Higher Dimensional Convex Hull Algorithms

1 Convex hulls

1.0.1 Definitions

The following definitions will be used throughout. Define

- S_d: A d-Simplex
  The simplest convex polytope in $\mathbb{R}^d$. A d-simplex is always the convex hull of some $d+1$ affinely independent points. For example, a line segment is a $1-simplex$ i.e., the smallest convex subspace which contains two points. A triangle is a $2-simplex$ and a tetrahedron is a $3-simplex$.
- P: A Simplicial Polytope.
  A polytope where each facet is a $d-1$ simplex. By our assumption, a convex hull in 3-D has triangular facets and in 4-D, a convex hull has tetrahedral facets.
- $\mathcal{P}$: number of facets of P_i
- F: a facet of P
- R: a ridge of F
- G: a face of P
- $f_j(P_i\ )$: The number of j-faces of P . Note that $f_j(S_d) = C\binom{d+1}{j+1}$.

1.0.2 An Upper Bound on Time Complexity: Cyclic Polytope

Consider the curve M_d in $\mathbb{R}^d$ defined as $x(t) = (t^1, t^2, ..., t^d)$, $t \in \mathbb{R}$. Let H be the hyperplane defined as $a_0 + a_1x_1 + a_2x_2 + ... + a_dx_d = 0$. The intersection of M_d and H is the set of points that satisfy $a_0 + a_1t + a_2t^2 + ...a_dt^d = 0$. This polynomial has at most d real roots and therefore d real solutions. $\rightarrow H$ intersects M_d in at most d points. This brings us to the following definition: A convex polytope in $\mathbb{R}^d$ is a **cyclic polytope** if it is the convex hull of a

set of at least $d+1$ points on M_d.

Note the following related theorem:
A cyclic polytope has $C\binom{n}{k}$ $(k-1)$ faces for $0 \leq k \leq \frac{d}{2}$.

For a cyclic polytope, every d points creates a dimension $d-1$ face. Therefore, in d dimensions, given n points, there are at most $C\binom{n}{d}$ facets. Or, for dimension d,

Theorem 1.1 $f_{j-1} = C\binom{n}{j}, j \leq \frac{d}{2}$

From this, we conclude that **a convex polytope of dimension d can have at most $O(n^{\lfloor \frac{d}{2} \rfloor})$ facets**. Therefore, the lower bound for finding the convex hull of a set of points in dimension d is $\Omega(n^{\lfloor \frac{d}{2} \rfloor})$ (note that for $d = 2, d = 3, \Omega(n \log n)$ is a stronger lower bound, not based on the size of the output).

We also get the following lemma:

Lemma 1.2 *Let P_i be a convex polytope with n vertices in $\mathbb{R}^d$ and let p be a vertex of P . The number of faces of P which contains p in their boundary plus the number of incidences among them is in $O(n^{\lfloor \frac{d-1}{2} \rfloor})$.*

An intuitive explanation is that there is a hyperplane which separates p from P . This hyperplane cuts P in a $(d-1)$-dimensional polytope with at most $n-1$ vertices where each face has p in its boundary.

1.1 The beneath-beyond algorithm

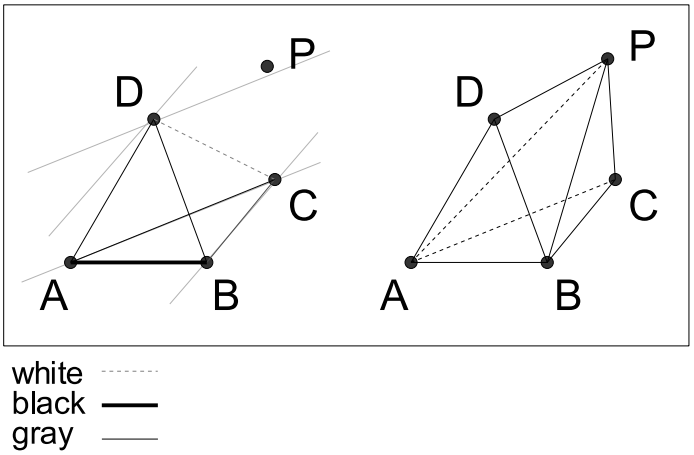

Figure 1: The beneath-beyond method. Illustration of the non-pyramidal update of the tetrahedron $ABCD$ with point P being the latest addition to the convex hull.

This algorithm incrementally builds up the convex hull, keeping track of current convex hull P_i using an incidence graph. Assume that every time a point is chosen to be added to the current convex hull, *it is not in the same affine space* as any of the facets of the current convex hull (see figure 2b). For instance, if the current convex hull is a tetrahedron, a new point to be added will not be coplanar with any of the faces of the tetrahedron.

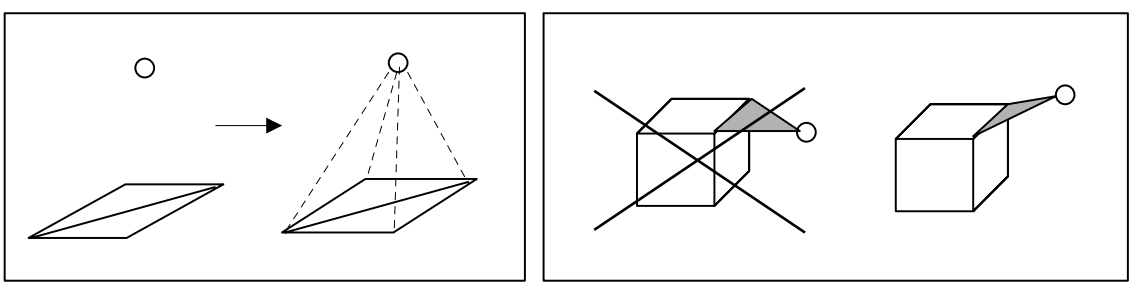

Figure 2: a) Pyramidal update b) Simplifying assumption that new point is not in the affine space of any of the facets of the current convex hull.

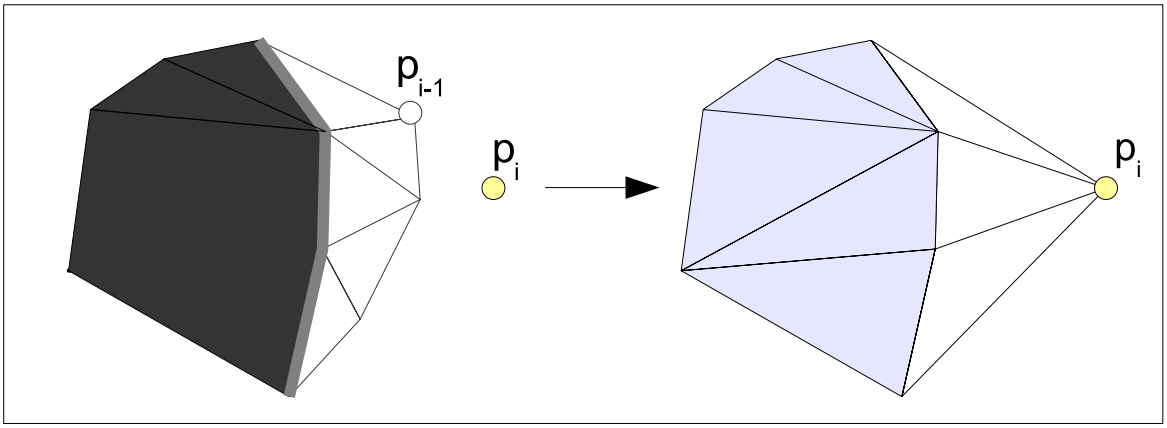

Figure 3: Non-pyramidal update: Consider in $\mathbb{R}^3$, if p_i were a light shining on the current convex hull. All the facets that the light hits (white) will have to be removed. Facets bordering these facets (gray facets) will induce new facets.

1.1.1 Algorithm

Given a set S of n points in $\mathbb{R}^d$,

1. Presort the points along one direction, say x_1. Let $S = \{p_0, p_1, \ldots, p_{n-1}\}$, be the input points after sorting. Process the points in increasing x_1 order.
2. Take the first d points, which define a facet, as the initial hull.
3. Let p_i be the point to be added to the hull at the ith stage. Let $P_i = conv(p_0, p_1, \ldots, p_{i-1})$ be the convex hull polytope built so far. The two kinds of hull updates are **pyramidal** and **non-pyramidal**.

   (a) A **pyramidal update** (see figure 2a) is done when $p_i \notin \text{aff}(p_0, p_1, \ldots, p_{i-1})$ – when p_i is not on the hyperplane defined by the current hull. A pyramidal update consists in adding a new node representing p_i to the incidence graph and connecting this node to all existing hull vertices by new edges .

   (b) A **non-pyramidal update** (see Figure 3) is done when the above condition is not met, i.e., p_i is in the affine subspace define by the current convex hull. In this case, faces that are visible from p_i are removed and new facets are created. Faces are each assigned a

color based on their position relative to p_i. A facet F of P_{i-1} is colored black or white depending on whether F is *seen* by p_i. A facet F is

- **White** iff the hyperplane containing F separates p_i and P_{i-1} .
- **Black** otherwise

A lower dimensional face G is colored either white, black, or gray, depending on the colors of the faces of which it is a subface. A face G is

- **White** if it is a subface of only white faces
- **Black** if it is a subface of only black faces
- **Gray** otherwise

White faces are visible from p_i. Black faces are not visible from p_i. Gray faces are the borders between visible and non-visible faces. The updated hull is obtained as follows. If a face G (includes facets and lower dimensional faces) is colored

i. **Black** $\rightarrow$ G remains in the convex hull
ii. **Gray** $\rightarrow$ G remains in the convex hull and G induces a new face G' with p_i. $G' = CH(G \bigcup p_i)$.
iii. **White** $\rightarrow$ G is deleted

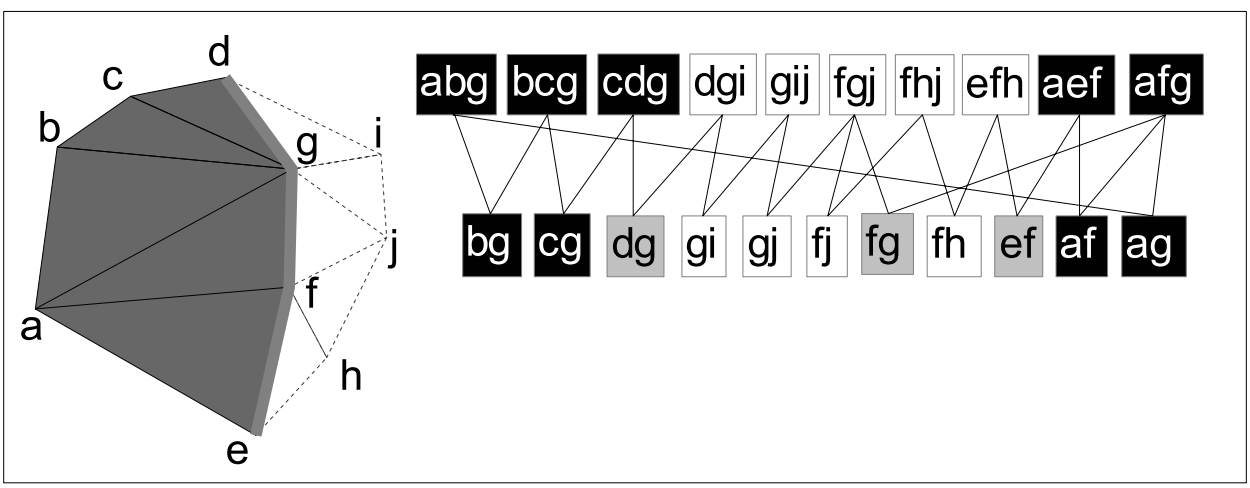

Figure 4:

1.1.2 Algorithm Analysis

Define

- $\mathcal{P}_i$: number of faces of P_i
- I_i: number of faces created in updating P_{i-1} to include p_i (note that since each gray face induces a new face, this is also the number of gray faces).
- D_i: number of faces that get deleted (note that this is also the number of white faces).

If we use a hyperplane H to cut the point p_i off from the rest of the polytope we can use 1.1 to find an upper bound on the number of new faces – the upper bound on the number of $d-2$ faces
$f_{d-2}(P_i) = O(i^{\lfloor \frac{d-1}{2} \rfloor})$ (see Lemma 1.2).

- **Pyramidal Update Time Complexity Analysis**
  Realize that the time complexity for a pyramidal update is linear in the number of faces. Therefore, the time complexity is $O(I_i)$.
- **Non-pyramidal Update Time Complexity Analysis**
  To compute time complexity of a non-pyramidal update, consider the two following facts

  1. One of the facets containing p_{i-1} must be white.
  2. White facets form a connected set in the graph in which a facet is represented by a node and a ridge is represented by an edge

  Therefore, we can compute the time complexity of the algorithm as

  1. $O(i^{\lfloor (d-1)/2 \rfloor})$: Identify one white facet by considering facets containing p_{i-1} (note the set of facets incident upon p_i has dimension $d-1$).
  2. $O(D_i + I_i)$: Determine the remaining white facets using DFS through the incidence graph.
  3. $O(D_i + I_i)$: Determine all the white and gray faces using DFS.
  4. $O(D_i)$: Delete all white faces.
  5. $O(I_i)$: Make new faces from gray faces.

The total amount of work done during non-pyramidal updates is $\sum_{i<n} O(D_i + I_i)$. Since only faces which have already been inserted can be deleted we know that $\sum_{i<n} O(D_i + I_i) \leq \sum_{i<n} O(I_i)$. So we can bound the time spent if we can bound the total number of faces inserted.

How can we bound I_i? Note that P_i contains at most i vertices, since only i points have been processed. Imagine taking a hyperplane and using it to cut p_i off from the rest of the polytope. The remaining polytope would be a $d-1$ dimensional polytope, which can have at most $O(i^{\lfloor (d-1)/2 \rfloor})$ faces (see Lemma 1.2). Thus the time complexity of the incremental algorithm is:

$$\sum_{i<n} O(D_i + I_i) \leq \sum_{i<n} O(I_i) \leq \sum_{i<n} O(i^{\lfloor (d-1)/2 \rfloor}) = O(n^{\lfloor (d+1)/2 \rfloor})$$

The presorting step takes time $O(n \log n)$, so the overall time complexity is $O(n \log n + n^{\lfloor (d+1)/2 \rfloor})$. This is the optimal worst case complexity for even d.

1.2 The Gift-Wrapping method

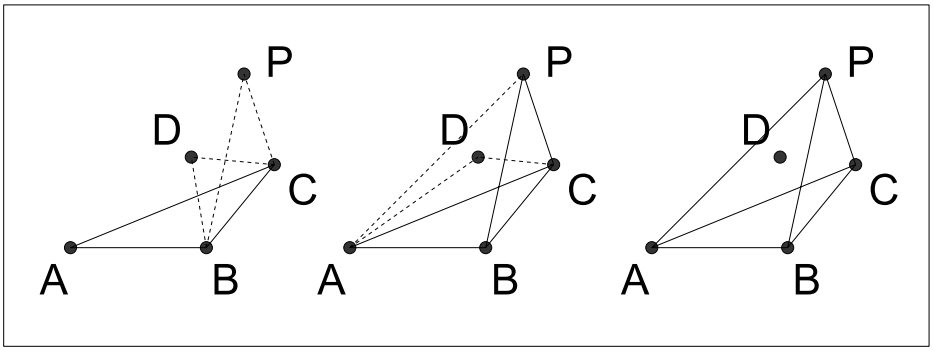

Figure 5: Gift-wrapping. Point D is interior to the tetrahedron formed by points A, B, C, and P, and is therefore not on the convex hull. We start with the triangle ABC and gift-wrap around the edge BC. The point P and facet BCP is added to the hull. Next, we gift-warp around edge AC and add facet ACP to the hull.

This algorithm was proposed by Chand and Kapur in 1970 and is discussed in [1], pg. 125-130. A rough sketch in $\mathbb{R}^3$ is presented here. Define

- $\mathcal{T}$: a "pool" of candidate subfacets for gift-wrapping

The important rules which will be used are:

- Every ridge is contained in exactly two facets.
- Every edge joins exactly two points.
- Given a facet F and a ridge R, one can find the "other" facet F' that contains R in $O(n)$ time.

1.2.1 Algorithm

1. Find some facet F and call its ridges "open" ridges
2. Put open ridges in pool $\mathcal{T}$
3. While $\mathcal{T}$ is not empty

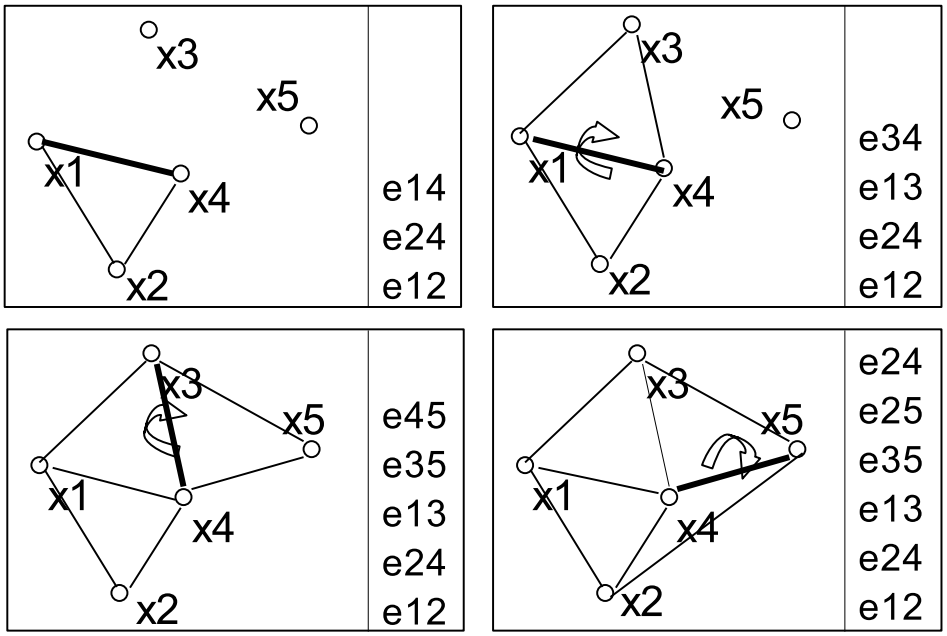

Figure 6: Illustration of discovering facets using the Gift-Wrapping method.

- $O(n)$**: Giftwrap over** R
  Determine, among all $p_i \in S$ which are not vertices of F, the point p' such that all other points are on one side of the hyperplane $\text{aff}(R \cup \{p'\})$.
- $O(c)$**: Update pool of open ridges**
  Add edges of newly discovered face to $\mathcal{T}$.

The gift-wrapping step is shown for a 3-D case in Figure 5.

1.2.2 Algorithm Analysis

Let $\mathcal{P}$be the number of facets of the convex hull of the n input points. Then the time taken by the algorithm is $O(n\mathcal{P})$. Thus the time taken by this algorithm is sensitive to the size of the output. However, also note that we can discover the same facet multiple times since each facets has three ridges. Note that $\mathcal{P}$can be as large as $n^{\lfloor d/2 \rfloor}$, giving a worst-case complexity of $O(n^{\lfloor (d+2)/2 \rfloor})$.

1.3 Seidel's shelling algorithm

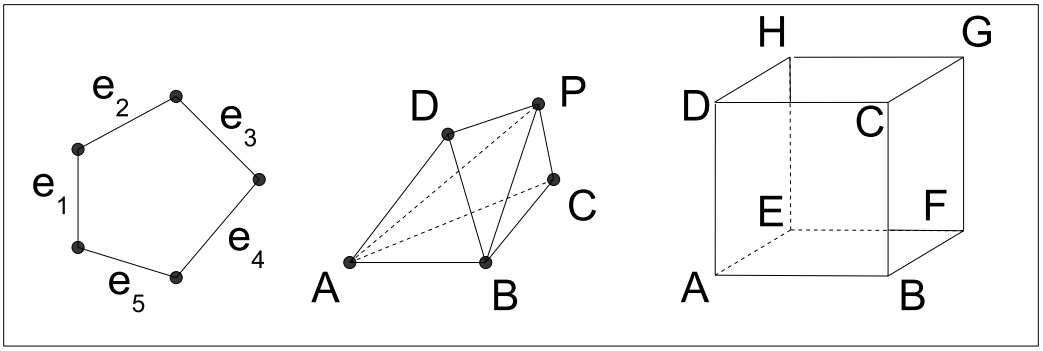

Figure 7: Shelling of a polytope. a) e_1, e_2, e_3, e_4, e_5 is a shelling of the convex polygon but e_1, e_2, e_4, e_5, e_3 is not. b) $ABC, ABD, BDP, BPC, APC, APD$ is a shelling, while $ABC, PBC, ADP, DBP, APC, ABD$ is not, since, for instance, the intersection of facet ADP with the union of facets ABC and PBC does not have any edge in it. c) $ABCD, ABEF, CDGH, EFGH, BCGF, AEHD$ is not a shelling because the intersection of $EFGH$ with the union of facets $ABCD$, $ABEF$, and $CDGH$ consists of the set of edges EF, GH which does not constitute a legal sub-shelling of the facet $EFGH$.

The two previous algorithms for the higher dimensional convex hull problem, the Beneath-Beyond method and the Gift-Wrapping algorithm have complexities of $O(n^{\lfloor (d+1)/2 \rfloor})$ and $O(n\mathcal{P})$ respectively. Depending on output size, gift-wrapping can be as bad as $O(n^{\lfloor (d+2)/2 \rfloor})$. Seidel's shelling algorithm [4] has a complexity of $O(n^2 + \mathcal{P} \log n)$. In the worst case $\mathcal{P} = n^{\lfloor \frac{d}{2} \rfloor}$, so the time complexity is $O(n^2 + n^{\lfloor \frac{d}{2} \rfloor} \log n)$, which is the best bound known for odd $d > 3$.

1.3.1 The shelling of a polytope

The shelling method works by traveling along a line and determining the facets of the convex hull as they become visible.

Consider a d-polytope P with n faces $F_1, F_2, ...F_n$. Imagine traveling along a directed straight line L that intersects the interior of P . Begin at a point on L that is in the interior of P . Continue until you pass through

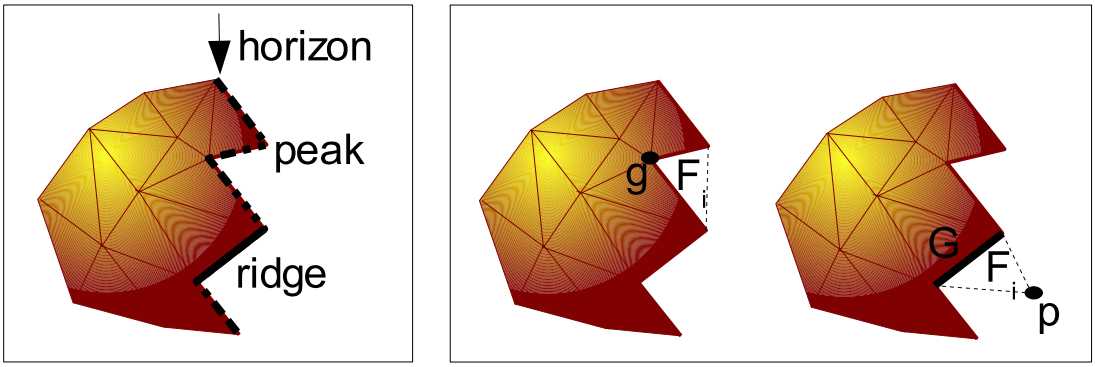

Figure 8: The shell is incrementally created so that it is fully connected. a) The horizon of the shell. b) The next facet to be discovered can intersect the horizon in one of two ways – at multiple ridges (left) or at a single ridge (right).

some facet F. This facet F is the first in the shelling order. Continue moving away from P along L and more facets will become visible. The order in which these facets appear is the shelling order. When no more facets are visible, jump back (or "move through infinity") to the "beginning" of L. All the facets of P that were not visible before are now visible. Still moving along L, but now towards P , facets will become invisible. The order in which they disappear constitutes the second part of the shelling order.

A **shelling** of a d-polytope P is an ordering of the facets of P , say $F_1, F_2, \ldots, F_m$ ($m = f_{d-1}(P)$, the number of $d-1$ faces or facets of P), so that for each i, $1 \leq i \leq n$,

$$F_i \cap \left(\bigcup_{j<i} F_j \right)$$

yields the non-empty initial portions of some shelling of P_i .

1.3.2 Linear Programming for Shelling

The shelling line L is parameterized by $x(t) = -\mathbf{a}/t$, $-\infty < t < +\infty$. The vector $\mathbf{a}$is the direction of L.

We can use linear programming to determine for every $p \in$S whether p is a vertex of P and, if it is, the first facet in the shelling that contains p. Let H_p be the hyperplane that contains that facet. All such H_p's are included in

$\mathbf{H}_i$. H_p is such that $\forall q \in$S, $q \in H_p^+$ and H_p intersects L "first." We need to find $\mathbf{n} = (n_1, n_2, \ldots, n_d)$ such that:

- $\langle q - p, \mathbf{n} \rangle \geq 0$
  The hyperplane through p keeps all $q \in$S to one side. Recall that $\langle a, b \rangle = |a||b|cos(\theta)$. Therefore, $\langle a, b \rangle \geq 0$ if $\frac{-\pi}{2} \leq \theta \leq \frac{-\pi}{2}$. We can insist that $\langle p, \mathbf{n} \rangle = 1$.
- $\langle \frac{-\mathbf{a}}{t} - p, \mathbf{n} \rangle = 0$. Remember that $x(t) = -\frac{\mathbf{a}}{t}$.
- Combining the above items:
  $\langle \frac{-\mathbf{a}}{t} - p, \mathbf{n} \rangle = 0$
  $\langle \frac{-\mathbf{a}}{t}, \mathbf{n} \rangle + \langle -p, \mathbf{n} \rangle = 0$
  $\langle \frac{-\mathbf{a}}{t}, \mathbf{n} \rangle = \langle p, \mathbf{n} \rangle$
  $\langle \frac{-\mathbf{a}}{t}, \mathbf{n} \rangle = 1$
  $\frac{-1}{t} \langle \mathbf{a}, \mathbf{n} \rangle = 1$
  $t = -\langle \mathbf{a}, \mathbf{n} \rangle$
  That is, $x(t)$ is on the same hyperplane as p.

So the linear program is: $\underline{t = -\langle \mathbf{a}, \mathbf{n} \rangle,\ \langle p - q, \mathbf{n} \rangle \geq 0,\ \langle p, \mathbf{n} \rangle = 1}$ If for some p, the LP is infeasible, then p is interior to CH(S). Otherwise, we can discover the first facet in the shelling order containing p.

1.3.3 A straight line shelling

Define

- L: A shelling line
- T: A set of horizon peaks
- $\mathbf{H}_i$: A set of hyperplanes
- H: A single hyperplane
- Horizon face: A face G of fch a *horizon face* at (time) t iff G is contained in two facets F_i and F_j of fch with $t_i < t$ and $t_j \geq t$. Thus a horizon face is part of one face already seen along the shelling line and one face not yet seen.
- G: Horizon ridge: A $(d-2)$-face
- g: Horizon peak: A $(d-3)$-face
- Horizon (at time t): The set of all horizon faces at time t.

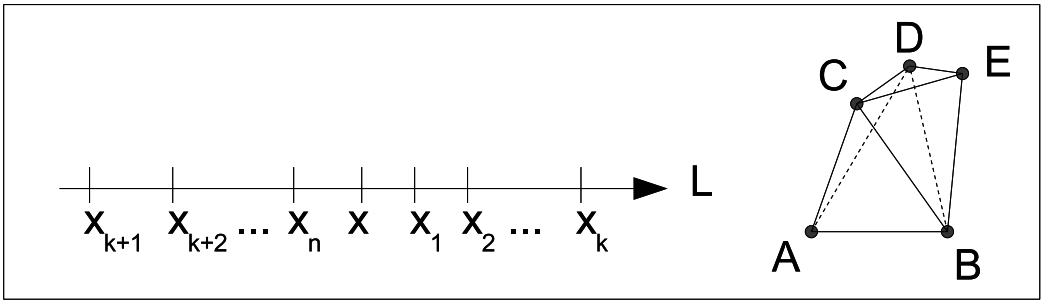

Figure 9: Shelling of a polytope. a) Shelling line. b) Determination of next facet in shelling order. Assume that the shelling order is (ABC, ACD, BCE, CDE, ABD, BDE). Then, after ABC has been discovered, the next facet ACD has only one ridge (AC) intersecting with ABC. The next facet BCE has also just one ridge (BC) intersecting with the union of the first two facets. The fourth facet CDE has two ridges, CD and CE intersecting with the union of the first three facets and these ridges along with the vertex C uniquely determine the facet CDE.

1.3.4 Using shelling to construct facets of the convex hull

Assumptions: No $d+1$ points lie in a common hyperplane. This assumption implies that P , the convex hull, is a simplicial d-polytope, i.e., all facets of P are $(d-1)$-simplices.

1.3.5 Algorithm

1. For each point $p \in$S solve a linear program with $n-1$ constraints and $d-1$ variables.

   (a) If the program is infeasible then eliminate p from S. It cannot be a vertex of CH(S).

   (b) If it is feasible then enter in the priority queue the time t_p and the set of points Q_p that when unioned with p that formed the hyperplane.

2. Let t be the minimum value in the priority queue. Report the facet corresponding to t as the first facet. Delete the t_p values from the priority queue.

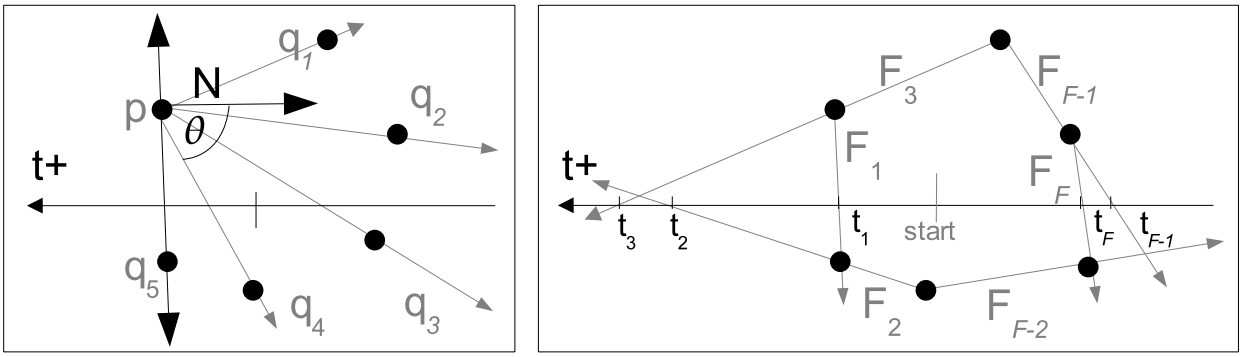

Figure 10: a) Given a point p, the linear program finds the point q_5 whose facet with p interesects the shelling line "first" and keeps all other points to one side ($\langle q_4 - p, \mathbf{n} \rangle > 0 \leftrightarrow cos(\theta) > 0$). b) Illustration of discovering facets in the shelling order.

3. Process each time in the priority queue. For each time t, there is an associated facet F to find. This facet can join the existing horizon in one of two ways: For each t in the priority queue, the still unknown facet will have order.

   (a) F intersects the horizon in more than 1 ridge. Therefore, F contains some horizon peak G and its two horizon ridges R_1, R_2. But, $dim(R_1 \cup R_2) = dim(R_1) + dim(R_2) - dim(R_1 \cap R_2)$ (note $R_1 \cap R_2 = G = (d-2) + (d-2) - (d-3) = d-1$ (a hyperplane)

   (b) The intersection consists of exactly one horizon ridge G. In this case we cannot deduce F_i from the structure of the horizon. However, F_i must contain a vertex $p \in$S that is not contained in G. Also, p is not contained in any F_j with $j < i$. In other words, F_i is the first facet in the shelling that contains p.

   For the algorithm, We proceed as follows. Find the minimum time t and its associated data. If

   (a) There is a single point p with the minimum time. This corresponds the F_i intersecting the horizon along a single ridge. Report $Q_p \cup p$ as the next facet.

In this case, the horizon changes by adding a new peak. We therefore have to compute the hyperplanes generated by this peak and its neighbor peaks and add these new hyperplanes (ie their times t to the priority queue.

(b) There are multiple points with the minimum time. This corresponds to F_i intersecting the horizon along multiple ridges and the multiple points are the horizon peaks. The union of the horizon peaks is the next facet.
The horizon has been updated by adding a new ridge. We use linear programming to find the point p that with that ridge will create facet of P. We add this hyperplane to the priority queue.

1.3.6 Algorithm Analysis

If the dimension d is fixed, each of the n linear programs can be solved in $O(n)$ time using Meggido's algorithm [5]. Thus the first step can be performed in $O(n^2)$ time. Each time a facet is reported the priority queue (of size at most $n^{\lfloor d/2 \rfloor}$) is updated. For fixed d, this takes time $O(F \log n)$ where $\mathcal{P}$is the number of facets reported. Thus the overall complexity is $O(n^2 + \mathcal{P} \log n)$. If $\mathcal{P}$is superlinear, this beats gift-wrapping. In the worst case, it is the best algorithm for odd d but is not provably optimal.

References

[1] F.P. Preparata, M.I. Shamos. *Computational Geometry*. Springer-Verlag, 1985.

[2] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, 1987.

[3] CS672 class notes, lecture 9. March, 1989.

[4] R. Seidel. *Constructing Higher-Dimensional Convex Hulls at Logarithmic Cost per Face*. STOC 86, 404-413.

[5] N. Meggido. *Linear Programming in Linear Time when the Dimension is Fixed*. JACM 31 (1984), 114-127.