Lecture II
GEOMETRY OF POINT SETS

We said geometry is about relationships among geometric objects. Points in Euclidean space are the simplest of geometric objects. What relationships hold in a finite set S of point? We begin by exploring one of the first non-trivial geometric object that might be defined on S, namely its convex hull. How does one compute it? Even when restricted to points in plane, this will introduce us to a wealth of recurrent ideas in geometric computation. In some sense, all the relationships in a finite set of points is captured by the notion of geometric sorting, introduced by Goodman and Pollack.

§1. Convex Hull of Point Sets

Convexity Concepts. We first introduce some concepts of convexity in the Euclidean space $\mathbb{R}^d$ (of any dimension $d \geq 1$). If $p, q \in \mathbb{R}^d$ are two points, then the closed line segment joining them is the set of points of the form

$$\lambda p + (1 - \lambda)q$$

for $0 \leq \lambda \leq 1$. This segment is denoted $[p, q]$. If we omit the endpoints of this segment, we denote it by (p, q). A set $R \subseteq \mathbb{R}^d$ of points is **convex** if for all $p, q \in R$, we $(p, q) \in R$. A point p is **extremal** in R if there is a closed half-space H that intersects R in exactly p. A **convex polyhedron** is a closed convex set P with a finite number of extremal points. If P is also bounded, it is a **convex polytope**. A **polyhedron** (resp. **polytope**) is a finite union of convex polyhedrons (resp. polytopes). In 2-dimensions, a polytope is call a **polygon**.

A convex polyhedron in $\mathbb{R}^d$ can be partitioned into a collection of **cells**. Each cell is a connected open set in a affine subspace of some dimension i ($i = 0, 1, \ldots, d$). A i-dimensional cell is called a i-cell. The 0-cells are called **vertices**; the 1-cells and $(d-1)$-cells are called **edges** and **facets**, respectively. We will require the cell to interrelate in nice ways. if c, c' are two cells, then the closure of c must be disjoint from c' or contains c'. There may be other restrictions on cells. We can restric such cells to be convex or even simplicial.

For the present, we will concentrate on the case $d = 2$. In this case, polytopes are call **polygons**. These shapes are ubiguitous in daily life, in the form of triangles, rectangles, etc. The subdivision of polygon into cells is intuitively clear: there is only one facet, which is the interior of the polygon. The boundary of the polygon is homeomorphic to a circle, and is naturally partitioned into an alternative sequence of vertices and edges. The number m of vertices is the same as the number of edges, and we call the polygon an m-gon.

The **convex hull** of a set S of points can be defined to be the smallest closed convex set that contains S – this is a convex polytope. Let $CHull(S)$ denote this set. We often identify $CHull(S)$ with the boundary of the convex polytope. In two dimensions, the boundary of a convex polygon with m vertices may be identified with a circular ordering[1] $(v_1, v_2, \ldots, v_m)$ of its vertices.

The convex hull problem is this: given a set S of points in the plane, we want to compute $CHull(S)$. See figure 1 for the convex hull of 8 points. The form of the output needs to be specified with some care: we not only want to determine the points in S which are in extremal position, we also want to list them in the correct circular order. Here, a possible ambiguity arises. Are points on the relative interior of a convex hull edge considered "extremal"? According to our definition of extremal below, such points are not extremal.

[1] The qualification "circular" means that we can choose any vertex of the If $(v_1, v_2, \ldots, v_n)$ is the circular ordering of a set of $n \geq 2$ points, then for any $i = 2, \ldots, n$, $(v_i, v_{i+1}, \ldots, v_n, v_1, v_2, \ldots, v_{i-1})$ denotes the same circular ordering.

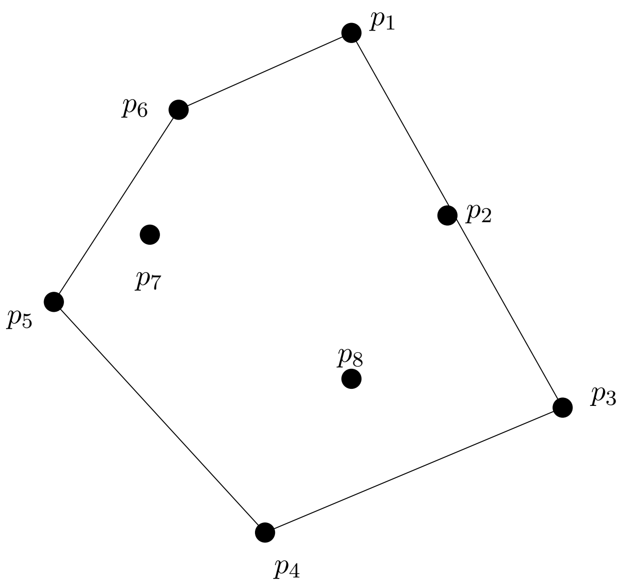

Figure 1: Convex hull of a set of points

But we may still want to list these points as convex hull points. As we will see, our solutions can handle both versions.

§2. Deciding Extremal Points

We reduce our problem to a subproblem: how to decide if an point $p \in S$ is an extremal point of $CHull(S)$? This problem is in turn reduced to an interesting **angular sorting** problem: given S and any $p \in S$, determine the circular ordering of all the other points in S with respect to their angular order seen from p. Then p is extremal iff one of the angles spanned at p by two consecutive points is more than 180 deg.

This would give an $O(n^2 \log n)$ time algorithm for computing convex hull: for each points $p \in S$, we can decide if p is extremal in $O(n \log n)$ time, by circular sorting about p. Once we have the set of extremal points, we can put them in circular order in another $O(n^2)$ time (How). Actually, we can do this in linear time if we just retain a little information from the circular sorting phase above. Namely, when we determined a point p to be extremal, we have also found two unique consecutive points p', p'' such that the angle $\angle(p', p, p'')$ is $> \pi$. This information yields the circular ordering of the convex hull points in a straightforward way.

Angular sorting illustrates the idea of "generic sorting" in which the objects to be sorted can be arbitray, provided a linear ordering can be defined on them. This linear ordering is implicitly given by the "comparison" operator. Such ideas are well-supported in a modern computer language such as `Java`.

§3. Signed Area of a Triangle

Let us attend to the details of doing angular sorting. The obvious way to do this is to compute the angles that each point $q \in S \setminus \{p\}$ makes with the positive x-axis. Then we sort these angles as numbers between 0 and 2π. But we prefer to avoid the computation of angles as this requires non-algebraic (transcendental) functions, which is inherently approximate. To do this, we need to define the signed area of an oriented triangle.

Let A, B, C be three points in the plane. The area $\alpha(A, B, C)$ of the triangle ABC is half of the determinant

$$\Delta := \det \begin{bmatrix} a & a' & 1 \\ b & b' & 1 \\ c & c' & 1 \end{bmatrix} = \det \begin{bmatrix} b-a & b'-a' \\ c-a & c'-a' \end{bmatrix} \tag{1}$$

where $A = (a, a')$, $B = (b, b')$, etc. This is a signed quantity. Clearly, $\alpha(A, B, C) = -\alpha(A, C, B)$, since the determinant of a matrix changes sign when we exchange a pair of adjacent rows in the matrix. We can justify this formula for $\alpha(A, B, C)$ by using cross-product formula, for 3-dimensional vectors. However, here is a more direct argument. First, translate the triangle so that A is at the origin: so $a = a' = 0$. First assume that both vectors (b, b') and (c, c') are in the first quadrant, with vector (b, b') making a smaller angle with the positive x-axis than vector (c, c'). Referring to figure 2, the area of the parallelogram P is equal to the area of the enclosing rectangle $(b + b')(c + c')$ minus the area of the two smaller rectangles $2R = 2bc'$, minus the area of the 4 triangles $2(T_1 + T_2) = bb' + cc'$. This yields

$$\Delta = (b + b')(c + c') - (2bc' + bb' + cc') = b'c - bc'.$$

In case the vector (b, b') makes a larger angle with the positive x-axis than vector (c, c'), we obtain the same area with an opposite sign. We leave it as an exercise to verify this formula even when the 2 vectors are not both in the first quadrant. Furthermore, if A, B, C are non-collinear, *the signed area of triangle ABC is positive iff A, B, C represents a counter-clockwise rotation.* There is a generalization to volume in 3 and higer dimensions.

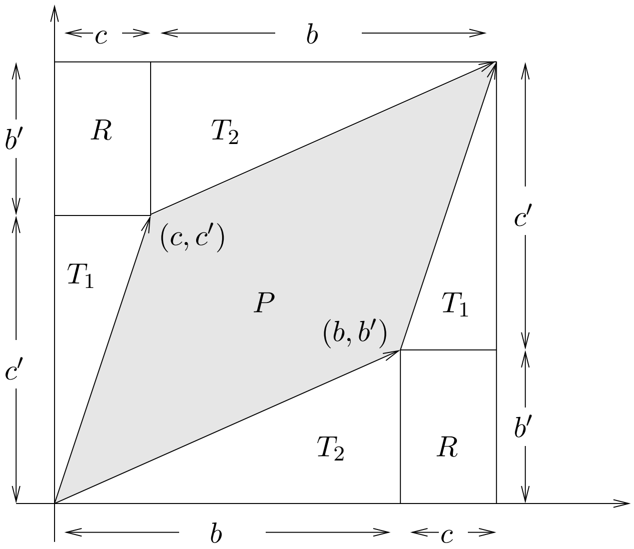

Figure 2: Area of parallelogram determined by vectors $(b, b'), (c, c')$.

§4. Angular Sorting

The **LeftTurn Predicate** is defined as follows: $\texttt{LeftTurn}(A, B, C) = TRUE$ if the point C is strictly to the left of the line directed from A to B. Thus

$$\texttt{LeftTurn}(A, B, C) \equiv AREA(A, B, C) > 0$$

How can we do circular sorting with the `LeftTurn` predicate? First, let us suppose A has the smallest y-coordinate among the points in S. Then the angular sorting of S about A is easily done: Let C be the sorted list of the points in S, where each point B is sorted by the angle $\angle(XAB)$ where X is any point to the right of A. For points $p, q \in S$, p appears before q iff $\texttt{LeftTurn}(A, p, q)$. If A does not have the smallest y-coordinate, we partition S into $S = S_0 \uplus S_1$ where with S_0 comprising those points whose y-coordinate are greated than the y-coordinate of A. We can now sort S_0 and S_1 separately, and join them into one circular list.

§5. Gift Wrapping

We can improve the $O(n^2 \lg n)$ algorithm for convex hulls to $O(nh)$ where h is the number of points on the convex hull of S. The idea is that we begin with any point p_1 on the convex hull of S. For instance, p_1 can be chosen to the the point with the smallest y-coordinate (if there are several choices, the smallest x-coordinate will do). We perform the circular sorting at p_1. This will lead us to the point p_2 which is next on the convex hull, and we again perform a circular sorting about p_2, etc. Thus we perform only h circular sorts.

But actually we need not perform sorting to find p_{i+1} from p_i. We only need to find the point with minimim angle in the direction p_{i-1} to p_i. This can be performed in $O(n)$ time. Thus the algorithm is $O(nh)$.

How can we effect this physical action of rotating until we find the next point p_{i+1}? Using the direction from p_{i-1} to p_i as the initial direction, we need to find the point p such that the direction from p_i to p is smallest. Using the ideas above, this is easy to do in linear time. Hence the complexity of this algorithm is $O(nH)$ where $H \le n$ is the size of the hull. If $H = o(\log n)$, this can be faster than an $O(n \log n)$ algorithm.

Graham Scan Graham gave one of the first $O(n \log n)$ algorithm for this problem.

1. The first step is to find the $P :: (x, y)$ which is lexicographically minimum (minimize x and then minimize y). Let $P_0 :: (x_0, y_0)$ be this point. Clearly, P_0 is extremal be any definition of this word.
2. Sort the other points angularly about P_0. Angles are measured counter-clockwise from the negative y-axis. If two points have the same angle, drop the one with smaller x-coordinate. Let
$$P_0, P_1, \ldots, P_{n-1}$$
be the list of sorted points. Note that P_{n-1} is on the convex hull.
3. Create a stack S of points. Initials S with (P_{n-1}, P_0). In general, if $S = (Q_0, Q_1, \ldots, Q_k)$ were $k \ge 2$, then let *top* refer to Q_k and *next* refer to Q_{k-1}. To push an element Q onto S would make $S = (Q_1, \ldots, Q_k, Q)$. Similarly, for popping an element.
4. We process the points P_i in the i-th stage. If P_i is to the left of the line from *next* to *top* then push(P_i). Else, pop the stack.

Correctness of Graham Scan: we leave this as an exercise.

Divide and Conquer This is quite intuitive: we choose a vertical line L such that at most half of the points lies strictly to the left, and at most half of the points lies strictly to the right. Recursively, we construct the convex hull of the left and right subsets. We then construct the convex hull of the whole set by constructing the two "bridges" (these are the lines that are the common tangents of both convex hulls, and that has the 2 convex hulls on one side of the line). Once we have these lines, we can easily construct the final convex hull. This gives rise to the classic recurrence

$$T(n) \leq n + 2T(n/2)$$

which has solution $T(n) = O(n \log n)$.

Lower Bounds It can be shown that within a reasonable model of computation, every convex hull algorithm requires $\Omega(n \log n)$ steps in the worst case. In this sense, we usually call an algorithm with running time $O(n \log n)$ an "optimal algorithm". But witness the next result.

Ultimate Convex Hull ALgorithm A remarkable algorithm was discovered by Kirkpatrick and Seidel: their algorithm takes time $O(n \log H)$. Thus, it is simultaneously better than gift-wrapping and $O(n \log n)$. Notice that we manage to beat other "optimal algorithms". Of course, this is no contradiction because we have changed the rules of the analysis: we took advantage of another complexity parameter, h. This parameter h is a function of some **output size parameter**. Algorithms that can take advantage of the output size parameters are said to be **output-sensitive**. This is a very important idea of computational geometry. In terms of this h parameter, a lower bound of $\Omega(n \log h)$ can be shown.

Incremental Algorithms and Randomization There is another very general approach called incremental convex hull. In 2-D, this is quite easy to understand. We will illustrate this using the Beneath-Beyond algorithm for convex hulls in every dimension.

REF: Edelsbrunner pa.ge 147.

The idea is to

We will return to this later, to introduce two other important themes in computational geometry: randomization and incremental algorithms. Moreover, in dimension 3 or more, we need more sophisticated data structures which we will return to.

Another direction is to generalize the problem to the "dynamic setting". Here the point set S is considered a dynamic point set which can change over time. We can insert or delete from S. Incremental algorithms can be regarded as the solution for the "semi-dynamic" case in which we can only insert, but not delete points. Yet another direction is to consider the convex hull computation in higher (or arbitrary) dimensions.

§6. Geometric Sorting

The problem of geometric sorting amounts to asking for the complete set of orientation relations among a set of points. For a planar point set S, this amounts to knowing the value of `LeftTurn`(p, q, r) for all triple p, q, r of points in S. Clearly, this problem can be solved in $O(n^3)$ time. But we can reduce this to $O(n^2 \log n)$ time as follows:

If S is a set of n points in $\mathbb{R}^d$, we can similarly ask for the signs of the determinants formed by any sequence of $d+1$ points in S. More precisely, if the points p_i of S are indexed by $1, \ldots, n$, define the function χ_S which assigns to each $d+1$ indices $(i_0, i_1, \ldots, i_d)$ the sign of the determinant $\det(p_{i_0}, \ldots, p_{i_d})$. Note that this is a determinant of the matrix whose jth row is $(p_{i_j}, 1)$. This function is called the **chirotope** of S, and there are several other equivalent formulations. When the properties of chirotopes are axiomatized, we obtain the theory of oriented matroids. We refer to the the book of Björner et al [1].

Exercises

Exercise 6.1: Verify the area formula for all possible disposition of the vectors (b, b') and (c, c'). $\diamondsuit$

Exercise 6.2: Give a formula for the signed area of a simple polygon whose vertices are $A_1, A_2, \ldots, A_n$ (for $n \geq 3$). Let $A_i :: (x_i, y_i)$ for each i. $\diamondsuit$

Exercise 6.3: Can you derive the signed area of a tetrahedron in 3 dimensions? What is the volume formula for a d-dimensional simplex for $d > 3$? (In d dimensions, there is a constant c_d analogous to $c_2 = 1/2$ in 2 dimensions.) $\diamondsuit$

End Exercises

References

[1] A. Björner, M. L. Vergnas, B. Sturmfels, N. White, and G. M. Ziegler. *Oriented Matroids.* Encyclopedia of Mathematics, Vol. 46. Cambridge University Press, 1993.