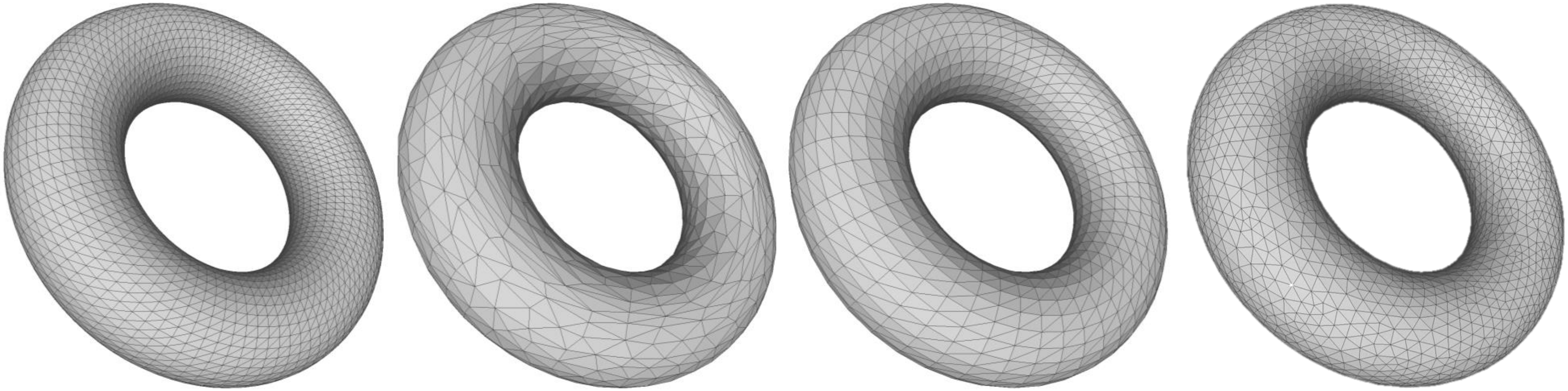
3D GEOMETRIC MODELING & PROCESSING
REMESHING

Paolo Cignoni

Istituto di Scienza e Tecnologie dell'Informazione,
Consiglio Nazionale delle Ricerche

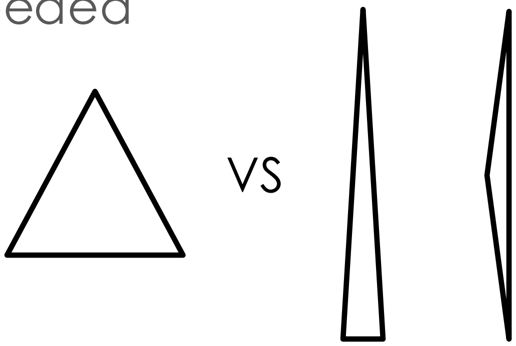
Remeshing

- Any discretization is an approximation
  - For the same abstract shape you can have many different discretizations
- No absolute ideal discretization exists



Remeshing

- Any discretization is an approximation
  - For the same abstract shape you can have many different discretizations
- No absolute ideal discretization exists
- Metrics depends on applications
  - Closeness/Distance
    - How far is my discretization from the intended shape
  - Conciseness
    - Number of primitive really needed
  - Shape/Robustness
    - Not all triangles are equals

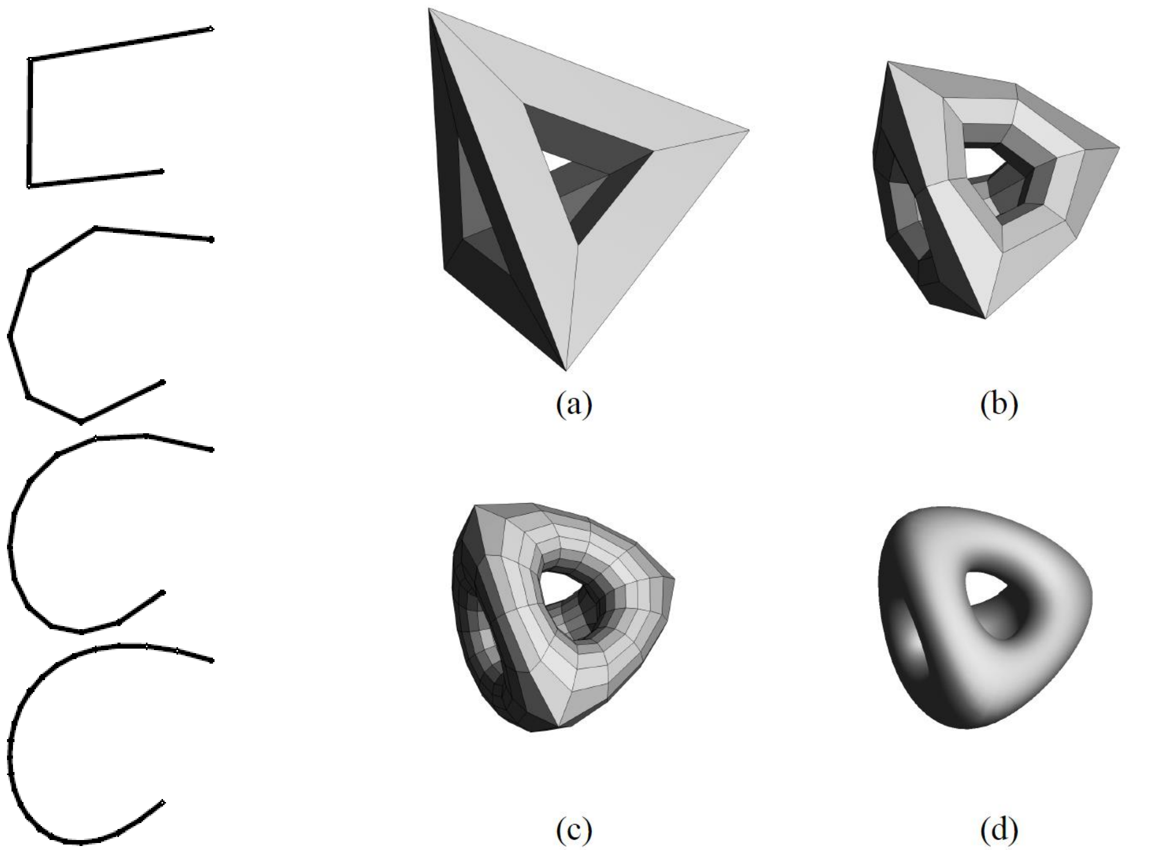


Remeshing

- **Refinement / Subdivision**
  - Your starting discretization is too coarse
    - Guess/invent information consistently
    - Metrics!
- **Coarsening / Simplification**
  - Your starting discretization is too dense
    - Drop less useful information
    - Metrics!

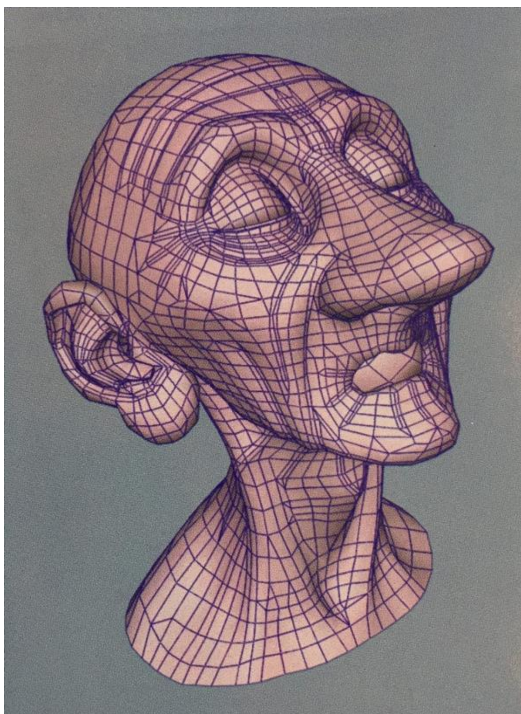
Subdivision Surfaces

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



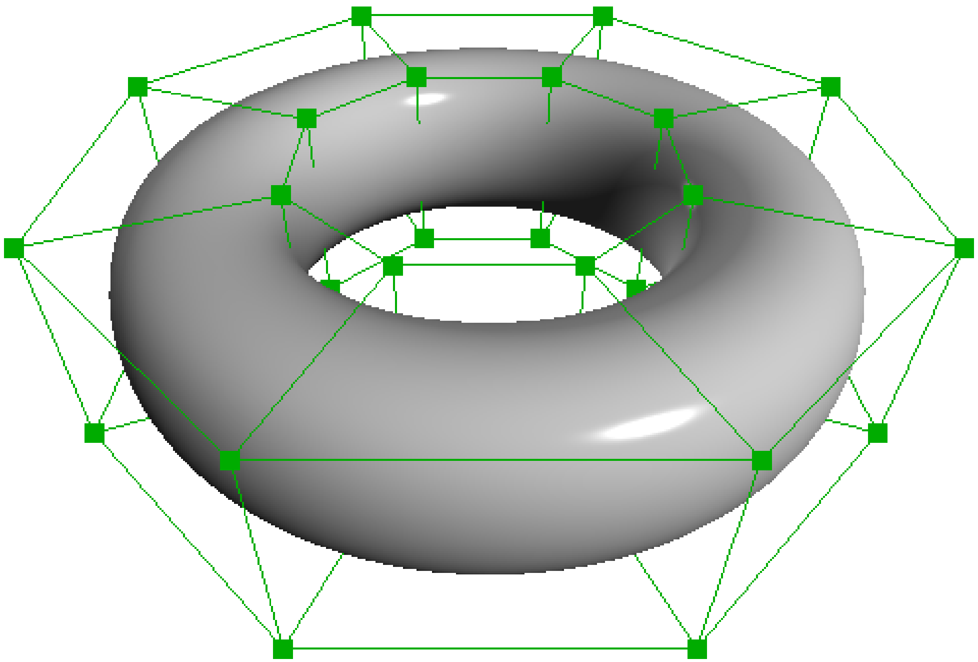
An example

- Geri's Game (1997)
- First non academic use of subdivisions surfaces

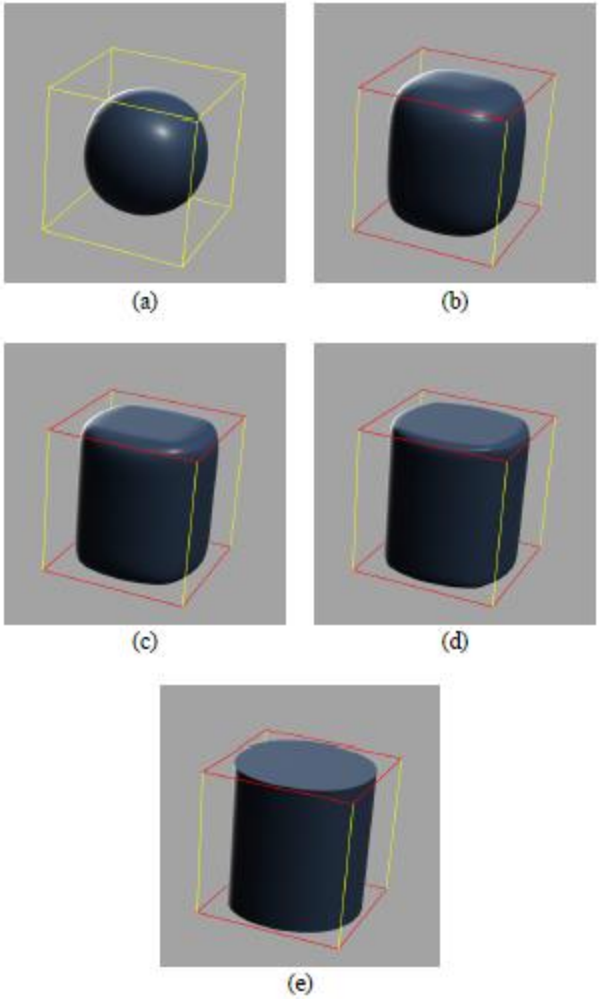


Motivation

- Why using them?
  - CONTROL
  - By adjusting the position of a few points of (a) you control the complex shape of a few control points



Sharp Features

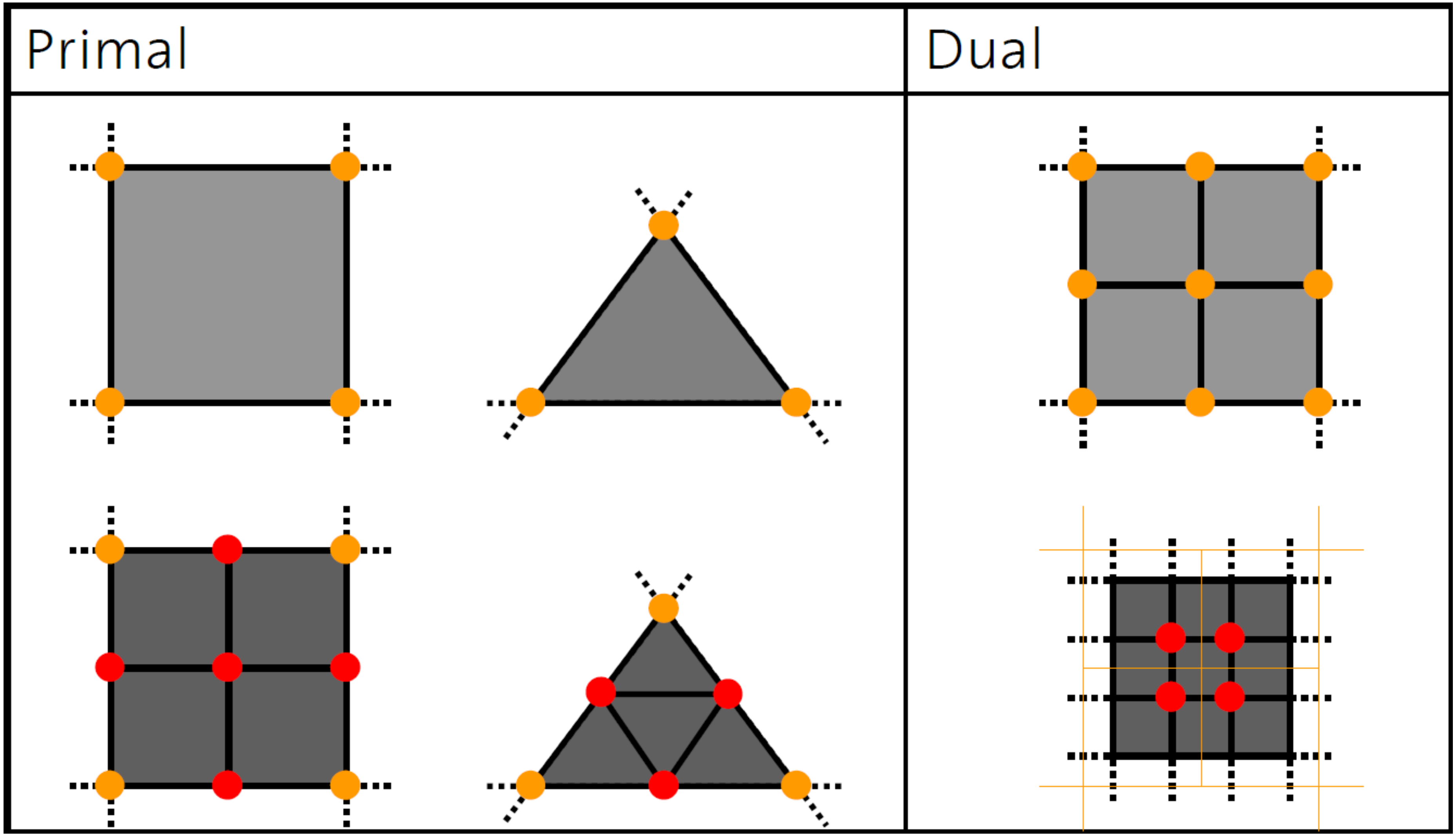


Subdivision Classification

Primal	Dual
Faces split into sub faces	New faces for each vertex, edge face

Approximating	Interpolating
Vertexes of the base mesh are just control points	Vertices of the base mesh stay fixed and you build a surface intepolating them

Subdivision Classification

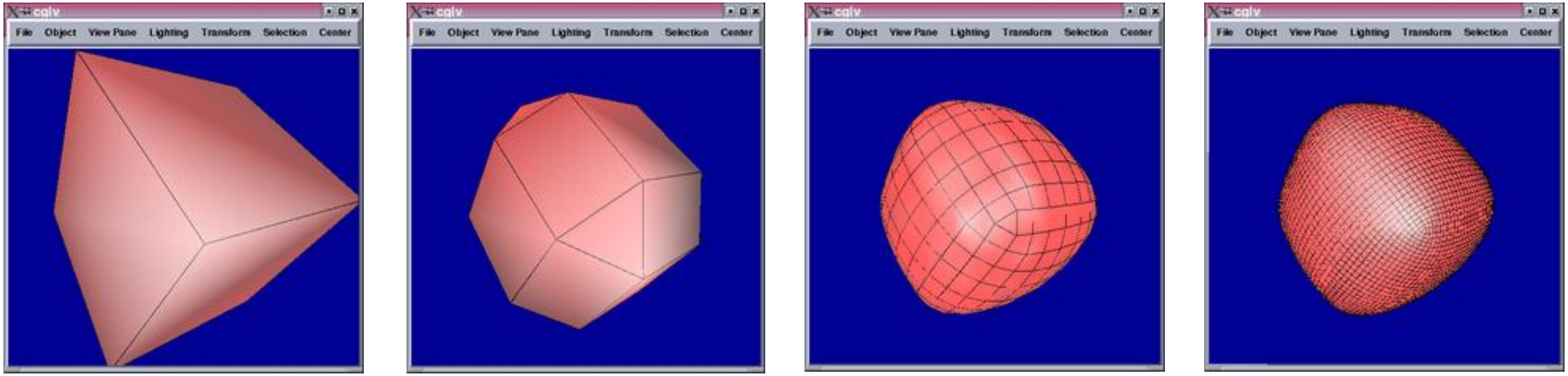


Subdivision Classification

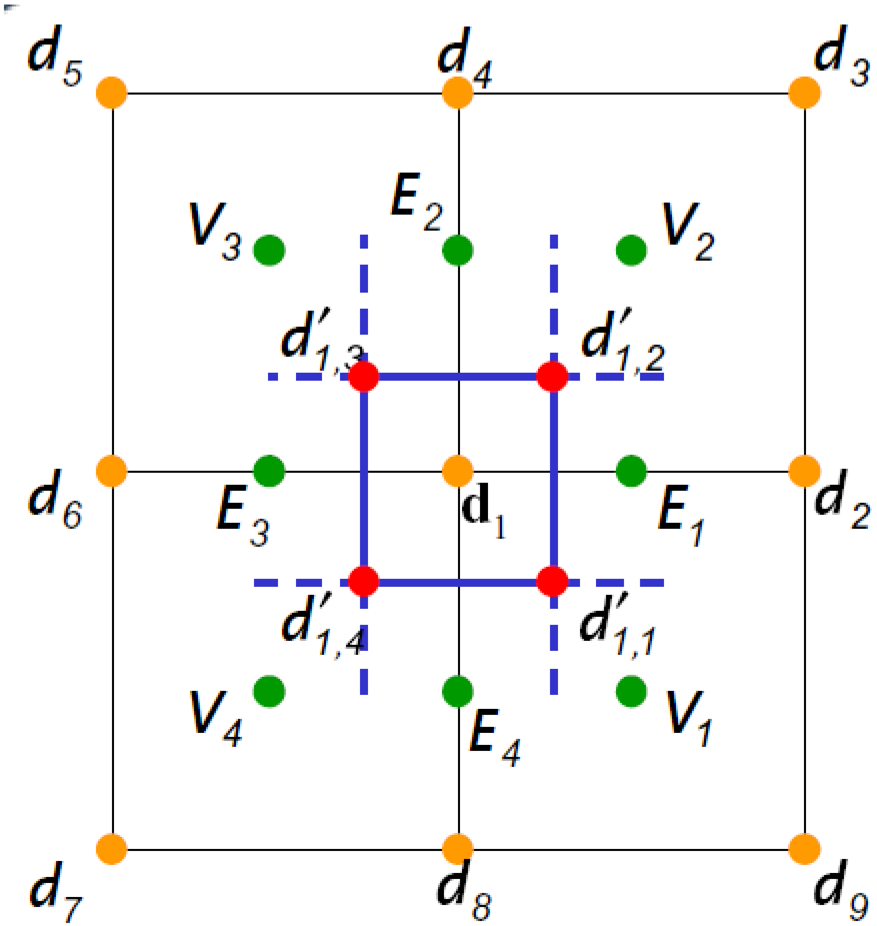
	Primal		Dual
	Triangles	Rectangles	
Approximating	Loop	Catmull-Clark	Doo-Sabin Midedge
Interpolating	Butterfly	Kobbelt	

Doo Sabin

- Dual, Approximating
- Polygonal mesh
- Creates a face for each vertex, edge and face



Doo Sabin



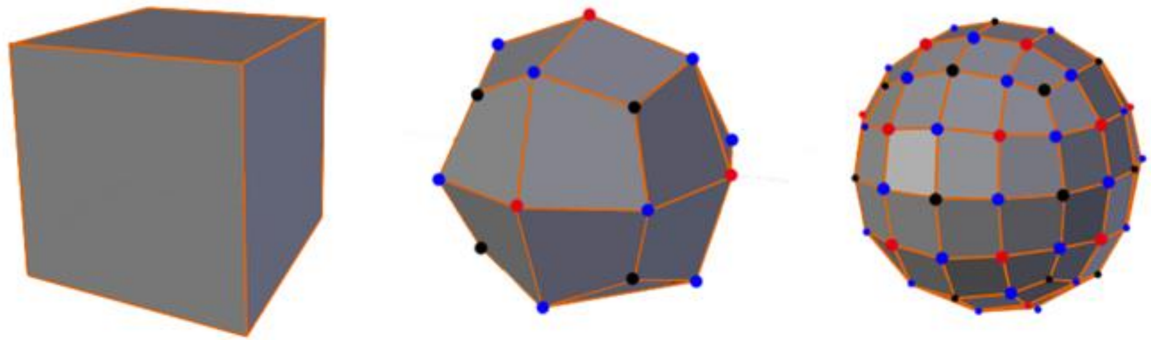
$$V_2 = \frac{1}{n} \cdot \sum_{j=1}^{n} d_j$$

$$E_i = \frac{1}{2}\left(d_1 + d_{2i}\right)$$

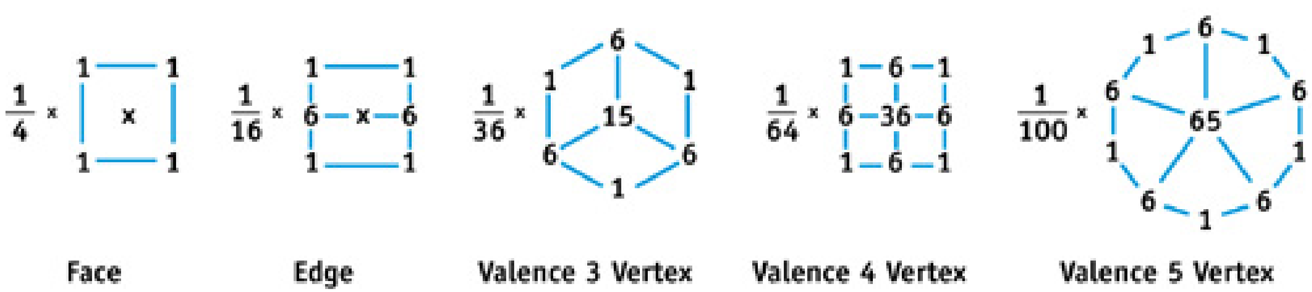
$$d'_{1,j} = \frac{1}{4}\left(d_1 + E_j + E_{j-1} + V_j\right)$$

Catmull Clark

- Polygonal / Primal / Approximating
  - 1 to 4 subdiv



  - New vertexes obtained from existing ones again using appropriate masks

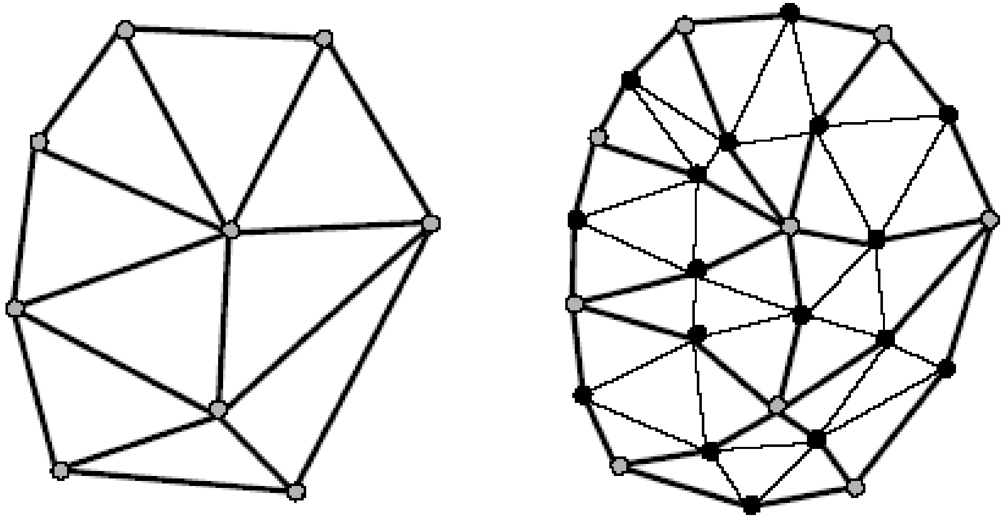


Catmull-Clark

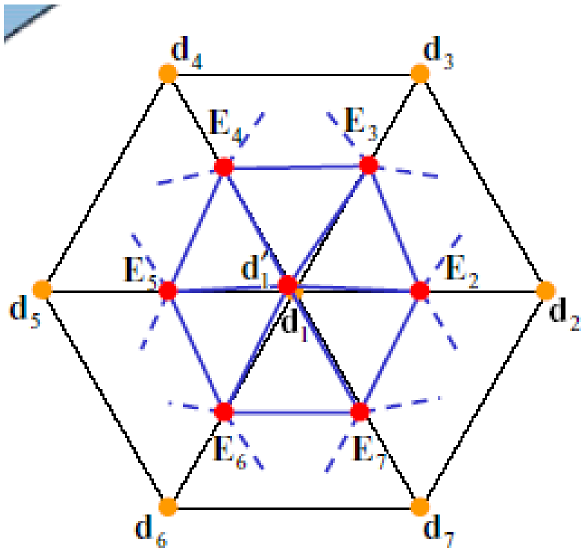
- Two Nice Properties
- Pure quad mesh after one subdivision step
- The limit surface and its derivative of Catmull–Clark subdivision surfaces can also be evaluated directly, without any recursive refinement.
  [Stam 1998]

Loop Scheme

- Triangular meshes, (*primal, approximating*)
- Edges are splitted and new vertices are reconnected to create new triangles



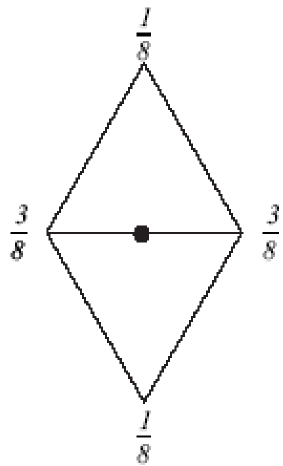
Loop Subdivision

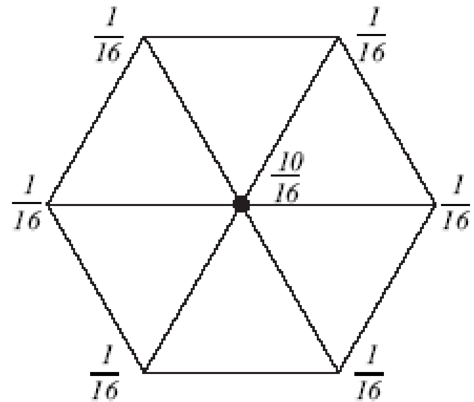


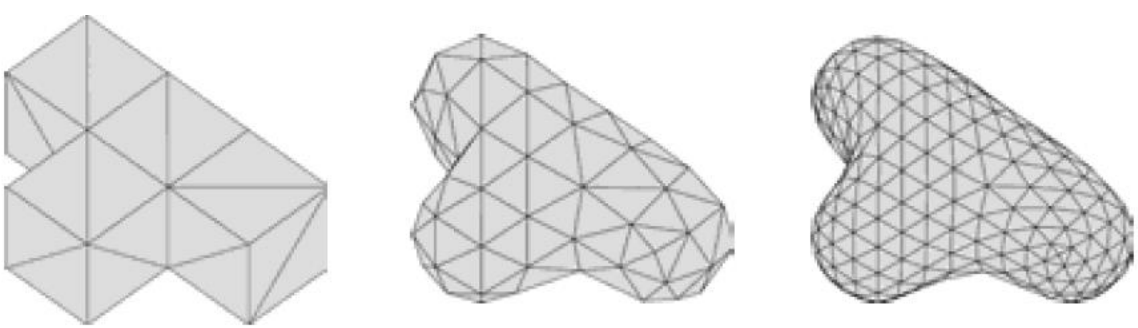
$$E_i = \frac{3}{8}(d_1 + d_i) + \frac{1}{8}(d_{i-1} + d_{i+1})$$

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

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



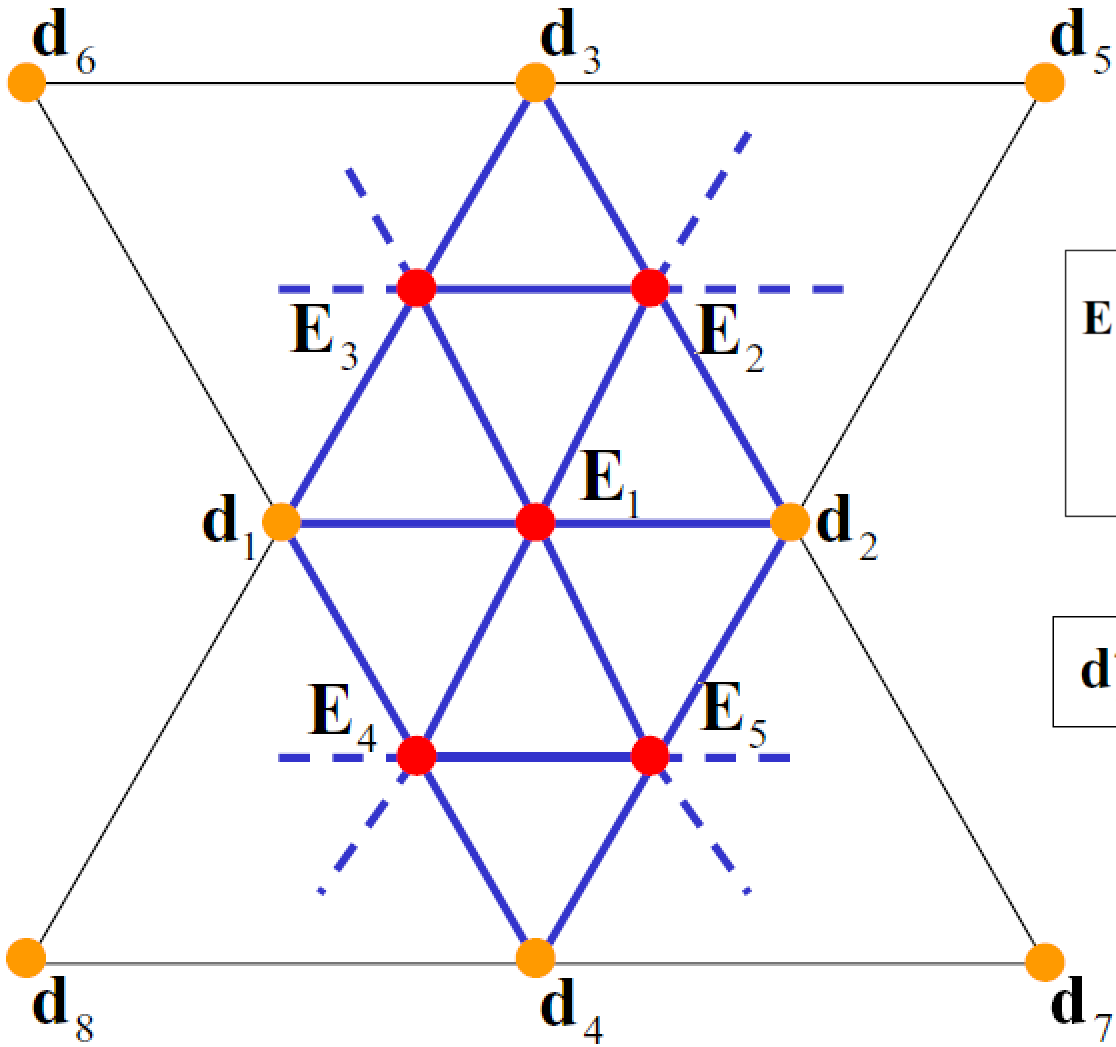




Butterfly subdivision

- **Primal / Triangular Meshes / Interpolating**
  - Continuous
  - C0 on extraordinary vertices(valence <4 or >7)
  - C1 elsewhere

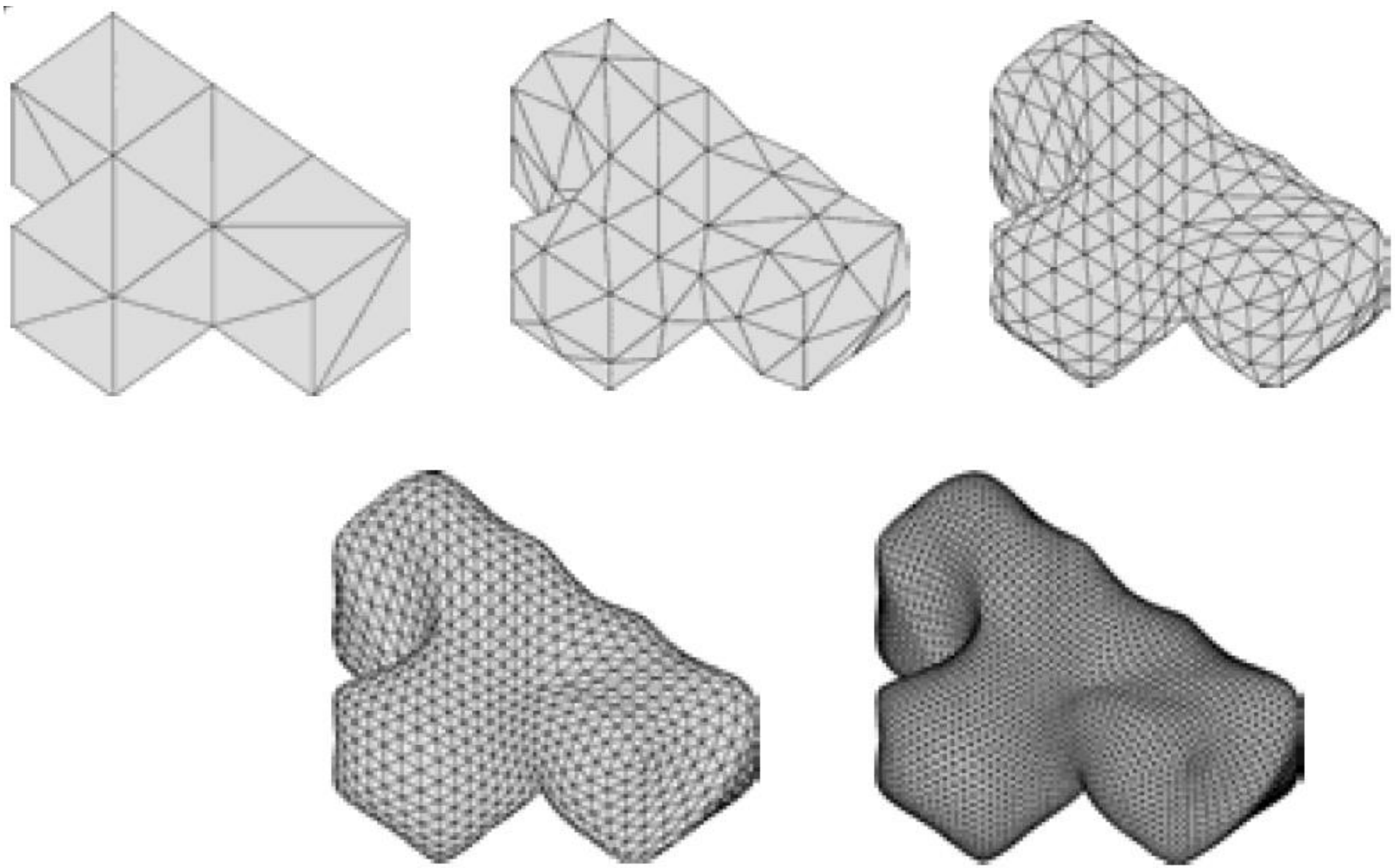
Butterfly subdivision



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

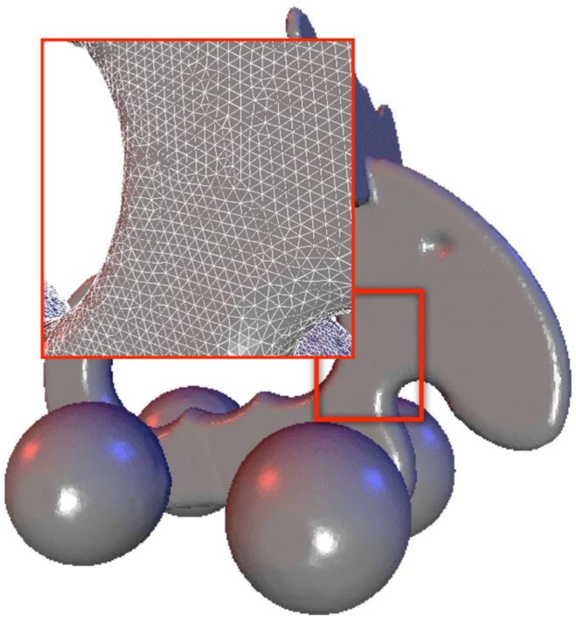
$$\mathbf{d}_i' = \mathbf{d}_i$$

Butterfly subdivision

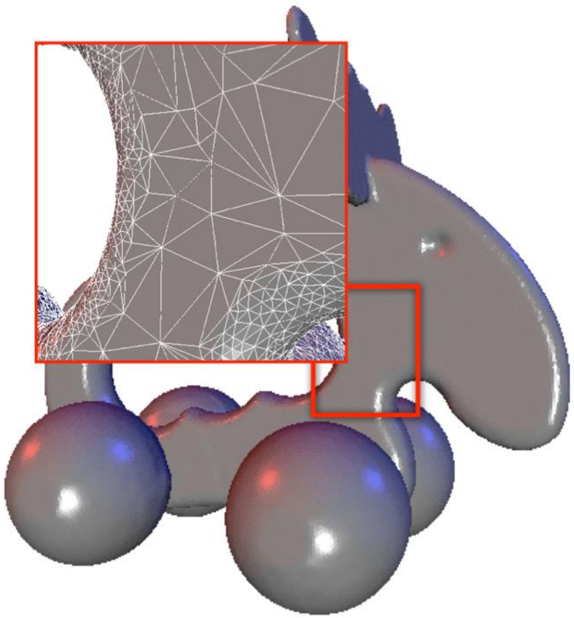


Simplification

- Reduce the amount of polygons composing a mesh with minimal effect on the geometry



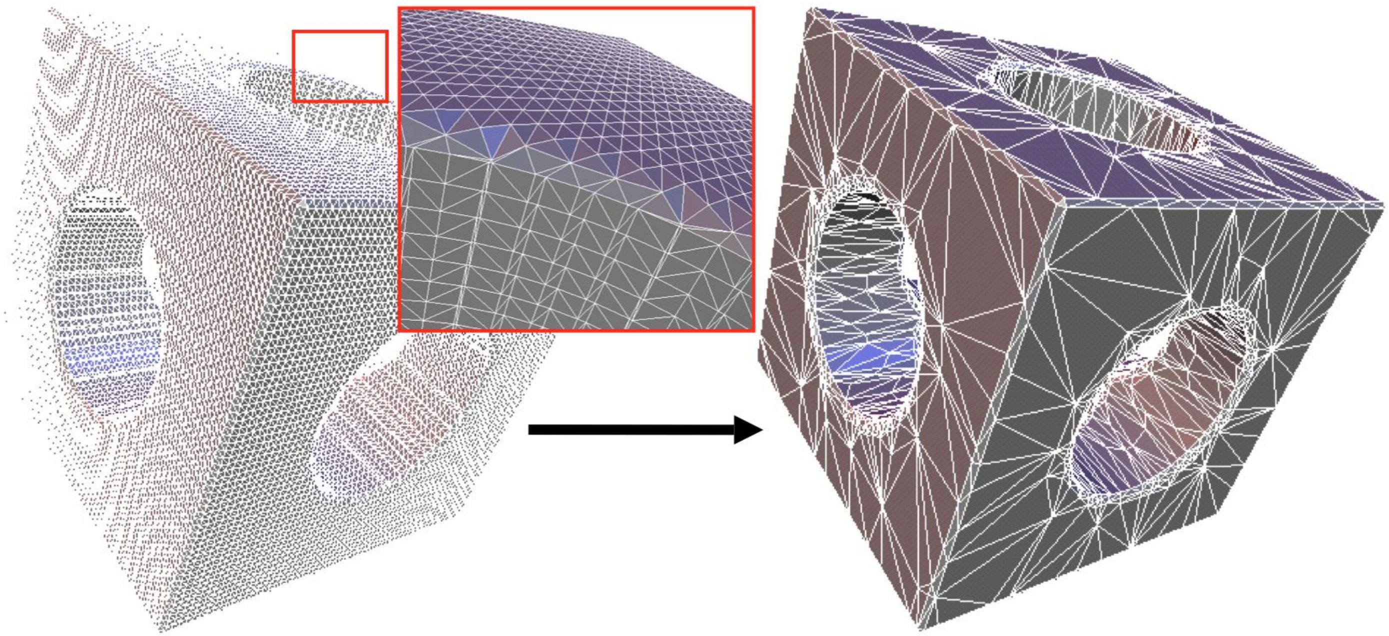
150 K triangles



80 K triangles

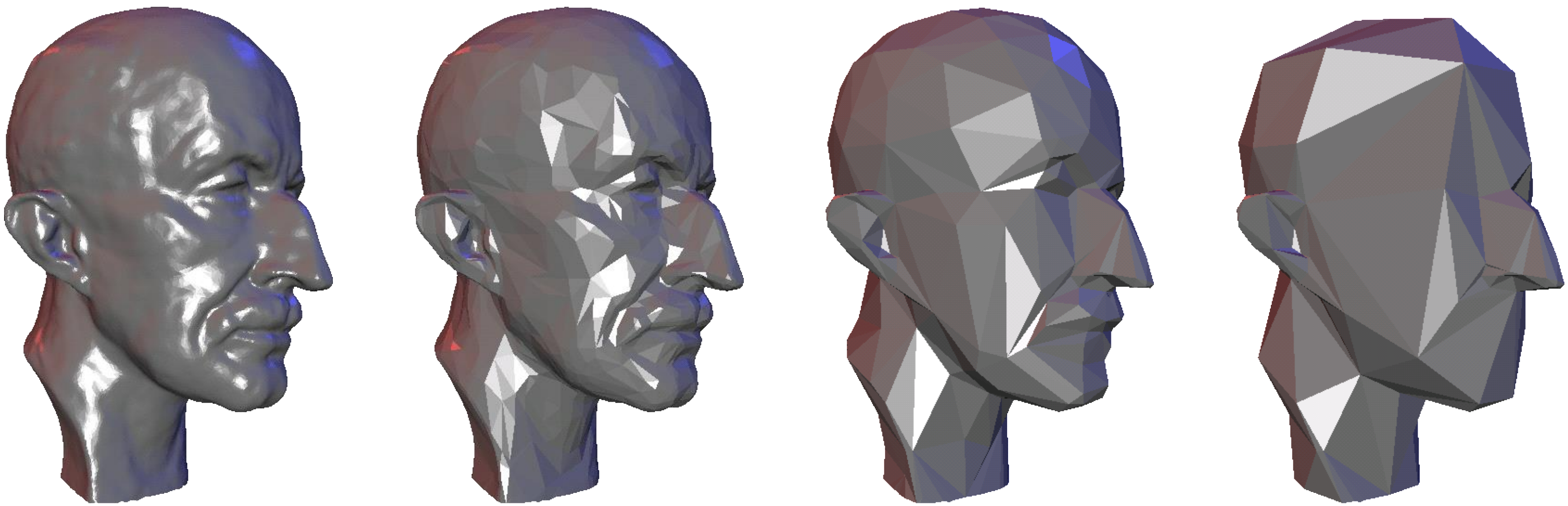
Applications

- Erase redoundant information with minimal effect on the geometry (in case of iso-surface extraction)



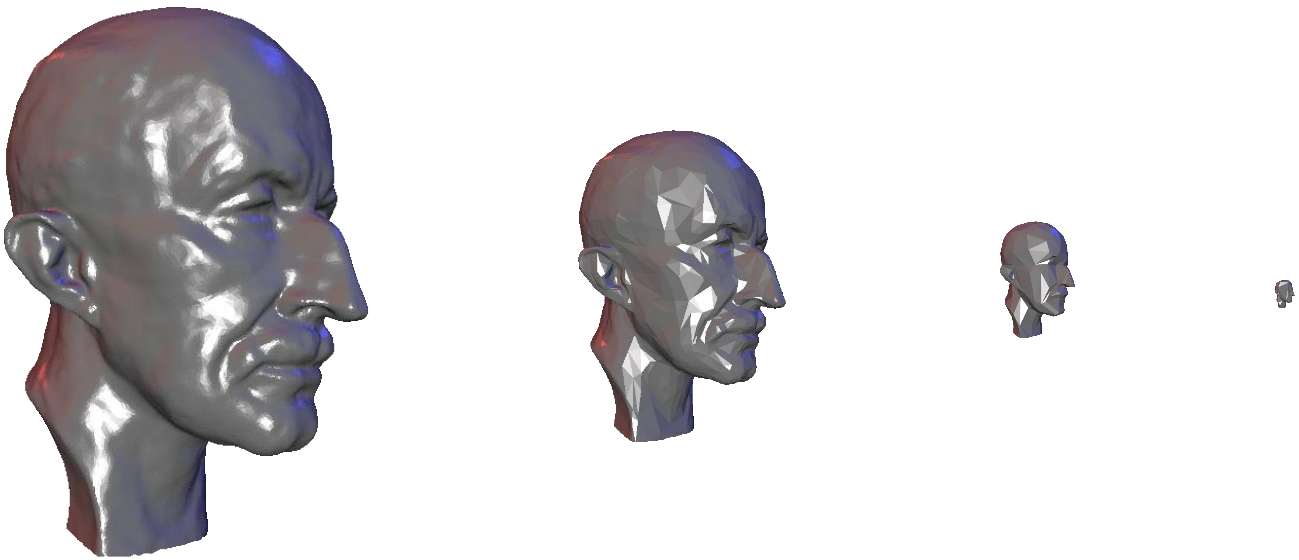
Applications

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



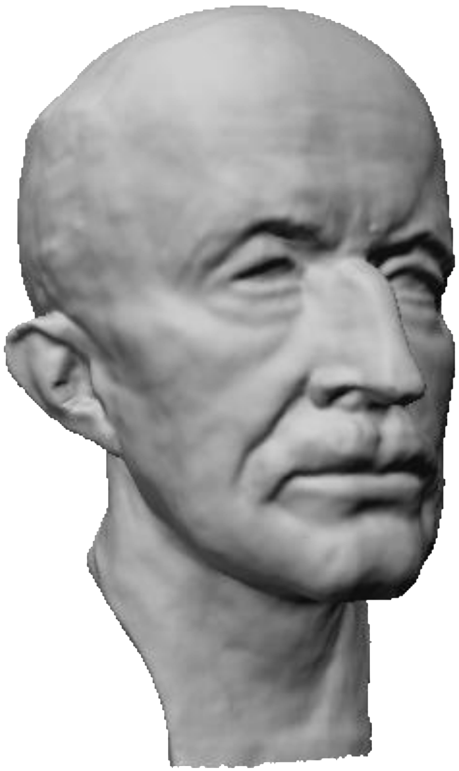
Applications

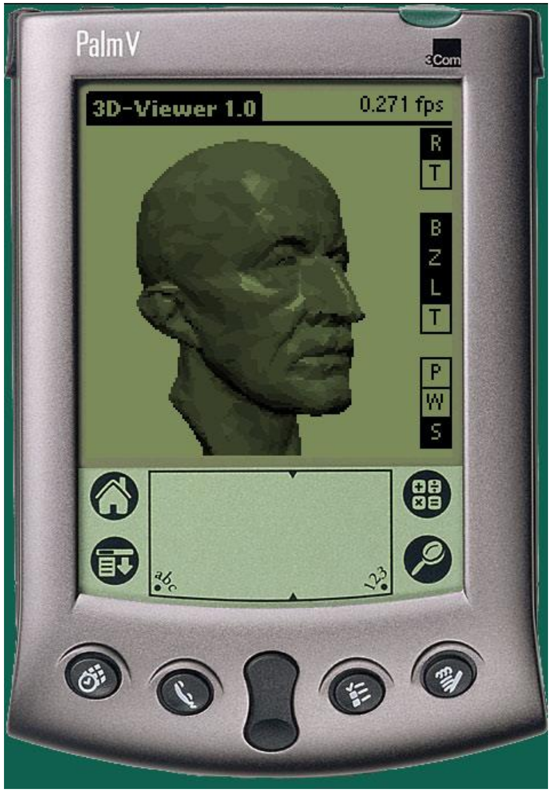
- Multi-resolution hierarchies for
  - efficient geometry processing
  - level-of-detail (LOD) rendering



Applications

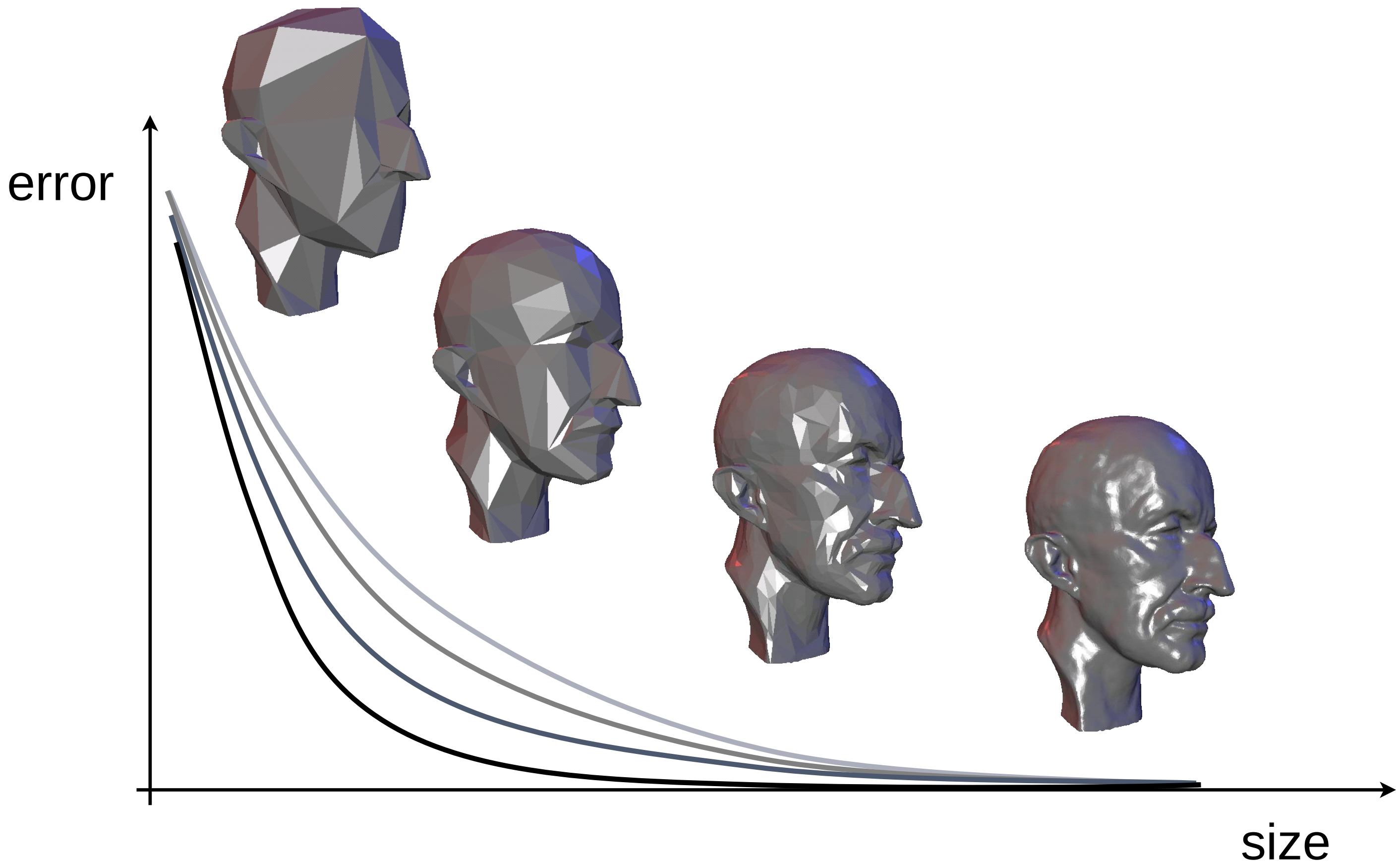
- • Adaptation to hardware capabilities





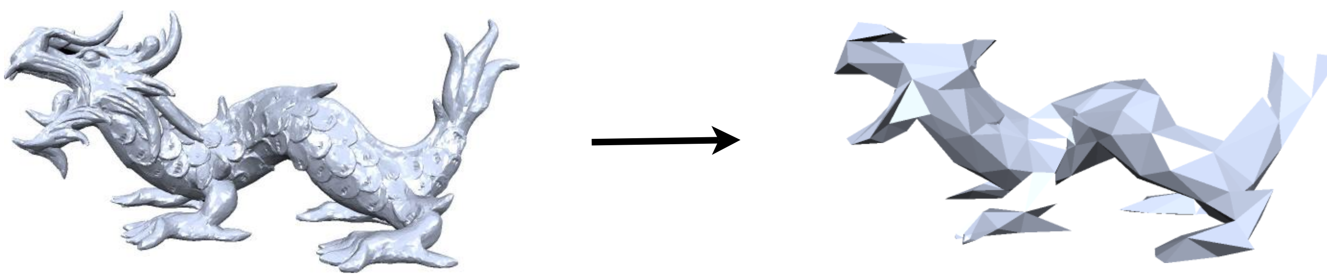
Size-Quality Tradeoff

- Complexity vs accuracy is a non linear relation



Problem statement

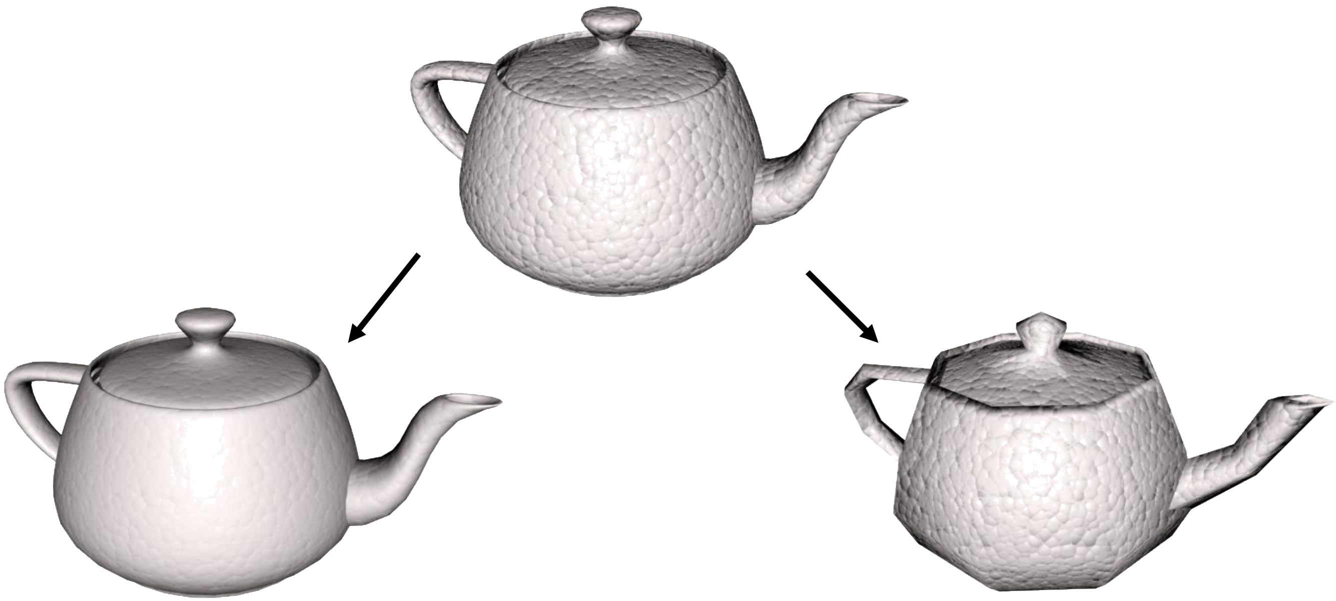
- Given: $M=(V,F)$
- Find: $M' = (V', F')$ such that
  - $|V'| = n < |V|$ and $||M - M'||$ is minimal, or
  - $||M - M'|| < \varepsilon$ and $|V'|$ is minimal
- Reduce the number of vertices minimizing the **approximation error**, or
- Keep the **error** below a threshold and minimize the number of vertices



M → M′

What is Approximation Error

- Quantifies the notion of “similarity” , Two kinds of similarity:
  - Geometric similarity (surface deviation)
  - Appearance similarity (material, normal…)



Appearance Similarity

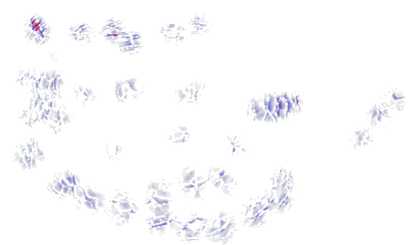
- Difference between two images: (trivial)

$$D(I_1, I_2) = \frac{1}{n^2} \sum_x \sum_y d(I_1(x,y), I_2(x,y))$$

- Difference between two objects: Integrate the above over all possible views

I_1

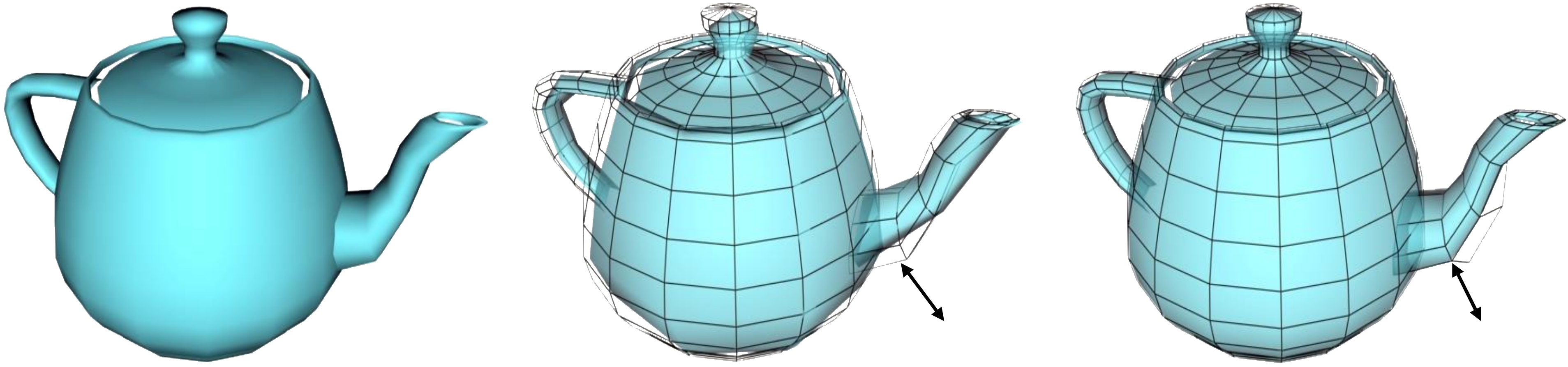
I_2



$I_1 - I_2$

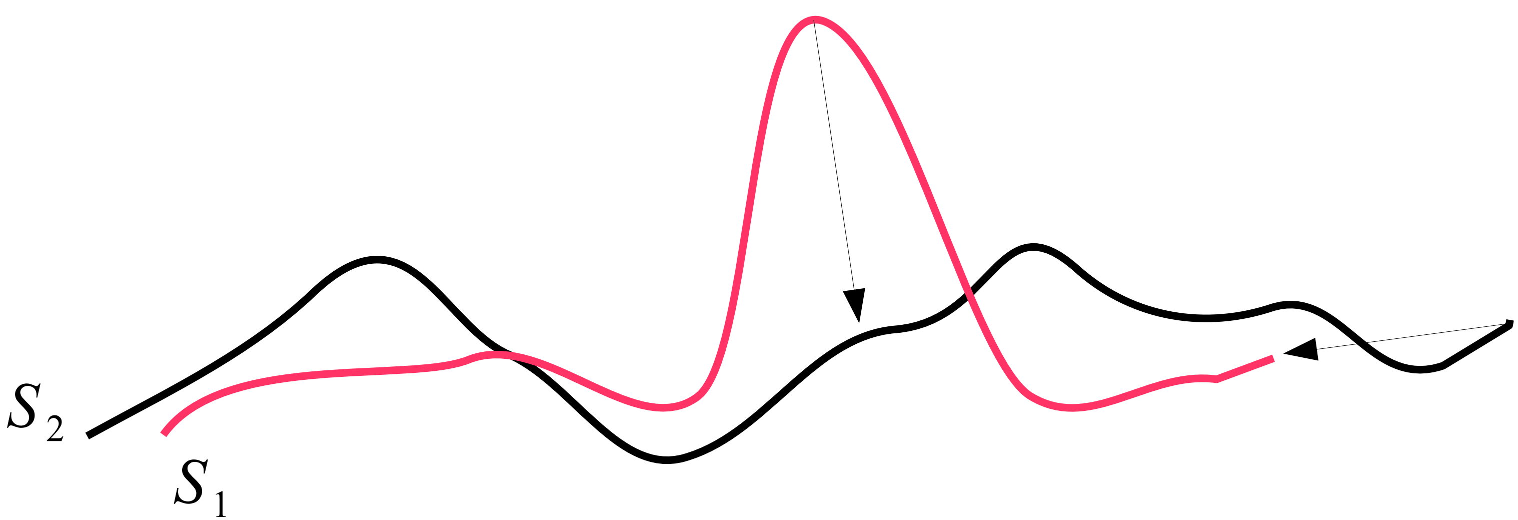
Geometric Similarity

- Two main components:
  - Distance function
  - Function Norm:
    - L_2: average deviation
    - L_{inf} : maximum deviation - Hausdorff distance



Hausdorff Distance

$$D_H(S_1, S_2) = \max_{x \in S_1}(\min_{x \in S_2} D(x, y))$$



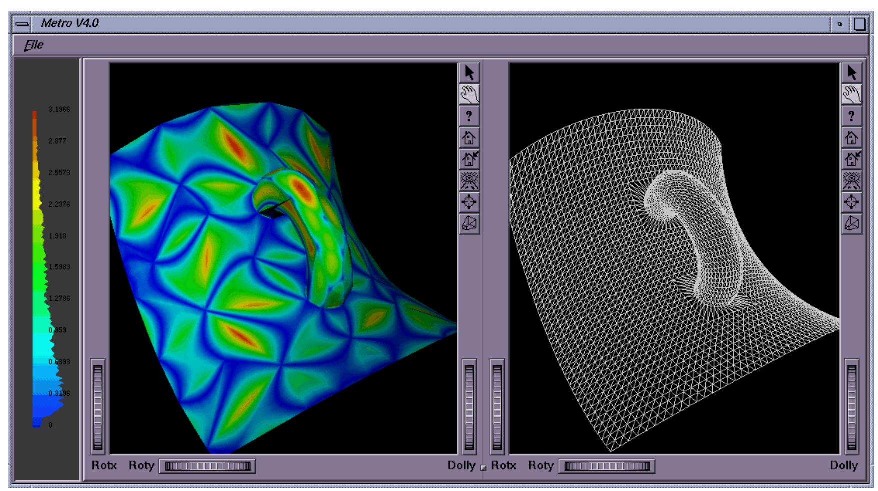
Symmetric version

$$D(S_1, S_2) = \max\{D_H(S_1, S_2), D_H(S_2, S_1))$$

Hausdorff Distance: How to compute

- Approximate as:
  1. Sample one surface surface (uniformly distributed)
  2. For each point compute $$\max_{y \in S_2} D(x, y)$$

Also consider using average distance



Problem statement

- Given: M=(V,F)
- Find: M' = (V', F') such that
  - $|V'| = n < |V|$ and $||M - M'||$ is minimal, or
  - $||M - M'|| < \varepsilon$ and $|V'|$ is minimal
- Reduce the number of vertices minimizing the **error**, or
- Keep the **error** below a threshold and minimize the number of vertices

HARD .. The space of solution is huge!!!

NP- Hardness

- It is NP-Hard to decide if a given surface of *n* vertexes can be ε-approximated with a surface composed by *k* vertices.

  Agarwal, Pankaj K., and Subhash Suri. "Surface approximation and geometric partitions." SIAM Journal on Computing 27.4 (1998): 1016-1035.

- But even the 2D version of the problem is NP-Hard
  - Simplifying a polyline to *k* vertexes so that it ε-approximate a optimal simplification using the undirected Hausdorff distance is NP-hard. The same holds when using the directed Hausdorff distance from the input to the output polyline, whereas the reverse can be computed in polynomial time.

    van Kreveld, Marc, Maarten Löffler, and Lionov Wiratma. "On Optimal Polyline Simplification using the Hausdorff and Frechet Distance." arXiv preprint arXiv:1803.03550 (2018).

Heuristics: Incremental methods

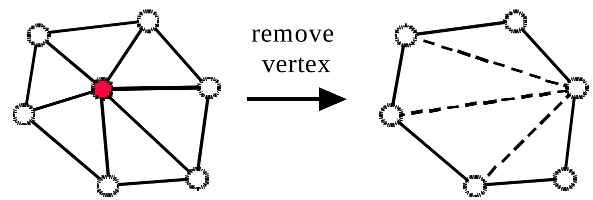
- Based on **Local Updates Operations**

- **All of the methods such that:**
  - simplification proceeds as a sequence of small changes of the mesh (in a greedy way)
  - each update reduces mesh size and [~monotonically] decreases the approximation precision

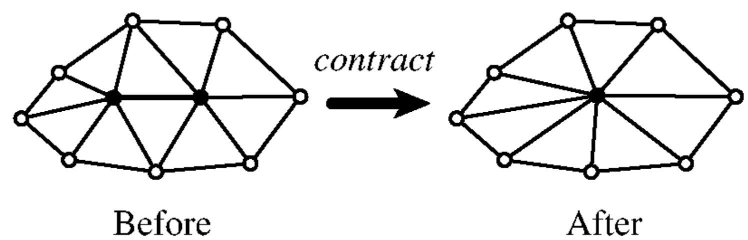
Local Operations

- vertex removal
- edge collapse
  - preserve location (one among the 2 vertex)
  - new location
- triangle collapse
  - preserve location (one among the 3 vertex)
  - new location

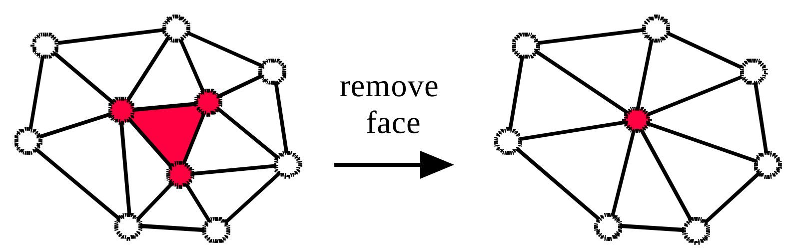
No. Faces



_n-2_



_n-2_



_n-4_

The common framework

- **Loop**{

select the element to be deleted/collapsed;

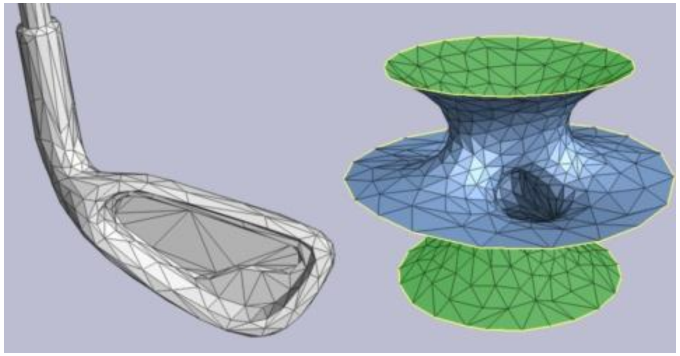
evaluate approximation introduced; (simulate the operation)

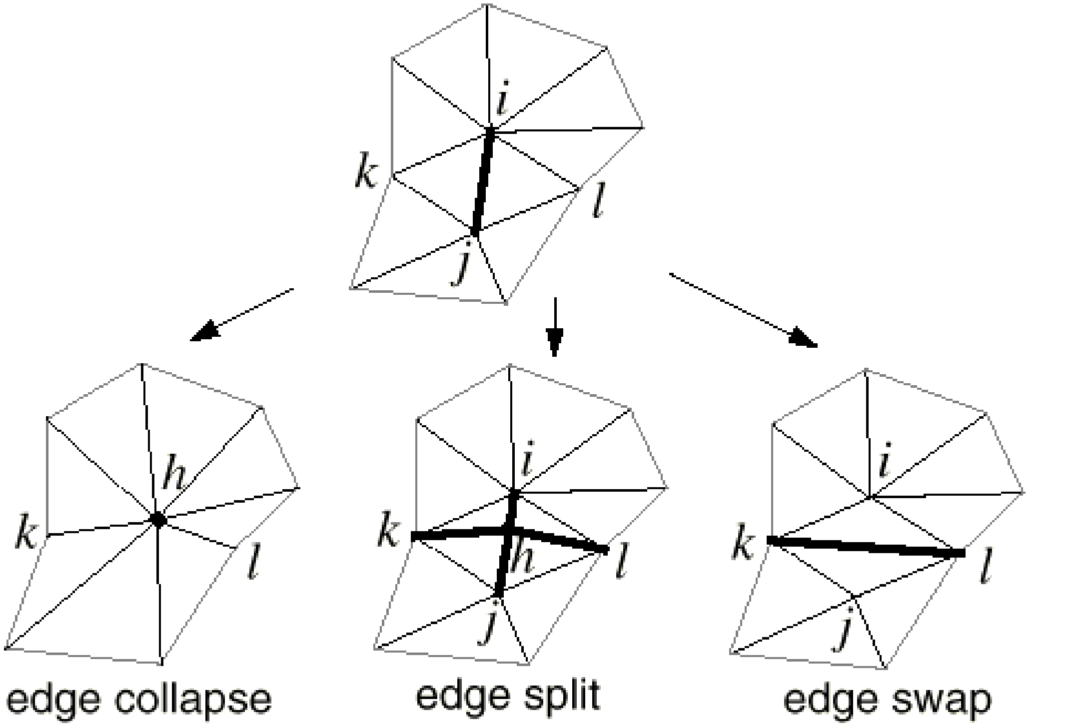
update the mesh after deletion/collapse;

} **until** mesh **size/precision** is satisfactory;

Mesh Optimization

- As in [Hoppe et al. '93]
- Simplification based on the iterative execution of :
  - edge collapsing
  - edge split
  - edge swap





Mesh Optimization

approximation quality evalued with an energy function :

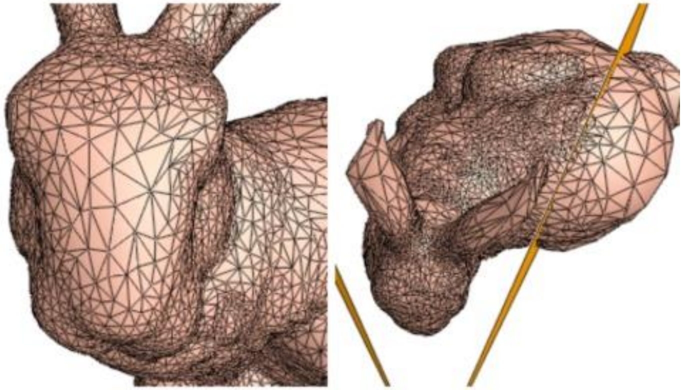
$$E\ (M) = E_{dist}\ (M) + E_{rep}\ (M) + E_{spring}\ (M)$$

which evaluates geometric **Fitness** and repr. **Compactness**

E_{dist} : sum of squared distances of the original points from M

E_{rep} : factor proportional to the no. of vertex in M

E_{spring} : sum of the edge lenghts



Greedy Approach (bounded error)

- For each region{
  1. evaluate quality after simulated operation
  2. put the operation in the heap (quality, region)

  }

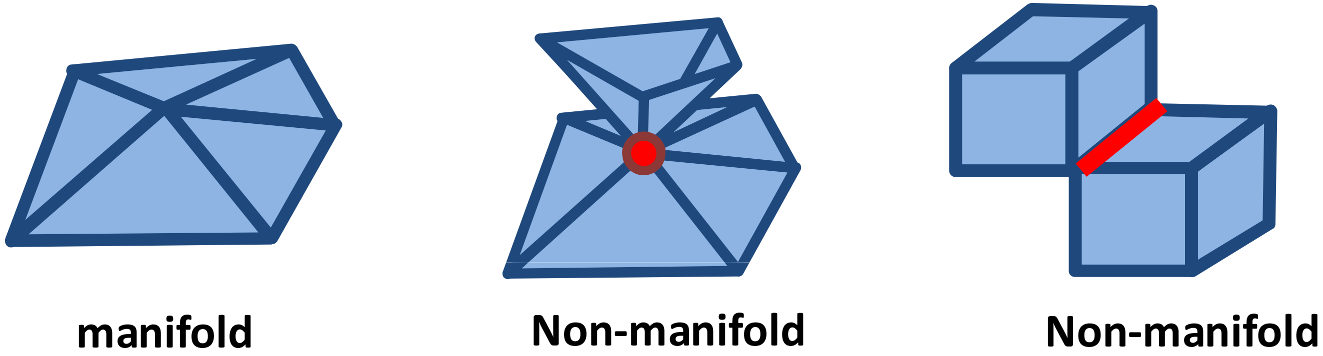
- Repeat{
  - pick best operation from the heap
  - If introduced error <**ε{**
    - Execute the operation
    - **Update heap**

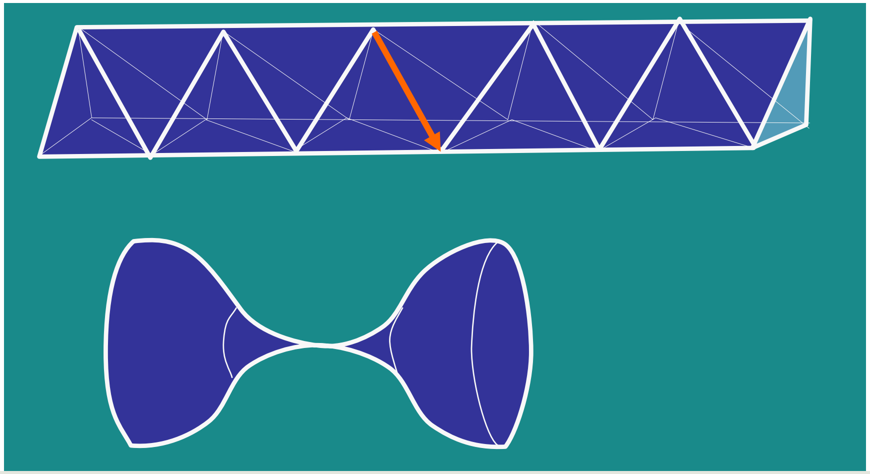
    }

} Until no further reduction possible

Simplification: Topology Preservation

- Edge collapse operation may create non manifoldness





Simplification: Topology Preservation

- Let Σ be a 2 simplicial complex without boundary Σ' is obtained by collapsing the edge e = (ab)
- Let Lk (σ) be the set of all the faces of the co-faces of σ disjoint from σ

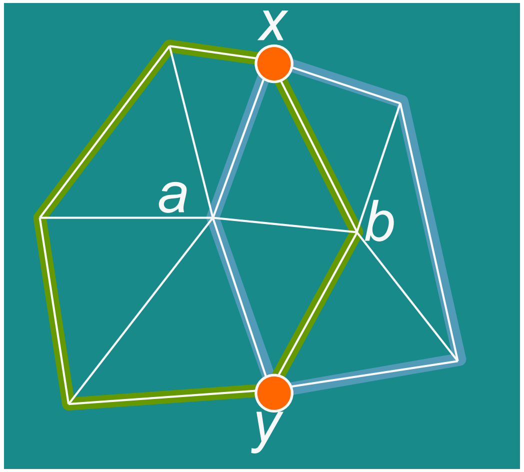
Σ and Σ' are homeomorphic ***iff***

$$Lk(a) \cap Lk(b) = Lk(ab)$$

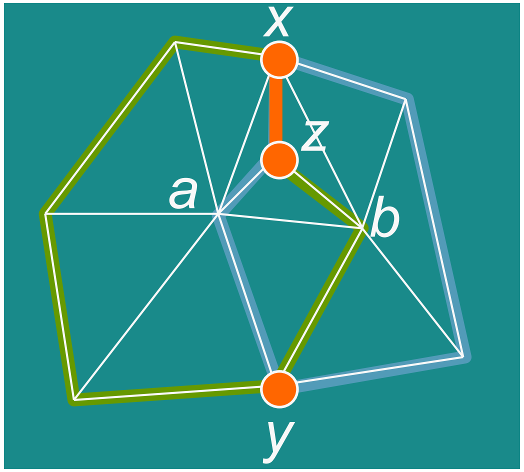
[Dey 99]

Simplification: Topology Preservation

- Lk(a) ∩ Lk(b)= {x,y }=Lk(ab)

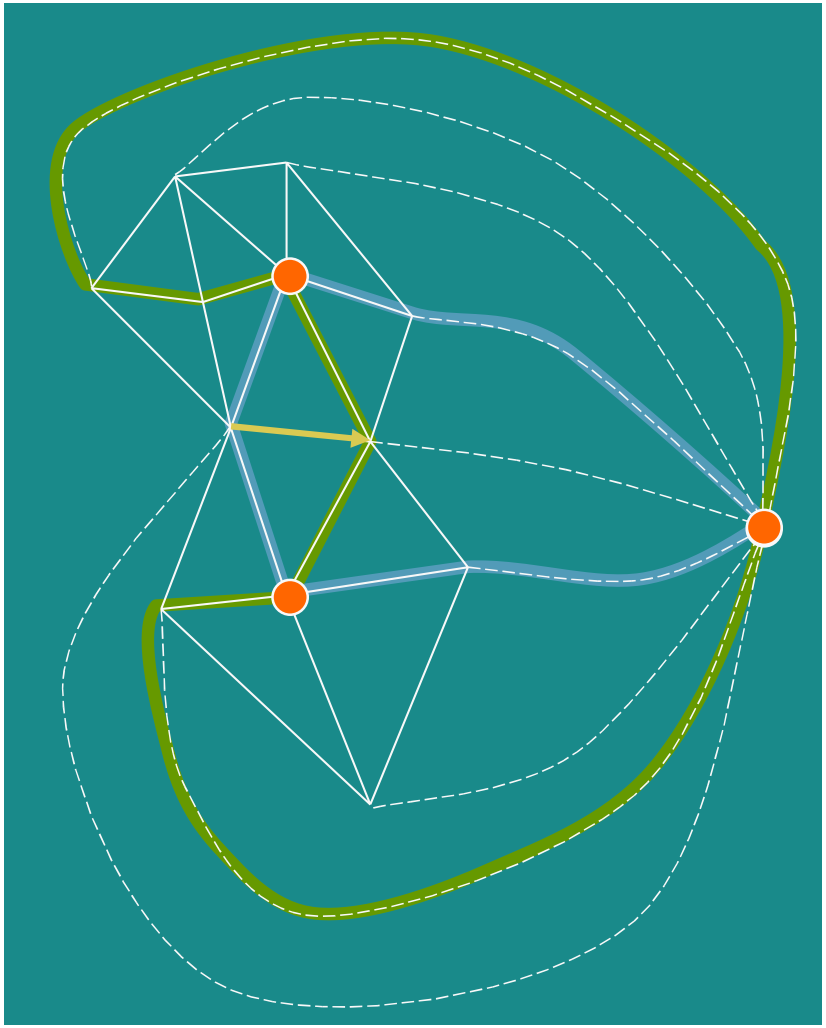


- Lk(a) ∩ Lk(b) ={x,y,z,zx} ≠ {y,z}= Lk(ab)



Topology Preservation

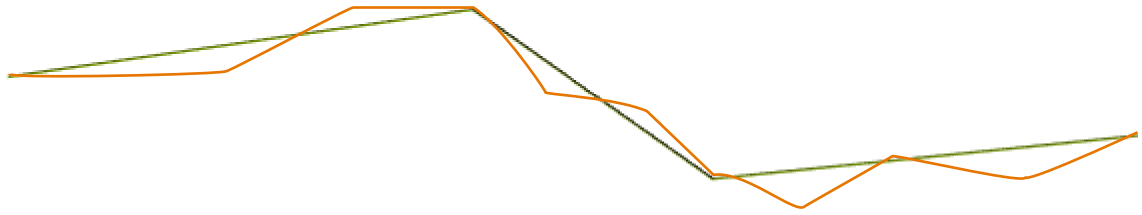
- Mesh with boundary can be managed by considering a dummy vertex v_d and, for each boundary edge e a dummy triangle connecting e with v_d.
- Think it wrapped on the surface of a sphere



Simplification: Efficient Evaluation

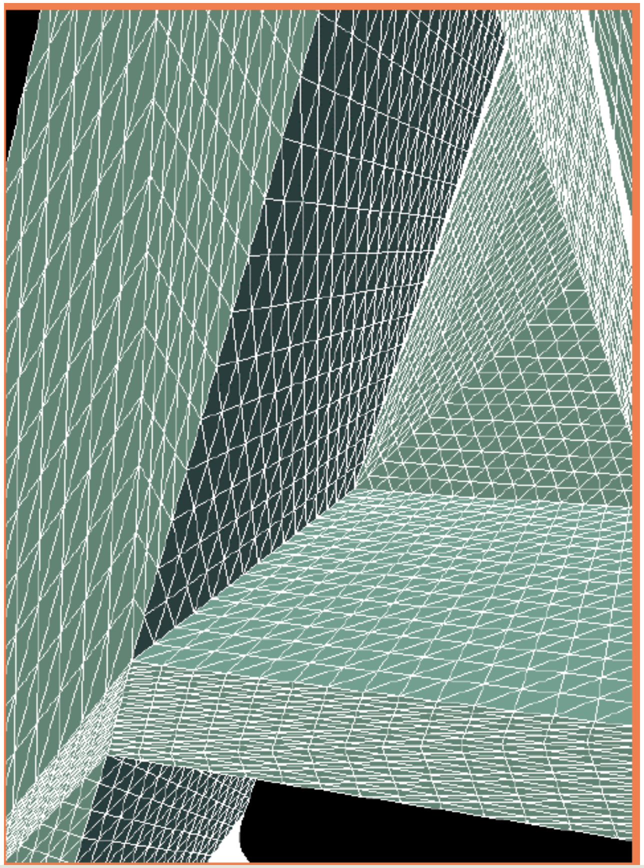
- Evaluating the error introduced by a collapse efficiently is not trivial

- Ideally use Hausdorff
  - problem: at the beginning is easy (few points approximate well H) but at the end it become costly (you need a lot of time to evaluate properly)



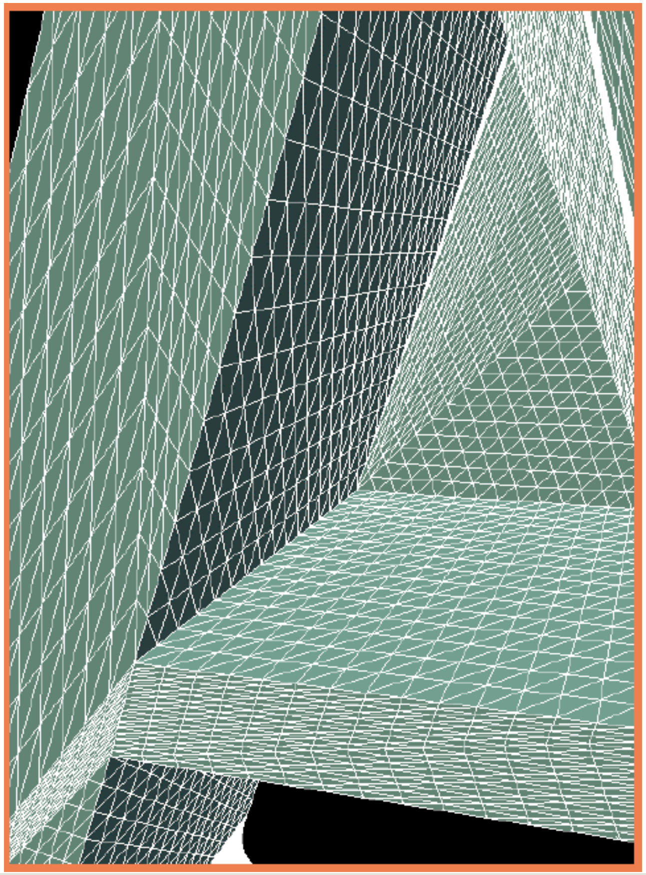
Interpolating Positions (edge collapse)

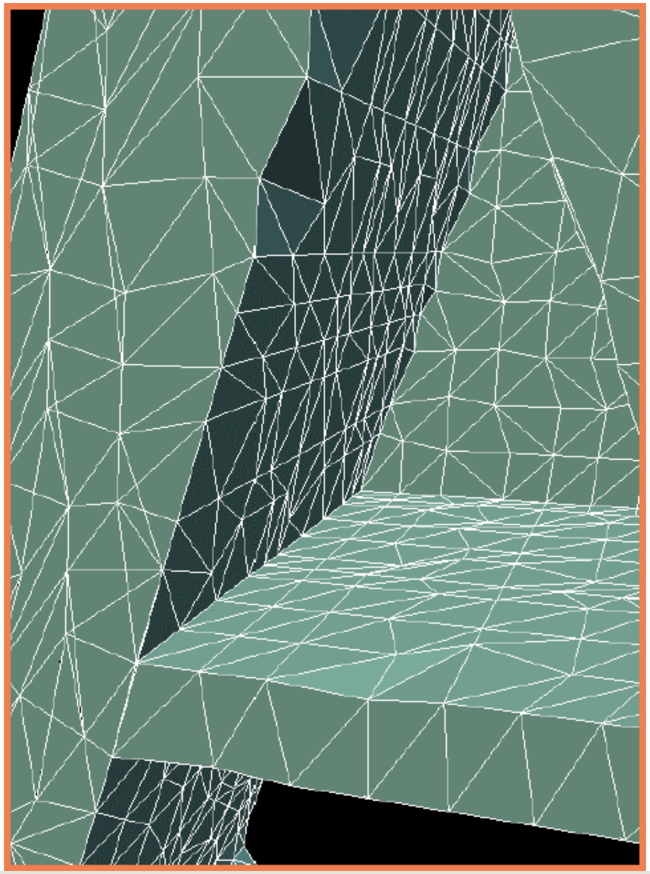
- Average Vertex Position



Interpolating Positions (edge collapse)

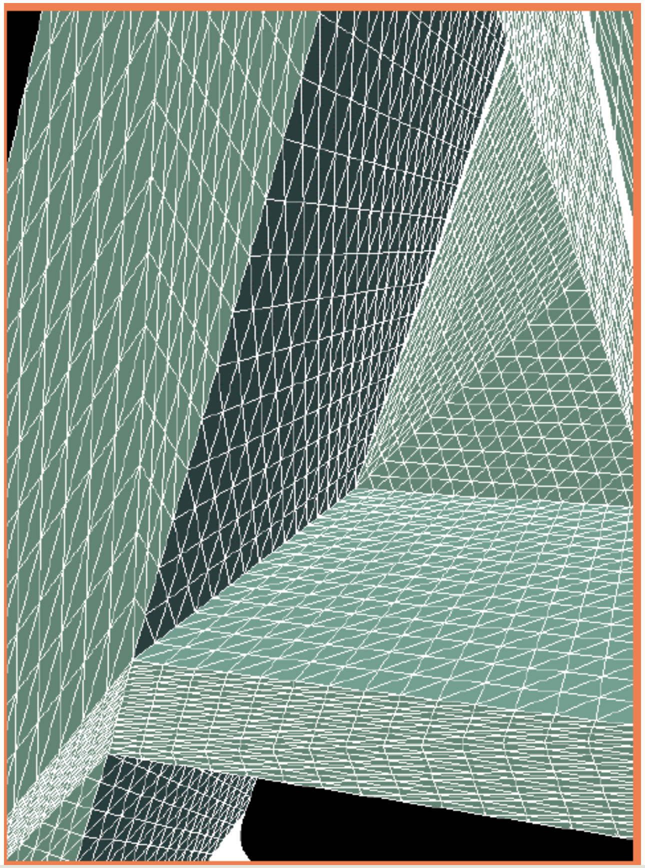
- Median Vertex Position

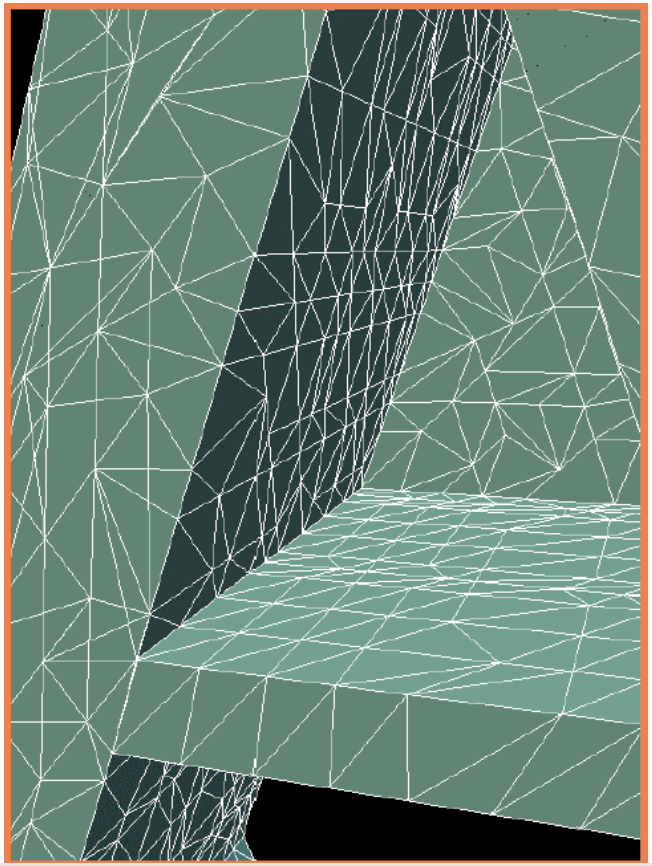




Interpolating Positions (edge collapse)

- Quadrics Error Minimization





Quadric Edge collapse

- Create a plane for each involved vertex, considering their Normals

- Place the position of the new vertex where it minimize the squared distance to the planes

- Involves solving a simple linear system

Quadric Error

Let $\mathbf{n}^T\mathbf{v} + d = 0$ be the equation representing a plane

The squared distance of a point $\boldsymbol{x}$ from the plane is

$$D(\mathbf{x}) = \mathbf{x}(\mathbf{n}\mathbf{n}^T)\mathbf{x} + 2d\mathbf{n}^T\mathbf{x} + d^2$$

This distance can be represented as a quadric

$$Q = (\mathrm{A}, \mathbf{b}, c) = (\mathbf{n}\mathbf{n}^T, d\mathbf{n}, d^2)$$

$$Q(\mathbf{x}) = \mathbf{x}\mathrm{A}\mathbf{x} + 2\mathbf{b}^T\mathbf{x} + c$$

also the sum of the distance of a point from a set of planes is still a quadric...

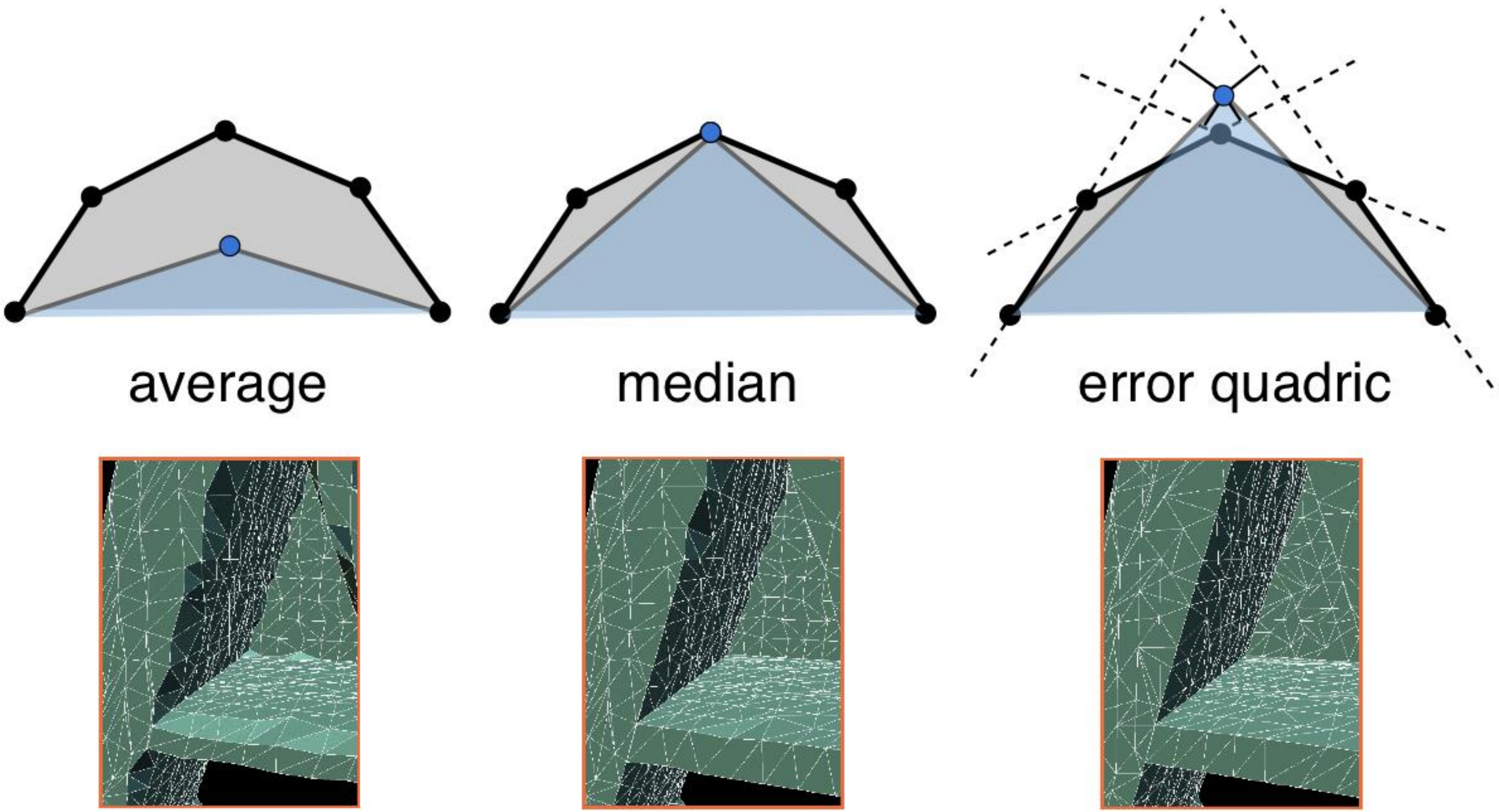
Quadric Error

The error is estimated by providing for each vertex *v* a quadric Q_V representing the sum of the all the squared distances from the faces incident in *v*

The error of collapsing an edge e=(*v*, *w*) can be evaluated as $Q_w(v)$.

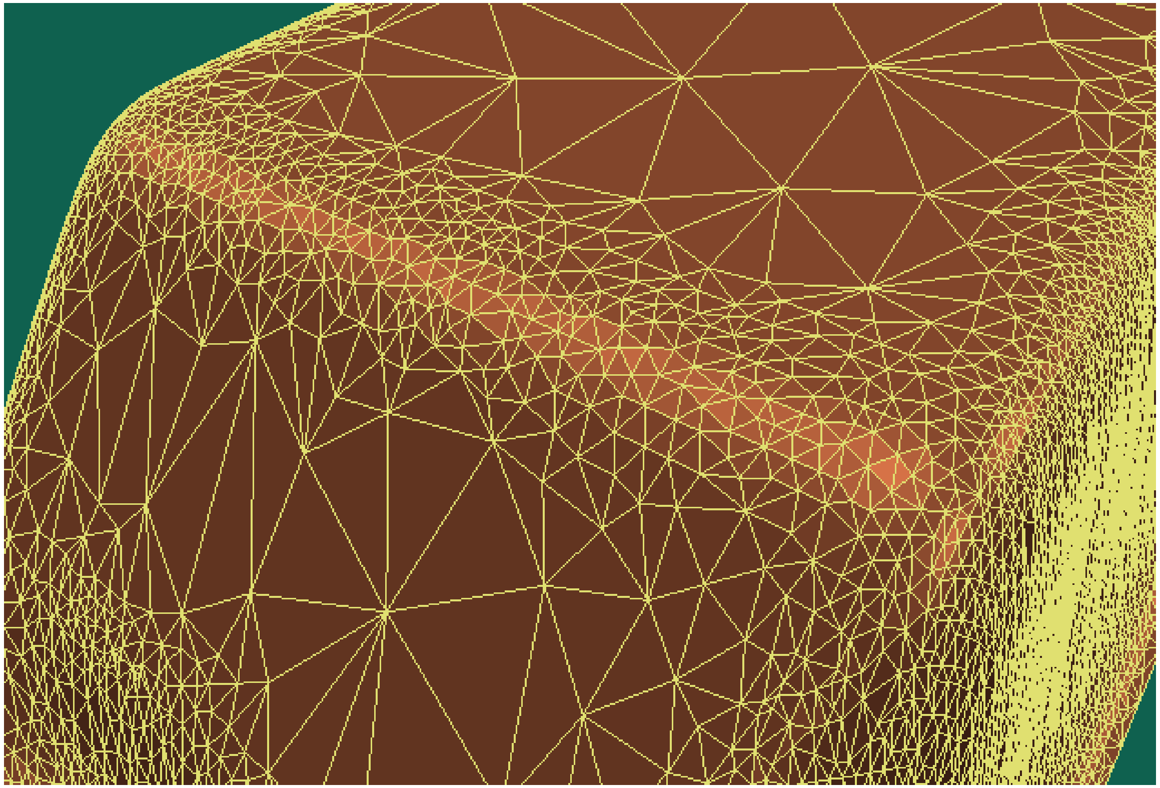
After the collapse the quadric of v is updated as follow $Q_v = Q_v + Q_w$

Quadric Edge collapse



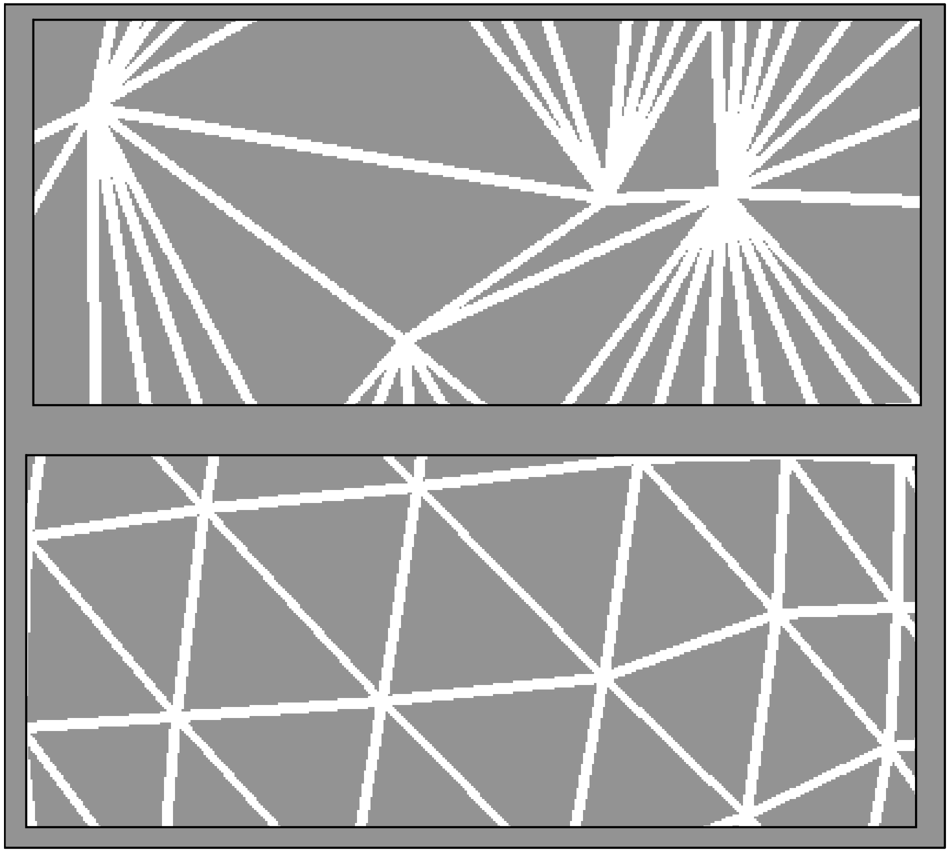
Triangle Quality

- Possibly adding an energy term that penalize bad shaped triangles

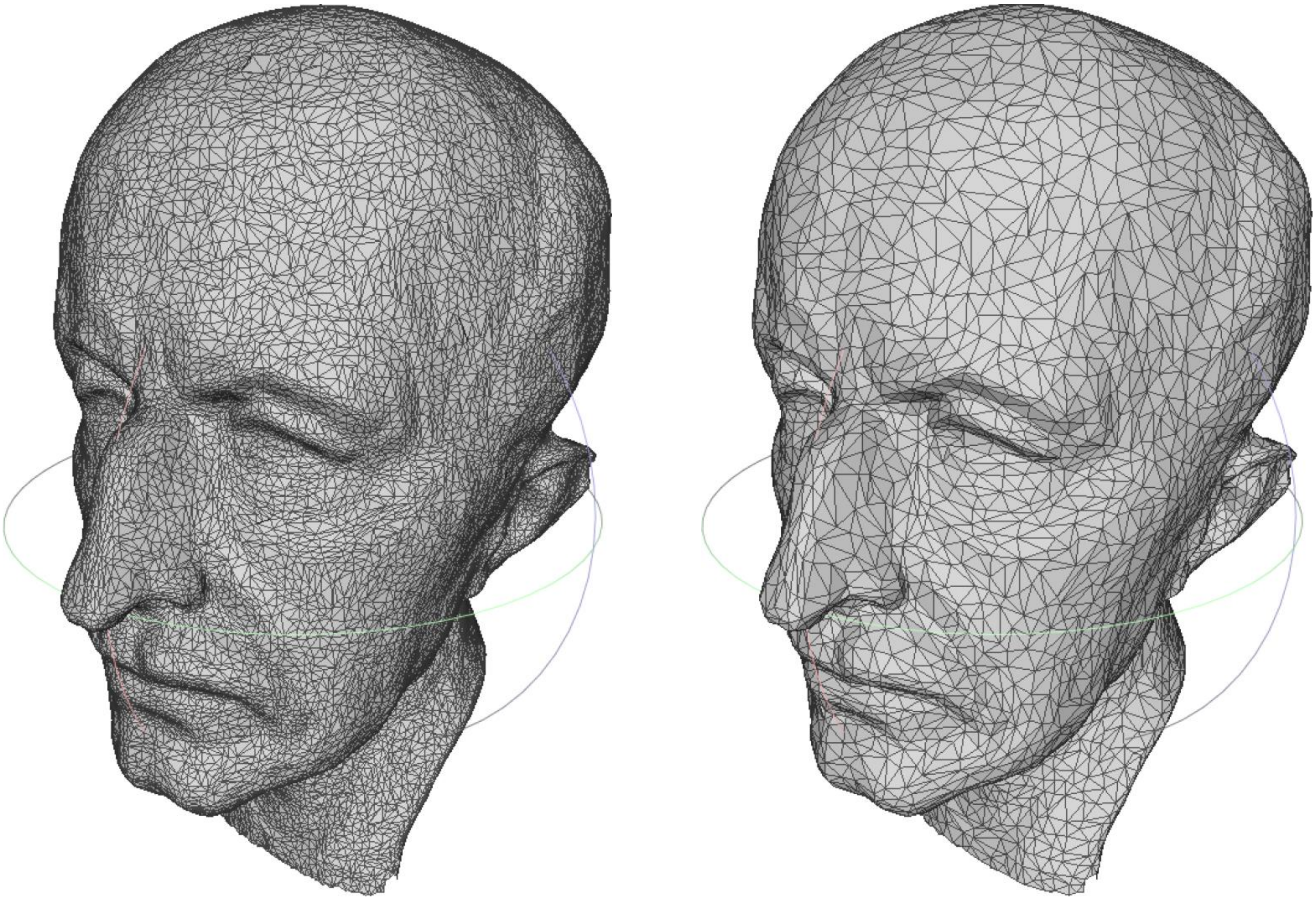


Triangle Quality

- Possibly adding an energy term that tend to balance valence



Examples : quadric edge collapse



Reduced from 50K to 12k faces

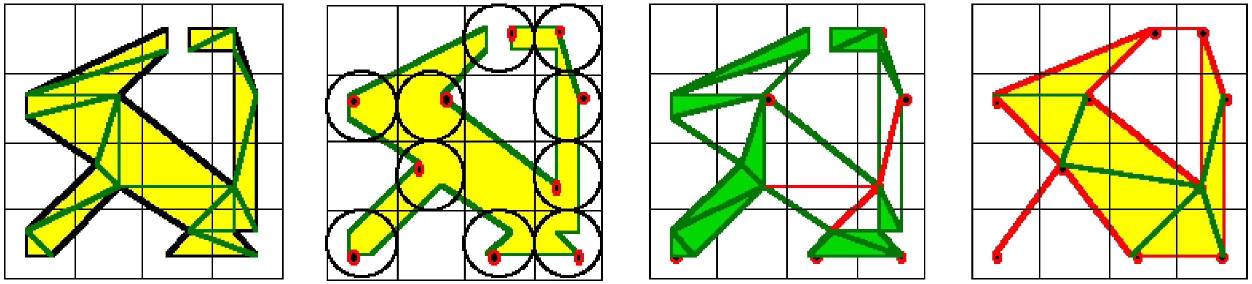
Clustering

Vertex Clustering **[Rossignac, Borrel `93]**

- detect and unify ***clusters*** of nearby vertices (discrete gridding and coordinates truncation)
- all faces with two or three vertices in a same cluster are removed

Note:

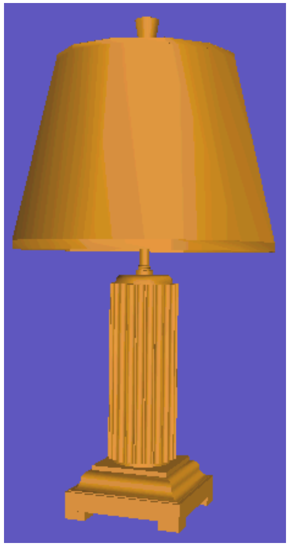
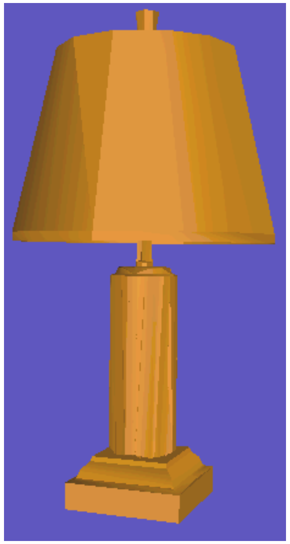
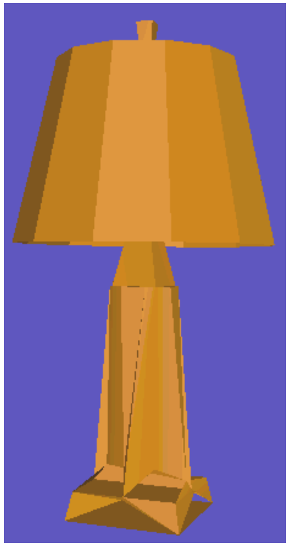
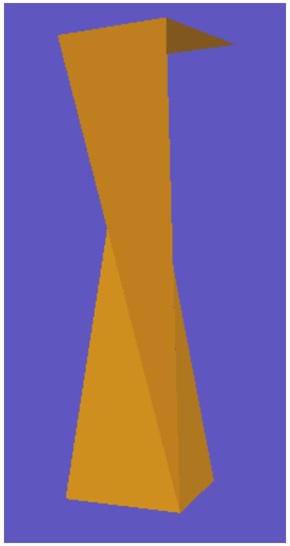
- It does not preserve topology (faces may degenerate to edges, genus may change)
- approximation depends on grid resolution (hard to predict exact final face number)



(figure by Rossignac)

Clustering -- Examples

Simplification of a table lamp, IBM 3D Interaction Accelerator

10,108 facets 1,383 facets 474 facets 46 facets