Overlays

Idee: Nutze Sweepline wie für Line-Intersections.

→ Event-Queue: Alle Segmente
Q

→ Status-Datenstruktur: Bal. Suchbaum von Segmenten
T

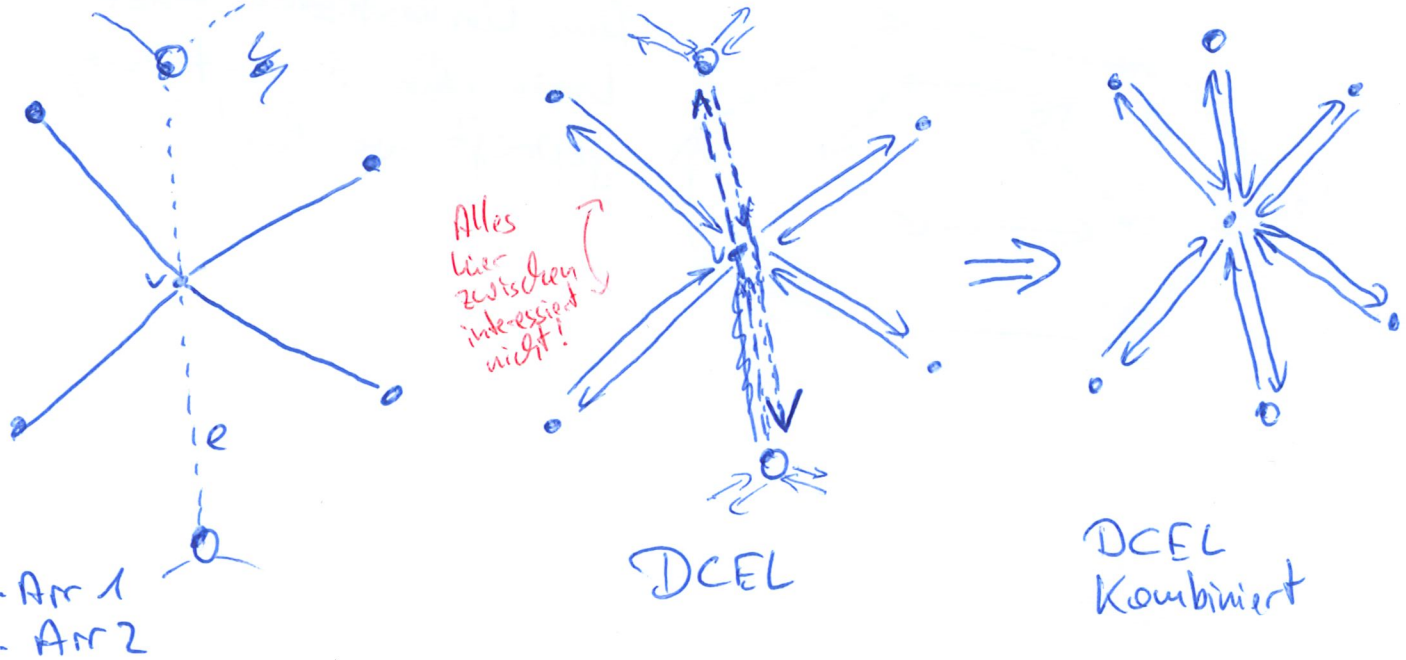
Nun kommt hinzu

→ Overlay-Datenstruktur: DCEL (zu Beginn Kopie von A_1 und A_2
D

Während des Sweeps berechnen wir zunächst nur die Knoten und Kanten Informationen. Die Faces betrachten wir später.

Was passiert während des Sweeps?

- Wie gehabt, wenn nur Kanten eines einzigen Arr. betrachtet werden
- Wir müssen D aktualisieren, wenn Kanten von beiden Arr. betrachtet werden.



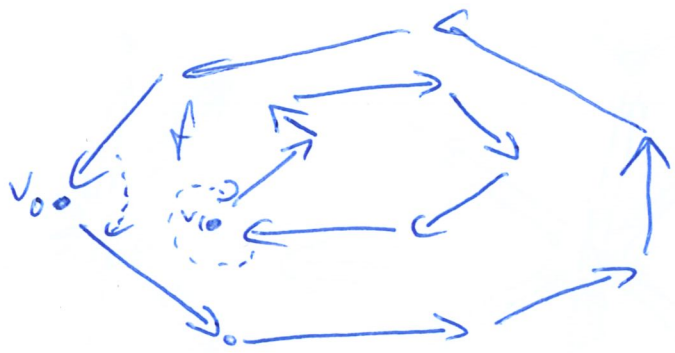
- Splitte Halbkanten zu e in je zwei weitere
  ↳ 4 Halbkanten
- Twins sind einfach gesetzt
- Wie ist das mit prev()/next()?
  ↳ Endpunkte von e können einfach aktualisiert werden

  ↳ um v:
  $\Omega(\delta(v))$ Zeit um prev/next zu aktualisieren
  (zwischen welchen Halbkanten liegen die neuen Kanten von v?)

  Da $\sum_{v \in V} \delta(v) \in \mathcal{O}(n)$ mit n Komplexität von S_1 und S_2 ändert das die asymptotische Laufzeit nicht.

Betrachte nun die Faces.

- Jedes Face (außer das unbounded face) besitzt eine eindeutige Boundary!
  → dies beschreibt das Face
- Jeder Kreis von Halbkanten sagt uns, ob sie ein Face begrenzen oder ein Loch erzeugen.



Am linkesten Knoten kann die Orientierung geprüft werden!

Betrachte folgenden Graphen $\mathcal{G}$.

- Knoten entsprechen Kreisen von Halbkanten (und ein imaginärer Kreis für das unbounded face)
- ~~Kanten~~ sind verbunden, wenn vom linkesten Knoten v Knoten c_1, c_2 von c_2 eine Kante von c_1 als erstes getroffen wird, sobald man einen Strahl von v nach links schießt.

[Bild Slides]
→ Erzeugt einen Wald

Dieser Graph kann während des plane sweeps erstellt werden!

Setzen der Face-Informationen ist dann nur noch Ablaufen und Setzen der Infos entlang aller Kanten, über die Komponenten des Graphs.