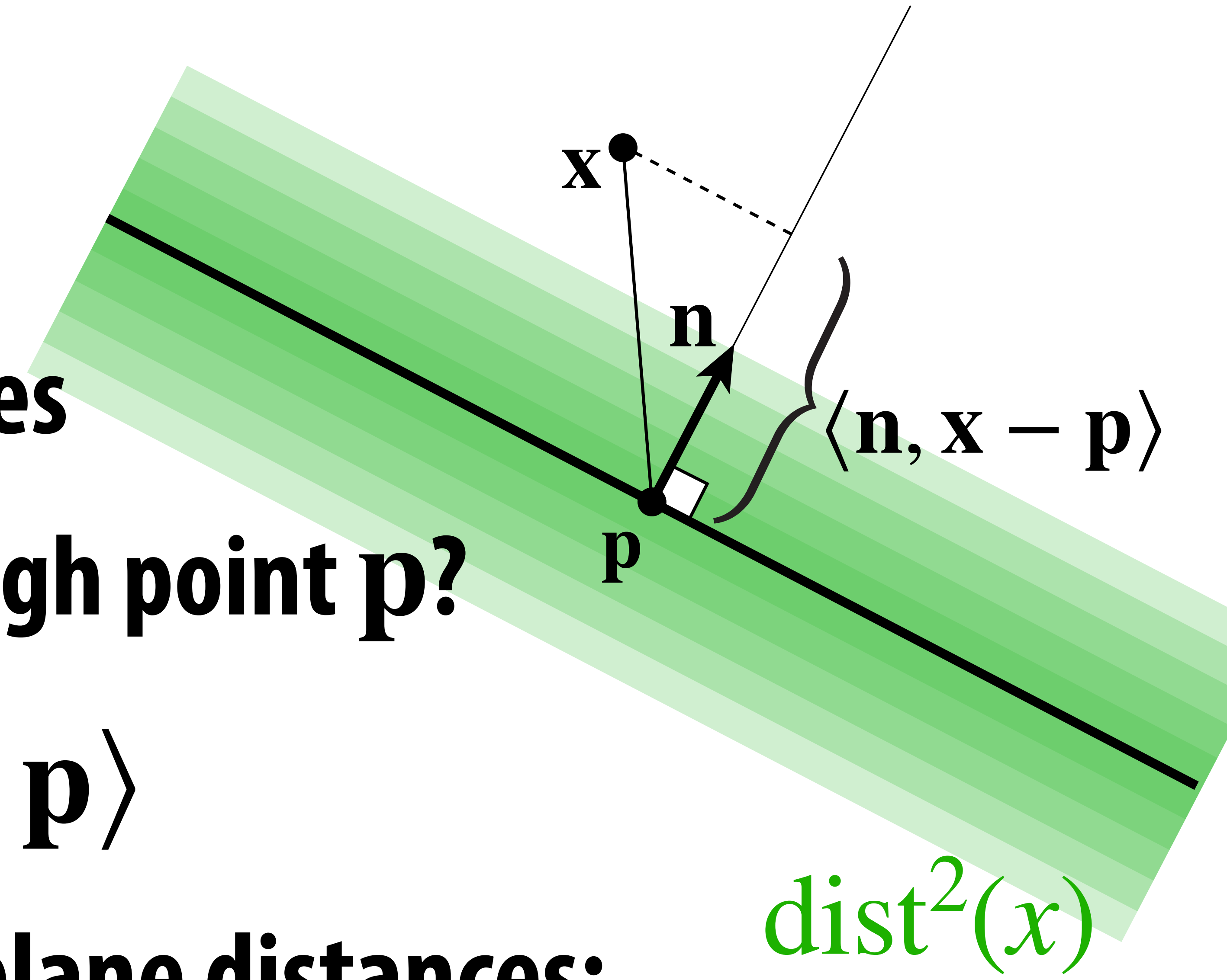
Geometric Queries

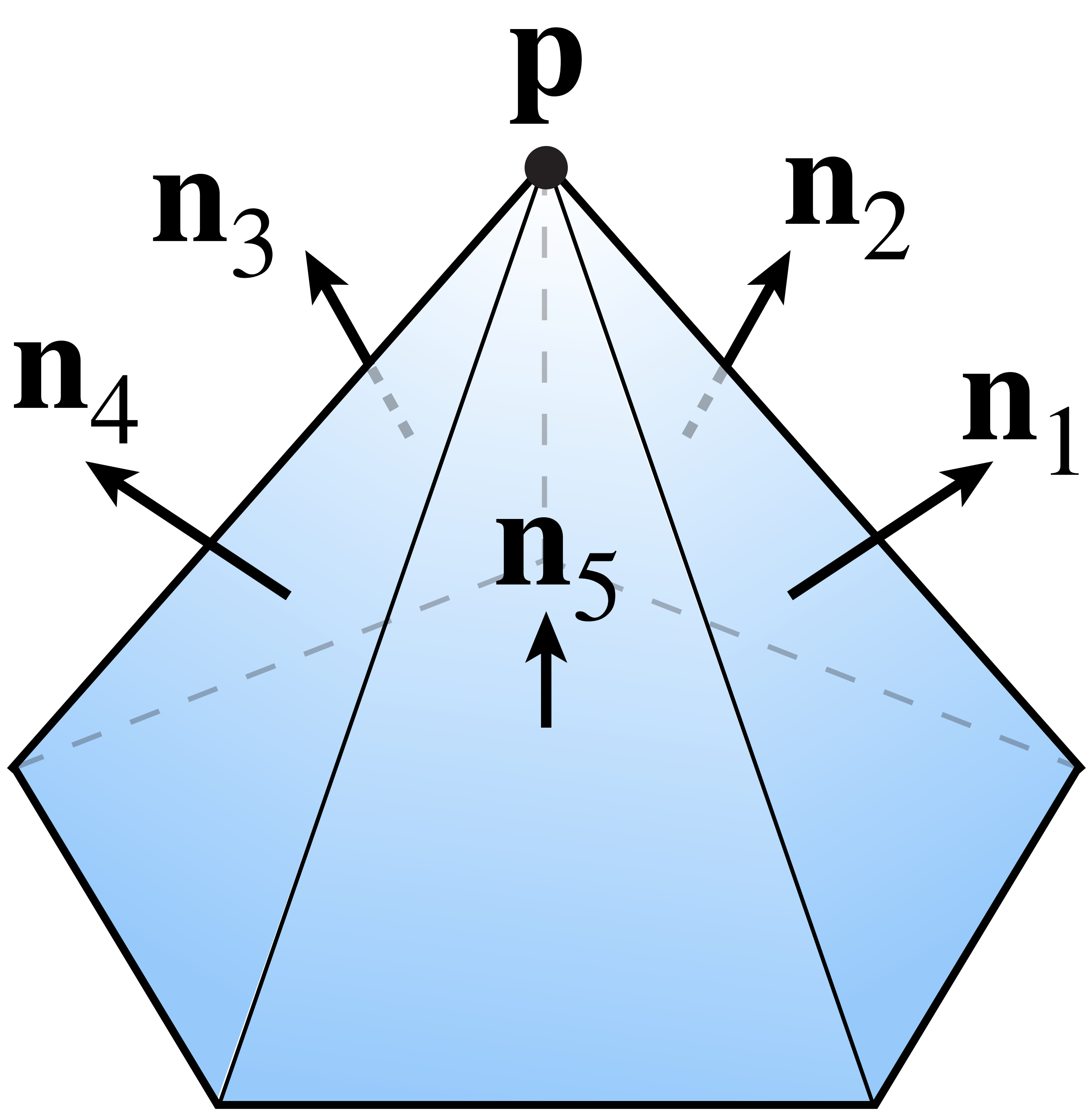
Computer Graphics
CMU 15-462/15-662

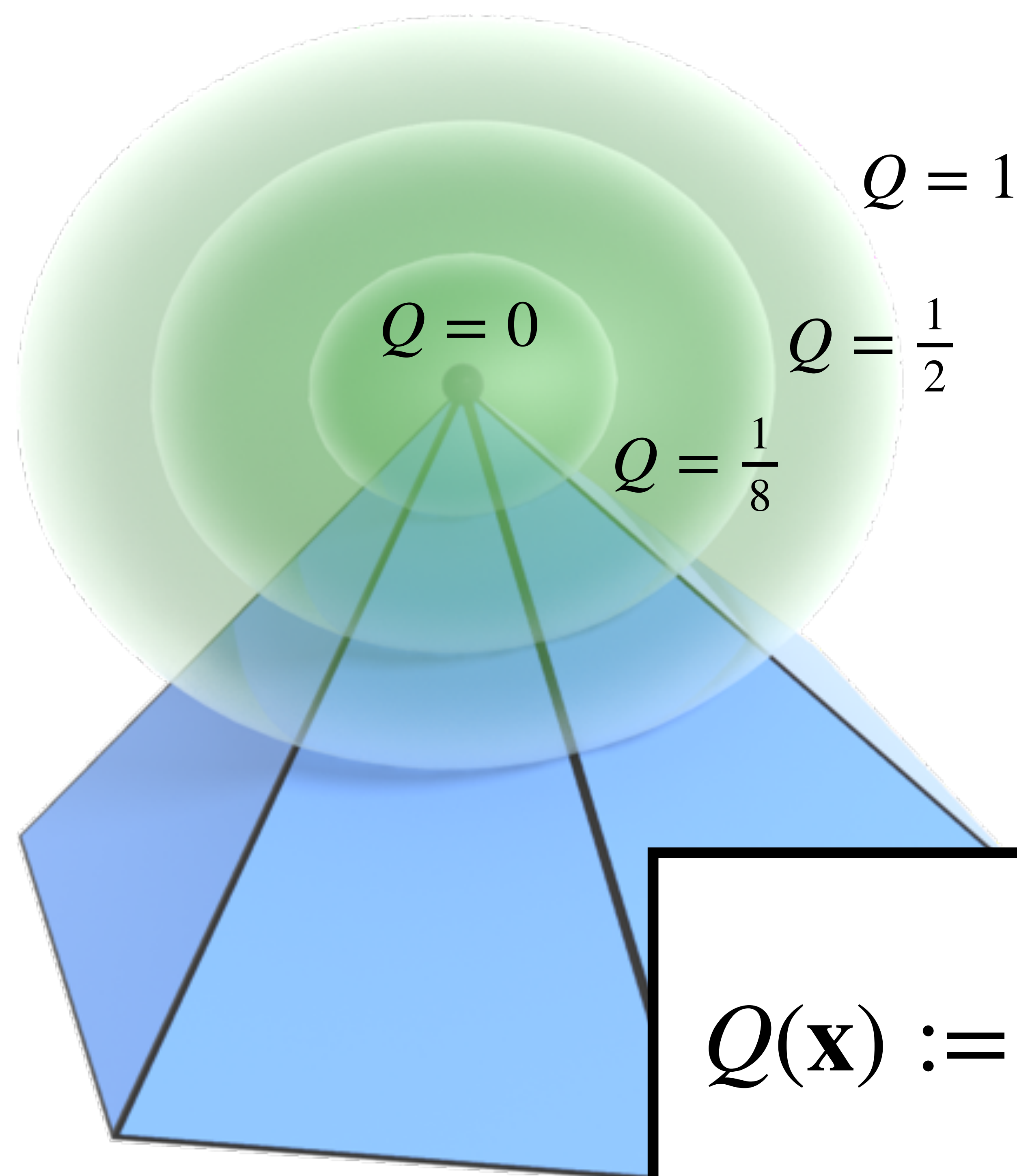
First, a review and wrap-up from Wednesday

Quadric Error Metric



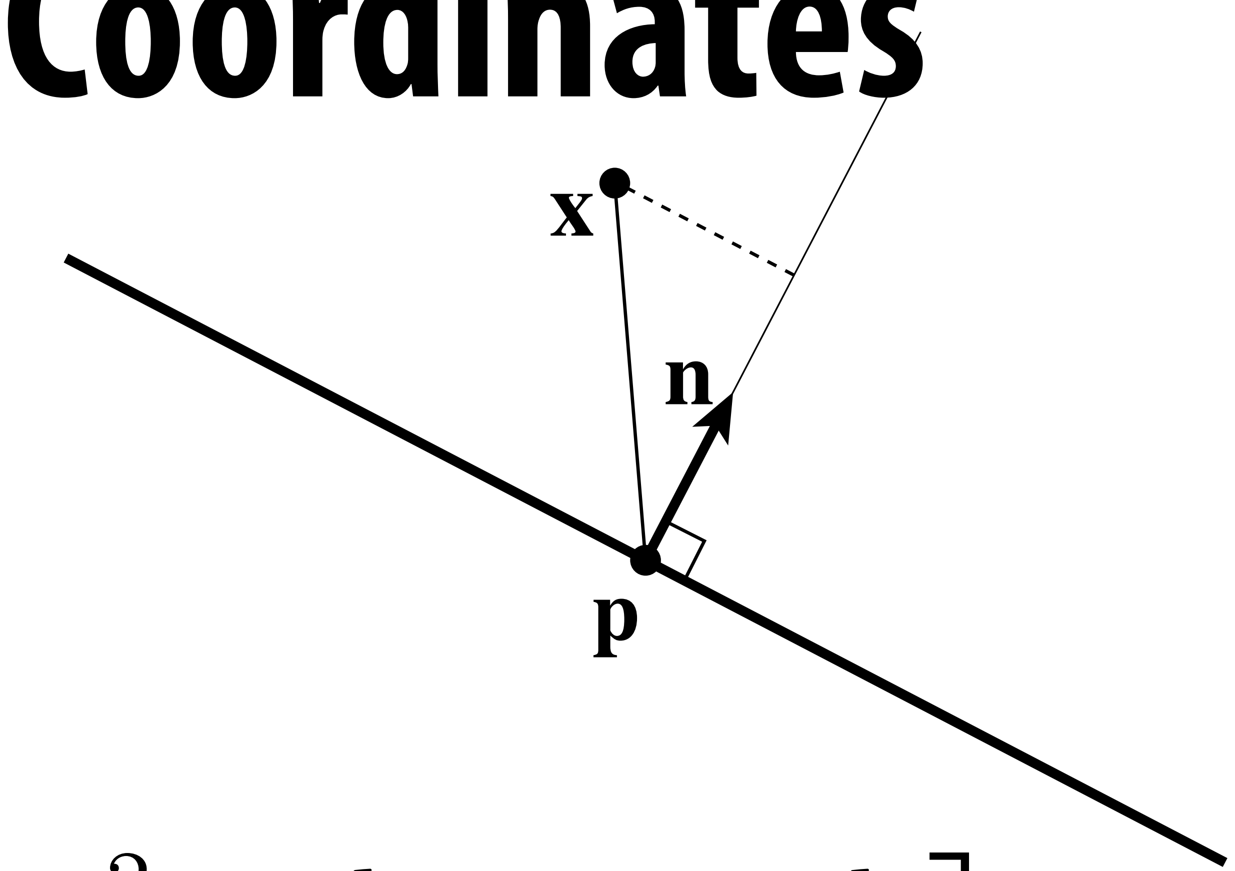
- Approximate distance to a collection of triangles
- Q: Distance to plane w/ normal $\mathbf{n}$ passing through point $\mathbf{p}$?
- A: $\mathrm{dist}(\mathbf{x}) = \langle \mathbf{n}, \mathbf{x} \rangle - \langle \mathbf{n}, \mathbf{p} \rangle = \langle \mathbf{n}, \mathbf{x} - \mathbf{p} \rangle$
- Quadric error is then sum of squared point-to-plane distances:





$$\boxed{Q(\mathbf{x}) := \sum_{i=1}^{k} \langle \mathbf{n}_i, \mathbf{x} - \mathbf{p} \rangle^2}$$

Quadric Error - Homogeneous Coordinates



- Suppose in coordinates we have
  - a query point $\mathbf{x} = (x, y, z)$
  - a normal $\mathbf{n} = (a, b, c)$
  - an offset $d := -\langle \mathbf{n}, \mathbf{p} \rangle$
- In homogeneous coordinates, let
  - $\mathbf{u} := (x, y, z, 1)$
  - $\mathbf{v} := (a, b, c, d)$

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

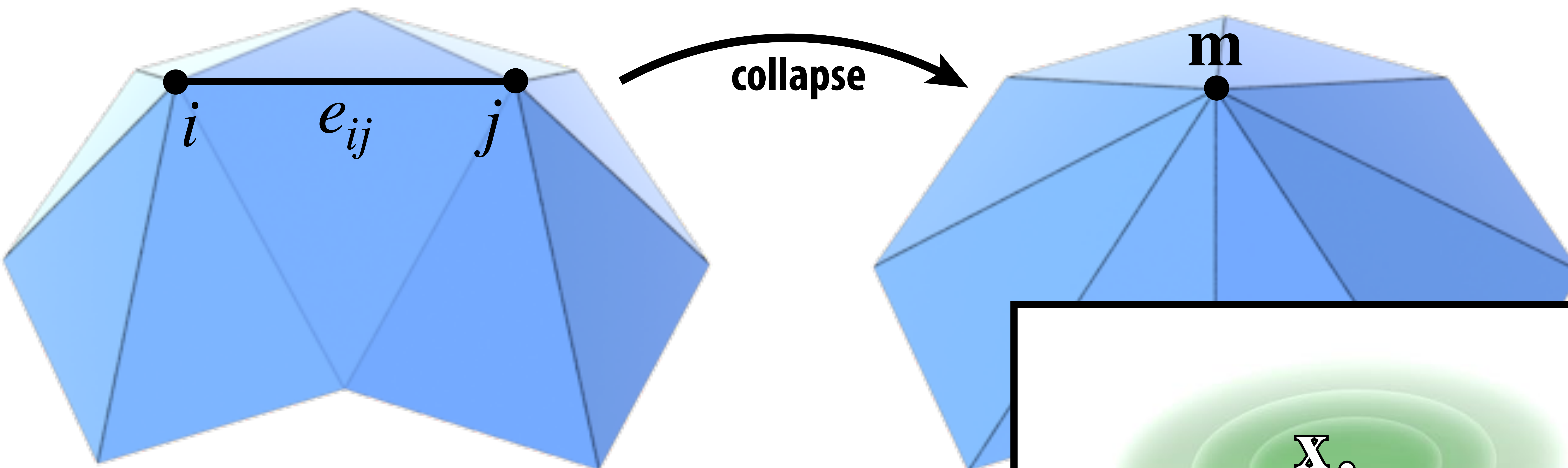
- Signed distance to plane is then just $\langle \mathbf{u}, \mathbf{v} \rangle = ax + by + cz + d$
- Squared distance is $\langle \mathbf{u}, \mathbf{v} \rangle^2 = \mathbf{u}^\mathsf{T}(\mathbf{v}\mathbf{v}^\mathsf{T})\mathbf{u} =: \mathbf{u}^\mathsf{T} K \mathbf{u}$
- Matrix $K = \mathbf{v}\mathbf{v}^T$ encodes squared distance to plane

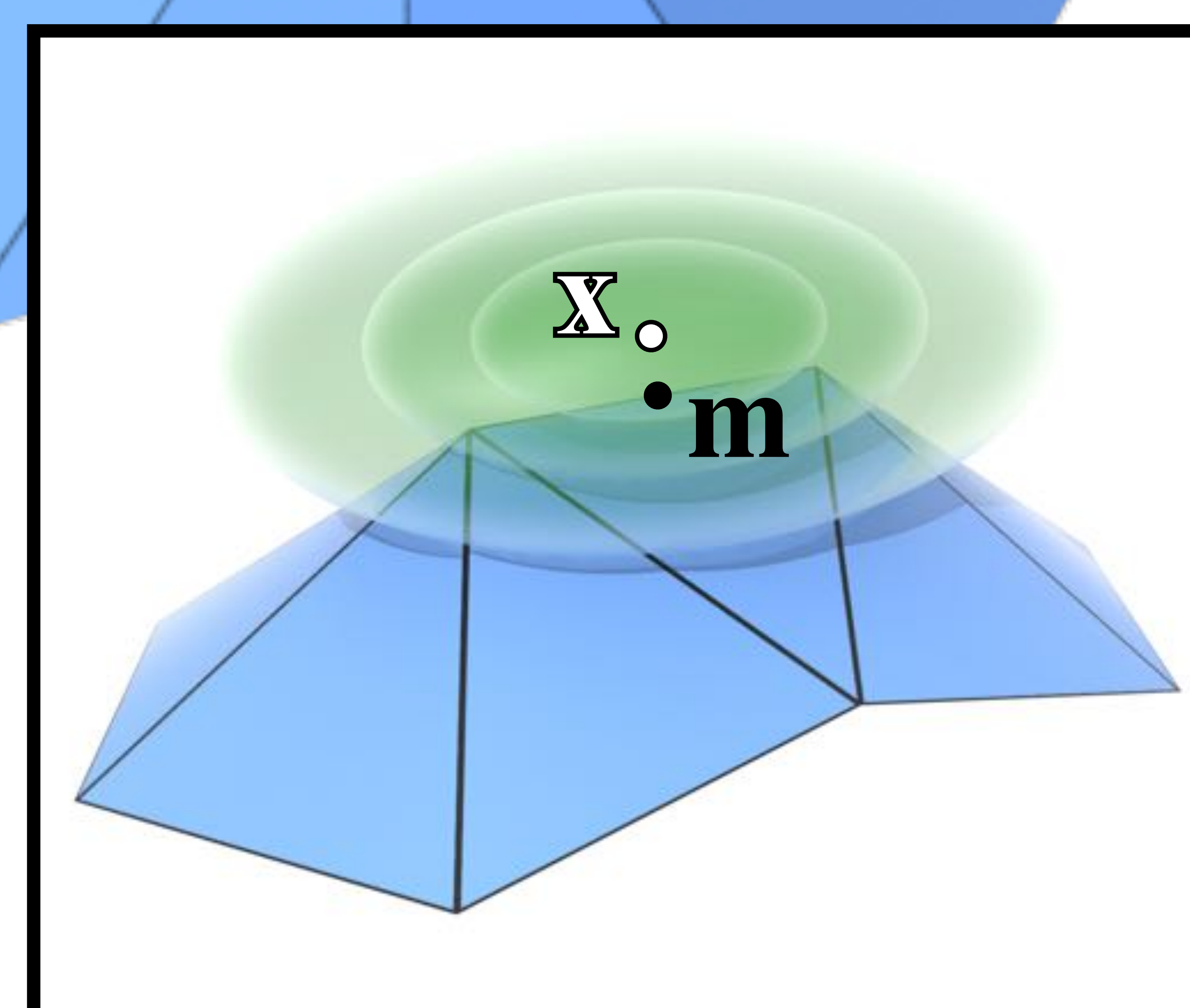
Key idea: <u>sum</u> of matrices $K \iff$ distance to <u>union</u> of planes

$$\mathbf{u}^\mathsf{T} K_1 \mathbf{u} + \mathbf{u}^\mathsf{T} K_2 \mathbf{u} = \mathbf{u}^\mathsf{T}(K_1 + K_2)\mathbf{u}$$

Quadric Error of Edge Collapse

- How much does it cost to collapse an edge e_{ij}?
- Idea: compute midpoint $\mathbf{m}$, measure error $Q(\mathbf{m}) = \mathbf{m}^\mathsf{T}(K_i + K_j)\mathbf{m}$
- Error becomes "score" for e_{ij}, determining priority





- Better idea: find point $\mathbf{x}$ that minimizes error!
- Ok, but how do we minimize quadric error?

Review: Minimizing a Quadratic Function

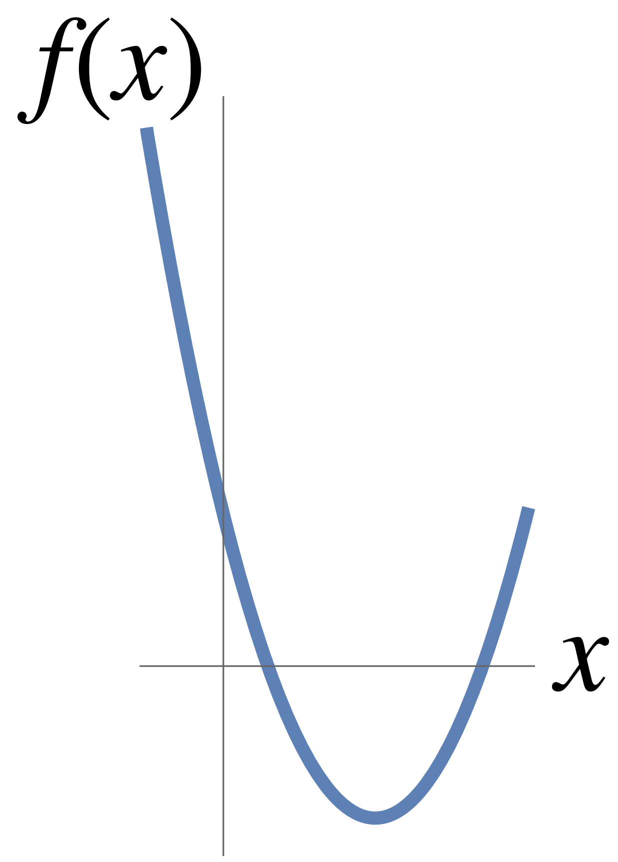
- Suppose you have a function $f(x) = ax^2 + bx + c$
- Q: What does the graph of this function look like?
- Could also look like this!
- Q: How do we find the minimum?
- A: Find where the function looks "flat" if we zoom in really close
- I.e., find point x where 1st derivative vanishes:

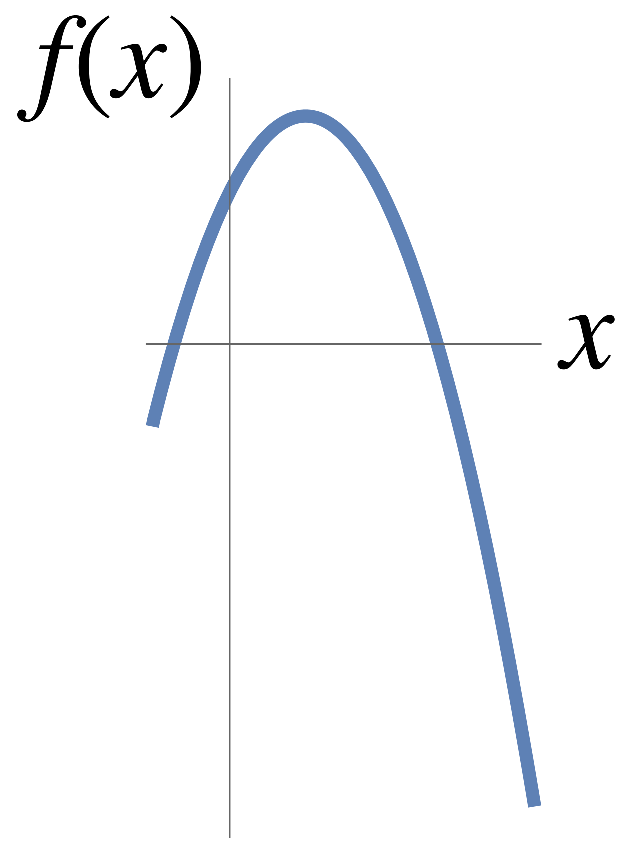
$$f'(x) = 0$$

$$2ax + b = 0$$

$$x = -b/2a$$

(What does x describe for the second function?)





Minimizing Quadratic Polynomial

- Not much harder to minimize a quadratic polynomial in n variables
- Can always write in terms of a symmetric matrix A
- E.g., in 2D: $f(x, y) = ax^2 + bxy + cy^2 + dx + ey + g$

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \qquad A = \begin{bmatrix} a & b/2 \\ b/2 & c \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} d \\ e \end{bmatrix}$$

$$f(x, y) = \mathbf{x}^\mathsf{T} A \mathbf{x} + \mathbf{u}^\mathsf{T} \mathbf{x} + g$$

(will have this same form for any n)

- Q: How do we find a critical point (min/max/saddle)?
- A: Set derivative to zero!

$$2A\mathbf{x} + \mathbf{u} = 0$$

$$\mathbf{x} = -\frac{1}{2} A^{-1} \mathbf{u}$$

(compare with our 1D solution)

$$x = -b/2a$$

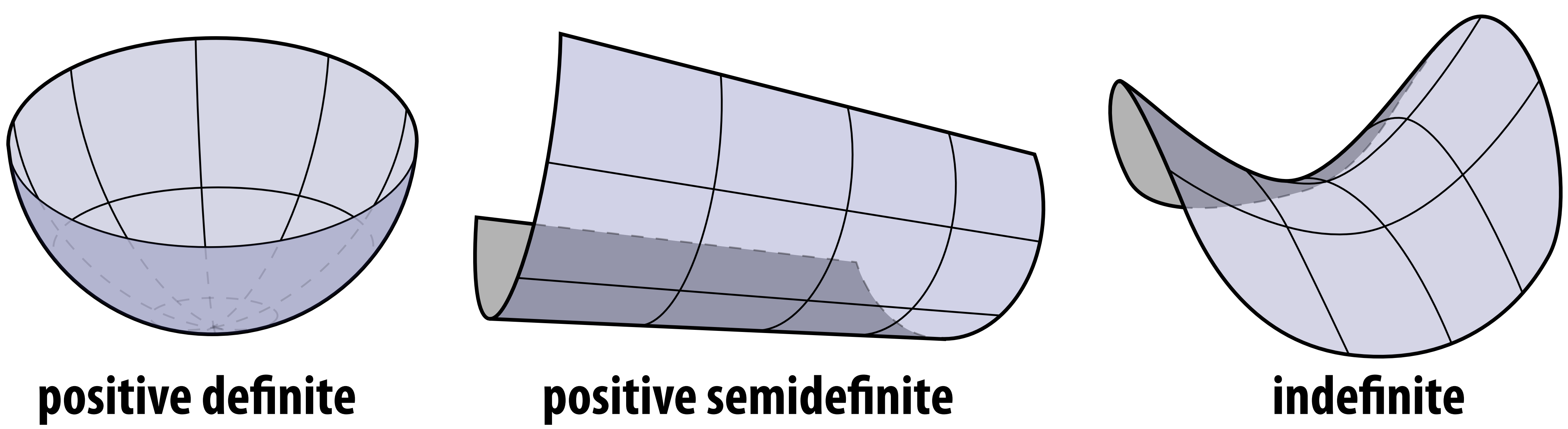
(Can you show this is true, at least in 2D?)

Positive Definite Quadratic Form

- Just like our 1D parabola, critical point is not always a min!
- Q: In 2D, 3D, nD, when do we get a minimum?
- A: When matrix A is positive-definite:

$$\mathbf{x}^{\mathsf{T}} A\, \mathbf{x} > 0 \quad \forall\, \mathbf{x}$$

- 1D: Must have $xax = ax^2 > 0$. In other words: a is positive!
- 2D: Graph of function looks like a "bowl":



Positive-definiteness **extremely important** in computer graphics: means we can find minimizers by solving linear equations. Starting point for many algorithms (geometry processing, simulation, ...)

Minimizing Quadric Error

- Find "best" point for edge collapse by minimizing quadratic form

$$\min_{\mathbf{u}\in\mathbb{R}^4} \mathbf{u}^T K \mathbf{u}$$

- Already know fourth (homogeneous) coordinate for a point is 1
- So, break up our quadratic function into two pieces:

$$\begin{bmatrix} \mathbf{x}^\mathsf{T} & 1 \end{bmatrix} \begin{bmatrix} B & \mathbf{w} \\ \mathbf{w}^\mathsf{T} & d^2 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$$

$$= \mathbf{x}^\mathsf{T} B \mathbf{x} + 2\mathbf{w}^\mathsf{T}\mathbf{x} + d^2$$

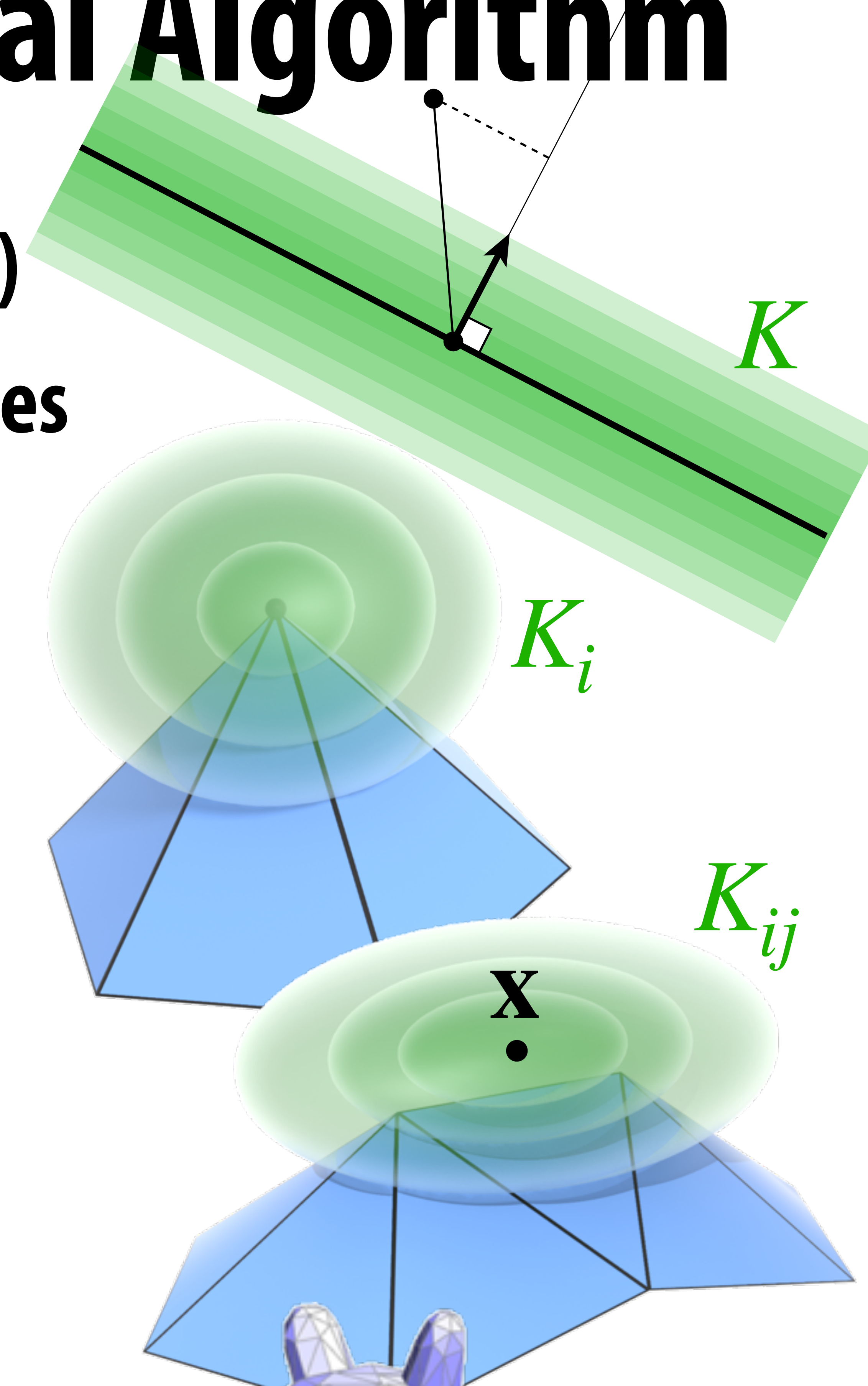
- Now we have a quadratic polynomial in the unknown position $\mathbf{x} \in \mathbb{R}^3$
- Can minimize as before:

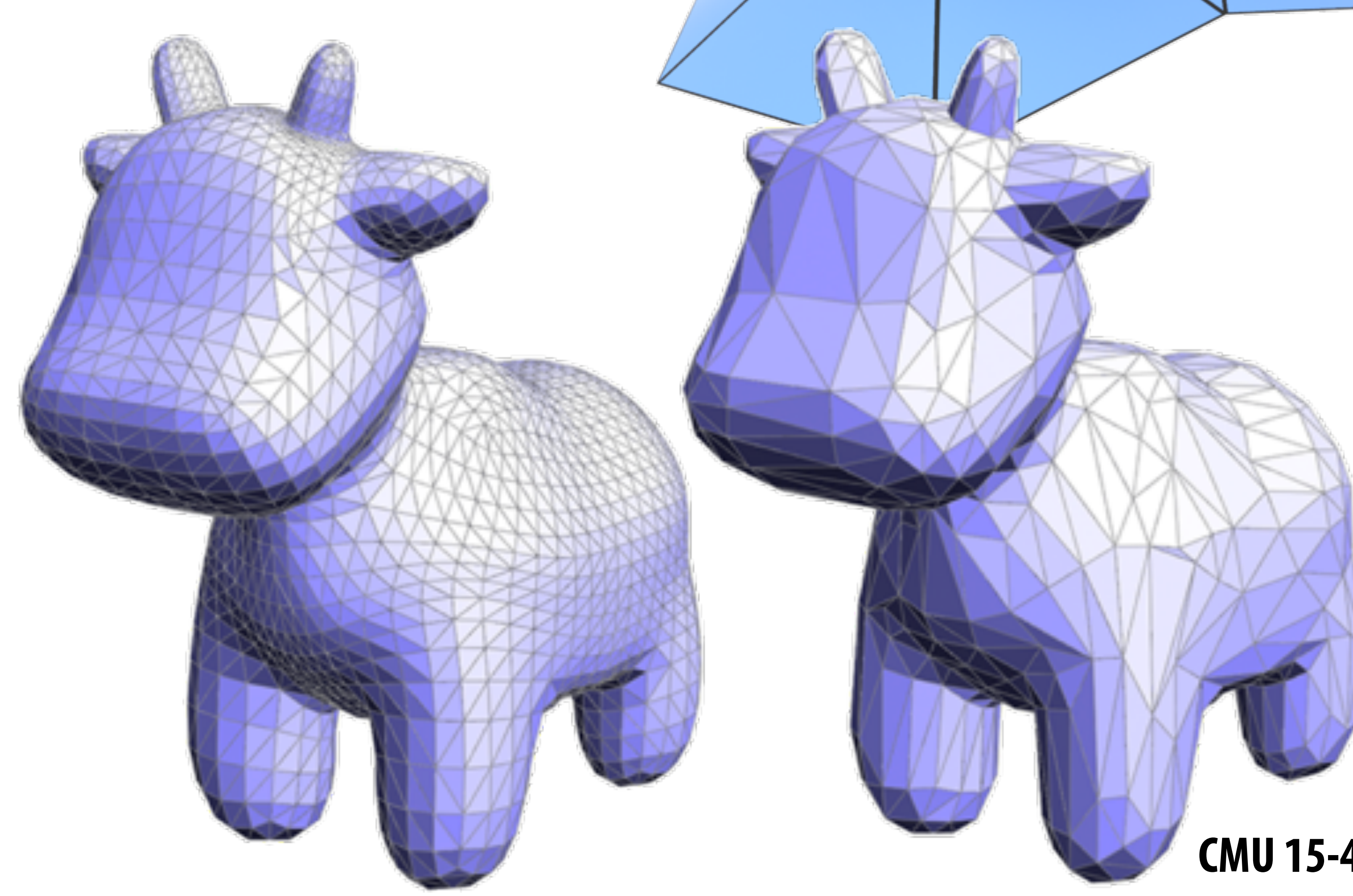
$$2B\mathbf{x} + 2\mathbf{w} = 0 \qquad \iff \qquad \mathbf{x} = -B^{-1}\mathbf{w}$$

Q: Why should B be positive-definite?

Quadric Error Simplification: Final Algorithm

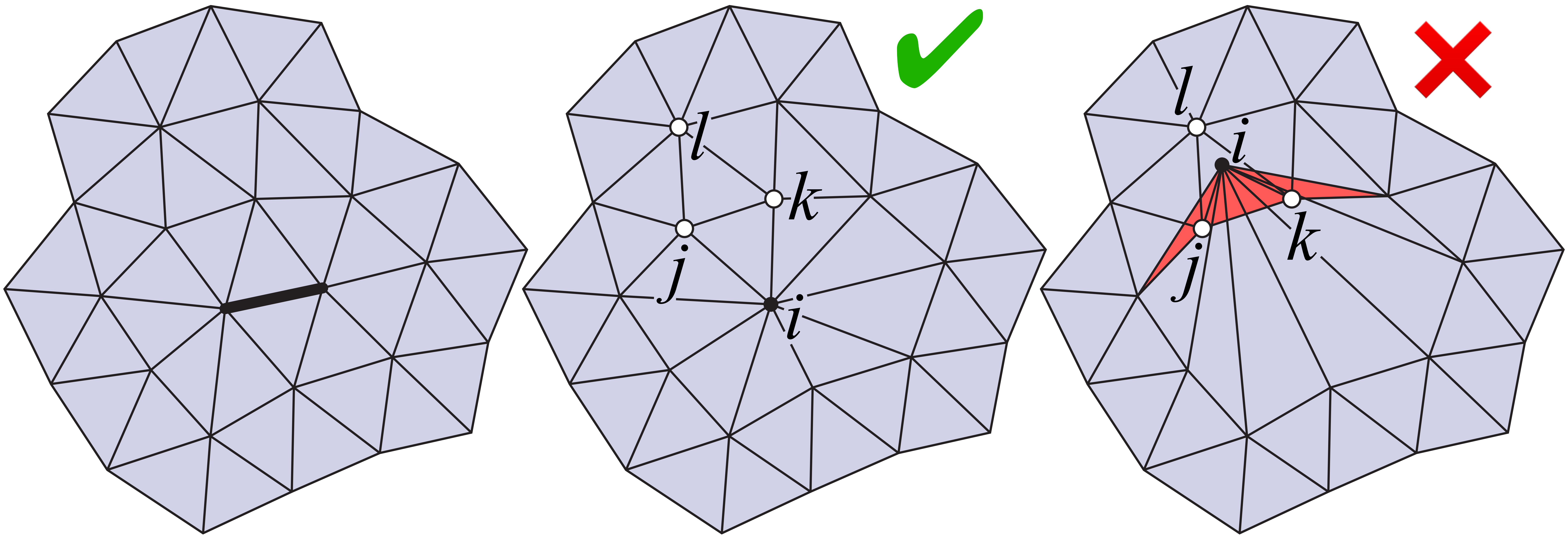
- Compute K for each triangle (squared distance to plane)
- Set K_i at each vertex to sum of Ks from incident triangles
- For each edge e_{ij}:
  - set $K_{ij} = K_i + K_j$
  - find point $\mathbf{x}$ minimizing error, set cost to $K_{ij}(\mathbf{x})$
- Until we reach target number of triangles:
  - collapse edge e_{ij} with smallest cost to optimal point $\mathbf{x}$
  - set quadric at new vertex to K_{ij}
  - update cost of edges touching new vertex
- More details in assignment writeup!





Quadric Simplification—Flipped Triangles

- Depending on where we put the new vertex, one of the new triangles might be "flipped" (normal points in instead of out):

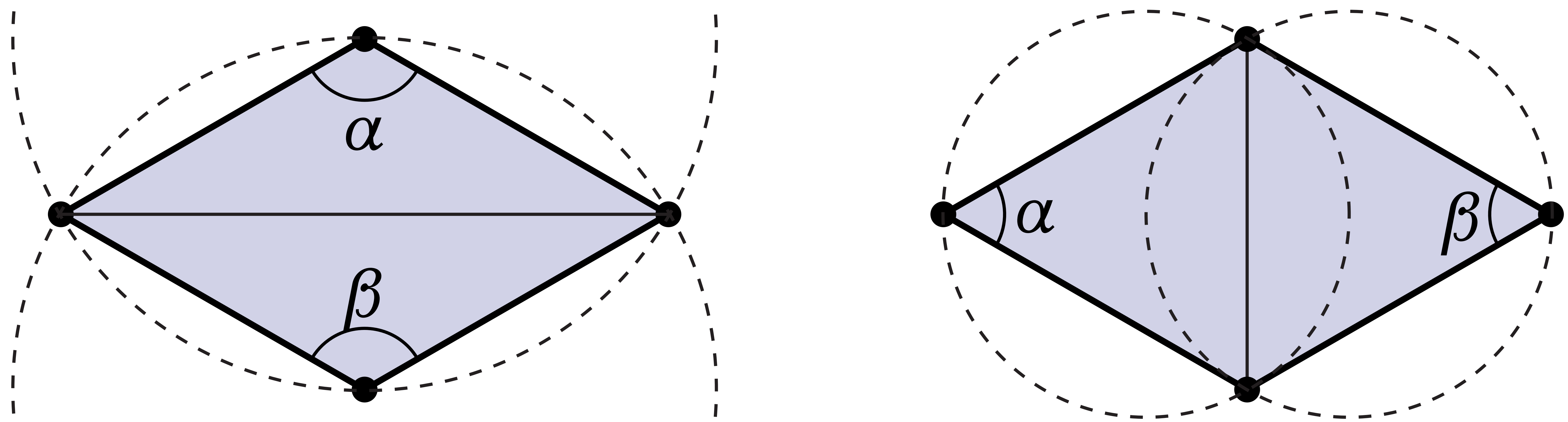


- Easy solution: for each triangle ijk touching collapsed vertex i, consider normals N_{ijk} and N_{kjl} (where kjl is other triangle containing edge jk)
- If $\langle N_{ijk}, N_{kjl} \rangle$ is negative, don't collapse this edge!

What if we're happy with the number of triangles, but want to improve quality?

How do we make a mesh "more Delaunay"?

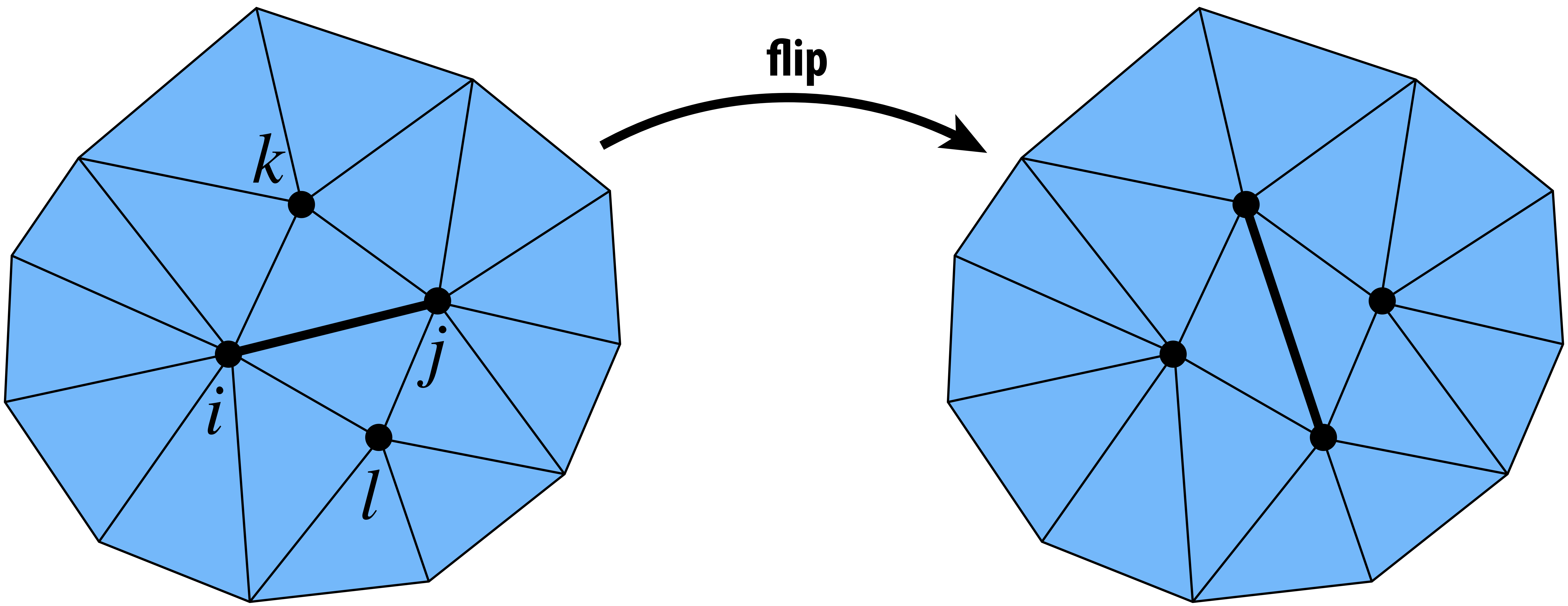
- Already have a good tool: edge flips!
- If $\alpha + \beta > \pi$, flip it!



- FACT: in 2D, flipping edges eventually yields Delaunay mesh
- Theory: worst case $O(n^2)$; doesn't always work for surfaces in 3D
- Practice: simple, effective way to improve mesh quality

Alternatively: how do we improve degree?

- Same tool: edge flips!
- If total deviation from degree-6 gets smaller, flip it!

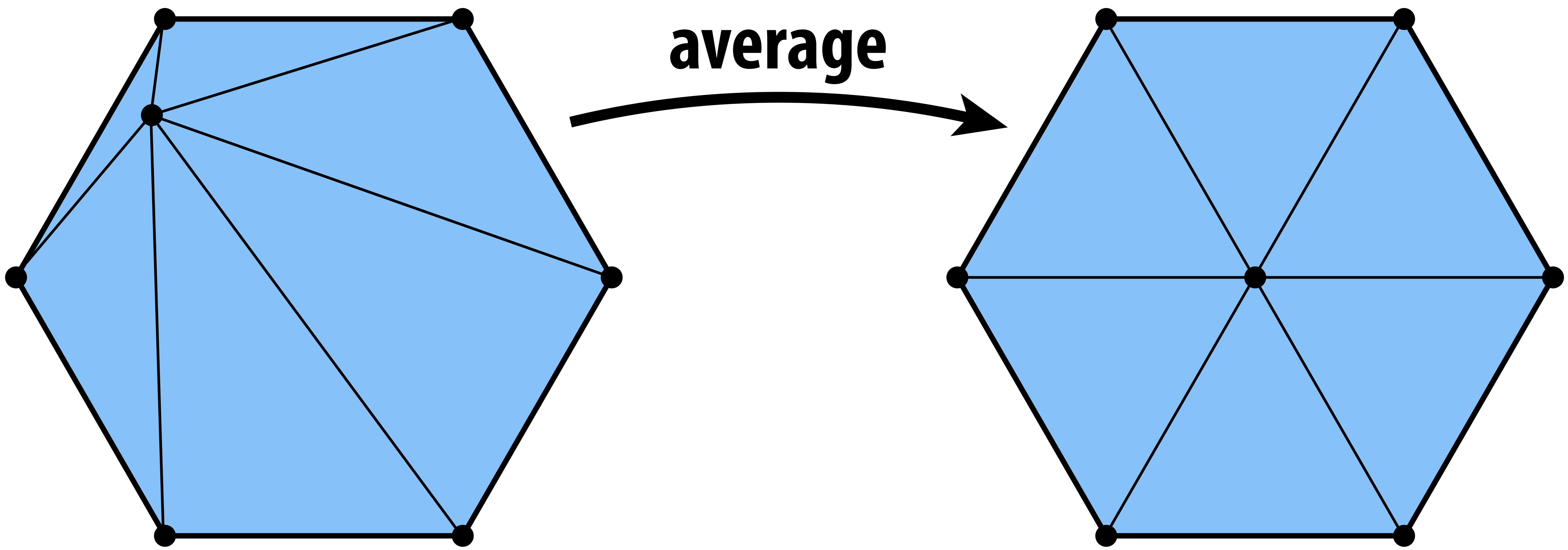


total deviation: $|d_i - 6| + |d_j - 6| + |d_k - 6| + |d_l - 6|$

- FACT: average degree approaches 6 as number of elements increases
- Iterative edge flipping acts like "discrete diffusion" of degree
- No (known) guarantees; works well in practice

How do we make a triangles “more round”?

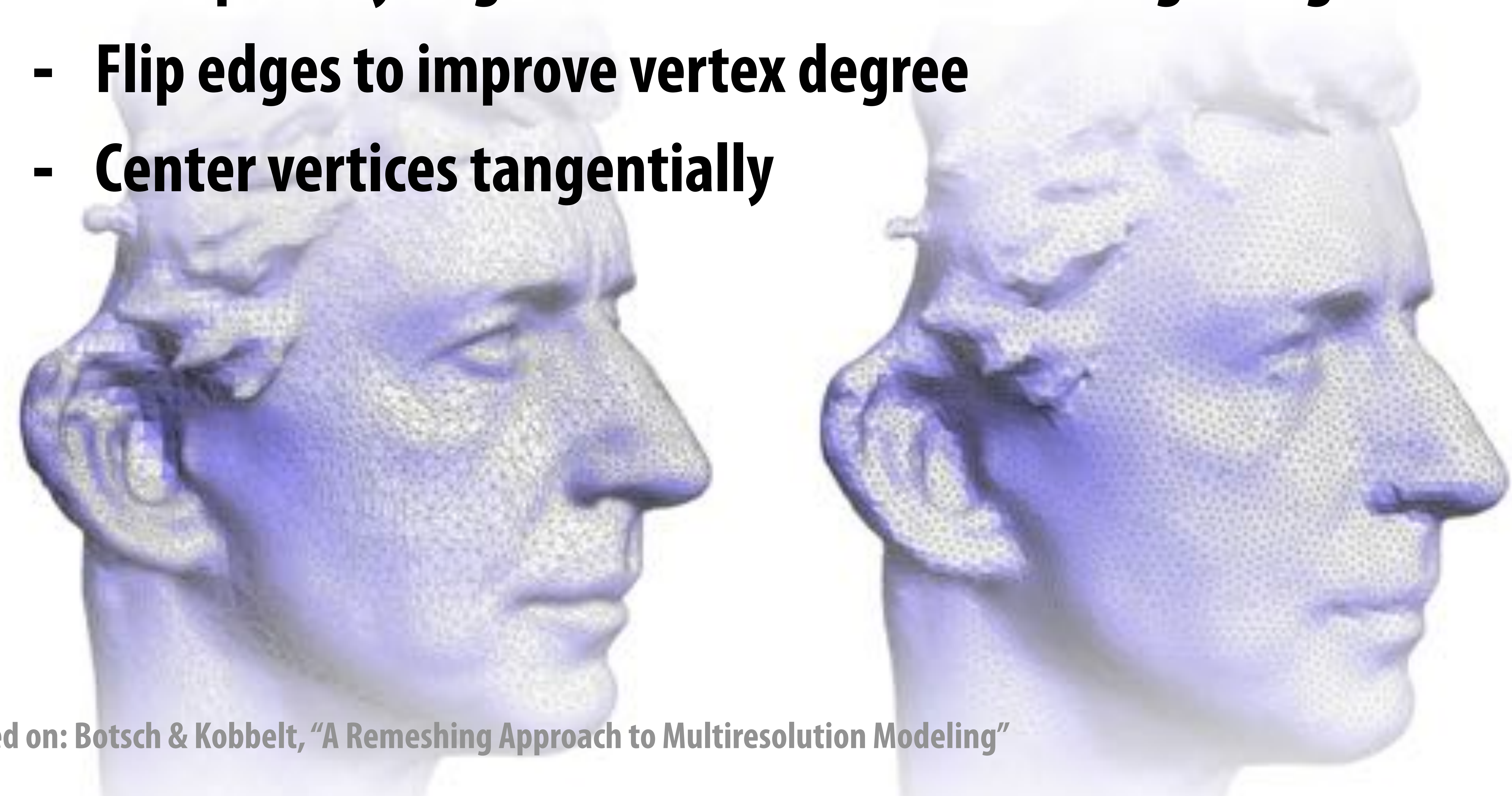
- Delaunay doesn’t guarantee triangles are “round” (angles near 60°)
- Can often improve shape by centering vertices:



- Simple version of technique called “Laplacian smoothing”
- On surface: move only in tangent direction
- How? Remove normal component from update vector

Isotropic Remeshing Algorithm

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

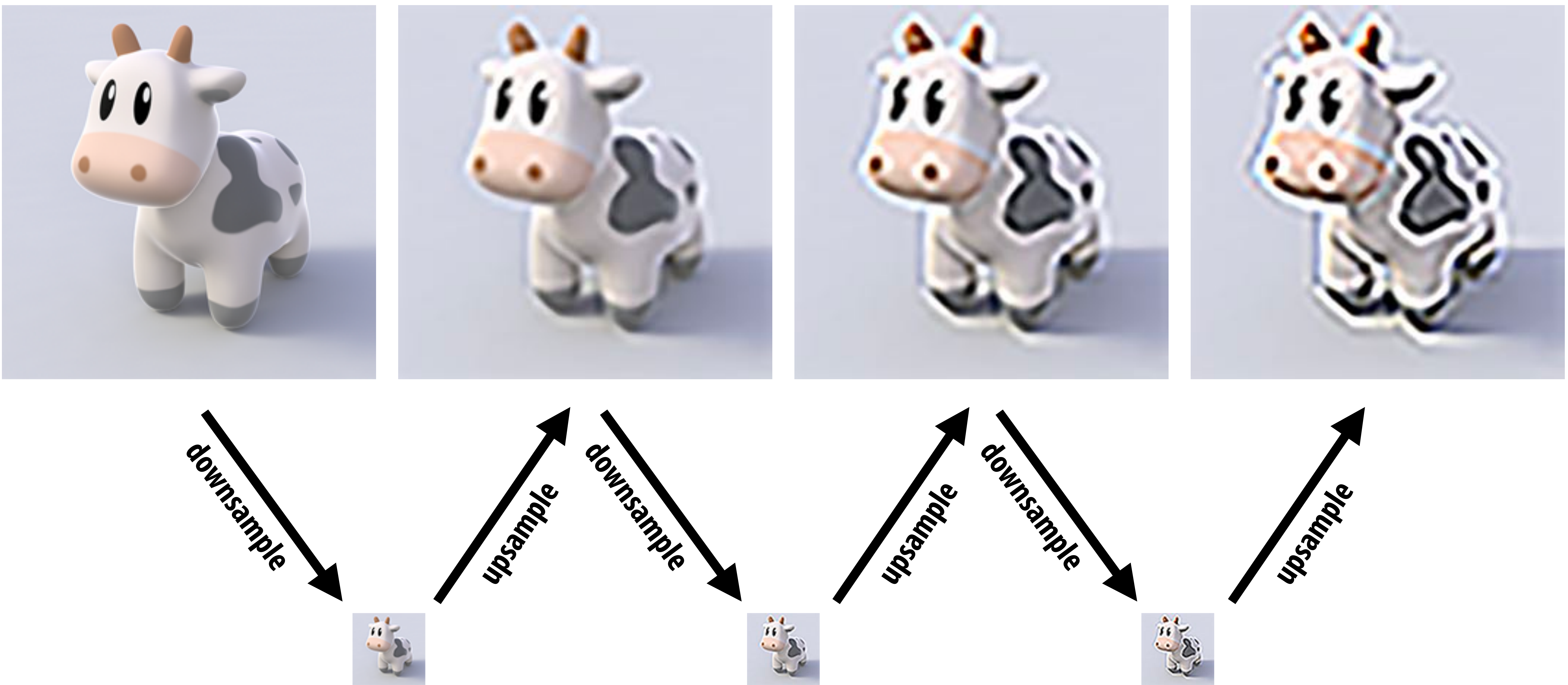


Based on: Botsch & Kobbelt, "A Remeshing Approach to Multiresolution Modeling"

What can go wrong when you resample a signal?

Danger of Resampling

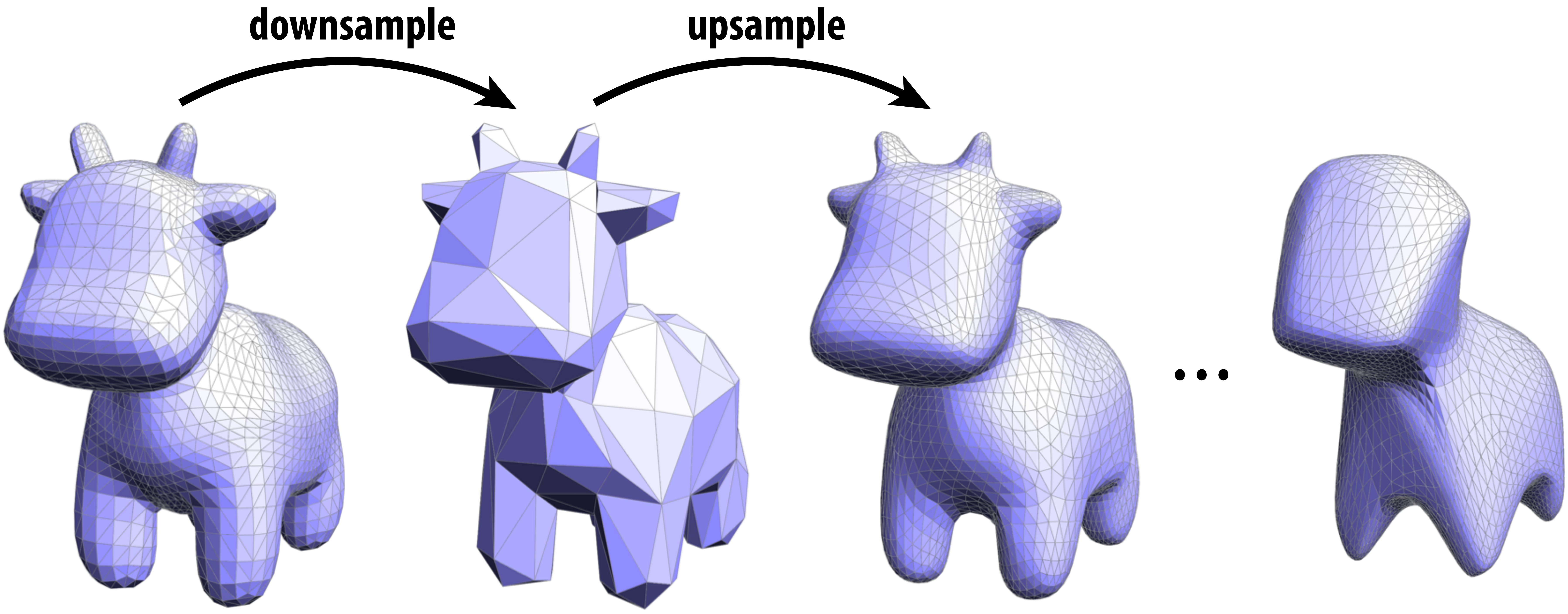
Q: What happens if we repeatedly resample an image?



A: Signal quality degrades!

Danger of Resampling

Q: What happens if we repeatedly resample a mesh?



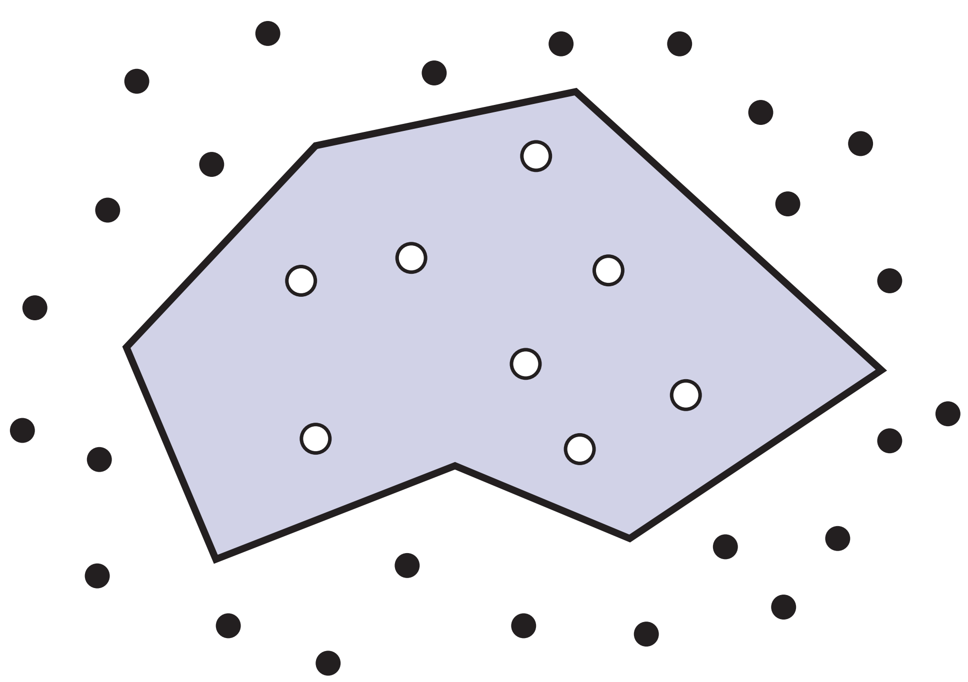
A: Signal also degrades!

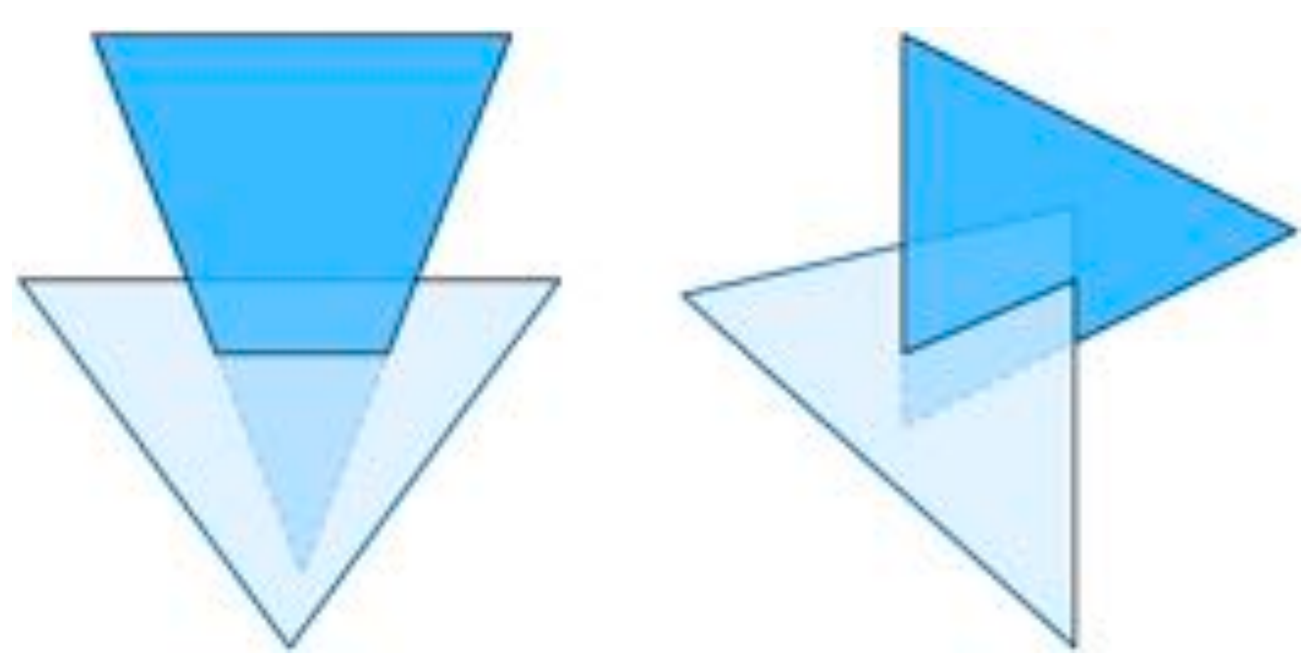
But wait: we have the original signal (mesh).

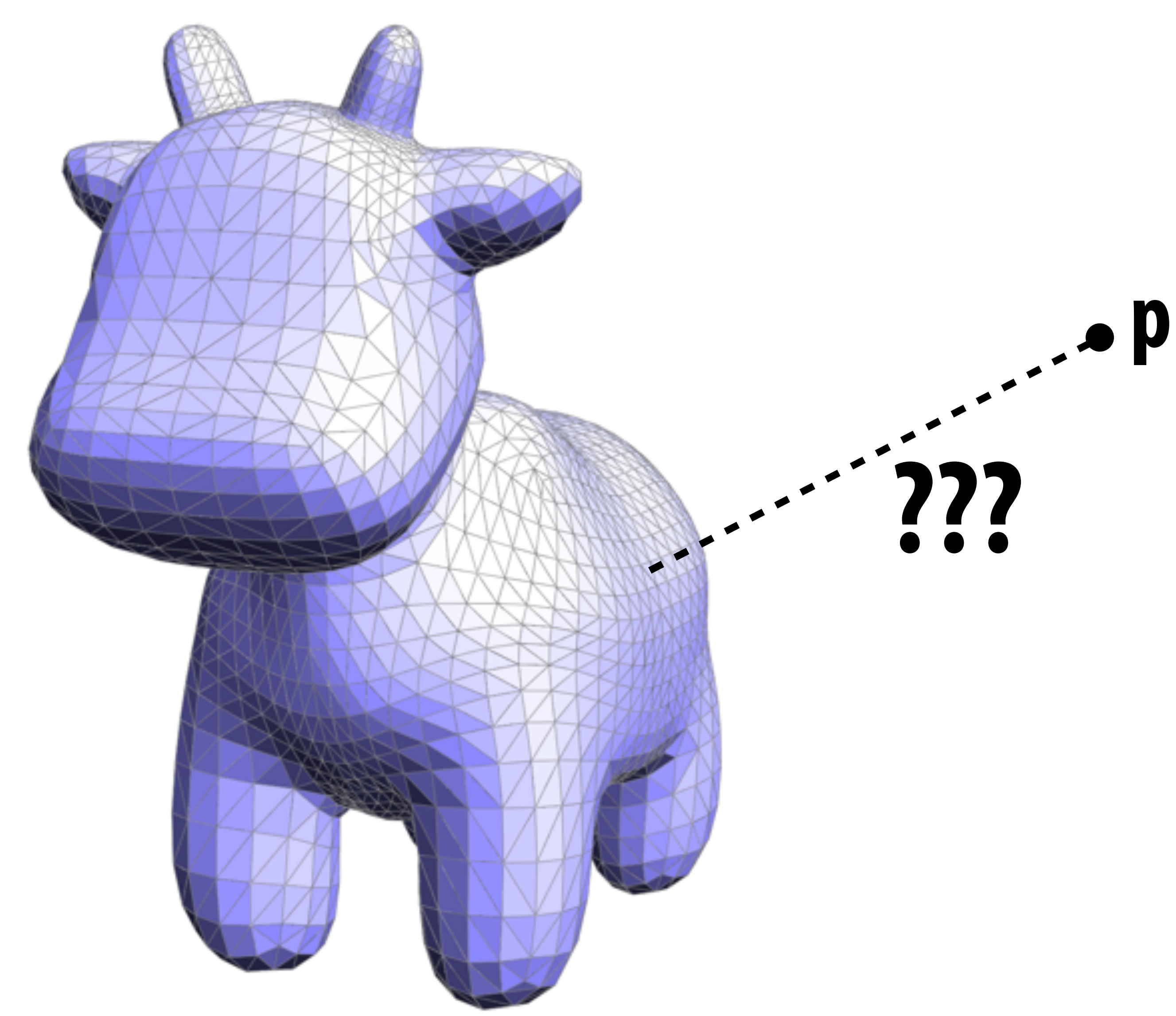
Why not just project each new sample point onto the closest point of the original mesh?

How do we project onto the original surface?

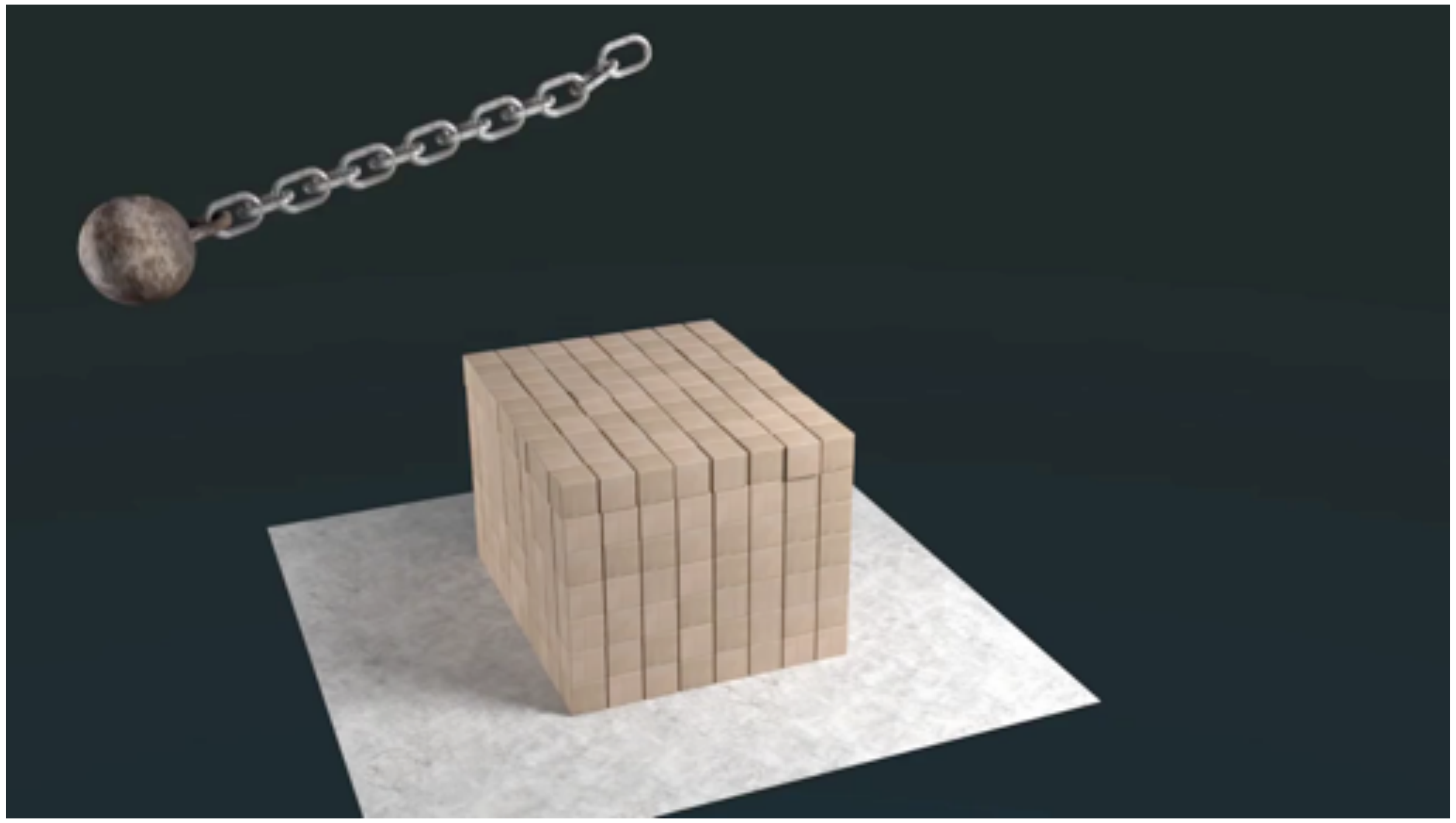
- Q: Given a point, in space, how do we find the closest point on a surface? Are we inside or outside the surface? How do we find intersection of two triangles? Etc.
- Do implicit/explicit representations make such tasks easier?
- What's the cost of the naïve algorithm, and how do we accelerate such queries for large meshes?
- So many questions!

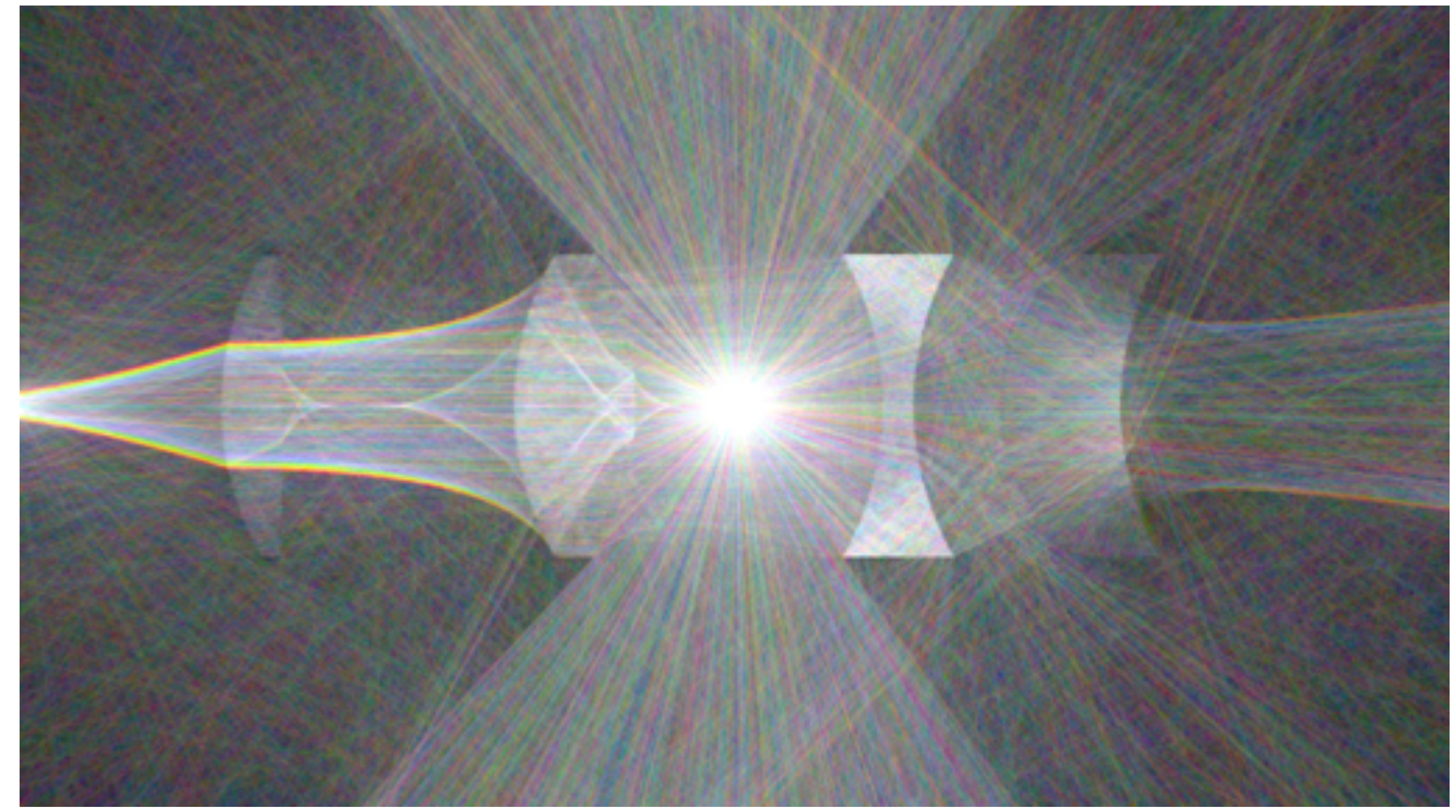


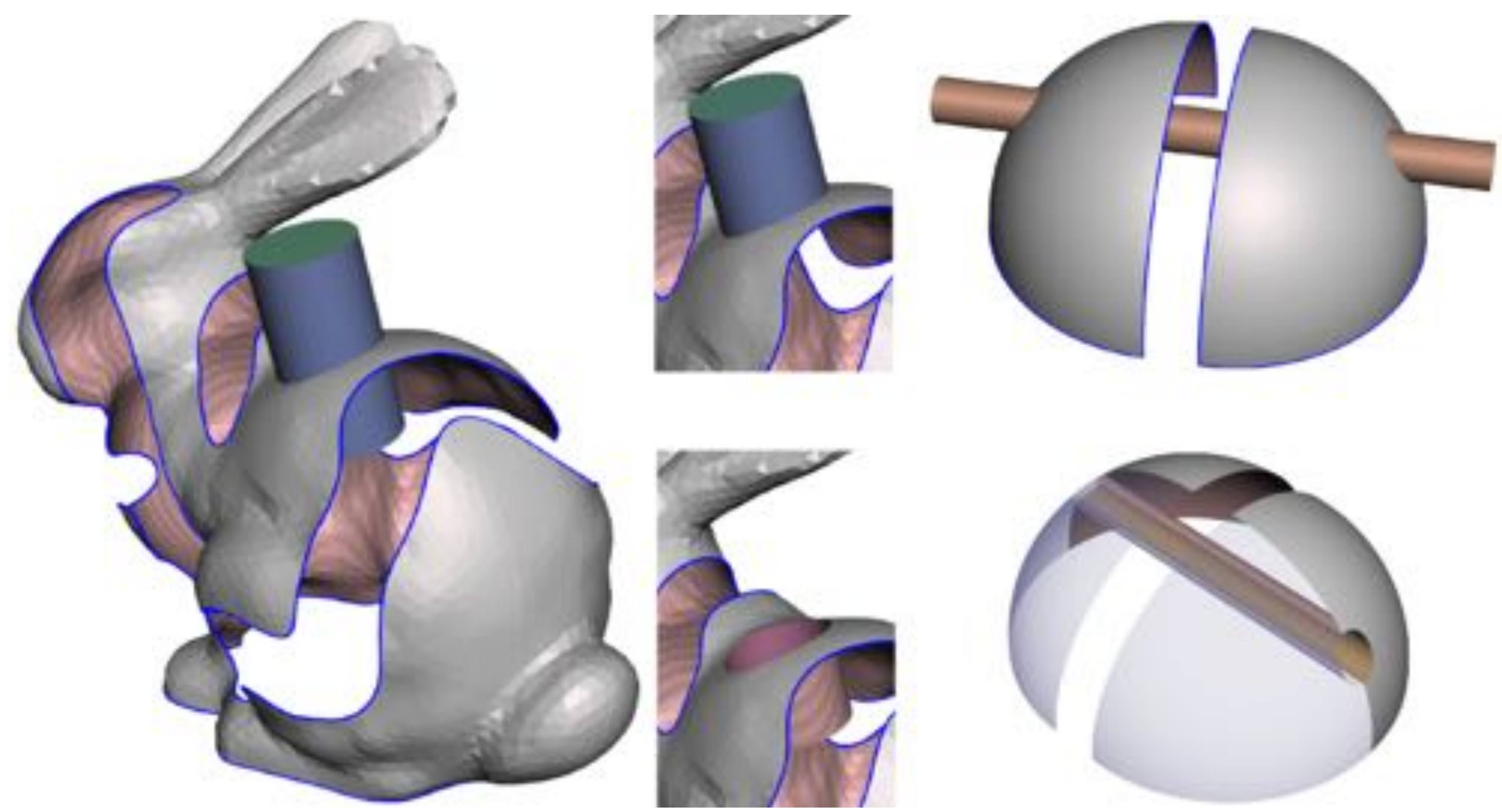




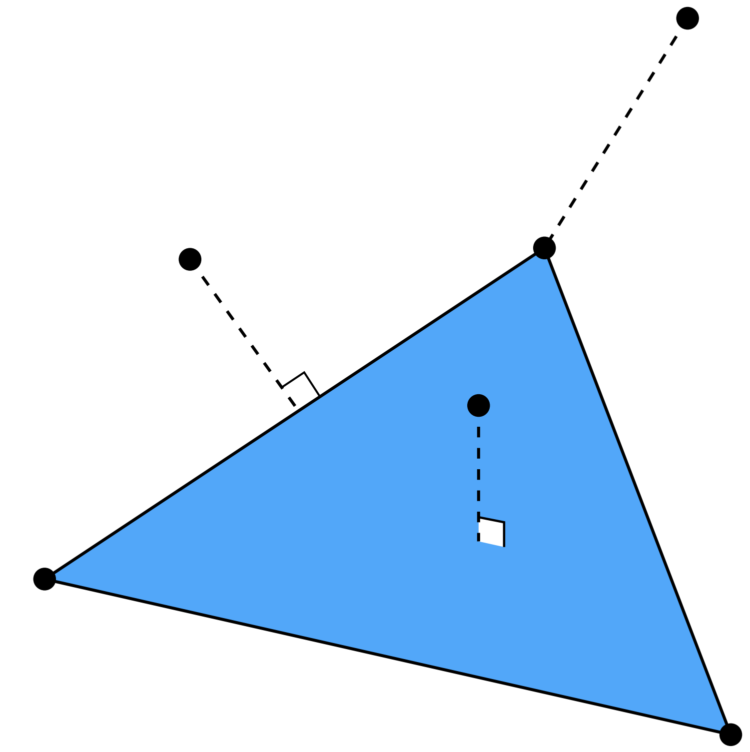
Geometric Queries—Motivation



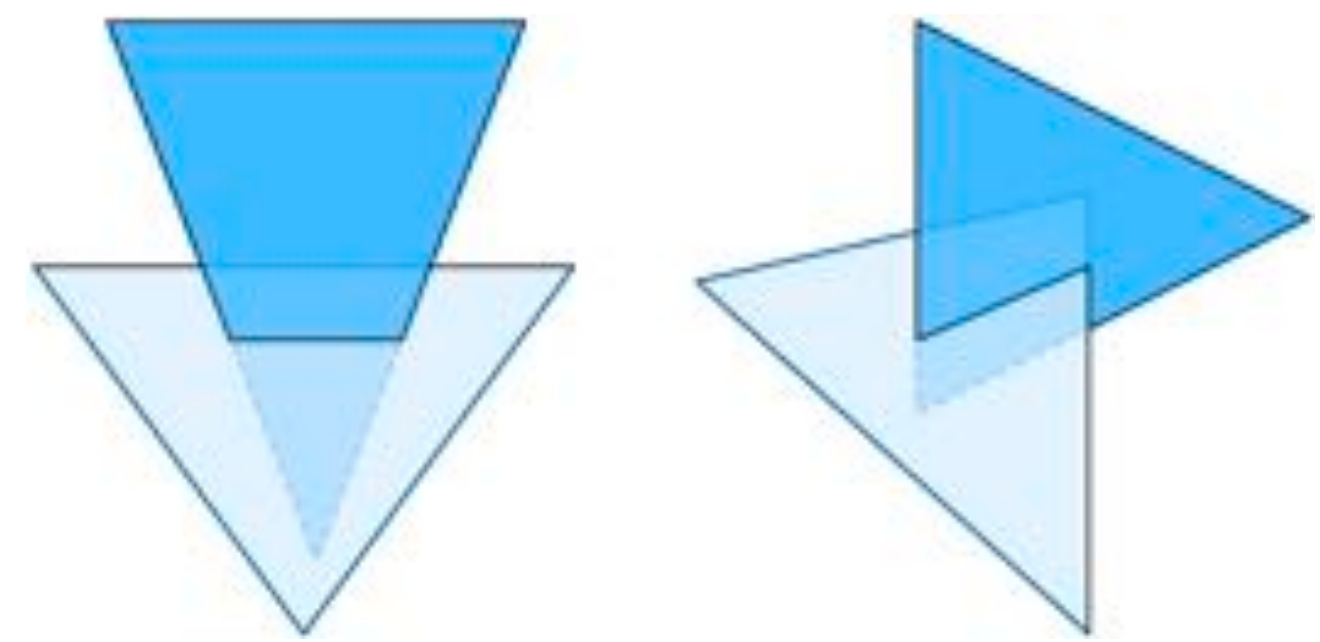




Many types of geometric queries



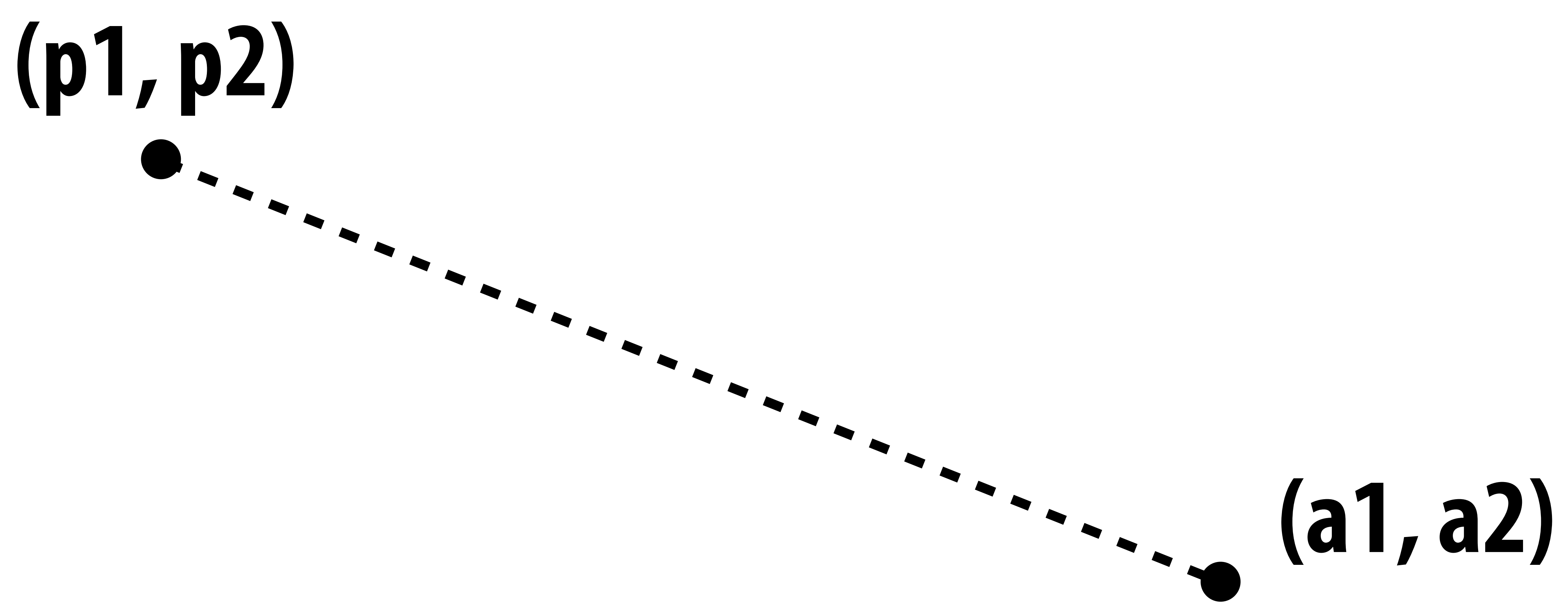
- Already identified need for "closest point" query
- Plenty of other things we might like to know:
  - Do two triangles intersect?
  - Are we inside or outside an object?
  - Does one object contain another?
  - ...



- Data structures we've seen so far not really designed for this...
- Need some new ideas!
- TODAY: come up with simple (read: slow) algorithms.
- NEXT TIME: intelligent ways to accelerate geometric queries.

Warm up: closest point on point

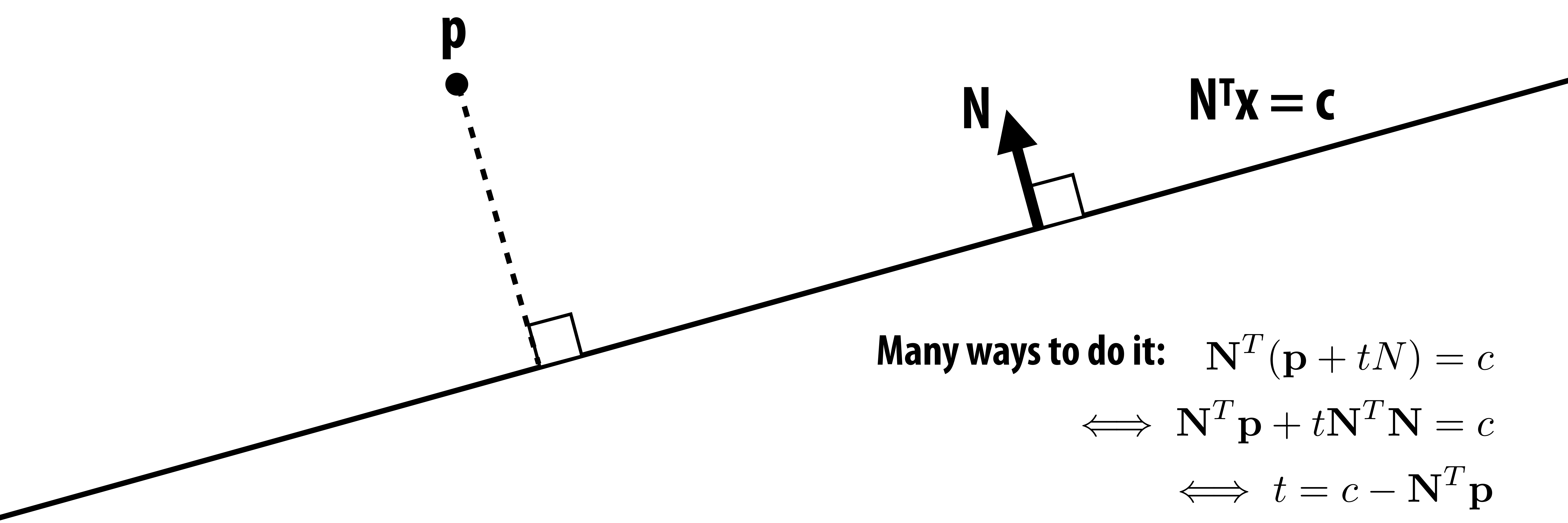
- Goal is to find the point on a mesh closest to a given point.
- Much simpler question: given a query point (p1,p2), how do we find the closest point on the point (a1,a2)?



Bonus question: what's the distance?

Slightly harder: closest point on line

- Now suppose I have a line $\mathbf{N}^T\mathbf{x} = c$, where $\mathbf{N}$ is the unit normal
- How do I find the point closest to my query point $\mathbf{p}$?

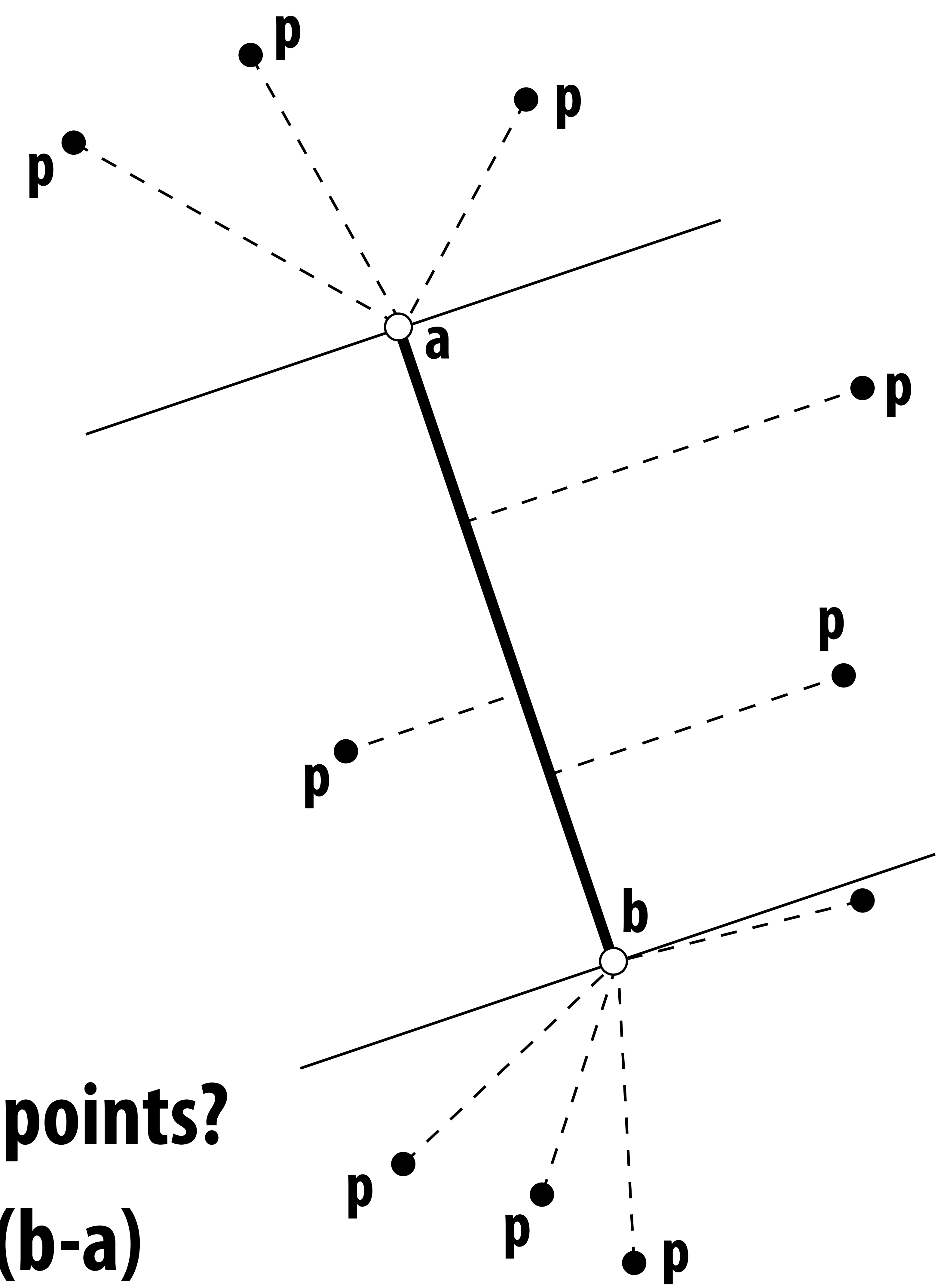


Many ways to do it:

$$\mathbf{N}^T(\mathbf{p} + tN) = c$$
$$\iff \mathbf{N}^T\mathbf{p} + t\mathbf{N}^T\mathbf{N} = c$$
$$\iff t = c - \mathbf{N}^T\mathbf{p}$$
$$\Rightarrow \mathbf{p} + t\mathbf{N} = \boxed{\mathbf{p} + (c - \mathbf{N}^T\mathbf{p})\mathbf{N}}$$

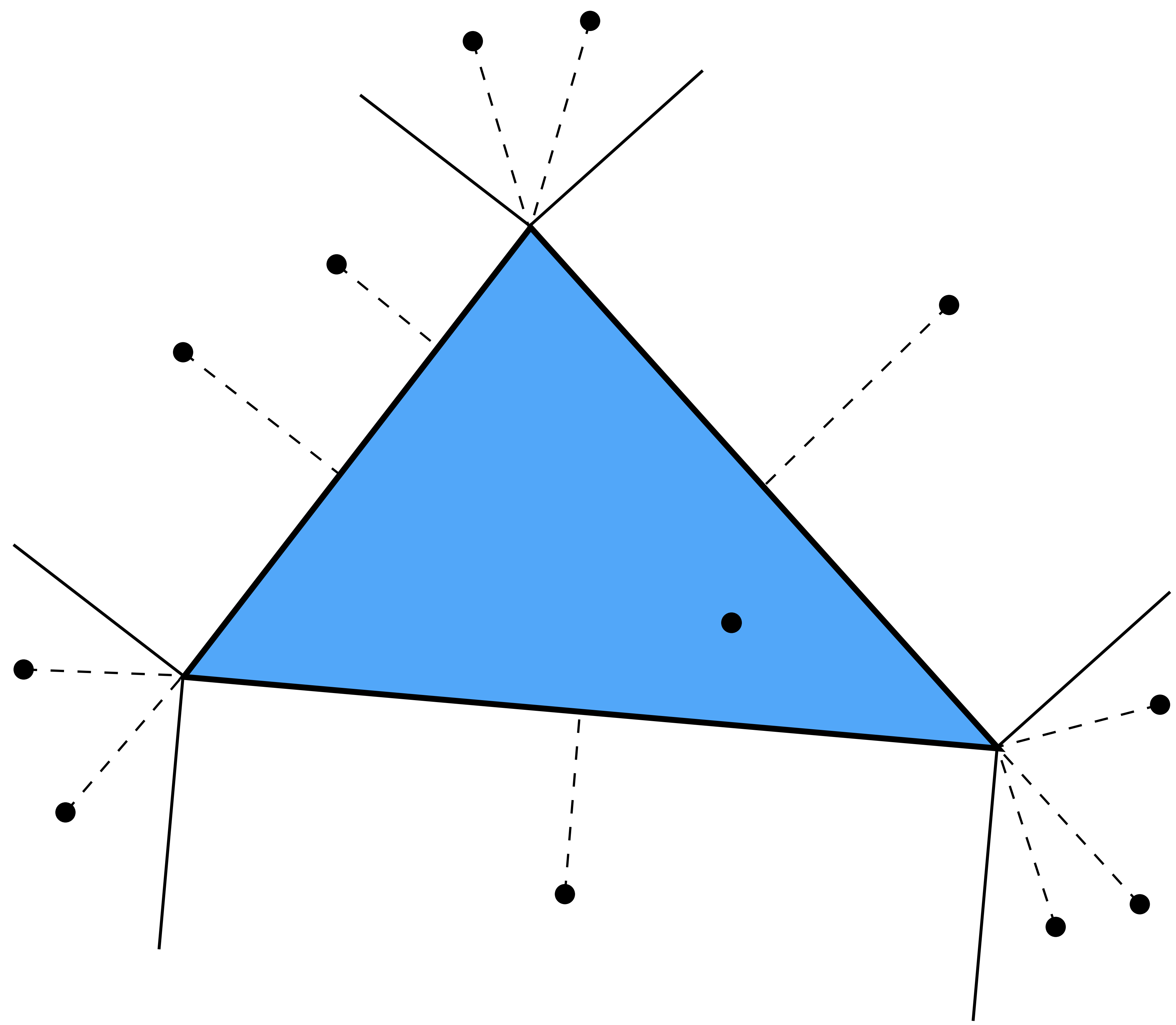
Harder: closest point on line segment

- Two cases: endpoint or interior
- Already have basic components:
  - point-to-point
  - point-to-line
- Algorithm?
  - find closest point on line
  - check if it's between endpoints
  - if not, take closest endpoint
- How do we know if it's between endpoints?
  - write closest point on line as a+t(b-a)
  - if t is between 0 and 1, it's inside the segment!



Even harder: closest point on triangle

- What are all the possibilities for the closest point?
- Almost just minimum distance to three segments:



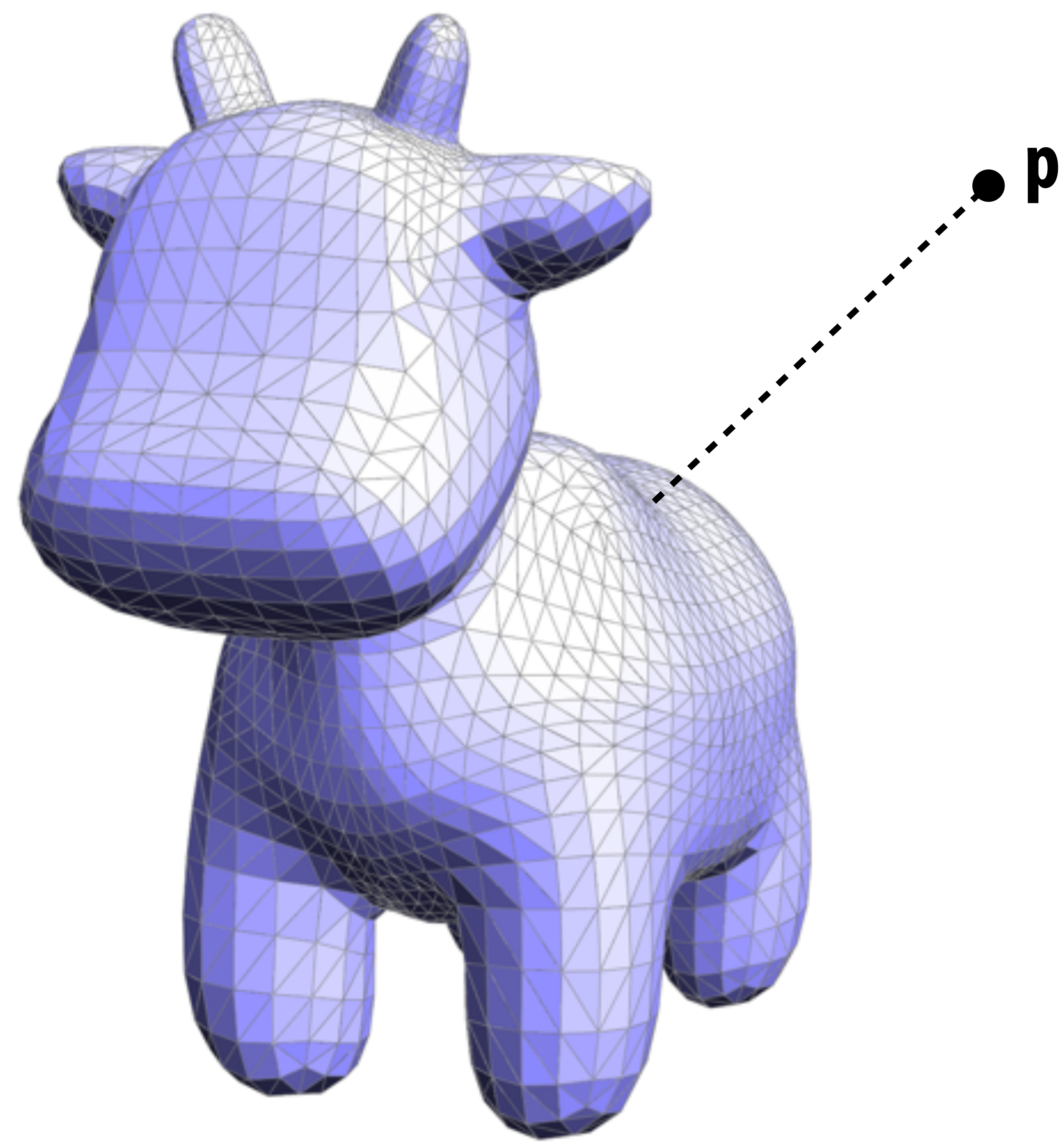
Q: What about a point inside the triangle?

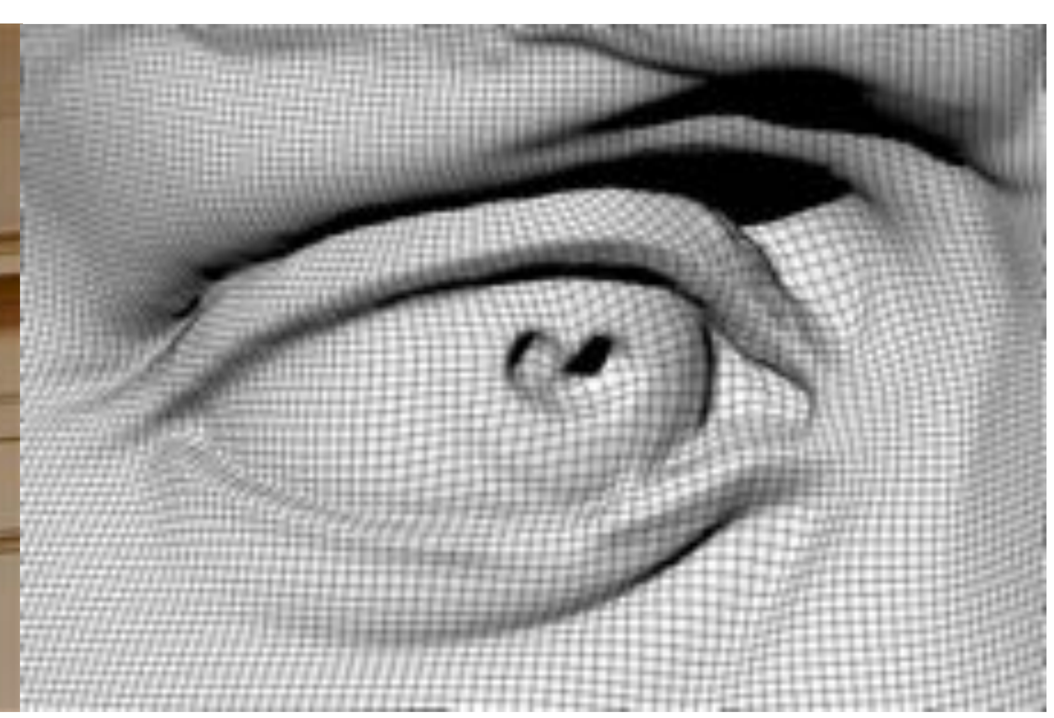
Closest point on triangle in 3D

- Not so different from 2D case
- Algorithm?
  - project onto plane of triangle
  - use half-space tests to classify point (vs. half plane)
  - if inside the triangle, we're done!
  - otherwise, find closest point on associated vertex or edge
- By the way, how do we find closest point on plane?
- Same expression as closest point on a line!
- E.g., $p + (c - N^T p) N$

Closest point on triangle mesh in 3D?

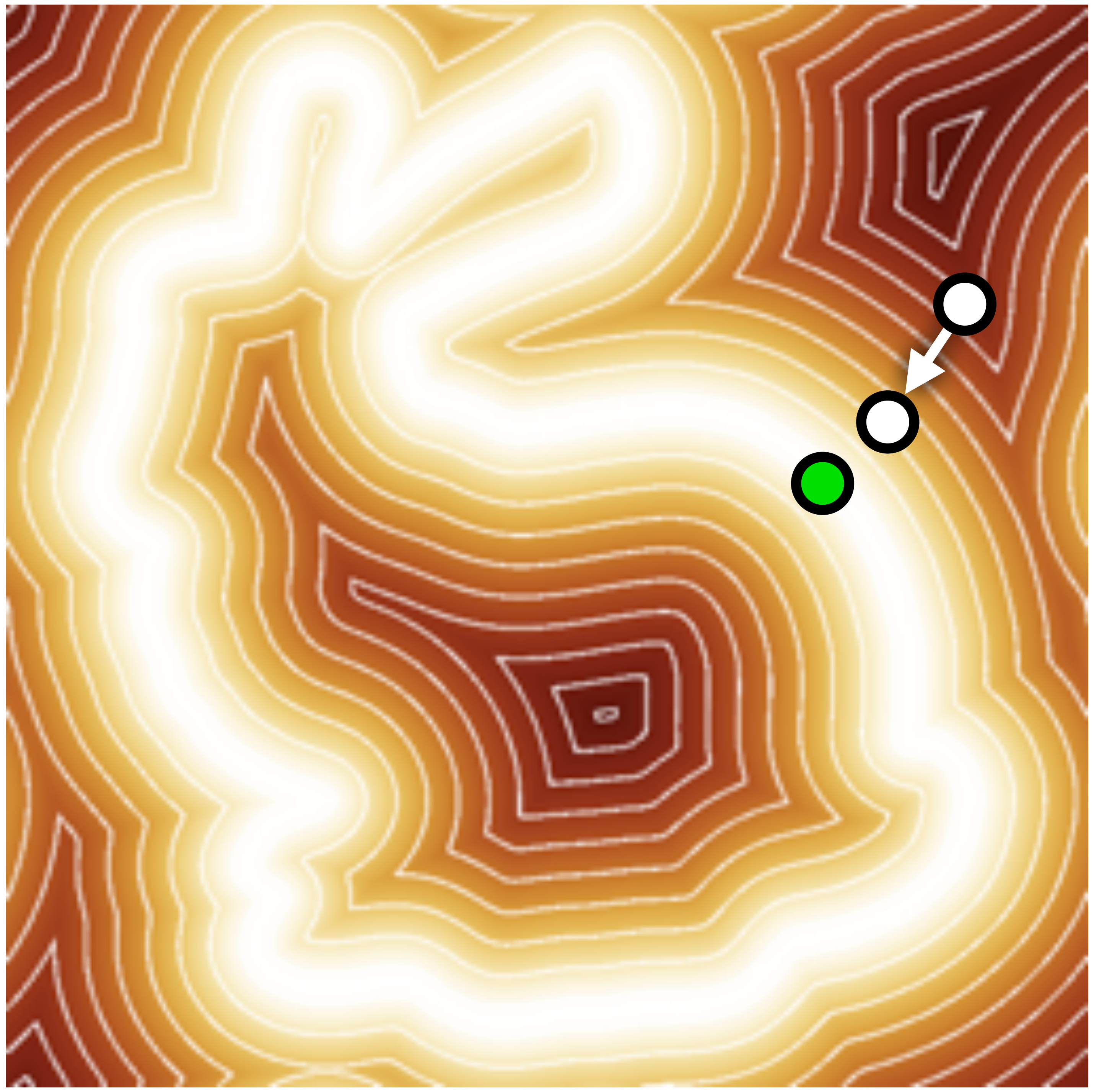
- Conceptually easy:
  - loop over all triangles
  - compute closest point to current triangle
  - keep globally closest point
- Q: What's the cost?
- What if we have billions of faces?
- NEXT TIME: Better data structures!





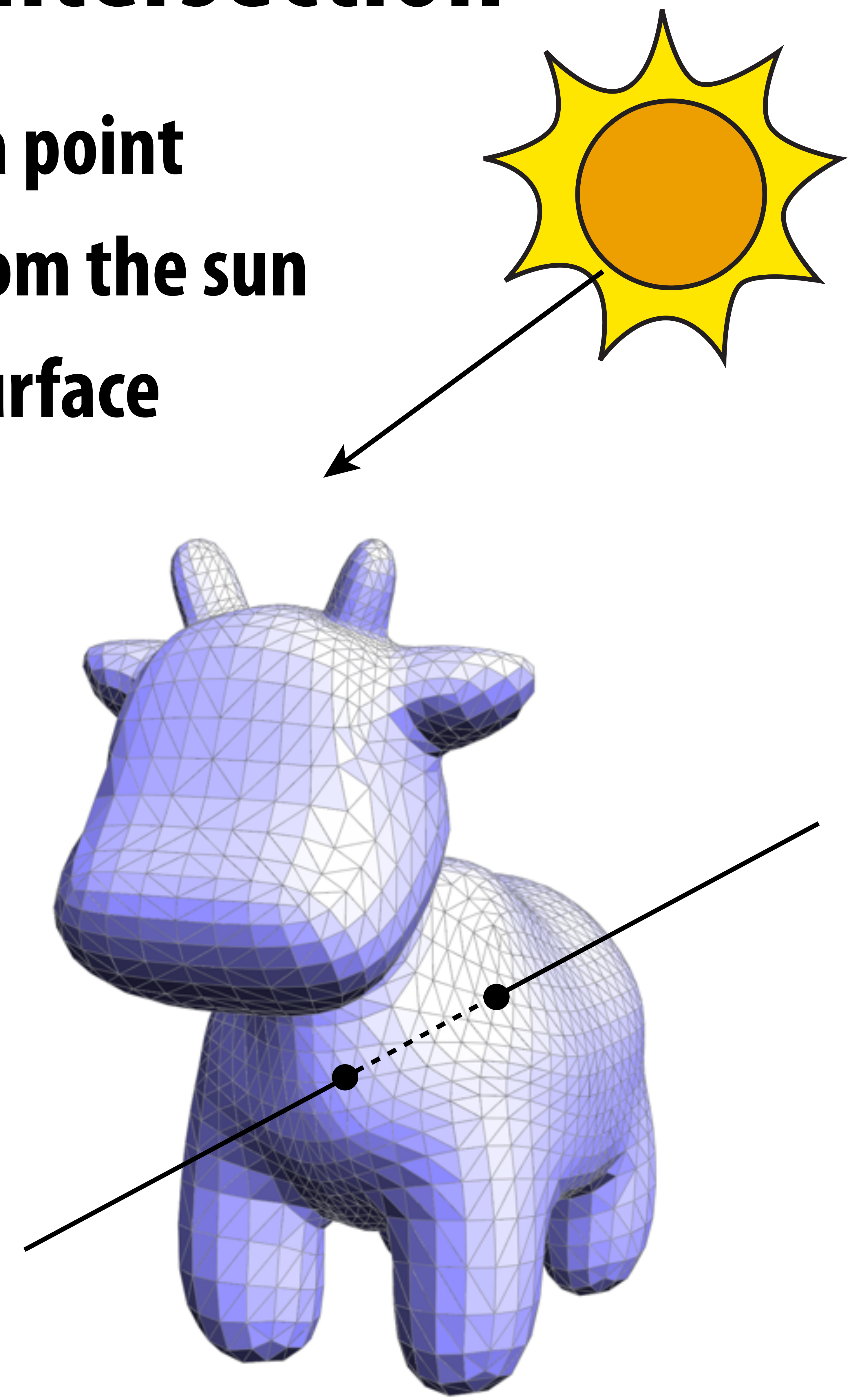
Closest point to implicit surface?

- If we change our representation of geometry, algorithms can change completely
- E.g., how might we compute the closest point on an implicit surface described via its distance function?
- One idea:
  - start at the query point
  - compute gradient of distance (using, e.g., finite differences)
  - take a little step (decrease distance)
  - repeat until we're at the surface (zero distance)
- Better yet: just store closest point for each grid cell! (speed/memory trade off)



Different query: ray-mesh intersection

- A "ray" is an oriented line starting at a point
- Think about a ray of light traveling from the sun
- Want to know where a ray pierces a surface
- Why?
  - GEOMETRY: inside-outside test
  - RENDERING: visibility, ray tracing
  - ANIMATION: collision detection
- Might pierce surface in many places!



Ray equation

- **Can express ray as**

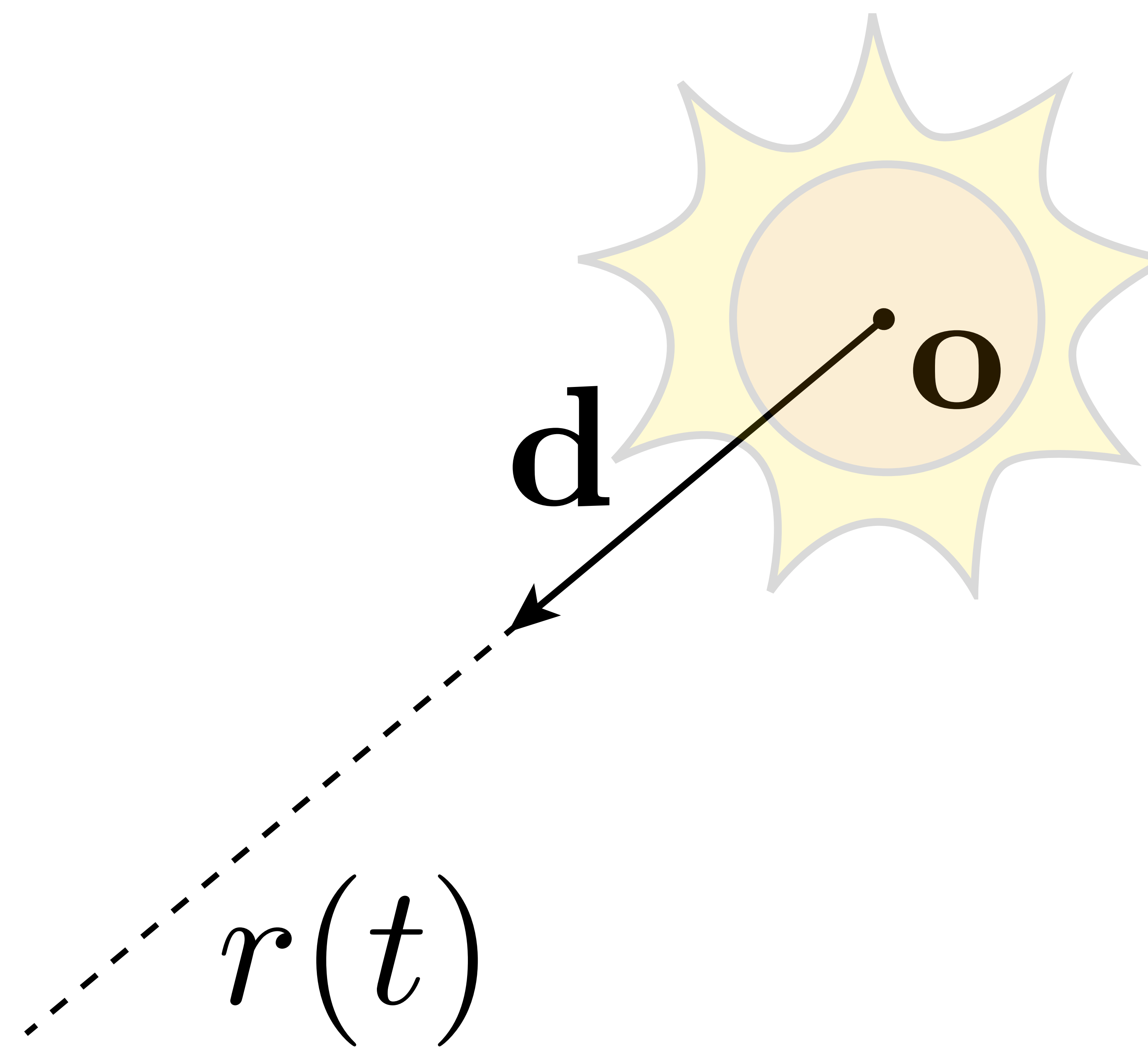
origin

unit direction

$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$$

point along ray

"time"



Intersecting a ray with an implicit surface

- **Recall implicit surfaces: all points x such that f(x) = 0**
- **Q: How do we find points where a ray pierces this surface?**
- **Well, we know all points along the ray: r(t) = o + td**
- **Idea: replace "x" with "r" in 1st equation, and solve for t**
- **Example: unit sphere**

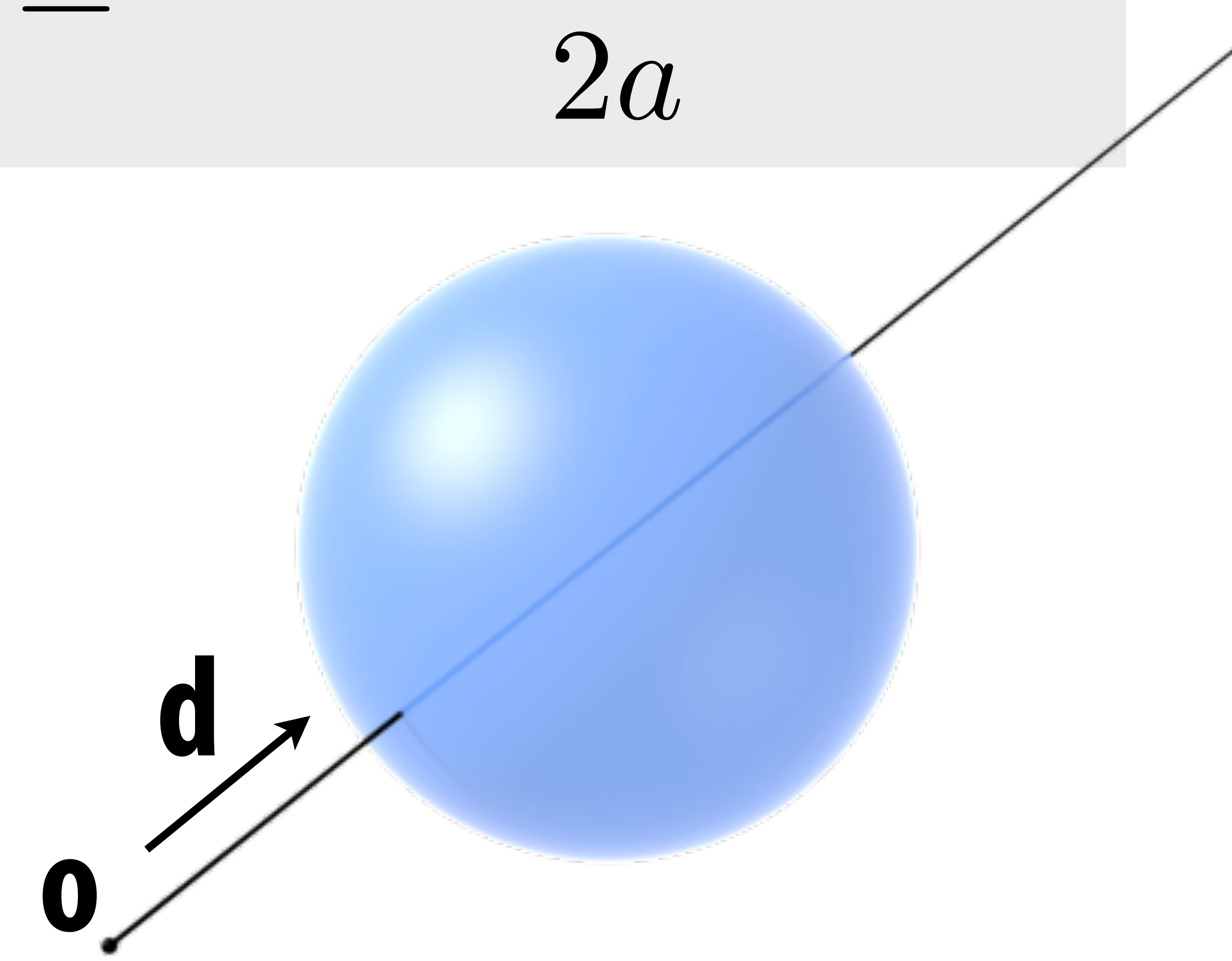
quadratic formula:

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$f(\mathbf{x}) = |\mathbf{x}|^2 - 1$$

$$\Rightarrow f(\mathbf{r}(t)) = |\mathbf{o} + t\mathbf{d}|^2 - 1$$

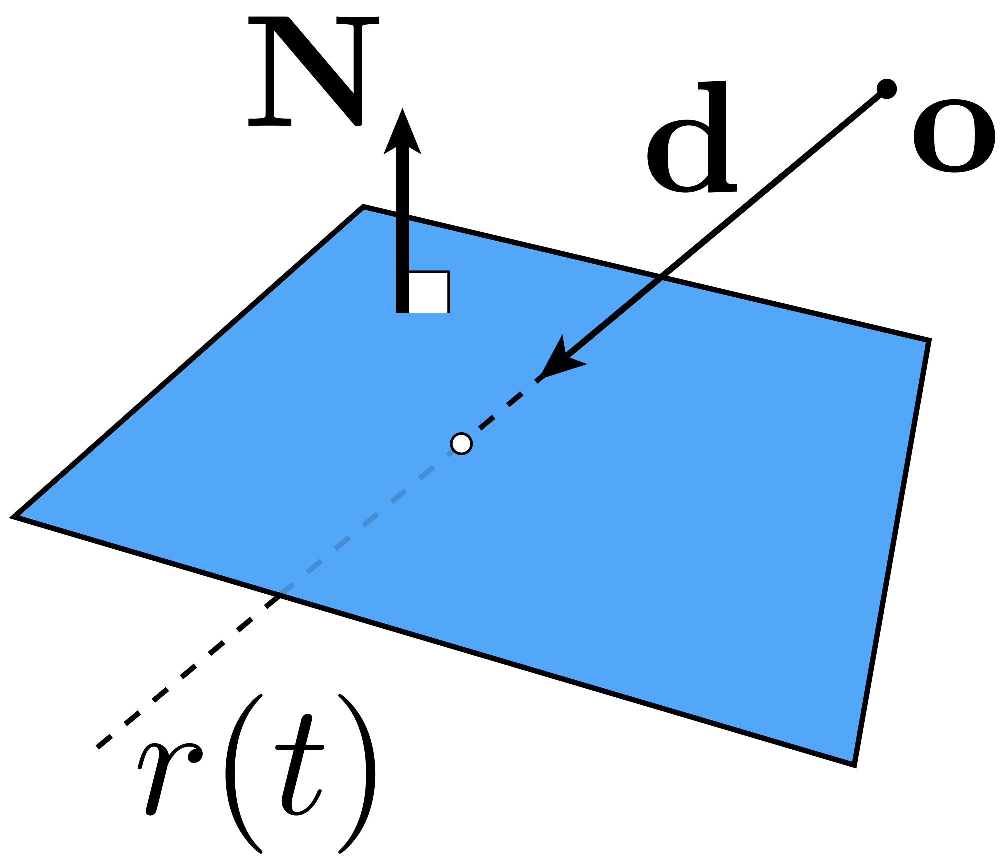
$$\underbrace{|\mathbf{d}|^2}_{a} t^2 + \underbrace{2(\mathbf{o} \cdot \mathbf{d})}_{b} t + \underbrace{|\mathbf{o}|^2 - 1}_{c} = 0$$



$$t = \boxed{-\mathbf{o} \cdot \mathbf{d} \pm \sqrt{(\mathbf{o} \cdot \mathbf{d})^2 - |\mathbf{o}|^2 + 1}}$$

Why two solutions?

Ray-plane intersection



- Suppose we have a plane $\mathbf{N}^\mathsf{T}\mathbf{x} = c$
  - N - unit normal
  - c - offset
- How do we find intersection with ray $r(t) = \mathbf{o} + t\mathbf{d}$?
- Key idea: again, replace the point x with the ray equation t:

$$\mathbf{N}^\mathsf{T}\mathbf{r}(t) = c$$

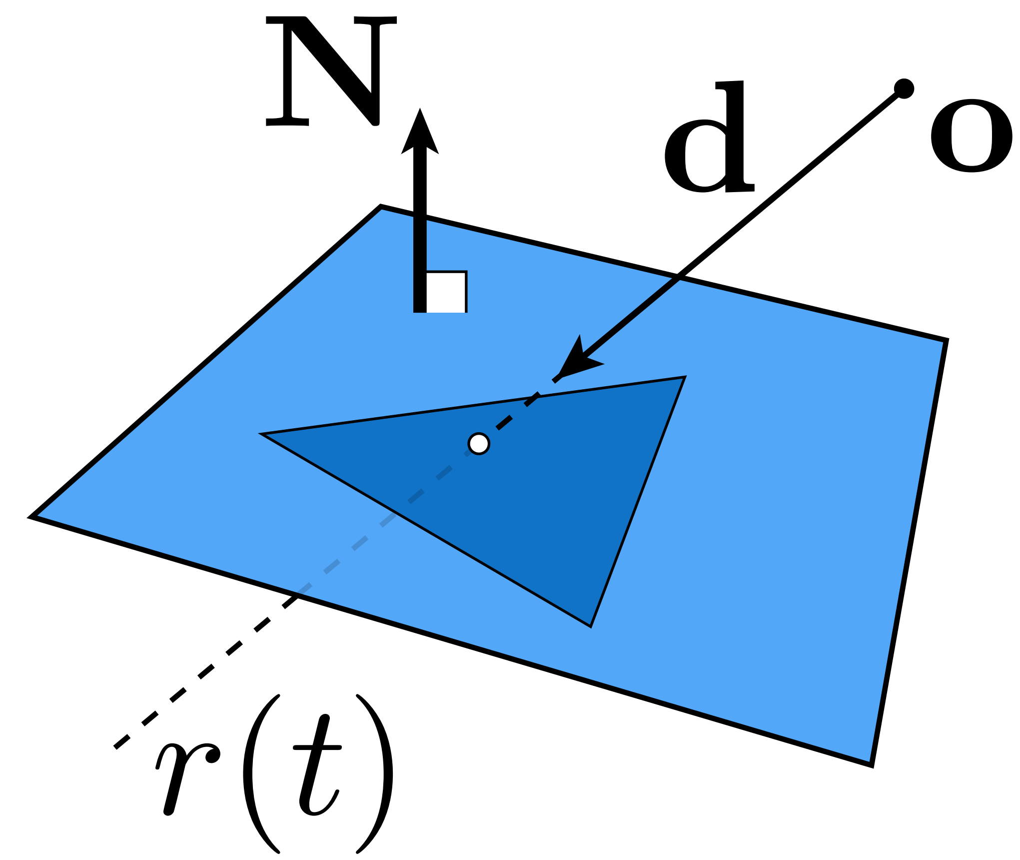
- Now solve for t:

$$\mathbf{N}^\mathsf{T}(\mathbf{o} + t\mathbf{d}) = c \qquad \Rightarrow t = \frac{c - \mathbf{N}^\mathsf{T}\mathbf{o}}{\mathbf{N}^\mathsf{T}\mathbf{d}}$$

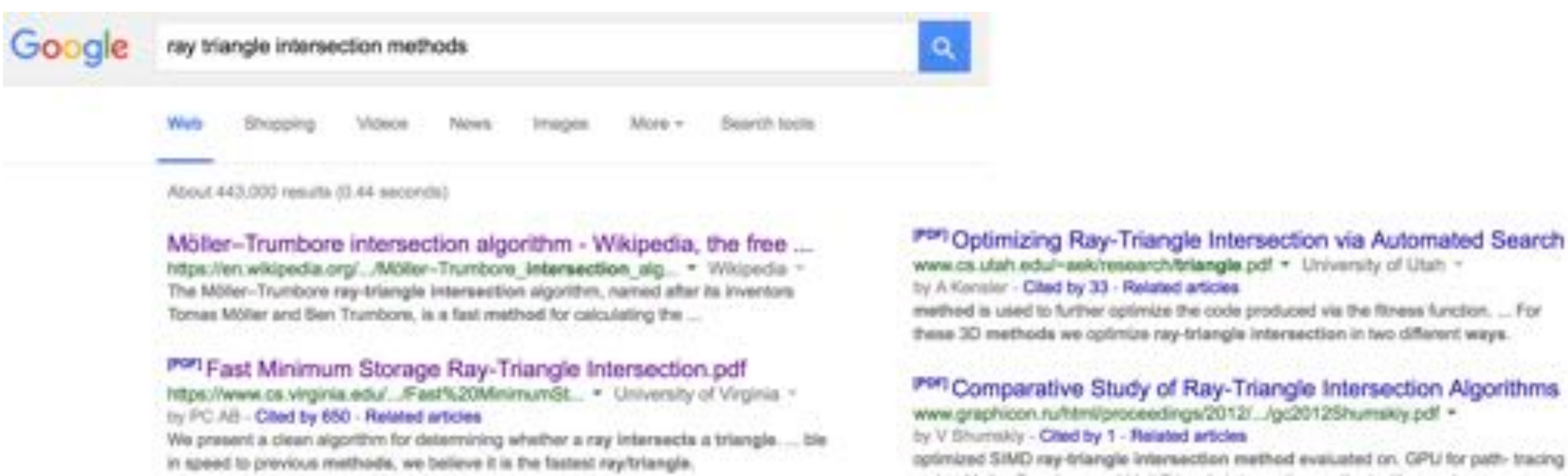
- And plug t back into ray equation:

$$r(t) = \mathbf{o} + \frac{c - \mathbf{N}^\mathsf{T}\mathbf{o}}{\mathbf{N}^\mathsf{T}\mathbf{d}}\mathbf{d}$$

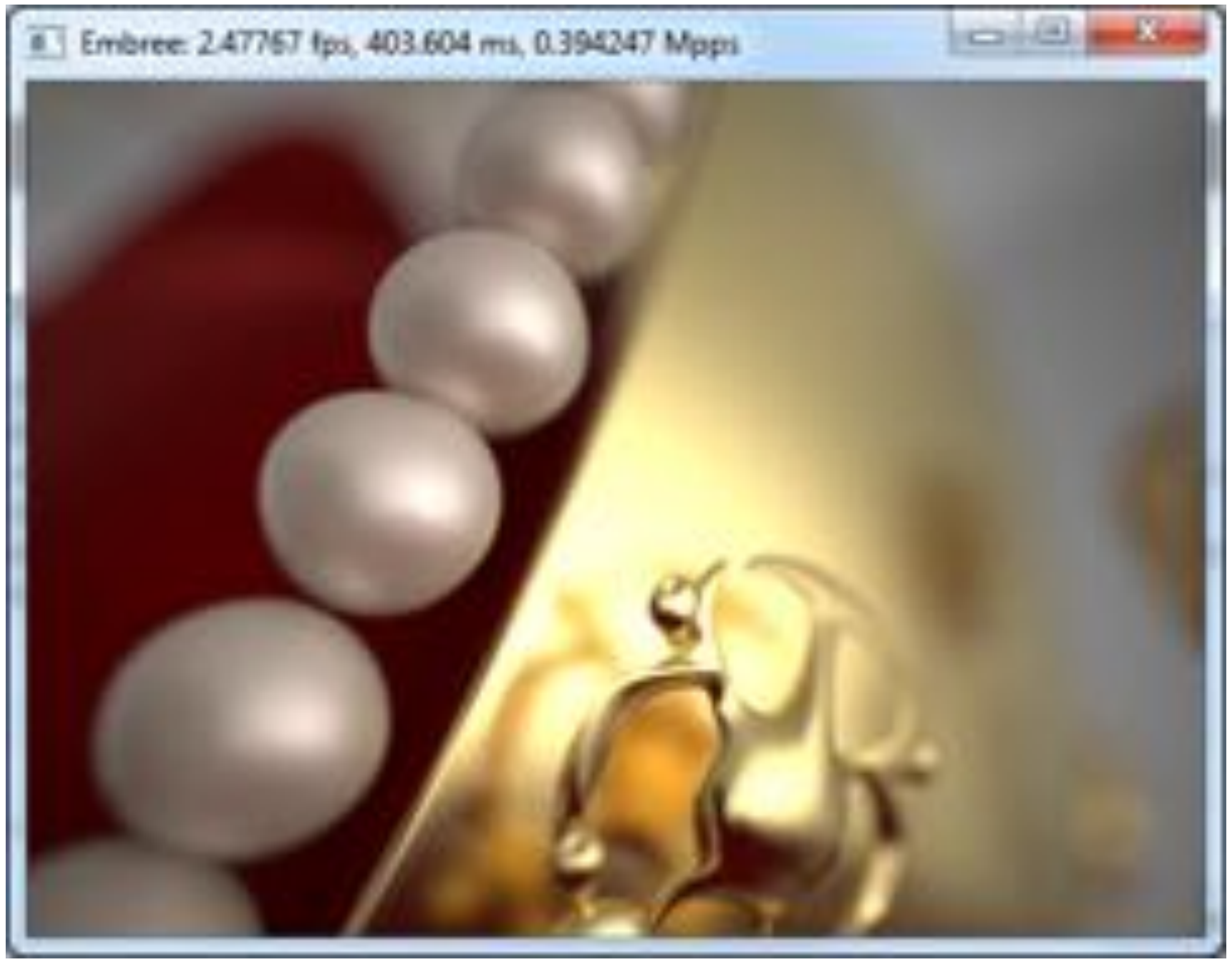
Ray-triangle intersection



- Triangle is in a plane...
- Not much more to say!
  - Compute ray-plane intersection
  - Q: What do we do now?
  - A: Why not compute barycentric coordinates of hit point?
  - If barycentric coordinates are all positive, point in triangle
- Actually, a lot more to say... if you care about performance!



Why care about performance?



Intel Embree

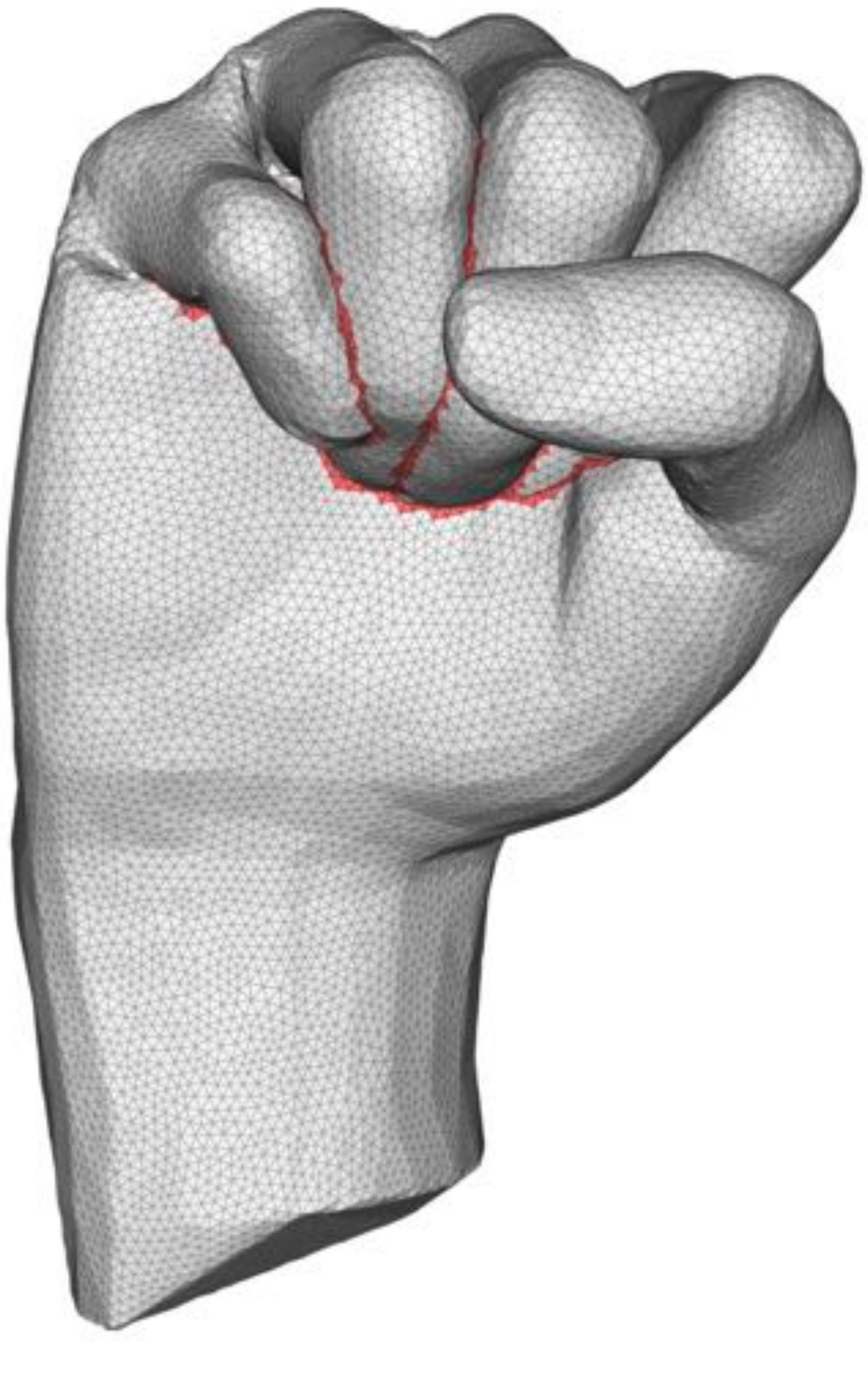
NVIDIA OptiX

Why care about performance?

“Brigade 3” real time path tracing demo

One more query: mesh-mesh intersection

- **GEOMETRY: How do we know if a mesh intersects itself?**
- **ANIMATION: How do we know if a collision occurred?**



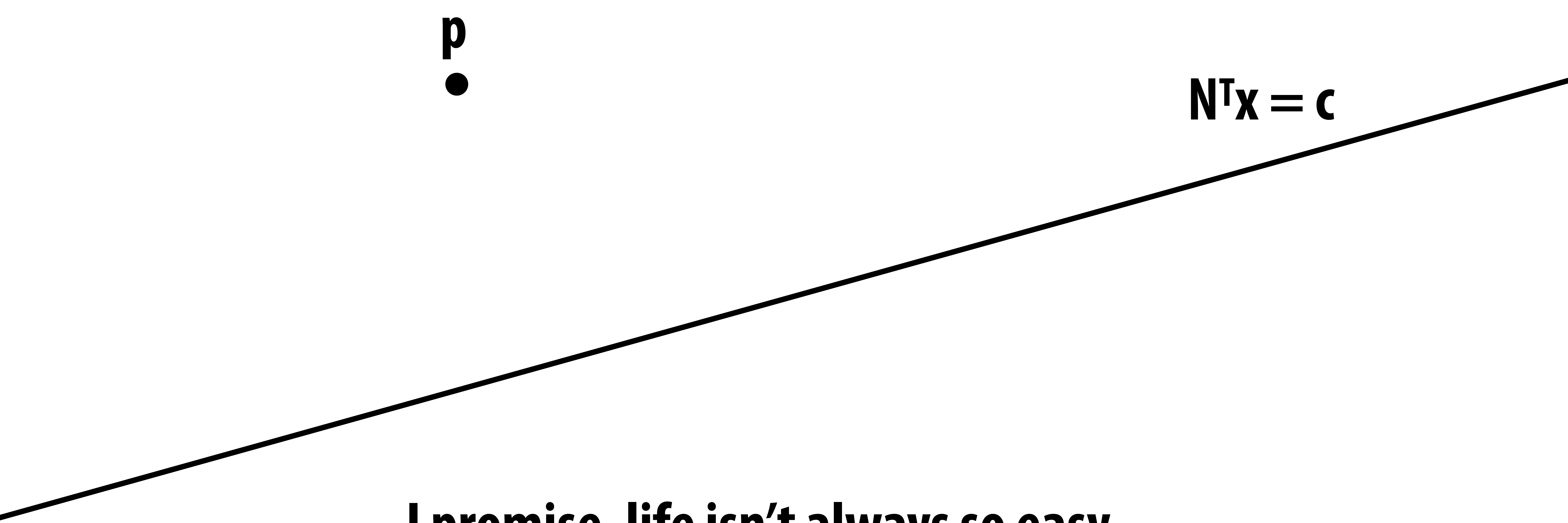
Warm up: point-point intersection

- Q: How do we know if p intersects a?
- A: ...check if they're the same point!

Sadly, life is not always so easy.

Slightly harder: point-line intersection

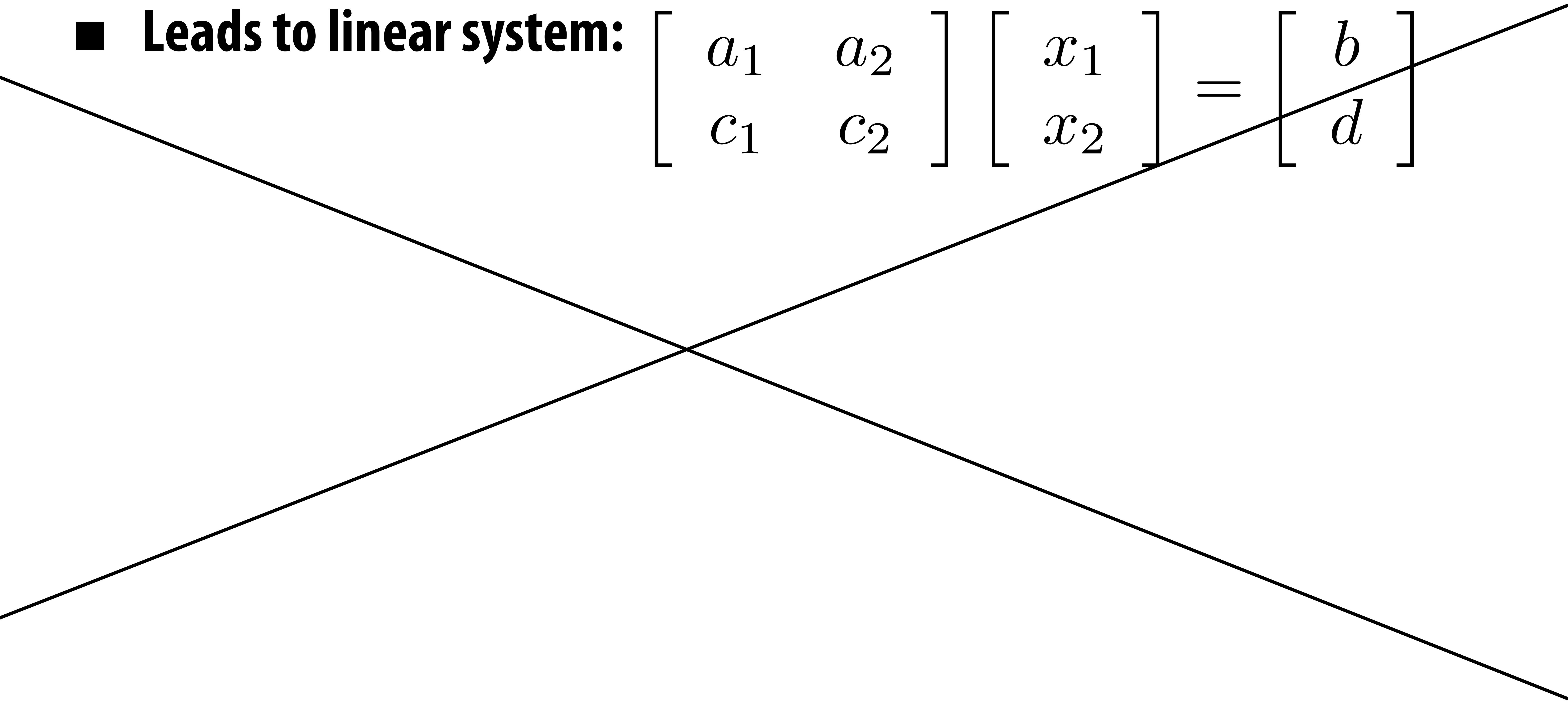
- Q: How do we know if a point intersects a given line?
- A: ...plug it into the line equation!



I promise, life isn't always so easy.

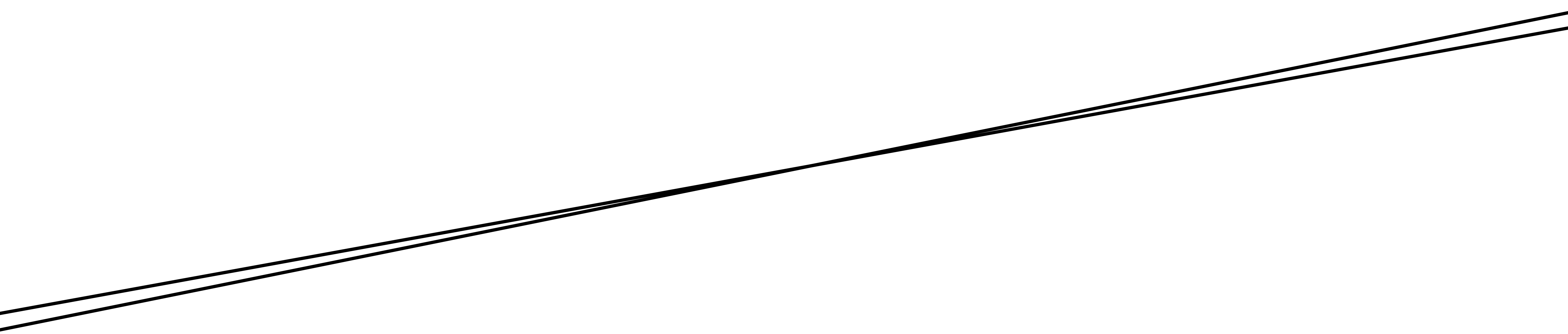
Finally interesting: line-line intersection

- **Two lines: ax=b and cx=d**
- **Q: How do we find the intersection?**
- **A: See if there is a simultaneous solution**
- **Leads to linear system:** $$\begin{bmatrix} a_1 & a_2 \\ c_1 & c_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}$$



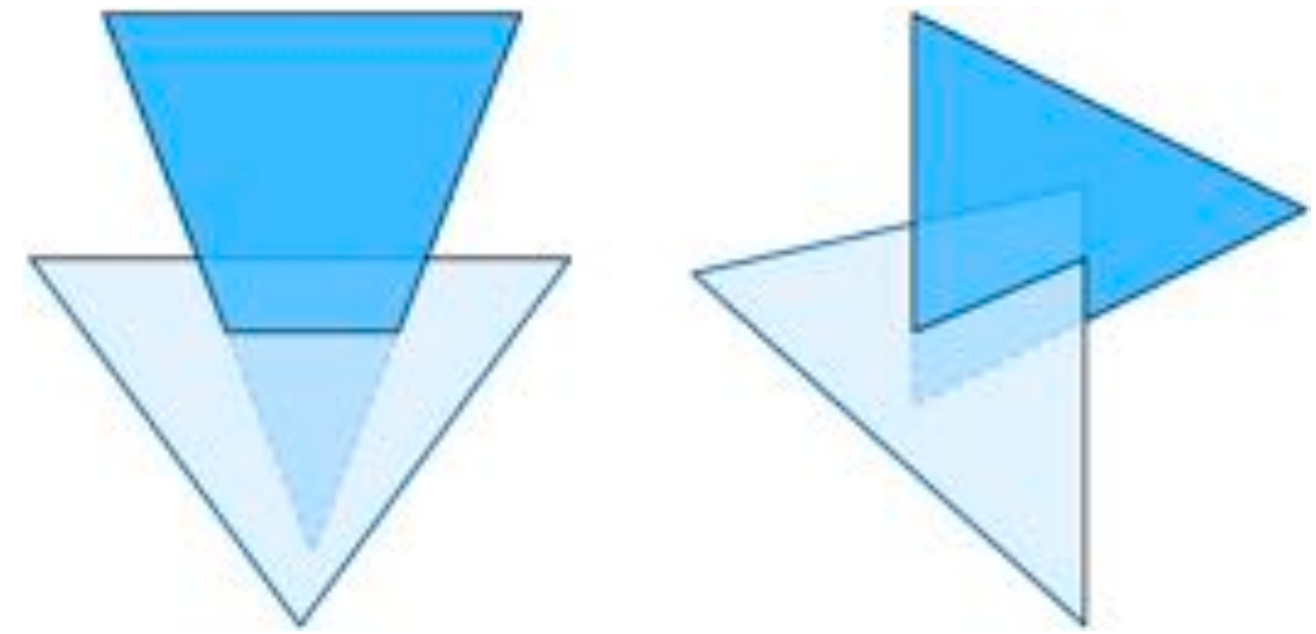
Degenerate line-line intersection?

- **What if lines are almost parallel?**
- **Small change in normal can lead to big change in intersection!**
- **Instability very common, very important with geometric predicates. Demands special care (e.g., analysis of matrix).**

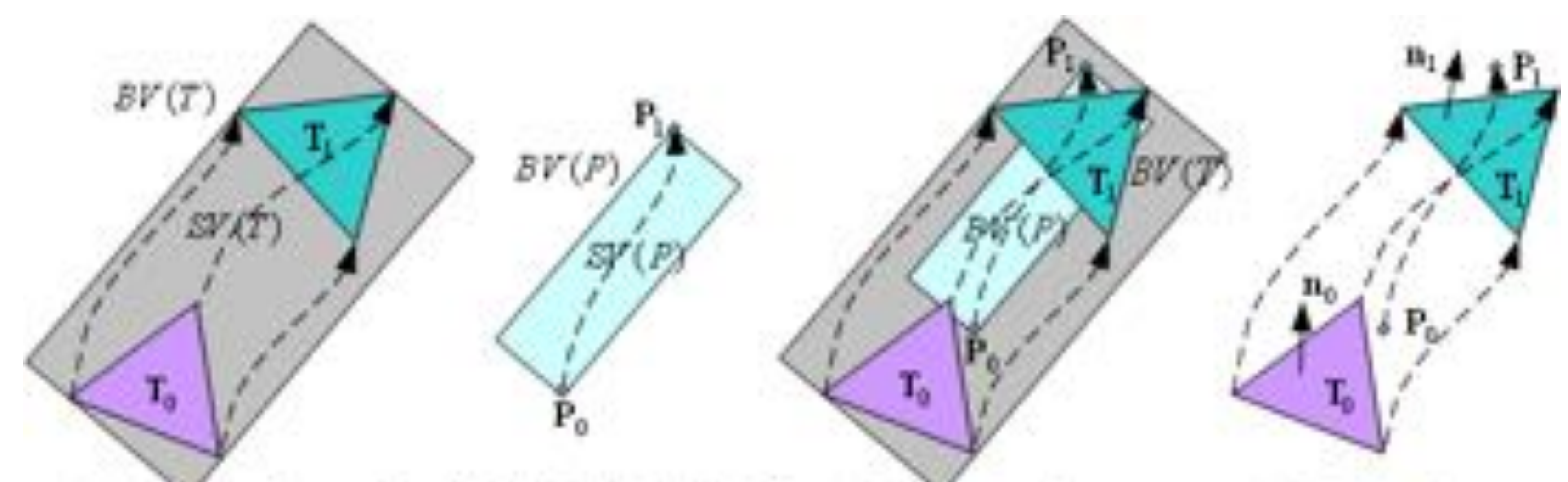


See for example Shewchuk, "*Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates*"

Triangle-Triangle Intersection?



- Lots of ways to do it
- Basic idea:
  - Q: Any ideas?
  - One way: reduce to edge-triangle intersection
  - Check if each line passes through plane
  - Then do interval test
- What if triangle is moving?
  - Important case for animation
  - Can think of triangles as prisms in time
  - Turns dynamic problem (nD + time) into purely geometric problem in (n+1)-dimensions



(a) Bounding volume of a deforming triangle (b) Bounding volume of a deforming vertex (c) Bounding volume test (d) Coplanarity test

Up Next: Spatial Acceleration Data Strucutres

- Testing every element is slow!
- E.g., linearly scanning through a list vs. binary search
- Can apply this same kind of thinking to geometric queries