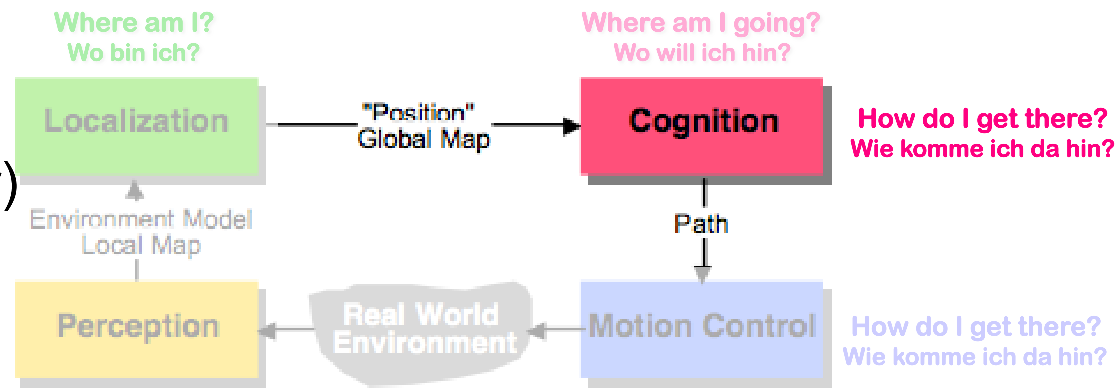
7.4 Pfadplanung

- **Gegeben**
  - Karte der Umgebung (Geometrie, Belegungsraster)
  - Startpose entspr. Karte
  - Zielposition oder -pose
- **Finde** den „besten“ Weg



… ist übersetzt in **Graphsuchproblem**
super-klassisches Problem aus Informatik, KI (heuristische Suche!), OR, …

- Daher hier nur Abriss
- Fokus auf Repräsentation d. Suchraums
- Plane kompletten Pfad! Für Ausführung halte Zwischenzielpunkte

***Jede* Art Planung erfordert**

- Information über Umgebung (evtl. nicht vollständig, nicht sicher)
- Stabilität, Gesetzmäßigkeit der Umgebung (evtl. nicht vollständig, nicht sicher)

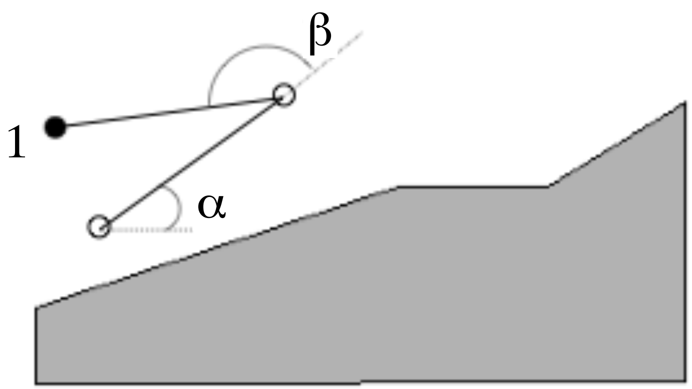
Arbeitsraum und Konfigurationsraum

- **Arbeitsraum** (*work space*): Der physische (geometrische, kartesisch modellierte) Raum, in dem d. Roboter sich bewegt
  - normalerweise 2D oder 3D
  - „das, was in einer Geometriekarte eingezeichnet ist“
- **Konfigurationsraum** (*configuration space*): Parameterraum, welcher mögliche **Konfigurationen** (Ort jedes Punktes des Roboterkörpers) des Roboters im Referenzsystem beschreibt
  - nicht eindeutig! (komplementäre K-Räume für 1 Roboter denkbar)
  - Pose (3D,6D) üblicher K-Raum für mobile Roboter
  - für Automations-Robotik üblich:
    K-Raum-Parameter = Freiheitsgrade

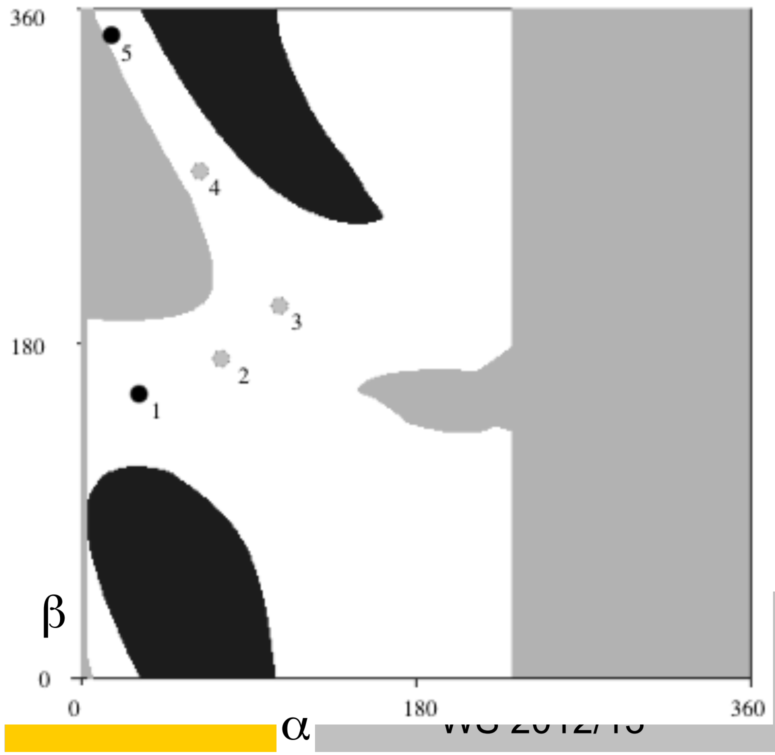
NB: Ist #effektive Freiheitsgrade < Dimensionalität des K-Raums, ist der Roboter nicht holonom!

Arbeits- und Konfigurationsraum, Bspl.

[redacted]



- 2 Freiheitsgrade (Schulter-, Ellenbogengelenk)
- Wände, Boden, Hindernis begrenzen Bewegung: **K-Raum-Hindernisse**
- Handpose hängt zusätzlich ab von Robotergeometrie (Länge Ober-, Unterarm)



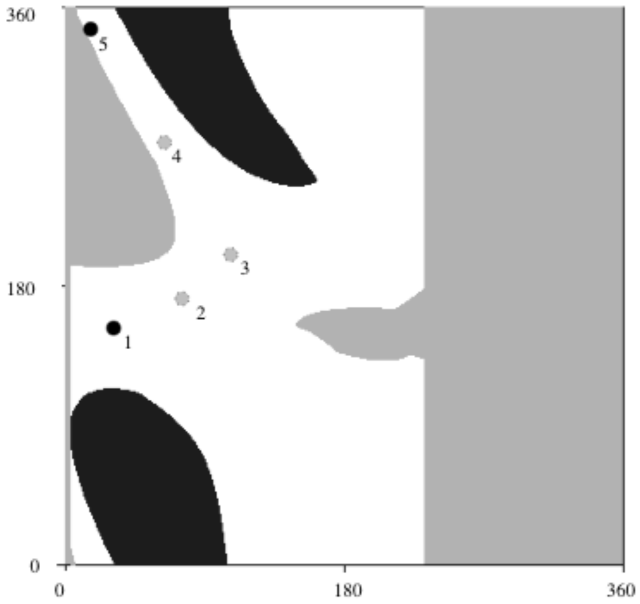
2D-Konfigurationsraum

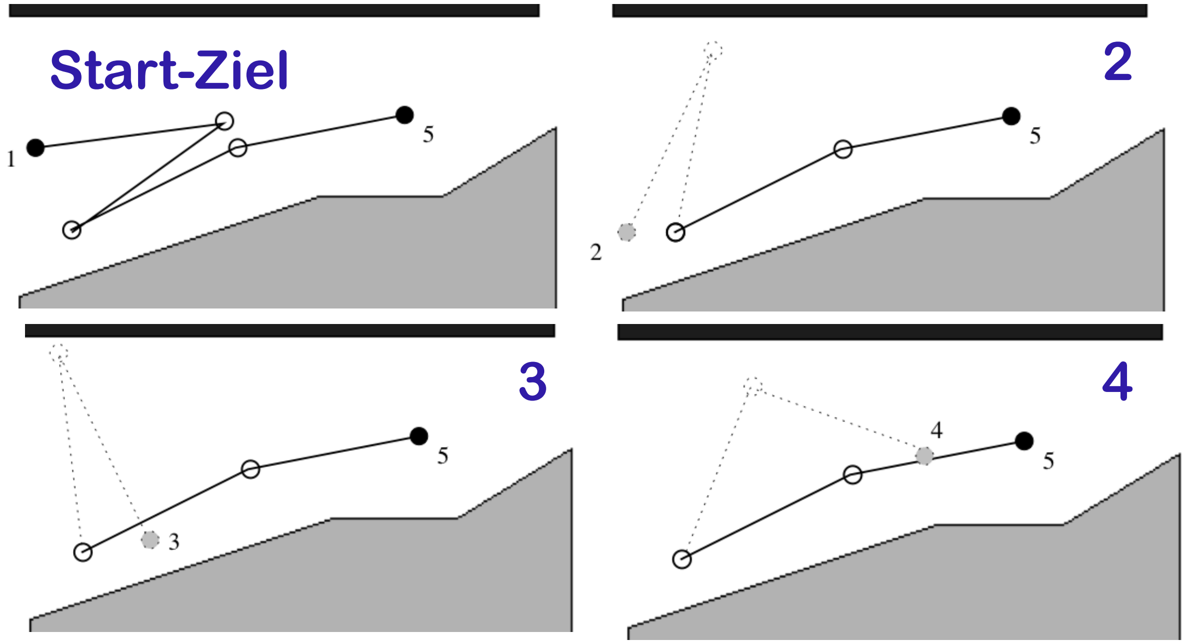
- Roboterkinematik und -Geometrie
- belegter Raum
- gegenüberliegende Seiten identifizieren!
- K-Raum-Posen umrechenbar in A-Raum-Posen

Sinn des Konfigurationsraums

Vereinfache Bewegungsplanung und/oder Umrechnung geplanter Trajektorien in Steuer/Regelkommandos!

Beispiel: Trajektorie von 1 bis 5 entspricht Steuerung entlang entsprechender Bahn im freien Konfigurationsraum





Berechne K-Raum nur,
wenn/soweit vorab bekannt und über längere Zeit konstant!

Wahleines Konfigurationsraums

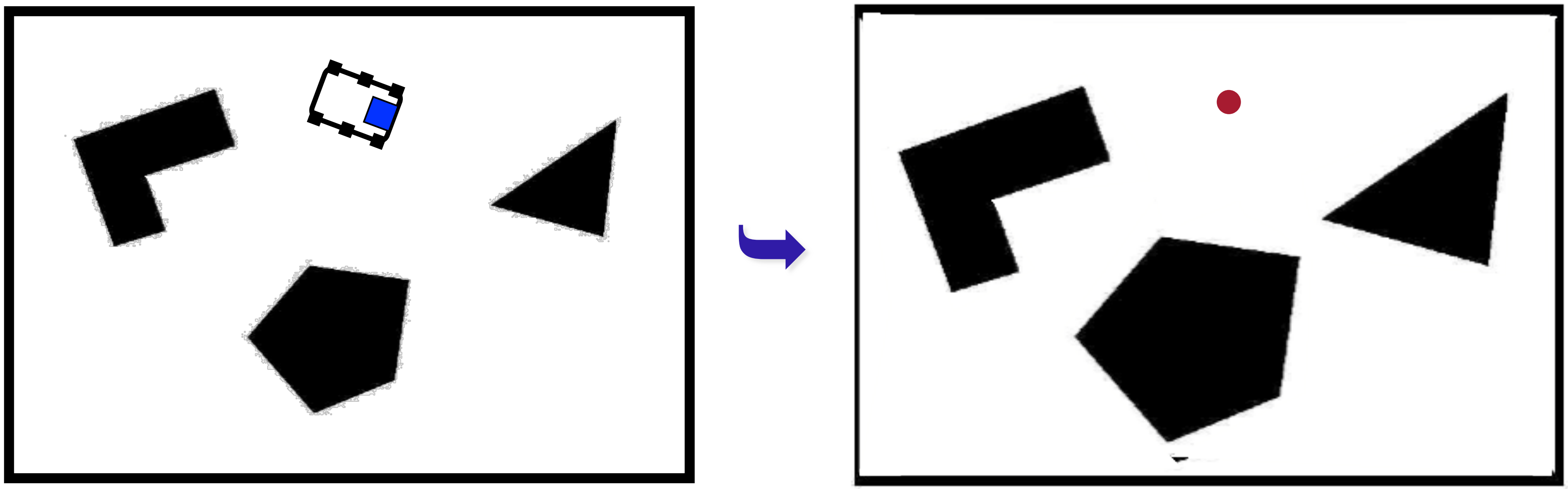
Gegeben A-Raum, wie wählt man dazu passenden K-Raum?

Es gibt nicht den K-Raum zur Pfadplanung mobiler Roboter!

- manchmal unterbestimmt
  (z.B. nicht-holonome Fahrzeuge: 2 DOF-KURT in 6-DOF-Poseraum)
- manchmal überbestimmt
  (z.B. redundante Kinematiken: 7DOF-Armroboter für 6DOF-Werkzeugpose)
- praktisch immer nichtlinear
  (z.B. begrenzte Knickwinkel, K-Raum-Hindernisse)

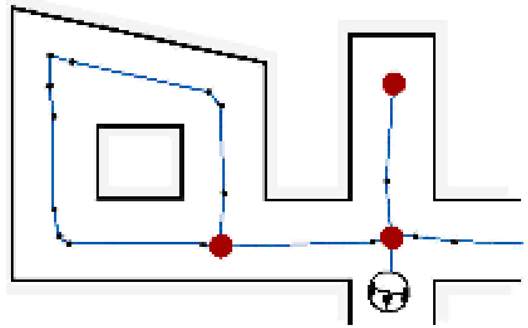
Üblicher 2D-K-Raum zur Pfadplanung

- Geh aus von 2D-Karte mit polygonalen Objekten
- Vergrößere alle Objekte um halben Roboterdurchmesser
- Reduziere Roboter auf (holonomen) Punkt
- Als Approximation für Roboter, die auf der Stelle drehen können (reduziert 3D-Pose x, z, θ auf 2D K-Raum)

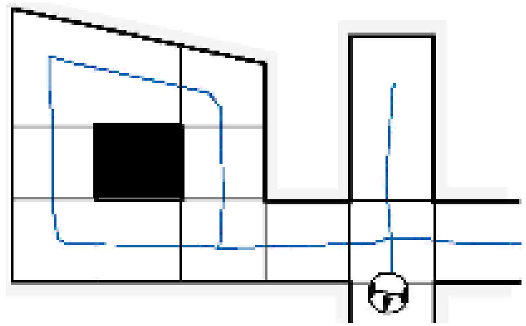


Drei Familien von Pfadplanungsverfahren

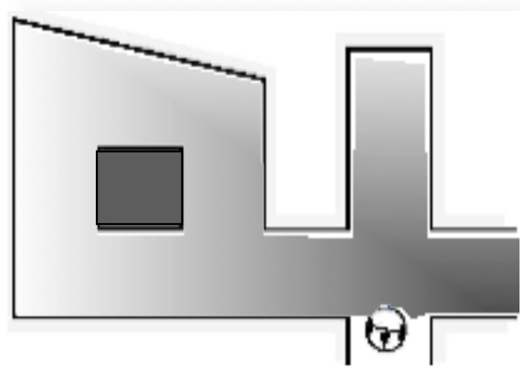
- **Straßenkarten** (***road maps***):
  Identifiziere Wege im Freiraum
  - Wohin Zwischenpositionen?
    Topologisch (merkmalbasiert), metrisch



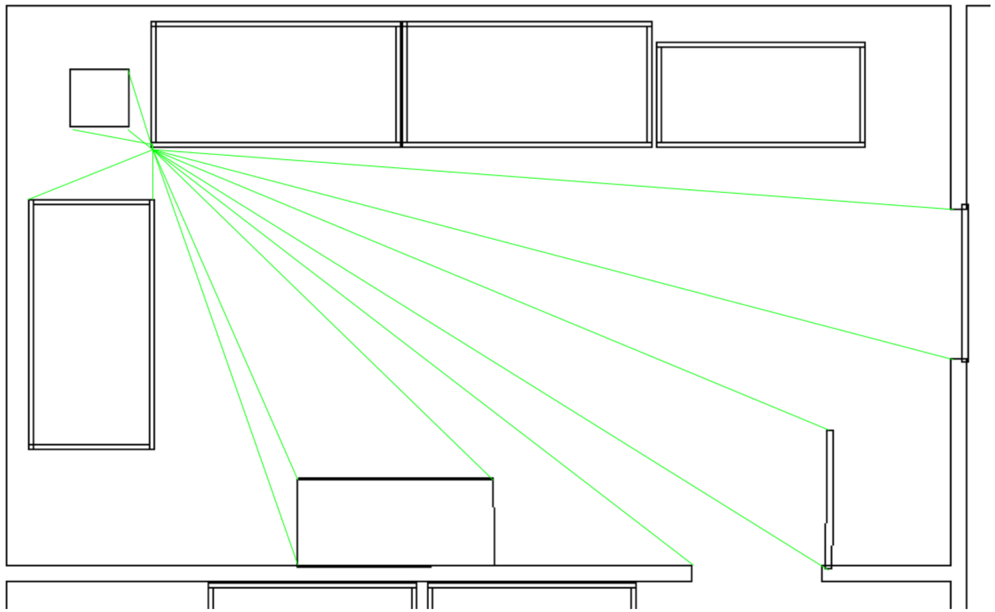
- **Parkettierung** (***cell decomposition***):
  Unterscheide freie vs unzugängliche Areale
  - Wohin Zellgrenzen?
    Metrisch (Raster), topologisch (merkmalbasiert)



- **Potenzialfelder** (***potential fields***)
  Überlagere Karte mit Gradienten („Nutzen“)



Straßenkarten I: Sichtbarkeitsgraphen



- **Knoten** sind alle Polygon-Ecken sowie Start- und Zielpunkt
- Verbinde alle Knotenpunkte, die jeweils frei sichtbar sind
- Suche kürzesten Weg (Summe der Streckenlängen) von Start nach Ziel entlang Sichtbarkeitskanten

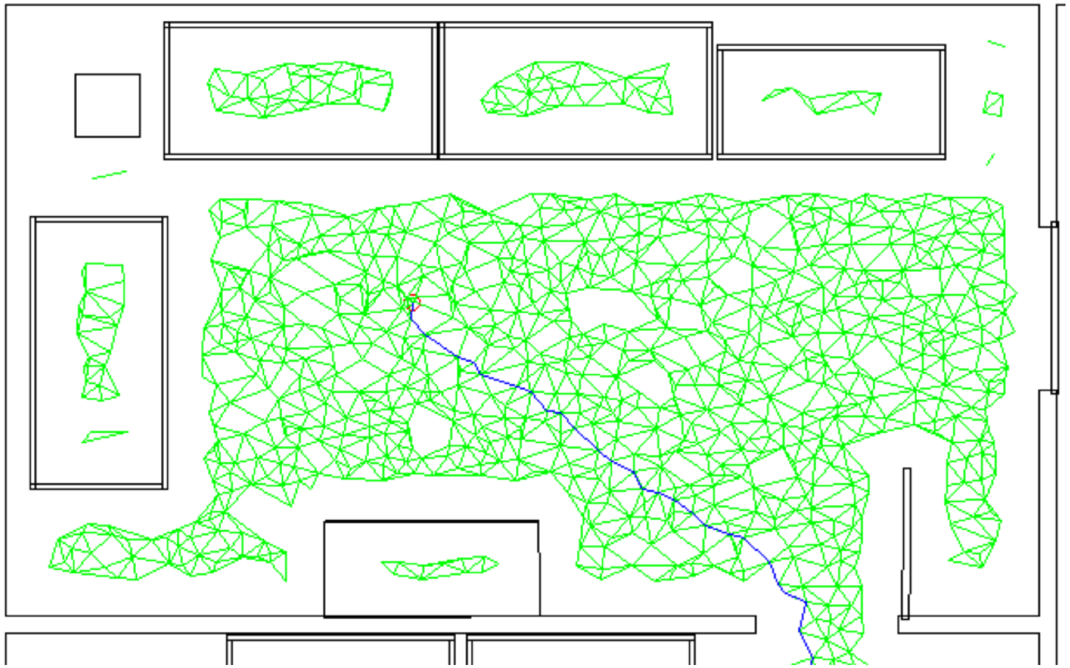
➥ Graphkonstruktion $O(|\mathrm{Knoten}|^2)$; Suche exponenziell i. Kanten

➥ Findet geometrisch kürzesten Weg („Ideallinie“)

➥ Weg ist schwierig zu fahren, kommt Hindernis-Ecken nah

- ➥ Ausführung erfordert sichere Hindernisvermeidung
- ➥ Tatsächlich gefahrener Weg möglicherweise nicht optimal kurz

Straßenkarten II: Probabilistisch



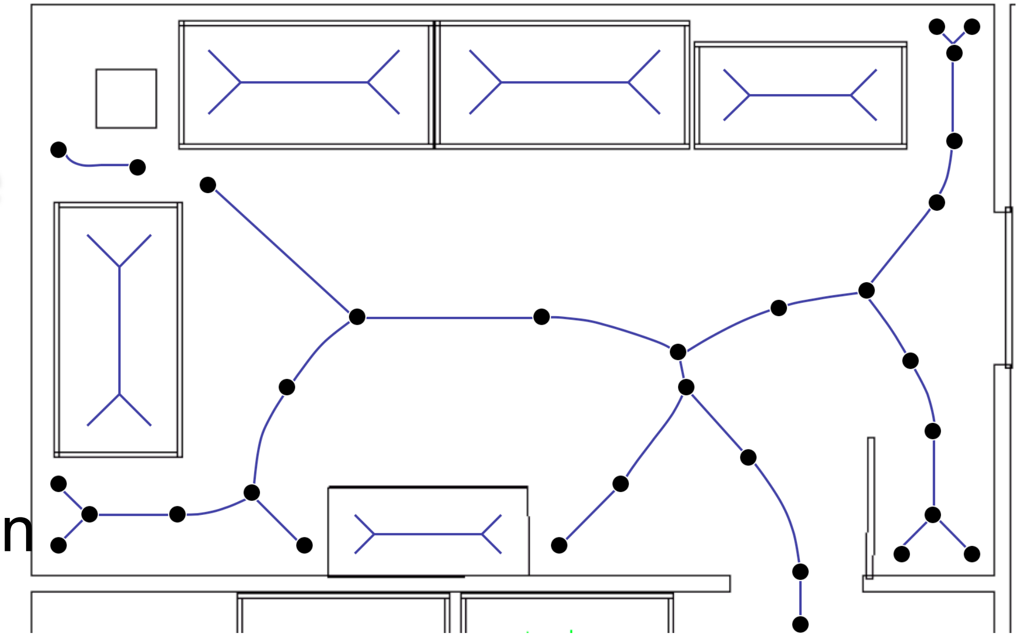
- Streue N Punkte gleichverteilt in den Freiraum + Start + Ziel
- Verbinde benachbarte Punkte durch planaren Graphen mit Kanten durch Freiraum
- Plane kürzesten Pfad entlang der Kanten

➥ Komplexität der Graph-Erzeugung und der Suche exponenziell in N, aber N wird überschaubar gewählt

➥ Pfad asymptotisch optimal; Hindernisabstand erhöht Länge

➥ Pfad kann unnötige „Zacken“ enthalten; ggf. glätten

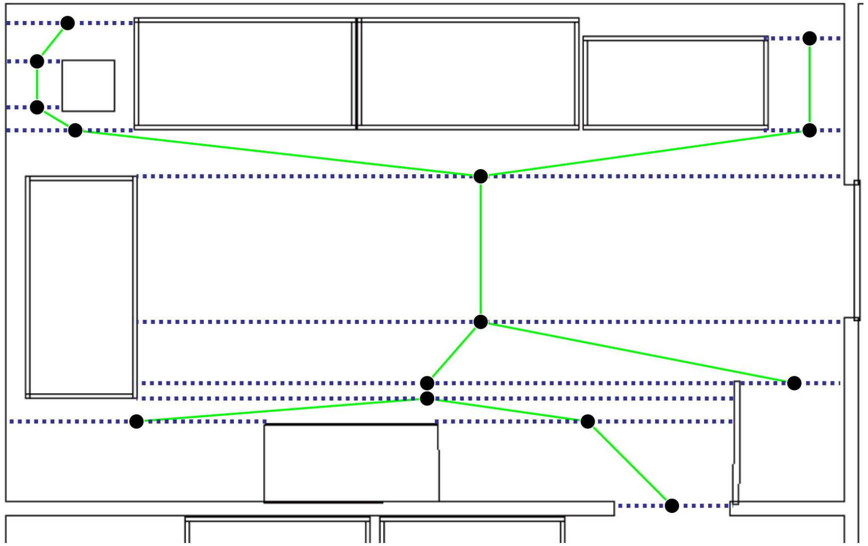
Straßenkarten III: Voronoi-Diagramme

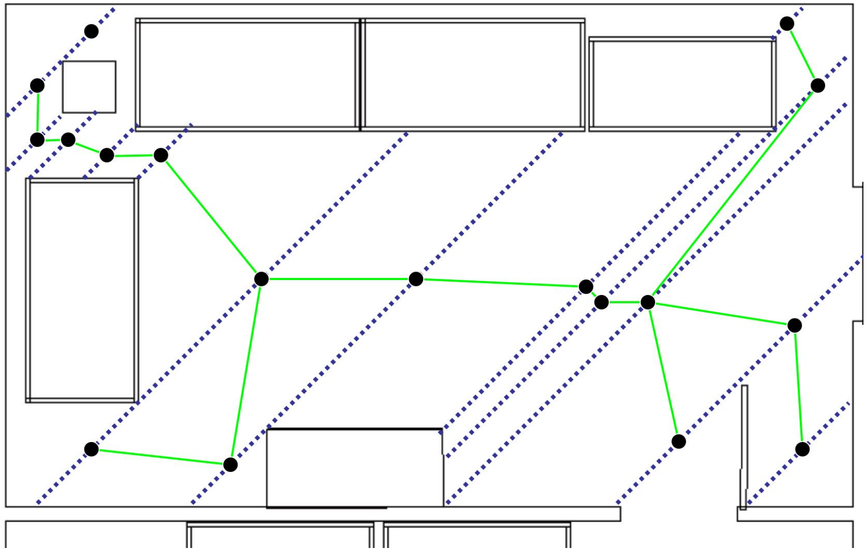


- **Voronoi-Graph**: Linien aus Punkten, die gleichen Abstand zu ihren beiden nächsten Hindernissen haben
- Plaziere Knoten auf Kreuzungen und auf „lange“ Linien
- Verbinde Start und Ziel mit ihren nächsten Voronoi-Linien
- Suche kürzesten Weg im Graphen

↪ Komplexität der Graphkonstruktion: $O(m \log m)$ für m Ecken
↪ Pfade von suboptimaler Länge
↪ Pfade maximieren Hindernisabstand, sind sicher fahrbar
↪ bei zu großem Hindernisabstand ggf Lokalisierungsprobleme

Parkettierung I: Exakte Zerlegung





- Zerlege Fläche durch parallele Linien an allen Polygon-Ecken (nicht notw. achsparallel)
- Plane Pfad topologisch im Nachbarschaftsgraphen
- Plane geometrisch (z.B. zwischen Mittelpunkten der zu überquerenden Liniensegmente plus ggf. Optimierung)

↪ Komplexität der Zerlegung $O(|\text{Ecken}|\cdot|\text{Polygone}|)$

↪ Zellen irregulär polygonal

↪ Pfad suboptimal; Kompromiss aus Länge & Hindernisabstand

↪ Verfahren selten eingesetzt; gut bei großem Freiraum

Parkettierung II: Rasterung

- Verwende Rasterung mit uniformer Zellengröße oder Quadtrees wie eingeführt (Kartierung)
- Verwende Zellmittelpunkte oder Mittelpunkte der Zellgrenzen als Suchknoten;
  Kanten zwischen Knoten aus Nachbarzellen

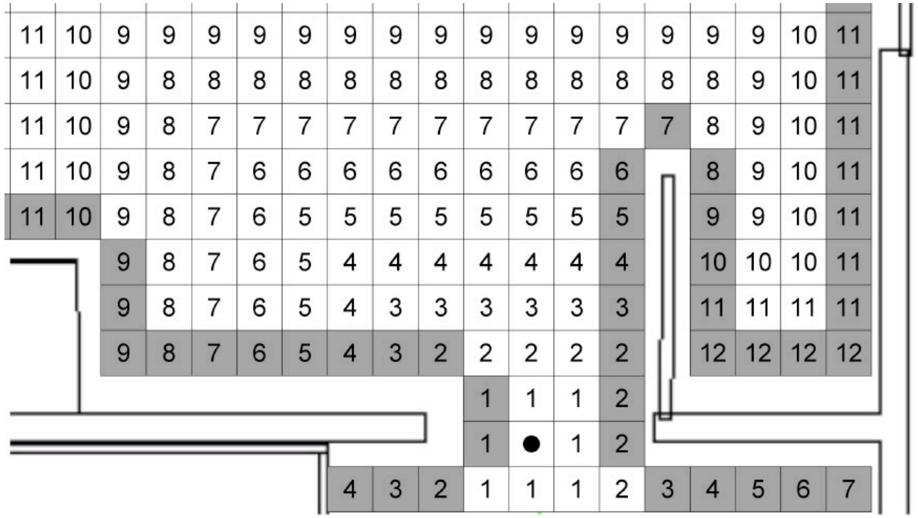
↪ Rasterkarte gegeben (Kartierung) oder kostet $O(|\text{Zellen}|)$

↪ Rasterung kann im Extremfall Wege blockieren

↪ Pfadplanung nah-optimal bis auf Blockierung

↪ Pfade durch Zellmitten/grenzen „zackelig";
Fahrsteuerung glättet

Wavefront-Planung zu konstanten Zielen



Falls oft zum <u>selben</u> Zielpunkt geplant werden muss:

- In Rasterkarte bewerte Zielzelle mit 0; trage rekursiv Kosten von Nachbarn bewerteter Zellen ein (Wavefront-Algorithmus)
- Ggf. bestrafe Zellen nahe Objekten (Straf-Term)
- Planung zu diesem Zielpunkt ist dann nicht mehr nötig: Folge von beliebigem Start dem Werte-Gradienten!

↪ Wavefront-Bewertung kostet $O(|\text{Zellen}|)$

↪ Pfadplanung nah-optimal

↪ Pfade wie bei (uniformen) Rasterkarten

Hybrid aus Parkettierung und Potenzialfeld

Potenzialfeldverfahren

- Modelliere Zielpunkt durch „Anziehung“, Hindernisse durch „Abstoßung“
- Pfadplanung ist Gradientenabstieg im „Potenzialfeld“
- Für Punkt $\boldsymbol{p}$:

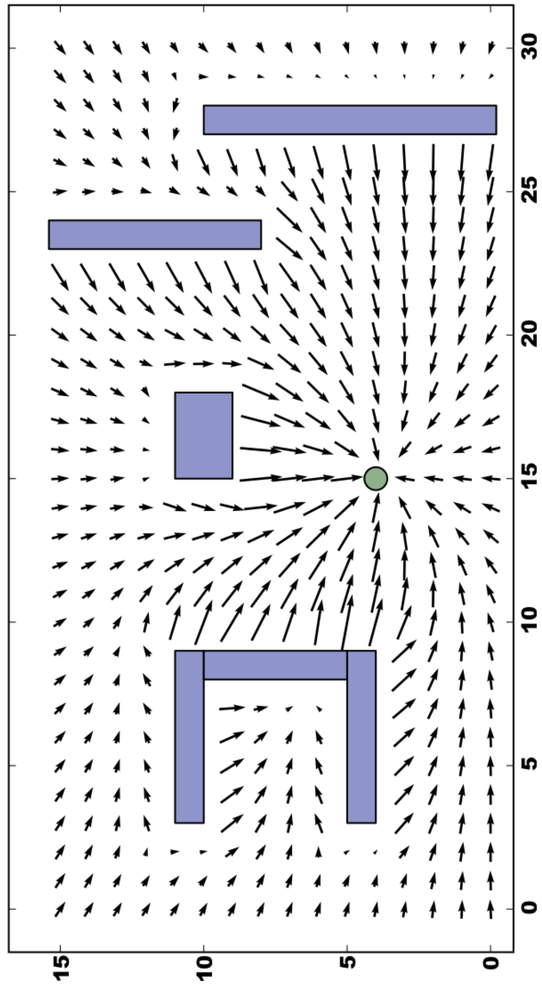
$$U(\boldsymbol{p}) = U_{\text{Ziel}}(\boldsymbol{p}) + \sum U_{\text{Obj}}(\boldsymbol{p})$$

$$F = -\begin{bmatrix} \partial U/\partial x \\ \partial U/\partial z \end{bmatrix}$$

U_{Obj} meist nur für <u>nahe</u> Hindernisse berücksichtigt

$$U_{\text{Ziel}}(\boldsymbol{p}) = c_U \cdot \operatorname{dist}\left(\boldsymbol{p}, \boldsymbol{p}_{\text{Ziel}}\right)$$

$$U_{Obj}(\boldsymbol{p}) = c_{Obj} \cdot \operatorname{dist}\left(\boldsymbol{p}, \boldsymbol{o}\right)^{-2}$$



➥ Berechnung der Karte kann beliebig aufwändig sein

➥ Pfad kann in lokale Potenzialminima führen („Roboterfallen“)

➥ Pfade „elegant“

Diskretisierte Version (z.B. in Rasterzellen): **Vektorfelder**

Und wenn man <u>doch</u> Orientierung braucht?

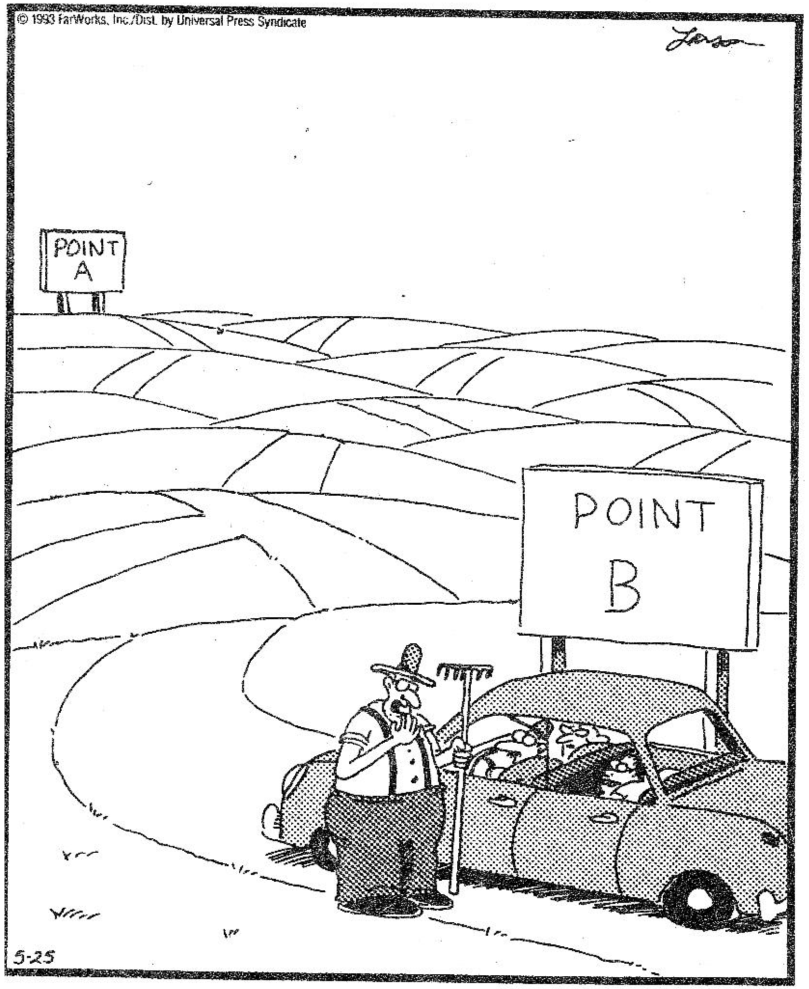
… dann kann es beliebig kompliziert werden

↳ *Piano Movers' Problem*

↳ Algorithmische Geometrie

Videos von LAAS, Toulouse
http://www.laas.fr/1-29634-Pole-RIA.php

Fazit Pfadplanung



"Well, lemme think. ... You've stumped me, son. Most folks only wanna know how to go the other way."

„How Do I Get There?“ – Der Planungsaspekt

- Online-Pfadplanung nutzt vereinfachten K-Raum
- Die eigentliche Planung geht mit Standard-Methoden à la A*-Suche
- Algorithmen mit reichen Kinematik- und Geometriemodellen untersucht *Algorithmische Geometrie*
- Pfadpläne sagen, wo es <u>ideal</u> nach Karte lang geht. Ausführung muss als geschlossene Regelung laufen!