CENG 789 – Digital Geometry Processing

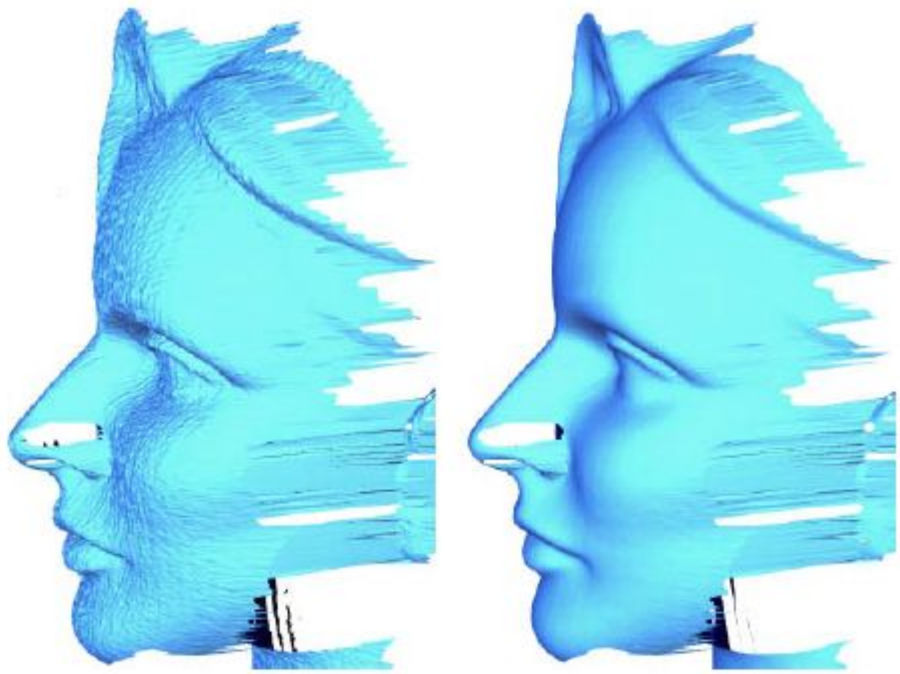
05- Smoothing and Remeshing

Asst. Prof. Yusuf Sahillioğlu

Computer Eng. Dept, MIDDLE EAST TECHNICAL UNIVERSITY, Turkey

Mesh smoothing

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



Mesh smoothing

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

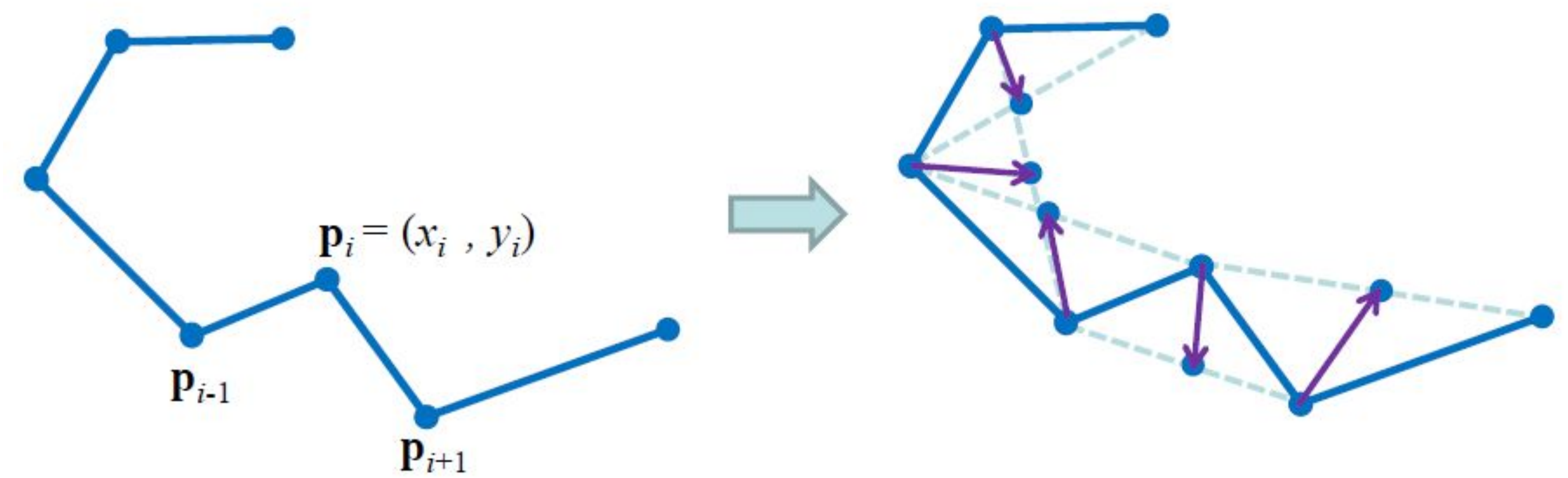
$$L_U(v) = \left(\frac{1}{n}\sum_i v_i\right) - v$$

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

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

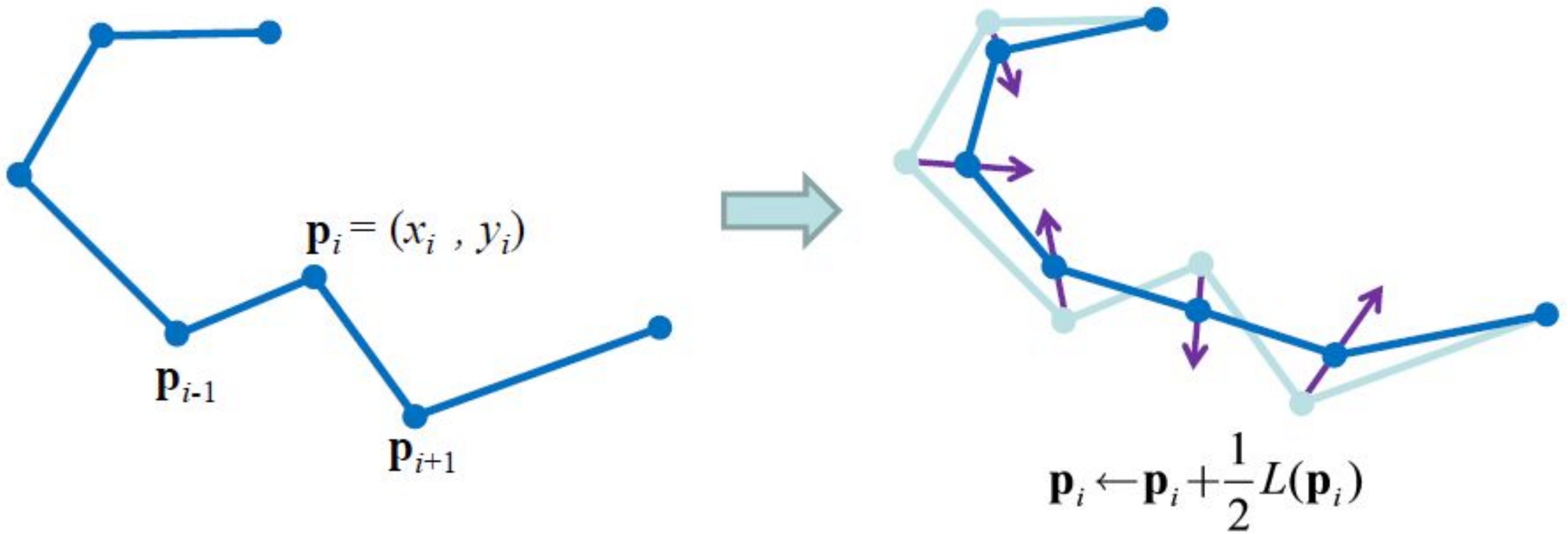
Mesh smoothing

- ✔ Illustration in 1D:



Mesh smoothing

✔ Illustration in 1D:



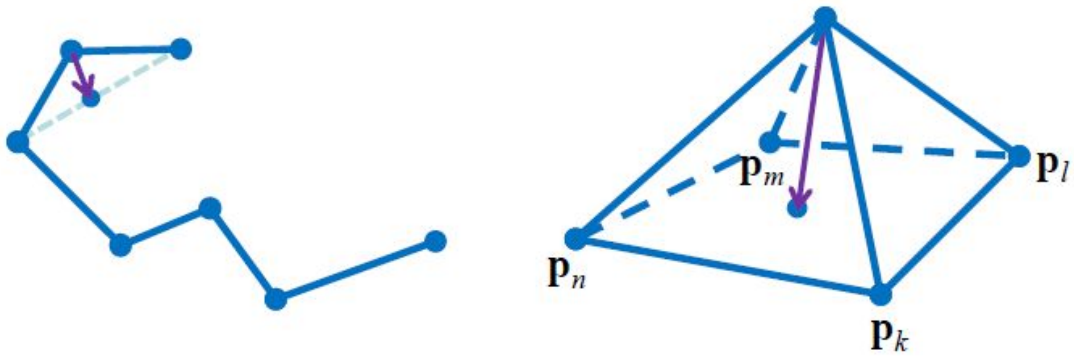
Mesh smoothing

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

Mesh smoothing

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

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



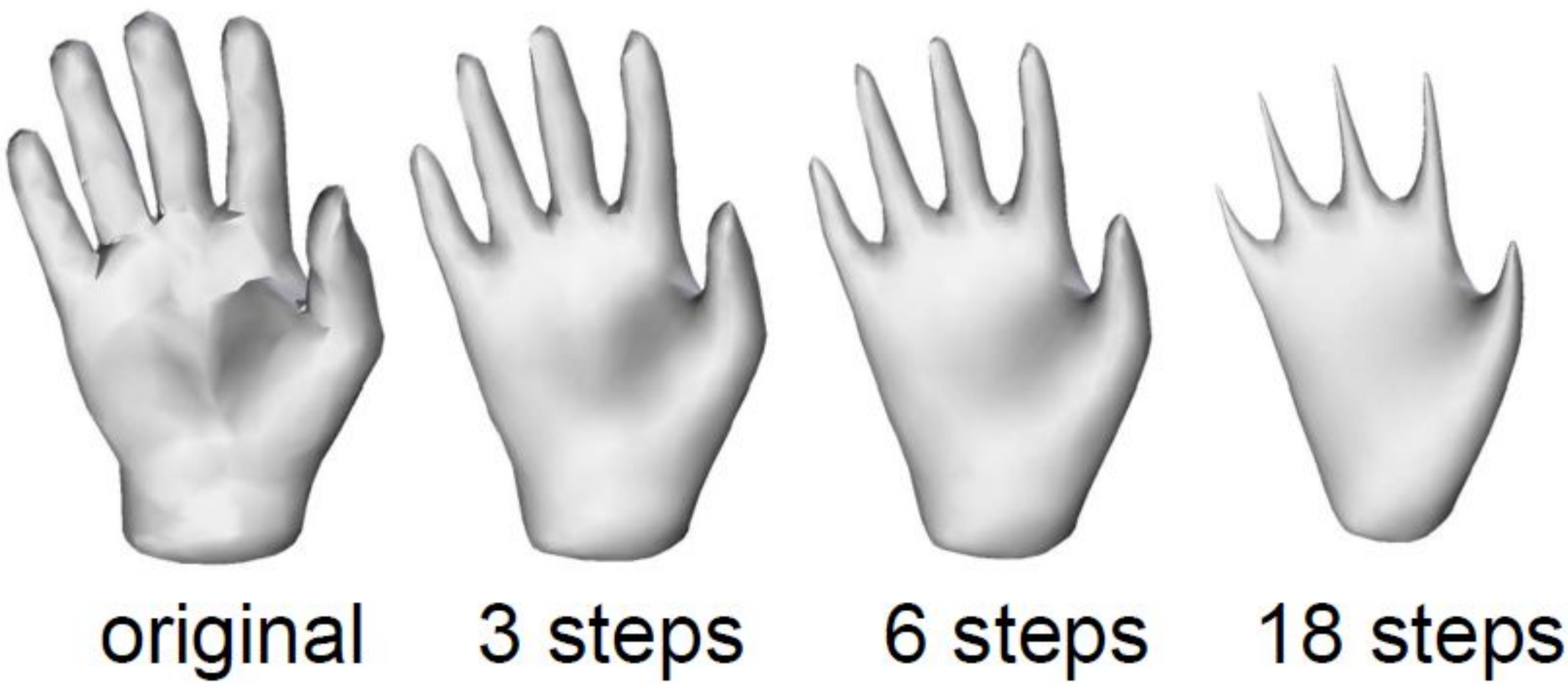
$$\Delta \mathbf{p}_i = \frac{1}{|N_i|}\left(\sum_{j \in N_i} \mathbf{p}_j\right) - \mathbf{p}_i$$

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

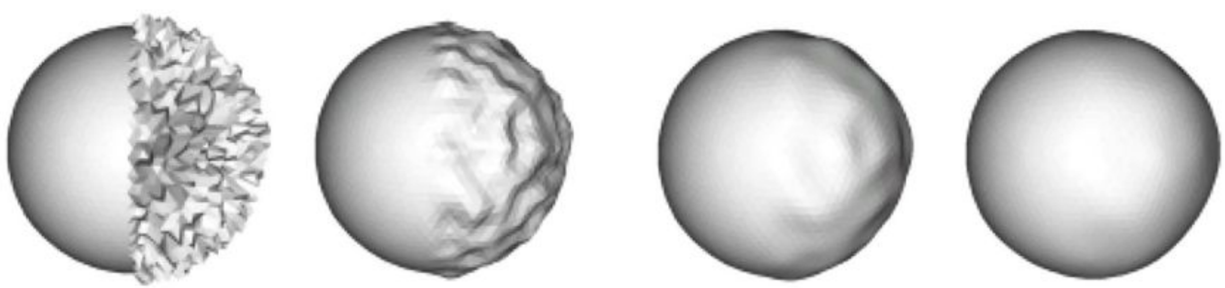
- ✔ Solution: shrinkage problem is remedied with an inflation term.
- ✔ This is introduced by the Mesh Fairing paper by Taubin in 1995.

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 $\mu < 0$



Remeshing

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

- ✔ In contrast to mesh repair (next class), the input of remeshing algorithms is usually assumed to be a manifold triangle mesh.

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

Remeshing

✔ Mesh elements: triangle vs. quadrangle.

Triangle

A triangle is the simplest polygon that is made up of three sides or edges connected by three vertices, making a three sided face. When modeling, triangles are typically a polygon type often avoided.

Triangles tend to pose a problem when subdividing geometry to increase resolution and when a mesh will be deformed or animated.

N-Gon

An n-gon is a polygon that is made up of five or more sides or edges connected by five or more vertices. It's important to keep in mind a n-gon is typically related to a five sided polygon, but it's not limited to just five sides.

A n-gon should always be avoided, they often pose problems at render time, when texturing and especially when deforming for animation.

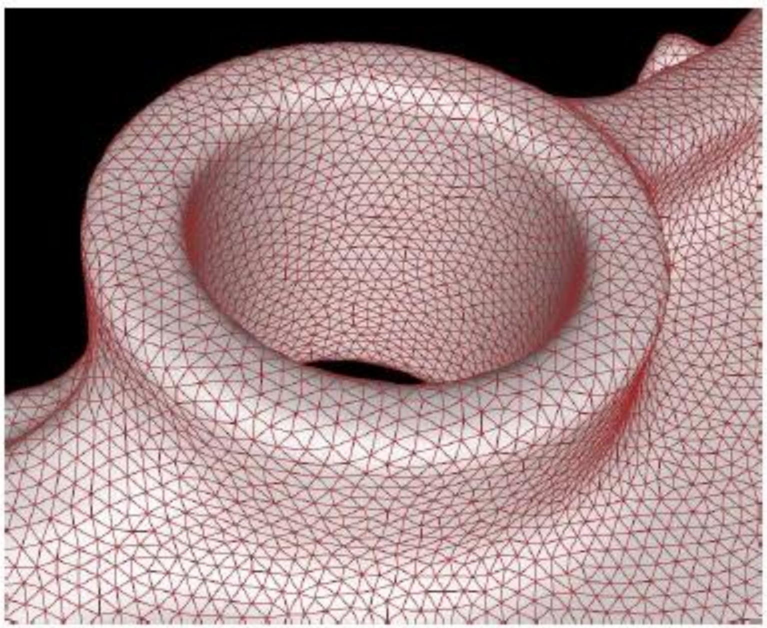
Quadrilaterals

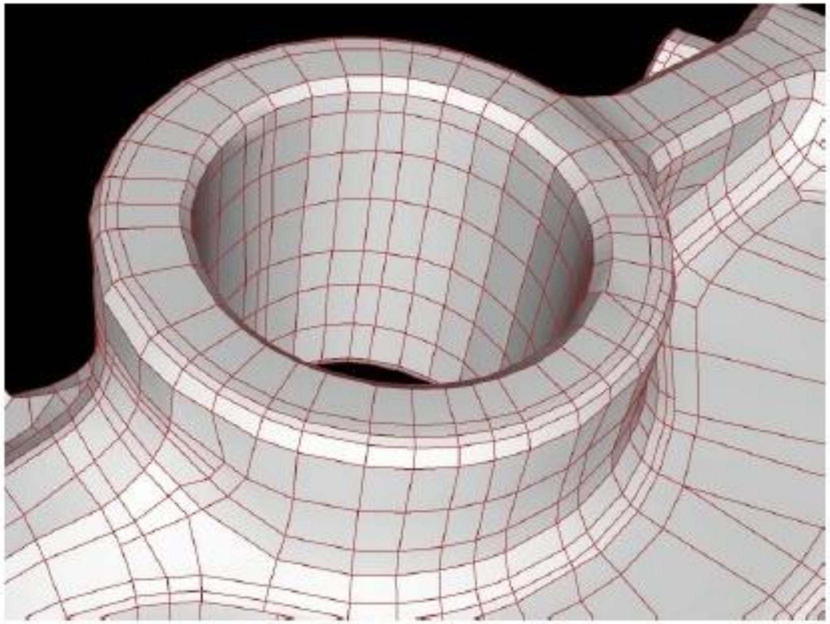
A quad is a polygon made up of four sides or edges that are connected by four vertices, making a four sided face. Quads are the polygon type that you'll want to strive for when creating 3D models.

Quads will ensure your mesh has clean topology and that your model will deform properly when animated.

Remeshing

✔ Mesh elements: triangle vs. quadrangle.





cleaner

edge directions not messy

Remeshing

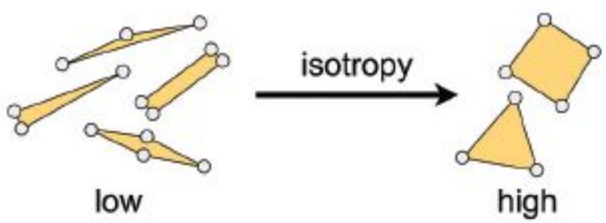
- ✔ Mesh elements: triangle vs. quadrangle.
- ✔ Favoring triangle meshes.
  - ✔ Four points or more may not be on the same plane, but three points always are (ignoring collinearity). This has the interesting property that scalar values vary linearly over the surface of the triangle.
  - ✔ This, in turn, means most if not all of what's needed to shade, texture map, and depth filter a triangle can be calculated using linear interpolation which can be done extremely fast in specialized hardware.
  - ✔ 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 else than triangles.

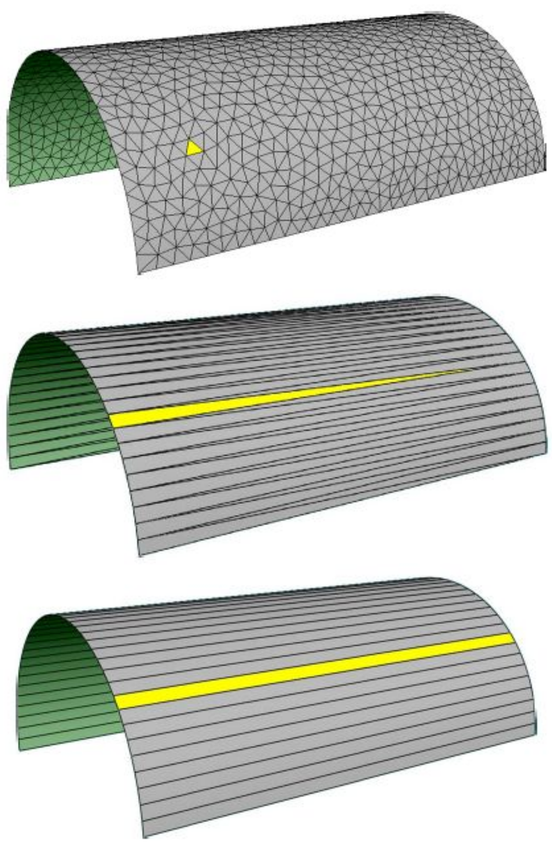
Remeshing

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

Remeshing

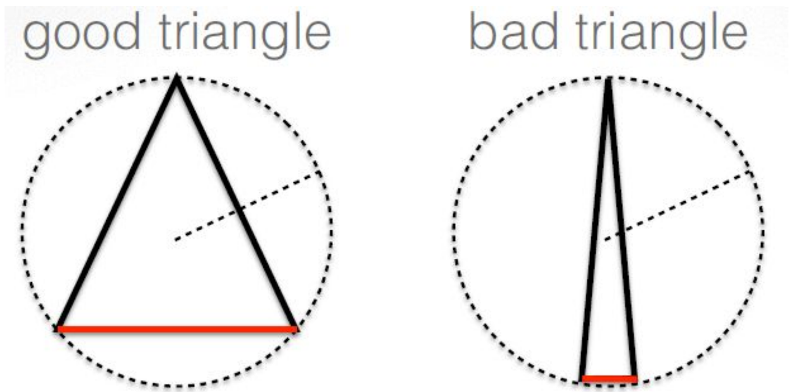
- ✔ Mesh elements: shape: isotropic vs. anisotropic.
- ✔ The shape of isotropic elements is locally uniform in all directions. Ideally, a triangle/quadrangle is isotropic if it is close to equilateral/square.

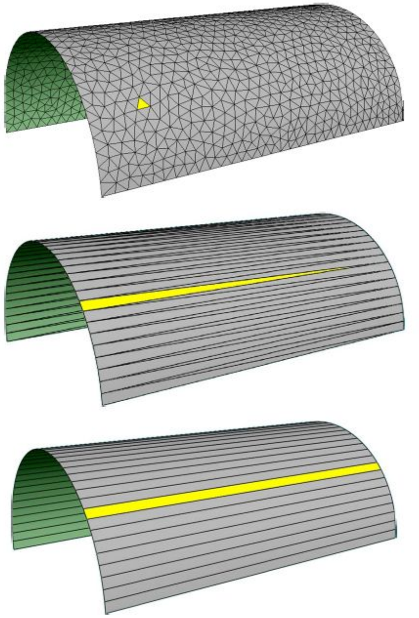




Remeshing

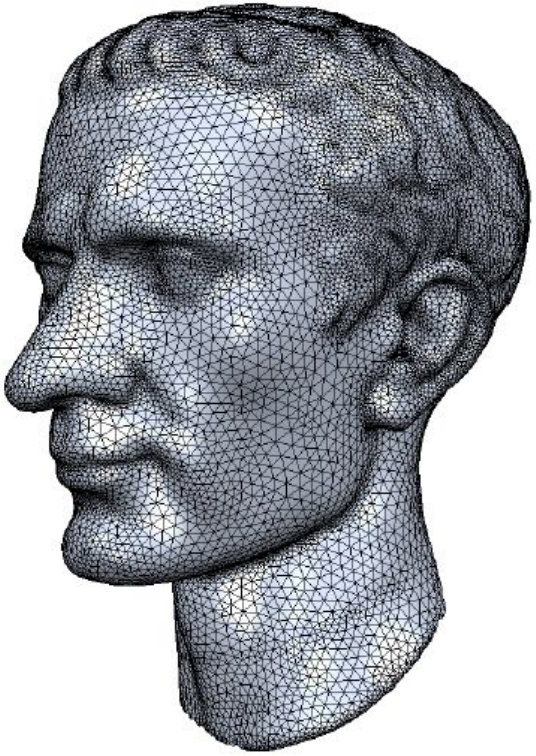
- Mesh elements: shape: isotropic vs. anisotropic.
- The shape of isotropic elements is locally uniform in all directions. Ideally, a triangle/quadrangle is isotropic if it is close to equilateral/square.
- Isotropic elements: roundness ~ 1. (favored in numerical apps, FEM).
- Roundness: ratio of circumcircle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





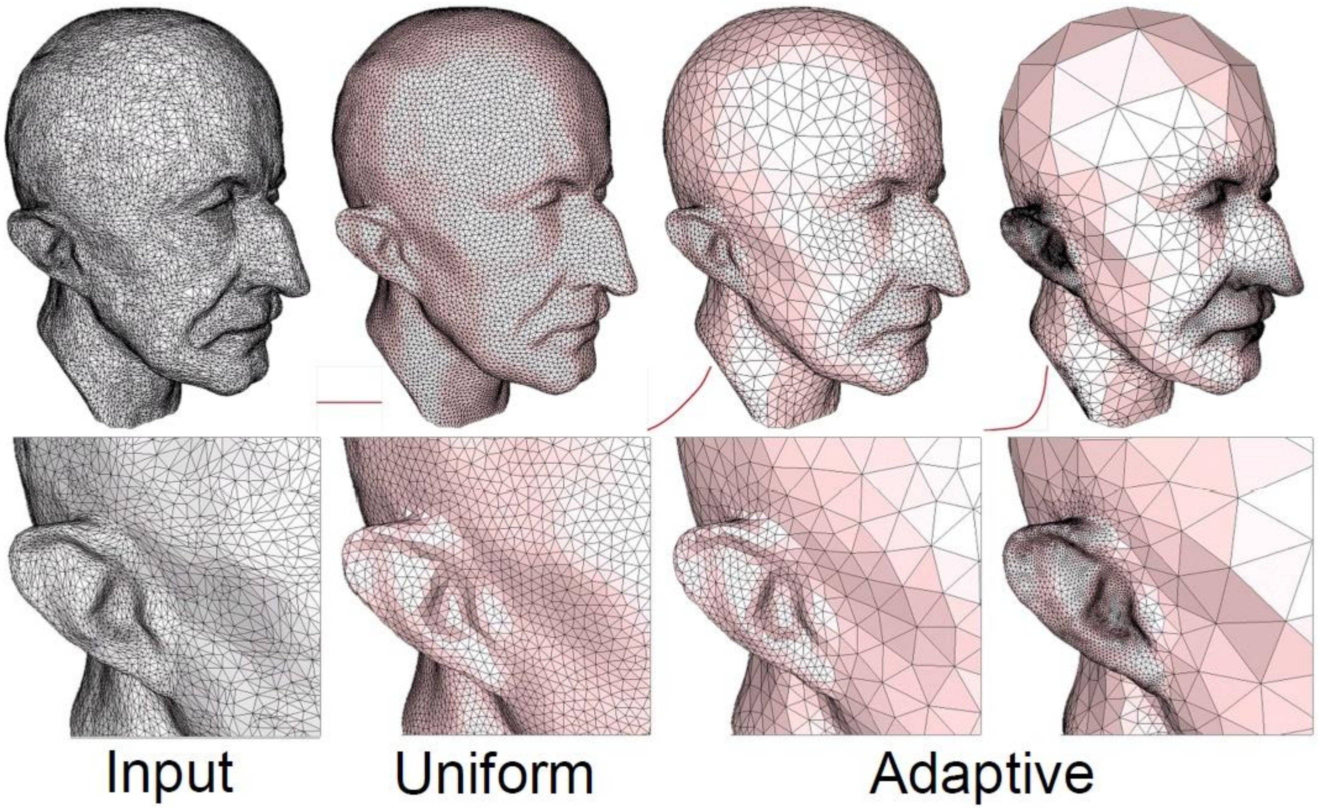
Remeshing

- ✔ Mesh elements: sampling: uniform vs. adaptive.
- ✔ Smaller elements are assigned to areas w/ high curvature.



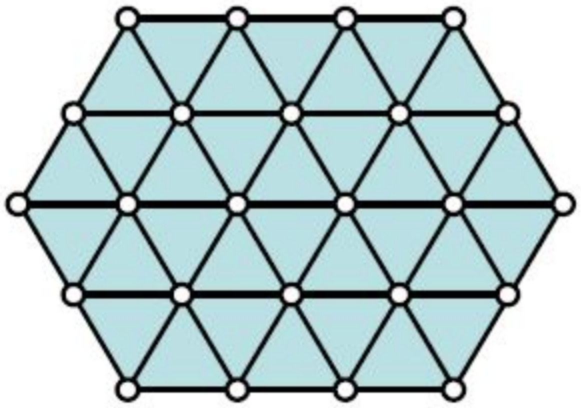
Remeshing

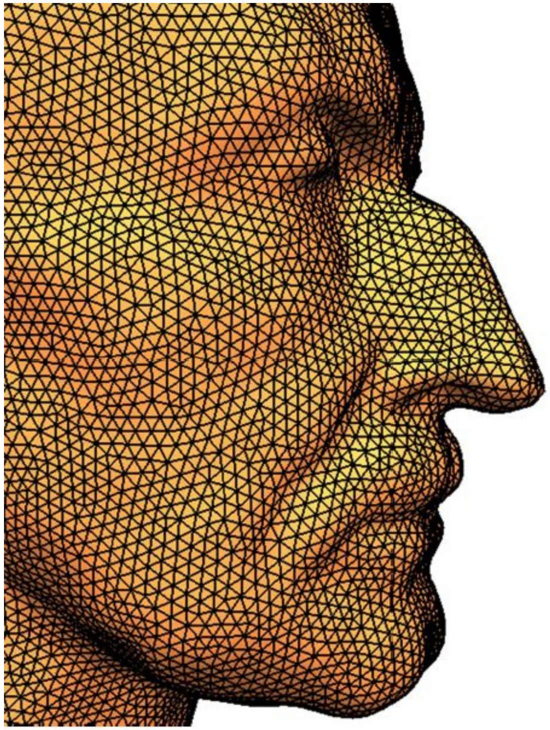
- ✔ Mesh elements: sampling: uniform vs. adaptive.
- ✔ Smaller elements are assigned to areas w/ high curvature.



Remeshing

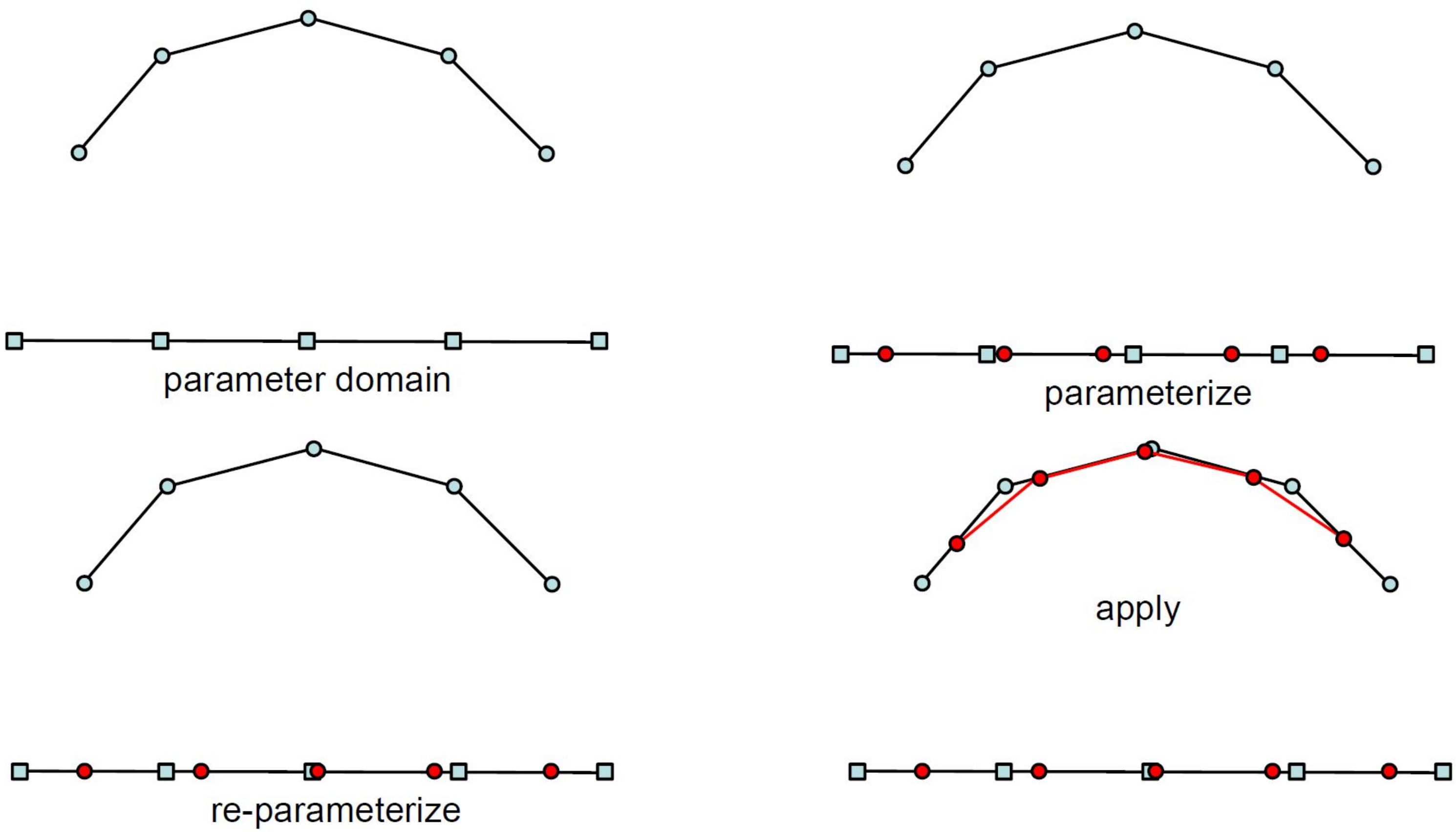
- ✔ Mesh elements: regularity: irregular vs. regular.
- ✔ Valence close to 6; ~equal edge lengths.





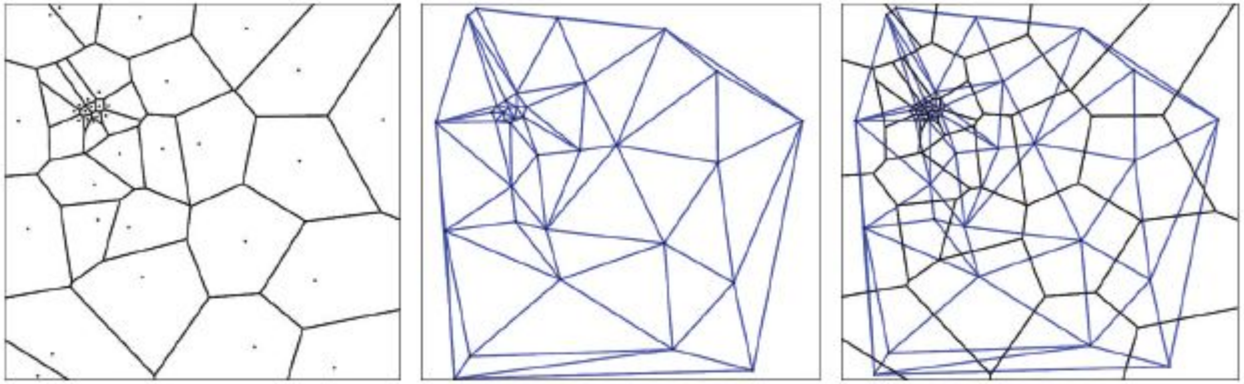
Remeshing

- ✔ Remeshing approaches: parametrization-based vs. surface-based.
- ✔ Param-based: map to 2D domain, do the remeshing (2d problem), lift up.



Remeshing

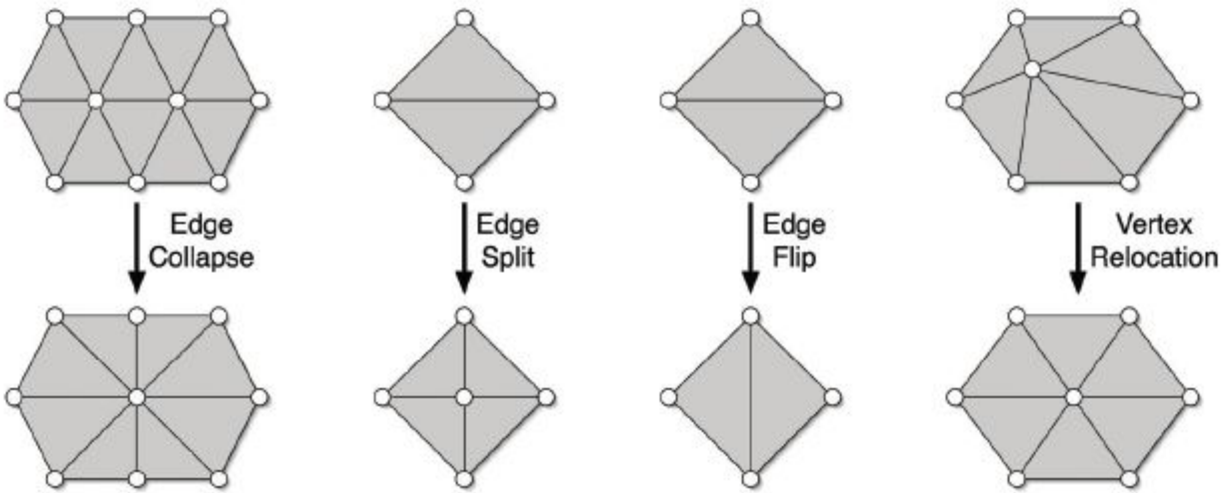
- ✔ Remeshing approaches: parametrization-based vs. surface-based.
- ✔ Param-based: map to 2D domain, do the remeshing (2d problem), lift up.
- ✔ Delaunay triangulation: maximize the min angle = no point in circumcircle of a triangle.



$$V(\mathbf{p}_i) = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x} - \mathbf{p}_i\| \leq \|\mathbf{x} - \mathbf{p}_j\|, \forall j \neq i\}.$$

Remeshing

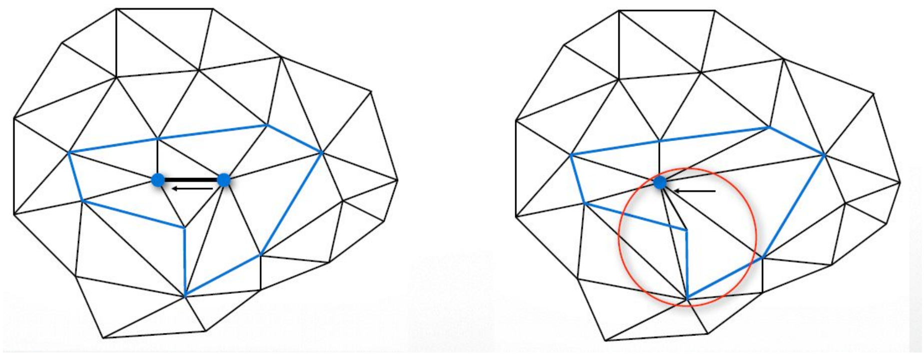
- ✔ Remeshing approaches: parametrization-based vs. surface-based.
- ✔ surface-based: work directly on the mesh embedded in 3D.

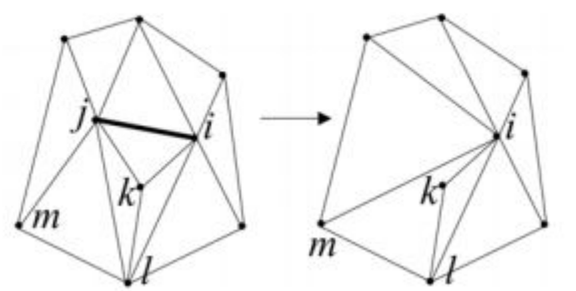


Local remeshing operators.

Remeshing

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

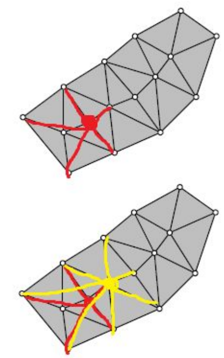




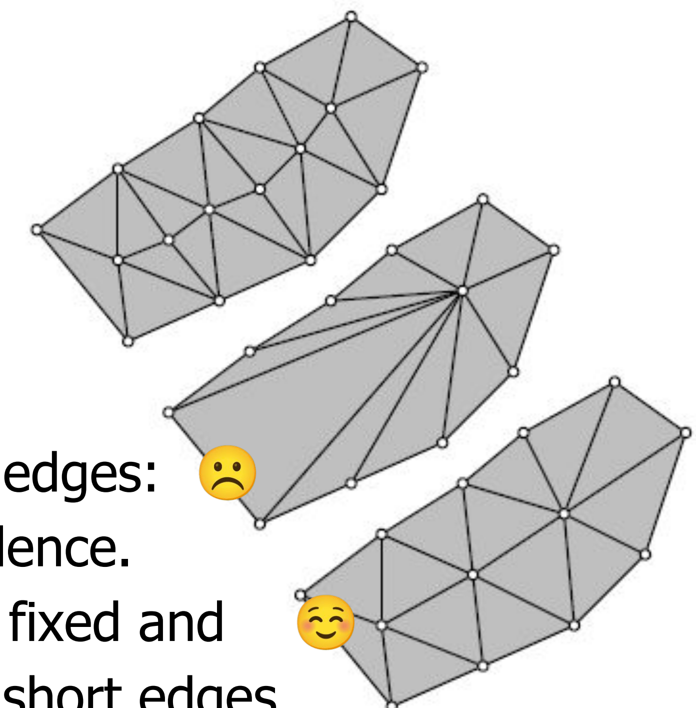
i) Normal flip after collapse! ii) intersection of 1-ring neighborhood of i and j contains 3+ vertices!

Remeshing

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



☹️



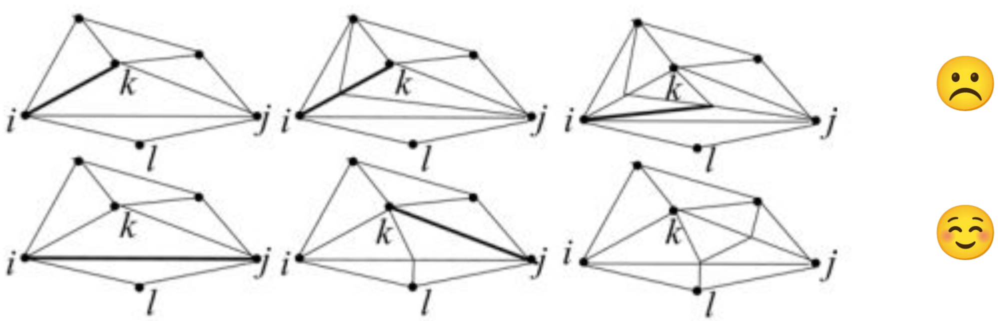
i) A heuristic while removing short edges: ☹️
Collapse into the vert w/ higher valence.
Works ‘cos high-valence verts stay fixed and ☺️
every collapse reduces # adjacent short edges.

Remeshing

✔ Remeshing approaches: parametrization-based vs. surface-based.

✔ surface-based: work directly on the mesh embedded in 3D.

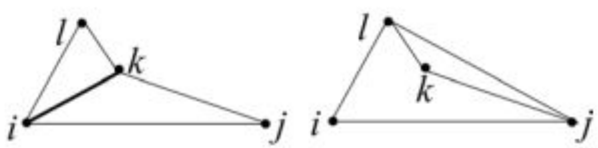
✔ Beware of illegal edge splits:



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

Remeshing

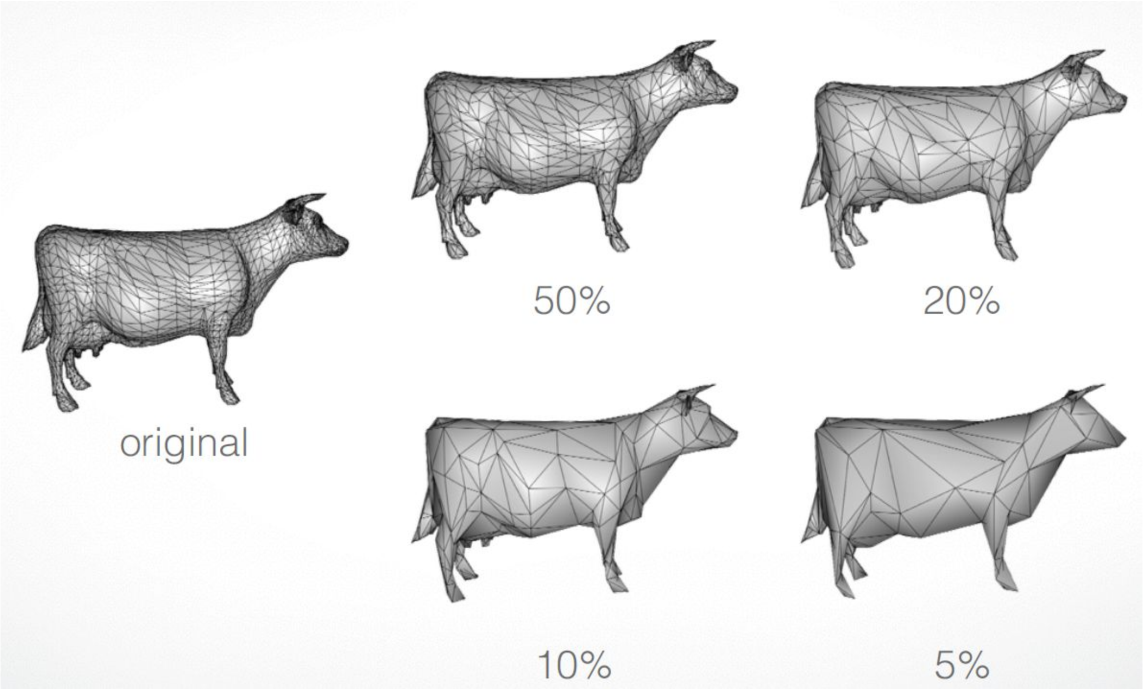
- ✔ Remeshing approaches: parametrization-based vs. surface-based.
- ✔ surface-based: work directly on the mesh embedded in 3D.
- ✔ Beware of illegal edge flips:



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.

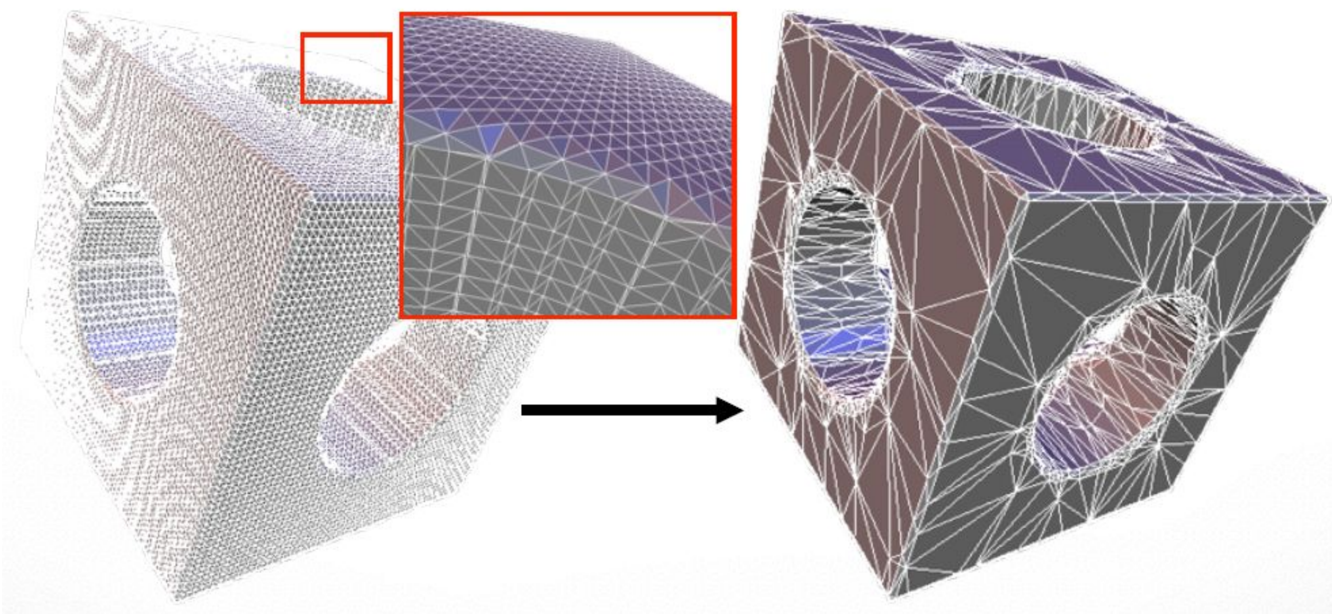
Remeshing

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



Remeshing

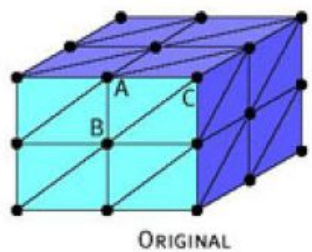
- ✔ Remeshing approaches: parametrization-based vs. surface-based.
- ✔ surface-based: work directly on the mesh embedded in 3D.
- ✔ A sequence of edge collapses, aka mesh decimation:



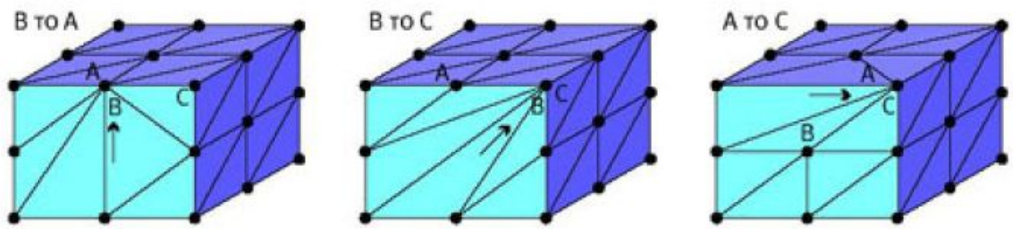
Isosurface extraction by Marching Cubes over-tessellates (left).

Remeshing

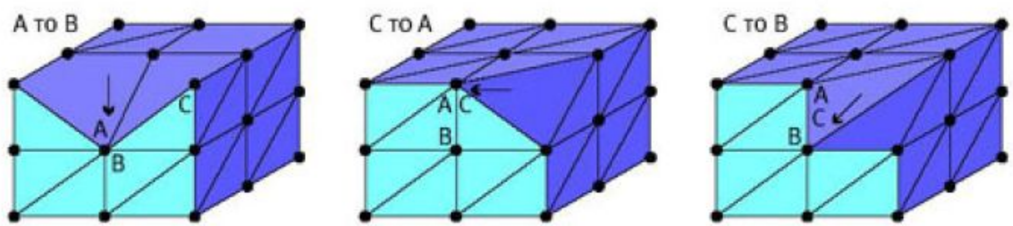
- ✔ Metric for edge collapses (other than edge length metric).
- ✔ Curvature factor is introduced as coplanar surfaces can be represented using fewer polygons than areas w/ a high curvature.



Good collapses:



Bad collapses:



Remeshing

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

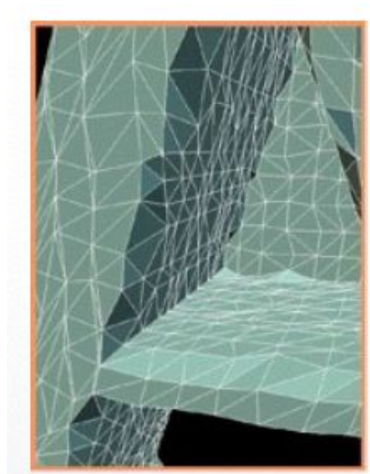
$$\operatorname{cost}(u,v) = \|u - v\| \times \max_{f \in \mathrm{T_u}} \left\{ \min_{g \in \mathrm{T_{uv}}} \left\{ \left(1 - \vec{n}_f \cdot \vec{n}_g \right) \div 2 \right\} \right\}$$

- ✔ Collapse cost of edge (u,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

- ✔ Metric for edge collapses (other than edge length metric).
- ✔ Error quadric: based on the observation that in the original model each vertex is the solution of the intersection of a set of planes.
- ✔ Such a set of planes is associatd w/ each vertex as supporting planes.
- ✔ The error at the vertex w.r.t. this set is the *sum of squared distances to its supporting planes*.
- ✔ Hence this error helps preserving the original details in the decimated model.

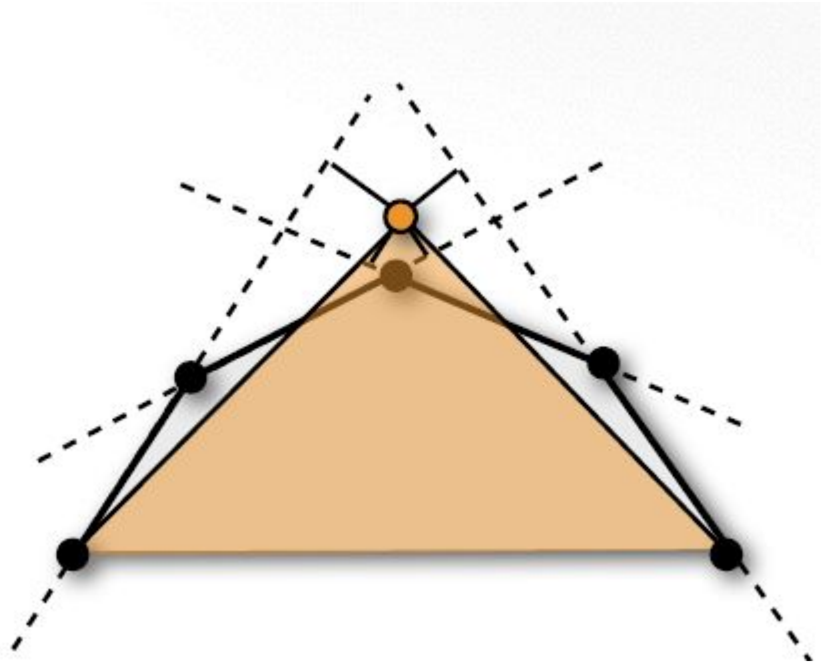
Edge-length metric:



r quadric metric:

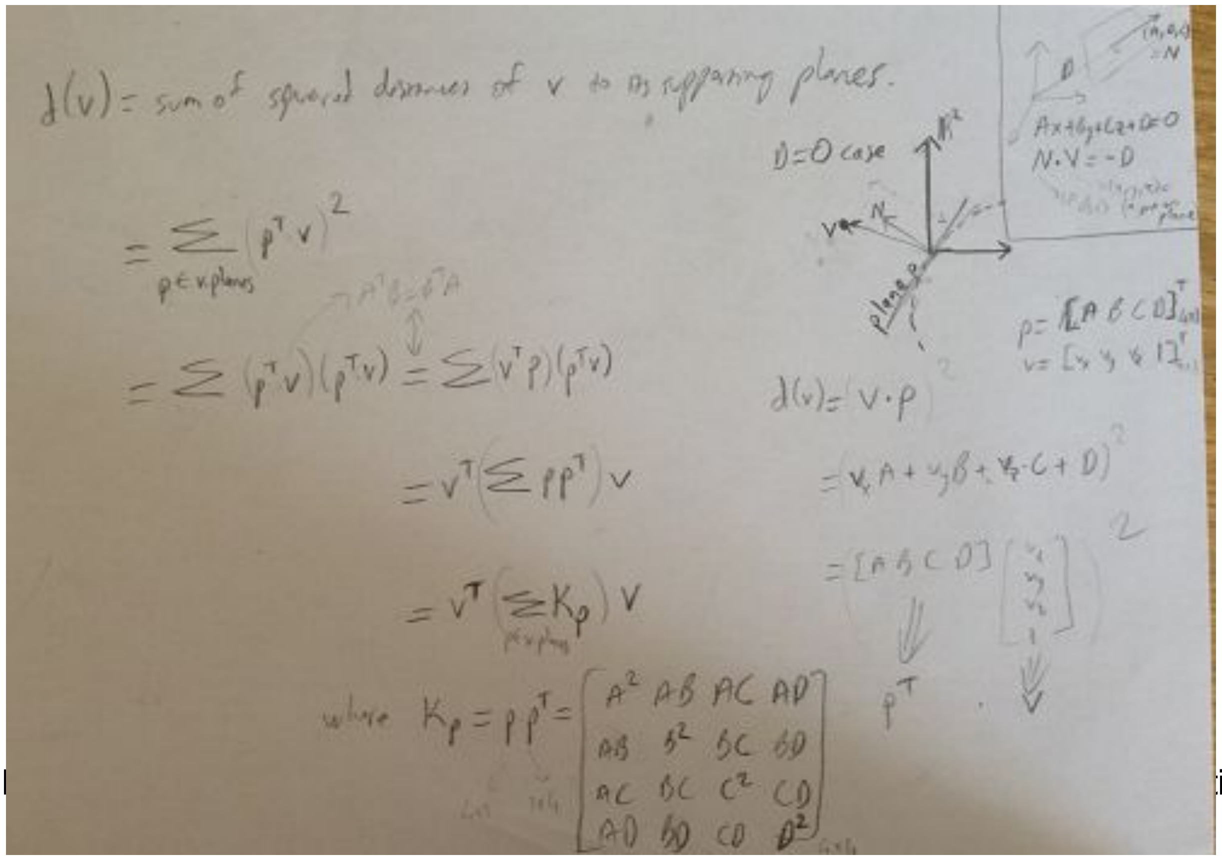
Remeshing

- ✔ Metric for edge collapses (other than edge length metric).
- ✔ Error quadric in 1D (supporting planes □ supporting lines).



Remeshing

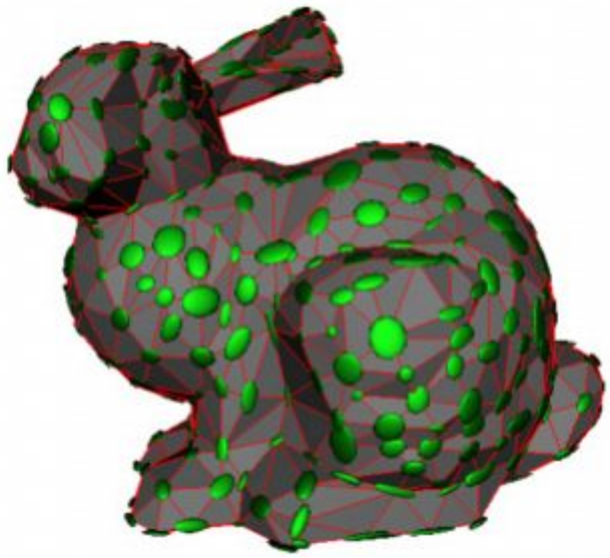
- ✔ Metric for edge collapses (other than edge length metric).
- ✔ Error quadric derivation.



- ✔ Error quadric [...]ing planes.

Remeshing

- ✔ Metric for edge collapses (other than edge length metric).
- ✔ Error quadric visualization.



Remeshing

- ✔ Metric for edge collapses (other than edge length metric).
- ✔ Algorithm:
  - ✔ Collapse cost of edge (u,v): compute the optimal contraction target vertex for this edge (for simplicity u collapses to v). The error at this proposed new vertex (v) becomes the cost of collapsing this edge.
  - ✔ Place all edges in a min-heap keyed on collapse costs.
  - ✔ Iteratively remove the edge w/ min cost, collapse it, update the costs of all involved edges.
  - ✔ Detail: original algorithm uses 'vertex pairs' = edges + 2-close-vertices.
  - ✔ Detail: original algorithm collapses u to a point p* that minimizes error Kp*.

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

  - ✔ Surface Simplification Using Quadric Error Metrics.

Remeshing

- ✔ Term project ideas.
  - ✔ Normal orientation correction: naïve algo (neighbor triangles have similar normals) vs. simple algo (http://www.seas.upenn.edu/~ladislav/takayama14simple/takayama14simple.html).
  - ✔ Linear mesh interpolation vs. nonlinear mesh interpolation.
  - ✔ Mesh decimation with curvature metric (slide 31) vs. error quadric metric (32)
  - ✔ Mesh segmentation using paper: Consistent mesh partitioning and skeletonisation using the shape diameter function.
  - ✔ Mesh skeleton extraction using the same paper.
  - ✔ Hole filling using http://www.cgal.org/gsoc/2012.html#holefill
  - ✔ Deformation model in Microsoft KinEtre.