

Implementierung einer Bewegungsplanung für einen Roboterarm

Implementation of a path-planning algorithm for a robot arm

Dipl.-Ing. Bertram Rohrmoser
Christopher Parlitz
Fraunhofer IPA
Stuttgart, Germany

Zusammenfassung

Dieser Artikel beschreibt die Implementierung einer Bahnplanung für einen Roboterarm auf einer mobilen Plattform. Hierbei wurde besonderer Wert auf effiziente Algorithmen gelegt, um die Bahn des Roboters schnell in einer unbekanntem statischen Umgebung anzupassen. Die entwickelte Planung erfolgt in zwei Phasen. Im ersten Schritt wird eine kollisionsfreie Trajektorie mit Hilfe eines Rapidly-Exploring Random Tree (RRT) bestimmt. Dieser Algorithmus von Steven M. LaValle [1] wurde in der Arbeit zur Performancesteigerung modifiziert. Die so gefundene Trajektorie wird im zweiten Schritt geglättet, indem die Glättung als nicht-lineares Optimierungsproblem betrachtet und mit Hilfe von Standardverfahren gelöst wird.

Summary

This paper describes the implementation of a path-planning algorithm for a robot-arm on a mobile platform. Emphasis was laid on efficient algorithms so that the path of the robot can be adapted quickly to a unknown static environment. The developed planning algorithm takes place in two phases. In the first step a collision-free trajectory is generated by using a Rapidly-Exploring Random Tree. This algorithm by S. LaValle [1] was adapted to increase the performance. In a second step this generated trajectory is smoothed by treating this smoothing as a non-linear optimization problem which is solved with standard tools.

1 Einführung

Ziel dieser Arbeit ist die Entwicklung einer Bewegungsplanung für den Roboterarm einer mobilen Plattform (siehe Abb. 1). Die wechselnde Einsatzumgebung erfordert die Anpassung der Planung an die jeweils aktuelle Umgebung. Mit einem 3D-Scanner wird hierzu zyklisch ein Umweltmodell erstellt, das aus den einzelnen gemessenen Hindernispunkten besteht. Für eine schnelle Kollisionskontrolle hat dies zur Folge, dass sie auf einer Datenstruktur basieren muss, die sowohl einen schnellen Aufbau als auch eine schnelle Bestimmung des minimalen Abstandes aus einer Menge von Punkten ermöglicht. Eine Datenstruktur, die diesen Anforderungen genügt, ist der aus der Datenbanktechnik bekannte UB-Tree, dessen Einsatz hier zur Bahnplanung vorgeschlagen wird.

Der RRT von LaValle ist in der Lage, Lösungen in einem großen Suchraum, der sich aus einer hohen Zahl von Freiheitsgraden ergibt, schnell zu finden. Doch der Zeitaufwand der Suche steigt mit der Erhöhung des Detaillierungsgrades durch die Verkleinerung der Diskretisierungsabstände exponentiell an. Der in dieser Arbeit vorgeschlagene Ansatz erweitert den RRT-Ansatz von LaValle um eine Schrittweitensteuerung, die gewährleistet, dass der maximale Abstand zwischen den Positionen eines Massenpunktes des Roboterarmes in aufeinanderfolgenden Konfigurationen eine vorgegebene Schranke nicht überschreitet. Hierdurch kann die Kollisionsprüfung im Arbeitsraum erfolgen und die Planungsgenauigkeit beliebig festgelegt werden.

Um Trajektorien, die nur mit Hilfe von RRTs ermittelt werden, für eine Roboterbewegung zu verwenden, ist eine sehr detaillierte Planung unter hohem Zeitaufwand nötig. Daher wurde bei dem hier entwickelten Planungsalgorithmus nur eine grobe globale, dafür aber schnelle Suche im Konfigurationsraum mit RRTs durchgeführt. Die resultierende Trajektorie wird anschließend als guter Startwert für eine lokale Optimierung genutzt. Da die lokale Optimierung im Gegensatz zur RRT-Planung lokale Eigenschaften wie den Gradienten der Gütefunktion nutzt, verspricht dieser Ansatz eine höhere Effizienz als eine reine RRT-Planung.

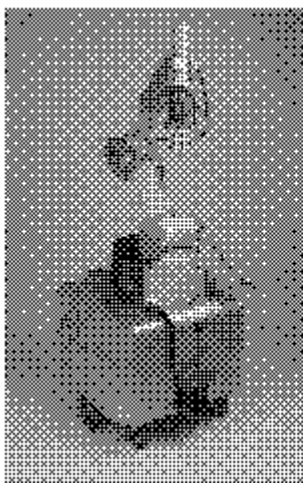


Abbildung 1: Der mobile Roboterarm „rob@work“

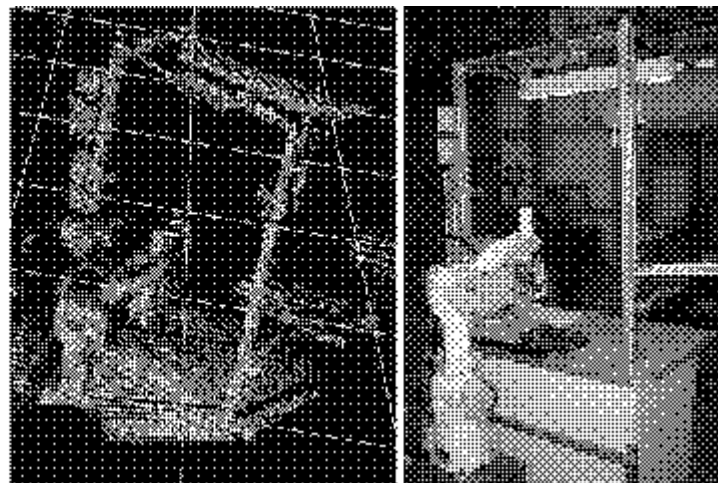


Abbildung 2: Beispiel eines ermittelten Umweltmodells

2 Dynamische Schrittweite bei RRTs

Die Rapidly-Exploring Random Trees führen eine Suche im Konfigurationsraum durch, dessen Dimension der Freiheitsgrade des Arms entspricht. Der von LaValle [1] beschriebene Algorithmus ist vereinfacht folgender: Ausgehend von der Start- und der Zielkonfiguration wachsen zwei Random Trees im Konfigurationsraum aufeinander zu, bis ein Kontakt zwischen den beiden Bäumen zustande kommt. Das Wachstum eines Baumes beginnt bei einer Konfiguration x_{mit} .

Für das Wachstum wird eine Konfiguration x_{rand} und der zu ihr nächstliegende Knoten x_{near} des Baumes ermittelt. Die Konfiguration, die man erhält, wenn man von x_{near} um den Abstand

Δt in Richtung x_{rand} schreitet, wird als neuer Knoten x_{new} hinzugefügt.

```

GENERATE_RRT( $x_{init}$ ,  $K$ ,  $\Delta t$ )
rrt.init( $x_{init}$ );
for k=1 to K do
   $x_{rand}$  ← RANDOM_STATE();
   $x_{near}$  ← NEAREST_NEIGHBOR( $x_{rand}$ );
   $x_{new}$  ← NEW_STATE( $x_{near}$ ,  $\Delta t$ );
  rrt.add_vertex( $x_{new}$ );
return rrt

```

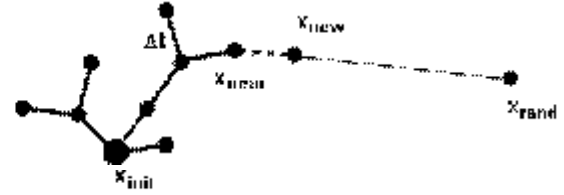


Abbildung 3: Erweiterung des RRTs

Ein entscheidender Parameter bei der Erzeugung eines RRTs ist die Schrittweite Δt , die festlegt, wie weit sich der Baum in Richtung der zufällig erzeugten Konfiguration bewegt. LaValle führt eine Kollisionsüberprüfung im Konfigurationsraum durch und legt die Schrittweite statisch fest. Die Umrechnung der Hindernisse in den Konfigurationsraum ist aber mit großem Aufwand verbunden, da sie bei jeder neuen Umgebungserfassung wiederholt werden muss.

Der hier vorgestellte Ansatz ist daher, die Kollisionsprüfung im kartesischen Raum durchzuführen und den Abstand s , um den sich ein Massepunkt des Roboterarms in zwei aufeinanderfolgenden Konfigurationen maximal bewegen darf, festzulegen, damit keine Kollision mit den Hindernissen auftreten kann. Dies erfordert eine Berechnung der Schrittweite Δt im Konfigurationsraum bei jeder Iteration, da sie von der aktuellen Konfiguration x_{near} abhängig ist.

Jeder Massenpunkt $\bar{P}_i(\bar{q})$ des Arms hängt von der aktuellen Konfiguration \bar{q} ab. Mittels der durch x_{near} und x_{rand} festgelegten Richtung \bar{r} , lässt sich Δt , als maximal erlaubte Schrittweite abhängig von \bar{q} , wie folgt berechnen:

$$\Delta \bar{P}_i = \bar{P}(\bar{q} + \Delta \bar{q}_i) - \bar{P}(\bar{q}) \quad (1)$$

$$s = \|\Delta \bar{P}_i\| \quad (2)$$

$$\Delta \bar{P}_i \approx \frac{\partial \bar{P}_i}{\partial \bar{q}} \Delta \bar{q}_i = \frac{\partial \bar{P}_i}{\partial \bar{q}} \bar{r} \Delta t_i \quad (3)$$

$$\Delta t = \min_i \{\Delta t_i\} \quad (4)$$

3 Kollisionsprüfung mit UB-Trees

Bevor ein x_{new} dem Baum hinzugefügt werden kann, muss überprüft werden, ob alle Massenpunkte des Roboters in dieser Konfiguration einen ausreichenden Abstand zu den Hindernissen haben. Ebenfalls muss bei der Optimierung einer gültigen Trajektorie weiterhin sichergestellt sein, dass eine Kollision mit Hindernissen ausgeschlossen ist.

Das mit dem 3D-Scanner ermittelte Umweltmodell ist eine Wolke von Hindernispunkten (siehe Abb. 2). Ein naiver Ansatz zur Kollisionskontrolle wäre ein Vergleich aller Hindernispunkte

mit den Massenpunkten des Roboterarms. Es sei n die Anzahl der Hindernispunkte und m die Anzahl der Massenpunkte. Der Aufwand für eine Kollisionsprüfung wäre dann $O(m \cdot n)$.

Eine wesentliche Performance-Steigerung kann mit Hilfe eines UB-Tree [2] erreicht werden. Der UB-Tree ist eine effiziente Datenstruktur, die mit einem einmaligen Aufwand von $O(n \lg n)$ aufgebaut und in der mit einem Aufwand von $O(\lg n)$ eine Bereichssuche durchgeführt werden kann (Für eine genaue Aufwandsermittlung siehe [2].) Hieraus folgt für eine Kollisionsabfrage ein Aufwand von $O(m \lg n)$. Die Performance des UB-Tree kann weiter durch eine dem Problem angepasste Wertebereichsauflösung und -größe der Hinderniskoordinaten gesteigert werden.

4 Glättung einer Trajektorie

Die Glättung einer Trajektorie kann erreicht werden, indem die Länge der Endeffektorbahn im Arbeitsraum unter Berücksichtigung von Randbedingungen minimiert wird. Bei diesem sich ergebenden nicht-linearem Optimierungsproblem (NLP-Problem) wird allgemein x^* gesucht, für das gilt:

$$f(x^*) = \min_{x \in \mathcal{S}} f(x) \quad (5)$$

$$\mathcal{S} = \{x \in \mathbb{R}^n : \bar{h}(x) = 0, \bar{g}(x) \geq 0\} \quad (6)$$

Dabei bestehen \bar{g} und \bar{h} aus jeweils n_g bzw. n_h einzelnen Gleichungen und

$$f(x) = \alpha_1 Q_1 + \alpha_2 Q_2 + \alpha_3 Q_3 \quad (7)$$

Um bei der Optimierung der einzelnen Konfigurationen einer Trajektorie eine Glättung zu erreichen, wird also die benötigte Gütefunktion $f(x)$ aus der gewichteten Summe der folgenden einzelnen Gütekriterien gebildet:

- Länge einer Bahn:

Grundsätzlich wird eine Minimierung der Bewegung des Endeffektors zwischen Start- und Ziellage im Arbeitsraum gefordert. Dazu werden aus den einzelnen Konfigurationen mittels der Vorwärtstransformation die Endeffektorkoordinaten berechnet. Sei \bar{P}_k die Lage des Endeffektors bei Konfiguration q_i , so lässt sich formulieren:

$$Q_1 = \|\bar{P}_2 - \bar{P}_1\|^2 + \dots + \|\bar{P}_n - \bar{P}_{n-1}\|^2 \quad (8)$$

- Länge einer Trajektorie:

$$Q_2 = \|\bar{q}_2 - \bar{q}_1\|^2 + \|\bar{q}_3 - \bar{q}_1\|^2 + \dots + \|\bar{q}_n - \bar{q}_{n-1}\|^2 \quad (9)$$

Wird nur die Länge der Bahn des Endeffektors im Arbeitsraum minimiert, haben Simulationen gezeigt, dass dies sehr große Winkeländerungen in den Gelenken hervorruft. Deutlich wird dies vor allem bei redundanten Systemen, bei denen sich Gelenkwinkel zum Teil gegenseitig aufheben und daher kaum Auswirkungen auf die Bewegung des Endeffektors haben.

Eine minimale Trajektorienlänge im Konfigurationsraum verhindert diese Effekte, der Endeffektor führt dann aber große Bögen im Arbeitsraum aus. Daher besteht die Gütefunktion hier aus beiden Kriterien, um eine möglichst ausgewogene Bewegung zu erhalten.

- Gleichmäßige Verteilung der Konfigurationen auf der Trajektorie (durch q_1 bis q_n diskretisiert):

$$Q_3 = (\|\bar{q}_2 - \bar{q}_1\| - \|\bar{q}_3 - \bar{q}_2\|)^2 + \dots + (\|\bar{q}_{n-1} - \bar{q}_{n-2}\| - \|\bar{q}_n - \bar{q}_{n-1}\|)^2 \quad (10)$$

Je gleichmäßiger die Konfigurationen eine Trajektorie beschreiben, umso leichter lässt sich daraus die eigentliche Ansteuerung an den Roboter berechnen, da z.B. eine lineare Interpolation zwischen den einzelnen Konfigurationen ausreichend ist.

Des Weiteren müssen die Randbedingungen \bar{g} bzw. \bar{h} angegeben werden. Diese bilden sich aus:

- Einhaltung eines Mindestabstandes (mit Hilfe von Abs. 3):
Der Sicherheitsabstand muss so groß gewählt werden, dass eine Kollision eines Hindernispunktes auch mit zwischen zwei Massenpunkten liegenden Teilen des Roboterarms ausgeschlossen ist. Dazu werden mehrere Ungleichungen g_i formuliert, die diese Bedingung sicherstellen.
- Beachtung von Beschränkungen der Gelenkwinkel:
Aus technischen Gründen gibt es grundsätzlich eine minimale und eine maximale Stellung eines Gelenks, die ebenfalls über Ungleichungen eingehalten werden.
- Feste Ränder:
Da die Ränder einer Trajektorie als Start- und Zielkonfiguration fest vorgegeben sind, müssen diese konstant bleiben. Dies wird über einfach formulierte Gleichungen h_j erreicht.

Zur Durchführung der Optimierung wurde DONLP2 [3] von Peter Spellucci ausgewählt.

5 Experimente und Tests

Implementiert wurde die Planung in der Programmiersprache C++. Sämtliche Parameter, wie z.B. die Kinematik des Roboterarms und Planungsgenauigkeit sind konfigurierbar, wobei die Kinematik mittels Denavit-Hartenberg-Parametern beschrieben wird. Die Visualisierung der Ergebnisse erfolgt mit VRML.

Zum jetzigen Zeitpunkt liegen noch keine experimentellen Ergebnisse vor. Doch konnte in Simulationen bereits die Flexibilität und Vielseitigkeit des Algorithmus gezeigt werden. In einer Simulationsszene war die Aufgabe einer 6-DOF Roboterarm-Kinematik über eine Wand zu greifen.

Abb. 4 zeigt hierzu den ungeglätteten und den optimierten Pfad des Endeffektors, Hindernispunkte sowie den Roboterarm in Start- und Zielkonfiguration.

Die Bestimmung einer gültigen Trajektorie mit Hilfe des abgewandelten RRT-Algorithmus erfolgt bei einer solchen Aufgabe je nach Genauigkeit meist unterhalb einer Zehntelsekunde. Die Glättung mit Hilfe des aktuell gewählten Optimierungsverfahrens liegt dagegen im Bereich einer Minute und muss für eine effektive Planung noch verbessert werden.

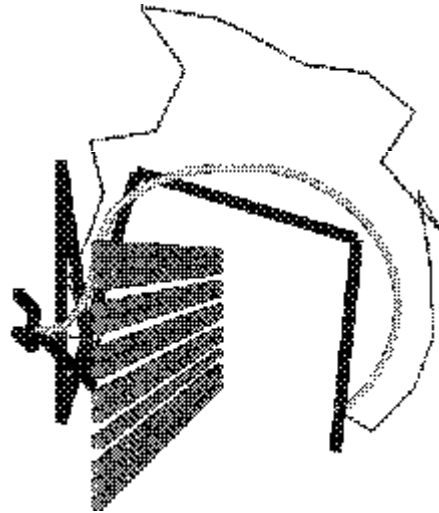


Abbildung 4: Ungeglätteter und optimierter Pfad über eine Wand

6 Ausblick

Der RRT ermöglicht eine sehr schnelle globale Suche in komplexen Umgebungen und liefert eine kollisionsfreie Trajektorie. Eine notwendige Glättung der gefundenen Trajektorie ist mit dem aktuell verwendeten Verfahren entgegen der Erwartung nicht schnell genug für eine zufriedenstellende Robotersteuerung. Allerdings ist die Leistungsfähigkeit des verwendeten Optimierungsalgorithmus noch nicht vollständig ausgeschöpft worden, sodass eine Performancesteigerung noch möglich ist, die Grundlage aktueller Arbeiten ist.

Geplante Experimente mit einem am Fraunhofer IPA entwickelten Serviceroboter „rob@work“ sollen die gemachten Simulationen bestätigen.

Die diesem Bericht zugrundeliegende Arbeit wurde teilweise im Rahmen des Leitprojekts MORPHA [4] mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IL902G/9 gefördert.

Literatur

- [1] LaValle, Steven: *Rapidly-Exploring Random Trees: A New Tool for Path Planning*, Technical Report No. 98-11, Dept. of Computer Science, Iowa State University, 1998
- [2] Bayer, Rudolf: *The Universal B-Tree for multidimensional Indexing*, Technische Universität München, 1996
- [3] Spellucci, P.: *DONLP2: SQP/ECQP-method for general continuous nonlinear programming*, http://mathematik.tu-darmstadt.de/ags/ag8/Mitglieder/spellucci_de.html
- [4] *MORPHA - intelligente antropomorphe Assistenzsysteme*, <http://www.morpha.de>, 2001