Karlsruhe Institute of Technology

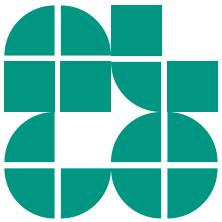
Vorlesung Algorithmische Geometrie

Voronoi-Diagramme & Delaunay-Triangulierungen

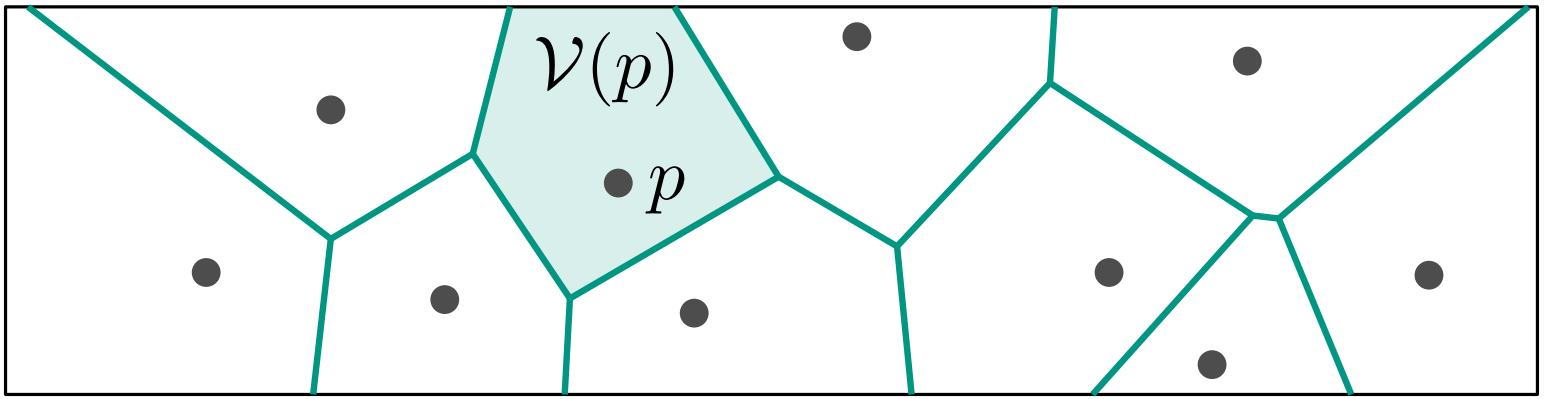
LEHRSTUHL FÜR ALGORITHMIK I · INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Martin Nöllenburg

07.06.2011

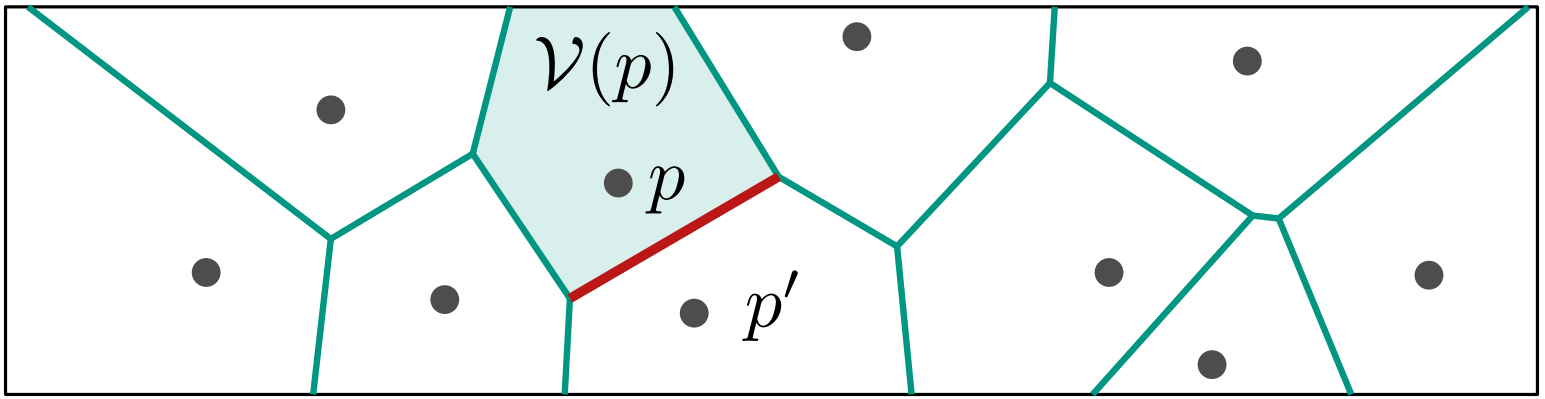


Erinnerung: Voronoi-Diagramm



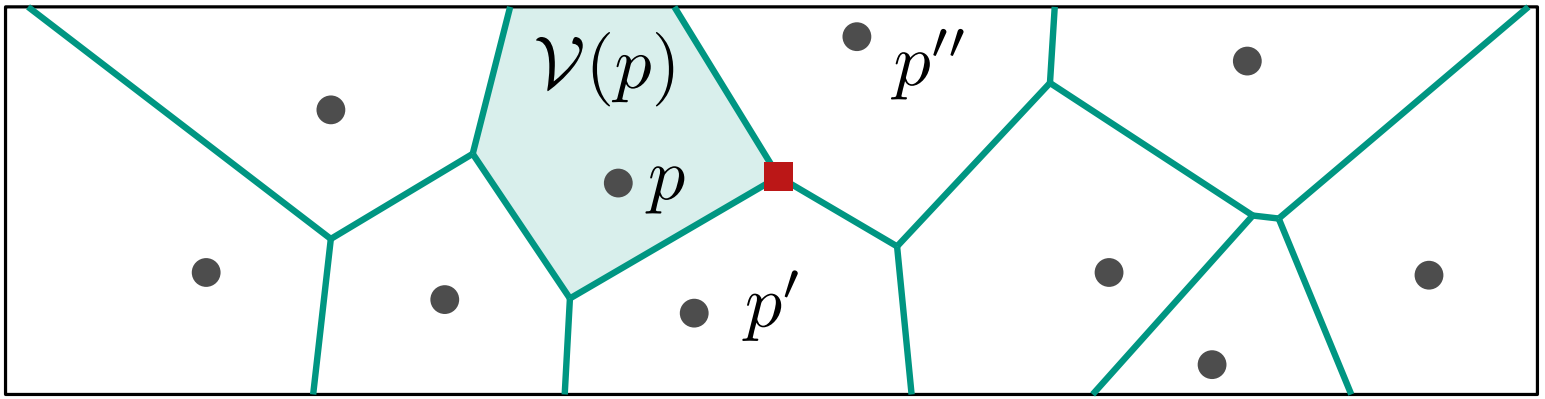
- $\mathcal{V}(p) = \{x \in \mathbb{R}^2 : |px| < |qx| \, \forall q \in P \setminus \{p\}\}$

Erinnerung: Voronoi-Diagramm



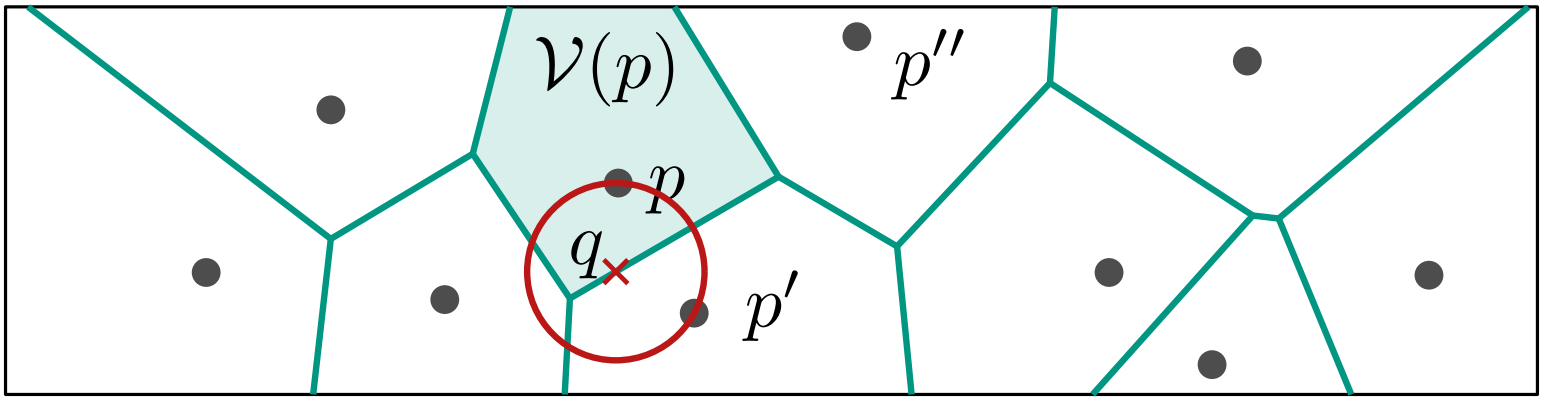
- $\mathcal{V}(p) = \{x \in \mathbb{R}^2 : |px| < |qx| \, \forall q \in P \setminus \{p\}\}$
- $\mathcal{V}(\{p, p'\}) = \{x \in \mathbb{R}^2 : |px| = |p'x| < |qx| \, \forall q \in P \setminus \{p, p'\}\}$

Erinnerung: Voronoi-Diagramm



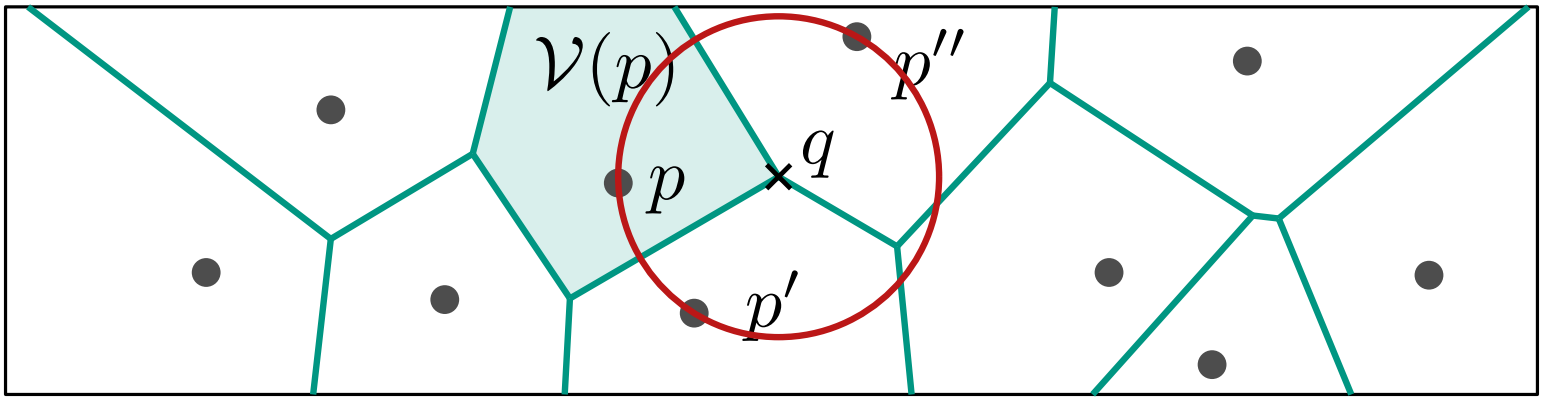
- $\mathcal{V}(p) = \{x \in \mathbb{R}^2 : |px| < |qx| \, \forall q \in P \setminus \{p\}\}$
- $\mathcal{V}(\{p, p'\}) = \{x \in \mathbb{R}^2 : |px| = |p'x| < |qx| \, \forall q \in P \setminus \{p, p'\}\}$
- $\mathcal{V}(\{p, p', p''\}) = \partial\mathcal{V}(p) \cap \partial\mathcal{V}(p') \cap \partial\mathcal{V}(p'')$

Erinnerung: Voronoi-Diagramm



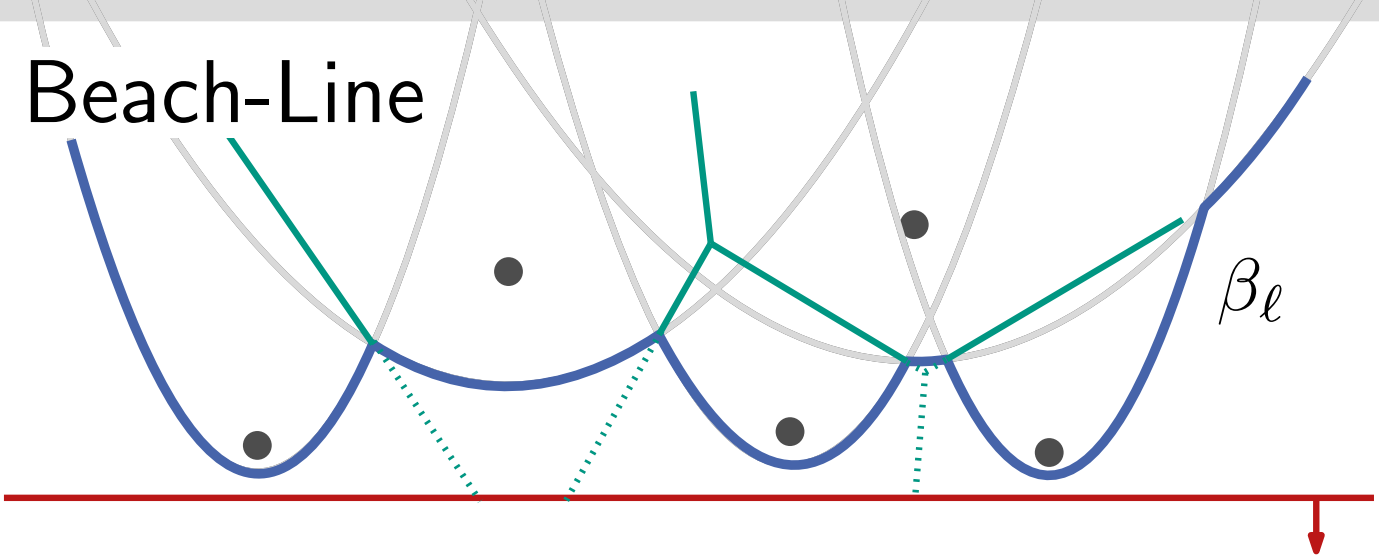
- $\mathcal{V}(p) = \{x \in \mathbb{R}^2 : |px| < |qx| \, \forall q \in P \setminus \{p\}\}$
- $\mathcal{V}(\{p, p'\}) = \{x \in \mathbb{R}^2 : |px| = |p'x| < |qx| \, \forall q \in P \setminus \{p, p'\}\}$
- $\mathcal{V}(\{p, p', p''\}) = \partial\mathcal{V}(p) \cap \partial\mathcal{V}(p') \cap \partial\mathcal{V}(p'')$
- $q \in \mathcal{V}(\{p, p'\}) \Leftrightarrow C_P(q) \cap P = \{p, p'\}$

Erinnerung: Voronoi-Diagramm



- $\mathcal{V}(p) = \{x \in \mathbb{R}^2 : |px| < |qx| \, \forall q \in P \setminus \{p\}\}$
- $\mathcal{V}(\{p, p'\}) = \{x \in \mathbb{R}^2 : |px| = |p'x| < |qx| \, \forall q \in P \setminus \{p, p'\}\}$
- $\mathcal{V}(\{p, p', p''\}) = \partial\mathcal{V}(p) \cap \partial\mathcal{V}(p') \cap \partial\mathcal{V}(p'')$
- $q \in \mathcal{V}(\{p, p'\}) \Leftrightarrow C_P(q) \cap P = \{p, p'\}$
- $q = \mathcal{V}(\{p, p', p''\}) \Leftrightarrow C_P(q) \cap P \supseteq \{p, p', p''\}$

Wdh: Beach-Line

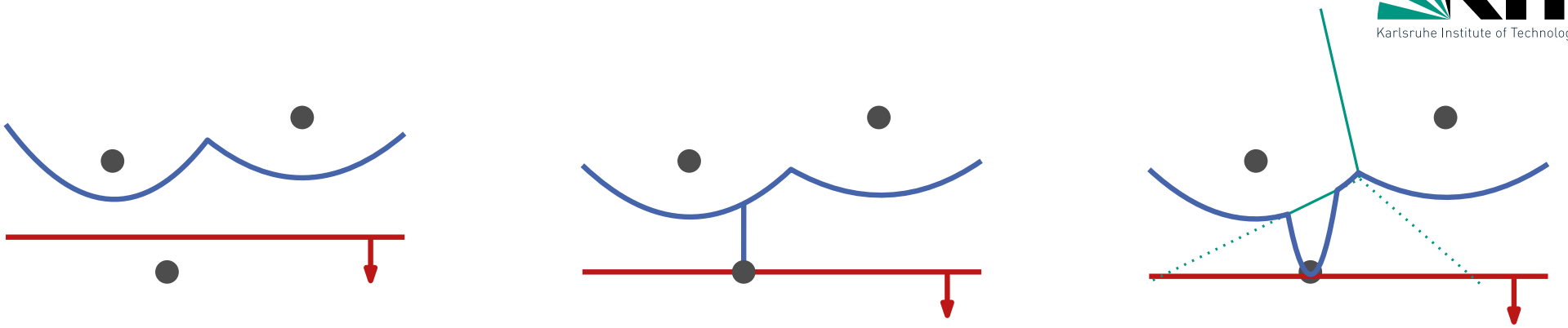


Definition: Die **Beach-Line** β_ℓ ist die untere Kontur der Parabeln f_p^ℓ für die bereits besuchten Punkte.

Beob.:
- Beach-Line ist x-monoton
- Schnittpunkte der Beach-Line „zeichnen“ die Voronoi- Kanten

Ziel: speichere (implizit) aktuelle Kontur β_ℓ statt $\mathrm{Vor}(P) \cap \ell$

Wdh: Punkt-Events

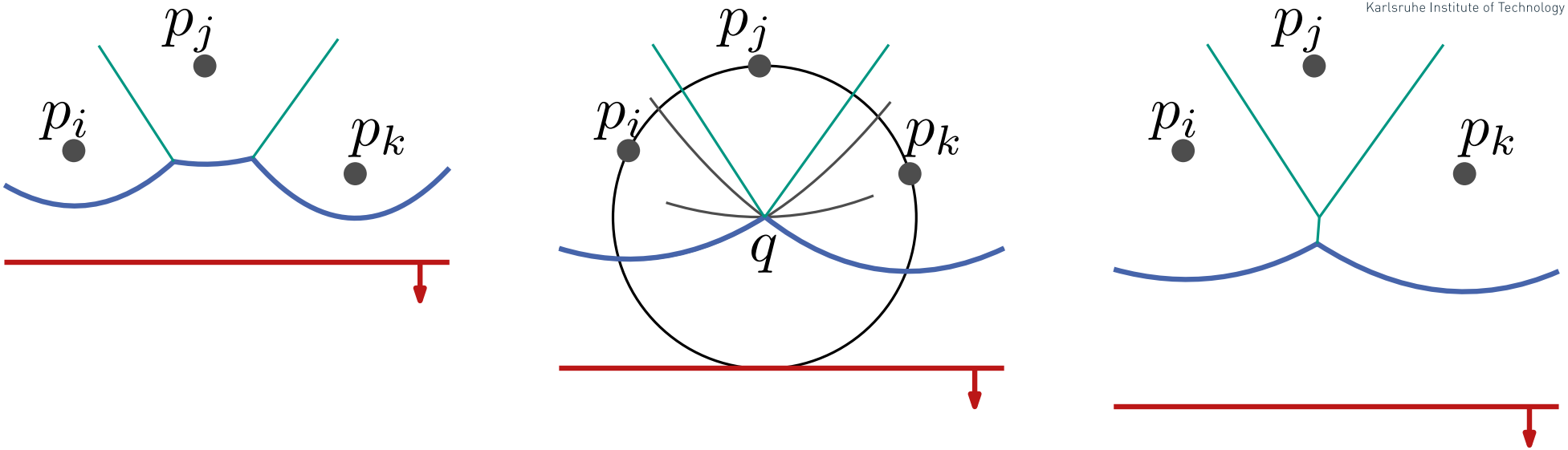


- trifft ℓ auf einen Punkt, kommt neue Parabel zu β_ℓ hinzu
- die beiden Schnittpunkte erzeugen neue Teilkante

Lemma: Neue Bögen auf β entstehen nur durch Punkt-Events.

Korollar: β besteht aus maximal $2n - 1$ Parabelbögen

Wdh: Kreis-Events



- verschwindet der Bogen für p_j so laufen $f^{\ell}_{p_i}, f^{\ell}_{p_j}, f^{\ell}_{p_k}$ durch einen gemeinsamen Punkt q
- Kreis $C_P(q)$ geht durch p_i, p_j, p_k und berührt ℓ
  $\Rightarrow q$ ist Voronoi-Knoten

Def.: Der unterste Punkt des Kreises durch drei Punkte mit konsekutiven Bögen auf β definiert ein **Kreis-Event**.

Lemma: Bögen von β werden nur durch Kreis-Events entfernt.

Lemma: Für jeden Voronoi-Knoten gibt es ein Kreis-Event.

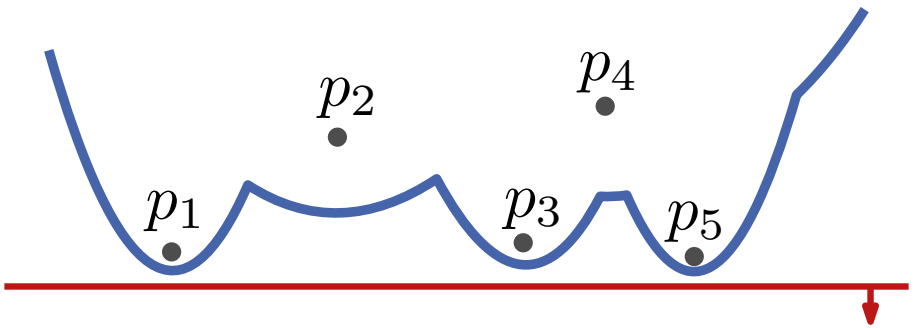
Datenstrukturen

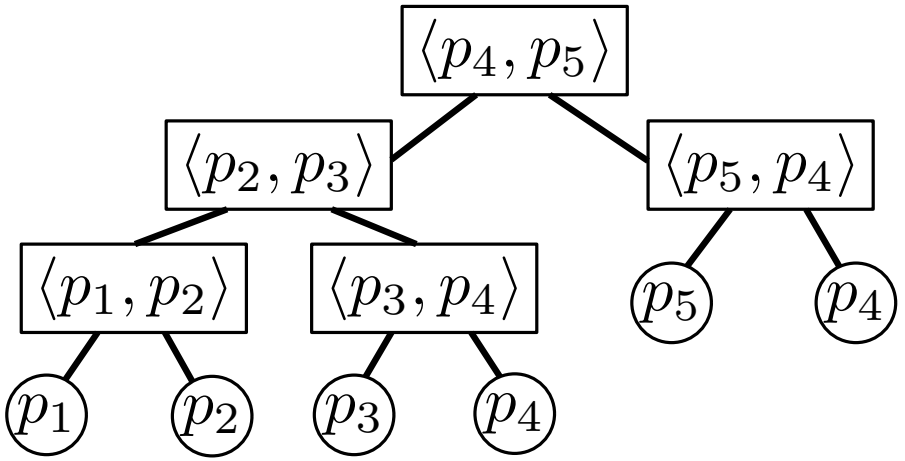
- doppelt-verkettete Kantenliste (DCEL) $\mathcal{D}$ für $\mathrm{Vor}(P)$
  Achtung: am Schluss bounding box wg. Halbgeraden einfügen

Datenstrukturen

- doppelt-verkettete Kantenliste (DCEL) $\mathcal{D}$ für $\mathrm{Vor}(P)$
  Achtung: am Schluss bounding box wg. Halbgeraden einfügen

- balancierter binärer Suchbaum $\mathcal{T}$ für implizite Beach-Line
  – Blätter entsprechen Parabelbögen von links nach rechts
  – innerer Knoten $\langle p_i, p_j \rangle$ entspricht Schnittpunkt von p_i und p_j
  – Pointer von inneren Knoten auf zugeh. Kanten in $\mathcal{D}$

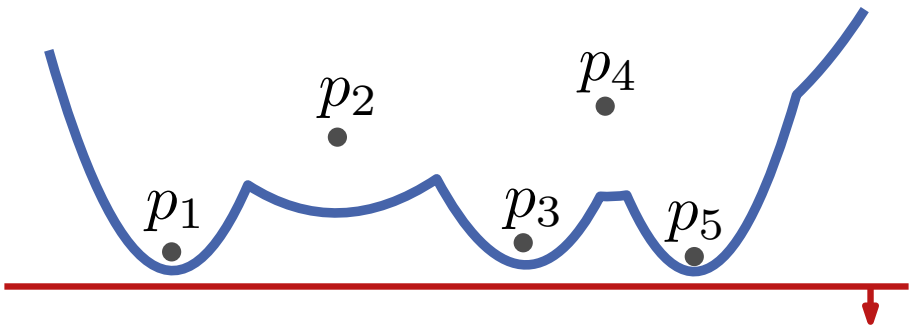


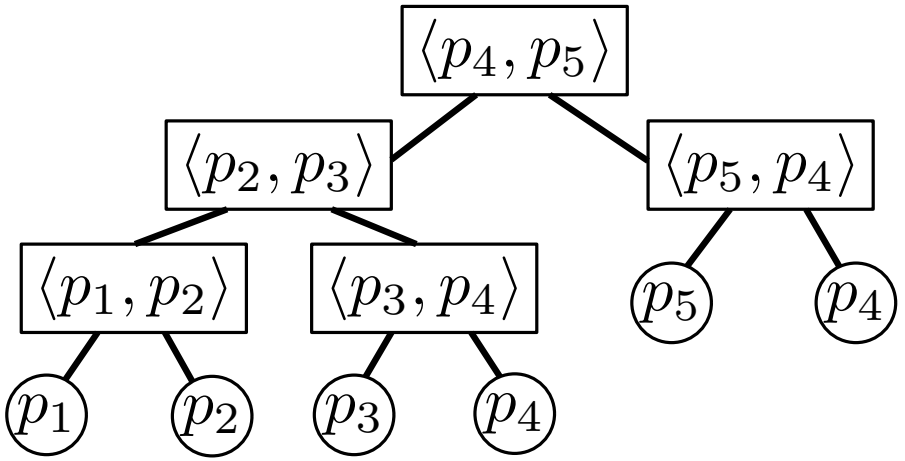


Datenstrukturen

- doppelt-verkettete Kantenliste (DCEL) $\mathcal{D}$ für $\mathrm{Vor}(P)$
  Achtung: am Schluss bounding box wg. Halbgeraden einfügen

- balancierter binärer Suchbaum $\mathcal{T}$ für implizite Beach-Line
  – Blätter entsprechen Parabelbögen von links nach rechts
  – innerer Knoten $\langle p_i, p_j \rangle$ entspricht Schnittpunkt von p_i und p_j
  – Pointer von inneren Knoten auf zugeh. Kanten in $\mathcal{D}$





- Priority Queue $\mathcal{Q}$ für die Punkt- und Kreis-Events
  – Pointer von Kreis-Events auf zugeh. Blätter in $\mathcal{T}$ und umgekehrt

Fortune's Sweep Algorithmus

VoronoiDiagram($P \subset \mathbb{R}^2$)
 $\mathcal{Q} \leftarrow$ new PriorityQueue(P) // Punkt-Events sortiert nach y
 $\mathcal{T} \leftarrow$ new BalancedBinarySearchTree() // Beach-Line
 $\mathcal{D} \leftarrow$ new DCEL() // DS für Vor(P)
 while not $\mathcal{Q}$.empty() **do**
  $p \leftarrow \mathcal{Q}$.ExtractMax()
  if p Punkt-Event **then**
   HandlePointEvent(p)
  else
   $\alpha \leftarrow$ Bogen von β, der entfernt werden soll
   HandleCircleEvent(α)
 behandle innere Restknoten von $\mathcal{T}$ (Halbgeraden von Vor(P))
 return $\mathcal{D}$

Fortune's Sweep Algorithmus

Steven Fortune, 1986

Karlsruhe Institute of Technology

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

$\mathcal{Q} \leftarrow$ new PriorityQueue(P) // Punkt-Events sortiert nach y

$\mathcal{T} \leftarrow$ new BalancedBinarySearchTree() // Beach-Line

$\mathcal{D} \leftarrow$ new DCEL() // DS für Vor(P)

while not $\mathcal{Q}$.empty() **do**

  $p \leftarrow \mathcal{Q}$.ExtractMax()

  if p Punkt-Event **then**

    HandlePointEvent(p)

  else

    $\alpha \leftarrow$ Bogen von β, der entfernt werden soll

    HandleCircleEvent(α)

behandle innere Restknoten von $\mathcal{T}$ (Halbgeraden von Vor(P))

return $\mathcal{D}$

Fortune's Sweep Algorithmus

Steven Fortune, 1986

Karlsruhe Institute of Technology

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

$\mathcal{Q} \leftarrow$ new PriorityQueue(P) // Punkt-Events sortiert nach y

$\mathcal{T} \leftarrow$ new BalancedBinarySearchTree() // Beach-Line

$\mathcal{D} \leftarrow$ new DCEL() // DS für Vor(P)

while not $\mathcal{Q}$.empty() **do**

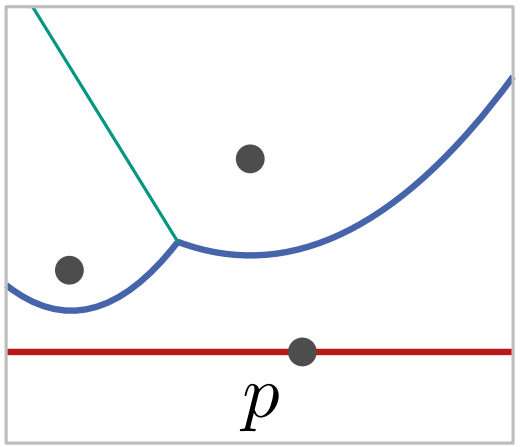
- $p \leftarrow \mathcal{Q}$.ExtractMax()
- **if** p Punkt-Event **then**
  - HandlePointEvent(p)
- **else**
  - $\alpha \leftarrow$ Bogen von β, der entfernt werden soll
  - HandleCircleEvent(α)

behandle innere Restknoten von $\mathcal{T}$ (Halbgeraden von Vor(P))

return $\mathcal{D}$

Punkt-Events behandeln

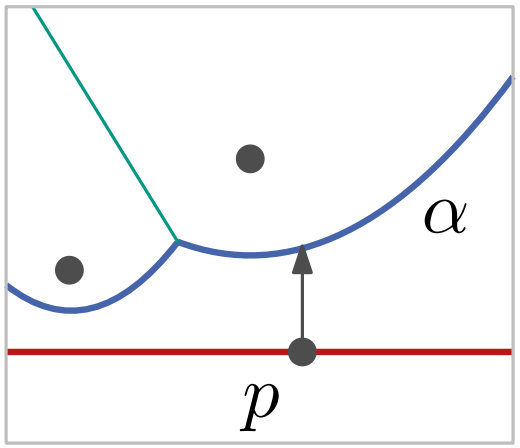
HandlePointEvent(Punkt p)



Punkt-Events behandeln

HandlePointEvent(Punkt p)

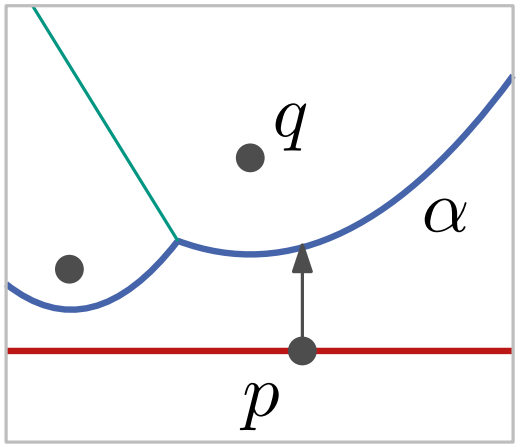
- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.



Punkt-Events behandeln

HandlePointEvent(Punkt p)

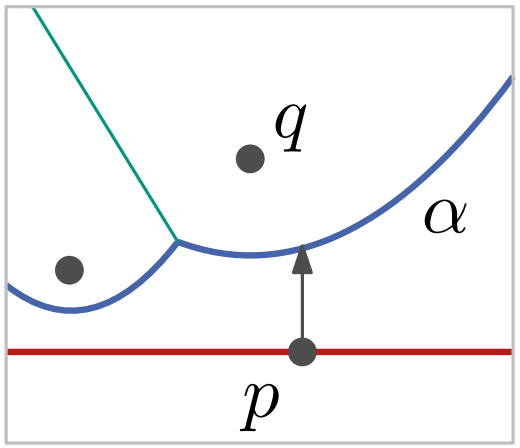
- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.
- Teile α in α_0 und α_2.
  Sei α_1 neuer Bogen für p.



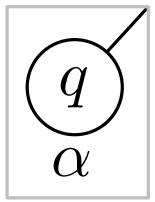
Punkt-Events behandeln

HandlePointEvent(Punkt p)

- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.
- Teile α in α_0 und α_2.
  Sei α_1 neuer Bogen für p.



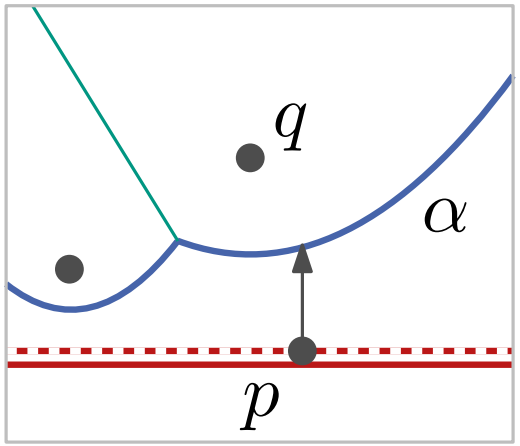
In $\mathcal{T}$:



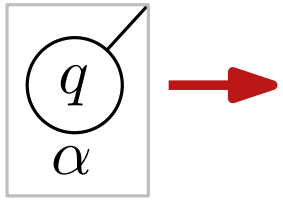
Punkt-Events behandeln

HandlePointEvent(Punkt p)

- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.
- Teile α in α_0 und α_2.
  Sei α_1 neuer Bogen für p.



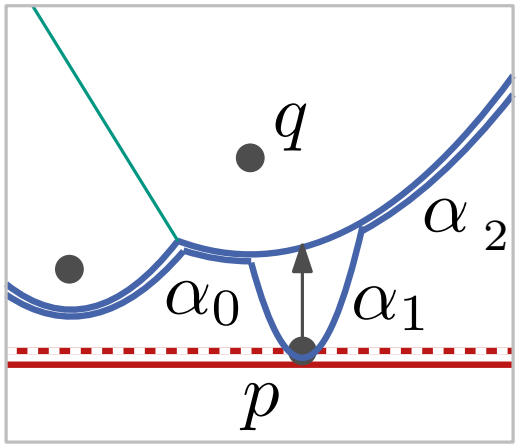
In $\mathcal{T}$:



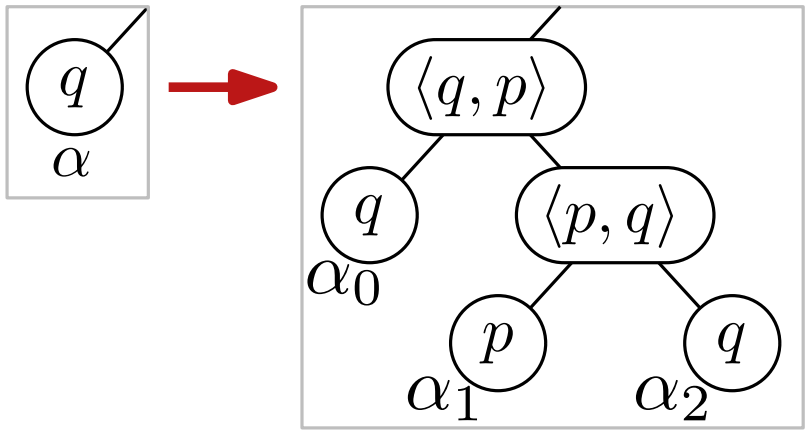
Punkt-Events behandeln

HandlePointEvent(Punkt p)

- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.
- Teile α in α_0 und α_2.
  Sei α_1 neuer Bogen für p.



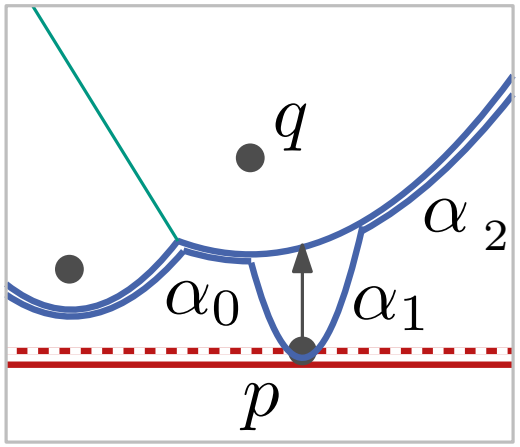
In $\mathcal{T}$:



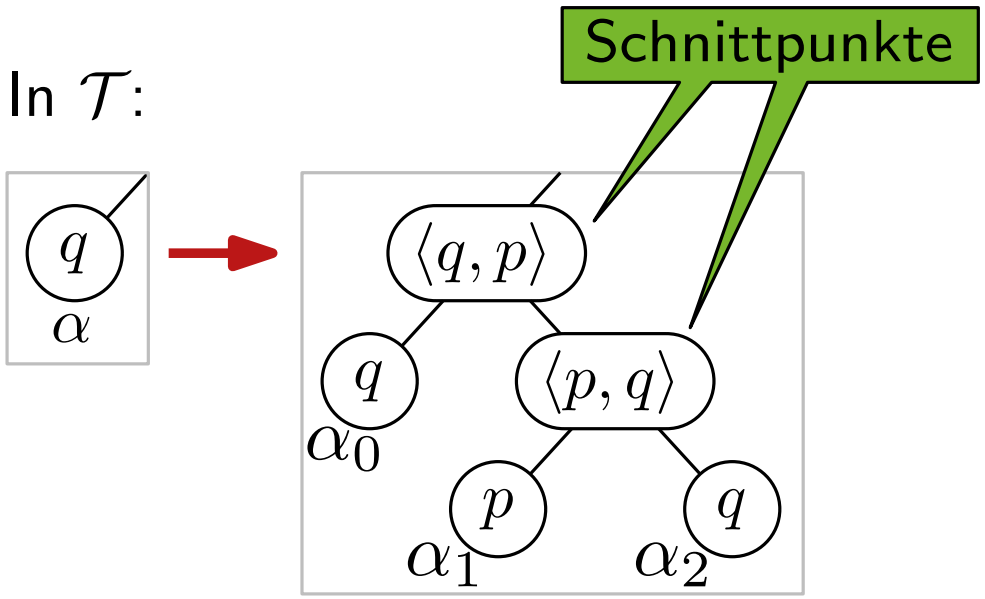
Punkt-Events behandeln

HandlePointEvent(Punkt p)

- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.
- Teile α in α_0 und α_2.
  Sei α_1 neuer Bogen für p.



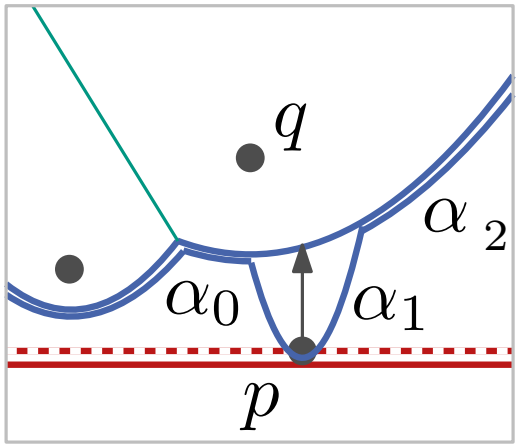
In $\mathcal{T}$:

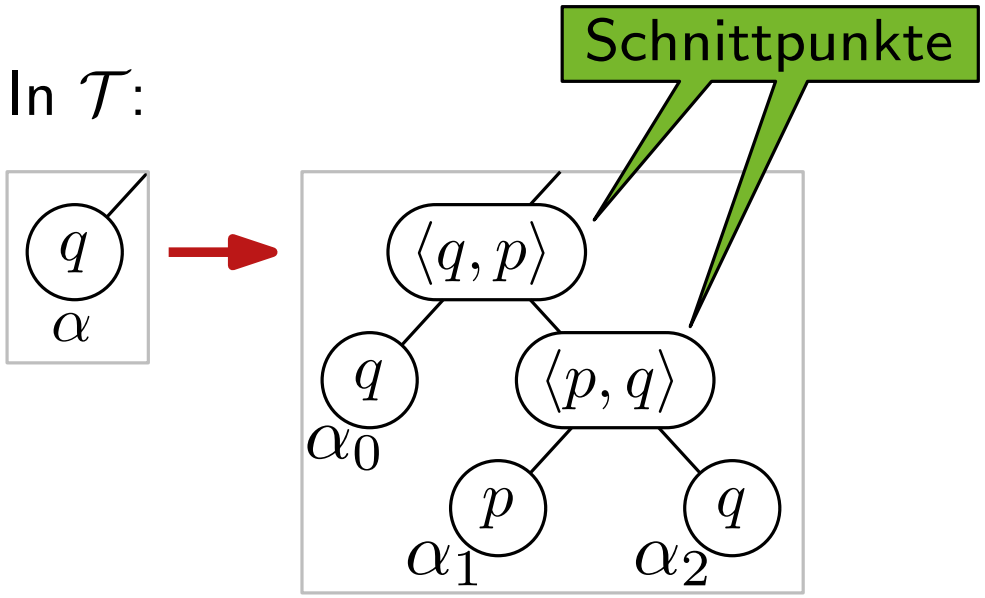


Punkt-Events behandeln

HandlePointEvent(Punkt p)

- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.
- Teile α in α_0 und α_2.
  Sei α_1 neuer Bogen für p.
- Füge Kanten $\langle q, p\rangle$ und $\langle p, q\rangle$ in $\mathcal{D}$ ein.

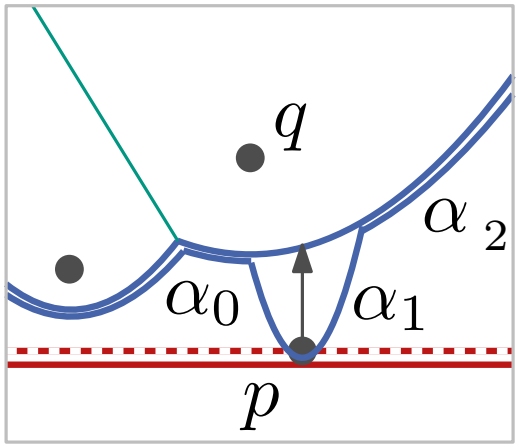




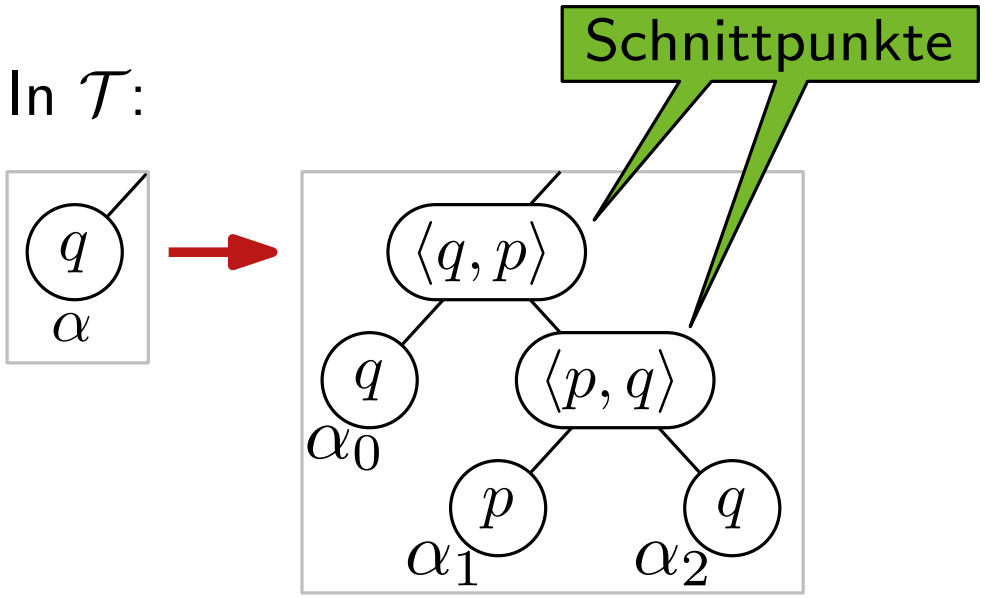
Punkt-Events behandeln

HandlePointEvent(Punkt p)

- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.
- Teile α in α_0 und α_2.
  Sei α_1 neuer Bogen für p.
- Füge Kanten $\langle q, p\rangle$ und $\langle p, q\rangle$ in $\mathcal{D}$ ein.
- Prüfe $\langle \cdot, \alpha_0, \alpha_1\rangle$ und $\langle \alpha_1, \alpha_2, \cdot\rangle$ auf Kreis-Events.



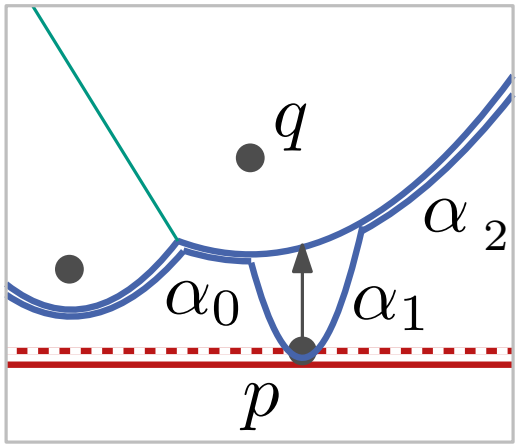
In $\mathcal{T}$:



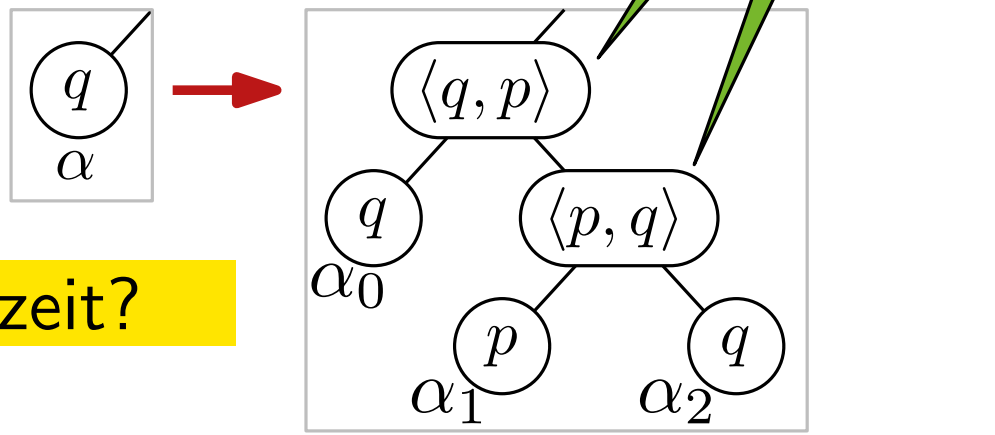
Punkt-Events behandeln

HandlePointEvent(Punkt p)

- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.
- Teile α in α_0 und α_2.
  Sei α_1 neuer Bogen für p.
- Füge Kanten $\langle q, p\rangle$ und $\langle p, q\rangle$ in $\mathcal{D}$ ein.
- Prüfe $\langle \cdot, \alpha_0, \alpha_1\rangle$ und $\langle \alpha_1, \alpha_2, \cdot\rangle$ auf Kreis-Events.



In $\mathcal{T}$:



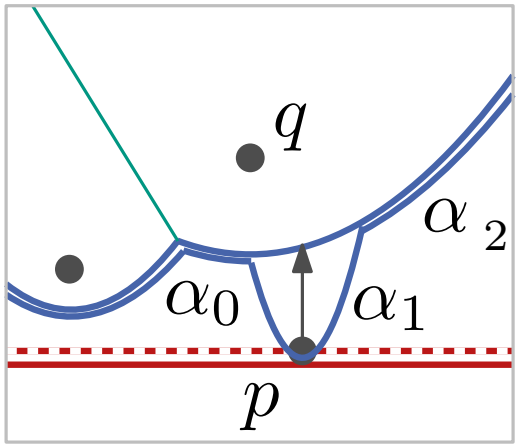
Laufzeit?

Punkt-Events behandeln

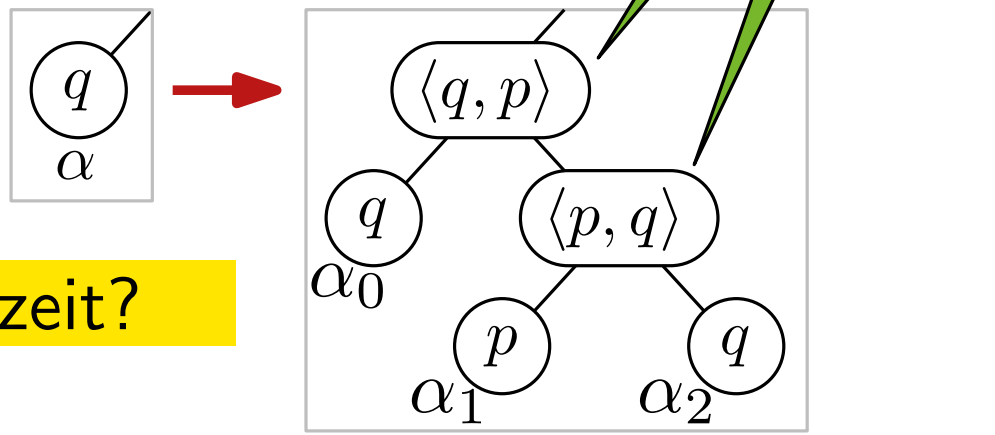
HandlePointEvent(Punkt p)

$O(\log n)$

- Suche in $\mathcal{T}$ den Bogen α vertikal über p.
  Hat α pointer auf Kreis-Event in $\mathcal{Q}$, lösche es aus $\mathcal{Q}$.
- Teile α in α_0 und α_2.
  Sei α_1 neuer Bogen für p.
- Füge Kanten $\langle q, p\rangle$ und $\langle p, q\rangle$ in $\mathcal{D}$ ein.
- Prüfe $\langle \cdot, \alpha_0, \alpha_1\rangle$ und $\langle \alpha_1, \alpha_2, \cdot\rangle$ auf Kreis-Events.



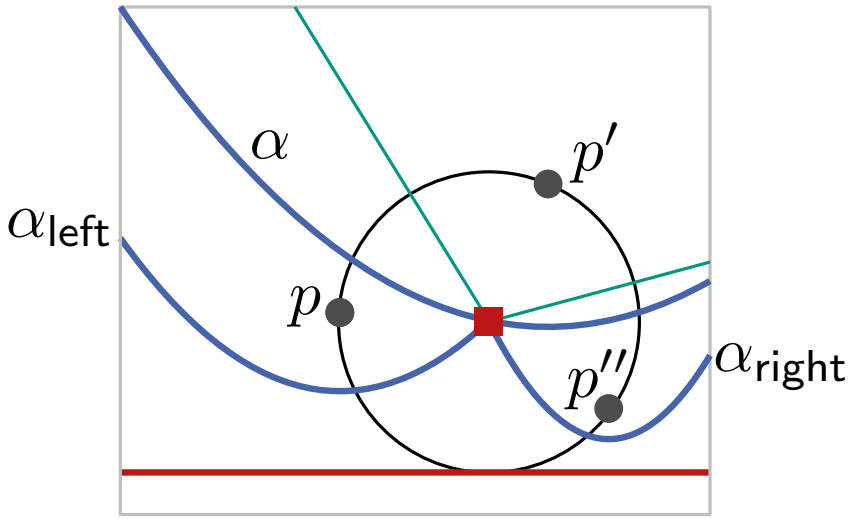
In $\mathcal{T}$:



Laufzeit?

Kreis-Events behandeln

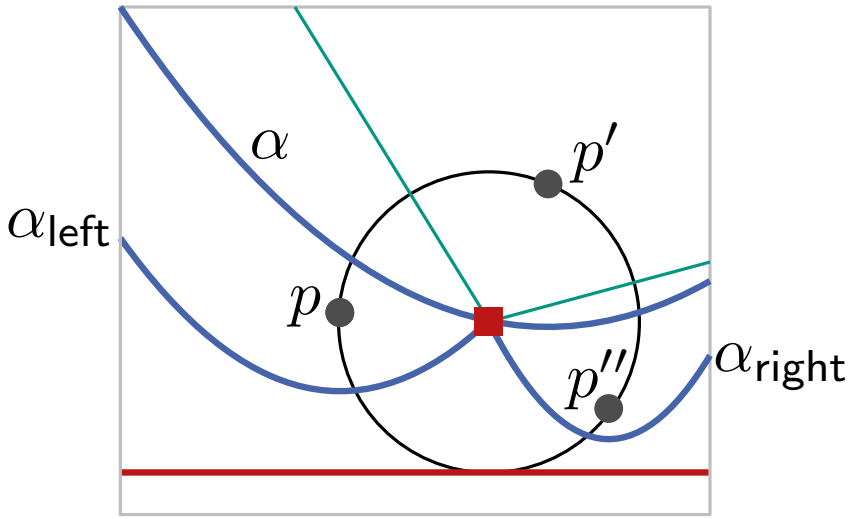
HandleCircleEvent(Bogen α)



Kreis-Events behandeln

HandleCircleEvent(Bogen α)

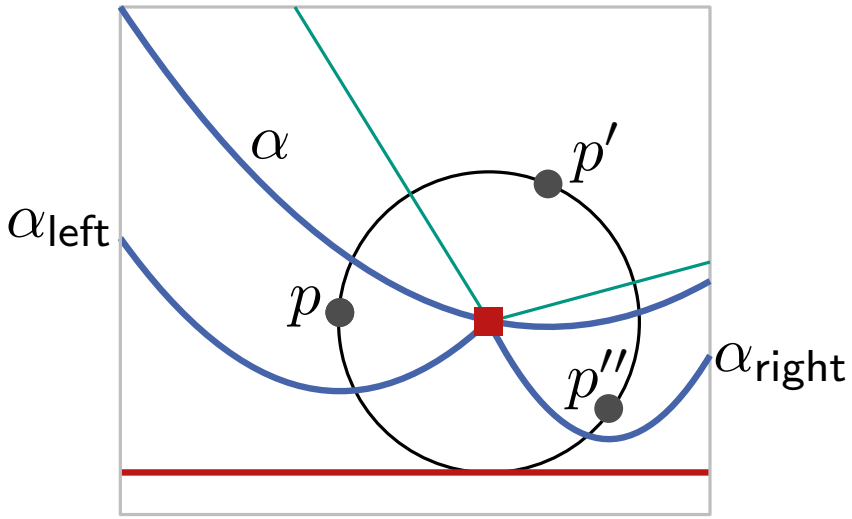
- $\mathcal{T}$.delete(α); Schnittpunkte in $\mathcal{T}$ updaten



Kreis-Events behandeln

HandleCircleEvent(Bogen α)

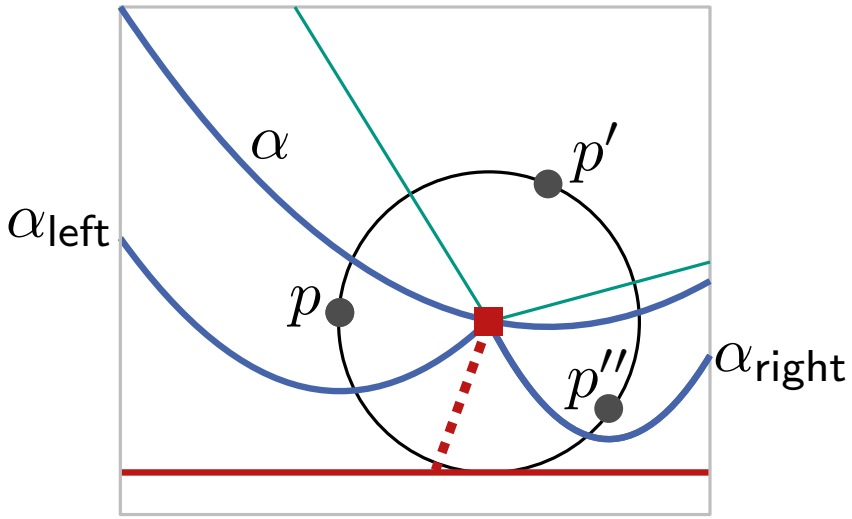
- $\mathcal{T}$.delete(α); Schnittpunkte in $\mathcal{T}$ updaten
- Entferne alle Kreis-Events zu α aus $\mathcal{Q}$.



Kreis-Events behandeln

HandleCircleEvent(Bogen α)

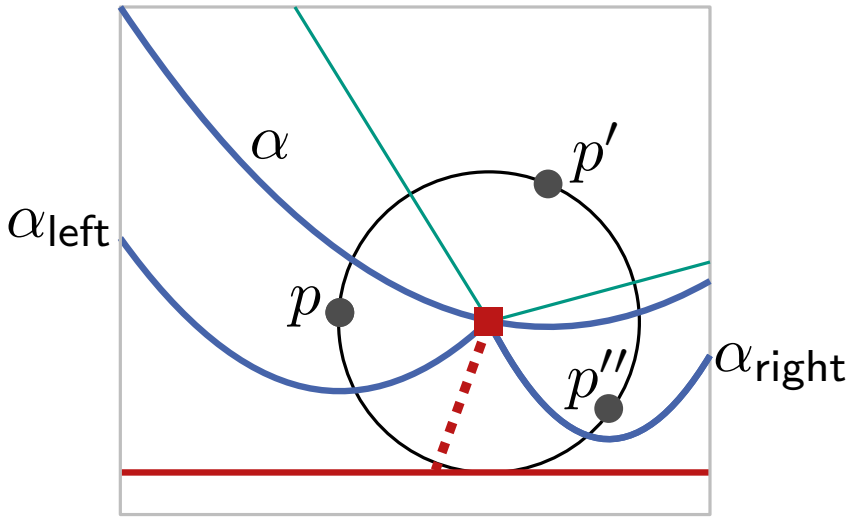
- $\mathcal{T}$.delete(α); Schnittpunkte in $\mathcal{T}$ updaten
- Entferne alle Kreis-Events zu α aus $\mathcal{Q}$.
- Füge Knoten $\mathcal{V}(\{p, p', p''\})$ und Kante $\mathcal{V}(\{p, p''\})$ in $\mathcal{D}$ ein.



Kreis-Events behandeln

HandleCircleEvent(Bogen α)

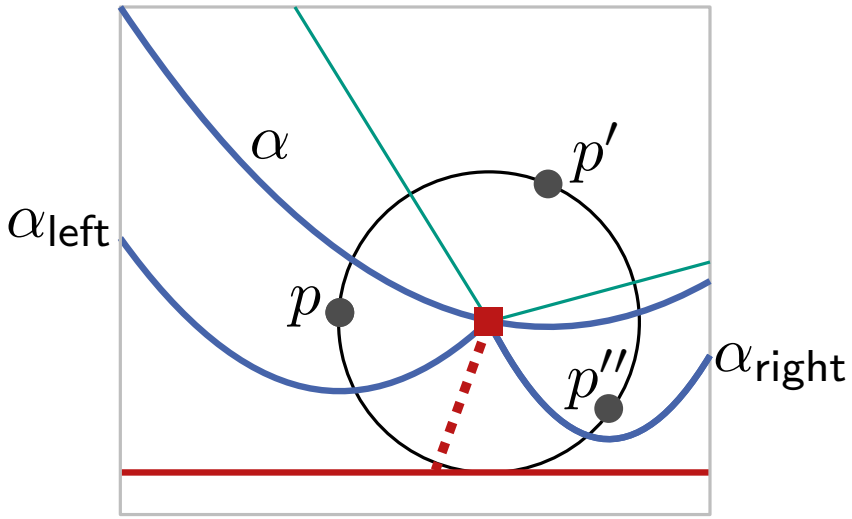
- $\mathcal{T}$.delete(α); Schnittpunkte in $\mathcal{T}$ updaten
- Entferne alle Kreis-Events zu α aus $\mathcal{Q}$.
- Füge Knoten $\mathcal{V}(\{p, p', p''\})$ und Kante $\mathcal{V}(\{p, p''\})$ in $\mathcal{D}$ ein.
- Update potenzielle Kreis-Events $\langle \cdot, \alpha_{\text{left}}, \alpha_{\text{right}} \rangle$ und $\langle \alpha_{\text{left}}, \alpha_{\text{right}}, \cdot \rangle$ in $\mathcal{Q}$.



Kreis-Events behandeln

HandleCircleEvent(Bogen α)

- $\mathcal{T}$.delete(α); Schnittpunkte in $\mathcal{T}$ updaten
- Entferne alle Kreis-Events zu α aus $\mathcal{Q}$.
- Füge Knoten $\mathcal{V}(\{p, p', p''\})$ und Kante $\mathcal{V}(\{p, p''\})$ in $\mathcal{D}$ ein.
- Update potenzielle Kreis-Events $\langle \cdot, \alpha_{\text{left}}, \alpha_{\text{right}} \rangle$ und $\langle \alpha_{\text{left}}, \alpha_{\text{right}}, \cdot \rangle$ in $\mathcal{Q}$.



Laufzeit?

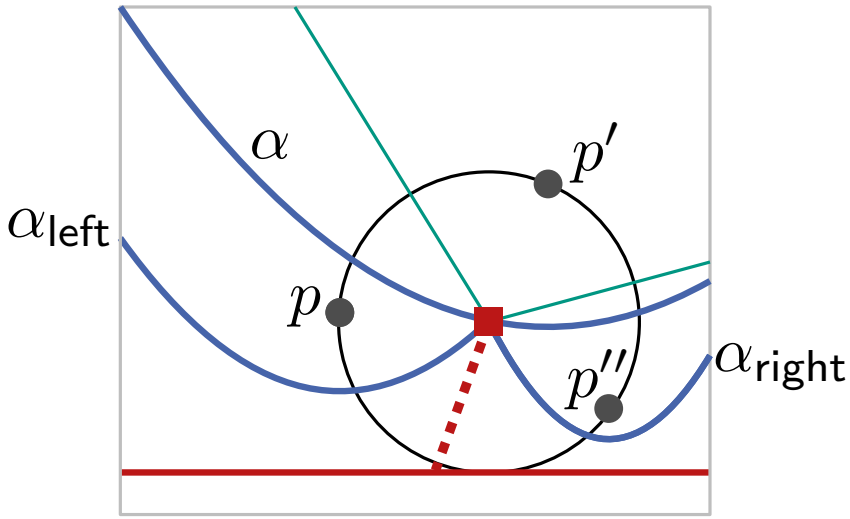
Kreis-Events behandeln

Karlsruhe Institute of Technology

$O(\log n)$

HandleCircleEvent(Bogen α)

- $\mathcal{T}$.delete(α); Schnittpunkte in $\mathcal{T}$ updaten
- Entferne alle Kreis-Events zu α aus $\mathcal{Q}$.
- Füge Knoten $\mathcal{V}(\{p, p', p''\})$ und Kante $\mathcal{V}(\{p, p''\})$ in $\mathcal{D}$ ein.
- Update potenzielle Kreis-Events $\langle \cdot, \alpha_{\text{left}}, \alpha_{\text{right}} \rangle$ und $\langle \alpha_{\text{left}}, \alpha_{\text{right}}, \cdot \rangle$ in $\mathcal{Q}$.



Laufzeit?

Fortune's Sweep Algorithmus

Satz: Für eine Menge P von n Punkten berechnet Fortune's Sweep Algorithmus das Voronoi-Diagramm $\mathrm{Vor}(P)$ in $O(n \log n)$ Zeit und $O(n)$ Platz.

Fortune's Sweep Algorithmus

Satz: Für eine Menge P von n Punkten berechnet Fortune's Sweep Algorithmus das Voronoi-Diagramm $\mathrm{Vor}(P)$ in $O(n \log n)$ Zeit und $O(n)$ Platz.

Beweisskizze:

- jedes Event benötigt $O(\log n)$ Zeit
- n Punkt-Events
- $\leq 2n - 5$ Kreis-Events ($=$ #Knoten von $\mathrm{Vor}(P)$)
- Fehlalarme erzeugen & löschen bereits inklusive

Fortune's Sweep Algorithmus

Satz: Für eine Menge P von n Punkten berechnet Fortune's Sweep Algorithmus das Voronoi-Diagramm $\mathrm{Vor}(P)$ in $O(n \log n)$ Zeit und $O(n)$ Platz.

Beweisskizze:

- jedes Event benötigt $O(\log n)$ Zeit
- n Punkt-Events
- $\leq 2n - 5$ Kreis-Events ($=$ #Knoten von $\mathrm{Vor}(P)$)
- Fehlalarme erzeugen & löschen bereits inklusive

Bemerkung:

degenerierte Eingaben diesmal unproblematisch:

- Kreis-Events vor Punkt-Events, sonst Reihenfolge beliebig
- Kreis-Events für $k \geq 4$ Punkte: Länge-0 Kanten und Knotenduplikate im Postprocessing entfernen
- zweideutiges Punktevent: beliebigen Bogen wählen

Diskussion

Karlsruhe Institute of Technology

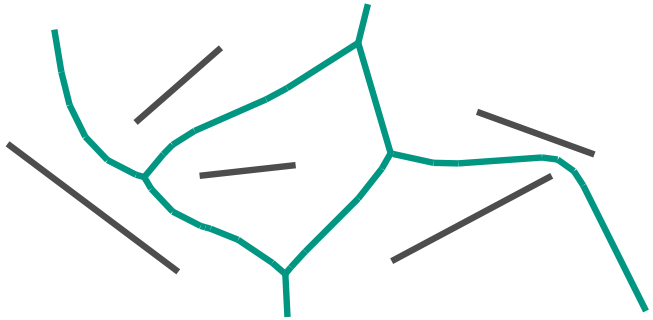
Gibt es weitere Varianten von Voronoi-Diagrammen?

Diskussion

Gibt es weitere Varianten von Voronoi-Diagrammen?

Ja! Beispielsweise kann der Algorithmus bei gleicher Laufzeit und Platzbedarf auch für Voronoi-Diagramme von Strecken modifiziert werden.

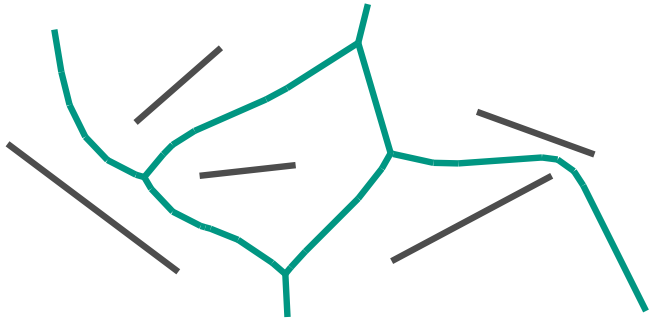
Auch andere Metriken wie L_p oder additiv/ multiplikativ gewichtete Voronoi-Diagramme sind möglich.



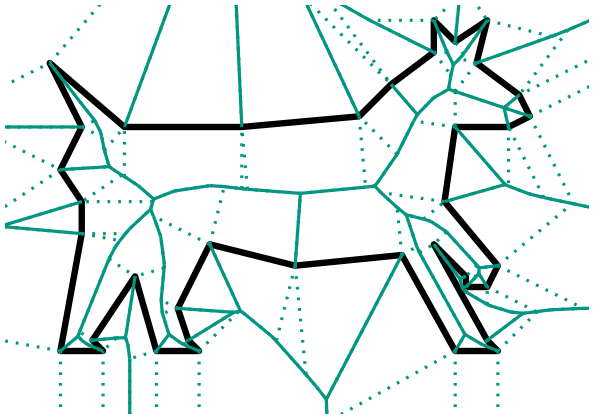
Diskussion

Gibt es weitere Varianten von Voronoi-Diagrammen?

Ja! Beispielsweise kann der Algorithmus bei gleicher Laufzeit und Platzbedarf auch für Voronoi-Diagramme von Strecken modifiziert werden.



Auch andere Metriken wie L_p oder additiv/ multiplikativ gewichtete Voronoi-Diagramme sind möglich.



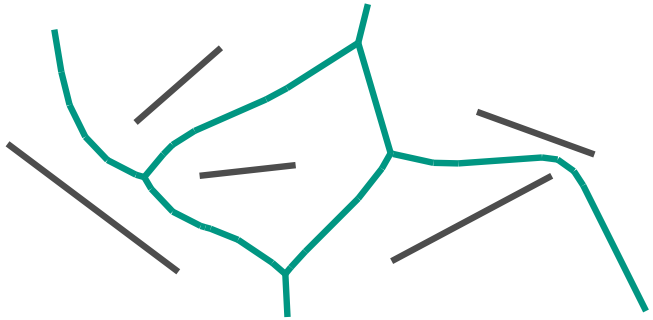
Voronoi-Diagramm für Polygone definieren die sog. Mittelachse, die z.B. in der Bildverarbeitung wichtig ist.

Auch farthest-point Voronoi-Diagramme sind möglich.

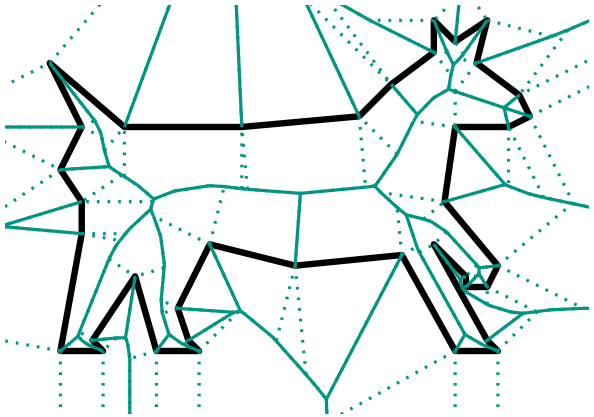
Diskussion

Gibt es weitere Varianten von Voronoi-Diagrammen?

Ja! Beispielsweise kann der Algorithmus bei gleicher Laufzeit und Platzbedarf auch für Voronoi-Diagramme von Strecken modifiziert werden.



Auch andere Metriken wie L_p oder additiv/ multiplikativ gewichtete Voronoi-Diagramme sind möglich.



Voronoi-Diagramm für Polygone definieren die sog. Mittelachse, die z.B. in der Bildverarbeitung wichtig ist.

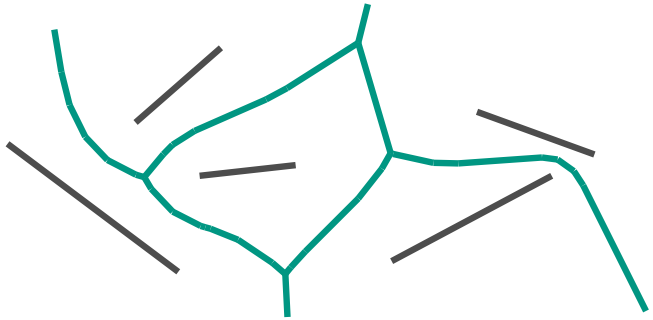
Auch farthest-point Voronoi-Diagramme sind möglich.

Was passiert in höheren Dimensionen?

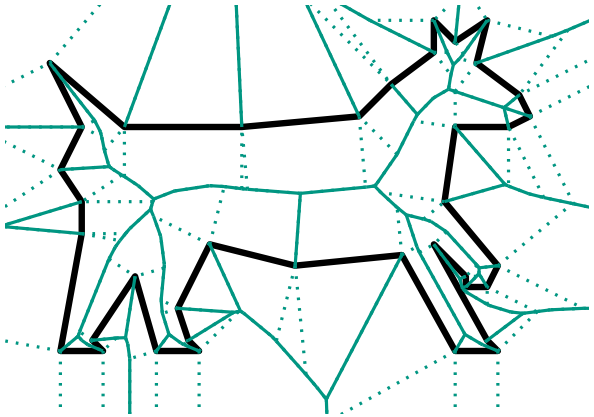
Diskussion

Gibt es weitere Varianten von Voronoi-Diagrammen?

Ja! Beispielsweise kann der Algorithmus bei gleicher Laufzeit und Platzbedarf auch für Voronoi-Diagramme von Strecken modifiziert werden.



Auch andere Metriken wie L_p oder additiv/ multiplikativ gewichtete Voronoi-Diagramme sind möglich.



Voronoi-Diagramm für Polygone definieren die sog. Mittelachse, die z.B. in der Bildverarbeitung wichtig ist.

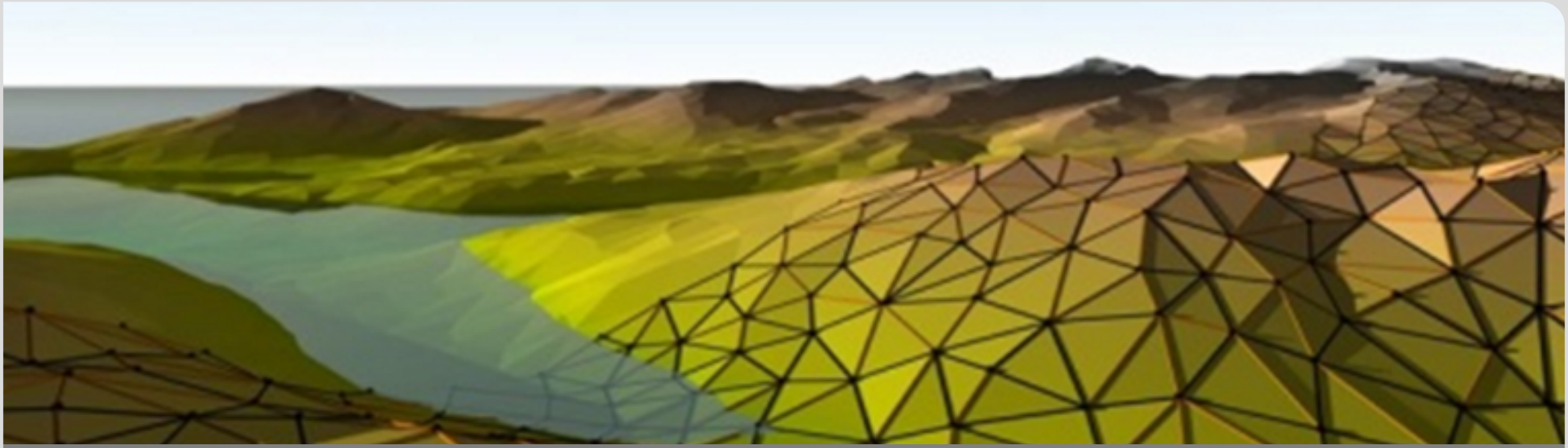
Auch farthest-point Voronoi-Diagramme sind möglich.

Was passiert in höheren Dimensionen?

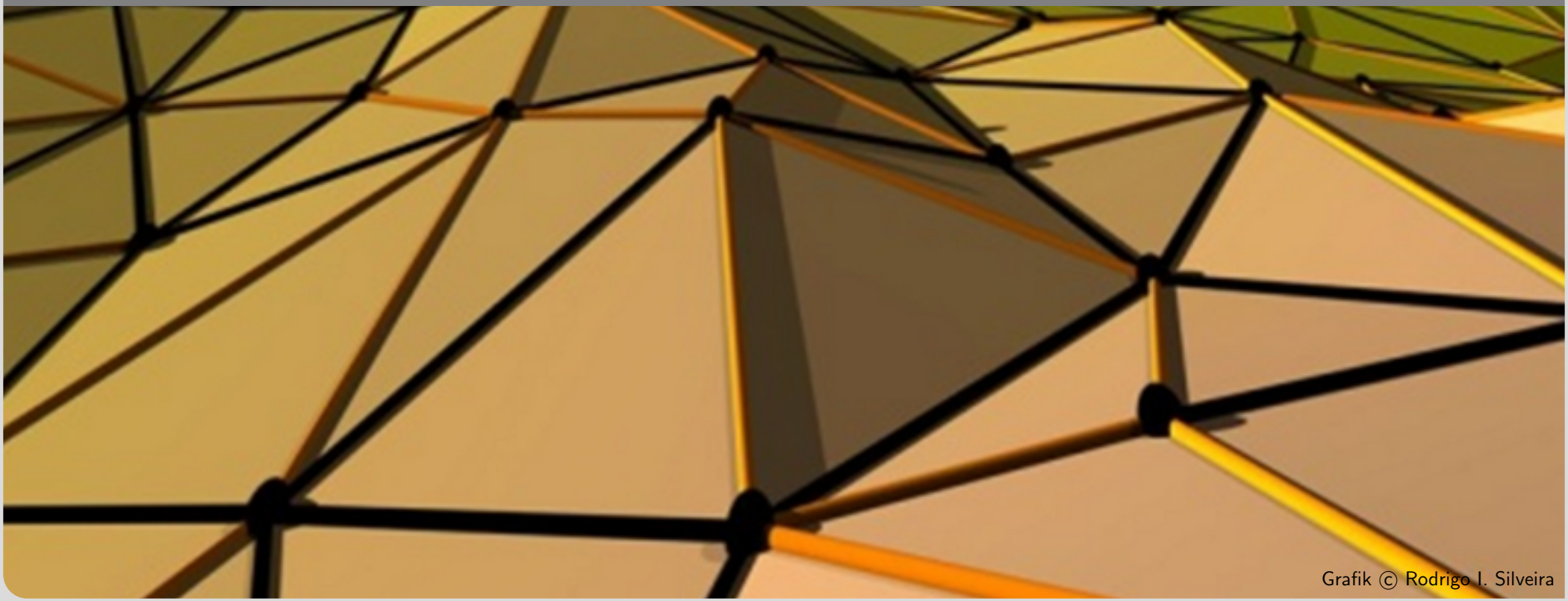
Die Komplexität von $\mathrm{Vor}(P)$ steigt auf $\Theta(n^{\lceil d/2 \rceil})$ und die Laufzeit zur Berechnung auf $O(n \log n + n^{\lceil d/2 \rceil})$.

Anregungen aus der Zwischenevaluation

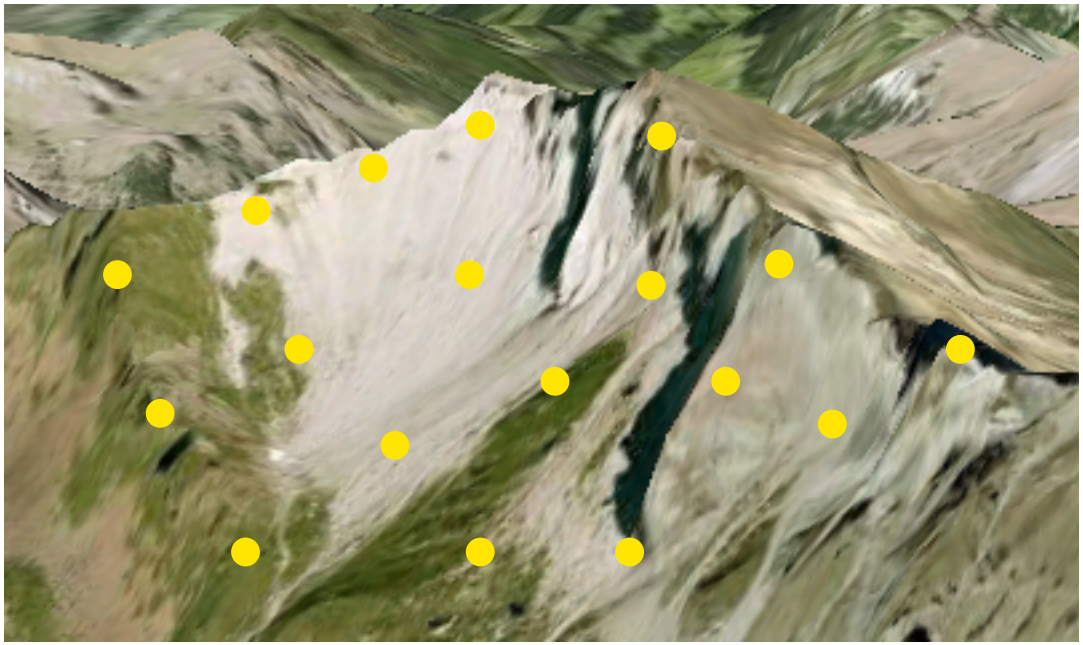
- **Druckversion der Folien**
  ab sofort online verfügbar
- **Beweise und Tafel/Folien**
  alle Beweise in der Literatur (meist [BCKO08]) detailliert nachlesbar
- **Musterlösung für Übungsblätter**
  Andreas wird die Lösungsideen schriftlich skizzieren
- **Folien schneller und fehlerfrei online stellen**
  sind auch bisher immer kurz nach der Vorlesung verfügbar; bei Fehlern bitte Bescheid geben!



Delaunay-Triangulierungen



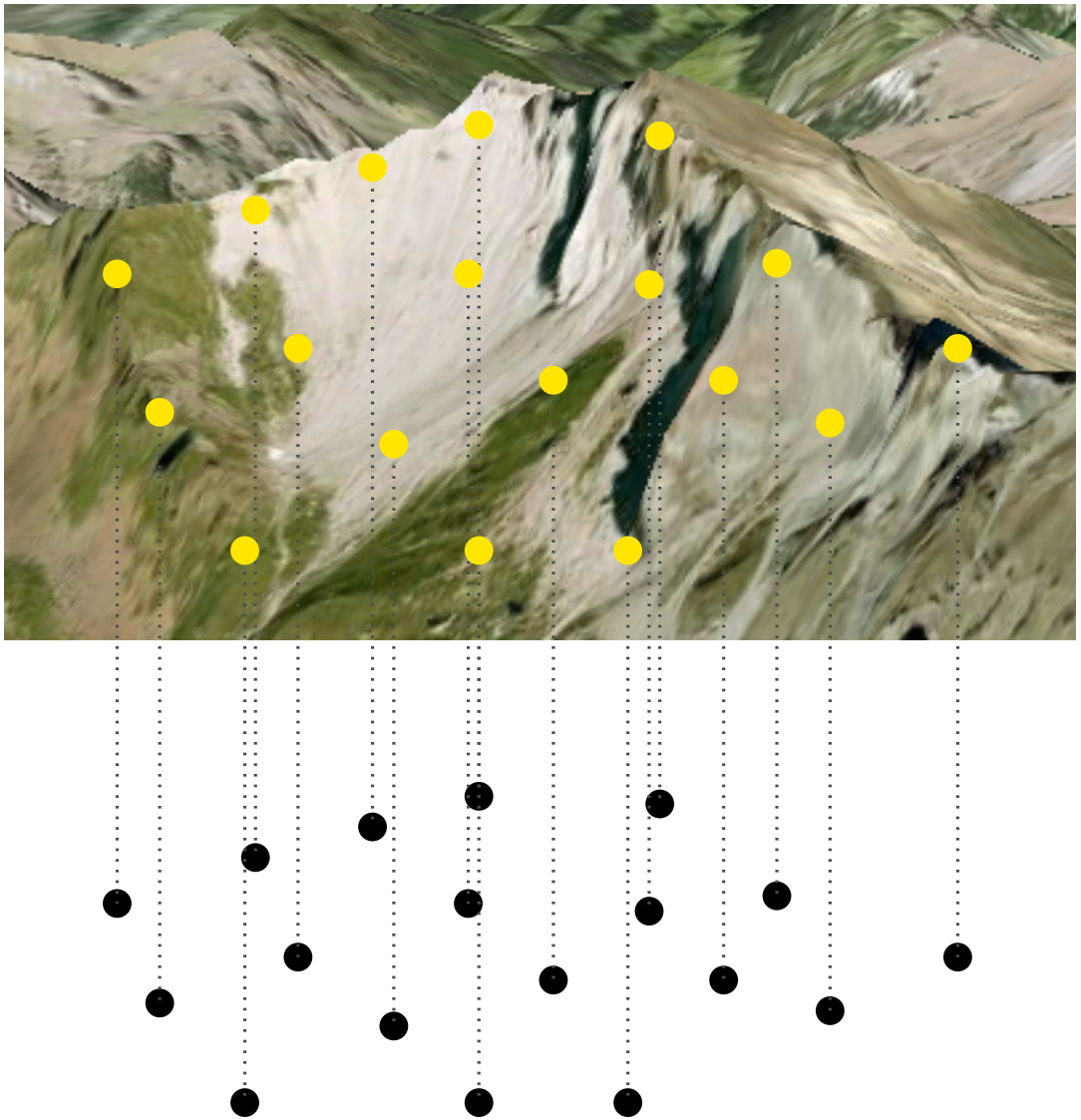
Terrainmodellierung



Höhenmesspunkte

$p = (x_p, y_p, z_p)$

Terrainmodellierung



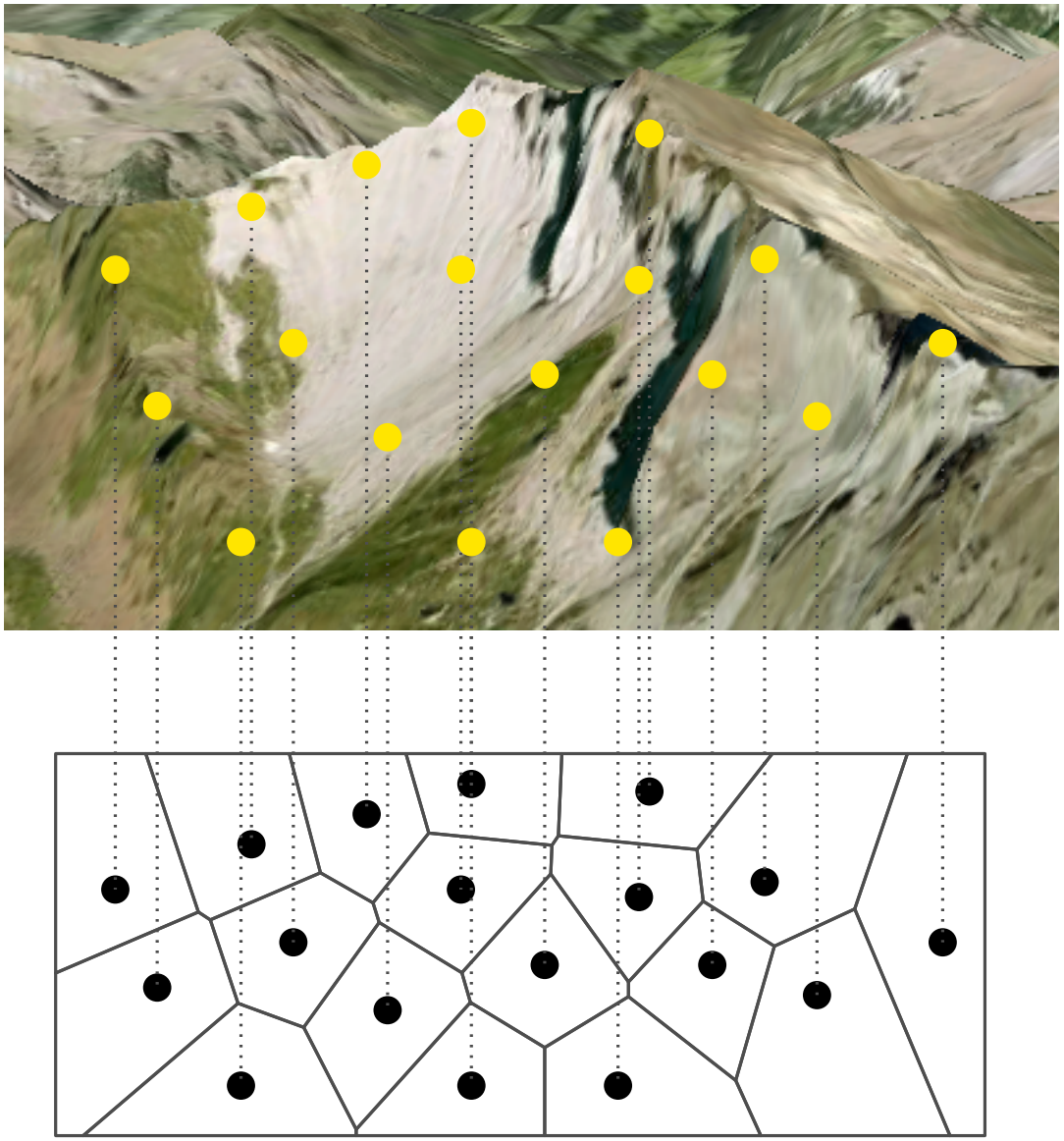
Höhenmesspunkte

$p = (x_p, y_p, z_p)$

Projektion

$\pi(p) = (p_x, p_y, 0)$

Terrainmodellierung



Höhenmesspunkte

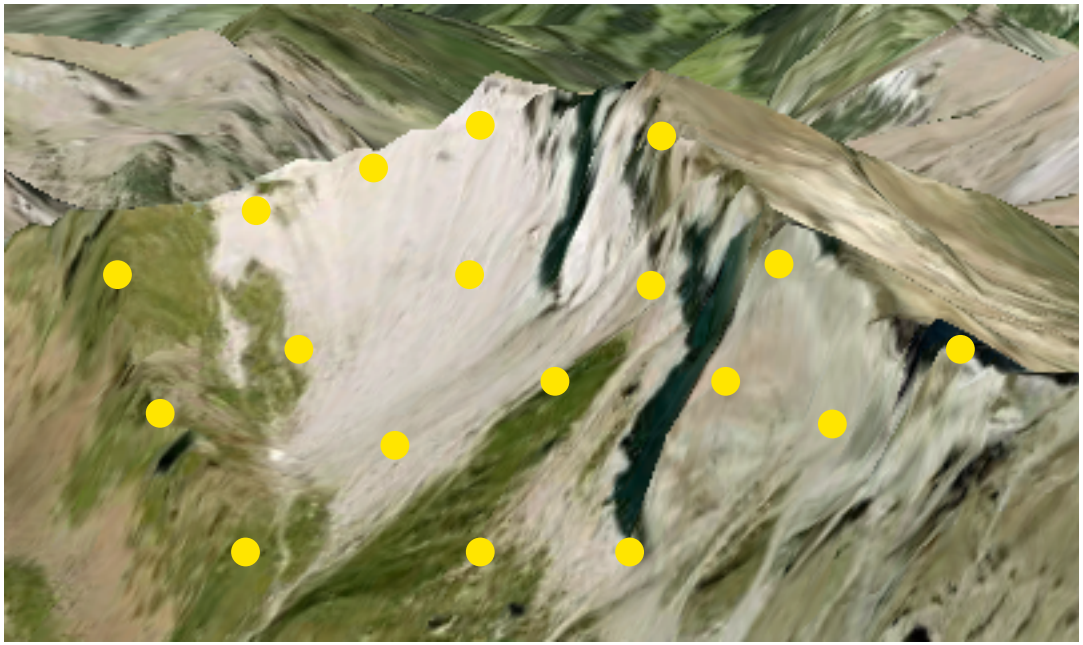
$p = (x_p, y_p, z_p)$

Projektion

$\pi(p) = (p_x, p_y, 0)$

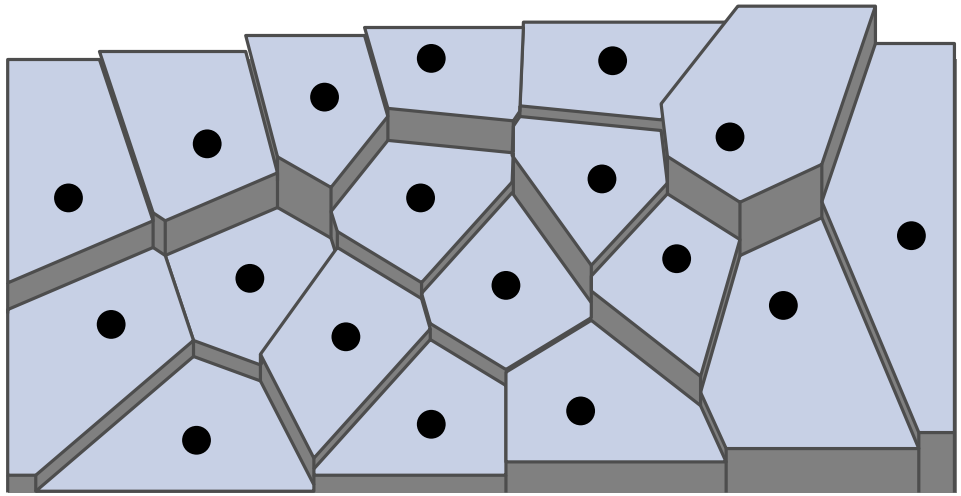
Interpolation 1: jeder Punkt bekommt Höhe des nächsten Messpunktes

Terrainmodellierung



Höhenmesspunkte

$p = (x_p, y_p, z_p)$

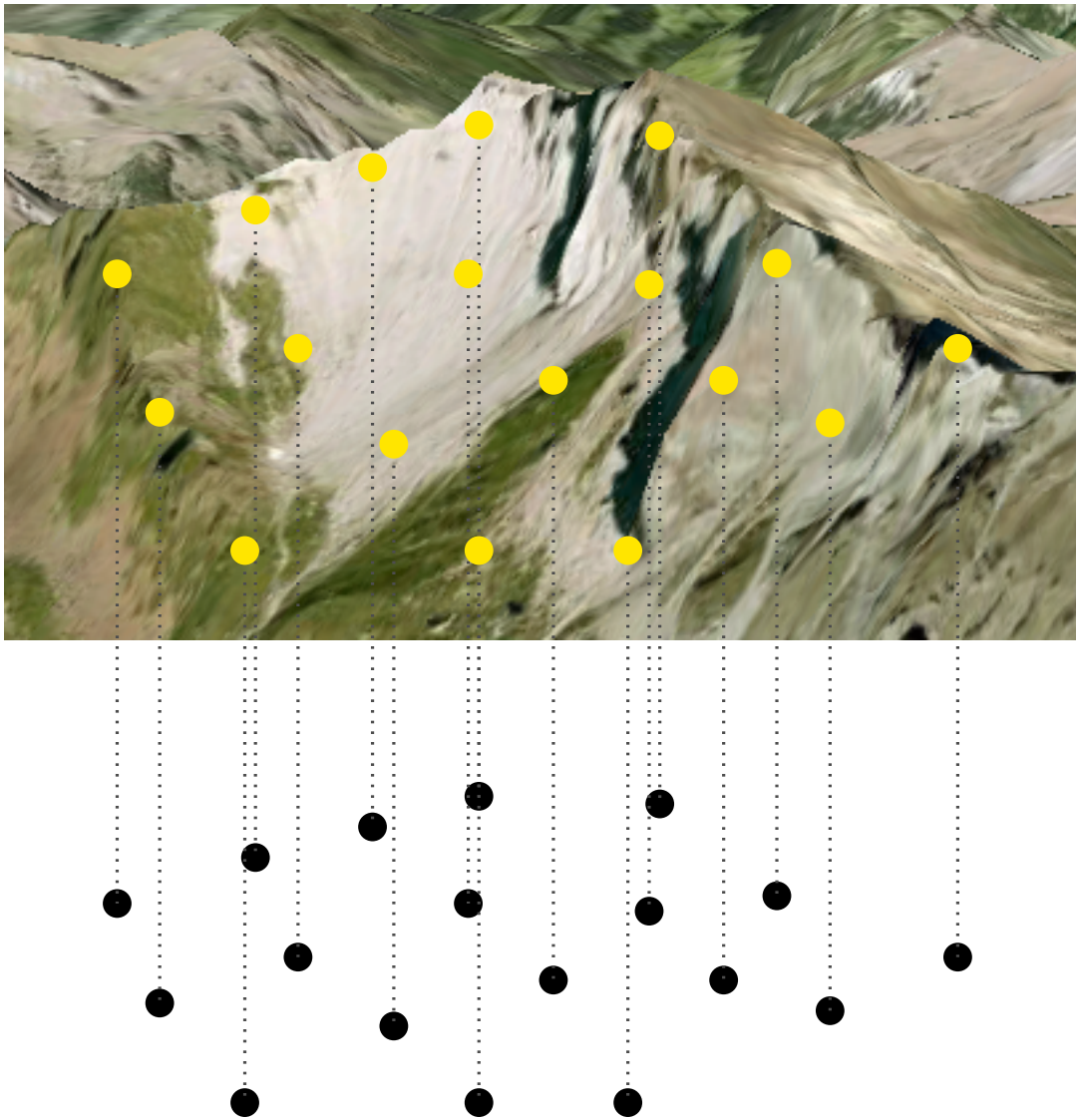


Projektion

$\pi(p) = (p_x, p_y, 0)$

Interpolation 1: jeder Punkt bekommt Höhe des nächsten Messpunktes

Terrainmodellierung



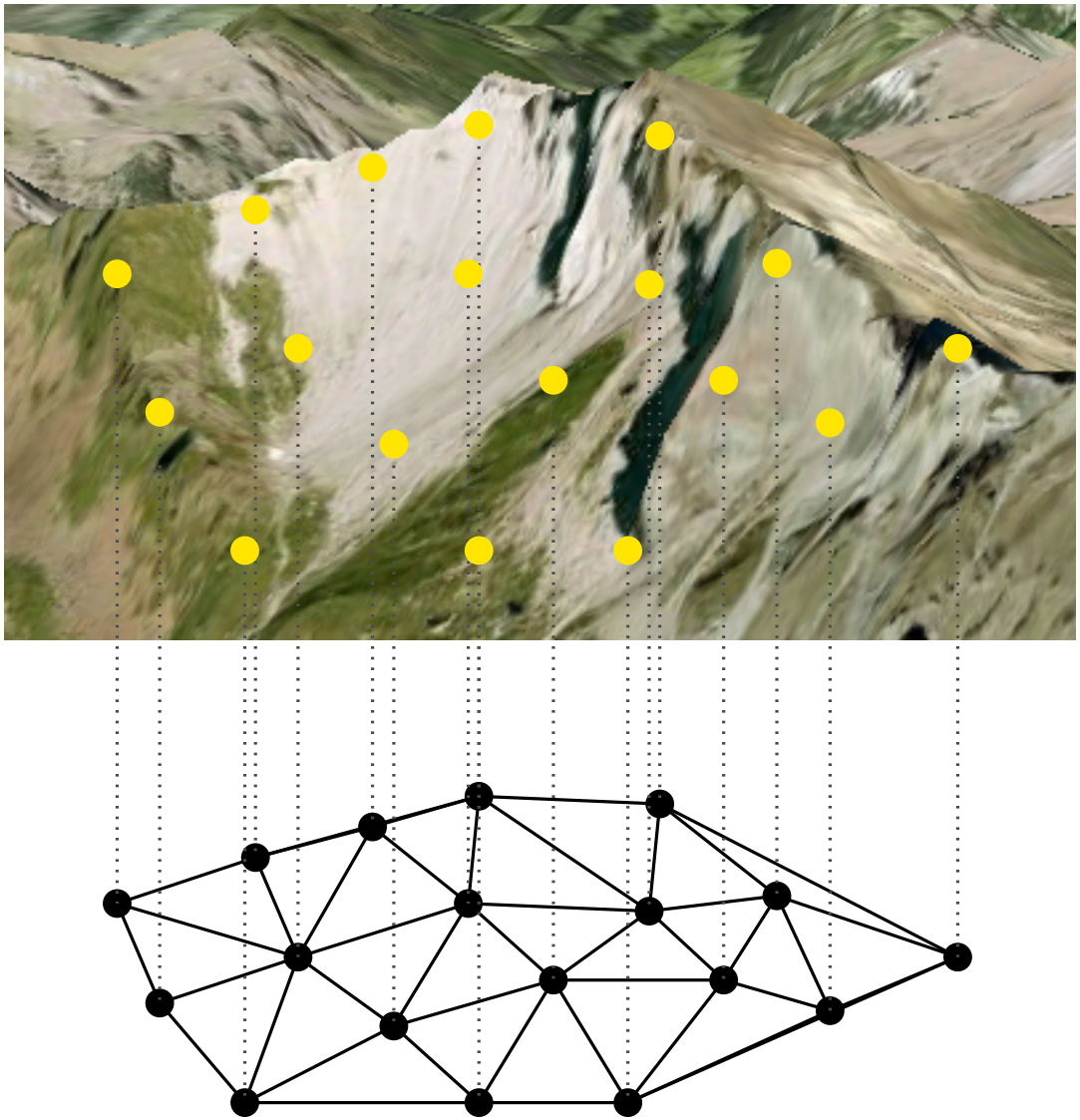
Höhenmesspunkte

$p = (x_p, y_p, z_p)$

Projektion

$\pi(p) = (p_x, p_y, 0)$

Terrainmodellierung



Höhenmesspunkte

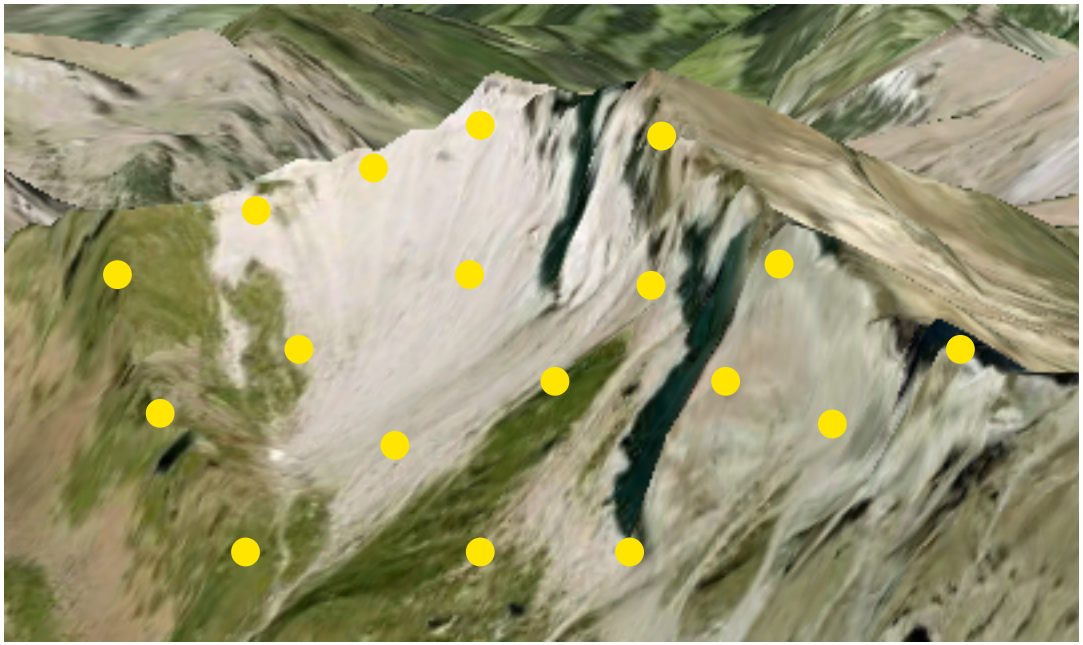
$p = (x_p, y_p, z_p)$

Projektion

$\pi(p) = (p_x, p_y, 0)$

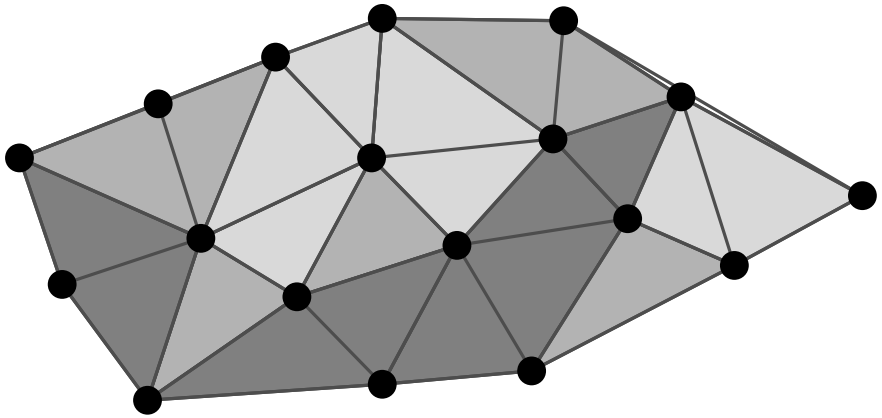
Interpolation 2: trianguliere Messpunkte und interpoliere auf Dreiecken

Terrainmodellierung



Höhenmesspunkte

$p = (x_p, y_p, z_p)$

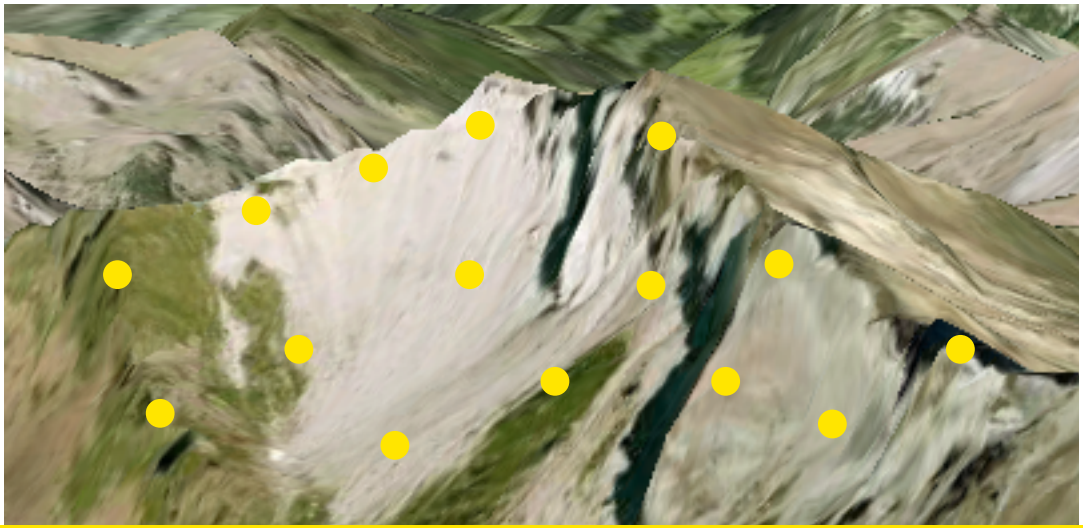


Projektion

$\pi(p) = (p_x, p_y, 0)$

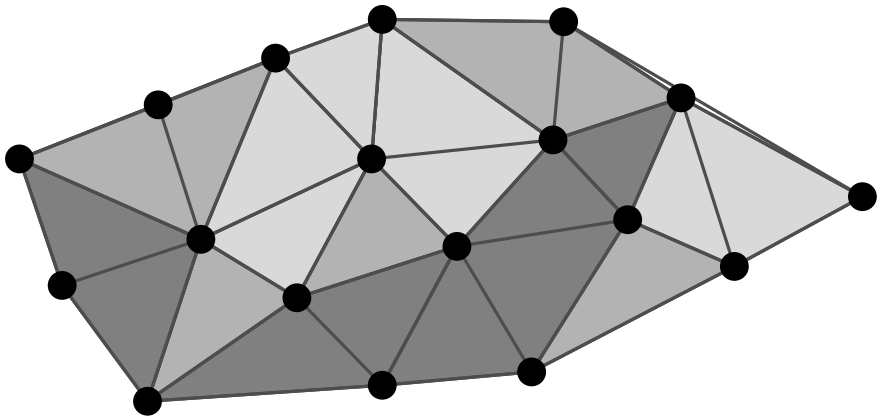
Interpolation 2: trianguliere Messpunkte und interpoliere auf Dreiecken

Terrainmodellierung



Höhenmesspunkte
$p = (x_p, y_p, z_p)$

Aber wie sieht eine gute Triangulierung aus?

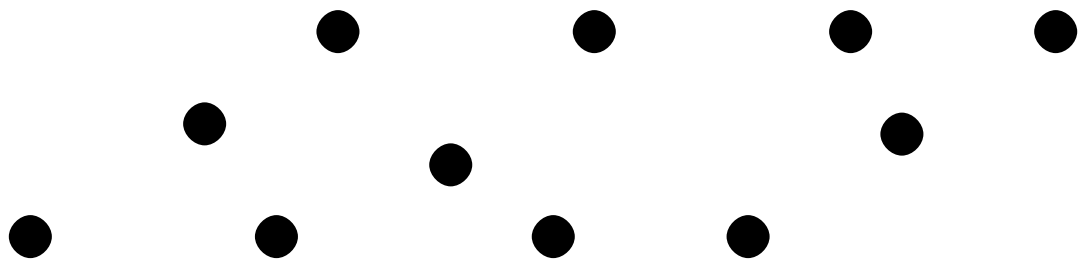


Projektion
$\pi(p) = (p_x, p_y, 0)$

Interpolation 2: trianguliere Messpunkte und interpoliere auf Dreiecken

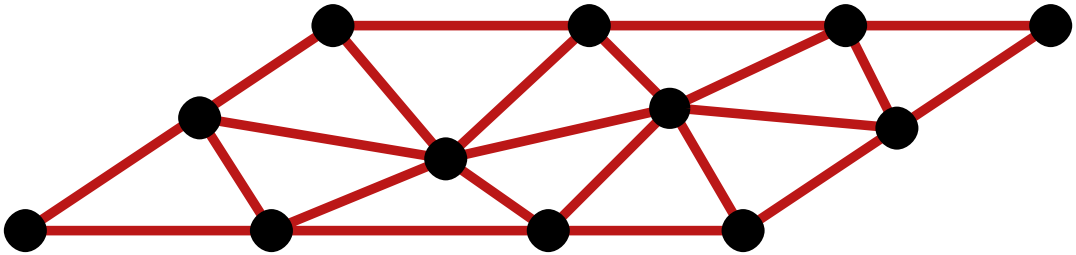
Triangulierung von Punkten

Def.: Eine **Triangulierung** einer Punktmenge $P \subset \mathbb{R}^2$ ist eine maximale planare Unterteilung mit Knotenmenge P.



Triangulierung von Punkten

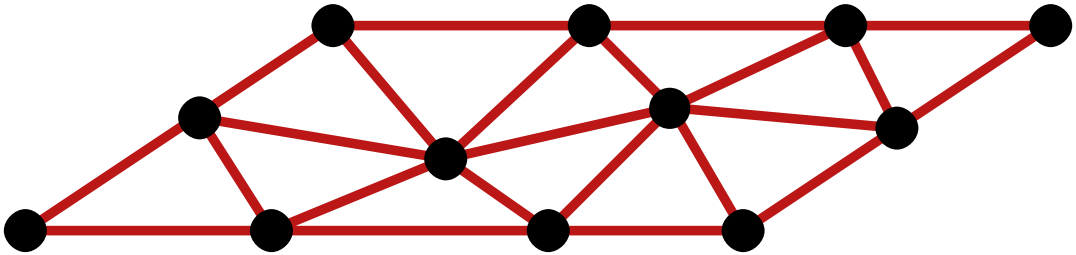
Def.: Eine **Triangulierung** einer Punktmenge $P \subset \mathbb{R}^2$ ist eine maximale planare Unterteilung mit Knotenmenge P.



Beob.:

Triangulierung von Punkten

Def.: Eine **Triangulierung** einer Punktmenge $P \subset \mathbb{R}^2$ ist eine maximale planare Unterteilung mit Knotenmenge P.

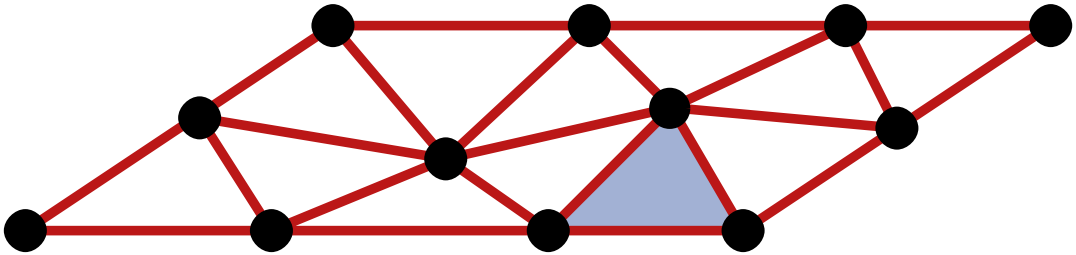


Beob.:

- alle inneren Facetten sind Dreiecke

Triangulierung von Punkten

Def.: Eine **Triangulierung** einer Punktmenge $P \subset \mathbb{R}^2$ ist eine maximale planare Unterteilung mit Knotenmenge P.

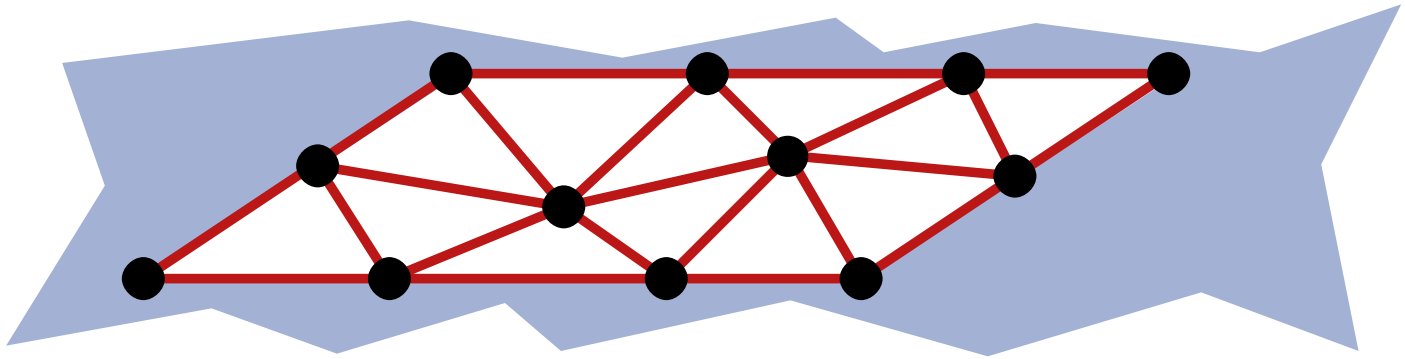


Beob.:

- alle inneren Facetten sind Dreiecke

Triangulierung von Punkten

Def.: Eine **Triangulierung** einer Punktmenge $P \subset \mathbb{R}^2$ ist eine maximale planare Unterteilung mit Knotenmenge P.

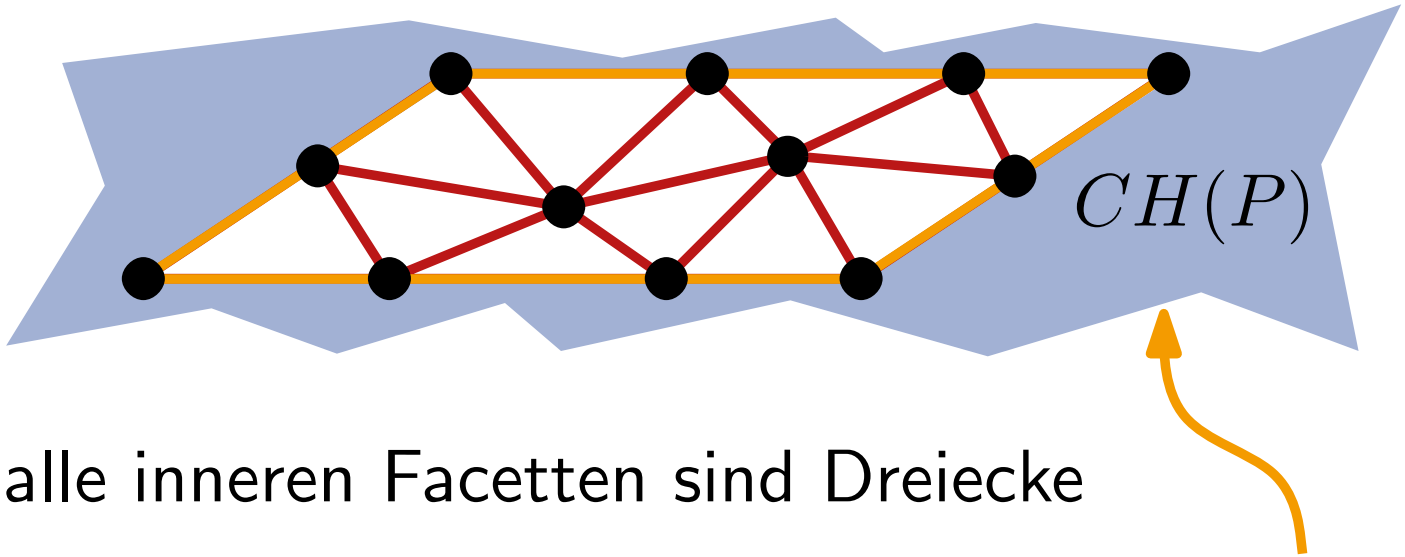


Beob.:

- alle inneren Facetten sind Dreiecke
- äußere Facette ist Komplement der konvexen Hülle

Triangulierung von Punkten

Def.: Eine **Triangulierung** einer Punktmenge $P \subset \mathbb{R}^2$ ist eine maximale planare Unterteilung mit Knotenmenge P.

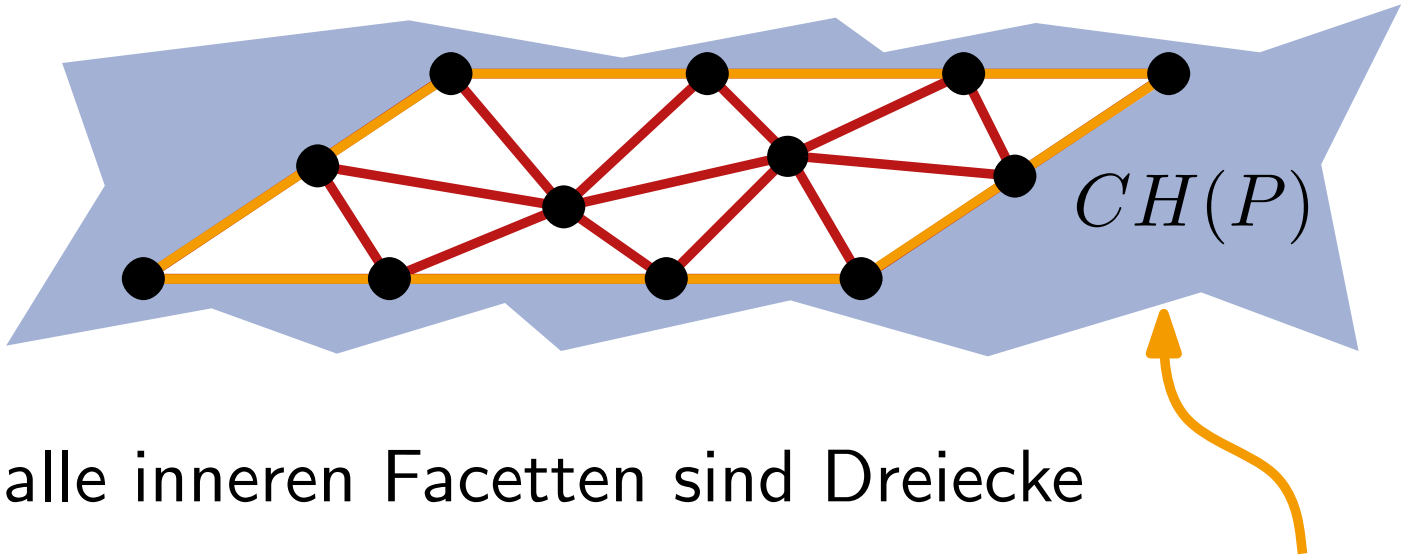


Beob.:

- alle inneren Facetten sind Dreiecke
- äußere Facette ist Komplement der konvexen Hülle

Triangulierung von Punkten

Def.: Eine **Triangulierung** einer Punktmenge $P \subset \mathbb{R}^2$ ist eine maximale planare Unterteilung mit Knotenmenge P.



Beob.:

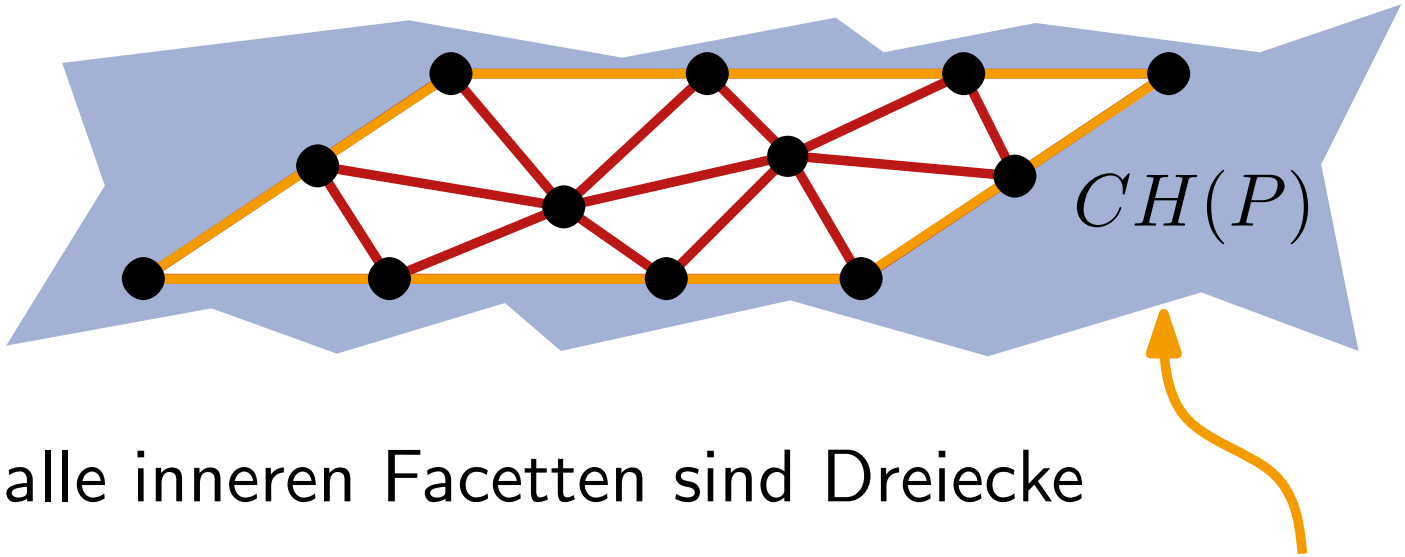
- alle inneren Facetten sind Dreiecke
- äußere Facette ist Komplement der konvexen Hülle

Satz 1: Sei P eine Menge von n nicht kollinearen Punkten und h die Anzahl Punkte auf $CH(P)$.

Dann hat *jede* Triangulierung von P $t(n,h)$ Dreiecke und $e(n,h)$ Kanten.

Triangulierung von Punkten

Def.: Eine **Triangulierung** einer Punktmenge $P \subset \mathbb{R}^2$ ist eine maximale planare Unterteilung mit Knotenmenge P.



Beob.:

- alle inneren Facetten sind Dreiecke
- äußere Facette ist Komplement der konvexen Hülle

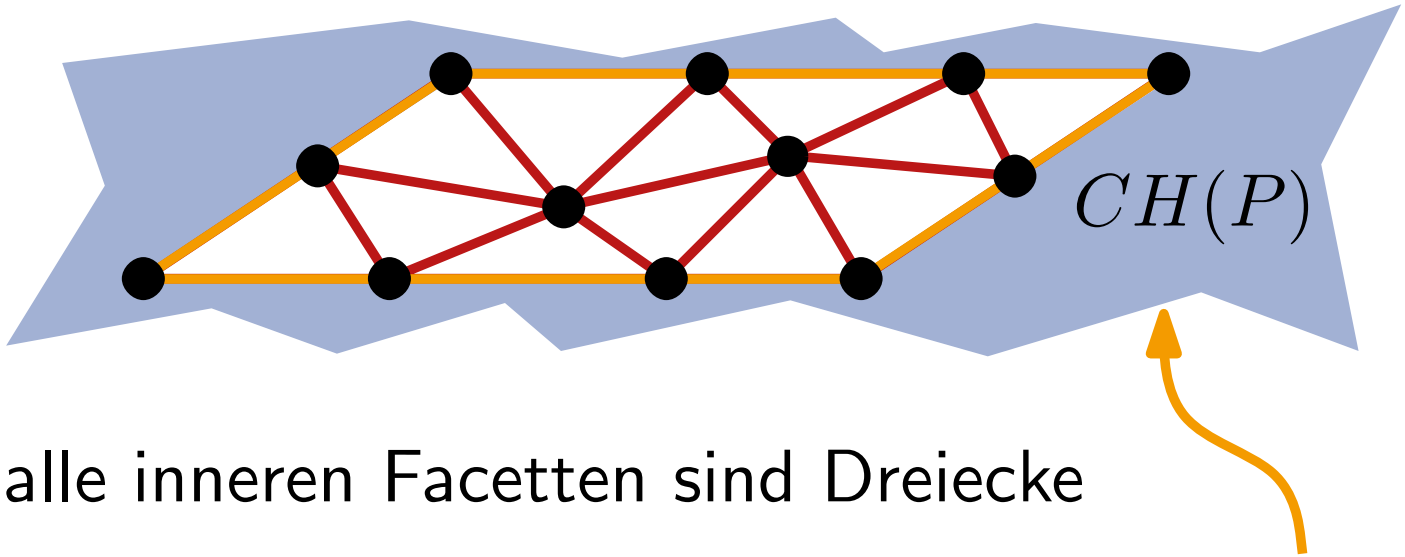
Satz 1: Sei P eine Menge von n nicht kollinearen Punkten und h die Anzahl Punkte auf $CH(P)$.

Dann hat *jede* Triangulierung von P $t(n,h)$ Dreiecke und $e(n,h)$ Kanten.

Berechne t und e!

Triangulierung von Punkten

Def.: Eine **Triangulierung** einer Punktmenge $P \subset \mathbb{R}^2$ ist eine maximale planare Unterteilung mit Knotenmenge P.



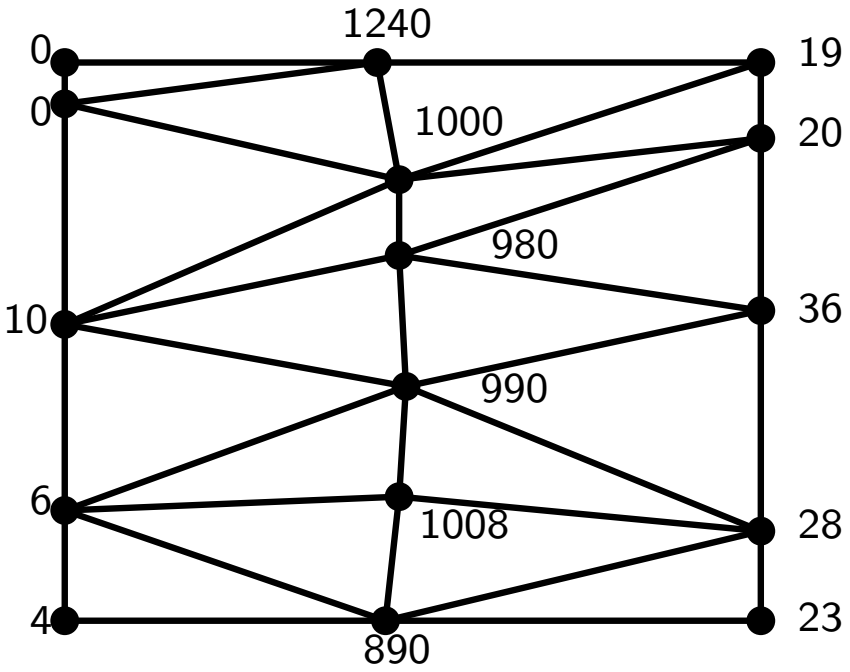
Beob.:

- alle inneren Facetten sind Dreiecke
- äußere Facette ist Komplement der konvexen Hülle

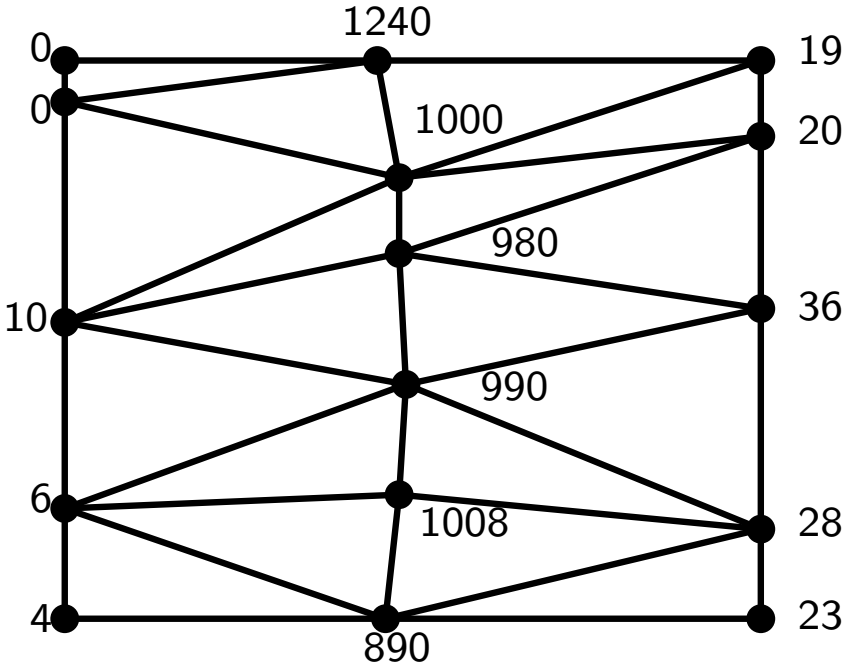
Satz 1: Sei P eine Menge von n nicht kollinearen Punkten und h die Anzahl Punkte auf $CH(P)$.

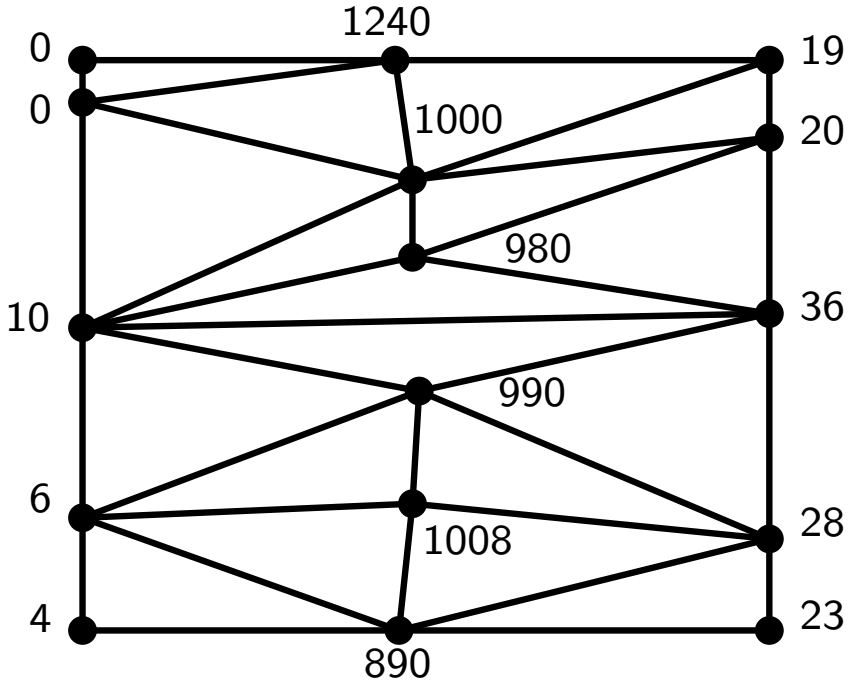
Dann hat *jede* Triangulierung von P $(2n-2-h)$ Dreiecke und $(3n-3-h)$ Kanten.

Zurück zur Höheninterpolation

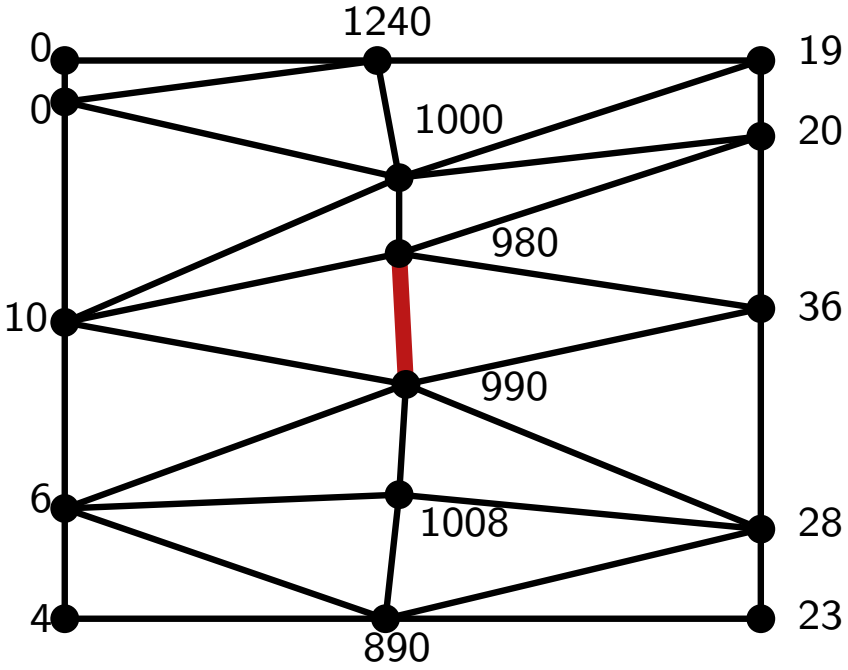


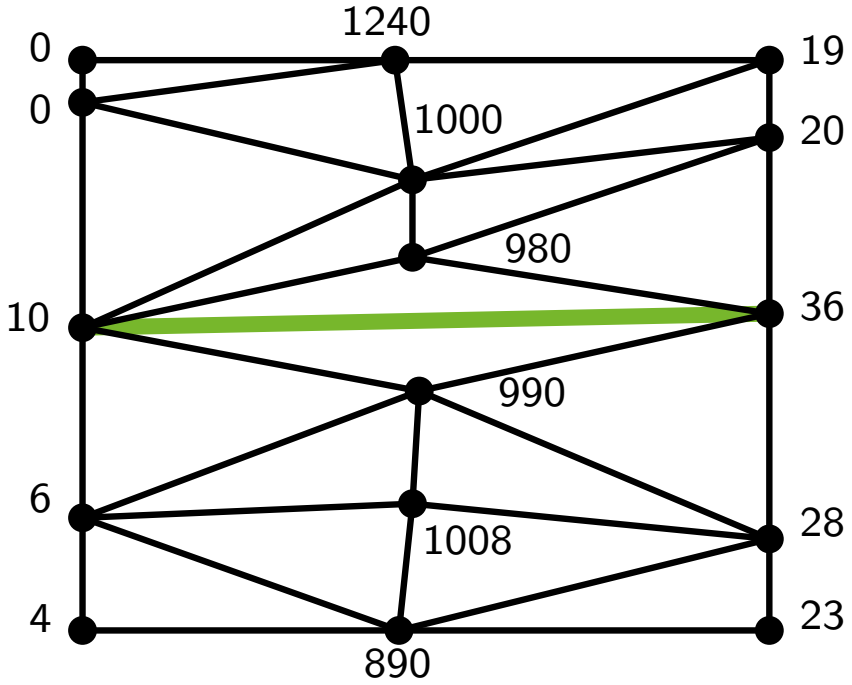
Zurück zur Höheninterpolation



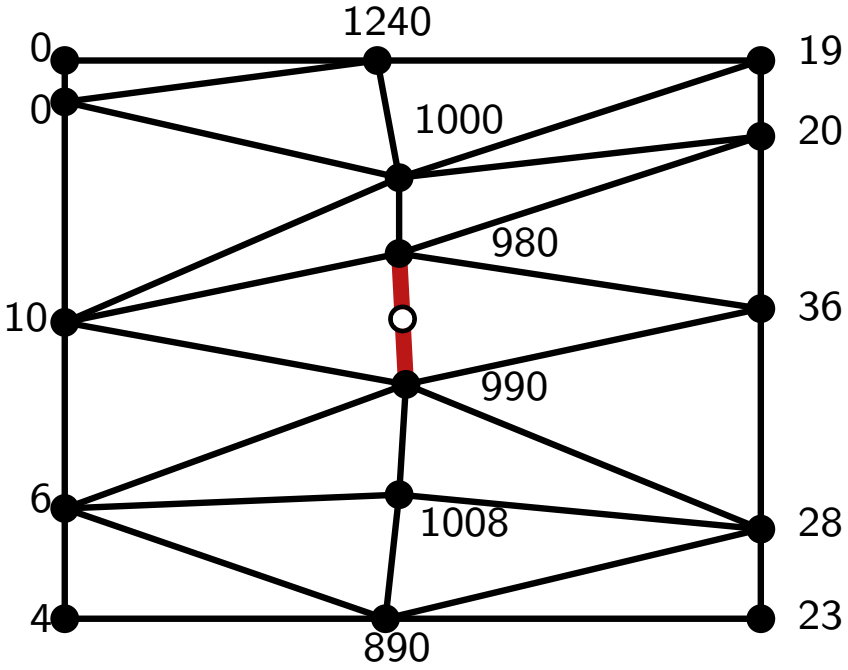


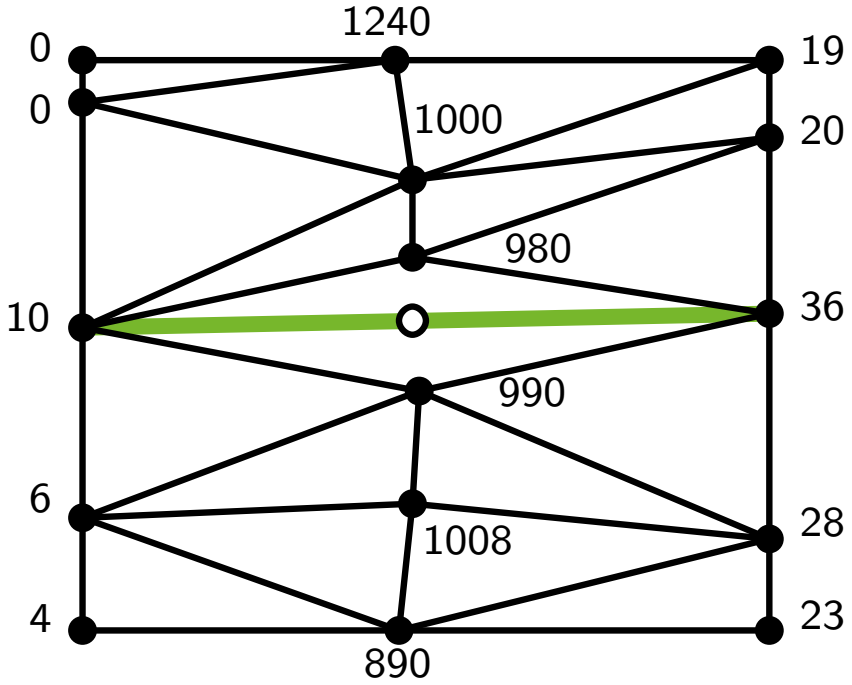
Zurück zur Höheninterpolation



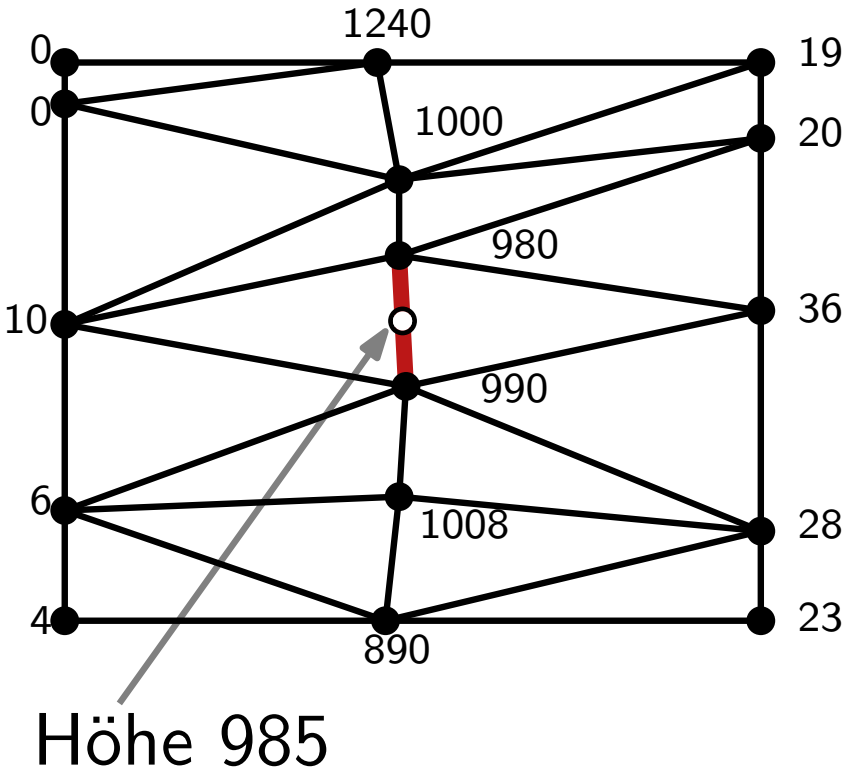


Zurück zur Höheninterpolation

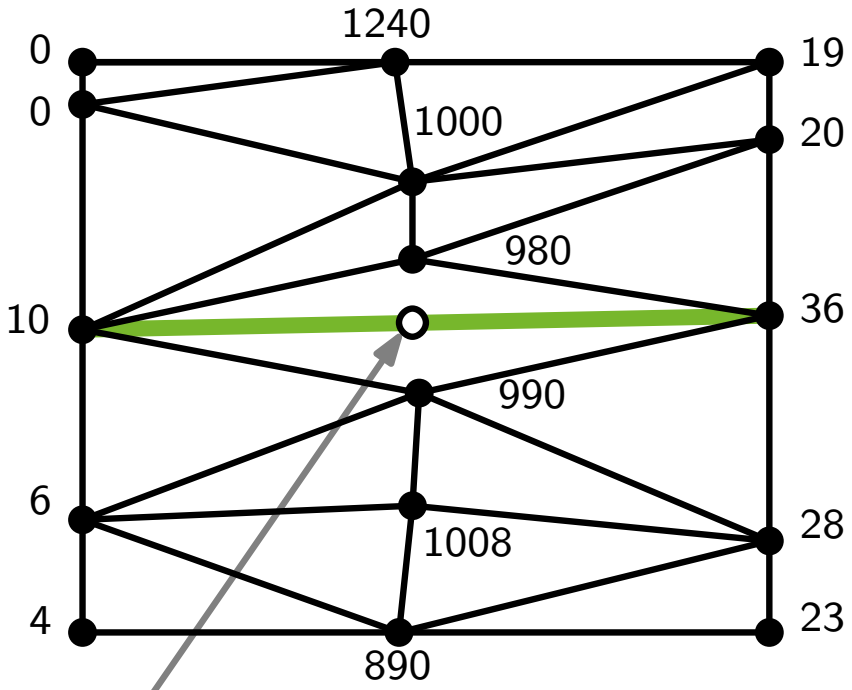




Zurück zur Höheninterpolation

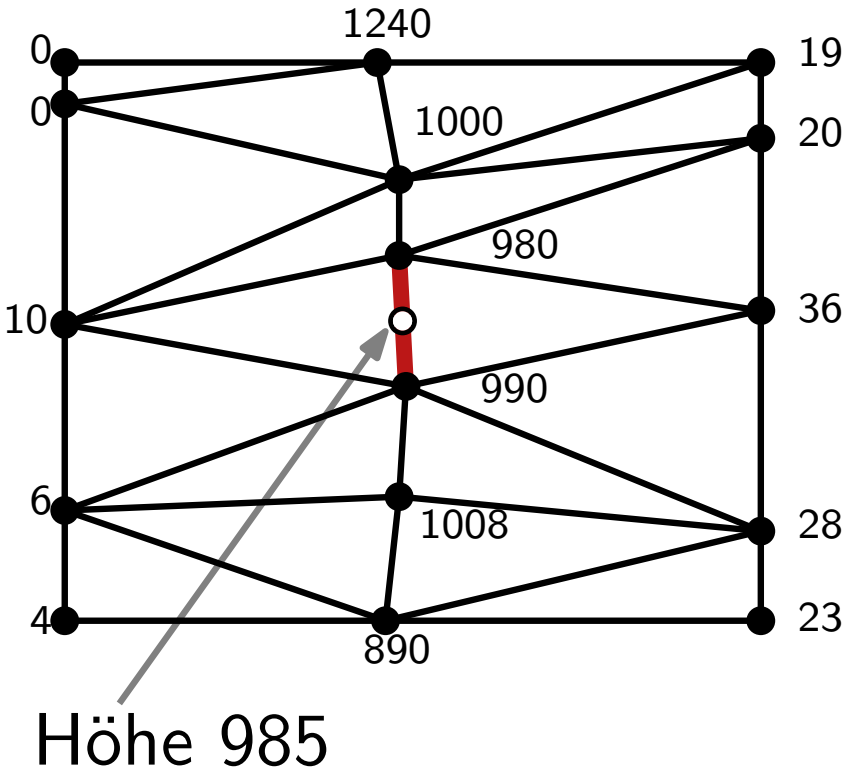


Höhe 985

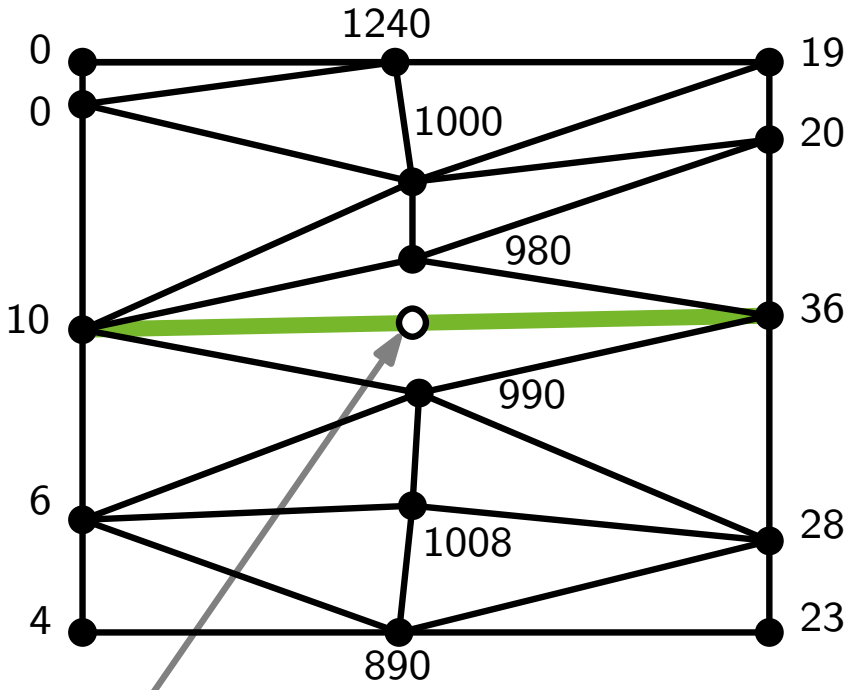


Höhe 23

Zurück zur Höheninterpolation



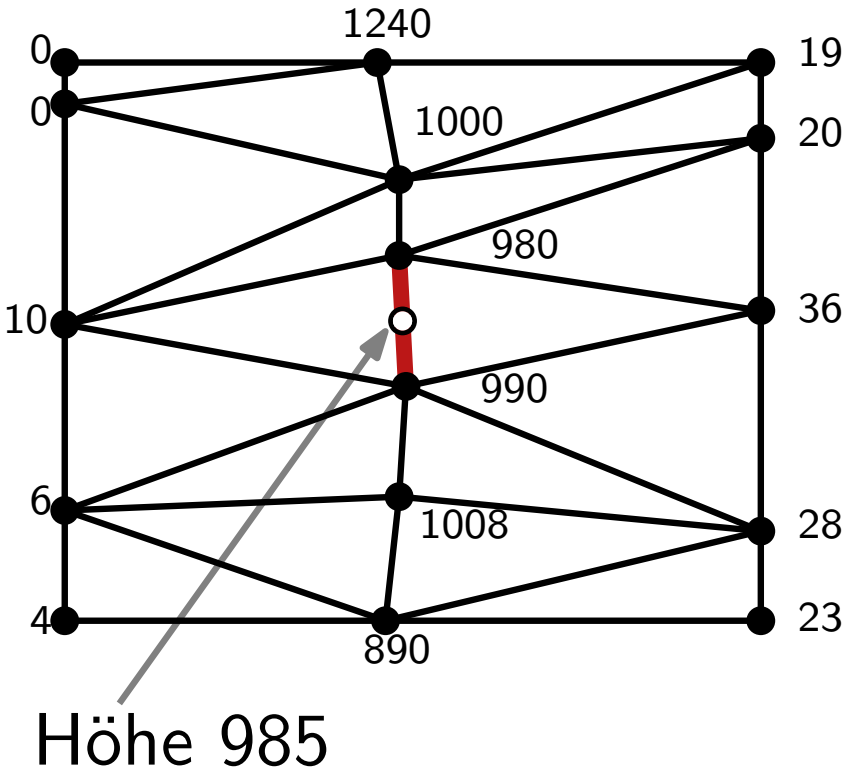
Höhe 985



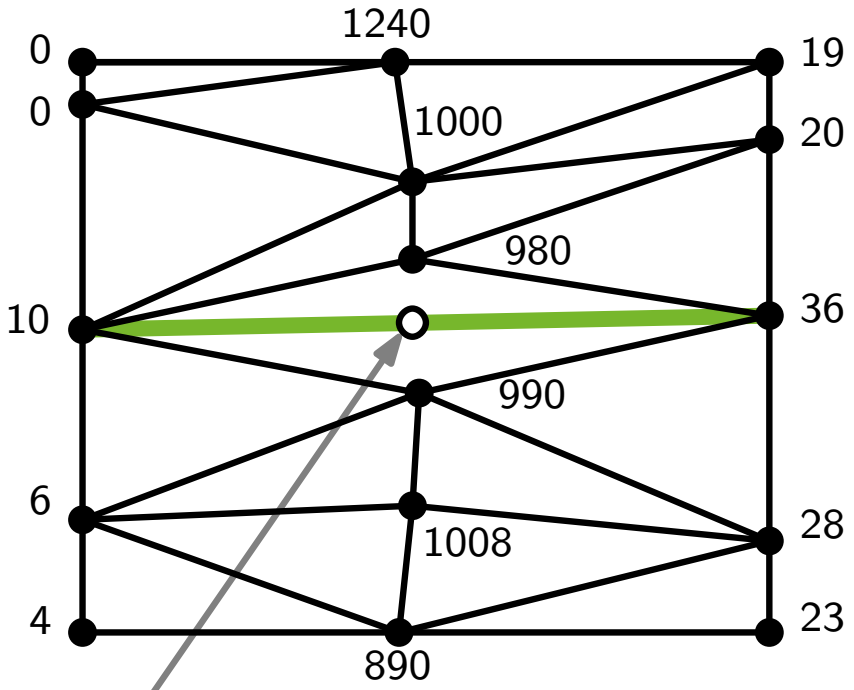
Höhe 23

Intuitiv: Vermeide zu schmale Dreiecke!

Zurück zur Höheninterpolation



Höhe 985



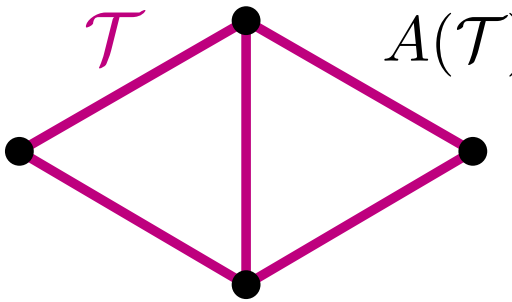
Höhe 23

Intuitiv: Vermeide zu schmale Dreiecke!

Oder: maximiere die kleinsten Dreieckswinkel!

Winkeloptimale Triangulierungen

Def.: Sei $P \subset \mathbb{R}^2$ eine Punktemenge, $\mathcal{T}$ eine Triangulierung von P und m die Anzahl der Dreiecke. Dann ist $A(\mathcal{T}) = (\alpha_1, \ldots, \alpha_{3m})$ der **Winkelvektor** von $\mathcal{T}$ mit den sortierten Dreieckswinkeln $\alpha_1 \leq \cdots \leq \alpha_{3m}$.

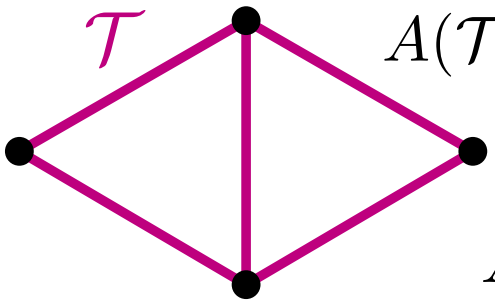


$$A(\mathcal{T}) = (60°, 60°, 60°, 60°, 60°, 60°)$$

Winkeloptimale Triangulierungen

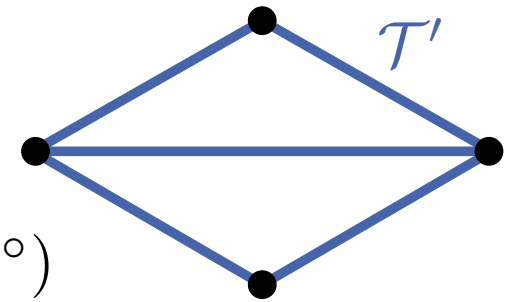
Def.: Sei $P \subset \mathbb{R}^2$ eine Punktemenge, $\mathcal{T}$ eine Triangulierung von P und m die Anzahl der Dreiecke. Dann ist $A(\mathcal{T}) = (\alpha_1, \ldots, \alpha_{3m})$ der **Winkelvektor** von $\mathcal{T}$ mit den sortierten Dreieckswinkeln $\alpha_1 \leq \cdots \leq \alpha_{3m}$.

Für zwei Triangulierungen $\mathcal{T}$ und $\mathcal{T}'$ von P definiere die Ordnung $A(\mathcal{T}) > A(\mathcal{T}')$ als lexikographische Ordnung.



$$A(\mathcal{T}) = (60°, 60°, 60°, 60°, 60°, 60°)$$

$$A(\mathcal{T}') = (30°, 30°, 30°, 30°, 120°, 120°)$$

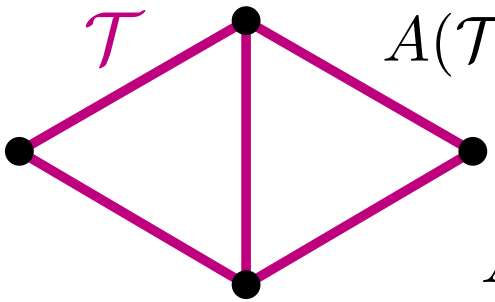


Winkeloptimale Triangulierungen

Def.: Sei $P \subset \mathbb{R}^2$ eine Punktemenge, $\mathcal{T}$ eine Triangulierung von P und m die Anzahl der Dreiecke. Dann ist $A(\mathcal{T}) = (\alpha_1, \ldots, \alpha_{3m})$ der **Winkelvektor** von $\mathcal{T}$ mit den sortierten Dreieckswinkeln $\alpha_1 \leq \cdots \leq \alpha_{3m}$.

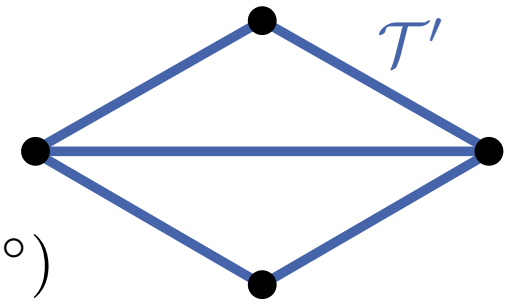
Für zwei Triangulierungen $\mathcal{T}$ und $\mathcal{T}'$ von P definiere die Ordnung $A(\mathcal{T}) > A(\mathcal{T}')$ als lexikographische Ordnung.

$\mathcal{T}$ heißt **winkeloptimal**, falls $A(\mathcal{T}) \geq A(\mathcal{T}')$ für alle Triangulierungen $\mathcal{T}'$ von P.



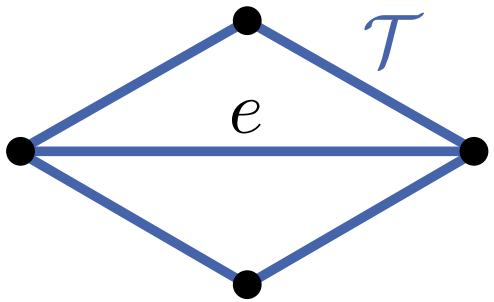
$$A(\mathcal{T}) = (60°, 60°, 60°, 60°, 60°, 60°)$$

$$A(\mathcal{T}') = (30°, 30°, 30°, 30°, 120°, 120°)$$



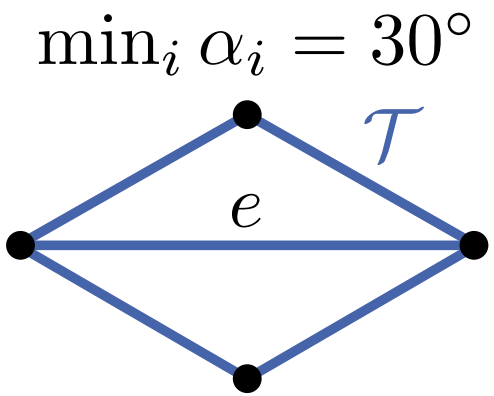
Kantenflips

Def.: Sei $\mathcal{T}$ eine Triangulierung. Eine Kante e von $\mathcal{T}$ heißt **unzulässig**, wenn der kleinste Winkel der zu e inzidenten Dreiecke durch einen Kantenflip größer wird.



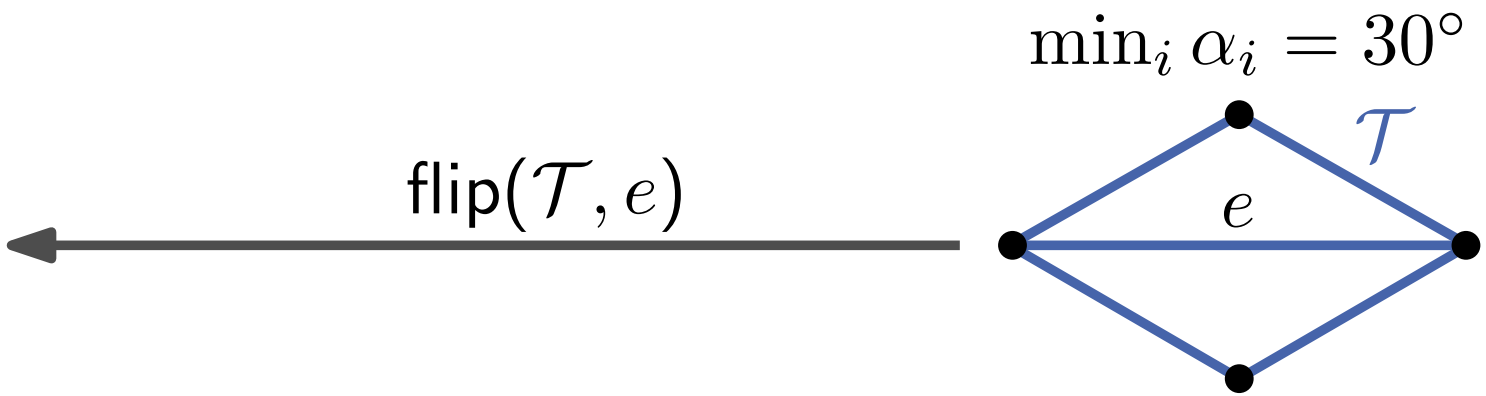
Kantenflips

Def.: Sei $\mathcal{T}$ eine Triangulierung. Eine Kante e von $\mathcal{T}$ heißt **unzulässig**, wenn der kleinste Winkel der zu e inzidenten Dreiecke durch einen Kantenflip größer wird.



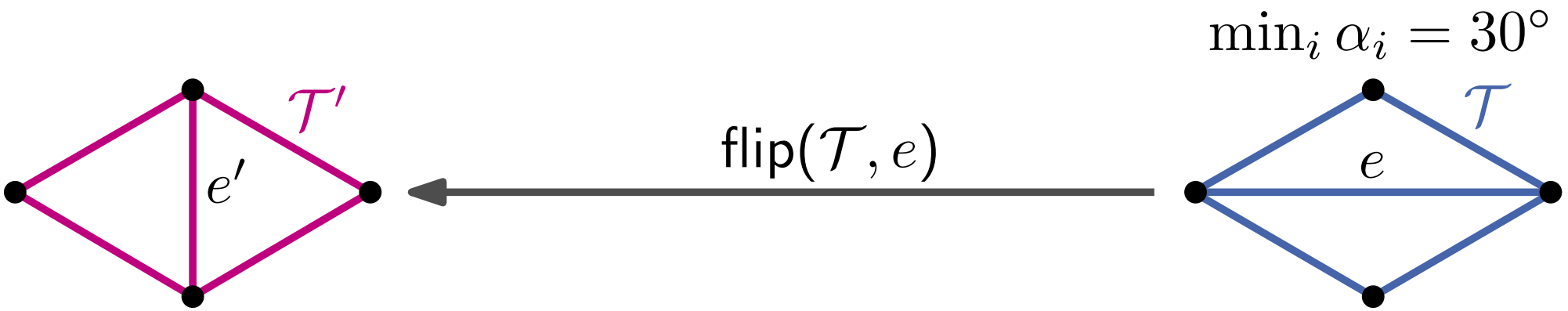
Kantenflips

Def.: Sei $\mathcal{T}$ eine Triangulierung. Eine Kante e von $\mathcal{T}$ heißt **unzulässig**, wenn der kleinste Winkel der zu e inzidenten Dreiecke durch einen Kantenflip größer wird.



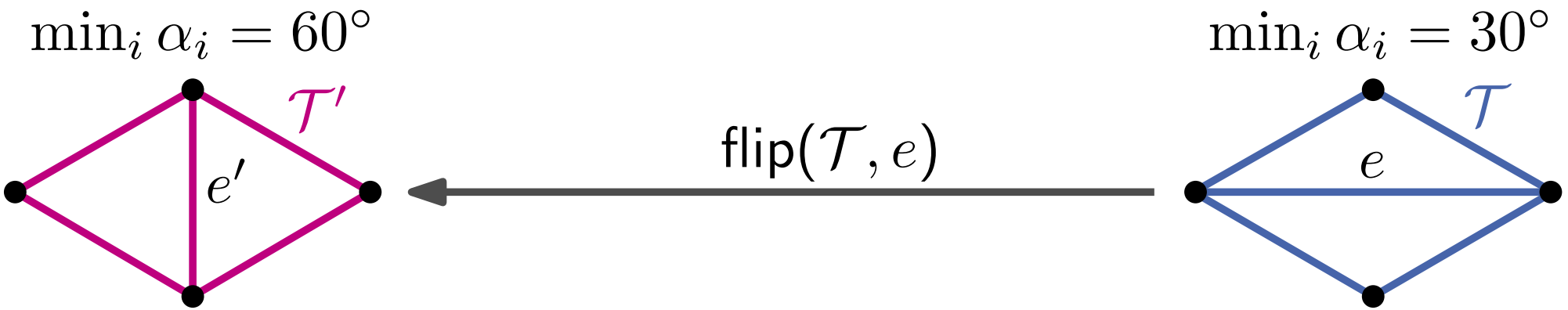
Kantenflips

Def.: Sei $\mathcal{T}$ eine Triangulierung. Eine Kante e von $\mathcal{T}$ heißt **unzulässig**, wenn der kleinste Winkel der zu e inzidenten Dreiecke durch einen Kantenflip größer wird.



Kantenflips

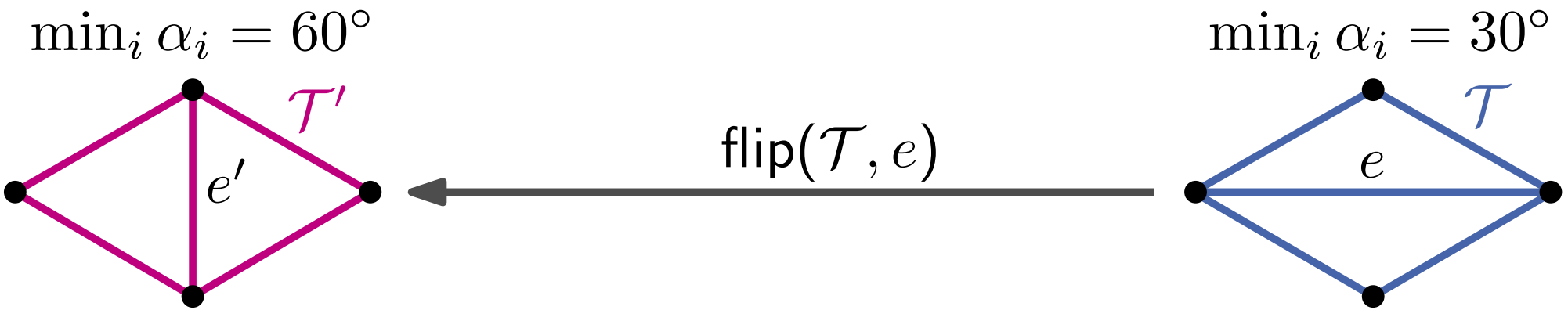
Def.: Sei $\mathcal{T}$ eine Triangulierung. Eine Kante e von $\mathcal{T}$ heißt **unzulässig**, wenn der kleinste Winkel der zu e inzidenten Dreiecke durch einen Kantenflip größer wird.



Kantenflips

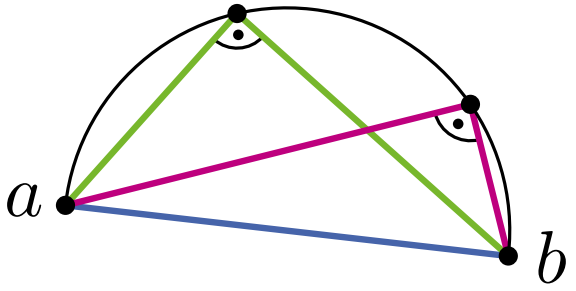
Def.: Sei $\mathcal{T}$ eine Triangulierung. Eine Kante e von $\mathcal{T}$ heißt **unzulässig**, wenn der kleinste Winkel der zu e inzidenten Dreiecke durch einen Kantenflip größer wird.

Beob.: Sei e eine unzulässige Kante von $\mathcal{T}$ und $\mathcal{T}' = \text{flip}(\mathcal{T}, e)$. Dann gilt $A(\mathcal{T}') > A(\mathcal{T})$.



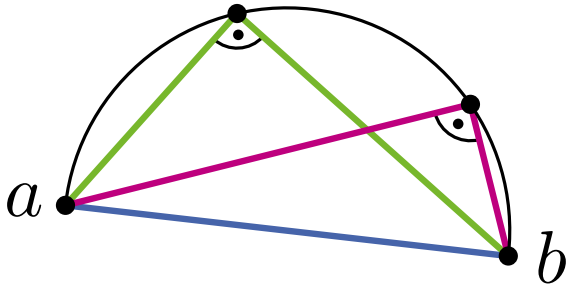
Der Satz von Thales

Satz 2: Alle Dreiecke aus den Endpunkten des Kreisdurchmessers und eines Halbkreispunktes sind rechtwinklig.

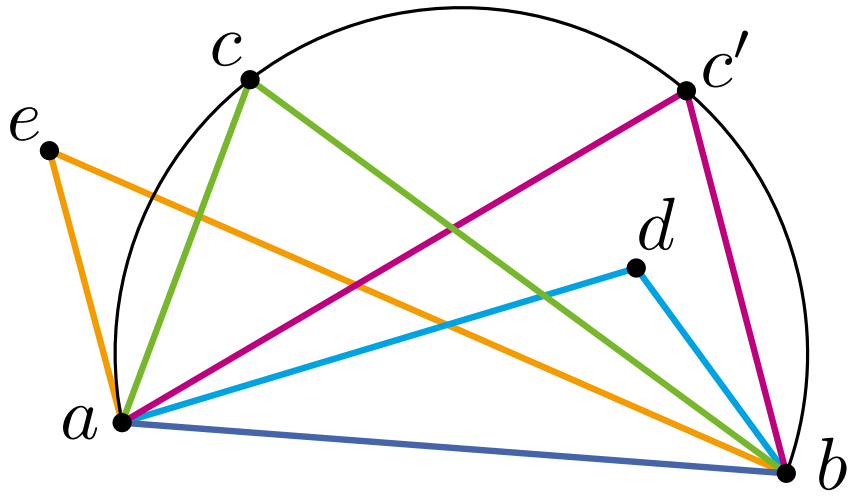


Der Satz von Thales

Satz 2: Alle Dreiecke aus den Endpunkten des Kreisdurchmessers und eines Halbkreispunktes sind rechtwinklig.



Satz 2′: Alle Dreiecke aus den Endpunkten einer Sekante $\ell = \overline{ab}$ und eines Kreispunktes c auf der gleichen Seite von ℓ haben den gleichen Winkel an c. Für Dreiecke Δabd mit d innerhalb des Kreises gilt $\angle adb > \angle acd$, für e außerhalb des Kreises gilt $\angle aeb < \angle acd$.

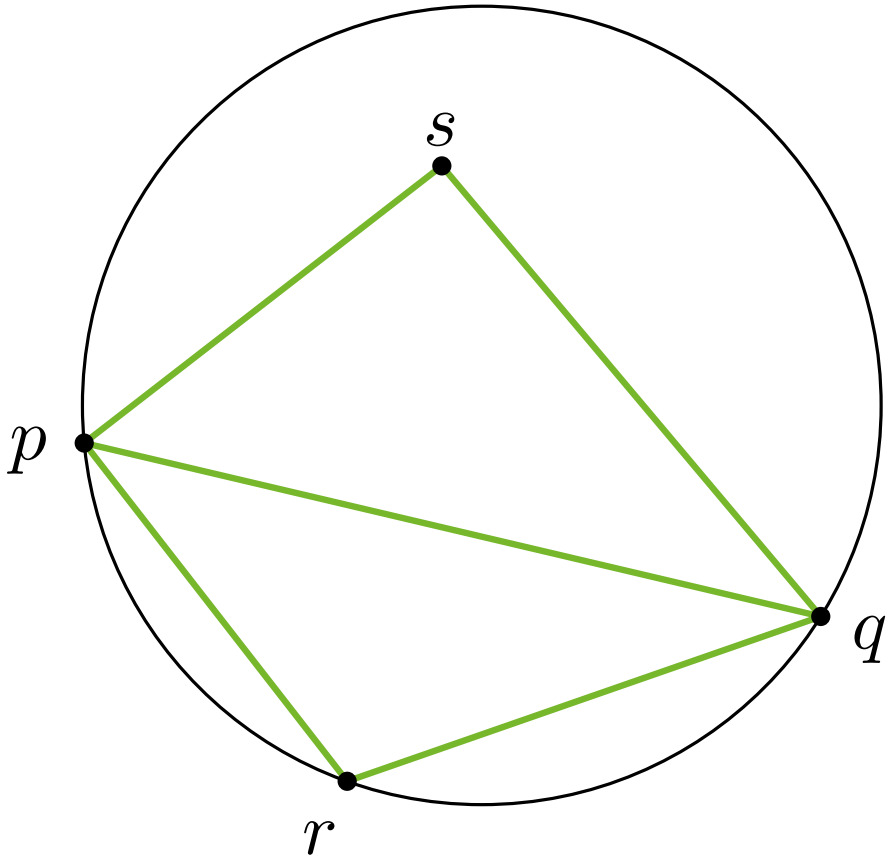


$$\angle aeb < \angle acb = \angle ac'b < \angle adb$$

Zulässige Triangulierungen

Lemma 1: Seien Δprq und Δpqs zwei Dreiecke in $\mathcal{T}$ und C der Umkreis von Δprq. Dann ist $\overline{pq}$ unzulässig genau dann wenn $s \in \text{int}(C)$.
Bilden p, q, r, s ein konvexes Viereck ($s \notin \partial C$) so ist entweder $\overline{pq}$ oder $\overline{rs}$ unzulässig.

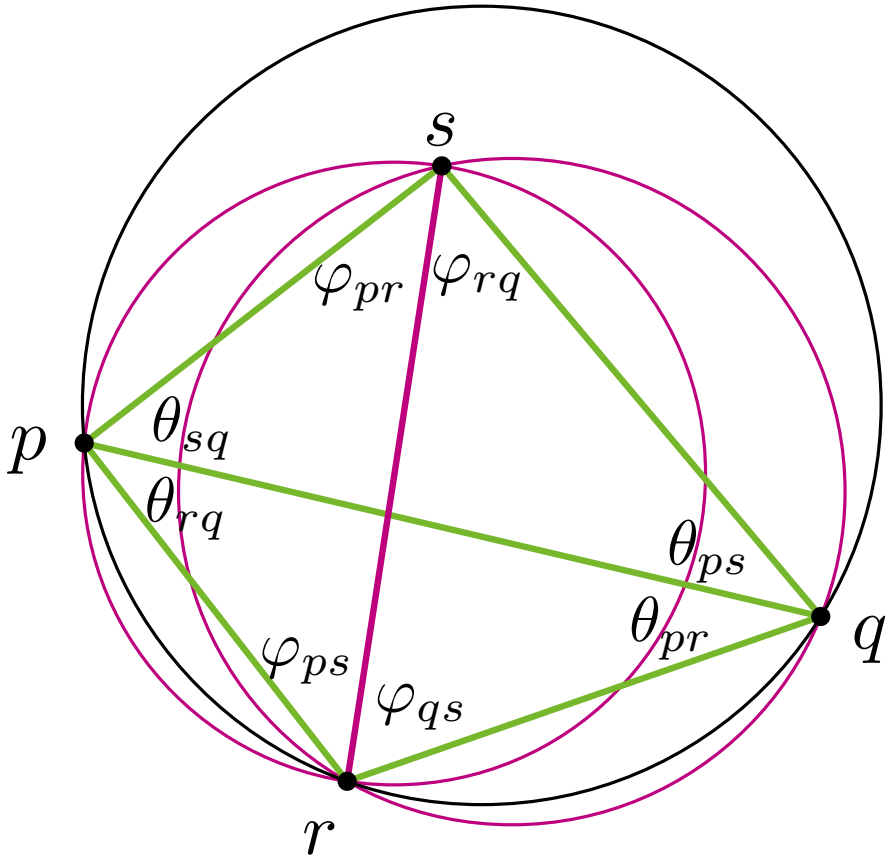
Beweisskizze:



Zulässige Triangulierungen

Lemma 1: Seien Δprq und Δpqs zwei Dreiecke in $\mathcal{T}$ und C der Umkreis von Δprq. Dann ist $\overline{pq}$ unzulässig genau dann wenn $s \in \text{int}(C)$.
Bilden p, q, r, s ein konvexes Viereck ($s \notin \partial C$) so ist entweder $\overline{pq}$ oder $\overline{rs}$ unzulässig.

Beweisskizze:

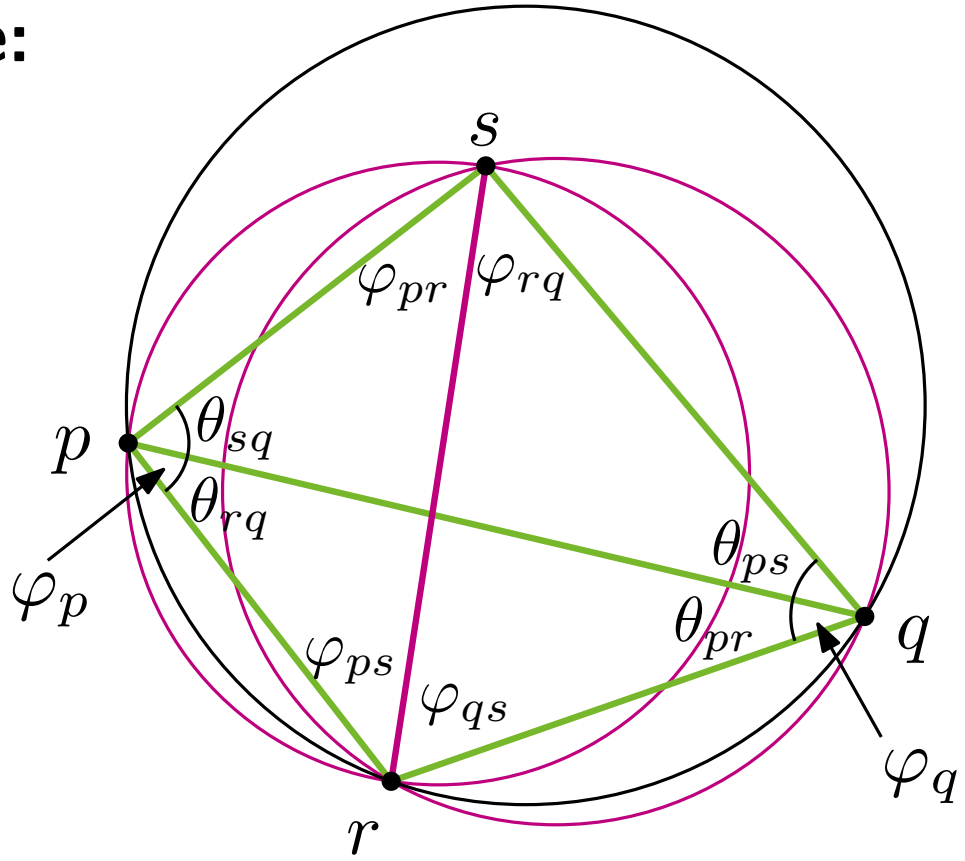


$$\varphi_{pr} > \theta_{pr}$$
$$\varphi_{ps} > \theta_{ps}$$
$$\varphi_{rq} > \theta_{rq}$$
$$\varphi_{qs} > \theta_{qs}$$

Zulässige Triangulierungen

Lemma 1: Seien Δprq und Δpqs zwei Dreiecke in $\mathcal{T}$ und C der Umkreis von Δprq. Dann ist $\overline{pq}$ unzulässig genau dann wenn $s \in \text{int}(C)$.
Bilden p, q, r, s ein konvexes Viereck ($s \notin \partial C$) so ist entweder $\overline{pq}$ oder $\overline{rs}$ unzulässig.

Beweisskizze:



$$\varphi_{pr} > \theta_{pr}$$
$$\varphi_{ps} > \theta_{ps}$$
$$\varphi_{rq} > \theta_{rq}$$
$$\varphi_{qs} > \theta_{qs}$$
$$\varphi_{p} > \theta_{rq}$$
$$\varphi_{q} > \theta_{pr}$$

Zulässige Triangulierungen

Lemma 1: Seien Δprq und Δpqs zwei Dreiecke in $\mathcal{T}$ und C der Umkreis von Δprq. Dann ist $\overline{pq}$ unzulässig genau dann wenn $s \in \text{int}(C)$.
Bilden p, q, r, s ein konvexes Viereck ($s \notin \partial C$) so ist entweder $\overline{pq}$ oder $\overline{rs}$ unzulässig.

Bem.:

- Charakterisierung symmetrisch bzgl. r und s
- $s \in \partial C \Rightarrow \overline{pq}$ und $\overline{rs}$ zulässig
- Kante unzulässig $\Rightarrow$ Viereck konvex

Zulässige Triangulierungen

Lemma 1: Seien Δprq und Δpqs zwei Dreiecke in $\mathcal{T}$ und C der Umkreis von Δprq. Dann ist $\overline{pq}$ unzulässig genau dann wenn $s \in \text{int}(C)$.
Bilden p, q, r, s ein konvexes Viereck ($s \notin \partial C$) so ist entweder $\overline{pq}$ oder $\overline{rs}$ unzulässig.

Bem.:
- Charakterisierung symmetrisch bzgl. r und s
- $s \in \partial C \Rightarrow \overline{pq}$ und $\overline{rs}$ zulässig
- Kante unzulässig $\Rightarrow$ Viereck konvex

Def.: Triangulierung ohne unzulässige Kanten heißt **zulässig**.

Zulässige Triangulierungen

Lemma 1: Seien Δprq und Δpqs zwei Dreiecke in $\mathcal{T}$ und C der Umkreis von Δprq. Dann ist $\overline{pq}$ unzulässig genau dann wenn $s \in \text{int}(C)$.
Bilden p, q, r, s ein konvexes Viereck ($s \notin \partial C$) so ist entweder $\overline{pq}$ oder $\overline{rs}$ unzulässig.

Bem.:
- Charakterisierung symmetrisch bzgl. r und s
- $s \in \partial C \Rightarrow \overline{pq}$ und $\overline{rs}$ zulässig
- Kante unzulässig $\Rightarrow$ Viereck konvex

Def.: Triangulierung ohne unzulässige Kanten heißt **zulässig**.

Gibt es immer eine zulässige Triangulierung?

Zulässige Triangulierungen

Lemma 1: Seien Δprq und Δpqs zwei Dreiecke in $\mathcal{T}$ und C der Umkreis von Δprq. Dann ist $\overline{pq}$ unzulässig genau dann wenn $s \in \text{int}(C)$.
Bilden p, q, r, s ein konvexes Viereck ($s \notin \partial C$) so ist entweder $\overline{pq}$ oder $\overline{rs}$ unzulässig.

Bem.:
- Charakterisierung symmetrisch bzgl. r und s
- $s \in \partial C \Rightarrow \overline{pq}$ und $\overline{rs}$ zulässig
- Kante unzulässig $\Rightarrow$ Viereck konvex

Def.: Triangulierung ohne unzulässige Kanten heißt **zulässig**.

while $\mathcal{T}$ hat unzulässige Kante e **do**
  flip$(\mathcal{T}, e)$
return $\mathcal{T}$

Zulässige Triangulierungen

Lemma 1: Seien Δprq und Δpqs zwei Dreiecke in $\mathcal{T}$ und C der Umkreis von Δprq. Dann ist $\overline{pq}$ unzulässig genau dann wenn $s \in \text{int}(C)$.
Bilden p, q, r, s ein konvexes Viereck ($s \notin \partial C$) so ist entweder $\overline{pq}$ oder $\overline{rs}$ unzulässig.

Bem.:
- Charakterisierung symmetrisch bzgl. r und s
- $s \in \partial C \Rightarrow \overline{pq}$ und $\overline{rs}$ zulässig
- Kante unzulässig $\Rightarrow$ Viereck konvex

Def.: Triangulierung ohne unzulässige Kanten heißt **zulässig**.

while $\mathcal{T}$ hat unzulässige Kante e **do**
  flip$(\mathcal{T}, e)$
return $\mathcal{T}$

terminiert, da $A(\mathcal{T})$ wächst und
#Triangulierungen endlich ($< 30^n$, [Sharir, Sheffer 2011])

Zwischenbilanz

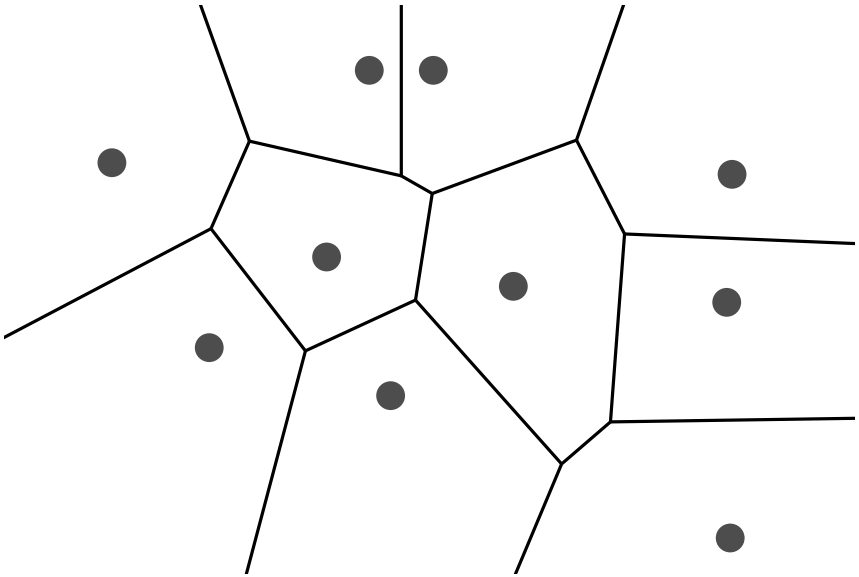
Es gilt: Jede winkeloptimale Triangulierung ist zulässig.

Aber ist jede zulässige Triangulierung auch winkeloptimal?

Die Delaunay-Triangulierung

Sei $\mathrm{Vor}(P)$ das Voronoi-Diagramm einer Punktmenge P.

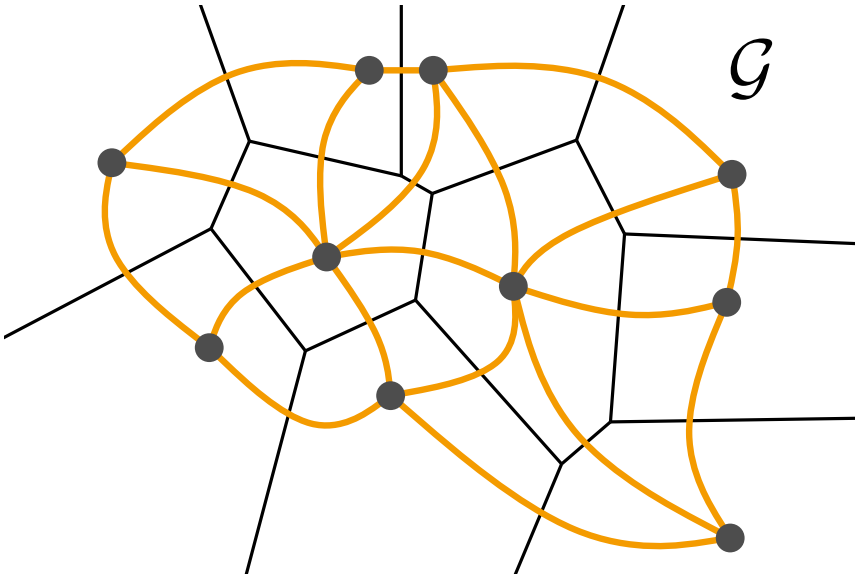
Def.: Der Graph $\mathcal{G} = (P, E)$ mit
$E = \{pq \mid \mathcal{V}(p) \text{ und } \mathcal{V}(q) \text{ sind benachbart}\}$
heißt **Dualgraph** von $\mathrm{Vor}(P)$.



Die Delaunay-Triangulierung

Sei $\mathrm{Vor}(P)$ das Voronoi-Diagramm einer Punktmenge P.

Def.: Der Graph $\mathcal{G} = (P, E)$ mit
$E = \{pq \mid \mathcal{V}(p) \text{ und } \mathcal{V}(q) \text{ sind benachbart}\}$
heißt **Dualgraph** von $\mathrm{Vor}(P)$.

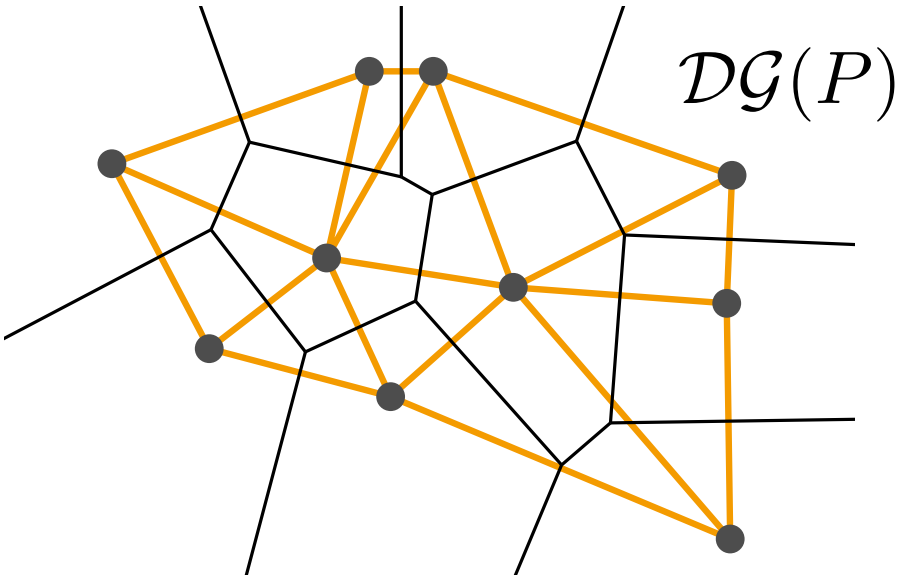


Die Delaunay-Triangulierung

Sei $\mathrm{Vor}(P)$ das Voronoi-Diagramm einer Punktmenge P.

Def.: Der Graph $\mathcal{G} = (P, E)$ mit
$E = \{pq \mid \mathcal{V}(p) \text{ und } \mathcal{V}(q) \text{ sind benachbart}\}$
heißt **Dualgraph** von $\mathrm{Vor}(P)$.

Def.: Die geradlinige Zeichnung von $\mathcal{G}$ heißt **Delaunay-Graph** $\mathcal{DG}(P)$.



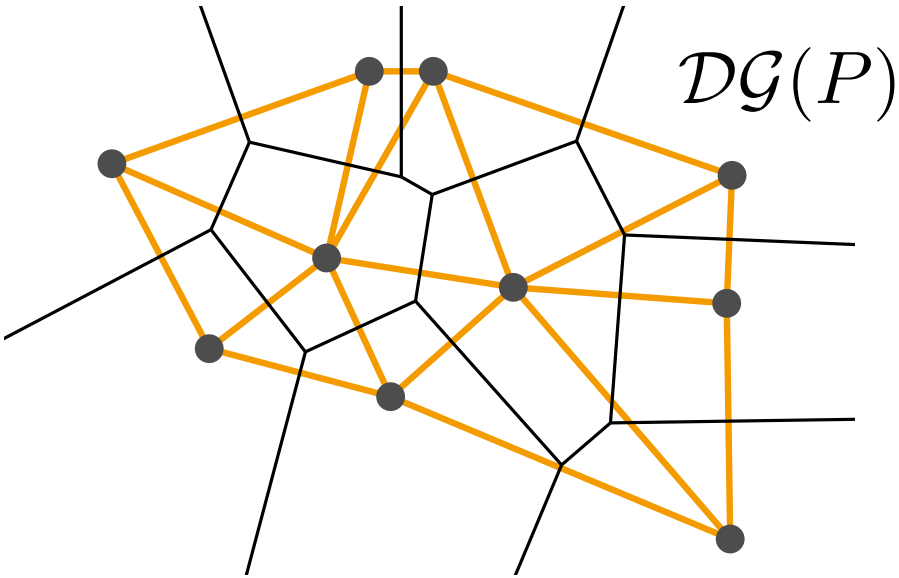
Die Delaunay-Triangulierung

Sei $\mathrm{Vor}(P)$ das Voronoi-Diagramm einer Punktmenge P.

Def.: Der Graph $\mathcal{G} = (P, E)$ mit
$E = \{pq \mid \mathcal{V}(p) \text{ und } \mathcal{V}(q) \text{ sind benachbart}\}$
heißt **Dualgraph** von $\mathrm{Vor}(P)$.

Def.: Die geradlinige Zeichnung von $\mathcal{G}$ heißt **Delaunay-Graph** $\mathcal{DG}(P)$.

Georgy Voronoy
(1868–1908)



Boris Delone
(1890–1980)

Eigenschaften

Satz 3: $\mathcal{DG}(P)$ ist kreuzungsfrei.

Eigenschaften

Satz 3: $\mathcal{DG}(P)$ ist kreuzungsfrei.

Beweisskizze:

Der Bisektor $b(p,q)$ definiert eine Voronoi-Kante
$\Leftrightarrow \exists r \in b(p,q)$ mit $C_P(r) \cap P = \{p,q\}$.

bzw.

Die Kante pq ist in $\mathcal{DG}(P)$
$\Leftrightarrow$ es gibt einen leeren Kreis $C_{p,q}$ mit p und q auf dem Rand.

Eigenschaften

Satz 3: $\mathcal{DG}(P)$ ist kreuzungsfrei.

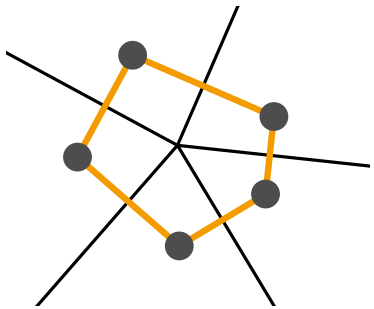
Beweisskizze:

Der Bisektor $b(p,q)$ definiert eine Voronoi-Kante
$\Leftrightarrow \exists r \in b(p,q)$ mit $C_P(r) \cap P = \{p,q\}$.

bzw.

Die Kante pq ist in $\mathcal{DG}(P)$
$\Leftrightarrow$ es gibt einen leeren Kreis $C_{p,q}$ mit p und q auf dem Rand.

Beob.: Ein Voronoi-Knoten v in $\mathrm{Vor}(P)$ mit Grad k entspricht einem konvexen k-Eck in $\mathcal{DG}(P)$.



Eigenschaften

Satz 3: $\mathcal{DG}(P)$ ist kreuzungsfrei.

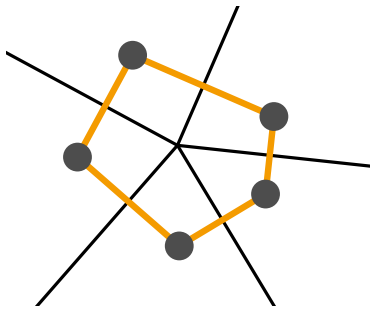
Beweisskizze:

Der Bisektor $b(p,q)$ definiert eine Voronoi-Kante
$\Leftrightarrow \exists r \in b(p,q)$ mit $C_P(r) \cap P = \{p,q\}$.

bzw.

Die Kante pq ist in $\mathcal{DG}(P)$
$\Leftrightarrow$ es gibt einen leeren Kreis $C_{p,q}$ mit p und q auf dem Rand.

Beob.: Ein Voronoi-Knoten v in $\mathrm{Vor}(P)$ mit Grad k entspricht einem konvexen k-Eck in $\mathcal{DG}(P)$.

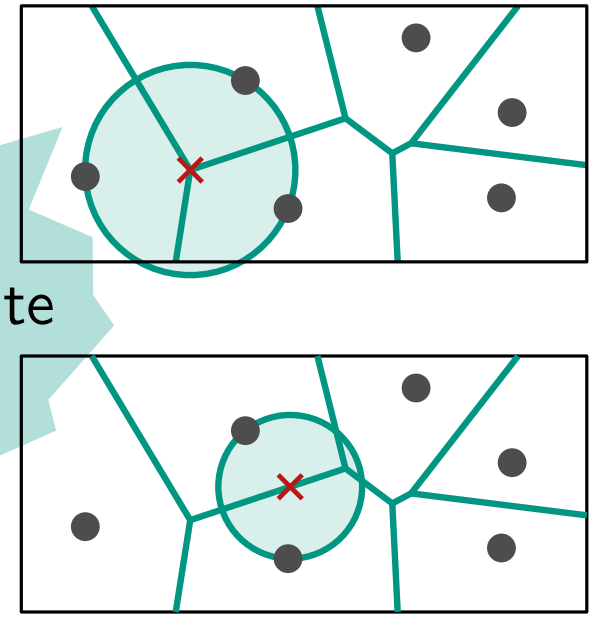


Ist P in allgemeiner Lage (keine 4 Punkte auf eine Kreis), so sind alle Facetten in $\mathcal{DG}(P)$ Dreiecke $\rightarrow$
Delaunay-Triangulierung

Charakterisierung

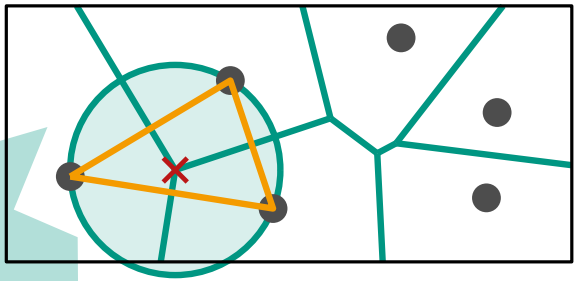
Satz über Voronoi-Diagramme:

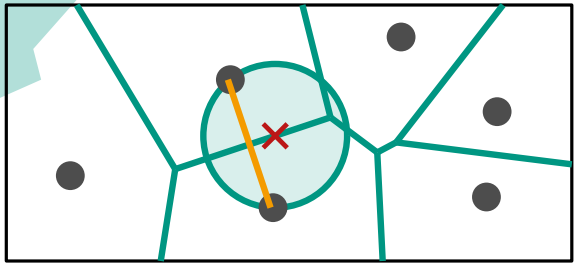
- Ein Punkt q ist ein Voronoi-Knoten
  $\Leftrightarrow |C_P(q) \cap P| \geq 3$,
- der Bisektor $b(p_i, p_j)$ definiert eine Voronoi-Kante
  $\Leftrightarrow \exists q \in b(p_i, p_j)$ mit $C_P(q) \cap P = \{p_i, p_j\}$.



Charakterisierung

Satz über Voronoi-Diagramme:





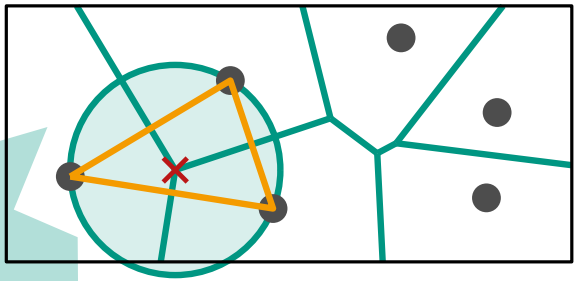
- Ein Punkt q ist ein Voronoi-Knoten
  $\Leftrightarrow |C_P(q) \cap P| \geq 3$,
- der Bisektor $b(p_i, p_j)$ definiert eine Voronoi-Kante
  $\Leftrightarrow \exists q \in b(p_i, p_j)$ mit $C_P(q) \cap P = \{p_i, p_j\}$.

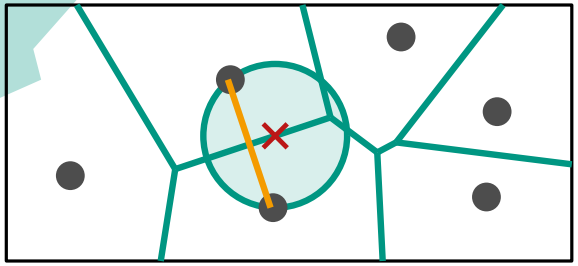
Satz 4: Sei P eine Menge von Punkten.

- Punkte p, q, r sind Knoten der gleichen Facette in $\mathcal{DG}(P) \Leftrightarrow$ Kreis durch p, q, r ist leer
- Kante pq ist in $\mathcal{DG}(P)$
  $\Leftrightarrow$ es gibt einen leeren Kreis $C_{p,q}$ durch p und q

Charakterisierung

Satz über Voronoi-Diagramme:





- Ein Punkt q ist ein Voronoi-Knoten
  $\Leftrightarrow |C_P(q) \cap P| \geq 3$,
- der Bisektor $b(p_i, p_j)$ definiert eine Voronoi-Kante
  $\Leftrightarrow \exists q \in b(p_i, p_j)$ mit $C_P(q) \cap P = \{p_i, p_j\}$.

Satz 4: Sei P eine Menge von Punkten.

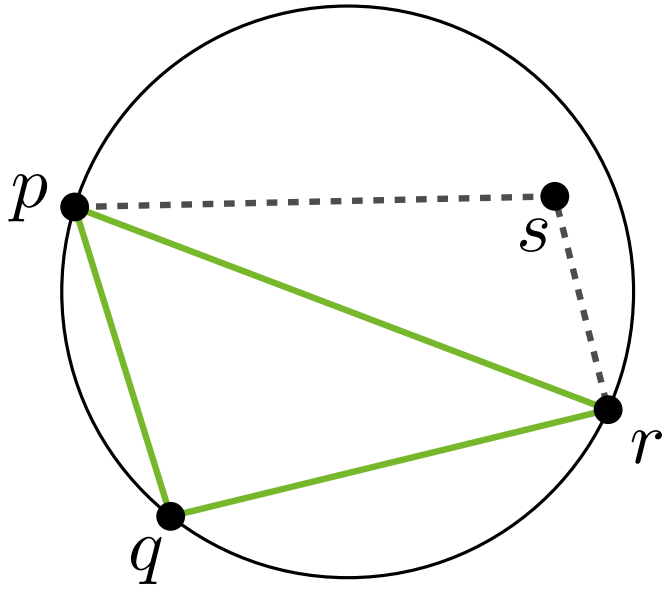
- Punkte p, q, r sind Knoten der gleichen Facette in $\mathcal{DG}(P)$ $\Leftrightarrow$ Kreis durch p, q, r ist leer
- Kante pq ist in $\mathcal{DG}(P)$
  $\Leftrightarrow$ es gibt einen leeren Kreis $C_{p,q}$ durch p und q

Satz 5: Sei P Punktmenge und $\mathcal{T}$ eine Triangulierung von P.
$\mathcal{T}$ ist Delaunay-Triangulierung
$\Leftrightarrow$ Umkreis jedes Dreiecks ist im Inneren leer.

Zulässigkeit und Delaunay-Triangulierungen

Satz 6: Sei P Punktmenge und $\mathcal{T}$ eine Triangulierung von P.
$\mathcal{T}$ ist zulässig $\Leftrightarrow$ $\mathcal{T}$ ist Delaunay-Triangulierung.

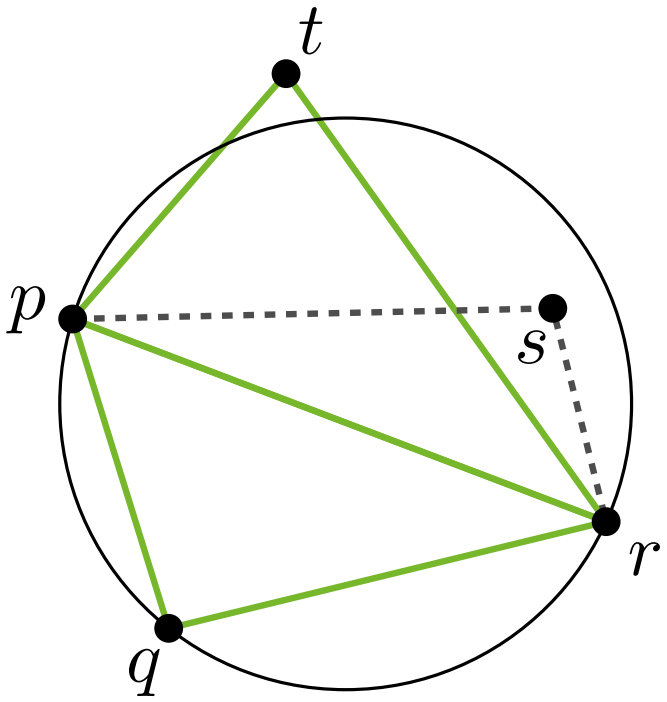
Beweisskizze:



Zulässigkeit und Delaunay-Triangulierungen

Satz 6: Sei P Punktmenge und $\mathcal{T}$ eine Triangulierung von P.
$\mathcal{T}$ ist zulässig $\Leftrightarrow$ $\mathcal{T}$ ist Delaunay-Triangulierung.

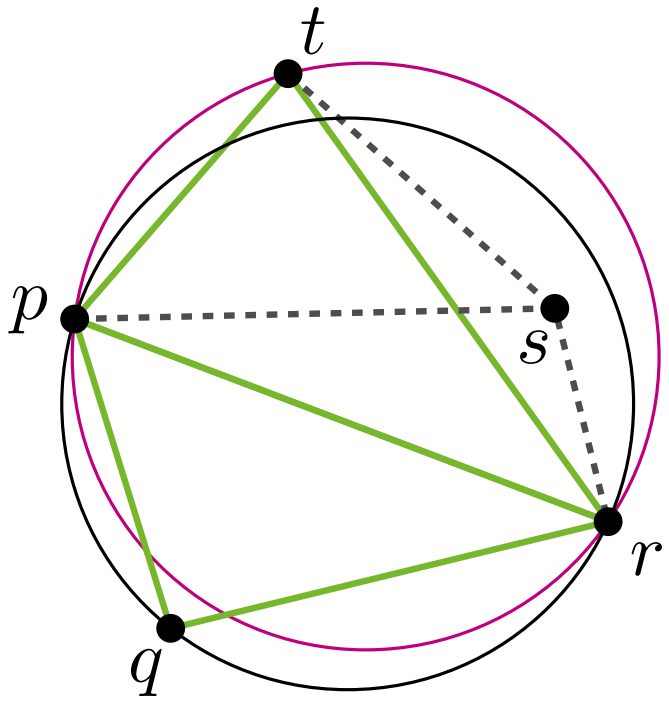
Beweisskizze:



Zulässigkeit und Delaunay-Triangulierungen

Satz 6: Sei P Punktmenge und $\mathcal{T}$ eine Triangulierung von P.
$\mathcal{T}$ ist zulässig $\Leftrightarrow$ $\mathcal{T}$ ist Delaunay-Triangulierung.

Beweisskizze:



Zulässigkeit und Delaunay-Triangulierungen

Satz 6: Sei P Punktmenge und $\mathcal{T}$ eine Triangulierung von P. $\mathcal{T}$ ist zulässig $\Leftrightarrow$ $\mathcal{T}$ ist Delaunay-Triangulierung.

Beob.: Ist P in allgemeiner Lage ist $\mathcal{DG}(P)$ eindeutig

Zulässigkeit und Delaunay-Triangulierungen

Karlsruhe Institute of Technology

Satz 6: Sei P Punktmenge und $\mathcal{T}$ eine Triangulierung von P.
$\mathcal{T}$ ist zulässig $\Leftrightarrow$ $\mathcal{T}$ ist Delaunay-Triangulierung.

Beob.: Ist P in allgemeiner Lage ist $\mathcal{DG}(P)$ eindeutig
$\Rightarrow$ zulässige Triangulierung ist eindeutig

Zulässigkeit und Delaunay-Triangulierungen

Satz 6: Sei P Punktmenge und $\mathcal{T}$ eine Triangulierung von P. $\mathcal{T}$ ist zulässig $\Leftrightarrow$ $\mathcal{T}$ ist Delaunay-Triangulierung.

Beob.: Ist P in allgemeiner Lage ist $\mathcal{DG}(P)$ eindeutig

$\Rightarrow$ zulässige Triangulierung ist eindeutig

wissen: $\mathcal{T}$ winkeloptimal $\Rightarrow$ $\mathcal{T}$ zulässig

Zulässigkeit und Delaunay-Triangulierungen

Satz 6: Sei P Punktmenge und $\mathcal{T}$ eine Triangulierung von P.
$\mathcal{T}$ ist zulässig $\Leftrightarrow$ $\mathcal{T}$ ist Delaunay-Triangulierung.

Beob.: Ist P in allgemeiner Lage ist $\mathcal{DG}(P)$ eindeutig

$\Rightarrow$ zulässige Triangulierung ist eindeutig

wissen: $\mathcal{T}$ winkeloptimal $\Rightarrow$ $\mathcal{T}$ zulässig

$\Rightarrow$ $\mathcal{DG}(P)$ winkeloptimal!

Zulässigkeit und Delaunay-Triangulierungen

Satz 6: Sei P Punktmenge und $\mathcal{T}$ eine Triangulierung von P. $\mathcal{T}$ ist zulässig $\Leftrightarrow$ $\mathcal{T}$ ist Delaunay-Triangulierung.

Beob.: Ist P in allgemeiner Lage ist $\mathcal{DG}(P)$ eindeutig

$\Rightarrow$ zulässige Triangulierung ist eindeutig

wissen: $\mathcal{T}$ winkeloptimal $\Rightarrow$ $\mathcal{T}$ zulässig

$\Rightarrow$ $\mathcal{DG}(P)$ winkeloptimal!

Ist P *nicht* in allgemeiner Lage, so ist zumindest für jede Triangulierung einer „großen“ Facette von $\mathcal{DG}(P)$ der minimale Winkel gleich (Übung!).

Zusammenfassung

Satz 7: Für n beliebige Punkte kann in $O(n \log n)$ Zeit eine Delaunay-Triangulierung berechnet werden.
(Voronoi-Diag. + Triangulierung „großer“ Facetten)

Zusammenfassung

Satz 7: Für n beliebige Punkte kann in $O(n \log n)$ Zeit eine Delaunay-Triangulierung berechnet werden.
(Voronoi-Diag. + Triangulierung „großer“ Facetten)

Korollar: Für n Punkte in allgemeiner Lage kann in $O(n \log n)$ Zeit eine winkeloptimale Triangulierung berechnet werden.
Sind die Punkte nicht in allgemeiner Lage, lässt sich zumindest eine Triangulierung mit maximalem kleinsten Winkel in $O(n \log n)$ Zeit berechnen.