Computational Geometry

8. Convex hull in 3d

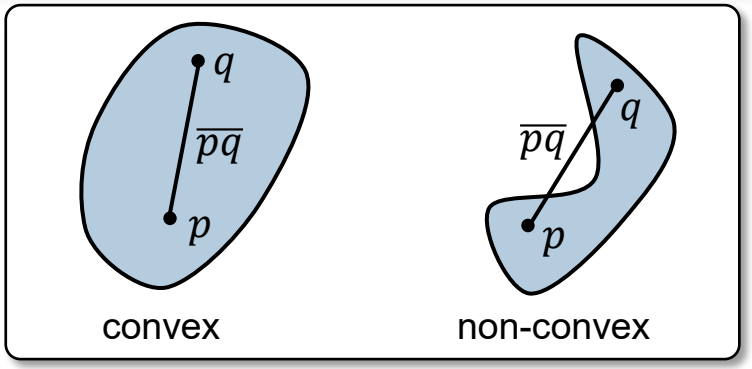
8.1 Complexity in 3d

Definition (Convex Set)

A set $P \in \mathbb{R}^d$ is called **convex**, iff for all points $p, q \in P$ the connecting line segment

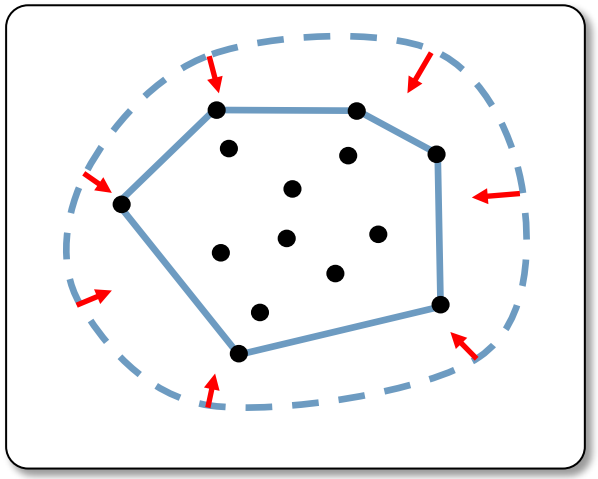
$$\overline{pq} = \{x | x = \lambda p + (1-\lambda)q, \lambda \in [0,1]\}$$

is contained in P, i.e. $\overline{pq} \subseteq P$.



Definition (Convex Hull)

The **convex hull** $\mathcal{CH}(P)$ of a set $P \subseteq \mathbb{R}^d$ is the smallest convex set containing P, i.e. it is the intersection of all convex sets containing P.

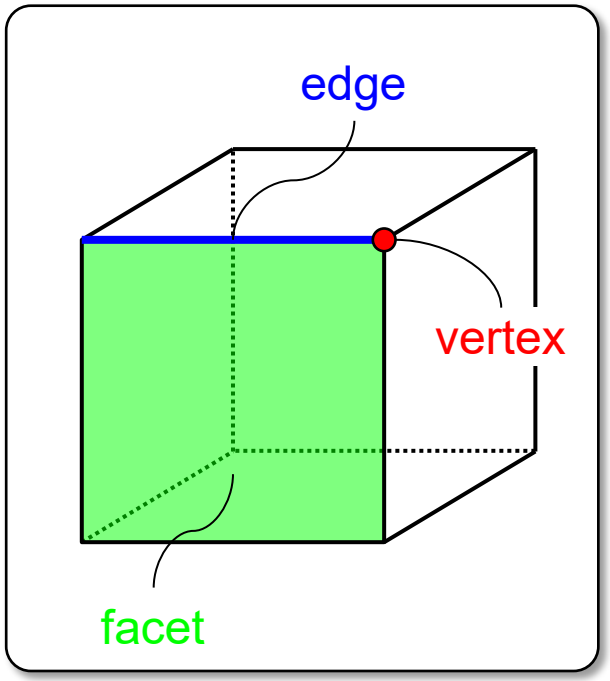


8.1 Complexity in 3d

Alternative Definition (Convex Hull)

The **convex hull** $\mathcal{CH}(P)$ of set $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$ is defined as the set of all convex combinations

$$\mathcal{CH}(P) = \left\{ \sum_{i=1}^{n} \lambda_i \, p_i \mid \lambda_i \geq 0, \sum_{i=1}^{n} \lambda_i = 1 \right\}.$$



- In 3d the convex hull is a polytope, i.e. an object bounded by polygons consisting of vertices, edges and facets.

8.1 Complexity in 3d

Theorem 1

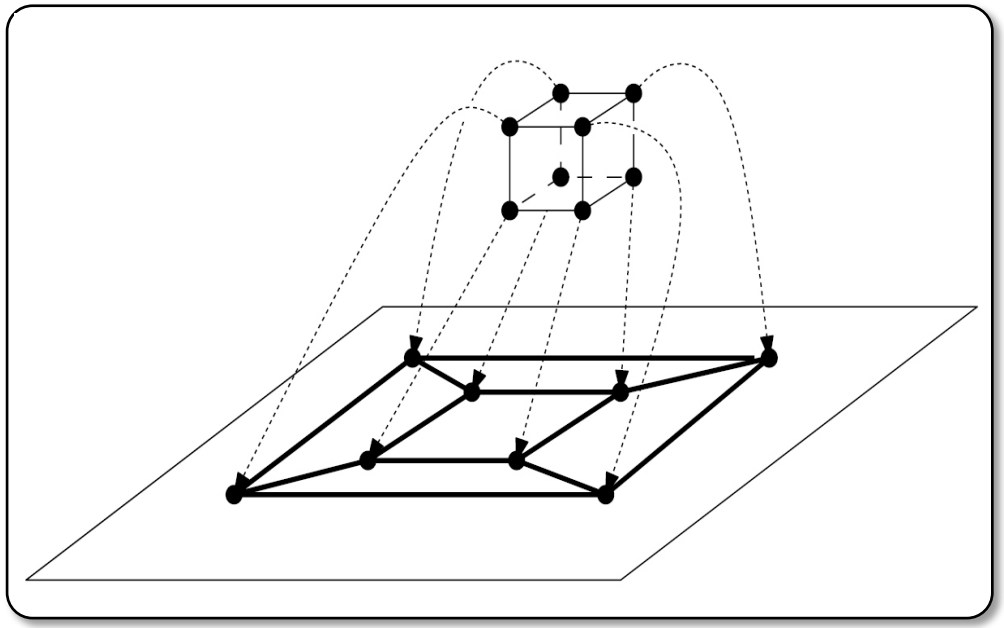
A convex polytope Q with n vertices contains at most

- $n_e \leq 3n - 6$ edges and
- $n_f \leq 2n - 4$ facets.

Proof

- Q corresponds to a planar graph, where Euler's formula holds

$$n - n_e + n_f = 2.$$



8.1 Complexity in 3d

- Because each edge corresponds to two facets and each facets has at least three edges, we get

$$2n_e \geq 3n_f.$$

- Substituting this into the Euler formula yields

$$n + n_f - 2 \geq 3n_f/2 \quad \text{and} \quad 2 - n + n_e \leq 2n_e/3.$$

- This simplifies to

$$n_f \leq 2n - 4 \quad \text{and} \quad n_e \leq 3n - 6.$$

□

- ▶ The complexity of the convex hull is $O(n)$.

8.1 Complexity in 3d

Corollary 2

The expected **valence** (number of edges) per vertex $p \in Q$ is at most 6 (average over all vertices).

Proof

- Because each edge connects exactly two vertices, the sum of valances is bounded by: $\sum val = 2n_e \leq 6n - 12.$ □

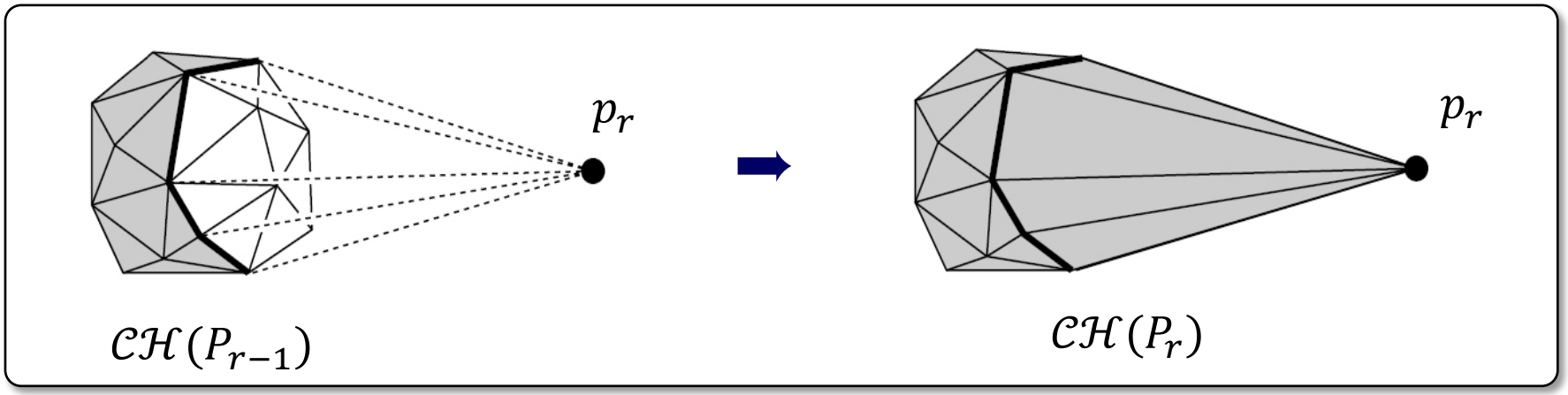
- Theorem 1 and Corollary 2 are valid for arbitrary polyhedrons of genus zero.
  - For higher genus (number of holes) or multiple connected components the Euler-formula can be adapted.

8.2 Algorithm

- To compute the convex hull in 3d, we will use a randomized incremental algorithm.
  - Starting from a tetrahedron of four vertices, add one vertex after the other in a random order to the convex hull

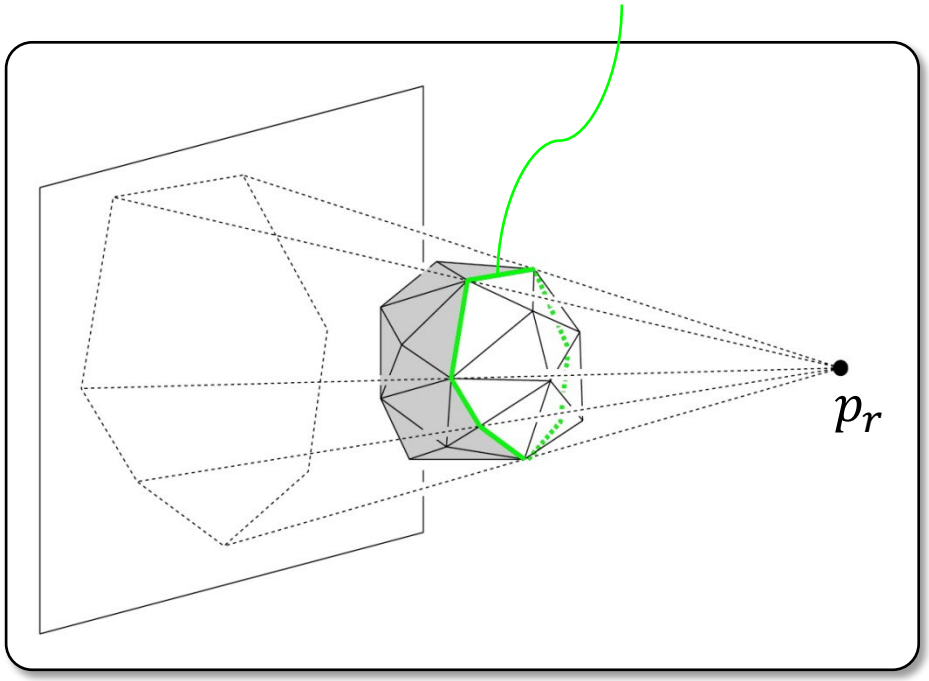
$$P_r = \{p_1, \dots, p_r\}, P_n = P.$$

  - If $p_r \in \mathcal{CH}(P_{r-1})$ nothing needs to be done: $\mathcal{CH}(P_{r-1}) = \mathcal{CH}(P_r)$.
  - Otherwise:



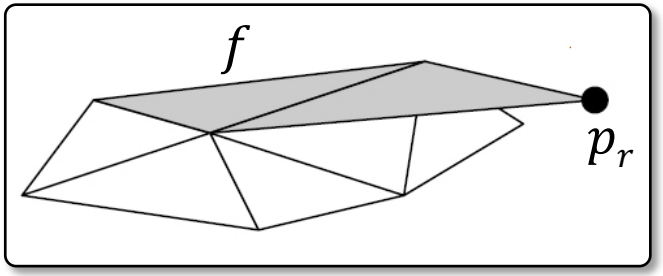
8.2 Algorithm

- To add p_r to P_{r-1}, all facets must be removed, that are visible from p_r.
  - p_r lies in the outer open half space of these facets.
- The boundary of this set of facets is called the ***horizon*** with respect to p_r.
  - It is a simple, convex, closed curve of edges of $\mathcal{CH}(P_{r-1})$
  - Connecting the horizon to p_r, yields the missing facets for the convex hull $\mathcal{CH}(P_r)$.

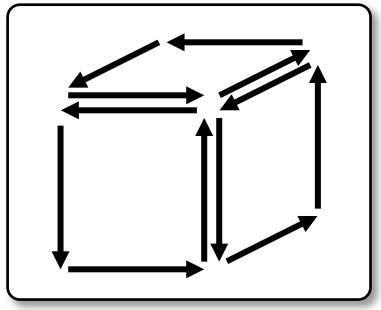


8.2 Algorithm

- **Special case:** If p_r is contained in the plane of one or more facets of P_{r-1}, the respective facets are merged.
  - Test for each edge of the horizon, if the adjacent facets (one old and one new) lie in the same plane.
  - In this case remove this edge from $\mathcal{CH}(P_r)$.

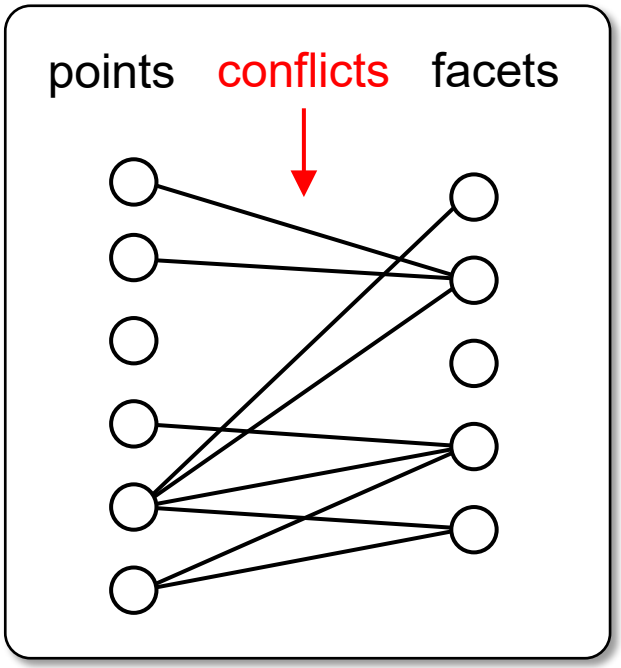


- A data structure for the convex hull to support this type of operations is a **doubly linked list**.
  - Edges are oriented such that each facet is oriented counter-clockwise, as seen from the outside.



8.2 Algorithm

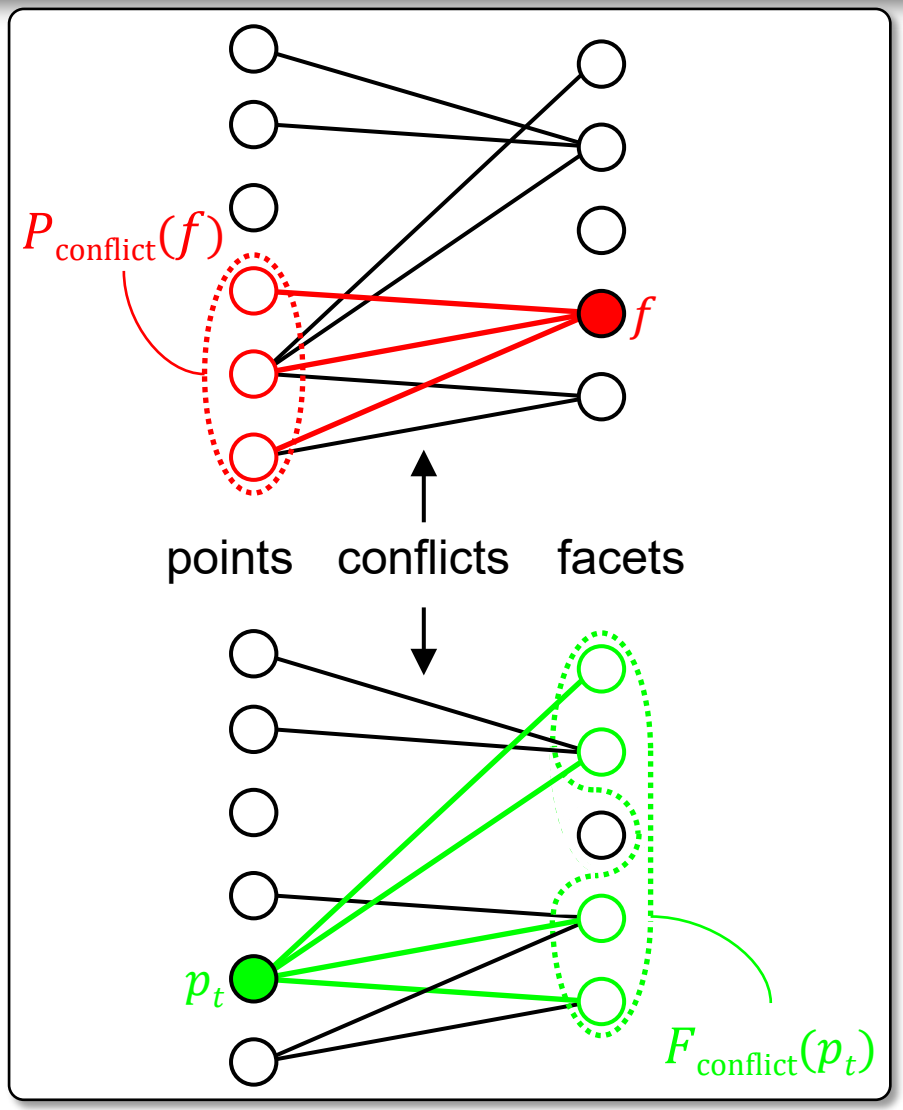
- If all facets were tested for visibility on insertion of p_r, the resulting runtime of the algorithm would be $O(n^2)$.
  - Use a data structure to accelerate this process to an expected runtime of $O(n \log n)$: **conflict graph $\mathcal{G}$**.
- If a facet $f \in \mathcal{CH}(P_r)$ is visible from a point $p \in P$, both cannot be part of the convex hull at the same time.
- The conflict graph connects all facets of $\mathcal{CH}(P_r)$ with all visible points of P.



8.2 Algorithm

- For each facet $f \in \mathcal{CH}(P_r)$ a set $P_{\text{conflict}}(f) \subseteq P \backslash P_r$ of points is maintained, that can see f.
- For each point $p_t, t > r$, a set $F_{\text{conflict}}(p_t)$ of facets of $\mathcal{CH}(P_r)$ is maintained, that can be seen from p_t.

- The conflict graph is bipartite.



8.2 Algorithm

Lemma 3

For a random subset P_r of P the corresponding conflict graph of $\mathcal{CH}(P_r)$ contains at most $n + 2r - 4$ vertices and (expected) $2(n - r)$ edges.

Proof

- Theorem 1 implies that the number of facets of $\mathcal{CH}(P_r)$ is bounded by $2r - 4$. ⇨ This yields the vertex-bound.
- Denote by d an arbitrary direction and $q \in P \backslash P_r$.
  - The probability that q is more extreme in direction d than all other points $p \in P_r$ is exactly $\frac{1}{r+1}$.

8.2 Algorithm

- Now, consider a facet $f \in \mathcal{CH}(P_r)$ with normal d.
  - The expected value for the number of conflicts with f is

$$E(\,|P_{\text{conflict}}(f)|\,) = \frac{n-r}{r+1},$$

    because the events, that a point from $P \backslash P_r$ is more extreme than any other point in P_r, are independent.
- Because $\mathcal{CH}(P_r)$ has at most $2r-4$ facets
  - all with the same expected number of conflicts $E(\,|P_{\text{conflict}}(f)|\,)$,

the expected value for the sum of all conflicts, i.e. number of edges in the conflict graph, is

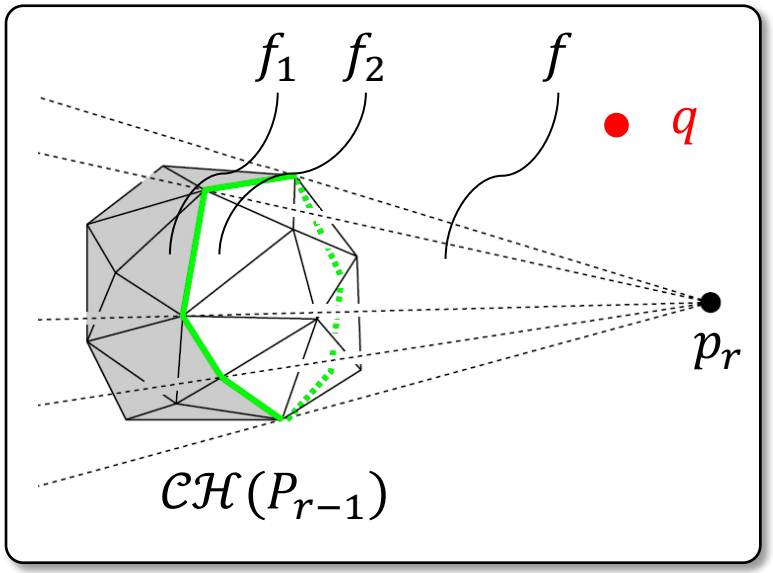
$$E(\text{number of edges}) \leq \frac{(2r-4)(n-r)}{r+1} < 2(n-r).$$

□

8.2 Algorithm

- How to update the conflict graph efficiently on the fly?
- Removals of points or faces from $\mathcal{G}$ are easy to handle.
- Denote by f a facet, that is generated on the insertion of p_r.
  - Denote by f_1, f_2 the two faces adjacent to the respective edge of the horizon of $\mathcal{CH}(P_{r-1})$.
  - Another point $q \in P \setminus P_r$ can cause a conflict with f only, if it was already in conflict with f_1 or f_2, i.e.

$$P_{\text{conflict}}(f) \subseteq P_{\text{conflict}}(f_1) \cup P_{\text{conflict}}(f_2).$$



8.2 Algorithm

- To update the conflict graph, only adjacent facets corresponding to edges of the horizon need to be considered.
  - After the generation of the new facets, the old facets and corresponding edges are removed from the conflict graph.

- **Simplification:** To simplify the algorithm we first assume that the points in P are in **general position**, i.e. no four points lie on the same plane.
  - All facets of the convex hull are triangles.

8.2 Algorithm

Algorithm 1: ConvexHull3D(Point Set P)

Input: Set $P = \{p_1, \ldots, p_n\}$ in random order and general position.
Output: The convex hull $\mathcal{CH}(P)$.

```
 1: C = CH(p_1, p_2, p_3, p_4);
 2: Initialize the conflict graph G for all visible pairs
    (f, p_i) for f ∈ C, i = 5, ..., n;
 3: for (r = 5, ..., n) {
 4:   if (F_conflict(p_r) ≠ ∅) then {
 5:     Remove all f ∈ F_conflict(p_r) from C and determine horizon H;
 6:     for all (e ∈ H) {
 7:       Determine the faces f_1, f_2 adjacent to e;
 8:       Add the facet f = (e, p_r) to C;
 9:       Add a knot corresponding to f to G;
10:       for all (p ∈ P_conflict(f_1) ∪ P_conflict(f_2)) {
11:         if (f is visible from p) then Add edge to G;
12:     } }
13:     Remove facets/edges corr. to F_conflict(p_r) and p_r from G;
14: } }
```

8.3 Analysis

Theorem 4

Algorithm 1 computes the convex hull in $O(n \log n)$ expected time and $O(n)$ expected memory.

Proof

- Memory:
  - Theorem 1 implies that the size of a convex hull is linear in n.
  - Lemma 3 implies that the expected size of a conflict graph is linear in n and r for every step r.

8.3 Analysis

- Runtime:
  - The runtime depends on the number of inserted and removed facets in C and knots and edges in $\mathcal{G}$.
  - Because only previously inserted primitives can later be removed, it suffices to count only the inserted primitives.
  - The expected value for the number of inserted facets rsp. knots (lines 8 and 9) when adding p_r depends on the location of p_r:
    - If it lies in the interior or on the boundary of $\mathcal{CH}(P_{r-1})$, no facets are generated.
    - Otherwise, Corollary 2 implies that on average at most 6 facets are generated, since p_r becomes a new vertex of $\mathcal{CH}(P_r)$.
    - ► The total number of primitives generated in lines 8 and 9 is expected to be linear in n.

8.3 Analysis

- The runtime is dominated by the number of edges added to $\mathcal{G}$ in line 11 for each new facet.
- The proof of Lemma 3 implies, that the expected value for the number of conflicts of a facets generated in step r is

$$E(\,|P_{\text{conflict}}(f)|\,) = \frac{n-r}{r+1}.$$

- Because the expected value for the valence of a vertex is 6, the expected value for the size of the horizon is also 6: $E(|H|) = 6$.
- This yields a total expected number of conflicts (NoC) of

$$E(\text{NoC}) \leq \sum_{r=1}^{n} 6\frac{n-r}{r+1} < 6n\sum_{r=1}^{n}\frac{1}{r} = O(n\log n)\,.$$

□

8.3 Analysis

- Without the assumption that the points are in general position the same runtime can be achieved [1,2].
- Some necessary changes are:

(1) Find initially four points that are not coplanar.

  - For two points p_1 and p_2 find p_i, $i = 3, \ldots, n$, that is not collinear.
  - Find a further point that is not coplanar.
  - If no such points exists, uses the 2d algorithm to compute the convex hull.
  - Otherwise, compute a random permutation of the remaining points.

8.3 Analysis

(2) In the loop in line 6 for the edges e on the horizon, a test for coplanarity of p with the facets f_1, f_2 adjacent to e is necessary.

- In this case, p is added with its facet and edge e is replaced by two edges connected to p and ...
- ... and the rest of the loop can be ignored, because the extended facet is already in the conflict graph.
- Otherwise, the loop is processed as usual.

8.4 Remarks

- An application for this algorithm is the computation of a triangulation of points on the sphere.
  - For points in general position (no four co-circular points) this yields the Delaunay triangulation.
- There is also a divide-and-conquer-algorithm to compute the convex hull in 3d with runtime $O(n \log n)$, see [3].
- The combinatorial complexity of the convex hull for dimensions $d > 3$ is not linear, but $\Theta(n^{\lfloor d/2 \rfloor})$.
  - A d-simplex contains $\binom{d+1}{r+1}$ r-simplexes ($r < d$).
  - Thus, use an output-size sensitive algorithm [4]...

8.5 Literature

[1] Marc de Berg et al., *Computational Geometry: Algorithms and Applications*, 2nd Edition, Springer, 2000, Chapters 11 and 9.

[2] K.L. Clarkson and P.W. Shor, *Applications of random sampling in computational geometry*, II. Discrete Comput. Geom., 4:387-421, 1989.

[3] F.P. Preparata and S.J. Hong, *Convex hulls of finite sets of points in two and three dimensions*, Commun. ACM, 20:87-93, 1977.

[4] T. Chan, *Optimal Output-Sensitive Convex Hull Algorithms in Two and Three Dimensions*, Discrete & Computational Geometry 16 (1996): 361-368.