Parallele Bewegungsplanung in dynamischen Umgebungen[1]

CHRISTIAN WURLL, DOMINIK HENRICH, HEINZ WÖRN

August 1997

Institut für Prozeßrechentechnik und Robotik (IPR)
Universität Karlsruhe (TH), Kaiserstraße 12, 76128 Karlsruhe
[wurll | dHenrich | woern@ira.uka.de]

Dieser interne Bericht gibt einen Überblick über die aktuellen Forschungsergebnisse aus dem gleichnamigen Projekt. Hierbei wird das Problem der praktikablen Bewegungsplanung für Industrieroboter in dynamischen Umgebungen angegangen. Der Grundalgorithmus ohne wesentliche off-line Berechnungen basiert auf der A-Suche und arbeitet im impliziten, diskretisierten Konfigurationsraum. Die Kollisionen werden im kartesischen Arbeitsraum durch hierarchische Abstandsberechnung im gegebenen CAD-Modell erkannt. Eine zyklische Aufteilung des Suchraums auf die einzelnen Prozessoren ermöglicht eine gut skalierbare Parallelverarbeitung für nachrichten-gekoppelte Rechnersysteme. Die Leistungsfähigkeit des Bewegungsplaners wird an einem Satz von Benchmark-Problemen validiert. Unterstützt durch eine optimale Diskretisierung zeigt der neuartige Ansatz einen linearen Speedup. Für Umgebungen mit unbewegten Hindernissen liegen die Laufzeiten im Sekundenbereich. Zur weiteren Beschleunigung der Bewegungsplanung wird erstmalig eine heuristische hierarchische Suche im impliziten Konfigurationsraum eingeführt. Für zweidimensionale Benchmark-Probleme ergibt die Hierarchisierung eine starke Reduktion des Suchaufwandes.*

Keywords: Bewegungsplanung, A*-Algorithmus, Bestensuche, Parallelverarbeitung, Lastverteilung, Domain Decomposition, Hierarchische Diskretisierung, Kollisionserkennung

1 Einleitung

Die automatisierte Bewegungsplanung in der Robotik ist ein traditionelles Problem und von großem Interesse, da ein durchschlagender Einzug in industrielle Applikationen noch nicht stattgefunden hat [Kamal96]. Eine prägnante Einführung in dieses Gebiet ist dem Buch von [Latombe91] zu entnehmen. Eine darauf aufbauende aktuellere Zusammenstellung ist in dem Übersichtsartikel [Hwang92] zu finden.

Ausgehend von einem gegebenen Umweltmodell wird bei der *Bahnplanung* nur die Kinematik, also die Position und Orientierung der roboterbeschreibenden Objekte und deren mechanischen Beziehungen zueinander betrachtet. Unter Berücksichtigung des Robotermodells gilt es, für einen Roboter einen kollisionsfreien Weg (Folge von Zwischenpunkten) von einer vorgegebenen Start- zu einer Zielkonfiguration zwischen den Hindernissen zu finden. Das Problem hierbei ist der exponentiell steigende Rechenaufwand bei Hinzunahme von zusätzlichen Freiheitsgraden des Roboters. Schon bei

[1] Diese Arbeit ist am Institut für Prozeßrechentechnik und Robotik (IPR) der Universität Karlsruhe unter der Leitung von Prof. Dr. U. Rembold, Prof Dr. H. Wörn und Prof. Dr. R. Dillmann entstanden und wurde im Rahmen des DFG-Projektes „Skalierbare Algorithmen für die parallele Bewegungsplanung in dynamischen Umgebungen“ im Schwerpunktprogramm „Effiziente Algorithmen für diskrete Probleme und ihre Anwendungen“ gefördert. Weitere Informationen gibt es auf der Web-Seite der PaRo-Gruppe (Parallele Robotik) des IPRs unter http://wwwipr.ira.uka.de/~paro/.

einem Roboter mit sechs rotatorischen Gelenken und 360° Grad Bewegungsfreiheit pro Gelenk ergibt sich bei einer Diskretisierung von 1° ein sechsdimensionaler Zustandsraum (*Konfigurationsraum*) mit $360^6 \approx 2.2*10^{15}$ Zuständen. In diesem Raum existieren in Abhängigkeit von den Hindernissen erlaubte und verbotene Zustände, die durch Kollisionsbetrachtung erkannt werden müssen (siehe Abbildung 1-1). Geht man zum Beispiel davon aus, daß eine Kollisionsuntersuchung 10 ms dauert, so muß die Bahnplanung so erfolgen, daß maximal 100 Zustände auf Kollision untersucht werden dürfen, um innerhalb einer Sekunde eine Lösung zu erhalten.

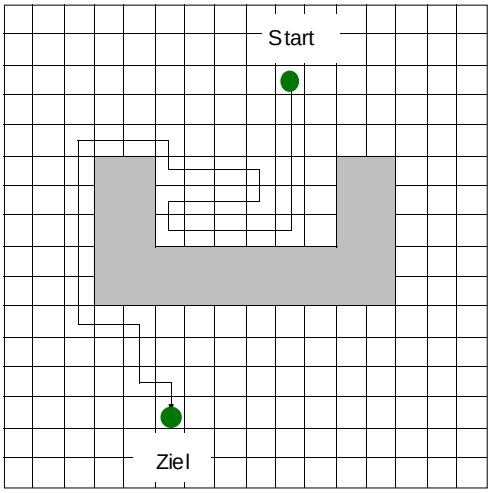


Abbildung 1-1: Vereinfachte Darstellung einer Bahnplanung im zweidimensionalen Konfigurationsraum.

Für die zukünftigen Einsatzgebiete der Robotik, wie z.B. Recycling, Demontage oder Medizintechnik, werden hohe Anforderungen an die Bewegungsplanung gestellt. Wenn in den bisherigen Einsatzbereichen der Robotik Planer für zwei oder drei Freiheitsgrade noch ausreichten, so sind heute Planer für sechs Freiheitsgrade erforderlich. Erschwerend kommt hinzu, daß sich bei zukünftigen Anwendungen die Umgebungen, in denen die Industrieroboter arbeiten, dynamisch verändern können. *Dynamische Umgebungen* liegen vor, wenn sich Hindernisse bewegen, die Umweltgeometrie sich zwischen zwei Planungen verändert oder wenn der Roboter ein Objekt gegriffen hat. Im letzten Fall hat die kinematische Kette (Roboter und gegriffenes Objekt) seine Geometrie verändert, was ebenfalls zu einer Veränderung des Konfigurationsraumes führt. Aus diesem Grund darf ein leistungsfähiger Bewegungsplaner, der in dynamischen Umgebungen arbeiten soll, auch off-line keine zeitaufwendigen Berechnungen enthalten. Ist die Bewegungsplanung hinreichend schnell, so können die Algorithmen darüber hinaus auf verwandte Problemstellungen wie z.B. die CAD-Einbausimulation, die rechnergestützten Chirurgie oder das Molekül-Design u.a. übertragen werden.

Sequentielle Rechner sind heute nur noch durch großen technologischen Aufwand zu beschleunigen, da man sich mit der Leistungsfähigkeit den physikalischen Grenzen nähert. Sie scheiden daher für einen längerfristigen Ansatz für rechenintensive Problemstellungen aus. Als Ausweg bietet sich der Einsatz von paralleler Datenverarbeitung an. Mit Parallelrechnern ist durch Hinzunahme von weiteren Prozessoren und mit entsprechend komplexen Problemen potentiell eine beliebig große Leistungssteigerung zu erreichen. Dafür muß das Problem bewältigt werden, die vorhandene Parallelität effektiv auszunutzen.

Zur Bewertung paralleler Algorithmen kann eine Reihe von Kriterien, wie z.B. Beschleunigung (*Speedup*), Effizienz und Skalierbarkeit aufgestellt werden [Lee80]. Bei der *Effizienz* sind schnelle Algorithmen (im allgemeinen) und eine gute Ausnutzung der vorhandenen Prozessoren (bei der Parallelität) gefragt. Möchte man eine größere Beschleunigung der Problemlösung durch die Hinzunahme

von weiteren Prozessoren erreichen, so ist die *Skalierbarkeit* ein Maß dafür, inwieweit die Effizienz des Algorithmus aufrecht erhalten wird. Leider kann die Rechenzeit einer fest vorgegebenen Problemlösung durch parallele Verarbeitung im Grenzfall maximal bis auf den inhärent sequentiellen Anteil reduziert werden (Amdahls law) [Gustafson88]. Für eine Einführung in den Entwurf und die Analyse von skalierbaren parallelen Algorithmen siehe [Kumar94].

Im folgenden wird in Abschnitt 2 ein Überblick über den Stand der Forschung auf den Gebieten der Bahnplanung und der Graphensuche gegeben. In Abschnitt 3 wird die Arbeitsumgebung vorgestellt, die im Rahmen des Projektes entwickelt wurde. Inhalt des 4. Abschnittes ist der sequentiellen Grundansatz, der von den Autoren zur Lösung der Aufgabe verwendet wurde. Abschnitt 5 beschreibt die notwendigen Maßnahmen zur Parallelisierung der Bewegungsplanung. Neben der Parallelisierung ist die Hierarchisierung eine wichtige Methode zur Beschleunigung der Planung. Diese wird in Abschnitt 6 näher betrachtet. Der Bericht endet mit einer Zusammenfassung und einem Ausblick in Abschnitt 7.

2 Stand der Forschung

In diesem Abschnitt erfolgt ein Überblick über den Stand der Forschung in dem Bereich der Bewegungsplanung. Im Anschluß daran wird gesondert der Bereich Graphensuche im Hinblick auf die Bewegungsplanung betrachtet.

2.1 Bahnplanung

Die Komplexität der Bahnplanung ist schon seit längerem als PSPACE-schwer bekannt [Reif79]. Ein traditioneller Lösungsansatz ist die Transformation der Hindernisse in den von [Lozano-Pérez79] eingeführten Konfigurationsraum. Darauf aufbauend wird dann eine geeignete Repräsentation des Freiraums berechnet, die sich auf einen Graphen reduzieren läßt. In diesem Graphen kann schließlich ein Weg von der Start- zur Zielkonfiguration gesucht werden. Zur Suche wird z.B. der A*-Algorithmus [Hart68] oder Dijkstra's Kürzeste-Wege-Algorithmus [Dijkstra59] eingesetzt.

Zur Beschleunigung der Suche benutzte Faverjon für die Freiraum-Repräsentation eine hierarchische Octree-Darstellung [Faverjon84]. Um eine optimale Lösung zu finden, wurde in [Gerke86] die dynamische Programmierung vorgeschlagen. Hörmann definierte den kartesischen Raum als Konfigurationsraum, wodurch die Transformation der Hindernisse entfällt [Hörmann87]. Es existieren auch zahlreiche Arbeiten, die durch andersartige Zerlegung des Konfigurationsraumes [Adolphs92, Quinlan93] oder durch Vereinfachung der Roboter- und Hindernisgeometrie [Fink91] kürzere Rechenzeiten erreichen.

Vorteilhaft bei diesen Verfahren zur offline-Planung ist die Möglichkeit, alle Faktoren wie z.B. Einschränkung der Umgebung, Dynamik des Roboters oder Optimalität der gesuchten Bahn berücksichtigen zu können. Von der Anwenderseite erfahren diese Ansätze allerdings eine geringe Akzeptanz, weil sie zu rechenaufwendig sind und sich nicht eignen bei Veränderung der Umwelt, bei sich bewegenden Objekten, in kooperierenden Robotersystemen und in Echtzeit-Steuerungen [Yu95].

Für eine Bahnplanung in dynamischen Umgebungen ist die explizite Berechnung des Konfigurationsraum auf Grund der zeit- und speicheraufwendigen Hindernistransformation nicht mehr praktikabel. In diesem Sinne sucht Verwer einen Weg im Konfigurationsraum und überprüft eine mögliche Kollision im Arbeitsraum [Verwer90]. Kondo sucht mit unterschiedlichen Parametersätzen im implizit gegebenen Konfigurationsraum, um die Wegfindung durch Vermeidung von unnötigen zeitaufwendigen Kollisionstest zu beschleunigen [Kondo91]. Aufbauend auf der Abstandsberechnung nach [Gilbert88], dem zweistufigen Planer nach [Faverjon87] zerlegt Chen den Konfigurationsraum hierar-

chisch, um so die Suchzeit zu reduzieren [Chen92]. Elias entwickelte ebenfalls einen zweistufigen Ansatz. Nach einer Zerlegung des Konfigurationsraumes sucht er in der groben Auflösung (20°) mit dem A*-Algorithmus nach einer Verbindung von Start zum Ziel. Trifft die Suchfront auf eine gemischte Zelle wird in der feineren Auflösung (5°) mit Hilfe einer Breitensuche ein Weg geplant [Elias94].

Eine weitere Methode zur Beschleunigung der Bahnplanung ist der Einsatz von paralleler Datenverarbeitung. Einen detaillierten Überblick und eine Klassifikation der verschiedenen Ansätze ist in [Henrich96] zu finden. Zu den parallelen Algorithmen in der Bahn- oder Bewegungsplanung gibt es nur relativ wenige Vorarbeiten. Das Resumee des Überblicks ist, daß für die parallele Bewegungsplanung ein erhöhter Forschungsbedarf besteht.

Betrachtet man insgesamt die Arbeiten bezüglich paralleler und hierarchischer Bewegungsplanung, so zeigt sich, daß durch Parallelisierung und Hierarchisierung die Planungszeit von mehreren Stunden auf wenige Sekunden reduziert werden kann. Leider werden bei den bisherigen parallelen und hierarchischen Ansätzen eine der drei folgenden entscheidenden Einschränkungen vorgenommen:

- Entweder werden nur einfache geometrische Beschreibungen der Objekte, wie z.B. grobe Rasterdarstellungen oder Liniensegmente, verwendet.
- Oder es werden keine dynamischen Hindernisse für den Roboter betrachtet, so daß z.B. eine offline Transformation der Hindernisse in den Konfigurationsraums möglich wird.
- Oder die Anzahl der berücksichtigten Freiheitsgrade wird verringert, so daß der Suchraum kleiner ist und nur eingeschränkt für reale Probleme einsetzbar ist.

2.2 Graphensuche

Der von den Autoren entwickelte effiziente parallele Bewegungsplaner für Industrieroboter baut auf der Graphensuche auf. Die Graphensuche ist ein Teilgebiet der kombinatorischen Optimierung und es gibt eine Vielzahl von Vorarbeiten und algorithmischen Varianten. Die grundlegenden Strategien stellen die Bestensuche (best-first, jumptracking) und die Tiefensuche (depth-first, backtracking) dar. Eine Einführung in Suchverfahren wird in [Horowitz89] gegeben. Ein Vergleich der unterschiedlichen Ausprägungen der Suchverfahren ist in [Kumar83] zu finden.

Die Herausforderung bei (Graphen-) Suchverfahren ist der enorme Speicherbedarf in Form von Suchknoten zur internen Steuerung der Suche. Für die verbreiteten Suchalgorithmen, wie z.B. Bestensuche oder A*, wächst der Speicherbedarf im schlimmsten Fall exponentiell mit der Suchtiefe. Es gibt mehrere Methoden um den maximalen Speicherbedarf zu reduzieren. Zu der Grundmethode gehört die Tiefensuche, welche an sich schon speichereffizient ist. Leider ist die Tiefensuche nur in Bäumen sinnvoll anwendbar, da bei Gittern sehr viele redundante Wege auftreten und damit der Suchraum expnentiell vergrößert werden kann. Weitere interessante Varianten zu speichereffizienten Suchverfahren sind MA* [Chakrabarti89], iterative Tiefensuche [Korf85], MREC [Sen89] und PRA* [Evett90].

Im Vergleich mit anderen Planungs- und Optimierungsmethoden (Genetische Algorithmen und sonstige randomisierte Verfahren) haben die Graphensuchverfahren verschiedene Vorteile. Neben der Exaktheit des Verfahrens (d.h. es kann immer die optimale Lösung gefunden werden) ist auch die schnellere Berechnung von suboptimalen Lösungen möglich. Denn unter der Voraussetzung, daß die heuristische Bewertungsfunktion des Suchverfahrens für den Knoten n die Form $f(n) = (1-w) * g(n) + w * h(n)$ besitzt, dann kann gezeigt werden, daß die Kosten der suboptimalen Lösungen nur um den Faktor $w / (1 - w)$ teurer als die der optimalen Lösung sind [Shapiro92].

Echtzeitfähige Suchverfahren wurden bislang nur von wenigen Autoren untersucht. Korf präsentierte mehrere Varianten von echtzeitfähigen A*-Algorithmen [Korf85, Korf87, Korf88, Korf90]. Durch Modifikation der heuristischen Bewertung des aktuellen Knotens in Abhängigkeit von seinen

Nachfolgern wird die Suche entscheidend beschleunigt. Im Gegensatz zum konventionellen A*-Algorithmus wird darüber hinaus in jedem Iterationsschritt eine Aktion (z.B. eine physikalische Bewegung des Roboters) ausgeführt und nicht bis zur endgültigen Lösungsfindung gewartet. Ishida entwickelte aufbauend auf den Ansätzen nach Korf und in Anlehnung an [Pohl71] eine echtzeitfähige bidirektionale Suche [Ishida96].

3 Arbeitsumgebung

Das Forschungsziel ist die Entwicklung eines parallelen Bewegungsplaners für Industrieroboter, der für den Einsatz in dynamischen Umgebungen geeignet ist und somit keine wesentlichen offline Berechnungen enthält. In den folgenden Abschnitten wird dazu die Arbeitsumgebung vorgestellt, zunächst werden als Rahmen in Abbildung 2-1 die Beziehungen zwischen den Software-Komponenten aufgezeigt. Gegeben ist ein Weltmodell in dem Robotersimulationssystem ROBCAD. Darauf aufbauend wurde eine Applikation entwickelt (Abschnitt 3.1), welche eine Auswertung der geplanten Bewegungsbahnen (Abschnitt 3.2) und die Erstellung von Benchmark-Problemen zuläßt (Abschnitt 3.3). Als Rechner-Hardware wird für die Applikation eine Silicon Graphics und für die parallele Bewegungsplanung die IPR-ParaStation verwendet (Abschnitt 3.4).

3.1 Simulationsumgebung

Zum Erstellen von Aufgabenstellungen, zum Aufrufen den Planers, sowie zur Visualisierung der Ergebnisse wurde mit Hilfe des Robotersimulationssystems ROBCAD ein Modul erstellt. In Abbildung 3-2 ist die entwickelte Applikation mit einem geladenen Benchmark-Problem dargestellt. Eine ausführliche Beschreibung der einzelnen Funktionen ist [Katz96] zu entnehmen.

3.2 Bewertungskriterien

Der in diesem Projekt erstellte Bewegungsplaner soll in der ersten Phase mittels Simulation ausführlich getestet und bewertet werden. Für die Bewertung ist es notwendig, bestimmte Kriterien zu definieren, die einen objektiven Vergleich mit anderen Planern ermöglichen. Hierbei ergab sich jedoch das Problem, daß in der Literatur kein zufriedenstellender Ansatz zur Bewertung von Bewegungsplanern existiert. Der meist benutzte Ansatz besteht darin, die benötigte Planungszeit anzugeben [Gupta96].

Ein Ansatz zum Vergleichen verschiedener Algorithmen ist in [Hwang92] angegeben. Die Algorithmen werden hierbei nach dem Weltmodell, mit dem sie arbeiten, eingeteilt. Darauf aufbauend werden verschiedene Fähigkeiten (Anzahl der Freiheitsgrade, global/lokal, Geschwindigkeit, ...) des Algorithmus bestimmt und tabellarisch angegeben. Diese Angaben sagen jedoch nur etwas über die Algorithmen, die zur Bahnplanung benutzt werden, aus. Unter Berücksichtigung dieser Problematiken wurde im Rahmen des Projektes ein Satz von Bewertungskriterien aufgestellt.

Neben dem trivialen Kriterium „Planer hat eine Lösung gefunden“, welches vor allem für lokale Planungsansätze interessant ist, können weitere Kriterien angegeben werden. Diese lassen sich in zwei Gruppen einteilen:

1. Die verschiedenen Zeiten, die der Planer zur Generierung der Bewegungsbahn benötigt, z.B:
   - die Zeit für notwendige Offline-Berechnungen,
   - die Zeit zur Anpassung des Weltmodells an dynamische Veränderungen und
   - die Zeit zur Planung der Bewegung.

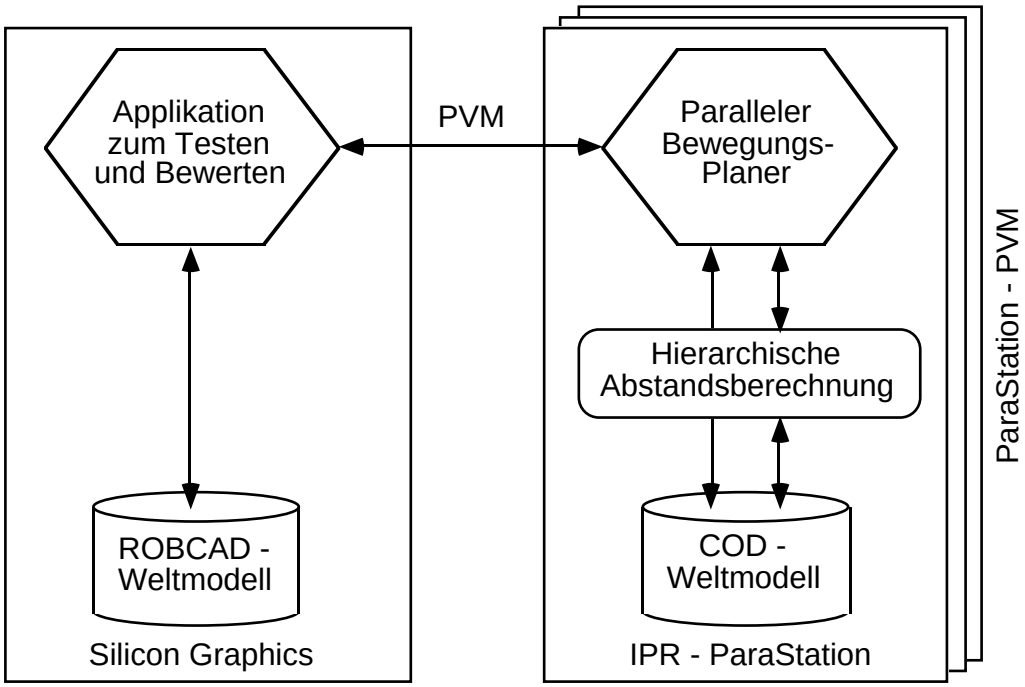


Abbildung 3-1: Struktur der entwickelten Software-Komponenten für den parallelen Bewegungsplaner

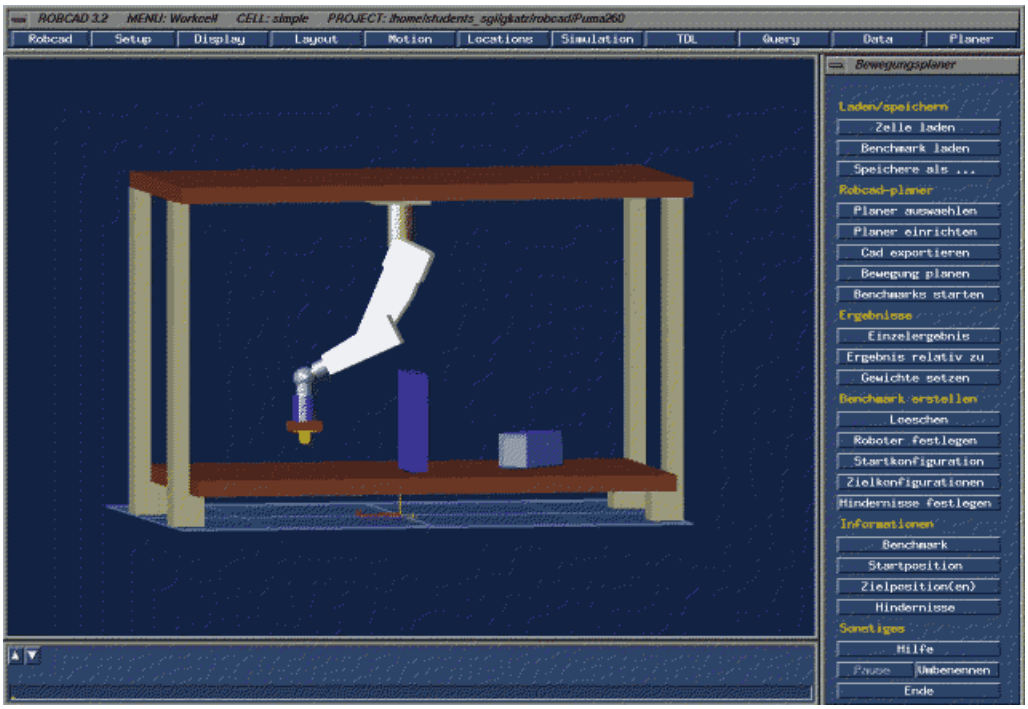


Abbildung 3-2: Bildschirmkopie der entwickelten ROBCAD-Applikation für den parallelen Bewegungsplaner

2. Die Qualität der gefundenen Lösung, z.B:
    - die gewichtete Summe der Gelenkbewegungen.

Eine ausführliche Herleitung und mathematische Formulierung dieser Kriterien ist aus [Katz96] zu entnehmen.

3.3 Benchmark-Satz

Eine objektive Einschätzung der Leistungsfähigkeit eines Bewegungsplaners kann nur dann gewährleistet werden, wenn die Aufgabenstellungen, die der Planer lösen muß und für die die Kriterien nach Abschnitt 3.2 gemessen werden, vergleichbar mit den Aufgabenstellungen der anderen Planer sind. Ein Problem hierbei ist, daß die Planer mit verschiedenen Robotertypen arbeiten, wodurch ein direkter Vergleich nicht durchführbar ist. Daher benötigt man eine Vorgehensweise, mit der man die die Bahnplaner roboterunabhängig vergeleichen kann.

Da ein Raum mit mehr als drei Dimensionen für den Menschen schwer vorstellbar ist, wird von einer Spezifikation der Benchmark-Probleme im realen Raum abgesehen. Eine mögliche Alternative stellt der Konfigurationsraum dar. In einem vereinfachten 2D-Schema können die Hindernisse sowie die Start- und Zielkonfigurationen definiert werden. Für den Planervergleich im realen Raum müssen dann aus den 2D-Schemata roboterabhängige Arbeitszellen gemäß den Modellierungshinweisen in [Katz96] aufgebaut werden.

Ausgehend von den 2D-Benchmark-Problemen SIMPLE, STAR, BOTTLENECK und TRAP aus [Hwang96], wurden in [Katz96][2] die entsprechenden 6DOF-Benchmark-Probleme für einen Puma260 aufgestellt. Der von Hwang vorgeschlagene Benchmarksatz wurde von Katz um das Benchmark-Problem DETOUR erweitert. Dieses zusätzliche Benchmark-Problem dient zur Überprüfung, ob der Planer auch die Wegkosten mit bei der Lösungsfindung berücksichtigt. In Abbildung 3-3 sind diese Testumgebungen jeweils für zwei und sechs Freiheitsgrade dargestellt.

Im Vergleich zu realen industriellen Umgebungen heben diese einzelnen Benchmark-Problem typische geometrische Konstellationen im Konfigurationsraum hervor. Durch die starke Betonung, wie z.B. die Sackgasse bei TRAP, wird zwar der Planungsaufwand in den meisten Fällen erheblich vergrößter, aber es kann dadurch auch auf die spezifischen Schwächen der jeweiligen Ansätze zur Bewegungsplanung besser eingegangen werden.

3.4 Parallelrechner-Plattform

Als Parallelrechnerplattform für den parallelen Bewegungsplaner wird ein Workstation-Cluster, die IPR-ParaStation, eingesetzt (siehe Abbildung 3-4). Aufgrund der großen Verbreitung, einem guten Preisleistungsverhältnis und der leichten Erweiterbarkeit wurde das Cluster aus Standard-PCs vom Typ Pentium mit 133 MHz aufgebaut. Eine Einsteckkarte ermöglicht eine Hochleistungskommunikation und feingranulare Parallelisierung. Eine zyklische, gitterförmige Vernetzung erlaubt darüber hinaus eine flaschenhalsfreie parallele Kommunikation (siehe Abbildung 3-5). Zwischen zwei PCs wird bei der Kommunikation von jeweils 1000 Nachrichten mit 1 bis 1024 Bytes im Mittel eine Latenzzeit von 65,7 µsec und ein Datendurchsatz von 12,1 Mbit / sec. erreicht [Wurll97].

---

[2] Alle Benchmark-Probleme sind in verschiedenen CAD-Formaten auf der Homepage der PaRo-Gruppe (Parallelverarbeitung in der Robotik) des Instituts für Prozeßrechentechnik und Robotik im World Wide Web (WWW) weltweit verfügbar unter der Adresse: http://wwwipr.ira.uka.de/~paro/.

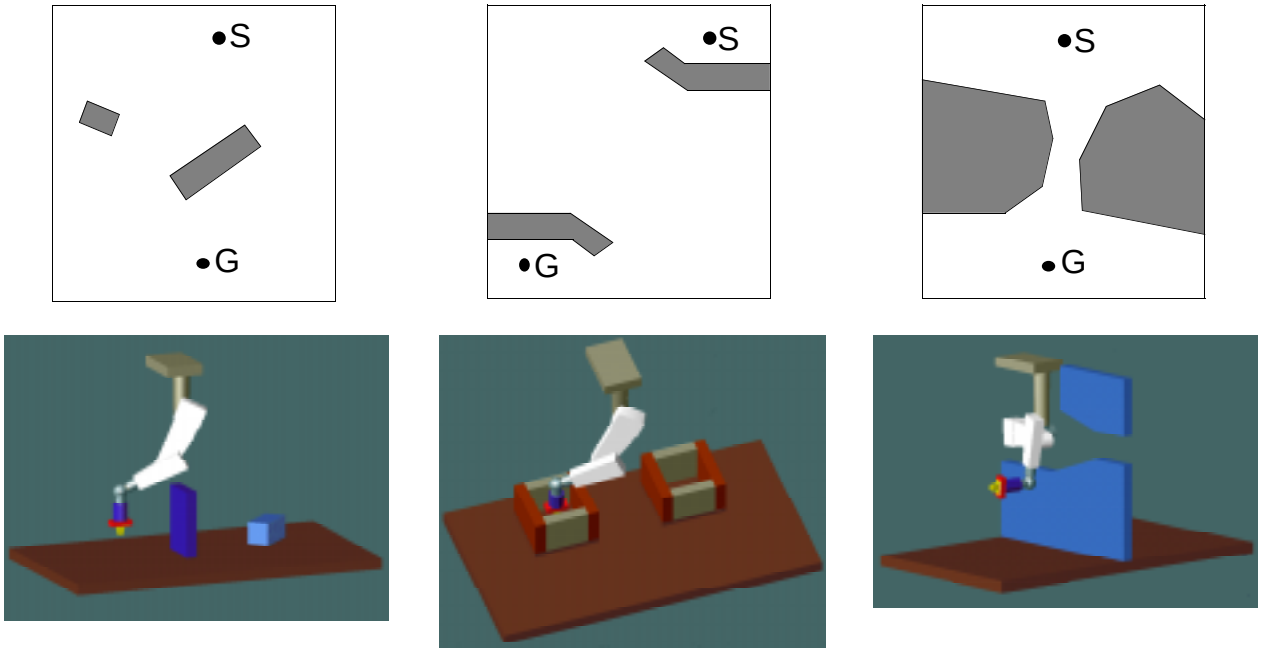


Abbildung 3-3: 2D-Schemata mit Angabe der Start- (S) und Zielposition (G) sowie die entsprechenden 3D-Testumgebungen für die Bewegungsplanung mit 6 Freiheitsgraden für die Benchmarkprobleme SIMPLE, STAR, BOTTLENECK, TRAP und DETOUR

Abbildung 3-4: Der Arbeitsplatz an der IPR-ParaStation

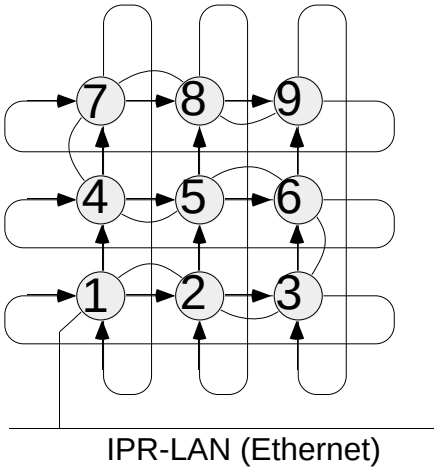


Abbildung 3-5: DieNetzwerktopologie der IPR-ParaStation mit Prozessoren 1,…,9

4 Sequentielle Bewegungsplanung

4.1 Grundansatz

Der in diesem Projekt entwickelte Bewegungsplaner basiert auf dem bekannten A*-Algorithmus [Hart68]. Dieser arbeitet in dem impliziten, diskretisierten Konfigurationsraum, der durch die Gelenkwinkelbereiche des Roboters aufgespannt wird. Kern des Algorithmus ist ein sukzessive aufgebauter Graph, der alle interessanten Wege vom Start bis zum Ziel enthält. Beginnend mit der Start-Konfiguration als Wurzelknoten werden in jedem Iterationsschritt alle möglichen Nachfolger des aktuellen Knotens generiert und entsprechend einer Bewertungsfunktion in die Prioritätenliste OPEN eingetragen. Der expandierte Knoten kommt in die Hashtabelle CLOSED, die alle bis dahin gefundenen Teilwege enthält. Vor der Expansion des nächsten Knotens aus OPEN wird dieser zunächst auf Kollisionsfreiheit getestet. Dies erfolgt mit Hilfe einer hierarchischen Abstandsberechnung im kartesischen Arbeitsraum (siehe Abschnitt 4.3) [Henrich97].

Insgesamt wird also im impliziten, dargestellten Konfigurationsraum gesucht und im explizit repräsentierten Arbeitsraum Kollisionen erkannt. Auf diese Weise kann die zeit- und speicheraufwendige Transformation der Hindernisse in den Konfigurationsraum sowie eine Freiraumberechnung vermieden werden und erlaubt somit die Planung auch für dynamische Umgebungen.

4.2 Optimale Diskretisierung

Die Diskretisierung des Konfigurationsraums hat einen entscheidenden Einfluß auf die Effizienz des Bahnplaners. Ist die Diskretisierung zu fein, dann wird der Suchraum so groß, so daß der Planer unter Umständen aus Speicherplatzproblemen keine Lösung berechnen kann. Ist die Diskretisierung zu grob, dann kann der Fall eintreten, daß der Planer auf Grund der zu grob approximierten Hindernisse keine Lösung findet. Erschwerend kommt hinzu, daß eine heuristisch eingestellte Diskretisierung für die eine Umgebung sehr gute Ergebnisse liefern kann, die gleiche Einstellung bei einer anderen Umgebung aber gänzlich versagt. Wie soll jetzt aber nun die Diskretisierung eingestellt werden?

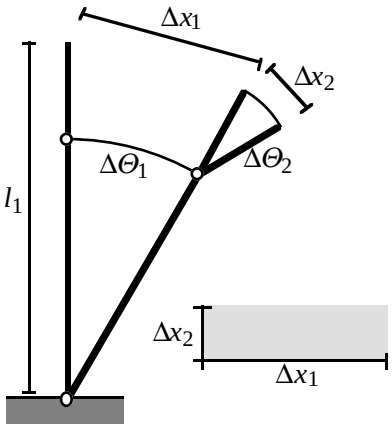


Abbildung 4-1.: Resultierende maximale kartesische Bewegung $\Delta x_1, \Delta x_2$ eines Roboterarmes der Länge l_1 bei der Veränderung des 1. und 2. Gelenkes um die Winkel $\Delta\Theta_1$ und $\Delta\Theta_2$

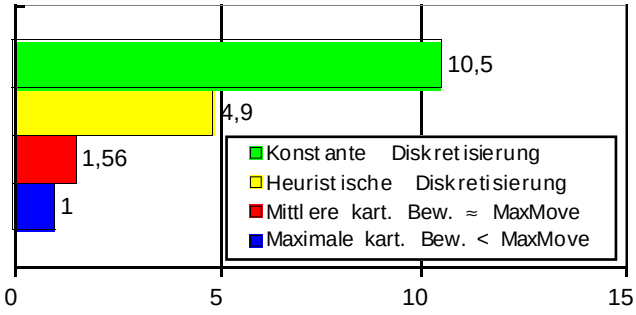


Abbildung 4-2: Verhältnis der maximalen kartesischen Bewegung $\max\{\Delta x_i\}$ zur minimalen kartesischen Bewegung $\min\{\Delta x_i\}$ bei unterschiedlichen Diskretisierungarten $\Delta\Theta$

Je nach Wahl der Diskretisierung hat die Bewegung des Gelenkes *i* um die Schrittweite $\Delta\Theta_i$ unterschiedliche maximale Bewegungen Δx_i im realen Raum zur Folge (siehe Abbildung 4-1). Da im Rahmen der Bahnplanung allerdings der kartesische Abstand mit Hilfe der in [Katz96] eingeführten MaxMove-Tabellen in den Gelenkwinkelraum umgerechnet wird, bewirken die ungleichmäßigen Seitenverhältnisse in der Regel eine ineffiziente Ausnutzung des Abstands. Damit kann es vorkommen, daß Knoten (Roboterkonfigurationen) als kollidierend bewertet werden, obwohl sie real kollisionsfrei sind.

Aus dieser Überlegung heraus wurde aufbauend auf [Qin96] ein Verfahren zur optimalen Diskretisierung in den vorhandenen Planer integriert. In Abhängigkeit einer benutzervorgegebenen maximalen Bewegung des Roboters im realen Raum (MaxMove) werden die resultierenden Gelenkdiskretisierungen über einen trigonometrischen Zusammenhang berechnet. Eine ausführliche Beschreibung ist [Qin96] sowie [Beeh97] zu entnehmen. Diese Wahl der automatischen Diskretisierung hat zur Folge, daß die Umrechnung des kartesischen Abstands in den Gelenkwinkelraum gleichmäßige Seitenverhältnisse bewirken und somit der Abstand besser ausgenutzt werden kann. In Abbildung 4-2 sind für unterschiedliche Diskretisierungsarten die Seitenverhältnisse der maximalen zu minimalen kartesischen Bewegung dargestellt.

Per Definition sind die Seitenverhältnisse der optimalen Diskretisierung (Maximale kart. Bew. < MaxMove) gleich Eins. Wie es zu erwarten war, schneidet die konstante Diskretisierung $(\Delta\Theta_i = 1)$ mit einem Seitenverhältnis von 10,5 am schlechtesten ab. Die heuristische Diskretisierung $(\Delta\Theta = [2,2,4,4,6,6])$ liegt mit 4,9 allerdings immer noch weit weg vom Optimum.

4.3 Abstandsberechnung

Die Kollisionserkennung ist eine der wesentlichen Funktionen der Bewegungsplanung (siehe auch Abschnitt 4.1). In jedem Iterationsschritt muß überprüft werden, ob eine Konfiguration erlaubt ist oder nicht. Je schneller also die Kollisionserkennung durchgeführt werden kann, umso schneller erfolgt dann auch die Planung einer Bewegung.

Eine Erweiterung der Kollisionserkennung ist die Abstandsberechnung. Während eine Kollisionserkennung lediglich die Aussagen „frei“ bzw. „kollidierend“ zuläßt, liefert eine Abstandsberech-

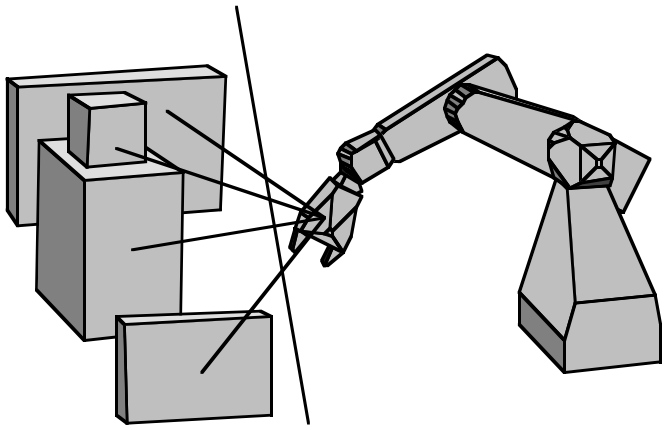


Abbildung 4-3: Prinzip einer Abstandsberechnung zwischen Roboter und Hindernissen im realen Arbeitsraum.

nung wesentlich mehr Informationen. Aus dem Abstand kann zum Beispiel eine Strecke und nicht nur ein Punkt als „frei“ definiert werden. Darüber hinaus ist die Abstandsberechnung die Grundlage für die hierarchische Bewegungsplanung (siehe Abschnitt 5), da größere Hindernisabstände auch größere Schritte im Konfigurationsraum erlauben. Ohne eine erneute Abstandsberechnung kann im Rahmen einer Abstandsfortschreibung auch über benachbarte Konfigurationen eine Aussage über deren Zustand gemacht werden. Schließlich kann der jeweilige Abstand bei der Trajektorienplannung zur Berechnung der möglichen Geschwindigkeitsprofile verwendet werden, da große Hindernisabstände eine Umfahrung mit höherer Geschwindigkeit erlauben.

In diesem Sinne wird hier eine Abstandsberechnung verwendet und diese zur Vermeidung der komplexen und speicheraufwendigen Hindernistransformation im Arbeitsraum durchgeführt. Aufbauend auf dem gegebenen CAD-Modell in Form von konvexen Polyedern werden zwischen Roboter und Hindernissen der kleinste Abstand berechnet.

Zur Beschleunigung dieser Vorgehensweise wurde in [Henrich92] zusätzlich eine hierarchische Beschreibung der Hindernisse und Roboter aufgebaut (siehe Abbildung 4-4a). Bei einer Abstandsberechnung werden dann zunächst die höchsten korrespondierenden Hierarchieebenen miteinander geschnitten. Im Überschneidungsfalle müssen die entsprechenden Objekte weiter verfeinert werden (siehe Abbildung 4-4b). Diese Vorgehensweise wird so lange fortgesetzt bis eine Kollision bestätigt oder eine untere Schranke für den minimalen Abstand zwischen Roboter und Hindernissen zurückgegeben werden kann. Um den Rechenaufwand weiter zu reduzieren werden alle Roboterobjekte durch sogenannte r-Zylinder approximiert [Henrich97].

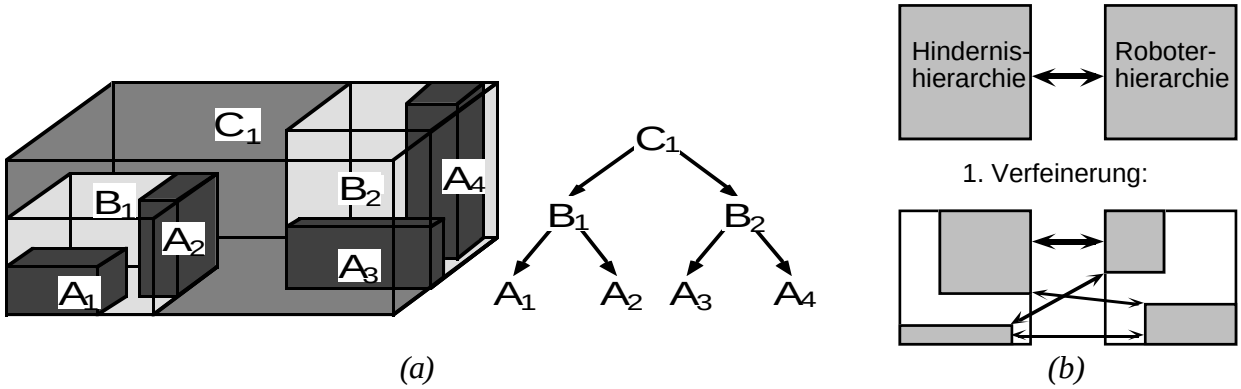


Abbildung 4-4: Komponenten bei der hierarchischen Abstandsberechnung: (a) Aufbau einer Objekthierarchie, (b) Prinzip der sukzessiven Verfeinerung der Hindernis- und Roboterhierarchie

5 Parallele Bewegungsplanung

Die parallele Datenverarbeitung dient zur Beschleunigung rechenaufwendiger Funktionen. Sie sollte also dort eingesetzt werden, wo sequentielle Lösungsansätze zu langsam sind. Im Rahmen der Bewegungsplanung gibt es dafür zwei mögliche Ansatzpunkte. Zum einen kann die Kollisionserkennung an sich, als eine der aufwendigsten Teilfunktionen parallelisiert werden, und zum anderen kann das Gesamtproblem parallelisiert werden. In den Abschnitten 5.1 und 5.2 werden diese beiden Varianten genauer vorgstellt. In den Abschnitten 5.3 und 5.4 wird dann noch auf zwei wichtige Aspekte der parallelen A*-Suche eingegangen.

5.1 Parallele Abstandsberechnung

Die Kollisionserkennung ist eine der wichtigsten Funktionen im Rahmen der Bewegungsplanung. Es zeigt sich, daß ein Großteil des Gesamtaufwands für die Kollisionserkennung benötigt wird. So wird zum Beispiel in [Metivier90] von einem Anteil bis zu 80% ausgegangen. Aus diesem Grund wurde in [Gontermann97] untersucht, inwieweit sich die Kollisionserkennung und damit auch die Bewegungsplanung durch Parallelisierung beschleunigen läßt.

Aufbauend auf dem bekannten besten sequentiellen Ansatz wurde dieser um drei Mechanismen erweitert. Kern des besten Ansatzes ist eine Bestesuche, die durch Einführung von lokalen OPEN-Listen je Prozessor parallelisiert wird. Der dabei notwendige Austausch von Suchbaumknoten erfolgt mittels gewichtetem Random-polling. Dies führt zu einer nahezu perfekten Lastverteilung. Dabei kann die Terminierung der Bestensuche durch einen 2-farbigen Token im virtuellen Ring zuverlässig festgestellt werden. Die Experimente zeigen eine Beschleunigung der Abstandsberechnung insbesondere bei wachsender Umweltgröße.

Um nun eine genauere Abschätzung für den erreichbaren Speedup zu erhalten, kann der Anteil der einzelnen Arbeitsschritte am Gesamtaufwand bestimmt werden. Unter der idealisierenden Annahme, daß der Zeitaufwand für die Bestensuche durch die Parallelisierung nicht mehr ins Gewicht fällt, erhält man z.B. für den Benchmark BOTTLENECK einen maximal erreichbaren Speedup von 4,86. Bei dem hier eingesetzten Parallelrechner, der IPR-ParaStation, mit neun Prozessoren, kann sich der Zeitaufwand für die Bestensuche (39,23 s) bei einer idealen Parallelisierung höchstens um den Faktor 9 verringern, d.h. auf ca. 4,36 s. Der maximal erreichbare Speedup beträgt damit nur noch 3,40. Analog beträgt für die WBK1000 Umwelt mit 1.000 Hindernissen der maximale Speedup 26,95 und auf einen Parallelrechner mit neun Prozessoren 6,94.

Die Abbildung 5-1 zeigt die Berechnungszeiten und die Beschleunigungen für drei unterschiedlich große Umweltmodelle (BOTTLENECK, WBK100, WBK1000[3]). Die Berechnungen wurden sequentiell und parallel mit 2, 4 und 8 Prozessoren durchgeführt. Die Ergebnisse zeigen, daß die relative Beschleunigung durch Parallelisierung mit der Anzahl von Hindernissen zunimmt. Dies liegt einerseits an der Reduktion des sequentiellen Anteils im parallelen Gesamtalgorithmus von ca. 20% auf unter 2%. Andererseits verbessert sich das Verhältnis Kommunikation (Knotenaustausch) zu Kalkulation (Abstandsberechnungen). Der erreichte Speedup für eine Umgebung mit 10 bzw. 1000 Hindernissen beträgt 1,5 bzw. 3. Dagegen ist der bei dieser Parallelisierung maximal mögliche Speedup 3,4 bzw. 6,94. Somit wird etwa die Hälfte der bei dieser Parallelisierung maximal erreichbaren Beschleunigung erzielt.

[3] WBK100 bzw. WBK1000 sind industrielle Umweltmodelle mit 100 bzw. 1000 Objekten.

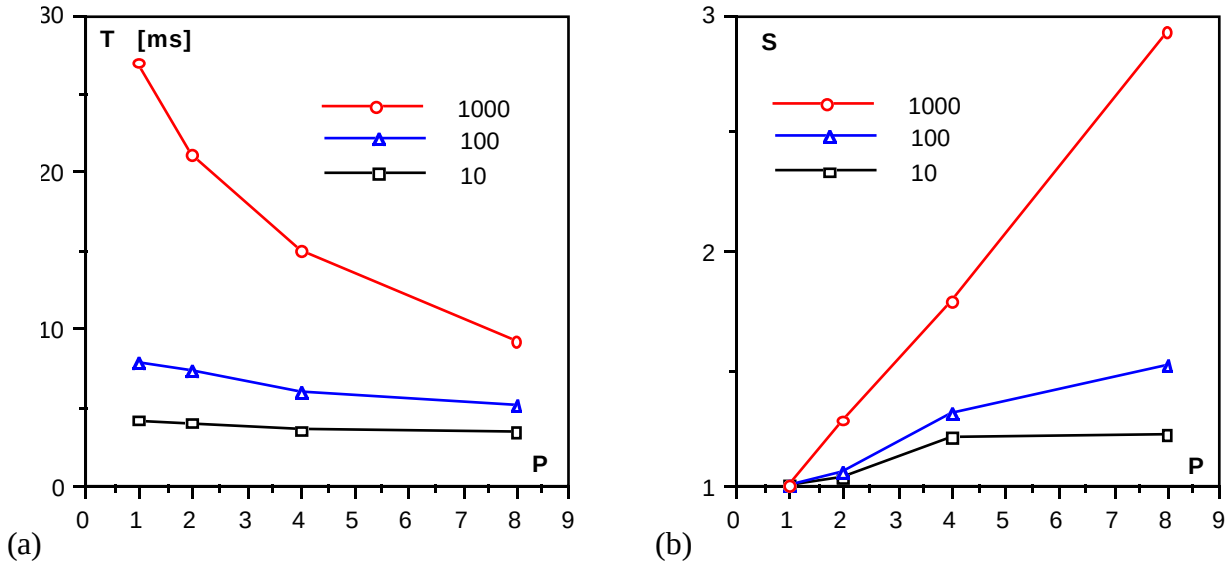


Abbildung 5-1: Experimentelle Ergebnisse der parallelen Kollisionserkennung mit P Prozessoren in Umgebungen mit ca. 10, 100 und 1000 Hindernissen (BOTTLENECK, WBK100, WBK1000): a) Laufzeiten T und b) Beschleunigungen S (Speedups) bei P Prozessoren

5.2 Parallele A*-Suche

Der Kernansatz des im Rahmen des Projektes entwickelten Bewegungsplaners baut auf dem A*-Algorithmus auf (siehe Abschnitt 4.1). Die zu verrichtende Arbeit beim A*-Algorithmus besteht in der Expansion und der anschließenden Verarbeitung von Knoten, die in der OPEN-Liste gespeichert werden. Deshalb liegt es zur Parallelisierung nahe, die Knoten aus der OPEN-Liste allen Prozessoren zugänglich zu machen, um die Arbeit auf alle Prozessoren zu verteilen. Dann bleibt nur noch die Frage, ob alle Prozessoren aus einer gemeinsamen globalen Liste mit Arbeit versorgt werden, oder ob jeder Prozessor seine eigene lokale OPEN-Liste verwaltet. Da allerdings jeder Zugriff auf eine globale OPEN-Liste bei einem nachrichtengekoppelten System einen erheblichen Kommunikationsaufwand zur Folge hat, wird hier die lokale Variante ausgewählt.

Zur Parallelisierung der A*-Suche wird eine Verteilungsfunktion *f(Θ)* definiert, die den Konfigurationsraum in *d*-dimensionale Hyperwürfel mit der Kantenlänge *b* unterteilt und diese zyklisch auf die verfügbaren Prozessoren abbildet:

$$f\left(\vec{\Theta}\right) := 1 + \sum_{i=1}^{d} \left(\frac{\theta_i}{\Delta\theta_i * b} \right) \bmod p$$

In Abhängigkeit der automatisch berechneten Gelenkdiskretisierung $\Delta\Theta_i$ wird somit jeder Konfiguration $\vec{\Theta} = [\theta_1, \ldots, \theta_d]$ eindeutig ein Hyperwürfel und damit ein Prozessor zugewiesen (siehe Abbildung 5-2). Die zweidimensionalen Würfel werden durch die Verteilungsfunktion zyklisch auf die vier Prozessoren $P_1, \ldots, P_4$ verteilt. Außerdem werden durch diese Art der Verteilung gleichzeitig die Kommunikationspfade im parallelen Algorithmus festgelegt.

Durch die Aufteilung des Konfigurationsraums in *d*-dimensionale Hyperwürfel entsteht für jeden der Prozessoren eine lokale OPEN-Liste. Durch die fehlende globale Sicht der Prozessoren kommt es allerdings vor, daß von einem Prozessor Knoten expandiert werden, die lokal gesehen zwar die niedrigsten Knoten aufweisen, global gesehen aber über den Kosten des besten Knotens liegen. Eine ausführliche Beschreibung des parallelen Bewegungsplaners ist [Sandmann97, Wurll98] zu entnehmen.

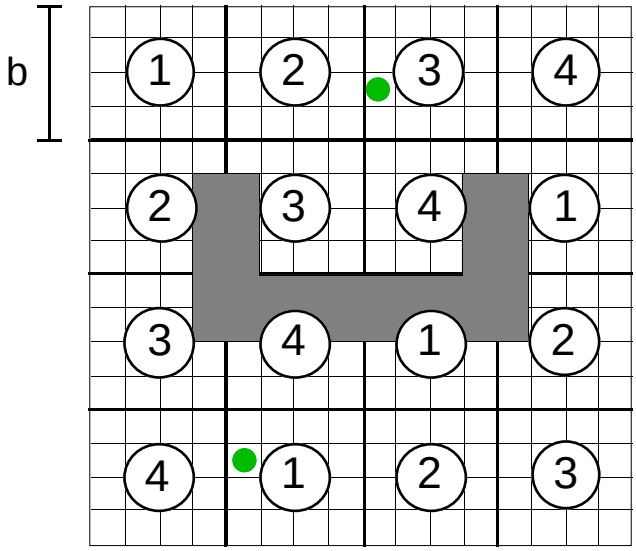


Abbildung 5-2: Aufteilung des zweidimensionalen Konfigurationsraum mit der Kantenlänge b = 2 und zyklischer Abbildung der Hypercuben auf die Prozessoren 1 bis 4

In Abbildung 5-3 sind die Laufzeiten und in Abbildung 5-4 die erreichten Speedups für die oben vorgestellten Benchmark-Probleme dargestellt. Es ist deutlich zu erkennen, daß die Parallelisierung eine Reduktion der Laufzeiten zur Folge hat und die Speedups linearares bzw. sogar superlineares Verhalten aufweisen. Die parallelen Laufzeiten der einzelnen Aufgaben liegen alle im Sekundenbereich.

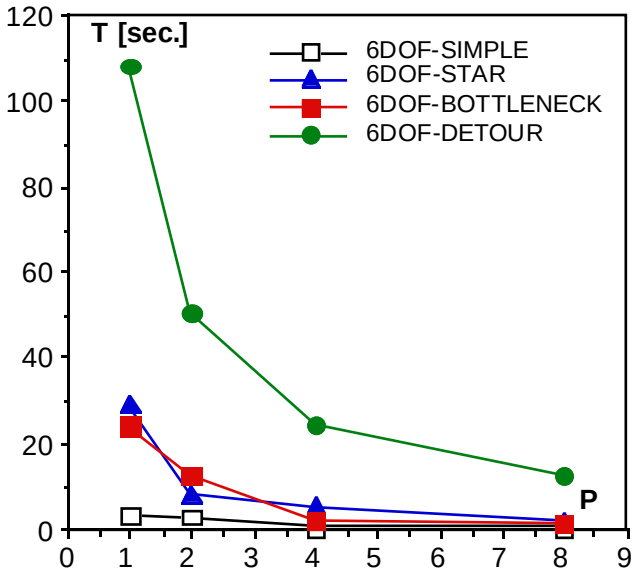


Abbildung 5-3: Laufzeit T bei P Prozessoren für die unterschiedlichen Benchmark-Probleme

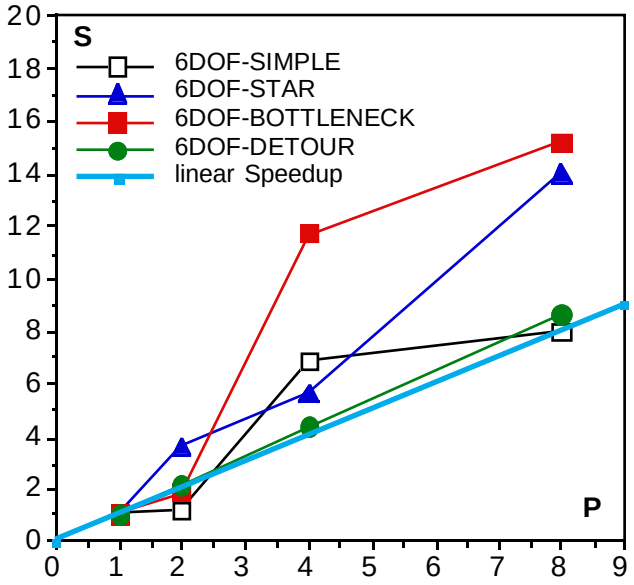


Abbildung 5-4: Speedup S bei P Prozessoren für die unterschiedlichen Benchmark-Probleme

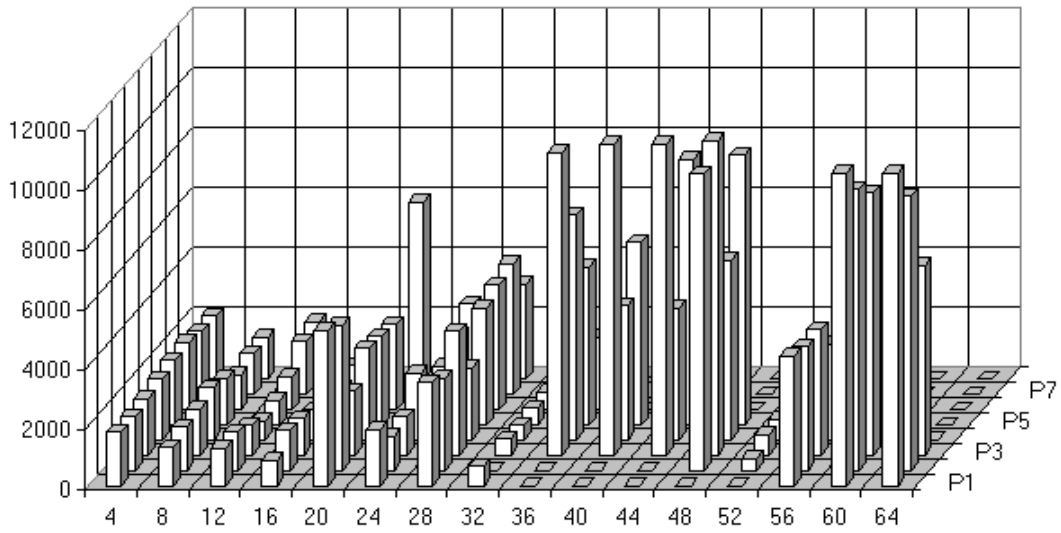


Abbildung 5-5: Anzahl der Abstandsberechnungen als Maß für die Lastverteilung in Abhängigkeit von der Kantenlänge b für die Prozessoren P_1 bis P_8 für den Benchmark STAR

5.3 Einfluß der Würfelgröße

Die Effizienz einer Parallelisierung hängt entscheidend von der Lastverteilung ab. Hierbei unterscheidet man zwischen statischer und dynamischer Lastverteilung. Während im *dynamischen* Fall die Arbeit zur Laufzeit je nach Belastung umverteilt werden kann, so legt man sich bei der *statischen* Variante zu Beginn der Planung fest. Da allerdings eine dynamische Lastverteilung pro Knoten einen im Vergleich zum Bearbeitungsaufwand, hohen Kommunikationsaufwand bedeutet, wurde für den parallelen Bewegungsplaner eine statische Verteilung gewählt. Nach Abbildung 5-5 wird für kleine Hyperwürfel eine annähernd gleichmäßige Lastverteilung erreicht, so daß der potentielle Nachteil einer statischen Verteilung nicht ins Gewicht fällt.

Neben der Umweltgeometrie hat die Größe der Hyperwürfel einen entscheidenden Einfluß auf die Lastverteilung und damit auf die parallele Suche. Wählt man eine kleine Kantenlänge *b*, dann ergibt sich eine sehr gleichmäßige Verteilung der Prozessoren auf den Konfigurationsraum, aber die Anzahl der Nachrichten nimmt stark zu. Bei großen Werten für *b* werden die Hyperwürfel entsprechend größer und es dauert länger bis Knoten an den Würfelgrenzen verschickt werden müssen. Je später aber Knoten verschickt werden, umso häufiger kommt es vor, daß die Prozessoren global gesehen unnötige Arbeit verrichten. Die Vermutung liegt also nahe, daß es eine optimale Kantenlänge *b* gibt, die gewissermaßen einen Kompromiß darstellt zwischen möglichst wenigen Nachrichten (größere Kantenlänge) und einem möglichst geringen Anteil an unnötiger Arbeit (kleinere Kantenlänge).

Für die Benchmark-Probleme STAR und DETOUR wurden die parallelen Laufzeiten Tp in Abhängigkeit von der Kantenlänge *b* eines Hyperwürfels für jeweils $p = 8$ Prozessoren gemessen. Ungünstiger Weise unterliegen die parallelen Laufzeiten für größere Hyperwürfel sehr hohen Schwankungen (siehe Abbildung 5-6). Die Gründe dafür liegen für größere Hyperwürfel an der teilweise sehr ungleichen Lastverteilung, die sich aus der unterschiedlichen Aufteilung des Konfigurationsraums unter den Prozessoren ergibt. Wie Abbildung 5-5 entnommen werden kann, ist die Anzahl der Abstandsberechnungen in der STAR-Umgebung bei kleinen Würfeln relativ niedrig und die Abstandsberechnungen sind gleichmäßig auf die acht Prozessoren verteilt. Bei größeren Würfeln hingegen steigt zum einen die Anzahl der Abstandsberechnungen stark an, zum anderen sind die Abstandsberechnungen sehr ungleichmäßig auf die einzelnen Prozessoren verteilt.

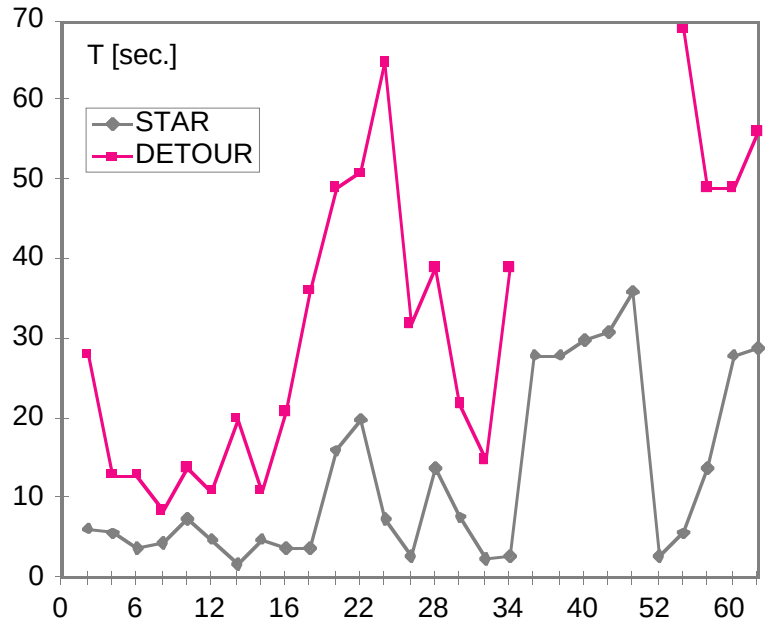


Abbildung 5-6: Laufzeiten T bei 8 Prozessoren für die Benchmark-Probleme STAR und DETOUR in Abhängigkeit der Würfelgröße b

Zusammenfassend läßt sich festhalten, daß die Wahl der optimalen Würfelgröße stark von der gewählten Umgebung abhängt und daß die parallelen Laufzeiten starken Schwankungen bei einer Vergrößerung der Würfel unterliegen. Dies macht eine analytische Vorhersage über das Laufzeitverhalten des Planers für größere Würfel schwierig. Beim aktuellen Stand der Entwicklung wurden für die weiteren Testergebnisse daher heuristisch kleine Kantenlängen (b 16) für die Aufteilung des Konfigurationsraums gewählt, da sie die besten Laufzeitergebnisse ergaben.

5.4 Bündeln von Nachrichten

Der letzte Abschnitt zeigt, daß mit kleineren Würfeln die Lastverteilung nur auf Kosten eines höheren Kommunikationsaufwands verbessert werden kann. Daher liegt die Idee nahe, nicht für jeden entfernten Knoten eine Nachricht zu generieren, sondern mehrere entfernte Knoten zu einer einzigen Nachricht zusammenzufassen.

Die Laufzeitmessungen für die Benchmark-Probleme STAR und DETOUR bei einer Kantenlänge von $b = 2$ und Prozessoren $p = 8$ zeigen allerdings, daß eine Zusammenfassung der Knoten zu einer Nachricht über mehrere Expansionen hinweg negative Auswirkungen auf die parallele Laufzeit des Bewegungsplaners hat (siehe Abbildung 5-7). Dies liegt vor allem daran, daß entfernte Knoten für einen Prozessor zu lange zurückgehalten werden. Damit expandiert ein Prozessor möglicherweise sehr kostenungünstige Knoten, weil durch die Verzögerung der Sendeoperation auf einem anderen Prozessor keine besseren Knoten zur Verfügung stehen. Deshalb wurde für alle weiteren Laufzeitmessungen auf eine Zusammenfassung von entfernten Knoten zu einer Nachricht über mehrere Expansionen hinweg gänzlich verzichtet. Außerdem nimmt bei einer Vergrößerung der Kantenlänge b die Anzahl der entfernten Knoten deutlich ab und es gibt keinen so großen Bedarf mehr für die Bündelung von entfernten Knoten.

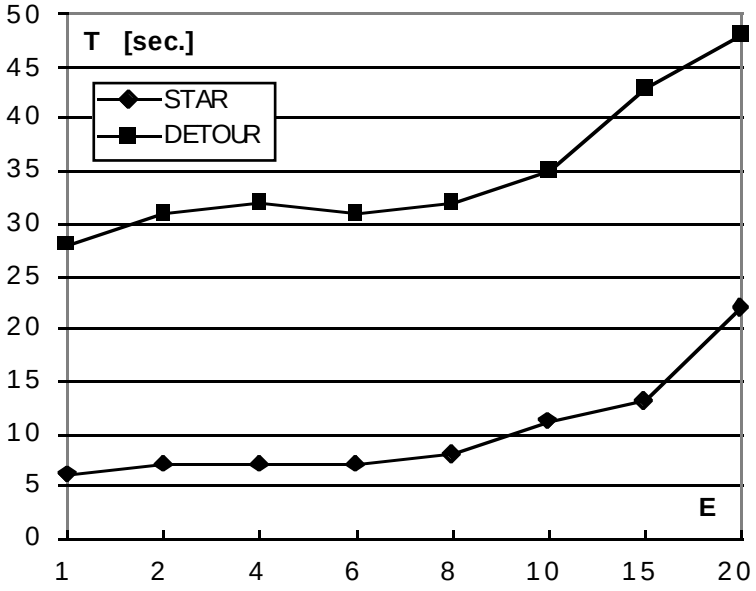


Abbildung 5-7: Laufzeiten T bei 8 Prozessoren abhängig von der Anzahl der Expansionen E pro Nachricht

6 Hierarchische Bewegungsplanung

Im Rahmen der Entwicklung des Bewegungsplaners wurde klar, daß schon bei der Suche in zweidimensionalen Umgebungen relativ viele Knoten zur Generierung eines Lösungspfades expandiert werden müssen. Für reale Anwendungen ist daher zu erwarten, daß der parallele Bewegungsplaner nicht in der Lage ist, einen Pfad in annehmbarer Zeit zu generieren. Darum ist es dringend erforderlich, zusätzlich die Anzahl der Knoten im Suchraum zu verkleinern.

6.1 Heuristische Hierachisierung

Die Anzahl der Knoten im Suchraum kann prinzipiell durch eine Vergrößerung der Gelenkschrittweite reduziert werden, da die Diskretisierung als Basis direkt in die Größe des Konfigurationsraum miteingeht (ca. 10^{15} Knoten bei 1° bzw. ca. 10^{9} Knoten bei 10° Diskretisierung). Bei einer Verdopplung der Schrittweite wird die Anzahl der Knoten im sechsdimensionalen Konfigurationsraum um den Faktor $2^6 = 64$ reduziert, bei einer Vervierfachung bereits um den Faktor $4^6 = 4096$. Eine Vergrößerung der Schrittweite birgt allerdings die Gefahr, daß der Planer bei Engpässen im Konfigurationsraum keinen Weg mehr findet, d.h. die Einsatzfähigkeit des Planers wäre auf bestimmte Umgebungen beschränkt.

Die Idee einer Vergrößerung der Schrittweite ist dennoch der Schlüssel zu einer starken Reduzierung des Suchraum. Doch anstatt die Schrittweite für den gesamten Konfigurationsraum zu erhöhen, wird sie an die Gegebenheiten der Umgebung angepaßt. Es entsteht eine variable Schrittweite, die bei kleiner Entfernung zu Hindernissen relativ klein und bei großer Entfernung zu Hindernissen relativ groß ist (siehe Abbildung 6-1).

Zur Integration dieser *heuristischen Hierarchisierung* in den vorhandenen Planer müssen folgende drei Modifikationen durchgeführt werden:

- Die Basis für die Hierarchisierung bilden *d*-dimensionale Hyperwürfel mit der Kantenlänge *b*, wie sie bei der Parallelisierung des Algorithmus eingeführt wurden. Kleinere Hyperwürfel entstehen durch sukzessive Zerteilung größerer Würfel, wobei die Kantenlänge eines Würfel

stets halbiert wird (siehe Abbildung 6-2a). Ausgehend vom größtmöglichen Würfel mit der Kantenlänge b_{max} entstehen so nacheinander Würfel mit den Kantenlänge $b_{max}/2$, $b_{max}/4$,…, $b = 1$. Der Punkt in der Mitte eines Würfels ist dabei der **Repräsentant** aller Konfigurationen, die durch den Würfel überdeckt werden. Zur einfacheren Beschreibung der Hyperwürfel wurde die Hierarchisierungsstufes eingeführt, die ein Maß für die Größe eines Würfels darstellt.

- Die notwendige Überprüfung auf bereits betrachtete Knoten im Suchraum muß auf Grund der Hierarchisierung um die **Abbildungsfunktion** eines beliebigen Knotens auf seinen Repräsentation erweitert werden. Eine solche Funktion zeigt Abbildung 6-2b, in der alle hellgrau unterlegten Konfigurationen auf den Repräsentanten eines Würfels mit der Kantenlänge b abgebildet werden.
- Mit der Einführung einer variablen Schrittweite muß auch die **Nachfolgergenerierung** entsprechend erweitert werden. Die modifizierte Operation ist derart gwählt, daß bei der Expansion eines Knotens alle Nachfolger gerade eben außerhalb des Würfels liegen (siehe Abbildung 6-2c). Durch die Abbildungsfunktion wird der Nachfolger dann seinem Repräsentanten zugeordnet.

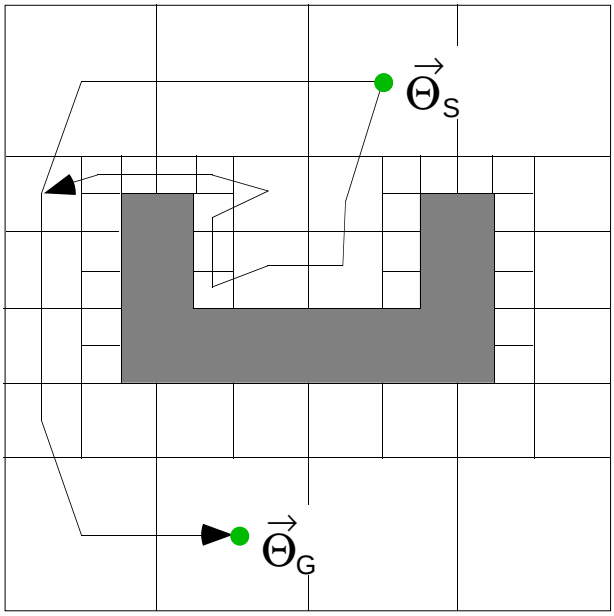


Abbildung 6-1: Hierarchische Diskretisierung des Konfigurationsraums mit zwei eingezeichneten Lösungswegen von der Startkonfiguration Θ_S zur Zielkonfiguration Θ_G

6.2 Experimentelle Ergebnisse

Abbildung 6-3b zeigt ein Beispiel, das die typischen Eigenschaften einer Hierarchisierung verdeutlicht. In der Nähe der Hindernisse werden nach wie vor viele Knoten expandiert, weil der Abstand der Knoten zu den Hindernissen nicht zur Bildung größerer Würfel ausreicht. Bei ausreichender Entfernung zu den Hindernissen können entsprechend große Würfel gebildet werden, mit der Folge, daß speziell in diesen Bereichen die Anzahl der expandierten Knoten deutlich zurückgeht. Wurden bei der gewichteten Suche (Abbildung 6-3a) noch 1216 Knoten expandiert, so sind es mit einer zusätzlichen Hierarchisierung nur noch 244.

Um die große Anzahl von Knotenexpansion in der Nähe von Hindernissen zu verhindern, muß man dafür sorgen, daß Knoten mit einem größeren Abstand zu Hindernissen bevorzugt expandiert werden (Abbildung 6-3c). Dies erreicht man am einfachsten mit einer Reduzierung der heuristischen Kosten für solche Knoten. Da die Hierarchisierungsstufe eines Knotens mit zunehmendem Abstand zu Hindernissen ebenfalls größer wird, kann man die Kostenreduzierung von der Hierarchisierungsstufe eines Knotens abhängig machen.

In dem Beispiel müssen zur Lösungsfindung lediglich noch 52 Knoten expandiert werden. Ausführliche Erläuterungen zur heuristischen Hierarchisierung und der Kostenreduzierung sind aus [Sandmann97] zu entnehmen.

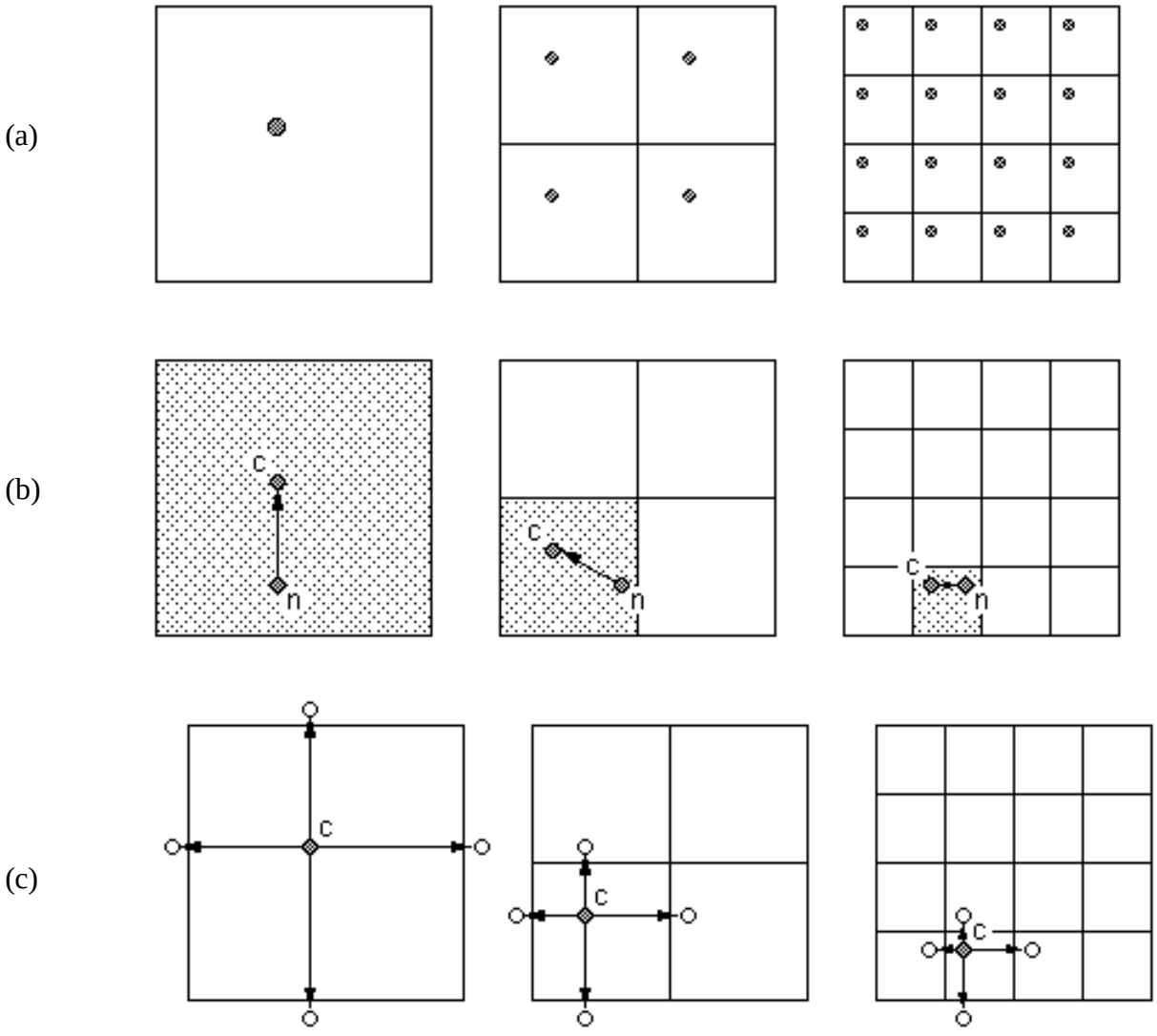
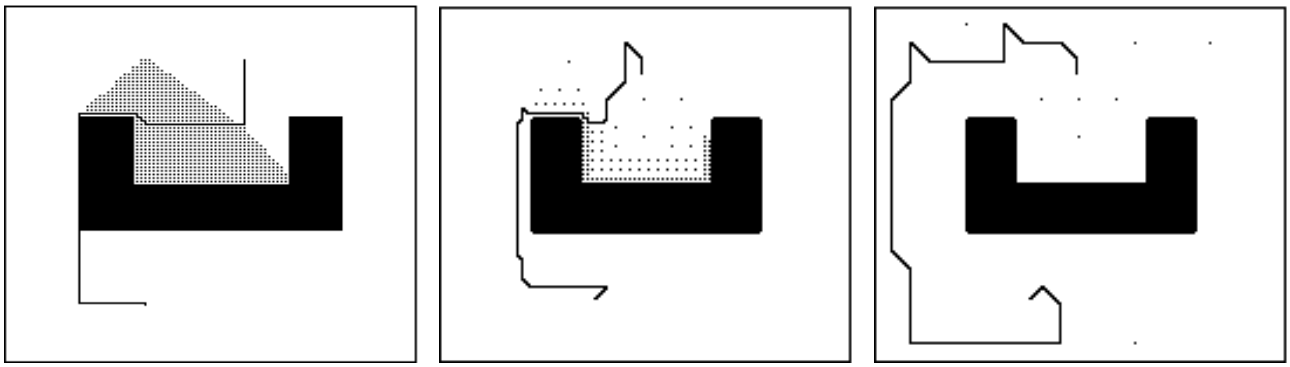


Abbildung 6-2: Graphische Darstellung der notwendigen Erweiterungen zur Einführung der heuristischen Hierarchisierung . (a) Repräsentanten der Hyperwürfel in drei aufeinanderfolgenden Hierarchisierungsstufen, (b) Abbildung eines Knotens n innerhalb eines Hyperwürfels auf den entsprechenden Repräsentanten c, (c) Modifizierte Standardoperatoren zur Bestimmung der Nachbarn in den unterschiedlichen Hierarchisierungsstufen.



(a) *(b)* *(c)*

Abbildung 6-3: Die expandierten Knoten und der Lösungspfad bei unterschiedlichen Suchvarianten: (a) äquidistante Suche, (b) heuristisch hierarchisierte Suche , (c) heuristisch hierarchisierte Suche mit Bevorzugung hoher Stufen

Die heuristische Hierarchisierung wurde im zweidimensionalen Fall auf die Benchmark-Probleme angewendet. In Abbildung 6-4 ist der Einfluß der Hierarchisierung mit Kostenreduktion für die verschiedenen Benchmark-Probleme dargestellt. Es ist deutlich zu erkennen, daß die Hierarchisierung eine starke Reduktion des Suchraumes zur Folge hat, und diese somit eine effiziente Methode zur Beschleunigung der Bahnplanung ist.

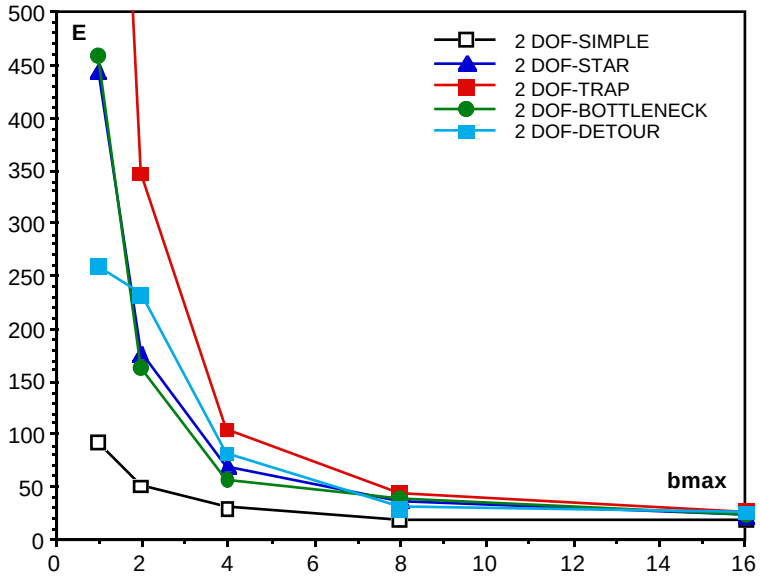


Abbildung 6-4: Anzahl expandierter Knoten E bei maximaler Würfelgröße b_{max} für die verschiedenen zweidimensionalen Benchmark-Probleme

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

Die automatisierte Bewegungsplanung für Industrieroboter in der Robotik ist ein traditionelles Problem und von großem Interesse, da ein durchschlagender Einzug in industrielle Applikationen noch nicht stattgefunden hat. Ziel des DFG-Projektes „Skalierbare Algorithmen für die parallele Bewegungsplanung in dynamischen Umgebungen“ ist es, die Bewegungsplanung industrietauglich zu ma-

chen und Algorithmen zu entwickeln, die robust und leistungsfähig genug sind, um in schnellstmöglicher Zeit eine kollisionsfreie Trajektorie zu berechnen.

Im Rahmen dieser ersten Projektphase wurde ein paralleler Bewegungsplaner für Industrieroboter entwickelt. Der Bewegungsplaner basiert auf dem bekannten A*-Algorithmus und arbeitet im impliziten, diskretisierten Konfigurationsraum, der durch die Gelenkwinkelbereiche des Roboters aufgespannt wird. Die Kollisionen werden im kartesischen Arbeitsraum durch hierarchische Abstandsberechnung im gegebenen CAD-Modell erkannt. Zur Validierung der Leistungsfähigkeit des Planers wurde ein Satz von Benchmark-Problemen konzipiert, diese in einem Robotersimulationssystem modelliert und im WWW weltweit zur Verfügung gestellt. Unterstützt durch eine optimale Diskretisierung zeigt der gut skalierbare Ansatz einen linearen bzw. superlinearen Speedup. Die Planungszeiten liegen für statische Umgebungen im Sekundenbereich. Zur Lösungsfindung benötigt der parallele Bewegungsplaner keine wesentlichen Offline-Berechnungen. Damit ist das primäre Ziel der ersten Projektphase erfüllt.

Zur weiteren Beschleunigung des Planers wurde ein erster Ansatz zur hierarchischen Suche im impliziten Konfigurationsraum erarbeitet und implementiert. Die Anwendung der heuristischen Hierarchisierung auf die zweidimensionalen Benchmark-Probleme zeigen das große Potential dieser Methode. Die Übertragung auf den sechsdimensionalen Fall sowie die Entwicklung eines vollständigen Ansatzes ist in der zweiten Projektphase vorgesehen.

Die Ergebnisse der ersten Projektphase und die Ideen zur weiteren Beschleunigung des Bewegungsplaners stellen somit eine gute Ausgangsbasis für die Fortsetzung des Forschungsvorhabens dar, um diesen in dynamischen Umgebungen einsetzen zu können.

7.2 Ausblick

Bis zum Ende der ersten Projektphase werden noch eine Reihe von Arbeiten beendet werden, zu denen zum Zeitpunkt der Berichtserstellung nur Teilergebnisse vorlagen.

In [Bordon97] wird derzeit für den entwickelten parallelen Bewegungsplaner eine zusätzliche Glättung der Trajektorie hinzugefügt. Diese ist notwendig, um die berechnete Bahn mit einem realen Roboter abfahren zu können. Durch die Kenntnis des Abstandswertes in jedem Knotenpunkt der Trajektorie ist noch genügend Spielraum vorhanden, die Knoten entsprechend eines Gütekriteriums zu verschieben. Der Vorteil dieser Vorgehensweise besteht darin, daß die Glättung zu jedem Zeitpunkt terminiert werden kann, und trotzdem eine gültige und ausführbare Lösung zur Verfügung steht. Zur weiteren Beschleunigung soll die Glättung ebenfalls parallelisiert werden, um die Rechenleistung des am Institut vorhandenen Workstation-Clusters effizient auszunutzen.

Bei der Bewegungsplanung für Industrieroboter ist neben dem Start auch immer die Zielposition gegeben. Damit ist zusätzlich zur Suche von Start zum Ziel auch eine Suchfront vom Ziel zum Start möglich. Experimente haben gezeigt, daß die Laufzeiten je nach Suchrichtung zum Teil großen Schwankungen unterliegen. Die Ursache dafür ist in der Regel ein Hindernis vom Typ TRAP, das in der einen Richtung einfach umlaufen werden kann, aber in der anderen Richtung zeitaufwendig durchsucht werden muß. Daher wird in [Beeh97] zur Zeit untersucht, inwieweit eine *bidirektionale* Suche, also eine gleichzeitige Suche vom Start zum Ziel und umgekehrt, diese Richtungsanomalien ausgleichen kann. Durch die Mehrdeutigkeit der inversen Kinematik ist sogar eine *multidirektionale* Suche, also eine Suche von einem Start zu mehreren Zielen, möglich, da eine kartesisch vorgegebene Zielposition mehrere Konfigurationen besitzen kann.

8 Literaturverzeichnis

[Adolphs92] Adolphs P.: „Echtzeifähiges Verfahren zur Kollisionsvermeidung für Industrieroboter instationärer und sich dynamisch verändernder Umgebung.“, VDI-Fortschritt-Berichte, Reihe 20, Nr. 62, VDI-Verlag, Düsseldorf, 1992.

[Beeh97] Beeh F.: „Erweiterung des parallelen Bewegungsplaners“, Diplomarbeit in Bearbeitung, Institut für Prozeßrechentechnik und Robotik, Universität Karlsruhe, 1997.

[Bordon97] Bordon U.: „Parallele Glättung von Robotertrajektorien“, Studienarbeit in Bearbeitung, Institut für Prozeßrechentechnik und Robotik, Universität Karlsruhe, 1997.

[Chakrabarti89] Chakrabarti P. P., et al., "Heuristic search in restricted memory", Artificial Intelligence, vol 41, pp. 197-221, 1989.

[Chen92] Chen P.C, Hwang Y.K.: „SANDROS: A motion planner with performance proportional to task difficulty“, IEEE Int. Conf. on Robotics and Automation, 1992.

[Dijkstra59] Dijkstra E.W.: „A note on two problems in connection with graphs“, Numerische Mathematik 1, pp. 269-271, 1959.

[Elias94] Elias J.: „Robot path planning: configuration space hierarchical representation and searching“, Intelligent Manufactoring Systems, pp. 267-271, 1994.

[Evett90] Evett M., et al., "PRA*: A memory limited heuristic search procedure for the connection machine", Proc. of the Third Symp. on the Frontiers of Massively Parallel computation (Frontiers-90), pp 145-149, 1990.

[Faverjon84] Faverjon B.: „Obstacle avoidance using an octree in the configuration space of a manipulator“, IEEE Conf. on Robotics and Automation, 1984.

[Faverjon87] Faverjon B., Tournassoud P.: „A .local approach for path planning of manipulators with a high number of degrees of freedom“, IEEE Int. Conf. on Robotics and Automation, 1987.

[Fink91] Fink B.: „Zur schnellen und kollisionsfreien Bahnplanung für Industrieroboter.“, VDI-Fortschritt-Berichte, Reihe 8, Nr. 242, VDI-Verlag, Düsseldorf, 1991.

[Gerke86] Gerke W.: „Zur Kollisionsvermeidenden Regelung von Industrierobotern.“, Dissertation, VDI-Fortschritt-Berichte, Reihe 8, Nr. 107, VDI-Velag, Düsseldorf, 1986.

[Gilbert88] Gilbert E.G., Johnson D.W., Keerthi S.S.: „A fast procedure for computing the distance between complex objects in three-dimensional space“, IEEE Journal of Robotics and Automation, vol. 4, no. 2, pp. 193-203, April 1988.

[Gontermann97] Gontermann St.: „Parallele Kollisionserkennung“, Diplomarbeit, Institut für Prozeßrechentechnik und Robotik, Universität Karlsruhe, 1997.

[Gupta96] Gupta K.: „Practical Motion Planning: An overview and state of the art“, IEEE Workshop on Practical Motion Planning in Robotics, Minneapolis, 1996.

[Gustafson88] Gustafson J. L.: "Reevaluation Amdahl's law", Communications of the ACM, vol 31, no 5, pp. 532-533, 1988.

[Hart68] Hart P.E., Nilsson N.J., Raphael B.: „A formal basis for the heuristic determination of minmum cost paths“, IEEE Trans. Syst. Sci. Cybern, pp. 100-107, 1968.

[Henrich92] Henrich D., Cheng X., "Fast Distance Computation for On-line Collision Detection with Multi-Arm Robots", IEEE International Conference on Robotics and Automation, Nice, France, May 10.-15., pp. 2514-2519, 1992.

[Henrich96] Henrich D., "Parallel processing approaches to motion planning - An overview", IEEE Int. Conf. on Robotics and Automation, Minnessota, USA, pp. 3289-3294, 1996.

[Henrich97a] Henrich D., „Fast motion planning by parallel processing - A review“, Accepted for: Journal of Intelligent and Robotic Systems, 1997.

[Henrich97b] Henrich D., Gontermann St., Wörn H.: „Schnelle Kollisionserkennung durch parallele Abstandsberechnung“ In: 13. Fachgespräch Autonome mobile Systeme (AMS´97), Stuttgart, 6. + 7. Oktober, Springer-Verlag, Reihe „Informatik Aktuell“, 1997.

[Hörmann87] Hörmann K., "Ein Verfahren zur Planung kollisionsfreier Bahnen für Industrieroboter", Dissertation, Universität Karlsruhe, 1987.

[Horowitz89] Horowitz E., Sahni S., "Fundamentals of Computer Algorithms", Computer Software Engineering Series. Sir Isaac Pitman & Sons Ltd., London, 12. Ed., 1989.

[Hwang92] Hwang Y. K., Ahuja N., "Gross motion planning – A survey", ACM Computing Surveys, vol 24, no 3, Sept. 1992.

[Hwang96] Hwang Y.K.: „Completeness vs. Efficiency in Real Applications of Motion Planning“, “, IEEE Workshop on Practical Motion Planning in Robotics, Minneapolis, 1996.

[Ishida96] Ishida T.: „Real-Time bidirectional search: Coordinated problem solving in uncertain situations“, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 18, no. 6, June 1996.

[Kamal96] Kamal L., Gupta K., del Pobil, A.P.: „Practical motion planning in robotics: Current approaches and future directions“, IEEE Robotics & Automation Magazine, Dec. 1996.

[Katz96] Katz G.: „Konzeption einer Entwicklungsumgebung unter ROBCAD für die parallele Bewegungsplanung“, Diplomarbeit, Institut für Prozeßrechentechnik und Robotik, Universität Karlsruhe, 1997.

[Kondo91] Kondo K.: „Motion planning with six degrees of freedom by multistrategic bidirectional heuristic free-space enumeration“, IEEE Trans. on Robotics and Automation, vol. 7, no. 3, pp. 267-277, June 1991.

[Korf85] Korf R. E., "Depth-first iterative deepening - An optimal admissible tree search", Artificial Intelligence, vol 27, pp 97-109, 1985.

[Korf87] Korf R. E.: „Real-Time Heuristic Search: First Results“, Artificial Intelligence, 1987.

[Korf88] Korf R. E., „Real-Time Heuristic Search: New Results“, Artificial Intelligence, 1988.

[Korf90] Korf R. E., "Real-time heurstic search", Artificial Intelligence, vol 42, pp. 189-211, 1990.

[Kumar83] Kumar V., Kanal L. N., "A general branch-and-bound formulation for understanding and synthesizing an/or-tree search procedures", Artificial Intelligence, vol 21, pp 179-198, 1983.

[Kumar94] Kumar V., et al., "Introduction to Parallel Computing. Design and Analysis of Algorithms", Benjamin/Cummings, 1994.

[Latombe91] Latombe J. C., "Robot motion planning", Kluwer Academic Publishers, 1991.

[Lozano-Pérez79] Lozano-Pérez T., Wesley M.A.: „An algorithm for planning collision-free paths among polyhedral obstacles“, Commun. ACM 22, pp. 560-570, 1979.

[Metivier90] Metivier C., Urbschat R.: „Run-time statistical analysis of a robot motion planning algorithm“, Early draft, Stanford University, ERL 445, Stanford, C.A., 1990.

[Qin96] Qin C., Henrich D.: „Path planning for industrial robot arms - A parallel randomized approach“, In Proc. of the International Symposium on Intelligent Robotic Systems (SIRS´96), Lissabon, Portugal, pp. 65-72, July 22-26, 1996.

[Quinlan93] Quinlan S, Khatib O.: „Towards Real-Time Execution of Motion Tasks“, Experimental Robotics 2, Springer, pp. 241-254, 1993.

[Reif79] Reif J.H., "Complexity of the Mover's Problem and generalizations", Proc. of the 20th IEEE Symposium on Foundations of Computer Science, pp. 421-427, 1979.

[Rembold94] Rembold U, Levi P., "Realzeitsysteme zur Prozeßautomatisierung“, Hanser Studienbücher der Informatik, 1994.

[Sandmann97] Sandmann St.: „Entwicklung eines parallelen Bewegungsplaners“, Diplomarbeit, Institut für Prozeßrechentechnik und Robotik, Universität Karlsruhe, 1997.

[Shapiro92] Shapiro, S.C. (ed.): „Encyclopedia of Artifical Intelligence“, vol. 2, pp. 1460-73, Second Edition, John Wiley & Sons, Inc., New York, 1992.

[Sen89] Sen A. K., Bagchi A., "Fast recursive formulations for best-first search that allow controlled use of memory", Proc. of the Eleventh Int. Joint Conf. on Artificial Intelligence (IJCAI-89), pp 297-302, 1989.

[Verwer90] Verwer B.J.H.: „A muliresolution work space, multiresolution configuration space approach to solve the path planning problem“, IEEE Conf. on Robotics and Automation, 1990.

[Wurll97] Wurll C., Henrich D.: „Ein Workstation-Cluster für paralleles Rechnen in Robotik-Anwendungen“, akzeptiert in: APS´97, 4. ITG / GI - Fachtagung Arbeitsplatz-Rechensysteme, Koblenz-Landau, 1997.

[Wurll98] Wurll C., Henrich H., Wörn H.: „Parallel motion planning for industrial robots in dynamic environments“, Accepted for Robotics´98, Third ASCE Specialty Conference on Robotics for Challenging Environments, Albuquerque, New Mexico, April 26-30, 1998.

[Yu95] Yu J.: „Ein kollisionsvermeidendes Roboterregelungssystem auf Transputerbasis.“, VDI-Fortschritt-Bericht, Reihe 8, Nr. 493, VDI-Verlag, Düsseldorf, 1995.