Convex hulls

Samuel Kob

Seminar "Selected Topics in Efficient Algorithms"
Technical University of Munich

July 19, 2024

Table of Contents

Table of Contents

Computational geometry

Computational geometry is devoted to the study of algorithms which can be stated in terms of geometry

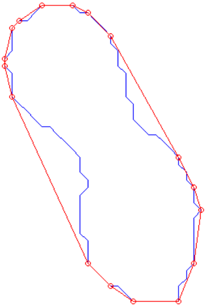


Figure 1: Convex hull 2D

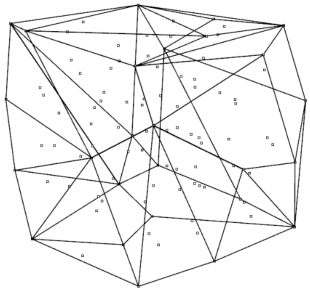


Figure 2: Convex hull 3D

Computational geometry

One of the oldest sections in computer science.
Divided in 2 types of problems: static and dynamic problems
Important applications include

- Computer graphics
- Computer vision
- Computer aided design (CAD)
- Mathematical visualization
- Robotics
- Integrated circuit design

Convex set

First we have to understand what a convex set is. A subset $S \subseteq P$ is convex, if $\forall x, y \in S$, $\overline{xy}$ is contained entirely in S too.

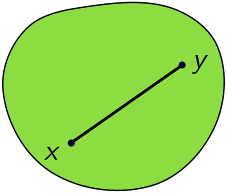


Figure 3: Convex set

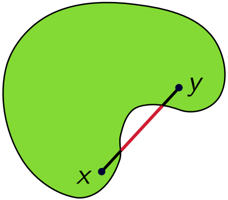


Figure 4: Non convex set

Convex hulls

A convex hull of a set P is therefore the smallest convex set CH that contains it.

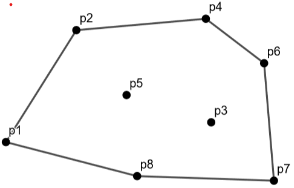


Figure 5: Convex hull with enumerations

$$P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}$$
$$CH = (p_1, p_2, p_4, p_6, p_7, p_8)$$

Table of Contents

Naive approach

First of all: What do we expect from our algorithm?

Input: $P = \{p_1, p_2 \ldots p_n\}$
Output: ordered list of vertices representing convex hull of P in clockwise order, e.g. $CH = (p_1, p_2, p_4, \ldots, p_8)$

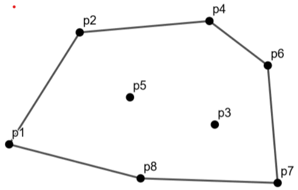


Figure 6: Convex hull

Argument 1

Let $p, q \in P$ and (p, q) edge of $CH(P)$. Then $\forall r \in P \setminus \{p, q\}$, r lies to the right of (p, q).

Idea of using this argument to compute $CH(P)$:

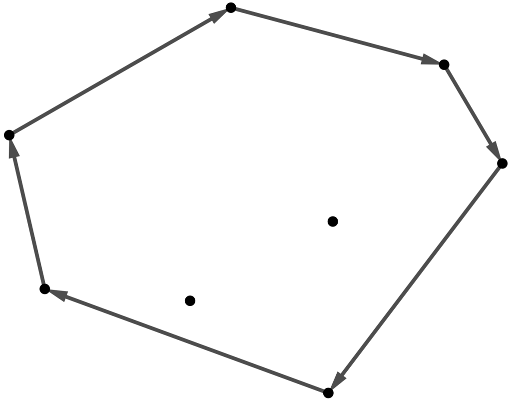
1. Loop through all pairs of points in P
2. Check if Argument 1 applies
3. Construct $CH(P)$ out of valid pairs

Let's construct an algorithm for the idea:

$E = \emptyset$
$Pairs = (p, q) \in P \times P$ with $p \neq q$

Loop trough all $(p, q) \in Pairs$ and check $\forall r \in P \setminus \{p, q\}$, that r lies on the right of (p, q).
If $\exists r$ that not lies on the right, (p, q) can't be part of $CH(P)$.
Else (p, q) is part of the convex hull and add (p, q) to E.

Construct $CH(P)$ out of E



Two steps are still not entirely clear

$E = \emptyset$
$Pairs = (p, q) \in P \times P$ with $p \neq q$

Loop trough all $(p, q) \in Pairs$ and check $\forall r \in P \setminus \{p, q\}$, that r lies on the right of (p, q).
If $\exists r$ that not lies on the right, (p, q) can't be part of $CH(P)$.
Else (p, q) is part of the convex hull and add (p, q) to E.

Construct $CH(P)$ out of E

How do we know if r lies to the right of (p, q)?

$p = (x_p, y_p), q = (x_q, y_q)$

$\overrightarrow{pq} = (x_q - x_p, y_q - y_p)$
$\overrightarrow{pr} = (x_r - x_p, y_r - y_p)$

Calculate the determinant of $(\overrightarrow{pq}, \overrightarrow{pr})$.

$$s_{pqr} = det \begin{pmatrix} x_q - x_p & x_r - x_p \\ y_q - y_p & y_r - y_p \end{pmatrix}$$

The sign of the determinant s_{pqr} tells us if r is to the right of (p, q)

- $s_{pqr} < 0$: r lies to the right
- $s_{pqr} > 0$: r lies to the left
- $s_{pqr} = 0$: r lies on the line of (p, q)

This can determine, where exactly the point is located. The time complexity of this check is $O(1)$, because it uses just a constant amount of arithmetical operations.

One step is still not entirely clear

$E = \emptyset$
$Pairs = (p, q) \in P \times P$ with $p \neq q$

Loop trough all $(p, q) \in Pairs$ and check $\forall r \in P \setminus \{p, q\}$, that r lies on the right of (p, q).
If $\exists r$ that not lies on the right, (p, q) can't be part of $CH(P)$.
Else (p, q) is part of the convex hull and add (p, q) to E.

Construct $CH(P)$ out of E

How do we construct $CH(P)$ out of E?

We know:

- $CH(P)$ should be a list of vertices, sorted in clockwise order
- E contains directed edges of the convex hull, not sorted

Directed edges means origin and destination of edge
All edges have all other points on their right
$CH(P)$ is list of origins and destination of edges in E

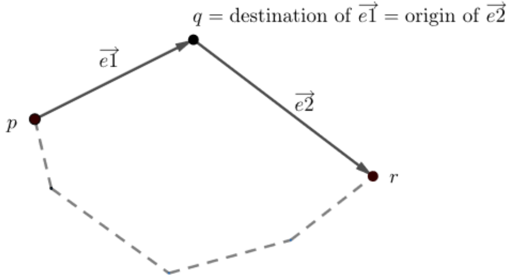


Figure 7: Directed edges in $CH(P)$

Let CH be an empty list
Remove any $(p, q) \in E$, append p to CH
Search in $(q, r) \in E$, remove (q, r) from E and append q to CH
Repeat the previous step until $r = p$

Problem: What do we do if more than 2 points lie on a straight line?

We check if point r lies on $\overline{pq}$.

If r lies on $\overline{pq}$, continue with next point.
Else $\overline{pq}$ cannot be part of $CH(P)$.

The time complexity of this naive convex hull computation is $O(n^3)$.

$|Pairs|$ is $n^2 - n$. For each of this pair we check another $n - 2$ pairs if they lie on the right side. This results in $O(n^3)$

The step to check if a point lies on the right side takes like seen before $O(1)$.
The computation of $CH(P)$ in the last step takes $O(n^2)$.

$$O(n^3) + O(1) + O(n^2) = O(n^3)$$

Incremental algorithm

The naive approach is slow, not robust and handles things in weird ways.
We try a proper algorithm technique:
Incremental algorithms

Short explanation: Updating existing solution after adding an item of input

The idea:

1. Input: Sequence of points, sorted by x-coordinate
2. Add one by one point to a list that holds the convex line/hull until that point
3. Compute first upper hull, then lower hull
4. Connect the two hulls

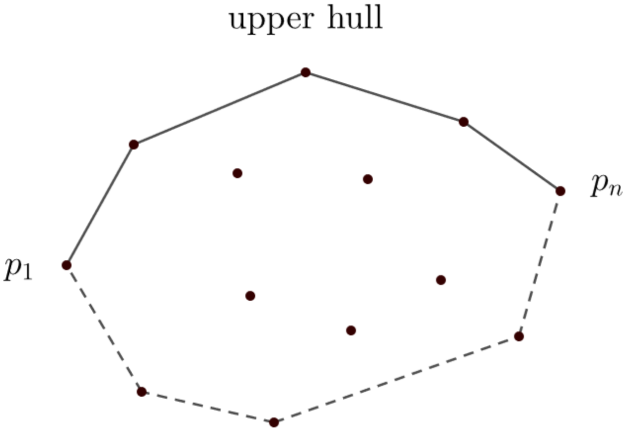


Figure 8: Representation of an upper hull

Computing upper hull

Sorting list of points by x-coordinate P, getting $S = (p_1, \ldots, p_n)$
Given a valid upper hull of points $p_1 \ldots p_{i-1}$, we compute the hull for $p_1 \ldots p_i$.

How do we compute/check the new hull after adding a point?

Argument 2

Walking a convex hull in clockwise order, there are only right turns at every vertex.

By knowing argument 2, we then can check if after adding p_i to the convex hull until p_{i-1}, the last 3 points of the hull are still making a right turn. If they do so, p_i is valid in the hull and we continue.

What if after adding p_i, the last 3 points do not make a right turn?
We remove the middle point and check again.

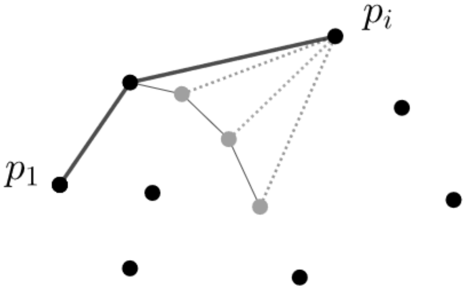


Figure 9: Removing middle point until right turn is made

Computing the upper hull

Let CH_{upper} be a list which contains the upper vertices in left-to-right order.
Appending p_i to CH_{upper} is valid, because p_i is the most right point.
Now check if the last three points, p_{i-2}, p_{i-1} and p_i are making a right turn.

If so, continue with adding p_{i+1}, else repeat removing the middle point until last 3 points make a right turn.

Computing the lower hull is exactly the same, except that CH_{lower} starts at p_n and works it way from right to left.

The algorithm

Sort points from P by x-coordinate, getting $(p_1, \ldots, p_n)$

$CH_{upper} \leftarrow [p_1, p_2]$

Loop i = 3 to n

Append p_i to CH_{upper}

while $|CH_{upper}| > 2$ **and** last 3 points not make a right turn

remove middle point

And again some issues to resolve:

1. 2 points with same x-coordinate
2. Collinear points should not appear in convex hull, only endpoints

1. Sort points first by ascending x-coordinate, then by ascending y-coordinate.

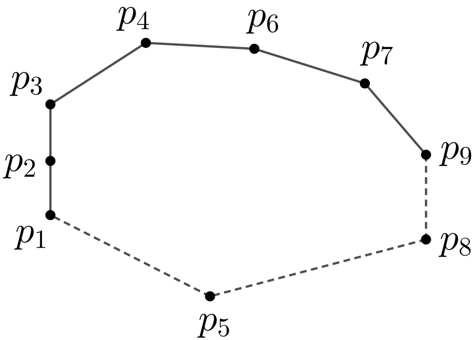


Figure 10: Sorted points with same x-coordinates

2. If 3 points are collinear $\Rightarrow$ no right turn is made (check fails and middle gets removed)

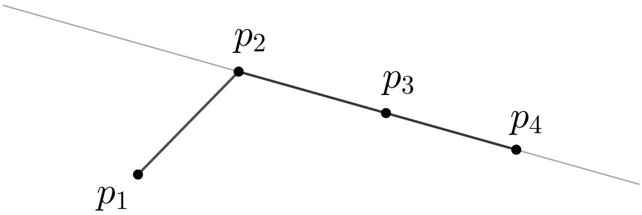


Figure 11: Collinear points

p_3 gets removed, no right turn is made

The time complexity of IncrementalConvexHull is $O(n \log n)$
Sorting points: $O(n \log n)$
for-loop bounded by n

Table of Contents

Introduction

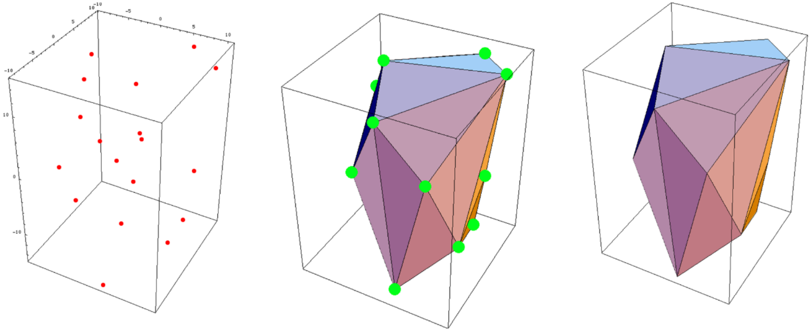


Figure 12: Convex hull in 3 dimensions

Convex hulls also exist in 3 dimensions.
Used for instance in collision detection in computer animations.

But why?
We want to know if 2 objects, Obj_1 and Obj_2 intersect. Most of the time, they do not $\Rightarrow$ faster to just approximate Obj_1 and Obj_2 and if convex hulls of them intersect, make more complex test with the exact object.

Computation

Again incremental algorithm.
Let $P = \{p_1, \ldots, p_n\}$

1. We choose 4 points in P.
   They should not lie on a common plane $\Rightarrow$ convex hull is tetrahedron
2. Random permutation for $p_5, \ldots p_n$. For $r >= 1$, $P_r := \{p_1, \ldots, p_r\}$.
3. For $r = 5$ to n
   $CH(P_r) =$ insert p_r into $CH(P_{r-1})$

To store $CH(P_r)$ we use a DCEL (doubly-connected edge list). The vertices are 3 dimensional.

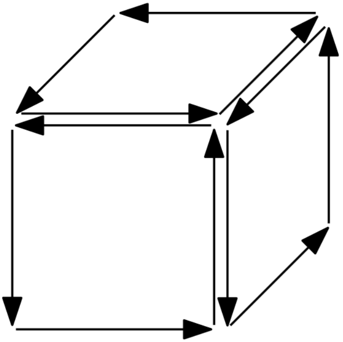


Figure 13: DCEL-representation

But how to insert p_r into $CH(P_{r-1})$?

We want to achieve:

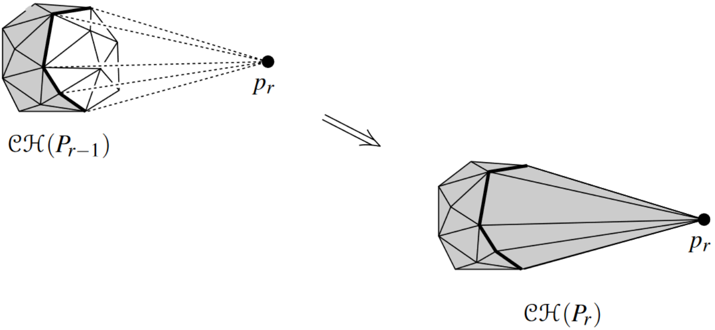


Figure 14: Adding point to $CH(P_{r-1})$

But how to insert p_r into $CH(P_{r-1})$?

2 cases:

- p_r lies already in $CH(P_{r-1})$. Then $CH(P_r) = CH(P_{r-1})$
- p_r lies outside of $CH(P_{r-1})$.

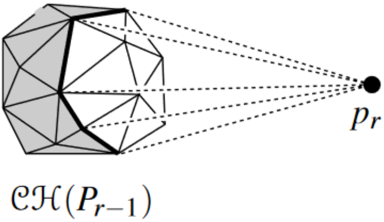
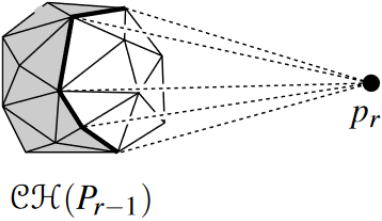


Figure 15: Horizon of visible and non visible points

We have visible regions (white) and invisible regions (gray) looking from p_r onto $CH(P_{r-1})$.



The 2 regions are seperated by edges from $CH(P_{r-1})$, we call this curve *horizon* of p_r on $CH(P_{r-1})$.

The horizon sets the boundary for faces we keep and faces that have to be removed in order to add p_r.

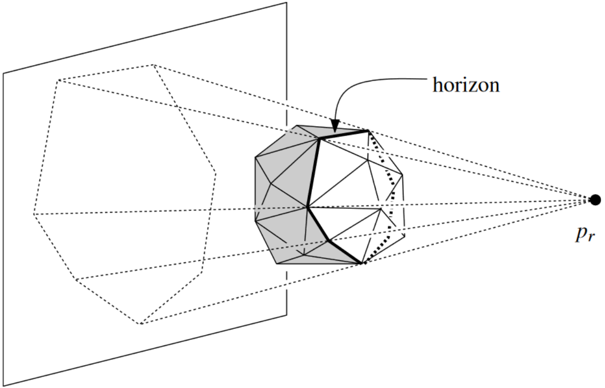


Figure 16: Projected horizon

Idea to insert p_r into $CH(p_{r-1})$:

- Keep invisible faces
- Remove visible faces
- Add new faces

Would be easy if we know all visible faces from $CH(P_r) \Rightarrow$ remove visible faces from the DCEL and compute new ones connecting p_r to *horizon*.
How do we find visible faces?

Naive:
Test every face for every point $\Rightarrow O(n^2)$

Conflict graph:
We maintain some additional information for each face $f \in CH(P_{r-1})$ and p_t with $t > r$

- $P_{conflict}(f) \subseteq \{p_r, \ldots, p_n\}$: all points that can see face f
  $\Rightarrow$ are in conflict because not both p and f can be part of the convex hull at the same time
- $F_{conflict}(p_t) \subseteq$ planes in $CH(P_r)$: contains all faces f that p_t can see

Bipartite conflict graph to store $P_{conflict}(f)$ and $F_{conflict}(p_t)$. One set is P_r (one node for each point not inserted yet) and the other is all faces f of the current convex hull.

A node $p_t \in P$ is connected to a face f of $CH(P_r)$ if $r < t$ and p_t can see f.

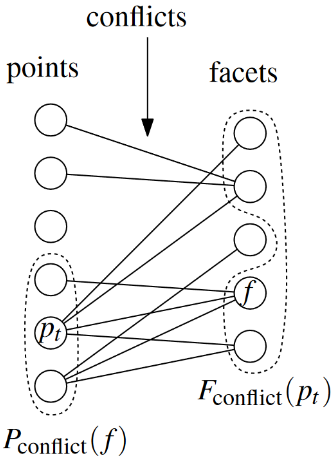


Figure 17: Conflict graph

Insertion of p_r into $CH(P_{r-1})$

We look up in $F_{conflict}(p_r)$, get the faces visible from p_r, and replace them by new faces connecting p_r to the horizon.

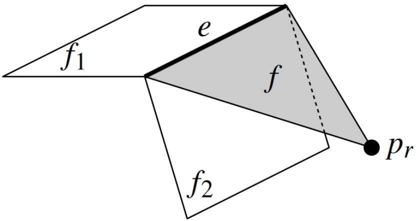


Figure 18: New faces to connect p_r

How to maintain correctness of conflict graph

Initialize conflict graph for $CH(P_4) \Rightarrow$ loop through points and determine it.

Update conflict graph after addition of p_r:

- Discard neighbors of p_r (visible faces from p_r)
- Discard p_r
- Insert nodes for new faces, which connect p_r to the *horizon*
- Find conflicts for all newly created faces

Find conflicts for all newly created faces

- p_t must have seen f_1 and f_2 in $CH(P_{r-1})$
- Therefore check all points in conflict lists of f_1 and f_2
- Add p_t if it sees face f and edge e

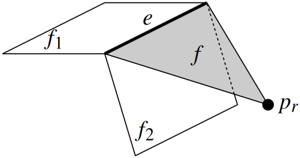


Figure 19: Adding face f

What we know now:
Rough idea of the algorithm
How to insert p_r into $CH(p_{r-1})$

Let's see the full algorithm to compute the convex hull

Input. A set P of n points in three-space.
Output. The convex hull $\mathcal{CH}(P)$ of P.

Find four points p_1, p_2, p_3, p_4 in P that form a tetrahedron.
$\mathcal{C} \leftarrow \mathcal{CH}(\{p_1, p_2, p_3, p_4\})$
Compute a random permutation $p_5, p_6, \ldots, p_n$ of the remaining points.
Initialize the conflict graph $\mathcal{G}$ with all visible pairs (p_t, f), where f is a facet of $\mathcal{C}$ and $t > 4$.
for $r \leftarrow 5$ **to** n
  do (∗ Insert p_r into $\mathcal{C}$: ∗)
    if $F_{\text{conflict}}(p_r)$ is not empty (∗ that is, p_r lies outside $\mathcal{C}$ ∗)
      then Delete all facets in $F_{\text{conflict}}(p_r)$ from $\mathcal{C}$.
        Walk along the boundary of the visible region of p_r (which consists exactly of the facets in $F_{\text{conflict}}(p_r)$) and create a list $\mathcal{L}$ of horizon edges in order.

for all $e \in \mathcal{L}$

  do Connect e to p_r by creating a triangular facet f.

    if f is coplanar with its neighbor facet f' along e

      then Merge f and f' into one facet, whose conflict list is the same as that of f'.

      else (∗ Determine conflicts for f: ∗)

        Create a node for f in $\mathcal{G}$.

        Let f_1 and f_2 be the facets incident to e in the old convex hull.

        $P(e) \leftarrow P_{\text{conflict}}(f_1) \cup P_{\text{conflict}}(f_2)$

        for all points $p \in P(e)$

          do If f is visible from p, add (p, f) to $\mathcal{G}$.

Delete the node corresponding to p_r and the nodes corresponding to the facets in $F_{\text{conflict}}(p_r)$ from $\mathcal{G}$, together with their incident arcs.

return $\mathcal{C}$

Conclusion

What we saw in the last hour:

- Computational geometry
- Convex hulls in 2D:
  Naive approach, incremental algorithm
- Convex hulls in 3D

Any questions?

References

Berg, Mark de et al. *Computational Geometry*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000.

Hernando, Roger. *Convex hull algorithms in 3D*. https://dccg.upc.edu/people/vera/wp-content/uploads/2014/11/GA2014-ConvexHulls3D-Roger-Hernando.pdf. Accessed: 2024–03-07.

Wenk, Carola. *3D Convex Hulls*. https://www.cs.tulane.edu/~carola/teaching/cmps6640/spring16/slides/3D_convexHull.pdf. Accessed: 2024–03-07. 2016.

References

Figure 1 https://www.crisluengo.net/archives/408/?p=405
Figure 2 https://www.genysis.cloud/documentation/convex-hull
Figure 3 https://en.wikipedia.org/wiki/Convex_set
Figure 4 https://en.wikipedia.org/wiki/Convex_set
Figure 5 Created by Samuel Kob
Figure 6 Created by Samuel Kob
Figure 7 Created by Samuel Kob
Figure 8 Created by Samuel Kob
Figure 9 Created by Samuel Kob
Figure 10 Created by Samuel Kob

References

Figure 11 Created by Samuel Kob
Figure 12 Wenk, Carola. *3D Convex Hulls*
Figure 13 Berg, Mark de et al. *Computational Geometry*
Figure 14 Berg, Mark de et al. *Computational Geometry*
Figure 15 Berg, Mark de et al. *Computational Geometry*
Figure 16 Berg, Mark de et al. *Computational Geometry*
Figure 17 Berg, Mark de et al. *Computational Geometry*
Figure 18 Berg, Mark de et al. *Computational Geometry*
Figure 19 Berg, Mark de et al. *Computational Geometry*