CS 164 & CS 266: Computational Geometry

Lecture 7

3d convex hulls

David Eppstein
University of California, Irvine

Fall Quarter, 2025

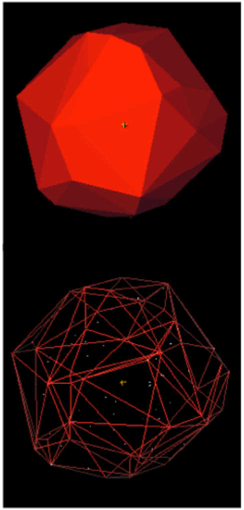
Convex polyhedra

Convex polyhedron

The convex hull of a finite set of points p_i in 3d is:

- all convex combinations of points, where a convex combination is a weighted average $\left(\sum w_i p_i\right) / \left(\sum w_i\right)$ for non-negative weights w_i
- the union of tetrahedra defined by subsets of four points p_i
- the intersection of all half-spaces that contain the points p_i

[Pbierre 2015]



Faces and their dimensions

The faces of a 3d polyhedron are its vertices, edges, and facets

Vertices are 0-dimensional, edges are 1d, facets are 2d

(Sometimes the facets are called faces, but we need a word for all of these things.)

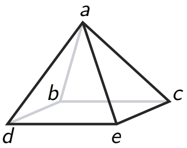
It is convenient to think of the empty set (dimension $= -1$) and the whole polyhedron (dimension $= 3$) as also being faces

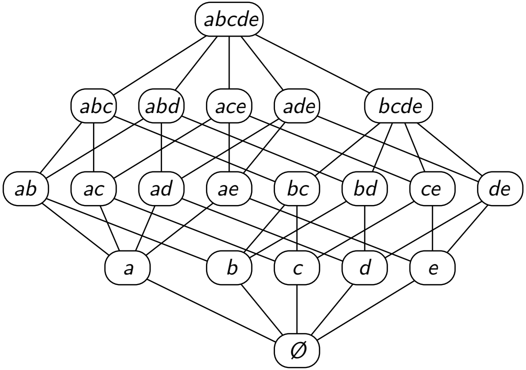
Face $=$ any intersection of the polyhedron with a closed halfspace whose boundary is disjoint from the interior of the polyhedron

(This generalizes to d-dimensional convex hulls: "polytopes")

The face lattice

Faces of all dimensions and their inclusion relations





Convex hull problem: Convert set of points to this structure

Euler's formula

For 3d convex polyhedra: $V - E + F = 2$

Or, in terms of face lattice with empty set and whole polyhedron

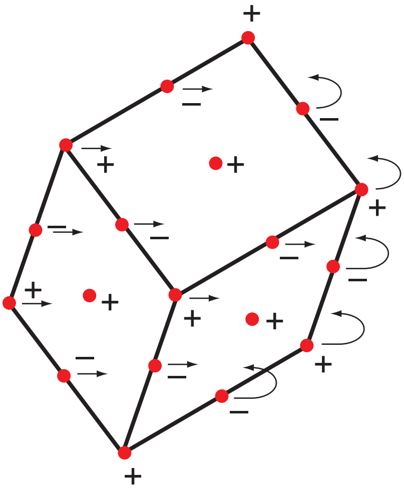
$$\sum_{\mathrm{dim}=-1}^{d} (-1)^{\mathrm{dim}} \times (\#\ \text{faces of dim}) = 0$$

This formula works for convex hulls in any dimension!

For instance, in 2d, it says: $\#$ vertices $= \#$ edges

One of many proofs

- Rotate polyhedron so each face has exactly one vertex with minimum z-coordinate, and exactly one with maximum z-coordinate
- Place positive charge at each vertex, negative on each edge, positive in each facet, so total charge is $V - E + F$
- Rotate charges around z-axis into nearby facets
- Each facet gets -1 total from a path of edges and vertices along one side, cancelling its own $+1$ charge
- The only charges left are $+1$ at the north and south poles



3d hulls have linear complexity

$$V - E + F = 2$$

$$2E \geq 3F \quad \iff \quad \tfrac{2}{3}E \geq F \quad \iff \quad E \geq \tfrac{3}{2}F$$

(There are 2 faces per edge and at least three edges per face, so the number of face-edge incidences is exactly $2E$ and at least $3F$.)

Combine:

$$V - E + \tfrac{2}{3}E \geq 2 \quad \iff \quad E \leq 3V - 6 = O(V)$$

$$V - \tfrac{3}{2}F + F \geq 2 \quad \iff \quad F \leq 2V - 4 = O(V)$$

Complexity blows up in higher dimensions

The worst case for convex hulls in d dimensions is given by n points with coordinates $(x, x^2, x^3, \ldots, x^d)$ (for n different values of x)

$$\#\ \text{faces} = \Theta\left(n^{\lfloor d/2 \rfloor}\right)$$

So n^2 for dimensions 4, 5; n^3 for dimensions 6, 7, etc.

An unsolved problem: Suppose you have a 4d convex hull of n points, and it also has $O(n)$ three-dimensional faces. How many 1d edges and 2d polygons can it have?

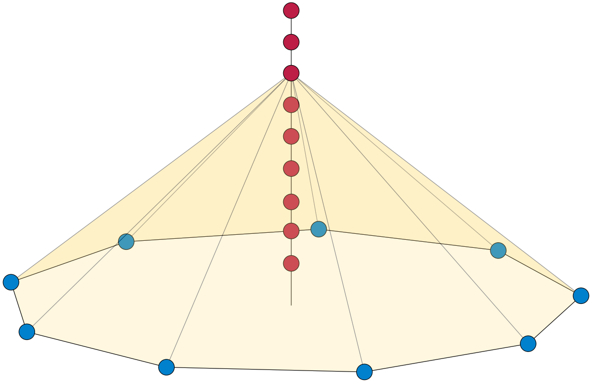
Unknown whether linear or nonlinear

[Eppstein et al. 2003]

3d convex hull algorithms in general

Graham scan isn't efficient

Graham scan: add points in sorted order by one coordinate (say z)



Bad example: Start with a cone of $n/2$ vertices and then keep making the peak higher

Like lifting a circus tent on its center pole

Each added peak makes $n/2$ new faces ⇒ total time $\Theta(n^2)$

Some ideas that do work

Like mergesort

Split arbitrarily into sets of $n/2$ points

Recurse

Merge the two hulls

[Chazelle 1992]

Like quicksort

Split at median x-coordinate

Recurse

"Gift-wrap" the two disjoint hulls

[Preparata and Hong 1977]

Both of these methods can be made to run in $O(n \log n)$ time

… but the details are complicated …

The algorithm from the book

Randomized incremental algorithms

A general method for designing algorithms,
useful for many different computational geometry problems

(not just convex hulls)

Incremental: Add input objects one at a time,
maintaining solution of what has been added

(we saw this already for line arrangements)

Randomized incremental: Add in random order

(can help avoid worst-case complexity of adding an item)

Random permutations

Given n items, there are $n!$ possible permutations; we want to make them all equally likely (like shuffling cards)

To permute n items listed in an array, $A[0], \ldots, A[n-1]$:

```
for $i = 1, 2, \ldots, n-1$:
    Choose a random number $j$ from $0, \ldots, i$
    Swap $A[i], A[j]$        (does nothing if $i = j$)
```

Time is obviously $O(n)$

By induction, after i swaps, the permutation of the first $i+1$ items is uniformly random (all permutations equally likely)

Main idea and data structures

Main idea: randomized incremental (add points in random order)

Maintain DCEL of hull vertices, edges, and facets

Also maintain "conflict graph":

- Vertices: points that have not been added, and facets of current hull
- Edges: pairs of a point and a facet that it can see
- Represent as list of visible facets for each point, and list of conflict points for each facet

How to initialize the data structures

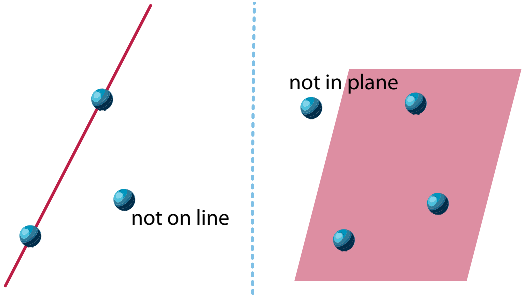
Initial hull

Choose two arbitrary points

Find point not on their line

Find point not on their plane

Result is a tetrahedron



Conflict graph

Check whether each point can see each tetrahedron face
(3d version of left-right orientation test)

Points that cannot see anything are inside the tetrahedron and can be removed

Main algorithm

For each remaining point p, in a random order:

- The conflict graph lists all of the facets that it can see
- Cut out those facets from the hull, leaving a hole, and remove all vertices and edges that become disconnected from the rest of the hull

- For each boundary edge e of the hole, make a new triangle T_e connecting e to p
- If T_e is in the same plane as the other facet on edge e, merge them
  (the merged face keeps the same conflict list as it had before)
- If a triangle T_e is not merged, it needs a new conflict list.
  Take the union of the conflict lists of the two old facets on edge e, and check whether each point can see the new triangle T_e.

A tiny amount of probability theory

When different random choices produce different values of x, the expected value of x is their weighted average, weighted by probabilities:

$$E[x] = \sum_{\text{choice } y} \Pr(y) \cdot (\text{value of } x \text{ when choice is } y)$$

If x is 0 or 1, then its expected value equals the probability that $x = 1$

Linearity of expectation: $E[\sum \cdots] = \sum E[\cdots]$

(Because E is a sum and this is just changing the order of two sums)

So: The expected number of things that happen equals the sum of their probabilities

(for whatever things you're trying to count)

Partial analysis: How much does DCEL change?

After adding point i, what is expected $\#$ edges we just added?

- Current set of i points has $\leq 3i - 6$ edges
- Edge e was just added if we just added one of its 2 endpoints
- Because of the random permutation, each of the i points is equally likely to be the one we just added
- So probability we just added edge e is $2/i$

Expected number of new edges $\leq (3i-6) \cdot \frac{2}{i} = 6 - \frac{12}{i} < 6$

(linearity of expectation)

Partial analysis: How much does DCEL change?

Expected # new edges in each step is < 6

$\Rightarrow$ expected # edges for whole algorithm is $< 6n$

(linearity of expectation again)

So total expected change to DCEL, over whole algorithm, is $O(n)$

Rest of analysis: How much do conflict lists change?

The book refers to Section 9.5, which considers an abstract framework for problems likef this:

- We have a sequence of randomly inserted points
- As they are inserted, they cause the creation of certain small "configurations" in the partial output (could be: triangles in current convex hull but book uses edge and its two incident faces)
- Each configuration C is defined by a d-tuple of points (here $d = 3$ or 4), and has a conflict list of "killer" points $K(C)$
- C only appears when its defining d-tuple is chosen first, ahead of all k killers, with probability $1/\binom{k+d}{d}$
- We want to know the total size of all the conflict lists that appear

Rest of analysis: How much do conflict lists change?

Theorem 9.15: Expected total size of all conflict lists is at most

$$\sum_{i=1}^{n} d^2 \cdot \frac{n-i}{i} \cdot \frac{E[\#\text{ configs after } i\text{th insertion}]}{i}$$

Each term in the sum bounds the expected size of the new conflict lists after inserting the ith point.

You can calculate it by summing, for each C and i, the product of its conflict list with the probability that it appears in the ith step

Configurations with bigger conflict lists are much less likely to appear in later insertions so the bigger size contribution is balanced by a smaller probability of appearing

The details in the book's proof are not very enlightening.

Rest of analysis: How much do conflict lists change?

Theorem 9.15: Expected total size of all conflict lists is at most

$$\sum_{i=1}^{n} d^2 \cdot \frac{n-i}{i} \cdot \frac{E[\#\text{ configs after }i\text{th insertion}]}{i}$$

For us, d is a constant, and $\#$ configs $= O(i)$, so the first and last factors are $O(1)$

We can simplify the middle factor by replacing $n - i$ by n (only makes sum bigger), giving:

$$\sum_{i=1}^{n} O\left(\frac{n}{i}\right) = O\left(n \sum_{i=1}^{n} \frac{1}{i}\right) = O(n \log n).$$

References and image credits I

Bernard Chazelle. An optimal algorithm for intersecting three-dimensional convex polyhedra. *SIAM Journal on Computing*, 21(4):671–696, 1992. doi: 10.1137/0221041.

David Eppstein, Greg Kuperberg, and Günter M. Ziegler. Fat 4-polytopes and fatter 3-spheres. In Andras Bezdek, editor, *Discrete Geometry: In honor of W. Kuperberg's 60th birthday*, volume 253 of *Pure and Applied Mathematics*, pages 239–265. Marcel Dekker, 2003.

Pbierre. 3D convex hull of a 120 point cloud. Licensed under the Creative Commons Attribution-Share Alike 4.0 International license, October 19 2015. URL `https://commons.wikimedia.org/wiki/File:3D_Convex_Hull.tiff`.

F. P. Preparata and S. J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Communications of the ACM*, 20(2):87–93, 1977. doi: 10.1145/359423.359430.