AlgGeo 13.1
Bewegungsplanung

bisher: (fest) nur vollständige Information

~> nicht immer realistisch

Bsp: Entkommen aus Labyrinth

Labyrinth bekannt

-> kürzesten Weg berechnen und folgen

-> ~ Dijkstra

Labyrinth unbekannt

-> aufbrechen und erkunden

Maß des Erfolgs

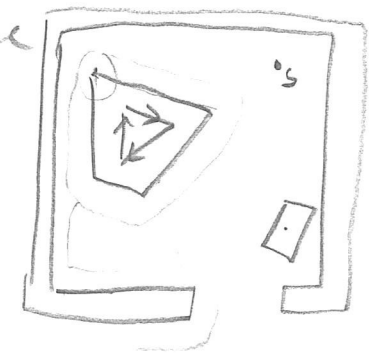
← Rechenzeit

→ Pfadlänge
Erfolgsgarantie?

Anwendung: autonomer Roboter

Labyrinthmodell: endliche Menge einfacher, geschlossener polygonaler Ketten

Entkommen ≙ konvexe Hülle der Hindernisse verlassen



Roboter: punktförmig
-> deckt auch kreisförmig mit ab

Sensoren: Tastsensor
Sichtsensor

Erinnerung: Distanz?
Drehungen?
Koordinaten?

AlgGeo 13.2

Idee? "Linke-Hand Regel"

-> benötigt Tastsensor

Schritt 1: in beliebige Richtung loslaufen -> Ausgang o. Wand

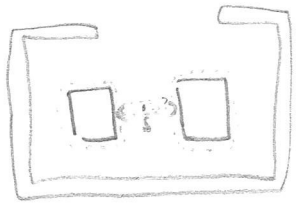
2: linke Hand an die Wand und Wand folgen -> Ausgang

Problem: lösen uns nie von Hindernis

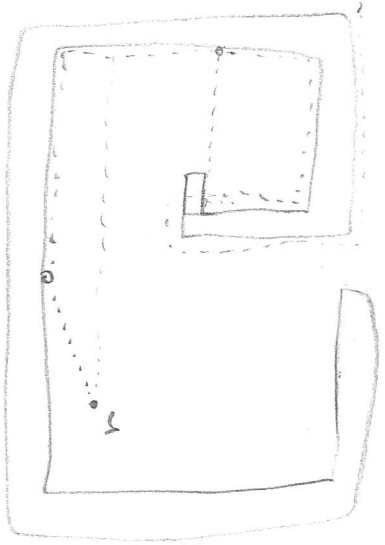
Ideen 1) lösen wenn wir "Ersthandabdruck" erreicht | Problem:

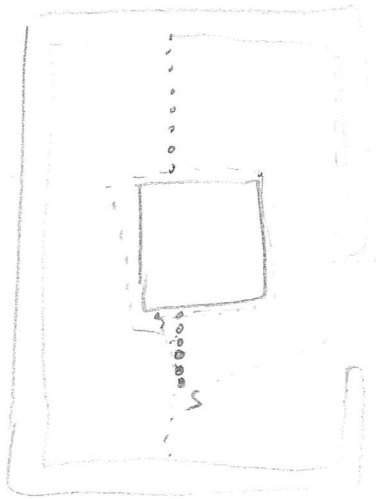
2) Winkel merken -> ursprüngliche Richtung = 0

sobald wir am Hindernis wieder Drehung $\bmod 2\pi = 0$ haben -> lösen in ursprüngliche Richtung



Idee 3) ↗ so wie Idee 2) aber nur bei Drehung = 0.





Lsg Idee 2) Idee 3)

=> "Pledge"-Algorithmus

Th. 7.1. Pledge funktioniert.

L. 7.2. Winkelzahler immer negativ.

-> Sketch der Argument

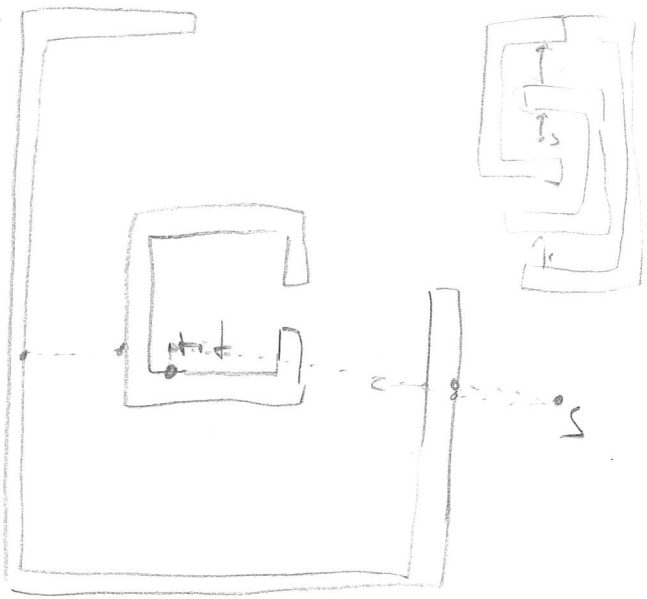
Beginnen jedes Hindernis negativ und lösen uns sobald es 0 wird

also "bevor" es positiv werden kann. □

Alg Geo 13.3

Suchen im Labyrinth

Gegeben: Roboterkoordinaten, Zielkoordinaten t, Tastsensor
↳ Entfernung zum Ziel & zurückgelegte Strecken



Idee: sich t mit jedem Hindernis nähern.

1 -> auf t zulaufen bis Hindernis,

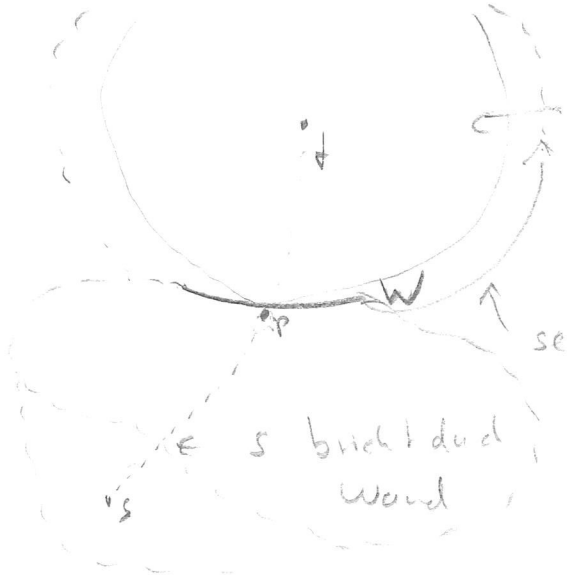
2 -> auf Hindernis den Punkt am nächsten zu t finden

3 -> dahin gehen und wieder Richtung t laufen

klar: jedes Schritt kommt t näher

=> kein Hindernis 2-mal besucht

unklar: „funktioniert" Schritt 3, d.h. ist der Weg frei



=> weg ist frei

Effizienz? ≙ Weglänge

$$|st| + \frac{3}{2}\sum_{i=1}^{n} U_i$$

U_i = Umfänge jeden Hindernisses

Pledge & Suchen bei beliebig schlecht

Widerspruchsbeweis: Annahme: Es gibt einen Ausweg aber Pledge findet ihn nicht.

L.7.3. Pledge gerät in eine Endlosschleife
d.h. wiederholt ein bestimmtes Wegstück P immer wieder
↳ auch polygonale Kette

endlich viele Abbiegekandidaten für Pledge

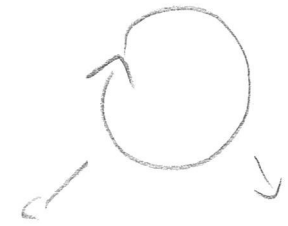
1) Eckpunkte der Hindernisse
2) Auftreffpunkte in "gerader" Richtung hinter Eckpunkten

+ Pledge deterministisch

-> Fall 1) irgendwann gleicher Eckpunkt mit gleichem Winkelzähler zum zweiten Mal => Schleife
Fall 2) irgendwann keine Eckpunkte mit Winkelzähler 0 mehr -> Pledge macht Schleife an einem Hindernis. □

2 Möglichkeiten

P schnittfrei



+2π pro Runde

oder

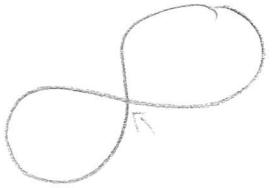
-2π pro Runde

-> Zähler wird positiv

=> §L7.2

-> kann keinen Ausweg geben.

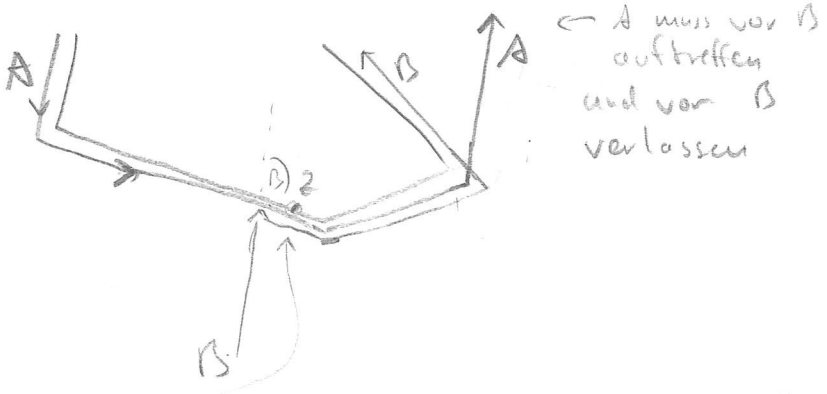
P selbstschneidend. Eigentlich Schnitt muss an einem Hindernis passieren



± 0 pro Runde

L.7.4.

P schneidet sich nicht selbst



↳ Winkelzähler von A und Bunterschiedlich

$W_B(z) = -\beta$

$W_A(z) = -\beta \equiv k \cdot 2\pi$ => B wird zuerst 0
=> lässt sich also §L7.2 □

AlgGeo

Behauptung: es geht ohne Koordinaten, nur mit Komplex

Erkenntnisse:

∇: geradeaus auf d zu

3 Grundskills

R: rechter Hand bis zum nächsten Ecke

L: linker ——— " ———

∃ Weg von s nach t in ∇LR - kodierung ⇝ Wort aus Σ^* mit $\Sigma = \{\nabla, L, R\}$

endliche Menge von Knickpunkten
- Startpunkt s
- Eckpunkte
- Schnittpunkte von Eckpunkten aus

} Set $\{p_1, p_2, \dots, p_m\}$

L.78

∃ universelles Wort, das von jedem Punkt $\in p_i$ zu ↓ führt.

klar ∃ $w_i = w(p_i)$ für jeden punkt p_i nach ↓.

Fall 1: Startpunkt $s = p_1$ ⇝ w_1 ✓

Fall 2: $s = p_2$ => deterministisch führt w_1 angewendet auf p_2 zu $q_2 \in P_i$

=> $w_2 = w_1\, w(q_2)$

$s = p_3$ => führt w_2 zu q_3 => $w_1\, w(q_2)\, w(q_2)$

⋮

p_m => schwieriges Wort □

Problem: Wort unbekannt. -> aber wir probieren einfach alle abzählbar vielen Worte durch. => sehr ineffizient.