Introduction to meshes processing

Ruding LOU
ruding.lou@ensam.eu

Arts et Métiers

References

- Book
  - « Polygon Mesh Processing » by Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, Bruno Levy
- Slides
  - Eurographics course notes « Geometric Modeling Based on Triangle Meshes » by Mario Botsch, Mark Pauly, Christian Rössl, Stephan Bischoff, Leif Kobbelt
- Numerous documents available on web site
  - Courses
  - Papers

Objective

- Analysis and manipulation of meshes
  - Tools and methods
  - Algorithms and programs
- Table of contents
  - Basics
  - Simplification
  - Subdivision
  - Deformation
  - Alignment
  - Smoothing
  - Hole-filling

Introduction to meshes processing

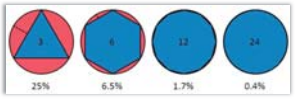
Mesh basics

Ruding LOU
ruding.lou@ensam.eu

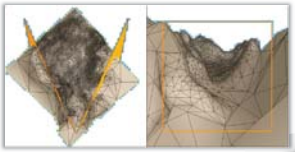
Arts et Métiers

Polygonal meshes

- Piecewise linear approximation
  - Approximation error: $O(h^2)$

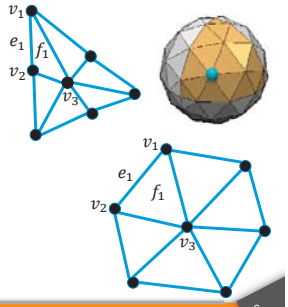


- A good representation
  - Arbitrary topology
  - Piecewise smooth surfaces
  - Adaptive refinement
  - Efficient rendering



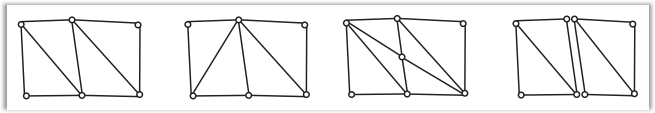
Definitions: Mesh

- Mesh connectivity: graph
  - A **polygonal mesh** consists of three kinds of **mesh elements**: vertices, edges, faces.
  - A **triangle mesh** consists of three kinds of mesh elements: vertices, edges, **triangles**.
    - $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$
    - $\mathcal{E} = \{e_1, e_2, \dots, e_k\}, \quad e_i \in \mathcal{V} \times \mathcal{V}$
    - $\mathcal{F} = \{f_1, f_2, \dots, f_m\}, \quad f_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$
- Mesh geometry: vertex positions
  - $\boldsymbol{\mathcal{p}} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n\}, \quad \mathbf{p}_i \in \mathbb{R}^3$

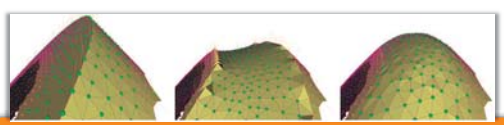


Definitions: Mesh

- The information describing the mesh elements are ***mesh connectivity*** and ***mesh geometry***.
- The **mesh connectivity**, or topology, describes the incidence relations among mesh elements.

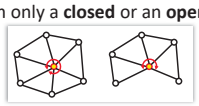


- The **mesh geometry** specifies the position and other geometric characteristics of each vertex.

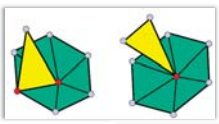


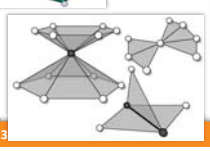
Definitions: Topology

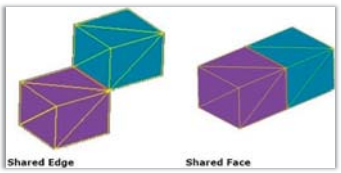
- A mesh is **manifold** if and only if
  each edge is incident to only one or two faces and
  the faces incident to a vertex form only a **closed** or an **open** fan.



- The following examples are **Non-Manifold** meshes !

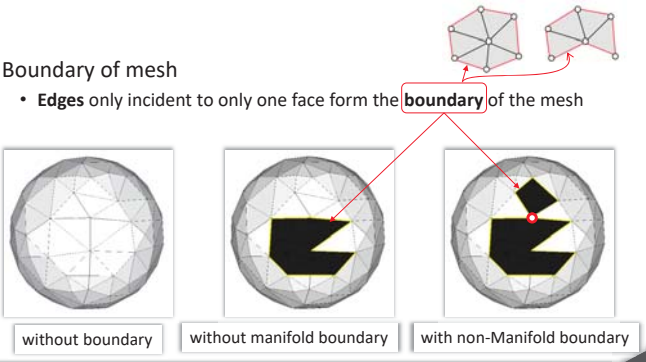






Definitions: Topology

- Manifold mesh **without boundary**
  - If every vertex has a closed fan, the given manifold has no boundary.
- Boundary of mesh
  - **Edges** only incident to only one face form the **boundary** of the mesh

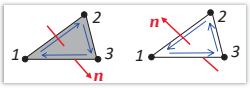


without boundary | without manifold boundary | with non-Manifold boundary

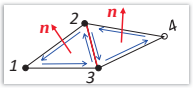
Definitions: Surface orientation

- The **orientation** of a face is a cyclic order of the incident vertices.

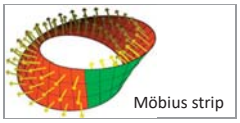
  Face orientation: clockwise or anticlockwise order in which the vertices listed. *e.g.* 1→2→3 or 1→3→2



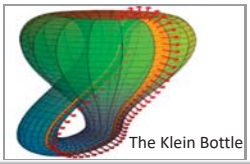
- The orientation of two adjacent faces is **compatible**, if the two vertices of the common edge are in opposite order.



- A manifold mesh is **orientable** if any two adjacent faces have compatible orientation.
- Not all manifolds are orientable. The most well-known ones are Möbius band and Klein bottle



Möbius strip



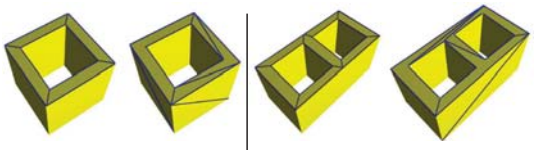
The Klein Bottle

Definitions: Euler characteristic

- Given a 2-**manifold** mesh $\boldsymbol{M}$ without boundary, the **Euler-Poincaré characteristic** of M is $\chi(\boldsymbol{M}) = \boldsymbol{V} - \boldsymbol{E} + \boldsymbol{F}$, where $\boldsymbol{V}$, $\boldsymbol{E}$ and $\boldsymbol{F}$ are the number of vertices, number of edges, and number of faces.

V=8, E=12, F=6 χ(M)=2 | V=16, E=32, F=16 χ(M)=0 | V=28, E=56, F=26 χ(M)=-2

- Euler-Poincaré characteristic $\chi(\boldsymbol{M}) = \boldsymbol{V} - \boldsymbol{E} + \boldsymbol{F}$ is independent of tessellation.

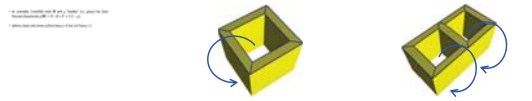


V=16, E=32, F=16 χ(M)=0 | V=16, E=36, F=20 χ(M)=0 | V=24, E=48, F=22 χ(M)=-2 | V=28, E=56, F=26 χ(M)=-2

Definitions: Euler characteristic

- An orientable 2-manifold mesh $\boldsymbol{M}$ with g "handles" (i.e., genus) has Euler-Poincaré characteristic $\chi(\boldsymbol{M}) = \boldsymbol{V} - \boldsymbol{E} + \boldsymbol{F} = 2\,(1 - g)$.
- Spheres, boxes and convex surfaces have g = 0; but, tori have g = 1.

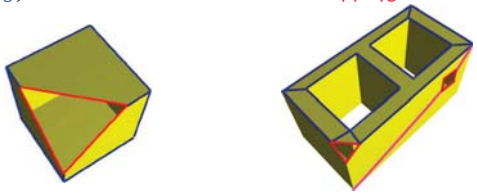
(a) sphere (b) torus (c) 2-torus



g = 0 $\chi(M)=2(1-0)=2$ | g = 1 $\chi(M)=2(1-1)=0$ | g = 2 $\chi(M)=2(1-2)=-2$

Definitions: Euler characteristic

- The **boundary** of an orientable 2-manifold is the union of a set of simple polygons. Since each polygon bounds a face, these "boundary faces" may be added back to form a manifold without boundary so that Euler-Poincaré characteristic can be applied.
- The Euler-Poincaré characteristic of an orientable 2-manifold with boundary is $\chi(\boldsymbol{M}) = 2\,(1 - g) - \partial$ where ∂ is the number of boundary polygons.



V=10, E=15, F=6
g = 0, ∂ = 1
$\chi(M)=V-E-F=1$
$\chi(M)=2(1-0)-1=1$

V=30, E=54, F=20
g = 2, ∂ = 2
$\chi(M)=V-E-F=-4$
$\chi(M)=2(1-0)-2=-4$

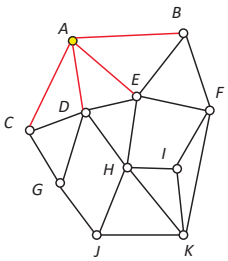
Definitions: Vertex degree

Vertex degree or valence

- number of incident edges
- *e.g.* $\deg(A) = 4,\ \ \deg(E) = 5$

Regular mesh

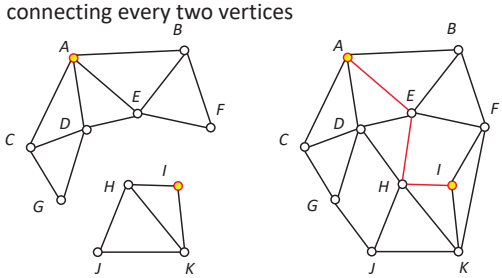
- All vertex degrees are equal



Definitions: Connected mesh

• Connected mesh

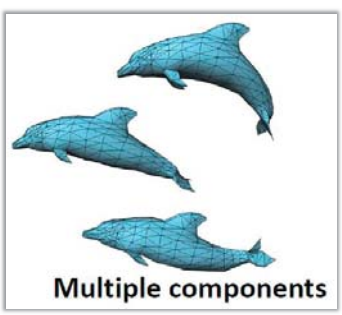
path of edges connecting every two vertices

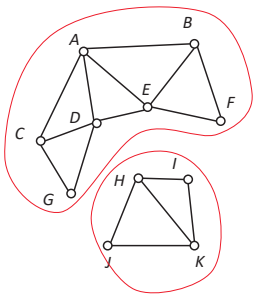


Definitions: Connected mesh

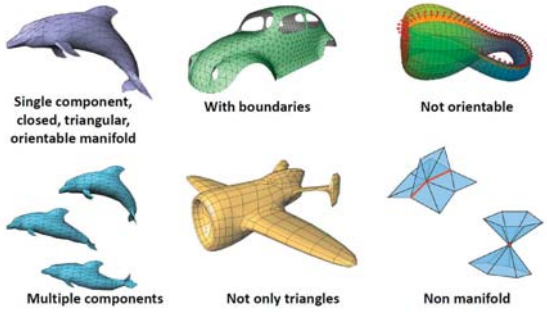
• Connected component

Maximally connected sub-mesh



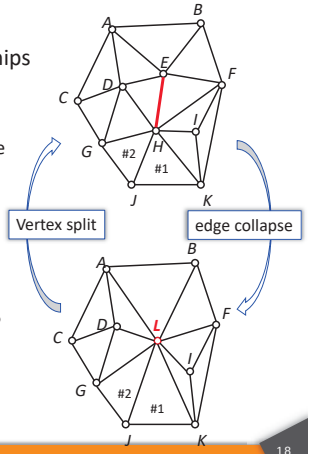


Mesh Zoo



Mesh data structures

- What should be stored?
  - Connectivity: Adjacency relationships
  - Geometry: 3D coordinates
  - Attributes
    - e.g. normal, color, texture coordinate
    - per vertex, per face, per edge
- What we can do ?
  - Connectivity queries
    - What are the vertices of face #2?
    - Is vertex A adjacent to vertex H ?
    - Which faces are adjacent to face #1?
  - Modifications
    - Remove/add a vertex/face
    - Vertex split, edge collapse



Mesh data structures

How good is a data structure?

- Time to construct (preprocessing)
- Time to answer a query
- Time to perform an operation
- Space complexity
- Redundancy

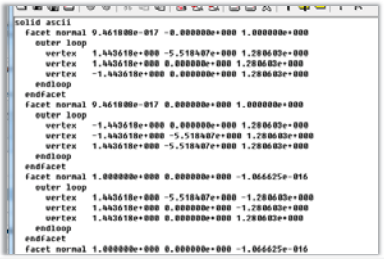
Mesh data structures

- Face Set
- Shared Vertex
- Half Edge
- Face Based Connectivity
- Edge Based Connectivity
- Adjacency Matrix
- Corner Table

Mesh data structures – Face Set

- Simple
- STL file format
- No connectivity
- Redundancy

faces	v_1	v_2	...	v_k
1	(x,y,z)	(x,y,z)	...	(x,y,z)
2	(x,y,z)	(x,y,z)		(x,y,z)
...	...	...	...	...
n	(x,y,z)	(x,y,z)	...	(x,y,z)



The **orientation** of each face depends on the order of the 3 vertices

Mesh data structures – Shared Vertex

Indexed face set (Shared Vertex)

- Faces list vertex references – " Shared vertices "
- Commonly used (e.g. OFF file format itself)
- Connectivity
- No neighborhood

	Vertices
1	(x_1, y_1, z_1)
2	(x_2, y_2, z_2)
...	...

faces			
Vertex index	Vertex index	...	Vertex index
2	1	...	3
...	...	...	...

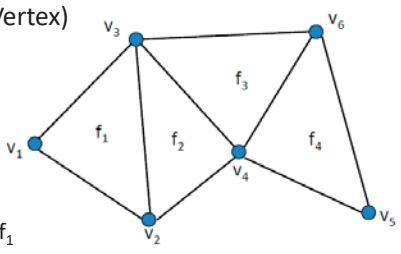
The **orientation** of each face depends on the order of the 3 vertices

Mesh data structures – Shared Vertex

Indexed face set (Shared Vertex)

Vertices
(x_1, y_1, z_1)
(x_2, y_2, z_2)
...

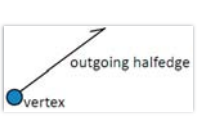
Faces		
3	1	2
3	2	4
...	...	...

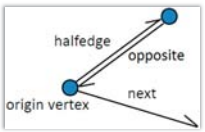


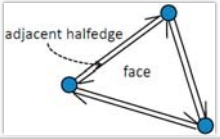
- What are the vertices of f_1
  - First triplet from face list
- What are the one-ring neighbors (faces) of v_3?
  - Requires a full pass over all faces
- Are vertices v_1 and v_5 adjacent?
  - Requires a full pass over all faces

Mesh data structures – Half Edge

- Vertex stores
  - Position
  - 1 outgoing halfedge
- Halfedge stores
  - 1 incident vertex index (origin vertex or the vertex it points to)
  - 1 incident face index
  - next, previous, opposite halfedge indices
- Face stores
  - 1 incident halfedge index



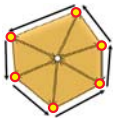




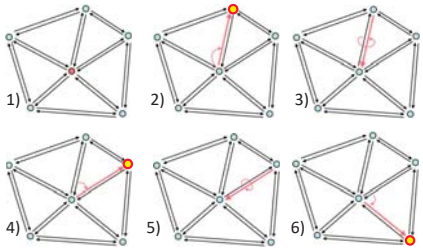
Mesh data structures – Half Edge

- **One-Ring Traversal**

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



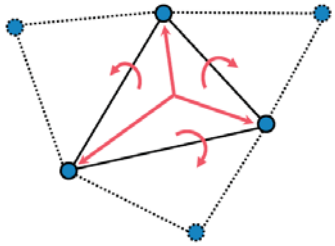
Vertex	Hafedge	Face
- position - outgoing half edge	- incident vertex - incident face - next halfedge - previous halfedge - opposite halfedge	- Incident - halfedge



Mesh data structures – Face Based Connectivity

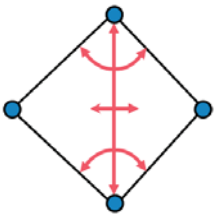
- Vertex:
  - position
  - 1 adjacent face index
- Face:
  - 3 vertex indices
  - 3 neighboring face indices

- No (explicit) edge information



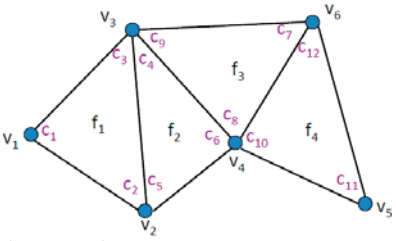
Mesh data structures – Edge Based Connectivity

- Vertex:
  - position
  - 1 adjacent edge index
- Edge:
  - 2 vertex indices
  - 2 neighboring face indices
  - 4 edges
- Face
  - 1 edge index



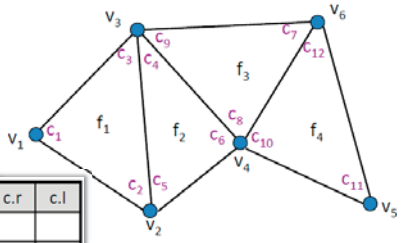
Mesh data structures – Corner Table

- Corner is a vertex with one of its incident triangles
  - Corner – **c** (*e.g.* c_1)
  - Triangle – **c.t** (*e.g.* f_1)
  - Vertex – **c.v** (*e.g.* v_1)
  - Next corner in c.t – **c.n** (*e.g.* c_2)
  - Previous corner in c.t – **c.p** (*e.g.* c_3) (== c.n.n)
  - Corner opposite c – **c.o** (*e.g.* c_6)
  - Edge **E** opposite c not incident on c.v (*e.g.* edge v_2-v_3)
  - Triangle **T** adjacent to c.t across **E** (*e.g.* f_2)
  - c.o.v vertex of **T** that is not incident on **E** (*e.g.* v_4)
  - Right corner – **c.r** – corner opposite c.n (== c.n.o)
  - Left – **c.l** (== c.p.o == c.n.n.o)



Mesh data structures – Corner Table

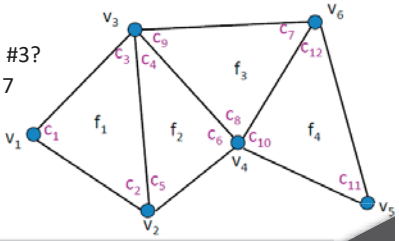
- Corner is a vertex with one of its incident triangles



corner	c.v	c.t	c.n	c.p	c.o	c.r	c.l
c_1	v_1	f_1	c_2	c_3	c_6		
c_2	v_2	f_1	c_3	c_1			c_6
c_3	v_3	f_1	c_1	c_2		c_6	
c_4	v_3	f_2	c_5	c_6		c_7	c_1
c_5	v_2	f_2	c_6	c_4	c_7	c_1	
c_6	v_4	f_2	c_4	c_5	c_1		c_7

Mesh data structures – Corner Table

- Store:
  - Corner table
  - For each vertex – a list of all its corners
- Corner number j*3 – 2, j*3 – 1 and j*3 match face number j
  - e.g. c_4 , c_5 and c_6 for face f_2
- What are the vertices of face #3?
  - Check c.v of corners 9, 8, 7



Master 2 Recherche MTI 3D

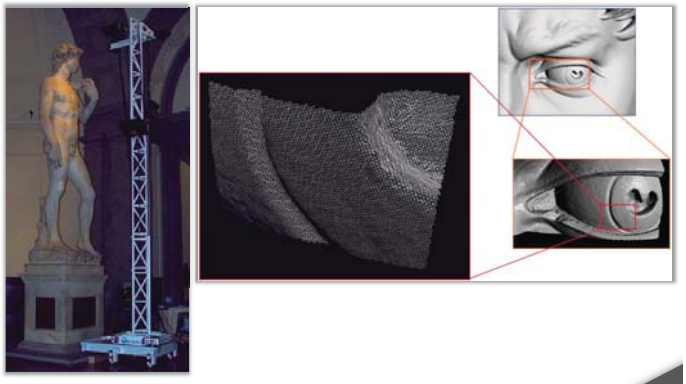
Introduction to meshes processing

Mesh simplification

Ruding LOU
ruding.lou@ensam.eu

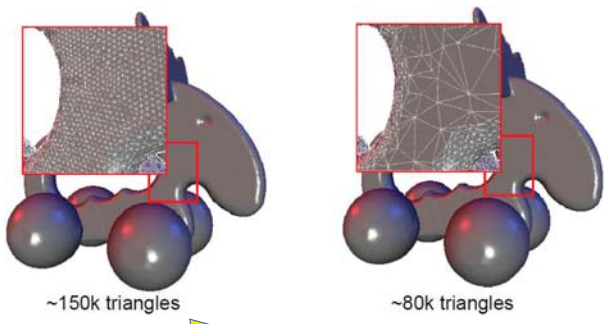
Applications

- Oversampled 3D scan data



Applications

- Oversampled 3D scan data



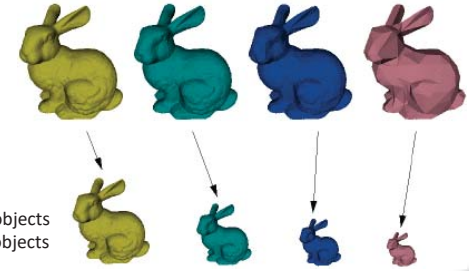
Same density everywhere

Applications

- Adaptation to hardware capabilities

Applications

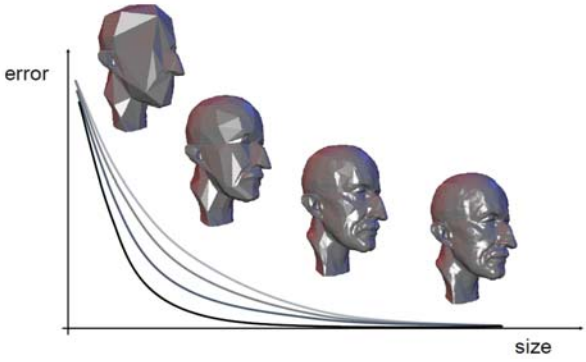
- Multi-resolution hierarchies for
  - reducing information content
  - efficient geometry processing / rendering
  - level-of-detail (LOD) rendering



✓ Refined mesh for close objects
✓ Simplified mesh for far objects

Applications

- Size-Quality Tradeoff

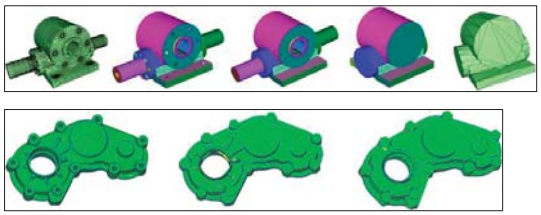


Applications

- Performance Requirements
  - Offline
    - Generate model at given level(s) of detail
    - Emphasis on quality
  - Real-time
    - Generate model at given level(s) of detail
    - Emphasis on **speed**
    - **Requires preprocessing**
    - **Time/space/quality tradeoff**

Categories of simplification

- **Decimation**: Reducing the number of elements
- **Idealization** : Removing features / Topology simplifying



Problem statement

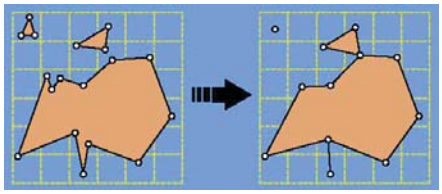
- Given: $\mathcal{M} = (\mathcal{V}, \mathcal{F})$
- Find: $\mathcal{M}' = (\mathcal{V}', \mathcal{F}')$ such that
  - $|\mathcal{V}'| < |\mathcal{V}|$ and $\|\mathcal{M} - \mathcal{M}'\|$ is minimal, or
  - $\|\mathcal{M} - \mathcal{M}'\| < \epsilon$ and $|\mathcal{V}'|$ is minimal
- Respect additional fairness criteria
  - normal deviation, triangle shape, scalar attributes, etc.

common scalar attributes include color, texture coordinates

Mesh decimation methods

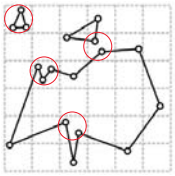
- **Vertex clustering**
  - Cluster Generation
  - Computing a representative
  - Mesh generation
  - ~~Topology changes~~
- Incremental decimation

- Cluster Generation
  - Uniform 3D grid
  - Merge all vertices within the same cell

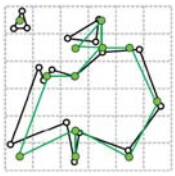


Rossignac, J. and Borrel, P. (1993). Multi-resolution 3d approximations for rendering complex scenes. In *Geometric Modeling in Computer Graphics*, Genova, Italy, pp.455–465.

- Cluster Generation
  - Uniform 3D grid
    - Partition space into cells
    - Map vertices to cluster cells
  - Merge all vertices within the same cell
    - triangles with multiple corners in one cell will degenerate
    - Optimal position for the representative in each cell

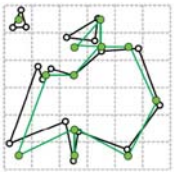


clustering

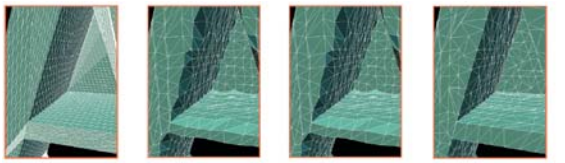


merging

- Cluster Generation
- Computing a representative
  - Cell center
  - Old vertices position
  - Average/median vertex position
  - Error quadrics
- Mesh generation
- ~~Topology changes~~

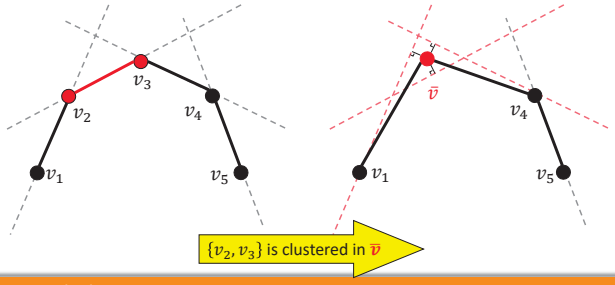


- Computing a representative
  - Average/median vertex position
  - Error quadrics

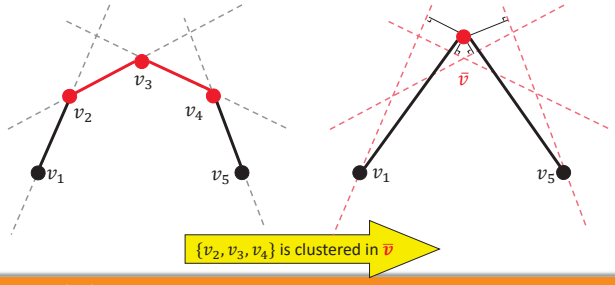


Average vertex position Median vertex position Error quadrics

- Error Quadrics
  - Patch is expected to be piecewise flat
  - Minimize distance to neighboring triangles' planes

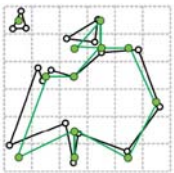


- Error Quadrics
  - Patch is expected to be piecewise flat
  - Minimize distance to neighboring triangles' planes



- Error Quadrics
- Measuring Error with Planes
- Why base error on planes?
  - Faster, but less accurate, than distance-to-surface
  - Simple linear system for minimum-error position
  - Drawback: unlike surface, planes are infinite

- Cluster Generation
- Computing a representative
- Mesh generation
  - Clusters $p \leftrightarrow \{p_0, p_1, \ldots, p_n\}$, $q \leftrightarrow \{q_0, q_1, \ldots, q_n\}$
  - Connect (p, q) if there was an edge between (p_i, q_j)
- ~~Topology changes~~



Mesh decimation methods

- Vertex clustering
- 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

General Setup

Greedy Optimization

- For each region
  - evaluate quality after decimation
  - enqueue (quality, region)
- End each
- Repeat:
  - get best mesh region from queue
  - apply decimation operator
  - update queue
- Until no further reduction possible

General Setup

Greedy Optimization with error control

- For each region
  - evaluate quality after decimation
  - enqueue (quality, region)
- End each
- Repeat:
  - get best mesh region from queue
  - if error < ε
    - apply decimation operator
    - update queue
- Until no further reduction possible

Mesh decimation methods

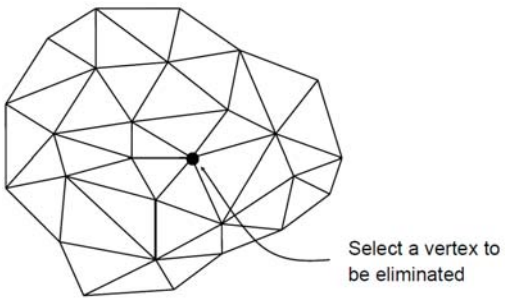
- Vertex clustering
- Incremental decimation
  - General Setup
  - **Decimation operators**
  - Error metrics
  - Fairness criteria
  - ~~Topology changes~~

Decimation operators

- What is a "region" ?
- What are the DOF for re-triangulation?
- Classification
  - Subsampling vs. filtering
  - Inverse operation → progressive meshes

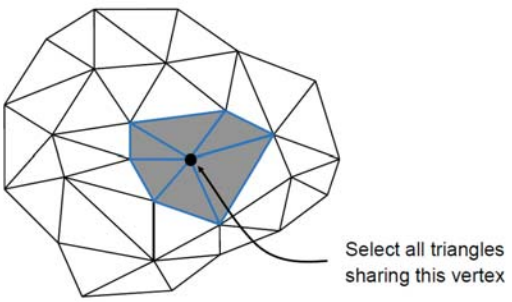
Decimation operators

- Vertex Removal

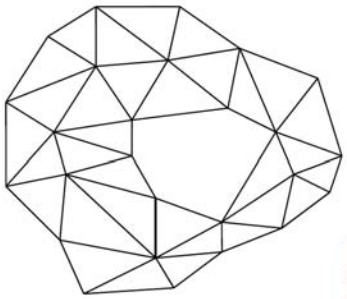


Decimation operators

- Vertex Removal

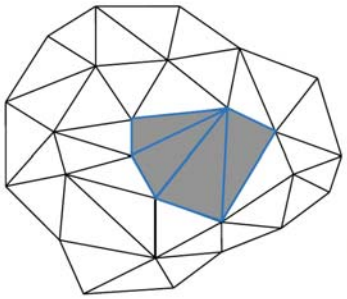


- Vertex Removal



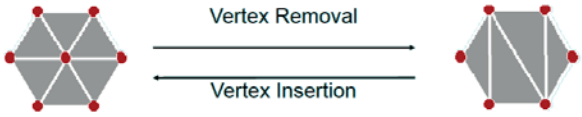
Remove the selected triangles, creating the hole

- Vertex Removal



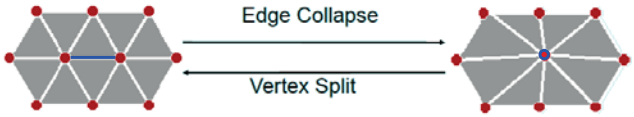
Fill the hole with new triangles

- Vertex Removal



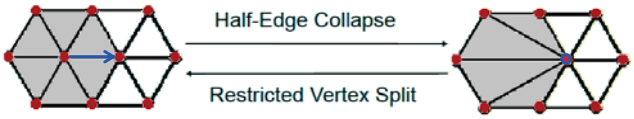
  - Sequence of local operations
    - Remove vertex, re-triangulate hole
      - Combinatorial degrees of freedom
      - Involve near neighbors - only small patch affected in each operation
    - Each operation introduces error
      - Find and apply operation which introduces the least error
    - Remaining vertices are subset of original vertex set
    - Number of vertices and faces: $\begin{cases} v = v - 1 \\ f = f - 2 \end{cases}$

- Edge Collapse



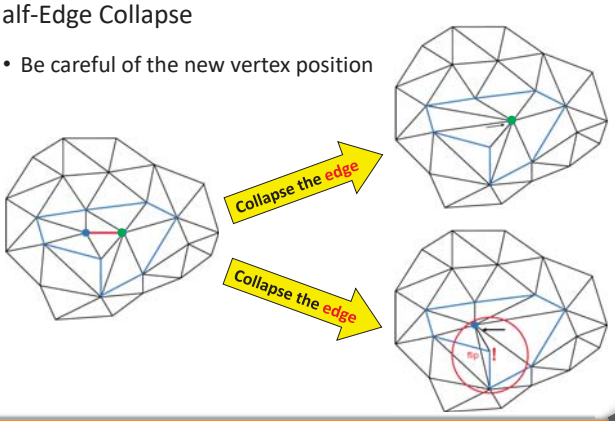
  - Merge two adjacent vertices
  - Introduces new vertices not present in the original mesh
  - Define new vertex position
    - Continuous degrees of freedom
  - Number of vertices and faces: $\begin{cases} v = v - 1 \\ f = f - 2 \end{cases}$

- Half-Edge Collapse

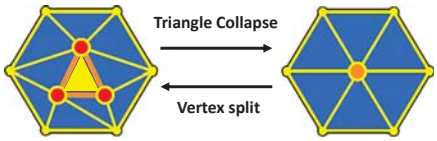


- Collapse edge into one end point
  - Special case of vertex removal
  - Special case of edge collapse
- No degrees of freedom

- Half-Edge Collapse
  - Be careful of the new vertex position



- Triangle Collapse



  - Merge three vertices in a triangle
  - Define new vertex position
    - Continuous degrees of freedom
  - Number of vertices and faces: $\begin{cases} v = v - 2 \\ f = f - 4 \end{cases}$

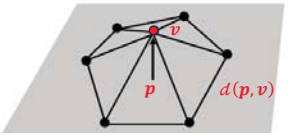
- Vertex clustering
- Incremental decimation
  - General Setup
  - Decimation operators
  - **Error metrics**
  - Fairness criteria
  - ~~Topology changes~~

Local Error Metrics

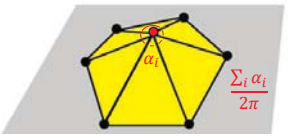
- **Local** error: Compare new patch with previous iteration
  - Fast
  - Accumulates error
  - Memory-less
- **Global** error: Compare new patch with original mesh
  - Slow
  - Better quality control
  - Can be used as termination condition
  - Must remember the original mesh throughout the algorithm

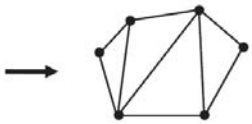
Local Error Metrics

- Local distance to mesh
  - Compute distance to average plane



  - Compute discrete curvature





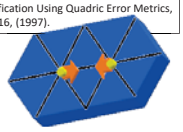
  - No comparison to original geometry

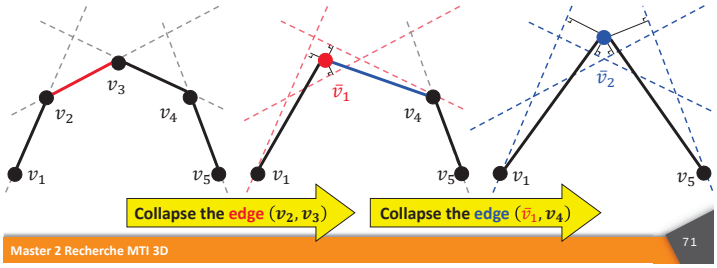
Global Error Metrics

- Error quadrics
  - Squared distance to planes at vertex
  - No bound on true error, **pseudo-global**
  - All original planes persist during the whole simplification process

Edge Collapse Algorithm - **QSlim**

M. Garland and P. S. Heckbert, Surface Simplification Using Quadric Error Metrics, In *Proceedings of SIGGRAPH 97*, pages 209–216, (1997).



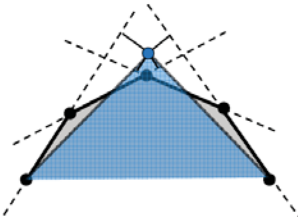


Mesh decimation methods

- Vertex clustering
- Incremental decimation
  - General Setup
  - Decimation operators
  - Error metrics
  - **Fairness criteria**
  - ~~Topology changes~~

Fairness criteria

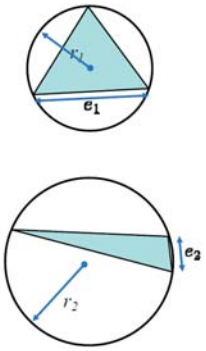
- Rate quality of decimation operation
  - **Approximation error**
  - Triangle shape
  - Dihedral angles
  - Valence balance
  - Other semantics
  - …



Fairness criteria

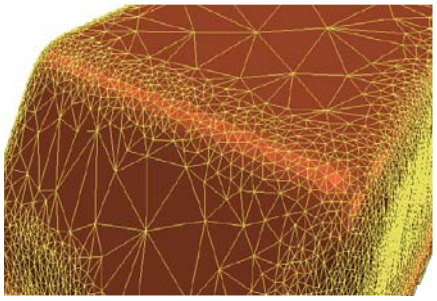
- Rate quality of decimation operation
  - Approximation error
  - **Triangle shape**
  - Dihedral angles
  - Valence balance
  - Other semantics
  - …

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



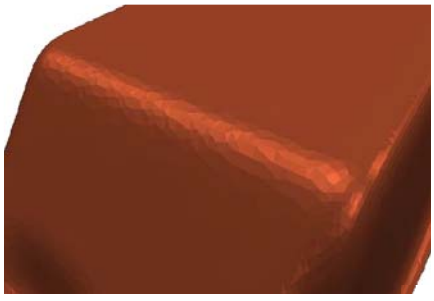
Fairness criteria

- Rate quality of decimation operation
  - Approximation error
  - **Triangle shape**
  - Dihedral angles
  - Valence balance
  - Other semantics
  - …

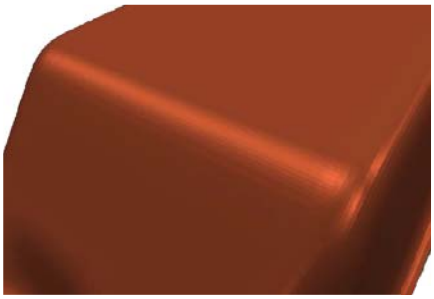


Fairness criteria

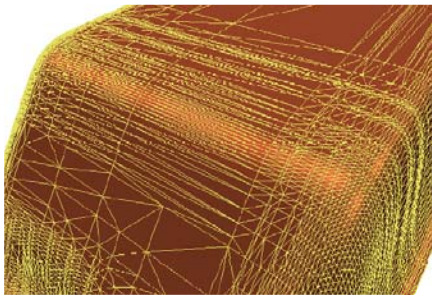
- Rate quality of decimation operation
  - Approximation error
  - **Triangle shape**
  - Dihedral angles
  - Valence balance
  - Other semantics
  - …



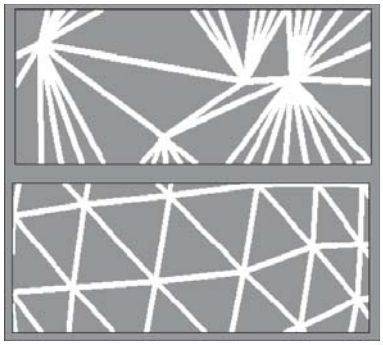
- Rate quality of decimation operation
  - Approximation error
  - Triangle shape
  - **Dihedral angles**
  - Valence balance
  - Other semantics
  - …



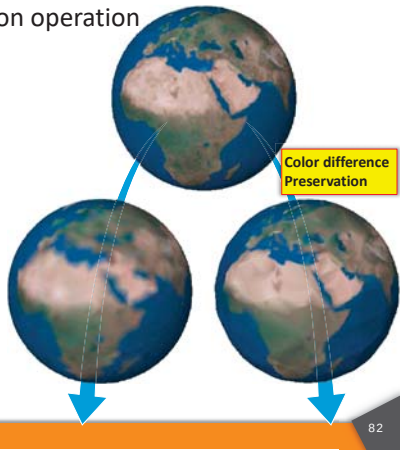
- Rate quality of decimation operation
  - Approximation error
  - Triangle shape
  - **Dihedral angles**
  - Valence balance
  - Other semantics
  - …



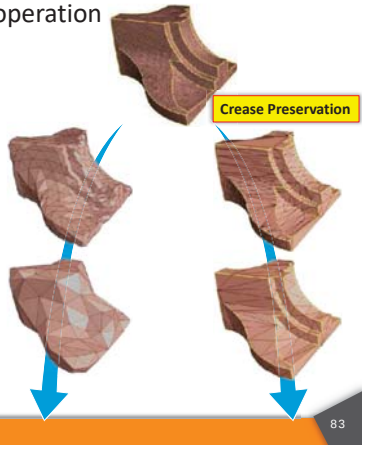
- Rate quality of decimation operation
  - Approximation error
  - Triangle shape
  - Dihedral angles
  - **Valence balance**
  - Other semantics
  - …



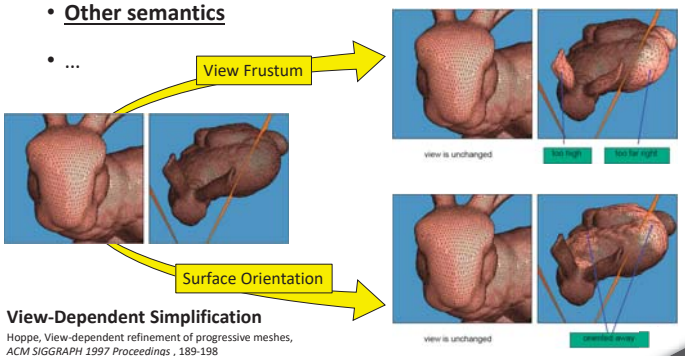
- Rate quality of decimation operation
  - Approximation error
  - Triangle shape
  - Dihedral angles
  - Valence balance
  - **Other semantics**
  - …



- Rate quality of decimation operation
  - Approximation error
  - Triangle shape
  - Dihedral angles
  - Valence balance
  - **Other semantics**
  - …



- Rate quality of decimation operation
  - **Other semantics**
  - …



View-Dependent Simplification

Hoppe, View-dependent refinement of progressive meshes, *ACM SIGGRAPH 1997 Proceedings* , 189-198

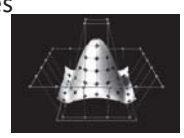
- Vertex clustering
  - fast, but difficult to control simplified mesh
  - ~~topology changes, non-manifold meshes~~
  - global error bound, but often not close to optimum
- Incremental decimation with quadric error metrics
  - good trade-off between mesh quality and speed
  - ~~explicit control over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

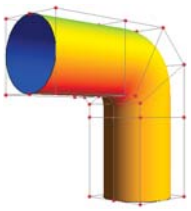
Introduction to meshes processing

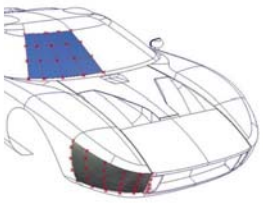
Subdivision Surfaces

Ruding LOU
ruding.lou@ensam.eu

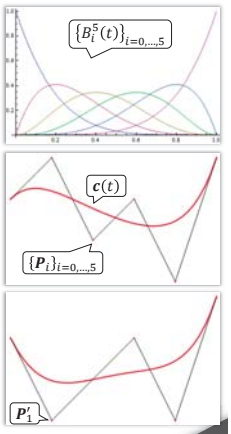
- Sometimes need more than polygon meshes
  - Smooth surfaces
- Traditional geometric modeling
  - Bézier
  - B-Spline (Basis Spline)
  - NURBS (Non-uniform rational basis spline)







- **Bézier curve**
  - Control points
    $\{ \boldsymbol{P}_i \}_{i=0,1,\dots,n}$
  - Bernstein basis polynomials of degree n
    $\left\{ B_i^n(t) = \binom{n}{i}(1-t)^{n-i} \right\}_{i=0,1,\dots,n}$
  - Bézier curve
    $\boldsymbol{c}(t) = \sum_{i=0}^{n} B_i^n(t)\,\boldsymbol{P}_i \quad t \in [0,1]$
  - Adjustment of control points
    - shape of the curve
    - global influence



- **B-spline curve**
  - Control points $\{ \boldsymbol{P}_i \}_{i=0,1,\dots,n}$
  - Spline basis polynomials of degree p

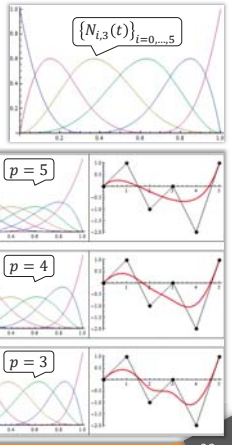
$$N_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \le t \le t_{i+1} \\ 0 & \text{else} \end{cases}$$

$$N_{i,p}(t) = \frac{t - t_i}{t_{i-p} - t_i} N_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(t)$$

  - B-spline curve

$$\boldsymbol{s}(t) = \sum_{i=0}^{n} N_{i,p}(t)\,\boldsymbol{P}_i$$

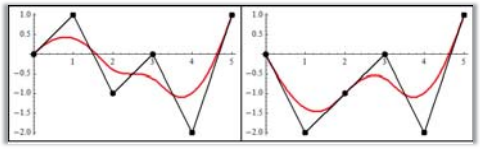
  - Degree p of basis polynomials
    - $p = n$: Bézier curve
    - $p = 1$: Control polygon

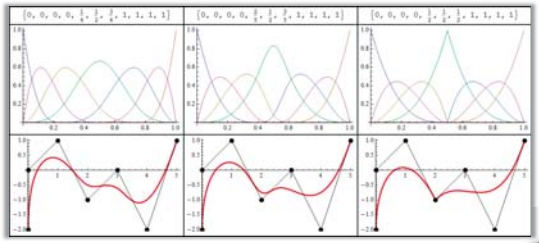


- Cubic B-spline curve

Demo on line: http://geometrie.foretnik.net/files/NURBS-en.swf

  - Adjustment
    - control points
    - knot vector



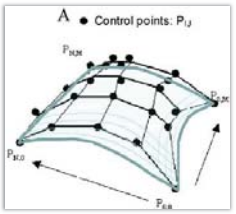


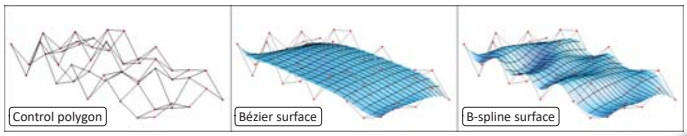
- **Bézier and B-spline surface**
  - Control points $\{ \boldsymbol{P}_{i,j} \}_{\substack{i=0,1,\dots,m \\ j=0,1,\dots,n}}$
  - Bézier surface of bidegree (m, n)

$$\boldsymbol{c}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} B_i^m(u) B_j^n(v)\,\boldsymbol{P}_{i,j} \quad u,v \in [0,1]$$

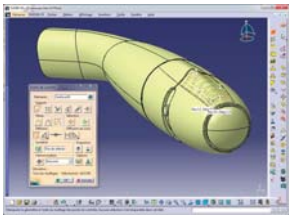
  - B-spline surface of bidegree (k, l)

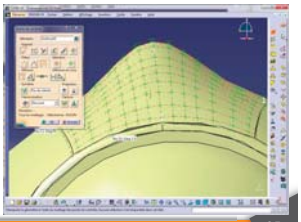
$$\boldsymbol{s}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} B_i^k(u) B_j^l(v)\,\boldsymbol{P}_{i,j} \quad u,v \in [0,1]$$

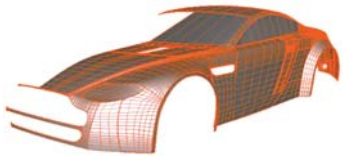




- A single B-Spline patch is either a topological disk, a tube or a torus
- Must use many B-Spline
  - patches to model
  - complex geometry
- When deforming a surface made of B-Spline patches, cracks arise at the seams

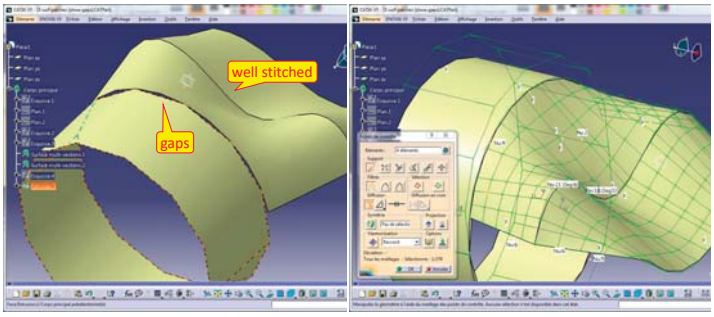




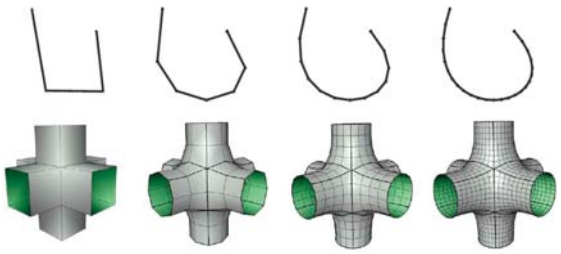


- Examples of gaps between the surface patches

Surface patches | control polygons

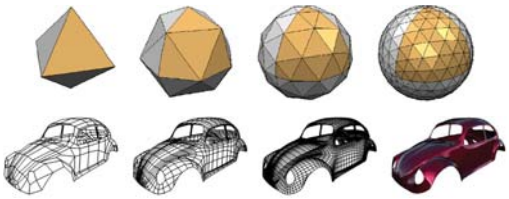


- Subdivision defines a smooth curve or surface as the limit of a sequence of successive refinements



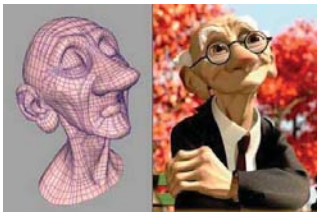
Subdivision Surfaces

- Generalization of spline curves / surfaces
  - Arbitrary control meshes
  - Successive refinement (subdivision)
  - Converges to smooth limit surface
  - Connection between splines and meshes



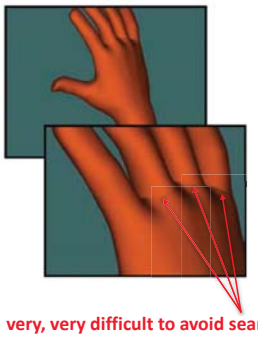
Example: Geri's Game (Pixar), 1997

- Subdivision used for
  - Geri's hands and head
  - Clothing
  - Tie and shoes



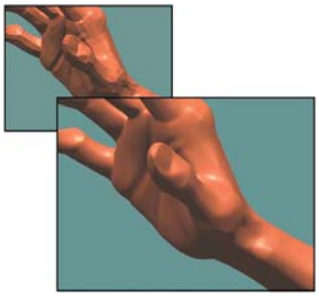
Example: Geri's Game (Pixar)5, 1997

- Woody's hand (NURBS) from *Pixar's Toy Story*



- very, very difficult to avoid seams

- Geri's hand (subdivision) from *Pixar's Geri's Game*



- **solves the "stitching" problem**
- **a single smooth surface is defined**

Example: Geri's Game (Pixar)

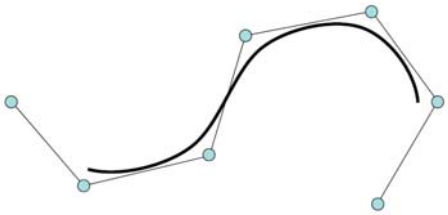
- Sharp and semi-sharp features

Subdivision Curves

- Given a control polygon…

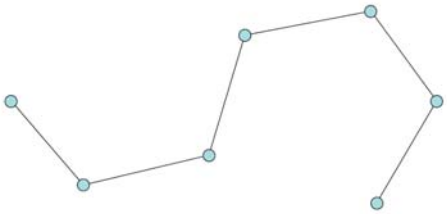
  ... find a smooth curve related to that polygon.

- Subdivision Curve Types
  - Approximating
  - Interpolating



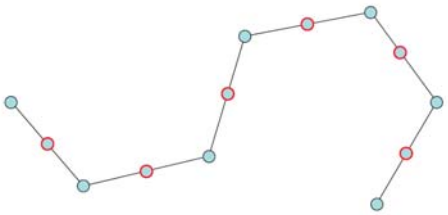
Approximating

- Given a control polygon



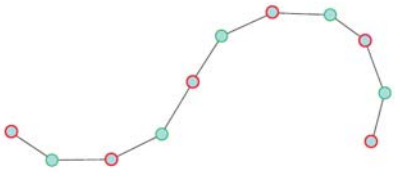
Approximating

- Given a control polygon
  - splitting step: split each edge in two



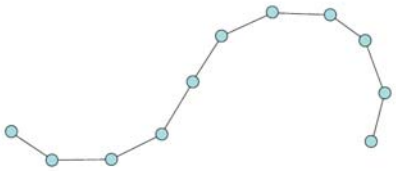
Approximating

- Given a control polygon
  - splitting step: split each edge in two
  - averaging step: relocate each (original) vertex according to some (simple) rule...



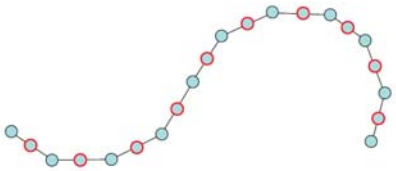
Approximating

- Given a control polygon
  - splitting step: split each edge in two
  - averaging step: relocate each (original) vertex according to some (simple) rule...
- start over ...



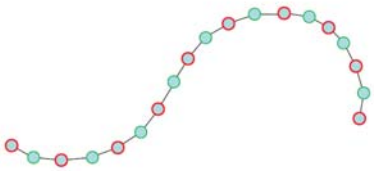
Approximating

- Given a control polygon
  - splitting step: split each edge in two
  - averaging step: relocate each (original) vertex according to some (simple) rule...
- start over ...
  - splitting ...



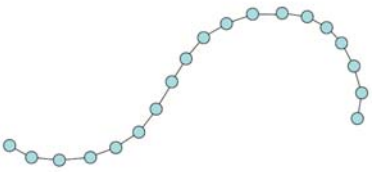
Approximating

- Given a control polygon
  - splitting step: split each edge in two
  - averaging step: relocate each (original) vertex according to some (simple) rule...
- start over ...
  - splitting ...
  - averaging ...



Approximating

- Given a control polygon
  - splitting step: split each edge in two
  - averaging step: relocate each (original) vertex according to some (simple) rule...
- start over ...
  - splitting ...
  - averaging ...
- and so on...



Approximating

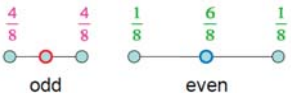
- If the rule is designed carefully...
  - the control polygons will converge to a smooth limit curve !

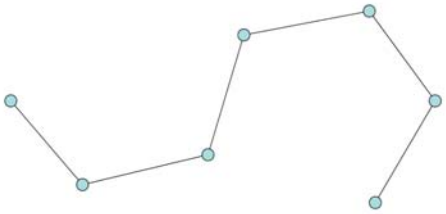
Equivalent to ...

- Insert <u>single</u> new point at mid-edge
- <u>Filter</u> entire set of points.
  - e.g. Catmull-Clark rule: Filter with (1/8, 6/8, 1/8)

Approximating: B-spline curves

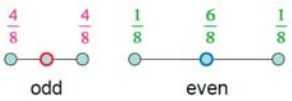
- Cubic B-spline (Catmull-Clark)

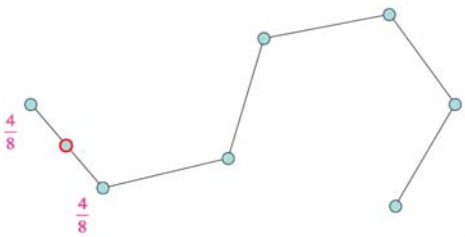




Approximating: B-spline curves

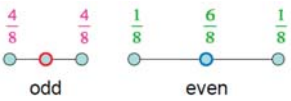
- Cubic B-spline (Catmull-Clark)

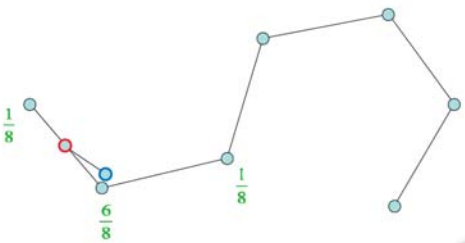




Approximating: B-spline curves

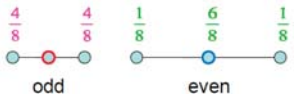
- Cubic B-spline (Catmull-Clark)

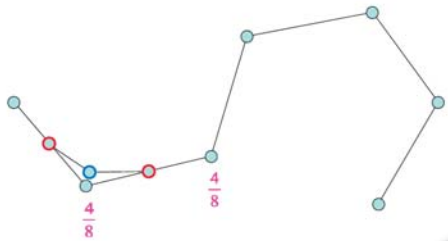




Approximating: B-spline curves

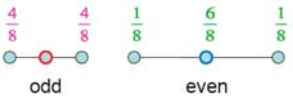
- Cubic B-spline (Catmull-Clark)

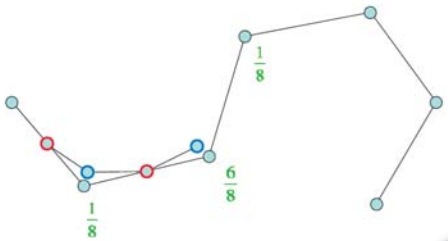




Approximating: B-spline curves

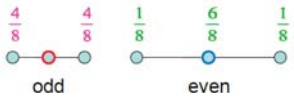
- Cubic B-spline (Catmull-Clark)

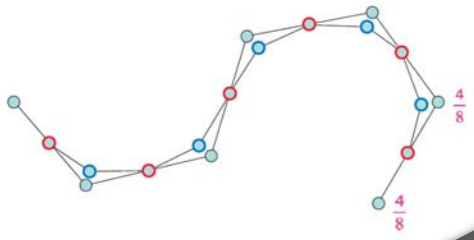




Approximating: B-spline curves

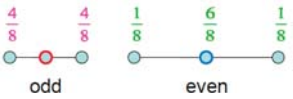
- Cubic B-spline (Catmull-Clark)





Approximating: B-spline curves

- Cubic B-spline (Catmull-Clark)



Approximating: B-spline curves

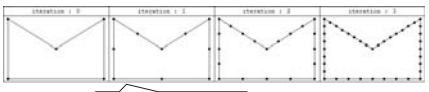
- Mask of size n yields $\mathcal{C}^{n-1}$ curve
  - Linear B-spline
    - Even coefficient (1)
  - Quadratic B-spline
    - Even coefficients $\left(\frac{3}{4}, \frac{1}{4}\right)$
  - Cubic B-spline
    - Even coefficients $\left(\frac{1}{8}, \frac{6}{8}, \frac{1}{8}\right)$

Approximating: B-spline curves

Distinguish between odd and even points

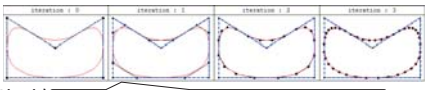
- odd (insert new points)
- even (update old points)

- Linear B-spline
  - Odd coefficients $\left(\frac{1}{2}, \frac{1}{2}\right)$
  - Even coefficient (1)



Black: Polygon subdivision

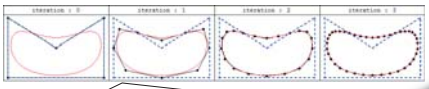
- Quadratic B-spline (Chaikin)
  - Odd coefficients $\left(\frac{1}{4}, \frac{3}{4}\right)$
  - Even coefficients $\left(\frac{3}{4}, \frac{1}{4}\right)$



- Cubic B-spline (Catmull-Clark)

Red: analytic def. of B-spline, Black: Polygon subdivision

  - Odd coefficients $\left(\frac{4}{8}, \frac{4}{8}\right)$
  - Even coefficients $\left(\frac{1}{8}, \frac{6}{8}, \frac{1}{8}\right)$



Red: analytic def. of B-spline, Black: Polygon subdivision

Approximating: B-spline curves

B-splines

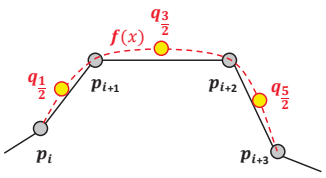
- smooth
- must be tessellated
  - sampling issues
  - triangle size issue
  - cracking concern
- have uniform resolution
  - detail must be global
- require regular grid
- complex topology hard
  - no corners, holes
  - ~~trimming hard~~
  - stitching hard
  - creases and sharp edges hard

Subdivision

- limit surfaces are smooth
- gives meshes
  - subdivide as needed
  - always connected
  - get as many poly as you need
- put details where needed
  - detail is multi-resolution
- works with arbitrary mesh
- any topology can be handled
  - easy to make corners, holes
  - ~~trimming easy~~
  - stitching easy
  - creases and sharp edges easy

Interpolating (4-point Scheme)

- Keep 4 old vertices
- Generate new vertices by fitting cubic curve to old vertices
- $\mathcal{C}^1$continuous limit curve



Interpolation of 4 points:

$$(0, \boldsymbol{p}_i), (1, \boldsymbol{p}_{i+1}), (2, \boldsymbol{p}_{i+2}), (3, \boldsymbol{p}_{i+3})$$

is

$$f(x) = a\,x^3 + bx^2 + cx + d$$

where

$$\begin{cases} f(0) = \boldsymbol{p}_i \\ \cdots = \cdots \\ f(3) = \boldsymbol{p}_{i+3} \end{cases}$$

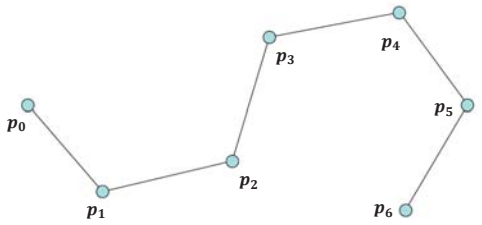
so new vertices could be:

$$q_{\frac{1}{2}} = f\left(\frac{1}{2}\right) = \frac{1}{16}(5\boldsymbol{p}_i + 15\boldsymbol{p}_{i+1} - 5\boldsymbol{p}_{i+2} + \boldsymbol{p}_{i+3})$$

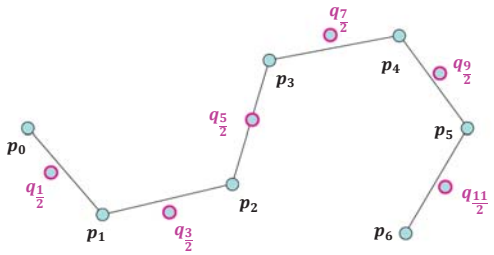
$$q_{\frac{3}{2}} = f\left(\frac{3}{2}\right) = \frac{1}{16}(-\boldsymbol{p}_i + 9\boldsymbol{p}_{i+1} + 9\boldsymbol{p}_{i+2} - \boldsymbol{p}_{i+3})$$

$$q_{\frac{5}{2}} = f\left(\frac{5}{2}\right) = \frac{1}{16}(\boldsymbol{p}_i - 5\boldsymbol{p}_{i+1} + 15\boldsymbol{p}_{i+2} + 5\boldsymbol{p}_{i+3})$$

- Example

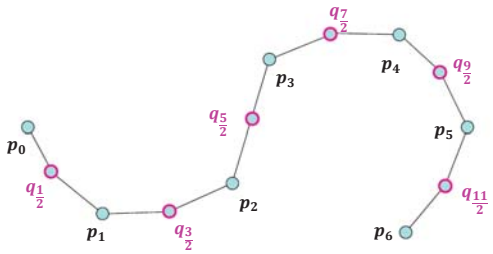


- Example



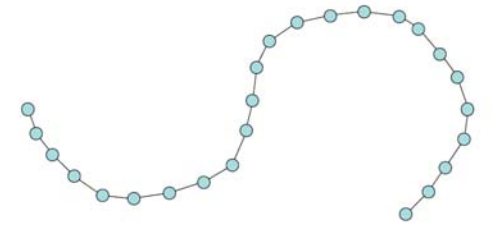
$$\begin{cases} q_{\frac{1}{2}} = \frac{1}{16}(5\boldsymbol{p}_0 + 15\boldsymbol{p}_1 - 5\boldsymbol{p}_2 + \boldsymbol{p}_3) \\ q_{i+\frac{1}{2}} = \frac{1}{16}(-\boldsymbol{p}_i + 9\boldsymbol{p}_{i+1} + 9\boldsymbol{p}_{i+2} - \boldsymbol{p}_{i+3}) \quad \forall i = 1,2,3,4 \\ q_{\frac{11}{2}} = \frac{1}{16}(\boldsymbol{p}_3 - 5\boldsymbol{p}_4 + 15\boldsymbol{p}_5 + 5\boldsymbol{p}_6) \end{cases}$$

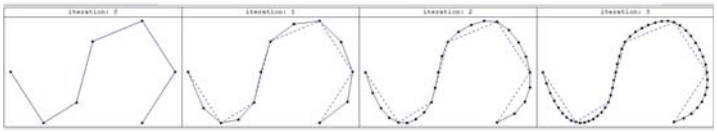
- Example



$$\begin{cases} q_{\frac{1}{2}} = \frac{1}{16}(5\boldsymbol{p}_0 + 15\boldsymbol{p}_1 - 5\boldsymbol{p}_2 + \boldsymbol{p}_3) \\ q_{i+\frac{1}{2}} = \frac{1}{16}(-\boldsymbol{p}_i + 9\boldsymbol{p}_{i+1} + 9\boldsymbol{p}_{i+2} - \boldsymbol{p}_{i+3}) \quad \forall i = 1,2,3,4 \\ q_{\frac{11}{2}} = \frac{1}{16}(\boldsymbol{p}_3 - 5\boldsymbol{p}_4 + 15\boldsymbol{p}_5 + 5\boldsymbol{p}_6) \end{cases}$$

- Example





- Classification of subdivision schemes

Primal	Faces are split into sub-faces
Dual	Vertices are split into multiple vertices

Approximating	Control points are not interpolated
Interpolating	Control points are interpolated

- Classification of subdivision schemes

	Primal (face split)	
	Triangular meshes	*Quad Meshes*
Approximating	Loop (C^2)	Catmull-Clark (C^2)
Interpolating	Butterfly (C^1)	Kobbelt (C^1)

- Many other methods …

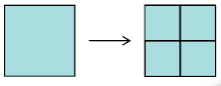
Dual (vertex split)
Doo-Sabin, Midedge(C^1)
Biquartic (C^2)

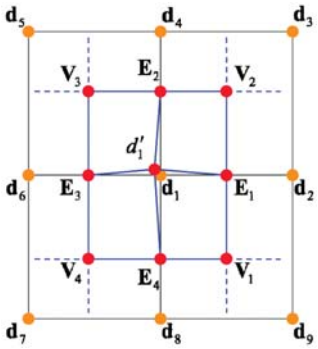
- Classification of subdivision schemes

	Primal (face split)		Dual (vertex split)
	Triangles	*Quadrangles*	
Approximating	Loop	**Catmull-Clark**	Doo-Sabin Midedge
Interpolating	Butterfly	Kobbelt	

- Generalization of bi-**cubic** B-Splines
- **Primal**, **approximation** subdivision scheme
- Applied to **quadrangle meshes**
- Generates G2 continuous limit surfaces:
  - C1 for the set of finite extraordinary points
    - Vertices with valence ≠ 4
  - C2 continuous everywhere else

Smoothness of curves and surfaces [*from wikipedia*]
A curve or surface can be described as having G^n continuity, n being the increasing measure of smoothness. Consider the segments either side of a point on a curve:
G^0: The curves touch at the join point.
G^1: The curves also share a common tangent direction at the join point.
G^2: The curves also share a common center of curvature at the join point.



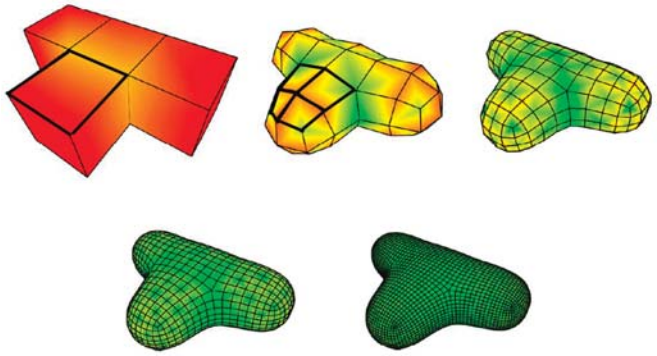


$$\mathbf{v}_2 = \frac{1}{n} \times \sum_{j=1}^{n} \mathbf{d}_j$$

$$\mathbf{E}_i = \frac{1}{4}(\mathbf{d}_1 + \mathbf{d}_{2i} + \mathbf{v}_i + \mathbf{v}_{i+1})$$

$$\mathbf{d}'_1 = \frac{m-3}{m}\mathbf{d}_1 + \frac{2}{m}\mathbf{R} + \frac{1}{m}\mathbf{S}$$

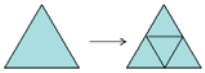
where $\mathbf{R} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{E}_i$, $\mathbf{S} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{v}_i$



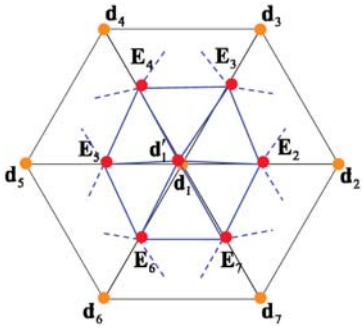
- Classification of subdivision schemes

	Primal (face split)		Dual (vertex split)
	Triangles	*Quadrangles*	
Approximating	**Loop**	Catmull-Clark	Doo-Sabin Midedge
Interpolating	Butterfly	Kobbelt	

- Generalization of bi-**quadratic** splines
- **Primal**, **approximating** subdivision scheme
- Applied to **triangle meshes**
- Generates G2 continuous limit surfaces:
  - C1 for the set of finite extraordinary points
    - Vertices with valence ≠ 6
  - C2 continuous everywhere else



C. T. Loop (1987), **Smooth Subdivision Surfaces Based on Triangles**, *M.S. Thesis, Department of Mathematics*, University of Utah.

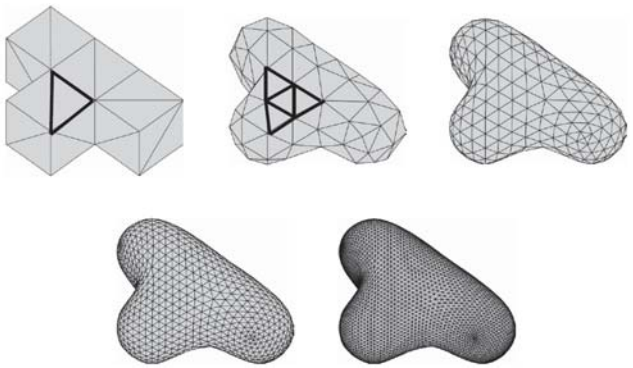


$$\mathbf{E}_i = \frac{3}{8} \times (\mathbf{d}_1 + \mathbf{d}_i) + \frac{1}{8}(\mathbf{d}_{i-1} + \mathbf{d}_{i+1})$$

$$\mathbf{d}'_1 = \alpha_n \mathbf{d}_1 + \frac{(1-\alpha_n)}{n}\sum_{j=2}^{n+1} \mathbf{d}_j$$

where $\alpha_n = \frac{3}{8} + \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{n}\right)^2$

C. T. Loop (1987), **Smooth Subdivision Surfaces Based on Triangles**, *M.S. Thesis, Department of Mathematics*, University of Utah.

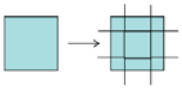


C. T. Loop (1987), **Smooth Subdivision Surfaces Based on Triangles**, *M.S. Thesis, Department of Mathematics*, University of Utah.

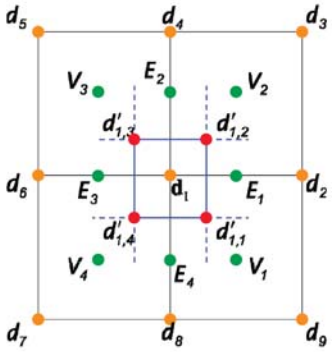
- Classification of subdivision schemes

	Primal (face split)		Dual (vertex split)
	Triangles	*Quadrangles*	
Approximating	Loop	Catmull-Clark	**Doo-Sabin** Midedge
Interpolating	Butterfly	Kobbelt	

- Generalization of bi-**quadratic** B-Splines
- **Dual**, **approximating** subdivision scheme
- Applied to **quadrangle meshes**
- Generates G1 continuous limit surfaces:
  - C0 for the set of finite extraordinary points
    - Center of irregular polygons (n ≠ 4) after 1 subdivision step
  - C1 continuous everywhere else



Doo-Sabin Subdivision



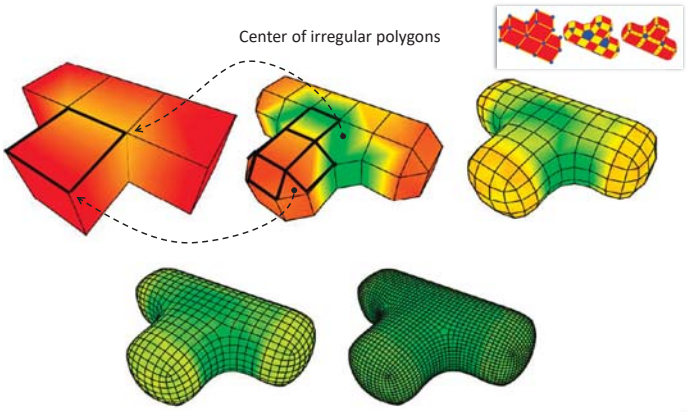
$$\mathbf{v}_2 = \frac{1}{n} \times \sum_{j=1}^{n} \mathbf{d}_j$$

$$\mathbf{E}_i = \frac{1}{2}(\mathbf{d}_1 + \mathbf{d}_{2i})$$

$$\mathbf{d}'_{1,j} = \frac{1}{4}(\mathbf{d}_1 + \mathbf{E}_j + \mathbf{E}_{j-1} + \mathbf{v}_j)$$

Doo-Sabin Subdivision

Center of irregular polygons



Subdivision surfaces

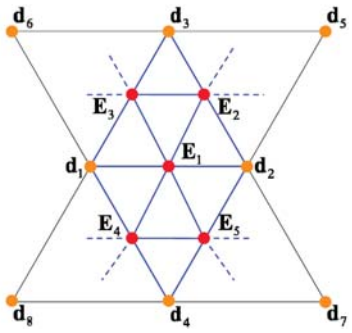
- Classification of subdivision schemes

	Primal (face split)		Dual (vertex split)
	Triangles	*Quadrangles*	
Approximating	Loop	Catmull-Clark	Doo-Sabin Midedge
Interpolating	**Butterfly**	Kobbelt	

Butterfly Subdivision

- **Primal**, **interpolating** scheme
- Applied to **triangle meshes**
- Generates G1 continuous limit surfaces:
  - C0 for the set of finite extraordinary points
    - Vertices of valence = 3 or > 7
  - C1 continuous everywhere else

Butterfly Subdivision

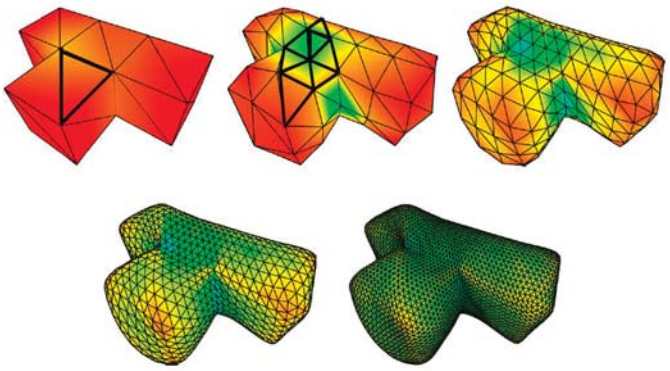


$$\mathbf{E}_1 = \frac{1}{2}(\mathbf{d}_1 + \mathbf{d}_2) + \omega(\mathbf{d}_3 + \mathbf{d}_4) - \frac{\omega}{2}(\mathbf{d}_5 + \mathbf{d}_6 + \mathbf{d}_7 + \mathbf{d}_8)$$

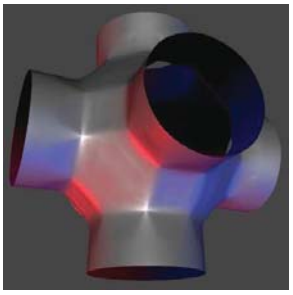
e.g. $\omega = \frac{1}{8}$

$$\mathbf{d}'_i = \mathbf{d}_i$$

Butterfly Subdivision



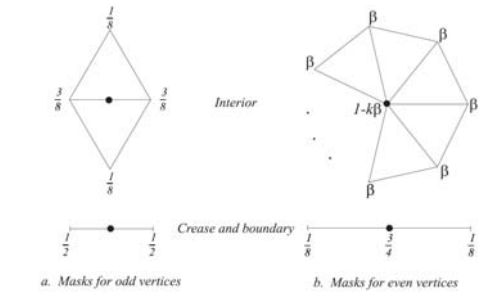
Butterfly Subdivision



Surface is smooth except for the extraordinary points

Remark

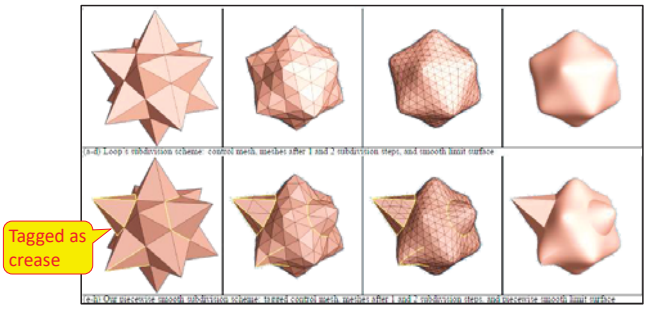
- Different masks applied on the boundary and crease
- Example: ***Loop***



a. Masks for odd vertices *b. Masks for even vertices*

Remark

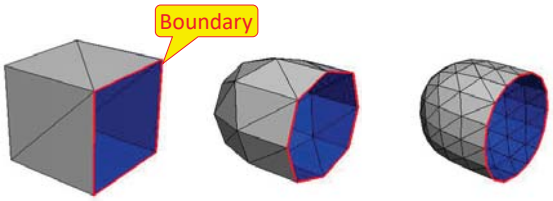
- Different masks apply on the boundary and crease
- Example: ***Loop***



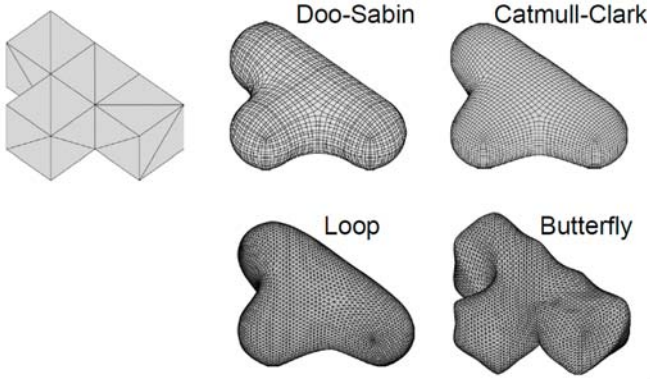
H. Hoppe et *al.* , **Piecewise smooth surface reconstruction**, *Proc. SIGGRAPH'94* (1994), pp. 295–302

Remark

- Different masks apply on the boundary and crease
- Example: ***Loop***

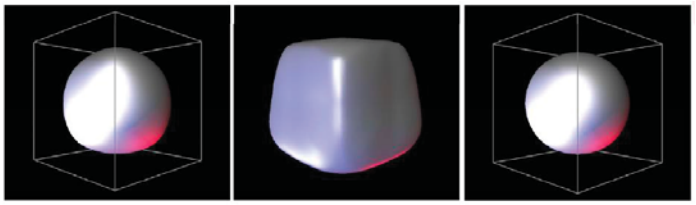


Comparison



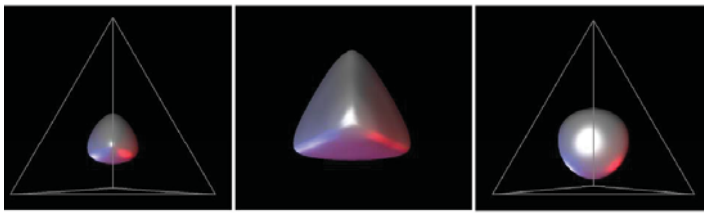
Comparison

- Subdividing a cube
  - Loop result is assymetric, because cube was triangulated first
  - Both Loop and Catmull-Clark are better then Butterfly (C^2 vs. C^1)
  - Interpolation vs. smoothness



Comparison

- Subdividing a tetrahedron
  - Same insights
  - Severe shrinking for approximating schemes



Comparison

- Loop and Catmull-Clark best when interpolation is not required
- Loop best for triangular meshes
- Catmull-Clark best for quad meshes
  - Don't triangulate and then use Loop

Introduction to meshes processing

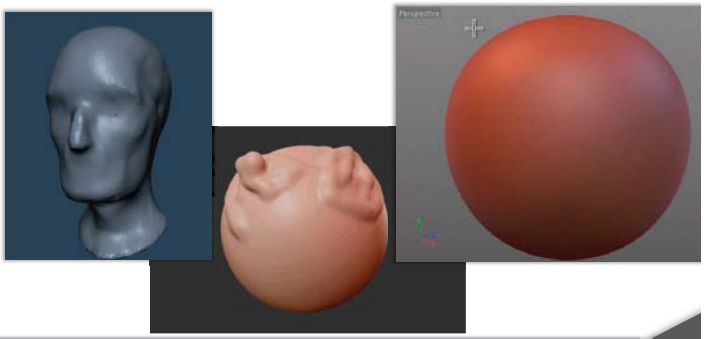
Mesh deformation

Ruding LOU
ruding.lou@ensam.eu

Motivation

- Easy modeling

  generate new shapes by deforming existing ones



Motivation

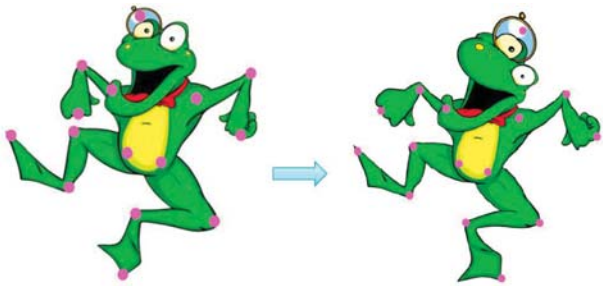
- Character posing for animation

Key frame model preparation

Animation

Challenges

- User says as little as possible …

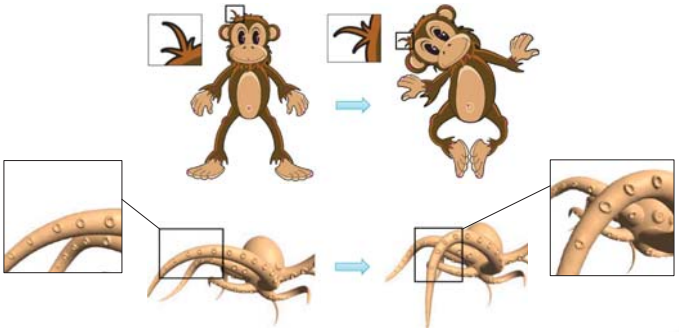


…algorithm deduces the rest

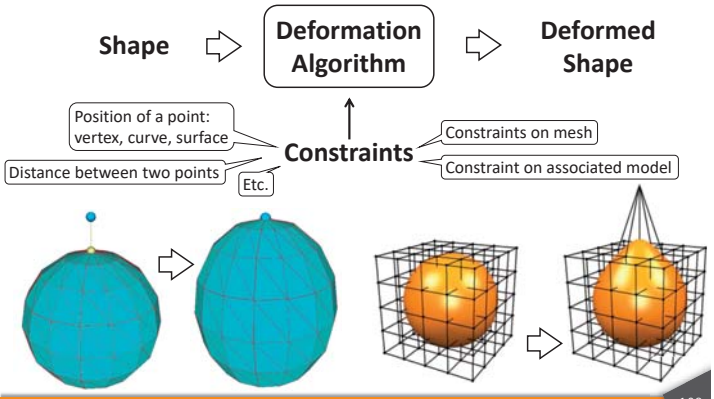
Challenges

- "Intuitive deformation"

  global change + local detail preservation

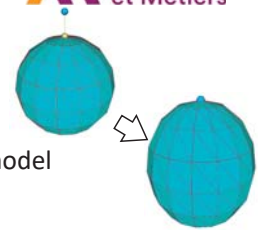


Rules of the Game

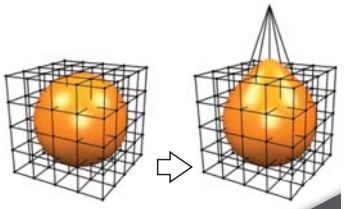


Rules of the Game

- Surface-based deformation
  - **Differential coordinates**

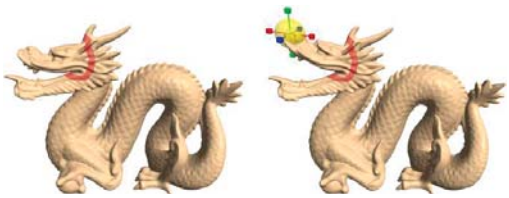


- Deformation based on mechanical model
  - Force density method
- Space deformation
  - Using cage



Laplacian surface deformation

- Control mechanism
  - Handles (vertices) moved by user
  - Region of influence (ROI)

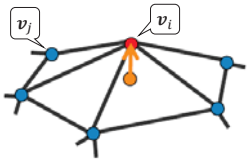


O. Sorkine et *al*, (2004), **Laplacian surface editing**, *SGP '04*, pp. 175–184.

Laplacian surface deformation

- Laplacian Coordinates

$$\boldsymbol{\delta}_i = \boldsymbol{L}(\boldsymbol{v}_i) = \boldsymbol{v}_i - \sum_{j \in N(i)} \frac{1}{\deg_i} \boldsymbol{v}_j = \sum_{j \in N(i)} \frac{1}{\deg_i} (\boldsymbol{v}_i - \boldsymbol{v}_j)$$



O. Sorkine et *al*, (2004), **Laplacian surface editing**, *SGP '04*, pp. 175–184.

Laplacian surface deformation

- Deformation
- Pose modeling constraints for vertices $C \subset V$

$$\boldsymbol{v}'_i = \boldsymbol{u}_i \quad \forall i \in C$$

- Solve $\boldsymbol{v}'_i \in \boldsymbol{P}'$ in the system

Laplacian coordinates of original mesh

Laplacian coordinates of deformed mesh : $\boldsymbol{\delta}'_i = \boldsymbol{L}(\boldsymbol{v}'_i)$

$$\begin{cases} \forall i \in V & \boldsymbol{\delta}_i - \boldsymbol{\delta}'_i = \boldsymbol{0} \\ \forall i \in C & \boldsymbol{u}_i - \boldsymbol{v}'_i = \boldsymbol{0} \end{cases}$$

Shape constraints

User constraints

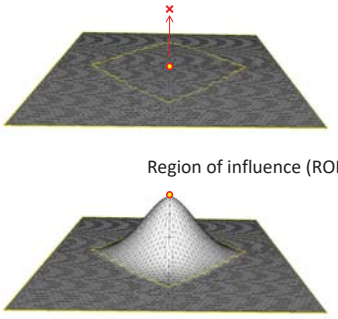
- No exact solution, minimize squared error

$$\boldsymbol{P}' = \arg\min_{\boldsymbol{P}'} \sum_{i \in V} \|\boldsymbol{\delta}_i - \boldsymbol{\delta}'_i\|^2 + \sum_{i \in C} \|\boldsymbol{u}_i - \boldsymbol{v}'_i\|^2$$

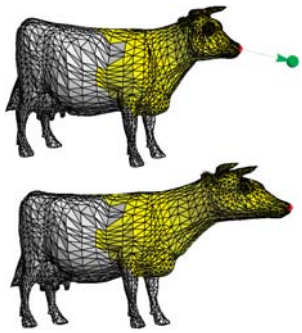
O. Sorkine et *al*, (2004), **Laplacian surface editing**, *SGP '04*, pp. 175–184.

Laplacian surface deformation

- Examples

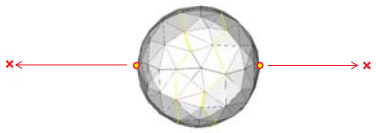


Region of influence (ROI)

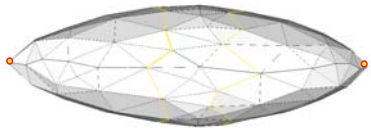


Laplacian surface deformation

- Examples

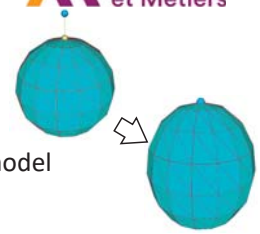


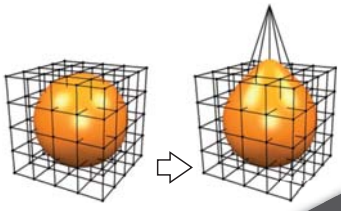
Unchanged zone



Rules of the Game

- Surface-based deformation
  - Differential coordinates
- Deformation based on mechanical model
  - **Force density method**
- Space deformation
  - Using cage





Force density method

- Coupling a simple mechanical model to mesh
  - Spring network (nodes and springs):
    - Each node has null mass
    - Each edge (i) connecting two nodes can be seen as a spring with a null initial length and a stiffness q_i (force density).
  - External forces
    - To preserve the static equilibrium state of the structure, external forces f_n have to be applied to the node n.

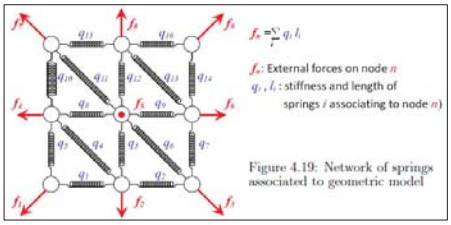


Figure 4.19: Network of springs associated to geometric model

Schek, H.J., (1974), **The force density method for form finding and computation of general networks**, *App. Mech. and Eng.*, 3(2), 115–134.

Force density method

- Coupling a simple mechanical model to mesh
  - Deformation
    - through the manipulation of a restricted set of external forces.
    - minimization of a quantity (e.g. relative to the external forces)

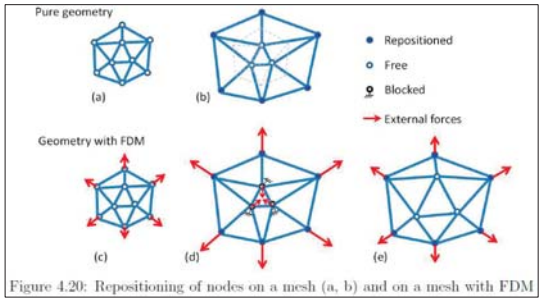


Figure 4.20: Repositioning of nodes on a mesh (a, b) and on a mesh with FDM

Schek, H.J., (1974), **The force density method for form finding and computation of general networks**, *App. Mech. and Eng.*, 3(2), 115–134.

Force density method

- Coupling a simple mechanical model to mesh
  - Deformation
    - through the manipulation of a restricted set of external forces.
    - minimization of a quantity (e.g. relative to the external forces)

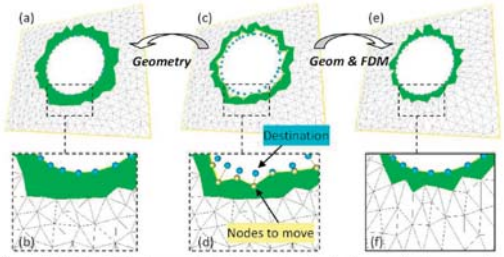
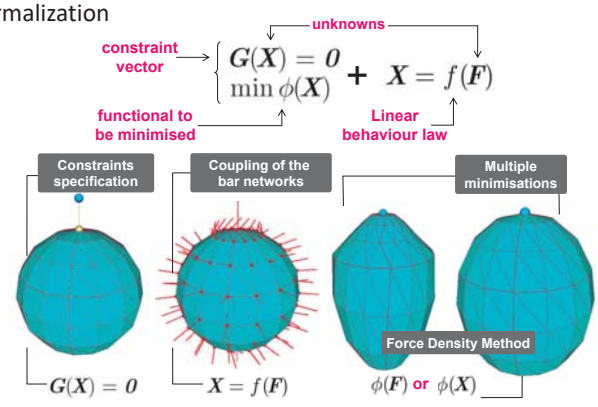


Figure 4.22: Nodal displacement definitions (c, d), displacement of pure mesh (a, b) and displacement of mesh coupled with a FDM mechanical model (e, f)

Schek, H.J., (1974), **The force density method for form finding and computation of general networks**, *App. Mech. and Eng.*, 3(2), 115–134.

Force density method

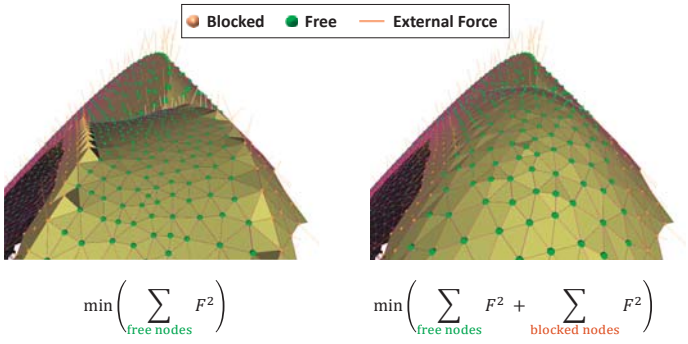
- Formalization



J-P Pernot *et al* (2006), **Filling holes in meshes using a mechanical model to simulate the curvature variation minimization**, *Computer Graphics*. (30), 892-902.

Force density method

- Minimization of sum of external forces



J-P Pernot *et al* (2006), **Filling holes in meshes using a mechanical model to simulate the curvature variation minimization**, *Computer Graphics*. (30), 892-902.

Force density method

- Minimization of sum of external forces

Model repairing

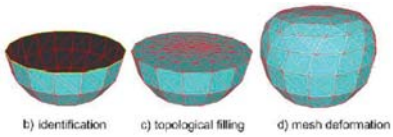


Figure: after the identification a hole contour, the missing area is filled with a topological grid whose final shape results from a deformation process.

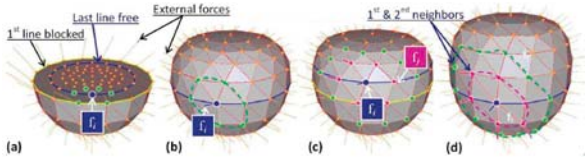


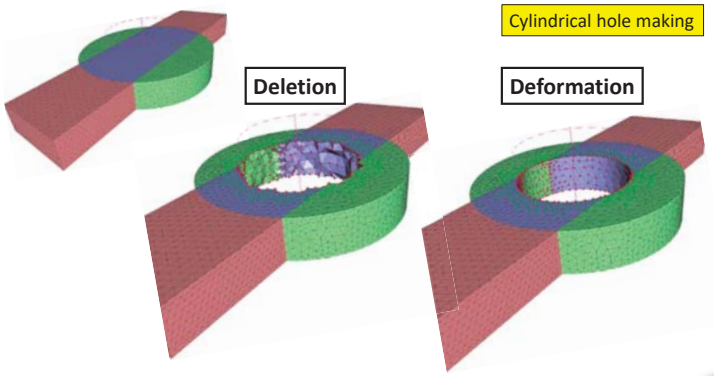
Figure: minimization of external forces only applied on the free nodes (a), minimization of external forces applied on the free and blocked nodes (b), minimization of external force variation on free nodes (c) and minimization of external forces variation on free and blocked nodes (d)

J-P Pernot *et al* (2006), **Filling holes in meshes using a mechanical model to simulate the curvature variation minimization**, *Computer Graphics*. (30), 892-902.

Force density method

- Minimization of sum of external forces under constraints

Cylindrical hole making

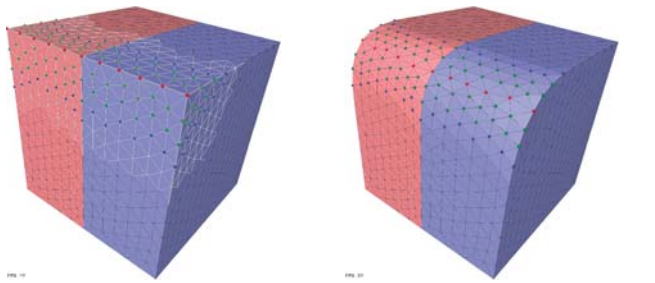


R. Lou et *al*, (2010), **Direct modification of semantically-enriched Finite Element Meshes**, *IJSM* (16), 1-2, pp. 81-108

Force density method

- Minimization of sum of external forces under constraints

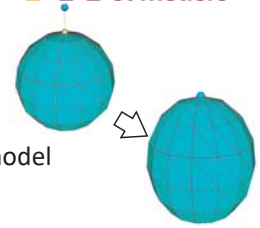
Sharp edge filleting

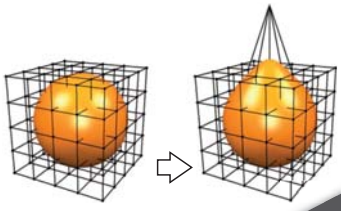


R. Lou et *al*, (2015), **Filleting sharp edges of multi-partitioned volume finite element meshes**, *Engineering Computations*, 32(1), pp.129-154

Rules of the Game

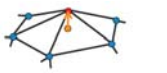
- Surface-based deformation
  - Differential coordinates
- Deformation based on mechanical model
  - Force density method
- Space deformation
  - **Using cage**

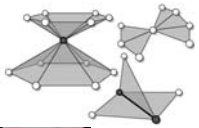




Space-based deformation

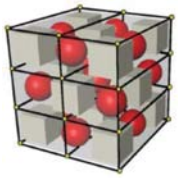
- Drawbacks of surface-Based Deformation
  - Non-linear optimization
  - In case of topological inconsistencies
  - In the presence of geometric degeneracies
  - When deforming several models

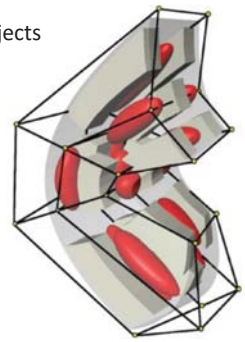




Space-based deformation

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





T.W. Sederberg, S.R. Parry, (1986), **Free-form deformation of solid geometric models**, *SIGGRAPH Computer Graphics* 20 (4): 151–160.

Cage based deformation

- Impose a grid of control points $\boldsymbol{P}_{i,j,k}$ on the parallelepiped.
- Define a local coordinate system on a parallelepiped region.
  - Parallelepiped Space (S,T,U)
- Compute the local coordinates of the model points in the parallelepiped region.
  - Conversion $(x,y,z) \rightarrow (s,t,u)$

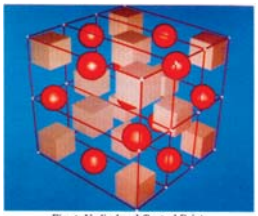


Fig. 4 Undisplaced Control Points

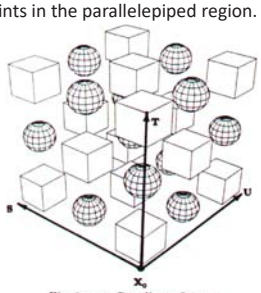
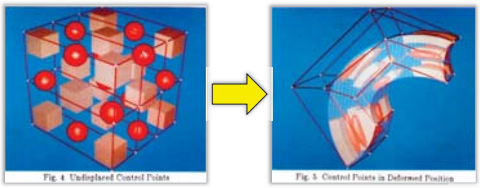


Fig. 3 x,t,u Coordinate System

T.W. Sederberg, S.R. Parry, (1986), **Free-form deformation of solid geometric models**, *SIGGRAPH Computer Graphics* 20 (4): 151–160.

Cage based deformation

- Impose a grid of control points $\boldsymbol{P}_{i,j,k}$ on the parallelepiped.
- Define a local coordinate system on a parallelepiped region.
  - Parallelepiped Space (S,T,U)
- Compute the local coordinates of the model points in the parallelepiped region.
  - Conversion $(x,y,z) \rightarrow (s,t,u)$
- Move the control points $\boldsymbol{P}_{i,j,k}$ of the parallelepiped
- Evaluate new position of model point (x',y',z') according to (s,t,u).



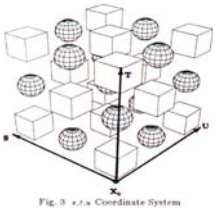
T.W. Sederberg, S.R. Parry, (1986), **Free-form deformation of solid geometric models**, *SIGGRAPH Computer Graphics* 20 (4): 151–160.

Cage based deformation

- Compute the bounding box: e.g. axis-aligned

$$\boldsymbol{X}_0 = (x_{min}, y_{min}, z_{min})$$

$$\begin{cases} S = (1,0,0) \\ T = (0,1,0) \\ U = (0,0,1) \end{cases}$$



- Control points: e.g. equidistant

$$\boldsymbol{P}_{i,j,k} = \boldsymbol{X}_0 + \frac{i}{i_{max}} S + \frac{j}{j_{max}} T + \frac{k}{k_{max}} U$$

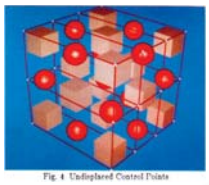


Fig. 4 Undeplaned Control Points

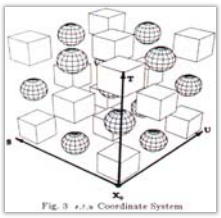
T.W. Sederberg, S.R. Parry, (1986), **Free-form deformation of solid geometric models**, *SIGGRAPH Computer Graphics* 20 (4): 151–160.

Cage based deformation

- Local parameterization:
  - A point $\boldsymbol{X}$ in the STU coordinate system:

$$\boldsymbol{X} = \boldsymbol{X}_0 + sS + tT + uU$$

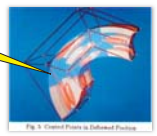
$$\begin{cases} s = \dfrac{T \times U \cdot (\boldsymbol{X} - \boldsymbol{X}_0)}{T \times U \cdot S} & 0 < s < 1 \\ t = \dfrac{S \times U \cdot (\boldsymbol{X} - \boldsymbol{X}_0)}{S \times U \cdot T} & 0 < t < 1 \\ u = \dfrac{S \times T \cdot (\boldsymbol{X} - \boldsymbol{X}_0)}{S \times T \cdot U} & 0 < u < 1 \end{cases}$$



- New position of deformed point:

$$\boldsymbol{X}' = \sum_{i=0}^{i_{max}} \sum_{j=0}^{j_{max}} \sum_{k=0}^{k_{max}} B_i^{i_{max}}(s) B_j^{j_{max}}(t) B_k^{k_{max}}(u) \boldsymbol{P}'_{i,j,k}$$

Bezier solid



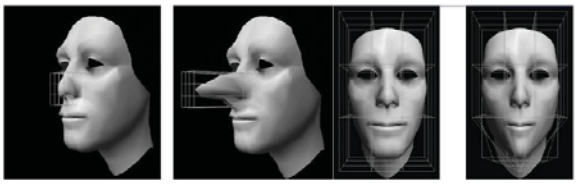
B is Bernstein polynomial: $B_v^n(x) = \binom{n}{v} x^v (1-x)^{n-v}, \quad v = 0, \ldots, n.$

T.W. Sederberg, S.R. Parry, (1986), **Free-form deformation of solid geometric models**, *SIGGRAPH Computer Graphics* 20 (4): 151–160.

Cage based deformation

- Experimentations

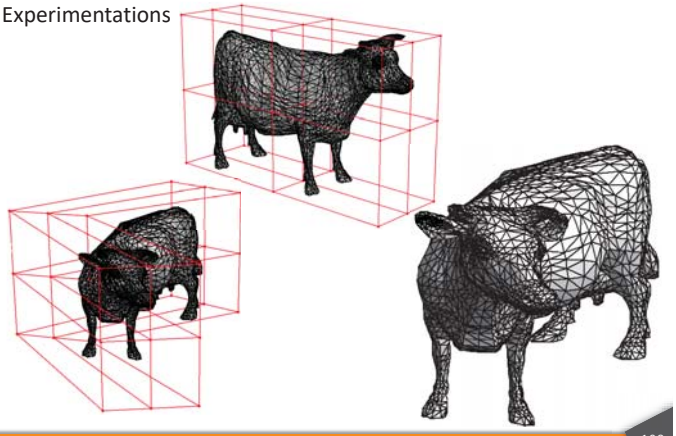
Fig. 14 Local FFD Fig. 15 Global FFD Fig. 16 Final Product



T.W. Sederberg, S.R. Parry, (1986), **Free-form deformation of solid geometric models**, *SIGGRAPH Computer Graphics* 20 (4): 151–160.

Cage based deformation

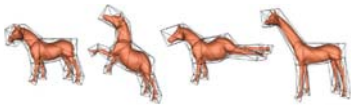
- Experimentations



Cage based deformation

- Advantages
  - Independent of surface geometry
  - Intuitive, interactive
  - Efficient, flexible

- Shortcoming
  - The parallelepipedic shape of lattice limits deformation shape, e.g. can hardly design a circular bump on the surface
  - Discontinuity at lattice boundary
  - Global influence with Bezier basis functions
  - The cage could be optimized to better fit the mesh



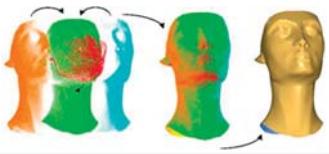
Ju T, Schaefer S, Warren J, (2005), **Mean value coordinates for closed triangular meshes**, *ACM SIGGRAPH*, 24-3, pp.561-66.

Master 2 Recherche MTI 3D

Introduction to meshes processing

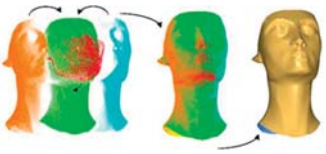
Mesh alignment

Ruding LOU
ruding.lou@ensam.eu



Mesh Alignment

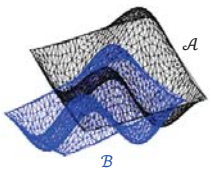
- Needs of mesh alignment
  - Reverse engineering
  - Virtual assembly



A. Sharf, M. Blumenkrants, A. Shamir, D. Cohen-Or, **SnapPaste: An interactive technique for easy mesh composition**, *The Visual Computer*, 22 (9–11) (2006), pp. 835-844

  - etc.

- Two surfaces: mesh $\mathcal{A}$ and mesh $\mathcal{B}$
- Mesh $\mathcal{B}$ should be aligned with the $\mathcal{A}$



Rotation and Tranlation

Mesh Alignment

- Two surfaces: mesh $\mathcal{A}$ and mesh $\mathcal{B}$
- Mesh $\mathcal{B}$ should be aligned with the $\mathcal{A}$
- Apply a rigid transformation (rotation and translation) on each vertex coordinates $\boldsymbol{q}_i \in \mathcal{B}$

$$\boldsymbol{q}'_i = \mathbf{R}\,\boldsymbol{q}_i + \mathbf{t}$$

where $\mathbf{R} = R_x(\alpha)\, R_y(\beta)\, R_z(\gamma)$ is the rotation matrix

$\mathbf{t} = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$ is the translation vector

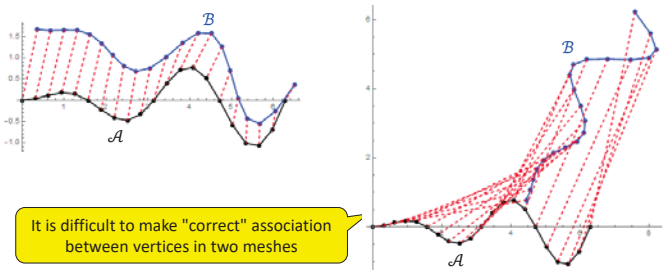
Unknowns: $\alpha, \beta, \gamma, t_x, t_y, t_z$

- Minimize the error: $r(\mathbf{R}, \mathbf{t}) = d^2(\mathcal{B}', \mathcal{A})$

How to evaluate this ?

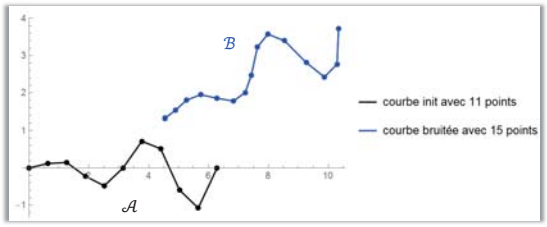
Y. Chen and G. Medioni, **Object Modelling by Registration of Multiple Range Images**, *J. Image and Vision Computing*, 10(3):145–155, April 1992

- Two surfaces: mesh $\mathcal{A}$ and mesh $\mathcal{B}$
- Associate each vertex $\boldsymbol{q}_i \in \mathcal{B}$ with a vertex $\boldsymbol{p}_{q_i} \in \mathcal{A}$
- Approximate the distance by $d^2(\mathcal{B}, \mathcal{A}) = \sum_i \left\| \boldsymbol{q}_i - \boldsymbol{p}_{q_i} \right\|^2$



It is difficult to make "correct" association between vertices in two meshes

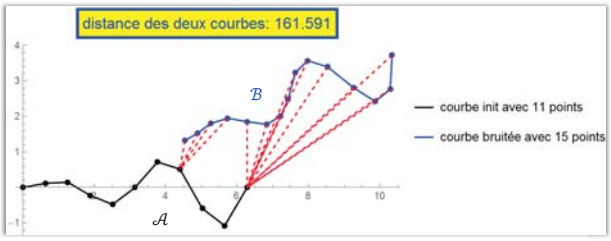
- Two surfaces: mesh $\mathcal{A}$ and mesh $\mathcal{B}$
- Mesh $\mathcal{B}$ should be aligned with the $\mathcal{A}$
- **Numerical methods: Iterative Closest Point (ICT)**



- Numerical methods: Iterative Closest Point (ICT)

Associate each i^{th} vertex $\boldsymbol{q}_i^{(0)} \in \mathcal{B}^{(0)}$ with the closest vertex $\boldsymbol{p}_{q_i}^{(0)} \in \mathcal{A}$

Evaluate the quadric error : $r^{(0)} = \sum_i \left\| \boldsymbol{q}_i^{(0)} - \boldsymbol{p}_{q_i}^{(0)} \right\|^2$



- Numerical methods: Iterative Closest Point (ICT)

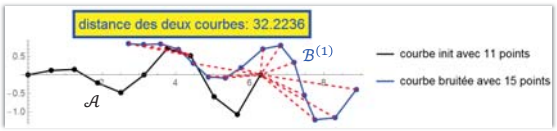
Do (at 1^{th} iteration)

Compute the unknown transformation that minimize the quadric error :

$$\mathbf{R}^{(0)}, \mathbf{t}^{(0)} = \underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}} \sum_i \left\| \left(\mathbf{R}\, \boldsymbol{q}_i^{(0)} + \mathbf{t} \right) - \boldsymbol{p}_{q_i}^{(0)} \right\|^2$$

Apply the estimated transformation to $\mathcal{B}^{(0)}$:

$$\boldsymbol{q}_i^{(1)} = \mathbf{R}^{(0)} \boldsymbol{q}_i^{(0)} + \mathbf{t}^{(0)}$$



- Numerical methods: Iterative Closest Point (ICT)

Do (at 1^{th} iteration)

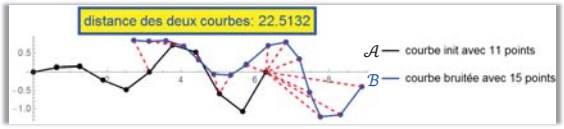
Compute the unknown transformation that minimize the quadric error

Apply the estimated transformation to $\mathcal{B}^{(0)}$

Associate each i^{th} vertex $\boldsymbol{q}_i^{(1)} \in \mathcal{B}^{(1)}$ with the closest vertex $\boldsymbol{p}_{q_i}^{(1)} \in \mathcal{A}$

Evaluate the quadric error : $r^{(1)} = \sum_i \left\| \boldsymbol{q}_i^{(1)} - \boldsymbol{p}_{q_i}^{(1)} \right\|^2$

If $\left| r^{(1)} - r^{(0)} \right| \geq \epsilon$ **Do again**



- Numerical methods: Iterative Closest Point (ICT)

Do (at k^{th} iteration)

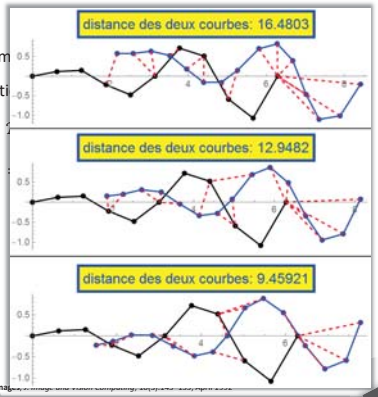
Compute the unknown transform

Apply the estimated transformati

Associate each i^{th} vertex $\boldsymbol{q}_i^{(k)} \in$

Evaluate the quadric error : $r^{(k)}$

Repeat while $\left| r^{(k)} - r^{(k-1)} \right| \geq \epsilon$



- Numerical methods: Iterative Closest Point (ICT)

Do (at k^{th} iteration)

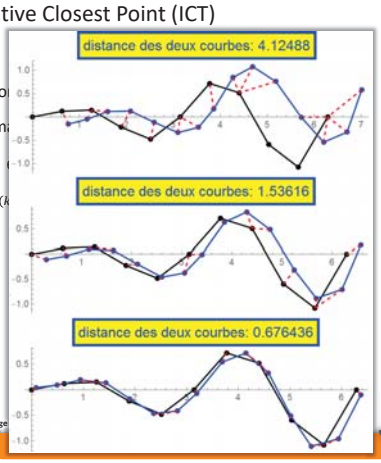
Compute the unknown transfo

Apply the estimated transform

Associate each i^{th} vertex $\boldsymbol{q}_i^{(k)}$

Evaluate the quadric error : $r^{(k)}$

Repeat while $\left| r^{(k)} - r^{(k-1)} \right| \geq \epsilon$



Algorithm: Iterative Closest Point (ICT)

Associate each i^{th} vertex $\boldsymbol{q}_i^{(0)} \in \mathcal{B}^{(0)}$ with the closest vertex $\boldsymbol{p}_{q_i}^{(0)} \in \mathcal{A}$

Evaluate the quadric error : $r^{(0)} = \sum_i \left\| \boldsymbol{q}_i^{(0)} - \boldsymbol{p}_{q_i}^{(0)} \right\|^2$

Do (at k^{th} iteration)

Compute the unknown transformation that minimize the quadric error :

$$\mathbf{R}^{(k-1)}, \mathbf{t}^{(k-1)} = \underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}} \sum_i \left\| \left(\mathbf{R}\, \boldsymbol{q}_i^{(k-1)} + \mathbf{t} \right) - \boldsymbol{p}_{q_i}^{(k-1)} \right\|^2$$

Apply the estimated transformation to $\mathcal{B}^{(k-1)}$: $\boldsymbol{q}_i^{(k)} = \mathbf{R}^{(k-1)} \boldsymbol{q}_i^{(k-1)} + \mathbf{t}^{(k-1)}$

Associate each i^{th} vertex $\boldsymbol{q}_i^{(k)} \in \mathcal{B}^{(k)}$ with the closest vertex $\boldsymbol{p}_{q_i}^{(k)} \in \mathcal{A}$

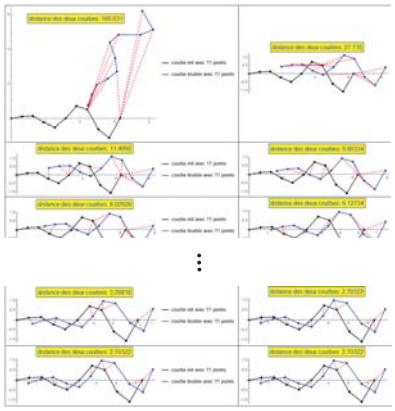
Evaluate the quadric error : $r^{(k)} = \sum_i \left\| \boldsymbol{q}_i^{(k)} - \boldsymbol{p}_{q_i}^{(k)} \right\|^2$

Repeat while $\left| r^{(k)} - r^{(k-1)} \right| \geq \epsilon$

Mesh Alignment

- **Iterative Closest Point**
- Alignment precision

  vs.
- Mesh density
- Experiment:
  - Example with **11** points
  - Local minimum

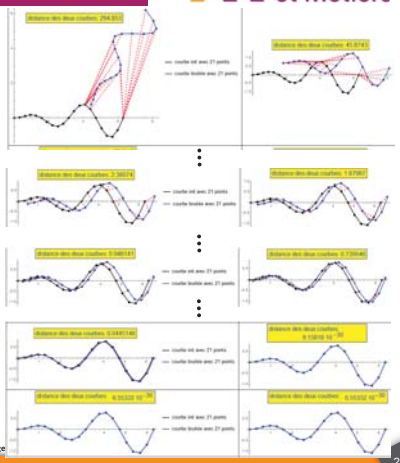


Y. Chen and G. Medioni, **Object Modelling by Registration of Multiple Range Images**, *J. Image and Vision Computing*, 10(3):145–155, April 1992

Mesh Alignment

- **Iterative Closest Point**
- Alignment precision

  vs.
- Mesh density
- Experiment:
  - Example with **22** points
  - Global minimum



Y. Chen and G. Medioni, **Object Modelling by Registration of Multiple Range**

Introduction to meshes processing

Mesh smoothing

Ruding LOU
ruding.lou@ensam.eu

Mesh Smoothing

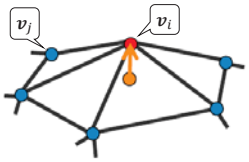
- Reduce the noise due to data acquisition
- Filter out the "high frequency" component
- Mesh smoothing: Apply the filtering technique to the geometry

Point cloud | Reconstructed mesh | Smoothed mesh

Mesh Smoothing

- Definition of Laplacian coordinate for a vertex $\boldsymbol{v}_i$

$$\boldsymbol{L}(\boldsymbol{v}_i) = \frac{1}{\|N_1(v_i)\|} \sum_{j \in N_1(v_i)} (\boldsymbol{v}_i - \boldsymbol{v}_j)$$



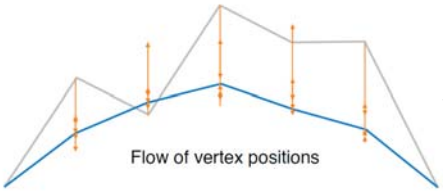
- $N_1(v_i)$: the first neighborhood of the vertex v_i
- Average difference between a vertex and its first neighborhood.

G. Taubin, (1995), **A signal processing approach to fair surface design**, *Proceedings of the 22nd SIGGRAPH*, p.351-358, NY, USA.

Mesh Smoothing

- Laplacian smoothing

$$\boldsymbol{v}_i^{(n+1)} = \boldsymbol{v}_i^{(n)} - \lambda \boldsymbol{L}\left(\boldsymbol{v}_i^{(n)}\right)$$



$\lambda = 1$

- Iterative algorithm
- Control the smoothing intensity with the parameter $\lambda \in [0,1]$

G. Taubin, (1995), **A signal processing approach to fair surface design**, *Proceedings of the 22nd SIGGRAPH*, p.351-358, NY, USA.

Mesh Smoothing

- Laplacian smoothing

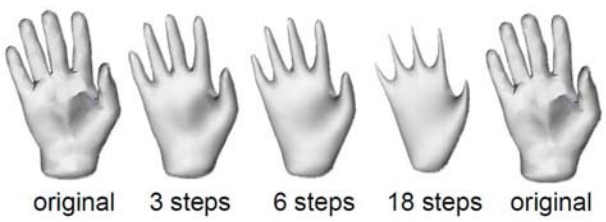
$$\boldsymbol{v}_i^{(n+1)} = \boldsymbol{v}_i^{(n)} - \lambda \boldsymbol{L}\left(\boldsymbol{v}_i^{(n)}\right)$$

- Iterative algorithm
- Control the smoothing intensity with the parameter $\lambda \in [0,1]$

G. Taubin, (1995), **A signal processing approach to fair surface design**, *Proceedings of the 22nd SIGGRAPH*, p.351-358, NY, USA.

Mesh Smoothing

- Laplacian smoothing **problem**: shrinking effect



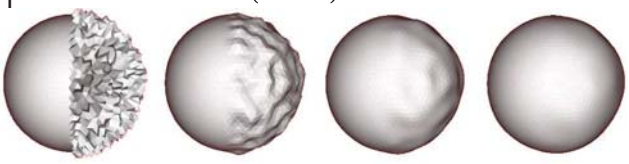
- The Laplacian smoothing iterations shrinks the mesh
- Closed surface converges to a single point

G. Taubin, (1995), **A signal processing approach to fair surface design**, *Proceedings of the 22nd SIGGRAPH*, p.351-358, NY, USA.

Mesh Smoothing

- Laplacian smoothing **problem**: shrinking effect
- **Solution** with two steps: Shrink + Inflate

$$\begin{cases} \boldsymbol{v}_i^{(n+1)'} = \boldsymbol{v}_i^{(n)} - \lambda \boldsymbol{L}\left(\boldsymbol{v}_i^{(n)}\right) & \text{Shrink} \quad (0 < \lambda < -\mu) \\ \boldsymbol{v}_i^{(n+1)} = \boldsymbol{v}_i^{(n+1)'} - \mu \boldsymbol{L}\left(\boldsymbol{v}_i^{(n+1)'}\right) & \text{Inflate} \quad (\mu < 0) \end{cases}$$

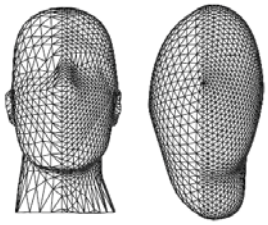


Original | 10 iterations | 50 iterations | 200 iterations

G. Taubin, (1995), **A signal processing approach to fair surface design**, *Proceedings of the 22nd SIGGRAPH*, p.351-358, NY, USA.

Mesh Smoothing

- Laplacian smoothing **problem** when the triangle sizes are not uniform: unwished distortion of the shape



(a) (b)

Figure 7: *Application of operato*

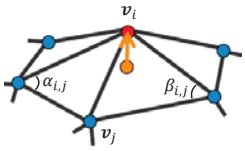
M. Desbrun, M. Meyer, P. Schroder, A. Barr, (1999), **Implicit fairing of irregular meshes using diffusion and curvature flow**, *Proceeding SIGGRAPH*, p.317-324.

Mesh Smoothing

- Laplacian smoothing **problem** when the triangle sizes are not uniform: unwished distortion of the shape
- **Solution**: assign a cotangent weight to each the vertex

$$\boldsymbol{L}(\boldsymbol{v}_i) = \frac{1}{\sum_{j \in N_1(v_i)} \omega_{ij}} \sum_{j \in N_1(v_i)} \omega_{ij}(\boldsymbol{v}_i - \boldsymbol{v}_j)$$

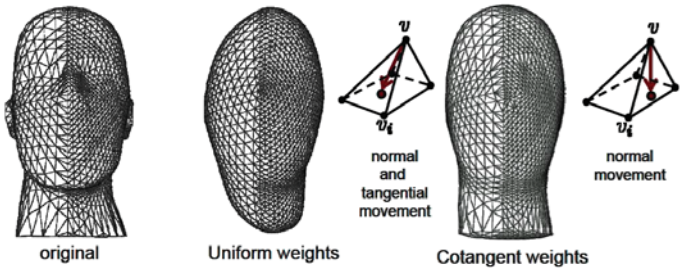
$$\left(\text{where } \omega_{ij} = \frac{\cot \alpha_{i,j} + \cot \beta_{i,j}}{2}\right)$$



M. Desbrun, M. Meyer, P. Schroder, A. Barr, (1999), **Implicit fairing of irregular meshes using diffusion and curvature flow**, *Proceeding SIGGRAPH*, p.317-324.

Mesh Smoothing

- Laplacian smoothing **problem** when the triangle sizes are not uniform: unwished distortion of the shape
- **Solution**: assign a cotangent weight to each the vertex



M. Desbrun, M. Meyer, P. Schroder, A. Barr, (1999), **Implicit fairing of irregular meshes using diffusion and curvature flow**, *Proceeding SIGGRAPH*, p.317-324.

Master 2 Recherche MTI 3D

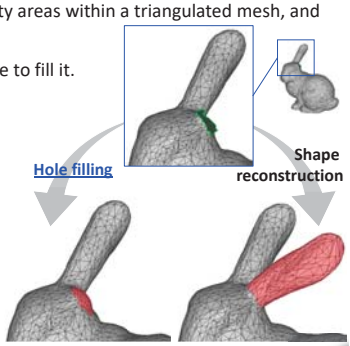
Introduction to meshes processing

Hole-filling in meshes

Ruding LOU
ruding.lou@ensam.eu

Mesh Hole Filling

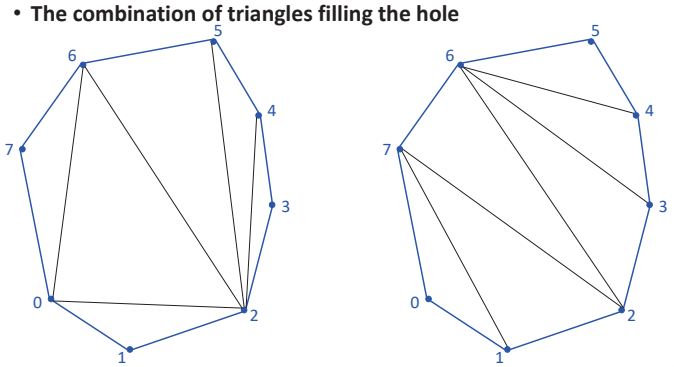
- **Why**: during digitization tasks, some parts of the object can be occluded by others and the scanner is unable to reach certain regions.
- **What**: these missing data generate empty areas within a triangulated mesh, and the boundary consists of an edge loops.
- **Solution**: create new triangles in the hole to fill it.



E. Pérez, S. Salamanca, P. Merchán, and A. Adán, (2016), **A comparison of hole-filling methods in 3D**, *Int. J. Appl. Math. Comput. Sci.*, 26 – 4, pp.885-903.

Mesh Hole Filling

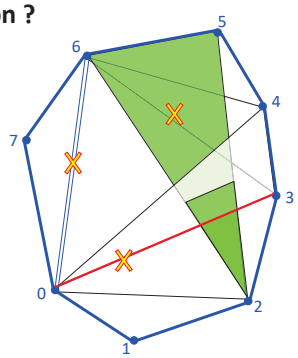
- **Direct triangulation in the hole**
- **The combination of triangles filling the hole**



P. Liepa, (2003), **Filling holes in meshes**, *Eurographics symposium on Geometry processing (SGP '03)*, pp.200-205.

Mesh Hole Filling

- **What is a correct triangulation ?**
  - **Watertight**
  - **Without self-intersection**
  - **Not overlapping**
  - **Manifold edges**

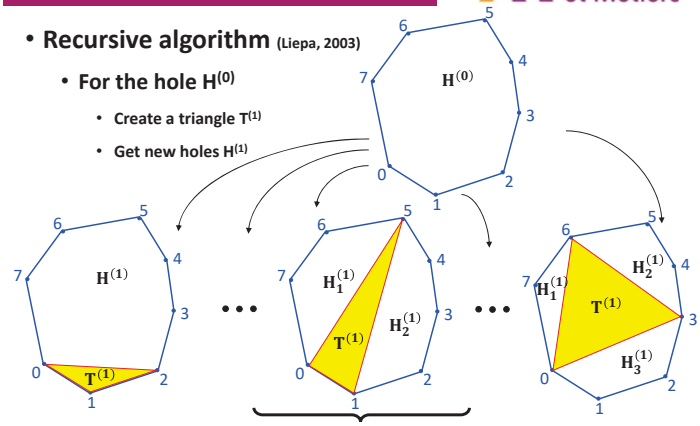


- **How to ensure a correct triangulation ?**

P. Liepa, (2003), **Filling holes in meshes**, *Eurographics symposium on Geometry processing (SGP '03)*, pp.200-205.

Mesh Hole Filling

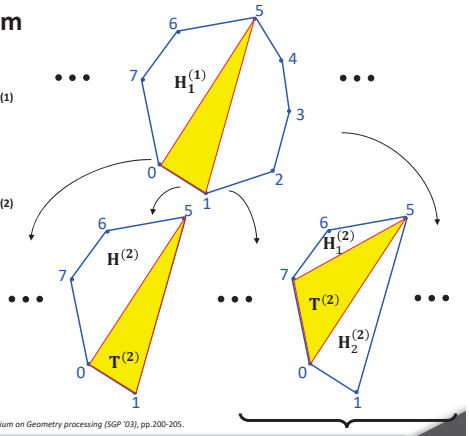
- **Recursive algorithm** (Liepa, 2003)
  - For the hole $H^{(0)}$
    - Create a triangle $T^{(1)}$
    - Get new holes $H^{(1)}$



P. Liepa, (2003), **Filling holes in meshes**, *Eurographics symposium on Geometry processing (SGP '03)*, pp.200-205.

Mesh Hole Filling

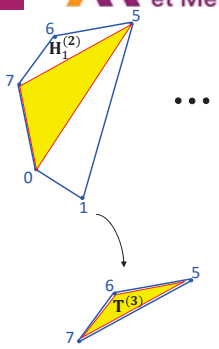
- **Recursive algorithm**
  - For the hole $H^{(0)}$
    - Create a triangle $T^{(1)}$
    - Get new holes $H^{(1)}$
  - For each hole $H^{(1)}$
    - Create a triangle $T^{(2)}$
    - Get new holes $H^{(2)}$



P. Liepa, (2003), **Filling holes in meshes**, *Eurographics symposium on Geometry processing (SGP '03)*, pp.200-205.

Mesh Hole Filling

- **Recursive algorithm**
  - For the hole $H^{(0)}$
    - Create a triangle $T^{(1)}$
    - Get new holes $H^{(1)}$
  - For each hole $H^{(1)}$
    - Create a triangle $T^{(2)}$
    - Get new holes $H^{(2)}$
  - For each hole $H^{(2)}$
    - Create a triangle $T^{(3)}$
    - Any new hole appears: ***End***.

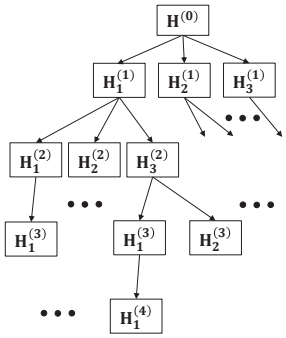


P. Liepa, (2003), **Filling holes in meshes**, *Eurographics symposium on Geometry processing (SGP '03)*, pp.200-205.

Mesh Hole Filling

- **Recursive algorithm**

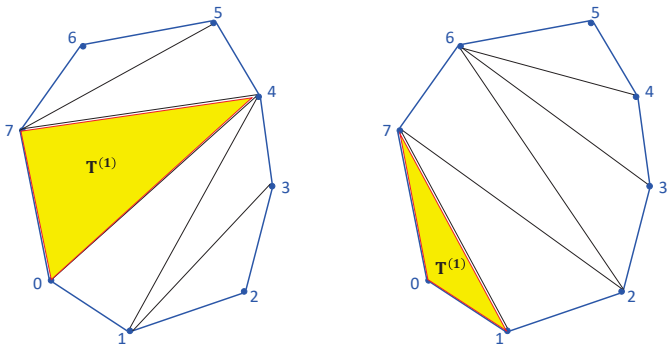
fill (Hole: *h*)
  create a triangle in the hole *h*
  compute new hole list *NH*
  for each element *e* in *NH*
    fill (e)
  end for



P. Liepa, (2003), **Filling holes in meshes**, *Eurographics symposium on Geometry processing (SGP '03)*, pp.200-205.

Mesh Hole Filling

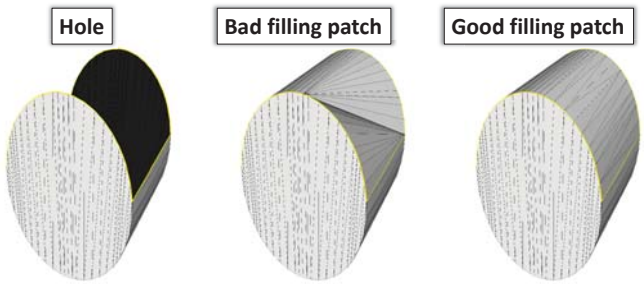
- **Combinatorial degrees of freedom**



P. Liepa, (2003), **Filling holes in meshes**, *Eurographics symposium on Geometry processing (SGP '03)*, pp.200-205.

Mesh Hole Filling

- **Combinatorial degrees of freedom**
- **Which is the best combination ?**
- **Optimization of the filling patch by minimizing the surface area**



P. Liepa, (2003), **Filling holes in meshes**, *Eurographics symposium on Geometry processing (SGP '03)*, pp.200-205.

Mesh Hole Filling

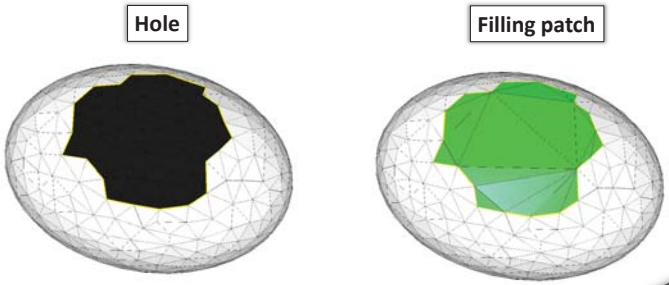
- **Greedy recursive algorithm**

fill (Hole: *h*) : Real
  for each possible triangle t_i to create
    $h_i \leftarrow h$
    create a triangle t_i in the hole h_i
    $a_i \leftarrow$ area (t_i)
    compute new hole list Nh_i in h_i
    for each element *e* in NH_i
      $a_i \leftarrow a_i +$ **fill** (e)
    end for
  end for
  $n \leftarrow \arg\min_i a_i$
  $h \leftarrow h_n$
  return a_n

P. Liepa, (2003), **Filling holes in meshes**, *Eurographics symposium on Geometry processing (SGP '03)*, pp.200-205.

Mesh Hole Filling

- **Different size between original and new triangles**
- **Bad approximation of the underlying surface**
- **=> needs of refining the patch: insert new points**

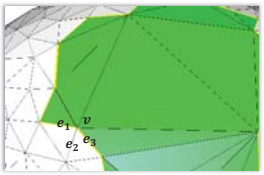


P. Liepa, (2003), **Filling holes in meshes**, *Eurographics symposium on Geometry processing (SGP '03)*, pp.200-205.

- **Filling patch refinement**

Refinement algorithm

for each vertex $\boldsymbol{v}$ on the hole boundary
  $\sigma(\boldsymbol{v}) \leftarrow$ average length of the adjacent edges in the original mesh
end for



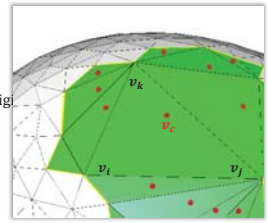
Example: $\sigma(v) \leftarrow \frac{\|e_1\|+\|e_2\|+\|e_3\|}{3}$

P. Liepa, (2003), **Filling holes in meshes**, *Eurographics symposium on Geometry processing (SGP '03)*, pp.200-205.

- **Filling patch refinement**

Refinement algorithm

for each vertex $\boldsymbol{v}$ on the hole boundary
  $\sigma(\boldsymbol{v}) \leftarrow$ average length of the adjacent edges in the orig
end for
repeat
  for each triangle $\boldsymbol{T}_{i,j,k}$ in the filling patch
    $\boldsymbol{v}_c \leftarrow$ centroid of $\boldsymbol{T}_{i,j,k}$
    $\sigma(\boldsymbol{v}_c) \leftarrow \frac{\sigma(v_i)+\sigma(v_j)+\sigma(v_k)}{3}$



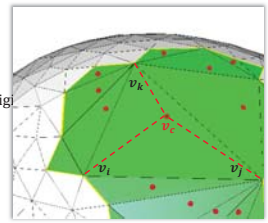
Author's suggestion: $\alpha \leftarrow \sqrt{2}$

P. Liepa, (2003), **Filling holes in meshes**, *Eurographics symposium on Geometry processing (SGP '03)*, pp.200-205.

- **Filling patch refinement**

Refinement algorithm

for each vertex $\boldsymbol{v}$ on the hole boundary
  $\sigma(\boldsymbol{v}) \leftarrow$ average length of the adjacent edges in the orig
end for
repeat
  for each triangle $\boldsymbol{T}_{i,j,k}$ in the filling patch
    $\boldsymbol{v}_c \leftarrow$ centroid of $\boldsymbol{T}_{i,j,k}$
    $\sigma(\boldsymbol{v}_c) \leftarrow \frac{\sigma(v_i)+\sigma(v_j)+\sigma(v_k)}{3}$
    for each $m \in \{i,\ j,\ k\}$
      if $\alpha\|\boldsymbol{v}_c - \boldsymbol{v}_m\| > \sigma(\boldsymbol{v}_c)$ and $\alpha\|\boldsymbol{v}_c - \boldsymbol{v}_m\| > \sigma(\boldsymbol{v}_m)$ then
        replace $\boldsymbol{T}_{i,j,k}$ with new triangles $\boldsymbol{T}_{c,j,k}$, $\boldsymbol{T}_{i,c,k}$ and $\boldsymbol{T}_{i,j,c}$
      end if
    end for
  end for
end for



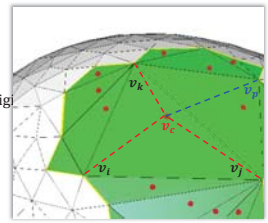
Author's suggestion: $\alpha \leftarrow \sqrt{2}$

P. Liepa, (2003), **Filling holes in meshes**, *Eurographics symposium on Geometry processing (SGP '03)*, pp.200-205.

- **Filling patch refinement**

Refinement algorithm

for each vertex $\boldsymbol{v}$ on the hole boundary
  $\sigma(\boldsymbol{v}) \leftarrow$ average length of the adjacent edges in the orig
end for
repeat
  for each triangle $\boldsymbol{T}_{i,j,k}$ in the filling patch
    $\boldsymbol{v}_c \leftarrow$ centroid of $\boldsymbol{T}_{i,j,k}$
    $\sigma(\boldsymbol{v}_c) \leftarrow \frac{\sigma(v_i)+\sigma(v_j)+\sigma(v_k)}{3}$
    for each $m \in \{i,\ j,\ k\}$
      if $\alpha\|\boldsymbol{v}_c - \boldsymbol{v}_m\| > \sigma(\boldsymbol{v}_c)$ and $\alpha\|\boldsymbol{v}_c - \boldsymbol{v}_m\| > \sigma(\boldsymbol{v}_m)$ then
        replace $\boldsymbol{T}_{i,j,k}$ with new triangles $\boldsymbol{T}_{c,j,k}$, $\boldsymbol{T}_{i,c,k}$ and $\boldsymbol{T}_{i,j,c}$
        relax the edges $\boldsymbol{e}_{i,j}$, $\boldsymbol{e}_{i,k}$ and $\boldsymbol{e}_{j,k}$
      end if
    end for
  end for
end for



Example: $\boldsymbol{e}_{(j,k)}$ is replaced by $\boldsymbol{e}_{(c,p)}$

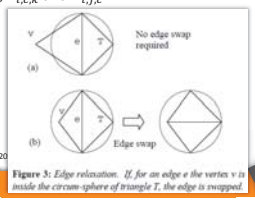


Figure 3: *Edge relaxation. If, for an edge e the vertex v is inside the circum-sphere of triangle T, the edge is swapped.*

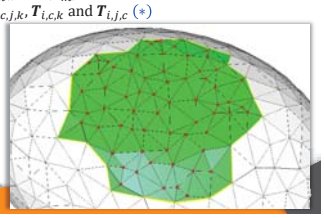
P. Liepa, (2003), **Filling holes in meshes**, *Eurographics symposium on Geometry processing (SGP '03)*, pp.200-20

- **Filling patch refinement**

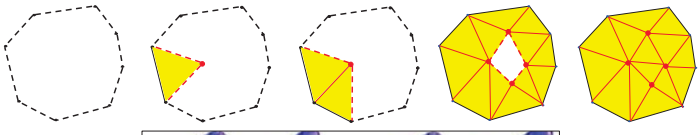
P. Liepa, (2003), **Filling holes in meshes**, *Eurographics symposium on Geometry processing (SGP '03)*, pp.200-205.

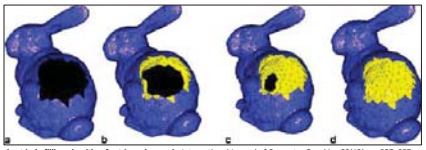
Refinement algorithm

for each vertex $\boldsymbol{v}$ on the hole boundary
  $\sigma(\boldsymbol{v}) \leftarrow$ average length of the adjacent edges in the original mesh
end for
repeat
  for each triangle $\boldsymbol{T}_{i,j,k}$ in the filling patch
    $\boldsymbol{v}_c \leftarrow$ centroid of $\boldsymbol{T}_{i,j,k}$
    $\sigma(\boldsymbol{v}_c) \leftarrow \frac{\sigma(v_i)+\sigma(v_j)+\sigma(v_k)}{3}$
    for each $m \in \{i,\ j,\ k\}$
      if $\alpha\|\boldsymbol{v}_c - \boldsymbol{v}_m\| > \sigma(\boldsymbol{v}_c)$ and $\alpha\|\boldsymbol{v}_c - \boldsymbol{v}_m\| > \sigma(\boldsymbol{v}_m)$ then
        replace $\boldsymbol{T}_{i,j,k}$ with new triangles $\boldsymbol{T}_{c,j,k}$, $\boldsymbol{T}_{i,c,k}$ and $\boldsymbol{T}_{i,j,c}$ (*)
        relax the edges $\boldsymbol{e}_{i,j}$, $\boldsymbol{e}_{i,k}$ and $\boldsymbol{e}_{j,k}$
      end if
    end for
  end for
  repeat
    relax all interior edges of the filling patch
  until no edges were swapped
until no new triangles were created in (*)



- **Another filling method: advancing front mesh (AFM) technique**
- **Overview**
  - **Updated boundary: dashed line**
  - **Newly created vertices and edges: red color**

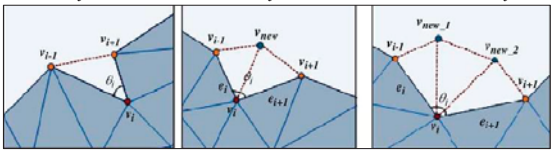




W. Zhao, S. Gao, H. Lin, (2007), **A robust hole-filling algorithm for triangular mesh**, *International Journal of Computer Graphics*, 23(12), pp.987–997

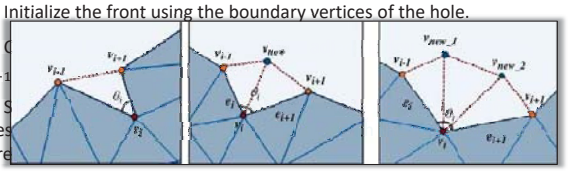
- **Another filling method: advancing front mesh (AFM) technique**
  - **Step 1**. Initialize the front using the boundary vertices of the hole.
  - **Step 2**. Calculate the angle θ_i between two adjacent boundary edges(e_i and e_{i+1}) at each vertex v_i on the front.
  - **Step 3**. Starting from the vertex v_i with the smallest angle θ_i, create new triangles on the plane determined by e_i and e_{i+1} with the three rules shown in Figure below.

$\boldsymbol{\theta_i \le 75°}$ $\quad$ $\boldsymbol{75° < \theta_i \le 135°}$ $\quad$ $\boldsymbol{135° < \theta_i}$



W. Zhao, S. Gao, H. Lin, (2007), **A robust hole-filling algorithm for triangular mesh**, *International Journal of Computer Graphics*, 23(12), pp.987–997

- **Another filling method: advancing front mesh (AFM) technique**
  - **Step 1**. Initialize the front using the boundary vertices of the hole.
  - **Step 2**. C
  - **Step 3**. S
  - **Step 4**. Compute the distance between each newly created vertex and every related boundary vertex; if the distance between them is less than the given threshold, they are merged.
  - **Step 5**. Update the front.
  - **Step 6**. Repeat steps 2 through 5 until the whole region has been patched by all newly created triangles.



W. Zhao, S. Gao, H. Lin, (2007), **A robust hole-filling algorithm for triangular mesh**, *International Journal of Computer Graphics*, 23(12), pp.987–997

Introduction to meshes processing

Software: MeshLab

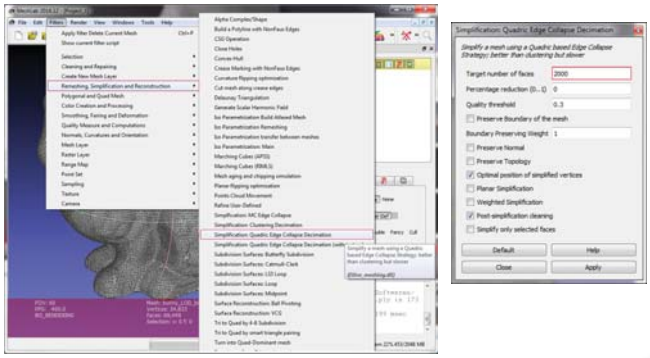
Ruding LOU
ruding.lou@ensam.eu

Arts et Métiers

MeshLab

Arts et Métiers

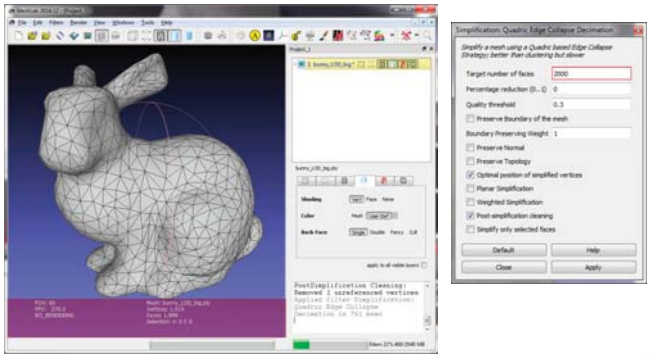
- Simplificaiton



MeshLab

Arts et Métiers

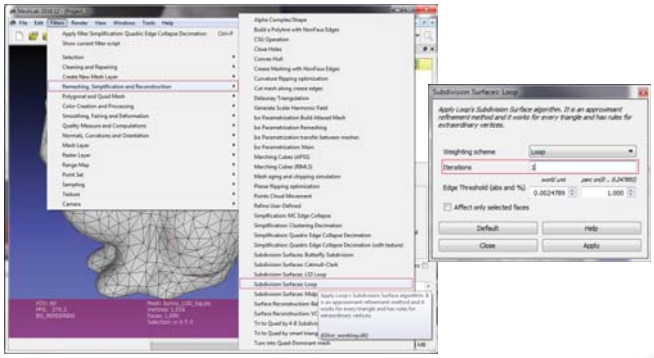
- Simplificaiton



MeshLab

Arts et Métiers

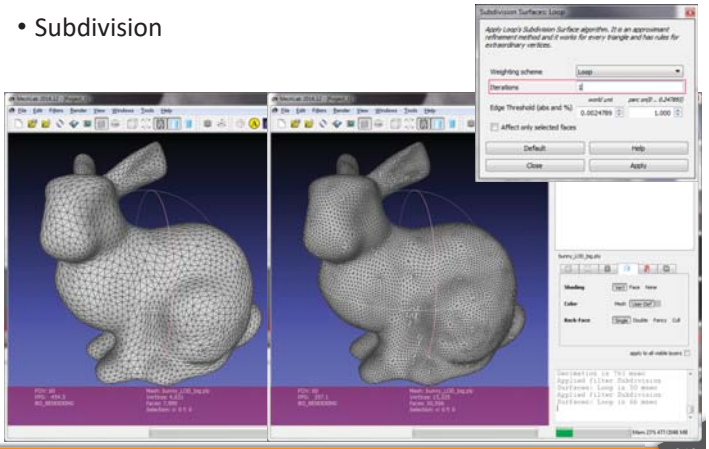
- Subdivision



MeshLab

Arts et Métiers

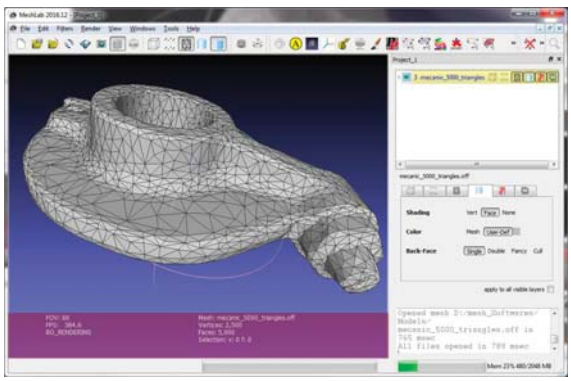
- Subdivision



MeshLab

Arts et Métiers

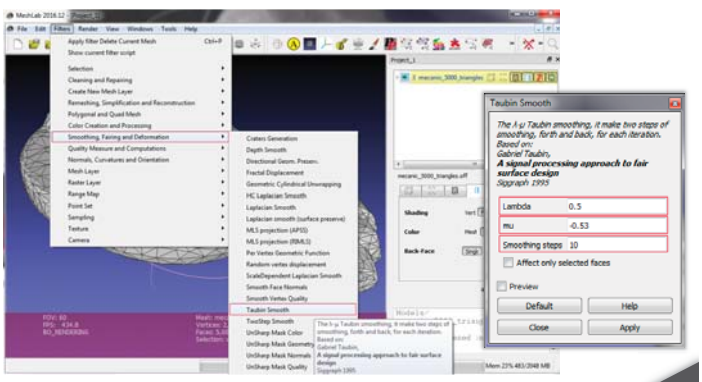
- Smoothing



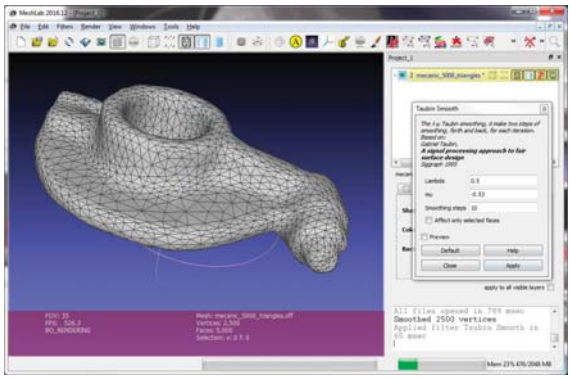
MeshLab

Arts et Métiers

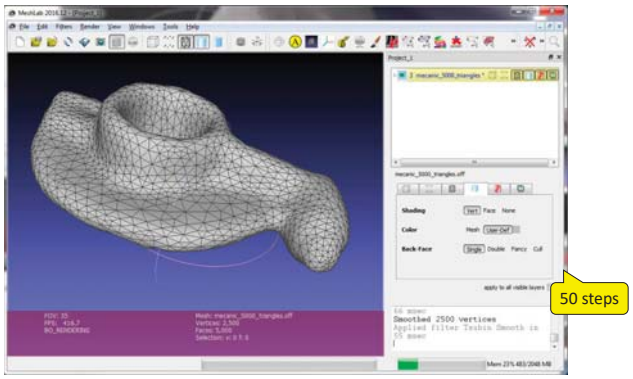
- Smoothing



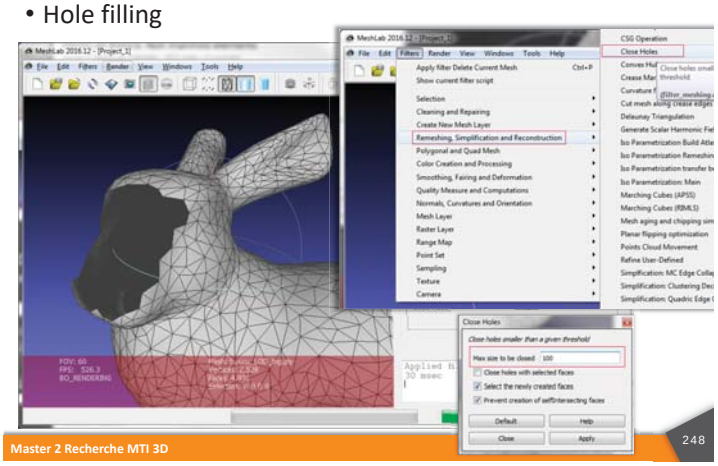
- Smoothing



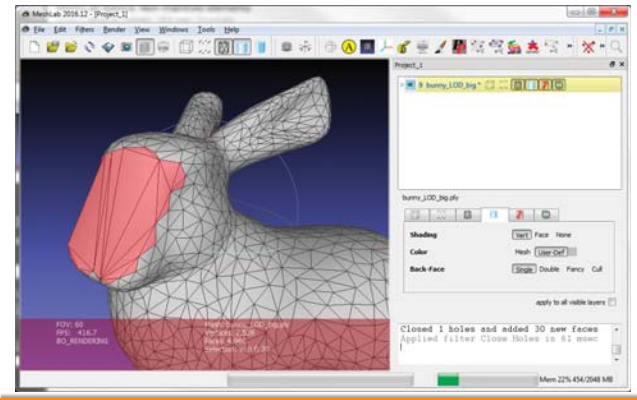
- Smoothing



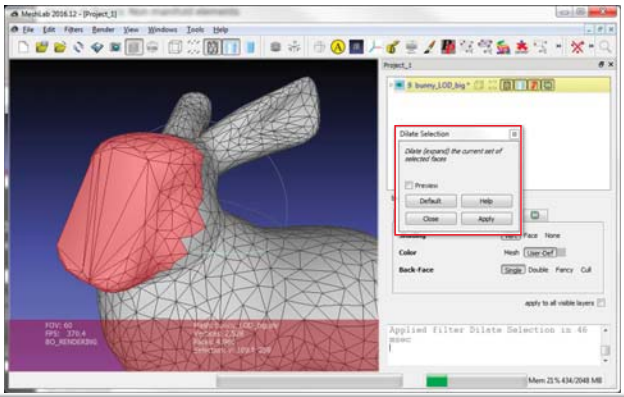
- Hole filling



- Hole filling



- Hole filling



- Hole filling + subdivision