Kapitel 6

Navigation

Zu den Aufgaben der Navigationskomponente gehören

- Wegeplanung in einer topologischen oder geographischen Karte
  Festlegung von Wegpunkten
  Kriterien: Optimalität in Bezug auf
  Fahrzeiten
  Weglängen
  Hindernisabstände
  } widersprüchliche Anforderungen
  Auswahl durch übergeordneten Planer
- Bahnplanung:
  Festlegung des genauen Kurses zwischen Wegpunkten
- Bahnregelung:
  Überwachen der Einhaltung der vorgegebenen Bahn

6.1. Wegeplanung

6.1.1. Wegeplanung auf einem topologischen Graphen

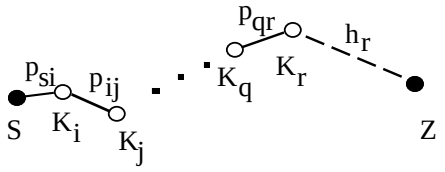
6.1.1.1. A-Algorithmus*

Gegeben:
Zusammenhängender topologischer Graph mit Knoten $K_1,\ldots, K_n$ und Kanten s_{ij} zwischen Knoten K_i und K_j und "Preisen" p_{ij} an den Kanten
(nicht vollständig, kann Zyklen haben; Preise = Fahrtkosten , Wegstrecken, …)

Gesucht:
Billigste Verbindung von Startknoten S zu Zielknoten Z

Zu jedem Knoten K_i gibt es eine optimistische Abschätzung h_i des Preises, der für den direkten Weg von K_i --> Z zu zahlen ist. Bei Z ist h = 0.

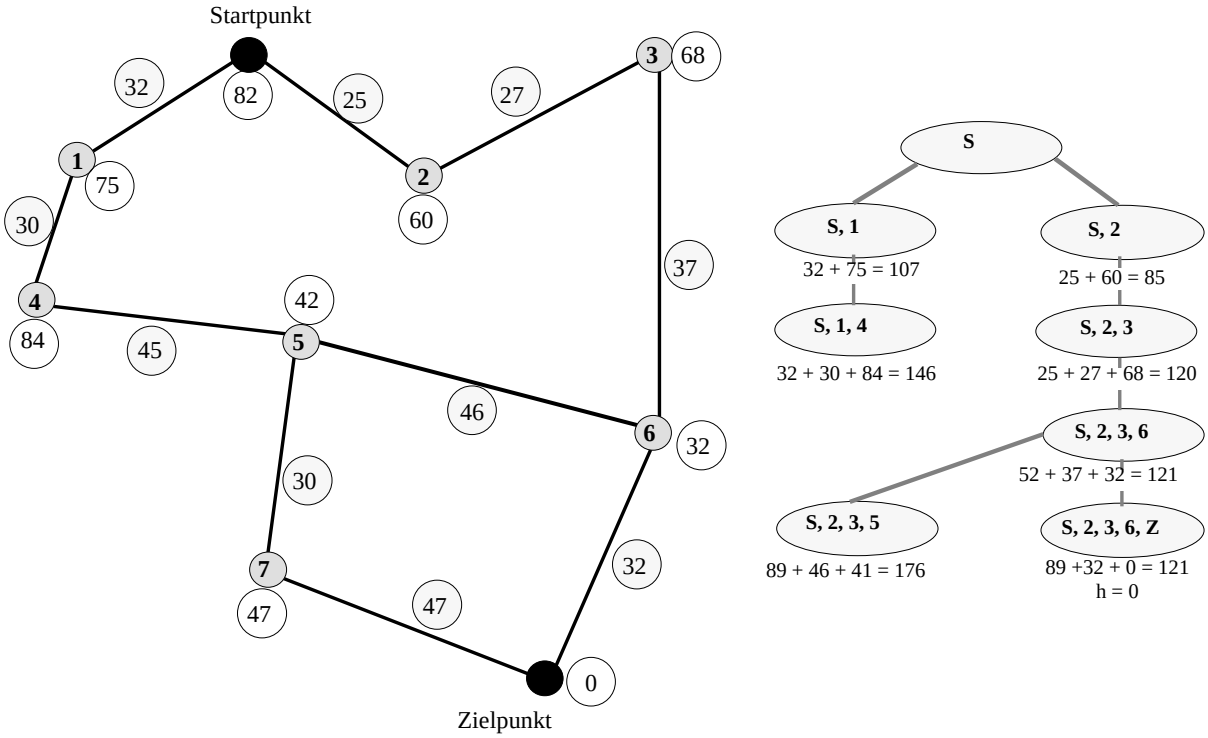
Die Gesamtkosten eines Weges $(S, K_i, K_j, \ldots, K_q, K_r)$ sind $g = (p_{si} + p_{ij} + \ldots + p_{qr}) + h_r$.



Ein Algorithmus, der die kostengünstigste Lösung findet, ist 1969 von Nilson angegeben worden, der A*-Algorithmus.

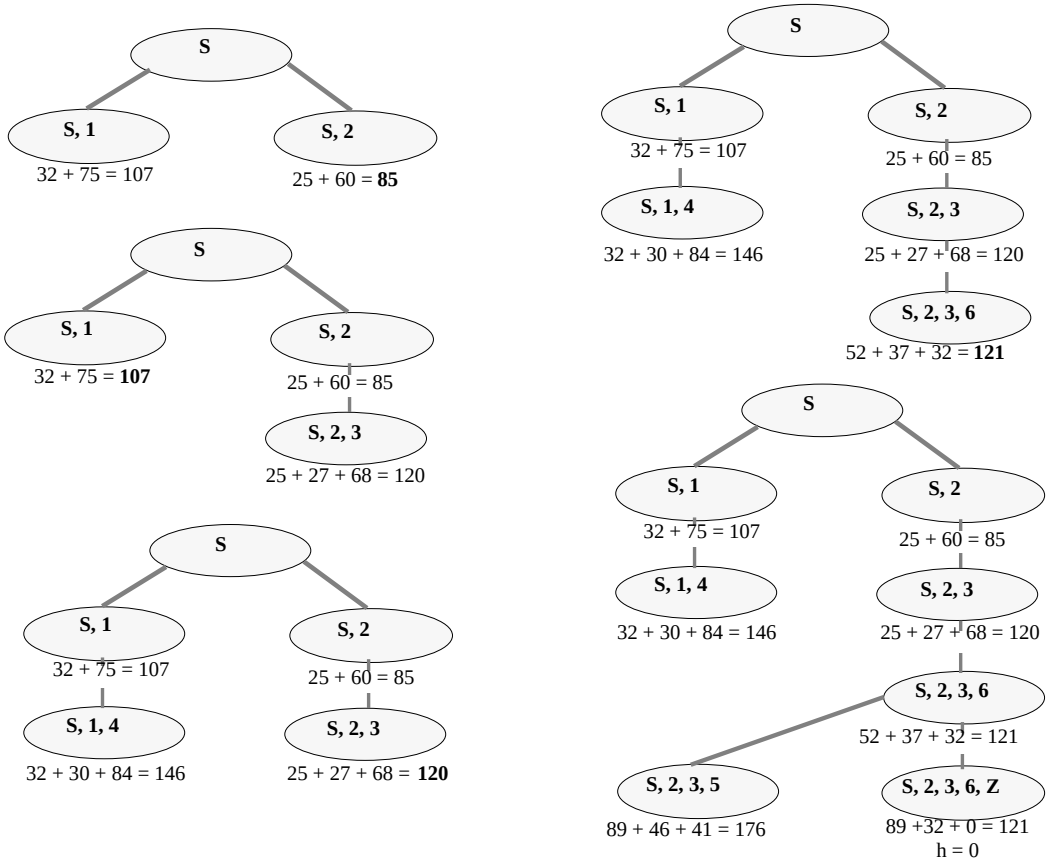
Sei W_r der kostengünstigste Weg von S nach Z bis zum Knoten K_r . Sei K_q der letzte Knoten, der besucht wurde. Entwickle K_r : von K_r aus werden außer K_q die Knoten $K_{s1}, \ldots, K_{sn}$ erreicht mit Kosten $g_{s1}, \ldots, g_{sn}$. Seien die geringsten Kosten beim Weg über K_v erreicht. Dann entwickle K_v weiter. K_v braucht dabei nicht zu den Knoten $K_{s1}, \ldots, K_{sn}$ zu gehören, sondern kann Endpunkt eines anderen schon betrachteten Weges sein. Merke alle bisherigen Wege und ihre Kosten. Wenn h = 0 geworden ist, dann ist das Ziel Z erreicht.

Das Vorgehen wird an untenstehendem Beispiel erläutert:

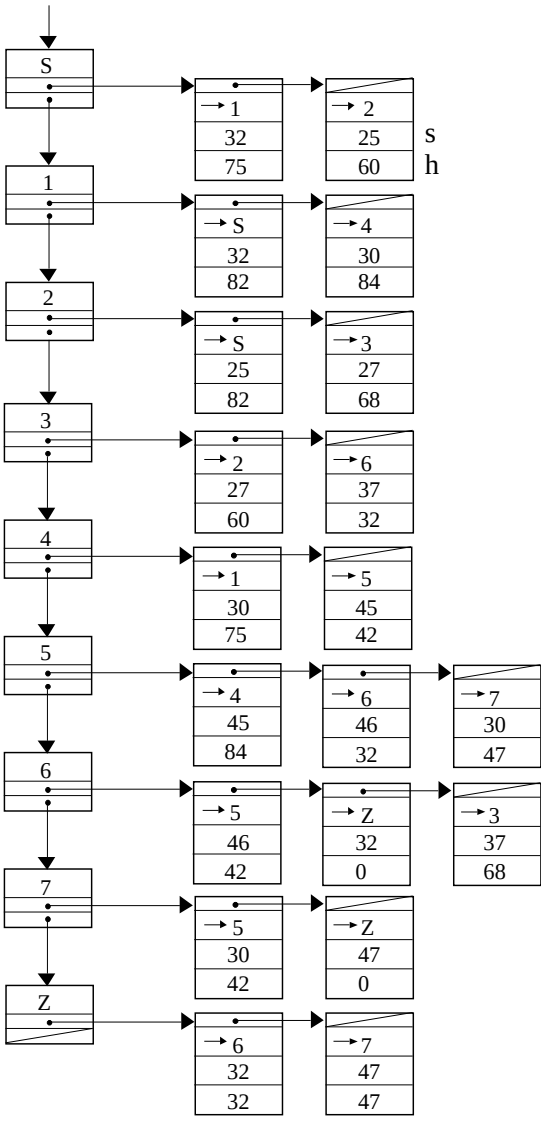


Bei jedem Knoten ist h_r angegeben und an den Kanten der Preis p_{qr} (Entfernung).

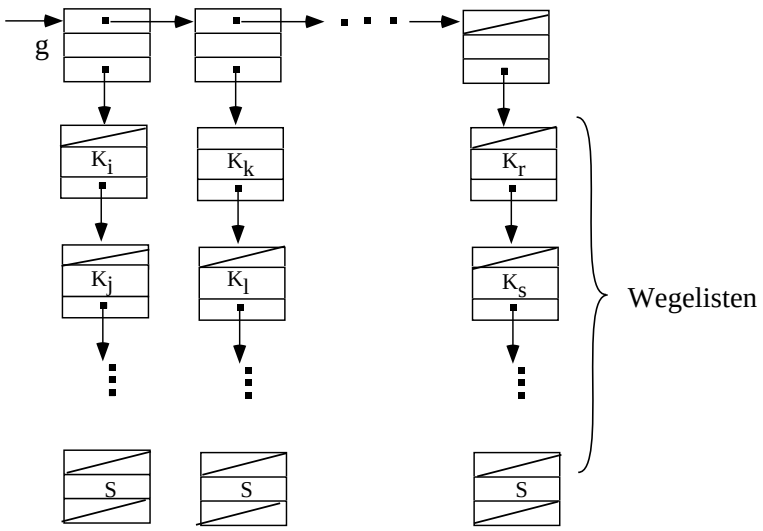
Das folgende Bild zeigt die Abfolge der Schritte des A*-Algorithmus am obigen Beispiel.



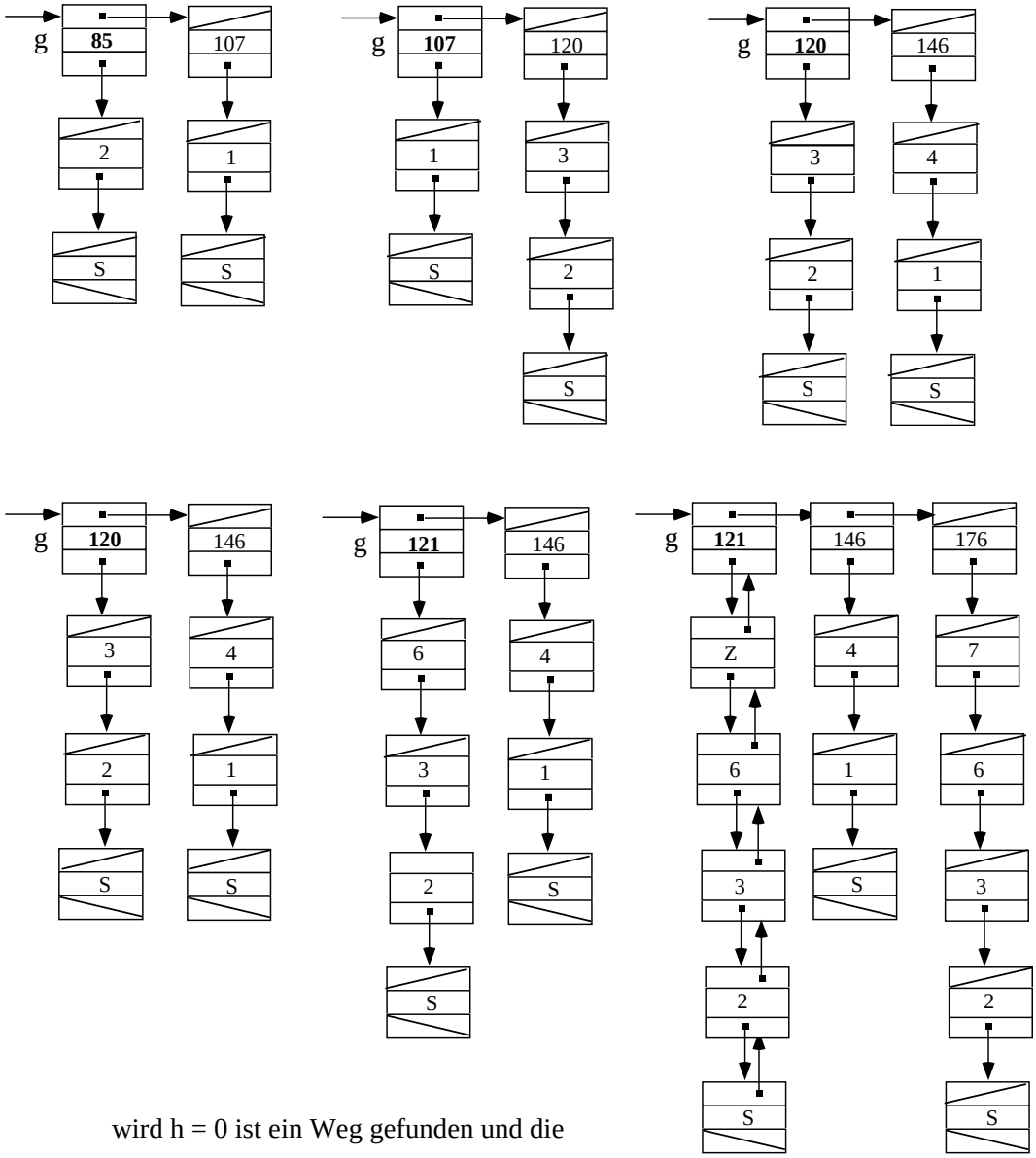
Die Implementierung kann z.B. den Graphen als Adjazenzliste darstellen mit den Zusatzangaben über die Preise s und die optimistische Abschätzung der Kosten zum Ziel h.



Daneben wird eine Blattliste aufgebaut, nach steigenden Werten von g geordnet.



Auf ihr wird der A*-Algorithmus entwickelt: die Liste enthält alle untersuchten Wege, ist vom letzten Knoten zum Anfangsknoten S hin verkettet, aber mit NIL in den Rückwärtspointern, und ist nach g geordnet und terminiert, wenn h = 0 gefunden wurde.



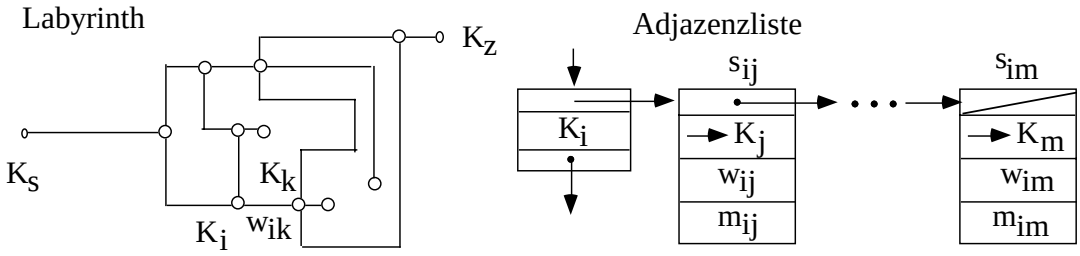
wird h = 0 ist ein Weg gefunden und die

Wegeliste wird rückwärts verkettet:Weg von S --> Z gefunden

6.1.1.2. Wegeplanung in einem Labyrinth

Gegeben:
Ein Labyrinth in Form eines attributierten Graphen mit Knoten K_i, einem Startknoten K_S und einem Zielknoten K_Z, der allerdings erst an Hand eines Attribut erkannt wird, wenn er unmittelbar gefunden wurde.
Es gibt keine Abschätzung h_i zum Zielknoten, aber die Entfernungen w_{ik} zwischen Knoten K_i und K_k sind bekannt. Der Graph kann Zyklen enthalten.

Gesucht:
Ein Weg $(K_S,\ldots,K_i,\ldots K_Z)$ von K_S nach K_Z



Der Graph sei implementiert als Adjazenzliste mit der Entfernung w_{ij} bei der Kante s_{ij} und einer Markierung m_{ij} ε {"nichts", "frei", "belegt", "gesperrt"} neben dem Pointer auf den Knoten K_j.

Backtracking Algorithmus zum Lösen eines Labyrinths

Initialisierung: i:= a; w := 0; Markierung für alle Kanten "nichts"

(*) wenn Knoten $K_i = K_z$ ==> Ziel erreicht; w = Weglänge zum Ziel;◊

sonst im Knoten K_i betrachte alle n Kanten s_{ij}, …, s_{im} ,
die von K_i ausgehen; K_i wurde von K_j her erreicht;

wenn m_{ij} = "nichts" {* erster Besuch bei Knoten K_i *}
==> m_{ij} := "begangen;
wenn es Kanten mit Markierung "nichts" gibt
==> markiere alle Kanten mit "frei"
nimm eine Kante (s_{ik}); m_{ik} := "begangen"
fahre nach K_k; w := w + w_{ik}; i:= k; -->(*)

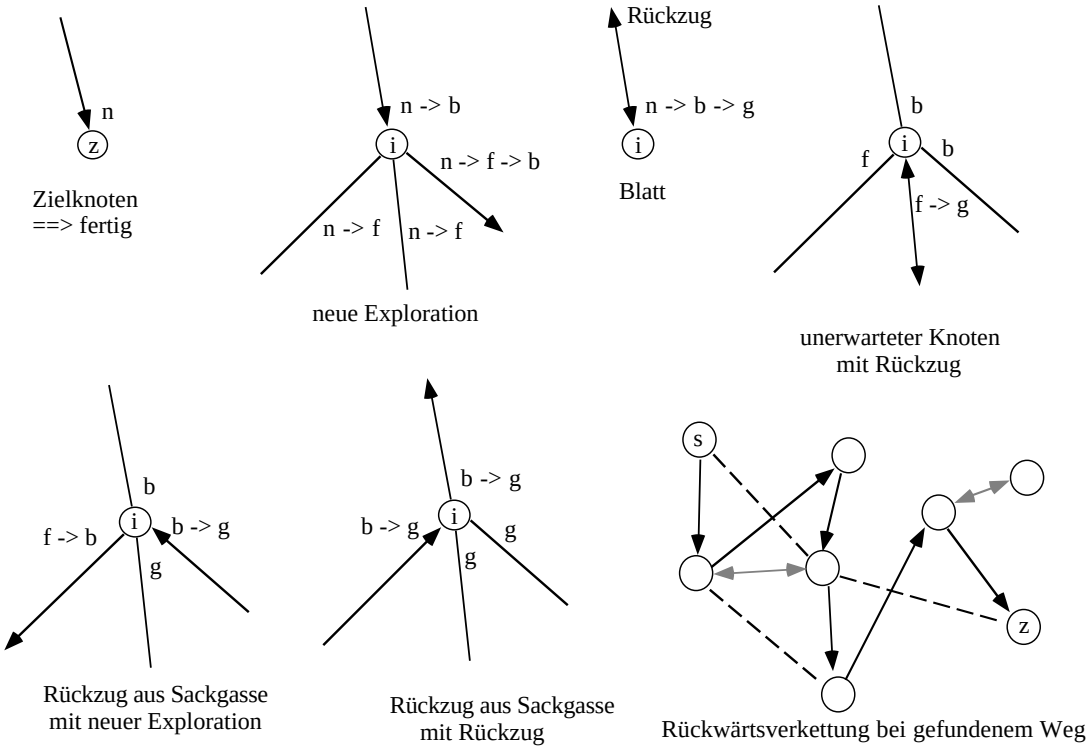
sonst {* Blatt des Graphen *}
m_{ij} := "gesperrt" {*Sackgasse*}
fahre nach K_j zurück; w := w - w_{ij}; i := j; -->(*)

wenn m_{ij} = "frei" {* Knoten wird unerwartet erneut besucht*}
==> m_{ij} := "gesperrt" {*Sackgasse*}
fahre nach K_j zurück; w := w - w_{ij}; i := j; -->(*)

wenn m_{ij} = "begangen" {* Rückzug aus einer Sackgasse*}
m_{ij} := "gesperrt";
wenn es Kante s_{ik} mit Markierung "frei" gibt
==> m_{ik} := "begangen"
fahre nach K_k; w := w + w_{ik}; i := k; -->(*)

sonst {* alle Kanten bis auf s_{ir} sind gesperrt *}
m_{ir} := "gesperrt"
fahre nach K_r zurück; w := w - w_{ir}; i:= r; -->(*)

Der Entscheidungsbaum im Algorithmus ist hier noch einmal mit den Abkürzungen n für "nichts", f für "frei", b für "belegt" und g für "gesperrt" verdeutlicht:



6.1.2. Wegeplanung auf geometrischen Karten

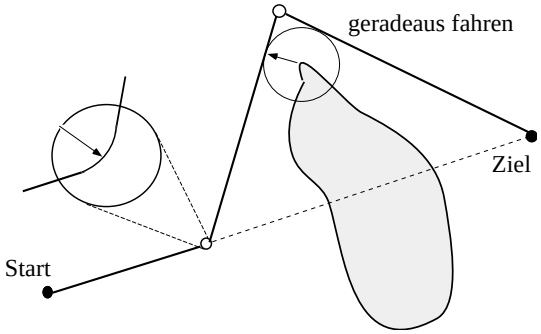
Festlegung von Kurspunkten auf einem Weg vom Start zum Ziel zwischen denen befahrbare Wege liegen.

6.1.2.1. Küstenschifffahrt

Freiflächen sehr groß gegen Abmessungen des Roboters. Abstand zur Küste (Hindernisse) nicht kleiner als d_{min} bei ausreichendem Freiraum und d_{min} >> Roboterabmessungen

Planung gerader Kurse mit "Knicken"

Kurvenradien << d_{min}



Finde Weg über Kurspunkte zwischen zwei Punkten (Start und Ziel)

\# wenn zwischen den Punkten P_S und P_Z ein Hindernis liegt,

suche Seite mit minimalen Punkt maximaler Entfernung zur Geraden Start - Ziel

schlage Kreis mit d_{min} um diesen Punkt,

* lege Tangenten von P_S und P_Z an den Kreis; Sie schneiden sich in Punkt P

untersuche die Geraden P_S - P und P - P_Z auf Hindernisfreiheit ---> #

sonst wenn kein Hindernis im Wege liegt,

suche Punkt minimaler Entfernung zum Hindernis

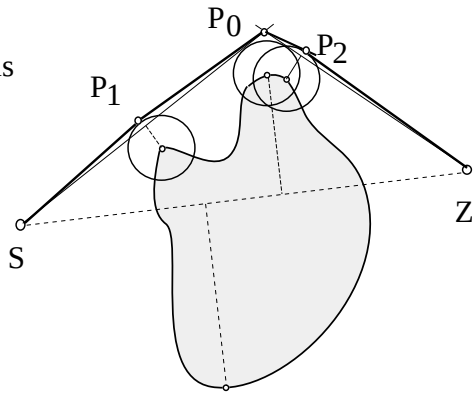
Schlage Kreis mit d_{min} um diesen Punkt

wenn der Kreis die Gerade schneidet,

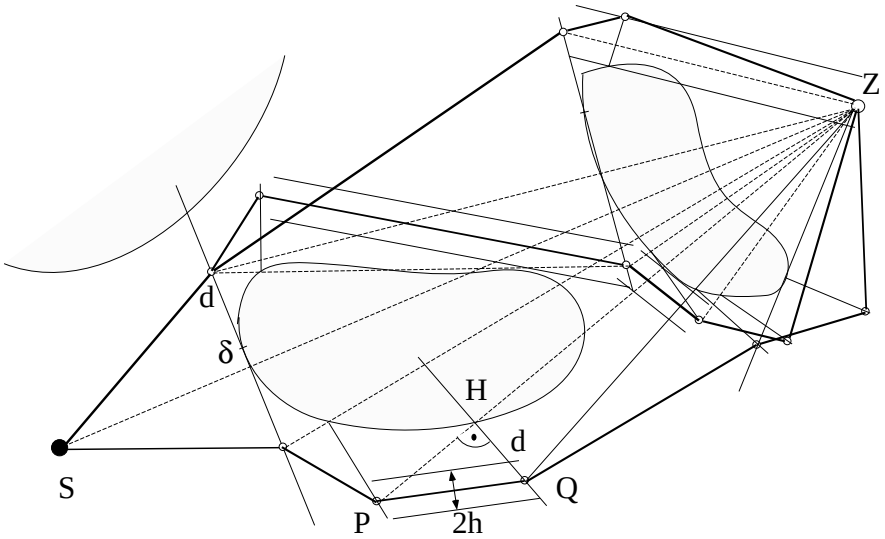
weiter bei *

sonst Hindernis weiter als d_{min} entfernt

fahre in gerader Linie zwischen Kurspunkten

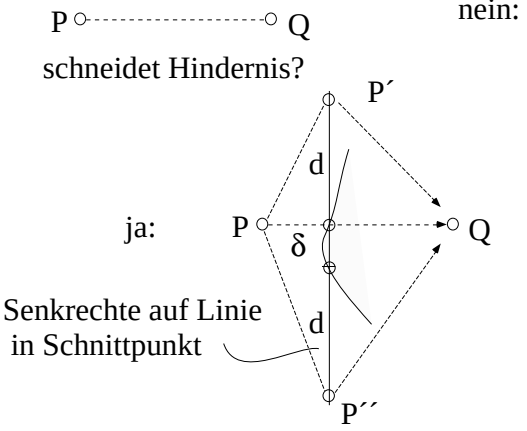


Wegeplanung mit Ausweichpunkten (Schweikard)



(P, Z) schneidet Hindernis bei H; gehe senkrecht zur Linie um d in den Freiraum nach Q; überprüfe rekursiv, ob Linie parallel zu (P, Q) im Abstand h Hindernis schneidet.

Schnittuntersuchung

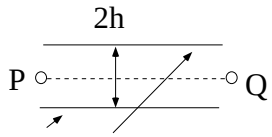


- im Freiraum:
  Umfahrpunkt P´ in Entfernung d
  ==> untersuche (P, P´) und (P´, Q)

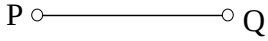
- verläßt Hindernis nach Strecke δ:
  Umfahrpunkt P´´ in Entfernung δ + d
  ==> untersuche (P, P´´) und (P´´, Q)

(P, Q) schneidet kein Hindernis

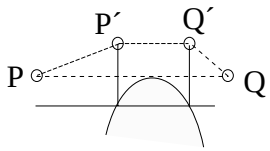
überprüfe Fahrstrasse ± h



schneiden Hindernis?

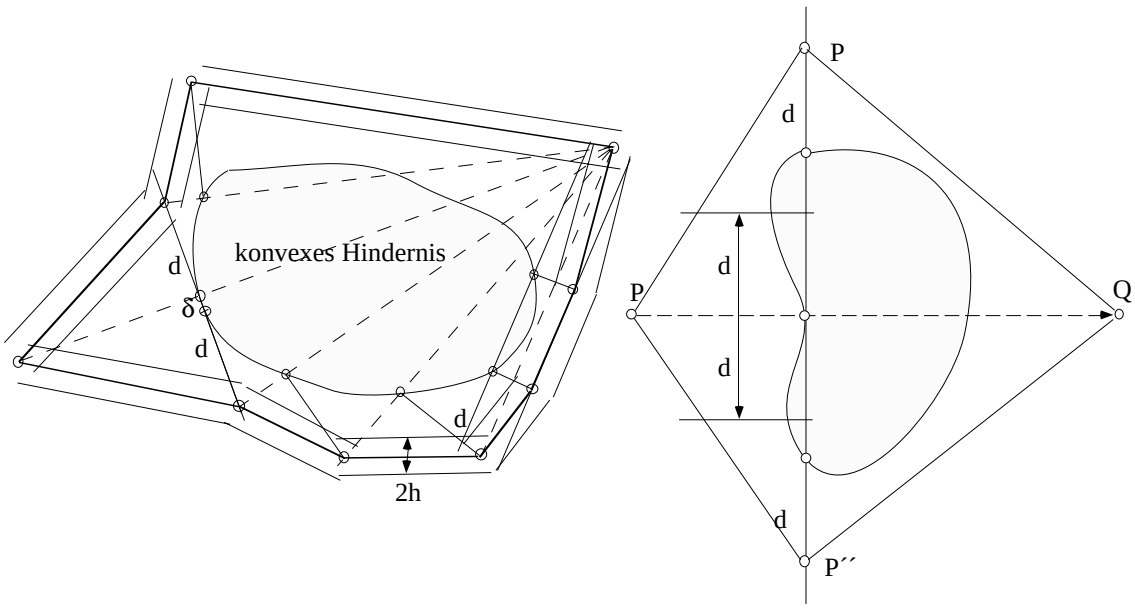
nein: 

Wegstrecke zw. P und Q
untersuche nächstes Teilstück

ja: 

untersuche (P;P´), (P´; Q´), (Q´, Q)

Das folgende Beispiel zeigt das Umfahren eines konkaven und eines konvexen Hindernisses mit diesem Algorithmus.



Wegeplanung in Küstennähe

mit d_{min} ~ Roboterabmessungen und nicht kreisförmigem Roboter

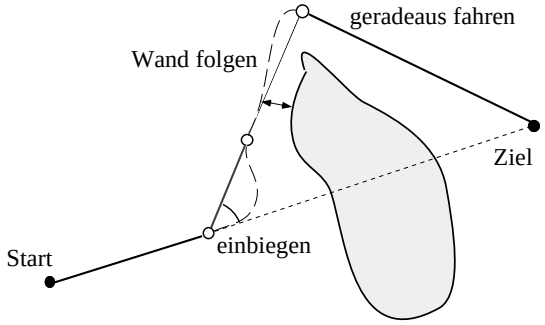
Festlegen von Kurspunkten so, daß zwischen ihnen

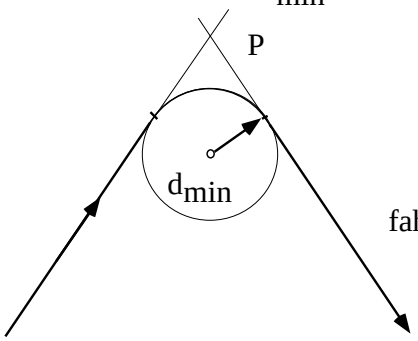
mit bekannten Verfahren navigiert werden kann

(einbiegen, Wand folgen,…)

parametrisiert durch Geschwindigkeit,

Sensorreichweite und d_{min}





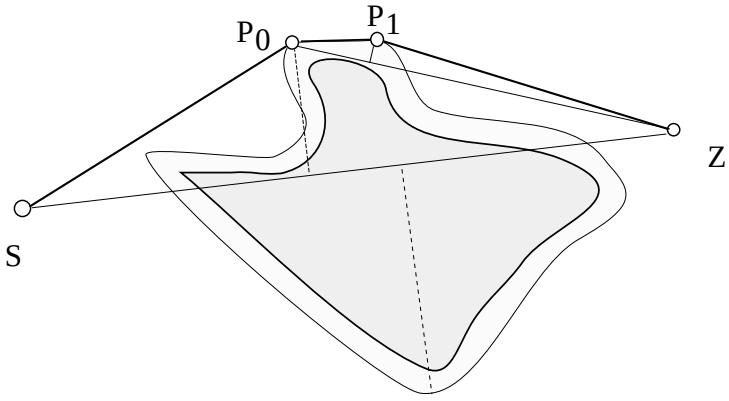
fahren auf dem Grenzkreis statt zum Schnittpunkt

---> Bahnplanung

Aufblähen der Hindernisse (Lonzano-Perez)

Wenn der Robot kreisförmig ist ($r < d_{min}$) oder als punktförmig (bei großem d_{min}) angesehen werden kann, blähe Hindernisse vorab um d_{min} auf.

Berechne Wege mit aufgeblähten Hindernissen und $d_{min} = 0$ (es entfallen die Berechnungen der Tangenten und Minimalentfernungen).



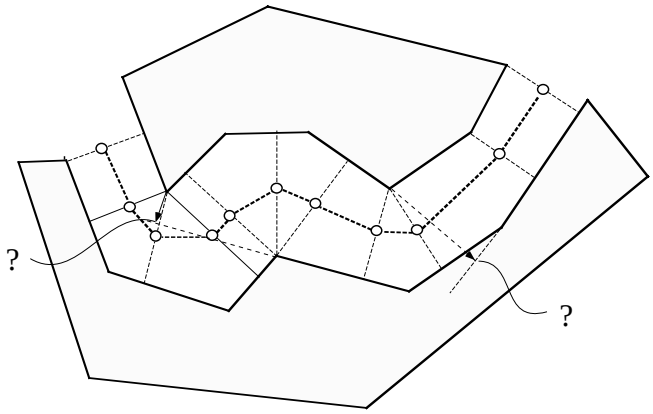
6.1.2.2. Flußschifffahrt

Der Roboter bewegt sich zwischen Hindernissen, deren Abstände nicht mehr groß sind gegen die Roboterabmessungen.

Gesucht:
Kurspunkte mit max. Abstand zu Hindernissen

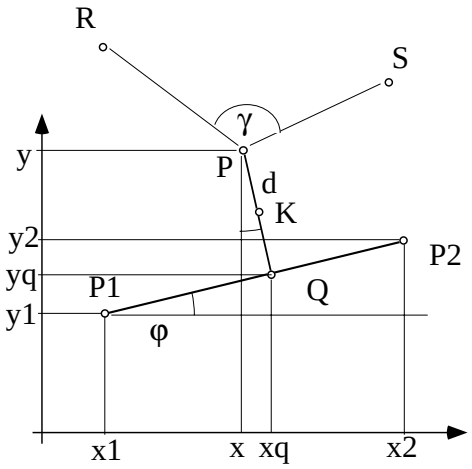
Der Roboter soll zwischen Hindernissen hindurchfahren.

Darstellung durch Polygonkarte



- suche konvexe Hindernispunkte
- fälle das Lot auf das Gegenufer
- berücksichtige nur Lote, mit Fußpunkt auf dem Gegenufer
- die Mittelpunkte der Lote sind die gesuchten Kurspunkte

Seien (R, P) und (P, S) Polygonlinien des einen, (P1, P2) Polygonlinie des anderen Ufers, Winkel (S, P, R) = $\gamma < 180°$ ==> P konvexer Punkt; Q Fußpunkt des Lots auf (P1, P2)



$$\operatorname{tg}\varphi = \frac{y2 - y1}{x2 - x1}$$

$$d = (y2 - y1 + \operatorname{tg}\varphi\,(x - x1))\cos\varphi$$

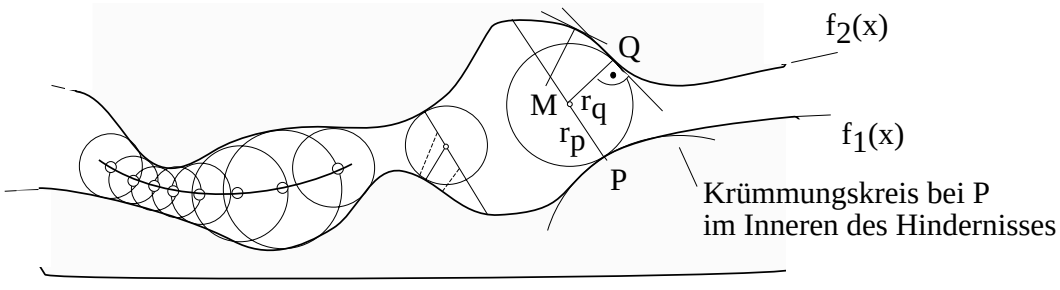
$$xq = x + d\sin\varphi$$

$$yq = y - d\cos\varphi$$

$x1 \le xq \le x2$? & $y1 \le yq \le x2$? — ja ==> Kurspunkt K: $kx = x + (xq - x)/2$, $ky = y + (yq - y)/2$; nein ==> neue Linie suchen

Man findet so die Kurspunkte in der Freifläche zwischen den Hindernissen.

Bei kontinuierlichen Hindernislinien ist die bildliche Vorgehensweise: drücke einen Ballon hindurch, der beide Ufer berührt; sein Mittelpunkt beschreibt eine Bahn mit maximalem Abstand zu den Ufern.

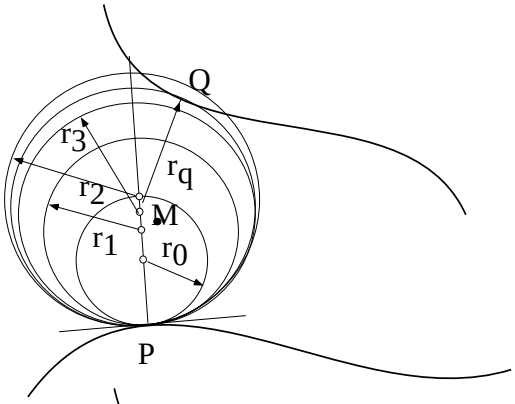


Der Mittelpunkt eines Berührkreises wird so gefunden:

- Sei P konvexer Punkt der Kurve $y = f_1(x)$; d.h. $d^2 f_1(x) / dx^2 < 0$.
- Bilde Normale auf P: $y = m \cdot x + b$ mit $m = - 1 / f_1'(x_p)$ und $b = y_p - m \cdot x_p$.
- Bilde Normale auf dem Gegenufer in Q. Die Normalen schneiden sich in M.
- Variiere Q bis $r_p = | M, P | = r_q = |M, Q |$ ==> M ist Mittelpunkt des Berührkreises.

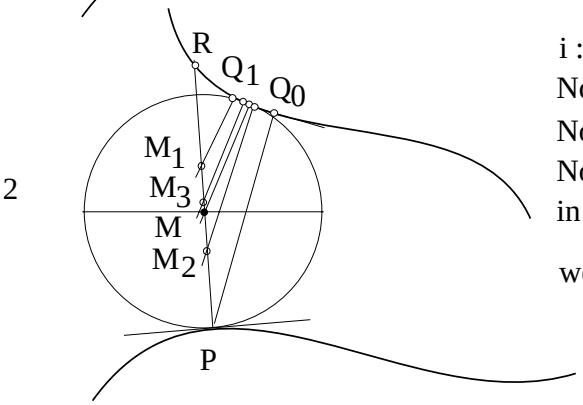
Iteration zum Berührkreis

1



$r := r_0$; $\Delta r := r_0$
* Kreis schneidet Gegenufer ?
nein : $r := r + \Delta r$; --> *
ja: Abstand der Schnittpunkte < ε ?
ja: $r_q = r$; M Kurspunkt; ◊
nein: $\Delta r := \Delta r/2$; $r := r - \Delta r$;
$\Delta r := \Delta r/2$; --> *

2



i := 0;
Normale in P schneidet $f_2(x)$ in R bei $x = x_r$
Normale in Q_0 schneidet $f_1(x)$ in P bei $x = x_p$
Normale Q_{i+1} bei $x_{q_{i+1}} = x_r + (x_r - x_{q_i}) / 2$
in Q_i schneidet (P, R) in M_i

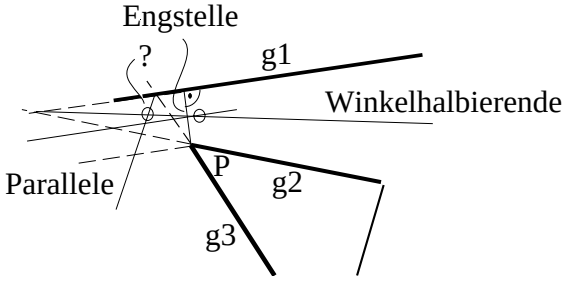
wenn $|(|M_i, P| - |M_i, Q_i|)| < \varepsilon$
==> $r_q := |M_i, P|$; Kurspunkt M := M_i
sonst iteriere

Verallgemeinerte Kegel (nach Brooks)

Suche Kurven gleichweiten Abstandes zwischen Polygonlinien.

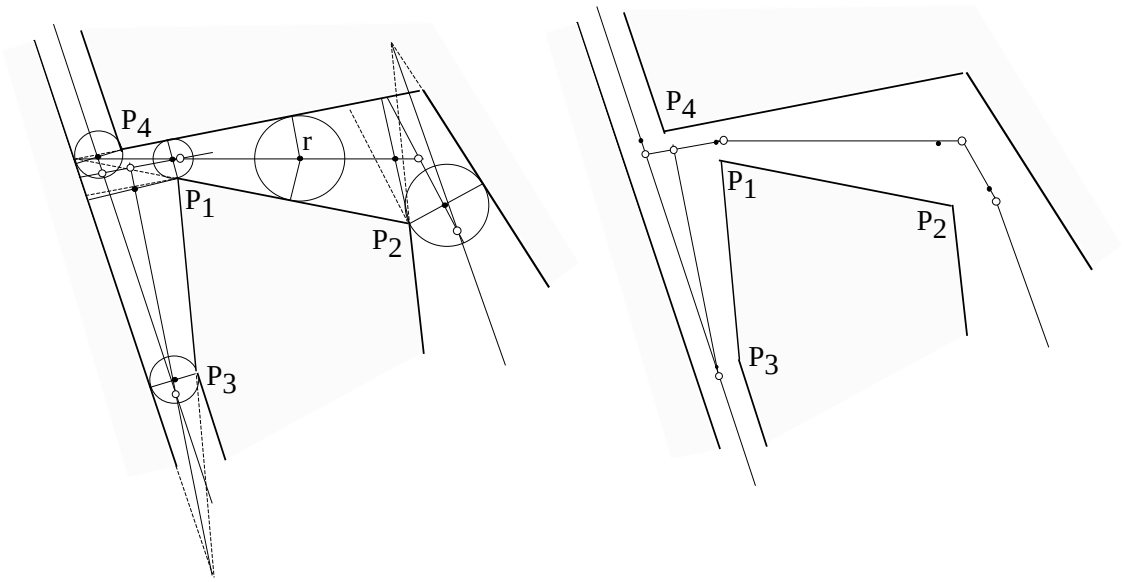
Betrachte Polygonlinien g1, g2, g3. Linien g2 und g3 bilden einen Eckpunkt P.

Linien g1 und g2 bilden einen spitzen Winkel, die Winkelhalbierende ist eine Kurve gleichen Abstands bis zur Engstelle. Die Winkelhalbierende g3, g1 schneidet die andere Winkelhalbierende in einem Punkt, der näher an g1 liegt, als der halbe Abstand |P, g1| an der Engstelle.

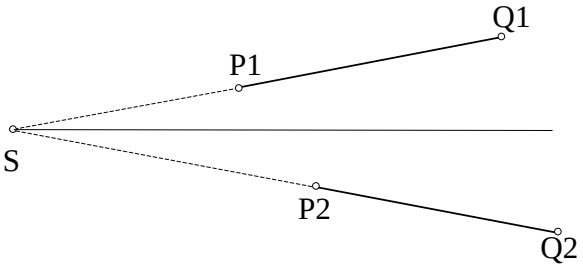


Eine Parallele zu g1 im halben Abstand der Engstelle ist eine Kurve minimalen Abstands zu g1. Diese Parallele und die Winkelhalbierende bilden die Mittellinie eines verallgemeinerten Kegels. Der Schnittpunkt ist ein Wegepunkt für die Navigation.

Das folgende Bild zeigt Hindernisse mit verallgemeinerten Kegeln und den Wegpunkten.



Die Winkelhalbierende berechnet sich aus den Geraden wie folgt:



$$y = \frac{y_{q1} - y_{p1}}{x_{q1} - x_{p1}} (x - x_{p1}) + y_{p1}$$

$$y = tg\ \alpha_1 \ (x - x_{p1}) + y_{p1}$$

$$y = \frac{y_{q2} - y_{p2}}{x_{q2} - x_{p2}} (x - x_{p2}) + y_{p2}$$

$$y = tg\ \alpha_2 \ (x - x_{p2}) + y_{p2}$$

Schnittpunkt S : $(tg\ \alpha_1 - tg\ \alpha_2)\ x_s = tg\ \alpha_1\ x_{p1} - tg\ \alpha_2\ x_{p2} + y_{p2} - y_{p1}$

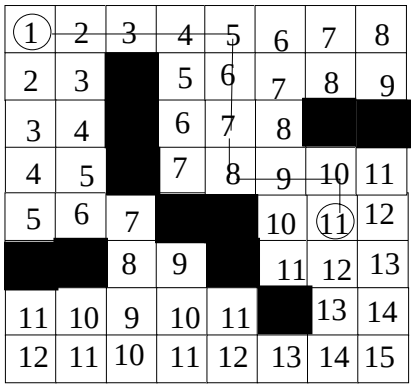
$$x_s = (tg\ \alpha_1\ x_{p1} - tg\ \alpha_2\ x_{p2} + y_{p2} - y_{p1}) / (tg\ \alpha_1 - tg\ \alpha_2)$$

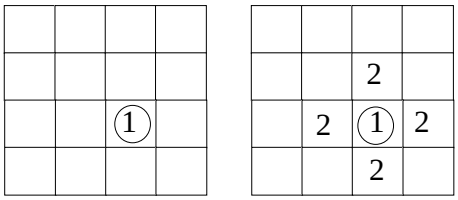
$$ys = tg\ \alpha_1 \ (x_s - x_{p1}) + y_{p1}$$

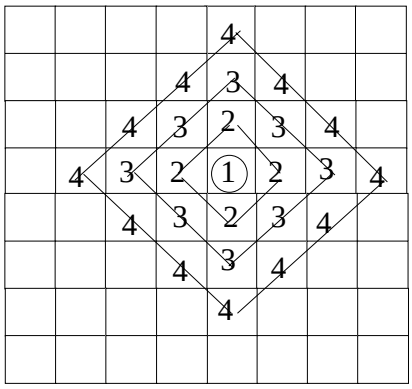
Winkelhalbierende: $y = tg(\ \alpha_2 + \alpha_1)/2\ \ (x - x_s) + y_s$

6.1.2.3. Navigation auf Rasterkarte

Finden des kürzesten Weges: vom Start ausgehend Welle über das Raster laufen lassen.





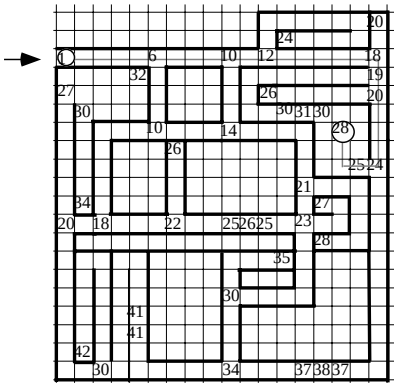


die Welle läuft um Hindernisse herum

vom Ziel her absteigende Folge von Wegpunkten kürzester Weg

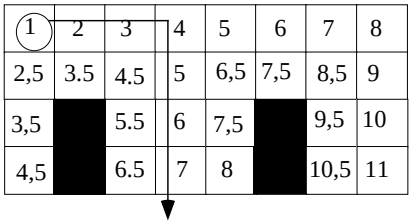
Lösung eines Labyrinths

mit Finden des kürzesten Weges



Beim Fluten dominiert die kleinere Zahl

Bevorzugen von Wegen entfernt von Hindernissen



neben Hindernissen wird die Welle höher

(hier um 0,5 erhöht)

==> Weg wird in den Raum

zwischen Hindernissen getrieben

Algorithmus zur Berechnung der Entfernungswellen im Raster

Zu Anfang seien die Zellen der Rasterkarte entweder als Freiraum (= 0) oder als Hindernis (= -1) gekennzeichnet. Der Algorithmus erfasst nacheinander alle Zellen mit z = 0, ausgehend vom Startpunkt S. Es müssen dabei für den jeweils nächsten Schritt nur die Zellen an der Wellenfront betrachtet werden.

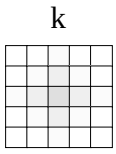
Fluten der Rasterkarte mit Entfernungswerten:
zwei Stacks Stack S_1 und S_0 werden abwechselnd gefüllt und geleert

Initialisierung:
Freiraumzellen z = 0; Hinderniszellen z = -1; Stack $S_1 = S_0$ = leer

```
begin
    i := 1;  j := 0;
    Startzelle z :=1; push S0;
    repeat
        i := (i + 1) mod 2;  j := (j + 1) mod 2
        repeat
            pop Stack Si;   untersuche Umgebung der Zelle;
            für alle Zellen mit z = 0      falls min (O, S, W, N) = m > 0
                                                z>0
                                                ==>  z := m+1; push Stack Sj;
                                           sonst tue nichts;
        until Stack Si = leer;
    until Stack Sj = leer;
end.
```

ntersuche Umgebung der Zelle:

sei die Zelle $z_{i,k}$, dann ist ihre Umgebung $\begin{cases} O: z_{i,k+1} \\ S: z_{i+1,k} \\ W: z_{i,k-1} \\ N: z_{i-1,k} \end{cases}$



$z_{i,k+1} = 0$ ==> Umgebung der Zelle $\begin{cases} O: z_{i,k+2} \\ S: z_{i-1,k+1} \\ W: z_{i,k} \\ N: z_{i-1,k+1} \end{cases}$ falls $\min_{z>0}(O, S, W, N) = m > 0$ ==> $z_{i,k+1} := m+1$; push Stack S_j;

$z_{i+1,k} = 0$ ==> Umgebung der Zelle $\begin{cases} O: z_{i+1,k+1} \\ S: z_{i+2,k} \\ W: z_{i+1,k-1} \\ N: z_{i,k} \end{cases}$ falls $\min_{z>0}(O, S, W, N) = m > 0$ ==> $z_{i+1,k} := m+1$; push Stack S_j;

$z_{i,k-1} = 0$ ==> Umgebung der Zelle $\begin{cases} O: z_{i,k} \\ S: z_{i+1,k-1} \\ W: z_{i,k-2} \\ N: z_{i1,k-1} \end{cases}$ falls $\min_{z>0}(O, S, W, N) = m > 0$ ==> $z_{i,k-1} := m+1$; push Stack S_j;

$z_{i-1,k} = 0$ ==> Umgebung der Zelle $\begin{cases} O: z_{i-1,k+1} \\ S: z_{i,k} \\ W: z_{i-1,k-1} \\ N: z_{i-2,k} \end{cases}$ falls $\min_{z>0}(O, S, W, N) = m > 0$ ==> $z_{i-1,k} := m+1$; push Stack S_j;

6.1.2.4. Quadtree-basierte Wegeplanung

- Einstufige Planung

  Voraussetzung:
  Karte in Quadtree-Darstellung mit weißen und schwarzen Knoten.

  Planung:
  A*-Algorithmus mit …
  - Expansion eines Knotens: suche Nachbarknoten mit freier Fläche.
  - Expansion eines Knotens im nächsten Schritt:
    Expansion von vertikalen oder horizontalen hindernisfreien Nachbarknoten.
    Jeweils der Knoten mit minimalem Wert der Gewichtsfunktion wird expandiert.

  Aufbau der Gewichtsfunktion f:

C:	hindernisfreier Knoten
P:	Vorgänger von C auf diesem Pfad
g(P):	Kosten des Weges von S nach P
d(P, C):	aktuelle Entfernung von P nach C
O(C):	Entfernung von C zum nächsten Hindernis
O_{max}:	max{ O(C) | ∀ C des Quadtrees}
α:	Konstante, die den Mindestabstand zu Hindernissen bestimmt
h(C):	optimistisch abgeschätzte Entfernung zwischen C und Ziel Z

  $$\mathbf{f(C) = g(P) + d(P, C) + \alpha \cdot (O_{max} - O(C)) + h(C)}$$

  Resultat:
  Eine Liste von hindernisfreien Knoten variierender Größe.

  Vorteile der einstufigen Wegeplanung:
  - Die Berechnung des Weges ist effizient.
  - α legt die Distanz zu Hindernissen fest.
  - Kein großer Overhead bei Quadtree-Repräsentation.

- Zweistufige Wegeplanung

  Voraussetzung:
  Karte in Quadtree-Darstellung mit weißen, grauen und schwarzen Knoten.

  Vorgehensweise:
  - Der Pfad wird auf möglichst grobem Level geplant und erst in zweiter Stufe innerhalb der grauen Knoten weiterentwickelt.
  - Graue Knoten werden nur soweit als unbedingt notwendig verfeinert.
  - Kleinste Quadrate ≈ Fahrzeugabmessungen.

  Vorteile der zweistufigen Wegeplanung:
  - der Algorithmus ist kostengünstig, wenn viele kleine Hindernisse nur wenige große weiße Knoten zulassen
  - unbekannte Regionen lassen sich elegant handhaben als graue Knoten mit hohen Kosten

6.1.2.5. Flächendeckendes Fahren

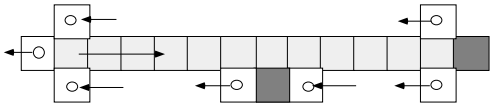
egeben: eine Rasterkarte

der AMR füllt (mit seiner Bürste) das Raster aus und kann im Raster abbiegen

esucht: ein Weg, der über alle freien Rasterpunkte führt

orgehensweise:

- versuche gerade Bahnen zu fahren;
- markiere und merke Anfangs- und Endpunkte von noch zu fahrenden Bahnen parallel zur gerade gefahrenen Bahn
- markiere gefahrene Wege
- am Ende einer Bahn (Hindernis oder markierter Punkt)

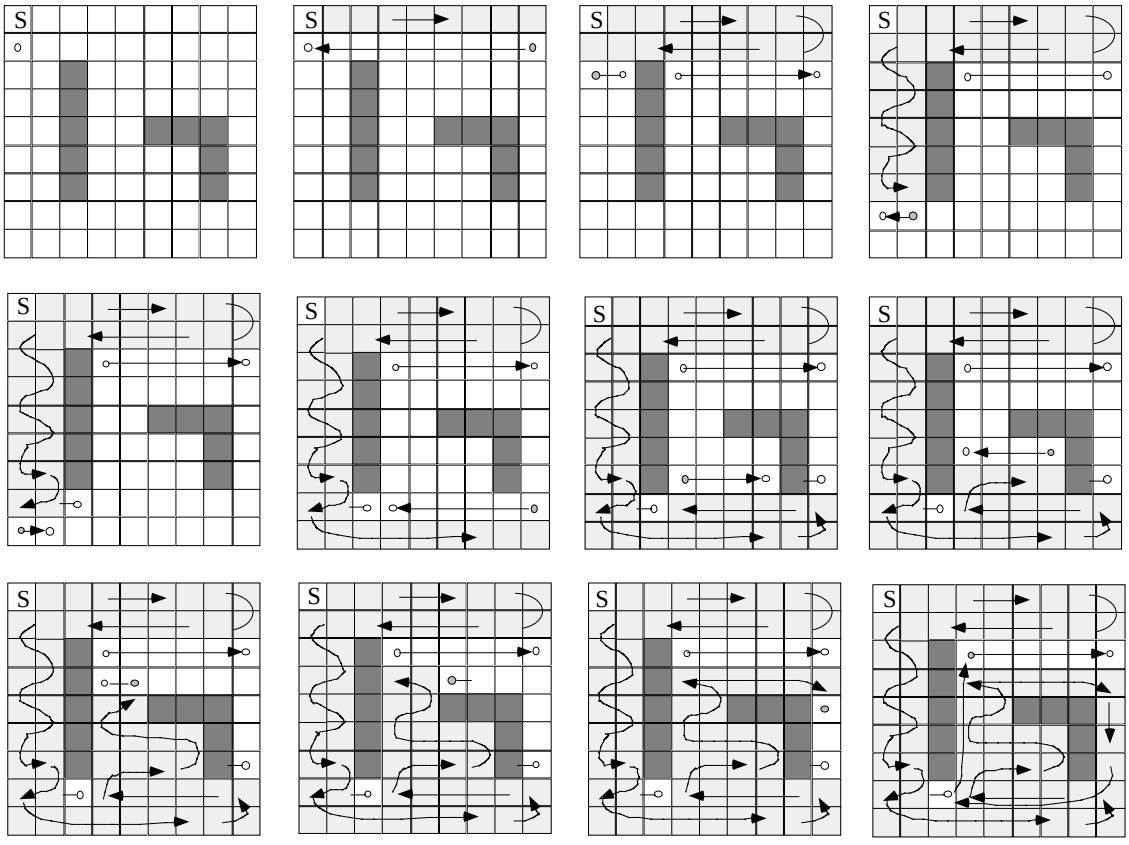


solange Stack nicht leer:

fahre zum Anfangspunkt der auf dem Stack gemerkten Bahn;

sonst: fertig

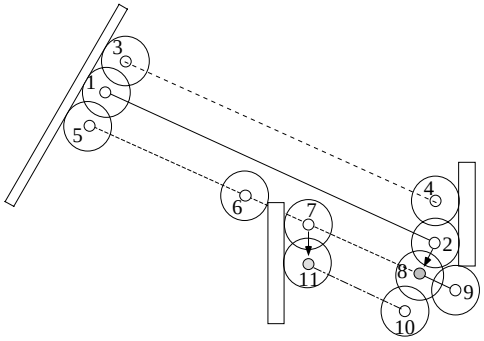
Nachfolgend wird das flächendeckende Fahren an einem Beispiel gezeigt. Gegeben ist eine gerasterte Hinderniskarte und ein Startpunkt und eine Startrichtung.



6.1.2.6. Flächendeckendes Fahren in einer Polygonkarte

Es werden parallele Bahnen zur Hauptrichtung angelegt:

- Analysiere vorherrschende Richtung in der Karte.
- Beginne an einer Wand in Freiraum hinein in dieser Richtung Weg zu planen.
- Lege Kursendpunkte rechts und links parallel der Bahn an und bringe sie auf den Stack.
- Beim Auftreten von Hindernissen rechts oder links der Bahn lege Kursanfangspunkte an.
- Beim erneuten Erreichen von Freiraum lege Kursendpunkte an; bringe auf den Stack.
- Beim Treffen der Bahn auf eine Wand lege Kursanfangspunkte an und bringe auf Stack.
- Wenn Stack nicht leer
  hole Kursanfangs- und endpunkt vom Stack;
  plane Weg dorthin durch gereinigte Fläche;
  verfolge die vom Stack geholte Bahn; weiter bei *
- sonst fertig.

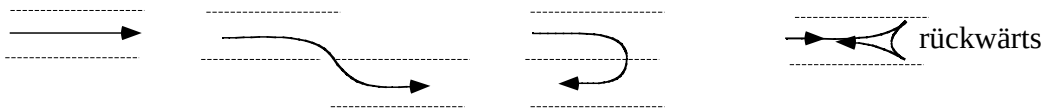


Überlappung der Bahnen

für flächendeckendes Fahren

Abstand <Durchmesser des AMR

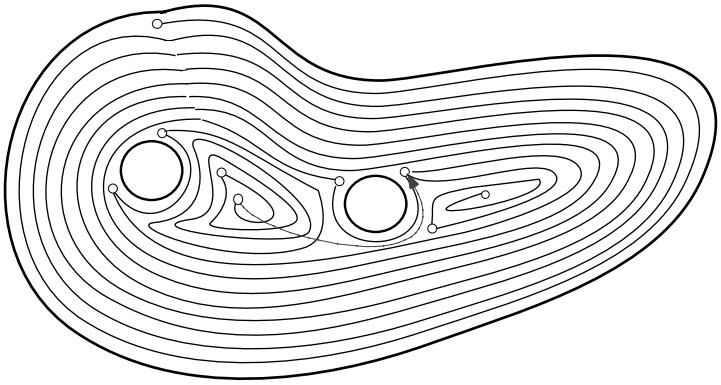
elementare Fertigkeiten: (Hefner, 1996)



geradeaus fahren | Fahrbahnwechsel | Kehrtwende (U-turn) | Kehrtwende mit Rückstoßen

6.1.2.7. Flächendeckendes Fahren mit "wall following"

- Lege parallele Bahnen in einer Spirale an die Wand und schon gefahrene Strecken.
- Markiere Eckpunkte, wo die Fahrt ohne Richtungsänderung nicht fortgesetzt werden kann.
- Wird ein Eckpunkt erreicht, suche Eckpunkte die an Freifläche grenzen.
- Fahre durch schon befahrenes Gebiet dorthin.
- Setze Wandfolgen fort, bis keine Eckpunkte mehr an Freiflächen grenzen.



6.1.3. Exploration

Ziel der Navigation ist ein Weg, der alle Teile eines Raumes erfasst.

Der AMR habe einen Sensor mittlerer Reichweite d mit $r << d < R$

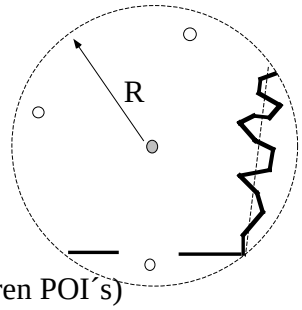
r = Robotradius; R typ. Entfernung in der Umgebung

∗ - Inspiziere Karte vom momentanen Standpunkt aus

- suche Durchfahrten (Breite ≥ 3r , dahinter freier Raum)
- suche "Wände"(vom Roboter einsehbar, ohne Durchfahrten)
- lege Anfahrpunkte (points of interest, POI)

  in Entfernung ≈ 4/5 R vom Robot

  - in die Mitte von Durchfahrten
  - oder 2/3 R vor eine "Wand"
  - oder in den freien Raum (90° zu anderen POI´s)



- wenn innerhalb der Sensoreichweite ein POI liegt,

  der nicht als schon "befahren" markiert ist, lösche ihn
- lege alle POI´s auf den Stack,
- wenn Stack ist nicht leer,

  hole POI vom Stack; markiere ihn als "befahren"

  fahre durch exploriertes Gelände zu diesem POI

  weiter bei ∗
- sonst: fertig

- wenn innerhalb der Sensoreichweite ein POI liegt,

  der nicht als schon "befahren" markiert ist, lösche ihn

- lege alle POI´s auf den Stack,

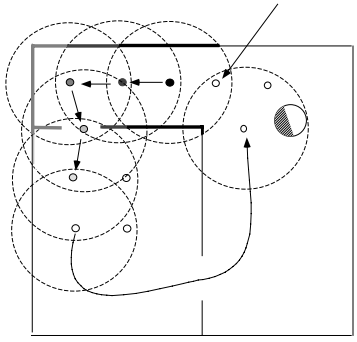
- wenn Stack ist nicht leer,

  hole POI vom Stack;

  fahre durch exploriertes Gelände zu diesem POI;

  markiere ihn als "befahren";

  weiter bei ∗;



- sonst: fertig

der gesamte Raum ist exploriert

6.2. Bahnplanung

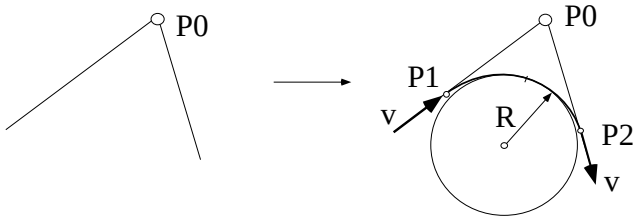
Gegeben:
Folge von Kurspunkten mit geraden Strecken dazwischen

Gesucht:
eine mit den Beschränkungen des AMR befahrbare Bahn

- kinematische Beschränkungen:
  bei gegebener Fahrzeuggeschwindigkeit v minimal fahrbarer Kurvenradius R
- geometrische Beschränkungen:
  Fahrzeugabmessungen erfordern freien Weg zu beiden Seiten der Bahn

6.2.1. Kinematische Beschränkungen bei Kurvenfahrt

Übergang:

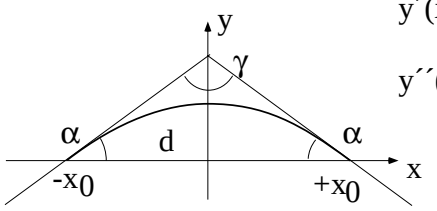


- Geschwindigkeit v bestimmt den Radius R des minimalen Krümmungskreises

  durch die tolerierbaren Querkräfte im Scheitelpunkt der Bahn $K_{quer} \sim \omega^2$; $\omega = v/R$

- Übergang von der Geraden zur Kurvenbahn krümmungsstetig:

  nach Transformation in ein angepaßtes Koordinatensystem



$y'(x_0) = -\operatorname{tg}\alpha; \qquad y'(-x_0) = \operatorname{tg}\alpha \qquad y'(0) = 0$

$y''(x_0) = y''(-x_0) = 0$

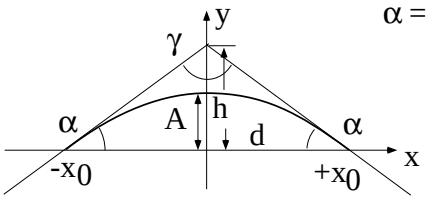
Krümmung K im Scheitelpunkt ist

$$K = \frac{y''(x)}{\{1 + y'^2(x)\}^{3/2}}\Bigg|_{x=0} = y''(0) \text{ mit } |K(0)| = \frac{1}{R}$$

Es ist relativ einfach eine krümmungsstetige Funktion zu finden, sei es eine trigonometrische Funktion, ein Polynom oder eine Spline. Schwierig ist die Integration der Funktion zur Bestimung des Weges aus v und ω.

6.2.1.1. Lösung mit trigonometrischen Funktionen

Es soll eine Funktion gefunden werden, die eine symmetrische Bahn beschreibt, die bei $(-x_0, 0)$ unter einem Winkel α beginnt und bei $(x_0, 0)$ unter dem Winkel -α endet.



$\alpha = (180° - \gamma)/2$

$$y(x) = A \cdot \cos \frac{\pi}{2x_0} \cdot x$$

$$y'(x) = - A \cdot \frac{\pi}{2x_0} \cdot \sin \frac{\pi}{2x_0} \cdot x$$

$$y''(x) = - A \cdot (\frac{\pi}{2x_0})^2 \cos \frac{\pi}{2x_0} \cdot x$$

$y(x_0) = y(-x_0) = 0$

$$y'(x0) = - A \frac{\pi}{2x_0} \cdot 1 = - \text{tg}\,\alpha \implies A = 2x_0 / \pi \cdot \text{tg}\,\alpha$$

$$y''(0) = - A \cdot (\frac{\pi}{2x_0})^2 \implies R = \frac{4\,x_0^2}{\pi^2 A} = \frac{2x_0}{\pi\,\text{tg}\,\alpha} \implies x_0 = \pi/2\ R\ \text{tg}\,\alpha$$

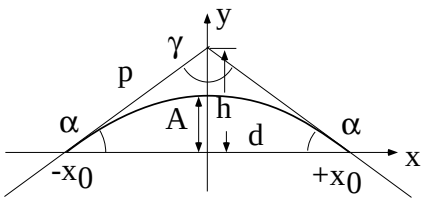
vorgegeben: $A = R\ \text{tg}^2\alpha$

von der Wegeplanung : α; durch kinemat. Beschränkung: max. Querkraft ==> R

$$\implies x_0 = \frac{\pi\ R\ \text{tg}\,\alpha}{2} \qquad h = x_0 \cdot \text{tg}\,\alpha$$

Abweichung Scheitelpunkt - Schnittpunkt $h - A = x_0 \cdot \text{tg}\,\alpha\,(1 - 2/\pi) = R\ \text{tg}^2\alpha\,(\pi/2 - 1)$

6.2.1.2. Lösung durch Polynom vierten Grades



(1) $y(x) = ax^4 + bx^3 + cx^2 + dx + e$

(2) $y'(x) = 4ax^3 + 3bx^2 + 2cx + d$

(3) $y''(x) = 12ax^2 + 6bx + 2c$

(4) $y(x0) = y(-x0) = 0$ ==> symmetrische Funktion ==> $b = d = 0$

(5) $y(x_0) = 0 \implies ax_0^4 + cx_0^2 + e = 0$

(6) $y(0) = A \implies e = A$

(7) $y'(-x_0) = -4a\,x_0^3 - 2cx_0 = \text{tg}\,\alpha$

(8) $y''(0) = 2c = -1/R \implies c = -1/(2R)$;

(9) $y''(x_0) = 12ax_0^2 + 2c = 0 \implies a = - c/(6x_0^2) = \frac{1}{12\ R\ x_0^2}$

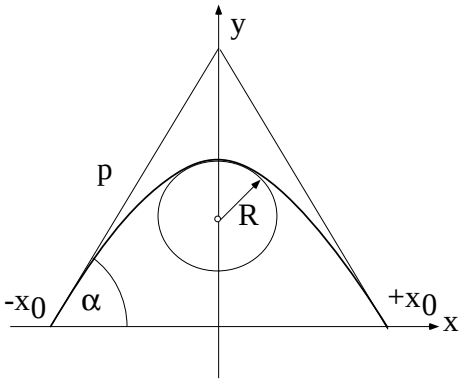
(7) $\implies -4a\,x_0^3 - 2cx_0 = \text{tg}\,\alpha \implies x_0(\frac{1}{R} - \frac{1}{3R}) = \text{tg}\,\alpha \implies x_0 = 3R/2 \cdot \text{tg}\,\alpha$

(5) ==> $(3R/2 \cdot \mathrm{tg}\,\alpha)\,2/(12\,R) - 1/(2R)\,(3R/2 \cdot \mathrm{tg}\,\alpha)^2 + e = 0$

$e = 9/4\,R^2\,\mathrm{tg}^2\alpha\,(1/2R - 1/12R)$

$e = 15/16\,R\,\mathrm{tg}^2\alpha$ $\qquad x_0 = 3R/2 \cdot \mathrm{tg}\,\alpha$ $\qquad p = x_0/\cos\alpha$

Polynom 4. Grades $y = -\dfrac{1}{27\,R^3\,\mathrm{tg}^2\alpha}\,x^4 + \dfrac{1}{2R}\,x^2 + 15/16\,R\,\mathrm{tg}^2\alpha$



Abstand Scheitelpunkt - Schnittpunkt

$h - A = 3R/2\,\mathrm{tg}^2\alpha - 15/16\,R\,\mathrm{tg}^2\alpha$

$h - A = 9/16\,R\,\mathrm{tg}^2\alpha$ (0,56) $\qquad p = \dfrac{3R\,\mathrm{tg}\,\alpha}{2\cos\alpha}$

bei trigonometr. Funktion $h - A = (\pi/2 - 1)\,R\,\mathrm{tg}^2 a$ (0,57) $\qquad p = \dfrac{\pi\,R\,\mathrm{tg}\,\alpha}{2\cos\alpha}$

6.2.1.3. Ansteuern der Bahn

- Folge $\omega_i = \omega(t_i)$ mit $t_i = t_0 + i\Delta t$ ist zu erzeugen,

  damit mit v= const. der AMR die Kurve durchfährt

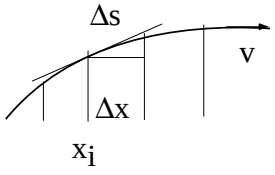
- über Krümmung $K(x) = \dfrac{y''(x)}{(1 + y'^2(x))^{3/2}}$ bestimmt sich $\omega(x) = v\,K(x)$

- zur Bestimmung von ω_i ist die Krümmung K_i an Stellen x_i zu bestimmen,

  die jeweils um die Strecke Δs, durchfahren in kostanter Zeit Δt, entfernt sind

- die Strecken Δs bestimmen sich aus $\Delta s = v\,\Delta t = (1 + y'^2(x))^{1/2}\,\Delta x$

==> $x_{i+1} = x_i + \dfrac{v\,\Delta t}{(1 + y'^2(x))^{1/2}}$



Es ist die Krümmung der Bahn für die Berechnung der Bahnpunkte x_i zu bestimmen.

Berechnung der Krümmung für die trigonometrische Funktion

$$(x) = \frac{y''(x)}{(1 + y'^2(x))^{3/2}} \qquad y(x) = A \cdot \cos \frac{\pi}{2x_0} \cdot x \qquad x_0 = \pi/2\ R\ tg\ \alpha \qquad A = R\ tg^2\alpha$$

$$y(x) = R\ tg^2\alpha\ \cos x/(R\ tg\ \alpha)$$

$$y'(x) = -\ tg\ \alpha\ \sin x/(R\ tg\ \alpha)$$

$$y''(x) = -\ 1/R\ \cos x/(R\ tg\ \alpha)$$

$$K(x) = \frac{-\ 1/R\ \cos x/(R\ tg\ \alpha)}{(1 + tg^2\alpha\ \sin^2 x/(R\ tg\ \alpha))^{3/2}}$$

für gegebenes R und α sind die Kurve 4. Grades

und die Cosinusfunktion sich recht ähnlich insbesondere in p: 1.5 vs 1,57 R $\frac{tg\ \alpha}{\cos\ \alpha}$

6.2.1.4. Berechnung der Bogenlänge der Bahn von $-x_0$ bis zum Scheitelpunkt

$$s = \int_{-x_0}^{0} (1 + y'^2(x))^{1/2}\ dx$$

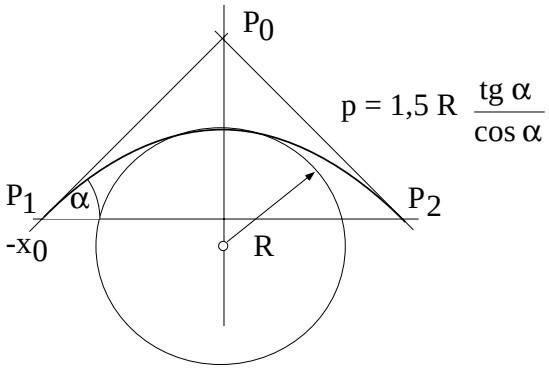
i. allg. nicht geschlossen integrierbar

damit ist die Zeit T unbestimmt,

wann der Scheitelpunkt erreicht wird

==> numerische Integration der Bogenlänge

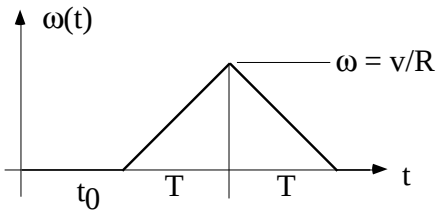
der trigonometr. Funktion ==> s = T v



6.2.1.5. Klothoidenbahn

Vorgehen zum Abfahren der Bahn:

- berechne die Bogenlänge s der trigonometr. Bahn und T = s/v
- bremse v1 ab auf v bis P1; Zeit t = t0
- erzeuge für Zeit T ein $\omega(t) = a\ (t - t0)$ mit $a = v / (R \cdot T)$; bei $t = t0 + T$ ist $\omega(t) = v/R$
- lasse für T $\omega(t)$ abnehmen von v / R nach Null, dann ist der AMR bei P2;
- beschleunige auf v2



$$\varphi(t) = \int a(t-t0)\,dt = (1/2\ a\ t^2 - a\ t_0\ t)$$

$$x(t) = a \int v \cos(1/2\ t^2 - t_0\ t)\,dt$$

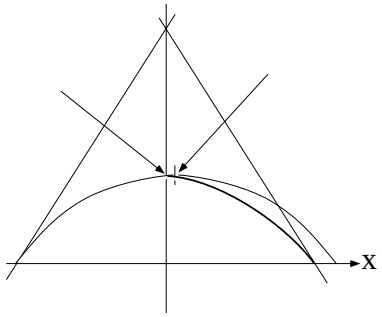
$$y(t) = a \int v \sin(1/2\ t^2 - t_0\ t)\,dt$$

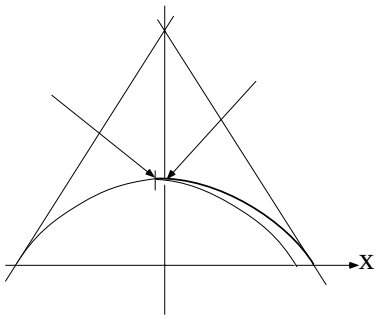
} Klothoide

nicht geschlossen integrierbar

$$\omega(t) = \begin{cases} \frac{v}{R\,T}\,(t - t_0) & t_0 \le t \le t_0 + T \\ v/R - \frac{v}{\quad}\,(t - t_0 - T) & t_0 + T \le t \le t_0 + 2T \end{cases}$$

6.2.1.6. Bahnkorrektur





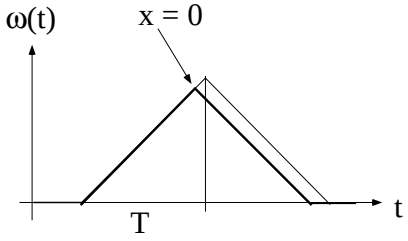
wenn $x = 0$ bei $t - t_0 < T$

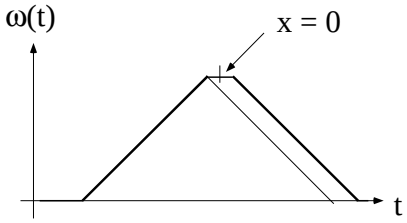
lasse $\omega(t)$ rückwärts laufen

wenn $x < 0$ bei $t = t0 + T$

stoppe Anstieg von $\omega(t)$ bis $t = t_0 + T$

lasse $\omega(t)$ rückwärts laufen





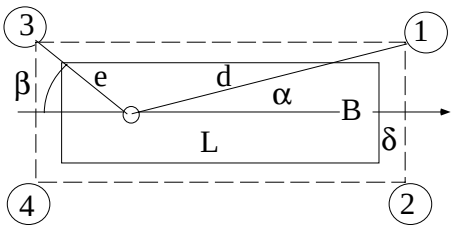
6.2.2. Bahnplanung unter geometrischen Einschränkungen

Die Bahnplanung muss die gegebenen Abmessungen des AMR bei der Planung berücksichtigen. Sie haben Einfluß auf die Gestalt des Weges, der für den AMR freizuhalten ist.

6.2.2.1. Runder AMR

Planung durch Aufblähen der Hindernisse um $(r + \delta)$ mit r = Radius des AMR und Sicherheitsabstand δ und Planung für einen punktförmigen Robot.

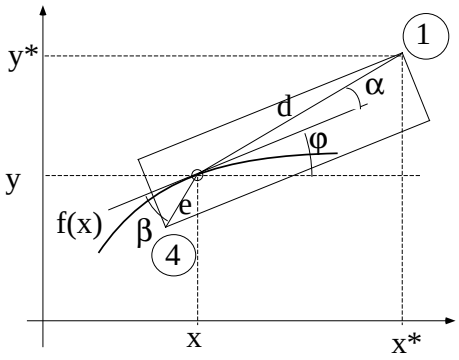
6.2.2.2. Rechteckiger AMR, Länge L, Breite B mit Sicherheitsabstand δ



- Fahrt auf gerader Strecke:
  überprüfe auf Hindernisse innerhalb einer Bahn der Breite B+δ
- Fahrt auf einer Kurve y = f(x):

  überprüfe auf Hindernisfreiheit auf einem Weg (x*, y*)

  begrenzt durch das Ausschwenken des AMR



Eckpunkt 1 läuft auf einer Kurve

mit $\varphi = \arctan f'(x)$

$x_1^* = x + d \cos(\alpha+\varphi)$

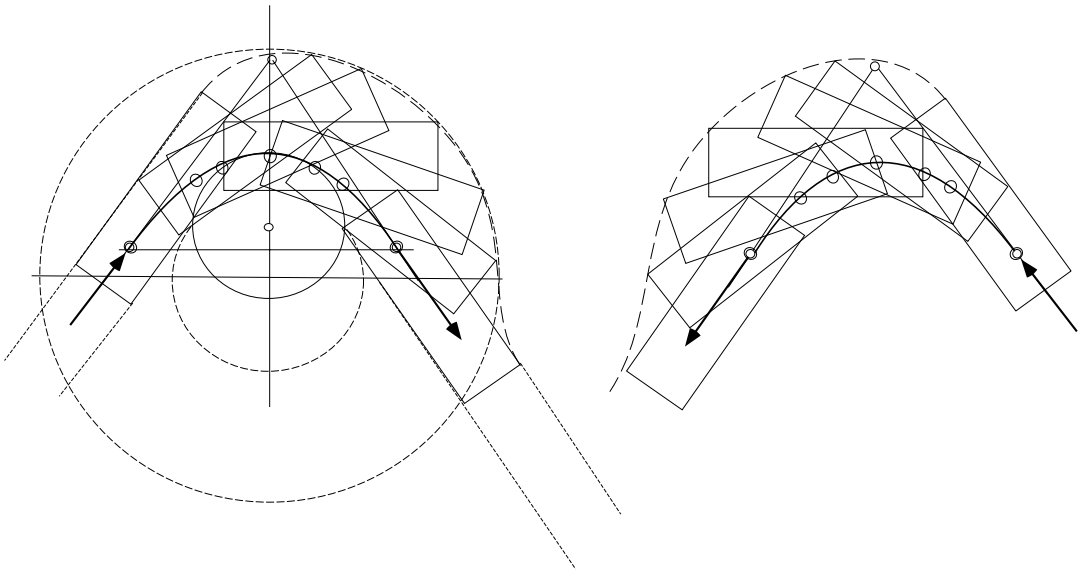
$y_1^* = y + d \sin(\alpha+\varphi)$

und Eckpunkt 4 auf einer Kurve

$x_4^* = x - e \cos(90° - \beta - \varphi)$

$y_4^* = y - e \sin(90° - \beta - \varphi)$

Beispiel für den Weg, den ein rechteckiger AMR braucht:



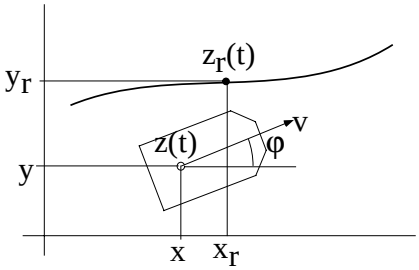
6.3. Bahnregelung

Es ist eine Bahn $z_r(t)$ vorgegeben (Referenzbahn);

auf ihr läuft mit Sollgeschwindigkeit ein Referenzpunkt zur Zeit t bei (x_r, y_r)

Der Roboter befindet sich i.allg. nicht perfekt auf der Referenzbahn

zum Zeitpunkt t bei $(x, y, \varphi) = z(t)$



Regelungsaufgabe:

bestimme Steuerungsfunktionen

$v^*(t)$ und $\omega^*(t)$ so, dass der AMR

aus beliebigem Anfangszustand $(z(t_0), z'(t_0))$

in die Referenzbahn übergeht:

$$\lim_{z \to \infty} z(t) = z_r(t)$$

Es ist $x' = v \cos\varphi$ $\quad x(t_0) = x_0$ $\qquad \varphi' = \omega$ $\quad \varphi(t_0) = \varphi_0$

$y' = v \sin\varphi$ $\quad y(t_0) = y_0$

$x'' = v' \cos\varphi - v\,\omega \sin\varphi$ $\qquad x'(t_0) = x_0'$

$y'' = v' \sin\varphi + v\,\omega \cos\varphi$ $\qquad y'(t_0) = y_0'$

Der Fehlervektor ist $z_e = \begin{pmatrix} x_e \\ y_e \end{pmatrix} = \begin{pmatrix} x_r - x \\ y_r - y \end{pmatrix}$

eschwindigkeit

$x_e' = x_r' - v \cos\varphi$ $\qquad x_e(t_0) = x_{e0} = x_{r0} - x_0$

$y_e' = x_r' - v \sin\varphi$ $\qquad y_e(t_0) = y_{e0} = y_{r0} - y_0$

Beschleunigung

$x_e'' = x_r'' - (v' \cos\varphi - v\,\omega \sin\varphi)$ $\qquad x_e'(t_0) = x_{e0}'$

$y_e'' = y_r'' - (v' \sin\varphi + v\,\omega \cos\varphi)$ $\qquad y_e'(t_0) = y_{e0}'$

6.3.1. Berechnung der Steuerungsfunktionen v(t) und ω(t)

Es laufe auf der Bahn der Referenzpunkt $(x_r(t), y_r(t))$ mit einer Geschwindigkeit $v < v_{max}$. Dann kann, auch bei Abweichungen des AMR von der Sollbahn, er den Referenzpunkt wieder erreichen, indem er beschleunigt. Der AMR läuft auf der aktuellen Bahn (x(t), y(t)).

aus $x'' = v' \cos\varphi - v\,\omega \sin\varphi$

und $y'' = v' \sin\varphi + v\,\omega \cos\varphi$ ==> $x'' \cos\varphi + y'' \sin\varphi = v'(t)$

Integration von $v'(t)$ ==> $v(t) = v(t_0) + \int_{t_0}^{t} \{x''(t) \cos\varphi(t) + y''(t) \sin\varphi(t)\} dt$

==> $(y'' \cos\varphi - x'' \sin\varphi) = v\,\omega$

==> $\omega(t) = (y''(t) \cos\varphi(t) - x''(t) \sin\varphi(t)) \frac{1}{v(t)}$

Es soll nun aus dem bekannten Fehlervektor und seinen Ableitungen ein v und ω bestimmt werden, das den Fehler zu Null werden lässt. Das ist äquivalent der

uche nach einem stabilen Gleichgewicht: $(z_e, z_e') = 0$

. h. der Fehlervektor wird und bleibt Null

ntspricht: suche nach einer **Ljapunov-Funktion** $V(z_e, z_e')$

it 1) $V(z_e, z_e') = 0$ für $[z_e, z_e'] = 0$

2) $V(z_e, z_e') > 0$ für $[z_e, z_e'] \neq 0$

3) $V'(z_e, z_e') \leq 0$ für alle $[z_e, z_e']$

xistiert eine solche Funktion

=> (z_e, z_e') ist asymptotisch stabil,

d. h. für jeden Anfangspunkt (z_0, z_0') und jede Kurve s mit (x_r, y_r, x_r', y_r')

kann der Referenzpunkt erreicht werden: $\lim_{t \to \infty} z(t) = z_r(t)$

usatzforderung kann mit erfüllt werden:

das Funktional $J(v(t), \omega(t), t) = \int_{t_0}^{t} (z_e^T z_e + z_e'^T R_1 z_e' + z_e''^T R_2 z_e'') dt \overset{!}{=} \text{Min}$

R_1 und R_2 sind positiv definit und symmetrisch

(gewichtete Summe des quadratischen Fehlervektors und seiner Ableitungen

minimal ==> glatte Kurve)

6.3.1.1. Angabe der Ljapunov-Funktion

sei $z'' = z_r'' + Q_1 z_e' + Q_0 z_e$ (Lösung dieser Differentialgleichung)

it $Q_1 = \begin{pmatrix} q_{1x} & 0 \\ 0 & q_{1y} \end{pmatrix}$ und $Q_0 = \begin{pmatrix} q_{0x} & 0 \\ 0 & q_{0y} \end{pmatrix}$ und $q_{1x}, q_{1y} > 0; \quad q_{0x}, q_{0y} > 0$

==> (z_e, z_e') ist stabiler Gleichgewichtszustand

Beweis über die Konstruktion einer Ljapunov-Funktion:

$V(z_e, z_e') = 1/2(z_e^T Q_0 z_e + z_e'^T z_e)$ (Ansatz als quadratische Form)

1) $[z_e, z_e'] = 0$ ==> $V(z_e, z_e') = 0$

2) $V(z_e, z_e') = 1/2 \{ (q_{0x} x_e^2 + q_{0y} y_e^2) + (x_e'^2 + y_e'^2)\}$

$V(z_e, z_e') > 0$ für alle $[z_e, z_e'] \neq 0$ wegen $q_{0x}, q_{0y} > 0$

3) $V'(z_e, z_e') = z_e'^T Q_0 z_e + z_e'^T z_e''$

wegen $z_e'' + Q_1 z_e' + Q_0 z_e = 0$ ist

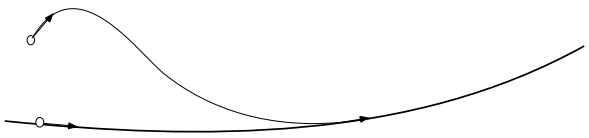
$V'(z_e, z_e') = z_e'^T Q_0 z_e + z_e'^T (- Q_1 z_e'' - Q_0 z_e) = - z_e'^T Q_1 z_e'$

wegen q_{1x} und $q_{1y} > 0$ ==>

$V'(z_e, z_e') = - z_e'^T Q_1 z_e' \leq 0$ für alle $[z_e, z_e']$

==> $V(z_e, z_e') = 1/2(z_e^T Q_0 z_e + z_e'^T Q_1 z_e')$ ist eine Ljapunov-Funktion

==> (z_e, z_e') ist global asymptotisch stabil



Unabhängig von der Anfangsposition des AMR kann er die Sollbahn erreichen, wenn die Bahn so geführt wird, dass die Differentialgleichung

$z'' = z_r'' + Q_1 z_e' + Q_2 z_e$ erfüllt wird.

Das ist die Lösung des Paares von Differentialgleichungen

$x'' = x_r'' + q_{1x} (x_r' - x') + q_{0x} (x_r - x)$ mit $q_{0x}, q_{1x} > 0$
$y'' = y_r'' + q_{1y} (y_r' - y') + q_{0y} (y_r - y)$ mit $q_{0y}, q_{1y} > 0$

6.3.1.2. Bestimmung von x´´(t) und y´´(t) aus xr´´, yr´´und ze´und ze

sei v_{max} = max. Wert der Steuerungsfunktion

und v'_{max} = max. Wert der Geschwindigkeit der Steuerungsfunktion

wähle $q_{0x} = q_{0y} = 0{,}01 \left(\frac{v'_{max}}{v_{max}} \right)^2$

und $q1x = q1y = 2\zeta\, 0{,}1 \frac{v'_{max}}{v_{max}}$ mit $\zeta = 0{,}7 \ldots 1$

Die Differentialgleichungen müssen nicht explizit gelöst werden, sondern y´´(t) und x´´(t) werden direkt zu den Steuerungsfunktionen v(t) und ω(t) integriert.

6.3.2. Beweis des Satzes von Ljapunov

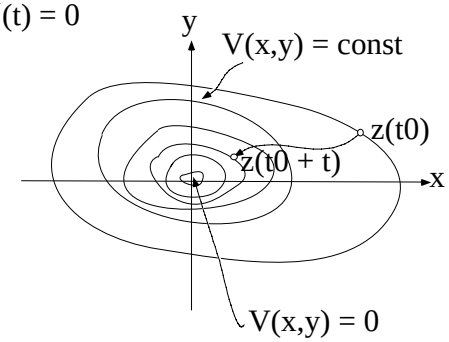
Gegeben sei eine Funktion V(z,z´). Der Parameter z(t) sei eine Bahn in einem mehrdimensionalen Raum; eine Vektorfunktion.

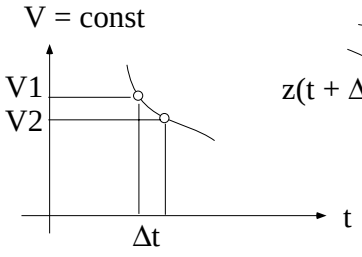
Es erfülle die Funktion V(z,z´) die o. g. Bedingungen:

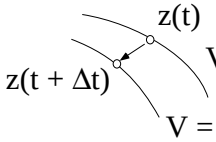
V(z, z´) = 0 für (z,z´) = 0; V(z,z´) > 0 für (z, z´) ≠ 0; V´(z, z´) ≤ 0 für alle (z, z`)

==> (z,z´) asymptotisch stabil: $\lim_{t \to \infty} z(t) = 0$; $\lim_{t \to \infty} z'(t) = 0$

(z, z´) = V(x(t), y(t), x´(t), y´(t)) = V(x, y; t)







der Punkt z(t) bewegt sich

in Richtung V(x, y) = 0

==> (z, z´) = 0

V´(t) ≤ 0 ==> V2 ≤ V1 ==> $\lim_{t \to \infty} (x(t), y(t)) = 0$

$\lim_{t \to \infty} (x'(t), y'(t)) = 0$