Convex hulls in $\mathbb{R}^3$

Sándor Kisfaludi-Bak

Computaional Geometry
Summer semester 2020

max planck institut
informatik

Overview

- Computaitonal model, input and output

Overview

- Computaitonal model, input and output

- Preparata–Hong divide&conquer algorithm (1977)

Overview

- Computaitonal model, input and output
- Preparata–Hong divide&conquer algorithm (1977)
- Clarkson–Shor randomized incremental construction (1989)

Overview

- Computaitonal model, input and output

- Preparata–Hong divide&conquer algorithm (1977)

- Clarkson–Shor randomized incremental construction (1989)

- Chan's algorithm in $\mathbb{R}^3$ (1996)

Overview

- Computaitonal model, input and output
- Preparata–Hong divide&conquer algorithm (1977)
- Clarkson–Shor randomized incremental construction (1989)
- Chan's algorithm in $\mathbb{R}^3$ (1996)
- Higher-dimensional convex hulls

Convex hull: input and output

Input: Points with coordianate pairs $(x, y, z) \in \mathbb{R}^3$

$$(e, \pi, 1), (3, 3, \sqrt{5}), (2.95, 2.9, 2^{1.2}), (\sqrt{11}, 3.05, \sqrt[3]{3})$$

Convex hull: input and output

Input: Points with coordianate pairs $(x, y, z) \in \mathbb{R}^3$

$$(e, \pi, 1), (3, 3, \sqrt{5}), (2.95, 2.9, 2^{1.2}), (\sqrt{11}, 3.05, \sqrt[3]{3})$$

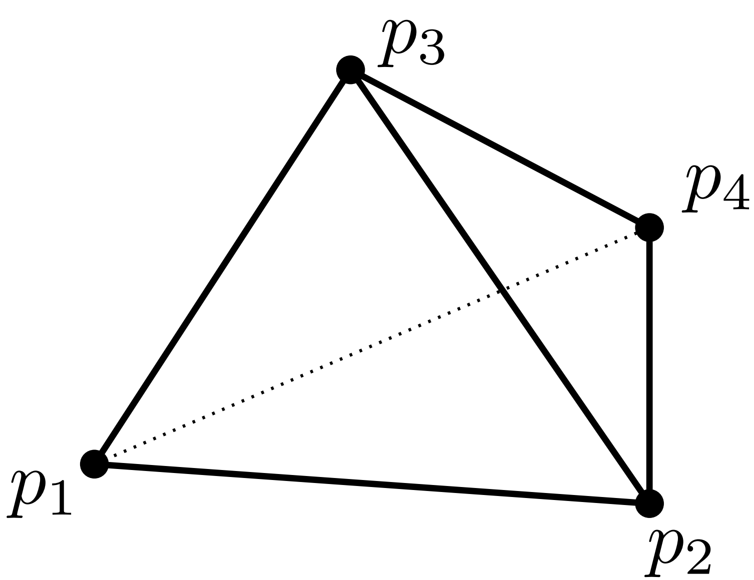
Output: planar graph of the vertices and edges of $\mathrm{conv}(P)$

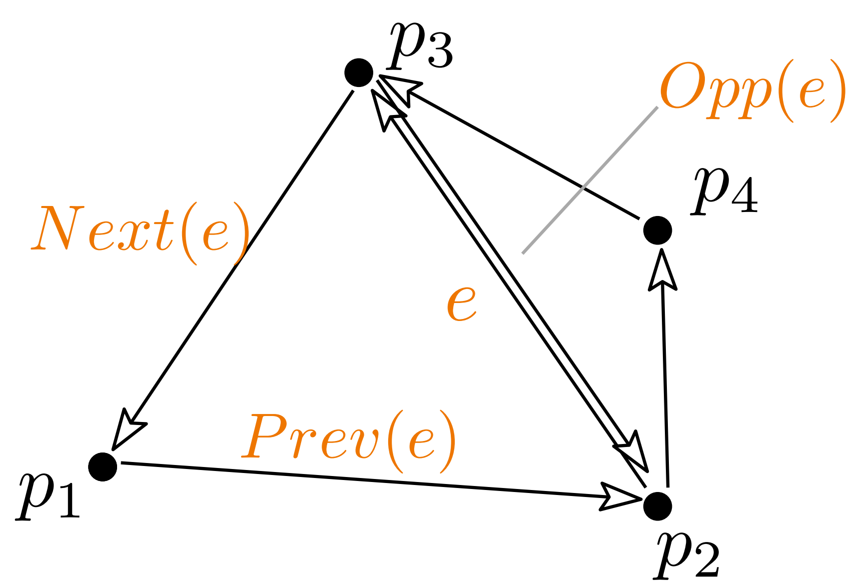
Convex hull: input and output

Input: Points with coordianate pairs $(x, y, z) \in \mathbb{R}^3$

$$(e, \pi, 1), (3, 3, \sqrt{5}), (2.95, 2.9, 2^{1.2}), (\sqrt{11}, 3.05, \sqrt[3]{3})$$

Output: planar graph of the vertices and edges of $\mathrm{conv}(P)$





Doubly connected edge list, facets are ccw cycles from outside
arcs know: opposite, next, prev arc

Convex hull: complexity

Convex hull: complexity

Claim A convex hull with h vertices has complexity $O(h)$.

- Euler's formula: $h - \#(edges) + \#(faces) = 2$.

Convex hull: complexity

Claim A convex hull with h vertices has complexity $O(h)$.

- Euler's formula: $h - \#(edges) + \#(faces) = 2$.
- Each face has ≥ 3 incident edges.
  Each edge is incident to 2 faces.

$$2 \cdot \#(edges) \geq 3 \cdot \#(faces)$$

Convex hull: complexity

Claim A convex hull with h vertices has complexity $O(h)$.

- Euler's formula: $h - \#(edges) + \#(faces) = 2$.
- Each face has ≥ 3 incident edges.
  Each edge is incident to 2 faces.

$$2 \cdot \#(edges) \geq 3 \cdot \#(faces)$$

$$\Rightarrow \#(edges) = h + \#(faces) - 2$$

$$\leq h + \frac{2}{3}\#(edges) - 2$$

Convex hull: complexity

Claim A convex hull with h vertices has complexity $O(h)$.

- Euler's formula: $h - \#(edges) + \#(faces) = 2$.
- Each face has ≥ 3 incident edges.
  Each edge is incident to 2 faces.

$$2 \cdot \#(edges) \geq 3 \cdot \#(faces)$$

$$\Rightarrow \#(edges) = h + \#(faces) - 2$$

$$\leq h + \frac{2}{3}\#(edges) - 2$$

$$\Rightarrow \#(edges) \leq 3h - 6$$

Convex hull: complexity

Claim A convex hull with h vertices has complexity $O(h)$.

- Euler's formula: $h - \#(edges) + \#(faces) = 2$.
- Each face has ≥ 3 incident edges.
  Each edge is incident to 2 faces.

$$2 \cdot \#(edges) \geq 3 \cdot \#(faces)$$

$$\begin{aligned} \Rightarrow \#(edges) &= h + \#(faces) - 2 \\ &\leq h + \frac{2}{3}\#(edges) - 2 \end{aligned}$$

$$\Rightarrow \#(edges) \leq 3h - 6$$

Total complexity (vertices+edges): $\leq 4h - 6 = O(h)$. □

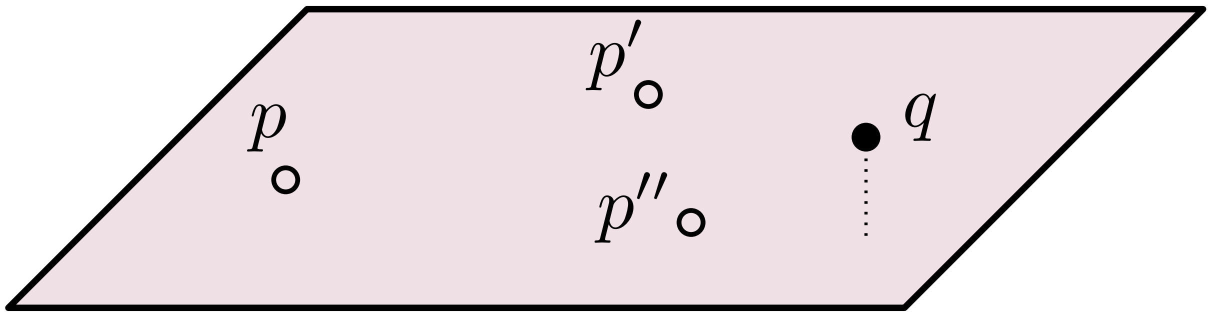
Basic operation and naive approach

Suppose no 3 pts on one line, no 4 pts in one plane.

Basic operation and naive approach

Suppose no 3 pts on one line, no 4 pts in one plane.

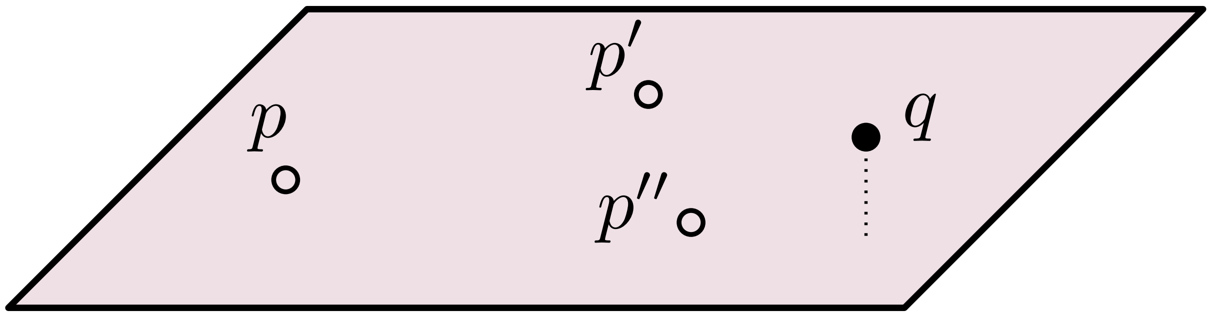
In $O(1)$ time, decide if q is above/below/on plane $pp'p''$



Basic operation and naive approach

Suppose no 3 pts on one line, no 4 pts in one plane.

In $O(1)$ time, decide if q is above/below/on plane $pp'p''$

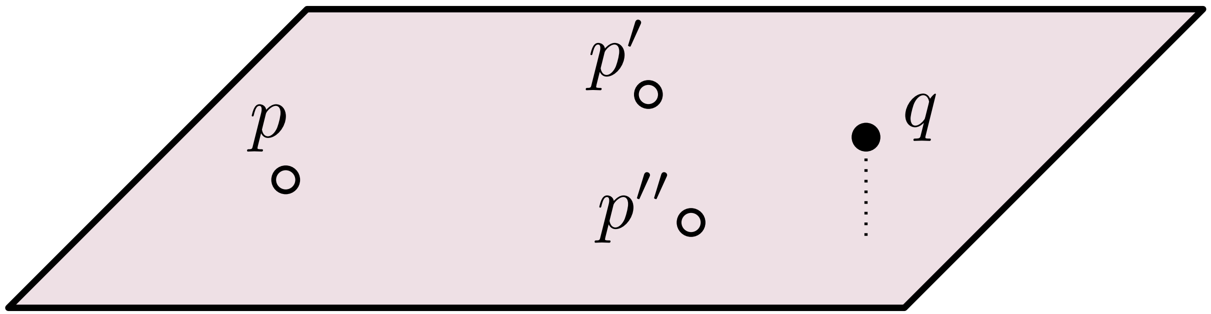


$$\begin{vmatrix} p_1 & p_2 & p_3 & 1 \\ p'_1 & p'_2 & p'_3 & 1 \\ p''_1 & p''_2 & p''_3 & 1 \\ q_1 & q_2 & q_3 & 1 \end{vmatrix}$$

Basic operation and naive approach

Suppose no 3 pts on one line, no 4 pts in one plane.

In $O(1)$ time, decide if q is above/below/on plane $pp'p''$



$$\begin{vmatrix} p_1 & p_2 & p_3 & 1 \\ p'_1 & p'_2 & p'_3 & 1 \\ p''_1 & p''_2 & p''_3 & 1 \\ q_1 & q_2 & q_3 & 1 \end{vmatrix}$$

Naive Convex Hull in $\mathbb{R}^3$

For each $p, p', p'' \in P$,

check if all $q \in P \setminus \{p, p', p''\}$ is on same side of plane $pp'p''$.

If yes, then $pp'p''$ is a face.

Running time: $\binom{n}{3} \cdot (n-3) \cdot O(1) = O(n^4)$

$\mathbb{R}^3$ convex hull with divide and conquer (Preparata–Hong, 1977)

Divide and conquer

$(p_1, \ldots, p_n) = \textsc{LexicographicSort}(P)$
$H = \textsc{Hull3dim}((p_1, \ldots, p_n))$

function $\textsc{Hull3dim}((p_1, \ldots, p_n))$
  if $n \leq 4$ **then**
    return $\textsc{NaiveHull}((p_1, \ldots, p_n))$
  $H_1 = \textsc{Hull3dim}((p_1, \ldots, p_{\lfloor n/2 \rfloor}))$
  $H_2 = \textsc{Hull3dim}((p_{\lfloor n/2 \rfloor + 1}, \ldots, p_n))$
  $H = \textsc{Merge}(H_1, H_2)$
  return H

Divide and conquer

$(p_1, \ldots, p_n) = \textsc{LexicographicSort}(P)$ $O(n \log n)$
$H = \textsc{Hull3dim}((p_1, \ldots, p_n))$

function $\textsc{Hull3dim}((p_1, \ldots, p_n))$
  if $n \leq 4$ **then**
    return $\textsc{NaiveHull}((p_1, \ldots, p_n))$
  $H_1 = \textsc{Hull3dim}((p_1, \ldots, p_{\lfloor n/2 \rfloor}))$
  $H_2 = \textsc{Hull3dim}((p_{\lfloor n/2 \rfloor + 1}, \ldots, p_n))$
  $H = \textsc{Merge}(H_1, H_2)$
  return H

Divide and conquer

$(p_1, \ldots, p_n) = \textsc{LexicographicSort}(P)$ $\quad O(n \log n)$

$H = \textsc{Hull3dim}((p_1, \ldots, p_n))$

function $\textsc{Hull3dim}((p_1, \ldots, p_n))$

  if $n \leq 4$ **then**

    return $\textsc{NaiveHull}((p_1, \ldots, p_n))$

  $H_1 = \textsc{Hull3dim}((p_1, \ldots, p_{\lfloor n/2 \rfloor}))$

  $H_2 = \textsc{Hull3dim}((p_{\lfloor n/2 \rfloor + 1}, \ldots, p_n))$

  $H = \textsc{Merge}(H_1, H_2)$

  return H

$$T(n) = 2T(n/2) + O(n)$$

Recursion depth$= O(\log n) \ \Rightarrow \ T(n) = O(n \log n)$

Divide and conquer

$(p_1, \ldots, p_n) = \textsc{LexicographicSort}(P)$ $O(n \log n)$
$H = \textsc{Hull3dim}((p_1, \ldots, p_n)$

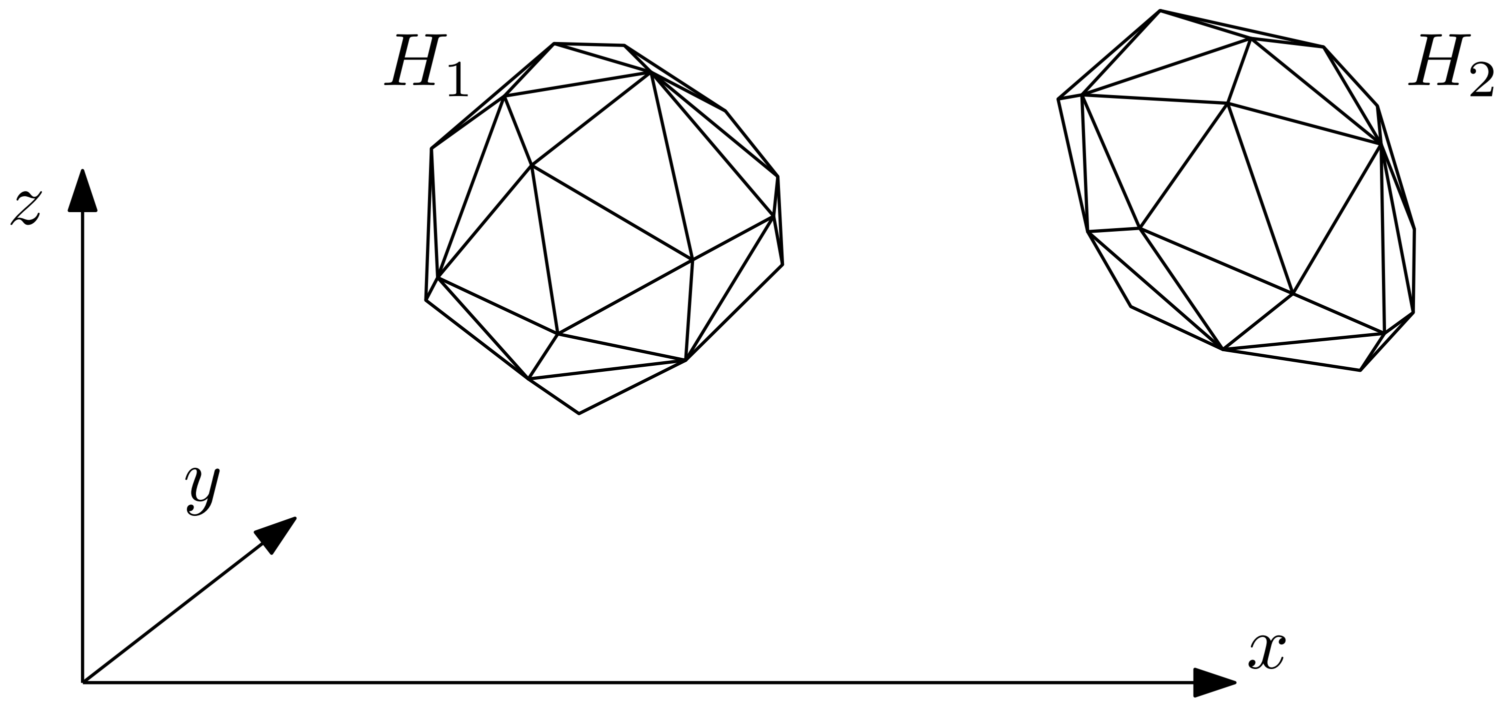
function $\textsc{Hull3dim}((p_1, \ldots, p_n))$
if $n \leq 4$ **then**
return $\textsc{NaiveHull}((p_1, \ldots, p_n))$
$H_1 = \textsc{Hull3dim}((p_1, \ldots, p_{\lfloor n/2 \rfloor}))$
$H_2 = \textsc{Hull3dim}((p_{\lfloor n/2 \rfloor + 1}, \ldots, p_n))$
$H = \boxed{\textsc{Merge}(H_1, H_2)}$
return H

$$T(n) = 2T(n/2) + O(n)$$

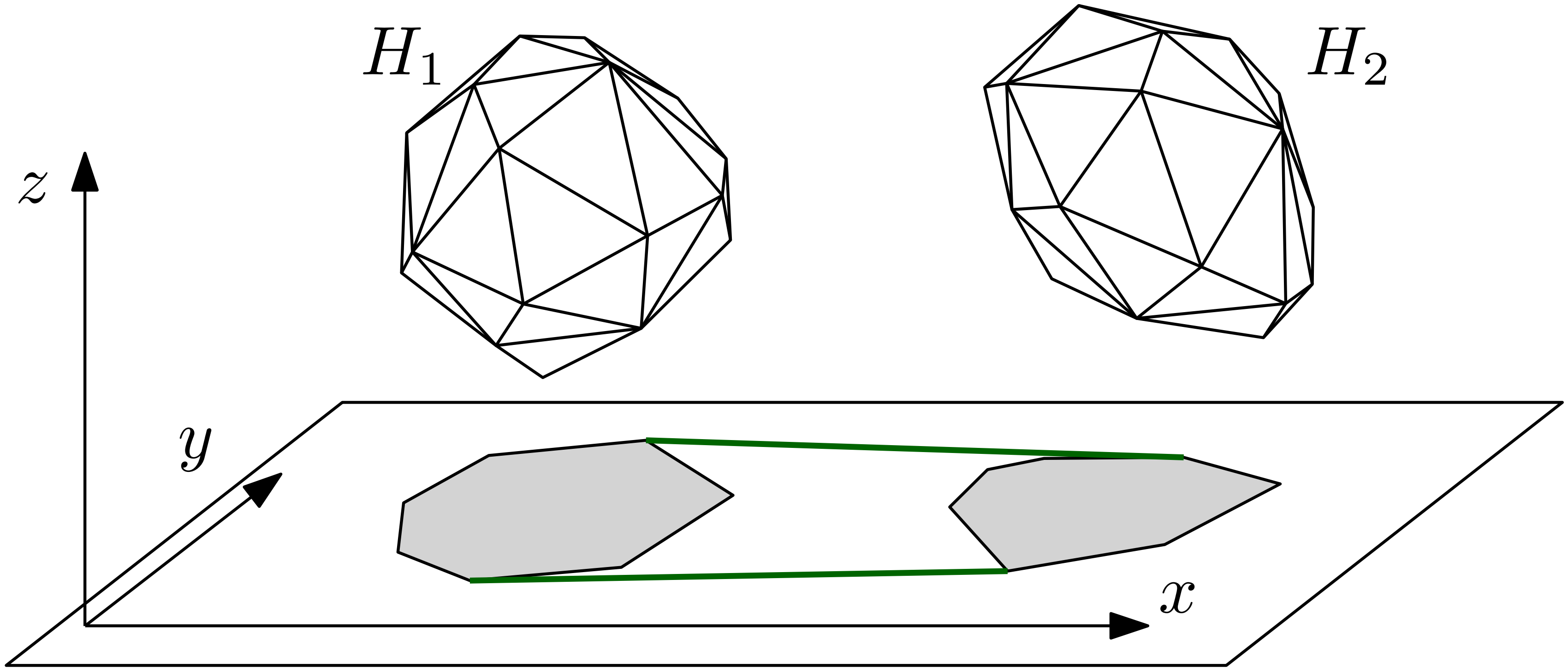
Recursion depth$= O(\log n) \Rightarrow T(n) = O(n \log n)$

We need to merge in $O(n)$ time!

Starting the merge



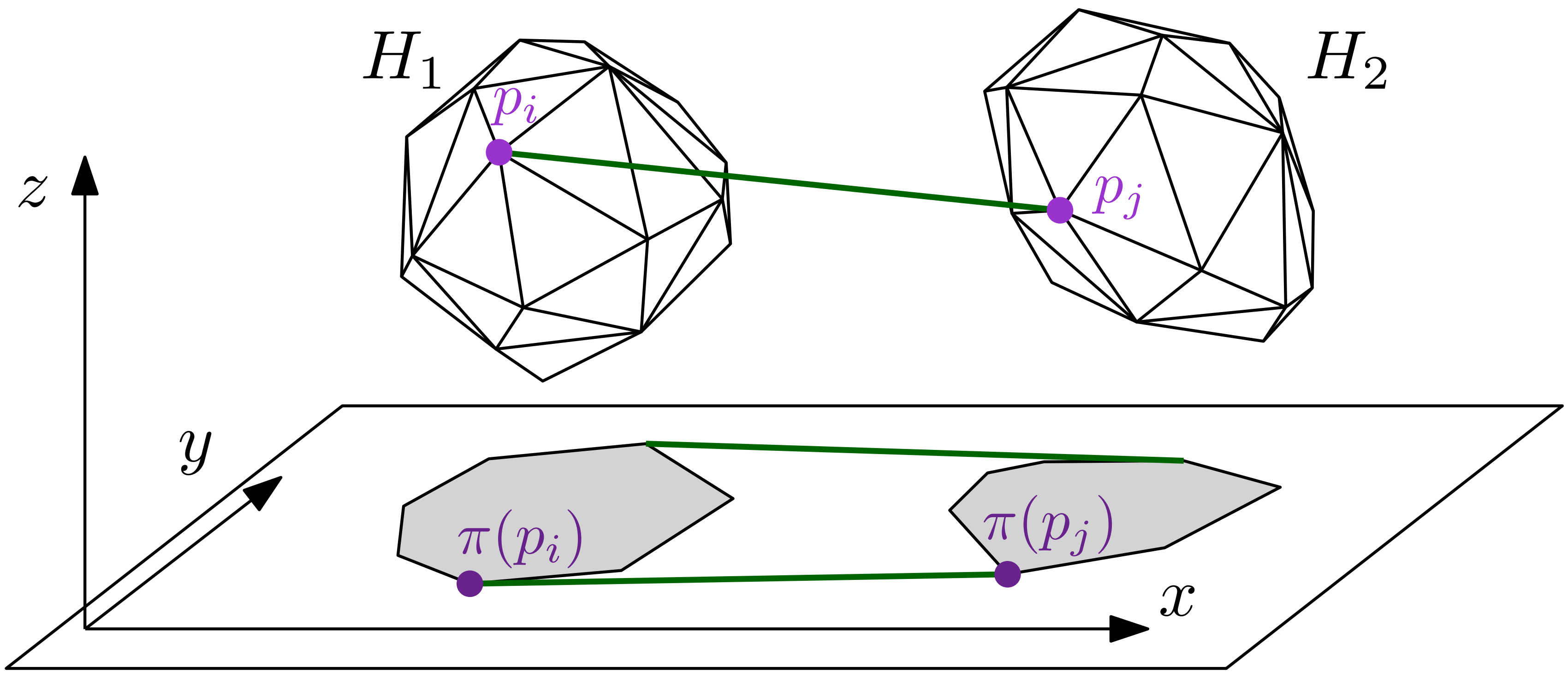
Starting the merge



Project H_1 and H_2 to xy plane

Get common tangents (as in Assignment 1/7) $\rightarrow O(n)$

Starting the merge



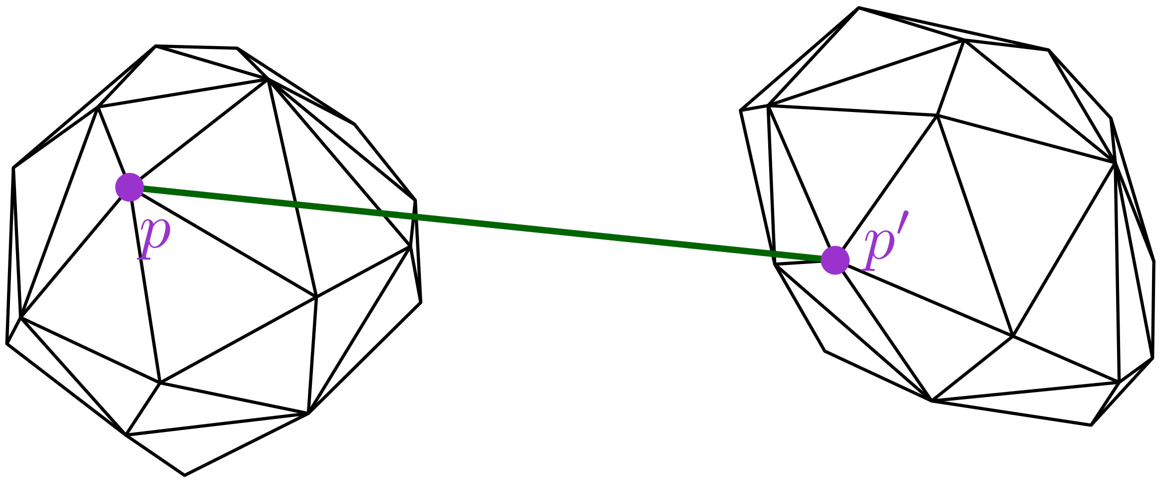
Project H_1 and H_2 to xy plane

Get common tangents (as in Assignment 1/7) $\rightarrow O(n)$

$\pi(p_i)\pi(p_j)$ is segment of $\pi(\mathrm{conv}(H_1)) \cup \pi(\mathrm{conv}(H_2))$

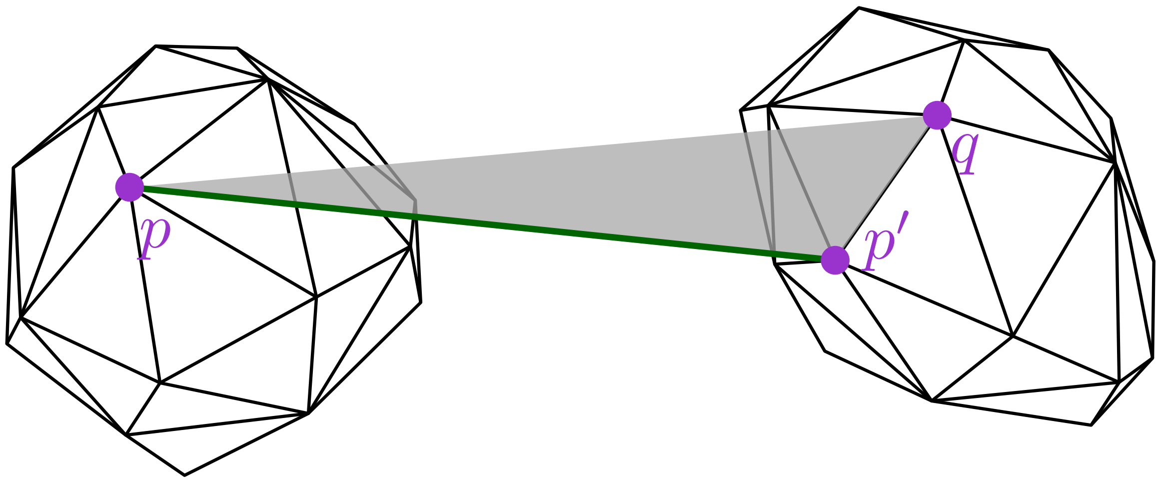
$\Rightarrow p_i p_j$ is a segment of $\mathrm{conv}(P)$.

Merging naively



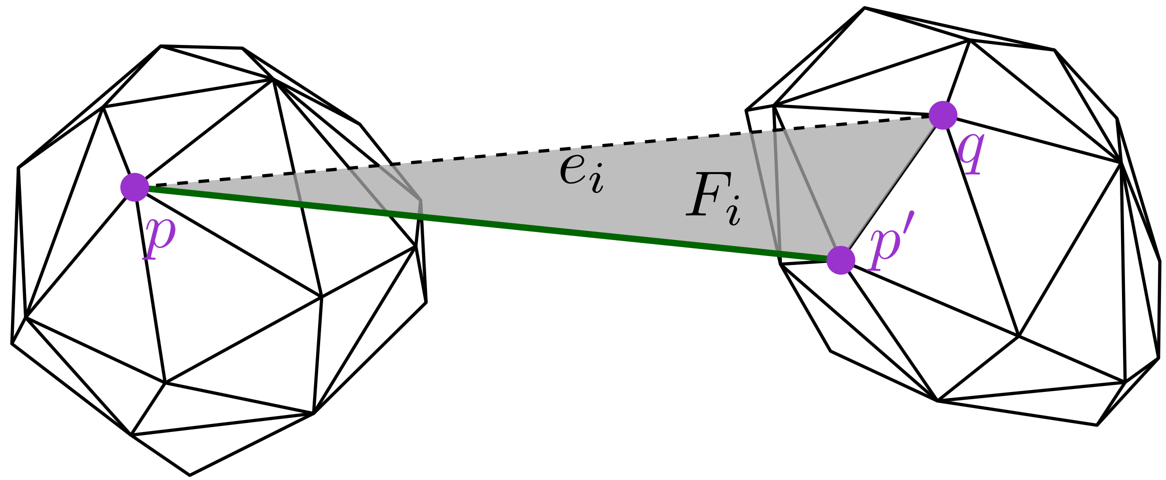
Find $q \in P$ s.t. plane $pp'q$ has smallest angle with plane $pp'\pi(p)$.

Merging naively



Find $q \in P$ s.t. plane $pp'q$ has smallest angle with plane $pp'\pi(p)$.

Merging naively

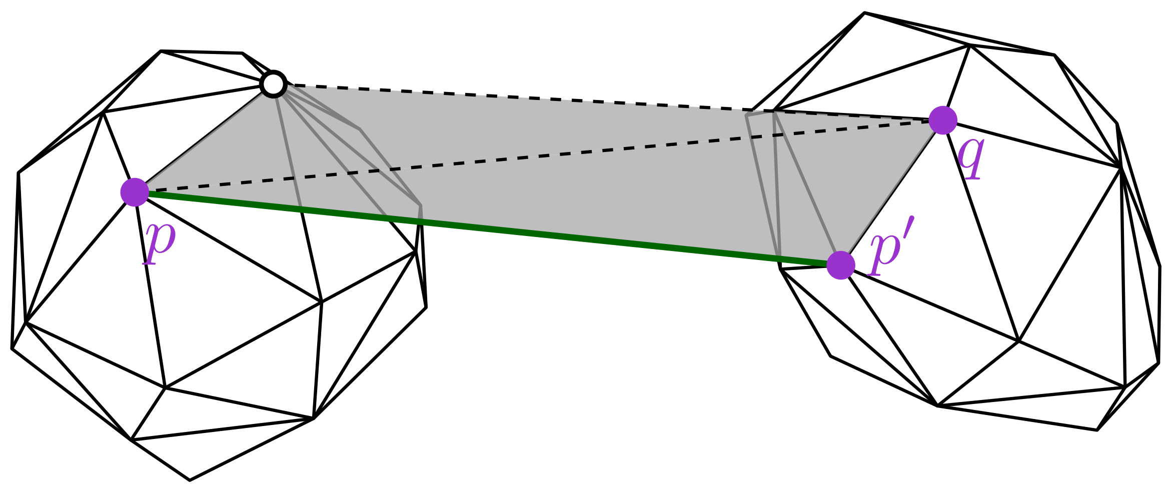


Find $q \in P$ s.t. plane $pp'q$ has smallest angle with plane $pp'\pi(p)$.

Last face found: F_i, last edge found: e_i

Repeat: Find $p \in P$ maximizing $\sphericalangle\Big((\text{plane } e_i p), (\text{plane } F_i)\Big)$

Merging naively

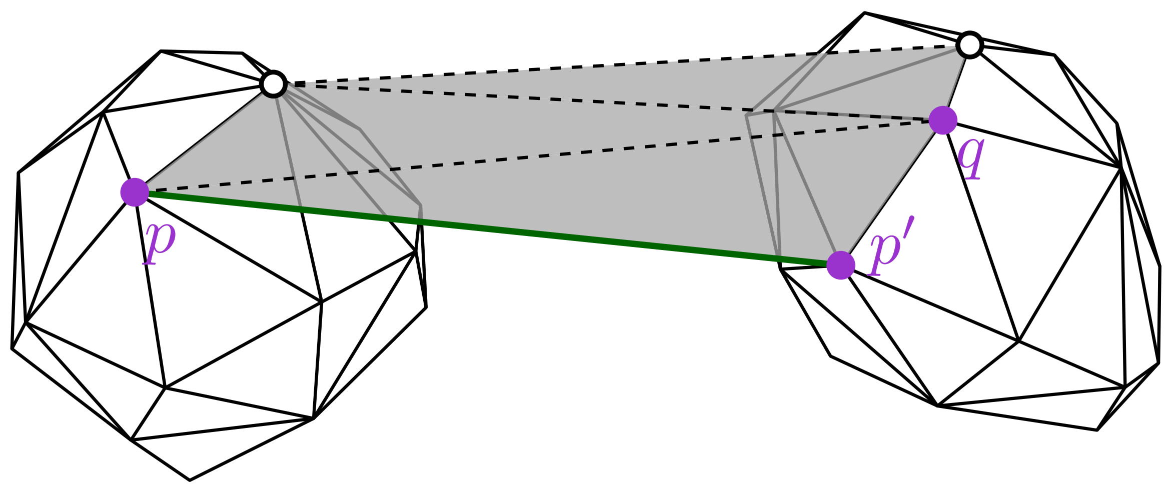


Find $q \in P$ s.t. plane $pp'q$ has smallest angle with plane $pp'\pi(p)$.

Last face found:F_i, last edge found: e_i

Repeat: Find $p \in P$ maximizing $\sphericalangle\Big((\text{plane } e_i p), (\text{plane } F_i)\Big)$

Merging naively

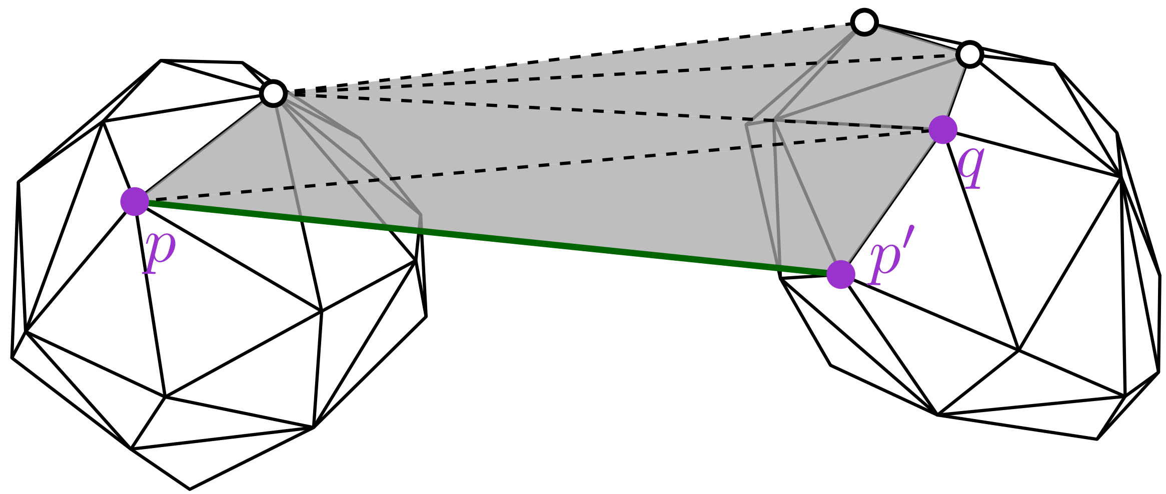


Find $q \in P$ s.t. plane $pp'q$ has smallest angle with plane $pp'\pi(p)$.

Last face found:F_i, last edge found: e_i

Repeat: Find $p \in P$ maximizing $\sphericalangle\Big((\text{plane } e_i p), (\text{plane } F_i)\Big)$

Merging naively

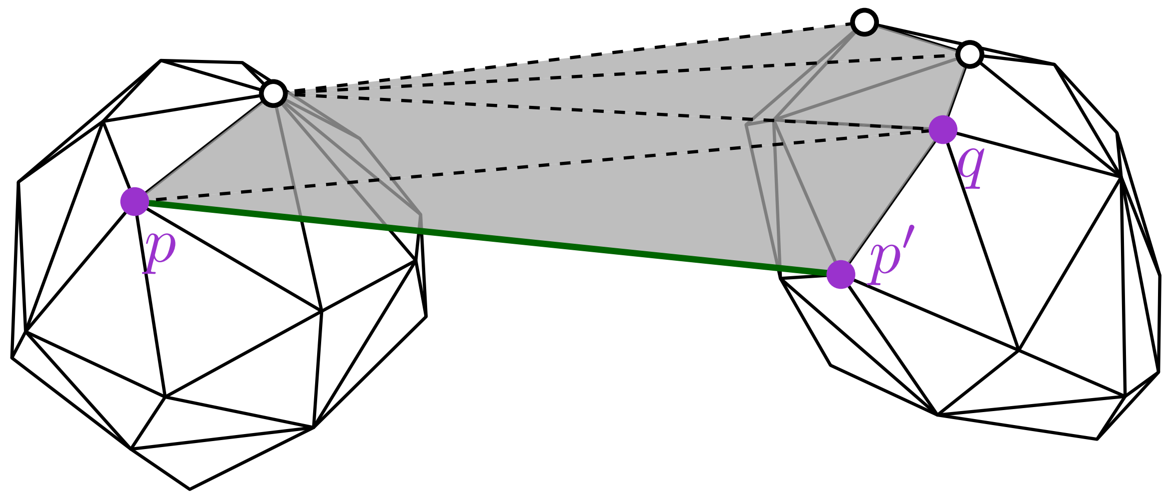


Find $q \in P$ s.t. plane $pp'q$ has smallest angle with plane $pp'\pi(p)$.

Last face found: F_i, last edge found: e_i

Repeat: Find $p \in P$ maximizing $\sphericalangle\Big((\text{plane } e_i p), (\text{plane } F_i)\Big)$

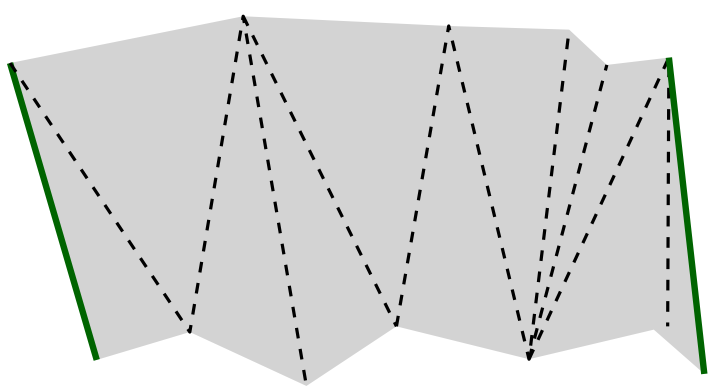
Merging naively



Find $q \in P$ s.t. plane $pp'q$ has smallest angle with plane $pp'\pi(p)$.

Last face found: F_i, last edge found: e_i

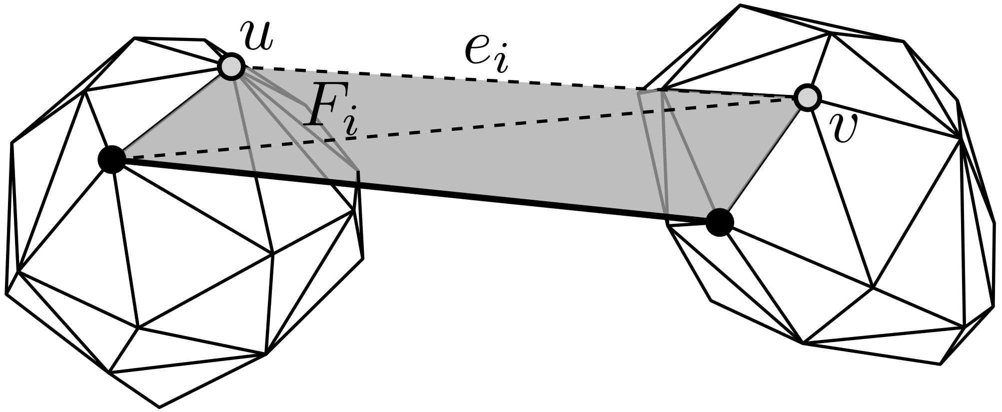
Repeat: Find $p \in P$ maximizing $\sphericalangle\Big((\text{plane } e_i p), (\text{plane } F_i)\Big)$ $\rightarrow O(n)$



The result is a "cylinder".

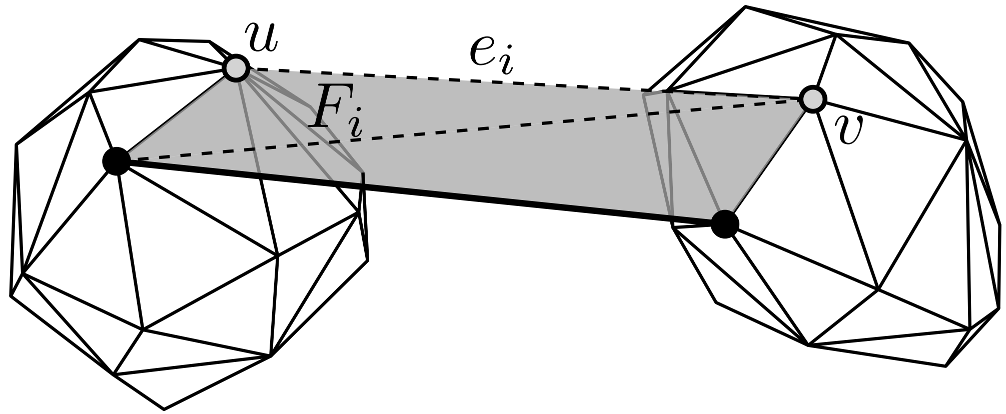
Naive runtime: $O(n^2)$

Smart merge idea



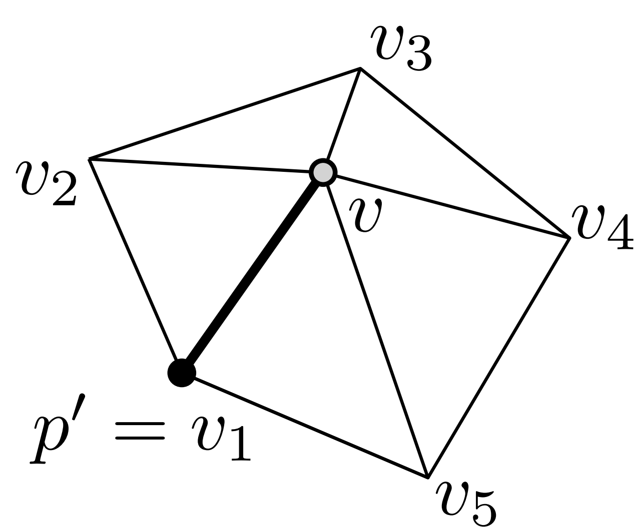
$\max_{p \in P} \sphericalangle\Big((\text{plane } uvp), (\text{plane } F_i)\Big) \rightarrow$ p must be neighbor of u or v

Smart merge idea

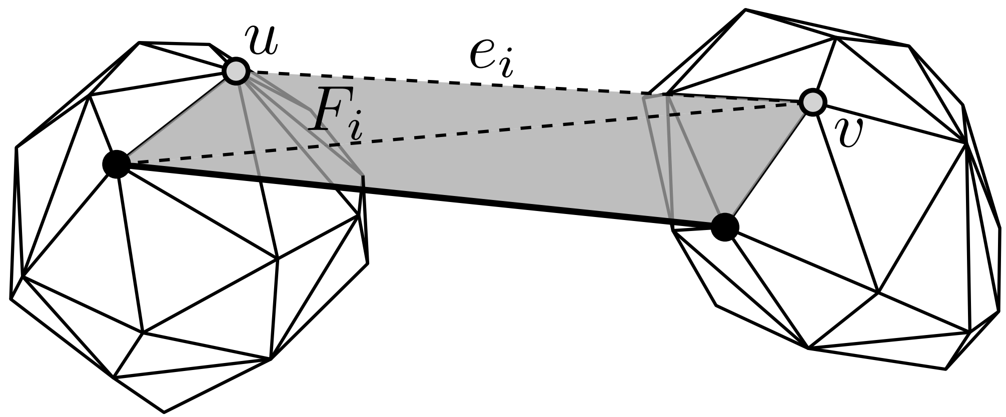


$\max_{p \in P} \sphericalangle\Big((\text{plane } uvp), (\text{plane } F_i)\Big) \rightarrow$ p must be neighbor of u or v

Maximize in $N(u)$ and $N(v)$ separately.
Let $p' \in N(v)$ s.t. $vp' \in \mathrm{conv}(P)$ is already known



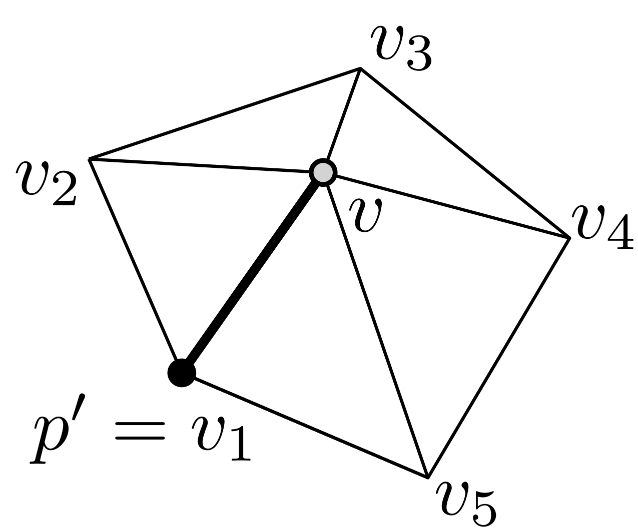
Smart merge idea



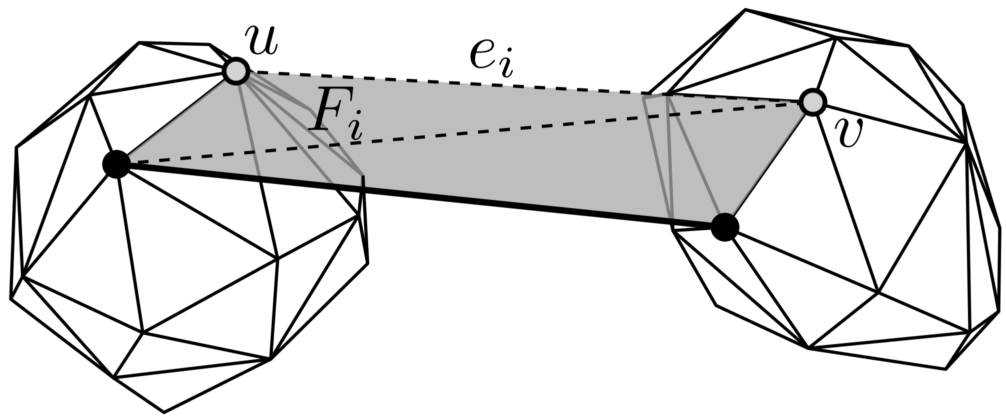
$\max_{p \in P} \sphericalangle\Big((\text{plane } uvp), (\text{plane } F_i)\Big) \rightarrow$ p must be neighbor of u or v

Maximize in $N(u)$ and $N(v)$ separately.
Let $p' \in N(v)$ s.t. $vp' \in \text{conv}(P)$ is already known

Check angles of $N(v)$ in cw order.



Smart merge idea

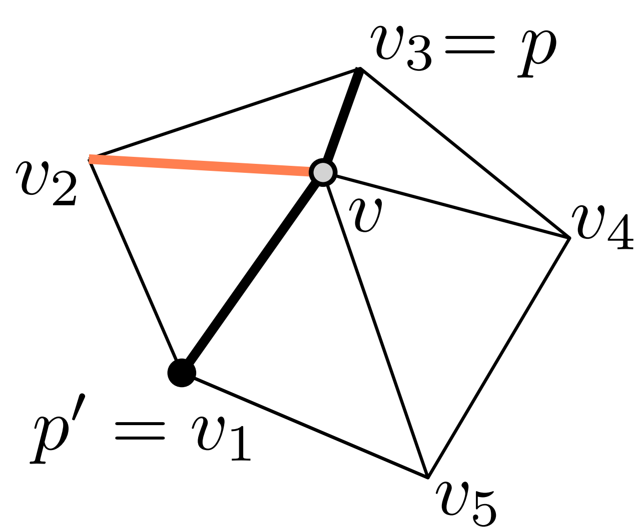


$\max_{p \in P} \sphericalangle\Big((\text{plane } uvp), (\text{plane } F_i)\Big) \rightarrow p$ must be neighbor of u or v

Maximize in $N(u)$ and $N(v)$ separately.
Let $p' \in N(v)$ s.t. $vp' \in \mathrm{conv}(P)$ is already known

Check angles of $N(v)$ in cw order.

If $v_i \in p$ has largest angle
$\Rightarrow v_2, \ldots, v_{i-1}$ are *inside* $\mathrm{conv}(P)$!



Smart merge

function $\textsc{Merge}(H_1, H_2)$

  $uv =$ starting edge (form common tangent of $\pi(H_1)$ and $\pi(H_2)$)

  $\hat{u} = \text{argmax}_{u' \in N(u)} \sphericalangle\big(uvu', uv\pi(u)\big)$

  $\hat{v} = \text{argmax}_{v' \in N(v)} \sphericalangle\big(uvv', uv\pi(u)\big)$

  repeat

    if $\hat{v}$ has larger angle than $\hat{u}$ **then**

      $v_{prev} = v, \quad v = \hat{v}$

      Add face $F = uvv_{prev}$ to H

      If F is coplanar with previous face, merge.

      for $i = 2$ to $|N(v)|$ **do** ▷ in cw order

        if $\sphericalangle\big(uvv_i, uvv_{prev}\big) > \sphericalangle\big(uvv_{i-1}, uvv_{prev}\big)$ **then**

          Remove edge vv_{i-1} from H_2

        else

          $\hat{v} = v_{i-1}$, **Break**

    else

      (same, but swap v and u, H_2 and H_1, cw and ccw)

  until $uv =$ starting edge

  return merge of cylinder H with H_1 and H_2

Smart merge

function $\textsc{Merge}(H_1, H_2)$

$uv =$ starting edge (form common tangent of $\pi(H_1)$ and $\pi(H_2)$)

$\hat{u} = \mathrm{argmax}_{u' \in N(u)} \sphericalangle(uvu', uv\pi(u))$

$\hat{v} = \mathrm{argmax}_{v' \in N(v)} \sphericalangle(uvv', uv\pi(u))$ init

repeat

if $\hat{v}$ has larger angle than $\hat{u}$ **then**

$v_{prev} = v, \quad v = \hat{v}$

Add face $F = uvv_{prev}$ to H Face add

If F is coplanar with previous face, merge.

for $i = 2$ to $|N(v)|$ **do** ▷ in cw order

if $\sphericalangle(uvv_i, uvv_{prev}) > \sphericalangle(uvv_{i-1}, uvv_{prev})$ **then**

Remove edge vv_{i-1} from H_2

else

$\hat{v} = v_{i-1}$, **Break** Step

else

(same, but swap v and u, H_2 and H_1, cw and ccw)

until $uv =$ starting edge

return merge of cylinder H with H_1 and H_2 Cylinder merge

Smart merge analysis

Init: $O(n)$

Face add: $O(1)$ (overall $O(n)$)

Smart merge analysis

Init: $O(n)$

Face add: $O(1)$ (overall $O(n)$)

Step: Each comparison results in either

$\rightarrow$ deleting an edge

$\rightarrow$ making the step

Smart merge analysis

Init: $O(n)$

Face add: $O(1)$ (overall $O(n)$)

Step: Each comparison results in either

$\rightarrow$ deleting an edge

$\rightarrow$ making the step

H_1, H_2 have $O(n)$ edges

Final cylinder has size $O(n)$

$\Rightarrow$ Amortized $O(1)$ (overall $O(n)$) time.

Smart merge analysis

Init: $O(n)$

Face add: $O(1)$ (overall $O(n)$)

Step: Each comparison results in either

$\rightarrow$ deleting an edge

$\rightarrow$ making the step

H_1, H_2 have $O(n)$ edges

Final cylinder has size $O(n)$

$\Rightarrow$ Amortized $O(1)$ (overall $O(n)$) time.

Cylinder merge: $O(1)$ time per cylinder boundary edge.
Boundary has size $O(n)$.

Smart merge analysis

Init: $O(n)$

Face add: $O(1)$ (overall $O(n)$)

Step: Each comparison results in either

$\rightarrow$ deleting an edge

$\rightarrow$ making the step

H_1, H_2 have $O(n)$ edges

Final cylinder has size $O(n)$

$\Rightarrow$ Amortized $O(1)$ (overall $O(n)$) time.

Cylinder merge: $O(1)$ time per cylinder boundary edge.
Boundary has size $O(n)$.

$\Rightarrow$ Merge takes $O(n)$ time.

Smart merge analysis

Init: $O(n)$

Face add: $O(1)$ (overall $O(n)$)

Step: Each comparison results in either

$\rightarrow$ deleting an edge

$\rightarrow$ making the step

H_1, H_2 have $O(n)$ edges

Final cylinder has size $O(n)$

$\Rightarrow$ Amortized $O(1)$ (overall $O(n)$) time.

Cylinder merge: $O(1)$ time per cylinder boundary edge.
Boundary has size $O(n)$.

$\Rightarrow$ Merge takes $O(n)$ time.

Preparata–Hong divide&conquer takes $O(n \log n)$ time. ■

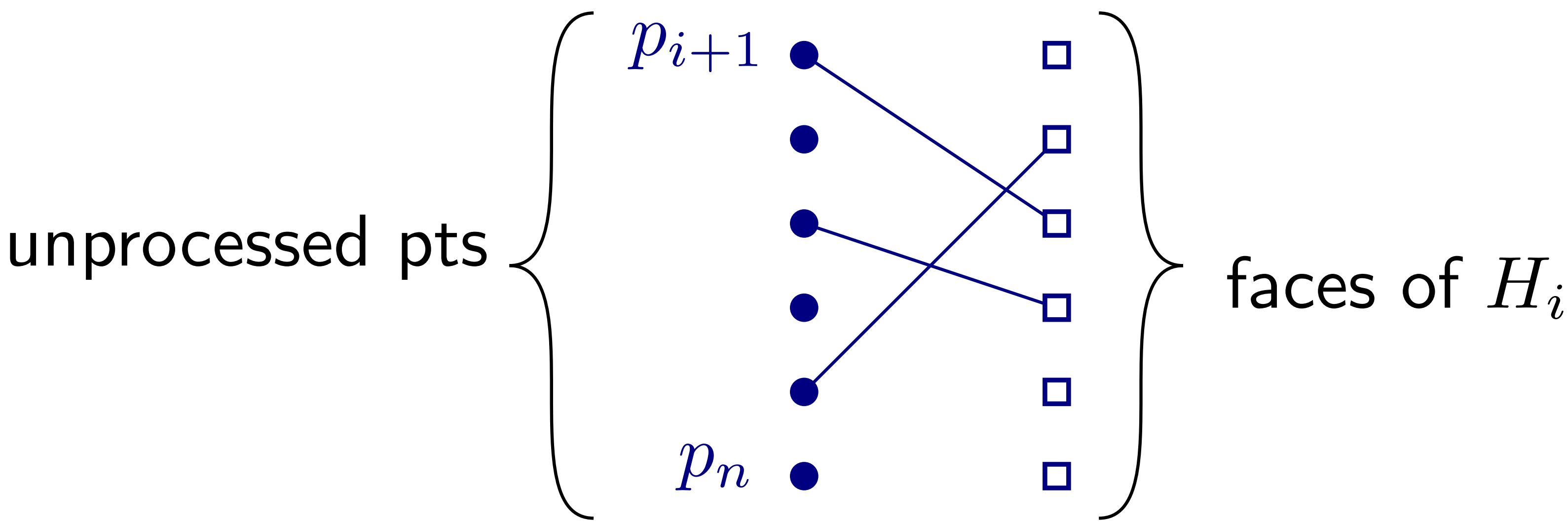
$\mathbb{R}^3$ Convex hull with rand. incremental construction (Clarkson and Shor 1989)

Ideas

- Add points one at a time, update
  $H_i =$ planar graph for $\mathrm{conv}(p_1, \ldots, p_i)$.

Ideas

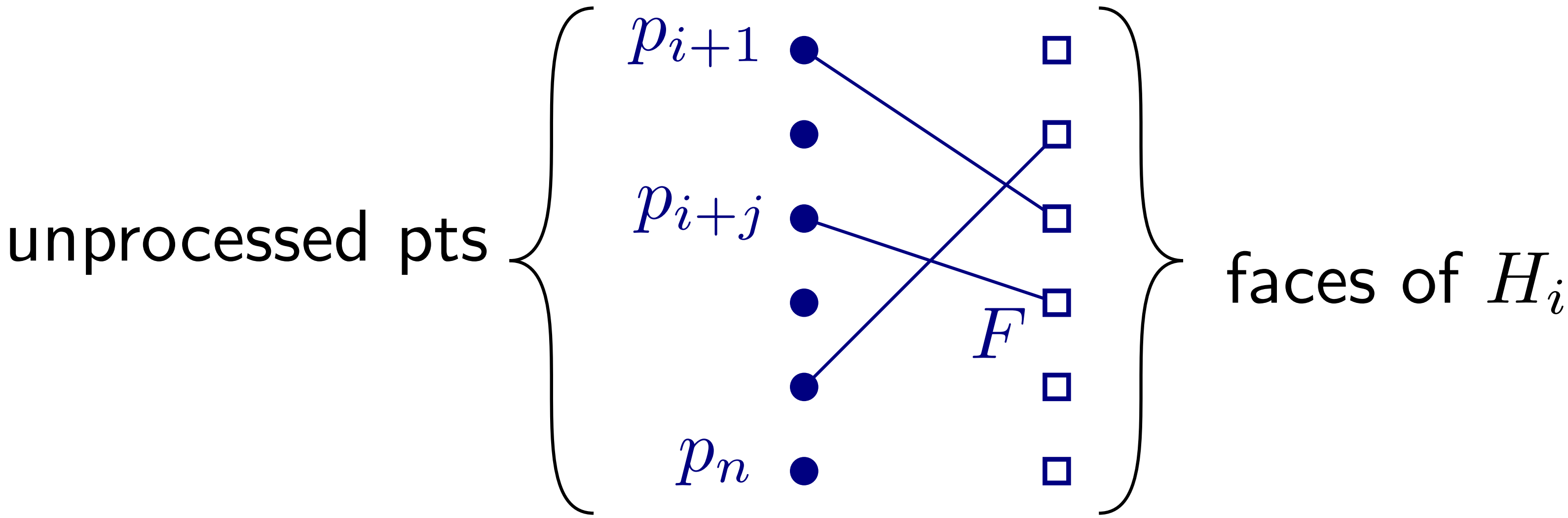
- Add points one at a time, update H_i = planar graph for $\mathrm{conv}(p_1, \ldots, p_i)$.
- Maintain a *conflict graph*: C_i



Ideas

- Add points one at a time, update
  H_i = planar graph for $\mathrm{conv}(p_1, \ldots, p_i)$.

- Maintain a *conflict graph*: C_i

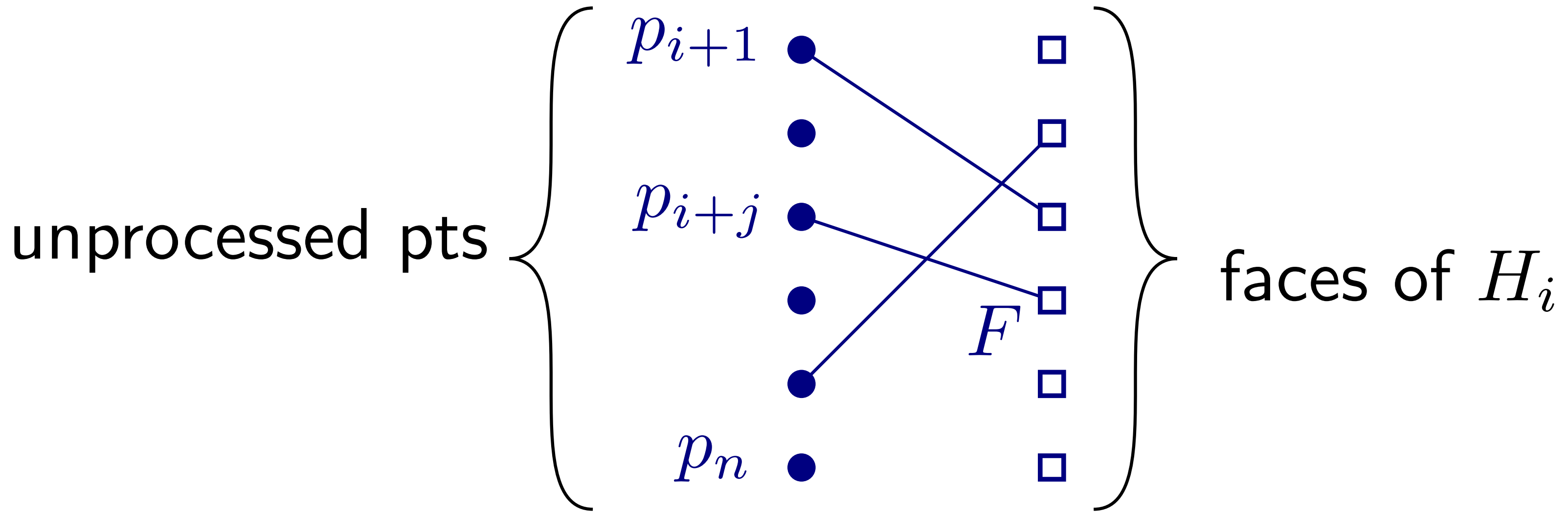


Conflict edge from p_{i+j} to face F of H_i
iff plane of F separates p_{i+j} from $\mathrm{conv}(p_1, \ldots, p_i)$

Ideas

- Add points one at a time, update H_i = planar graph for $\mathrm{conv}(p_1, \ldots, p_i)$.

- Maintain a *conflict graph*: C_i



Conflict edge from p_{i+j} to face F of H_i
iff plane of F separates p_{i+j} from $\mathrm{conv}(p_1, \ldots, p_i)$

$\Rightarrow$ if we add p_{i+j}, we must delete F

Algorithm overview

Find 4 points forming a tetrahedron, set up H_4

Algorithm overview

Find 4 points forming a tetrahedron, set up H_4

Randomly permute the other points: $p_4, \ldots, p_n$, and set up C_4

$n - 4$ unproc. pts
4 faces $\Rightarrow O(n)$

Algorithm overview

Find 4 points forming a tetrahedron, set up H_4

Randomly permute the other points: $p_4, \ldots, p_n$, and set up C_4

"Randomized incremental construction"

$n-4$ unproc. pts
4 faces $\Rightarrow O(n)$

Algorithm overview

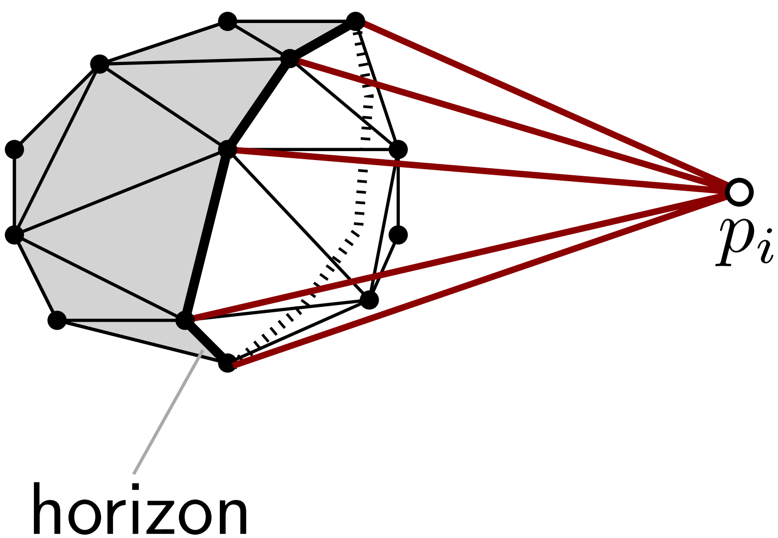
Find 4 points forming a tetrahedron, set up H_4

Randomly permute the other points: $p_4, \ldots, p_n$, and set up C_4

$n-4$ unproc. pts
4 faces $\Rightarrow O(n)$

"Randomized incremental construction"

Adding p_i:



Algorithm overview

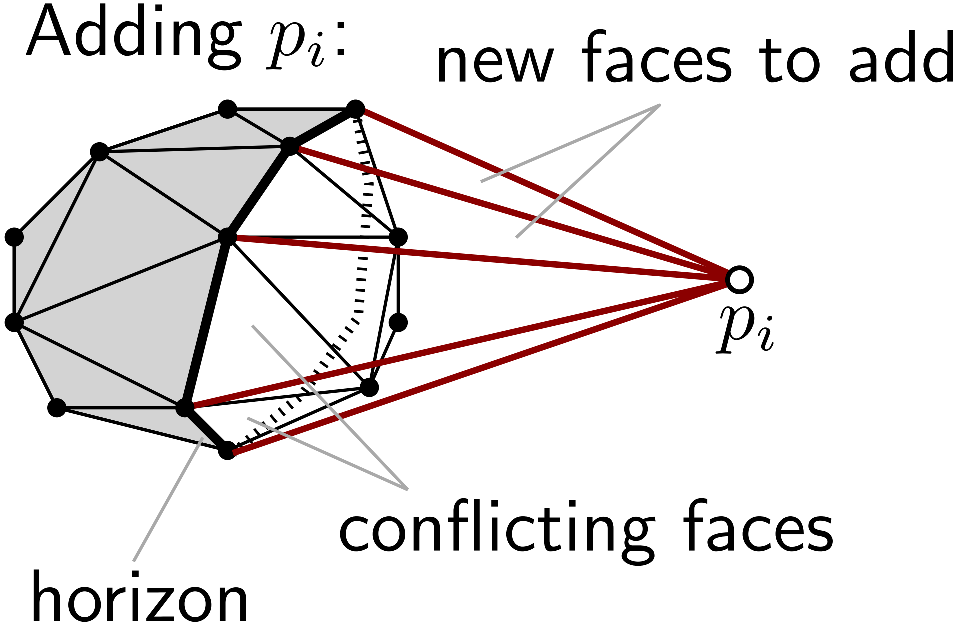
Find 4 points forming a tetrahedron, set up H_4

Randomly permute the other points: $p_4, \ldots, p_n$, and set up C_4

$n-4$ unproc. pts
4 faces $\Rightarrow O(n)$

"Randomized incremental construction"

Adding p_i:



Algorithm overview

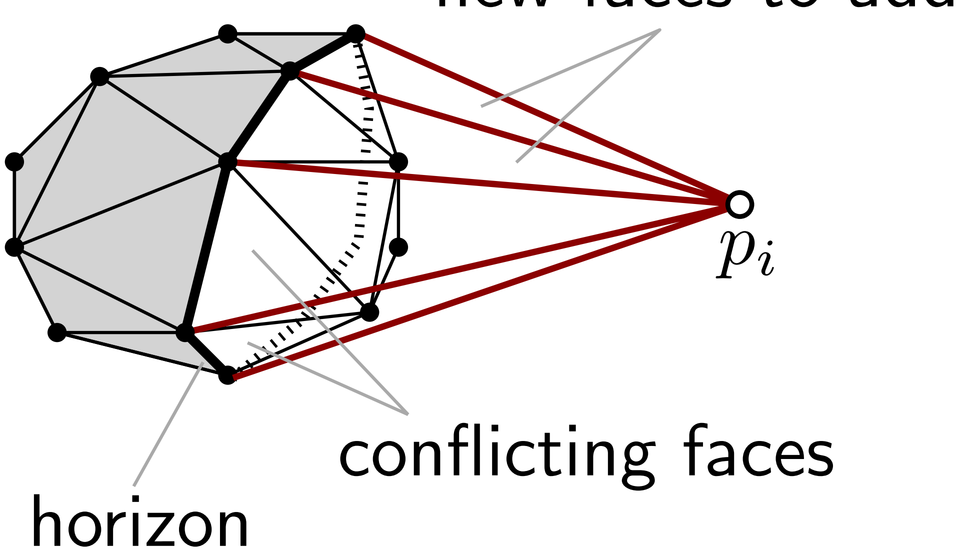
Find 4 points forming a tetrahedron, set up H_4

Randomly permute the other points: $p_4, \ldots, p_n$, and set up C_4

"Randomized incremental construction"

$n-4$ unproc. pts
4 faces $\Rightarrow$ $O(n)$

Adding p_i:



$\rightarrow$ remove conflicting faces using conflict graph

Algorithm overview

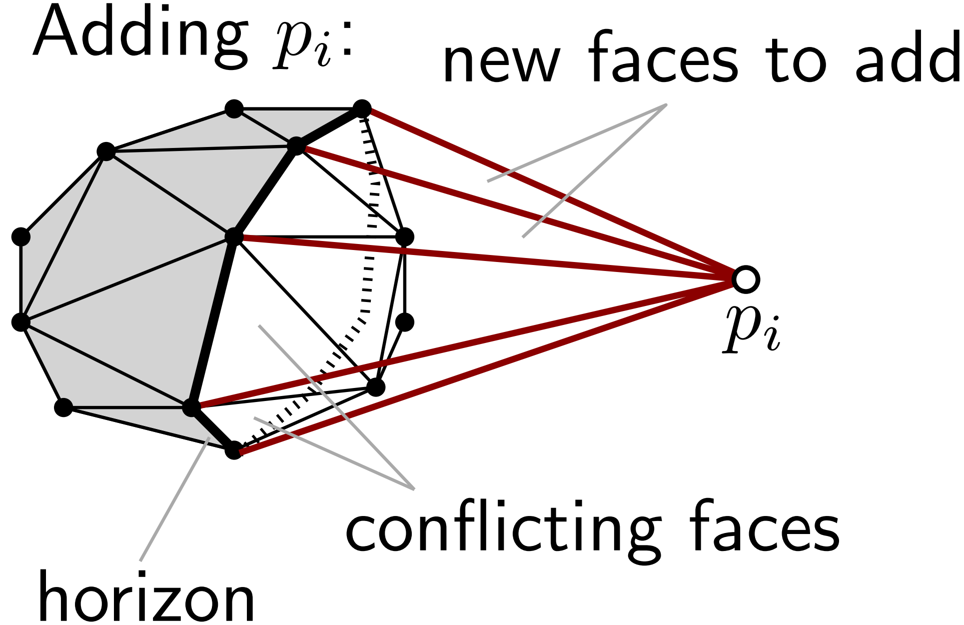
Find 4 points forming a tetrahedron, set up H_4

Randomly permute the other points: $p_4, \ldots, p_n$, and set up C_4

$n - 4$ unproc. pts
4 faces $\Rightarrow O(n)$

"Randomized incremental construction"

Adding p_i:



$\rightarrow$ remove conflicting faces using conflict graph

$\rightarrow$ walk horizon and add new faces

Algorithm overview

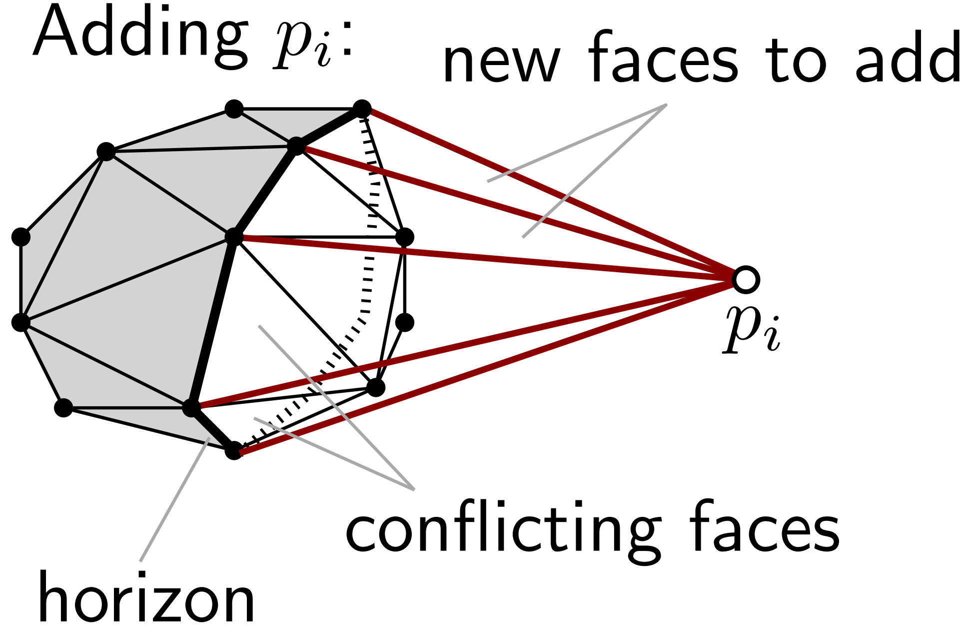
Find 4 points forming a tetrahedron, set up H_4

Randomly permute the other points: $p_4, \ldots, p_n$, and set up C_4

"Randomized incremental construction"

$n - 4$ unproc. pts
4 faces $\Rightarrow O(n)$

Adding p_i:



$\rightarrow$ remove conflicting faces using conflict graph

$\rightarrow$ walk horizon and add new faces

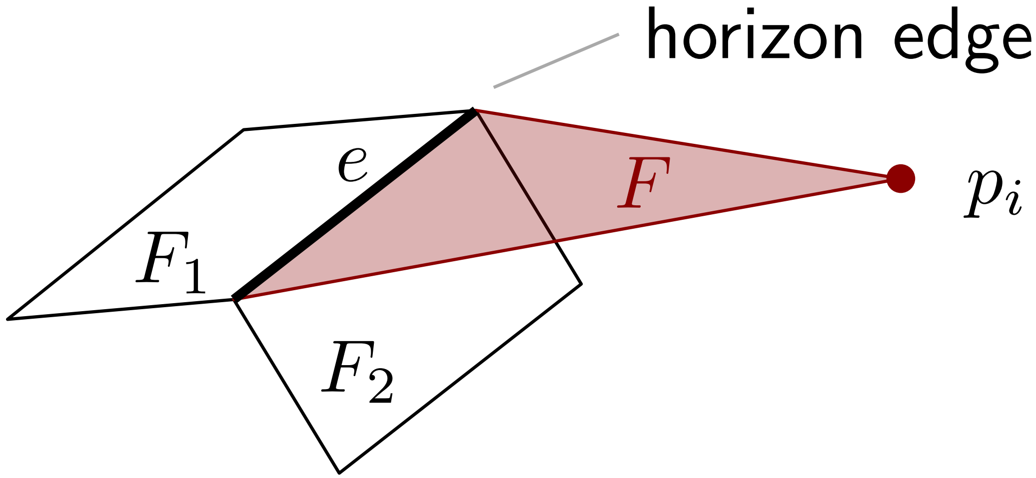
if coplanar face:
$\rightarrow$ merge faces.
same conflict list!

Updating conflict lists

- Remove p_i from conflict graph

Updating conflict lists

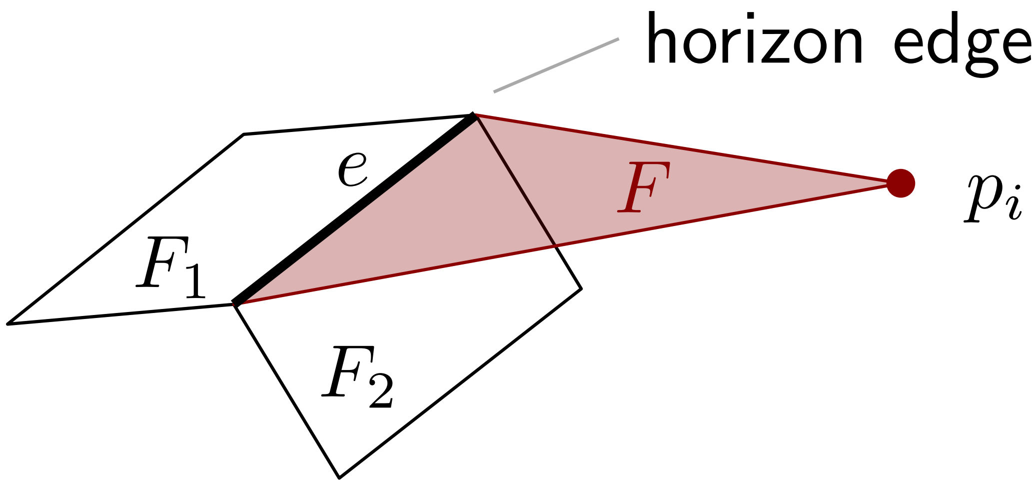
- Remove p_i from conflict graph
- Add new face F:



$$Conf(F) \subseteq Conf(F_1) \cup Conf(F_2)$$

Updating conflict lists

- Remove p_i from conflict graph
- Add new face F:



$$Conf(F) \subseteq \underbrace{Conf(F_1) \cup Conf(F_2)}$$

For each $p \subseteq U_i^e$ that sees F,
add conflict edge (F, p)

Running time analysis - lemmas

Theorem The Clarkson-Shor 3d convex hull algorithm works in $O(n \log n)$ time.

Running time analysis - lemmas

Theorem The Clarkson-Shor 3d convex hull algorithm works in $O(n \log n)$ time.

Lemma The algo. creates at most $6n - 20$ faces *in expectation*.

Running time analysis - lemmas

Theorem The Clarkson-Shor 3d convex hull algorithm works in $O(n \log n)$ time.

Lemma The algo. creates at most $6n - 20$ faces *in expectation*.

Lemma We have

$$\mathbb{E}\left(\sum_{i=5}^{n} \sum_{e \in \mathrm{horizon}(i)} |U_i^e|\right) = O(n \log n).$$

Running time analysis - lemmas

Theorem The Clarkson-Shor 3d convex hull algorithm works in $O(n \log n)$ time.

Lemma The algo. creates at most $6n - 20$ faces *in expectation*.

Long proof, see Dutch book

Lemma We have

$$\mathbb{E}\left(\sum_{i=5}^{n} \sum_{e \in \text{horizon}(i)} |U_i^e|\right) = O(n \log n).$$

Backwards analysis

Lemma The algo. creates at most $6n - 20$ faces *in expectation*.

Backwards analysis

Lemma The algo. creates at most $6n - 20$ faces *in expectation*.

Imagine removing p_n, then $p_{n-1}, \ldots, p_5$ from $\mathrm{conv}(P)$. Fix i.
$\deg^i(p)$:= degree of p in the convex hull of $p_1, \ldots, p_i$

Backwards analysis

Lemma The algo. creates at most $6n - 20$ faces *in expectation*.

Imagine removing p_n, then $p_{n-1}, \ldots, p_5$ from $\mathrm{conv}(P)$. Fix i.

$\deg^i(p)$:= degree of p in the convex hull of $p_1, \ldots, p_i$

- $\mathrm{conv}(p_1, \ldots, p_i)$ has $\leq 3i - 6$ edges

  $\Rightarrow \sum_{j=1}^{i} \deg^i(p_j) \leq 6i - 12$.

Backwards analysis

Lemma The algo. creates at most $6n-20$ faces *in expectation*.

Imagine removing p_n, then $p_{n-1},\dots,p_5$ from $\operatorname{conv}(P)$. Fix i.

$\deg^i(p)$:= degree of p in the convex hull of $p_1,\dots,p_i$

- $\operatorname{conv}(p_1,\dots,p_i)$ has $\leq 3i-6$ edges
  $\Rightarrow \sum_{j=1}^{i} \deg^i(p_j) \leq 6i-12.$
- $\deg(p_1)+\deg(p_2)+\deg(p_3)+\deg(p_4) \geq 12$

Backwards analysis

Lemma The algo. creates at most $6n-20$ faces *in expectation*.

Imagine removing p_n, then $p_{n-1}, \ldots, p_5$ from $\mathrm{conv}(P)$. Fix i.

$\deg^i(p)$:= degree of p in the convex hull of $p_1, \ldots, p_i$

- $\mathrm{conv}(p_1, \ldots, p_i)$ has $\leq 3i-6$ edges
  $\Rightarrow \sum_{j=1}^{i} \deg^i(p_j) \leq 6i-12.$
- $\deg(p_1) + \deg(p_2) + \deg(p_3) + \deg(p_4) \geq 12$
- p_i is a random element of $\{p_5, \ldots, p_i\}$

$$\mathbb{E}(\deg^i(p_i)) = \frac{\sum_{j=5}^{i} \deg(p_j)}{i-4} \leq \frac{6i-12-12}{i-4} = 6$$

Backwards analysis

Lemma The algo. creates at most $6n-20$ faces *in expectation*.

Imagine removing p_n, then $p_{n-1}, \ldots, p_5$ from $\mathrm{conv}(P)$. Fix i.
$\deg^i(p)$:= degree of p in the convex hull of $p_1, \ldots, p_i$

- $\mathrm{conv}(p_1, \ldots, p_i)$ has $\leq 3i-6$ edges
  $\Rightarrow \sum_{j=1}^{i} \deg^i(p_j) \leq 6i-12.$
- $\deg(p_1) + \deg(p_2) + \deg(p_3) + \deg(p_4) \geq 12$
- p_i is a random element of $\{p_5, \ldots, p_i\}$

$$\mathbb{E}(\deg^i(p_i)) = \frac{\sum_{j=5}^{i} \deg(p_j)}{i-4} \leq \frac{6i-12-12}{i-4} = 6$$

$$\Rightarrow \mathbb{E}(\text{total \#created faces}) = 4 + \sum_{j=5}^{n} \mathbb{E}(\deg^j(p_j)) \leq 6n-20.$$

□

Chan's algorithm in $\mathbb{R}^3$

Chan's algorithm recap and changes

P into size m groups P_j $(j = 1, \ldots, \lceil n/m \rceil)$

Precompute each $\mathrm{conv}(P_j)$ in $O(m \log m)$

Chan's algorithm recap and changes

P into size m groups P_j $(j = 1, \ldots, \lceil n/m \rceil)$

Precompute each $\mathrm{conv}(P_j)$ in $O(m \log m)$

Run modified gift wrapping for m steps:

- Find tangent q^j in each P_j in $O(\log m)$
- Wrap to largest angle tangent among q^j

Chan's algorithm recap and changes

P into size m groups P_j $(j = 1, \ldots, \lceil n/m \rceil)$

Precompute each $\mathrm{conv}(P_j)$ in $O(m \log m)$

Run modified gift wrapping for m steps:

- Find tangent q^j in each P_j in $O(\log m)$
- Wrap to largest angle tangent among q^j

$i = 1, \ldots, \lceil \log \log n \rceil$

$m = 2^{2^i}$

Chan's algorithm recap and changes

P into size m groups P_j $(j = 1, \ldots, \lceil n/m \rceil)$

Precompute each $\mathrm{conv}(P_j)$ in $O(m \log m)$

~~Graham's scan~~
Preparata–Hong

Run modified gift wrapping for m steps:

Find tangent q^j in each P_j in $O(\log m)$

Wrap to largest angle tangent among q^j

$i = 1, \ldots, \lceil \log \log n \rceil$

$m = 2^{2^i}$

Chan's algorithm recap and changes

P into size m groups P_j $(j = 1, \ldots, \lceil n/m \rceil)$

Precompute each $\mathrm{conv}(P_j)$ in $O(m \log m)$

~~Graham's scan~~
Preparata–Hong

$i = 1, \ldots, \lceil \log \log n \rceil$
$m = 2^{2^i}$

Run modified gift wrapping for m steps:

- Find tangent q^j in each P_j in $O(\log m)$
- Wrap to largest angle tangent among q^j

Wrap done on edge e as in cylinder of Preparata–Hong
BFS on faces: new face $\to$ find neighboring edges
BFS has $|E(H^*)| = |E(H)| \leq 3h - 6$ steps

Chan's algorithm recap and changes

P into size m groups P_j $(j = 1, \ldots, \lceil n/m \rceil)$

Precompute each $\mathrm{conv}(P_j)$ in $O(m \log m)$

~~Graham's scan~~
Preparata–Hong

Run modified gift wrapping for m steps:

Find tangent q^j in each P_j in $O(\log m)$

Wrap to largest angle tangent among q^j

$i = 1, \ldots, \lceil \log \log n \rceil$
$m = 2^{2^i}$

Wrap done on edge e as in cylinder of Preparata–Hong
BFS on faces: new face $\rightarrow$ find neighboring edges
BFS has $|E(H^*)| = |E(H)| \leq 3h - 6$ steps

Needs Dobkin–Kirkpatrick hierarchical representation for P_j
$\rightarrow$ computing in $O(n)$ coming up!

Chan's algorithm recap and changes

P into size m groups P_j $(j = 1, \ldots, \lceil n/m \rceil)$

Precompute each $\mathrm{conv}(P_j)$ in $O(m \log m)$

Run modified gift wrapping for m steps:

- Find tangent q^j in each P_j in $O(\log m)$
- Wrap to largest angle tangent among q^j

$i = 1, \ldots, \lceil \log \log \sout{n} \rceil$

$m = 2^{2^i}$

~~Graham's scan~~ Preparata–Hong

$3n - 6$

Wrap done on edge e as in cylinder of Preparata–Hong

BFS on faces: new face $\to$ find neighboring edges

BFS has $|E(H^*)| = |E(H)| \leq 3h - 6$ steps

Needs Dobkin–Kirkpatrick hierarchical representation for P_j

$\to$ computing in $O(n)$ coming up!

Chan's algorithm recap and changes

P into size m groups P_j $(j = 1, \ldots, \lceil n/m \rceil)$

Precompute each $\mathrm{conv}(P_j)$ in $O(m \log m)$

~~Graham's scan~~
Preparata–Hong

$i = 1, \ldots, \lceil \log \log \not{n} \rceil$
$m = 2^{2^i}$
$3n - 6$

Run modified gift wrapping for m steps:

Find tangent q^j in each P_j in $O(\log m)$

Wrap to largest angle tangent among q^j

Wrap done on edge e as in cylinder of Preparata–Hong
BFS on faces: new face $\rightarrow$ find neighboring edges
BFS has $|E(H^*)| = |E(H)| \leq 3h - 6$ steps

Needs Dobkin–Kirkpatrick hierarchical representation for P_j
$\rightarrow$ computing in $O(n)$ coming up!

Running time analysis remains unchanged. $O(n \log h)$

Dobkin–Kirkpatrick hierarchy

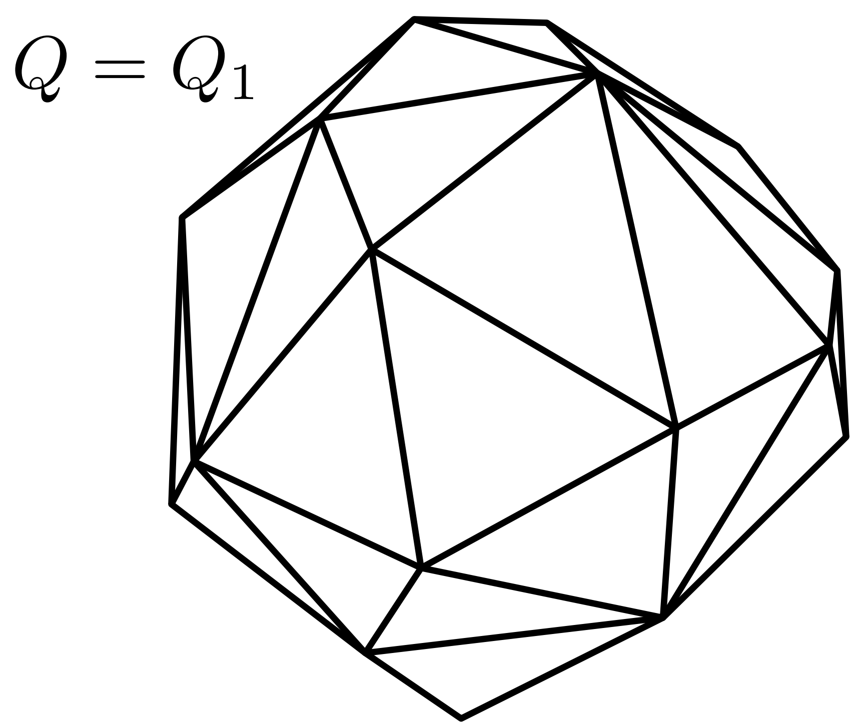
Given convex polytope Q in $\mathbb{R}^3$, a polytope sequence $Q_1, Q_2, \ldots, Q_k$ is a DK hierarchy of Q if

1. $Q_1 = Q$ and Q_k is a tetrahedron
2. $Q_i \supset Q_{i+1}$ and $V(Q_i) \supset V(Q_{i+1})$
3. $V(Q_i) \setminus V(Q_{i+1})$ is an independent set in $G(Q_i)$.

Dobkin–Kirkpatrick hierarchy

Given convex polytope Q in $\mathbb{R}^3$, a polytope sequence $Q_1, Q_2, \ldots, Q_k$ is a DK hierarchy of Q if

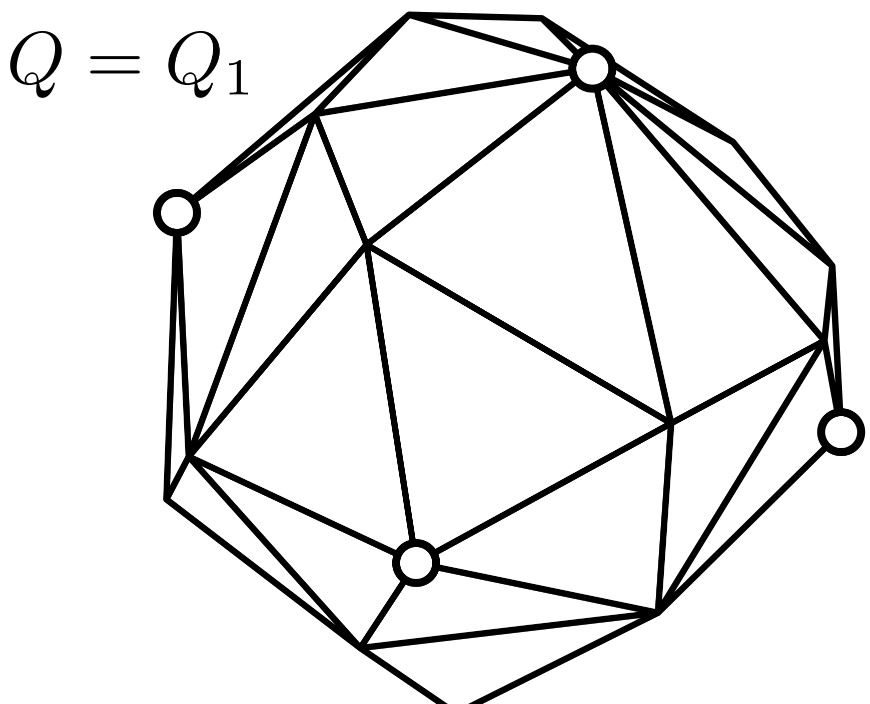
1. $Q_1 = Q$ and Q_k is a tetrahedron
2. $Q_i \supset Q_{i+1}$ and $V(Q_i) \supset V(Q_{i+1})$
3. $V(Q_i) \setminus V(Q_{i+1})$ is an independent set in $G(Q_i)$.



Dobkin–Kirkpatrick hierarchy

Given convex polytope Q in $\mathbb{R}^3$, a polytope sequence $Q_1, Q_2, \ldots, Q_k$ is a DK hierarchy of Q if

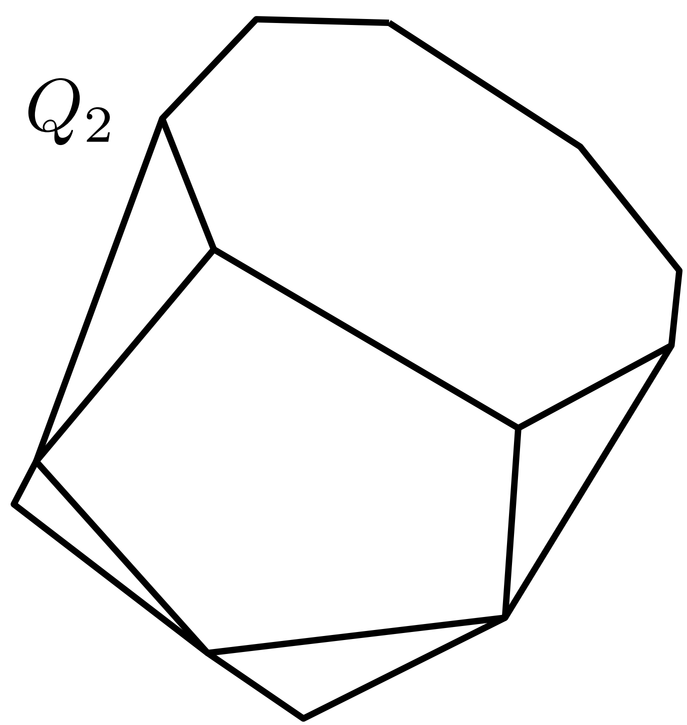
1. $Q_1 = Q$ and Q_k is a tetrahedron
2. $Q_i \supset Q_{i+1}$ and $V(Q_i) \supset V(Q_{i+1})$
3. $V(Q_i) \setminus V(Q_{i+1})$ is an independent set in $G(Q_i)$.



Dobkin–Kirkpatrick hierarchy

Given convex polytope Q in $\mathbb{R}^3$, a polytope sequence $Q_1, Q_2, \ldots, Q_k$ is a DK hierarchy of Q if

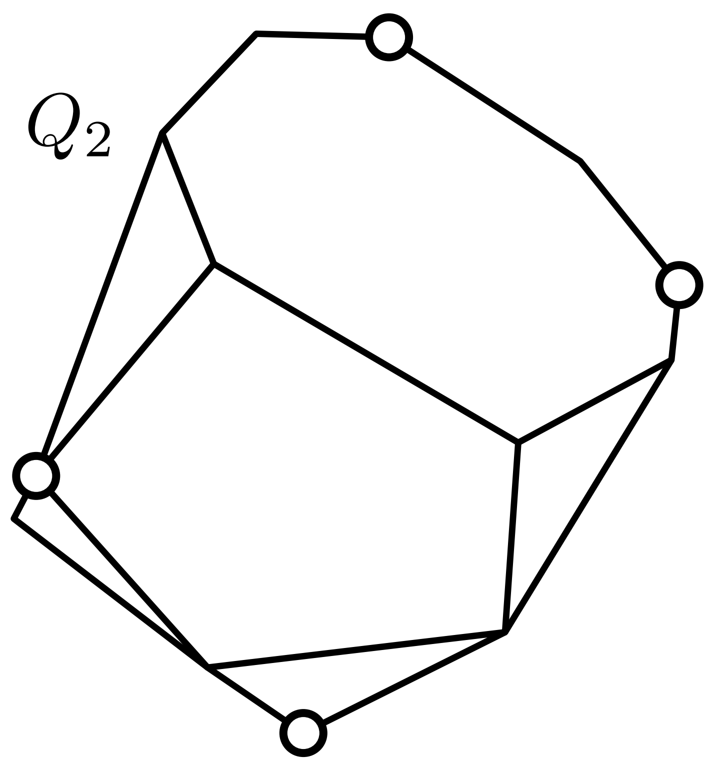
1. $Q_1 = Q$ and Q_k is a tetrahedron
2. $Q_i \supset Q_{i+1}$ and $V(Q_i) \supset V(Q_{i+1})$
3. $V(Q_i) \setminus V(Q_{i+1})$ is an independent set in $G(Q_i)$.



Dobkin–Kirkpatrick hierarchy

Given convex polytope Q in $\mathbb{R}^3$, a polytope sequence $Q_1, Q_2, \ldots, Q_k$ is a DK hierarchy of Q if

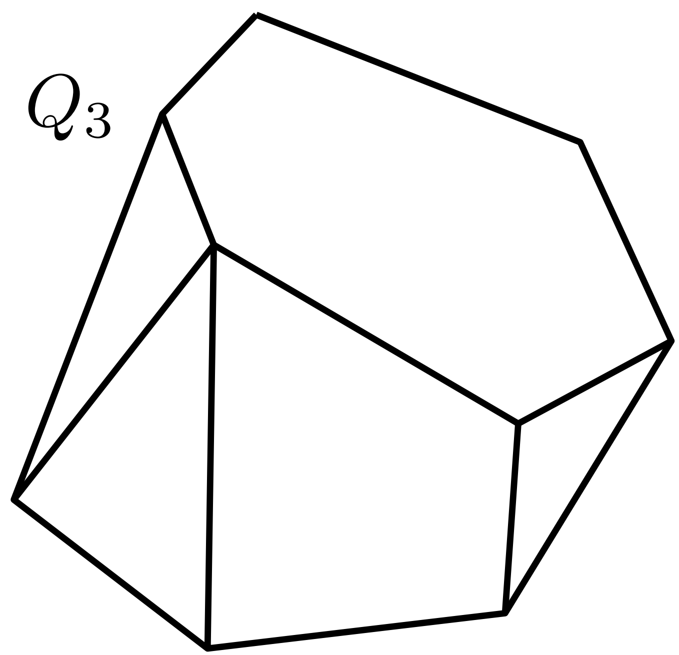
1. $Q_1 = Q$ and Q_k is a tetrahedron
2. $Q_i \supset Q_{i+1}$ and $V(Q_i) \supset V(Q_{i+1})$
3. $V(Q_i) \setminus V(Q_{i+1})$ is an independent set in $G(Q_i)$.



Dobkin–Kirkpatrick hierarchy

Given convex polytope Q in $\mathbb{R}^3$, a polytope sequence $Q_1, Q_2, \ldots, Q_k$ is a DK hierarchy of Q if

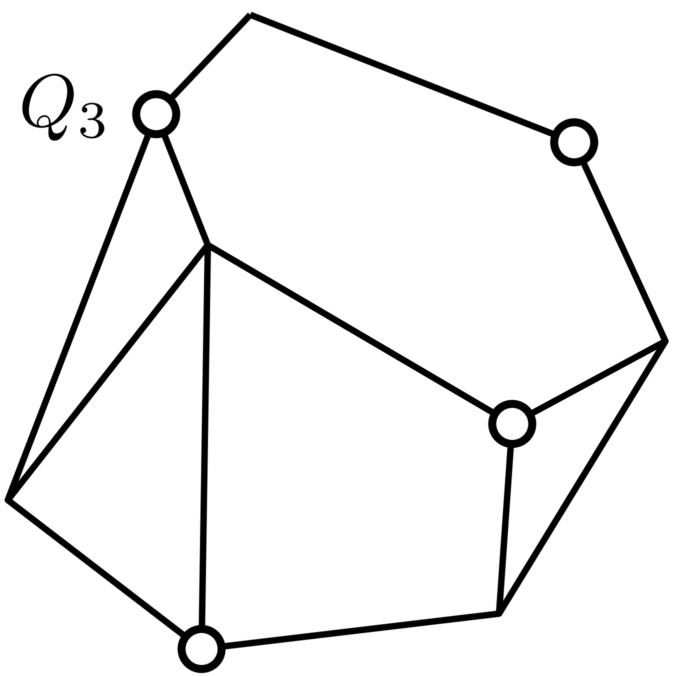
1. $Q_1 = Q$ and Q_k is a tetrahedron
2. $Q_i \supset Q_{i+1}$ and $V(Q_i) \supset V(Q_{i+1})$
3. $V(Q_i) \setminus V(Q_{i+1})$ is an independent set in $G(Q_i)$.



Dobkin–Kirkpatrick hierarchy

Given convex polytope Q in $\mathbb{R}^3$, a polytope sequence $Q_1, Q_2, \ldots, Q_k$ is a DK hierarchy of Q if

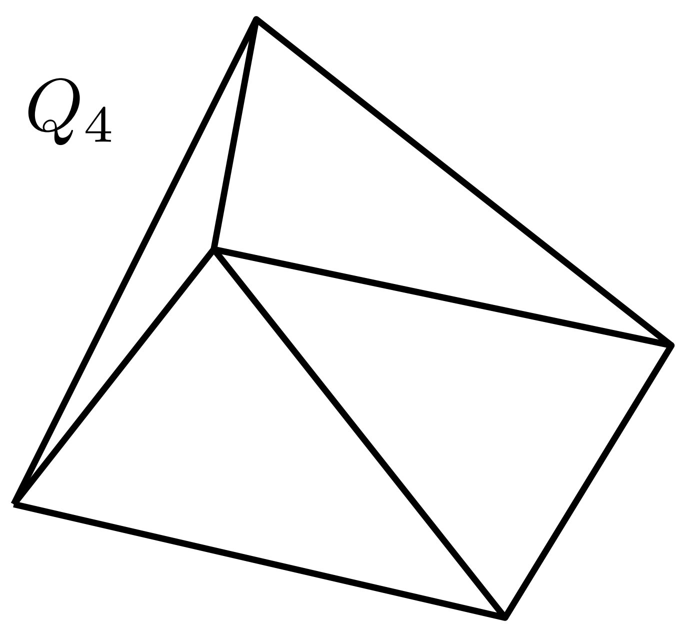
1. $Q_1 = Q$ and Q_k is a tetrahedron
2. $Q_i \supset Q_{i+1}$ and $V(Q_i) \supset V(Q_{i+1})$
3. $V(Q_i) \setminus V(Q_{i+1})$ is an independent set in $G(Q_i)$.



Dobkin–Kirkpatrick hierarchy

Given convex polytope Q in $\mathbb{R}^3$, a polytope sequence $Q_1, Q_2, \ldots, Q_k$ is a DK hierarchy of Q if

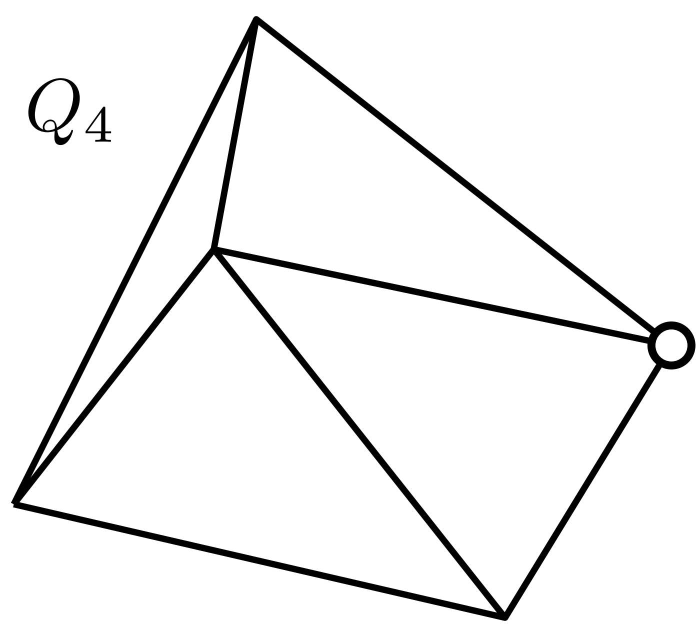
1. $Q_1 = Q$ and Q_k is a tetrahedron
2. $Q_i \supset Q_{i+1}$ and $V(Q_i) \supset V(Q_{i+1})$
3. $V(Q_i) \setminus V(Q_{i+1})$ is an independent set in $G(Q_i)$.



Dobkin–Kirkpatrick hierarchy

Given convex polytope Q in $\mathbb{R}^3$, a polytope sequence $Q_1, Q_2, \ldots, Q_k$ is a DK hierarchy of Q if

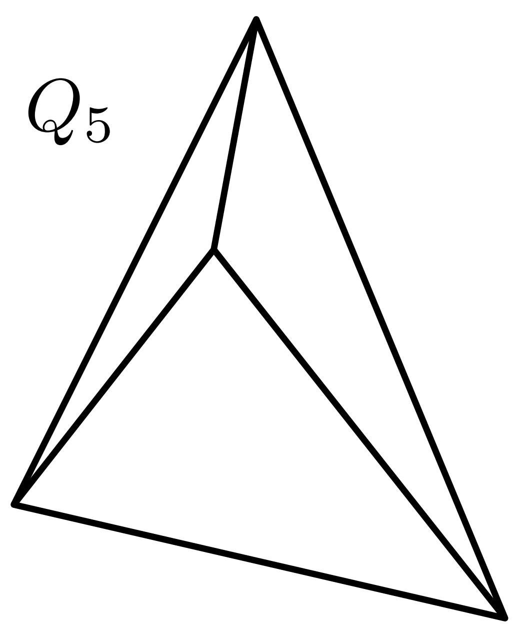
1. $Q_1 = Q$ and Q_k is a tetrahedron
2. $Q_i \supset Q_{i+1}$ and $V(Q_i) \supset V(Q_{i+1})$
3. $V(Q_i) \setminus V(Q_{i+1})$ is an independent set in $G(Q_i)$.



Dobkin–Kirkpatrick hierarchy

Given convex polytope Q in $\mathbb{R}^3$, a polytope sequence $Q_1, Q_2, \ldots, Q_k$ is a DK hierarchy of Q if

1. $Q_1 = Q$ and Q_k is a tetrahedron
2. $Q_i \supset Q_{i+1}$ and $V(Q_i) \supset V(Q_{i+1})$
3. $V(Q_i) \setminus V(Q_{i+1})$ is an independent set in $G(Q_i)$.



Constructing the DK hierarchy

Theorem Given Q, a DK hierarchy with $k = O(\log n)$, size $\sum_{i=1}^{k}(|V(Q_i)|) = O(n)$ and degree

$$\max_i \max\{\deg_{G(Q_i)}(v) \mid v \in V(Q_i) \setminus V(Q_{i+1})\} \leq 11$$

can be computed in $O(n)$ time.

Constructing the DK hierarchy

Theorem Given Q, a DK hierarchy with $k = O(\log n)$, size $\sum_{i=1}^{k}(|V(Q_i)|) = O(n)$ and degree

$$\max_i \max\{\deg_{G(Q_i)}(v) \mid v \in V(Q_i) \setminus V(Q_{i+1})\} \leq 11$$

can be computed in $O(n)$ time.

Proof. Iteratively remove set S, a greedy maximal independent set among vertices of degree ≤ 11.

Claim: $|S| \geq |V(Q)|/24$.
Suppose not: $|S| < |V(Q)|/24$
$\Rightarrow \bigcup_{s \in S} N[s] < |V(Q)|/2$
$\Rightarrow G(Q)$ has $\geq |V(Q)|/2$ vertices of degree ≥ 12
$\Rightarrow G(Q)$ has $\geq (|V(Q)|/2) \cdot 12/2 = 3|V(Q)|$ edges

Constructing the DK hierarchy

Theorem Given Q, a DK hierarchy with $k = O(\log n)$, size $\sum_{i=1}^{k}(|V(Q_i)|) = O(n)$ and degree

$$\max_i \max\{\deg_{G(Q_i)}(v) \mid v \in V(Q_i) \setminus V(Q_{i+1})\} \leq 11$$

can be computed in $O(n)$ time.

Proof. Iteratively remove set S, a greedy maximal independent set among vertices of degree ≤ 11.

Claim: $|S| \geq |V(Q)|/24$.
Suppose not: $|S| < |V(Q)|/24$
$\Rightarrow \bigcup_{s \in S} N[s] < |V(Q)|/2$
$\Rightarrow G(Q)$ has $\geq |V(Q)|/2$ vertices of degree ≥ 12
$\Rightarrow G(Q)$ has $\geq (|V(Q)|/2) \cdot 12/2 = 3|V(Q)|$ edges

Euler's formula: $E(Q) \leq 3|V(Q)| - 6$

□

for $i = 1$ to $\lceil \log\log(3n-6) \rceil$ **do**
  $m = 2^{2^i}$
  for $j = 1$ to $\lceil n/m \rceil$ **do**
    Create group P_j
    $H_j =$Preparata–Hong(P_j)
    Compute dual D–K hierarchy of H_j
  $F_0 =$ starting face, e_1, e_2, e_3: ccw arcs of $E(F)$,
  $H.add(F_0)$, $Queue = (e_1,e_2,e_3)$, $Seen = \{e_1,e_2,e_3\}$,
  for $s = 2$ to m **do**
    $e = Queue.next$
    for $j = 1$ to $\lceil n/m \rceil$ **do**
      $q^j = TangentFind(e, dualDK(H_j))$
    $q =$ point q^j maximizing $\sphericalangle\big((\text{plane } e, q^j), F(e)\big)$
    $H.add$(face e, q), $e', e'' =$ next arcs of face e, q
    if $e' \notin Seen$ **then** $Queue.add(e'), Seen.add(e')$
    if $e'' \notin Seen$ **then** $Queue.add(e''), Seen.add(e'')$
return H

```
for i = 1 to ⌈log log(3n − 6)⌉ do
    m = 2^{2^i}
    for j = 1 to ⌈n/m⌉ do
        Create group P_j
        H_j = Preparata–Hong(P_j)
        Compute dual D–K hierarchy of H_j   ⌈n/m⌉ · O(m)
    F_0 = starting face,  e_1, e_2, e_3: ccw arcs of E(F),
    H.add(F_0), Queue = (e_1,e_2,e_3), Seen = {e_1,e_2,e_3},
    for s = 2 to m do
        e = Queue.next
        for j = 1 to ⌈n/m⌉ do
            q^j = TangentFind(e, dualDK(H_j))
        q = point q^j maximizing ∢((plane e, q^j), F(e))
        H.add(face e, q), e′, e″ = next arcs of face e, q
        if e′ ∉ Seen then Queue.add(e′), Seen.add(e′)
        if e″ ∉ Seen then Queue.add(e″), Seen.add(e″)
return H
```

for $i = 1$ to $\lceil \log\log(3n-6) \rceil$ **do**

  $m = 2^{2^i}$

  for $j = 1$ to $\lceil n/m \rceil$ **do**

    Create group P_j

    $H_j =$Preparata–Hong(P_j)

    Compute dual D–K hierarchy of H_j $\lceil n/m \rceil \cdot O(m)$

  $F_0 =$ starting face, e_1, e_2, e_3: ccw arcs of $E(F)$,

  $H.add(F_0)$, $Queue = (e_1,e_2,e_3)$, $Seen = \{e_1,e_2,e_3\}$,

  for $s = 2$ to m **do**

Self-balancing BST

max size$= h$

    $e = Queue.next$

    for $j = 1$ to $\lceil n/m \rceil$ **do**

      $q^j = TangentFind(e, dualDK(H_j))$

    $q =$ point q^j maximizing $\sphericalangle\big((\text{plane } e, q^j), F(e)\big)$

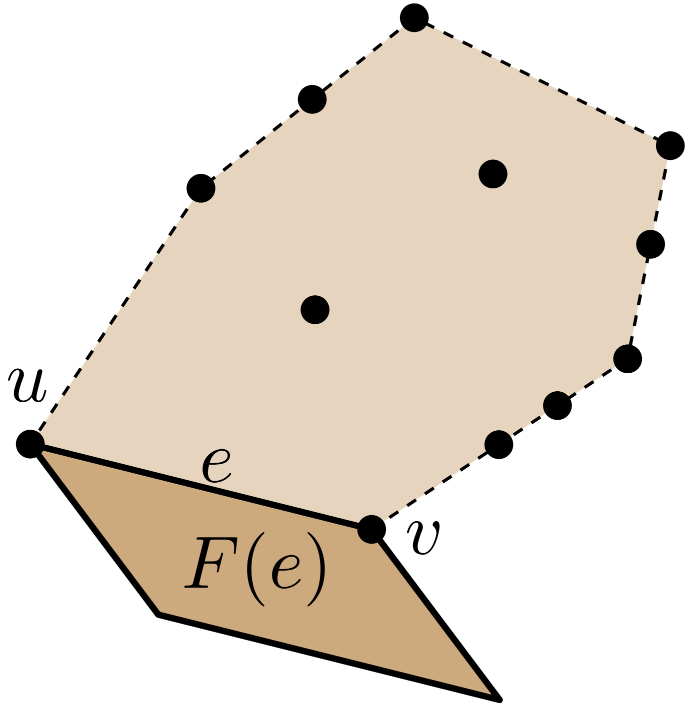
    $H.add$(face e, q), $e', e'' =$ next arcs of face e, q

    if $e' \notin Seen$ **then** $Queue.add(e'), Seen.add(e')$

    if $e'' \notin Seen$ **then** $Queue.add(e''), Seen.add(e'')$

return H

Handling degeneracies

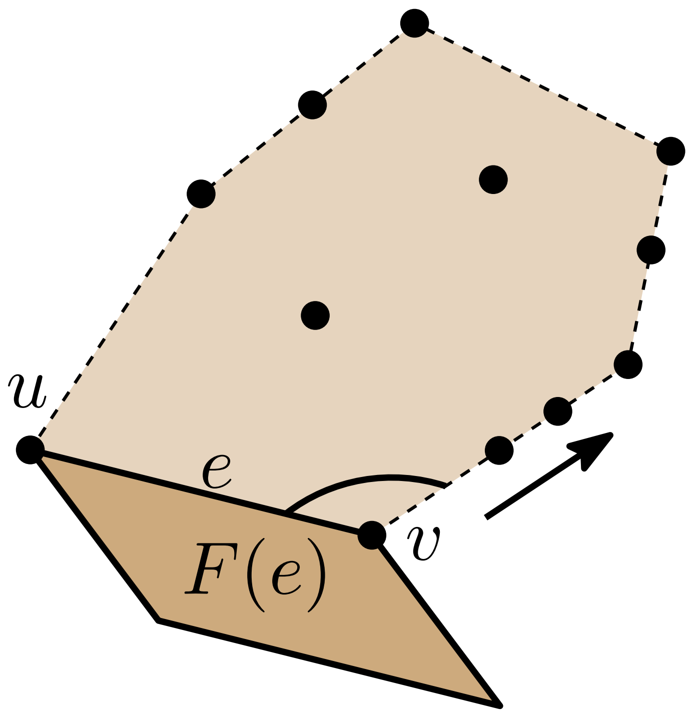


Handling degeneracies

Among $Q = \operatorname{argmax}_{q'} \left(\sphericalangle F(e), (\text{plane } uvq')\right)$

q should maximze $\sphericalangle(uvq)$

and among these maximize $\operatorname{dist}(v, q)$

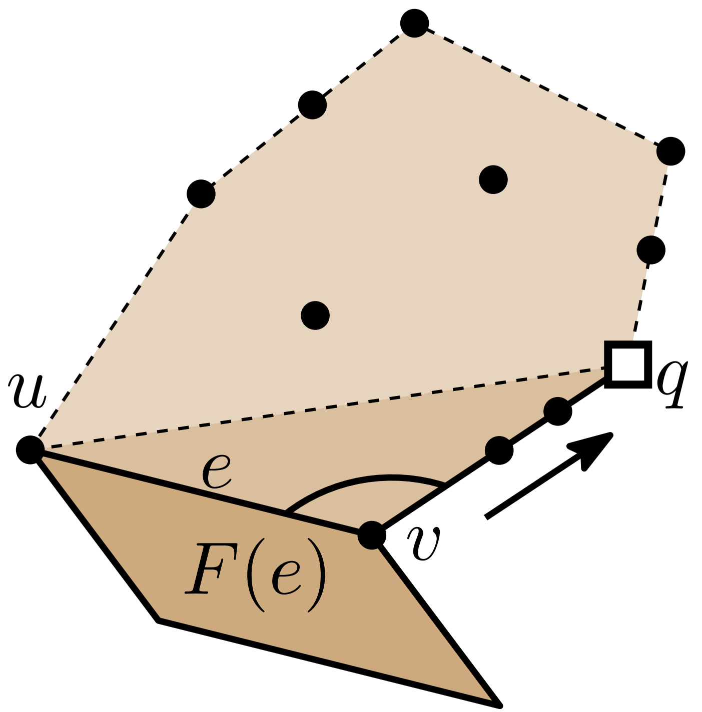


Handling degeneracies

Among $Q = \operatorname{argmax}_{q'} \left(\sphericalangle F(e), (\text{plane } uvq')\right)$

q should maximze $\sphericalangle(uvq)$

and among these maximize $\operatorname{dist}(v, q)$

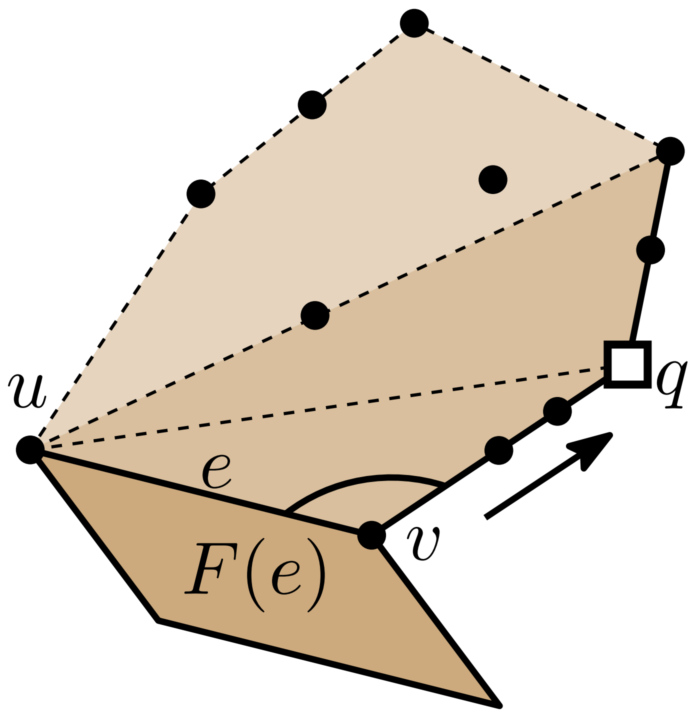


Handling degeneracies

Among $Q = \operatorname{argmax}_{q'} \left(\sphericalangle F(e), (\text{plane } uvq')\right)$

q should maximze $\sphericalangle(uvq)$

and among these maximize $\operatorname{dist}(v, q)$

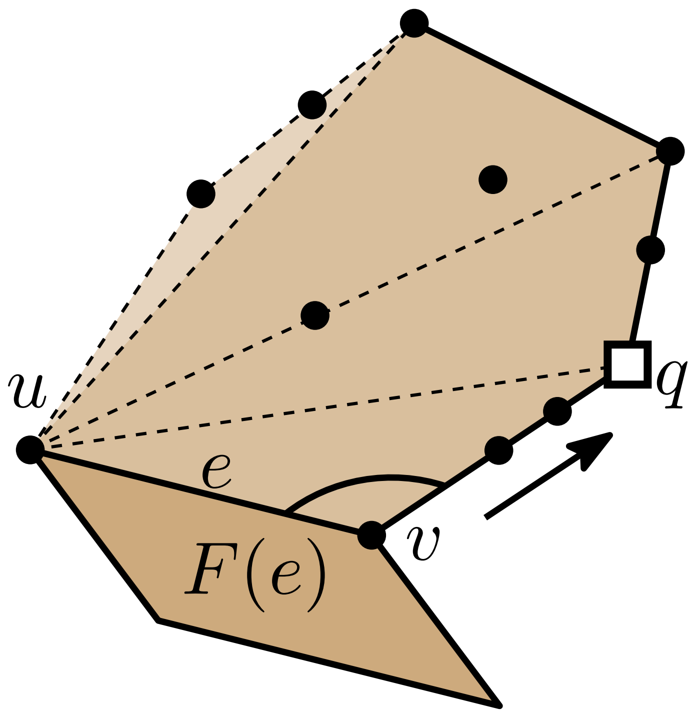


Handling degeneracies

Among $Q = \operatorname{argmax}_{q'} \left(\sphericalangle F(e), (\text{plane } uvq')\right)$

q should maximze $\sphericalangle(uvq)$

and among these maximize $\operatorname{dist}(v, q)$

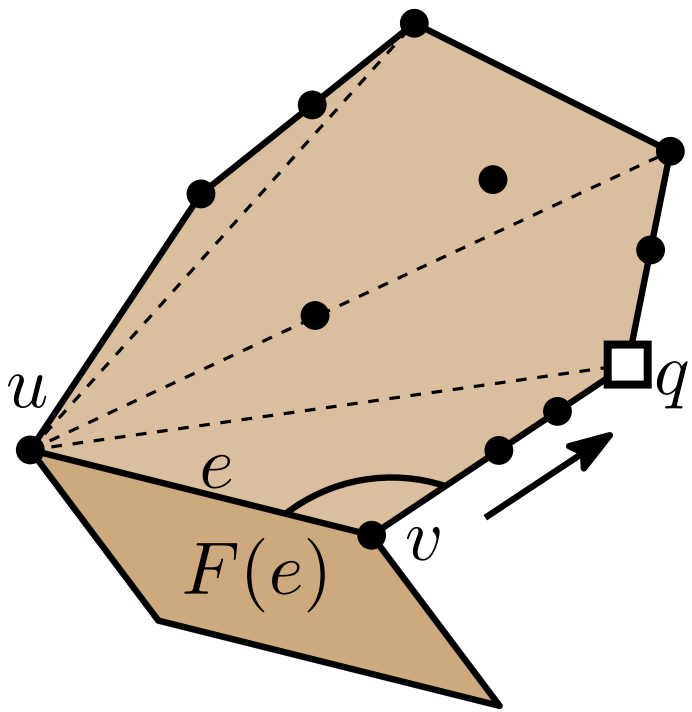


Handling degeneracies

Among $Q = \mathrm{argmax}_{q'}\left(\sphericalangle F(e), (\text{plane } uvq')\right)$

q should maximze $\sphericalangle(uvq)$

and among these maximize $\mathrm{dist}(v, q)$



H is a triangulation of the hull.

Postprocess: merge at edge if neighboring faces are coplanar

$(\rightarrow O(n))$

Higher-dimensional convex hulls

Many faces, many facets

In $\mathbb{R}^d$, we have $0, 1, \ldots, (d-1)$-dimensional faces.

vertices   edges   facets

Many faces, many facets

In $\mathbb{R}^d$, we have $0, 1, \ldots, (d-1)$-dimensional faces.

vertices edges facets

Given a set P of n points in $\mathbb{R}^d$, compute:

(a) all facets of $\mathrm{conv}(P)$

(b) all vertices of $\mathrm{conv}(P)$

(c) all faces of $\mathrm{conv}(P)$

Many faces, many facets

In $\mathbb{R}^d$, we have $0, 1, \dots, (d-1)$-dimensional faces.

vertices edges facets

Given a set P of n points in $\mathbb{R}^d$, compute:
(a) all facets of $\mathrm{conv}(P)$
(b) all vertices of $\mathrm{conv}(P)$
(c) all faces of $\mathrm{conv}(P)$

There can be up to $\Theta(n^{\lfloor d/2 \rfloor})$ facets!

Many faces, many facets

In $\mathbb{R}^d$, we have $0, 1, \ldots, (d-1)$-dimensional faces.

vertices edges facets

Given a set P of n points in $\mathbb{R}^d$, compute:
(a) all facets of $\mathrm{conv}(P)$
(b) all vertices of $\mathrm{conv}(P)$
(c) all faces of $\mathrm{conv}(P)$

There can be up to $\Theta(n^{\lfloor d/2 \rfloor})$ facets!

Let P be distinct points on the moment curve

$$\{(x, x^2, x^3, \ldots, x^d) \mid x \in \mathbb{R}\}.$$

Then $\mathrm{conv}(P)$ has the maximum number of k-faces for all $k \in [d]$.

Many faces, many facets

In $\mathbb{R}^d$, we have $0, 1, \ldots, (d-1)$-dimensional faces.

vertices edges facets

Given a set P of n points in $\mathbb{R}^d$, compute:
(a) all facets of $\operatorname{conv}(P)$
(b) all vertices of $\operatorname{conv}(P)$
(c) all faces of $\operatorname{conv}(P)$

There can be up to $\Theta(n^{\lfloor d/2 \rfloor})$ facets!

Let P be distinct points on the moment curve

$$\{(x, x^2, x^3, \ldots, x^d) \mid x \in \mathbb{R}\}.$$

Then $\operatorname{conv}(P)$ has the maximum number of k-faces for all $k \in [d]$.

$O(n \log n + n^{\lfloor d/2 \rfloor})$ achieved by many algorithms