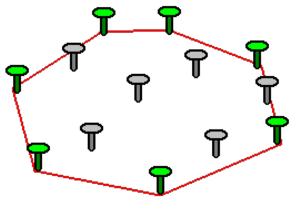
Convex Hull

Contents

Convex Hull

The convex hull is a ubiquitous structure in computational geometry. Even though it is a useful tool in its own right, it is also helpful in constructing other structures like Voronoi diagrams, and in applications like unsupervised image analysis.

We can visualize what the convex hull looks like by a thought experiment. Imagine that the points are nails sticking out of the plane, take an elastic rubber band, stretch it around the nails and let it go. It will snap around the nails and assume a shape that minimizes its length. The area enclosed by the rubber band is called the convex hull of P. This leads to an alternative definition of the convex hull of a finite set P of points in the plane: it is the unique convex polygon whose vertices are points from P and which contains all points of P.

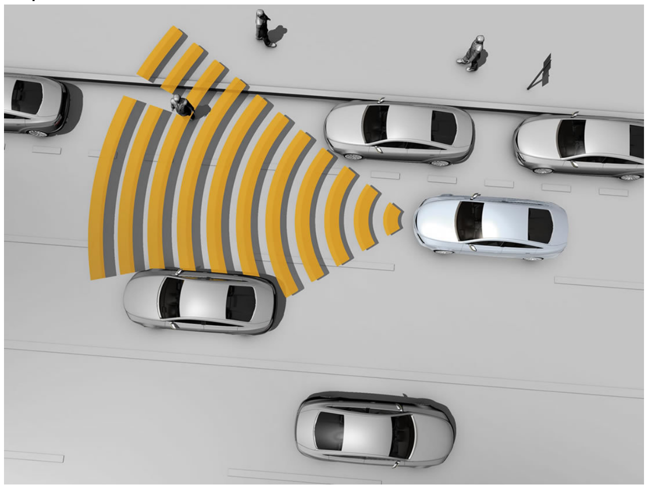


The set of green nails are the convex hull of the collection of the points.

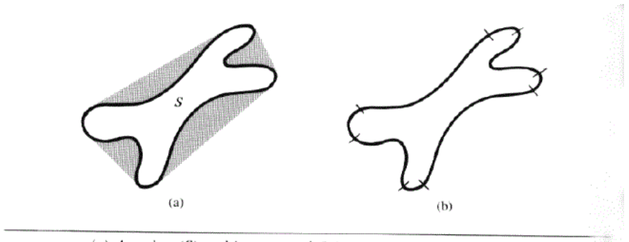
Applications

A few of the applications of the convex hull are:

- **Collision avoidance**: If the convex hull of a car avoids collision with obstacles then so does the car. Since the computation of paths that avoid collision is much easier with a convex car, then it is often used to plan paths.



- **Smallest box**: The smallest area rectangle that encloses a polygon has at least one side flush with the convex hull of the polygon, and so the hull is computed at the first step of minimum rectangle algorithms. Similarly, finding the smallest three-dimensional box surrounding an object depends on the 3D-convex hull.
- **Shape analysis**: Shapes may be classified for the purposes of matching by their "convex deficiency trees", structures that depend for their computation on convex hulls.



(a) A region (S) and its convex deficiency (shaded); (b) partitioned boundary.

Formal definitions of Convexity and Convex Hulls

DEFINITION

Convexity A set S is convex if $x \in S$ and $y \in S$ implies that the segment $xy \subseteq S$. This can be taken as the primary definition of convexity. Note that this definition does not specify any particular dimensions for the points, whether S is connected, bounded, unbounded, closed or open.

We have now developed an intuitive definition of the convex hull. Let us now look at more precise definitions of the convex hull.

DEFINITION

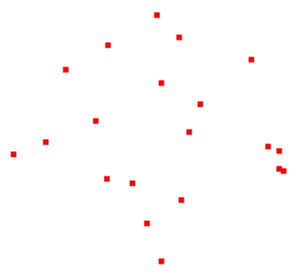
Convex hull

The convex hull of a set of points S is the intersection of all half-spaces that contain S. A half space in two dimensions is the set of points on or to one side of a line. This notion generalizes to higher dimensions. A *half-space* is the set of points on or to one side of a plane and so on.

Note that the convex hull of a set is a closed "solid" region which includes all the points on its interior. Often the term is used more loosely in computational geometry to mean the boundary of this region, since it is the boundary that we compute, and that implies the region.

Graham's Algorithm

Graham's scan algorithm is a method of computing the convex hull of a finite set of points in the plane with time complexity $O(n \log n)$.The algorithm finds all vertices of the convex hull ordered along its boundary .



The procedure in Graham's scan is as follows:

- Find the point with the lowest y coordinate. If there are two points with same y value, then the point with smaller x coordinate value is considered. Put the bottom-most point at first position.
- Consider the remaining $n-1$ points and sort them by polar angle in counterclockwise order around points$[0]$. If polar angle of two points is same, then put the nearest point first.
- Create an empty stack S and push points$[0]$, points$[1]$ and points$[2]$ to$S$.
- Process remaining $n-3$ points one by one. Do the following for every point 'points$[i]$'
  - Keep removing points from stack while orientation of following 3 points is not counterclockwise (or they don't make a left turn).
    - Point next to top in stack
    - Point at the top of stack
    - Points$[i]$
  - Push points[i] to S

A pseudocode implementation of the above procedure is:

```
#Let p_0 be the point in G with the smallest y-coordinate. In case there is a tie, it is the left most element.
#Let {p_1, p_2, ... ,p_m } be the set of remaining point in G, sorted by polar angle in clock wise order around p_0( if more than one
Stack S
S.push(p_0)
S.push(p_1)
S.push(p_2)
function next_to_top(stack)
   returns the point one entry below the top of the stack S
function top(stack)
  returns the top point of a stack



for i <- 3 to m
        do while the angle formed by points next_to_top(S), top(s), and p_i make a turn which is not left(either straight or to the ri
                do S.pop()
             S.push(p_i,S)
return S
```

Runtime of the graham's scan

• Step 1: $O(n)$+$O(n \log n)$ for setting up and sorting

• Step 2: $O(1)$ constant time for pushing items into the stack

• Step 3: $O(n)$ each point gets pushed once withing the for loop

• Step 4 $O(n)$ for popping within the loop , each point gets popped once at most

• Total running time: $O(n \log n)$

The bottleneck of the algorithm is sorting the points by polar angles. This operation as we have seen requires $O(n \log n)$ time.

Extreme Edges

Identifying extreme edges of the convex hull is somewhat easy. An edge is extreme if every point on S is on or to one side of the line determined by the edge. It seems easiest to detect this by treating the edge as directed, and specifying one of the two possible directions as determining the "side". Let the left side of a directed edge be inside. Phrased negatively, a directed edge is not extreme if there is some point that is not left of it or on it. This is the formulation we use in the pseudo-code below.

```
method Extreme_Edges:
```

This algorithm clearly runs in $O(n^3)$ time because there are three nested loops, each costing $O(n)$.

Gift Wrapping

A minor variation of the Extreme Edge algorithm will both improve it by a factor of n and output the points in the order in which they occur around the hull boundary. The idea is to use on extreme edge as an anchor for finding the next. This works because we know that the extreme edges are kinked into a convex polygon. Since the most vertices this polygon can have is n, the number of extreme edges is $O(n)$. The anchored search will only explore $O(n)$ candidates, rather than $O(n^2)$ candidates in our extreme edge algorithm above.

Cite as: Convex Hull. *Brilliant.org*. Retrieved 12:27, November 19, 2019, from https://brilliant.org/wiki/convex-hull/