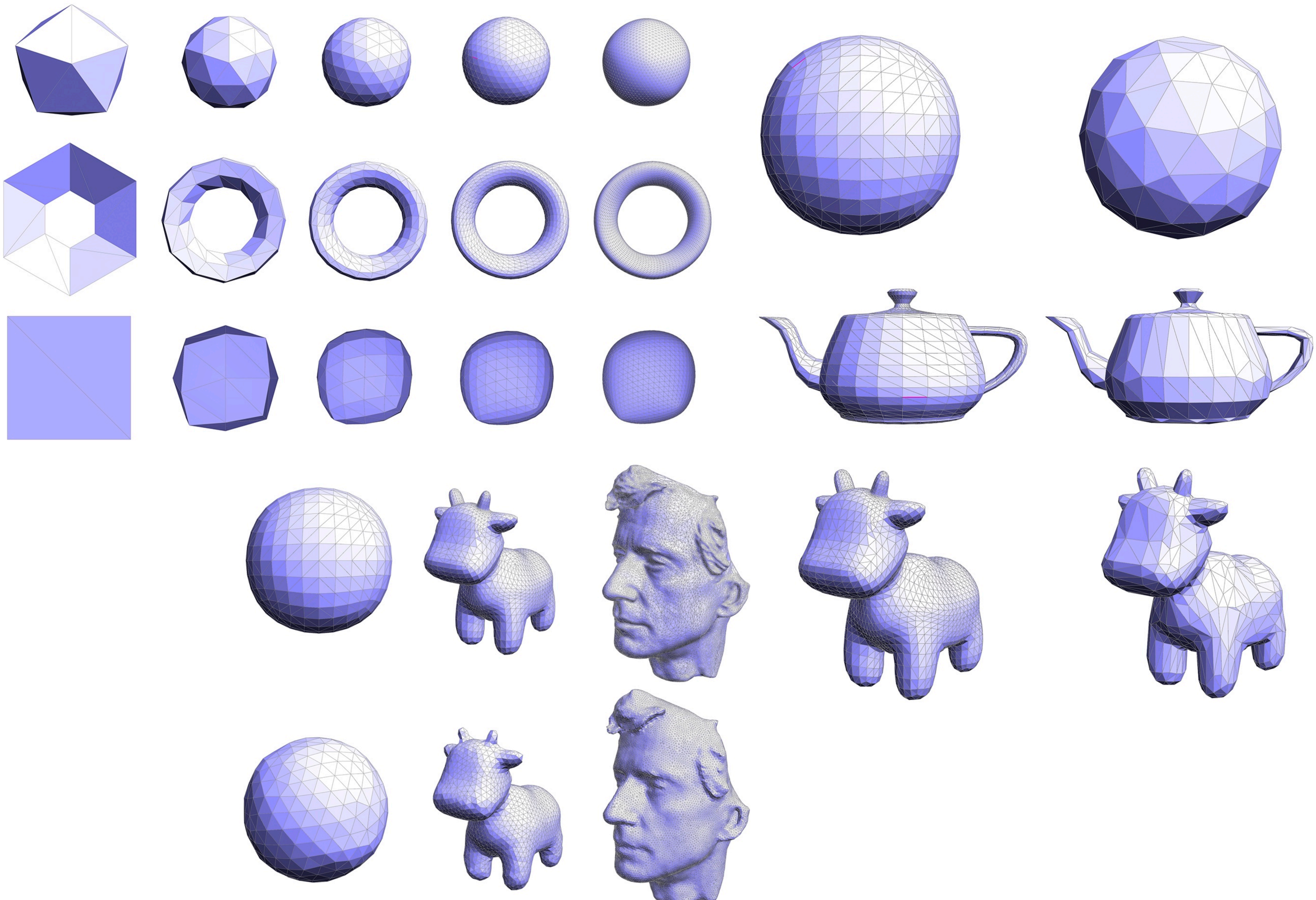
Lecture 9:

Geometric Queries

Computer Graphics

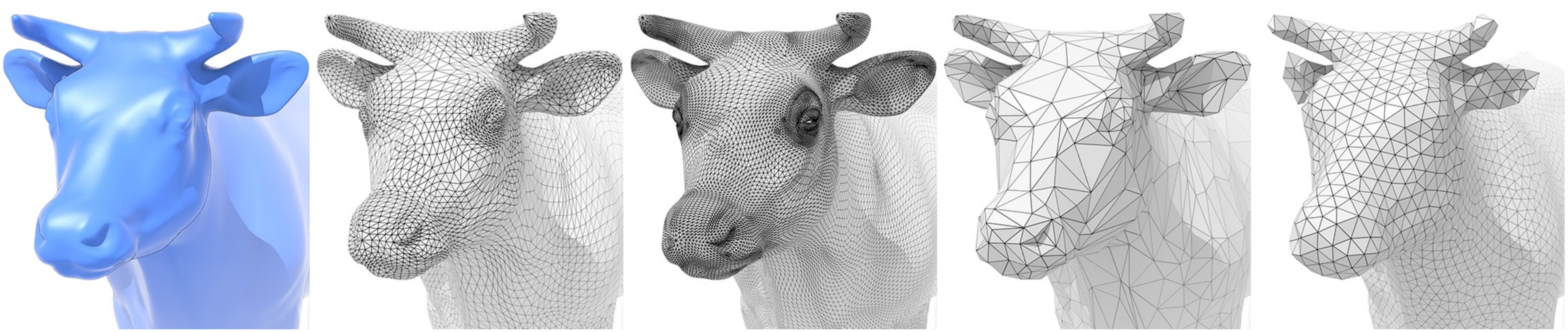
CMU 15-462/15-662, Fall 2015

Assignment 2, Part II is out!



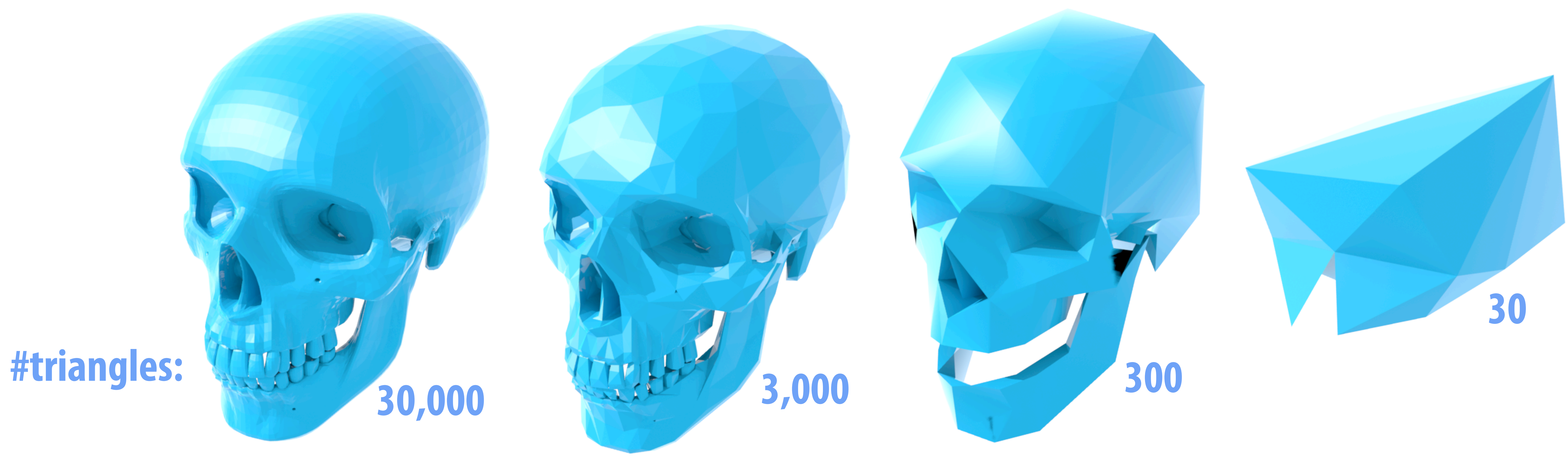
Last time: Geometry Processing

- Extend signal processing to curved shapes
  - encounter familiar issues (sampling, aliasing, etc.)
  - some new challenges (irregular sampling, no FFT, etc.)
- Focused on resampling triangle meshes
  - local: edge flip, split, collapse
  - global: subdivision, quadric error, isotropic remeshing
- Today: what kind of geometric queries *can't* we answer yet?



Simplification via Quadric Error Metric

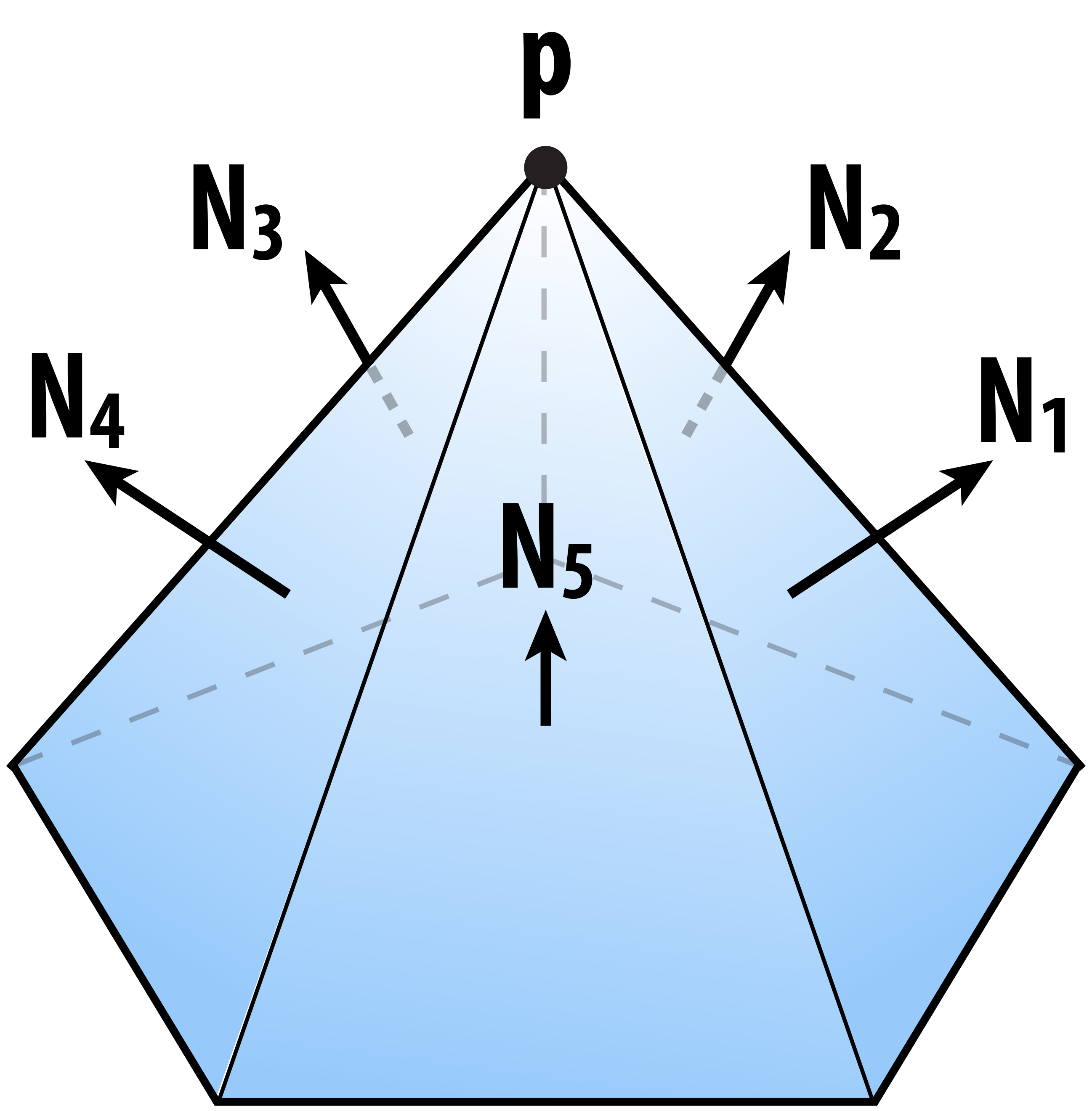
- One popular scheme: iteratively collapse edges
- Which edges? Assign score with *quadric error metric**
  - approximate distance to surface as sum of distance to aggregated triangles
  - iteratively collapse edge with smallest score
  - greedy algorithm... great results!

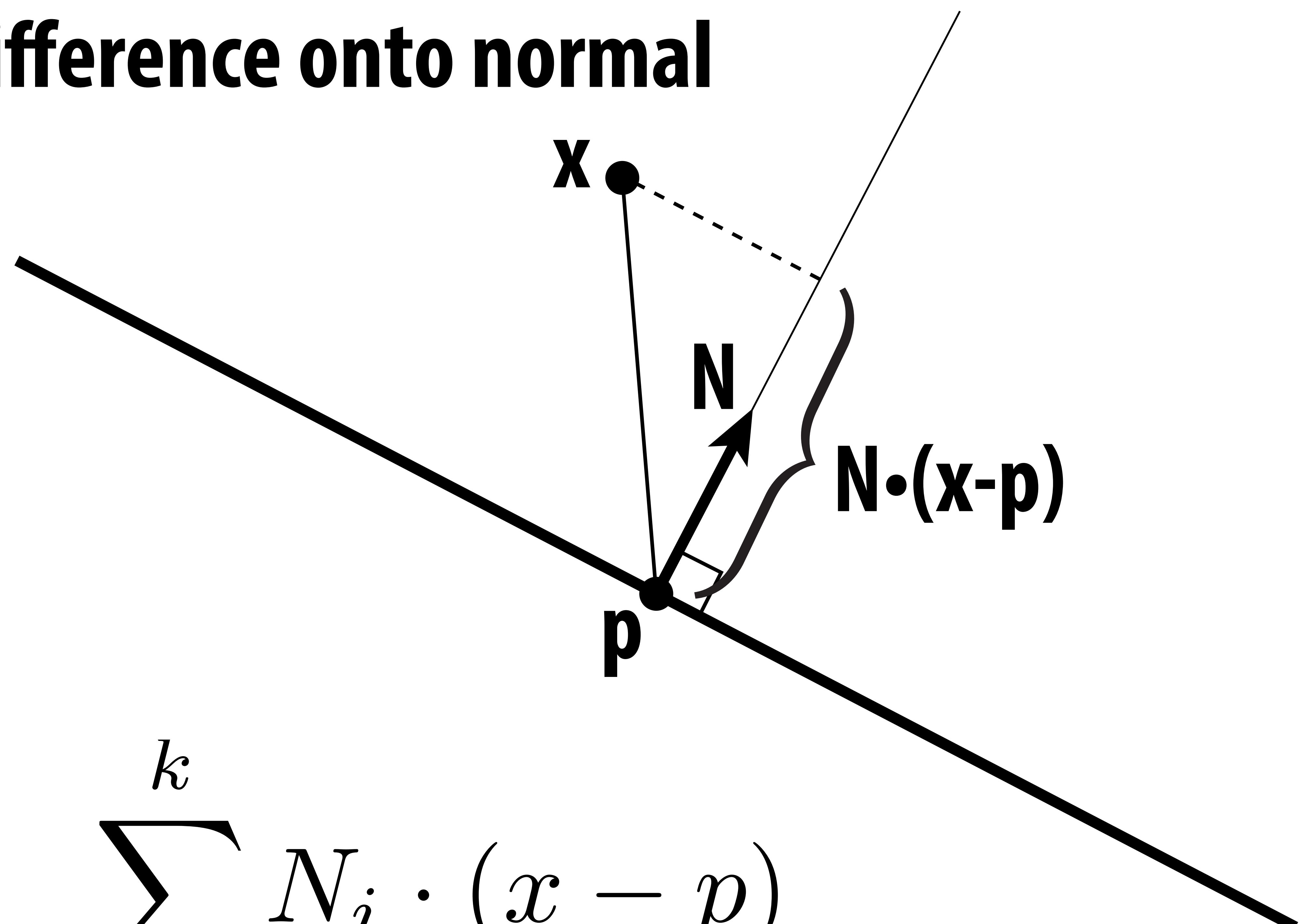


*invented here at CMU! (Garland & Heckbert 1997)

Quadric Error Metric

- Approximate distance to a collection of triangles
- Distance is sum of point-to-plane distances
  - Q: Distance to plane w/ normal N passing through point p?
  - A: N•(x-p), i.e., project difference onto normal
- Sum of distances:





$$\sum_{i=1}^{k} N_i \cdot (x - p)$$

Quadric Error - Homogeneous Coordinates

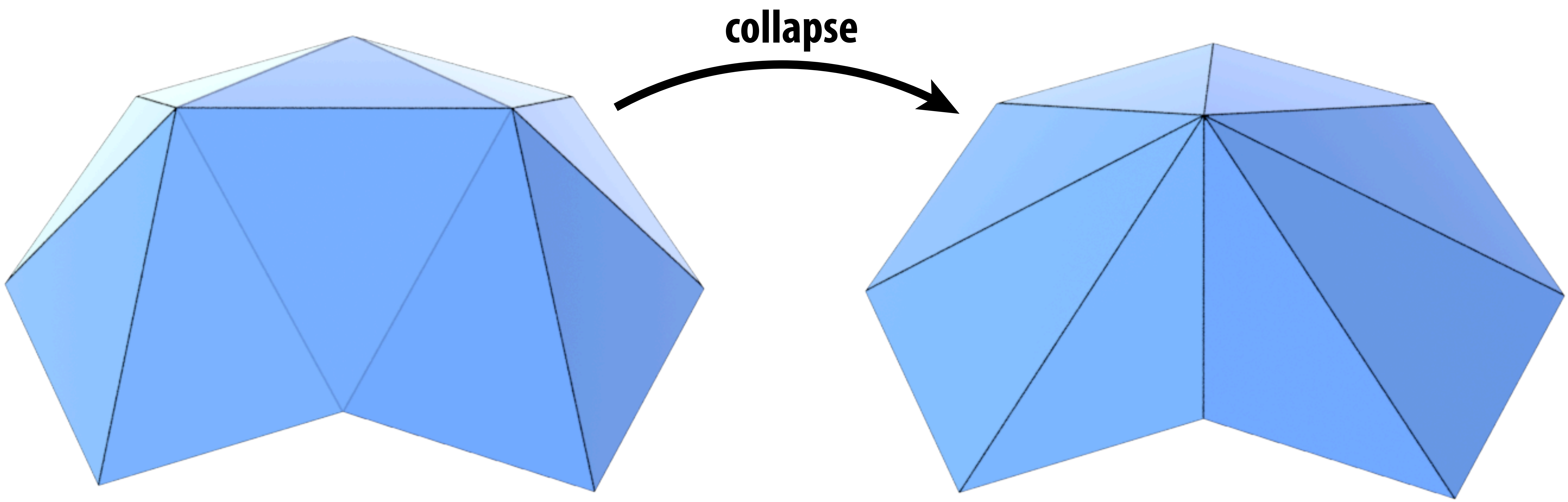
- **Suppose in coordinates we have**
  - **a query point (x,y,z)**
  - **a normal (a,b,c)**
  - **an offset d := -(p,q,r) • (a,b,c)**

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

- **Then in homogeneous coordinates, let**
  - **u := (x,y,z,1)**
  - **v := (a,b,c,d)**
- ***Signed* distance to plane is then just u•v = ax+by+cz+d**
- ***Squared* distance is $(u^Tv)^2 = u^T(vv^T)u =: u^TKu$**
- **Key idea: *matrix K encodes distance to plane***
- **K is symmetric, contains 10 unique coefficients (small storage)**

Quadric Error of Edge Collapse

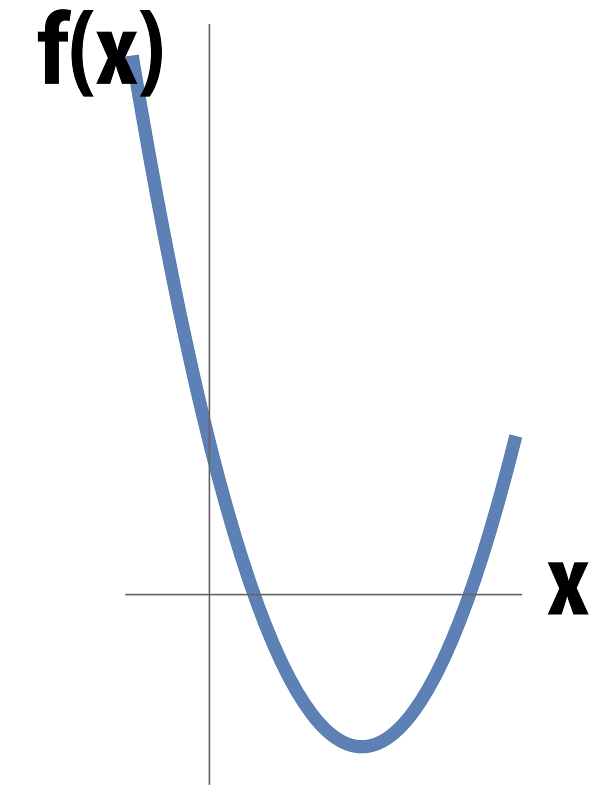
- **How much does it cost to collapse an edge?**
- **Idea: compute edge midpoint, measure quadric error**



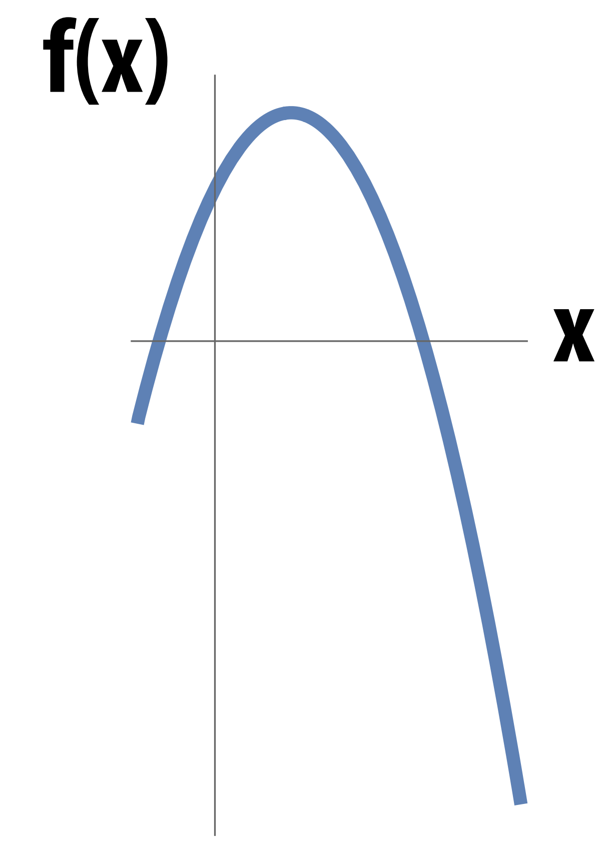
- **Better idea: use point that *minimizes quadric error* as new point!**
- **Q: How do we minimize quadric error?**

Review: Minimizing a Quadratic Function

- Suppose I give you 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: Look for the point where the function isn't changing (if we look "up close")
- I.e., find the point where the *derivative* vanishes

$$f'(x) = 0$$

$$2ax + b = 0$$

$$x = -b/2a$$

(What about our second example?)

Minimizing a Quadratic Form

- A *quadratic form* is just a generalization of our quadratic polynomial from 1D to nD
- E.g., in 2D: $f(x,y) = ax^2 + bxy + cy^2 + dx + ey + g$
- Can always (always!) write quadratic polynomial using a *symmetric* matrix (and a vector, and a constant):

$$f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a & b/2 \\ b/2 & c \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d & e \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + g$$

$$= \mathbf{x}^{\mathsf{T}} A \mathbf{x} + \mathbf{u}^{\mathsf{T}} x + g$$

(this expression works for *any* n!)

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

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

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

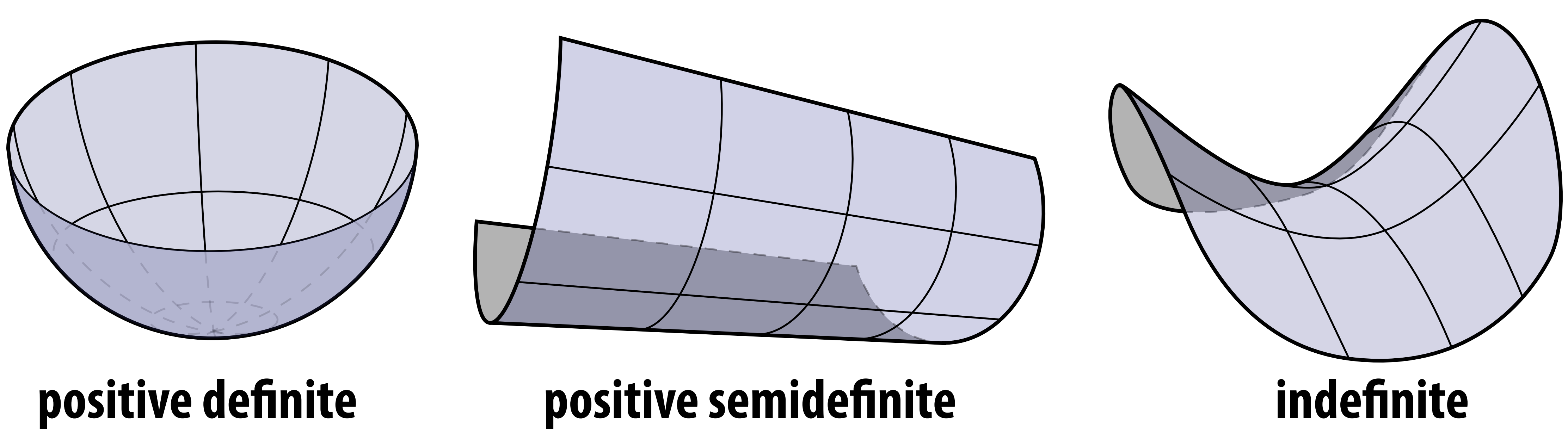
(Can you show this is true, at least in 2D?)

Positive Definite Quadratic Form

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

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

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



- Positive-definiteness is *extremely important* in computer graphics: it means we can find a minimum by solving linear equations. Basis of many, many modern algorithms (geometry processing, simulation, ...).

Minimizing Quadratic Error

- **Find "best" point for edge collapse by minimizing quad. form**

$$\min_u \mathbf{u}^\mathsf{T} K \mathbf{u}$$

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

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

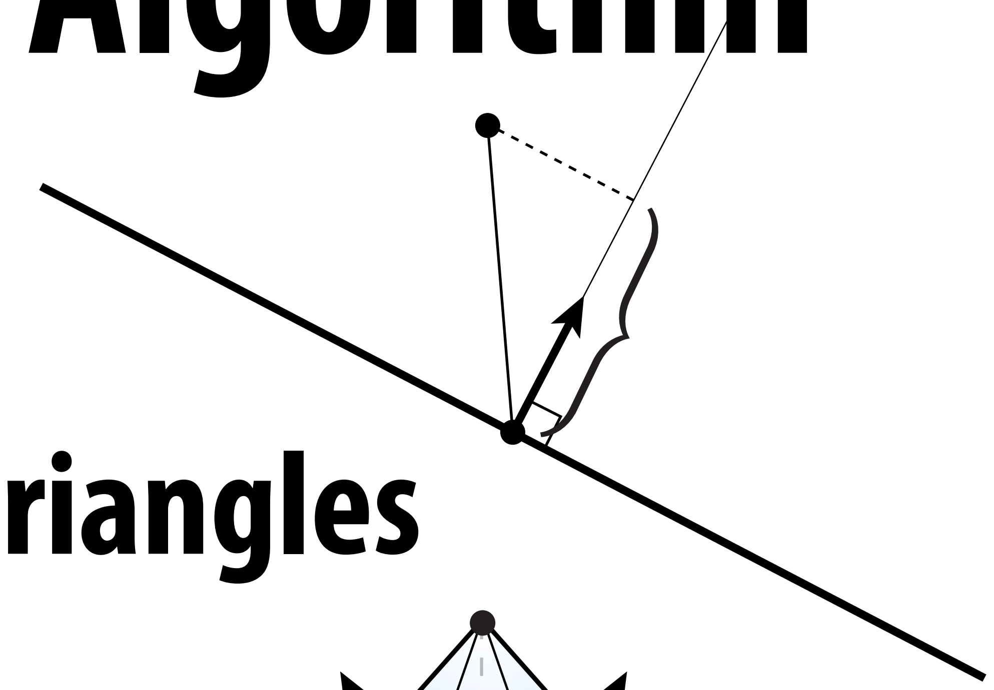
$$= \mathbf{x}^\mathsf{T} B \mathbf{x} + 2\mathbf{w}^\mathsf{T}\mathbf{x} + d^2$$

- **Now we have a quadratic form in the 3D position x.**
- **Can minimize as before:**

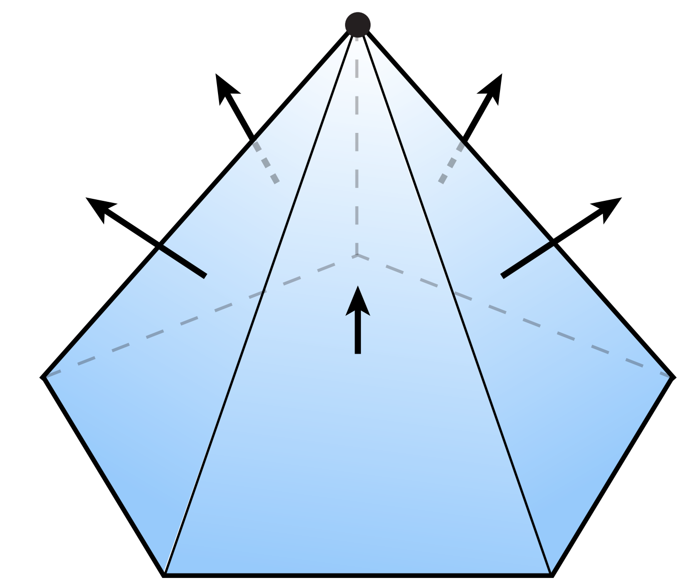
$$2B\mathbf{x} + 2\mathbf{w} = 0 \qquad \Longleftrightarrow \qquad \mathbf{x} = -B^{-1}\mathbf{w}$$

(Q: Why should B be positive-definite?)

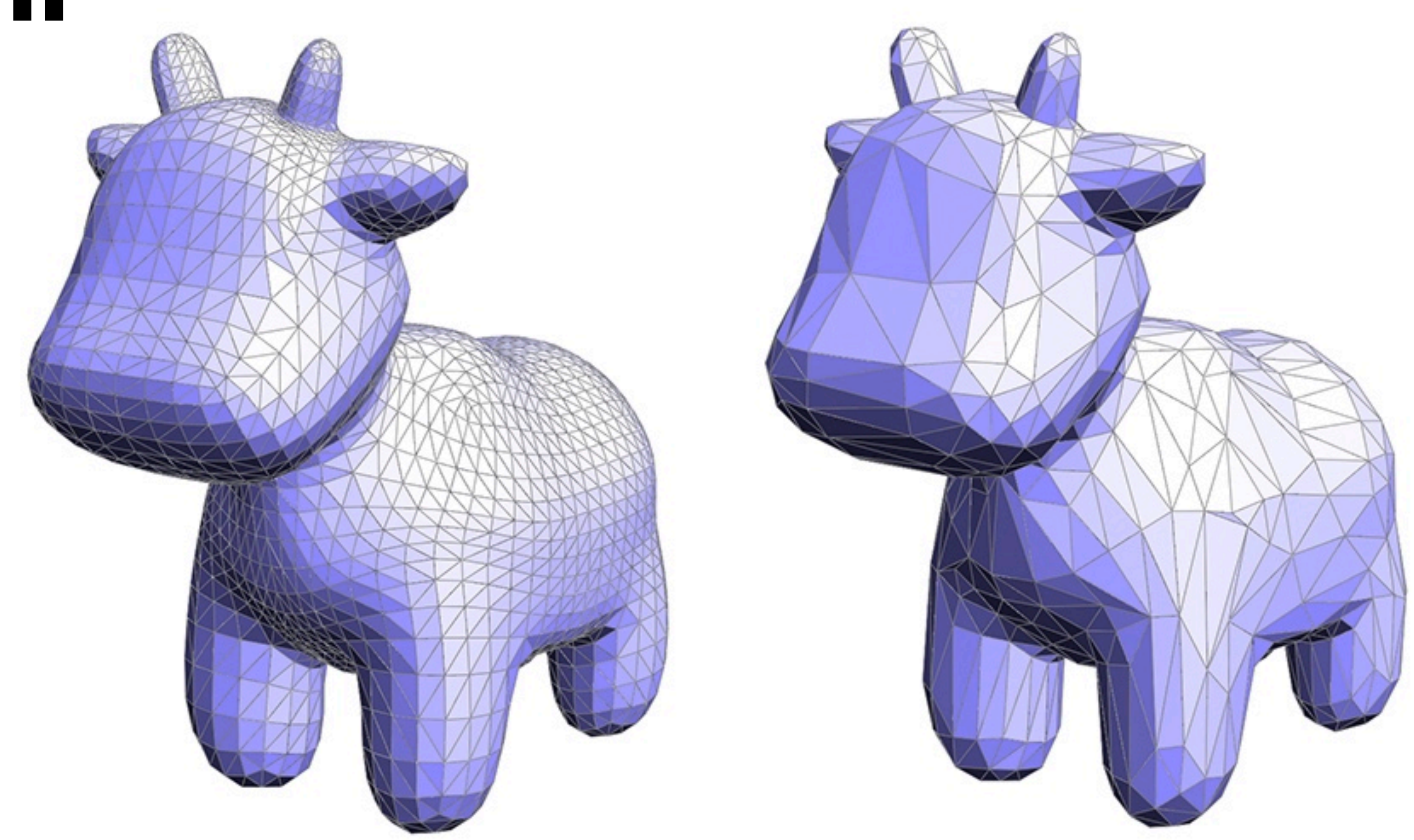
Quadric Error Simplification: Final Algorithm



- Compute K for each triangle (distance to plane)
- Set K at each vertex to sum of Ks from incident triangles
- Set K at each edge to sum of Ks at endpoints
- Find point at each edge minimizing quadric error



- Until we reach target # of triangles:
  - collapse edge (i,j) with smallest cost to get new vertex m
  - add K_i and K_j to get quadric K_m at m
  - update cost of edges touching m
- *More details in assignment writeup!*

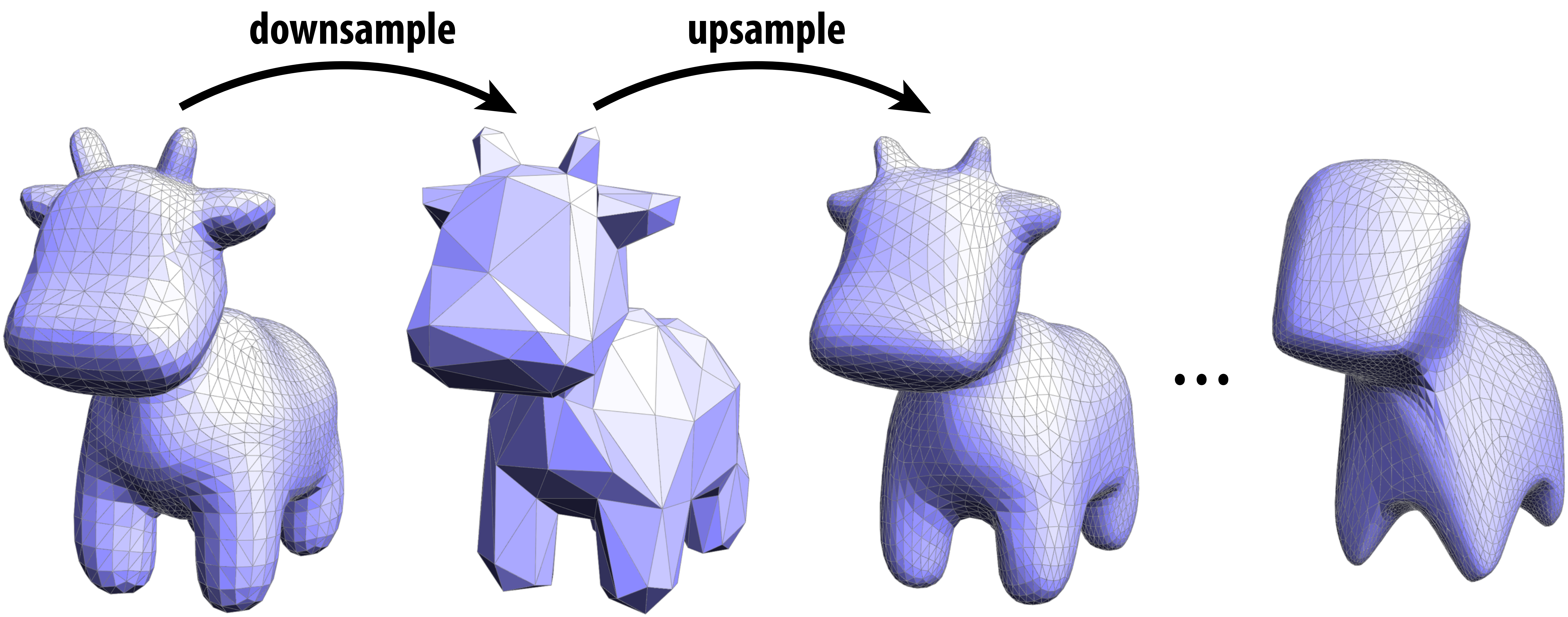


Isotropic Remeshing

- **Conceptually much simpler algorithm**
- ***More detail in the assignment writeup!***

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

Demo: Danger of Resampling

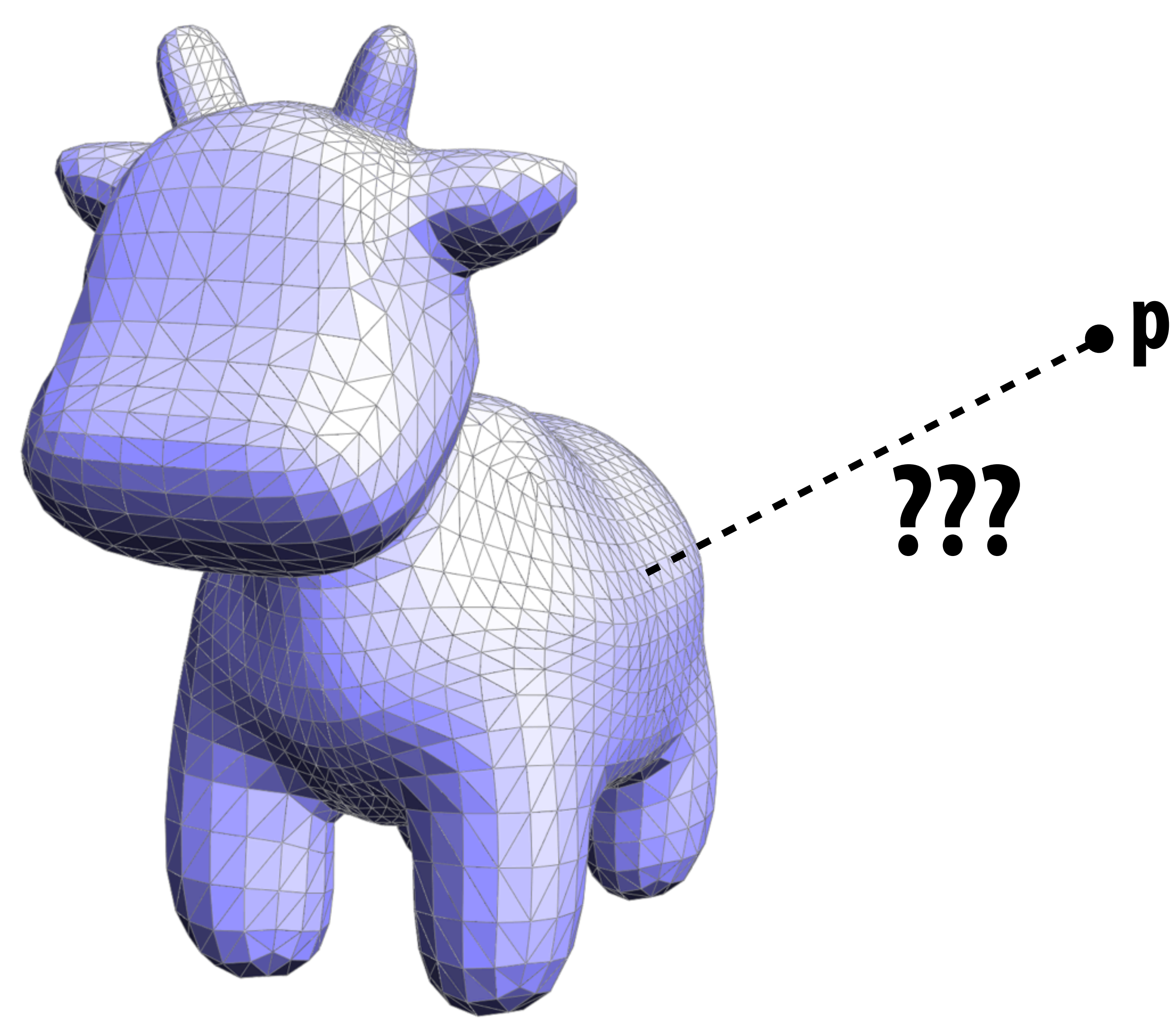


(Q: What happens with an image?)

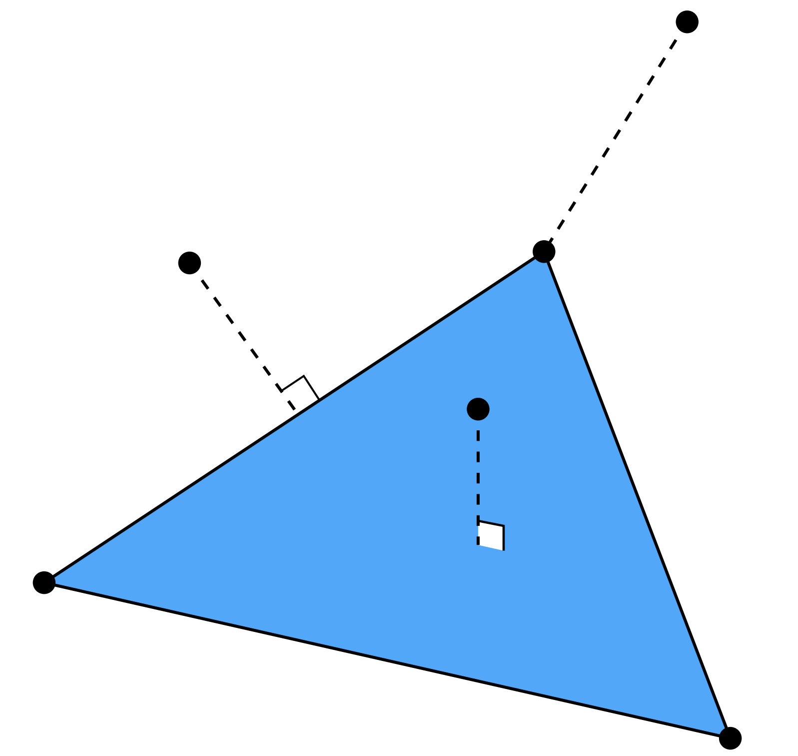
But wait: we have the original mesh. Why not just project each new sample point onto the closest point of the original mesh?

Geometric Queries

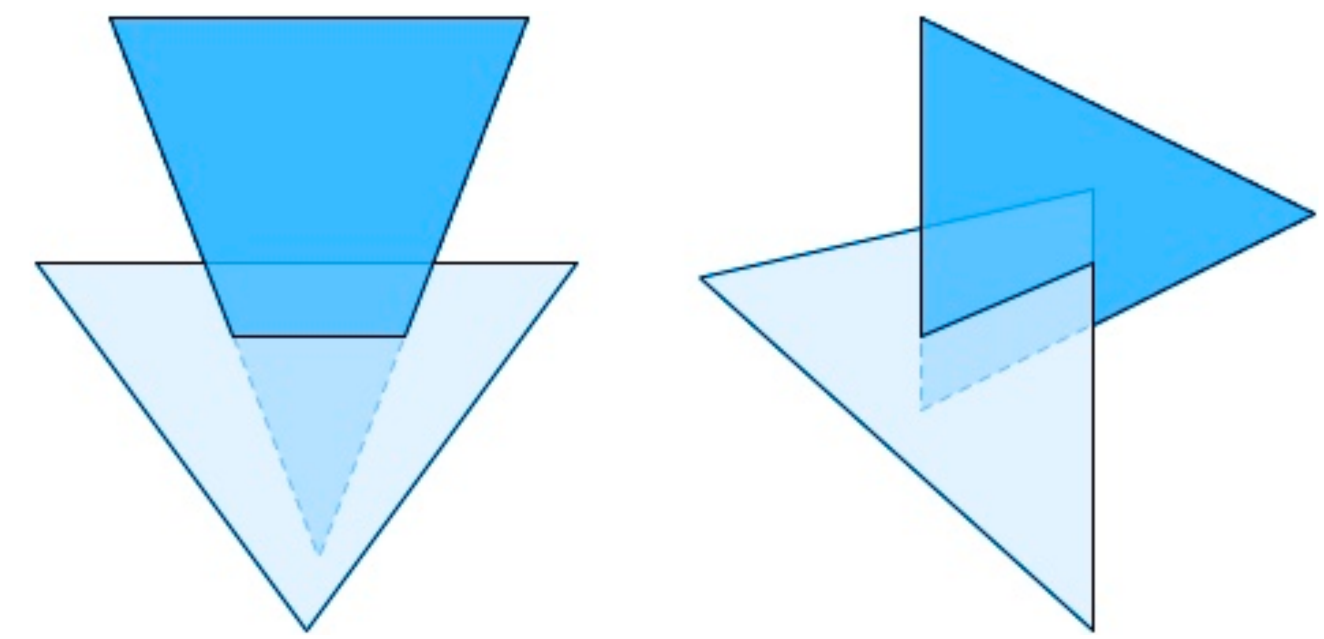
- Q: Given a point, in space (e.g., a new sample point), how do we find the closest point on a given surface?
- Q: Does implicit/explicit representation make this easier?
- Q: Does our halfedge data structure help?
- Q: What's the cost of the naïve algorithm?
- Q: How do we find the distance to a single triangle anyway?
- So many questions!



Many types of geometric queries



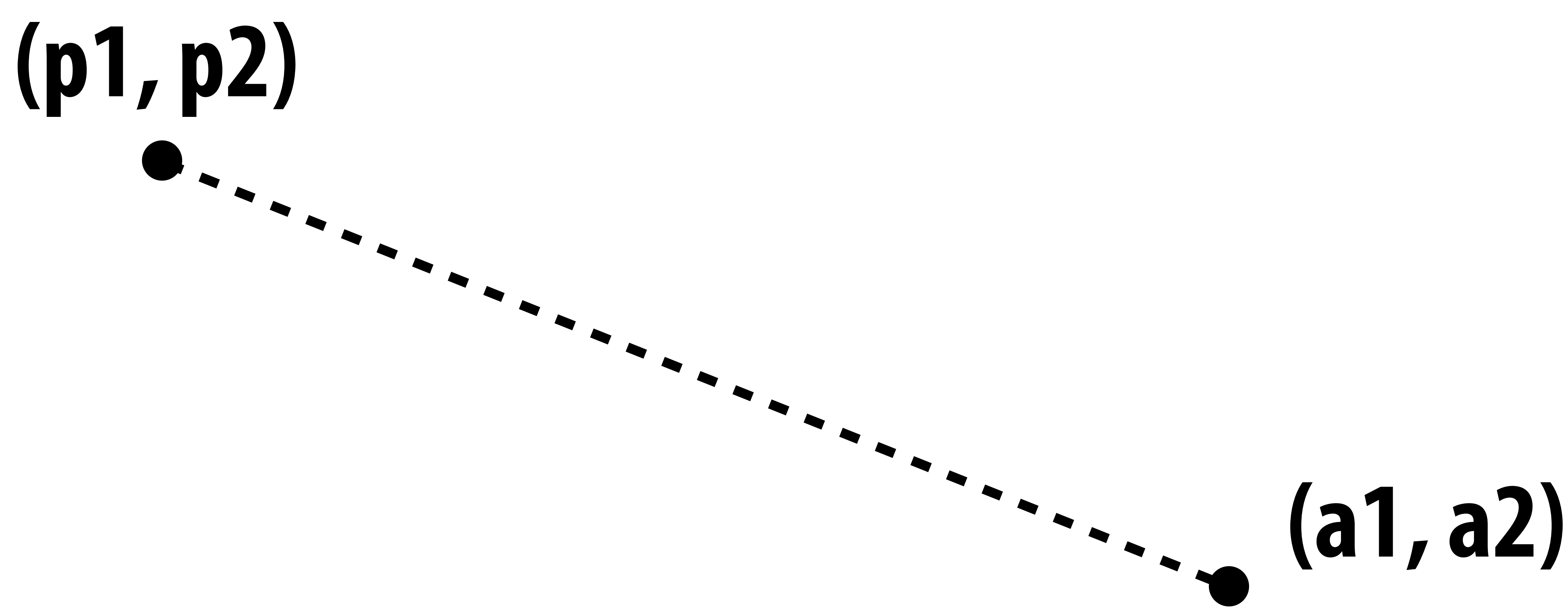
- Already identified need for “closest point” query
- Plenty of other things we might like to know:
  - Do two triangles intersect?
  - Are we inside or outside an object?
  - Does one object contain another?
  - ...



- 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 lecture: intelligent ways to accelerate geometric queries.

Warm up: closest point on point

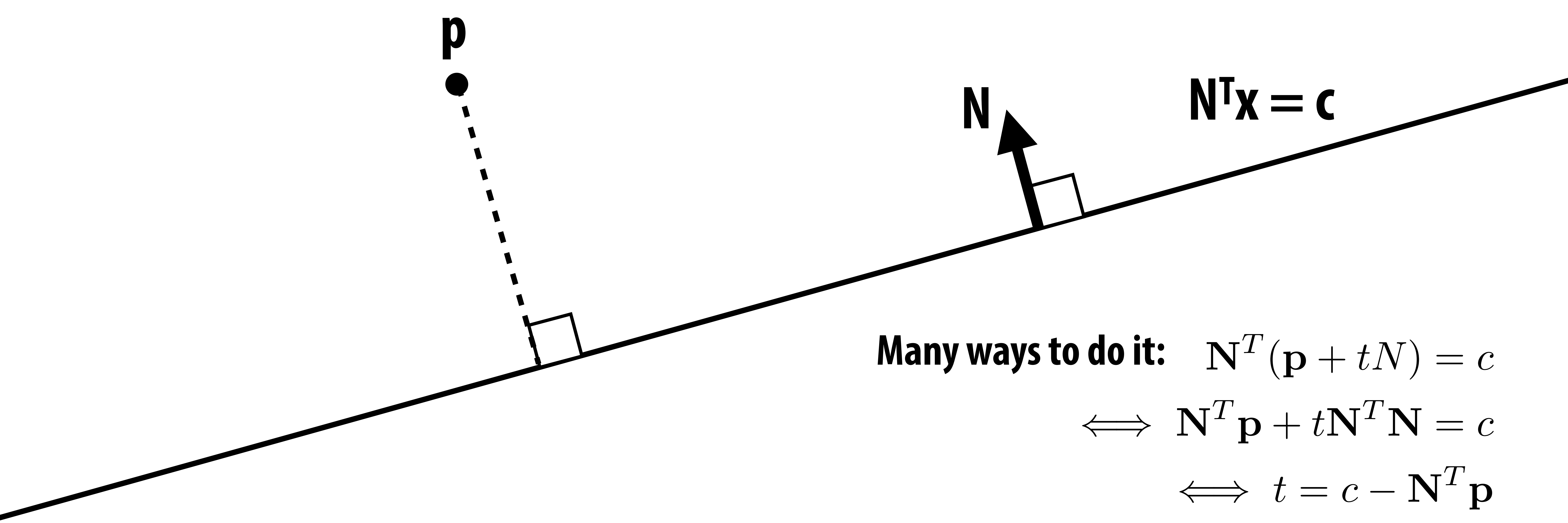
- Goal is to find the point on a mesh closest to a given point.
- *Much* simpler question: given a query point (p1,p2), how do we find the closest point on the point (a1,a2)?



Bonus question: what's the distance?

Slightly harder: closest point on line

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

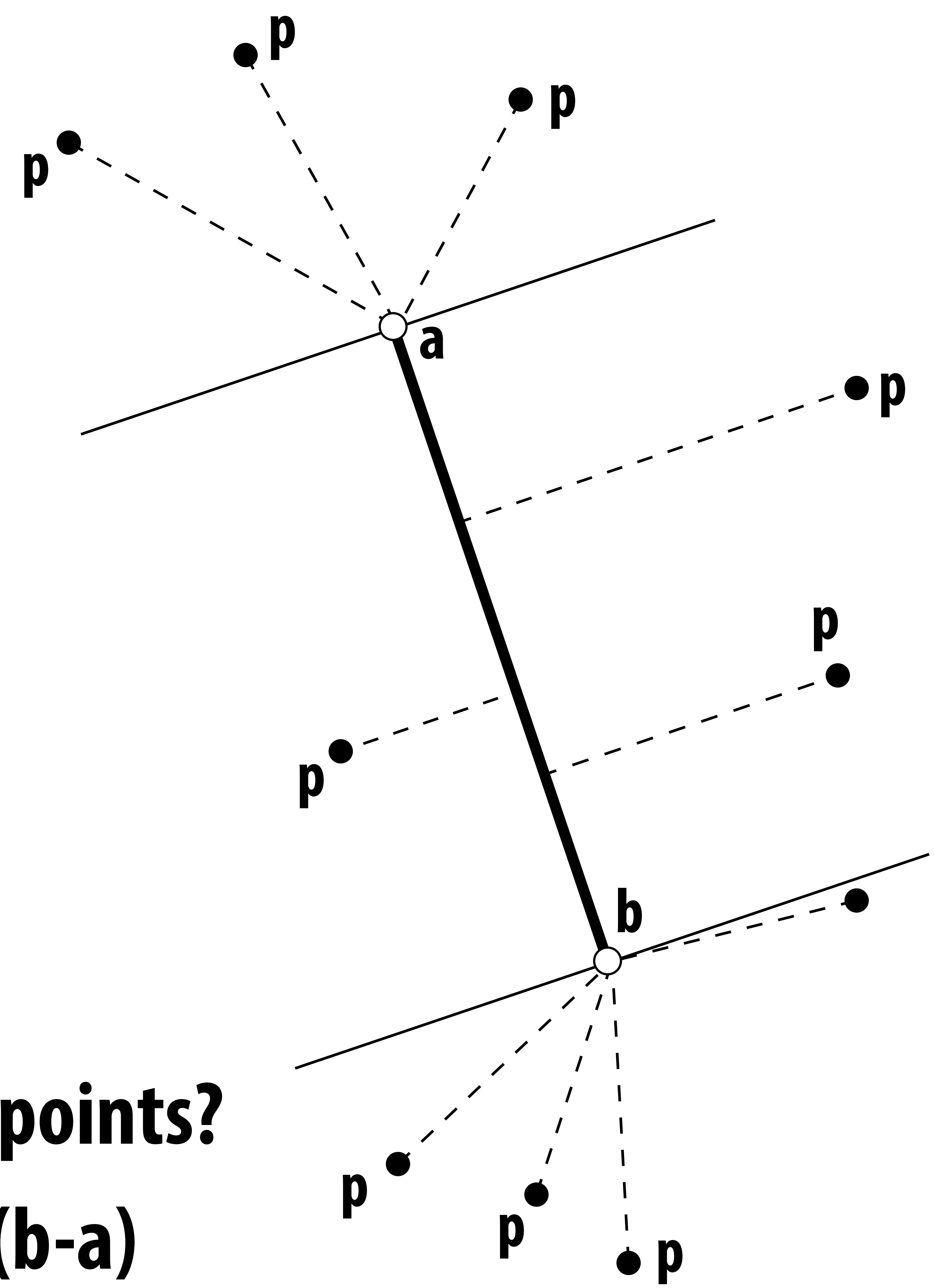


Many ways to do it:

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

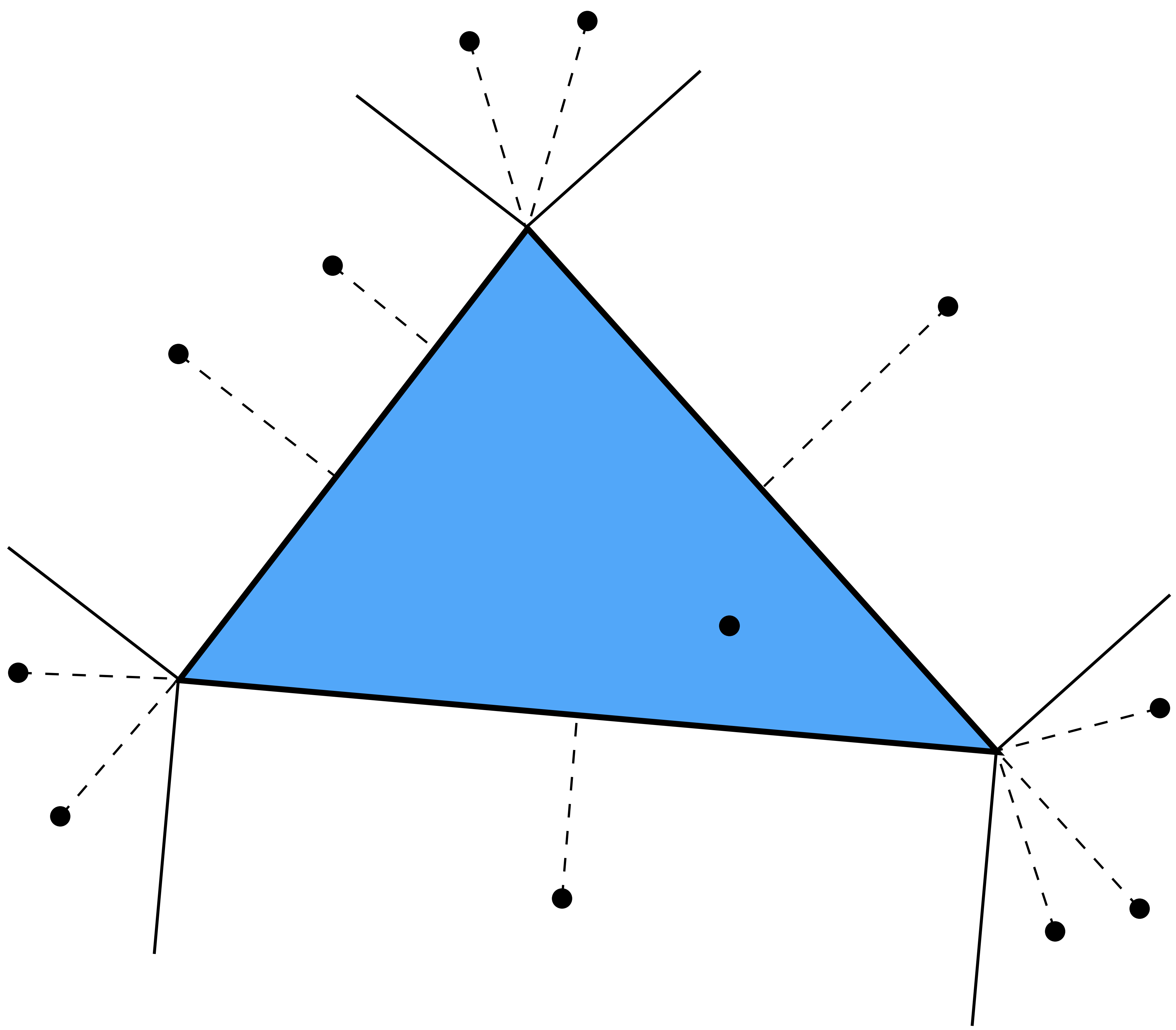
Harder: closest point on line segment

- 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:



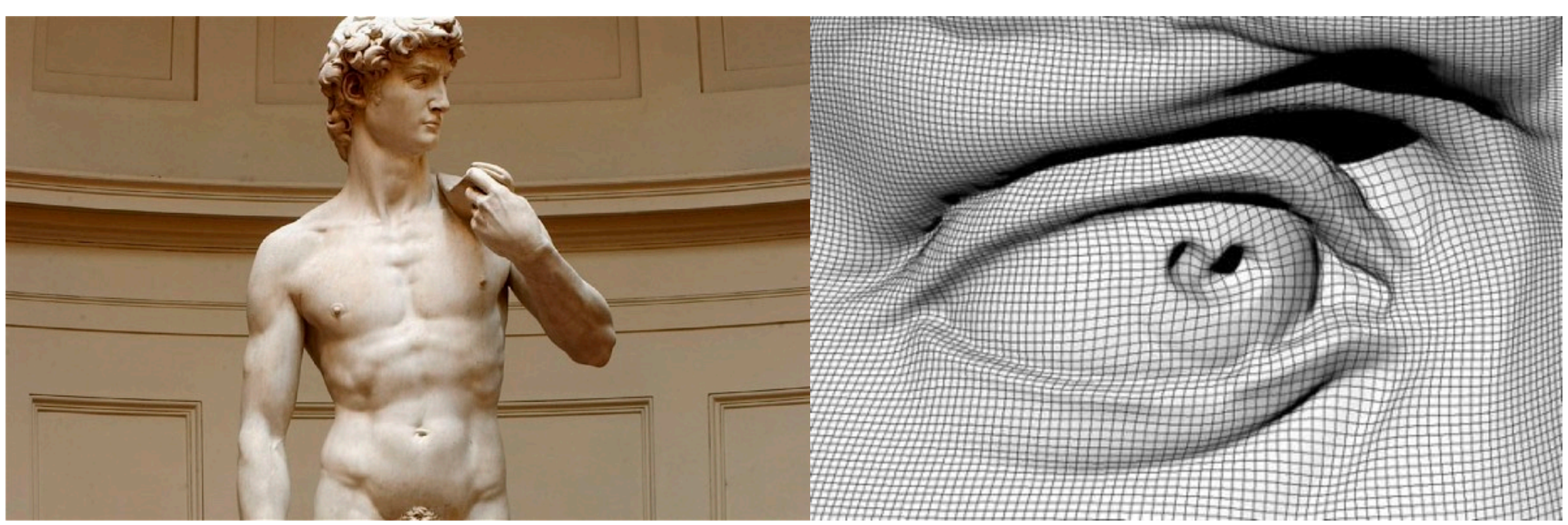
Question: what about a point inside the triangle?

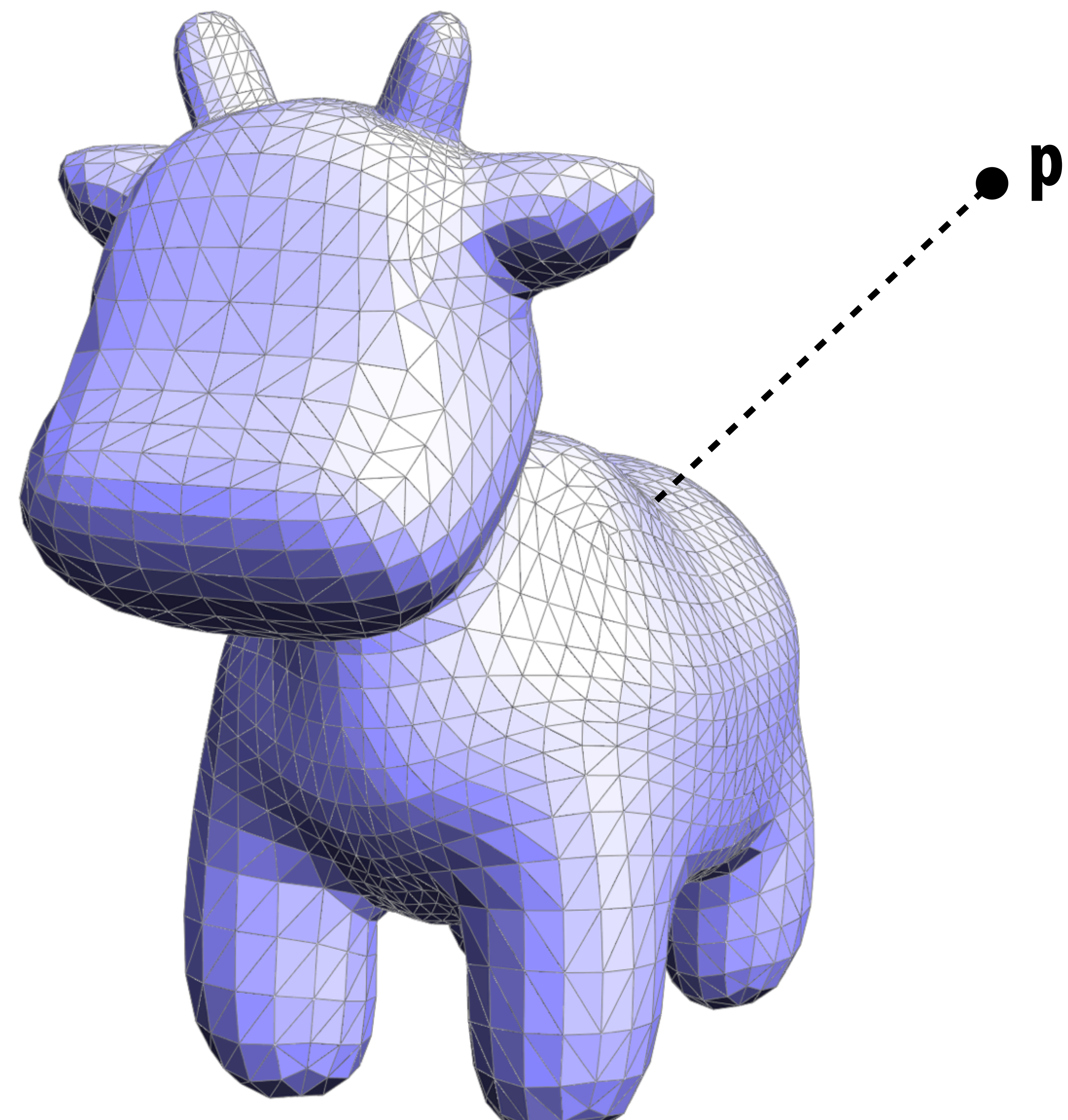
Closest point on triangle in 3D

- Not so different from 2D case
- Algorithm?
  - project onto plane of triangle
  - use half-plane tests to classify point
  - 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?

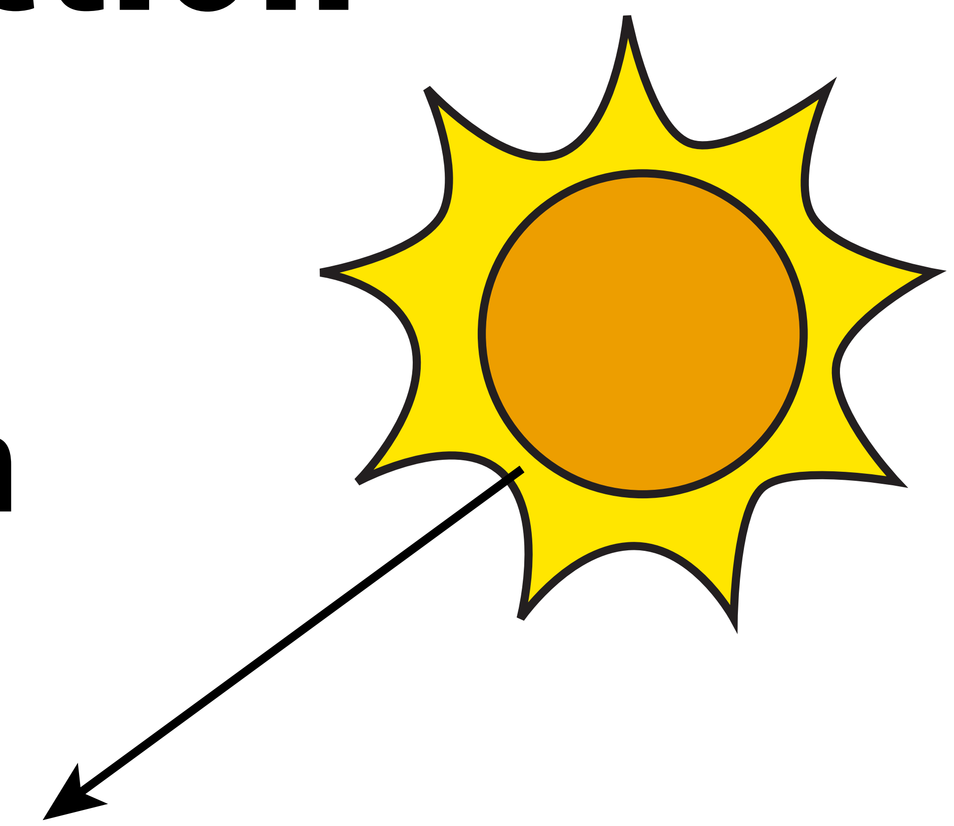
- Conceptually easy:
  - loop over all triangles
  - compute closest point to current triangle
  - keep globally closest point
- Q: What's the cost? Does halfedge help?
- What if we have *billions* of faces?
- (Next time!)

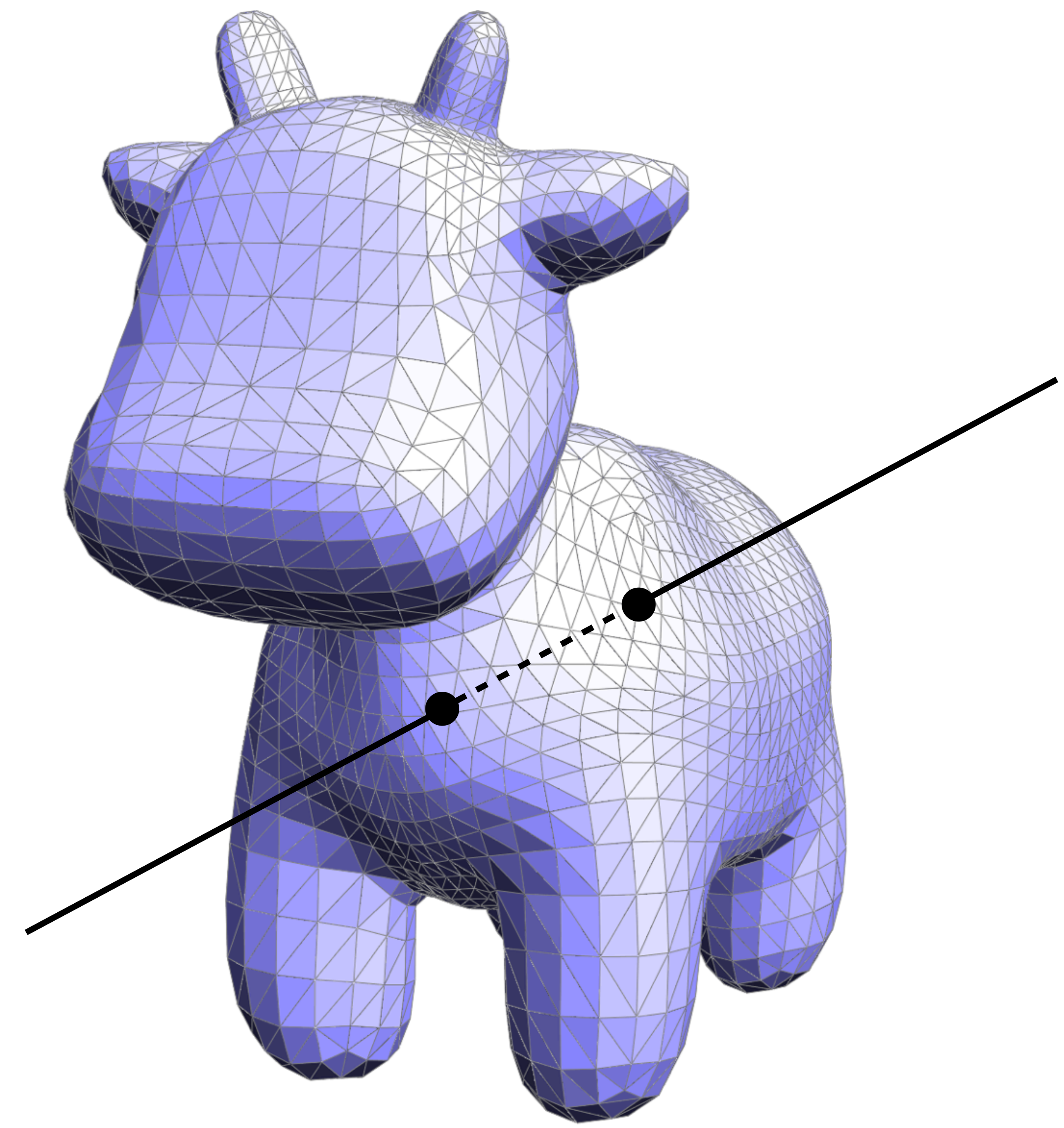




Different query: ray-mesh intersection

- 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
  - SIMULATION: collision detection
- Might pierce surface in many places!





Ray equation

- Can express ray as

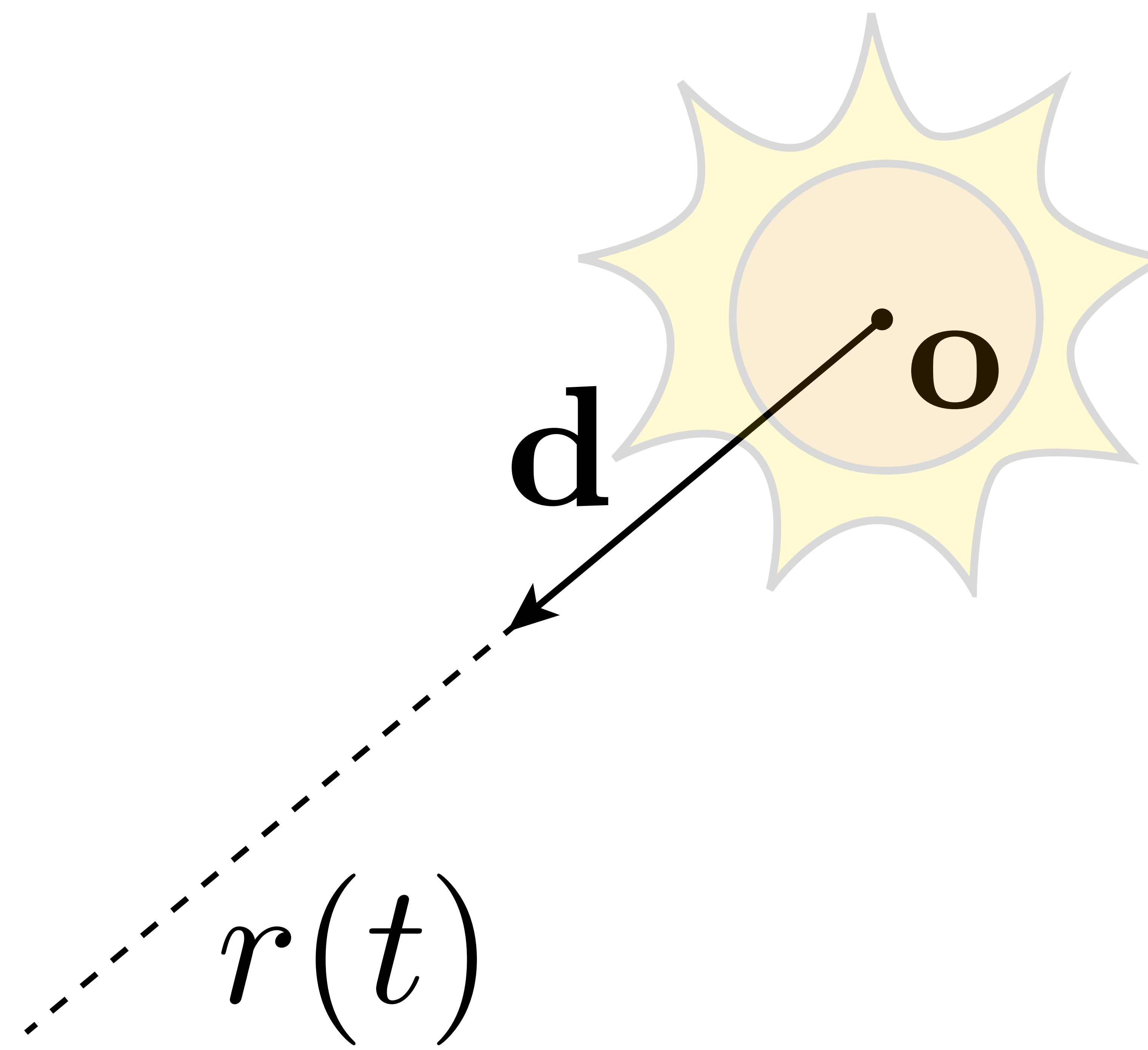
origin

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

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

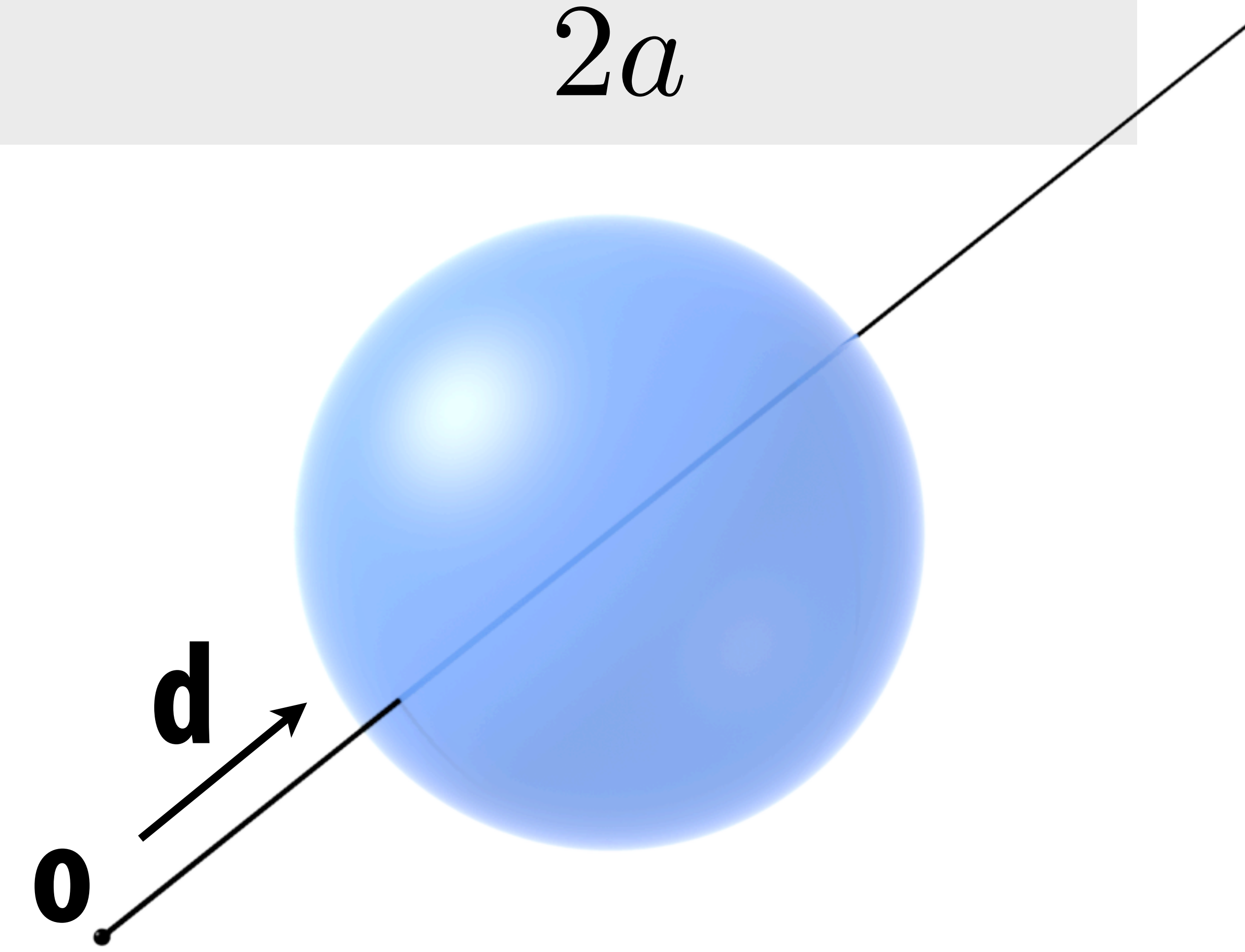
$$\Rightarrow f(\mathbf{r}(t)) = |\mathbf{o} + t\mathbf{d}|^2 - 1$$

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

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

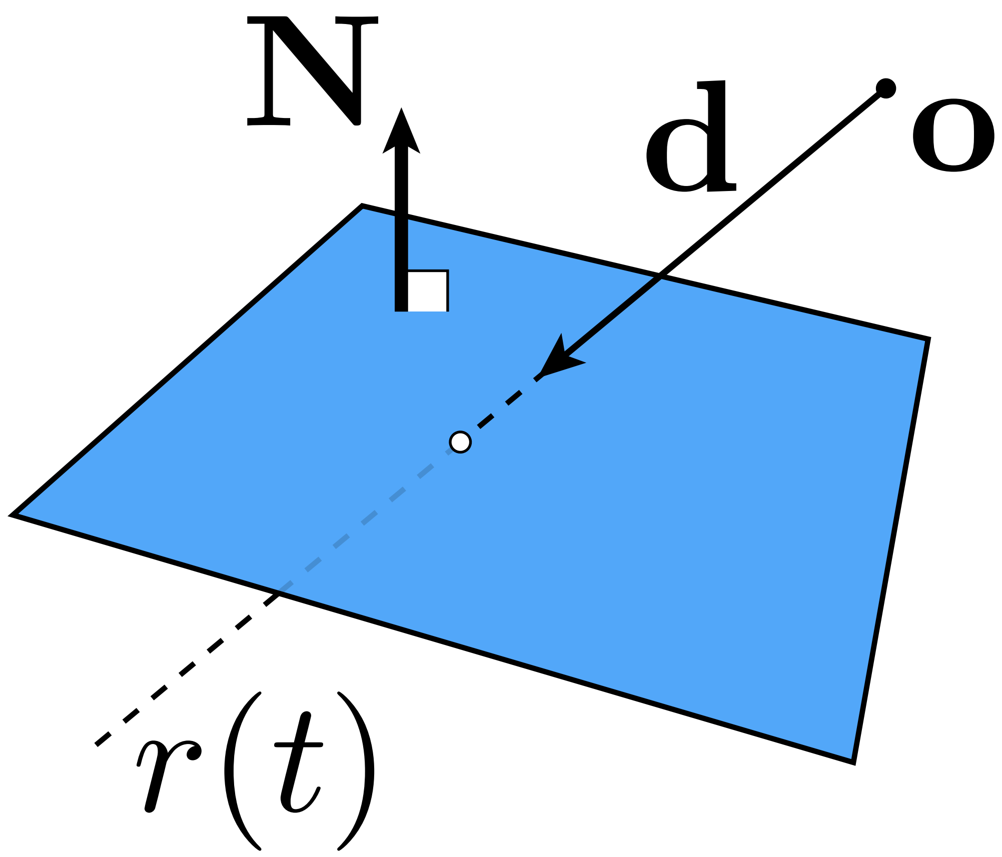
quadratic formula:

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



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:

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

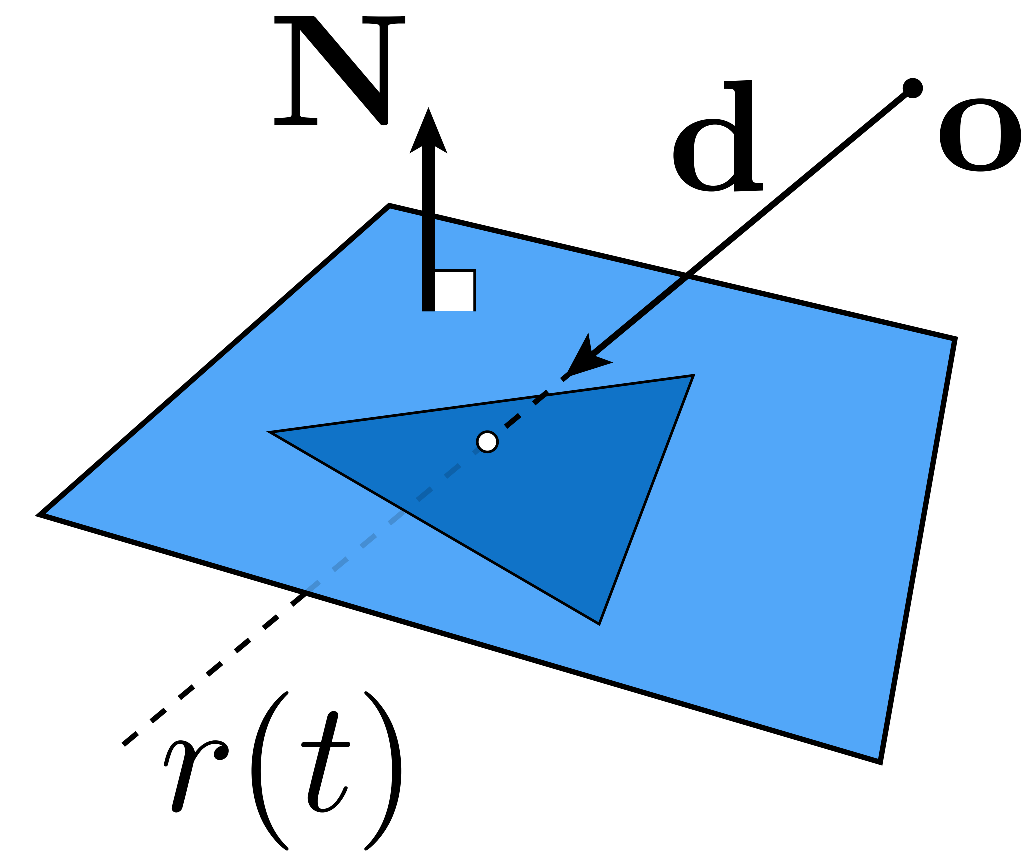
- Now solve for t:

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

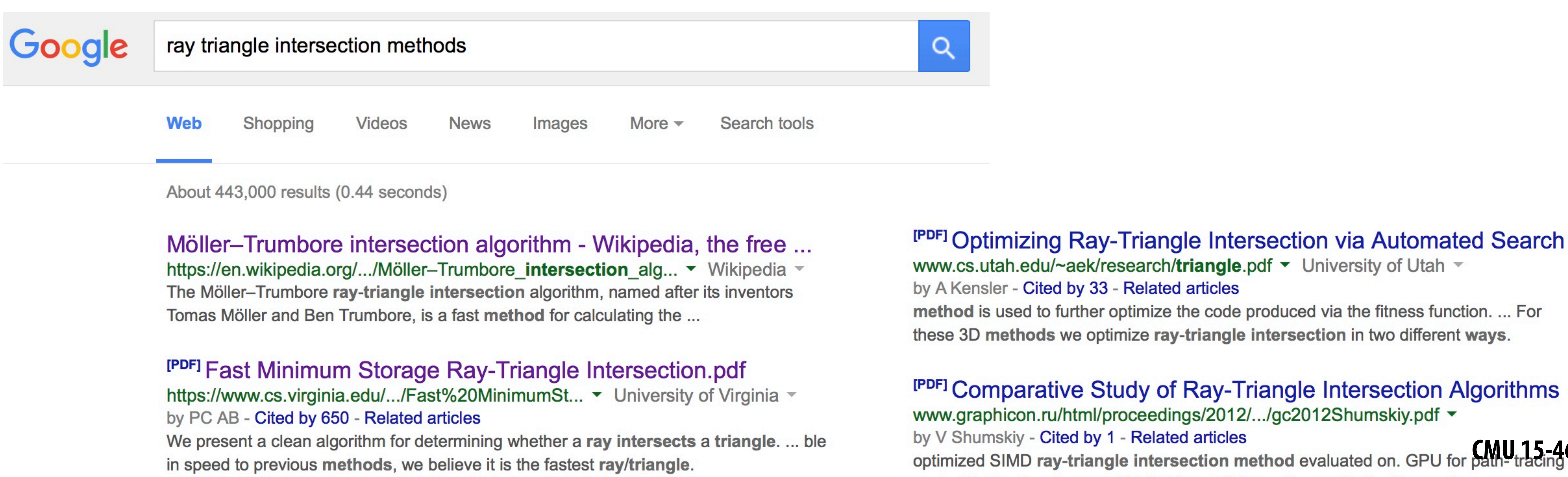
- And plug t back into ray equation:

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

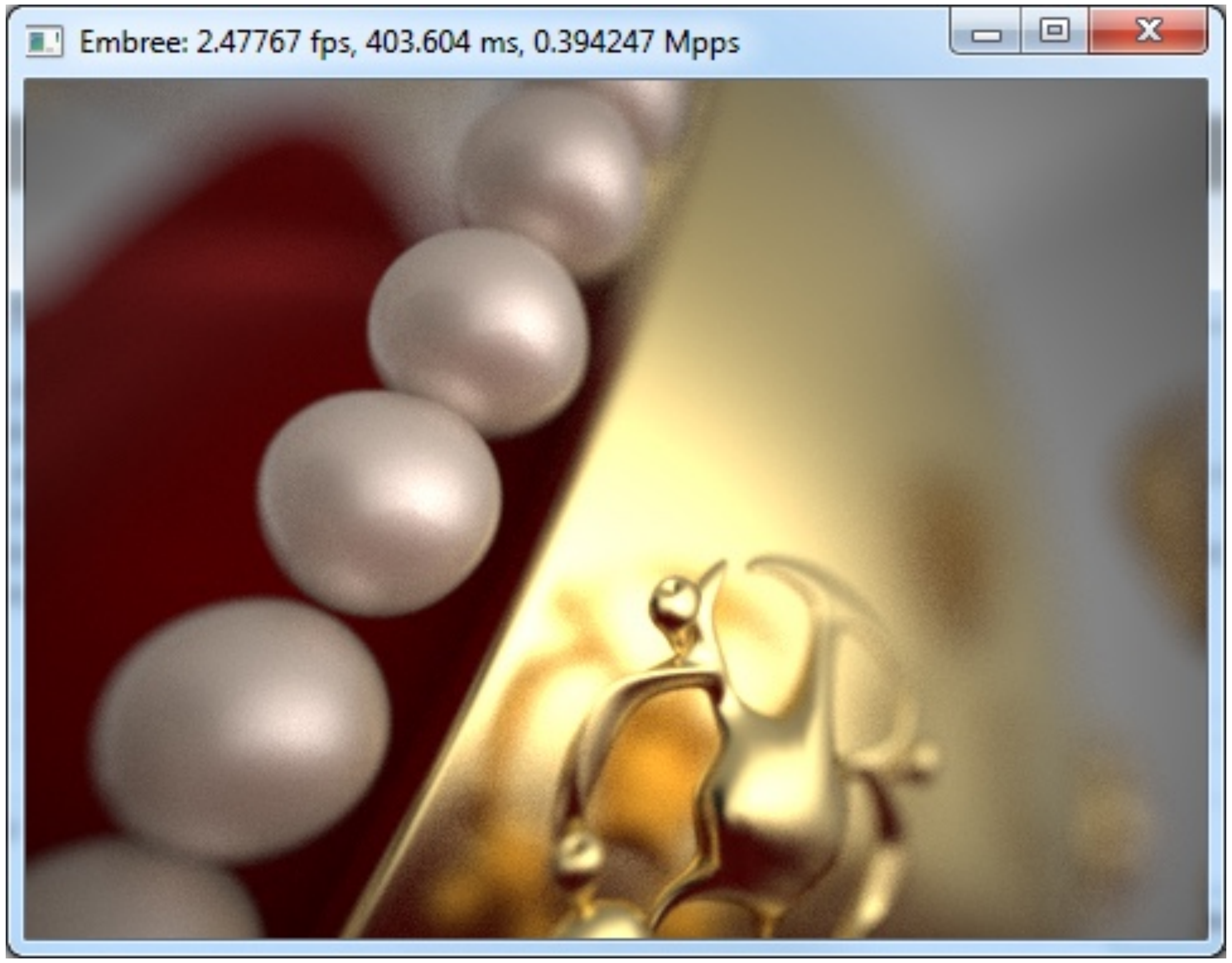
Ray-triangle intersection



- 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

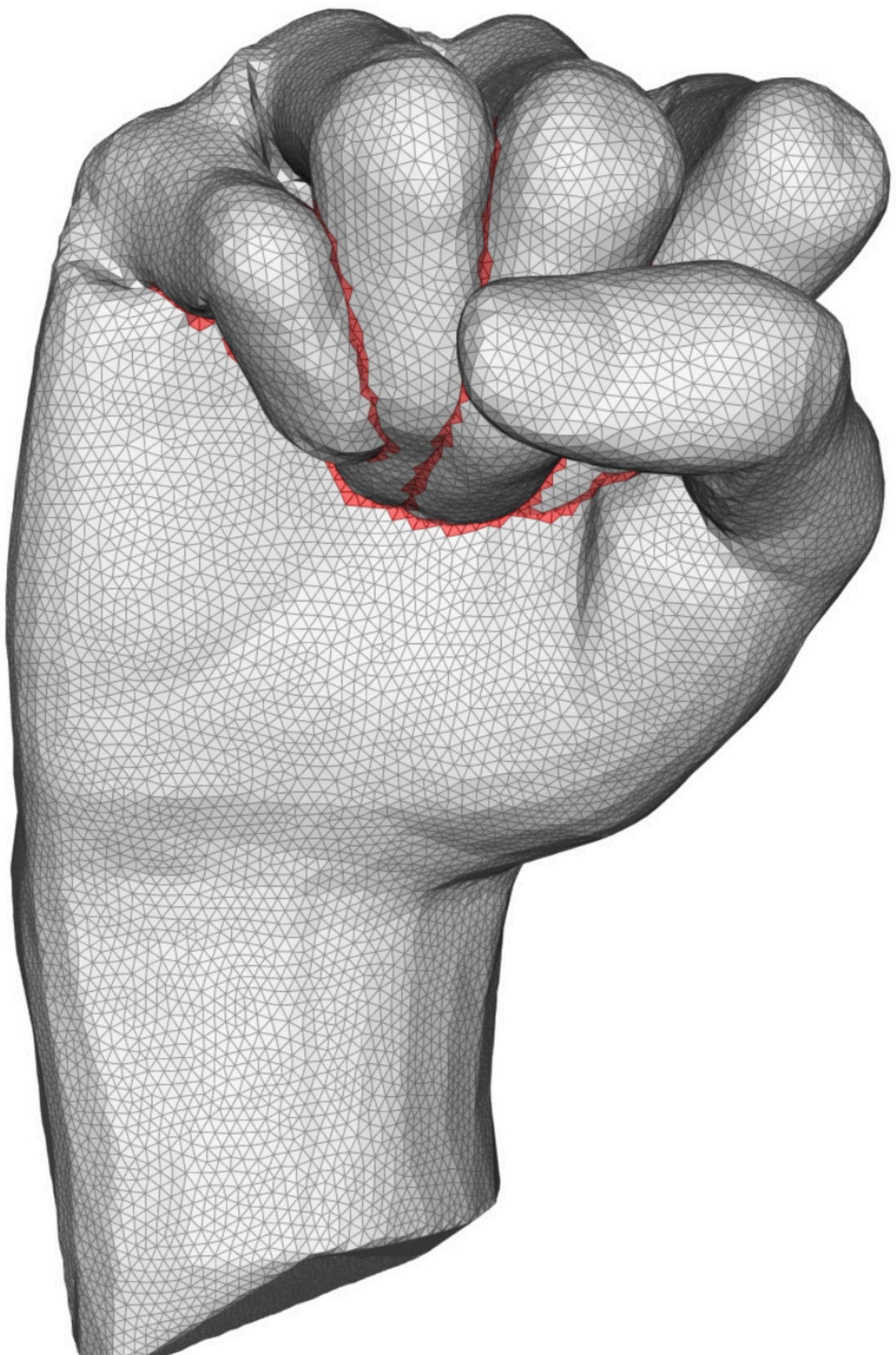
NVIDIA OptiX

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?



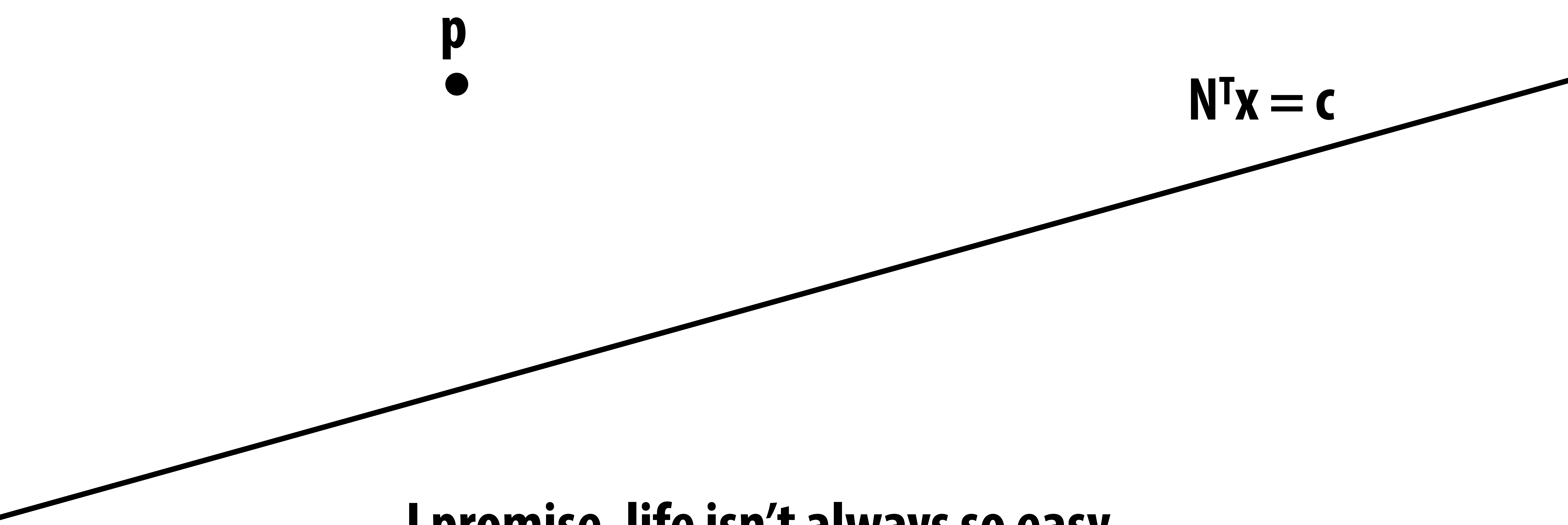
Warm up: point-point intersection

- **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

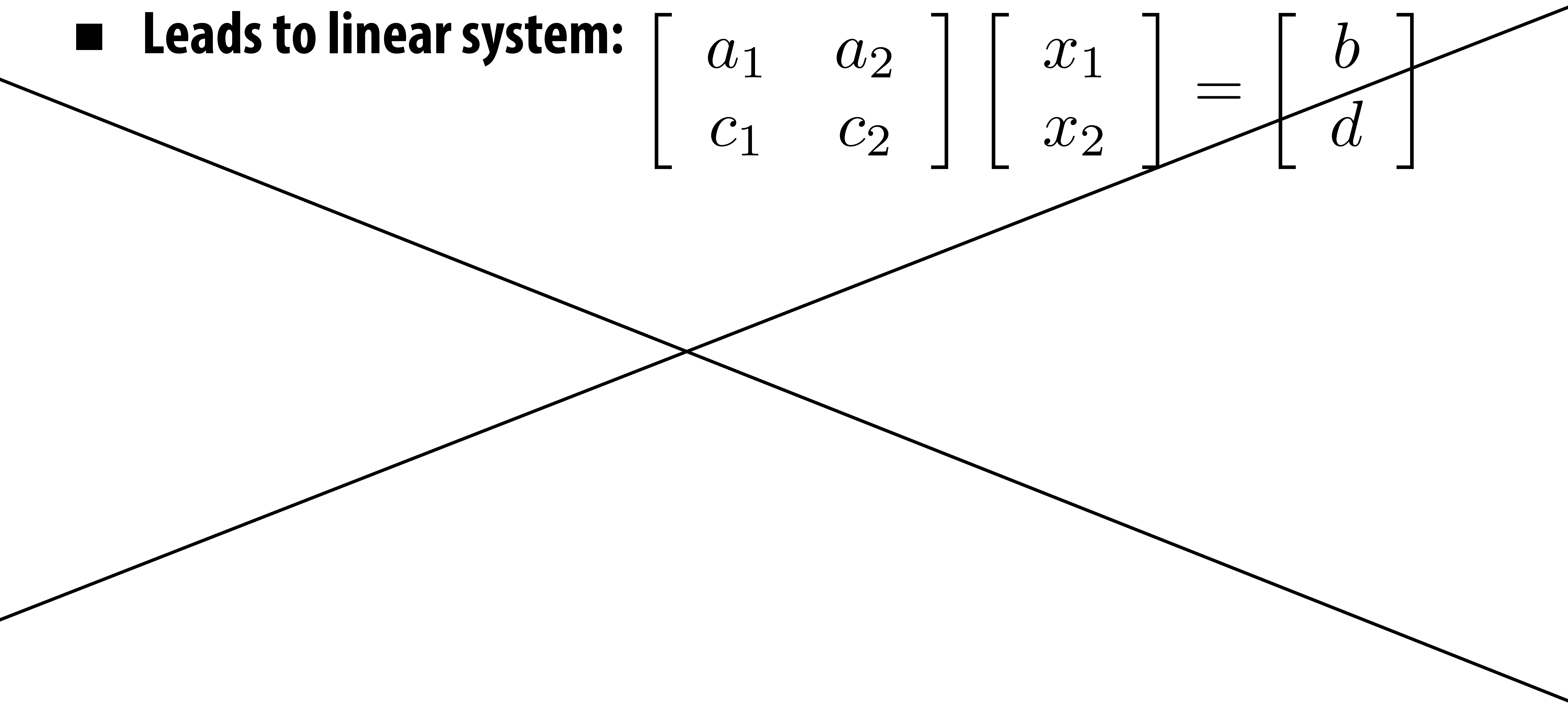
- Q: How do we know if a point intersects a given line?
- A: ...plug it into the line equation!



I promise, life isn't always so easy.

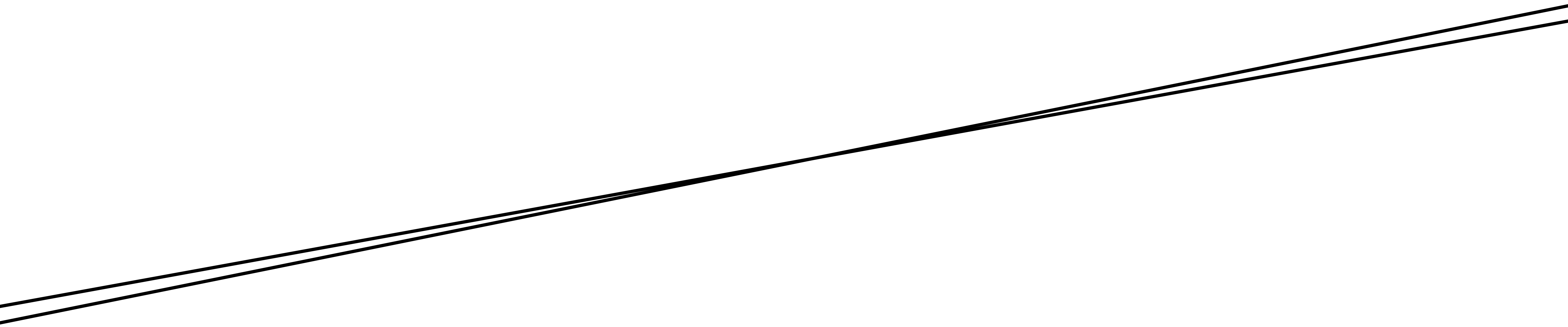
Finally interesting: line-line intersection

- **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:** $$\begin{bmatrix} a_1 & a_2 \\ c_1 & c_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}$$

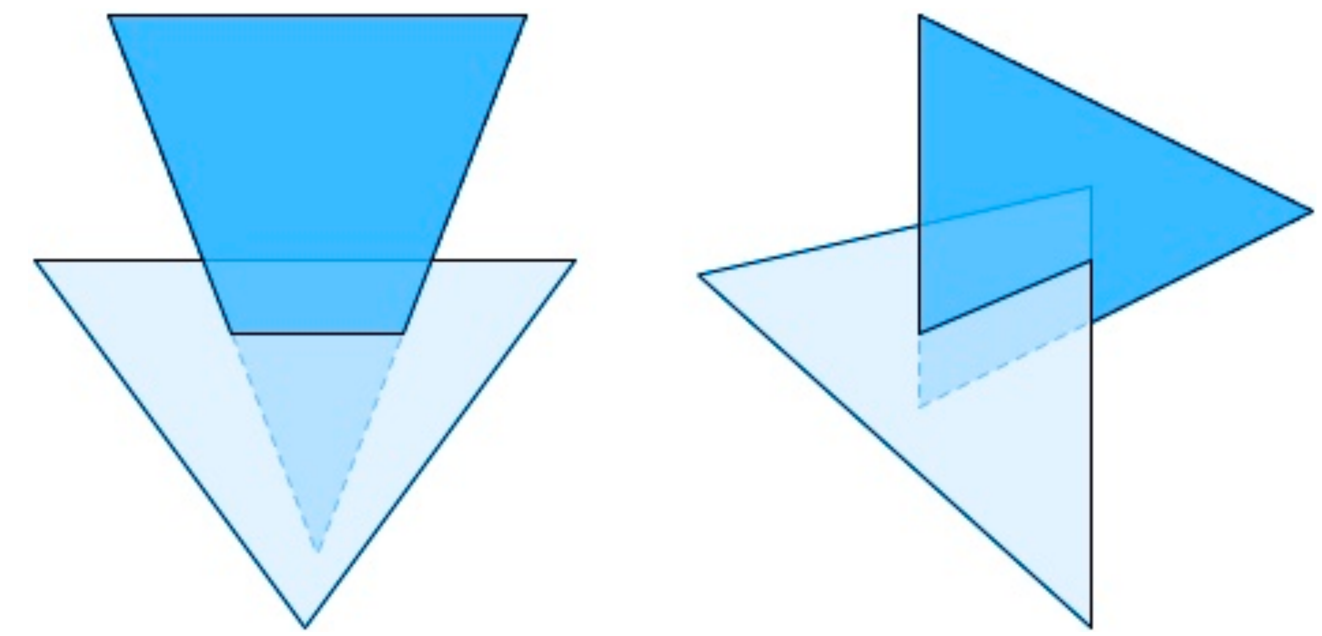


Degenerate line-line intersection?

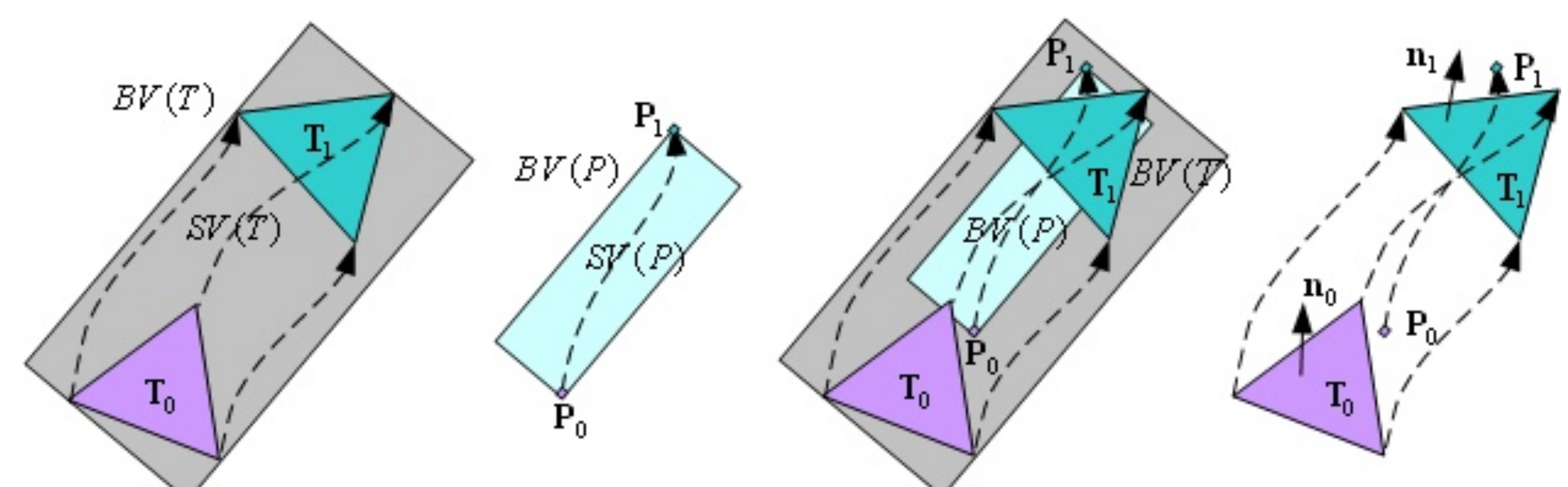
- What if lines are almost parallel?
- Small change in normal can lead to big change in intersection!
- Instability very common, very important with geometric predicates. Demands special care (e.g., analysis of matrix).



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
  - Will say more when we talk about animation!



(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