KOREA ADVANCED INSTITUTE OF SCIENCE AND TECHNOLOGY
KAIST
한국과학기술원
SINCE 1971

Polytopes and convex hulls in higher dimensions

Geometric Computing

Hyperplane

- A set of points in R^d satisfying a linear equality of the form $a_1x_1 + \ldots + a_dx_d = a_0$ where x_j's denote the coordinates in R^d, the coefficients a_j's are fixed real numbers, and $a_1, \ldots, a_d$ are not all zero.
- d=2, a line
- d=3, a plane

Half-space

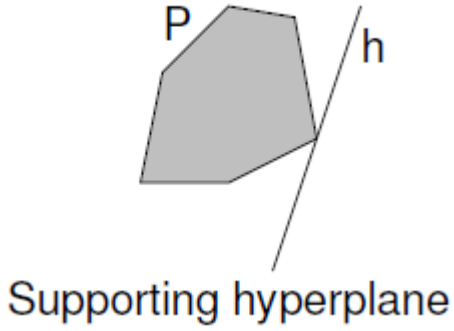
- A set of points in R^d satisfying a linear inequality of the form $a_1x_1 + \ldots + a_dx_d \geq a_0$ where x_j's denote the coordinates in R^d, the coefficients a_j's are fixed real numbers, and $a_1, \ldots, a_d$ are not all zero.
- The simplest convex set in R^d.
- The boundary of the half-space is the hyperplane.

Polytope

- Let N be any set of half-spaces in R^d.
- Their intersection H(N), if nonempty, is called the convex polytope formed by N.
- A convex polytope with dimension d is called d-polytope.

Face

- A hyperplane h *supports* a polytope P if h ∩ P is nonempty and P is entirely contained within one of the halfspaces bounded by h.
- The intersection of P with any supporting hyperplane is called a face of P.
- Faces are also convex polytopes of dimension from 0 to d-1.

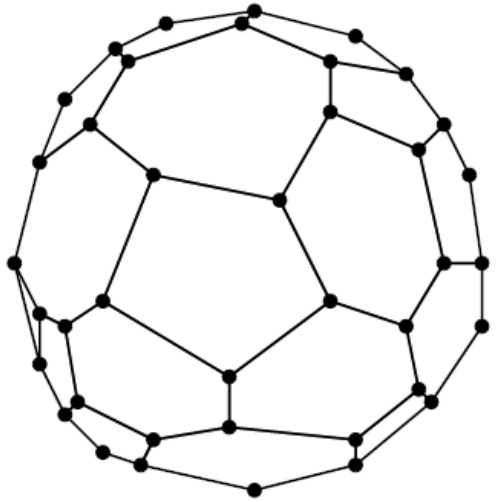


Face

- d-face : d-polytope itself
- (d-1)-face : facet
- d-2-face : subfacet
- 0-face : vertex
- 1-face : edge
- The size of polytope = the total number of its faces of all dimensions.

Simple polytope

- A half-space in N is called *redundant* if H(N) is unaffected by its removal. Otherwise, it is *nonredundant*.
- A hyperplane bounding a nonredundant half-space in N is called a *bounding* or a *nonredundant* hyperplane of H(N).
- The polytope H(N) is called *simple* if every j-face of H(N) is contained in exactly d-j bounding hyperplanes.



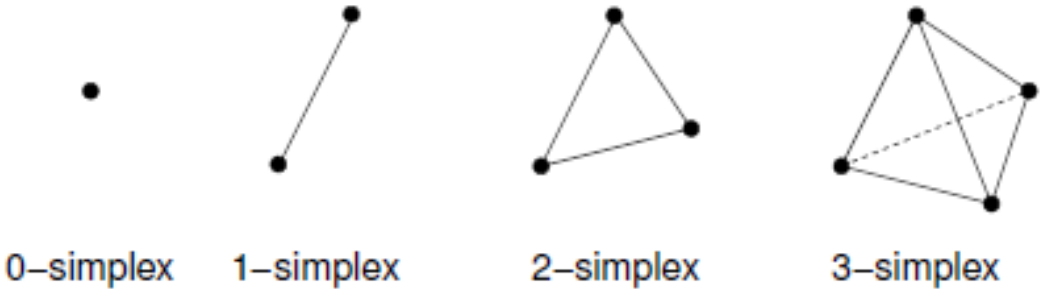
Simple Polytope

Affine independence

- K points $p_1,\ldots, p_k$ are said to be *affinely independent* if (k-1) vectors p_2-p_1, p_3-p_1 ,…, p_k-p_1 are linearly independent.

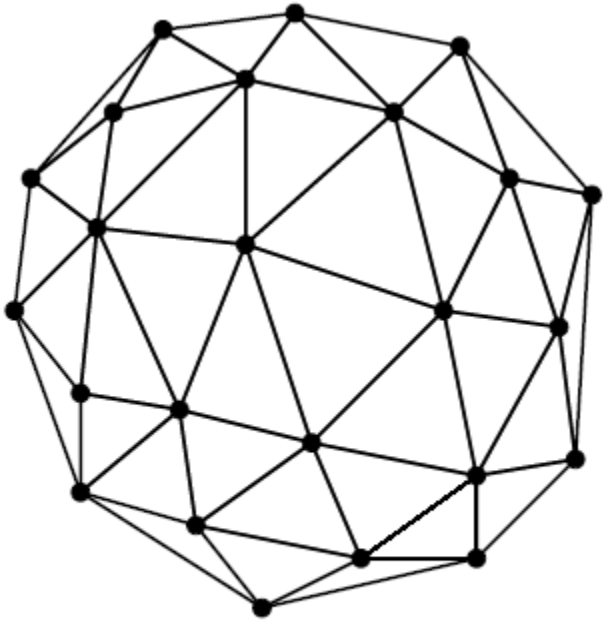
Simplex

- A d-polytope is called a *d-simplex* if it is the convex hull of d+1 affinely independent points.
- 3-simplex : a tetrahedron
- 2-simplex : a triangle
- 1-simplex : a line segment
- 0-simplex : a vertex



Simplicial polytope

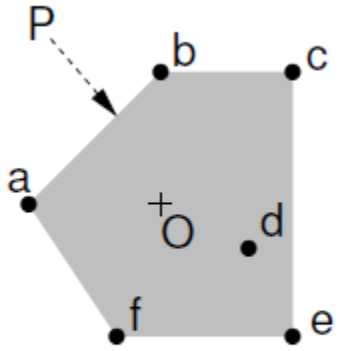
- A d-polytope is called simplicial if each of its facets is a simplex.

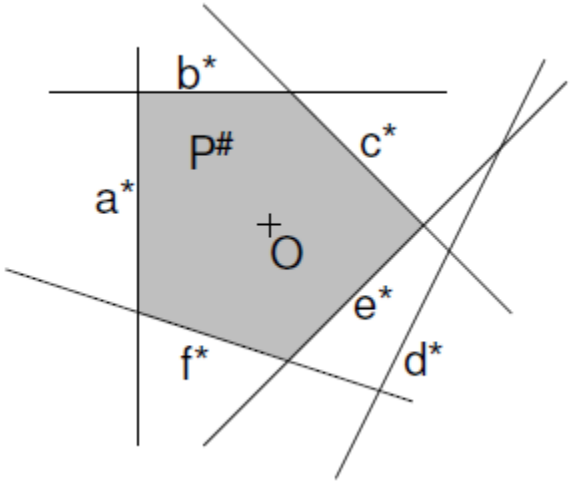


Simplicial Polytope

Duality transform

- Transform T maps a point $p = (p_1,\ldots, p_d)$ to the hyperplane $<p,x> = p_1 x_1 + \ldots + p_d x_d = 1$, and vice versa. (assume $p_1 ,\ldots, p_d$ are not all zero).
- This transform maps a convex polytope (containing an origin) into a convex hull.





Simplicial vs. Simple polytopes

- Among all polytopes with a fixed number of vertices, simplicial polytopes maximize the number of faces of all higher degrees.
- Dually, simple polytopes maximize the number of faces of all lower degrees.

Simplicial Polytope

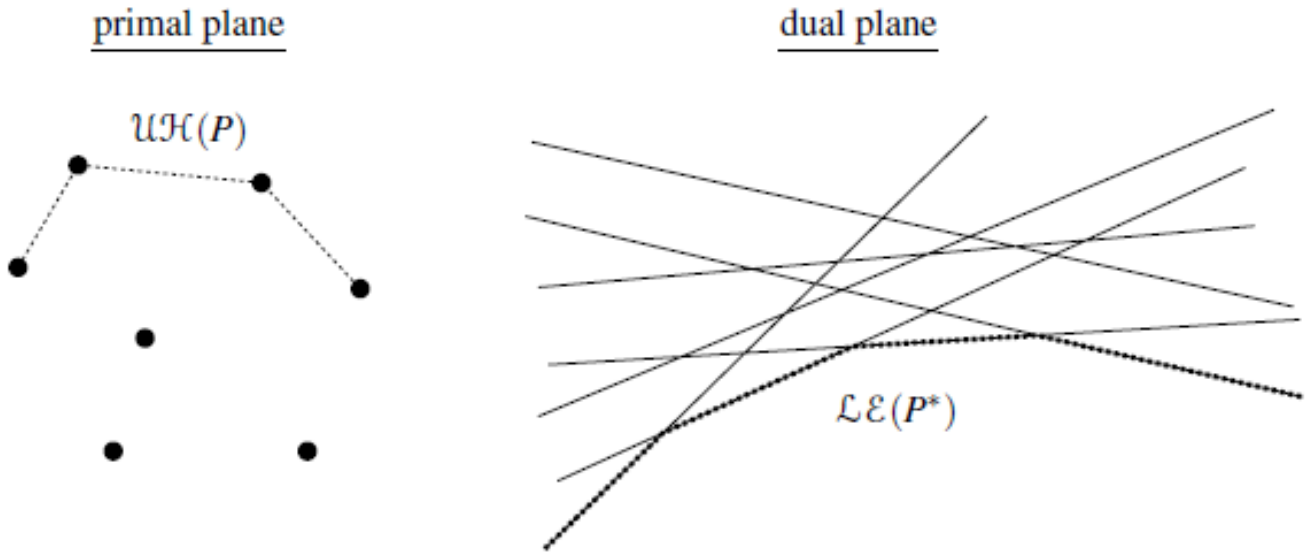
Simple Polytope

Upper bound theorem

- A polytope defined by the convex hull of n points in R^d has $O(n^{\lfloor d/2 \rfloor})$ facets.
- A polytope defined by the intersection of n halfspaces in R^d has $O(n^{\lfloor d/2 \rfloor})$ vertices.

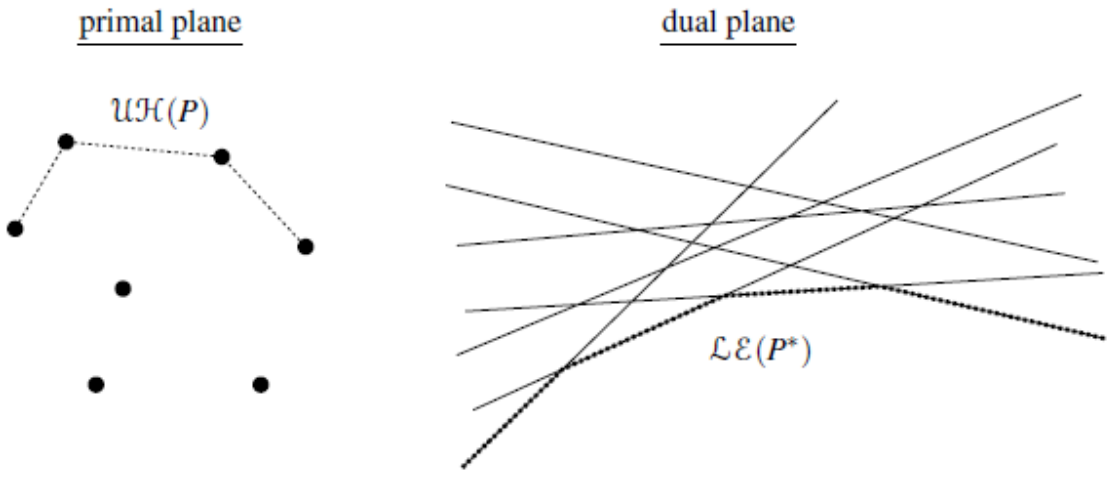
Convex hulls and half-space intersection

- Upper convex hull UH(P) : convex hull edges that have P below their supporting line.
- Lower envelope LE(P*) : the boundary of the bottom cell in the arrangement of P*.



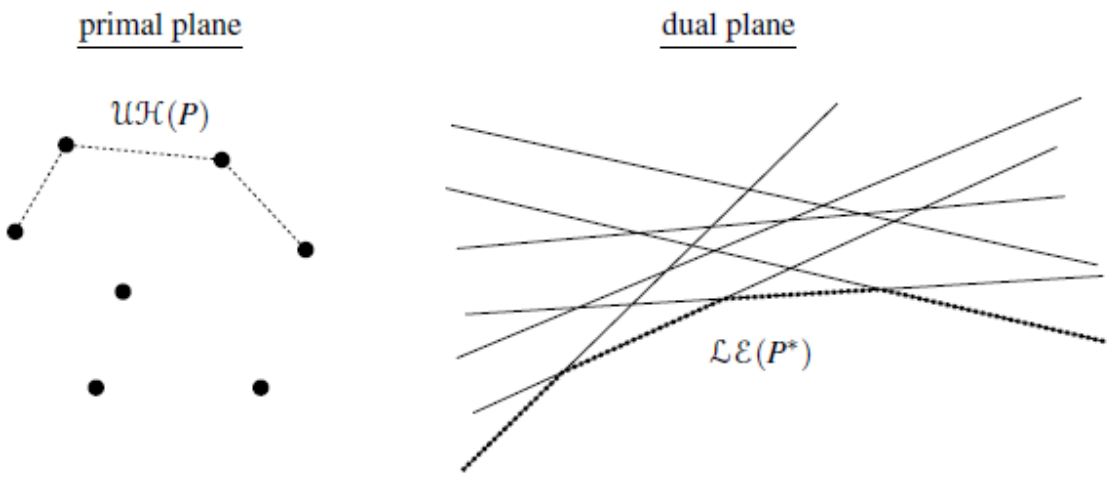
Convex hulls and half-space intersection

- A point p appears as a vertex in upper convex hull if and only if there is a non-vertical line l through p s.t. all other points of P lie below l.
- In dual plane, there is a point l* on the line p* s.t. l* lies below all other lines of P* -> p* is an edge in the bottom cell of the arrangement A(P*).
- Left-to-right list of vertices in UH(P) -> right-to-left list of edges of LE(P*)



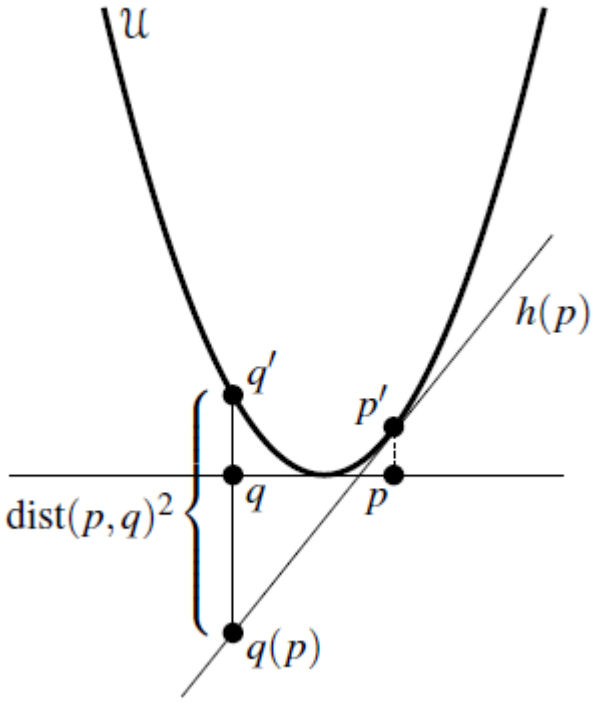
Convex hulls and half-space intersection

- p and q form an upper convex hull edge if and only if all other points in P lie below the line l through p and q.
- In dual plane, all lines r*, with r $\in$ P \ {p,q}, lie above the intersection point l* of p* and q*. -> the condition p*$\cap$q* is a vertex of LE(P*).
- Similarly, lower convex hull of P corresponds to upper envelope of P*.



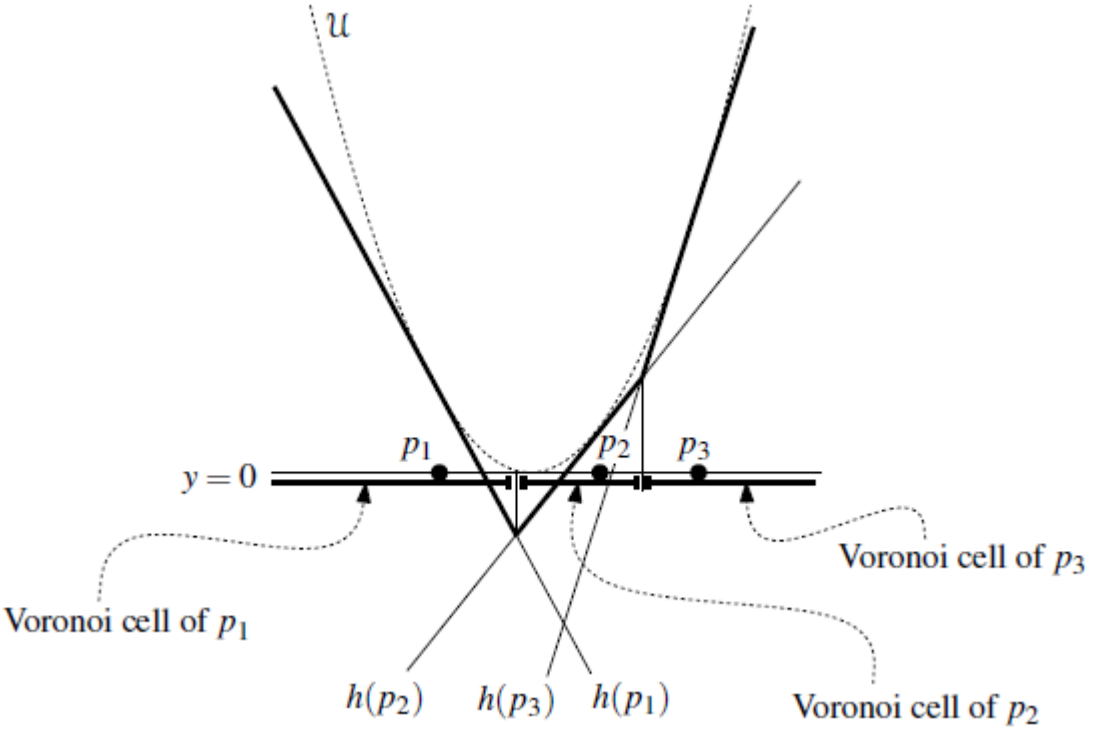
Voronoi diagram and half-space intersection

- U:=($z=x^2+y^2$) : the unit paraboloid
- p := (p_x, p_y,0)
- p' := (p_x, p_y, $p_x^2+ p_y^2$)
- h(p) : nonvertical plane $z = 2p_x x + 2p_y y - p_x^2 - p_y^2$
- h(p) contains the point p'.
- q := (q_x, q_y,0)
- q' := (q_x, q_y, $q_x^2+ q_y^2$)
- The vertical line through q intersects h(p) in $q(p):=(q_x, q_y , 2p_x q_x + 2p_y q_y - p_x^2 - p_y^2)$.
- The vertical distance between q' and q(p) is $q_x^2+ q_y^2 -2p_x q_x - 2p_y q_y + p_x^2+ p_y^2 = dist(p, q)^2$.
- Hence, the plane h(p) and U encodes the distance between p and any other point in the plane z=0.



Voronoi diagram and half-space intersection

- P : planar point set lying in z=0
- H:= { h(p) | p ∈ P}
- UE(H) : upper envelope of the planes in H
- Projection of UE(H) on the plane z=0 is the Voronoi diagram of P.



Voronoi diagram and half-space intersection

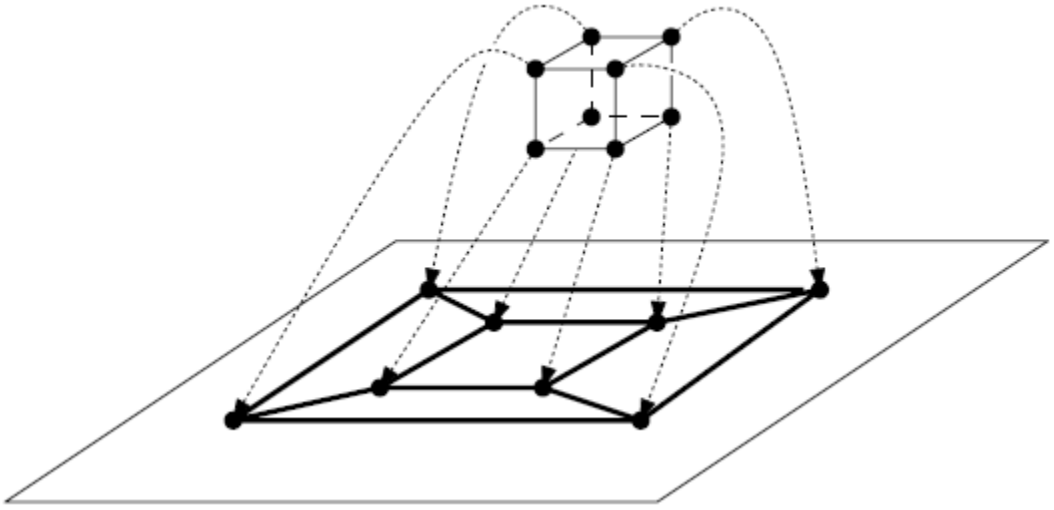
- Theorem : Projection of UE(H) on the plane z=0 is the Voronoi diagram of P.
- Pf) Show that the Voronoi cell of a point $p \in P$ is the projection of the facet of UE(H) that lies on the plane h(p).
- Let q be a point in the plane z=0 lying in the Voronoi cell of p.
- dist(q, p) < dist (q, r) for all $r \in P\backslash\{p\}$.
- Need to prove that the vertical line through q intersects UE(H) at a point lying on h(p).
- For a point $r \in P$, the plane h(r) is intersected by the vertical line through q at the point $q(r):=(q_x, q_y , q_x^2+ q_y^2 - \text{dist}(q, r)^2)$.
- Of all points in P, p has the smallest distance to q, so q(p) is the highest intersection point.

Voronoi diagram and convex hulls

- We can compute a Voronoi diagram in the plane by computing the upper envelope of a set of planes in 3-space.
- The upper envelope of a set of planes in 3-space corresponds to the lower convex hull of the points H*.
- Thus, we can use the 3-D convex hull algorithm to compute planar Voronoi diagram.
- The projection of the lower convex hull of H* on the plane z=0 is the Delaunay graph of P.

Complexity of 3D-convex hull

- We can interpret the boundary of a 3D convex hull as a planar graph.
- So we can use Euler theorem



Complexity

Theorem

Let P be an convex polytope with n vertices. The number of edges of P is at most $3n-6$, and the number of facets of P is at most $2n-4$.

$$n - n_e + n_f = 2$$

$$2n_e \geq 3n_f$$

->

$$n_f \leq 2n-4 \text{ and } n_e \leq 3n-6$$

Corollary

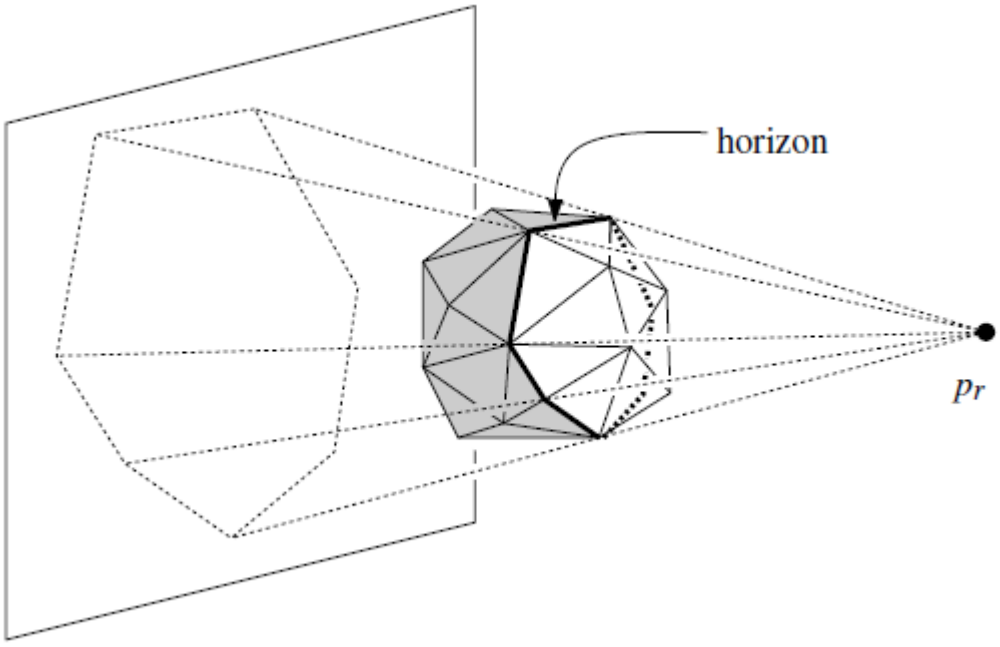
The complexity of the convex hull of a set of n points in 3-dimensional space is $O(n)$.

Randomized Incremental Algorithm

1. Choose 4 points in P that do not lie in a common plane, so that their convex hull is a tetrahedron.
2. Compute a random permutation of the remaining points.
3. To update $CH(P_{r-1})$ -> $CH(P_r)$
   - if P_r is in $CH(P_{r-1})$ -> nothing to be done
   - if P_r is outside $CH(P_r)$ -> need to update CH

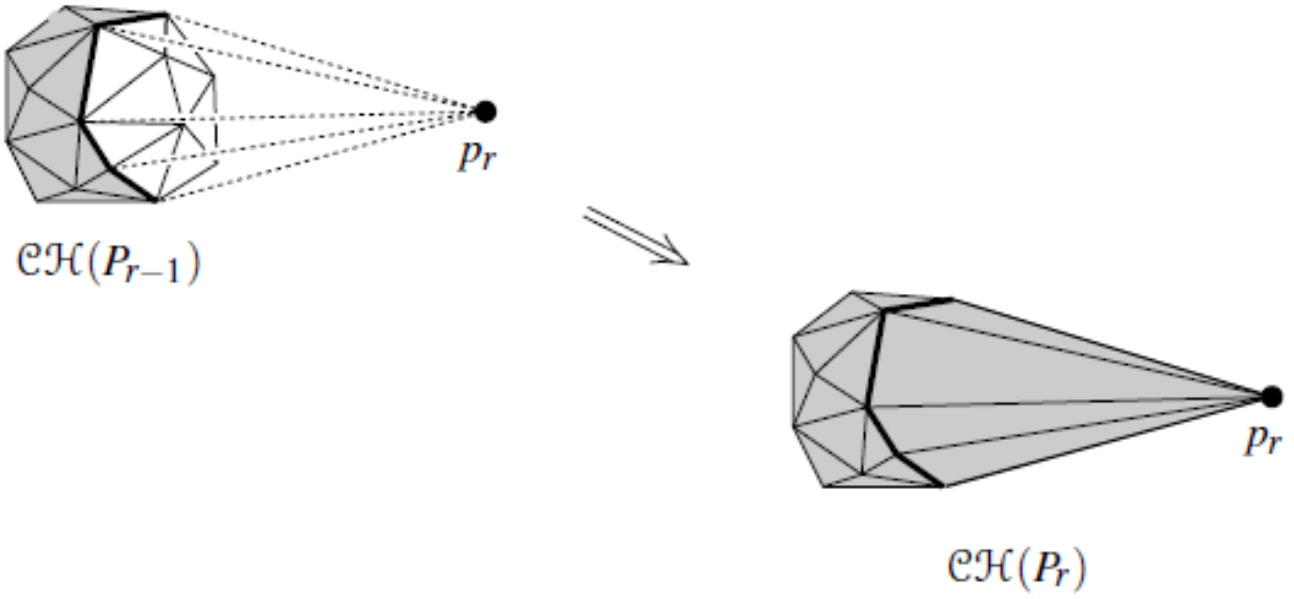
Horizon

- Closed boundary curve of visible facets region



Add a point to the convex hull

- Connect p_r to the point on the horizon : visible facets are deleted.



bottleneck

Visibility test

: we can find all visible facets from p_r

in O(r)

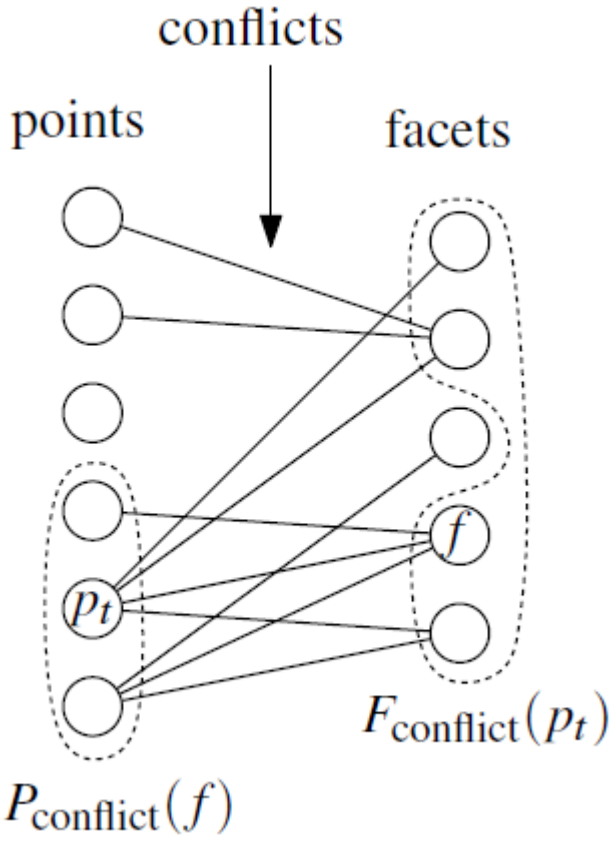
-> $O(n^2)$ algorithm.

Can we do better?

Conflict Graph

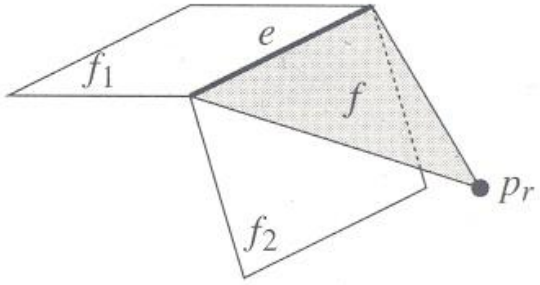
For each facet f of current convex hull, maintain points that can see f.

1. Bipartite graph (points not yet inserted vs. current facets)
2. $P_{conflict}(f)$: points not yet inserted that can see f
3. $F_{conflict}(P_t)$: current facets that point P_t can see.
4. Connect edge if visible



Update Conflict Graph

- Initialize: linear time
- Discard nodes of facets visible from p_r
- Discard p_r
- Add new facets node
- Add new edges
  - Key idea: a point which can see new facet ‘f’ also can see old facet ‘f1’ or ‘f2’ or both.



Algorithm

1. Find a initial tetrahedron. (p_1, p_2, p_3, p_4)
2. Random permutation of remaining points.
3. Initialize Conflict Graph
4. Incremental Construction
   - If p_r is inside CH -> do nothing
   - Else
     - Delete visible facets of p_r
     - Add new triangle facets
     - Update conflict graph

Algorithm CONVEXHULL(P)

Input. A set P of n points in three-space.

Output. The convex hull $\mathcal{CH}(P)$ of P.

1. Find four points p_1, p_2, p_3, p_4 in P that form a tetrahedron.
2. $\mathcal{C} \leftarrow \mathcal{CH}(\{p_1, p_2, p_3, p_4\})$
3. Compute a random permutation $p_5, p_6, \ldots, p_n$ of the remaining points.
4. Initialize the conflict graph $\mathcal{G}$ with all visible pairs (p_t, f), where f is a facet of $\mathcal{C}$ and $t > 4$.
5. **for** $r \leftarrow 5$ **to** n
6. **do** ($*$ Insert p_r into $\mathcal{C}$: $*$)
7. **if** $F_{\text{conflict}}(p_r)$ is not empty ($*$ that is, p_r lies outside $\mathcal{C}$ $*$)
8. **then** Delete all facets in $F_{\text{conflict}}(p_r)$ from $\mathcal{C}$.
9. Walk along the boundary of the visible region of p_r (which consists exactly of the facets in $F_{\text{conflict}}(p_r)$) and create a list $\mathcal{L}$ of horizon edges in order.
10. **for** all $e \in \mathcal{L}$
11. **do** Connect e to p_r by creating a triangular facet f.
12. **if** f is coplanar with its neighbor facet f' along e
13. **then** Merge f and f' into one facet, whose conflict list is the same as that of f'.
14. **else** ($*$ Determine conflicts for f: $*$)
15. Create a node for f in $\mathcal{G}$.
16. Let f_1 and f_2 be the facets incident to e in the old convex hull.
17. $P(e) \leftarrow P_{\text{conflict}}(f_1) \cup P_{\text{conflict}}(f_2)$
18. **for** all points $p \in P(e)$
19. **do** If f is visible from p, add (p, f) to $\mathcal{G}$.
20. Delete the node corresponding to p_r and the nodes corresponding to the facets in $F_{\text{conflict}}(p_r)$ from $\mathcal{G}$, together with their incident arcs.
21. **return** $\mathcal{C}$

Analysis

Lemma -1)

The expected number of facets created by ConvexHull *is at most* $6n - 20$.

$\deg(p_r, CH(P_r))$

$=$# of facets created during $CH(P_{r-1}) \rightarrow CH(P_r)$

$=$# of facets deleted during $CH(P_r) \rightarrow CH(P_{r-1})$

$=$# of facets adjacent to p_r

$=$# of edges adjacent to p_r

Analysis

by *Euler* formula

the number of edges with r vertices $\leq 3r-6$

the sum of the degrees of $CH(P_r) \leq 6r-12$

(*randomized incremental*)

the expected *degree* of $p_r \leq 6-12/r = (6r-12)/r$

$$\mathrm{E}[\deg(p_r, CH(P_r))] = \frac{1}{r-4}\sum_{i=5}^{r}\mathrm{E}[\deg(p_i, CH(P_r))] \leq 6$$

$$\text{expected number of facets} = 4+\sum_{r=5}^{n}\mathrm{E}[\deg(p_r, CH(P_r))] \leq 4+6(n-4) = 6n-20.$$

Backwards analysis

- We look at $CH(P_r)$ and imagine removing vertex p_r; the number of facets destroyed = the number of facets created by the insertion of p_r into $CH(P_{r-1})$. The probability that p_r is the last point inserted among P_r is 1/r.
- Backwards analysis is a simple technique used to analyze randomized incremental algorithms. “analyze a randomized algorithm as if it were running backwards in time, from output to input.”

Analysis

Lemma – 2)

Algorithm ConvexHull computes the convex hull of a set P of n points in R^3 in $O(n \lg n)$ expected time, where the expectation is with respect to the random permutation used by the algorithm.

$$E[\sum_{r=5}^{n} card(F_{conflict}(p_r))] = O(n)$$

because the number of facets deleted due to the addition of p_r is O(n)

We need to bound the expected vaule of $\sum_{e} card(P(e))$, where summation is over all horizon edges that appear at some stage of the algorithm.

Configuration space

- A framework for randomized incremental algorithms.
- A configuration space is (X,Π,D,K).
- X : input, a finite set of objects. $|X|=n$
- Π : set of configurations
- D : defining sets
- K : killing sets
- For a configuration $\Delta \in \Pi$, elements of $D(\Delta)$ define Δ, and elements of $K(\Delta)$ are in conflict with Δ.
- $| K(\Delta) |$: conflict size of Δ

Conditions for configuration space

- The maximum card(D(Δ)) is a constant d, called the *maximum degree* of the configuration space.
- Number of configurations sharing the same defining set should be bounded by a constant.
- D(Δ) $\cap$ K(Δ) = $\varnothing$ for all configurations $\Delta \in \Pi$

Active configuration

- A configuration Δ is active over a subset $S \subseteq X$ if $D(\Delta) \subseteq S$ and $K(\Delta) \cap S = \emptyset$.
- $T(S) := \{\Delta \in \Pi : D(\Delta) \subseteq S \text{ and } K(\Delta) \cap S = \emptyset\}$: set of active configurations.
- The goal is to compute T(S).

Trapezoidal maps

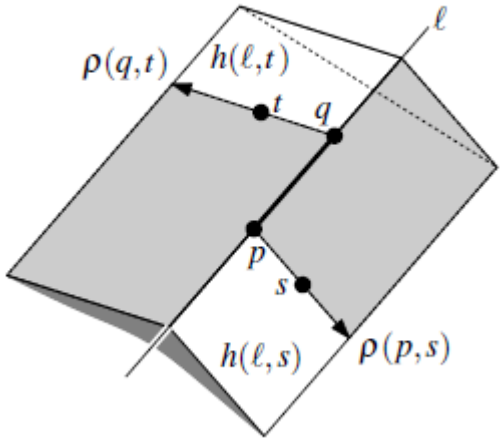
- X : segments
- Π : all trapezoids appearing in the trapezoidal map for any $S \subseteq X$
- D(Δ) : segments defining trapezoid Δ
- K(Δ) : segments that intersect trapezoid Δ
- T(S) : trapezoids of the trapezoidal map of S

Delaunay triangulations

- X : points
- Π : triangles formed by 3 (non-collinear) points in X
- D(Δ) : points that form the vertices of Δ
- K(Δ) : points lying inside the circumcircle of Δ
- T(S) : triangles in DT(S)

3-D convex hulls

- X : set P of points
- A flap Δ is defined as an ordered four-tuple of points (p, q, s, t) that do not all lie in a plane.
- D(Δ) : { p, q, r, s }
- A point x $\in$ X is in K(Δ) if and only if it lies in one of the following regions :
  - outside the closed 3-d wedge defined by h(l, s) and h(l, t)
  - inside h(l, s) but outside the closed 2-d wedge defined by ρ(q, t) and by ρ(q, t) and ρ(p, s)
  - inside h(l, t) but outside the closed 2-d wedge defined by ρ(q, t) and ρ(q, p)
  - inside the line l but outside segment pq
  - inside the half-line ρ(p, s) but outside segment ps
  - inside the half-line ρ(q, t) but outside segment qt



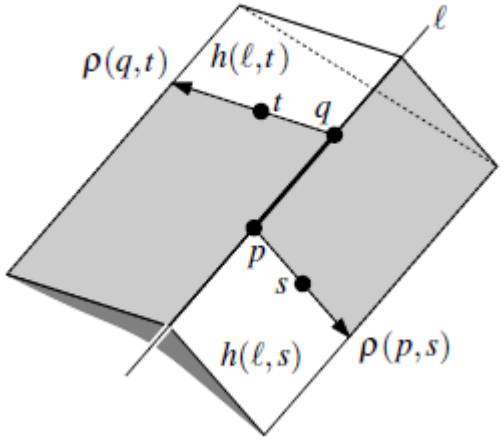
*Lemma*3)

A flap $\Delta = (p,q,s,t)$ *is in* $\mathrm{T}(S)$ *if and only if* $\overline{pq}, \overline{ps}$, *and* $\overline{qt}$ *are edges of the convex hull CH(S), there is a facet f1 incident to* $\overline{pq}$ *and* $\overline{ps}$, *and a different facet f2 incident to* $\overline{pq}$ *and* $\overline{qt}$.

Furthermore, if one of the facets f1 or f2 is visible from a point $x \in P$ *then* $x \in \mathrm{K}(\Delta)$

*Theorem*9.15)

$$\sum_{\Delta} card(\mathrm{K}(\Delta)) \le \sum_{r=1}^{n} d^2 \left(\frac{n-r}{r}\right)\left(\frac{E[card(\mathrm{T}(X_r))]}{r}\right)$$



*Lemma*4)

The expected vaule of $\sum_{e} card(P(e))$, *where summationis*

over all horizon edges that appear at some stage of the algorithm,

is $O(n \lg n)$

$$\sum_{e} card(P(e)) \le \sum_{\Delta} card(\mathrm{K}(\Delta)) \le \sum_{r=1}^{n} d^2 (\frac{n-r}{r})(\frac{E[card(\mathrm{T}(P_r))]}{r})$$

$$\le \sum_{r=1}^{n} 4^2 (\frac{n-r}{r})(\frac{6r-12}{r}) = O(n \log n)$$