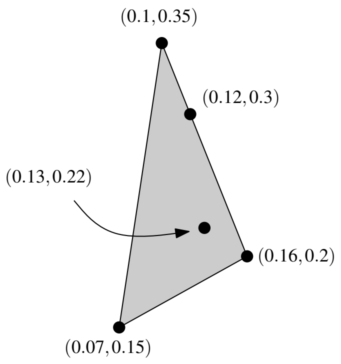
# 11 Convex Hulls

## Mixing Things

The output of oil wells is a mixture of several different components, and the proportions of these components vary between different sources. This can sometimes be exploited: by mixing together the output of different wells, one can produce a mixture with proportions that are particularly favorable for the refining process.

Let's look at an example. For simplicity we assume that we are only interested in two of the components—call them $A$ and $B$—of our product. Assume that we are given a mixture $\xi_1$ with 10% of component $A$ and 35% of component $B$, and another mixture $\xi_2$ with 16% of $A$ and 20% of $B$. Assume further that what we really need is a mixture that contains 12% of $A$ and 30% of $B$. Can we produce this mixture from the given ones? Yes, mixing $\xi_1$ and $\xi_2$ in the ratio $2:1$ gives the desired product. However, it is impossible to make a mixture of $\xi_1$ and $\xi_2$ that contains 13% of $A$ and 22% of $B$. But if we have a third mixture $\xi_3$ containing 7% of $A$ and 15% of $B$, then mixing $\xi_1$, $\xi_2$, and $\xi_3$ in the ratio of $1:3:1$ will give the desired result.



What has all this to do with geometry? This becomes clear when we represent the mixtures $\xi_1$, $\xi_2$, and $\xi_3$ by points in the plane, namely by $p_1 := (0.1, 0.35)$, $p_2 := (0.16, 0.2)$, and $p_3 := (0.07, 0.15)$. Mixing $\xi_1$ and $\xi_2$ in the ratio $2:1$ gives the mixture represented by the point $q := (2/3)p_1 + (1/3)p_2$. This is the point on the segment $\overline{p_1 p_2}$ such that $\text{dist}(p_2, q) : \text{dist}(q, p_1) = 2:1$, where $\text{dist}(.,.)$ denotes the distance between two points. More generally, by mixing $\xi_1$ and $\xi_2$ in varying ratios, we can produce the mixtures represented by any point on the line segment $\overline{p_1 p_2}$. If we start with the three base mixtures $\xi_1$, $\xi_2$, and $\xi_3$, we can produce any point in the triangle $p_1 p_2 p_3$. For instance, mixing $\xi_1$, $\xi_2$, and $\xi_3$ in the ratio $1:3:1$ gives the mixture represented by the point $(1/5)p_1 + (3/5)p_2 + (1/5)p_3 = (0.13, 0.22)$.

What happens if we don't have three but $n$ base mixtures, for some $n > 3$, represented by points $p_1, p_2, \ldots, p_n$? Suppose that we mix them in the ratio $l_1 : l_2 : \cdots : l_n$. Let $L := \sum_{j=1}^{n} l_j$ and let $\lambda_i := l_i / L$. Note that

$$\lambda_i \geqslant 0 \text{ for all } i \qquad \text{and} \qquad \sum_{i=1}^{n} \lambda_i = 1.$$

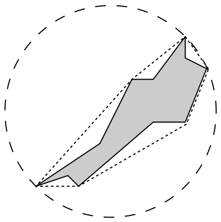
The mixture we get by mixing the base mixtures in the given ratio is the one represented by

$$\sum_{i=1}^{n} \lambda_i p_i.$$

Such a linear combination of the points $p_i$ where the $\lambda_i$ satisfy the conditions stated above—each $\lambda_i$ is non-negative, and the sum of the $\lambda_i$ is one—is called a *convex combination*. In Chapter 1 we defined the *convex hull* of a set of points as the smallest convex set containing the points or, more precisely, as the intersection of all convex sets containing the points. One can show that the convex hull of a set of points is exactly the set of all possible convex combinations of the points. We can therefore test whether a mixture can be obtained from the base mixtures by computing the convex hull of their representative points, and checking whether the point representing the mixture lies inside it.

What if there are more than two interesting components in the mixtures? Well, what we have said above remains true; we just have to move to a space of higher dimension. More precisely, if we want to take $d$ components into account we have to represent a mixture by a point in $d$-dimensional space. The convex hull of the points representing the base mixtures, which is a convex polytope, represents the set of all possible mixtures.

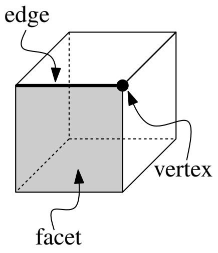
Convex hulls—in particular convex hulls in 3-dimensional space—are used in various applications. For instance, they are used to speed up collision detection in computer animation. Suppose that we want to check whether two objects $\mathcal{P}_1$ and $\mathcal{P}_2$ intersect. If the answer to this question is negative most of the time, then the following strategy pays off. Approximate the objects by simpler objects $\widehat{\mathcal{P}_1}$ and $\widehat{\mathcal{P}_2}$ that contain the originals. If we want to check whether $\mathcal{P}_1$ and $\mathcal{P}_2$ intersect, we first check whether $\widehat{\mathcal{P}_1}$ and $\widehat{\mathcal{P}_2}$ intersect; only if this is the case do we need to perform the—supposedly more costly—test on the original objects.



There is a trade-off in the choice of the approximating objects. On the one hand, we want them to be simple so that intersection tests are cheap. On the other hand, simple approximations most likely do not approximate the original objects very well, so there is a bigger chance we have to test the originals. Bounding spheres are on one side of the spectrum: intersection tests for spheres are quite simple, but for many objects spheres do not provide a good approximation. Convex hulls are more on the other side of the spectrum: intersection tests for convex hulls are more complicated than for spheres—but still simpler than for non-convex objects—but convex hulls can approximate most objects a lot better.

## 11.1 The Complexity of Convex Hulls in 3-Space

In Chapter 1 we have seen that the convex hull of a set $P$ of $n$ points in the plane is a convex polygon whose vertices are points in $P$. Hence, the convex hull has at most $n$ vertices. In 3-dimensional space a similar statement is true: the convex hull of a set $P$ of $n$ points is a convex polytope whose vertices are points in $P$ and, hence, it has at most $n$ vertices. In the planar case the bound on the number of vertices immediately implies that the complexity of the convex hull is linear, since the number of edges of a planar polygon is equal to the number of vertices. In 3-space this is no longer true; the number of edges of a polytope can be higher than the number of vertices. But fortunately the difference cannot be too large, as follows from the following theorem on the number of edges and facets of convex polytopes. (Formally, a *facet* of a convex polytope is defined to be a maximal subset of coplanar points on its boundary. A facet of a convex polytope is necessarily a convex polygon. An *edge* of a convex polytope is an edge of one of its facets.)



**Theorem 11.1** *Let $\mathcal{P}$ be a convex polytope with $n$ vertices. The number of edges of $\mathcal{P}$ is at most $3n-6$, and the number of facets of $\mathcal{P}$ is at most $2n-4$.*

*Proof.* Recall that Euler's formula states for a connected planar graph with $n$ nodes, $n_e$ arcs, and $n_f$ faces the following relation holds:

$$n - n_e + n_f = 2.$$

Since we can interpret the boundary of a convex polytope as a planar graph—see Figure 11.1—the same relation holds for the numbers of vertices, edges, and facets in a convex polytope. (In fact, Euler's formula was originally stated in

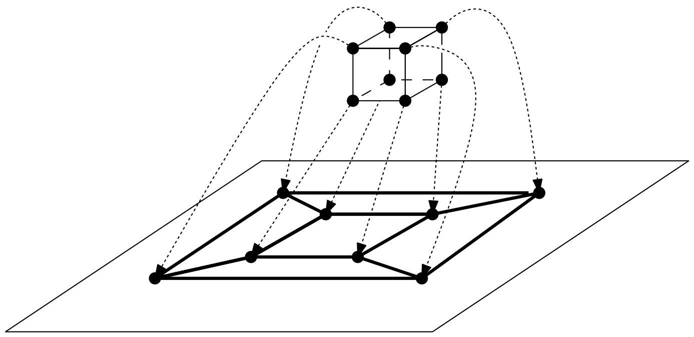


*Figure 11.1*
A cube interpreted as a planar graph: note that one facet maps to the unbounded face of the graph

terms of polytopes, not in terms of planar graphs.) Every face of the graph corresponding to $\mathcal{P}$ has at least three arcs, and every arc is incident to two faces, so we have $2n_e \geqslant 3n_f$. Plugging this into Euler's formula we get

$$n + n_f - 2 \geqslant 3n_f/2,$$

so $n_f \leqslant 2n-4$. Applying Euler's formula once more, we see that $n_e \leqslant 3n-6$. For the special case that every facet is a triangle—the case of a *simplicial polytope*—the bounds on the number of edges and facets of an $n$-vertex polytope are exact, because then $2n_e = 3n_f$. ◻

Theorem 11.1 also holds for non-convex polytopes whose so-called *genus* is zero, that is, polytopes without holes or tunnels; for polytopes of larger genus similar bounds hold. Since this chapter deals with convex hulls, however, we refrain from defining what a (non-convex) polytope exactly is, which we would need to do to prove the theorem in the non-convex case.

If we combine Theorem 11.1 with the earlier observation that the convex hull of a set of points in 3-space is a convex polytope whose vertices are points in $P$, we get the following result.

**Corollary 11.2** *The complexity of the convex hull of a set of $n$ points in three-dimensional space is $O(n)$.*

## 11.2 Computing Convex Hulls in 3-Space

Let $P$ be a set of $n$ points in 3-space. We will compute $\mathcal{CH}(P)$, the convex hull of $P$, using a randomized incremental algorithm, following the paradigm we have met before in Chapters 4, 6, and 9.

The incremental construction starts by choosing four points in $P$ that do not lie in a common plane, so that their convex hull is a tetrahedron. This can be done as follows. Let $p_1$ and $p_2$ be two points in $P$. We walk through the set $P$ until we find a point $p_3$ that does not lie on the line through $p_1$ and $p_2$. We continue searching $P$ until we find a point $p_4$ that does not lie in the plane through $p_1$, $p_2$, and $p_3$. (If we cannot find four such points, then all points in $P$ lie in a plane. In this case we can use the planar convex hull algorithm of Chapter 1 to compute the convex hull.)

Next we compute a random permutation $p_5, \dots, p_n$ of the remaining points. We will consider the points one by one in this random order, maintaining the convex hull as we go. For an integer $r \geqslant 1$, let $P_r := \{p_1, \dots, p_r\}$. In a generic step of the algorithm, we have to add the point $p_r$ to the convex hull of $P_{r-1}$, that is, we have to transform $\mathcal{CH}(P_{r-1})$ into $\mathcal{CH}(P_r)$. There are two cases.

- If $p_r$ lies inside $\mathcal{CH}(P_{r-1})$, or on its boundary, then $\mathcal{CH}(P_r) = \mathcal{CH}(P_{r-1})$, and there is nothing to be done.

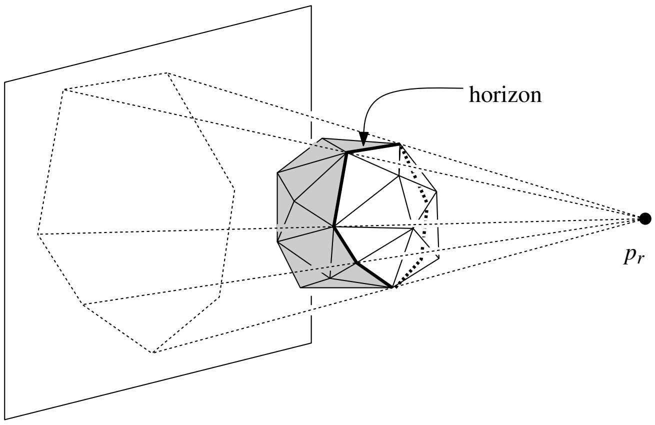
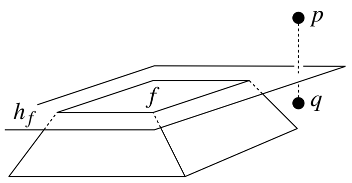


*Figure 11.2*
The horizon of a polytope

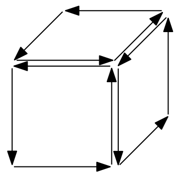
- Now suppose that $p_r$ lies outside $\mathcal{CH}(P_{r-1})$. Imagine that you are standing at $p_r$, and that you are looking at $\mathcal{CH}(P_{r-1})$. You will be able to see some facets of $\mathcal{CH}(P_{r-1})$—the ones on the front side—but others will be invisible because they are on the back side. The visible facets form a connected region on the surface of $\mathcal{CH}(P_{r-1})$, called the *visible region* of $p_r$ on $\mathcal{CH}(P_{r-1})$, which is enclosed by a closed curve consisting of edges of $\mathcal{CH}(P_{r-1})$. We call this curve the *horizon* of $p_r$ on $\mathcal{CH}(P_{r-1})$. As you can see in Figure 11.2, the projection of the horizon is the boundary of the convex polygon obtained by projecting $\mathcal{CH}(P_{r-1})$ onto a plane, with $p_r$ as the center of projection. What exactly does "visible" mean geometrically? Consider the plane $h_f$ containing a facet $f$ of $\mathcal{CH}(P_{r-1})$. By convexity, $\mathcal{CH}(P_{r-1})$ is completely contained in one of the closed half-spaces defined by $h_f$. The face $f$ is visible from a point if that point lies in the open half-space on the other side of $h_f$.



$f$ is visible from $p$, but not from $q$

The horizon of $p_r$ plays a crucial role when we want to transform $\mathcal{CH}(P_{r-1})$ to $\mathcal{CH}(P_r)$: it forms the border between the part of the boundary that can be kept—the invisible facets—and the part of the boundary that must be replaced—the visible facets. The visible facets must be replaced by facets connecting $p_r$ to its horizon.

Before we go into more details, we should decide how we are going to represent the convex hull of points in space. As we observed before, the boundary of a 3-dimensional convex polytope can be interpreted as a planar graph. Therefore we store the convex hull in the form of a doubly-connected edge list, a data structure developed in Chapter 2 for storing planar subdivisions. The only difference is that vertices will now be 3-dimensional points. We will keep the convention that the half-edges are directed such that the ones bounding any face form a counterclockwise cycle *when seen from the outside* of the polytope.



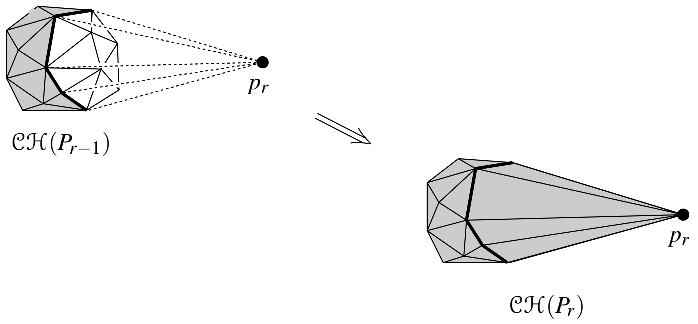
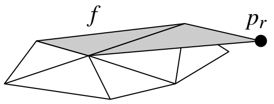


*Figure 11.3*
Adding a point to the convex hull

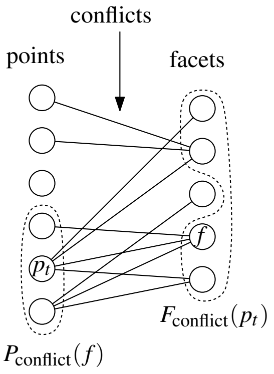
Back to the addition of $p_r$ to the convex hull. We have a doubly-connected edge list representing $\mathcal{CH}(P_{r-1})$, which we have to transform into a doubly-connected edge list for $\mathcal{CH}(P_r)$. Suppose that we knew all facets of $\mathcal{CH}(P_{r-1})$ visible from $p_r$. Then it would be easy to remove all the information stored for these facets from the doubly-connected edge list, compute the new facets connecting $p_r$ to the horizon, and store the information for the new facets in the doubly-connected edge list. All this will take linear time in the total complexity of the facets that disappear.

There is one subtlety we should take care of after the addition of the new facets: we have to check whether we have created any coplanar facets. This happens if $p_r$ lies in the plane of a face of $\mathcal{CH}(P_{r-1})$. Such a face $f$ is not visible from $p_r$ by our definition of visibility above. Hence, $f$ will remain unchanged, and we will add triangles connecting $p_r$ to the edges of $f$ that are part of the horizon. Those triangles are coplanar with $f$, and so they have to be merged with $f$ into one facet.



In the discussion so far we have ignored the problem of finding the facets of $\mathcal{CH}(P_{r-1})$ that are visible from $p_r$. Of course this could be done by testing every facet. Since such a test takes constant time—we have to check to which side of a given plane the point $p_r$ lies—we can find all visible facets in $O(r)$ time. This would lead to an $O(n^2)$ algorithm. Next we show how to do better.

The trick is that we are going to work ahead: besides the convex hull of the current point set we shall maintain some additional information, which will make it easy to find the visible facets. In particular, we maintain for each facet $f$ of the current convex hull $\mathcal{CH}(P_r)$ a set $P_{\text{conflict}}(f) \subseteq \{p_{r+1}, p_{r+2}, \ldots, p_n\}$ containing the points that can see $f$. Conversely, we store for every point $p_t$, with $t > r$, the set $F_{\text{conflict}}(p_t)$ of facets of $\mathcal{CH}(P_r)$ visible from $p_t$. We will say that a point $p \in P_{\text{conflict}}(f)$ is *in conflict with* the facet $f$, because $p$ and $f$ cannot peacefully live together in the convex hull—once we add a point $p \in P_{\text{conflict}}(f)$ to the convex hull, the facet $f$ must go. We call $P_{\text{conflict}}(f)$ and $F_{\text{conflict}}(p_t)$ *conflict lists*.

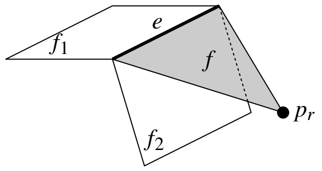


We maintain the conflicts in a so-called *conflict graph*, which we denote by $\mathcal{G}$. The conflict graph is a bipartite graph. It has one node set with a node for every point of $P$ that has not been inserted yet, and one node set with a node for every facet of the current convex hull. There is an arc for every conflict between a point and a facet. In other words, there is an arc between a point $p_t \in P$ and facet $f$ of $\mathcal{CH}(P_r)$ if $r < t$ and $f$ is visible from $p_t$. Using the conflict graph $\mathcal{G}$, we can report the set $F_{\text{conflict}}(p_t)$ for a given point $p_t$ (or $P_{\text{conflict}}(f)$ for a given facet $f$) in time linear in its size. This means that when we insert $p_r$ into $\mathcal{CH}(P_{r-1})$, all we have to do is to look up $F_{\text{conflict}}(p_r)$ in $\mathcal{G}$ to get the visible facets, which we can then replace by the new convex hull facets connecting $p_r$ to the horizon.

Initializing the conflict graph $\mathcal{G}$ for $\mathcal{CH}(P_4)$ can be done in linear time: we simply walk through the list of points $P$ and determine which of the four faces of $\mathcal{CH}(P_4)$ they can see.

To update $\mathcal{G}$ after adding a point $p_r$, we first discard the nodes and incident arcs for all the facets of $\mathcal{CH}(P_{r-1})$ that disappear from the convex hull. These are the facets visible from $p_r$, which are exactly the neighbors of $p_r$ in $\mathcal{G}$, so this is easy. We also discard the node for $p_r$. We then add nodes to $\mathcal{G}$ for the new facets we created, which connect $p_r$ to the horizon. The essential step is to find the conflict lists of these new facets. No other conflicts have to be updated: the conflict set $P_{\text{conflict}}(f)$ of a facet $f$ that is unaffected by the insertion of $p_r$ remains unchanged.

The facets created by the insertion of $p_r$ are all triangles, except for those that have been merged with existing coplanar facets. The conflict list of a facet of the latter type is trivial to find: it is the same as the conflict list of the existing facet, since the merging does not change the plane containing the facet. So let's look at one of the new triangles $f$ incident to $p_r$ in $\mathcal{CH}(P_r)$. Suppose that a point $p_t$ can see $f$. Then $p_t$ can certainly see the edge $e$ of $f$ that is opposite $p_r$. This edge $e$ is a horizon edge of $p_r$, and it was already present in $\mathcal{CH}(P_{r-1})$. Since $\mathcal{CH}(P_{r-1}) \subset \mathcal{CH}(P_r)$, the edge $e$ must have been visible from $p_t$ in $\mathcal{CH}(P_{r-1})$ as well. That can only be the case if one of the two facets incident to $e$ in $\mathcal{CH}(P_{r-1})$ is visible from $p_t$. This implies that the conflict list of $f$ can be found by testing the points in the conflict lists of the two facets $f_1$ and $f_2$ that were incident to the horizon edge $e$ in $\mathcal{CH}(P_{r-1})$.



We stated earlier that we store the convex hull as a doubly-connected edge list, so changing the convex hull means changing the information in the doubly-connected edge list. To keep the code short, however, we have omitted all explicit references to the doubly-connected edge list in the pseudocode below, which summarizes the convex hull algorithm.

```
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 $p_5, p_6, \dots, p_n$ 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 corre-
                  sponding to the facets in $F_{\text{conflict}}(p_r)$ from $\mathcal{G}$, together with
                  their incident arcs.
21. return $\mathcal{C}$
```

## 11.3* The Analysis

As usual when we analyse a randomized incremental algorithm, we first try to bound the expected structural change. For the convex hull algorithm this means we want to bound the total number of facets created by the algorithm.

**Lemma 11.3** *The expected number of facets created by* CONVEXHULL *is at most* $6n-20$.

*Proof.* The algorithm starts with a tetrahedron, which has four facets. In every stage $r$ of the algorithm where $p_r$ lies outside $\mathcal{CH}(P_{r-1})$, new triangular facets connecting $p_r$ to its horizon on $\mathcal{CH}(P_{r-1})$ are created. What is the expected number of new facets? As in previous occasions where we analyzed randomized algorithms, we use backwards analysis. We look at $\mathcal{CH}(P_r)$ and imagine removing vertex $p_r$; the number of facets that disappear due to the removal of $p_r$ from $\mathcal{CH}(P_r)$ is the same as the number of facets that were created due to the insertion of $p_r$ into $\mathcal{CH}(P_{r-1})$. The disappearing facets are exactly the ones incident to $p_r$, and their number equals the number of edges incident to $p_r$ in $\mathcal{CH}(P_r)$. We call this number the degree of $p_r$ in $\mathcal{CH}(P_r)$, and we denote it by $\deg(p_r, \mathcal{CH}(P_r))$. We now want to bound the expected value of $\deg(p_r, \mathcal{CH}(P_r))$.

By Theorem 11.1 a convex polytope with $r$ vertices has at most $3r-6$ edges. This means that the sum of the degrees of the vertices of $\mathcal{CH}(P_r)$, which is a convex polytope with $r$ or less vertices, is at most $6r-12$. Hence, the average degree is bounded by $6-12/r$. Since we treat the vertices in random order, it seems that the expected degree of $p_r$ is bounded by $6-12/r$. We have to be a little bit careful, though: the first four points are already fixed when we generate the random permutation, so $p_r$ is a random element of $\{p_5, \dots, p_r\}$, not of $P_r$. Because $p_1, \dots, p_4$ have total degree at least 12, the expected value of $\deg(p_r, \mathcal{CH}(P_r))$ is bounded as follows:

$$\begin{aligned}
\mathrm{E}[\deg(p_r, \mathcal{CH}(P_r))] &= \frac{1}{r-4}\sum_{i=5}^{r}\deg(p_i, \mathcal{CH}(P_r)) \\
&\leqslant \frac{1}{r-4}\left(\left\{\sum_{i=1}^{r}\deg(p_i, \mathcal{CH}(P_r))\right\} - 12\right) \\
&\leqslant \frac{6r-12-12}{r-4} = 6.
\end{aligned}$$

The expected number of facets created by CONVEXHULL is the number of facets we start with (four) plus the expected total number of facets created during the additions of $p_5, \dots, p_n$ to the hull. Hence, the expected number of created facets is

$$4 + \sum_{r=5}^{n}\mathrm{E}[\deg(p_r, \mathcal{CH}(P_r))] \leqslant 4 + 6(n-4) = 6n-20. \qquad \square$$

Now that we have bounded the total amount of structural change we can bound the expected running time of the algorithm.

**Lemma 11.4** *Algorithm* CONVEXHULL *computes the convex hull of a set $P$ of $n$ points in $\mathbb{R}^3$ in $O(n \log n)$ expected time, where the expectation is with respect to the random permutation used by the algorithm.*

*Proof.* The steps before the main loop can certainly be done in $O(n \log n)$ time. Stage $r$ of the algorithm takes constant time if $F_{\text{conflict}}(p_r)$ is empty, which is when $p_r$ lies inside, or on the boundary of, the current convex hull.

If that is not the case, most of stage $r$ takes $O(\text{card}(F_{\text{conflict}}(p_r)))$ time, where card() denotes the cardinality of a set. The exceptions to this are the lines 17–19 and line 20. We shall bound the time spent in these lines later; first, we bound $\text{card}(F_{\text{conflict}}(p_r))$. Note that $\text{card}(F_{\text{conflict}}(p_r))$ is the number of facets deleted due to the addition of the point $p_r$. Clearly, a facet can only be deleted if it has been created before, and it is deleted at most once. Since the expected number of facets created by the algorithm is $O(n)$ by Lemma 11.3, this implies that the total number of deletions is $O(n)$ as well, so

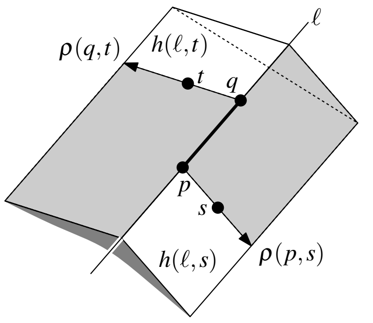
$$\mathrm{E}[\sum_{r=5}^{n} \text{card}(F_{\text{conflict}}(p_r))] = O(n).$$

Now for lines 17–19 and line 20. Line 20 takes time linear in the number of nodes and arcs that are deleted from $\mathcal{G}$. Again, a node or arc is deleted at most once, and we can charge the cost of this deletion to the stage where we created it. It remains to look at lines 17–19. In stage $r$, these lines are executed for all horizon edges, that is, all edges in $\mathcal{L}$. For one edge $e \in \mathcal{L}$, they take $O(\text{card}(P(e)))$ time. Hence, the total time spent in these lines in stage $r$ is $O(\sum_{e \in \mathcal{L}} \text{card}(P(e)))$. To bound the total expected running time, we therefore have to bound the expected value of

$$\sum_{e} \text{card}(P(e)),$$

where the summation is over all horizon edges that appear at any stage of the algorithm. We will prove below that this is $O(n \log n)$, which implies that the total running time is $O(n \log n)$. $\square$

We use the framework of *configuration spaces* from Chapter 9 to supply the missing bound. The universe $X$ is the set of $P$, and the configurations $\Delta$ correspond to convex hull edges. However, for technical reasons—in particular, to be able to deal correctly with degenerate cases—we attach a half-edge to both sides of the edge. To be more precise, a *flap* $\Delta$ is defined as an ordered four-tuple of points $(p,q,s,t)$ that do not all lie in a plane. The defining set $D(\Delta)$ is simply the set $\{p,q,s,t\}$. The killing set $K(\Delta)$ is more difficult to visualize. Denote the line through $p$ and $q$ by $\ell$. Given a point $x$, let $h(\ell,x)$ denote the half-plane bounded by $\ell$ that contains $x$. Given two points $x$, $y$, let $\rho(x,y)$ be the half-line starting in $x$ and passing through $y$. A point $x \in X$ is in $K(\Delta)$ if and only if it lies in one of the following regions:



- outside the closed convex 3-dimensional wedge defined by $h(\ell,s)$ and $h(\ell,t)$,
- inside $h(\ell,s)$ but outside the closed 2-dimensional wedge defined by $\rho(p,q)$ and $\rho(p,s)$,

- inside $h(\ell,t)$ but outside the closed 2-dimensional wedge defined by $\rho(q,t)$ and $\rho(q,p)$,
- inside the line $\ell$ but outside the segment $\overline{pq}$,
- inside the half-line $\rho(p,s)$ but outside the segment $\overline{ps}$,
- inside the half-line $\rho(q,t)$ but outside the segment $\overline{qt}$.

For every subset $S \subseteq P$, we define the set $\mathcal{T}(S)$ of active configurations—this is what we want to compute—as prescribed in Chapter 9: $\Delta \in \mathcal{T}(S)$ if and only if $D(\Delta) \subseteq S$ and $K(\Delta) \cap S = \emptyset$.

**Lemma 11.5** *A flap $\Delta = (p,q,s,t)$ is in $\mathcal{T}(S)$ if and only if $\overline{pq}$, $\overline{ps}$, and $\overline{qt}$ are edges of the convex hull $\mathcal{CH}(S)$, there is a facet $f_1$ incident to $\overline{pq}$ and $\overline{ps}$, and a different facet $f_2$ incident to $\overline{pq}$ and $\overline{qt}$. Furthermore, if one of the facets $f_1$ or $f_2$ is visible from a point $x \in P$ then $x \in K(\Delta)$.*

We leave the proof—which involves looking precisely at the cases when points are collinear or coplanar, but which is otherwise not difficult—to the reader.

As you may have guessed, the flaps take over the role of the horizon edges.

**Lemma 11.6** *The expected value of $\sum_e \operatorname{card}(P(e))$, where the summation is over all horizon edges that appear at some stage of the algorithm, is $O(n \log n)$.*

*Proof.* Consider an edge $e$ of the horizon of $p_r$ on $\mathcal{CH}(P_{r-1})$. Let $\Delta = (p,q,s,t)$ be one of the two flaps with $\overline{pq} = e$. By Lemma 11.5, $\Delta \in \mathcal{T}(P_{r-1})$, and the points in $P \setminus P_r$ that can see one of the facets incident to $e$ are all in $K(\Delta)$, so $P(e) \subseteq K(\Delta)$. By Theorem 9.15, it follows that the expected value of

$$\sum_{\Delta} \operatorname{card}(K(\Delta)),$$

where the summation is over all flaps $\Delta$ appearing in at least one $\mathcal{T}(P_r)$, is bounded by

$$\sum_{r=1}^{n} 16\left(\frac{n-r}{r}\right)\left(\frac{\mathrm{E}\big[\operatorname{card}(\mathcal{T}(P_r))\big]}{r}\right).$$

The cardinality of $\mathcal{T}(P_r)$ is twice the number of edges of $\mathcal{CH}(P_r)$. Therefore it is at most $6r-12$, so we get the bound

$$\sum_{e} \operatorname{card}(P(e)) \leqslant \sum_{\Delta} \operatorname{card}(K(\Delta)) \leqslant \sum_{r=1}^{n} 16\left(\frac{n-r}{r}\right)\left(\frac{6r-12}{r}\right) \leqslant 96 n \ln n. \quad \square$$

This finishes the last piece of the analysis of the convex hull algorithm. We get the following result:

**Theorem 11.7** *The convex hull of a set of $n$ points in $\mathbb{R}^3$ can be computed in $O(n \log n)$ randomized expected time.*

## 11.4* Convex Hulls and Half-Space Intersection

In Chapter 8 we have met the concept of duality. The strenth of duality lies in that it allows us to look at a problem from a new perspective, which can lead to more insight in what is really going on. Recall that we denote the line that is the dual of a point $p$ by $p^*$, and the point that is the dual of a line $\ell$ by $\ell^*$. The duality transform is incidence and order preserving: $p \in \ell$ if and only if $\ell^* \in p^*$, and $p$ lies above $\ell$ if and only if $\ell^*$ lies above $p^*$.

Let's have a closer look at what convex hulls correspond to in dual space. We will do this for the planar case. Let $P$ be a set of points in the plane. For technical reasons we focus on its *upper convex hull*, denoted $\mathcal{UH}(P)$, which consists of the convex hull edges that have $P$ *below* their supporting line—see the left side of Figure 11.4. The upper convex hull is a polygonal chain that connects the leftmost point in $P$ to the rightmost one. (We assume for simplicity that no two points have the same $x$-coordinate.)

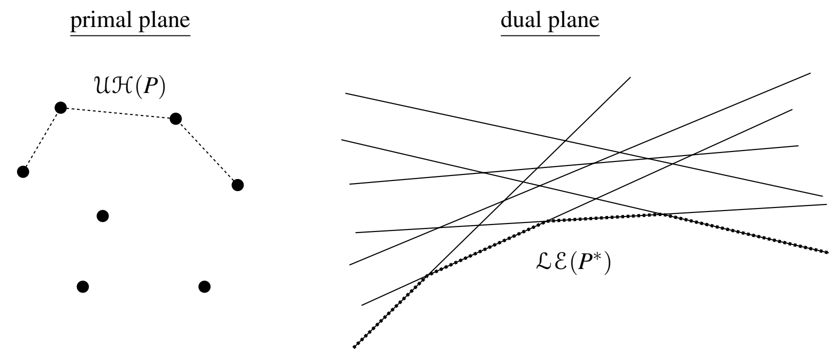


*Figure 11.4*
Upper hulls correspond to lower envelopes

When does a point $p \in P$ appear as a vertex of the upper convex hull? That is the case if and only if there is a non-vertical line $\ell$ through $p$ such that all other points of $P$ lie below $\ell$. In the dual plane this statement translates to the following condition: there is a point $\ell^*$ on the line $p^* \in P^*$ such that $\ell^*$ lies below all other lines of $P^*$. If we look at the arrangement $\mathcal{A}(P^*)$, this means that $p^*$ contributes an edge to the unique bottom cell of the arrangement. This cell is the intersection of the half-planes bounded by a line in $P^*$ and lying below that line. The boundary of the bottom cell is an $x$-monotone chain. We can define this chain as the minimum of the linear functions whose graphs are the lines in $P^*$. For this reason, the boundary of the bottom cell in an arrangement is often called the *lower envelope* of the set of lines. We denote the lower envelope of $P^*$ by $\mathcal{LE}(P^*)$—see the right hand side of Figure 11.4.

The points in $P$ that appear on $\mathcal{UH}(P)$ do so in order of increasing $x$-coordinate. The lines of $P^*$ appear on the boundary of the bottom cell in order of decreasing slope. Since the slope of the line $p^*$ is equal to the $x$-coordinate of $p$, it follows that the left-to-right list of points on $\mathcal{UH}(P)$ corresponds exactly to the right-to-left list of edges of $\mathcal{LE}(P^*)$. So the upper convex hull of a set of points is essentially the same as the lower envelope of a set of lines.

Let's do one final check. Two points $p$ and $q$ in $P$ form an upper convex hull edge if and only if all other points in $P$ lie below the line $\ell$ through $p$ and $q$. In the dual plane, this means that all lines $r^*$, with $r \in P \setminus \{p,q\}$, lie above the intersection point $\ell^*$ of $p^*$ and $q^*$. This is exactly the condition under which $p^* \cap q^*$ is a vertex of $\mathcal{LE}(P^*)$.

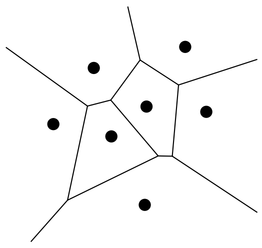
What about the *lower convex hull* of $P$ and the *upper envelope* of $P^*$? (We leave the precise definitions to the reader.) By symmetry, these concepts are dual to each other as well.

We now know that the intersection of *lower half-planes*—half-planes bounded from above by a non-vertical line—can be computed by computing an upper convex hull, and that the intersection of *upper half-planes* can be computed by computing a lower convex hull. But what if we want to compute the intersection of an arbitrary set $H$ of half-planes? Of course, we can split the set $H$ into a set $H_+$ of upper half-planes and a set $H_-$ of lower half-planes, compute $\bigcup H_+$ by computing the lower convex hull of ${H_+}^*$ and $\bigcup H_-$ by computing the upper convex hull of ${H_-}^*$, and then compute $\bigcap H$ by intersecting $\bigcup H_+$ and $\bigcup H_-$.

But is this really necessary? If lower envelopes correspond to upper convex hulls, and upper envelopes correspond to lower convex hulls, shouldn't then the intersection of arbitrary half-planes correspond to full convex hulls? In a sense, this is true. The problem is that our duality transformation cannot handle vertical lines, and lines that are close to vertical but have opposite slope are mapped to very different points. This explains why the dual of the convex hull consists of two parts that lie rather far apart.

It is possible to define a different duality transformation that allows vertical lines. However, to apply this duality to a given set of half-planes, we need a point in the intersection of the half-planes. But that was to be expected. As long as we do not want to leave the Euclidean plane, there cannot be any general duality that turns the intersection of a set of half-planes into a convex hull, because the intersection of half-planes can have one special property: it can be empty. What could that possibly correspond to in the dual? The convex hull of a set of points in Euclidean space is always well defined: there is no such thing as "emptiness." (This problem is nicely solved if one works in oriented projective space, but this concept is beyond the scope of this book.) Only once you know that the intersection is not empty, and a point in the interior is known, can you define a duality that relates the intersection with a convex hull.

We leave it at this for now. The important thing is that—although there are technical complications—convex hulls and intersections of half-planes (or half-spaces in three dimensions) are essentially dual concepts. Hence, an algorithm to compute the intersection of half-planes in the plane (or half-spaces in three dimensions) can be given by dualizing a convex-hull algorithm.



## 11.5* Voronoi Diagrams Revisited

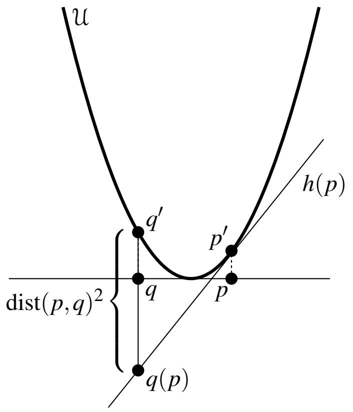
In Chapter 7 we introduced the Voronoi diagram of a set of points in the plane. It may come as a surprise that there is a close relationship between planar Voronoi diagrams and the intersection of upper half-spaces in 3-dimensional space. By the result on duality of the previous section, this implies a close relation between planar Voronoi diagrams and lower convex hulls in 3-space.

This has to do with an amazing property of the unit paraboloid in 3-space. Let $\mathcal{U} := (z = x^2 + y^2)$ denote the unit paraboloid, and let $p := (p_x, p_y, 0)$ be a point in the plane $z = 0$. Consider the vertical line through $p$. It intersects $\mathcal{U}$ in the point $p' := (p_x, p_y, p_x^2 + p_y^2)$. Let $h(p)$ be the non-vertical plane $z = 2p_x x + 2p_y y - (p_x^2 + p_y^2)$. Notice that $h(p)$ contains the point $p'$. Now consider any other point $q := (q_x, q_y, 0)$ in the plane $z = 0$. The vertical line through $q$ intersects $\mathcal{U}$ in the point $q' := (q_x, q_y, q_x^2 + q_y^2)$, and it 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

$$q_x^2 + q_y^2 - 2p_x q_x - 2p_y q_y + p_x^2 + p_y^2 = (q_x - p_x)^2 + (q_y - p_y)^2 = \text{dist}(p,q)^2.$$



Hence, the plane $h(p)$ encodes—together with the unit paraboloid—the distance between $p$ and any other point in the plane $z = 0$. (Since $\text{dist}(p,q)^2 \geqslant 0$ for any point $q$, and $p' \in h(p)$, this also implies that $h(p)$ is the tangent plane to $\mathcal{U}$ at $p'$.)

The fact that the plane $h(p)$ encodes the distance of other points to $p$ leads to a correspondence between Voronoi diagrams and upper envelopes, as explained next. Let $P$ be a planar point set, which we imagine to lie in the plane $z = 0$ of 3-dimensional space. Consider the set $H := \{h(p) \mid p \in P\}$ of planes, and let $\mathcal{UE}(H)$ be the upper envelope of the planes in $H$. We claim that the projection of $\mathcal{UE}(H)$ on the plane $z = 0$ is the Voronoi diagram of $P$. Figure 11.5 illustrates this one dimension lower: the Voronoi diagram of the points $p_i$ on the line $y = 0$ is the projection of the upper envelope of the lines $h(p_i)$.

**Theorem 11.8** *Let $P$ be a set of points in 3-dimensional space, all lying in the plane $z = 0$. Let $H$ be the set of planes $h(p)$, for $p \in P$, defined as above. Then the projection of $\mathcal{UE}(H)$ on the plane $z = 0$ is the Voronoi diagram of $P$.*

*Proof.* To prove the theorem, we will show that the Voronoi cell of a point $p \in P$ is exactly the projection of the facet of $\mathcal{UE}(H)$ that lies on the plane $h(p)$. Let $q$ be a point in the plane $z = 0$ lying in the Voronoi cell of $p$. Hence, we have $\text{dist}(q,p) < \text{dist}(q,r)$ for all $r \in P$ with $r \neq p$. We have to prove that the vertical line through $q$ intersects $\mathcal{UE}(H)$ at a point lying on $h(p)$. Recall that 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 $P$, the point $p$ has the smallest distance to $q$, so $q(p)$ is the highest intersection point. Hence, the vertical line through $q$ intersects $\mathcal{UE}(H)$ at a point lying on $h(p)$, as claimed. $\square$

This theorem implies that we can compute a Voronoi diagram in the plane by computing the upper envelope of a set of planes in 3-space. By Exercise 11.10 (see also the previous section), the upper envelope of a set of planes in 3-space is in one-to-one correspondence to the lower convex hull of the points $H^*$, so we can immediately use our algorithm CONVEXHULL.

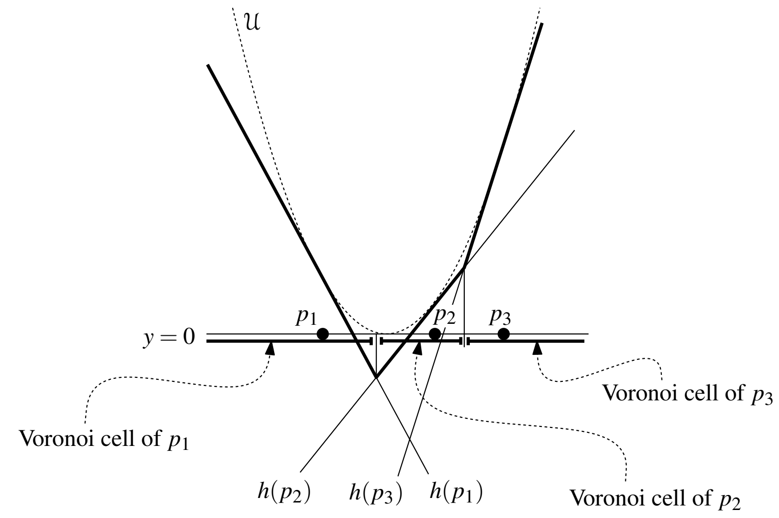


*Figure 11.5*
The correspondence between Voronoi diagrams and upper envelopes

Not surprisingly, the lower convex hull of $H^*$ has a geometric meaning as well: its projection on the plane $z = 0$ is the Delaunay graph of $P$.

## 11.6 Notes and Comments

The early convex hull algorithms worked only for points in the plane—see the notes and comments of Chapter 1 for a discussion of these algorithms. Computing convex hulls in 3-dimensional space turns out to be considerably more difficult. One of the first algorithms was the “gift wrapping” algorithm due to Chand and Kapur [84]. It finds facet after facet by “rotating” a plane over known edges of the hull until the first point is found. The running time is $O(nf)$ for a convex hull with $f$ facets, which is $O(n^2)$ in the worst case. The first algorithm to achieve $O(n \log n)$ running time was a divide-and-conquer algorithm by Preparata and Hong [322, 323]. Early incremental algorithms run in time $O(n^2)$ [223, 344]. The randomized version presented here is due to Clarkson and Shor [133]. The version we presented needs $O(n \log n)$ space; the original paper gives a simple improvement to linear space. The idea of a conflict graph, used here for the first time in this book, also comes from the paper of Clarkson and Shor. Our analysis, however, is due to Mulmuley [290].

In this chapter we have concentrated on 3-dimensional space, where convex hulls have linear complexity. The so-called *Upper Bound Theorem* states that the worst-case combinatorial complexity of the convex hull of $n$ points in $d$-dimensional space—phrased in dual space: the intersection of $n$ half-spaces—is $\Theta(n^{\lfloor d/2 \rfloor})$. (We proved this result for the case $d = 3$, using Euler’s relation.) The algorithm described in this chapter generalizes to higher dimensions, and is optimal in the worst case: its expected running time is $\Theta(n^{\lfloor d/2 \rfloor})$. Interestingly, the best known deterministic convex hull algorithm for odd-dimensional spaces is based on a (quite complicated) derandomization of this algorithm [97]. Since the convex hull in dimensions greater than three can have non-linear complexity, output-sensitive algorithms may be useful. The best known output-sensitive algorithm for computing convex hulls in $\mathbb{R}^d$ is due to Chan [82]. Its running time is $O(n \log k + (nk)^{1-1/(\lfloor d/2 \rfloor+1)} \log^{O(1)} n)$, where $k$ denotes the complexity of the convex hull. A good overview of the many results on convex-hull computations is given in the survey by Seidel [347]. Readers who want to know more about the mathematical aspects of polytopes in higher dimensions can consult Grünbaum's book [194], which is a classical reference for polytope theory, or Ziegler's book [399], which treats the combinatorial aspects.

In Section 11.5 we have seen that the Voronoi diagram of a planar point set is the projection of the upper envelope of a certain set of planes in 3-dimensional space. A similar statement is true in higher dimensions: the Voronoi diagram of a set of points in $\mathbb{R}^d$ is the projection of the upper envelope of a certain set of hyperplanes in $\mathbb{R}^{d+1}$. Not all sets of (hyper)planes define an upper envelope whose projection is the Voronoi diagram of some point set. Interestingly, any upper envelope *does* project onto a so-called power diagram, a generalization of the Voronoi diagram where the sites are spheres rather than points [25].

## 11.7 Exercises

11.1 In Chapter 1 we defined the convex hull of a set $P$ of points as the intersection of all convex sets containing the points. In the current chapter we saw another definition: the convex hull of $P$ is the set of all convex combinations of the points in $P$. Prove that these two definitions are equivalent, that is, prove that a point $q$ is a convex combination of points in $P$ if and only if $q$ lies in every convex set containing $P$.

11.2 Prove that the worst case running time of algorithm ConvexHull is $O(n^3)$, and that there are sets of points where a bad choice of the random permutation makes the algorithm actually need $\Theta(n^3)$ time.

11.3 Describe a randomized incremental algorithm to compute the convex hull of $n$ points in the plane. Describe how to deal with degeneracies. Analyze the expected running time of your algorithm.

11.4 In many applications, only a small percentage of the points in a given set $P$ of $n$ points are extreme. In such a case, the convex hull of $P$ has less than $n$ vertices. This can actually make our algorithm ConvexHull run faster than $\Theta(n \log n)$.

Assume, for instance, that the expected number of extreme points in a random sample of $P$ of size $r$ is $O(r^\alpha)$, for some constant $\alpha < 1$. (This is true when the set $P$ has been created by picking points uniformly at random in a ball.) Prove that under this condition, the running time of the algorithm is $O(n)$.

11.5 The convex hull of a set $P$ of $n$ points in 3-dimensional space can also be computed by "rotating" a plane over known edges of the convex hull, thereby discovering new facets. Give a detailed description of an algorithm using this approach, and analyze its running time.

11.6 Describe a data structure that allows you to test whether a query point $q$ lies inside a convex polytope with $n$ vertices in $\mathbb{R}^3$. (*Hint:* Use the results from Chapter 6.)

11.7 Define a simple polytope to be a region in 3-space that is topologically equivalent to a ball (but not necessarily convex) and whose boundary consists of planar polygons. Describe how to test in $O(n)$ time whether a point lies inside a simple polytope with $n$ vertices in 3-dimensional space.

11.8 Describe a randomized incremental algorithm to compute the intersection of half-planes, and analyze its expected running time. Your algorithm should maintain the intersection of the current set of half-planes. To figure out where to insert a new half-plane, maintain a conflict graph between the vertices of the current intersection and the half-planes that are still to be inserted.

11.9 Describe a randomized incremental algorithm to compute the intersection of half-spaces in 3-dimensional space, and analyze its expected running time. Maintain a conflict graph analogous to the previous exercise.

11.10 In this exercise you have to work out the details of a 3-dimensional duality transformation. Given a point $p := (p_x, p_y, p_z)$ in $\mathbb{R}^3$, let $p^*$ be the plane $z = p_x x + p_y y - p_z$. For a non-vertical plane $h$, define $h^*$ such that $(h^*)^* = h$. Give a definition of the upper convex hull $\mathcal{UH}(P)$ of a set of points $P$ and the lower envelope $\mathcal{LE}(H)$ of a set $H$ of planes in 3-space, similar to the way they were defined for the planar case in Section 11.4.

Show the following properties.

- A point $p$ lies on a plane $h$ if and only if $h^*$ lies on $p^*$.
- A point $p$ lies above $h$ if and only if $h^*$ lies above $p^*$.
- A point $p \in P$ is a vertex of $\mathcal{UH}(P)$ if and only if $p^*$ appears on $\mathcal{LE}(P^*)$.
- A segment $\overline{pq}$ is an edge of $\mathcal{UH}(P)$ if and only if $p^*$ and $q^*$ share an edge on $\mathcal{LE}(P^*)$.
- Points $p_1, p_2, \dots, p_k$ are the vertices of a facet $f$ of $\mathcal{UH}(P)$ if and only if ${p_1}^*, {p_2}^*, \dots, {p_k}^*$ support facets of $\mathcal{LE}(P^*)$ that share a common vertex.