

Robotik I: Einführung in die Robotik

Bewegungsplanung

Tamim Asfour

KIT-Fakultät für Informatik, Institut für Anthropomatik und Robotik (IAR)
Hochperformante Humanoide Technologien (H²T)

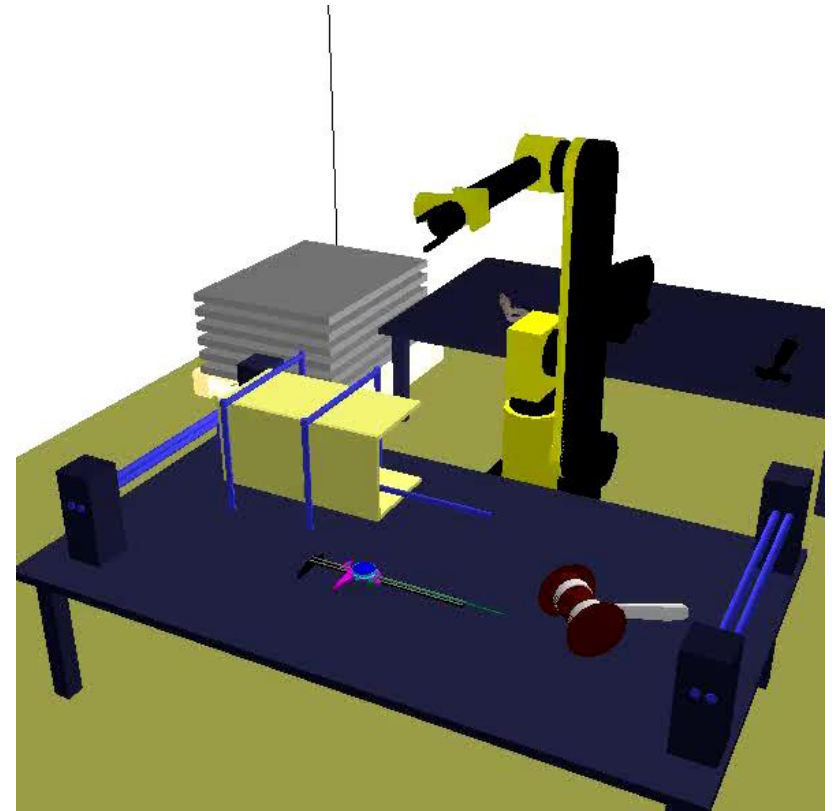
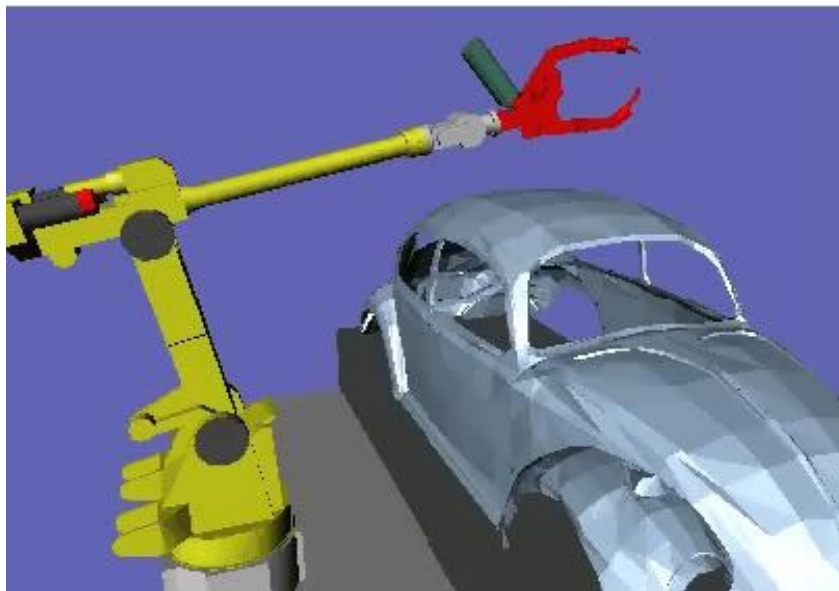


Inhalt

- Motivation
- Grundlagen der Bewegungsplanung
- Pfadplanung für mobile Roboter
- Bewegungsplanung für Manipulatoren

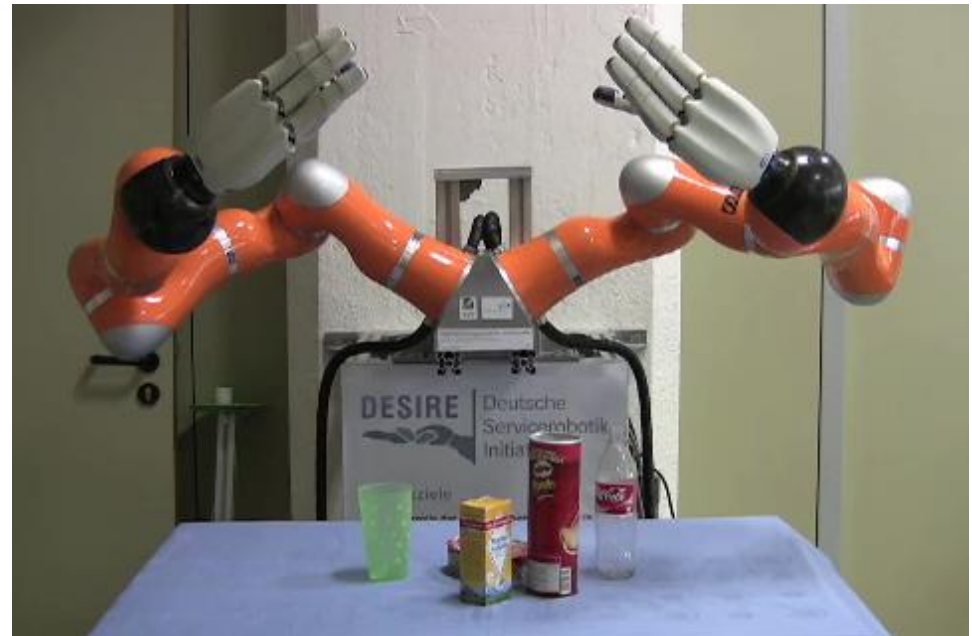
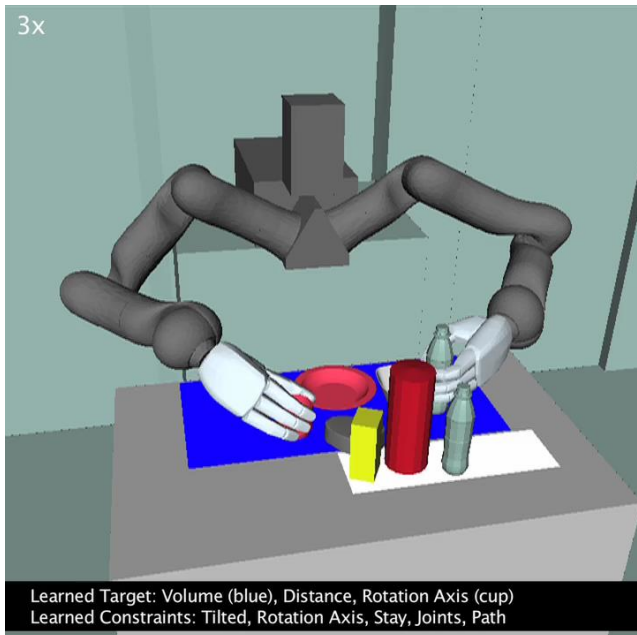
Bewegungsplanung: Motivation

Erzeugen einer **kollisionsfreien** Trajektorie unter Berücksichtigung verschiedener **Ziele** und **Einschränkungen**



Bewegungsplanung: Motivation

Erzeugen einer kollisionsfreien Trajektorie unter Berücksichtigung verschiedener Ziele und Einschränkungen



Inhalt

- Motivation
- Grundlagen der Bewegungsplanung
 - Problemstellung
 - Definitionen
 - Begriffsbildung
 - Problemklassen
- Pfadplanung für mobile Roboter
- Bewegungsplanung für Manipulatoren

Grundlagen der Bewegungsplanung: Problemstellung

■ Gegeben

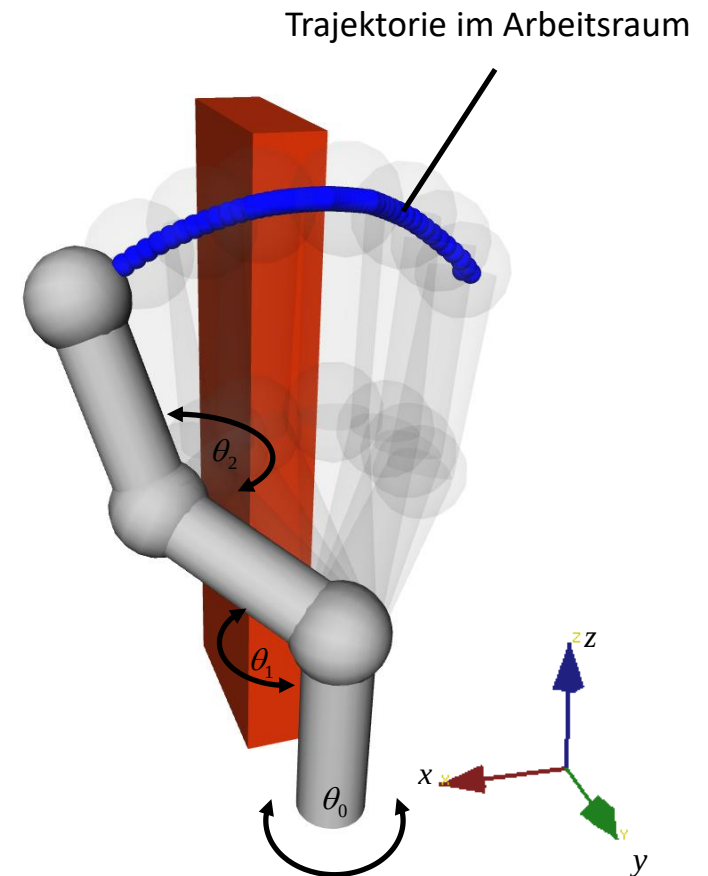
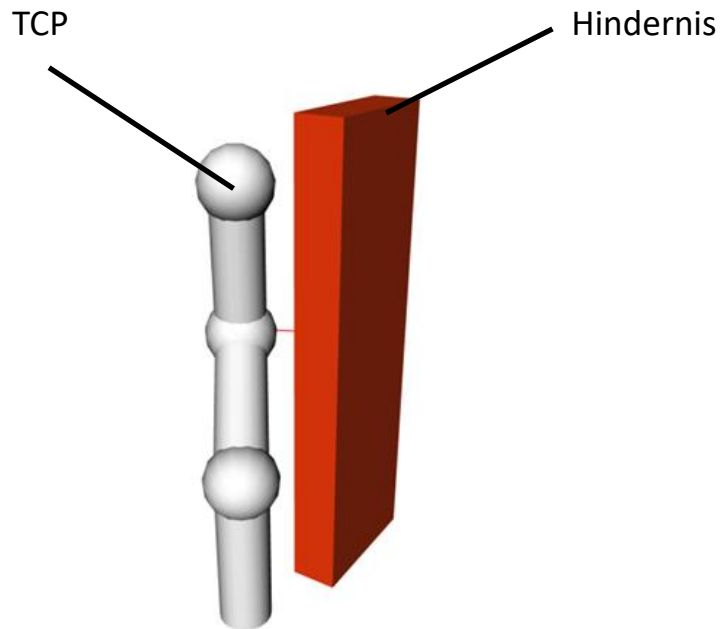
- Konfigurationsraum $\mathcal{C}$
- Startkonfiguration $\mathbf{q}_{start} \in \mathcal{C}$
- Zielkonfiguration $\mathbf{q}_{ziel} \in \mathcal{C}$

■ Gesucht

- Stetige Trajektorie $\tau: [0,1] \rightarrow \mathcal{C}$ mit
 - $\tau(0) = \mathbf{q}_{start}$
 - $\tau(1) = \mathbf{q}_{ziel}$
- Unter Berücksichtigung von
 - Gütekriterien
 - Neben- und Randbedingungen
 - Zwangsbedingungen

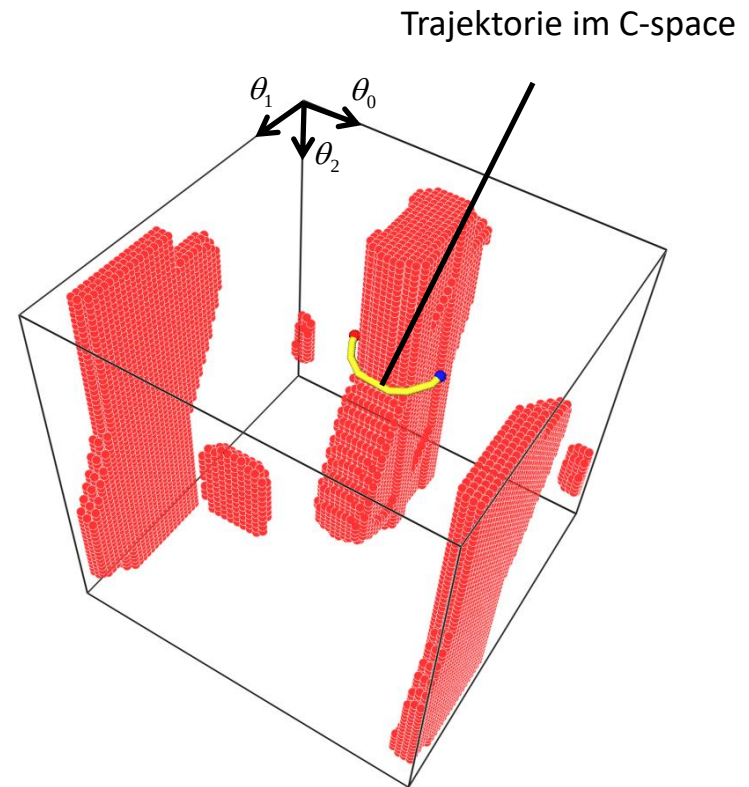
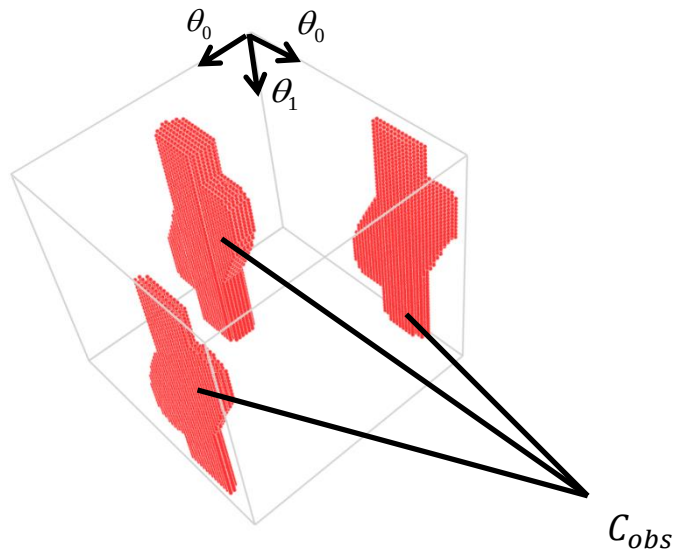
Einführung: Arbeitsraum

- W : Kartesischer Raum $\mathbb{R}^6$
- Tool Center Point (TCP)

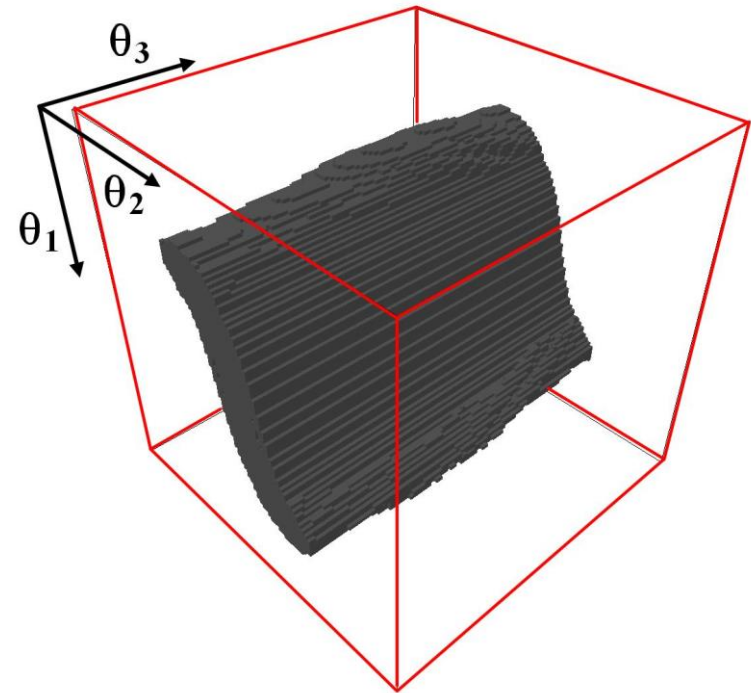
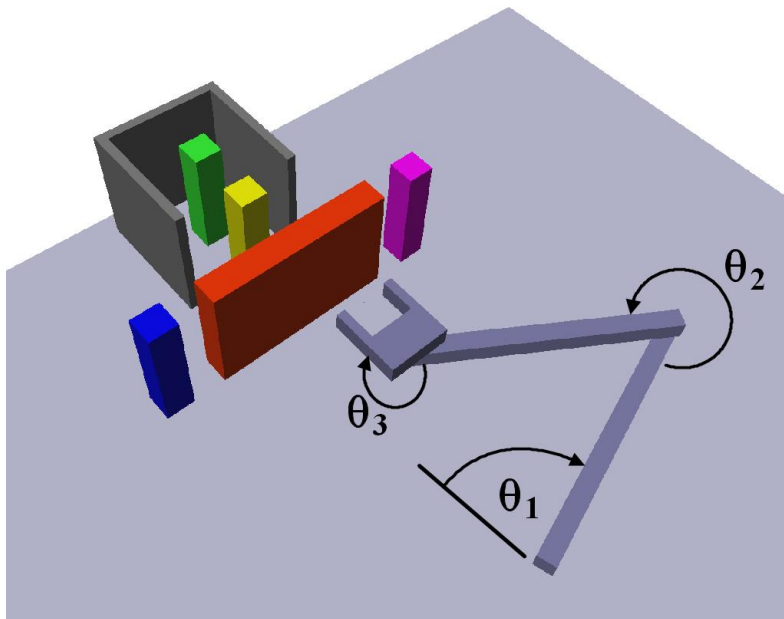


Einführung: Konfigurationsraum C

- C : n-dimensionaler Konfigurationsraum(C-Space)
- C_{free} : Alle kollisionsfreie Konfigurationen
- C_{obs} : Alle Konfigurationen, die zur einer Kollision führen
- $C = C_{free} \cup C_{obs}$

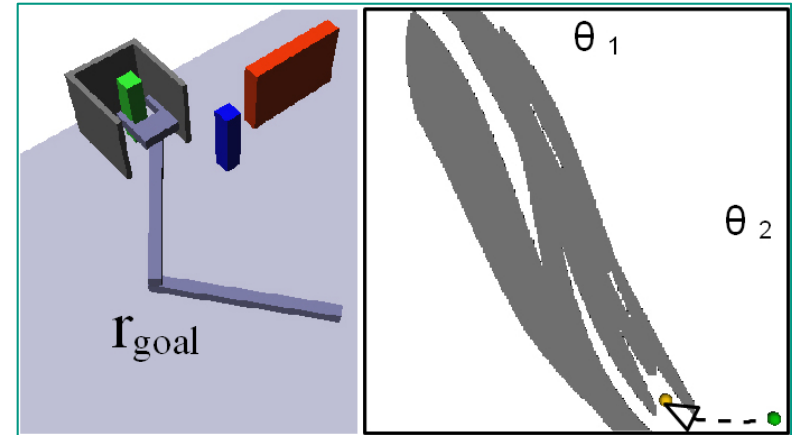
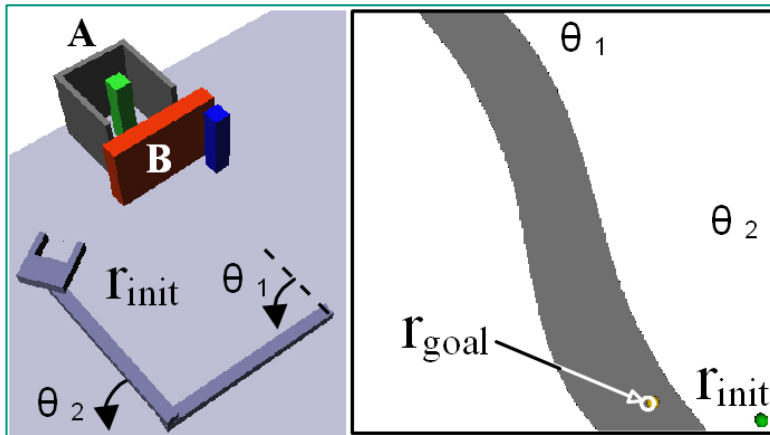


Arbeitsraum vs. Konfigurationsraum



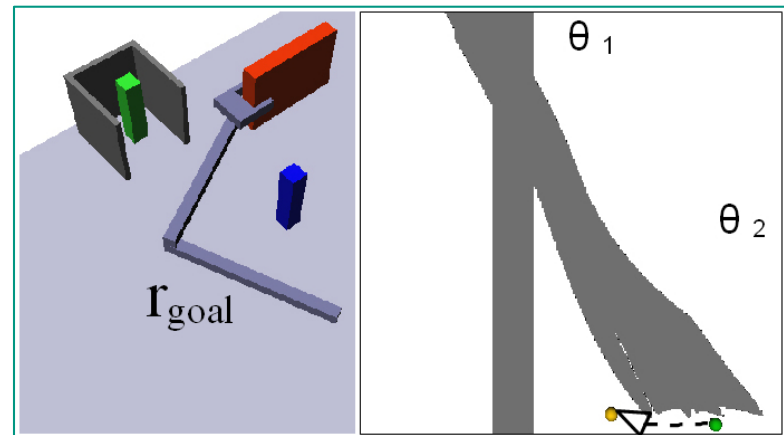
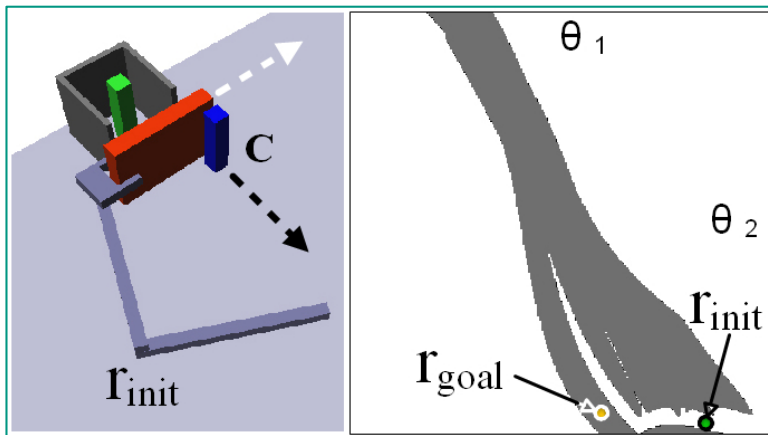
Einführung: Konfigurationsraum C

- Wie wirken sich Änderungen im Arbeitsraum auf den C-Raum?



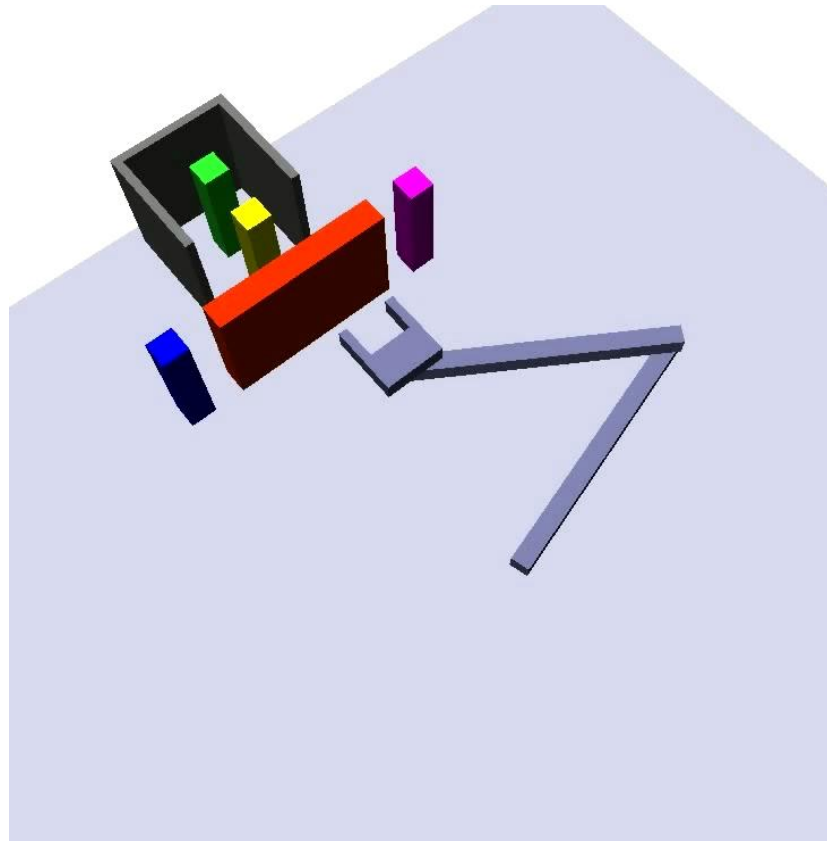
Einführung: Konfigurationsraum C

- Wie wirken sich Änderungen im Arbeitsraum auf den C-Raum?



Einführung: Konfigurationsraum C

- C_{free} und C_{obs} ändern sich während der Ausführung



Grundlagen der Bewegungsplanung: Definitionen (1)

■ Konfiguration

Eine **Konfiguration** $q \in \mathcal{C}$ beschreibt den Zustand eines Roboters

- als Lage und Orientierung im euklidischen Raum oder
- als Gelenkwinkelvektor im Gelenkwinkelraum.

■ Konfigurationsraum

Der **Konfigurationsraum** $\mathcal{C}$ eines Roboters R ist der Raum aller möglicher Konfigurationen von R .

Grundlagen der Bewegungsplanung: Definitionen (2)

■ Arbeitsraumhindernis

Ein **Arbeitsraumhindernis** H ist der Raum, welcher von einem Objekt im Arbeitsraum eingenommen wird.

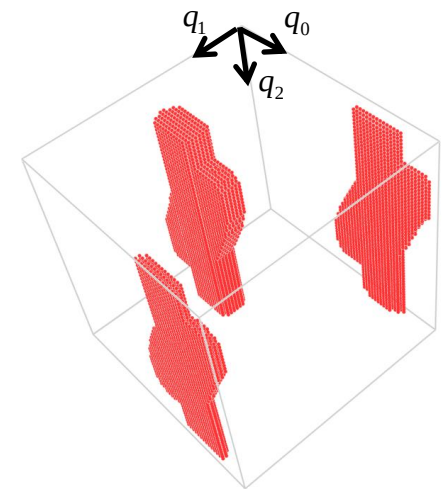
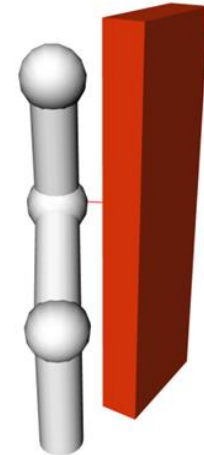
■ Konfigurationsraumhindernis

Ein **Konfigurationsraumhindernis** C_H ist die Menge aller Punkte des Konfigurationsraumes C , welche zu einer Kollision mit dem Hindernis H führen.

■ Hindernisraum

Der **Hindernisraum** C_{obs} ist die Menge aller Konfigurationsraumhindernisse

$$C_{obs} = \bigcup_i C_{H_i}$$



Grundlagen der Bewegungsplanung: Definitionen (3)

■ Freiraum

- Der **Freiraum** ist die Menge aller Punkte aus $\mathcal{C}$, welche nicht im Hindernisraum $\mathcal{C}_{obs}$ liegen

$$\mathcal{C}_{free} = \{q \in \mathcal{C} \mid q \notin \mathcal{C}_{obs}\} = \mathcal{C} \setminus \mathcal{C}_{obs}$$

- Aufwand für die Berechnung des Freiraums: $O(m^n)$
 - n : Anzahl der Bewegungsfreiheitsgrade des Roboters
 - m : Anzahl der Hindernisse
- Für komplexere Kinematiken kann $\mathcal{C}_{free}$ nicht effizient berechnet werden
- Verwendung **approximativer Verfahren** zur vereinfachten Repräsentation von $\mathcal{C}_{free}$

Grundlagen der Bewegungsplanung: Definitionen (4)

■ Umweltmodellierung

- **Exakt:** Beispielsweise über CSG (constructed solid geometry), in Form einer algebraischen Beschreibung
- **Approximiert:** Die Umwelt wird durch Näherungen beschrieben (Boxen, verallgemeinerte Zylinder, Polyeder,....)

Grundlagen der Bewegungsplanung: Begriffsbildung (1)

■ Pfadplanung

- Starres Objekt (z.B. mobiler Roboter, autonomes Fahrzeug)
- 2D Problem (Position: x, y)
- 3D Problem (Position: x, y ; Rotation: α)
→ Piano Mover's Problem

■ Bewegungsplanung

- Mehrkörpersystem (z.B. Roboterarme, Systeme mit mehreren Robotern)
- Hochdimensionale Problemstellungen

■ Randbedingungen, auch Zwangsbedingungen

- Globale Randbedingungen: Limitieren den gültigen Konfigurationsraum, z.B.
 - Aufrechte Position des Endeffektors, minimale Motorströme, etc.
- Lokale Randbedingungen: Schränken die Übergänge zwischen Konfigurationen ein, z.B.
 - Nicht-holonome Fahrzeuge können sich nicht seitlich bewegen oder auf der Stelle drehen
 - max. Geschwindigkeit/Beschleunigung einhalten

■ Komplexität

Allgemeine Planungsaufgaben sind PSPACE-vollständig (engl. PSPACE-complete).

- Können von deterministischen Turingmaschinen mit polynomielltem Platz (Speicherplatz) entschieden werden
- Untere und obere Schranke der Komplexität $NP \subseteq PSPACE \subseteq EXPTIME$, also NP-hartes Problem

■ Vollständiger Algorithmus

Ein vollständiger Algorithmus findet für spezielle Planungsprobleme mindestens eine Lösung oder erkennt in endlicher Zeit, dass keine Lösung existiert.

■ Randomisierter Algorithmus

Randomisierte Algorithmen verwenden Zufallsgrößen, um den Ablauf zu steuern, wobei oft heuristische Annahmen genutzt werden, um die Berechnung zu beschleunigen

■ Auflösungsvollständiger Algorithmus

Ist ein approximativer Algorithmus für eine diskretisierte Problemstellung vollständig, wird er auflösungsvollständig genannt, (engl. resolution complete)

■ Probabilistisch-vollständiger Algorithmus

Ein probabilistisch-vollständiger Algorithmus (engl. probabilistically complete) findet mindestens eine Lösung falls sie existiert. D.h. die Wahrscheinlichkeit, dass eine Lösung gefunden wird, konvergiert mit fortlaufender Zeit gegen eins.

Allerdings kann mit probabilistisch-vollständigen Algorithmen nicht ermittelt werden, ob keine Lösung existiert.

Grundlagen der Bewegungsplanung: Problemklassen I

■ Klasse a)

Bekannt: vollständiges Umweltmodell
vollständige Neben-, Rand- und Zwangsbedingungen

Gesucht: Kollisionsfreie Trajektorie vom Start- zum Zielzustand

■ Klasse b)

Bekannt: unvollständiges Umweltmodell
unvollständige Neben-, Rand- und Zwangsbedingungen

Gesucht: Kollisionsfreie Trajektorie vom Start- zum Zielzustand

Problem: Kollision mit unbekannten Objekten

Grundlagen der Bewegungsplanung: Problemklassen II

■ Klasse c)

Bekannt: zeitvariantes Umweltmodell (bewegliche Hindernisse)
Gesucht: Kollisionsfreie Trajektorie vom Start- zum Zielzustand
Problem: Hindernisse in Ort und Zeit variant

■ Klasse d)

Bekannt: kein Umweltmodell
Gesucht: Kollisionsfreie Trajektorie vom Start- zum Zielzustand
Problem: Kartographieren

■ Klasse e)

Bekannt: zeitvariantes Umweltmodell
Gesucht: Trajektorie zu einem beweglichen Ziel (Rendezvous-Problem)
Problem: Zielzustand in Ort und Zeit beweglich

Inhalt

- Motivation
- Grundlagen der Bewegungsplanung
- Pfadplanung für mobile Roboter
 - Graphenbasiert
 - Potentialfelder
- Bewegungsplanung für Manipulatoren

Pfadplanung für mobile Roboter

- Gegeben
 - 2D Weltmodell (z.B. Straßenkarte)
 - Start- und Zielpositionen q_{start} und q_{ziel}
- Gesucht: Günstigste Verbindung von q_{start} nach q_{ziel}
- Ansatz:
 - **Konstruiere** ein Netz W von Wegen in C_{free}
 - Bilde q_{start} und q_{ziel} auf die nächsten Knoten q'_{start} und q'_{ziel} in W ab
 - **Suche** in W einen Weg von q'_{start} nach q'_{ziel}
 - Finde einen Weg zwischen q_{start} und q'_{start} , sowie zwischen q'_{ziel} und q_{ziel}

Pfadplanung für mobile Roboter: 2 Problem

1. Konstruktion des Wegenetzes W

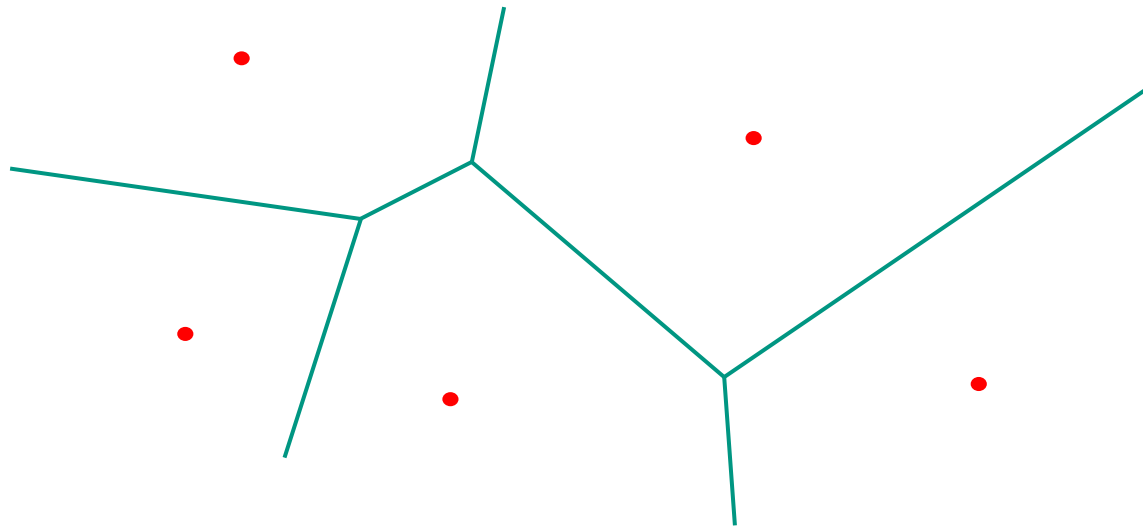
- Retraktionsverfahren, z.B. Voronoi-Diagramm
- Sichtgraphen
- Zellzerlegung

2. Suche in W

- Baumsuche
- A*

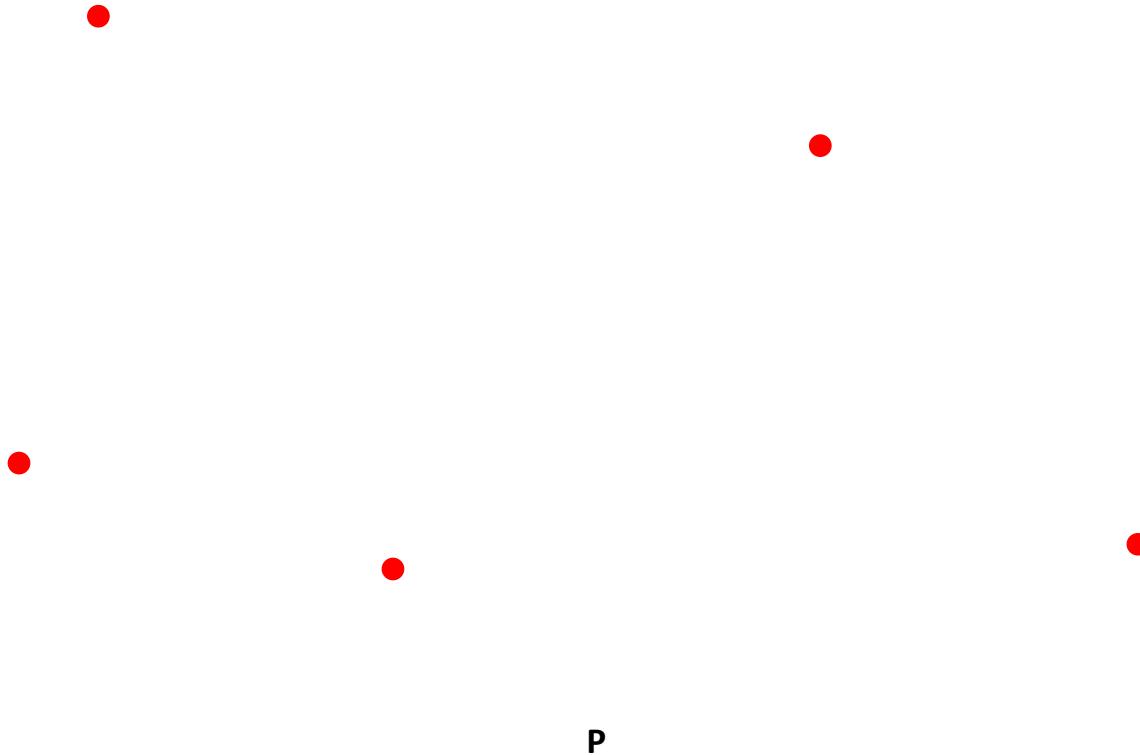
Voronoi-Diagramme

- Visualisiert **die Zerlegung eines Raumes** in Regionen basierend auf vorgegebenen Punkten.
- Eine **Region** ist definiert als die **Menge aller Punkte**, deren **Abstand zum Zentrum geringer** ist **als zu allen anderen Zentren**.
- Alle **Punkte auf der Grenze** zwischen zwei Regionen besitzen den **gleichen Abstand** zum eigenen und zum **benachbarten Zentrum**.



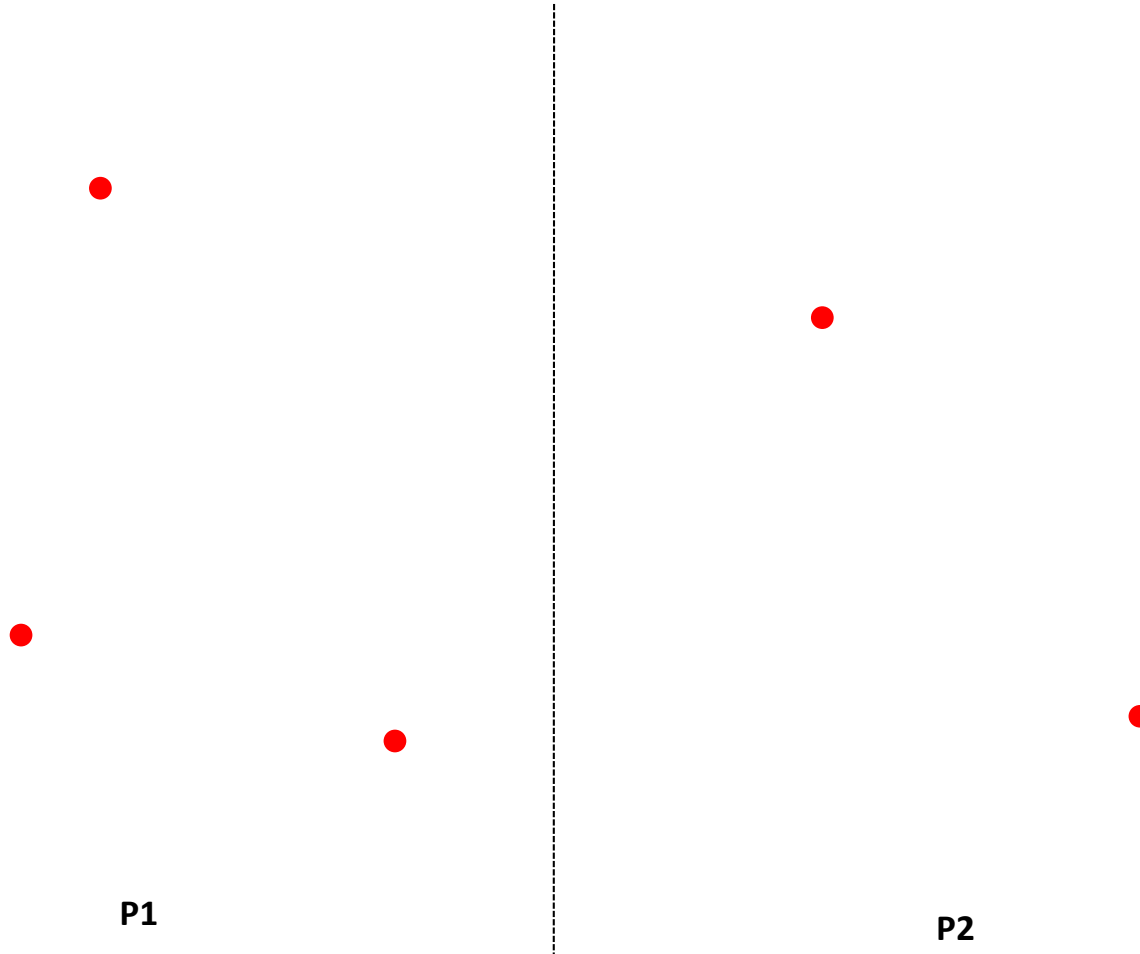
Voronoi-Diagramm: Konstruktion I

■ Gegebene **Punktmenge P**



Voronoi-Diagramm: Konstruktion II

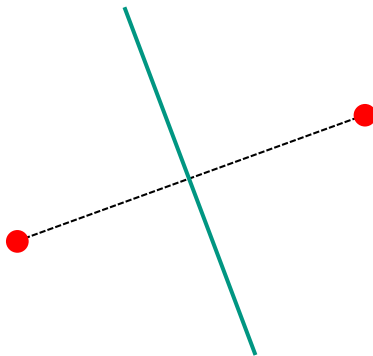
■ Teile P in zwei etwa gleich große Teilmengen P_1 und P_2



Voronoi-Diagramm: Konstruktion III

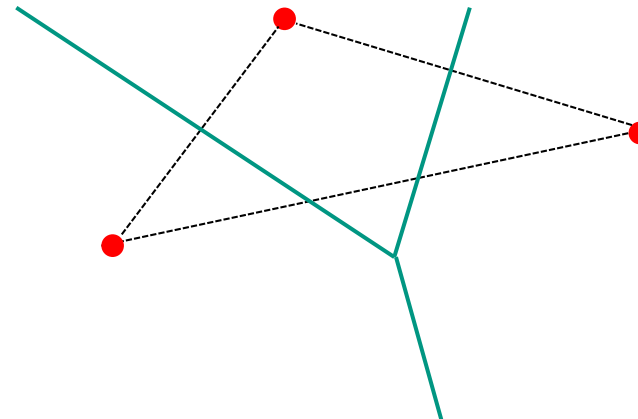
- Durch **rekursive Unterteilung** der Punktmengen kann das Problem der Erstellung eines Voronoi-Diagramms auf **zwei einfache Fälle** reduziert werden.

Fall 1: 2 Punkte



Die **Mittelsenkrechte** bildet das Voronoi-Diagramm

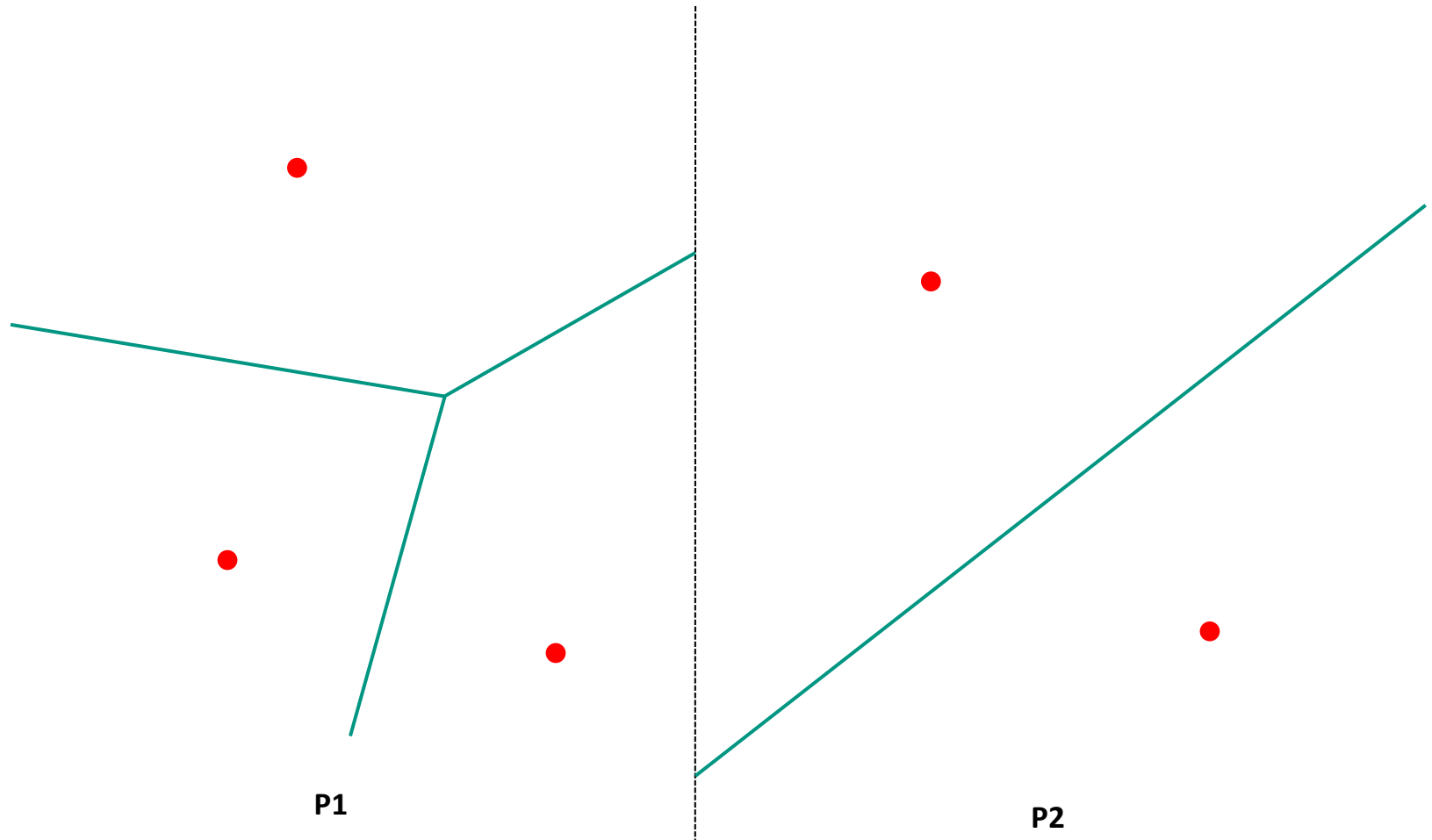
Fall 2: 3 Punkte



Die **Mittelsenkrechten aller Punktpaare** werden am gemeinsamen **Schnittpunkt abgeschnitten**

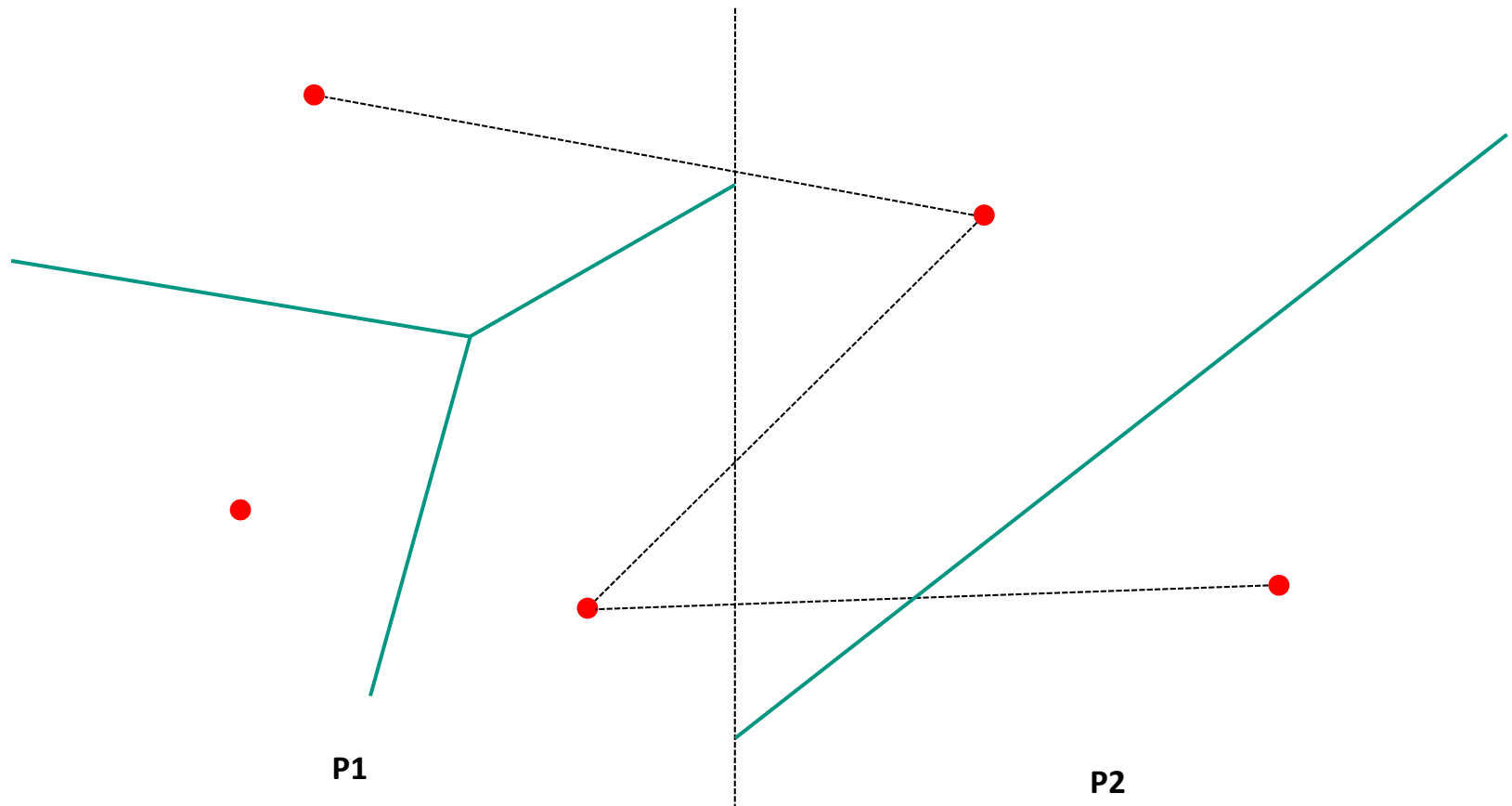
Voronoi-Diagramm: Konstruktion IV

■ Konstruiere **Voronoi-Diagramme** für **P1** und **P2**



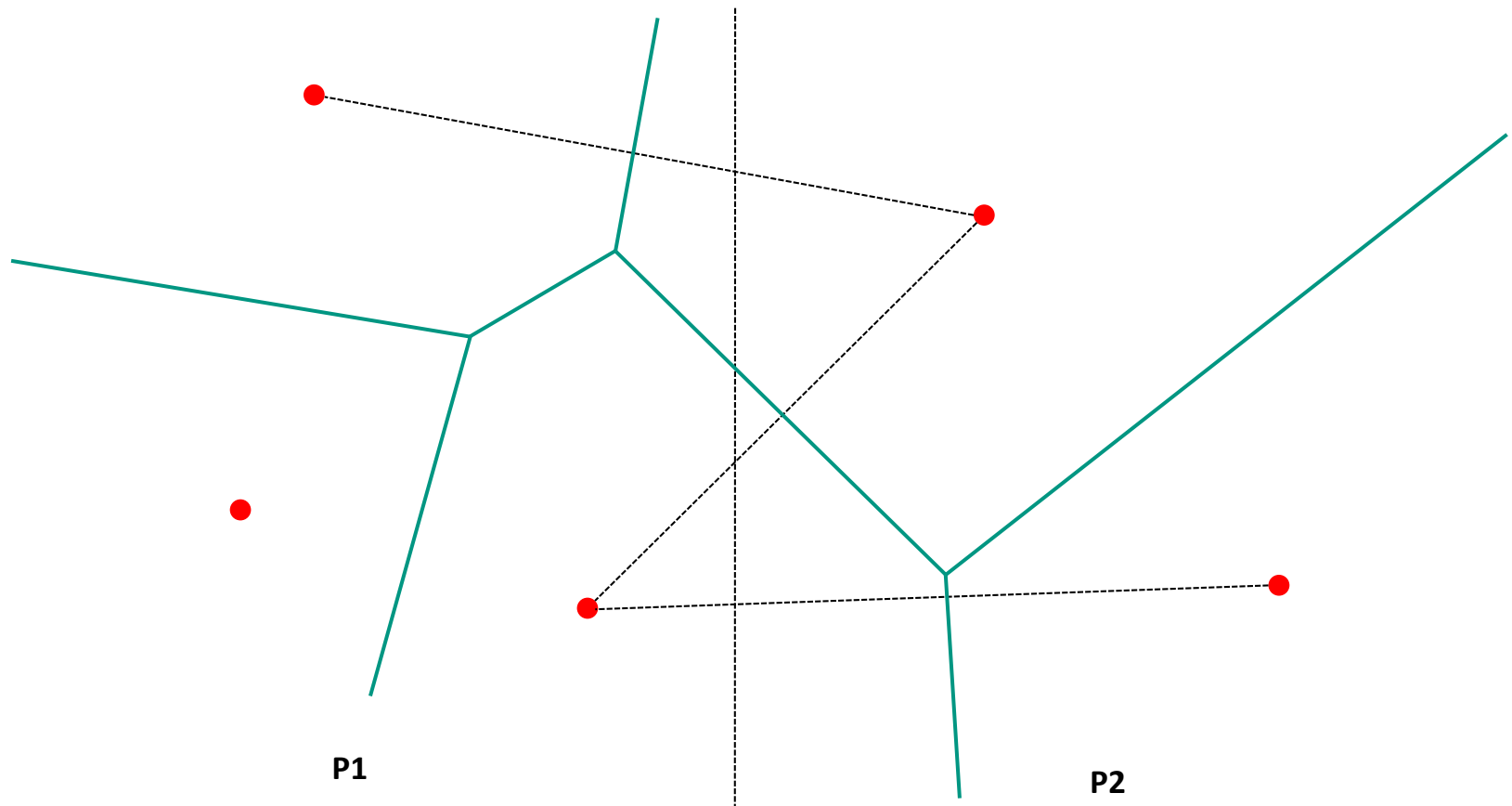
Voronoi-Diagramm: Konstruktion V

- **Verschmelze** die Voronoi-Diagramme für **P1** und **P2**
 - **Verbinden** der nächsten **Nachbarn** entlang der Trennungslinie



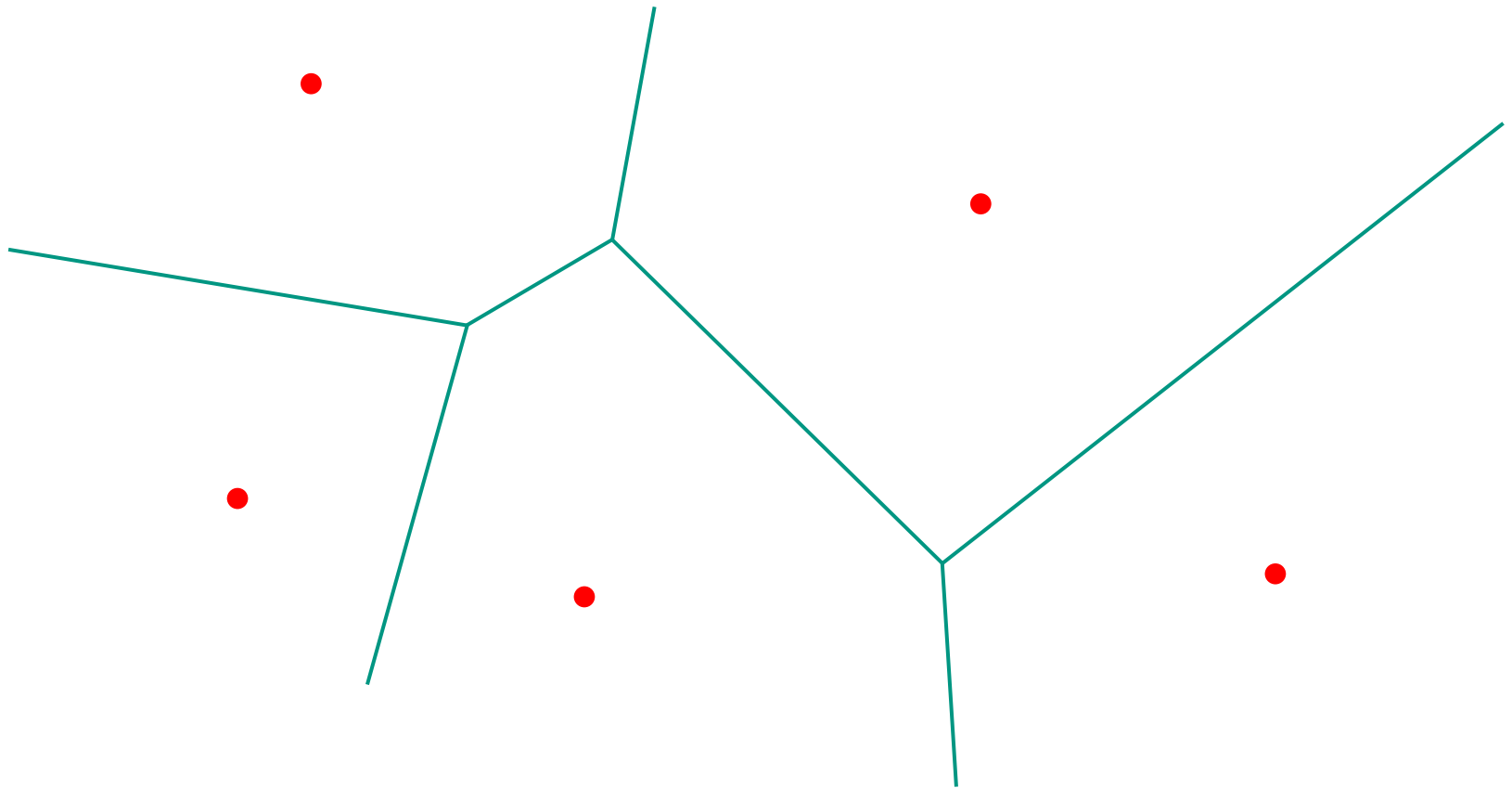
Voronoi-Diagramm: Konstruktion VI

- **Verschmelze** die Voronoi-Diagramme für **P1** und **P2**
 - Einzeichnen und Abschneiden der neuen **Mittelsenkrechten**

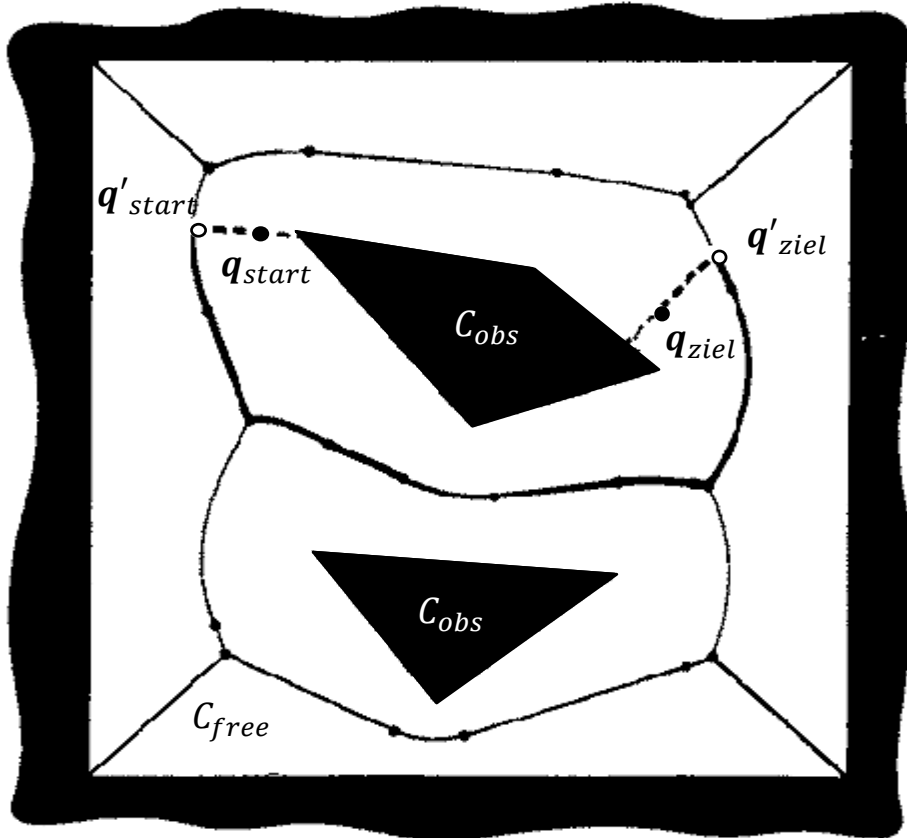


Voronoi-Diagramm: Konstruktion VII

■ Fertiges Voronoi-Diagramm für P



Voronoi-Diagramme: Vor- und Nachteile



■ Vorteile:

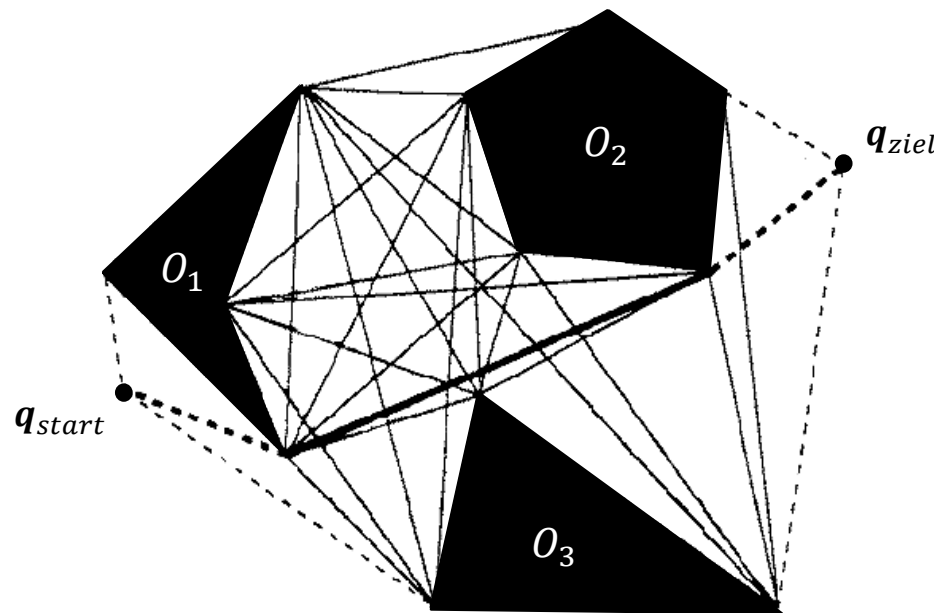
- **Maximaler Abstand** zu Hindernissen
- Ein Roboter kann mit Hilfe von Abstandssensoren **leicht prüfen**, ob der richtige Weg abgefahren wird

■ Nachteile

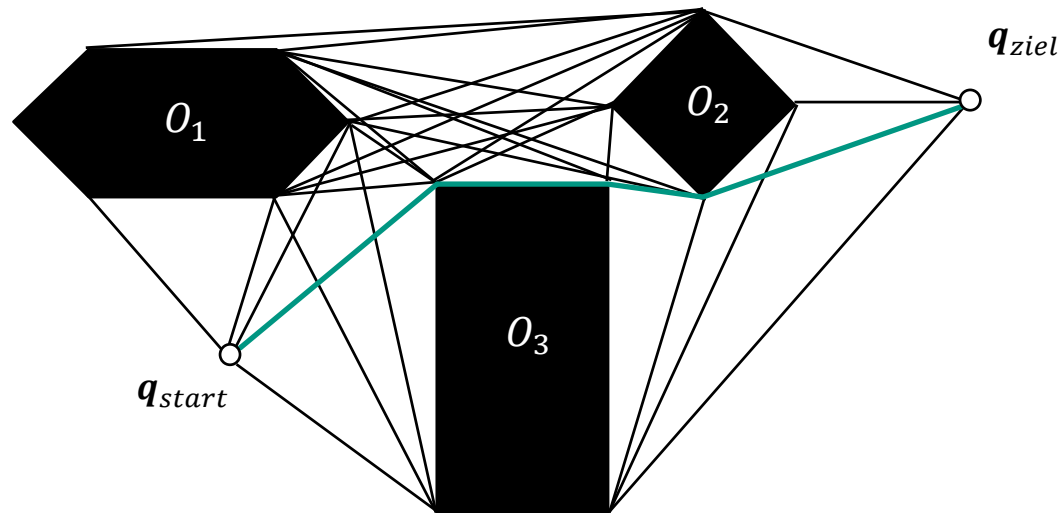
- In der Regel ist der **Weg nicht der kürzeste**.
- Bei **wenigen Hindernissen** werden nur **wenige Wege** generiert.

Sichtgraphen: Konstruktion

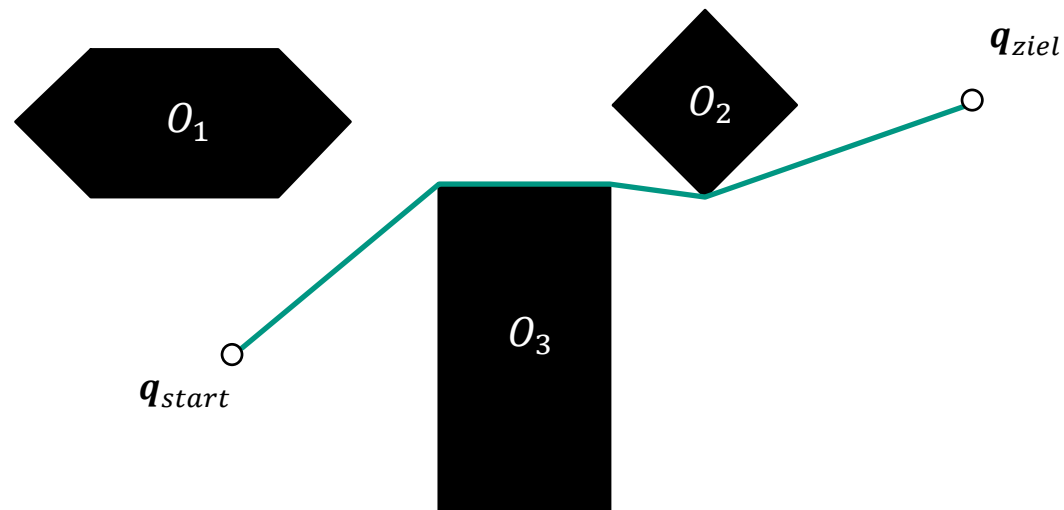
- **Verbinde** jedes Paar von **Eckpunkten** auf dem **Rand** von $\mathcal{C}_{\text{free}}$ durch ein gerades Liniensegment, wenn das Segment **kein Hindernis** schneidet
- Verbinde q_{start} und q_{ziel} analog



Sichtgraphen: Beispiel I



Sichtgraphen: Beispiel II



Sichtgraphen: Vor- und Nachteile

■ Vorteile:

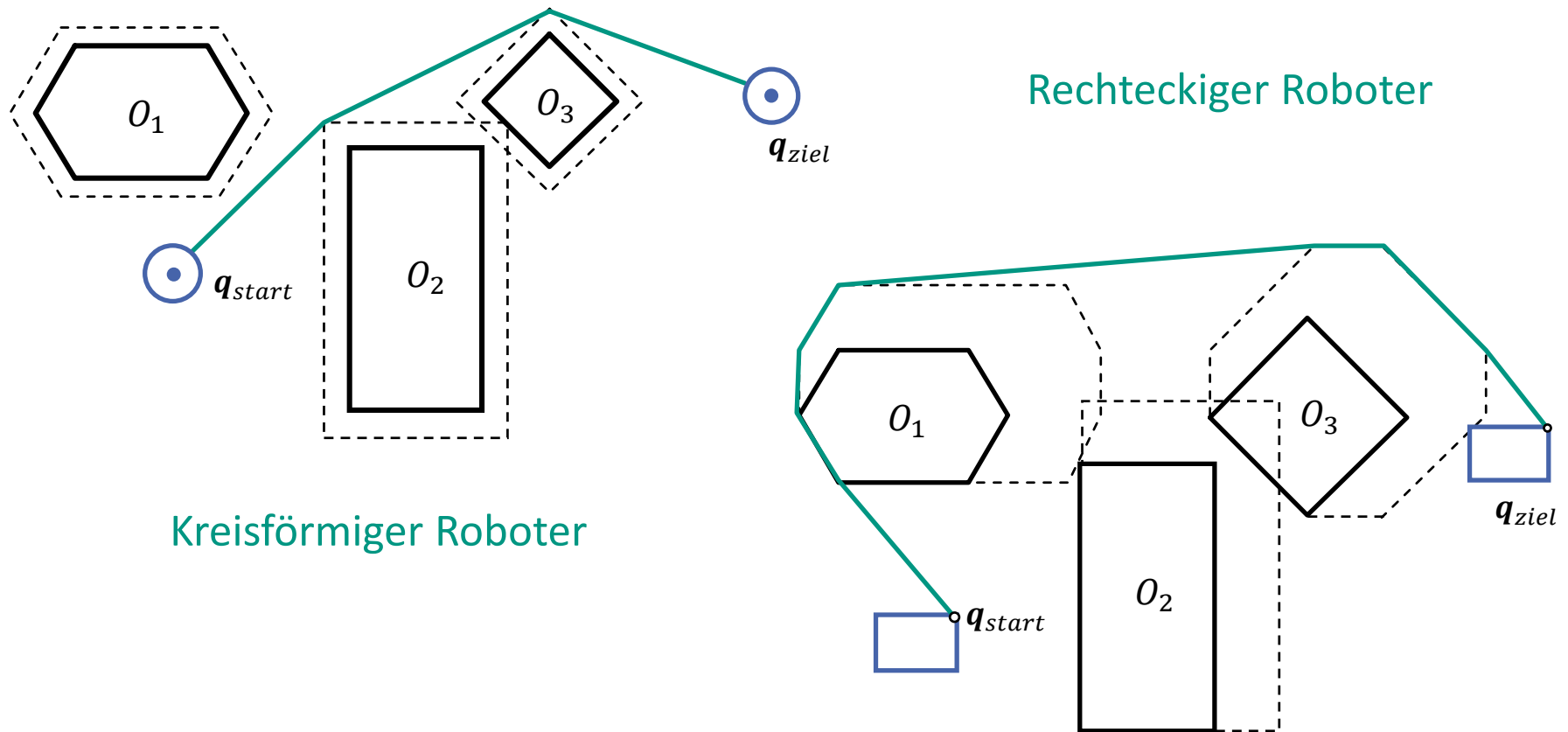
- Wenn ein **Weg gefunden** ist, ist es auch der **kürzeste** Weg.
- Methode ist **exakt**, wenn nur **zwei translatorische Freiheitsgrade** existieren und sowohl Roboter als auch Hindernisse durch **konvexe Polygone** dargestellt werden können.

■ Nachteile:

- Wege sind nicht zwingend kollisionsfrei, da **Hinderniskanten** auch **Wegsegmente** sein können.
- Abhilfe: **Erweiterung der Hindernisse**.
- Methode auch **im $\mathbb{R}^3$** anwendbar, jedoch sind die gefundenen Wege i. A. **nicht die kürzesten Wege**.

Sichtgraphen: Erweiterung der Hindernisse

- Hindernisse werden um Roboterform erweitert



Zellzerlegung

■ Vorgehen:

1. Zerlege C_{free} in Zellen, so dass ein Weg zwischen zwei Konfigurationen innerhalb einer Zelle leicht zu finden ist
2. Stelle die Nachbarschaft (Adjazenz) in einem Graphen dar
3. Suche den optimalen Weg von $\mathbf{q}_{start}$ nach $\mathbf{q}_{ziel}$ in dem Graphen

■ Es gibt zwei Zerlegungsarten:

- Exakte Zerlegung
- Approximative Zerlegung

Exakte Zellzerlegung

■ Zerlegung des Freiraumes C_{free} in Zellen Z_i , so dass:

■ Die Zellen sich nicht überlappen

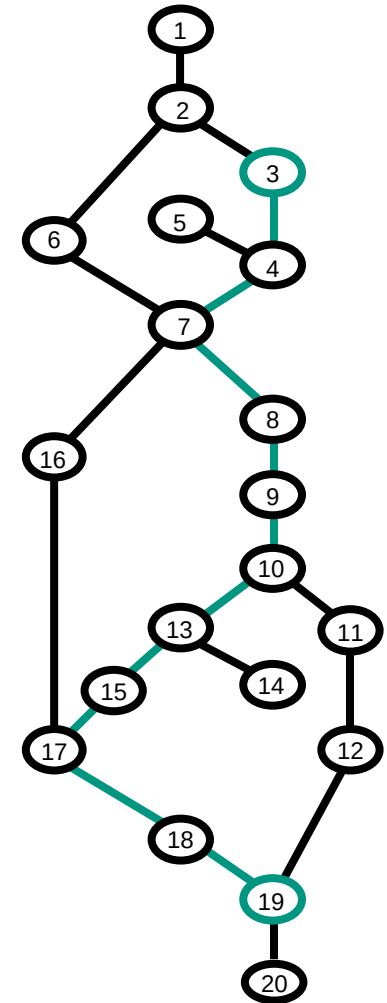
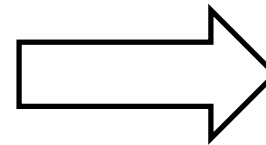
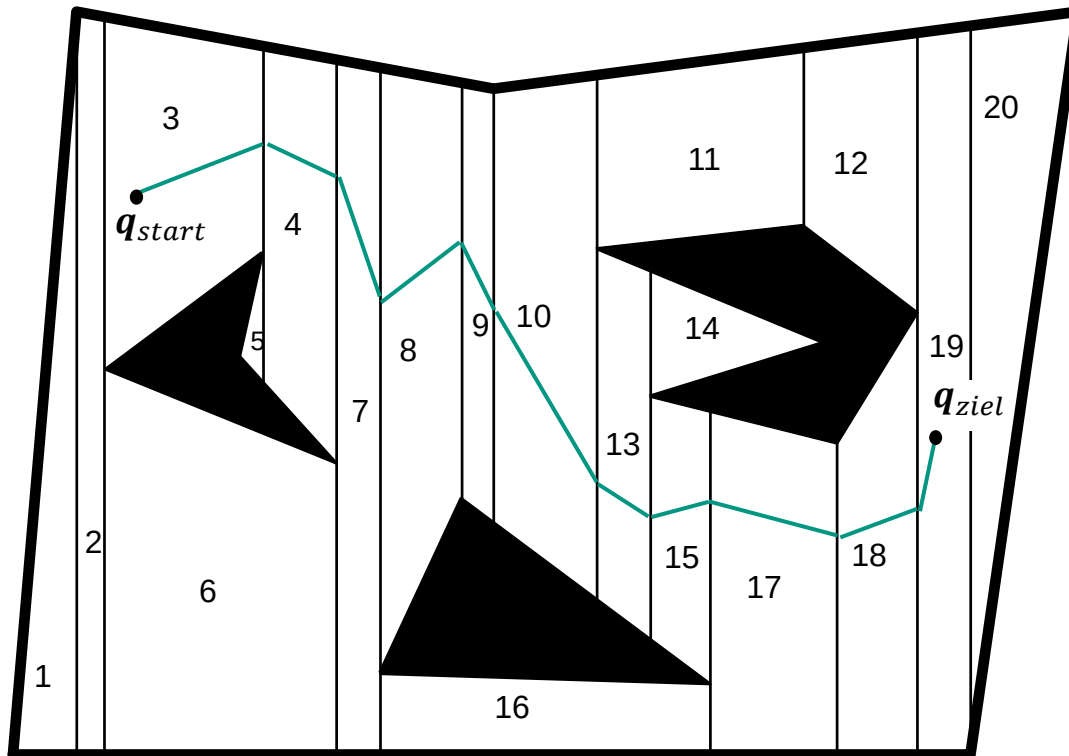
$$\forall i, k, i \neq k: Z_i \cap Z_k = \emptyset$$

■ Die Vereinigungsmenge aller Z_i ist C_{free}

$$\bigcup_{i=1}^n Z_i = C_{free}$$

Exakte Zellzerlegung: Beispiel

■ Exakte Zellzerlegung mit Line-Sweep



Approximative Zellzerlegung

■ Vorgehen:

1. Zerlege den Freiraum C_{free} in Zellen von vordefinierter Form (z.B. rechteckig)
2. Wenn eine Zelle nicht vollständig in C_{free} liegt, verringere die Größe und zerlege die Zelle weiter (z.B. Quadtree)
3. Wende diesen Schritt bis zu einer Minimalgröße der Zellen an

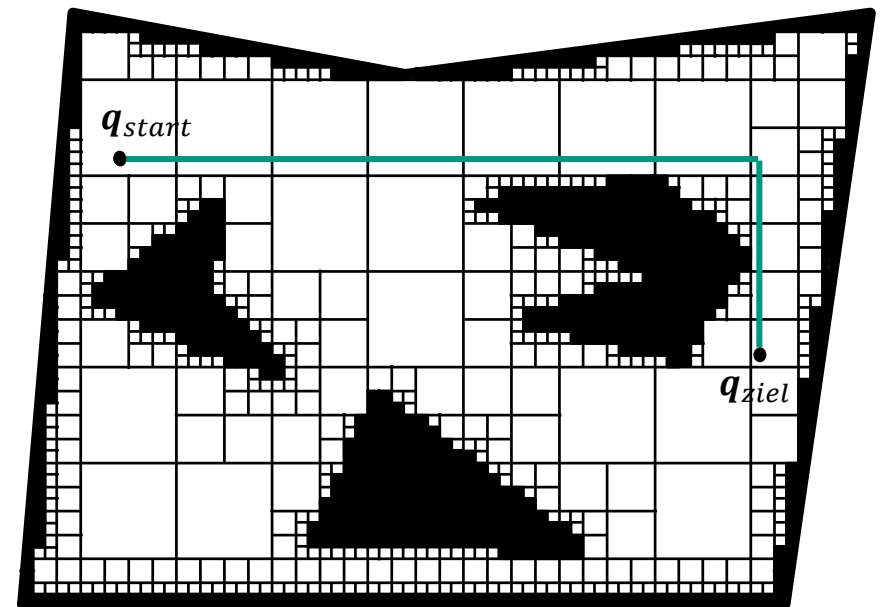
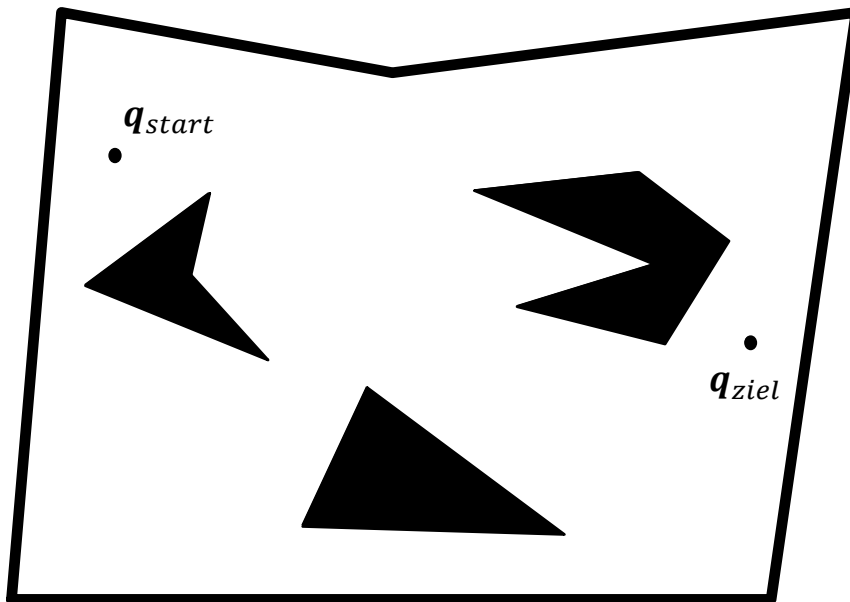
■ Vorteil

- Einfache Zerlegung und damit einfachere Wegsuche

■ Nachteil

- Der Freiraum kann i.A. nur annähernd beschrieben werden

Approximative Zellzerlegung: Beispiel



Pfadplanung für mobile Roboter: 2 Problem

1. Konstruktion des Wegenetzes W

- Retraktionsverfahren, z.B. Voronoi-Diagramm
- Sichtgraphen
- Zellzerlegung

2. Suche in W

- Baumsuche
- A*

Baumsuche

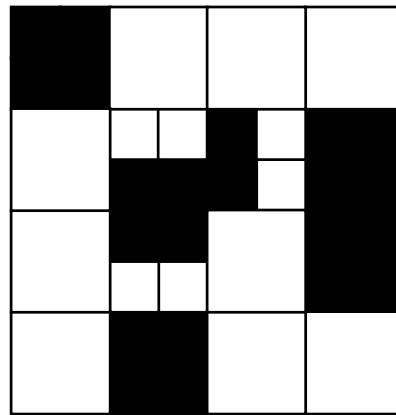
- Anwendungsfall:
 - Mobiler Roboter
 - 2D-Arbeits- und Konfigurationsraum

- Darstellung des Konfigurationsraums als **Quadtree**
 - Rekursive Unterteilung des Konfigurationsraums in Kacheln
 - Kacheln sind entweder frei oder ein Hindernis

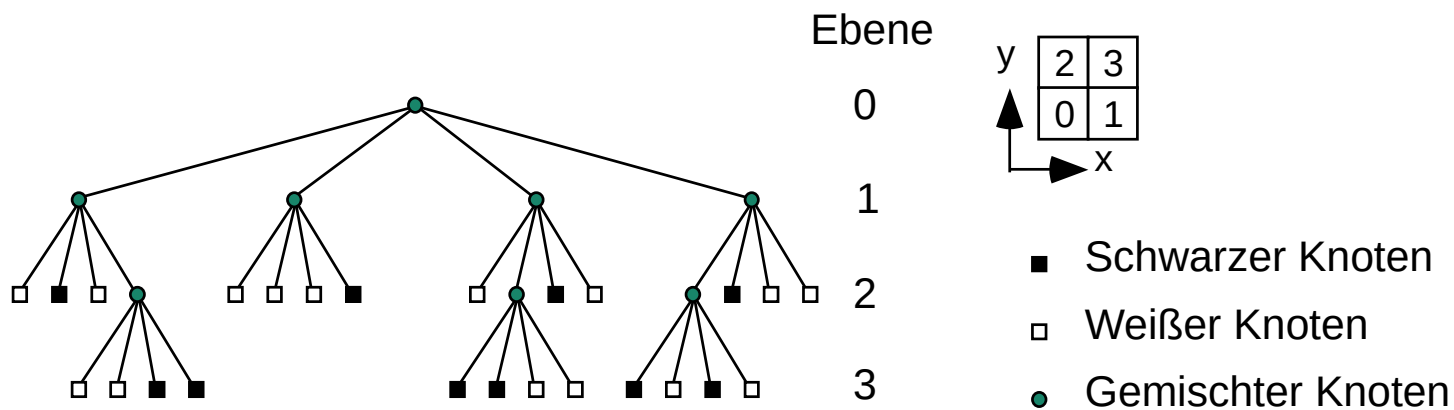
- Bewegungsplanung:
 - Kacheln finden, in denen sich Start- bzw. Zielkonfiguration befinden
 - Benachbarte freie Kacheln des Baums vom Start zum Ziel verbinden
 - Kollisionsfreie Routenplanung durch freie Kacheln

Baumsuche: Quadtree I

Darstellung des Konfigurationsraums als Quadtree

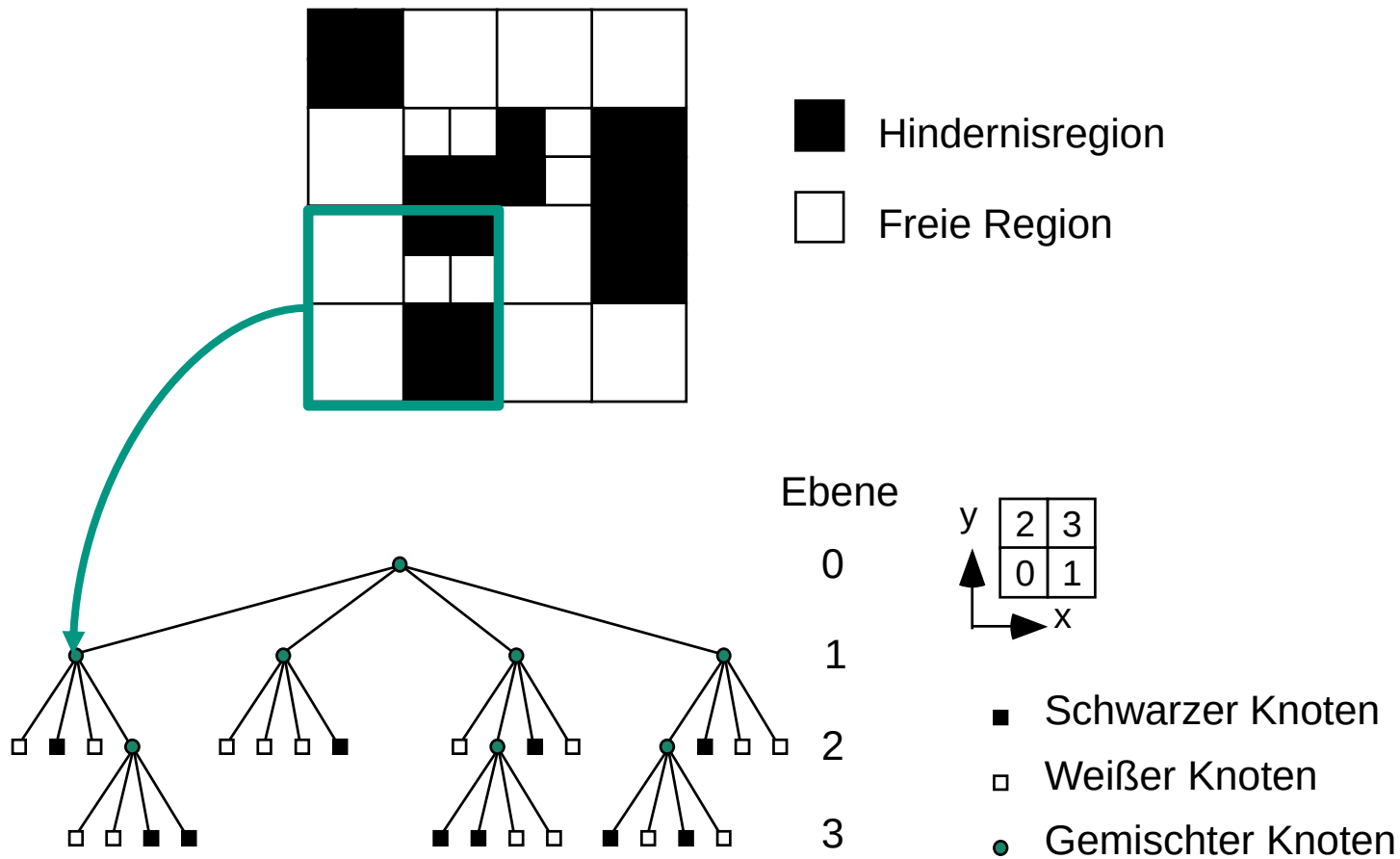


- Hindernisregion
- Freie Region



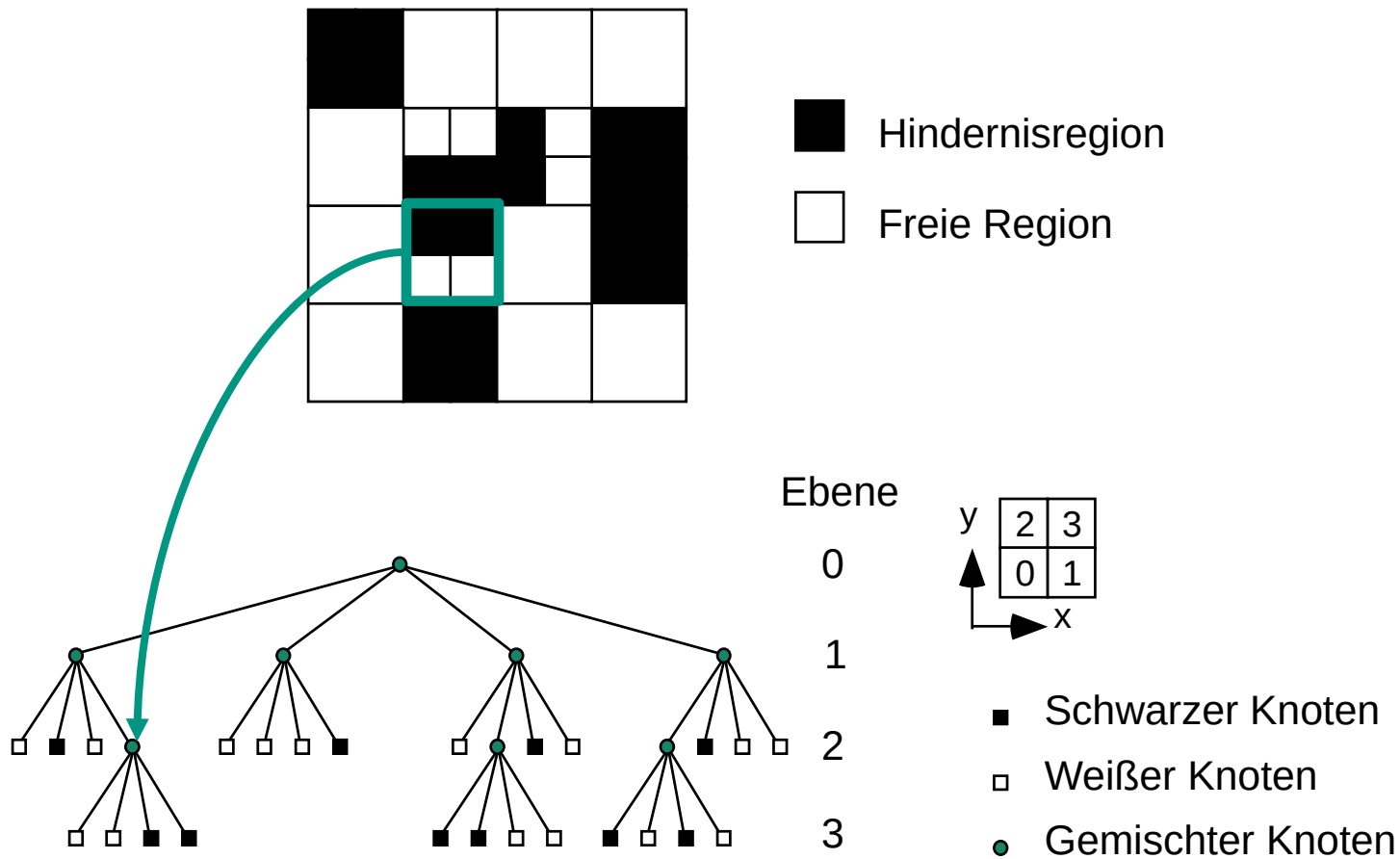
Baumsuche: Quadtree II

Beispiel: Ebene 1, Bereich 2



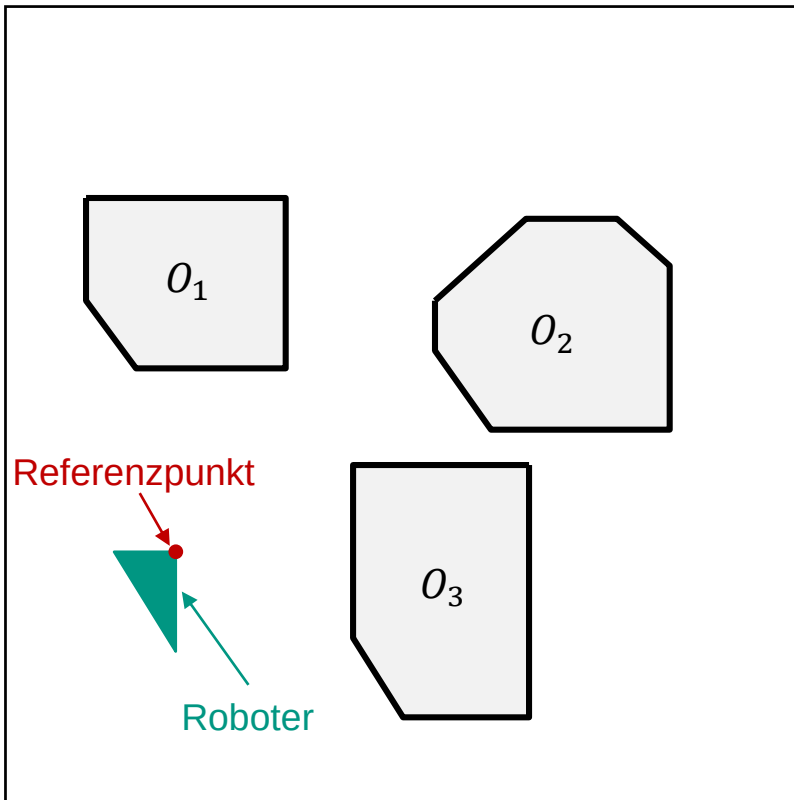
Baumsuche: Quadtree III

Beispiel Ebene 2, Bereich 1



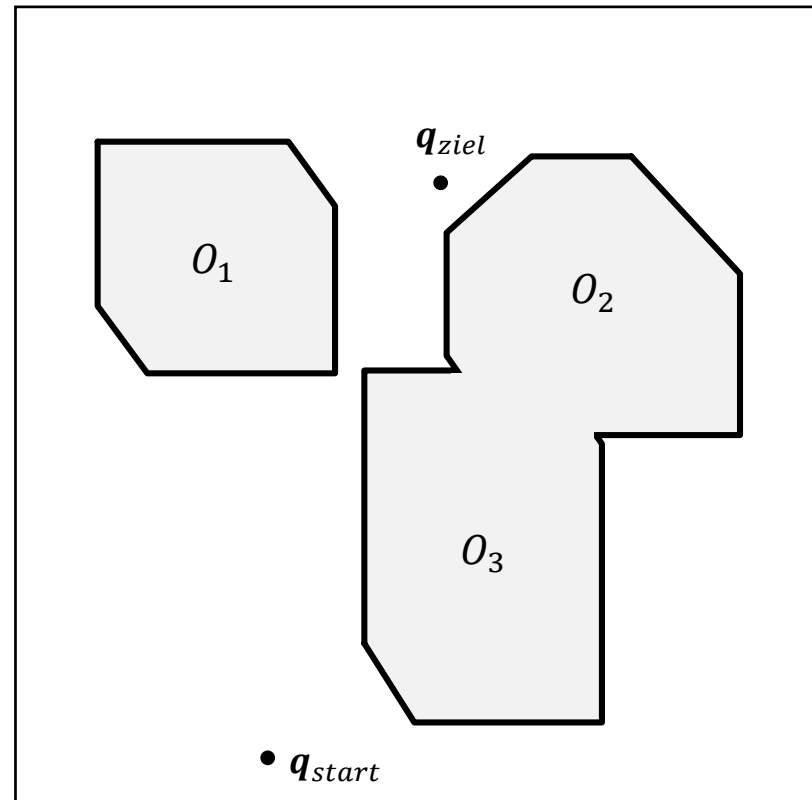
Baumsuche: Beispiel I

Arbeitsraum



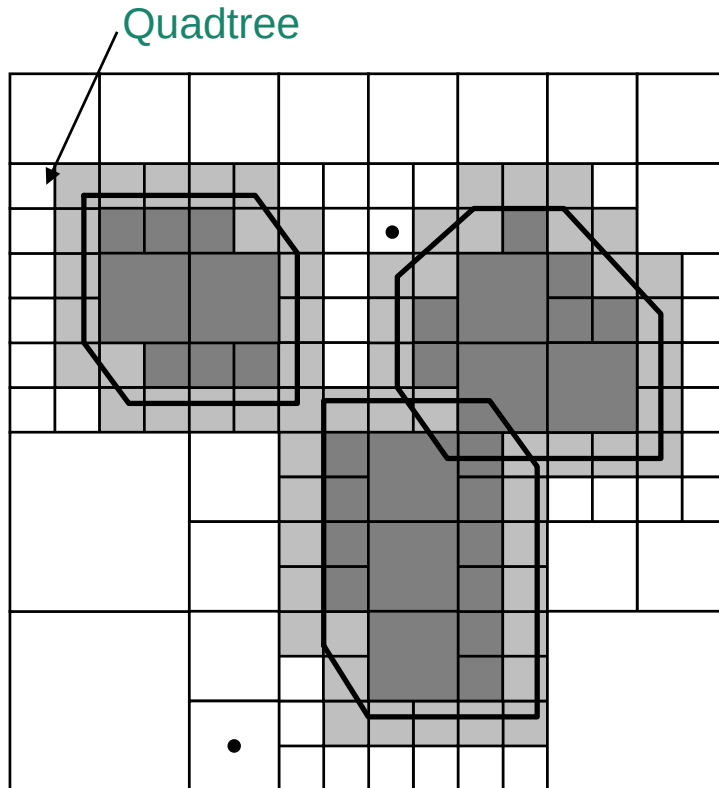
Arbeitsraum eines Roboters mit Hindernissen

Konfigurationsraum

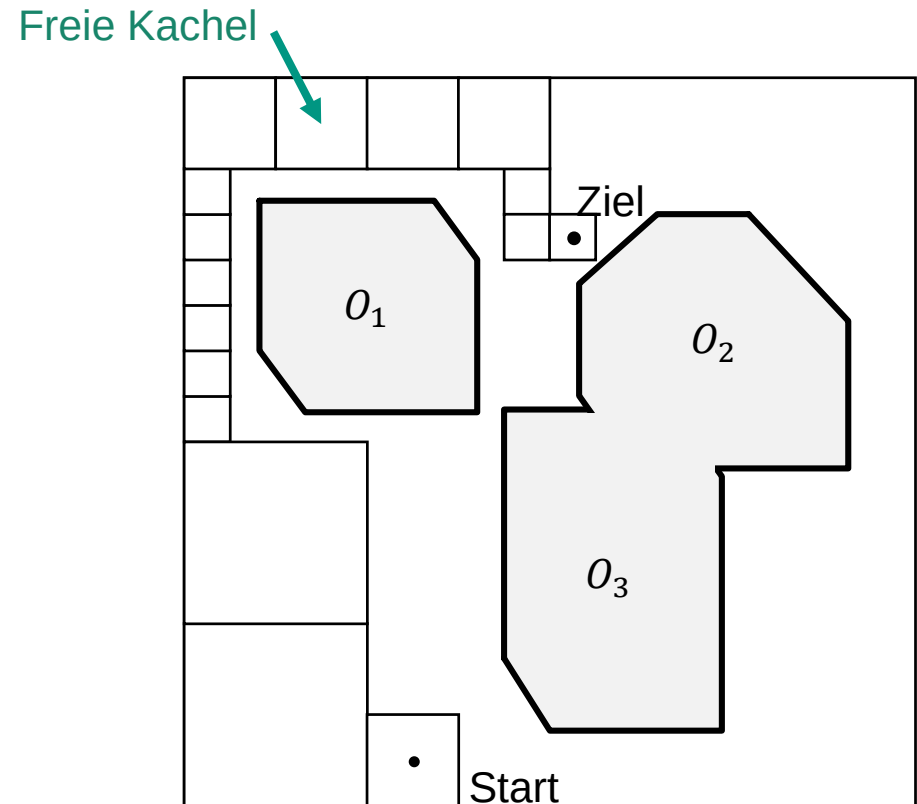


Konfigurationsraum für das vorliegende Robotersystem

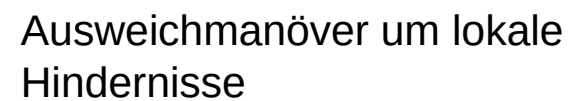
Baumsuch: Beispiel II



Zerlegung des Konfigurationsraums
in Kachelzonen



Gesucht: Folge von freien Kacheln
vom Start- zum Zielpunkt



Robotik I: Einführung in die Robotik

Bewegungsplanung

Raphael Grimm, Fabian Paus, Tamim Asfour

KIT-Fakultät für Informatik, Institut für Anthropomatik und Robotik (IAR)
Hochperformante Humanoide Technologien (H²T)



Pfadplanung für mobile Roboter: 2 Probleme

1. Konstruktion des Wegenetzes W

- Retraktionsverfahren, z.B. Voronoi-Diagramm
- Sichtgraphen
- Zellzerlegung

2. Suche in W

- Baumsuche
- A*-Algorithmus

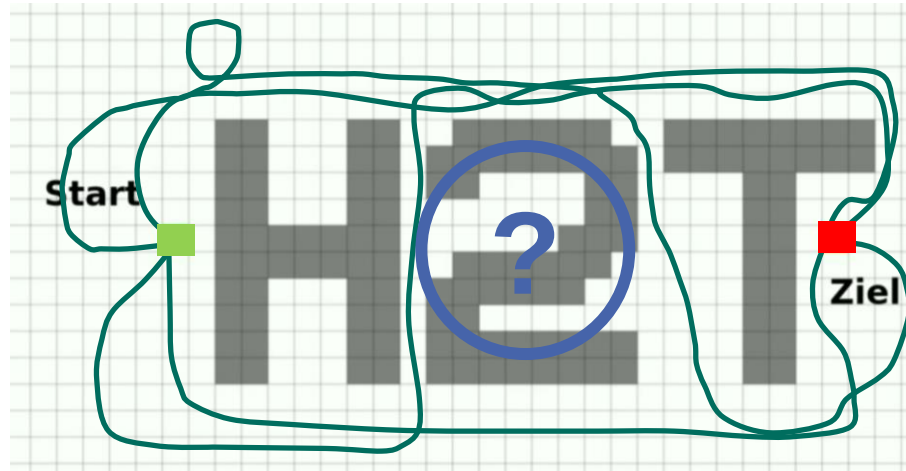
A*-Algorithmus

- Motivation: Kürzester Pfad von Start nach Ziel



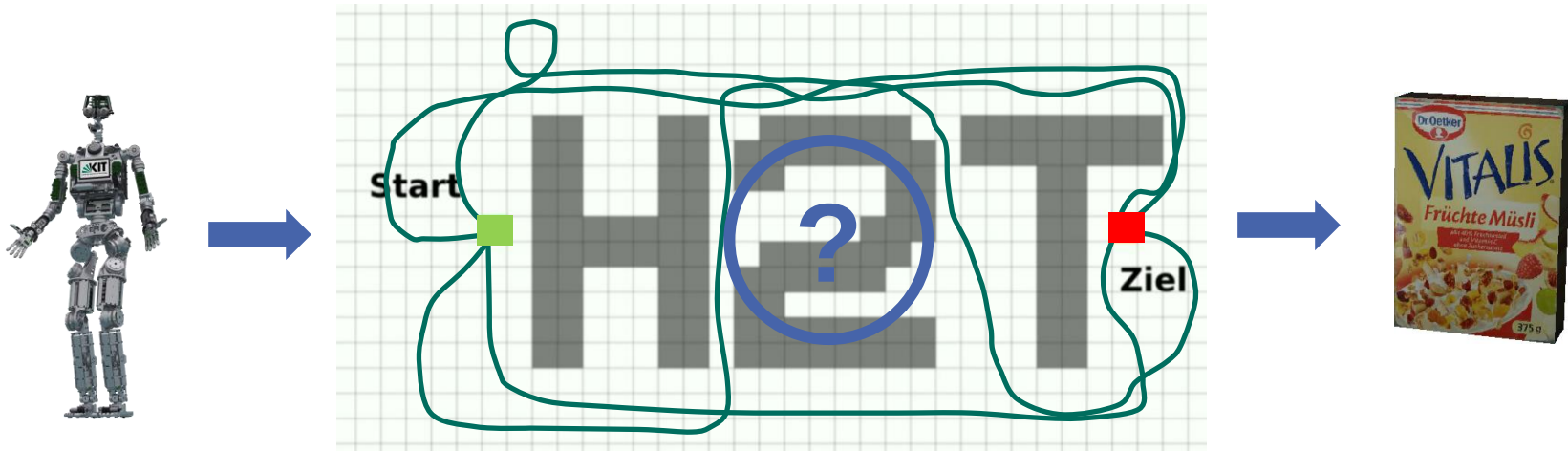
A*-Algorithmus

- Motivation: Kürzester Pfad von Start nach Ziel



A*-Algorithmus

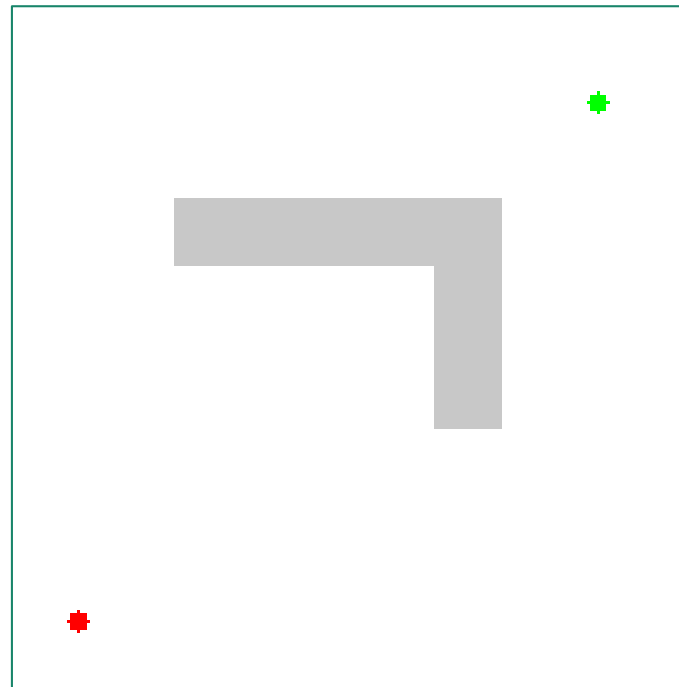
- Motivation: Kürzester Pfad von Start nach Ziel



- A* ist einer der beliebtesten Algorithmen zur Routenplanung
- Kostenfunktion ist $f(x) = g(x) + h(x)$
 - $g(x)$ entspricht Kosten von Start-Knoten nach Knoten x
 - $h(x)$ entspricht **geschätzten** Kosten von Knoten x nach Ziel-Knoten

A*-Algorithmus

- A* (“A Stern”) ist ein Algorithmus zur **Bestensuche**
- Findet den optimalen Pfad von einem Startknoten v_{start} zu einem Zielknoten v_{ziel}
- Optimalität in Bezug auf die **Pfadkosten** (z.B. kürzester Weg, kürzeste Zeit, kleinste Kantengewichte, usw.)



https://de.wikipedia.org/wiki/A*-Algorithmus

A*-Algorithmus

- Iterativer Ansatz
- Es werden zwei Knotenlisten verwaltet
 - **Open Set** O : Noch zu besuchende Knoten
 - **Closed Set** C : Bereits besuchte Knoten
- **Update**: Für einen besuchten Knoten v_n :
 - **Vorgängerknoten** $pred(v_n)$
 - **Akkumulierte Kosten**, um v_n zu erreichen: $g(v_n)$
 - **Heuristik** für die erwarteten Kosten zum Ziel: $h(v_n)$
- **Initialisierung**
 - $O = \{v_{start}\}$
 - $C = \{\}$
 - $g(v_i) = \infty, \quad 1 \leq i \leq K$
 - $g(v_s) = 0$

A*-Algorithmus

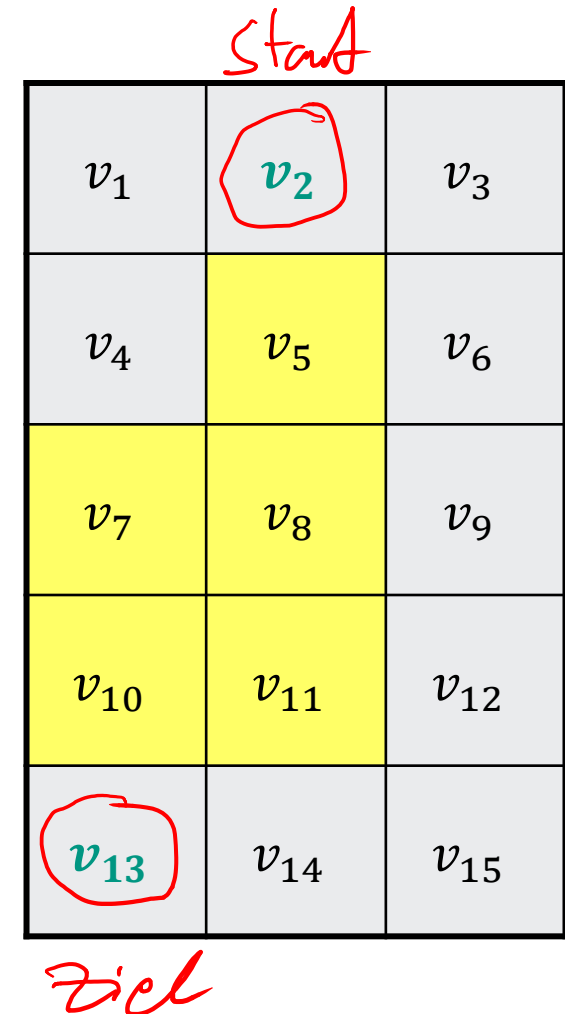
■ Algorithmus

Solange $O \neq \emptyset$

- Bestimme den zu erweiternden Knoten
 - Finde $v_i \in O$ mit minimalem $f(v_i) = g(v_i) + h(v_i)$
- Wenn $v_i = v_{\text{ziel}}$
Lösung gefunden: Traversiere Vorgänger von v_i bis v_{start} erreicht ist.
- $O.\text{remove}(v_i)$
- $C.\text{add}(v_i)$
- **Update** für alle Nachfolger v_j von v_i durchführen
 - Wenn $v_j \in C$, dann überspringe v_j
 - Wenn $v_j \notin O$, dann $O.\text{add}(v_j)$
 - Wenn $g(v_i) + \text{cost}(v_i, v_j) < g(v_j)$
 - $g(v_j) = g(v_i) + \text{cost}(v_i, v_j)$
 - $h(v_j) = \text{heuristic}(v_j, v_{\text{ziel}})$
 - $\text{pred}(v_j) = v_i$

A*-Algorithmus: Beispiel

- Gitter mit 15 Knoten
- Finde den optimalen Pfad von v_2 nach v_{13}
 - Nur horizontale und vertikale Bewegungen erlaubt
 - Kosten:
 - Betreten einer grauen Zelle: 1
 - Betreten einer gelben Zelle: 4
 - Heuristik h : Euklidische Distanz zu v_{13}
(z.B. $h(v_{11}) = \sqrt{2}$)



A*-Algorithmus: Beispiel

■ Initialisierung:

$$O = \{v_2\}$$

$$C = \{\}$$

$$f(v_2) = 0 + h(v_2)$$

$$= 0 + \sqrt{4^2 + 1^2} = \sqrt{17}$$

v_1	v_2	v_3
v_4	v_5	v_6
v_7	v_8	v_9
v_{10}	v_{11}	v_{12}
v_{13}	v_{14}	v_{15}

A*-Algorithmus: Beispiel, Initialisierung

■ Initialisierung:

■ $O = \{v_2\}$

■ $f(v_2) = 0 + h(v_2) = \sqrt{4^2 + 1^2} \approx 4.12$

■ $C = \{\}$

v_1	v_2	v_3
v_4	v_5	v_6
v_7	v_8	v_9
v_{10}	v_{11}	v_{12}
v_{13}	v_{14}	v_{15}

A*-Algorithmus: Beispiel, Schritt 1

- Zustand:
 - $O = \{v_2\}$ $\min_v (f(v))$
 - $f(v_2) = 0 + h(v_2) = \sqrt{4^2 + 1^2} \approx 4.12$
 - $C = \{\}$

- Update:

Expandiere v_2

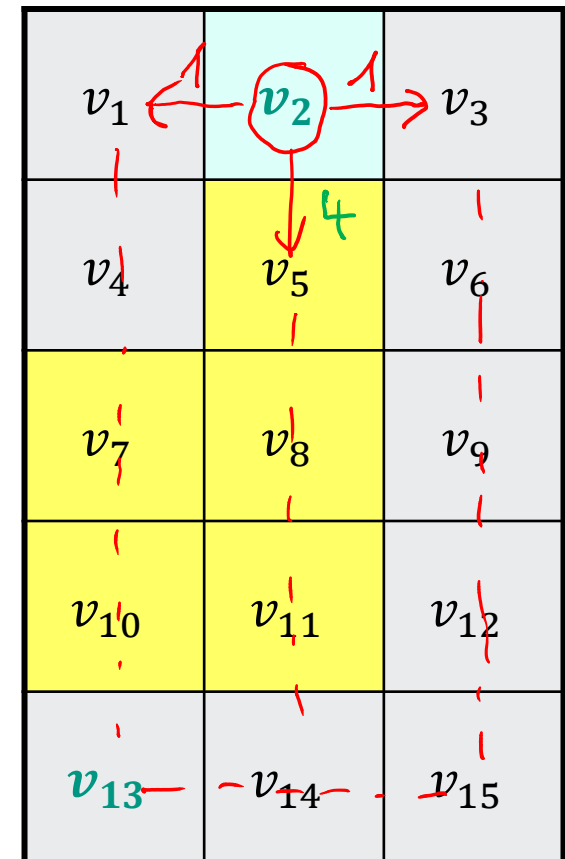
$$O = \{v_1, v_3, v_5\}$$

$$C = \{v_2\}$$

$$f(v_1) = 1 + 4 = 5$$

$$f(v_3) = 1 + \sqrt{4^2 + 2^2} = 1 + \sqrt{20}$$

$$f(v_5) = 4 + \sqrt{3^2 + 1^2} = 4 + \sqrt{10}$$



A*-Algorithmus: Beispiel, Schritt 1

■ Zustand:

■ $O = \{v_2\}$

■ $f(v_2) = 0 + h(v_2) = \sqrt{4^2 + 1^2} \approx 4.12$

■ $C = \{\}$

■ Update:

■ Expandiere v_2

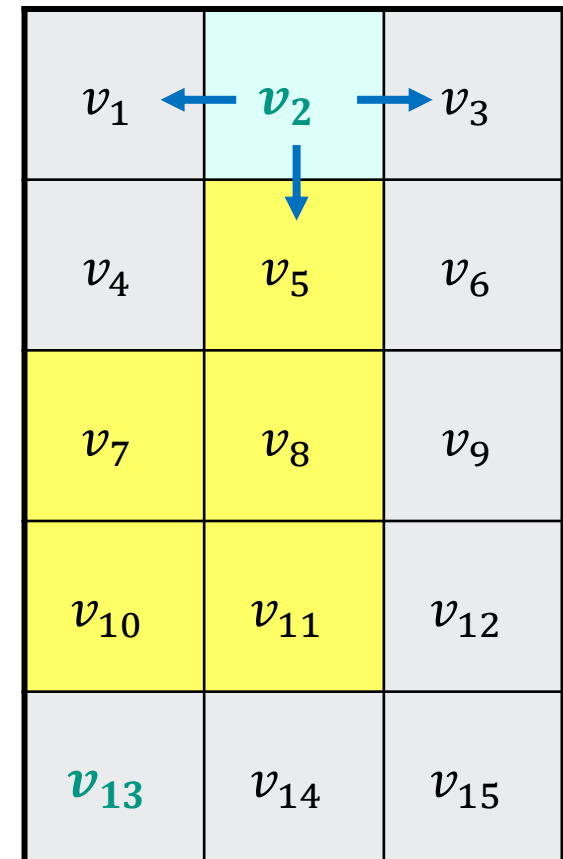
■ $O = \{v_1, v_3, v_5\}$

■ $f(v_1) = 1 + h(v_1) = 1 + 4 = 5$

■ $f(v_3) = 1 + h(v_3) = 1 + \sqrt{4^2 + 2^2} \approx 5.47$

■ $f(v_5) = 1 + h(v_5) = 4 + \sqrt{3^2 + 1^2} \approx 7.16$

■ $C = \{v_2\}$



A*-Algorithmus: Beispiel, Schritt 2

Zustand:

$$O = \{v_1, v_3, v_5\}$$

$$f(v_1) = 5$$

$$f(v_3) \approx 5.47$$

$$f(v_5) \approx 7.16$$

$$C = \{v_2\}$$

$\min_v f(v)$

Update:

Expandiere v_1

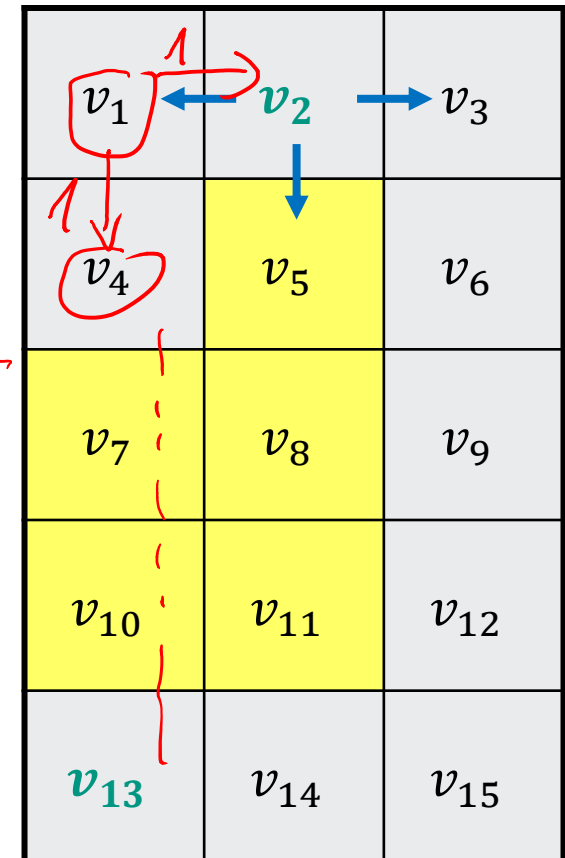
$$O = \{v_3, v_5, v_4\}$$

$$C = \{v_2, v_1\}$$

$$f(v_4) = 2 + 3 = 5$$

5

$$h(v_4) = 3$$



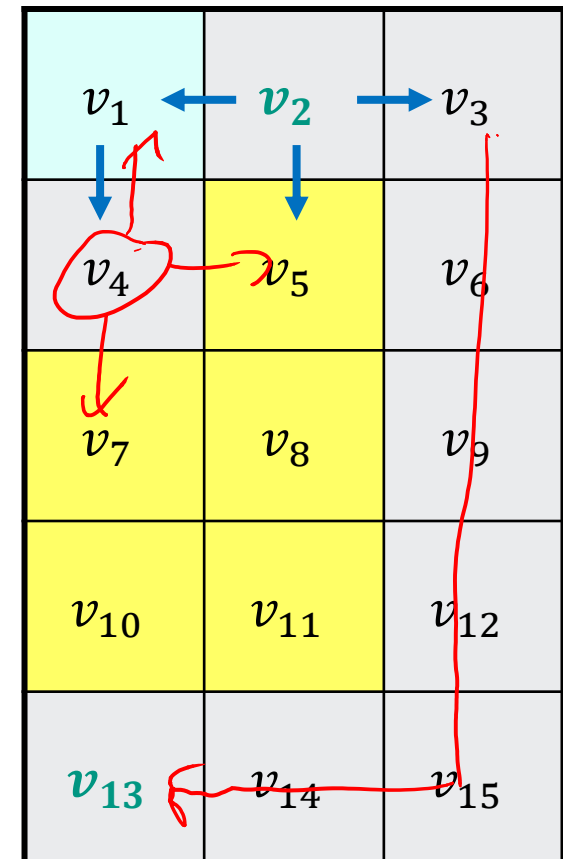
A*-Algorithmus: Beispiel, Schritt 2

■ Zustand:

- $O = \{v_1, v_3, v_5\}$
 - ~~$f(v_1) = 5$~~
 - $f(v_3) \approx 5.47$
 - $f(v_5) \approx 7.16$
- $C = \{v_2\}$

■ Update:

- Expandiere v_1
- $O = \{v_1, v_3, v_5, v_4\}$
 - $f(v_4) = 2 + h(v_4) = 2 + 3 = 5$
- $C = \{v_2, v_1\}$



A*-Algorithmus: Eigenschaften

- Findet eine **optimale Lösung**, wenn Heuristik h **zulässig**
 - Heuristik h ist **zulässig**, wenn sie die minimalen Kosten, das Ziel zu erreichen, nicht überschätzt
- A* ist auch **optimal effizient** für jede (zulässige) Heuristik h
 - Kein optimaler Algorithmus, der die gleiche Heuristik verwendet, besucht weniger Knoten als A*
- Wenn $\forall x: h(x) = 0$: **Dijkstra's Algorithmus**, d.h. $f = g$
 - Greedy Algorithmus: beachtet die Entfernung zum Ziel nicht
 - Besucht mehr Knoten als notwendig

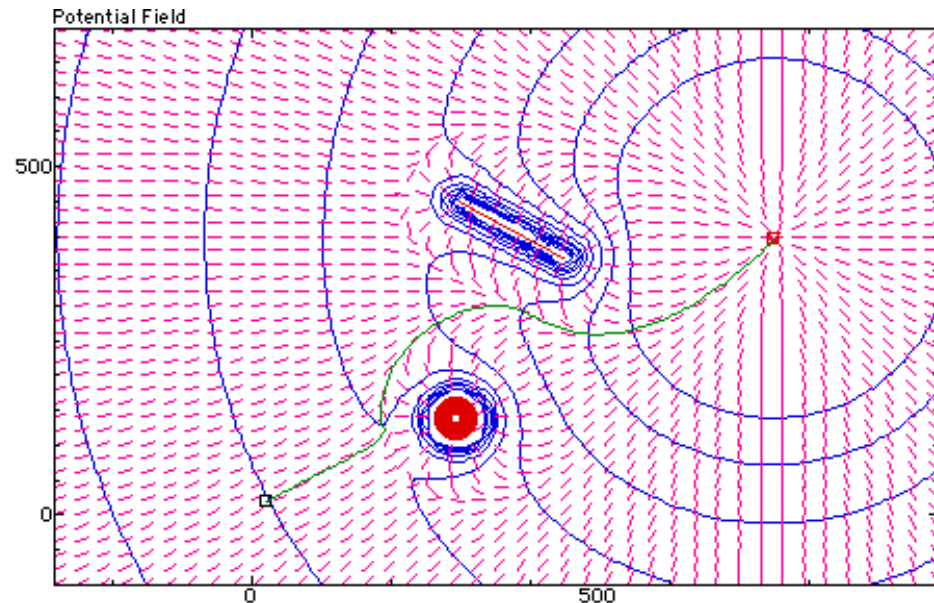
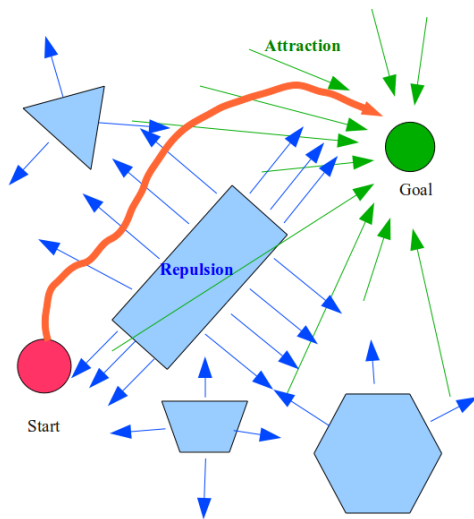
Inhalt

- Motivation
- Grundlagen der Bewegungsplanung
- Pfadplanung für mobile Roboter
 - Graphenbasiert
 - Potentialfelder
- Bewegungsplanung für Manipulatoren

Potentialfeld-Methode

Methode entwickelt für

- Bewegungsplanung [Kathib 1986]
- SLAM bei mobilen Roboter, d.h. [Prestes 2002]



O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots" , International Journal on Robotics Research (IJRR), 5(1):90--98, Spring, 1986

Potentialfelder (1)

- Der Roboter bewegt sich unter dem Einfluss von Kräften, welche ein Potentialfeld auf ihn ausübt

- **Definition:**

- Ein Potentialfeld U ist eine Skalarfunktion über dem Freiraum

$$U: \underline{C_{free}} \rightarrow \mathbb{R}$$

- Die Kraft in einem Punkt q des Potentialfeldes ist der negative Gradient in diesem Punkt

$$F(q) = -\nabla U(q)$$

Potentialfelder (2)

■ Abstoßendes Potenzial

- Hindernisse erzeugen ein abstoßendes Potential
- In großem Abstand zu Hindernissen ($> \rho_0$) soll der Roboter nicht beeinflusst werden

■ Beispiel:

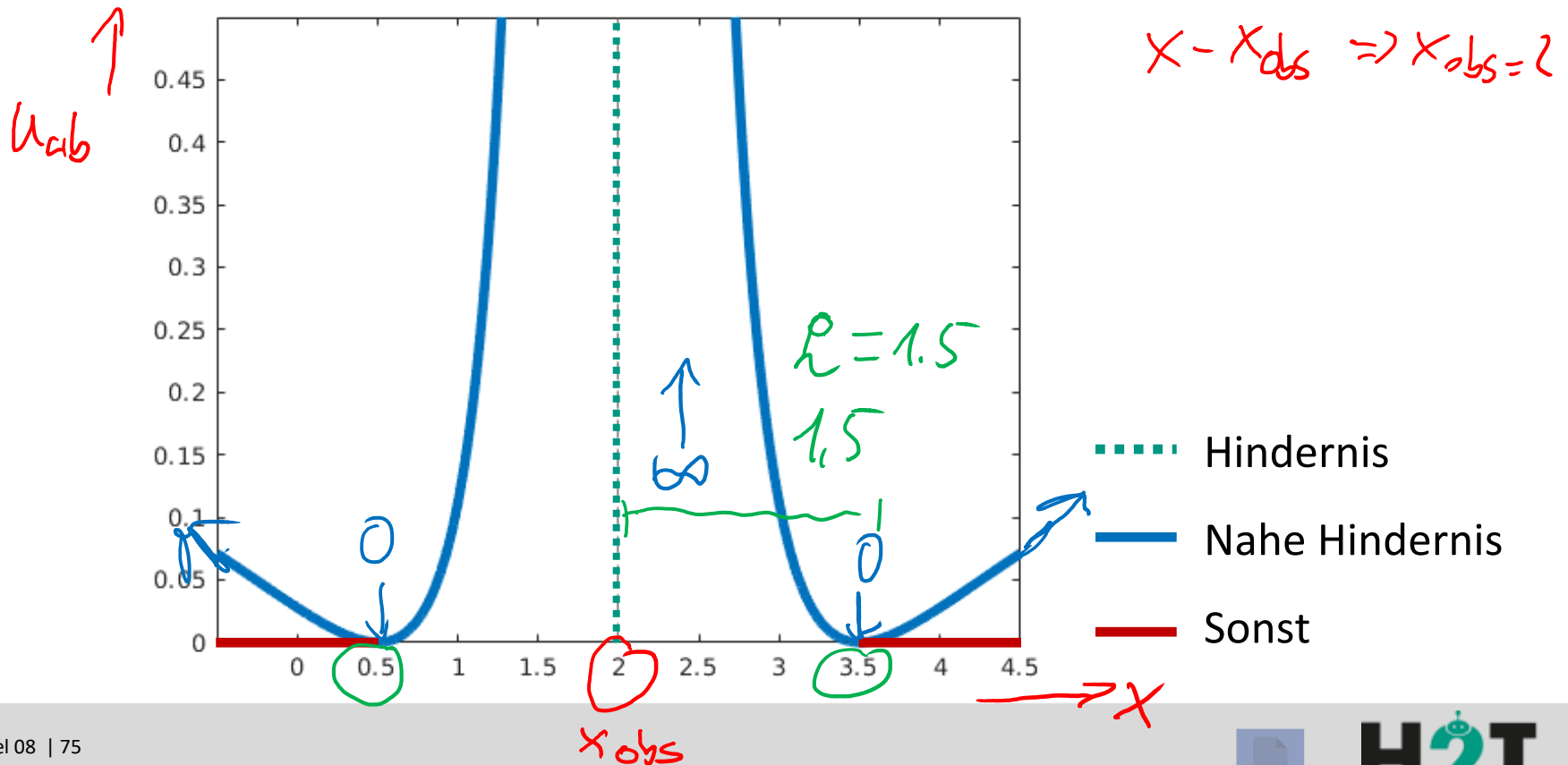
$$U_{ab}(\mathbf{q}) = \begin{cases} \frac{1}{2} v \left(\frac{1}{\rho(\mathbf{q}, \mathbf{q}_{obs})} - \frac{1}{\rho_0} \right)^2 & \text{für } \rho(\mathbf{q}, \mathbf{q}_{obs}) \leq \rho_0 \\ 0 & \text{sonst} \end{cases}$$

$\rho(\mathbf{q}, \mathbf{q}_{obs}) = \|\mathbf{q} - \mathbf{q}_{obs}\|$ ist die minimale Distanz zwischen Roboter und Hindernis

$$F_{ab} = -\nabla U_{ab} = v \left(\frac{1}{\rho(\mathbf{q}, \mathbf{q}_{obs})} - \frac{1}{\rho_0} \right) \cdot \frac{1}{\rho(\mathbf{q}, \mathbf{q}_{obs})^2} \cdot \frac{\mathbf{q} - \mathbf{q}_{obs}}{\rho(\mathbf{q}, \mathbf{q}_{obs})}$$

Potentialfeld: Hindernis-Beispiel

$$U_{ab}(x) = \begin{cases} \left(\frac{1}{\|x - 2\|} - \frac{1}{1.5} \right)^2 & \text{für } \|x - 2\| \leq 1.5 \\ 0 & \text{sonst} \end{cases}$$



Potentialfelder (3)

■ Anziehendes Potential

- Es soll möglichst nur ein Minimum in $\mathbf{q}_{ziel}$ geben

■ Lineare Funktion der Distanz zum Ziel:

$$U_{an}(\mathbf{q}) = k \cdot \|\mathbf{q} - \mathbf{q}_{ziel}\|$$

$$F_{an}(\mathbf{q}) = -\nabla U_{an}(\mathbf{q}) = -k \cdot \frac{\mathbf{q} - \mathbf{q}_{ziel}}{\|\mathbf{q} - \mathbf{q}_{ziel}\|}$$

- Für kleine Distanzen wird die Kraft sehr groß

Potentialfelder (4)

■ Quadratische Funktion der Distanz

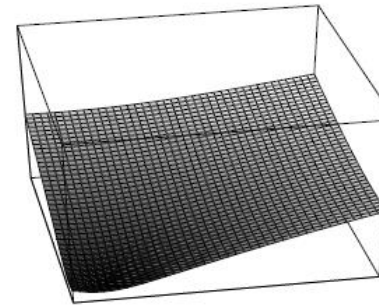
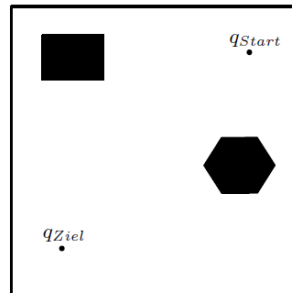
$$U_{an}(\mathbf{q}) = k \cdot \frac{1}{2} \|\mathbf{q} - \mathbf{q}_{ziel}\|^2$$

$$F_{an}(\mathbf{q}) = -\nabla U_{an}(\mathbf{q}) = -k \cdot (\mathbf{q} - \mathbf{q}_{ziel})$$

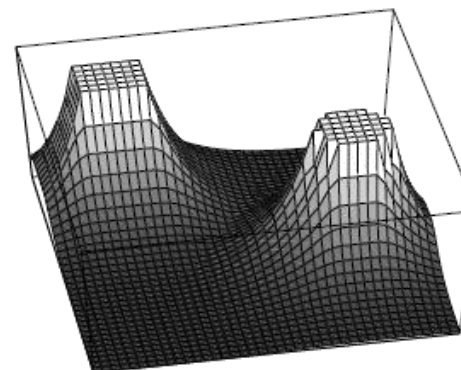
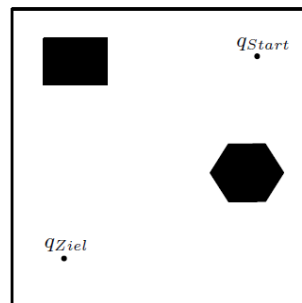
- Oft wird die Kombination aus linearer und quadratischer Funktion verwendet
 - Lineare Funktion, wenn weit vom Ziel entfernt
 - Quadratisch Funktion, wenn nah am Ziel

Potentialfelder: Beispiel

- Die **Zielstellung** q_{ziel} hat das anziehende Potential U_{an}

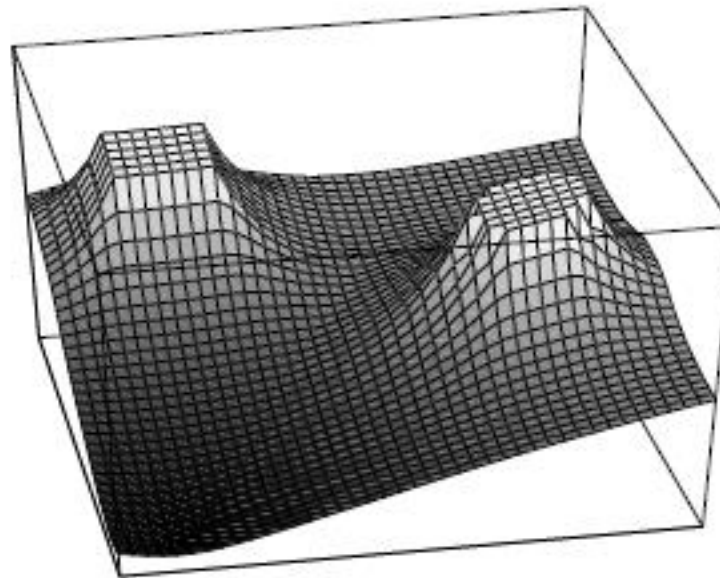


- Der **Hindernisraum** C_{obs} hat das abstoßende Potential U_{ab}



Potentialfelder: Beispiel

- Die Summe der einwirkenden Kräfte bestimmt die Richtung der Bewegung.
- Für das Potentialfeld gilt: $U(\mathbf{q}) = U_{an}(\mathbf{q}) + U_{ab}(\mathbf{q})$
- Für das Kräftefeld gilt: $F(\mathbf{q}) = F_{an}(\mathbf{q}) + F_{ab}(\mathbf{q})$



Potentialfelder: Lokale Minima

■ Lokale Minima

Durch Summation von U_{an} und U_{ab} kann U lokale Minima besitzen. Wenn der Roboter sich in Richtung des negativen Gradienten des Potentialfeldes bewegt, kann er in einem solchen lokalen Minimum “steckenbleiben”.

■ Maßnahmen:

- U_{an} und U_{ab} so definieren, dass U kein lokales Minimum hat, außer in $\mathbf{q}_{ziel}$
- Im Suchalgorithmus Techniken zur “Flucht” aus lokalen Minima anwenden

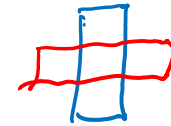
Inhalt

- Motivation
- Grundlagen der Bewegungsplanung
- Pfadplanung für mobile Roboter
- Bewegungsplanung für Manipulatoren
 - Probabilistic Roadmaps (PRM)
 - Dynamic Roadmaps (DRM)
 - Rapidly-exploring Random Trees (RRT)
 - Erweiterungen von RRT
 - Constrained RRT
 - RRT*
 - Enge Passagen
 - Dynamic Domain RRT
 - Bridge Sampling

Grundlagen der Bewegungsplanung: Begriffsbildung

■ Pfadplanung

- Starres Objekt (z.B. mobiler Roboter, autonomes Fahrzeug)
- 2D Problem (Position: x, y)
- 3D Problem (Position: x, y ; Rotation: α)
→ Piano Mover's Problem



■ Bewegungsplanung

- Mehrkörpersystem (z.B. Roboterarme, Systeme mit mehreren Robotern)
- Hochdimensionale Problemstellungen

■ Randbedingungen, auch Zwangsbedingungen

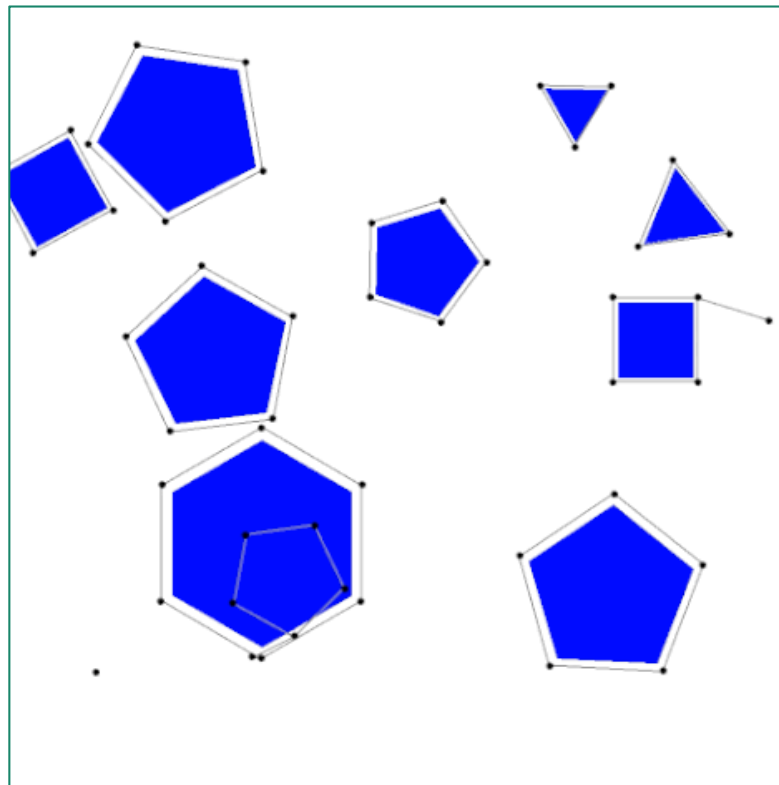
- Globale Randbedingungen: Limitieren den gültigen Konfigurationsraum
z.B. aufrechte Position des Endeffektors, maximale Motorströme, etc.
- Lokale Randbedingungen: Schränken die Übergänge zwischen Konfigurationen ein z.B.
Nicht-holonome Fahrzeuge, max. Geschwindigkeit/Beschleunigung

Inhalt

- Motivation
- Grundlagen der Bewegungsplanung
- Pfadplanung für mobile Roboter
- Bewegungsplanung für Manipulatoren
 - Probabilistic Roadmaps (PRM)
 - Dynamic Roadmaps (DRM)
 - Rapidly-exploring Random Trees (RRT)
 - Erweiterungen von RRT
 - Constrained RRT
 - RRT*
 - Enge Passagen
 - Dynamic Domain RRT
 - Bridge Sampling

Probabilistic Roadmaps (PRM)

- Kavraki, L. E.; Svestka, P.; Latombe, J.-C.; Overmars, M. H. (1996), "Probabilistic roadmaps for path planning in high-dimensional configuration spaces", IEEE Transactions on Robotics and Automation, 12 (4): 566–580, doi:10.1109/70.508439



https://en.wikipedia.org/wiki/Probabilistic_roadmap

Probabilistic Roadmaps (PRM)

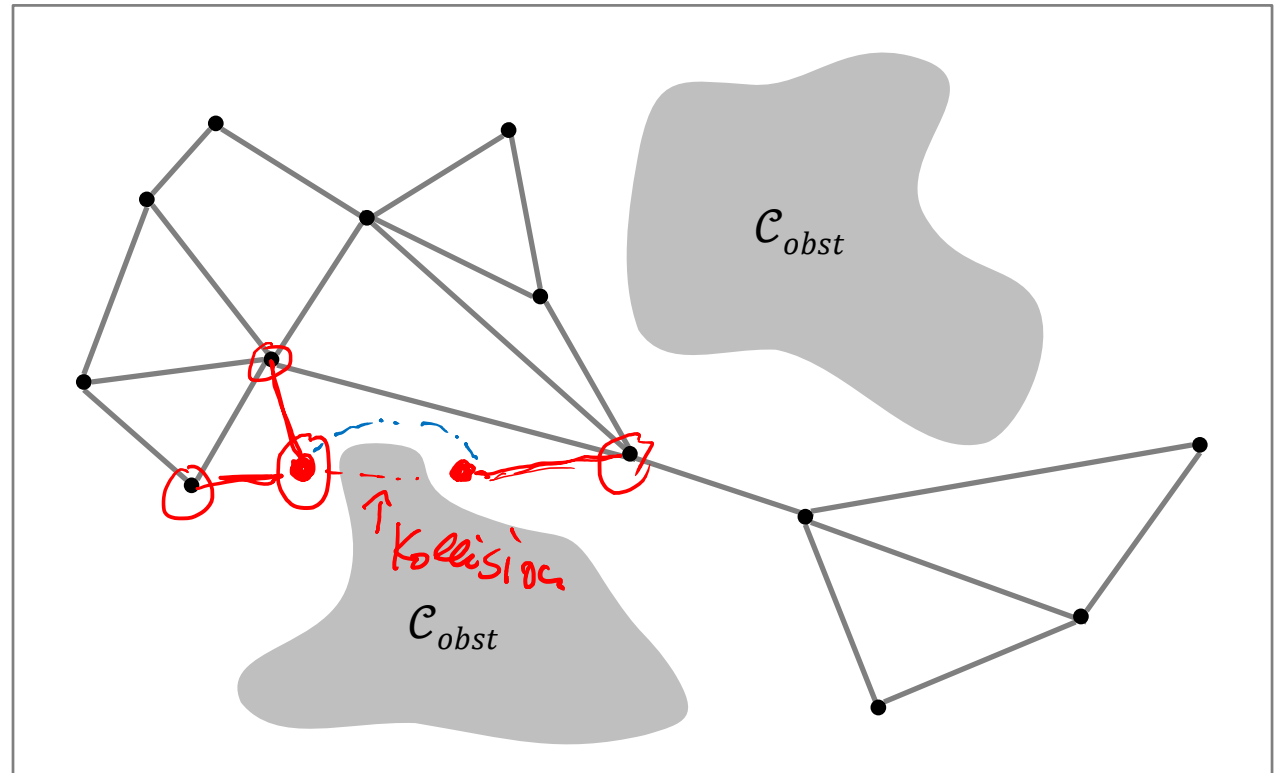
- Mehrere Anfragen (**multi-query**) in einer gleichbleibenden Umgebung
- **Approximation des Freiraumes** durch den Graphen (Roadmap)
 - Effizienter als die Erzeugung einer expliziten Repräsentation des Freiraumes (C_{free})

PRM Algorithmus

- **Schritt 1:** Vorverarbeitung
 - Erzeugung einer kollisionsfreien Graphen durch Wählen zufälliger Punkte (**Sampling**)
- **Schritt 2:** Anfrage
 - Verbinde q_{start} und q_{ziel} mit dem Graphen
 - Suche einen Weg von q_{start} nach q_{ziel} durch den Graphen

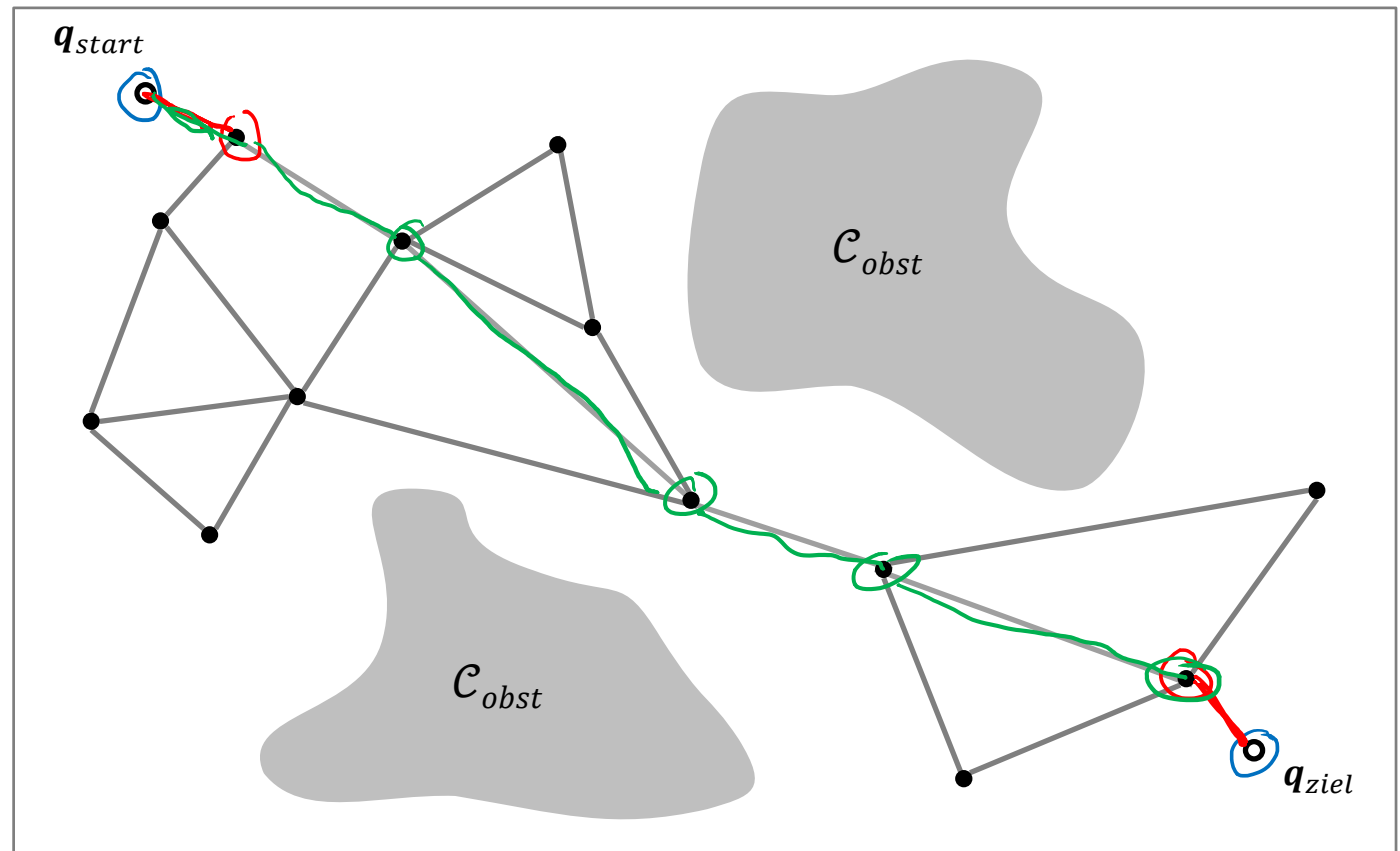
PRM: Vorverarbeitung

- Zufällige Erzeugung von kollisionsfreien Stichproben (Sampling)
- Lokale Planung: Stichproben werden über kollisionsfreie Pfade miteinander verbunden



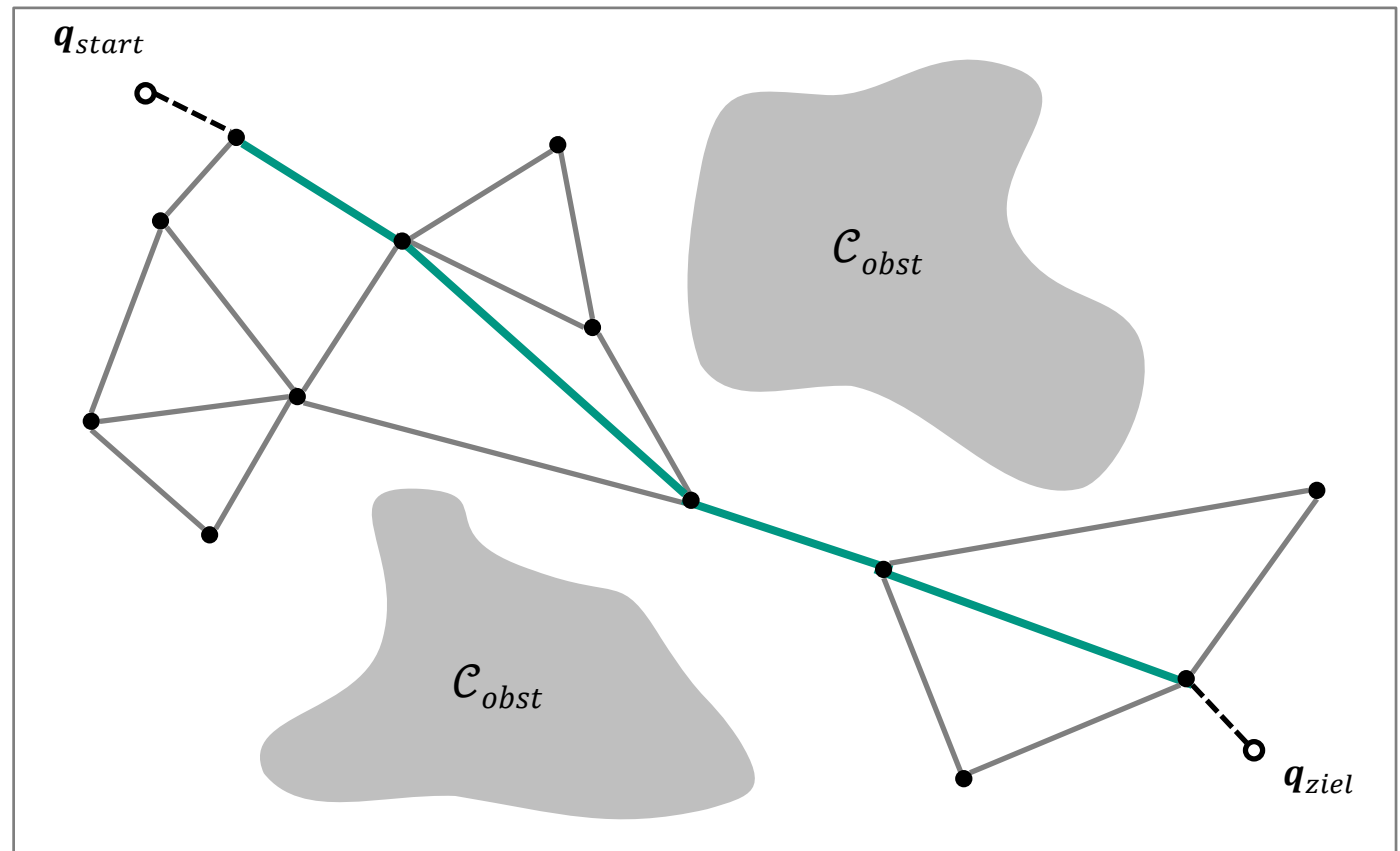
PRM: Anfrage

- Verbinde q_{start} und q_{ziel} mit dem Wegenetz
- Suche im Graphen (z.B. mit A*)



PRM: Anfrage

- Verbinde q_{start} und q_{ziel} mit dem Wegenetz
- Suche im Graphen (z.B. mit A*)



PRM: Konstruktion des Graphen

- N : Anzahl der Knoten im Graphen
- R : PRM, Graph
- Algorithmus:
 - Erzeugen von N zufälligen Konfigurationen in C_{free}
 - Einfügen der erzeugten Konfigurationen als Knoten in R
 - Für jeden Knoten $v_i \in R$
 - Finde die k nächsten Nachbarn von v_i aus R : $\mathcal{N}(v_i)$
 - Für jeden Knoten $v \in \mathcal{N}(v_i)$
 - Wenn es einen (neuen) kollisionsfreien Pfad von v nach v_i gibt, dann füge die Kante (v, v_i) in R ein
- Ergebnis: R

$k=2$

Lokale Planung

PRM: Eigenschaften

- Einmalige Konstruktion des Graphen
 - Mehrere Anfragen können effizient bearbeitet werden (**multi-query**)
- Randomisierter Ansatz zur Konstruktion (**probabilistisch**)
 - Exponentieller Anstieg der Laufzeit mit der Dimension des Konfigurationsraums wird vermieden
- Verfahren hängt stark vom verwendeten Sampling ab
 - Problem: Schmale Passagen zwischen Hindernissen
 - Lösungsansatz: Sampling in der Nähe von Hindernissen erhöhen
- Nicht vollständig, da der Graph C_{free} nur approximiert
 - Ein möglicher Pfad wird nicht unbedingt gefunden
 - Lösungsansatz: Erweitere den Graphen so, dass er zusammenhängend ist und jeder Punkt aus C_{free} von einem Knoten aus direkt erreichbar ist

PRM: Unterschiedliche Sampling-Strategien

■ Zufällig:

- Konfiguration wird zufällig generiert und auf Kollision geprüft

■ Grid:

- Konfigurationen werden mit diskreter Auflösung erzeugt
- Auflösung einzelner Zellen wird hierarchisch bestimmt

■ Halton:

- Halton-Menge: Menge von Punkten, die ein Bereich besser abdeckt als Grid
- Basiert auf dem mathematischen Konzept der Diskrepanz

■ Zellenbasiert:

- Sampling in Zellen mit kleiner werdenden Ausmaßen
- Zellgröße wird mit jeder Iteration verkleinert (z. B. auf $1/8$)

Geraerts, Roland, and Mark H. Overmars. "A comparative study of probabilistic roadmap planners." Algorithmic Foundations of Robotics V. Springer Berlin Heidelberg, 2004. 43-57.

Inhalt

- Motivation
- Grundlagen der Bewegungsplanung
- Pfadplanung für mobile Roboter
- Bewegungsplanung für Manipulatoren
 - Probabilistic Roadmaps (PRM)
 - Dynamic Roadmaps (DRM)
 - Rapidly-exploring Random Trees (RRT)
 - Erweiterungen von RRT
 - Constrained RRT
 - RRT*
 - Enge Passagen
 - Dynamic Domain RRT
 - Bridge Sampling

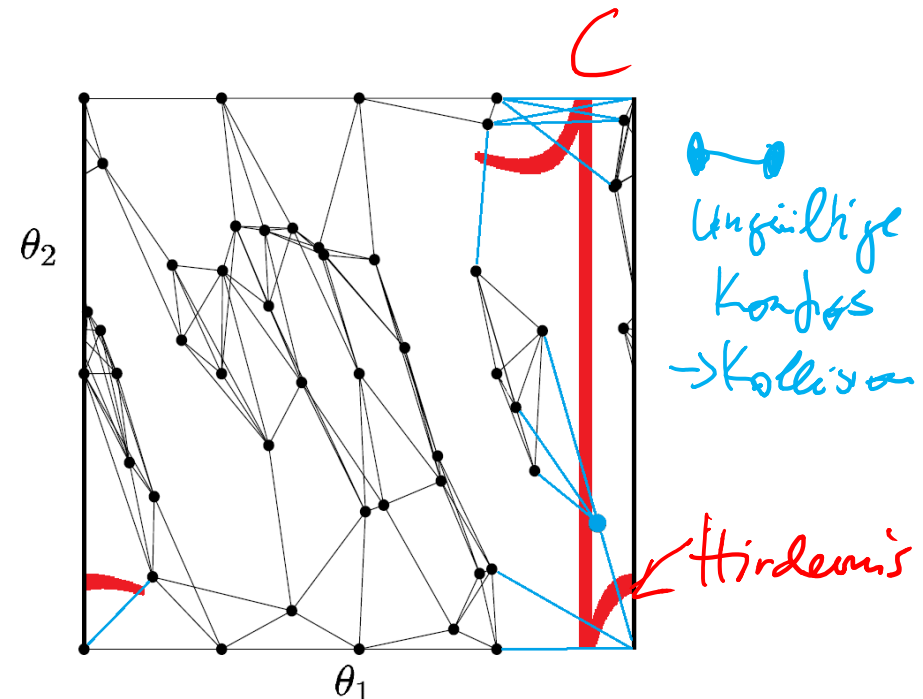
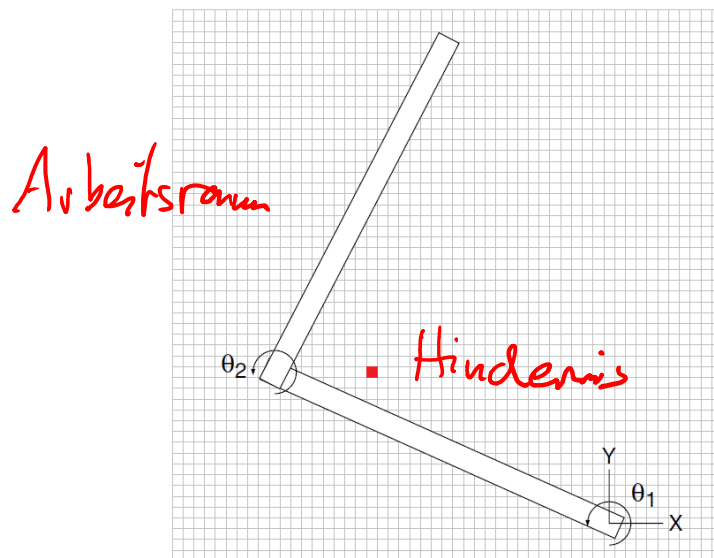
Dynamic Roadmaps (DRM)

- Mehrere Anfragen (**multi-query**) bei einer gleichbleibenden kinematischen Kette
- Vorverarbeitung
 - **Approximation des Konfigurationsraums** durch eine **Roadmap (Graph)**
 - **Approximation des Arbeitsraums** durch **Voxel** (Würfel)
 - **Abbildung ϕ_{wc} von Voxel $\rightarrow$ Roadmap (Knoten, Kanten)**
- Anfrage
 - Ermittle Voxel mit Hindernis
 - Anpassen der Roadmap
 - Planen in der angepassten Roadmap
- Kann direkt Punktwolken (**RGB-D**) nutzen

Dynamisch

DRM: Vorverarbeitung

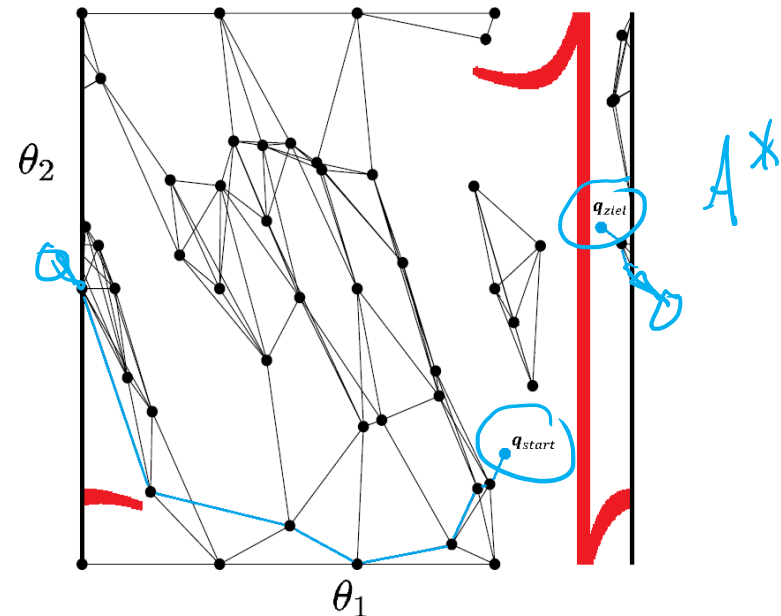
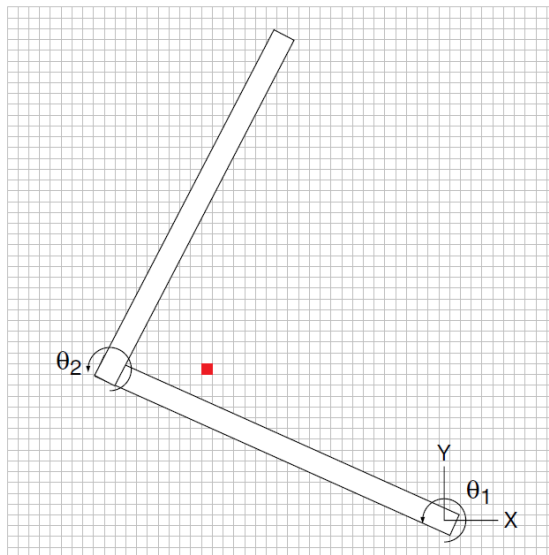
- Erzeugung einer selbstkollisionsfreien Roadmap durch Wählen zufälliger Punkte (**Sampling**)
 - Viele Punkte nötig, um in unterschiedlichen Umgebungen Lösungen zu finden
- Erzeugung der **Abbildung** ϕ_{wc} durch Kollisionsüberprüfung zwischen allen Knoten/Kanten und allen Voxeln
 - Sehr rechenaufwändig



Leven, Peter, and Seth Hutchinson. "A framework for real-time path planning in changing environments." The International Journal of Robotics Research 21.12 (2002): 999-1030.

DRM: Anfrage

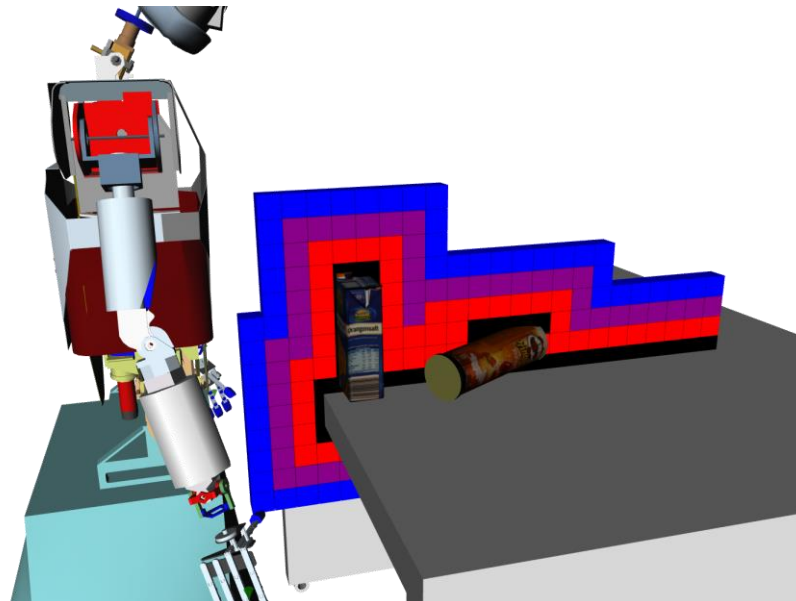
- Ermittle alle Voxel mit Hindernis
- Lösche alle zugehörigen Kanten und Knoten aus der Roadmap
 - Zugehörigen Kanten und Knoten werden mit ϕ_{wc} bestimmt
- Verbinde q_{start} und q_{ziel} mit dem Graphen
- Suche einen Weg von q_{start} nach q_{ziel} durch den Graphen



Leven, Peter, and Seth Hutchinson. "A framework for real-time path planning in changing environments." The International Journal of Robotics Research 21.12 (2002): 999-1030.

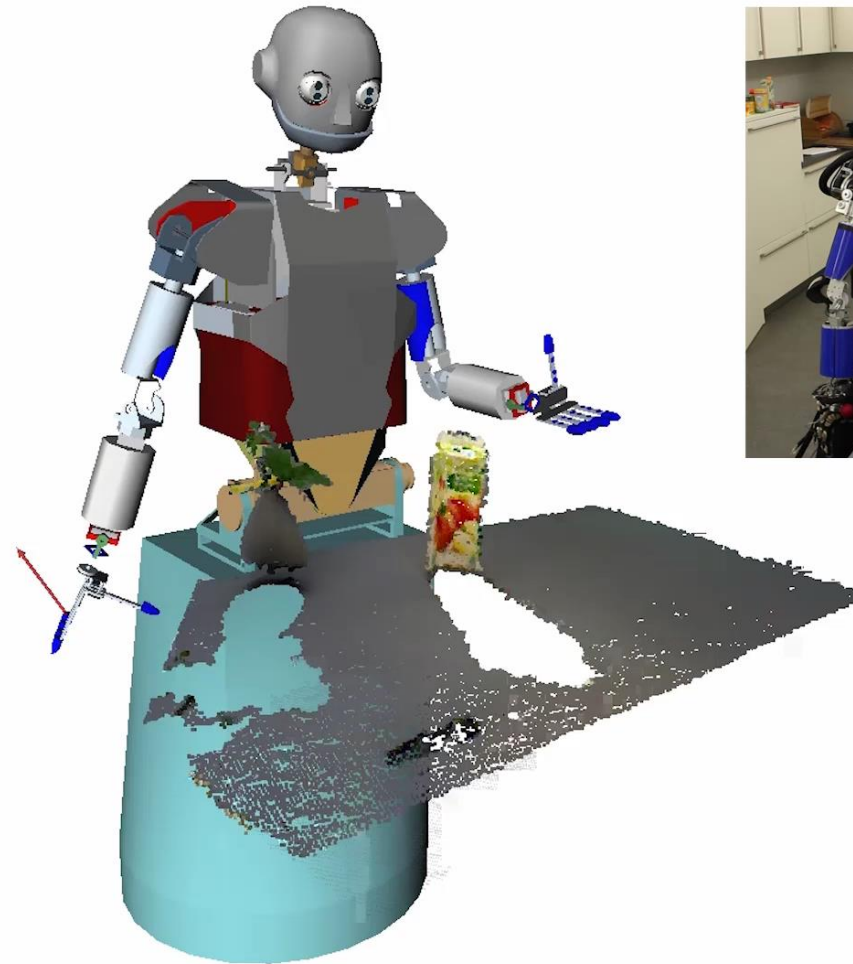
Distance Aware Dynamic Roadmap (DA-DRM)

- DRM sucht den kürzesten Weg → Bahn oft nahe an Objekten
 - Alle Voxel im **Sicherheitsabstand** löschen
 - Für Voxel nahe an Hindernissen Abstand zu Objekten berechnen
 - Kanten durch diese Voxel mit **höherem Gewicht** versehen
- Kürzester Pfad in der Roadmap ist im Arbeitsraum weiter von Objekten entfernt



Ausführung auf ARMAR-IIIa

2x



Inhalt

- Motivation
- Grundlagen der Bewegungsplanung
- Pfadplanung für mobile Roboter
- Bewegungsplanung für Manipulatoren
 - Probabilistic Roadmaps (PRM)
 - Dynamic Roadmaps (DRM)
 - Rapidly-exploring Random Trees (RRT)
 - Erweiterungen von RRT
 - Constrained RRT
 - RRT*
 - Enge Passagen
 - Dynamic Domain RRT
 - Bridge Sampling

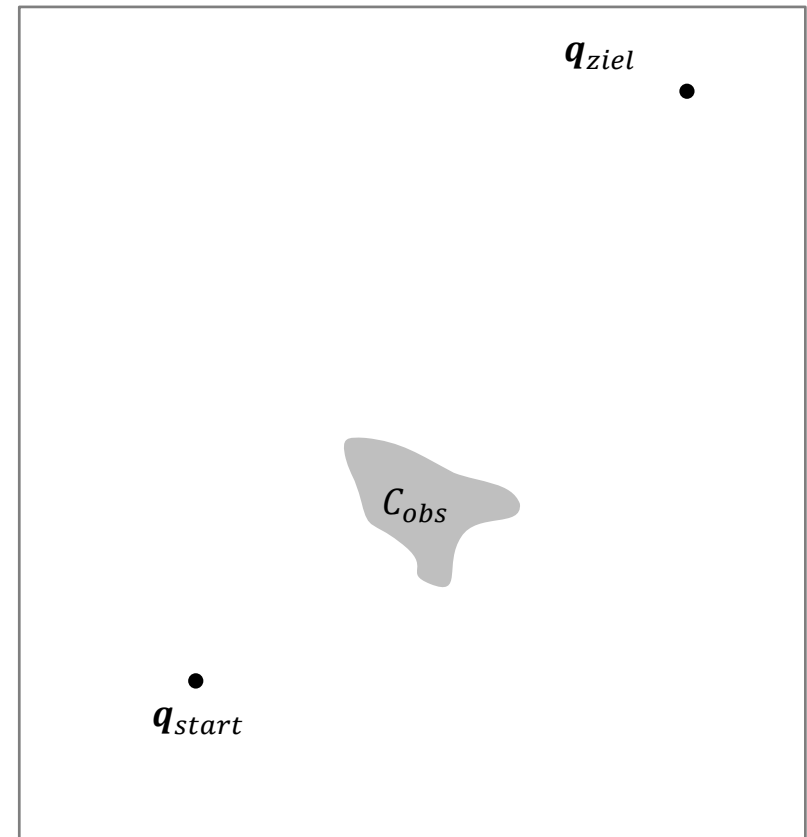
Rapidly-exploring Random Trees (RRTs)

- Im Gegensatz zu PRMs
 - Algorithmus zur Einmalanfrage
 - Keine Vorverarbeitung nötig
 - Keine Probleme mit sich verändernden Umgebungen / kinematischen Ketten
- Probabilistisch vollständiger, randomisierter Algorithmus
 - Keine Garantie, dass eine Lösung innerhalb eines Zeitlimits gefunden wird
 - Wenn eine Lösung existiert, wird sie gefunden (Laufzeit geht gegen Unendlich)
 - Terminiert nicht, wenn keine Lösung existiert
- Effizient für hochdimensionale Problemstellungen
- Erweiterungen der klassischen RRT für spezifische Problemstellungen
z.B. enge Durchgänge

RRT: Prinzip I

- Die Form von $\mathcal{C}_{obs}$ im Konfigurationsraum ist unbekannt
- Initialisierung des RRT
 - Erzeuge leeren Baum T
 - Füge q_{start} in T ein

Beispiel mit 2D Konfigurationsraum

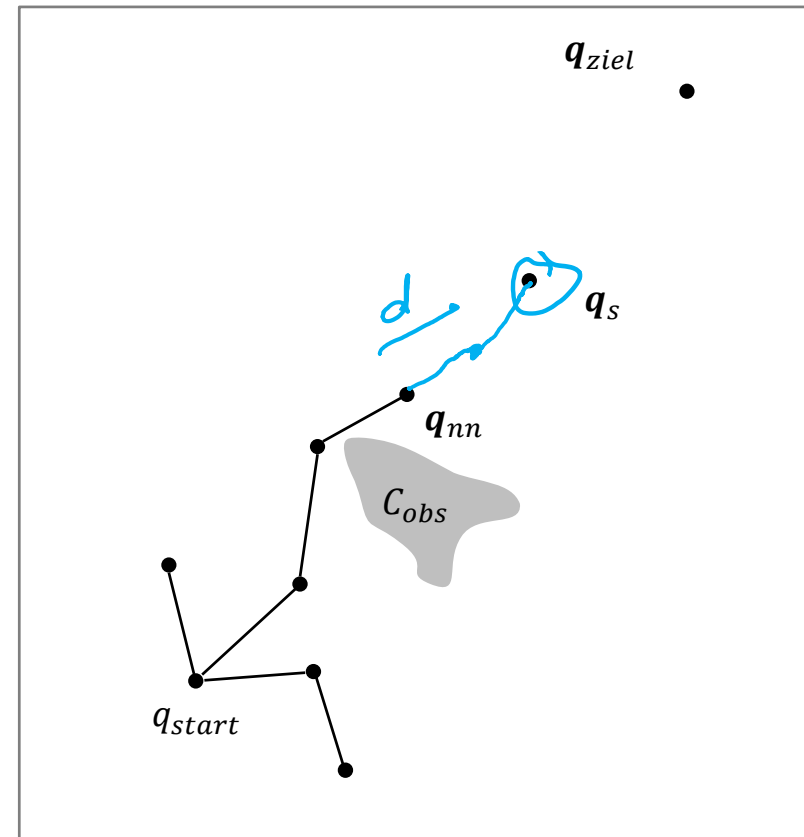


RRT: Prinzip II

■ Iteration

1. Erzeuge einen zufälligen Punkt q_s
2. Bestimme den nächsten Nachbarn q_{nn} in T
3. Füge Punkte auf der Verbindung zwischen q_s und q_{nn} in T ein
 - Mit der Schrittweite d
 - Prüfe jeden der Teilpfade auf Kollision mit C_{obs} .
 - Stoppe, wenn eine Kollision erkannt wurde.
4. Gehe zu 1.

Beispiel mit 2D Konfigurationsraum

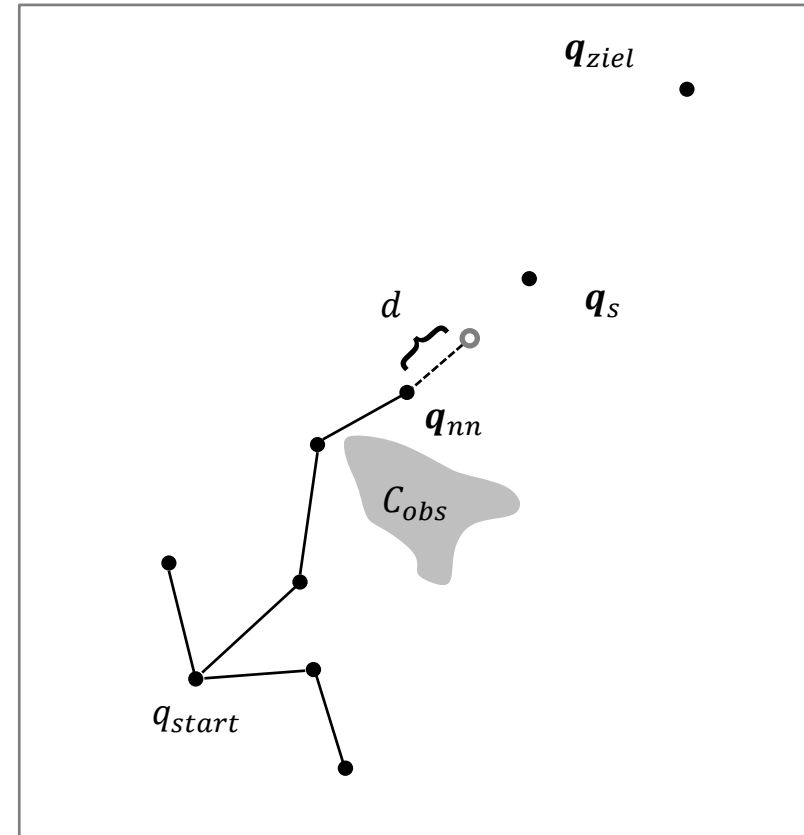


RRT: Prinzip II

■ Iteration

1. Erzeuge einen zufälligen Punkt q_s
2. Bestimme den nächsten Nachbarn q_{nn} in T
3. Füge Punkte auf der Verbindung zwischen q_s und q_{nn} in T ein
 - Mit der Schrittweite d
 - Prüfe jeden der Teilpfade auf Kollision mit C_{obs} .
 - Stoppe, wenn eine Kollision erkannt wurde.
4. Gehe zu 1.

Beispiel mit 2D Konfigurationsraum

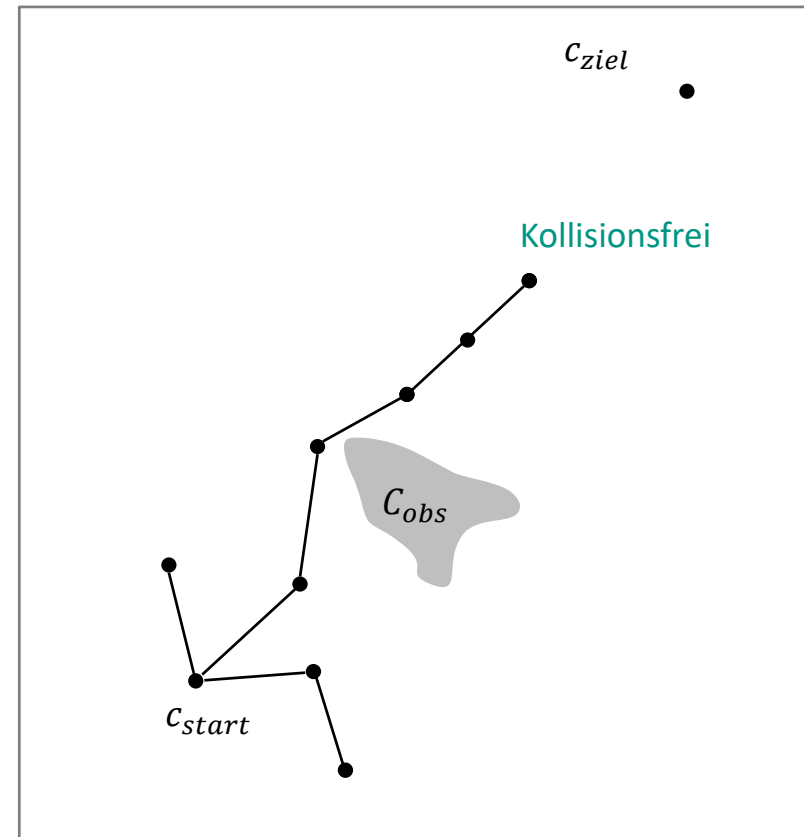


RRT: Prinzip III

■ Iteration

1. Erzeuge einen zufälligen Punkt q_s
2. Bestimme den nächsten Nachbarn q_{nn} in T
3. Füge Punkte auf der Verbindung zwischen q_s und q_{nn} in T ein
 - Mit der Schrittweite d
 - Prüfe jeden der Teilpfade auf Kollision mit C_{obs} . Stoppe, wenn eine Kollision erkannt wurde.
4. Gehe zu 1.

Beispiel mit 2D Konfigurationsraum

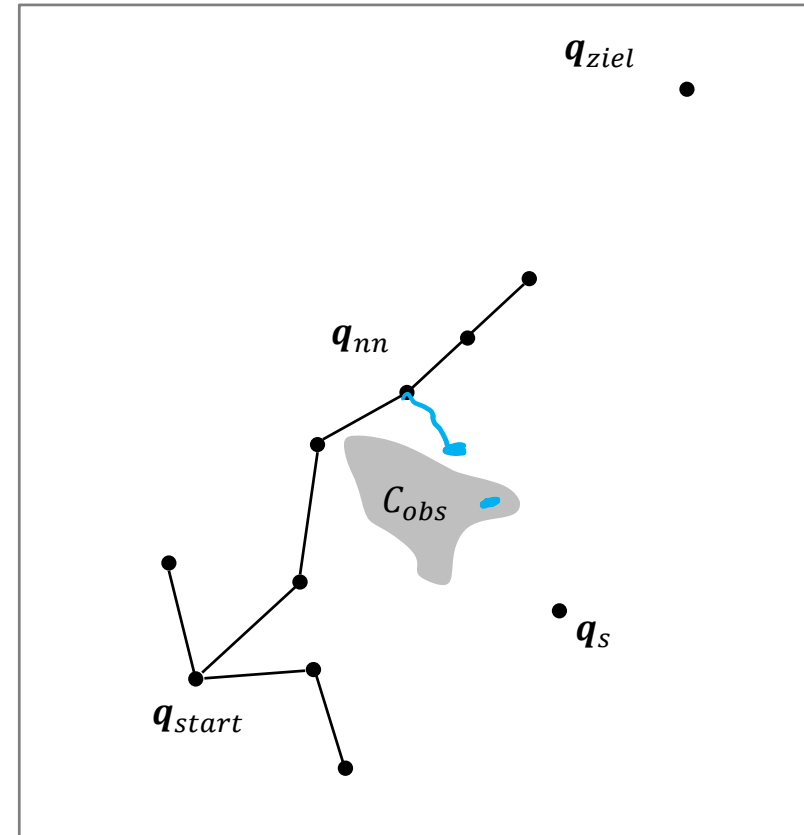


RRT: Prinzip IV

■ Iteration

1. Erzeuge einen zufälligen Punkt q_s
2. Bestimme den nächsten Nachbarn q_{nn} in T
3. Füge Punkte auf der Verbindung zwischen q_s und q_{nn} in T ein
 - Mit der Schrittweite d
 - Prüfe jeden der Teilpfade auf Kollision mit C_{obs} . Stoppe, wenn eine Kollision erkannt wurde.
4. Gehe zu 1.

Beispiel mit 2D Konfigurationsraum

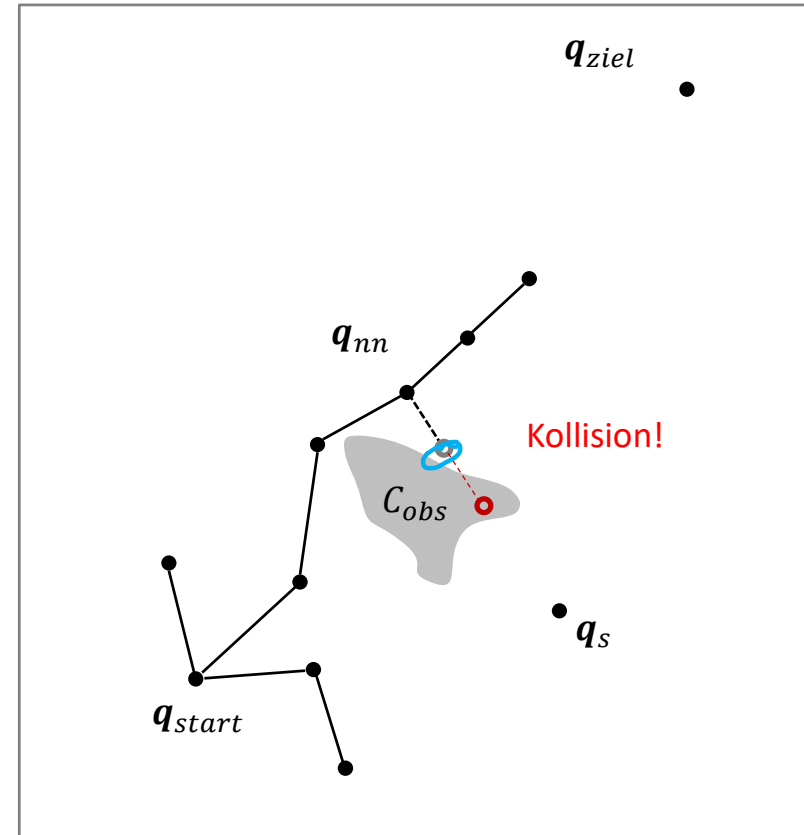


RRT: Prinzip IV

■ Iteration

1. Erzeuge einen zufälligen Punkt q_s
2. Bestimme den nächsten Nachbarn q_{nn} in T
3. Füge Punkte auf der Verbindung zwischen q_s und q_{nn} in T ein
 - Mit der Schrittweite d
 - Prüfe jeden der Teilpfade auf Kollision mit C_{obs} . Stoppe, wenn eine Kollision erkannt wurde.
4. Gehe zu 1.

Beispiel mit 2D Konfigurationsraum

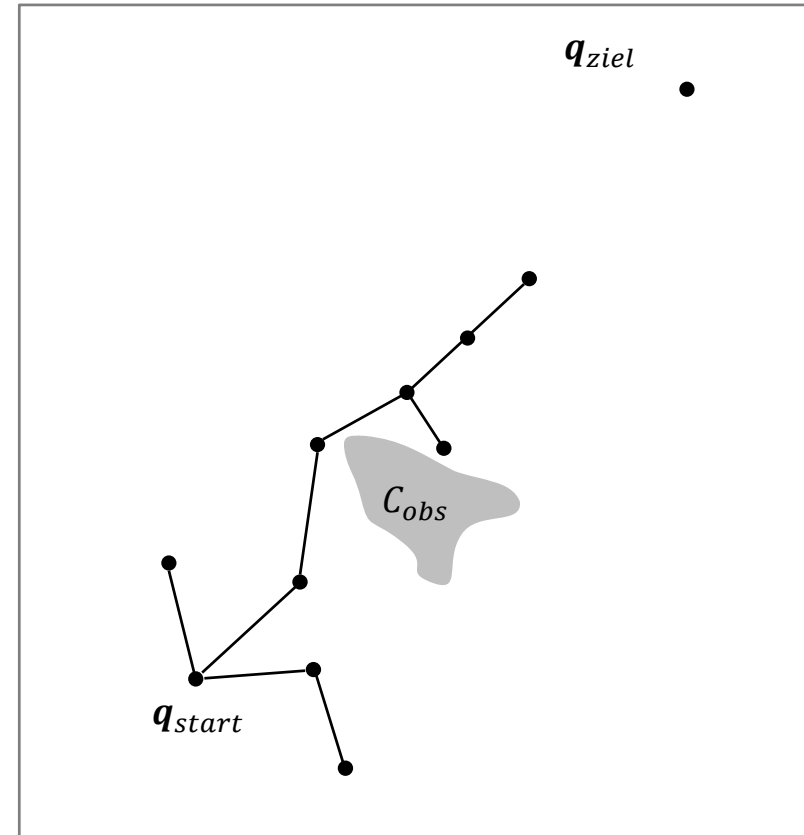


RRT: Prinzip V

■ Iteration

1. Erzeuge einen zufälligen Punkt q_s
2. Bestimme den nächsten Nachbarn q_{nn} in T
3. Füge Punkte auf der Verbindung zwischen q_s und q_{nn} in T ein
 - Mit der Schrittweite d
 - Prüfe jeden der Teilpfade auf Kollision mit C_{obs} . Stoppe, wenn eine Kollision erkannt wurde.
4. Gehe zu 1.

Beispiel mit 2D Konfigurationsraum



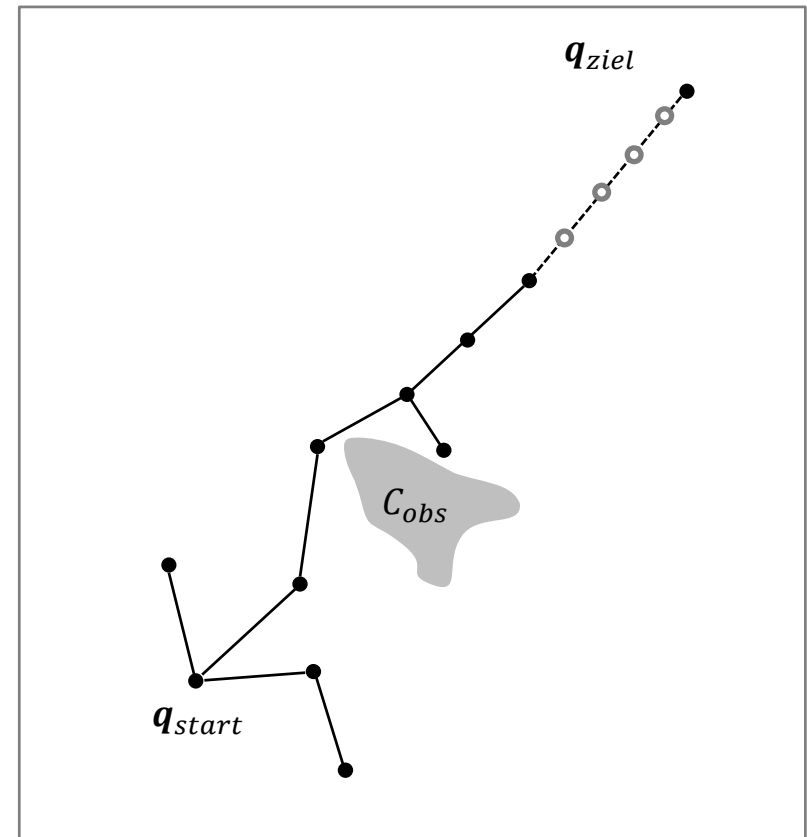
RRT: Prinzip VI

■ Iteration

1. Erzeuge einen zufälligen Punkt q_s
2. Bestimme den nächsten Nachbarn q_{nn} in T
3. Füge Punkte auf der Verbindung zwischen q_s und q_{nn} in T ein
 - Mit der Schrittweite d
 - Prüfe jeden der Teilpfade auf Kollision mit C_{obs} . Stoppe, wenn eine Kollision erkannt wurde.
4. Gehe zu 1.

- ## ■ Prüfe in jedem k -ten Schritt, ob q_{ziel} mit T verbunden werden kann

Beispiel mit 2D Konfigurationsraum



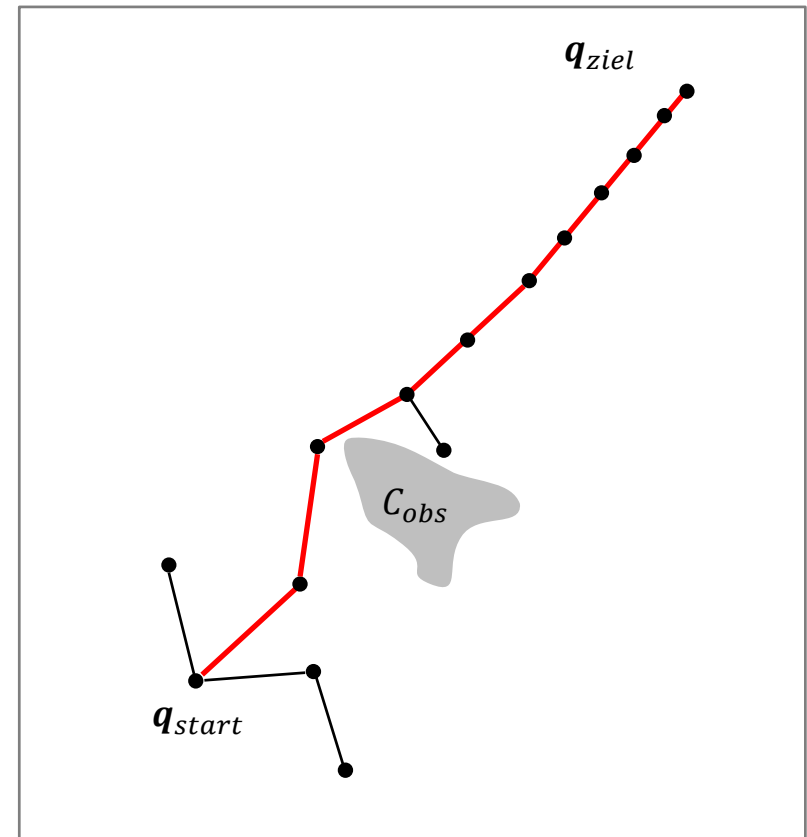
RRT: Prinzip VII

■ Iteration

1. Erzeuge einen zufälligen Punkt q_s
2. Bestimme den nächsten Nachbarn q_{nn} in T
3. Füge Punkte auf der Verbindung zwischen q_s und q_{nn} in T ein
 - Mit der Schrittweite d
 - Prüfe jeden der Teilpfade auf Kollision mit C_{obs} . Stoppe, wenn eine Kollision erkannt wurde.
4. Gehe zu 1.

- ## ■ Prüfe in jedem k -ten Schritt, ob q_{ziel} mit T verbunden werden kann

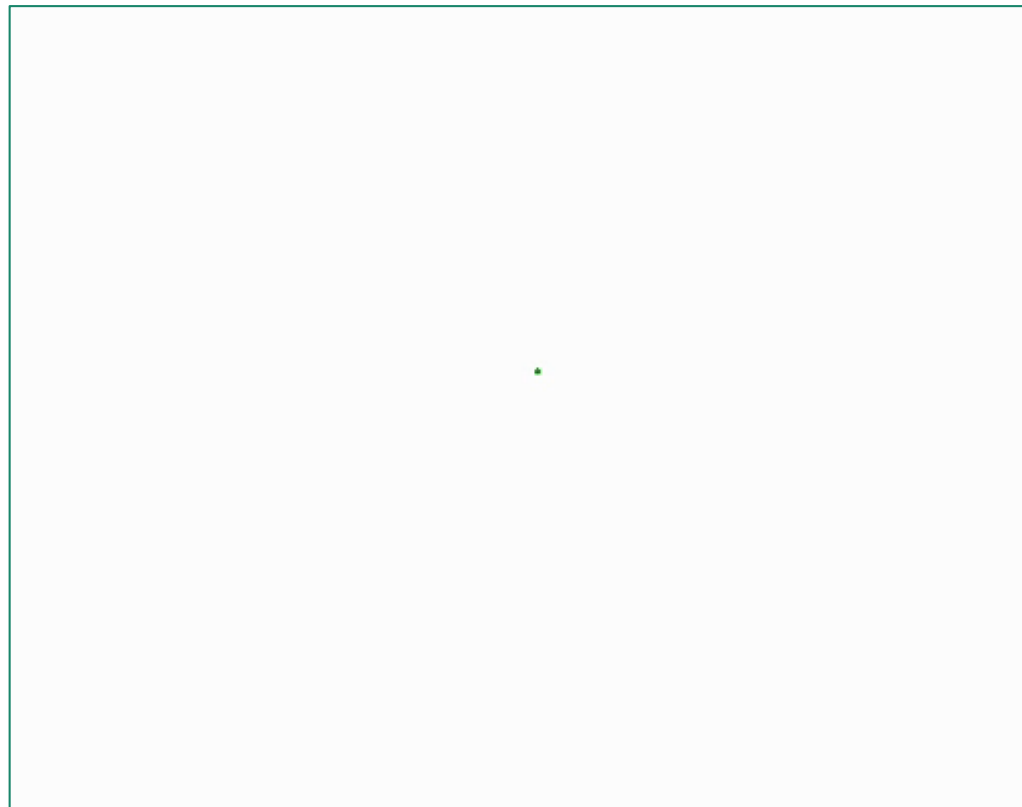
Beispiel mit 2D Konfigurationsraum



Lösung gefunden

RRT: Probabilistische Vollständigkeit I

- RRTs sind probabilistisch vollständig
- **Intuition:** T breitet sich gleichmäßig im Konfigurationsraum aus

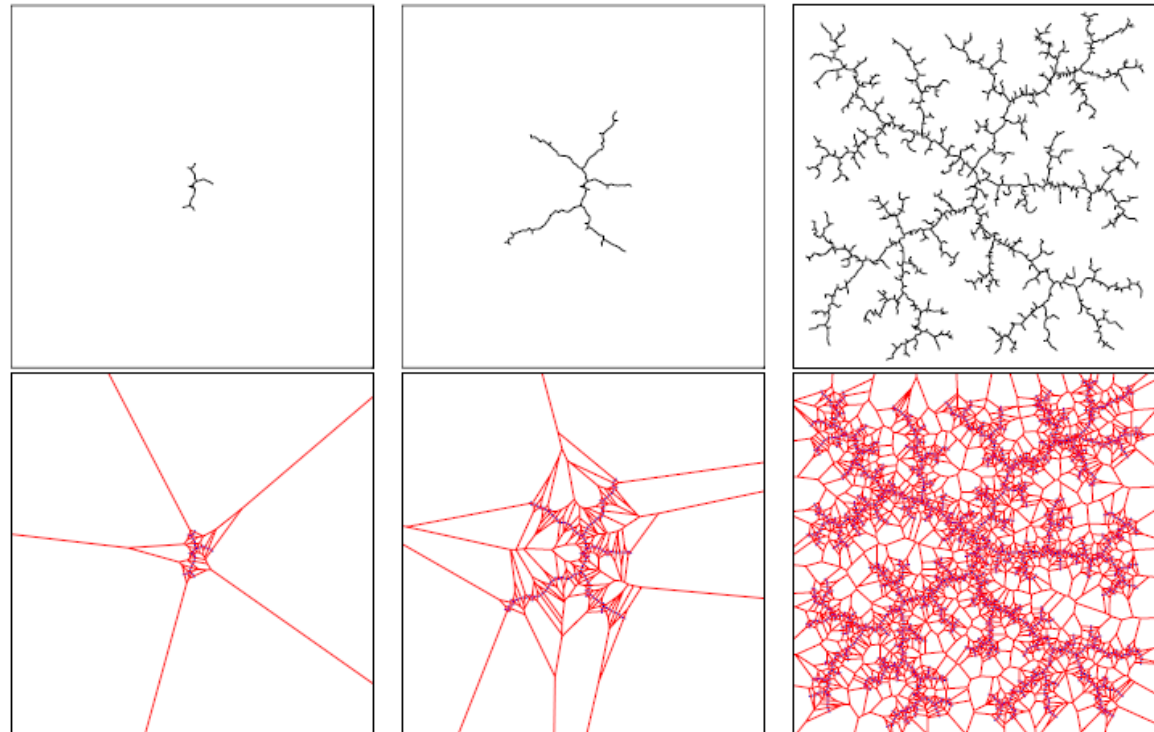


https://en.wikipedia.org/wiki/Rapidly-exploring_random_tree

RRT: Probabilistische Vollständigkeit II

- Die Wahrscheinlichkeit, dass ein Knoten von T erweitert wird, ist proportional zur Größe seiner Voronoi-Region

RRT

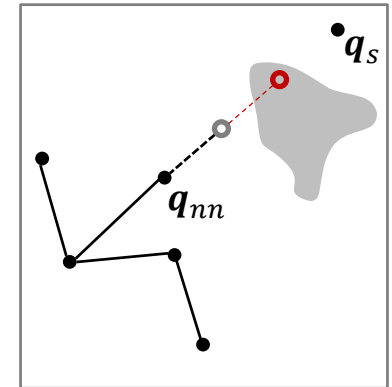


Voronoi-Regionen

J. J. Kuffner and S. M. LaValle, *RRT-connect: An efficient approach to single-query path planning*, *Proceedings IEEE International Conference on Robotics and Automation*. 2000, pp. 995-1001

RRT: Kollisionsprüfung I

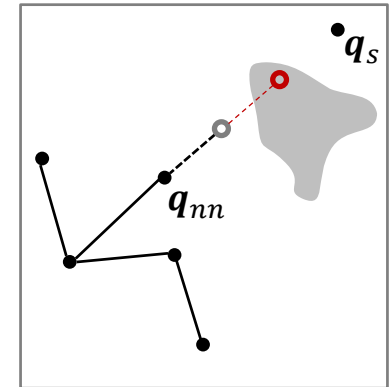
- C_{obs} ist nicht bekannt, wie kann geprüft werden, ob q_s kollisionsfrei erreicht werden kann?



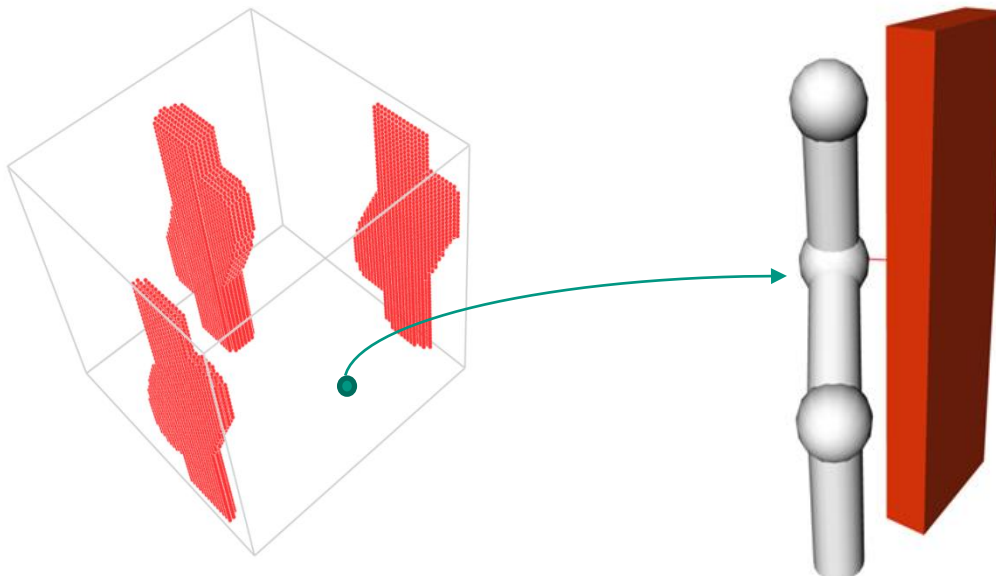
Kollision!

RRT: Kollisionsprüfung II

- C_{obs} ist nicht bekannt, wie kann geprüft werden, ob q_s kollisionsfrei erreicht werden kann?
- Jeder Punkt $q \in C$ beschreibt eine Konfiguration des Roboters
 - Kollisionsprüfung im Arbeitsraum durchführen



Kollision!



RRT: Kollisionsprüfung III

- Es müssen auch Pfadsegmente auf Kollision geprüft werden

- Continuous collision detection (CCD)

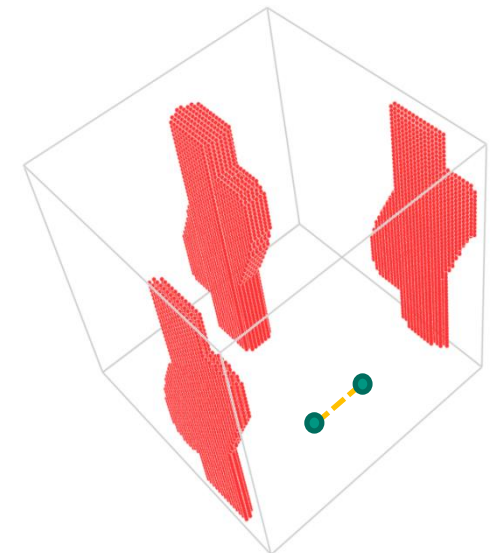
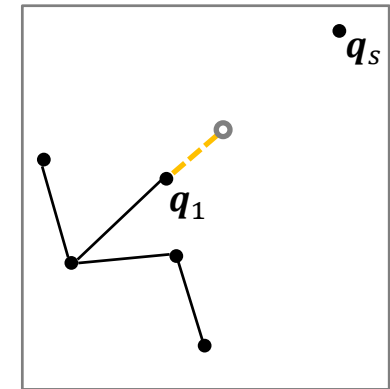
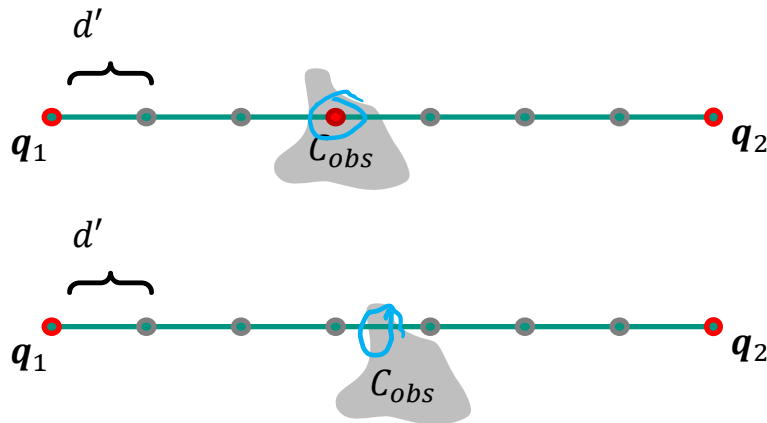
Exakt, aber langsam

- Sampling-basiertes Verfahren

- Einzelne Punkte auf dem Pfad werden geprüft

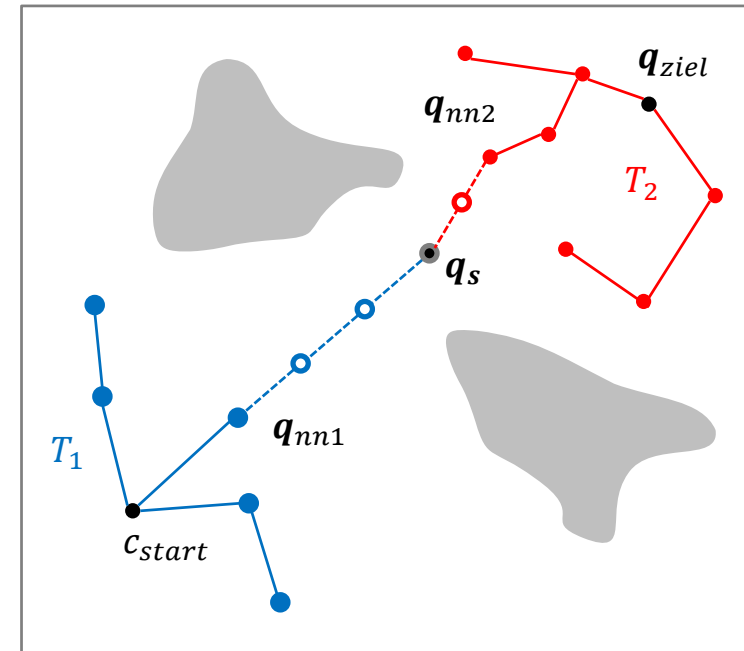
- Schnell, aber nicht exakt

- Beinhaltet einen Parameter (Sampling-Distanz d')



Bidirektionale RRTs

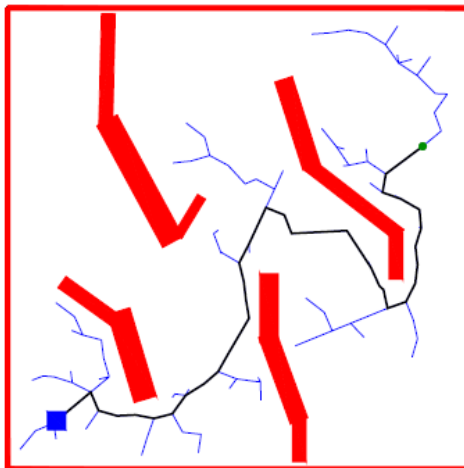
- Es werden zwei Bäume aufgebaut
 - T_1 ausgehend von q_{start}
 - T_2 ausgehend von q_{ziel}
- Zufällig gewählte Punkte q_s erweitern beide Bäume über:
 - $q_{nn,1}$ (Nächster Nachbar in T_1)
 - $q_{nn,2}$ (Nächster Nachbar in T_2)
- Eine Lösung ist gefunden, wenn beide Bäume mit q_s verbunden wurden
- Original-Version (ähnlich): **RRT-Connect**



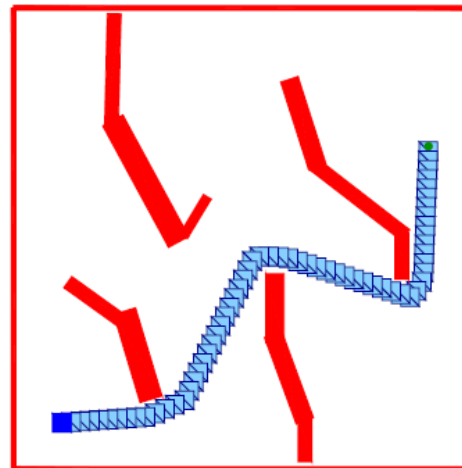
J. J. Kuffner and S. M. LaValle, *RRT-connect: An efficient approach to single-query path planning*,
Proceedings IEEE International Conference on Robotics and Automation. 2000, pp. 995-1001

RRT: Nachbearbeitung

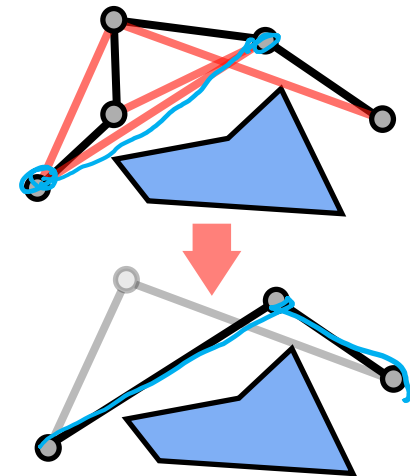
- Lösungen können durch Nachbearbeitung verbessert werden
 - Zufällige Wahl zweier Knoten im Lösungsweg
 - Falls die Verbindung kollisionsfrei ist, verbinde beide Knoten und lösche den dazwischenliegenden Knoten aus dem Lösungspfad
 - Erzeugt glattere Trajektorien



Ursprünglicher Lösungspfad

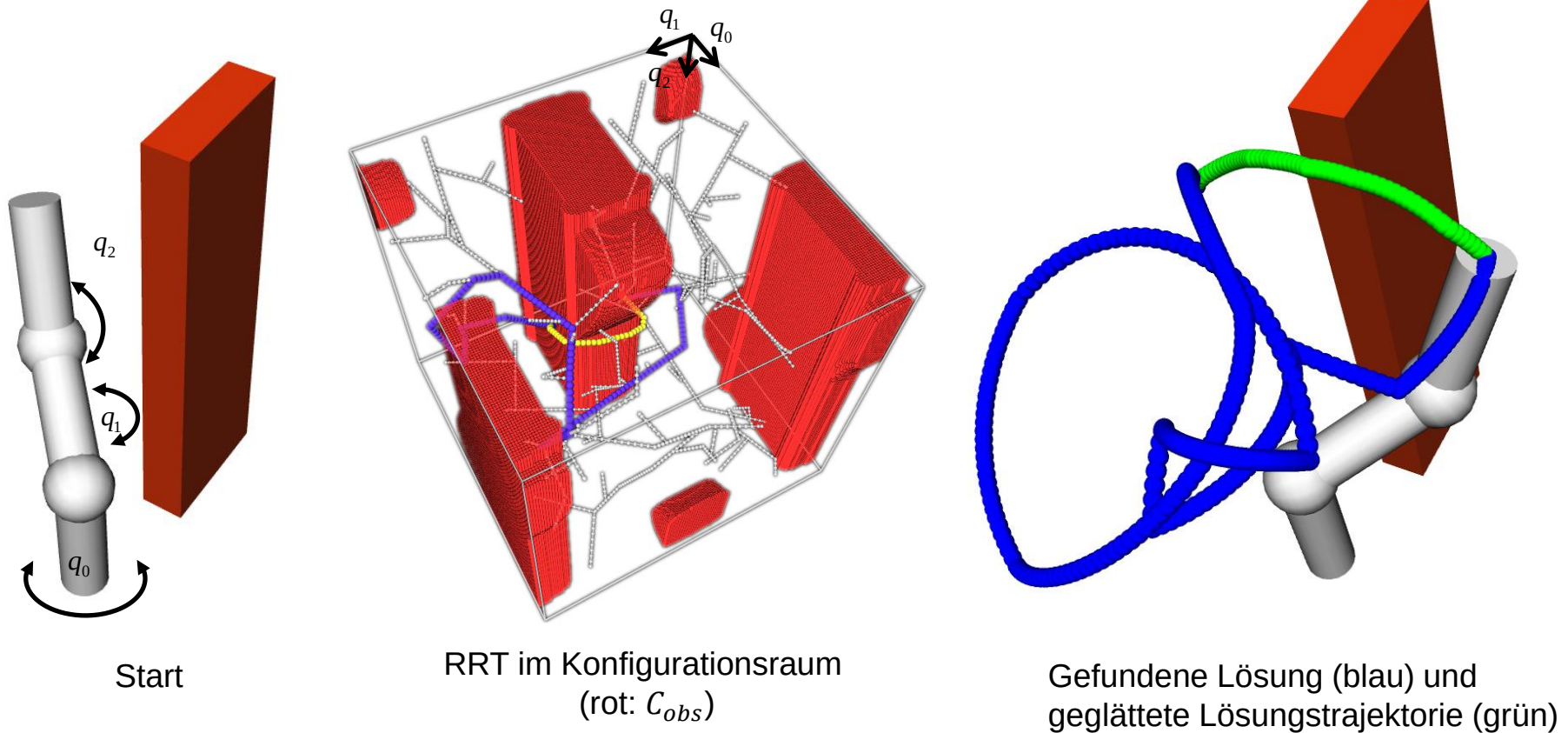


Geglätteter Lösungspfad



RRT: Beispiel

■ Beispiel mit 3D Konfigurationsraum



Inhalt

- Motivation
- Grundlagen der Bewegungsplanung
- Pfadplanung für mobile Roboter
- Bewegungsplanung für Manipulatoren
 - Probabilistic Roadmaps (PRM)
 - Dynamic Roadmaps (DRM)
 - Rapidly-exploring Random Trees (RRT)
 - Erweiterungen von RRT
 - Constrained RRT
 - RRT*
 - Enge Passagen
 - Dynamic Domain RRT
 - Bridge Sampling

Constrained RRT

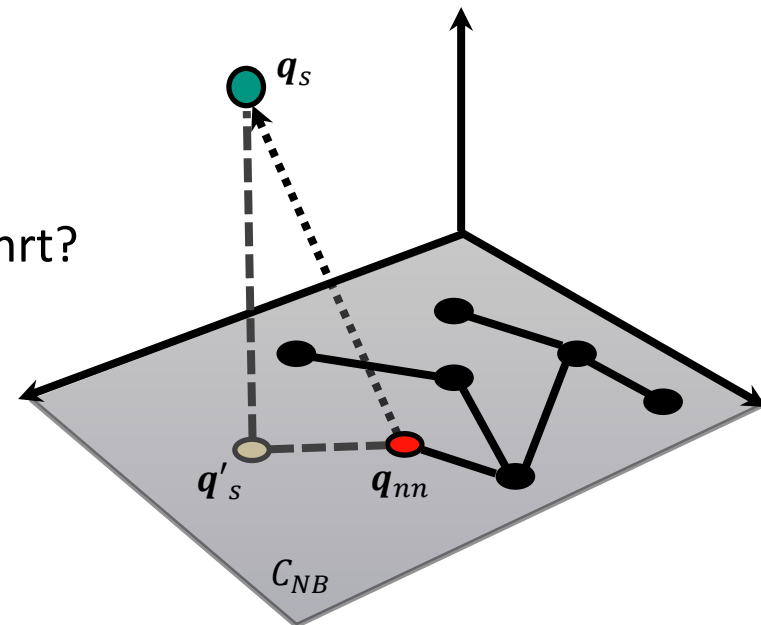
- Bei der Bewegungsplanung müssen evtl. **Nebenbedingungen (Constraints)** erfüllt werden, z.B.:
 - Gleichbleibende Orientierung des Endeffektors
 - (Statische) Stabilität eines zweibeinigen Roboters
- **Problem:** Nebenbedingungen können niederdimensionale Gebilde im Konfigurationsraum darstellen
 - z.B. Die Menge aller Konfigurationen q , die eine Nebenbedingung erfüllen bilden eine **Ebene im dreidimensionalen Konfigurationsraum**
 - Sampling-basierte Ansätze können diese Nebenbedingungen prinzipiell nicht erfüllen
- **Lösungsansätze:**
 - Randomized Gradient Descent (RGD)
 - First Order Retraction (FR)

Constrained RRT

- **Idee:** Projiziere eine Stichprobe q_s auf eine Konfiguration q'_s , die die Nebenbedingung erfüllt
- **Beispiel:** Eine Nebenbedingung NB bilde eine 2D Mannigfaltigkeit in $C_{NB} \subseteq C$

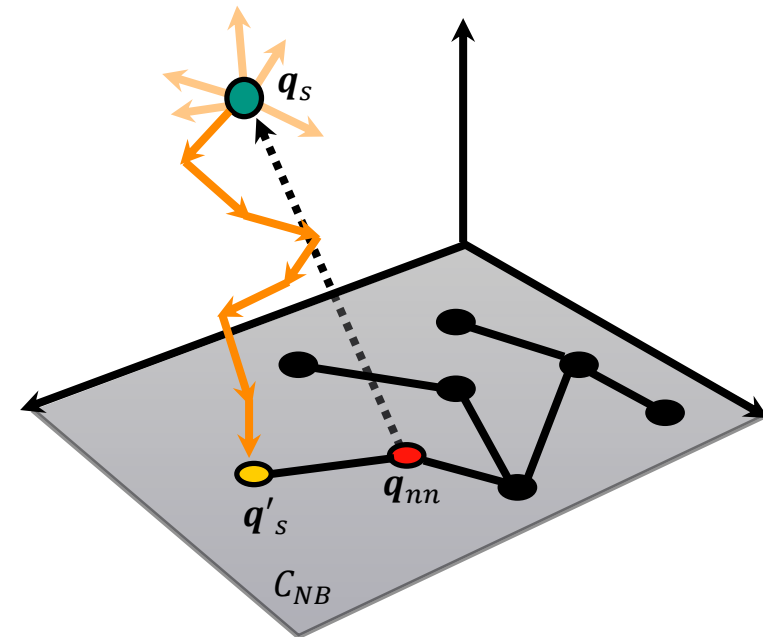
$$C_{NB} = \{q \in C : q \text{ erfüllt NB}\}$$

- **Problem:** Wie wird die Projektion durchgeführt?
 - Randomized Gradient Descent
 - First Order Retraction



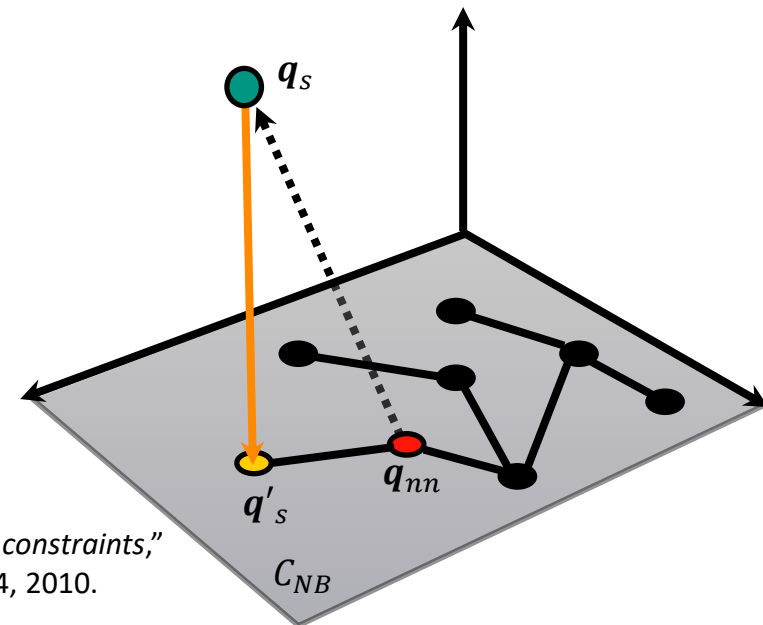
Constrained RRT: Randomized Gradient Descent

- Toleranzwert für Nebenbedingung: α
- Zufällige Bestimmung von n Nachbarn von q_s (in Hyperkugel mit Radius d_{max})
- Falls die Distanz eines Nachbarn zu C_{NB} kleiner als die Distanz von q_s zu C_{NB} , ersetze q_s mit diesem Nachbarn
- Wiederholen bis maximale Iterationszahl erreicht oder die Distanz von q_s zu C_{NB} kleiner ist als α
- Distanzmaß zu C_{NB} im Arbeitsraum (!) notwendig
- Keine Richtungsinformation notwendig



Constrained RRT: First Order Retraction

- Toleranzwert für die Nebenbedingung: α
- Jacobi-Matrix J liefert Richtungsinformation
- Berechnung wie bei Bestimmung der inversen Kinematik
 - $q'_s = q_s - J(q_s)^\# \Delta x_s$
 - Δx_s ist der Abstand von q_s zu C_{NB} im Arbeitsraum
- Distanzmaß zu C_{NB} im Arbeitsraum notwendig



M. Stilman, "Global manipulation planning in robot joint space with task constraints," IEEE Transactions on Robotics and Automation, vol. 26, no. 3, pp. 576–584, 2010.

Inhalt

- Motivation
- Grundlagen der Bewegungsplanung
- Pfadplanung für mobile Roboter
- Bewegungsplanung für Manipulatoren
 - Probabilistic Roadmaps (PRM)
 - Dynamic Roadmaps (DRM)
 - Rapidly-exploring Random Trees (RRT)
 - Erweiterungen von RRT
 - Constrained RRT
 - RRT*
 - Enge Passagen
 - Dynamic Domain RRT
 - Bridge Sampling

RRT*

- **Problem:** RRTs finden Trajektorien, die üblicherweise nicht optimal sind
- **RRT*** optimiert den Suchbaum iterativ während der Suche
 - Mit ausreichender Zeit wird der optimale Pfad zwischen q_{start} und q_{ziel} gefunden $\Rightarrow$ asymptotische Optimalität
- Optimierung des Suchbaums aufgeteilt in zwei Schritte
 - Ermittle zu jedem neuen Knoten die Kosten (z.B. Wegstrecke vom Startknoten)
 - Rewiring des Suchbaums beim Hinzufügen neuer Knoten
- **Nachteil:**
 - Uni-direktionaler Ansatz
 - Längere Laufzeiten (bis zu Faktor 30 im Vergleich zu uni-direktionalem RRT)

S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning.
The International Journal of Robotics Research, 30(7):846–894, Jan. 2011.

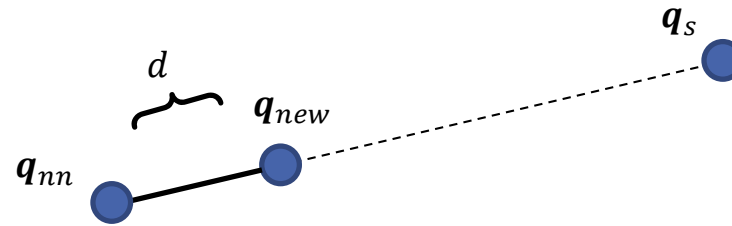
RRT*: Algorithmus

1. $\mathbf{q}_s = \text{SampleRandom}(C)$ // Erzeuge zufällige Konfiguration
2. $\mathbf{q}_{nn} = \text{NearestNeighbor}(\mathbf{q}_s, T)$ // Bestimme nächsten Nachbar
3. $\mathbf{q}_{new} = \text{Steer}(\mathbf{q}_{nn}, \mathbf{q}_s, d)$ // Gehe einen Schritt in Richtung $\mathbf{q}_s$
4. *if* $\neg \text{CollisionFreePath}(\mathbf{q}_{nn}, \mathbf{q}_{new})$: *goto* 1 // Kollisionsfreier Pfad?
5. $Q_{near} = \text{Near}(T, \mathbf{q}_{new}, r)$ // Alle Punkte mit max. Abstand r zu $\mathbf{q}_{new}$
6. $\mathbf{q}_{min} = \text{MinCostPath}(Q_{near}, \mathbf{q}_{new})$ // $\text{Cost}(\mathbf{q}_{min}) + \text{Cost}(\mathbf{q}_{min}, \mathbf{q}_{new})$ minimal
7. $\text{AddPath}(T, \mathbf{q}_{min}, \mathbf{q}_{new})$ // Füge Pfad von $\mathbf{q}_{min}$ zu $\mathbf{q}_{new}$ hinzu
8. $\text{Rewire}(T, \mathbf{q}_{new}, Q_{near})$ // Überprüfe Kanten zu Knoten in Q_{near}
9. *if* $\neg \text{Timeout}$: *goto* 1 // Nächste Iteration

RRT*: Funktionen

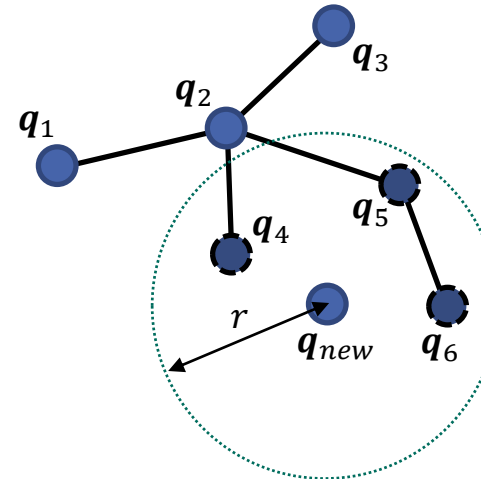
■ $q_{new} = \text{Steer}(q_{nn}, q_s, d)$

- Erzeuge Knoten q_{new}
- Gehe von q_{nn} in Richtung q_s
- Schrittweite d



■ $Q_{near} = \text{Near}(T, q_{new}, r)$

- Bestimme alle Knoten aus T deren Abstand zu q_{new} maximal r beträgt



$$Q_{near} = \{q_4, q_5, q_6\}$$

RRT*: Funktionen (II)

■ $Cost(q_i)$

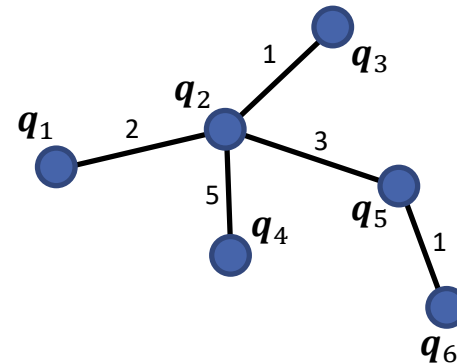
- Pfadkosten von q_{start} zu q_i

■ $Cost(q_a, q_b)$

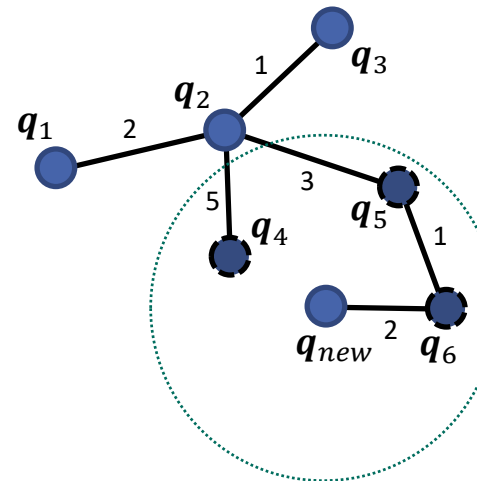
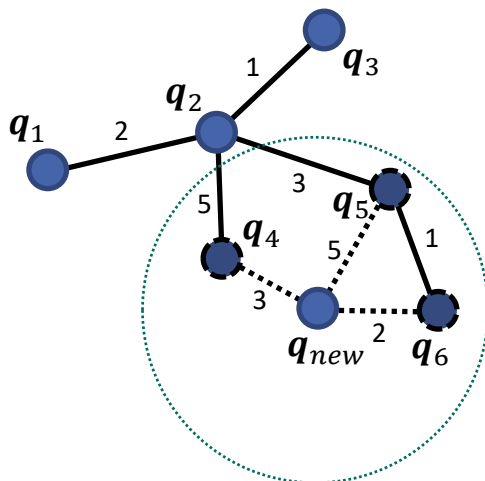
- Kosten der Verbindung von q_a zu q_b

■ $q_{min} = MinCostPath(Q_{near}, q_{new})$

- Bestimme $q_{min} \in Q_{near}$ so dass Pfadkosten $Cost(q_{min}) + Cost(q_{min}, q_{new})$ minimal sind (sowie kollisionsfrei)



$$Cost(q_6) = 2 + 3 + 1 = 6$$



$$q_{min} = q_6$$

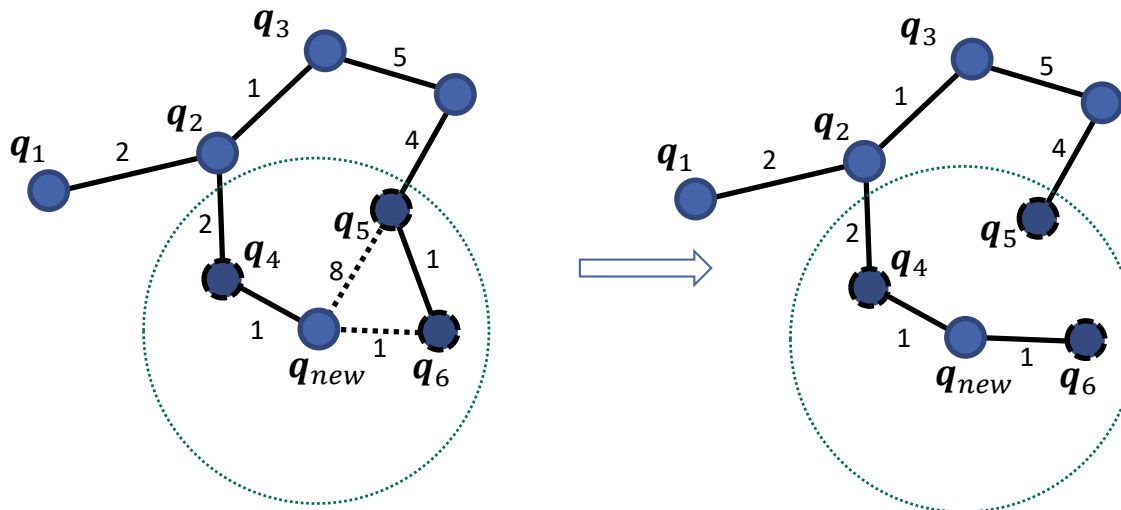
RRT*: Rewiring

■ $Rewire(T, q_{new}, Q_{near})$

- Überprüfe für alle $q_{near} \in Q_{near}$ ob

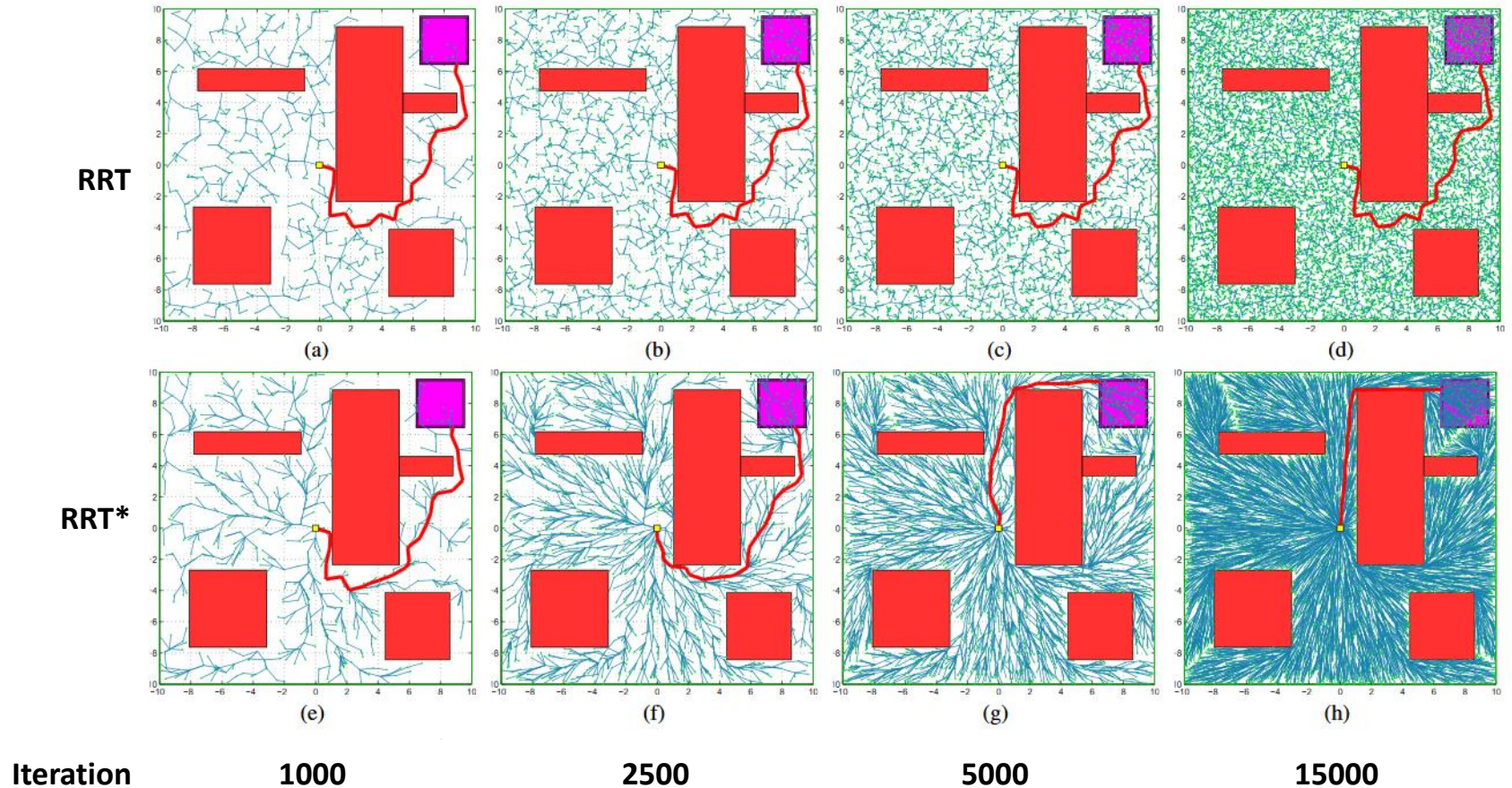
$$Cost(q_{new}) + Cost(q_{new}, q_{near}) < Cost(q_{near})$$

- Ersetze ggf. Verbindung zu q_{near} (falls günstiger und kollisionsfrei)



$$\begin{aligned} Cost(q_5) &= 12 \\ Cost(q_6) &= 13 \\ Cost(q_{new}) &= 5 \\ Cost(q_{new}) + Cost(q_{new}, q_5) &= 13 \\ Cost(q_{new}) + Cost(q_{new}, q_6) &= 6 \end{aligned}$$

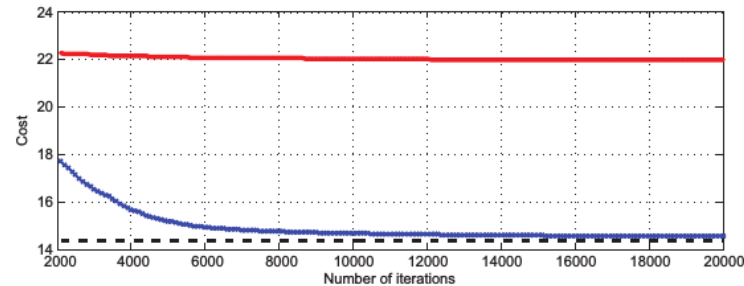
Vergleich RRT und RRT*



S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, Jan. 2011.

Vergleich RRT und RRT* (II)

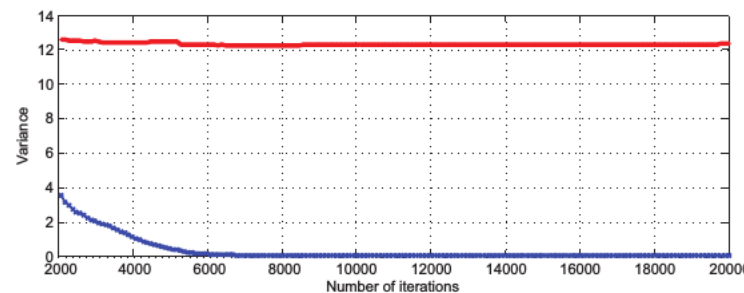
Pfadkosten der gefundenen Lösung
(Optimale Lösung: schwarz)



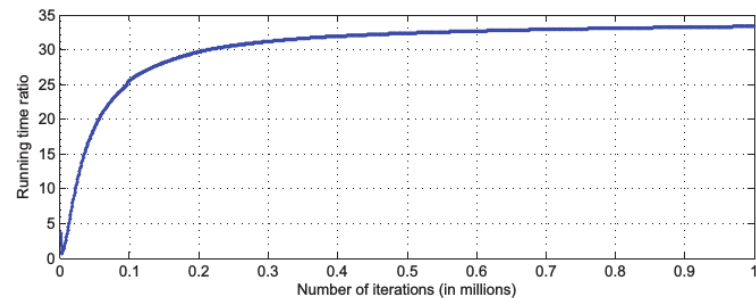
RRT

RRT*

Varianz der Pfadkosten



Verhältnis der Laufzeit RRT* / RRT
(angegeben für eine Iteration)



S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning.
The International Journal of Robotics Research, 30(7):846–894, Jan. 2011.

Inhalt

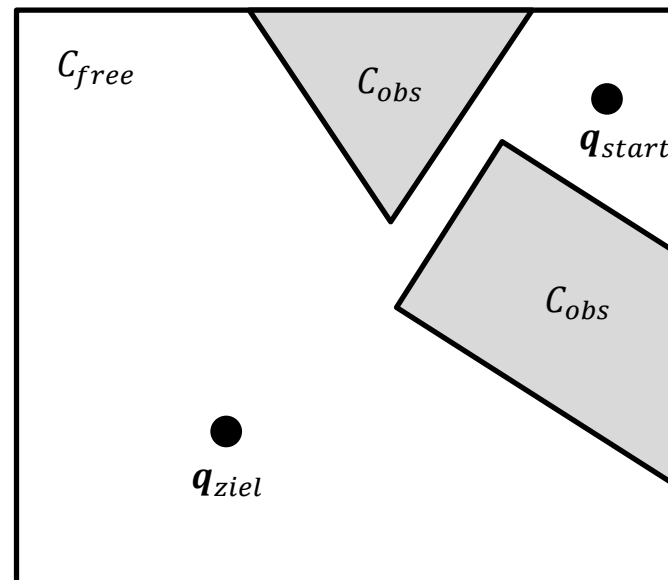
- Motivation
- Grundlagen der Bewegungsplanung
- Pfadplanung für mobile Roboter
- Bewegungsplanung für Manipulatoren
 - Probabilistic Roadmaps (PRM)
 - Dynamic Roadmaps (DRM)
 - Rapidly-exploring Random Trees (RRT)
 - Erweiterungen von RRT
 - Constrained RRT
 - RRT*
 - Enge Passagen
 - Dynamic Domain RRT
 - Bridge Sampling

Enge Passagen: Motivation

- Klassische RRTs bestimmen neue Punkte q_s durch **gleichverteilte Zufallswahl** im Konfigurationsraum C
- Ergebnis gleichverteilter Zufallswahl:
 - **Viele eher uninteressante Stichproben**
z.B. „mitten in C_{free} “, weit entfernt von Hindernissen
 - **Wenige interessante Stichproben**
z.B. nahe bei Hindernissen, insbesondere in engen Passagen zwischen zwei Hindernissen
- Klassische RRTs können viel Zeit benötigen, bis eine Lösung für einen Durchgang durch eine enge Passage gefunden wurde
- **Hauptidee der Verfahren:**
Sampling ist deutlich günstiger als Kollisionsprüfung von Pfaden im Baum

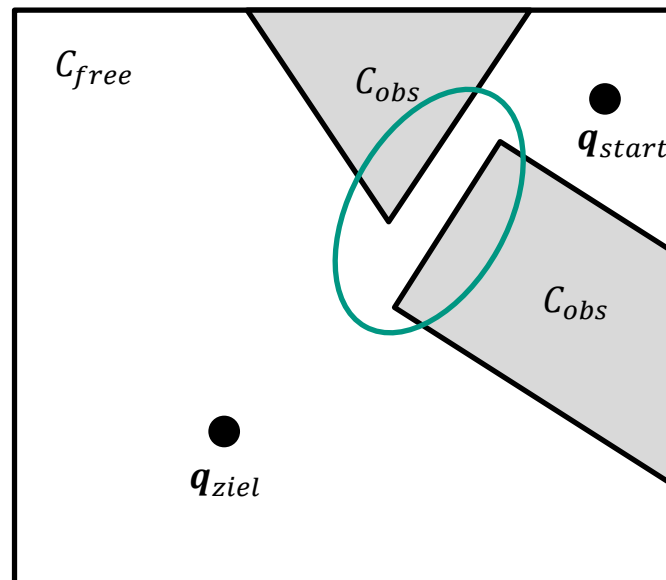
Enge Passagen: Beispiel

- Beispiel für enge Passagen in einem 2D Konfigurationsraum



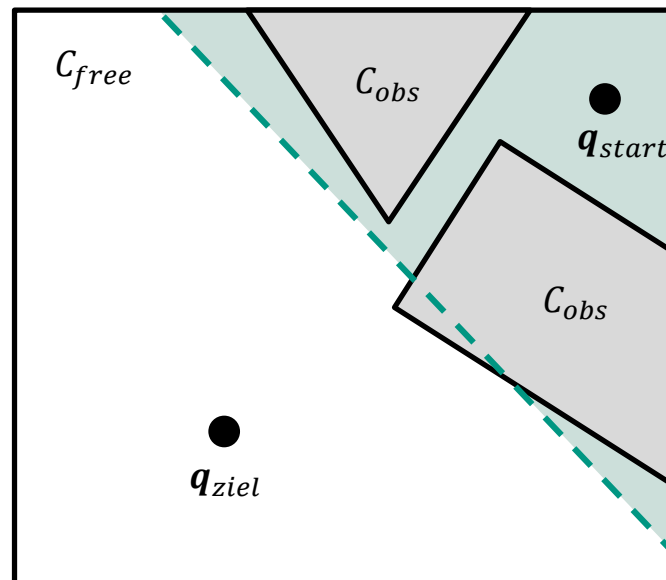
Enge Passagen: Beispiel

- Beispiel für enge Passagen in einem 2D Konfigurationsraum
- Geringe Wahrscheinlichkeit, dass neue Stichproben im Bereich der engen Passage liegen



Enge Passagen: Dynamic Domain RRT I

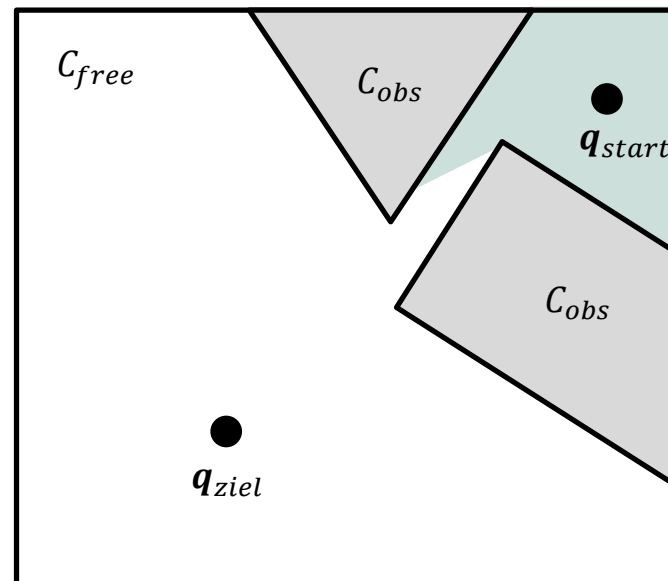
- **Problem:** RRTs erkennen enge Passagen nicht und können nicht zielgerichtet sampeln
- **Ideal:** Nur in sichtbarer Voronoi-Region eines Knotens sampeln



Voronoi-Region zu q_{start}

Enge Passagen: Dynamic Domain RRT (II)

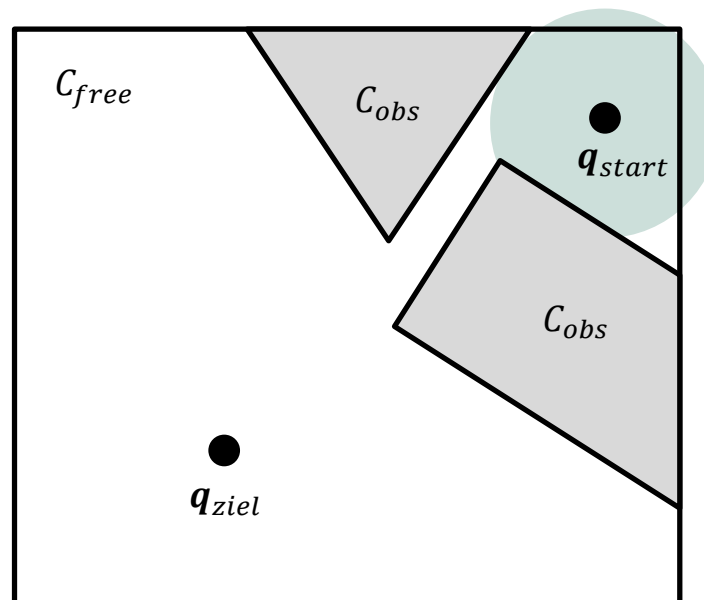
- **Problem:** RRTs erkennen enge Passagen nicht und können nicht zielgerichtet sampeln
- **Ideal:** Nur in sichtbarer Voronoi Region eines Knotens sampeln



Sichtbare Voronoi-Region zu q_{start}

Enge Passagen: Dynamic Domain RRT (III)

- **Aber:** Berechnung sichtbarer Voronoi-Regionen ist aufwendig
(keine explizite Darstellung der Hindernisregionen im Konfigurationsraum)
- **Stattdessen:** Approximation sichtbarer Voronoi-Regionen durch Kugeln mit Radius r (**Dynamic Domain**)



Approximierte sichtbare Voronoi-Region zu q_{start}
(Dynamic Domain)

Dynamic Domain RRT (IV)

- **Dynamic Domain RRT** beschränkt in der Nähe von Hindernissen die Sampling Domäne eines Knotens auf dessen Dynamic Domain (DD).
 - Initial wird der DD-Radius r jedes Knotens auf ∞ gesetzt. Sampling findet in gesamter Voronoi Region des Knotens statt.
 - Wenn während des RRT-Erweiterungsschritts keine Verbindung zu einem Knoten hergestellt werden kann, wird dessen DD-Radius auf einen festgelegten Wert R reduziert.
Knoten dieser Art werden **Grenzknoten** genannt da sie an der Grenze von C_{free} und C_{obs} liegen.
- **Sampling**
 - Ein Sample q_s wird verworfen, falls q_s außerhalb des DD-Radius seines nächsten Nachbarn liegt ($dist(q_s, q_{nn}) > q_{nn} \cdot r$)

A. Yershova, L. Jaillet, T. Simeon, and S. M. LaValle. Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3856–3861, Apr. 2005.

Dynamic Domain RRT (V)

- Bei engen Passagen werden häufige Kollisionsprüfungen vermieden und keine Expansionsversuche zu weit entfernten und unerreichbaren Knoten unternommen.

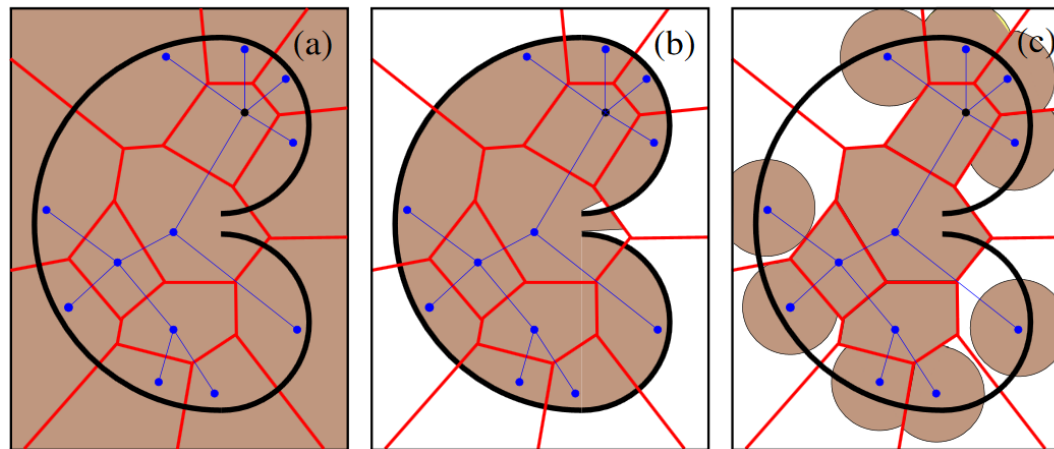
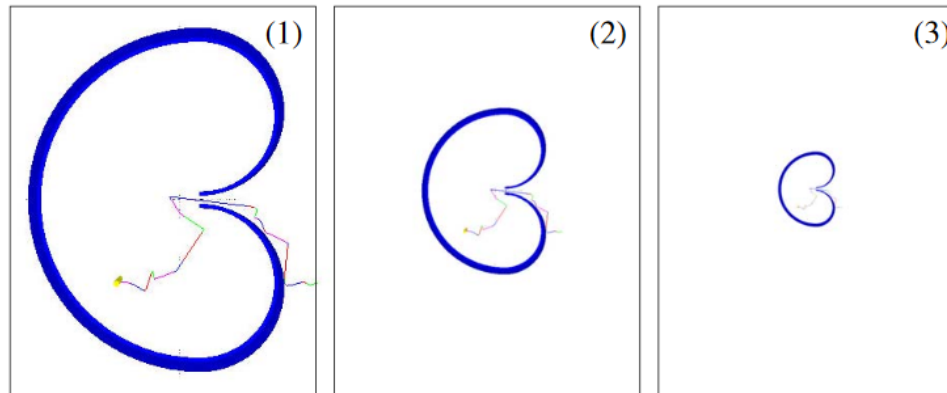


Fig. 3. For a set of points inside a bug trap different sampling domains are shown: (a) regular RRTs sampling domain, (b) visibility Voronoi region, (c) dynamic domain.

A. Yershova, L. Jaillet, T. Simeon, and S. M. LaValle. Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3856–3861, Apr. 2005.

Dynamic Domain RRT – Vergleich zu RRT

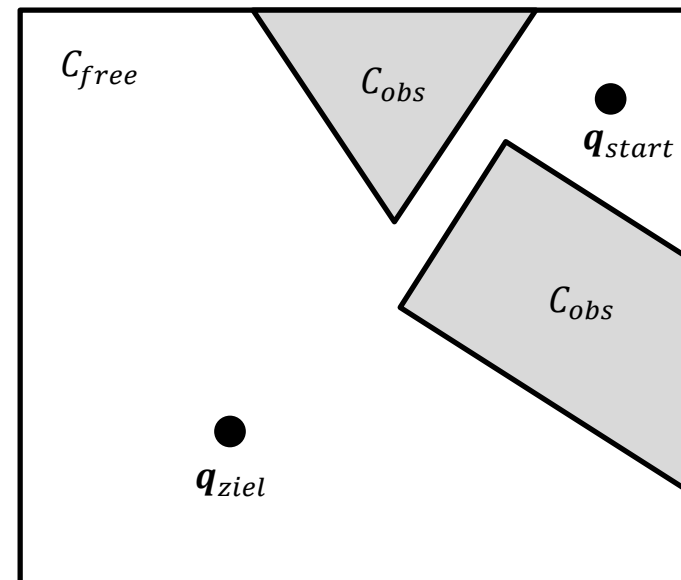


	Dynamic-Domain bi-RRT	bi-RRT
time (1)	0.4 sec	0.1 sec
no. nodes (1)	253	37
CD calls (1)	618	54
time (2)	2.5 sec	379 sec
no. nodes (2)	1607	6924
CD calls (2)	3751	781530
time (3)	1.6 sec	> 80000 sec
no. nodes (3)	1301	–
CD calls (3)	3022	–

A. Yershova, L. Jaillet, T. Simeon, and S. M. LaValle. Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3856–3861, Apr. 2005.

Enge Passagen: Bridge Sampling I

- **Idee:** Wähle zielgerichtet Punkte in engen Passagen für die nächste Stichprobe

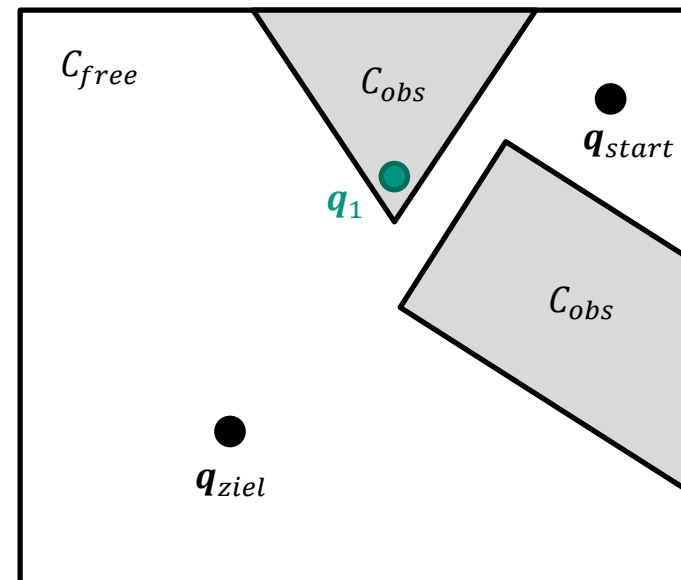


Enge Passagen: Bridge Sampling I

■ **Idee:** Wähle zielgerichtet Punkte in engen Passagen für die nächste Stichprobe

■ **Vorgehen:**

1. Wähle gleichverteilt einen zufälligen Punkt $q_1 \in C_{obs}$

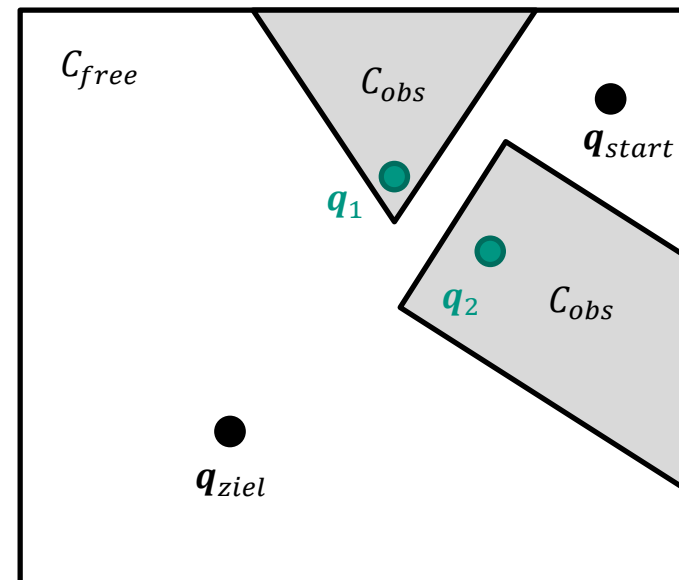


Enge Passagen: Bridge Sampling I

■ **Idee:** Wähle zielgerichtet Punkte in engen Passagen für die nächste Stichprobe

■ **Vorgehen:**

1. Wähle gleichverteilt einen zufälligen Punkt $\mathbf{q}_1 \in C_{obs}$
2. Wähle nach einer geeigneten Wahrscheinlichkeitsverteilung einen zweiten Punkt $\mathbf{q}_2 \in C_{obs}$ in der Nähe von $\mathbf{q}_1$

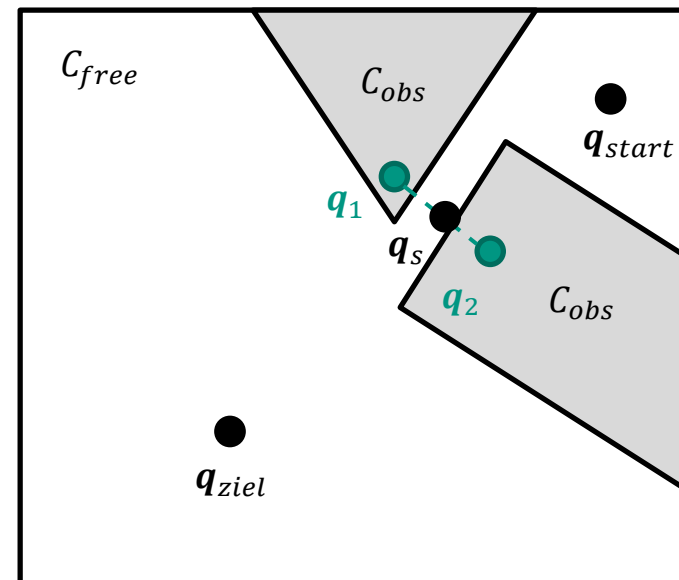


Enge Passagen: Bridge Sampling I

■ **Idee:** Wähle zielgerichtet Punkte in engen Passagen für die nächste Stichprobe

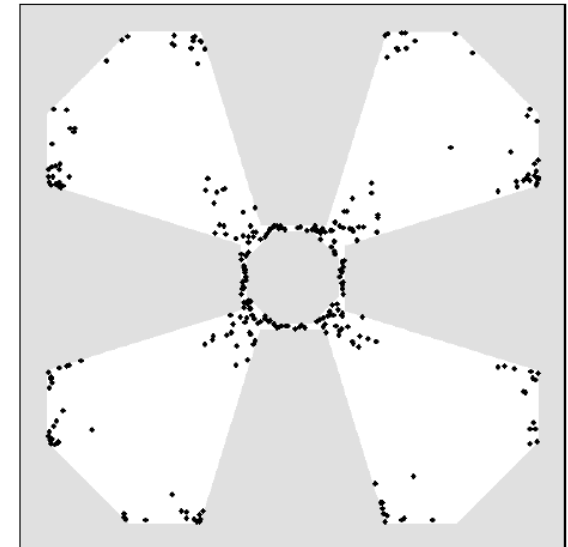
■ **Vorgehen:**

1. Wähle gleichverteilt einen zufälligen Punkt $q_1 \in C_{obs}$
2. Wähle nach einer geeigneten Wahrscheinlichkeitsverteilung einen zweiten Punkt $q_2 \in C_{obs}$ in der Nähe von q_1
3. Wenn der Mittelpunkt q_s zwischen q_1 und q_2 in C_{free} liegt, dann verwende ihn als neue Stichprobe für den RRT (oder die PRM)
4. Wiederhole



Enge Passagen: Bridge Sampling II

- **Bridge Sampling** erhöht die Stichprobendichte in interessanten Bereichen des Konfigurationsraumes C
 - Interessant sind Bereiche in der Nähe von Hindernissen, besonders in engen Passagen
- Bridge Sampling kann für RRTs und PRMs verwendet werden
- Das zentrale Element des Verfahrens, der **Bridge Test**, ist auch in hochdimensionalen Räumen effizient berechenbar



Stichproben beim Bridge Sampling
(Sun et al., 2005)

Z. Sun, D. Hsu, T. Jiang, H. Kurniawati, and J. Reif, "Narrow passage sampling for probabilistic roadmap planning," IEEE Transactions on Robotics, vol. 21, no. 6, pp. 1105–1115, 2005

Englische Begriffe

Deutsch	Englisch
Bewegungsplanung	Motion planning
Freiraum	Free space
Hindernis	Obstacle
Konfiguration	Configuration
Konfigurationsraum	Configuration space
Pfadplanung	Path planning
Potentialfeld	Potential field
Sichtgraph	View graph
Trajektorie	Trajectory
Zellzerlegung	Cell decomposition
Zwangsbedingung	Constraint
Neben- und Randbedingung	Constraint

Englische Begriffe

Deutsch	Englisch
Bestensuche	Best-first search
Enge Passagen	Narrow passages
Kern	Kernel
Nebenbedingung	Constraint
Projektion	Projection
Stichprobe	Sampling
Vorgängerknoten	Predecessor
Zulässig	Admissible