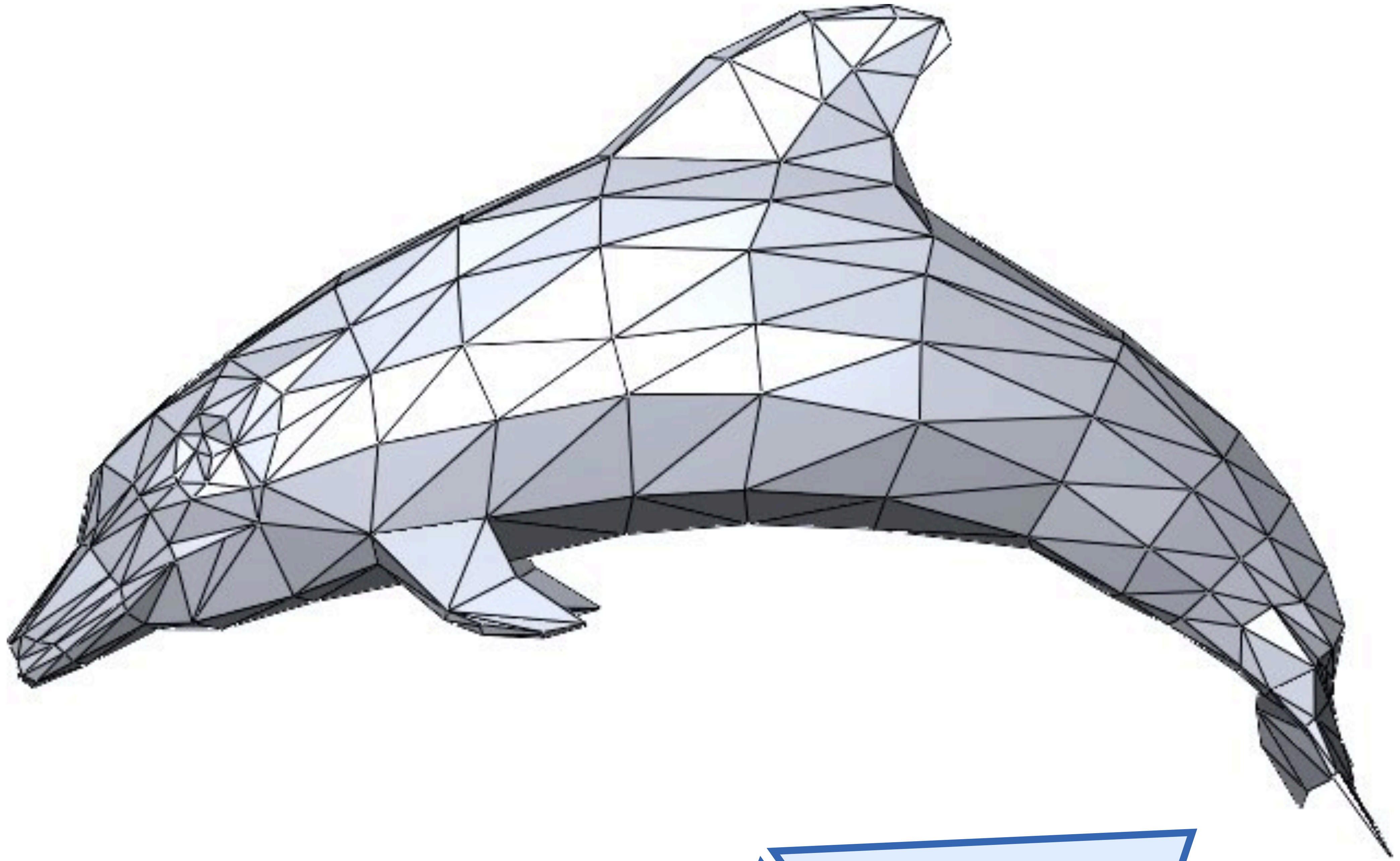
CSE 167 (FA22)
Computer Graphics: Digital Geometry Processing

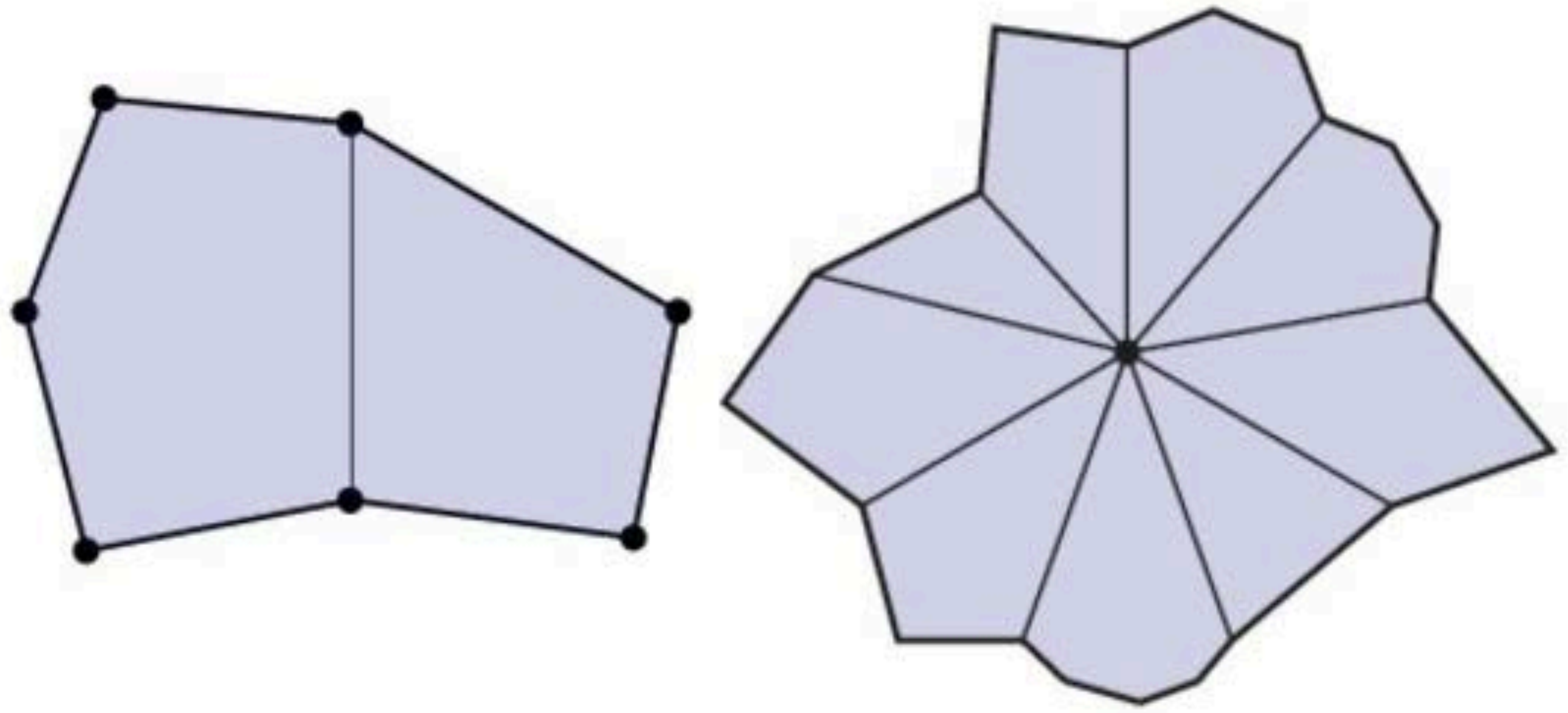
Albert Chern

UC San Diego
JACOBS SCHOOL OF ENGINEERING

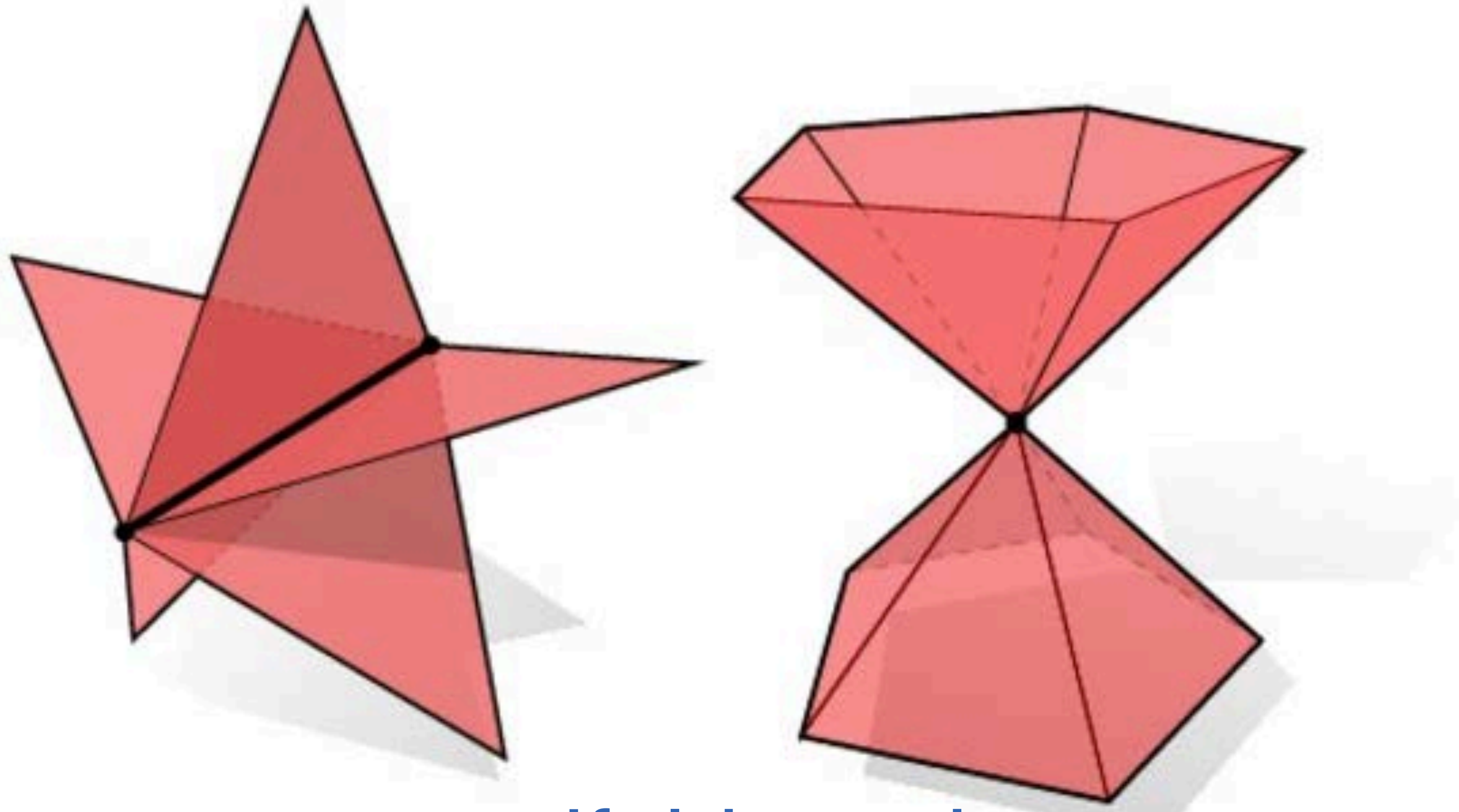
Surfaces

- A few weeks ago, we had a brief overview of surfaces
  - Triangle meshes
  - Modeling: generate surfaces by spline or subdivision

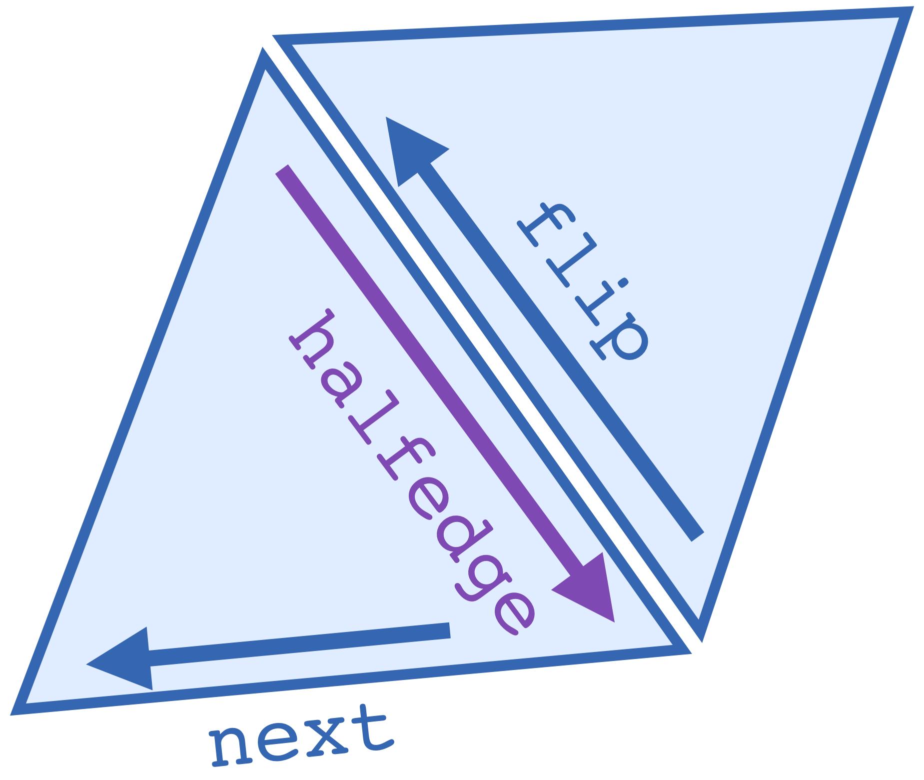




manifold (valid) mesh



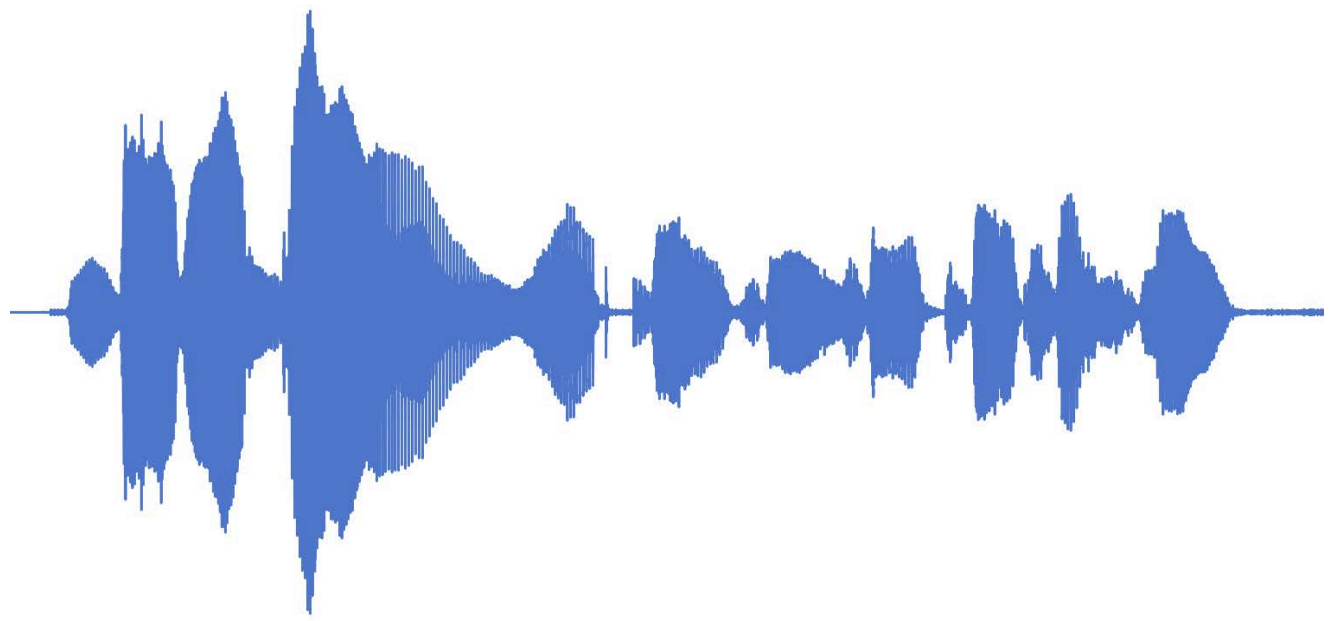
non-manifold mesh



half-edge data structure

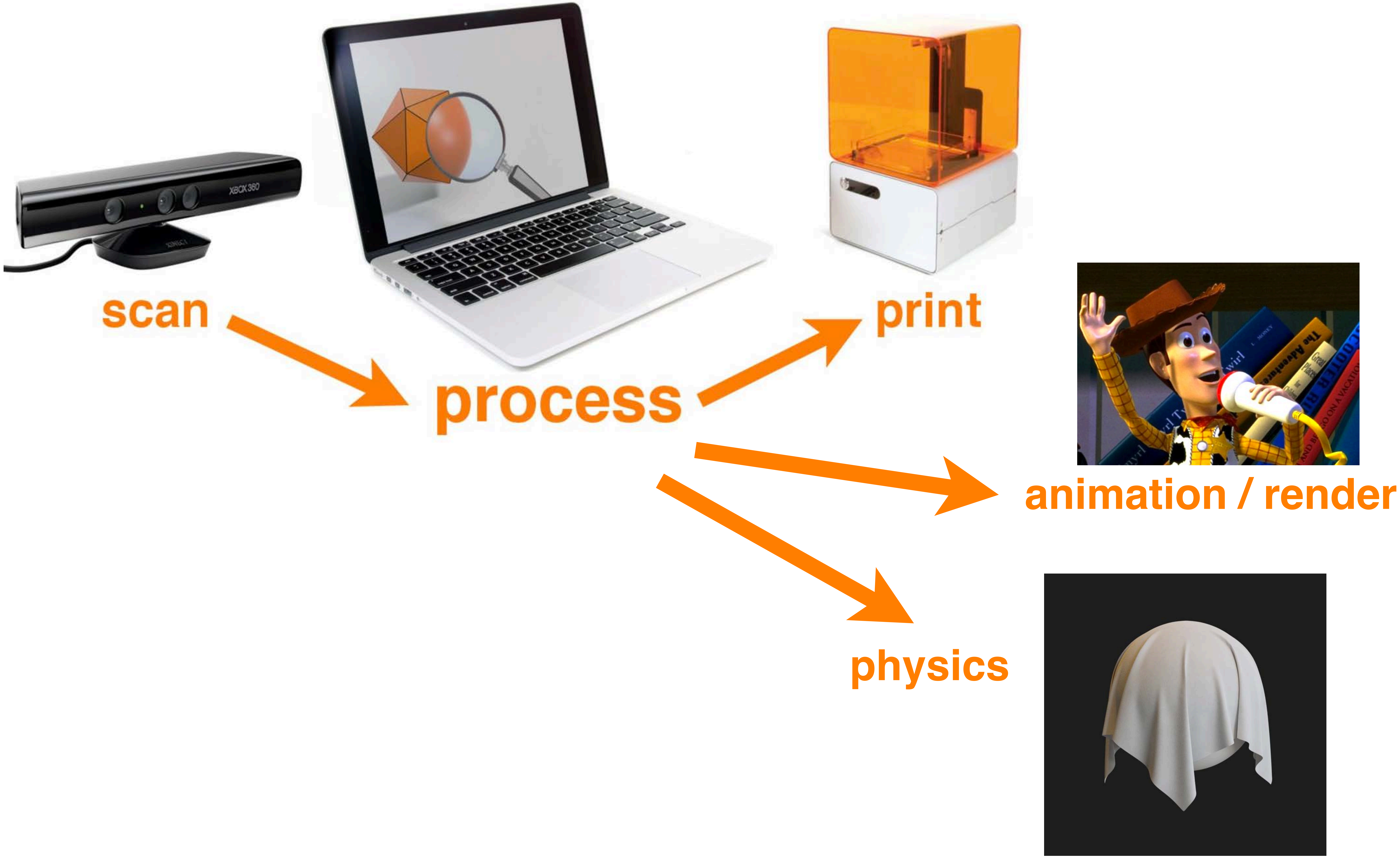
Geometry processing

- View discrete surface as a form of "signal data"
  - Traditional signal data: audio & images

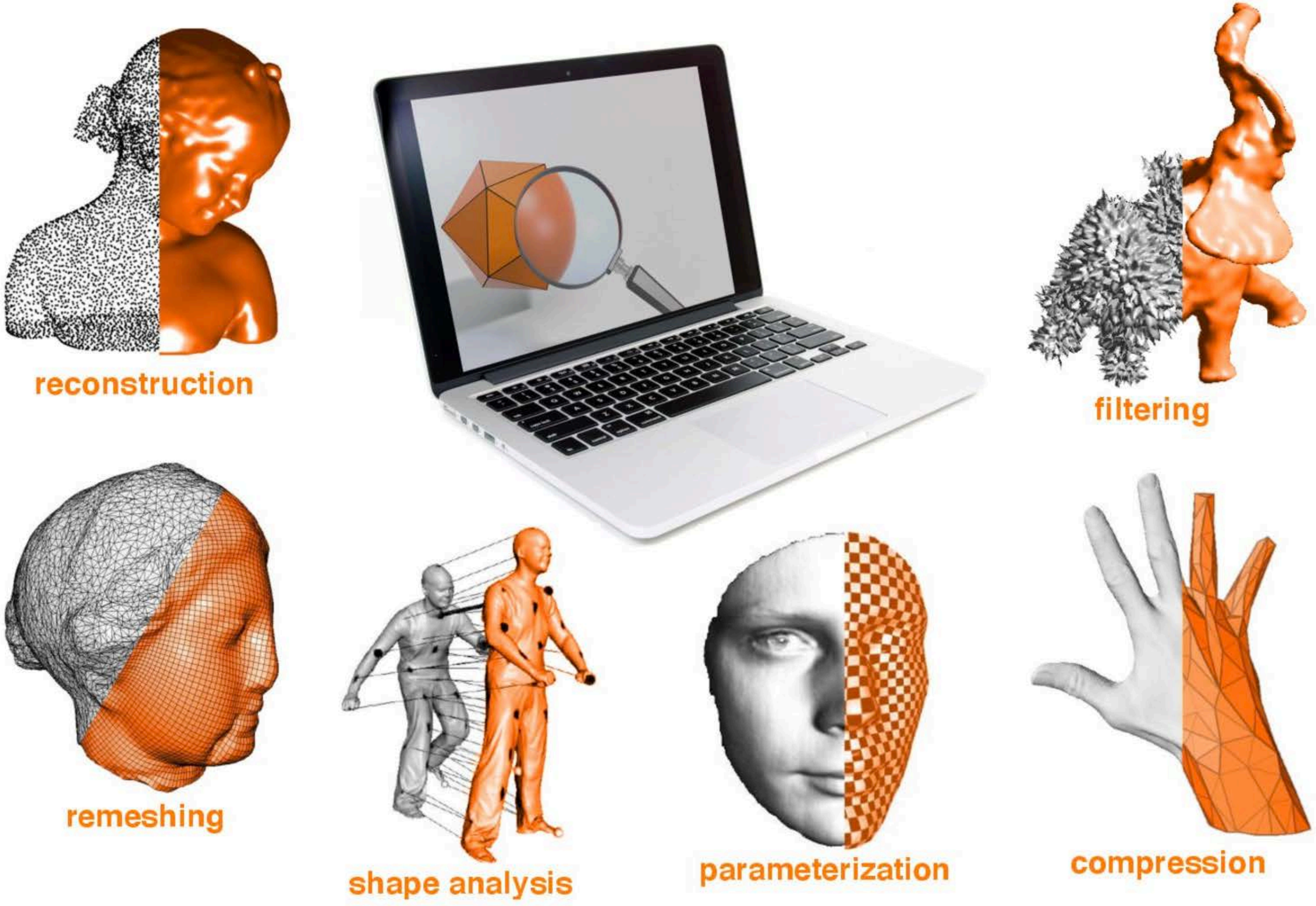


  - Geometric signal
  - Upsampling / downsampling / filtering / aliasing

Geometry processing

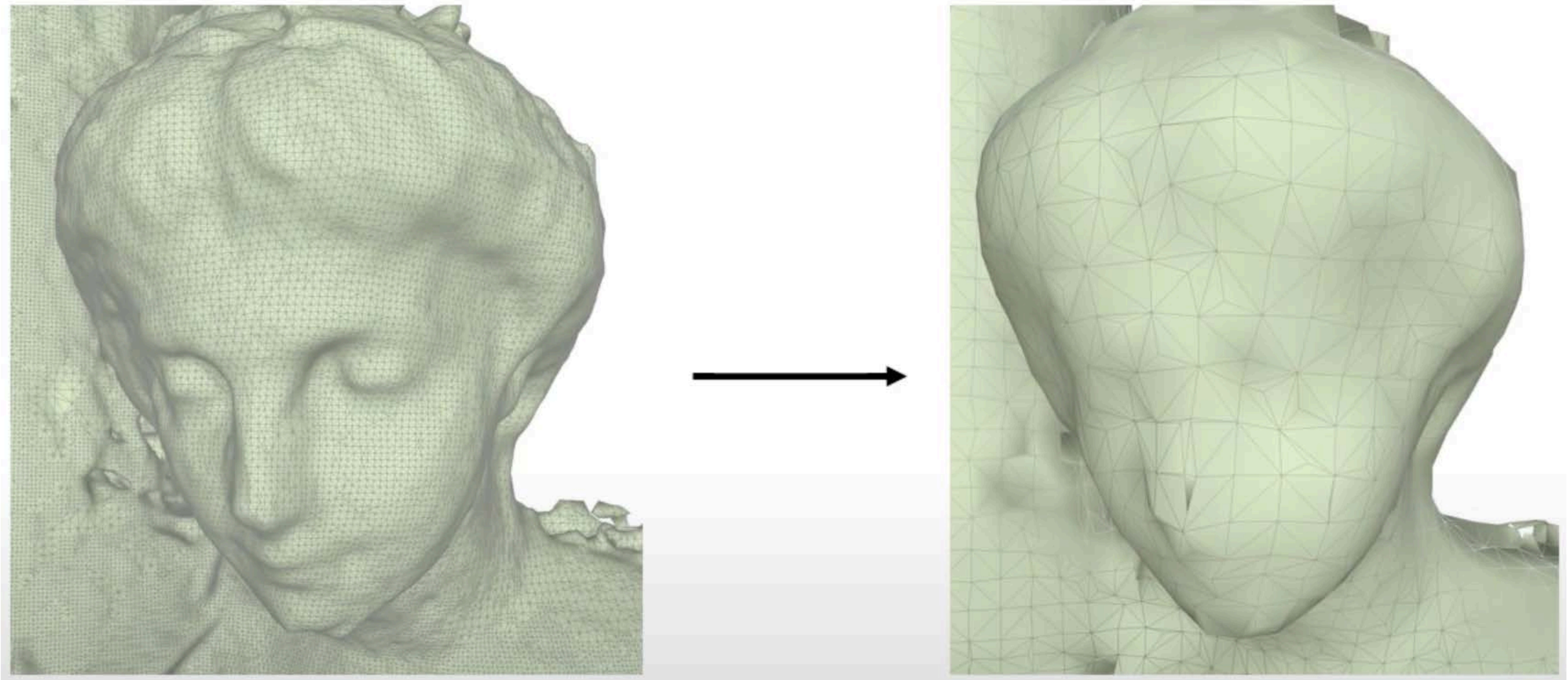


Tasks of geometry processing

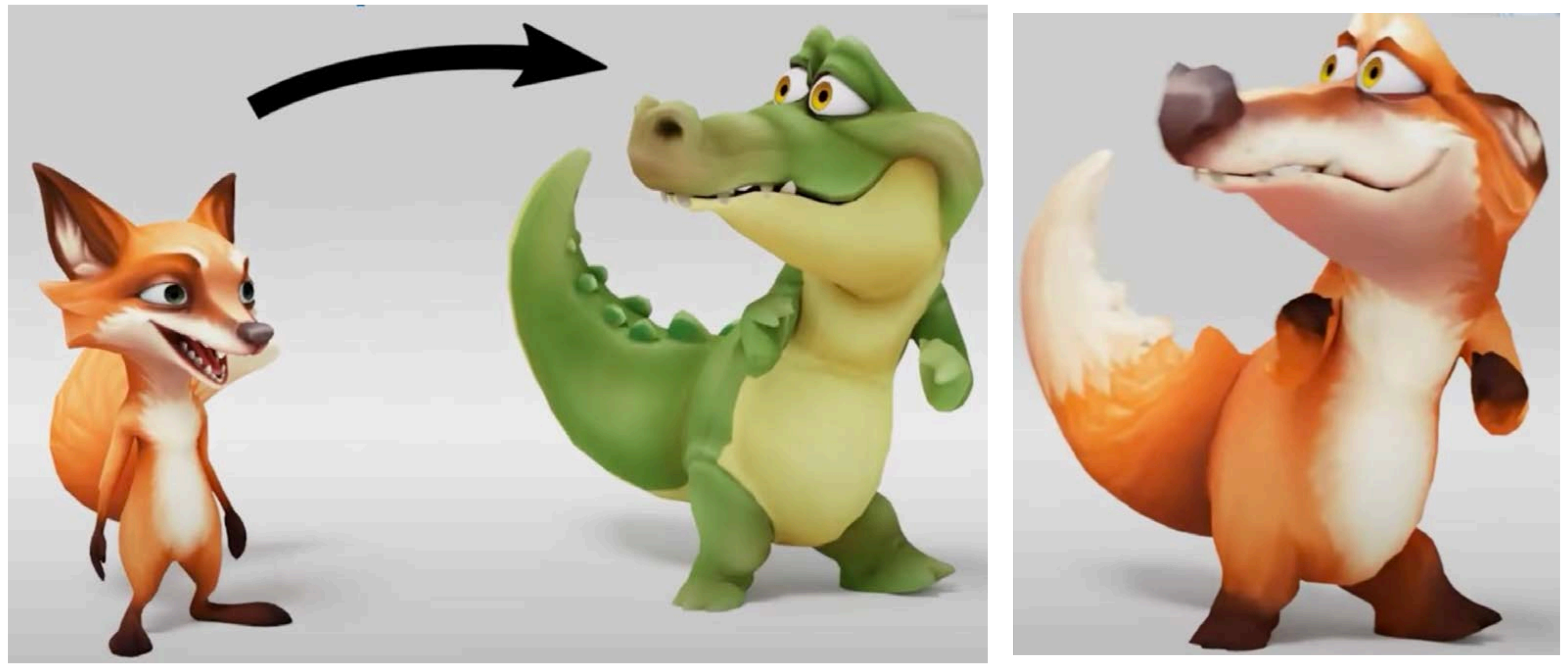


Symposium on Geometry Processing

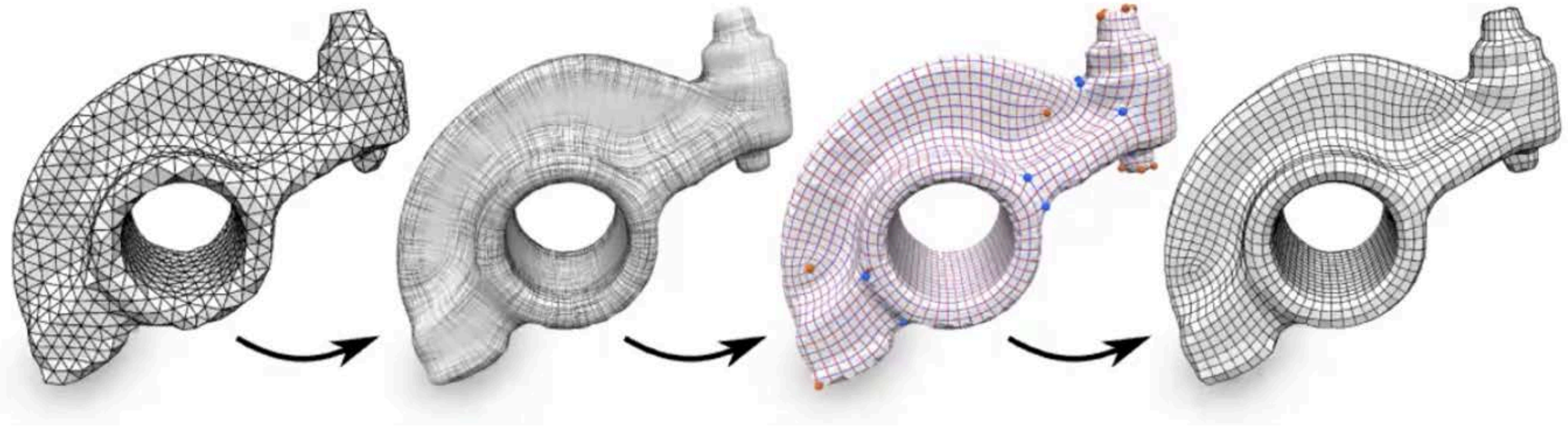
- SGP summer school
  http://school.geometryprocessing.org/



Shape approximation

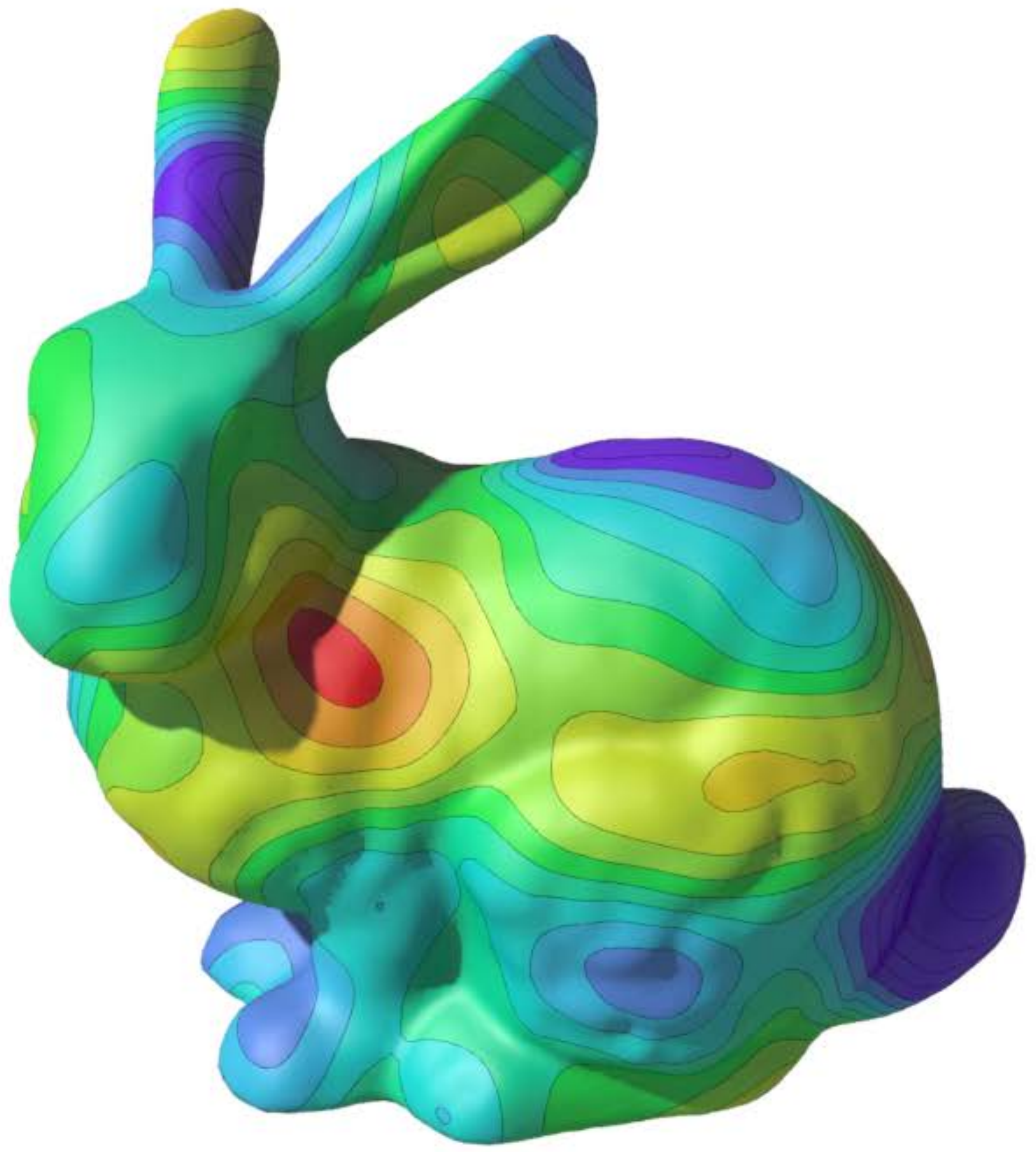


Maps between surfaces

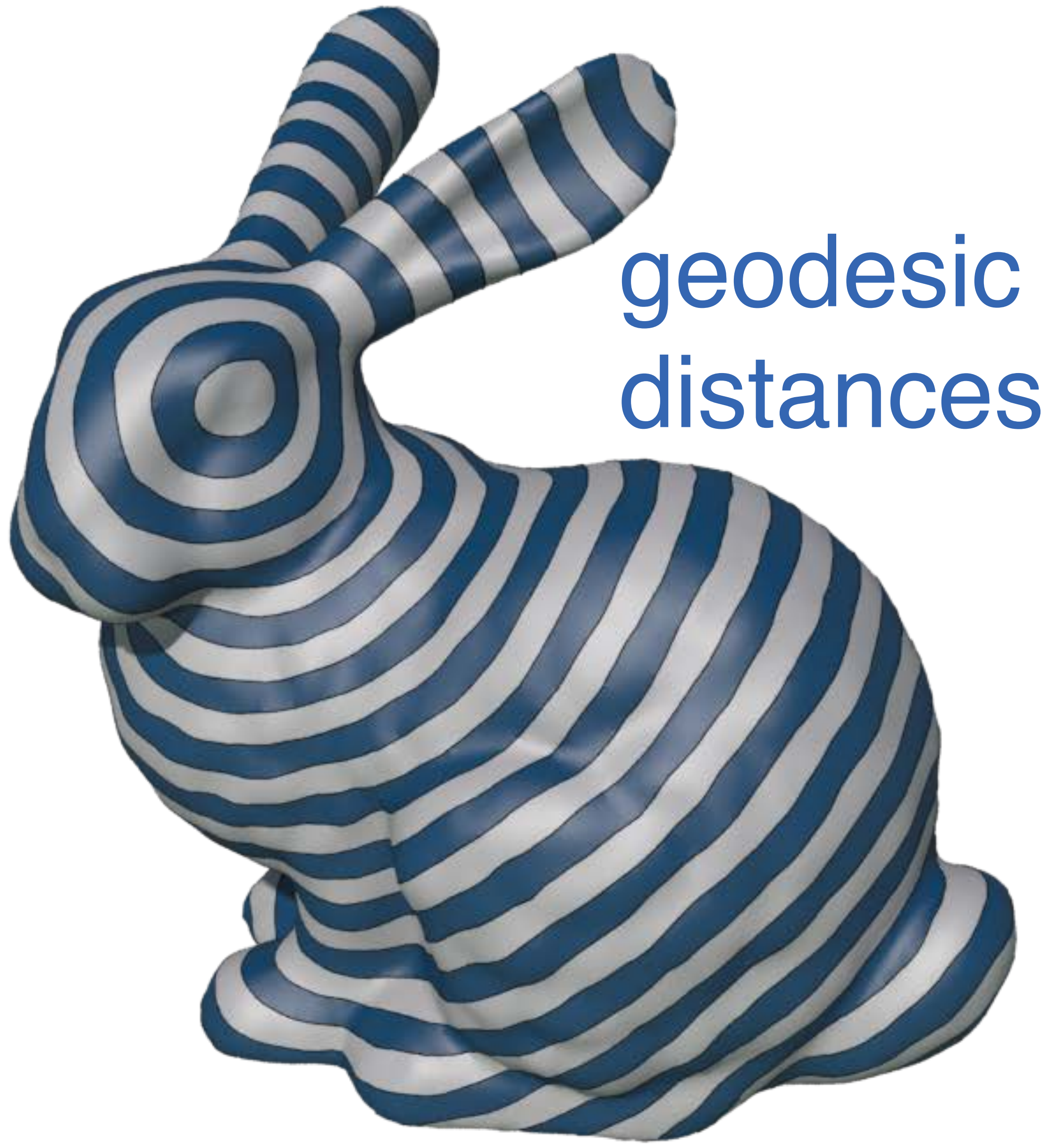


Directional field

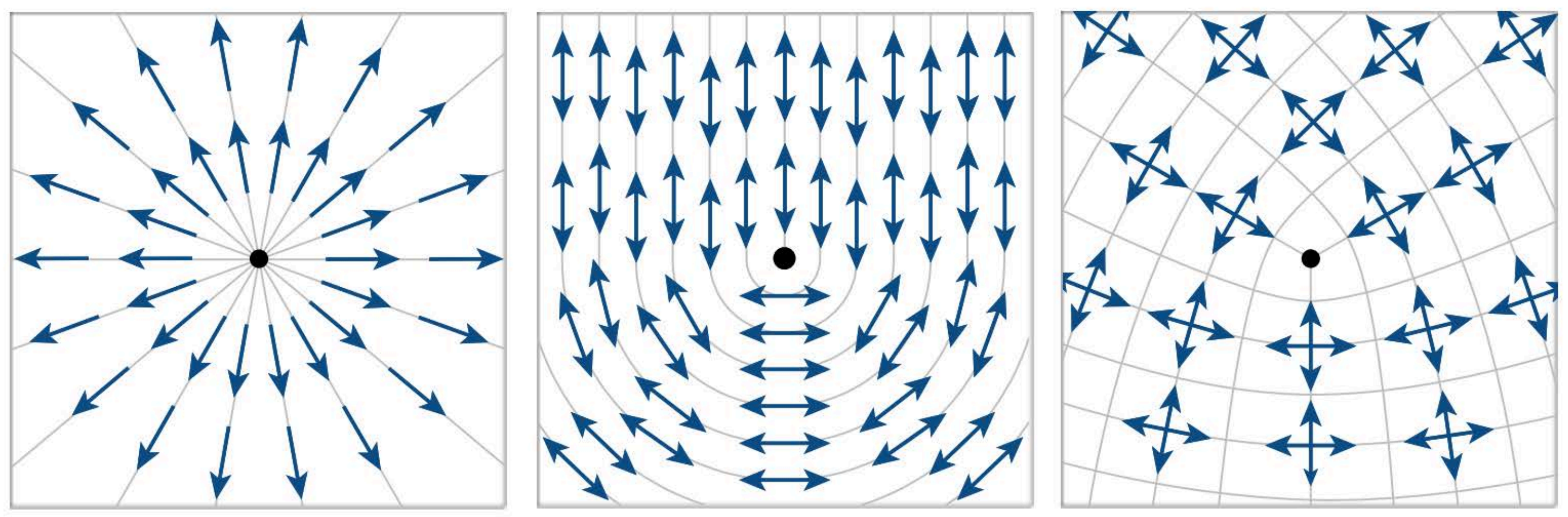
CSE274 (discrete differential geometry)

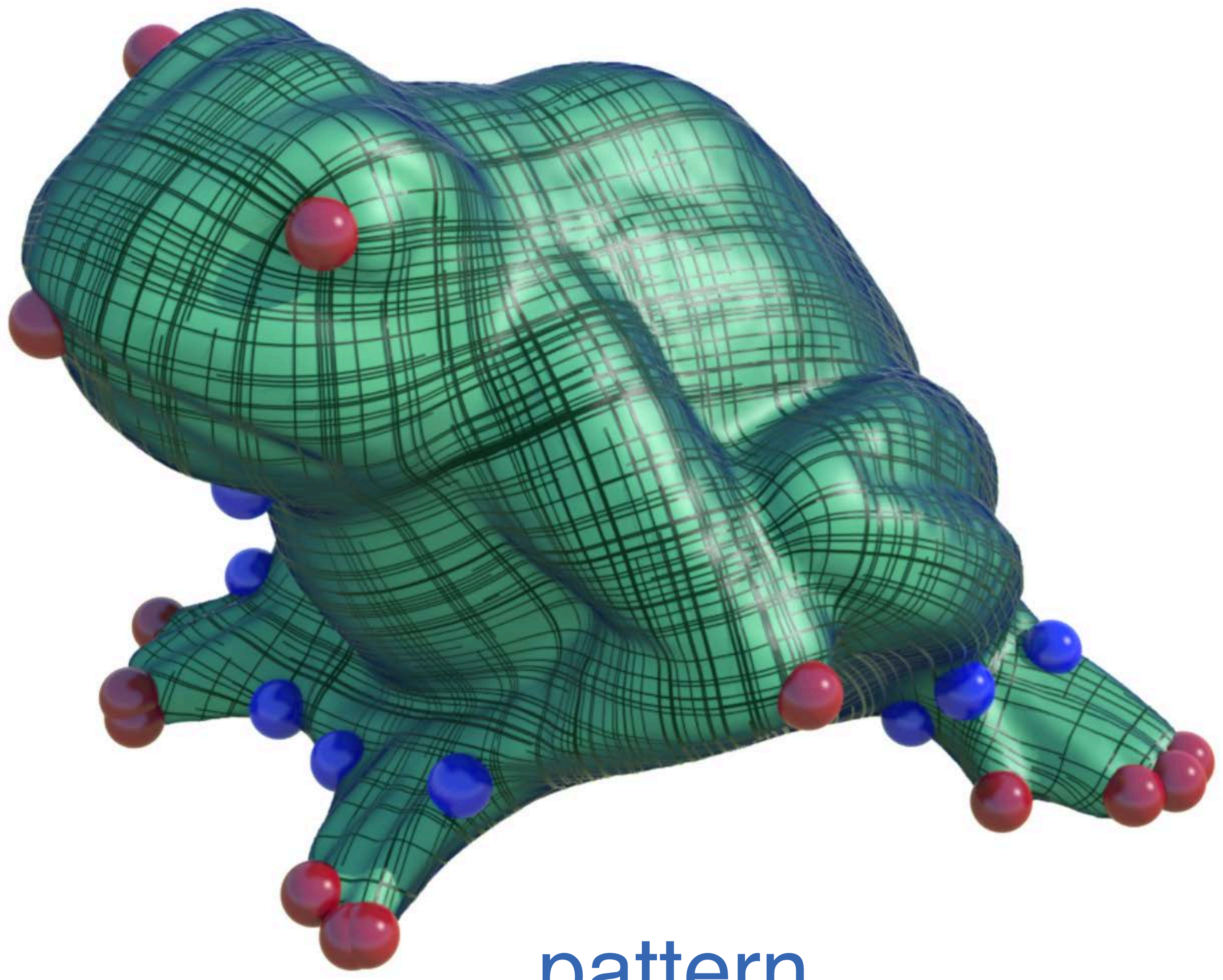


numerical partial
differential equations

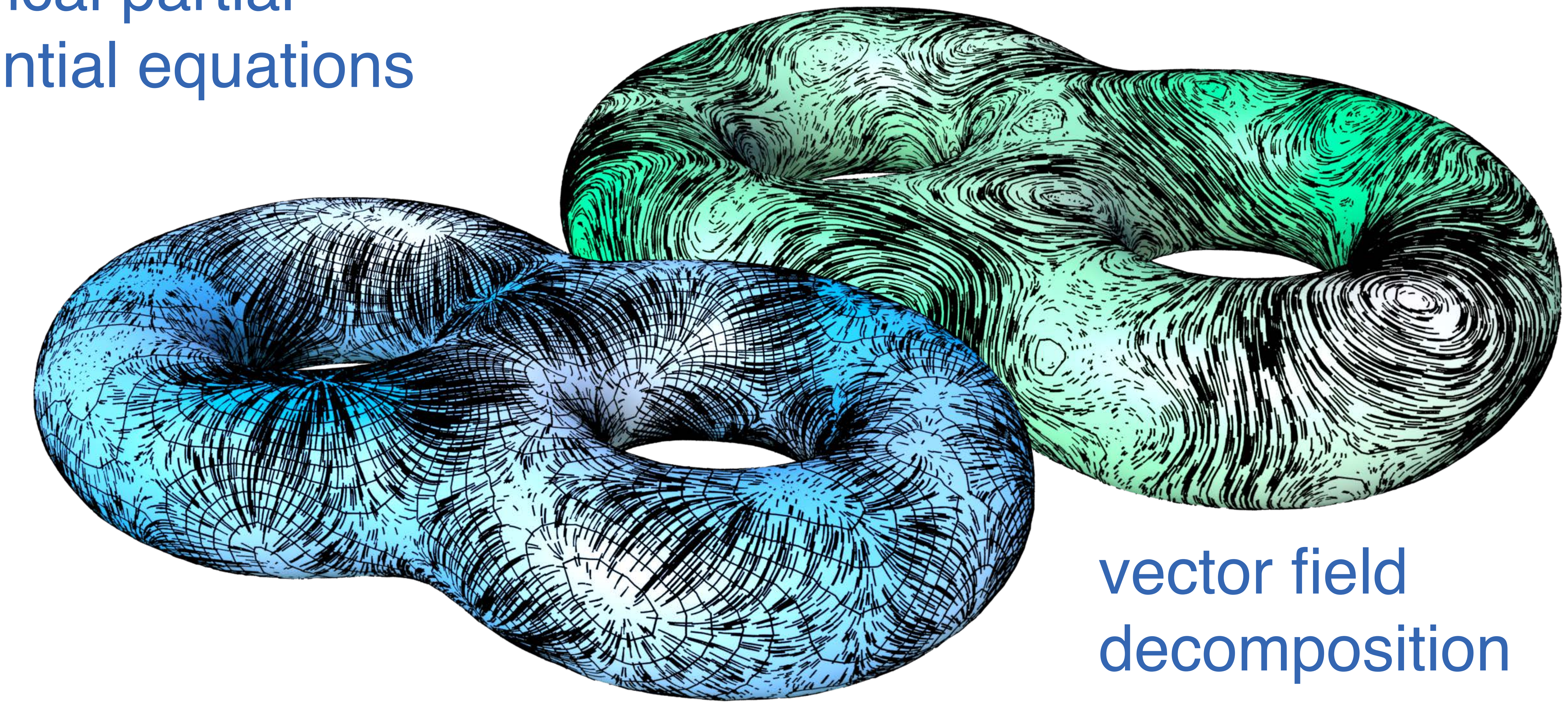


geodesic
distances





pattern
design

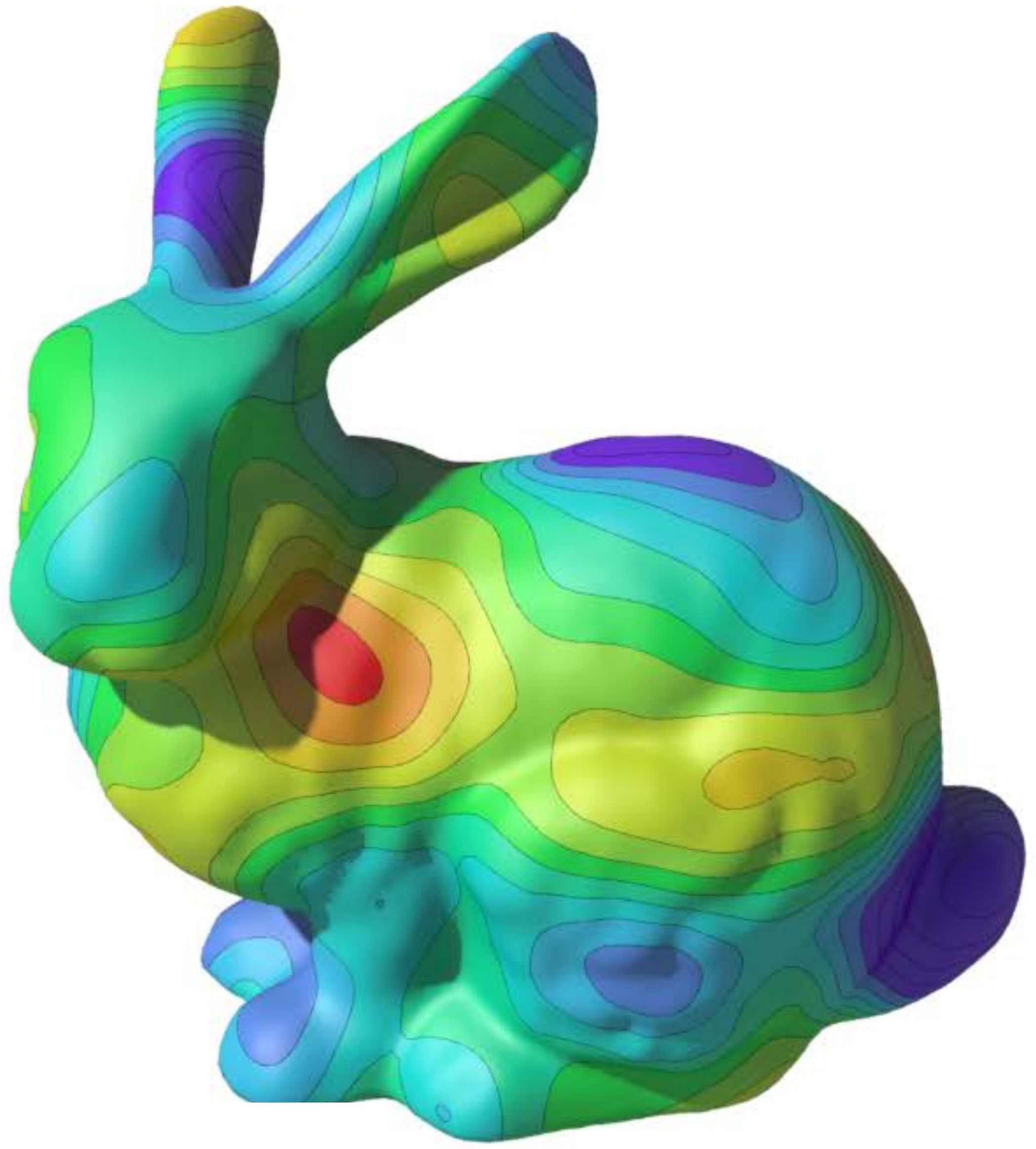


vector field
decomposition

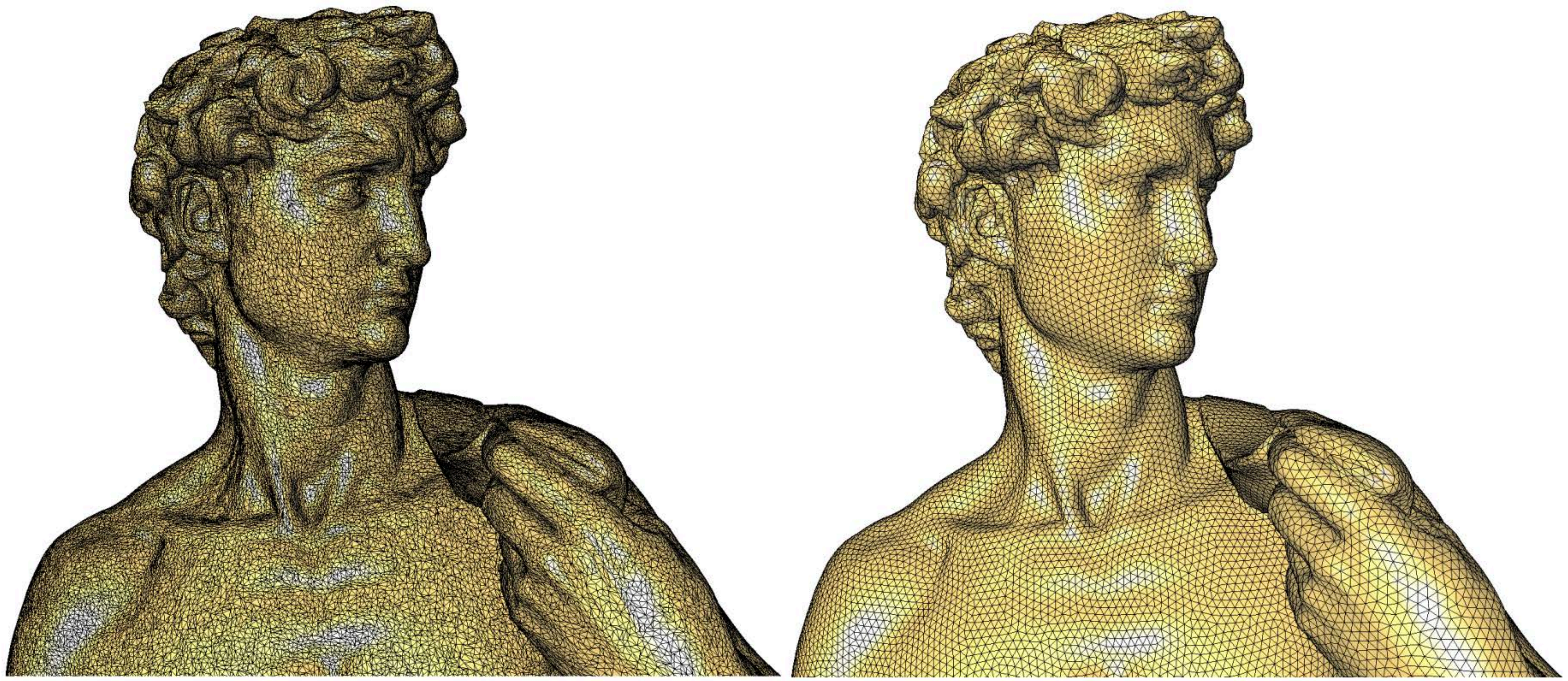
Today

- Surface processing using **Laplacian**

	1	
1	– 4	1
	1	



- Another topic: **Remeshing**



Laplacian

- Laplacian
- Remeshing

2nd derivative

- For a function of 1 variable $f(t)$

  the ***Laplacian*** of the function is its 2nd derivative

$$(\Delta f)(t) := \frac{d^2 f}{dt^2}(t)$$

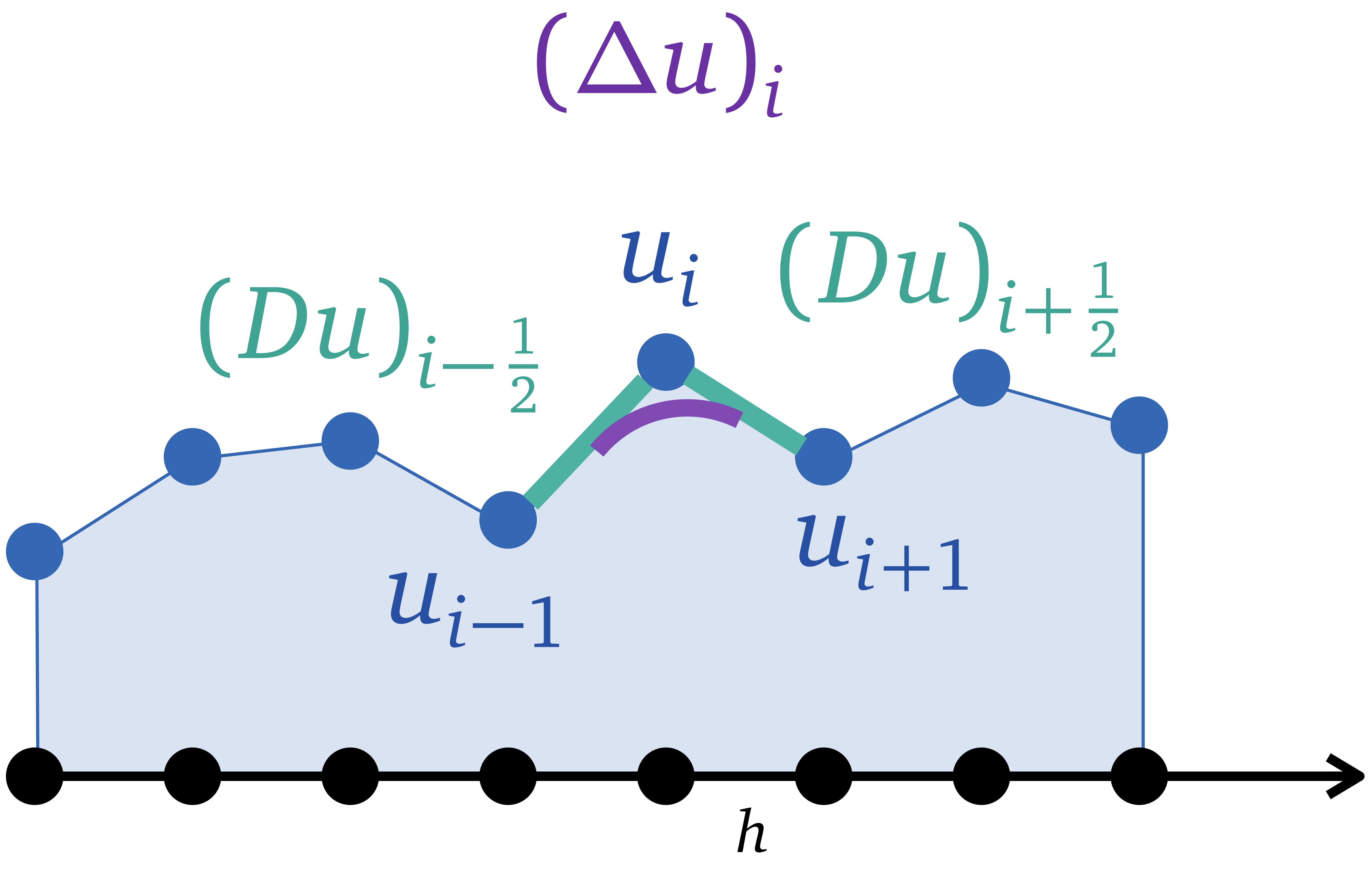
- Laplacian is usually denoted by Δ or ∇^2 or L
- 2nd derivative on 1D measures the difference between the function value at a point and the averaged function value around that point.
- Laplacian is a 2nd derivative on a 2D or 3D or surface domain measuring the deviation of value from the neighbor average

A simple discrete Laplacian

- In 1D $\Delta u(x) = u''(x)$
- Discretize 1D space into uniform grid
- grid size h
- Discrete 1st derivative

$$(Du)_{i-\frac{1}{2}} = \frac{u_i - u_{i-1}}{h}$$

$$(Du)_{i+\frac{1}{2}} = \frac{u_{i+1} - u_i}{h}$$



- Discrete 2nd derivative

average of neighbor

function value

$$(\Delta u)_i = \frac{(Du)_{i+\frac{1}{2}} - (Du)_{i-\frac{1}{2}}}{h} = \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = \frac{2}{h^2}\left(\frac{u_{i-1} + u_{i+1}}{2} - u_i\right)$$

In image processing (2D domain)

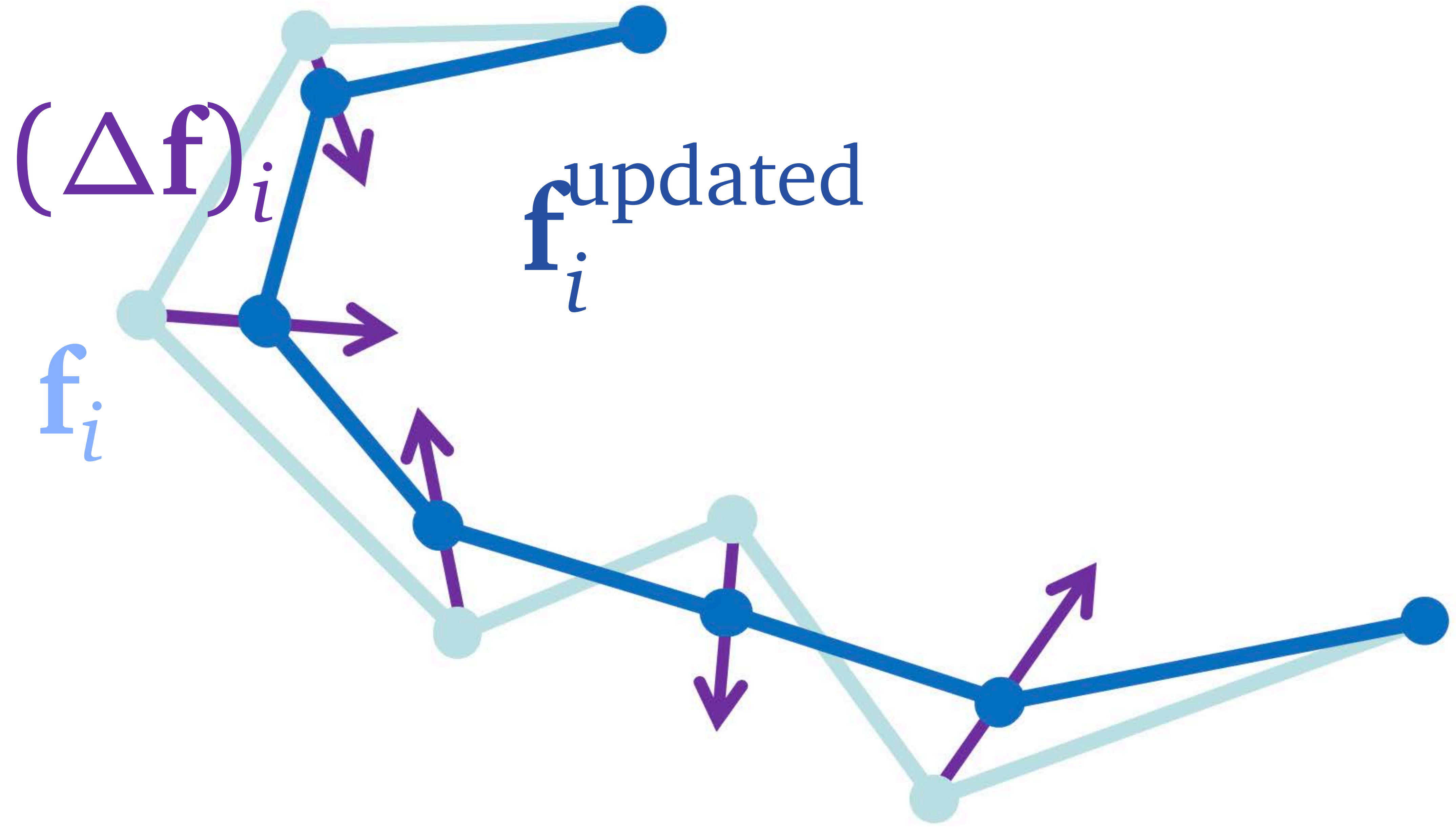
- Laplace filter

	1	
1	– 4	1
	1	

- In multivariable calculus $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$
- Useful for edge detection

For smoothing

- Laplacian measures how much the neighbor deviates from center

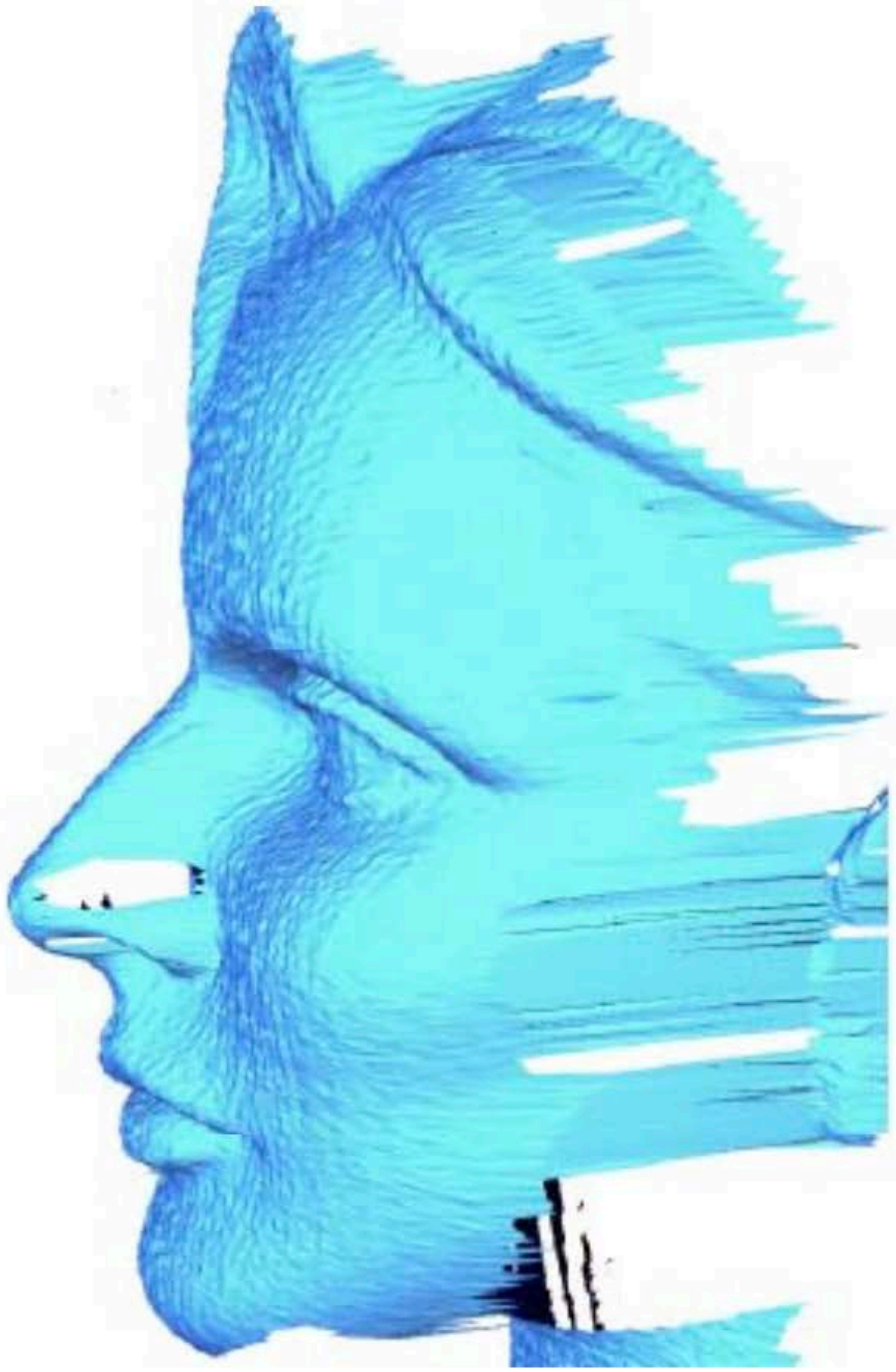


$$\frac{\partial}{\partial t}\mathbf{f}_i = (\Delta \mathbf{f})_i$$

This is also the heat diffusion equation

- Laplacian of the vertex position is proportional to the (mean) curvature of the curve/surface

For smoothing



raw 3D scanning data

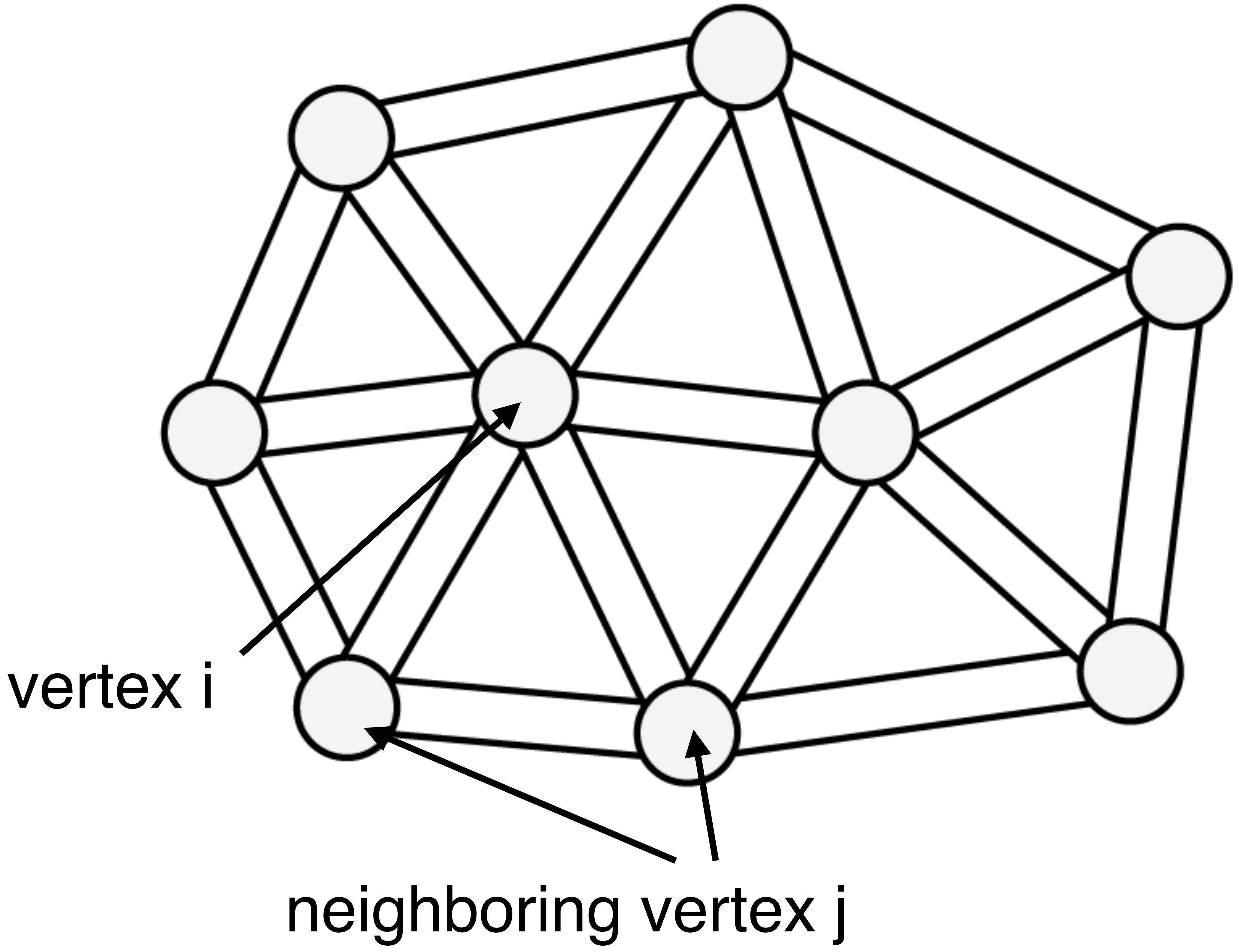
after a few steps of Laplacian smoothing
(mean curvature flow)

Laplacian on mesh or graph

edge weights

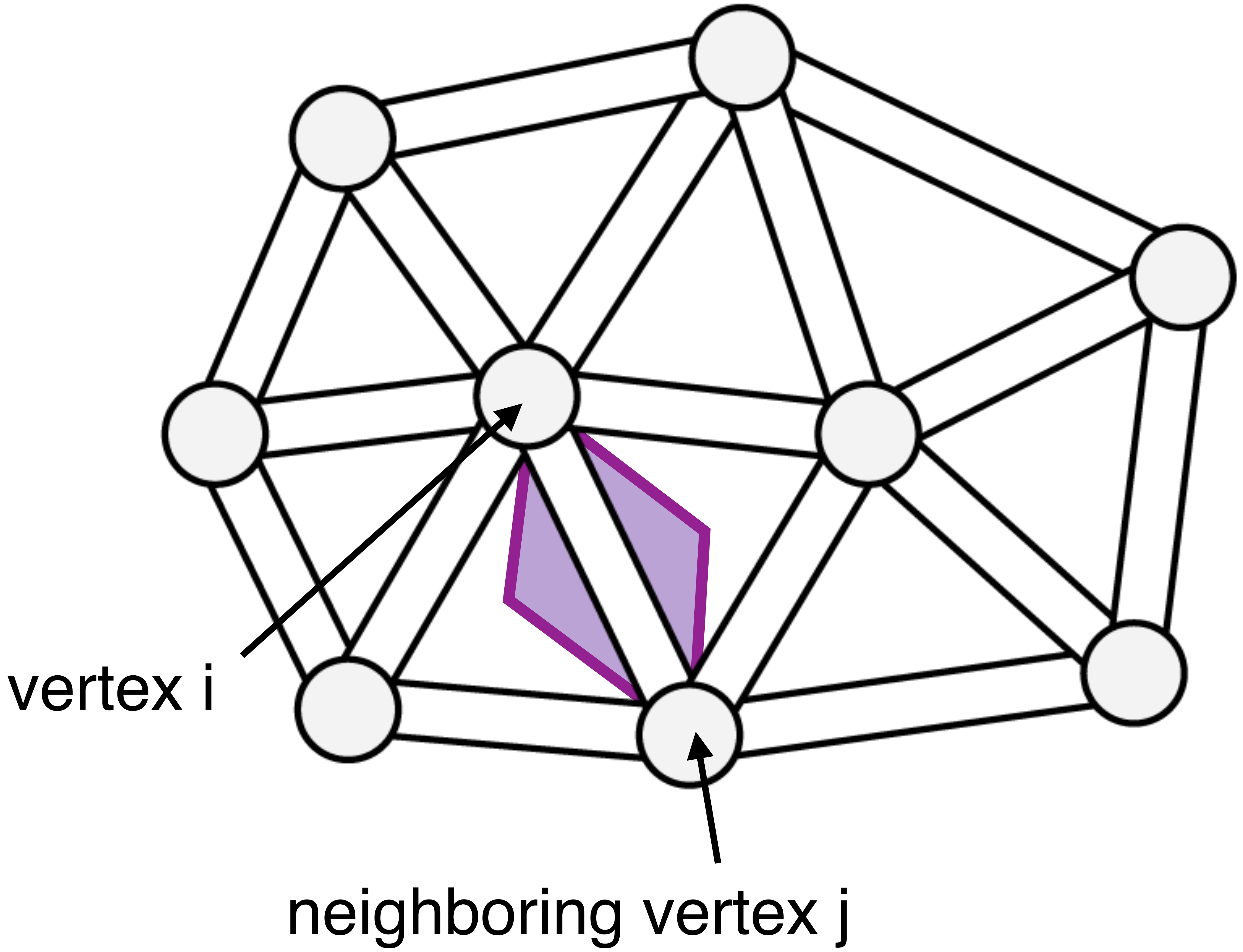
$$(\Delta u)_i = \sum_{j \in \mathrm{Neighbor}(i)} w_{ij}(u_j - u_i)$$

- In graph theory, people usually take edge weights to be all 1 (graph laplacian)
- In geometry processing, edge weights are chosen to mimic the effective *conductivity* on the edge



Laplacian on mesh or graph

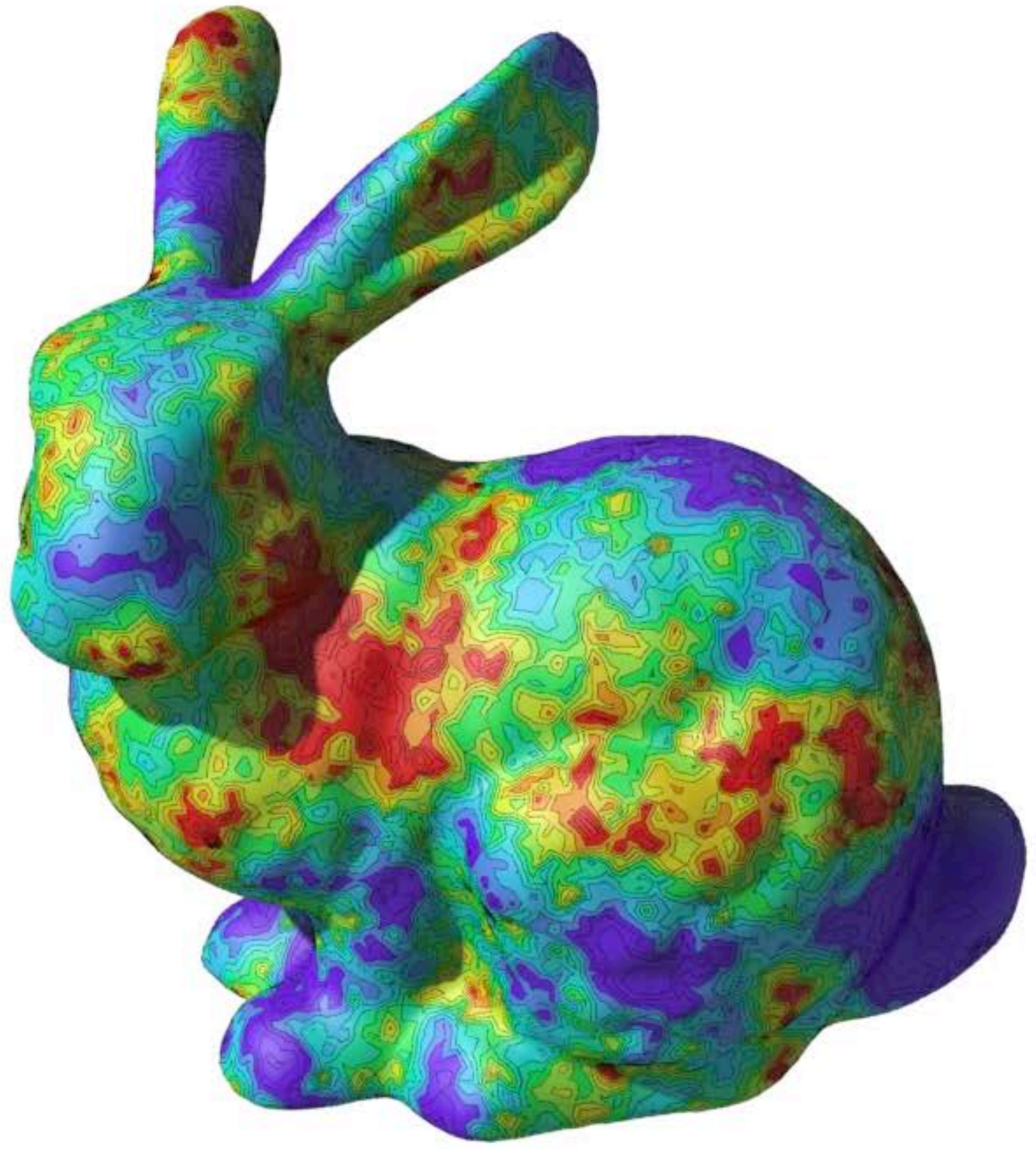
$$w_{ij} = \frac{\text{width of the kite}}{\text{edge length}}$$



- If the mesh quality is "good" then the discrete Laplacian approximates the continuous Laplacian well

Laplacian on mesh or graph

- With Laplacian, many image processing techniques (smoothing, curvature detection, reconstructions) can be done on surfaces.
- Many physical equation (usually only written in Cartesian coordinates) can be simulated on surfaces.



Remeshing

- Laplacian
- Remeshing

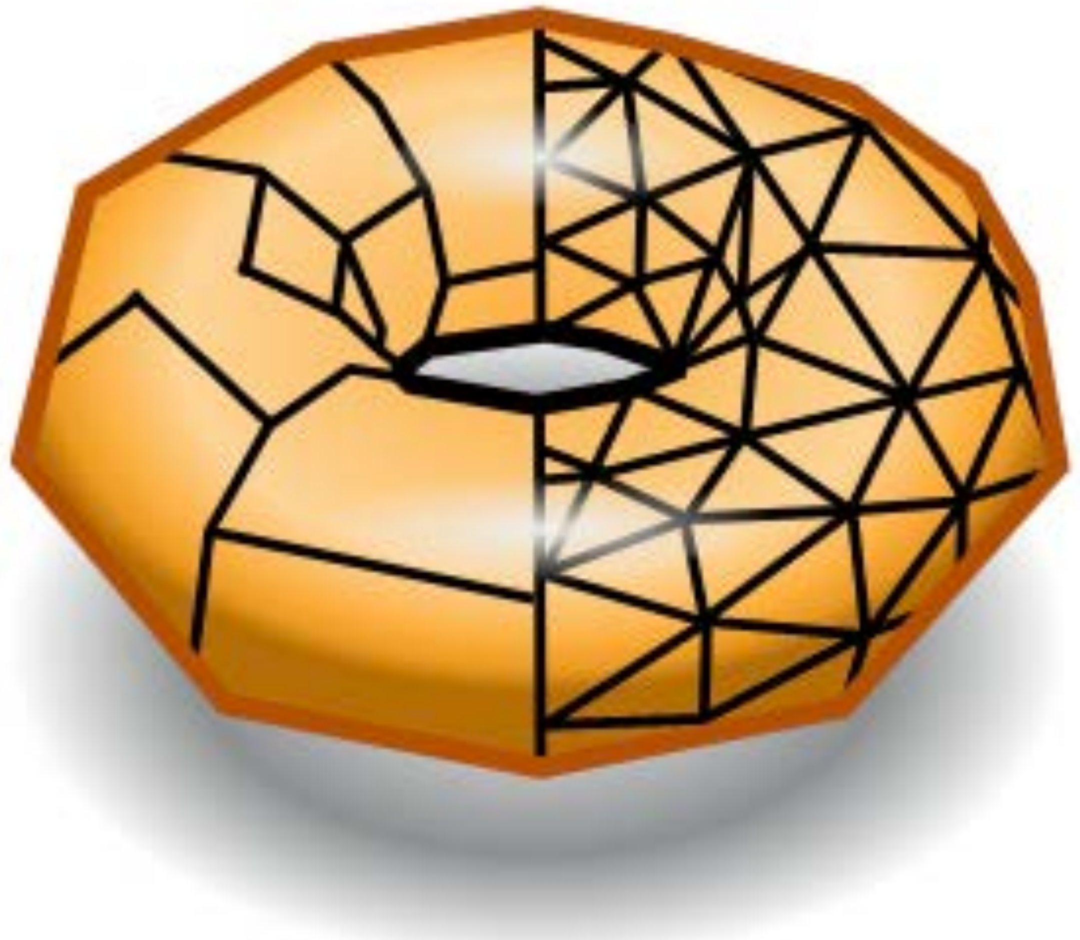
Today

- Let's talk about: **Remeshing**

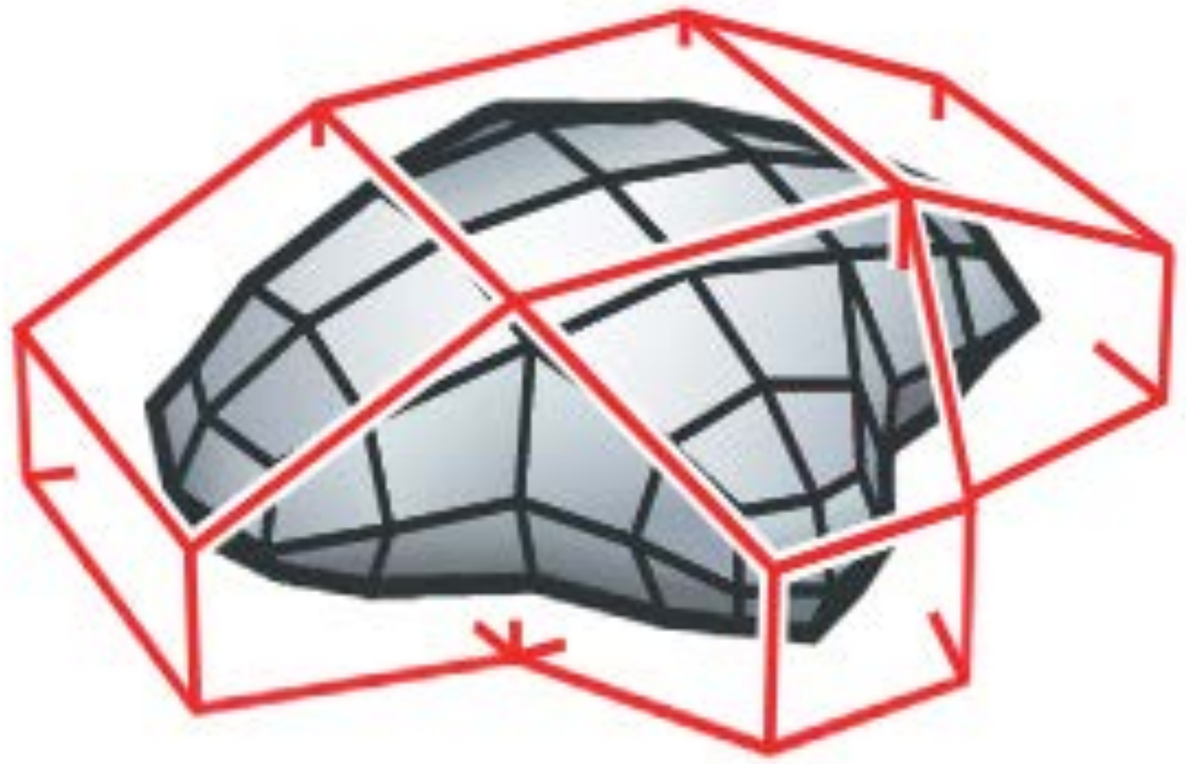
Today

- Let's talk about: **Remeshing**

Reduce polygon



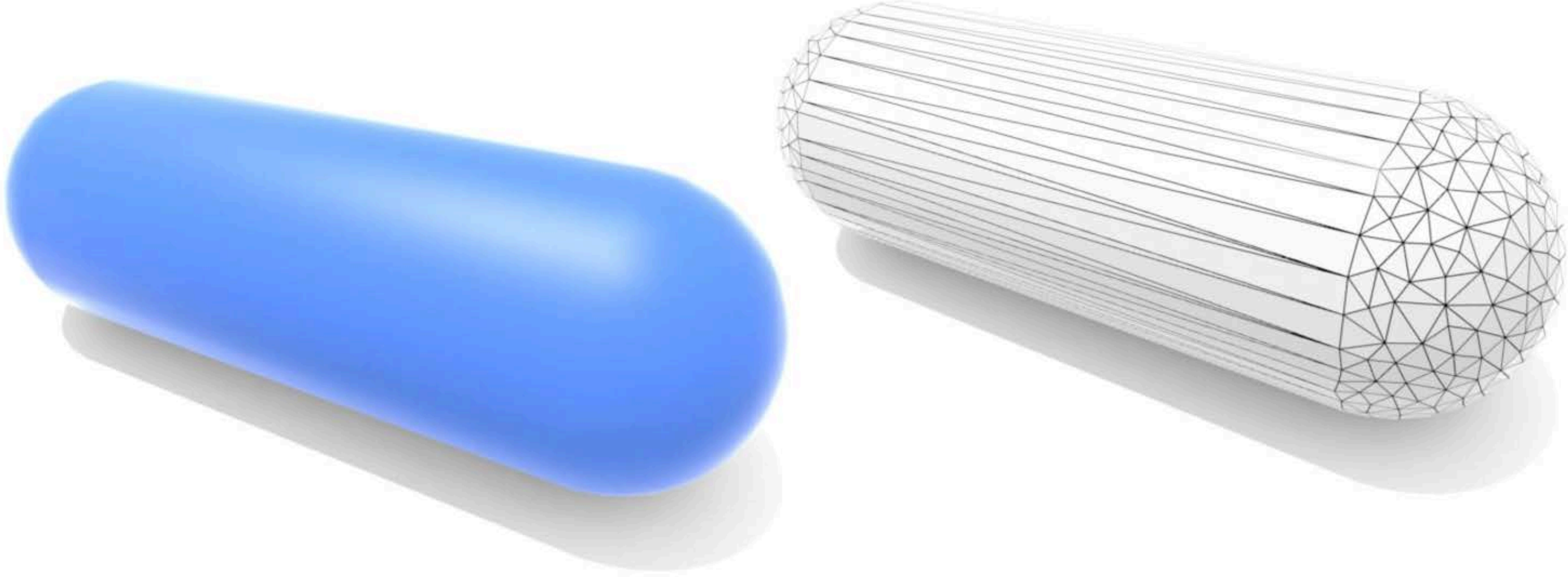
Obtain better triangle mesh



Subdivision

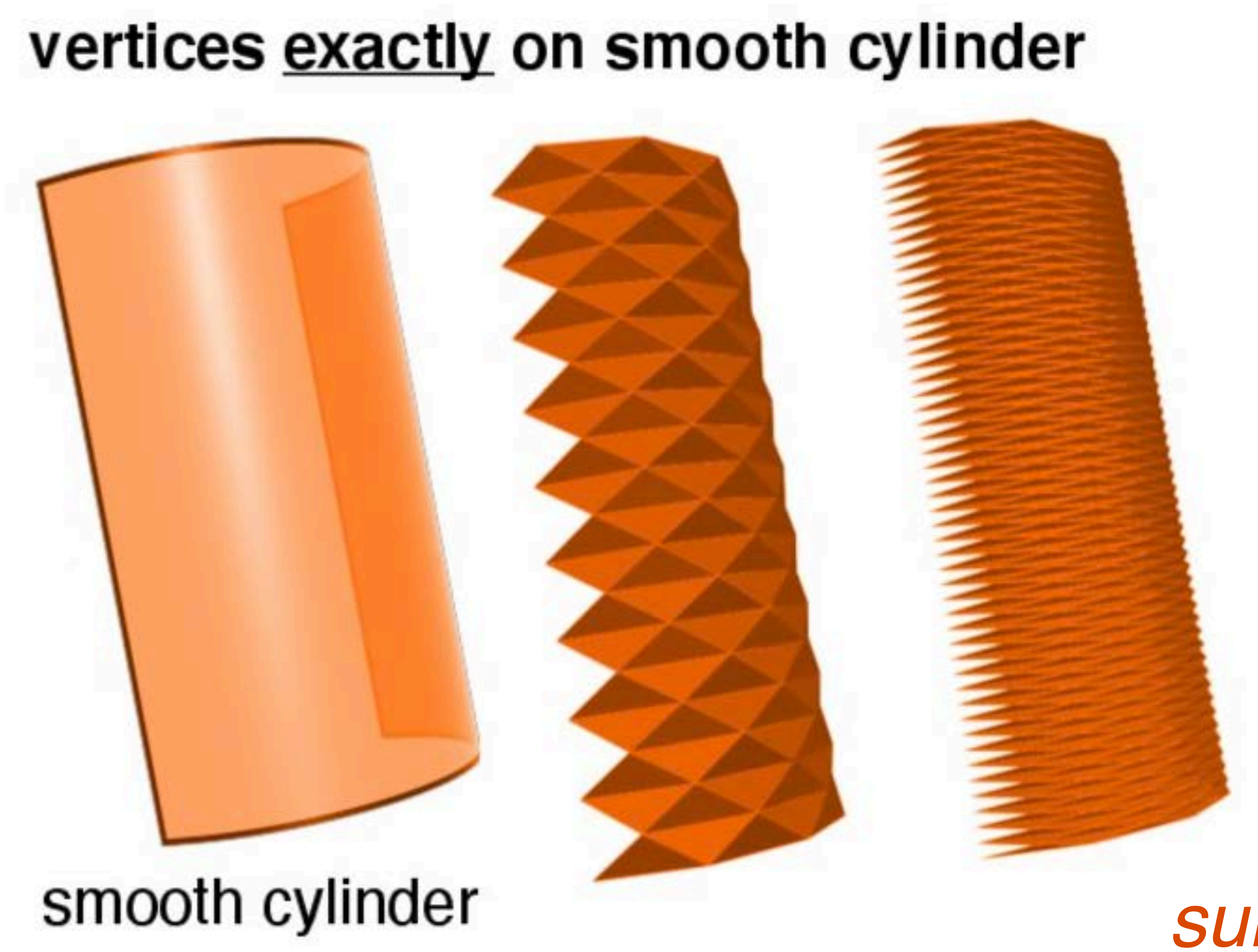
What is a “good” mesh?

- One idea: good approximation of the original shape!
- Keep only elements that contribute information about shape
- Add element where, e.g., curvature is large



What is a "good" mesh?

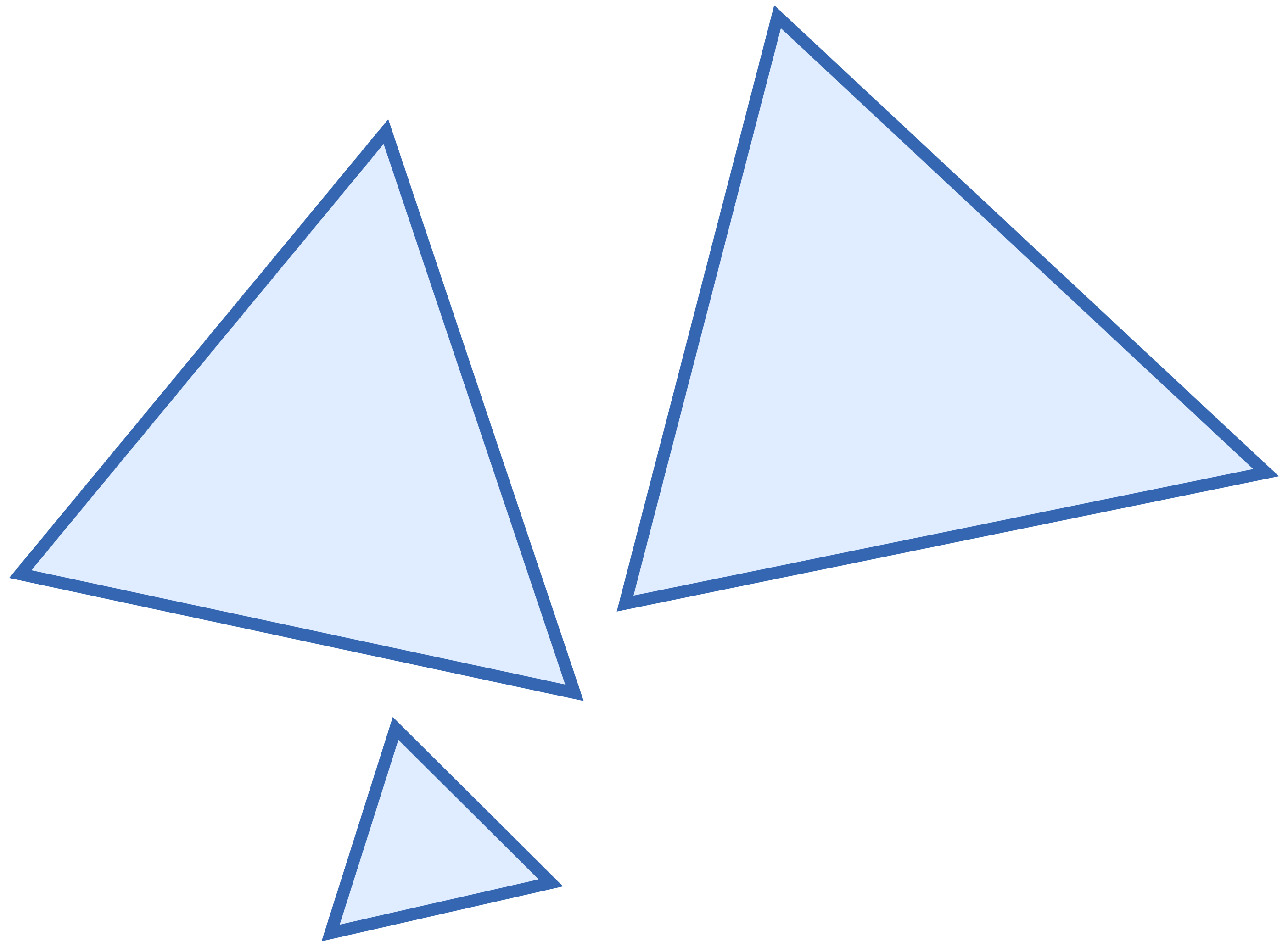
- One idea: good approximation of the original shape!
- Keep only elements that contribute information about shape
- Add element where, e.g., curvature is large
- "Good approximation" can be deceiving sometimes



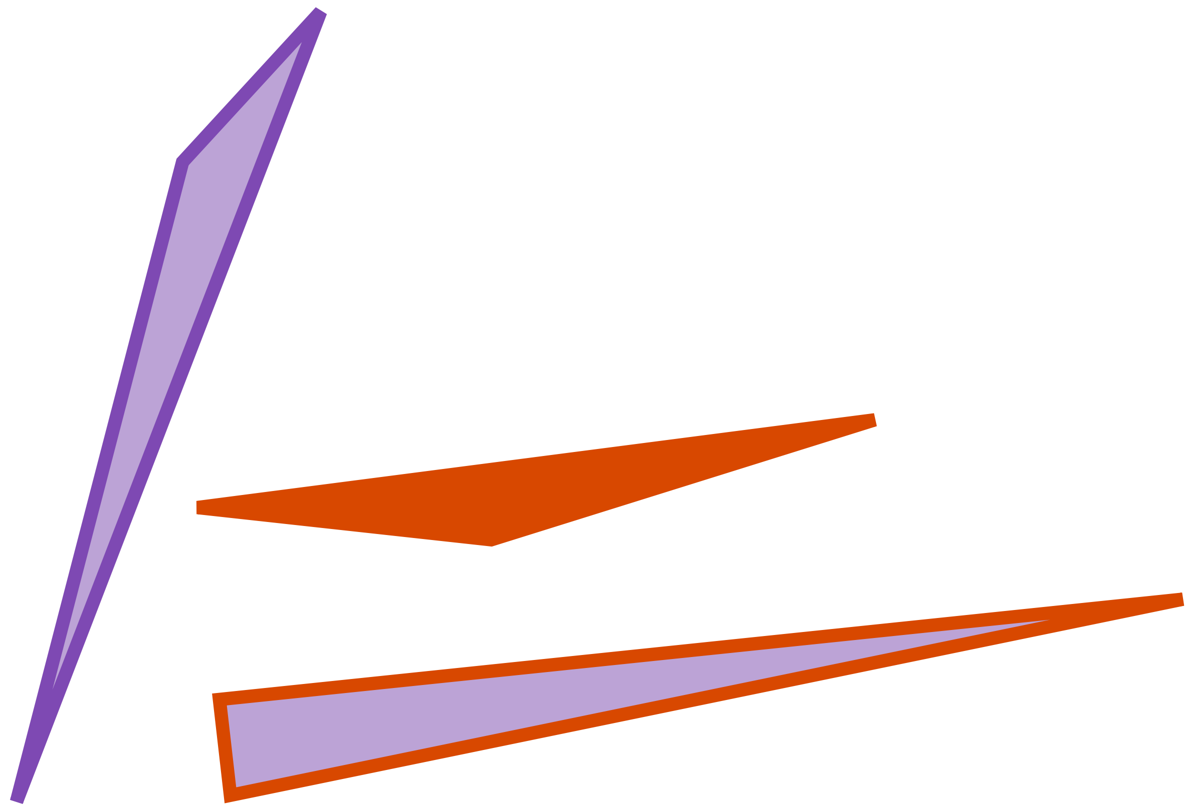
surface area doesn't converge under refinement

What is a “good” mesh?

- Another rule of thumb: *triangle shape*



“Good”

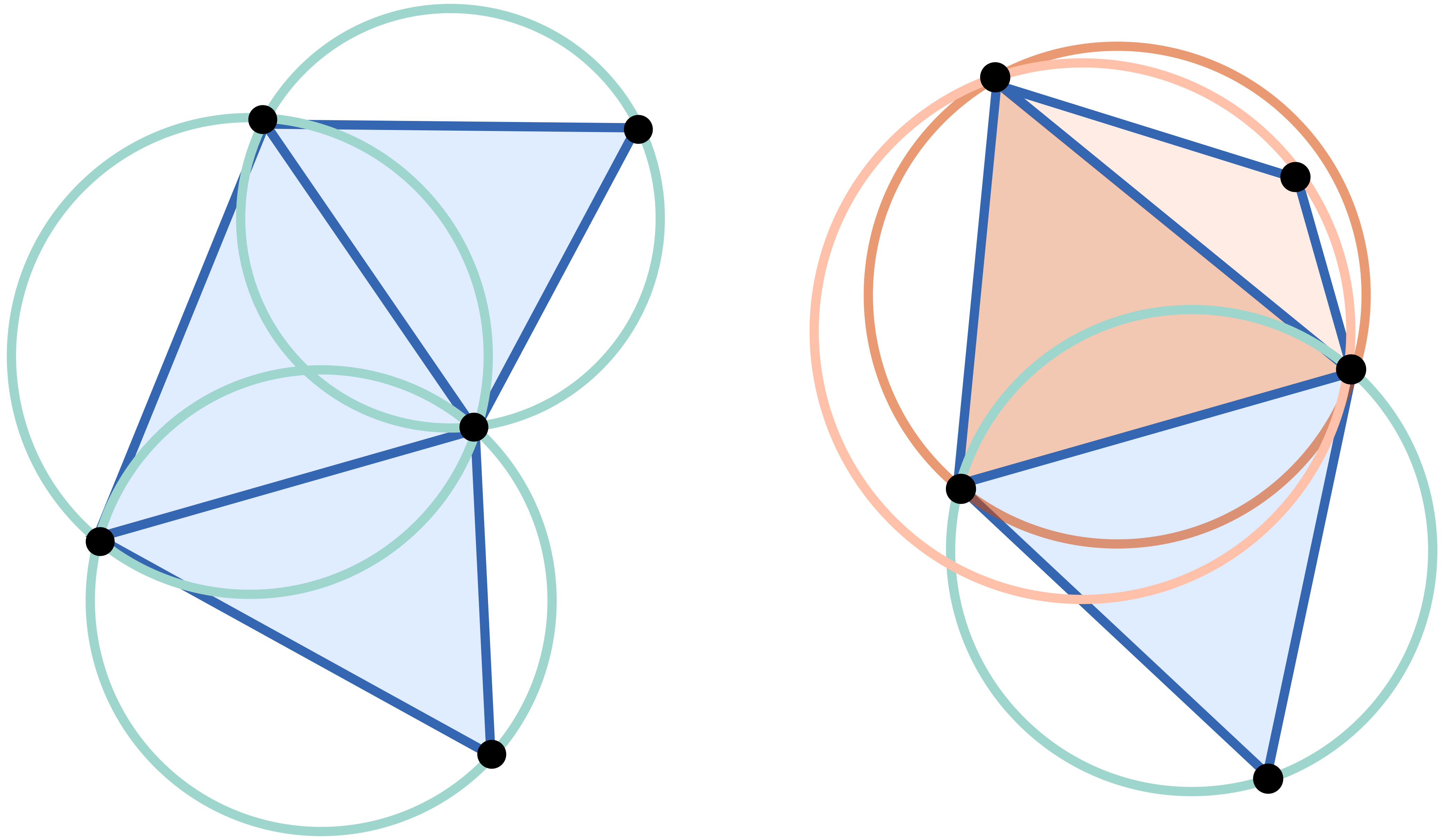


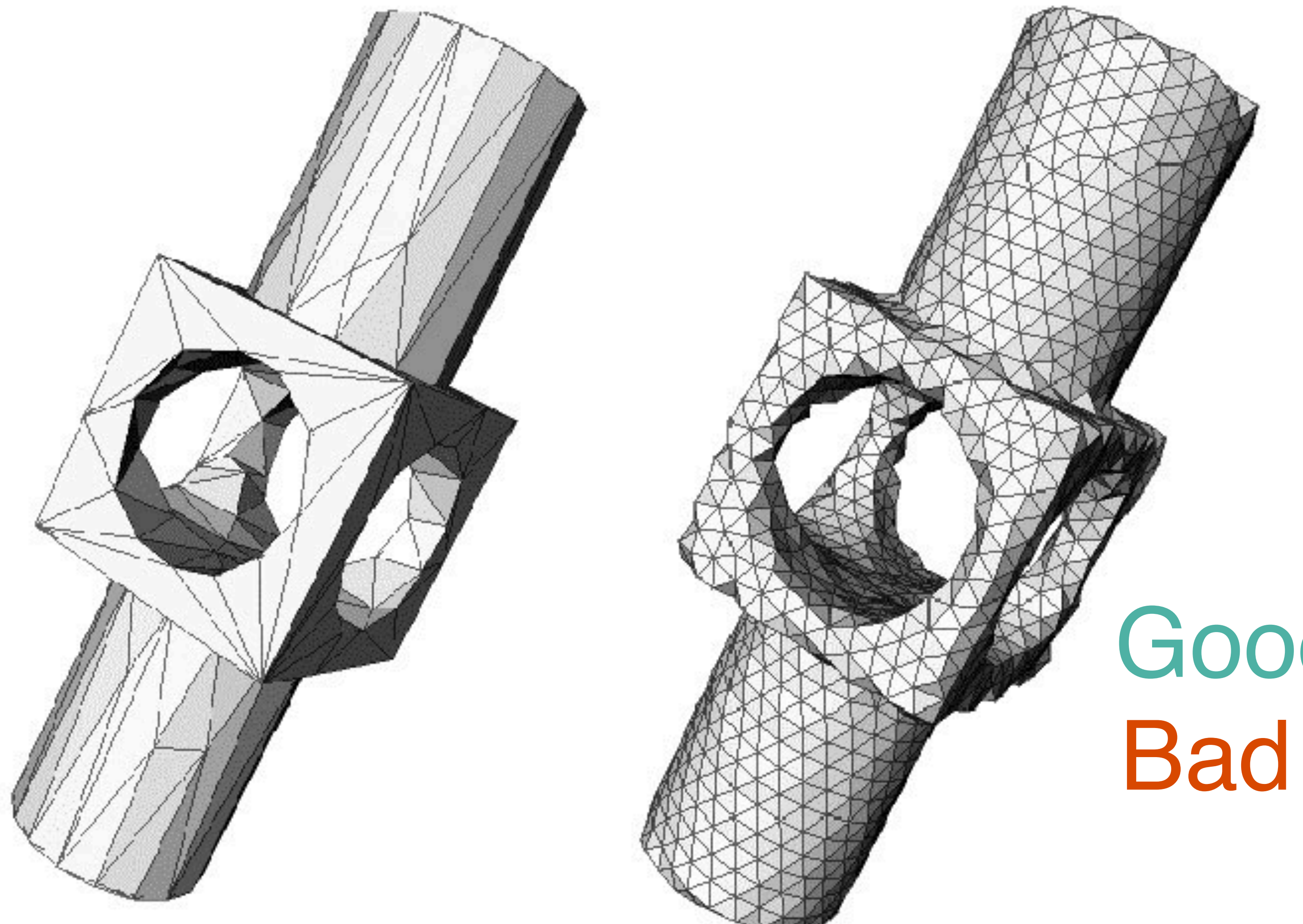
“Bad”

- For example, all angles close to 60 degrees
- A concrete characterization: Delaunay condition

Delaunay condition

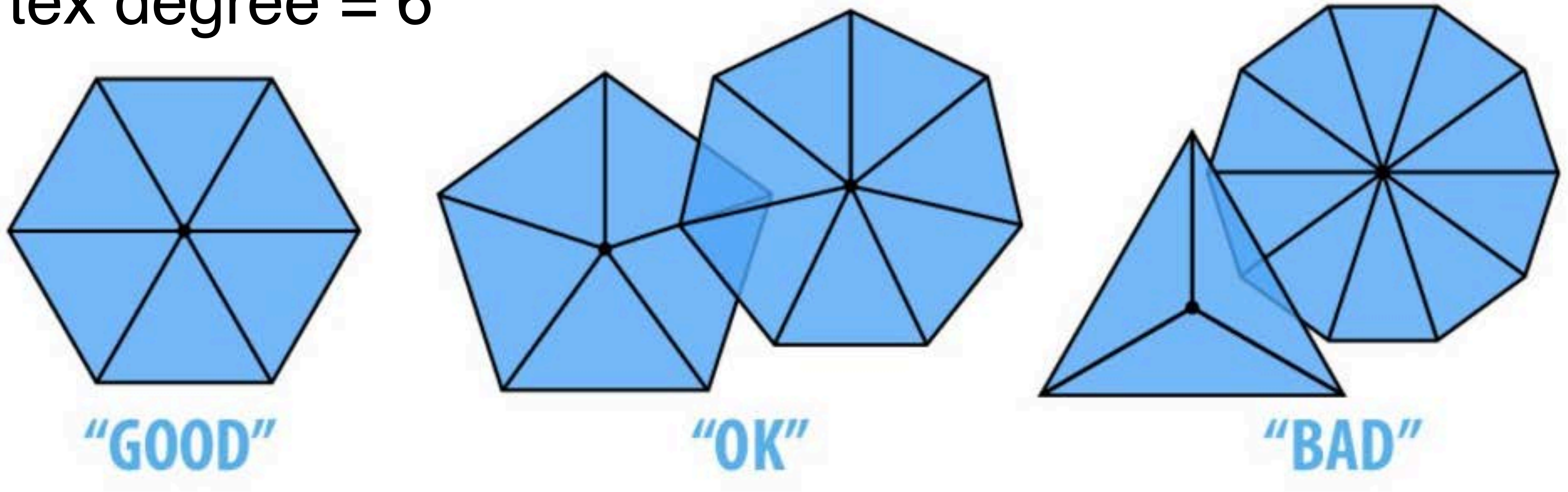
- A triangle mesh is **Delaunay** if the circumcircle of each triangle does not contain any vertex of any adjacent triangle
- Many desirable properties
  - Helps numerical accuracy / stability
  - Maximizes minimal angle
  - Smoothest linear interpolation
- Tradeoffs with efficient shape approximation

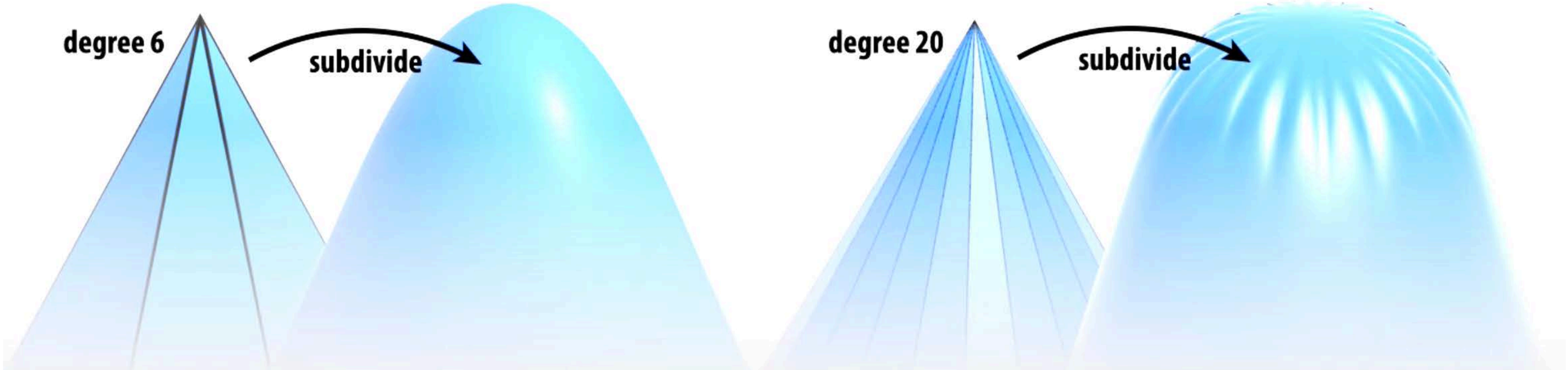




What is a “good” mesh?

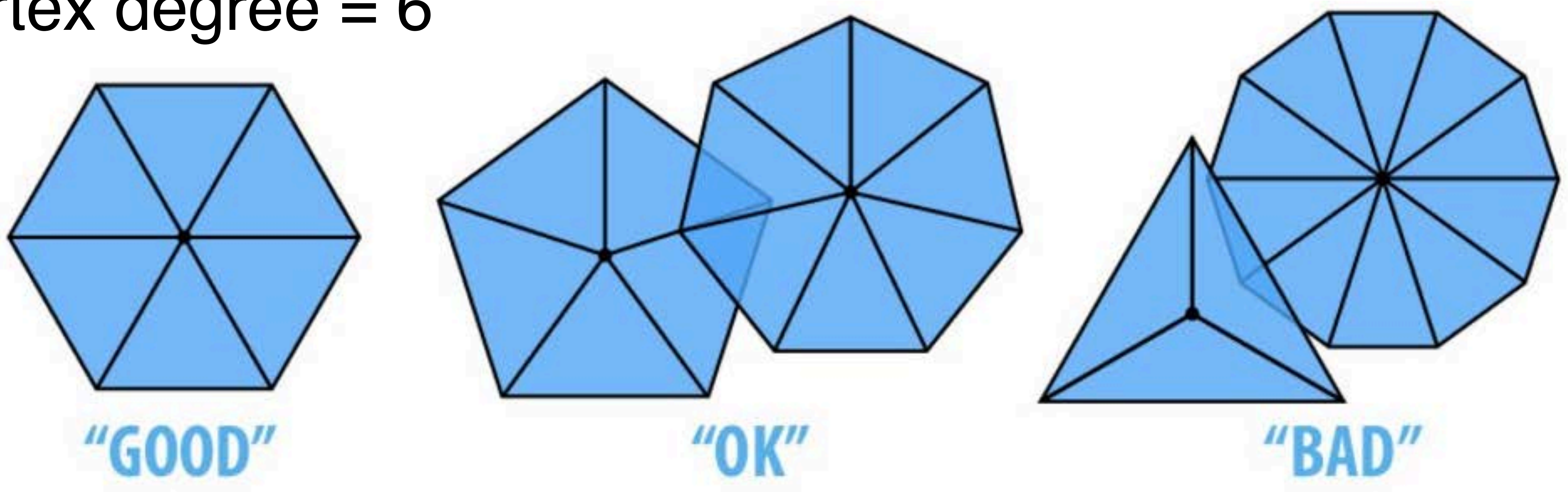
- Another rule of thumb: *regular vertex degree (valence)*
  - Regular: Vertex degree = 6





What is a "good" mesh?

- Another rule of thumb: *regular vertex degree (valence)*
  - Regular: Vertex degree = 6

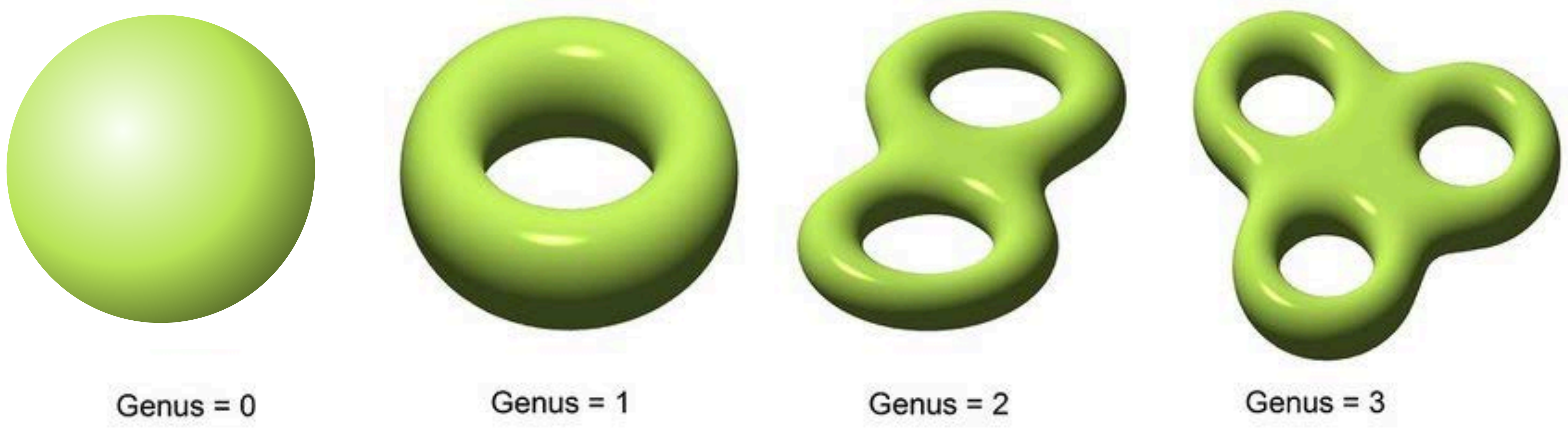


- It may be impossible to have all vertex degree = 6

  Euler–Poincaré Theorem

  #(Vertices) – #(Edges) + #(Faces) = 2 – 2 genus

- If all vertices are regular, then genus must be 1

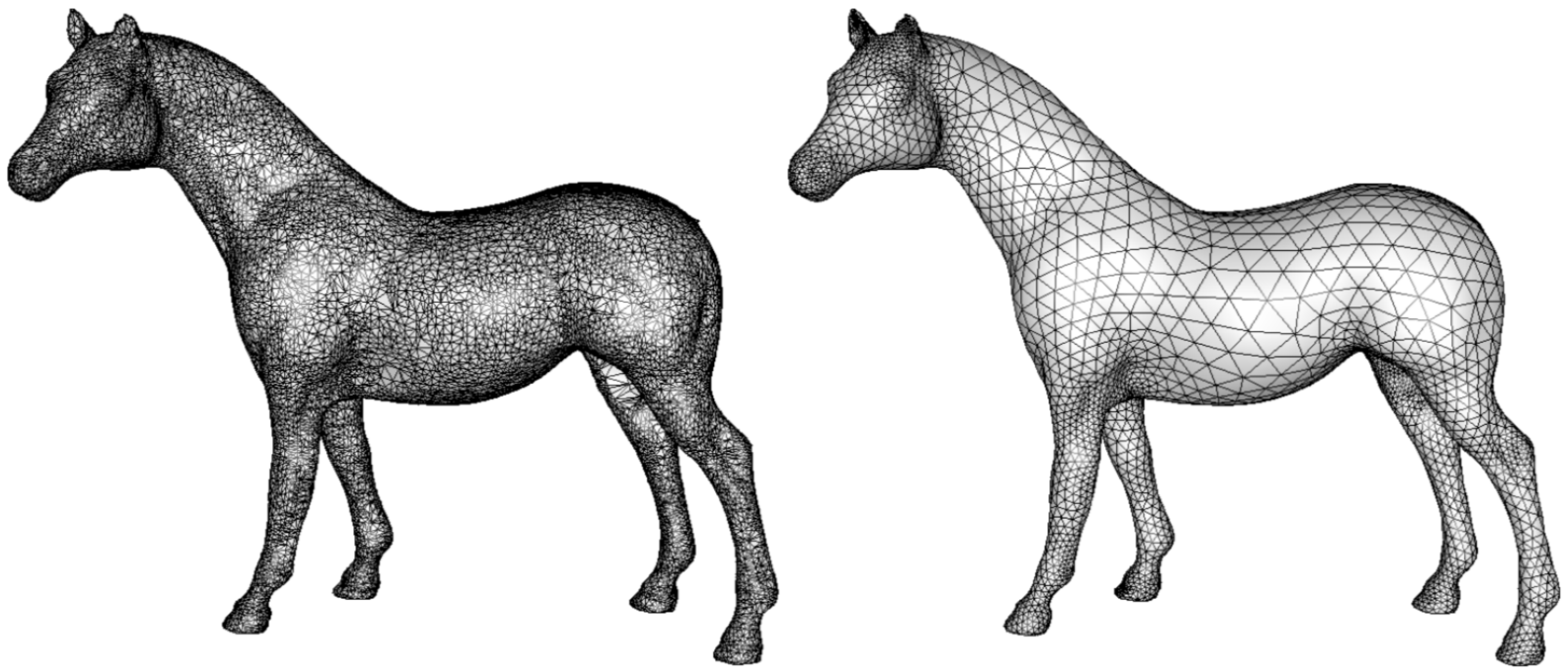


Remeshing

- General objectives of re-meshing
  - Shape approximation
  - As Delaunay (or equilateral-triangle) as possible
  - Vertex degree as regular as possible

- Mesh simplification

- Improve mesh quality



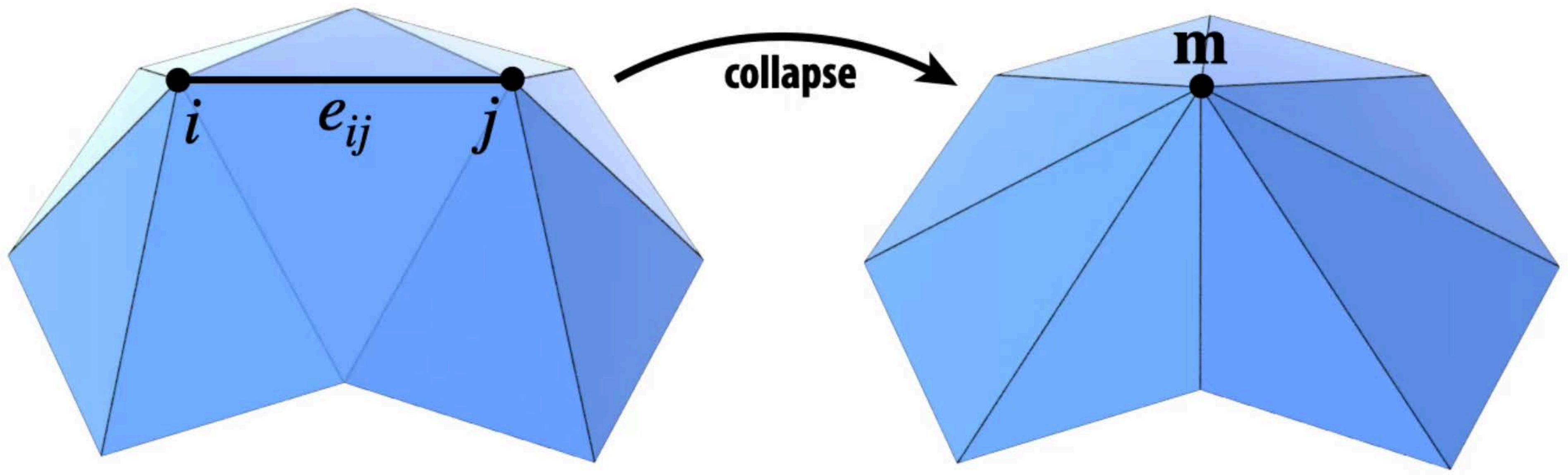
Remeshing

- Mesh simplification

- Improve mesh quality

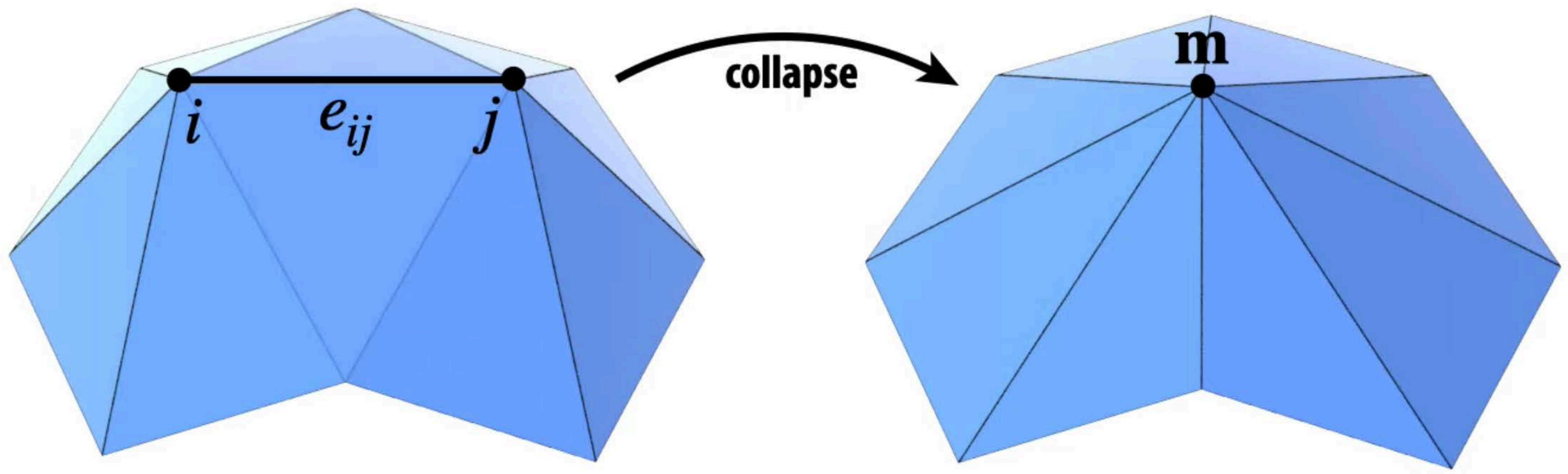
Mesh simplification

- Popular scheme: Iteratively collapse edges
- Greedy algorithm
  - Assign each edge a cost
  - Collapse edge with least cost
  - Repeat until target number of elements is reached
- Particularly effective cost function: **quadratic error metric**

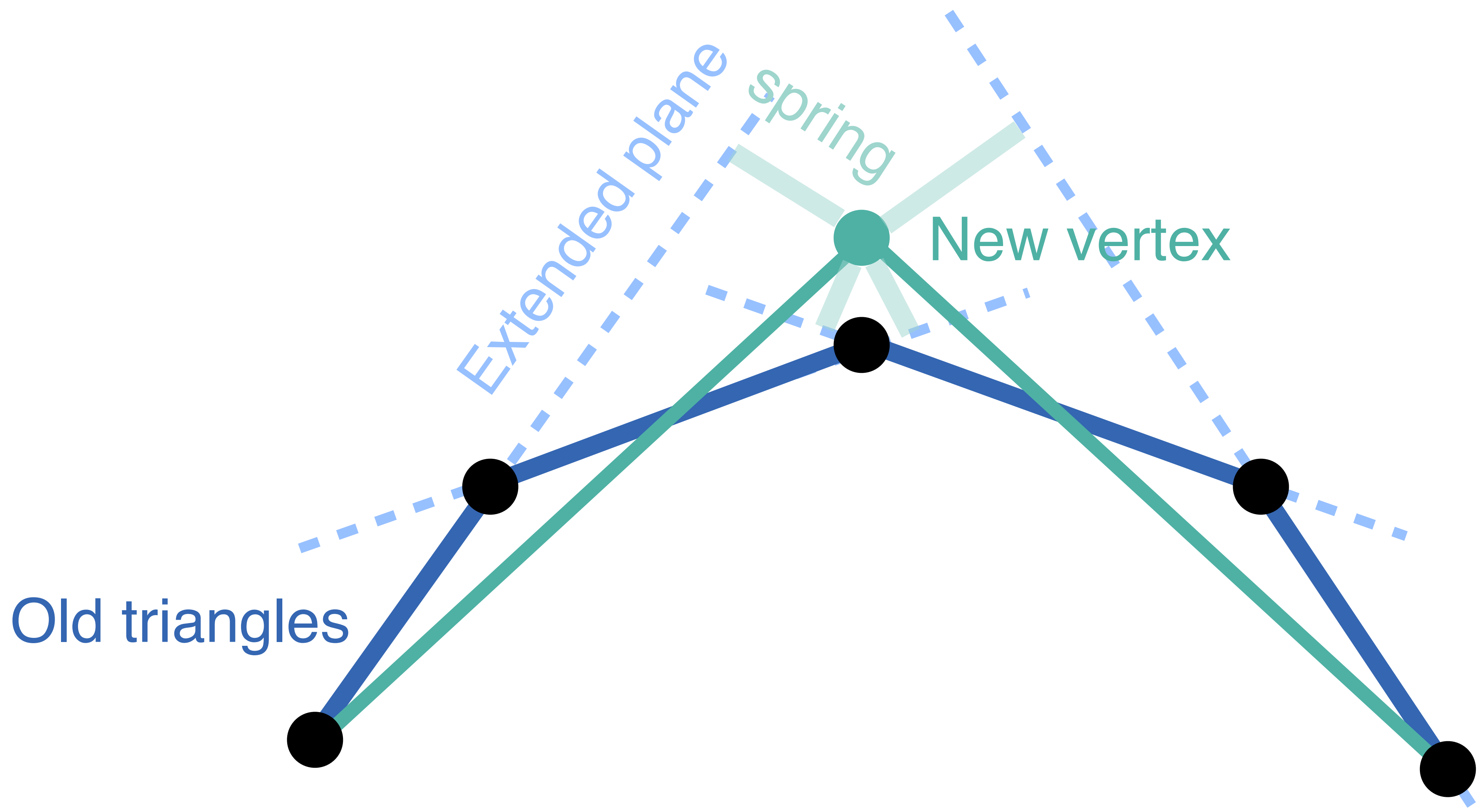


Mesh simplification

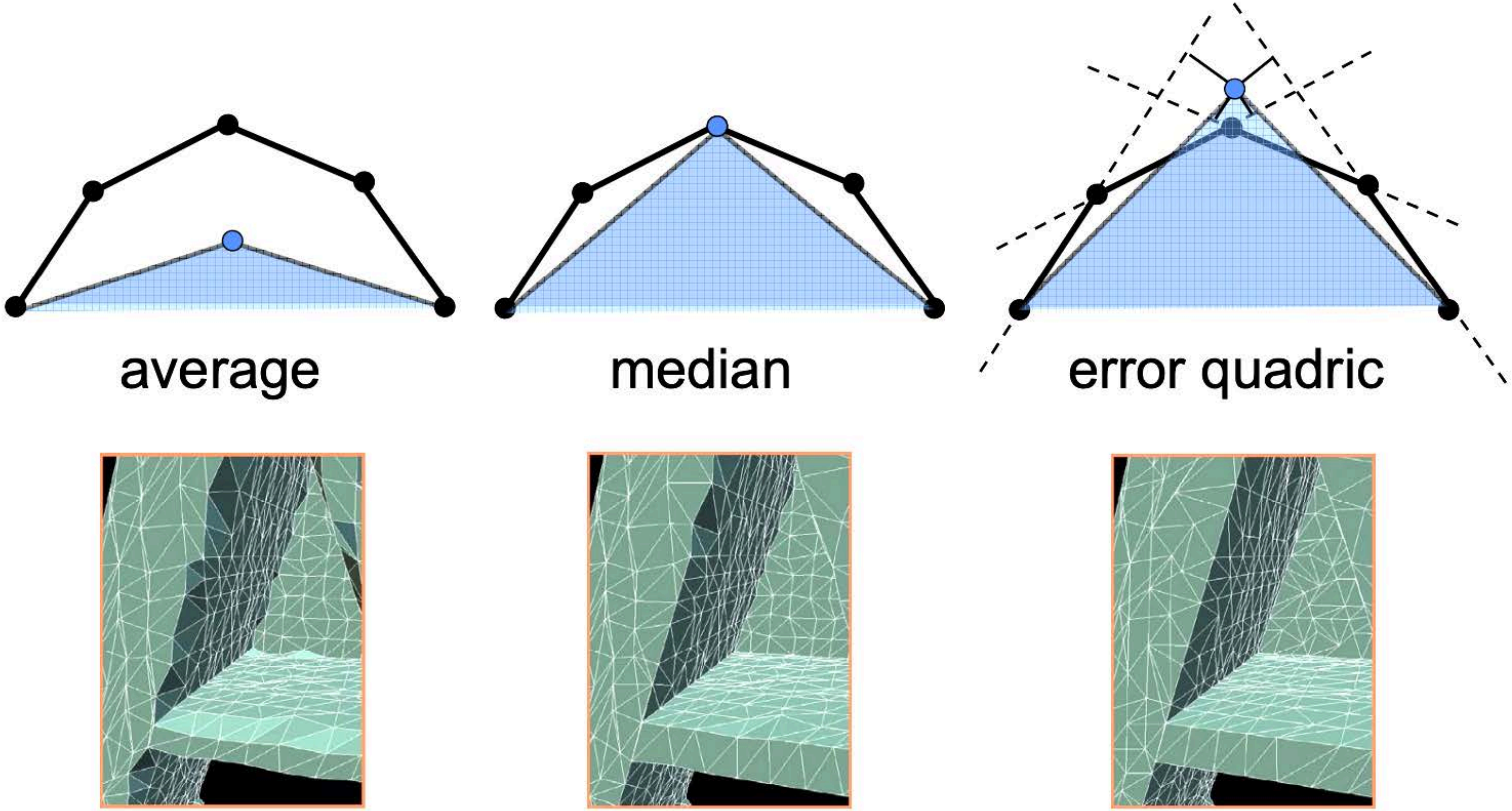
- If we would collapse this edge, where should we set the new vertex?



- Minimize the distance-squared to neighboring triangle planes



Mesh simplification



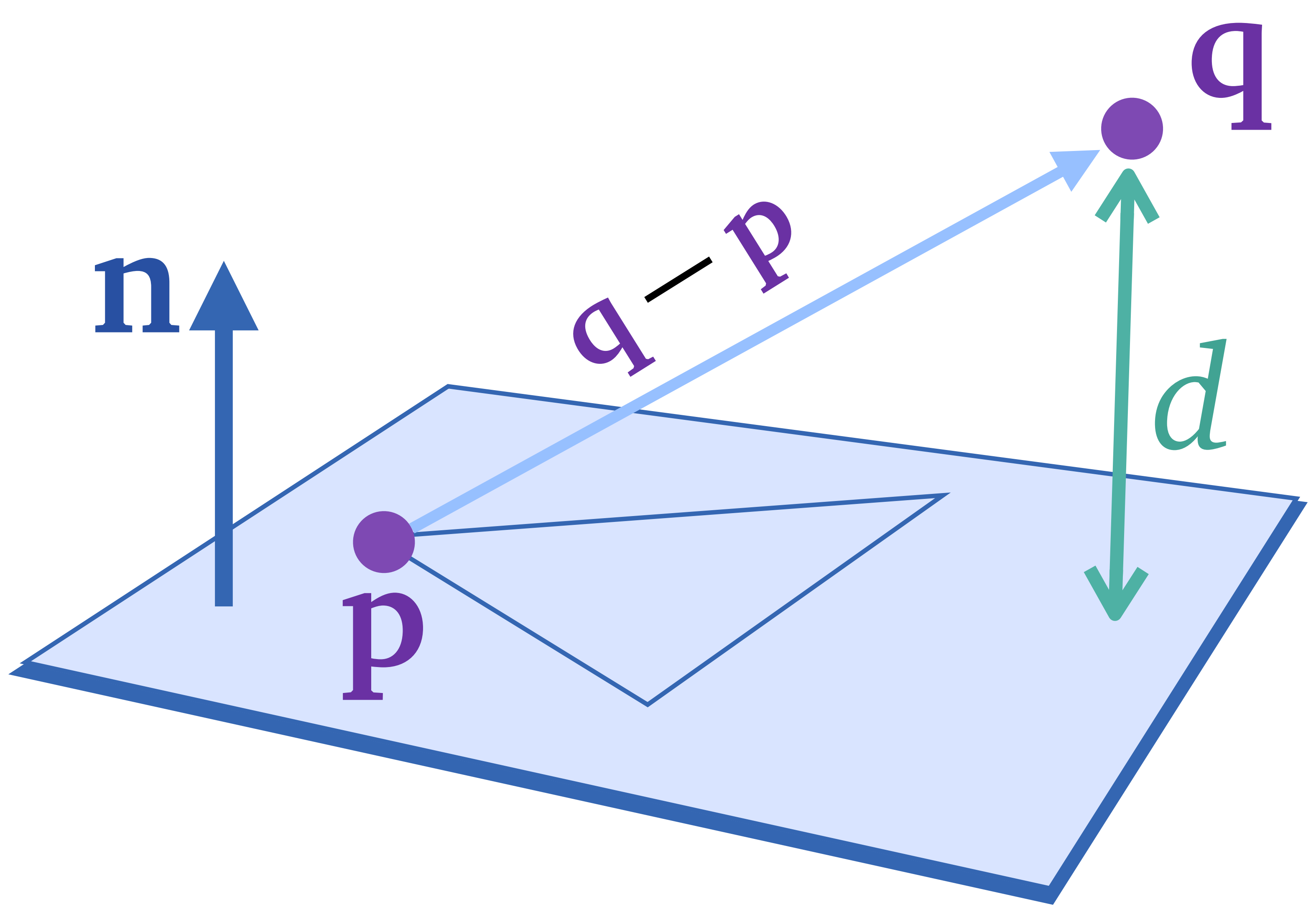
Mesh simplification

- What is the distance d between a point $\mathbf{q} \in \mathbb{R}^3$ and a plane?
- Suppose the plane passes through $\mathbf{p} \in \mathbb{R}^3$ with unit normal $\mathbf{n} \in \mathbb{S}^2$

$$d = \mathbf{n} \cdot (\mathbf{q} - \mathbf{p})$$

$$= \begin{bmatrix} — & \mathbf{n}^\mathsf{T} & — & (-\mathbf{n} \cdot \mathbf{p}) \end{bmatrix} \begin{bmatrix} | \\ \mathbf{q} \\ | \\ 1 \end{bmatrix}$$

$$= \mathbf{a}_{4\mathrm{D}}^\mathsf{T} \mathbf{q}_{4\mathrm{D}}$$



- Every plane is now a 4D row vector $\mathbf{a}_{4\mathrm{D}}^\mathsf{T}$

$$d^2 = \mathbf{q}_{4\mathrm{D}}{}^\mathsf{T} (\mathbf{a}_{4\mathrm{D}} \mathbf{a}_{4\mathrm{D}}^\mathsf{T}) \mathbf{q}_{4\mathrm{D}}$$

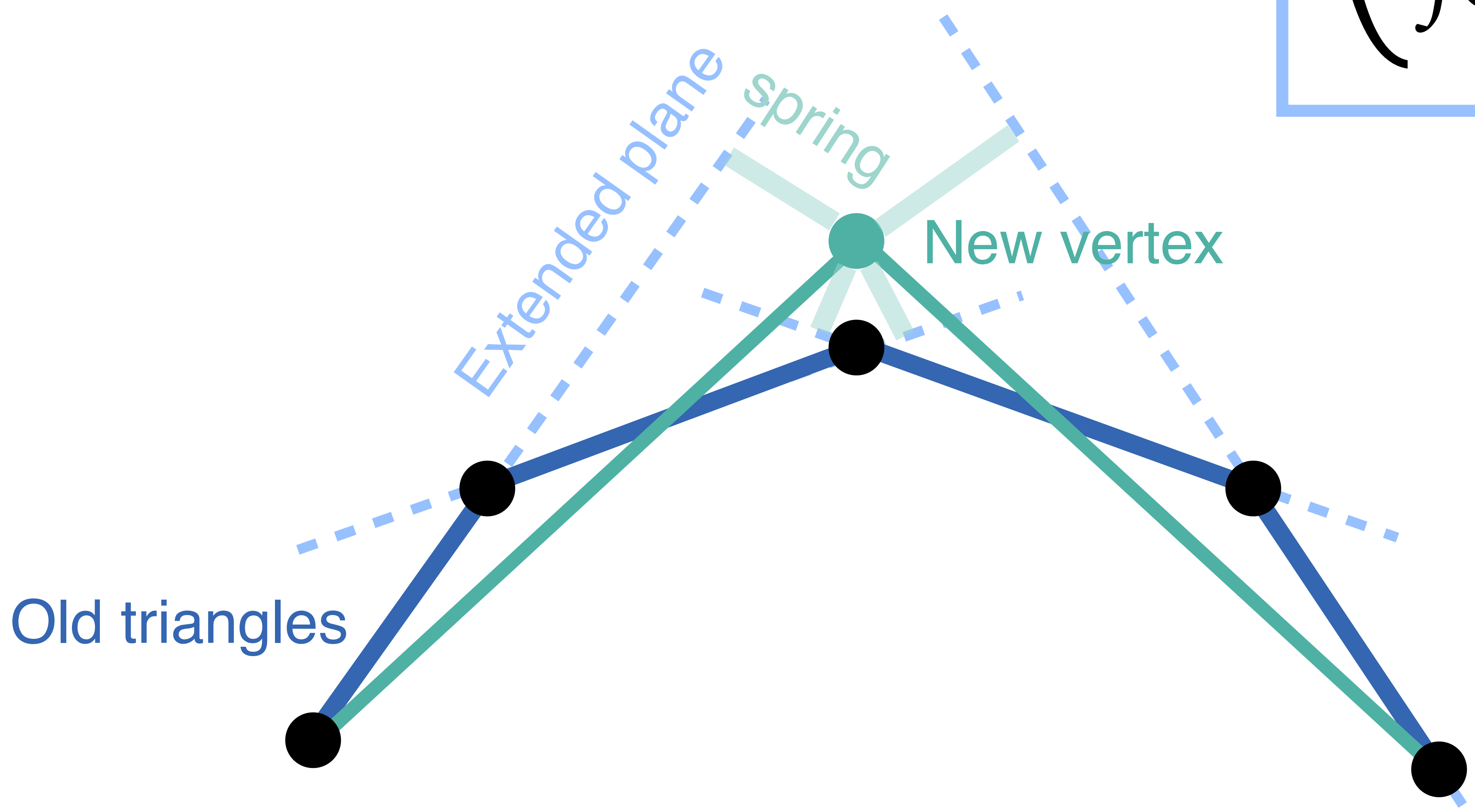
Mesh simplification

- What is the distance d between a point $\mathbf{q} \in \mathbb{R}^3$ and a plane?

$$d^2 = \mathbf{q}_{4D}{}^\intercal (\mathbf{a}_{4D} \mathbf{a}_{4D}^\intercal) \mathbf{q}_{4D}$$

- The total spring energy $U(\mathbf{q}) = \frac{1}{2} \mathbf{q}_{4D}^\intercal \left(\sum_{j \in \substack{\text{neighboring} \\ \text{triangles}}} \mathbf{a}_j \mathbf{a}_j^\intercal \right) \mathbf{q}_{4D}$

$\mathbf{K} \in \mathbb{R}^{4\times4}$

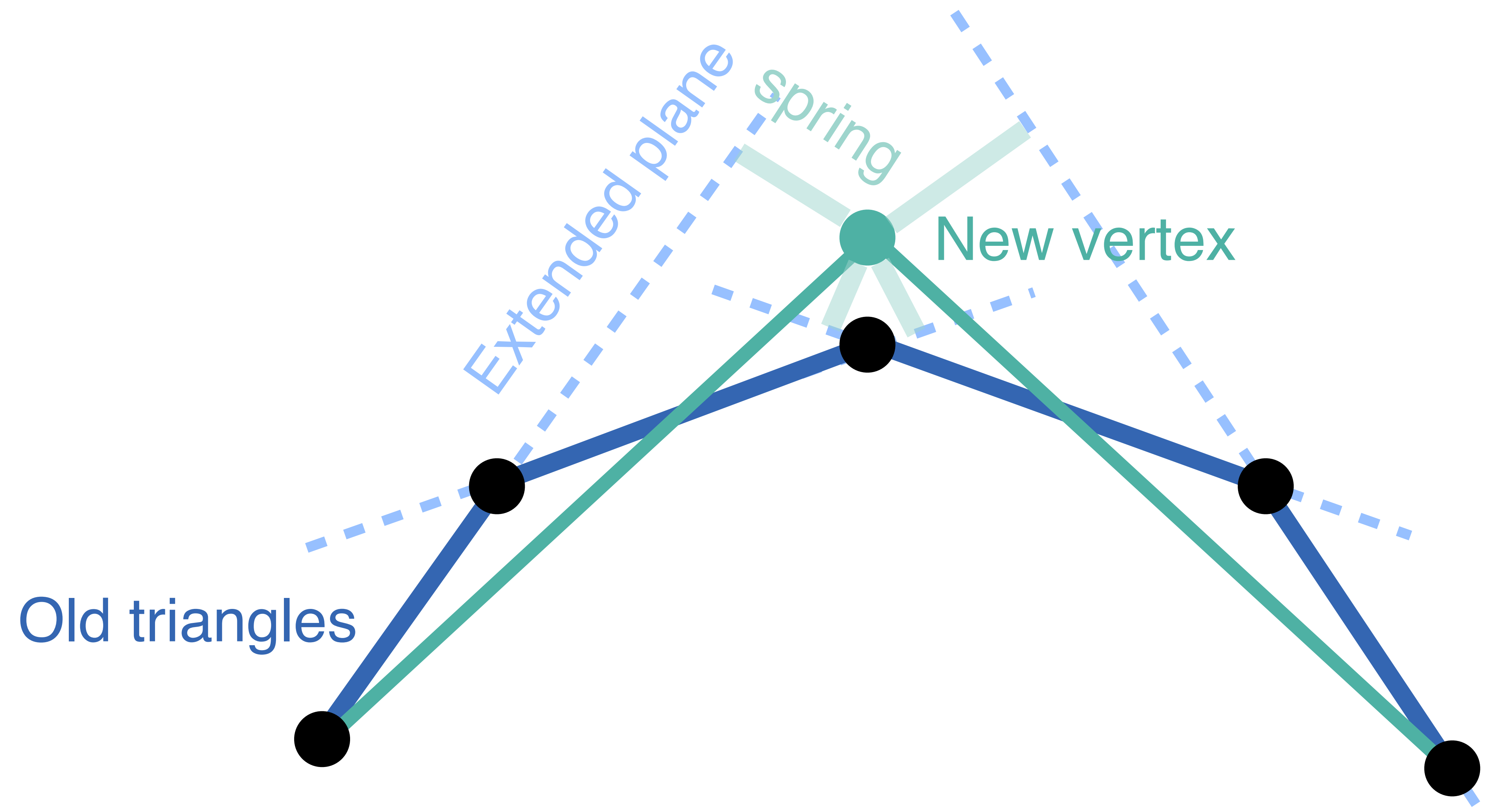


Mesh simplification

- The position $\mathbf{q} \in \mathbb{R}^3$ that minimizes the spring energy

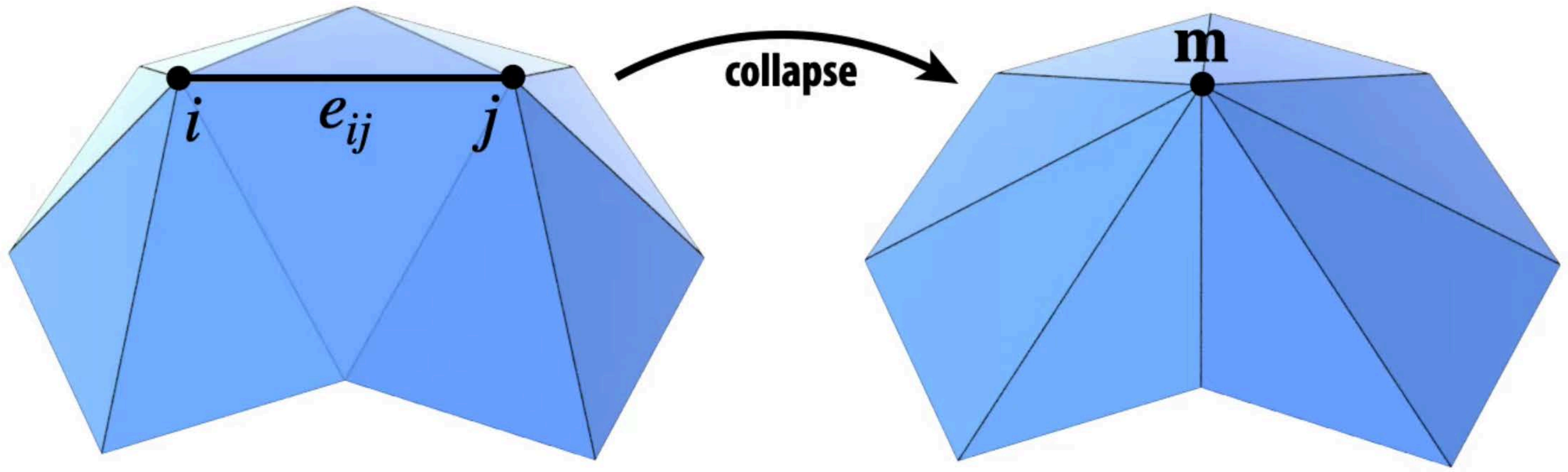
$$U(\mathbf{q}) = \frac{1}{2} \begin{bmatrix} \mathbf{q}^\intercal & 1 \end{bmatrix} \underbrace{\begin{bmatrix} \\ \mathbf{K}_{4\times 4} \\ \\ \end{bmatrix}}_{\begin{bmatrix} \mathbf{A}_{3\times 3} & \mathbf{b}_{3\times 1} \\ \mathbf{b}^\intercal & c \end{bmatrix}} \begin{bmatrix} \mathbf{q} \\ 1 \end{bmatrix}$$

is the solution to $\mathbf{A}\mathbf{q} = -\mathbf{b}$



Mesh simplification

Algorithm (Assign energy per edge)



- For each edge
  - Compute the spring matrix **K**

$$U(\mathbf{q}) = \frac{1}{2}\mathbf{q}_{4D}^{\mathsf{T}} \left(\sum_{j \in \text{neighboring triangles}} \mathbf{a}_j \mathbf{a}_j^{\mathsf{T}} \right) \mathbf{q}_{4D}$$

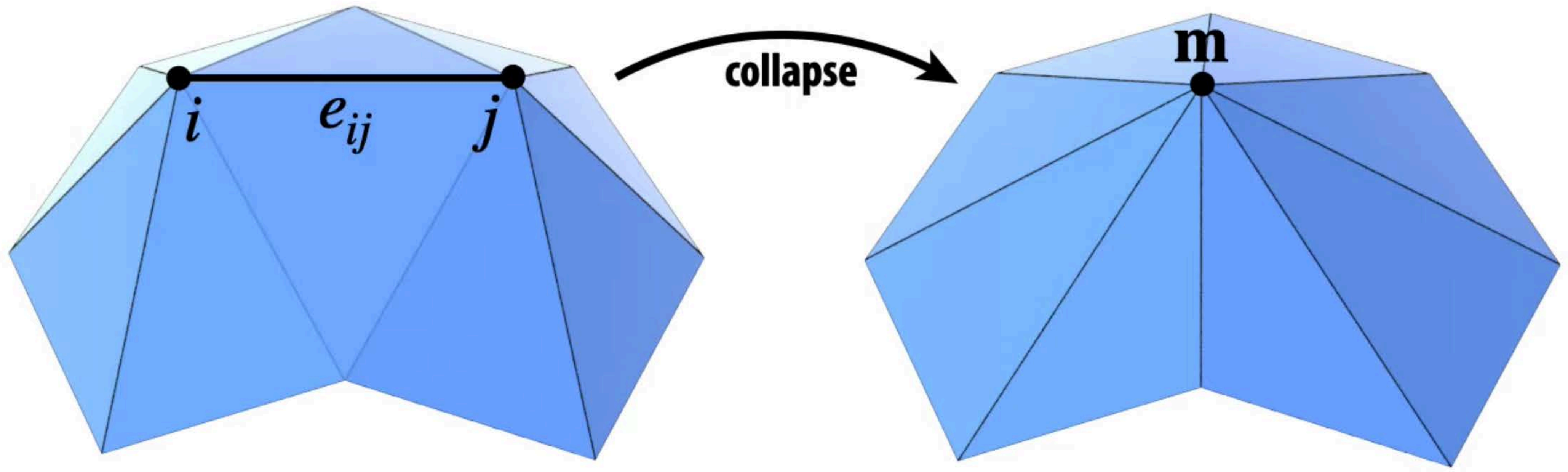
$$\mathbf{K} \in \mathbb{R}^{4\times 4}$$

$$\mathbf{K} = \begin{bmatrix} \mathbf{A}_{3\times 3} & \mathbf{b}_{3\times 1} \\ \mathbf{b}^{\mathsf{T}} & c \end{bmatrix}$$

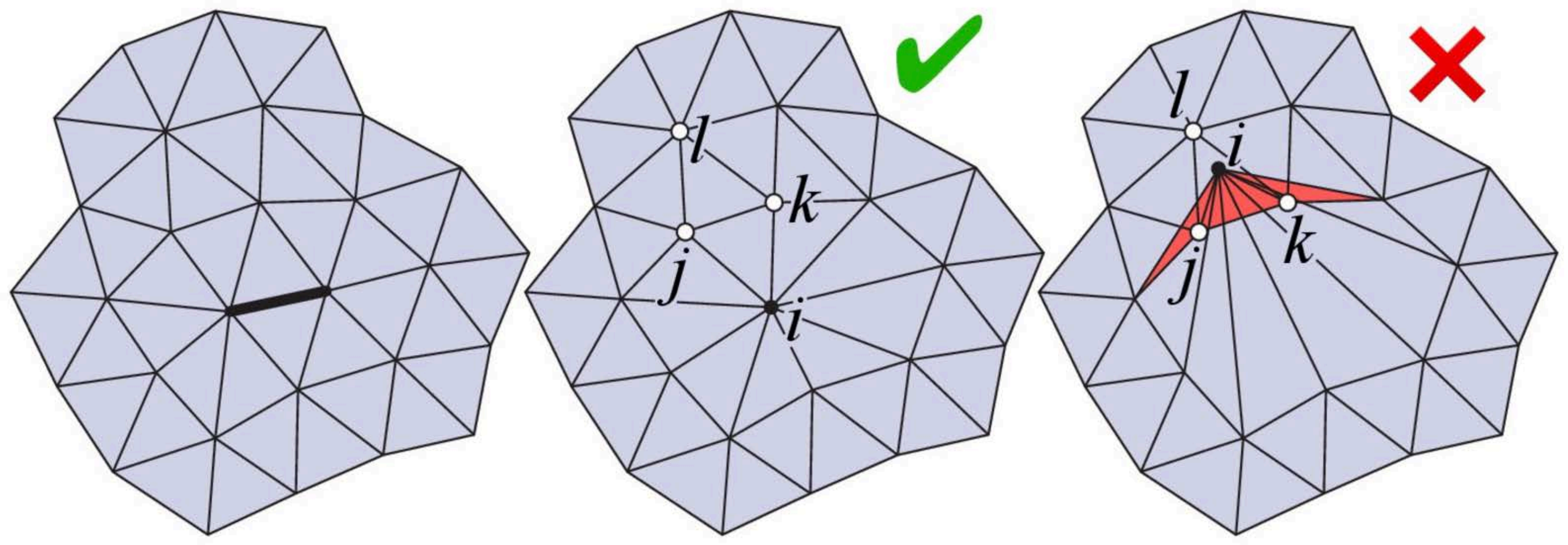
  - Compute the optimal new vertex position $\mathbf{q} = -\mathbf{A}^{-1}\mathbf{b}$
  - Assign the edge energy as $U(\mathbf{q})$
- EndFor

Mesh simplification

Algorithm (Edge Collapse)



- Assign each edge a cost
- Collapse edge with least cost
  - Exclude the cases where the triangles fold over
- Repeat until target number of elements is reached



Remeshing

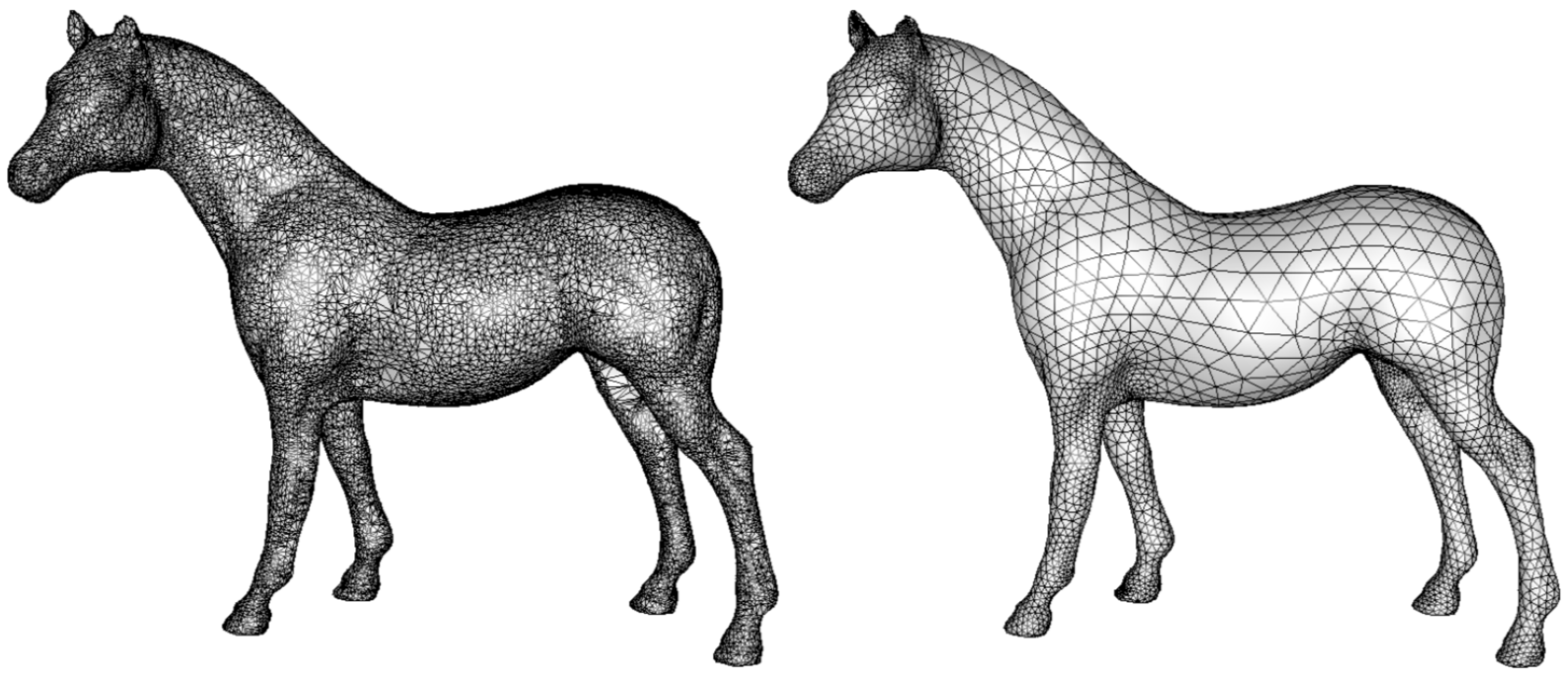
- Mesh simplification

- Improve mesh quality

Remeshing

- Mesh simplification

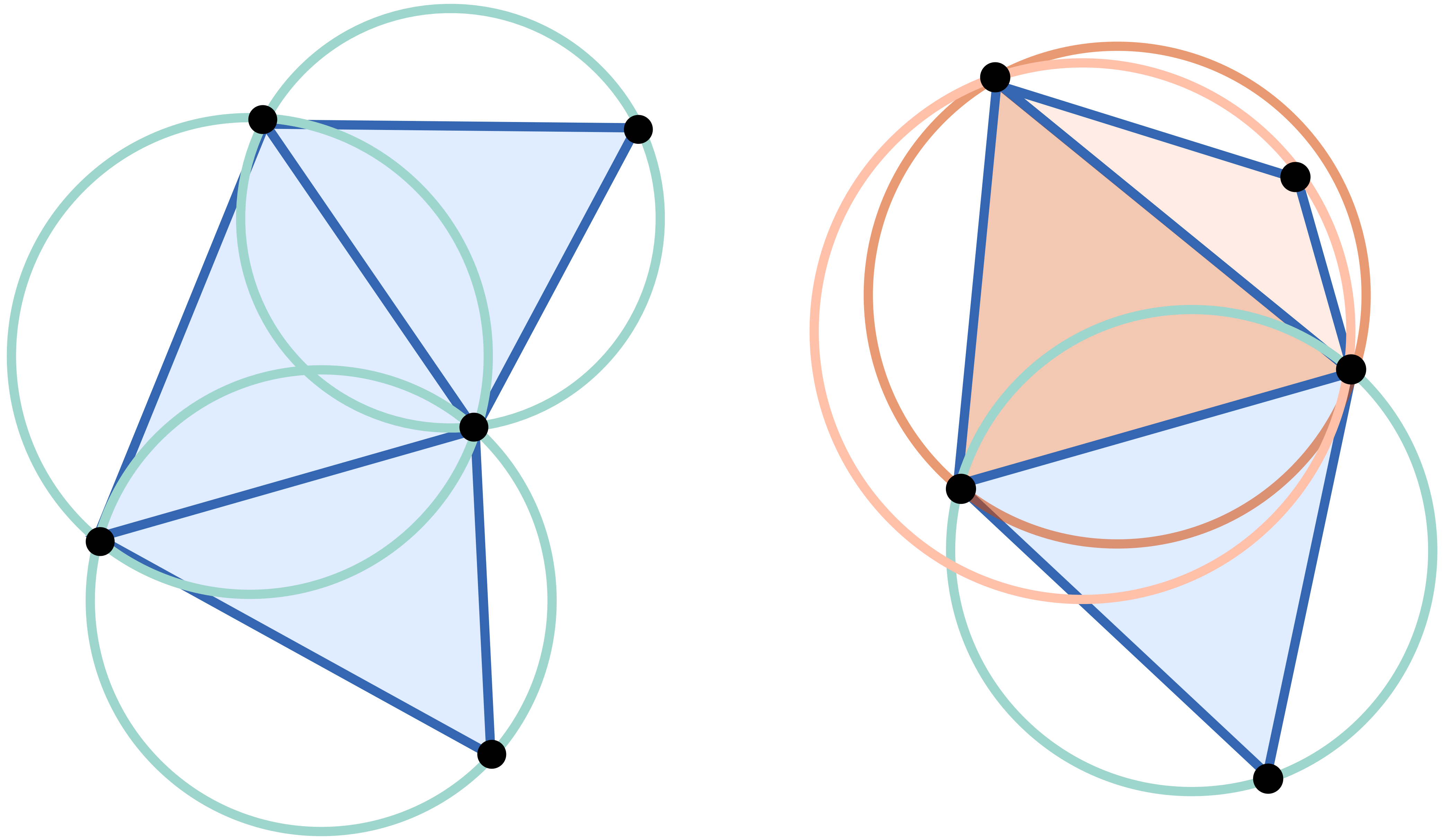
- Improve mesh quality



Make it more Delaunay

- A triangle mesh is **Delaunay** if the circumcircle of each triangle does not contain any vertex of any adjacent triangle

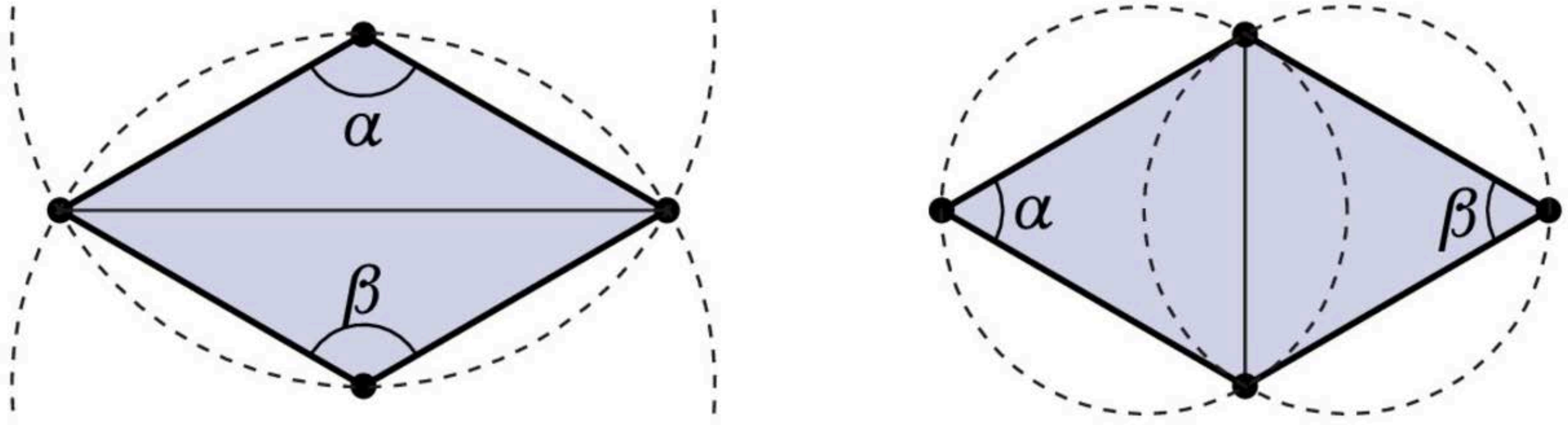
- How do we make a mesh "more Delaunay"?



Delaunay Non-Delaunay

Make it more Delaunay

- If $\alpha + \beta > 180^\circ$, flip the edge.

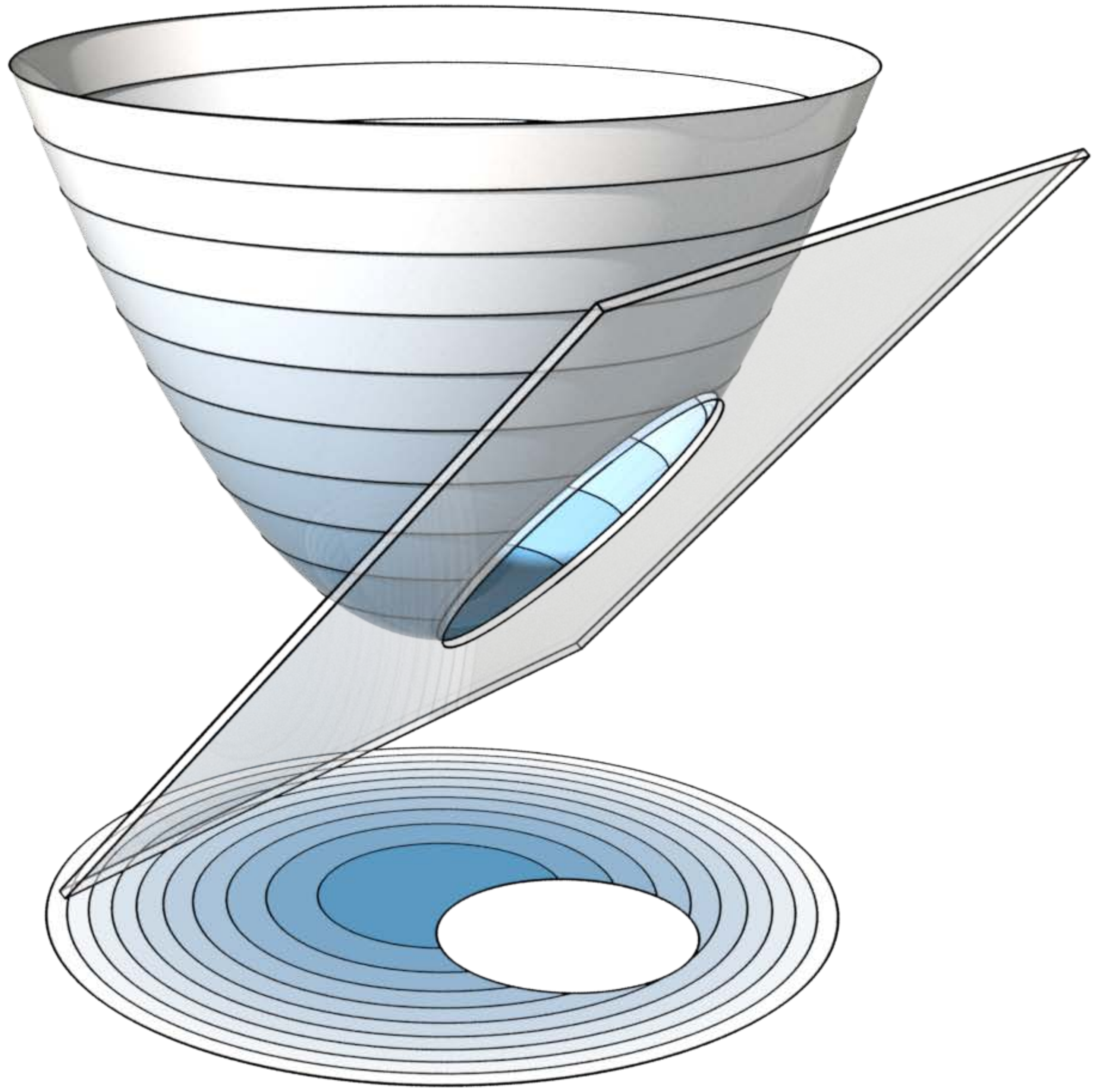


(Delaunay condition is equivalent to $\alpha + \beta \leq 180^\circ$ for all edges)

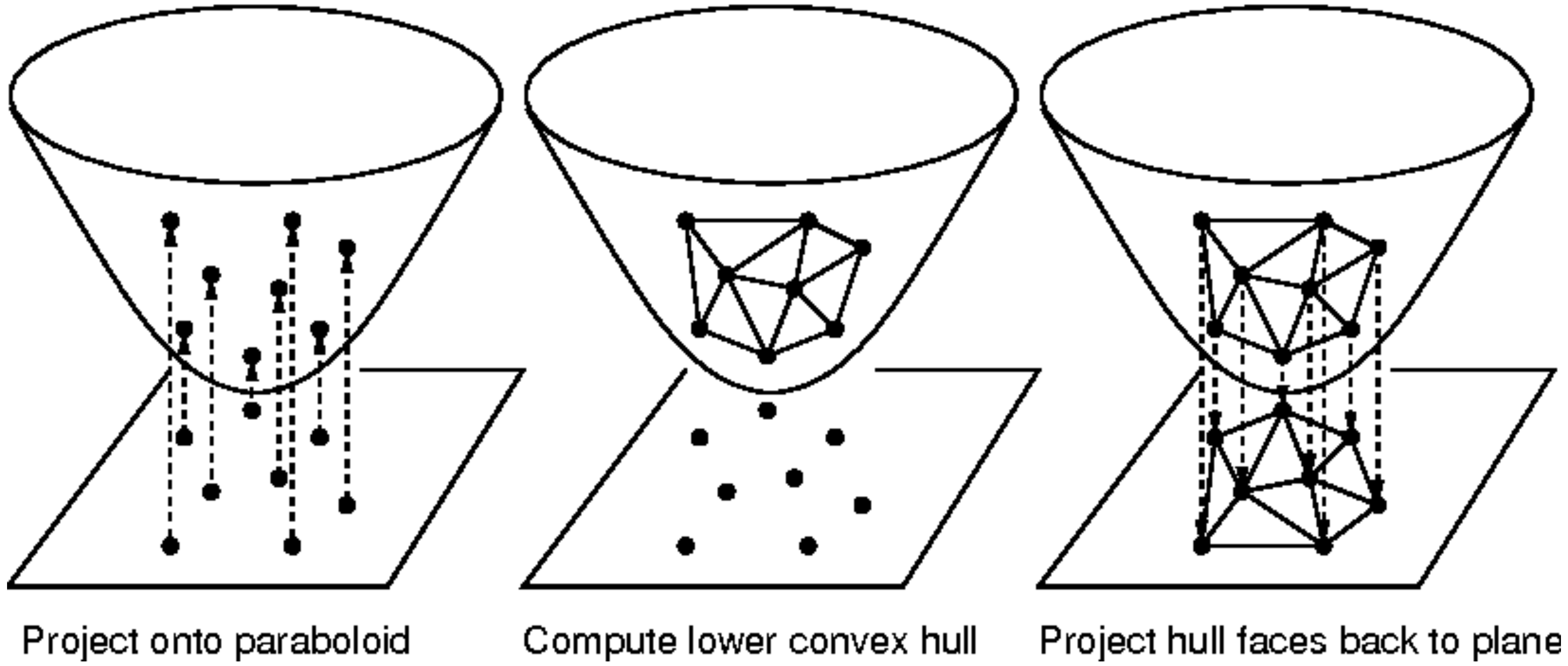
- For a planar (2D) mesh, this eventually yields Delaunay mesh
- For surfaces in 3D, this is still good heuristics in practice

Another algorithm for planar Delaunay

- Notice that every planar section of the paraboloid $z = x^2 + y^2$ is always circle when viewed from top



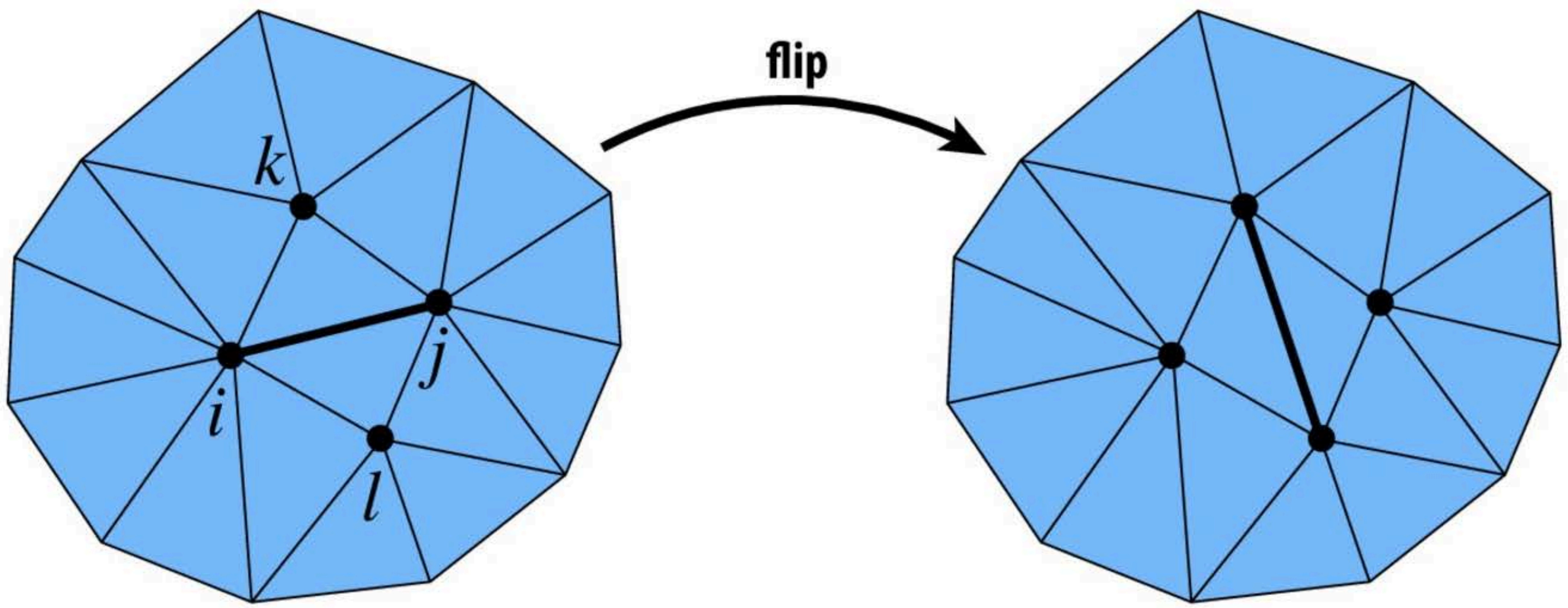
- Lift the vertices to the paraboloid



- Delaunay condition in the plane is equivalent to convexity of the lifted triangular mesh

Alternatively: Improve vertex degree

- How do we improve vertex degree?
- Same tool: Edge flip.
- If total deviation from 6 gets smaller, flip it!

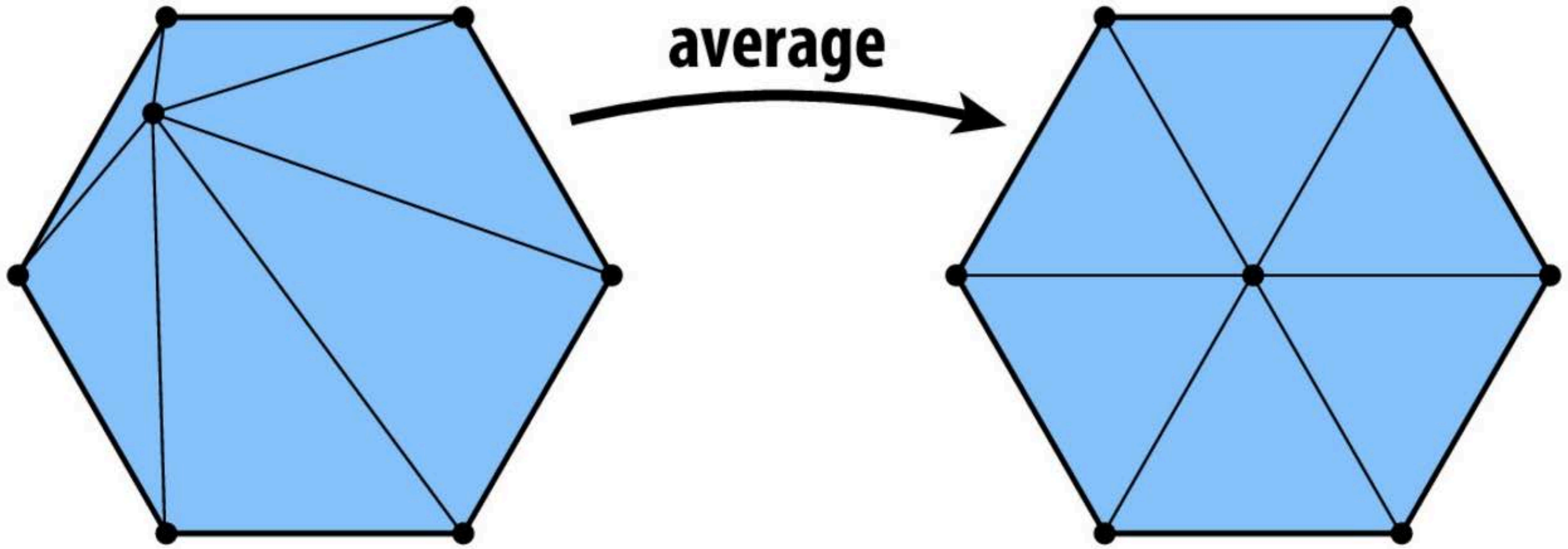


total deviation: $|d_i - 6| + |d_j - 6| + |d_k - 6| + |d_l - 6|$

- Works well in practice. No known guarantees.

Smoothing

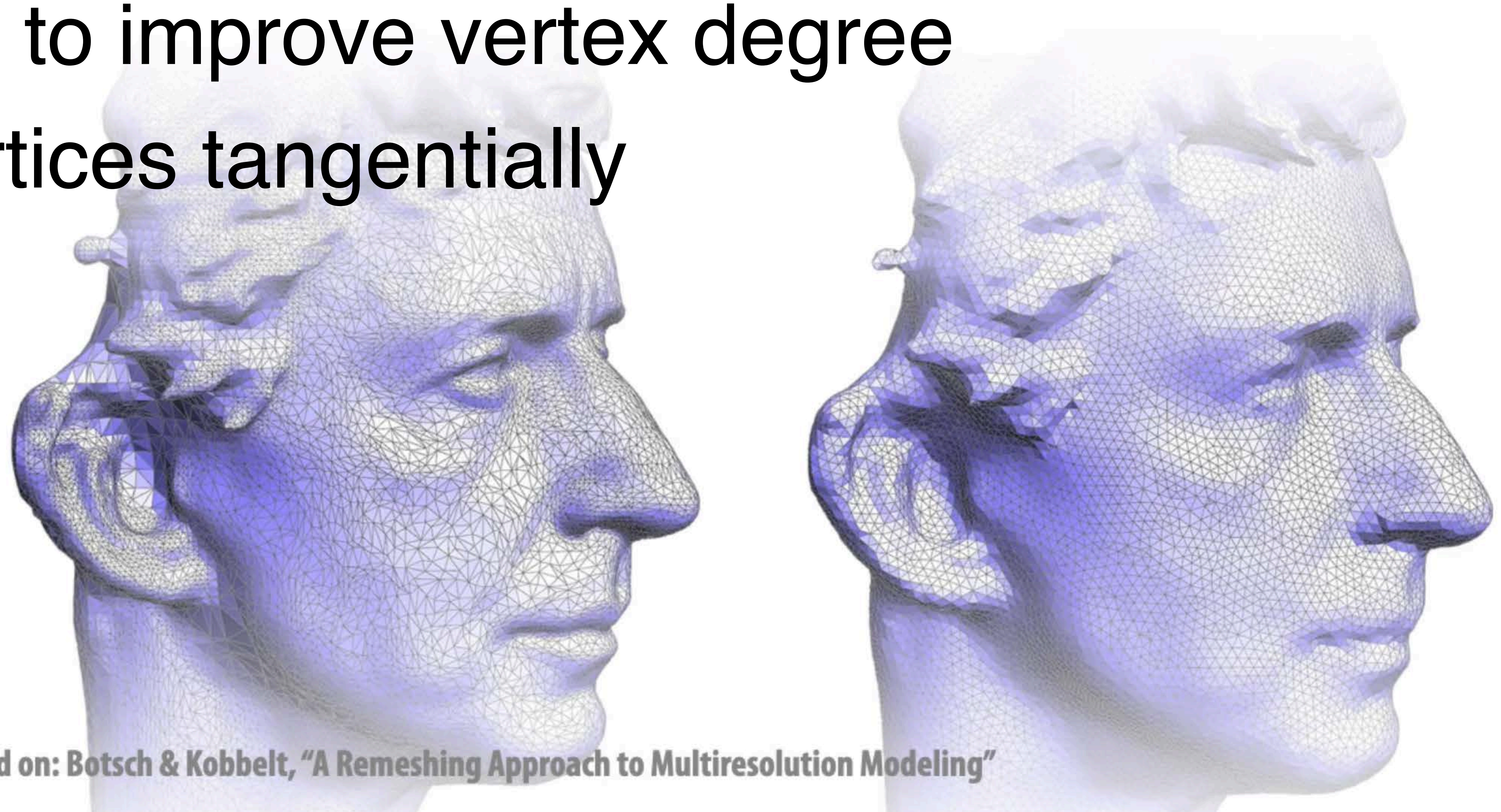
- Delaunay doesn't guarantee triangle angles are close to 60°
- Improve triangle shapes by centering vertices



- On surface: move only in the *tangent* direction
  - How? Remove the normal component of the update vector

Isotropic Remeshing Algorithm

- Put all tricks together: make triangles uniform in shape & size
- Repeat the 4 steps
  - Split any edge over 4/3rds mean edge length
  - Collapse edge less than 4/5ths mean edge length
  - Flip edges to improve vertex degree
  - Center vertices tangentially



Based on: Botsch & Kobbelt, "A Remeshing Approach to Multiresolution Modeling"

What we've learned today

- Surfaces are geometric signals
- Basic remeshing