Seminar Sommersemester 2016:
Automobile Systeme in der Automatisierung
Prof. Dr. Dieter Zöbel, Universität Koblenz-Landau, FB Informatik

Übersicht über die Pfadplanung

Benjamin Lange

Eingereicht: 23.07.2015 / Fertiggestellt: 15.07.2015

Zusammenfassung Die vorliegende Arbeit vermittelt einen Überblick über den Bereich der Pfadplanung. Es werden grundlegende Begriffe zum Thema eingeführt. Weiterhin wird modellhaft eine Vorgehensweise vorgestellt, um Pfade für automobilen Systemen zu planen. Diese gliedert sich in Methoden zur Repräsentation der Hindernisse und den eigentlichen Verfahren zur Pfadplanung.

Schlüsselwörter Pfadplanung, Roboter Bewegung, Potentialfeld, A*, Probabilistic Roadmap

1 Einleitung

In der heutigen Zeit konstruieren Menschen immer intelligentere Maschinen, die in unserer komplexen Gegenwart selbständig eine Vielzahl von Aufgaben übernehmen. Die Maschinen zu Beginn der Industrialisierung konnten für kaum mehr als simple mechanische Arbeiten eingesetzt werden. Wohingegen die Roboter heutiger Zeit durch Aufnahme von Informationen über Sensoren aus der Umwelt und der eigenständigen Auswertung dieser Daten, ein großes Maß an Autonomie dazugewonnen haben. Die Autonomie und Mobilität heutiger Roboter erlaubt es dem Menschen, Roboter zur Erkundung für schwer zugängliche Räume wie die Tiefsee (ABYSS) oder den Weltraum (MARS ROVER) zu benutzten. Es werden aber auch wichtige Gebiete wie die Medizin mit Assistenzsystemen ausgestattet, die Ärzte bei ihren komplexen Aufgaben unterstützten (CYBERKNIFE). Die Methoden der Pfadplanung beschränken sich in ihrer Anwendbarkeit nicht nur auf diese brisanten Felder, sondern sie werden auch in der biologischen Forschung eingesetzt (Bindungs- und Faltungsverhalten von Molekülen) oder in Computerspielen (Künstliche Intelligenz). Die Einsatzgebiete von Pfadplanungsalgorithmen vergrößern sich stetig und Wissenschaftler ersinnen immer robustere und schnellere Rechenvorschriften. Hierbei sind zentrale Themen wie Kollisionsvermeidung, Hindernisvermeidung von Interesse. Weitere wichtige Aspekte sind die benötigte Planungs- bzw. Ausführungszeit oder der Abdeckungsgrad [1]. Im Folgenden

wird auf die elementaren Begriffe der Pfadplanung eingegangen, die dazu verwendet werden, die Themen Hindernisrepräsentation, Kollisionsvermeidung und globale Pfadplanung tiefergehend zu beschreiben.

2 Einführung wichtiger Begriffe

Dieser Abschnitt befasst sich mit einigen elementaren Begriffen der Pfadplanung, sie werden umschrieben und gegebenenfalls mathematisch definiert.

2.1 Konfigurationsraum

Im Weiteren bezeichne ein **Roboter** eine autonome automatisierte Maschine, die ohne Eingriff des Menschen einprogrammierte Ziele verfolgen kann. Eine **Konfiguration** ist eine mögliche Kombination aus den Einstellungen der Teile eines Roboters, sie spezifiziert jeden Punkt des Roboters im Raum [1], für sie wird das Symbol k verwendet. Der Raum aller möglichen Konfigurationen, die ein Roboter einnehmen kann, heißt **Konfigurationsraum** und wird im Weiteren mit $\mathcal{K}_{Raum}$ bezeichnet. Die Dimensionen des $\mathcal{K}_{Raum}$ sind die **Freiheitsgrade** eines Roboters, diese setzen sich aus der minimalen Anzahl an Parametern zusammen, die nötig sind, um die Konfiguration des Roboters zu beschreiben [1].
Beispiel: Gegeben sei ein kreisförmiger Roboter, der sich ohne Rotation in der Ebene bewegen kann [1], dessen Konfiguration kann über die Koordinatengleichung des euklidischen Abstands zwischen zwei Punkten beschrieben werden. Sei r der Radius des Roboters und (x, y) beschreibe den Ort seines Zentrums innerhalb des zweidimensionalen euklidischen Raums $\mathbf{R}^2$. Die Menge aller Punkte, die ein Roboter im Raum in einer Konfiguration $k = (x, y)$ überdeckt, kann über die Menge $R(k)$ beschrieben werden:

$$R(x, y) = \{(x^{'}, y^{'}) \mid (x - x^{'})^2 + (y - y^{'})^2 \leq r^2\} \tag{1}$$

2.2 Hindernisse

Hindernisse sind Objekte innerhalb des $\mathcal{K}_{Raum}$, die den Roboter auf dem Weg von Start nach Ziel potentiell in seiner Wegfindung behindern. Hindernisse, die am selben Ort verbleiben heißen **statische Hindernisse**, wohingegen Hindernisse, die sich (unvorhergesehen) bewegen **dynamische Hindernisse** genannt werden. Eine Kollision ereignet sich, wenn sich die Konfiguration des Roboters mit einer **Hinderniskonfiguration** h aus dem **Hindernisraum** $\mathcal{H}$ in einem beliebigen Punkt überschneidet. Diese Konfiguration ist ein Element des **Kollisionsraums** $\mathcal{K}_{Kol}$. Weiterhin wird der **freie Konfigurationsraum** mit $\mathcal{K}_{Frei}$ bezeichnet. In dieser Teilmenge des $\mathcal{K}_{Raum}$ finden keinerlei Kollisionen mit Hindernissen statt. Mathematisch lässt sich folgende Beziehung zwischen $\mathcal{K}_{Kol}$,$\mathcal{K}_{Raum}$,$\mathcal{K}_{Frei}$ und $\mathcal{H}$ festhalten:

2.3 Pfad

Der **Pfad** ist eine kontinuierliche Bahn in $\mathcal{K}_{Frei}$. Diese Bahn muss von der **Startkonfiguration** $k(0)$ zur **Zielkonfiguration** $k(1)$ kollisionsfrei sein. Mathematisch betrachtet ist die Lösung eines Pfadplanungsproblems der kontinuierliche Übergang von Konfigurationen abbildet mit folgender Funktion [1]:

$$k : [0, 1] \rightarrow \mathcal{K}_{Raum} \tag{4}$$

Dabei gelte für eine beliebige Konfiguration $k(s)$ auf dem Pfad $k(s) \in \mathcal{K}_{Frei} \forall s \in [0, 1]$.

2.4 Aufgaben

Die Aufgaben, die ein Roboter zu lösen hat, können mit unscharfen Grenzen in die Bereiche Kartierung, Lokalisation, Navigation und Abdeckung eingeteilt werden [1].
Die **Kartierung** beschreibt das Problem der Erkundung und der sensorischen Erfassung unbekannter Gebiete, dabei werden die Daten so repräsentiert, dass sie in der Navigation, Lokalisation und Abdeckung von Nutzen sind. Beispielsweise bewegt sich der Roboter auf Rädern durch das Gelände und tastet mittels eines Lasers seine Umwelt ab. Diese Daten werden in einer internen Repräsentation, der Karte, abgespeichert.
Bei der **Lokalisation** wird durch Nutzung einer Karte und Interpretation der Sensordaten die Konfiguration des Roboters festgestellt. In einer realen Anwendung könnte beispielsweise das „Global Positioning System"(GPS) zu Lokalisation verwendet werden.
Die **Navigation** beschreibt das Problem der Suche eines kollisionsfreien Übergangs von einer Konfiguration zur nächsten. Anschaulich wird dies in der Pfadplanung, die bei semi-automatischen Einparksystemen von PKWs (PARK4U®[2]) verwendet wird.
Bei der **Abdeckung** werden die Punkte des $\mathcal{K}_{Raum}$ je nach Anforderung zu einem gewissen Grad mit Sensoren abgetastet. Eine Aufgabe, die eine vollständige Abdeckung des Raums fordert ist z. B. die Lackierung von Karosserieoberflächen an Fertigungsbändern.

2.5 Algorithmen

Pfadplanung kann vor oder während der Bewältigung einer Aufgabe durchgeführt werden, daher wird die „**offline**" und „**online**" Planung unterschieden. Die wenigsten Pfadplanungsprobleme lassen eine vollständige offline Planung zu, oft kostet diese zu viel Geld oder Zeit. Dem Roboter wird zudem jedwede Flexibilität genommen, wenn dieser zur Laufzeit keine Anpassungen an dessen Pfadplanung vornehmen kann. Pfadplanungsalgorithmen gelten als **optimal**, wenn eine Lösung existiert und diese auch gefunden wird. Ferner können Planer in **lokale** und **globale** Planer getrennt werden. Die lokalen Planer bearbeiten einen Teil des gesamten Pfadfindungsproblems und beziehen dabei nur Informationen aus der direkten Umgebung mit ein. Diese Teillösungen werden durch die globalen Planer zu einer Lösung von Startkonfiguration bis Zielkonfi-

3 Hindernisrepräsentation

Die Repräsentation von Hindernissen im Speicher des Roboters ist ein integraler Bestandteil der Pfadplanung. Bevor der Roboter überhaupt mit der Pfadplanung beginnen kann, müssen Informationen über seine Umgebung vorliegen. Die Quelle dieser Informationen können Datenbanken, andere Roboter oder die eingebauten Sensoren des Roboters selbst sein. Die Daten werden so im Speicher aufbereitet, damit sie von den Planungsverfahren möglichst effizient verarbeitet werden können. Als wichtigste Operationen in der Pfadplanung gelten die Kollisions- und Distanzberechnung, da diese am Häufigsten durchgeführt werden müssen. So stellen die Berechnungsgeschwindigkeiten dieser Operationen ein maßgebliches Kriterium für die Repräsentationswahl dar[3]. Es werden zwei Repräsentationen vorgestellt zum einen die Nutzung von Gittern und zum anderen Abbildung der Hindernisse über polyedrische Repräsentationen, dies sind mögliche Lösungsansätze für die Kartierungs- und Abdeckungsaufgabe.

3.1 Gitter

Die reguläre Implementation dieser Repräsentation von Hindernissen diskretisiert den $\mathcal{K}_{Raum}$ durch ein Feld mit rechtwinklig quadratischen Segmenten. Segmente die ein Hindernis darstellen werden schwarz gefärbt, sie gehören zu $\mathcal{H}$. Alle anderen Felder werden weiß gefärbt, sie gehören zu $\mathcal{K}_{frei}$. Diese Repräsentation eignet sich besonders gut für unregelmäßig geformte Objekte, da die Formen nur approximiert dargestellt werden und nicht jedes Detail berechnet werden muss[3]. Die Berechnung der Nachbarschaftsbeziehungen der einzelnen Zellen zu ihren Nachbarzellen benötigt viel Speicherplatz [3]. Um dem entgegenzuwirken, gibt es eine Vielzahl an Varianten. In Abbildung 1 sind zwei Varianten dargestellt. Die Originalrepräsentation der Objekte im Raum ist in 1 a) dargestellt. Der Quad Tree 1 c) ist eine Weiterentwicklung der Zell Repräsentation 1 b). Hier werden kleinere Zellen aus $\mathcal{K}_{frei}$ zu größeren zusammengefasst, um Speicher und Rechenaufwand zu sparen. Zusätzlich verkleinert der Quad Tree den Suchraum.

3.2 Polyedrische Repräsentation

Die polyedrische Repräsentation wird am häufigsten verwendet, da sich viele Körper gut durch die Vereinigung von Polyedern approximieren lassen. Ein Beispiel für die Repräsentation befindet sich in 1 d).
Die richtige Wahl der Repräsentation hängt davon ab, wie viel Speicher und Rechenleistung dem Roboter zur Verfügung stehen. Gitteransätze sind durch die Berechnun-

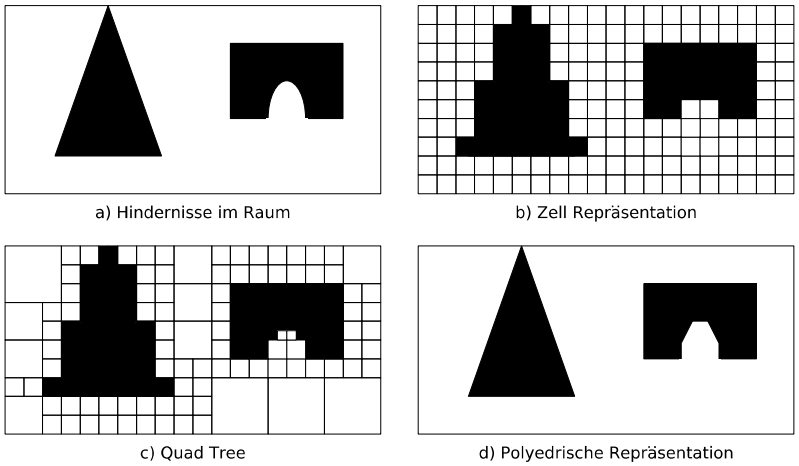

Abb. 1 Übersicht über die vorgestellten Hindernisrepräsentationen

4 Planungsverfahren

Dieser Abschnitt befasst sich exemplarisch mit einigen gängigen Pfadplanungsverfahren. Es werden der Sichtbarkeitsgraph, das Potentialfeld und die Probabilistic Roadmap vorgestellt, sie lösen die Navigations- und Lokalisationsaufgabe.

4.1 Der Sichtbarkeitsgraph

Der Sichtbarkeitsgraph ermöglicht es, den kürzesten kollisionsfreien Weg innerhalb eines $\mathcal{K}_{Raum}$ zu finden. Der Graph wird durch das Verbinden sichtbaren Ecken der Hindernisse erzeugt. Ein Beispiel hierfür findet sich in Abbildung 2. Damit der Robo-

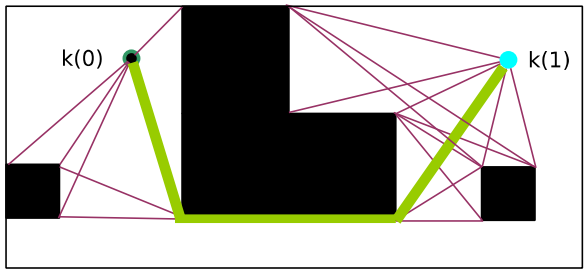

Abb. 2 Ein Sichtbarkeitsgraph mit kürzestem Pfad (grün), der Startkonfiguration k(0) und

um den Radius eines Kreises zu erweitern, dessen Umfang so groß wie die maximale Ausdehnung des Roboters. So kann ohne die Gefahr einer Kollision der Pfad des Roboters geplant werden, dies verringert gleichzeitig den Suchraum für die Navigation. Ein Sichtbarkeitsgraph mit n Ecken kann nach einem Algorithmus von Asano et al. [4] in $O(n^2)$ erzeugt werden.

4.2 Potentialfelder

Die Potentialfelder werden in der Robotik den virtuellen Feldern zugeordnet [5]. Die Grundidee ist ein Kraftfeld aufzuspannen, das durch die Hindernisse in der Umgebung des Roboters definiert ist [5]. Hierbei sind die wichtigsten Modellierungselemente die Definition der Potentialfeldfunktion und die Art der Reaktion des Roboters auf das Feld [5]. Der Roboter kann beispielsweise als „positiv geladener“ Partikel interpretiert werden, der vom stärksten „negativ geladenen“ Punkt, dem Ziel, angezogen wird. Hindernisse sind dabei selbst stark positiv geladen und stoßen den Roboter ab.
Das Potentialfeld kann im zweidimensionalen euklidischen Raum mathematisch als reellwertige differenzierbare Funktion $E : \mathsf{R}^2 \to \mathsf{R}$ beschrieben werden. Eine quadratisches Potentialfeld kann folgendermaßen definiert werden [5]:

$$E(x) = \frac{1}{2}(x-c)^{\mathrm{T}}\,(x-c) \tag{5}$$

x ist der Eingabewert von E beispielsweise ein Vektor mit kartesischen Koordinaten (x_1, x_2). c ist ein Vektor mit Konstanten. Der Gradient, welcher dem Roboter als Wegführung dienen soll ist definiert als [1]:

$$\nabla E = \begin{pmatrix} \frac{\delta E}{\delta x_1} & \frac{\delta E}{\delta x_2} & \cdots & \frac{\delta E}{\delta x_\mathrm{n}} \end{pmatrix}^T \tag{6}$$

Der Gradient 6 angewandt auf die quadratische Potentialfeldfunktion (5) lautet dann[5]:

$$\nabla E = x - c \tag{7}$$

Wenn der Roboter programmiert ist, dem negativen Gradienten zu folgen, so bewegt er sich auf c zu[5]. Diese Art von Potentialfeld wird auch Attraktor genannt, bei Umkehrung des Vorzeichens wandelt sich das Potentialfeld zu einem Repulsor, welches den Roboter dazu bringt, sich von c zu entfernen. Ein quadratisches Potentialfeld mit Attraktoren a und Repulsoren r ist gegeben durch [5]:

$$E(x) = \sum_{a \in \mathsf{A}} \frac{1}{2}(x-a)^{\mathrm{T}}\,(x-a) - \sum_{r \in \mathsf{R}} \frac{1}{2}(x-r)^{\mathrm{T}}\,(x-r) \tag{8}$$

Der zu (8) gehörige Gradient ist dann [5]:

$$\nabla E = \sum_{a \in \mathsf{A}} (x-a) - \sum_{r \in \mathsf{R}} (x-r) \tag{9}$$

die Lesbarkeit der Potentialfeldwerte. Die folgende Funktion beschreibt ein Potentialfeld, bei dem die Parameter α_a, γ_a die Stärke der Attraktoren regeln und β_a, γ_a die Stärke der Repulsoren [5]:

$$E(x) = -\sum_{a \in A} \alpha_a e^{(-\frac{\gamma_a}{2}||x-a||^2)} + \sum_{r \in R} \beta_r e^{(-\frac{\gamma_r}{2}||x-r||^2)} \tag{10}$$

Der zu (10) gehörige Gradient lautet [5]:

$$\nabla E = -\sum_{a \in A} \alpha_a \gamma_a (x-a) e^{(-\frac{\gamma_a}{2}||x-a||^2)} + \sum_{r \in R} \beta_r \gamma_r (x-r) e^{(-\frac{\gamma_r}{2}||x-r||^2)} \tag{11}$$

Veranschaulicht wird (10) in Abbildung 3[5]. Linien gleicher Farben zeigen einen konstanten Wert im Potentialfeld an, die Attraktoren (Feldlinien blau, türkis) sind mit einem (+) symbolisiert und die Repulsoren (Feldlinien gelb,rot) mit einem (∗). Der Roboter könnte durch Verschieben der Attraktoren und Repulsoren durch den Raum gelenkt werden, indem dieser die lokalen Potentiale bzw. Gradienten in seine Pfadplanung mit einbezieht [5].

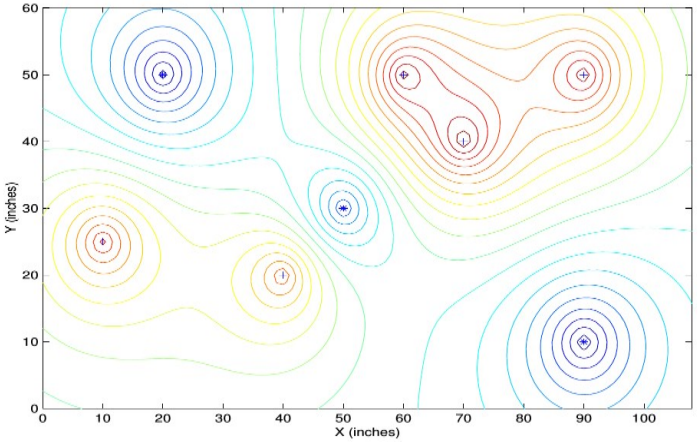

Abb. 3 Exponential Potentialfeld mit fünf Attraktoren (+) und drei Repulsoren (∗) [5]

Ein einfacher Gradientenabstieg wird in Tabelle 1 in Pseudocode beschrieben. Diesem Algorithmus folgend verfolgt der Roboter solange den Gradienten, bis dieser sich ausgelöscht, also gleich Null ist. Dabei sollte ein gewisser Spielraum um die Zielkoordinaten miteinbezogen werden [1], um unnötig vielen Korrekturen durch die Ungenauigkeit der Aktoren entgegenzuwirken. $\alpha(i)$ ist der Parameter für das Fortschreiten des Roboters

Tabelle 1 Ein einfacher Gradientenabstieg Algorithmus[1].

Gradient Descent
Eingabe: Berechne Gradienten von $\nabla E(x)$ am Punkt x **Ausgabe:** Eine Sequenz von Punkten $\{x(0), x(1), ..., x(i)\}$

$x(0) = x_{start}$
while $(\nabla E(x(i)) \neq 0)$ **do**
$x(i+1) = x(i) + \alpha(i)\nabla E(x(i))$
$i = i + 1$
end while

Die Potentialfelder werden oft nur als lokale Planer eingesetzt, da sie nicht optimal sind, denn sie sind anfällig gegenüber lokale Minima. So kann ein Roboter, der ausschließlich dem Gradienten folgt, hufeisenförmigen Hindernissen, die zwischen Start und Ziel liegen nicht entkommen. Der globale Planer kann diese Schwäche beispielsweise durch zufällige Richtungsänderungen (engl. random walks) ausgleichen. Diese und weitere Beschränkungen werden in [6] diskutiert.

4.3 Probabilistic Roadmap [1]

Die Probabilistic Roadmap (PRM) gehört zu den Stichproben (engl. sampling) Algorithmen in der Pfadplanung. Ein zentrales Unterscheidungsmerkmal ist, dass diese Algorithmen nicht versuchen, eine explizite Repräsentation der Hindernisse abzubilden [1]. Es wird sich auf eine Prozedur verlassen, um festzustellen, ob sich der Roboter in Kollision befindet oder nicht [1]. Es wurde gezeigt, dass PRM probabilistisch vollständig ist, das heißt PRM findet bei genügend langer Laufzeit eine Lösung, falls diese existiert [1]. PRM teilt die Planung in zwei Phasen auf. Die erste heißt Lernphase, in dieser Phase wird die Karte (engl. roadmap) mit Konfigurationen aus $\mathcal{K}_{frei}$ erstellt. Die zweite Phase heißt Fragephase (engl. query phase), hier werden benutzerdefinierte Anfragen, die Start- und Zielkonfigurationen enthalten, mit der Karte aus Phase eins, wenn möglich, verbunden. Tabelle 2 enthält den Pseudocode für die Erstellung der Karte. Dabei ist Δ ein lokaler Planer, der zwei Konfigurationen $(k, k') \in \mathcal{K}_{frei} \times \mathcal{K}_{frei}$ mit einem kollisionsfreien Pfad verbindet oder NULL zurückgibt, falls kein solcher Pfad gefunden werden kann. dist sei eine Distanz Funktion $\mathcal{K}_{Raum} \times \mathcal{K}_{Raum} \rightarrow \mathsf{R}^+ \cup \{0\}$. $G = (V, E)$ bezeichne einen ungerichteten Graphen mit V Knoten und E Kanten. In Phase 1 werden solange zufällige (z. B. nach Uniformverteilung) Konfigurationen aus $\mathcal{K}_{Raum}$ gezogen bis n Konfigurationen aus $\mathcal{K}_{frei}$ gefunden wurden. Die Konfigurationen werden der Knotenmenge V hinzugefügt. Jede dieser Konfigurationen wird darauf

Tabelle 2 Konstruktion der Karte, Phase 1 des PRM [1].

Algorithmus zur Kartenerstellung

Eingabe:
n : Anzahl der Knoten, die der Karte hinzugefügt werden
j : Anzahl der nächstgelegenen Nachbarn, die für jede Konfiguration untersucht werden
Ausgabe:
Karte $G = (V, E)$

```
V ← ∅
E ← ∅
while |V| < n do
  repeat
    k ← eine zufaellige Konfiguration in K_Raum
  until k kollisionsfrei
  V ← V ∪ k
end while
for all k ∈ V do
  N_k ← j naechstgelegenen Nachbarn von k aus V nach dist
    for all k' ∈ N_k do
      if (k, k') ∉ E and Δ(k, k') ≠ NULL then
        E ← E ∪ (k, k')
      end if
    end for
end for
```

Tabelle 3 Fragephase, Phase 2 des PRM [1].

Algorithmus zur Auflösen der Anfrage

Eingabe:
k_0 : Startkonfiguration
k_1 : Zielkonfiguration
j : Anzahl der nächstgelegenen Nachbarn, die für jede Konfiguration untersucht werden
Karte $G = (V, E)$ aus Phase 1 (siehe Tabelle 2)
Ausgabe:
Ein Pfad von k_0 bis k_1 oder *Fehlschlag*

```
N_k0 ← j der naechstgelegenen Nachbarn von k_0 aus V nach dist
N_k1 ← j der naechstgelegenen Nachbarn von k_1 aus V nach dist
V ← {k_0} ∪ {k_1} ∪ V
Setze k' als naechstgelegenen Nachbar von k_0 in N_k0
repeat
  if Δ(k_0, k') ≠ NULL then
    E ← (k_0, k') ∪ E
  else
   set k' als naechsten naechstgelenen Nachbar von k_0 in N_k0 ← N_k0 \ k'
until eine Verbindung war erfolgreich oder N_k0 ist leer
Setze k' als naechstgelegenen Nachbar von k_1 in N_k1
repeat
  if Δ(k_1, k') ≠ NULL then
    E ← (k_1, k') ∪ E
  else
    set k' als naechsten naechstgelenen Nachbar von k_1 in N_k1 ← N_k1 \ k'
until eine Verbindung war erfolgreich oder N_k1 ist leer
P ← kuerzester Weg (k_0, k_1, G)
```

Die ersten beiden Teile beschäftigen sich mit dem Versuch die Startkonfiguration k_0 und die Zielkonfiguration k_1 mit j ihnen am nächsten gelegenen Knoten, begrenzt durch die Distanzfunktion *dist*, mit dem lokalen Planer Δ aus der Karte G zu verbinden. Dies wird solange durchführt bis eine Verbindung mit einem $k^{'}$ geschlossen wurde oder keine Nachbarn in Entfernung *dist* mehr vorhanden sind. In Teil drei wird ein kürzester Pfad zwischen k_1 und k_0 gesucht, existiert ein solcher, wird er zurückgegeben ansonsten, schlägt der Algorithmus fehl und liefert eine entsprechende Fehlschlagsmeldung. Der kürzeste Pfad kann mit A* gesucht werden, wie in Abschnitt 5.1 vorgestellt. Für den vorgestellten simplen PRM sind einige vereinfachende Annahmen getroffen worden. Es wird davon ausgegangen, dass die Karte aus Phase 1 zusammenhängend ist und der lokale Planer symmetrisch und deterministisch arbeitet [1]. Weiterhin wurde nicht besonders auf die Verteilung zur Ziehung der Konfigurationen eingegangen, diese und weitere Implementationsdetails werden erschöpfend in [1] beschrieben.
Das vorgestellte Gerüst kann auf die jeweilige Problemstellung angepasst werden. Im Bereich der Stichprobenziehung und Distanzbestimmung kann die Berechnung beschleunigt werden, wie in [7] gezeigt wird. PRM ist probabilistisch vollständig, das heißt der Algorithmus wird mit hinreichend langer Laufzeit einen Pfad finden, insofern ein Pfad existiert. Ob PRM schnell einen Pfad findet, hängt vom Konfigurationsraum ab und dem verwendeten lokalen Planer.

5 Suchalgorithmen

Es existiert eine Fülle an Suchalgorithmen, um die richtige Folge von Konfigurationen zu finden, die den Roboter von der Startkonfiguration zur Zielkonfiguration bringen. Die richtige Wahl hängt von der Beschaffenheit des $\mathcal{K}_{Raum}$ bzw. der Größe des $\mathcal{K}_{frei}$ ab. Ist $\mathcal{K}_{frei}$ groß und es ist möglich, die Berechnungen parallel durchzuführen, empfiehlt sich die Breitensuche [3]. Falls der $\mathcal{K}_{frei}$ groß ist und die Rechenresourcen begrenzt kann eine Tiefen- oder Bestensuche eingesetzt werden [3]. Häufig wird aufgrund seiner Flexibilität der A* Algorithmus eingesetzt, daher wird dieser im folgenden Abschnitt beschrieben.

5.1 A*

A* (lies: A-star) wurde in 1968 entwickelt und kombiniert den Ansatz des kürzeste Wege Algorithmus von Dijkstra mit dem Heuristik Ansatz der Bestensuche [8].Dieser Algorithmus gehört zu den informierten Suchalgorithmen. Bei der Suche wird eine informierte Entscheidung darüber getroffen, ob es sich lohnt den aktuell betrachteten Knoten zu expandieren. Es dürfen weder unwichtige Knoten für die Zielführung expandiert, noch wichtige Knoten ignoriert werden [8]. Die Bewertung für diese Entscheidung wird durch eine Evaluationsfunktion f für einen Knoten n gefällt. In [8] sieht die Funktion folgendermaßen aus:

$$f(n) = g(n) + h(n) \tag{12}$$

Im Falle der Pfadplanung für Roboter könnte $h(n)$ bereits zu Beginn der Planung gegeben sein, beispielsweise durch die Länge der Luftlinie die n und s verbindet. Wie $f(n)$ arbeitet wird am Beispiel eines Graphen in Abbildung 4 klar. Gezeigt wird ein ungerichteter Graph mit Startknoten s und drei weitere Knoten $n1, n2, n3$. Die Zahlen an den Kanten geben die Kosten an. Die rot gestrichelte Kante ist eine gedachte direkte Verbindung zwischen Start s und Ziel $n3$ mit Kosten von 4 und diene als $h(n)$. Es wird mit s gestartet und mögliche Folgeknoten sind $n1$, $n2$. Die Schätzungen $\hat{g}(n1) = 3$ und $\hat{g}(n2) = 7$, so würde A* mit der Expansion von $n1$ fortfahren. $n1$ besitzt Folgeknoten $n2$ und $n3$ mit den Schätzungen $\hat{g}(n2) = 6$ und $\hat{g}(n3) = 5$. Der Pfad zu $n2$ wird aktualisiert, da der neue Pfad über $n1$ geringere Kosten hat. In diesem Fall wäre die Suche abgeschlossen und der kürzeste Pfad von s nach $n3$ gefunden.

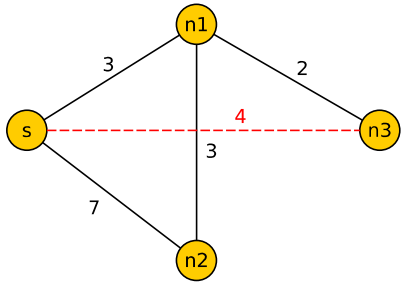

Abb. 4 Beispiel für die Arbeitsweise der Bewertungsfunktion

Die Beschreibung des A* aus dem Originalpaper[8] befindet sich in Tabelle 4. Hierbei seien T die präferierten Zielknoten und Γ der Nachfolgeroperator, der für einen gegebenen Knoten alle Nachbarn des Knotens ausgibt. A* ist optimal [8]. Die Laufzeit

Tabelle 4 Original A* beschrieben wie in [8].

A*
Eingabe: Startknoten s, Zielknotenmenge T, Graph mit positiven Kantengewichten **Ausgabe:** Pfade von s zu Zielknoten in T

1. Markiere s als offen und berechne $\hat{f}(s)$.
2. Selektiere den offenen Knoten n dessen Wert $\hat{f}$ am kleinsten ist. Löse dies stets zu Gunsten der Knoten $n \in \mathsf{T}$
3. Falls $n \in \mathsf{T}$, markiere n als geschlossen und terminiere.
4. Ansonsten, markiere n als geschlossen und wende den Nachfolgeroperator Γ auf n an. Berechne $\hat{f}$ für jeden offenen Nachfolgerknoten von n, der nicht bereits als geschlossen markiert ist. Markiere jeden geschlossenen Knoten n_i als offen, welcher ein Nachfolger von n ist und der aktuell ein niedrigeres $f(n_i)$ besitzt, als zu dem Zeitpunkt als n_i geschlossen wurde. Fahre fort mit Schritt 2.

Fall $O(|V^2|)$. A* wurde in viele Richtungen erweitert und angepasst, ein bekanntes Derivat ist D*.

6 Fazit und Ausblick

Im Verlauf dieser Arbeit wurde eine kleine Auswahl an Herangehensweisen aus dem großen Forschungsgebiet der Pfadplanung präsentiert. Der PRM-Algorithmus kann mit den Potentialfeldern als lokalen Planer und A* als Suchalgorithmus für den kürzesten Pfad eingesetzt werden, um die globale Pfadplanung zu realisieren. Die Auswahl der Subroutinen und Metriken eines Pfadplanungsalgorithmus müssen stets an das Problem angepasst werden. In [9] werden mögliche Bausteine eines Stichproben Algorithmus vorgestellt, die dazu dienen den Algorithmus auf das Problem zu spezialisieren. Verglichen mit dem Sichtbarkeitsgraph, welcher nur für die Planung im 2D Raum geeignet ist [3], kann der vorgestellte PRM zur Lösung hochdimensionaler Pfadplanungsprobleme eingesetzt werden, d. h. für Roboter, die Freiheitsgraden von 5 bis 12 besitzen [1].
Im Rahmen dieses Seminars wird noch in anderen Arbeiten tiefergehend auf das Thema Pfadplanung eingegangen siehe Themenblöcke „Pfadplanung mit stetiger Krümmung“, „Pfadplanung für Parkvorgänge“ und „Pfadplanung mit Unsicherheiten im Mess- und Stell- System“. Für einen Einblick in die aktuelle Forschung der Echtzeit Pfadplanung siehe [10].

Literatur

1. S. H. Howie Choset, Kevon Lynch, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion*. The MIT Press, 2005.
2. [Online]. Available: http://www.valeo.com/en/our-activities/comfort-and-driving-assistance-systems/technologies/park4u-81.html
3. Y. K. Hwang and N. Ahuja, “Gross motion planning - a survey,” *ACM Computing Surveys*, vol. 24, no. 3, p. 219, September 1992.
4. T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai, “Visibility of disjoint polygons,” *Algorithmica*, vol. 1, no. 1, pp. 49–63, November 1986.
5. R. W. Beard and T. W. McLain, “Motion planning using potential fields,” 2003.
6. Y. Koren and J. Borenstein, “Potential field methods and their inherent limitations for mobile robot navigation,” in *Proceedings of the IEEE Conference on Robotics and Automation, Sacramento, California*. IEEE, April 1991, pp. 1398–1404.
7. R. Geraerts and M. H. Overmars, “A comparative study of probabilistic roadmap planners,” *Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR’02)*, p. 43–57, 2002.
8. P. E. Hard, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Transactions of systems science and cybernetics*, vol. ssc-4, no. 2, July 1968.
9. A. S. López, R. Zapata, and M. A. O. Lama, “Sampling-based motion planning: A survey planificación de movimientos basada en muestreo: Un compendio,” *Computación y Sistemas*, vol. 12, no. 1, pp. 5–24, 2008, ISSN 1405-5546.
10. C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, “Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions,” *Loughborough University Institutional Repository*, p. 416–442, 2015.