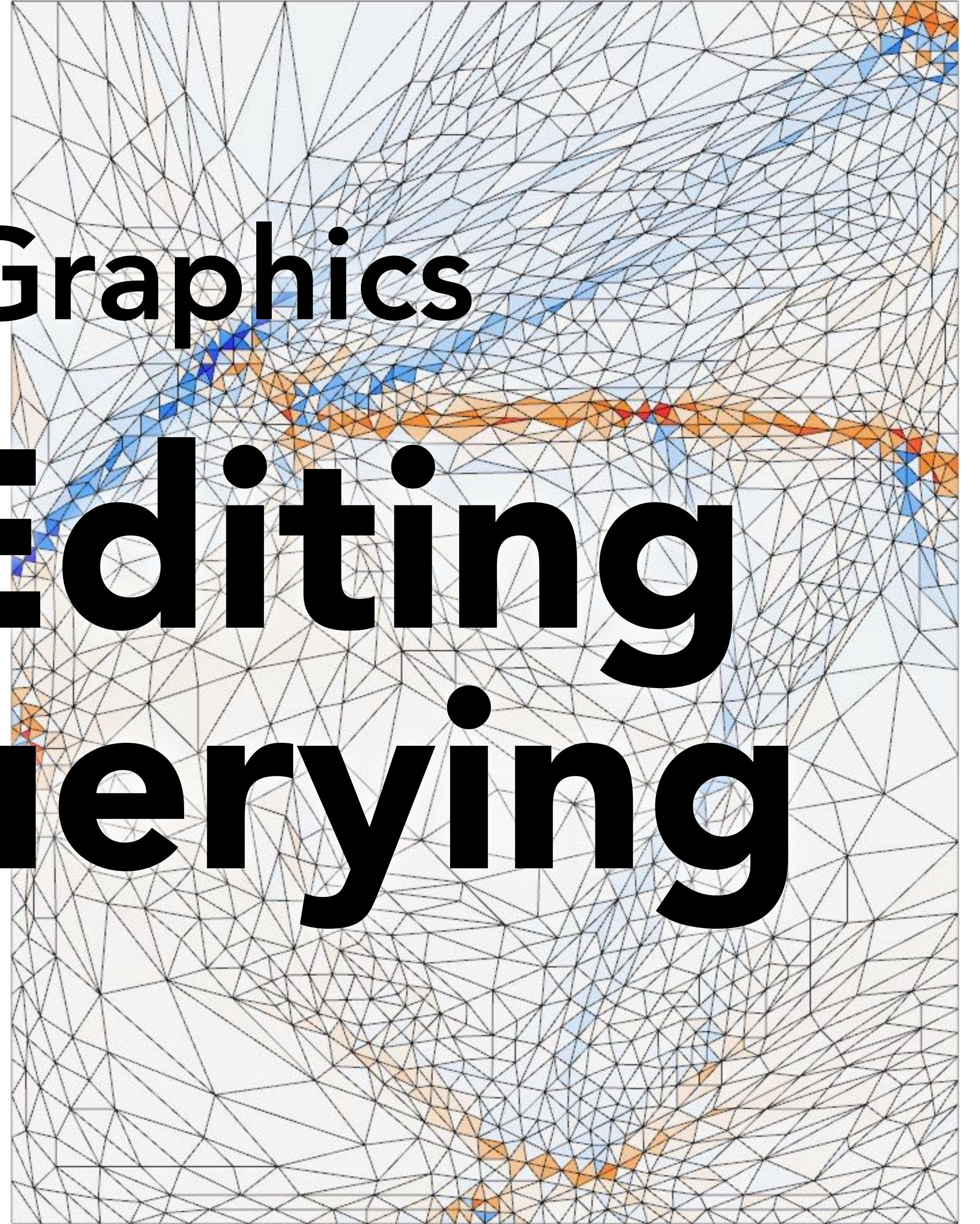


COL781: Computer Graphics

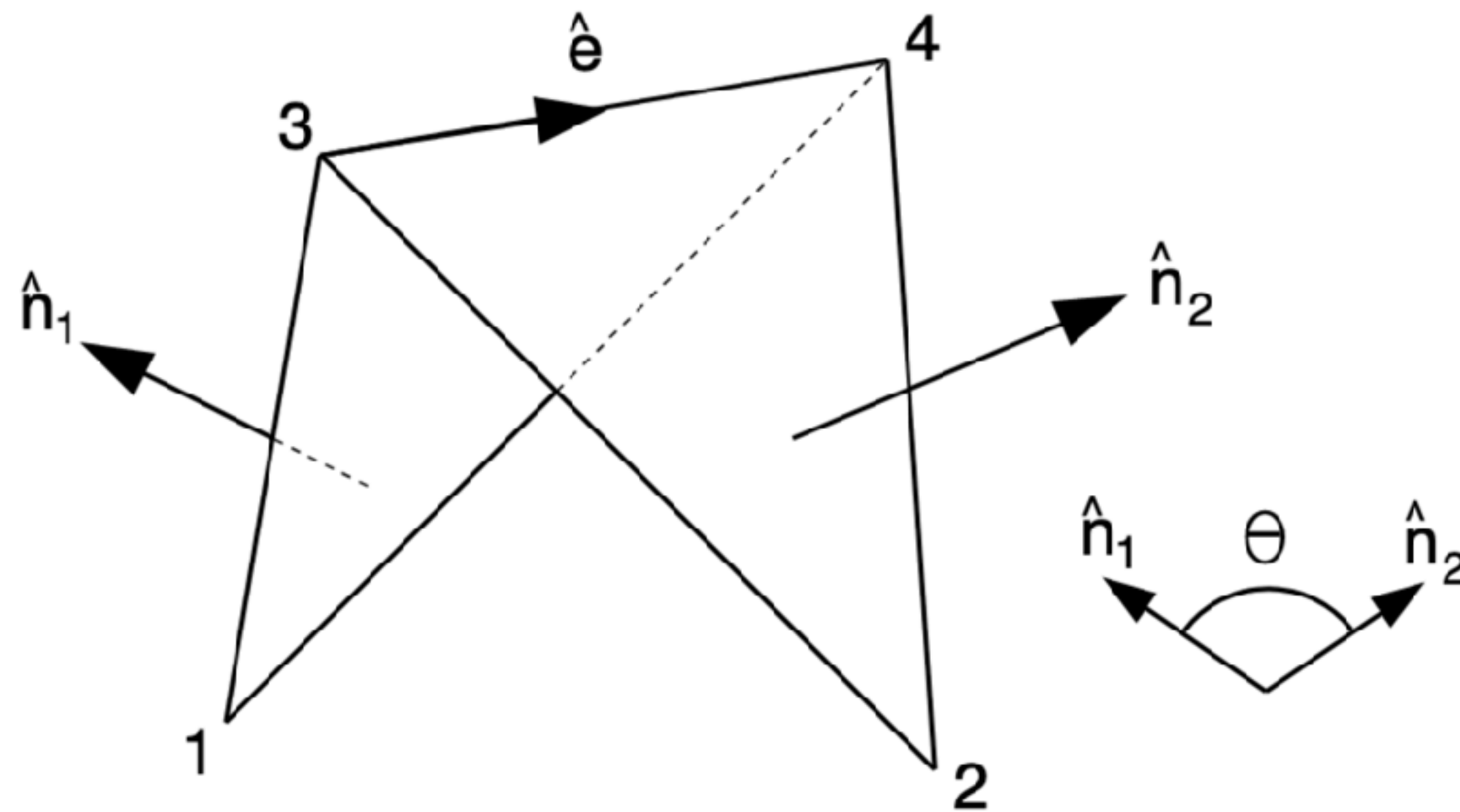
**18. Mesh Editing
and Querying**

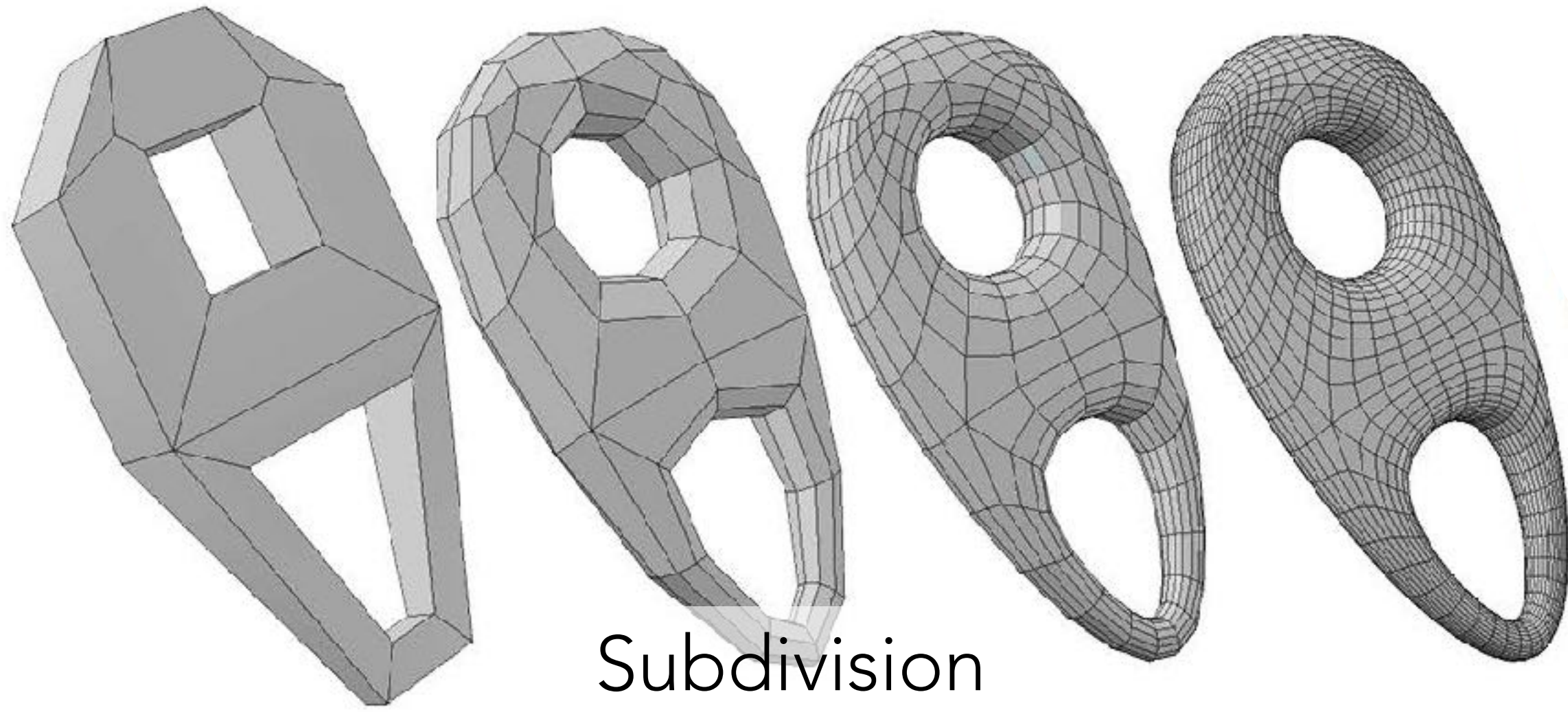


Practice problem

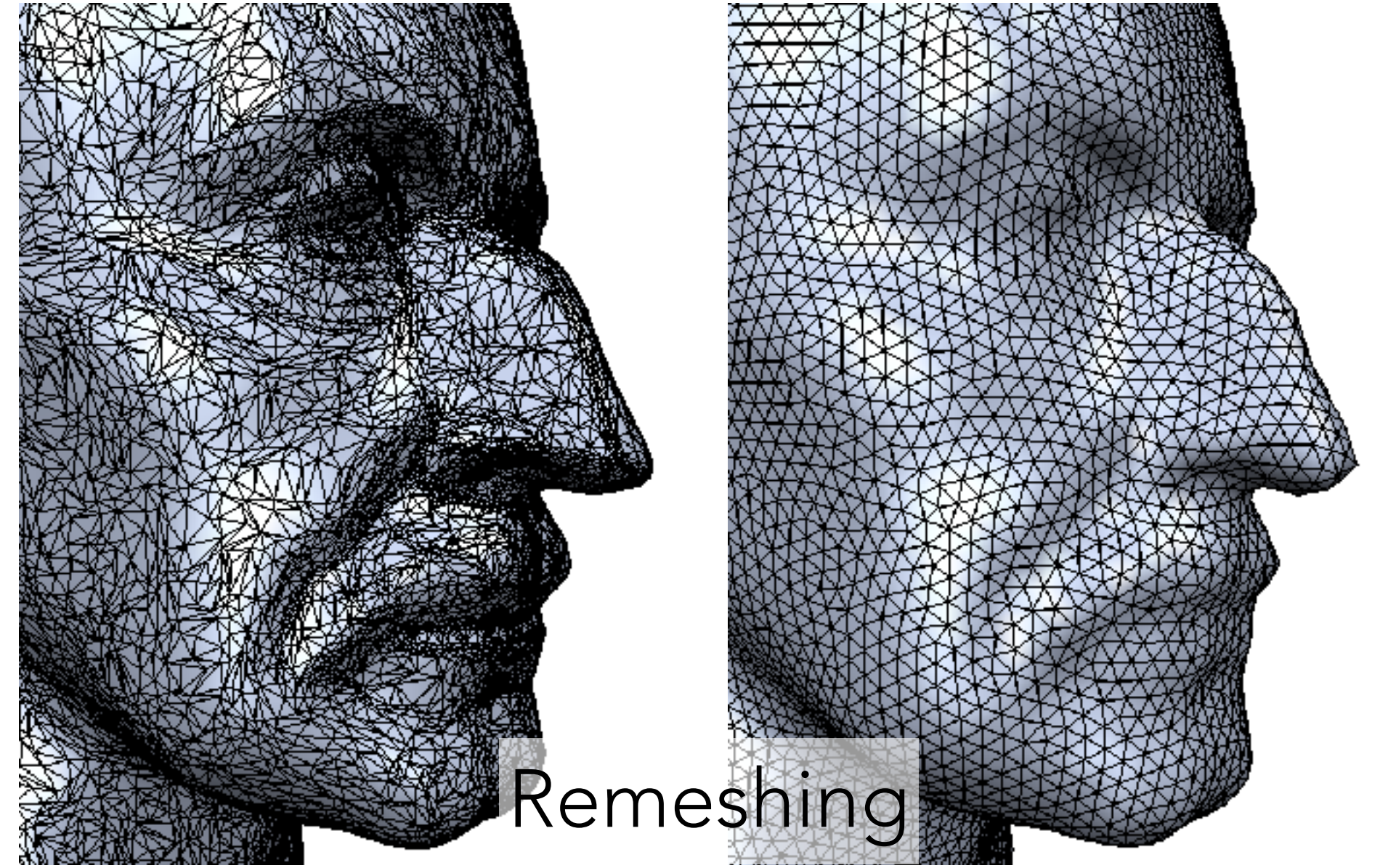
Using a half-edge representation of a triangle mesh, write (pseudo)code to find the "bending angle" at a given edge, i.e. the angle between the normals of the adjacent faces.

This is 180° minus the better-known dihedral angle.

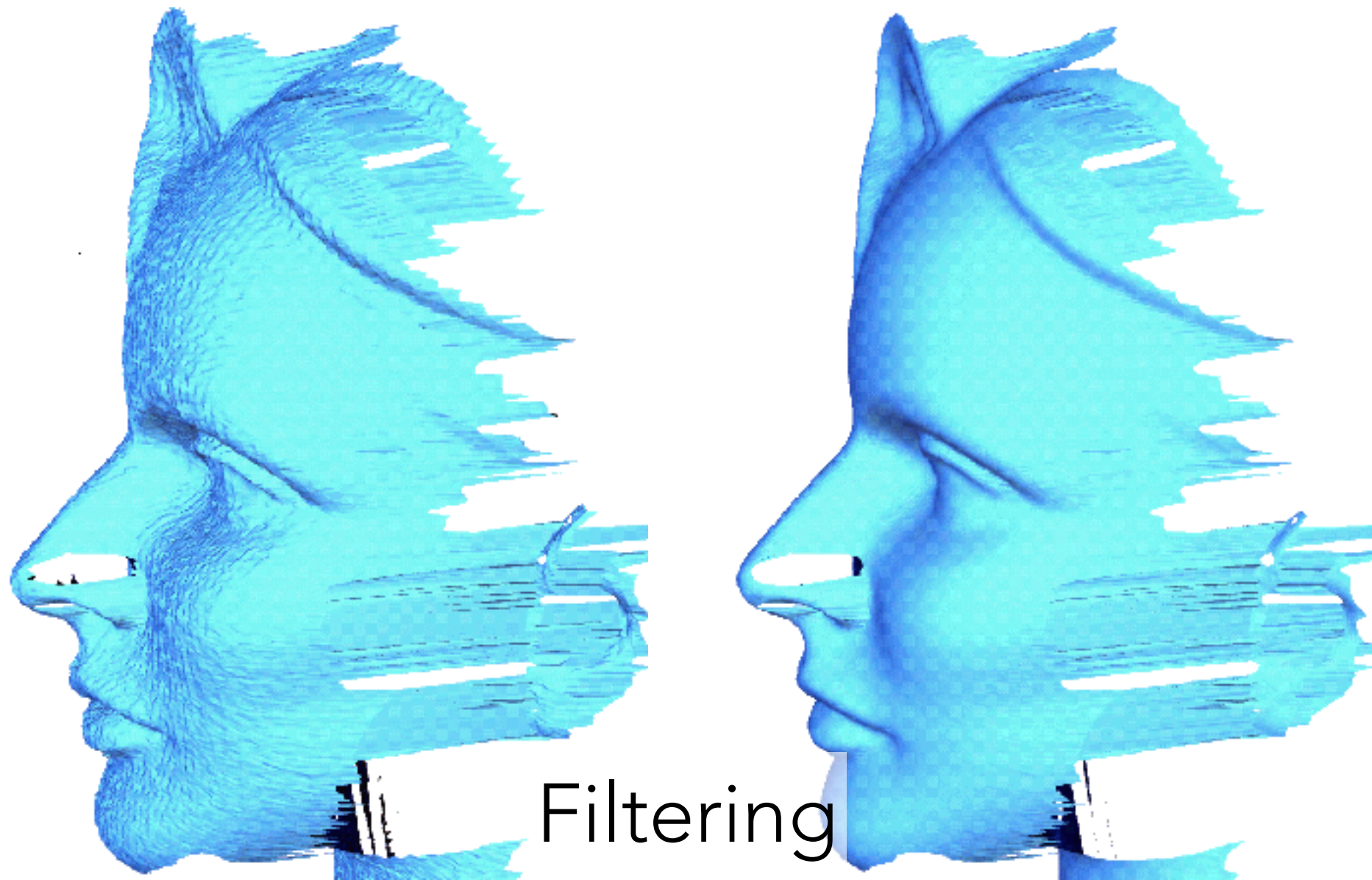




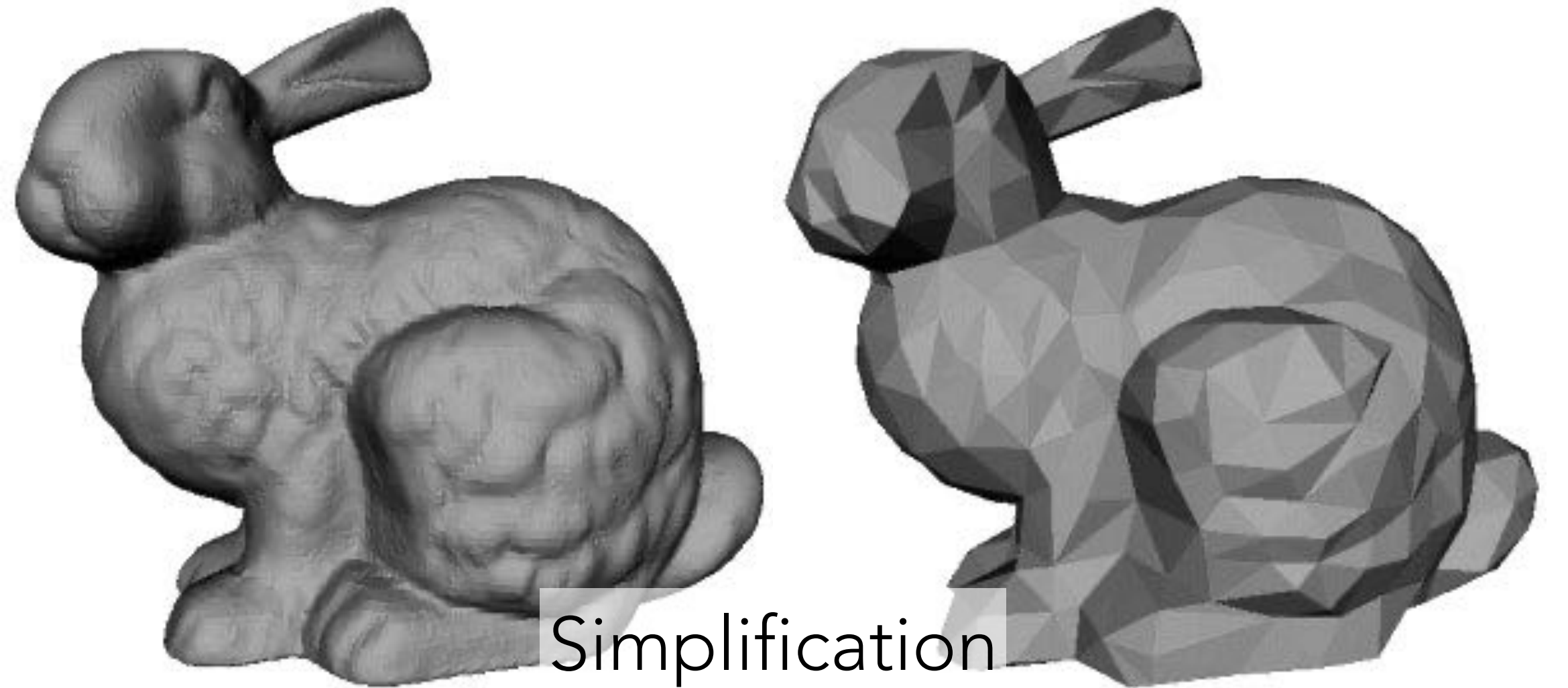
Subdivision



Remeshing



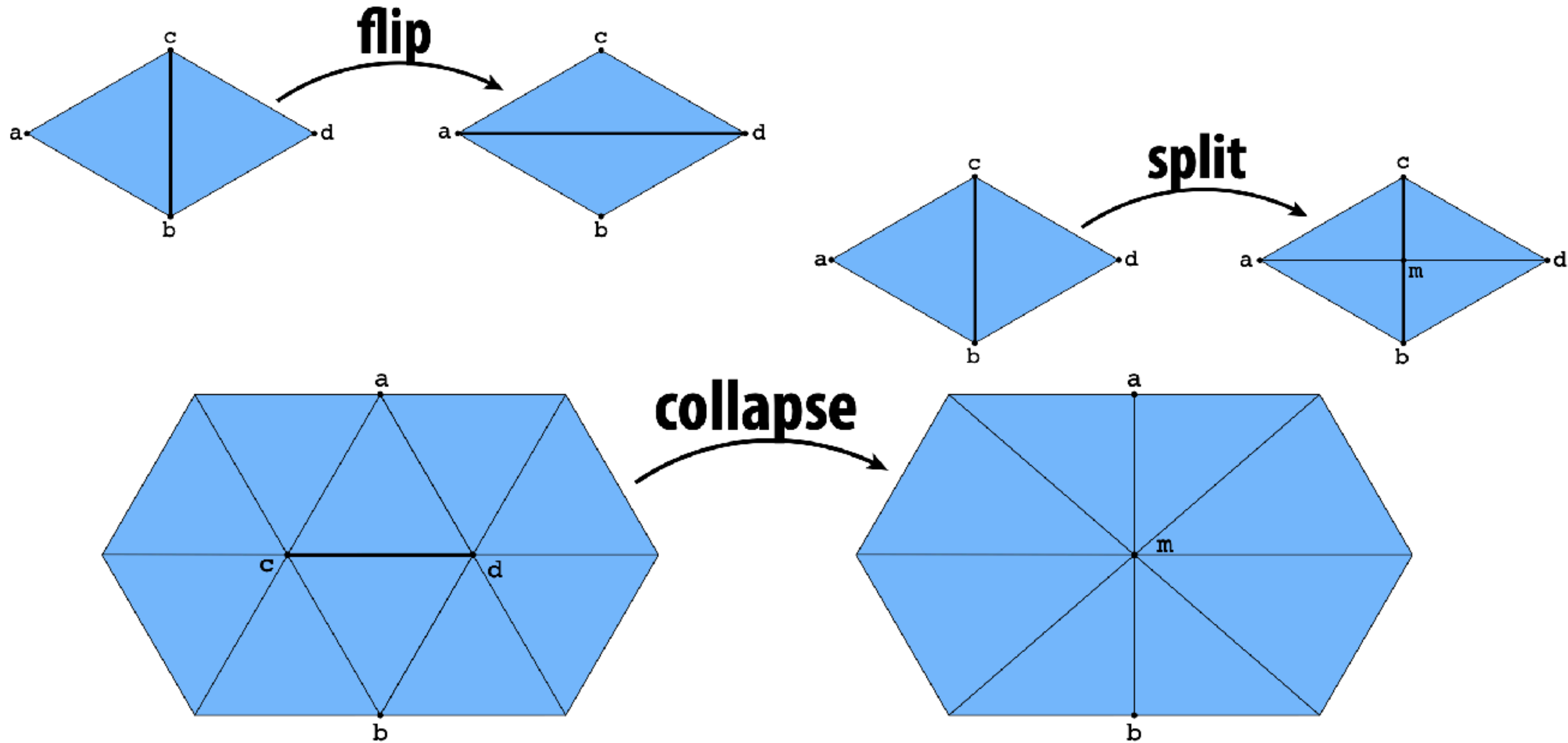
Filtering



Simplification

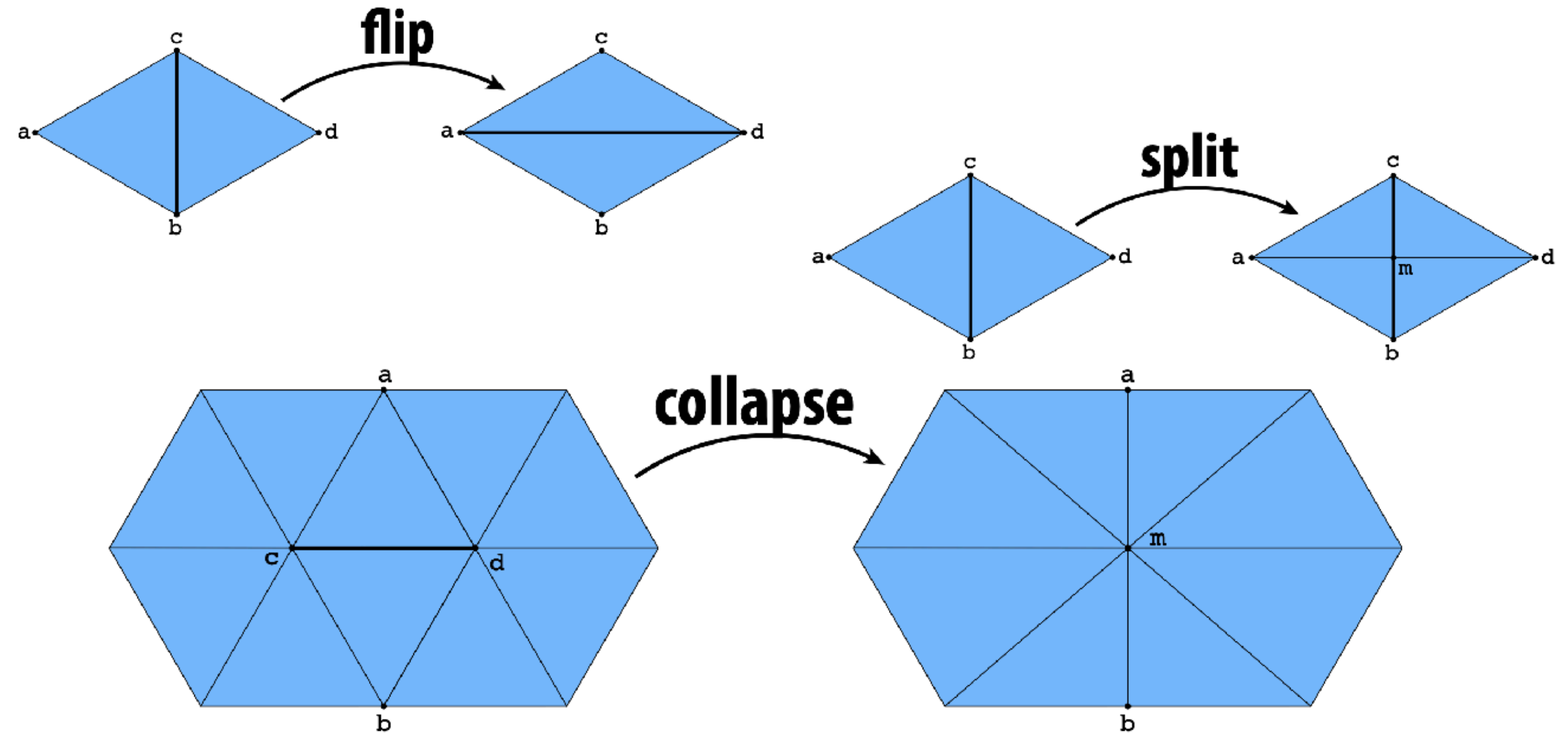
Mesh editing

Atomic mesh operations that preserve surface topology:



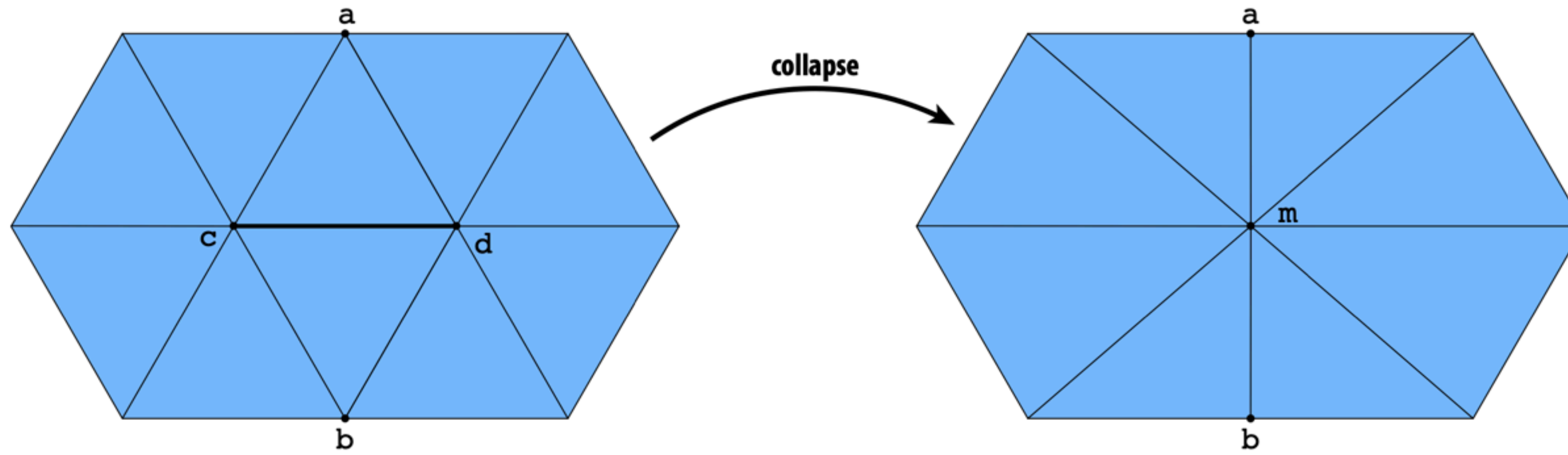
How to implement:

- Insert/delete elements
- Reassign pointers (carefully!) to maintain connectivity data



New vertices can be placed anywhere (but usually at the edge midpoint)

Other data, e.g. normals, may need to be recomputed



Here we have to be careful: The resulting mesh may not be manifold! (**How?**)

Solution: Collapse is only allowed if the edge's vertices have **exactly 2 adjacent vertices** in common.

Most high-level mesh editing operations are implemented in terms of these!

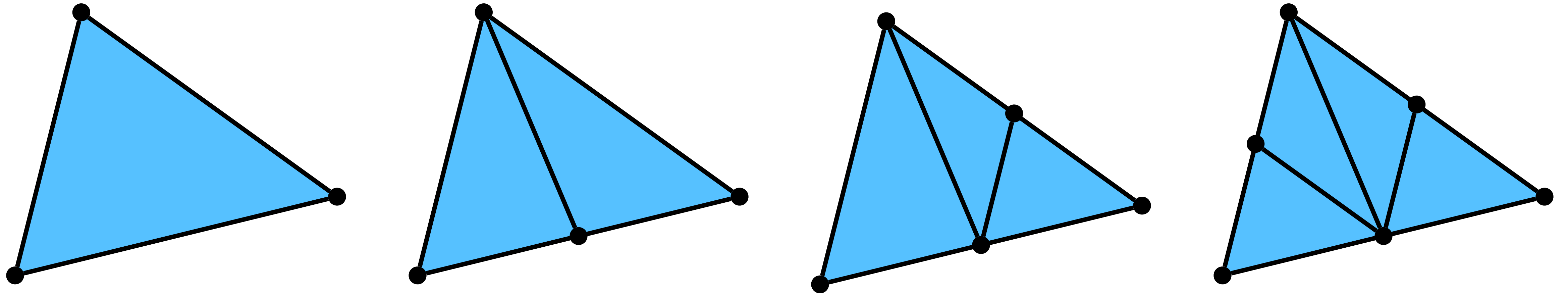
Oversimplified summary:

- **Subdivision:** split all edges
- **Simplification:** collapse unimportant edges
- **Regularization:** split long edges, collapse short edges

...So why do we need edge flips?

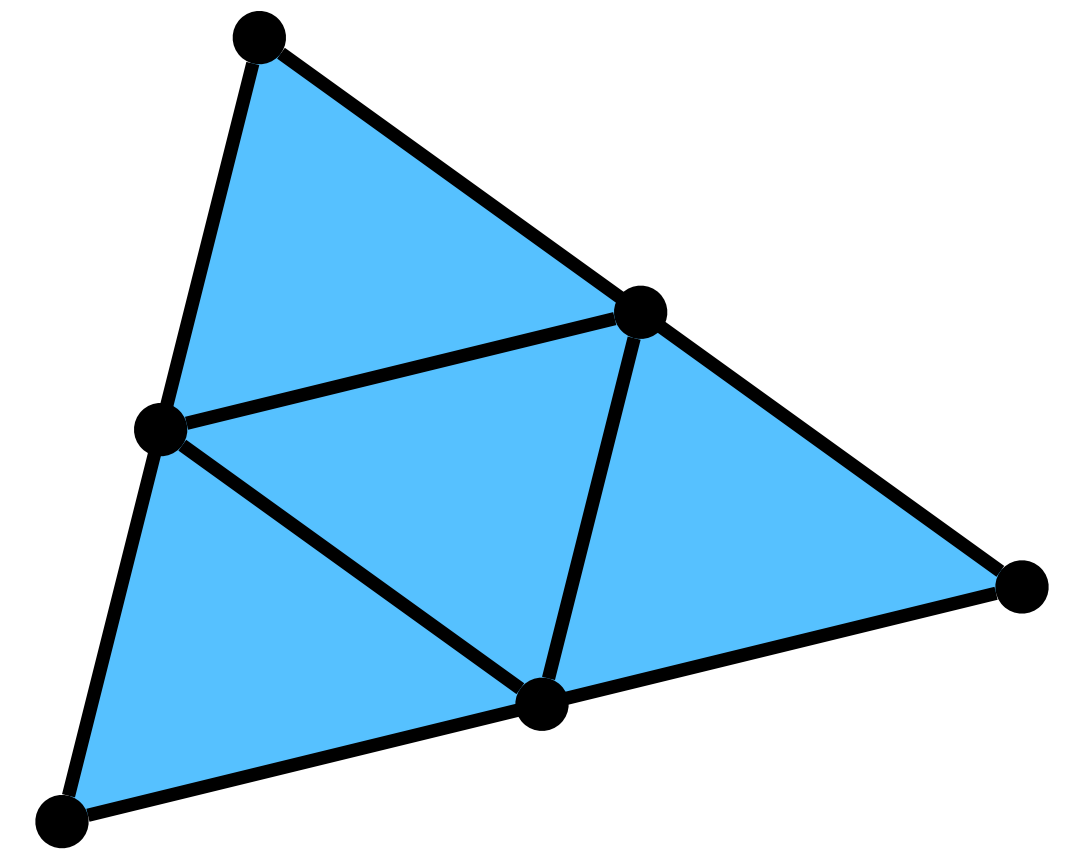


Let's subdivide this triangle by splitting all three edges...



Resulting mesh doesn't look so good until we flip one edge:

In general, edge flips are almost always necessary to obtain a "nice" triangulation.

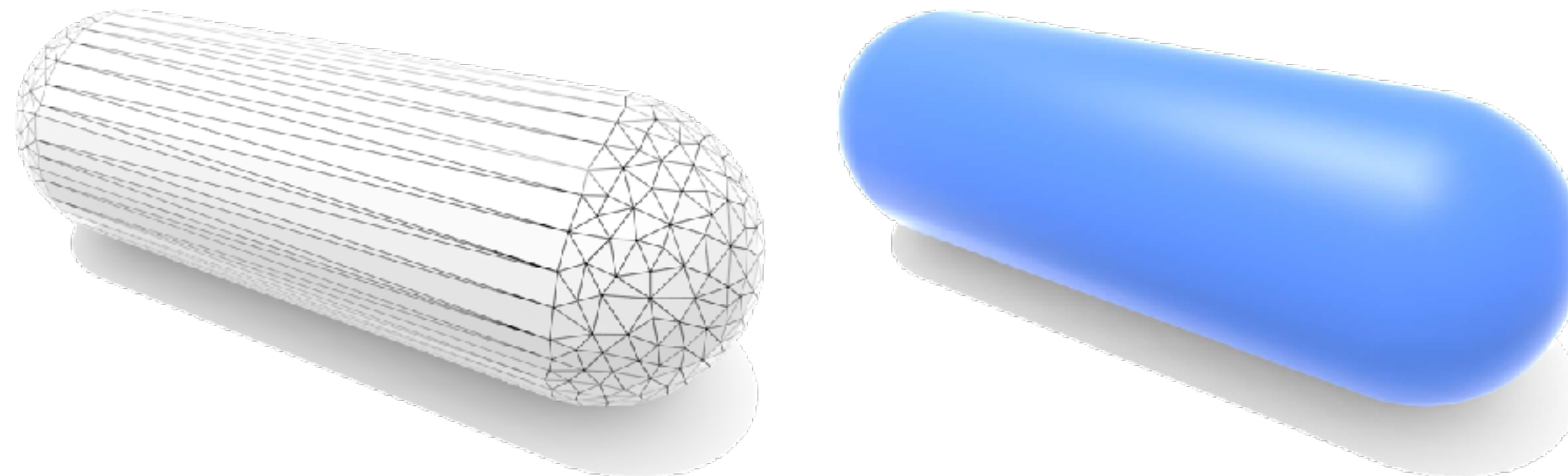
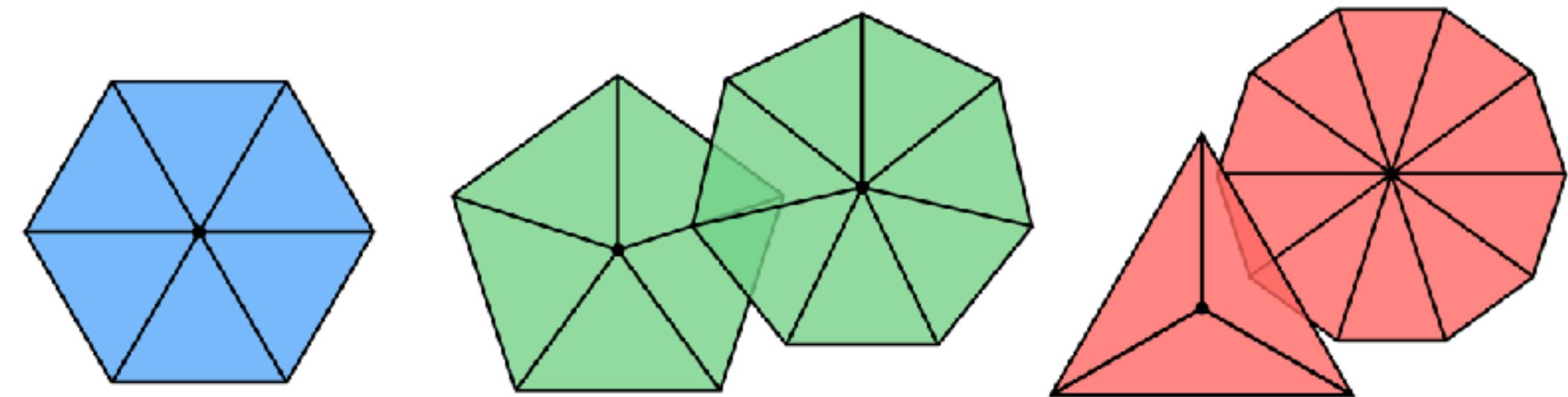
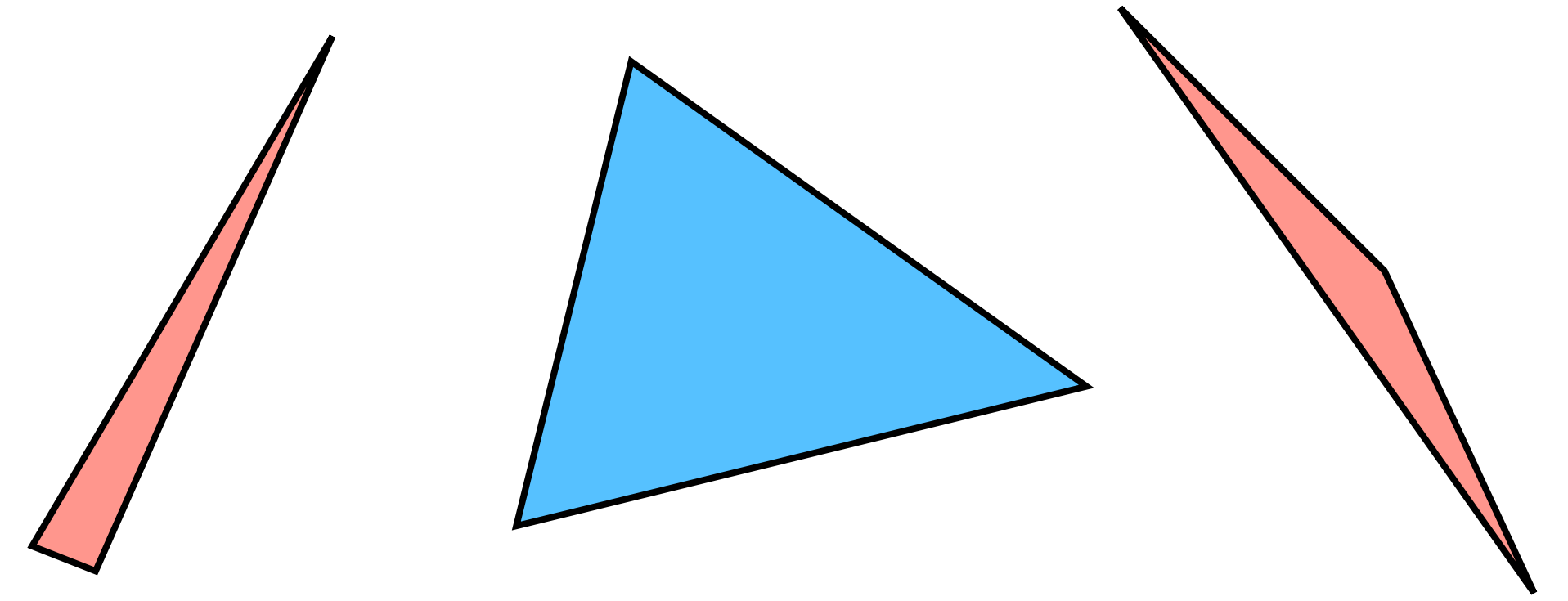


When is a mesh "nice"?

Various desirable properties:

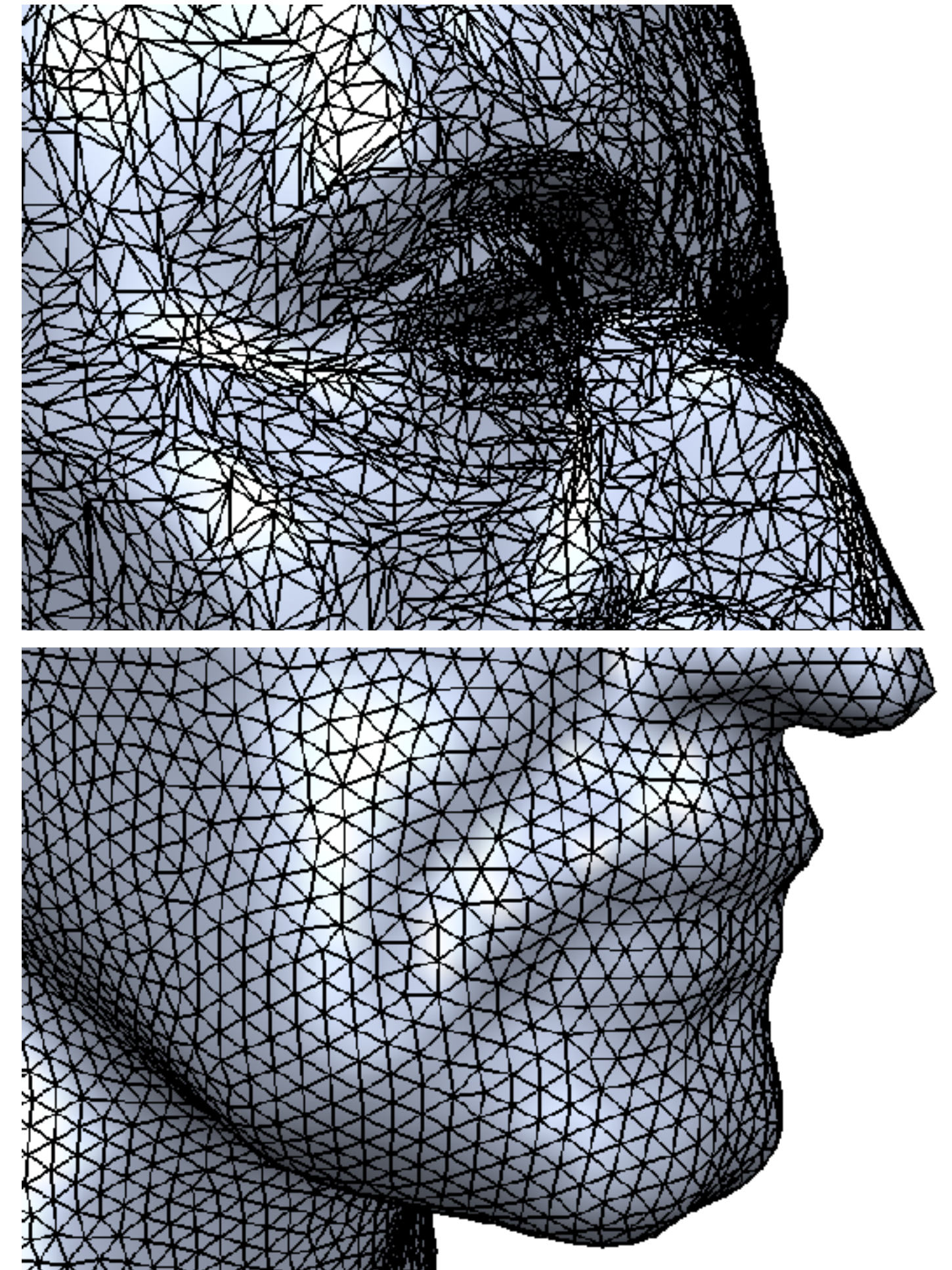
- Well-shaped polygons: triangles close to equilateral, not long and skinny
- Vertex degrees close to regular: 6 for triangle meshes, 4 for quad meshes
- Surface close to original (smooth) shape

Often this is a tradeoff!



Example: Isotropic remeshing [Botsch & Kobbelt 2004]

1. Split all edges longer than $\frac{4}{3} \times$ desired length
2. Collapse all edges shorter than $\frac{4}{5} \times$ desired length (unless this will create a long edge)
3. Flip edges to bring vertex degrees closer to 6
4. Move vertices tangentially towards average of neighbours
5. Repeat steps 1-4 several times



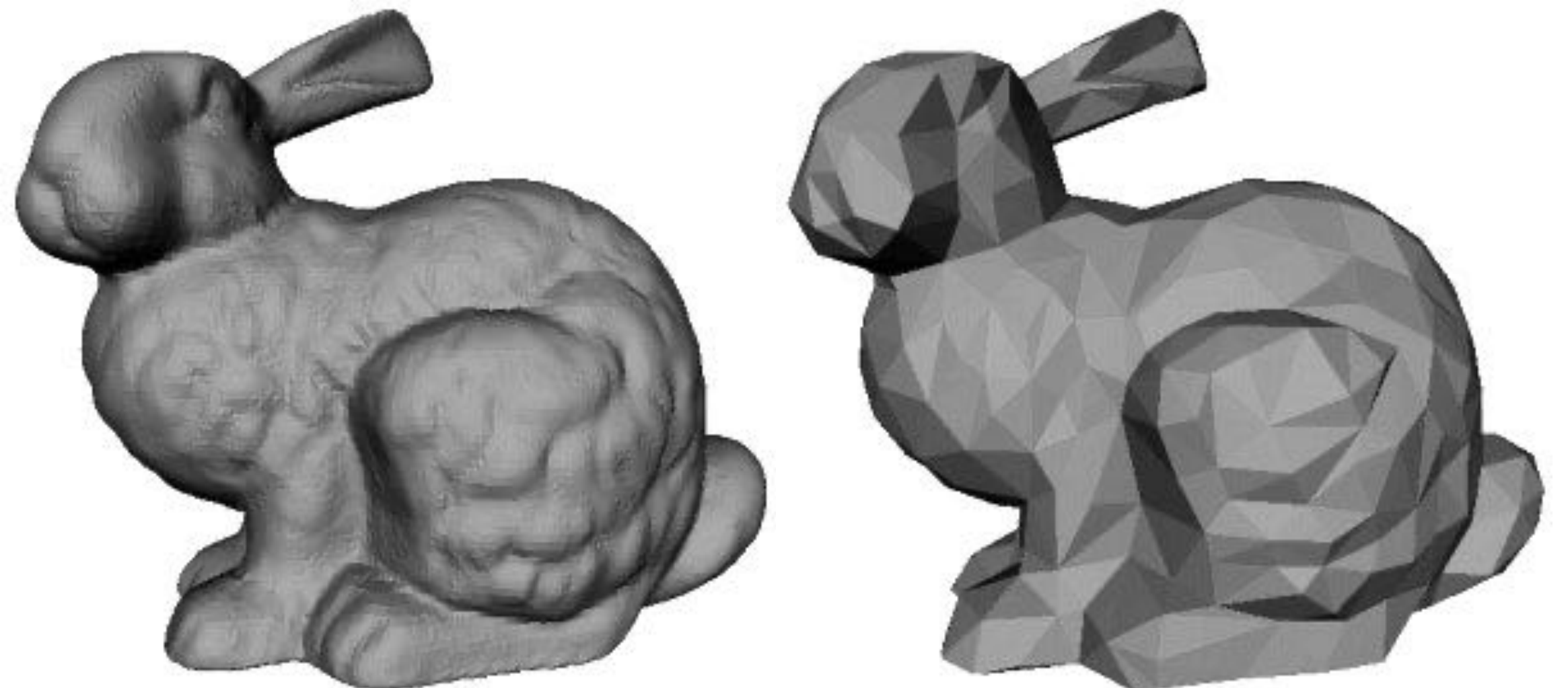
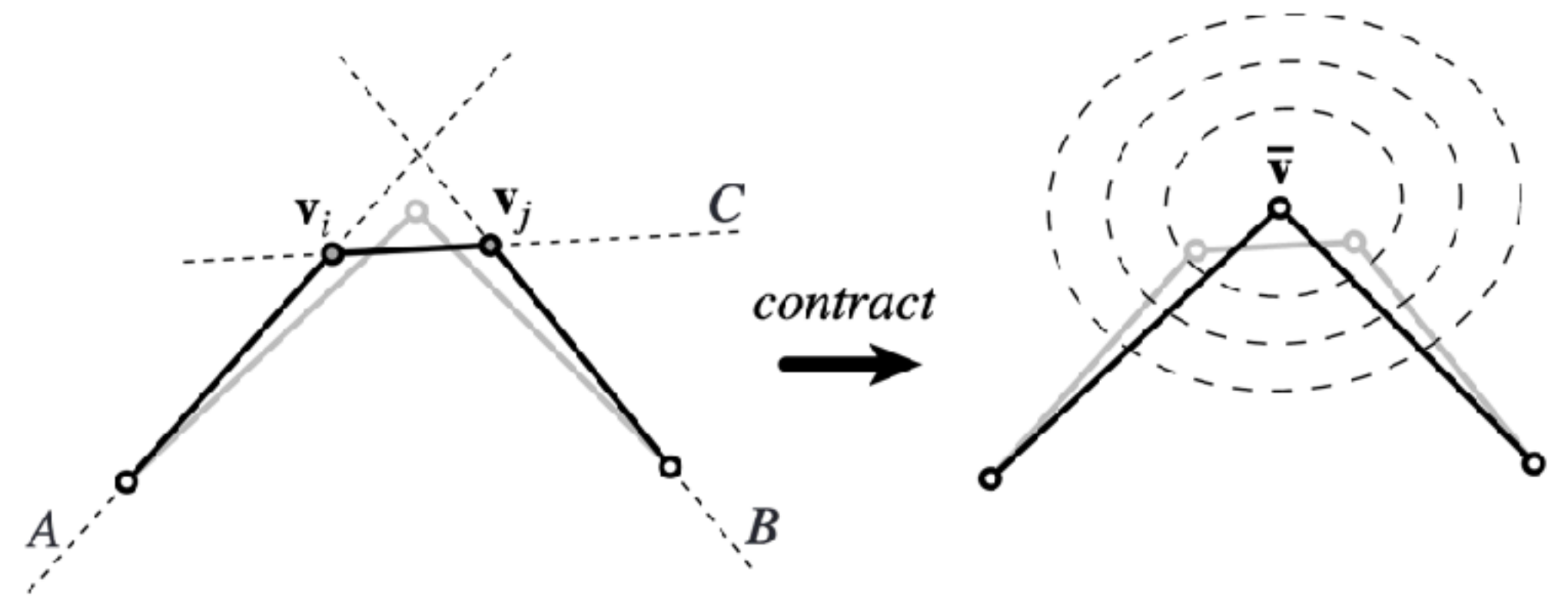
Example: Mesh simplification [Garland & Heckbert 1997]

Define cost of collapsing an edge \approx deviation of new vertex from original surface

1. Collapse edge with lowest cost
2. Repeat until desired number of vertices remain

Simple, greedy, **works really well!**

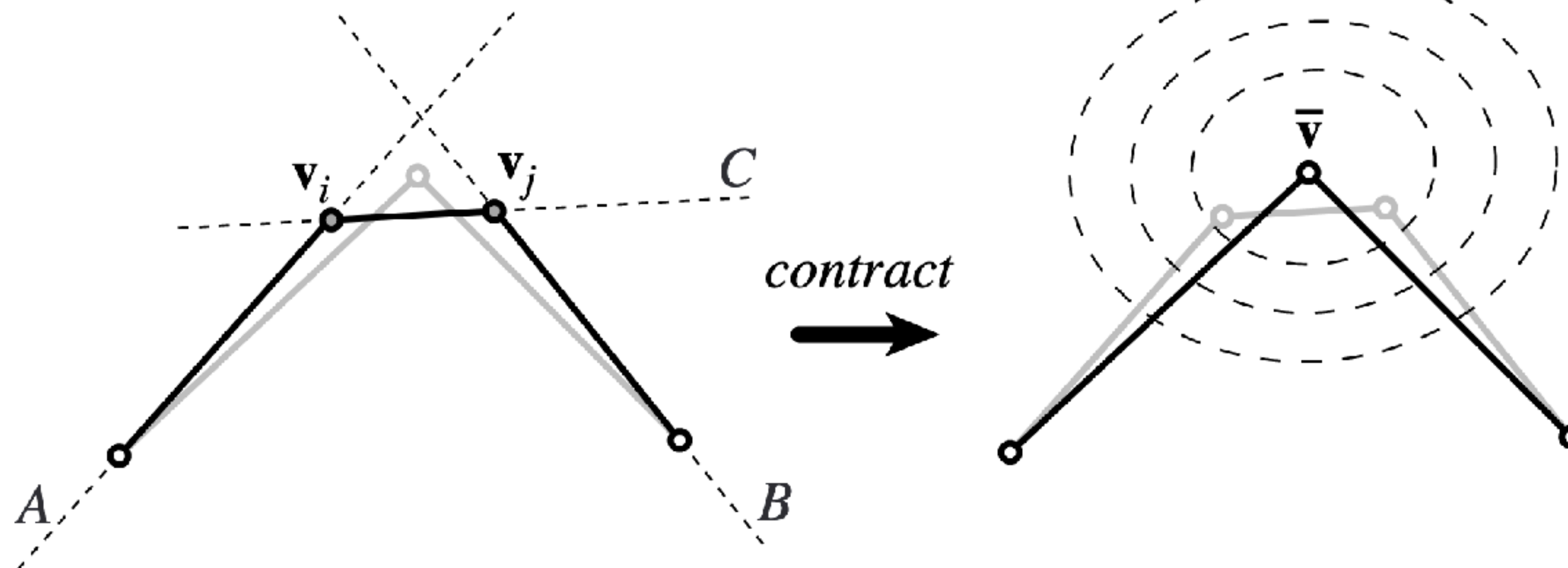
The trick: how to estimate deviation from surface efficiently?



For each face \mathbf{f} , error of a point $\mathbf{p} = \text{dist}(\mathbf{p}, \text{plane}(\mathbf{f}))^2 = (\mathbf{n} \cdot (\mathbf{p} - \mathbf{p}_0))^2$

For each vertex, "quadratic error" $Q(\mathbf{p}) = \sum \text{dist}(\mathbf{p}, \text{plane}(\mathbf{f}_i))^2$ is **quadratic** in \mathbf{p}

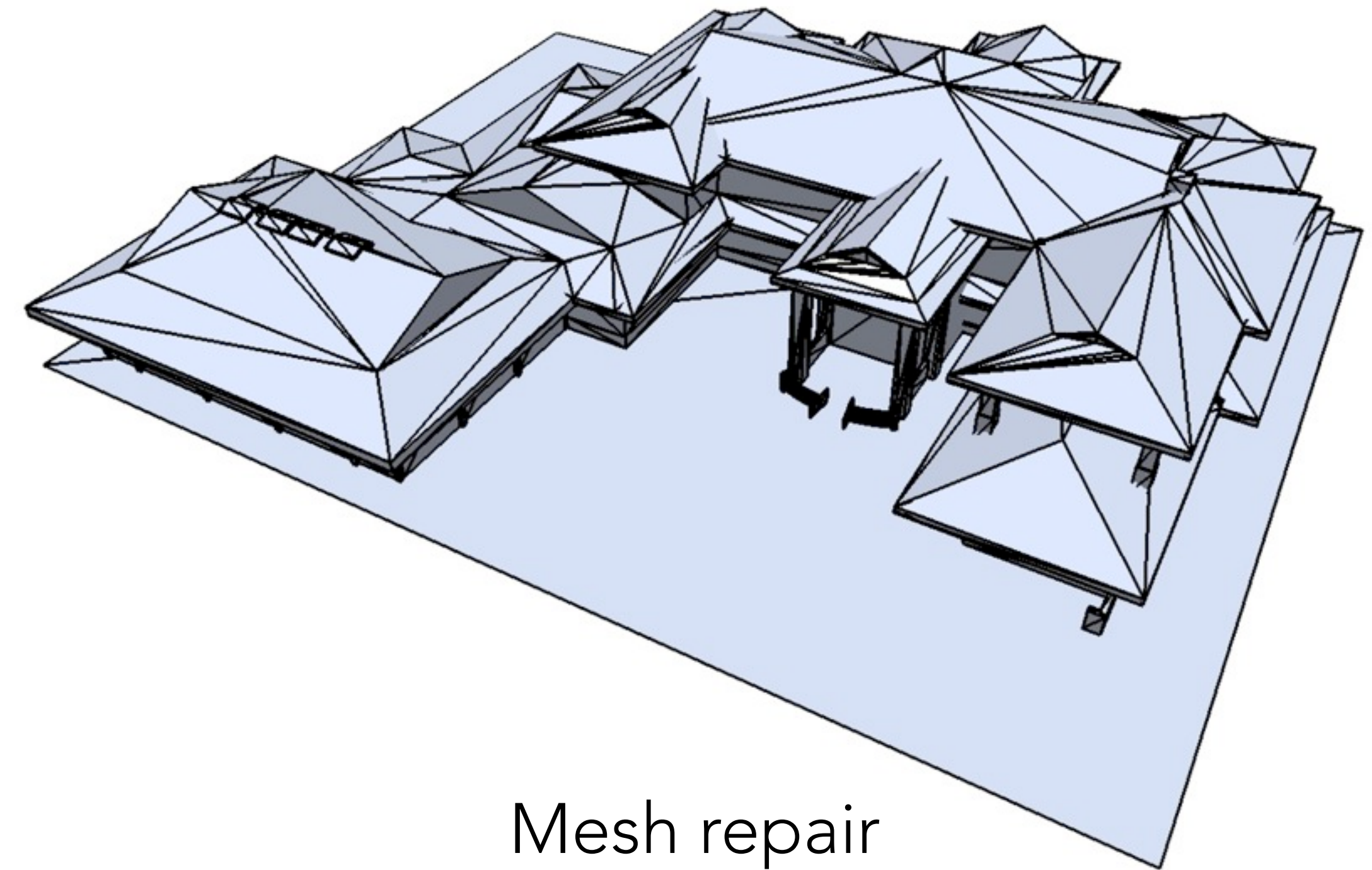
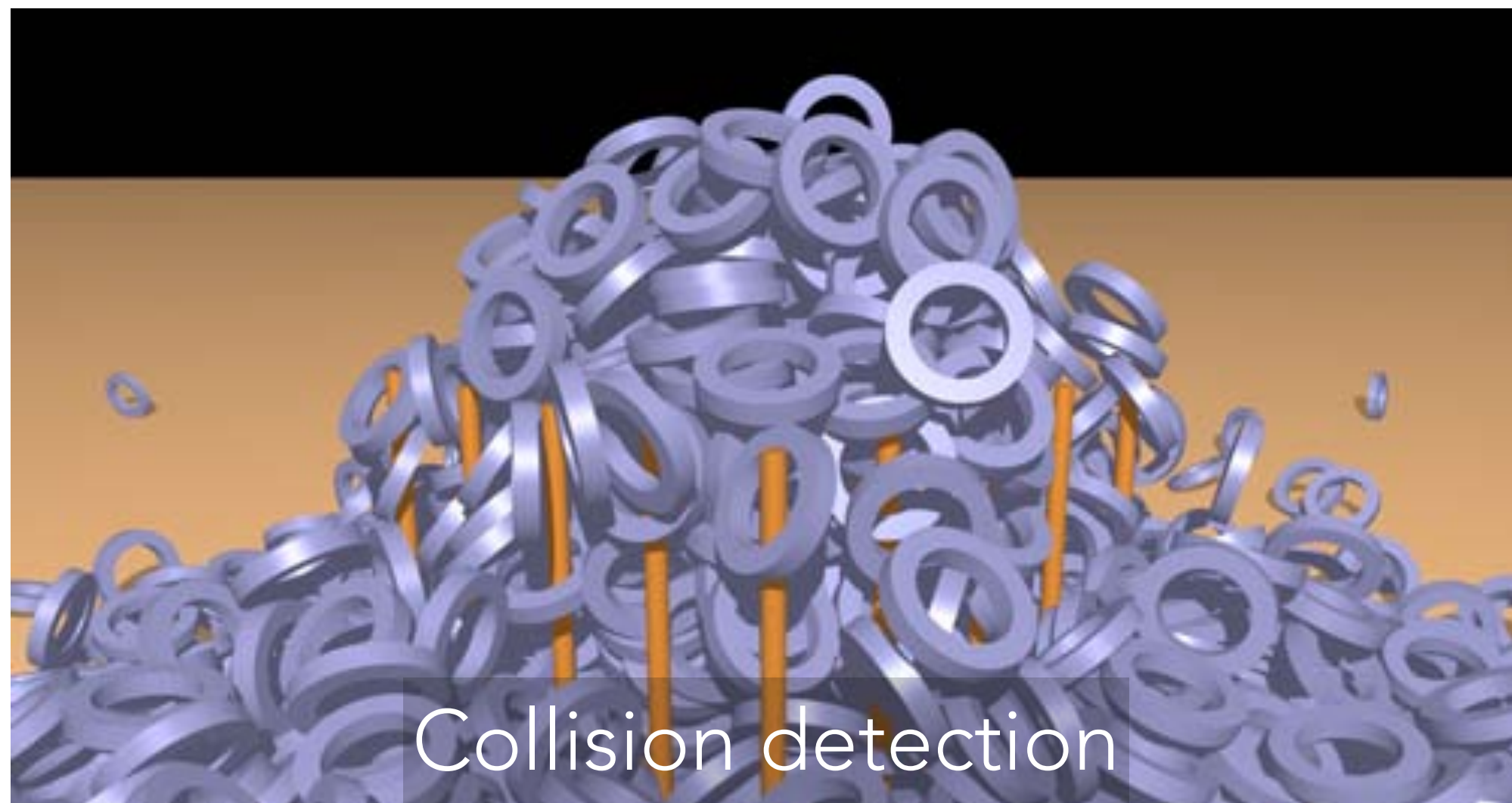
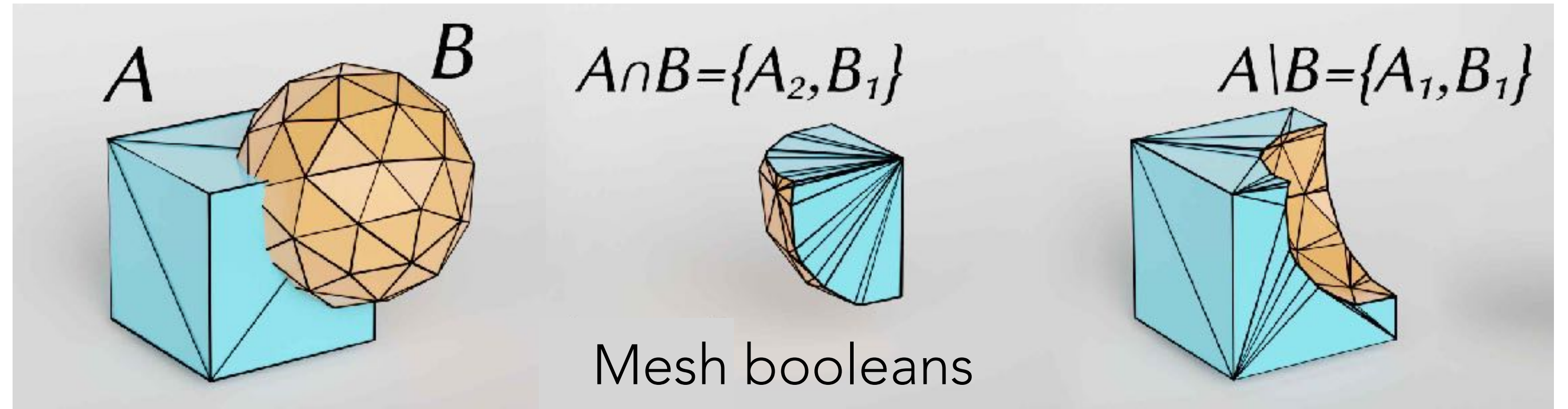
Cost of collapsing edge $(\mathbf{v}_i, \mathbf{v}_j)$ to new vertex $\bar{\mathbf{v}} = Q_i(\bar{\mathbf{v}}) + Q_j(\bar{\mathbf{v}})$
where $\bar{\mathbf{v}}$ is chosen to give the minimum cost.



Geometric queries

Once we have modeled a shape, we will still want to answer questions like:

- Given a point \mathbf{p} , what is the **closest point** on the shape?
- Is the point \mathbf{p} **inside or outside** the shape?
- Given a ray $\mathbf{o} + t\mathbf{d}$, where does the **ray intersect** the shape?
- Given two shapes S_1 and S_2 , what are the pair of points in closest **proximity**?
Do the two **shapes intersect** each other?



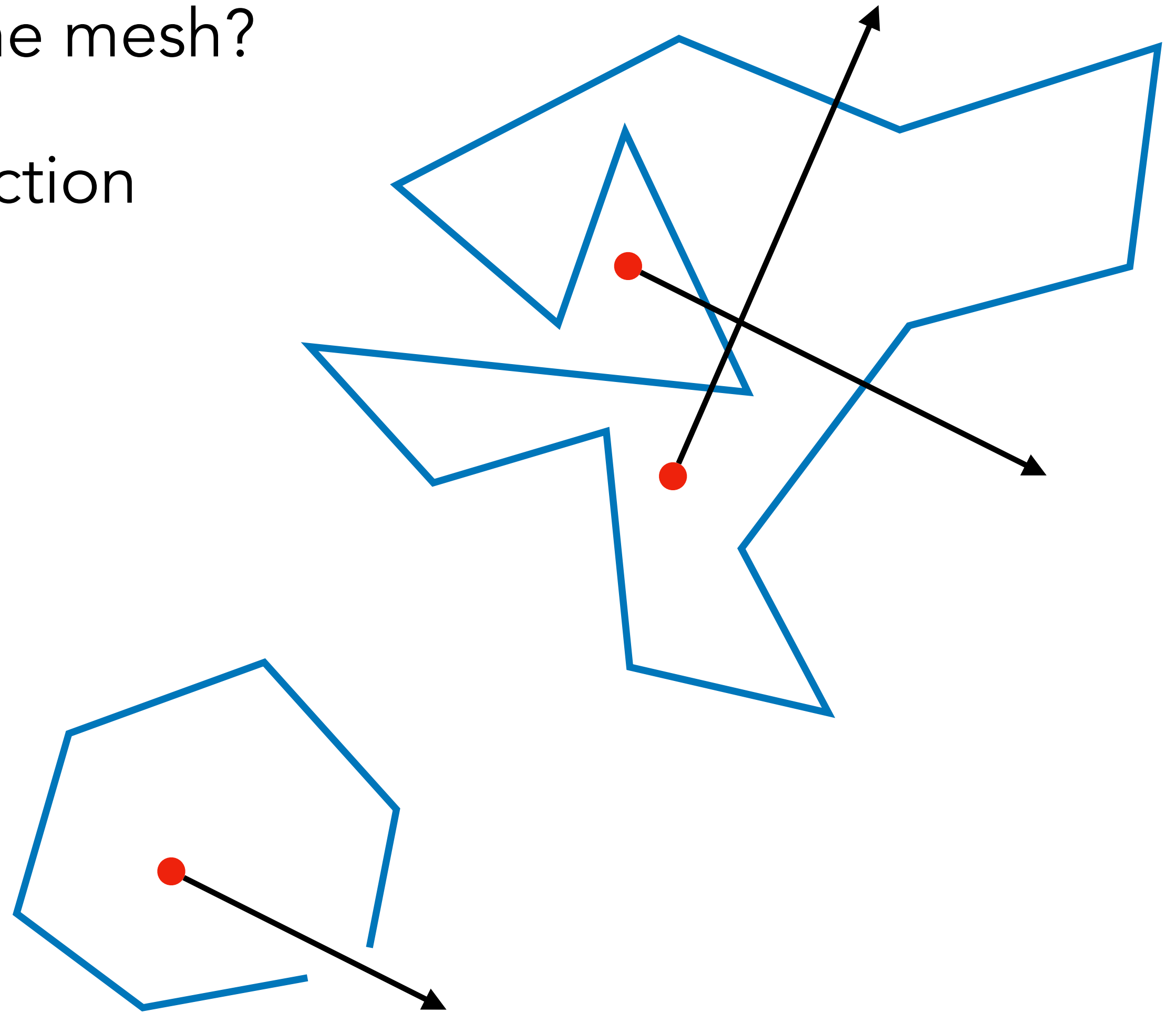
Inside/outside test

Given a point \mathbf{p} , is it inside or outside the mesh?

Trace a ray to infinity in an arbitrary direction and count the number of intersections.

Odd = inside, even = outside.

Of course, the mesh has to be closed for this to make sense...



Closest point on a line

The line between \mathbf{a} and \mathbf{b} is the set of points $\{\mathbf{a} + t(\mathbf{b} - \mathbf{a}) : t \in \mathbb{R}\}$.

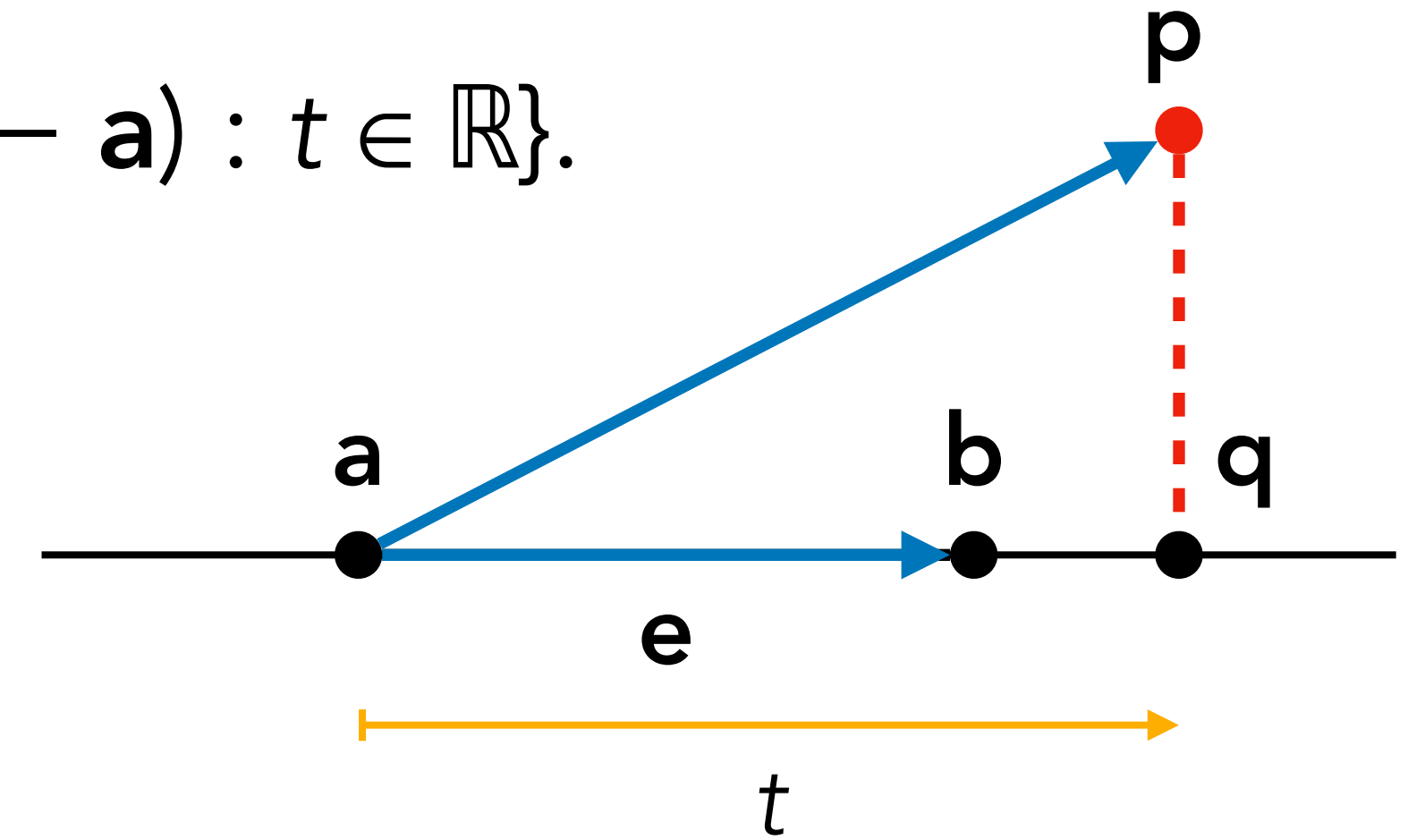
Let $\mathbf{e} = \mathbf{b} - \mathbf{a}$, $\hat{\mathbf{e}} = \mathbf{e}/\|\mathbf{e}\|$.

Closest point to \mathbf{p} on the line:

$$\mathbf{q} = \mathbf{a} + (\hat{\mathbf{e}} \cdot (\mathbf{p} - \mathbf{a}))\hat{\mathbf{e}}$$

Equivalently... $\mathbf{q} = \mathbf{a} + t\mathbf{e}$ where

$$t = \frac{\mathbf{e} \cdot (\mathbf{p} - \mathbf{a})}{\|\mathbf{e}\|^2}$$

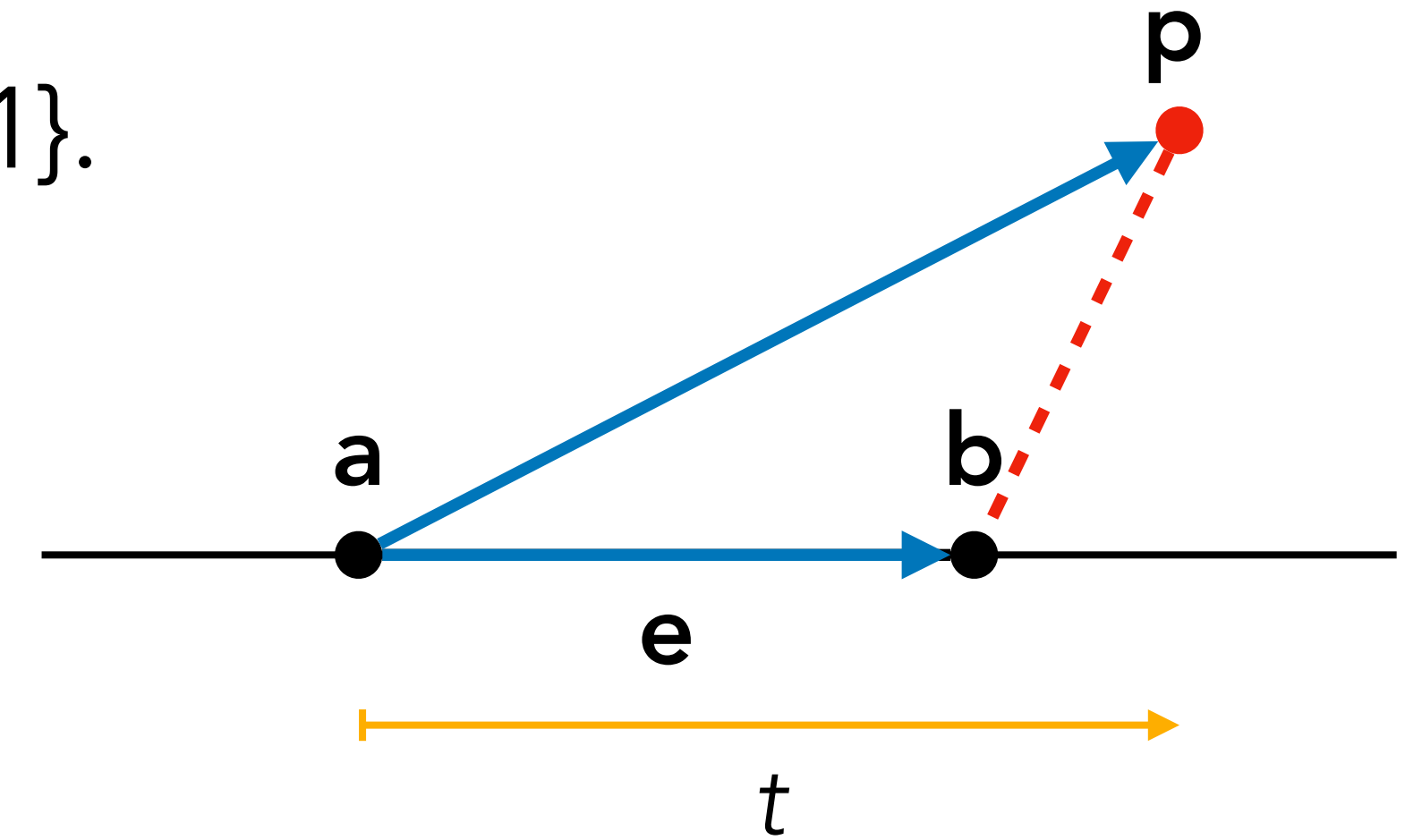


Closest point on a line segment

The line segment is the set of points $\{\mathbf{a} + t\mathbf{e} : 0 \leq t \leq 1\}$.

Closest point to \mathbf{p} on line segment:

- Project \mathbf{p} to the line by computing t as before.
- Project \mathbf{q} again to the line segment:
 - If $t < 0$: set $t = 0$
 - If $t > 1$: set $t = 1$

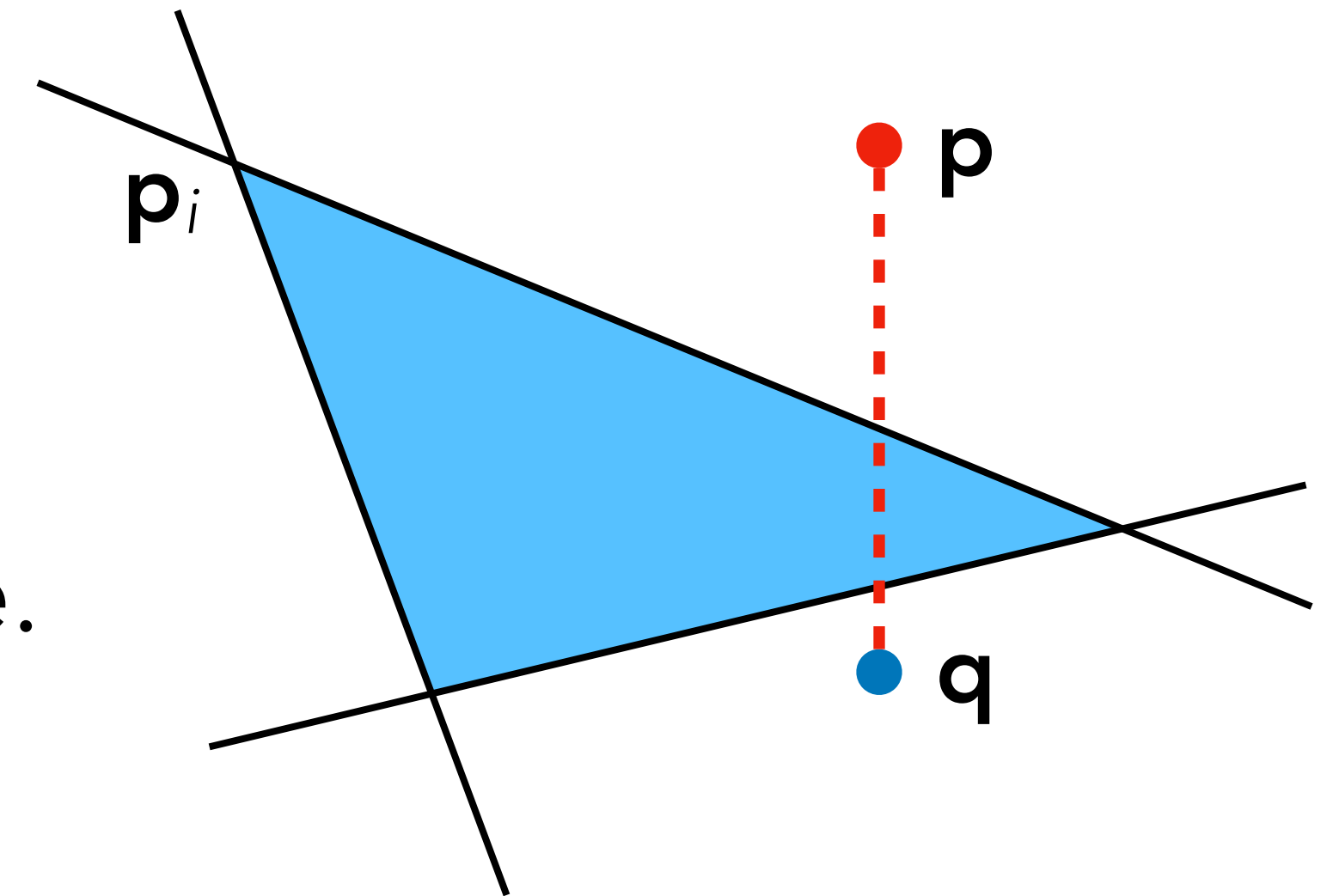


Closest point on a triangle

A triangle is the set of points $\{\mathbf{p}_0 + b_1\mathbf{e}_1 + b_2\mathbf{e}_2 : 0 \leq b_1, b_2, 1 - b_1 - b_2 \leq 1\}$ where $\mathbf{e}_1 = \mathbf{p}_1 - \mathbf{p}_0$, $\mathbf{e}_2 = \mathbf{p}_2 - \mathbf{p}_0$.

Closest point to \mathbf{p} on triangle:

- Solve $(\mathbf{p}_0 + b_1\mathbf{e}_1 + b_2\mathbf{e}_2) + h\mathbf{n} = \mathbf{p}$ for b_1, b_2, h .
 $\mathbf{q} = \mathbf{p}_0 + b_1\mathbf{e}_1 + b_2\mathbf{e}_2$ is the closest point on the plane.
Let $b_3 = 1 - b_1 - b_2$.
- If $0 \leq b_1, b_2, b_3 \leq 1$: closest point is \mathbf{q} .
- If $b_i < 0$: find closest point on opposite edge.

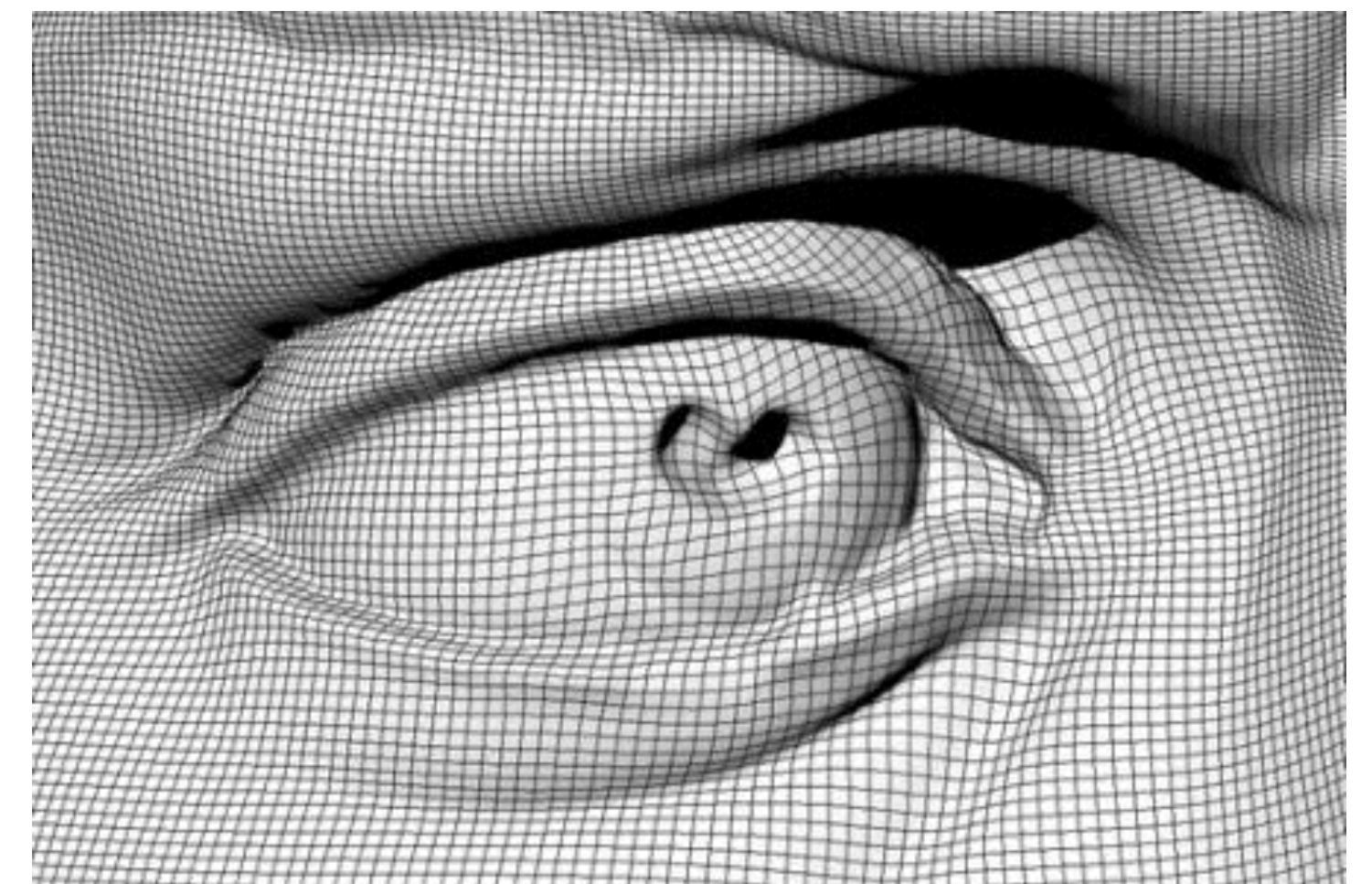


Closest point on a triangle mesh

Naïve algorithm:

- For each triangle, find the closest point \mathbf{q}_i and its distance $\|\mathbf{p} - \mathbf{q}_i\|$
- Return the \mathbf{q}_i with the lowest distance

Not scalable to high-resolution meshes!



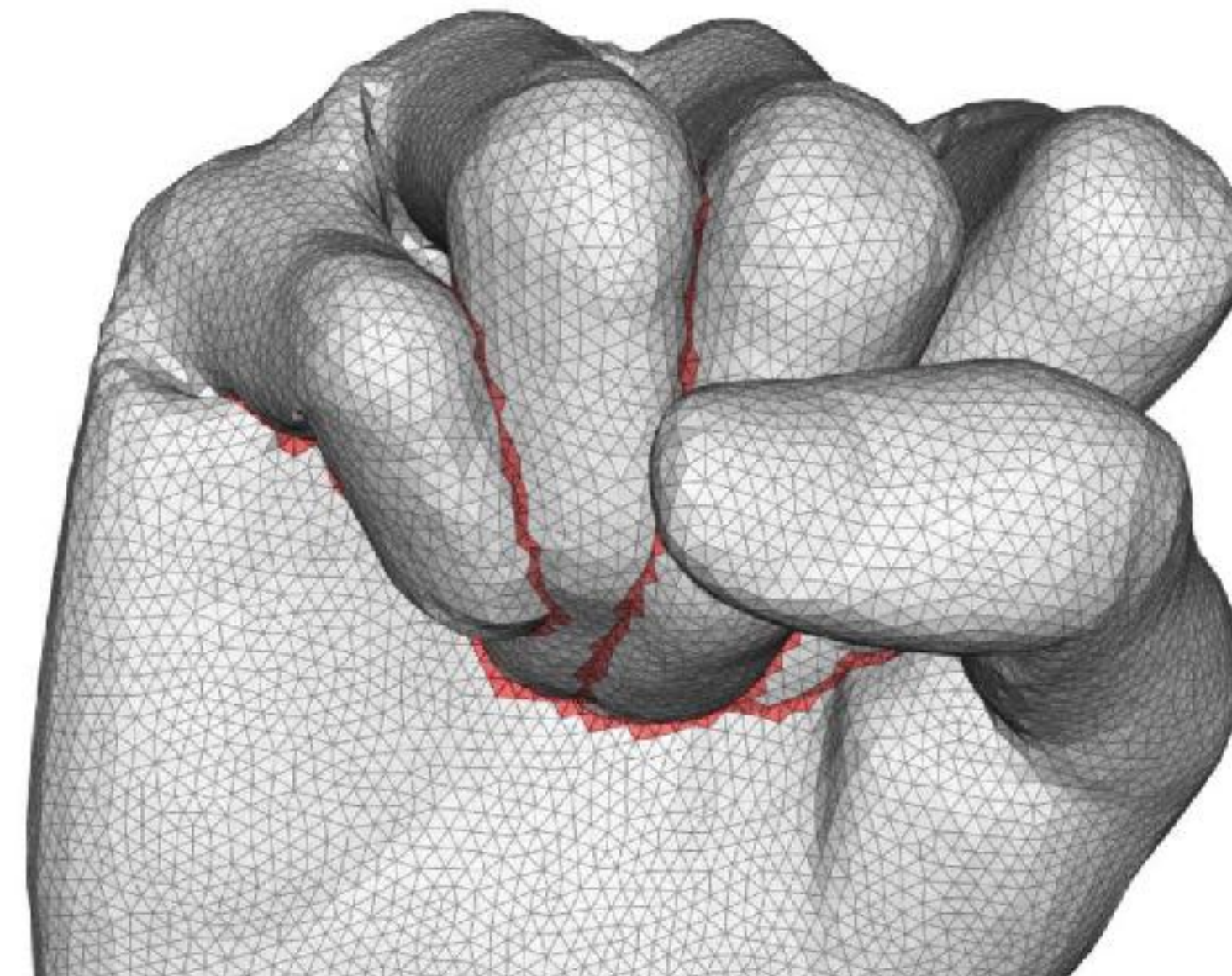
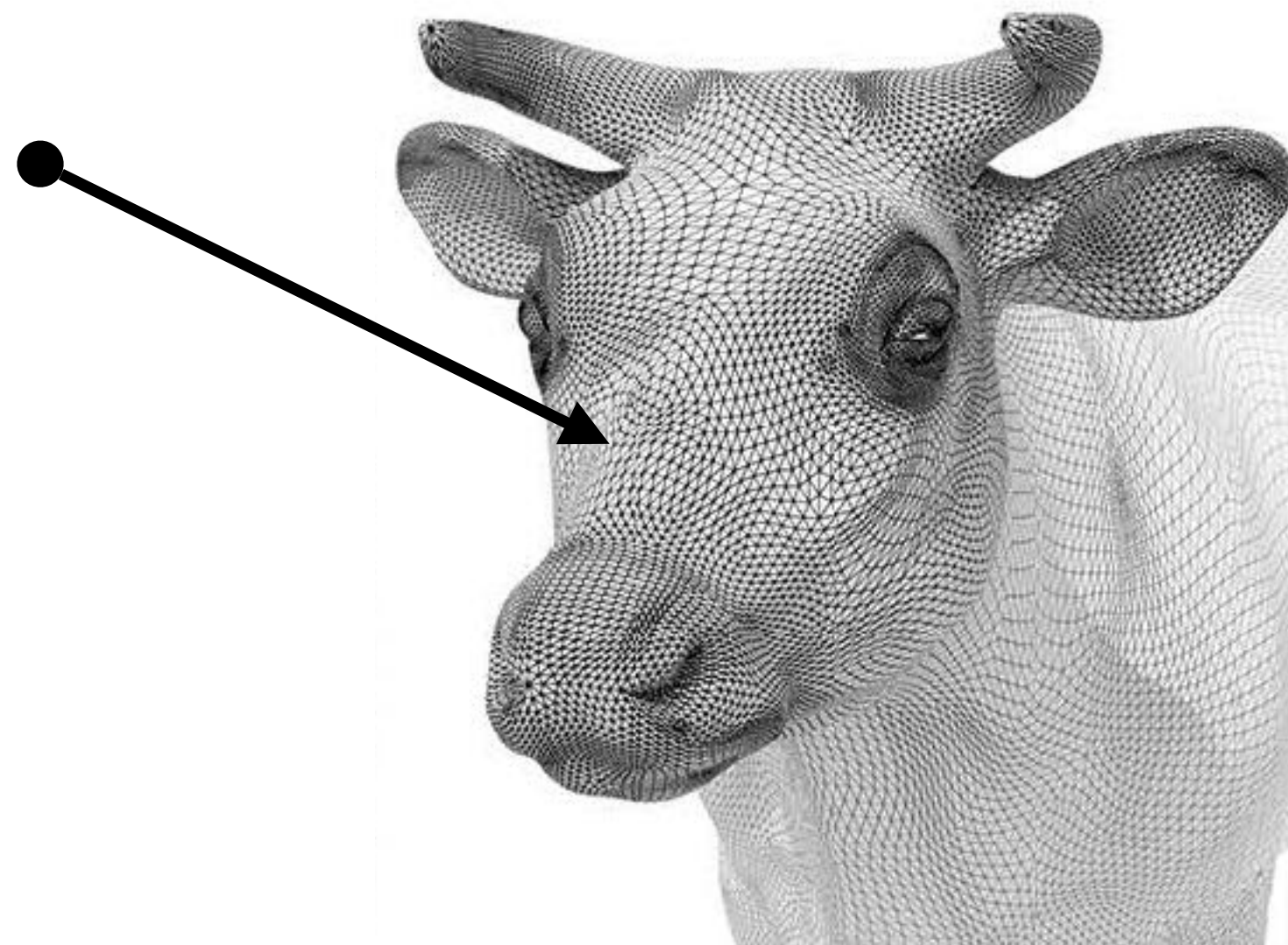
We have the same problem in ray tracing:

- Intersect **each triangle** with the ray and return the earliest?

and mesh-mesh intersection:

- Test **each pair of triangles** with each other and see if they intersect?

Next class: **spatial data structures** for accelerating such queries.



Triangle-triangle intersection

Early culling:

- If all vertices of one triangle are on same side of other triangle plane: no intersection

Otherwise:

- Intersection of triangle planes is a line $L = \{\mathbf{o} + t\mathbf{d}\}$
- Find t values where triangle edges cross L
- Check if t ranges overlap: 1D problem!

