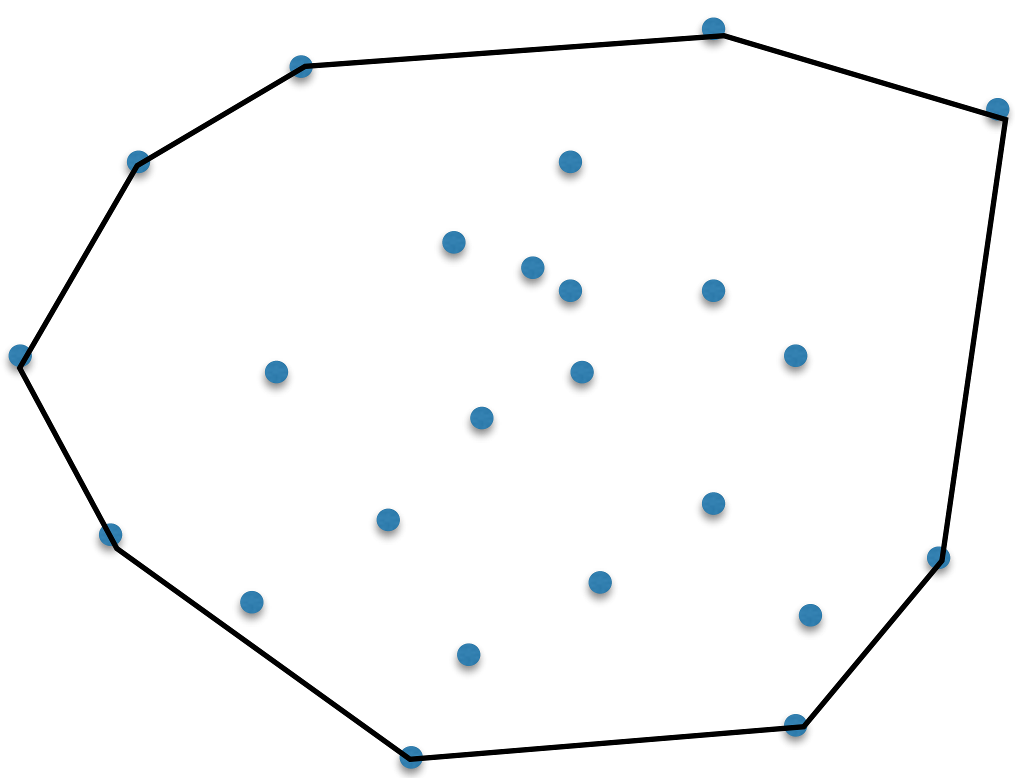
3D convex hulls

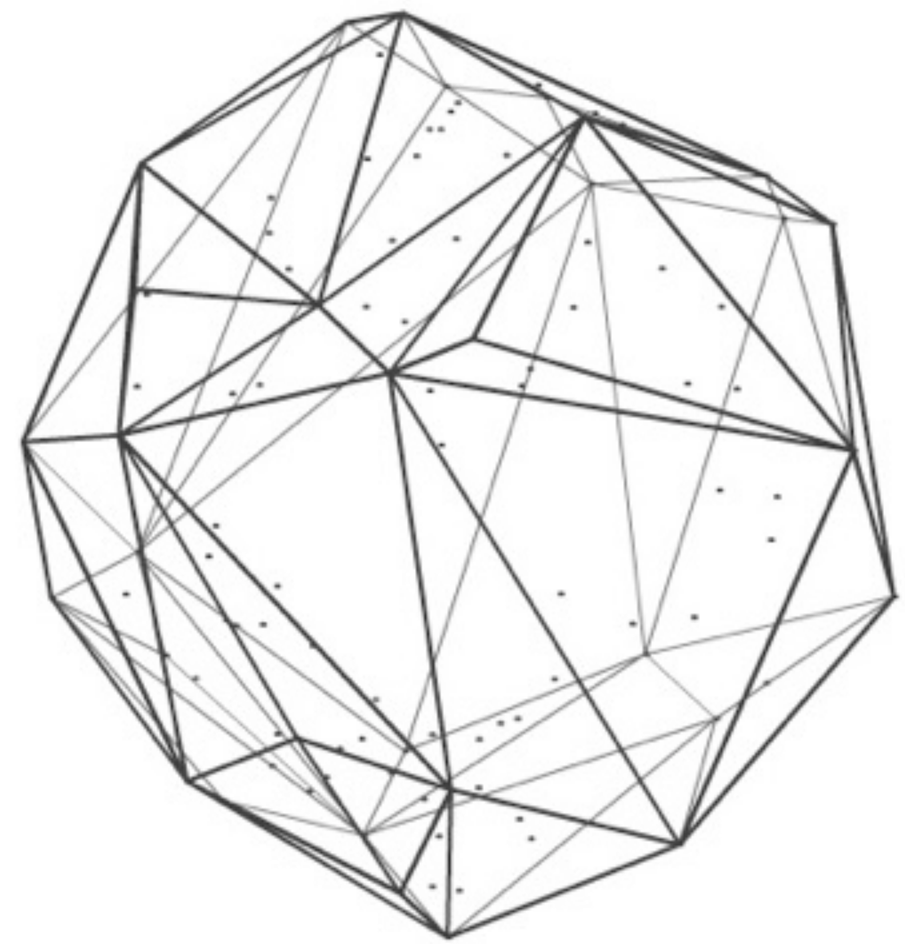
Computational Geometry [csci 3250]
Laura Toma
Bowdoin College

Convex Hull in 3D

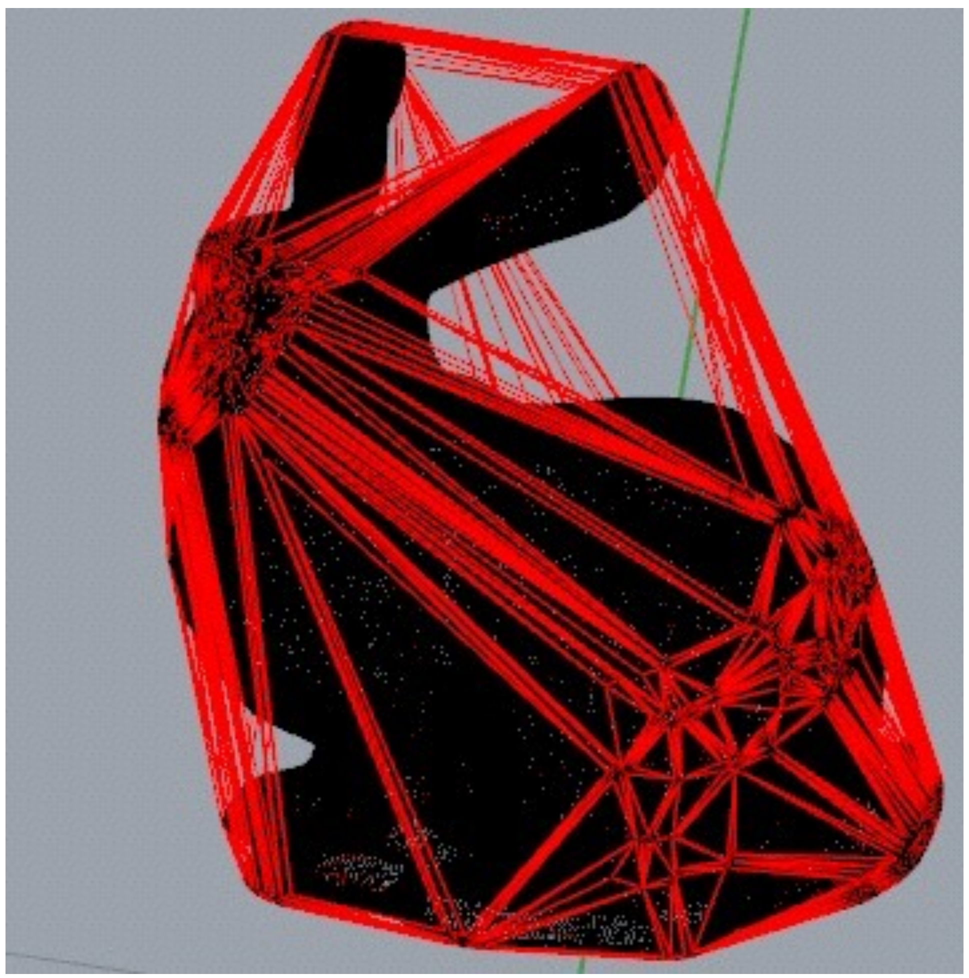
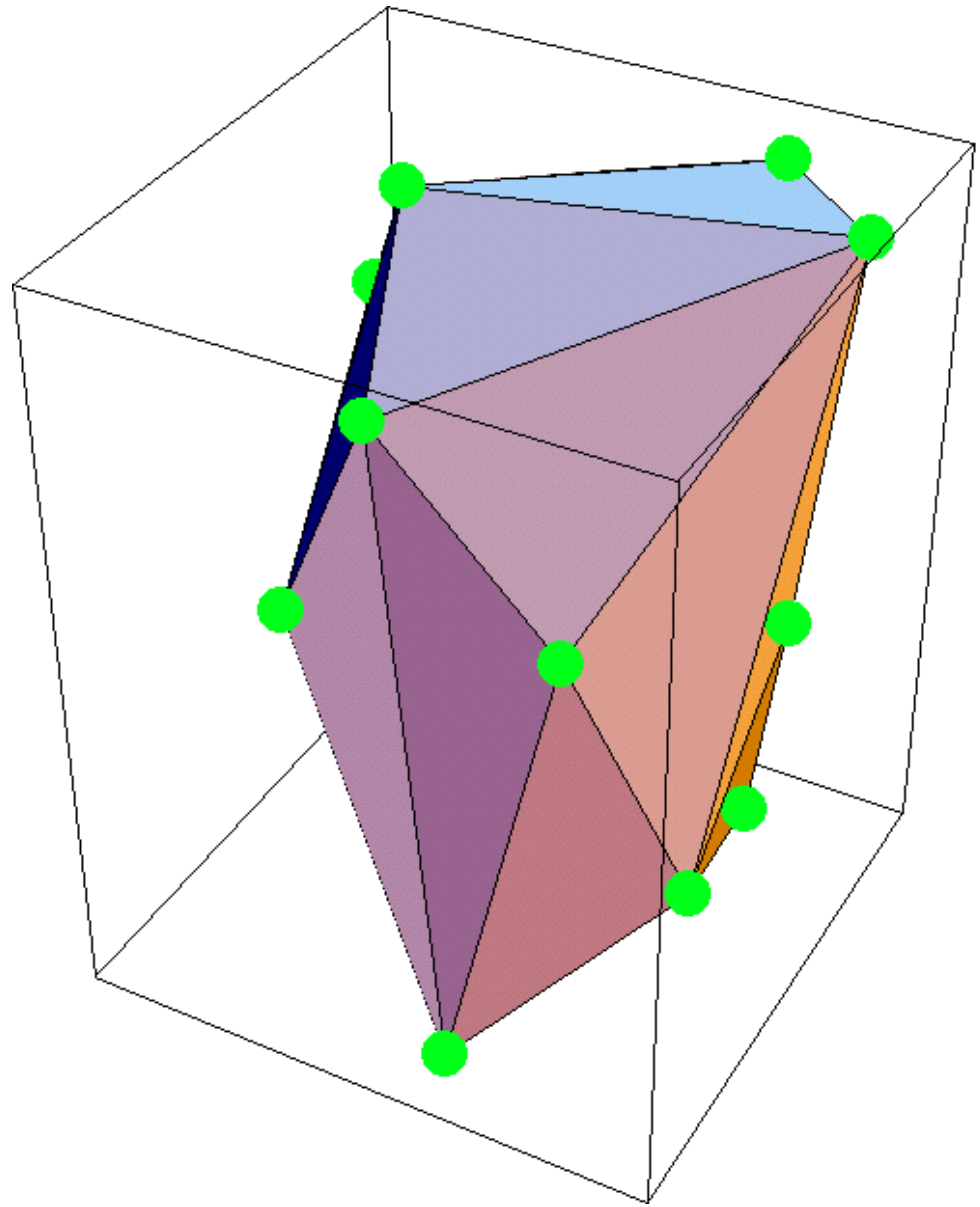
The problem: Given a set P of points in 3D, compute their convex hull



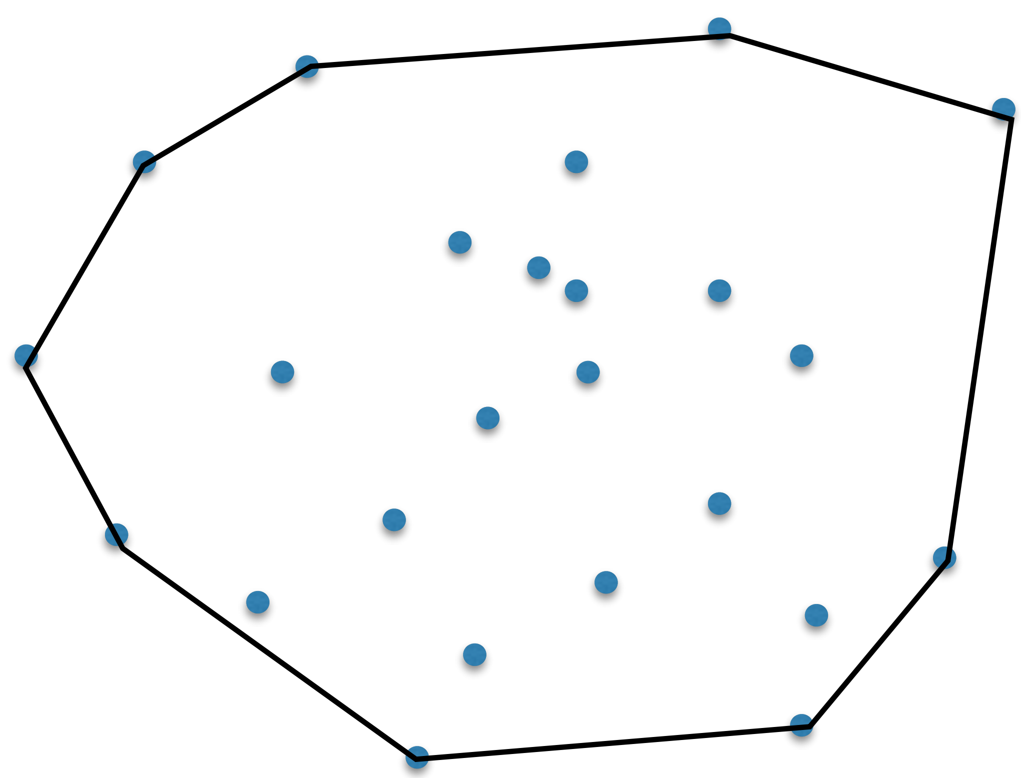
2D



3D

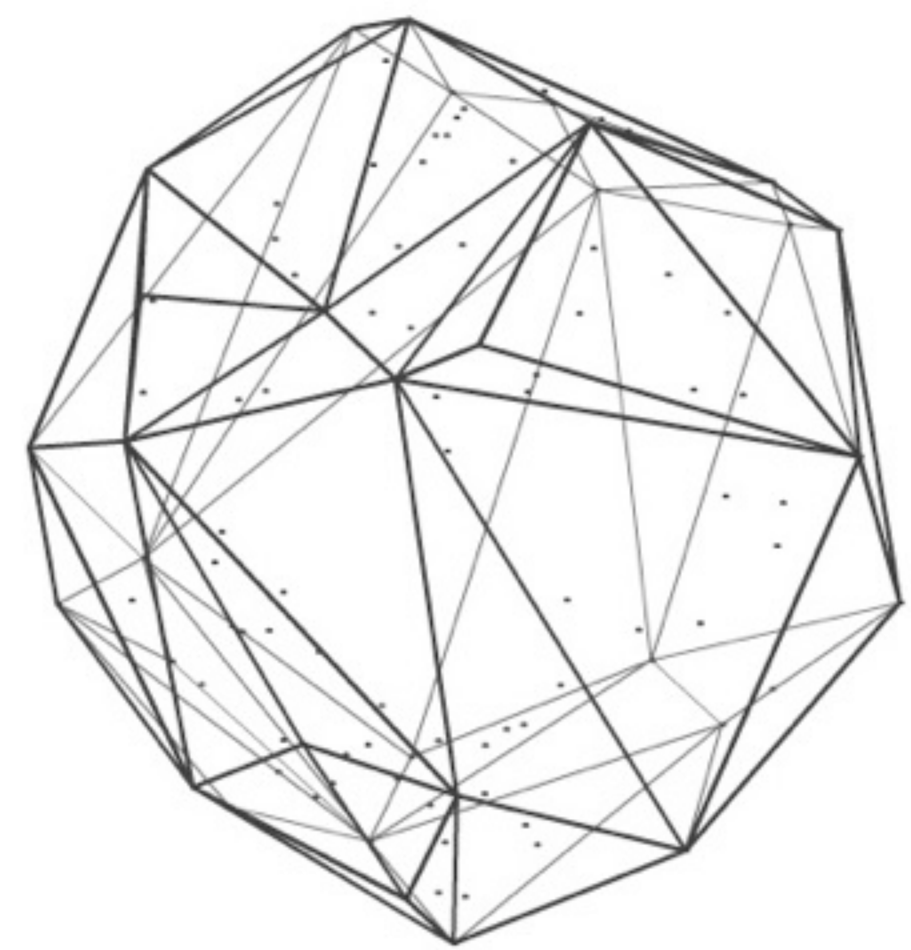



polygon



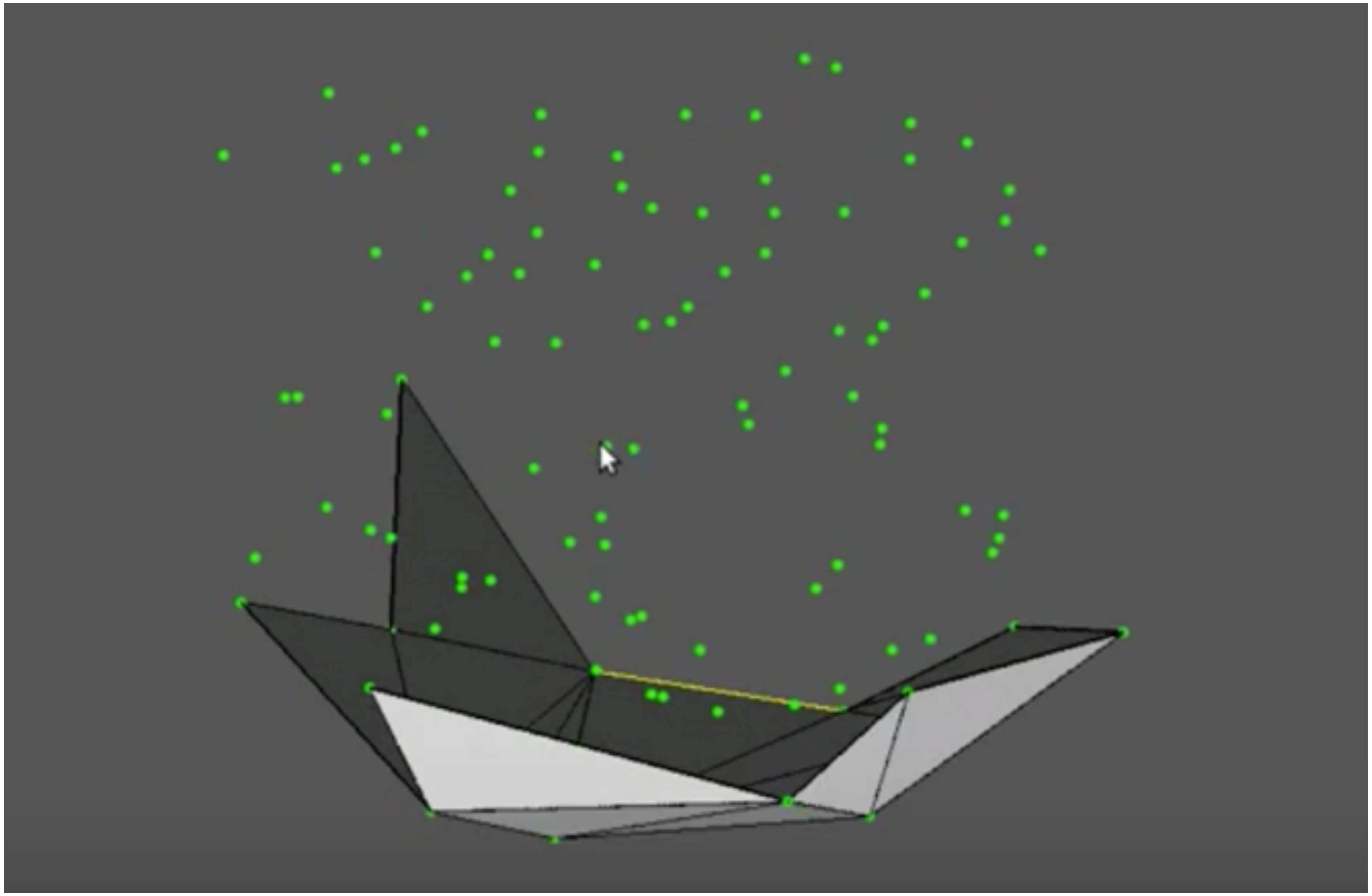
2D

polyhedron



3D

Gift wrapping in 3D



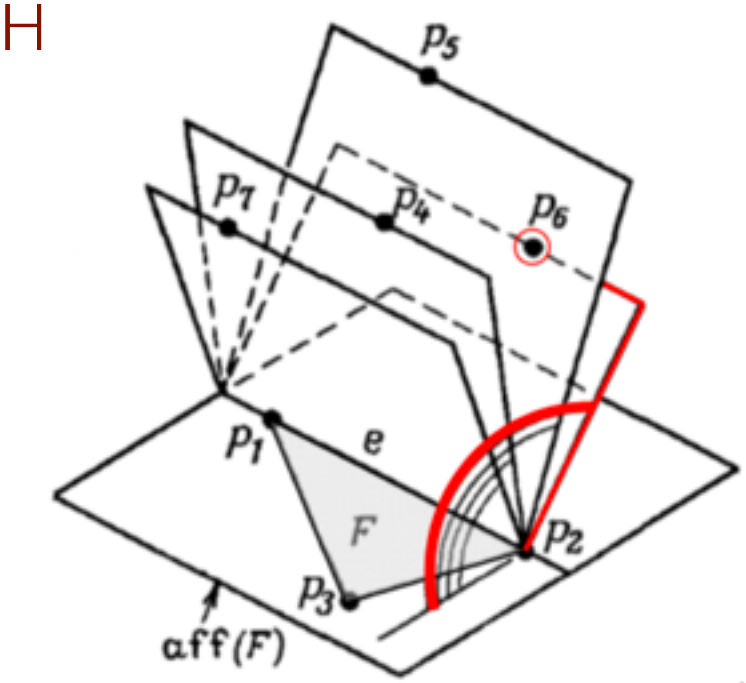
- YouTube
  - Video of CH in 3D (by Lucas Benevides)
  - Fast 3D convex hull algorithms with CGAL

Gift wrapping in 3D

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



Gift wrapping in 3D

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

- Implementation details
  - sketch more detailed pseudocode
  - finding first face?
  - what data structures do you need? how to keep track of vertices, edges, faces? how to store the connectivity of faces?

3d hull: divide & conquer

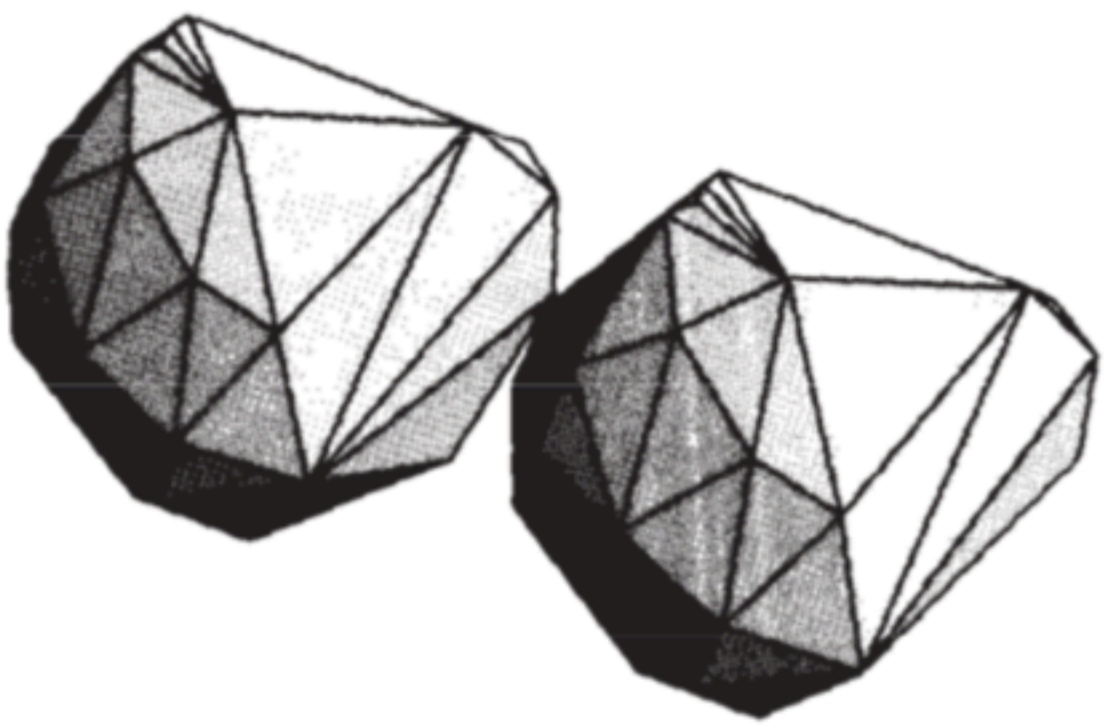
The same idea as 2D algorithm

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

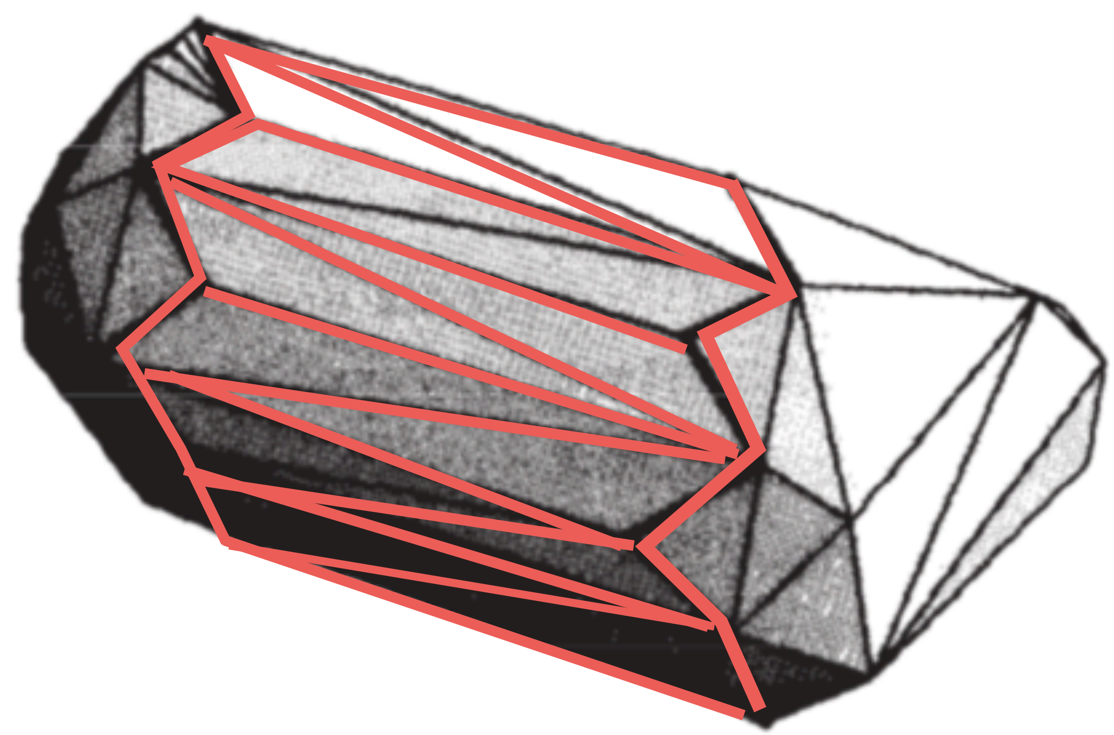
- If merge in O(n) time ==> O(n lg n) algorithm

Merge

- How does the merged hull look like?



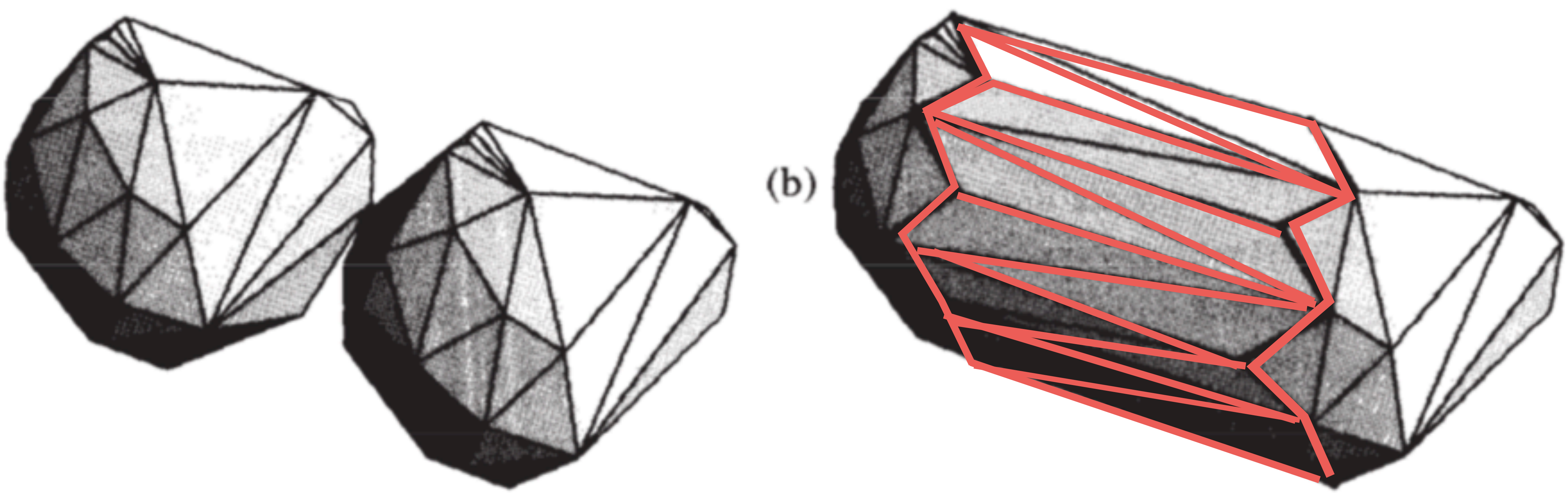
(b)



cylinder without end caps

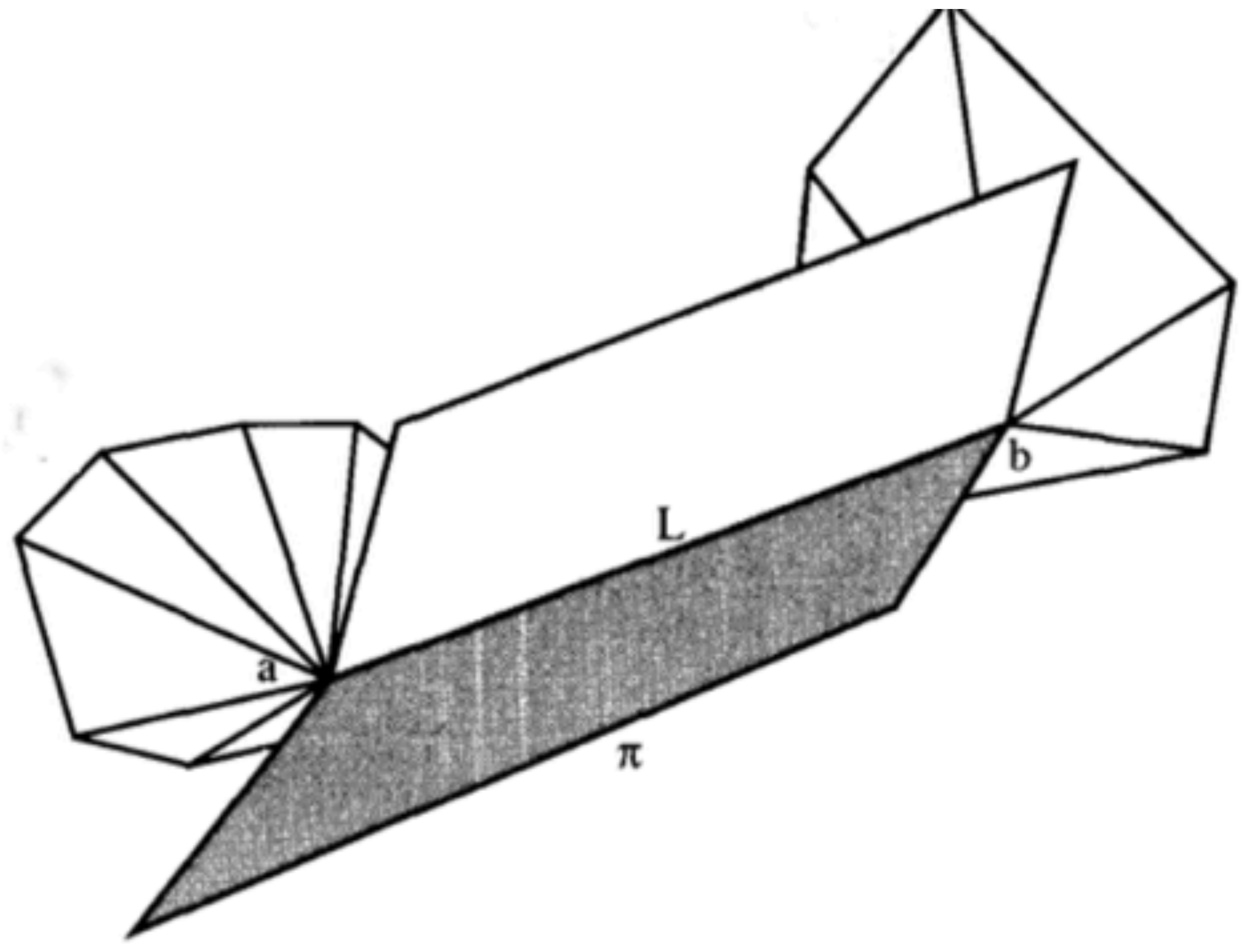
Merge

- 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

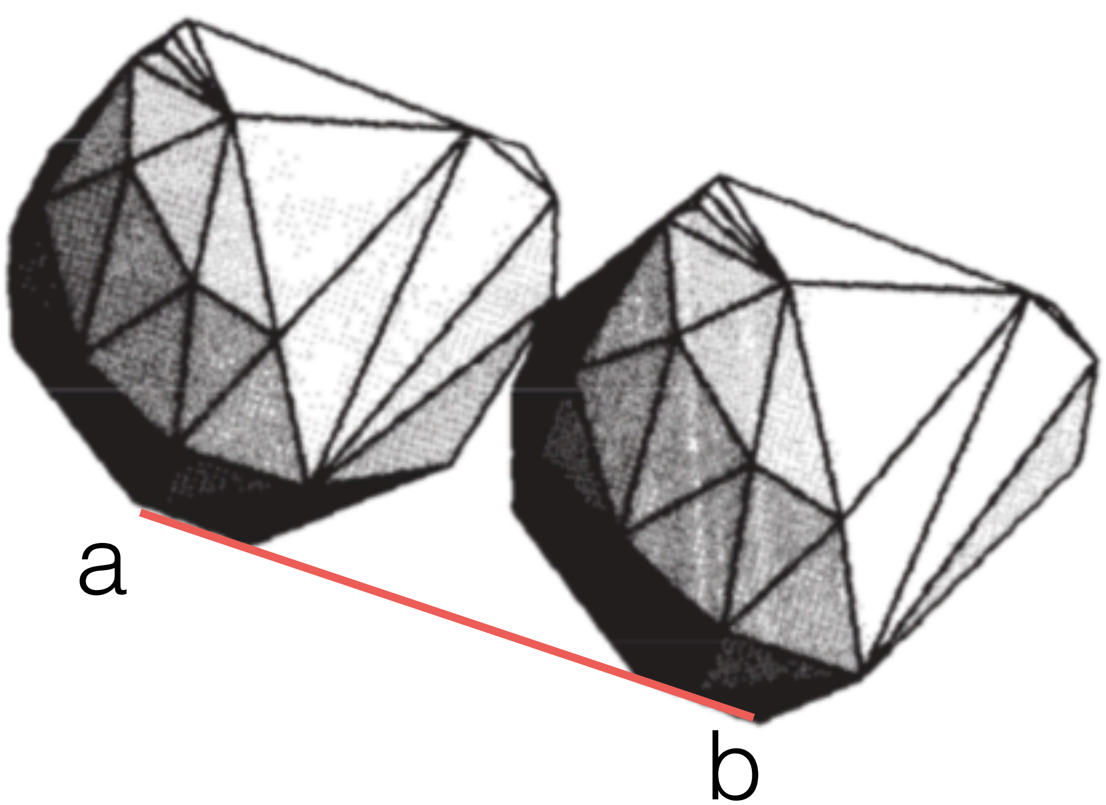


Claim:

- 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

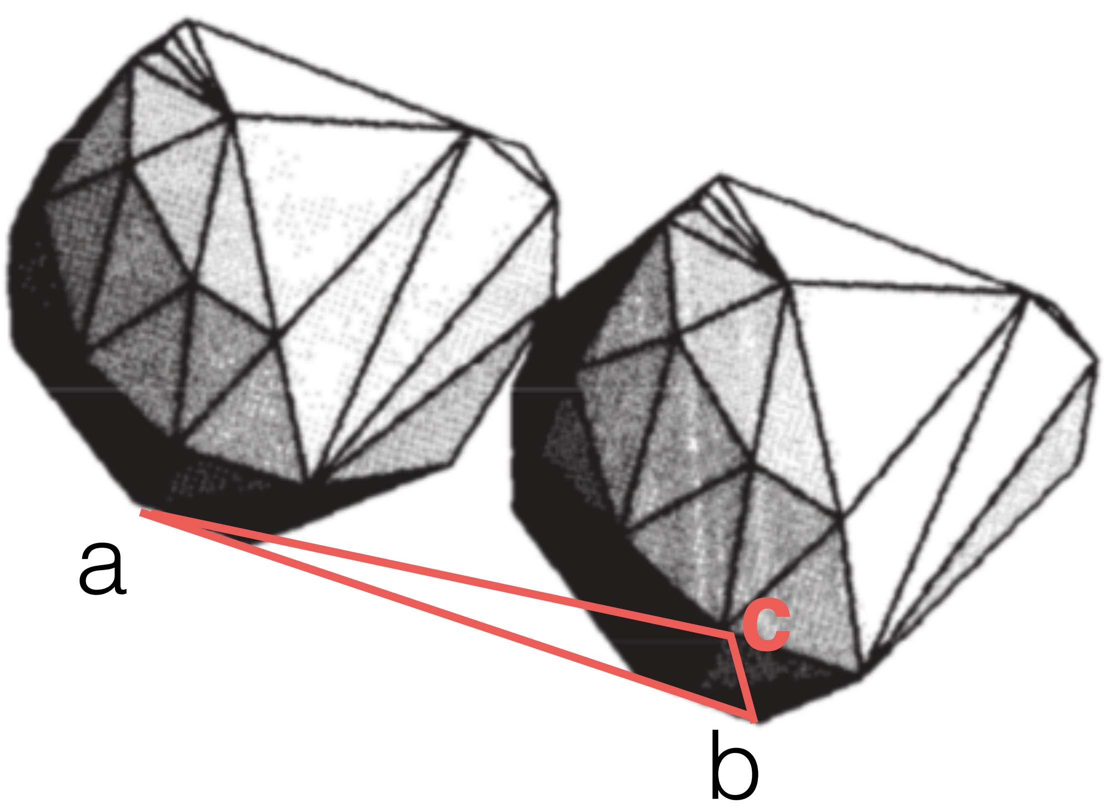
Merge

1. Find a common tangent ab



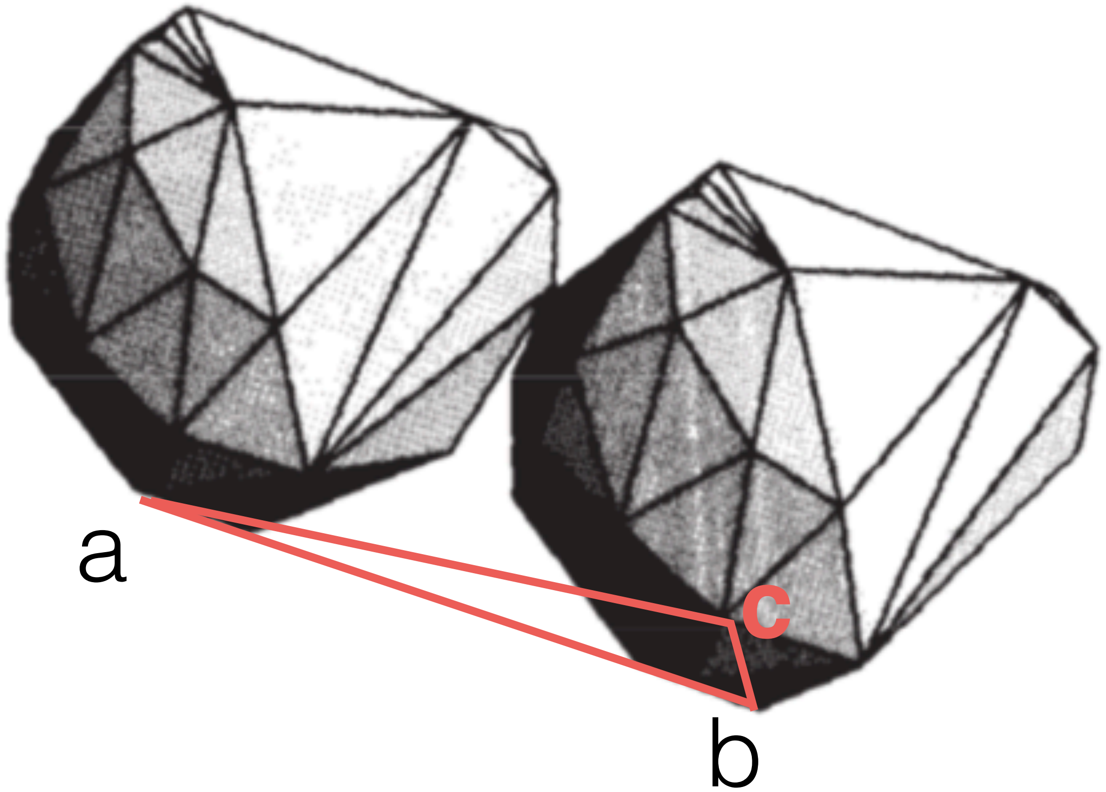
Merge

1. Find a common tangent ab
    - Now we need to find a triangle abc. Thus ac is an edge either on the left hull or on the right hull.



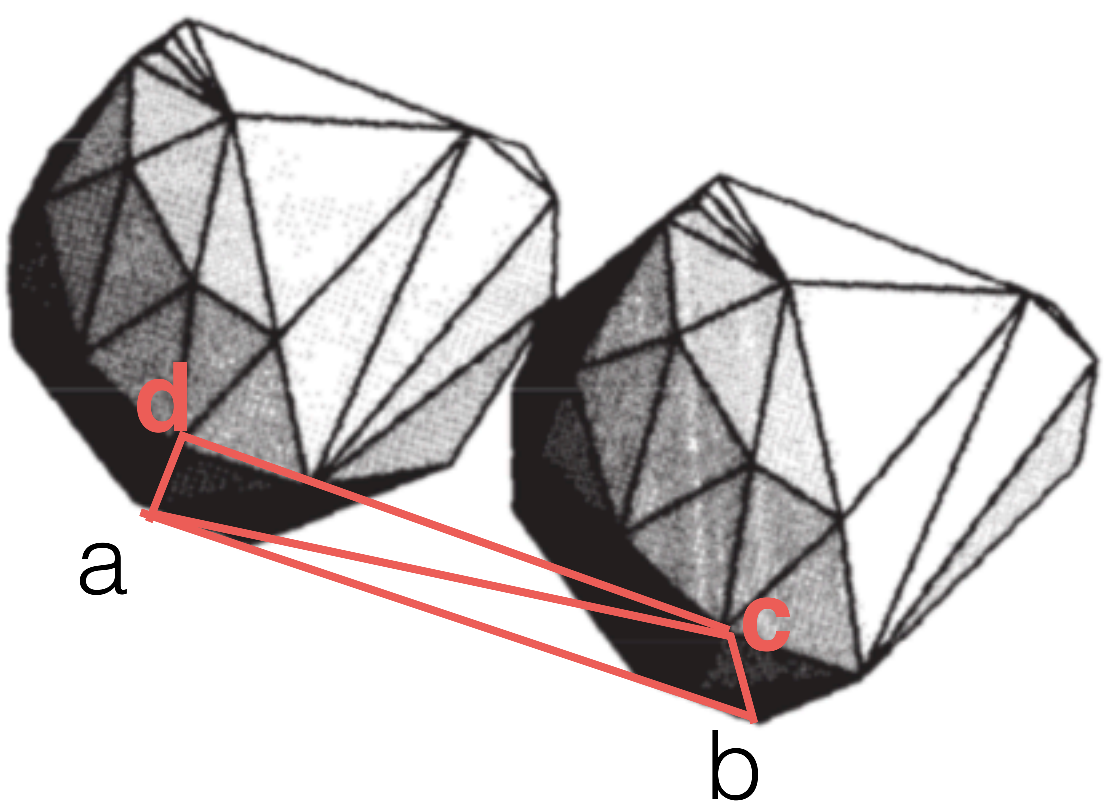
Merge

1. Find a common tangent ab
    - Now we need to find a triangle abc. Thus ac is an edge either on the left hull or on the right hull.
    - Now we have a new edge ac that's a tangent. Repeat.



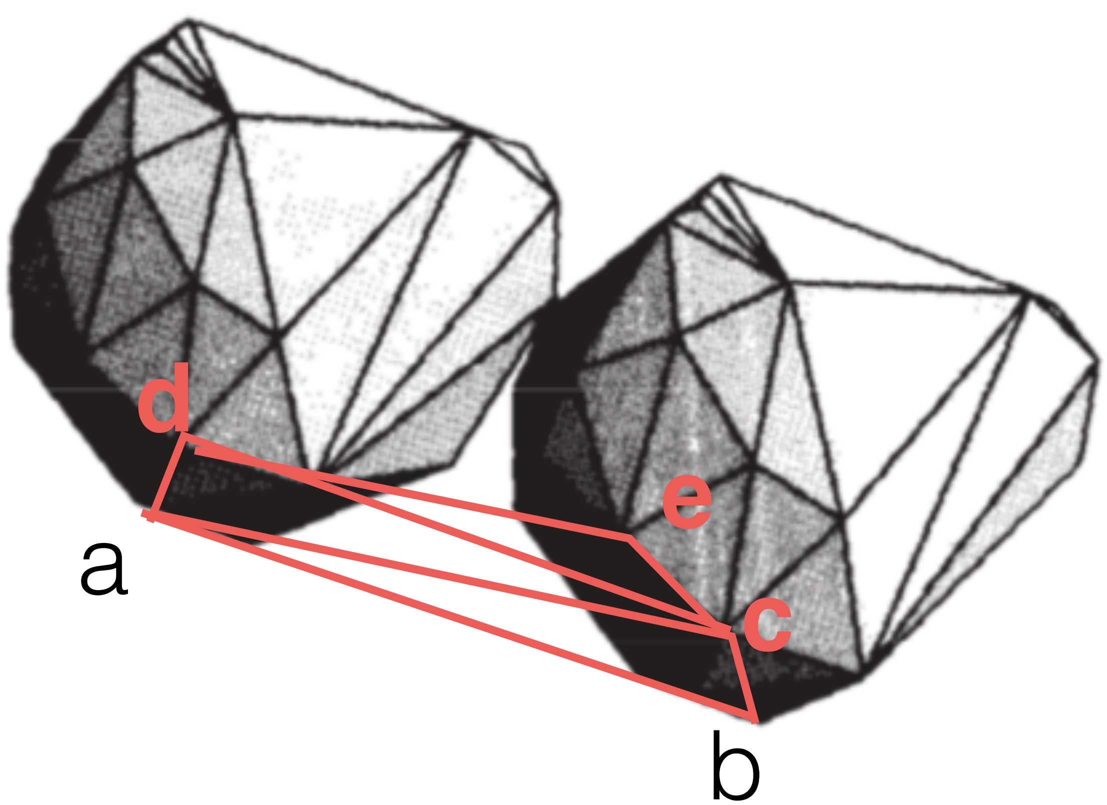
Merge

1. Find a common tangent ab
   - Now we need to find a triangle abc. Thus ac is an edge either on the left hull or on the right hull.
   - Now we have a new edge ac that's a tangent. Repeat.



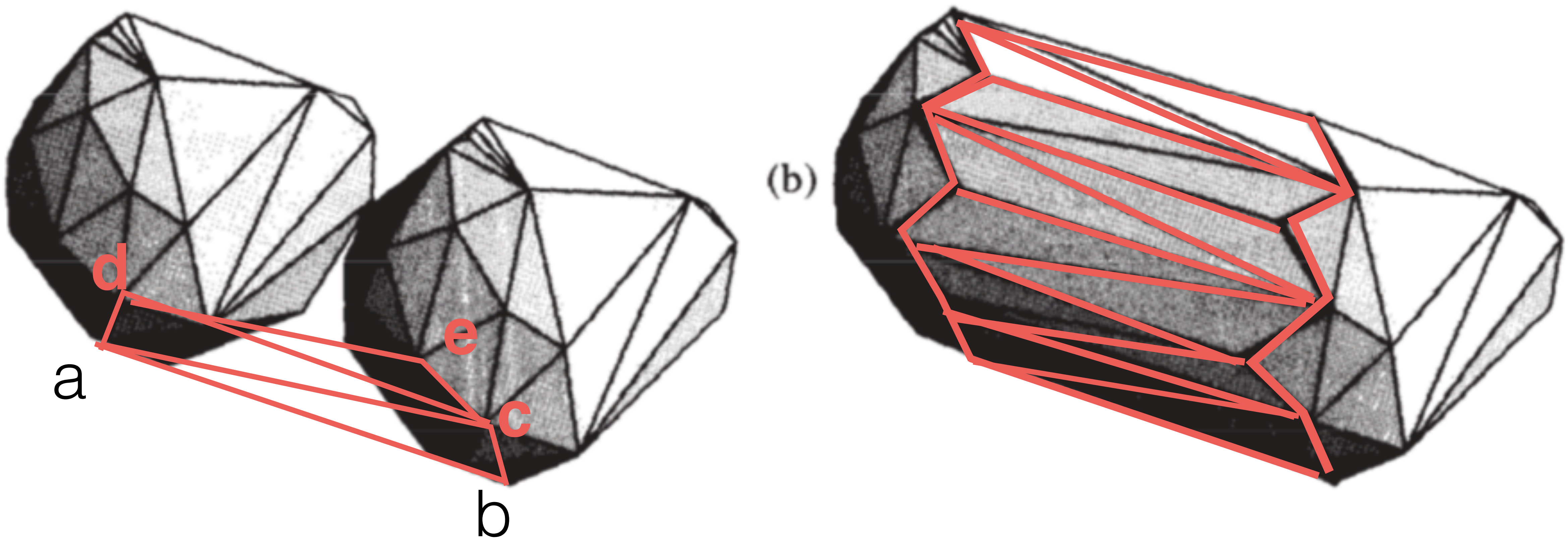
Merge

1. Find a common tangent ab
    - Now we need to find a triangle abc. Thus ac is an edge either on the left hull or on the right hull.
    - Now we have a new edge ac that's a tangent. Repeat.



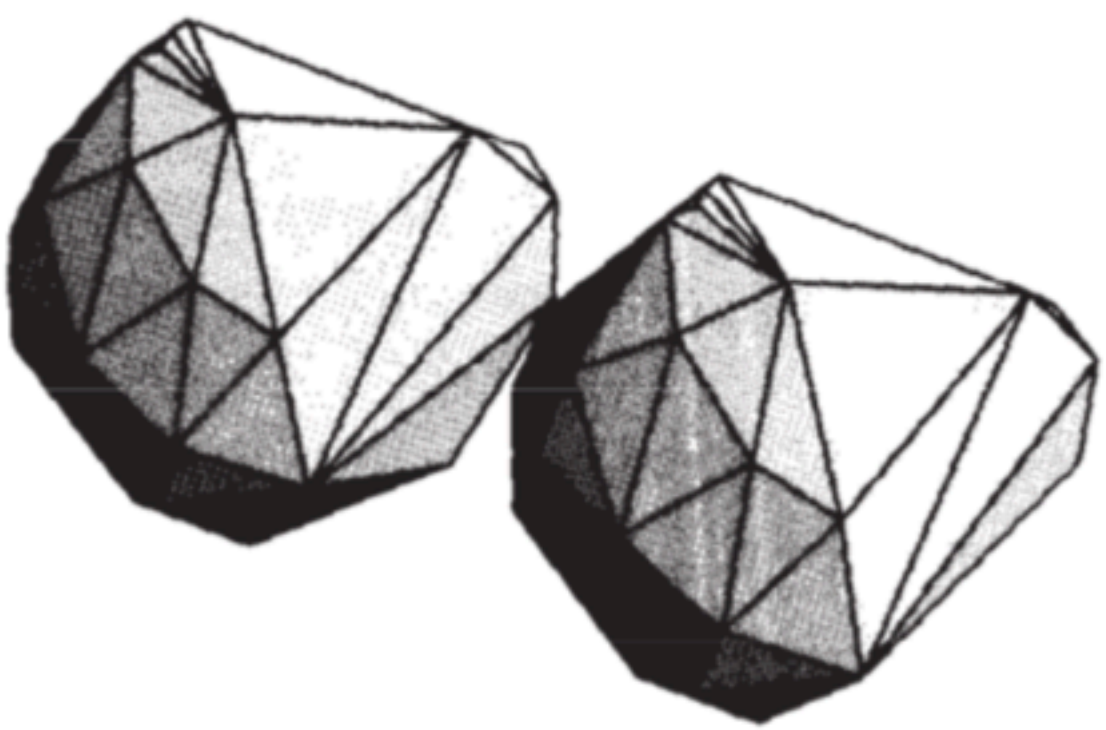
Merge

1. Find a common tangent ab
    - Now we need to find a triangle abc. Thus ac is an edge either on the left hull or on the right hull.
    - Now we have a new edge ac that's a tangent. Repeat.

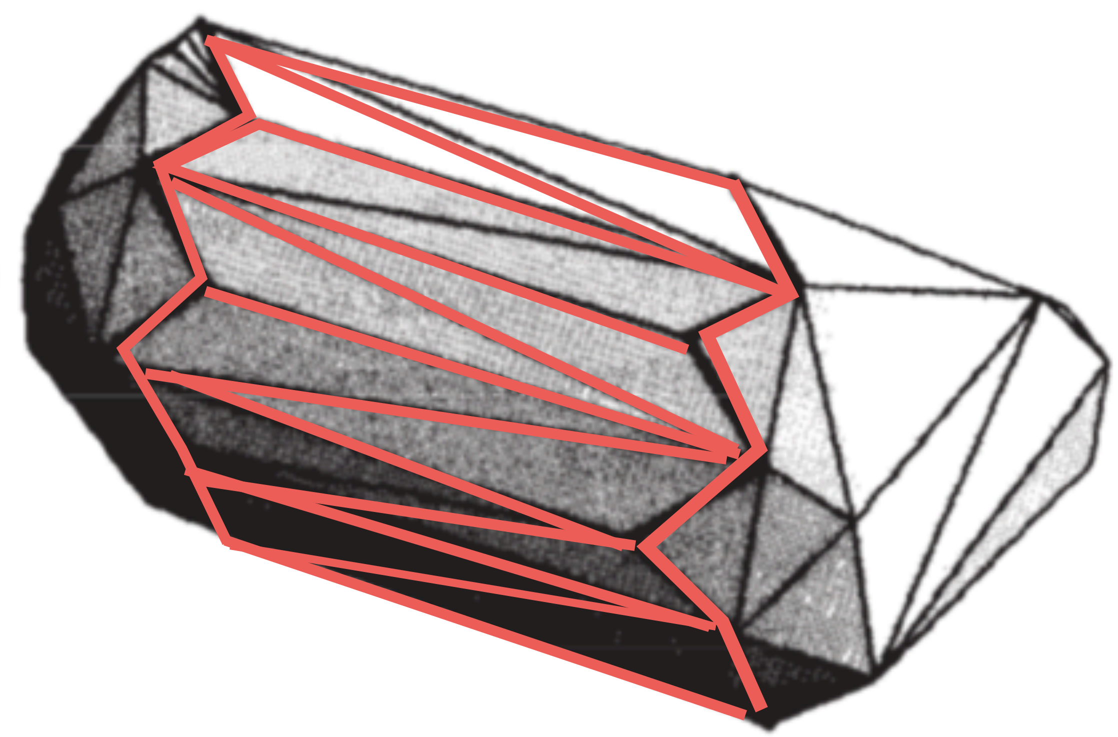


Merge

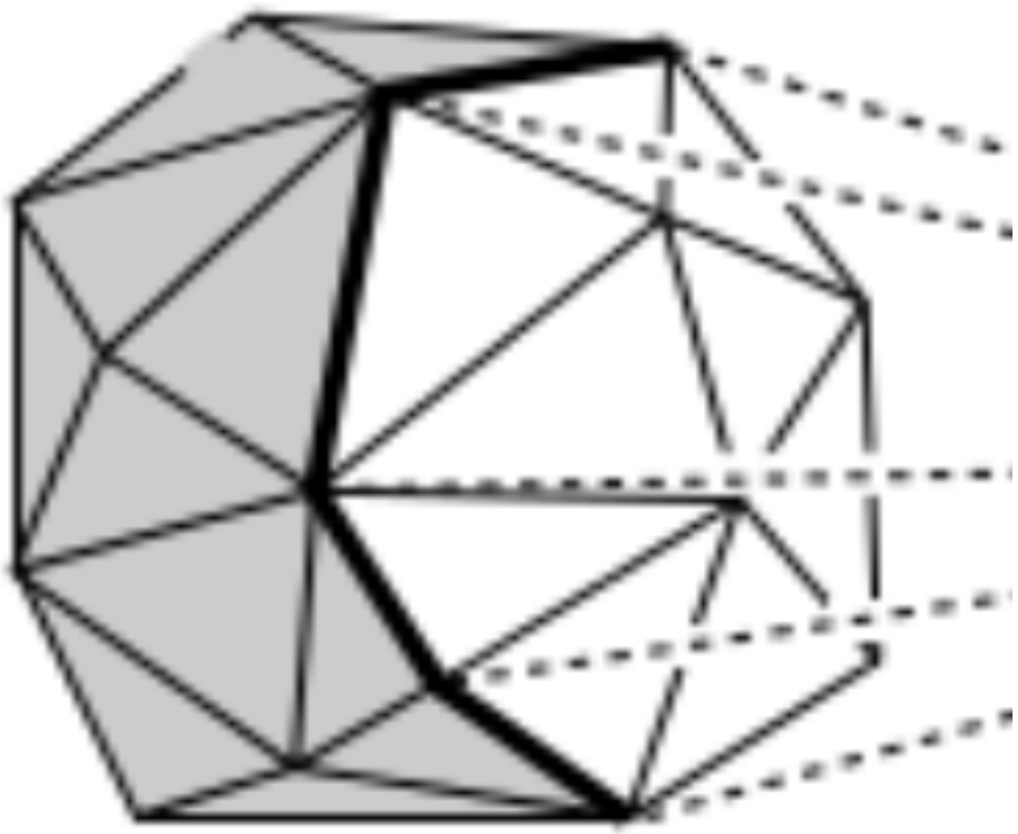
1. Find a common tangent ab
2. Start from ab and wrap around, to create the cylinder of triangles that connects the two hulls A and B
3. Find and delete the hidden faces that are “inside” the cylinder



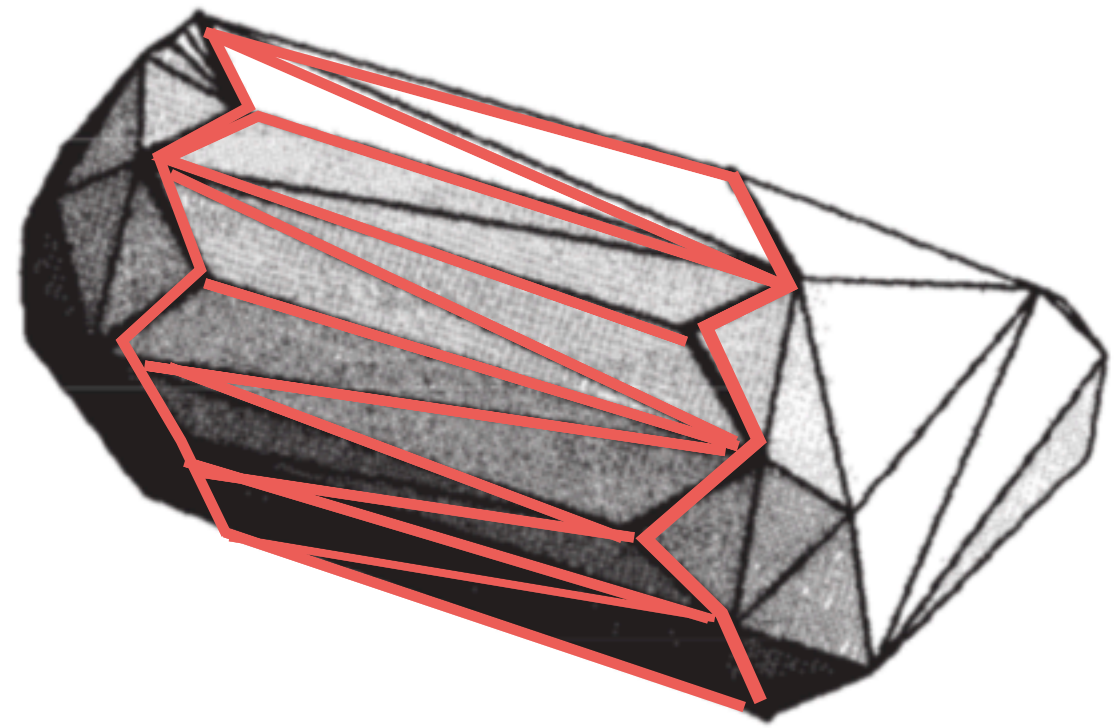
(b)



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