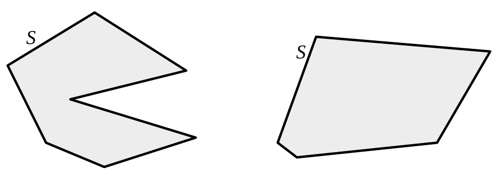
Computational Geometry: Convex Hulls

Outline

- Definitions
- Algorithms

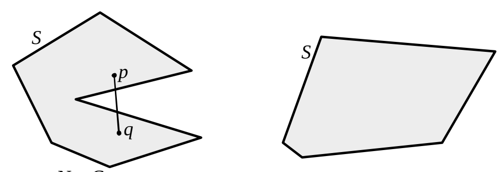
Definition I

A set S is <u>convex</u> if for any two points $p,q \in S$, the line segment $pq \subset S$.



Definition I

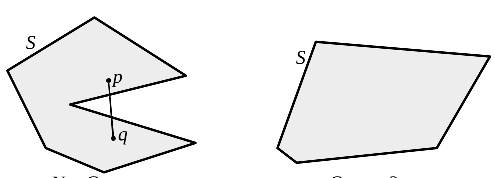
A set S is <u>convex</u> if for any two points $p,q \in S$, the line segment $pq \subset S$.



Not Convex

Definition I

A set S is <u>convex</u> if for any two points $p,q \in S$, the line segment $pq \subset S$.

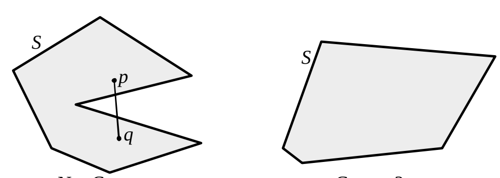


Not Convex

Convex?

Definition II

A set S is <u>convex</u> if it is the intersection of (possibly infinitely many) half-spaces.

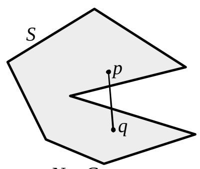


Not Convex

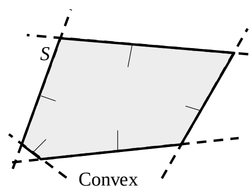
Convex?

Definition II

A set S is convex if it is the intersection of (possibly infinitely many) half-spaces.



Not Convex



Convex

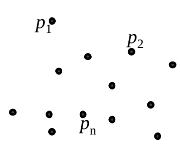
Outline

- Definitions
- Algorithms

Convex Hull

Definition:

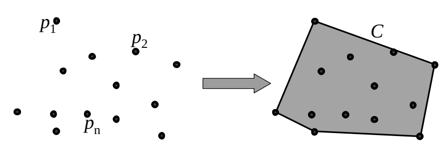
Given a finite set of points $P=\{p_1,\ldots,p_n\}$, the convex hull of P is the smallest convex set C such that $P\subset C$.



Convex Hull

Definition:

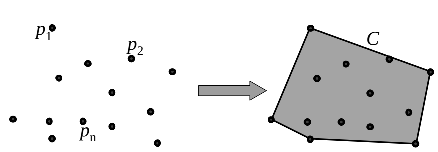
Given a finite set of points $P=\{p_1,\ldots,p_n\}$, the convex hull of P is the smallest convex set C such that $P\subset C$.



Examples

Two Dimensions:

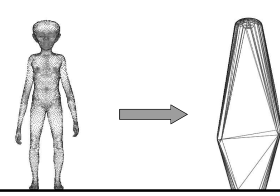
The convex hull of $P=\{p_1,\ldots,p_n\}$ is a set of line segments with endpoints in P.



Examples

Three Dimensions:

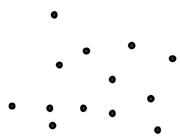
The convex hull of $P=\{p_1,\ldots,p_n\}$ is a triangle mesh with vertices in P.



Algorithms

Brute Force (2D):

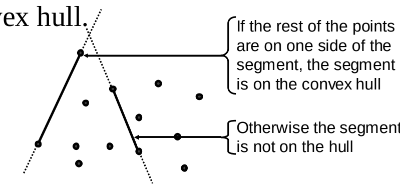
Given a set of points *P*, test each line segment to see if it makes up an edge of the convex hull.



Algorithms

Brute Force (2D):

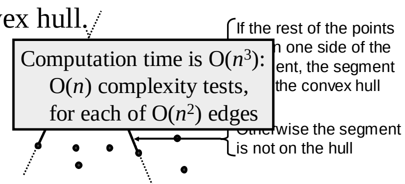
Given a set of points *P*, test each line segment to see if it makes up an edge of the convex hull.



Algorithms

Brute Force (2D):

Given a set of points *P*, test each line segment to see if it makes up an edge of the convex hull.



Computation time is O(n^3): O(n) complexity tests, for each of O(n^2) edges

Algorithms

Brute Force (2D):

Given a set of points *P*, test each line segment to see if it makes up an edge of the convex hull.

Computation time is O(n^3): O(n) complexity tests, for each of O(n^2) edges

In a *d*-dimensional space, the complexity is O(n^{d+1}).

Algorithms

There are many algorithms for computing the convex hull:

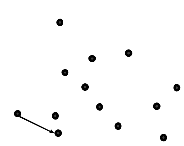
- Brute Force: O(n^3)
- Gift Wrapping
- Quickhull
- Divide and Conquer

Gift Wrapping

Key Idea:

Iteratively growing edges of the convex hull, we want to turn as little as possible.

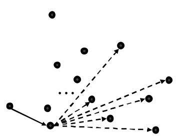
- Given a directed edge on the hull…



Gift Wrapping

Key Idea:

Iteratively growing edges of the convex hull, we want to turn as little as possible.

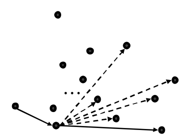


- Given a directed edge on the hull…
- Of all the vertices the next edge can can connect to…

Gift Wrapping

Key Idea:

Iteratively growing edges of the convex hull, we want to turn as little as possible.

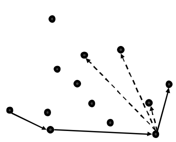


- Given a directed edge on the hull…
- Of all the vertices the next edge can can connect to…
- Choose the one which turns least.

Gift Wrapping

Key Idea:

Iteratively growing edges of the convex hull, we want to turn as little as possible.

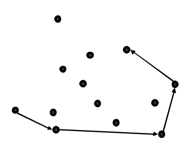


- Given a directed edge on the hull…
- Of all the vertices the next edge can can connect to…
- Choose the one which turns least.
- Repeat

Gift Wrapping

Key Idea:

Iteratively growing edges of the convex hull, we want to turn as little as possible.

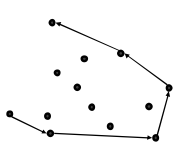


- Given a directed edge on the hull…
- Of all the vertices the next edge can can connect to…
- Choose the one which turns least.
- Repeat

Gift Wrapping

Key Idea:

Iteratively growing edges of the convex hull, we want to turn as little as possible.

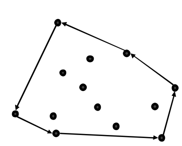


- Given a directed edge on the hull…
- Of all the vertices the next edge can can connect to…
- Choose the one which turns least.
- Repeat

Gift Wrapping

Key Idea:

Iteratively growing edges of the convex hull, we want to turn as little as possible.



- Given a directed edge on the hull…
- Of all the vertices the next edge can can connect to…
- Choose the one which turns least.
- Repeat

Algorithms

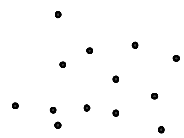
There are many algorithms for computing the convex hull:

- Brute Force: $O(n^3)$
- **Gift Wrapping**: $O(n^2)$
- Quickhull
- Divide and Conquer

Quickhull

Key Idea:

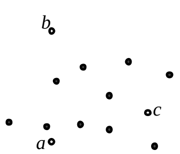
For all $a,b,c \in P$, the points contained in $\Delta abc \cap P$ cannot be on the convex hull.



Quickhull

Key Idea:

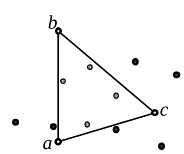
For all $a,b,c \in P$, the points contained in $\Delta abc \cap P$ cannot be on the convex hull.



Quickhull

Key Idea:

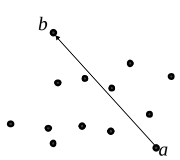
For all $a,b,c \in P$, the points contained in $\Delta abc \cap P$ cannot be on the convex hull.



Quickhull

Key Idea:

For all $a,b,c \in P$, the points contained in $\Delta abc \cap P$ cannot be on the convex hull.

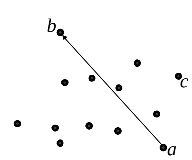


- Given a line segment $\overline{ab}$ …

Quickhull

Key Idea:

For all $a,b,c \in P$, the points contained in $\Delta abc \cap P$ cannot be on the convex hull.

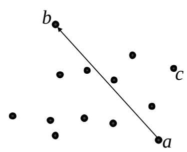


- Given a line segment $\overline{ab}$ …
- Find the point c, rightmost from $\overline{ab}$ …

Quickhull

Key Idea:

For all $a,b,c \in P$, the points contained in $\Delta abc \cap P$ cannot be on the convex hull.

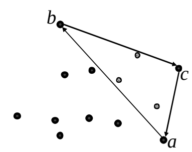


- Given a line segment $\overline{ab}$ …
- Find the point *c*, rightmost from $\overline{ab}$ …
- If *c* doesn't exist, return $\overline{ab}$ …

Quickhull

Key Idea:

For all $a,b,c \in P$, the points contained in $\Delta abc \cap P$ cannot be on the convex hull.

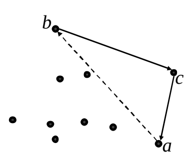


- Given a line segment $\overline{ab}$ …
- Find the point *c*, rightmost from $\overline{ab}$ …
- If *c* doesn't exist, return $\overline{ab}$ …
- Discard the points in Δabc …

Quickhull

Key Idea:

For all $a,b,c \in P$, the points contained in $\Delta abc \cap P$ cannot be on the convex hull.

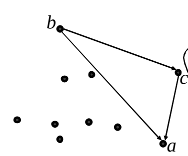


- Given a line segment $\overline{ab}$ …
- Find the point *c*, rightmost from $\overline{ab}$ …
- If *c* doesn't exist, return $\overline{ab}$ …
- Discard the points in Δabc …
- Repeat for left of $\overline{bc}$ and $\overline{ca}$ …

Quickhull

Key Idea:

For all $a,b,c \in P$, the points contained in $\Delta abc \cap P$ cannot be on the convex hull.

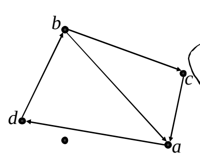


- Given a line segment $\overline{ab}$ …
- Find the point *c*, rightmost from $\overline{ab}$ …
- If *c* doesn't exist, return $\overline{ab}$ …
- Discard the points in Δabc …
- Repeat for left of $\overline{bc}$ and $\overline{ca}$ …
- Repeat for left of $\overline{ba}$ …

Quickhull

Key Idea:

For all $a,b,c \in P$, the points contained in $\Delta abc \cap P$ cannot be on the convex hull.

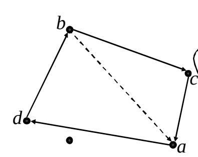


- Given a line segment $\overline{ab}$ …
- Find the point *c*, rightmost from $\overline{ab}$ …
- If *c* doesn't exist, return $\overline{ab}$ …
- Discard the points in Δabc …
- Repeat for left of $\overline{bc}$ and $\overline{ca}$ …
- Repeat for left of $\overline{ba}$ …

Quickhull

Key Idea:

For all $a,b,c \in P$, the points contained in $\Delta abc \cap P$ cannot be on the convex hull.

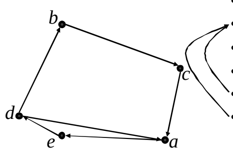


- Given a line segment $\overline{ab}$ …
- Find the point *c*, rightmost from $\overline{ab}$ …
- If *c* doesn't exist, return $\overline{ab}$ …
- Discard the points in Δabc …
- Repeat for left of $\overline{bc}$ and $\overline{ca}$ …
- Repeat for left *ba* …

Quickhull

Key Idea:

For all $a,b,c \in P$, the points contained in $\Delta abc \cap P$ cannot be on the convex hull.

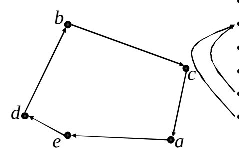


- Given a line segment $\overline{ab}$ …
- Find the point c, rightmost from $\overline{ab}$ …
- If c doesn't exist, return $\overline{ab}$ …
- Discard the points in Δabc …
- Repeat for left of $\overline{bc}$ and $\overline{ca}$ …
- Repeat for left ba …

Quickhull

Key Idea:

For all $a,b,c \in P$, the points contained in $\Delta abc \cap P$ cannot be on the convex hull.



- Given a line segment $\overline{ab}$ …
- Find the point c, rightmost from $\overline{ab}$ …
- If c doesn't exist, return $\overline{ab}$ …
- Discard the points in Δabc …
- Repeat for left of $\overline{bc}$ and $\overline{ca}$ …

- Repeat for left of $\overline{ba}$ …

Algorithms

There are many algorithms for computing the convex hull:

- Brute Force: $O(n^3)$
- Gift Wrapping: $O(n^2)$
- **Quickhull**: $O(n\log n)$ – $O(n^2)$
- Divide and Conquer

Divide and Conquer

Key Idea:

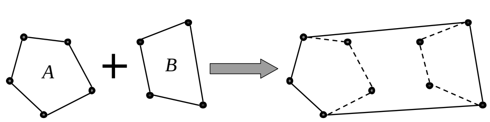
Finding the convex hull of small sets is easier than finding the hull of large ones.

Divide and Conquer

Key Idea:

Finding the convex hull of small sets is easier than finding the hull of large ones.

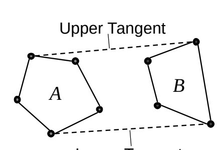
All we need is a fast way to merge hulls.



Divide and Conquer

Merging Hulls:

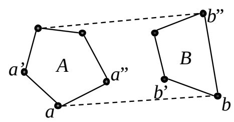
Need to find the tangents joining the hulls.



Divide and Conquer

Observation:

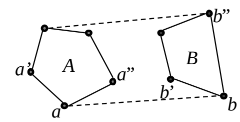
The edge $\overline{ab}$ is a tangent if the two points about *a* and the two points about *b* are on the same side of $\overline{ab}$.



Divide and Conquer

Proof:

The edge $\overline{ab}$ is a tangent if the points on both hulls are all on one side of it.

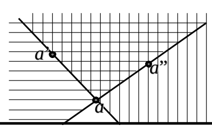


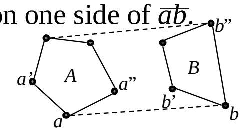
Divide and Conquer

Proof:

The edge $\overline{ab}$ is a tangent if the points on both hulls are all on one side of it.

If *a*’ and *a*” are on the same side of $\overline{ab}$, then all of *A* must be on one side of $\overline{ab}$.

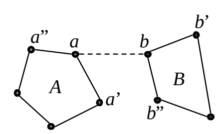




Divide and Conquer

Tangent Algorithm:

– Find an edge $\overline{ab}$ between *A* and *B* that does not intersect the two hulls.

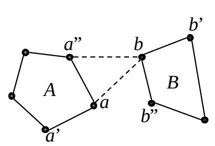


Divide and Conquer

Tangent Algorithm:

– Find an edge $\overline{ab}$ between *A* and *B* that does not intersect the two hulls.

– While *a*’ and *a*” are not to the left of $\overline{ab}$, rotate *a* clock-wise.



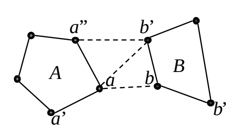
Divide and Conquer

Tangent Algorithm:

– Find an edge $\overline{ab}$ between *A* and *B* that does not intersect the two hulls.

– While *a*’ and *a*” are not to the left of $\overline{ab}$, rotate *a* clock-wise.

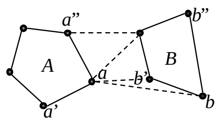
– While *b*’ and *b*” are not to the left of $\overline{ab}$, rotate *b* counter-clock-wise.



Divide and Conquer

Tangent Algorithm:

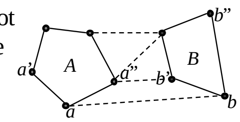
- Find an edge $\overline{ab}$ between *A* and *B* that does not intersect the two hulls.
- While *a*’ and *a*” are not to the left of $\overline{ab}$, rotate *a* clock-wise.
- While *b*’ and *b*” are not to the left of $\overline{ab}$, rotate *b* counter-clock-wise.



Divide and Conquer

Tangent Algorithm:

- Find an edge $\overline{ab}$ between *A* and *B* that does not intersect the two hulls.
- While *a*’ and *a*” are not to the left of $\overline{ab}$, rotate *a* clock-wise.
- While *b*’ and *b*” are not to the left of $\overline{ab}$, rotate *b* counter-clock-wise.
- Repeat



Algorithms

There are many algorithms for computing the convex hull:

- Brute Force: O(n^3)
- Gift Wrapping: O(n^2)
- Quickhull: O($n\log n$) – O(n^2)
- **Divide and Conquer**: O($n\log n$)