Verfahren der Wegeplanung

Alexander Gran

Institut für Mensch-Maschine-Interaktion

WS 2006/2007

Motivation

Forderungen der Anwender:

Roboter sollen autark agieren.

Dabei sollen sie schnell am Ziel ankommen aber nicht gegen Wände laufen.

Motivation

Weitere Anwendungen:

Biologie

Ligand Anlagerungen bei der Medikamentenherstellung, Proteinfaltung.

Serviceroboter

Dauerndes reagieren auf die (unbekannte) Umgebung, Gefahrenvermeidung.

Computer Animation

Einzelprogrammeirung der Freiheitsgerade ist mühsam, Highlevel Schnittstellen sind gewünscht

Roboter $\mathcal{A}$

Ein abstraktes Objekt $\mathcal{A}$ (Körper, Flugzeug, kinematische Kette etc), dass eine bestimmte Konfiguration q haben kann.

Formalismus

Roboter $\mathcal{A}$

Ein abstraktes Objekt $\mathcal{A}$ (Körper, Flugzeug, kinematische Kette etc), dass eine bestimmte Konfiguration q haben kann.

Umgebender Raum $\mathcal{W}$

Im allgemeinen entweder $\mathbb{R}^3$ oder $\mathbb{R}^2$, theoretisch beliebige Räume.

Formalismus

Roboter $\mathcal{A}$

Ein abstraktes Objekt $\mathcal{A}$ (Körper, Flugzeug, kinematische Kette etc), dass eine bestimmte Konfiguration q haben kann.

Umgebender Raum $\mathcal{W}$

Im allgemeinen entweder $\mathbb{R}^3$ oder $\mathbb{R}^2$, theoretisch beliebige Räume.

Konfiguration q

Die Konfiguration q hängt vom Roboter $\mathcal{A}$ ab:

- Kinematische Ketten haben Tupel aus Winkeln
- Ein Flugzeug hat Richtung und Position
- Starre Körper haben allgemein Translationen und Rotationen

Formalismus

Hindernis Raum $\mathcal{O}$

Teilmenge von $\mathcal{W}$, die alle nicht passierbaren Punkte enthält. Nicht notwendigerweise endlich.

Formalismus

Hindernis Raum $\mathcal{O}$

Teilmenge von $\mathcal{W}$, die alle nicht passierbaren Punkte enthält. Nicht notwendigerweise endlich.

Konfigurations Raum $\mathcal{C}$

Alle Konfigurationen q, abhängig vom Roboter $\mathcal{A}$:

- Bei begrenzten kinematischen Ketten reicht $\mathbb{R}^n$, wobei n die Anzahl der Verbindungsstellen ist
- Ein Flugzeug nimmt vielleicht $R^{(3x2)}$
- Rotationen erzeugen oft unbegrenzte Möglichkeiten, z.B. S^1, mit S als Einheitskreis

Blockierter Konfigurations Raum $\mathcal{C}_{obs}$

Nicht möglichen Konfigurationen q, eingeschränkt durch Umgebung und den Roboter selber.
Der freie Konfigurationsraum ist $\mathcal{C}_{free}$. Berechnung ist nur in Trivialfällen möglich, sonst durch probieren.

Formalismus

Blockierter Konfigurations Raum $\mathcal{C}_{obs}$

Nicht möglichen Konfigurationen q, eingeschränkt durch Umgebung und den Roboter selber.
Der freie Konfigurationsraum ist $\mathcal{C}_{free}$. Berechnung ist nur in Trivialfällen möglich, sonst durch probieren.

Pfad

Folge von Konfigurationen q_i aus $\mathcal{C}_{free}$, die Start q_s und Ziel q_z miteinander verbinden.

Blockierter Konfigurations Raum $\mathcal{C}_{obs}$

Nicht möglichen Konfigurationen q, eingeschränkt durch Umgebung und den Roboter selber.
Der freie Konfigurationsraum ist $\mathcal{C}_{free}$. Berechnung ist nur in Trivialfällen möglich, sonst durch probieren.

Pfad

Folge von Konfigurationen q_i aus $\mathcal{C}_{free}$, die Start q_s und Ziel q_z miteinander verbinden.

Satz

Das finden eines Pfades ist NP-Vollständig (Piano Movers Problem).

Formalismus

Problemtypen:

- Holonome Systeme: $||\mathcal{C}|| = ||\mathcal{W}||$ Bsp: Flugzeug (6=6)
- Holonome, redundante Systeme $||\mathcal{C}|| > ||\mathcal{W}||$ Bsp: Menschliche Hand $(7 > 6)$
- Nicht holonome Systeme, $||\mathcal{C}|| < ||\mathcal{W}||$Bsp: Auto (3<6)

Formalismus

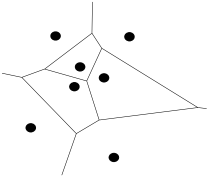
Problemtypen:

- Holonome Systeme: $||\mathcal{C}|| = ||\mathcal{W}||$ Bsp: Flugzeug (6=6)
- Holonome, redundante Systeme $||\mathcal{C}|| > ||\mathcal{W}||$ Bsp: Menschliche Hand $(7 > 6)$
- Nicht holonome Systeme, $||\mathcal{C}|| < ||\mathcal{W}||$Bsp: Auto (3<6)

- Globale Probleme, $\mathcal{C}_{obs}$ ist komplett bekannt
- Lokale Probleme, $\mathcal{C}_{obs}$ wird nach und nach gesehen
- Dyanmische Probleme, $\mathcal{C}_{obs}$ ändert sich mit mehreren Abfragen

Voronoi Diagramm

Aufteilung der Ebene in Flächen
Minimaler Abstand der Flächenpunkte zu Messpunkten

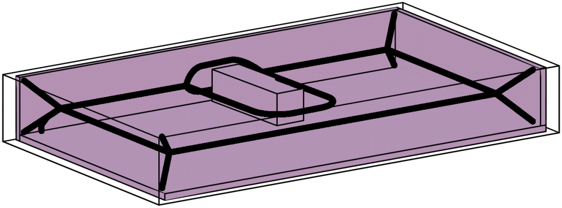


Generalized Voronoi Diagramm

Punktemengen statt Punkten.
Immernoch beschränkt auf $\mathbb{R}^2$

HGVG: Hierarchical generalized Voronoi Graph

Incrementell aufgebaute Karte zum Wegfinden, besteht aus generalisierten Voronoi Diagrammen. Immer verbunden, wenn Weg möglich.



Probleme:

- Mathematik ist nicht trivial
- Wenig Erkenntnisse ob besser oder schlechter als andere Verfahren

Vorteile:

- Funktioniert :)
- Kommt mit lokalen Sicht Informationen aus

Sichtgraph

Idee

Algorithmus sucht in der Ebene nach dem Weg um die Ecken

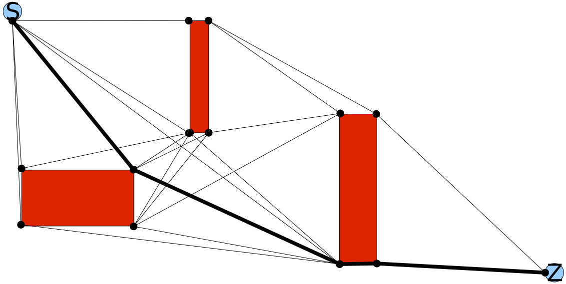
Vorgehen: Verbindungen zwischen den Ecken von Start, Ziel und den Hindernissen finden.

Sichtgraph

Idee

Algorithmus sucht in der Ebene nach dem Weg um die Ecken

Vorgehen: Verbindungen zwischen den Ecken von Start, Ziel und den Hindernissen finden.



Sichtgraph

Probleme:

- Funktioniert nur in $\mathbb{R}^2$
- Diskretisierung (Finden der Ecken) nicht trivial

Vorteile:

- Algorithmus ist sehr schnell
- Kann lokal und global arbeiten.

Potential Felder

Vorgehen:

1. Weise jedem Hinderniss ein Potential zu
2. Potentialverteilung (bzw. Funktion) ergibt "Landkarte"
3. Hohes Potential = Hinderniss
4. Niedriges Potential = Tal = freier Weg

Einfacher Ansatz

Laufe immer in Richtung von niedrigem Potential (senkrecht zu den Gradienten)

Lokale Minima müssen vermieden werden (Sackgassen).

Potential Felder

Vorgehen:

1. Weise jedem Hinderniss ein Potential zu
2. Potentialverteilung (bzw. Funktion) ergibt "Landkarte"
3. Hohes Potential = Hinderniss
4. Niedriges Potential = Tal = freier Weg

Einfacher Ansatz

Laufe immer in Richtung von niedrigem Potential (senkrecht zu den Gradienten)

Lokale Minima müssen vermieden werden (Sackgassen).

Verbesserter Ansatz

1. Plane Weg durch die Täler
2. Laufe dann kollisionsfrei von Tal zu Tal
3. Bei Fehlschlag, wähle anderen Weg in 1

Potential Felder

Probleme:

- Verbesserter Ansatz funktioniert nur bei niedrig dimensionalen Räumen
- Einfacher Ansatz muss lokale Minima vermeiden
- Verbesserter Algorithmus muss eventuell oft probieren.
- Wahl der Potentialfunktion ist sehr schwierig
- Engpässen sind problematisch
- Findet ev. keine Lösung

Potential Felder

Probleme:

- Verbesserter Ansatz funktioniert nur bei niedrig dimensionalen Räumen
- Einfacher Ansatz muss lokale Minima vermeiden
- Verbesserter Algorithmus muss eventuell oft probieren.
- Wahl der Potentialfunktion ist sehr schwierig
- Engpässen sind problematisch
- Findet ev. keine Lösung

Vorteile:

- Algorithmus ist schnell
- Überführung von $\mathcal{C}_{free}$ in den Graphen ist einfach (Täler verbinden)

A*-Algorithmus

- 1988 entwickleter Algorithmus, Verknüpft heuristische und formale Mehtoden
- Sucht kürzeste Wege auf Graphen
- Durch Heuristik werden naheliegende Knoten zuerst untersucht
- Wenn Heuristik “gut”, sind Wege optimal

A*-Algorithmus

- 1988 entwickleter Algorithmus, Verknüpft heuristische und formale Mehtoden
- Sucht kürzeste Wege auf Graphen
- Durch Heuristik werden naheliegende Knoten zuerst untersucht
- Wenn Heuristik “gut”, sind Wege optimal

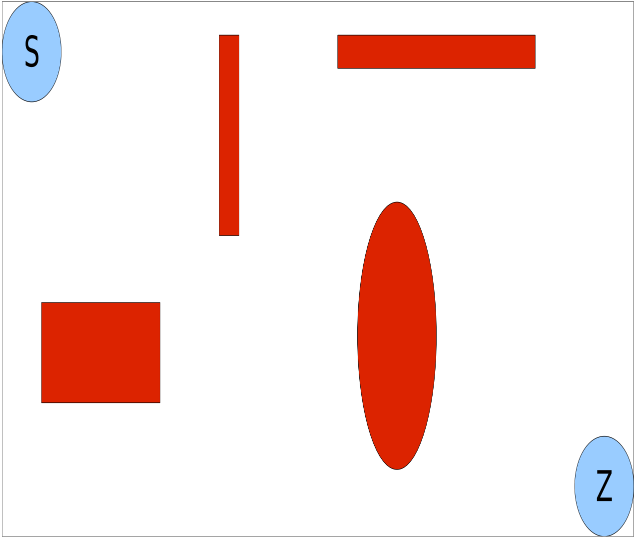
Graphen?

A* arbeitet auf Graphen, wir haben aber $\mathcal{W} = \mathbb{R}^3$ bzw $\mathcal{W} = \mathbb{R}^2$.

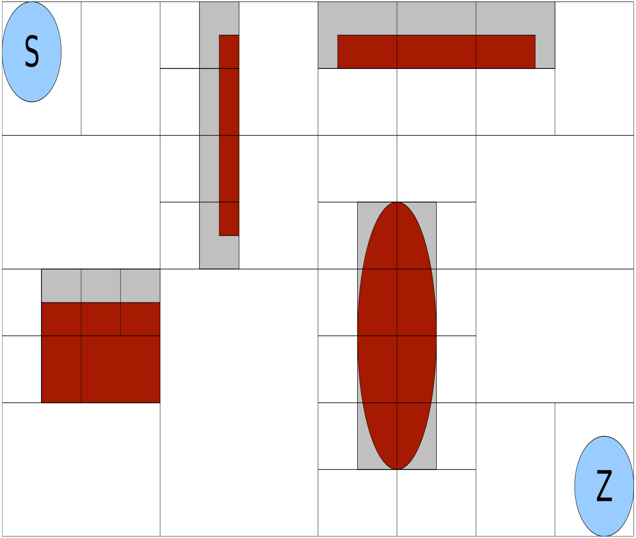
Exkurs: Transformation von $\mathbb{R}^2$ in einen Graphen

- Verwendung von Quadtrees zur eddizienten Diskretisierung!
- Idee: Rekursive Aufteilung des Raumes in Quadrate.
- Analog im $\mathbb{R}^3$: Octtree, Alternative: R*-Bäume
- Auch inkrementell und dynamisch möglich

A*-Algorithmus



A*-Algorithmus



A*-Algorithmus

Vereinfachtes Vorgehen von A*:

1. Raum in Graph überführen
2. Alle Knoten bekommen Kosten ∞, Start Kosten 0
3. Mittels Prioritätswarteschlange immer den Knoten v mit dem kleinsten Wert $f(v) := g(v) + h(v)$ betrachten.
   $g(v) :=$ Kosten von Start zu v, $h(v) :=$ Heuristikkosten über v zum Ziel
4. An diesem v alle Nachbarn betrachten, f neu berechnen.
5. Ende wenn Ziel gefunden, oder Knoten zweimal betrachtet wird (Fehler)
6. Rückwärts den Weg berechnen

A*-Algorithmus

Vereinfachtes Vorgehen von A*:

1. Raum in Graph überführen
2. Alle Knoten bekommen Kosten ∞, Start Kosten 0
3. Mittels Prioritätswarteschlange immer den Knoten v mit dem kleinsten Wert $f(v) := g(v) + h(v)$ betrachten.
   $g(v) :=$ Kosten von Start zu v, $h(v) :=$ Heuristikkosten über v zum Ziel
4. An diesem v alle Nachbarn betrachten, f neu berechnen.
5. Ende wenn Ziel gefunden, oder Knoten zweimal betrachtet wird (Fehler)
6. Rückwärts den Weg berechnen

Laufzeit: $O(|V| \log |V| + |E|)$, mit Fibonacci Heap als Prioritätswarteschlange.

Probabilistic Roadmap Methods

Aufteilung in zwei Schritte:

- Lernphase (auch Preprocessing, Konstruktionsphase)
- Abfragephase

Probabilistic Roadmap Methods

Aufteilung in zwei Schritte:

- Lernphase (auch Preprocessing, Konstruktionsphase)
- Abfragephase

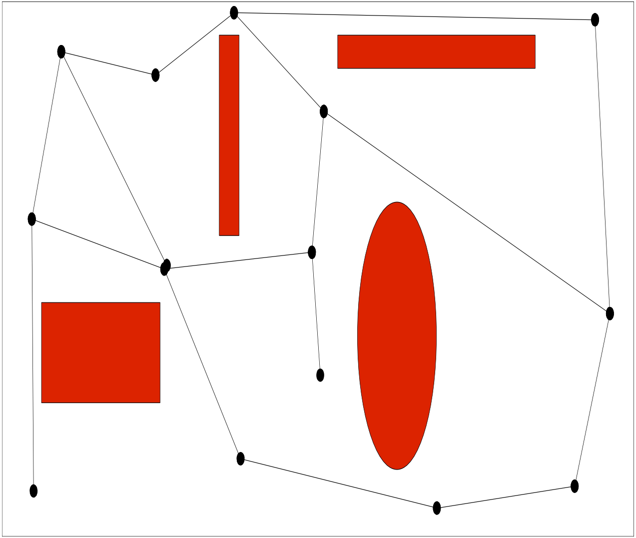
Lernphase

Annäherung von $\mathcal{C}_{free}$ durch zufälliges Auswählen und Verbinden von Konfigurationen

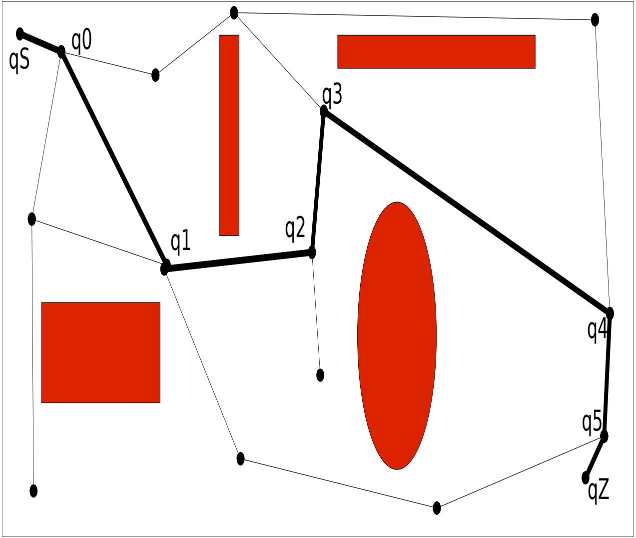
Abfragephase

Einfache kürzeste Wegesuche auf dem entstandenen Graphen (sog. Roadmap).

PRM Methoden



PRM Methoden



PRM Methoden

- Effizient bei statischen Problem (Amortisation der Lernphase während der Abfragephasen)
- Bei wechseldem $\mathcal{C}$ ist ein Lazy-PRM Ansatz notwendig
- Enge Passagen problematisch wegen zufälliger Auswahl
- Komplexere Lernmethoden möglich (Optimierung des Graphen)

Rapidly-Exploring Random Tree

Randomisiertes "Entecker" Verfahren.

- Löst auch nicht holonome Systeme hohen Gerades (z.B. $\mathbb{R}^{12}$(!))
- Strebt in unendeckte Gebiete
- Benötigt keine Suche nach Verbindungen zwischen Konfigurationen
- Ideal für "enge" Passagen

Rapidly-Exploring Random Tree

Randomisiertes “Entecker” Verfahren.

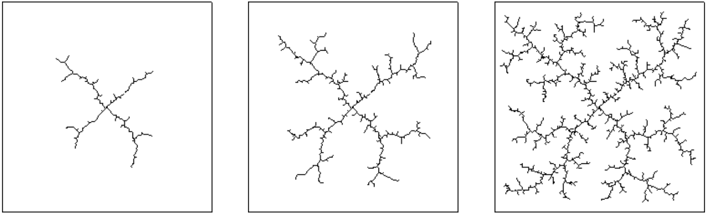
- Löst auch nicht holonome Systeme hohen Gerades (z.B. $\mathbb{R}^{12}$(!))
- Strebt in unendeckte Gebiete
- Benötigt keine Suche nach Verbindungen zwischen Konfigurationen
- Ideal für “enge” Passagen

Vorgehen:

1. Zufällige Konfiguration q_{rand} wählen
2. Nächsten Baumknoten q_{near} finden
3. Mittels Konvergenz q_{next} wählen, ein Stück von q_{near} nach q_{rand}
4. q_{next} an q_{near} mittels Kante anfügen.

Rapidly-Exploring Random Tree

Ein RRT, $C = [0, 100] \times [0, 100] \subset \mathbb{R}^2$
$q_{init} = (50, 50)$ (Baumwurzel)



Vergleich der Verfahren

Übersicht:

Verfahren	Raum	lokal	dyn. h	Sonstiges
Voronoi	$\mathbb{R}^2$	-	-	einfach
HGVG	$\mathbb{R}^3$	$\surd$	-	kompliziert
Sichtgraph	$\mathbb{R}^2$	$\surd$	$\surd$	sehr schnell
Potential	$\mathbb{R}^3$	-	-	schlägt ev. fehl
Verb. Potential	$\mathbb{R}^n$	-	-	wahl der Fkt. schwer
A*	$\mathbb{R}^3$	-	-	schnell
PRM	$\mathbb{R}^n$	$\surd$	$\surd$	schnell
RRT	$\mathbb{R}^n$	$\surd$	$\surd$	schnell, effizient

Ausblick

RRT ist wohl das erfolgsversprechendste Single Query Verfahren

Es gibt viel aktive Forschungsarbeit

In der Praxis oft Kombination von mehreren Verfahren

Neuronale Netzwerke

Ausblick