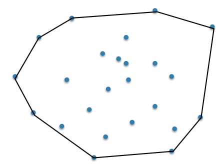
3D convex hulls

Computational Geometry [csci 3250]
Laura Toma
Bowdoin College

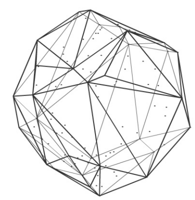
1

Convex Hull in 3D

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

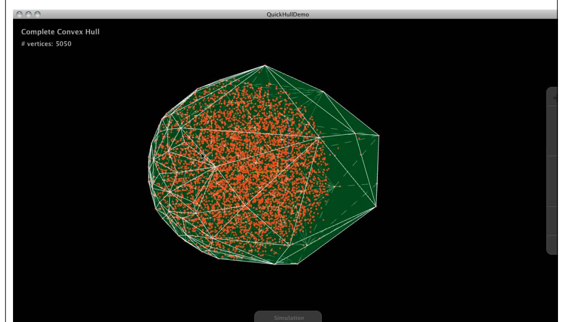


2D

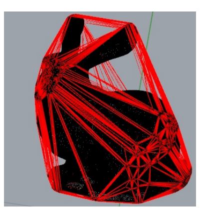


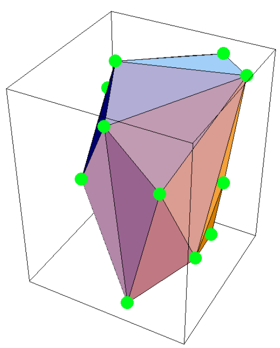
3D

2

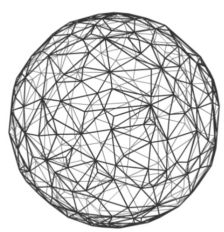


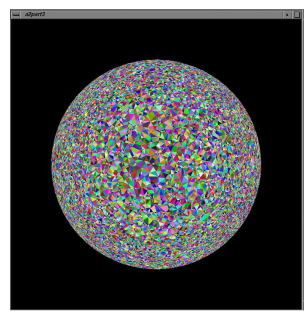
3





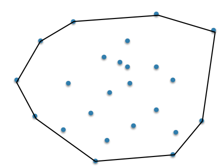
4





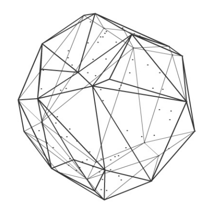
5

polygon



2D

polyhedron

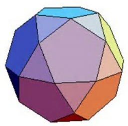


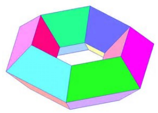
3D

6

Polyhedron

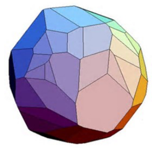
- region of space whose boundary consists of vertices, edges and faces
- faces intersect properly
- neighborhood of any point on P is homeomorphic to a disk
- surface of P is connected





Convexity

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



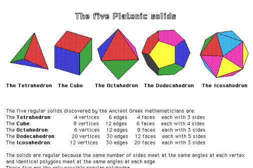
convex

non-convex

convex polyhedron : polytop

Platonic solids

- Regular polygon
  - equal sides and angles
- Regular polytop
  - faces are congruent regular polygons and the number of faces incident to each vertex is the same (and equal angles)
- Surprisingly, there exist only 5 regular polytops



Euler's formula

- Euler noticed a remarkable regularity in the number of vertices, edges and faces of a polyhedron (w/o holes).

- **Euler's formula: V - E + F = 2**

- One proof idea:
  - flatten the polygon to a plane
  - prove the formula for a tree
  - prove for any planar graph by induction on E

Euler's formula

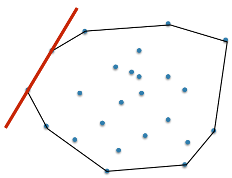
- Consider a polyhedron P with V=n vertices
- Triangulate its faces. This maximizes E and F.

- V-E+F=2
- 3F=2E …..
- ..
- E = 3V - 6 = O(n)
- F = 2V - 4 = O(n)

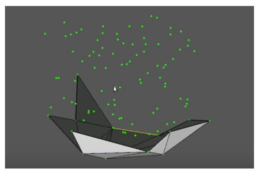
- The number of vertices, edges and faces in a polyhedron are linearly related.

Some properties

- All faces of CH are extreme (and all extreme edges of P are on the CH)
- All internal angles between faces are < 180
- ..



2D



3D

13

From 2D to 3D

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

?

14

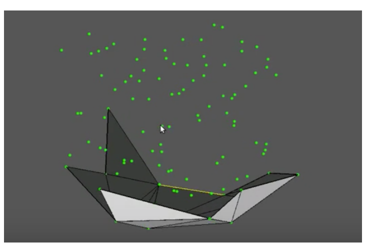
Naive algorithm in 3D

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
- Analysis: $O(n^4)$

15

Gift wrapping in 3D



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

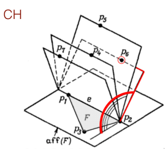
16

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



17

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?

18

From 2D to 3D

2D		3D	
Naive	$O(n^3)$	$O(n^4)$	
Gift wrapping	O(nh)	$O(n \cdot F)$	
Graham scan	O(n lg n)	no	!!
Quickhull	O(n lg n), $O(n^2)$	?	
Incremental	O(n lg n)		
Divide-and-conquer	O(n lg n)		

19

From 2D to 3D

2D		3D	
Naive	$O(n^3)$	$O(n^4)$	
Gift wrapping	O(nh)	$O(n \cdot F)$	
Graham scan	O(n lg n)	no	!!
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)	

20

3d hull: divide & conquer

The same idea as 2D algorithm

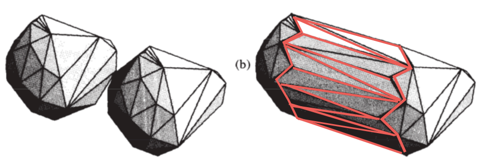
- divide points in two halves P1 and P2
- recursively find CH(P1) and CH(P2)
- merge

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

21

Merge

- How does the merged hull look like?

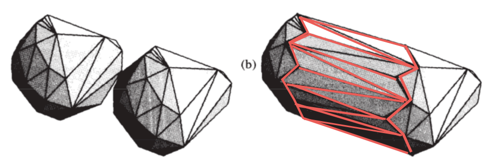


cylinder without end caps

22

Merge

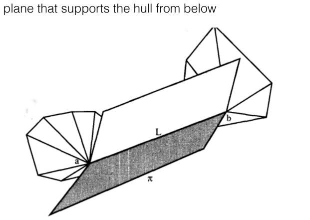
- Idea: Start with the lower tangent, wrap around, find one face at a time.



23

Merge

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



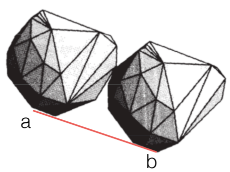
Claim:

- When we rotate Pl 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

24

Merge

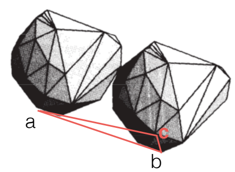
1. Find a common tangent ab



25

Merge

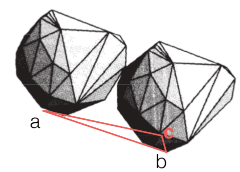
1. Find a common tangent ab
   - Now we need to find a triangle abc. Thus ac is an edge either on the left hull or on the right hull.



26

Merge

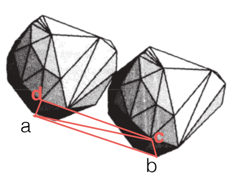
1. Find a common tangent ab
   - Now we need to find a triangle abc. Thus ac is an edge either on the left hull or on the right hull.
   - Now we have a new edge ac that's a tangent. Repeat.



27

Merge

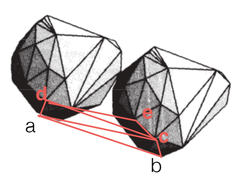
1. Find a common tangent ab
   - Now we need to find a triangle abc. Thus ac is an edge either on the left hull or on the right hull.
   - Now we have a new edge ac that's a tangent. Repeat.



28

Merge

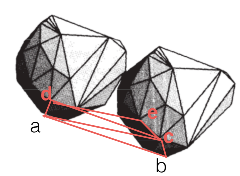
1. Find a common tangent ab
   - Now we need to find a triangle abc. Thus ac is an edge either on the left hull or on the right hull.
   - Now we have a new edge ac that's a tangent. Repeat.

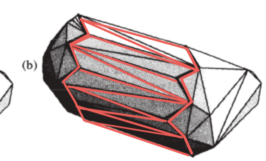


29

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.

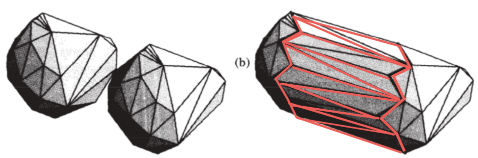




30

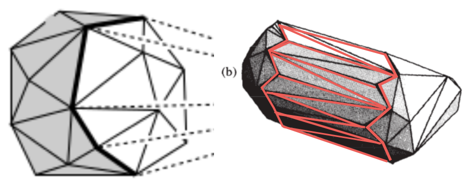
Merge

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



31

The hidden faces



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

32

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

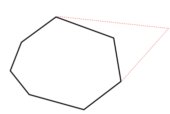
33

Incremental 3D hull

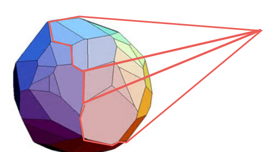
34

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 $p_1..p_i$



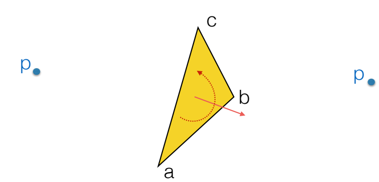
2D



3D

35

Point in front/behind face



ps is left of (behind) abc

abc not visible from p

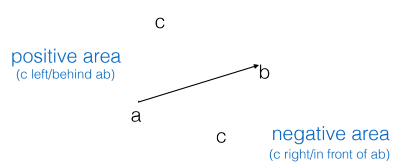
p is right of (in front) abc

abc visible from p

36

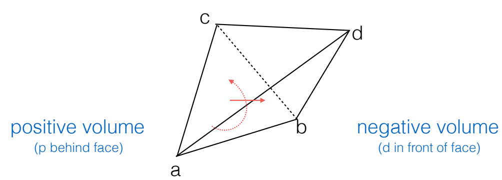
2D

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

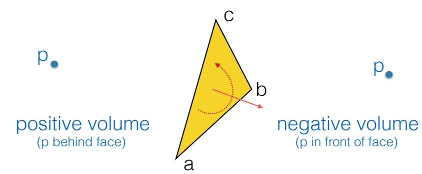


3D

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

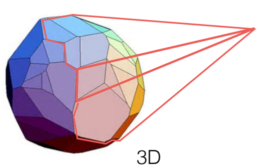


- Assume all faces oriented counterclockwise (their normals determined by the right-hand rule point towards the outside of P)



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