[Convex Hull in 3D]
Computational Geometry (CSc 635)

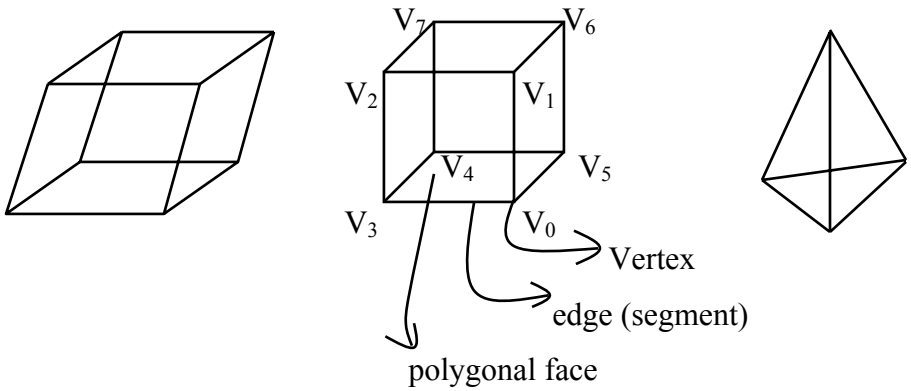
Jagdish Bhatta

Central Department of Computer Science & Information Technology
Tribhuvan University

Convex Hull in Three Dimensions

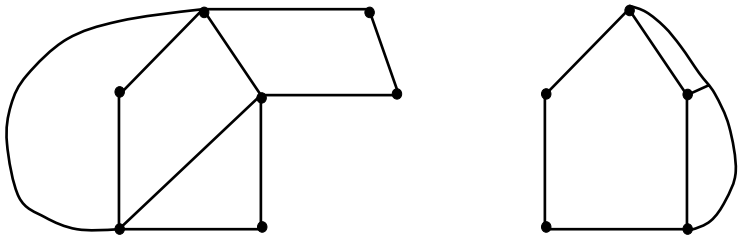
Polyhedra: A polyhedron is a generalization of two dimensional polygon to three dimensions. It is a region of space whose boundary consists of finite number of flat polygonal faces, any pair of which is either disjoint or meet at edges & vertices.

The surface of each polyhedron consists of vertices, line segments & polygonal faces.



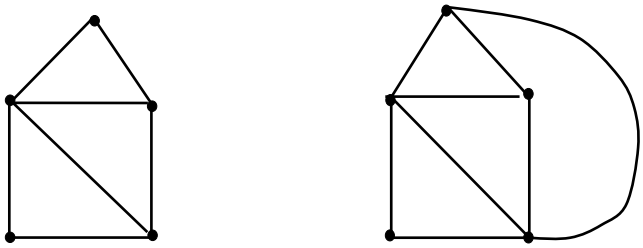
(Thus, a polyhedron is a region of space bounded by a finite set of polygons)

Planar graph: A graph G = (V, E) is called planar if it can be drawn in the plane without intersecting its edges.



Planar graph Non-planar graph

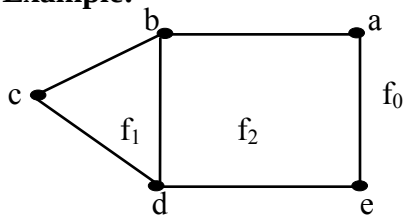
Planar Geometric Graph: Planar geometric graph is a planar graph in which all edges are Euclidean straight lines.



P.G. graph planar graph but not planar geometric graph

These planar graph subdivides the plane into different regions; 0-dimensiional vertices, 1-dimensional edges & 2-dimensional faces (polygon). So, these three: vertices, edges & faces form component of planar geometric graphs.

Example:



V = {a, b, c, d, e}

E = {(a, b), (b, c), (c, d), (d, e), (e, a), (b, d)}

F = {f0, f1, f2}

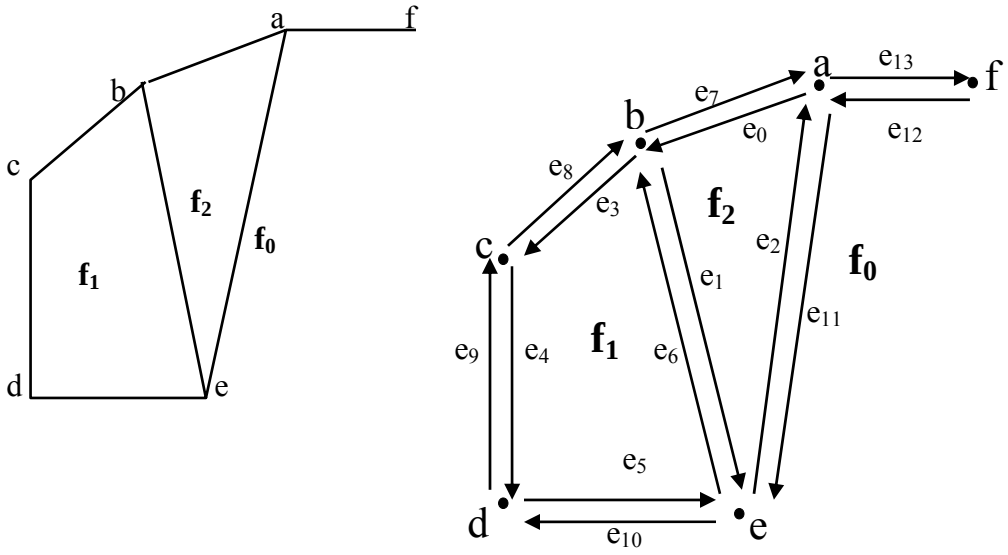
Data Structure for Planar Graphs: (for polytopes/convex polyhedral in 3D)

Doubly Connected Edge List:

DCEL is an efficient data structure for representing the planar graphs. The DCEL is a common edge-based representation. In this representation, each edge is viewed as a pair of oppositely directed halt edge.

This DCEL itself consist of three lists;

- ***Edge list:*** It consist of vertices (Source & target), faces (left & right) and edges (next & previous) & twin half edge.
- ***Face list:*** it consist of the adjacent edge for which the face is incident face.
- ***Vertex list:*** it consists of vertex coordinates & the incident edge.



Record for Half Edge:

- Each half edge will have entry for twin half edge, previous halt edge, next half edge, start vertex, incident face.

Edge Incident	Twin half edge	Prev	Next	Start Vertex	Incident face
e_0	e_7	e_2	e_1	a	f_2
e_1	e_6	e_0	e_2	b	f_2
e_2	e_{11}	e_1	e_0	e	f_2
e_3	e_8	e_6	e_4	b	f_1
e_4	e_9	e_3	e_5	c	f_1
...	…	…	…	…	…
...	…	…	…	…	…
...	…	…	…	…	…
e_{12}	e_{13}	e_{13}	e_{11}	f	f_0
e_{13}	e_{12}	e_7	e_{12}	a	f_0

Record for Vertex:

- Each vertex record will have its entry for x-coordinate, y-coordinate, incident edge, other information (if required).

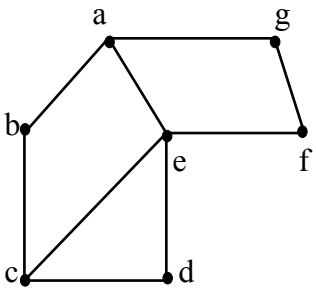
Index (Vertex)	X	Y	Incident edge (Any one)	Other Info.
a	a.x	a.y	e_2	…….
b	b.x	b.y	e_0	…….
..	..	..	..	…….
..	..	..	..	…….
f	f.x	f.y	e_{13}	…….

Record for Faces:

- Each face record will have entry for boundary edge to which face is incident & for other info (if required)

Index (face)	Boundary Edge (Any edge of the face)	Other Info
f0	e_{12}	……
f1	e_3	……..
f2	e_0	…….

H/W: Maintain DCEL for:



Euler's Formula for Polyhedra:

Let V, E, & F are the vertices, edges & faces of a polyhedra respectively, then they are related as;

$$V - E + F = 2$$

This relation is known as Euler's formula.

Theorem: For a given polyhedral with V = n vertices, E edges & F faces respectively, then V – E + F = 2 and E & F are each O(n).

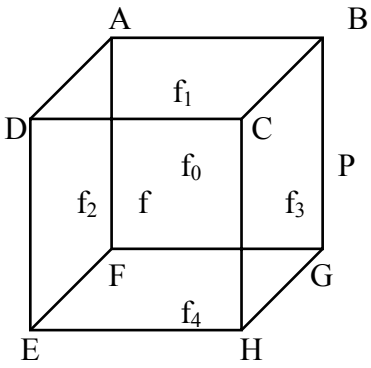
Proof:

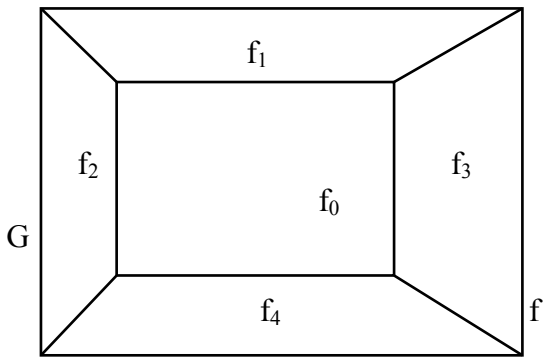
First part showing V – E + F = 2

Step 1: converting polyhedron surface into planar graph:

Let P be a polyhedron surface. First flatten the surface P into the planar graph G. For this;

- Choose any arbitrary face f of P & remove it leaving hole in the surface.
- Stretch the hole wider & wider until it becomes much larger & results into planar graph G.





- The nodes of planar graph G are the vertices of P & edges of G are edges of P.
- Each face of P, except f, becomes a bounded face of G & f becomes unbounded face of G.

Step 2:

Consider a restricted case where G is tree of V vertices & E edges.

By the definition of tree,

Clearly, V = E + 1

A tree has only one face which is the exterior face. So F = 1.

Now, Euler's formula for tree is;

V – E + F = V – E + 1 = (E +1) – E + 1 = 2 which is true.

Step 3:

Suppose Euler's formula is true for all connected graph with no more than E – 1 edge. Let G be a graph with V vertices, E edges & F faces respectively. If G is a tree we are done. So, suppose G is not tree, then it must have a cycle. Let e be an edge of G in the same cycle.

The graph G' = G – e is connected with V vertices, E – 1 edges & F – 1 faces, since removal of edge e joins two faces together.

So, by induction hypothesis;

V – (E – 1) + (F – 1) = 2

i.e. V – E + F = 2

Second part showing E & F are O (n):

Consider the faces of polyhedron are triangular, if not we can triangulate the faces without affecting the number of vertices. (So all faces are considered triangular)

If we count the edges face by face then each face has three edges so we get 3F edges. Since, an edge is shared by two faces; this doubles the count of edges. So, 3F = 2E.

Now substituting this value in Euler's formula;

V – E + F = 2

V – E + $\frac{2E}{3}$ = 2

V – 2 = E/3

Or, E = 3V – 6 < 3V = 3n

⇒ E = O (n)

V – E + F = 2

V – $\frac{3F}{2}$ + F = 2

V – 2 = $\frac{3F}{2}$ – F = $\frac{F}{2}$

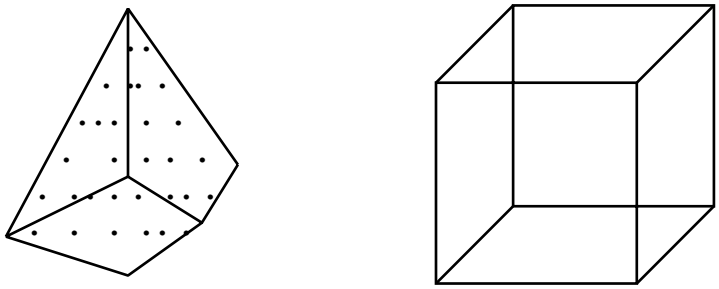
i.e. 2V – 4 = F

i.e. F = 2V – 4 = 2n – 4 < 2n

∴ F = O (n)

Convex Hull in 3D:

A convex hull of a set of points S in 3D is the smallest convex polyhedron containing all the points of S.



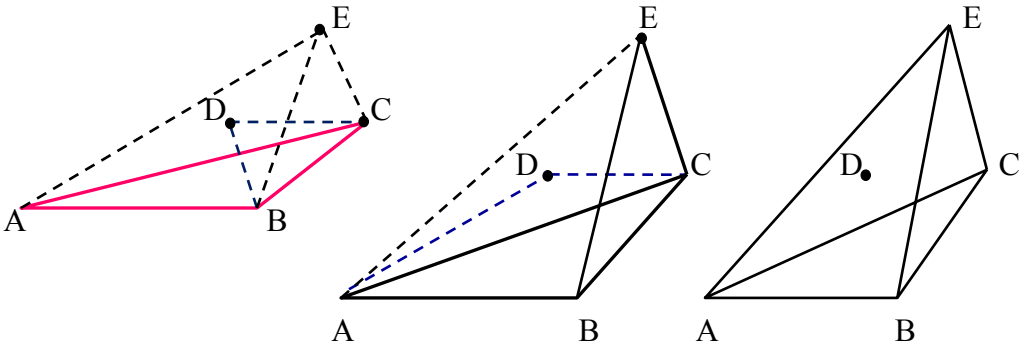
(1) Gift Wrap (Chand-Kapoor) Algorithm: -

Key Idea: Given one facet (a (d – 1) – face) of the convex hull, find neighboring facet of the hull by "wrapping" a (n-1)-dimensional affine set around the point set. Continue from each facet to its neighbors until all facets are found.

E.g.: In d = 3, imagine wrapping a sheet of 2-dimensional wrapping paper around a 3-dimensional gift box.

The algorithmic description is:

- Find a point with lowest y-coordinate, p_0, say as a starting point.
- Find other two point's p_1& p_2 with next lowest y-coordinate and construct a triangular face $p_0p_1p_2$ as a first face of convex polyhedron. So, at any step connected portion of the hull is constructed.
- Select a face, say F of this partial hull and an edge e whose second adjacent face remains to be found.
- Now bent a plane containing the face F over e toward the points in set until first point p on S is encountered. Then, the convex region formed by {p, e} is a new triangular face of the hull. The selection of point p can be characterized by minimum turning angle from the plane containing F.
- Continue wrapping in the same manner until no point remains to be processed.



In the Figure above; Point D is interior to the tetrahedron formed by points A, B, C & E. Therefore D is not on the hull. Here, we start with triangular face ABC & gift-wrap around BC. The point E & facet BCE is added to the hull. Next we gift-wrap around edge AC & add facet ACE to the hull.

Complexity Analysis:

- Clearly, complexity of this algorithm is $O(n^2)$, as there are O (n) faces that are to be wrapped. Wrapping each face takes O(n), hence complexity turns out to be $O(n^2)$.
- However if no. of faces of the hull is predefined, say F, a constant, then the time complexity will be O(Fn), which is considered to be linear.

(2) Divide & Conquer Approach:-

The approach of this algorithm is same as that in 2D- divide & conquer algorithm for hull construction i.e.

- At first sort the points by their x-coordinates & divide the set into two, approximately equal halves.
- Recursively construct the hull of each half & merge.

However, this approach is difficult to implement and is not used as frequently in practice as other asymptotically slower algorithms such as the "incremental".

Description of Algorithm:

Divide Step: Sort the points by x-coordinate. The first half of the sorted list will be in first half part and remaining points will be in second half. The division of points is continued until size is greater than 3.

Conquer (Computation of hull) Step: If the size of point set is 2, hull is an edge only and if the size is 3 then the hull is a triangular face.

Combine Step: Let A & B be the two hulls to be merged. The hull of A ∪ B will add a single band of faces with topology of cylinder without end caps. Each of the face uses at least one edge of either A or B.

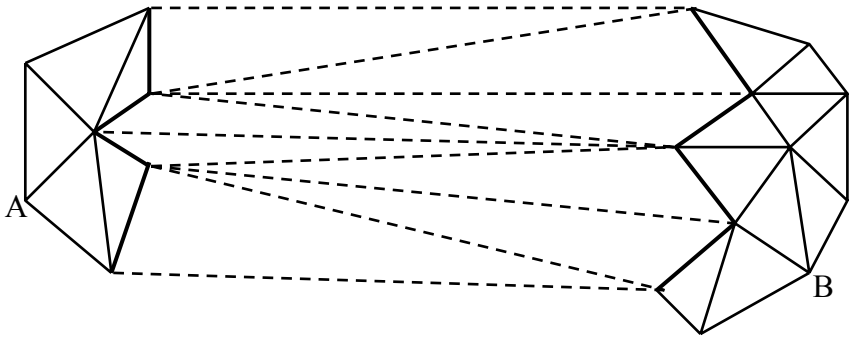


Fig: - Bold lines are edges of separate hulls & dotted lines are edges added during merging A & B. So, dotted line & Bold lines together constitute the triangular faces.

Here, we are merging two hulls A & B adding a cylinder of faces. For this we have following steps;

- If π is a plane that supports A & B form below at a point a & b respectively, then line ab is one of the edge of the new face to combine A & B.
- Rotating the plane π about line L = ab, new points of contacts can be obtained.

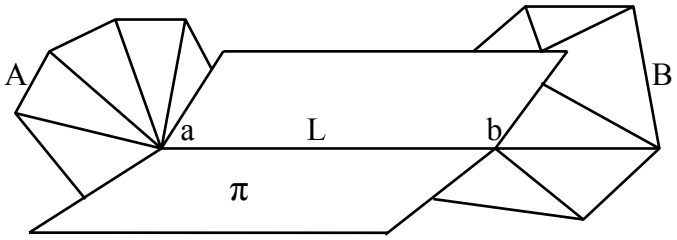


Fig: - Plane π bent toward polytopes A & B along line L.

- If first, new contact point in A is, say c, and then ac is edge of A to which a face connects B.
- To find a & b, research all neighbors of a & b to which plane π makes minimum angle is the next point hit by π first.
- Merging band of faces has a triangular faces (a, b, c) in which ac is an edge of hull A & b is a vertex of hull B.
- Repeat like this the wrapping closes upon itself.

Complexity Analysis:

Same as in case of 2D, we should be careful to merging step so that it can be completed in O (n). Hence algorithm recurrence relation will be;

$$T(n) = 2\ T(n/2) + O(n).$$

Solving this, it results O(nlogn). Hence complexity of the algorithm is O(nlogn).

(3) Incremental Approach:

The overall structure of 3D-incremental algorithm is identical to that of two dimensional versions. In this method, we have the following idea;

- Initially, take first three points p_0, p_1, p_2 & construct a hull which is
  H_2 = ConvexHull(p_0, p_1, p_2); a triangular face.
- At any i^{th} iteration, compute the hull H; as
  $H_i = H_{i-1} \cup p_i$

For computing H_i from H_{i-1}, for next additional point p_i, we may have two cases;

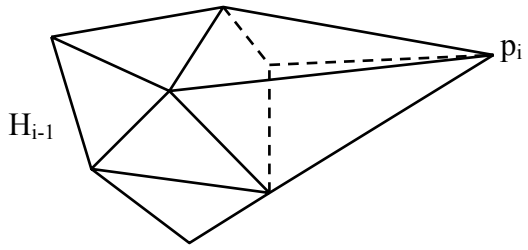
Case I: $p_i \in H_{i-1}$

For this case, discard p_i. Here, if p_i lies to positive side of every plane determined by a face of H_{i-1}. It can be done by left of triangle test by using volume of tetrahedron, just as left segment test is based on area of triangle. If all the faces are oriented consistently, the volumes must all have the same sign. (+ve in our assumption as for left turn test)

Case II: $p_i \notin H_{i-1}$

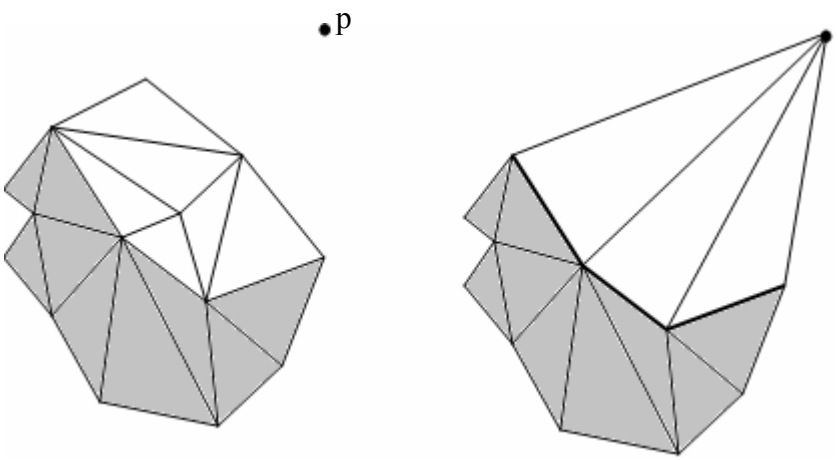
In this case, compute the cone tangent to H_{i-1} whose apex is pi and construct new hull with $H_{i-1} \cup p_i$. To determine whether $p_i \notin H_{i-1}$, we can check the turn test using volume of tetrahedron as above & if any one test is negative then $p_i \notin H_{i-1}$.

In 3D, there are tangent planes rather than tangent lines. These planes bound a cone of triangle faces each of whose apexes is p_i & whose base is an edge of H_{i-1}.



To draw the tangent cone, we have to find out the base of the cone, which is the faces of the hull H_{i-1}. So, only the visible faces from point p_i will be the discarded ones & their edges will become the bases of the cone faces apexed at p_i.

If we can determine which faces of H_{i-1} are visible to p_i, then will know enough to find the border edges & therefore construct the cone. To determine whether a triangular face is (a, b, c) is visible from p_i, we can check the signed volume of tetrahedron (a, b, c, p_i), p_i is visible if and only if volume is negative.



Complexity Analysis:

Clearly, there are O (n) faces & O (n) edges; so computing for each face, the loop will iterates for n^2 times. Clearly, the complexity will be O (n^2).