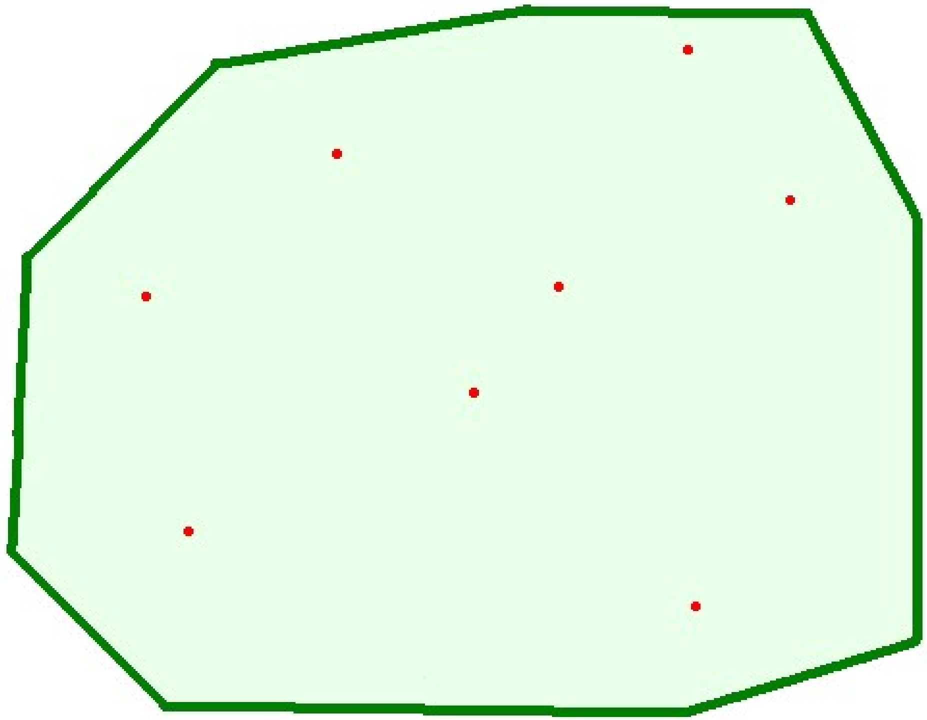
Voronoi-Diagramme und Delaunay-Triangulation

Voronoi-Diagramm

Problembeispiele

Voronoi-Diagramm

Beispiel Brandwächter im Wald

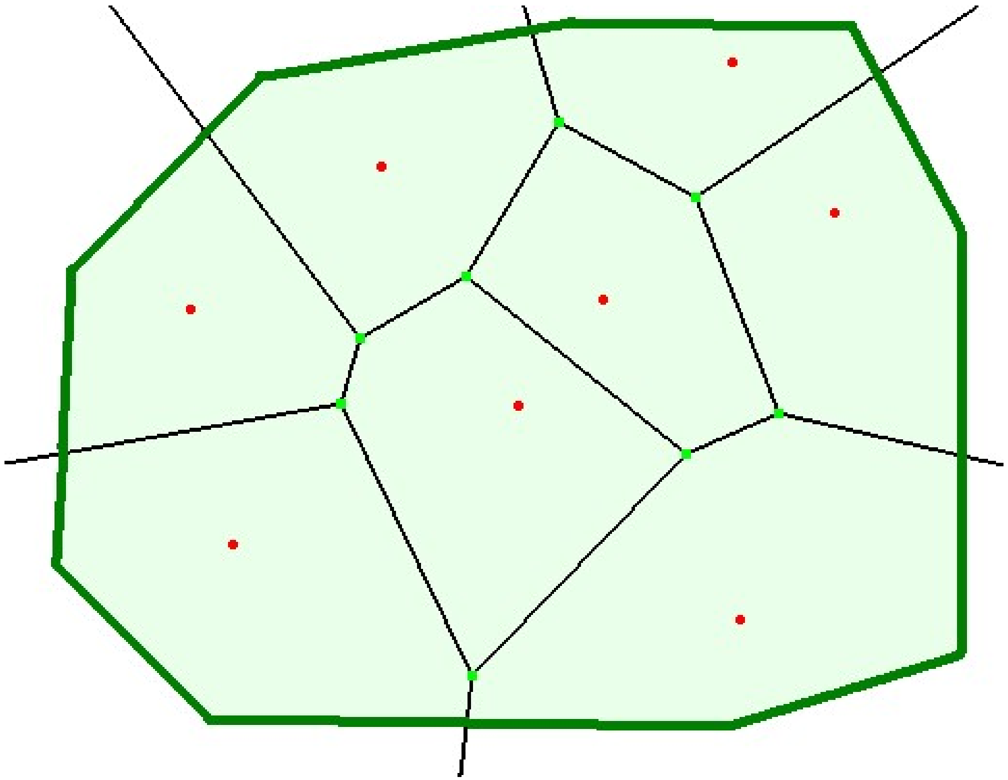


Grüne Fläche: Wald
Rote Punkte: Brandwachtürme

Frage: Welche Fläche muss jeder Brandwachturm überwachen?

Voronoi-Diagramm

Beispiel Brandwächter im Wald



Antwort: Jeder Brandwächter überwacht die Bäume, die näher zu seinem Turm sind (Voronoi-Regionen), als zu jedem anderen.

Das „Voronoi-Diagramm“ zeichnet die Linien zwischen den Gebieten

Voronoi-Diagramm

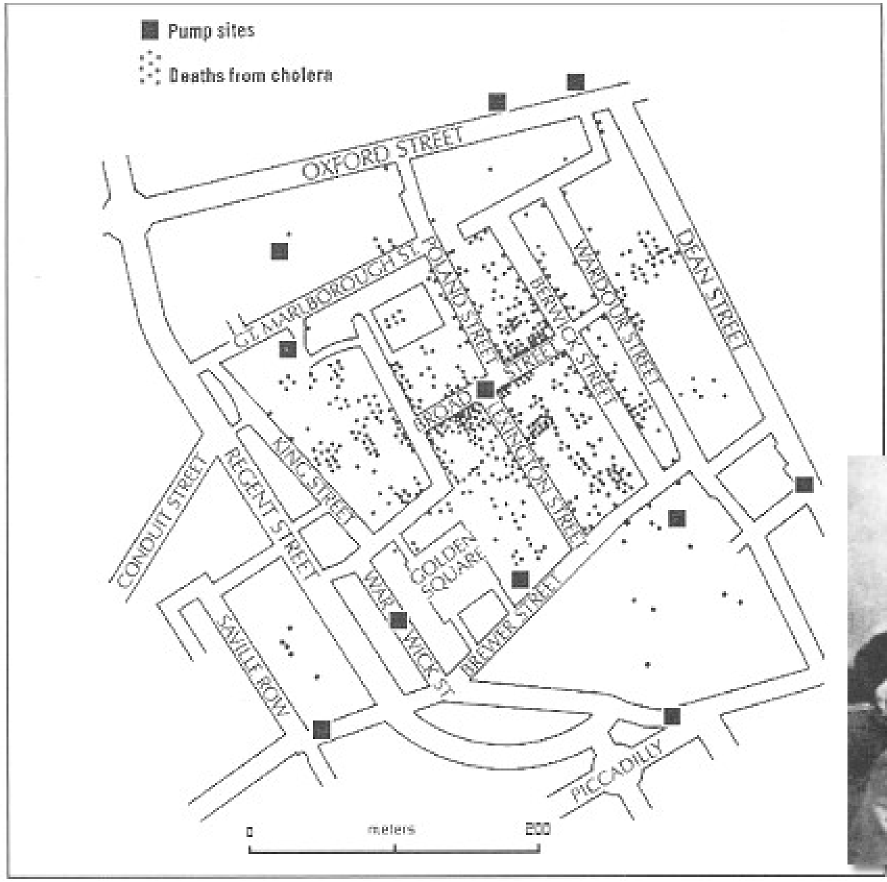
Weitere Beispiele

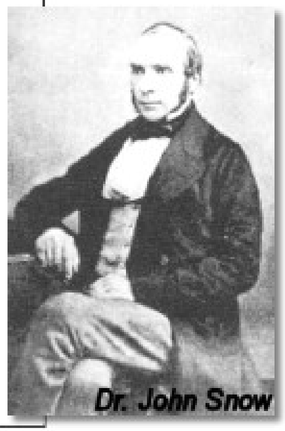
- Nächstgelegenes, öffentliches Telefon (akademischer Klassiker ;-)
- Positionierung von Funkmasten
- Bahnplanung in Hindernislandschaft

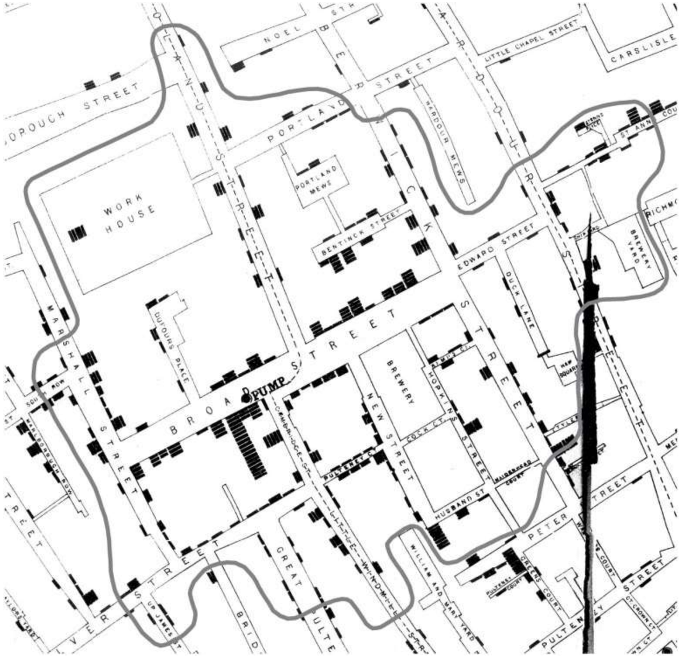
Voronoi-Diagramm

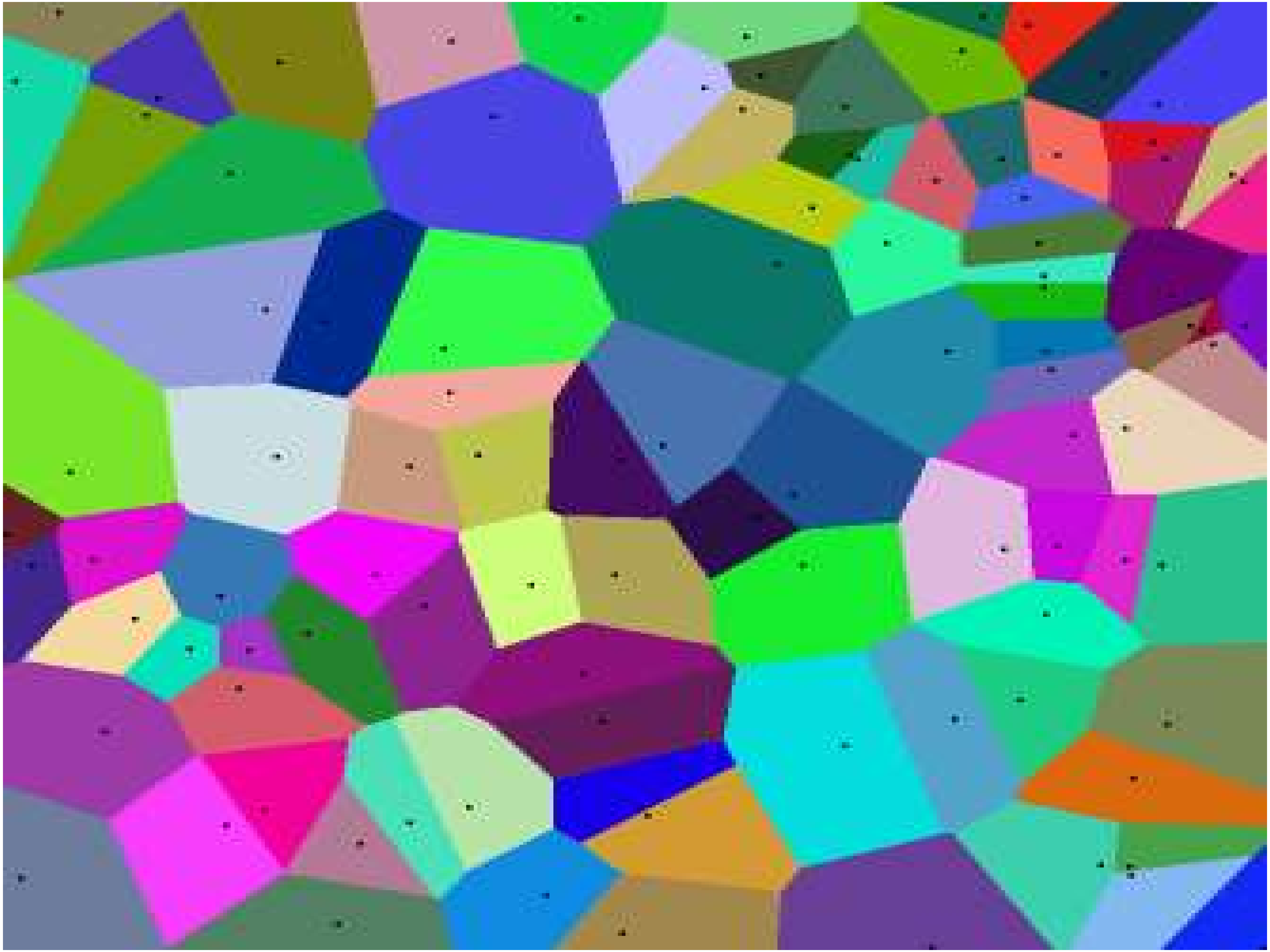
Weitere Beispiele

- Erste Anwendung: 1855, Choleraepidemie London, John Snow









Voronoi-Diagramm

Definition:

gegeben: n Punkte $P = \{p_1 \dots p_n\}$

gesucht: Regionen $N_1 \dots N_n$. (Einzugsbereiche)

$$N_i = \{p \in \mathbb{R}^2 \mid d(p, p_i) \leq d(p, p_j) \ \forall j\}$$

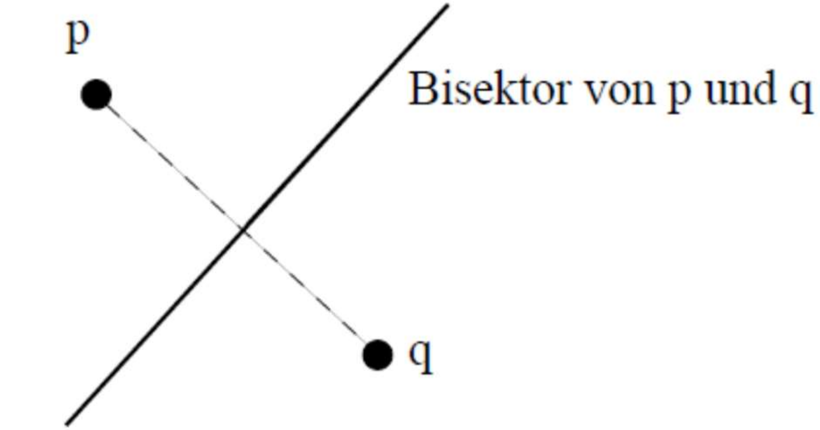
Voronoi-Diagramm *VD(P)* einer Punktmenge *P*:

Menge der Punkte, die zu mindestens zwei Einzugsbereichen gehören

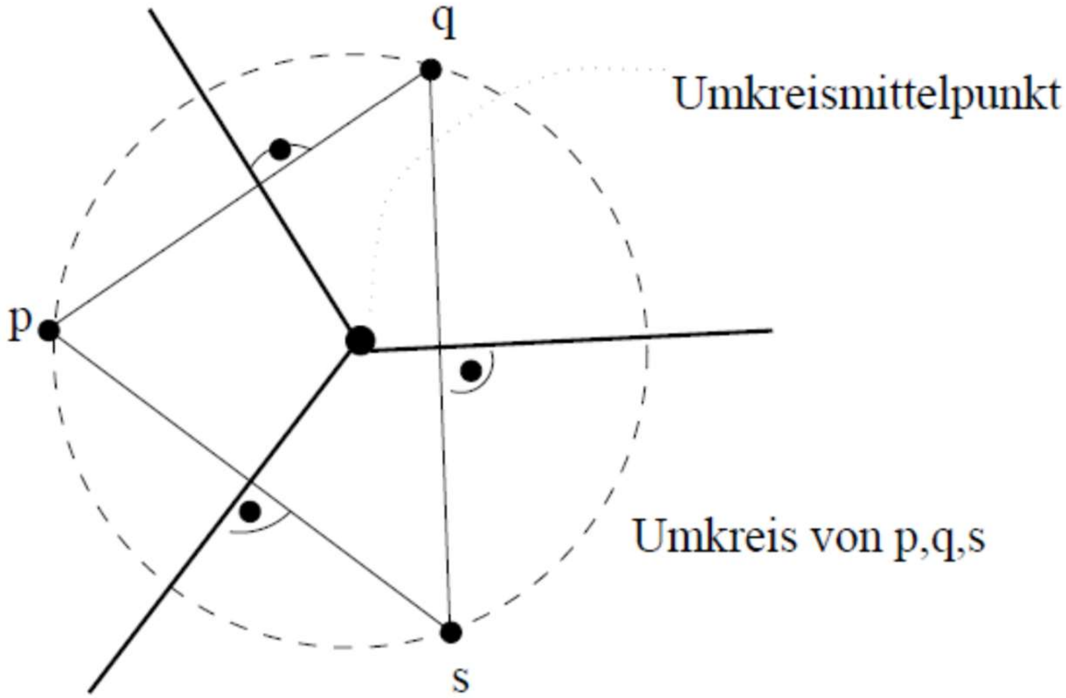
Voronoi-Diagramm

Beispiel:

n=2:



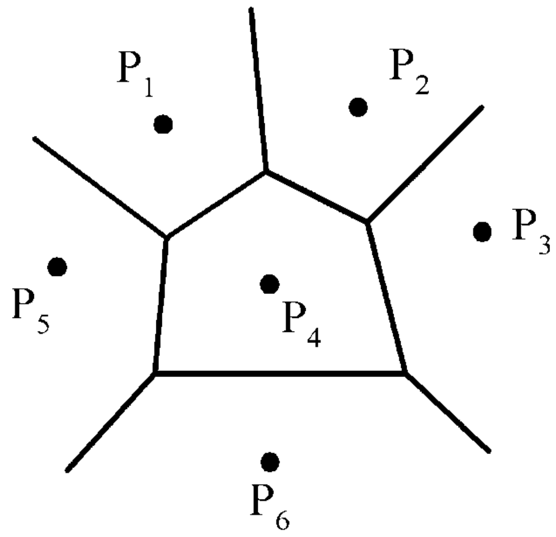
n=3:



Voronoi-Diagramm

Voronoi-Diagramm einer endlichen Punktmenge in 2D:

Graph aus Ecken und Kanten
Kanten sind Strecken oder Halbgeraden



Algorithmen: später…

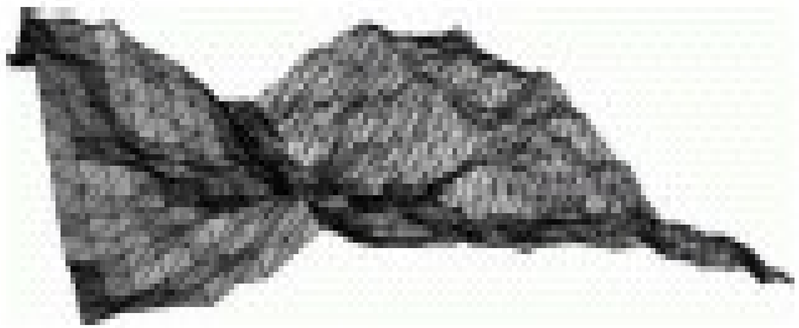
Delaunay-Triangulierung

Standard-Werkzeug in Computergrafik, 3D-Modellierung,…

Terrain-Visualisierung

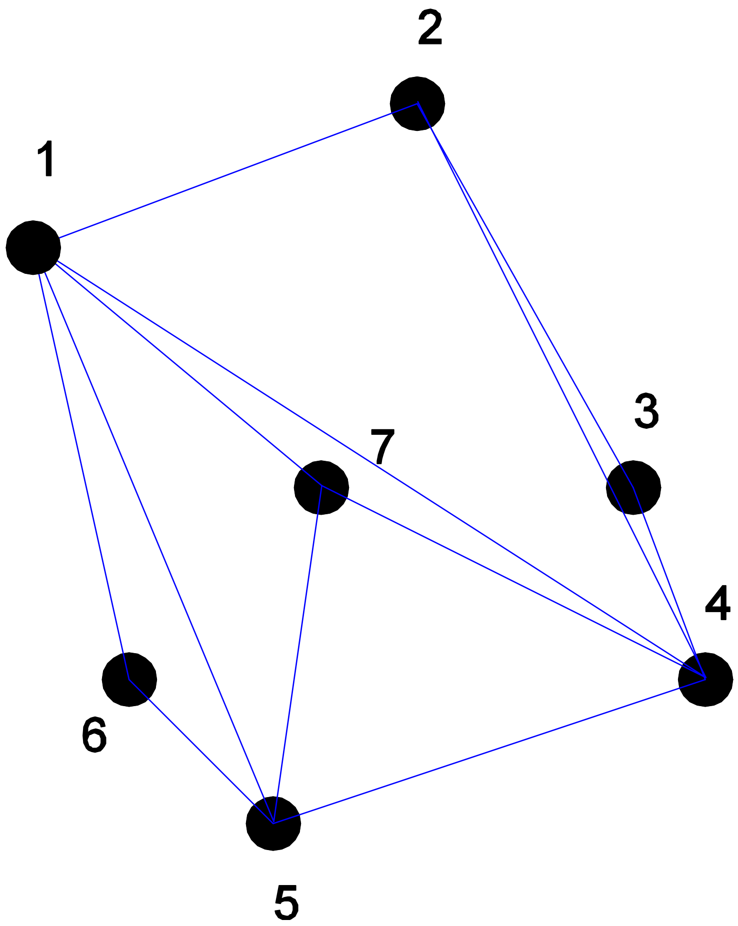
Umwandlung von Karten in 3D (bzw. 2.5D)

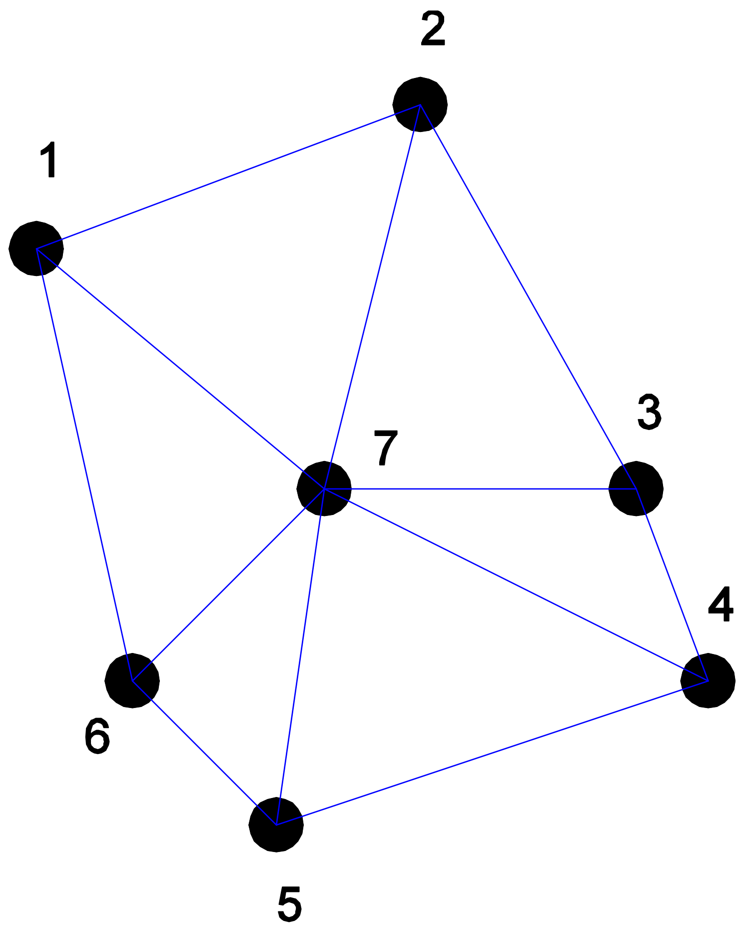
- Erfordern gute Triangulierungen von Punktmengen



Delaunay-Triangulierung

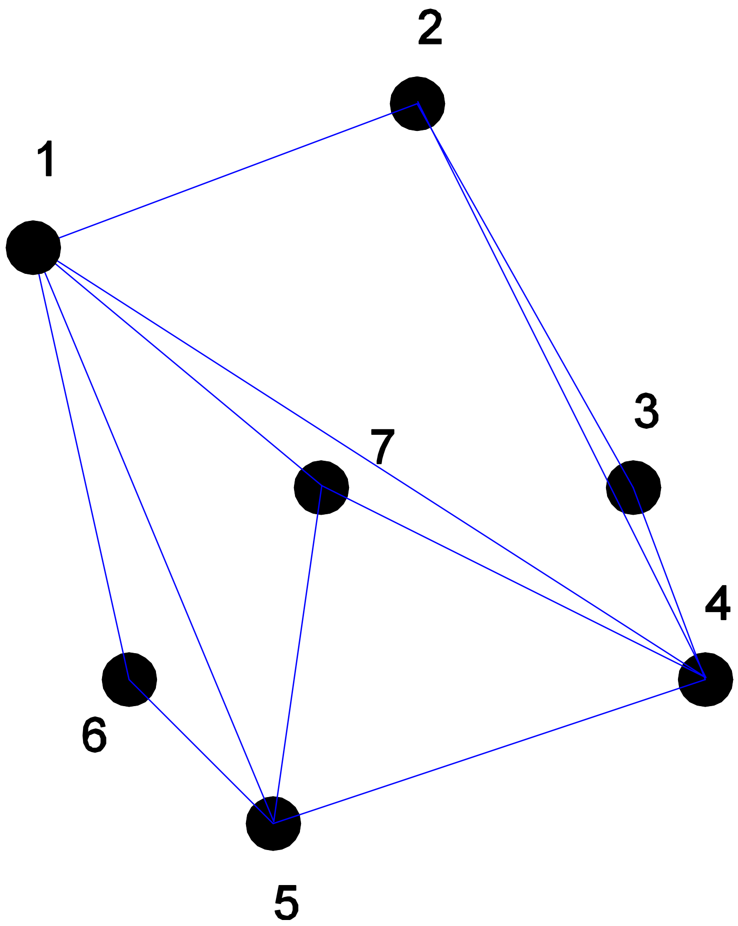
Problem: Finden einer „guten“ Triangulierung

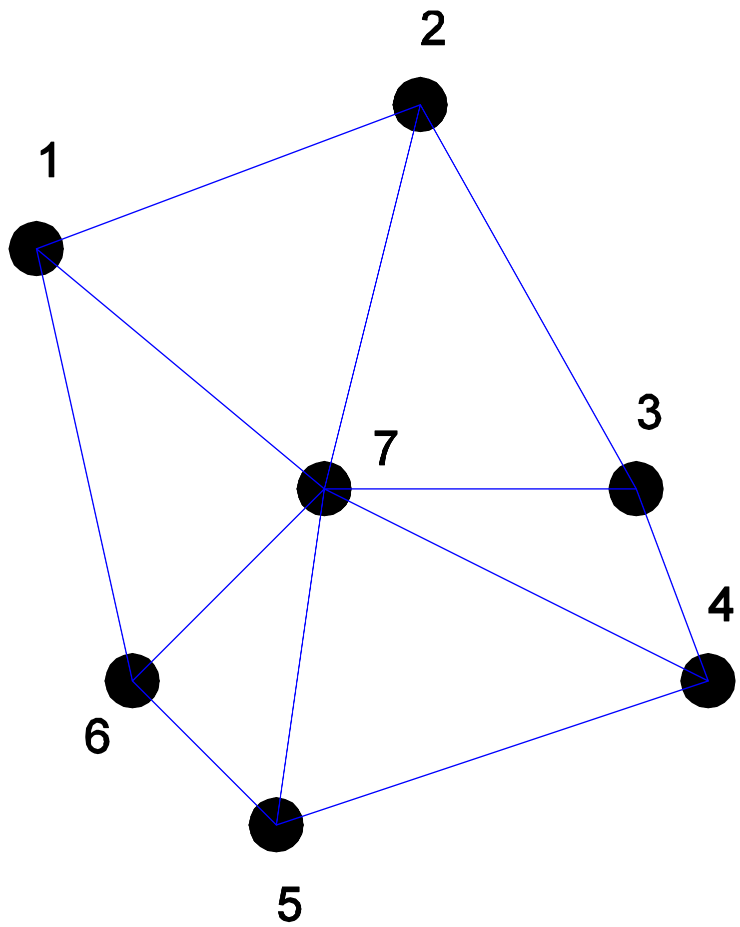




Delaunay-Triangulierung

Problem: Was ist gut???

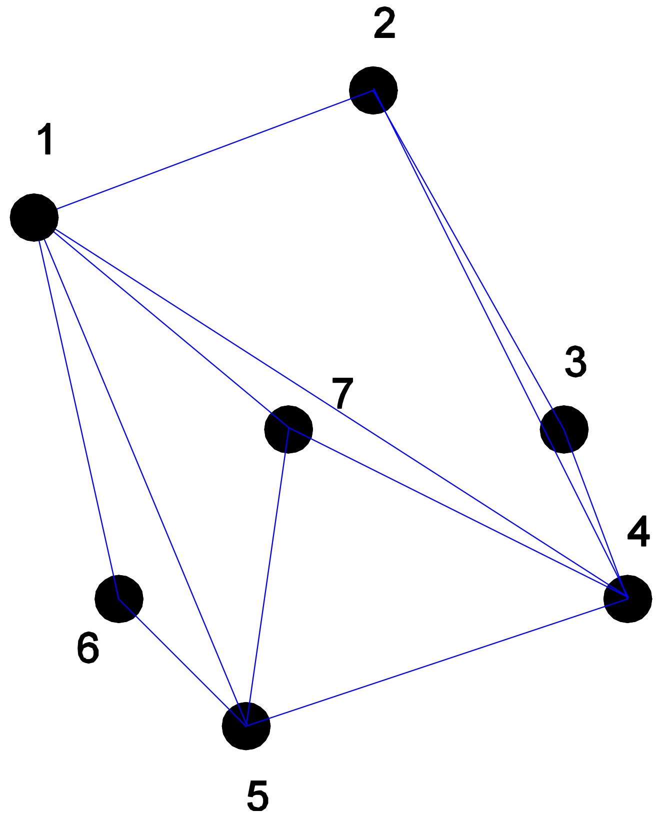


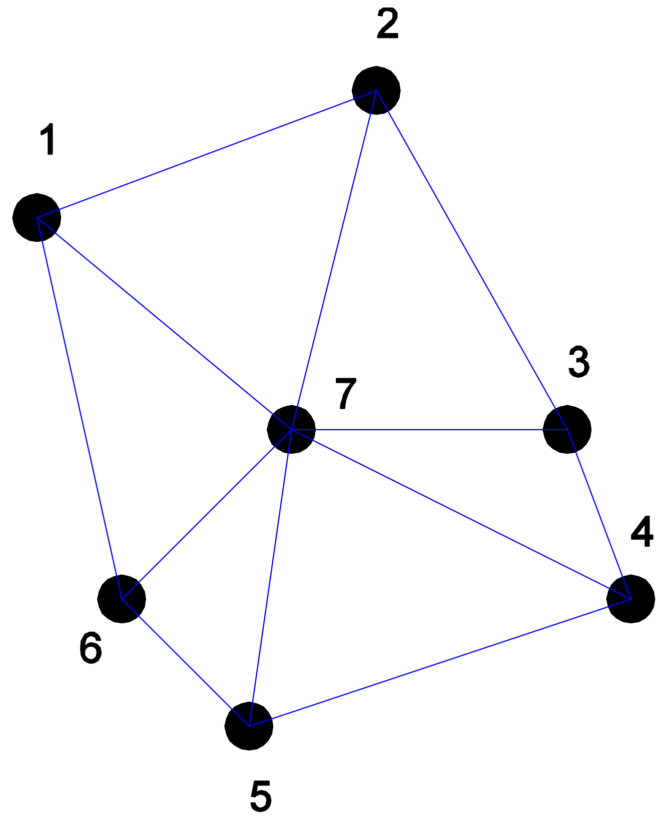


Delaunay-Triangulierung

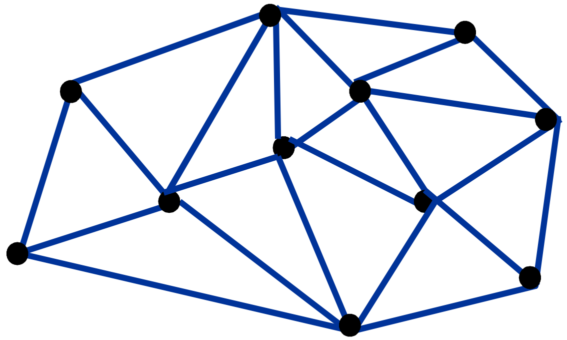
Problem: Was ist gut???

- Keine „schmalen, spitzen“ Dreiecke
- Dreiecke mit maximalen Innenwinkeln





Delaunay-Triangulierung



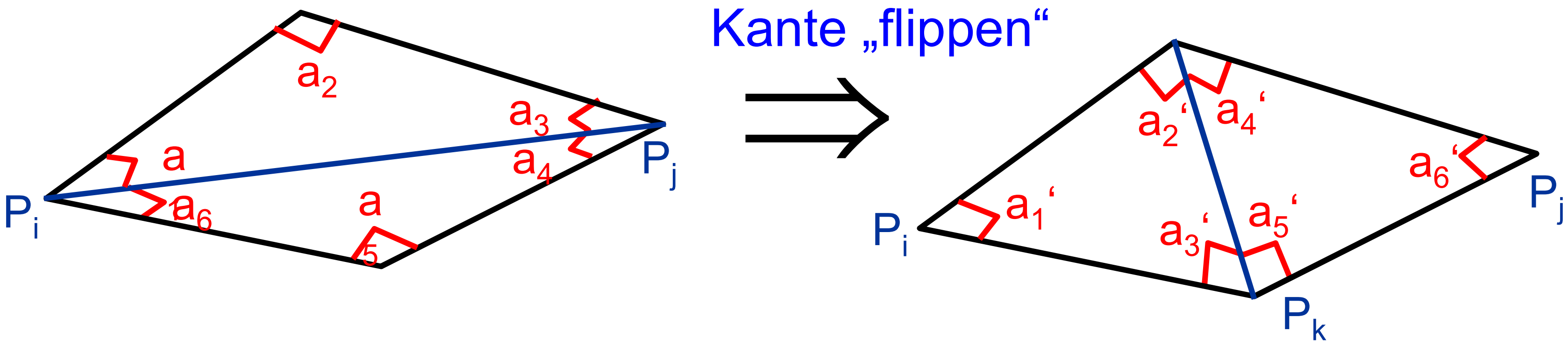
Winkelvektor:

Sei *T(P)* eine Triangulierung von *P* (Menge mit n Punkten).
Annahme: *T(P)* hat *m* Dreiecke.
Betrachte die *3m* Winkel der Dreiecke von *T(P)*, aufsteigend sortiert.
$A(T) = \{ a_1, \ldots, a_{3m} \}$ heißt *Winkelvektor* von *T*.

Triangulierungen können lexikographisch nach *A(T)* sortiert werden.

Eine Triangulierung *T(P)* heißt *winkeloptimal*, wenn $A(T(P)) \geq A(T'(P))$ für alle Triangulierungen *T´* von *P*.

Delaunay-Triangulierung



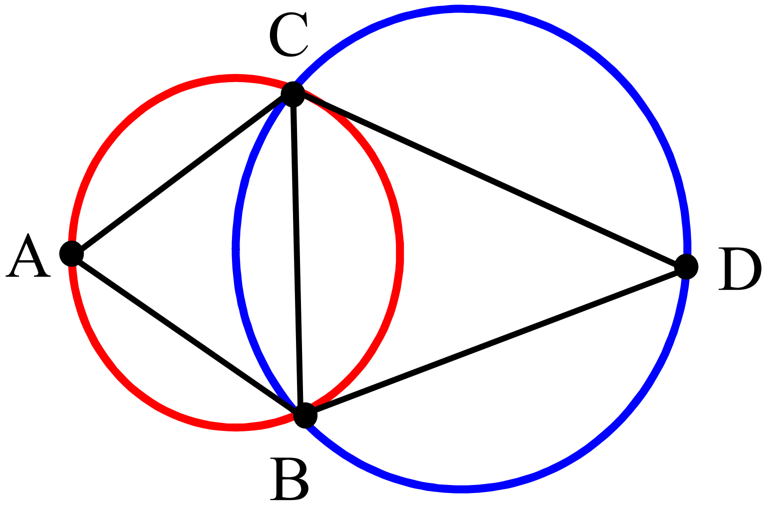
Illegale Kanten

Die Kante $p_i p_j$ ist illegal wenn $$\min_{1 \le i \le 6} \alpha_i < \min_{1 \le i \le 6} \alpha'_i$$

Anm: Sei *T* eine Triangulierung mit einer illegalen Kante *e*.
Sei *T´* die Triangulierung, die aus *T* durch „Flippen“ von *e entsteht*.
Dann ist $A(T') > A(T)$.

ABER: „Flippen“ erfordert rekursive Betrachtung von Nachbardreiecken!!!

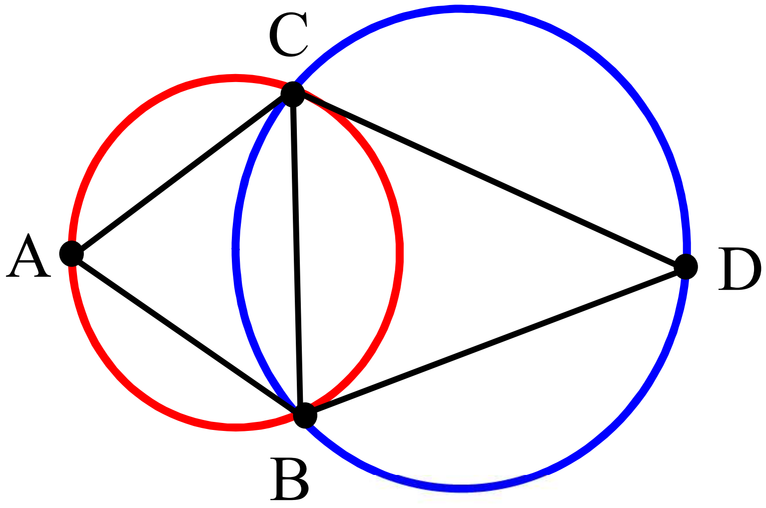
Delaunay-Triangulierung



Kreis-Kriterium:

Eine Triangulierung erfüllt das Kreis-Kriterium genau dann, wenn sich im Umkreis jedes Dreiecks der Triangulierung kein weiterer Punkt der zu triangulierenden Punktmenge befindet.

Delaunay-Triangulierung

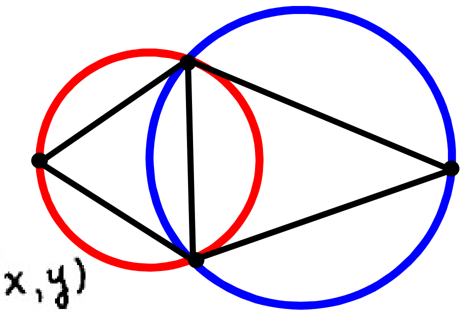


Kreis-Kriterium:
Effizient abprüfbar :

A, B, C counterclockwise und D liegt im Umkreis von ABC $\Leftrightarrow$

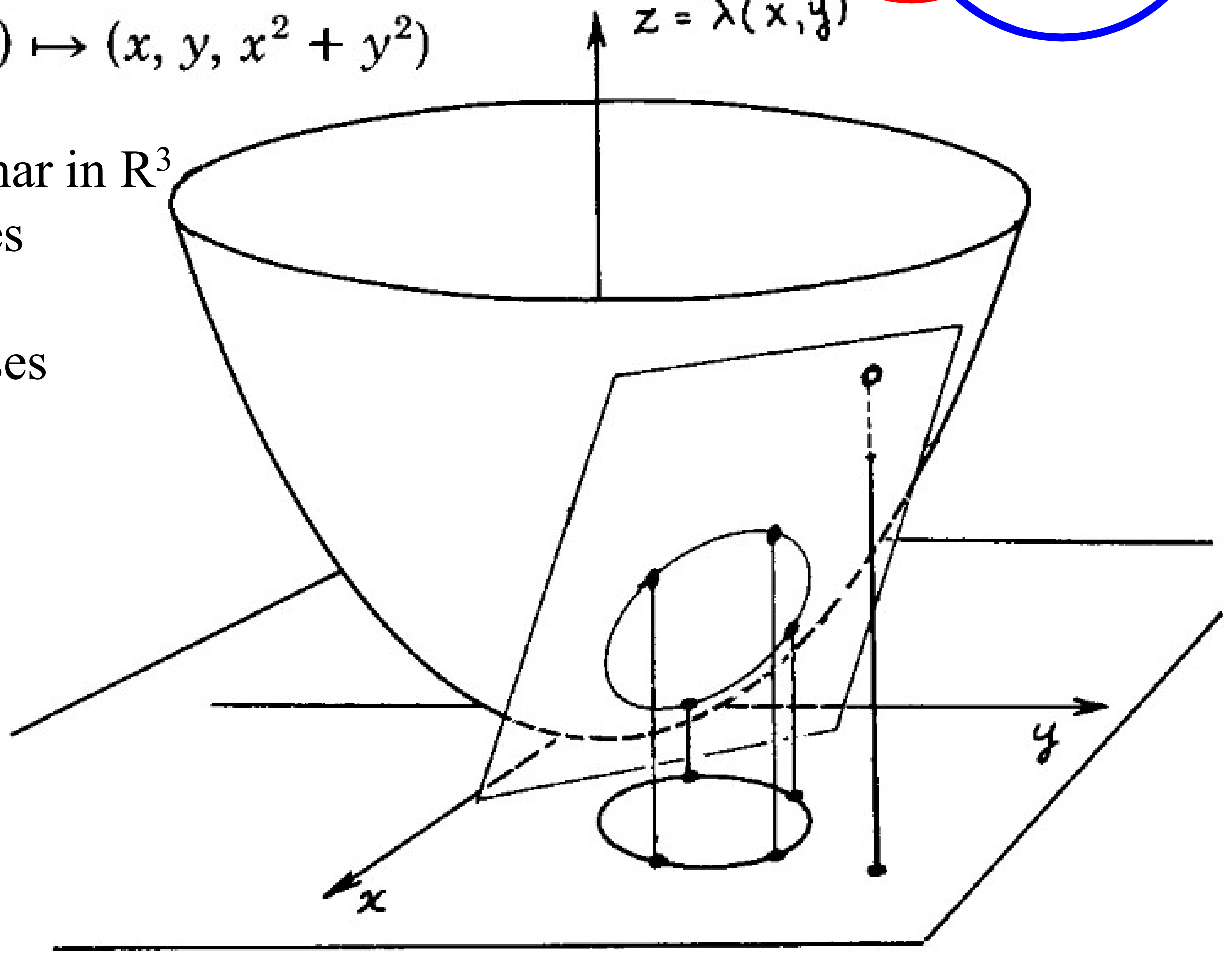
$$\begin{vmatrix} A_x & A_y & A_x^2 + A_y^2 & 1 \\ B_x & B_y & B_x^2 + B_y^2 & 1 \\ C_x & C_y & C_x^2 + C_y^2 & 1 \\ D_x & D_y & D_x^2 + D_y^2 & 1 \end{vmatrix} = \begin{vmatrix} A_x - D_x & A_y - D_y & (A_x - D_x)^2 + (A_y - D_y)^2 \\ B_x - D_x & B_y - D_y & (B_x - D_x)^2 + (B_y - D_y)^2 \\ C_x - D_x & C_y - D_y & (C_x - D_x)^2 + (C_y - D_y)^2 \end{vmatrix} > 0$$

Delaunay-Triangulierung



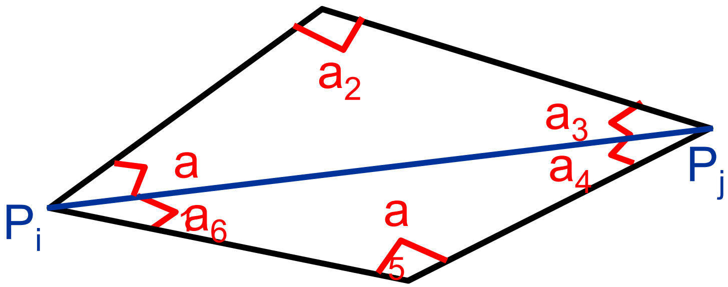
Beweisskizze:

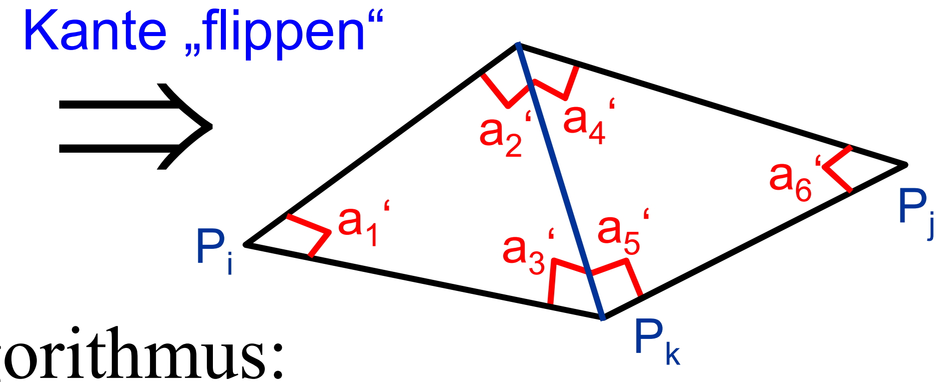
Einbettung in R^3: $\lambda : (x, y) \mapsto (x, y, x^2 + y^2)$



Punkte auf Kreis → koplanar in R^3
Punkt innerhalb des Kreises
→ „südlich“ der Ebene
Punkt ausserhalb des Kreises
→ „nördlich“
Determinante:
  Volumen des Spats
  ABC ccw:
  det > 0
    → D „nördlich“
    → D innerhalb

Delaunay-Triangulierung





Skizze für „naiven“ Algorithmus:

- initiale Triangulierung erstellen
- Alle Kanten betrachten, ob sie geflippt werden müssen (Kreis-Kriterium)
- Falls ja: vom Flip-Viereck aus alle Kanten auf Flippen testen (im schlimmsten Fall: alle)

Komplexität: $\mathbf{O(n^2)}$

Delaunay-Triangulierung

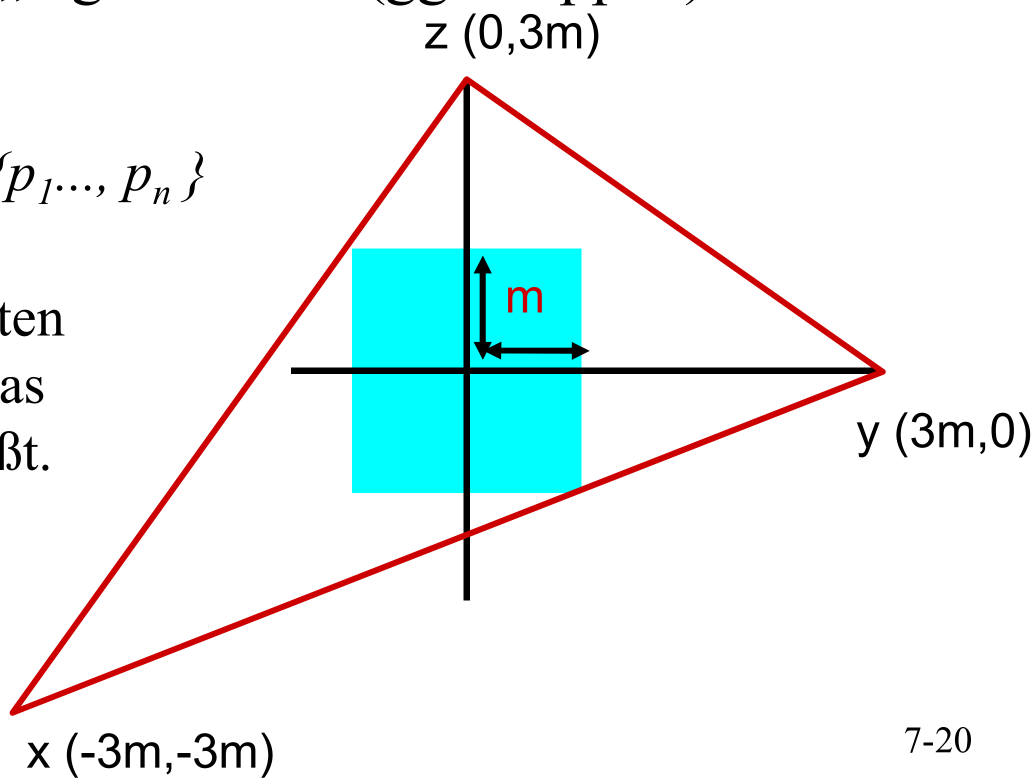
Inkrementeller Algorithmus

Idee:

- Mit umfassendem Dreieck beginnen
- Punkt für Punkt einfügen
- neu entstehende Kanten „legalisieren“ (ggf. flippen)

Gegeben: Punktmenge $P = \{p_1 \ldots, p_n\}$

m: Maximum aller Koordinaten
Berechne Dreieck (x, y, z), das die Punkte $p_1 \ldots, p_n$ umschließt.

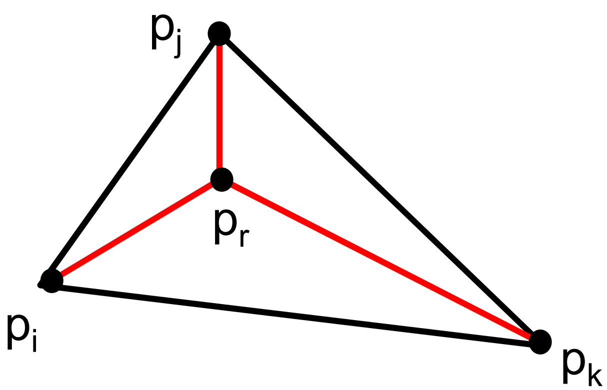


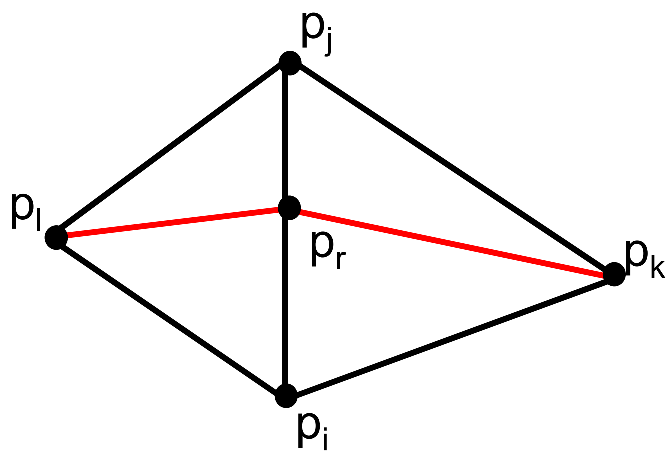
Delaunay-Triangulierung

Inkrementeller Algorithmus

Punkt einfügen:

2 Fälle: p_r liegt innerhalb eines Dreiecks
p_r liegt auf einer Kante

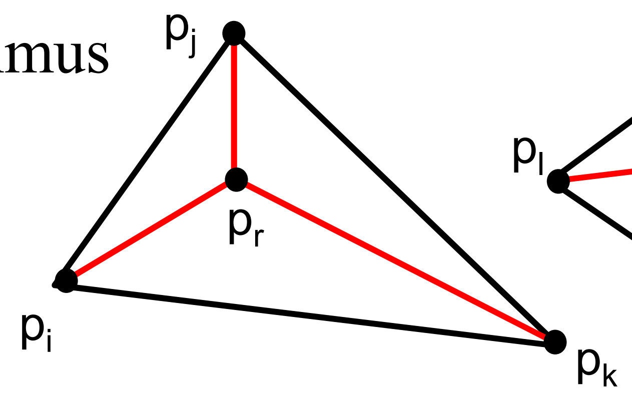


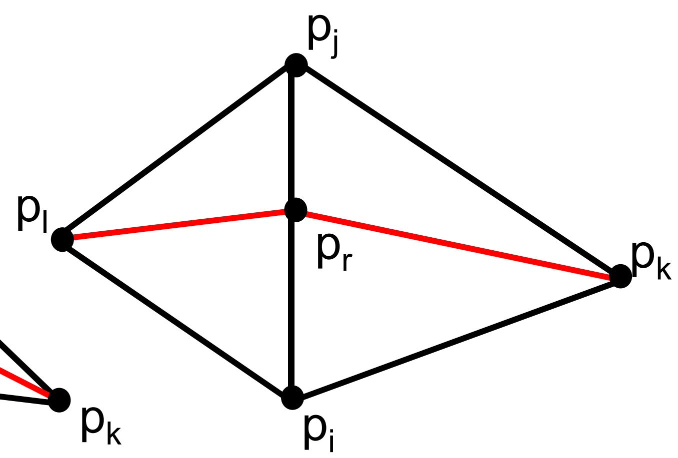


Delaunay-Triangulierung

Inkrementeller Algorithmus

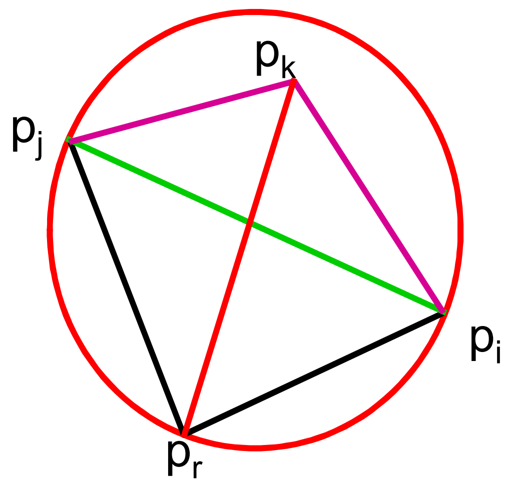
"Legalisieren"





Legalize $(p_r, \overline{p_i p_j}, T)$:
if $\overline{p_i p_j}$ is illegal
then let $p_i p_j p_k$ be the triangle adjacent
to $p_r p_i p_j$ along $\overline{p_i p_j}$.
flip $\overline{p_i p_j}$, i.e. replace $\overline{p_i p_j}$ by $\overline{p_r p_k}$

Legalize $(p_r, \overline{p_i p_k}, T)$
Legalize $(p_r, \overline{p_k p_j}, T)$

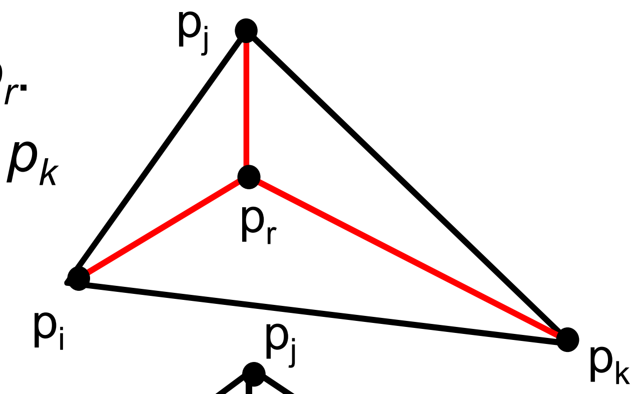


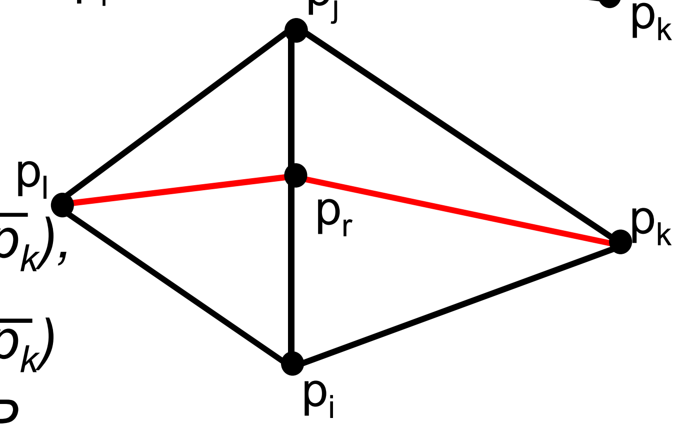
Delaunay-Triangulierung

Inkrementeller Algorithmus

In: Punktmenge $P = \{p_1..., p_n\}$, Out: DT(P) von P

1. $DT(P) = T = (x, y, z)$
2. for $r = 1$ to n do
3. find a triangle $p_i p_j p_k \in T$, that contains p_r.
4. if p_r lies in the interior of the triangle $p_i p_j p_k$
5. then split $p_i p_j p_k$
6. Legalize(p_r, $\overline{p_i p_j}$), Legalize(p_r, $\overline{p_i p_k}$), Legalize(p_r, $\overline{p_j p_k}$)
7. if p_r lies on an edge of $p_i p_j p_k$ (say $\overline{p_i p_j}$)
8. then split $p_i p_j p_k$ and $p_i p_j p_l$
   Legalize (p_r, $\overline{p_i p_l}$), Legalize (p_r, $\overline{p_i p_k}$),
   Legalize (p_r, $\overline{p_j p_l}$), Legalize (p_r, $\overline{p_j p_k}$)
9. Delete (x, y, z) with all incident edges to P

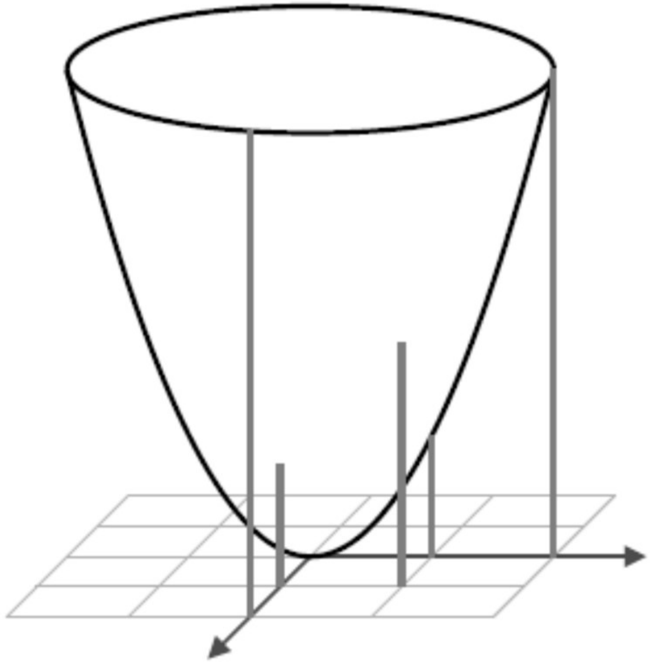




Delaunay-Triangulierung

Alternative: Einbettung in R^{n+1}

Zusammenhang mit konvexer Hülle



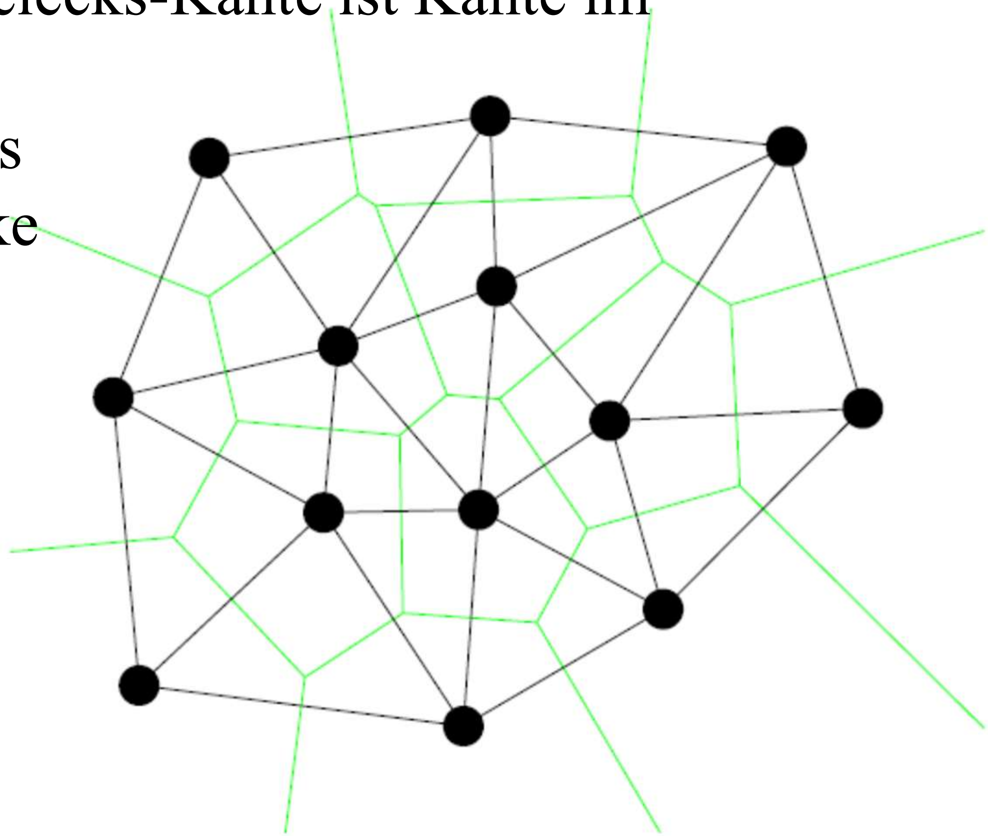
Idee:
Punkte (x1,…xn) auf Paraboloid projizieren
$f:(x_1,\ldots x_n) \rightarrow (x_1,\ldots,x_n,\Sigma x_i^2)$
Konvexe Hülle in R^{n+1} bilden (!!!)
Nach unten zeigende Facetten der konvexen Hülle nach R^n rückprojizieren

`s. auch http://www.pi6.fernuni-hagen.de/GeomLab/VoroGlide/`

Voronoi-Diagramm

Zusammenhang mit Delaunay-Triangulierung

Voronoi-Diagramm ist dualer Graph
- Mittelsenkrechte einer Dreiecks-Kante ist Kante im Voronoi-Diagramm
- Umkreis-Mittelpunkt eines Delaunay-Dreiecks ist Ecke im Voronoi-Diagramm



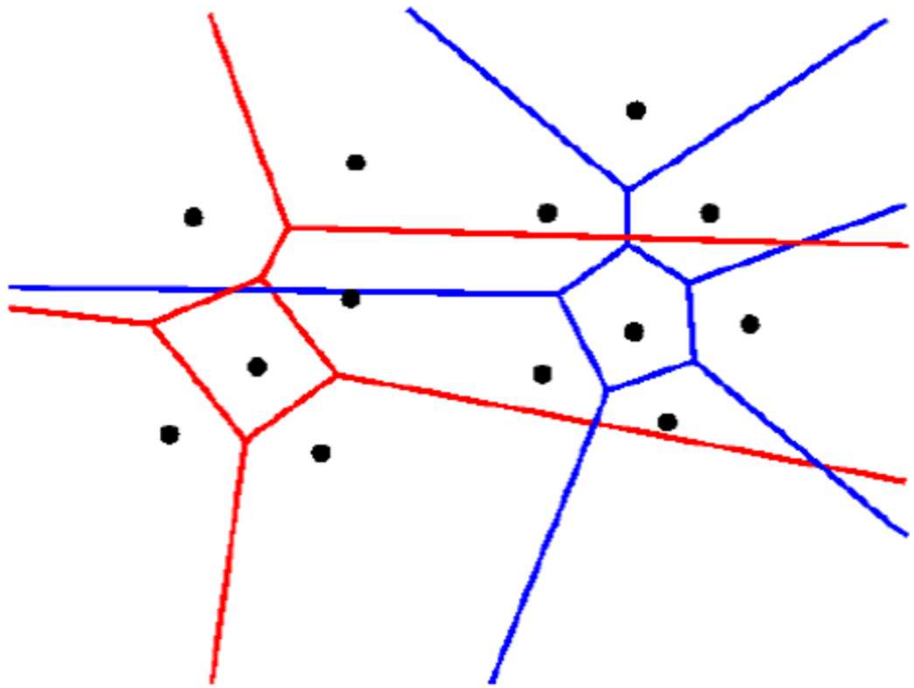
Voronoi-Diagramm

Algorithmen zur Berechnung des Voronoi-Diagramms
- Line-Sweep-Algorithmus
- Divide-and-Conquer-Algorithmus

Voronoi-Diagramm

Divide-and-Conquer-Algorithmus

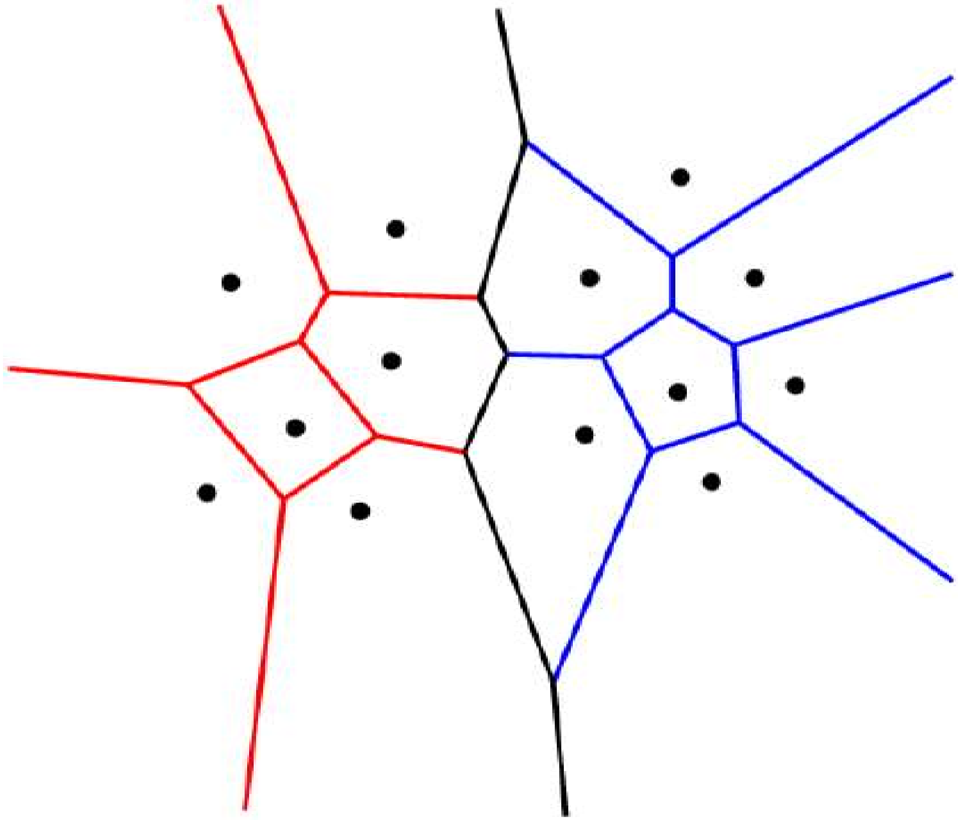
- Partitionierung der Punktmenge entlang Trenngerade
- Rekursive Erzeugung primitiver Voronoi-Diagramme



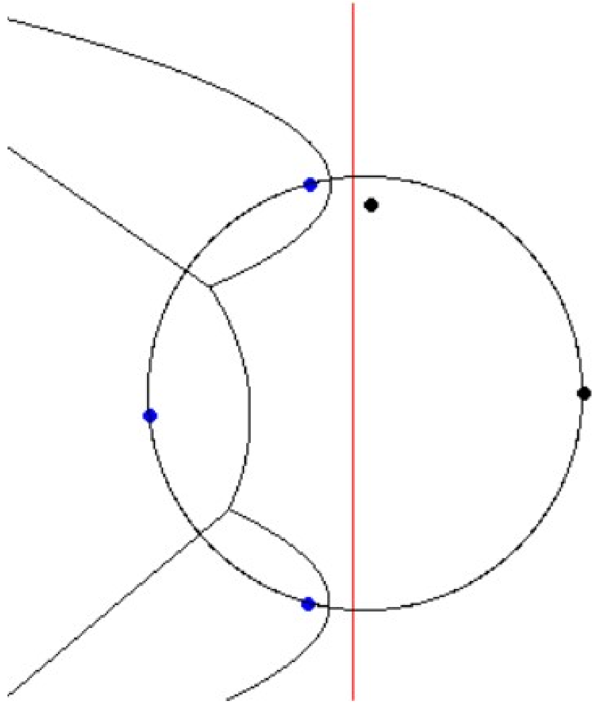
Voronoi-Diagramm

Divide-and-Conquer-Algorithmus

- Verschmelzen je zweier Voronoi-Diagramme



Voronoi-Diagramm



Fortune's Algorithmus (S. Fortune, 1986)
Line-Sweep-Algorithmus

„Beach Line": Sequenz von Parabelsegmenten

- Parabel: Voronoi-Diagramm zwischen Punkt und Linie
- Schnitte der Parabeln „zeichnen" Voronoi-Diagramm
- Events:
  - „Circle Event": Umkreis dreier Punkte wird „gültig"
  - „Site Event": Neuer Punkt führt neues Parabelsegment ein
- Komplexität: O(n log(n))

https://www.desmos.com/calculator/ejatebvup4
https://www.algovision.org/Algovision/pool.html
https://jacquesheunis.com/post/fortunes-algorithm
https://blog.ivank.net/fortunes-algorithm-and-implementation.html