Technische Universität Braunschweig

Einführung in algorithmische Geometrie – Visibility

Arne Schmidt

Heute

Für Art Gallery müssen wir berechnen, was ein Wächter sieht

Welche Techniken gibt es?

Was funktioniert am besten?

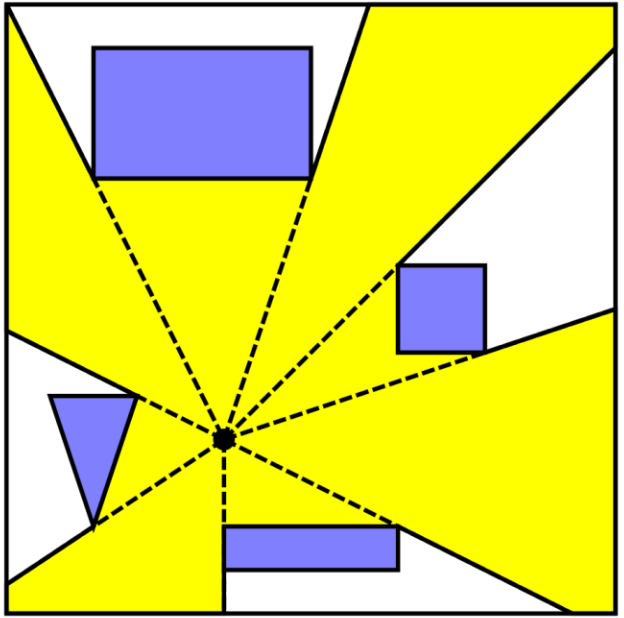
Kapitel 4.3 – Sichtbarkeitspolygone

Definition (Wiederholung)

Definition 4.1
Gegeben sei ein Polygon P und ein Punkt $v \in P$.

Das **Visibilitypolygon** $\mathcal{V}_P(v)$ von v ist definiert über alle Punkte $p \in P$, sodass $\overline{vp} \subset P$.

Lemma 4.11
Das Visibilitypolygon ist ein einfaches Polygon.



Zeitkomplexität

Lemma 4.12
Sei P ein Polygon mit n Ecken und h Löchern. Jeder Algorithmus zur Bestimmung eines Visibilitypolygons benötigt $\Omega(n + h \log h)$ Zeit.

Beweis:
Das Visibilitypolygon kann Komplextität $\Omega(n)$ besitzen, daher benötigt jeder Algorithmus $\Omega(n)$ Zeit.

Für den zweiten Teil reduzieren wir das Sortierproblem auf unser Problem in Linearzeit. Könnten wir in $o(h \log h)$ Zeit das Visibilitypolygon bestimmen, könnten wir in Zeit $o(h \log h)$ sortieren. Ein Widerspruch.

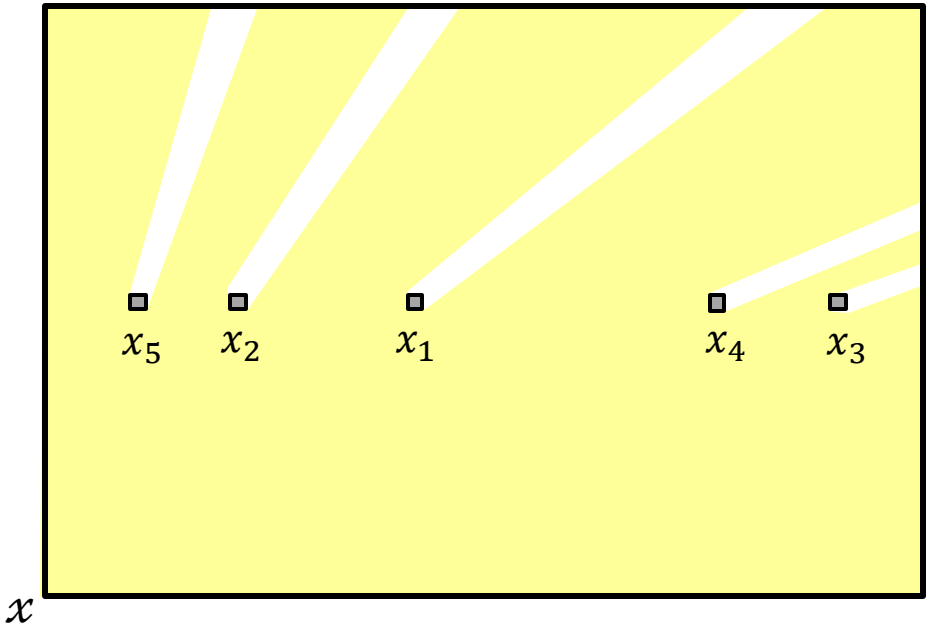
Reduktion: Sortieren auf Visibility

Sei $\{x_1, \dots, x_n\} \subset \mathbb{N}$.

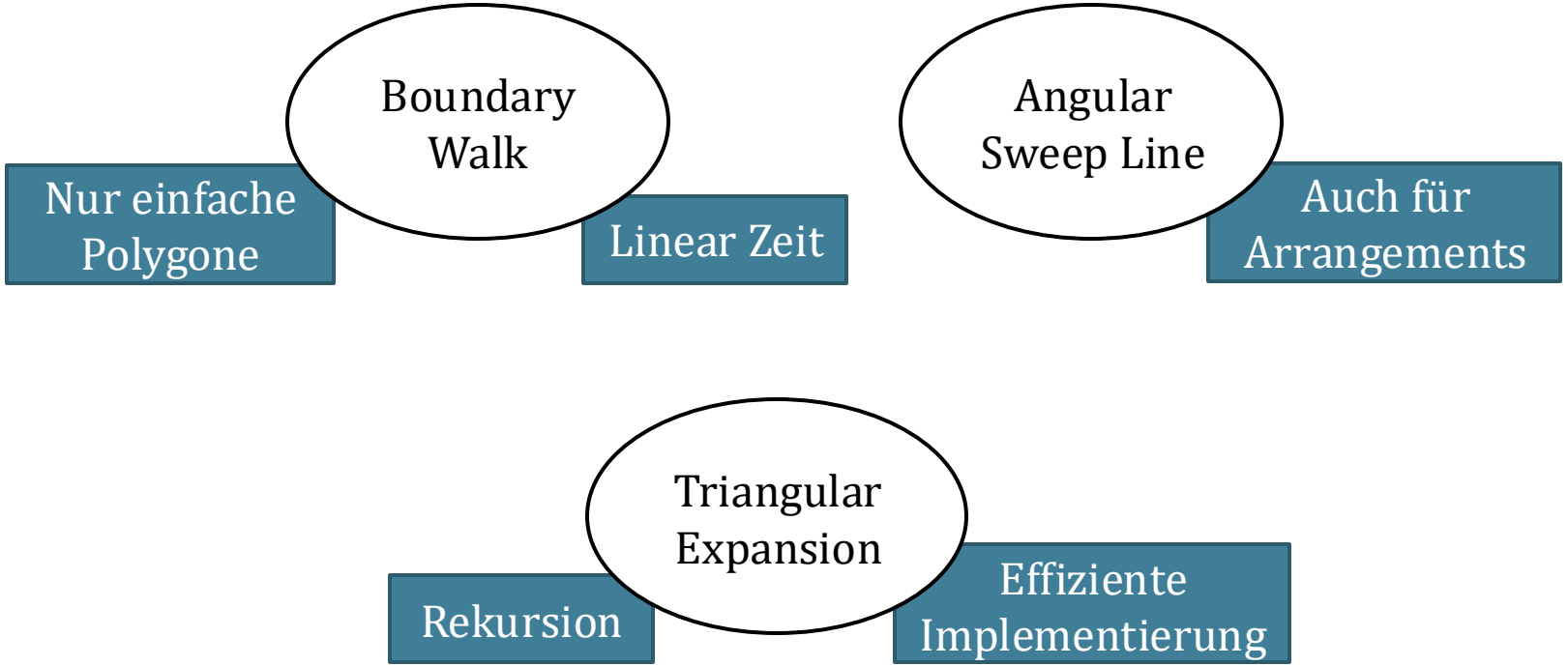
Betrachte das Polygon mit

- Äußerer Hülle
  $\left(-1, -\frac{\Delta}{2}\right), \left(\Delta, -\frac{\Delta}{2}\right), \left(\Delta, \frac{\Delta}{2}\right), \left(-1, \frac{\Delta}{2}\right)$
- Löchern
  $(x_i, 0) \pm (\varepsilon, \varepsilon)$

Berechne nun $\mathcal{V}\left(x = \left(-1, -\frac{\Delta}{2}\right)\right)$



Drei Algorithmen



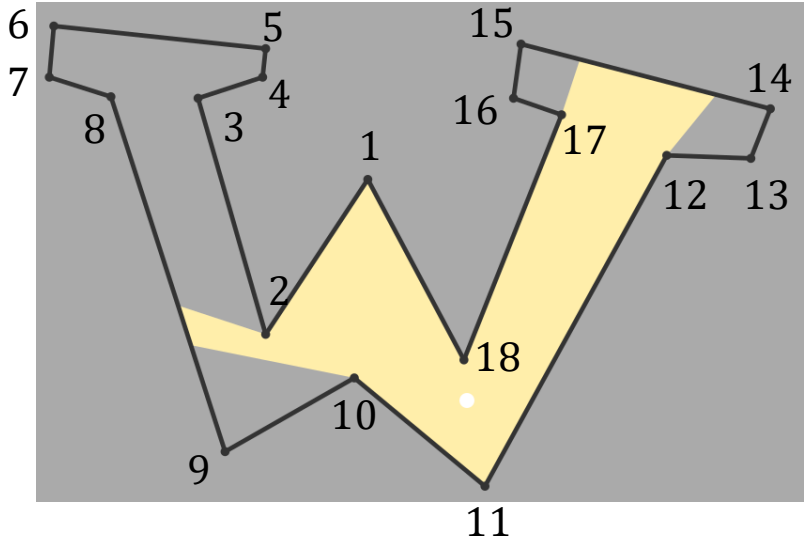
Kapitel 4.3.1 – Boundary Walk

Idee 1

Die Ecken des Polygons sind aufsteigend nummeriert.

In dieser Reihenfolge laufen wir auch die Ecken von $\mathcal{V}_P$ ab.

Laufe den Rand ab und halte einen Stack, welches $\mathcal{V}_P$ nach Ablaufen der ersten i Knoten beschreibt.



Idee 2: CW-Kanten

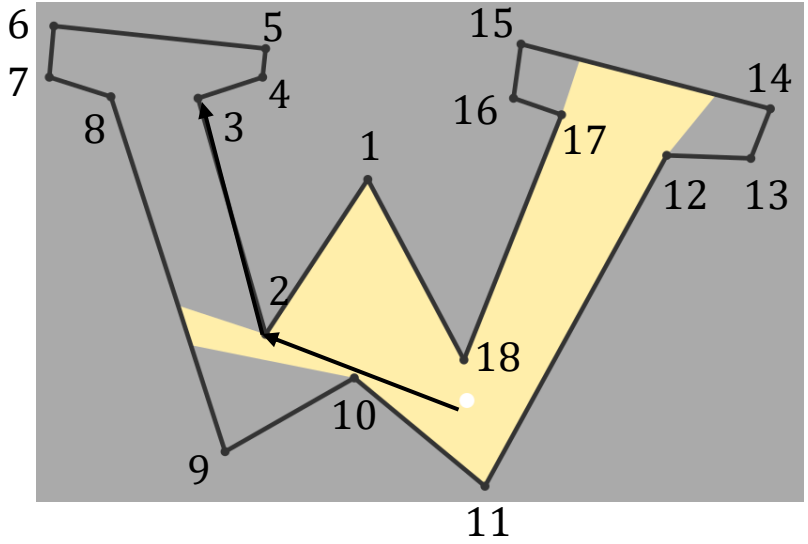
Kante 2-3 bildet mit dem Sichtpunkt einen Rechtsknick.

Diese Kante kann kein Teil von $\mathcal{V}_P$ sein!

Zwei Situationen:

- Die Kante schränkt die Sicht des aktuellen Polygons auf dem Stack ein.
  - → Lösche Knoten vom Stack, bis wieder alles sichtbar ist
- Die Kante schränkt die Sicht nicht ein.
  - → Suche entlang des Polygons bis wieder eine Kante sichtbar ist.

$\mathcal{V}_P$ Knoten auf dem Rand werden dabei dem Stack hinzugefügt.



Details

Visibility of a Simple Polygon*

D. T. LEE

Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, Illinois 60201

Received September 18, 1981; revised March 8, 1982

The hidden line problem in the plane is studied and an optimal algorithm for finding the visibility polygon of a point inside or outside a simple polygon is given. The algorithm uses only one stack, instead of three as used in a previously known algorithm by El-Gindy and Avis (*J. Algor.* **2**, 1981, 186–197). As a result, the algorithm is simpler to implement and easier to understand and its correctness can be easily verified.

CORRECTIONS TO LEE'S VISIBILITY POLYGON ALGORITHM

B. JOE and R. B. SIMPSON

Dept. of Computing Science, University of Alberta, Edmonton, Alberta, Canada T6G 2H1

Dept. of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

Abstract.

We present a modification and extension of the (linear time) visibility polygon algorithm of Lee. The algorithm computes the visibility polygon of a simple polygon from a viewpoint that is either interior to the polygon, or in its blocked exterior (the cases of viewpoints on the boundary or in the free exterior being simple extensions of the interior case). We show by example that the original algorithm by Lee, and a more complex algorithm by El Gindy and Avis, can fail for polygons that wind sufficiently. We present a second version of the algorithm, which does not extend to the blocked exterior case.

Theorem 4.13
Das Visibilitypolygon in einfachen Polygonen kann in Zeit $O(n)$ berechnet werden.

Kapitel 4.3.2 – Angular Sweep Line

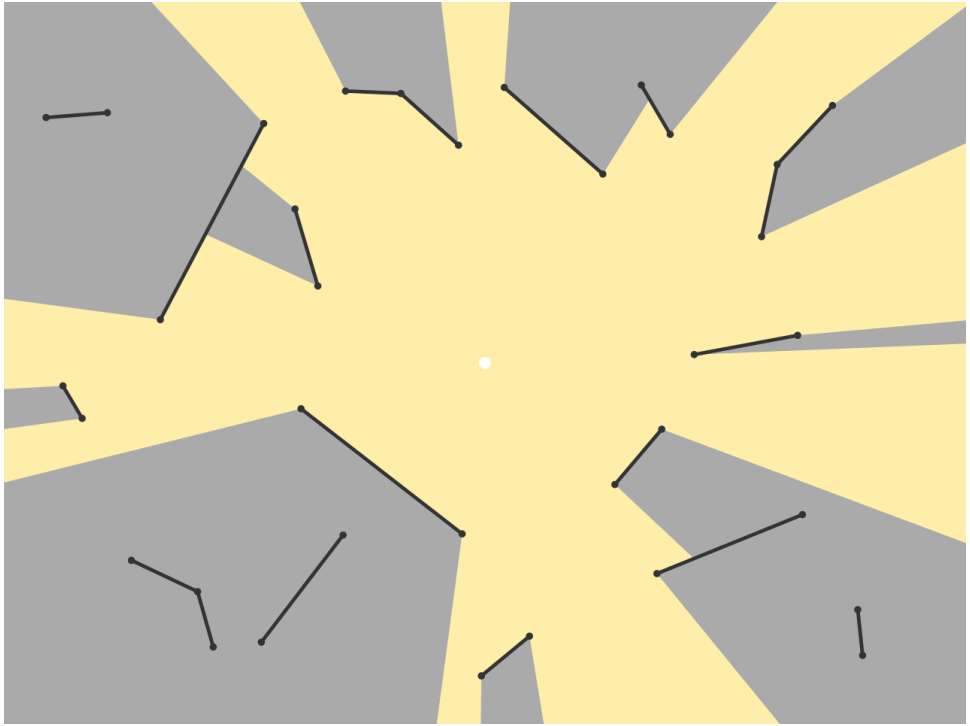
Sweep-Line Revisited

Lasse eine Linie in geeigneter Form die Ebene durchlaufen / rotieren / …

Währenddessen:

- Bearbeite Event Points.
- Füge ggf. Event Points hinzu.
- Verwalte eine Datenstruktur, um Event Points effizient zu bearbeiten.
- Verwalte eine Datenstruktur für die Ausgabe.

Wir nehmen im folgenden an, dass wir nur eine Menge von nicht-schneidenen Segmenten erhalten.



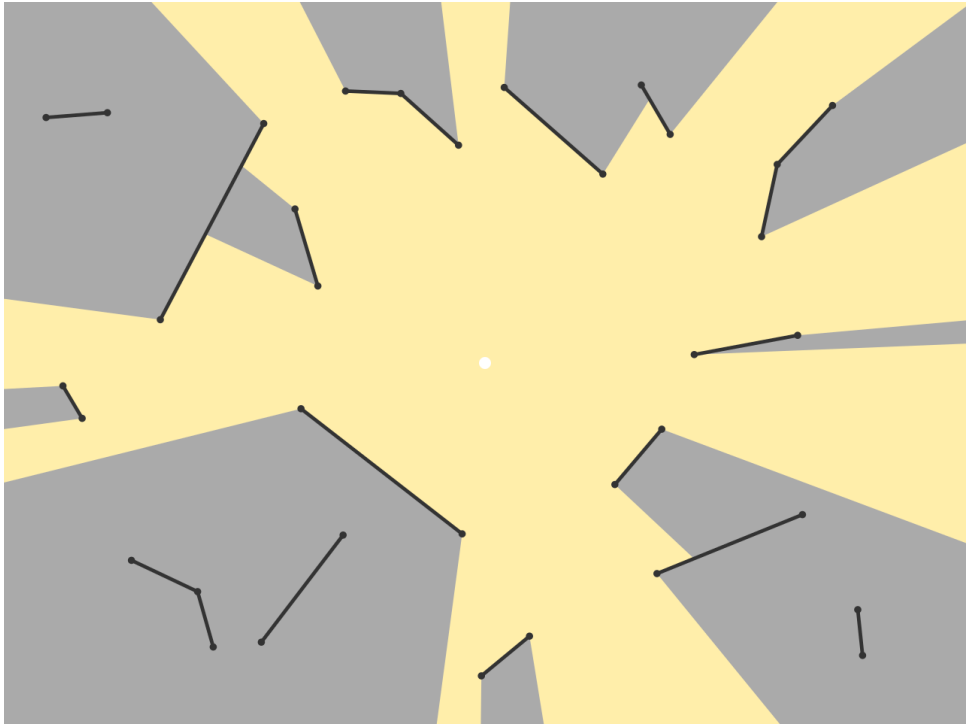
Event-Points

Was sind für Visibilitypolygone die Event Points?

Die Endpunkte der Segmente!
Nur dort kann sich das Visibilitypolygon verändern.

Unterscheide zwei Fälle:

- Es starten neue Segmente.
- Es enden Segmente

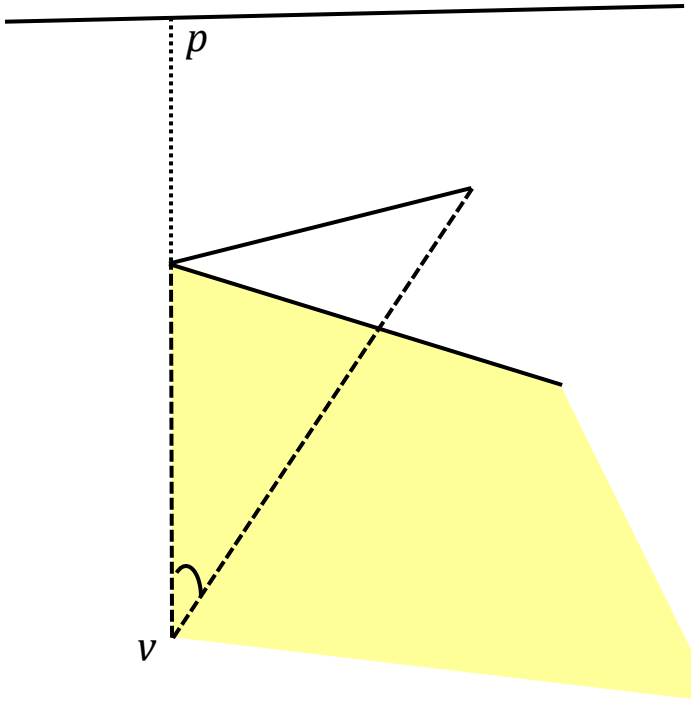


Segmente enden

Enden nur Segmente im Event Point, fallen wir auf eine Kante zurück, die weiter hinten liegt.

Wie erhalten wir das nächstliegende Segment?

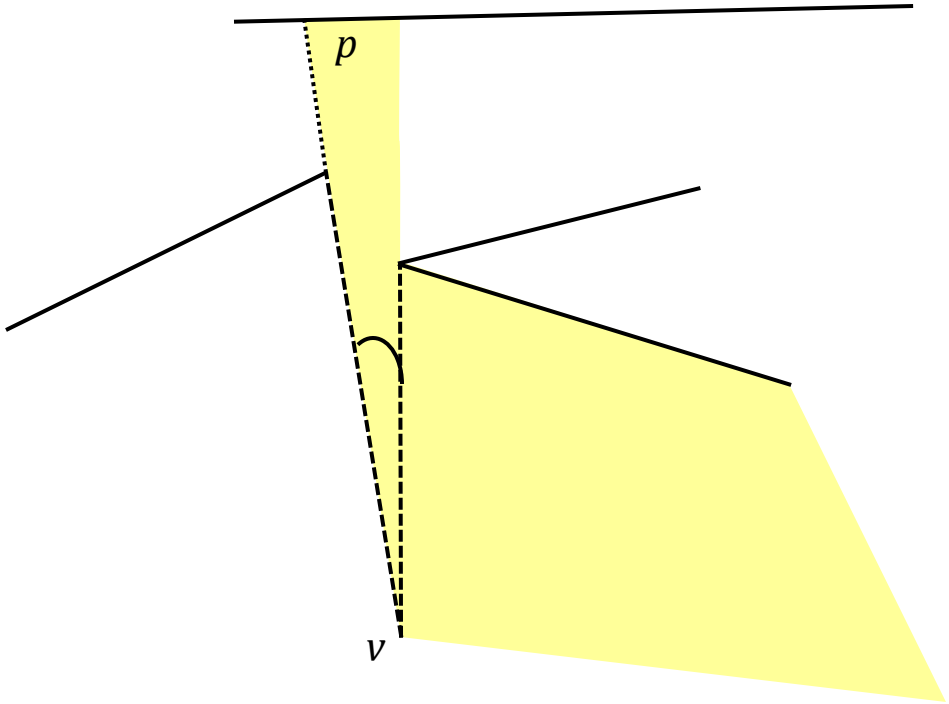
Speichere Segmente in einem balancierten Suchbaum bzgl. Distanz zu v!



Segmente beginnen

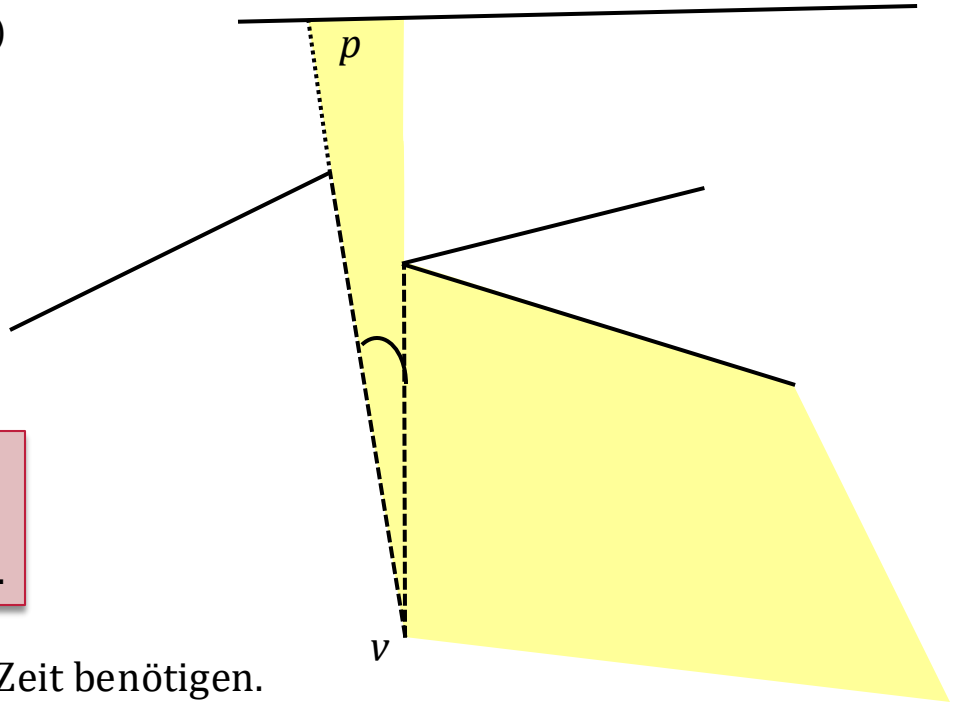
Beginnt ein Segment, tauchen zwei Fälle auf:

1. Es ist nicht sichtbar:
   Füge alle beginnenden Segmente in den Suchbaum ein.
2. Es ist sichtbar!
   a) Bestimme letzten sichtbaren Punkt p auf zuletzt gesehendem Segment.
   b) Füge p und Endpunkt zum Polygon hinzu.



Laufzeit

- Endpunkte im Kreis sortieren: $O(n \log n)$
- n Event Points:
  - Segmente in Datenstruktur hinzufügen / löschen: $O(\log n)$
  - Visibilitypolygon erweitern / nächstes Segment finden: $O(\log n)$



Theorem 4.14
Das Visibilitypolygon kann in worst-case-optimaler Zeit $O(n \log n)$ berechnet werden.

Es gibt Algorithmen, die nur $O(n + h \log h)$ Zeit benötigen.

n + h log h

SIAM J. COMPUT.
Vol. 24, No. 1, pp. 184–201, February 1995

012

AN OPTIMAL ALGORITHM FOR COMPUTING VISIBILITY IN THE PLANE*

PAUL J. HEFFERNAN† AND JOSEPH S. B. MITCHELL‡

Abstract. The authors give an algorithm to compute the visibility polygon from a point among a set of h pairwise-disjoint polygonal obstacles with a total of n vertices. The algorithm uses $O(n)$ space and runs in optimal time $\Theta(n + h \log h)$, improving the previous upper bound of $O(n \log n)$. A direct consequence of the algorithm is an $O(n + h \log h)$ time algorithm for computing the convex hull of h disjoint simple polygons.

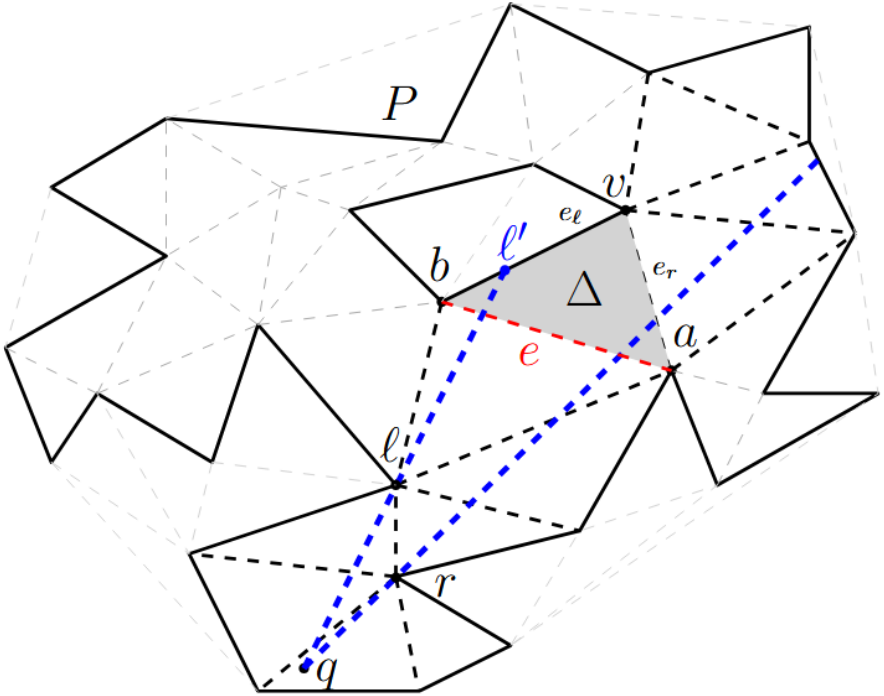
Kapitel 4.3.3 – Triangular Expansion

Idee – Triangulation

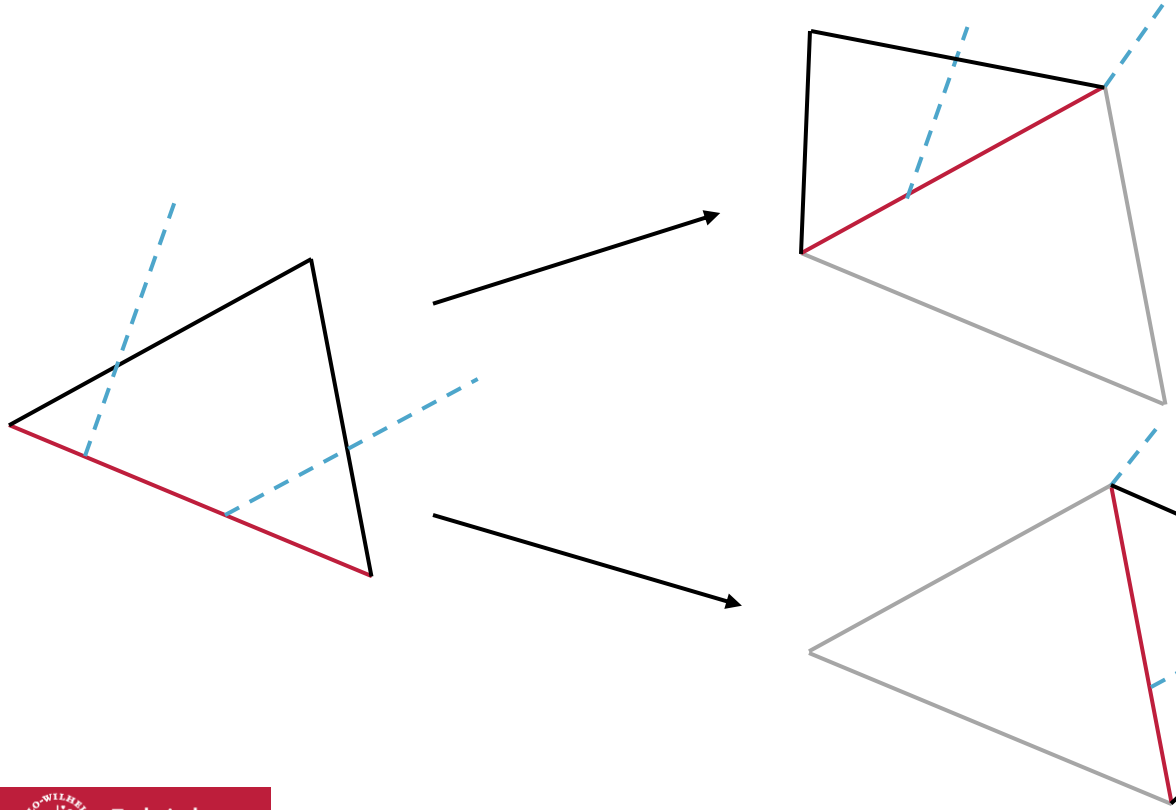
Nutze eine Triangulation, um das Visibilitypolygon nach und nach aufzubauen.

Dafür eignet sich Rekursion:

- Ein Dreieck wird mit einem Sichtbereich über die Kante betreten.
- Bestimme, bei welchen benachbarten Dreiecken wir mit welchem Sichtbereich weiter machen



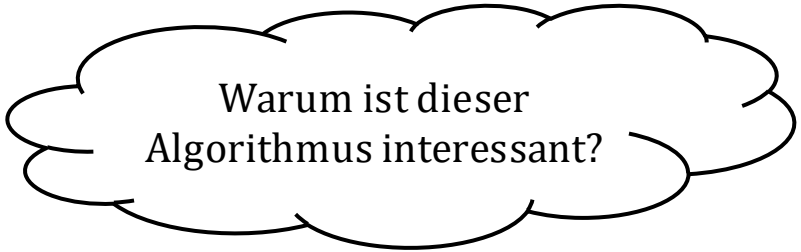
Rekursionsschritt



Wichtig:
Die Rekursion bricht ab, wenn die entsprechende Kante vom Dreieck dem Rand des Polygons gehört.

Laufzeit: Triangular Expansion

Theorem 4.15
Triangular Expansion berechnet in triangulierten Polygonen ein Visibilitypolygon in Zeit $O(nh)$, wobei n die Anzahl der Ecken und h die Anzahl an Löchern im Polygon sind.



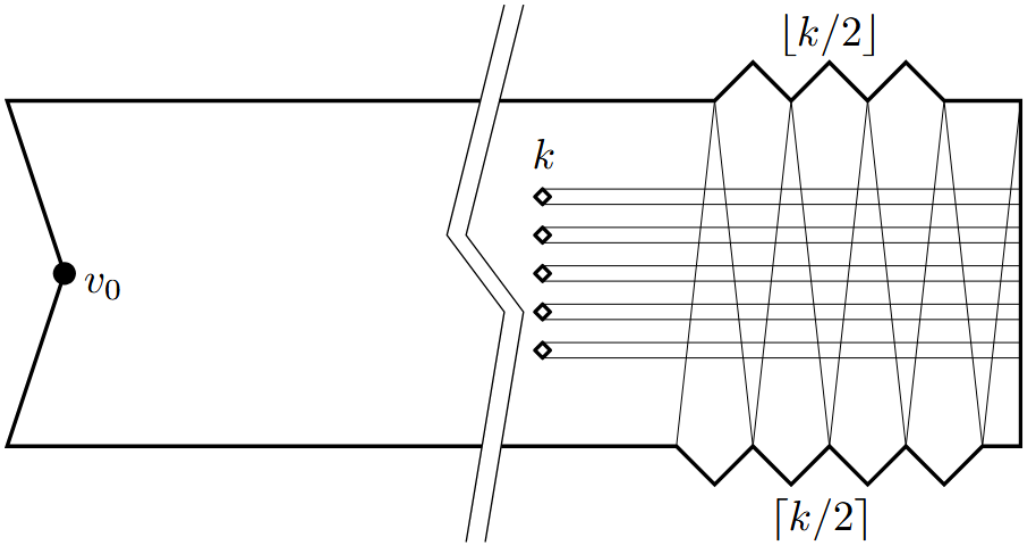
Alle anderen Algorithmen müssen immer das gesamte Polygon durchlaufen.

Hier müssen nur notwendige Dreiecke durchlaufen werden!

Worst-Case-Beispiel

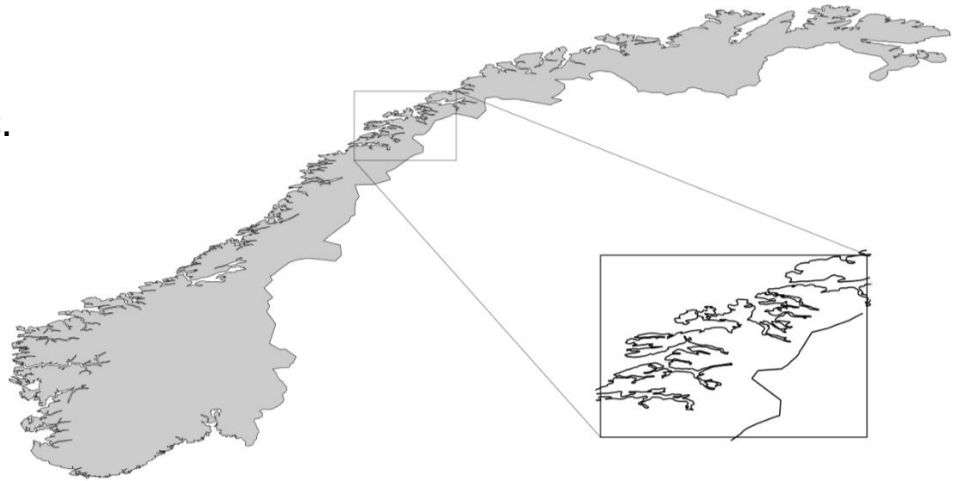
Dreiecke können im Worst-Case sehr oft rekursiv aufgerufen werden.

→ Worst-Case-Lauftzeit ist in $\Omega(n^2)$



Algorithmen in der Praxis

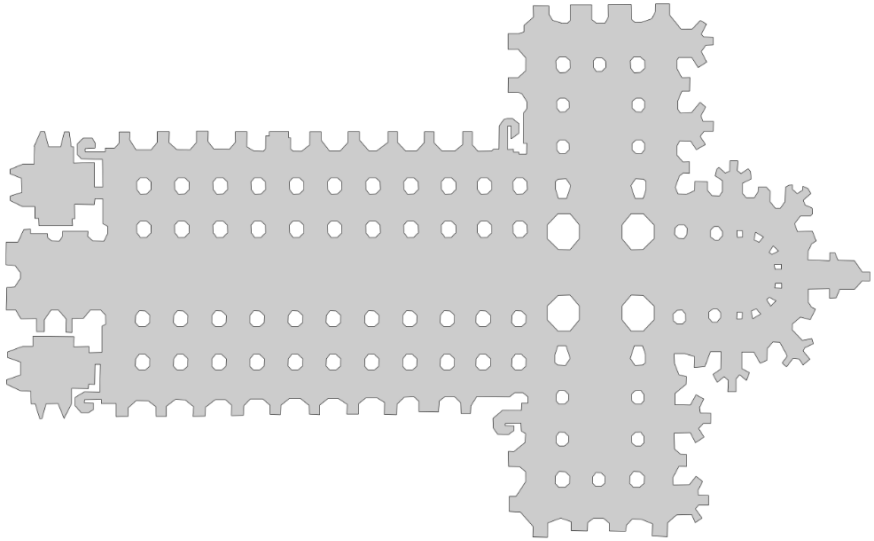
Aufgabe:
Berechne das Visibilitypolygon von jedem der 20981 Knoten des einfachen Polygons.



Alg.	T_{PrePro}	$T_{Queries}$	T_{Total}	Avg
S	—	117.43 s	117.43 s	5.60 ms
R	—	1193.29 s	1193.29 s	56.87 ms
T	0.21 s	3.66 s	3.88 s	0.18 ms

Algorithmen in der Praxis

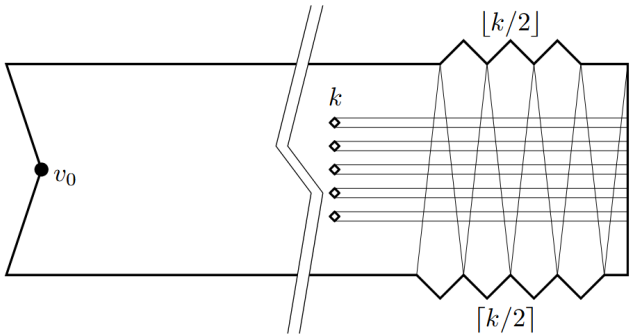
Aufgabe:
Berechne das Visibilitypolygon von jedem der 1209 Knoten des Polygons.

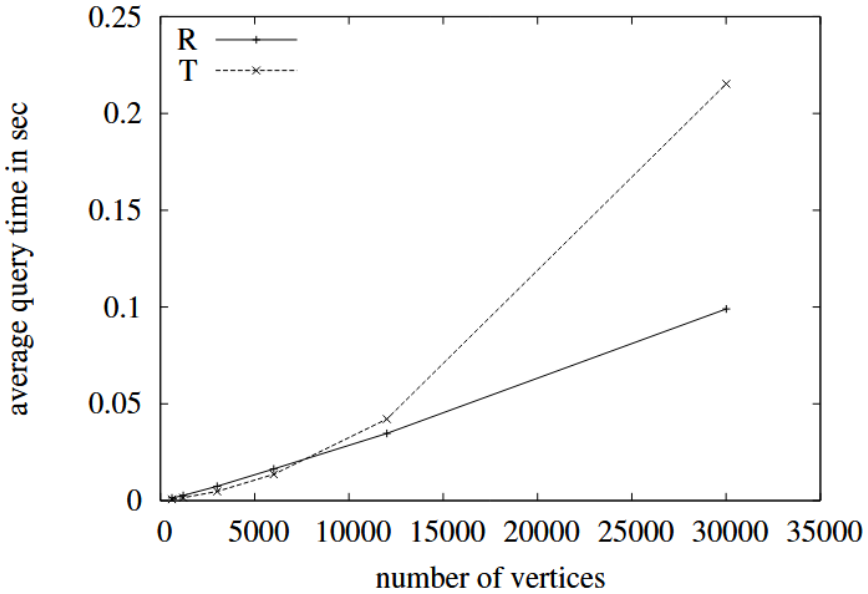


Alg.	T_{PrePro}	$T_{Queries}$	T_{Total}	Avg
R	—	1.35 s	1.35 s	1.112 ms
T	0.004 s	0.04 s	0.04 s	0.004 ms

Algorithmen in der Praxis

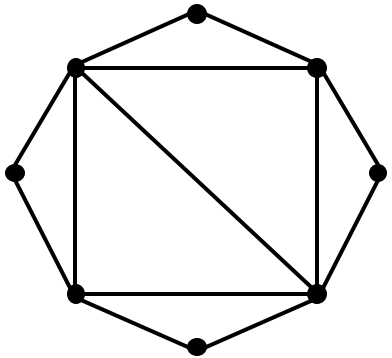
Aufgabe:
Berechne das Visibilitypolygon von jedem Knoten des Worst-Case-Polygons.

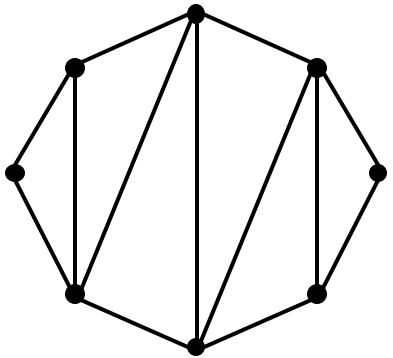




Können wir einfach anders triangulieren?

Varianten von Triangulationen





Verschiedene Zielfunktionen:

- Minimiere das Gewicht der Triangulation (Länge der Kanten)
  - $O(n^3)$ Algorithmus in einfachen Polygonen
  - NP-schwer für Polygone mit Löchern
- Maximiere den kleinsten Innenwinkel ([Constrained] Delaunay-Triangulation)
  - $O(n \log n)$ Algorithmus in einfachen Polygonen

Wrap-Up

Visibilitypolygone können in worst-case-optimaler Zeit berechnet werden.

Für die Praxis macht es uU Sinn, theoretisch schlechtere Algorithmen zu nutzen

Nächste Woche

Zusammenfassung, Klausurvorbereitung und Fragestunde!

Fragen und/oder Wünsche bitte bis **Mittwoch, 16.06. um 12 Uhr** per Mail schicken.