Voronoi Diagrams

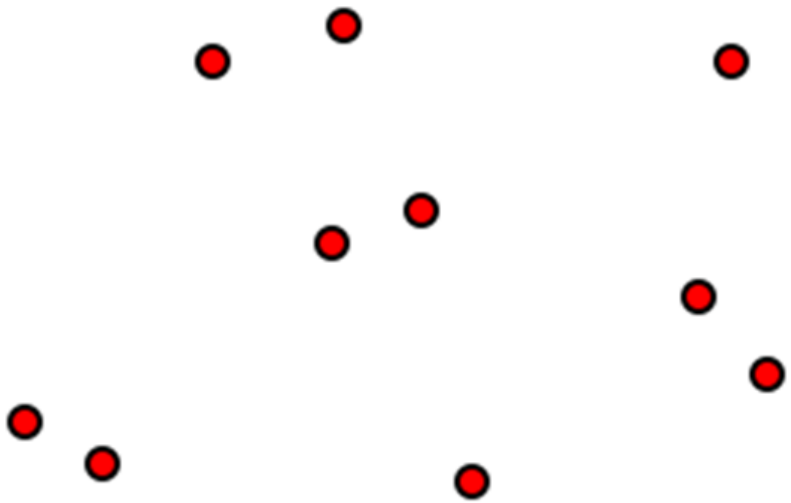
Christian Wellenbrock

December 1, 2009

Das Voronoi Diagramm

Problemstellung

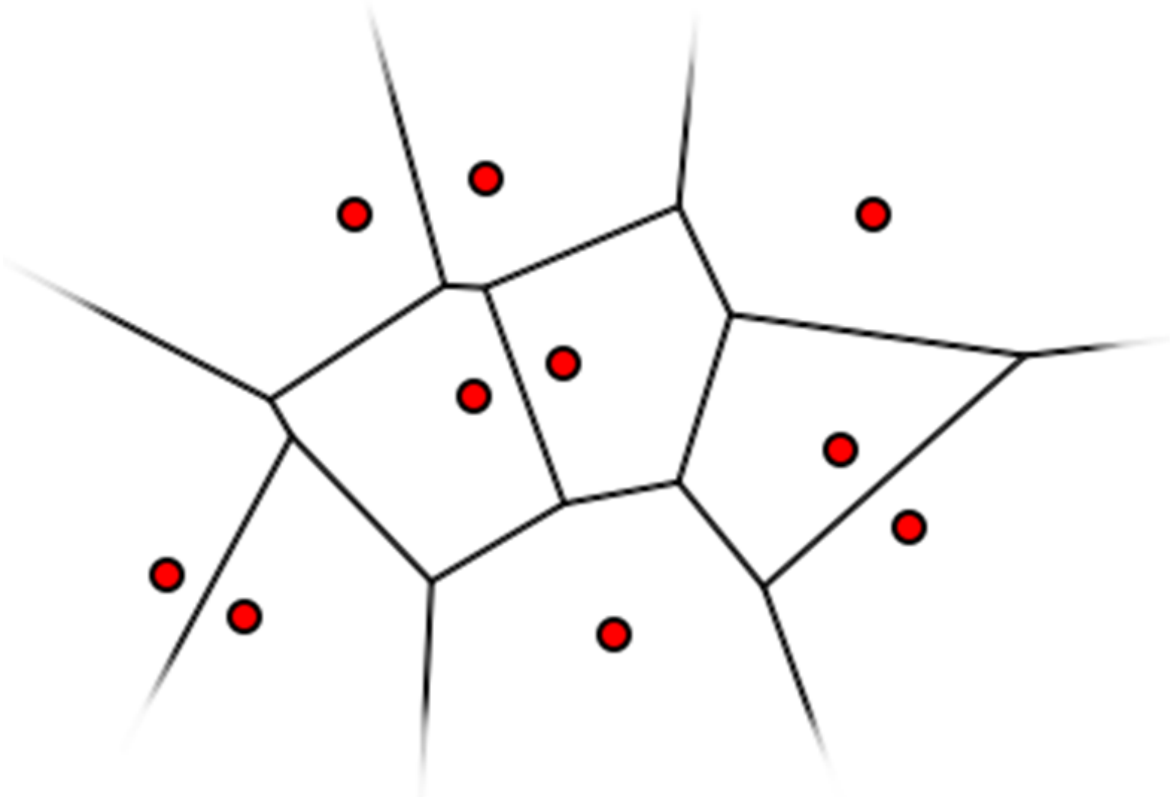
Gegeben: Menge der Zentren $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$



Das Voronoi Diagramm

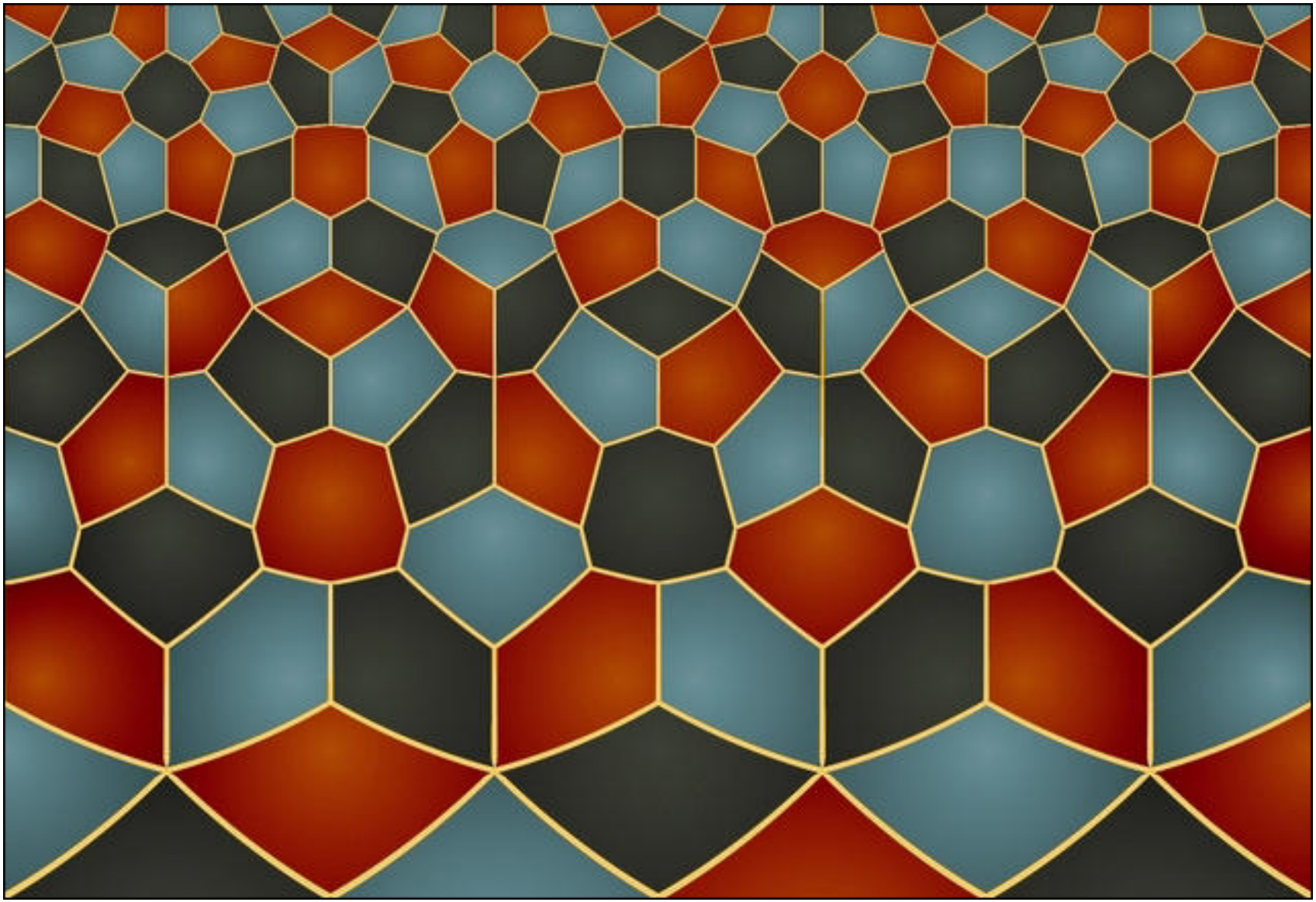
Problemstellung

Gegeben: Menge der Zentren $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$
Gesucht: Partitionierung $\mathrm{Vor}(P)$ von $\mathbb{R}^2$, die jedem $q \in \mathbb{R}^2$ das euklidisch nächste Zentrum p_i zuordnet

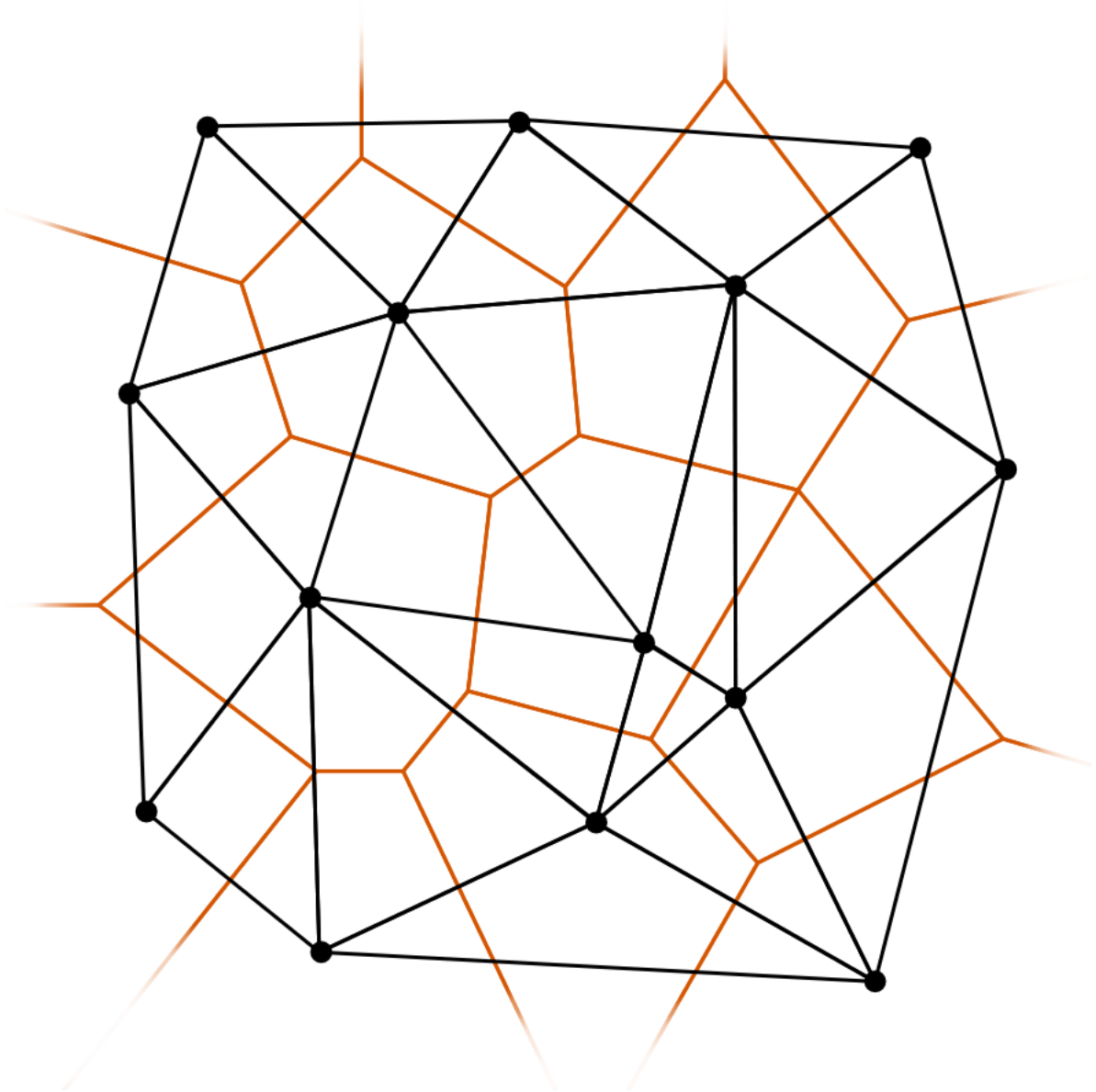


Warum Voronoi-Diagramme?

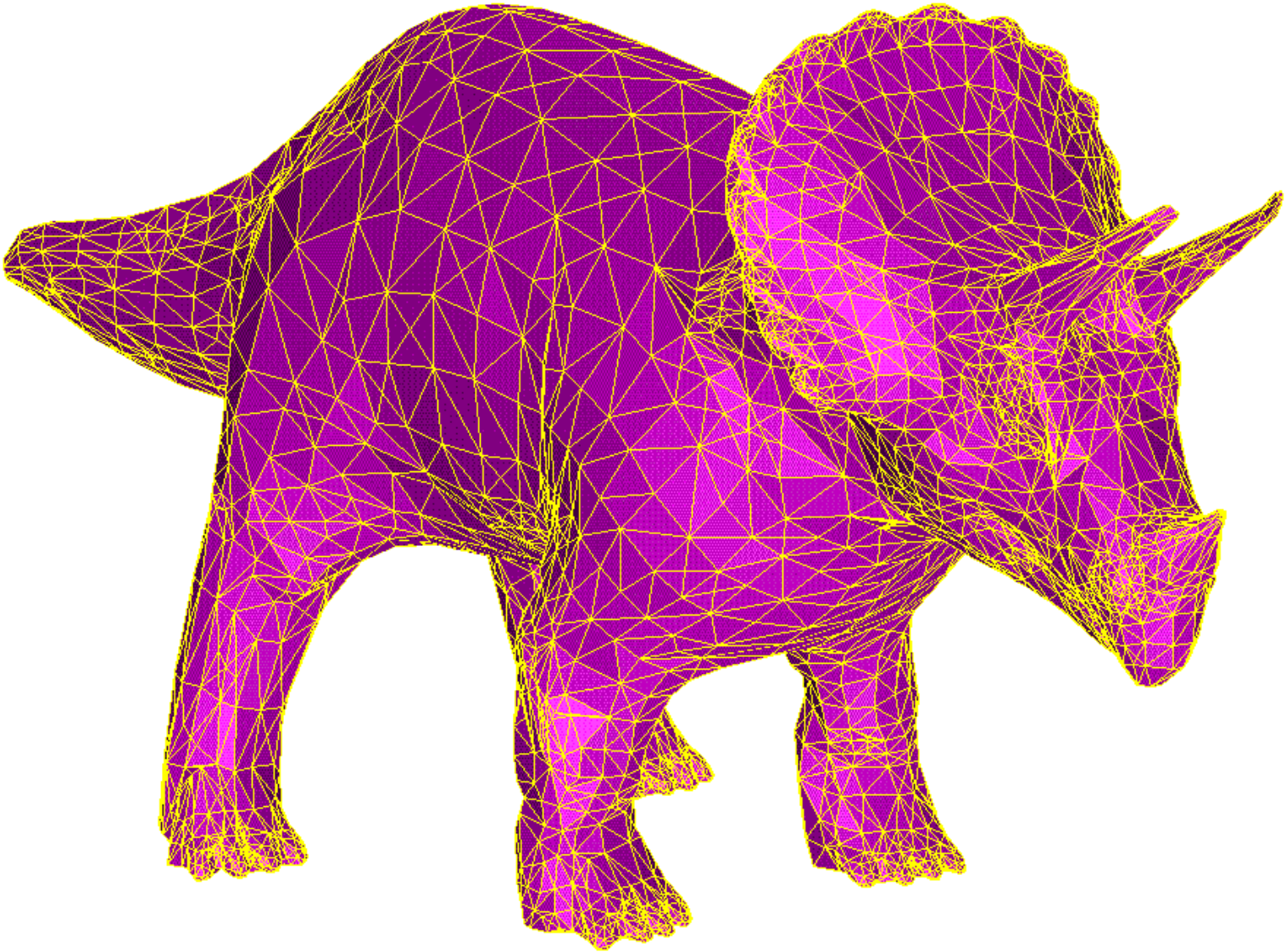
Warum Voronoi-Diagramme?



Warum Voronoi-Diagramme?



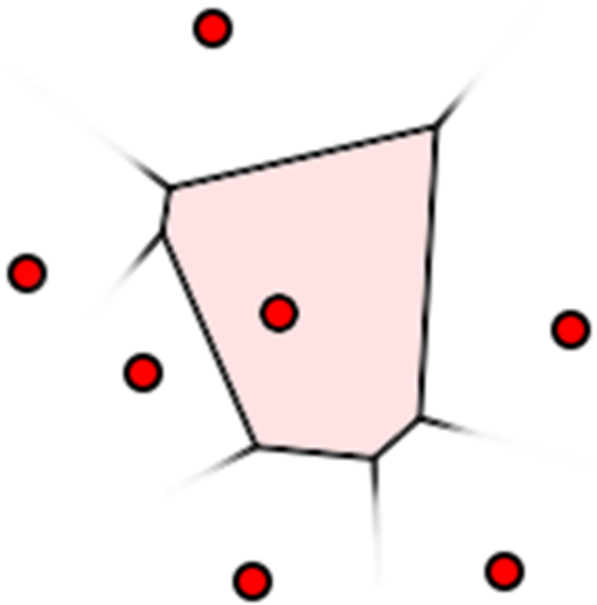
Warum Voronoi-Diagramme?



Definitionen

Voronoi-Region von $p \in P$:

$$V(p) = \{q \in \mathbb{R}^2 : \forall p' \in P \backslash \{p\} : \text{dist}(q, p) < \text{dist}(q, p')\}$$



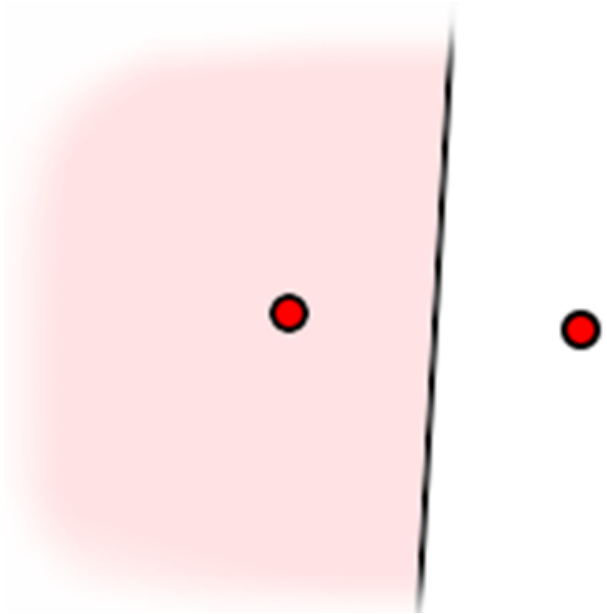
Definitionen

Voronoi-Region von $p \in P$:

$$V(p) = \{q \in \mathbb{R}^2 : \forall p' \in P \backslash \{p\} : \text{dist}(q, p) < \text{dist}(q, p')\}$$

Halbebene von $p \in P$ bezüglich $p' \in P$:

$$h(p, p') = \{q \in \mathbb{R}^2 : dist(q, p) < dist(q, p')\}$$



Einstieg

Definitionen

Voronoi-Region von $p \in P$:

$$V(p) = \{q \in \mathbb{R}^2 : \forall p' \in P \backslash \{p\} : \text{dist}(q, p) < \text{dist}(q, p')\}$$

Halbebene von $p \in P$ bezüglich $p' \in P$:

$$h(p, p') = \{q \in \mathbb{R}^2 : dist(q, p) < dist(q, p')\}$$

Folgerung

$$V(p) = \bigcap_{p' \in P \backslash \{p\}} h(p, p')$$

Struktur von Voronoi Diagrammen

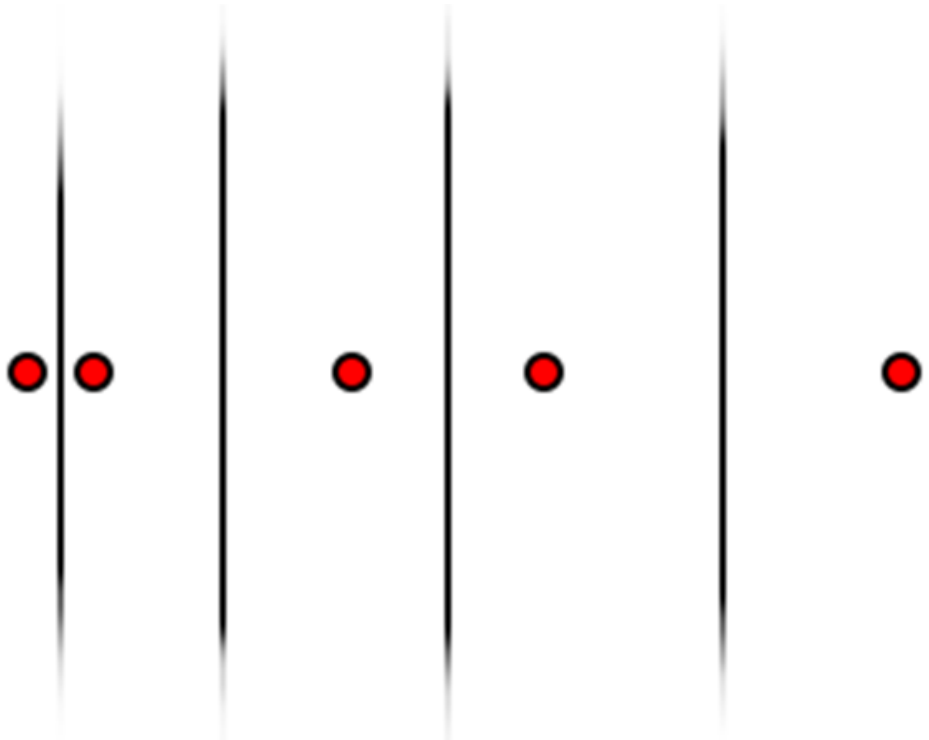
Satz

Ist P kollinear, so besteht $\mathrm{Vor}(P)$ aus $n-1$ parallelen Geraden. Andernfalls ist $\mathrm{Vor}(P)$ zusammenhängend und besteht aus Strecken und Geraden.

Struktur von Voronoi Diagrammen

Satz

Ist P kollinear, so besteht $\mathrm{Vor}(P)$ aus $n-1$ parallelen Geraden. Andernfalls ist $\mathrm{Vor}(P)$ zusammenhängend und besteht aus Strecken und Geraden.



Komplexität

Es gibt n Voronoi-Regionen mit jeweils höchstens $n-1$ Ecken und $n-1$ Kanten. Die Komplexität des Voronoi Diagramms liegt somit in $\mathcal{O}(n^2)$

Komplexität

Es gibt n Voronoi-Regionen mit jeweils höchstens $n-1$ Ecken und $n-1$ Kanten. Die Komplexität des Voronoi Diagramms liegt somit in $\mathcal{O}(n^2)$

Satz

Die Anzahl der Knoten und Kanten eines Voronoi Diagramms liegt in $\mathcal{O}(n)$.

Komplexität

Es gibt n Voronoi-Regionen mit jeweils höchstens $n-1$ Ecken und $n-1$ Kanten. Die Komplexität des Voronoi Diagramms liegt somit in $\mathcal{O}(n^2)$

Satz

Die Anzahl der Knoten und Kanten eines Voronoi Diagramms liegt in $\mathcal{O}(n)$.

Beweis

Klar falls P kollinear.

Komplexität

Es gibt n Voronoi-Regionen mit jeweils höchstens $n-1$ Ecken und $n-1$ Kanten. Die Komplexität des Voronoi Diagramms liegt somit in $\mathcal{O}(n^2)$

Satz

Die Anzahl der Knoten und Kanten eines Voronoi Diagramms liegt in $\mathcal{O}(n)$.

Beweis

Klar falls P kollinear. Ansonsten:

- Verbinde alle Halbgeraden mit neuem Punkt p_∞

Komplexität

Es gibt n Voronoi-Regionen mit jeweils höchstens $n-1$ Ecken und $n-1$ Kanten. Die Komplexität des Voronoi Diagramms liegt somit in $\mathcal{O}(n^2)$

Satz

Die Anzahl der Knoten und Kanten eines Voronoi Diagramms liegt in $\mathcal{O}(n)$.

Beweis

Klar falls P kollinear. Ansonsten:

- Verbinde alle Halbgeraden mit neuem Punkt p_∞
- Verwende Eulerschen Polyedersatz auf dem entstandenen Graphen: $(e+1)+n-k=2$

Komplexität

Es gibt n Voronoi-Regionen mit jeweils höchstens $n-1$ Ecken und $n-1$ Kanten. Die Komplexität des Voronoi Diagramms liegt somit in $\mathcal{O}(n^2)$

Satz

Die Anzahl der Knoten und Kanten eines Voronoi Diagramms liegt in $\mathcal{O}(n)$.

Beweis

Klar falls P kollinear. Ansonsten:

- Verbinde alle Halbgeraden mit neuem Punkt p_∞
- Verwende Eulerschen Polyedersatz auf dem entstandenen Graphen: $(e+1)+n-k=2$
- Summiere Grade aller Knoten: $2k \geq 3(e+1)$

Komplexität

Es gibt n Voronoi-Regionen mit jeweils höchstens $n-1$ Ecken und $n-1$ Kanten. Die Komplexität des Voronoi Diagramms liegt somit in $\mathcal{O}(n^2)$

Satz

Die Anzahl der Knoten und Kanten eines Voronoi Diagramms liegt in $\mathcal{O}(n)$.

Beweis

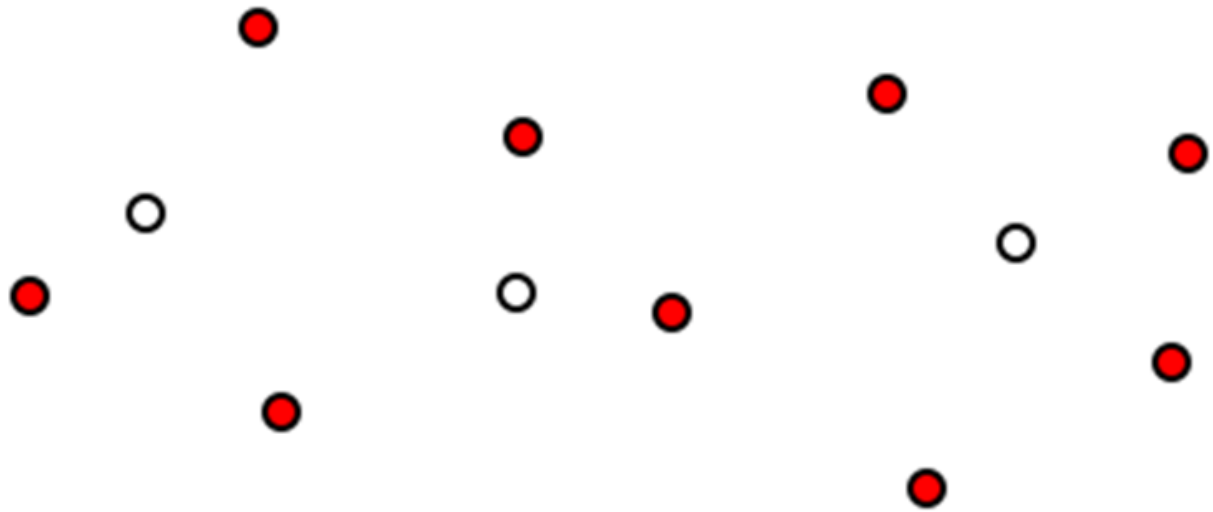
Klar falls P kollinear. Ansonsten:

- Verbinde alle Halbgeraden mit neuem Punkt p_∞
- Verwende Eulerschen Polyedersatz auf dem entstandenen Graphen: $(e+1)+n-k=2$
- Summiere Grade aller Knoten: $2k \geq 3(e+1)$
- Einsetzen liefert $e \leq 2n+5$ und $k \leq 3n-6$

Charakterisierung

Definition

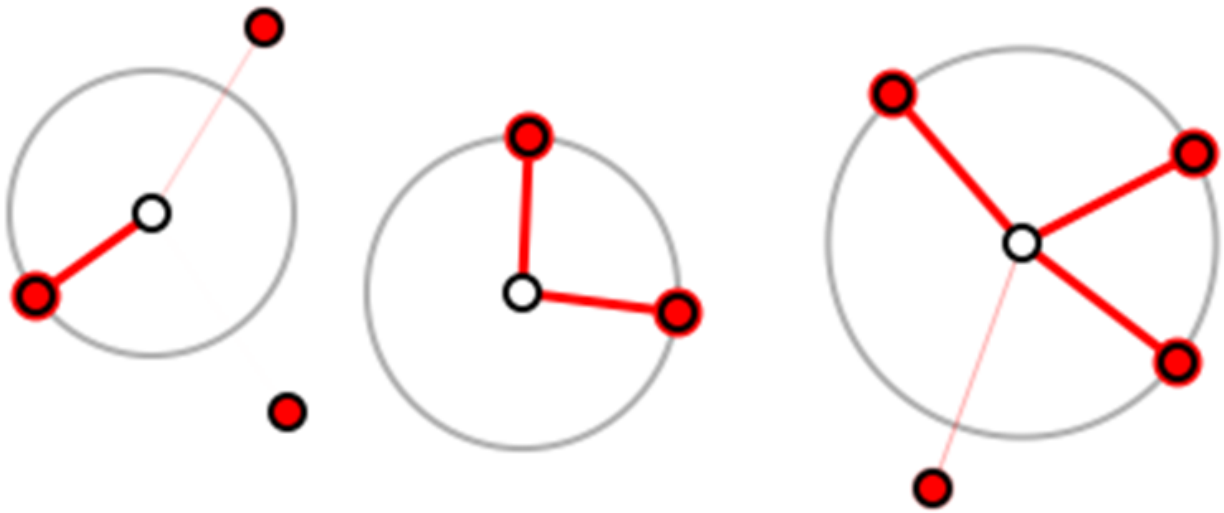
Zu $q \in \mathbb{R}^2$ sei $C_P(q)$ der größte offene Kreis um q, der kein $p \in P$ enthält.



Charakterisierung

Definition

Zu $q \in \mathbb{R}^2$ sei $C_P(q)$ der größte offene Kreis um q, der kein $p \in P$ enthält.



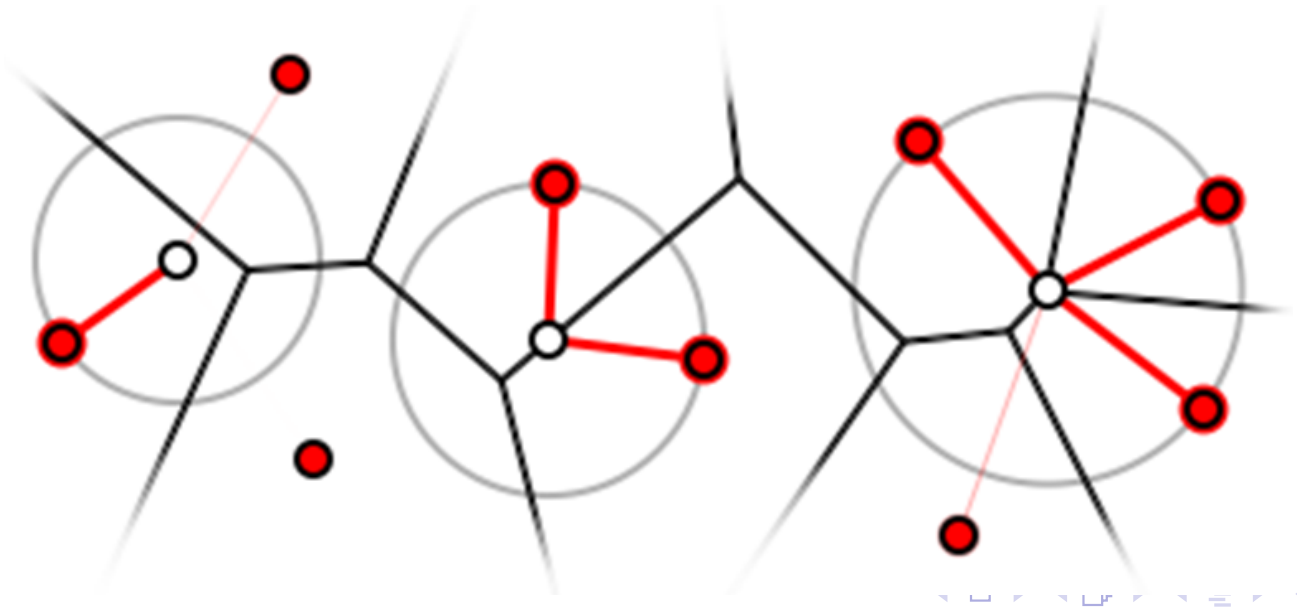
Charakterisierung

Definition

Zu $q \in \mathbb{R}^2$ sei $C_P(q)$ der größte offene Kreis um q, der kein $p \in P$ enthält.

Satz

- $q \in \mathbb{R}^2$ ist Knoten von $\mathrm{Vor}(P)$ $\iff$ $|P \cap \partial C_P(q)| \geq 3$



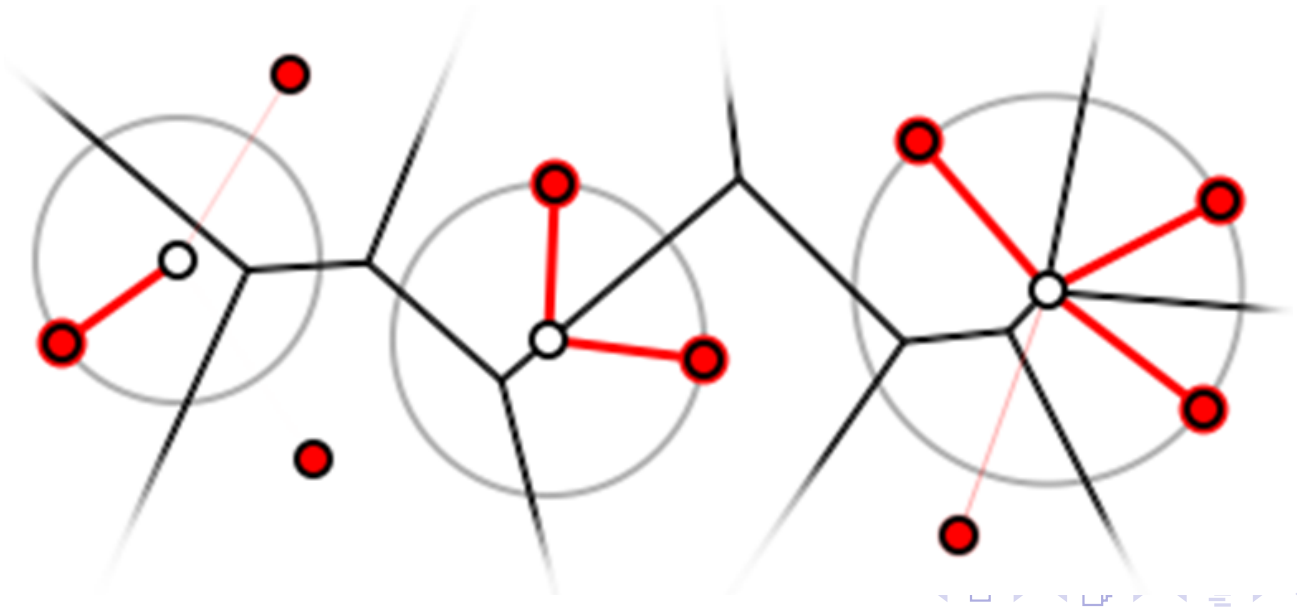
Charakterisierung

Definition

Zu $q \in \mathbb{R}^2$ sei $C_P(q)$ der größte offene Kreis um q, der kein $p \in P$ enthält.

Satz

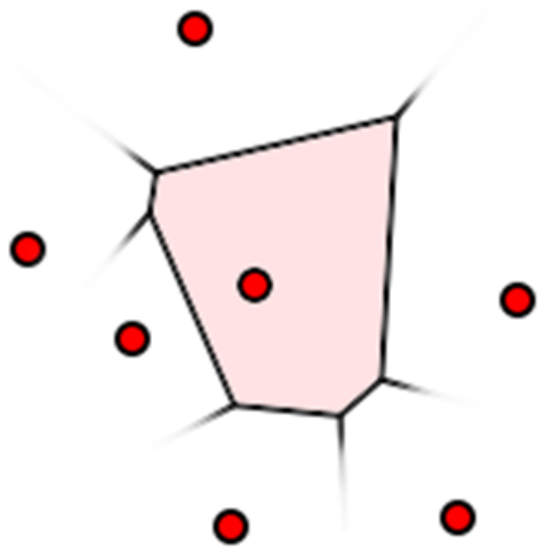
- $q \in \mathbb{R}^2$ ist Knoten von $\mathrm{Vor}(P) \iff |P \cap \partial C_P(q)| \geq 3$
- $q \in \mathbb{R}^2$ liegt auf Kante von $\mathrm{Vor}(P) \iff |P \cap \partial C_P(q)| = 2$



Berechnung des Voronoi-Diagramms

Naiver Ansatz

Berechne jede Voronoi-Region $V(p)$ als Schnitt der Halbebenen $h(p, p')$ für $p' \in P \backslash \{p\}$.



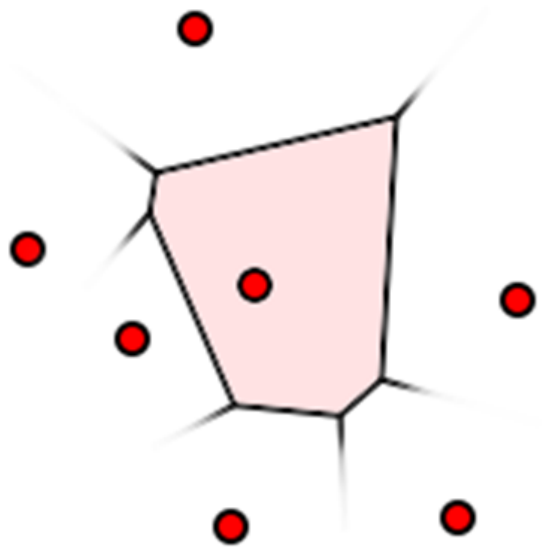
Berechnung des Voronoi-Diagramms

Naiver Ansatz

Berechne jede Voronoi-Region $V(p)$ als Schnitt der Halbebenen $h(p, p')$ für $p' \in P \backslash \{p\}$.

Laufzeit

- pro Region $\mathcal{O}(n \log n)$



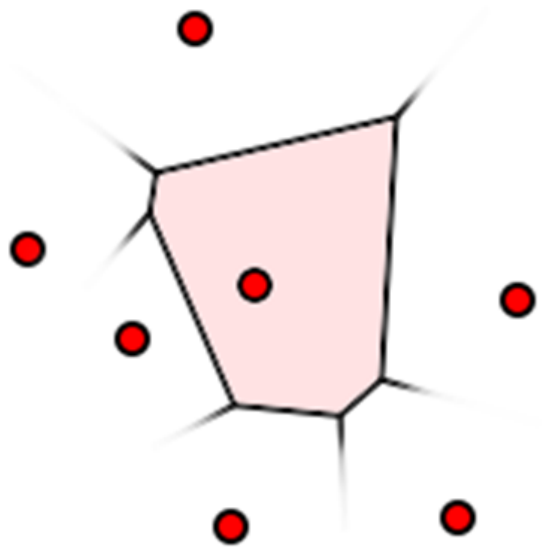
Berechnung des Voronoi-Diagramms

Naiver Ansatz

Berechne jede Voronoi-Region $V(p)$ als Schnitt der Halbebenen $h(p, p')$ für $p' \in P \backslash \{p\}$.

Laufzeit

- pro Region $\mathcal{O}(n \log n)$
- Gesamtlaufzeit in $\mathcal{O}(n^2 \log n)$



Fortune's Algorithm

Wir betrachten einen Sweep-Line-Algorithmus mit Laufzeit in $\mathcal{O}(n \log n)$.

Fortune's Algorithm

Wir betrachten einen Sweep-Line-Algorithmus mit Laufzeit in $\mathcal{O}(n \log n)$.

Sweep-Line Prinzip

- Schiebe vertikale Gerade ℓ von links nach rechts über Ebene

Fortune's Algorithm

Wir betrachten einen Sweep-Line-Algorithmus mit Laufzeit in $\mathcal{O}(n \log n)$.

Sweep-Line Prinzip

- Schiebe vertikale Gerade ℓ von links nach rechts über Ebene
- Konstruiere unterwegs die gesuchte Lösung

Fortune's Algorithm

Wir betrachten einen Sweep-Line-Algorithmus mit Laufzeit in $\mathcal{O}(n \log n)$.

Sweep-Line Prinzip

- Schiebe vertikale Gerade ℓ von links nach rechts über Ebene
- Konstruiere unterwegs die gesuchte Lösung
- Betrachte dazu den Schnitt der Geraden ℓ mit der Lösung

Fortune's Algorithm

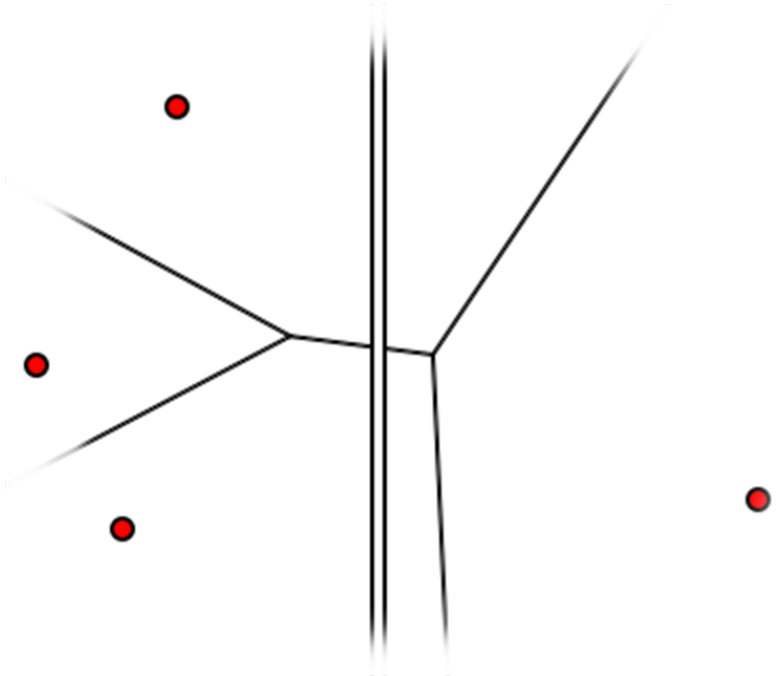
Wir betrachten einen Sweep-Line-Algorithmus mit Laufzeit in $\mathcal{O}(n \log n)$.

Sweep-Line Prinzip

- Schiebe vertikale Gerade ℓ von links nach rechts über Ebene
- Konstruiere unterwegs die gesuchte Lösung
- Betrachte dazu den Schnitt der Geraden ℓ mit der Lösung

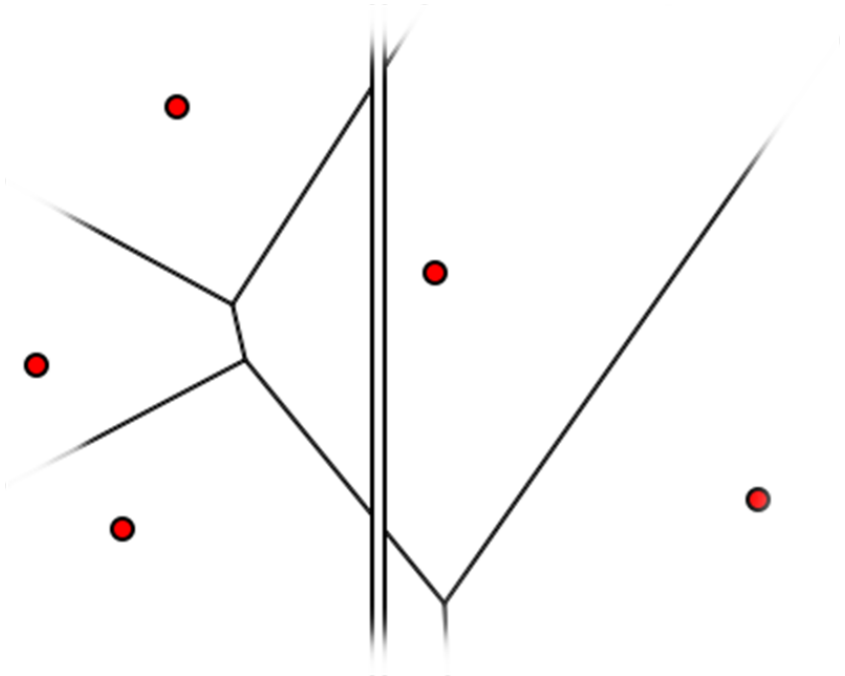
Dies ist hier **nicht** möglich: Das Voronoi Diagramm links von ℓ hängt auch von Zentren rechts von ℓ ab!

Fortune's Algorithm



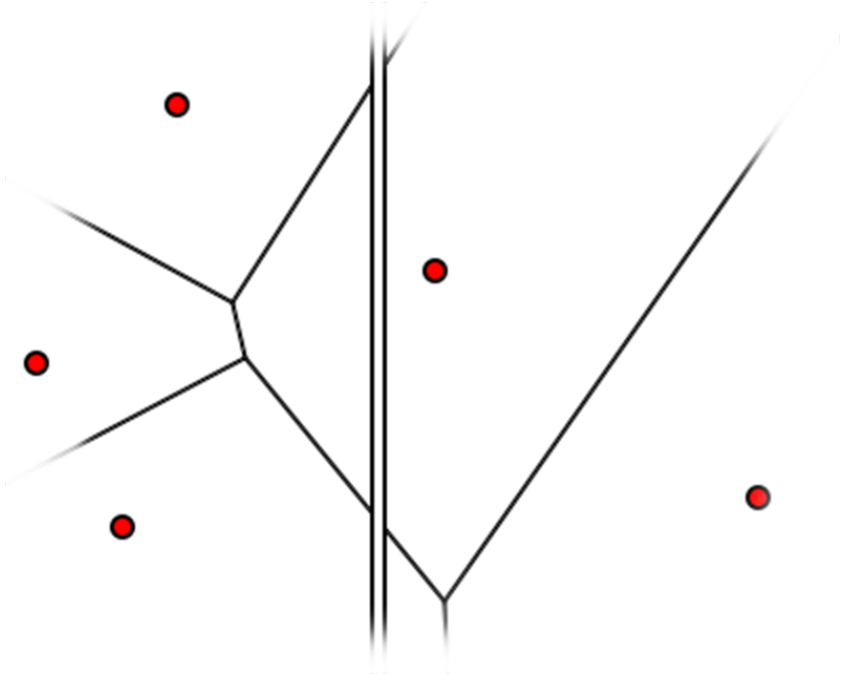
Dies ist hier **nicht** möglich: Das Voronoi Diagramm links von ℓ hängt auch von Zentren rechts von ℓ ab!

Fortune's Algorithm



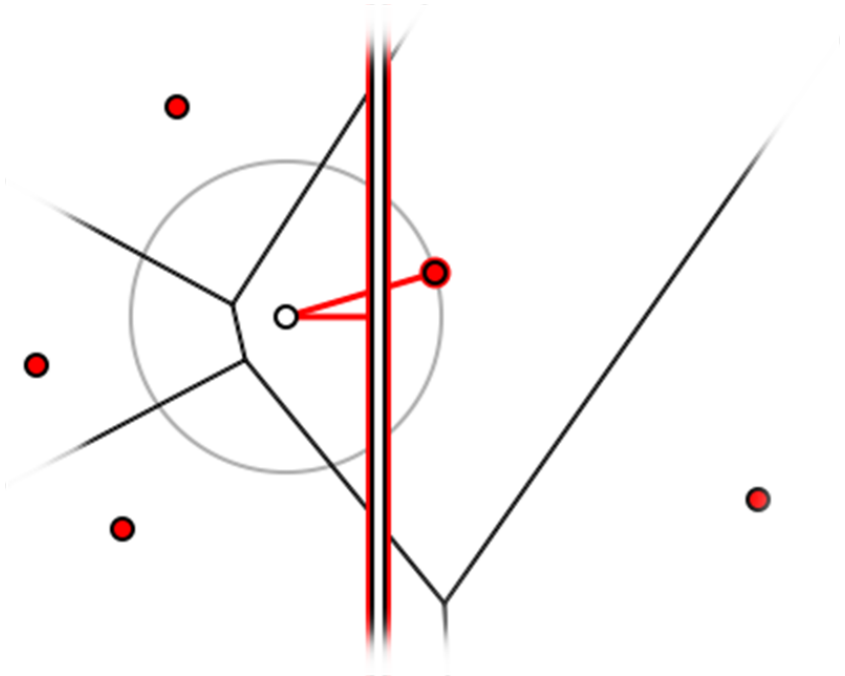
Dies ist hier **nicht** möglich: Das Voronoi Diagramm links von ℓ hängt auch von Zentren rechts von ℓ ab!

Welcher Teil des Voronoi-Diagramms ist bekannt?



Für welche $q \in \mathbb{R}^2$ links von ℓ kennen wir bereits das nächste Zentrum $p \in P$?

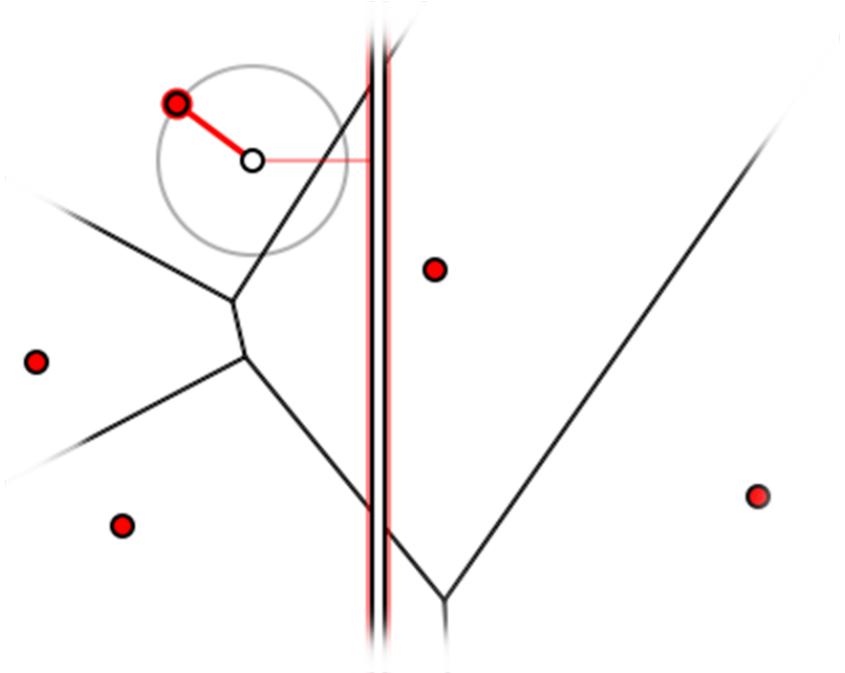
Welcher Teil des Voronoi-Diagramms ist bekannt?



Für welche $q \in \mathbb{R}^2$ links von ℓ kennen wir bereits das nächste Zentrum $p \in P$?

- Für $p' \in P$ rechts von ℓ gilt $dist(q, p') \geq dist(q, \ell)$

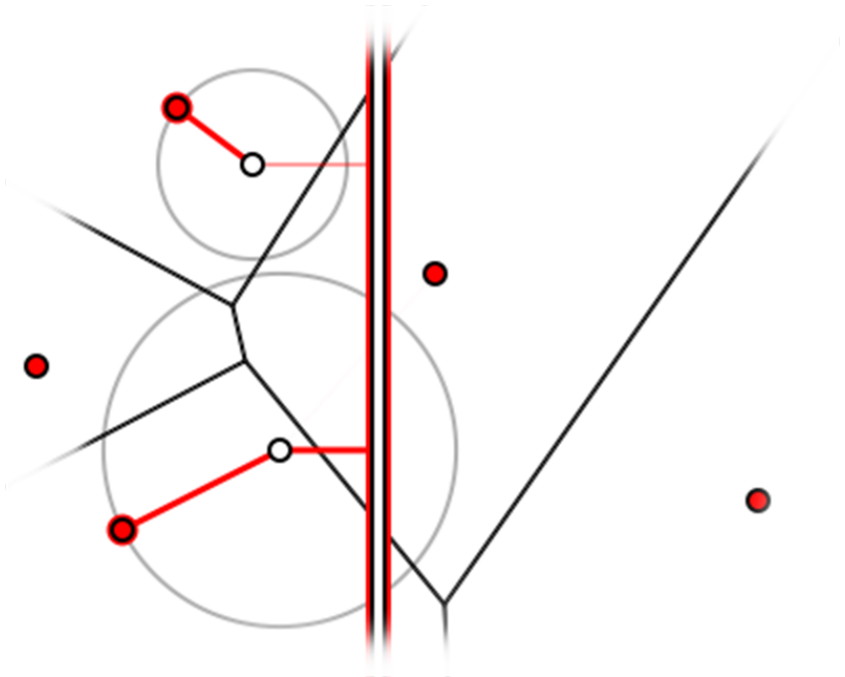
Welcher Teil des Voronoi-Diagramms ist bekannt?



Für welche $q \in \mathbb{R}^2$ links von ℓ kennen wir bereits das nächste Zentrum $p \in P$?

- Für $p' \in P$ rechts von ℓ gilt $dist(q, p') \geq dist(q, \ell)$
- Aus $dist(q, p) < dist(q, \ell)$ folgt somit $dist(q, p) < dist(q, p')$

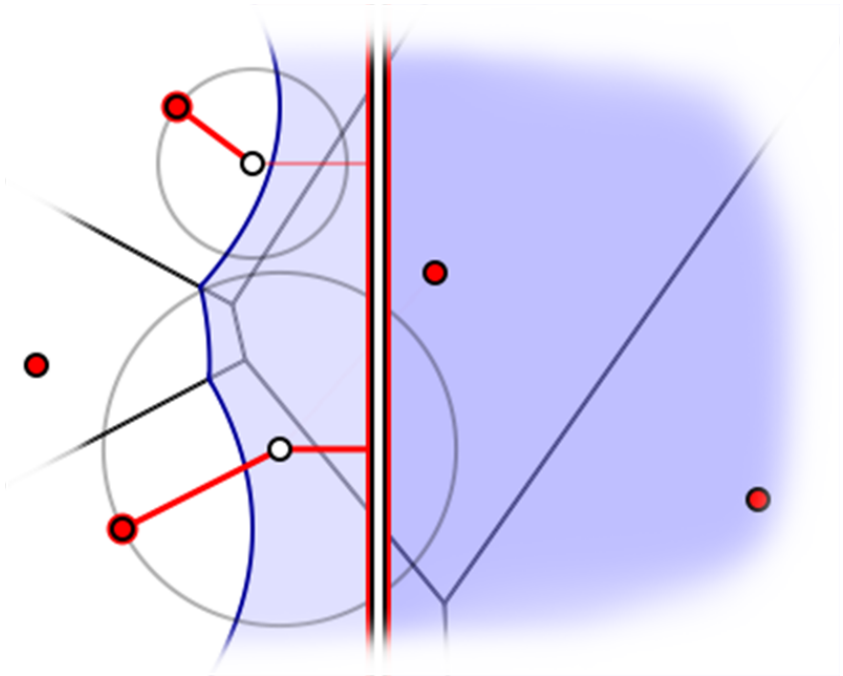
Welcher Teil des Voronoi-Diagramms ist bekannt?



Für welche $q \in \mathbb{R}^2$ links von ℓ kennen wir bereits das nächste Zentrum $p \in P$?

- Für $p' \in P$ rechts von ℓ gilt $dist(q, p') \geq dist(q, \ell)$
- Aus $dist(q, p) < dist(q, \ell)$ folgt somit $dist(q, p) < dist(q, p')$
- Für jedes $p \in P$ ist das Voronoi-Diagramm bekannt in
$$\{q \in \mathbb{R}^2 : dist(q, p) < dist(q, \ell)\}$$

Welcher Teil des Voronoi-Diagramms ist bekannt?

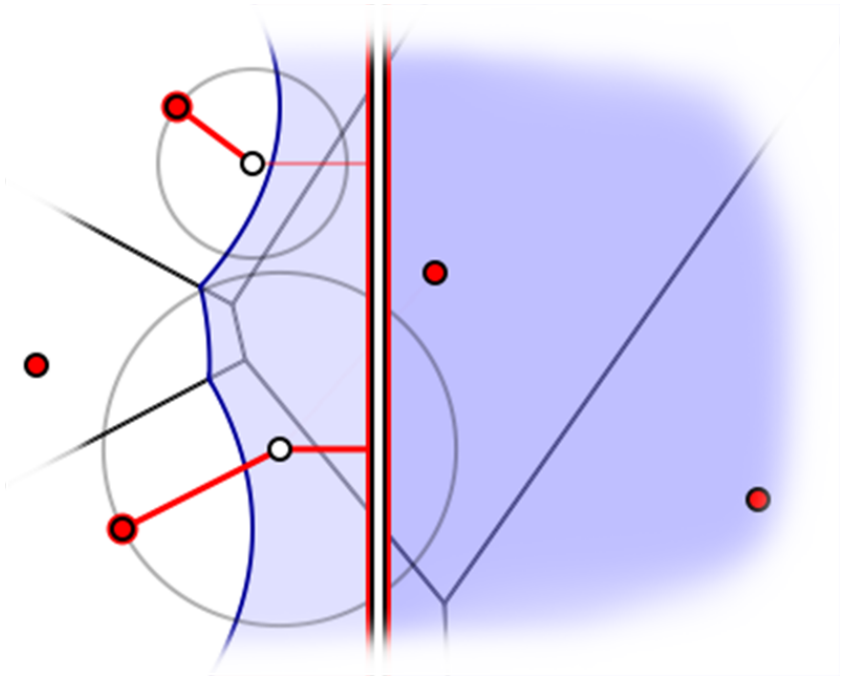


Für welche $q \in \mathbb{R}^2$ links von ℓ kennen wir bereits das nächste Zentrum $p \in P$?

- Insgesamt ist das Voronoi-Diagramm somit bekannt in

$$G = \bigcup_{p \in P} \{q \in \mathbb{R}^2 : \text{dist}(q, p) < \text{dist}(q, \ell)\}$$

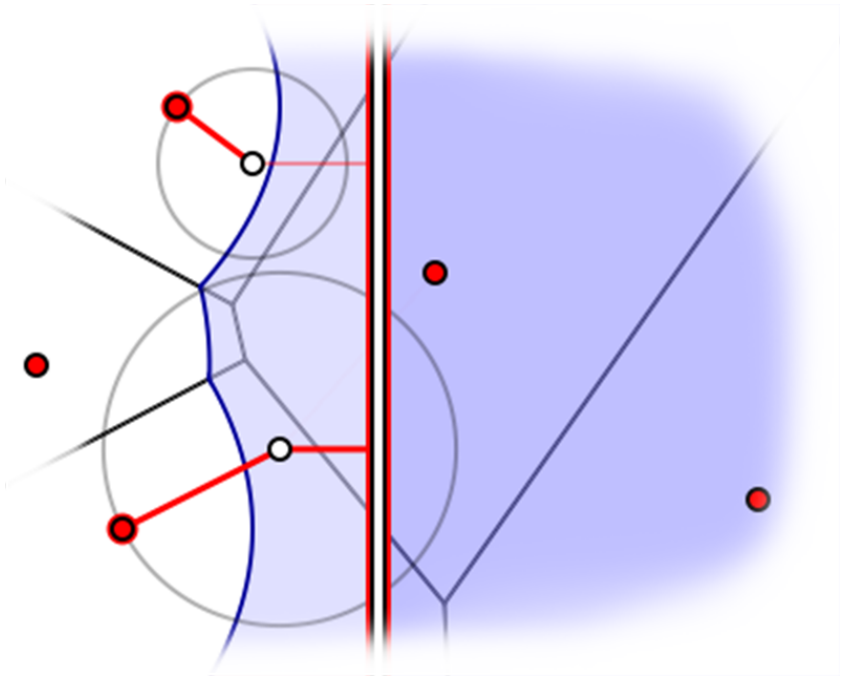
Welcher Teil des Voronoi-Diagramms ist bekannt?



Für welche $q \in \mathbb{R}^2$ links von ℓ kennen wir bereits das nächste Zentrum $p \in P$?

- ∂G heißt Beach-Line und besteht aus parabolischen Bögen

Welcher Teil des Voronoi-Diagramms ist bekannt?



Für welche $q \in \mathbb{R}^2$ links von ℓ kennen wir bereits das nächste Zentrum $p \in P$?

- ∂G heißt Beach-Line und besteht aus parabolischen Bögen
- Die Schnittpunkte dieser Bögen zeichnen die Voronoi-Kanten

Wann ändert sich die Struktur der Beach-Line?

Site Event

- ℓ erreicht Zentrum $p \in P$

Wann ändert sich die Struktur der Beach-Line?

Site Event

- ℓ erreicht Zentrum $p \in P$
- Ein Punkt auf der Beach-Line ist von p genau so weit entfernt wie vom zugehorigen Zentrum

Wann ändert sich die Struktur der Beach-Line?

Site Event

- ℓ erreicht Zentrum $p \in P$
- Ein Punkt auf der Beach-Line ist von p genau so weit entfernt wie vom zugehorigen Zentrum
- Neuer Bogen entsteht

Wann ändert sich die Struktur der Beach-Line?

Site Event

- ℓ erreicht Zentrum $p \in P$
- Ein Punkt auf der Beach-Line ist von p genau so weit entfernt wie vom zugehorigen Zentrum
- Neuer Bogen entsteht
- Dieser teilt höchstens einen alten Bogen in zwei Teile
  $\Longrightarrow$ Beach-Line besteht aus höchstens $2n - 1$ Bögen

Wann ändert sich die Struktur der Beach-Line?

Site Event

- ℓ erreicht Zentrum $p \in P$
- Ein Punkt auf der Beach-Line ist von p genau so weit entfernt wie vom zugehorigen Zentrum
- Neuer Bogen entsteht
- Dieser teilt höchstens einen alten Bogen in zwei Teile
  $\Longrightarrow$ Beach-Line besteht aus höchstens $2n - 1$ Bögen

Circle Event

- Ein Bogen zieht sich zu einem Punkt zusammen

Wann ändert sich die Struktur der Beach-Line?

Site Event

- ℓ erreicht Zentrum $p \in P$
- Ein Punkt auf der Beach-Line ist von p genau so weit entfernt wie vom zugehorigen Zentrum
- Neuer Bogen entsteht
- Dieser teilt höchstens einen alten Bogen in zwei Teile
  $\Longrightarrow$ Beach-Line besteht aus höchstens $2n - 1$ Bögen

Circle Event

- Ein Bogen zieht sich zu einem Punkt zusammen
- Ein Punkt auf der Beach-Line ist von mindestens drei Zentren genau so weit entfernt wie von ℓ

Wann ändert sich die Struktur der Beach-Line?

Site Event

- ℓ erreicht Zentrum $p \in P$
- Ein Punkt auf der Beach-Line ist von p genau so weit entfernt wie vom zugehorigen Zentrum
- Neuer Bogen entsteht
- Dieser teilt höchstens einen alten Bogen in zwei Teile
  $\Longrightarrow$ Beach-Line besteht aus höchstens $2n - 1$ Bögen

Circle Event

- Ein Bogen zieht sich zu einem Punkt zusammen
- Ein Punkt auf der Beach-Line ist von mindestens drei Zentren genau so weit entfernt wie von ℓ
- Dieser Punkt ist Voronoi-Knoten

Datenstrukturen

Was brauchen wir?

- Repräsentation des bereits bekannten Voronoi-Diagramms

Datenstrukturen

Was brauchen wir?

- Repräsentation des bereits bekannten Voronoi-Diagramms
- Event Queue für Site Events und Circle Events

Datenstrukturen

Was brauchen wir?

- Repräsentation des bereits bekannten Voronoi-Diagramms
- Event Queue für Site Events und Circle Events
- Repräsentation der Beach-Line

Datenstrukturen

Was brauchen wir?

- Repräsentation des bereits bekannten Voronoi-Diagramms
- Event Queue für Site Events und Circle Events
- Repräsentation der Beach-Line

Voronoi-Diagramm

- Doppelt verkettete Kantenliste D

Datenstrukturen

Was brauchen wir?

- Repräsentation des bereits bekannten Voronoi-Diagramms
- Event Queue für Site Events und Circle Events
- Repräsentation der Beach-Line

Voronoi-Diagramm

- Doppelt verkettete Kantenliste D
- Füge genügend große Bounding Box hinzu
  $\Longrightarrow$ alle Kanten werden endlich

Beach Line als balancierter binärer Baum T

Blätter

- Repräsentieren Bögen

Beach Line als balancierter binärer Baum T

Blätter

- Repräsentieren Bögen
- Speichern zugehöriges Zentrum

Beach Line als balancierter binärer Baum T

Blätter

- Repräsentieren Bögen
- Speichern zugehöriges Zentrum
- Pointer auf zugehöriges Circle Event in der Event Queue

Beach Line als balancierter binärer Baum T

Blätter

- Repräsentieren Bögen
- Speichern zugehöriges Zentrum
- Pointer auf zugehöriges Circle Event in der Event Queue

Innere Knoten

- Repräsentieren Schnittpunkte der Beach Line

Beach Line als balancierter binärer Baum T

Blätter

- Repräsentieren Bögen
- Speichern zugehöriges Zentrum
- Pointer auf zugehöriges Circle Event in der Event Queue

Innere Knoten

- Repräsentieren Schnittpunkte der Beach Line
- Speichern zu linkem und rechtem Bogen gehörende Zentren

Beach Line als balancierter binärer Baum T

Blätter

- Repräsentieren Bögen
- Speichern zugehöriges Zentrum
- Pointer auf zugehöriges Circle Event in der Event Queue

Innere Knoten

- Repräsentieren Schnittpunkte der Beach Line
- Speichern zu linkem und rechtem Bogen gehörende Zentren
- Damit kann bei Site Events der zugehöriger Bogen in $\mathcal{O}(\log n)$ gefunden werden

Beach Line als balancierter binärer Baum T

Blätter

- Repräsentieren Bögen
- Speichern zugehöriges Zentrum
- Pointer auf zugehöriges Circle Event in der Event Queue

Innere Knoten

- Repräsentieren Schnittpunkte der Beach Line
- Speichern zu linkem und rechtem Bogen gehörende Zentren
- Damit kann bei Site Events der zugehöriger Bogen in $\mathcal{O}(\log n)$ gefunden werden
- Pointer auf Kante in Kantenliste die dieser Schnittpunkt zeichnet

Event Queue

Prioritätswarteschlange Q

- Priorisiert nach x-Koordinate

Event Queue

Prioritätswarteschlange Q

- Priorisiert nach x-Koordinate
- Enthält die bereits bekannten zukünftigen Ereignisse

Event Queue

Prioritätswarteschlange Q

- Priorisiert nach x-Koordinate
- Enthält die bereits bekannten zukünftigen Ereignisse
- Site Event $\longrightarrow$ Position des neuen Zentrums

Event Queue

Prioritätswarteschlange Q

- Priorisiert nach x-Koordinate
- Enthält die bereits bekannten zukünftigen Ereignisse
- Site Event $\longrightarrow$ Position des neuen Zentrums
- Circle Event $\longrightarrow$ rechtester Punkt des Kreises
  enthält Pointer auf Blatt in T, das verschwindenden Bogen repräsentiert

Circle Events

- Alle Site Events sind von Anfang an bekannt

Circle Events

- Alle Site Events sind von Anfang an bekannt
- Die Circle Events jedoch nicht

Circle Events

- Alle Site Events sind von Anfang an bekannt
- Die Circle Events jedoch nicht
- Bei jedem Event entstehen neue Tripel benachbarter Bögen und alte verschwinden

Circle Events

- Alle Site Events sind von Anfang an bekannt
- Die Circle Events jedoch nicht
- Bei jedem Event entstehen neue Tripel benachbarter Bögen und alte verschwinden
- Der Algorithmus muss die zugehörigen Circle Events verwalten

Circle Events

- Alle Site Events sind von Anfang an bekannt
- Die Circle Events jedoch nicht
- Bei jedem Event entstehen neue Tripel benachbarter Bögen und alte verschwinden
- Der Algorithmus muss die zugehörigen Circle Events verwalten

Nicht jedes Tripel führt zu einem Circle Event!

Circle Events

- Alle Site Events sind von Anfang an bekannt
- Die Circle Events jedoch nicht
- Bei jedem Event entstehen neue Tripel benachbarter Bögen und alte verschwinden
- Der Algorithmus muss die zugehörigen Circle Events verwalten

Nicht jedes Tripel führt zu einem Circle Event!

- Schnittpunkte müssen nicht zusammenlaufen

Circle Events

- Alle Site Events sind von Anfang an bekannt
- Die Circle Events jedoch nicht
- Bei jedem Event entstehen neue Tripel benachbarter Bögen und alte verschwinden
- Der Algorithmus muss die zugehörigen Circle Events verwalten

Nicht jedes Tripel führt zu einem Circle Event!

- Schnittpunkte müssen nicht zusammenlaufen
- Tripel kann vor Circle Event verschwinden

Circle Events

Ablauf bei jedem Event

- Teste für jedes neue Tripel benachbarter Bögen ob deren Schnittpunkte zusammenlaufen. Füge gegebenenfalls ein Circle Event in Q ein.

Circle Events

Ablauf bei jedem Event

- Teste für jedes neue Tripel benachbarter Bögen ob deren Schnittpunkte zusammenlaufen. Füge gegebenenfalls ein Circle Event in Q ein.
- Prüfe für jedes verschwindende Tripel ob ein entsprechendes Circle Event in Q existiert. Lösche dieses gegebenenfalls.

Circle Events

Ablauf bei jedem Event

- Teste für jedes neue Tripel benachbarter Bögen ob deren Schnittpunkte zusammenlaufen. Füge gegebenenfalls ein Circle Event in Q ein.
- Prüfe für jedes verschwindende Tripel ob ein entsprechendes Circle Event in Q existiert. Lösche dieses gegebenenfalls.

Satz

Jeder Voronoi-Knoten wird durch ein Circle-Event gefunden.

Algorithmus

Algorithm 1: VoronoiDiagram(P)

```
1 Initialize Q with all site events, initialize an empty tree T and an
  empty edge list D
2 while Q is not empty do
3     Remove the event with smallest x-coordinate from Q
4     if the event is a site event, occuring at site p_i then
5         HandleSiteEvent(p_i)
6     else
7         HandleCircleEvent(γ), where γ is the leaf of T
          representing the arc that will disappear
8 Compute a bounding box that contains all vertices of the Voronoi
  diagram and attach the half-infinite edges to the bounding box by
  updating the edge list D appropriately
9 Return D
```

Algorithmus

Algorithm 2: HandleSiteEvent(p_i)

1. If T is empty, insert p_i and return. Otherwise continue
2. Search in T for the arc α left of p_i. If the leaf representing α has a pointer to a circle event in Q, deleted it from Q.
3. Replace the leaf of T that represents α with a subtree having three leaves. The middle leaf stores the new site p_i and the other two leaves store the site p_j that was originally stored with α. Store the tuples (p_j, p_i) and (p_i, p_j) representing the new breakpoints at the two new internal nodes.
4. Create new half-edge records in the Voronoi diagram structure for the edge separating $V(p_i)$ and $V(p_j)$, which will be traced out by the two new breakpoints.
5. Check the new triples of consecutive arcs to see if the breakpoints converge. If so, insert the circle events into Q and add pointers between the node in T and the node in Q.

Algorithmus

Algorithm 3: HandleCircleEvent(γ)

1. Delete the leaf γ that represents the disappearing arc α from T. Update the breakpoints at the internal nodes. Delete all circle events involving α from Q.
2. Add the center of the circle causing the event as a vertex to the doubly-connected edge list D. Create a new half-edge corresponding to the new breakpoint of the beach line. Attach both to the half-edges that end at the vertex.
3. Check the new triples of consecutive arcs to see if the two breakpoints of the triple converge. If so, insert the corresponding circle event into Q and set pointers between the new circle event in Q and the corresponding leaf of T.

Degenierte Fälle

Beispiele

- Zwei Site Events mit gleicher x-Koordinate

Degenierte Fälle

Beispiele

- Zwei Site Events mit gleicher x-Koordinate
- Zusammenfallende Circle Events

Degenierte Fälle

Beispiele

- Zwei Site Events mit gleicher x-Koordinate
- Zusammenfallende Circle Events
- Site Event rechts von Schnittpunkt der Beach-Line

Degenierte Fälle

Beispiele

- Zwei Site Events mit gleicher x-Koordinate
- Zusammenfallende Circle Events
- Site Event rechts von Schnittpunkt der Beach-Line

$\rightsquigarrow$ Degenerierte Fälle werden korrekt behandelt

Auswertung

Laufzeit

- Baum- und Queue-Operationen in $\mathcal{O}(\log n)$

Auswertung

Laufzeit

- Baum- und Queue-Operationen in $\mathcal{O}(\log n)$
- Listenoperationen in $\mathcal{O}(1)$

Auswertung

Laufzeit

- Baum- und Queue-Operationen in $\mathcal{O}(\log n)$
- Listenoperationen in $\mathcal{O}(1)$
- Pro Event konstante Anzahl solcher Operationen

Auswertung

Laufzeit

- Baum- und Queue-Operationen in $\mathcal{O}(\log n)$
- Listenoperationen in $\mathcal{O}(1)$
- Pro Event konstante Anzahl solcher Operationen
  $\Longrightarrow$ Laufzeit pro Event in $\mathcal{O}(\log n)$

Auswertung

Laufzeit

- Baum- und Queue-Operationen in $\mathcal{O}(\log n)$
- Listenoperationen in $\mathcal{O}(1)$
- Pro Event konstante Anzahl solcher Operationen
  $\Longrightarrow$ Laufzeit pro Event in $\mathcal{O}(\log n)$
- Ein Site Event pro Zentrum

Auswertung

Laufzeit

- Baum- und Queue-Operationen in $\mathcal{O}(\log n)$
- Listenoperationen in $\mathcal{O}(1)$
- Pro Event konstante Anzahl solcher Operationen
  $\Longrightarrow$ Laufzeit pro Event in $\mathcal{O}(\log n)$
- Ein Site Event pro Zentrum
  $\Longrightarrow$ n Site Events

Auswertung

Laufzeit

- Baum- und Queue-Operationen in $\mathcal{O}(\log n)$
- Listenoperationen in $\mathcal{O}(1)$
- Pro Event konstante Anzahl solcher Operationen
  $\Longrightarrow$ Laufzeit pro Event in $\mathcal{O}(\log n)$
- Ein Site Event pro Zentrum
  $\Longrightarrow$ n Site Events
- Ein Circle Event pro Voronoi-Knoten

Auswertung

Laufzeit

- Baum- und Queue-Operationen in $\mathcal{O}(\log n)$
- Listenoperationen in $\mathcal{O}(1)$
- Pro Event konstante Anzahl solcher Operationen
  $\Longrightarrow$ Laufzeit pro Event in $\mathcal{O}(\log n)$
- Ein Site Event pro Zentrum
  $\Longrightarrow$ n Site Events
- Ein Circle Event pro Voronoi-Knoten
  $\Longrightarrow$ $k \in \mathcal{O}(n)$ Circle Events

Auswertung

Laufzeit

- Baum- und Queue-Operationen in $\mathcal{O}(\log n)$
- Listenoperationen in $\mathcal{O}(1)$
- Pro Event konstante Anzahl solcher Operationen
  $\Longrightarrow$ Laufzeit pro Event in $\mathcal{O}(\log n)$
- Ein Site Event pro Zentrum
  $\Longrightarrow$ n Site Events
- Ein Circle Event pro Voronoi-Knoten
  $\Longrightarrow$ $k \in \mathcal{O}(n)$ Circle Events
  $\Longrightarrow$ Gesamtanzahl der Events in $\mathcal{O}(n)$

Auswertung

Laufzeit

- Baum- und Queue-Operationen in $\mathcal{O}(\log n)$
- Listenoperationen in $\mathcal{O}(1)$
- Pro Event konstante Anzahl solcher Operationen
  $\Longrightarrow$ Laufzeit pro Event in $\mathcal{O}(\log n)$
- Ein Site Event pro Zentrum
  $\Longrightarrow$ n Site Events
- Ein Circle Event pro Voronoi-Knoten
  $\Longrightarrow$ $k \in \mathcal{O}(n)$ Circle Events
  $\Longrightarrow$ Gesamtanzahl der Events in $\mathcal{O}(n)$
  $\Longrightarrow$ Gesamtlaufzeit in $\mathcal{O}(n \log n)$

Hauptergebnis

Satz

Das Voronoi-Diagramm lässt sich in $\mathcal{O}(n \log n)$ berechnen.

Hauptergebnis

Satz

Das Voronoi-Diagramm lässt sich in $\mathcal{O}(n \log n)$ berechnen.

Geht es besser?

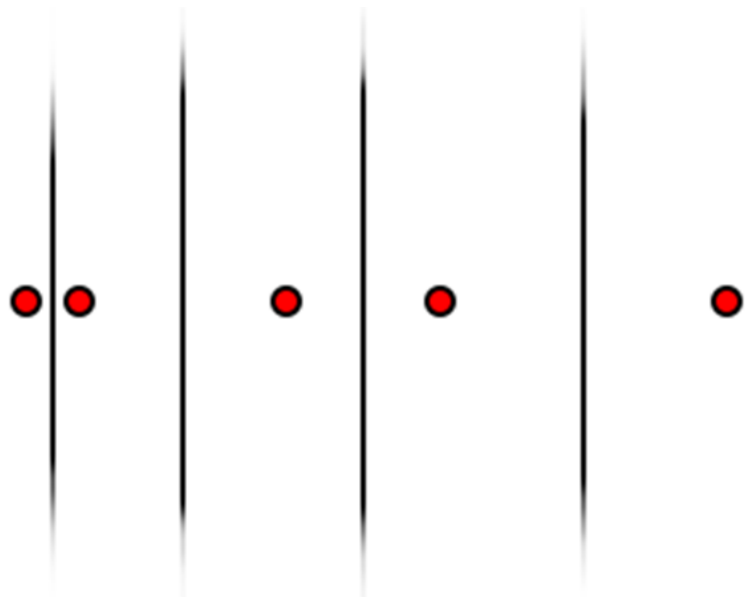
Hauptergebnis

Satz

Das Voronoi-Diagramm lässt sich in $\mathcal{O}(n \log n)$ berechnen.

Geht es besser?

- n Zahlen lassen sich mit einem Voronoi-Diagramm sortieren



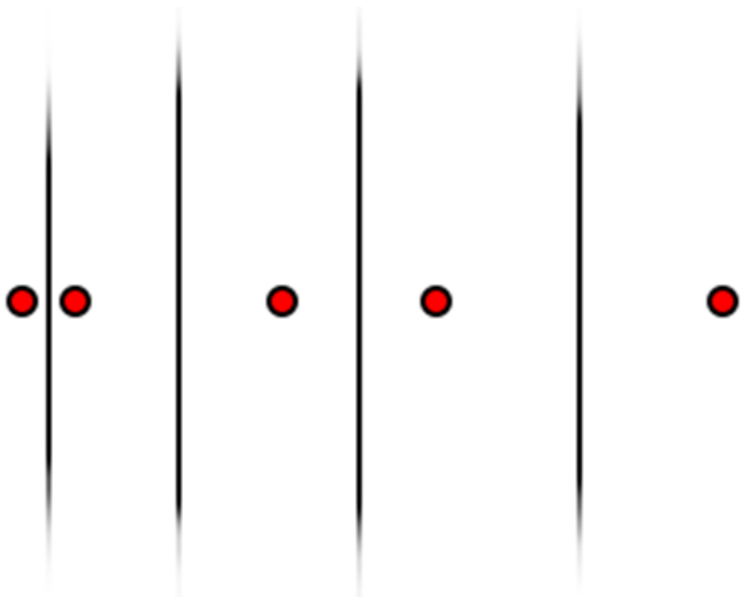
Hauptergebnis

Satz

Das Voronoi-Diagramm lässt sich in $\mathcal{O}(n \log n)$ berechnen.

Geht es besser?

- n Zahlen lassen sich mit einem Voronoi-Diagramm sortieren
- Berechnung des Voronoi-Diagramms in $\Omega(n \log n)$



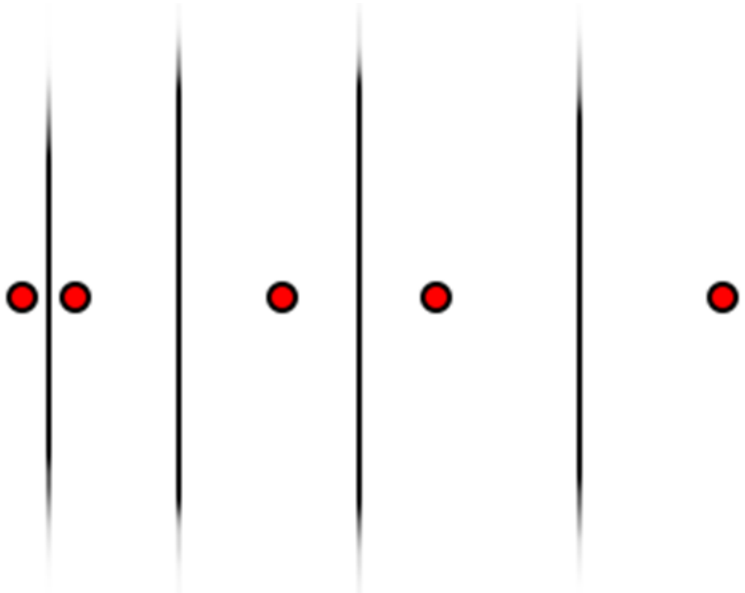
Hauptergebnis

Satz

Das Voronoi-Diagramm lässt sich in $\mathcal{O}(n \log n)$ berechnen.

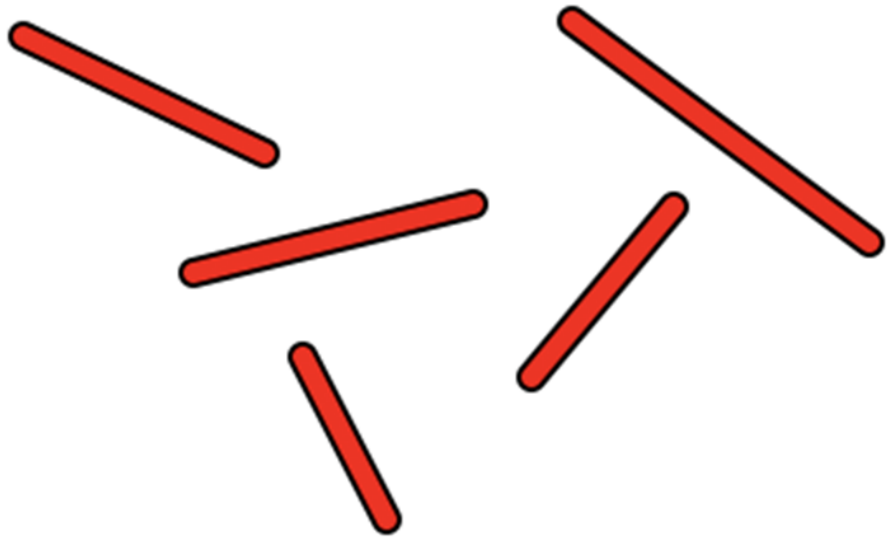
Geht es besser?

- n Zahlen lassen sich mit einem Voronoi-Diagramm sortieren
- Berechnung des Voronoi-Diagramms in $\Omega(n \log n)$
- Fortune's Algorithmus ist optimal!



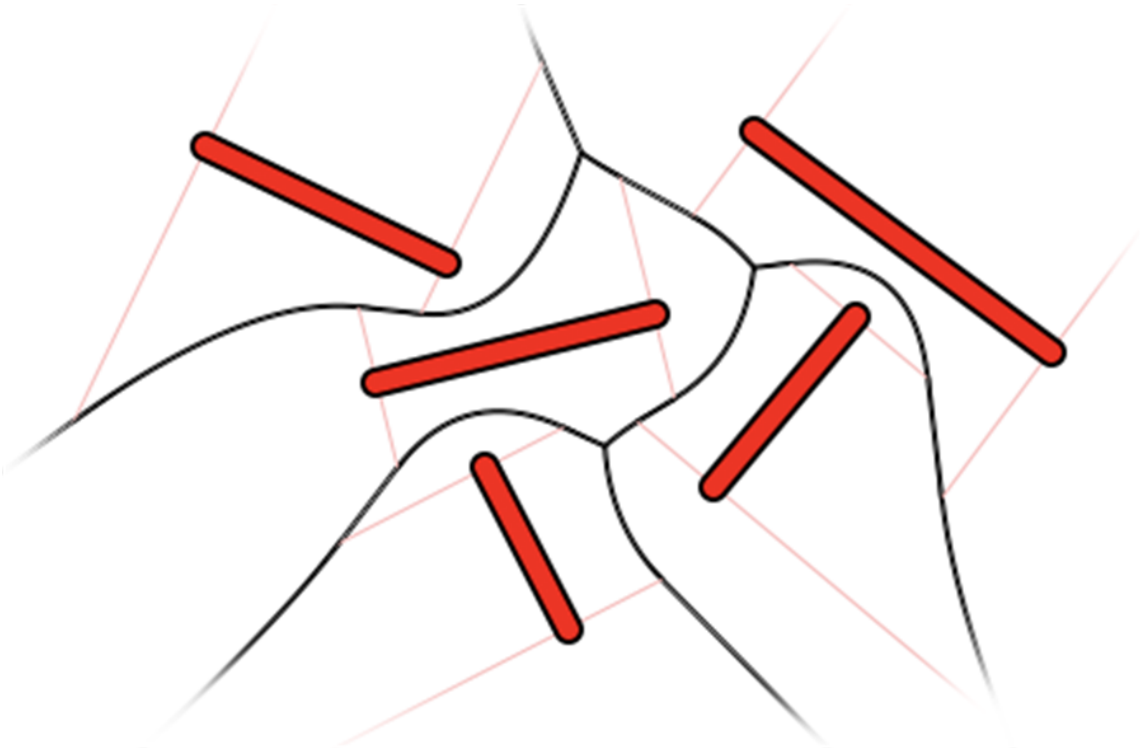
Voronoi Diagramme von Strecken

Statt Punkten $P = \{p_1, \ldots, p_n\}$ betrachten wir jetzt Strecken $S = \{s_1, \ldots, s_n\}$ als Zentren der Voronoi-Regionen.



Voronoi Diagramme von Strecken

Statt Punkten $P = \{p_1, \ldots, p_n\}$ betrachten wir jetzt Strecken $S = \{s_1, \ldots, s_n\}$ als Zentren der Voronoi-Regionen.



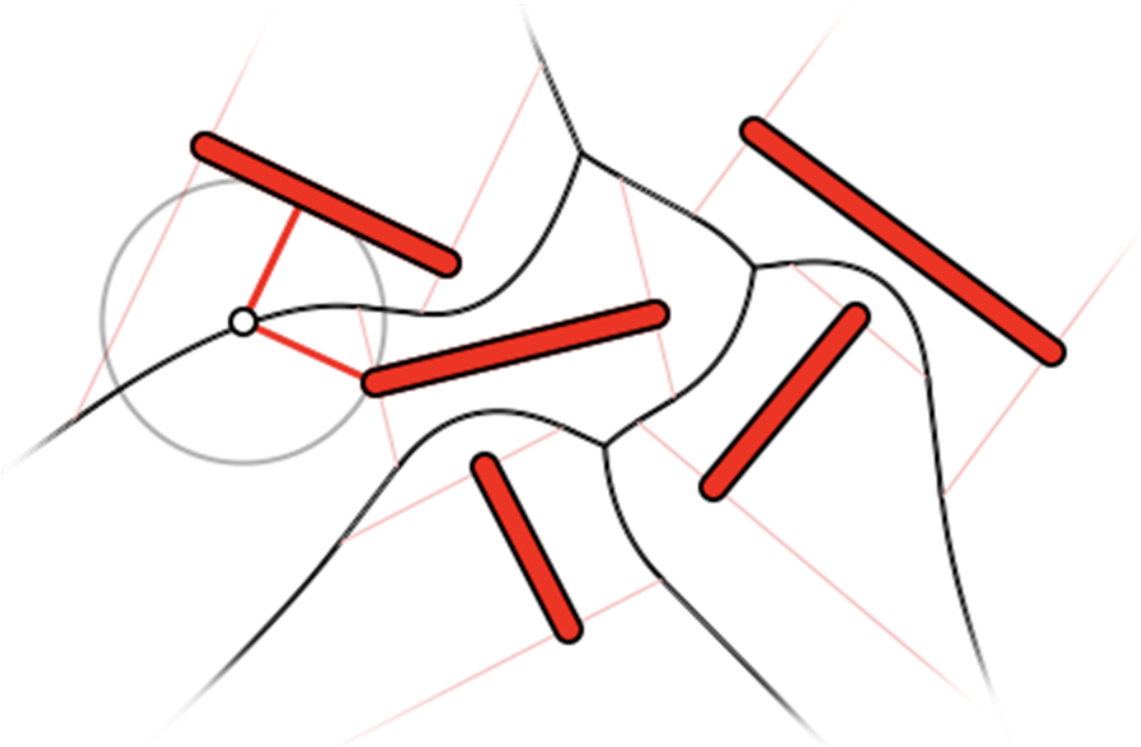
Voronoi Diagramme von Strecken

Statt Punkten $P = \{p_1, \ldots, p_n\}$ betrachten wir jetzt Strecken $S = \{s_1, \ldots, s_n\}$ als Zentren der Voronoi-Regionen.

Definition

Für $q \in \mathbb{R}^2$ definiere den Abstand $\text{dist}(q, s)$ zu $s \in S$ als

$$\text{dist}(q, s) = \min_{p \in s} \text{dist}(q, p)$$



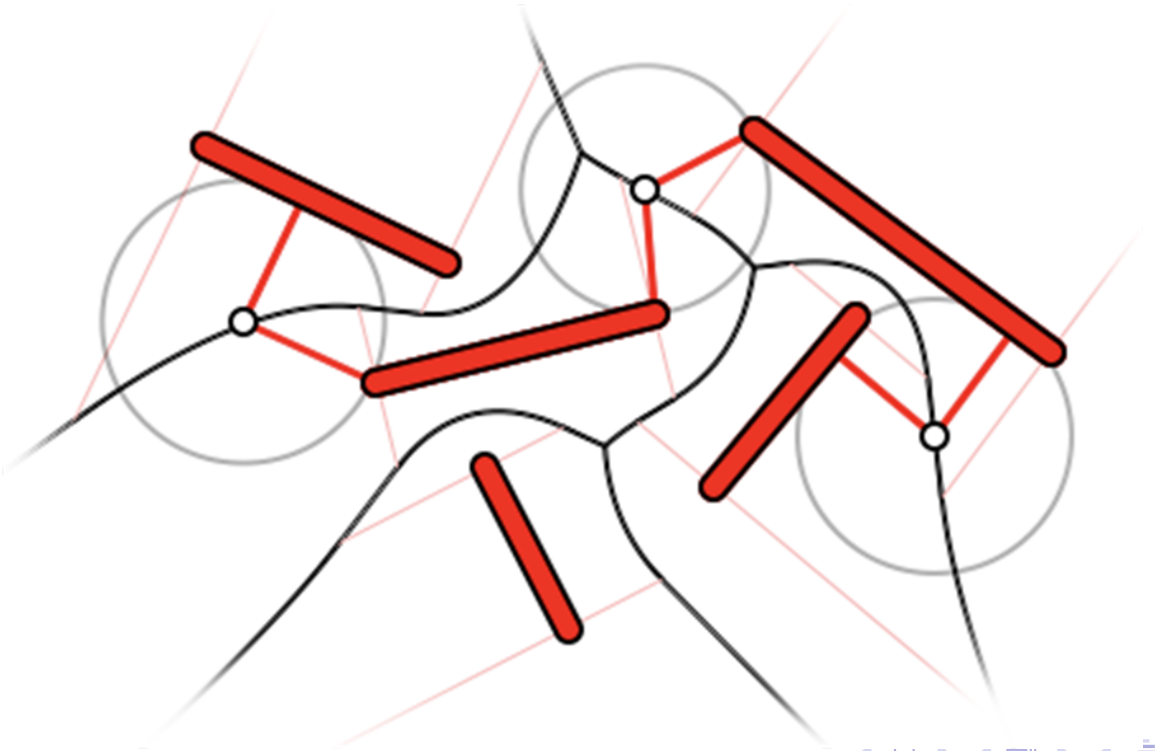
Voronoi Diagramme von Strecken

Statt Punkten $P = \{p_1, \ldots, p_n\}$ betrachten wir jetzt Strecken $S = \{s_1, \ldots, s_n\}$ als Zentren der Voronoi-Regionen.

Definition

Für $q \in \mathbb{R}^2$ definiere den Abstand $\text{dist}(q, s)$ zu $s \in S$ als

$$\text{dist}(q, s) = \min_{p \in s} \text{dist}(q, p)$$



Voronoi Diagramme von Strecken

Statt Punkten $P = \{p_1, \ldots, p_n\}$ betrachten wir jetzt Strecken $S = \{s_1, \ldots, s_n\}$ als Zentren der Voronoi-Regionen.

Definition

Für $q \in \mathbb{R}^2$ definiere den Abstand $\text{dist}(q, s)$ zu $s \in S$ als

$$\text{dist}(q, s) = \min_{p \in s} \text{dist}(q, p)$$

Bisektoren zweier Zentren bestehen nun aus bis zu 7 Teilen, die jeweils entweder eine Strecke oder ein parabolischer Bogen sind.

Voronoi Diagramme von Strecken

Statt Punkten $P = \{p_1, \ldots, p_n\}$ betrachten wir jetzt Strecken $S = \{s_1, \ldots, s_n\}$ als Zentren der Voronoi-Regionen.

Definition

Für $q \in \mathbb{R}^2$ definiere den Abstand $\text{dist}(q, s)$ zu $s \in S$ als

$$\text{dist}(q, s) = \min_{p \in s} \text{dist}(q, p)$$

Bisektoren zweier Zentren bestehen nun aus bis zu 7 Teilen, die jeweils entweder eine Strecke oder ein parabolischer Bogen sind. Die Komplexität des Voronoi-Diagramms ist weiterhin in $\mathcal{O}(n)$.

Beach-Line

Zur Konstruktion der Beach-Line betrachten wir weiterhin nur den Teil von S, der links von ℓ liegt. Sie besteht nun aus Strecken und parabolischen Bögen.

Beach-Line

Zur Konstruktion der Beach-Line betrachten wir weiterhin nur den Teil von S, der links von ℓ liegt. Sie besteht nun aus Strecken und parabolischen Bögen.

verschiedene Rollen der Schnittpunkte

Ein Schnittpunkt fährt eine Voronoi-Kante ab
$\Longleftrightarrow$ Die zugehörigen Teile der Beach-Line gehören zu unterschiedlichen Strecken aus S

Beach-Line

Zur Konstruktion der Beach-Line betrachten wir weiterhin nur den Teil von S, der links von ℓ liegt. Sie besteht nun aus Strecken und parabolischen Bögen.

verschiedene Rollen der Schnittpunkte

Ein Schnittpunkt fährt eine Voronoi-Kante ab
$\Longleftrightarrow$ Die zugehörigen Teile der Beach-Line gehören zu unterschiedlichen Strecken aus S

unterschiedliches Verhalten der Schnittpunkte

Ein Schnittpunkt fährt eine Parabel ab
$\Longleftrightarrow$ Ein Teil gehört zu einem Endpunkt einer Strecke und der andere zum Inneren einer anderen Strecke

Event Queue

Site Events

- Startpunkte teilen ein Stück der Beach-Line in zwei Teile und fügen vier weitere hinzu

Event Queue

Site Events

- Startpunkte teilen ein Stück der Beach-Line in zwei Teile und fügen vier weitere hinzu
- Endpunkte ersetzen einen Schnittpunkt der Beach-Line durch parabolischen Bogen

Event Queue

Site Events

- Startpunkte teilen ein Stück der Beach-Line in zwei Teile und fügen vier weitere hinzu
- Endpunkte ersetzen einen Schnittpunkt der Beach-Line durch parabolischen Bogen

Circle Events

- Signalisieren das Verschwinden eines Teils der Beach-Line

Event Queue

Site Events

- Startpunkte teilen ein Stück der Beach-Line in zwei Teile und fügen vier weitere hinzu
- Endpunkte ersetzen einen Schnittpunkt der Beach-Line durch parabolischen Bogen

Circle Events

- Signalisieren das Verschwinden eines Teils der Beach-Line
- Induzieren einen Voronoi-Knoten

Sweep-Line-Algorithmus

Fortune's Algorithmus lässt sich erweitern

- Deutlich mehr Fallunterscheidungen

Sweep-Line-Algorithmus

Fortune's Algorithmus lässt sich erweitern

- Deutlich mehr Fallunterscheidungen

Laufzeit

- Pro Event in $\mathcal{O}(\log n)$

Sweep-Line-Algorithmus

Fortune's Algorithmus lässt sich erweitern

- Deutlich mehr Fallunterscheidungen

Laufzeit

- Pro Event in $\mathcal{O}(\log n)$
- Insgesamt $\mathcal{O}(n)$ Ereignisse

Sweep-Line-Algorithmus

Fortune's Algorithmus lässt sich erweitern

- Deutlich mehr Fallunterscheidungen

Laufzeit

- Pro Event in $\mathcal{O}(\log n)$
- Insgesamt $\mathcal{O}(n)$ Ereignisse
  $\Longrightarrow$ Gesamtlaufeit in $\mathcal{O}(n \log n)$

Voronoi Diagrams

Christian Wellenbrock

December 1, 2009