# Administration

Textbook: Computational Geometry – algorithms and application (3rd edition), the lectures will be following this book.

Homework assignment: will consist 30% of the final grade. There will be a total of 4 homework assignment during this quarter.

Midterm at week 5: 30%

Optional programming assignment for extra 10 points.

## What is Computational Geometry?

Many areas in CS use algorithms that originated in CG, like astrophysics, robotics, computer vision etc.

Computer graphics has many applications of CG, like model simplification, motion graphics (animation) etc.

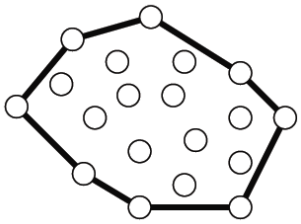
# Topics covered

**Convex Hull**

Given points in the plane $p_1, \dots, p_n$ we want to find an object that contains all points – the convex hull.

Convex Set:

A geometric set is **convex** if for any two points $p, q$ in the set, the line $(p, q)$ is also contained in the set. The smallest convex set for a given set of points, or any objects other than points, is called the **convex hull**.



The problems we will deal with is: given a set of points / objects, find the convex hull containing them.

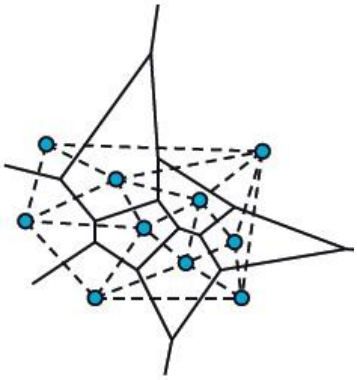
**Triangulation and Partitioning**:

Triangulation is dividing a complex object into a set of simple objects, usually triangles (or tetrahedron in the 3-dimensional space, or simplex in general).

**Voronoi Tessellation problems**

Assume we have points $p_1, \dots, p_n$. we would like to decompose the plane into $n$ parts, each closest to each of the points. This decomposition is the **Voroni decomposition** or **Voroni diagram** of the plane.

In case of 2 points is easy – you connect them to each other, find the midpoint and cross a line between them which divides the plane into two parts.



Note that some of the Voroni regions are unbounded and some are closed and bounded. When we calculate the convex hull of those points, some of them form the outline of the convex hull – the Voroni regions of those points will be the ones unbounded. Everything inside the convex hull will have a bounded region. This is the basic relation between computing the convex hull and the VD (Voroni decomposition).

Application: service providers distribution across a city – you would want to go to the service center / use antenna of the Voroni region you're at.

**Delaunay Triangulation**:

If we connect all neighboring regions (those that share a Voroni edge), we get the dotted diagram – which is called the Delaunay Triangulation. This triangulation is subject to the fact that the smallest angle is as large as possible.

Voroni diagram and Delaunay triangulation are dual object – for a given VD you can calculate the DT, and vice versa.

This is important for high dimensions – as computing the DT is easier than the VD, and computing the first allows knowing the later.

## Search in Geometrical Space

Range queries:

Given $n$ points and a region, we want to know what points fall in the region. The regions could be boxes, spheres etc.

Nearest neighbor search queries:

Determine what points are closest to a given query point.

All the above are the geometrical type of problems we will be dealing with. Next we will focus techniques to solves such problems.

## Fundamentals

The data is a $k$-dimensional Euclidean space: a k-sized vector of real numbers.

The measurement we will be using is usually the Euclidean distance between the points: $d(p_1, p_2) = \sqrt{\sum_{i=1}^{k} |y_i - x_i|^2}$ – the length of the line between two given points in $\mathbb{R}^k$.

Line:

The line between two points $p_1, p_2$ is parameterized as: $\alpha p_1 + (1-\alpha)p_2$, that is any point on the line $q$ can be described using this formula for some $\alpha$.

The affine combination (or convex combination) of the points $p_1, p_2$ is the line that goes between these two points. In this case, $\alpha$ in the formula above is restricted to $\alpha \in [0,1]$.

$k$-dimensional plane:

Parameterization: $\alpha_1 p_1 + \cdots + \alpha_k p_k$, where $\sum_{i=1}^{k} \alpha_i = 1$

Inner product:

For two vectors $u, v$ the inner product is defined: $u \cdot v = \sum_{i=1}^{d} u_i v_i$

Length:

The length of a vector $v$ is defined: $\|v\| = \sqrt{v^t \cdot v}$ – the square root of the inner product of the vector by itself (by its transposed).

Normalization:

Given a non-zero vector $v$, we can define the unit vector correlated to $v$: $\frac{v}{\|v\|}$ – normalizing the vector by its length. That means its length will always be 1, since: $\left\| \frac{v}{\|v\|} \right\| = \frac{1}{\|v\|} \cdot \|v\| = 1$

Distance between points:

The distance between two points $p, q$ is defined: $dist(p,q) = \|p - q\|$ - that is: $\sqrt{\sum (p_i - q_i)^2}$

Suppose we have a triabgle $p_1, p_2, p_3$ defined by the matrix:

$$\begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{pmatrix}$$

Then the determinant is: $D(p_1, p_2, p_3) = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$

This determinant is either $= 0, > 0, < 0$:

- $= 0$: all points are on the same line, i.e. the convex combination of the triangle is empty, so any point can be written as a convex combination of the other two points ($\alpha p_1 + (1-\alpha)p_2$. This is called a zero orientation.
- $> 0$: a "left turn" – a counterclockwise turn from $p_1 \to p_2, p_2 \to p_3$. This is called a positive orientation.
- $< 0$: a "right turn" – a clockwise turn from $p_1 \to p_2, p_2 \to p_3$. This is called a negative orientation.

The last two points are called the triangle orientation test. Note that the orientation is dependent on the order of the points.

**Question**:

Given 3 points in $\mathbb{R}^2$, how long does it take to compute the determinant? Constant!

## Affine Geometry

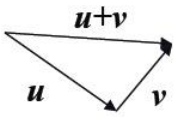
The elements of an affine geometry are:

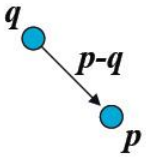
- Set of scalars (the real numbers) $S$.
- Set of points $P$.
- Set of vectors $V$.

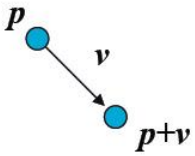
When subtracting / adding points we get a vector; When adding a point to a vector, we get another point.

Arithmetic:

- Scalar vector multiplication results with a vector: $S \cdot V \to V$
- Vector addition results with a vector: $V + V \to V$
- Point subtraction results with a vector: $P - P \to V$
- Point and vector addition results with a point: $P + V \to P$



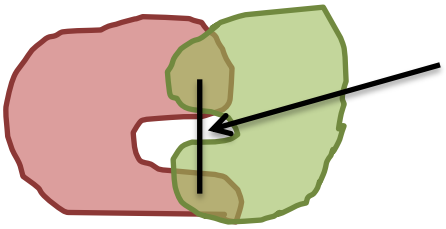




**Convex Set**:

A set such that for any two points in the set, drawing a line between them is entirely in the set as well.

Properties:

1. The intersection of two convex sets is convex: for $S_1, S_2$ convex sets, $S_1 \cap S_2$ is convex.
   That is not necessarily the case for concave sets, and the same for union:

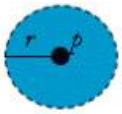


This part is outside the intersection

2. Let $L \subset E^k$ (a subset of points), then the convex hull of $L$ equals the intersection of all convex sets containing $L$. That is, if we take all possible sets that contain all points in $L$, the intersection of them will be the convex hull of $L$.
3. The convex hull of a finite set $S$ of points in $E^2$ is a simple polygon, and any hull vertex is a point in $S$.
   If the original data is spheres and not points, then the convex hull will not be a linear object, it will have curvatures, but for points it will be a polygon.

Neighborhood:

A neighborhood of a point $p$ is the set of points that their distance from $p$ is strictly less than some $r > 0$.



Interior point: a point $p \in S$ is an interior point if there is some $r > 0$ such that the neighborhood about $p$ of radius $r$ is contained within $S$.

Exterior point: if it is an interior point of $\bar{S}$ - the complement of $S$ (i.e. $\mathbb{R}^k - S$).

Boundary point: point that is neither interior nor exterior to $S$. Some of the neighborhood will be inside $S$ and some outside.

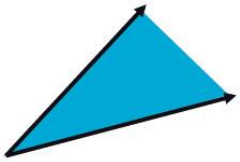
Boundary: the boundary of the set, which is defined as the set of all boundary points is denoted as $\delta$.

- If none of the boundary points are in $S$, it is an open set.
- if the complement of $S$ is open, then $S$ is a closed set.

Bounded set: A set that can be enclosed in a ball of finite radius.

Compact set: a closed and bounded set.

Note that a set can be closed but not compact (i.e. unbounded):



## Warm-up algorithm

The convex hull problem:

Given a set of $n$ points $P$ in the plane, find the representation of $P$'s convex hull.

The convex hull is a closed convex polygon, which is best represented as a counterclockwise enumeration of the vertices of the convex hull (the points that enclose it).

We're interested only in the extreme points, i.e. of all points on an edge of the boundary of the convex hull, we're interested only in the ends of that edge.

Let $p_1, \dots, p_n$ in the plane, then the way to identify the convex hull is by identifying the edges of it. In order to verify whether a line between two points $p, q$ is an edge of the convex hull, is by looking at the remaining $n - 2$ points – they all should be sitting on one side of the edge.

If the line is NOT part of the convex hull edges, some of the points will fall on one side and some on the other.

How can the side into which the $n - 2$ fall can be verified?

For any point $r$ in those $n - 2$ points we can form triplets $< p, q, r_i >$ which are triangles, and check their orientation – if they are all positive, they're on the same side; same as if they are all negative.

The complexity of this method will be $(n - 3) \cdot O(1)$ for a pair $p, q$. Since we have $\binom{n}{2}$ point pairs, the total complexity is $\Theta(n^3)$ (since $\binom{n}{2}$ is $\Theta(n^2)$).

This is a very inefficient algorithm.

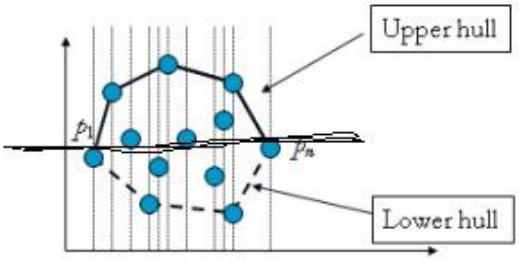
## Graham's Scan

The algorithm is an incremental algorithm.

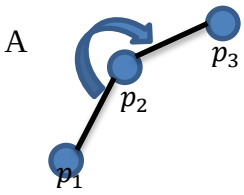
Observation: given $n$ points, those with the lowest/highest $x$ values and lowest/highest $y$ values are part of the convex hull. The reason is that for instance for the lowest $x$ value, all other points will be to the right of it. This gives us the bounding box of the convex hull.

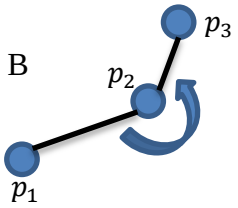
Graham's scan:

- identifies the points with the lowest and highest $x$.
- Identifies two segments – the upper and lower hulls (or: envelopes), where the union of both is the convex hull we're looking for:




Then to form the upper hull: Sort all points by their $x$ values, run through them and note which points going from $p_1$ to $p_n$ do not violate the convex hull property, i.e. are in a negative orientation.

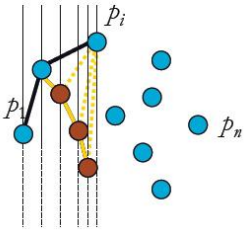



In A there is no violation, but in B there is a violation, and so $p_2$ is not part of the convex hull.

The case for the lower hull is symmetric.

- Let $p_1, \ldots, p_n$ denote the incremental order of points sorted by their $x$ values. It takes $n \lg n$ time to sort the points.
- Walk around the upper hull from left to right, observe that each consecutive triplet along the hull makes a right-hand turn, i.e. for $< p, q, r >$: $D(p, q, r) < 0$. If $D(p, q, r) > 0$, then $q$ is not in the convex hull boundry.




The red points in the diagram were first added, but then when got to $p_i$ they were eliminated as they were discovered as interior points.

For the lower hull, we need to run over the same order of points, only now looking for **left** turns instead of right ones.

Running time:

The running time is $O(n \lg n)$.

When processing point $p_i$, denote $D_i$ as the number of points you will drop that are before $p_i$ and now violate the convex hull condition. The number of times we will re-compute the triplet condition is:

$$\sum_{i=1}^{n}(D_i + 1) = n + \sum_{i=1}^{n} D_i$$

The $D_i$ part is for calculating the orientation condition for each of the points you drop (where each such calculation takes $O(1)$ time), and the extra 1 is for the last point you check that actually does satisfy the condition.

Now let's bound $\sum_{(i=1)}^{n} D_i$:

Each of the $n$ points is pushed onto the stack once, and once it is deleted we will never delete it again. Therefore $\sum_{(i=1)}^{n} D_i \leq n$, thus after sorting the total running time is $O(n)$. That would be the case also for the lower hull.

Therefore the total running time is $O(n \lg n)$ – much better than $O(n^3)$.

Furthermore, this is the lower bound for any convex hull calculation algorithm.

The reason for that is that we can't sort faster. Moreover, we can reduce the sorting problem to the convex hull problem, thus convex hull is bounded below by $O(n \lg n)$.

Optimality of Graham's algorithm

Graham's algorithm, although seems optimal, is in fact not.

The complexity might depend on how many points will be identified as the points that form the convex hull, i.e. an **output-sensitive** algorithm. For instance, if we have simple convex hull that is a box, that is only 4 points form it, then Graham's algorithm is not optimal.