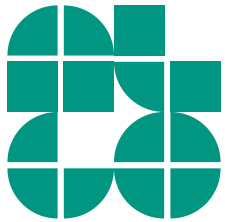
Vorlesung Algorithmische Geometrie

Konvexe Hülle im $\mathbb{R}^3$

LEHRSTUHL FÜR ALGORITHMIK I · INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Andreas Gemsa

10.05.2011



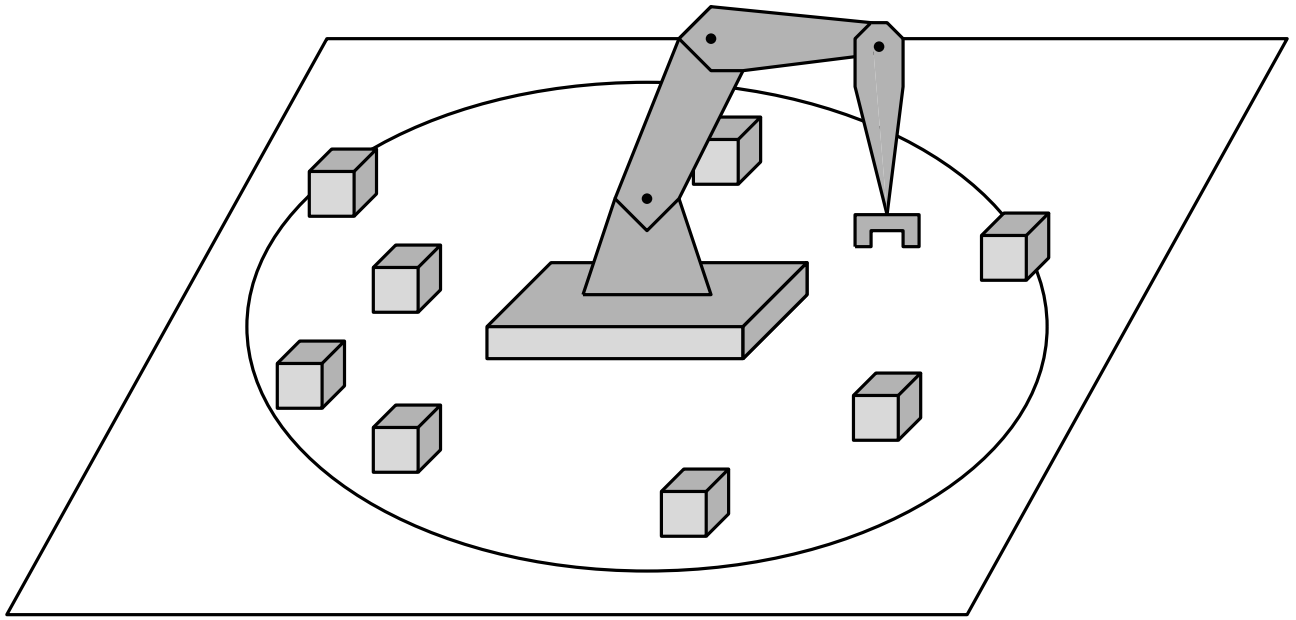
Ablauf

Karlsruhe Institute of Technology

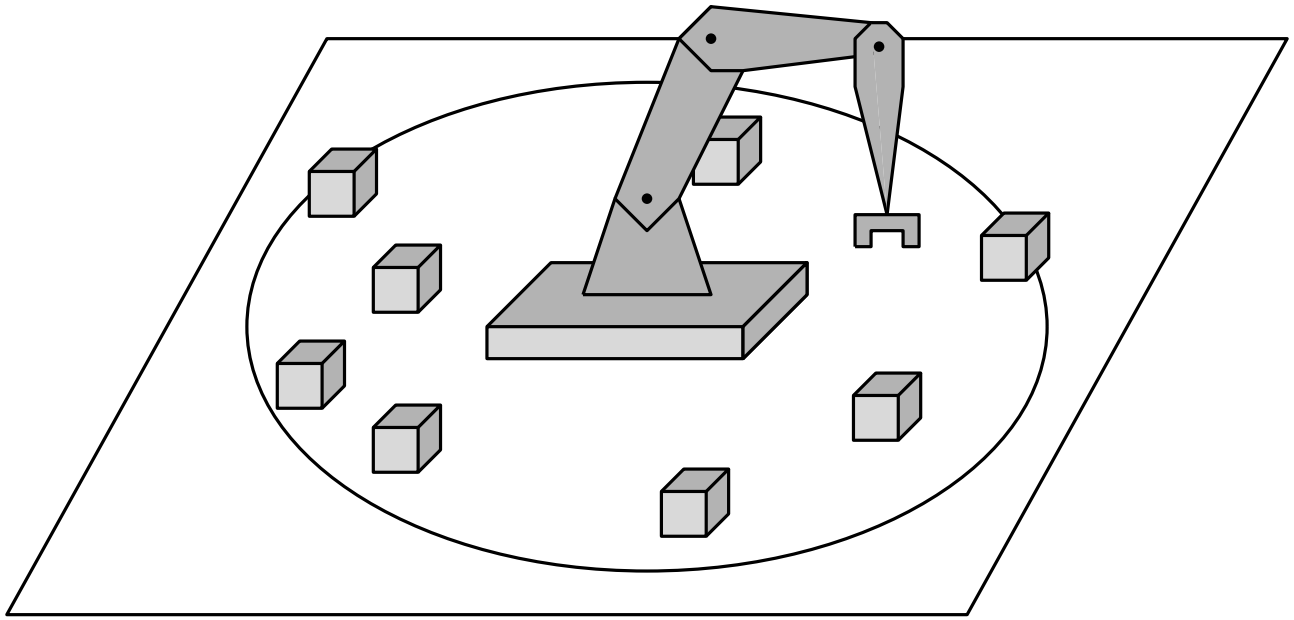
1. Smallest Enclosing Disk

2. Konvexe Hüllen im $\mathbb{R}^3$

Problem

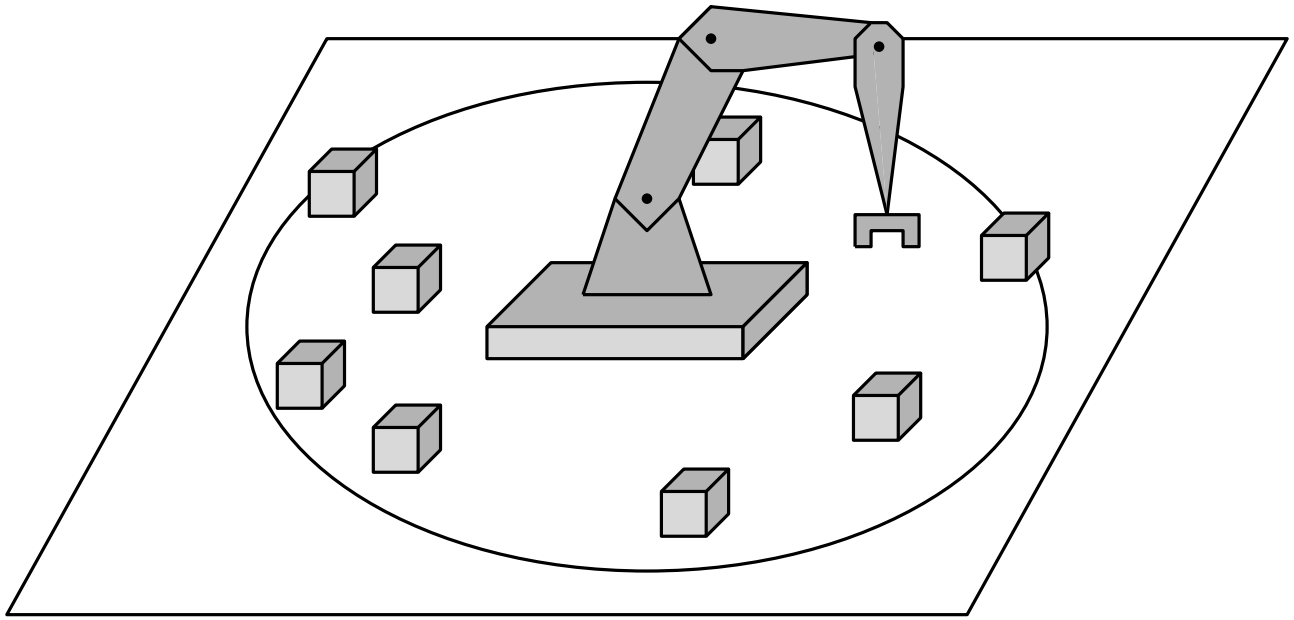


Problem



Wie findet man eine gute Position für den Roborterarm?

Problem



Wie findet man eine gute Position für den Roborterarm?

Smallest Enclosing Disc

Smallest Enclosing Disk

Def.: Sei P eine n-elementige Menge von Punkten im $\mathbb{R}^2$. Die smallest enclosing disc für ist P ist der Kreis der alle Punkte aus P enthält und minimalen Radius hat.

Smallest Enclosing Disk

Def.: Sei P eine n-elementige Menge von Punkten im $\mathbb{R}^2$. Die smallest enclosing disc für ist P ist der Kreis der alle Punkte aus P enthält und minimalen Radius hat.

Dieses mal: Ähnlicher Ansatz wie in der letzten Woche (randomisiert + inkrementell)

Smallest Enclosing Disk

Def.: Sei P eine n-elementige Menge von Punkten im $\mathbb{R}^2$. Die smallest enclosing disc für ist P ist der Kreis der alle Punkte aus P enthält und minimalen Radius hat.

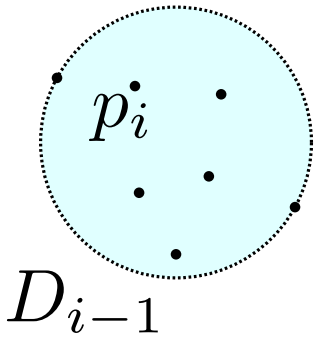
Dieses mal: Ähnlicher Ansatz wie in der letzten Woche (randomisiert + inkrementell)

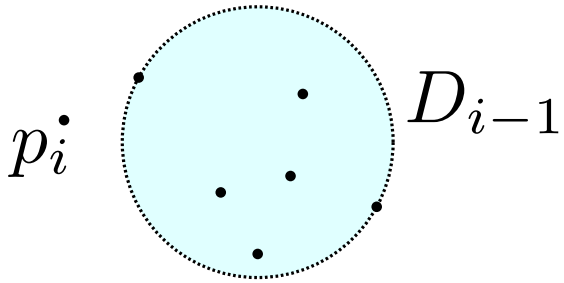
Idee: Erzeuge zufällige Permutation von P. Sei $P_i := \{p_1, \ldots, p_i\}$. Füge dann Punkte nach und nach hinzu. Dabei verwalten wir immer D_i die *smallest enclosing disk*.

Smallest Enclosing Disk

Lemma: Sei $2 < i < n$, und sei P_i und D_i wie gerade dann gilt:

(i) Wenn $p_i \in D_{i-1}$, dann gilt $D_i = D_{i-1}$
(ii) Wenn $p_i \notin D_{i-1}$, dann liegt p_i auf dem Rand von D_i

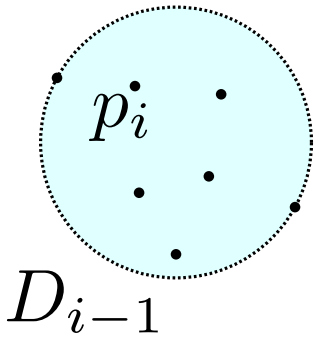


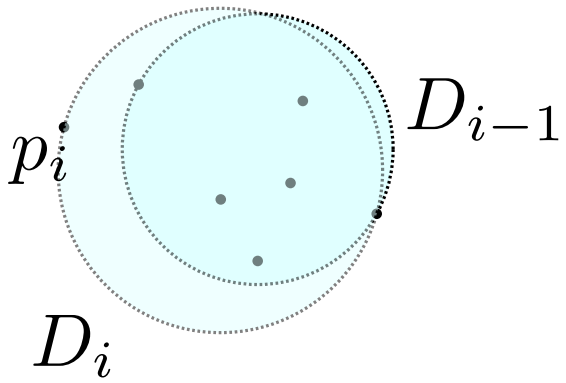


Smallest Enclosing Disk

Lemma: Sei $2 < i < n$, und sei P_i und D_i wie gerade dann gilt:

(i) Wenn $p_i \in D_{i-1}$, dann gilt $D_i = D_{i-1}$
(ii) Wenn $p_i \notin D_{i-1}$, dann liegt p_i auf dem Rand von D_i





Smallest Enclosing Disk

Karlsruhe Institute of Technology

$\textsc{MiniDisc}(P \subset \mathbb{R}^2)$

P $= \textsc{RandomPermutation}(P)$

Sei D_2 die smallest enclosing disc

for $i \leftarrow 3$ bis n **do**

  if $p_i \in D_{i-1}$ **then** $D_i = D_{i-1}$

  else $D_i = \textsc{MiniDiscWithPoint}(\{p_1, \ldots, p_{i-1}\}, p_i)$

return D_i

Smallest Enclosing Disk

$\textsc{MiniDisc}(P \subset \mathbb{R}^2)$

P $= \textsc{RandomPermutation}(P)$

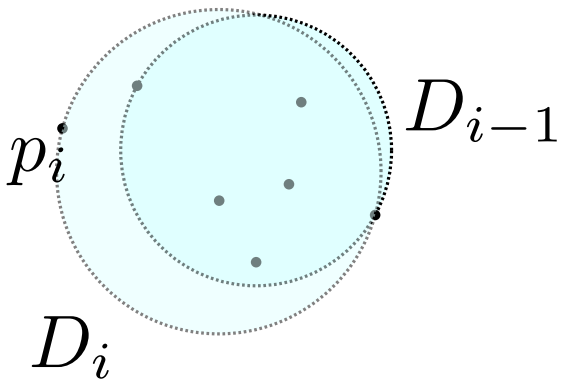
Sei D_2 die smallest enclosing disc

for $i \leftarrow 3$ bis n **do**

  if $p_i \in D_{i-1}$ **then** $D_i = D_{i-1}$

  else $D_i = \textsc{MiniDiscWithPoint}(\{p_1, \ldots, p_{i-1}\}, p_i)$

return D_i



Smallest Enclosing Disk

$\textsc{MiniDisc}(P \subset \mathbb{R}^2)$

P $= \textsc{RandomPermutation}(P)$

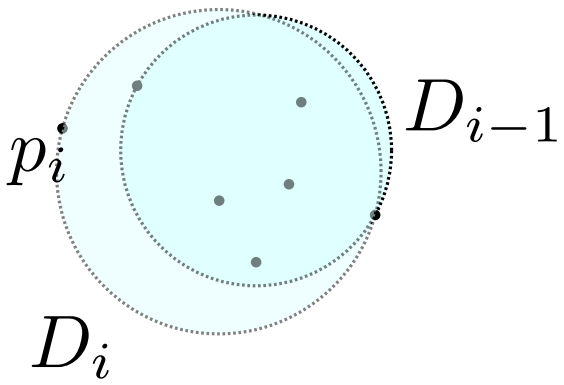
Sei D_2 die smallest enclosing disc

for $i \leftarrow 3$ bis n **do**

  if $p_i \in D_{i-1}$ **then** $D_i = D_{i-1}$

  else $D_i = \textsc{MiniDiscWithPoint}(\{p_1, \ldots, p_{i-1}\}, p_i)$

return D_i



Smallest Enclosing Disk

$\textsc{MiniDiscWithPoint}(\{p_1, \ldots, p_{i-1}\}, q)$

P $= \textsc{RandomPermutation}(\{p_1, \ldots, p_{i-1}\})$

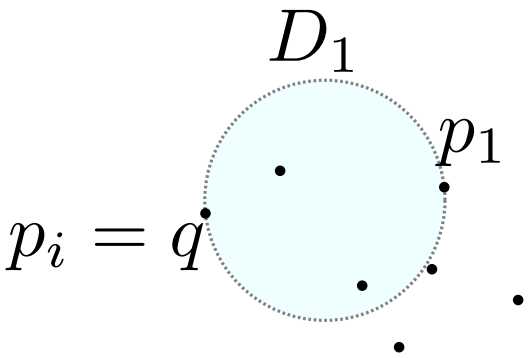
Sei D_1 die smallest enclosing disc die p_1 und q auf dem Rand hat

for $j \leftarrow 2$ bis $i-1$ **do**

  if $p_j \in D_{j-1}$ **then** $D_j = D_{j-1}$

  else $D_j = \textsc{MiniDiscWith2Points}(\{p_1, \ldots, p_{j-1}\}, p_j, q)$

return D_j



Smallest Enclosing Disk

$\textsc{MiniDiscWithPoint}(\{p_1, \ldots, p_{i-1}\}, q)$

P $= \textsc{RandomPermutation}(\{p_1, \ldots, p_{i-1}\})$

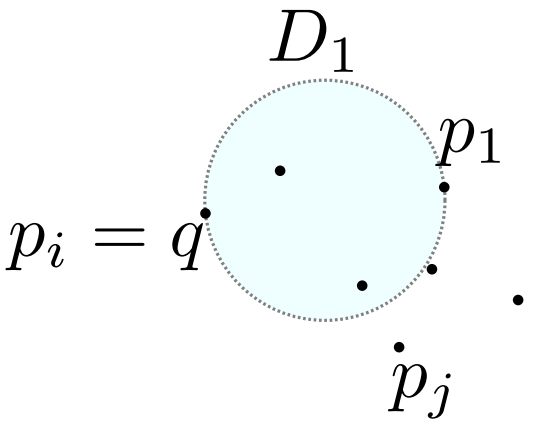
Sei D_1 die smallest enclosing disc die p_1 und q auf dem Rand hat

for $j \leftarrow 2$ bis $i-1$ **do**

  if $p_j \in D_{j-1}$ **then** $D_j = D_{j-1}$

  else $D_j = \textsc{MiniDiscWith2Points}(\{p_1, \ldots, p_{j-1}\}, p_j, q)$

return D_j



Smallest Enclosing Disk

MiniDiscWith2Points$(\{p_1, \ldots, p_{j-1}\}, q_1, q_2)$

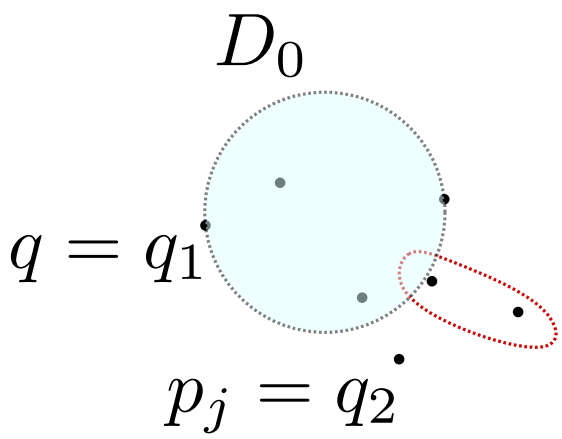
Sei D_0 die smallest enclosing disc bei der q_1 und q_2 auf dem Rand liegen

for $k \leftarrow 1$ to $j-1$ **do**

  if $p_k \in D_{k-1}$ **then** $D_k = D_{k-1}$

  else $D_k =$ disc bei der q_1, q_2 und p_k auf dem Rand liegen

return D_j



Smallest Enclosing Disk

MiniDiscWith2Points$(\{p_1, \ldots, p_{j-1}\}, q_1, q_2)$

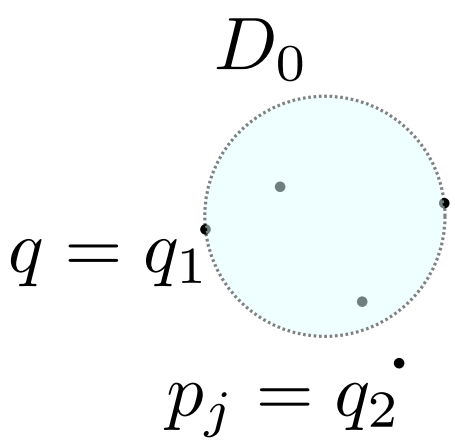
Sei D_0 die smallest enclosing disc bei der q_1 und q_2 auf dem Rand liegen

for $k \leftarrow 1$ to $j-1$ **do**

  if $p_k \in D_{k-1}$ **then** $D_k = D_{k-1}$

  else $D_k =$ disc bei der q_1, q_2 und p_k auf dem Rand liegen

return D_j



Smallest Enclosing Disk

MiniDiscWith2Points$(\{p_1, \ldots, p_{j-1}\}, q_1, q_2)$

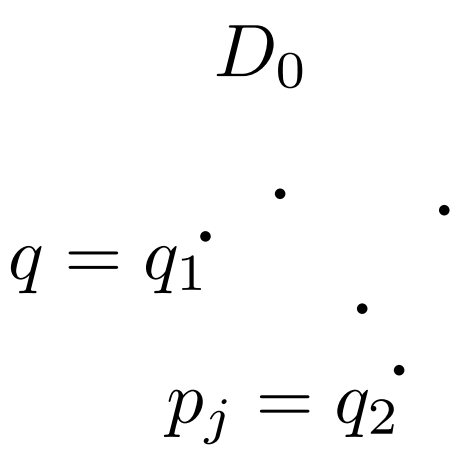
Sei D_0 die smallest enclosing disc bei der q_1 und q_2 auf dem Rand liegen

for $k \leftarrow 1$ to $j-1$ **do**

  if $p_k \in D_{k-1}$ **then** $D_k = D_{k-1}$

  else $D_k =$ disc bei der q_1, q_2 und p_k auf dem Rand liegen

return D_j



Smallest Enclosing Disk

MiniDiscWith2Points($\{p_1, \ldots, p_{j-1}\}, q_1, q_2$)

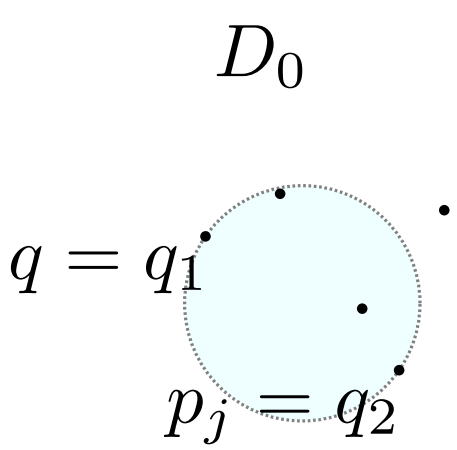
Sei D_0 die smallest enclosing disc bei der q_1 und q_2 auf dem Rand liegen

for $k \leftarrow 1$ to $j-1$ **do**

  if $p_k \in D_{k-1}$ **then** $D_k = D_{k-1}$

  else $D_k =$ disc bei der q_1, q_2 und p_k auf dem Rand liegen

return D_j



Smallest Enclosing Disk

MiniDiscWith2Points$(\{p_1, \ldots, p_{j-1}\}, q_1, q_2)$

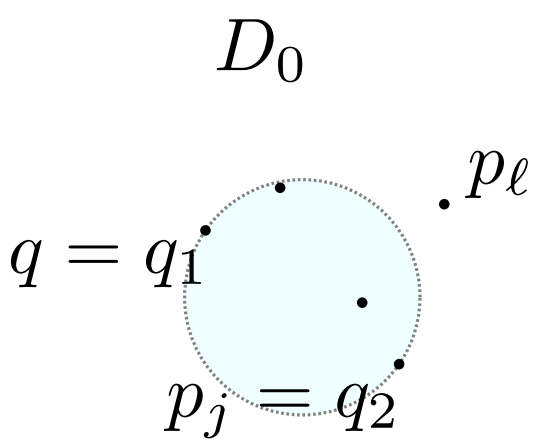
Sei D_0 die smallest enclosing disc bei der q_1 und q_2 auf dem Rand liegen

for $k \leftarrow 1$ to $j-1$ **do**

  if $p_k \in D_{k-1}$ **then** $D_k = D_{k-1}$

  else $D_k =$ disc bei der q_1, q_2 und p_k auf dem Rand liegen

return D_j



Smallest Enclosing Disk

MiniDiscWith2Points($\{p_1, \ldots, p_{j-1}\}, q_1, q_2$)

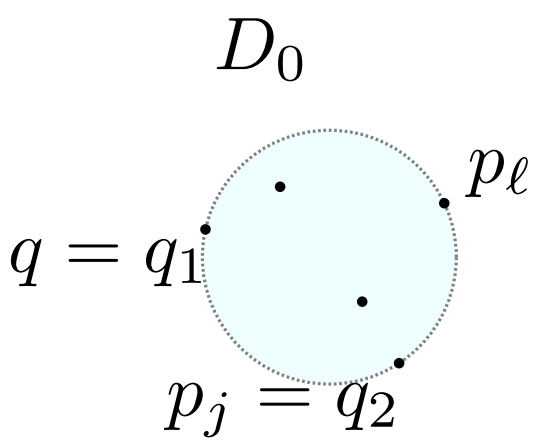
Sei D_0 die smallest enclosing disc bei der q_1 und q_2 auf dem Rand liegen

for $k \leftarrow 1$ to $j-1$ **do**

  if $p_k \in D_{k-1}$ **then** $D_k = D_{k-1}$

  else $D_k =$ disc bei der q_1, q_2 und p_k auf dem Rand liegen

return D_j



Analyse

Theorem: Die smallest enclosing disc für n Punkte in der Ebene kann in erwartet $\mathcal{O}(n)$ berechnet werden.

Analyse

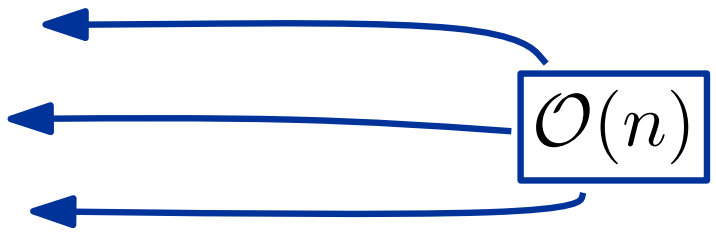
Theorem: Die smallest enclosing disc für n Punkte in der Ebene kann in erwartet $\mathcal{O}(n)$ berechnet werden.

MiniDisc
MiniDiscWithPoint
MiniDiscWith2Points

Analyse

Theorem: Die smallest enclosing disc für n Punkte in der Ebene kann in erwartet $\mathcal{O}(n)$ berechnet werden.

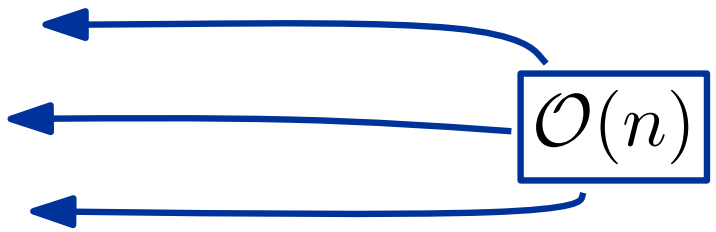
MiniDisc
MiniDiscWithPoint
MiniDiscWith2Points



Analyse

Theorem: Die smallest enclosing disc für n Punkte in der Ebene kann in erwartet $\mathcal{O}(n)$ berechnet werden.

MiniDisc
MiniDiscWithPoint
MiniDiscWith2Points



Wie wahrscheinlich ist es, dass MiniDiscWithPoint MiniDiscWith2Points aufruft?

Analyse

Wie wahrscheinlich ist es, dass MiniDiscWithPoint MiniDiscWith2Points aufruft?

Analyse

Wie wahrscheinlich ist es, dass MiniDiscWithPoint MiniDiscWith2Points aufruft?

Rückwärtsanalyse

Analyse

Wie wahrscheinlich ist es, dass MiniDiscWithPoint MiniDiscWith2Points aufruft?

Rückwärtsanalyse

Idee: Fixiere $P_i := \{p_1, \ldots, p_i\}$ und sei D_i die smallest enclosing disc für P_i. Außerdem liege q auf dem Rand von D_i.

Analyse

Wie wahrscheinlich ist es, dass MiniDiscWithPoint MiniDiscWith2Points aufruft?

Rückwärtsanalyse

Idee: Fixiere $P_i := \{p_1, \ldots, p_i\}$ und sei D_i die smallest enclosing disc für P_i. Außerdem liege q auf dem Rand von D_i.

Entferne nun ein $p_j \in P_i$. Wann ändert sich die smallest enclosing disc?

Analyse

Wie wahrscheinlich ist es, dass MiniDiscWithPoint MiniDiscWith2Points aufruft?

Rückwärtsanalyse

Idee: Fixiere $P_i := \{p_1, \ldots, p_i\}$ und sei D_i die smallest enclosing disc für P_i. Außerdem liege q auf dem Rand von D_i.

Entferne nun ein $p_j \in P_i$. Wann ändert sich die smallest enclosing disc?

- Höchstens 2 aus den i Knoten dafür verantwortlich.

Analyse

Wie wahrscheinlich ist es, dass MiniDiscWithPoint MiniDiscWith2Points aufruft?

Rückwärtsanalyse

Idee: Fixiere $P_i := \{p_1, \ldots, p_i\}$ und sei D_i die smallest enclosing disc für P_i. Außerdem liege q auf dem Rand von D_i.

Entferne nun ein $p_j \in P_i$. Wann ändert sich die smallest enclosing disc?

- Höchstens 2 aus den i Knoten dafür verantwortlich.

$$\mathcal{O}(n) + \sum_{i=2}^{n} \left(\mathcal{O}(i)\tfrac{2}{i}\right) = \mathcal{O}(n)$$

Analyse

Wie wahrscheinlich ist es, dass MiniDisc MiniDiscWithPoint aufruft?

Analyse

Wie wahrscheinlich ist es, dass MiniDisc MiniDiscWithPoint aufruft?

Gleiche Idee liefert gleiches Ergebnis

Analyse

Wie wahrscheinlich ist es, dass MiniDisc MiniDiscWithPoint aufruft?

Gleiche Idee liefert gleiches Ergebnis

Theorem: Die smallest enclosing disc für n Punkte in der Ebene kann in erwartet $\mathcal{O}(n)$ mit linearen Speicherplatz verbrauch berechnet werden.

□

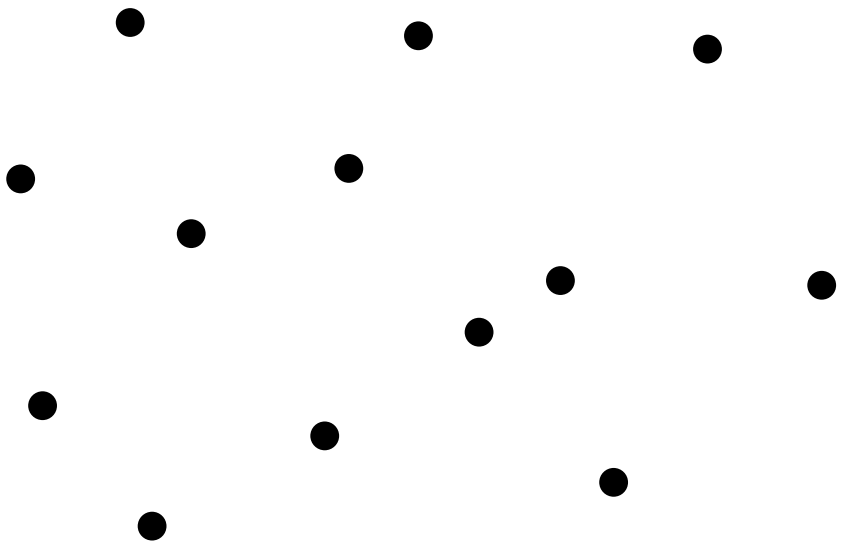
Ablauf

1. Smallest Enclosing Disk

2. Konvexe Hüllen im $\mathbb{R}^3$

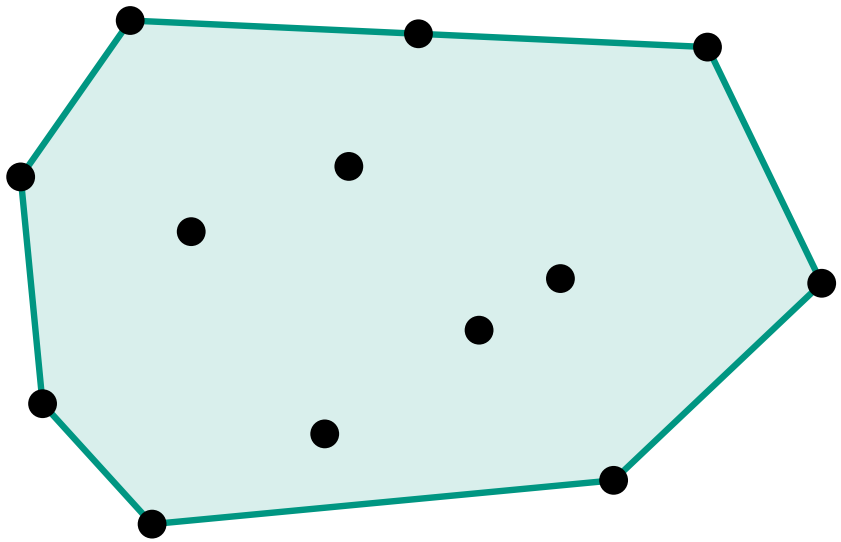
Die konvexe Hülle

Def: Eine Menge $S \subseteq \mathbb{R}^2$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.



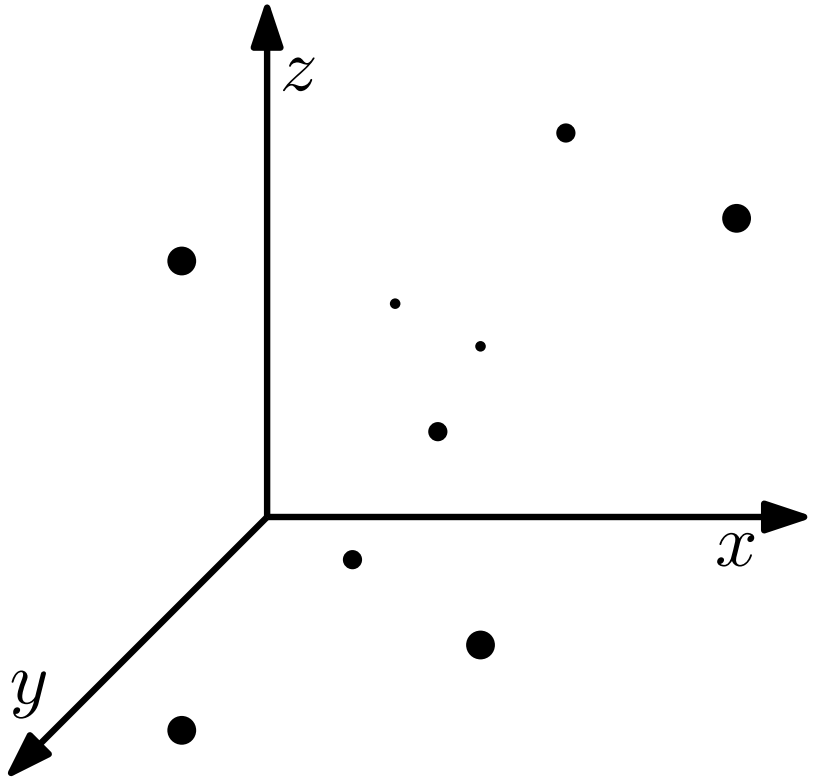
Die konvexe Hülle

Def: Eine Menge $S \subseteq \mathbb{R}^2$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.



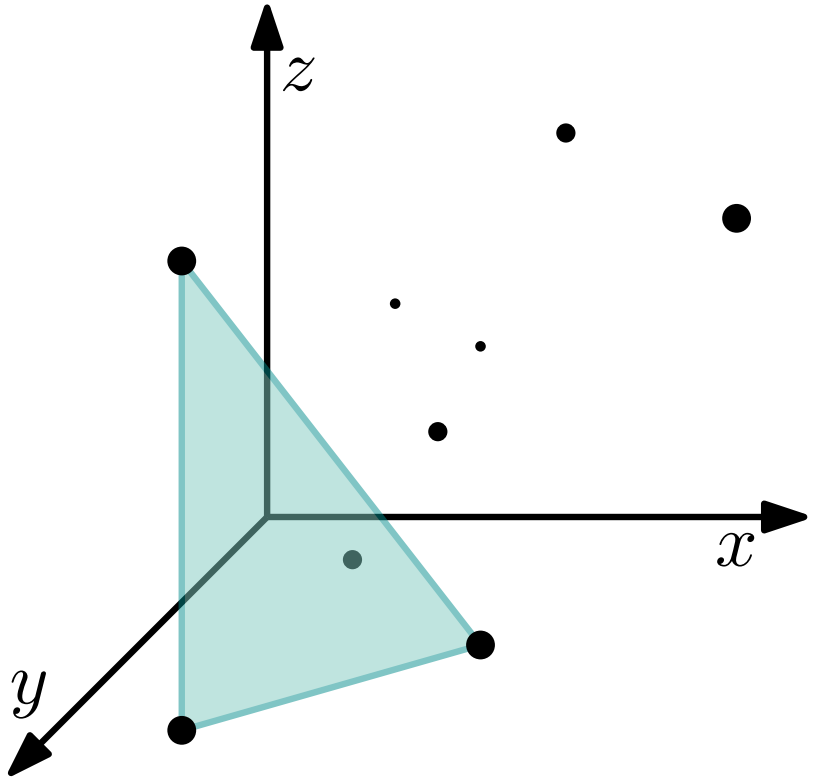
Die konvexe Hülle

Def: Eine Menge $S \subseteq \mathbb{R}^3$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.



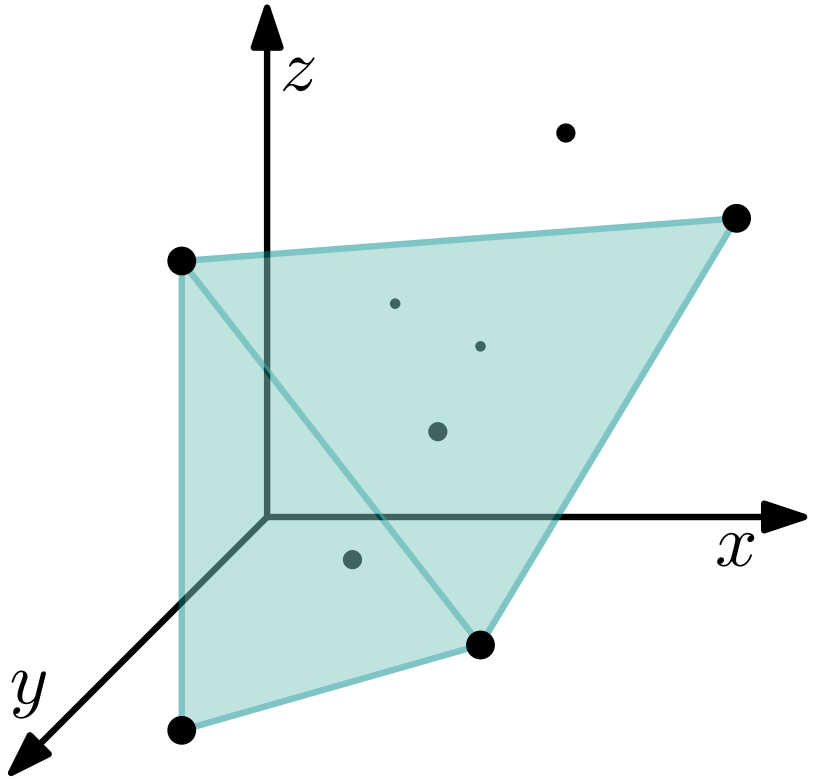
Die konvexe Hülle

Def: Eine Menge $S \subseteq \mathbb{R}^3$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.



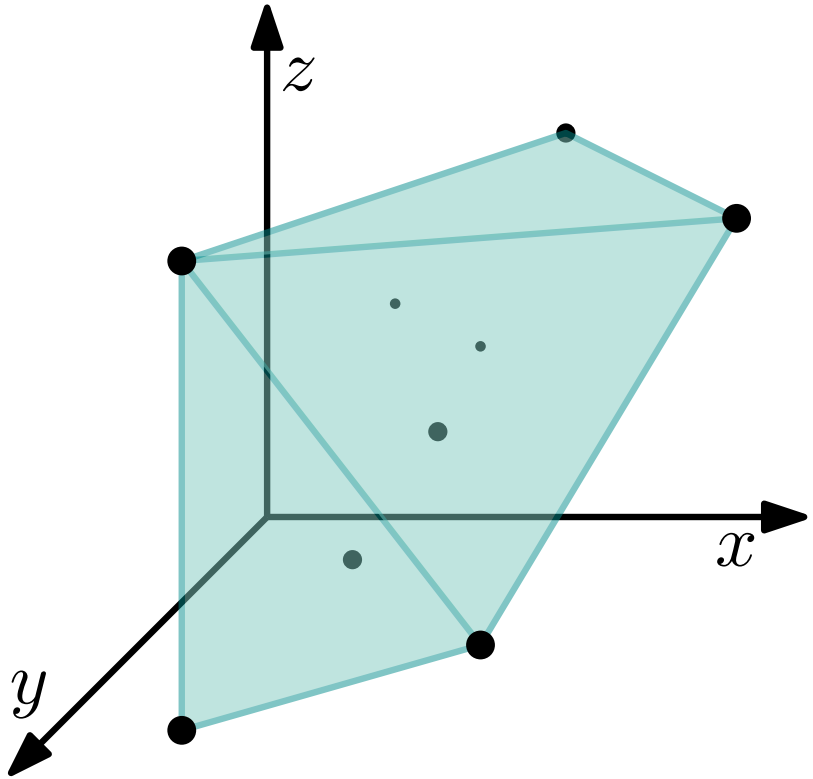
Die konvexe Hülle

Def: Eine Menge $S \subseteq \mathbb{R}^3$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.



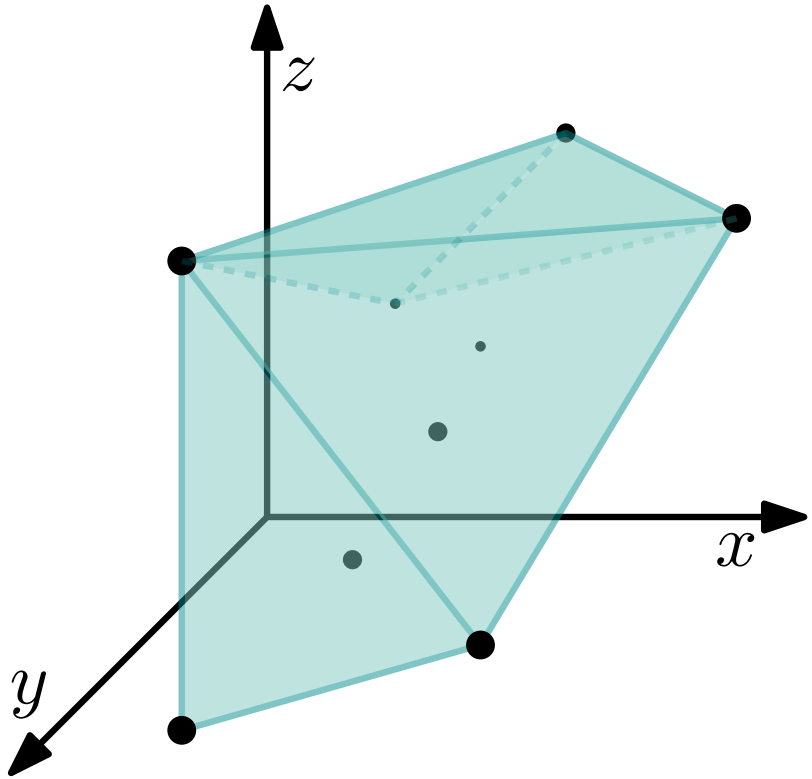
Die konvexe Hülle

Def: Eine Menge $S \subseteq \mathbb{R}^3$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.



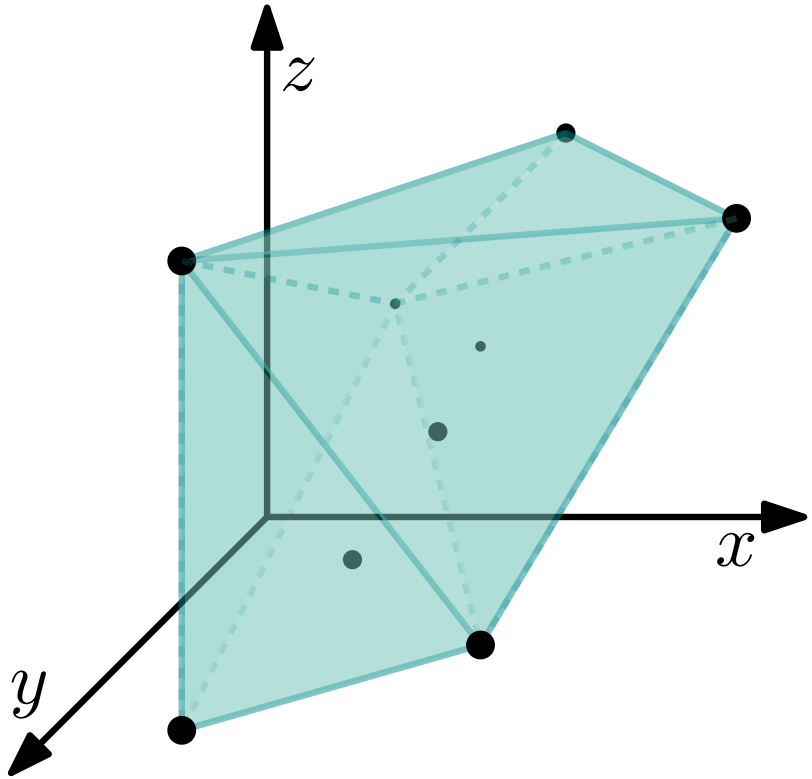
Die konvexe Hülle

Def: Eine Menge $S \subseteq \mathbb{R}^3$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.



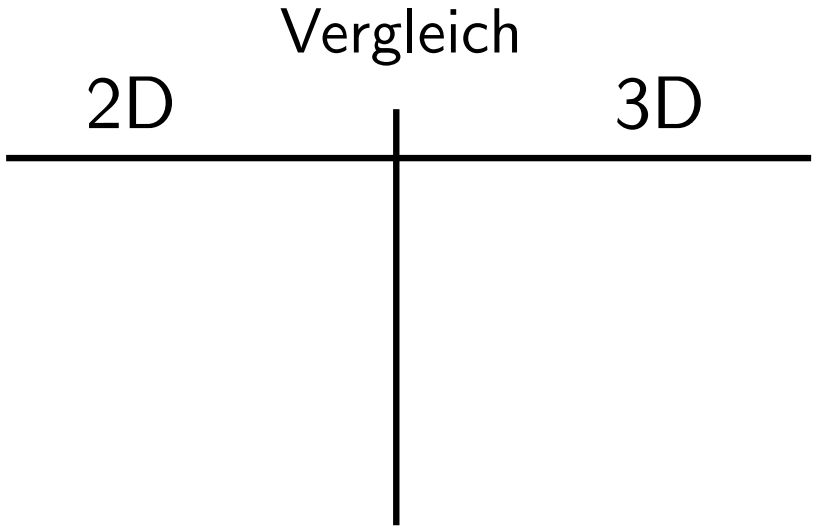
Die konvexe Hülle

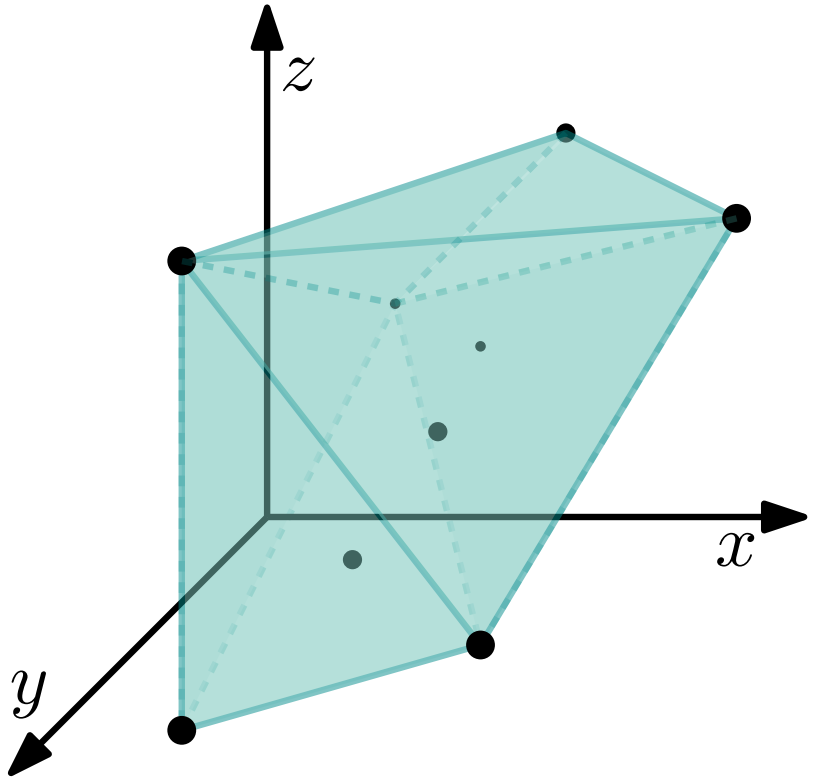
Def: Eine Menge $S \subseteq \mathbb{R}^3$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.



Die konvexe Hülle

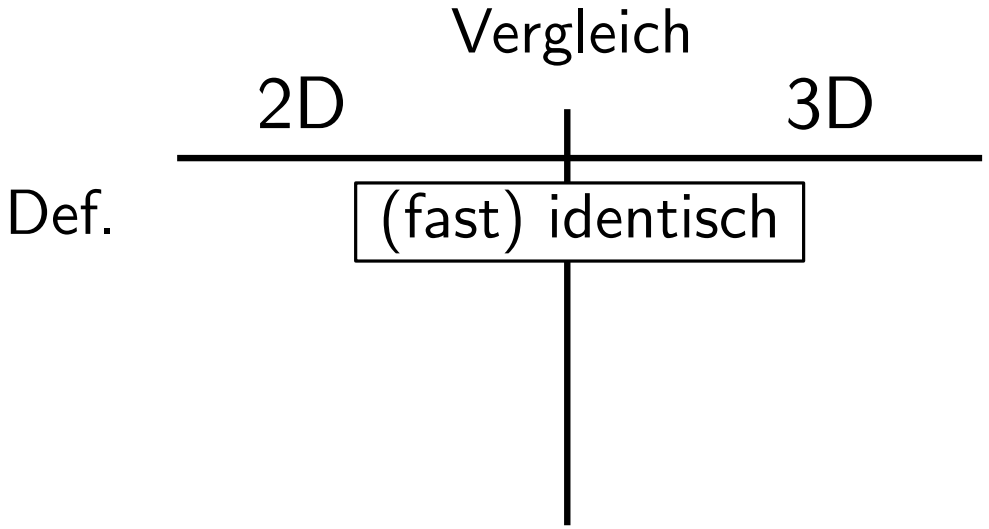
Def: Eine Menge $S \subseteq \mathbb{R}^3$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.

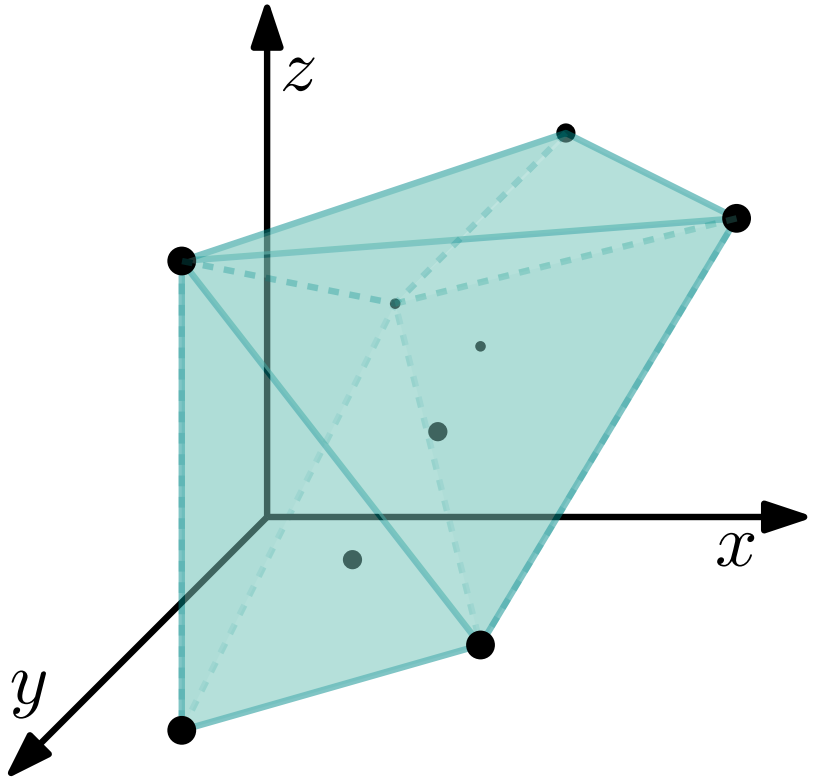




Die konvexe Hülle

Def: Eine Menge $S \subseteq \mathbb{R}^3$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.

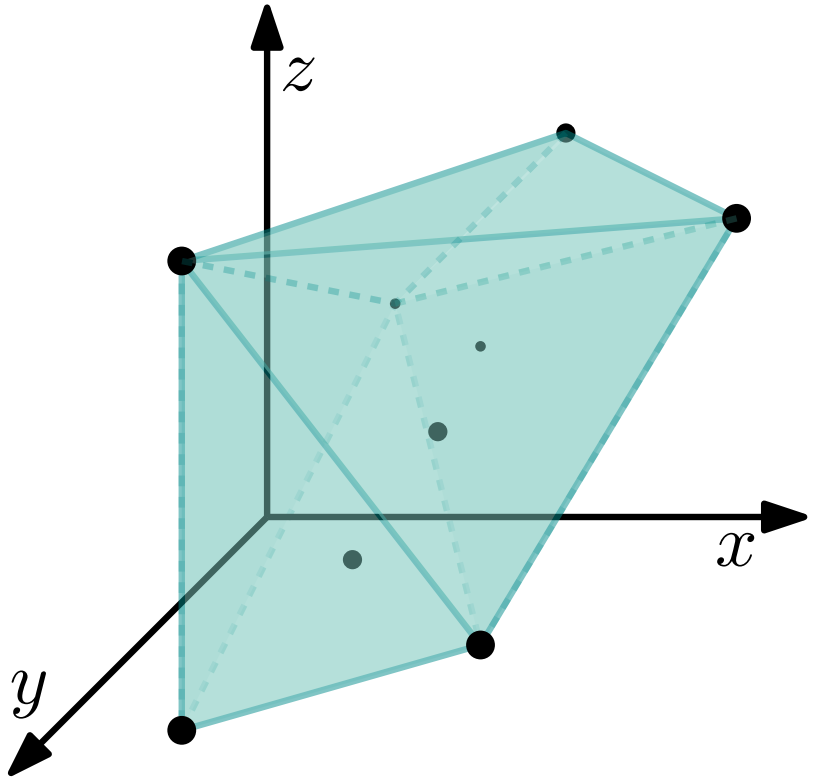




Die konvexe Hülle

Def: Eine Menge $S \subseteq \mathbb{R}^3$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.

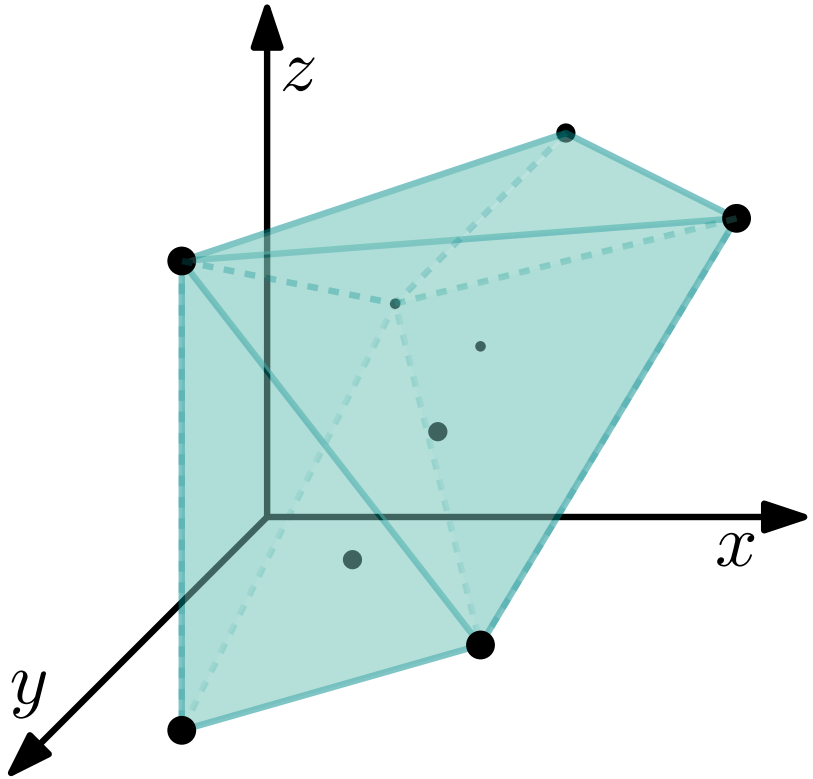
Vergleich	2D	3D
Def.	(fast) identisch	
Art	Polygon	



Die konvexe Hülle

Def: Eine Menge $S \subseteq \mathbb{R}^3$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.

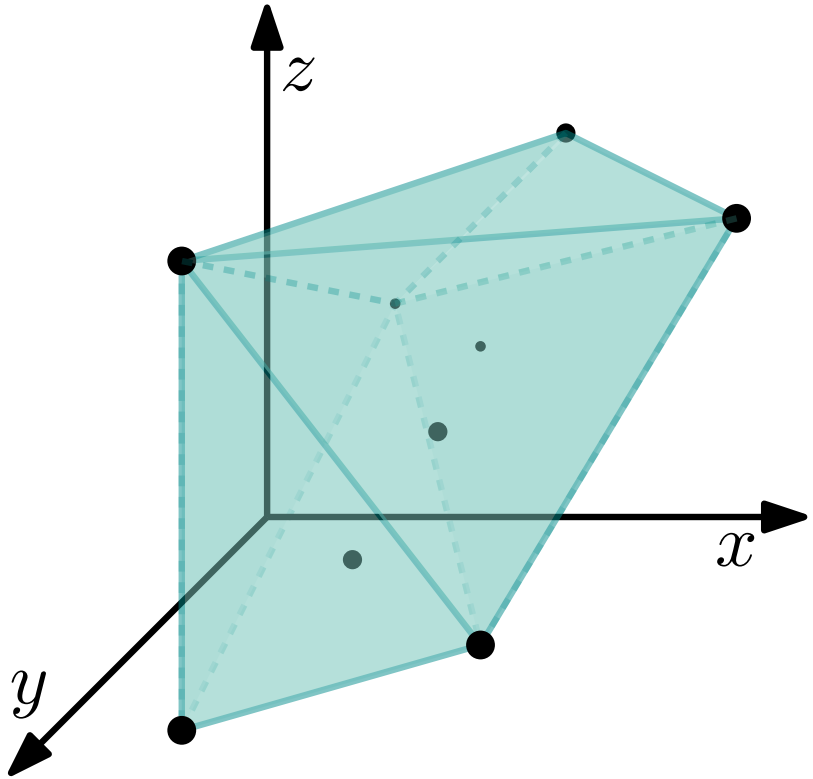
Vergleich	2D	3D
Def.	(fast) identisch	
Art	Polygon	Polytop



Die konvexe Hülle

Def: Eine Menge $S \subseteq \mathbb{R}^3$ heißt **konvex**, wenn für je zwei Punkte $p, q \in S$ auch gilt $\overline{pq} \in S$.
Die **konvexe Hülle** $CH(S)$ von S ist die kleinste konvexe Menge, die S enthält.

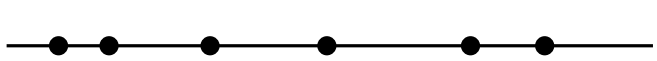
Vergleich	2D	3D
Def.	(fast) identisch	
Art	Polygon	Polytop
Kompl.	$\mathcal{O}(n)$ Kanten	?



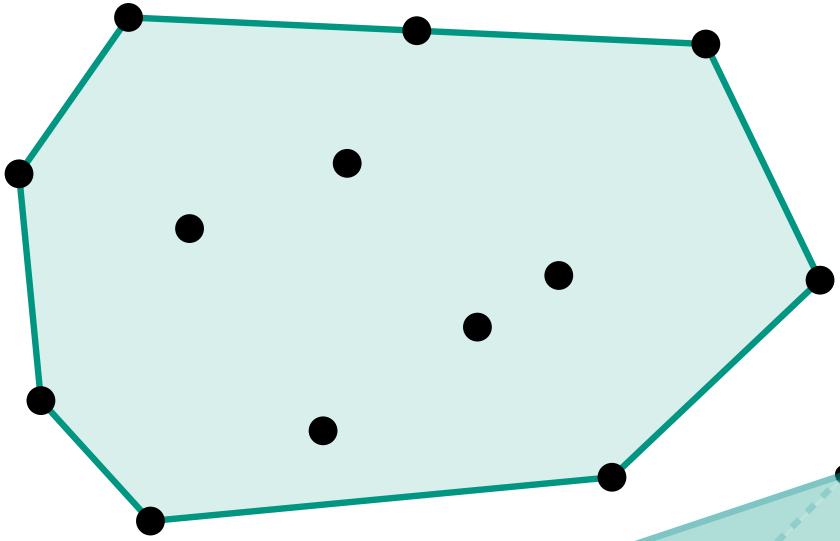
Komplexität konvexer Hüllen

Dimension	Komplexität
1	?
2	$\mathcal{O}(n)$
3	?

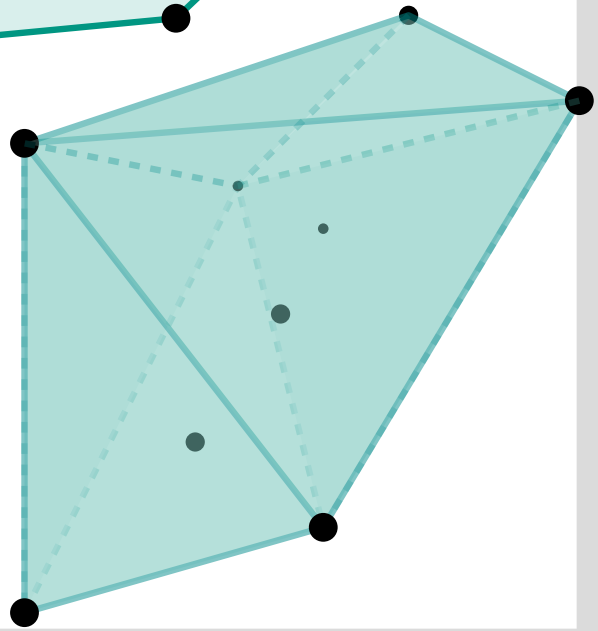
1D



2D



3D

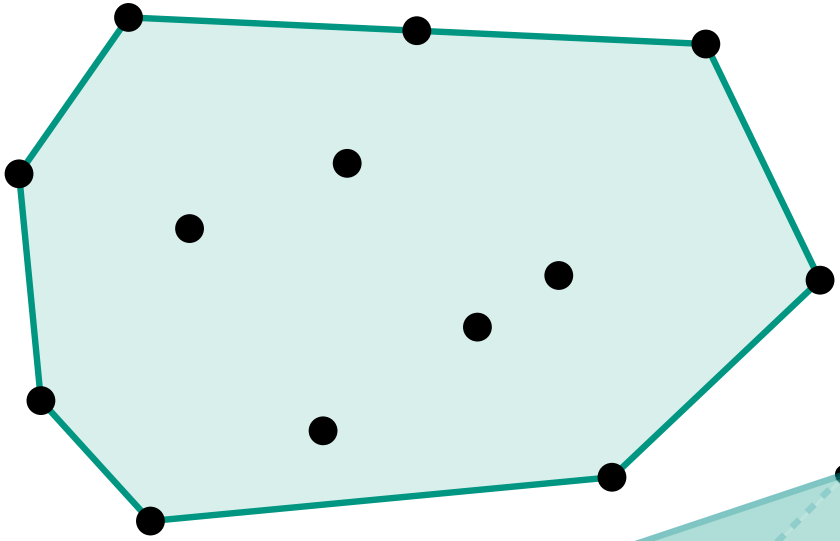


Komplexität konvexer Hüllen

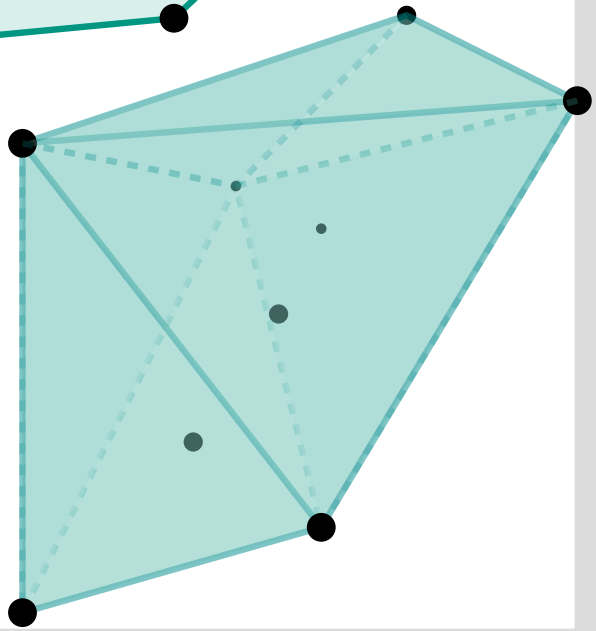
Dimension	Komplexität
1	?
2	$\mathcal{O}(n)$
3	?

1D

2D



3D

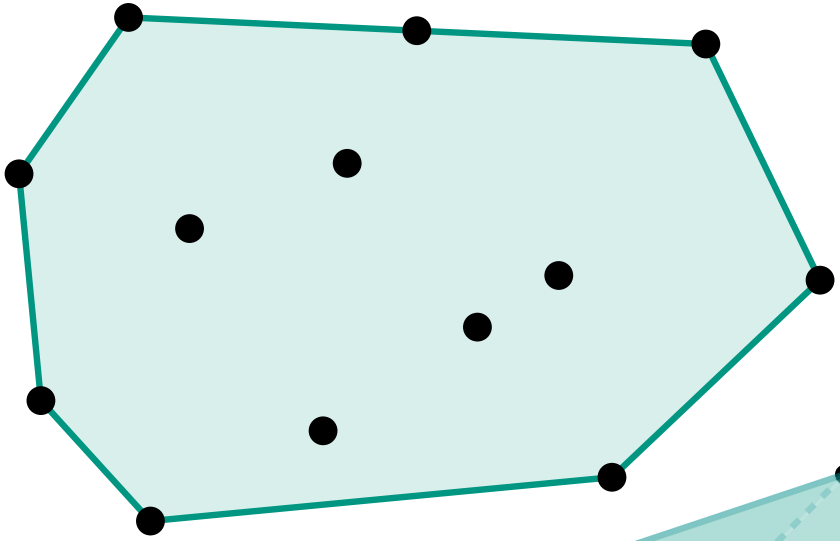


Komplexität konvexer Hüllen

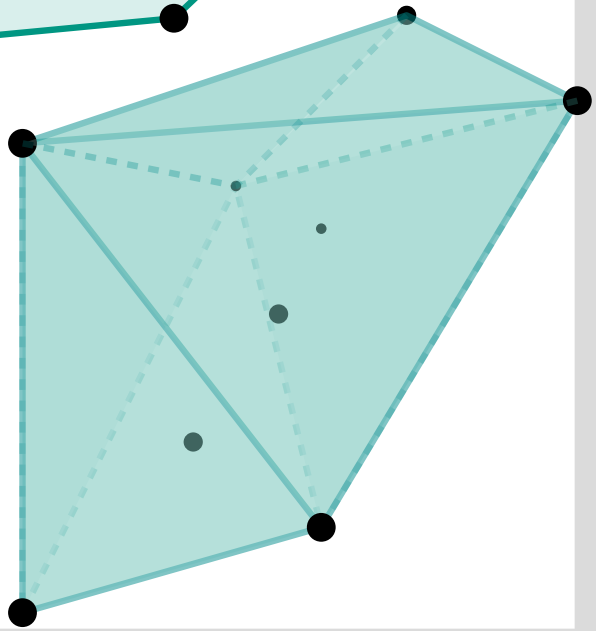
Dimension	Komplexität
1	$\mathcal{O}(1)$
2	$\mathcal{O}(n)$
3	?

1D

2D



3D

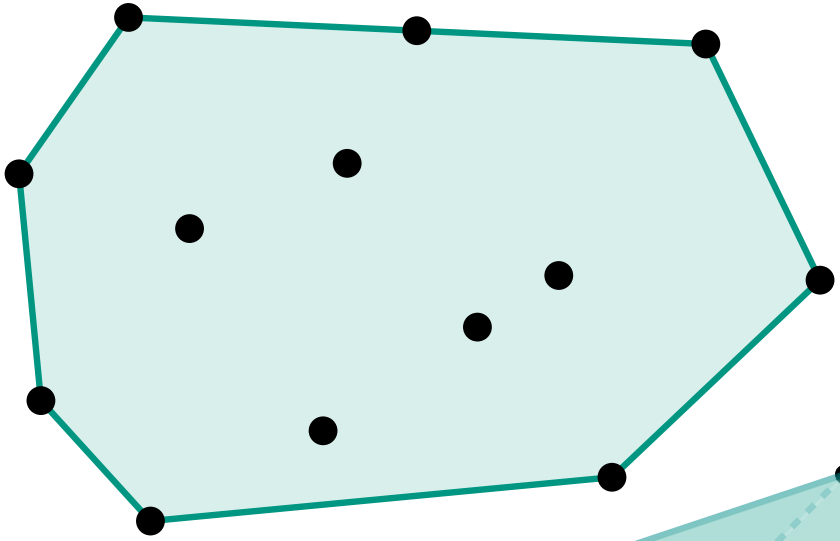


Komplexität konvexer Hüllen

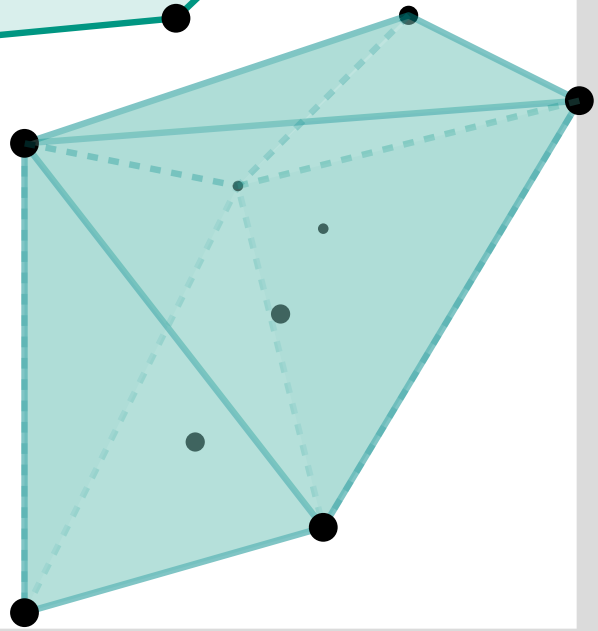
Dimension	Komplexität
1	$\mathcal{O}(1)$
2	$\mathcal{O}(n)$
3	?

1D

2D



3D



Theorem:

Sei $\mathcal{P}$ ein konvexes Polytop mit n Knoten. Dann ist die Anzahl der Kanten in $\mathcal{P}$ höchsten $3n - 6$ und die Anzahl der Facetten höchstens $2n - 4$.

Komplexität konvexer Hüllen in 3D

Karlsruhe Institute of Technology

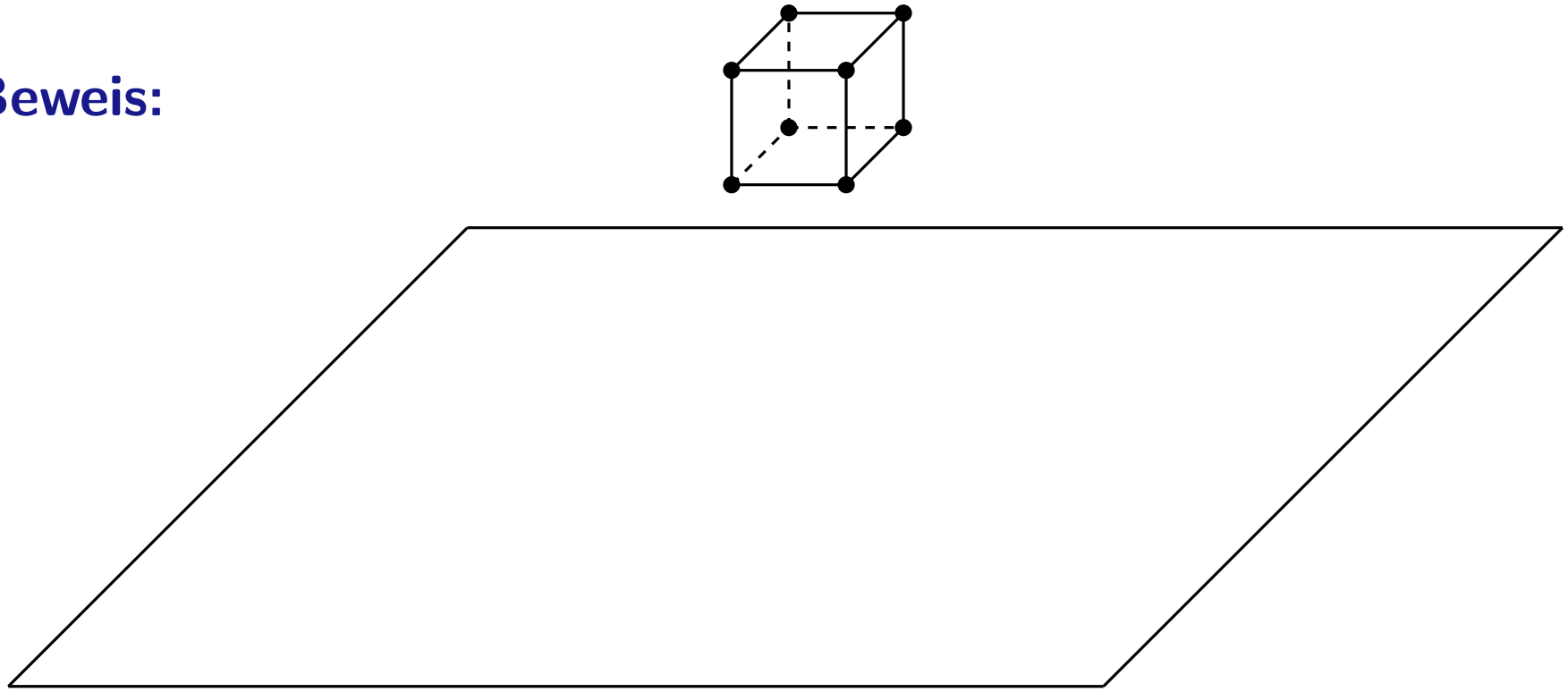
Theorem:
Sei $\mathcal{P}$ ein konvexes Polytop mit n Knoten. Dann ist die Anzahl der Kanten in $\mathcal{P}$ höchsten $3n - 6$ und die Anzahl der Facetten höchstens $2n - 4$.

Beweis:

Komplexität konvexer Hüllen in 3D

Theorem:
Sei $\mathcal{P}$ ein konvexes Polytop mit n Knoten. Dann ist die Anzahl der Kanten in $\mathcal{P}$ höchsten $3n - 6$ und die Anzahl der Facetten höchstens $2n - 4$.

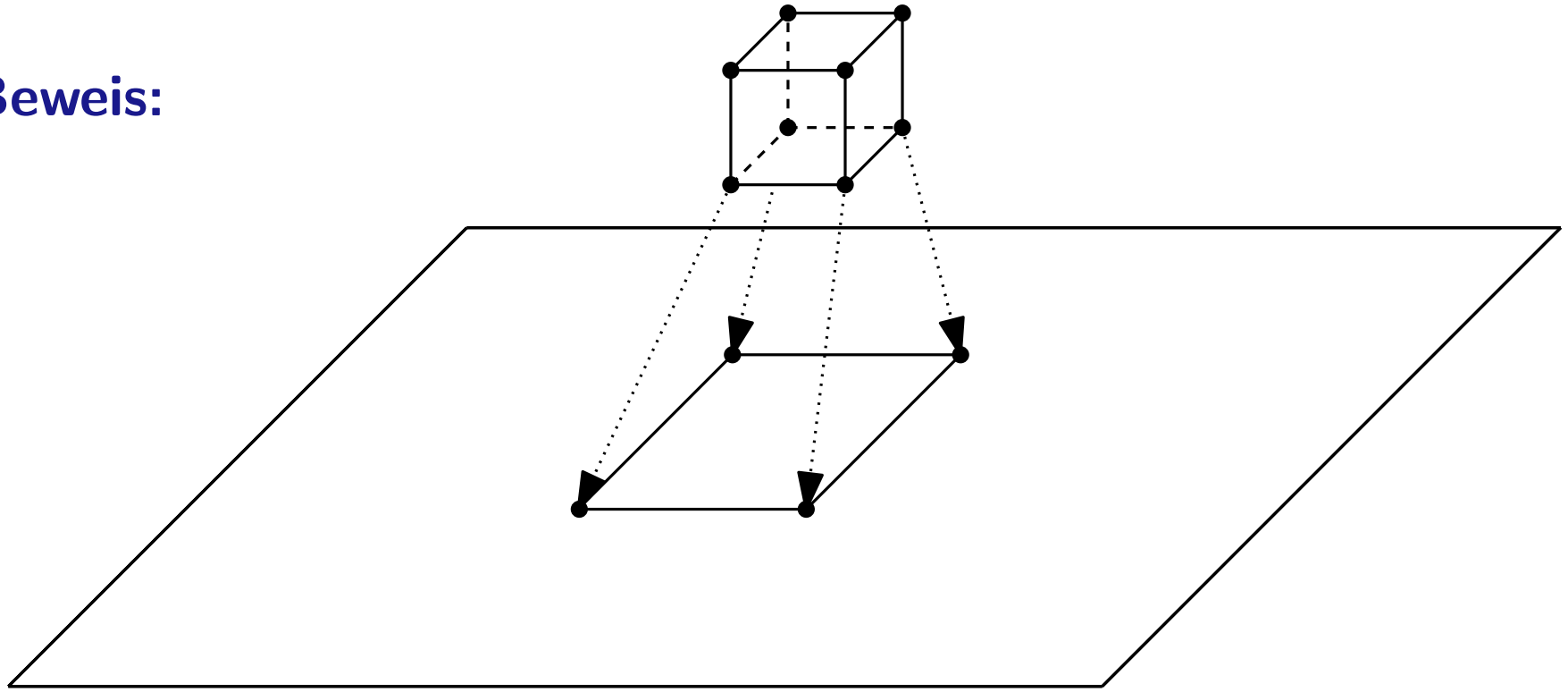
Beweis:



Komplexität konvexer Hüllen in 3D

Theorem:
Sei $\mathcal{P}$ ein konvexes Polytop mit n Knoten. Dann ist die Anzahl der Kanten in $\mathcal{P}$ höchsten $3n - 6$ und die Anzahl der Facetten höchstens $2n - 4$.

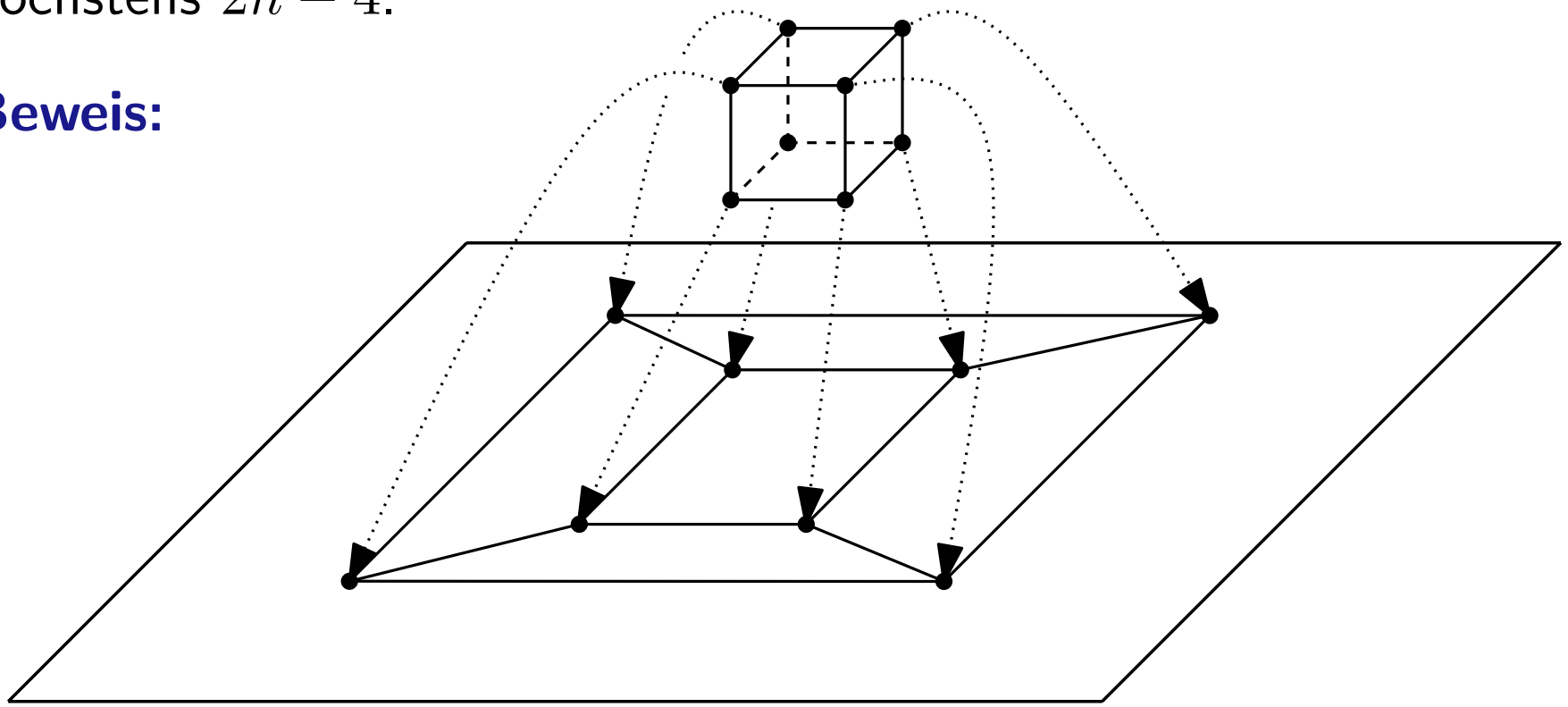
Beweis:



Komplexität konvexer Hüllen in 3D

Theorem:
Sei $\mathcal{P}$ ein konvexes Polytop mit n Knoten. Dann ist die Anzahl der Kanten in $\mathcal{P}$ höchsten $3n - 6$ und die Anzahl der Facetten höchstens $2n - 4$.

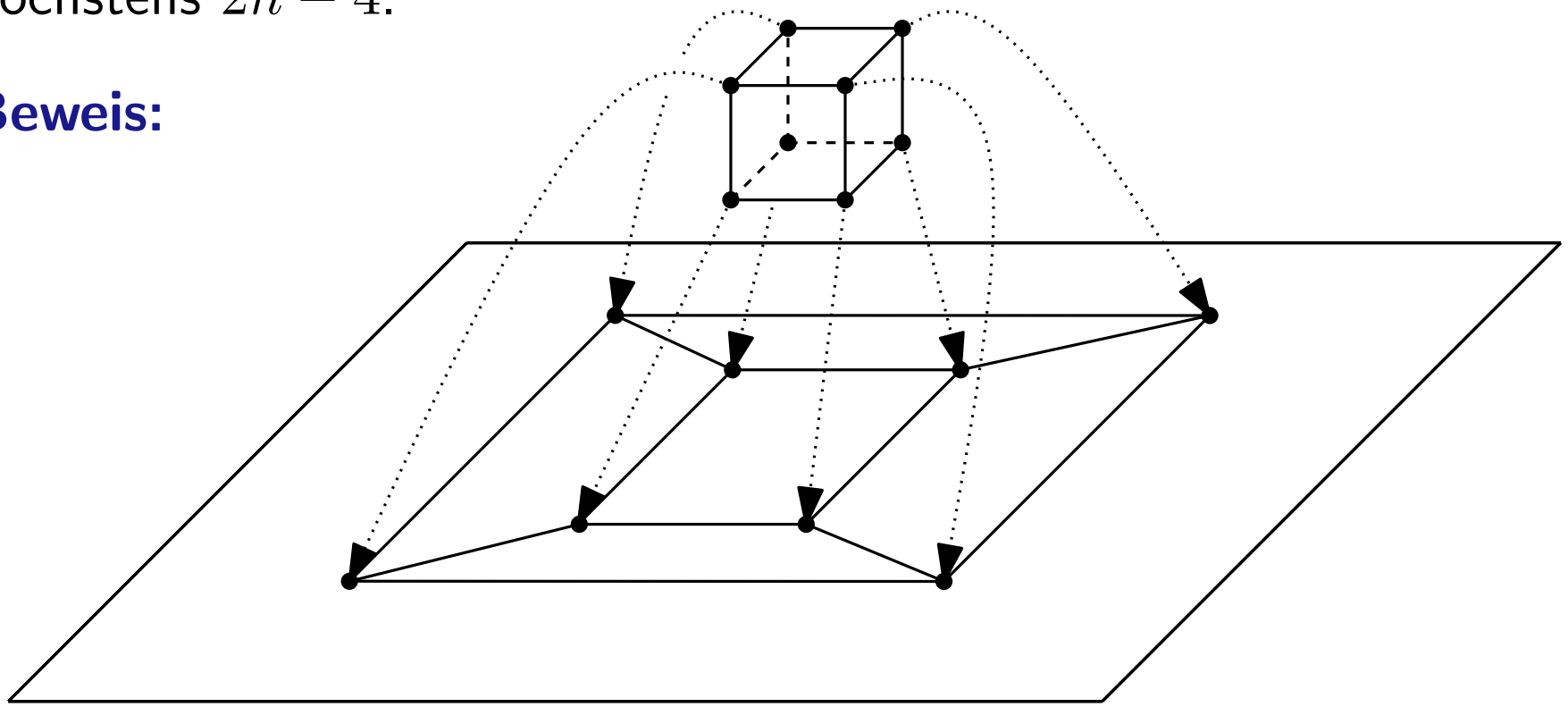
Beweis:



Komplexität konvexer Hüllen in 3D

Theorem:
Sei $\mathcal{P}$ ein konvexes Polytop mit n Knoten. Dann ist die Anzahl der Kanten in $\mathcal{P}$ höchsten $3n - 6$ und die Anzahl der Facetten höchstens $2n - 4$.

Beweis:

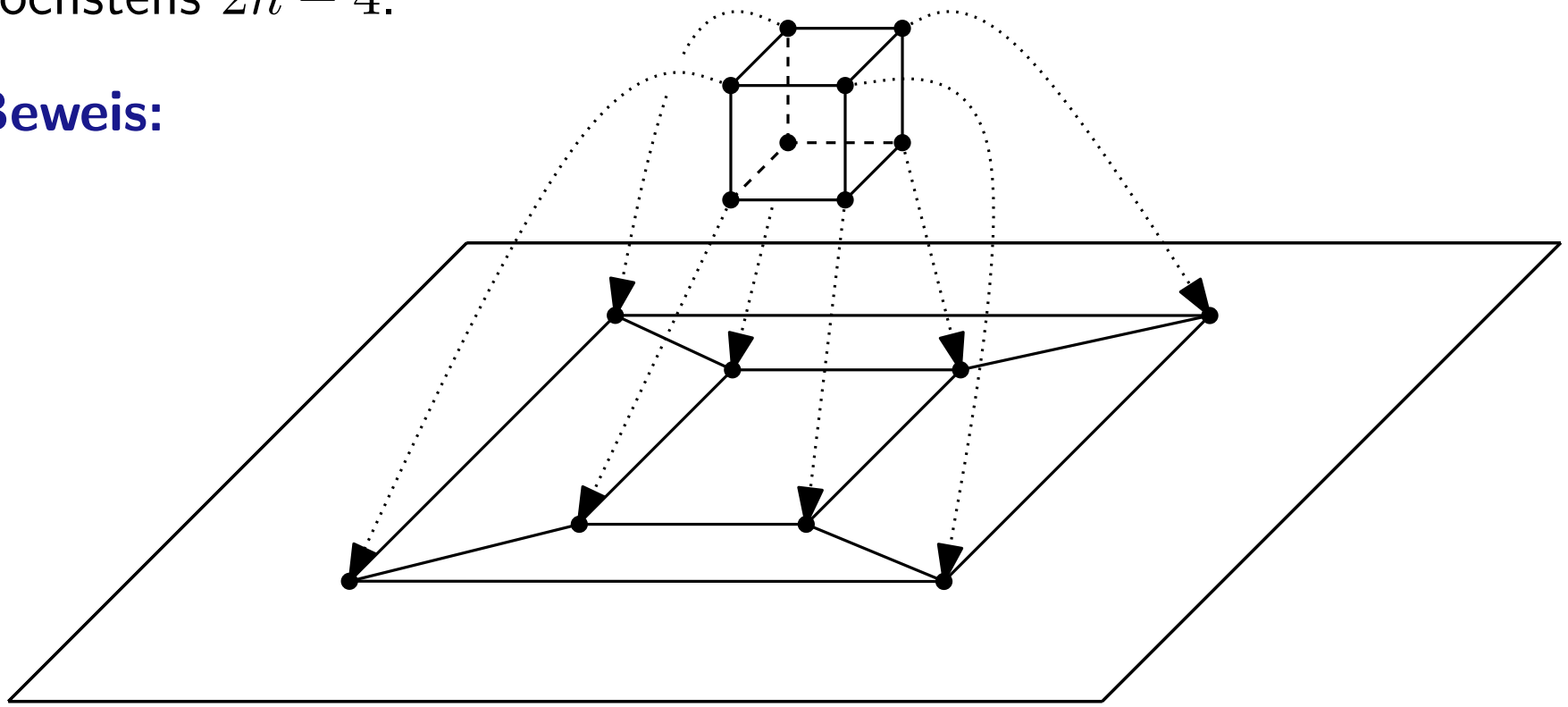


s. Tafel

Komplexität konvexer Hüllen in 3D

Theorem:
Sei $\mathcal{P}$ ein konvexes Polytop mit n Knoten. Dann ist die Anzahl der Kanten in $\mathcal{P}$ höchsten $3n - 6$ und die Anzahl der Facetten höchstens $2n - 4$.

Beweis:

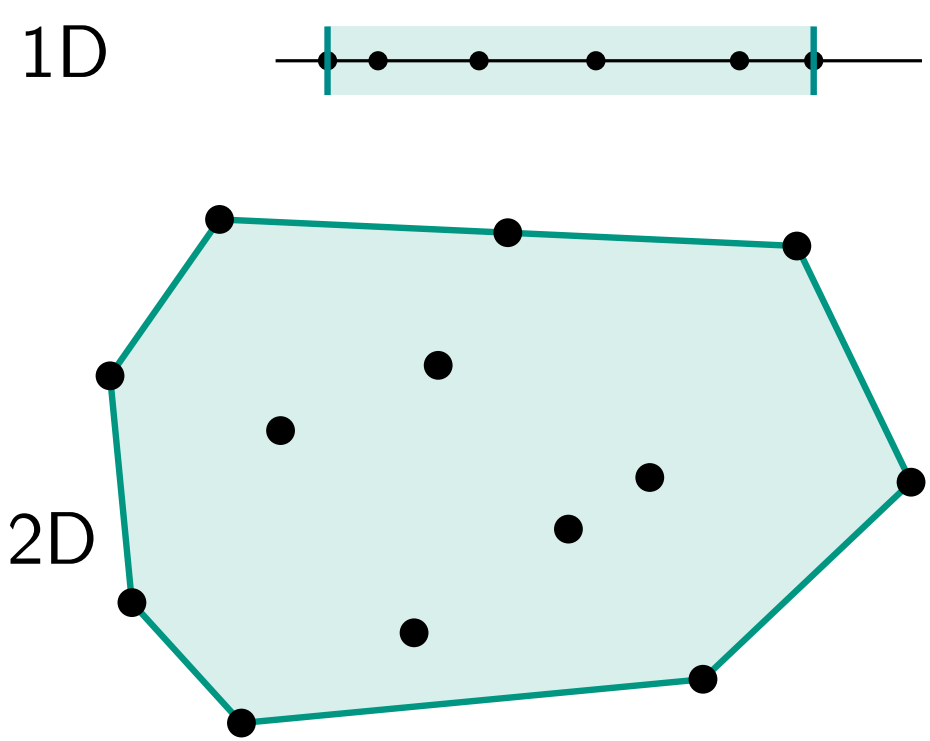


s. Tafel

□

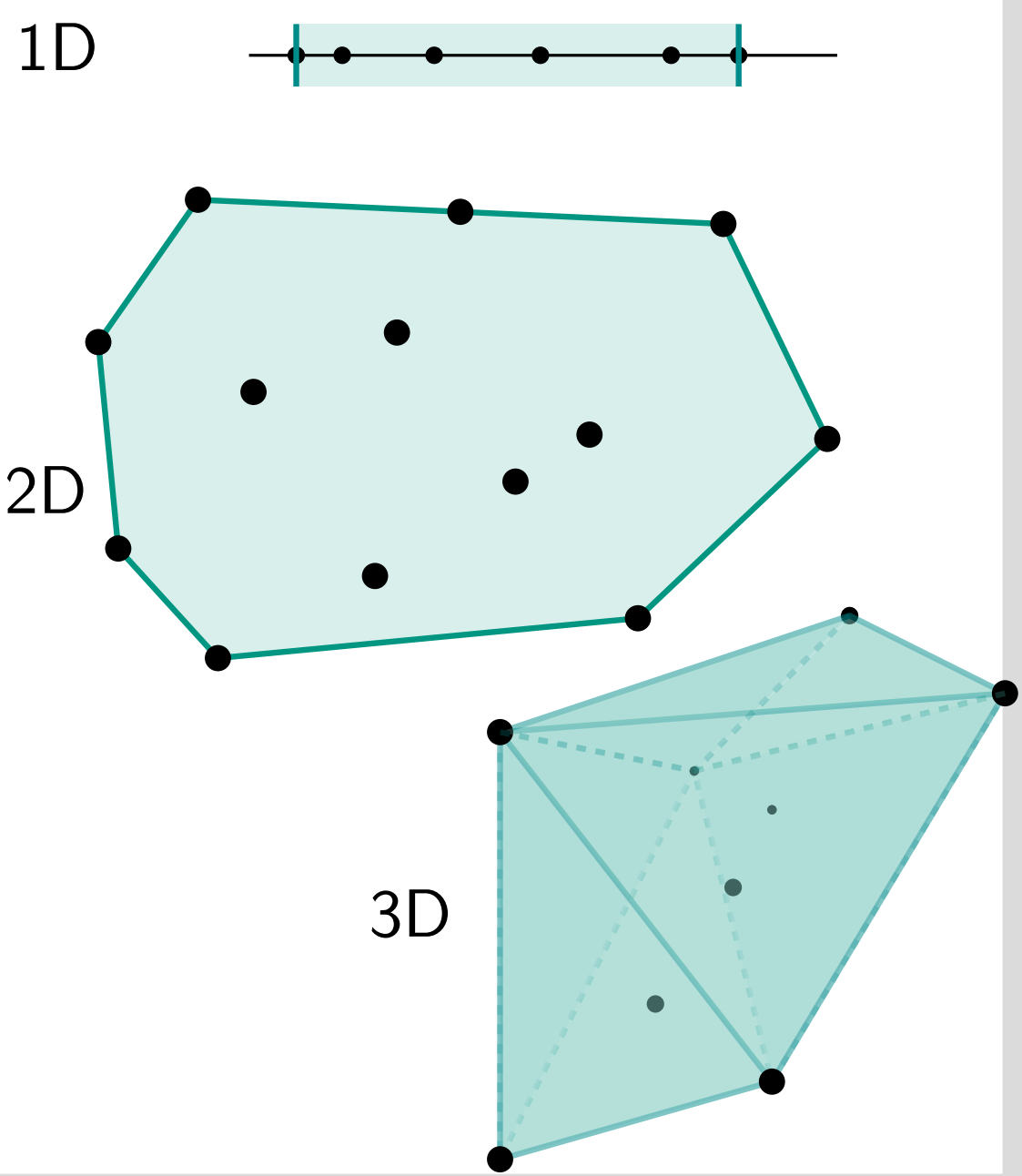
Komplexität konvexer Hüllen

Dimension	Komplexität
1	$\mathcal{O}(1)$
2	$\mathcal{O}(n)$
3	?



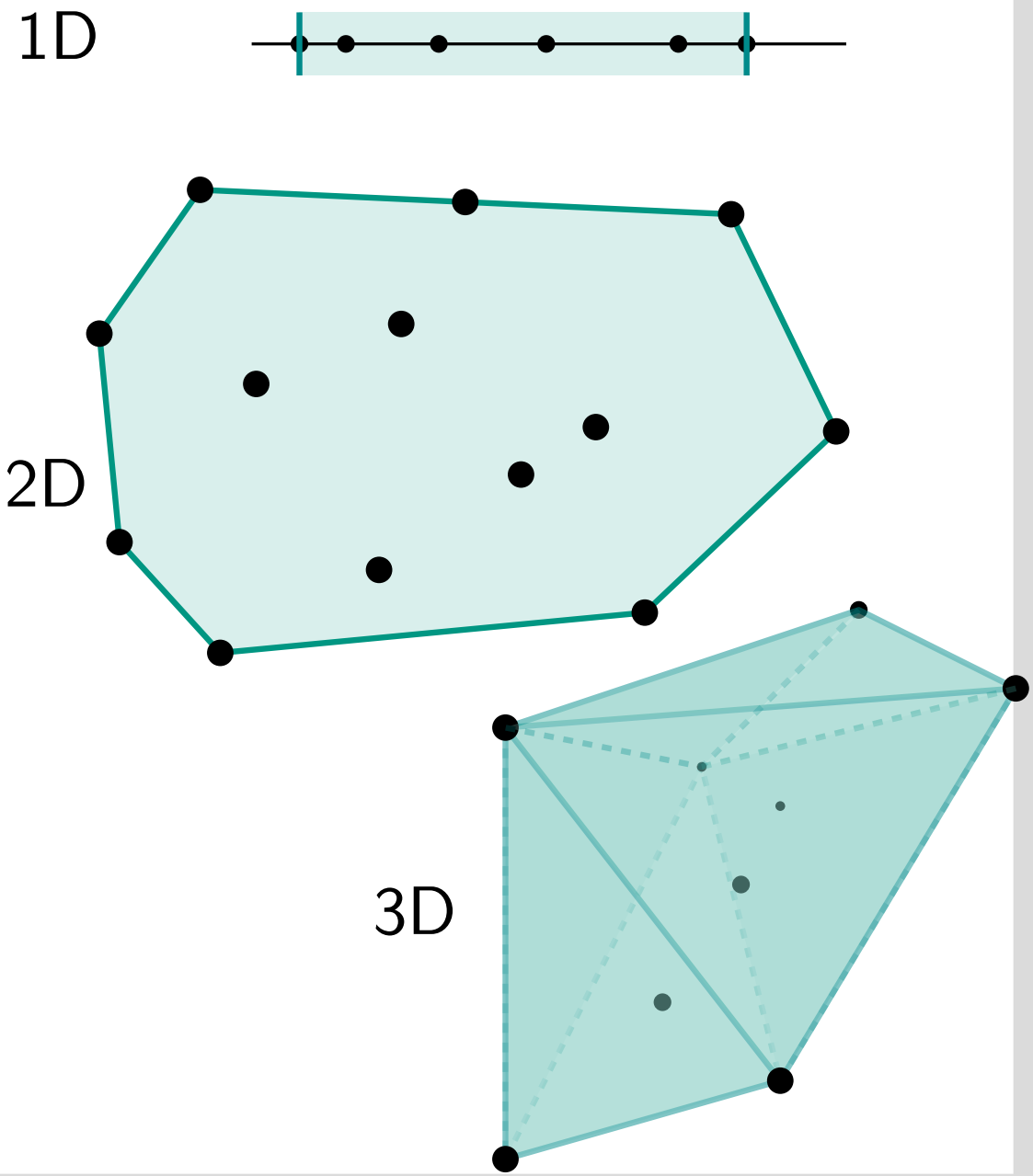
Komplexität konvexer Hüllen

Dimension	Komplexität
1	$\mathcal{O}(1)$
2	$\mathcal{O}(n)$
3	$\mathcal{O}(n)$



Komplexität konvexer Hüllen

Dimension	Komplexität
1	$\mathcal{O}(1)$
2	$\mathcal{O}(n)$
3	$\mathcal{O}(n)$
d	?



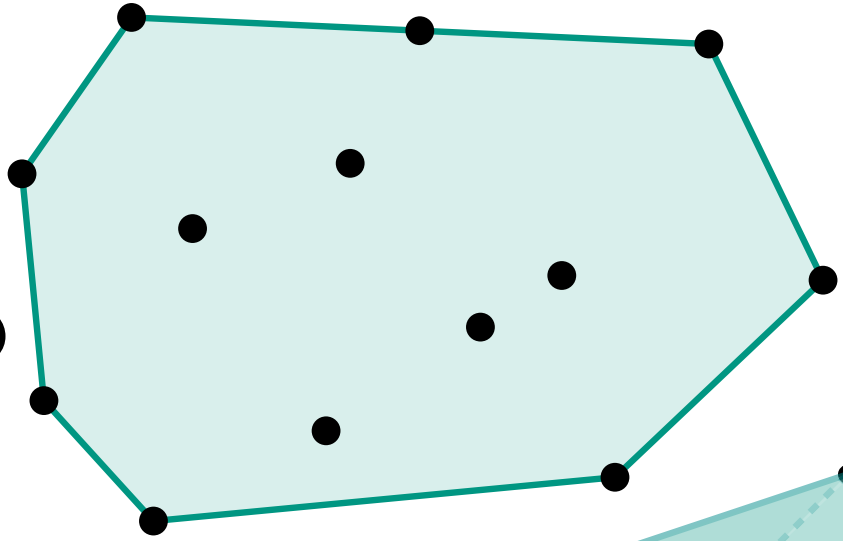
Komplexität konvexer Hüllen

Dimension	Komplexität
1	$\mathcal{O}(1)$
2	$\mathcal{O}(n)$
3	$\mathcal{O}(n)$
d	$\Theta(n^{\lfloor d/2 \rfloor})$

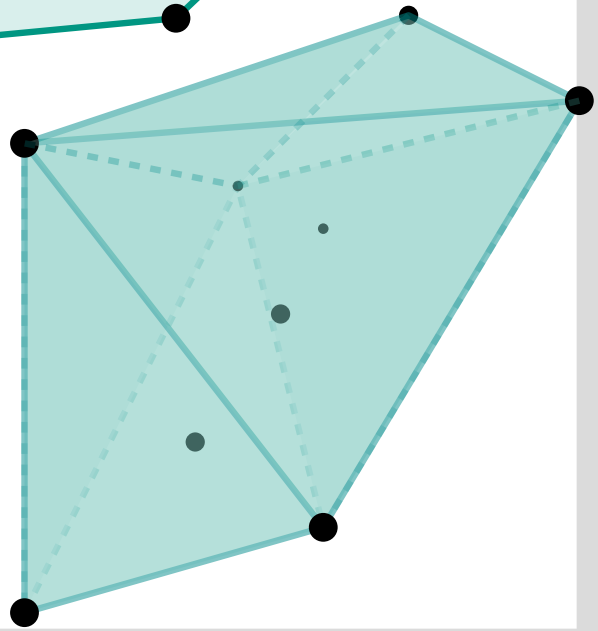
Upper Bound Theorem

1D

2D



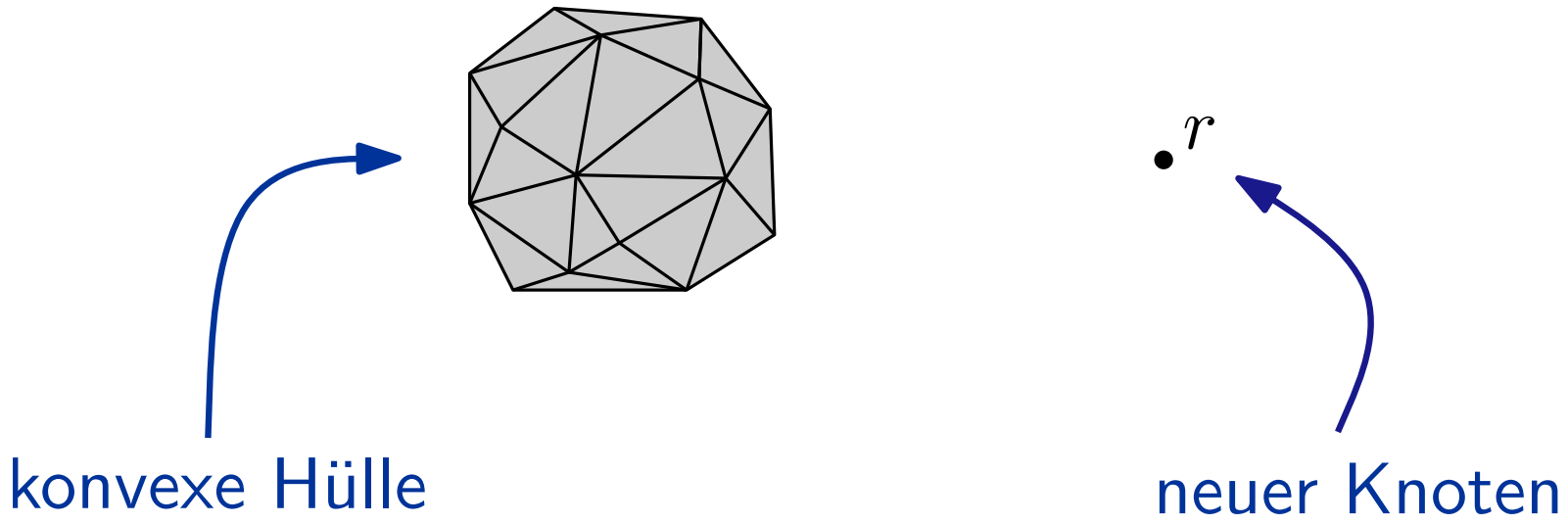
3D



Erster Schritt

Angenommen:

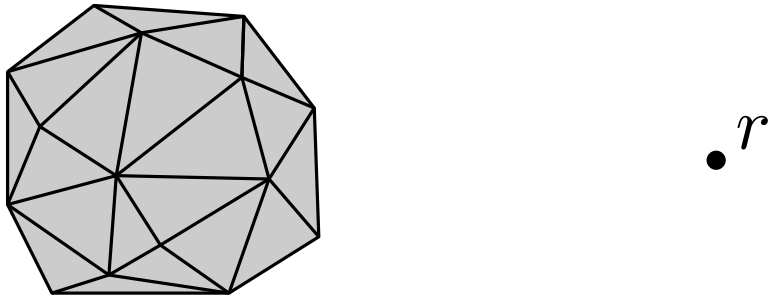
Konvexe Hülle von $r-1$ Knoten gegeben. Wie erweitern zu einer konvexen Hülle für r Knoten?



Erster Schritt

Angenommen:

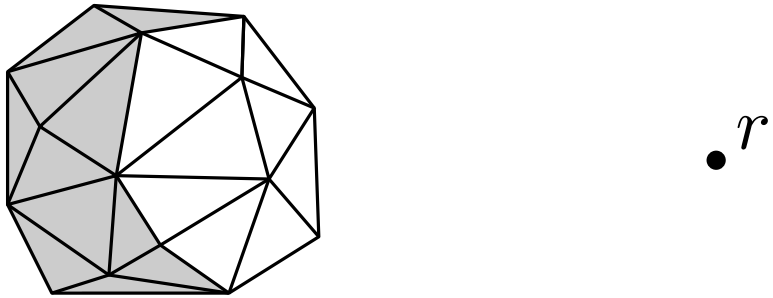
Konvexe Hülle von $r-1$ Knoten gegeben. Wie erweitern zu einer konvexen Hülle für r Knoten?



Erster Schritt

Angenommen:

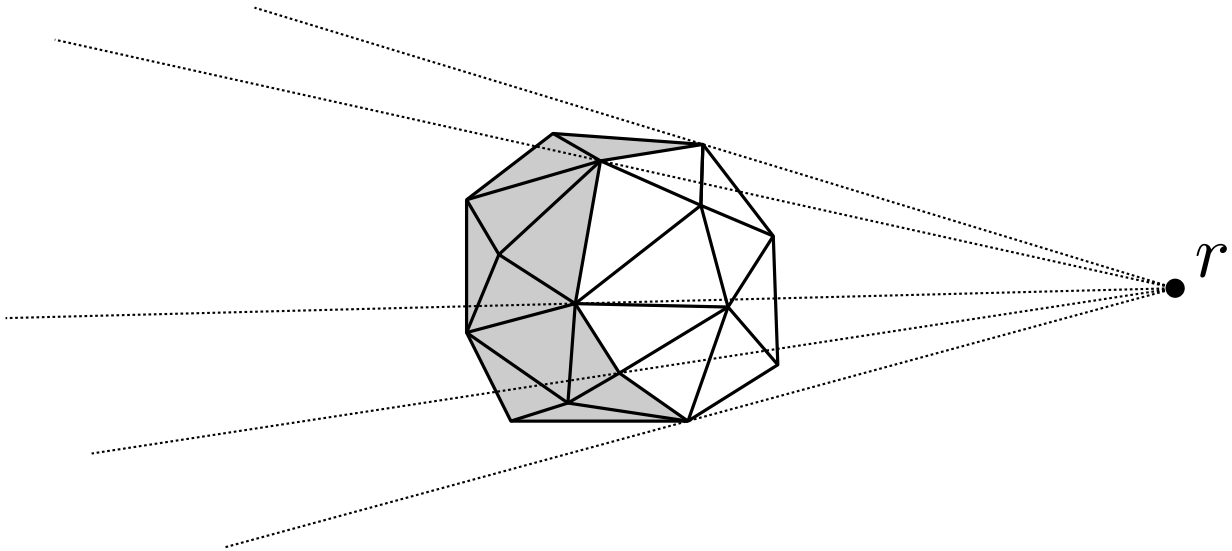
Konvexe Hülle von $r-1$ Knoten gegeben. Wie erweitern zu einer konvexen Hülle für r Knoten?



Erster Schritt

Angenommen:

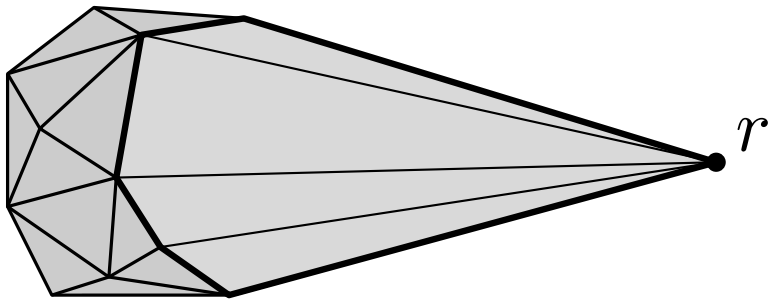
Konvexe Hülle von $r-1$ Knoten gegeben. Wie erweitern zu einer konvexen Hülle für r Knoten?



Erster Schritt

Angenommen:

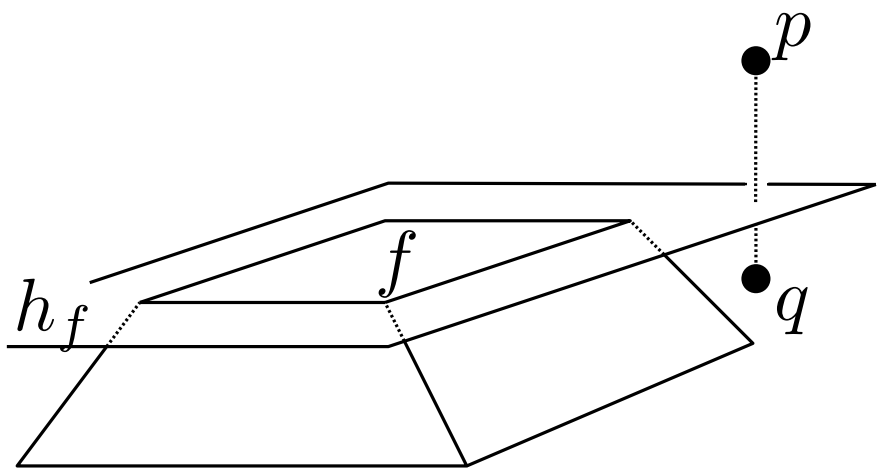
Konvexe Hülle von $r-1$ Knoten gegeben. Wie erweitern zu einer konvexen Hülle für r Knoten?



Erster Schritt

Angenommen:

Konvexe Hülle von $r-1$ Knoten gegeben. Wie erweitern zu einer konvexen Hülle für r Knoten?

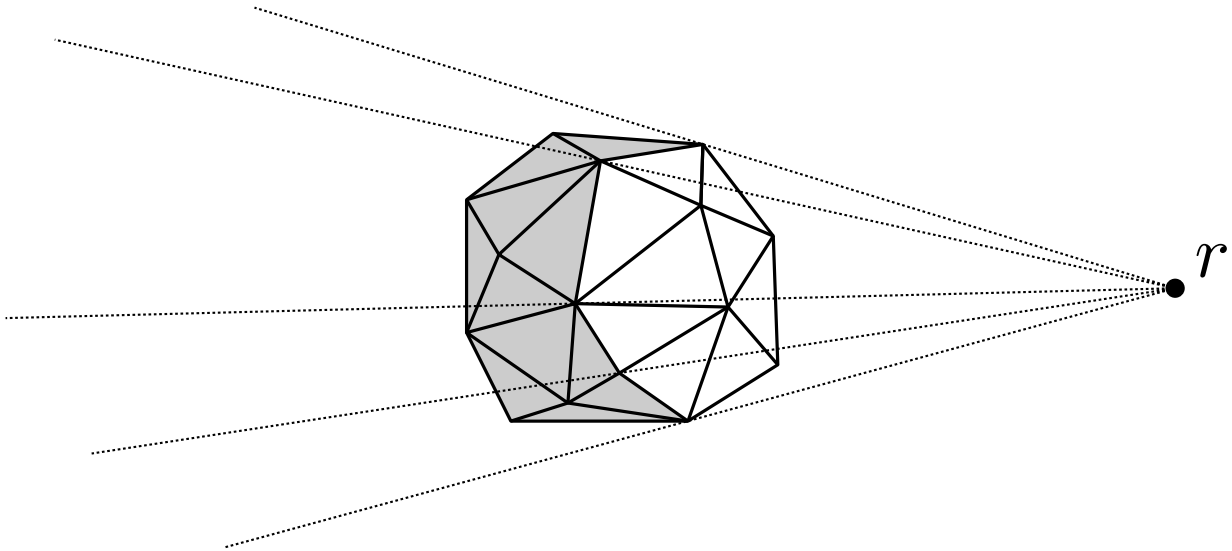


Facette f sichtbar von p aber nicht von q aus

Erster Schritt

Angenommen:

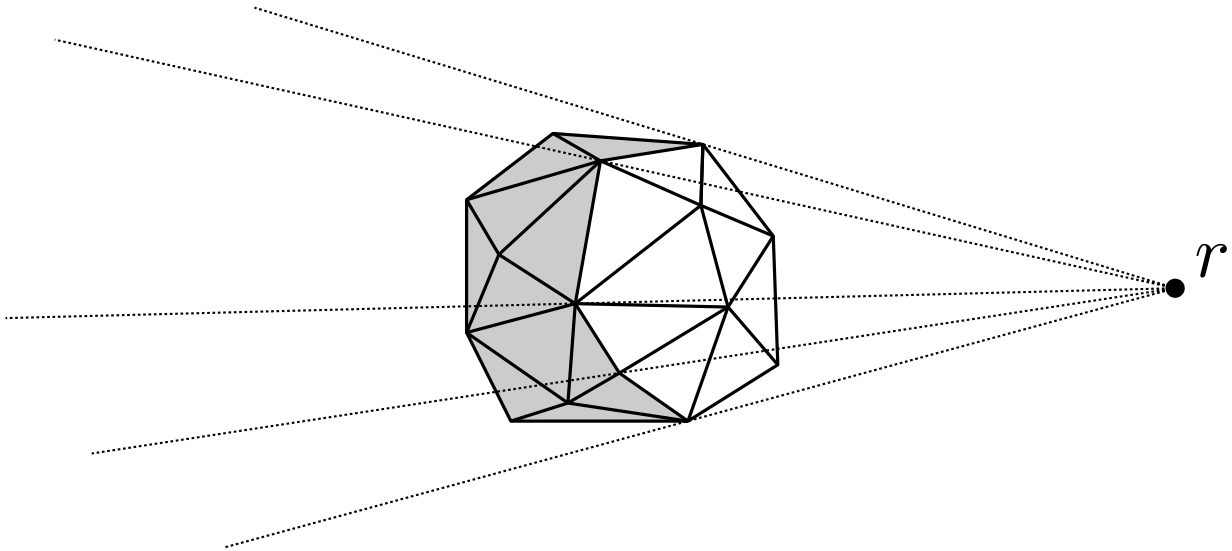
Konvexe Hülle von $r-1$ Knoten gegeben. Wie erweitern zu einer konvexen Hülle für r Knoten?



Erster Schritt

Angenommen:

Konvexe Hülle von $r-1$ Knoten gegeben. Wie erweitern zu einer konvexen Hülle für r Knoten?



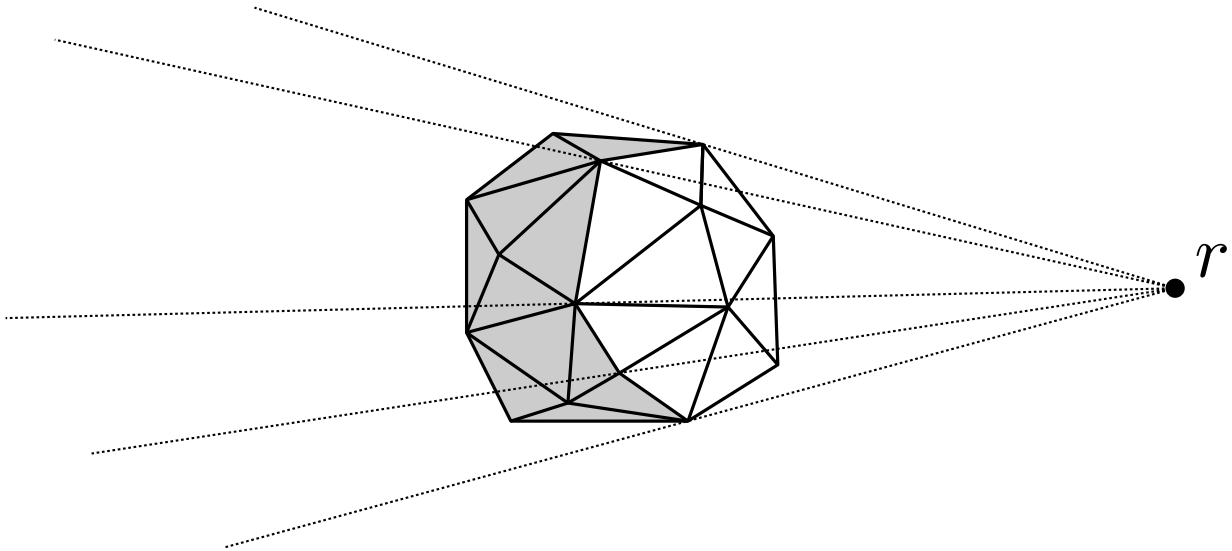
Aufgabe:

Überlege einen $\mathcal{O}(n^2)$ Algorithmus der die konvexe Hülle berechnet.

Erster Schritt

Angenommen:

Konvexe Hülle von $r-1$ Knoten gegeben. Wie erweitern zu einer konvexen Hülle für r Knoten?

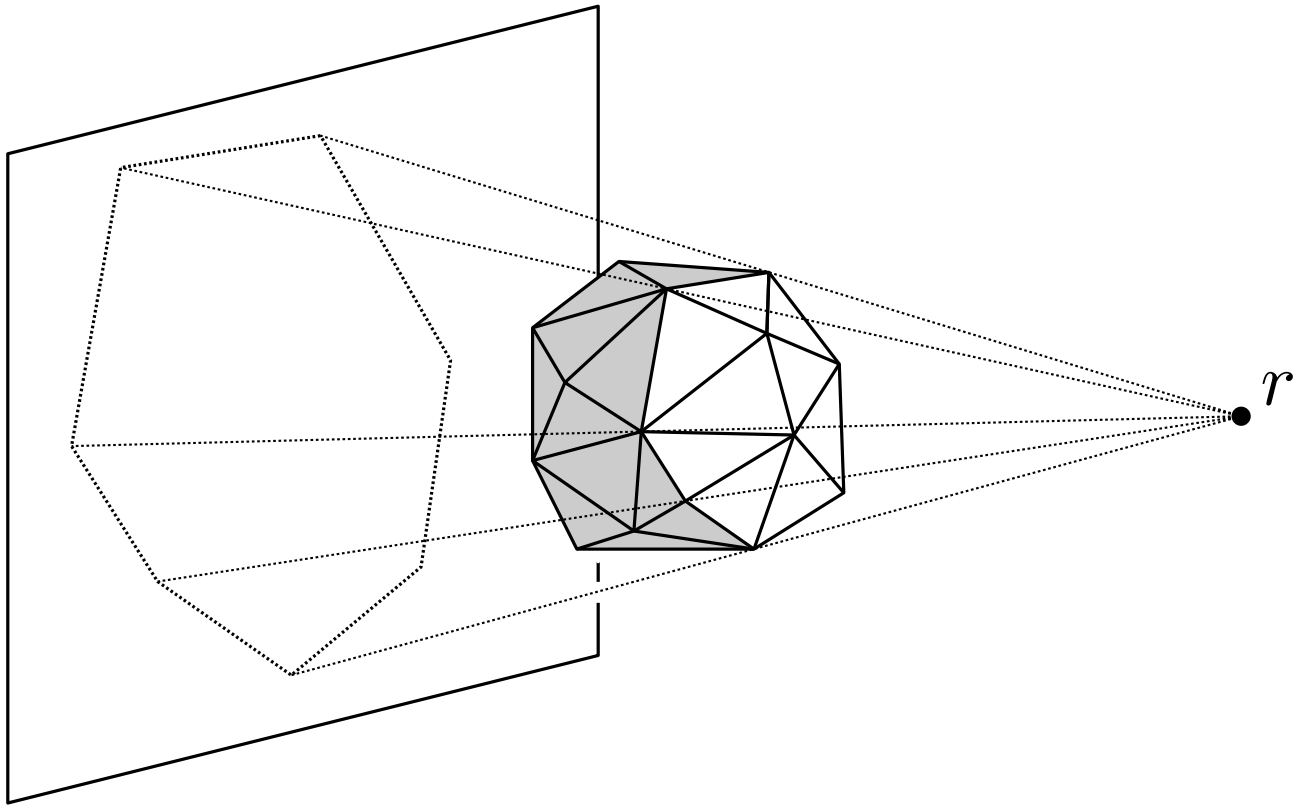


Geht's schneller?

Erster Schritt

Angenommen:

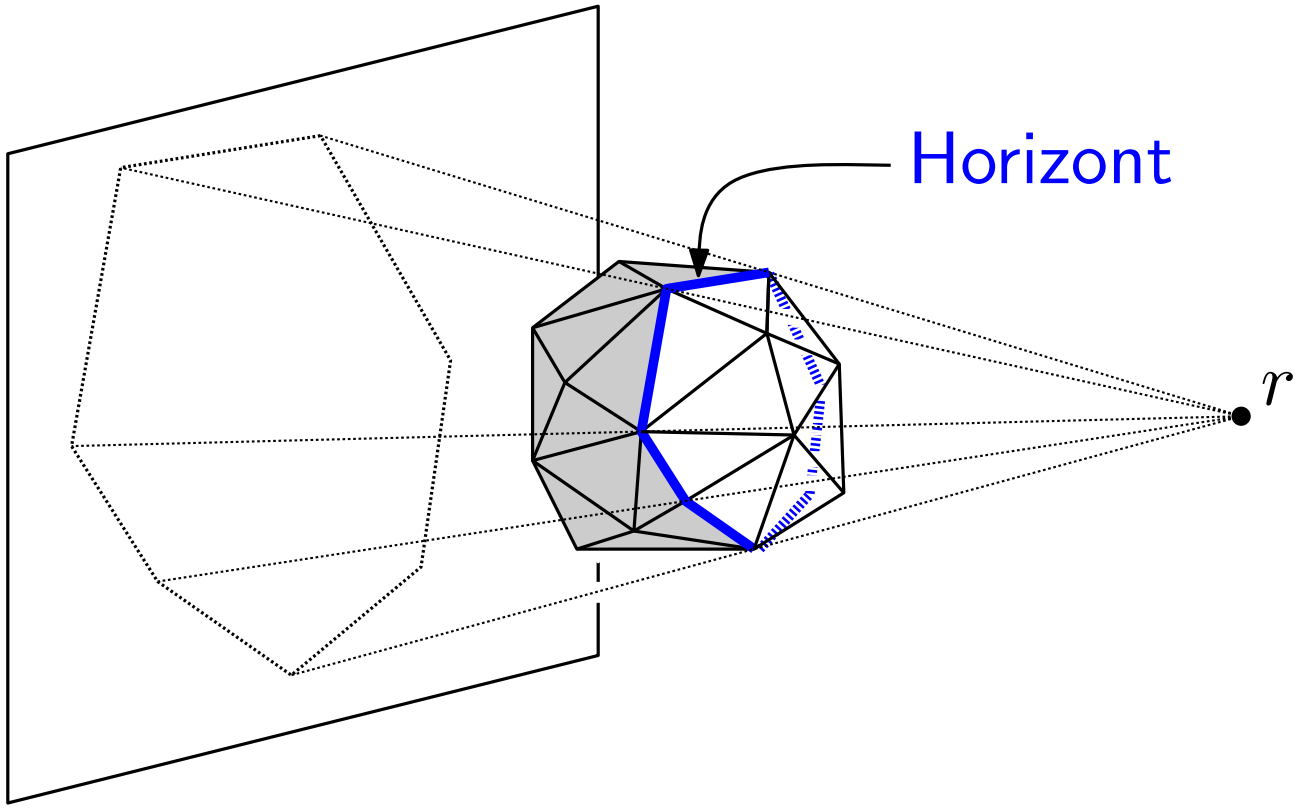
Konvexe Hülle von $r-1$ Knoten gegeben. Wie erweitern zu einer konvexen Hülle für r Knoten?



Erster Schritt

Angenommen:

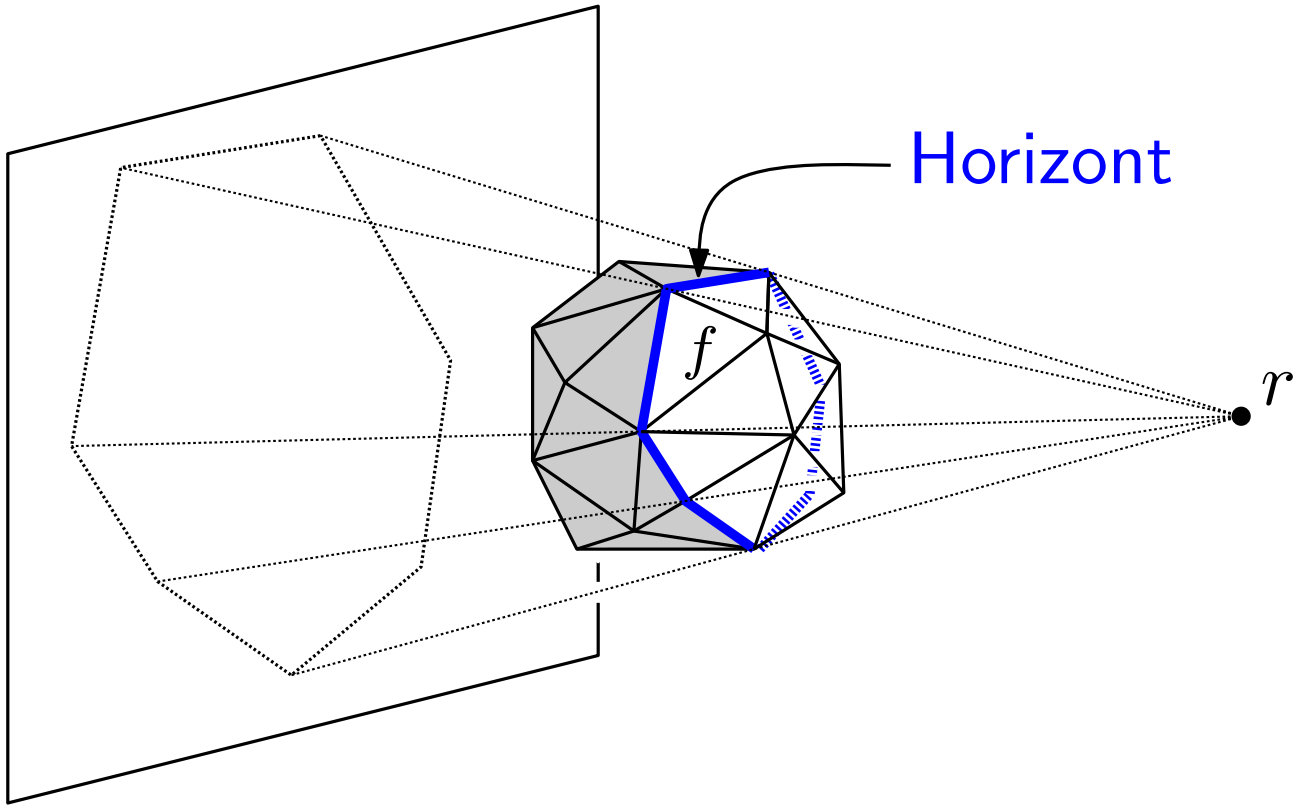
Konvexe Hülle von $r-1$ Knoten gegeben. Wie erweitern zu einer konvexen Hülle für r Knoten?



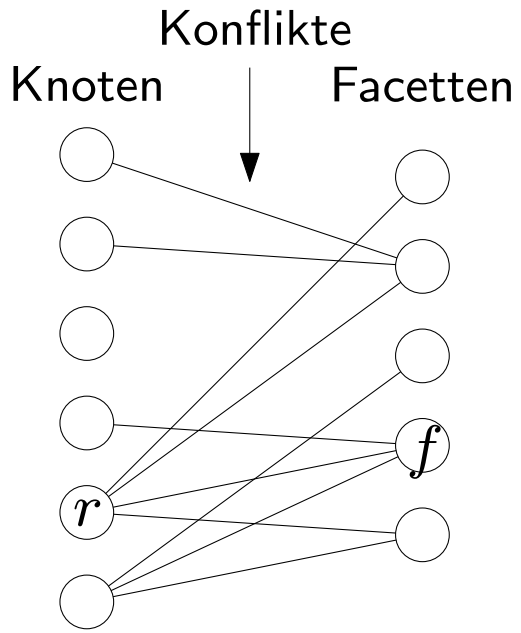
Erster Schritt

Angenommen:

Konvexe Hülle von $r-1$ Knoten gegeben. Wie erweitern zu einer konvexen Hülle für r Knoten?



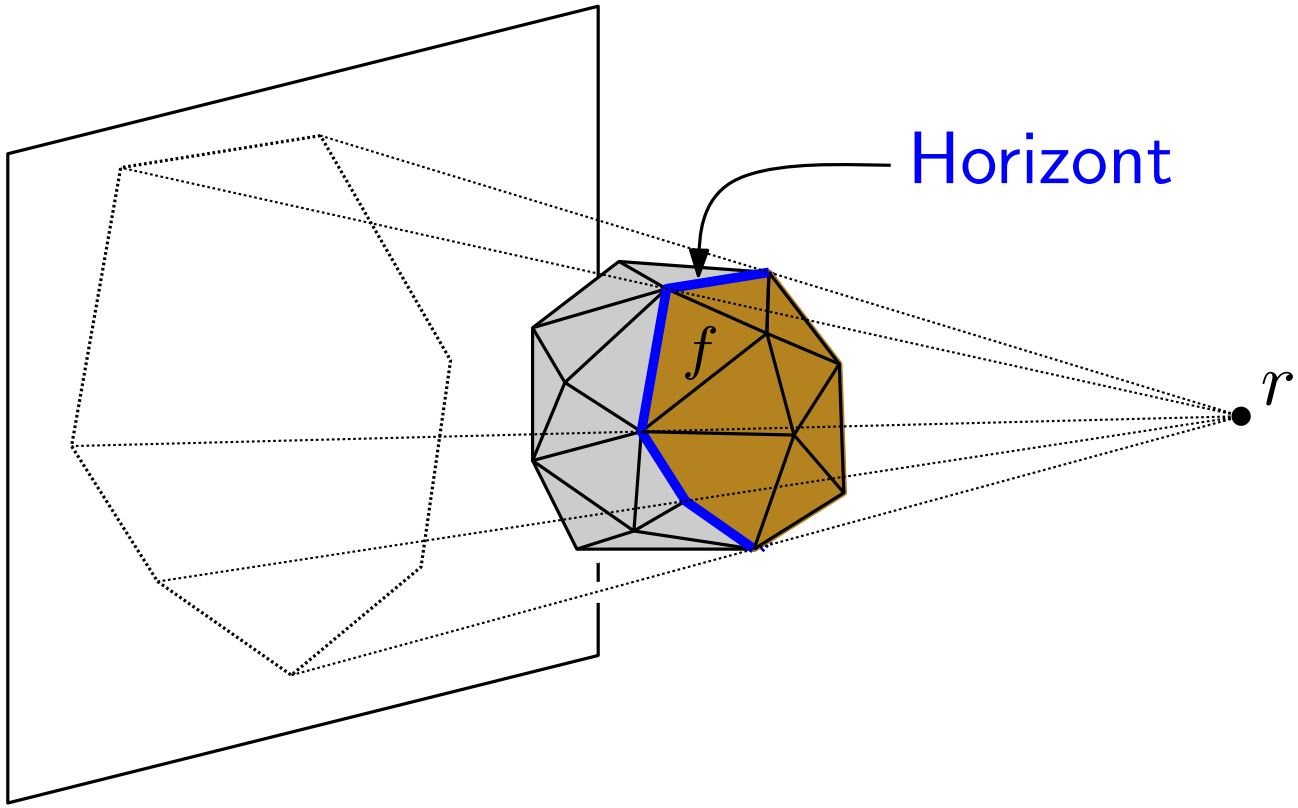
Konfliktgraph G

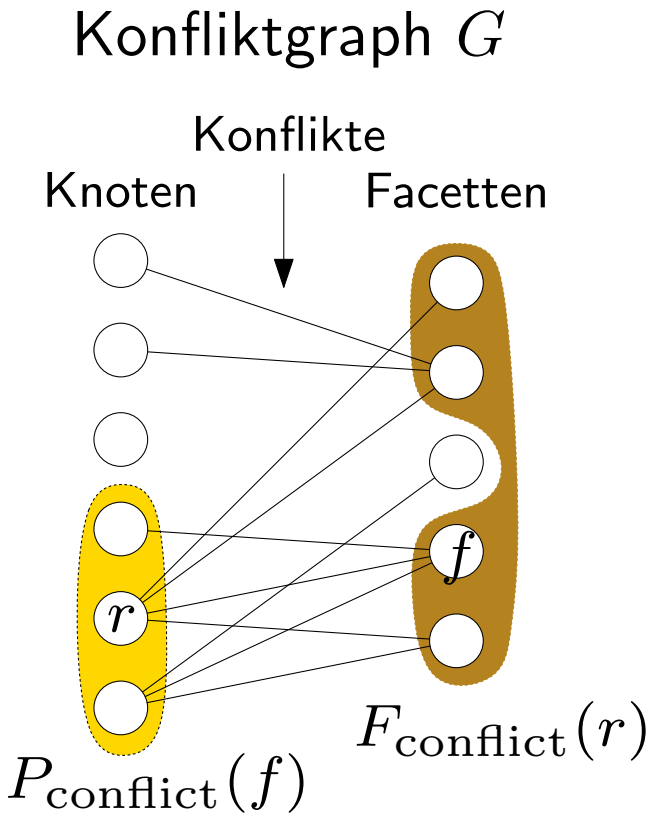


Erster Schritt

Angenommen:

Konvexe Hülle von $r-1$ Knoten gegeben. Wie erweitern zu einer konvexen Hülle für r Knoten?

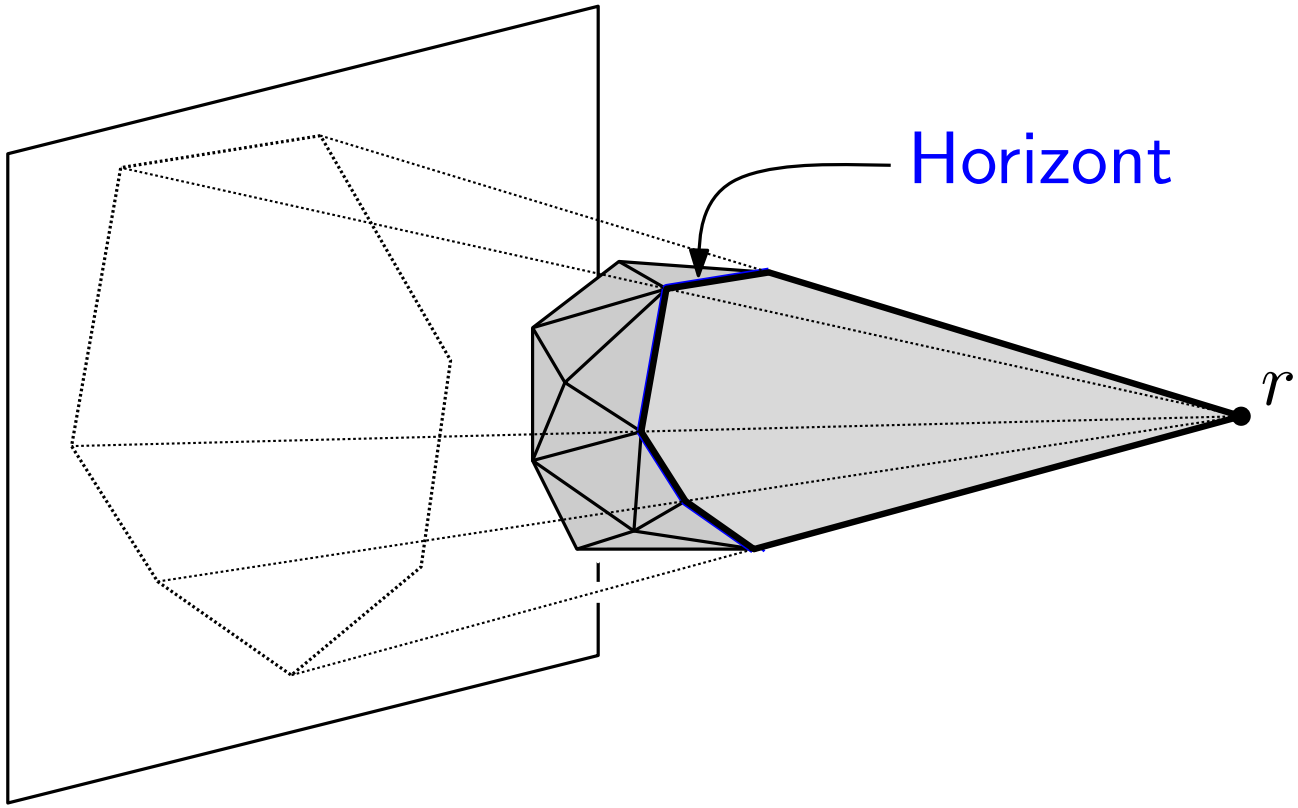


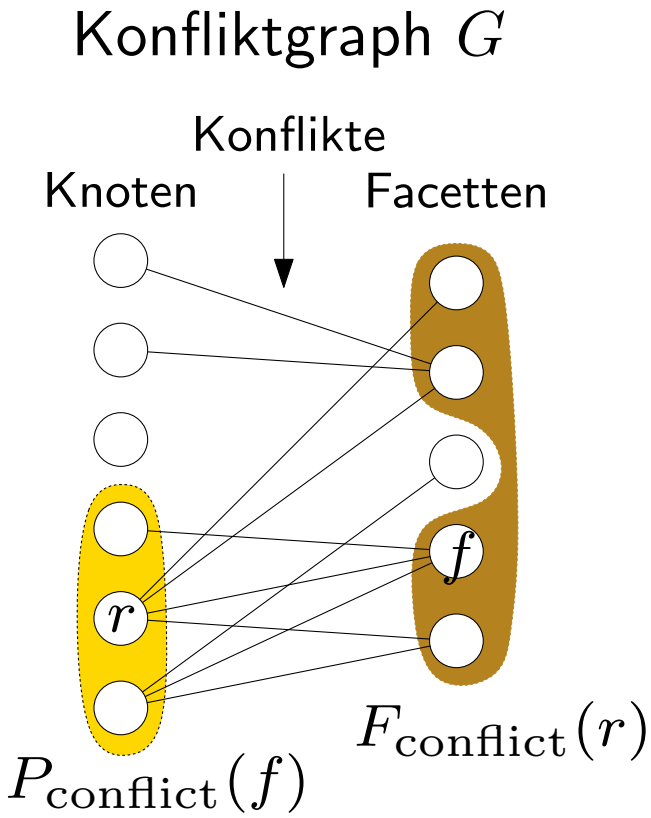


Erster Schritt

Angenommen:

Konvexe Hülle von $r-1$ Knoten gegeben. Wie erweitern zu einer konvexen Hülle für r Knoten?

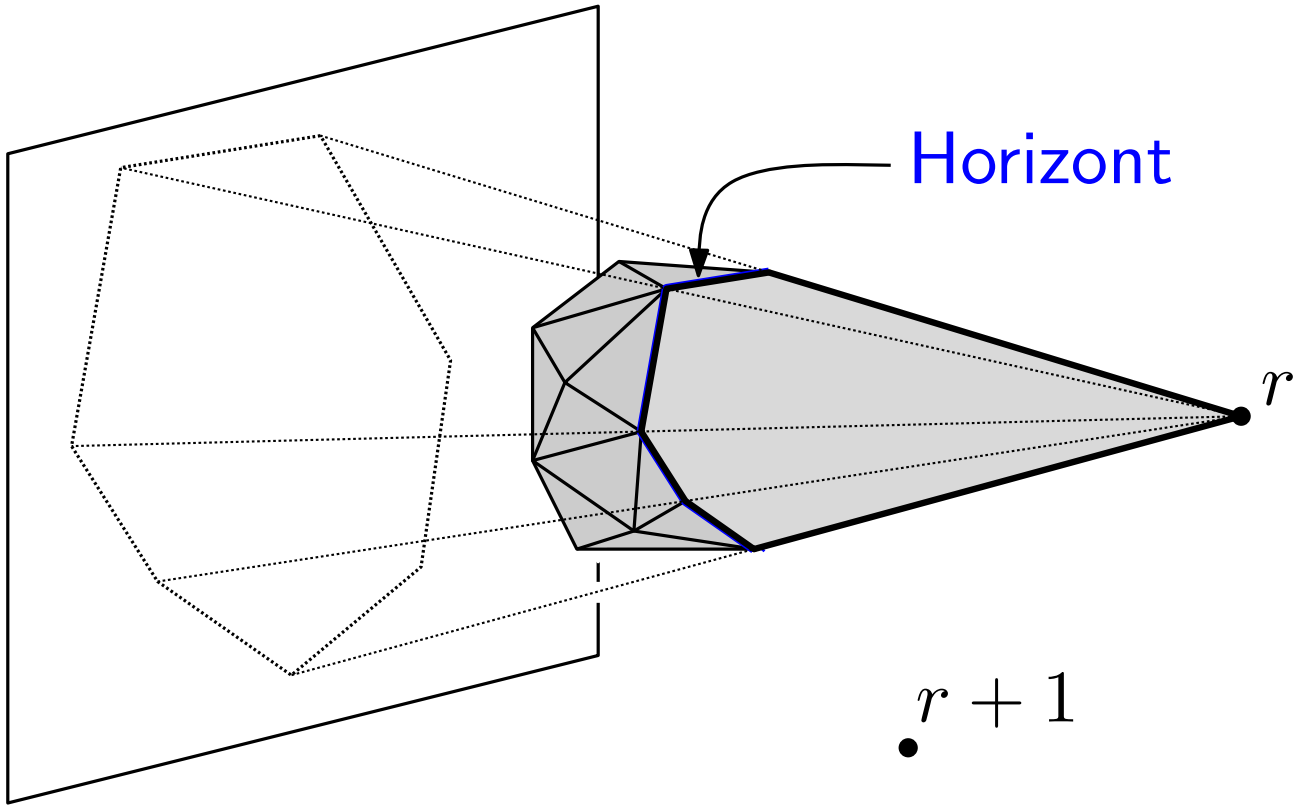


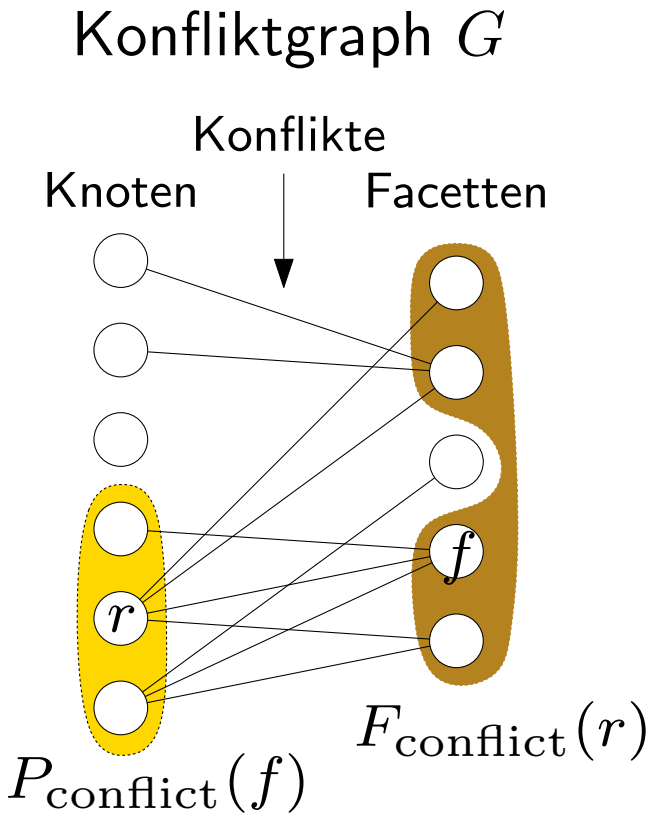


Erster Schritt

Angenommen:

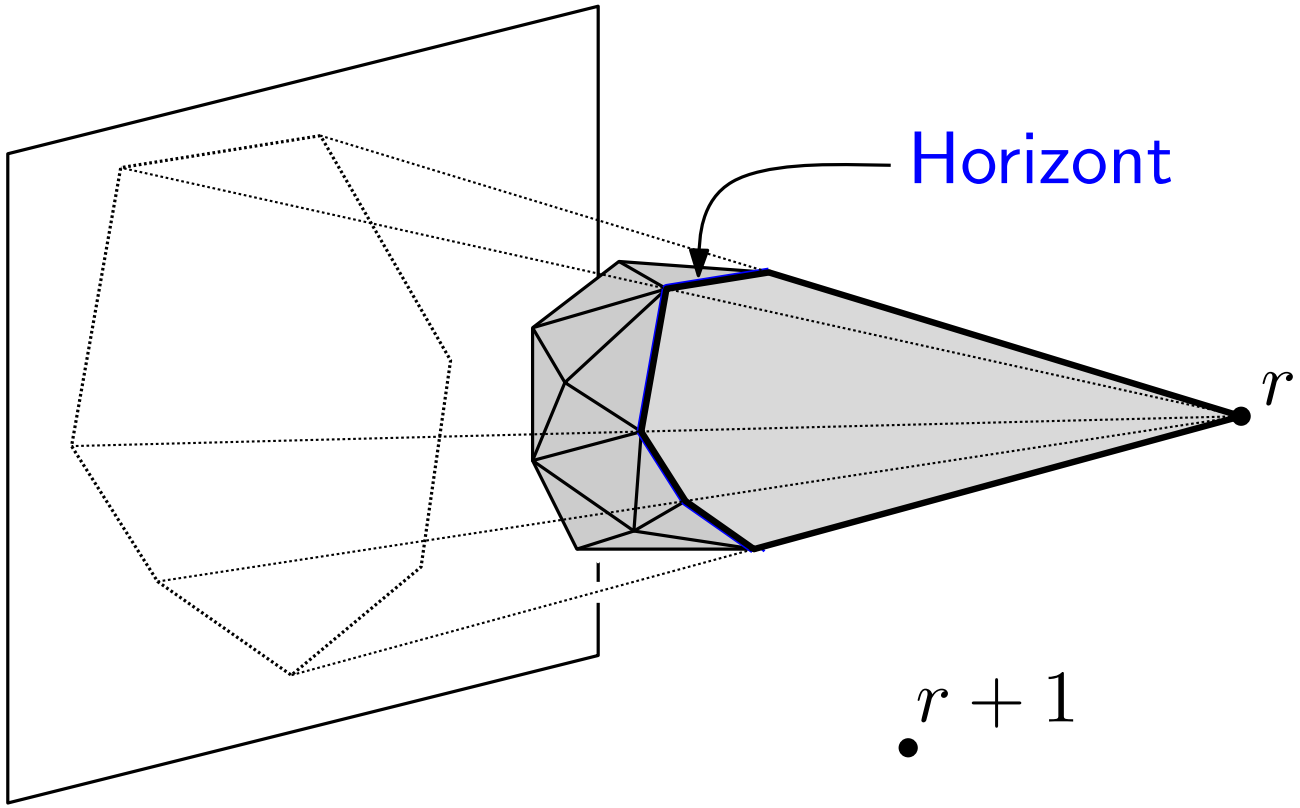
Konvexe Hülle von $r-1$ Knoten gegeben. Wie erweitern zu einer konvexen Hülle für r Knoten?

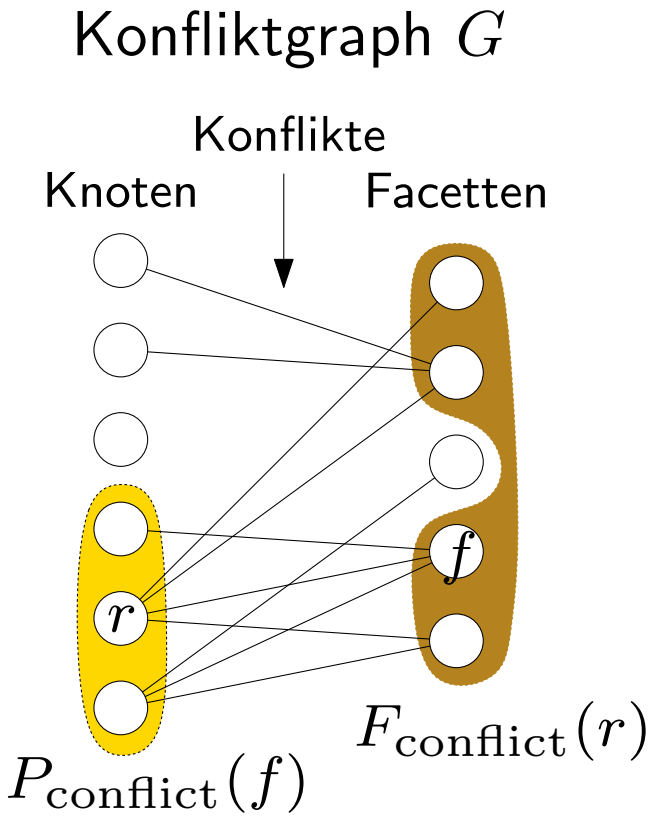




Erster Schritt

Angenommen:
Konvexe Hülle von $r-1$ Knoten gegeben. Wie erweitern zu einer konvexen Hülle für r Knoten?



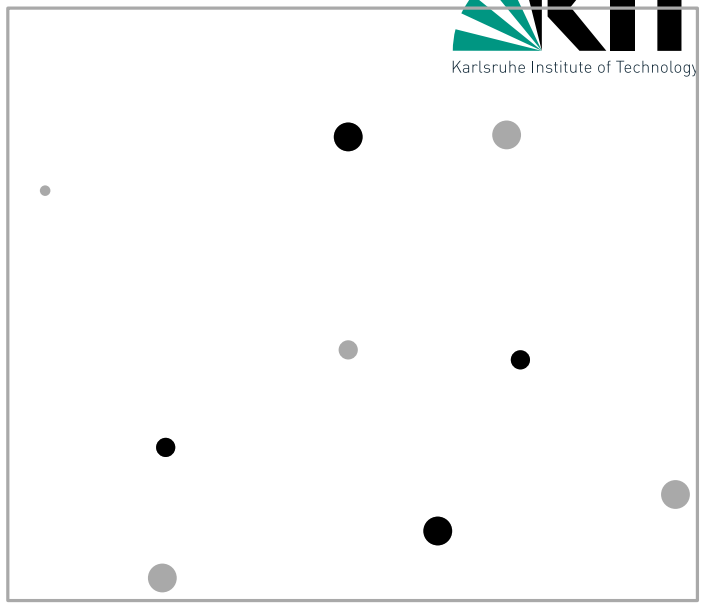


Wie bestimmt man neue Konflikte?

3dConvexHull($P \subset \mathbb{R}^3$)

3dConvexHull($P \subset \mathbb{R}^3$)

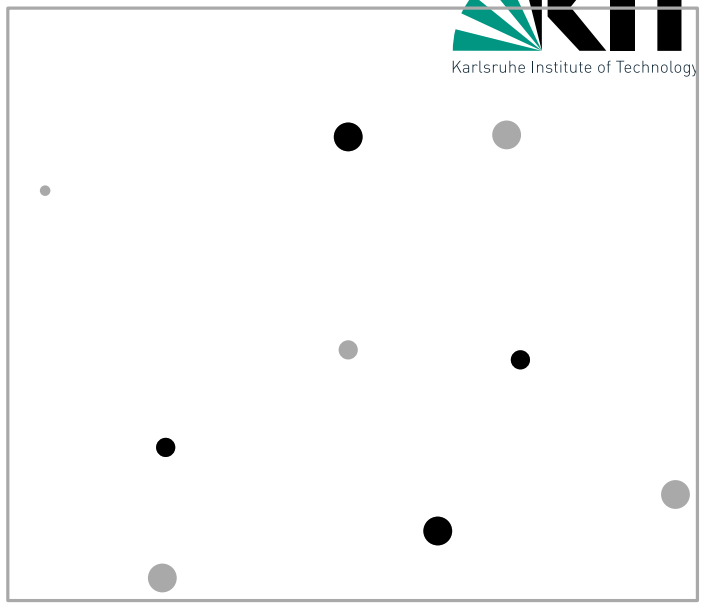
wähle $P' \subseteq P$ (4 nicht koplan. Punkte)



$3\text{DCONVEXHULL}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

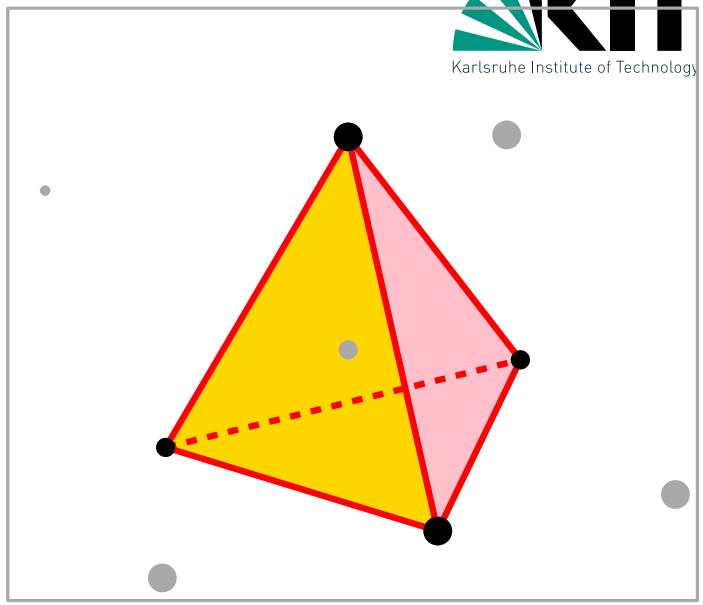
$C \leftarrow \mathsf{CH}(P')$;



$3\text{DCONVEXHULL}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

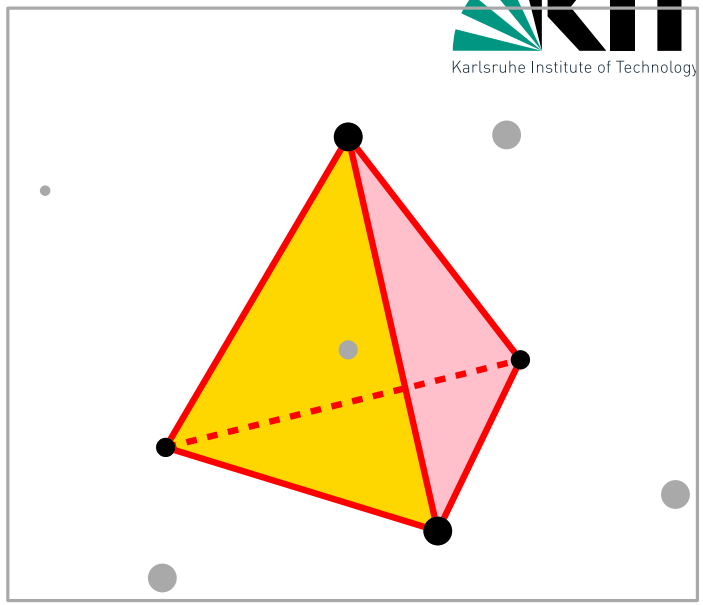
$C \leftarrow \mathsf{CH}(P')$;



3dConvexHull($P \subset \mathbb{R}^3$)

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \mathsf{CH}(P')$; $P \leftarrow P \setminus P'$

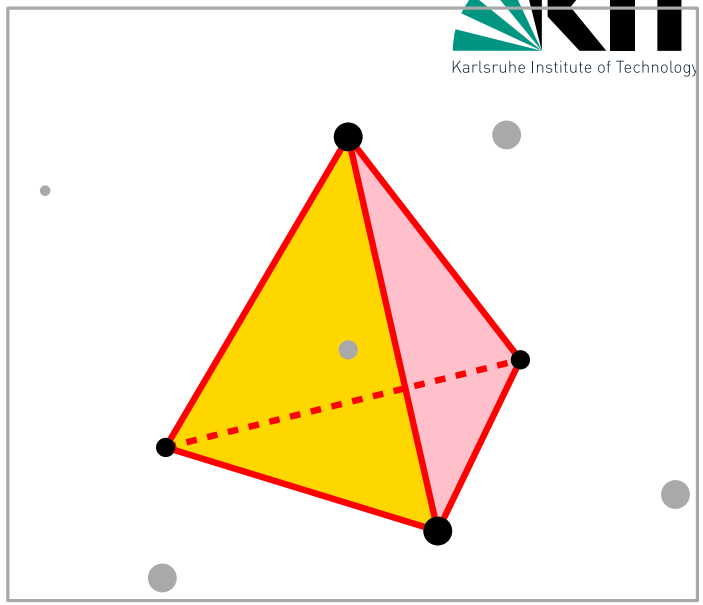


$3\text{DCONVEXHULL}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \mathsf{CH}(P')$; $P \leftarrow P \setminus P'$

$\mathsf{P} = \text{RANDOMPERMUTATION}(\mathrm{P})$

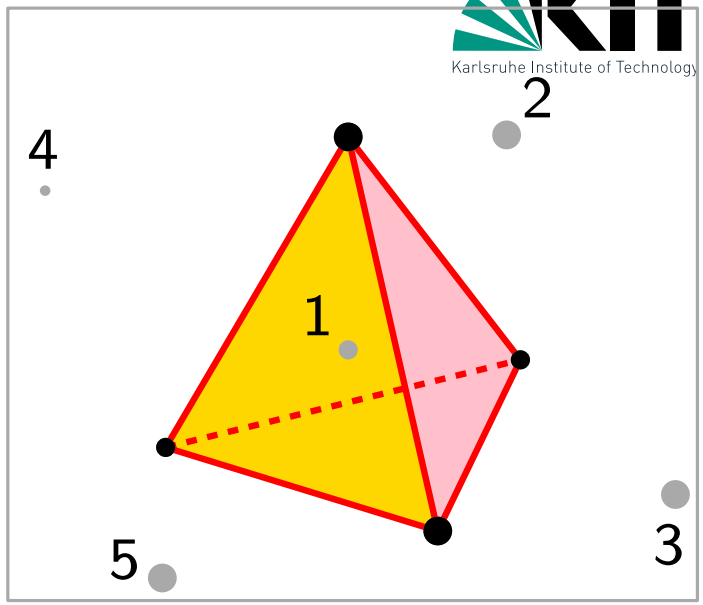


$3\text{DCONVEXHULL}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \mathsf{CH}(P')$; $P \leftarrow P \setminus P'$

$\mathsf{P} = \text{RANDOMPERMUTATION}(\mathrm{P})$



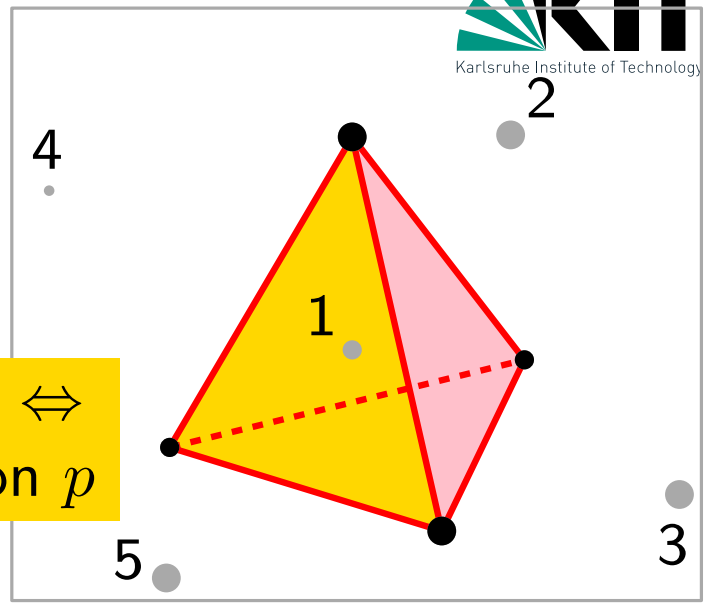
$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \mathsf{CH}(P')$; $P \leftarrow P \setminus P'$

$\mathsf{P} = \text{RandomPermutation}(\mathrm{P})$

initializiere Konfliktgraph G : (p, f) Kante $\Leftrightarrow$ f sichtbar von p



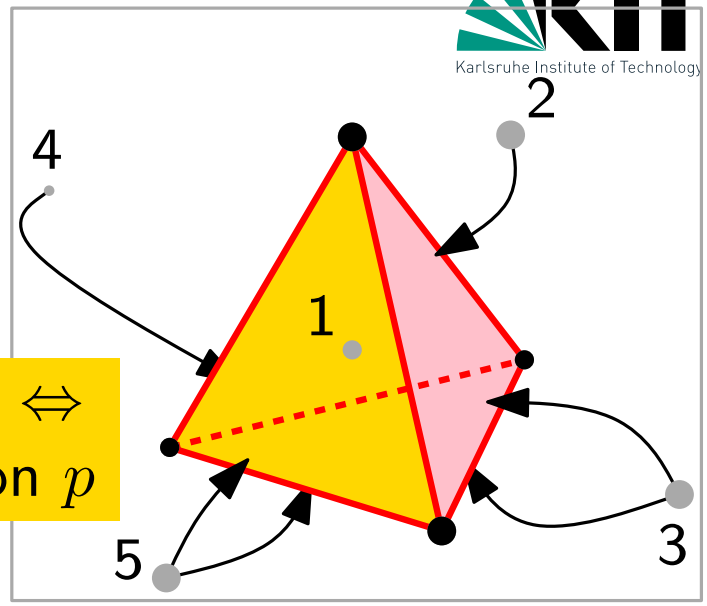
$3\text{DCONVEXHULL}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \mathsf{CH}(P')$; $P \leftarrow P \setminus P'$

$\mathsf{P} = \text{RANDOMPERMUTATION}(\mathrm{P})$

initializiere Konfliktgraph G : (p, f) Kante $\Leftrightarrow$ f sichtbar von p



$3\text{DCONVEXHULL}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

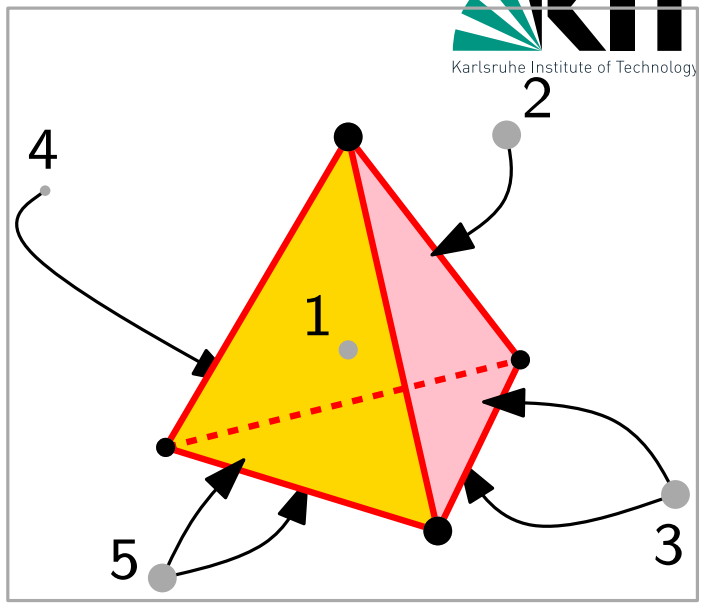
$C \leftarrow \mathsf{CH}(P')$; $P \leftarrow P \setminus P'$

$\mathsf{P} = \text{RANDOMPERMUTATION}(\mathrm{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

$p \leftarrow P$.remove_first()



$3\text{DCONVEXHULL}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

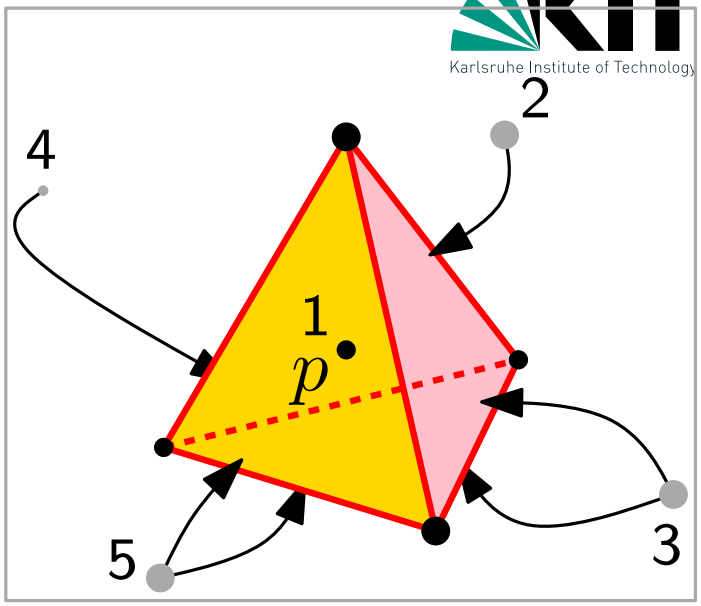
$C \leftarrow \mathsf{CH}(P')$; $P \leftarrow P \setminus P'$

$\mathsf{P} = \text{RANDOMPERMUTATION}(\mathrm{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

$p \leftarrow P$.remove_first()



$3\text{DCONVEXHULL}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \mathsf{CH}(P')$; $P \leftarrow P \setminus P'$

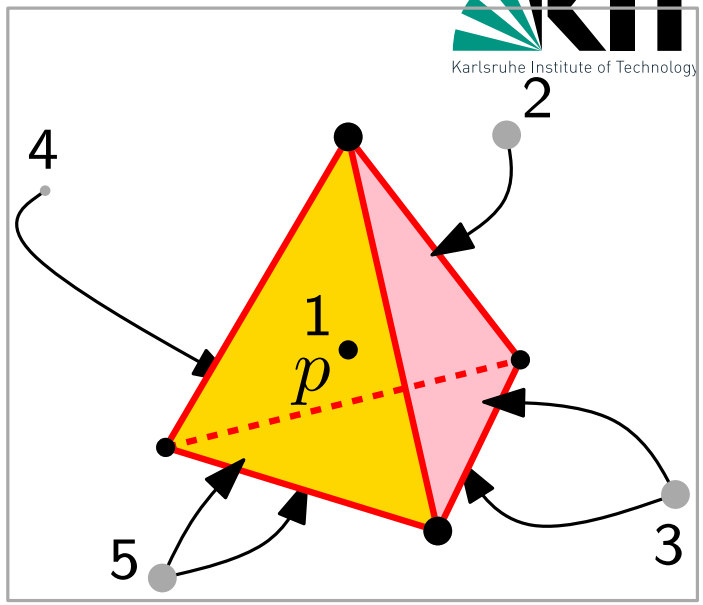
$\mathsf{P} = \text{RANDOMPERMUTATION}(\mathrm{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** //



$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \mathsf{CH}(P')$; $P \leftarrow P \setminus P'$

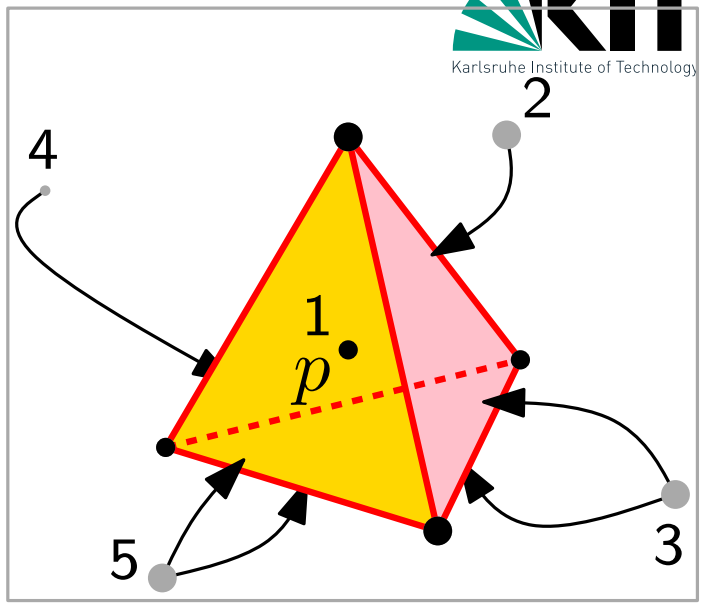
$\mathsf{P} = \text{RandomPermutation}(\mathrm{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C



$3\text{DCONVEXHULL}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \mathsf{CH}(P')$; $P \leftarrow P \setminus P'$

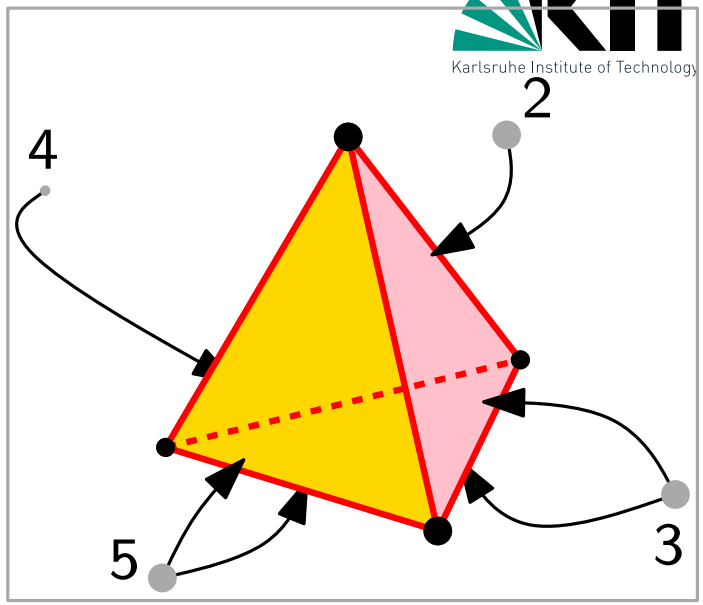
$\mathsf{P} = \text{RANDOMPERMUTATION}(\mathrm{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C



$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \mathsf{CH}(P')$; $P \leftarrow P \setminus P'$

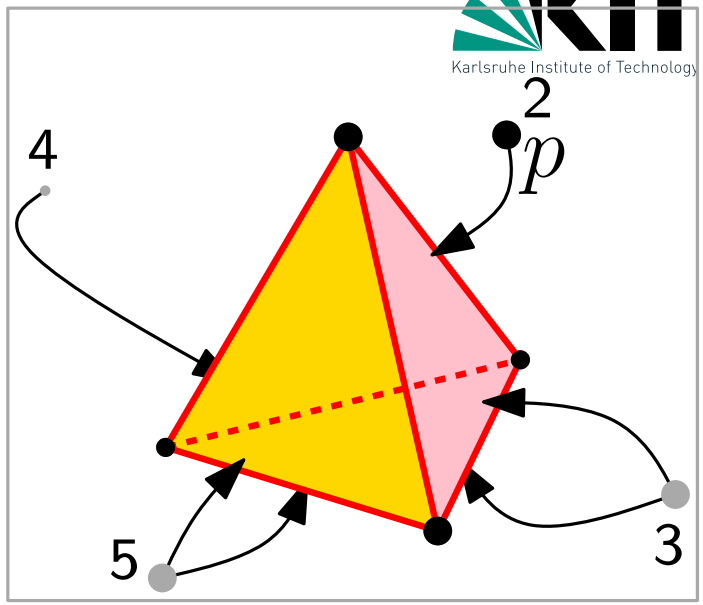
$\mathsf{P} = \text{RandomPermutation}(\mathrm{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C



$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \mathsf{CH}(P')$; $P \leftarrow P \setminus P'$

$\mathsf{P} = \text{RandomPermutation}(\mathrm{P})$

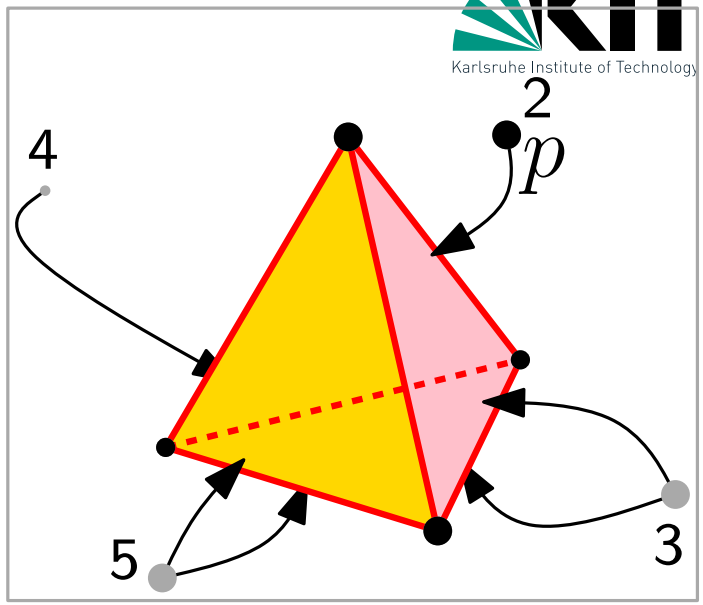
initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C



$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \mathsf{CH}(P')$; $P \leftarrow P \setminus P'$

$\mathsf{P} = \text{RandomPermutation}(\mathrm{P})$

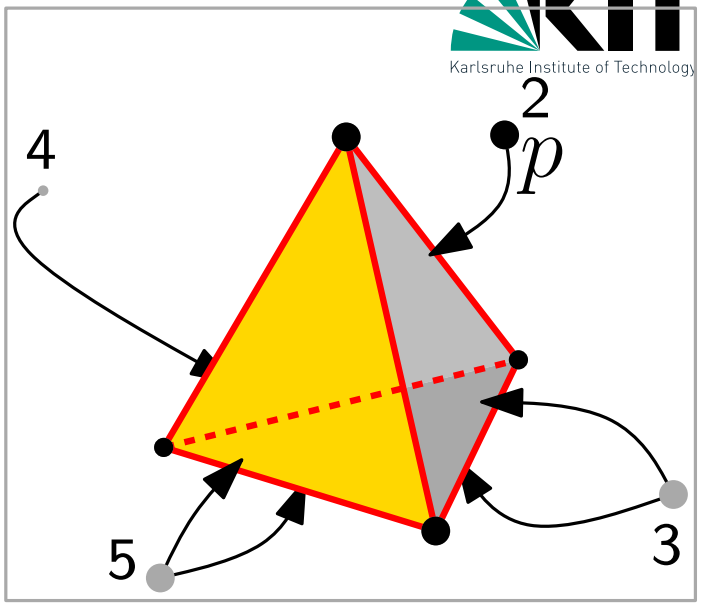
initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C



$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \text{CH}(P')$; $P \leftarrow P \setminus P'$

$\text{P} = \text{RandomPermutation}(\text{P})$

initializiere Konfliktgraph G

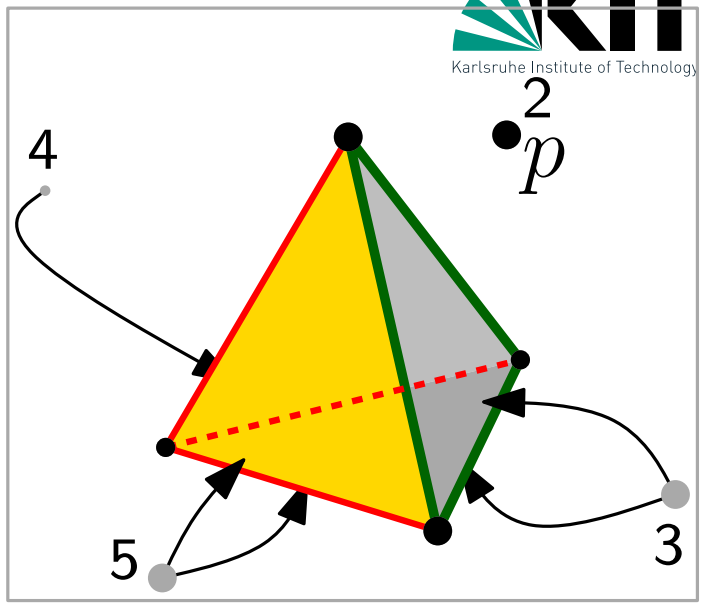
while $P \neq \emptyset$ **do**

$p \leftarrow P$.remove_first()

if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C

$\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind



$3\text{DCONVEXHULL}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \text{CH}(P')$; $P \leftarrow P \setminus P'$

$\text{P} = \text{RANDOMPERMUTATION}(\text{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

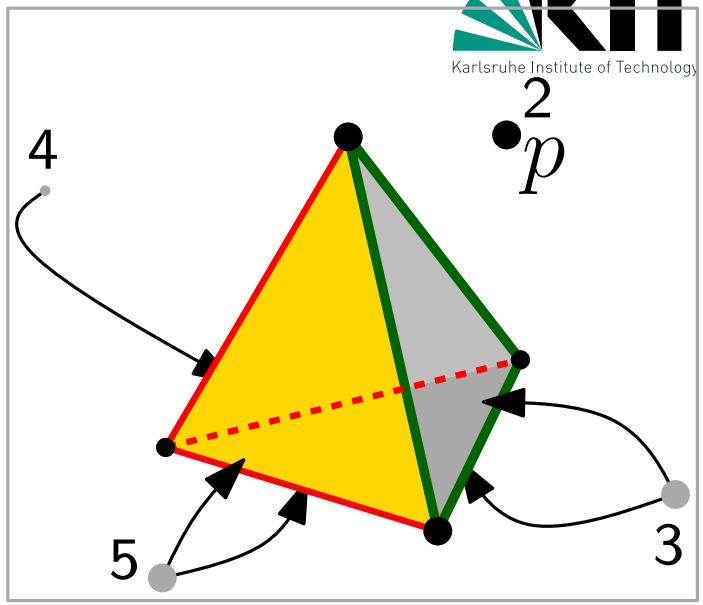
  $p \leftarrow P$.remove_first()

  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C

    $\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind

    foreach $e \in \mathcal{L}$ **do**



$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \text{CH}(P')$; $P \leftarrow P \setminus P'$

$\text{P} = \text{RandomPermutation}(\text{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

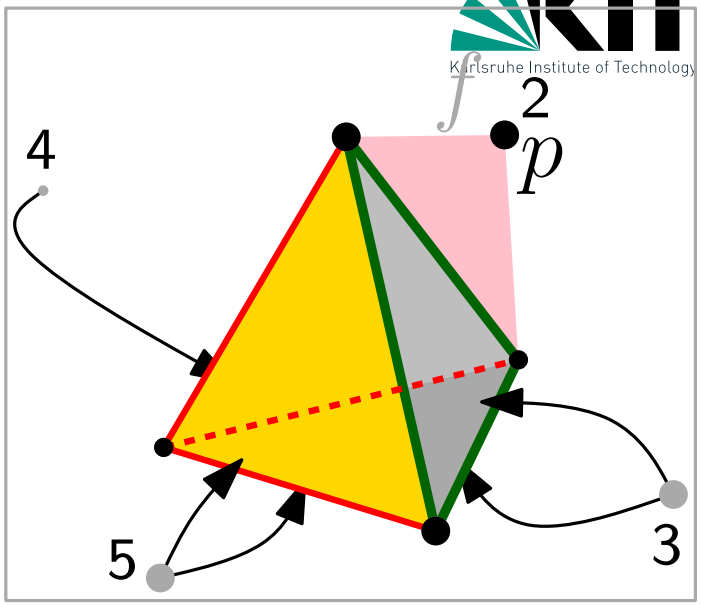
  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C

    $\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind

    foreach $e \in \mathcal{L}$ **do**

      $f \leftarrow$ erzg. Facette$_C(e, p)$; erzg. Knoten für f in G



3dConvexHull($P \subset \mathbb{R}^3$)

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \mathsf{CH}(P')$; $P \leftarrow P \setminus P'$

P = RandomPermutation(P)

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

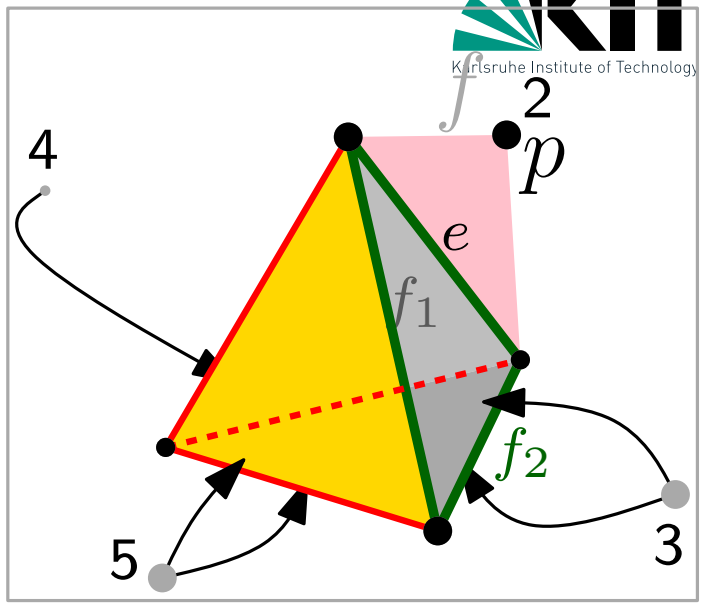
    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C

    $\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind

    foreach $e \in \mathcal{L}$ **do**

      $f \leftarrow$ erzg. Facette$_C(e, p)$; erzg. Knoten für f in G

      $(f_1, f_2) \leftarrow$ vorher_inzident$_C(e)$



$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \text{CH}(P')$; $P \leftarrow P \setminus P'$

$\text{P} = \text{RandomPermutation}(\text{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C

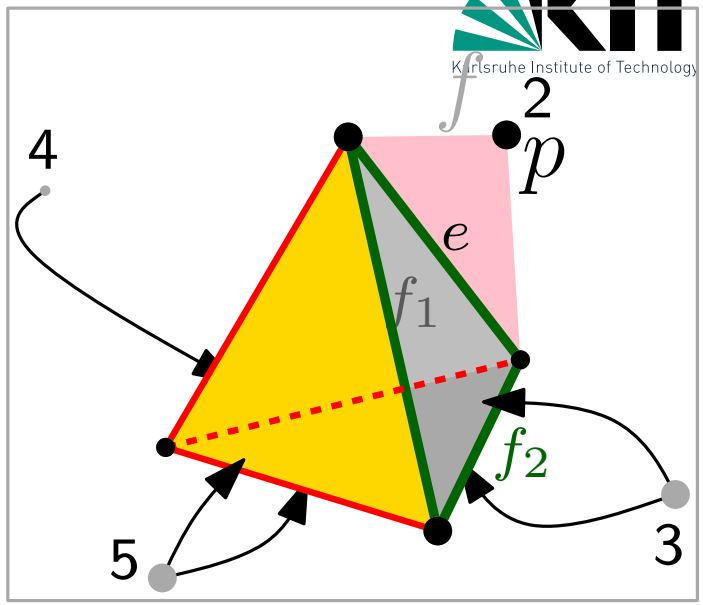
    $\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind

    foreach $e \in \mathcal{L}$ **do**

      $f \leftarrow$ erzg. Facette$_C(e, p)$; erzg. Knoten für f in G

      $(f_1, f_2) \leftarrow$ vorher_inzident$_C(e)$

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



$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \mathsf{CH}(P')$; $P \leftarrow P \setminus P'$

$\mathsf{P} = \text{RandomPermutation}(\mathrm{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C

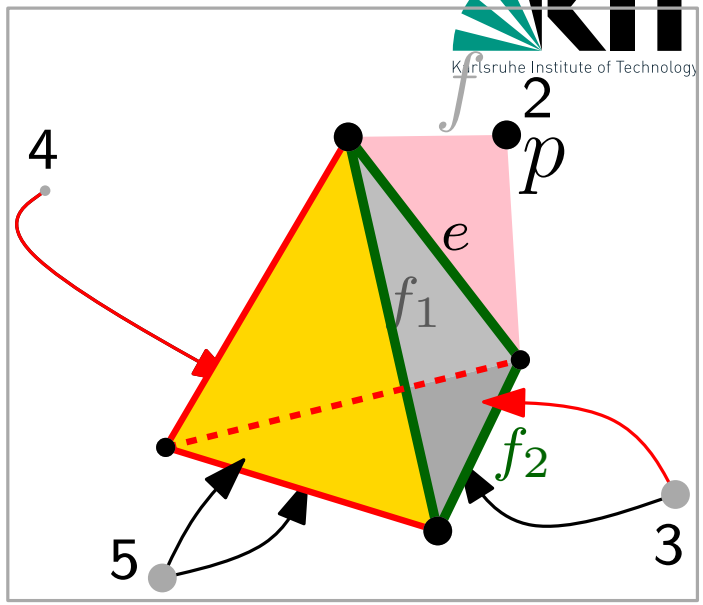
    $\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind

    foreach $e \in \mathcal{L}$ **do**

      $f \leftarrow$ erzg. Facette$_C(e, p)$; erzg. Knoten für f in G

      $(f_1, f_2) \leftarrow$ vorher_inzident$_C(e)$

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



3dConvexHull($P \subset \mathbb{R}^3$)

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \mathsf{CH}(P')$; $P \leftarrow P \setminus P'$

P = RandomPermutation(P)

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C

    $\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind

    foreach $e \in \mathcal{L}$ **do**

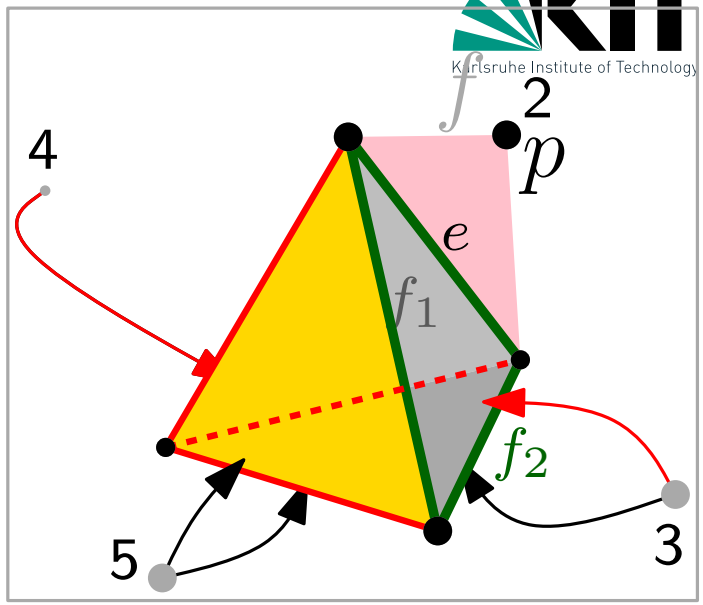
      $f \leftarrow$ erzg. Facette$_C(e, p)$; erzg. Knoten für f in G

      $(f_1, f_2) \leftarrow$ vorher_inzident$_C(e)$

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

      foreach $p \in P(e)$ **do**

        if f sichtbar von p **then** add_edge(p, f) zu G



$\textsc{3dConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \mathsf{CH}(P')$; $P \leftarrow P \setminus P'$

$\mathrm{P} = \textsc{RandomPermutation}(\mathrm{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C

    $\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind

    foreach $e \in \mathcal{L}$ **do**

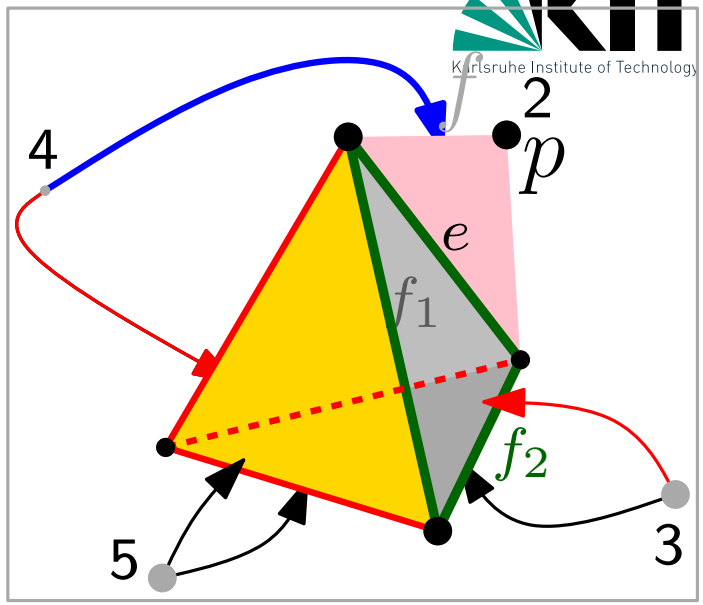
      $f \leftarrow$ erzg. Facette$_C(e, p)$; erzg. Knoten für f in G

      $(f_1, f_2) \leftarrow$ vorher_inzident$_C(e)$

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

      foreach $p \in P(e)$ **do**

        if f sichtbar von p **then** add_edge(p, f) zu G



$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \text{CH}(P')$; $P \leftarrow P \setminus P'$

$\text{P} = \text{RandomPermutation}(\text{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

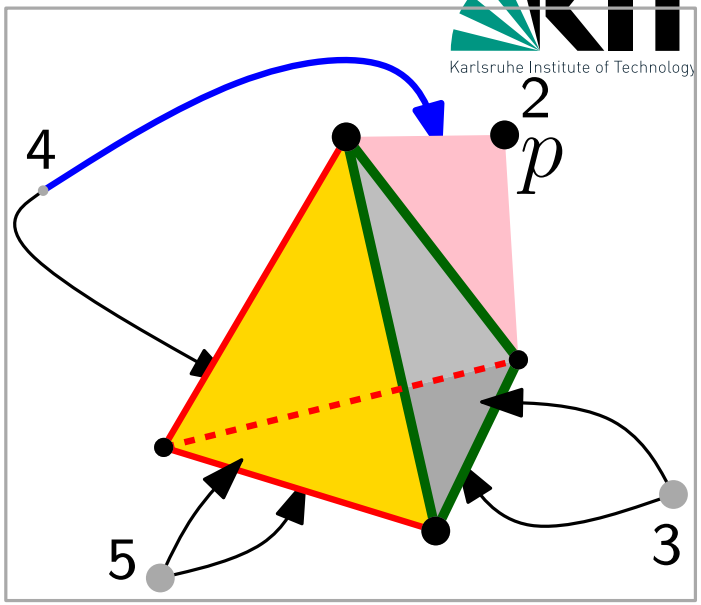
  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C

    $\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind

    foreach $e \in \mathcal{L}$ **do**

      $f \leftarrow$ erzg. Facette$_C(e, p)$; erzg. Knoten für f in G



$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \text{CH}(P')$; $P \leftarrow P \setminus P'$

$\text{P} = \text{RandomPermutation}(\text{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

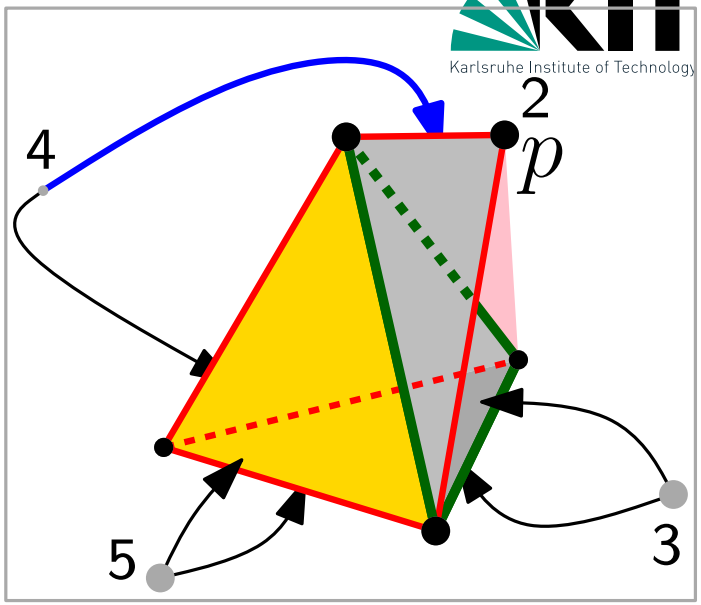
  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C

    $\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind

    foreach $e \in \mathcal{L}$ **do**

      $f \leftarrow$ erzg. Facette$_C(e, p)$; erzg. Knoten für f in G



$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \text{CH}(P')$; $P \leftarrow P \setminus P'$

$\text{P} = \text{RandomPermutation}(\text{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

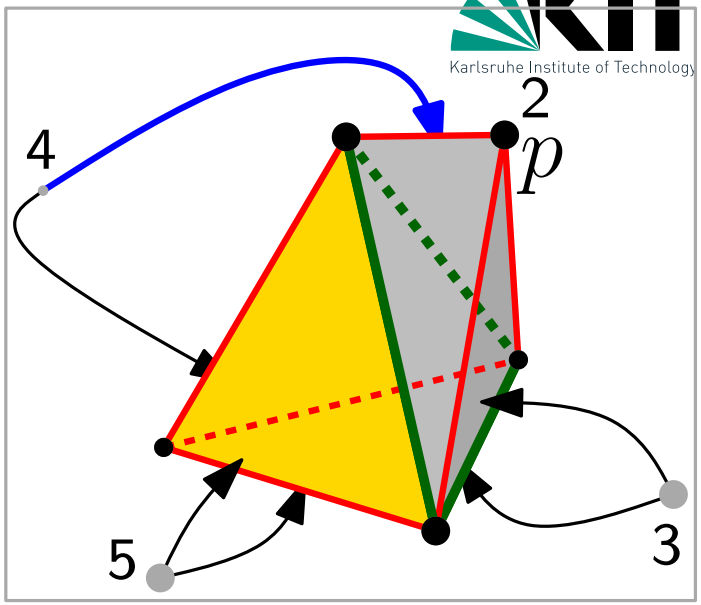
  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C

    $\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind

    foreach $e \in \mathcal{L}$ **do**

      $f \leftarrow$ erzg. Facette$_C(e, p)$; erzg. Knoten für f in G



$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \text{CH}(P')$; $P \leftarrow P \setminus P'$

$\text{P} = \text{RandomPermutation}(\text{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C

    $\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind

    foreach $e \in \mathcal{L}$ **do**

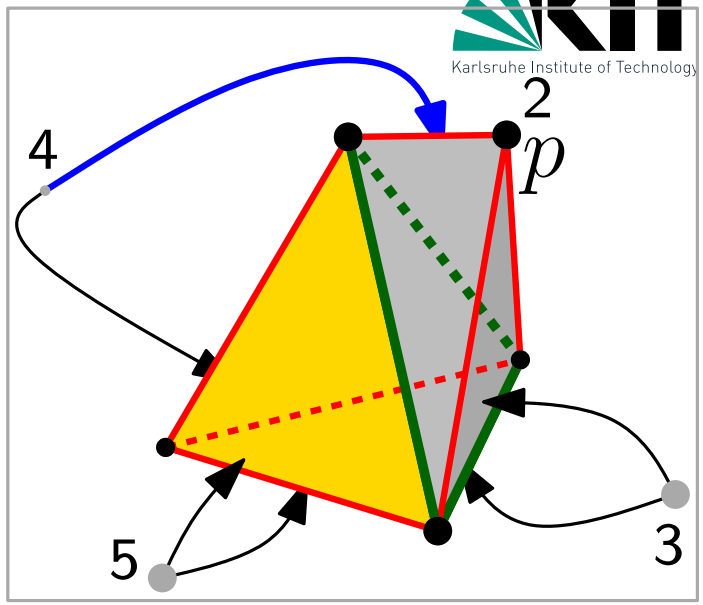
      $f \leftarrow$ erzg. Facette$_C(e, p)$; erzg. Knoten für f in G

      $(f_1, f_2) \leftarrow$ vorher_inzident$_C(e)$

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

      foreach $p \in P(e)$ **do**

        if f sichtbar von p **then** add_edge(p, f) zu G



$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \text{CH}(P')$; $P \leftarrow P \setminus P'$

$\text{P} = \text{RandomPermutation}(\text{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C

    $\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind

    foreach $e \in \mathcal{L}$ **do**

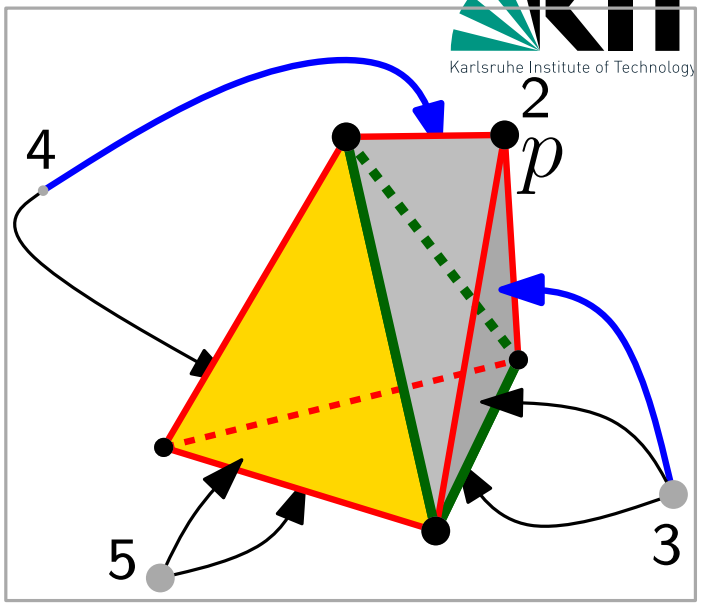
      $f \leftarrow$ erzg. Facette$_C(e, p)$; erzg. Knoten für f in G

      $(f_1, f_2) \leftarrow$ vorher_inzident$_C(e)$

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

      foreach $p \in P(e)$ **do**

        if f sichtbar von p **then** add_edge(p, f) zu G



$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \text{CH}(P')$; $P \leftarrow P \setminus P'$

$\text{P} = \text{RandomPermutation}(\text{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C

    $\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind

    foreach $e \in \mathcal{L}$ **do**

      $f \leftarrow$ erzg. Facette$_C(e, p)$; erzg. Knoten für f in G

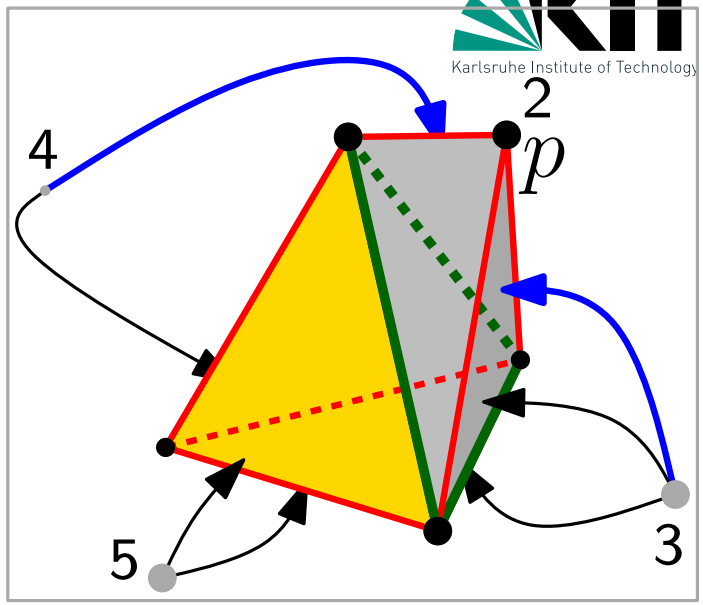
      $(f_1, f_2) \leftarrow$ vorher_inzident$_C(e)$

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

      foreach $p \in P(e)$ **do**

        if f sichtbar von p **then** add_edge(p, f) zu G

    lösche Knoten $\{p\} \cup F_{\text{conflict}}(p)$ von G



$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \text{CH}(P')$; $P \leftarrow P \setminus P'$

$\text{P} = \text{RandomPermutation}(\text{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C

    $\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind

    foreach $e \in \mathcal{L}$ **do**

      $f \leftarrow$ erzg. Facette$_C(e, p)$; erzg. Knoten für f in G

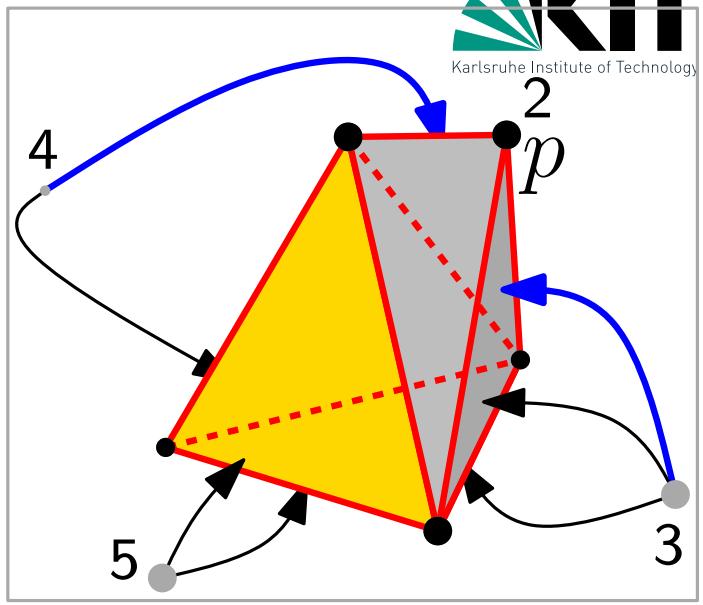
      $(f_1, f_2) \leftarrow$ vorher_inzident$_C(e)$

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

      foreach $p \in P(e)$ **do**

        if f sichtbar von p **then** add_edge(p, f) zu G

    lösche Knoten $\{p\} \cup F_{\text{conflict}}(p)$ von G



$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \text{CH}(P')$; $P \leftarrow P \setminus P'$

$\text{P} = \text{RandomPermutation}(\text{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C

    $\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind

    foreach $e \in \mathcal{L}$ **do**

      $f \leftarrow$ erzg. Facette$_C(e, p)$; erzg. Knoten für f in G

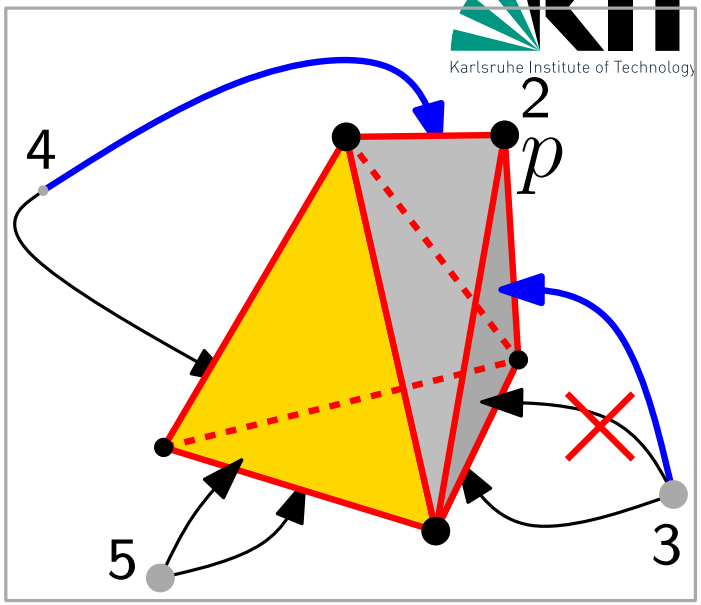
      $(f_1, f_2) \leftarrow$ vorher_inzident$_C(e)$

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

      foreach $p \in P(e)$ **do**

        if f sichtbar von p **then** add_edge(p, f) zu G

    lösche Knoten $\{p\} \cup F_{\text{conflict}}(p)$ von G



$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \text{CH}(P')$; $P \leftarrow P \setminus P'$

$\text{P} = \text{RandomPermutation}(\text{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

  $p \leftarrow P$.remove_first()

  if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C

    $\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind

    foreach $e \in \mathcal{L}$ **do**

      $f \leftarrow$ erzg. Facette$_C(e, p)$; erzg. Knoten für f in G

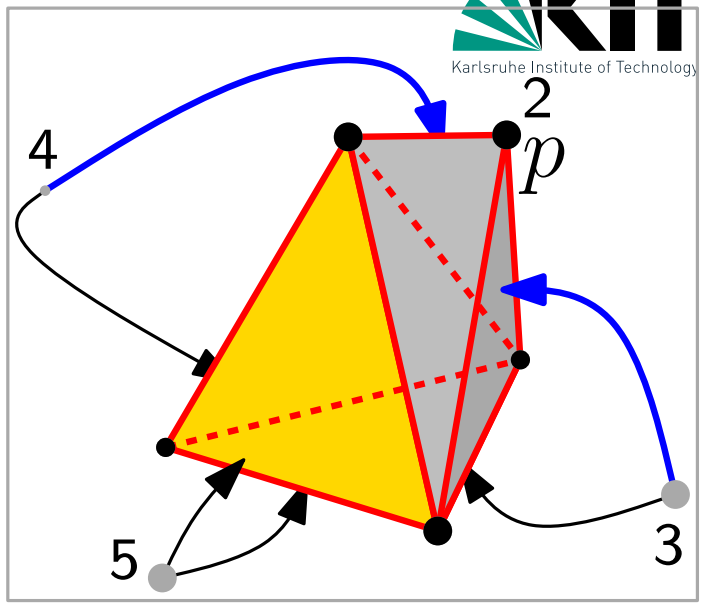
      $(f_1, f_2) \leftarrow$ vorher_inzident$_C(e)$

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

      foreach $p \in P(e)$ **do**

        if f sichtbar von p **then** add_edge(p, f) zu G

    lösche Knoten $\{p\} \cup F_{\text{conflict}}(p)$ von G



$3\text{DConvexHull}(P \subset \mathbb{R}^3)$

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)

$C \leftarrow \text{CH}(P')$; $P \leftarrow P \setminus P'$

$\text{P} = \text{RandomPermutation}(\text{P})$

initializiere Konfliktgraph G

while $P \neq \emptyset$ **do**

$p \leftarrow P$.remove_first()

if $F_{\text{conflict}}(p) \neq \emptyset$ **then** // p liegt außerhalb von C

Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C

$\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind

foreach $e \in \mathcal{L}$ **do**

$f \leftarrow$ erzg. Facette$_C(e, p)$; erzg. Knoten für f in G

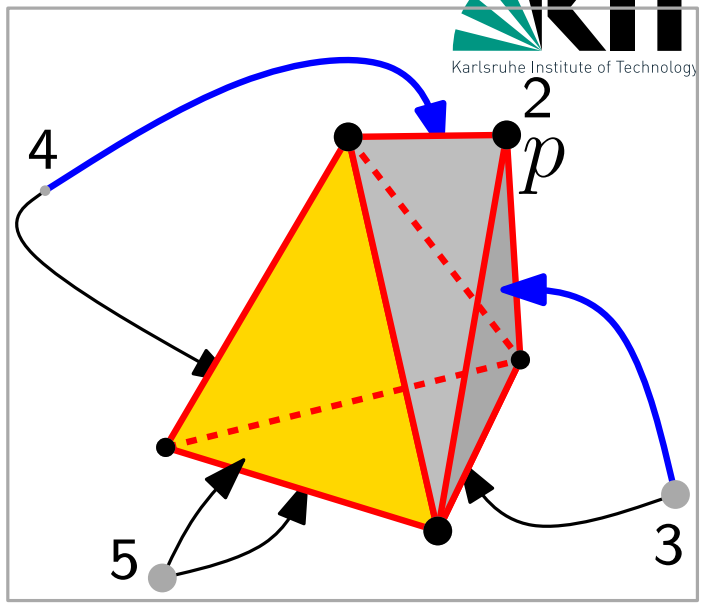
$(f_1, f_2) \leftarrow$ vorher_inzident$_C(e)$

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

foreach $p \in P(e)$ **do**

if f sichtbar von p **then** add_edge(p, f) zu G

lösche Knoten $\{p\} \cup F_{\text{conflict}}(p)$ von G



Analyse

Analyse

Idee:

Begrenze die strukturelle Änderung
Das bedeutet: Begrenze #Facetten, die der Algorithmus erzeugt.

Analyse

Idee: Begrenze die strukturelle Änderung
Das bedeutet: Begrenze #Facetten, die der Algorithmus erzeugt.

Lemma: Die erwartete Anzahl an erzeugen Facetten ist höchstens $6n - 20$.

Analyse

Idee: Begrenze die strukturelle Änderung
Das bedeutet: Begrenze #Facetten, die der Algorithmus erzeugt.

Lemma: Die erwartete Anzahl an erzeugen Facetten ist höchstens $6n - 20$.

Theorem: Die konvexe Hülle von einer Menge von n Punkten im $\mathbb{R}^3$ kann in erwartet $O(n \log n)$ Zeit berechnet werden.

Thm
3dConvexHull($P \subset \mathbb{R}^3$)

wähle $P' \subseteq P$ (4 nicht koplan. Punkte)
$C \leftarrow \mathsf{CH}(P')$; $P \leftarrow P \setminus P'$
P = RandomPermutation(P)
initializiere Konfliktgraph G
while $P \neq \emptyset$ **do**
  $p \leftarrow P$.remove_first()
  if $F_{\text{conflict}}(p) \neq \emptyset$ **then**
    Lösche alle Facetten $F_{\text{conflict}}(p)$ aus C
    $\mathcal{L} \leftarrow$ Horizont-Kanten die von p aus sichtbar sind
    foreach $e \in \mathcal{L}$ **do**
      $f \leftarrow$ erzg. Facette$_C(e, p)$; erzg. Knoten für f in G
      $(f_1, f_2) \leftarrow$ vorher_inzident$_C(e)$
      $P(e) \leftarrow P_{\text{conflict}}(f_1) \cup P_{\text{conflict}}(f_2)$
      foreach $p \in P(e)$ **do**
        if f sichtbar von p **then** add_edge(p, f) zu G
    lösche Knoten $\{p\} \cup F_{\text{conflict}}(p)$ von G

Diskussion

Optimale Laufzeit?

Diskussion

Optimale Laufzeit?

Divide-and-Conquer Algorithmus mit Laufzeit $\mathcal{O}(n \log n)$

Diskussion

Optimale Laufzeit?

Divide-and-Conquer Algorithmus mit Laufzeit $\mathcal{O}(n \log n)$

Was ist mit Dimensionen > 3 ?

Diskussion

Optimale Laufzeit?

Divide-and-Conquer Algorithmus mit Laufzeit $\mathcal{O}(n \log n)$

Was ist mit Dimensionen > 3 ?

Durch das *Upper Bound Theorem* ist bekannt, dass die Komplexität der konvexen Hülle einer Punktmenge bestehend aus n Punkten im d-dimensionalen Raum in $\Theta(n^{\lfloor d/2 \rfloor})$ ist. Der hier vorgestellte Algorithmus kann für höhere Dimensionen angepasst werden.
Der beste Ausgabe-Sensitive Algorithmus für die Berechnung konvexer Hüllen im $\mathbb{R}^d$ hat eine Laufzeit von:

Diskussion

Optimale Laufzeit?

Divide-and-Conquer Algorithmus mit Laufzeit $\mathcal{O}(n \log n)$

Was ist mit Dimensionen > 3 ?

Durch das *Upper Bound Theorem* ist bekannt, dass die Komplexität der konvexen Hülle einer Punktmenge bestehend aus n Punkten im d-dimensionalen Raum in $\Theta(n^{\lfloor d/2 \rfloor})$ ist. Der hier vorgestellte Algorithmus kann für höhere Dimensionen angepasst werden.
Der beste Ausgabe-Sensitive Algorithmus für die Berechnung konvexer Hüllen im $\mathbb{R}^d$ hat eine Laufzeit von:

$$\mathcal{O}(n \log k + (nk)^{1-1/(\lfloor d/2 \rfloor + 1)} \log^{\mathcal{O}(1)} n)$$

Diskussion

Optimale Laufzeit?

Divide-and-Conquer Algorithmus mit Laufzeit $\mathcal{O}(n \log n)$

Was ist mit Dimensionen > 3 ?

Durch das *Upper Bound Theorem* ist bekannt, dass die Komplexität der konvexen Hülle einer Punktmenge bestehend aus n Punkten im d-dimensionalen Raum in $\Theta(n^{\lfloor d/2 \rfloor})$ ist. Der hier vorgestellte Algorithmus kann für höhere Dimensionen angepasst werden.
Der beste Ausgabe-Sensitive Algorithmus für die Berechnung konvexer Hüllen im $\mathbb{R}^d$ hat eine Laufzeit von:

$$\mathcal{O}(n \log k + (nk)^{1-1/(\lfloor d/2 \rfloor + 1)} \log^{\mathcal{O}(1)} n)$$

Bis übermorgen!