Professor Diane Souvaine
Scribe: Mikhail Urinson

Convex Hull in Higher Dimensions

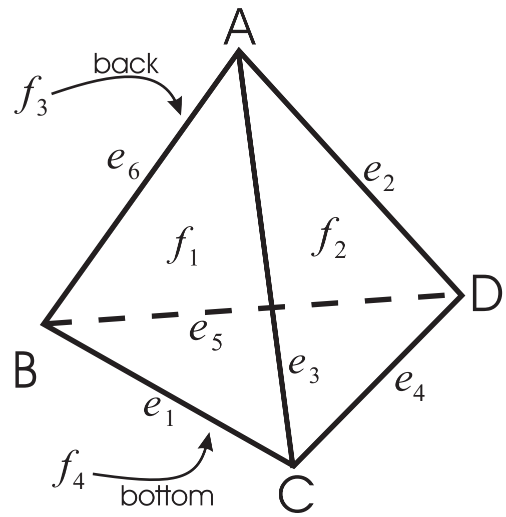
1 Introduction

This lecture describes a data structure for representing convex polytopes and a divide and conquer algorithm for computing convex hull in 3 dimensions. Let S be a set of n points in $\Re^3$. Convex hull of S ($CH(S)$) is the smallest convex polytope that contains all n points. Since the boundary of this polytope is planar, it can be efficiently represented by the data structure described in the next section (only true for 3D).

2 Data Structure

Doubly-Connected Edge List (DCEL) [1] is a data structure for representing polytopes in 3D (see Figure 1 for DCEL representation of tetrahedron). It consists of 3 lists, containing the following data:

1. *Edge List*
   - Vertices: Source and Target
   - Faces: Left and Right
   - Edges: Next (with respect to left face) and Previous (with respect to right face)
2. *Face List*
   - Normal vector
   - One adjacent edge
3. *Vertex List*
   - Coordinates
   - One adjacent edge



Vertex List

	x	y	z	One Edge
A				2
B				1
C				3
D				2

Edge List

	Vertex		Face		Edge	
	S	T	L	R	Next	Prev
e1	B	C	1	4	3	5
e2	D	A	2	3	3	5
e3	A	C	2	1	4	6
e4	C	D	2	4	2	1
e5	B	D	4	3	4	6
e6	A	B	1	3	1	2

Face List

	Normal			One Edge
	x	y	z	
f1				1
f2				4
f3				6
f4				5

Figure 1: DCEL Representation of Tetrahedron

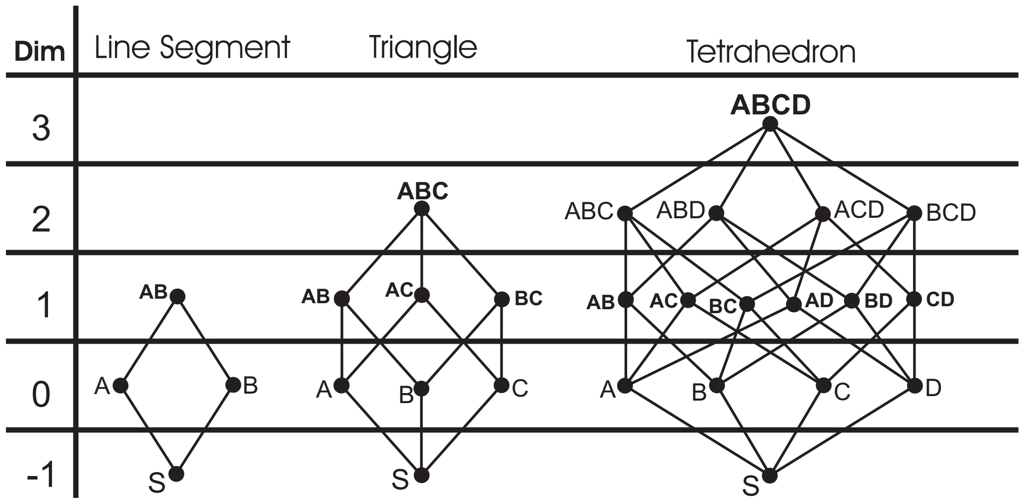


Figure 2: Incidence Graphs of Simplexes

DCEL groups primitive components of a polytope into sets according to their dimensionality and provides linkage between adjacent dimensions. Incidence graph is a similar data structure for representing polytopes. Incidence graph consists of $d + 2$ partitions, corresponding to $-1 \leq k \leq d$ dimensions (5 partitions in 3D). Partitions -1 and d consist of a single element each, the "source" and the represented polytope respectively. Nodes of partition $k, 0 \leq k \leq d - 1$ are the k-faces of the polytope: 0-face is a vertex, 1-face is an edge, 2-face is a face etc. (see examples in Figure 2) Edges of an incidence graph connect incident elements of adjacent dimensions, which makes it equivalent to DCEL (even though data stored by the two data structures explicitly is different).

Note that reversing the order of dimensions, which is equivalent to flipping an incidence graph upside down, gives a geometric dual of a polytope. Incidence graphs are good for visualising this operation. Each of the simplexes shown in Figure 2 has an equivalent geometric dual. A cube and its geometric dual, an octahedron, are shown in Figure 3, and cube's incidence graph is given in Figure 4.

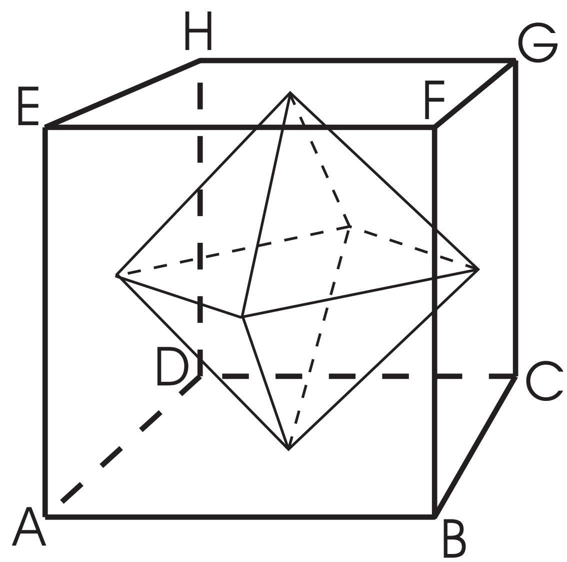


Figure 3: Cube and its Geometric Dual

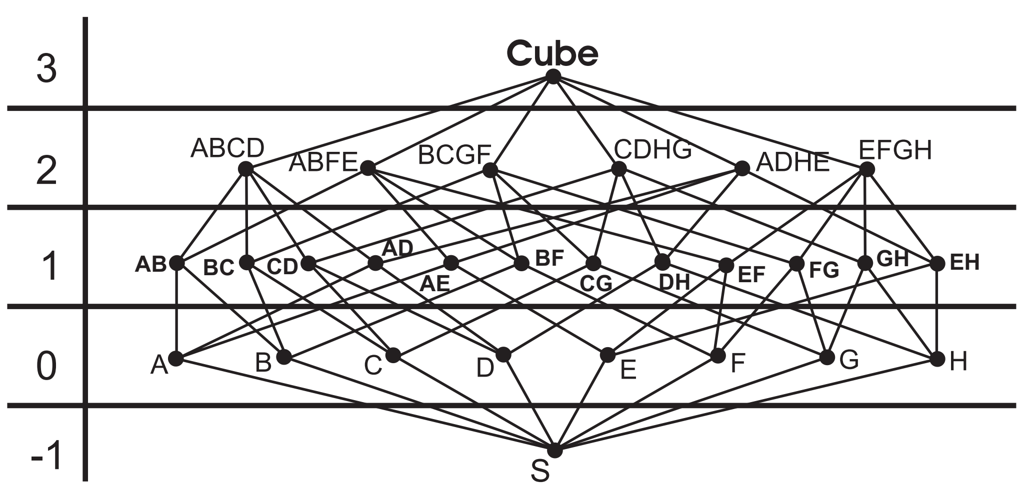


Figure 4: Cube Adjacency Lattice

3 Divide and Conquer Algorithm (Preparata and Hong)

Given a set of points S in $\Re^3$, presort them with respect to x_1-coordinate and let P represent the resulting order. Call *ConvexHull(P, n)* given below.

Algorithm 1 *ConvexHull(P, n)*

if n $\leq$ 7 **then**
  Compute $CH(P)$ by brute force
else
  DIVIDE:
    $k = \lfloor n/2 \rfloor$
    $P_1 = \{p_1, p_2 ... p_k\}$
    $P_2 = \{p_{k+1}, p_{k+2} ... p_n\}$
  RECUR:
    $ConvexHull(P_1, k)$
    $ConvexHull(P_2, n - k)$
  MERGE:
    $CH(P) = Merge(CH(P_1), CH(P_2))$

To illustrate the algorithm, suppose P_1 is a tetrahedron and P_2 is a cube as shown in Figure 5.

Note: There exists a hyperplane H_0 orthogonal to x_1-axis such that H_0 separates P_1 and P_2. H_0 intersects CH(P) in a 2D convex polygon (see Figure 6). Each facet and edge of $CH(P)$, which is not a facet or edge of P_1 or P_2 **must** intersect H_0. These facets define a "sleeve".

Assuming that all facets are triangles, the following facets of $P_1(P_2)$ should be removed:

1. Any facet F of P_1, for which there is a vertex q of P_2 such that q is on the "wrong" side of the plane containing F. Call such a facet "red".

2. Any edge e of P_1, which is contained only in red facets unless e is a border edge. Call these edges "red".

3. Any vertex v of P_1, which is only incident to red edges unless v is part of the border.

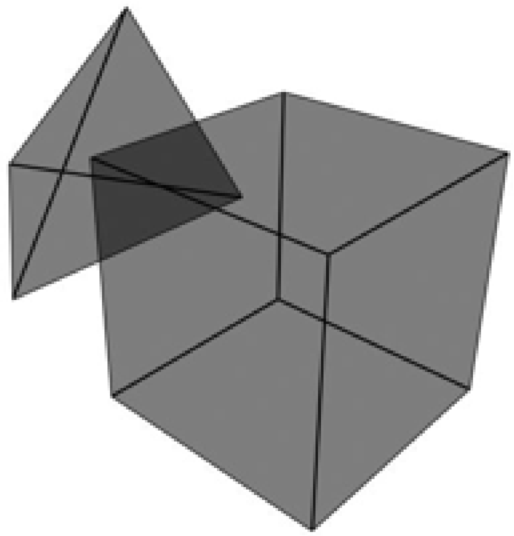


Figure 5: Merging a Tetrahedron and a Cube

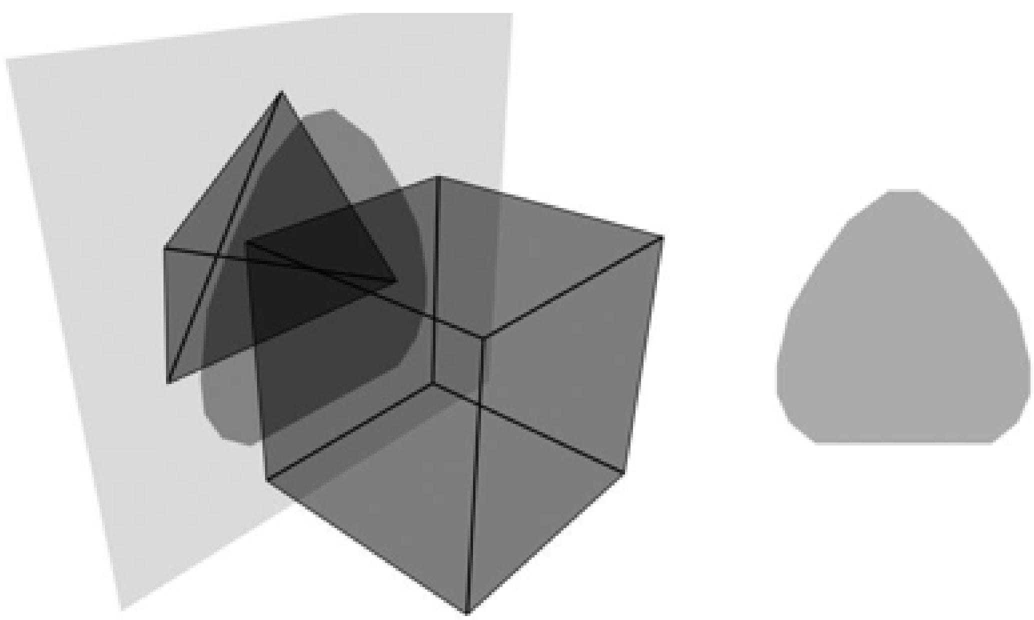


Figure 6: Intersection of H_0 and CH(P)

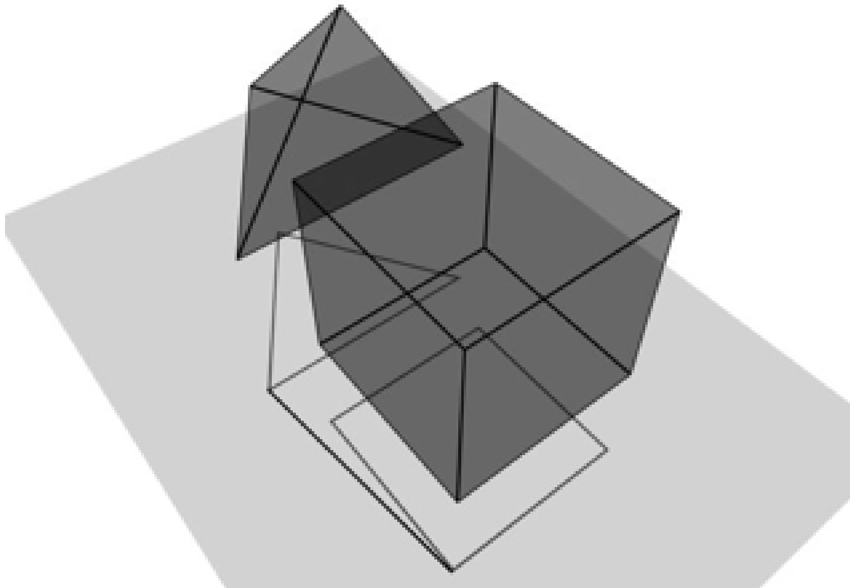


Figure 7: Projection of a Sleeve Edge

Claim 3.1 *The red facets and edges of P_1 form a connected component (proof by induction on a dual graph).*

Claim 3.2 *One red facet of P_1 can be found in $O(|P_1|)$ time as follows. Take vertex v of P_1 with maximum x_1-coordinate and some vertex w of P_2. Find a facet of P_1 that contains v and for which w is beyond.*

Claim 3.3 *If the border of the sleeve is known, all red facets of P_1 can be found in $O(|P_1|)$ time. Find one red facet as described above and perform depth-first search, backtracking at border edges.*

Determining the Sleeve

1. Orthogonally project P_1 and P_2 onto x_1-x_2 plane X. Produce convex polygons Q_1 and Q_2 separated by $X \cap H = l$.

2. Find bridge over l. This bridge is the projection of a "new" edge of the sleeve. Call this sleeve edge (p_{init}, q_{init}). (see Figure 7)

3. Assume $e = (p, q)$ is a non-bridge edge of the sleeve. We want to find the "new" sleeve facet that contains e.

4. Let $CCW(p) = \{p_0, p_1 ... p_\lambda\}$ be vertices of P_1 adjacent to p in CCW order.
   Let $CW(q) = \{q_0, q_1 ... q_\rho\}$ be vertices of P_2 adjacent to q in CW order.

5. Suppose $p_t \in CCW(p)$ and $q_t \in CW(q)$.

6. Search $CCW(p)$ starting at p_t for the first p_i, such that the hyperplane H_1 spanned by p, q, and p_i keeps p_{i-1} and p_{i+1} on the same side.
   Search $CW(q)$ starting at q_t for the first q_j, such that the hyperplane H_2 spanned by p, q, and q_j keeps q_{j-1} and q_{j+1} on the same side.
   H_1 is tangent to P_1 at $\overline{pp_i}$.
   H_2 is tangent to P_2 at $\overline{qq_j}$.

7. Select either H_1 or H_2 as follows:
   (a) Pick H_1 iff q_j is on the same side of H_1 as p_i.
   (b) Pick H_2 iff p_i is on the same side of H_2 as q_j.
   (c) Assume we selected H_1, then
      - $\{p, q, p_i\}$ spans a facet of the sleeve
      - $\{p, p_i\}$ is a border edge
      - $\{p_i, q\}$ is a "new" (non-border) edge of the sleeve.
   (d) Gift-wrap over this new edge.
   (e) In $CCW(p)$ start search at p_i.
   (f) In $CW(q)$ start search at q_j.

8. Repeat steps 3 - 8 until (p_{init}, q_{init}) is reached.

Claim 3.4 *For every vertex p (q) of the sleeve, $CCW(p)$ and $CW(q)$ is traversed in total at most once. Therefore, time necessary to find sleeve is $O(\sum deg(p) + \sum deg(q)) = O(|P_1| + |P_2|)$. Hence, merge takes linear time.*

Steps 1 and 2 (projection and finding a bridge in 2D) take linear and $\log(|P_1| + |P_2|)$ time respectively. The two searches in Step 6 take linear time because each vertex p and q (in $CCW(p)$ and $CW(q)$ respectively) is

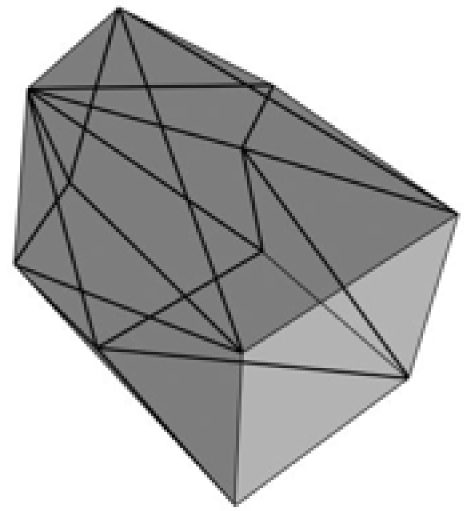


Figure 8: The Sleeve

traversed in total at most once: $O(\sum_p deg(p) + \sum_q deg(q)) = O(|P_1| + |P_2|)$. Since all interior (red) facets can be removed in linear time as described above (Claims 3.1-3.3), the whole merge takes linear time. Hence, the recurrence equation for the time complexity of the *ConvexHull* algorithm is $T(n) = 2T(n/2)+n$. Therefore, the running time of the divide and conquer algorithm for convex hull in 3 dimensions is $O(n \log n)$.

References

[1] D.E. Muller and F.P.Preparata, Finding the intersection of two convex polyhedra, *Theoretical Computer Science* 7:217-236, 1978.

[2] F.P. Preparata and S.J. Hong, Convex Hulls of finite sets of points in two and three dimensions, *Communications of the ACM*, 20:87-93, 1977.