Karlsruhe Institute of Technology

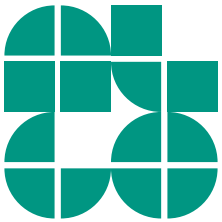
Übung Algorithmische Geometrie

Konvexe Hülle im $\mathbb{R}^3$

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

Andreas Gemsa

19.05.2011



Zwei Kleinigkeiten

Karlsruhe Institute of Technology

Aufgabe 1: b) $\Omega(\sqrt{n})$ statt $\mathcal{O}(\sqrt{n})$

Beweis aus der Vorlesung *hochverdächtig*

Ablauf

Karlsruhe Institute of Technology

Übungsblatt

Neuer Algorithmus

Aufgabe 1

Worst Case Laufzeit

a) Zeige, dass die worst-case Laufzeit von ConvexHull in $\mathcal{O}(n^3)$ liegt.

```
3dConvexHull(P ⊂ ℝ³)
    wähle P' ⊆ P (4 nicht koplan. Punkte)
    C ← CH(P'); P ← P \ P'
    P = RandomPermutation(P)
    initializiere Konfliktgraph G
    while P ≠ ∅ do
        p ← P.remove_first()
        if F_conflict(p) ≠ ∅ then
            Lösche alle Facetten F_conflict(p) aus C
            ℒ ← Horizont-Kanten die von p aus sichtbar sind
            foreach e ∈ ℒ do
                f ← erzg. Facette_C(e, p); erzg. Knoten für f in G
                (f1, f2) ← vorher_inzident_C(e)
                P(e) ← P_conflict(f1) ∪ P_conflict(f2)
                foreach p ∈ P(e) do
                    if f sichtbar von p then add_edge(p, f) zu G
            lösche Knoten {p} ∪ F_conflict(p) von G
```

Aufgabe 1

Worst Case Laufzeit

a) Zeige, dass die worst-case Laufzeit von ConvexHull in $\mathcal{O}(n^3)$ liegt.

b) Zeige, dass es Punktmengen gibt für die ConvexHull tatsächlich eine Laufzeit von $\Theta(n^3)$ benötigt, wenn eine "schlechte" Permutation erzeugt wird.

Aufgabe 1

Worst Case Laufzeit

a) Zeige, dass die worst-case Laufzeit von ConvexHull in $\mathcal{O}(n^3)$ liegt.

b) Zeige, dass es Punktmengen gibt für die ConvexHull tatsächlich eine Laufzeit von $\Theta(n^3)$ benötigt, wenn eine "schlechte" Permutation erzeugt wird.

Aufgabe 2

Alternativer-Algorithmus

Idee: 'rotiere' Ebene um bereits bekannte Kante der konvexen Hülle. Laufzeit $\mathcal{O}(n^2)$. Beschreibung?

Aufgabe 2

Alternativer-Algorithmus

Idee: 'rotiere' Ebene um bereits bekannte Kante der konvexen Hülle. Laufzeit $\mathcal{O}(n^2)$. Beschreibung?

Woher kennen wir diese Idee... ?

Aufgabe 2

Karlsruhe Institute of Technology

Aufgabe 3

Zufallszahlen

Zufallszahlengenerator kann in $O(1)$ nur ein zufälliges Bit erzeugen.

a) Beschreibe ein Verfahren um mit einem solchen Zufallszahlen-Generator eine zufällige Permutation von n Zahlen zu bestimmen.

b) Was is die Laufzeit von dem in a) beschriebenem Verfahren?

Aufgabe 3

Karlsruhe Institute of Technology

$\textsc{RandomInt}(k)$

$\mathcal{T} = k$ Zufallsbits

return Anzahl 1en in $\mathcal{T}$

Aufgabe 3

Karlsruhe Institute of Technology

$\textsc{RandomInt}(k)$

$d = \lceil \log_2 k \rceil$

result $= 0$

for $i \leftarrow 1$ to d **do**

  result = result + $2^i \cdot$ zufälliges Bit

return result

Aufgabe 3

Karlsruhe Institute of Technology

$\textsc{RandomInt}(k)$

$d = \lceil \log_2 k \rceil$

result $= 0$

for $i \leftarrow 1$ to d **do**

  result = result + $2^i \cdot$ zufälliges Bit

if result $> k$ **then**

  return $k-$result

return result

Aufgabe 3

Karlsruhe Institute of Technology

$\textsc{RandomInt}(k)$

$d = \lceil \log_2 k \rceil$

result $= 0$

for $i \leftarrow 1$ to d **do**

  result = result + $2^i \cdot$ zufälliges Bit

if result $> k$ **then**

  return $\textsc{RandomInt}$(k);

return result

Ablauf

Karlsruhe Institute of Technology

Übungsblatt

Neuer Algorithmus

Neuer Algorithmus

Karlsruhe Institute of Technology

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

Neuer Algorithmus

Karlsruhe Institute of Technology

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

if $|P| \leq 3$ **then**

  return $CH(P)$

Halbiere P in linke Teilmenge P_ℓ und rechte Teilmenge P_r

$CH(P_\ell) =$ DivideConquerConvexHull(P_ℓ)

$CH(P_r) =$ DivideConquerConvexHull(P_r)

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$

return $CH(P)$

Neuer Algorithmus

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

if $|P| \leq 3$ **then**

  return $CH(P)$

Halbiere P in linke Teilmenge P_ℓ und rechte Teilmenge P_r

$CH(P_\ell) =$ DivideConquerConvexHull(P_ℓ)

$CH(P_r) =$ DivideConquerConvexHull(P_r)

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$

return $CH(P)$

Neuer Algorithmus

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

if $|P| \leq 3$ **then**

  return $CH(P)$

Halbiere P in linke Teilmenge P_ℓ und rechte Teilmenge P_r

$CH(P_\ell) =$ DivideConquerConvexHull(P_ℓ)

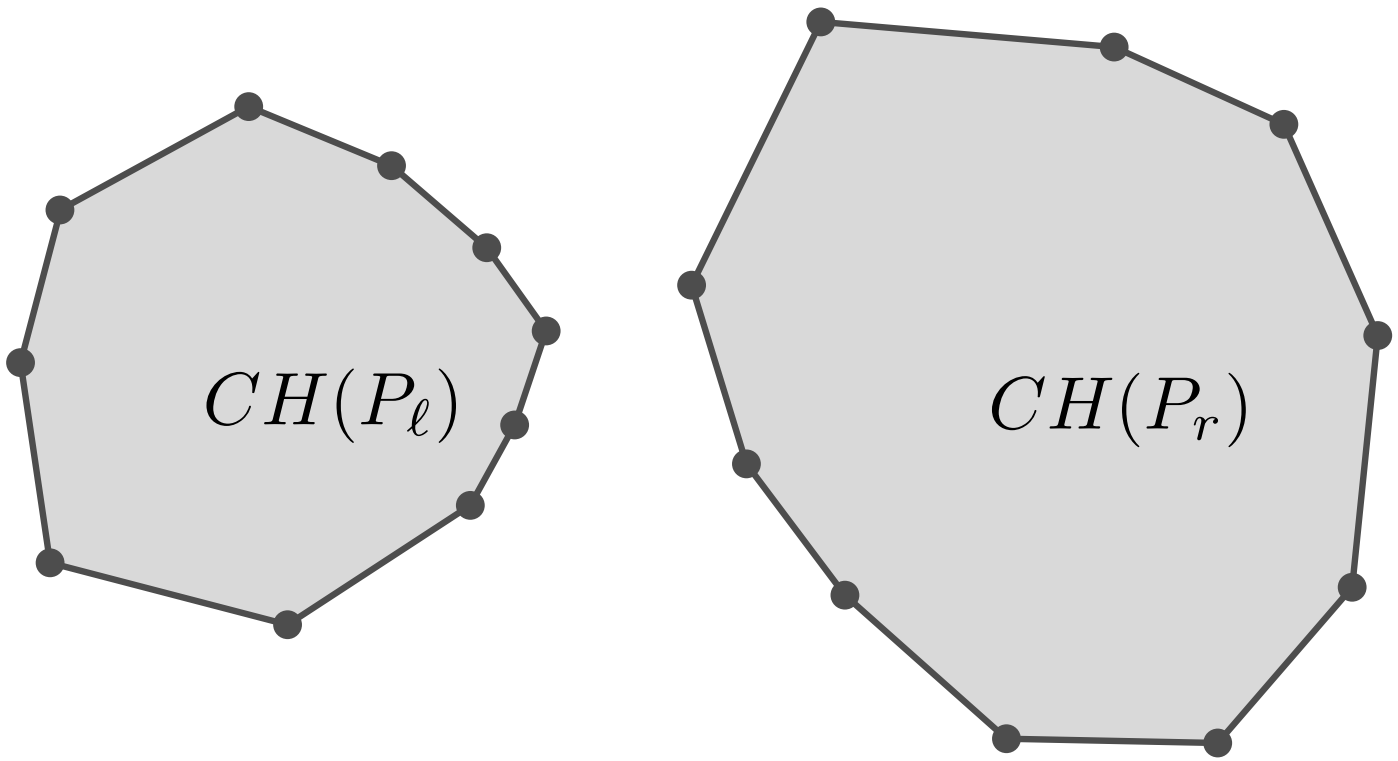
$CH(P_r) =$ DivideConquerConvexHull(P_r)

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$

return $CH(P)$

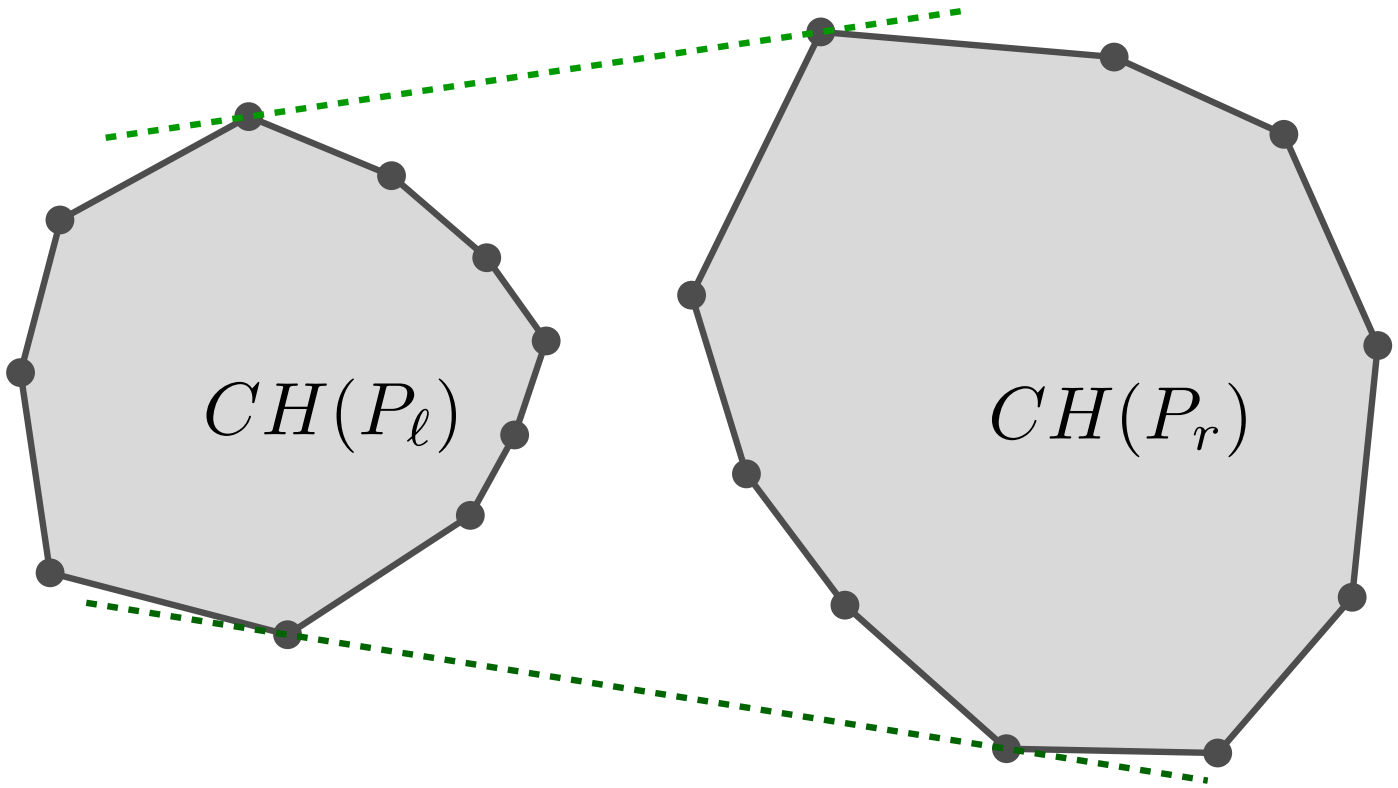
Vereinigung

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$



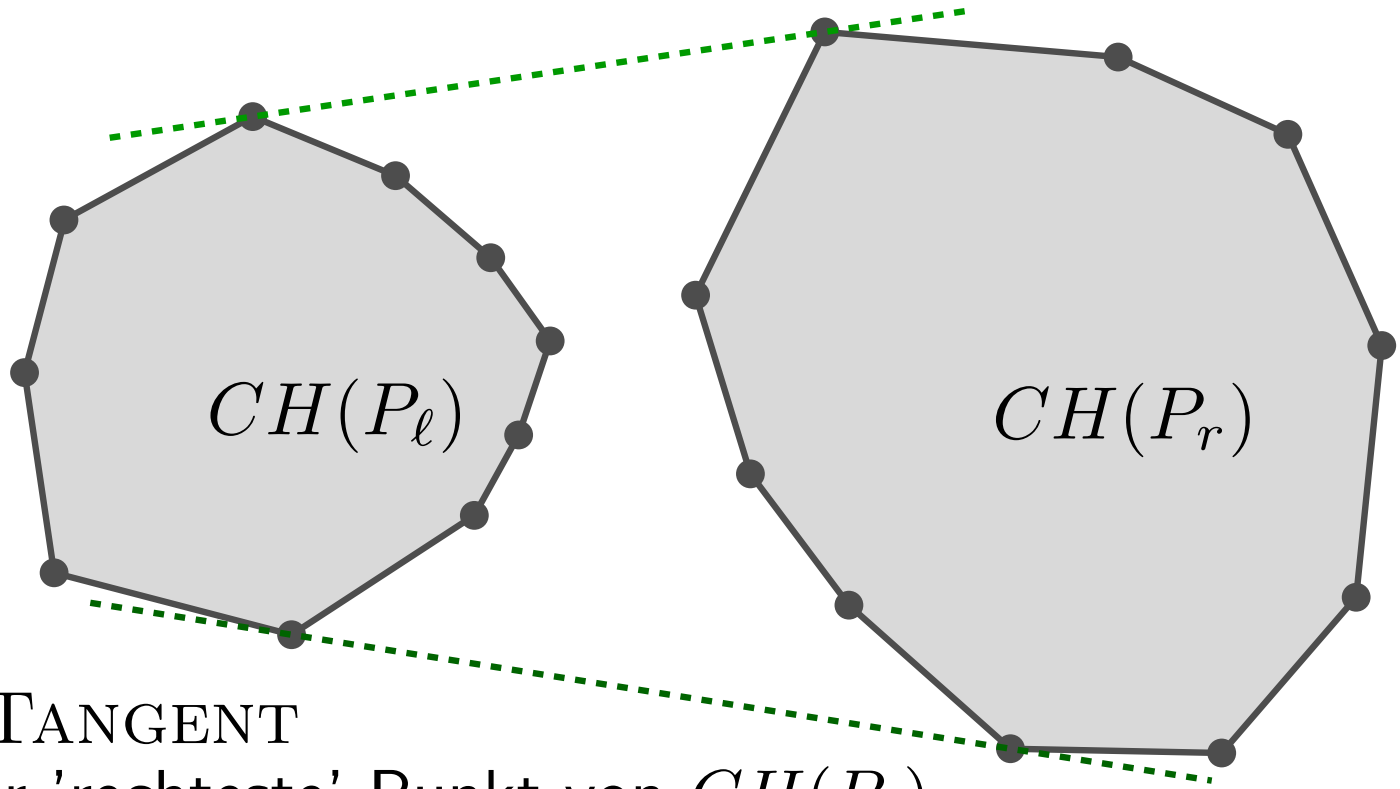
Vereinigung

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$



Vereinigung

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$



LOWER TANGENT

Sei a der 'rechteste' Punkt von $CH(P_\ell)$

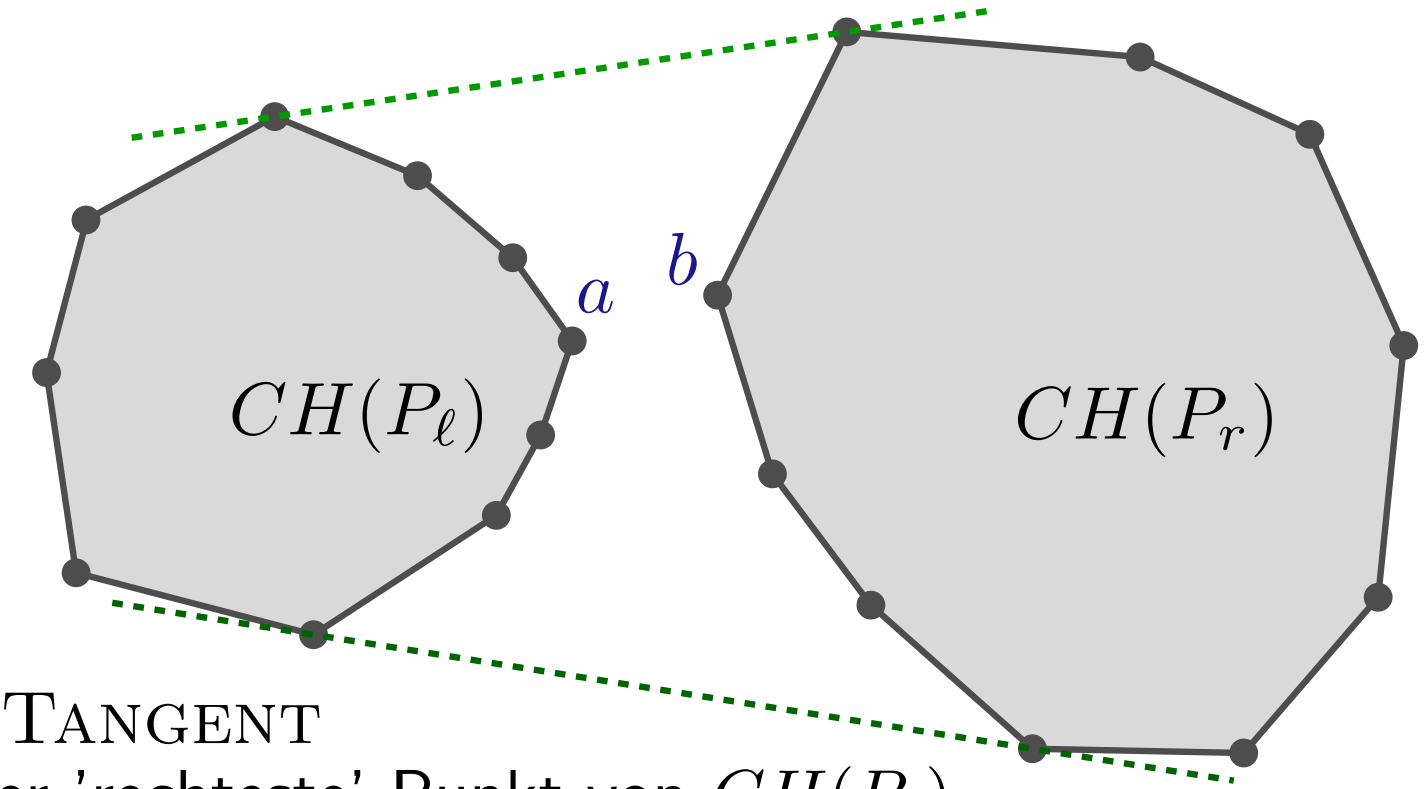
Sei b der 'linkeste' Punkt von $CH(P_r)$

while ab nicht untere Tangente **do**

- **while** ab nicht untere Tangente von $CH(P_\ell)$ **do** $a = a$.pred
- **while** ab nicht untere Tangente von $CH(P_r)$ **do** $b = b$.succ

Vereinigung

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$



LOWER TANGENT

Sei a der 'rechteste' Punkt von $CH(P_\ell)$

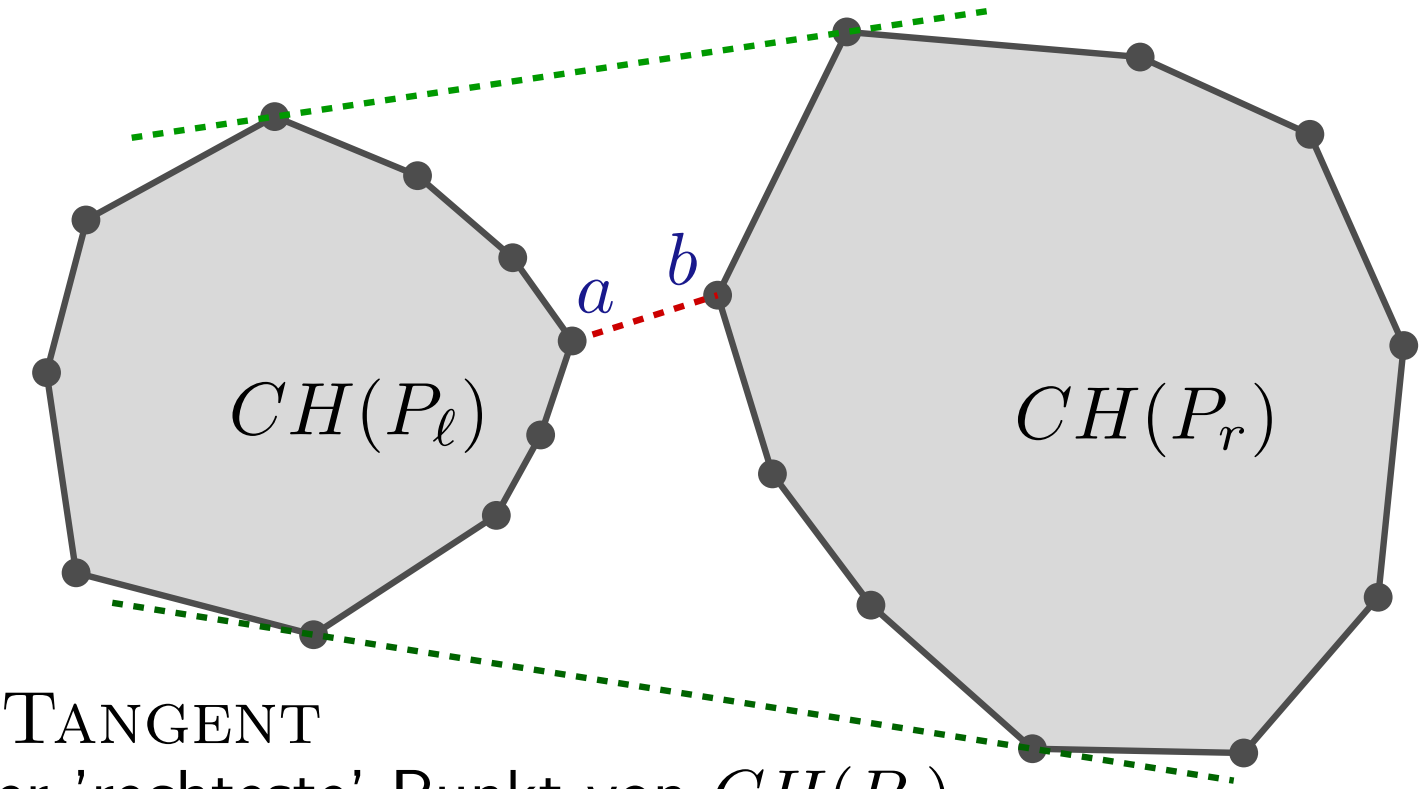
Sei b der 'linkeste' Punkt von $CH(P_r)$

while ab nicht untere Tangente **do**

- **while** ab nicht untere Tangente von $CH(P_\ell)$ **do** $a = a$.pred
- **while** ab nicht untere Tangente von $CH(P_r)$ **do** $b = b$.succ

Vereinigung

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$



LOWER TANGENT

Sei a der 'rechteste' Punkt von $CH(P_\ell)$

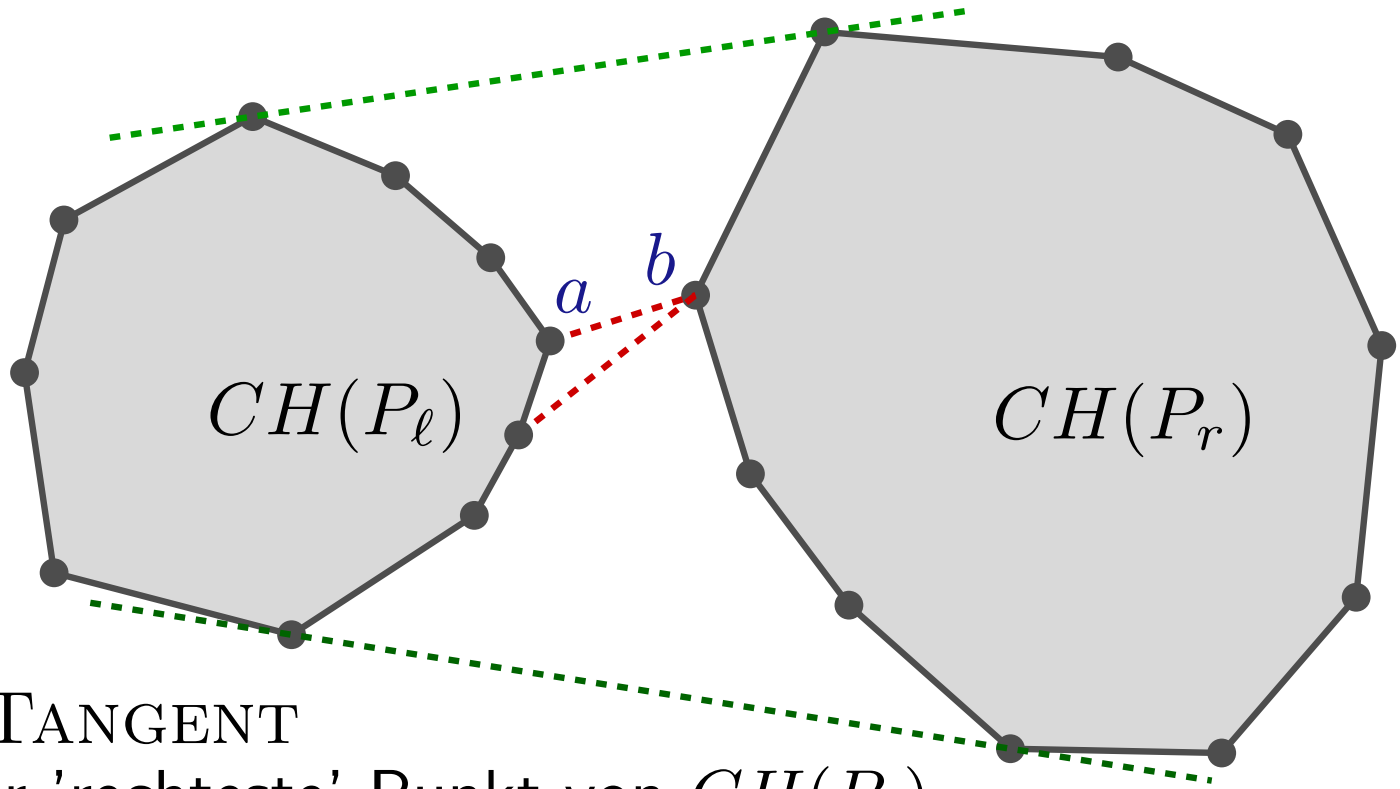
Sei b der 'linkeste' Punkt von $CH(P_r)$

while ab nicht untere Tangente **do**

- **while** ab nicht untere Tangente von $CH(P_\ell)$ **do** $a = a$.pred
- **while** ab nicht untere Tangente von $CH(P_r)$ **do** $b = b$.succ

Vereinigung

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$



LOWER TANGENT

Sei a der 'rechteste' Punkt von $CH(P_\ell)$

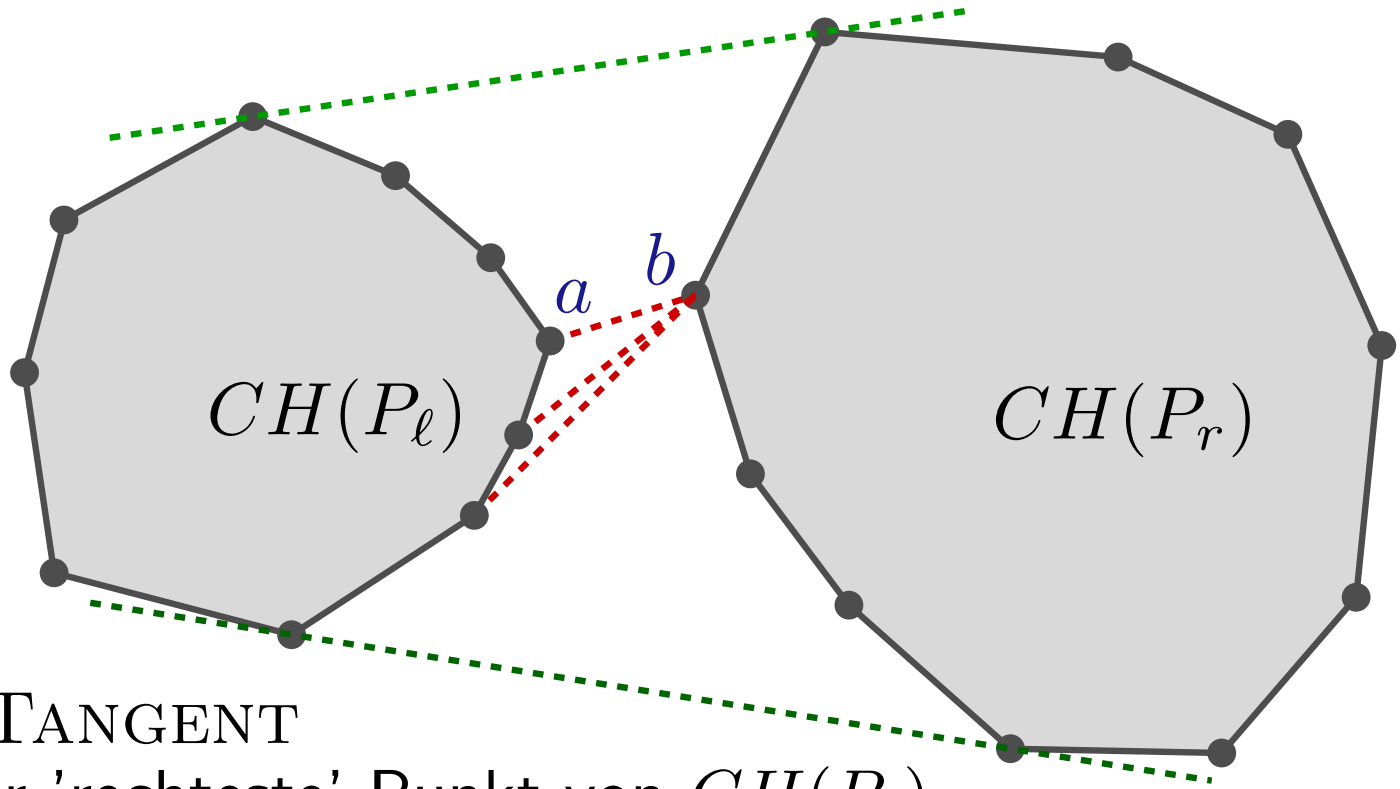
Sei b der 'linkeste' Punkt von $CH(P_r)$

while ab nicht untere Tangente **do**

- **while** ab nicht untere Tangente von $CH(P_\ell)$ **do** $a = a$.pred
- **while** ab nicht untere Tangente von $CH(P_r)$ **do** $b = b$.succ

Vereinigung

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$



LOWER TANGENT

Sei a der 'rechteste' Punkt von $CH(P_\ell)$

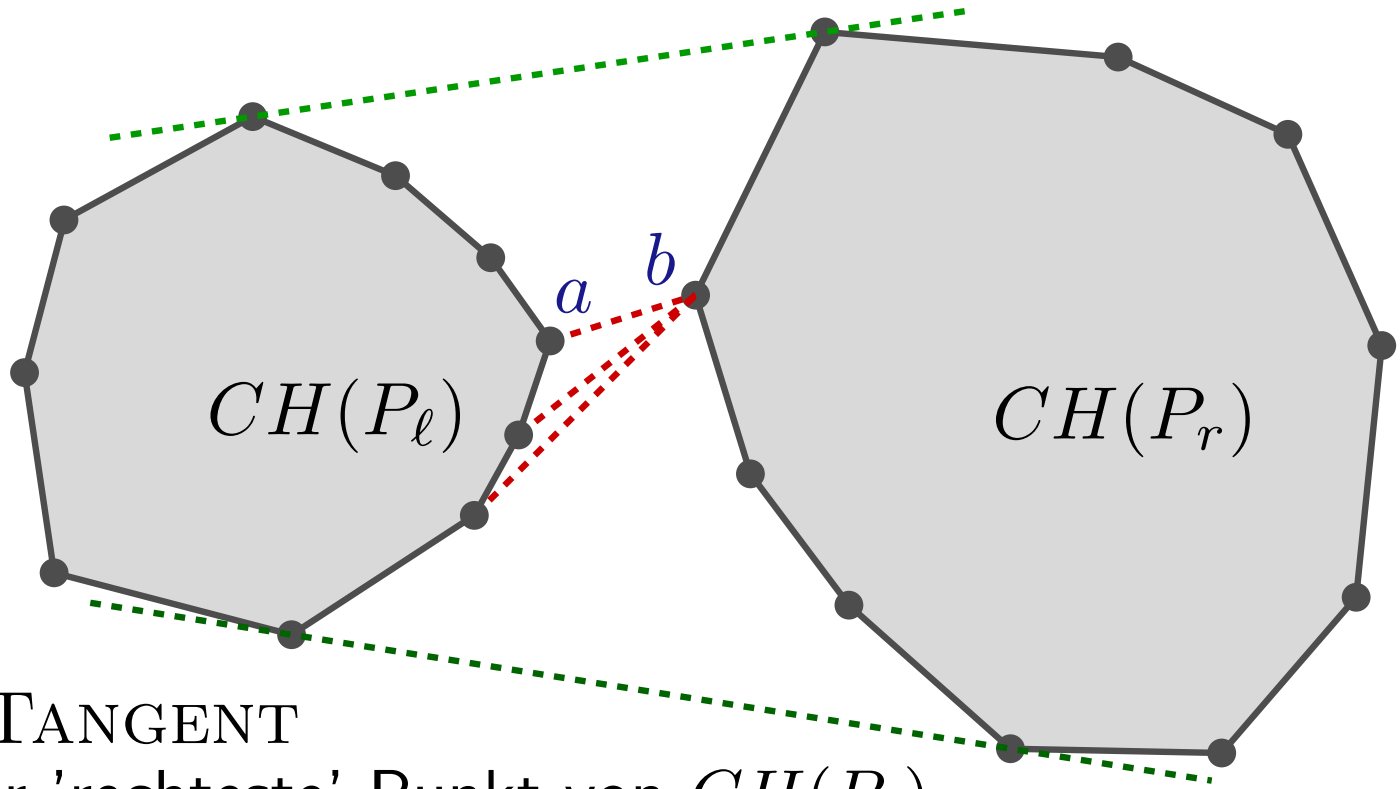
Sei b der 'linkeste' Punkt von $CH(P_r)$

while ab nicht untere Tangente **do**

- **while** ab nicht untere Tangente von $CH(P_\ell)$ **do** $a = a$.pred
- **while** ab nicht untere Tangente von $CH(P_r)$ **do** $b = b$.succ

Vereinigung

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$



LOWER TANGENT

Sei a der 'rechteste' Punkt von $CH(P_\ell)$

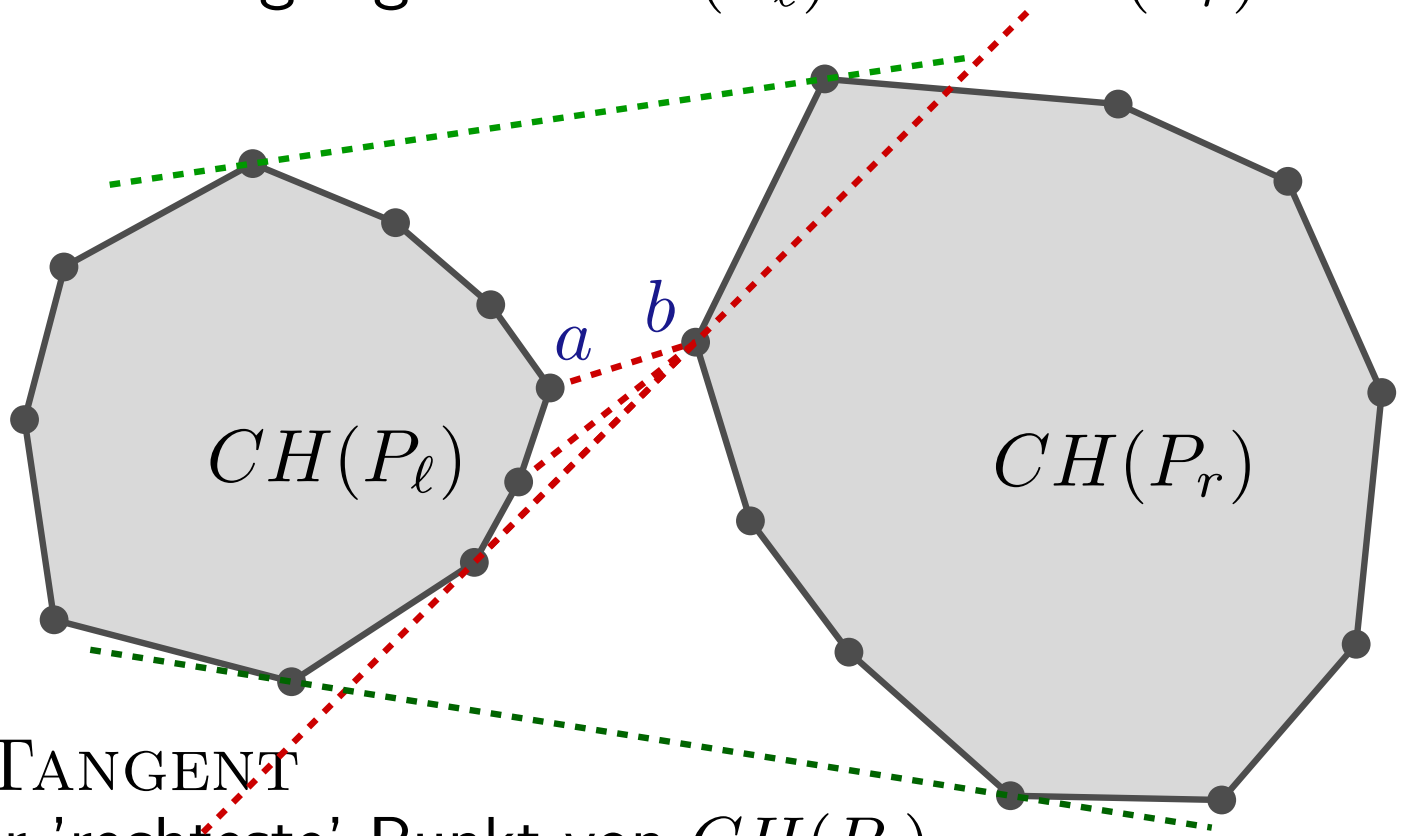
Sei b der 'linkeste' Punkt von $CH(P_r)$

while ab nicht untere Tangente **do**

- **while** ab nicht untere Tangente von $CH(P_\ell)$ **do** $a = a$.pred
- **while** ab nicht untere Tangente von $CH(P_r)$ **do** $b = b$.succ

Vereinigung

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$



LOWER TANGENT

Sei a der 'rechteste' Punkt von $CH(P_\ell)$

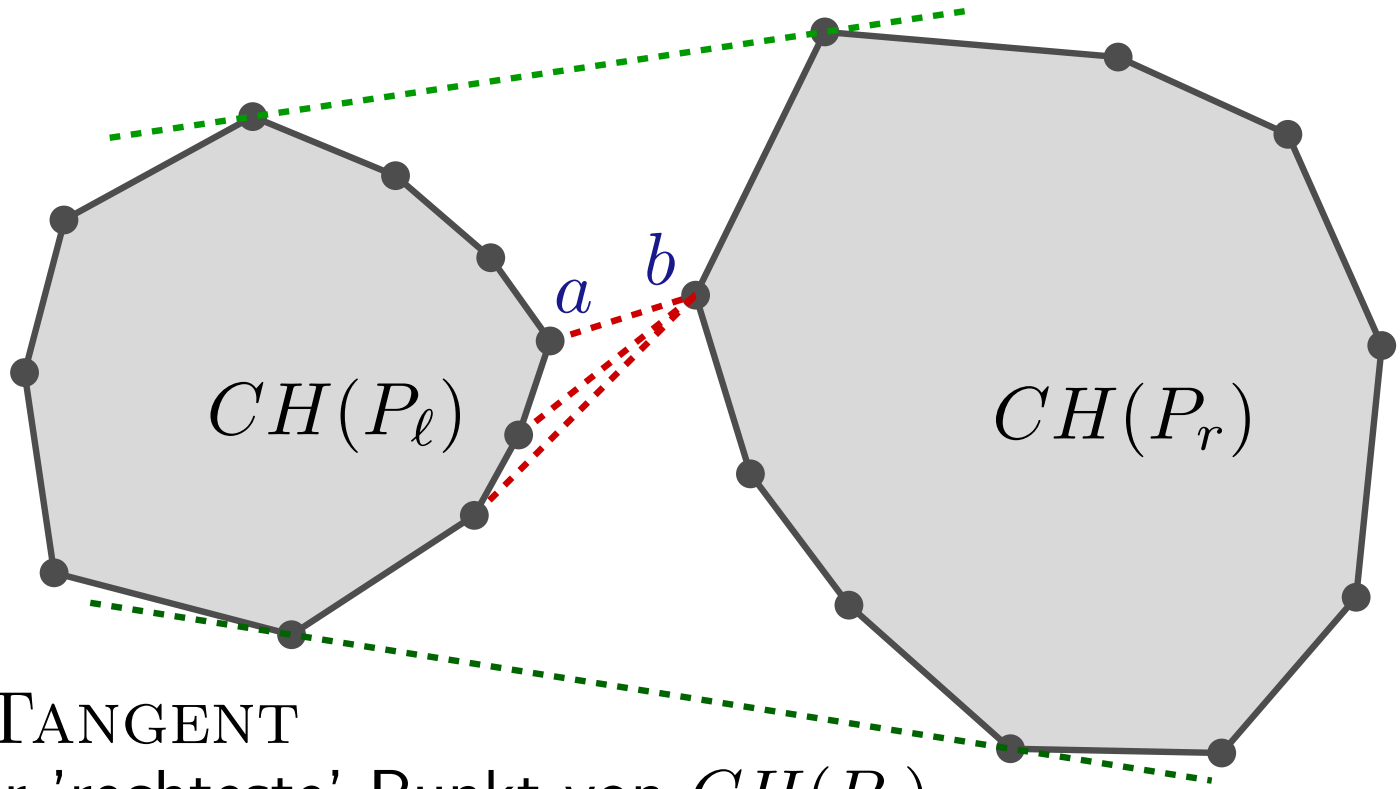
Sei b der 'linkeste' Punkt von $CH(P_r)$

while ab nicht untere Tangente **do**

- **while** ab nicht untere Tangente von $CH(P_\ell)$ **do** $a = a$.pred
- **while** ab nicht untere Tangente von $CH(P_r)$ **do** $b = b$.succ

Vereinigung

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$



LOWER TANGENT

Sei a der 'rechteste' Punkt von $CH(P_\ell)$

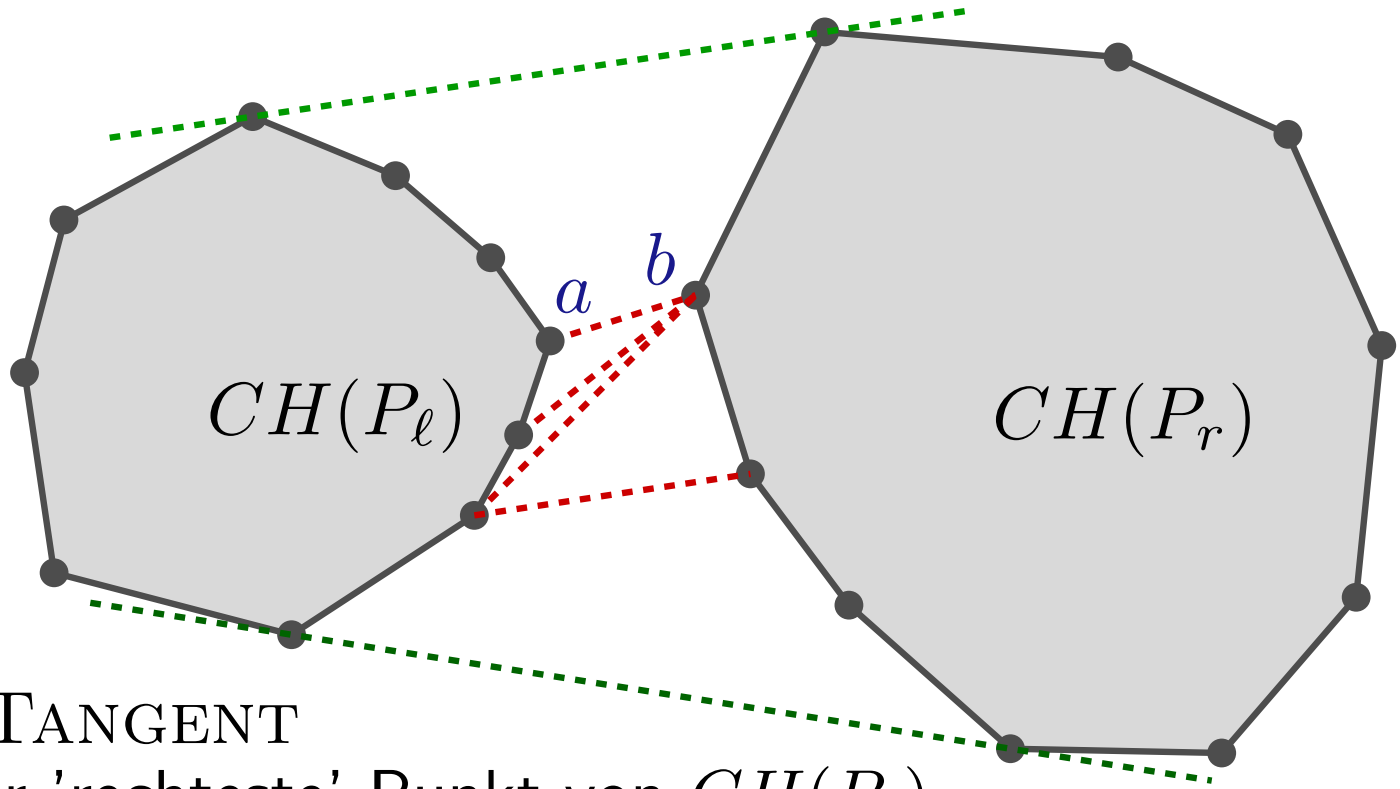
Sei b der 'linkeste' Punkt von $CH(P_r)$

while ab nicht untere Tangente **do**

- **while** ab nicht untere Tangente von $CH(P_\ell)$ **do** $a = a$.pred
- **while** ab nicht untere Tangente von $CH(P_r)$ **do** $b = b$.succ

Vereinigung

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$



LOWER TANGENT

Sei a der 'rechteste' Punkt von $CH(P_\ell)$

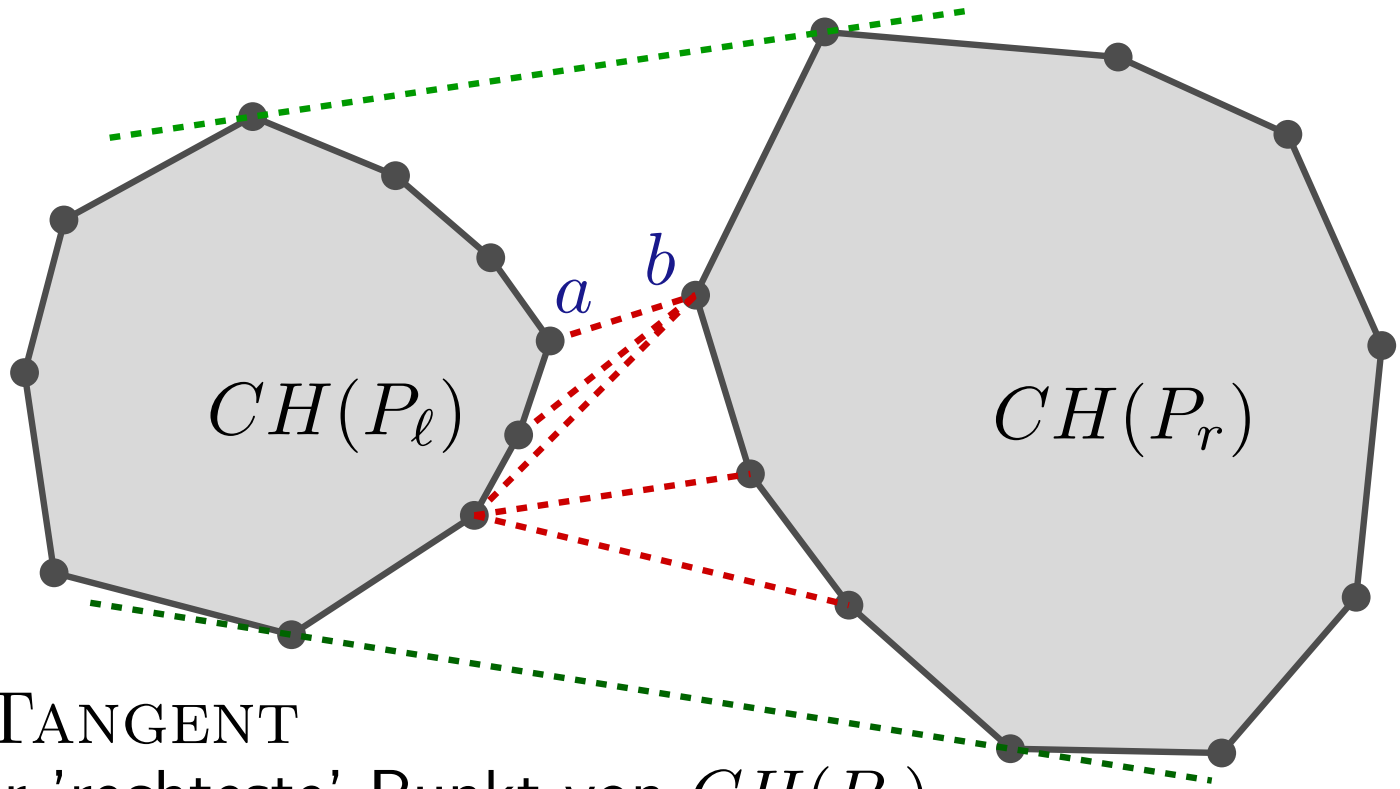
Sei b der 'linkeste' Punkt von $CH(P_r)$

while ab nicht untere Tangente **do**

- **while** ab nicht untere Tangente von $CH(P_\ell)$ **do** $a = a$.pred
- **while** ab nicht untere Tangente von $CH(P_r)$ **do** $b = b$.succ

Vereinigung

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$



LOWER TANGENT

Sei a der 'rechteste' Punkt von $CH(P_\ell)$

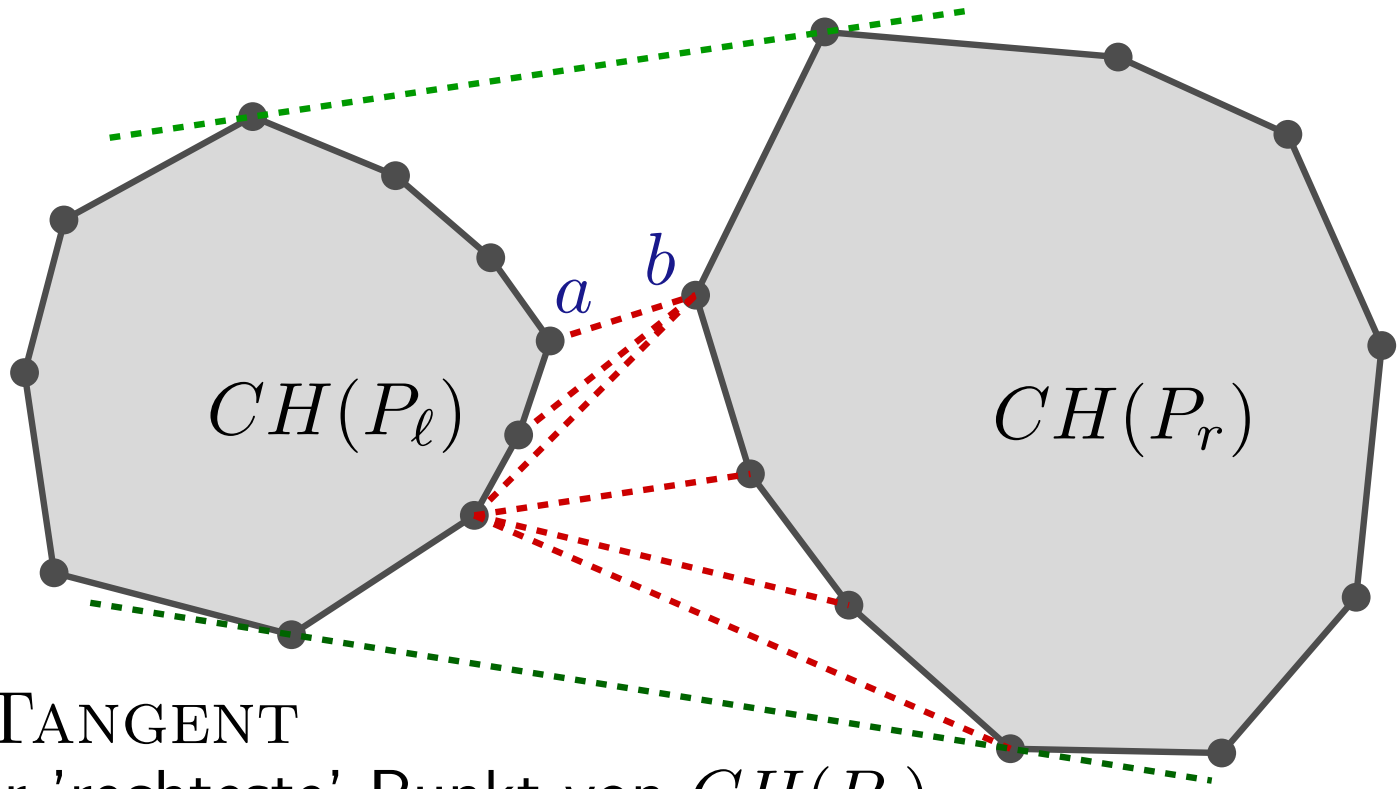
Sei b der 'linkeste' Punkt von $CH(P_r)$

while ab nicht untere Tangente **do**

- **while** ab nicht untere Tangente von $CH(P_\ell)$ **do** $a = a$.pred
- **while** ab nicht untere Tangente von $CH(P_r)$ **do** $b = b$.succ

Vereinigung

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$



LOWER TANGENT

Sei a der 'rechteste' Punkt von $CH(P_\ell)$

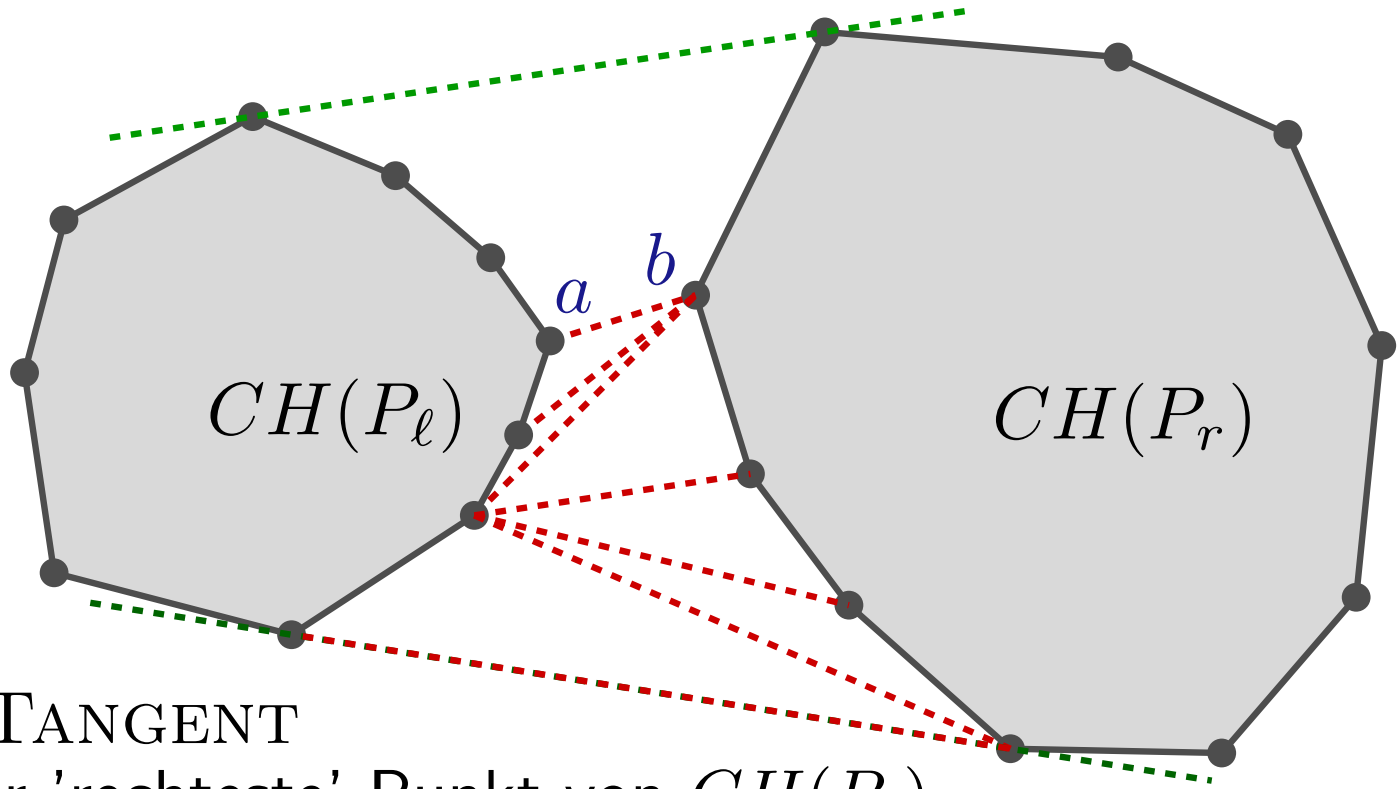
Sei b der 'linkeste' Punkt von $CH(P_r)$

while ab nicht untere Tangente **do**

- **while** ab nicht untere Tangente von $CH(P_\ell)$ **do** $a = a$.pred
- **while** ab nicht untere Tangente von $CH(P_r)$ **do** $b = b$.succ

Vereinigung

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$



LOWER TANGENT

Sei a der 'rechteste' Punkt von $CH(P_\ell)$

Sei b der 'linkeste' Punkt von $CH(P_r)$

while ab nicht untere Tangente **do**

  while ab nicht untere Tangente von $CH(P_\ell)$ **do** $a = a$.pred

  while ab nicht untere Tangente von $CH(P_r)$ **do** $b = b$.succ

Vereinigung

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$

LOWER TANGENT

Sei a der 'rechteste' Punkt von $CH(P_\ell)$

Sei b der 'linkeste' Punkt von $CH(P_r)$

while ab nicht untere Tangente **do**

  while ab nicht untere Tangente von $CH(P_\ell)$ **do** $a = a$.pred

  while ab nicht untere Tangente von $CH(P_r)$ **do** $b = b$.succ

Laufzeit?

Karlsruhe Institute of Technology

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

if $|P| \leq 3$ **then**

  return $CH(P)$

Halbiere P in linke Teilmenge P_ℓ und rechte Teilmenge P_r

$CH(P_\ell) =$ DivideConquerConvexHull(P_ℓ)

$CH(P_r) =$ DivideConquerConvexHull(P_r)

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$

return $CH(P)$

Laufzeit?

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

if $|P| \leq 3$ **then**

  return $CH(P)$

Halbiere P in linke Teilmenge P_ℓ und rechte Teilmenge P_r

$CH(P_\ell) =$ DivideConquerConvexHull(P_ℓ)

$CH(P_r) =$ DivideConquerConvexHull(P_r)

$CH(P) =$ Vereinigung von $CH(P_\ell)$ und $CH(P_r)$

return $CH(P)$

$$T(n) = \begin{cases} \mathcal{O}(1) & , |P| \leq 3 \\ 2T(n/2) + \mathcal{O}(n) & , \text{sonst} \end{cases}$$

Das war's!

Karlsruhe Institute of Technology

Bis nächste Woche!

Nächster Termin:

Donnertag, 26.05, 10:15 Uhr

Raum 131, Gebäude 50.34