|  |
| --- |
| **Lecture 2: Orientations and Convex Hulls** |
| Reading: Chapter 1 in the 4M’s (de Berg, van Kreveld, Overmars, Schwarzkopf). The divide-and-conquer algorithm is given in Joseph O’Rourke’s, “Computational Geometry in C.” O’Rourke’s book is also a good source for information about orientations and some other geometric primitives. |
| **Orientation:**  In order to make discrete decisions, we would like a geometric operation that operates on points in a manner that is analogous to the relational operations (<;=;>) with numbers. There does not seem to be any natural intrinsic way to compare two points in d-dimensional space, but there is a natural relation between ordered (d + 1)-tuples of points in d-space, which extends the notion of binary relations in 1-space, called orientation. Given an ordered triple of points hp; q; ri in the plane, we say that they have positive orientation if they define a counterclockwise oriented triangle, negative orientation if they define a clockwise oriented triangle, and zero orientation if they are collinear (which includes as well the case where two or more of the points are identical). Note that orientation depends on the order in which the points are given. |
|  |
| Figure 1: Orientations of the ordered triple (p; q; r). |
| Orientation is formally defined as the sign of the determinant of the points given in homogeneous coordinates, that is, by prepending a 1 to each coordinate. For example, in the plane, we define |
|  |
| Observe that in the 1-dimensional case, Orient(p; q) is just q¡p. Hence it is positive if p < q, zero if p = q, and negative if p > q. Thus orientation generalizes <;=;> in 1-dimensional space. Also note that the sign of the orientation of an ordered triple is unchanged if the points are translated, rotated, or scaled (by a positive scale factor). A reflection transformation, e.g., f(x; y) = (¡x; y), reverses the sign of the orientation. In general, applying any affine transformation to the point alters the sign of the orientation according to the sign of the matrix used in the transformation.  In general, given an ordered 4-tuple points in 3-space, we can also define their orientation as being either positive (forming a right-handed screw), negative (a left-handed screw), or zero (coplanar). This can be generalized to any ordered (d + 1)-tuple points in d-space. |
| **Areas and Angles:**  The orientation determinant, together with the Euclidean norm can be used to compute angles in the plane. This determinant Orient(p; q; r) is equal to twice the signed area of the triangle ;&delta pqr (positive if CCW and negative otherwise). Thus the area of the triangle can be determined by dividing this quantity by 2. In general in dimension d the area of the simplex spanned by d + 1 points can be determined by taking this determinant and dividing by (d!). Once you know the area of a triangle, you can use this to compute the area of a polygon, by expressing it as the sum of triangle areas. (Although there are other methods that may be faster or easier to implement.) Recall that the dot product returns the cosine of an angle. However, this is not helpful for distinguishing positive from negative angles. The sine of the angle (the signed angled from vector p-q to vector r - q ) can be computed as sin µ = |p – q|| r-q| Orient(q, p, r), (Notice the order of the parameters.) By knowing both the sine and cosine of an angle we can unambiguously determine the angle. |
| **Convexity:**  Now that we have discussed some of the basics, let us consider a fundamental structure in computational geometry, called the convex hull. We will give a more formal definition later, but the convex hull can be defined intuitively by surrounding a collection of points with a rubber band and letting the rubber band snap tightly around the points. |
|  |
| Figure 2: A point set and its convex hull. |
| There are a number of reasons that the convex hull of a point set is an important geometric structure. One is that it is one of the simplest shape approximations for a set of points. It can also be used for approximating more complex shapes. For example, the convex hull of a polygon in the plane or polyhedron in 3-space is the convex hull of its vertices. (Perhaps the most common shape approximation used in the minimum axis-parallel bounding box, but this is trivial to compute.) |
| Also many algorithms compute the convex hull as an initial stage in their execution or to filter out irrelevant points. For example, in order to find the smallest rectangle or triangle that encloses a set of points, it suffices to first compute the convex hull of the points, and then find the smallest rectangle or triangle enclosing the hull. |
| **Convexity:**  A set *S* is *convex* if given any points any convex combination of *p* and *q* is in *S*, or equivalently, the line segment  **Convex hull:**  The *convex hull* of any set *S* is the intersection of all convex sets that contains *S*, or more intuitively, the smallest convex set that contains *S*. Following our book’s notation, we will denote this .  An equivalent definition of convex hull is the set of points that can be expressed as convex combinations of the points in S. (A proof can be found in any book on convexity theory.) Recall that a convex combination of three or more points is an affine combination of the points in which the coefficients sum to 1 and all the coefficients are in the interval [0; 1]. |
| **How do we compute the convex hull?**  Before we can answer this question we must ask another question: what does it mean to compute the convex hull? As we have seen, the convex hull of P is a convex polygon. A natural way to represent a polygon is by listing its vertices in clockwise order, starting with an arbitrary one. So the problem we want to solve is this: given a set P = {p1, p2, . . . , pn} of points in the plane, compute a list that contains those points from P that are the vertices of CH(P), listed in clockwise order. |
|  |
| C:\AppData\Local\Temp\msohtmlclip1\01\clip_image001.pngC:\AppData\Local\Temp\msohtmlclip1\01\clip_image001.png |
|  |
|  |
|  |
|  |
|  |
| **Incremental Algorithm of Convex Hull** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |