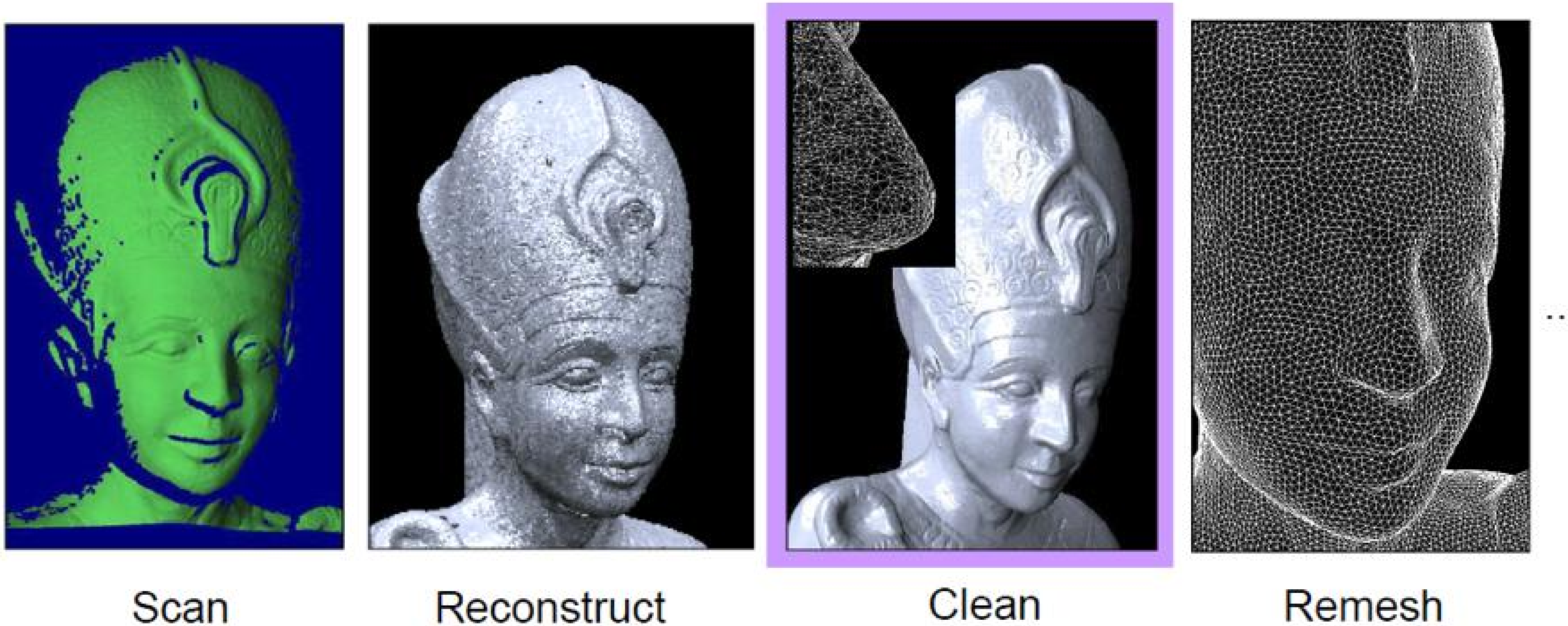
CENG 789 – Digital Geometry Processing

08- Smoothing, Remeshing, Subdivision

Prof. Dr. Yusuf Sahillioğlu

Computer Eng. Dept, MIDDLE EAST TECHNICAL UNIVERSITY, Turkey

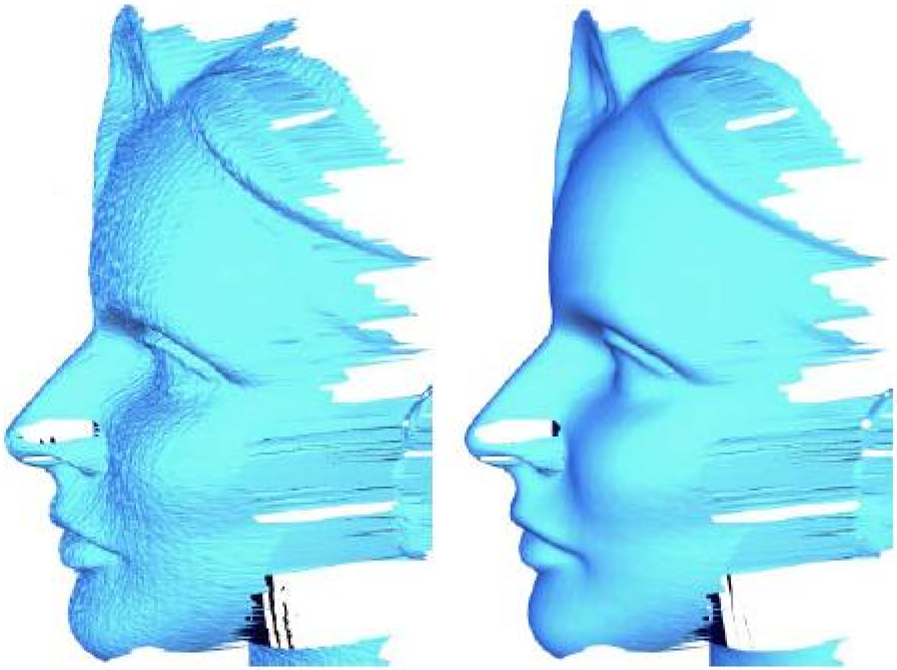
Mesh Processing Pipeline



- ✓ We will do cleaning/smoothing and remeshing today.
- ✓ Remeshing (typically) followed by mesh deformation and/or 3D printing.

Mesh smoothing

- ✓ Idea: Filter out high frequency noise (common in scanners).



- ✓ Noise causes feature vectors to be less consistent from dset to dset.
- ✓ Alignments are more difficult to compute in the presence of noise.
- ✓ Visually unattractive in games, movies, etc.

Mesh smoothing

- ✓ Solution: Uniform Laplace operator (Laplacian smoothing).

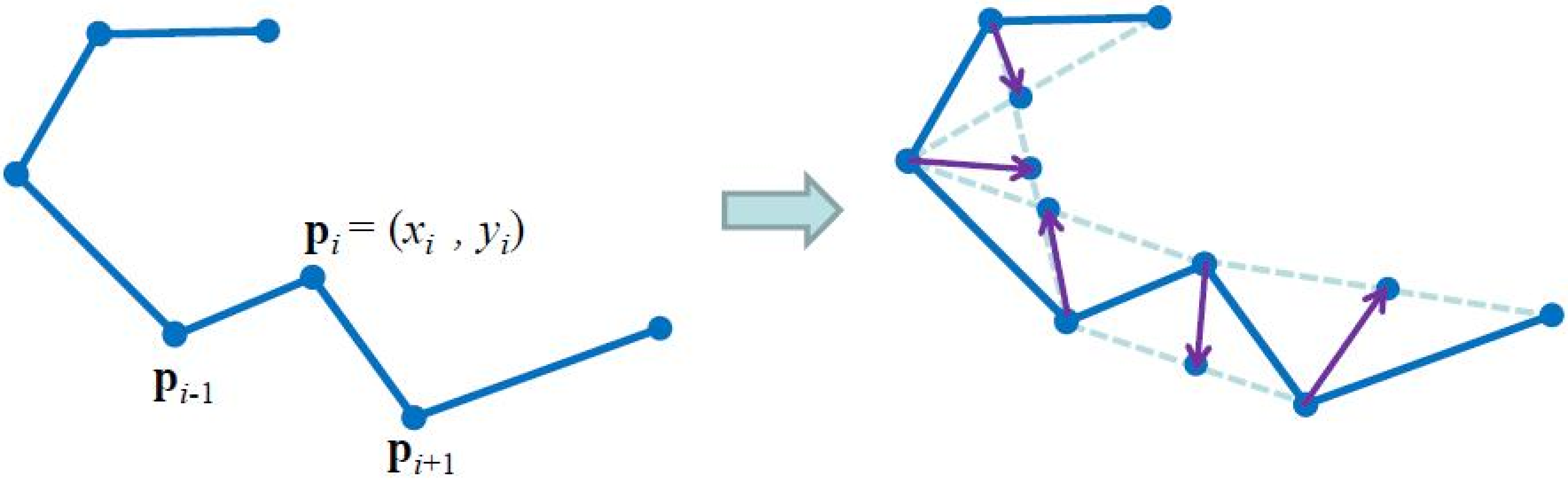
$$L_U(v) = \left(\frac{1}{n}\sum_i v_i\right) - v$$

$$v' = v + \frac{1}{2} \cdot L_U(v)$$

- ✓ Do it in parallel, i.e., use original coordinates although they might have been updated previously.

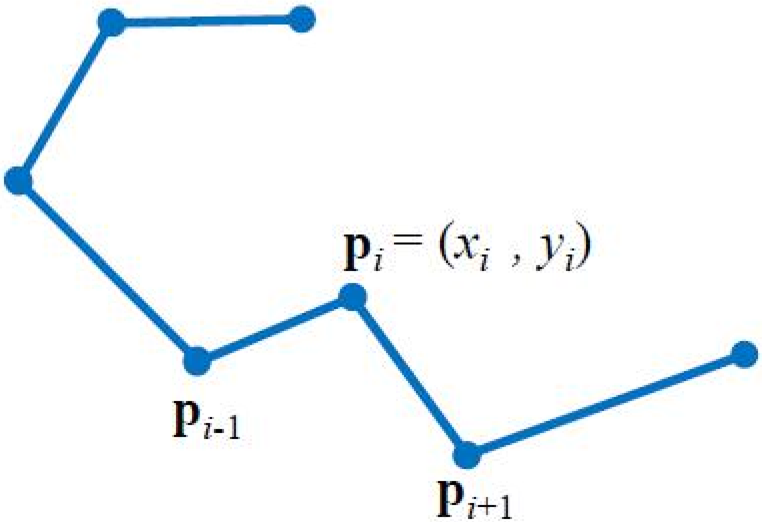
Mesh smoothing

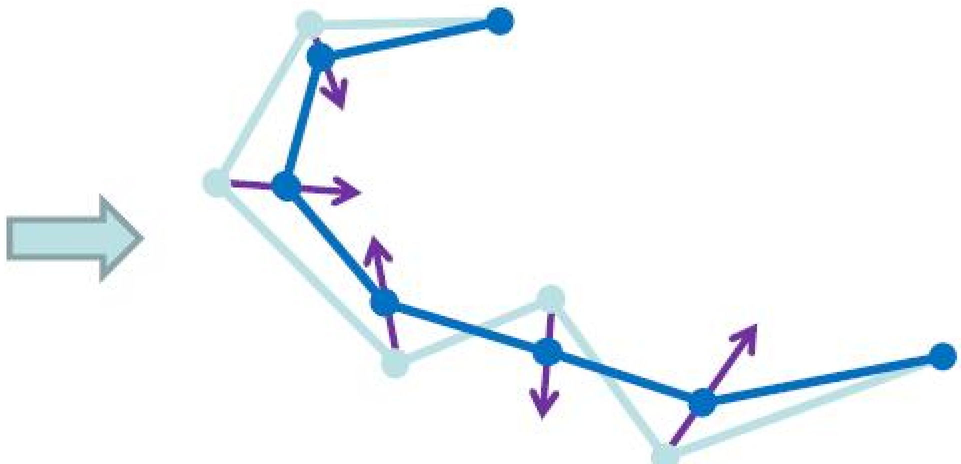
✓ Illustration in 1D:



Mesh smoothing

✓ Illustration in 1D:





$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \frac{1}{2} L(\mathbf{p}_i)$$

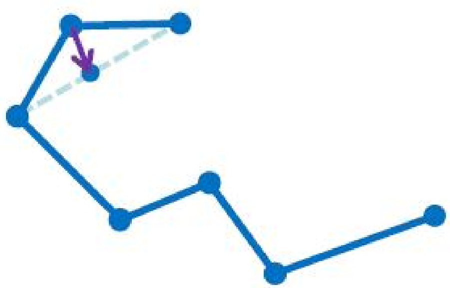
Mesh smoothing

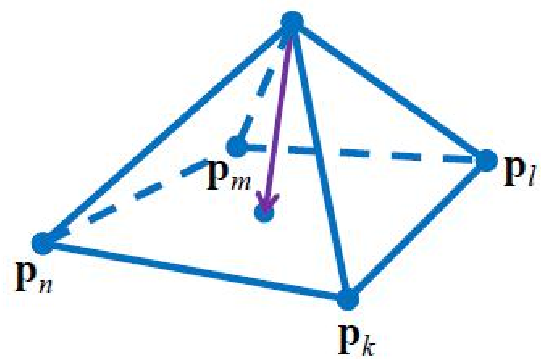
- ✓ Observation: close curve converges to a single point?

Mesh smoothing

✓ Illustration in 2D: Same as for curves (1D).

$$\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} + \lambda \Delta \mathbf{p}_i^{(t)}$$





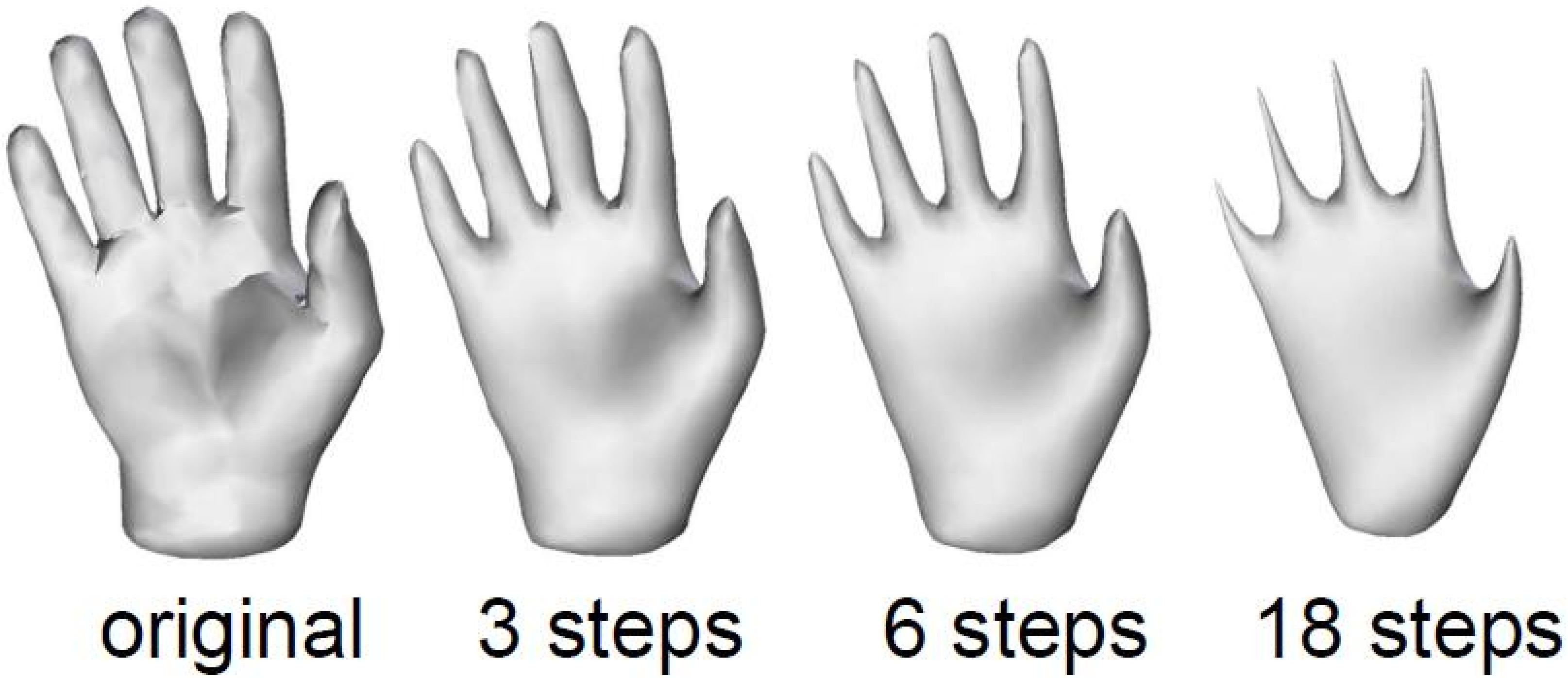
$$\Delta \mathbf{p}_i = \frac{1}{|N_i|}\left(\sum_{j \in N_i} \mathbf{p}_j\right) - \mathbf{p}_i$$

Mesh smoothing

- ✓ Observation: close mesh, e.g., sphere, converges to a single point.

Mesh smoothing

- ✓ Observation: shrinkage problem.
- ✓ Repeated iterations of Laplacian smoothing shrinks the mesh.



Mesh smoothing

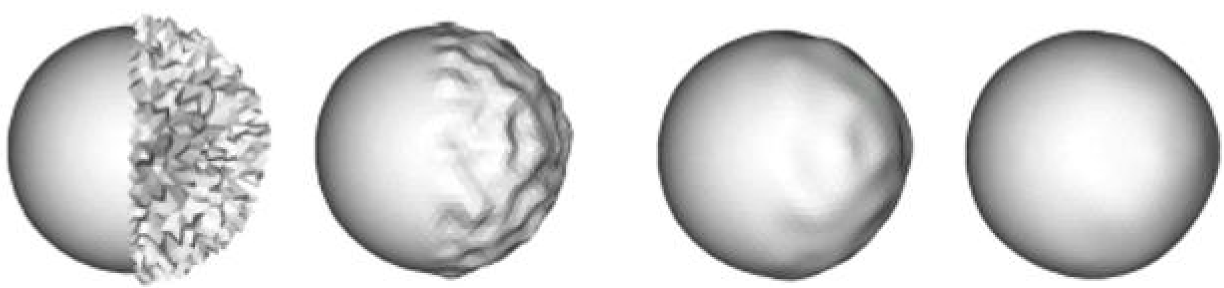
- ✓ Solution2: shrinkage problem is remedied with an inflation term.
- ✓ This is introduced by the Mesh Fairing paper by Taubin in 1995.

Iterate:

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \lambda \Delta \mathbf{p}_i \qquad \text{Shrink}$$

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \mu \Delta \mathbf{p}_i \qquad \text{Inflate}$$

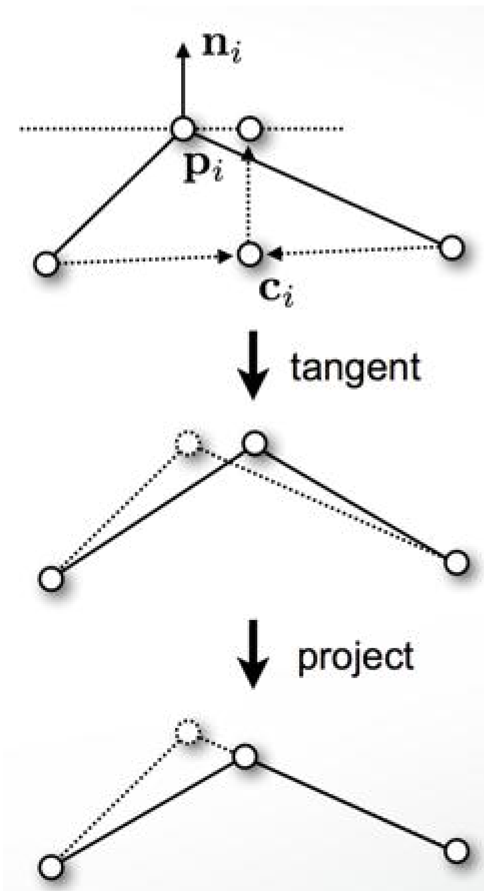
with $\lambda > 0$ and $\mu < 0$ and $|\mu| > \lambda$

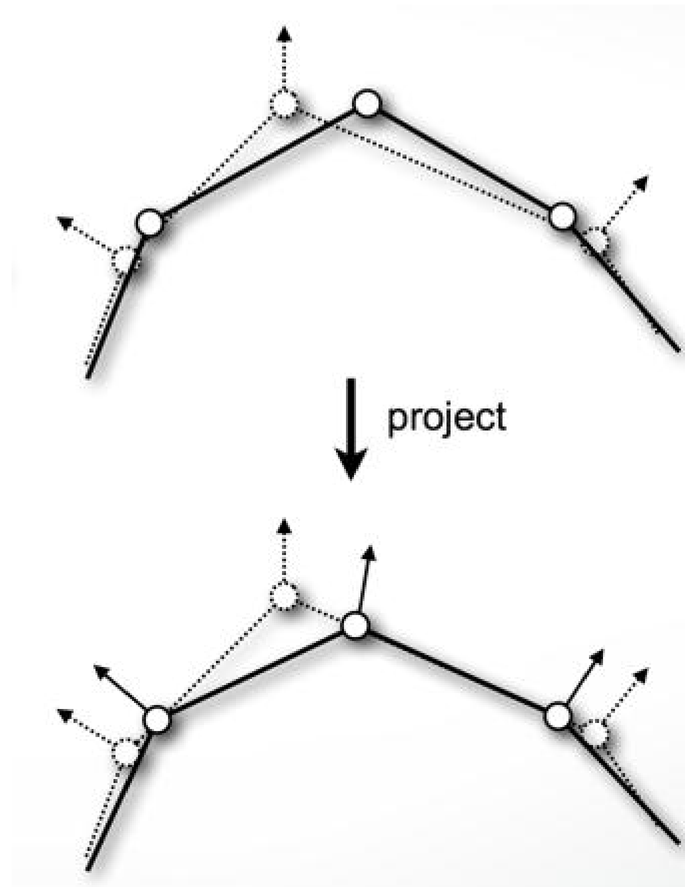


original 10 steps 50 steps 200 steps

Mesh smoothing

✓ Solution2: shrinkage problem is remedied with tangential projection.

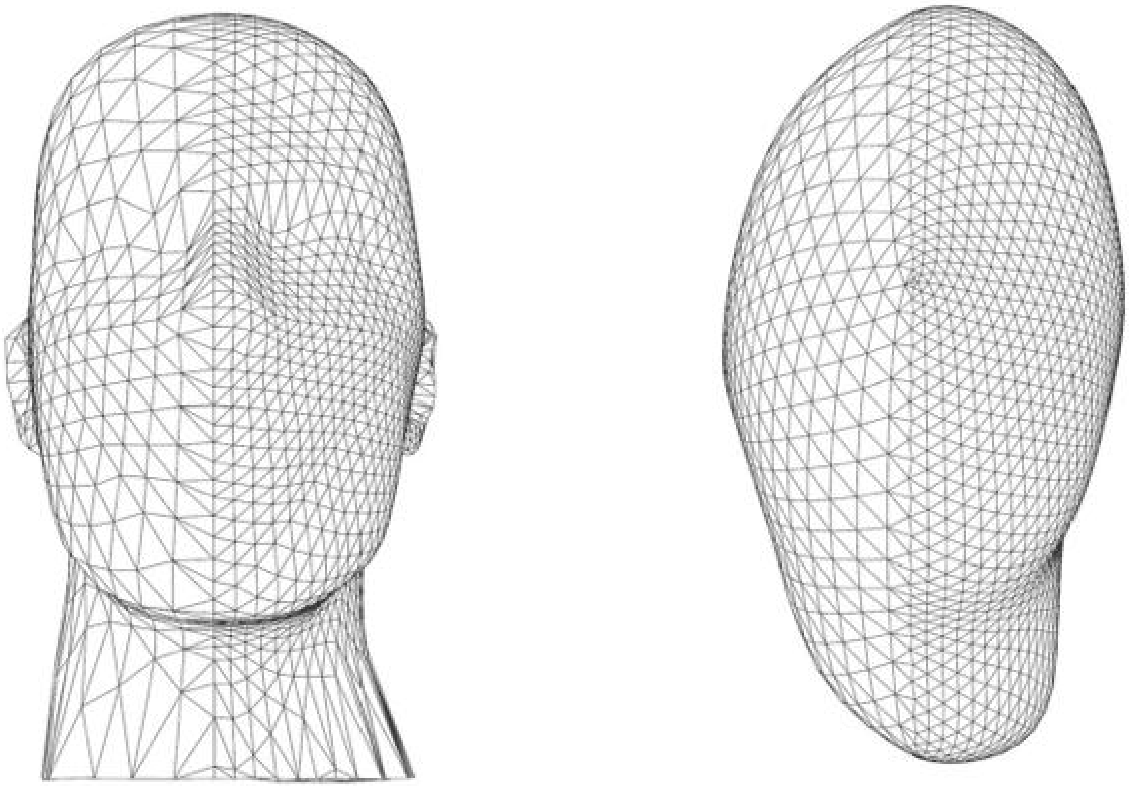




$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \lambda\left(\mathbf{I} - \mathbf{n}_i\mathbf{n}_i^T\right)(\mathbf{c}_i - \mathbf{p}_i)$$

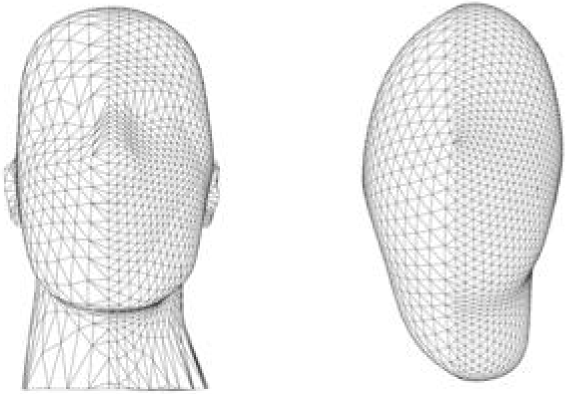
Mesh smoothing

✓ Another problem: Results based on uniform weighting depend on triangle shape.



Mesh smoothing

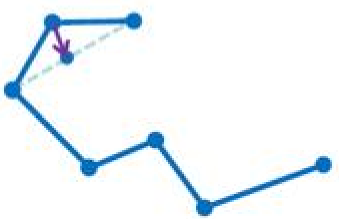
✓ Another problem: Results based on uniform weighting depend on triangle shape.



$$\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} + \lambda \Delta \mathbf{p}_i^{(t)}$$

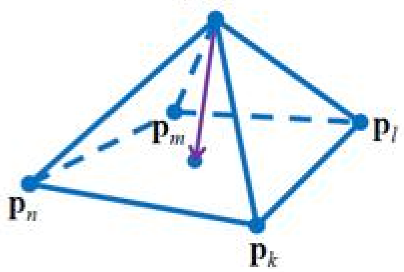
What is $\Delta \mathbf{p}_i$?

1D



$$\frac{1}{2}\left(\mathbf{p}_{i+1} + \mathbf{p}_{i-1}\right) - \mathbf{p}_i$$

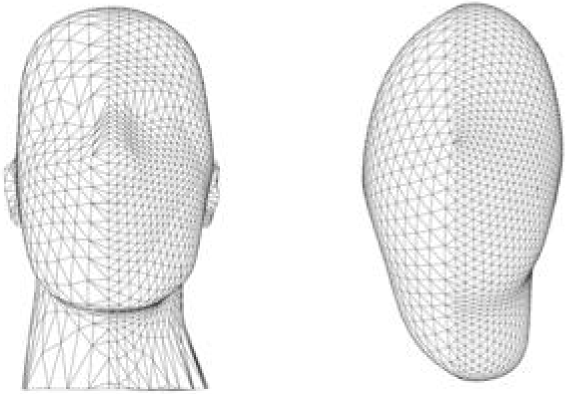
2D



$$\frac{1}{|N_i|}\left(\sum_{j \in N_i} \mathbf{p}_j\right) - \mathbf{p}_i$$

Mesh smoothing

- ✓ Another problem: Results based on uniform weighting depend on triangle shape.

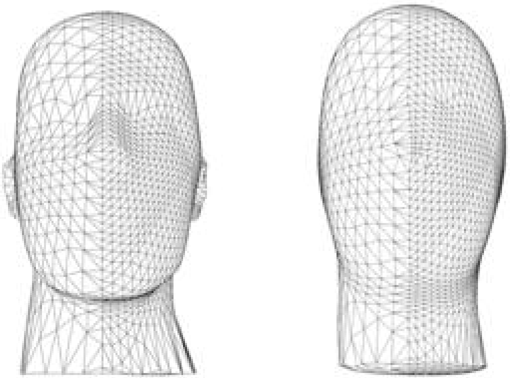


- ✓ Problem in 1D: same (uniform) weight for both neighbors although one is closer.

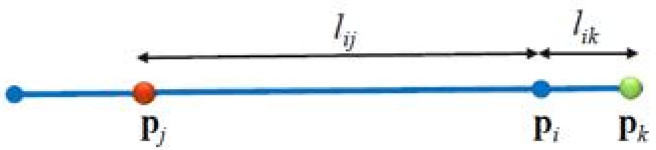
$$\boxed{\frac{1}{2}}(\mathbf{p}_{i+1}+\mathbf{p}_{i-1})-\mathbf{p}_i$$

Mesh smoothing

✓ Another problem: Results based on uniform weighting depend on triangle shape.



✓ Solution: use a smarter weight instead.

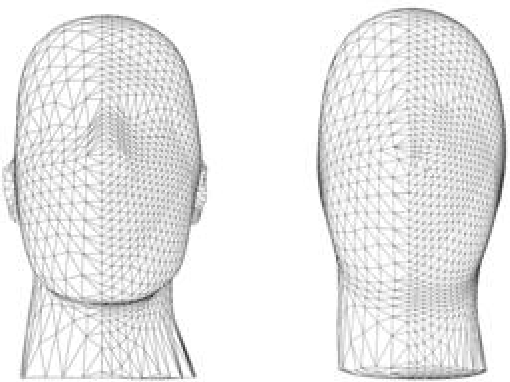


$$w_{ij} = \frac{1}{l_{ij}} \qquad w_{ik} = \frac{1}{l_{ik}} \qquad L(\mathbf{p}_i) = \frac{w_{ij}\mathbf{p}_j + w_{ik}\mathbf{p}_k}{w_{ij} + w_{ik}} - \mathbf{p}_i$$

✓ Straight 1D curves will be invariant to smoothing with this inverse distance weighting, i.e., $L(\mathbf{p}_i) = 0$.

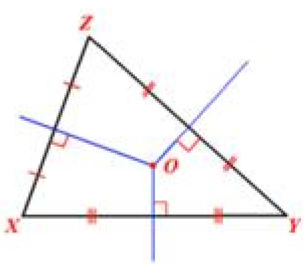
Mesh smoothing

- ✓ Another problem: Results based on uniform weighting depend on triangle shape.

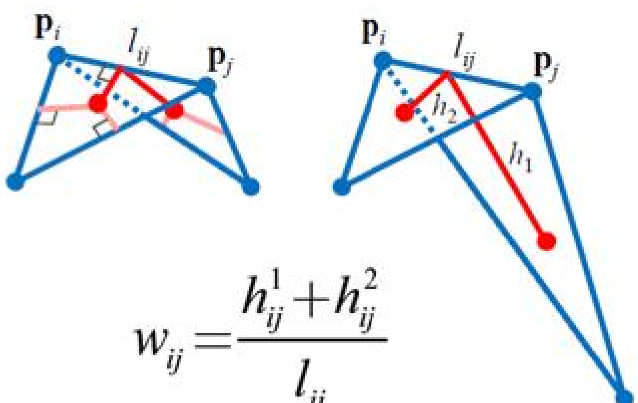


- ✓ Solution: use a smarter weight instead.
- ✓ Same idea extends to 2D triangular surfaces using cotangent weights.

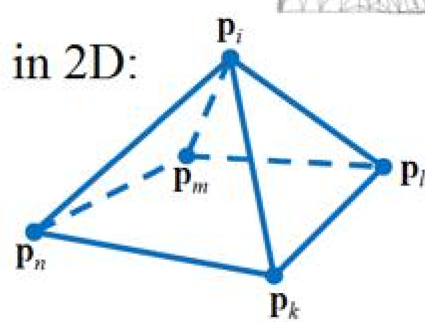
$w_{ij} = \frac{1}{l_{ij}}$ in 1D is replaced as follows in 2D:



Here, O is the circumcenter



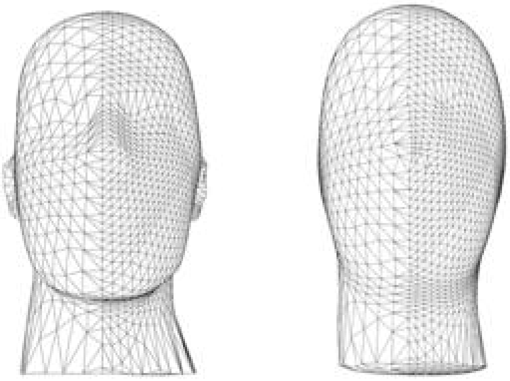
$$w_{ij} = \frac{h_{ij}^1 + h_{ij}^2}{l_{ij}}$$



$$L(\mathbf{p}_i) = \frac{1}{\sum_{j \in N_i} w_{ij}} \left(\sum_{j \in N_i} w_{ij} \mathbf{p}_j \right) - \mathbf{p}_i$$

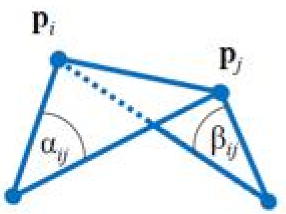
Mesh smoothing

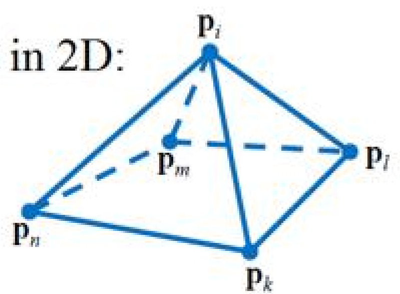
- ✓ Another problem: Results based on uniform weighting depend on triangle shape.



- ✓ Solution: use a smarter weight instead.
- ✓ Same idea extends to 2D triangular surfaces using cotangent weights.

$w_{ij} = \dfrac{1}{l_{ij}}$ in 1D is replaced as follows in 2D:





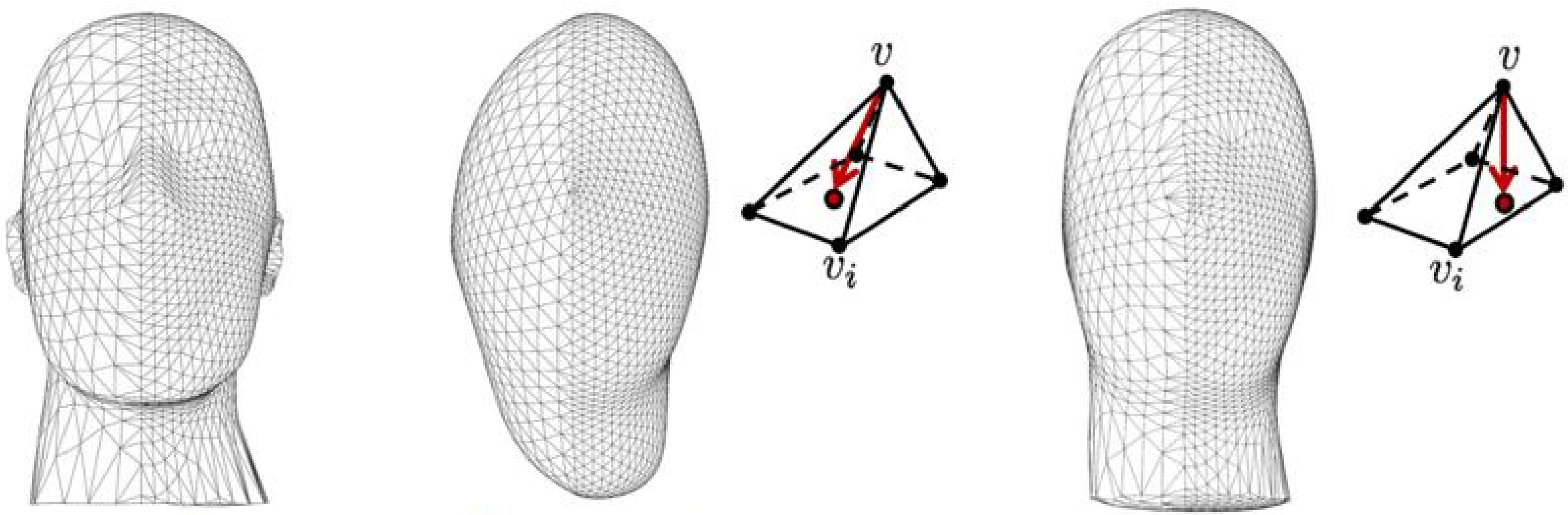
$$w_{ij} = \frac{h_{ij}^1 + h_{ij}^2}{l_{ij}} = \frac{1}{2}\left(\cot\alpha_{ij} + \cot\beta_{ij}\right)$$

$$L(\mathbf{p}_i) = \frac{1}{\sum_{j\in N_i} w_{ij}}\left(\sum_{j\in N_i} w_{ij}\mathbf{p}_j\right) - \mathbf{p}_i$$

Planar meshes will be invariant to smoothing ☺.

Mesh smoothing

- ✓ Another problem: Results based on uniform weighting depend on triangle shape.
- ✓ Solution: use a geometry-aware weight instead; cotan weights in 2D.



original

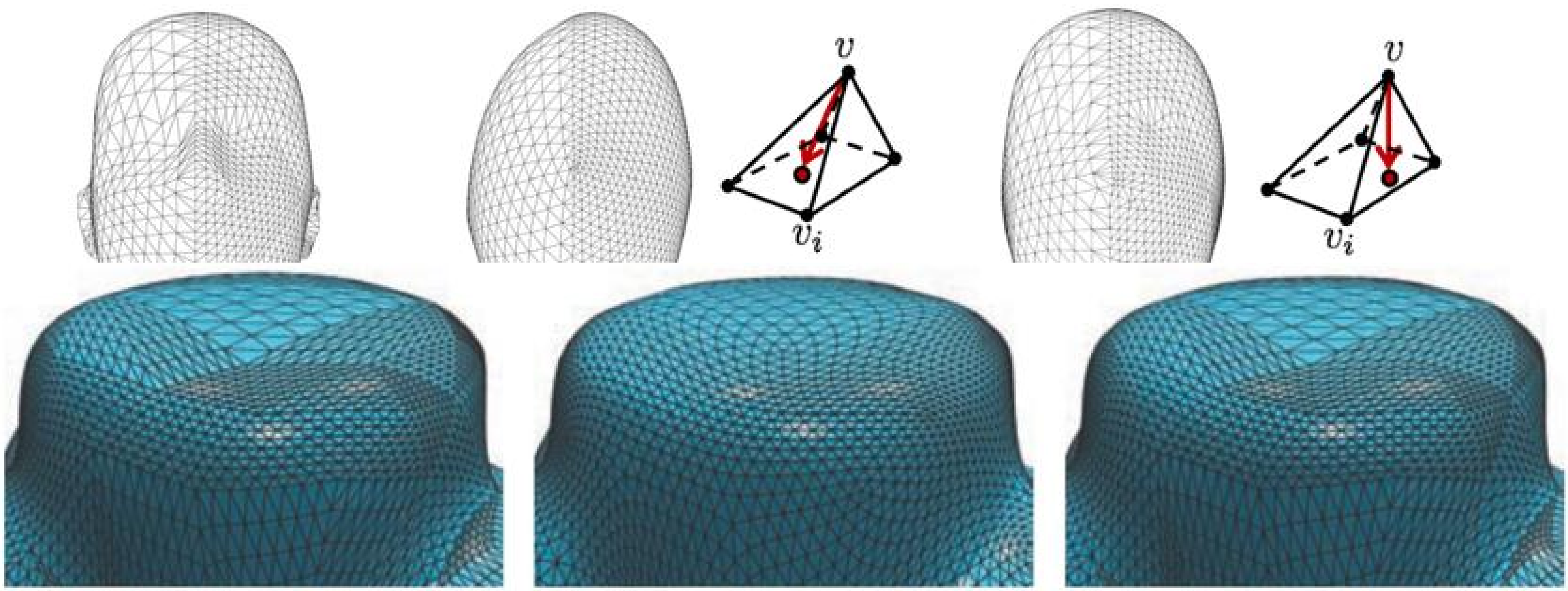
Uniform weights
pnts always in centroids
(original geometry ruined)

Cotangent weights
geometry-aware

Mesh smoothing

- ✓ Another problem: Results based on uniform weighting depend on triangle shape.

- ✓ Solution: use a geometry-aware weight instead; cotan weights in 2D.



Remeshing

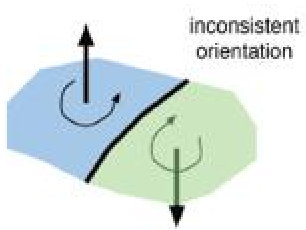
- ✓ Given a 3D mesh, compute another mesh, whose elements satisfy some quality requirements, while approximating the input acceptably.

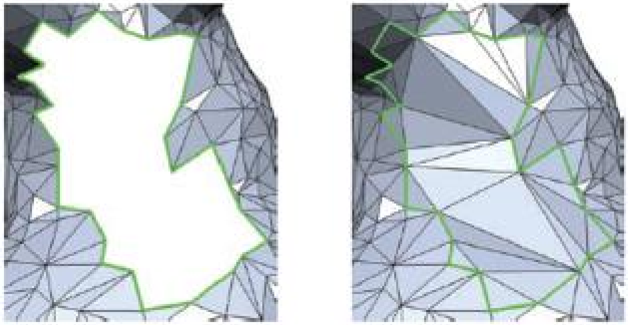
- ✓ In short, mesh quality improvement, aka Isotropic Remeshing.

- ✓ Mesh quality: sampling density, regularity, and shape of mesh elements.

- ✓ In contrast to Mesh Repairing*, the input of remeshing algorithms is usually assumed to be a manifold mesh in consistent orientation.

*

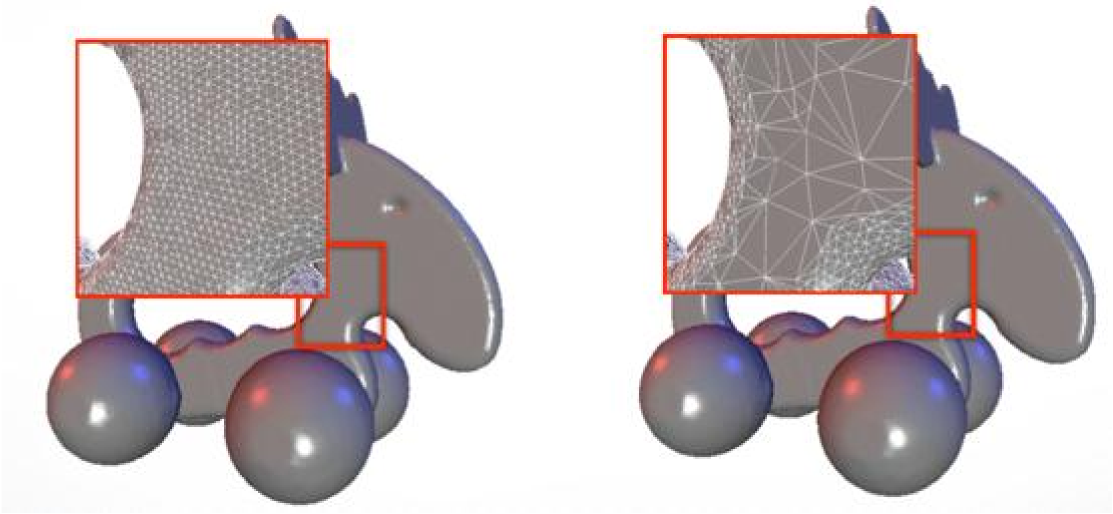




See 3D Printing Slides 70-74.

Remeshing

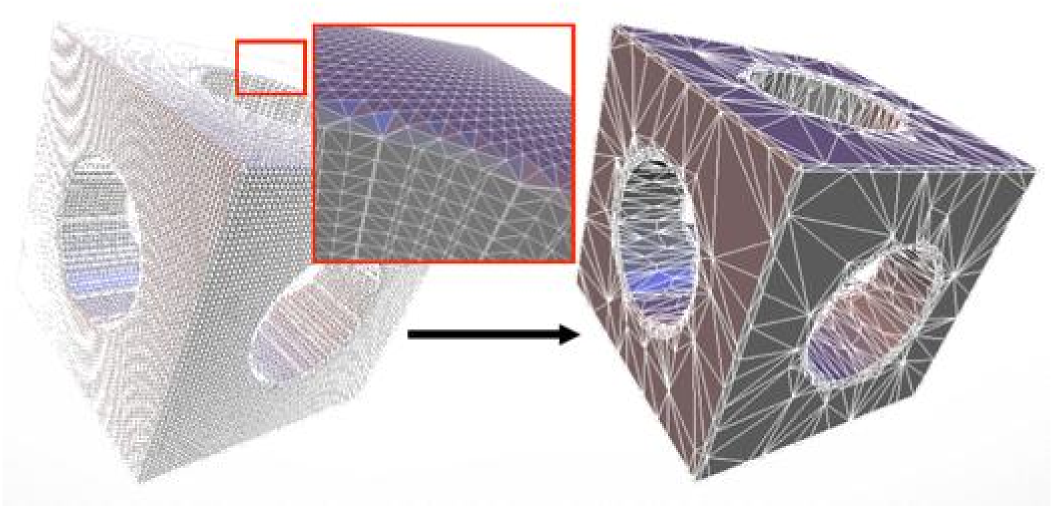
- ✓ Need remeshing to process acquisition hardware raw data for memory consumption reduction, computational efficiency and accuracy.
- ✓ Remesh globally (from scratch).



- ✓ 3D scanners uniformly oversample to capture every possible detail without any a priori knowledge of the surface content (left).

Remeshing

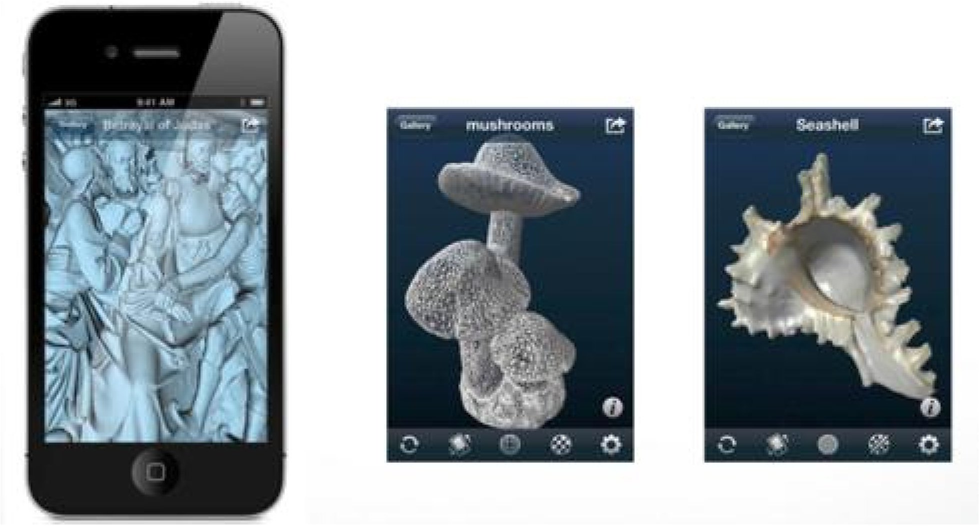
- ✓ Need remeshing to process acquisition hardware raw data for memory consumption reduction, computational efficiency and accuracy.
- ✓ Remesh globally (from scratch).



- ✓ Surface reconstruction algorithms, e.g., Marching Cubes isosurface extraction, over-tessellates to catch every possible detail (left).

Remeshing

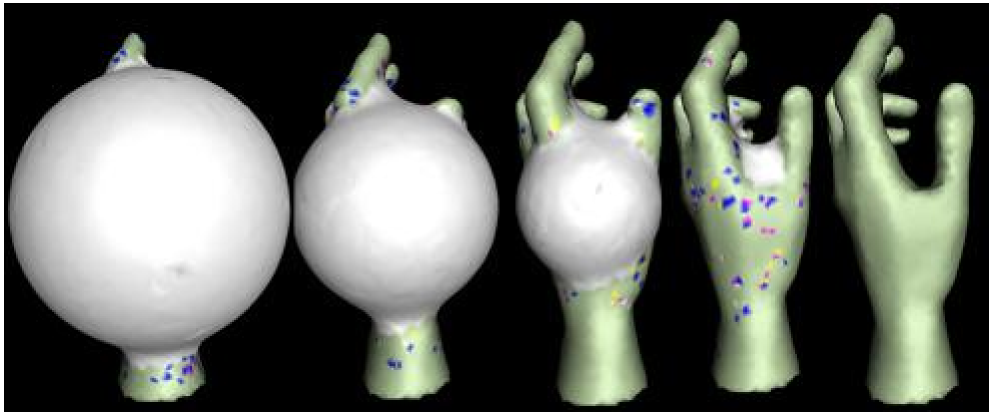
- ✓ Need remeshing to process acquisition hardware raw data for memory consumption reduction, computational efficiency and accuracy.
- ✓ Remesh globally (from scratch).

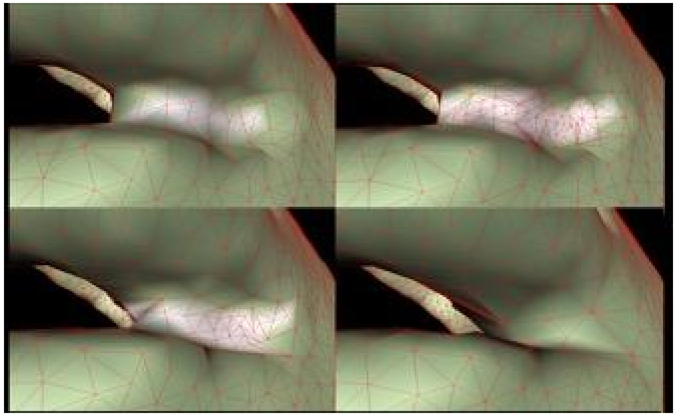


- ✓ 3D content everywhere, including new hardware that are not as powerful and resourceful as the standard desktops/workstations.

Remeshing

- ✓ Need remeshing 'cos badly-shaped elements (triangle, tetrahedron) can endanger the numerical stability of simulations (plasticity, elasticity, fracture/breaking, tearing, etc.). Such a sim couldn't run very far.
- ✓ Remesh locally (only the trouble-maker elements).

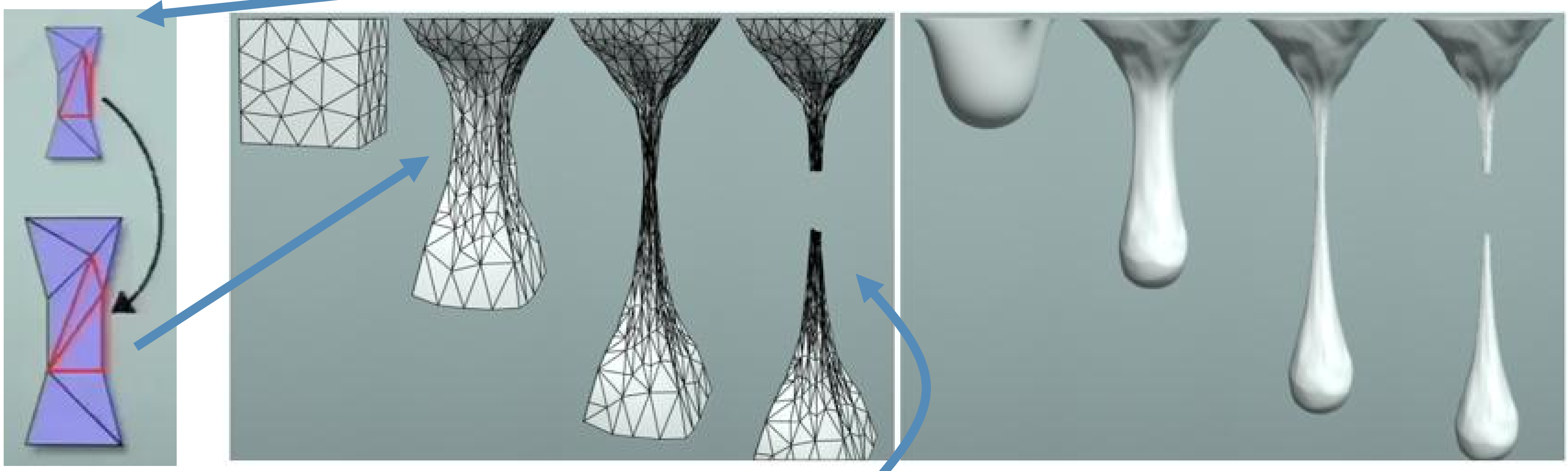




- ✓ Frozen neighborhood stops the OUT points early; remesh to give them enough power for further penetration → mesh of adequate resolution.
- ✓ Split edge if midpoint OUT: Shape from silhouette using topology-adaptive mesh deformation.

Remeshing

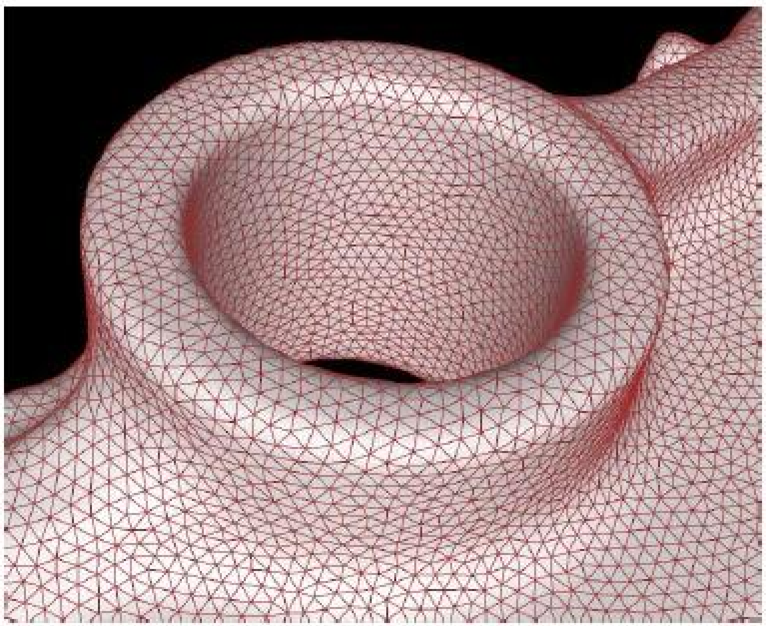
- ✓ Need remeshing 'cos badly-shaped elements (triangle, tetrahedron) can endanger the numerical stability of simulations (plasticity, elasticity, fracture/breaking, tearing, etc.). Such a sim couldn't run very far.
- ✓ Remesh locally (only the badly-shaped elements).

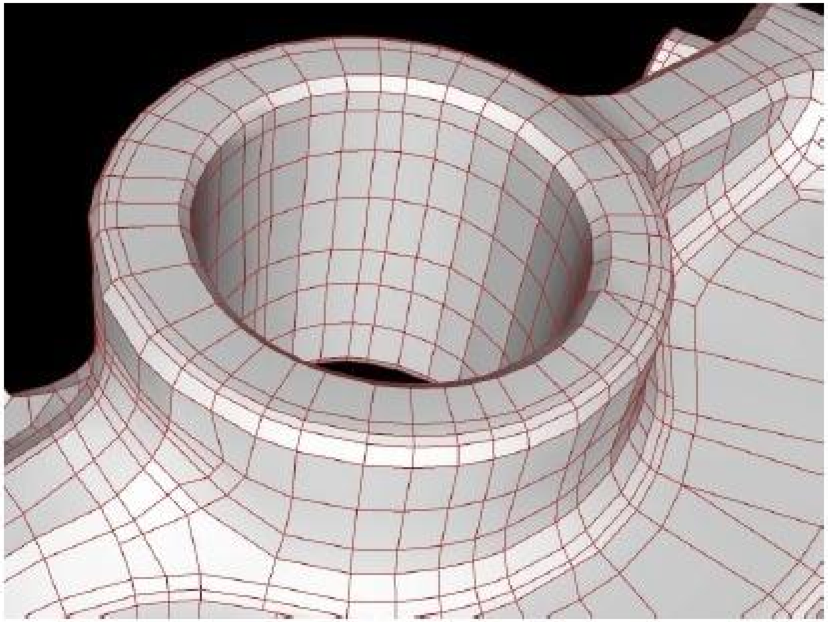


- ✓ A single mesh cannot represent all states of the obj being simulated.
- ✓ Commit if element improves: greedy, fast, dynamic remeshing, sim-friendly.

Remeshing

✓ Mesh elements: triangle vs. quadrangle.

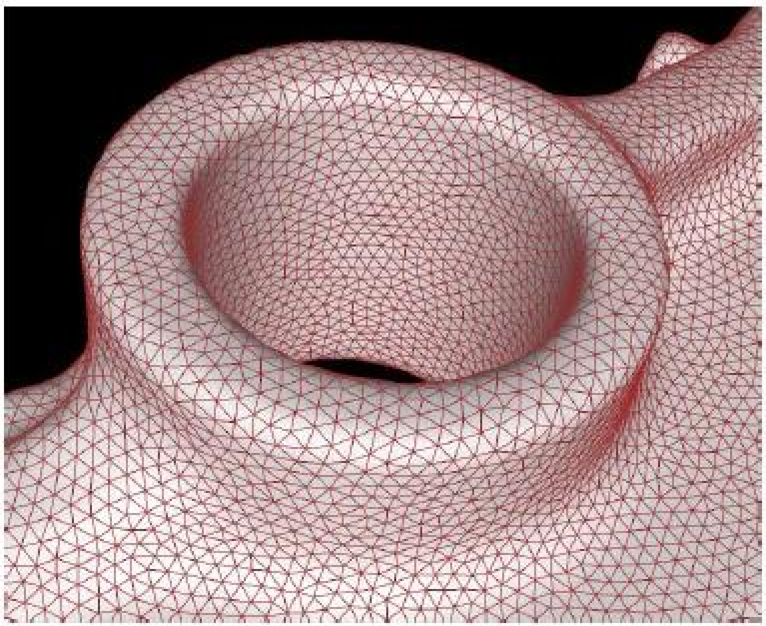


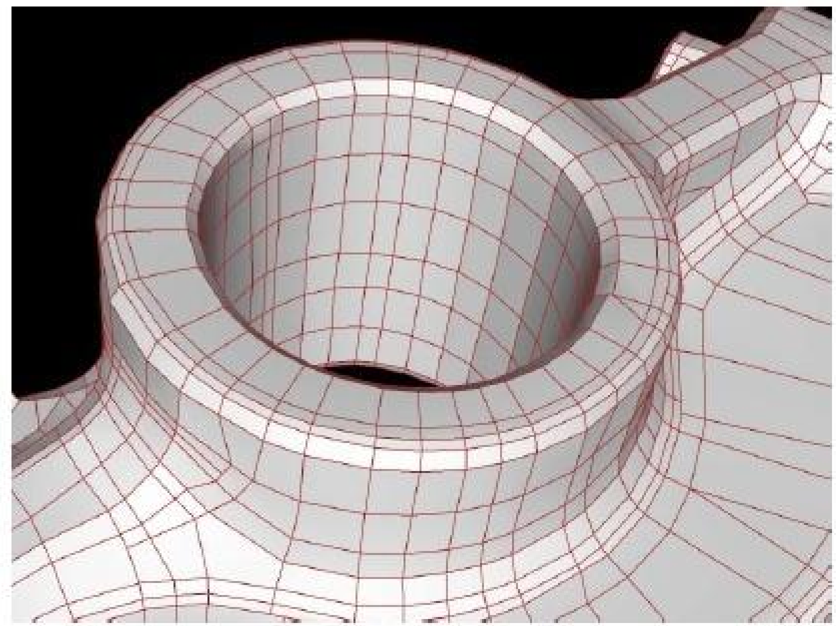


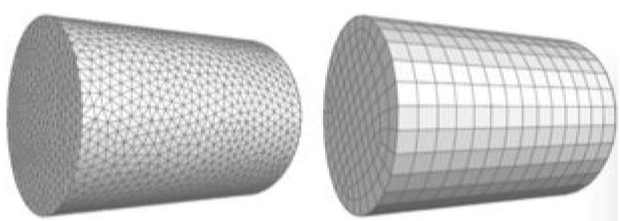
+ simplest primitive.
+ heavily-optimized algorithms.
+ specialized hardware.

Remeshing

✓ Mesh elements: triangle vs. quadrangle.







+ aligns to principal curvature dirs.
+ sub-d friendly.
+ anisotropy w/o bad angles.

Remeshing

- ✓ Mesh elements: triangle vs. quadrangle.
- ✓ Advantages of triangle meshes (more popular than quads).
  - ✓ Four points or more may not be on the same plane, but three points always are (unambiguous unless colinear). This has the interesting property that scalar values vary linearly over the surface of the triangle.
  - ✓ This, in turn, means a lot for shading, texture mapping. A simple (barycentric) lerp, which can be done extremely fast in specialized hardware, is sufficient.
  - ✓ Triangles are the simplest* primitive, so algorithms dealing with triangles can be heavily optimized, e.g., fast point-in-triangle test.
  - ✓ * every object can be split into triangles but a triangle cannot be split into anything other than triangles.

Remeshing

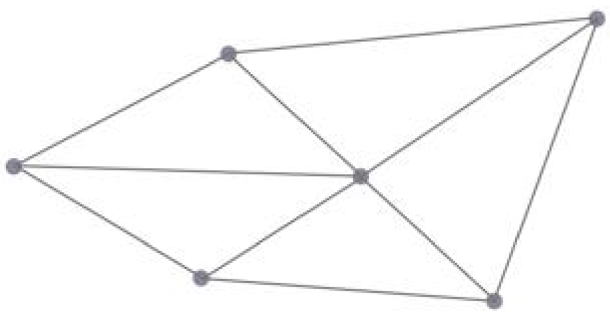
- ✓ Mesh elements: triangle vs. quadrangle.
- ✓ Advantages of quad meshes.
  - ✓ For surface PDE problems, such as fluid and cloth simulations, quad mesh is a natural representation that simplifies the formulation of the problem.
  - ✓ Catmull-Clark subdivision surfaces work with quad meshes.
  - ✓ NURBS models require a quadrangular base domain.
  - ✓ Anisotropy without bad angles: 
  - ✓ Bilinear interpolation instead of barycentric interpolation //not an advantage.

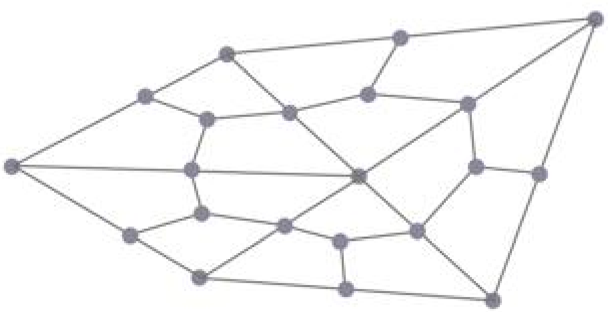
Remeshing

- ✓ Mesh elements: triangle vs. quadrangle.
- ✓ Quad to tri conversion?
- ✓ Tri to quad conversion?

Remeshing

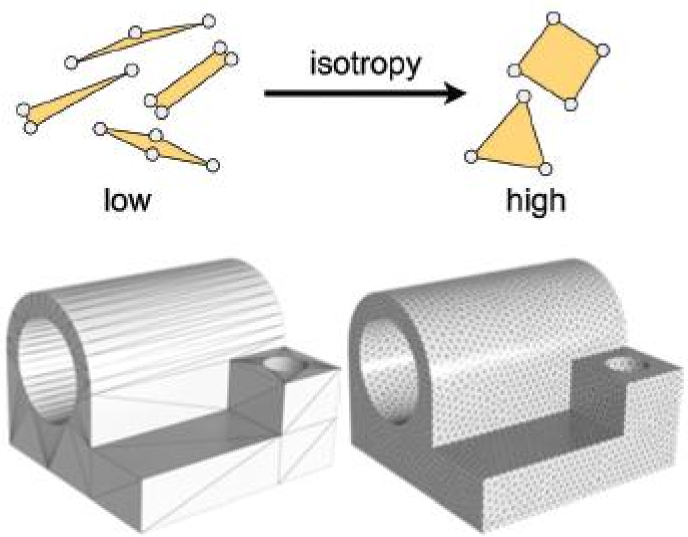
- ✓ Mesh elements: triangle vs. quadrangle.
- ✓ Quad to tri conversion? Connect the disconnected within each quad.
- ✓ Tri to quad conversion? Catmull-Clark split.

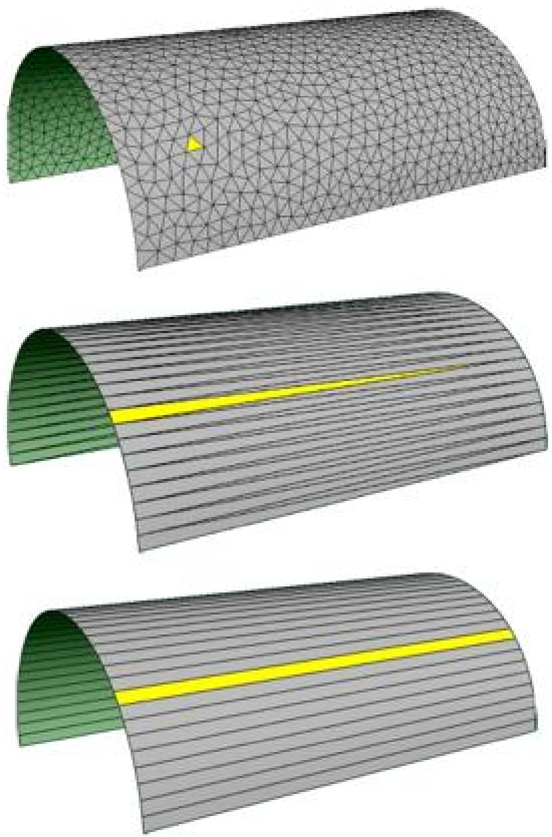




Remeshing

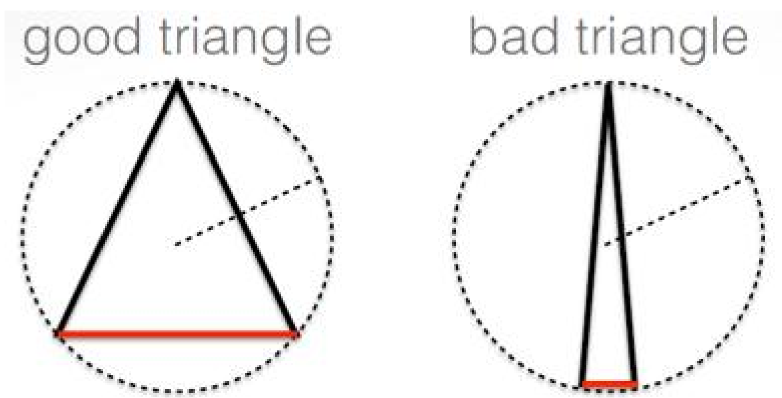
- ✓ Mesh quality criterion, shape: isotropic vs. anisotropic.
- ✓ The shape of isotropic elements is locally uniform in all directions. Ideally, a triangle/quadrangle is isotropic if it is close to equilateral/square.



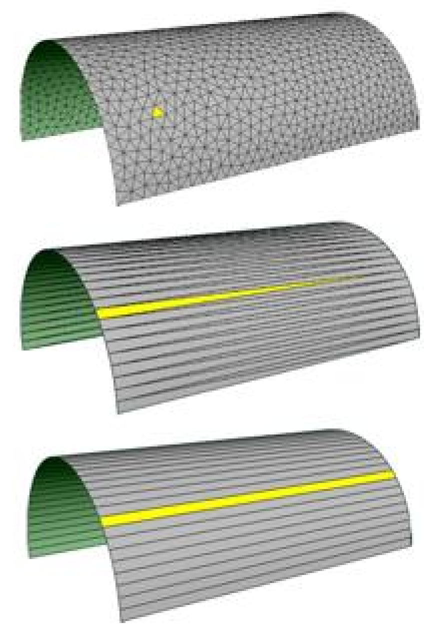


Remeshing

- ✓ Mesh quality criterion, shape: isotropic vs. anisotropic.
- ✓ The shape of isotropic elements is locally uniform in all directions. Ideally, a triangle/quadrangle is isotropic if it is close to equilateral/square.
- ✓ Isotropic elements: roundness ~ 1. (favored in numerical apps, FEM).
- ✓ Roundness: ratio of <u>circumcircle radius</u> to the length of the shortest edge.

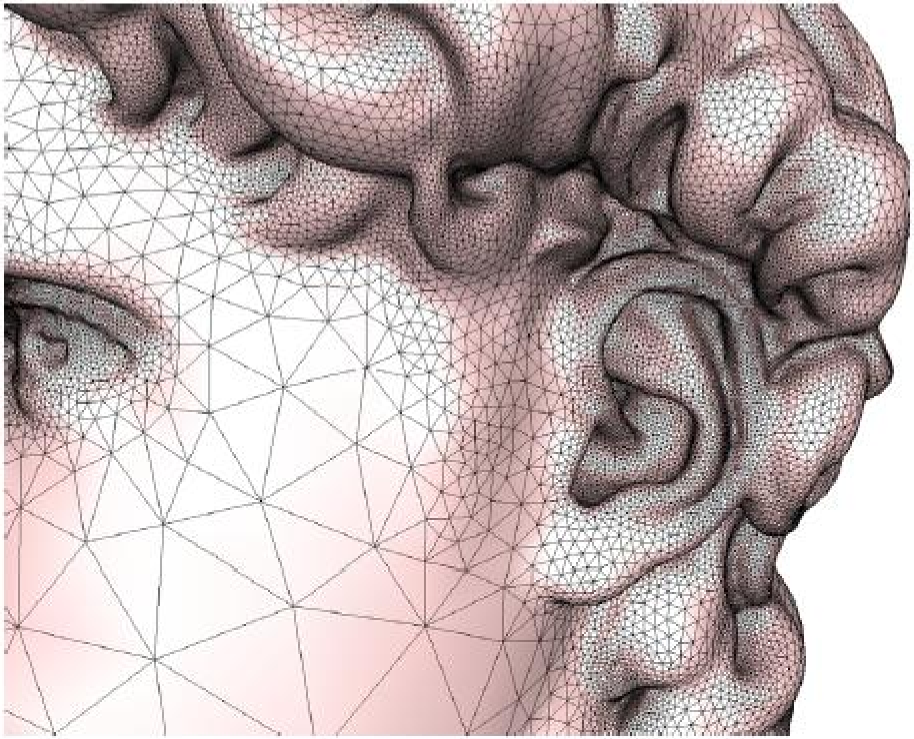


$$A = \frac{|a|\cdot|b|\cdot|c|}{4\cdot r} = \frac{|a\times b|}{2}$$



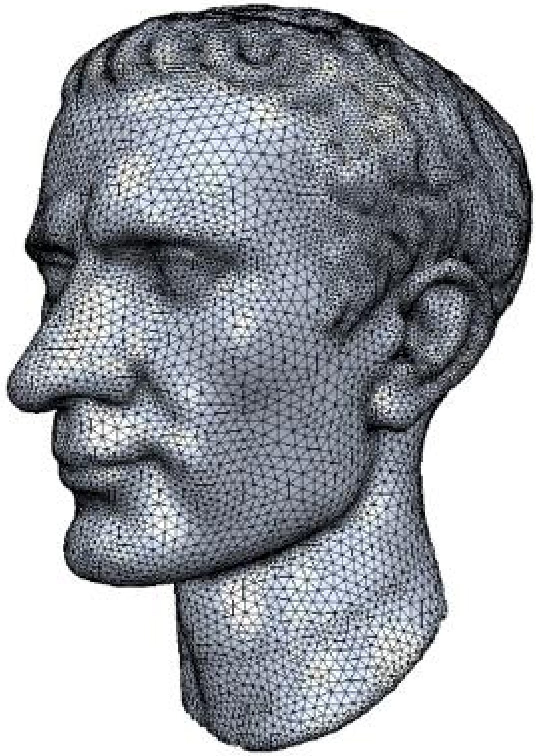
Remeshing

- ✓ Mesh quality criterion, sampling: uniform vs. adaptive.
- ✓ Smaller elements are assigned to areas w/ high curvature.



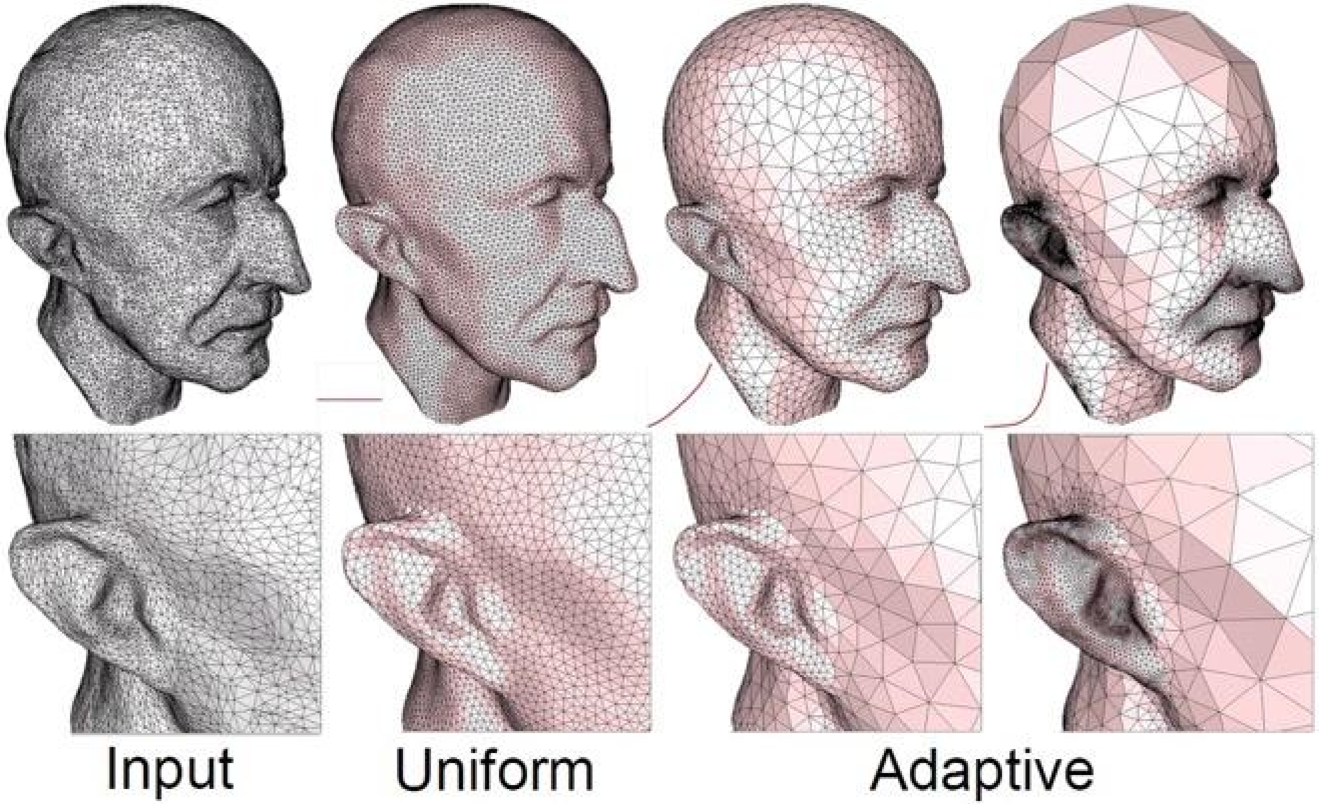
Remeshing

- ✓ Mesh quality criterion, sampling: uniform vs. adaptive.
- ✓ Smaller elements are assigned to areas w/ high curvature.



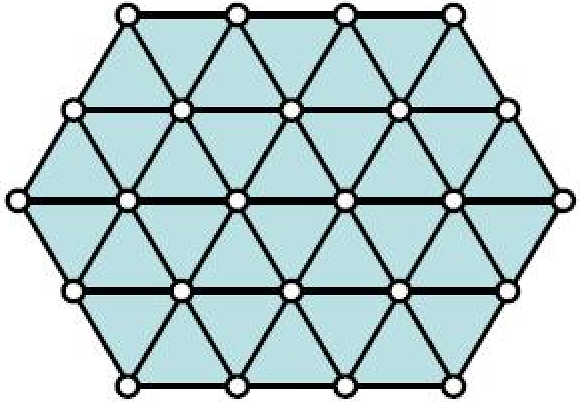
Remeshing

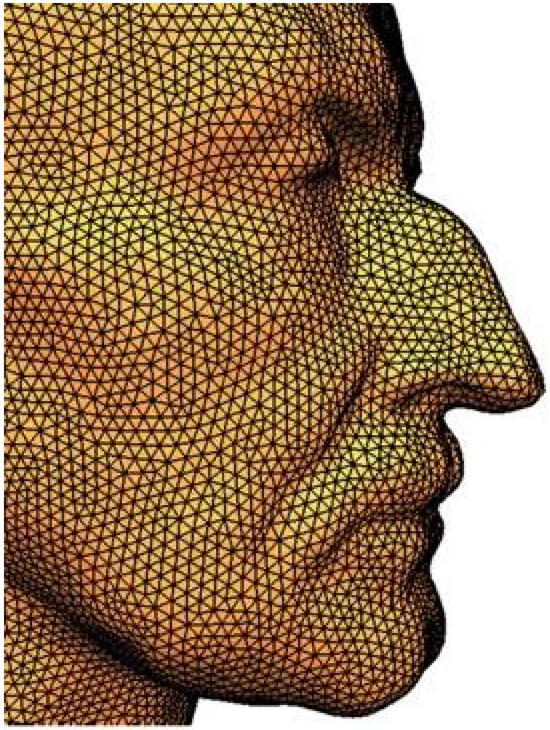
- ✓ Mesh quality criterion, sampling: uniform vs. adaptive.
- ✓ Smaller elements are assigned to areas w/ high curvature.



Remeshing

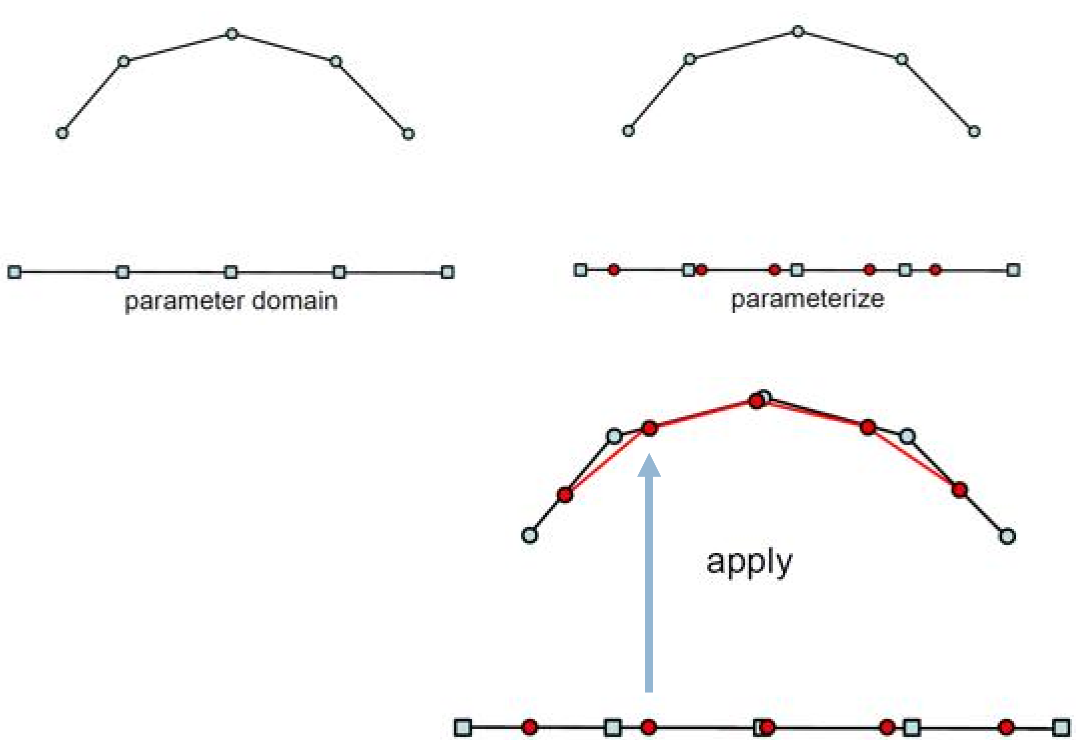
- ✓ Mesh quality criterion, regularity: irregular vs. regular.
- ✓ Valence close to 6; ~equal edge lengths.





Remeshing

- ✓ Remeshing approaches: parameterization-based vs. surface-based.
  (variational) (incremental)
- ✓ Param-based: map to 2D, do the remeshing (2D problem), lift it up.

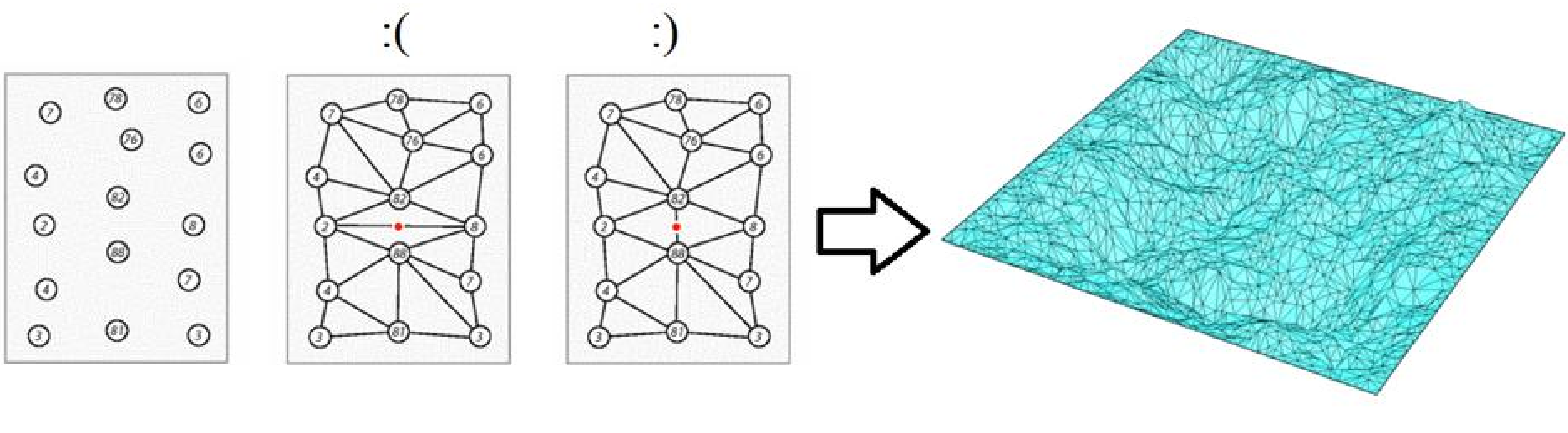


reds distributed better.

← reds lifted up.

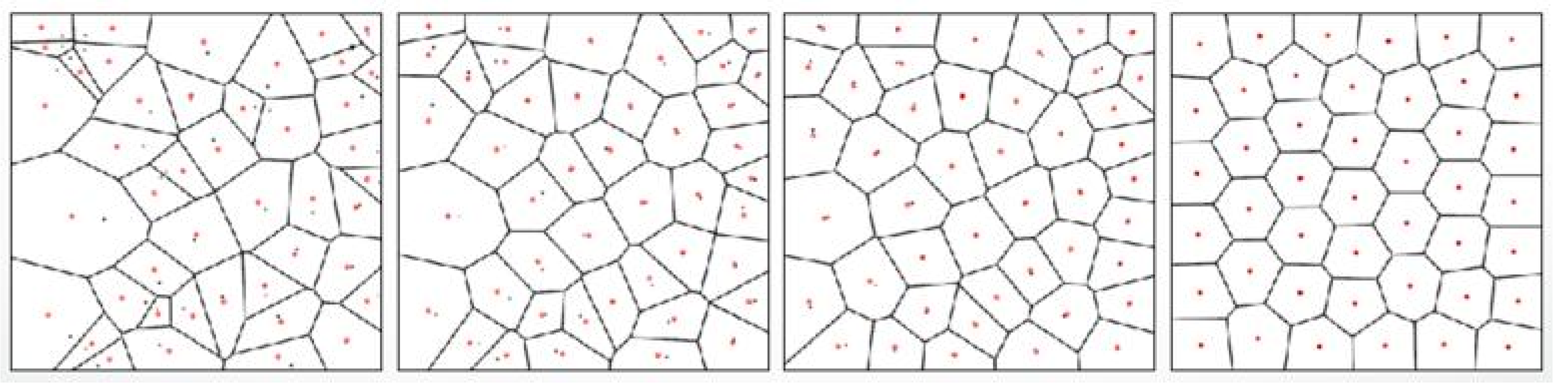
Remeshing

- ✓ Remeshing approaches: parameterization-based vs. surface-based.
- ✓ Param-based: map to 2D, do the triangulation (2D problem), lift it up.
- ✓ Delaunay triangulation: maximize the min angle = no point inside the circumcircle of a triangle.



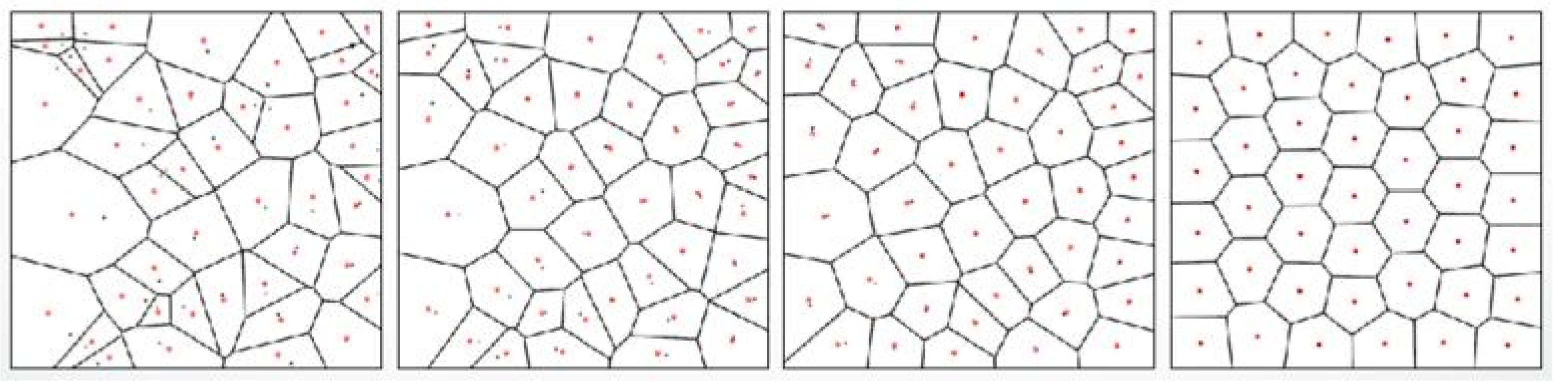
Remeshing

- ✓ Remeshing approaches: parameterization-based vs. surface-based.
- ✓ Param-based: map to 2D, do the triangulation (2D problem), lift it up.
- ✓ Obtain a cool distribution via Centroidal Voronoi Diagram, then lift up.
  - ✓ Compute Voronoi diagram of the given points pi (black).
  - ✓ Move points pi to centroids ci (red) of their Voronoi cells vi.
  - ✓ Repeat above until satisfaction.



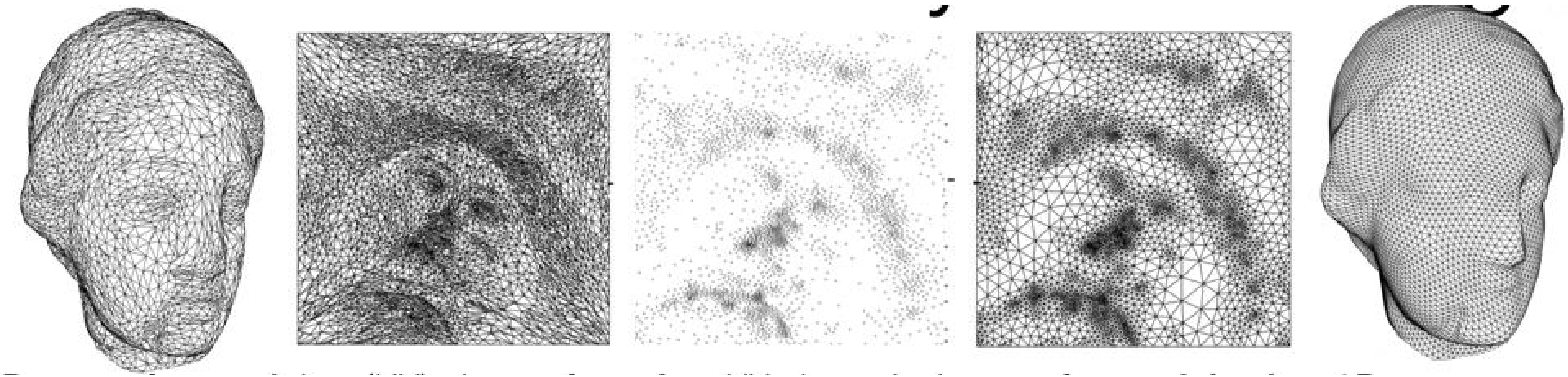
Remeshing

- ✓ Remeshing approaches: parameterization-based vs. surface-based.
- ✓ Param-based: map to 2D, do the distribution (2D problem), lift it up.
- ✓ Obtain a cool distribution via Centroidal Voronoi Diagram, then lift up.
- ✓ CVD can also be computed directly on the surface (Incremental remeshing approach). See paper: Isotropic Surface Remeshing Using Constrained Centroidal Delaunay Mesh.



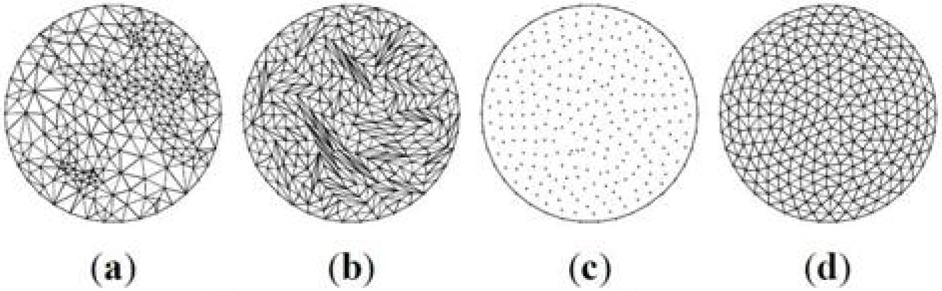
Remeshing

- ✓ Remeshing approaches: parameterization-based vs. surface-based.
- ✓ Param-based: map to 2D, do the triangulation (2D problem), lift it up.
  - ✓ See my Mesh Parameterization slides for the details of 3D to 2D mapping.

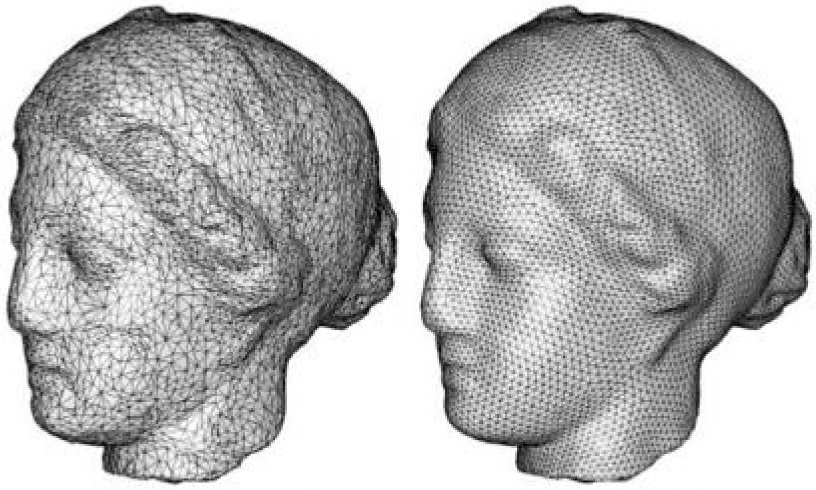


Remeshing

- ✓ Remeshing approaches: parameterization-based vs. surface-based.
- ✓ Param-based: map to 2D, do the triangulation (2D problem), lift it up.
  - ✓ See my Mesh Parameterization slides for the details of 3D to 2D mapping.



//b: area-equalized
version of a.
//c: edges discarded.
//d: 2D mesh to be lifted.

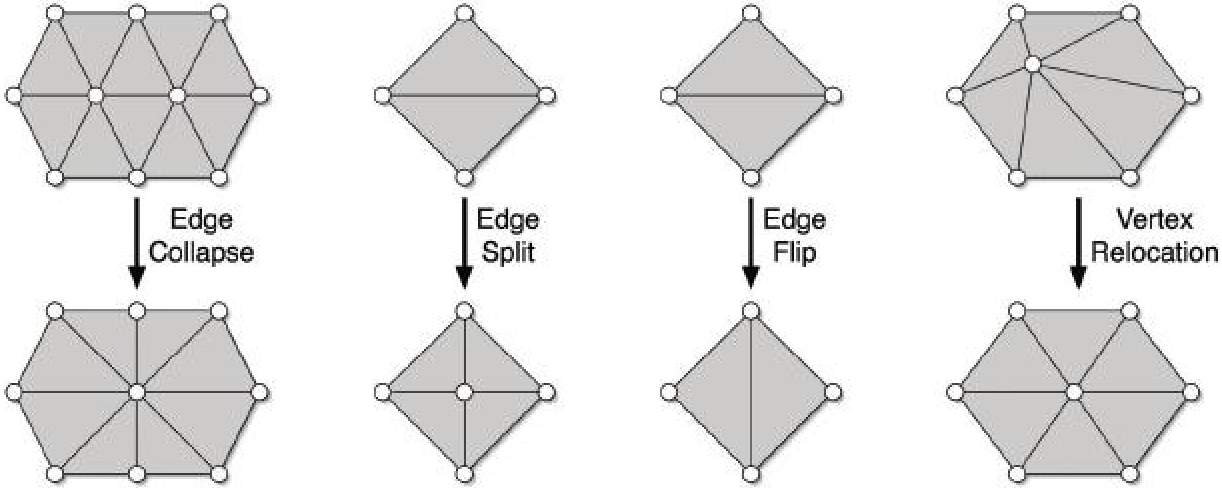


original (8,268 vertices) remesh (9,240 vertices)

Remeshing

✓ Remeshing approaches: parameterization-based vs. surface-based.

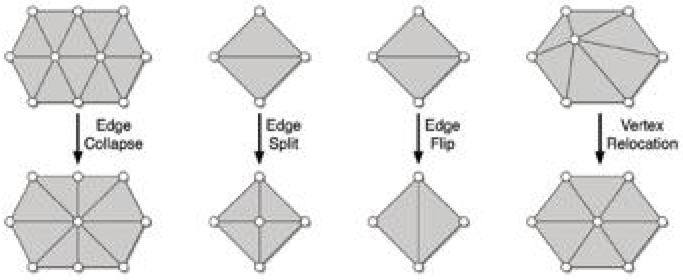
✓ Surface-based: work directly on the mesh embedded in 3D.



Local remeshing operators.

Remeshing

- ✓ Remeshing approaches: parameterization-based vs. surface-based.
- ✓ Surface-based: work directly on the mesh embedded in 3D.

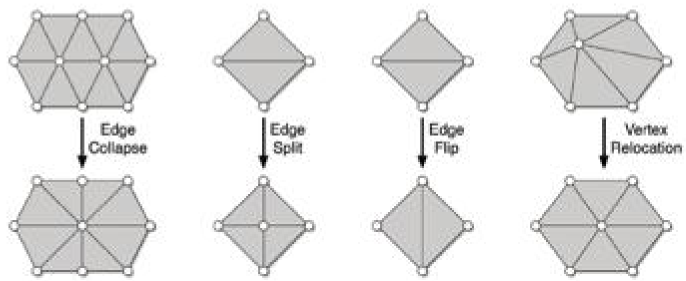


Local remeshing operators.

- ✓ Generic algo.
  - ✓ Specify edge length range [Lmin, Lmax].
  - ✓ Collapse edges shorter than Lmin.
  - ✓ Split edges longer than Lmax (or decide based on endpnts' curvatures).
  - ✓ Flip edges to get closer to valence 6 (Euler).
  - ✓ Vertex shift by Laplace smoothing.
- ✓ Lmax = 4/3L and Lmin = 4/5L where L is the target edge length.

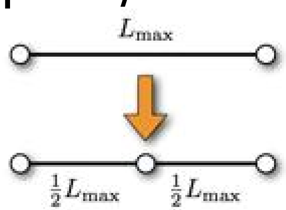
Remeshing

- ✓ Remeshing approaches: parameterization-based vs. surface-based.
- ✓ Surface-based: work directly on the mesh embedded in 3D.



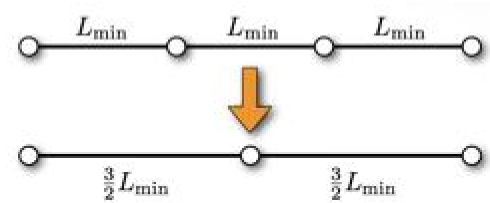
Local remeshing operators.

- ✓ Split 4/3L



$$\begin{aligned} |L_{\max} - L| &= \left|\frac{1}{2}L_{\max} - L\right| \\ \Rightarrow L_{\max} &= \frac{4}{3}L \end{aligned}$$

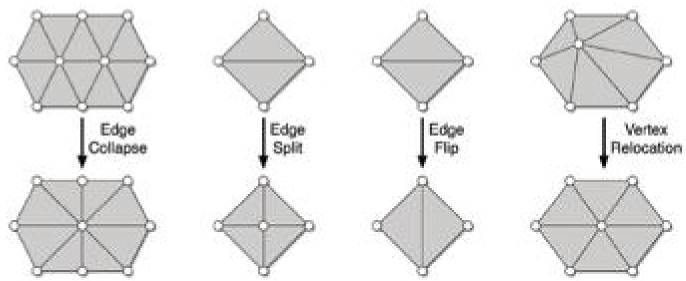
Collapse 4/5L



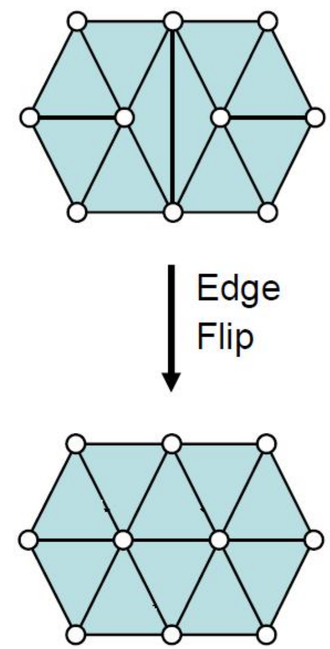
$$\begin{aligned} |L_{\min} - L| &= \left|\frac{3}{2}L_{\min} - L\right| \\ \Rightarrow L_{\min} &= \frac{4}{5}L \end{aligned}$$

Remeshing

- ✓ Remeshing approaches: parameterization-based vs. surface-based.
- ✓ Surface-based: work directly on the mesh embedded in 3D.



Local remeshing operators.

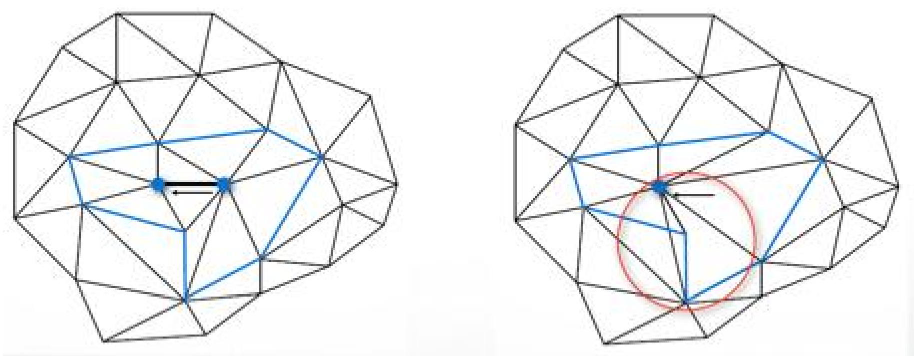


$$\sum_{i=1}^{4}\left(\mathrm{valence}(v_i) - \mathrm{opt\_valence}(v_i)\right)^2$$

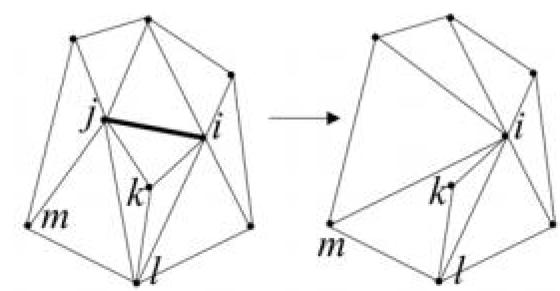
- ✓ Flip edges to get closer to valence 6 (interior), 4 (boundary).
  - ✓ Compute sum above before and after flip; if decreases, do it.

Remeshing

- ✓ Remeshing approaches: parameterization-based vs. surface-based.
- ✓ Surface-based: work directly on the mesh embedded in 3D.
- ✓ Beware of illegal edge collapses:



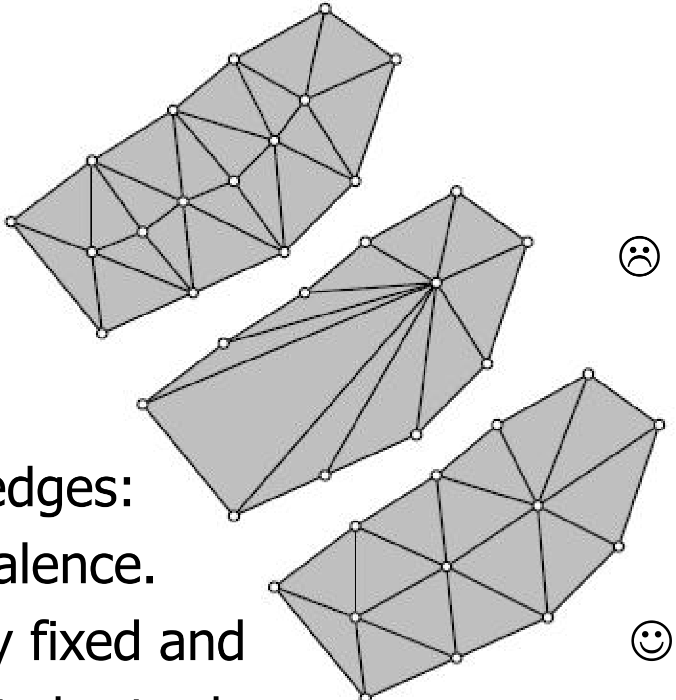
i) Normal flip after collapse!



ii) intersection of 1-ring neighborhood of i and j contains 3+ vertices!

Remeshing

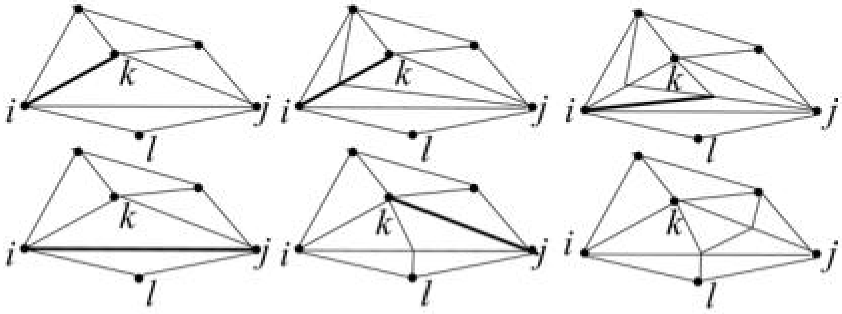
- ✓ Remeshing approaches: parameterization-based vs. surface-based.
- ✓ Surface-based: work directly on the mesh embedded in 3D.
- ✓ Beware of bad series of edge collapses:



i) A heuristic while removing short edges:
Collapse into the vert w/ higher valence.
Works 'cos high-valence verts stay fixed and
every collapse reduces # adjacent short edges.

Remeshing

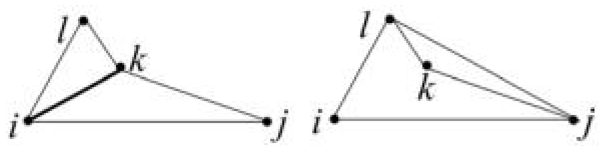
- ✓ Remeshing approaches: parameterization-based vs. surface-based.
- ✓ Surface-based: work directly on the mesh embedded in 3D.
- ✓ Beware of illegal edge splits:



i) Infinite-loop problem if you split shorter edges first (top row)!

Remeshing

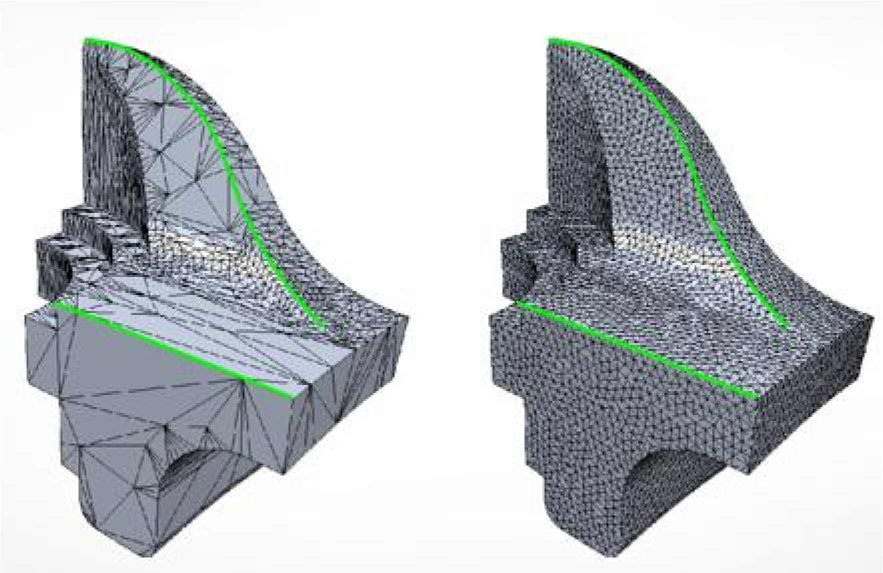
- ✓ Remeshing approaches: parameterization-based vs. surface-based.
- ✓ Surface-based: work directly on the mesh embedded in 3D.
- ✓ Beware of illegal edge flips:



i) edge is adjacent to 2 tris whose union is not a convex quadrilateral!
convex if no projection (of the 4th vert) is inside the tri (defined by the other 3 verts) //4th vert is projected to the plane defined by the other 3.

Remeshing

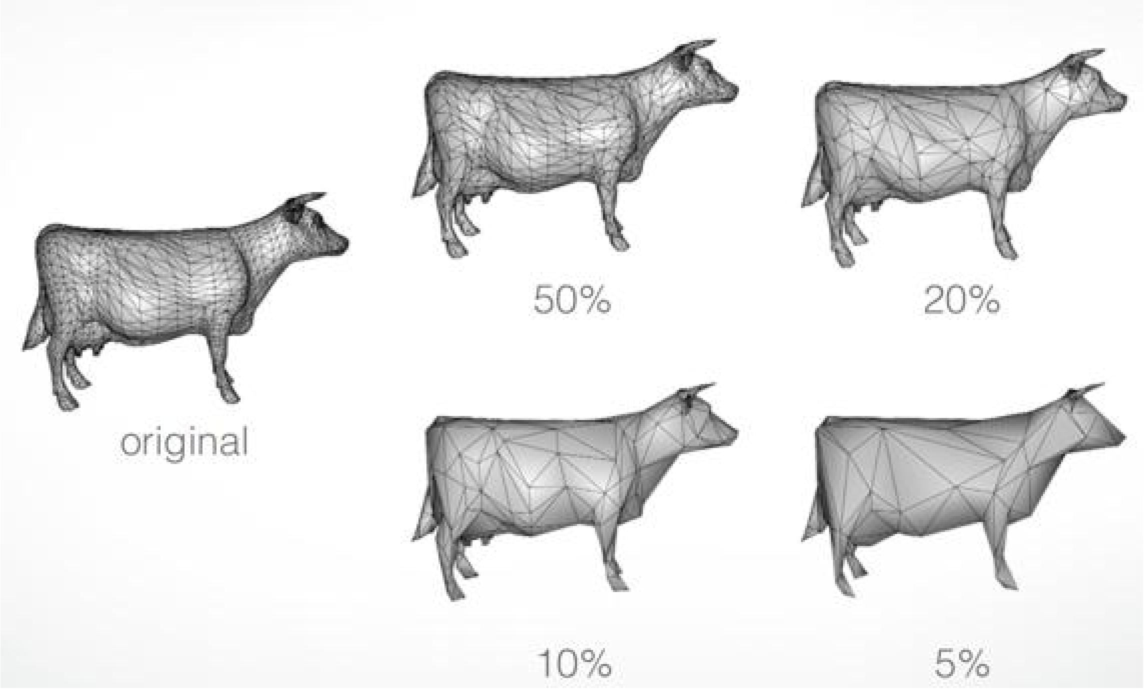
- ✓ Remeshing approaches: parameterization-based vs. surface-based.
- ✓ Surface-based: work directly on the mesh embedded in 3D.
- ✓ Feature preservation during remeshing is an important issue.



- ✓ Don’t flip feature edges.
- ✓ Collapse only along features. Project to feature curves.
- ✓ Don’t touch corner vertices.

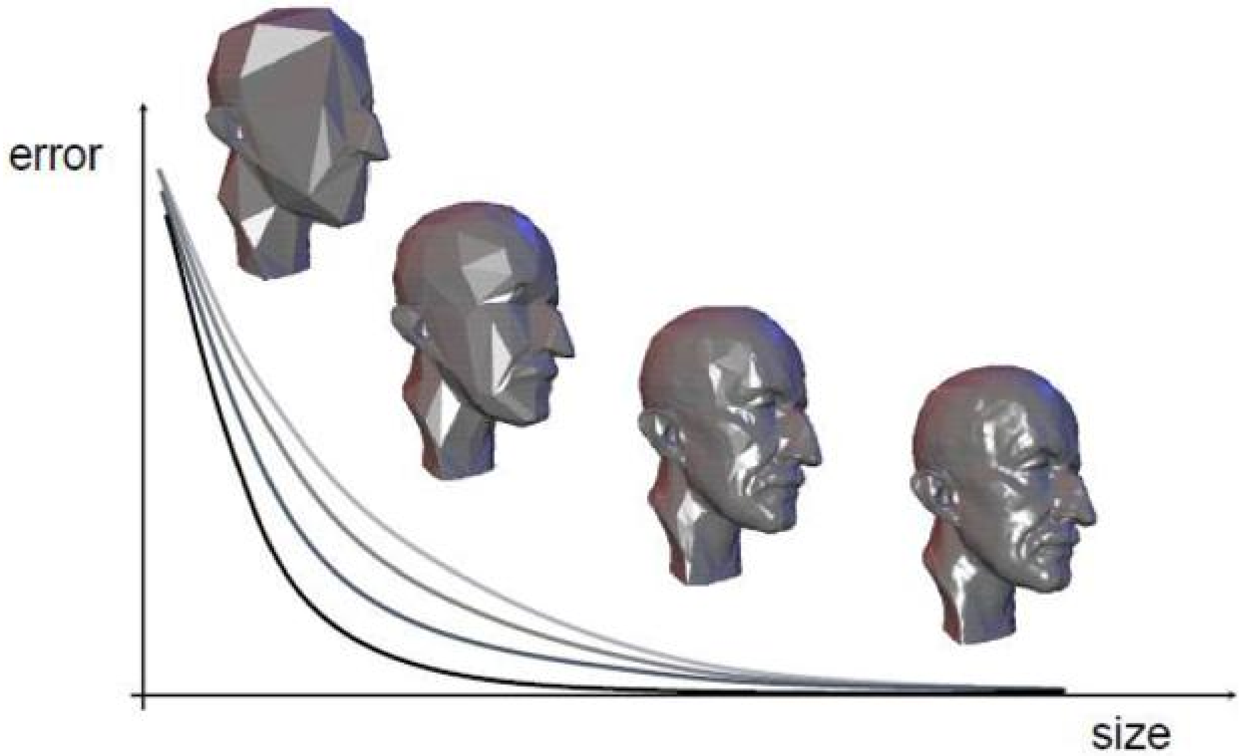
Remeshing

- ✓ Remeshing approaches: parameterization-based vs. surface-based.
- ✓ Surface-based: work directly on the mesh embedded in 3D.
- ✓ A sequence of edge collapses, aka mesh decimation:



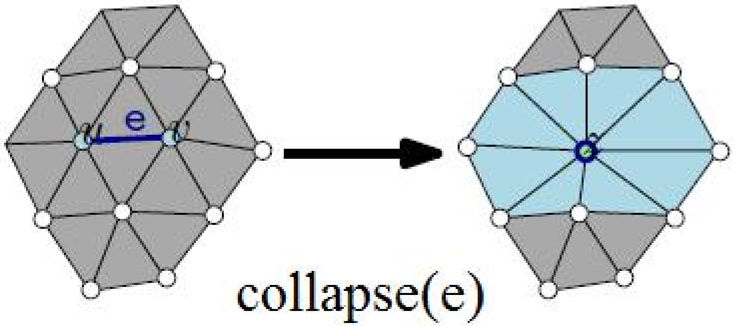
Remeshing

- ✓ Remeshing approaches: parameterization-based vs. surface-based.
- ✓ Surface-based: work directly on the mesh embedded in 3D.
- ✓ Mesh decimation/simplification has goals and trade-offs.



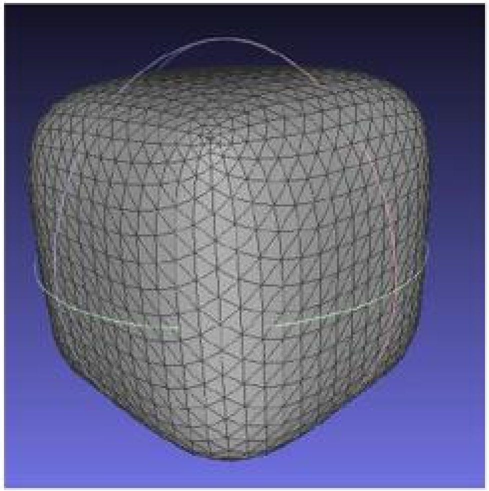
Remeshing

- ✓ Generic algorithm for mesh decimation/simplification:
  - ✓ Put all edges into a priority queue Q keyed on some cost function.
    - ✓ A common cost function is the length of an edge.
  - ✓ While Q not empty
    - ✓ Extract min from Q //edge to be contracted.
    - ✓ Collapse edge.
    - ✓ Update local costs for neighboring edges.

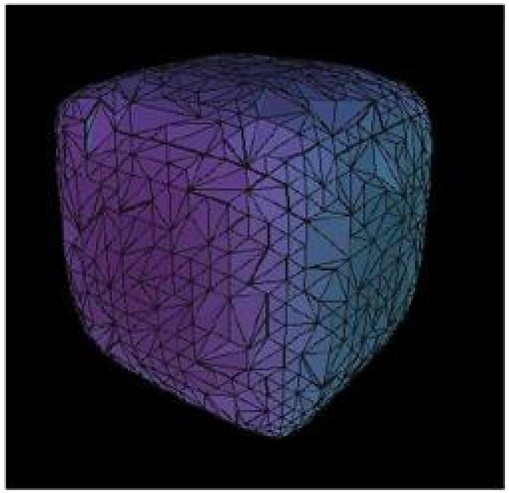


Remeshing

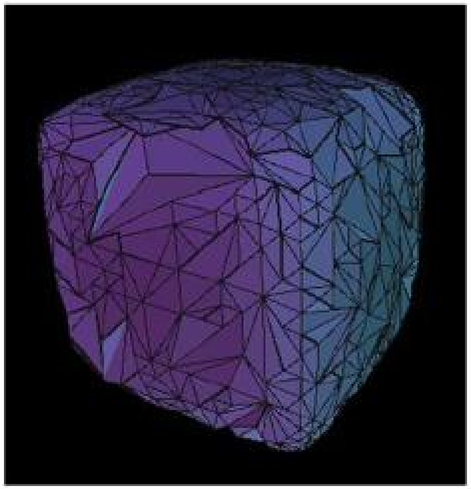
✓ Edge length as a cost function is intuitive (get rid of small edges) but not so robust.



Original



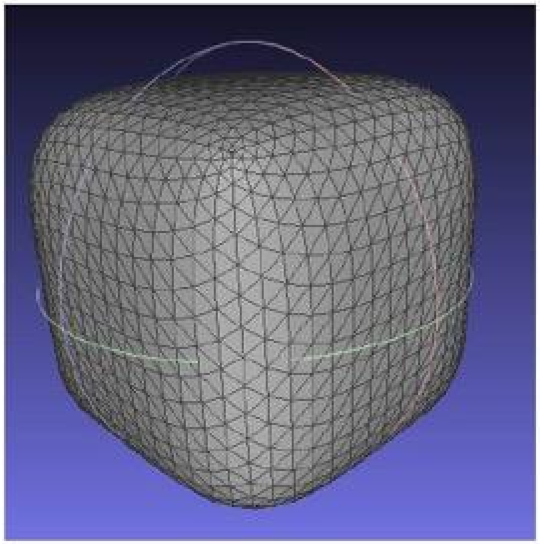
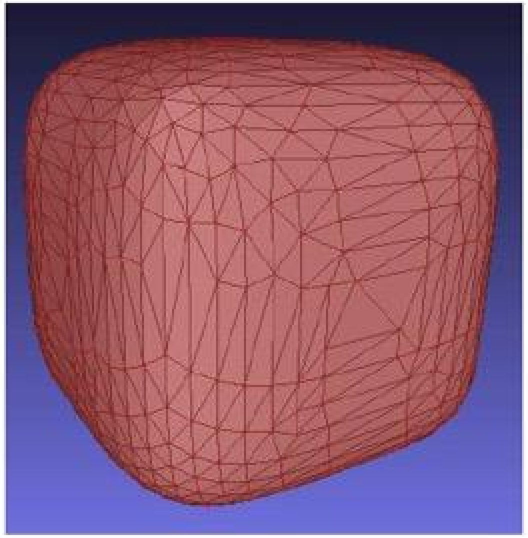
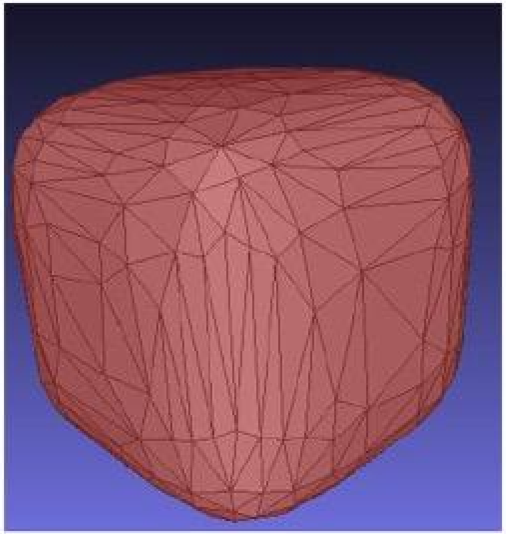
Cost = length



Cost = random

Remeshing

✓ Edge length as a cost function is intuitive (get rid of small edges) but not so robust.

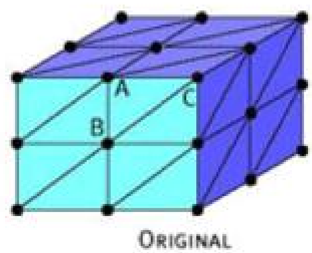
  

Original

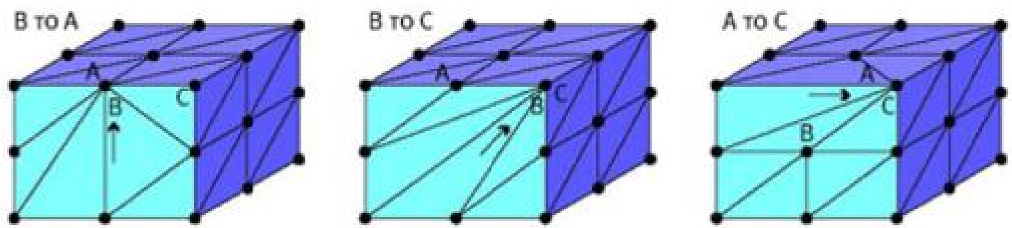
Better costs (next slides)

Remeshing

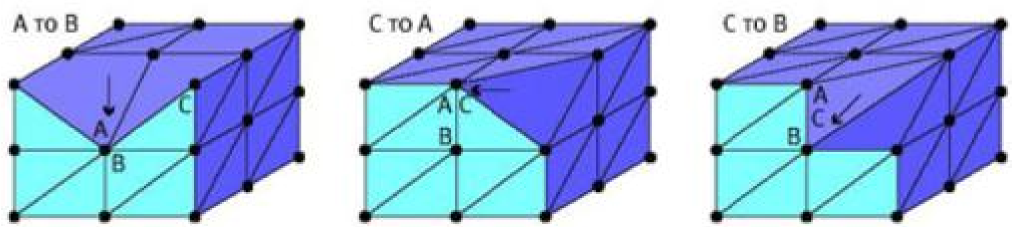
- ✓ Cost for edge collapses (other than edge length cost).
- ✓ Curvature factor is introduced as coplanar surfaces can be represented using fewer polygons than areas w/ a high curvature.



Good collapses:



Bad collapses:



Remeshing

- ✓ Cost for edge collapses (other than edge length cost).
- ✓ Curvature factor is introduced as coplanar surfaces can be represented using fewer polygons than areas w/ a high curvature.

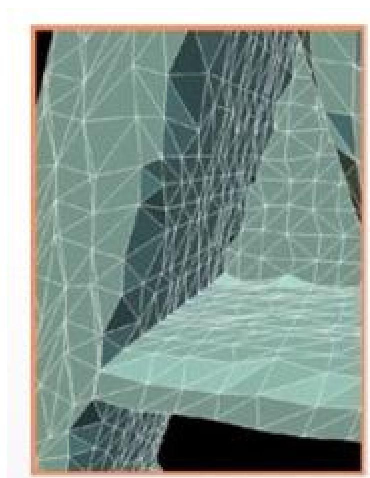
$$\mathrm{cost}(u,v) = \|u - v\| \times \max_{f \in \mathrm{T_u}} \left\{ \min_{g \in \mathrm{T_{uv}}} \left\{ \left(1 - \vec{n}_f \cdot \vec{n}_g \right) \div 2 \right\} \right\}$$

- ✓ Cost of collapsing u to v: Tu is the set of triangles that contain the vertex u and Tuv is the set of triangles that share the edge (u,v).
- ✓ Cost is length (||u-v||) multiplied by a curvature factor (< 1).
- ✓ Curvature factor computed by comparing the dot products of all involved face normals to find the largest angle b/w 2 faces.

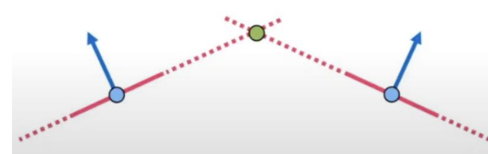
Remeshing

- ✓ Cost for edge collapses (other than edge length cost).
- ✓ Error quadric: based on the observation that in the original model each vertex is the solution of the intersection of a *set* of (supporting) planes.
- ✓ Error quadric represnt different **quadratic** dists such as dists to planes.
- ✓ The error at the vertex w.r.t. this *set* is the sum of **squared** distances to its supporting planes.
    - ✓ Surface Simplification Using Quadric Error Metrics, Garland & Heckbert, '97.
- ✓ This error helps preserving the original details in the decimated model.

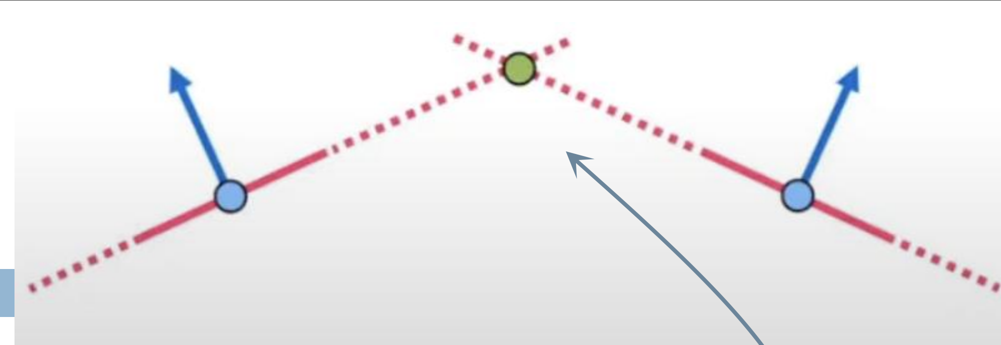
Edge-length metric:



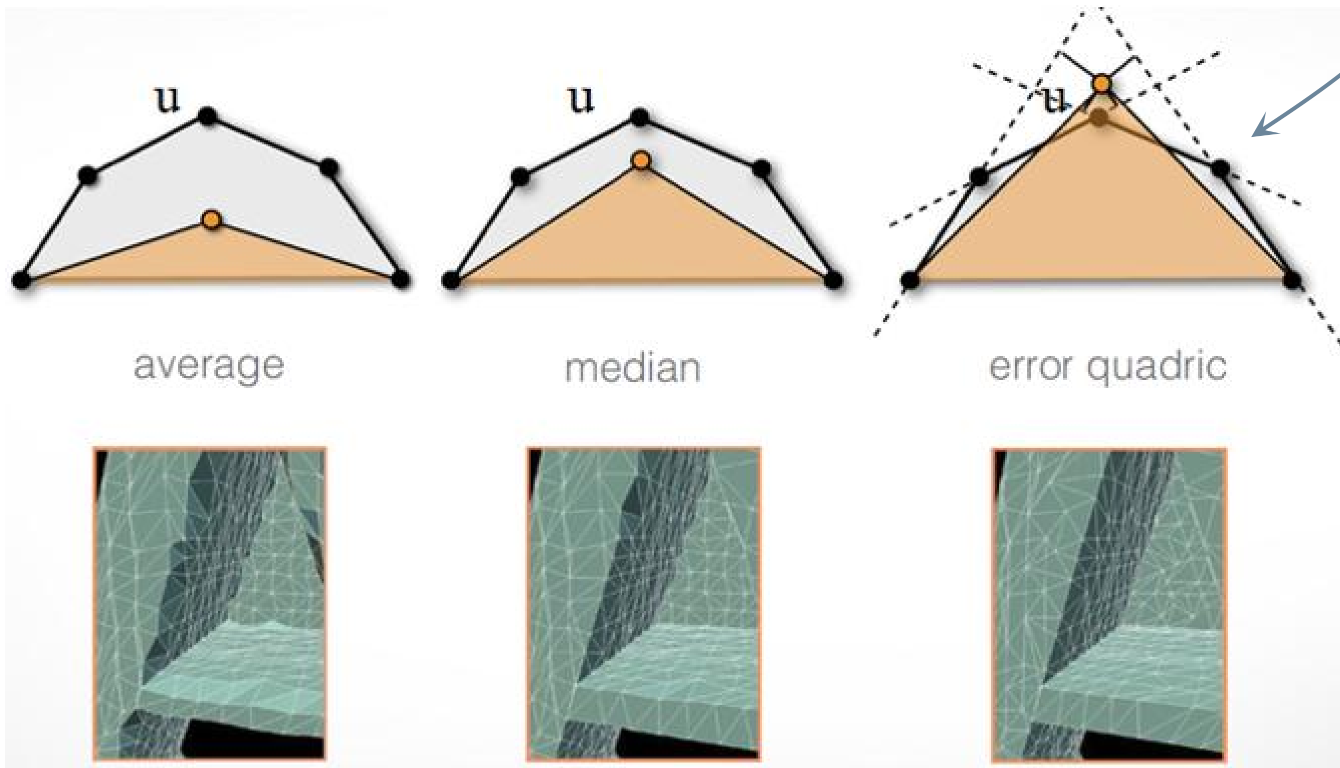
Error quadric metric:



Remeshing



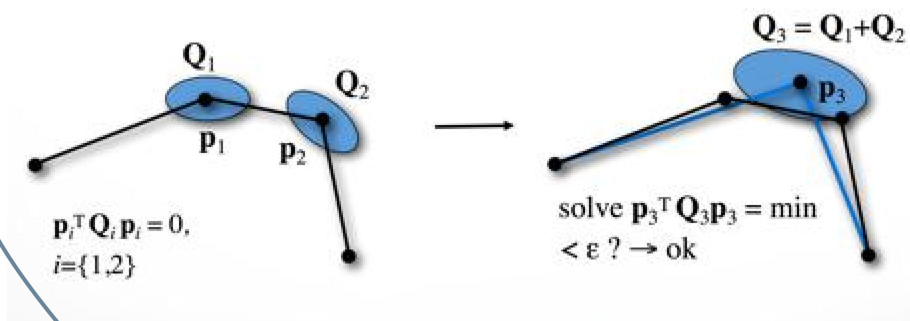
- ✓ Cost for edge collapses (other than edge length cost).
- ✓ Error quadric in 1D (supporting planes → supporting lines).



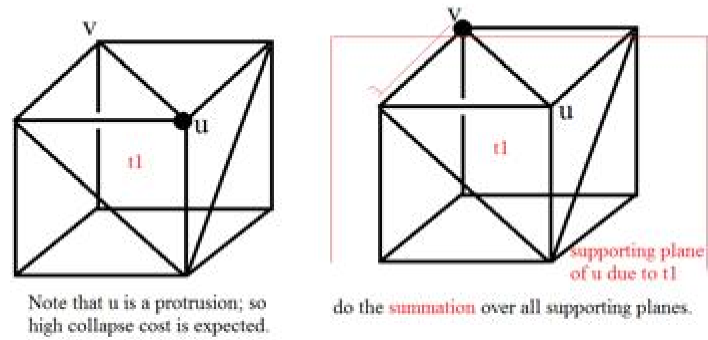
Put u to the center or median of neighbors. to a place that minimizes quadric err., i.e., sum of distances to support lines.

Remeshing

- ✓ Cost for edge collapses (other than edge length cost).
- ✓ First associate a symmetric 4x4 Q matrix to each vertex $v = [x\ y\ z\ 1]^T$.
  - ✓ Just 10 floating points per vertex (symmetric 4x4: see Slide 67).
- ✓ Define the error at v as: $d(v) = v^T Q v$.
- ✓ For a given contraction (v1, v2) → v', use Q' for the error: $d(v') = v'^T Q' v'$ where $Q' = Q1 + Q2$.



sum of squared distances to v's supporting planes.



v1=u, v2=v, v'=v in this example.
d(v') will be high: bad collapse.

Remeshing

- ✓ Cost for edge collapses (other than edge length cost).
- ✓ First associate a symmetric 4x4 Q matrix to each vertex $v = [x\ y\ z\ 1]^T$.
  - ✓ Just 10 floating points per vertex (quite compact).
- ✓ Define the error at v as: $d(v) = v^T Q v$.
- ✓ For a given contraction $(v1, v2) \rightarrow v'$, use Q' for the error: $d(v') = v'^T Q' v'$ where $Q' = Q1 + Q2$.
- ✓ Contraction point v' is the point that minimizes d(v').
  - ✓ For this, we set the partial derivatives of d(v') to zero, yielding

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \bar{\mathbf{v}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \Rightarrow \quad \bar{\mathbf{v}} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

- ✓ Can be verified by taking partl dervs of:

$$\mathbf{v}^T\mathbf{Q}\mathbf{v} = q_{11}x^2 + 2q_{12}xy + 2q_{13}xz + 2q_{14}x + q_{22}y^2 + 2q_{23}yz + 2q_{24}y + q_{33}z^2 + 2q_{34}z + q_{44}$$

- ✓ Quadric surfaces are the graphs of any equation that can be put into the general form: $Ax^2 + By^2 + Cz^2 + Dxy + Exz + Fyz + Gx + Hy + Iz + J = 0$.

Remeshing

- ✓ Cost for edge collapses (other than edge length cost).
- ✓ If matrix below has no inverse, then we use v1, v2, or midpoint as v'.
  - ✓ Not invertible in flat areas.

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \bar{\mathbf{v}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad \bar{\mathbf{v}} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Remeshing

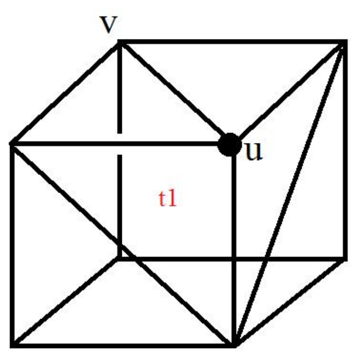
- ***Cost*** for edge collapses (other than edge length cost).
- Algo is then:
  - Compute Q matrices for all vertices.
  - Select all valid pairs (connected by edge, or too close).
  - Compute contraction point v′ for each valid pair (v1, v2). The error $v'^T (Q1+Q2) v'$ becomes the ***cost*** of contraction.
    - Note that $v'^T (Q1+Q2) v' = v'^T Q1 v' + v'^T Q2 v'$.
  - Place all pairs into a min heap keyed on costs.
  - Extract min pair, contract it, update costs of the involved pairs.

- $v'^T Q1 v'$: sum of the distances^2 of the supporting planes of v1 to v′.
- $v'^T Q2 v'$: sum of the distances^2 of the supporting planes of v2 to v′.

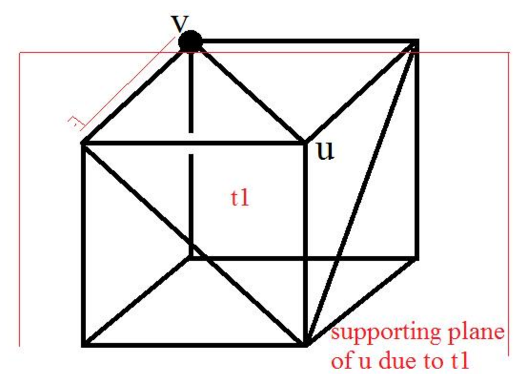
- Only remaining issue: what are the initial Q matrices?

Remeshing

- ✓ Cost for edge collapses (other than edge length cost).
- ✓ Q: error at each vertex is the sum of the squared distances of the supporting planes to that vertex.



Note that u is a protrusion; so high collapse cost is expected.

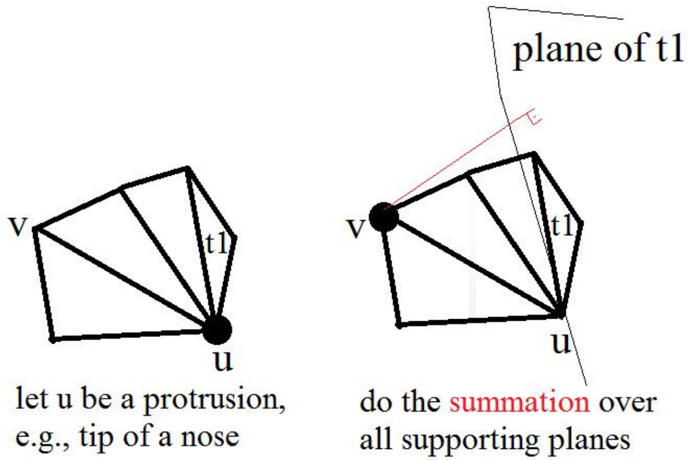


do the summation over all supporting planes.

- ✓ If contraction point of u is v, then we have a high distance to plane t1.

Remeshing

- ✓ Cost for edge collapses (other than edge length cost).
- ✓ Q: error at each vertex is the sum of the squared distances of the supporting planes to that vertex.



Remeshing

- ✓ Cost for edge collapses (other than edge length cost).
- ✓ Q captures this error if it is defined in the following way: d(v) = $\sum_{\mathbf{p}\in \text{planes}(\mathbf{v})} (\mathbf{p}^{\mathsf{T}}\mathbf{v})^2$

$$
\begin{aligned}
d(v) &= \sum_{\mathbf{p}\in \text{planes}(\mathbf{v})} (\mathbf{v}^{\mathsf{T}}\mathbf{p})(\mathbf{p}^{\mathsf{T}}\mathbf{v}) \\
&= \sum_{\mathbf{p}\in \text{planes}(\mathbf{v})} \mathbf{v}^{\mathsf{T}}(\mathbf{p}\mathbf{p}^{\mathsf{T}})\mathbf{v} \\
&= \mathbf{v}^{\mathsf{T}}\left(\sum_{\mathbf{p}\in \text{planes}(\mathbf{v})} \mathbf{K}_{\mathbf{p}}\right)\mathbf{v}
\end{aligned}
$$

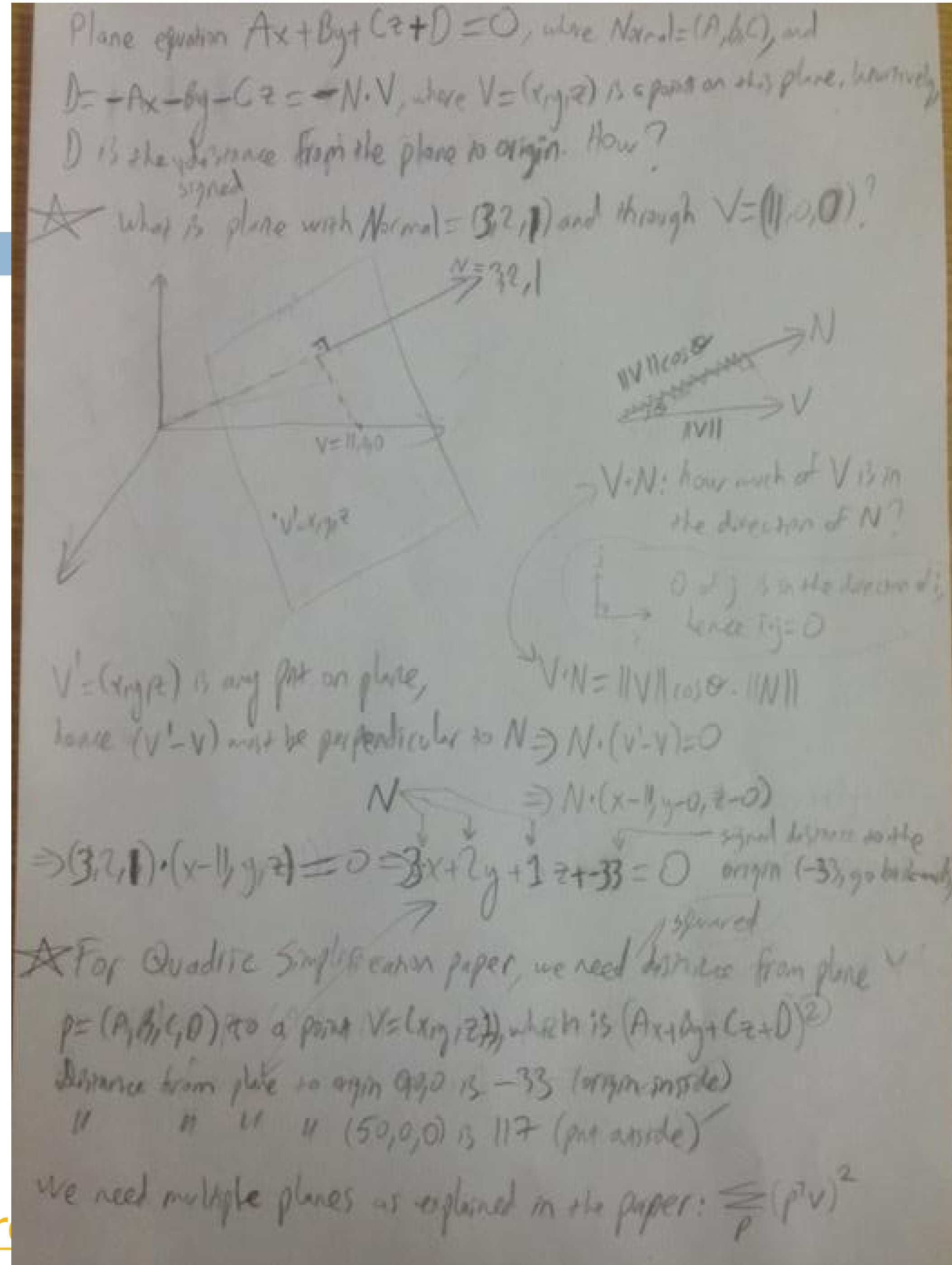
- ✓ p0v0 + p1v1 + p2v2, dot product, meaning the length of the projection of v on plane normal n = [p0, p1, p2]$^{\mathsf{T}}$.

$$
\mathbf{K}_{\mathbf{p}} = \mathbf{p}\mathbf{p}^{\mathsf{T}} = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}
$$

- ✓ Kp can be used to find the squared distance of any point in space to the plane p. We can sum them into Q and represent an entire set of planes by a single matrix Q.
- ✓ Note that the initial error estimate d(v) for each vertex is 0, since each vertex lies in the planes of all its incident triangles.

Remeshing

- ✓ Cost for edge collapses.
- ✓ Recall plane equation to understand d(v) better:
- ✓ p0, p1, p2, p3 become A, B, C, D here.


Plane equation $Ax+By+Cz+D=0$, where Normal $=(A,B,C)$, and
$D=-Ax-By-Cz=-N\cdot V$, where $V=(x,y,z)$ is a point on this plane. Intuitively
D is the signed distance from the plane to origin. How?
★ What is plane with Normal $=(3,2,1)$ and through $V=(11,0,0)$?
N=3,2,1
V=11,0,0
V'=x,y,z
||V||cosθ
N
V
||V||
V·N: how much of V is in the direction of N?
0 of j is in the direction of i, hence i·j=0
$V'=(x,y,z)$ is any pt on plane,
hence $(V'-V)$ must be perpendicular to $N \Rightarrow N\cdot(V'-V)=0$
$V\cdot N=\|V\|\cos\theta\cdot\|N\|$
N
$\Rightarrow N\cdot(x-11,y-0,z-0)$
$\Rightarrow (3,2,1)\cdot(x-11,y,z)=0 \Rightarrow 3x+2y+1z+-33=0$
signed distance to the origin (-33, go backward)
★ For Quadric Simplification paper, we need squared distance from plane
$p=(A,B,C,D)$ to a point $V=(x,y,z,1)$, which is $(Ax+By+Cz+D)^2$
Distance from plane to origin 0,0,0 is -33 (origin inside)
" " " " (50,0,0) is 117 (pt outside)
We need multiple planes as explained in the paper: $\sum_p (p^T v)^2$

- ✓ Cool plane generator: https://technology.cpm.or

Remeshing

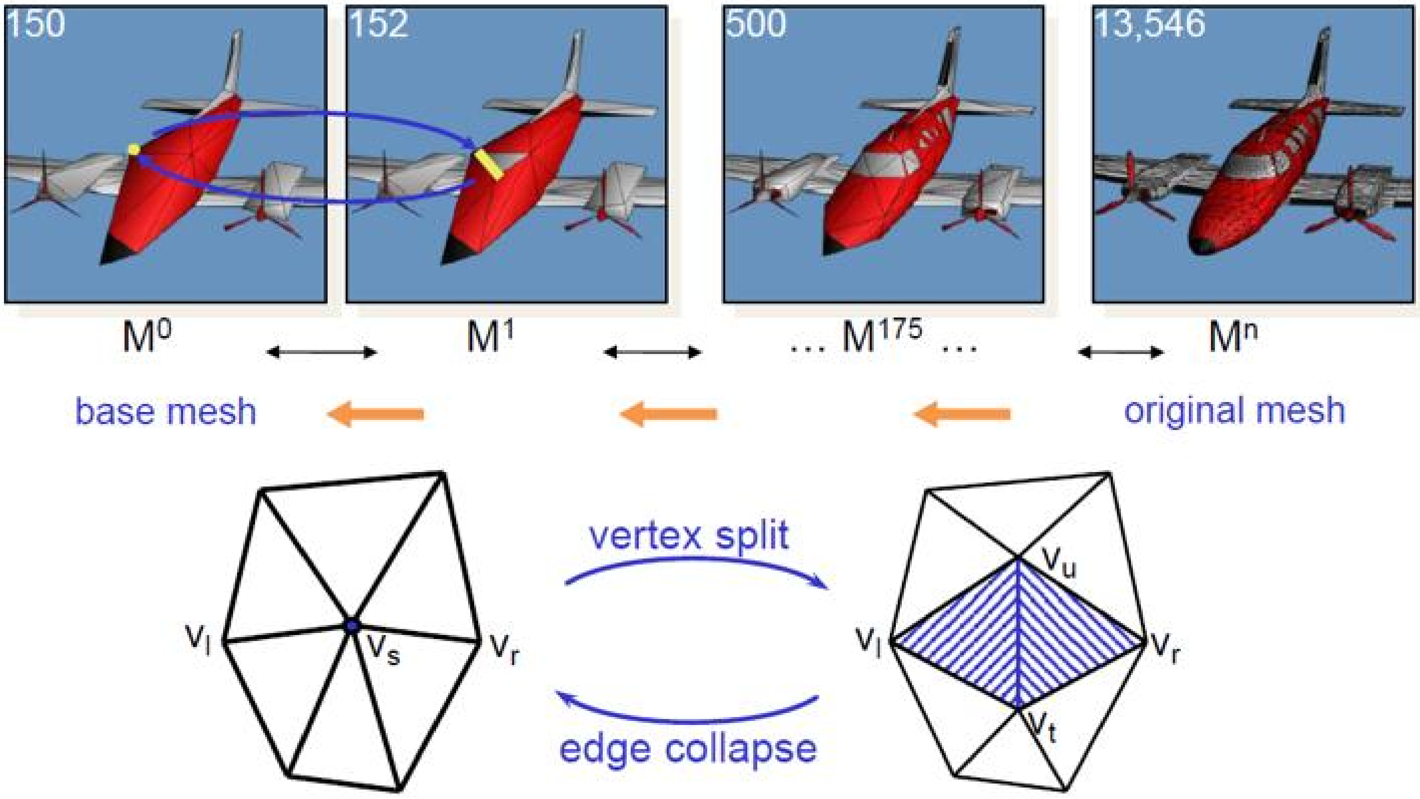
- ✓ Cost for edge collapses (other than edge length cost).
- ✓ Replacing distance to supporting planes error metric with distance to one-ring neighbors error metric.
  - ✓ More efficient as 5 floats per vertex stored instead of 10 (Slide 61).
- ✓ $d(v) = \sum ||v - vi||^2 = \sum (v - vi)^T(v - vi) = \sum v^Tv - 2v^T \sum vi + \sum vi^Tvi = n\, v^Tv - 2v^T \sum vi + \sum vi^Tvi$ (we've n neighbors around v (sum 1 to n)).

- ✓ 5 floats: n, $\sum vi$ (has 3 components), $\sum vi^Tvi$ to be applied to v for d(v).

- ✓ Contraction point is the minimum of d(v), hence set derv to 0, yielding $2nv - 2 \sum vi = 0 \Rightarrow v' = (\sum vi) / n$ is the contraction point.

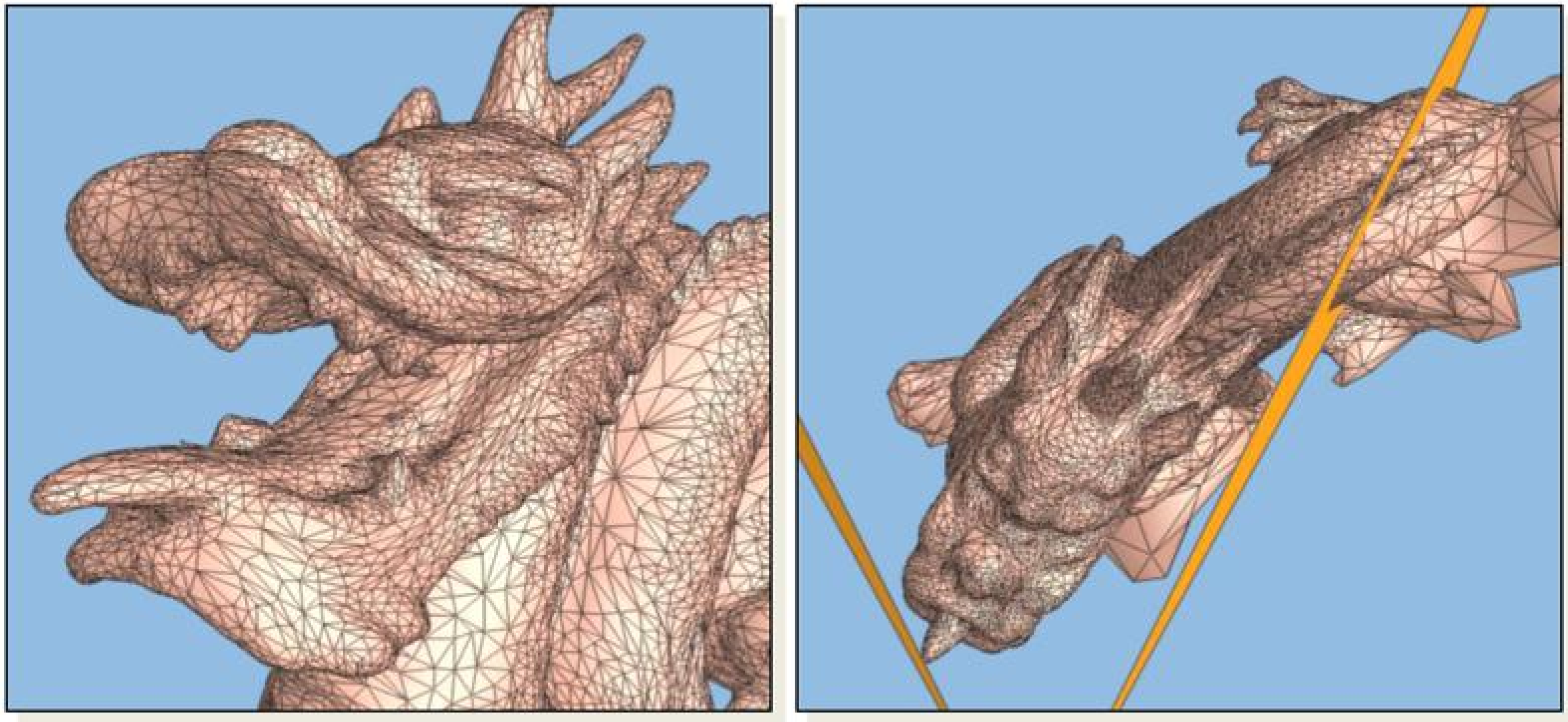
Remeshing

- ✓ Progressive meshes as an option to undo remeshing on the fly.
- ✓ Alternating vertex split and edge collapse operations.
  - ✓ Progressive meshes, H. Hoppe, 1996.



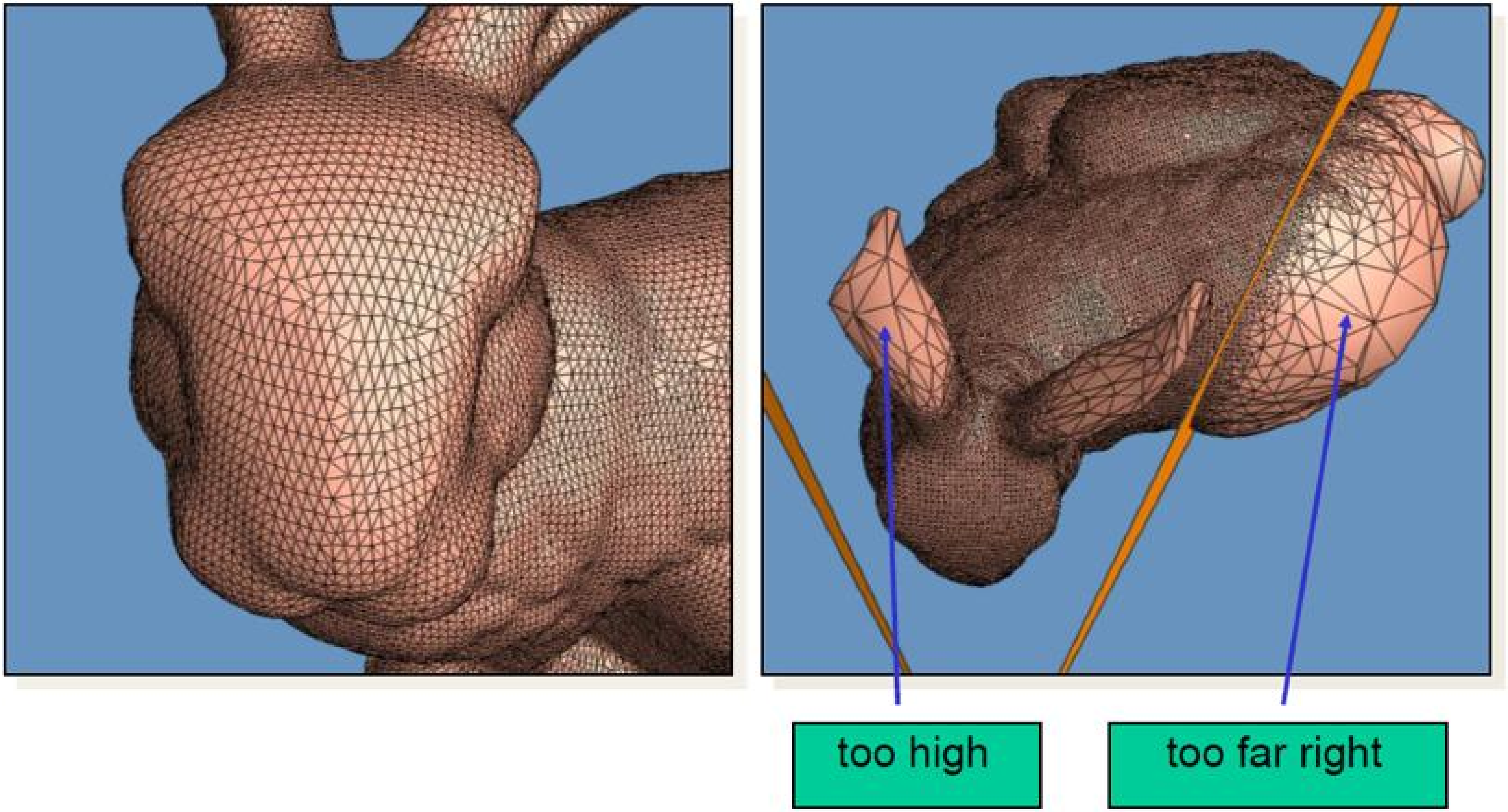
Remeshing

- ✓ Progressive meshes as an option to undo remeshing on the fly.
- ✓ Good for view-dependent rendering (LOD).



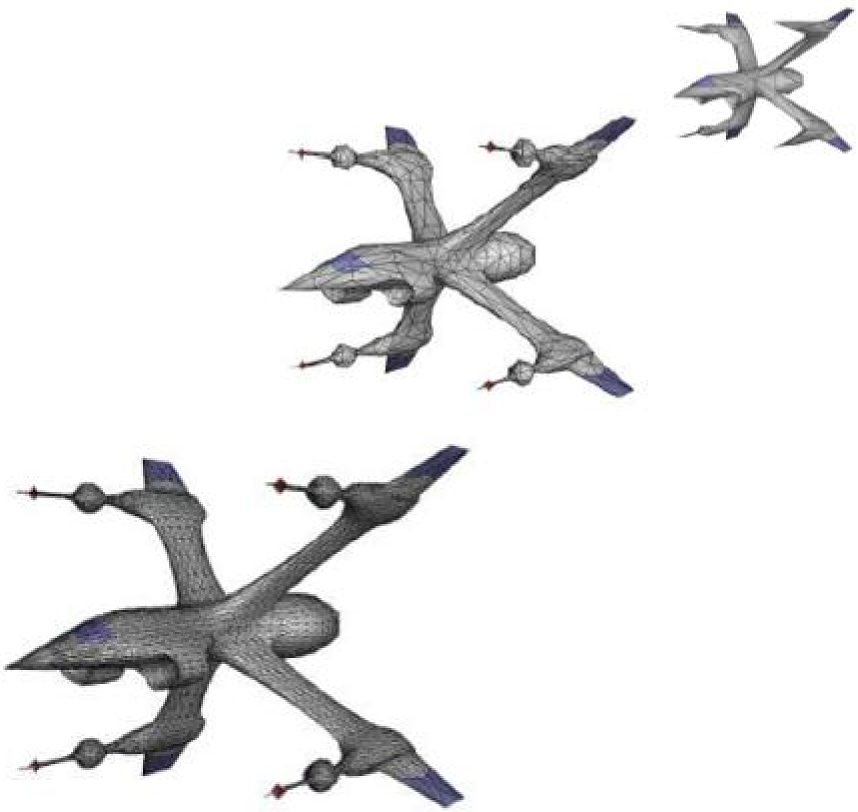
Remeshing

- ✓ Progressive meshes as an option to undo remeshing on the fly.
- ✓ Good for view-dependent rendering (LOD).



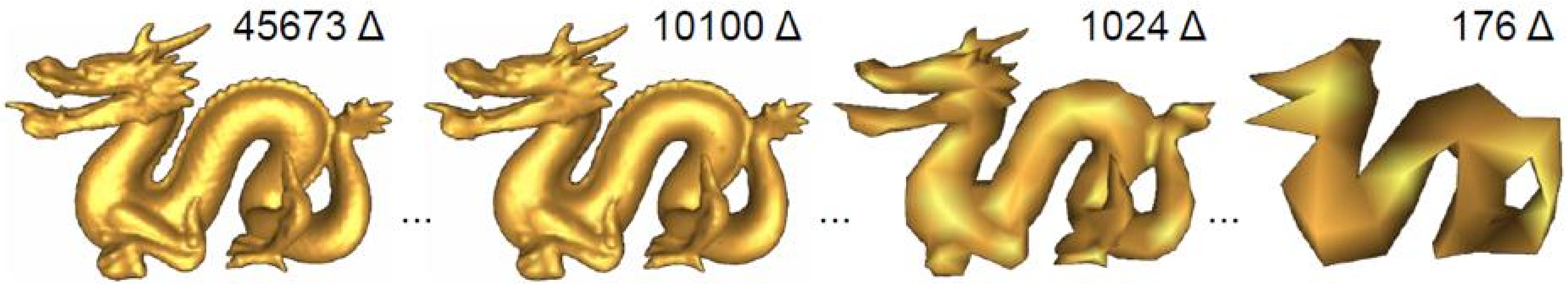
Remeshing

- ✓ Progressive meshes as an option to undo remeshing on the fly.
- ✓ Good for distance-dependent rendering (LOD).



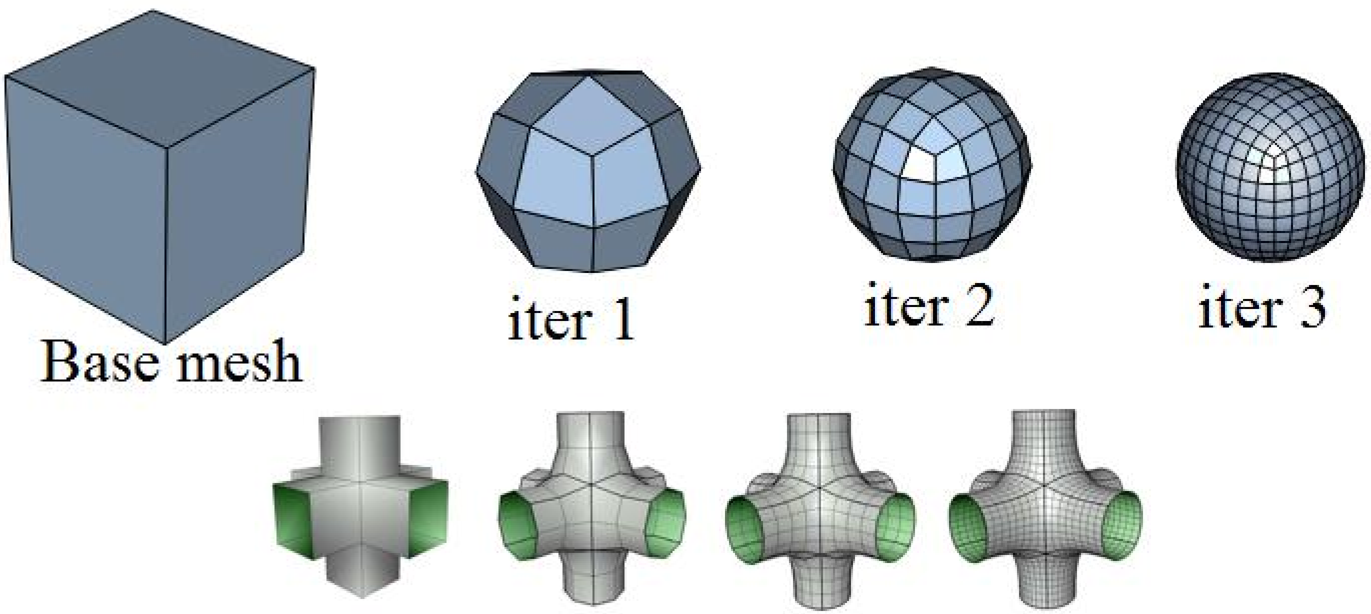
Remeshing

- ✓ Progressive meshes as an option to undo remeshing on the fly.
- ✓ Good for distance-dependent rendering (LOD).



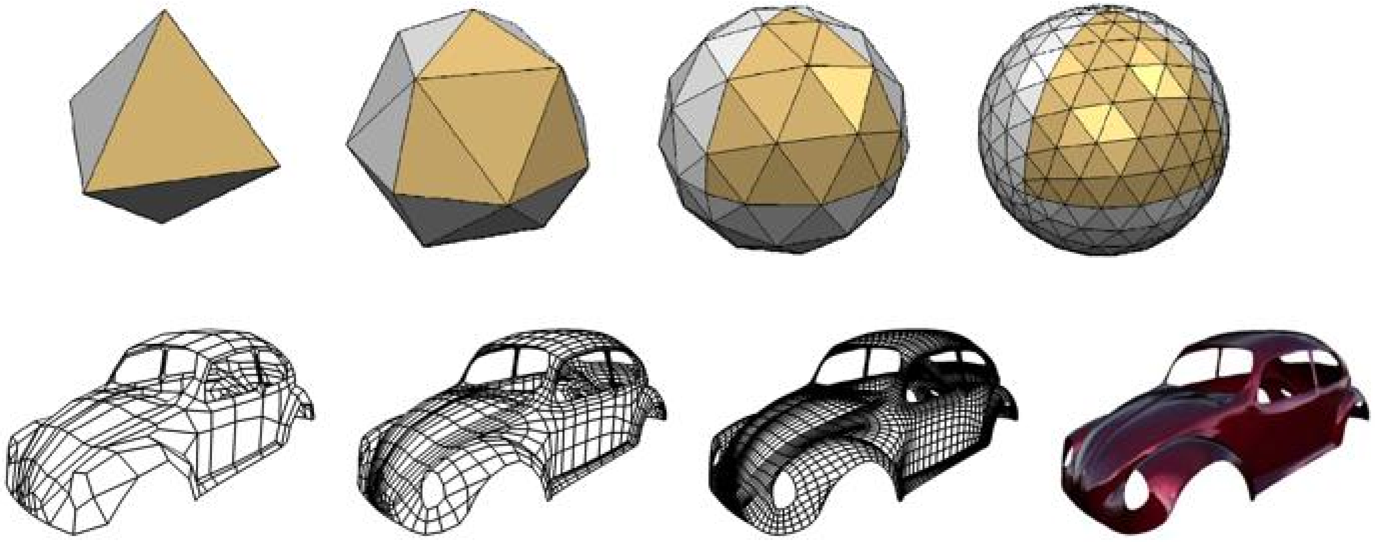
Subdivision Surfaces

- ✓ Related to remeshing.
- ✓ Sub-d surface: a method to represent a smooth surface via a coarser polygon mesh (base mesh).
- ✓ Take base mesh.
- ✓ Subdivide it; creating new vertices and faces (like our edge split).
- ✓ Compute positions of new verts based on positons of nearby old verts.



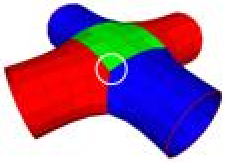
Subdivision Surfaces

- ✓ Related to remeshing.
- ✓ Sub-d surface: a method to represent a smooth surface via a coarser polygon mesh (base mesh).
- ✓ Take base mesh.
- ✓ Subdivide it; creating new vertices and faces (topology update).
- ✓ Compute positions of new verts based (geometry update).

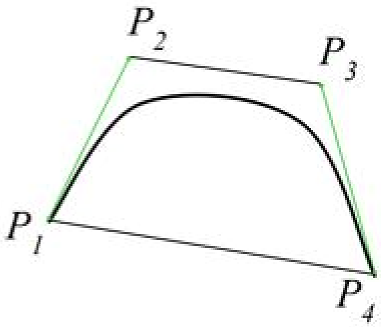


Subdivision Surfaces

- ✓ Instead of updating a parameter t along a parametric curve (Bezier) or parameters u, v over a parametric grid (NURBS), subdivision surfaces repeatedly refine from a coarse set of control points.
  - ✓ Sub-d surface is an alternative to NURBS patches.
  - ✓ NURBS patches are hard to join with nice continuity across an edge or a corner.



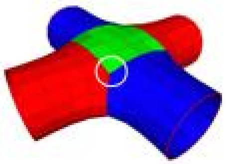
  - ✓ Cubic Bezier Curve:



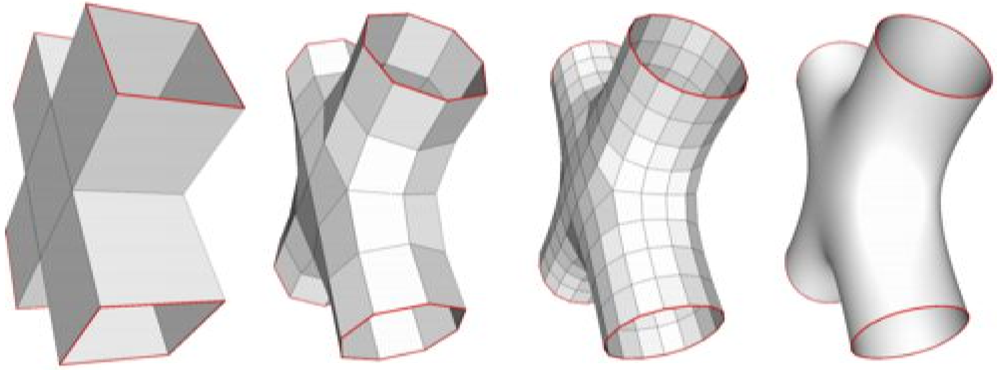
$$P(t) = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t) P_3 + t^3 P_4$$

Subdivision Surfaces

- ✓ Instead of updating a parameter t along a parametric curve (Bezier) or parameters u, v over a parametric grid (NURBS), subdivision surfaces repeatedly refine from a coarse set of control points.
  - ✓ Sub-d surface is an alternative to NURBS patches.
  - ✓ NURBS patches are hard to join with nice continuity across an edge or a corner.

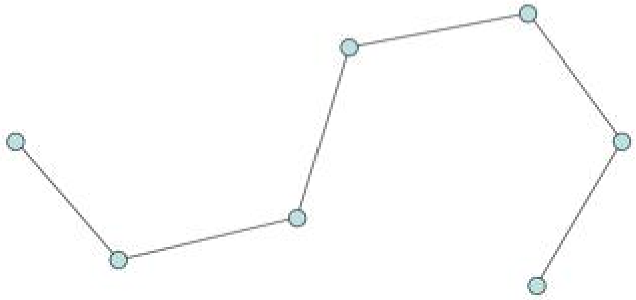


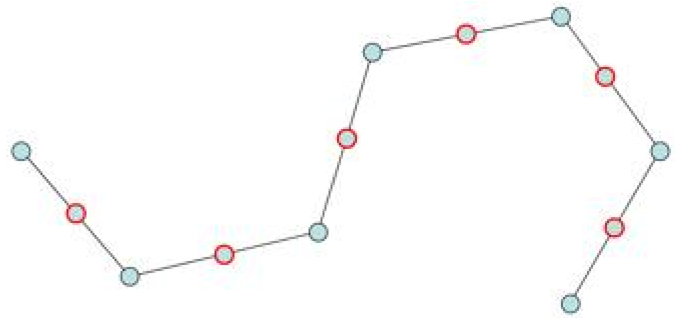
- ✓ Each step of refinement adds new faces and vertices.
- ✓ The process converges to a smooth limit surface.



Subdivision Curves

- ✓ Approximating curve (nearby control points); there's also interpolating curves (through control points).

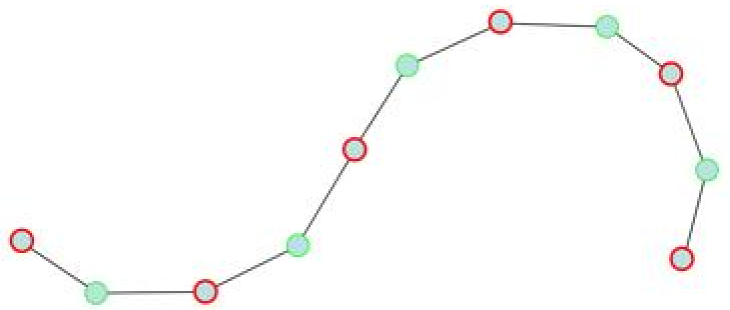


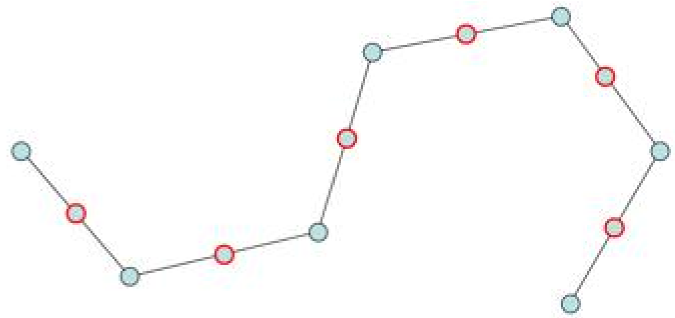


Split each edge in two.
-- topology update --

Subdivision Curves

✓ Approximating curve (nearby control points); there's also interpolating curves (through control points).



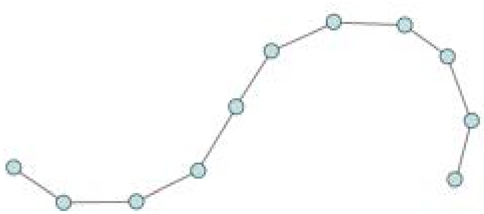


Relocate each original vertex.
-- geometry update --

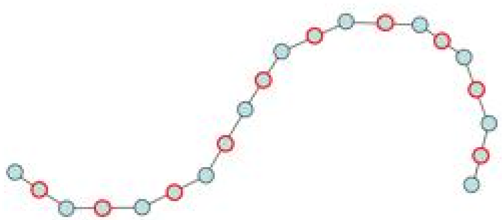
Rule: halfway through the center of (new) neighbors, like Laplacian smoothing.

Subdivision Curves

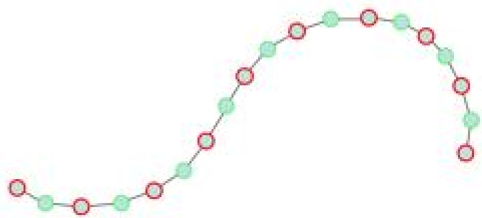
✓ Approximating curve (nearby control points); there's also interpolating curves (through control points).



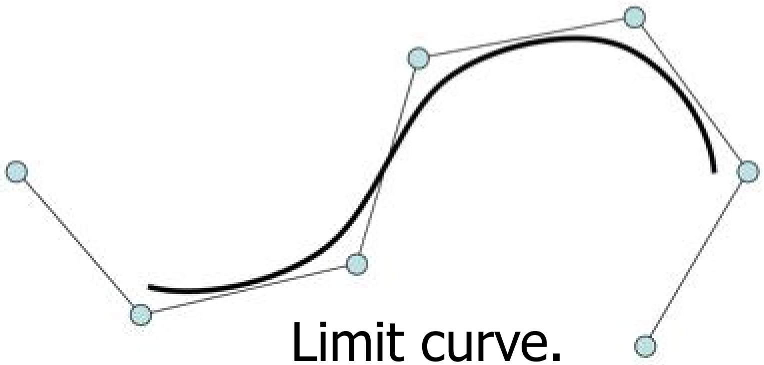
Start over.



Split.



Relocate.

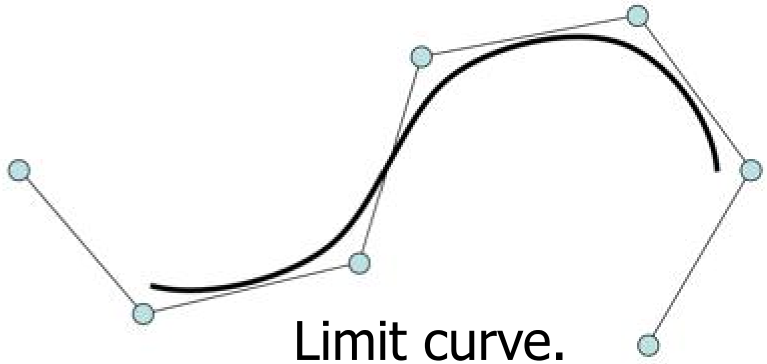


Limit curve.

Subdivision Curves

- ✓ Approximating curve (nearby control points); there's also interpolating curves (through control points).

- ✓ Same effect if the geometry rule is changed as follows (Catmull-Clark):
  - ✓ All (not just the originals) are moved in triplets. Middle is weighted by 6/8 and left-right by 1/8; hence the filter/mask: (1/8, 6/8, 1/8).



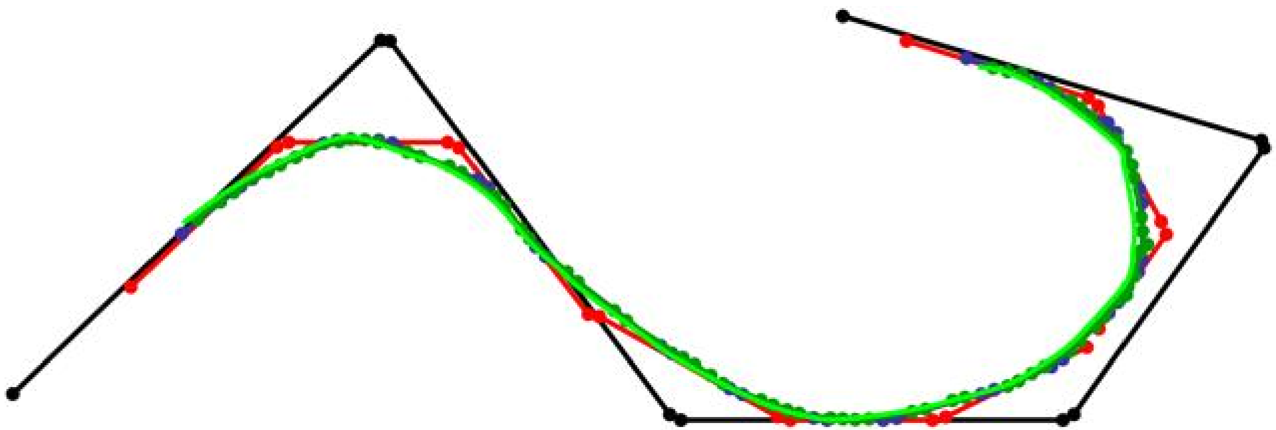
Limit curve.

Subdivision Curves

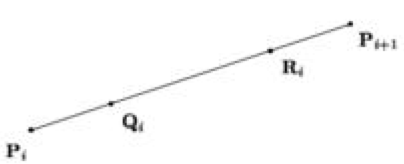
- ✓ Corner cutting.
- ✓ Insert two new vertices at ¼ and ¾ of each edge.
- ✓ Remove the old vertices.
- ✓ Connect the new vertices.

Subdivision Curves

- ✓ Corner cutting.
- ✓ Application of this scheme to a non-closed base (polyline) gives:



- ✓ aka Chaikin curve subdivision (1974).

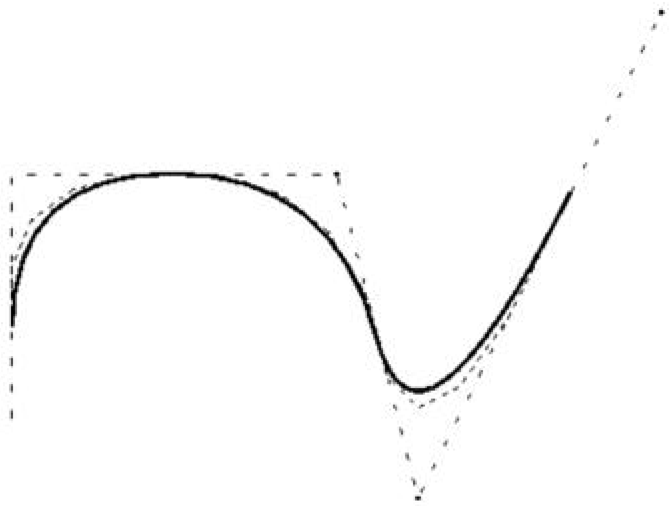


$$Q_i = \frac{3}{4}P_i + \frac{1}{4}P_{i+1}$$

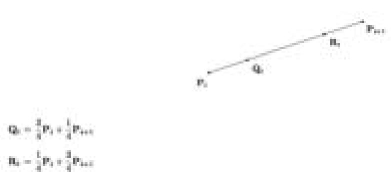
$$R_i = \frac{1}{4}P_i + \frac{3}{4}P_{i+1}$$

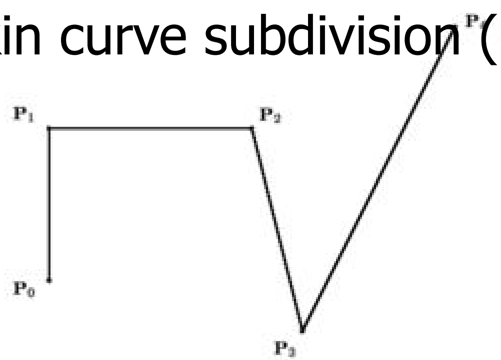
Subdivision Curves

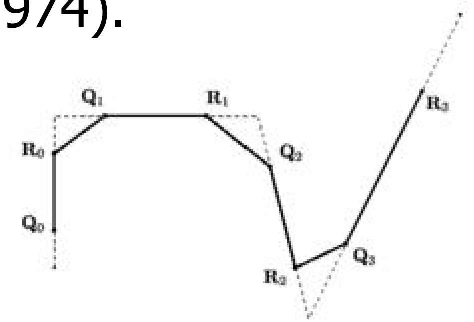
- ✓ Corner cutting.
- ✓ Application of this scheme to a non-closed base (polyline) gives:

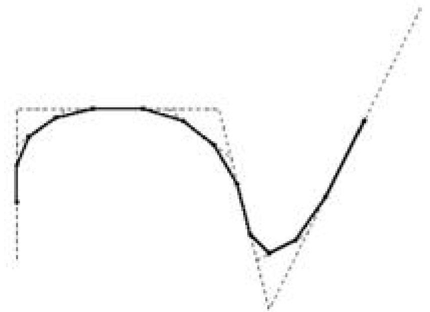


- ✓ aka Chaikin curve subdivision (1974).



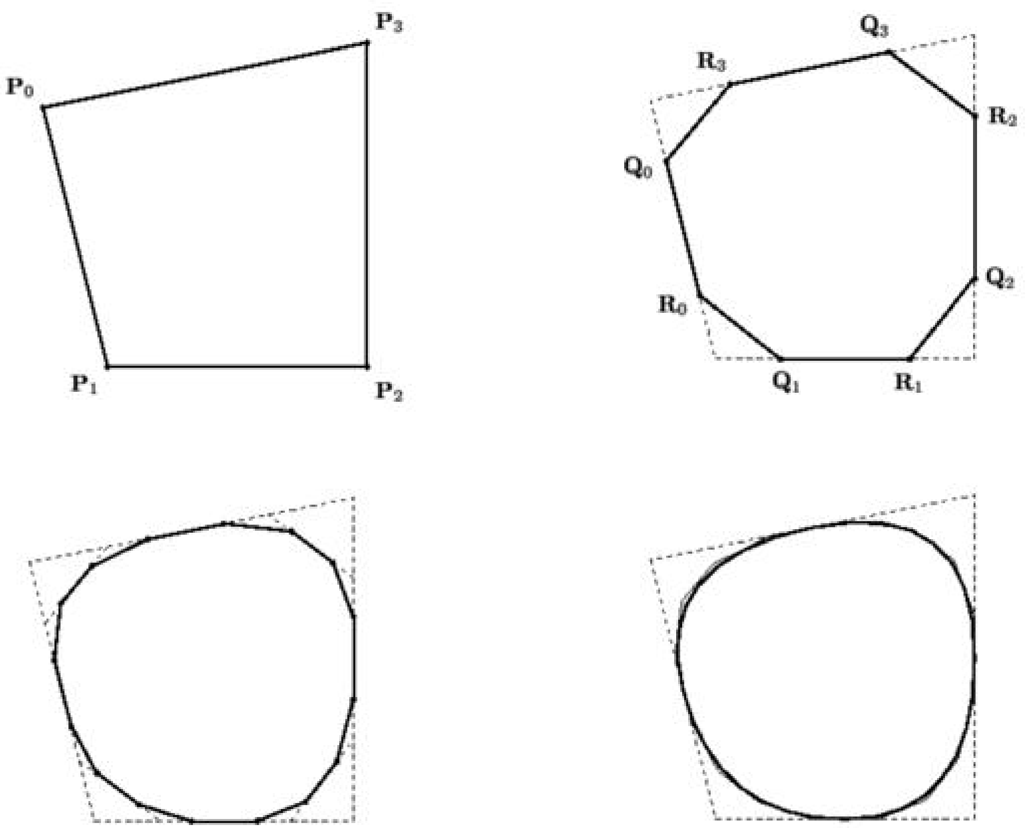






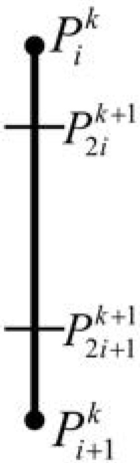
Subdivision Curves

- ✓ Corner cutting.
- ✓ Application of this scheme to a closed base (polygon) gives:



Subdivision Curves

✓ Chaikin can be coded as follows:



$$P_{2i}^{k+1} = (3/4)P_i^k + (1/4)P_{i+1}^k \quad \leftarrow Even$$

$$P_{2i+1}^{k+1} = (1/4)P_i^k + (3/4)P_{i+1}^k \quad \leftarrow Odd$$

k is the generation. Each gen has twice as many ctrl pnts as before.

Note that even and odd ctrl pnts are generated differently.

Borders, if any, are treated specially.

Subdivision Curves

- ✓ Chaikin can be coded as follows:

$$\begin{bmatrix} \vdots \\ P_{2i-2}^{k+1} \\ P_{2i-1}^{k+1} \\ P_{2i}^{k+1} \\ P_{2i+1}^{k+1} \\ P_{2i+2}^{k+1} \\ P_{2i+3}^{k+1} \\ \vdots \end{bmatrix} = \frac{1}{4} \begin{bmatrix} & & & \vdots & & & & \\ \cdots & 0 & 3 & 1 & 0 & 0 & 0 & \cdots \\ & 0 & 1 & 3 & 0 & 0 & 0 & \\ & 0 & 0 & 3 & 1 & 0 & 0 & \\ & 0 & 0 & 1 & 3 & 0 & 0 & \\ & 0 & 0 & 0 & 3 & 1 & 0 & \\ & 0 & 0 & 0 & 1 & 3 & 0 & \\ & & & \vdots & & & & \end{bmatrix} \begin{bmatrix} \vdots \\ P_{i-2}^{k} \\ P_{i-1}^{k} \\ P_{i}^{k} \\ P_{i+1}^{k} \\ P_{i+2}^{k} \\ P_{i+3}^{k} \\ \vdots \end{bmatrix}$$

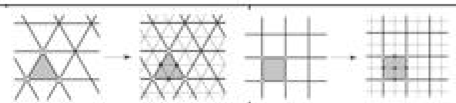
- ✓ This vector notation compresses the scheme to a kernel: $h = (1/4)[\ldots,0,0,1,3,3,1,0,0,\ldots]$
- ✓ Eigenanalysis of this matrix proves continuity of the limit curve.
  - ✓ Turns out that the limit curve is a quadratic B-spline.

Subdivision Surfaces

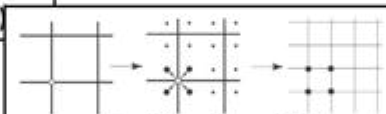
✓ Categorization:

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



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



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

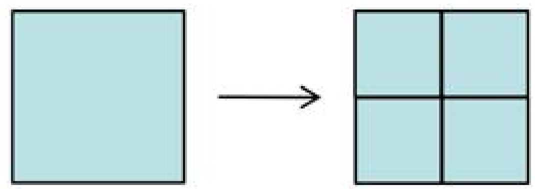
Subdivision Surfaces

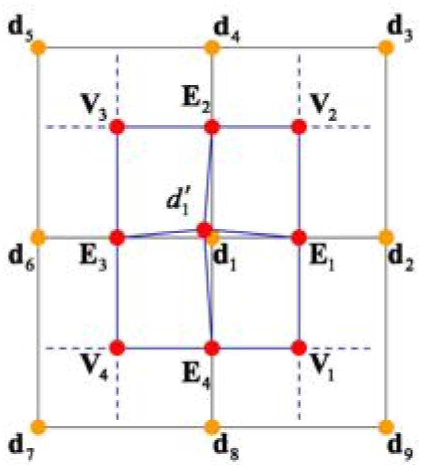
✓ Categorization of classical schemes:

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

Subdivision Surfaces

- ✓ Catmull-Clark subdivision to refine quad surfaces/meshes.





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

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

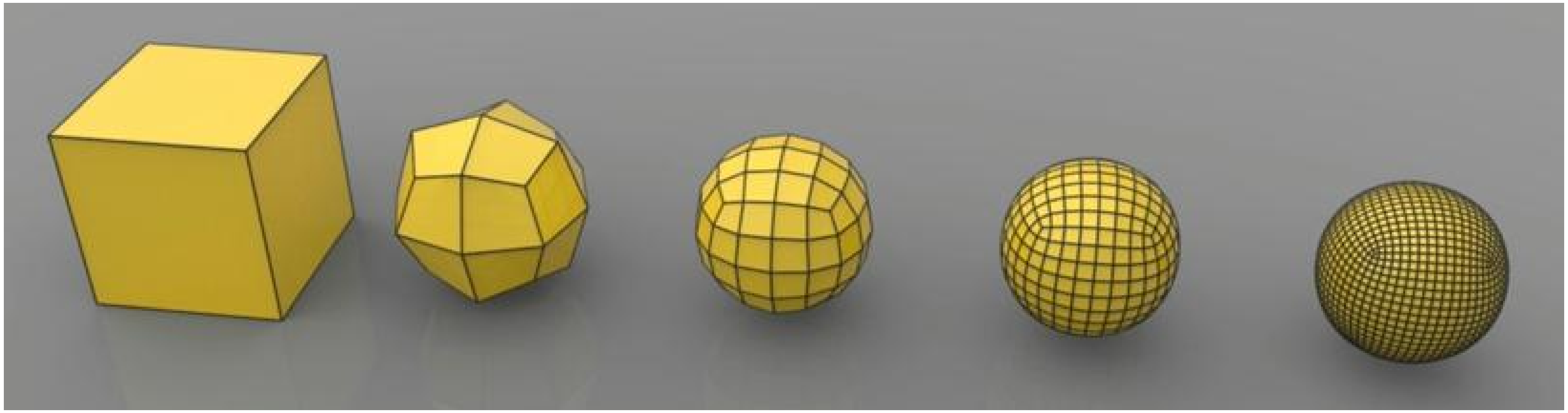
$$\mathbf{d}_1' = \frac{(n-3)}{n}\mathbf{d}_1 + \frac{2}{n}\mathbf{R} + \frac{1}{n}\mathbf{S}$$

$$\mathbf{R} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{E}_i \quad \mathbf{S} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{V}_i$$

- ✓ Let's expand these equations into ready-to-code statements.

Subdivision Surfaces

✓ Catmull-Clark subdivision to refine quad surfaces/meshes.

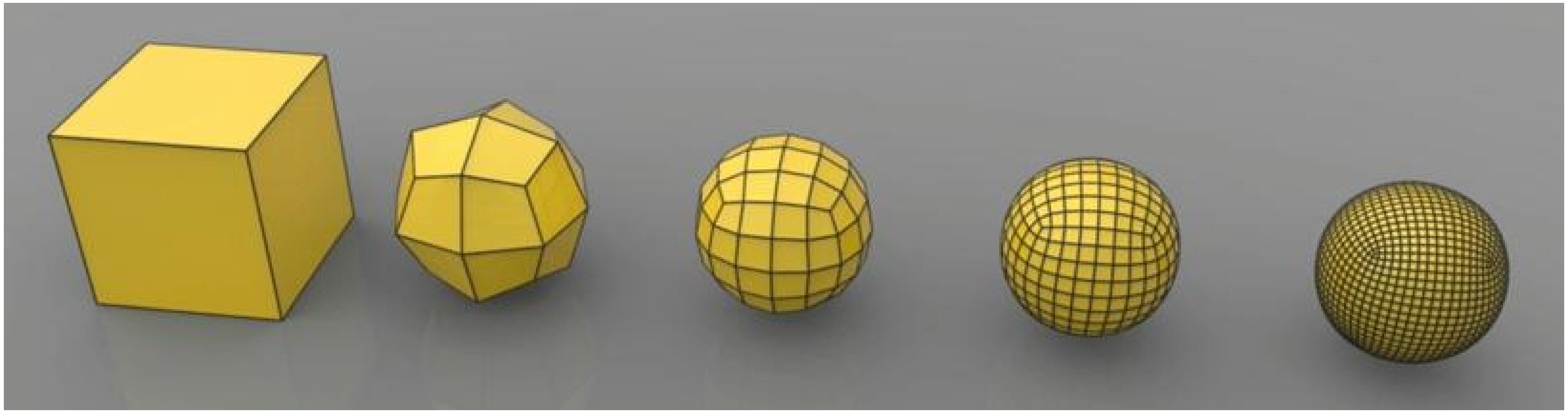


Base mesh

Catmull-Clark limit surface

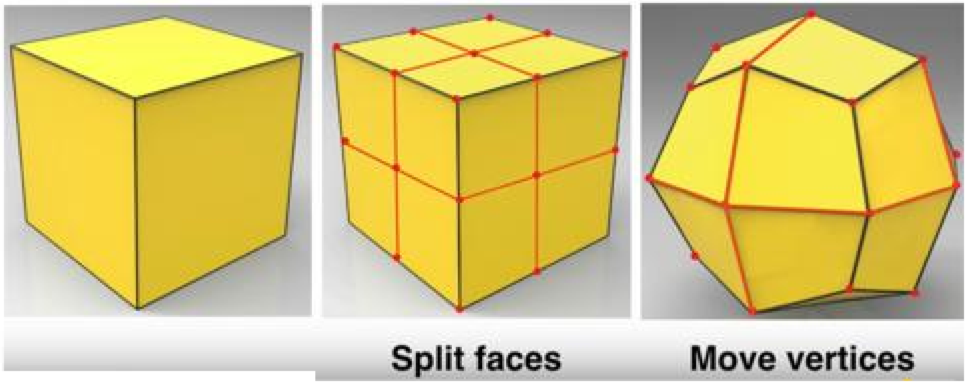
Subdivision Surfaces

✓ Catmull-Clark subdivision to refine quad surfaces/meshes.



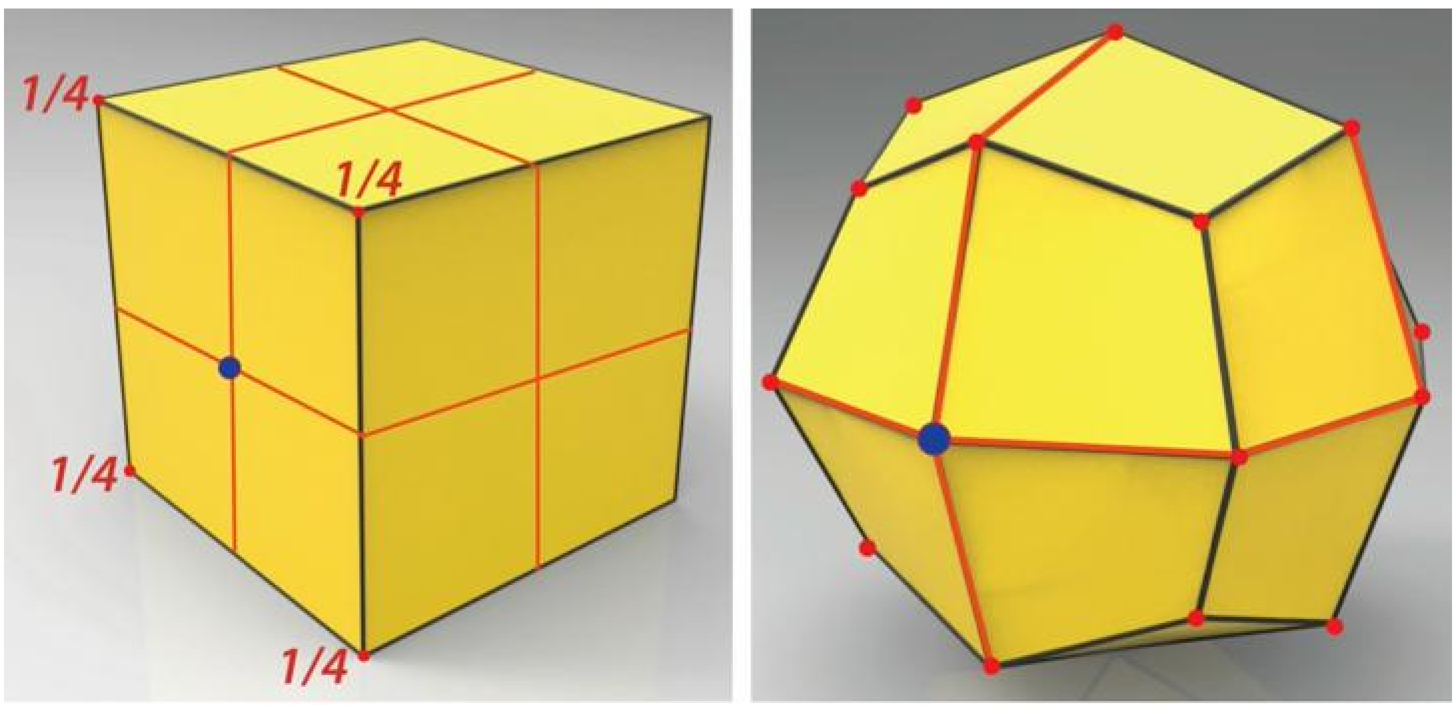
Base mesh

Catmull-Clark limit surface



Subdivision Surfaces

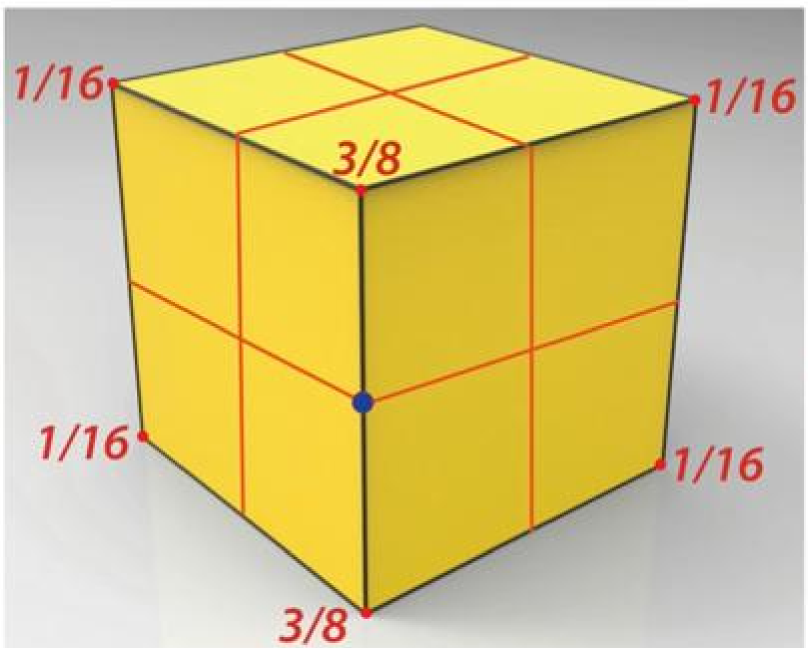
- ✓ Catmull-Clark subdivision to refine quad surfaces/meshes.
- ✓ Weighted averages of the original verts used to compute new geometry.

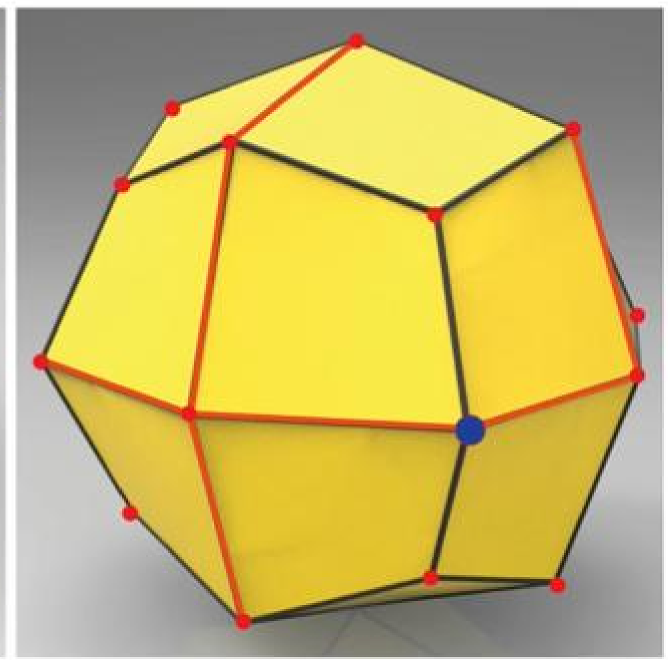


- ✓ A face point is computed using these 4 weights.

Subdivision Surfaces

- ✓ Catmull-Clark subdivision to refine quad surfaces/meshes.
- ✓ Weighted averages of the original verts used to compute new geometry.

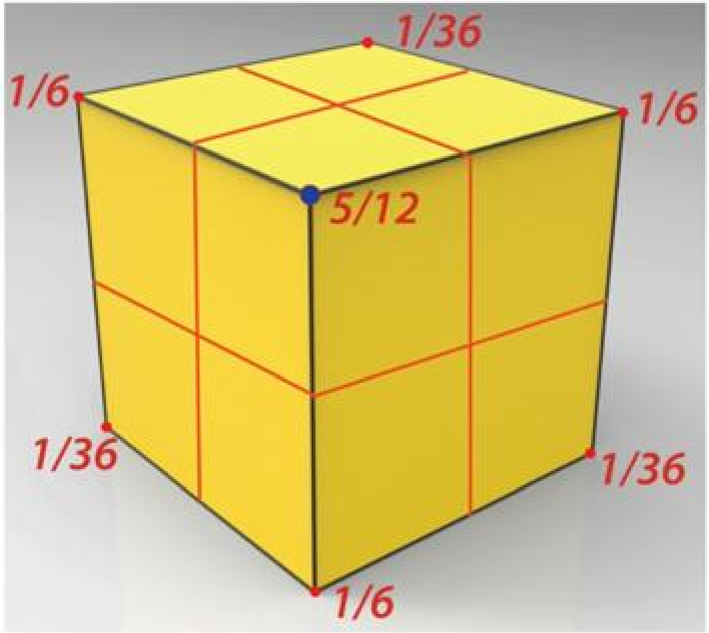


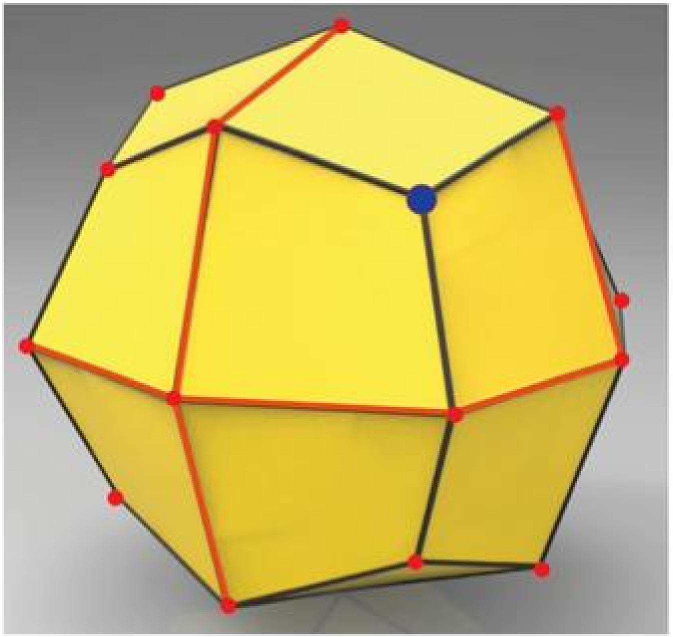


- ✓ An edge point is computed using these 6 weights.

Subdivision Surfaces

- ✓ Catmull-Clark subdivision to refine quad surfaces/meshes.
- ✓ Weighted averages of the original verts used to compute new geometry.

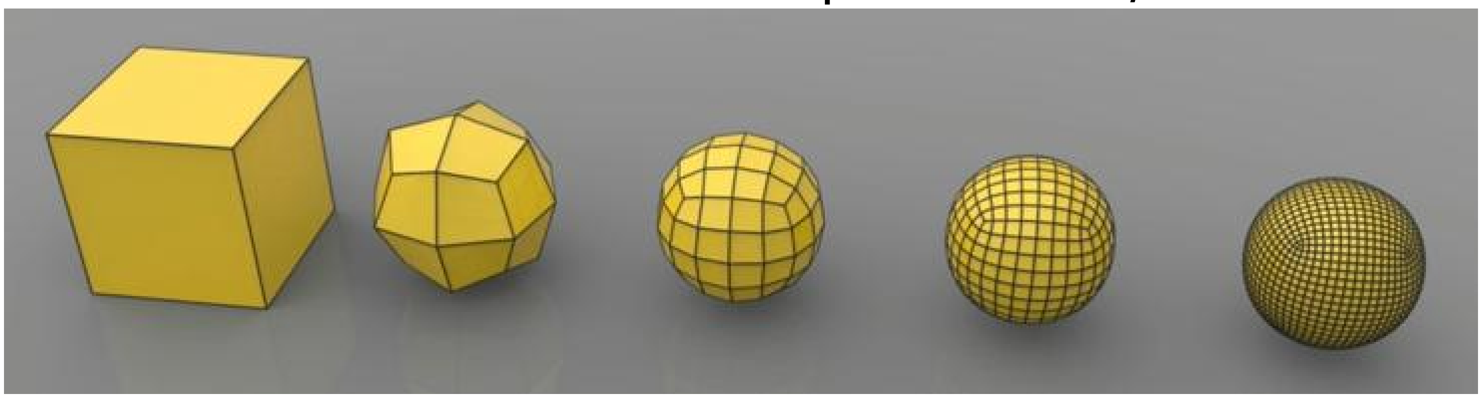


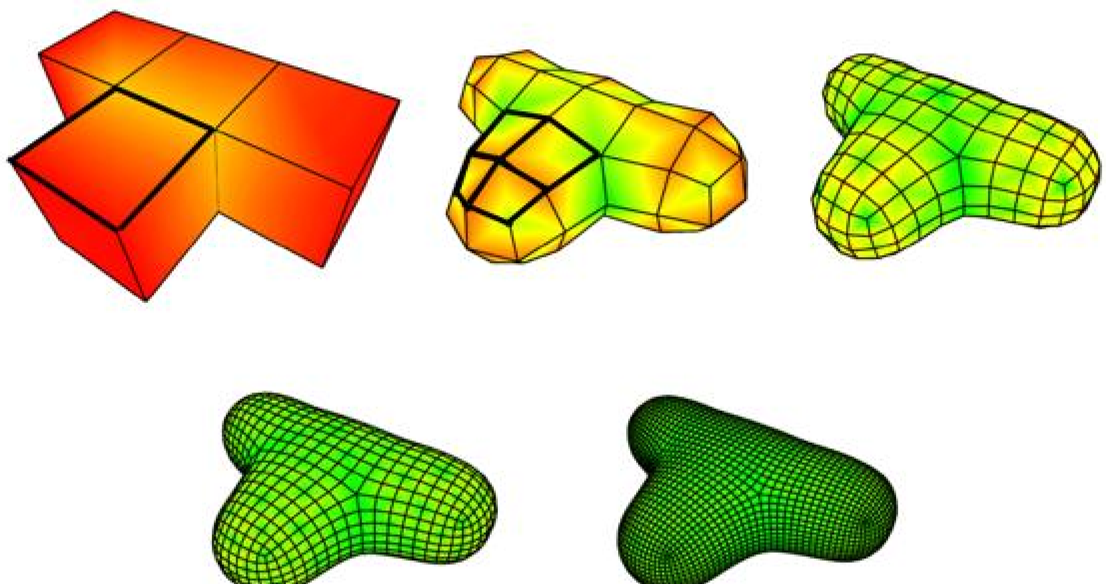


- ✓ A vertex point is computed using these 7 weights.

Subdivision Surfaces

✓ Catmull-Clark subdivision to refine quad surfaces/meshes.





Subdivision Surfaces

- ✓ √3-subdivision to refine triangular surfaces/meshes.
  - ✓ https://www.graphics.rwth-aachen.de/media/papers/sqrt31.pdf

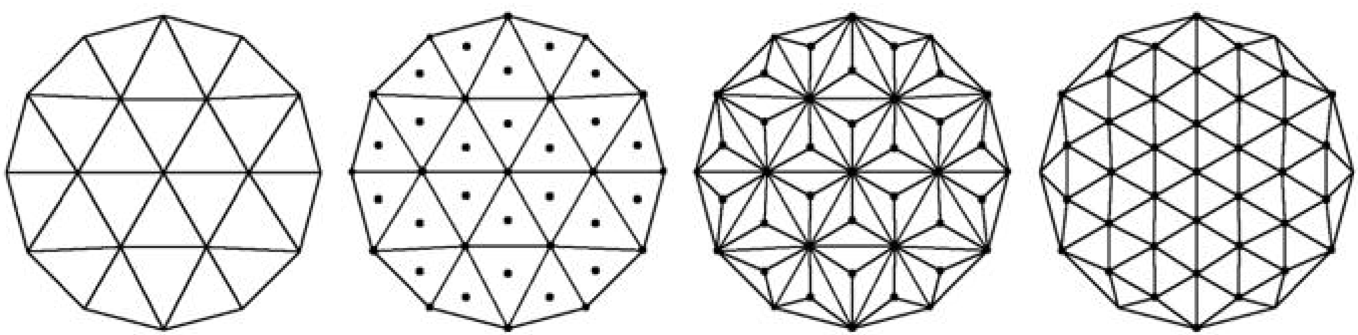


FIGURE 5. $\sqrt{3}$ Subdivision. From left to right: original mesh, added vertices at the midpoints of the faces (step 1), connecting the new points to the original mesh (step 1), flipping the original edges to obtain a new set of faces (step 3). Step 2 involves shifting the original vertices and is not shown.

- ✓ Step 2: move each original vertex v to a new position p by averaging v with the positions of its original neighboring vertices v_i for $0 \le i \le n-1$.

$$\mathbf{p} = (1 - a_n)\mathbf{v} + \frac{a_n}{n}\sum_{i=0}^{n-1}\mathbf{v}_i \qquad a_n = \frac{4 - 2\cos\left(\frac{2\pi}{n}\right)}{9}$$

Subdivision Surfaces

- ✓ √3-subdivision.
- ✓ Loop subdivision.

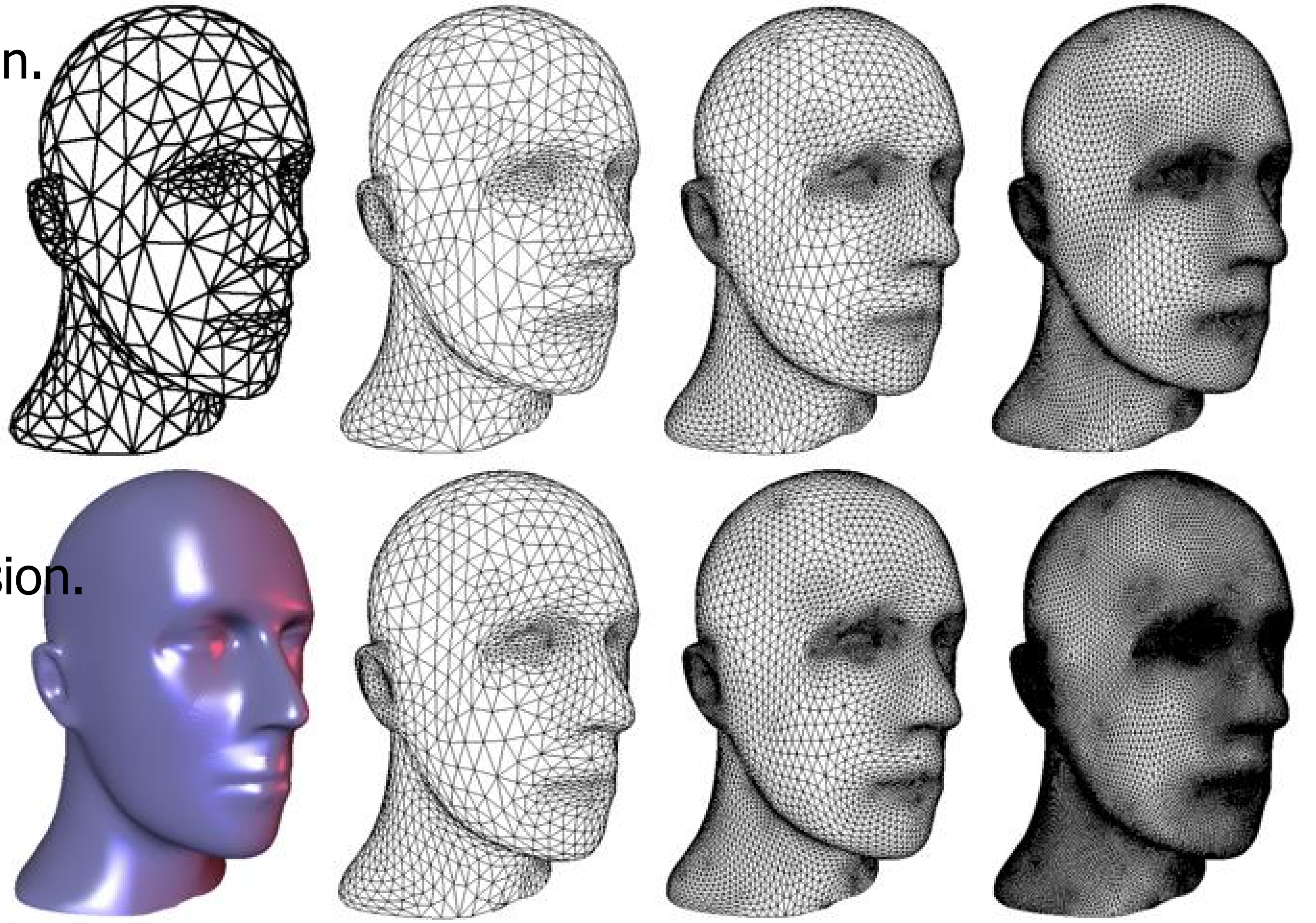
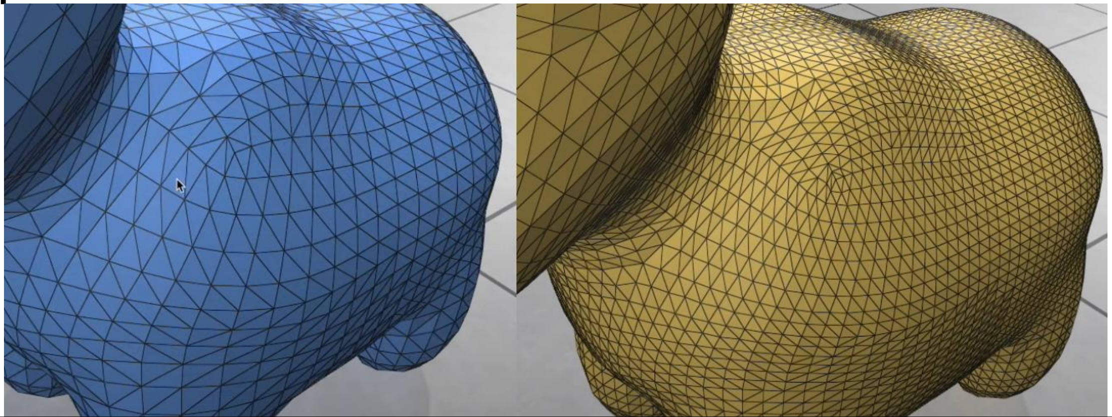


Figure 13: *Sequences of meshes generated by the $\sqrt{3}$-subdivision scheme (top row) and by the Loop subdivision scheme (bottom row). Although the quality of the* **limit** *surfaces is the same (C^2), $\sqrt{3}$-subdivision uses an alternative refinement operator that increases the number of triangles slower than Loop's. The relative complexity of the corresponding meshes from both rows is (from left to right) $\frac{3}{4} = 0.75$, $\frac{9}{16} = 0.56$, and $\frac{27}{64} = 0.42$. Hence the new subdivision scheme yields a much finer gradation of uniform hierarchy levels.*

Subdivision Surfaces

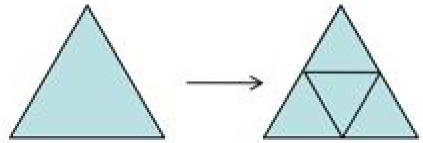
- ✓ 4-to-1 subdivision.
  - ✓ For each edge e=(u,v), which is shared by triangles t1 and t2
    - ✓ Split e from the middle to replace t1 w/ 2 triangles and t2 w/ 2
    - ✓ Add the new edge to set F if it connects original vertex to the new split-vertex (middle) and it is neither u nor v.
  - ✓ Flip each edge in F

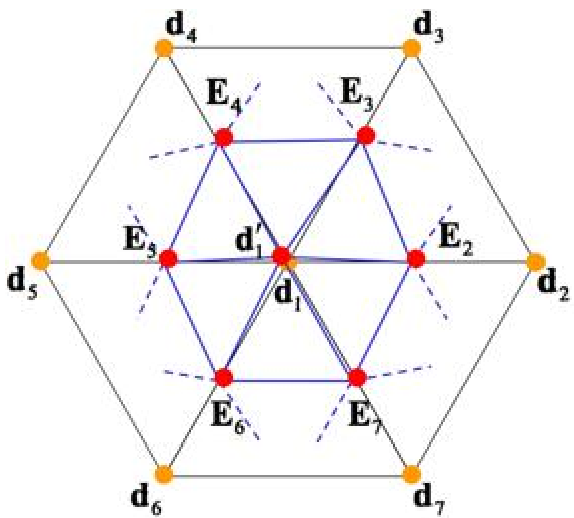
- ✓ An interpolating scheme as the original vertex locations are not updated/approximated.



Subdivision Surfaces

✓ Loop subdivision to refine triangular surfaces/meshes.





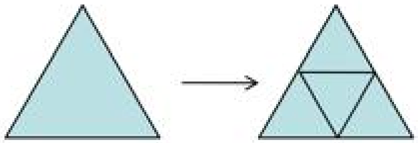
$$E_i = \frac{3}{8}(d_1 + d_i) + \frac{1}{8}(d_{i-1} + d_{i+1})$$

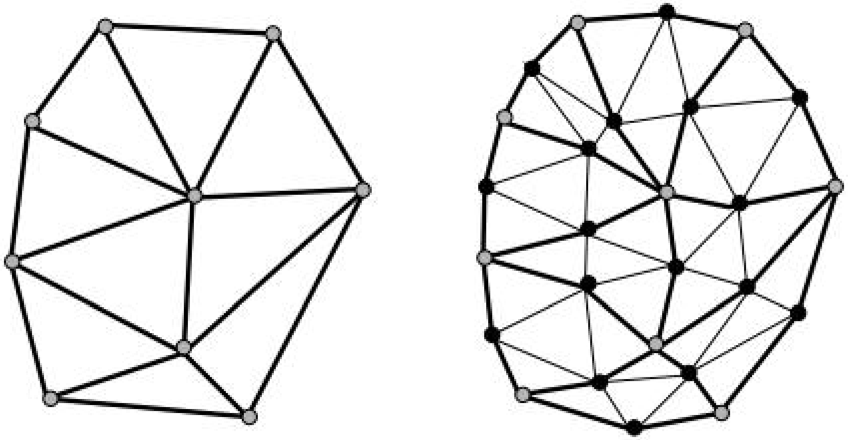
$$\mathbf{d}_1' = \alpha_n \mathbf{d}_1 + \frac{(1-\alpha_n)}{n}\sum_{j=2}^{n+1}\mathbf{d}_j$$

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

Subdivision Surfaces

- ✓ Loop subdivision to refine triangular surfaces/meshes.

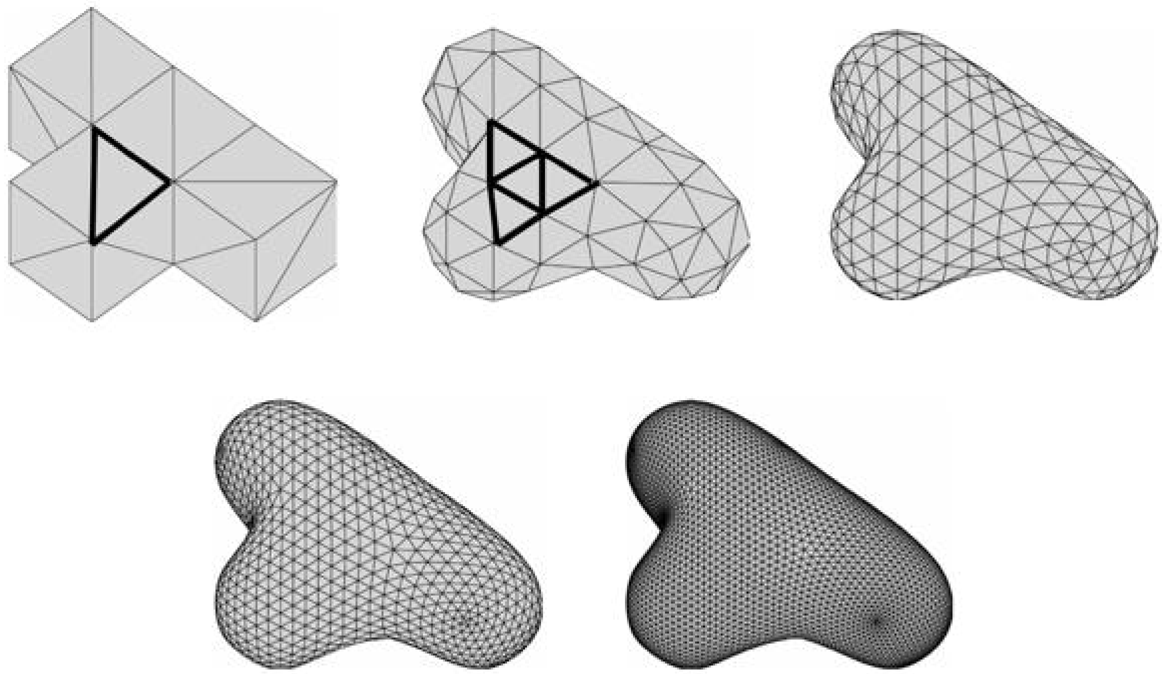




Refinement of a triangular mesh. New vertices are shown as black dots. Each edge of the control mesh is split into two, and new vertices are reconnected to form 4 new triangles, replacing each triangle of the mesh.

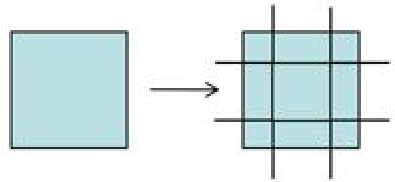
Subdivision Surfaces

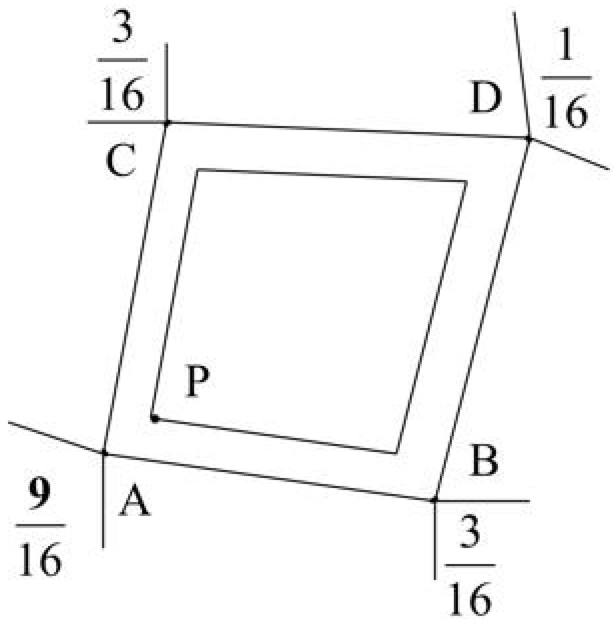
✓ Loop subdivision to refine triangular surfaces/meshes.



Subdivision Surfaces

- ✓ Doo-Sabin subdivision to refine quad surfaces/meshes. Both quad and triangle faces in the output.

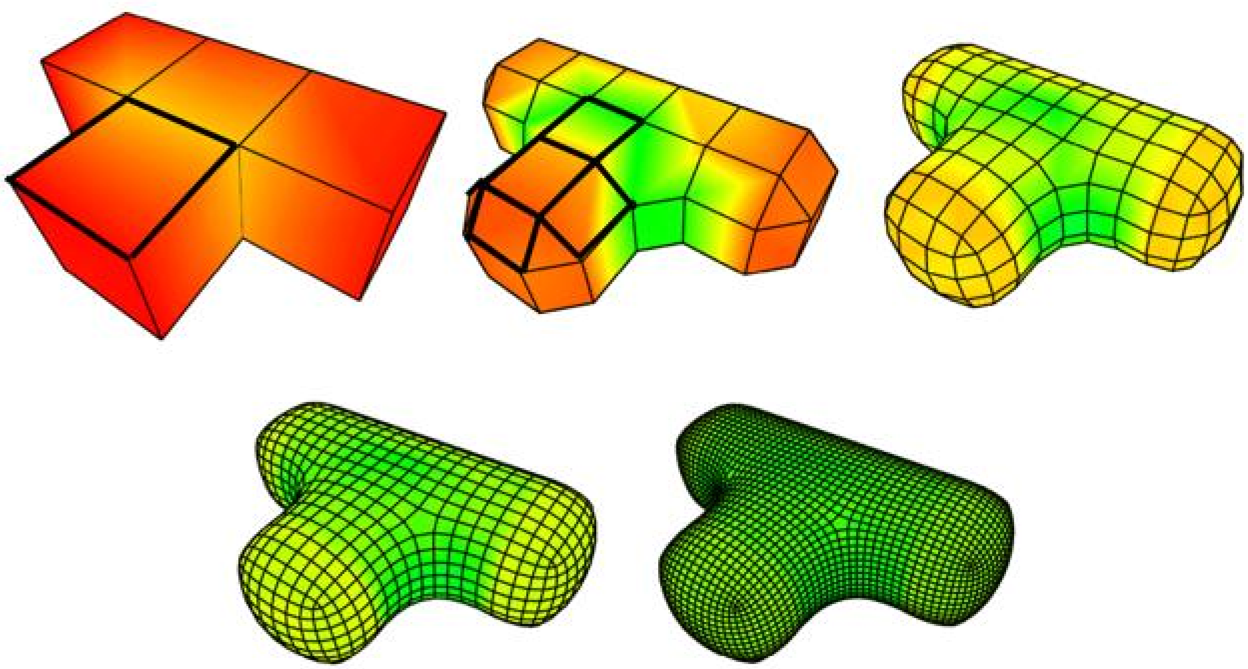




$$P = (9/16)\,A + (3/16)\,B + (3/16)\,C + (1/16)\,D$$

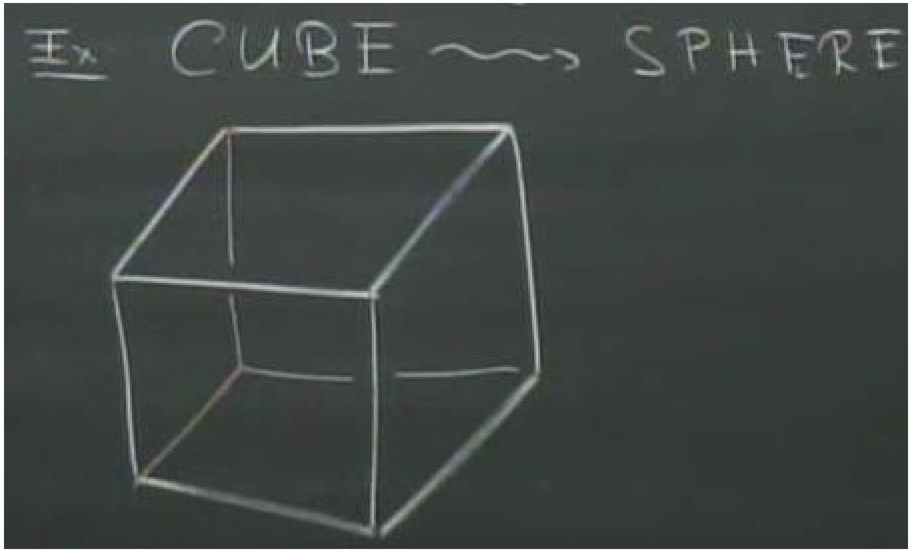
Subdivision Surfaces

- ✓ Doo-Sabin subdivision to refine quad surfaces/meshes. Both quad and triangle faces in the output.



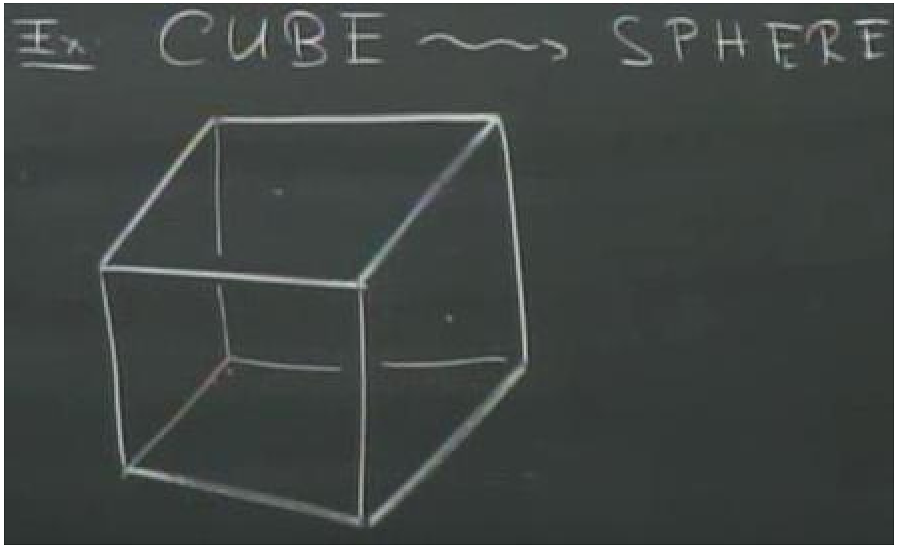
Subdivision Surfaces

✓ Doo-Sabin subdivision scheme; e.g., converts cube to sphere.



Subdivision Surfaces

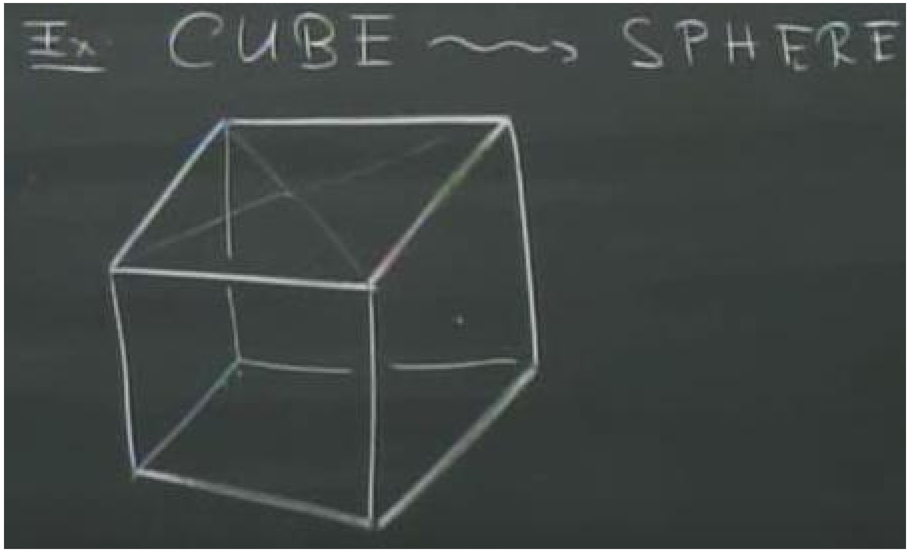
- ✓ Doo-Sabin subdivision scheme; e.g., converts cube to sphere.



- ✓ Take centers of each polygonal face (not has to be quads).

Subdivision Surfaces

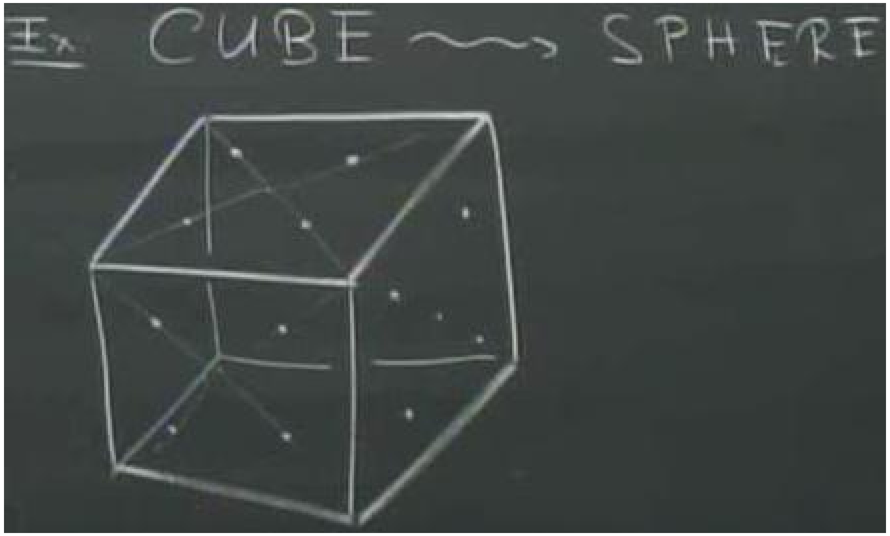
✓ Doo-Sabin subdivision scheme; e.g., converts cube to sphere.



✓ (Virtually) connect centers to the parent polygon's vertices.

Subdivision Surfaces

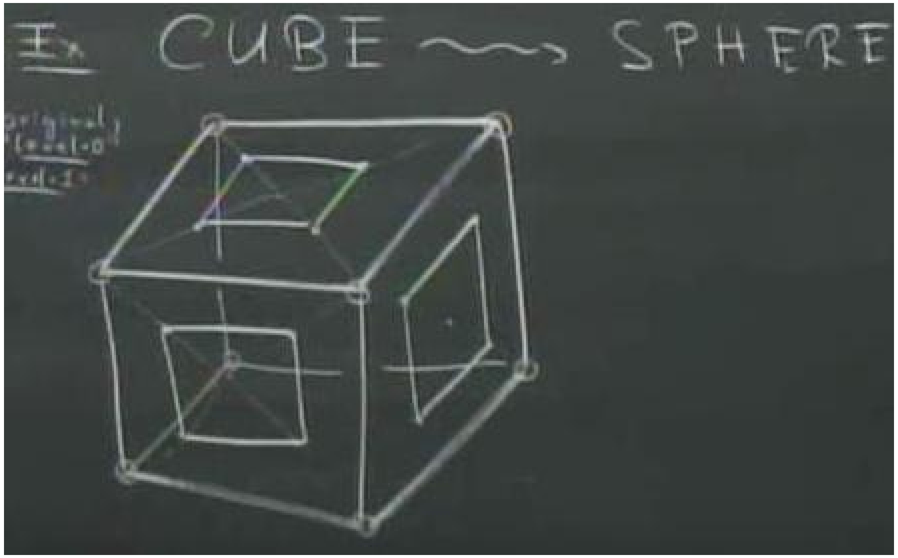
- ✓ Doo-Sabin subdivision scheme; e.g., converts cube to sphere.



- ✓ Take midpoints of the virtual connectors.
- ✓ All these midpnts are the next generation of pnts generated by base.

Subdivision Surfaces

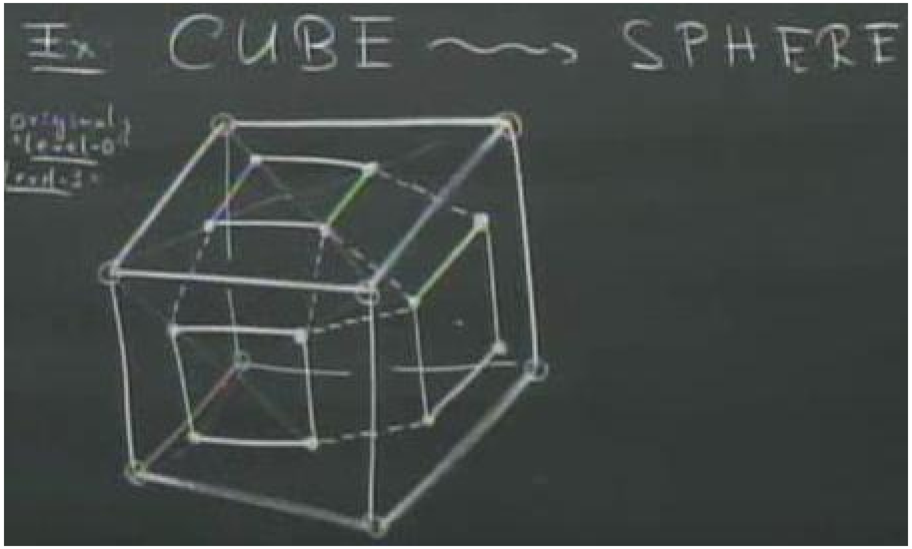
- ✓ Doo-Sabin subdivision scheme; e.g., converts cube to sphere.



- ✓ Connect this new generation as shown above (creates axis-aligned quads).

Subdivision Surfaces

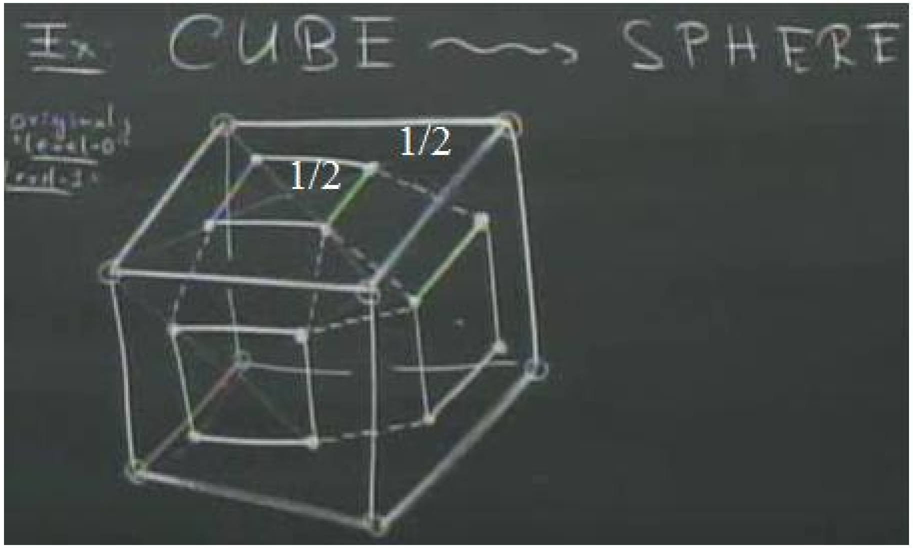
- ✓ Doo-Sabin subdivision scheme; e.g., converts cube to sphere.



- ✓ More connections (creates slopped triangles; called corner-cutting).
- ✓ Other side not shown for visual clarity. Continue for a refined sphere.

Subdivision Surfaces

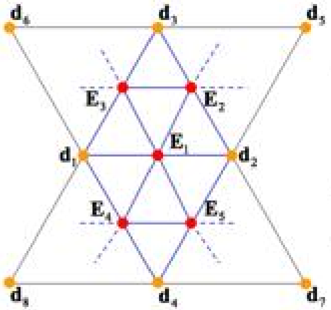
- ✓ Doo-Sabin subdivision scheme; e.g., converts cube to sphere.



- ✓ Certain combinations of the surrounding vertices: subdivision mask.
- ✓ Subdivision mask decides the new vertex coordinates.
- ✓ Here (1/2, 1/2) mask is used 'cos midpnts are selected.

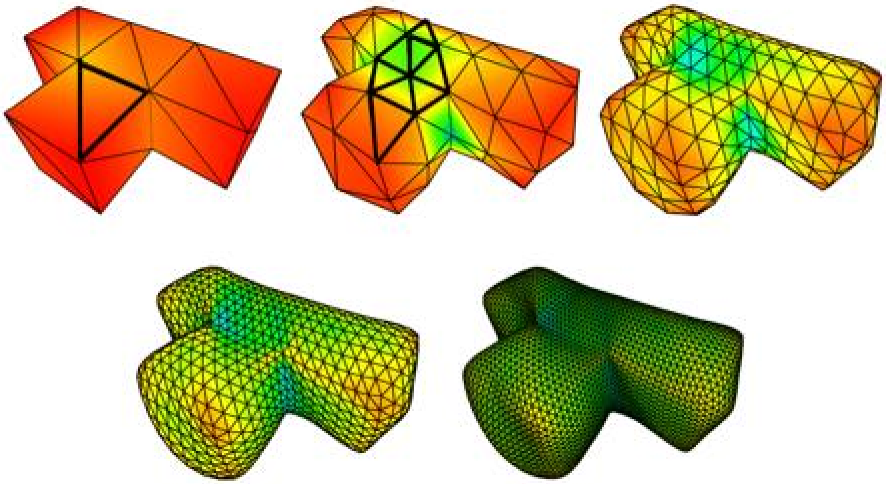
Subdivision Surfaces

- ✓ Butterfly subdivision to refine triangular surfaces/meshes (interpolting).



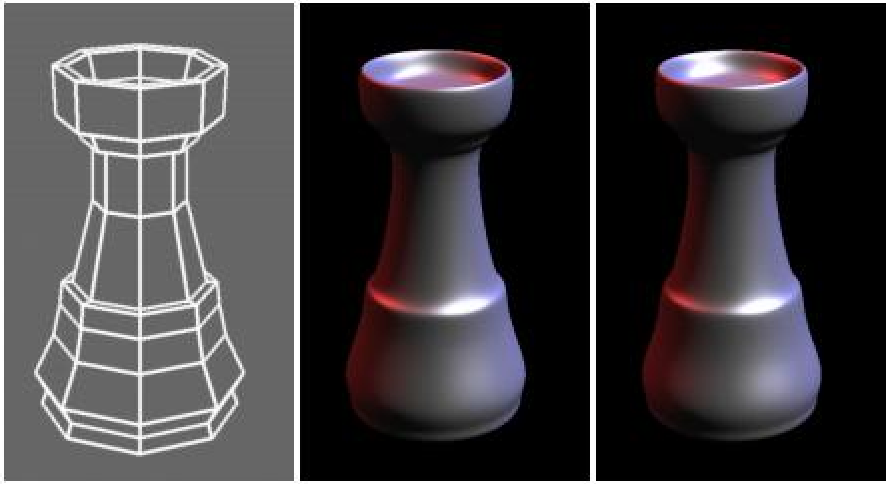
$$E_1 = \frac{1}{2}(d_1 + d_2) + \omega(d_3 + d_4) - \frac{\omega}{2}(d_5 + d_6 + d_7 + d_8)$$

$$d_i' = d_i$$



Subdivision Surfaces

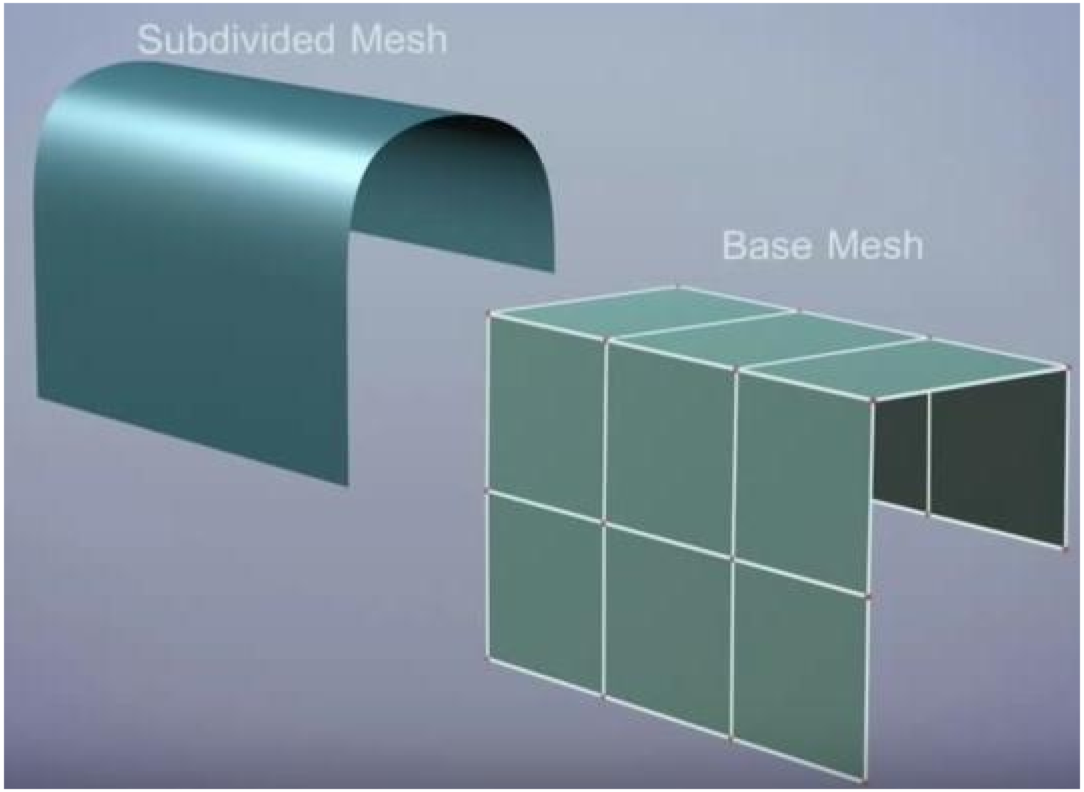
- ✓ Related to remeshing.
- ✓ Sub-d surface: a method to represent a smooth surface via a coarser polygon mesh (base mesh).
- ✓ Loop subdivision scheme by Charles Loop, 1987.



Initial mesh *Loop* *Catmull-Clark*

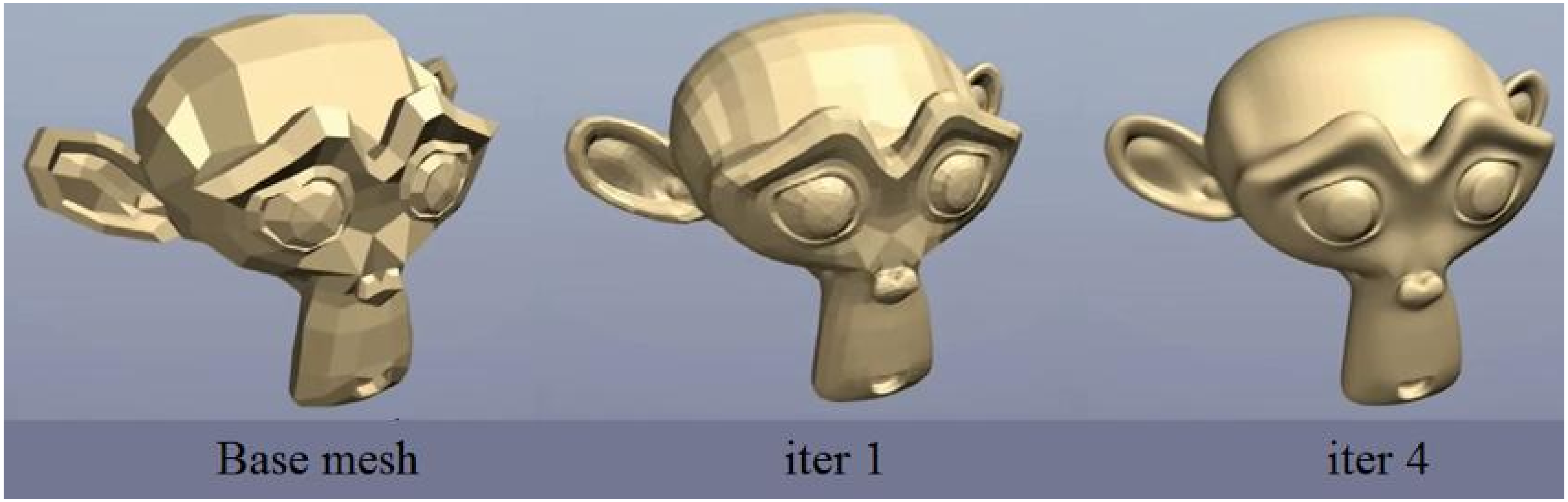
Subdivision Surfaces

- ✓ Related to remeshing.
- ✓ Sub-d surface: a method to represent a smooth surface via a coarser polygon mesh (base mesh).



Subdivision Surfaces

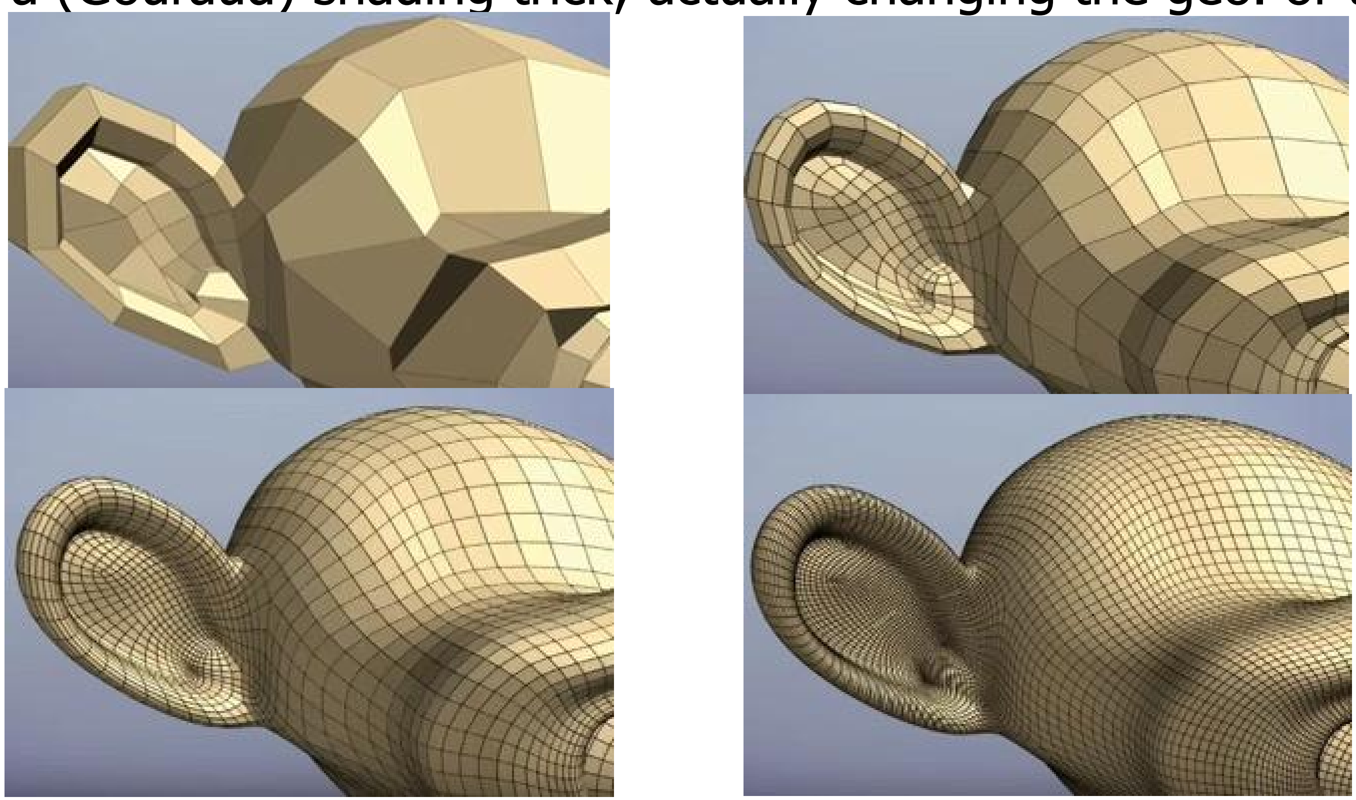
- ✓ Related to remeshing.
- ✓ Sub-d surface: a method to represent a smooth surface via a coarser polygon mesh (base mesh).



- ✓ Like progressive meshes you can use it for efficient distance-dependent rendering (LOD).

Subdivision Surfaces

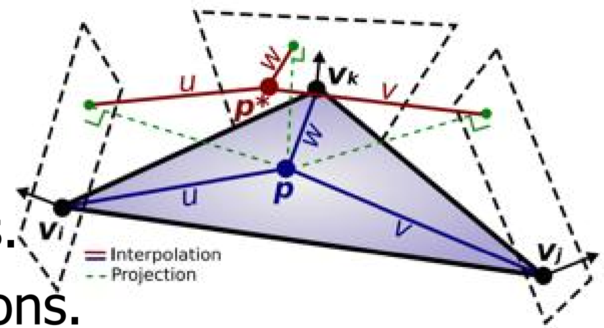
- ✓ Related to remeshing.
- ✓ Sub-d surface: a method to represent a smooth surface via a coarser polygon mesh (base mesh).
- ✓ Not a (Gouraud) shading trick; actually changing the geo. of the model.



Subdivision Surfaces

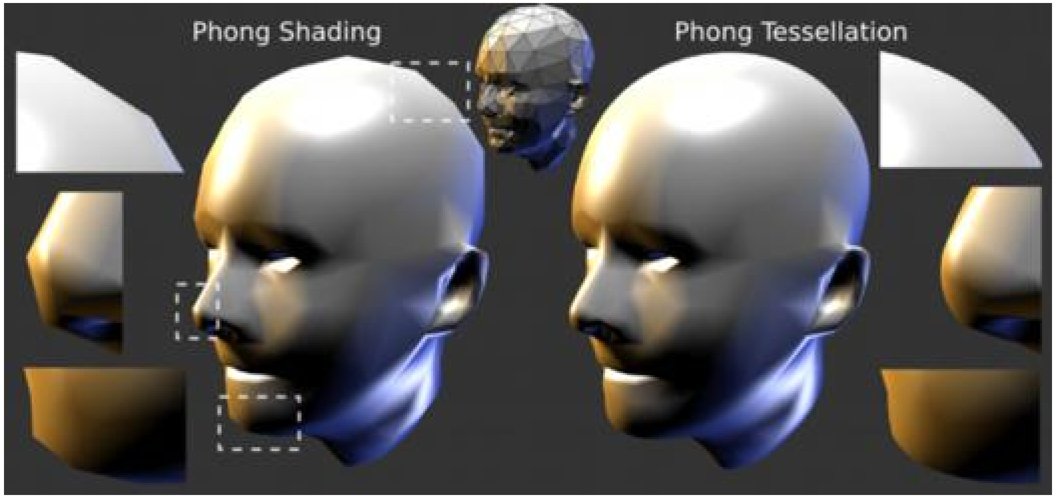
- ✓ Related to remeshing.
- ✓ Sub-d surface: a method to represent a smooth surface via a coarser polygon mesh (base mesh).
- ✓ Phong Tessellation, 2008, is an interpolating subdivision and does not exhibit the shrinking effects of the approximating refinement schemes, e.g., Loop or Catmull-Clark subdivision.
- ✓ Idea: use tangent planes at mesh vertices to replace triangles with quadratic patches for smoother display.

1. Compute linear tessellation.
2. Project the resulting point orthogonally onto the 3 tangent planes defined by the triangle verts.
3. Compute barycentric interp. of these projections.



Subdivision Surfaces

- ✓ Related to remeshing.
- ✓ Sub-d surface: a method to represent a smooth surface via a coarser polygon mesh (base mesh).
- ✓ Phong Tessellation, 2008, is an interpolating subdivision and does not exhibit the shrinking effects of the approximating refinement schemes, e.g., Loop or Catmull-Clark subdivision.
- ✓ Idea: use tangent planes at mesh vertices to replace triangles with quadratic patches for smoother display.



Potential Project Topics

- ✓ Normal orientation correction: naïve algorithm (neighbor triangles have similar normals) vs. simple this algorithm:
  - ✓ https://www.cs.utah.edu/~ladislav/takayama14simple/takayama14simple.html
- ✓ Mesh repairing using http://www.cgal.org/gsoc/2012.html#holefill
  - ✓ 3D Printing Slides 70-74 also useful.
- ✓ Implementing: Interactive Geometry Remeshing //parameterization-based remeshing.
- ✓ Implementing: Isotropic Surface Remeshing without Large and Small Angles, or Isotropic Surface Remeshing using Constrained Centroidal Delaunay Mesh //surface-based remeshing.