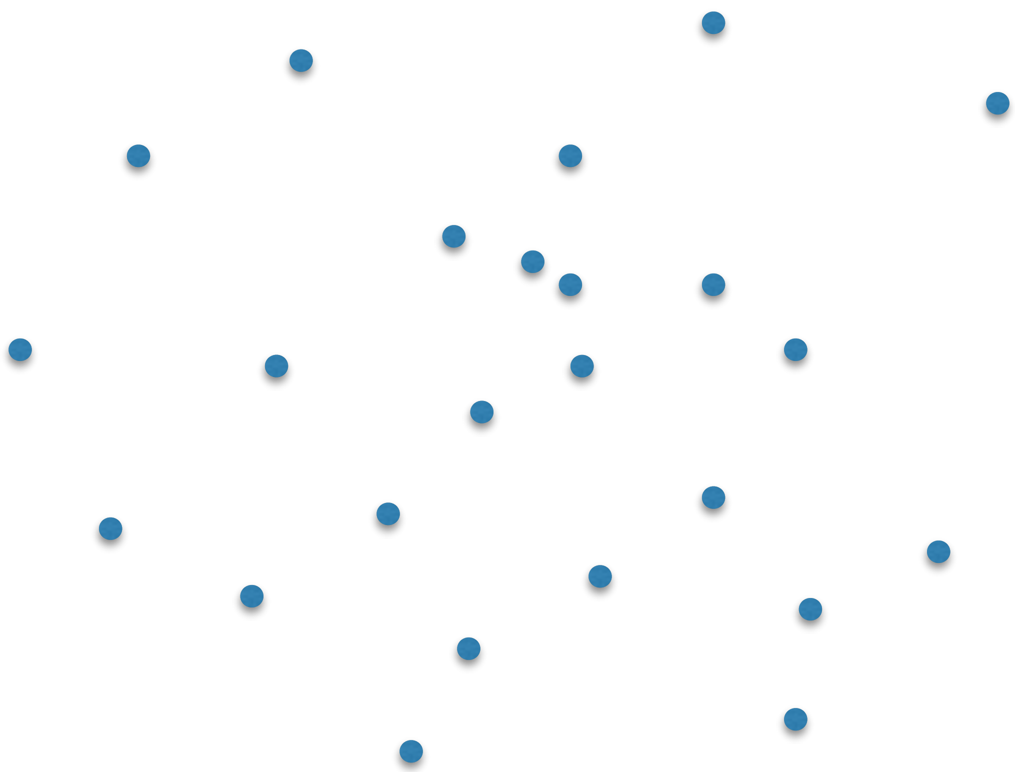
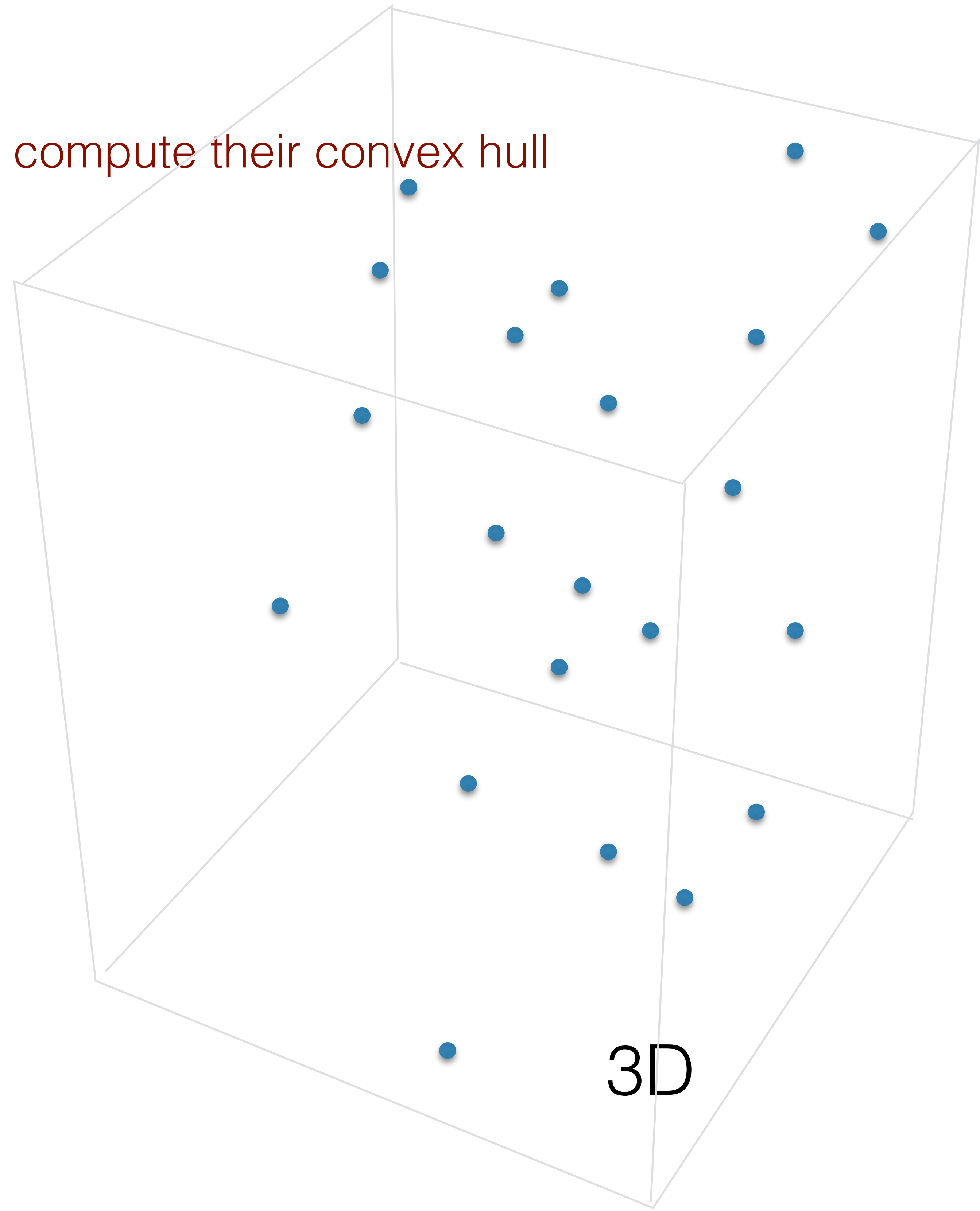
3D convex hulls

Computational Geometry [csci 3250]
Laura Toma
Bowdoin College

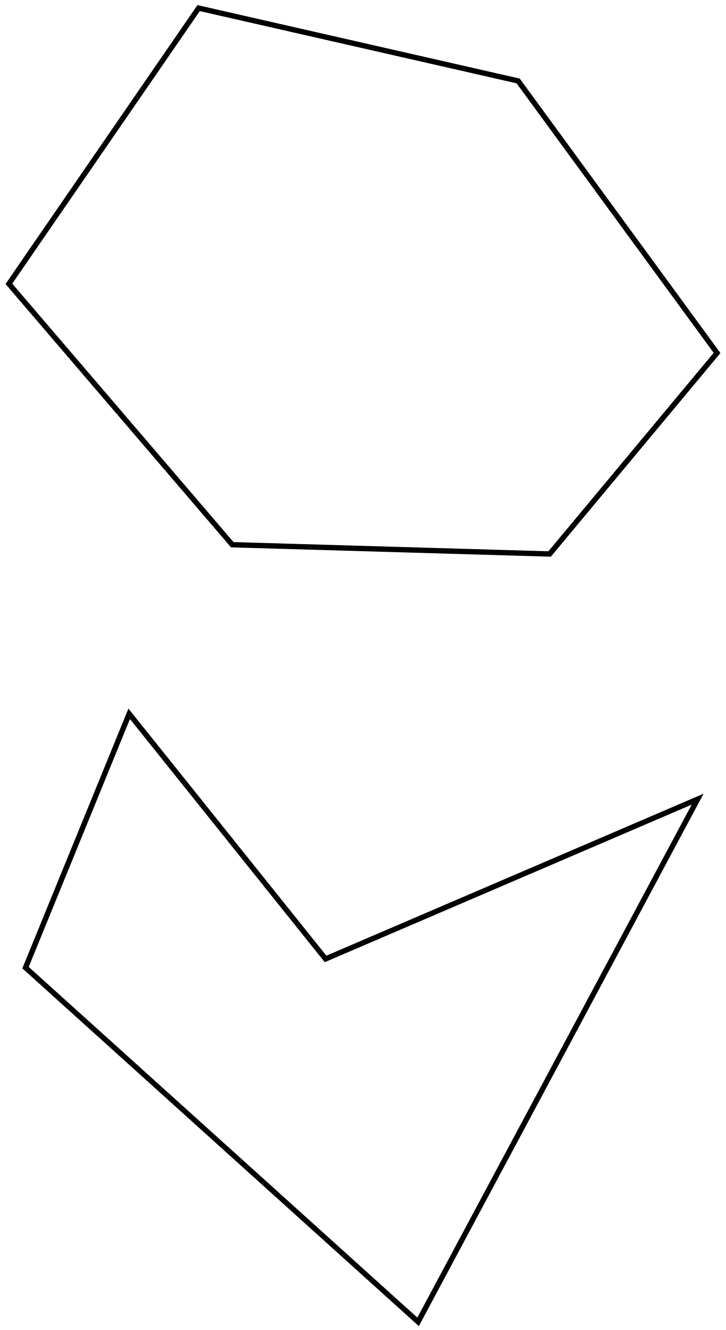
Convex Hulls

The problem: Given a set P of points, compute their convex hull



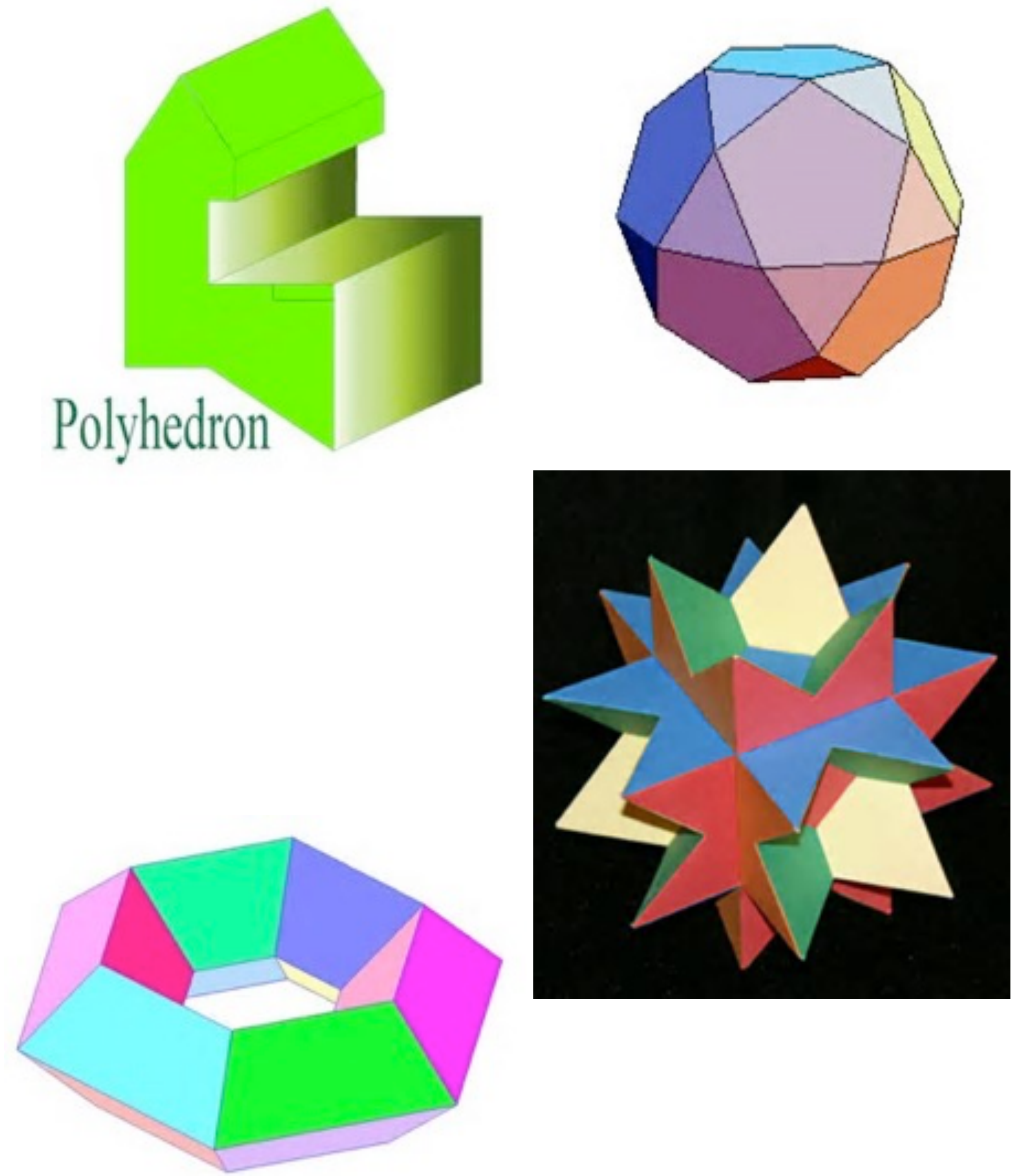


2D



polygon

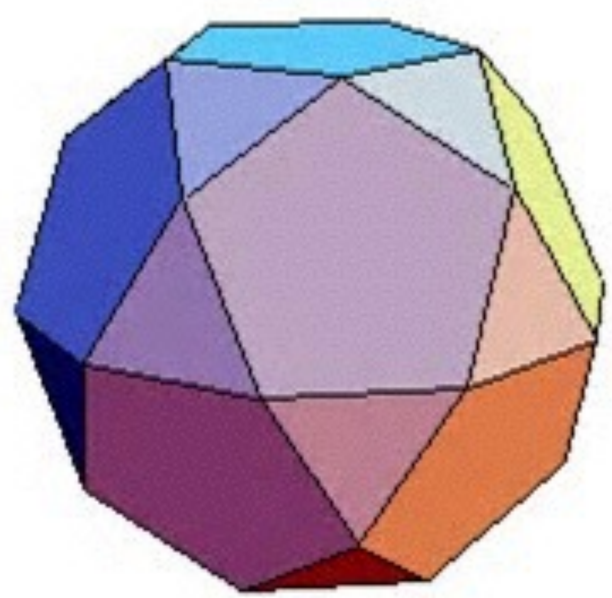
3D

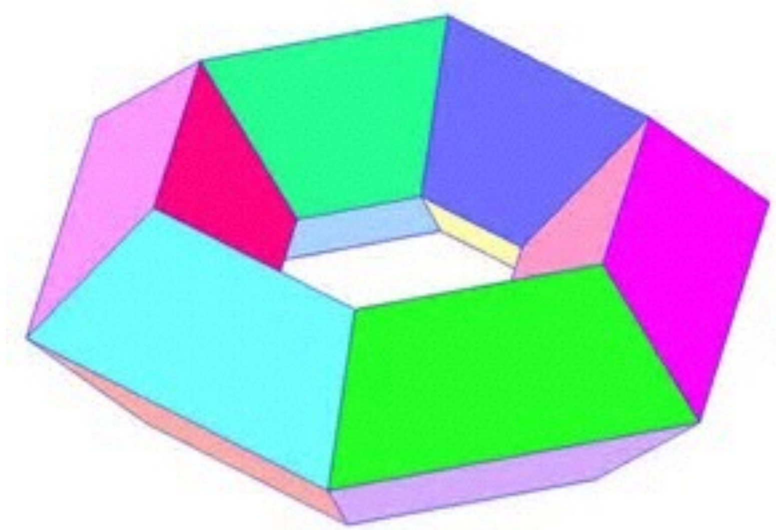


polyhedron

Polyhedron

- region of space whose boundary consists of vertices, edges and faces
- faces intersect properly
- neighborhood of any point on P is homeomorphic to a disk
- surface of P is connected



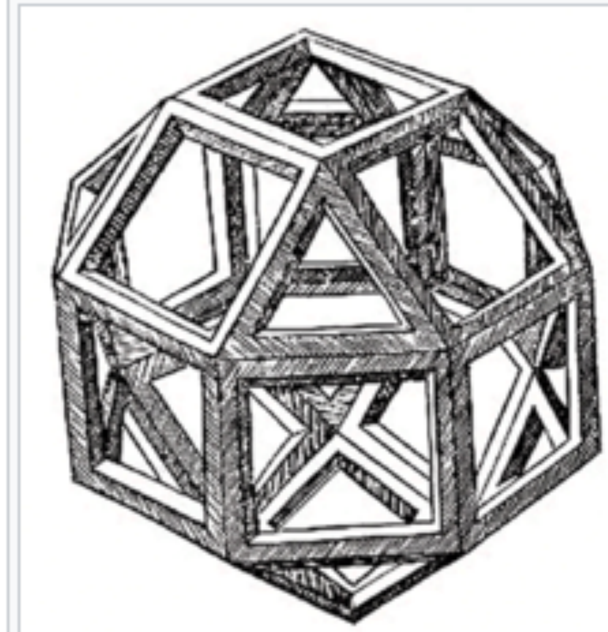


WIKIPEDIA
The Free Encyclopedia

Definition [edit]

Convex polyhedra are well-defined, with several equivalent standard definitions. However, the formal mathematical definition of polyhedra that are not required to be convex has been problematic. Many definitions of "polyhedron" have been given within particular contexts,[1] some more rigorous than others, and there is not universal agreement over which of these to choose. Some of these definitions exclude shapes that have often been counted as polyhedra (such as the self-crossing polyhedra) or include shapes that are often not considered as valid polyhedra (such as solids whose boundaries are not manifolds). As Branko Grünbaum observed,

> "The Original Sin in the theory of polyhedra goes back to Euclid, and through Kepler, Poinsot, Cauchy and many others ... at each stage ... the writers failed to define what are the polyhedra".[2]

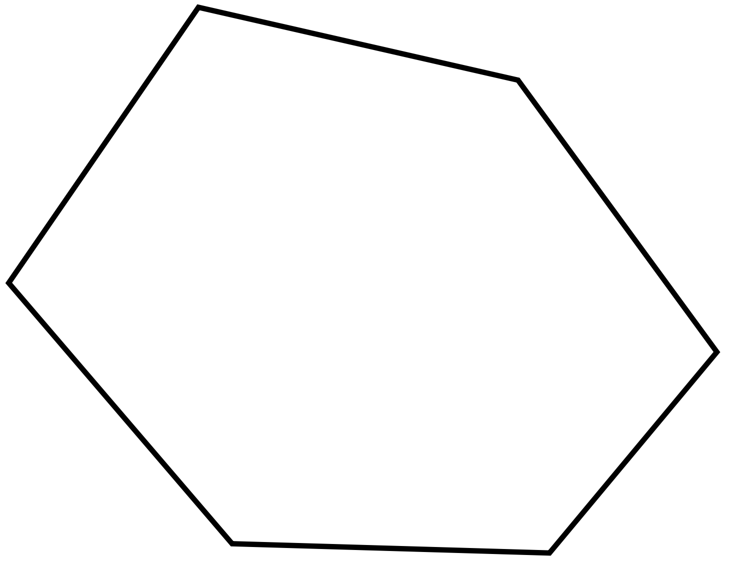


A skeletal polyhedron (specifically, a rhombicuboctahedron) drawn by Leonardo da Vinci to illustrate a book by Luca Pacioli

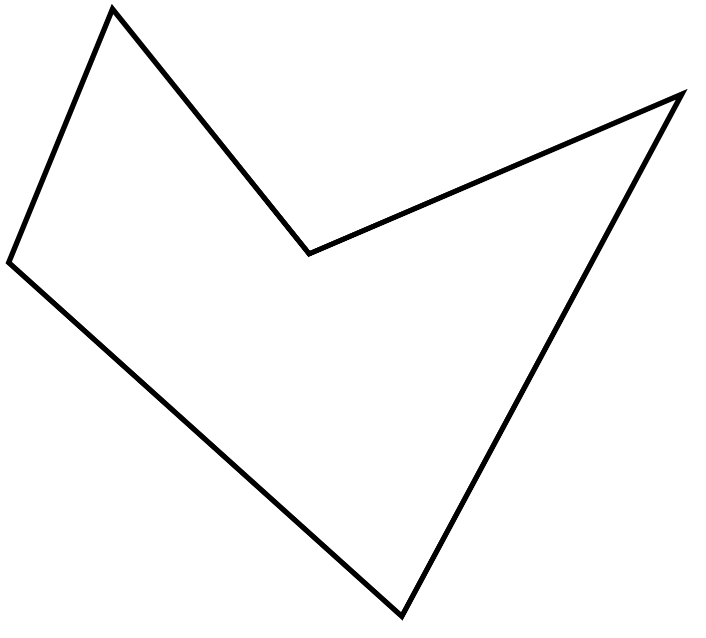
Nevertheless, there is general agreement that a polyhedron is a solid or surface that can be described by its vertices (corner points), edges (line segments connecting certain pairs of vertices), faces (two-dimensional polygons), and sometimes by its three-dimensional interior volume. One can distinguish among these different definitions according to whether they describe the polyhedron as a solid, whether they describe it as a surface, or whether they describe it more abstractly based on its incidence geometry.

Convexity

A polygon P is **convex** if for any p, q in P, the segment pq lies entirely in P.



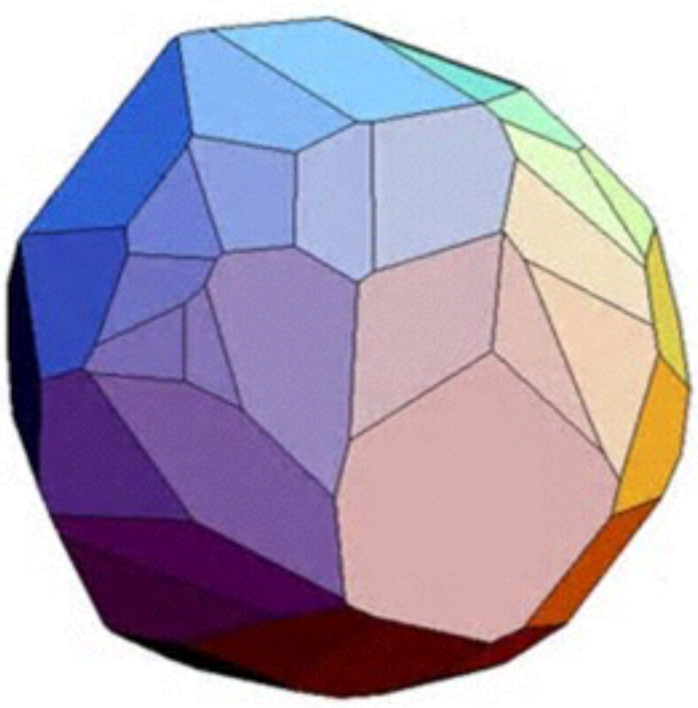
convex



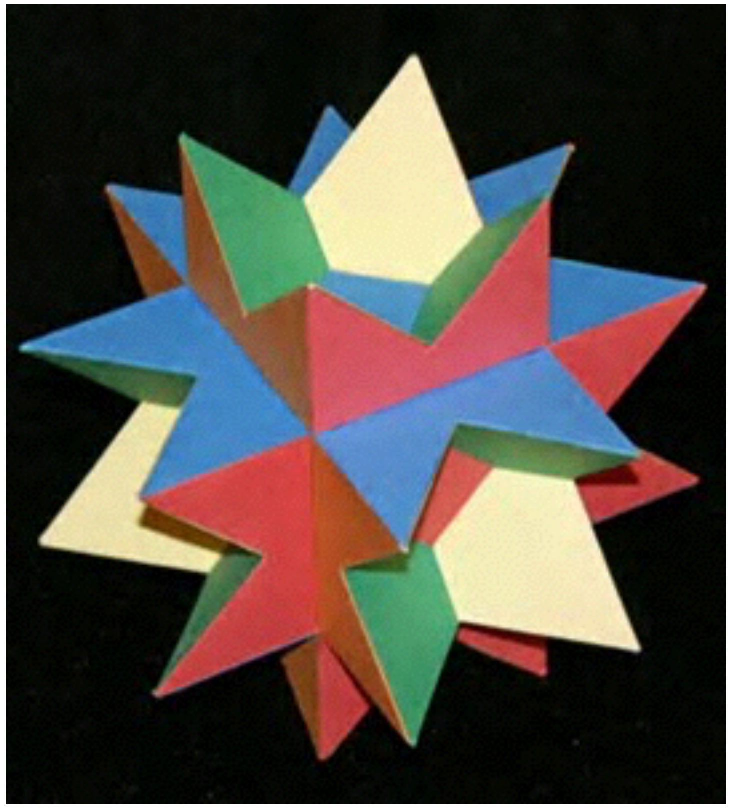
non-convex

Convexity

A polyhedron P is **convex** if for any p, q in P, the segment pq lies entirely in P.



convex



non-convex

convex polyhedron : polytop

digression start

Regular polygons in 2D

- A regular polygon has equal sides and angles

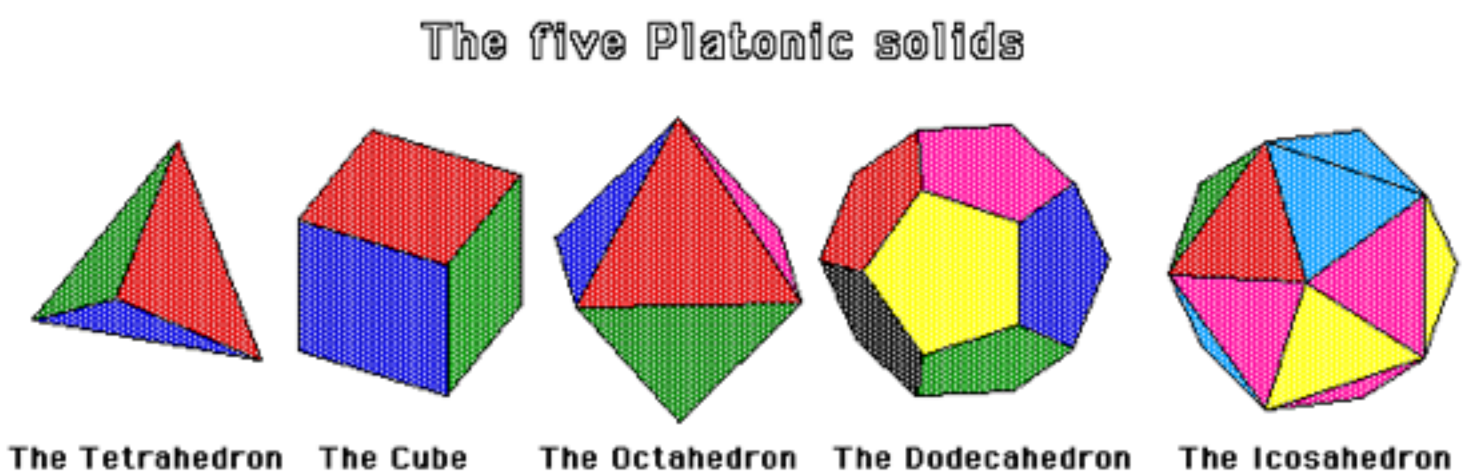
Set of convex regular n-gons

Regular polygons

Regular polytops in 3D

- Regular polytop:
  - faces are congruent regular polygons
  - the number of faces incident to each vertex is the same (and equal angles)

Surprisingly, there exist only 5 regular polytops



The five regular solids discovered by the Ancient Greek mathematicians are:

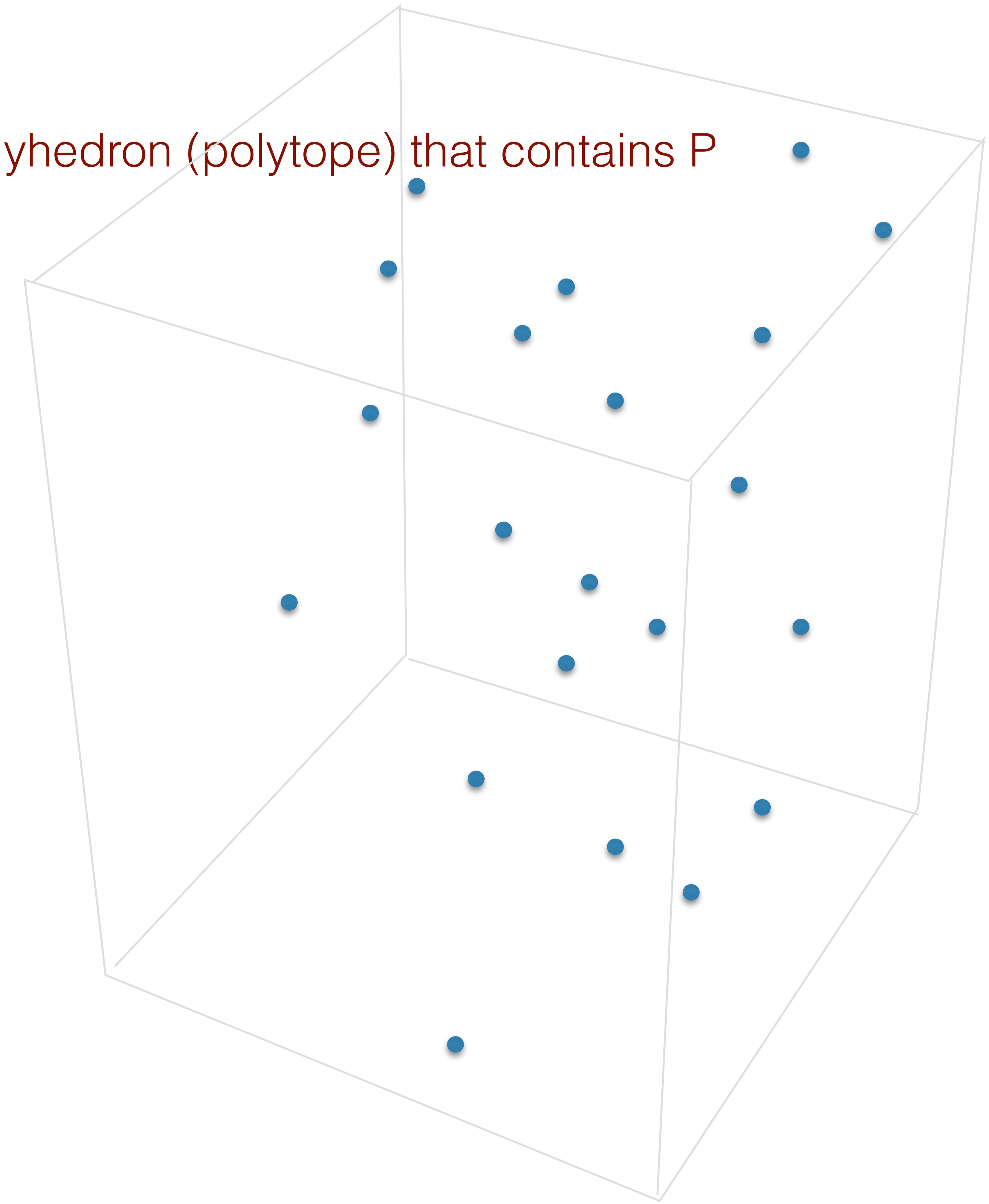
The **Tetrahedron**:	4 vertices	6 edges	4 faces	each with 3 sides
The **Cube**:	8 vertices	12 edges	6 faces	each with 4 sides
The **Octahedron**:	6 vertices	12 edges	8 faces	each with 3 sides
The **Dodecahedron**:	20 vertices	30 edges	12 faces	each with 5 sides
The **Icosahedron**:	12 vertices	30 edges	20 faces	each with 3 sides

The solids are regular because the same number of sides meet at the same angles at each vertex and identical polygons meet at the same angles at each edge.
These five are the only possible regular polyhedra.

digression end

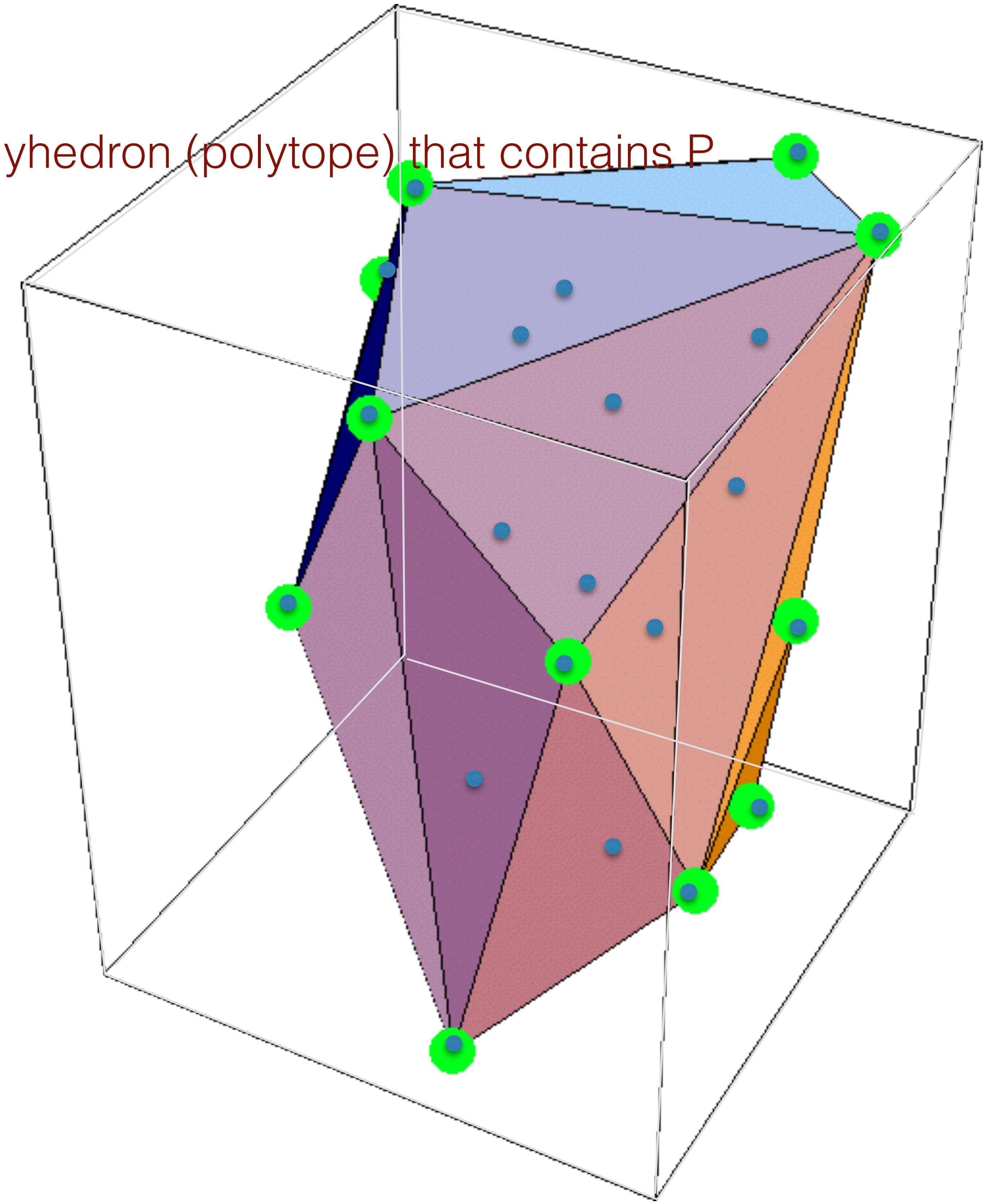
Convex Hulls in 3D

3D convex hull = smallest convex polyhedron (polytope) that contains P



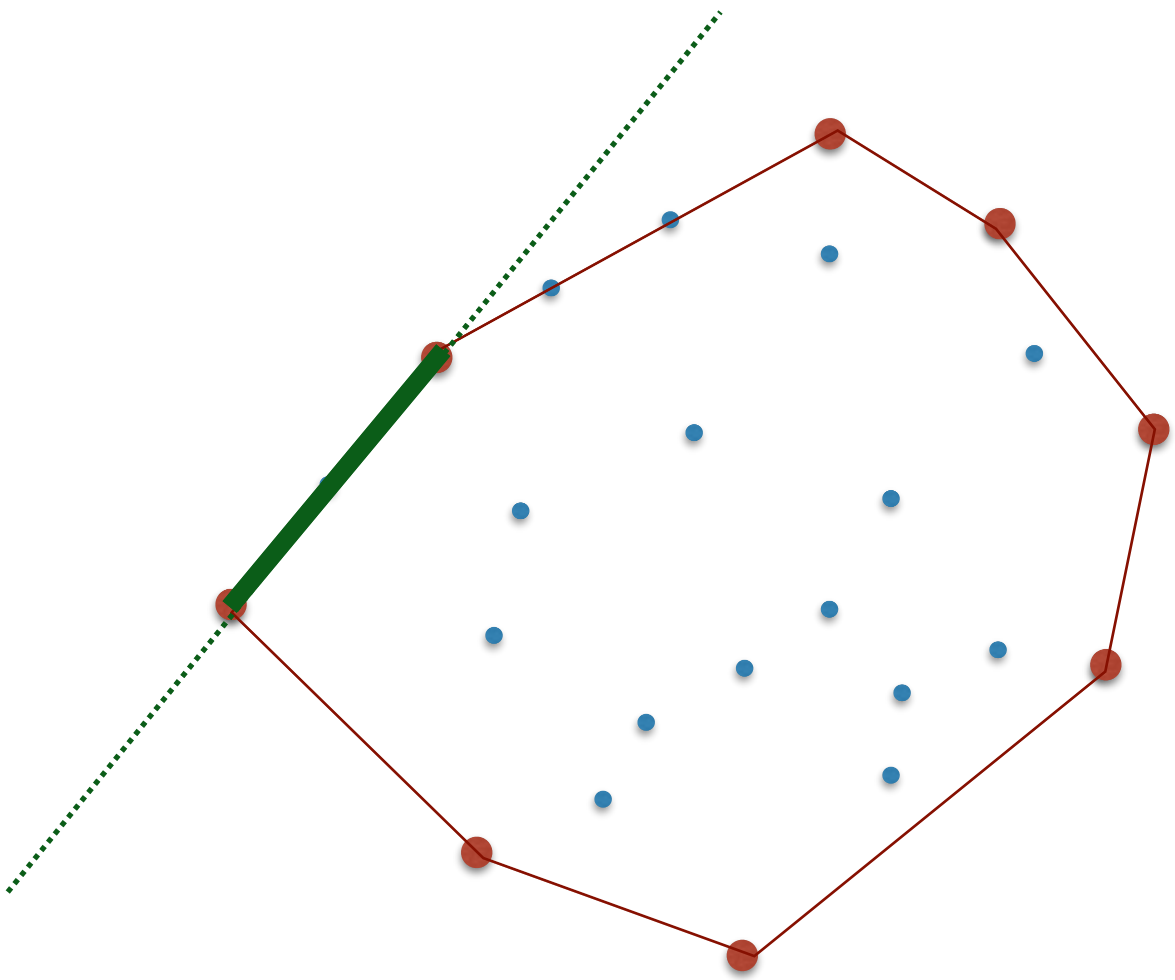
Convex Hulls in 3D

3D convex hull = smallest convex polyhedron (polytope) that contains P



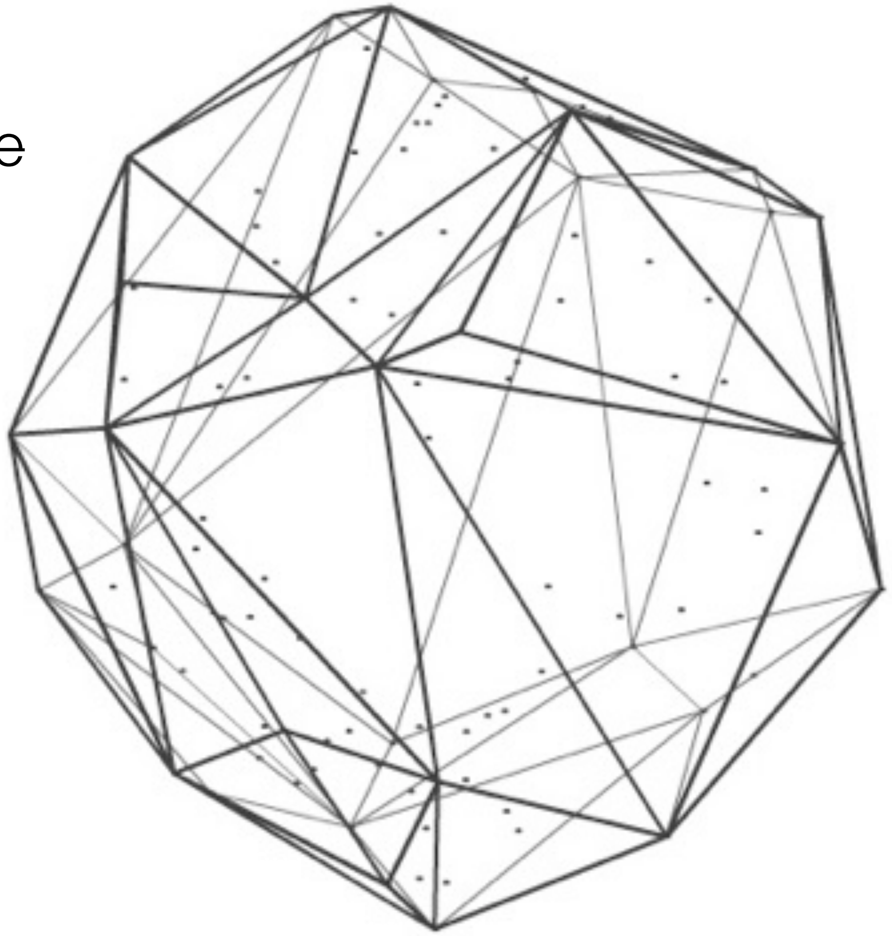
Properties of **2d** hull

- 2d hull consists of edges and vertices
- All edges of hull are extreme and all extreme edges of P are on the hull
- All points of hull are extreme and all extreme points of P are on the hull
- All internal angles are < 180
- Walking counterclockwise—> left turns
- Points on CH are sorted in radial order wrt a point inside

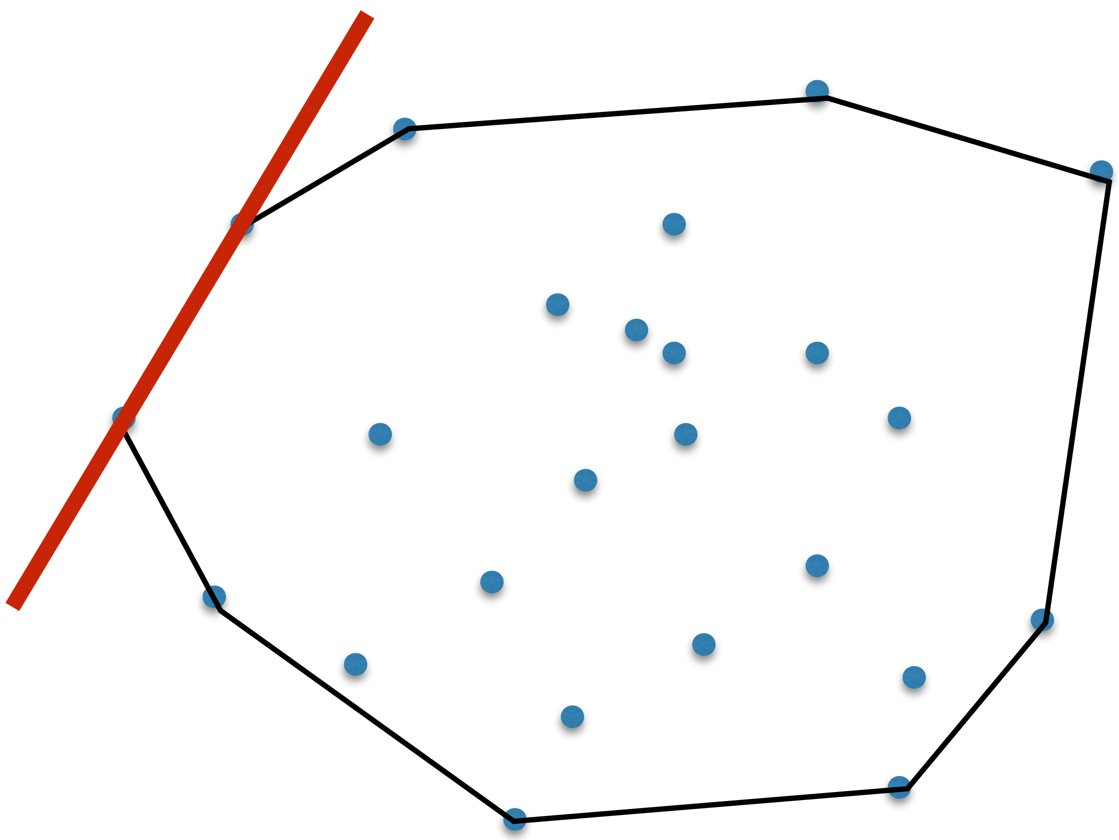


Properties of **3d** hull

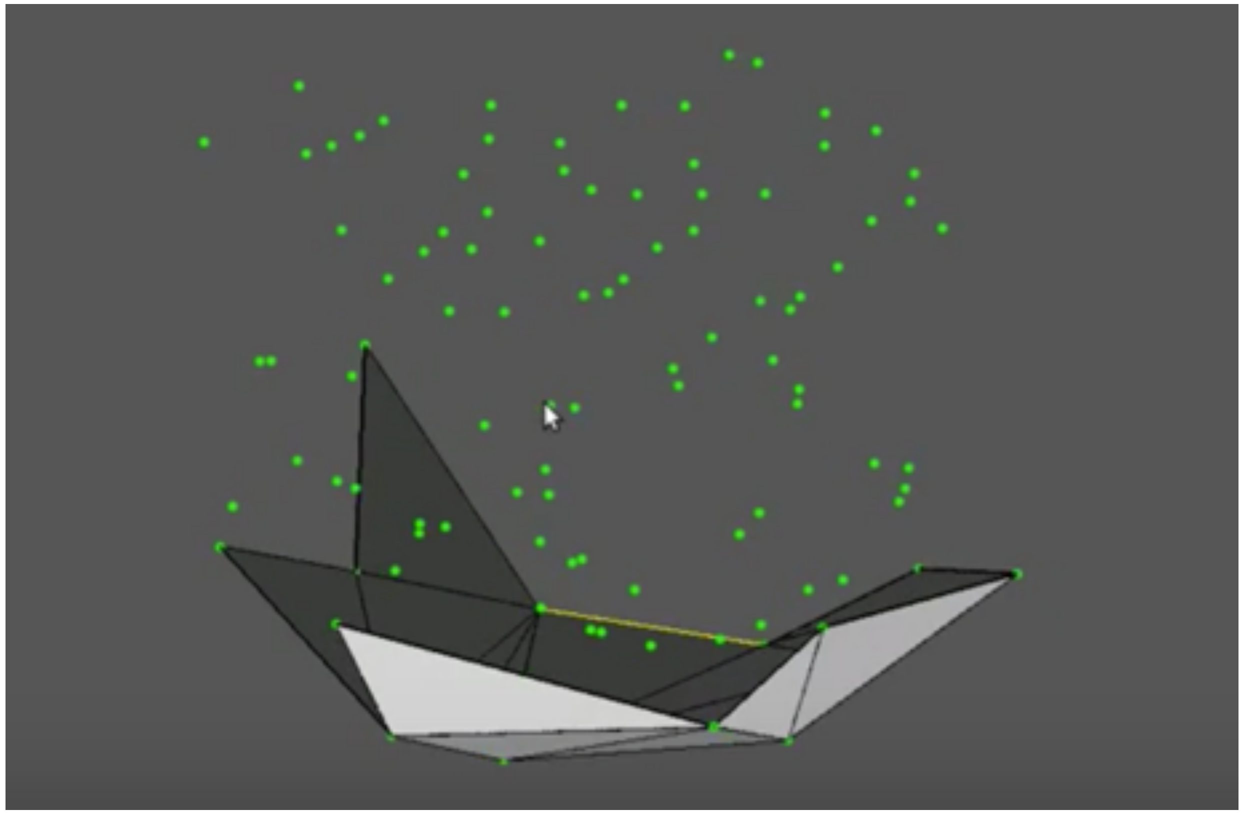
- 3d hull consists of: faces, edges and vertices
- All edges of hull are extreme and all extreme edges of P are on the hull
- All points of hull are extreme and all extreme points of P are on the hull
- All faces of hull are extreme and all extreme faces are on the hull
- All internal angles between faces are < 180
- ~~Walking counterclockwise —> left turns~~
- ~~Points on CH are sorted in radial order wrt a point inside~~



Faces, edges, vertices on the hull are **extreme**.

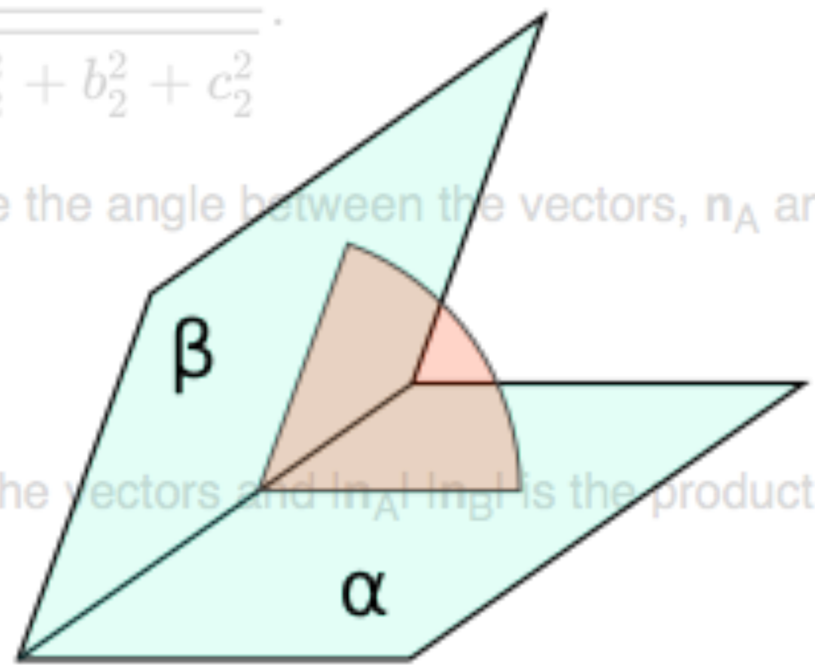


2D

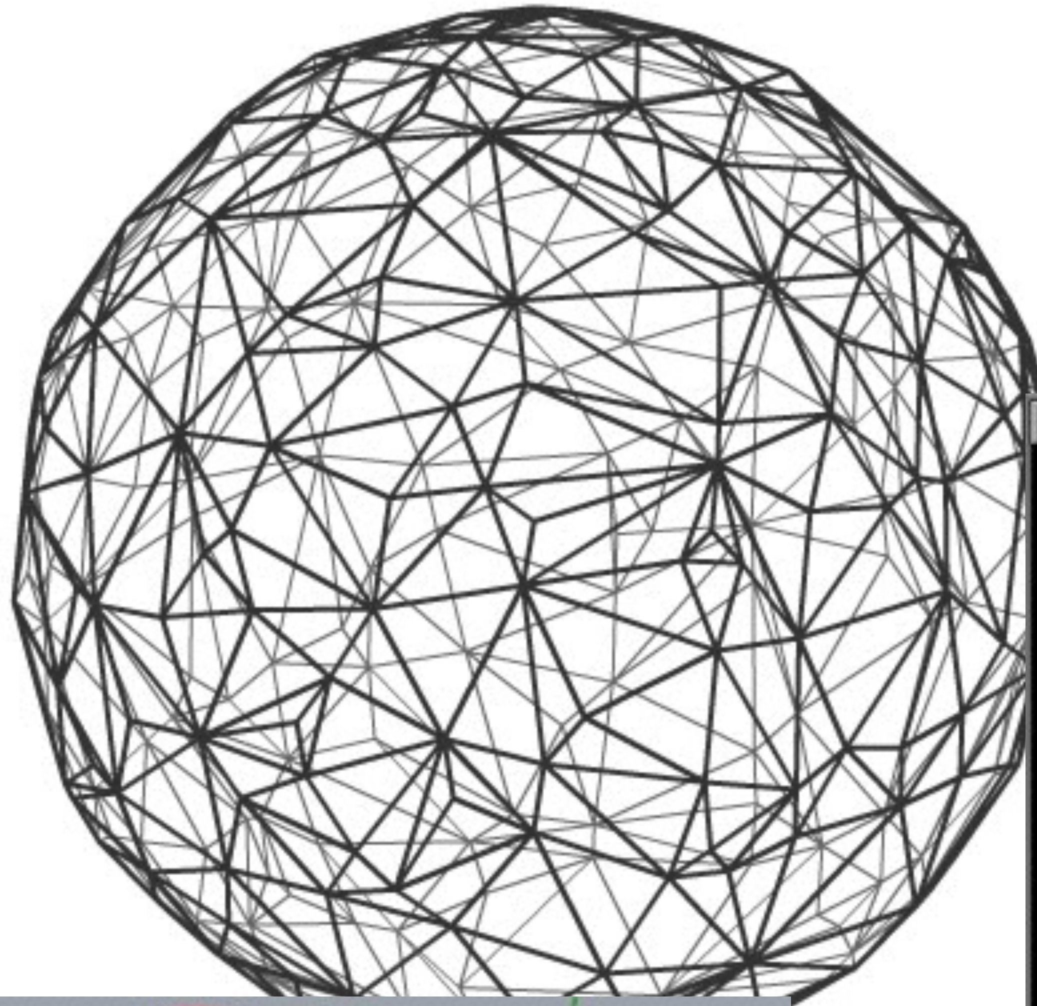


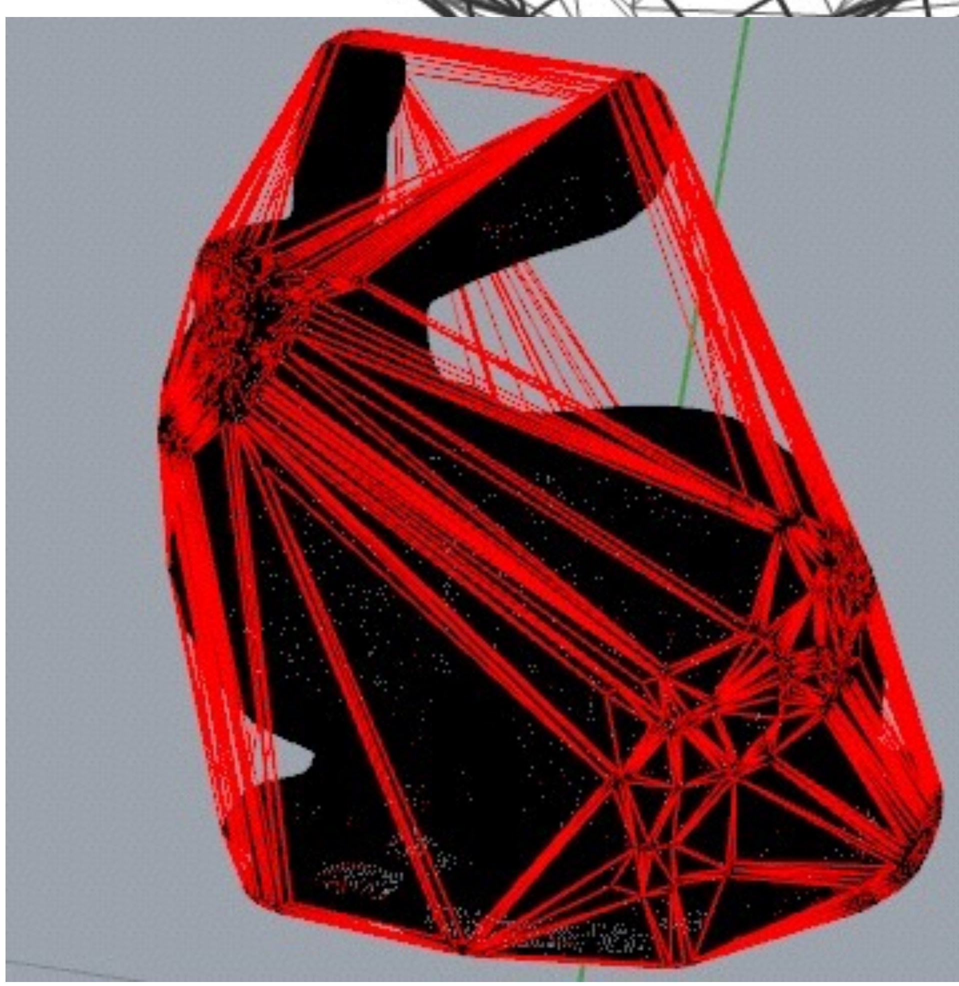
3D

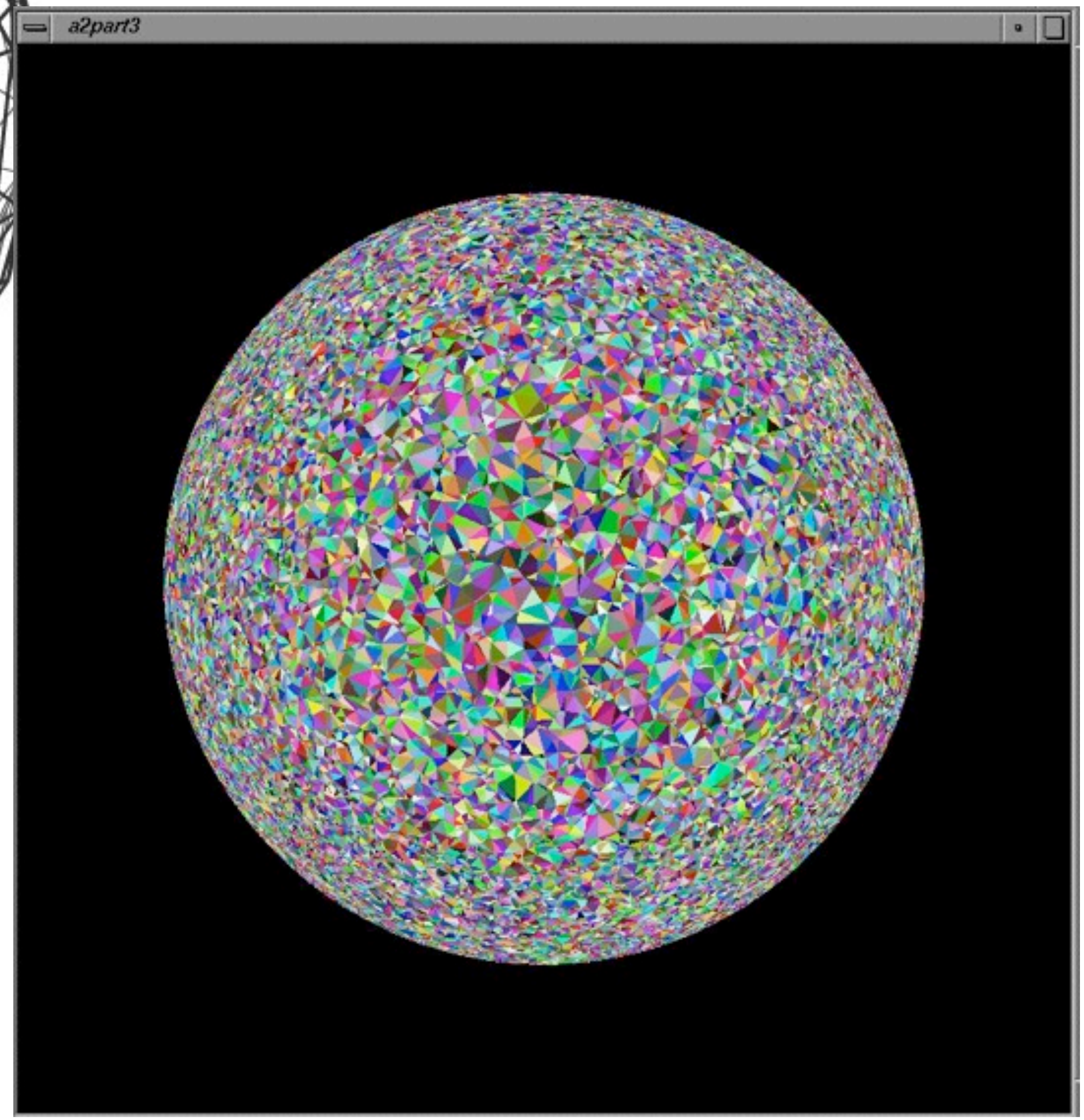
A **dihedral angle** is the angle between two intersecting planes.



Angle between two planes (α, β, green) in a third plane (pink) which cuts the line of intersection at right angles






a2part3

Computing the Hull

2D		3D
Naive	$O(n^3)$	
Gift wrapping	O(nh)	
Graham scan	O(n lg n)	Extend to 3D?
Quickhull	O(n lg n), $O(n^2)$	
Incremental	O(n lg n)	
Divide-and-conquer	O(n lg n)	

3d hull: Naive algorithm

Algorithm

- For every triplet of points (pi,pj,pk):
  - check if plane defined by it is extreme
  - if it is, add it to the list of CH faces

- Briefly sketch how to determine if a triplet is extreme and analyze it

```
is_extreme(point3d a, point3d b, point3d c,  vector<point3d> P)
```

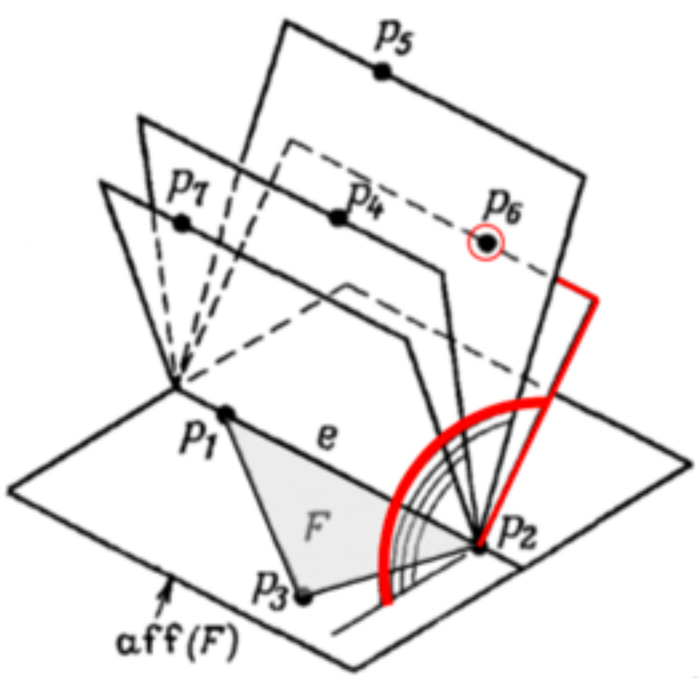
- 3d hull naive: running time?

3d hull: Gift wrapping

Algorithm

- find a face guaranteed to be on the CH
- REPEAT
  - find an edge e of a face f that's on the CH, and such that the face on the other side of e has not been found.
  - for all remaining points pi, find the angle of (e,pi) with f
  - find point pi with the minimal angle; add face (e,pi) to CH

- Analysis: O(n x F), where F is the number of faces on CH



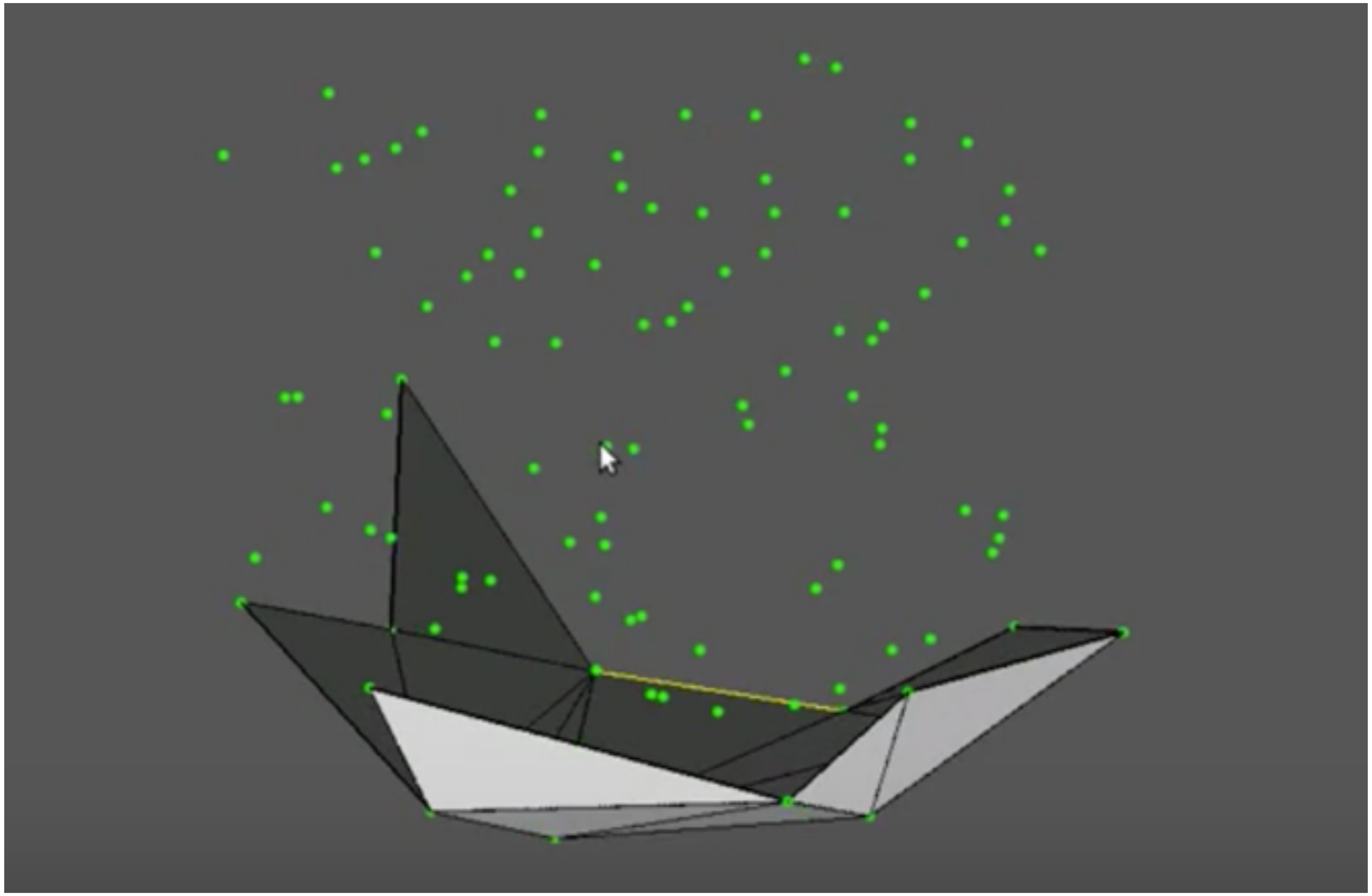
3d hull: Gift wrapping

Algorithm

- find a face guaranteed to be on the CH
- REPEAT
    - find an edge e of a face f that's on the CH, and such that the face on the other side of e has not been found.
    - for all remaining points pi, find the angle of (e,pi) with f
    - find point pi with the minimal angle; add face (e,pi) to CH

- To think
    - finding first face?
    - How to keep track of the hull? we'll need to store the connectivity (what faces are adjacent, for an edge which faces its adjacent to, etc)
    - How to keep track of the boundary of the hull (the edges that have only one face discovered)?

Gift wrapping in 3D



- YouTube
  - Video of CH in 3D (by Lucas Benevides)
  - Fast 3D convex hull algorithms with CGAL

From 2D to 3D

2D		3D
Naive	$O(n^3)$	$O(n^4)$
Gift wrapping	O(nh)	$O(n \times F)$
Graham scan	O(n lg n)	does not extend to 3D
Quickhull	O(n lg n), $O(n^2)$	
Incremental	O(n lg n)	
Divide-and-conquer	O(n lg n)	

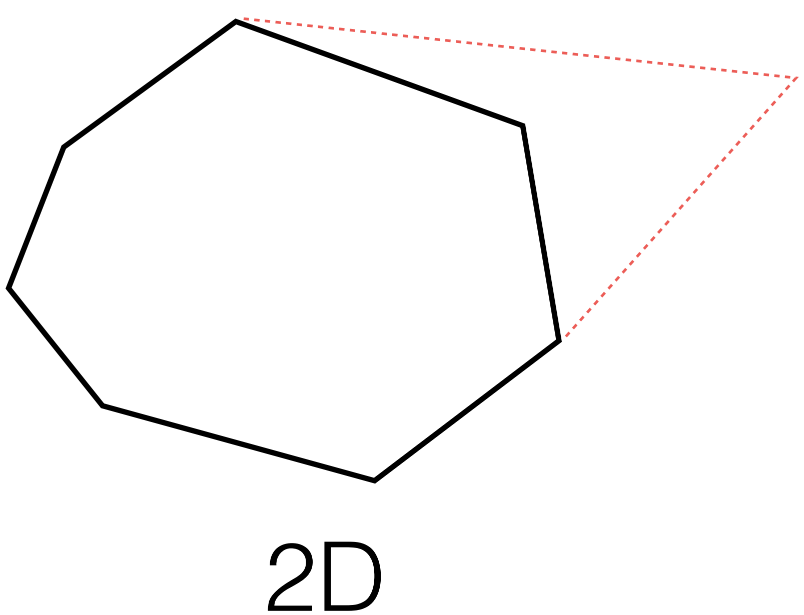
From 2D to 3D

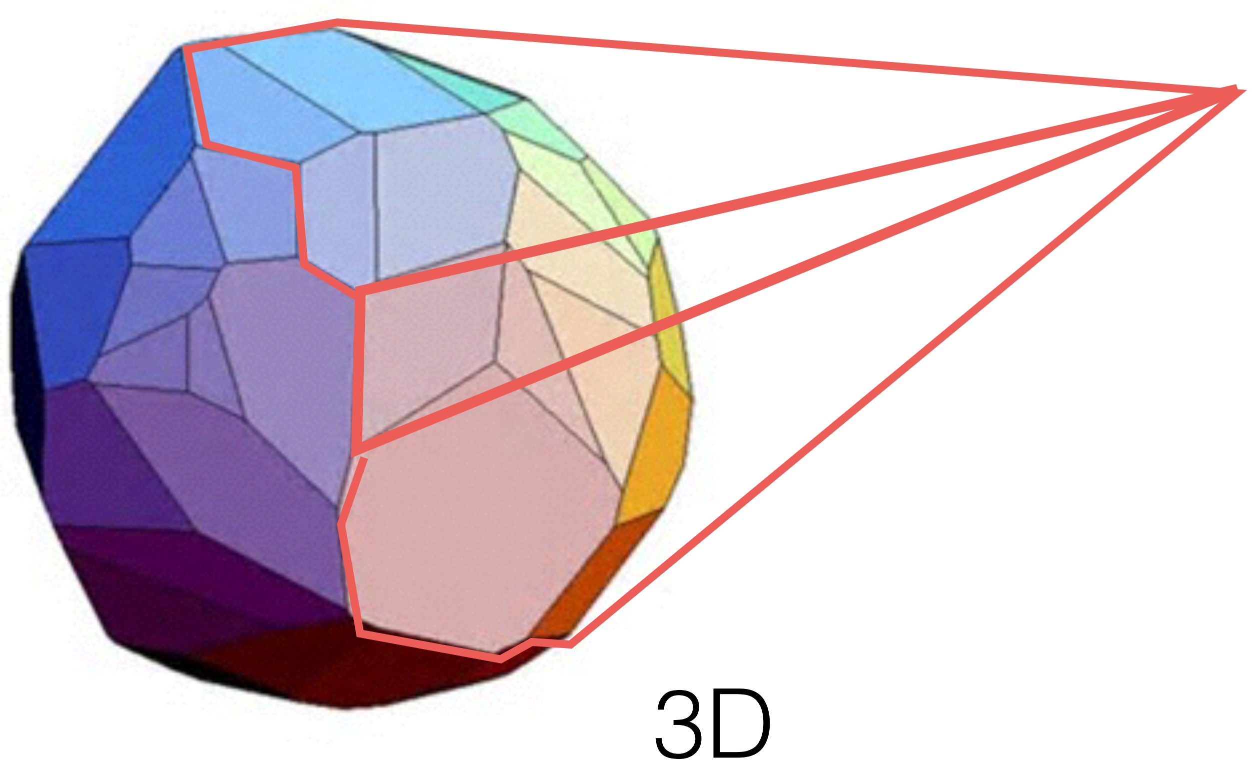
2D		3D
Naive	$O(n^3)$	$O(n^4)$
Gift wrapping	$O(nh)$	$O(n \times F)$
Graham scan	$O(n \lg n)$	does not extend to 3D
Quickhull	$O(n \lg n)$, $O(n^2)$	yes
Incremental	$O(n \lg n)$	$O(n^2)$
Divide-and-conquer	$O(n \lg n)$	$O(n \lg n)$

Incremental 3D hull

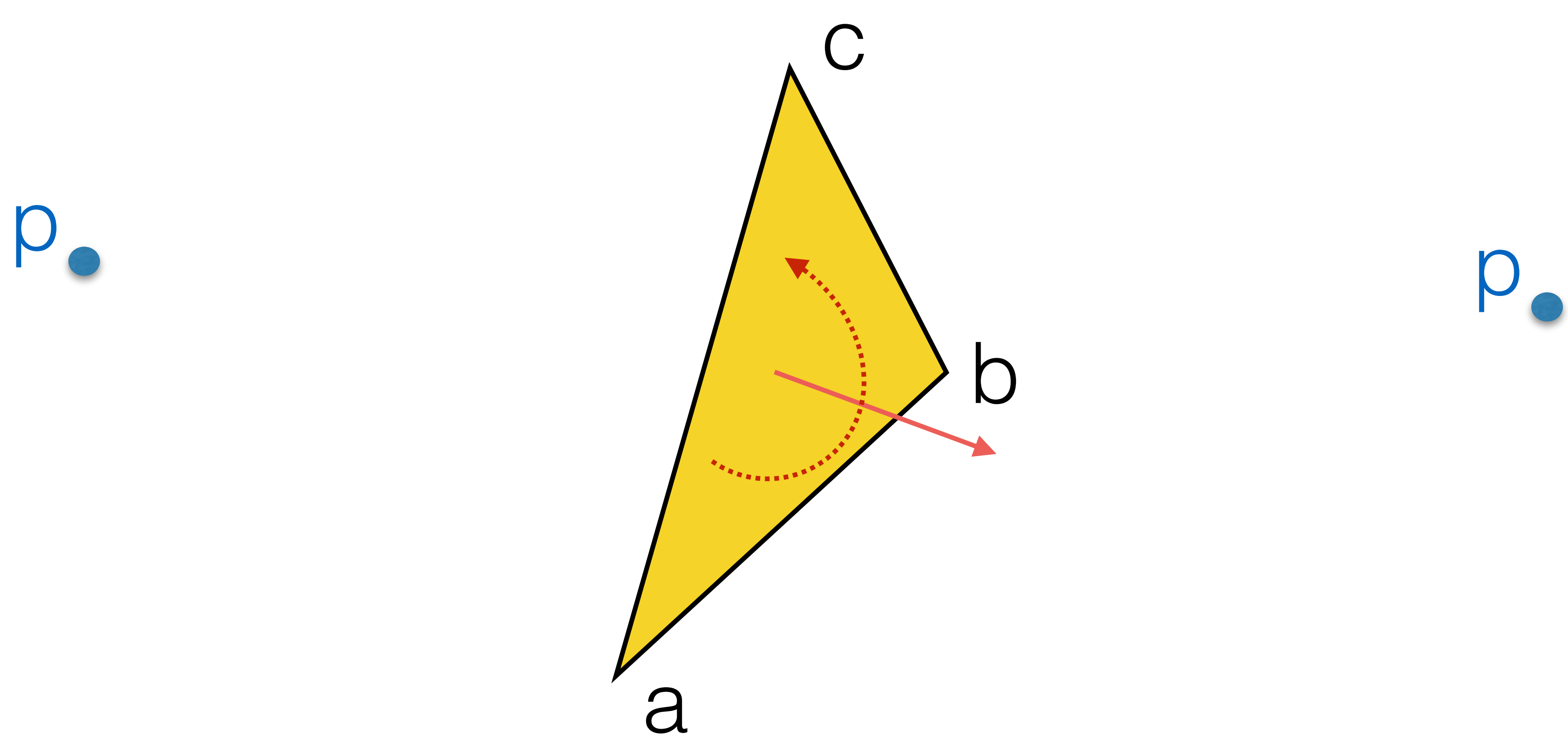
Incremental

- CH = {p1,p2,p3}
- for i= 4 to n
    - //CH represents the CH of $p_1..p_{i-1}$
    - add p_i to CH and update CH to represent the CH of $p_1..p_i$





Terminology: Point in front/behind face



ps is left of (behind) abc

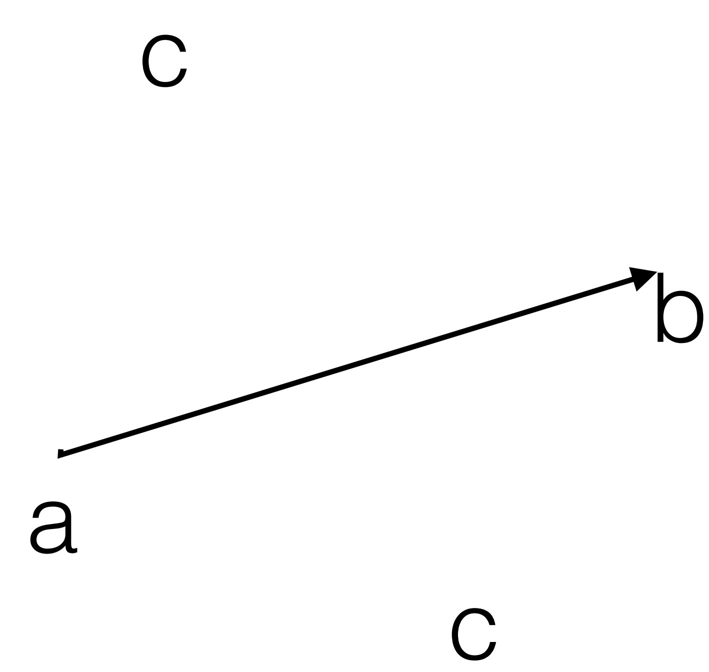
abc not visible from p

p is right of (in front) abc

abc visible from p

2D

$$2\ \text{signedArea}(a,b,c) = \det \begin{vmatrix} a.x & a.y & 1 \\ b.x & b.y & 1 \\ c.x & c.y & 1 \end{vmatrix}$$



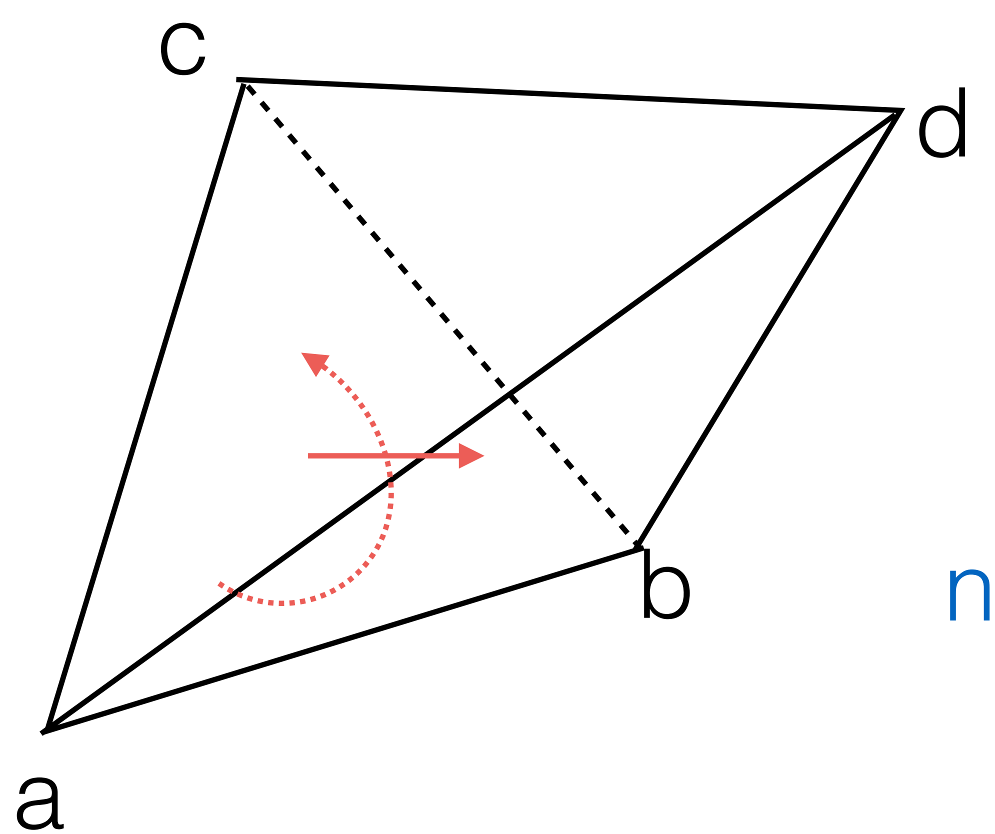
positive area
(c left/behind ab)

negative area
(c right/in front of ab)

3D

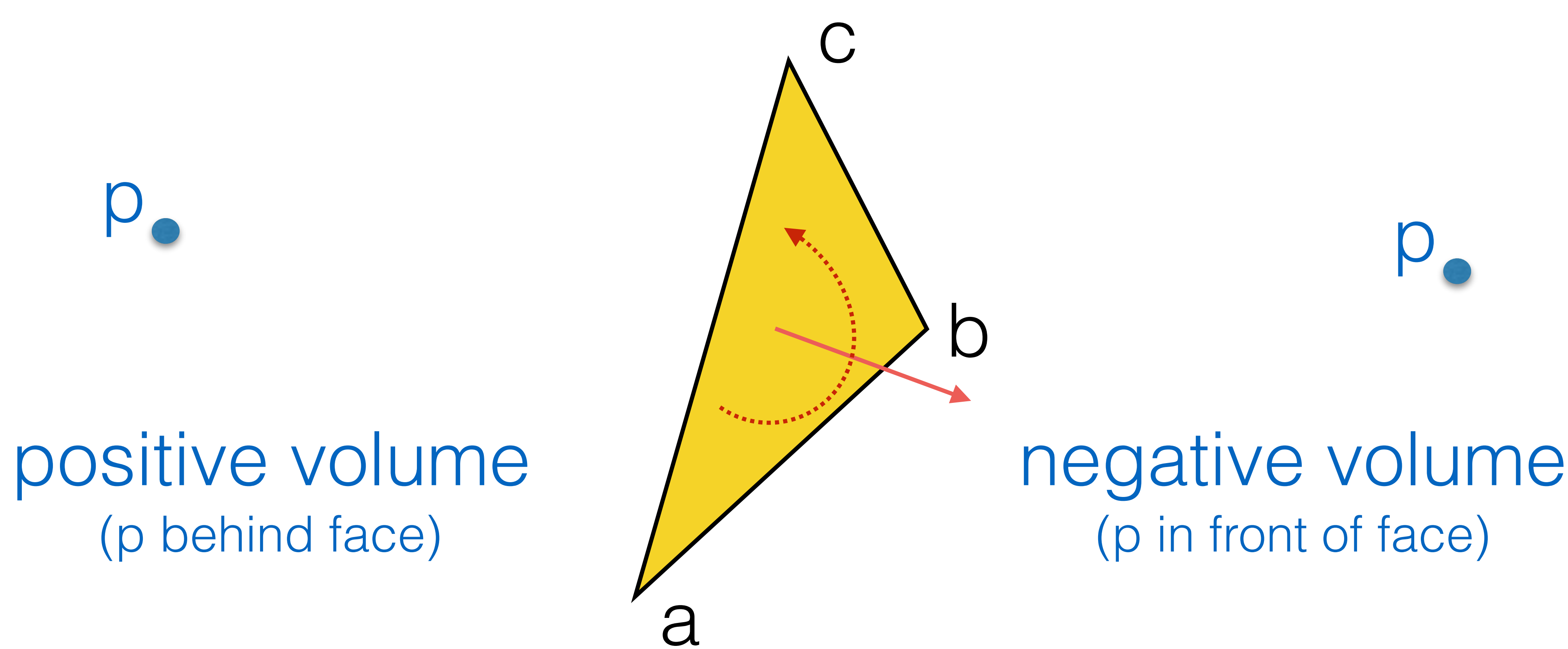
$$6\ \text{signedVolume}(a,b,c,d) = \det \begin{vmatrix} a.x & a.y & a.z & 1 \\ b.x & b.y & b.z & 1 \\ c.x & c.y & c.z & 1 \\ d.x & d.y & d.z & 1 \end{vmatrix}$$

positive volume
(p behind face)



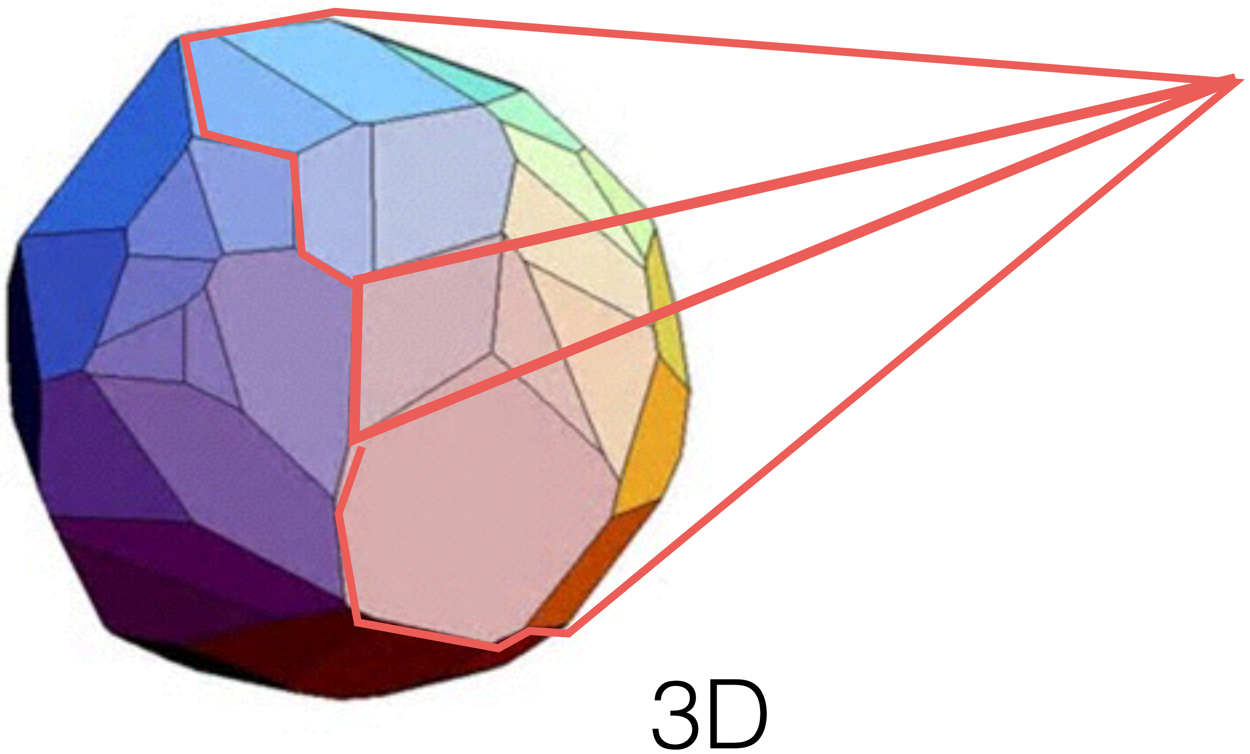
negative volume
(d in front of face)

- Assume all faces oriented counterclockwise so that their normals determined by the right-hand rule point towards the **outside** of P.



is_visible(a,b,c,p): return signedVolume(a,b,c,p) < 0

Incremental



The visible faces are precisely those that need to be discarded
The edges on the boundary of the visible region are the basis of the cone

Incremental

Algorithm: incremental hull 3d

- initialize H = p1, p2, p3, p4
- for i = 5 to n do:
  - for each face f of H do:
    - compute volume of tetrahedron formed by (f,pi)
    - if volume < 0: f is visible
  - if no faces are visible
    - discard pi (pi must be inside H)
  - else
    - find border edge of all visible faces
    - for each border edge e construct a face (e,pi) and add to H
    - for each visible face f: delete f from H

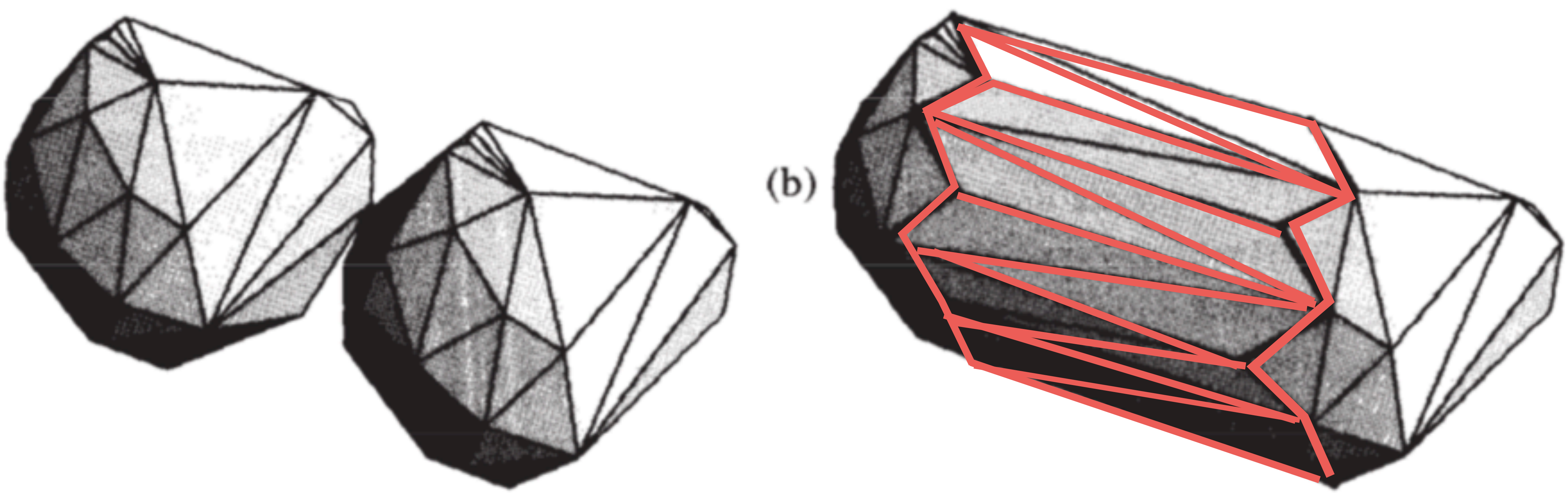
3d hull: divide & conquer

The same idea as 2D algorithm

- divide points in two halves P1 and P2
- recursively find CH(P1) and CH(P2)
- merge

- If merge in O(n) time ==> O(n lg n) algorithm

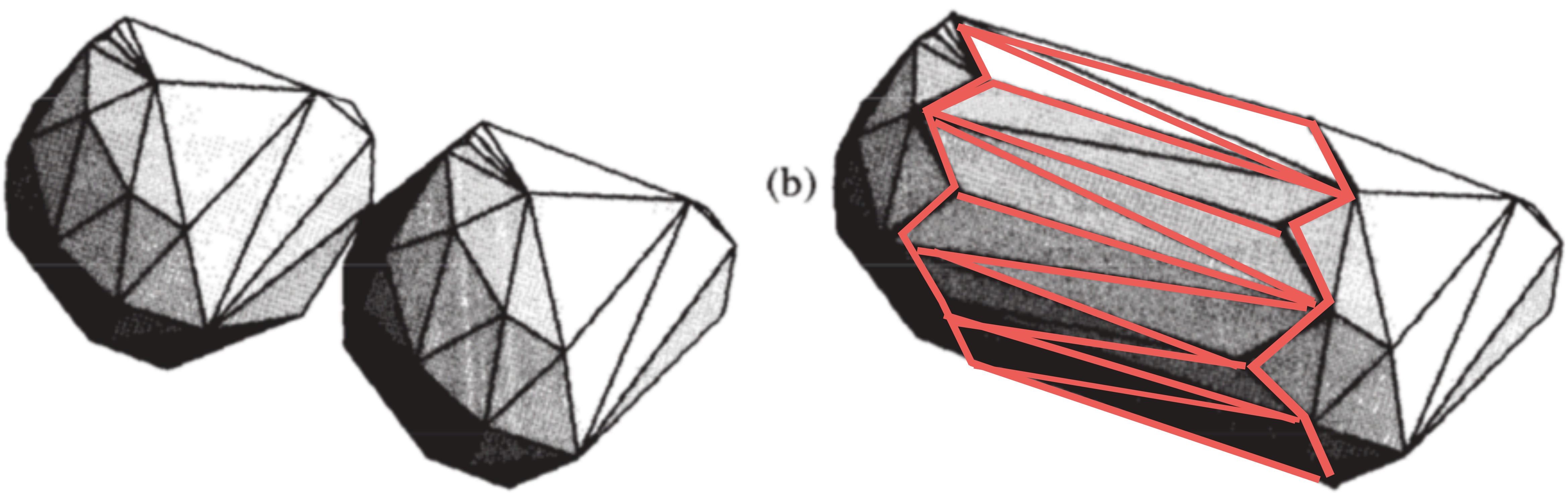
Merge



Merged hull: cylinder without end caps

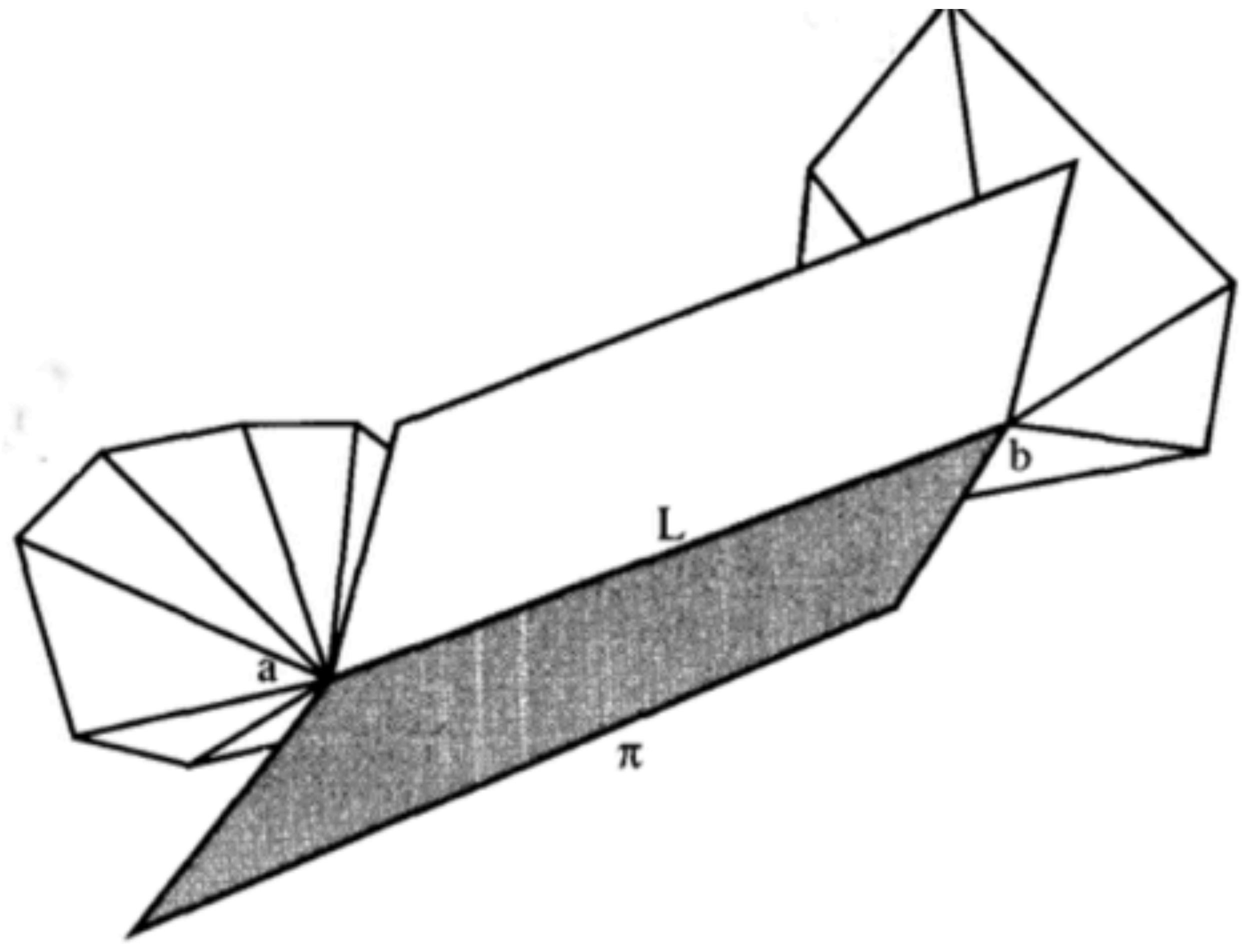
Merge

- Idea: Start with the lower tangent, wrap around, find one face at a time.



Merge

- Let PI be a plane that supports the hull from below

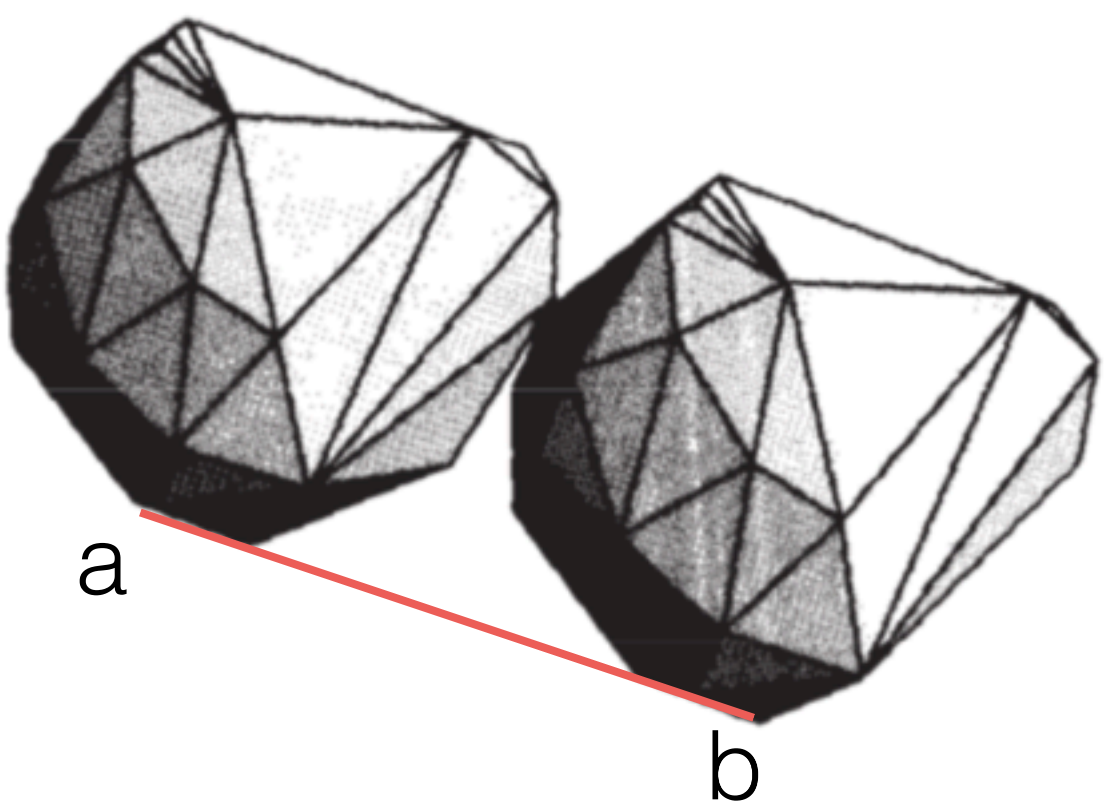


Claim:

- When we rotate PI around ab, the first vertex hit c must be a vertex adjacent to a or b
- c has the smallest angle among all neighbors of a,b

Merge

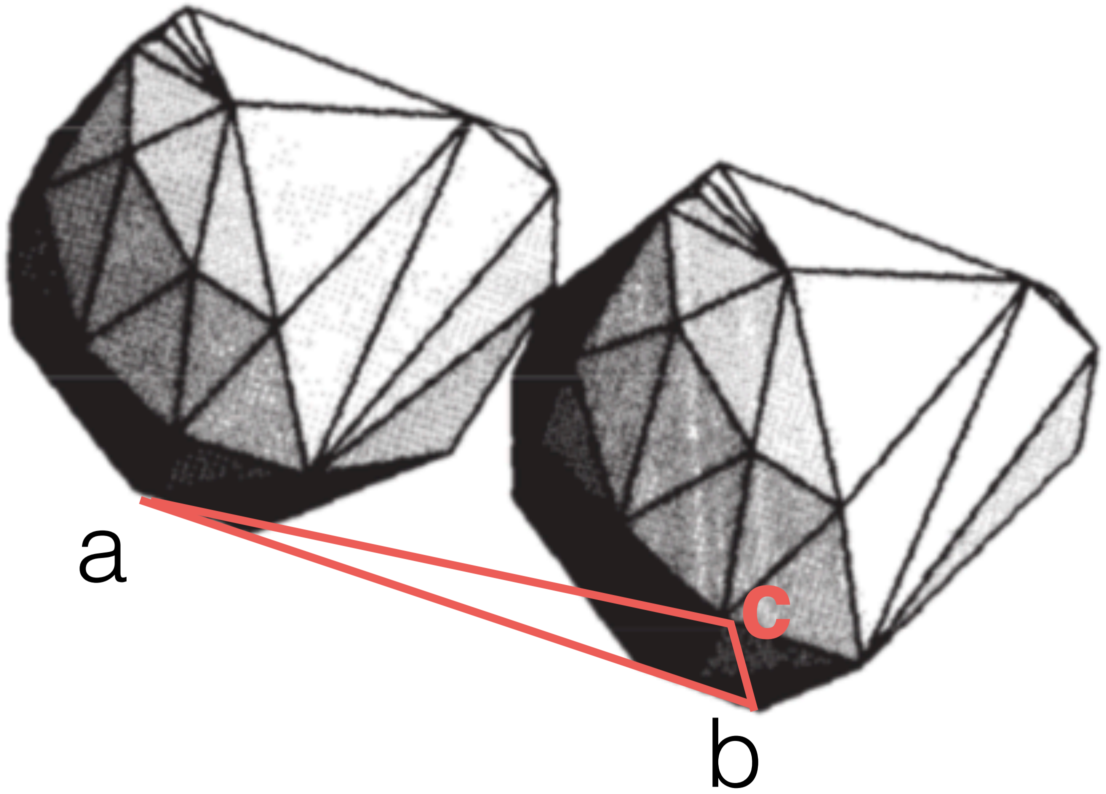
1. Find a common tangent ab



Merge

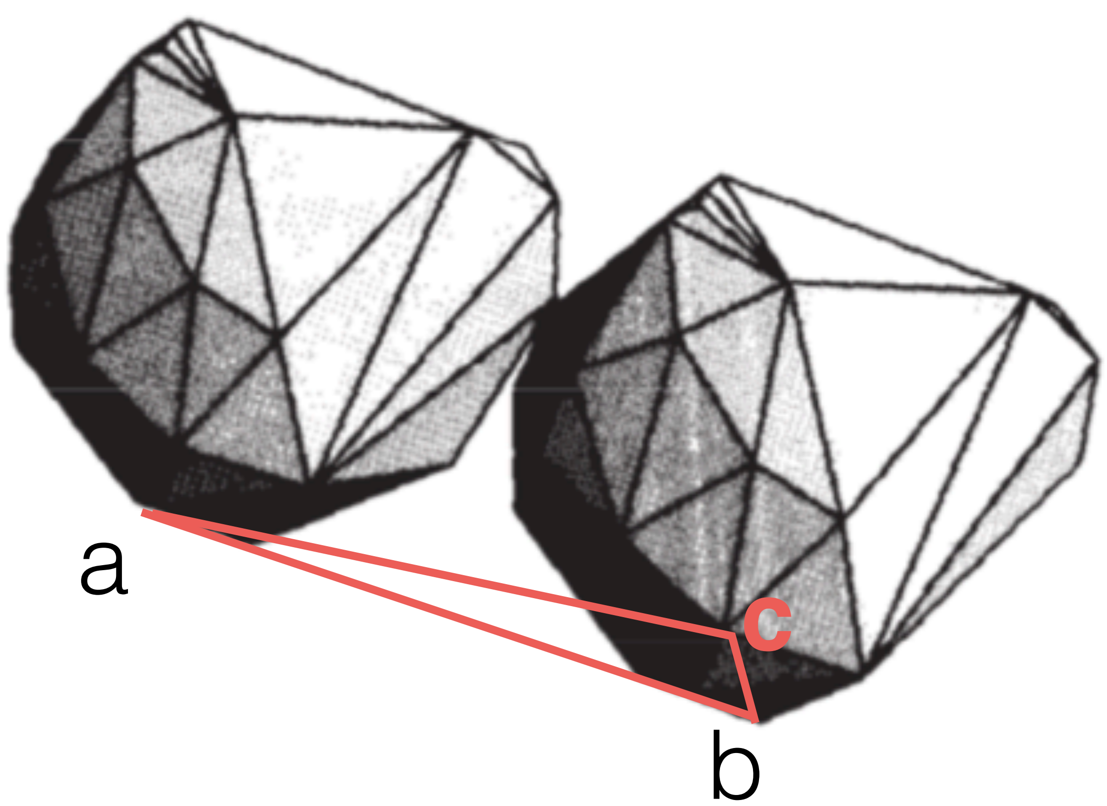
1. Find a common tangent ab

    - Now we need to find a triangle abc. If c is on the right hull, then bc is an edge on the right hull.



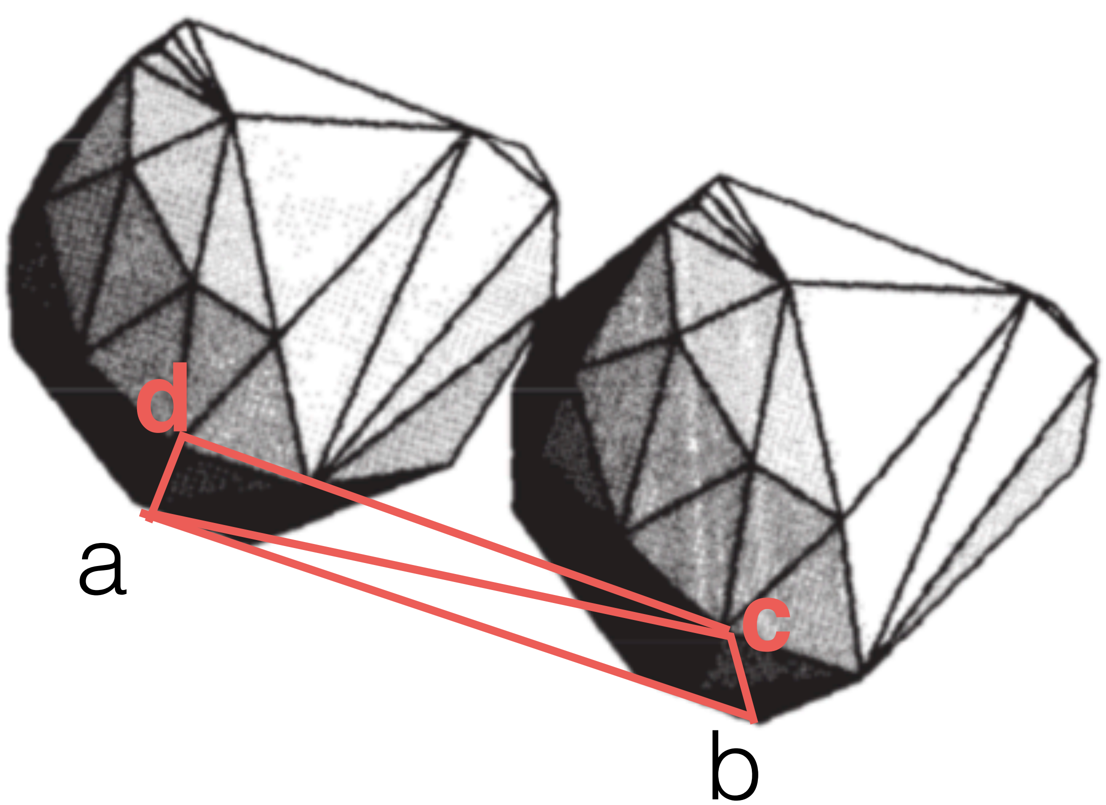
Merge

1. Find a common tangent ab
    - Now we need to find a triangle abc. If c is on the right hull, then bc is an edge on the right hull.
    - Now we have a new edge ac that's a tangent. Repeat.



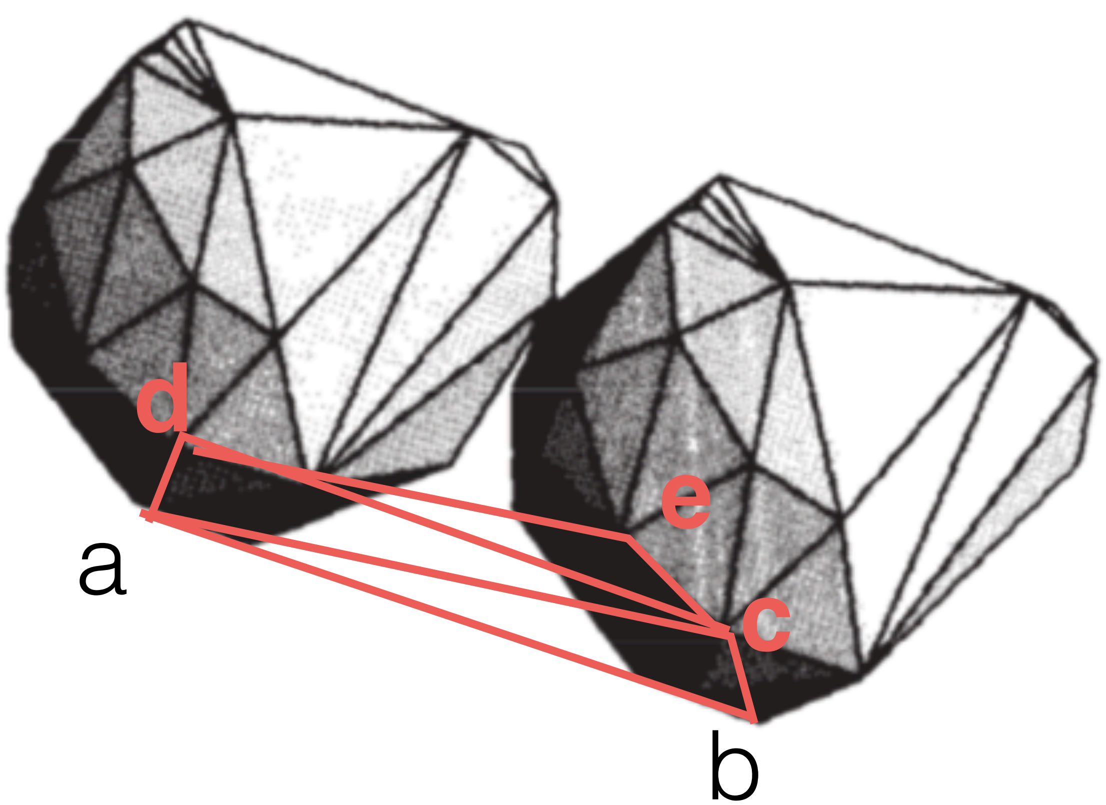
Merge

1. Find a common tangent ab
    - Now we need to find a triangle abc. If c is on the right hull, then bc is an edge on the right hull. .
    - Now we have a new edge ac that's a tangent. Repeat.



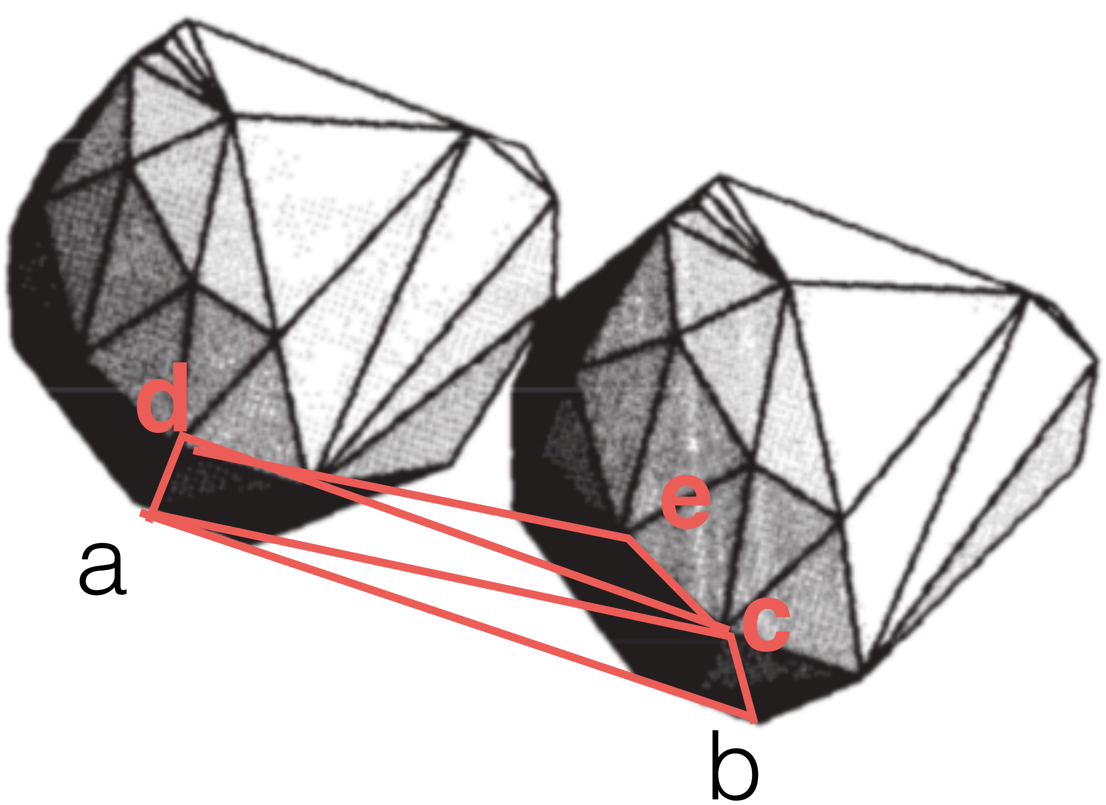
Merge

1. Find a common tangent ab
   - Now we need to find a triangle abc. If c is on the right hull, then bc is an edge on the right hull.
   - Now we have a new edge ac that's a tangent. Repeat.

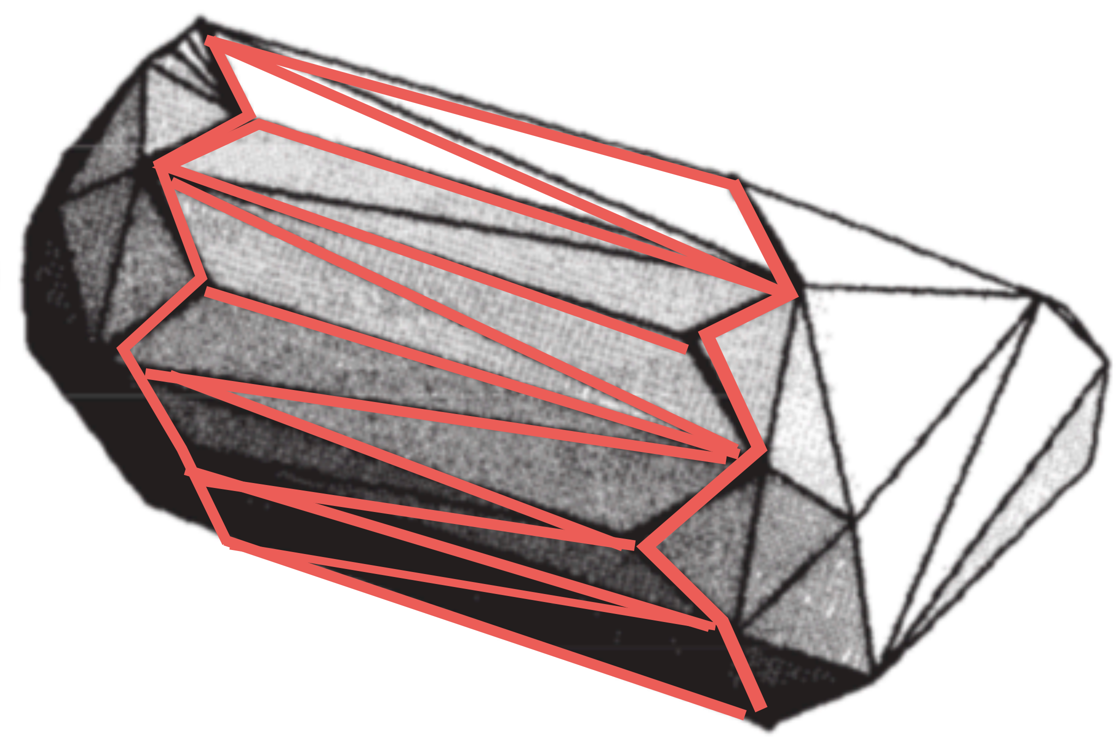


Merge

1. Find a common tangent ab
   - Now we need to find a triangle abc. If c is on the right hull, then bc is an edge on the right hull.
   - Now we have a new edge ac that's a tangent. Repeat.

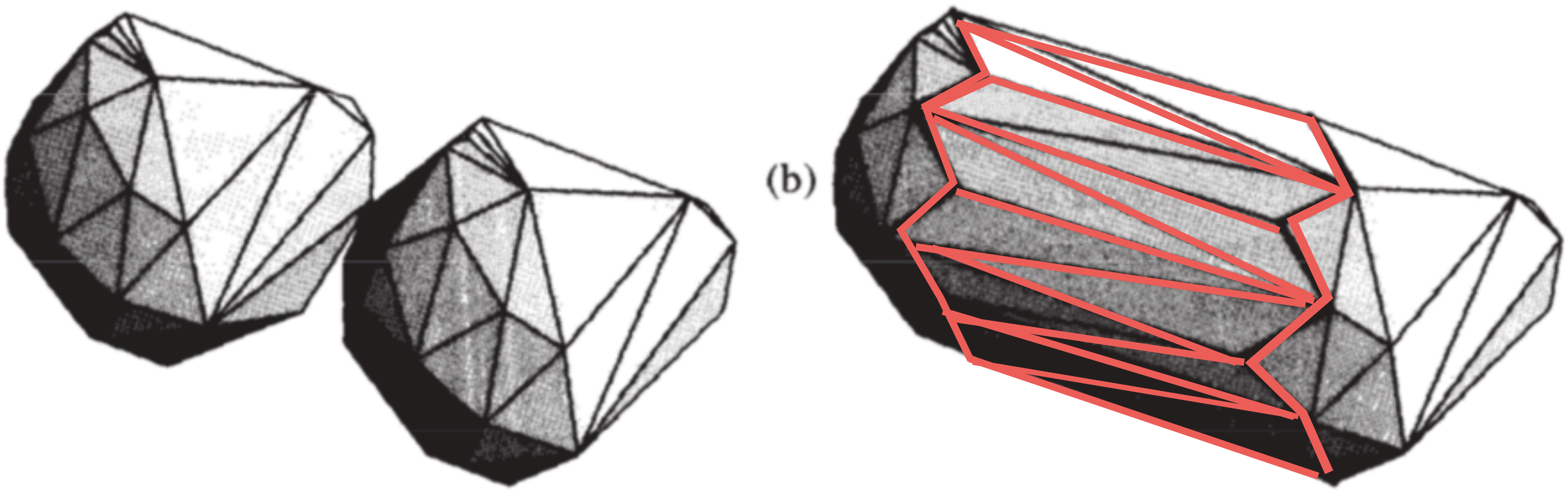


(b)

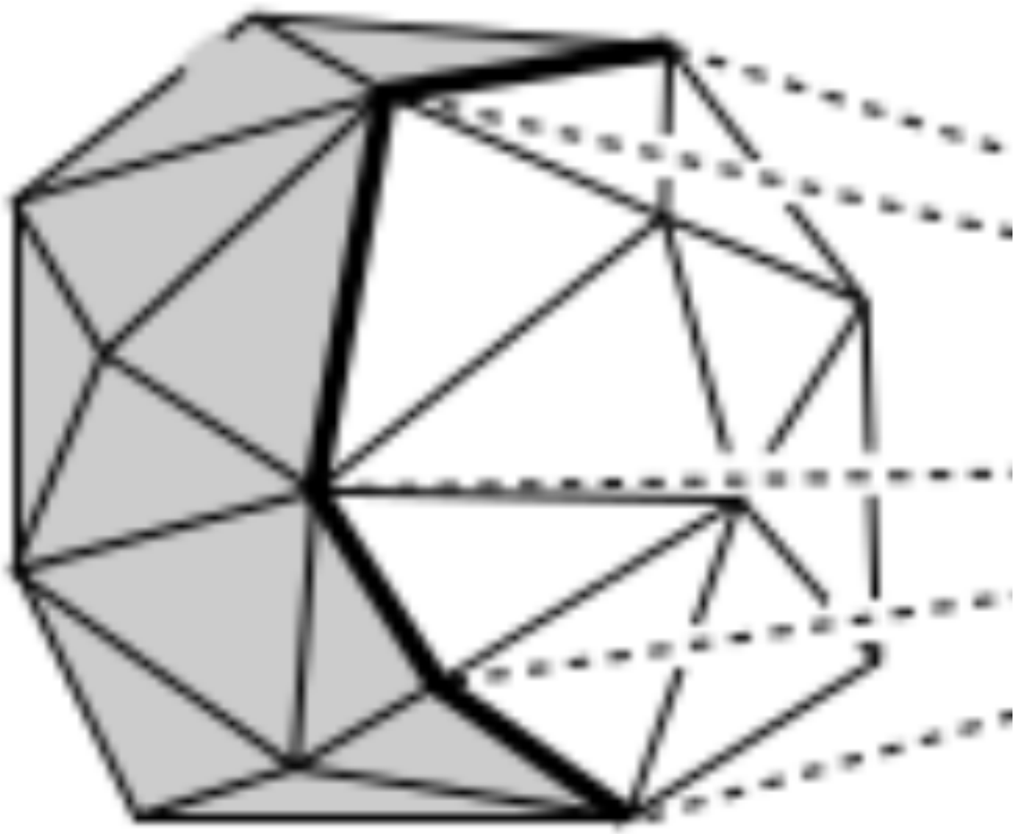


Merge

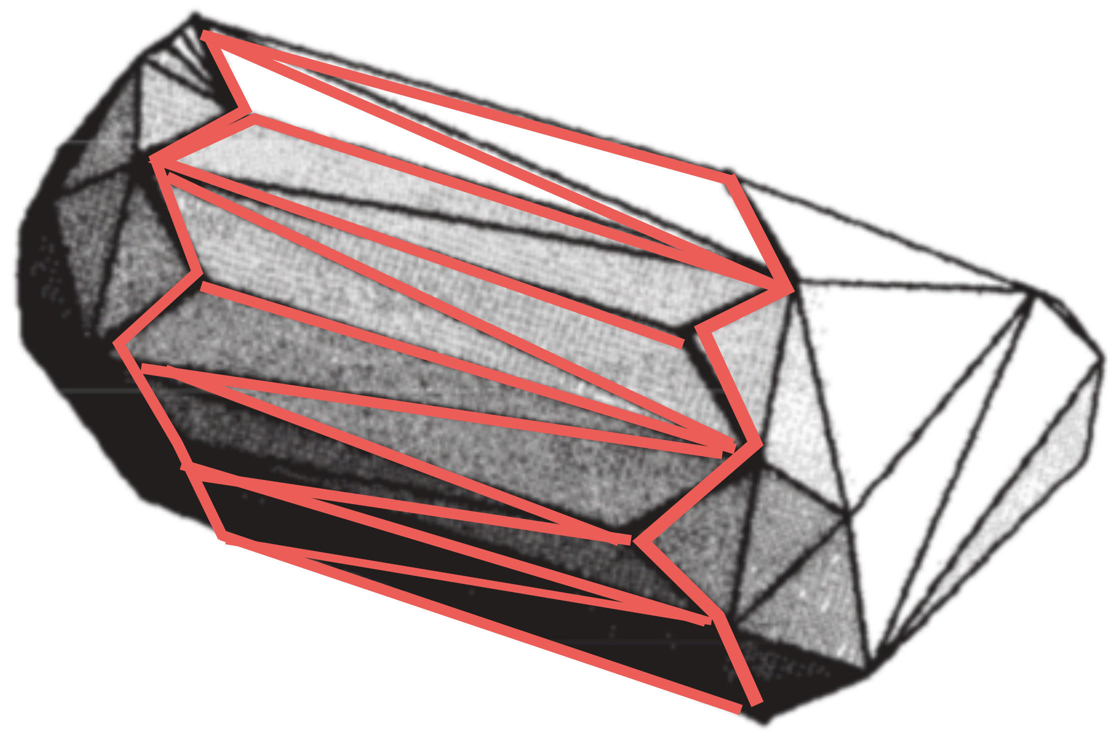
1. Find a common tangent ab
2. Start from ab and wrap around, to create the cylinder of triangles that connects the two hulls A and B
3. Find and delete the hidden faces that are “inside” the cylinder



The hidden faces



(b)



- start from the edges on the boundary of the cylinder
- BFS or DFS faces “towards” the cylinder
- all faces reached are inside

3d hull: divide & conquer

- Theoretically important and elegant
- Of all algorithms that extend to 3D, DC& is the only one that achieves optimal (n lg n)
- Difficult to implement
- The slower algorithms (quickhull, incremental) preferred in practice