Computational Geometry

Convexity and convex hull algorithms

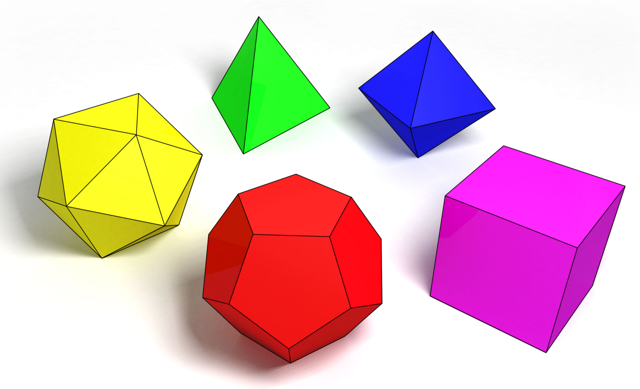
Vissarion Fisikopoulos

Department of Informatics & Telecommunications
National & Kapodistrian University of Athens

Spring 2025

Why Convex Sets?

- The simplest generalization of linear sets, covering important problems and applications
- Optimization in convex sets (linear programming)
- Representation of complex objects



Computational Model

Real RAM (Random Access Machine):

- Exact representation, storage of real numbers in $O(1)$ space
- Unit-time memory access
- Unit-time, absolute precision for basic operations in $\mathbb{R}$

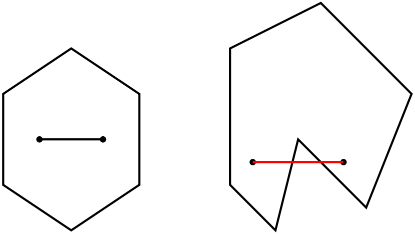
Implementations:

- A satisfactory implementation of the model is the CGAL library.
- Numerical errors in computational geometry can lead to incorrect results or cause program crashes.

Definition of Convex Hull

Definition (Convexity)

A set S is **convex** if and only if $a, b \in S \Rightarrow$ the line segment $(a, b) \subset S$.



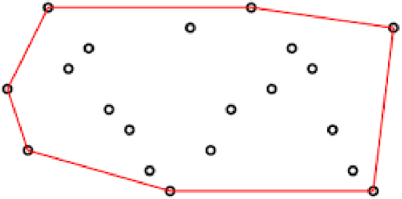
Exercise

Equivalently, S is convex if and only if there exists a point $p \in S$ from which all points of S are visible, meaning they can be connected by a line segment lying entirely within S.

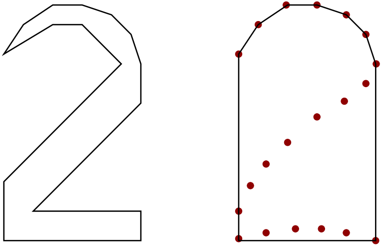
Convex Hull (CH) in Two Dimensions

Definition (CH2)

- n points $A_1, A_2, \ldots, A_n$ in $\mathbb{R}^2$.
- The convex hull (CH) of a set of points is the smallest (area, perimeter, num of points) **convex** set (polygon) that contains all A_i.



Convex Hull in 2 Dimensions



A non-convex polygon and the construction of the CH of points in the plane.

Combinations of Points

Remark

We often identify a point A with the vector $(0, A)$, which is not (free), meaning it does not move in space.

Definition (Combinations of Points/Vectors A_i)

- Linear combination: $\lambda_1 A_1 + \cdots + \lambda_n A_n,\ \ \lambda_i \in \mathbb{R}$.
- Positive (conical) combination: $\lambda_1 A_1 + \cdots + \lambda_n A_n,\ \ \lambda_i \geq 0$.
- Affine combination: $\lambda_1 A_1 + \cdots + \lambda_n A_n,\ \ \sum_i \lambda_i = 1$.
- Convex combination: $\lambda_1 A_1 + \cdots + \lambda_n A_n,\ \ \sum_i \lambda_i = 1,\ \lambda_i \geq 0$.

Affine Combination

Remark

Given an affine combination of $A_1, \ldots, A_n$, the point

$$P = \lambda_1 A_1 + \cdots + \lambda_n A_n, \quad \sum_i \lambda_i = 1.$$

Equivalently:

$$P = A_n + \lambda_1 (A_1 - A_n) + \cdots + \lambda_{n-1} (A_{n-1} - A_n),$$

for any $\lambda_1, \ldots, \lambda_{n-1} \in \mathbb{R}$. If we set A_n as the origin $A_n = 0$, then P is a linear combination of $A_1, \ldots, A_{n-1}$.

Combinations of Points: Example

Example (Combinations)

Let $A_1, A_2 \in \mathbb{R}^2$ be linearly independent:

- Linear: $\{\lambda_1 A_1 + \lambda_2 A_2 : \lambda_i \in \mathbb{R}\} = \mathbb{R}^2$.
- Positive: $\{\lambda_1 A_1 + \lambda_2 A_2,\ \lambda_i \geq 0\}$ = cone of A_1, A_2, with vertex at $(0, 0)$.
- Affine combination: $\{\lambda_1 A_1 + \lambda_2 A_2 :\ \lambda_1 + \lambda_2 = 1\}$ = line passing through A_1, A_2.
- Convex combination: $\{\lambda_1 A_1 + \lambda_2 A_2 :\ \lambda_1 + \lambda_2 = 1,\ \lambda_i \geq 0\}$ = line segment (A_1, A_2).

Properties of the CH

Corollary

- *The vertices $P_1, \ldots, P_k$ of the CH belong to the input set $A_1, \ldots, A_n$.*
- *The points of the CH are convex combinations of the vertices: $\lambda_1 P_1 + \cdots + \lambda_k P_k,\ \sum_i \lambda_i = 1,\ \lambda_i \geq 0$, and therefore also of the A_i.*
- *Every convex combination of the P_i, or the A_i, belongs to the CH.*

Proposition (Carathéodory)

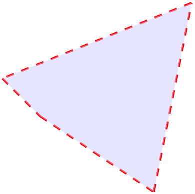
Every point of the CH is a convex combination of at most 3 vertices: $\lambda_1 P_1 + \lambda_2 P_2 + \lambda_3 P_3,\ \sum_i \lambda_i = 1,\ \lambda_i \geq 0$.

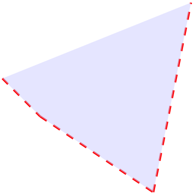
Intersection of halfspaces

An equivalent representation of a convex polygon P with k edges is the intersection of k half-planes:

$$P = \bigcap_{i=1}^{k} H_i$$

where H_i is the half-plane defined by the line of the i-th edge and contains the remaining edges.
Conversely, any intersection of a finite number of half-planes is a bounded convex polygon or an unbounded convex polygonal region.





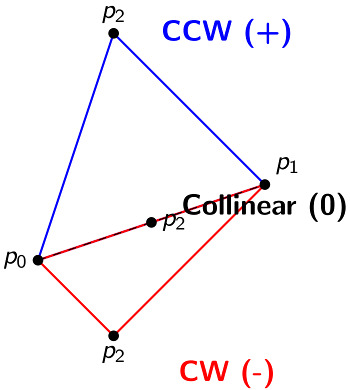
Orientation Predicate - CCW

- **Predicate**: A test for a geometric property. The output takes discrete values (e.g., 2 or 3 values).
- The **orientation predicate** determines if three points $p_0, p_1, p_2 \in \mathbb{R}^2$ define a positive or negative turn (Right-Hand Rule).

Orientation Predicate - CCW

Vectors $v_i = (p_0, p_i)$, their rotation is:

- Negative if and only if $v_1 \times v_2$ has 3rd coordinate < 0 (ClockWise, CW).
- Positive if and only if $v_1 \times v_2$ has 3rd coordinate > 0 (CounterClockWise, CCW).
- Undefined if and only if $v_1 \times v_2$ has 3rd coordinate $= 0$ (3 collinear p_i meaning v_i are parallel).



Computation of CCW

Lemma

The **CCW** *of points $p_i = (x_i, y_i)$ reduces to the sign of the determinant:*

$$\det \begin{bmatrix} x_1 - x_0 & y_1 - y_0 \\ x_2 - x_0 & y_2 - y_0 \end{bmatrix}$$

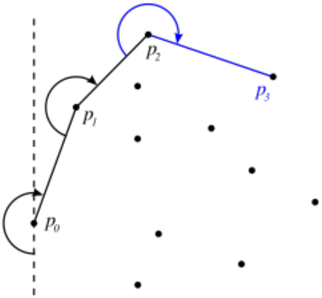
Lemma

- *The CCW computes on which side (half-plane) of the line through p_0, p_1 the point p_2 lies.*
- *The CCW computes the direction of the turn defined by the points p_0, p_1, p_2 in this order. The sign is positive if the turn is counterclockwise, and zero if the three points are collinear.*

Gift Wrapping

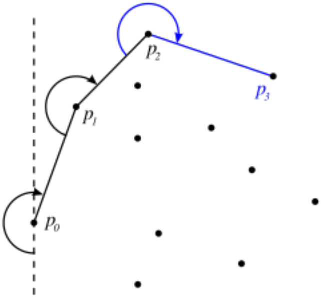
Jarvis Algorithm

- Start with the leftmost point p_0.
- Iterate over all points to find the one that minimizes the angle with the current edge
- At point p_k, select a candidate point u, then for each point x if $\mathrm{CCW}(p_k, u, x) > 0$ update u with x



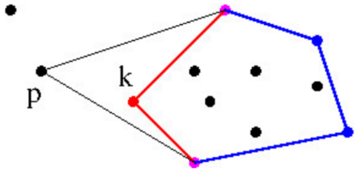
Wrapping Algorithm Complexity

- Initialization: $O(n)$.
- Iterates h (#-CH-vertices) times, each step takes $O(n)$.
- Total time: $O(nh)$.



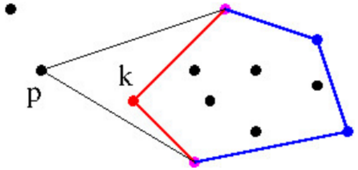
Incremental Algorithm

- The convex hull is updated with each new point.
- Sorting points lexicographically.
- Setup:
  - Current point p; previous point k
  - red/blue edges: visible/non-visible edges of current hull
  - purple vertices: intersections of red/blue edges



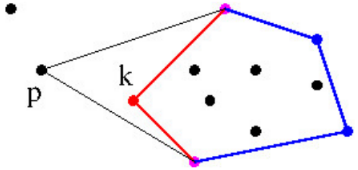
Beneath-Beyond Algorithm

- Input: n points in $\mathbb{R}^2$, in general position.
- Output: Edge and vertex chain of the convex hull.
  1. Sort points lexicographically.
  2. Initialize convex polygon with three points.
  3. For each new point p, update the convex hull structure.
     - Examine the edges incident to k: is there a red one?
     - Coloring: Starting from a red edge, find all red edges and two blue edges, i.e., two purple vertices.
     - Replace the red edges with two new ones: each defined by p and a maroon vertex.



Complexity of Incremental Algorithm

- Initialization: $O(n \log n)$.
  - Finding red edge: $O(1)$; total $O(n)$.
  - Coloring all red edges $<$ # all created edges $< 2n$.
  - Updating convex hull: $O(1)$; total $O(n)$.
- Total time: $O(n \log n)$.



Beneath-and-Beyond predicates

Ordering of the x-coordinates, i.e., deciding whether $x_i < x_j \in \mathbb{R}$, is determined by the sign of the determinant:

$$\det \begin{bmatrix} x_i & 1 \\ x_j & 1 \end{bmatrix}$$

Edge coloring (A, B) with respect to a new point P: - The edge is red/blue if and only if the line through it places P and the existing convex polygon in different/same half-planes. - Equivalently, if and only if the two (nonzero) signs

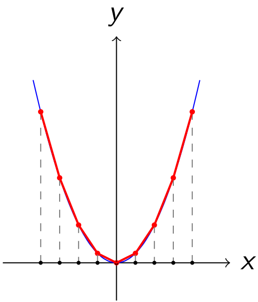
$$\operatorname{sign}\det \begin{bmatrix} A_x & A_y & 1 \\ B_x & B_y & 1 \\ P_x & P_y & 1 \end{bmatrix}, \quad \operatorname{sign}\det \begin{bmatrix} A_x & A_y & 1 \\ B_x & B_y & 1 \\ Q_x & Q_y & 1 \end{bmatrix}$$

differ/are equal, where Q is any point in the existing convex polygon.

Lower Bound on Convex Hull Complexity

Key Observations:

- Convex hull computation has the same lower bound as sorting.
- Reduction: Given numbers $x_1, \ldots, x_n$, construct points (x_i, x_i^2).
- These points lie on a convex parabola; their convex hull gives a sorted order.
- Sorting has a lower bound of $\Omega(n \log n)$



Convex Hull Algorithms

- 1970: **Gift wrapping (Jarvis march)** — $O(nh)$
- 1972: **Graham scan** — $O(n \log n)$
- 1977: **Quickhull** — Expected $O(n \log n)$, worst-case $O(n^2)$
- 1977: **Divide and conquer (Merge hull)** — $O(n \log n)$
- 1979: **Monotone chain (Andrew's algorithm)** — $O(n \log n)$
- 1984: **Incremental convex hull algorithm** — $O(n \log n)$
- 1986: **Kirkpatrick–Seidel algorithm** — $O(n \log h)$
- 1996: **Chan's algorithm** — $O(n \log h)$

Convex Hull Algorithms

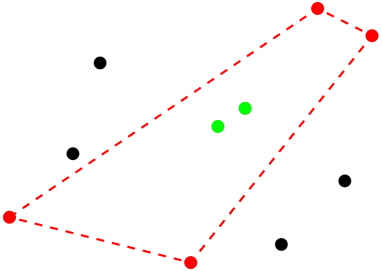
Summary:

- Algorithms range from $O(nh)$ to optimal $O(n \log h)$ complexity.
- Divide and conquer, Graham scan, and monotone chain are widely used $O(n \log n)$ methods.
- Chan's algorithm and Kirkpatrick–Seidel algorithm achieve optimal output-sensitive performance.

Akl–Toussaint Heuristic

Reducing the Number of Points for Convex Hull Computation

- Selecting an initial set of extreme points (e.g., the four points with min/max x and y coordinates).
- Discarding any point that lies inside the quadrilateral formed by these extreme points.



References

Books:

- F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer, 1985.
- M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, Springer, 2008.
- J. O'Rourke, *Computational Geometry in C*, Cambridge University Press, 1998.

Papers:

- S. G. Akl and G. T. Toussaint, "A fast convex hull algorithm," Information Processing Letters, 1978.
- R. L. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," IPL, 1972.
- T. M. Chan, "Optimal output-sensitive convex hull algorithms in two and three dimensions," Discrete & Computational Geometry, 1996.

Online Resources:

- `https://en.wikipedia.org/wiki/Convex_hull`
- `https://www.cgal.org/` (CGAL Library)
- `https://www.boost.org/doc/libs/release/libs/geometry/doc/html/geometry/reference/algorithms/convex_hull.html` (Boost.Geometry)
- A History of Linear-time Convex Hull Algorithms for Simple Polygons