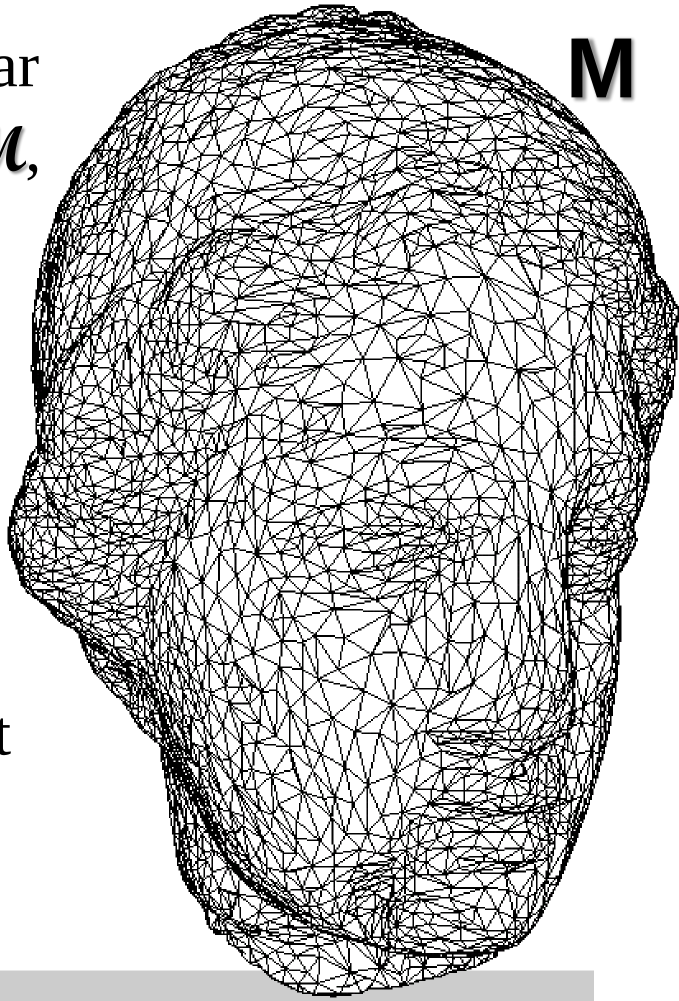
Mesh Fairing

$\mathcal{M}$

M a piecewise linear approximation of $\mathcal{M}$,

X is a surface parameterization of $\mathcal{S}$

X is the vertex set of **M**



M

Fairness: low variation in curvature.

Move vertices to achieve - Mesh topology stays the same.

Surface Fairing

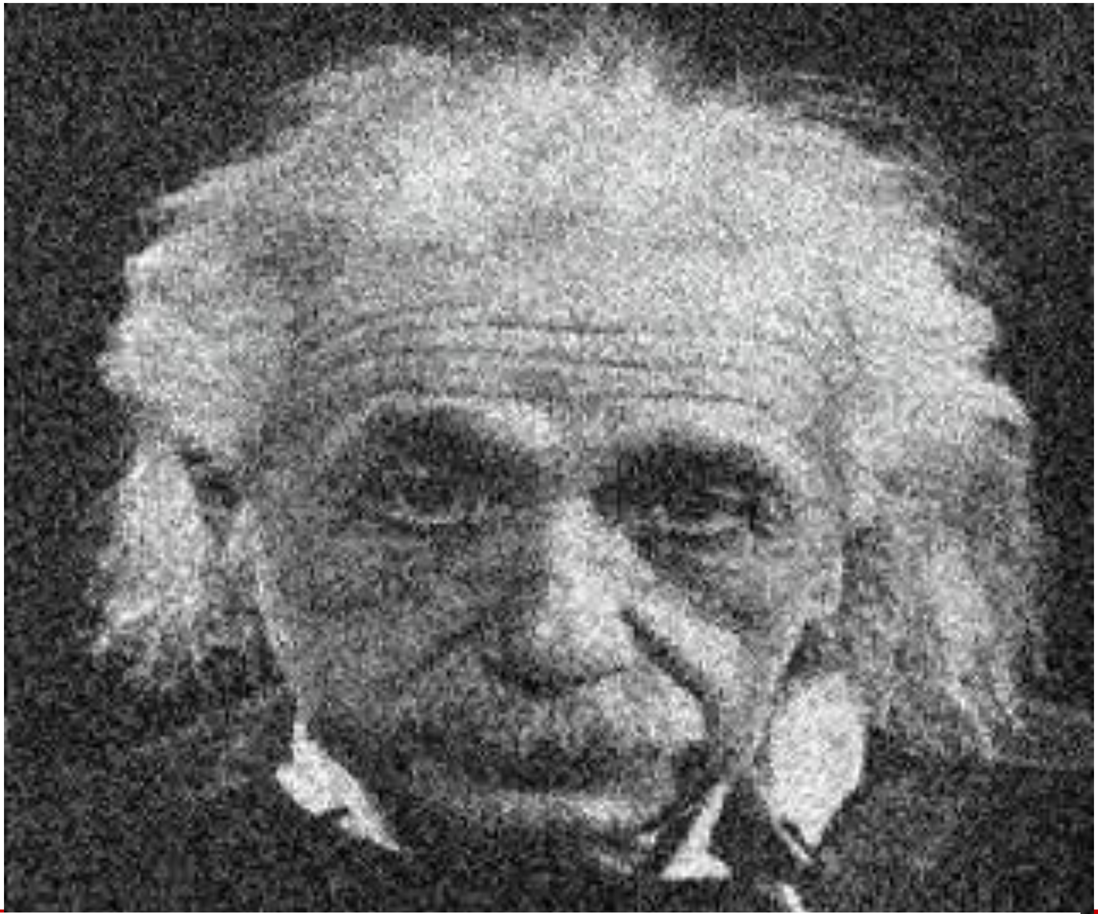
- **A surface smoothing method, named fairing, removes undesirable noise and uneven edges from discrete surfaces.**

- Extend the low-pass filter in image processing

- **Idea:** a low pass filter corresponds to diffusion flow.

Image Denoising by linear diffusion flow

$$\frac{\partial u}{\partial t} = \Delta u \quad u_0 = \text{noisy image} \qquad \boldsymbol{u(t) = G_{\sqrt{2t}} * u(0)} \qquad \sigma = \sqrt{2t}$$



[heat equation = Gaussian filter Witkin-Koenderink '83-84]

Diffusion Flow on Meshes

1) Mean Curvature Flow (MCF)

$$\frac{\partial X}{\partial t} = \Delta_s X \quad \Longrightarrow \quad \frac{\partial X}{\partial t} = -\vec{H}N = \Delta_s X$$

2) Surface Diffusion Flow

$$\frac{\partial X}{\partial t} = \Delta_s H(\mathrm{X})$$

3) MCF + data fidelity

$$\frac{\partial X}{\partial t} = \Delta_S X + \lambda(f - X^0), \quad X\big|_{t=0} = X^0$$

Noisy mesh

MCF

SDF

and its curvature

igea 268K faces

Discrete diffusion flow on mesh

- Considering a uniform discretization of the time interval [0, T], T > 0, and using a temporal time step τ,
- *the approximation* of an evolving surface at the n-th time step is denoted by a spatial position vector X^n

$$\frac{\partial X}{\partial t} \approx \frac{X^{n+1} - X^{n}}{\tau}$$

- $L \in R^{N_V \times N_V}$ (connectivity matrix) represents the discrete Laplace-Beltrami operator

$$\Delta_M X \approx L\, X$$

In matrix vector form

Discrete diffusion flow on mesh

- Considering a uniform discretization of the time interval [0, T], T > 0, and using a temporal time step τ, *the approximation* of an evolving surface at the n-th time step by MCF

$$\frac{\partial X}{\partial t} = \Delta_s X$$

- Given the position vector X^n we get X^{n+1} by applying explicit schemes:

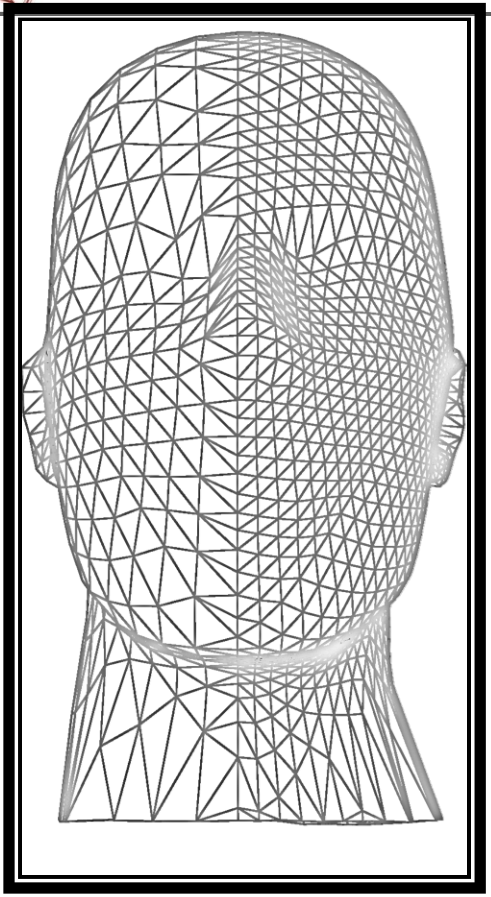
$$X^{n+1} = X^n + \tau L^n X^n$$

- or implicit schemes

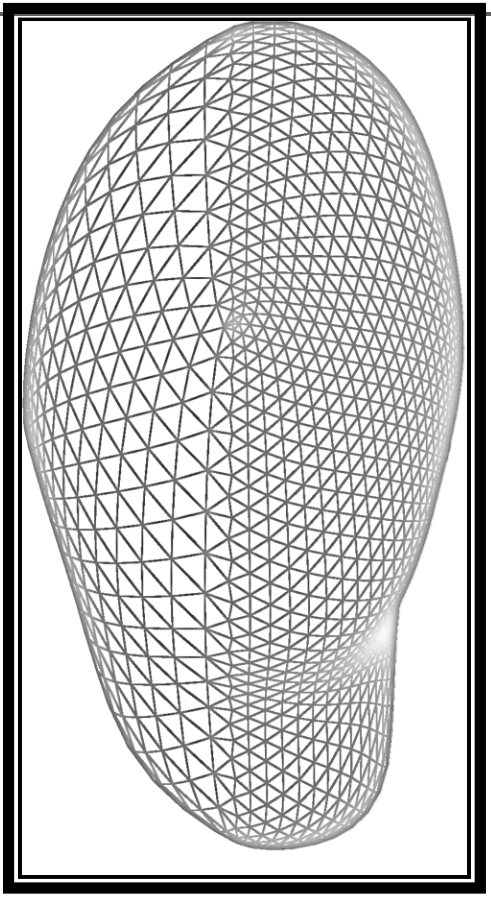
$$(I - \tau L^n) X^{n+1} = X^n$$

where L represents the discrete Laplace-Beltrami operator

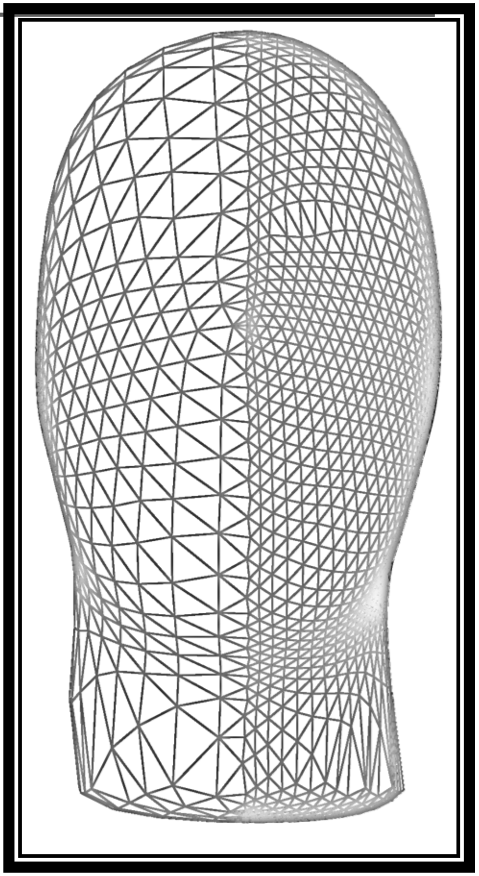
Initial Mesh



umbrella operator



Fujiwara operator



Laplace Beltrami operator applied to

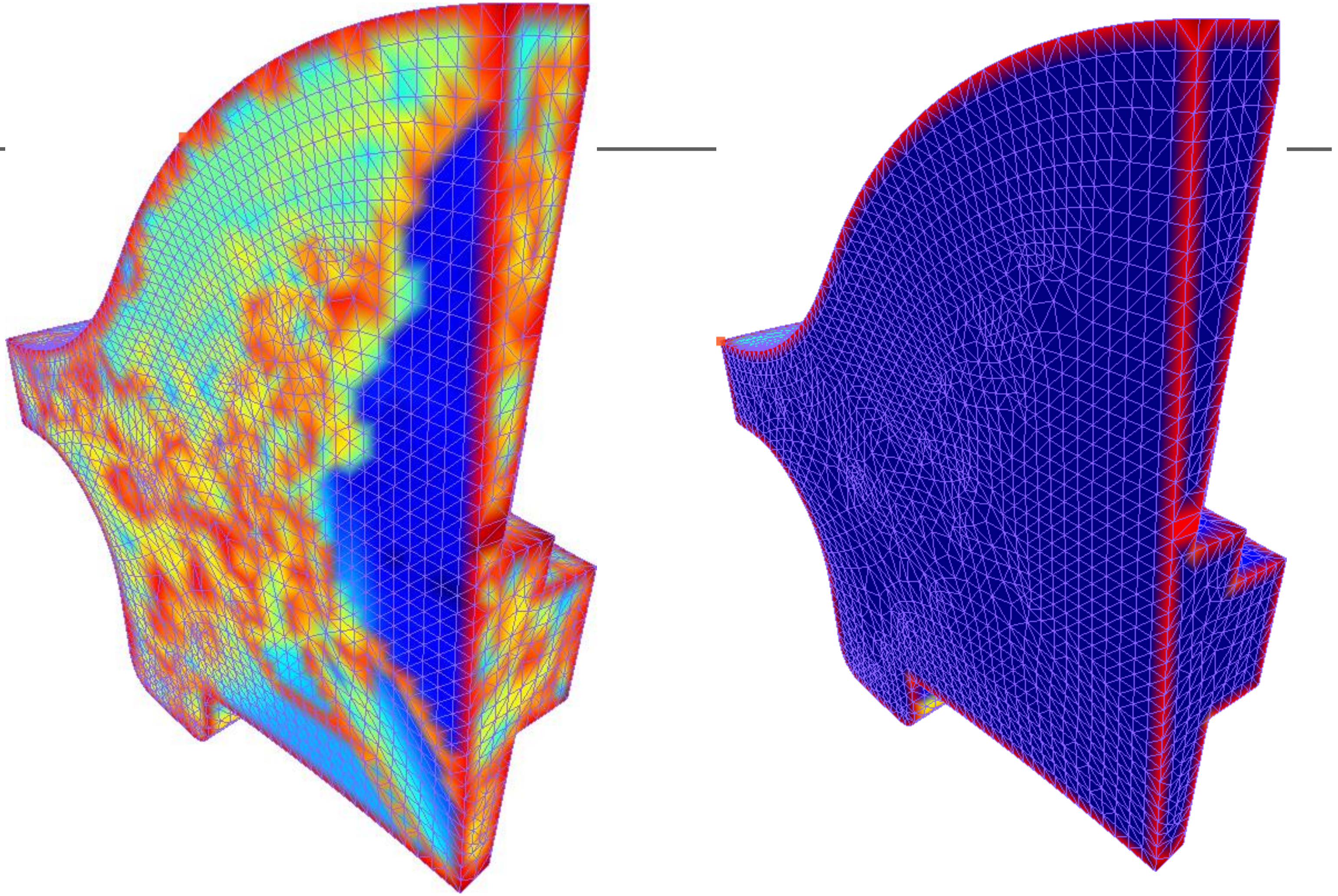
the coordinate function X = (x,y,z)

Desbrun et al., Siggraph 1999 Desbrun et al., Siggraph 1999

Discrete Laplacian

Note that since the Laplacian measures the difference between a vertex and the average values of its neighbors, we expect that is equal to zero on constant functions c, so:

$$\Rightarrow Lc = 0$$



Results of applying **umbrella** weights on the left and **cotangent** weights on the right.

Material

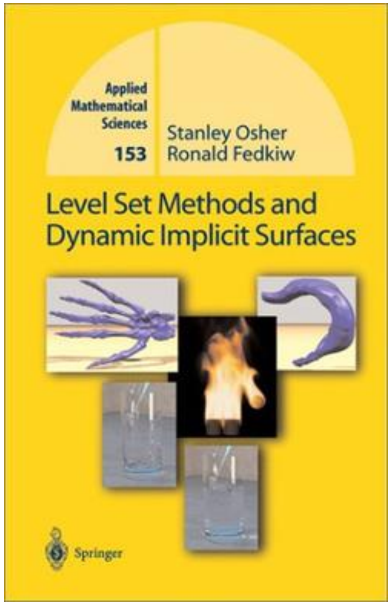
Level Set Methods and Dynamic Implicit Surfaces

Stanley Osher & Ronald Fedkiw

Series: Applied Mathematical Sciences, Vol. 153 Springer, 2003

ISBN-10: 0-387-95482-1

ISBN-13: 978-0-387-95482-0



Polygon Mesh Processing

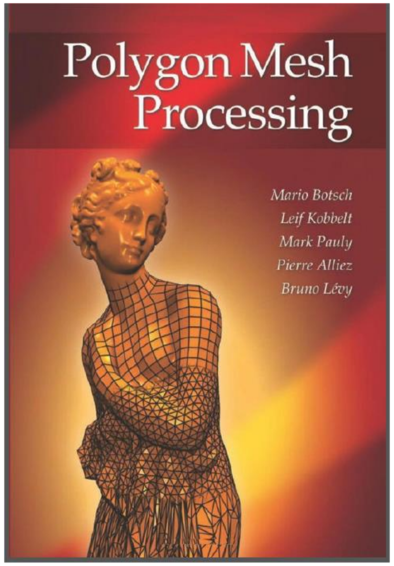
Mario Botsch, Leif Kobbelt

Mark Pauly, Pierre Alliez, Bruno Levy

A K Peters, Ltd.

Natick, Massachusetts 2008

ISBN 978-1-56881-426-1



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Serena Morigi

Dipartimento di Matematica

serena.morigi@unibo.it

http://www.dm.unibo.it/~morigi