Kurzfassung

Kooperierende Manipulatoren werden in der Zukunft der Robotik eine immer größere Rolle spielen: Gemeinsames Transportieren schwerer, biegeschlaffer oder zerbrechlicher Objekte, Bearbeitungsvorgänge, die den Arbeitsraum eines Manipulators überschreiten und dadurch ermöglicht werden, daß ein zweiter Manipulator das zu bearbeitende Werkstück kontinuierlich im Arbeitsraum des ersten hält oder Aufgaben, bei denen ein Manipulator ein Objekt fixiert, während ein zweiter es manipuliert, sind interessante Aufgaben, die nur durch den Einsatz kooperierender Manipulatoren gelöst werden können. Speziell letztere, die Aufgabenklasse *Fixieren - Manipulieren* wird in den zukünftigen Anweldungsfeldern der Robotik - Dienstleistungs- und Serviceroboter - bei denen Objekte ohne dedizierte Fixiervorrichtungen in nicht auf Robotern zugeschnittenen Umgebungen manipuliert werden müssen.

Allerdings reichen herkömmliche Methoden zur Programmierung kooperierender Manipulatoren nicht mehr aus. Vielmehr sind *aufgabenorientierte* Programmiertechniken erforderlich, bei denen Aufgaben losgelöst von der Ausführungsebene der Manipulatoren beschrieben und durch entsprechende Planungswerkzeuge in eine ausführbare Form umgesetzt werden. Ein in dieser Arbeit entwickelter Formalismus dient zur *aufgabenorientierten Beschreibung von Elementaraufgaben für kooperierende Manipulatoren.* Elementaraufgaben sind dabei Aufgaben, die auf der Bewegungsebene nicht mehr sinnvoll weiter in Teilaufgaben zerlegt werden können. Dieser Formalismus bildet eine unterste Schicht, die die Ausführungsebene abschirmt. Durch ihn werden *Beziehungen* zwischen in einer Aufgabe relevanten Koordinatensystemen durch geometrische und auf Sensorinformation beruhende Bedingungen spezifiziert.

Diese Spezifikation muß unter anderem durch ein Bahnplanungsverfahren in kollisionsfreie Bahnen für die ausführenden Manipulatoren umgesetzt werden. Zur Entwicklung eines Bahnplanungsverfahrens wird zuerst die Darstellung der den betrachteten Aufgaben zugeordneten Konfigurationsräume diskutiert. Das auf diesen basierende Bahnplanungverfahren wurde in Hinblick auf effiziente Planung ohne Vorverarbeitung der Umgebung in realistischen 3D-Umgebungen entwickelt, um in ein Online-Planungssystem integriert werden zu können. Der Planer besteht aus einer lokalen Planungskomponente, die einen kollisionsfreien Weg zu einem (Zwischen-)ziel plant, und einer globalen Komponente, die den freien Konfigurationsraum (also den Raum der gültigen Manipulatorstellungen) mit einem immer dichter werdenden Netz aus freien Wegen überzieht, bis Start und Ziel der Planungsaufgabe durch einen Weg dieses Netzes verbunden werden können.

Inhaltsverzeichnis

.

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

Im Bereich der Automatisierung besteht seit längerem der Trend zur flexiblen Fertigung mit geringen Losgrößen. Im Zuge dieser Entwicklung werden zunehmend flexiblere Methoden der Fertigung erforderlich: dedizierte Vorrichtungen (etwa Fixier- oder Zuführvorrichtungen) bringen hohe Entwicklungs- und Fertigungskosten mit sich, die sich gerade bei geringen Losgrößen im Stückpreis nachteilig niederschlagen; ebenso kostet die Umrüstung von Fertigungszellen für verschiedene Produkttypen wertvolle Produktions- und Arbeitszeit. Hier sind also neue Lösungen gefordert, oben erwähnte dedizierte Vorrichtungen sind durch frei programmierbare, leicht anpaßbare Systeme zu ersetzen. Der Weg geht also von der Fertigungszelle mit einem Manipulator und verschiedenen Spezialvorrichtungen hin zur Fertigungszelle mit mehreren kooperierenden Manipulatoren.

Damit reduzieren sich zwar der Aufwand für die Neuentwicklung und die Herstellung von Spezialvorrichtungen sowie die Umrüstzeiten von Fertigungszellen. Durch die zunehmende Komplexität bei der Programmierung kooperierender Manipulatoren steigen allerdings die Softwareentwicklungskosten erheblich. Es müssen also entsprechende Konzepte entwickelt werden, die die Komplexität der Programmierung kooperierender Manipulatoren beherrschbar machen und eine effiziente Programmierung weitestgehend unterstützen.

Ein mächtiges Konzept, das diese Anforderungen erfüllen kann, ist die aufgabenorientierte Programmierung. Hierbei beschränkt sich die Programmierung von Manipulatoren auf die Definition mehr oder weniger komplexer Aufgaben (etwa “Hole Bauteil B” bis hin zu “Fertige Baugruppe X”), die dann von einem entsprechend mächtigen System selbständig in Manipulatoraktionen umgesetzt werden. Der Bereich der aufgabenorientierten Programmierung setzt sich aus mehreren Teilgebieten zusammen, von der Planung von Manipulatoraktionssequenzen aus komplexen Aufgaben bis zur Planung kollisionsfreier Trajektorien.

Gerade im letzteren Bereich steigt die Komplexität beim Übergang von einem alleinstehenden zu mehreren kooperierenden Manipulatoren durch die Bahnplanung in dynamischen Umgebungen stark an und ist mit bisherigen Werkzeugen nicht oder nur ungenügend beherrschbar.

Neben der höheren Flexibilität, die durch ihren Einsatz erreicht wird, erschließen kooperierende Manipulatoren neue Einsatzbereiche für Manipulatoren. So ist durch den Einsatz mehrerer Manipulatoren ein Transport von Objekten möglich, der mehrere weit auseinanderliegende Greifpunkte erfordert. Dies kann etwa erforderlich sein, um sperrige Objekte zu transportieren. Durch mehrere weit auseinanderliegende Greifpunkte ist ein stabiler Transport möglich. Ein weiteres Beispiel ist der Transport biegeschlaffer Teile, die ebenfalls erst durch mehrere Greifpunkte stabil transportiert werden können. Ein weiterer, neuer Einsatzbereich sind Aufgaben, die eine kontinuierliche Bearbeitung erfordern, diese aber den Arbeitsbereich eines Manipulators überschreiten würde. Dies kann etwa bei der Bearbeitung von langgestreckten Objekten der Fall sein, oder bei Bearbeitungstrajekto-

rien, die starke Orientierungsänderungen der Toolspitze des bearbeitenden Manipulators erforderlich machen. Hierbei kann ein zweiter Manipulator eingesetzt werden, um das zu bearbeitende Objekt kontinuierlich im Arbeitsbereich des bearbeitenden Manipulators zu halten.

Ein weiterer Punkt, der für die zunehmende Wichtigkeit kooperierender Manipulatoren spricht, ist deren Einsatz in Umgebungen, die nicht speziell für den Einsatz von Manipulatoren konstruiert sind. Beispiele für solche Anwendungsfelder sind der Einsatz von Manipulatoren im Weltraum [HLF94], um teure bemannte Missionen zu reduzieren, die in letzter Zeit häufiger diskutierten "Home-Robots" [Gag93], die diverse Tätigkeiten im Haushaltsbereich verrichten sollen und Service-Roboter im sozio-technischen Umfeld zur Unterstützung des Pflegepersonals in Krankenhäusern, Pflegeheimen oder in vielen weiteren außerindustriellen Bereichen [Sch94b, Asa94, Kop94]. In all diesen Bereichen sind flexible Handhabungsfähigkeiten erforderlich, ohne daß – wie im industriellen Umfeld – Spezialvorrichtungen wie Fixier- oder Zuführeinrichtungen vorhanden sind. Diese Flexibilität kann durch den Einsatz kooperierender Manipulatoren erreicht werden. Ein Objekt kann dann von einem Manipulator fixiert und in für eine Manipulation günstigen Stellungen gehalten werden, während ein zweiter Manipulator die entsprechenden Aufgaben ausführt.

1.2 Zielsetzung

Bei Manipulatoren sind verschiedene Arten der Kooperation möglich. So können mehrere Manipulatoren gleichzeitig in demselben Arbeitsraum getrennte Aufgaben ausführen, wobei eine Kooperation nur in Hinblick auf eine kollisionsfreie Ausführung der Einzelaufgaben stattfindet. Diese Art der Kooperation wird in [YZ92, ZLJ89] als punktweise Koordination (*point-wise coordination*) bezeichnet, bei der die Bewegungen mehrerer Manipulatoren bei einer begrenzten Anzahl von Punkten synchronisiert werden. Eine andere Art der Kooperation sind Aufgaben, die – wie oben erwähnt – von mehreren Manipulatoren gemeinsam durchgeführt werden, wobei feste Abhängigkeiten zwischen den Toolkoordinatensystemen der Manipulatoren auftreten. Diese machen eine enge zeitliche und räumliche Synchronisation der Manipulatoren erforderlich. Diese Art der Kooperation wird in [YZ92, ZLJ89] als trajektorienweise Koordination (*trajectory-wise coordination*) bezeichnet. In dieser Arbeit wird nur die letztere Aufgabenklasse betrachtet, der Begriff *Kooperation* bezieht sich in der Folge nur auf diese Aufgabenklasse.

Bei der gemeinsamen Manipulation von Objekten durch mehr als einen Manipulator ergeben sich neue Randbedingungen und Anforderungen gegenüber Einzelmanipulatoren. Dies sind etwa die erforderliche Kraftregelung der Manipulatoren durch bei einer gemeinsamen Manipulation auftretende Kraftschleifen. Weiterhin ist die Sicherstellung der Kollisionsfreiheit für Manipulatoren in einer dynamischen Umgebung zu erwähnen. Diese Randbedingungen erschweren eine roboterbezogene Programmierung von kooperierenden Manipulatoren und sprechen für eine von entsprechenden Werkzeugen unterstützte, komfortable

aufgabenorientierte Programmierung. Da die aufgabenorientierte Programmierung das Zusammenwirken vieler komplexer Systeme erfordert und die Programmierung kooperierender Manipulatoren ein sehr neues Gebiet darstellt, das noch viele Fragen offenläßt, wird hier eine tiefe, aber bereits von der Ausführungs- und Bewegungsebene abstrahierende Schicht betrachtet, die eine Grundlage für die aufgabenorientierte Programmierung kooperierender Manipulatoren darstellt. Es wird ein Formalismus zur aufgabenorientierten Spezifikation von *elementaren Manipulatoraktionen* für kooperierende Manipulatoren vorgestellt. Unter *elementaren Manipulatoraktionen* sind (gegebenenfalls) sensorgeregelte oder sensorgeführte Aktionen zu verstehen, für die eine weitere Zerlegung in Aktionssequenzen auf Bewegungsebene nicht mehr sinnvoll ist.

Wie bereits in der Einleitung erwähnt, ist die aufgabenorientierte Programmierung von Manipulatoren eine sehr mächtige Programmiermethode. An ein System, das diese Programmiermethode realisiert, werden allerdings hohe Anforderungen zur intelligenten Lösung komplexer Probleme gestellt.

Ein System zur aufgabenorientierten Programmierung von Manipulatoren muß unter anderem in der Lage sein, selbständig Manipulatorbewegungen zu erzeugen, die verschiedenen physikalisch oder konstruktiv bedingten Einschränkungen genügen:

- Unkontrollierte Berührungen zwischen einem Manipulator und Objekten aus seiner Umgebung bzw. mit sich selbst (im folgenden Kollisionen genannt) sind zu vermeiden. Sie können zu hohen Kräften und somit zu Störungen im vorgesehenen Ablauf oder zu mechanischen Schäden führen.
- Konstruktiv bedingt sind die Bereiche der möglichen Gelenkstellungen begrenzt. Zur Durchführung einer Aufgabe sind daher nur Manipulatorstellungen möglich, deren Gelenkstellungen sich innerhalb der manipulatorspezifischen Gelenkgrenzen befinden.

Auch fortgeschrittene, aufgabenorientiert programmierbare Systeme, wie etwa KAMRO, ein zweiarmiges Manipulatorsystem der Universität Karlsruhe [HR91, DKS91], oder der beim ROTEX-Projekt der DLR im Weltraum eingesetzte Manipulator [HBDH94], sind nur durch erhebliche Vereinfachungen in der Lage, diesen Anforderungen gerecht zu werden. So wird bei beiden Systemen davon ausgegangen, daß ein freier Arbeitsraum zur Verfügung steht, in dem die erforderlichen Manipulatorbewegungen ausgeführt werden können. So geht man bei KAMRO, der für Montageaufgaben konstruiert wurde, davon aus, daß es bei horizontalen Transferbewegungen in einer Sicherheitshöhe innerhalb eines vorgegebenen Arbeitsbereiches sowie bei vertikalen Bewegungen zum Greifen von Werkstücken zu keiner Kollision mit Hindernissen kommen kann. Beim sogenannten "Freiflieger-Experiment" im ROTEX-Projekt wird eine online geplante Trajektorie verfahren, um ein durch Bildverarbeitung lokalisiertes, freifliegendes Objekt zu greifen. Allerdings darf dieses Objekt nicht allzuweit von einer Sollposition entfernt sein, da sonst die Kollisionsfreiheit der geplanten Trajektorie nicht mehr gewährleistet ist.

Diese Annahme, daß ein Freiraum vorhanden sei, in dem kollisionsfreie Manipulatorbewegungen möglich sind, bedeutet aber eine starke Einschränkung der Aufgabengebiete für

aufgabenorientiert programmierbare Systeme. Um diese Einschränkungen zu umgehen, ist der Einsatz von Werkzeugen zur Planung von kollisionsfreien Bahnen für die eingesetzten Manipulatoren erforderlich.

Der Arbeitsraum eines einzelnen Manipulators ist – seien Kollisionen mit Hindernissen einstweilen von der Betrachtung ausgeschlossen – von seiner Geometrie bestimmt. Er ist somit für einen Manipulator fest und kann in die Programmierung von Manipulatoren einfließen, sei es implizit durch die Erfahrung und Intuition eines Roboterprogrammierers oder explizit durch Festlegung von Teilbereichen des Arbeitsraums als Freiräume, in denen Manipulatorbewegungen ohne Betrachtung der Erreichbarkeit möglich sind.

Im Gegensatz zum Arbeitsraum bei Einzelmanipulatoren ist die Menge der Stellungspaare für beide Manipulatoren, bei denen sich die Effektoren in einer vorgegebenen Stellung zueinander befinden – in der Folge als *Tooltransformation* bezeichnet –, intuitiv schwer erfaßbar, wie bereits Duelen und Kirchhoff in [DKHM87] feststellen: "Da die Erzeugung zeit- und raumsynchroner Bewegungsvorschriften für die beteiligten Kinematiken hohe Anforderungen an die Planungsintelligenz stellt, läßt sich mit dem intuitiven Geschick des Menschen keine annähernd effiziente Bewegungssynthese durchführen." Hinzu kommt, daß sich bei einer Vielzahl von Aufgaben die Relativstellung zwischen den Effektoren während der Aufgabenausführung ändert.

Den erweiterten Möglichkeiten von kooperierenden Manipulatoren steht also eine erheblich kompliziertere Programmierbarkeit solcher Systeme gegenüber. Um eine effektive Programmierung zu bewerkstelligen, sind mit dem vorgestellten Formalismus spezifizierte Elementaraufgaben durch entsprechende Planungswerkzeuge – unter anderem durch Bahnplaner – in eine von Manipulatoren ausführbare Form umzusetzen. Es wird daher ein Bahnplanungsverfahren für kooperierende Manipulatoren mit abhängigen Effektorkoordinatensystemen vorgestellt, mit dem für eine Teilklasse der mit dem im folgenden beschriebenen Formalismus spezifizierbaren Aufgaben kollisionsfreie Bahnen generiert werden.

1.3 Überblick

In Kapitel 2 werden Grundlagen dieser Arbeit dargestellt. Dies betrifft die aufgabenorientierte Programmierung von Manipulatoren, über die ein Überblick gegeben wird. Desweiteren werden Methoden zur Umweltmodellierung dargestellt, die die Grundlage einer in Kapitel 5 beschriebenen Spezifikationssprache bilden, sowie Grundlagen der Bahnplanung, die benutzt werden, um durch Bahnplanungswerkzeuge für die spezifizierten Aufgaben kollisionsfreie Bahnen zu ermitteln.

In Kapitel 3 wird ein Überblick über den Stand der Technik gegeben. Dabei werden zwei bereits realisierte aufgabenorientiert programmierbare Systeme untersucht. Desweiteren werden die aktuell zur Verfügung stehenden Methoden zur Programmierung kooperierender Manipulatoren diskutiert. Eine grundlegende Eigenschaft kooperierender Manipulatoren ist das Auftreten von Kraftschleifen. Daher wird ein Verfahren zur Spezifikation der Behandlung auftretender Kräfte und Momente untersucht. Um spezifizierte Aufgaben in

ausführbare Form umzusetzen, sind unter anderem Bahnplanungsverfahren erforderlich. Daher werden existierende Verfahren untersucht und auf ihre Anwendbarkeit für die hier gegebene Aufgabenklasse für kooperierende Manipulatoren hin diskutiert. Abschließend werden hieraus Anforderungen an eine Sprache zur Spezifikation von Aufgaben für kooperierende Manipulatoren und an Bahnplanungswerkzeuge zur Umsetzung spezifizierter Aufgaben in ausführbare Form abgeleitet.

In Kapitel 4 wird eine formale Methode zur Beschreibung von Aufgaben für kooperierende Manipulatoren dargestellt. Diese basiert auf einer Darstellung der zu spezifizierenden Aufgabe durch eine Menge von Koordinatensystemen und Beziehungen zwischen diesen. Zur Spezifikation der Beziehungen wird das Konzept der zeitabhängigen Transformationsräume eingeführt, das eine sehr mächtige, allgemeine Spezifizierbarkeit von Beziehungen zwischen Koordinatensystemen erlaubt.

Ausgehend von dieser formalen Aufgabendarstellung wird in Kapitel 5 eine Sprache zur Spezifikation der betrachteten Aufgaben entworfen.

Zur Planung kollisionsfreier, synchroner Bahnen für durch diese Sprache spezifizierte Aufgaben sind spezielle Bahnplanungsverfahren erforderlich. Die entsprechenden Planungsstrategien werden in Kapitel 6 vorgestellt. Anhand mehrerer Bahnplanungsbeispiele, die von einem implementierten Prototypen durchgeführt wurden, werden diese Konzepte verifiziert.

In Kapitel 7 wird eine Zusammenfassung der vorgestellten Konzepte und Ergebnisse gegeben sowie in einem Ausblick auf offene Forschungsgebiete und Probleme hingewiesen.

1.4 Ergebnisse

In dieser Arbeit wird eine Methode zur aufgabenorientierten Spezifikation von Elementaraufgaben für kooperierende Manipulatoren und deren Umsetzung in ausführbare Form durch ein Bahnplanungsverfahren vorgestellt. Die Spezifikation von Elementaraufgaben erfolgt dabei aus Gründen der Allgemeinheit auf einer tiefen, von der Bewegungsebene abstrahierenden Schicht.

Zur Spezifikation der Beziehungen wird das Konzept der zeitabhängigen Transformationsräume eingeführt, das eine sehr mächtige, allgemeine Spezifizierbarkeit von Beziehungen zwischen Koordinatensystemen mittels Constraints erlaubt.

In den Spezifikationsformalismus wird ebenfalls Sensorinformation zur sensorgestützten Kompensation von durch Ungewißheiten entstehenden Abweichungen und zur Terminierung von Elementaraufgaben einbezogen. Insbesondere wird die Einbeziehung von Kräften und Momenten in die Spezifikationssprache dargestellt.

Zur Umsetzung spezifizierter Aufgaben in die Bewegungsebene wird ein Bahnplanungsverfahren für kooperierende Manipulatoren vorgestellt, das den sich aus der Aufgabenspezifikation ergebenden Anforderungen genügt: Es werden, im Gegensatz zu bisherigen Bahnplanern, Ziel*räume* unterschiedlicher Dimension anstelle von Ziel*konfigurationen* behandelt.

Für die spezifizierten Beziehungen zwischen den auftretenden Koordinatensystemen ergeben sich zum einen zeitliche Abhängigkeiten. Zum anderen können Beziehungen – spezifiziert mittels geometrischer Constraints – eine unterschiedliche Anzahl von Freiheitsgraden besitzen und in einigen Freiheitsgraden auf spezifizierte Bereiche eingeschränkt sein. Die vorgestellten Planungsstrategien sind in der Lage, Zeitabhängigkeiten sowie spezifizierte geometrische Constraints zu behandeln.

Durch die Abhängigkeiten zwischen den Effektorkoordinatensystemen der Manipulatoren ergeben sich starke Einschränkungen an den freien Konfigurationsraum. Der geringe Anteil des freien Konfigurationsraums am gesamten Konfigurationsraum macht effiziente Planungsstrategien erforderlich, wie sie in dieser Arbeit beschrieben werden.

2 Grundlagen

2.1 Aufgabenorientierte Programmierung

Die Programmierung von Manipulatoren kann auf verschiedenen Ebenen erfolgen. In [Lev88] wird von der roboterorientierten Ebene die aufgabenorientierte Ebene unterschieden. Letztere wird in mehrere Stufen unterteilt:

- Objektorientiertes Programmieren abstrahiert von der geometrischen Beschreibung einer Aufgabe. Das Programmiersystem kennt die Lage der vom Programmierer symbolisch referenzierten Objekte. Beispiel: "GRASP BOLT"
- Aufgabenorientiertes Programmieren erfolgt mit einfachen natürlichsprachlichen Konstrukten, etwa "INSERT PEG INTO HOLE". Das Programmiersystem verfügt über das Wissen, diese Befehle umzusetzen.
- Programmieren mit komplexen Befehlen wie etwa "ASSEMBLE LYE PUMP". Diese Befehle werden durch entsprechende "Wissensbasen, Bewegungsplaner und Suchstrategien" [Lev88] in ausführbare Form umgesetzt.

In [Wer93] wird ein Schichtenmodell (Abb. 1) vorgeschlagen, das fünf Ebenen der Manipulatorprogrammierung unterscheidet.

Abstraktionsebenen	Funktion	Beispiel
Ebene 5: aufgabenorientiert	abstrakte, natürlichsprachliche Beschreibung der Aufgabe	"Montiere Baugruppe"
Ebene 4: objektorientiert	objektorientierte Spezifikation der Roboteraktion	"Füge Bauteil A in Bauteil B"
Ebene 3: roboterorientiert	Spezifikation des Programmablaufs in Form expliziter Roboterkommandos	"Fahre an Position P1"
Ebene 2: armorientiert	Spezifikation der Bewegung durch Folge kartesischer Bahnpunkte	(X, Y, Z, Φ, Θ, Ψ)
Ebene 1: achsorientiert	Vorgabe der Gelenkwinkel für jede Roboterachse	$(\Theta_1, \Theta_2, \Theta_3, \cdots, \Theta_6)$

Abbildung 1: Ebenen der Roboterprogrammierung (aus [Wer93])

Die Ebenen 4 und 5 werden als *implizite Programmierung* bezeichnet. "Der Programmierer muß nur noch angeben, *was* zu tun ist, während das Programmiersystem bestimmt, *wie* dies zu erreichen ist" [Wer93].

Zur impliziten Programmierung von Manipulatoren sind folgende drei Phasen erfoderlich [Wer93, Lev88, LP83a].

- Weltmodellierung
- Aufgabenspezifikation
- Umsetzung der Spezifikation in von Manipulatoren ausführbare Form

In den siebziger Jahren wurden verschiedene Systeme zur aufgabenorientierten Programmierung von Manipulatoren vorgeschlagen (AL [FTB+75], LAMA [LPW77], AUTOPASS [LW77]), aus denen sich die erforderlichen Architekturen herauskristallisierten. Ein wesentliches Resultat dieser Projekte war, daß die Komplexität der gestellten Aufgabe größer war als angenommen, weswegen keines dieser Projekte zu einer durchgängigen Implementierung gelangte [Wer93].

In den achziger Jahren wurden dann für eingeschränkte Problemklassen aufgabenorientiert programmierbare Systeme realisiert, aus deren Architektur Prinzipien für die aufgabenorientierte Programmierung kooperierender Manipulatoren abgeleitet werden können. In Kap. 3 werden zwei dieser Systeme vorgestellt und diskutiert.

2.2 Umweltmodellierung

Nach [Wer93] muß das Weltmodell eines aufgabenorientiert programmierbaren Systems folgende Anforderungen erfüllen:

- Geometrische Beschreibung aller Betriebsmittel und Objekte des Anwendungsgebiets
- Physikalische Beschreibung der Objekte
- Kinematische Beschreibung aller miteinander verbundenen Objekte
- Beschreibung der charakteristischen Merkmale der Roboter und Betriebsmittel, z.B. Gelenkbereiche, Beschleunigungsgrenzen, Sensorfähigkeiten
- Räumliche Anordnung der Objekte und Betriebsmittel, ggf. einschließlich der Unsicherheiten bezüglich ihrer Lage

Im folgenden werden zwei Methoden zur Realisierung eines Umweltmodells vorgestellt, die die Grundlage der in dieser Arbeit im Rahmen der Aufgabenspezifikation verwendeten Beschreibung der Umwelt bilden.

2.2.1 Das Framekonzept

Einen Ansatz zur Umweltmodellierung, der speziell auf die räumliche Anordnung von Objekten eingeht und sich in vielen gebräuchlichen Roboterprogrammiersprachen wiederfindet, stellt das Framekonzept dar. Jedem Objekt wird ein Ursprungskoordinatensystem zugewiesen. Die Lage, also die Position und Orientierung eines Objekts im Raum, wird durch die Beschreibung des Ursprungskoordinatensystems dieses Objekts angegeben. Die Lage dieses Koordinatensystems wird dabei durch Angabe einer aus einer Translation und einer Rotation zusammengesetzten Transformation relativ zu einem Bezugskoordinatensystem beschrieben. Da im folgenden nur von aus Rotationen und Translationen zusammengesetzten Transformationen die Rede ist, wird der Begriff *Transformation* hierfür verwendet. Transformationen werden meist durch homogene Matrizen oder Sechstupeln aus Translation und RPY- bzw. Eulerwinkeldarstellung angegeben [Pau81]. Die Zusammenfassung der Translation und Rotation und des Bezugssystems zu einer gemeinsamen Struktur wird allgemein als *Frame* bezeichnet.

Frames stellen also die Lage eines Koordinatensystems relativ zu einem Bezugskoordinatensystem dar. Das Bezugskoordinatensystem kann nun das Weltkoordinatensystem sein. In diesem Fall spricht man von einem *absoluten* Frame. Ebenso ist als Bezugskoordinatensystem ein beliebiges anderes (ebenfalls durch ein Frame beschriebenes) Koordinatensystem möglich, wobei man von einem *relativen* Frame spricht. Die Beschreibung der Lage einer Menge von Objekten durch Frames ist auf verschiedene Art und Weise in allen gängigen Roboterprogrammiersprachen realisiert.

Zusätzlich kann die Beziehung zwischen Koordinatensystemen mit Attributen versehen werden. In der Roboterprogrammiersprache AL [FTB+75] etwa werden Frames durch die Art der Verbindung zwischen den durch ein relatives Frame in Beziehung gesetzten Objekte charakterisiert, die besagt, ob ein Objekt sich mitbewegt, wenn das Objekt, zu dem ersteres durch ein Frame in Beziehung gesetzt ist, bewegt wird.

In Abb. 2 wird eine Situation durch Frames beschrieben: Ein Tisch, dargestellt durch sein Ursprungskoordinatensystem **T**, wird durch ein absolutes Frame, also in Bezug auf das Weltkoordinatensystem beschrieben. Drei weitere Gegenstände auf dem Tisch werden jeweils durch ein relatives Frame entweder in Bezug auf die Tischplatte **P** (**S** und **B1**) oder den Klotz **B1** (**B2**) beschrieben.

Aufbauend auf durch Frames geometrisch beschriebene Situationen werden prozedural Roboterprogramme erstellt, die durch entsprechende Bewegungen eine Start- in eine Zielsituation überführen. Diese Programme referenzieren die durch die Frames beschriebenen und symbolisch bezeichneten Koordinatensysteme. Beispielsweise bringt der Bewegungsbefehl `MOVE GP` das Effektorkoordinatensystem mit dem Koordinatensystem, das durch das Frame **GP** beschrieben ist, zur Übereinstimmung, bringt also den Effektor in eine Greifposition.

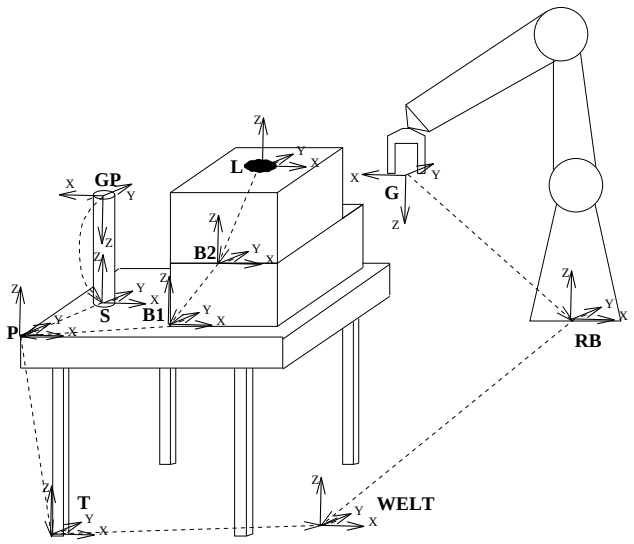


Abbildung 2: Umweltmodellierung mit Frames

2.2.2 Umweltmodellierung mit CAD-Systemen

Zur Erstellung und Verwaltung von Umweltmodellen stehen heute zwar in Teilbereichen leistungsfähige CAD-Systeme zur Verfügung, die obige Anforderungen an ein Umweltmodell aber erst teilweise erfüllen. So sind herkömmliche Simulationssysteme ([Tec92, Den92]) lediglich in der Lage, die Geometrie von Objekten zu simulieren, nicht aber physikalische Effekte. Die Modellierung physikalischer Eigenschaften wird derzeit in sich im Forschungsstadium befindenden Systemen untersucht. In dem Simulationssystem USIS [WBK94] etwa wird die Auswirkung von Gravitation auf Gegenstände simuliert [Ste92]. In [Sch94a] wurden effiziente, abstandsbildgebende Verfahren in das Simulationssystem IGRIP integriert.

Umweltmodelle aus CAD-Systemen werden entweder direkt [Fis94a] oder transformiert in ein dediziertes Umweltmodell zur Realisierung von Bahnplanungssystemen verwendet. Zur Planung von Aufgaben, die Sensorinformation berücksichtigen, ist allerdings die Simulation von physikalischen Effekten unumgänglich. In [BHHL93] wird ein Verfahren zur Simulation eines Kraft-Momenten-Sensors beschrieben, das im Robotik-Labor der DLR in das Simulationssystem KISMET [Küh90] integriert wurde. Ebenfalls ist in diesem System die Simulation von Abstandssensoren möglich. Basierend auf dieser Simulation werden aus einer Spezifikation von Sollvorgaben für Sensorwerte Regelungsvorschriften parametriert, die dann in der realen Umgebung eingesetzt werden können [BAH94].

2.3 Grundlagen der Bahnplanung

2.3.1 Konfigurationsräume und Wege

Bei der Betrachtung von Bahnplanungsproblemen verwendet man im allgemeinen eine Abbildung des realen Raums auf einen *Konfigurationsraum*, um eine Vereinfachung des Bahnplanungsproblems zu erreichen. In einem Konfigurationsraum wird ein Roboter nicht – wie im realen Raum – durch geometrische Objekte dargestellt, sondern durch einen einzigen Punkt. Da somit anstatt für ein aus mehreren aneinandergekoppelten Objekten bestehendes Objekt lediglich ein Weg für einen Punkt durch eine Hindernisumgebung zu finden ist, führt dies zu einer erheblichen Vereinfachung des Bahnplanungsproblems.

Die *Konfiguration* q eines Objekts ist die Spezifikation der Position eines jeden Punktes dieses Objekts relativ zu einem Bezugskoordinatensystem [Lat91]. Der *Konfigurationsraum* $\mathbf{C}$ eines Objekts $\mathbf{A}$ ist dann der Raum aller Konfigurationen q von $\mathbf{A}$.

Ein *Weg* τ von einer Startkonfiguration $\mathbf{q}_{Start}$ zu einer Zielkonfiguration $\mathbf{q}_{Ziel}$ ist eine stetige Abbildung

$$\tau : [0,1] \rightarrow \mathbf{C} \qquad \mathbf{C}\text{: Menge der Konfigurationen}$$

wobei gilt:

$$\tau(0) = \mathbf{q}_{Start} \qquad \text{und} \qquad \tau(1) = \mathbf{q}_{Ziel}$$

In der Realität werden zur Darstellung von Wegen meist Folgen von Konfigurationen verwendet, wobei zwei aufeinanderfolgende Konfigurationen einen hinreichend kleinen Abstand haben.

2.3.2 Das Bahnplanungsproblem

Das klassische Bahnplanungsproblem besteht darin, für einen Roboter $\mathbf{A}$ einen kollisionsfreien Weg von einer Startstellung zu einer Zielstellung durch eine Menge von Hindernissen $\mathbf{B}_1, \ldots, \mathbf{B}_n$ zu finden. Der Begriff *Stellung* bezeichnet dabei Position und Orientierung. Eine Formalisierung dieses Problems ist in [Lat91] angegeben:

> Sei $\mathbf{A}$ ein einzelnes starres Objekt – der *Roboter* –, das sich in einem euklidischen Raum $\mathbf{W}$ (dargestellt durch $\mathbb{R}^N$, mit $N = 2$ oder 3), dem *Arbeitsraum*, bewegt.
>
> Seien $\mathbf{B}_1, \ldots, \mathbf{B}_n$ unbewegliche, starre Objekte in $\mathbf{W}$, die *Hindernisse*.
>
> Seien sowohl die geometrische Beschreibung von $\mathbf{A}, \mathbf{B}_i, i \in \{1, \ldots, n\}$ als auch die Stellungen der $\mathbf{B}_i$ in $\mathbf{W}$ exakt bekannt. Weiterhin sei $\mathbf{A}$ durch keine kinematischen Beschränkungen eingeschränkt ($\mathbf{A}$ heißt *freifliegendes Objekt*).

> Das Bahnplanungsproblem ist: Gegeben seien Start- und Zielstellung (*Konfiguration*) von **A** in **W**. Erzeuge einen *Weg* τ, der Kontakte mit den Hindernissen $\mathbf{B}_i$ vermeidet, mit der Startstellung beginnt und mit der Zielstellung endet. Melde einen Fehler, wenn kein solcher Weg exisitiert.

Aus dem klassischen Bahnplanungsproblem werden verschiedene Spezialisierungen abgeleitet, unter anderem das hier betrachtete Bahnplanungsproblem für mehrere artikulierte Roboter mit abhängigen Effektorkoordinatensystemen.

2.3.3 Darstellung des Konfigurationsraums für artikulierte Roboter

Artikulierte Roboter bestehen aus mehreren durch Gelenke verbundenen, festen Objekten. Eine Abbildung des realen Raums in den Konfigurationsraum muß also gewährleisten, daß eine Konfiguration jeden Punkt des artikulierten Roboters im realen Raum eindeutig beschreibt. Weiterhin sollte sie so geartet sein, daß der verwendete Konfigurationsraum von möglichst niedriger Dimension ist, um den Suchraum für die Bahnplanung möglichst klein zu halten.

Zur Repräsentation von Konfigurationsräumen für einzeln eingesetzte Roboter finden sich in der Literatur zwei verschiedene Darstellungsarten:

Gelenkparameter Hierbei wird der Konfigurationsraum des Roboters durch seine Gelenkparameter aufgespannt, jedem seiner Gelenke ist eine Achse im Konfigurationsraum zugeordnet. Besteht also ein Roboter **A** aus den n Gelenken $\mathbf{J}_1, \ldots, \mathbf{J}_n$ mit den Wertebereichen $W_i = [J_i^{min}, J_i^{max}] \subset \mathbb{R}$, so gilt für den Konfigurationsraum **C**:

$$\mathbf{C} = W_1 \times \ldots \times W_n \subset \mathbb{R}^n$$

Diese Methode wird etwa in [Gla91b, Gla90, Gla91a] verwendet. Diese Darstellungsart ist an der Roboterkinematik orientiert, eine eindeutige Darstellung ist direkt gegeben. Durch diese Art lassen sich auch Konfigurationsräume von redundanten Robotern leicht darstellen.

Stellung des Effektorkoordinatensystems Bei dieser Darstellungsart wird der Konfigurationsraum des Roboters durch Position und Orientierung des Effektorkoordinatensystems aufgespannt. Bewegt sich der betrachtete Roboter in einem dreidimensionalen Arbeitsraum, so sind zur Darstellung sechs Parameter, je drei für die Position und die Orientierung, erforderlich. Zusätzlich ist hierbei impliziert, daß die Stellung aller Robotergelenke durch die Stellung seines Effektorkoordinatensystems eindeutig bestimmt ist. Dies ist allerdings für viele nichtredundante Roboter nicht der Fall. Die Eindeutigkeit wird hierbei erst durch die Angabe eines *kinematischen Zustands* [Hör88] erreicht.

Die möglichen kinematischen Zustände eines einfachen Linienroboters sind in Abb. 3 verdeutlicht. Zu einer gegebenen Effektorposition sind die beiden Gelenkwinkelkonfigurationen (α_1, β_1) und (α_2, β_2) möglich. Durch die Angabe eines kinematischen Zustands $\mathbf{k} \in \{$`above, below`$\}$ kann eindeutig bestimmt werden, welche der beiden möglichen Gelenkwinkelkonfigurationen gemeint ist. Die kinematischen Zustände etwa eines PUMA560 werden durch ein Tripel **(arm,elbow,wrist)** beschrieben, wobei jedes Element zwei Zustände annehmen kann.

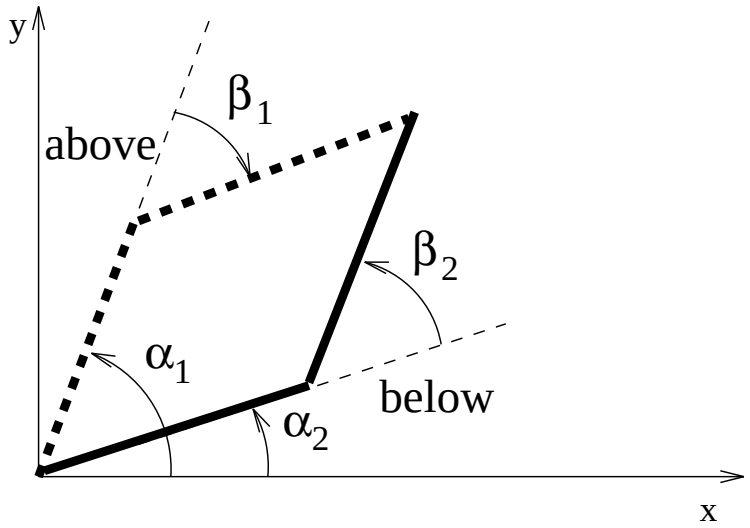


Abbildung 3: Kinematische Zustände

Diese Art der Darstellung des Konfigurationsraums wird etwa in einem von Hörmann [Hör88] vorgestellten Planungsverfahren verwendet.

Bei diesem Verfahren ist zusätzlich der kinematische Zustand mitzubetrachten. Dies bewirkt eine Erhöhung der Dimensionalität des Konfigurationsraums. Dieser vermeintliche Nachteil wird aber dadurch ausgeglichen, daß wegen der Mehrdeutigkeit der inversen Kinematik zu in kartesischen Koordinaten gegebenen Start- und Zielstellungen mehrere Gelenkparametersätze existieren. Diese müssen durch Planer, die im Gelenkparameterkonfigurationsraum arbeiten, durch zusätzliche Planungsläufe in ihrer Kombinatorik betrachtet werden.

Nachteilig ist allerdings, daß wegen der inversen Kinematik bei der Planung für den Manipulator ein Konfigurationsraumwechsel nicht – wie bei der Gelenkparameterdarstellung – auf triviale Art und Weise erfolgen kann.

Bei diesem Verfahren entstehen im allgemeinen für das betrachtete Koordinatensystem (den TCP bzw. das Objektkoordinatensystem des transportierten Objekts) kurze Bahnen. In vielen Fällen (Transport schwerer Objekte, gemeinsamer Transport durch mehrere Manipulatoren, wie u.a. in dieser Arbeit betrachtet) stellen kurze, gerade Bahnen für dieses Koordinatensystem einen Vorteil dar gegenüber kurzen Bahnen im Gelenkparameterraum,

die häufig zu verschlungenen, längeren Bahnen für den TCP führen.

3 Stand der Technik

In diesem Kapitel werden zunächst zwei Implementationen aufgabenorientiert programmierbarer Systeme analysiert. Danach werden Methoden zur Programmierung kooperierender Manipulatoren vorgestellt und diskutiert.

Desweiteren ergeben sich durch die erweiterte Funktionalität kooperierender Manipulatoren neue Anforderungen an die Aufgabenspezifikation: durch die gemeinsame Manipulation von Objekten durch zwei Manipulatoren treten geschlossene kinematische Ketten auf, was aus mechanischer Sicht das Auftreten von Kraftschleifen bedeutet. Daher wird ein Spezifikationsansatz zur Behandlung von Kräften und Momenten (*nachgiebige Bewegung, Compliant Motion*) betrachtet.

Im Anschluß hieran werden existierende Ansätze zur Bahnplanung beschrieben und diskutiert, inwieweit sie den Anforderungen gerecht werden, die mit der hier vorgestellten Sprache spezifizierten Aufgaben für kooperierende Manipulatoren in ausführbare Bahnen umzusetzen.

Aus diesen Betrachtungen werden abschließend Anforderungen an eine Sprache zur Spezifikation von Aufgaben für kooperierende Manipulatoren und an Bahnplanungsverfahren zur Umsetzung dieser in verfahrbare Bahnen abgeleitet.

3.1 Aufgabenorientiert programmierbare Systeme

3.1.1 Fallstudie: Mauerbau

Eine Realisierung eines aufgabenorientiert programmierbaren Systems erfolgte am Institut für Informatik der Technischen Universität München im Rahmen des SFB 331, *Informationsverarbeitung in autonomen, mobilen Handhabungssystemen* [33194]. Hierbei baut ein Manipulator auf einer drehbaren Scheibe eine Mauer aus Holzklötzen nach einem zuvor eingegebenen Umriß der Mauer auf [Fis88, Boc90].

Die Architektur dieses aufgabenorientiert programmierbaren Systems ist in Abb. 4 dargestellt. Dabei wird eine generelle Architektur für flexible Fertigungsumgebungen für den Speziallfall Mauerbau adaptiert.

Eine *globale Produktionssteuerungsschicht* erhält als Eingabe den Plan einer zu bauenden Mauer. Aus diesem und dem Bestand eines Bausteinlagers wird ein konkreter Bauplan erzeugt, in dem die Art und Position der verwendeten Bausteine festgelegt wird.

Dieser Plan wird an die darunterliegende *lokale Produktionssteuerungsschicht* übergeben. In dieser Schicht wird der Plan in eine Sequenz von Aktionen der Art "Nimm Klotz <X> und lege ihn an Position <P>" transformiert, die Ausführung dieser Aktionen einzeln angestoßen und die Statusmeldungen der Einzelaktionen nach ihrer Ausführung ausgewertet. In dieser Ebene sind zur Erhöhung des Autonomiegrads des Systems Fehlerbehandlungsstrategien realisiert. Diese werden je nach Status der zuletzt beendeten Teilaufgabe angestoßen.

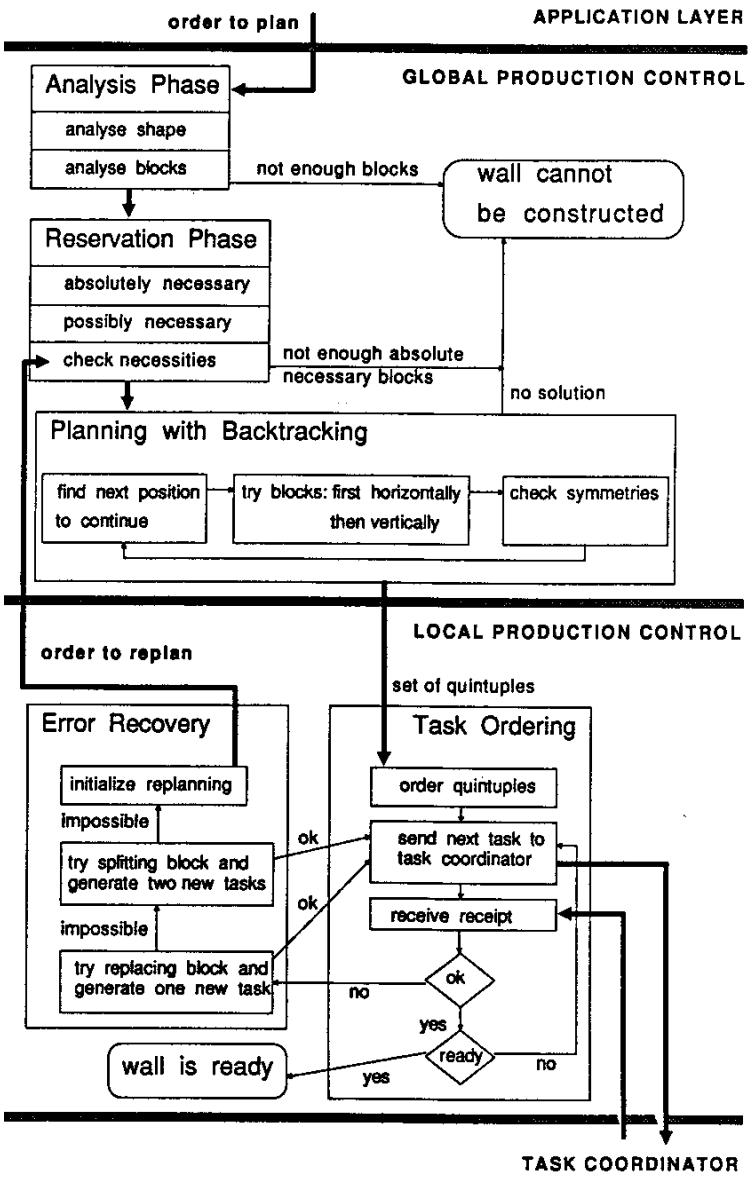


Abbildung 4: Systemarchitektur MAUERBAU (aus [Boc90])

Als mögliche Fehler sind der Verlust eines Blocks während des Transports bzw. das Fehlen eines Blocks im Lager modelliert. Kann ein Fehler nicht lokal behoben werden (etwa durch Verwendung eines Ersatzsteins), kann eine Neuplanung der Aufgabe mit verändertem Lagerbestand und teilweisem Abbau der Mauer in der *Globalen Produktionssteuerungsschicht* veranlaßt werden.

Die *Lokale Produktionssteuerungsschicht* übergibt die einzelnen Aufgaben an den Aufga-

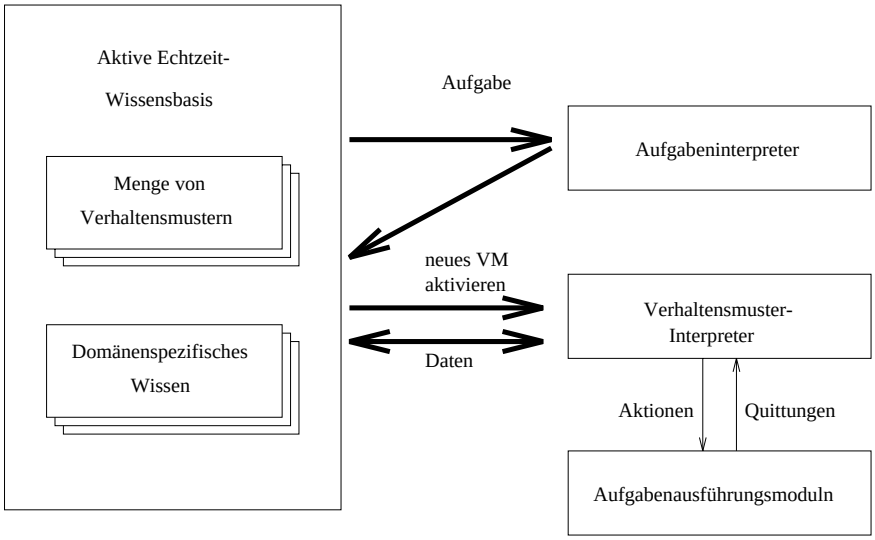


Abbildung 5: Struktur der Aufgabenkoordinationsschicht (aus [Boc90])

benkoordinator (Abb. 5). Der Aufgabenkoordinator besteht aus einem Aufgabeninterpreter und einem *Verhaltensmusterinterpreter*. Der Aufgabeninterpreter bildet die Schnittstelle für aufgabenorientiert beschriebene Aktionen aus den oberen Schichten. Er analysiert die ihm übergebenen Aktionen, ermittelt zugehörige Verhaltensmuster und übergibt diese an den Verhaltensmusterinterpreter.

Ein Verhaltensmuster beschreibt, wie eine aufgabenorientiert beschriebene Aktion ("nimm <Objekt> und lege es nach <Stellung>", "Kalibriere <Einheit>", "Abbruch", "Pause", "Weiter", "Gib Status") in *primitive Aktionen* (Greif-, Bewegungs- und Synchronisationsbefehle) umzusetzen ist. In den Verhaltensmustern sind neben den auszuführenden primitiven Aktionen auch Aufträge an die verschiedenen Planungs- und Verwaltungsinstanzen und Fehlerbehandlungsstrategien modelliert.

Der Verhaltensmusterinterpreter setzt die ihm übergebenen Verhaltensmuster in Aufträge an die entsprechenden Planungsinstanzen und Aufgabenausführungsmodule um (hier ein Manipulator und eine drehbare, programmgesteuert start- und anhaltbare Scheibe). Um etwa eine Aktion der Art "Nimm Klotz <X> und lege ihn an Position <P>" in primitive Aktionen aufzuspalten, wird zuerst die augenblickliche Position von <X> aus einer Lagerverwaltungsinstanz besorgt und ein einfacher Feinplaner mit der Planung einer Bewegung zur Ablage des Klotzes an Position <P> beauftragt. Dann erzeugt ein einfacher Bahnplaner eine kollisionsfreie Bewegung für den Transport des Klotzes. Aus der durch die Planer zur Verfügung gestellten Information wird dann eine entsprechende Sequenz von primitiven Aktionen geplant.

In diesem Fallbeispiel ist eine wegen folgender Aspekte sehr interessante Architektur für aufgabenorientiert programmierbare Systeme realisiert:

- Komplexe Aufgaben werden in eine Folge von fest definierten *primitiven Aktionen* transformiert. Diese werden unter Einbeziehung von verschiedenen Planungswerkzeugen in für die Exekutivebene ausführbare Form umgesetzt.

- Zur Realisierung der Reaktion auf Umweltereignisse werden Statusrückmeldungen bei der Ausführung *primitiver Aktionen* durch die Exekutivebene berücksichtigt. Diese Fähigkeit ist grundlegend für die Autonomie eines aufgabenorientiert programmierbaren Systems.

- Da nur *Pick-and-Place*-Aufgaben betrachtet werden und die Synchronisation zwischen verschiedenen Ausführungseinheiten sich nur auf wechselseitiges Warten und nicht – wie bei den in dieser Arbeit betrachteten Aufgaben für kooperierende Manipulatoren – auf zeitlich synchrone, sequentialisierte Zusammenarbeit bezieht, werden keine Mechanismen zur *gleichzeitigen* synchronen Bewegungsausführung benötigt. Somit reicht die Verwendung von in herkömmlichen Roboterprogrammiersprachen vorhandenen Befehlen aus.

3.1.2 KAMRO

Eine sehr weit fortgeschrittene Architektur für ein Manipulatorsystem, das aufgabenorientiert programmiert werden kann, wird in [HR91, DKS91] beschrieben. An der Universität Karlsruhe wird seit 1985 ein Zweiarmmanipulator entwickelt, der in der Lage ist, autonom Montageaufgaben zu lösen. Der Zweiarmmanipulator besteht dabei aus zwei Manipulatoren des Typs PUMA 260, die – kopfüber hängend – auf einem kartesisch steuerbaren Wagen montiert sind.

Die Robotersteuerung besteht aus einem Online-Planungssystem zur Aufgabentransformation und einer Echtzeit-Robotersteuerung. Diese beiden Systeme befinden sich jeweils auf einem eigenen Rechnersystem, die über ein lokales Netzwerk miteinander verbunden sind. Daneben befindet sich in der Robotersteuerung noch ein Bildverarbeitungssystem, das die Objekterkennung und -lokalisation leistet. Als Eingabe erhält dieses System einen Aktionsplan in Form eines Präzedenzgraphen. Die Aufgabe des Online-Planungssystems ist es nun, diesen impliziten Aktionsplan zu interpretieren, das Bildverarbeitungssystem zu aktivieren, wenn die Position eines zu montierenden Bauteils unbekannt ist, und Sequenzen von expliziten Manipulatoroperationen (dieser Begriff wird im nächsten Abschnitt erklärt) zu erzeugen, die dann an die Robotersteuerung gesendet werden.

Die Knoten des Aktionsplans sind einfache, symbolisch beschriebene Teilaufgaben, die als *Implizite Elementaroperationen (IEO, implicit elementary operation)* bezeichnet werden. Beispiele hierfür sind die IEO's PICK (Aufnehmen eines Bauteils), PLACE (Montieren eines Bauteils), EXCHANGE (Übergeben eines Bauteils zwischen den beiden Manipulatoren) und REGRASP (Umgreifen). Da die Ausführung von Manipulatoraktionen vom aktuellen Umweltstatus (etwa der Position von Werkstücken) abhängig ist, werden durch diese IEO's nur die Ziele der Manipulatoraktionen vorgegeben. Diese werden dann durch

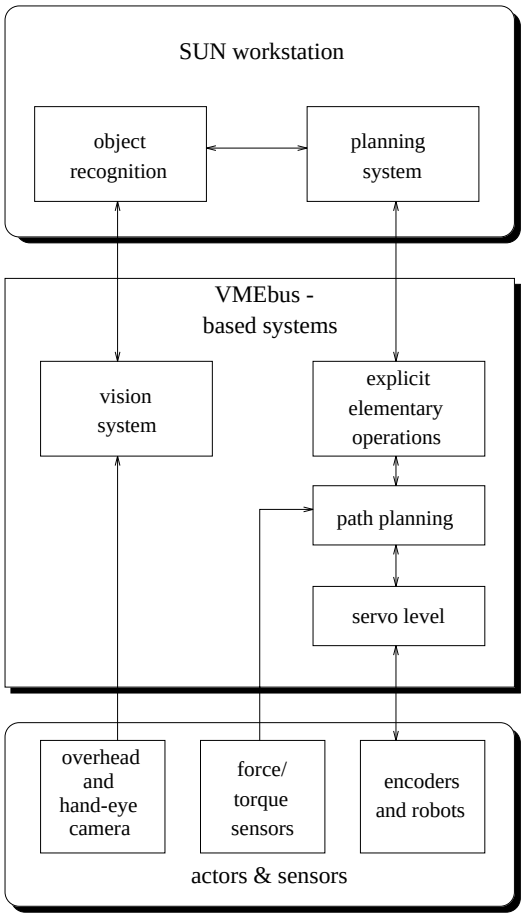


Abbildung 6: Systemarchitektur KAMRO

das Online-Planungssystem in Sensoraktionen zur Bestimmung des Umweltstatus und, parametrisiert mit den Ergebnissen der Sensoraktionen, Sequenzen von sensorgeführten bzw. sensorüberwachten, geometrisch beschriebenen Elementaroperationen umgesetzt [DKS91]. Die Schnittstelle zwischen dem Online-Planungssystem und der Robotersteuerung wird durch diese sogenannten *Expliziten Elementaroperationen (EEO, explicit elementary operation)* gebildet.

Beispiele für EEO's sind etwa TRANSFER (horizontale Bewegung), FINEMOTION (vertikale Bewegung) oder JOIN (Montagebewegung). Sie sind in der Lage, automatisch komplexe, sensorgeführte Aufgaben auszuführen und in begrenztem Umfang auf unvorhergesehene Ereignisse zu reagieren. So wird etwa bei der EEO JOIN beim Einsetzen eines Stifts in ein Loch erst eine vertikale Bewegung (*coarse search motion*) bis zum Auftreten einer bestimmten Kraft durchgeführt. Läßt die Z-Position des Stifts nun darauf schließen, daß der Stift korrekt eingesetzt wurde, ist die EEO korrekt beendet. Ansonsten wird eine kreisförmige tastende Suchbewegung (*fine search motion*) mit einer leichten Andruckkraft in Z-Richtung

durchgeführt, bis sich diese Kontaktkraft sprunghaft verringert (was auf das Vorhandensein der gesuchten Fügestelle hindeutet). Dann wird eine in X- und Y-Kraftrichtung sowie in X-, Y- und Z-Momentenrichtung sensorgeregelte Bewegung in Z-Richtung ausgeführt, bis die Kraft in Z-Richtung sprunghaft ansteigt, der Stift also korrekt eingesetzt werden konnte.

Eine EEO kann also wiederum in eine Sequenz von elementaren Manipulatorbewegungen, etwa die einzelnen Suchbewegungen, aufgeteilt werden, die durch einen dedizierten Onlineplaner – in der KAMRO-Terminologie als *Elementaroperationsmodul (EOM, elementary operation module)* bezeichnet – unter Berücksichtigung von Sensorinformation ausgeführt werden.

Die eigentlichen elementaren Manipulatoraktionen treten hier also als Bestandteile der EEO's auf. Neben einfachen Bewegungsbefehlen (EEO TRANSFER) können dies auch sensorgeregelte Operationen (nachgiebiges Einsetzen eines Stifts) oder sensorüberwachte Operationen (Bewegung nach unten, bis eine bestimmte Kraft auftritt, dann abbrechen) sein. Außerdem steht dem dedizierten Onlineplaner (EOM) Statusinformation nach der Ausführung einer elementaren Manipulatoraktion zur Verfügung (etwa die exakte Position beim Abbruch einer sensorüberwachten Bewegung).

Zusammenfassend kann festgestellt werden, daß an der Schnittstelle zur Ausführungsebene – einem Prinzip der Informatik folgend, demzufolge komplexe Probleme in möglichst einfache, elementare Einheiten zerlegt werden – ebenfalls *Elementaroperationen* verwendet werden. Diese werden im KAMRO-System durch eigene, parametrisierbar aufrufbare Module realisiert.

Für eine neu einzuführende Aufgabenklasse ist also jeweils ein neues Modul zu implementieren, eine Spezifikation allgemeiner Aufgaben und deren automatische Umsetzung ist somit nicht möglich. So wird beispielsweise die EEO JOIN durch eine spiralförmige Suchtrajektorie (s. Abb. 22, S. 74) mit nachfolgendem senkrechten Einsetzen realisiert. Alternative Strategien (etwa Suchen mit schräg gehaltenem Bolzen und Aufrichten des Bolzens nach dem Finden der Passung) können nicht spezifiziert werden, hierzu ist eine eigene EEO zu implementieren.

Eine allgemeine Spezifizierbarkeit von Elementaroperationen wird durch die in dieser Arbeit vorgestellte Spezifikationsmethode ermöglicht. Ebenso ist bei KAMRO momentan die Kooperation der beiden Manipulatoren auf eine exklusive Freiraumbenutzung bzw. eine punktweise Synchronisation bei Umgreifaufgaben beschränkt. Die Beschränkung der exklusiven Freiraumnutzung ist für kooperierende Manipulatoren nicht tragbar. Hierfür sind Bahnplanungsverfahren erforderlich, die synchrone, kollisionsfreie Bewegungen für mehrere Manipulatoren in einem gemeinsamen Arbeitsraum generieren. Ein derartiges Verfahren wird im Rahmen dieser Arbeit vorgestellt. In neueren Arbeiten [GM91, LMT91] wurde eine hybride Positions-Kraft-Regelung realisiert, die die Grundlage für die gemeinsame Ausführung von Aufgaben durch beide Manipulatoren bildet, zudem aber neue Bahnplanungswerkzeuge erforderlich macht. Die Tatsache, daß für eine neue Aufgabe EEO's neu implementiert werden müssen, stellt für zukünftige Anwendungen als aufgabenorientiert

programmierbares System eine starke Einschränkung dar. Lediglich bestimmte, bereits vorgesehene und speziell implementierte Elementaroperationen können ausgeführt werden. Wichtig wäre allerdings die Fähigkeit, die *vollständigen* Fähigkeiten eines Zweiarmmanipulators etwa als Spezifikationsschnittstelle an die Online-Planungsschicht zur Verfügung zu stellen.

3.2 Roboterorientierte Programmierung kooperierender Manipulatoren

Zur Programmierung kooperierender Manipulatoren ist deren Synchronisation erforderlich. In mehreren Arbeiten wurden hierbei grundlegende Konzepte vorgestellt. Anschließend wird die Einbettung dieser Konzepte in eine roboterorientierte Programmiersprache beschrieben und die damit verbundenen Probleme diskutiert.

3.2.1 Synchronisation kooperierender Manipulatoren

Die Kooperation von Manipulatoren in industriellen Anwendungen erfolgt überwiegend durch die Ausführung getrennter Aufgaben durch mehrere Manipulatoren in gemeinsamen Arbeitsräumen. Dabei werden die Manipulatoren derart synchronisiert, daß der gemeinsame Arbeitsraum jeweils exklusiv benutzt wird. Somit kann jeder der Manipulatoren ebenso wie ein alleinstehend eingesetzter Manipulator (mit Ausnahme der Synchronisation) ohne Berücksichtigung seiner Kooperationspartner programmiert werden. Einen Schritt weiter gehen Online-Kollisionsvermeidungsstrategien [FH87, HGB94, FM94], bei denen programmierte Punkt-zu-Punkt-Bewegungen online dergestalt modifiziert werden, daß eine kollisionsfreie Benutzung eines gemeinsamen Arbeitsraums ermöglicht wird. Dadurch ist eine unabhängige Programmierung mehrerer einen gemeinsamen Arbeitsraum benutzender Manipulatoren möglich, die Vermeidung von Kollisionssituationen erfolgt zur Laufzeit.

Grundlegende Synchronisationskonzepte, die über die gemeinsame Arbeitsraumbenutzung hinausgehen und eine gemeinsame Aufgabenausführung durch mehrere Manipulatoren ermöglichen, sind das Master-Slave-Konzept [PB90] und das Konzept von synchronen Trajektorien [DKHM87]. Beim Master-Slave-Konzept wird ein Manipulator zum sogenannten *Master* erklärt. Das Bewegungsprogramm des Masters wird – wie von alleinstehenden Manipulatoren gewohnt – relativ zu einem festen Bezugskoordinatensystem verfahren. Das Bewegungsprogramm der sogenannten *Slaves* hingegen wird relativ zum Effektorkoordinatensystem des Masters verfahren. Hintergrund dieses Konzepts ist die Vereinfachung der Programmierung kooperierender Manipulatoren dadurch, daß das die eigentliche Manipulation ausführende Programm der Slaves getrennt von der Bewegung des Masters in festen Objektkoordinaten (des vom Master gehaltenen Objekts) realisiert werden kann. Das Problem, Bewegungen für die Manipulatoren zu finden, die kollisionsfrei und innerhalb der Gelenkgrenzen verfahren werden können, bleibt im Master-Slave-Konzept allerdings bestehen.

Das Konzept der "synchronen Bewegungsvorschriften" [DKHM87] (im folgenden als *synchrone Trajektorien* bezeichnet) dient ebenfalls der Programmierung kooperierender Manipulatoren zur Ausführung einer gemeinsamen Aufgabe. Dabei werden lediglich vorgegebene Trajektorien synchron gestartet und ebenfalls zeitlich synchronisiert verfahren. Die Trajektorien sind dabei in einem festen Koordinatensystem angegeben. Wegen seiner Symmetrie ist dieses Konzept gut für die Ausführung von in Planungsinstanzen erzeugten Trajektorien geeignet und wird für die Ausführungsschnittstelle des hier vorgestellten Bahnplanungsverfahrens verwendet.

3.2.2 Spracherweiterungen für kooperierende Manipulatoren

In [Tsa91] beschreibt Tsai Konzepte zur Erweiterung der von GMFanuc Robotics verwendeten Sprache RAIL [FV82, Tsa91] für kooperierende Roboter. Hierbei wird von einer einzigen mehrprozeßfähigen Steuerung für mehrere Roboter ausgegangen. Es werden als Erweiterung des **MOVE**-Befehls sogenannte Gruppenbewegungen eingeführt. Diese können einem von drei Typen angehören:

- Simultan

  Die Bewegungen der einzelnen Roboter beginnen und enden zum selben Zeitpunkt.

  Beispiel:

  ```
  VAR  robot_approach_path : path in group 1
       table_pos_1 : path in group 2

  ...

  MOVE ALONG robot_approach_path,
       ALONG table_pos_1, SIMULTANEOUS
  ```

- Unabhängig oder überlappend

  Die Bewegungen mehrerer Roboter erfolgen nebenläufig ohne feste Beziehungen zwischen Start- und Endzeitpunkten.

  Beispiel:

  ```
  ...

  MOVE ALONG robot_approach_path NOWAIT

  ...

  MOVE ALONG table_pos_1 NOWAIT
  ```

- Koordiniert

  Dieses Sprachkonzept realisiert das Master-Slave-Konzept. Die Definition eines im Effektorkoordinatensystem des Slave zu verfahrenden Pfads erfolgt durch die Angabe von Master und Slave als Bewegungsgruppen bei der Definition des Pfads in der Zeile `paint_path_2 : path in group 1 and 2`. Der erste Befehl innerhalb einer Bewegungsgruppe beschreibt die Bewegung des Masters im Weltkoordinatensystem, die darauf folgenden Befehle beschreiben die Bewegungen der Slaves relativ zum Toolkoordinatensystem des Masters.

  Beispiel:

```
VAR  table_path_1 : path in group 1
     paint_path_2 : path in group 1 and 2

...


MOVE ALONG table_path_1,
     ALONG paint_path_2, SIMULTANEOUS
```

Während für alleinstehend eingesetzte Manipulatoren Roboterprogrammiersprachen der zweiten Generation [BJ83], wie VALII [SGIS85], RAIL [FV82], AL [FTB+75] oder AML [Tay82], mit den in ihnen vorhandenen Bewegungsbefehlen ein ausreichendes Sprachmittel darstellen, ist eine Programmierung von kooperierenden Manipulatoren bei Aufgaben mit abhängigen Effektorkoordinatensystemen mit Erweiterungen dieser Sprachen nur noch beschränkt möglich. Das Problem der Vermeidung von Kollisionen, das bei alleinstehend eingesetzten Robotern in einer statischen Umgebung vom Roboterprogrammierer intuitiv gelöst werden kann, ist bei gemeinsamen Bewegungen mehrerer Manipulatoren in einem gemeinsamen Arbeitsraum nur noch schwer ohne zusätzliche Hilfsmittel zu bewältigen. Zur Programmierung kooperierender Manipulatoren sind somit Sprachmittel erforderlich, die die Integration von Bahnplanungswerkzeugen unterstützen und von der Manipulatorebene abstrahieren. Dies leistet die in dieser Arbeit vorgestellte Spezifikationsmethode.

Ein weiteres Problem stellt die Form des Arbeitsraums der Manipulatoren dar. Während er bei alleinstehend eingesetzten Manipulatoren ebenfalls statisch und vom Programmierer intuitiv erfaßbar ist, hängt er bei kooperierenden Manipulatoren von der Transformation zwischen den Effektorkoordinatensystemen ab und ist somit nicht mehr statisch. Diese Zusammenhänge wurden in einer Arbeit deutlich, in der Bewegungssequenzen zur Lösung des Rubic's Cube mit zwei Manipulatoren realisiert wurden [KP94]. Roboterprogrammiersprachen auf der Bewegungsebene, wie etwa der oben beschriebene Ansatz, stellen somit kein adäquates Mittel zur Programmierung kooperierender Manipulatoren dar. Vielmehr ist die Programmierung auf Aufgabenebene und die Unterstützung durch Bahnplanungswerkzeuge zur Realisierung der hier betrachteten Aufgabenklasse erforderlich. Spracherweiterungen

wie die hier betrachtete können lediglich als Zwischenschicht zur Bedienung der Schnittstelle zu den Manipulatorsteuerungen dienen.

Ein zentraler Punkt bei der Programmierung kooperierender Manipulatoren ist – wie bereits erwähnt – die Behandlung von Kräften, die durch auftretende geschlossene kinematische Ketten entstehen. Die Integration von Sensorik, insbesondere von Kraft-Momenten-Sensoren, muß also von einem System zur Programmierung kooperierender Manipulatoren gewährleistet sein, findet aber in dieser Spracherweiterung keine Berücksichtigung. In der hier vorgestellten Spezifikationsmethode ist die Berücksichtigung von Sensorinformation – insbesondere von Kräften und Momenten – integriert.

3.3 Geschlossene kinematische Ketten und Kontaktkräfte

Innerhalb von geschlossenen kinematischen Ketten, die von den Manipulatoren und einem gemeinsam manipulierten Objekt gebildet werden, treten *innere Kräfte* auf. So sind etwa beim gemeinsamen Transport eines starren Objekts durch zwei Manipulatoren die beim Greifen auftretenden Kräfte und Momente zu kompensieren, um mechanische Beschädigungen des manipulierten Objekts oder der Manipulatoren zu vermeiden. Ein anderes Beispiel ist etwa der Transport einer gespannten Feder. Hier müssen als innere Kräfte die zum Spannen erforderlichen Kräfte aufgebracht werden. Desweiteren treten auch bei Fügevorgängen, bei denen jeder der beiden Manipulatoren ein zu montierendes Bauteil hält, innere Kräfte auf.

Zusätzlich zur Behandlung innerer Kräfte kann bei manchen Aufgaben die Behandlung äußerer Kräfte erforderlich werden. Äußere Kräfte sind Kräfte, die zwischen dem manipulierten Objekt und Objekten der Umgebung auftreten. Beispiele sind etwa Fügevorgänge zwischen einem von beiden Manipulatoren gehaltenen Objekt und einem mit dem Weltkoordinatensystem fest verbundenen Objekt.

Das Konzept der nachgiebigen Bewegung (*compliant motion*), behandelt diese Problemklasse. Es wird in mehreren Arbeiten sowohl aus theoretischer als auch aus regelungstechnischer Sicht ausführlich behandelt. Dabei sei auf theoretischer Seite besonders eine in diesem Bereich grundlegende Arbeit von Mason [Mas82] erwähnt.

Nachgiebige Bewegung kann nach Mason durch Angabe eines *Compliance Frames* spezifiziert werden. Entlang der Koordinatenachsen dieses Frames werden die Richtungen und Beträge der gewünschten Kräfte und Momente für eine gegebene Aufgabe spezifiziert. Neben der Angabe von Kräften und Momenten ist im Konzept von Mason die Spezifikation einer bestimmten Geschwindigkeit in den entsprechenden Freiheitsgraden vorgesehen. Somit kann sowohl eine Kraft/Moment- als auch eine Positionsregelung spezifiziert werden.

Als Beispiel sind hier zwei Standardaufgaben angegeben: In Abb. 7 soll ein Bolzen in eine Passung eingeführt werden. Das Compliance-Frame befindet sich hierbei an der Spitze des einzusetzenden Bolzens. Entlang der X- und Y-Achsen des Compliance-Frames sollen hierbei keine Kräfte und Momente auftreten. Der rotatorische Freiheitsgrad um die Z-Achse

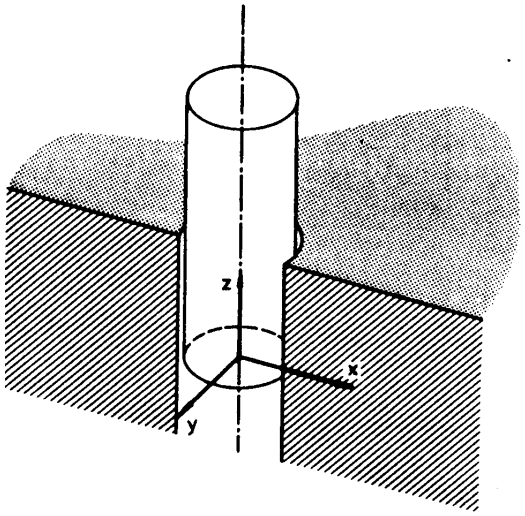


Abbildung 7: Einsetzen eines Bolzens in eine Passung (*peg-in-hole*) (aus [Mas82])

bleibt undefiniert, der translatorische Freiheitsgrad entlang der Z-Achse kann durch Angabe einer Geschwindigkeit ("Einführen mit konstanter Geschwindigkeit x mm/s") spezifiziert werden.

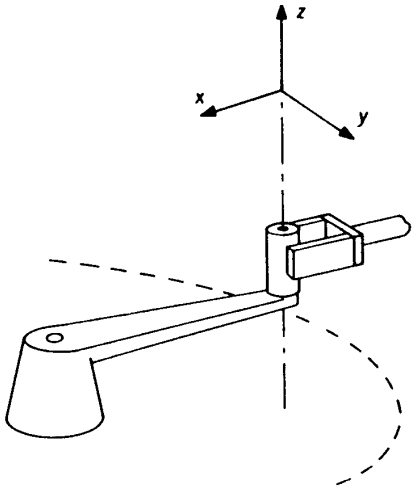


Abbildung 8: Drehen einer Kurbel (aus [Sch86])

In Abb. 8 soll eine Kurbel gedreht werden. Das Compliance-Frame befindet sich in der Drehachse des Kurbelgriffs. Zum Drehen der Kurbel soll eine bestimmte Geschwindigkeit in Richtung der Y-Achse verfahren werden. Die Kräfte und Momente in allen anderen Freiheitsgraden sollen kompensiert werden. Durch Spezifikation der Kräfte und Momente in einem orthogonalen Compliance-Frame kann eine sehr große Klasse von nachgiebigen

Bewegungen spezifiziert werden.

Die Arbeit von Mason wurde in mehreren Nachfolgearbeiten von De Schutter und Leysen [Sch86, Sch88, Ley91] zu einem Spezifikationsformalismus für *compliant motion* verfeinert. Dieser Formalismus übernimmt das Konzept des Compliance-Frames, das in diesem Ansatz als *Taskframe* bezeichnet wird.

Neben den beiden Typen "kraftgeregelt" (`force`) und "positionsgeregelt" (`velocity`) für die Freiheitsgrade des Task-Frames wird für die rotatorischen Freiheitsgrade der Typ "modellbasiert verfolgen" (`track`) eingeführt. Dieser ist etwa für Konturverfolgungsaufgaben erforderlich und besagt, daß der spezifizierte Freiheitsgrad mit dem Freiheitsgrad eines entsprechenden Frames in einem Modell in Übereinstimmung gebracht werden soll. Dies soll am Beispiel einer 2D-Konturverfolgungsaufgabe verdeutlicht werden. In der Aufgabe in Abb. 9 wird der rotatorische Freiheitsgrad α_{zt} des Taskframes als vom Typ `track` spezifiziert. Dies bedeutet hier, daß die Orientierung des Taskframes so geregelt werden muß, daß es mit dem Modellframe (X-Achse x_o, Y-Achse y_o) zur Deckung kommt. Das Modellframe ist hierbei durch ein Modell der Aufgabe bestimmt, seine X-Achse liegt tangential zur Konturoberfläche an dem Punkt, an dem das Tool die Oberfläche berührt, sein Ursprung ist der Berührpunkt zwischen Tool und Konturoberfläche. Ein *Modell* der Aufgabe ist also zur Aufgabenausführung unbedingt erforderlich.

Eine zusätzliche Erweiterung von De Schutter ist die Einführung von Terminierungsbedingungen für die Aufgabenausführung. Es werden folgende Klassen von Terminierungsbedingungen aufgeführt, für die auch logische Verknüpfungen angegeben werden können:

- Zeitbedingung

  Die Bewegung endet, wenn ein bestimmtes Zeitintervall vergangen ist.

- Kraftbedingung

  Die Bewegung endet, wenn die Kraft in einer bestimmten Taskframe-Richtung eine gegebene Schranke überschreitet.

- Entfernungsbedingung

  Die Bewegung endet, wenn der Endeffektor eine gegebene Entfernung in einer bestimmten Taskframe-Richtung zurückgelegt hat.

- Positionsbedingung

  Die Bewegung endet, wenn eine Komponente der Endeffektorposition eine bestimmte Schranke überschreitet.

Für manche Aufgaben können nicht alle Freiheitsgrade des Taskframes durch obige Typen spezifiziert werden. Beim Drehen einer Kurbel, bei der der Griff um seine Z-Achse rotieren kann, gibt es keine natürliche Einschränkung für diesen Freiheitsgrad, wenn der Griff frei drehbar an der Kurbel montiert ist. De Schutter löst dieses Problem durch Einführung

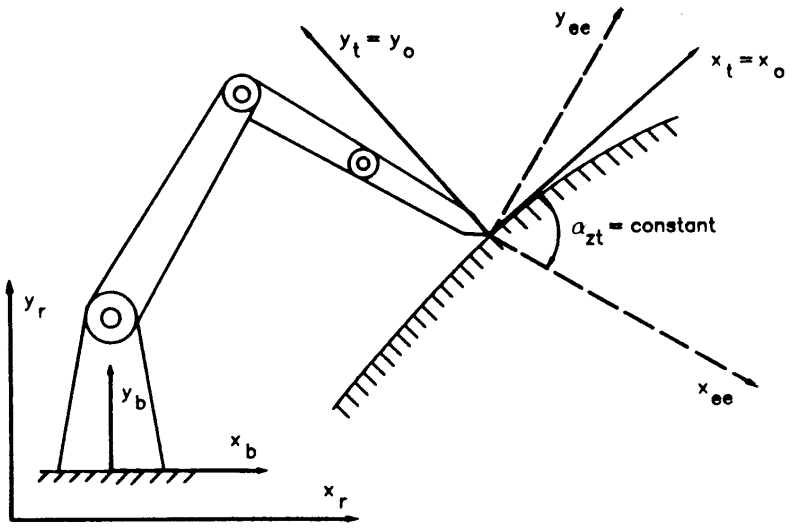


Abbildung 9: Verfolgen einer Oberfläche in 2D (aus [Sch86])

einer Endeffektor-Bewegungseinschränkungsmatrix, die direkt in einen Regelungsalgorithmus eingeht. Dies ist eine Diagonalmatrix mit Elementen 1 (DOF spezifiziert) bzw. 0 (DOF nicht eingeschränkt). Nicht spezifizierte Freiheitsgrade können dann entweder dadurch behandelt werden, daß man Bewegungen für sie verbietet oder sie dazu verwendet, die Beweglichkeit des Manipulators zu erhöhen.

Desweiteren gibt es Aufgaben, bei denen das Taskframe nicht fest mit dem Effektorkoordinatensystem verbunden ist, sondern in einem (oder mehreren) Freiheitsgraden beweglich ist. Beim Entlangziehen eines Stifts über eine Tischkante bleibt das Taskframe am Berührpunkt zwischen Tischkante und Stift, bewegt sich also entlang der Z-Achse des Stifts, während alle anderen Freiheitsgrade relativ zum Ursprungskoordinatensystem des Stifts fest bleiben. Durch Einführen einer Taskframe-Bewegungseinschränkungsmatrix – analog zu obiger Matrix – werden die Freiheitsgrade spezifiziert, in denen das Taskframe relativ zu sich unbewegt bleibt, auch wenn sich das Effektorframe bewegt.

Die Spezifikation der oben angeführten Aufgabe *Einsetzen eines Bolzens in eine Passung* mittels De Schutter's Formalismus stellt sich folgendermaßen dar:

```
Move compliantly
     with task frame
          defined fixed to the end effector
               at the tip of the peg
          and with task frame directions :
               x_t : force : 0 N
               y_t : force : 0 N
               z_t : velocity : -v mm/s
               α_xt: force : 0 Nm
               α_yt: force : 0 Nm
               α_zt: velocity : 0 rad/s
     until z_t-force exceeds F N
          and z-travelled distance in task frame exceeds s mm.
```

Zusammenfassend kann gesagt werden, daß die Spezifikation einer nachgiebigen Bewegung nach De Schutter aus folgenden Komponenten besteht:

- Definition des Taskframes mit
  - seinem Bezugskoordinatensystem (und ggf. den erlaubten Bewegungsrichtungen)
  - den Typen der einzelnen Freiheitsgrade (`force`, `velocity`, `track`)
- Endeffektor-Bewegungseinschränkungsmatrix
- Taskframe-Bewegungseinschränkungsmatrix
- Terminierungsbedingung

Da diese Ansätze zur Spezifikation der Behandlung auftretender Kräfte und Momente aus Sicht der Regelungstechnik entstanden sind, wurden Konstrukte eingeführt, die die Umsetzung der Spezifikation in Regelungsgesetze vereinfachen. Dies sind die Endeffektor- und Taskframe-Bewegungseinschränkungsmatrizen. Aus Sicht der reinen Aufgabenspezifikation sind derartige Spezialkonstrukte allerdings nicht wünschenswert.

De Schutter's Formalismus spezifiziert die Beziehungen zwischen den beteiligten Koordinatensystemen allerdings nur informell (`fixed to the end effektor`, `at the tip of the peg`, ...). Für eine automatische Umsetzung der Spezifikation im Rahmen der aufgabenorientierten Programmierung ist allerdings eine formale Spezifikation erforderlich. Ein weiterer Schwachpunkt stellt die Spezifikation von modellbasierten Aufgabenframes bei Konturverfolgungsaufgaben dar. De Schutter beschreibt diese Frames nur als "Computer-Repräsentation des physikalischen Aufgabenframes" und geht auf diese nicht näher ein. In dieser Arbeit wird dieser Formalismus zur Beschreibung auftretender Kräfte verwendet, obige Schwachpunkte werden im Rahmen der formalen Aufgabenbeschreibung beseitigt.

3.4 Umsetzung von Aufgabenspezifikationen durch Planungswerkzeuge

Ziel einer Aufgabenspezifikation bei der aufgabenorientierten Programmierung von Manipulatoren ist es, dem Programmierer die – gerade bei kooperierenden Manipulatoren – komplexe Programmierung der Manipulatoren dadurch zu erleichtern, daß verschiedene Tools den Programmiervorgang unterstützen. Der Programmierer hat hierbei nur die auszuführende Aufgabe zu beschreiben, die zur Verfügung stehenden Tools leisten dann die Umsetzung in eine von Manipulatoren ausführbare Form.

In Abb. 10 finden sich auf der linken Seite die in [Wer93] unterschiedenen drei Ebenen der Manipulatorprogrammierung wieder. Auf der aufgabenorientierten Ebene werden komplexe, symbolisch beschriebene Aufgaben spezifiziert. Beispiele hierfür sind etwa Aufgaben der Art "*Fertige Bauteil X*" oder "*Koche eine Tasse Kaffee*". Aufgaben dieser Schicht müssen durch Aufgabenplaner in Sequenzen von Elementaraufgaben zerlegt werden [Fis92]. Die Planung von Aufgabensequenzen erfolgt dabei in Abhängigkeit von der Aufgabenausführung. So wird der Status von Elementaraufgaben nach ihrer Ausführung in die Planung der weiteren Aufgabensequenz einbezogen. Dadurch können Fehlerbehandlungsstrategien im Falle einer fehlerhaft ausgeführten Elementaraufgabe in die Ausführung einer komplexen Aufgabe integriert werden. Ebenso ist hierdurch eine Reaktion auf Ungenauigkeiten nach der Ausführung von Elementaraufgaben möglich.

Symbolisch beschriebene Elementaraufgaben der objektorientierten Ebene, etwa der Form "*Greife Tasse T*" oder "*Füge den Bolzen B in die Passung P*" können nun bei kooperierenden Manipulatoren nicht mehr direkt – wie bei einzeln eingesetzten Manipulatoren – in Bewegungs- und Greifbefehle umgesetzt werden. Vielmehr ergeben sich durch die erweiterte Funktionalität kooperierender Manipulatoren (parallele Bewegungen, gleichzeitiges Bewegen/Manipulieren von Objekten, Auftreten geschlossener kinematischer Ketten) komplexere Anforderungen an die Umsetzung spezifizierter Elementaraufgaben in eine ausführbare Form. Daher wird in dieser Arbeit eine Zwischenschicht eingeführt, mittels der Elementaraufgaben für kooperierende Manipulatoren spezifiziert werden können. Diese Schicht leistet eine implizite (*was* ist zu tun statt *wie* ist etwas zu tun), von den verwendeten Manipulatoren abstrahierende, Beschreibung der Elementaraufgaben.

Symbolisch beschriebene Elementaraufgaben werden dann von entsprechenden Planungswerkzeugen – Greifplanern und Feinbewegungsplanern – in Elementaraufgaben umgewandelt, die durch die hier vorgestellte Spezifikationssprache beschrieben werden. Greifplaner [VMT90, HH90a] liefern zu einem gegebenen Objekt und einem gegebenen Greifer Greifpunkte am Objekt, an denen ein stabiler Griff möglich ist. Feinbewegungen – Bewegungen mit Kontakt zwischen Objekten – sind vor allem bei Montageaufgaben erforderlich [HH90b, Hör88]. Feinbewegungsplaner planen Trajektorien, die dann mit der hier vorgestellten Spezifikationssprache beschrieben werden können.

Um mit der hier vorgestellten Spezifikationssprache geometrisch spezifizierte Elementaraufgaben in eine von Manipulatoren ausführbare Form umzusetzen, sind weitere Planungs-

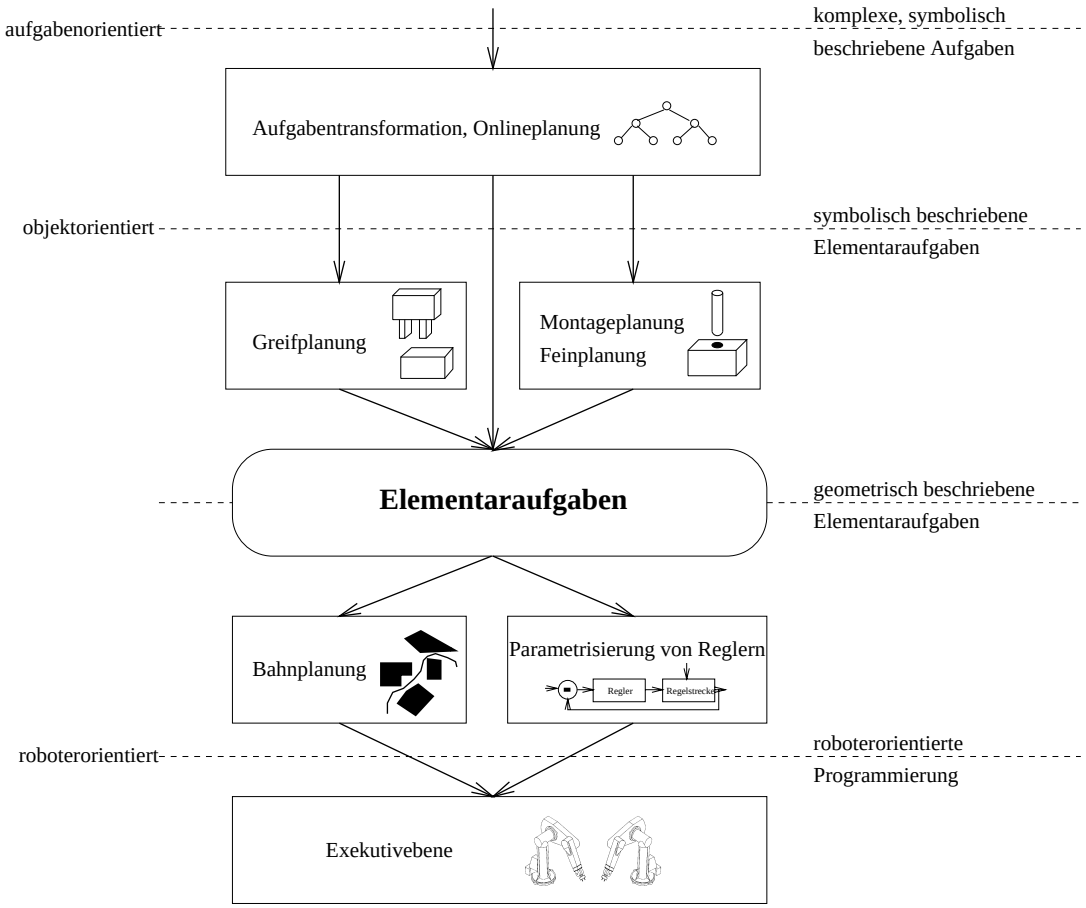


Abbildung 10: Planungskomponenten in einem aufgabenorientiert programmierbaren System

instanzen erforderlich:

- Generierung bzw. Parametrisierung von Regelungsvorschriften

  Um die in der vorgestellten Sprache spezifizierten, auf Sensorinformation basierenden Constraints umzusetzen, ist die Generierung bzw. Parametrisierung von Regelungsvorschriften erforderlich. Diese setzen Sensorinformation in Bewegungen der Manipulatoren um. Ansätze hierzu sind die Arbeiten von De Schutter und Leysen zur Behandlung von nachgiebiger Bewegung (s. Kap. 3.3) oder der in [BAH94] vorgestellte Ansatz, mit dem die Generierung bzw. Parametrisierung von Regelungsvorschriften, die auf Kraft-Momenten-Sensoren, Abstandssensoren und Bildverarbeitung basieren, möglich ist.

- Bewegungsplanung

Ein Teilbereich der Bewegungsplanung ist die Bahnplanung, also die Planung kollisionsfreier, von den Manipulatoren verfahrbarer Bahnen. Ein Bahnplanungsverfahren für eine Teilklasse der durch die vorgestellte Sprache spezifizierten Elementaraufgaben wird in Kap. 6 beschrieben.

3.5 Bahnplanung

3.5.1 Bahnplanung für Einzelmanipulatoren

Bahnplanung, die Suche nach einem kollisionsfreien Weg für ein Objekt von einer Start- zu einer Zielstellung durch eine Umgebung mit Hindernissen, ist ein schwierig zu lösendes Problem. In mehreren theoretischen Arbeiten [SSH87, Can88] wurde die Komplexität des Bahnplanungsproblems betrachtet. Der aktuelle Stand der Forschung auf dem Gebiet der Komplexität besagt, daß das allgemeine Wegplanungsproblem PSPACE-hart ist und eine untere Grenze hat, die exponentiell in der Anzahl der Freiheitsgrade des Konfigurationsraums ist [HN92].

Im Bereich der Bahnplanungsverfahren gibt es eine Vielzahl von Ansätzen, einen Überblick hierzu geben [Lat91, HN92]. Ein in der Anfangszeit der Bahnplanung verfolgter Ansatz sind vollständige Planungsalgorithmen für Konfigurationsräume mit niedriger Dimension, bei denen die exponentielle Komplexität beherrschbar bleibt [LP87]. Hierbei wird eine explizite Repräsentation des diskretisierten Konfigurationsraums – etwa eine Raster- oder Baumdarstellung – benutzt, um mittels vollständiger Suchalgorithmen einen Weg zum Ziel zu finden. Als derartige Algorithmen werden Wellenfrontalgorithmen [Lat91], Potentialfeldmethoden oder Skelettieralgorithmen verwendet. Wegen des oben erwähnten exponentiellen Aufwands in den Freiheitsgraden sind vollständige Suchalgorithmen daher nur etwa bis zu fünfdimensionalen Konfigurationsräumen effektiv verwendbar und werden vorzugsweise im Bereich der Bahnplanung für autonome Fahrzeuge verwendet. Diese Planungsalgorithmen werden auch als auflösungs-vollständig (*resolution-complete*,[Lat91]) bezeichnet. Zu einer gegebenen Auflösung der expliziten Darstellung wird ein existierender Weg gefunden. Findet der Planer keinen Weg, so kann immer noch ein Weg bei einer feineren Auflösung existieren. Wählt man die Auflösung im Bereich der zu erwartenden Ungenauigkeiten (etwa durch die Umweltmodellierung oder durch die beim Verfahren des Wegs erzielbare Genauigkeit), so findet der Planer sicher einen Weg, wenn einer existiert.

Einer der leistungsfähigsten Planer dieser Klasse wird von Ralli [RH94] vorgestellt. Dieser Planer bewältigt Manipulatoren mit 5 Freiheitsgraden. Für ein angegebenes Beispiel wird eine relativ grobe Diskretisierung des Konfigurationsraums in 2.86×10^6 Konfigurationen durchgeführt. Die Berechnung des freien Konfigurationsraums in einem Vorverarbeitungsschritt erfordert etwa 55 min. auf einer leistungsfähigen Workstation. Dieser Aufwand kann gerechtfertigt werden, wenn Planungsläufe mit verschiedenen Start- und Zielpunkten in derselben unveränderten Umgebung durchgeführt werden. Für Onlineplaner in sich verändernden Umgebungen, die eine Neuberechnung der expliziten Konfigurationsraum-

darstellung erforderlich machen, ist dieses Verfahren ungeeignet. Da zudem der Aufwand zur Berechnung der expliziten Konfigurationsraumdarstellung exponentiell in der Anzahl der Freiheitsgrade wächst, würde dies für Konfigurationsräume mit mindestens sechs Dimensionen den Bereich des effizient Berechenbaren schnell übersteigen.

Daher wurden in verschiedenen Arbeiten [Gla91b, BL90, BL91] Planungsverfahren entwickelt, die auf eine explizite Darstellung des Konfigurationsraums verzichten und stattdessen Heuristiken zur Suche nach einem kollisionsfreien Weg verwenden, die überwiegend lokale Information über den Konfigurationsraum benutzen.

Im Planungsverfahren RPP (*randomized path planner*) von Barraquand und Latombe [BL91] wird an einem Objekt, für das ein Weg zu planen ist, eine Menge von *Kontrollpunkten* definiert. Für jeden dieser Kontrollpunkte wird ein Potential berechnet. Dieses besteht aus einer Kombination aus einem abstoßenden Potential, das die minimale Entfernung zu einem Hindernis beschreibt, und einem anziehenden Potential, das die Entfernung zur Zielstellung des Kontrollpunkts beschreibt. Diese Potentiale für die einzelnen Kontrollpunkte werden zu einem Gesamtpotential zusammengefaßt, das dann den Abstand einer Konfiguration zur Zielkonfiguration beschreibt. Dieses Potential wird lokal in Form des Gradienten ausgewertet, um Bewegungen für die Kontrollpunkte entlang des negierten Gradienten des Gesamtpotentials zu bestimmen. Diese Gradientenbewegungen führen in manchen Situationen in lokale Minima. Um aus diesen zu entweichen, verwendet RPP zufällige, Brownschen Molekularbewegungen gleichende Bewegungen. RPP durchschreitet durch diese Abfolge von Bewegungen entlang des negierten Gradienten und Zufallsbewegungen den freien Konfigurationsraum. Mit zunehmender Länge des Wegs werden immer größere Bereiche des freien Konfigurationsraums erforscht. Durch diese Abfolge von Potentialsuche und Entweichen aus lokalen Minima verringert sich der Abstand zum Ziel mit zunehmender Planungsdauer. Somit entsteht eine Überdeckung des freien Konfigurationsraums durch einen langen, Brownschen Molekularbewegungen entsprechenden, kollisionsfreien Weg.

Im ZZ-Verfahren von Glavina [Gla91b] wird noch weniger Information über den Konfigurationsraum verwendet. Es wird lediglich getestet, ob eine gegebene Konfiguration kollisionsfrei ist oder nicht. Das ZZ-Verfahren versucht, eine Start- und eine Zielkonfiguration im Konfigurationsraum geradlinig zu verbinden. Trifft der Planer dabei auf ein Hindernis, versucht er, durch ein Gleitschrittverfahren (s. Kap. 6.6.2) an der Oberfläche des Hindernisses entlangzugleiten und so dem Hindernis auszuweichen, solange er dadurch der Zielkonfiguration näher kommt. Schlägt dieser Versuch fehl, wird versucht, die beiden Konfigurationen in umgekehrter Richtung zu verbinden, was in vielen Fällen zum Erfolg führt. Gelingt es nicht, Start und Ziel so zu verbinden, werden zufällige Zwischenziele erzeugt und nach obiger Methode mit den bereits existierenden Start-, Ziel- und Zwischenzielkonfigurationen zu verbinden versucht. Dadurch wird mit zunehmender Planungsdauer der freie Konfigurationsraum durch einen dichter werdenden Graphen aus kollisionsfreien Konfigurationen und Wegen zwischen diesen ausgefüllt, bis Start und Ziel in einer Zusammenhangskomponente dieses Graphen liegen. Im Vergleich zu RPP wird dadurch die gewonnene Information über den freien Konfigurationsraum besser genutzt.

Diese sogenannten heuristischen Planer, deren Suchheuristiken auf die explizite Darstellung des Konfigurationsraums verzichten, können effektiv auch höhere Dimensionen bewältigen. Nur lokale Information benutzend werden in diesen Verfahren *Wege* durch den Konfigurationsraum aufgebaut, die *immer näher* auf ein Ziel hinführen. Der Begriff *immer näher* wird hierbei jeweils durch ein Abstandsmaß beschrieben. Bei Barraquand [BL90, BL91] ist dies der euklidische Abstand der Kontrollpunkte von ihrer Sollstellung im dreidimensionalen kartesischen Raum, bei Glavina wird ein gewichteter euklidischer Abstand von durch Gelenkwinkeln beschriebenen Manipulatorkonfigurationen betrachtet.

Ein Problem von heuristischen Planern ist das Steckenbleiben in lokalen Minima durch die Benutzung lediglich lokaler Information. Daher sind spezielle globale Suchstrategien erforderlich, wie etwa zufällige Bewegungen oder Aufbau eines Graphen aus Zwischenzielen, um diese Sackgassen zu vermeiden. Desweiteren muß bei heuristischen Planern der Vollständigkeitsbegriff abgeschwächt werden. Barraquand und Latombe [BL90, BL91] führen hierzu den Begriff der *probabilistischen Vollständigkeit (probabilistically complete)* ein: Für gegen Unendlich gehende Planungszeiten konvergiert die Wahrscheinlichkeit, einen Weg zu finden, wenn einer existiert, gegen 1. Hierin liegt ein Nachteil heuristischer Planer begründet. So kann ein heuristischer Planer nicht dazu verwendet werden, um die Nichtexistenz von Lösungen für eine Planungsaufgabe nachzuweisen. Für realistische Anwendungen, bei denen das schnelle Auffinden einer Lösung im Vordergrund steht, ist die Leistungsfähigkeit heuristischer Planer jedoch ausreichend. So ist das ZZ-Verfahren [Gla91b] auch bei komplizierten Aufgaben in der Lage, Lösungen im Sekundenbereich zu liefern.

Eine Arbeit, die sich auf die Realisierung einer mächtigen globalen Strategie konzentriert, wird von Kavraki in [KL94b, KL94a] vorgestellt. In diesem Ansatz wird der freie Konfigurationsraum analog zu Glavina [Gla91b] durch einen Graphen aus Zwischenzielen und freien Wegen, die diese verbinden, repräsentiert. Dies erfolgt in einem Vorverarbeitungsschritt, in dem zufällige Konfigurationen generiert, auf Kollisionsfreiheit getestet und mit den N nächstgelegenen Nachbarn durch einen einfachen, schnellen lokalen Planer zu verbinden versucht werden. In vielen Fällen entstehen dabei mehrere Zusammenhangskomponenten des Graphen, und Knoten, die nur spärlich mit anderen Knoten verbunden sind, weil sie in "schwierigen Gegenden des Konfigurationsraums liegen"[KL94b]. In einem zweiten Schritt wird versucht, den Graphen in diesen schwierigen Gegenden zu verbessern, indem selektiv neue Knoten in der Nachbarschaft spärlich mit Nachbarn verbundener Knoten generiert werden. Dadurch werden gegebenenfalls bislang getrennte Zusammenhangskomponenten verbunden und die Repräsentation schwieriger, enger Konfigurationsraumbereiche verbessert. Nach dieser Vorverarbeitung des Konfigurationsraums werden Wege zwischen verschiedenen Konfigurationen in derselben Umgebung sehr schnell gefunden, indem Start- und Zielkonfiguration mit dem jeweils nächstgelegenen Knoten des Graphen verbunden und ein Weg durch den Graphen gesucht wird.

Ein Nachteil dieses Verfahrens bleibt allerdings der notwendige Neuaufbau des Graphen, wenn Veränderungen im Arbeitsraum auftreten. Eine Verschränkung von Planung und Aufbau eines Graphen wie bei Glavina erscheint daher günstiger, da sich die Umgebung

bei verschiedenen Aufgaben im allgemeinen ebenfalls ändert.

Die Grundidee heuristischer Verfahren – speziell das von Glavina vorgeschlagene – bilden wegen ihrer Eignung für höherdimensionale Konfigurationsräume die Grundlage der in dieser Arbeit beschriebenen Planungsverfahren. In dieser Arbeit werden Aufgaben für kooperierende Manipulatoren betrachtet. Als Planungsziele werden nicht nur, wie für die oben beschriebenen Planungsverfahren, einzelne Konfigurationen, sondern Räume von Konfigurationen vorgegeben. Daher sind wegen der hier betrachteten Aufgabenklasse wesentliche Erweiterungen der Planungsverfahren erforderlich.

3.5.2 Bahnplanung für kooperierende Manipulatoren

Während im Bereich der Bahnplanung zur Koordination der Bewegung mehrerer Manipulatoren in einem gemeinsamen Arbeitsbereich mehrere Arbeiten veröffentlicht wurden [FH87, FM94, HGB94], gibt es nur wenige Ansätze, die sich mit Bahnplanung für Manipulatoren beschäftigen, die eine gemeinsame Aufgabe zu erfüllen haben. Das Problem der kooperativen Ausführung einer Aufgabe durch mehrere Manipulatoren wurde bislang überwiegend auf der Regelungsebene behandelt.

Ein Ansatz zur Planung von Bearbeitungstrajektorien für realistische Anwendungen wird in [PK91] vorgestellt. Auf dem zu bearbeitenden Werkstück wird eine Bearbeitungstrajektorie $\mathbf{T_T(t)}$ in Werkstückkoordinaten festgelegt. Dies erfolgt durch Angabe einer Sequenz ausreichend nahe nebeneinanderliegender Interpolationspunkte. An einem Manipulatorarm ist das Werkstück befestigt, während der zweite Arm das Werkzeug trägt. Zusätzlich kann eine Vorzugslagenbedingung definiert werden. In dem in Abb. 11 dargestellten Beispiel soll die Z-Achse des Werkzeugs antiparallel zur Z-Achse des Weltkoordinatensystems liegen.

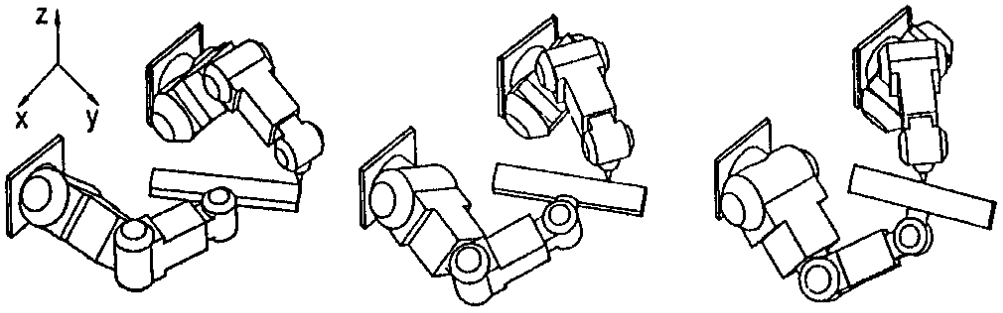


Abbildung 11: Verfahren einer Bearbeitungstrajektorie (aus [PK91])

Die so definierte Aufgabe kann dann als Gleichung von homogenen Transformationsmatrizen beschrieben werden (es werden die Originalbezeichnungen aus [PK91] verwendet).

$$\mathbf{T_B^{B2}T_{R2}(q^2)T_{W_s}T_T(t)} = \mathbf{T_B^{B1}T_{R1}(q^1)T_{W_g}} \tag{1}$$

Obige Vorzugslagenbedingung kann dann folgendermaßen formuliert werden:

$$\mathbf{T_B^{B2}T_{R2}(q^2)T_{W_s}T_T(t)} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \end{bmatrix} \tag{2}$$

mit

$\mathbf{T_B^{B1}}, \mathbf{T_B^{B2}}$	Transformation vom Basiskoordinatensystem des Doppelarmmanipulators in die Bezugskoordinatensysteme der Arme
$\mathbf{T_{R1}(q^1)}, \mathbf{T_{R2}(q^2)}$	Vorwärtstransformationen für die Arme
$\mathbf{T_{W_s}}$	Transformation vom Greiferkoordinatensystem in das Werkstückkoordinatensystem
$\mathbf{T_T(t)}$	vorgegebene Trajektorie auf dem Werkstück, definiert in Werkstückkoordinaten
$\mathbf{T_{W_g}}$	Transformation vom Greiferkoordinatensystem in das Werkzeugkoordinatensystem
$\mathbf{q^1}, \mathbf{q^2}$	Gelenkparameter des Manipulators **R1** bzw. **R2**

Aus den beiden Gleichungen 1) und 2) ergibt sich ein nichtlineares, unterbestimmtes Gleichungssystem, das inkrementell für jeden Zeitpunkt $\mathbf{t}$ durch das Newton-Raphson-Verfahren [Sto83] mit den vorhergehenden Gelenkparametern als jeweiliger Startwert gelöst wird. Um die Lösungen in dem sich wegen der Unterbestimmung ergebenden Lösungsraum zu bewerten, wird eine Potentialfunktion eingesetzt. Mittels dieser Potentialfunktion kann eine Optimierung der geplanten Bahn nach verschiedenen Kriterien durchgeführt werden. Pritschow und Koch beschreiben eine Potentialfunktion, die die Gelenke von ihren Grenzen entfernt hält [PK91]. Dieser Ansatz läßt allerdings das grundlegende Problem auftretender Kollisionen unberücksichtigt. Da es sich bei diesem Verfahren um ein rein lokales Verfahren handelt, bleibt auch das Problem von Sackgassen – hier lediglich das Steckenbleiben in einer zu planenden Trajektorie wegen einer drohenden Gelenkgrenzüberschreitung, da Kollisionen unberücksichtigt bleiben – ungelöst. Das in dieser Arbeit vorgestellte Bahnplanungsverfahren ist in der Lage, kollisionsfreie Bahnen für kooperierende Manipulatoren zu planen und dabei auftretende Sackgassen zu vermeiden.

In [DKHM87] wird derselbe Ansatz beschrieben, sowie die Verifikation der Methode anhand zweier Industriemanipulatoren. Dazu werden die für jeden der beiden Manipulatoren geplanten Trajektorien als Linearsätze von Einzelstellungen in den Steuerrechner geladen und zeitlich synchron gestartet. Eventuell entstehende Abweichungen werden durch eine Online-Korrektur der Verfahrgeschwindigkeit ausgeglichen. Duelen und Kirchhoff weisen auch auf das Hauptproblem dieses Ansatzes hin, daß die geplanten Bahnen ohne Berücksichtigung von Kollisionen generiert werden. Kollisionsvermeidung wäre etwa durch eine

Potentialfunktion, die den minimalen Abstand zu einem Hindernis beschreibt, möglich. Eine mathematische Beschreibung und Auswertung dieser Potentialfunktion ist zwar unter großem Aufwand möglich, dennoch bleibt auch hier das Problem von Sackgassen durch die rein lokale Betrachtung des Trajektorienplanungsproblems ungelöst.

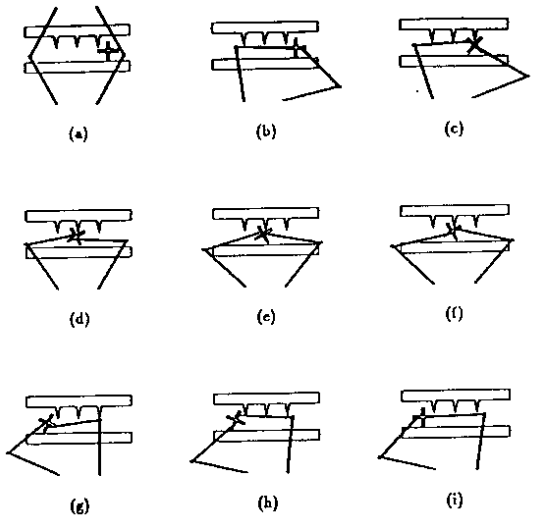


Abbildung 12: Transport mit zwei Manipulatoren (aus [KL92])

In [KL92] beschreibt Koga drei verschiedene Strategien zur Bahnplanung für zwei Manipulatoren in Hindernisumgebungen. Diese beschränken sich dabei auf Transportaufgaben, wobei Umgreifaktionen mitbetrachtet werden. Die Planung erfolgt im zweidimensionalen Raum. Für die ersten beiden Planer werden linienartige Manipulatoren mit zwei Freiheitsgraden verwendet, für den dritten Planer zwei linienartige Manipulatoren mit drei Freiheitsgraden. In dem in Abb. 12 dargestellten Beispiel sollen die beiden Manipulatoren, die sich in einer Ebene über den Hindernissen befinden, das sternförmige Objekt von rechts nach links durch die Hindernisse transportieren. Um Kollisionen der Manipulatoren miteinander zu vermeiden, sollen, wenn nötig, Umgreifoperationen durchgeführt werden.

Als Suchstrategie wird bei den ersten beiden Planern eine vollständige Suchstrategie (Best-First Suche) verwendet. Zur Darstellung des Konfigurationsraums werden die kartesischen Konfigurationen des zu transportierenden Objekts (Planer 1) bzw. die Gelenkparameter eines Manipulators, aus dem die Konfiguration des zweiten Manipulators durch Rückwärtsrechnung bestimmt wird (Planer 2), verwendet.

Der dritte Planer wurde für Manipulatoren mit mehr als zwei Freiheitsgraden entwickelt. Aus diesem Grund wird die vollständige Suche aufgegeben und eine heuristische Potentialsuche, eine Modifikation des bereits beschriebenen RPP (s. Kap. 3.5.1) mit sogenannten *random walks* zum Entweichen aus Sackgassen verwendet. Diese Strategie wurde dahingehend modifiziert, daß beim Erreichen eines lokalen Minimums statt einer Zufallsbewegung

mit einer bestimmten Wahrscheinlichkeit eine Umgreifbewegung für einen der beiden Manipulatoren ausgeführt wird.

Koga konzentriert sich in seinen Arbeiten auf die Integration von Umgreifoperationen in den Planungsvorgang, betrachtet hierzu aus Komplexitätsgründen lediglich einfache 2D-Szenen. Ebenso werden lediglich Greifpunkte (feste Transformationen) und keine zeitabhängigen oder durch Restriktionen beschriebene Transformationen betrachtet.

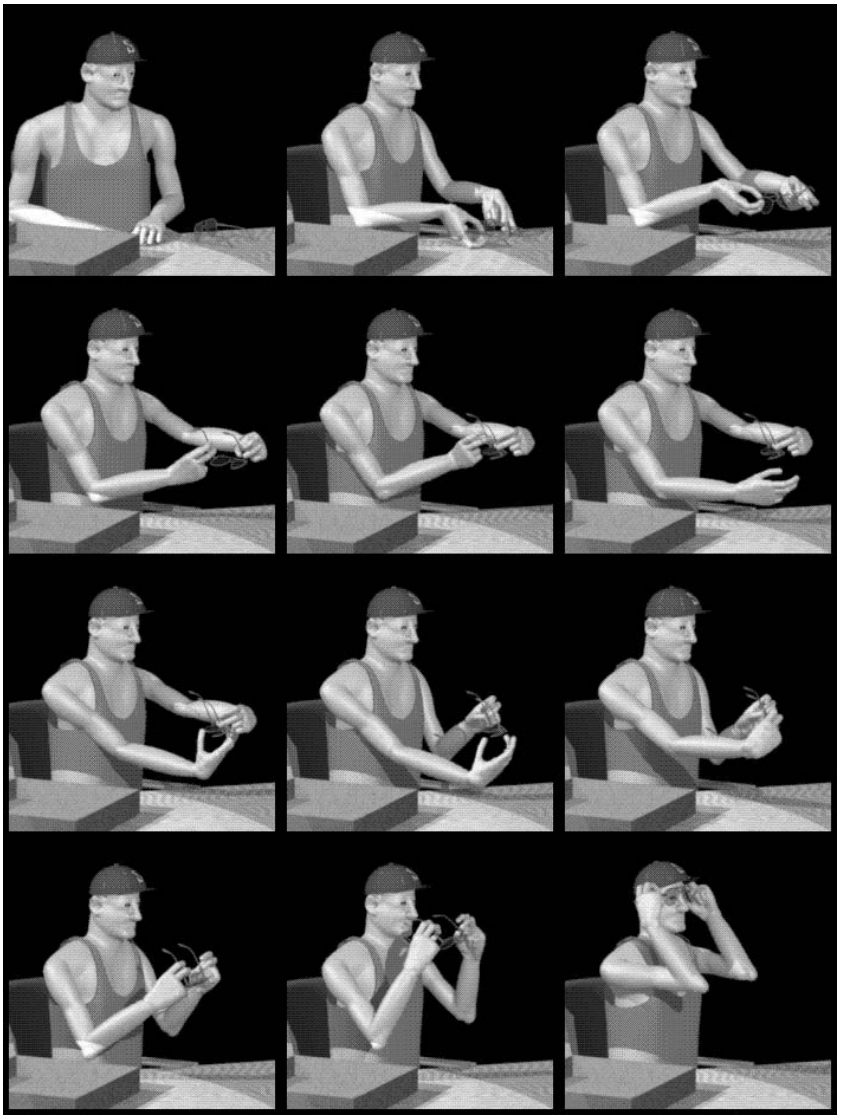


Abbildung 13: Aufsetzen einer Brille (aus [KKKL94])

In einer Fortführung dieser Arbeiten [KKKL94] wird ein Planungsverfahren vorgestellt, das Bewegungen für Transportaufgaben mit Umgreifaktionen in realistischen 3D-Umgebungen

plant. Als Beispiele werden das Aufsetzen einer Brille (s. Abb. 13) bzw. das Umdrehen eines auf einem Tisch liegenden Bretts durch ein Modell eines Menschen vorgestellt. Dabei wird eine endliche Menge möglicher Greifpunkte vorgegeben. Bei der Planung einer Bahn für das manipulierte Objekt – die Darstellung des Konfigurationsraums erfolgt durch die kartesischen Konfigurationen des Objekts – wird eine Liste der Paare von Griffen mitgeführt, die bei der aktuellen Stellung des Objekts kollisionsfrei und durch beide Manipulatoren erreichbar sind. Dabei werden lediglich die Griffpaare betrachtet, die auch schon in der Liste der Vorgängerstellung enthalten waren. Wird diese Liste leer, so gibt es kein Griffpaar, das eine Bewegung von der Startstellung bis zur aktuellen Stellung ermöglicht. Daher wird diese Stellung markiert, die Liste mit allen Griffpaaren initialisiert, und die Planung weitergeführt. Das Ergebnis dieser Planung ist nun eine Sequenz von Bahnstücken, die ohne Umgreifbewegung durchfahren werden können. An den zuvor markierten Übergangsstellen werden anschließend Umgreifbewegungen (*transit paths*) für jeweils einen Manipulator geplant.

Allerdings beschränkt sich auch dieses Verfahren auf *Transport*aufgaben. Relativbewegungen zwischen Effektoren und manipuliertem Objekt, wie sie im Rahmen der vorliegenden Arbeit vorgegeben werden können, können nicht spezifiziert werden.

3.6 Anforderungen

Aus diesen einführenden Betrachtungen können nun zusammenfassend Anforderungen an einen Spezifikationsformalismus für kooperierende Manipulatoren mit abhängigen Effektorkoordinatensystemen sowie an ein Bahnplanungsverfahren zur Umsetzung spezifizierter Aufgaben in kollisionsfreie Bahnen abgeleitet werden. Daran anschließend wird dann ein Spezifikationsformalismus dargestellt.

Zu spezifizierende Aufgabenklassen Im folgenden wird die Aufgabenklasse beschrieben, die durch den in Kap. 4 dargestellten Formalismus spezifiziert werden soll.

AK1 Elementaroperationen

Es werden nur *elementare* Manipulatoraktionen und keine Sequenzen derselben beschrieben. Elementare Manipulatoraktionen sind hierbei Aktionen, die auf der Ebene von Manipulatorbewegungen nicht mehr weiter sinnvoll zerlegt werden können. Insbesondere werden keine komplexen Aufgaben betrachtet, die etwa durch ein Online-Planungssystem abhängig von einem sensoriell erfaßten Ausführungsstatus weiter in alternative Elementaroperationen zerlegt und als unterschiedliche Abläufe ausgeführt werden.

AK2 Aufgaben mit abhängigen Tool-Koordinatensystemen

Inhalt des Spezifikationsformalismus sind nur Aufgaben, die von mehreren Manipulatoren gemeinsam ausgeführt werden, bei denen also Abhängigkeiten zwischen den Effektorkoordinatensystemen auftreten.

Allgemeine Anforderungen In diesem Abschnitt werden, basierend auf der Diskussion der existierenden Ansätze, die Anforderungen zusammengefaßt, die an einen Spezifikationsformalismus zur Beschreibung der hier betrachteten Aufgaben gestellt werden.

AA1 Beschreibung auf unterster ausführungsunabhängiger Ebene

Die Programmierung kooperierender Manipulatoren erreicht durch die in Kap. 3.2.2 erwähnten Gründe (kollisionsfreie Zusammenarbeit, intuitiv nicht mehr erfaßbarer gemeinsamer Arbeitsraum) einen Komplexitätsgrad, der mit herkömmlichen, bewegungsorientierten Programmierverfahren nur noch schwer beherrschbar ist. Der hier dargestellte Spezifikationsformalismus soll daher von der Ausführungsschicht abstrahieren. Dies bedeutet, daß keine Manipulatorbewegungen vorgegeben werden, sondern diese erst von Bahnplanungswerkzeugen generiert werden.

AA2 Möglichst allgemeine, umfassende Beschreibungsmächtigkeit

Es gibt bereits Systeme, die eine aufgabenorientierte Schnittstelle für eine im jeweiligen Fall gegebene, stark eingeschränkte Domäne zur Verfügung stellen (s. Kap. 3.1.1, 3.1.2). Ziel dieser Arbeit ist jedoch die Bereitstellung einer möglichst allgemeinen, domänenunabhängigen Spezifikationsmöglichkeit. Auf dieser Grundlage können dann allgemeine, aufgabenorientiert programmierbare Systeme komfortabel realisiert werden.

AA3 Übersichtlichkeit

Die Fülle der Information, die für eine elementare Manipulatoraktion bereitzustellen ist, soll in übersichtlicher und lesbarer Art und Weise spezifiziert werden können.

AA4 Abstraktion von den verwendeten Manipulatoren

Um eine höhere Flexibilität bei Handhabungsvorgängen und möglichst große Unabhängigkeit von den verwendeten Systemen und der Umgebung, in der die Aufgabe ausgeführt werden soll, zu erreichen, muß die Programmierung von Manipulatoren aufgabenbezogen und nicht mehr manipulatorbezogen erfolgen. Eine *aufgabenorientierte* Aufgabenspezifikation muß von den sich durch die verwendeten Systeme – Manipulator und Manipulatorumgebung – ergebenden Erfordernissen möglichst abstrahieren [Lev88]. Die Abbildung auf die aktuelle Umgebung muß durch entsprechende Planungswerkzeuge erfolgen.

AA5 Spezifikation der eine Aufgabe beschreibenden Information

Zur Beschreibung einer Aufgabe müssen die Manipulatoren, die manipulierten Objekte und die Umgebung, in der eine Aufgabe ausgeführt wird, in einer geeigneten Art und Weise dargestellt werden. Desweiteren herrschen zwischen den Manipulatoren, den manipulierten Objekten und der Umwelt verschiedene Arten von Beziehungen, die ebenfalls beschrieben werden müssen:

- Eine fundamentale Beziehung sind Bewegungen. Bewegungen treten in verschiedener Form auf, etwa bei der Bearbeitung eines Werkstücks oder bei der sensoriellen Erfassung eines Objekts, etwa der Beobachtung durch eine Kamera. Bewegungen in allen auftretenden Arten müssen durch den Formalismus spezifiziert werden können. Dies impliziert ebenfalls die Beschreibung zu verfahrender Geschwindigkeiten und Beschleunigungen.

- Allerdings sind Beziehungen zwischen den beteiligten Objekten im allgemeinen nicht nur durch feste Relativpositionen oder Bewegungen zwischen Objekten charakterisiert. Vielmehr können Beziehungen in einigen Freiheitsgraden unspezifiziert oder auf bestimmte Bereiche beschränkt sein. Beispiele derartiger Beziehungen sind etwa die Inspektion eines Objekts durch eine Kamera, wobei der Blickwinkel und der Kameraabstand in vorgegebenen Bereichen variieren können, oder die Forderung, daß ein flüssigkeitsgefüllter Behälter beim Transport nicht gekippt werden darf. Derartige Einschränkungen müssen ebenfalls Bestandteil einer Aufgabenspezifikation sein.

- Die Reaktion auf Umweltereignisse und die Berücksichtigung des Zustands der Umgebung bei der Aufgabenausführung ist grundlegend für die Autonomie aufgabenorientiert programmierbarer Systeme [FK85, Lev88, Wer93]. Ebenso ist die Berücksichtigung von in geschlossenen kinematischen Ketten auftretenden Kräften und Momenten grundlegend für die Programmierung kooperierender Manipulatoren. In dem hier vorgestellten Spezifikationsformalismus ist somit eine Schnittstelle zur Integration von Sensorinformation, insbesondere von Kräften und Momenten, zu realisieren.

Anforderungen an das Bahnplanungsverfahren Im folgenden werden die Anforderungen zusammengefaßt, die bei der Entwicklung von Bahnplanungsstrategien zur Umsetzung spezifizierter Aufgaben in kollisionsfreie Bahnen maßgeblich sind.

PL1 Planung kollisionsfreier, im Arbeitsbereich der Manipulatoren liegender Bahnen

Der Planungsalgorithmus soll ein Paar von zeitlich synchron zu verfahrenden Bahnen liefern. Bei den gefundenen Lösungen dürfen keine Kollisionen zwischen den beteiligten Objekten auftreten, es sei denn, die Aufgabenstellung macht einen Kontakt zwischen Objekten erforderlich (etwa Greifpunkte, oder bei Montageaufgaben oder Transportaufgaben, bei denen ein Objekt nur auf einer Oberfläche entlanggeschoben werden darf). Desweiteren müssen alle auftretenden Manipulatorkonfigurationen innerhalb der Gelenkgrenzen der verwendeten Manipulatoren liegen.

PL2 Planung für realistische Umgebungen

Die Planungsstrategien sollen so konzipiert werden, daß sie für Planungsaufgaben in realistischen Umgebungen eingesetzt werden können. Dies erfordert zum einen die Beherrschung einer genügend großen Anzahl von Freiheitsgraden: Die Planung erfolgt in einer dreidimensionalen Umgebung, in der sich Objekte in sechs Freiheitsgraden bewegen können. Die verwendeten Manipulatoren sollen daher ebenfalls über sechs Freiheitsgrade verfügen. Desweiteren muß in Betracht gezogen werden, daß Kollisionstests im Umweltmodell eine beschränkende Ressource darstellen, die die Effizienz der Planungsverfahren entscheidend beeinflußt.

PL3 Effiziente Lösungen zum Einsatz in einem Online-Planungssystem

Die Entwicklung der in Kap. 6 beschriebenen Planungsstrategien erfolgt vor dem Hintergrund der Anwendbarkeit in der Realität, als Bestandteil eines Online-Planungssystems. Daher steht die Effizienz der Planungsstrategien im Vordergrund. Wie später gezeigt wird, kann dies nur zu Lasten der Vollständigkeit der Planung gehen, daß also für manche Aufgaben keine Lösung gefunden wird, obwohl eine existiert. Dennoch sollen möglichst viele realistische Aufgaben gelöst werden können.

Desweiteren werden in realistischen Anwendungen nur selten mehrere Planungen in derselben Umgebung durchgeführt. Manipulationsvorgänge führen im allgemeinen Veränderungen in der Umgebung herbei. Planungsalgorithmen, die auf einer meist zeitraubenden Vorverarbeitung der Umgebung basieren und dadurch eine hohe Effizienz für Planungen in derselben unveränderten Umgebung erzielen, sind für praktische Anwendungen nicht geeignet.

PL4 Aufgaben mit zeitlich veränderlichen Abhängigkeiten zwischen Koordinatensystemen

In einigen Beziehungen zwischen Objekten treten durch spezifizierte Relativbewegungen zeitliche Abhängigkeiten auf. Das Bahnplanungsverfahren muß in der Lage sein, diese zu behandlen.

PL5 Berücksichtigung geometrischer Restriktionen

Im allgemeinen sind bei der Planung realistischer Aufgaben geometrische Einschränkungen an die Lösungen zu berücksichtigen. Die Planungsstrategien sollen angegebene geometrische Einschränkungen berücksichtigen.

PL6 Aufgaben mit Zielräumen als Planungsziel

Im allgemeinen werden durch die spezifizierten Aufgaben als Planungsziele nicht einzelne Zielkonfigurationen, sondern Räume von Konfigurationen vorgegeben. Diese müssen von einem Bahnplaner behandelt werden können.

4 Aufgabenspezifikation für kooperierende Manipulatoren

Die Spezifikation von Elementaroperationen für kooperierende Manipulatoren soll von den ausführenden Manipulatoren abstrahieren und beschreiben, *was* zur Durchführung einer Aufgabe erforderlich ist, nicht *wie* dies zu geschehen hat. Im folgenden werden daher Elementaroperationen formal beschrieben. Aus dieser formalen Beschreibung wird dann in Kap. 5 eine formale Sprache zur Spezifikation von Elementaroperationen für kooperierende Manipulatoren abgeleitet.

Zur formalen Beschreibung von Elementaraufgaben sind nun zum einen verschiedene Objekte durch ihre Lage und ihre Beziehungen zueinander zu beschreiben. Entsprechend dem in Kap. 2.2.1 beschriebenen Frame-Konzept erfolgt die Darstellung der Lage durch ihnen zugeordnete Ursprungskoordinatensysteme. Weiterhin sind die die Aufgabe durchführenden Manipulatoren mit ihren Werkzeugen zu beschreiben. Da nach Anforderung **AA4** in der Aufgabenspezifikation von den Manipulatoren abstrahiert wird, werden die Manipulatoren lediglich durch ein einziges, für die Aufgabenausführung relevantes Koordinatensystem dargestellt. Dies ist das Effektorkoordinatensystem, das auch als Toolkoordinatensystem oder Tool Center Point (TCP) bezeichnet wird.

Die Aufgabe kann dann durch eine Beschreibung der Beziehungen der beteiligten Objekte zueinander spezifiziert werden. Allerdings ist für die Beschreibung der Beziehungen eine hohe Ausdrucksmächtigkeit erforderlich. Die Anforderungen an einen Beschreibungsformalismus für die Beziehungen werden im folgenden näher untersucht. Die in einer Aufgabenspezifikation auftretenden Beziehungen können, wie nachfolgend in einigen Beispielen gezeigt wird, recht unterschiedlicher Natur sein. Betrachten wir etwa eine Transportaufgabe (Abb. 14).

Die Beziehung zwischen dem Weltkoordinatensystem und dem Ursprungskoordinatensystem des zu transportierenden Objekts ist dadurch gekennzeichnet, daß sie zu Aufgabenbeginn die Startstellung des Objekts beschreibt und am Aufgabenende dessen Endstellung. Es können aber auch noch weitere Anforderungen an sie gestellt werden. Ist etwa ein flüssigkeitsgefüllter Behälter zu transportieren, so darf die Z-Achse des Behälters während des Transports nicht mehr als um einen bestimmten Neigungswinkel α gegenüber der Z-Achse des Weltkoordinatensystems gekippt werden. Weitere bei einer Transportaufgabe auftretende Beziehungen sind die Transformationen zwischen dem jeweiligen Greifer und dem transportierten Objekt. Diese sind während der Aufgabenausführung fest, allerdings muß über sie der Sachverhalt beschrieben werden, daß während des Transports keine Kräfte oder Momente auftreten, die zu Beschädigungen des transportierten Objekts führen.

Als weiteres Beispiel soll eine Montageaufgabe dienen (Abb. 15). Eine weitere interessante Beziehung tritt zwischen den zu montierenden Werkstücken auf. Diese Beziehung, beispielsweise eine von einem Montageplaner erzeugte Montagetrajektorie, ist eine Funktion der Zeit. In manchen Richtungen bzw. Freiheitsgraden sind zudem bei der Montage auf-

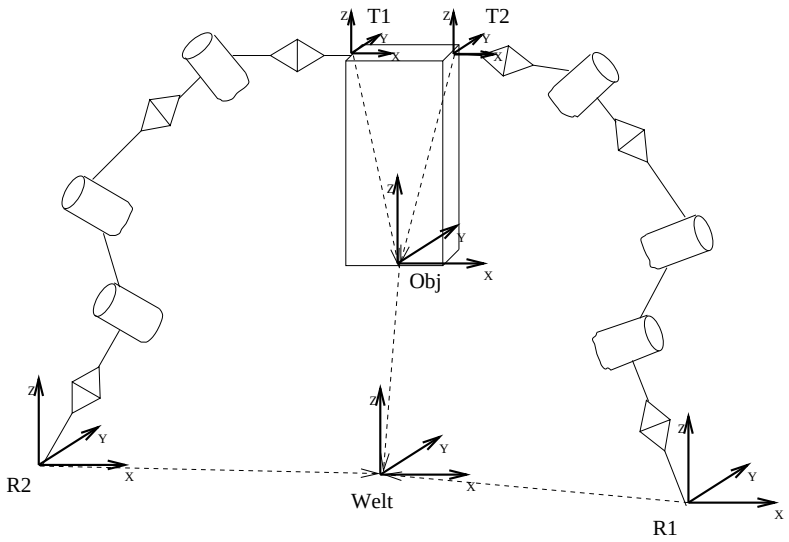


Abbildung 14: Schema einer Transportaufgabe

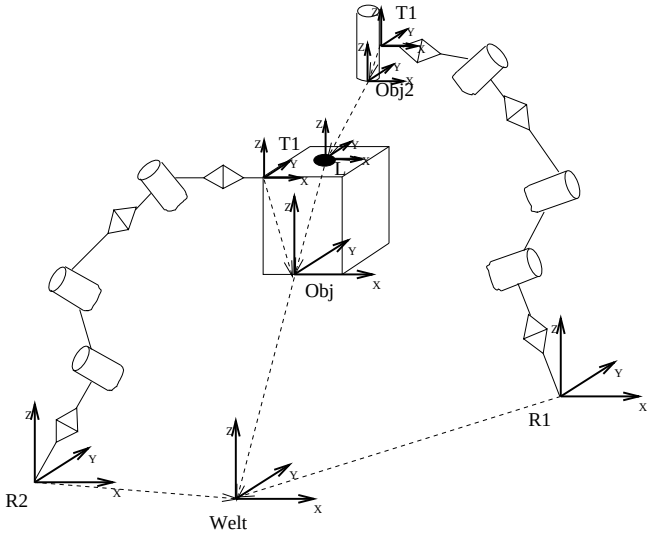


Abbildung 15: Schema einer Montageaufgabe

tretende Kräfte und Momente zu berücksichtigen. Diese können, wie das bereits in Kapitel 3.3 erwähnte Beispiel eines in eine Passung einzufügenden Bolzens zeigt, wiederum von zwei verschiedenen Arten sein: Zum einen müssen die beim Einführen des Bolzens auftretenden Kräfte und Momente kompensiert werden. Zum anderen darf die Einführbewegung nur solange erfolgen, bis der Bolzen bis zum Ende der Passung eingefügt worden ist. Diese beiden Arten, auf Sensorinformation zu reagieren (s. Kap. 4.1) sind bei der Spezifikation vorzusehen.

Als weiteres Beispiel sei eine Bearbeitungsaufgabe (Bild 16) beschrieben: Ein Manipulator soll mittels eines Schleifgeräts den Gußgrat an einem Werkstück entfernen. Für den Fall,

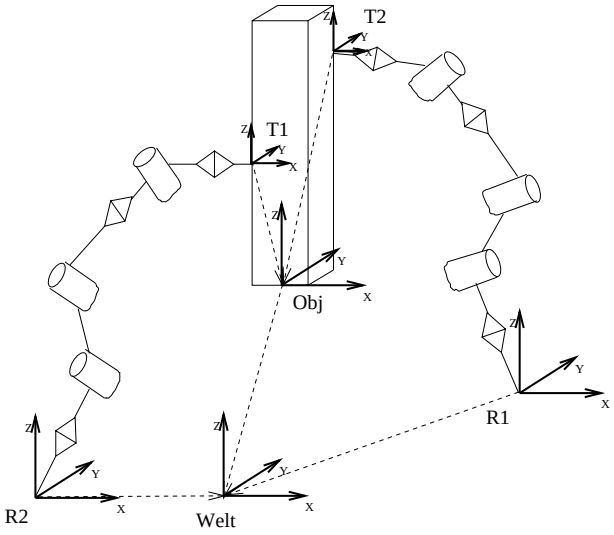


Abbildung 16: Schema einer Bearbeitungsaufgabe

daß die zu verfahrende Bearbeitungstrajektorie den Arbeitsraum des bearbeitenden Manipulators übersteigt (etwa bei starken Orientierungsänderungen oder langen Teilstücken) kann ein zweiter Manipulator das Werkstück während der Bearbeitung kontinuierlich innerhalb des Arbeitsraums des ersten Manipulators halten. Die Bearbeitungstrajektorie stellt eine Beziehung zwischen dem zu bearbeitenden Werkstück und dem bearbeitenden Tool dar und ist ebenfalls eine Funktion der Zeit.

Ein abschließendes Beispiel ist die Beobachtung eines durch einen Manipulator bewegten Objekts durch eine an einem zweiten Manipulator befestigte Kamera. Die Transformation zwischen der Kamera und dem beobachteten Objekt kann hierbei relativ frei spezifiziert werden, etwa, daß die Kamera zwischen 10 und 30 cm vom zu beobachtenden Objekt entfernt sein soll, und daß der Ursprung des Objekts nicht weiter als 3 cm von der optischen Achse der Kamera entfernt sein soll.

Analysiert man Beispiele dieser Art, so erkennt man, daß sehr verschiedene Arten von Beziehungen auftreten können:

- während der gesamten Aufgabenausführung konstante Transformationen
- sich während der Aufgabenausführung verändernde (zeitabhängige) Transformationen
- nur zu ausgezeichneten Zeitpunkten ist genau eine Transformation vorgegeben
- die zulässigen Transformationen sind durch geometrische Einschränkungen wie “das Objekt darf beim Transport nicht gekippt werden”, “die Kamera soll zwischen 10 cm und 30 cm vom Beobachtungspunkt entfernt sein” oder “der Block soll auf der Tischoberfläche entlanggeschoben werden” charakterisiert

- die zulässigen Transformationen sind durch Bedingungen über Sensorinformation beschrieben (z.B. *Compliant Motion*)
- verschiedene Freiheitsgrade sind unterschiedlich spezifiziert (z.B.: einige Freiheitsgrade sind exakt spezifiziert, andere bleiben offen oder einige Freiheitsgrade sind geometrisch spezifiziert, andere über Sensorinformation).

Im folgenden werden nun die zu beschreibenden Aufgaben formal dargestellt. Anschließend wird eine Spezifikationssprache beschrieben, mit der die Struktur der Aufgabe, das sind die auftretenden Koordinatensysteme und die Beziehungen, die zwischen diesen auftreten, spezifiziert wird. Mit dem vorgestellten Formalismus ist eine einheitliche Beschreibung von Beziehungen zwischen Koordinatensystemen möglich, die durch

- zeitliche Randbedingungen,
- geometrische Randbedingungen und
- auf Sensorinformation basierenden Randbedingungen

spezifiziert werden (Anforderung **AA5**). Als Spezialfall des letzten Punktes wird die Behandlung von Kräften und Momenten ausführlich untersucht, da die Behandlung von Kraftschleifen bei kooperierenden Manipulatoren ein bei den meisten Aufgaben auftretendes Teilproblem darstellt.

4.1 Einbeziehung von Sensorinformation in die Aufgabenbeschreibung

Eine der Anforderungen an einen Spezifikationsformalismus ist, daß die Einbeziehung von Sensorinformation in einen Spezifikationsformalismus erforderlich ist. In diesem Kapitel wird dargelegt, in welcher Form Sensorinformation darzustellen und zu spezifizieren ist.

Nach [Hag92] lassen sich Aktionen, die Sensorinformation berücksichtigen, in drei Klassen einteilen:

- *Sensorgesteuerte Aktionen* "sind dadurch gekennzeichnet, daß die Erfassung von Sensordaten und die Ausführung der von den Sensorresultaten abhängigen Aktion zeitlich getrennt ablaufen" [Hag92]. Die Integration der Sensorinformation erfolgt hierbei in der Planungsebene, nicht in der Ausführungsebene. Elementaraufgaben werden hierbei unter Berücksichtigung vorab erfaßter Sensorinformation (etwa der Lage eines zu greifenden Objekts) geplant. Sensorinformation wird somit nur bei der *Planung*, nicht aber bei der *Ausführung* einer Aufgabe berücksichtigt.
- *Sensorgeführte Aktionen*. Diese sind dadurch gekennzeichnet, daß die Sensorik "Bedingungen für einen erfolgreichen Abschluß einer Aktion" [Hag92] erfaßt.

- *Sensorgeregelte Aktionen.* Hierbei "dient sensorielle Information nicht nur dazu, Bedingungen für die Beendigung einer Aktion zu erkennen, sondern es werden laufend Sensorergebnisse in das Agieren einer aktiven Komponente rückgekoppelt. Dabei existiert immer eine Vorgabe für eine Bewegung oder eine Aktion, die je nach Einfluß einwirkender Ungewißheiten zur Ausführungszeit korrigiert wird" [Hag92].

Aus dieser Klassifikation kann abgeleitet werden, daß Sensorinformation bei der Spezifikation von Elementaraufgaben in zwei Bereichen verwendet wird, nämlich um

- Terminierungsbedingungen zu erkennen und
- durch Ungewißheiten entstehende Abweichungen laufend zu kompensieren (*auszuregeln*).

Im Bereich Fertigungsautomation eingesetzte Sensorik läßt sich in die Klassen

- Entfernungssensoren
- Taktile Sensoren (inklusive Kraft- und Momentensensoren)
- Sichtsensoren

einteilen [GWNO87]. In der Aufgabenbeschreibung wird Sensorinformation in vorverarbeiteter Form betrachtet, bei Bildern also etwa auf Featureebene, in der die Position und Orientierung einzelner Bildfeatures (etwa Eckpunkte von Objekten, spezielle Marken usw.) vorliegt. Zur formalen Aufgabenbeschreibung und zur Einbeziehung in den Spezifikationsformalismus ist die in einer Elementaraufgabe verwendete Sensorinformation **SI** formal zu beschreiben: Die für eine Aufgabe relevanten sensoriell erfaßbaren Ungewißheiten können durch eine für eine Aufgabe feste Anzahl n kontinuierlicher Größen beschrieben werden, also

$$\mathbf{SI} \subset \mathbb{R}^n$$

Die Sensorinformation wird dabei relativ zu einem Bezugskoordinatensystem angegeben. Für jede Komponente der Sensorinformation existiert ein Selektor, der den entsprechenden reellwertigen Bestandteil der Sensorinformation extrahiert.

Als Beispiele für in einer Aufgabe verwendete Sensorinformation seien Abstandswerte (durch einen Abstandssensor erfaßt), Position und Orientierung eines Objekts in einem vorverarbeiteten Bild oder durch einen Kraft-Momenten-Sensor erfaßte Kräfte und Momente erwähnt. Im letzteren Fall gilt: $\mathbf{SI} = \mathbb{R}^6$. Die Selektoren (`FORCE(TX)`, `FORCE(TY)`, `FORCE(TZ)`, `TORQUE(RX)`, `TORQUE(RY)`, `TORQUE(RZ)`) liefern die Kräfte und Momente entlang der sechs Raumrichtungen eines Bezugskoordinatensystems.

Mittels der Selektoren werden in einer Aufgabenbeschreibung die während der Aufgabenausführung durch Regelung einzuhaltenden Sollwerte der entsprechenden Sensorinformation und die die Aufgabe beendenden Terminierungsbedingungen formuliert (s. Kap. 4.4).

Die spezifizierten Sollvorgaben für Kräfte und Momente dienen als Führungsgrößen für einen Regler, dessen Regelungsvorschrift aus der spezifizierten Aufgabenstruktur zu generieren ist [BAH94], und der die durch die Sensorik aufgenommenen Sensorwerte in einen Positionsvektor zur Bahnkorrektur für die Manipulatoren umsetzt (s. Kap. 3.3).

4.2 Formale Aufgabenbeschreibung

Als Basis einer formalen Aufgabenbeschreibung wird das in Kap. 2.2.1 beschriebene Framekonzept verwendet, insbesondere dessen Eigenschaft, daß eine Situation geometrisch durch eine Menge von Koordinatensystemen und sie in Beziehung setzende Transformationen beschrieben wird. Dadurch werden die Positionen und Orientierungen der zu einer Szene gehörenden Objekte festgelegt. Die Beschreibung mit Hilfe des Framekonzepts verzichtet hierbei auf eine Einbeziehung der Zeit, eine Situation stellt den Zustand einer Aufgabe zu einem bestimmten Zeitpunkt dar. Übergänge zwischen diesen Situationen werden durch Operationen auf den Frames, den Bewegungsbefehlen, beschrieben. Im Gegensatz zu dieser prozeduralen Methode der Aufgabenbeschreibung wird im folgenden ein deklarativer Ansatz vorgestellt. Dazu ist das Framekonzept in zwei Punkten zu modifizieren:

Zum einen werden die Transformationen des Framekonzepts so erweitert, daß nicht mehr nur einzelne Situationen zu festen Zeitpunkten durch sie beschrieben werden. Vielmehr werden durch diese erweiterten Transformationen (im folgenden als *Beziehungen* bezeichnet) die Abhängigkeiten zwischen zwei Koordinatensystemen während des gesamten Aufgabenverlaufs beschrieben.

Desweiteren wird im Framekonzept durch Frames die Lage eines Koordinatensystems relativ zu *genau einem* Bezugskoordinatensystem eindeutig zu einem bestimmten Zeitpunkt beschrieben. Diese Darstellungsart ist hier nicht mehr ausreichend. Da nicht nur die Lage von Objekten beschrieben wird, sondern alle für eine Aufgabe relevanten Beziehungen, gibt es Koordinatensysteme, die durch mehrere Beziehungen (bzw. Folgen von Beziehungen) relativ zum Weltkoordinatensystem beschrieben werden. Als Beispiel sei eine Transportaufgabe angeführt (Abb. 17), bei der das Bezugskoordinatensystem des zu transportierenden Objekts durch eine Beziehung zum Weltkoordinatensystem *a)* (Start- und Zielstellung) als auch durch die beiden Beziehungssequenzen *b)* und *c)* (*Welt - Manipulatorbasis1/2 - Tool1/2 - Greifpunkt1/2 - Objekt*) beschrieben wird. Im Framekonzept treten derartige Schleifen von Beziehungen implizit dadurch auf, daß bei Bewegungsbefehlen beim Abschluß der Bewegung das Toolkoordinatensystem des Manipulators mit dem spezifizierten Frame in Übereinstimmung gebracht werden soll. Die Beschreibung eines Koordinatensystems durch mehrere Beziehungen kann durch eine Darstellung der Aufgabe als Graph geleistet werden, wie sie in Kap. 5.1 vorgestellt wird.

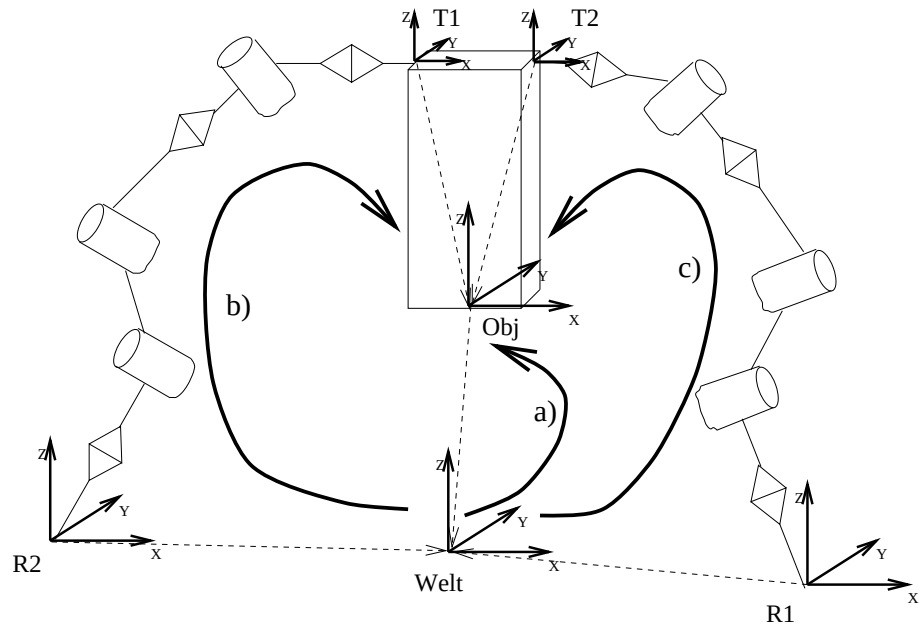


Abbildung 17: Beschreibung eines Koordinatensystems durch mehrere Beziehungen

Aufgabe Formal wird eine Aufgabe **A** durch eine Menge von Koordinatensystemen **F**, eine Menge von Beziehungen **B**, die jeweils einem Paar von Koordinatensystemen einen *zeitabhängigen Transformationsraum* zuordnen, eine Menge von Randbedingungen (*Constraints*) **C**, die die zeitabhängigen Transformationsräume beschreiben, und eine Menge von Terminierungsbedingungen **TB**, spezifiziert:

$$\mathbf{A} = (\mathbf{F}, \mathbf{B}, \mathbf{C}, \mathbf{TB}) \qquad mit \qquad \mathbf{F} = \{F_1, \cdots, F_L\}, \quad \mathbf{B} = \{B_1, \cdots, B_M\}, \\ \mathbf{C} = \{C_1, \cdots, C_N\}, \quad \mathbf{TB} = \{TB_1, \cdots, TB_O\}$$

Die formale Beschreibung der Beziehungen, zeitabhängigen Transformationsräume und Constraints wird im folgenden dargestellt, auf die Terminierungsbedingungen wird in Kap. 5.5.1 näher eingegangen.

Beziehungen Eine *Beziehung* B_{ij} ordnet zwei Koordinatensystemen einen zeitabhängigen Transformationsraum zu, sagt also aus, daß eine Relation zwischen den beiden Koordinatensystemen F_i und F_j besteht. Diese Relation wird durch den zugehörigen zeitabhängigen Transformationsraum $\hat{T_{ij}}$ spezifiziert, der zu jedem Zeitpunkt beschreibt, welche Transformationen das Koordinatensystem F_i in F_j überführen.

Formal kann eine Beziehung als

$$B_{ij} = (F_i, F_j, \hat{T_{ij}})$$

dargestellt werden. Wie $\hat{T_{ij}}$ spezifiziert ist, also welche Transformationen zu welchem Zeitpunkt zulässig sind, wird durch eine Menge von Constraints beschrieben.

Zeitabhängige Transformationsräume Um nun Sachverhalte auszudrücken, wie sie durch Beziehungen zu beschreiben sind und in Kap. 4 dargestellt wurden, muß eine über die statischen Beschreibungsmöglichkeiten des Framekonzepts hinausgehende Beschreibungsmethode verwendet werden. Hierzu werden *zeitabhängige Transformationsräume* eingeführt, die folgende Eigenschaften haben:

- Zu einem bestimmten Zeitpunkt beschreibt ein zeitabhängiger Transformationsraum nicht nur einzelne Transformationen, sondern *Mengen* von Transformationen. Der Zeitbegriff wird später ausführlich diskutiert.
- Die zu einem zeitabhängigen Transformationsraum gehörenden Transformationen werden *zeitabhängig* beschrieben; zu verschiedenen Zeitpunkten können verschiedene Mengen von Transformationen zum zeitabhängigen Transformationsraum gehören.
- Die zu einem zeitabhängigen Transformationsraum gehörenden Mengen von Transformationen werden durch Randbedingungen spezifiziert.

Formal ist ein zeitabhängiger Transformationsraum $\hat{T}$ eine Menge von Paaren aus Transformationen und Zeitpunkten:

$$\hat{T} \subset \mathbf{T} \times t \qquad t = [0, 1]$$

Dabei bezeichnet t die formale Zeit, $\mathbf{T}$ die Menge der homogenen Transformationen.

In Abb. 18 sind mehrere Beispiele zeitabhängiger Transformationsräume dargestellt. Von links nach rechts sind abgebildet: a) für den Start- und Zielzeitpunkt ist jeweils eine feste Transformation spezifiziert (gestrichelte Kreuze), dazwischen sind alle Transformationen zulässig (grau schraffierter Bereich). Durch einen derartigen zeitabhängigen Transformationsraum kann die Stellung eines zu transportierenden Objekts bei einer Transportaufgabe spezifiziert werden. In b) ist ein zeitabhängiger Transformationsraum dargestellt, der zu jedem Zeitpunkt eine feste, aber zeitlich veränderliche Transformation darstellt. Dies entspricht einer Bearbeitungstrajektorie. Der zeitabhängige Transformationsraum in c) realisiert eine konstante Transformation, beispielsweise einen Greifpunkt. In d) ist ein zeitabhängiger Transformationsraum dargestellt, der die erlaubten Transformationen auf einen zulässigen Bereich einschränkt, etwa zur Realisierung einer Einschränkung der Form: "Das Objekt darf während der Bearbeitung nicht gekippt werden".

Constraints Die Beschreibung zeitabhängiger Transformationsräume erfolgt durch eine Constraintmenge. Neben geometrischen Sachverhalten werden von den Constraints ebenfalls die Sollvorgaben für den von der Sensorik erfaßten Umgebungszustand beschrieben, somit ist die Sensorinformation $si \in \mathbf{SI}$ ebenfalls Bestandteil der Constraints. Ein Constraint C_j beschreibt die n_j zeitabhängigen Transformationsräume $\hat{T}_{c_j(1)}$ bis $\hat{T}_{c_j(n_j)}$, indem

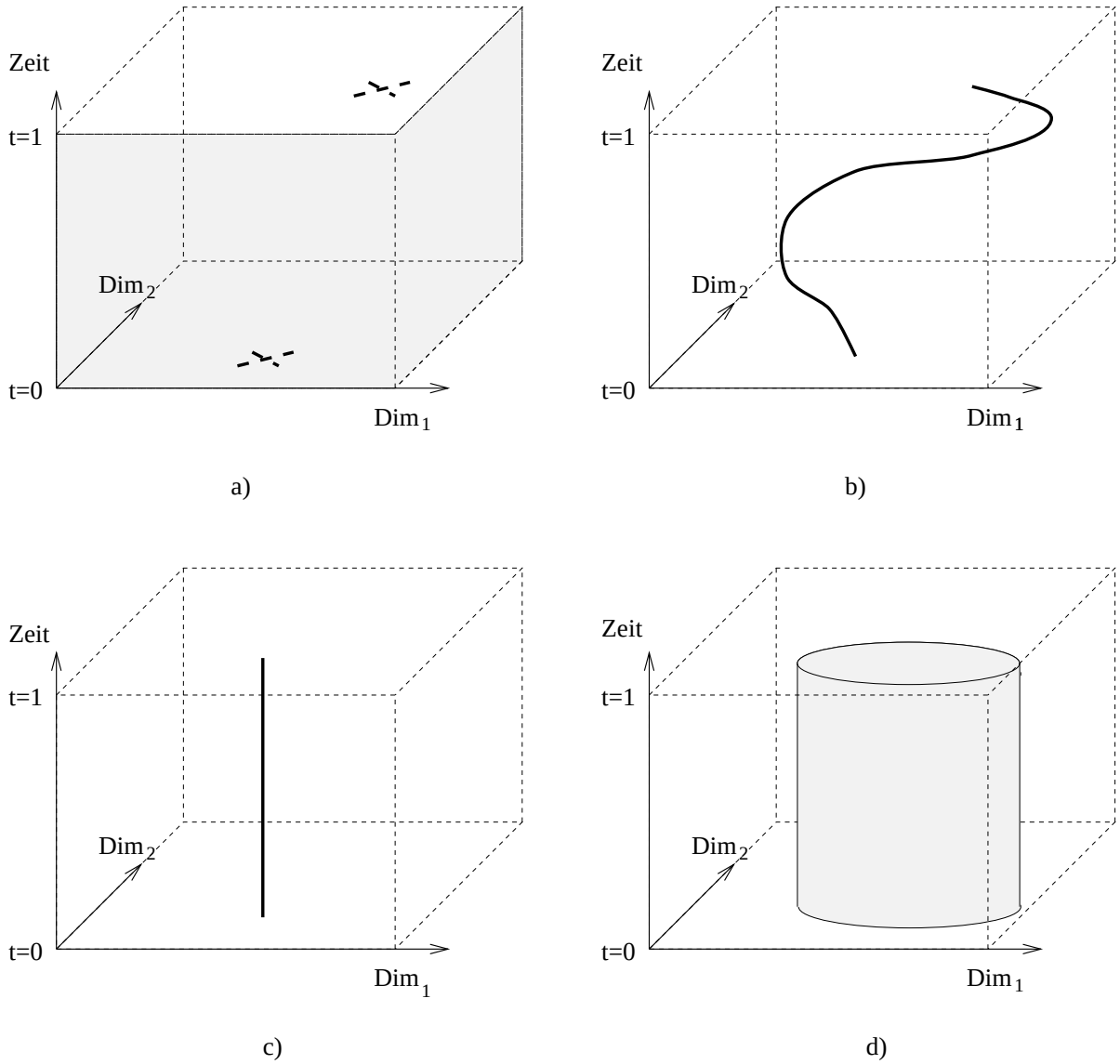


Abbildung 18: Beispiele für zeitabhängige Transformationsräume

es die Menge der zu diesen gehörenden Paare aus Transformationen und Zeitpunkten einschränkt. Dabei ist $c_j(k)$ die Nummer des k-ten, von C_j beschriebenen zeitabhängigen Transformationsraums, n_j die Anzahl der von C_j beschriebenen zeitabhängigen Transformationsräume. C_j ist somit eine Abbildung

$$C_j : \begin{cases} \mathbf{T}^{n_j} \times \mathbf{SI} \times [0,1] & \rightarrow & \{\mathbf{true}, \mathbf{false}\} \\ (T_{c_j(1)}, \cdots, T_{c_j(n_j)}, si, t) & \rightarrow & C_j(T_{c_j(1)}, \cdots, T_{c_j(n_j)}, si, t) \end{cases}$$

Sie besagt, ob ein Tupel von Transformationen $(T_{c_j(1)}, \cdots, T_{c_j(n_j)})$ zu einem Zeitpunkt t

die von ihr beschriebenen voneinander abhängigen zeitabhängigen Transformationsräume erfüllt. Die zu den in einer Aufgabe auftretenden Beziehungen gehörenden zeitabhängigen Transformationsräume werden von den Constraints als jeweils eine Menge von zu einer bestimmten Zeit gültigen Transformationen beschrieben, so daß zu jedem Zeitpunkt t gilt:

$$\begin{aligned}&\Big((T_1,t)\in\hat{T}_1\wedge\cdots\wedge(T_M,t)\in\hat{T}_M\Big)\Leftrightarrow\\ \Leftrightarrow&\Big(C_1(T_{c_1(1)},\cdots,T_{c_1(n_1)},si,t)\wedge\cdots\wedge C_N(T_{c_N(1)},\cdots,T_{c_N(n_N)},si,t)\Big)\end{aligned}$$

Dabei bezeichnet M die Anzahl der in der Aufgabe auftretenden zeitabhängigen Transformationsräume, N die Anzahl der diese beschreibenden Constraints.

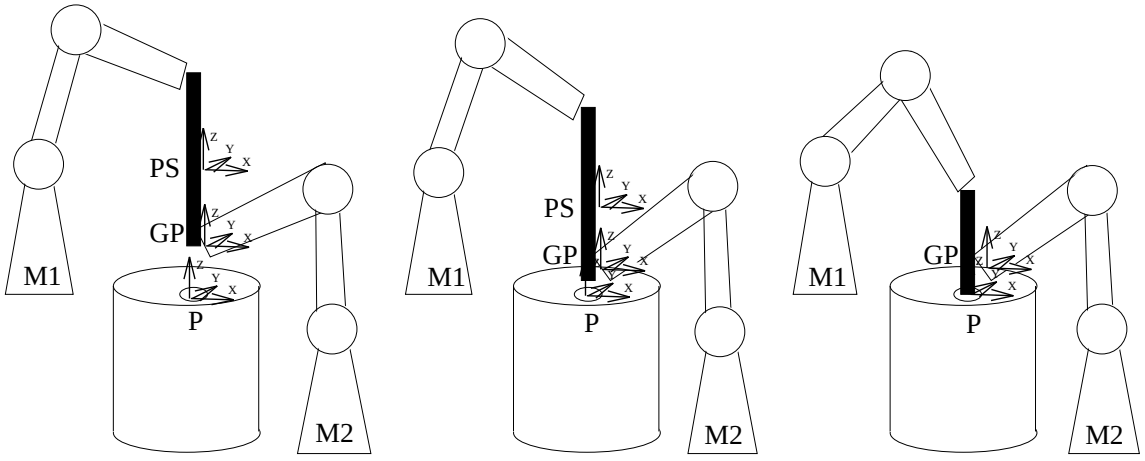


Abbildung 19: Einführen eines Peilstabs

Ein konstruiertes Beispiel für derart zu spezifizierende, voneinander abhängige Beziehungen stellt folgendes Szenario dar (Abb. 19). Ein flexibler Peilstab **PS** soll in einen Tank eingeführt werden. Um ein Einführen des Stabs, der wegen seiner Flexibilität zum Schwingen neigt, zu ermöglichen, wird er von einem Manipulator **M2** an seiner Spitze (die d Einheiten vom Ursprungskoordinatensystem des Stabs entfernt ist) so gegriffen, daß der Greifpunkt **GP** entlang des Stabs verschiebbar ist. Der Stab soll nun bis zu einem Sicherheitsabstand sa über der Peilöffnung **P** des Tanks an der Spitze gehalten werden, ab dann soll sich der Greifpunkt so verschieben, daß der Sicherheitsabstand eingehalten wird. Dieser Sachverhalt kann über ein Constraint beschrieben werden, das die Beziehungen (**PS**, **P**) und (**PS**, **GP**) gemeinsam beschreibt.

Sei ${}^{PS}T_P$ die Transformation, die das Koordinatensystem P im Bezugskoordinatensystem PS beschreibt (${}^{PS}T_{GP}$ analog) und $Z(T)$ die translatorische Komponente in Z-Richtung der Transformation T, so kann dieses Constraint formuliert werden als:

$$((Z({}^{PS}T_{GP})-Z({}^{PS}T_P)>sa)\wedge(Z({}^{PS}T_{GP})=-d))\vee(Z({}^{PS}T_{GP})-Z({}^{PS}T_P)=sa)$$

Die Spezifikation dieses Beispiels findet sich in Anhang C, Beispiel 3.

Eine Klasse von Beziehungen tritt in der Realität sehr häufig auf und wird daher gesondert betrachtet. Es sind dies Beziehungen, bei denen der zugehörige zeitabhängige Transformationsraum durch eine eigene Constraintmenge unabhängig von anderen Beziehungen beschrieben wird. Ein Constraint beschreibt nunmehr *genau einen* zeitabhängigen Transformationsraum, ist also von der Form

$$\hat{C}_i : \begin{cases} \mathbf{T} \times \mathbf{SI} \times [0,1] & \rightarrow & \{\mathbf{true}, \mathbf{false}\} \\ (T, si, t) & \rightarrow & \hat{C}_i(T, si, t) \end{cases}$$

Für die Menge $\hat{\mathbf{C}}$ dieser Constraints gilt

$$\hat{\mathbf{C}} \subset \mathbf{C}$$

Sie werden im folgenden Sprachentwurf wegen ihres häufigen Auftretens gesondert behandelt.

Die hier verwendete Darstellung der Constraints ist sehr allgemein. Für den Entwurf einer Sprache müssen sie allerdings weiter in eine leichter handhabbare Form zerlegt werden.

Die Menge der *elementaren Constraints* $\bar{\mathbf{C}}$ besteht aus Gleichungen und Ungleichungen über reellwertigen Ausdrücken, insbesondere über Abbildungen f der Form

$$f : \begin{cases} \mathbf{T} \times \mathbf{SI} \times [0,1] & \rightarrow & \mathbb{R} \\ (T, si, t) & \rightarrow & f(T, si, t) \end{cases}$$

Die Realisierung dieser Abbildungen durch konkrete Operationen wird im Rahmen der Spezifikationssprache in den Kapiteln 5.2 und 5.3 beschrieben.

Sind f, g derartige Abbildungen, so gilt

$$\bar{\mathbf{C}} = \{f \ \mathbf{op} \ g \quad | \quad \mathbf{op} \in \{=, <, >, \leq, \geq\}\}$$

Komplexe Constraints sind boolesche Ausdrücke über elementaren Constraints. Sei $\mathbf{C}$ die Menge der komplexen Constraints, $\bar{\mathbf{C}}$ die Menge der elementaren Constraints, dann gilt:

$$\bar{\mathbf{C}} \subset \mathbf{C} \tag{3}$$

$$c_1, c_2 \in \bar{\mathbf{C}} \ \Rightarrow \ \neg c_1 \in \mathbf{C} \tag{4}$$

$$c_1, c_2 \in \bar{\mathbf{C}} \ \Rightarrow \ c_1 \vee c_2 \in \mathbf{C} \tag{5}$$

$$c_1, c_2 \in \bar{\mathbf{C}} \ \Rightarrow \ c_1 \wedge c_2 \in \mathbf{C} \tag{6}$$

4.3 Die formale Zeit

Durch zeitabhängige Transformationsräume werden unter anderem Relativbewegungen zwischen Objekten spezifiziert. Ein Beispiel hierfür ist etwa eine beim Lackieren eines Werkstücks zu verfahrende relative Bewegung zwischen dem zu lackierenden Werkstück und der das Tool eines Manipulators darstellenden Spritzpistole. Ein anderes Beispiel ist ein zeitabhängiger Transformationsraum, der die Beziehung zwischen dem Weltkoordinatensystem und einem von einer Startstellung zu einer Zielstellung zu transportierenden Objekt beschreibt. Dieser zeitabhängige Transformationsraum stellt ebenfalls eine Bewegung dar, diese wird allerdings nur für den Beginn und das Ende der Aufgabe spezifiziert.

Um diese zeitabhängigen Transformationsräume zu spezifizieren, ist zuerst ein Zeitbegriff einzuführen. Die Verwendung der realen Zeit würde bedeuten, daß bei der Spezifikation des Ablaufs einer Aufgabe bereits genau festgelegt wird, welche Stellung zu welchem Zeitpunkt eingenommen wird.

Je nach verwendeter Umgebung kann dies aber für die Manipulatoren eine unmögliche Aufgabe darstellen oder aber bedeuten, daß die Zeit verbraucht werden muß, obwohl die Aufgabe schneller hätte erledigt werden können. Bei vielen Aufgaben ist die Festlegung der realen Zeit aber nicht erwünscht, vielmehr steht die Ausführbarkeit einer Aufgabe im Vordergrund. Zudem wird bei vielen Aufgaben impliziert, daß die Ausführung *möglichst schnell* erfolgen soll, ohne daß ein Roboterprogrammierer genaue Zeitschranken vorgibt. Für manche Aufgaben (etwa Bahnschweißen) sind dennoch Anforderungen an die reale Zeit durch Angabe einer Verfahrgeschwindigkeit oder einer Beschleunigung für eine zeitabhängige Transformation (s. Kap. 5.5.2) erforderlich. Für andere Aufgaben, etwa den Transport eines sich abkühlenden Objekts, kann es etwa erforderlich sein, daß sich das zu transportierende Objekt nach exakt 20 s an der Zielposition befinden muß, also eine Zeit*spanne* in Realzeit spezifiziert wird.

Aus diesen Betrachtungen lassen sich folgende Anforderungen an den zu verwendenden Zeitbegriff aufstellen:

- Zur Spezifikation des Aufgaben*verlaufs* werden keine Realzeitangaben verwendet. Es ist lediglich der zeitliche Ablauf zu beschreiben.
- Die Spezifikation von Zeitdauer, Geschwindigkeit und Beschleunigung für einzelne Transformationen erfolgt in Realzeiteinheiten (s, m/s, m/s^2).
- Die Angabe einer exakten Zeitdauer für eine Aufgabenausführung ist nur in wenigen Spezialfällen (s.o.) erforderlich. Im allgemeinen wird eine *möglichst schnelle* Aufgabenausführung impliziert. *Möglichst schnell* bedeutet eine unter Berücksichtigung aller spezifizierten Geschwindigkeiten und Beschleunigungen und unter Berücksichtigung der Manipulatoreigenschaften minimale Zeitdauer.

Somit ist eine Trennung der Spezifikation des zeitlichen Verlaufs einer Aufgabe von Randbedingungen für die reale Zeit erforderlich. Zur Spezifikation des zeitlichen Ablaufs einer

Aufgabe wird eine *formale Zeit* t eingeführt, deren Semantik folgendermaßen definiert ist:

- $t = 0$ bezeichnet den Aufgabenbeginn
- $t = 1$ bezeichnet das Aufgabenende
- Eine Skalierungsfunktion

$$skal(t) : \begin{cases} [0,1] & \rightarrow & \mathbb{R} \\ t & \rightarrow & skal(t) \end{cases}$$

  bildet die formale Zeit in die Realzeit ab. $skal$ ist eine stetige Funktion und erlaubt somit keine Zeitsprünge. Desweiteren ist $skal$ monoton, wodurch die Aufrechterhaltung des in der Aufgabenspezifikation beschriebenen zeitlichen Ablaufs gewährleistet ist.

Diese Skalierungsfunktion $skal$ wird von einer Komponente des Bewegungsplaners im Anschluß an die Planung einer kollisionsfreien Bahn so bestimmt, daß folgende Bedingungen erfüllt sind:

1. Die Randbedingungen an die Realzeit (Dauer, Geschwindigkeit, Beschleunigung) werden erfüllt.
2. Die manipulatorspezifischen Randbedingungen (maximale Verfahrgeschwindigkeit, maximale Beschleunigung) werden eingehalten.
3. Die Aufgabenausführung erfolgt *möglichst schnell*.

Diese Auftrennung in Bahnplanung und Betrachtung der Dynamik in einem Nachbearbeitungsschritt zur Bestimmung einer Skalierungsfunktion $skal$ findet sich auch bei Koga [KKKL94]. Ein Ansatz zur Bestimmung von Skalierungsfunktionen ist in [MA90] beschrieben.

In Kap. 5.4 werden die zur Spezifikation der formalen Zeit erforderlichen Sprachkonstrukte eingeführt.

4.4 Terminierung von Elementaraufgaben

In jeder Aufgabe existieren eine oder mehrere Beziehungen, die eine Bewegung initiieren. Durch die Einbeziehung von Sensorinformation ergeben sich für die Beendigung dieser Bewegungen (und somit der Aufgabe) über das Abfahren einer geometrisch festgelegten Bahn hinausgehende Möglichkeiten. Diese werden im folgenden betrachtet.

In Kap. 4.1 wurde dargelegt, daß von einem Sensor gelieferte Sensorinformation jeweils einem Bezugssystem zugeordnet ist. Desweiteren wurde dargelegt, daß mit Hilfe von Sensorinformation Beziehungen spezifiziert werden. Für die Spezifikation von Terminierungsbedingungen ergibt sich daraus, daß jeweils einer Beziehung eine oder mehrere Einzelterminierungsbedingungen zugeordnet werden.

Für die Beendigung einer Aufgabe können mehrere Einzelterminierungsbedingungen relevant sein. Diese Einzelterminierungsbedingungen sind jeweils an eine spezifizierte Beziehung gebunden. Da im allgemeinen mehrere Einzelterminierungsbedingungen in mehreren Beziehungen in einer Aufgabe auftreten können, ergeben sich Konstellationen von Einzelterminierungsbedingungen, die die Beendigung einer Aufgabe bedingen. Die Aufgabe gilt dann als beendet, wenn eine Gesamtterminierungsbedingung – bestehend aus einem booleschen Ausdruck über den Einzelterminierungsbedingungen – erfüllt ist. Da verschiedene Konstellationen von Einzelterminierungsbedingungen eine unterschiedliche Beendigung der Aufgabe bedeuten können (etwa ordentliche Beendigung einer Aufgabe oder fehlerhafte Beendigung), existieren zu einer Aufgabe im allgemeinen mehrere Gesamtterminierungsbedingungen.

Beispielsweise ist das Einsetzen eines Bolzens dann beendet, wenn der Bolzen in der Passung nicht mehr weiter eingesetzt werden kann *oder* eine bestimmte Mindesteinsetztiefe erreicht ist. Eine Aufgabe, bei der eine Feder während des Transports gespannt werden soll, ist dann beendet, wenn die erforderliche Spannkraft aufgebracht *und* das Transportziel erreicht ist *oder* die Distanz zwischen den Greifern eine bestimmte Schranke übersteigt (was auf einen Bruch der Feder oder ein Entgleiten aus den Greifern hindeutet).

Da neben Terminierungsbedingungen, die durch Sensorinformation spezifiziert werden, auch das Erreichen der formalen Zeit $\mathbf{T} = \mathbf{1}$ zur Terminierung einer Aufgabe beiträgt, ergeben sich für Beziehungen, die eine Bewegung realisieren, mehrere Kombinationen aus auf Sensorinformation und der formalen Zeit beruhenden Einzelterminierungsbedingungen:

- Nur zeitliche Terminierungsbedingung spezifiziert

  Beziehungen dieser Art spezifizieren eine nicht durch Sensoren beendete Bewegung, ohne die Betrachtung von Sensorinformation erstrecken sich spezifizierte Bewegungen über die gesamte formale Zeit.

- Sensorielle und zeitliche Terminierungsbedingungen spezifiziert

  Beziehungen dieser Art stellen Bewegungen dar, die entweder durch das Erreichen des Endes der Bewegung (formale Zeit $\mathbf{T} = \mathbf{1}$ oder durch das Eintreten einer durch Sensoren bestimmten Terminierungsbedingung beendet werden. Ein Beispiel hierfür ist etwa das oben erwähnte Einsetzen eines Bolzens, bei dem entweder das Erreichen einer Mindesteinsetztiefe (formale Zeit) oder das Erreichen des Passungsbodens (Sensorik) zur Beendigung der Bewegung führt.

- Nur sensorielle Terminierungsbedingungen

Dies sind Bewegungen, die nicht von Aufgabenanfang bis Aufgabenende geometrisch durch eine abzufahrende, vorgegebene Bahn vorgegeben sind. Der a priori unendliche Charakter dieser Bewegung wird dadurch spezifiziert, daß aus einem definierten Startzustand eine Bewegung in eine vorgegebene Richtung mit gegebener Geschwindigkeit erfolgt. Das Ende der Bewegung ist durch das Eintreten einer sensoriell bestimmten Terminierungsbedingung bestimmt, nicht aber durch ein vorab geometrisch bekanntes und modelliertes Ende der Bewegung. Es sind dies also Bewegungen der Art

Fahre mit Geschwindigkeit <G> in Richtung <R> bis das Ereignis <E> eintritt.

Ein Beispiel für eine derartige Bewegung ist eine zu einem Kontakt führende Suchbewegung: Ein Bolzen wird solange auf ein Objekt, in das er eingesetzt werden soll, zu bewegt, bis eine auftretende Kraft auf einen Objektkontakt schließen läßt.

In Kap. 5.5 werden die Sprachkonstrukte vorgestellt, die zur Spezifikation von Terminierungsbedingungen erforderlich sind.

4.5 Terminierungsstatus von Elementaraufgaben

Eine grundlegende Eigenschaft aufgabenorientiert programmierbarer Systeme ist ihre Fähigkeit zur Reaktion auf Ereignisse und Zustände in der Umgebung, in der sie ihre Aufgaben ausführen. Aus diesem Grund ist es erforderlich, Information über die Ausführung einer Elementaraufgabe und den sich daraus ergebenden Zustand der Umgebung an die die Aufgabenausführung veranlassenden Schichten zurückzuliefern.

Wichtig zur Entscheidung, mit welcher Sequenz von Elementaraufgaben zur Erfüllung einer komplexen Aufgabe fortzufahren ist, ist der Terminierungsstatus einer Elementaraufgabe, also die Information, welche Terminierungsbedingung zur Beendigung der Aufgabenausführung führte. Als Beispiel hierfür sei der in Kap. 5.5.1 beschriebene Suchvorgang nach der Passung, in die ein Bolzen einzusetzen ist, angeführt: Wurde die Aufgabe durch eine Terminierungsbedingung beendet, die wegen einer erhöhten Kraft entlang der Suchrichtung auf das Finden der gesuchten Passung hindeutet, so kann mit dem Einsetzen des Bolzens durch eine entsprechende Elementaraufgabe begonnen werden. Wurde hingegen die komplette Suchtrajektorie abgefahren, ohne daß die Passung gefunden wurde, so sind etwa Elementaraufgaben erforderlich, die mittels Bildverarbeitung die Passung erneut lokalisieren, bevor wiederum mit der kraft-momentengestützten Suche begonnen wird.

Zur Beschreibung des Terminierungszustands einer Elementaraufgabe wird der Bezeichner der Gesamtterminierungsbedingung zurückgeliefert, die zum Beenden der Aufgabe führte, für den Fall, daß das Erreichen des formalen Zeitpunkts $\mathbf{T = 1}$ zur Beendigung der Aufgabe führte, wird der Bezeichner `OK` zurückgeliefert.

Neben dem Terminierungsstatus der Aufgabe ist der Zustand der Umwelt nach der Aufgabenbeendigung erforderlich, um weitere Elementaraufgaben zu planen. Dieser wird zum einen durch die aktuellen Transformationen, die die einzelnen Beziehungen bei Aufgabenbeendigung einnehmen, beschrieben. Aus diesen ergibt sich der Startzustand der Beziehungen für die nachfolgende Elementaraufgabe: Wurde etwa die Stellung eines zu bearbeitenden Objekts durch eine Beziehung *Welt – Objekt* spezifiziert, die nur die Startstellung des Objekts beschreibt, so stellt gerade die Stellung des Objekts am Aufgabenende den Startzustand für die nachfolgende Aufgabe dar. Sie wird durch die Transformation beschrieben, die für die Beziehung *Welt – Objekt* am Aufgabenende herrscht.

Desweiteren sind die Belegungen am Aufgabenende der in der Aufgabe berücksichtigten Sensorwerte zurückzuliefern. Sie dienen dazu, den bislang nur symbolisch beschriebenen Aufgabenstatus am Aufgabenende näher zu qualifizieren.

Sei **TB** die Menge der Bezeichner der Gesamtterminierungsbedingungen, **T** die Menge der Transformationen, **n** die Anzahl der in einer Aufgabe spezifizierten Beziehungen und **SI** $\subset \mathbf{R}^m$ die in einer Aufgabe berücksichtigte Sensorinformation, so ergibt sich das zurückzuliefernde Ergebnis **RES** einer Elementaraufgabe formal zu

$$\mathbf{RES} = (tb, t, si), \quad tb \in \mathbf{TB}, t \in \mathbf{T}^n, si \in \mathbf{SI}$$

5 Eine Spezifikationssprache für Aufgaben für kooperierende Manipulatoren

Aus der im vorhergehenden Kapitel beschriebenen formalen Darstellung der betrachteten Aufgaben für kooperierende Manipulatoren wird nun im folgenden eine formale Sprache abgeleitet, die die Spezifikation dieser Aufgaben leistet.

5.1 Spezifikation der Aufgabenstruktur

Im folgenden wird ein Formalismus zur Aufgabenspezifikation beschrieben und die erforderlichen Konstrukte vorgestellt. Die Sprachbeschreibung findet sich vollständig im Anhang A.

Wie in den einleitenden Ausführungen bereits erwähnt, sind zur Spezifikation einer Aufgabe eine Menge von Objekten und Beziehungen zwischen diesen zu beschreiben (Anforderung **AA5**, *Spezifikation der Objekte*).

Bei der Spezifikation von Aufgaben wird von der Existenz eines Umweltmodells in Form einer geometrischen Datenbasis ausgegangen, in der die beteiligten Objekte modelliert sind. Eine Erstellung und Verwaltung einer derartigen Datenbasis ist etwa mit einem CAD-System [Tec92, WBK94, Den92] möglich. Um die in einer geometrischen Datenbasis modellierten Objekte im hier vorgestellten Formalismus referenzieren zu können, sind an die Schnittstelle zur geometrischen Datenbasis folgende Anforderungen gestellt:

- Jedem in der geometrischen Datenbasis modellierten Objekt ist ein eindeutiger Bezeichner zugeordnet.
- Jedem Objekt ist ein Ursprungskoordinatensystem zugeordnet, über dessen Lage die Lage des Objekts festgelegt ist.
- Es können zusätzliche Koordinatensysteme definiert werden, die jeweils über einen eindeutigen Bezeichner referenziert werden.
- Mehrere Objekte mit ihrer jeweiligen Lage können zu einer *Umgebung* zusammengefaßt werden, die ebenfalls über einen eindeutigen Bezeichner referenziert wird.

Die bei der Ausführung einer Aufgabe beteiligten Objekte können in drei Klassen eingeteilt werden:

1. Objekte, die die Umgebung bilden, in der eine Aufgabe ausgeführt wird.

   Die Objekte dieser Klasse bilden lediglich den Rahmen, in dem eine Aufgabe ausgeführt wird, und werden in der Aufgabenspezifikation ansonsten nicht referenziert. Sie werden daher zu einer *Umgebung* zusammengefaßt und in der Aufgabenspezifikation durch einen Bezeichner referenziert (etwa die Referenz auf eine als Einheit in

einer geometrischen Datenbasis verwaltete Menge von Objekten mit ihrer jeweiligen Lage relativ zum Weltkoordinatensystem).

In der hier vorgestellten Sprache werden diese Objekte durch das Konstrukt

ENVIRONMENT: <*Bezeichner*>

referenziert.

2. Objekte, die während einer Aufgabenausführung manipuliert werden.

   Die Objekte dieser Klasse werden in der Aufgabenbeschreibung durch ihr Ursprungskoordinatensystem repräsentiert, das durch einen eindeutigen Bezeichner beschrieben ist. Dieser Bezeichner referenziert das zum gleichnamigen Objekt gehörende Ursprungskoordinatensystem in der geometrischen Datenbasis.

3. Manipulatoren, die die Aufgabe ausführen.

   Die Objekte dieser Klasse, die Manipulatoren, werden zusammen mit der ersten Klasse in der *Umgebung* einer Aufgabe in der geometrischen Datenbasis referenziert. Dort werden alle relevanten Eigenschaften der Manipulatoren (geometrische und kinematische Beschreibung, Lage relativ zum Weltkoordinatensystem, ggf. zusätzliche physikalische Eigenschaften wie dynamische Beschreibung, Masse, Steifigkeit, Oberflächeneigenschaften usw.) repräsentiert und können durch einen Bezeichner referenziert werden. Für die Aufgabenausführung sind, wie in Kap. 3.6 dargelegt, lediglich die Effektorkoordinatensysteme der aufgabenausführenden Manipulatoren relevant. Diese werden im Formalismus durch den Bezeichner der Repräsentation des zugehörigen Manipulators in der geometrischen Datenbasis referenziert.

Die für eine Aufgabenspezifikation relevanten Objekte werden also gemeinsam mit ihrem jeweiligen Ursprungskoordinatensystem durch Bezeichner repräsentiert. Darüberhinaus können zusätzliche Hilfskoordinatensysteme definiert sein, denen eine spezielle Bedeutung zukommt (etwa dem *Compliance-Frame*, s. Kap. 3.3, oder einem von einer Kamera zu beobachtenden Punkt auf einem Objekt), oder eine Beschreibung der Aufgabe erleichtern. So ist beispielsweise, wie bereits in den Ausführungen zum Framekonzept (Kap. 2.2.1) erläutert, eine Beschreibung einer Aufgabe relativ zu einem Koordinatensystem, das die Platte eines Tisches repräsentiert, einfacher und anschaulicher als relativ zum Ursprungskoordinatensystem des Tisches, das möglicherweise in einem Tischbein liegt. Ein weiteres, bei allen Aufgabenspezifikationen erforderliches Koordinatensystem, das ebenfalls in diese Klasse der kein Objekt repräsentierenden Koordinatensysteme fällt, ist das Weltkoordinatensystem, das mit dem Bezeichner `Welt` referenziert wird.

Im Gegensatz zum Framekonzept, bei dem zur Spezifikation der Lage aller auftretenden Koordinatensysteme die Angabe des jeweiligen Bezugskoordinatensystems ausreicht, können hier Beziehungen zwischen beliebigen Koordinatensystemen bestehen. Aus diesem Grund wird die Struktur der Aufgabe - bestehend aus den beteiligten Koordinatensystemen und den zwischen diesen bestehenden Beziehungen - gemäß Anforderung **AA3** (*Übersichtlichkeit*) durch die graphische Darstellung eines Graphen mit gerichteten Kanten spezifiziert.

Dabei ist $\mathbf{F}$, die Menge der eindeutig bezeichneten Koordinatensysteme, die Knotenmenge des Graphen. Die Menge $\mathbf{B}$ der Kanten im Graph repräsentiert die in Kap. 4.2 beschriebenen, eindeutig bezeichneten Beziehungen zwischen den Koordinatensystemen. Nach [Vol91] wird dann der die Aufgabenstruktur darstellende Graph $\mathbf{G}$ durch das Tripel

$$\mathbf{G} = (\mathbf{F}, \mathbf{B}, h)$$

beschrieben, wobei die Abbildung $h : \mathbf{B} \rightarrow \mathbf{F} \times \mathbf{F}$ jeweils einem Start- und einem Zielknoten eine gerichtete Kante zuordnet. Die graphische Darstellung des Graphen $\mathbf{G}$ bildet dabei einen direkten Bestandteil des Spezifikationsformalismus.

Die Knoten des Graphen (Abb. 20) stellen die bezeichneten Koordinatensysteme dar. Dabei werden aus Gründen der besseren Lesbarkeit die verschiedenen Klassen von Knoten im Graphen unterschiedlich dargestellt. Die die Effektorkoordinatensysteme der Manipulatoren darstellenden Knoten werden mit abgerundeten Ecken und dicken Rändern, die Objekte repräsentierenden Knoten mit abgerundeten Ecken und dünnen Rändern und die Hilfskoordinatensysteme darstellenden Knoten rechteckig gezeichnet.

Die Kanten zwischen den Knoten repräsentieren die zwischen den Koordinatensystemen bestehenden Beziehungen. Da eine direkte Beschreibung einer Beziehung im Graphen die Übersichtlichkeit der Darstellung stark einschränken würde, werden die Kanten im Graphen lediglich mit einem (bedeutungstragenden) Bezeichner versehen. Über diesen wird die Beziehung in einem nachfolgenden Abschnitt der Spezifikation genauer beschrieben.

Die Richtung der Kanten bezeichnet das Ursprungskoordinatensystem, in dem die Beziehung beschrieben wird: die Spitze des Pfeils zeigt auf den Knoten, der das Ursprungskoordinatensystem darstellt.

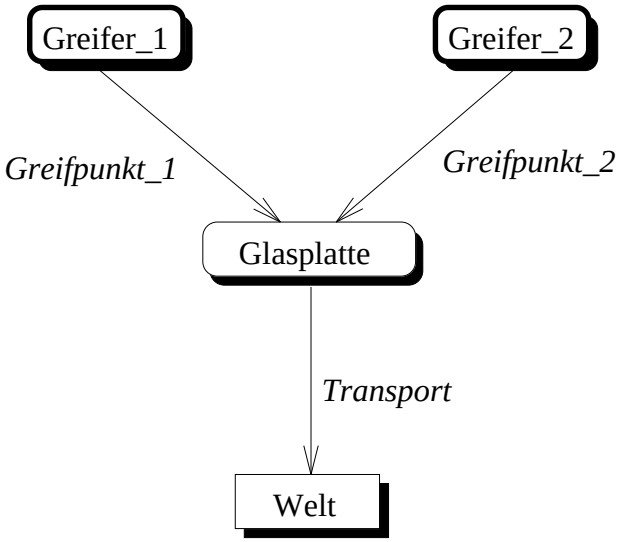


Abbildung 20: Transport einer Glasplatte

Als Beispiel wird in (Abb. 20) die Struktur einer Aufgabe dargestellt, bei der zwei Mani-

pulatoren eine Glasplatte transportieren. Die Effektorkoordinatensysteme werden als dick gezeichnete Knoten mit abgerundeten Ecken dargestellt. Als Objekt ist hierbei die zu transportierende Glasplatte durch das sie repräsentierende Ursprungskoordinatensystem spezifiziert. Als ein Objekt darstellender Knoten wird der zugehörige Knoten mit abgerundeten Ecken dargestellt. Desweiteren ist das Weltkoordinatensystem angegeben. Da diesem Koordinatensystem kein darstellbares Objekt zugeordnet ist, wird es in der graphischen Darstellung als Rechteck dargestellt. Als Beziehungen zwischen diesen Koordinatensystemen sind die Greifpunkte (Beziehung zwischen je einem Tool und der Glasplatte) angegeben. Die Pfeilrichtung von den Tools zur Glasplatte bedeutet, daß die Greifpunkte relativ zum Ursprungskoordinatensystem der Glasplatte angegeben werden. Desweiteren wird der Transportvorgang als Beziehung zwischen der Glasplatte und dem Weltkoordinatensystem beschrieben. Die Pfeilrichtung beschreibt, daß die Beziehung zwischen dem Ursprungskoordinatensystem der Glasplatte und dem Weltkoordinatensystem (die Start- und Zielstellung der Glasplatte) mit dem Weltkoordinatensystem als Bezugskoordinatensystem spezifiziert werden.

Aus der im vorangegangenen Kapitel beschriebenen formalen Darstellung von Aufgaben und der in diesem Kapitel vorgestellten Repräsentation der Struktur der Aufgabe läßt sich folgender Rahmen für eine Spezifikationssprache ableiten:

Eine Aufgabenbeschreibung besteht aus dem Namen der Aufgabe (**TASK**), der Darstellung der zu beschreibenden Objekte (**ENVIRONMENT**), der Beschreibung der Aufgabenstruktur (**GRAPH**), der Beschreibung der zwischen diesen Objekten herrschenden Beziehungen (**RELATIONS**), dem Beziehungen zwischen mehreren Koordinatensystemen beschreibenden globalen Constraint (**GLOBAL_CONSTRAINT:**) und der Beschreibung der Terminierungsbedingungen (**TERMINATION_CONDITIONS**):

AUFGABE ::= **TASK:** NAME
ENVIRONMENT: NAME
GRAPH: ⟨GRAPH⟩
RELATIONS: {BEZIEHUNG}$^+$
[**GLOBAL_CONSTRAINT:** G_CONSTRAINT]
[**TERMINATION_CONDITIONS:**]
{TERMINIER_GES_NAME : TERMINIER_GESAMT}$^+$

⟨GRAPH⟩ steht hierbei für die *graphische Darstellung* des die Aufgabenstruktur repräsentierenden Graphen.

5.2 Spezifikation der Beziehungen durch geometrische Constraints

In diesem Kapitel wird die Spezifikation der Beziehungen zwischen den an einer Aufgabenbeschreibung beteiligten Koordinatensystemen durch *geometrische* Constraints dargestellt. Das Hauptaugenmerk liegt dabei auf der Darstellung der Abbildung von zeitabhängigen Transformationsräumen in die reellen Zahlen durch Operationen.

In der Spezifikationssprache werden Beziehungen über den im Graphen verwendeten Namen referenziert und durch eine nachfolgende Spezifikation, bestehend aus einer Constraintmenge und Terminierungsbedingungen, die in Kap. 5.5.1 beschrieben werden, näher spezifiziert.

BEZIEHUNG ::= BEZIEHUNG_NAME :
[CONSTRAINT]
[**TERMINATION** : {TERMINIER_BED}$^+$]

In der formalen Darstellung der Aufgabenbeschreibung in Kap. 4.2 wurden zur Realisierung von Constraints über zeitabhängigen Transformationsräumen $\hat{T}$ Abbildungen der Form

$$f : \begin{cases} \mathbf{T} \times \mathbf{SI} \times [0,1] & \rightarrow & \mathbb{R} \\ (T, si, t) & \rightarrow & f(T, si, t) \end{cases}$$

eingeführt. Diese Abbildungen von zeitabhängigen Transformationen und Sensorinformation in die reellen Zahlen sind formale Darstellungen. Für eine Verwendung in einem Spezifikationsformalismus sind sie durch konkrete Operationen zu realisieren.

Häufig treten Constraints auf, die lediglich zeitabhängige Transformationsräume beschreiben. Daher werden zunächst Abbildungen ohne Einbeziehung der Sensorinformation der Form

$$f : \begin{cases} \mathbf{T} \times [0,1] & \rightarrow & \mathbb{R} \\ (T, t) & \rightarrow & f(T, t) \end{cases}$$

betrachtet. Auf die Spezifikation der Reaktion auf Sensorinformation wird anschließend in Kap. 5.5 eingegangen.

Zur Darstellung dieser Abbildungen durch Operationen werden zeitabhängige Transformationen durch Selektorfunktionen in einen Raum abgebildet, in dem ein Kalkül existiert. Als derartiger Raum bietet sich zum einen der Körper der reellen Zahlen an. Die Formulierung von Constraints stellt sich dann als Kette von Abbildungen dar. Transformationen (in ihrer Darstellung als homogene Matrix) werden durch Selektion einzelner Matrixelemente in den Körper der reellen Zahlen abgebildet. Diese Ausdrücke werden durch Gleichungen

und Ungleichungen auf boolesche Werte abgebildet, auf denen dann letztendlich boolesche Ausdrücke formuliert werden können:

$$\hat{\mathbf{T}} \overset{Sel.-Funkt.}{\rightarrow} \mathbb{R} \overset{(Un-)Gl.}{\rightarrow} \{\mathbf{T}, \mathbf{F}\}, \quad \Phi \overset{bool.Ausdr.}{\rightarrow} \{\mathbf{T}, \mathbf{F}\}$$

Φ bezeichnet dabei die Menge der Belegungen, also der Abbildungen von Aussagenvariablen (hier: Gleichungen und Ungleichungen) in booleschen Ausdrücken auf Wahrheitswerte.

Desweiteren bietet sich – da geometrische Zusammenhänge anschaulich beschrieben werden sollen – der Vektorraum der dreidimensionalen Vektoren mit anschließender Abbildung in die reellen Zahlen durch das Skalarprodukt an. In diesem Fall können Constraints durch eine Kette von Abbildungen dargestellt werden, indem zeitabhängige Transformationen durch Selektion von Spaltenvektoren in den Vektorraum der dreidimensionalen Vektoren abgebildet werden. Wie oben beschrieben, können auf diesen Ausdrücken boolesche Ausdrücke formuliert werden:

$$\hat{\mathbf{T}} \overset{Sel.-Funkt.}{\rightarrow} \mathbb{R}^3 \overset{Skalarprod.}{\rightarrow} \mathbb{R} \overset{(Un-)Gl.}{\rightarrow} \{\mathbf{T}, \mathbf{F}\} \overset{bool.Ausdr.}{\rightarrow} \{\mathbf{T}, \mathbf{F}\}$$

Für das globale Constraint G_CONSTRAINT, durch das die Beziehungen mehrerer Koordinatensysteme zueinander beschrieben wird, werden dieselben Sprachkonstrukte verwendet, allerdings ist die explizite Referenzierung von Koordinatensystemen bei Selektionen obligatorisch. Die Produktionen für das globale Constraint werden daher lediglich im Anhang aufgeführt.

5.2.1 Reellwertige Ausdrücke

Transformationen können als Sechstupel aus einem Translationsvektor und einer Rotationsdarstellung, etwa der Yaw-Pitch-Roll-Darstellung [Pau81], dargestellt werden. Analog kann ein zeitabhängiger Transformationsraum $\hat{T}$ als Sechstupel mit Abbildungen der formalen Zeit als Elemente dargestellt werden:

$$\hat{T} = (\texttt{TX(t)}, \texttt{TY(t)}, \texttt{TZ(t)}, \texttt{RX(t)}, \texttt{RY(t)}, \texttt{RZ(t)})$$

Die $\texttt{TX}, \cdots, \texttt{RZ}$ werden nun als Selektorfunktionen sel zur Abbildung von zeitabhängigen Transformationsräumen in die reellen Zahlen verwendet:

$$sel : \begin{cases} \hat{\mathbf{T}} & \rightarrow & \mathbf{R} \\ \hat{T} & \rightarrow & sel(\hat{T}) \end{cases}$$

Die Selektorfunktionen werden dann zur Formulierung der den zeitabhängigen Transformationsraum beschreibenden Constraints verwendet. Über diesen Selektorfunktionen können reellwertige Ausdrücke mit den zugrundeliegenden Operationen (Addition, Subtraktion,

Multiplikation, Division) und ein- und mehrstelligen Funktionen auf den reellen Zahlen formuliert werden.

Weiterhin können reellwertige Variablen – dargestellt durch beliebige Bezeichner – zur Beschreibung eines beliebigen, aber festen Werts innerhalb eines Constraints verwendet werden.

Als erstes Beispiel für die Spezifikation von Beziehungen mit oben vorgestellten Selektorfunktionen soll die Beziehung zwischen zwei Koordinatensystemen, deren Z-Achsen antiparallel sind, der Abstand der Z-Achsen weniger als 20 cm betragen soll und die sich zum Aufgabenende hin bis zur Übereinstimmung annähern sollen:

```
sqrt (TX(t) · TX(t) + TY(t) · TY(t)) < 20 · (1 - t) AND
RX(t) = 180 AND
RY(t) = 0 AND
RZ(t) = 0
```

Ein zweites Beispiel ist das Verfahren einer sinusförmigen Trajektorie, etwa zum Lackieren einer Oberfläche:

```
TX(t) = breite · t
TY(t) = hoehe · sin (anzahl_schwingungen · 2 · π · t) AND
TZ(t) = abstand AND
RX(t) = 180 AND
RY(t) = 0 AND
RZ(t) = 0
```

5.2.2 Ausdrücke der Vektoralgebra

Bei vielen Beziehungen sind geometrische Zusammenhänge, wie etwa "das Objekt darf beim Transport nicht gekippt werden", "die Kamera soll zwischen 10 cm und 30 cm vom Beobachtungspunkt entfernt sein" oder "der Block soll auf der Tischoberfläche entlanggeschoben werden", zu beschreiben. Dies ist durch die in Kap. 5.2.1 beschriebene, ausdrucksmächtige Methode der reellwertigen Selektorfunktionen durchaus möglich, führt aber im allgemeinen zu komplizierten, schwer nachvollziehbaren Ausdrücken.

Eine zweite Methode zur Realisierung der Abbildung von zeitabhängigen Transformationsräumen in die reellen Zahlen durch Operationen ist die Selektion von 3D-Vektoren, die die betrachteten Koordinatensysteme aufspannen. Durch Vektorraumoperationen und anschließende Abbildung in die reellen Zahlen durch Skalarproduktbildung wird eine für geometrische Sachverhalte naheliegendere Beschreibungsmöglichkeit zur Verfügung gestellt.

Zur Selektion der gewünschten Vektoren werden die vier Selektoren

```
ORIGIN, X_VECTOR, Y_VECTOR, Z_VECTOR
```

verwendet. Der Selektor `ORIGIN` liefert den Ursprungsvektor des jeweiligen Koordinatensystems. Die Selektoren `X_VECTOR, Y_VECTOR, Z_VECTOR` liefern den die jeweilige Achse des betrachteten Koordinatensystems repräsentierenden Einheitsvektor. Um die Koordinatensysteme zur Selektion zu referenzieren, werden die Namen der Koordinatensyteme verwendet, an die – getrennt durch einen Punkt – der jeweilige Selektor angehängt wird.

Auf den selektierten Vektoren und konstanten Vektoren können dann die vorhandenen Vektorraumoperationen (Vektoraddition, -subtraktion, skalares Produkt, Kreuzprodukt) durchgeführt werden. Durch Skalarproduktbildung werden diese vektorwertigen Ausdrücke in die reellen Zahlen abgebildet. Die formale Zeit wird bei dieser Methode erst *nach* der Abbildung durch das Skalarprodukt in die reellen Zahlen in Betracht gezogen: Anstatt Vektoren als Funktionen der formalen Zeit zu beschreiben, ist es naheliegender, die Beziehungen zwischen diesen Vektoren (die durch die Skalarproduktbildung quantifiziert werden) als abhängig von der formalen Zeit zu beschreiben.

Sei A das Bezugskoordinatensystem, in dem die Beziehung zwischen A und B dargestellt wird. Um diese Beziehung nun durch die Vektoren zu beschreiben, die die Koordinatensysteme aufspannen, können die interessierenden geometrischen Zusammenhänge auf zwei Arten dargestellt werden:

1. Es wird direkt die Transformation $\mathbf{R}$ zwischen den beiden Koordinatensystemen betrachtet. Dazu werden die $\mathbf{R}$ aufspannenden Vektoren im Bezugskoordinatensystem dargestellt. Repräsentiert man die Transformation $\mathbf{R}$ durch eine homogene Transformationsmatrix

$$\mathbf{R} = \begin{pmatrix} \mathbf{x} & \mathbf{y} & \mathbf{z} & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

   so liefern die Selektoren:

$$\mathbf{R}.\texttt{ORIGIN} = \mathbf{t}$$

$$\mathbf{R}.\texttt{X\_VECTOR} = \mathbf{x}$$

$$\mathbf{R}.\texttt{Y\_VECTOR} = \mathbf{y}$$

$$\mathbf{R}.\texttt{Z\_VECTOR} = \mathbf{z}$$

   Als Konstrukt in der Spezifikationssprache für diese Art der Selektion von Vektoren wird der Name der die beiden Koordinatensysteme in Relation setzenden Beziehung mit – durch einen Punkt getrennt – angehängtem Selektor verwendet.

   BEZIEHUNG_NAME . VEKTOR_SELEKTOR

   VEKTOR_SELEKTOR ::= **ORIGIN** | **X_VECTOR**
   | **Y_VECTOR** | **Z_VECTOR**

2. Jedes der beiden in Beziehung gesetzten Koordinatensysteme wird im Weltkoordinatensystem dargestellt. Die Selektion der Vektoren erfolgt analog zu 1) durch Angabe des Bezeichners des jeweiligen Koordinatensystems. Dazu dient das Konstrukt:

   KOORDSYS_NAME . VEKTOR_SELEKTOR

Wird in Constraints innerhalb von Beziehungen das zugehörende Referenzkoordinatensystem referenziert, ist die Bedeutung des Selektors eindeutig. Da dieser Fall sehr häufig auftritt, kann die Bezeichnung des Koordinatensystems hierbei entfallen.

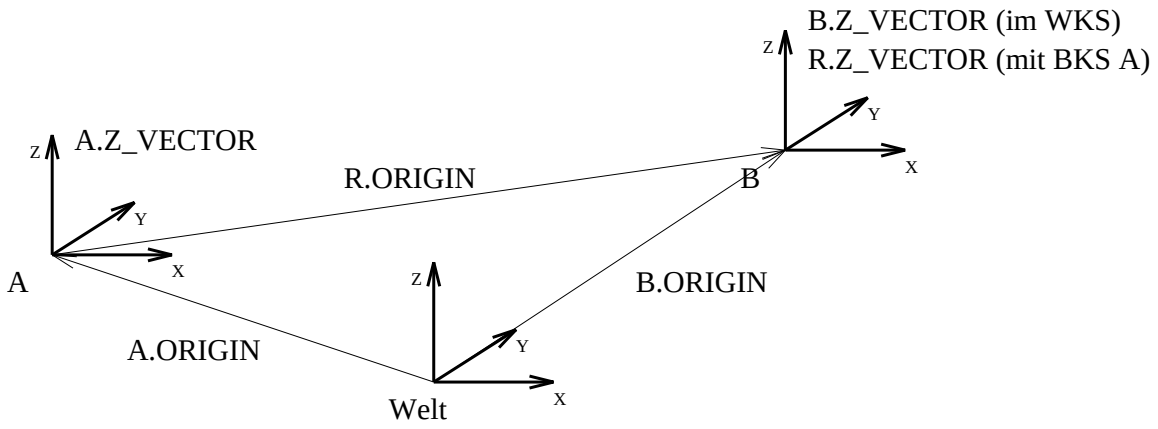


Abbildung 21: Selektion von Koordinatensysteme aufspannenden Vektoren

In Abb. 21 sind für die beiden Koordinatensysteme **A** und **B**, zwischen denen die Beziehung **R** existiert (Bezugskoordinatensystem **A**) die Selektion des Ursprungs und des die Z-Achse darstellenden Einheitsvektors der beiden Koordinatensysteme nach den beiden Möglichkeiten 1) und 2) skizziert.

Als Beispiel soll beschrieben werden, daß die Z-Achsen der Koordinatensysteme **A** und **B** antiparallel liegen sollen:

```
A.Z_VECTOR · B.Z_VECTOR = -1
```

Der Sachverhalt, daß die Entfernung der Ursprünge der durch die Beziehung **R** beschriebenen Koordinatensysteme mindestens 20 betragen soll, läßt sich beschreiben durch:

```
sqrt ( R.ORIGIN · R.ORIGIN ) < 20
```

Durch das Skalarprodukt können als Maße zum einen die Länge von Vektoren und zum anderen der Winkel zwischen Vektoren formuliert werden. Allerdings ist diese Formulierung bei komplexeren Ausdrücken nicht leicht lesbar. Daher liegt es nahe, für häufig verwendete Maße eine abkürzende Schreibweise einzuführen.

Komponenten eines Koordinatensystems Betrachtet man die Darstellung eines Koordinatensystems, so findet man folgende Komponenten:

- Ebenen

  Es werden durch jeweils zwei Basisvektoren eines Koordinatensystems drei Ebenen festgelegt. Diese werden im folgenden durch die Selektoren `XY_PLANE`, `XZ_PLANE`, `YZ_PLANE` referenziert.

- Achsen

  Jeder der das Koordinatensystem aufspannenden Basisvektoren gibt die Richtung einer Raumachse an. Im folgenden werden diese durch die Selektoren `X_AXIS`, `Y_AXIS`, `Z_AXIS` bezeichnet. Anders als oben werden hierbei nicht die Einheitsvektoren, sondern eine geometrische Darstellung der die Achsen darstellenden Geraden referenziert.

- Ursprung

  Ein weiterer Bestandteil eines Koordinatensystems ist sein Ursprungsvektor, der den Ursprung relativ zu einem Bezugskoordinatensystem (hier das Weltkoordinatensystem) beschreibt. Wie bereits oben dargelegt, wird der Ursprung durch den Selektor `ORIGIN` referenziert.

Wie im Anhang B gezeigt wird, lassen sich all diese Komponenten eines Koordinatensystems durch reellwertige Operationen und Vektoroperationen über den ein Koordinatensystem erzeugenden Vektoren darstellen.

Maße Um Beziehungen zwischen diesen Komponenten eines Koordinatensystems auszudrücken, werden nun Kurzschreibweisen für ein translatorisches und ein rotatorisches Maß zwischen obigen Komponenten *Ebene, Achse* und *Ursprung* eines Koordinatensystems eingeführt.

Das translatorische Maß, der Abstand (Schlüsselwort: `DIST`), zwischen den einzelnen Komponenten eines Koordinatensystems ist folgendermaßen definiert:

- Abstand Ebene – Ebene

  Minimaler Abstand zweier Punkte aus je einer Ebene, 0 für nicht parallele Ebenen, bei parallelen Ebenen die Länge eines Vektors, der in einem Punkt der einen Ebene beginnt und in einem Punkt der zweiten Ebene endet und auf beiden Ebenen senkrecht steht.

- Abstand Ebene – Achse

  minimaler Abstand eines Punktes der Achse und eines Punktes der Ebene. Der Abstand ist 0, wenn die Achse nicht parallel zur Ebene liegt, ansonsten die Länge eines Vektors, der in einem Punkt der Ebene beginnt, in einem Punkt der Achse endet und auf beiden senkrecht steht.

- Abstand Ebene – Ursprung

  minimaler Abstand des Ursprungs zur Ebene, Länge eines Vektors, der in der Ebene beginnt, im Ursprung endet und auf der Ebene senkrecht steht.

- Abstand Achse – Achse

  minimaler Abstand zweier Punkte aus je einer Achse. Die Länge eines Vektors, der in einem Punkt der einen Achse beginnt und in einem Punkt der anderen Achse endet und der auf beiden Achsen senkrecht steht.

- Abstand Achse – Ursprung

  minimaler Abstand des Ursprungs zur Achse, Länge eines Vektors, der in der Achse beginnt, im Ursprung endet und auf der Achse senkrecht steht.

- Abstand Ursprung – Ursprung

  Die Länge des die beiden Ursprünge verbindenden Vektors.

Ebenso kann als rotatorisches Maß der Winkel (Schlüsselwort: `ANGLE`) zwischen obigen Komponenten betrachtet werden:

- Winkel Ebene – Ebene

  kleinster Winkel zwischen den Normalenvektoren der beiden Ebenen.

- Winkel Ebene – Achse

  $(90^0 - \alpha)$, α ist der Winkel zwischen dem Normalenvektor der Ebene und dem Richtungsvektor der Achse.

- Winkel Ebene – Ursprung

  Da mit einem einzelnen Punkt keine Richtung assoziiert ist, ist dieser Winkel ohne Sinn.

- Winkel Achse – Achse

  kleinster Winkel zwischen den Richtungsvektoren der beiden Achsen.

- Winkel Achse – Ursprung

  ohne Sinn (s.o.)

- Winkel Ursprung – Ursprung

  ohne Sinn (s.o)

Beispiel:

Eine Kamera ist auf einen zu beobachtenden Punkt gerichtet. Das die Kameraposition repräsentierende Koordinatensystem soll dabei zwischen 20 cm und 30 cm vom zu beobachtenden Punkt entfernt sein. Die Neigung zwischen der optischen Achse der Kamera und der zu beobachtenden Oberfläche soll kleiner als 10^0 sein. Die X-Achsen der beiden Koordinatensysteme sollen dabei parallel bleiben.

```
Kameraposition:
   DIST (Kamera.ORIGIN, Blickpunkt.ORIGIN) > 20 cm AND
   DIST (Kamera.ORIGIN, Blickpunkt.ORIGIN) < 30 cm AND
   Kamera.Z_KOORD > 0 AND
   DIST (Blickpunkt.ORIGIN, Kamera.Z_AXIS) = 0 cm AND
   ANGLE (Blickpunkt.Z_AXIS, Kamera.Z_AXIS) > 170^0 AND
   ANGLE (Blickpunkt.X_AXIS, Kamera.XZ_PLANE) = 0^0
```

5.3 Vereinfachende Schreibweisen

Bei der Spezifikation von Aufgaben gibt es mehrere Klassen von zeitabhängigen Transformationsräumen, die häufig auftreten. Die vorgestellte Beschreibung von zeitabhängigen Transformationsräumen durch Constraints ist zwar sehr allgemein, kann aber – bedingt durch die hohe Mächtigkeit – zu einer umständlichen Beschreibung einfacher Sachverhalte führen. Daher werden für verschiedene, häufig auftretende Klassen von zeitabhängigen Transformationsräumen abkürzende Schreibweisen eingeführt, um gemäß Anforderung **AA3** (*Übersichtlichkeit*) eine einfache, übersichtliche Spezifikation zu gewährleisten.

5.3.1 Spezifikation von konstanten Transformationen

Eine häufig auftretende Klasse von zeitabhängigen Transformationsräumen sind konstante Transformationen.

Dies ist der einfachste Fall einer Transformation. Anstatt jeden einzelnen Freiheitsgrad für die gesamte Zeit `t` durch ein eigenes Constraint mit einem konstanten Wert zu belegen, wird die Darstellung der gewünschten Transformation als Sechstupel aus dem Translationsvektor und der Orientierungsdarstellung direkt angegeben.

Beispiel:

```
GREIFPUNKT_1 : (100,100,0,0,90,0)
```

5.3.2 Spezifikation von Start- und Zielzustand

Die Beschreibung des Start- bzw. Zielzustands einer Beziehung tritt ebenfalls sehr häufig auf. Anstatt nun jeden einzelnen Freiheitsgrad für die Zeit `t = 0` (Startzustand) bzw. `t = 1` (Zielzustand) durch ein eigenes Constraint mit einem konstanten Wert zu belegen, wird die Darstellung der gewünschten Transformation als konstante Transformation mit vorangestelltem Schlüsselwort `START:` bzw. `ZIEL:` angegeben.

Beispielsweise werden Start- und Zielstellung einer Beziehung, die den Transport eines Objekts beschreibt, folgendermaßen dargestellt:

```
Transport:
   START : (1200, 400, 850, 0, 0, 0) AND
   ZIEL  : (100, 0, 0, 0, 0, 0)
```

5.3.3 Spezifikation von Transformationen als Interpolation gegebener Stützstellen

Eine weitere Klasse häufig auftretender zeitabhängiger Transformationsräume sind fest vorgegebene Relativbewegungen zwischen zwei Koordinatensystemen. Bewegungen von einzeln eingesetzten Manipulatoren können als zeitabhängige Transformationen, nämlich zwischen dem Effektorkoordinatensystem und dem Weltkoordinatensystem, betrachtet werden. Eine einfache, anschauliche und weit verbreitete Methode zur Bewegung entlang einer vorgegebenen Bahn ist die Angabe einer Sequenz von nacheinander linear interpoliert anzufahrenden Transformationen, die als kontinuierliche Bewegung mit Überschleifen an den Stützstellen verfahren wird. Diese Möglichkeit zur Spezifikation von zeitabhängigen Transformationsräumen als Interpolation über eine Sequenz gegebener Transformationen wird im Spezifikationsformalismus übernommen. Neben der Linearinterpolation werden als weitere, die Darstellung kontinuierlicher Bahnen ermöglichende Interpolationsarten quadratische, kubische und Splineinterpolation zugelassen [BS87].

Die Spezifikation erfolgt durch ein die Interpolationsart angebendes Schlüsselwort, gefolgt von einer Sequenz zu interpolierender Transformationen, die von dem Schlüsselwort `END_INTERPOLATED` abgeschlossen wird :

INTERPOL_SPEZ ::= INTERPOL_TYP :
STELLUNG{, STELLUNG}$^+$
END_INTERPOLATED

INTERPOL_TYP ::= **LINEAR_INTERPOLATED**

| **QUADRATIC_INTERPOLATED**
| **CUBIC_INTERPOLATED**
| **SPLINE_INTERPOLATED**

Beispiel: Abfahren der Kante eines rechtwinkligen Objekts

```
Kante:
   LINEAR_INTERPOLATED
      (0, 0, 0, 0, 0, 0)
      (100, 0, 0, 0, 0, 0)
      (100, 0, 0, 0, 0, 90)
      (100, 100, 0, 0, 0, 90)
      (100, 100, 0, 0, 0, 180)
      (0, 100, 0, 0, 0, 180)
      (0, 100, 0, 0, 0, 270)
      (0, 0, 0, 0, 0, 270)
   END_LINEAR_INTERPOLATED
```

5.4 Die formale Zeit

Aus den Betrachtungen in Kap. 4.3 werden nun hier die entsprechenden Sprachkonstrukte zur Beschreibung der formalen Zeit abgeleitet. In der hier vorgestellten Sprache wird die Zeitvariable `t` eingeführt, mit deren Hilfe der *zeitliche Verlauf* der Aufgabe dargestellt wird. Desweiteren sind die beiden Zeitpunkte $t = 0$ (Aufgabenbeginn) und $t = 1$ (Aufgabenende) ausgezeichnet, darüberhinaus existieren keine weiteren ausgezeichneten Zeitpunkte.

Zwar sind Aufgaben denkbar, bei denen Transformationen auftreten, die auch zu diskreten Zeitpunkten spezifiziert sind, die *zwischen* Aufgabenbeginn und -ende liegen. Als Beispiel sollen die Seitenflächen eines quaderförmigen Objekts von einer an einem zweiten Manipulator befestigten Kamera inspiziert werden. Die Transformation zwischen dem Objekt und der Kamera wird dadurch spezifiziert, daß zu bestimmten Zeitpunkten die Blickpunkte der Kamera gegeben sind. Ein anderes Beispiel ist das Anbringen einer Sequenz von Schweißpunkten an einem Werkstück. Die Schweißpunkte sind zu bestimmten Zeitpunkten zwischen Aufgabenbeginn und -ende spezifiziert.

Allerdings sind diese Aufgaben nicht mehr *elementar*, da zu den spezifizierten Zeitpunkten weitere Aktionen, wie die Aufnahme eines Bildes oder das Anbringen eines Schweißpunkts, durchgeführt werden müssen. Sie können sinnvoll in weitere, elementare Aufgaben zerlegt werden, bei denen jeweils eine der interessierenden Zeitpunkte (Schweißpunkt bzw. Blickpunkt) das Ende der Elementaraufgabe darstellt.

Die Spezifikation der einzuhaltenden Realzeitbedingungen erfolgt im Formalismus durch folgende Konstrukte:

REALZEIT_CONSTR ::= **DURATION** VERGL_OP REAL_EXP **s**
GESCHW_SPEZ VERGL_OP REAL_EXP **cm/s**
BESCHL_SPEZ VERGL_OP REAL_EXP **cm/s²**

Hierbei können Geschwindigkeit und Beschleunigung für die jeweiligen Freiheitsgrade getrennt spezifiziert werden. Dies ist etwa für die Spezifikation nachgiebiger Bewegungen erforderlich, bei denen manche Freiheitsgrade kraftgeregelt und andere positionsgeregelt – durch Angabe von Verfahrgeschwindigkeit oder Beschleunigung – werden (s. Kap. 5.5.2).

GESCHW_SPEZ ::= **SPEED**
| **SPEED** (DOF_SELEKT)

BESCHL_SPEZ ::= **ACCELERATION**
| **ACCELERATION** (DOF_SELEKT)

5.5 Spezifikation von Beziehungen durch Constraints über Sensorinformation

Nachdem in Kapitel 4.1 dargelegt wurde, daß Sensorinformation zum einen zur Kompensation von durch Ungewißheiten entstehenden Abweichungen und zum anderen zum Erkennen von Terminierungsbedingungen erforderlich ist, werden nun hier die für die Spezifikation von Terminierungsbedingungen erforderlichen Sprachkonstrukte eingeführt. Anschließend wird die Behandlung von auftretenden Kräften und Momenten als für kooperierende Manipulatoren grundlegende Klasse von Sensorinformation diskutiert und in die Spezifikationssprache integriert.

5.5.1 Terminierungsbedingungen

Bei Aufgaben, die ohne Einbeziehung von Sensorinformation spezifiziert werden können, ergibt sich als einzige Terminierungsbedingung das Erreichen der formalen Zeit $\mathbf{T} = \mathbf{1}$. Bei derartigen Aufgabenbeschreibungen wird diese Terminierungsbedingung implizit angenommen und kein sie beschreibendes Sprachkonstrukt verwendet. Wird Sensorinformation in die Aufgabenspezifikation einbezogen, so können sich neben dem zeitlich bedingten Aufgabenende weitere, sensorinformationsbedingte Beendigungskriterien ergeben (s. Kap. 4.4).

Im Spezifikationsformalismus wird man diesem Sachverhalt dadurch gerecht, daß bei der Spezifikation einer Beziehung eine oder mehrere Einzelterminierungsbedingungen angegeben werden können. Diese sind jeweils durch einen Namen bezeichnet. In einem eigenen Abschnitt werden die Einzelbedingungen – referenziert durch ihren Namen – durch einen booleschen Ausdruck zu einer Gesamtbedingung verknüpft, die letztendlich die Beendigung einer Aufgabe bedingt. Es können auch Beziehungen spezifiziert werden, die allein zur Überwachung eines Abbruchkriteriums dienen.

Als Sprachkonstrukte werden die Einzelterminierungsbedingungen zu einer Beziehung in einem eigenen Abschnitt am Ende des Spezifikationsteils der jeweiligen Beziehung aufgelistet. Das Konstrukt zur Beschreibung einer Beziehung wird entsprechend erweitert:

BEZIEHUNG ::= BEZIEHUNG_NAME :
[CONSTRAINT]
[**TERMINATION**:{TERMINIER_BED}$^+$]

Der Spezifikationsteil einer Beziehung, in dem die Einzelterminierungsbedingungen spezifiziert werden, wird durch das Schlüsselwort `TERMINATION` eingeleitet. Auf ihn folgen die durch geometrische, auf Echtzeit oder Sensorinformation basierende Constraints beschriebenen Einzelterminierungsbedingungen TERMINIER_BED, jeweils mit einem Namen bezeichnet:

TERMINIER_BED ::= TERMINIER_BED_NAME : TERM_CONSTR

TERM_CONSTR ::= GEO_CONSTRAINT
| KRAFT_CONSTRAINT
| REALZEIT_CONSTR

Sobald das spezifizierte Constraint erfüllt ist, gilt die zugehörige Terminierungsbedingung als eingetreten. Am Ende der Aufgabenspezifikation werden die Einzelterminierungsbedingungen durch einen booleschen Ausdruck zu – im allgemeinen mehreren – jeweils durch einen eindeutigen Bezeichner gekennzeichneten Gesamtbedingungen zusammengefaßt. Dieser Abschnitt wird mit dem Schlüsselwort

`TERMINATION_CONDITIONS:`

eingeleitet. Jede der Gesamtterminierungsbedingungen ist als boolescher Ausdruck über den Namen von Einzelbedingungen realisiert:

TERMINIER_GESAMT ::= **T = 1**
| TERMINIER_BED_NAME
| **NOT** (TERMINIER_GESAMT)
| (TERMINIER_GESAMT) **AND** (TERMINIER_GESAMT)
| (TERMINIER_GESAMT) **OR** (TERMINIER_GESAMT)

In den Gesamtterminierungsbedingungen kann zusätzlich das Erreichen des Endes der formalen Zeit durch die Bedingung **T = 1** spezifiziert werden. Diese Terminierungsbedingung ist implizit immer vorhanden und wird daher in den Terminierungsbedingungen bei den einzelnen Beziehungen nicht explizit angegeben.

Als Beispiel sei hier das Suchen einer Passung vor dem Einsetzen eines Bolzens angeführt. Ein Bolzen wird entlang einer spiralenartigen Suchtrajektorie (gestrichelt) geführt, bis die gesuchte Passung gefunden wurde (s. Abb. 22). Für diese Aufgabe ergeben sich mehrere Terminierungsbedingungen. Zum einen kann die Aufgabe als beendet angesehen werden, wenn das gesuchte Loch gefunden wurde. Erkennbar ist das etwa daran, daß eines der Momente in X- oder Y-Richtung eine vorgegebene Schwelle übersteigt. Zum anderen ist die Aufgabe auch dann (erfolglos) beendet, wenn die spezifizierte Suchtrajektorie vollständig abgefahren wurde. Diese Aufgabe gilt also dann als beendet, wenn eine von mehreren Terminierungsbedingungen eingetreten ist.

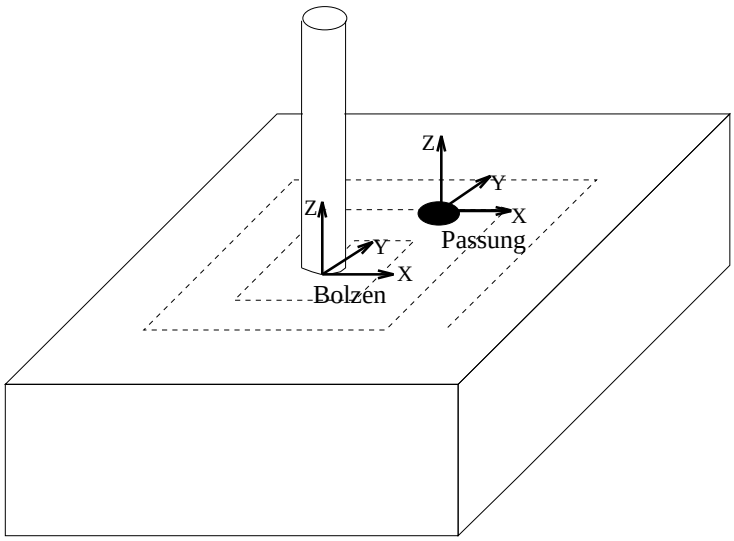


Abbildung 22: Suchtrajektorie beim Einfügen eines Bolzens

Die Gesamtterminierungsbedingung wird wie folgt spezifiziert:

```
TERMINATION_CONDITIONS:
   GEFUNDEN : EINGERASTET
   NICHT_GEFUNDEN : (T = 1)
```

Der Bezeichner `EINGERASTET` verweist dabei auf eine Einzelterminierungsbedingung, die bei der Beziehung zwischen den Koordinatensystemen `Bolzen` und `Passung` spezifiziert wird.

In Kap. 4.4 wurde der Sonderfall von nur durch sensorielle Terminierungsbedingungen beendeten Bewegungen angeführt. Derartige Bewegungen können mit den zur Verfügung stehenden Mitteln durch Angabe folgender Information spezifiziert werden:

- Definierte Startstellung (`START`)
- Angabe der Bewegung in eine bestimmte Richtung. Die Richtung muß einer Achse des Bezugskoordinatensystems entsprechen, was durch geeignete Wahl des Bezugskoordinatensystems immer erreicht werden kann. Die Angabe der Bewegung erfolgt durch Spezifikation einer Geschwindigkeit (`SPEED`) des Ursprungs des bewegten Koordinatensystems relativ zum Bezugskoordinatensystem für diesen Freiheitsgrad.
- Angabe einer Terminierungsbedingung

Als Beispiel für derartige Bewegungen sei das bereits angeführte Beispiel des Einsetzens eines Bolzens erwähnt:

Ein einzusetzender Bolzen (s. Abb. 7, S. 25) soll so lange in Richtung seiner Z-Achse bewegt werden, bis der Bolzen den Boden der Passung erreicht hat, also eine bestimmte Kraftschranke entgegen der Bewegungsrichtung überschritten wird.

```
Einsetzen:
   SPEED(TZ) = v mm/s
   TERMINATION:
      STEHT_AN: FORCE(TZ) > F N

TERMINATION_CONDITIONS:
   FERTIG : STEHT_AN
```

5.5.2 Spezifikation von Beziehungen mit nachgiebiger Bewegung

Die Behandlung von in geschlossenen kinematischen Ketten auftretenden Kräften und Momenten ist eine grundlegende Funktionalität bei der Beschreibung von Elementaraufgaben für kooperierende Manipulatoren. Nach Anforderung **AA5** (*Kräfte und Momente*) wird hier eine Methode zur Spezifikation der auftretenden Kräfte und Momente beschrieben. In

Kapitel 3.3 wurde bereits der von De Schutter entwickelte Formalismus zur Beschreibung von nachgiebigen Bewegungen vorgestellt. Dieser Formalismus kann im wesentlichen zur Spezifikation von Beziehungen mit nachgiebiger Bewegung verwendet werden. In ihm wurde ein Taskframe definiert, in dem die einzuhaltenden Bedingungen für Kräfte und Momente bzw. Verfahrgeschwindigkeiten in den einzelnen Freiheitsgraden spezifiziert werden. Einen Schwachpunkt stellt lediglich die Spezifikation der Lage des Taskframes dar, die in [Sch86] informell verbal spezifiziert wird. Im Rahmen des hier vorgestellten Formalismus wird der Formalismus von De Schutter nun dahingehend erweitert, daß die Lage des Taskframes nun exakt spezifiziert werden kann. Der Fall, daß das Taskframe sich entlang einer modellierten Bahn bewegt, wird in [Sch86] zwar erwähnt, aber nicht näher ausgeführt. Dieser Fall wird ebenfalls in dem hier vorgestellten Formalismus behandelt.

Lage des Taskframes Die Spezifikation der Lage des Taskframes in dem hier vorgestellten Formalismus erfolgt dadurch, daß das Taskframe als Hilfskoordinatensystem spezifiziert wird. Seine Lage kann dann durch Beziehungen zu den die beteiligten Objekte darstellenden Koordinatensystemen spezifiziert werden. Dies erfolgt durch eine feste Transformation zu einem der beteiligten Objekte. Beim in Kap. 3.3 als Beispiel vorgestellten Einsetzen eines Bolzens in eine Passung etwa wird das Taskframe durch eine feste Transformation zum einzusetzenden Bolzen spezifiziert, beim Drehen einer Kurbel durch eine feste Transformation – im Griff liegend – zur Kurbel.

Ein bei De Schutter angesprochener, aber dort nur teilweise gelöster Fall sind Taskframes, die sich entlang einer geometrisch spezifizierten Bahn bewegen. In dem hier vorgestellten Formalismus erfolgt die Spezifikation folgendermaßen: Die Bahn an dem gewünschten Objekt wird durch ein Hilfskoordinatensystem (*Modellframe*), das durch eine zeitabhängige Transformation in Beziehung zum Objekt gesetzt wird, spezifiziert. Die Lage des Taskframes wird dann relativ zu diesem Modellframe unter Verwendung modellbasierend verfolgender Freiheitsgrade (siehe unten) spezifiziert. Als Beispiel soll das Entgraten eines Spritzgußwerkstücks dienen: Im geometrischen Modell ist die endgültige Kontur des Werkstücks festgelegt. Somit kann die von der Toolspitze zu verfahrende Trajektorie als eine dieser Kontur entsprechende zeitabhängige Transformation zwischen dem Objektursprung und dem Modellframe spezifiziert werden, welches dann die Lage des Taskframes beschreibt (s. Beispiel 2, S. 79).

Der im Ansatz von De Schutter problematische Fall, daß sich das Taskframe relativ zu *beiden* beteiligten Objekten bewegen kann, und der durch die Einführung einer Bewegungseinschränkungsmatrix gelöst wurde (s. Kap. 3.3), kann mit den hier zur Verfügung stehenden Mitteln wesentlich eleganter formuliert werden. Da von einem Taskframe aus Beziehungen zu den beiden beteiligten Objekten zu modellieren sind, werden zur Spezifikation der Lage des Taskframes in jeder Beziehung nur die durch die Aufgabe jeweils festgelegten Freiheitsgrade spezifiziert. Im Falle des in Kap. 3.3 beschriebenen Entlangziehens eines Stifts an einer Tischkante wird in Beispiel 3 (Abb. 25, S. 81) die Z-Koordinate des Taskframes relativ zum Tisch (gleich der Z-Koordinate der Tischkante) spezifiziert,

alle anderen Freiheitsgrade werden relativ zum Stift spezifiziert.

Freiheitsgrade des Taskframes In einem Teil der Spezifikation sind die Typen der Freiheitsgrade des Taskframes zu spezifizieren. Wie in Kapitel 3.3 dargelegt, gibt es folgende Typen von Freiheitsgraden

- kraftgeregelt

  Ein Freiheitsgrad von diesem Typ wird kraftgeregelt. Die Sollkraft (bzw. das Sollmoment) wird spezifiziert. Kraftgeregelte Freiheitsgrade dienen zum Aufbringen einer spezifizierten Kraft bzw. zum Kompensieren von Kräften, die beim Kontakt starrer Gegenstände entstehen. Im Formalismus erfolgt dies durch ein Kraft-Constraint der Form:

KRAFT_CONSTR ::= **FORCE** (DOF_SELEKT) VERGL_OP REAL_EXP **N**
| **TORQUE** (DOF_SELEKT) VERGL_OP REAL_EXP **Nm**

- positionsgeregelt

  Für einen Freiheitsgrad von diesem Typ wird die Geschwindigkeit (bzw. Beschleunigung) in dieser Richtung spezifiziert. Die Spezifikation von Geschwindigkeit und Beschleunigung für einen Freiheitsgrad ist in Kap. 5.4 beschrieben.

- modellbasiert verfolgend

  Ein Freiheitsgrad von diesem Typ wird in seiner Lage mit dem entsprechenden Freiheitsgrad des geometrischen Modells der Bahn des Taskframes zur Übereinstimmung gebracht. Er ist also nur durch das geometrische Modell (bzw. der daraus abgeleiteten Modellierung einer Bahn) spezifiziert. Im Formalismus wird ein Freiheitsgrad von diesem Typ durch folgendes Konstrukt spezifiziert:

TRACK_CONSTR ::= **TRACK** (DOF_SELEKT)

Freiheitsgrade, für die weder eine natürliche, also durch die Struktur der Aufgabe gegebene, noch eine künstliche, vom Spezifizierenden gegebene Einschränkung existiert, werden bei De Schutter's Formalismus (Kap. 3.3) durch Angabe einer Diagonalmatrix mit 0-Elementen gekennzeichnet. Der Grund für die Einführung dieser Matrix ist die direkte Umsetzbarkeit der Spezifikation in eine Regelungsvorschrift. Im hier vorgestellten Formalismus werden sie in der Spezifikation weggelassen. Sie können, wie bereits De Schutter erwähnt, als unspezifizierte Freiheitsgrade vom Bewegungsplaner genutzt werden.

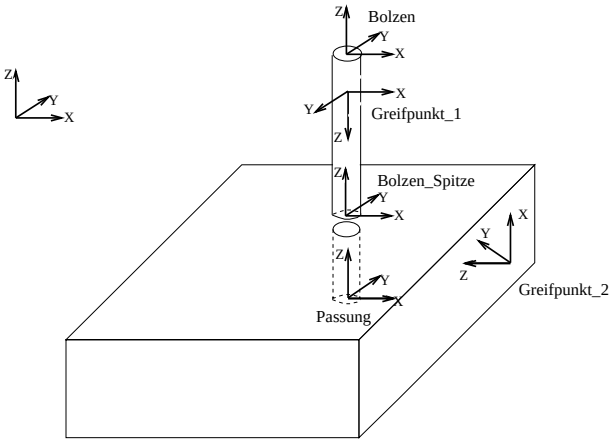
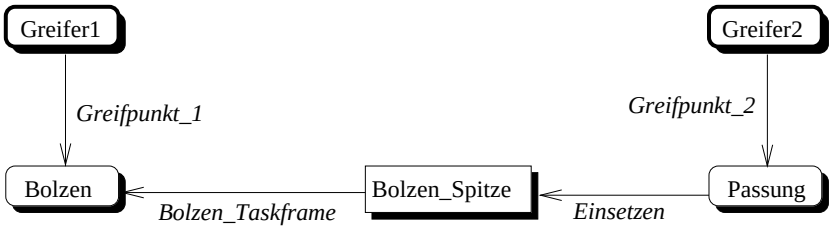


Abbildung 23: Lage der Koordinatensysteme beim Beispiel *Einfügen eines Bolzens*

Beispiel 1: Einfügen eines Bolzens in eine Passung (s. Abb. 23)

```
TASK: Bolzen_Montage
ENVIRONMENT: Zwei_Pumas
GRAPH:
```




```
RELATIONS:

Greifpunkt_1: (0, 0, -40, 0, 0, 0)

Greifpunkt_2: (40, 20, 10, 0, 90, 0)

Bolzen_Taskframe: (0, 0, 60, 0, 0, 0)

Einsetzen:
   START: (0, 0, 50, 180, 0, 0) AND
   FORCE(TX) = 0 N AND
   FORCE(TY) = 0 N AND
   SPEED(TZ) = v mm/s AND
   TORQUE(RX) = 0 Nm AND
   TORQUE(RY) = 0 Nm AND
   SPEED(RZ) = 0 grad/s
   TERMINATION :
      STEHT_AN: FORCE(TZ) < -F N

TERMINATION_CONDITIONS :
   BOLZEN_EINGESETZT : STEHT_AN OR ( T = 1 )
```

Beispiel 2: Entgraten eines Werkstücks

`Entgratbahn` beschreibt die Bahn des Modellframes, der das Entgratungswerkzeug folgen soll. Das Taskframe ist in der Spitze des Entgratungswerkzeugs zu modellieren. Das Entgraten erfolgt mit einer vorgegebenen Kraft entlang der Entgratungsrichtung. `Entgraten` stellt die Beziehung zwischen Taskframe und Modellframe dar.

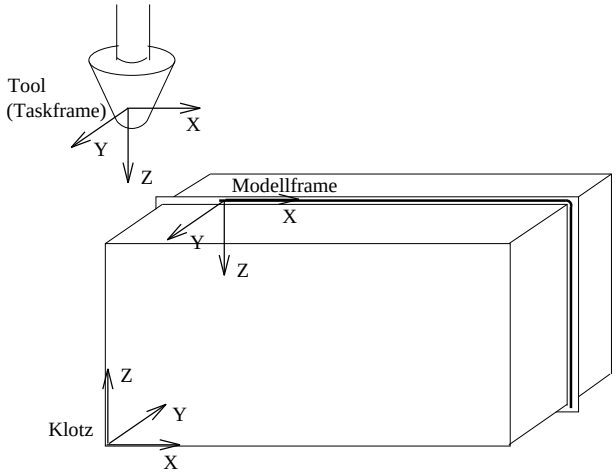
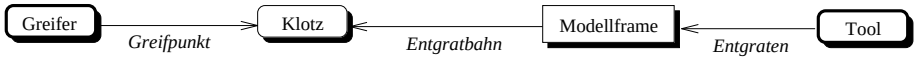


Abbildung 24: Entgraten eines Werkstücks

```
TASK: Entgraten
ENVIRONMENT: Zwei_Pumas
GRAPH:
```




```
RELATIONS:

Entgratbahn:
   LINEAR_INTERPOLATED
      (0, 10, 30, 0, 0, 0),
      (60, 10, 30, 0, 0, 0),
      (60, 10, 30, 0, -90, 0),
      (60, 10, 0, 0, -90, 0)
   END_INTERPOLATED

Entgraten:
   START: (100, 220, 40, 90, 90, 0) AND
   FORCE(TX) = 2 N AND
   TRACK(TY) AND
   TRACK(TZ) AND
   TRACK(RX) AND
   TRACK(RY) AND
   TRACK(RZ)
```

Beispiel 3: Entlangziehen eines Stabs an einer Tischkante

Der in Abb. 25 dargestellte Stab soll entlang seiner Z-Achse an der Kante des Tisches entlanggezogen werden. Dabei soll auf die Tischkante eine Kraft von 2 N ausgeübt werden.

Im Graphen sind die für dieses Beispiel interessanten Beziehungen `Stab_Taskframe` und `Entlangziehen` dargestellt, die wie folgt spezifiziert werden:

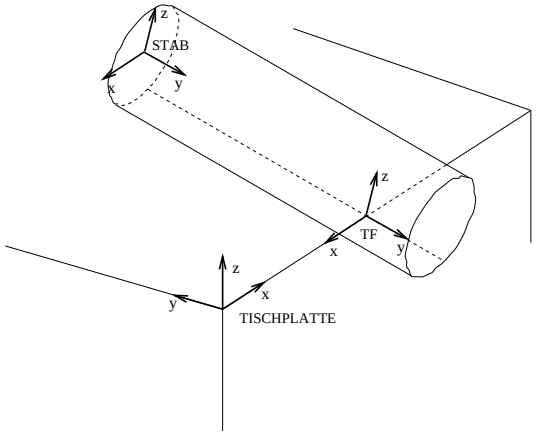
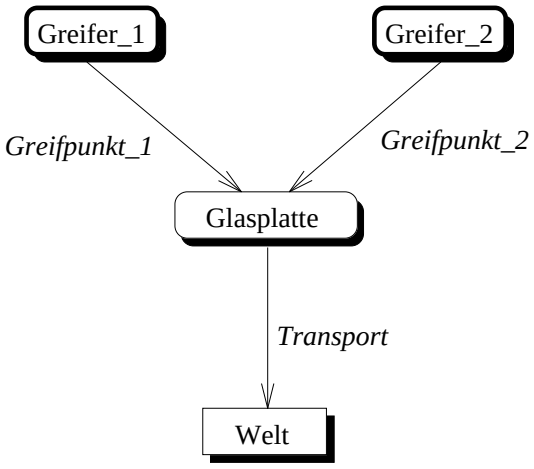


Abbildung 25: Ziehen eines Stabs über eine Tischkante (nach [Sch86])


```
Stab_Taskframe:
   START : (0, 100, 10, 0, 0, 0) AND
   TX(t) = 0 AND
   TZ(t) = - 10 AND
   RX(t) = 0 AND
   RY(t) = 0 AND
   RZ(t) = 0

Entlangziehen:
   START: (100, 0, 0, 0, 0, 180) AND
   TY(t) = 0 AND
   SPEED(TX) = 0 cm/s AND
   SPEED(TY) = v cm/s AND
   FORCE(TZ) = 2 N AND
   SPEED(RX) = 0 grad/s AND
   SPEED(RY) = 0 grad/s AND
   SPEED(RZ) = 0 grad/s
```

Beispiel 4: Transport einer Glasplatte; die Glasplatte soll so transportiert werden, daß in den Greifpunkten keine Kräfte auftreten, die zur Beschädigung der Glasplatte führen könnten.

```
TASK: Glasplattentransport
ENVIRONMENT: Zwei_Pumas
GRAPH:
```




```
RELATIONS:

Greifpunkt_1 :
   START : (-200, 0, 100, 0, 0, 0) AND
      FORCE ( TX ) = 0 N AND
      FORCE ( TY ) = 0 N AND
      TORQUE ( RX ) = 0 Nm AND
      TORQUE ( RY ) = 0 Nm AND
      TORQUE ( RZ ) = 0 Nm

Greifpunkt_2 :
   START : (200, 0, 100, 0, 0, 0) AND
      FORCE ( TX ) = 0 N AND
      FORCE ( TY ) = 0 N AND
      TORQUE ( RX ) = 0 Nm AND
      TORQUE ( RY ) = 0 Nm AND
      TORQUE ( RZ ) = 0 Nm

Transport :
   START : (100, -450, 100, 0, 0, 0) AND
   GOAL : (400, 800, 100, 0, 0, 0)
```

6 Bahnplanung für kooperierende Manipulatoren

Wie bereits in Kap. 3.4 dargelegt, sind im Rahmen der aufgabenorientierten Programmierung die durch den in Kap. 4 und 5 vorgestellten Formalismus spezifizierten Aufgaben nun durch

- Bahnplanungssysteme
- Skalierung der formalen zur realen Zeit und
- Generierung bzw. Parametrisierung von Regelungsvorschriften

in ein von Manipulatoren ausführbare Form umzusetzen. Im folgenden wird ein Bahnplanungsverfahren vorgestellt. Die Behandlung der Skalierung der formalen zur realen Zeit [MA90] und der Generierung bzw. Parametrierung von Regelungsvorschriften [BAH94] würde den Rahmen dieser Arbeit sprengen. Bezüglich dieser Teilprobleme sei auf die zitierten Ansätze verwiesen.

Neben allgemeinen Anforderungen an ein Bahnplanungsverfahren (**PL1**: Planung kollisionsfreier Bahnen, **PL2**: realistische Umgebungen, **PL3**: hohe Effizienz) ergeben sich gegenüber bisher existierenden Bahnplanungsverfahren neue Anforderungen durch die betrachtete Aufgabenklasse. Dies sind spezifizierte Zeitabhängigkeiten (**PL4**), variable Anzahl von Freiheitsgraden für verschiedene Aufgaben und Bereichseinschränkungen für die Freiheitsgrade durch geometrische Constraints (**PL5**), sowie Ziel*räume* als Planungsziele (**PL6**) anstelle von Zielkonfigurationen, wie sie in existierenden Bahnplanungsverfahren Verwendung finden.

Anhand einer Teilklasse der mit der in Kap. 5 vorgestellten Sprache spezifizierbaren Aufgaben werden die zu ihrer Umsetzung in kollisionsfreie, synchrone Bahnen für kooperierende Manipulatoren erforderlichen Bahnplanungsstrategien dargestellt. Diese Teilklasse wird in Kap. 6.2 beschrieben. Ergebnisse von Planungsläufen für verschiedene Aufgaben mit einem Prototypen werden im Anschluß an die Darstellung der Planungsstrategien vorgestellt.

Im Anschluß daran wird in Kap. 6.9 skizziert, wie die vorgestellten Planungsstrategien für den allgemeinen Fall erweitert werden können.

6.1 Darstellung von Konfigurationsräumen bei kooperierenden Manipulatoren

Möchte man den Konfigurationsraum mehrerer kooperierender Manipulatoren $R1, \ldots, Rn$ betrachten, so besteht der naheliegendste Weg darin, diesen als das kartesische Produkt der Konfigurationsräume der Einzelmanipulatoren darzustellen. Als Darstellungsart für die einzelnen Konfigurationsräume können beide oben verwendeten Verfahren, also sowohl die Gelenkwinkeldarstellung als auch die kartesische Darstellung, verwendet werden.

$$\mathbf{C}^{(R1,\ldots,Rn)} = \mathbf{C}^{R1} \times \mathbf{C}^{R2} \times \ldots \times \mathbf{C}^{Rn}$$

Zusätzlich sind durch die hier betrachtete Aufgabenstellung Abhängigkeiten der Effektorkoordinatensysteme gegeben. Somit kann man den freien Konfigurationsraum $\mathbf{C}_{free}$ als die Menge der Konfigurationen beschreiben, in denen

- keiner der Einzelmanipulatoren mit einem Arbeitsraumobjekt kollidiert
- die Einzelmanipulatoren nicht miteinander kollidieren und
- die Effektorkoordinatensysteme die geforderten Abhängigkeiten erfüllen.

Wird zur Konfigurationsraumdarstellung für die Einzelmanipulatoren die kartesische Stellung des Effektorkoordinatensystems verwendet, so müssen zusätzlich für die Konfigurationen des freien Konfigurationsraums

- die inverse Kinematik der so beschriebenen Manipulatoren eine Lösung besitzen und
- die zu dieser Lösung gehörenden Gelenkwinkelparameter innerhalb der zulässigen Grenzen liegen.

Zur Darstellung des freien Konfigurationsraums werden in den derzeit leistungsfähigsten Bahnplanungsverfahren (s. Kap. 3.5.1) für praktisch relevante Aufgaben im allgemeinen Diskretisierungsverfahren verwendet. Methoden zur direkten Transformation des realen Raums in den Konfigurationsraum, wie sie etwa in [Lat91] beschrieben werden, sind für realistische Anwendungen in höheren Dimensionen aus Effizienzgründen nicht praktikabel. Durch Abhängigkeiten zwischen den Effektorkoordinatensystemen kann die Dimension des freien Konfigurationsraums gegenüber dem freien Konfigurationsraum zweier Manipulatoren ohne Abhängigkeiten zwischen den Effektorkoordinatensystemen verringert werden. An einem einfachen zweidimensionalen Beispiel wird die Problematik dieser Verringerung der Dimension des freien Konfigurationsraums aufgezeigt (Abb. 26). Es werden zwei eingliedrige Linienmanipulatoren derselben Länge (**R1** und **R2**) dargestellt. In der ersten Abbildung jeder Zeile werden die Manipulatoren in ihrem Arbeitsraum dargestellt, in der zweiten der zugehörige freie Konfigurationsraum, aufgetragen über den beiden Gelenkwinkeln der Manipulatoren, in der dritten der jeweilige diskretisierte Konfigurationsraum. In Abb. 26 ist die diskretisierende Darstellung für zwei Beispiele skizziert. In Beispiel 1) ist der freie Konfigurationsraum durch die Zusatzbedingung eingeschränkt, daß der Abstand Δ zwischen den Spitzen der beiden Linienmanipulatoren **R1** und **R2** kleiner als **d** bleiben muß. In 1b) ist der freie Konfigurationsraum und in 1c) der diskretisierte freie Konfigurationsraum dargestellt.

Bei Diskretisierungsverfahren werden einzelne Konfigurationen rasterförmig auf Zugehörigkeit zum freien Konfigurationsraum getestet (Abb. 26, 1c, 2c). War die getestete Konfiguration kollisionsfrei, so wird daraus auf die Kollisionsfreiheit einer Umgebung des getesteten

Punkts bis zum halben Abstand zu den benachbarten Rasterpunkten geschlossen. Voraussetzung für dieses Verfahren ist, daß die Dimension des freien Konfigurationsraums gleich der Dimension des gesamten Konfigurationsraums ist, wie in Abb. 26, Beispiel 2) gezeigt wird.

Hierbei wird der freie Konfigurationsraum durch die Forderung, daß der Abstand Δ zwischen den Effektorkoordinatensystemen gleich dem Basisabstand d ist, auf einen eindimensionalen Unterraum eingeschränkt, dargestellt als Linie in 2b). Diese Reduktion der Dimension des freien Konfigurationsraums ist für Bahnplanungsverfahren allerdings problematisch: Da bei der Diskretisierung (Beispiel 2c) Teilräume mit niedrigerer Dimension bei den Tests nicht getroffen werden (im allgemeinen ist bei einer freien, bei der Diskretisierung getroffenen Wahl der Gelenkwinkel die Bedingung $\Delta = d$ nur in Ausnahmefällen erfüllt), kann ein freier Konfigurationsraum, dessen Dimension unter der des Konfigurationsraums liegt, mit dieser Methode nicht diskretisiert dargestellt werden.

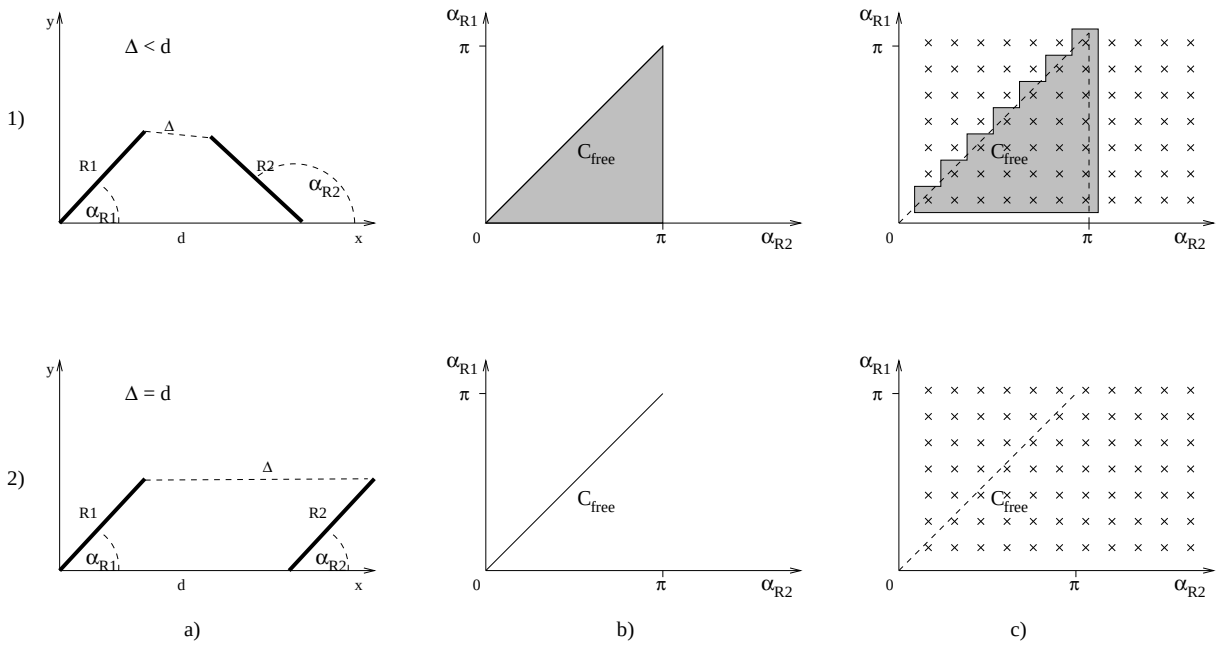


Abbildung 26: Diskretisierende Darstellung von Konfigurationsräumen

Zur Bahnplanung mit einem Planer, der auf der Diskretisierung des Konfigurationsraums basiert, ist also eine Darstellung des Konfigurationsraums erforderlich, bei der die Dimension des freien Konfigurationsraums gleich der Dimension des Konfigurationsraums ist. Um eine Möglichkeit einer Konfigurationsraumdarstellung herzuleiten, die dieser Anforderung gerecht wird, betrachten wir zunächst den Fall, daß die Effektorkoordinatensysteme vollständig, ohne Freiheitsgrade voneinander abhängen.

6.1.1 Darstellung des Konfigurationsraums bei fester Abhängigkeit der Effektorkoordinatensysteme

Sind zwei Manipulatoren durch eine *feste* Transformation zwischen den Effektorkoordinatensystemen gekoppelt, so kann die Konfiguration der beiden Manipulatoren folgendermaßen dargestellt werden:

Ist die Konfiguration eines Manipulators bekannt, so kann über die Transformation zwischen den Effektorkoordinatensystemen die Lage des Effektorkoordinatensystems des zweiten Manipulators bestimmt werden. Wie in Kap. 2.3.1 dargelegt wurde, ist die Konfiguration eines Manipulators aber durch die Lage seines Effektorkoordinatensystems und einen gegebenen *kinematischen Zustand* k_2 bestimmt. Somit kann der Konfigurationsraum $\mathbf{C}$ der beiden Manipulatoren durch die Beschreibung des Konfigurationsraums eines Manipulators und die Angabe des kinematischen Zustands für den zweiten Manipulator beschrieben werden.

Die Darstellung des Konfigurationsraums des ersten Manipulators kann – wie in Kap. 2.3.1 beschrieben – durch Gelenkparameter bzw. durch die Lage des Effektorkoordinatensystems und einen kinematischen Zustand k_1 erfolgen. Somit gilt für den Konfigurationsraum der beiden Manipulatoren:

Für die Gelenkparameterdarstellung:

$$\mathbf{C} = W_1 \times \ldots \times W_n \times k_2$$

Für die kartesische Darstellung mit Hilfe der Effektorkoordinatensysteme

$$\mathbf{C} = \mathbb{R}^{\mathbf{T}} \times \mathbb{R}^{\mathbf{R}} \times k_1 \times k_2$$

Der kinematische Zustand k_2 des über die inverse Kinematik betrachteten Manipulators im Fall der Gelenkwinkeldarstellung bzw. der *zusammengesetzte kinematische Zustand* $k_1 \times k_2$ der beiden über die inverse Kinematik betrachteten Manipulatoren im kartesischen Fall stellt also eine (diskrete) Dimension des Konfigurationsraums dar.

6.1.2 Freiheitsgrade in den Abhängigkeiten der Effektorkoordinatensysteme

In der Transformation zwischen den Effektorkoordinatensystemen treten bei verschiedenen Aufgaben Freiheitsgrade auf. Diese können etwa dadurch entstehen, daß die Transformation zwischen den Effektorkoordinatensystemen nicht in all ihren Freiheitsgraden spezifiziert ist. Ein Beispiel hierfür ist etwa folgende Aufgabe: Für einen Manipulator ist spezifiziert, daß die als Tool montierte Kamera ständig auf ein von einem anderen Manipulator bewegtes Objekt blickt. Dazu wird die Orientierung der Kamera relativ zum Objekt voll spezifiziert, ebenso die Position auf dem Objekt, die mit der optischen Achse der Kamera zusammenfällt. Allerdings kann die Entfernung der Kamera vom Objekt so spezifiziert

werden, daß sie in einem festgelegten Bereich variieren kann. Zur eindeutigen Beschreibung der Konfiguration ist somit auch eine Beschreibung dieses translatorischen Freiheitsgrads in der Transformation zwischen Kamera- und Objektkoordinatensystem erforderlich.

Ein weiterer, bei vielen Aufgaben auftretender Freiheitsgrad in der Transformation zwischen den Effektorkoordinatensystemen ist die formale Zeit. So kann etwa eine zu verfahrende Bearbeitungstrajektorie (etwa eine aufzubringende Schweißnaht) als eine solche zeitabhängige Transformation zwischen dem Effektor des bearbeitenden Manipulators und dem Effektor des Manipulators, der das Werkstück hält, dargestellt werden.

Durch die Einbeziehung zusätzlicher Freiheitsgrade in der Transformation zwischen den Effektorkoordinatensystemen in die Aufgabenbeschreibung wird der entsprechende Konfigurationsraum $\mathbf{C}_{\mathbf{ges}}$ um n Dimensionen, die diese n Freiheitsgrade beschreiben, erweitert:

$$\mathbf{C}_{\mathbf{ges}} = \mathbf{C} \times \mathbb{R}^n$$

6.2 Beschreibung der für die Bahnplanung betrachteten Aufgabenklasse

Der in den Kapiteln 4 und 5 beschriebene Spezifikationsformalismus für Aufgaben für kooperierende Manipulatoren ist sehr ausdrucksmächtig. Für die Darstellung der Planungsstrategien wird anhand einer Teilmenge der spezifizierbaren Aufgaben deren Umsetzung durch Bahnplanungswerkzeuge in die Bewegungsebene dargestellt. Im Anschluß an die Beschreibung der Planungsstrategien wird dann in Kap. 6.9 skizziert, wie die vorgestellten Planungsstrategien für eine umfassende, durch geometrische Constraints spezifizierte Klasse von Aufgaben verwendet werden können.

Die Klasse der mit dem in Kap. 4 beschriebenen Formalismus spezifizierbaren Aufgaben, für die das hier beschriebene Bahnplanungsverfahren konstruiert wurde, ist durch folgende Punkte charakterisiert, die Aufgabenstruktur mit den beiden kinematischen Ketten, den beteiligten Koordinatensystemen und den zwischen diesen auftretenden Beziehungen sind in Abb. 27 dargestellt:

- Es werden nichtredundante Manipulatoren verwendet. Ihre Basiskoordinatensysteme sind fest, gegeben durch die Transformationen $\mathbf{T}^{\mathbf{R1}}_{\mathbf{Welt}}$ und $\mathbf{T}^{\mathbf{R2}}_{\mathbf{Welt}}$.
- Es gibt ein ausgezeichnetes Koordinatensystem (*Objektkoordinatensystem*) **Obj** in der Aufgabenstruktur (s. Abb. 27). Für die Beziehung zwischen Weltkoordinatensystem und Ursprungskoordinatensystem eines manipulierten Objekts können durch geometrische Constraints beliebige Werte- oder Bereichseinschränkungen für die einzelnen Freiheitsgrade spezifiziert werden.
- Die Beziehungen zwischen dem Objektkoordinatensystem **Obj** und je einem Effektorkoordinatensystem (**T1, T2**) werden ohne offene Freiheitsgrade spezifiziert. Für sie

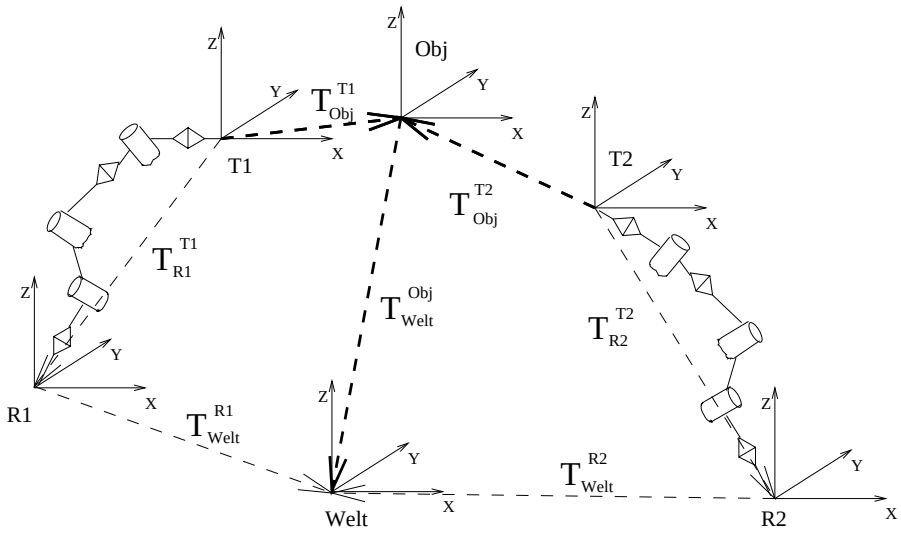


Abbildung 27: Struktur der betrachteten Aufgaben

ist zu jedem formalen Zeitpunkt *genau eine* Transformation $(\mathbf{T^{T1}_{Obj}}, \mathbf{T^{T2}_{Obj}})$ spezifiziert. Dies sind Beziehungen, die entweder feste Transformationen (durch das Sprachkonstrukt `FIXED`) oder fest vorgegebene Bewegungen (Sprachkonstrukte `INTERPOLATED` bzw. Spezifikation einer Funktion der Zeit für jeden translatorischen und rotatorischen Freiheitsgrad) spezifizieren. Somit ist die Lage der Effektorkoordinatensysteme relativ zum Objektkoordinatensystem zu jedem formalen Zeitpunkt wohldefiniert.

- Die Kette der Beziehungen $\mathbf{T^{Obj}_{Welt}}$ zwischen dem Objekt- und dem Weltkoordinatensystem **Welt** ist für den Startzeitpunkt der Aufgabe durch *genau eine* Transformation spezifiziert. Eine Transformation für den Zielzeitpunkt kann spezifiziert sein. Ist diese nicht spezifiziert, so muß zur sinnvollen Vorgabe eines Planungsziels eine der Transformationen $(\mathbf{T^{T1}_{Obj}}, \mathbf{T^{T2}_{Obj}})$ als zeitabhängig spezifiziert sein.

- Es sind außer den genannten keine weiteren Beziehungen durch geometrische Constraints spezifiziert.

- Da für Bahnplanungsprobleme die Einbeziehung von Sensorinformation nicht relevant ist, werden hier nur Aufgaben betrachtet, die durch geometrische Constraints spezifiziert sind. Sich auf Sensorinformation beziehende Constraints werden für die Bahnplanung nicht berücksichtigt.

Durch diese Teilmenge spezifizierbarer Aufgaben wird bereits eine große Menge praxisrelevanter Aufgaben abgedeckt: Es sind dies zum einen Aufgaben, bei denen ein Objekt von einer Start- zu einer Zielstellung transportiert wird, wobei der Transportvorgang gemeinsam von beiden Manipulatoren durchgeführt wird bzw. einer der beiden Manipulatoren eine relativ zum Objekt vorgegebene Bewegung durchführt. Zum anderen sind dies Aufgaben,

bei denen ein Manipulator ein zu manipulierendes Objekt hält, während der zweite Manipulator eine Relativbewegung zum Objekt (etwa eine Bearbeitungstrajektorie) durchführt. Die Lage des Objekts ist dabei frei oder durch geometrische Constraints in einigen Freiheitsgraden eingeschränkt.

Aufgabe des Bahnplaners ist es nun, ein Paar von synchronen Manipulatorbewegungen, darstellbar als zeitabhängige Transformationen und zugehörige kinematische Zustände $(\mathbf{T}^{\mathbf{T1}}_{\mathbf{R1}}(t), k_1, \mathbf{T}^{\mathbf{T2}}_{\mathbf{R2}}(t), k_2)$, zu finden, sodaß alle Konfigurationen dieser synchronen Bewegungen im zur Aufgabe gehörenden freien Konfigurationsraum $\mathbf{C}_{free}$ liegen.

6.3 Darstellung des Konfigurationsraums bei den betrachteten Aufgaben

Bei den betrachteten Aufgaben existiert zu jedem Zeitpunkt eine definierte Transformation zwischen den Effektorkoordinatensystemen der beiden Manipulatoren. Somit kann die Stellung eines Manipulators berechnet werden, wenn sein kinematischer Zustand vorgegeben und die Stellung des zweiten Manipulators bekannt ist. Zur Darstellung des Konfigurationsraums der beiden Manipulatoren kann dann eine Beschreibung des Konfigurationsraums *eines* Manipulators und des kinematischen Zustands des anderen Manipulators verwendet werden.

Als zusätzlicher Freiheitsgrad in der Transformation zwischen den Effektorkoordinatensystemen tritt die formale Zeit t auf. Eine Besonderheit dieses Freiheitsgrads stellt die geforderte Monotonie eines Weges in dieser Dimension dar: Der geplante Weg darf keinen Rückschritt in der formalen Zeit machen.

Somit bieten sich folgende Möglichkeiten zur Darstellung des Konfigurationsraums an (translatorische Freiheitsgrade im $\mathbb{R}^3$: $\mathbf{T} = 3$, rotatorische Freiheitsgrade $\mathbf{R} = 3$):

1 Verwendung einer Gelenkparameterdarstellung für das Effektorkoordinatensystem eines Manipulators. Die Stellung des zweiten Manipulators ist über die Transformation zwischen den Effektorkoordinatensystemen und einen vorgegebenen kinematischen Zustand bestimmt.

$$\mathbf{C}_1 = \mathbb{R}^6 \times k_2 \times t$$

2a Verwendung einer kartesischen Darstellung für einen Manipulator. Die Stellung des zweiten Manipulators ist wie unter 1) bestimmt.

$$\mathbf{C}_2 = \mathbb{R}^{\mathbf{T}} \times \mathbb{R}^{\mathbf{R}} \times k_1 \times k_2 \times t$$

2b Verwendung des Objektkoordinatensystems. Dies ist eine Variante von 2a), lediglich wird statt des TCP eines Manipulators das Objektkoordinatensystem verwendet, um

Aufgabe	Füllungsgrad
TRANSPORT	99.86 %
KISTE	99.93 %
STANGE	99.83 %
STANGE_TISCH	99.9 %
PLATTE	99.82 %

Tabelle 1: Füllungsgrad des Konfigurationsraums

die Stellung der beiden Manipulatoren zu ermitteln. Diese sind über die Transformationen zwischen dem Objekt- und den Effektorkoordinatensystemen und jeweils einen kinematischen Zustand bestimmt. Die Darstellung des Konfigurationsraums erfolgt wie in 2a), lediglich wird statt des TCP-Koordinatensystems eines Manipulators ein Objektkoordinatensystem zur Konfigurationsraumdarstellung verwendet.

Bei der Konfigurationsraumdarstellung 1) wird im Gegensatz zu 2a) und 2b) keine zusätzliche Beschreibung des kinematischen Zustands erforderlich. Wie bereits in Kap. 2.3.3 geschildert, stellt dies aber keinen Nachteil hinsichtlich des Suchraums dar, da im allgemeinen zu jeder Start- bzw. Zielstellung mehrere Gelenkparametersätze existieren.

Die Ansätze 2a) und 2b) unterscheiden sich lediglich um die Transformation zwischen Effektor- und Objektkoordinatensystem. Als symmetrische Lösung bietet sich zur Darstellung des Konfigurationsraums Methode 2b) an. Ein weiterer Vorteil ist die leichtere Formulierbarkeit von Randbedingungen für das Objektkoordinatensystem (z.B.: "Das Objekt soll während des Transports nicht gekippt werden").

In Abb. 28 sind eine Konfiguration aus einer Transportaufgabe (oben) und eine Darstellung zweier Dimensionen des zugehörigen Konfigurationsraums (unten) abgebildet. Die x- und y-Koordinaten des Konfigurationsraums entsprechen der Translation des von beiden Manipulatoren gegriffenen Objekts entlang der x- bzw. y-Achse. Die hellgrauen Flächen stellen Konfigurationsraumhindernisse dar, die durch Überschreitung der Gelenkwinkel eines oder beider Manipulatoren entstehen. Die mittelgrauen Flächen entstehen durch auftretende Kollisionen. Ein für die Entwicklung von Planungsstrategien wesentlicher Punkt ist der Anteil des freien Konfigurationsraums am gesamten Konfigurationsraum. Während der Anteil der Konfigurationsraumhindernisse in typischen Beispielen aus [Gla91b] bei 20 bis 70 Prozent liegt, wurden für Aufgaben für kooperierende Manipulatoren wesentlich höhere Füllungsgrade – das ist der Anteil der Konfigurationsraumhindernisse am gesamten Konfigurationsraum – des Konfigurationsraums ermittelt (s. Tabelle 1 und Kap. 6.8, S. 117). Der hohe Füllungsgrad des hier verwendeten Konfigurationsraums hat Auswirkungen auf die Konstruktion der Planungsstrategien und wird im folgenden näher diskutiert (s. Kap. 6.4).

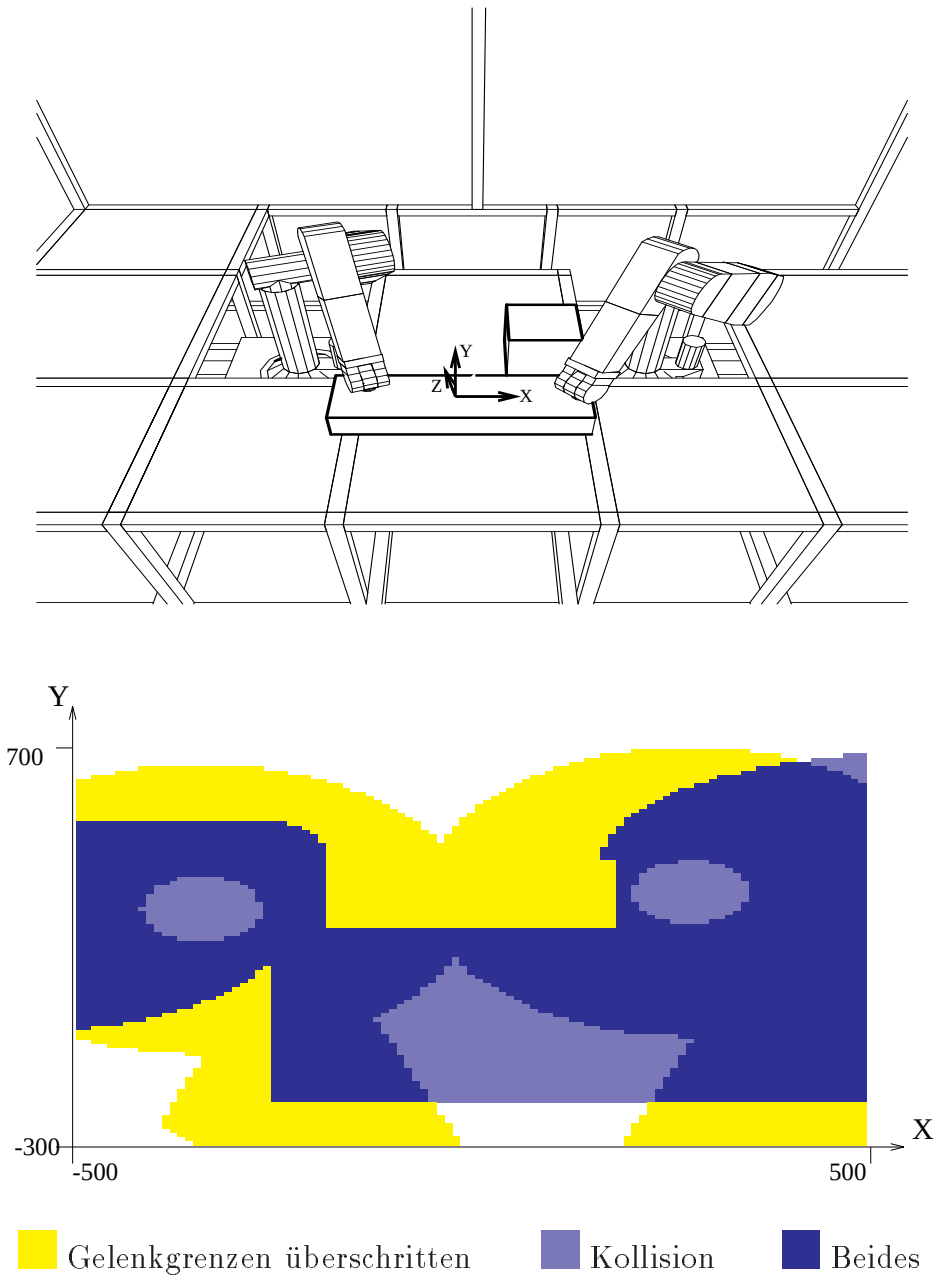


Abbildung 28: Zweidimensionale Projektion eines Konfigurationsraums

6.4 Zielräume

Vor der Darstellung der lokalen und globalen Planungsstrategien wird hier das Konzept der Zielräume eingeführt.

Die Verwendung von Zielräumen wird durch zwei Sachverhalte motiviert:

- Bei verschiedenen Aufgaben können durch die Spezifikationssprache globale Ziele mit verschiedener Dimension vorgegeben werden. Bei einer Transportaufgabe beispielsweise kann das Ziel durch eine einzelne Konfiguration, nämlich eine vorgegebene Stellung zur formalen Zeit $t = 1$, vorgegeben werden. Ebenso kann das Abstellen eines Objekts auf einer Tischplatte in einem vorgegebenen Bereich spezifiziert werden, was die Z-Translation sowie die X- und Y-Rotation der Beziehung zwischen Tisch und Objekt auf einen vorgegebenen Wert festlegt. Desweiteren ist bei einer Manipulationsaufgabe als Ziel lediglich *irgendeine* Konfiguration zur formalen Zeit $t = 1$ zu erreichen. Für eine allgemeine Beschreibung von Zielen von Planungsaufgaben sind also Ziel*räume* erforderlich. Einzelne Ziel*konfigurationen* sind hierfür nicht mehr ausreichend.
- Wie in Kap. 6.3 dargelegt wurde, ist der Anteil des freien Konfigurationsraums am gesamten Konfigurationsraum sehr niedrig. Umso wichtiger ist die Effizienz des lokalen Planers. Der Weg zu seinem Ziel soll in möglichst wenigen Fällen fehlschlagen. Durch die Verwendung von Ziel*räumen* statt Zielkonfigurationen kann die Effizienz der lokalen Planungsstrategie gesteigert werden, wie in den Kapiteln 6.6.2 und 6.6.3 dargelegt wird.

Ein Zielraum ZR ist eine Teilmenge eines Konfigurationsraums der Dimension N. Die zu einem Zielraum gehörenden Konfigurationen nehmen in jeder Dimension genau einen vorgegebenen Wert an oder umfassen die gesamte Wertemenge des Konfigurationsraums in dieser Dimension.

Formal stellt sich ein Zielraum folgendermaßen dar:

$$ZR(sel, q^{zr}) = \{q \quad | \quad \forall i \in \{1..N\} : (sel(i) = 0 \Rightarrow q_i \in \mathbf{W}_i) \wedge (sel(i) = 1 \Rightarrow q_i = q_i^{zr})\}$$

$\mathbf{W}_i$: Wertebereich des Konfigurationsraums in der i-ten Dimension
q_i : der Wert, den die Konfiguration q in der i-ten Dimension annimmt.
$sel(i)$: selektiert die frei bleibenden Dimensionen des Zielraums
$sel(i) = 0$: Freiheitsgrad i bleibt frei,
$sel(i) = 1$: Freiheitsgrad i ist durch q_i^{zr} auf einen konstanten Wert festgelegt.
q^{zr} : den Zielraum aufspannende Konfiguration, bestimmt die Werte der festgelegten Freiheitsgrade des Zielraums

Beispiele für derartige Unterräume sind Konfigurationen, ein (ansonsten uneingeschränkter) Unterraum zu einer bestimmten formalen Zeit oder eine Position zu einer bestimmten formalen Zeit ohne Berücksichtigung der Orientierung. Die Verwendung solcher Zielräume stellt eine allgemeinere Lösung als Zielkonfigurationen dar, es kann eine gemeinsame Strategie für Aufgaben mit Zielräumen verschiedener Dimension verwendet werden, da Zielkonfigurationen genauso als Zielraum (der Dimension 0) angegeben werden können, wie das Erreichen einer beliebigen Konfiguration zur formalen Zeit 1 (Zielraum der Dimension 6).

6.5 Allgemeine Planungsstrategie

Betrachtet man die bereits weit fortgeschrittenen Arbeiten auf dem Gebiet der Bahnplanung für einzelne Manipulatoren in höherdimensionalen Konfigurationsräumen, so kann man daraus ersehen, daß die sehr guten Planungsleistungen der fortschrittlichsten Planer [CH92, BL90, Gla91b] aus einem Zusammenspiel aus lokalen und globalen Planungsstrategien resultieren.

Lokale Planungsstrategien sind Strategien, die lediglich eine beschränkte Umgebung der aktuellen Konfiguration betrachten. Globale Planungsstrategien sind Strategien, die Information über den Konfigurationsraum in seiner Gesamtheit (etwa Repräsentationen der Topologie des Konfigurationsraums als Zusammenhangsgraph) betrachten. Lokale Planungsstrategien sind – wegen der streng lokalen Sicht auf den Konfigurationsraum – sehr effizient, können also sehr schnell einen Weg zwischen zwei Konfigurationen finden. Allerdings werden in manchen Situationen keine Wege gefunden, obwohl welche existieren. Ein lokaler Planer soll also eine Lösung in vielen Fällen finden, allerdings soll nicht versucht werden, die Menge der von einem lokalen Planer lösbaren Aufgaben durch einen *erheblichen* Mehraufwand beim Planungsvorgang zu erweitern. Hierzu werden globale Planungsstrategien konzipiert, die durch gezielten Einsatz des lokalen Planers eine grobe globale Betrachtung des Konfigurationsraums ermöglichen, insbesondere ein Steckenbleiben des lokalen Planers in lokalen Minima verhindern.

In den folgenden Abschnitten werden nun – unter Berücksichtigung der in Kap. 6.3, S. 89 beschriebenen Konfigurationsraumdarstellung – zuerst Strategien für einen lokalen Planer und dann globale Strategien beschrieben, die die Grundlage eines leistungsfähigen Planers für kooperierende Manipulatoren in realistischen Umgebungen bilden.

6.6 Lokale Planungsstrategien

Aus Effizienzgründen verwendet ein lokaler Planer nur *lokale* Information über den Konfigurationsraum. Er hat die Aufgabe, eine Startkonfiguration q_{start} mit einer Zielkonfiguration q_{ziel} durch einen Weg $\tau(q_{start}, q_{ziel})$ zu verbinden.

Eine sehr effiziente Methode, einen auf eine Zielkonfiguration hinführenden Weg zu pla-

nen, ist die Verwendung des direkten Wegs zwischen aktueller Start- und Zielkonfiguration und dessen schrittweiser Test auf Kollision. Diese Methode wird bei Kavraki [KL94a] und Glavina [Gla91b] verwendet. Bei Glavina wird durch die Einführung einer effizienten Strategie zum Umgehen von Hindernissen (*Gleiten*, s. Kap. 6.6.2) die Erfolgsquote des lokalen Planers erhöht.

Zum einen besitzen die hier auftretenden Konfigurationsräume einen sehr hohen Füllungsgrad (s. Kap. 6.3), der freie Konfigurationsraum ist sehr dünn. Die Verwendung des direkten Weges führt daher sehr häufig zu einem Versagen des lokalen Planers. Desweiteren entstehen die Konfigurationsraumhindernisse bei den hier betrachteten Aufgaben durch die Überlagerung mehrerer Kollisionsarten (Hinderniskollisionen plus Gelenküberschreitungen für jedes einzelne Gelenk). In solchen Bereichen des Konfigurationsraums ist ein Entlanggleiten wie bei Glavina nicht sehr erfolgversprechend: In diesen Gegenden liegen "... Stellen mit kantigen Rillen vor, nämlich da, wo Einzelhindernisse im Konfigurationsraum aneinanderstoßen. Der garantierte Ausweg aus einer solchen Situation ist nicht mehr mit einfachen Mitteln möglich." ([Gla91b], S. 32)

Eine weitere Methode, Wege zu planen, die immer näher auf ein Ziel zuführen, stellt die Verwendung einer Kombination aus einem zum Ziel hin anziehenden und von Konfigurationsraumhindernissen abstoßenden Potentialen dar. Durch die Verwendung eines abstoßenden Potentials ist die Einbeziehung von Information aus einer größeren Umgebung der aktuellen Konfiguration möglich, als dies bei obigen Verfahren der Fall ist. Dort wird lediglich die aktuelle Konfiguration bzw. eine minimale Umgebung (Gleiten) betrachtet. Durch die Einbeziehung von Information über eine größere Umgebung kann der Weg von Konfigurationsraumhindernissen entfernt gehalten werden. Dies verringert die Gefahr des Steckenbleibens in kleineren, lokalen Sackgassen und erhöht somit die Leistungsfähigkeit des lokalen Planers. Eine vollständige Vermeidung von Sackgassen ist – wegen der lokalen Sicht auf den Konfigurationsraum – nicht möglich.

Der Einsatz von Potentialen ist aber nur sinnvoll, wenn diese effizient berechenbar sind, um die erforderliche hohe Effizienz des lokalen Planers nicht zu gefährden. Für eine von beiden Kollisionsarten abstoßende Potentialfunktion sind Methoden zur Berechnung der Entfernung der beteiligten Manipulatoren von Gelenkgrenzen und der Entfernung des aus den Manipulatoren und den manipulierten Werkstücken zusammengesetzten Objekts vom nächstgelegenen Hindernis erforderlich.

Die Entfernung von Gelenkgrenzen ist über einfache Differenzbildung zwischen aktueller Gelenkstellung und Gelenkgrenzen effizient berechenbar. Zum anderen sind gerade die Konfigurationsraumhindernisse, die sich durch Gelenkgrenzen ergeben, durch die Überlagerung der Grenzüberschreitungen der einzelnen Gelenke mit vielen Kanten durchzogen. Desweiteren liefert Gleiten an Gelenkgrenzen Lösungen, bei denen in weiten Bereichen sich Gelenke an ihren Gelenkgrenzen befinden. In der Realität sind solche Lösungen unerwünscht, da etwa die sensorielle Korrektur von Ungenauigkeiten zur Überschreitung von Gelenkwinkelgrenzen führen kann bzw. Regelungsalgorithmen nicht mehr in allen Freiheitsgraden korrekt arbeiten können. Aus diesen Gründen wird für den lokalen Planer ein von

Gelenkgrenzen abstoßendes Potential verwendet.

Anders verhält es sich mit der Berechnung der Entfernung von Hindernissen. Algorithmen zur Berechnung von *Kollisionen* liegen auf derzeit üblichen Rechnern (ca. 100 SPECint92) bei 1 - 10 ms bei einfachen Szenarien, bei realistischen Szenarien liegen diese Zeiten eine Größenordnung schlechter [Gla91b, Den92]. Effiziente Algorithmen zur Berechnung des *Abstands* (zB. [GJK87]) liegen in ihren Zeiten noch wesentlich schlechter. Geschlossene Verfahren zur Kollisionsberechnung [Sch89] liegen ebenfalls jenseits oben erwähnter Zeiten. Da eine Hauptanforderung an lokale Planer die *effiziente* Generierung von Wegen ist, und eine Verbesserung der Planungsleistung daher nicht durch einen erheblichen Mehraufwand erkauft werden sollte, verbietet sich die Einbeziehung der Entfernung zu Konfigurationsraumhindernissen in ein abstoßendes Potential, da bei der Bestimmung *einer* Nachfolgekonfiguration die Potentialfunktion *mehrfach* auszuwerten ist. Stattdessen wird die von Glavina [Gla91b] vorgestellte und als *Entlanggleiten an Hindernissen* bezeichnete Strategie zur Behandlung von Kollisionen mit Arbeitsraumhindernissen verwendet.

Der Ablauf der lokalen Planungsstrategie ist im Flußdiagramm Abb. 29 dargestellt, die einzelnen Bestandteile (Schrittweise Konstruktion von Wegen, Gleitschritte, Schrittbestimmung mit Potential) werden in den nachfolgenden Kapiteln dargestellt.

6.6.1 Schrittweise Konstruktion von kollisionsfreien Wegen

Anstatt nun zu versuchen, vom lokalen Planer zu planende Wege in einer geschlossenen mathematischen Form zu ermitteln oder darzustellen, wird eine Diskretisierung vorgenommen. Dabei wird ein Weg durch eine Folge von kollisionsfreien Konfigurationen

$$(q_i)_{(i=1\ldots N)}, \quad \text{mit } q_0 = q_{start} \text{ und } q_N = q_{ziel}$$

approximiert. Dadurch wird das Wegfindungsproblem soweit vereinfacht, daß für eine endliche Menge von Konfigurationen nur jeweils ein binärer Test auf Kollisionsfreiheit durchgeführt werden muß.

Ein Schritt s von einer gegebenen Konfiguration q zu einer Nachfolgekonfiguration q_s kann nun folgendermaßen dargestellt werden:

$$s = (\Delta T, \Delta t)$$

wobei ΔT eine Transformation (dargestellt etwa durch eine homogene Matrix) und Δt einen Fortschritt in der *formalen Zeit* beschreibt.

Ein Schritt überführt nun eine Konfiguration $q = (T, t)$ in eine Nachfolgekonfiguration q_s, wobei gilt:

$$q_s = s \circ q = (\Delta T \circ T, \Delta t + t)$$

An diese Folge wird die Zusatzforderung gestellt, daß alle Konfigurationen, die auf der geradlinigen Verbindung zweier aufeinanderfolgender Konfigurationen liegen, ebenfalls kollisionsfrei sind. Methoden zur Lösung dieses Problems für einen einzelnen Manipulator werden

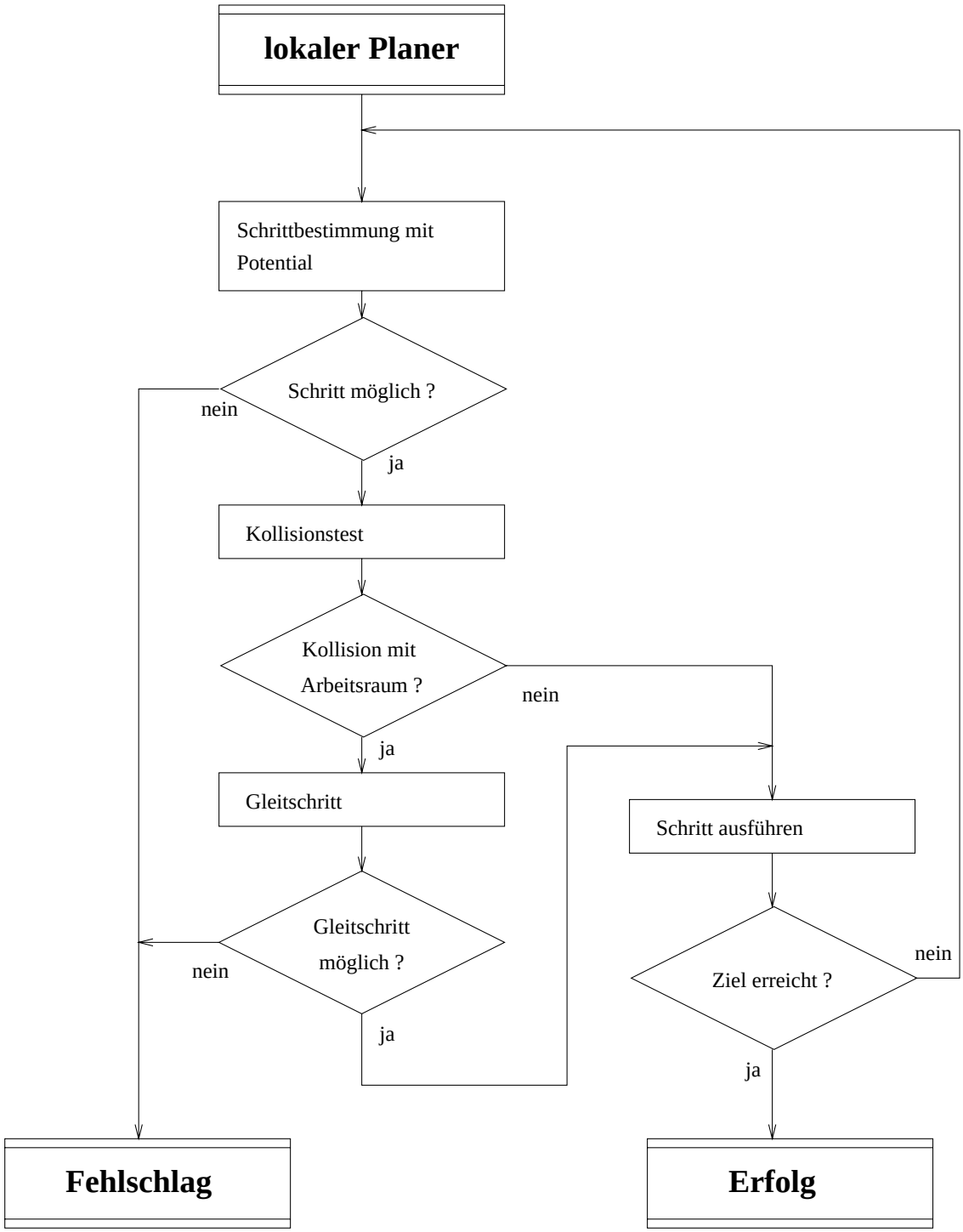


Abbildung 29: Flußdiagramm der lokalen Planungsstrategie

in mehreren Arbeiten [Gla91b, LP81, LP83b, LP87] beschrieben. Dabei werden Arbeitsraumhindernisse um eine Schutzschicht einer vorgegebenen Dicke d_{schutz} aufgebläht. Bewegt

sich nun kein Massenpunkt mp eines Manipulators um mehr als d_{schutz} beim Übergang von einer kollisionsfreien Konfiguration q zu einer Nachfolgekonfiguration q_s, so ist auch sichergestellt, daß alle Konfigurationen auf der geradlinigen Verbindung zwischen den beiden Konfigurationen q und q_s kollisionsfrei sind. Plant man nun eine Folge von kollisionsfreien Konfigurationen (q_i), sodaß sich kein Massenpunkt des Manipulators zwischen zwei aufeinanderfolgenden Konfigurationen weiter als d_{schutz} bewegt, so ist der Weg, der durch die geradlinige Verbindung der Konfigurationen q_i beschrieben wird, ebenfalls kollisionsfrei.

Für die in dieser Arbeit betrachteten Aufgaben mit zwei Manipulatoren wird dieser Ansatz wie folgt modifiziert: Die Manipulatoren und die manipulierten Werkstücke werden um eine Schutzschicht der Dicke d_{schutz} aufgebläht, nicht aber die Umgebungshindernisse. Kollisionstests werden mit den derart modifizierten Objekten durchgeführt. Bei der Konstruktion eines Schrittes ist dann auf die Einhaltung der folgenden beiden Schutzschichtbedingungen zu achten:

S1 Kein Massepunkt des *bewegten Objekts* darf bei einem Schritt die maximal zulässige Entfernung d_{schutz} überschreiten

S2 Kein Massepunkt der *beteiligten Manipulatoren* darf bei einem Schritt die maximal zulässige Entfernung d_{schutz} überschreiten

Da beide Manipulatoren von einer Schutzschicht umgeben sind, ist Bedingung S2 auch bei einer Relativbewegung der Manipulatoren um jeweils d_{schutz} aufeinander zu ausreichend.

Um nun die Einhaltung der Bedingungen S1 und S2 sicherzustellen, wird jeder Schritt folgendermaßen getestet:

Für (S1) ist für jeden Massepunkt des Objekts nachzuweisen, daß die bei dem betrachteten Schritt durchzuführende Transformation die maximal zulässige Entfernung nicht überschreitet. Dieser Test kann dadurch effizient durchgeführt werden, daß man einen Hüllquader um das Objekt bildet und nur die zurückgelegte Entfernung für die Eckpunkte e_i des Hüllquaders betrachtet, da kein im Hüllquader liegender Punkt weiter bewegt wird als dessen Eckpunkte. Auf die e_i wird dazu die Transformation des auszuführenden Schritts angewendet

$$e_i^s = \Delta T \circ e_i$$

und die maximale dabei durchgeführte Bewegung

$$d_{max}^{obj} = max\{\|e_i^s - e_i\| \quad | \quad i = 1 \ldots 8\}$$

bestimmt, die kleiner als d_{schutz} sein muß.

Um zu testen, ob kein Massepunkt der Manipulatoren bei dem betrachteten Schritt s die maximale Entfernung überschreitet (S2), werden die Lagen der Effektorkoordinatensysteme nach Ausführung des Schritts s ermittelt. Aus diesen werden die Gelenkstellungen der

Manipulatoren bestimmt. Aus den Gelenkparametern ließe sich nun – analog zur Bestimmung der maximalen Objektbewegung – für jedes Glied der Manipulatoren die Bewegung der Eckpunkte eines zugeordneten Hüllquaders berechnen. Eine effizientere Methode wird in [Gla91b] vorgeschlagen. Mittels dieses Verfahrens kann lediglich aus den Gelenkstellungsdifferenzen vor und nach der Schrittausführung die maximale Bewegung j_{max} eines Massepunkts des Manipulators abgeschätzt werden. Dabei wird jedem Gelenk i ein sogenannter *maximaler Wirkungsradius* l_i zugeordnet. Dies ist ein Faktor, der aus einer inkrementellen Bewegung des Gelenks Δj_i die maximale Bewegung eines Massenpunkts d^{mp}_{max} des Manipulators beschreibt, wenn alle anderen Gelenke unbewegt bleiben. Im schlimmsten Fall ergibt sich die maximale Bewegung eines Massenpunkts durch die Summe der durch die einzelnen Gelenkbewegungen hervorgerufenen Bewegungen:

$$d^{mp}_{max} = \sum_{i=1}^{6} l_i |\Delta j_i|$$

Die maximale Bewegung eines Massenpunkts des Objekts oder der Manipulatoren d_{max} ergibt sich dann zu:

$$d_{max} = max\{d^{obj}_{max}, d^{mp_1}_{max}, d^{mp_2}_{max}\}$$

Wobei $d^{mp_1}_{max}$ und $d^{mp_2}_{max}$ die maximale Bewegung eines Massenpunkts des jeweiligen Manipulators beschreiben.

6.6.2 Entlanggleiten an Hindernissen

Zur Behandlung von Kollisionen mit Arbeitsraumhindernissen wird versucht, an diesen entlangzugleiten. Die Gleitrichtung an Hindernisoberflächen ist a priori unbestimmt. Um aber die zielgerichtete Suche durch einen lokalen Planer beizubehalten und ein determiniertes Verhalten der Gleitstrategie zu realisieren, werden nur Gleitbewegungen ausgeführt, die näher zum Ziel hinführen. Ist dies nicht mehr möglich, so bricht der lokale Planer ab und gibt die Kontrolle an den globalen Planer zurück. Diese Heuristik wurde von Glavina [Gla90, Gla91b] für Gelenkwinkelkonfigurationsräume beschrieben. Sie versucht, im Falle einer Kollision mit einem Hindernis einen Schritt *zur Seite* (im folgenden als *Gleitschritt* bezeichnet) von der letzten kollisionsfreien Konfiguration aus zu finden. Hierzu wird nach dem Schmidtschen Orthogonalisierungsverfahren [Fis80] eine Orthonormalbasis der affinen Hyperebene H (Abb. 30) konstruiert, die auf dem Vektor senkrecht steht, der den zur Kollision führenden Schritt darstellt. Die Basisvektoren und ihre Negate stellen dann mögliche Seitrichtungen dar.

Durch Gleitschritte soll zum einen eine möglichst starke Seitbewegung erreicht werden, um mit jedem Gleitschritt möglichst weit an der Hindernisoberfläche entlangzugleiten. Gleichzeitig muß bei einem Gleitschritt die Entfernung zum Ziel abnehmen.

Um einen Gleitschritt zu bestimmen, wird durch ein Bisektionsverfahren des in die Kollision führenden Schritts s' eine kollisionsfreie Konfiguration q_m (s. Abb. 30) bestimmt, die

– im Rahmen der Auflösung des Bisektionsverfahrens – möglichst nah an der Hindernisoberfläche liegt und somit einen maximalen Fortschritt in Richtung Ziel gewährleistet. Von diesem Abzweigungspunkt wird dann eine Seitwärtsbewegung x durchgeführt, die orthogonal zur Richtung des letzten Schritt ist.

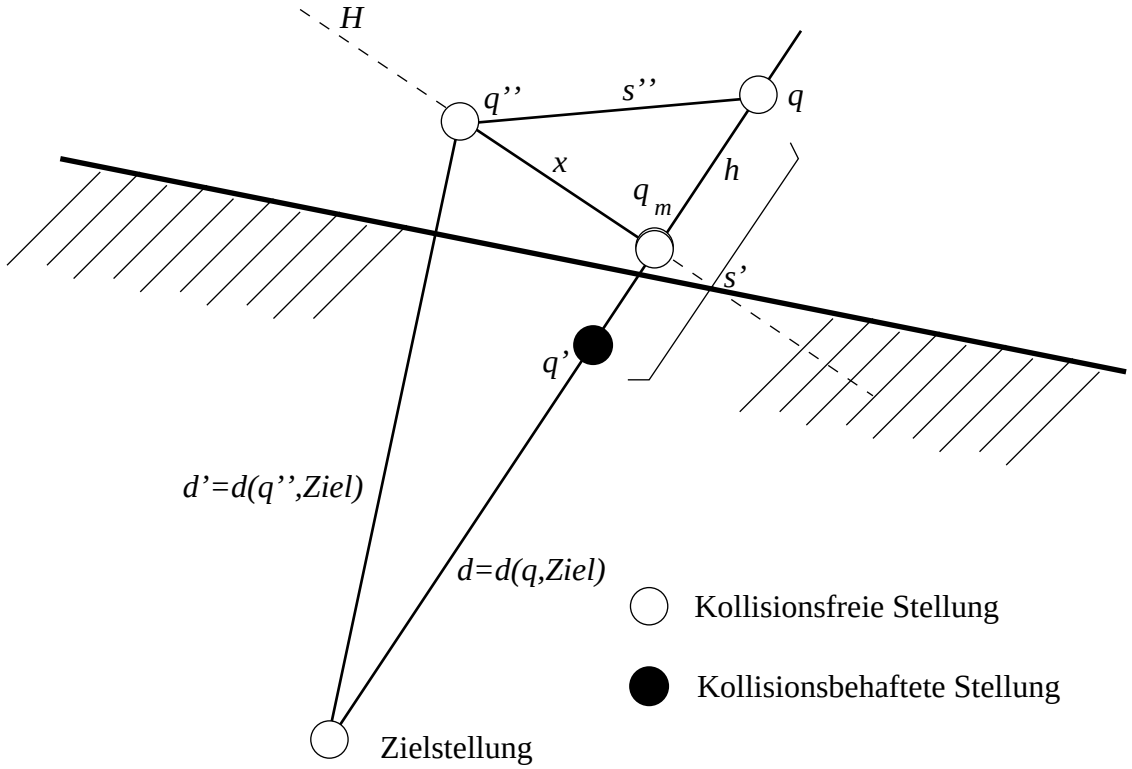


Abbildung 30: Konstruktion eines Gleitschritts

Der daraus resultierende Seitschritt s'' zur Konfiguration q'' muß dann zum einen näher zum Ziel führen ($d' < d$), zum anderen muß s'' so gewählt werden, daß sich kein Massepunkt des Objekts oder der Manipulatoren um mehr als d_{max} bewegt (s. *Schutzschichtbedingungen*, Kap. 6.6.1).

Während in dem durch Gelenkwinkel gebildeten Konfigurationsraum bei Glavina alle Dimensionen des Konfigurationsraums gleichartig sind und dadurch eine einfache Bildung einer Orthonormalbasis zu einem gegebenen Vektor ermöglichen, sind die Dimensionen (Translation, Rotation, formale Zeit) der hier verwendeten Konfigurationsraumdarstellung unterschiedlich in ihrer Bedeutung. Daher ist die Bildung einer gemeinsamen Orthogonalbasis nicht sinnvoll. Desweiteren kann sowohl für die translatorischen Freiheitsgrade als auch für die rotatorischen Freiheitsgrade getrennt die für die Einhaltung der Schutzschichtbedingung für das Objekt maximale Länge des Seitschritts x (Abb. 30) rechnerisch bestimmt werden.

Daher werden sowohl für die translatorische als auch für die rotatorische Komponente des Schritts je eine Orthogonalbasis erzeugt. Diese sind trivialerweise (da alle anderen

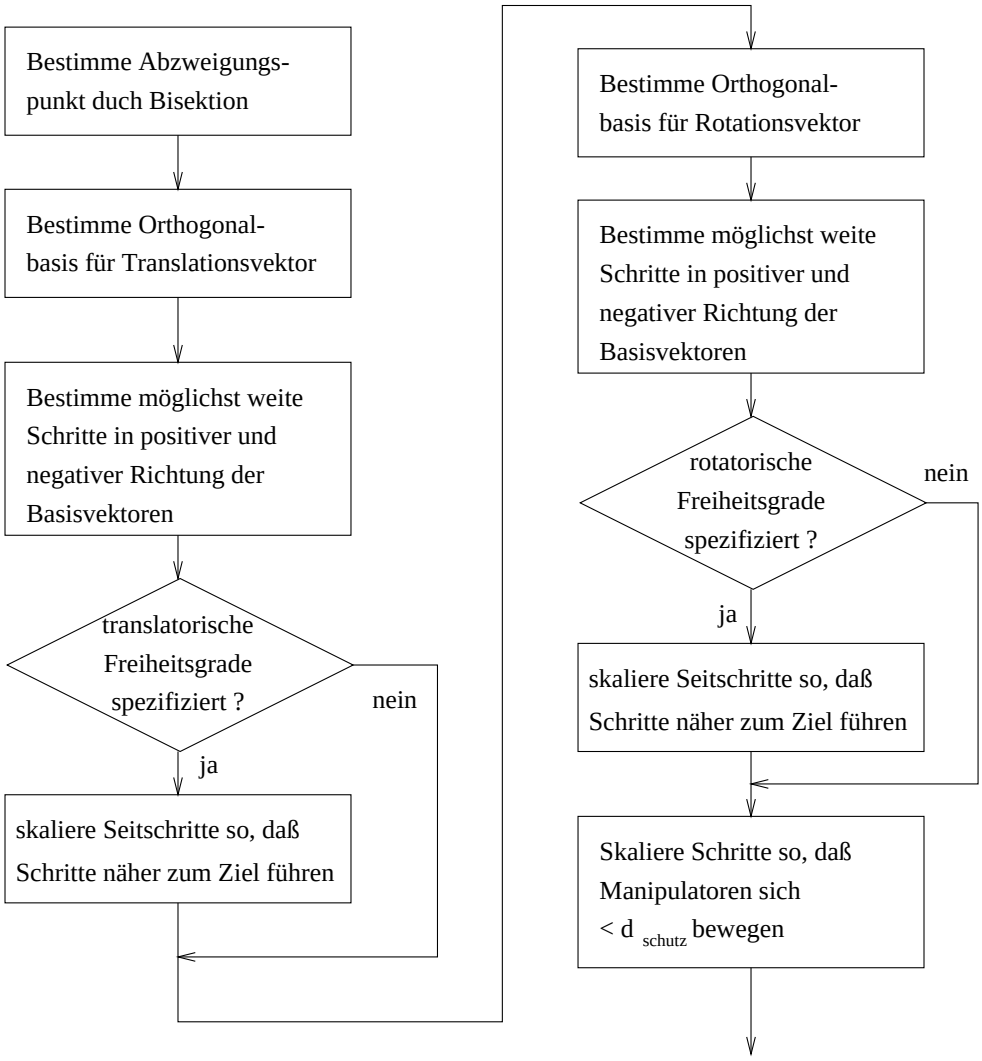


Abbildung 31: Ablauf der Generierung von Gleitschritten

Komponenten null sind) orthogonal zum gesamten Schritt. Für diese und ihre Negate wird jeweils ein maximaler Schritt bestimmt.

Für den translatorischen Fall ergeben sich folgende Bedingungen [Rie93]:

1. Die Schrittweite s'' von q nach q'' muß kleiner sein als die maximale Schrittweite d_{schutz}.

2. Die translatorische Distanz zum Ziel nach Ausführung des Gleitschrittes $d' = d(q'', Ziel)$ muß kleiner sein als die Distanz $d = d(q, Ziel)$ vorher. Wählt man $x = \sqrt{2dh - h^2}$, so ist die Seitschrittlänge in der Hinsicht maximal, daß immer noch eine Annäherung an das Ziel erfolgt. Durch den Fortschritt in der formalen Zeit, der durch den Teilschritt h erfolgte, ergibt sich insgesamt eine Annäherung an das Ziel.

Der rotatorische Fall wird analog behandelt. Das Vorgehen ist in Anhang D dargestellt.

Anschließend an die Bestimmung eines Gleitschritts ist dieser entsprechend der Schutzschichtbedingung für die Manipulatoren zu skalieren. Da die Rückwärtsrechnung, also die Abbildung, die einen Schritt auf eine Gelenkwinkeländerung abbildet, nichtlinear ist, reicht eine einfache lineare Skalierung

$$s_{i+1} = f \cdot s_i$$

mit dem Faktor

$$f = \frac{d_{schutz}}{d_{max}}$$

nicht aus. Anstatt eines Bisektions- oder Newtonverfahrens wird stattdessen – um mit einer möglichst geringen Anzahl der relativ teuren Rückwärtsrechnungen auszukommen – eine Skalierung des Schritts s_i mit einem verringerten Faktor

$$f = k \cdot \frac{d_{schutz}}{d_{max}} \qquad k < 1 \qquad (\text{zB.} \quad k = 0.9)$$

und eine anschließende Bestimmung von d_{max} durchgeführt, bis die Bedingung $d_{max} < d_{schutz}$ erfüllt ist. Dies ist in der Regel nach 2 bis 3 Iterationen der Fall, sodaß diese Methode effizienter ist als die Alternative einer Bisektionssuche.

Durch den Anteil h aus dem in die Kollision führenden Schritt ist zudem sichergestellt, daß im Seitschritt s'' ein Fortschritt in der formalen Zeit erfolgt.

Im Anschluß an die Generierung der Gleitschritte wird der erste kollisionsfreie Gleitschritt ausgeführt. Ist keiner der Gleitschritte kollisionsfrei, so bricht der lokale Planer an dieser Stelle erfolglos ab.

Ein Problem von Gleitschritten ist, daß seitliche Ausweichbewegungen zu einer Vergrößerung des Abstandes in einigen Dimensionen führt. Für Gleitschritte wird jedoch eine Verringerung des Abstands zum Ziel gefordert. Verwendet man statt Ziel*konfigurationen* Ziel*räume*, so bringt eine Seitbewegung in diesen Dimensionen bei Gleitschritten keine Vergrößerung des minimalen Abstands vom Zielraum mit sich. Dies führt dazu, daß Zielräume bei Gleitschritten wesentlich häufiger erreicht werden als einzelne Zielkonfigurationen.

In Abb. 32 ist dieser Sachverhalt für den zweidimensionalen Fall veranschaulicht. Während der Gleitvorgang in Richtung der Zielkonfiguration dadurch steckenbleibt, daß sich beim Gleiten der Abstand von der Zielkonfiguration in x-Richtung vergrößert, wird der Zielraum, dessen x-Dimension offen bleibt, erreicht, da die x-Dimension nicht in die Abstandsberechnung einbezogen wird, und eine Seitbewegung nicht zu einer Vergrößerung des Abstands zum Zielraum führt.

Ein anschauliches Beispiel stellt die Aufgabe TRANSPORT (Abb. 44, S. 124) dar. Eine kritische Stelle ist das Durchfahren der Engstelle zwischen den beiden Manipulatoren von vorne nach hinten (bzw. umgekehrt). Wird als Zwischenziel lediglich eine vorgegebene Position verwendet, während die Orientierung offenbleibt, so kann beim Gleiten eine erforderliche rotatorische Ausweichbewegung erfolgen, während die Entfernung in der Position abnimmt.

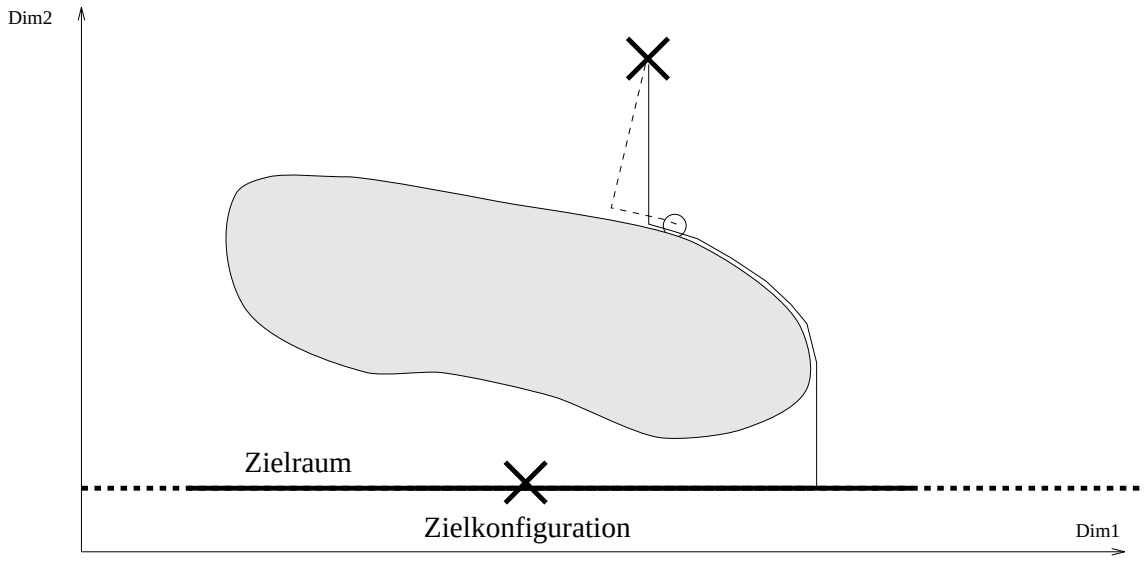


Abbildung 32: Ziel*räume* statt Zielkonfigurationen steigern den Erfolg von Gleitschritten

6.6.3 Die Potentialfunktion

Zielgerichtetes Potential Wie bereits weiter oben erwähnt wurde, sollen die während des Planungsvorgangs zu konstruierenden Wege von einer Start*konfiguration* zu einem Ziel*raum* hinführen.

Neben der Einschränkung der Effizienz bei Gleitschritten bedingen Zielkonfigurationen auch während der Potentialsuche eine starke Einschränkung für die Richtung des zu planenden Weges. Dies äußert sich in der strengen Zielvorgabe in allen Dimensionen, die durch das durch sie beschriebene Potential dem abstoßenden Potential der Konfigurationsraumhindernisse entgegenwirkt. Dieses starke Entgegenwirken des anziehenden gegenüber dem abstoßenden Potential kann ebenfalls dadurch verringert werden, daß statt einer Zielkonfiguration Zielräume verwendet werden. Da diese nicht in allen Dimensionen ein anziehendes Potential induzieren, wirkt in diesen freien Dimensionen lediglich das abstoßende Potential. Somit bleibt der Weg weiter von Konfigurationsraumhindernissen entfernt, während eine Annäherung an den Zielraum erfolgt.

Zur Realisierung des anziehenden Potentials wird folgende Potentialfunktion verwendet, die den Abstand einer gegebenen Konfiguration q zu einem Zielraum zr berechnet:

$$\mathbf{POT}_{anziehend}(q, zr) = \sqrt{\sum_{DOF} sqr\_dist_i(q, zr)}$$

$$sqr\_dist_i(q, zr) = \begin{cases} (q_i^{zr} - q_i)^2 & \text{wenn Freiheitsgrad } i \text{ selektiert} \\ 0 & \text{sonst} \end{cases}$$

q_{zr} : Konfiguration, die den Zielraum zr aufspannt
q_i : i-ter Freiheitsgrad der Konfiguration q

Abstoßendes Potential Wie bereits erwähnt, wird der freie Konfigurationsraum durch zwei Arten von Bedingungen begrenzt:

- Kollisionen zwischen Objekten, Manipulatoren und Hindernissen und
- Gelenkbegrenzungen der Manipulatoren

Da Hinderniskollisionen durch eine Gleitstrategie behandelt werden (s. Kap. 6.6.2), werden nur die Gelenkgrenzen durch ein abstoßendes Potential behandelt.

Die verwendete Potentialfunktion muß also umso größere Werte liefern, je näher sich Gelenke an ihren Gelenkgrenzen befinden. Dies wird durch folgende abstoßende, an [PK91] angelehnte Potentialfunktion geleistet:

$$\mathbf{POT}_{abstoßend}(q) = \sum_{n=1}^{2} \sum_{i=1}^{gel\_anz_n} \frac{1}{(og_i^n - J_i^n(q))(J_i^n(q) - ug_i^n)}$$

gel_anz_n : Anzahl der Gelenke des n-ten Manipulators
$J_i^n(q)$: Stellung des i-ten Gelenks des n-ten Manipulators
og_i^n : Obergrenze des i-ten Gelenks des n-ten Manipulators
ug_i^n : Untergrenze des i-ten Gelenks des n-ten Manipulators

Die Potentialfunktion Das abstoßende und das anziehende Potential werden zur Verwendung in einem lokalen Planungsalgorithmus in einer gewichteten Potentialsumme zusammengefaßt:

$$\mathbf{POT}(q, zr) = k \cdot \mathbf{POT}_{anziehend}(q, zr) + \mathbf{POT}_{abstoßend}(q)$$

Auf die Bestimmung des Gewichtungsfaktors k wird im folgenden Kapitel eingegangen. In Kap. 6.6.5 ff. wird die Berechnung von Schritten mittels dieser Potentialfunktion dargestellt.

6.6.4 Adaptive Gewichtung des anziehenden Potentials

Ein generelles Problem bei der Wegsuche mit einer Kombination aus anziehenden und abstoßenden Potentialen sind lokale Minima. Diese entstehen bei anziehenden Potentialen, die durch einfach zu berechnende Entfernungsfunktionen realisiert werden, durch von

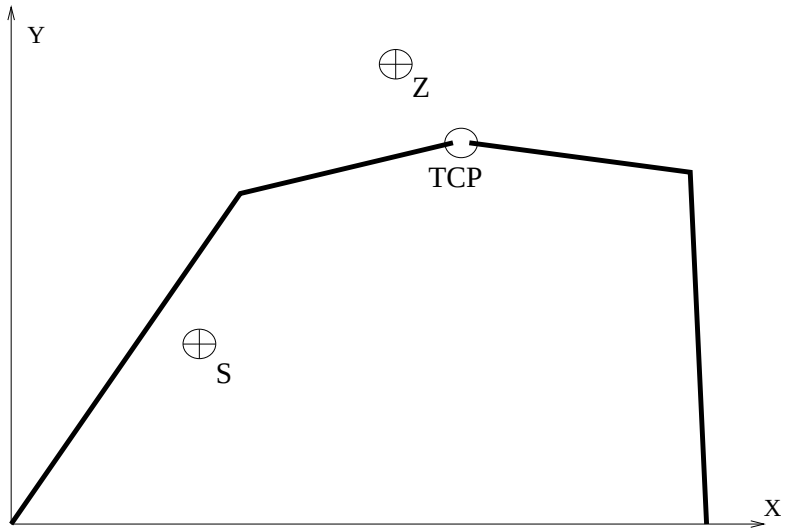


Abbildung 33: Zwei Linienroboter, Start- und Zielkonfiguration

Hindernissen gebildete Sackgassen. In Kombinationen von anziehenden und abstoßenden Potentialen ergeben sich durch die Überlagerung zweier Potentiale zusätzlich zu überwindende Sattelpunkte. Dieser Sachverhalt ist in Abb. 34 verdeutlicht. Dort wird die Kombination des von den Gelenkgrenzen abstoßenden Potentials der beiden Linienroboter in Abb. 33 und des zur Zielkonfiguration **Z** hin anziehenden Potentials dargestellt. Zur Darstellung des Konfigurationsraums werden dabei die kartesischen Koordinaten der im Punkt **TCP** zusammenfallenden Effektorspitzen der beiden Roboter verwendet. Eine von einem Gradientenabstiegsverfahren ermittelte, in der Startkonfiguration **S** beginnende Folge von Konfigurationen endet in einem lokalen Minimum vor dem zu durchfahrenden Sattel, hinter dem die Zielkonfiguration liegt. Es ist daher wünschenswert, bei der Planung Potentiale zu verwenden, die frei von lokalen Minima sind oder nur sehr wenige lokale Minima besitzen, damit der lokale Planer in möglichst vielen Fällen zu einer Lösung kommt, ohne in einer Sackgasse steckenzubleiben.

Die Konstruktion von Abstandspotentialen ohne lokale Minima ist nur durch eine globale Betrachtung des Konfigurationsraums möglich (etwa Wellenfront-Potentiale), dies scheidet wegen des zu hohen Aufwands aus (s. Kap. 3.5.1). Stattdessen wird im folgenden eine Methode beschrieben, die diese Nachteile umgeht und

- die Kombination von zielorientierten und abstoßenden Potentialen erlaubt
- durch Sattelpunkte entstehende lokale Minima überwindet und
- nur geringen Zusatzaufwand erfordert.

Von entscheidender Bedeutung für die Bestimmung einer günstigen Potentialfunktion ist die Gewichtung der beteiligten Potentiale: Überwiegt der abstoßende Anteil, so können

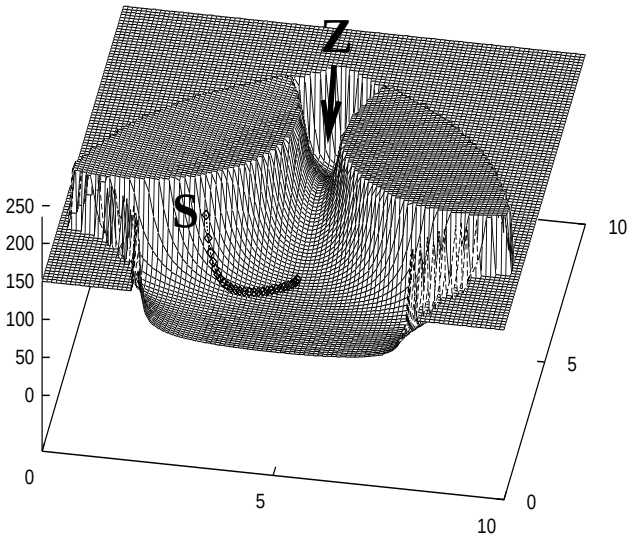


Abbildung 34: Steckenbleiben des lokalen Planers in einem lokalen Minimum

lokale Minima dieses Potentials zu lokalen Minima des Gesamtpotentials werden (Abb. 34). Gewichtet man den anziehenden Anteil zu stark, verschwinden zwar manche lokale Minima, aber der Effekt des abstoßenden Potentials, nämlich den Weg aus ungünstigen Bereichen fernzuhalten, wird zunichtegemacht, der Weg verläuft über weite Strecken in Gebieten mit hohem abstoßendem Potential (Abb. 35). Desweiteren existieren keine lokalen Kriterien, um a priori zu entscheiden, wie die Gewichtung durchzuführen ist. In Abb. 37 etwa, in der die Zielstellung (vorne) bereits über ein hohes abstoßendes Potential verfügt, liegt ein starkes lokales Minimum in direkter Nachbarschaft. Um es zu überwinden, ist ein extrem starkes, anziehendes Potential erforderlich. Seine Lage und Tiefe, die zur Ermittlung eines Gewichtungsfaktors erforderlich wäre, kann allerdings nur durch eine globale Betrachtung des Potentialfelds ermittelt werden.

Das Prinzip des hier vorgestellten Potentialsuchverfahrens besteht in einer adaptiven Gewichtung des zielorientierten (anziehenden) Anteils der verwendeten Potentialfunktion

$$\mathbf{POT}(q, zr) = \mathbf{AG} \cdot \mathbf{POT}_{anziehend}(q, zr) + \mathbf{POT}_{abstoßend}(q)$$

mit einem adaptiven Gewichtungsfaktor **AG**.

Solange ein Potentialabstiegsverfahren Schritte erzeugt, die sehr direkt zum Ziel hinführen, bedeutet dies, daß das zielorientierte Potential genügend Einfluß hat. Daher kann für den

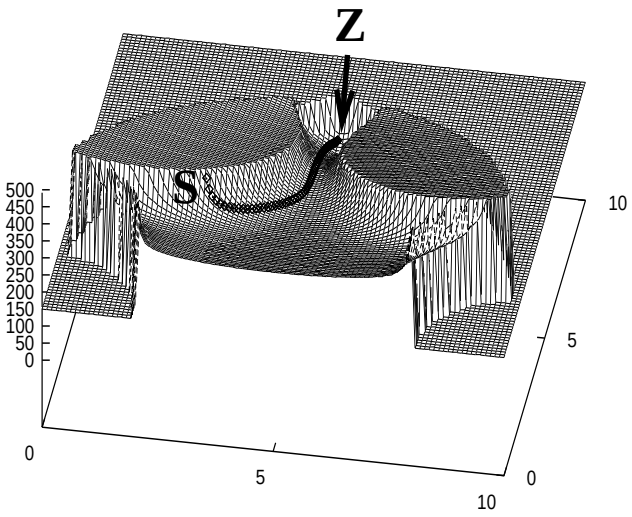


Abbildung 35: Eine ausreichende Gewichtung des anziehenden Potentials ermöglicht ein Überwinden eines Sattelpunkts

nächsten Schritt sein Einfluß verringert werden. Erhält man allerdings einen Schritt, der nicht direkt genug auf das Ziel zuführt, so ist der Einfluß des anziehenden Potentials zu gering und muß daher erhöht werden. Durch diese Methode ist zum einen gewährleistet, daß Sattelpunkte überwunden werden und zum anderen, daß der Einfluß des abstoßenden Potentials stets möglichst stark bleibt (s. Abb. 36,37).

Als Maß für die Adaption des Gewichtungsfaktors wird der Fortschritt f eines Schritts s aus einer aktuellen Konfiguration q in Richtung Ziel zr verwendet, dargestellt durch den Quotienten

$$f = \frac{\text{Verringerung des Zielabstands}}{\text{Schrittweite}} = \frac{d(s \circ q, zr) - d(q, zr)}{d(s \circ q, q)}$$

Dabei bezeichnet $s \circ q$ die Konfiguration, die sich durch Ausführung des Schritts s aus der Konfiguration q heraus ergibt.

Die Adaption des Gewichtungsfaktors erfolgt durch Skalierung mit n bzw. $\frac{1}{n}$, wenn eine untere Schranke u ($u \approx 0.6$) für f unterschritten bzw. eine obere Schranke o ($o \approx 0.9$) überschritten wird. Dabei ist der Wert von n unkritisch. Zu niedrige Werte führen zu einer Verlangsamung des lokalen Planers, da in manchen Fällen erst mehrere Skalierungsschritte

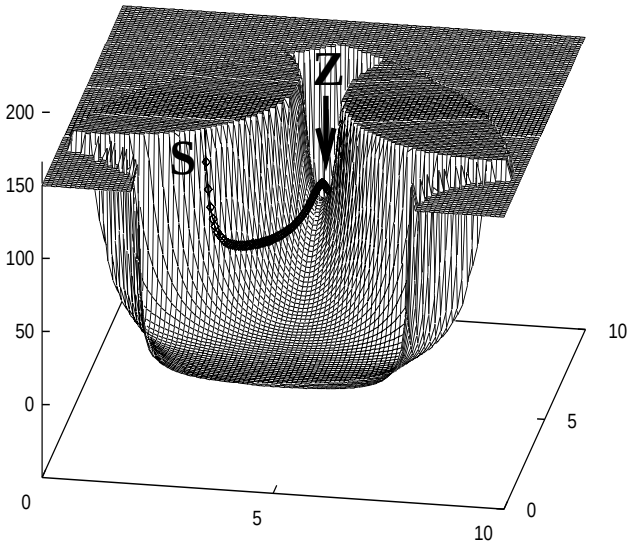


Abbildung 36: Adaptive Gewichtung, ein Weg vom Start vorne (hohes abstoßendes Potential) zum Ziel hinten (niedriges abstoßendes Potential) wurde gefunden

ausgeführt werden, bis ein zum Ziel hinführender Schritt gefunden wird. Zu hohe Werte dämpfen den Einfluß des abstoßenden Potentials sehr schnell. In praktischen Versuchen hat sich ($n \approx 2$) als guter Wert ergeben.

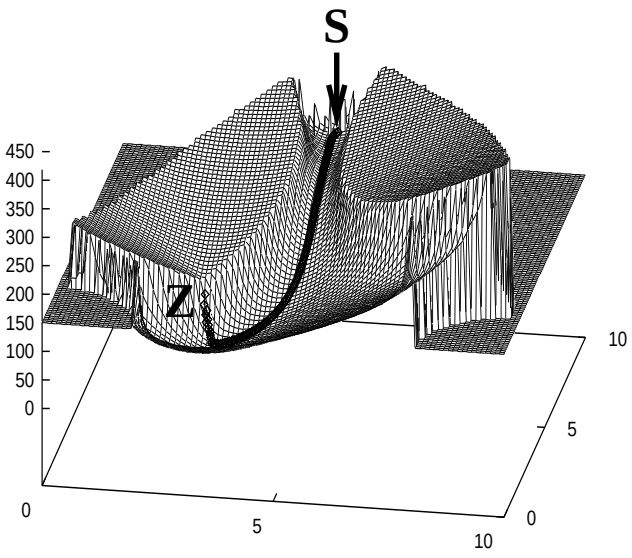


Abbildung 37: Adaptive Gewichtung, Start hinten (niedriges abstoßendes Potential), Ziel vorne (hohes abstoßendes Potential)

Algorithmus zur Bestimmung eines Schrittes:

DO
  berechne besten Schritt
  berechne Zielfortschritt f
  IF $f < u$ (untere Schranke)
  THEN BEGIN $\mathbf{AG} = n \cdot \mathbf{AG}$
    IF AG > Maximum
    THEN Abbruch (sonst etwa zu nahe an Gelenkgrenzen)
  END
  IF $f > o$ (obere Schranke)
  THEN $\mathbf{AG} = \frac{1}{n} \cdot \mathbf{AG}$
  IF näher am Ziel **AND** Schrittqualität ok
  THEN Schritt gefunden
WHILE Schrittqualität ok **AND** nicht am Ziel
IF am Ziel
  RETURN am Ziel
ELSE
  RETURN Abbruch

Der Algorithmus erhöht solange das anziehende Potential, bis ein Schritt gefunden wird, der näher zum Ziel hinführt. Überschreitet der Gewichtungsfaktor dabei eine obere Schranke, so wird abgebrochen, da keine weitere Annäherung an das Ziel mit ausreichender Berücksichtigung des abstoßenden Potentials mehr möglich ist. Der lokale Planer ist somit in einer Sackgasse (ggf. an einem zu steilen Sattelpunkt bzw. in der Nähe eines Zielpunkts, der extrem nahe an Gelenkgrenzen liegt) gelandet. Es ist dann Aufgabe der globalen Planungsstrategie, durch weitere Aufrufe des lokalen Planers die Planung voranzutreiben. Wird ein Schritt gefunden, so wird – entsprechend Abb. 29, S. 96 – dieser auf Kollision getestet und gegebenenfalls mit der Generierung eines Gleitschritts fortgefahren.

6.6.5 Schrittgenerierung mit der Potentialfunktion

Von einer aktuellen (Start- oder Zwischen-)konfiguration q aus ist es nun die Aufgabe des lokalen Planers, einen Schritt s_{opt} zu finden, sodaß gilt:

1. $mb(s_{opt}) < d_{schutz}$
2. $\forall s, s \neq s_{opt} \wedge mb(s) < d_{schutz} : POT(s_{opt} \circ q) < POT(s \circ q) \wedge mb(s_{opt}) < d_{schutz}$

d_{schutz} : Schutzschichtdicke
$mb(s)$: maximale Bewegung eines Massenpunkts des bewegten Objekts bei der Ausführung des Schritts s

Die zweite Bedingung fordert die Suche nach einem globalen Optimum einer Potentialfunktion in einem bestimmten Bereich. Die Suche nach einem optimalen Schritt steht allerdings im Widerspruch zur Effizienz. Da die Potentialfunktion zur Führung der lokalen Suche dient, steht die Suche nach einem *optimalen* Schritt nicht im Vordergrund. Daher wird die zweite Bedingung abgeschwächt: Ein vom Planer gefundener Schritt sollte *möglichst* gut sein, auf die Suche nach dem *besten* Schritt wird verzichtet, stattdessen wird ein Kompromiß aus effizienter Schritterzeugung und Optimierung der Gütefunktion eingegangen.

In diesem Abschnitt wird eine Methode zur effizienten, näherungsweisen Berechnung eines bezüglich der im vorausgehenden Abschnitt beschriebenen Potentialfunktion

$$\mathbf{POT} : (q, zr) \rightarrow \mathbb{R}$$

lokal optimalen Schritts vorgestellt. Es gilt, ein Minimum dieses Potentials in einem siebendimensionalen Raum innerhalb eines beschränkten Gebiets möglichst effizient zu berechnen. Die Gebietsgrenzen sind durch die maximale Bewegung eines Massenpunkts des bewegten Objekts bei einem Schritt vorgegeben.

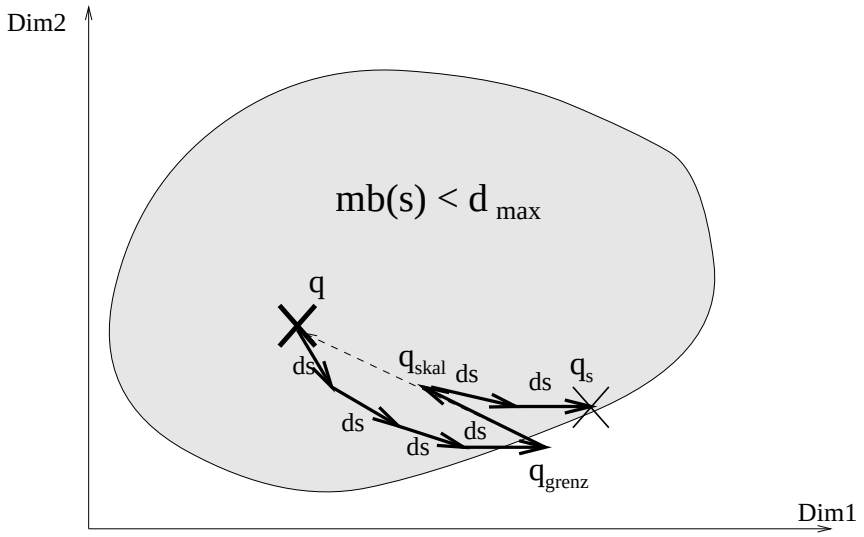


Abbildung 38: Gradientenabstieg bei begrenztem Potentialfeld

Eine Standardmethode für diese Problemstellung stellt ein Gradientenabstiegsverfahren dar. Ausgehend von der aktuellen Konfiguration wird iterativ der Gradient der Potentialfunktion berechnet (Aufwand: 7 Auswertungen der Potentialfunktion) und ein Suchschritt in negativer Gradientenrichtung ausgeführt, bis der Gradient null wird bzw. bedingt durch die Diskretisierung, die zu einem Einschwingen der Gradientensuche um ein lokales Minimum herum führen kann, sich der Wert der Potentialfunktion nicht mehr verbessert.

Allerdings stößt man bei dieser Methode auf zwei Probleme:

1. Gradientensuchmethoden streben auf ein lokales Minimum in negativer Gradientenrichtung zu und sind nicht in der Lage, globale Minima ausfindig zu machen.

2. Die Gradientenabstiegsmethode versagt an den nichtstetigen, sich durch die maximale Schrittlänge ergebenden Bereichsgrenzen, da dort der Gradient nicht mehr berechnet werden kann. Durch das anziehende Potential wird das Gesamtpotential meist für möglichst weit in Zielrichtung führende Schritte, also gerade an den Bereichsgrenzen, lokale Minima aufweisen.

Problem 1) kann durch eine Gradientenabstiegsmethode nicht umgangen werden. Das Problem 2) wird durch eine zusätzliche Heuristik in der Gradientenabstiegsmethode gelöst: Von einer aktuellen Konfiguration q (Abb. 38) werden inkrementelle Schritte ds entlang des negierten Gradienten der Potentialfunktion berechnet. Der Startwert für diese Iteration ist der leere Schritt. Nach jedem inkrementellen Schritt wird getestet, ob der Schritt von c bis zur gerade erreichten Konfiguration noch erlaubt ist, er also die Schutzschichtbedingung $mb(s) < d_{schutz}$ erfüllt. Ist dies der Fall, wird der nächste inkrementelle Schritt berechnet. Führt der Schritt zu einer Überschreitung der maximalen Schrittlänge (q_{grenz} in Abb. 38), so befindet man sich außerhalb des für die Schrittberechnung zulässigen Bereichs, ein wei-

terer Gradientenabstieg ist nicht mehr möglich. Da aber die Konfiguration, an dem der maximal zulässige Bereich überschritten wurde, im allgemeinen kein lokales Optimum darstellt, das Potential an seinem Rand aber nicht differenzierbar ist, wird eine Konfiguration q_{skal} *in der Nähe* der Konfiguration, an der der zulässige Bereich verlassen wurde, gesucht, um von dort aus den Gradientenabstieg fortzusetzen. Dies wird durch eine Skalierung des Schritts, der zum Verlassen des zulässigen Bereichs führte, mit einem Faktor $n < 1$ (etwa $n = 0.8$) erreicht. In Abb. 38 wird dies verdeutlicht: Der Schritt zur Konfiguration q_{grenz}, der zur Überschreitung der Grenze führte, wird skaliert. Dies führt zu der Konfiguration q_{skal} in der Nähe von q_{grenz}, aber innerhalb der Grenzen. Von hier aus wird das Gradientenabstiegsverfahren fortgesetzt. Dieses Verfahren terminiert, wenn das Potential beim Erreichen der Grenze keine Verbesserung gegenüber dem Potential der letzten die Grenze erreichenden Konfiguration darstellt oder bereits vor Erreichen der Grenze ein lokales Minimum gefunden wurde.

6.6.6 Behandlung von Wert- und Bereichseinschränkungen

In der hier behandelten Teilklasse von Aufgabenspezifikationen (s. Kap. 6.2) können geometrische Constraints spezifiziert sein. Durch die prototypisch realisierten Planungsstrategien werden als geometrische Constraints Wert- und Bereichseinschränkungen für einzelne Freiheitsgrade der Form

DOF_SELEKT VERGL_OP REAL_KONST

behandelt. In Kap. 6.9 wird die Erweiterung auf eine allgemeinere Klasse von Constraints skizziert.

Im folgenden wird beschrieben, wie die Umsetzung dieser beiden Klassen von in der Aufgabenspezifikation vorgegebenen Einschränkungen in der prototypischen Realisierung der Planungsverfahren erfolgt.

Einschränkungen auf einen Wert Wird ein Freiheitsgrad durch ein Constraint der Form

DOF_SELEKT = REAL_KONST

bestimmten Wert eingeschränkt (zB. $\mathbf{TZ} = 1000$, $\mathbf{RX} = 0$), so ist vorausgesetzt, daß derselbe Wert in der Start- und – falls vorhanden – Zielkonfiguration spezifiziert ist. Andernfalls ist die Aufgabe widersprüchlich spezifiziert. Der entsprechende Freiheitsgrad wird auf diesen Wert festgelegt und für die Schrittgenerierung durch Potentialauswertung sowie für die Generierung von Gleitschritten nicht mehr verändert.

Bereichseinschränkungen Geometrische Constraints der Form

DOF_SELEKT < REAL_KONST bzw. DOF_SELEKT > REAL_KONST

können durch zwei Methoden behandelt werden: Eine allgemeine Methode stellt ihre Einbeziehung in das abstoßende Potential dar, das umso größere Werte annimmt, je weiter man sich den Bereichsgrenzen nähert. Diese Methode wird in Kap. 6.9 beschrieben. Eine weitere, bei der Schrittgenerierung effizient einbeziehbare Methode, wird in der prototypischen Realisierung verwendet: Spezifizierte Bereichseinschränkungen werden für jeden generierten Schritt vor dem Kollisionstest überprüft. Führt ein Schritt in eine Konfiguration, die den spezifizierten Bereich überschreitet, so wird der Schritt als kollisionsbehaftet zurückgewiesen.

6.7 Globale Planungsstrategien

Ein lokalen Planungsstrategien inhärentes Problem ist das Steckenbleiben in Sackgassen. Eine solche kann lediglich dadurch *erkannt* werden, daß keine Nachfolgekonfiguration gefunden wird, die näher auf das Ziel zuführt. Es gibt aber keine effiziente, lokale Methode, um aus einer Sackgasse zu entweichen. Es sind nun globale Strategien erforderlich, die es dem Planer ermöglichen, aus Sackgassen zu entweichen.

6.7.1 Entweichen aus Sackgassen

Bei Planungsstrategien, die eine Potentialfeldsuche auf einem explizit dargestellten Konfigurationsraum realisieren, gibt es Ansätze, die dieses Problem lokal durch eine *Best-First-Suche* lösen (zB. der *best first planner* von Barraquand und Latombe [Lat91]). Anschaulich entspricht dies einem "Auffüllen" des lokalen Minimums. Dabei werden in einer gerasterten Konfigurationsraumdarstellung ausgehend von dem Punkt, an dem das lokale Minimum entdeckt wurde, alle Punkte der Umgebung untersucht, bis von einem dieser Punkte aus ein Nachbar gefunden wird, der ein niedrigeres Potential besitzt.

Durch eine *Best-first-Suche* sind also Teilvolumina des Konfigurationsraums zu untersuchen. In Konfigurationsräumen mit höheren Dimensionen ist aber bereits zur Untersuchung von Bereichen mit kleiner Ausdehnung ein hoher Aufwand erforderlich. So enthält bereits ein Hyperwürfel der Kantenlänge 10 Rastereinheiten in einem siebendimensionalen Konfigurationsraum 10^7 Rastereinheiten. Eine *Best-first-Suche* kommt aus Effizienzgründen somit als globale Strategie nicht in Betracht.

Eine in höherdimensionalen Konfigurationsräumen anwendbare Strategie zum Entweichen aus lokalen Minima sind *random walks*, zufällige, Brownschen Molekularbewegungen gleichende Bewegungen. In dem Planungsverfahren *randomized path planner* (RPP) [BL91] wird die Erzeugung von *random walks* in Kombination mit einem Gradientenabstiegsverfahren verwendet. Bleibt das Gradientenabstiegsverfahren in einem lokalen Minimum stecken, wird ein *random walk*, gefolgt von einer Gradientenabstiegsbewegung durchgeführt. Ist das dadurch erreichte lokale Minimum niedriger, so wird die Suche von hier aus weitergeführt. Andernfalls wird für eine vorgegebene Anzahl von Iterationsschritten versucht, durch diese Kombination ein niedrigeres lokales Minimum zu erreichen. Schlägt diese Ite-

ration fehl, so wird durch Backtracking ein früheres lokales Minimum ausgewählt, an dem die Suche fortgesetzt wird. Die Suche endet, sobald ein erreichtes lokales Minimum das Ziel ist.

Alternativ zu diesem Versuch, aus einem lokalen Minimum *lokal* zu entweichen, können Strategien verwendet werden, die versuchen, lokale Minima dadurch zu vermeiden, daß die Suche auf andere Bereiche des Konfigurationsraums verlagert wird. Eine solche Strategie wird für den Fall, daß ein Weg zwischen zwei Punkten im Konfigurationsraum gefunden werden soll, von Glavina [Gla91b] vorgeschlagen (s. Kap. 3.5.1).

Wie bereits in Kap. 6.4 dargelegt, ermöglicht die Verwendung von Zwischenzielräumen die Planung für Ziele verschiedener Dimension und verbessert zudem die Planungsleistung des lokalen Planers. Daher ist die hier zu verwendende globale Planungsstrategie für die Verwendung von Zwischenzielräumen auszulegen.

6.7.2 Generierung von Zwischenzielräumen

Beim Entwurf einer globalen Strategie, die die Verwendung von Zwischenzielräumen für einen lokalen Planer ermöglicht, ist zu berücksichtigen, daß ein lokaler Planer nicht in der Lage ist, zwei Zielräume miteinander zu verbinden. Stattdessen kann nur eine Start*konfiguration* mit einem Ziel*raum* verbunden werden. Gelingt die Verbindung, so entsteht im Zielraum eine dem Zielraum zugeordnete *Ankunftskonfiguration*. Es muß also in einem Zwischenzielraum bereits eine Konfiguration existieren, bevor dieser Zwischenzielraum mit einem weiteren Zwischenzielraum verbunden werden kann. Dies kann die Start- oder Zielkonfiguration bzw. eine Ankunftskonfiguration sein.

Die globale Strategie [Fis94a, Fis94b] basiert auf dem von Glavina vorgeschlagenen und in Kap. 6.7.1 beschriebenen Verfahren zur Konstruktion eines Graphen, der aus miteinander durch kollisionsfreie Wege verbindbaren Konfigurationen besteht. Diese Konfigurationen sind die Start- und Zielkonfiguration sowie zufällig gewählte Zwischenzielkonfigurationen. Um Zwischenzielräume anstelle von Zwischenzielkonfigurationen verwenden zu können, wird dieses Verfahren wie folgt erweitert.

Bei der Generierung von zufällig gewählten Zwischenzielräumen ist zu berücksichtigen, daß diese kollisionsfreie Konfigurationen enthalten müssen. Andernfalls sind sie nie erreichbar, was zu vermeidbaren Fehlversuchen des lokalen Planers führt. Eine Methode zur Generierung von Zwischenzielräumen besteht nun darin, zufällig eine kollisionsfreie Konfiguration zu bestimmen und aus dieser dann – durch Freigabe der entsprechenden Freiheitsgrade – einen Zwischenzielraum zu erzeugen. Zur Bestimmung einer zufälligen, kollisionsfreien Konfiguration werden zufällige Konfigurationen generiert und auf Kollision getestet, bis eine kollisionsfreie Konfiguration gefunden wurde. Vor allem bei Aufgaben mit vorgegebener Zielkonfiguration ist eine Freigabe der rotatorischen Freiheitsgrade, wie sie in dem realisierten Prototypen realisiert wurde, günstig, da auf diese Weise größere Distanzen im Konfigurationsraum vom lokalen Planer sehr erfolgreich überwunden werden können.

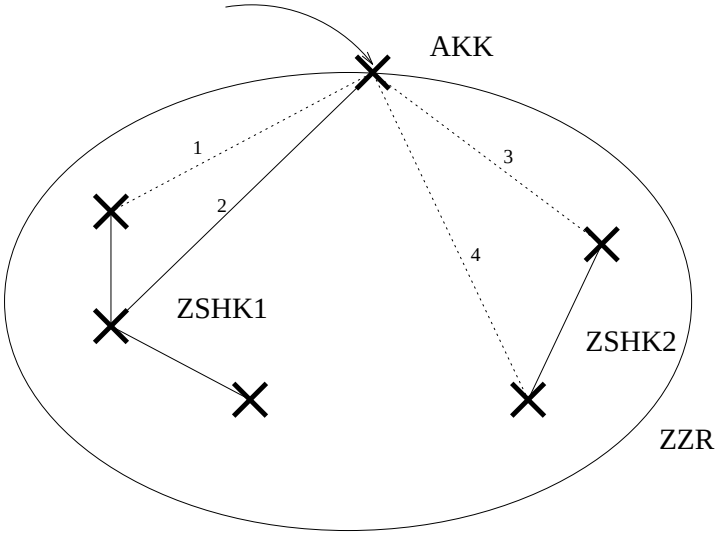


Abbildung 39: Verbindung einer Ankunftskonfiguration mit den Zusammenhangskomponenten in einem Zwischenzielraum

Da Zielräume nicht direkt miteinander verbunden werden können, sondern jeweils nur für eine Konfiguration mit einem Zielraum erfolgen kann, wird zur Verbindung zweier Zielräume die dem zweiten Zielraum nächstgelegene (Start-, Ziel- oder Ankunfts-) Konfiguration des ersten Zielraums ausgewählt und von dieser aus eine Verbindung zu erreichen versucht. Durch die globale Planungsstrategie wird ebenfalls die Verbindung in umgekehrter Richtung versucht. Eine stärkere Konzentration auf die Verbindung zweier Zielräume würde zuviel Aufwand für die lokale Planung bedeuten. Wegen der erforderlichen Balance zwischen lokaler und globaler Planungsstrategie ist dieser Aufwand nicht sinnvoll.

Die Verbindung einer Konfiguration mit einem Zielraum wird in Abb. 39 verdeutlicht. Wird ein Zielraum vom lokalen Planer erreicht, so entsteht eine Ankunftskonfiguration (AKK in Abb. 39), die dem Zielraum zugeordnet wird. Während der Planung entstehen dadurch i.a. mehrere Konfigurationen in einem Zielraum. Da das Erreichen eines Zielraums noch nicht die Erreichbarkeit aller Konfigurationen in diesem bedeutet, wird nach dem Entstehen einer neuen Ankunftskonfiguration versucht, diese mit den bereits existierenden Konfigurationen des Zielraums zu verbinden, um den Graphen aus freien Wegen zu verbessern. Da dies nicht immer möglich ist, können in einem Zielraum mehrere Zusammenhangskomponenten (ZSHK1 und ZSHK2 in Abb. 39) von verbindbaren Konfigurationen entstehen. Da die Konfigurationen einer Zusammenhangskomponente ohnehin miteinander verbindbar sind, wird versucht, die jeweils nächstgelegene Ankunftskonfiguration einer Zusammenhangskomponente zu erreichen, bis eine Verbindung erzielt (s. Abb 39, 2) bzw. alle Konfigurationen einer Zusammenhangskomponente zu erreichen versucht wurden (s. Abb 39, 3+4).

Die Suche ist beendet, wenn Start und Ziel in derselben Zusammenhangskomponente des Graphen liegen, also durch eine Folge $(q_i)_{(i=1..N)}$ von miteinander verbundenen Konfigurationen verbunden sind. Durch die zufällige Generierung von Zwischenzielen mit ebenfalls

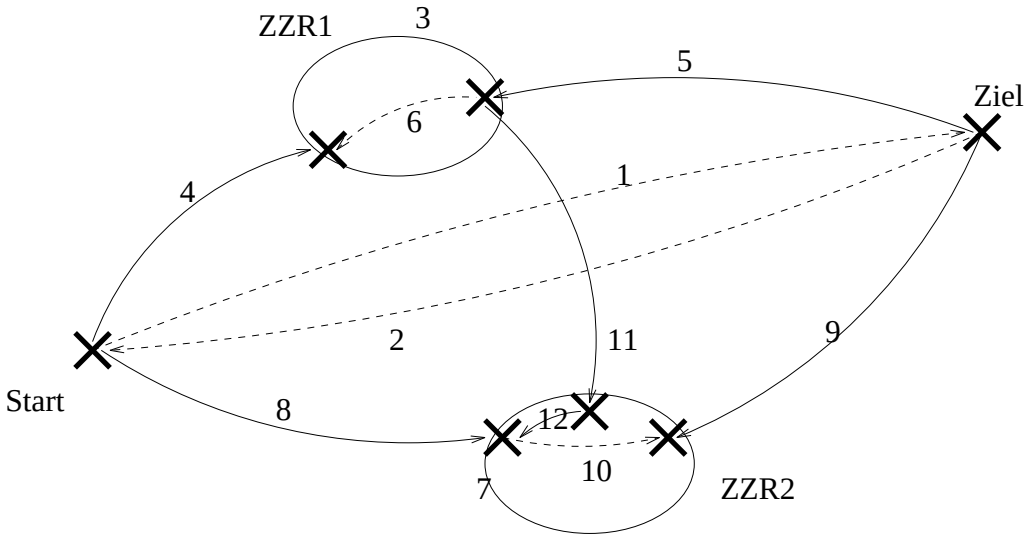


Abbildung 40: Zwischenzielräume, Beispiel Transport

zufällig gewählter formaler Zeit können Wege entstehen, die Abschnitte enthalten, die sich in der formalen Zeit rückwärts bewegen. Da ein Rückwärtslaufen der Zeit verboten ist, wird zusätzlich gefordert, daß obige Folge zeitmonoton sein muß, um einen gültigen Weg darzustellen, d.h.

$$\forall i = 1..N-1 : t(q_i) \leq t(q_{i+1}) \qquad t(q) : \text{formale Zeit einer Konfiguration}$$

Glavina merkt in [Gla91b] an, daß bei praktisch relevanten Aufgaben nur sehr wenige Zwischenziele zur Lösung der Aufgabe erforderlich sind. Der Mittelwert bei den dort betrachteten Aufgaben liegt bei 1.9 bis 6.1 Zwischenzielen, der Maximalwert bei 23. Diese Zahlen liegen bei den hier betrachteten Aufgaben noch niedriger (s. Kap. 6.8). Dennoch existieren – meist akademische – Aufgaben, die eine wesentlich höhere Anzahl von Zwischenzielen erfordern. Durch die in [Gla91b] vorgeschlagene Methode, jedes neue Zwischenziel mit allen existierenden Zwischenzielen zu verbinden, erhält man eine Komplexität $O(n^2)$ in der Anzahl der Zwischenziele. In einem existierenden Graphen aus freien Wegen ist der Gewinn an Information über die Topologie des freien Konfigurationsraums gering, wenn ein weit entfernter Knoten direkt mit einem Zwischenziel verbunden werden kann, der auch über die Verbindung mit einem nahegelegenen Knoten und einen Weg über mehrere Knoten im Graphen erreicht werden kann. Daher wird zum Aufbau des Graphen von Zwischenzielen ein neues Zwischenziel lediglich mit den N (N ca. 5 - 10) nächstgelegenen Zwischenzielräumen verbunden.

An einem Beispiel wird diese globale Strategie durch die Beschreibung eines Planungslaufs verdeutlicht (s. Abb. 40). Es wird (1) versucht, einen Weg vom Start zum Ziel zu finden. Als nächstes wird versucht, (2) vom Ziel zum Start zu gelangen. Schlägt beides fehl, wird (3) ein Zwischenzielraum erzeugt und (4) vom Start und (5) vom Ziel aus zu

erreichen versucht. Gelingt beides, wird (6) versucht, die beiden Ankunftskonfigurationen im Zwischenzielraum zu verbinden. Schlägt dies fehl, so wird ein weiterer Zwischenzielraum generiert (7). Dieser wird von Start (8) und Ziel (9) zu erreichen versucht. Bei der zweiten Ankunft im Zwischenzielraum wird die Ankunftskonfiguration mit der ersten Ankunftsposition zu verbinden versucht (10). Schlägt dies ebenfalls fehl, so wird der Zwischenzielraum von der nächstgelegenen Konfiguration im ersten Zwischenzielraum zu erreichen versucht (11). Gelingt dies, wird die neue Ankunftsposition mit den nächstgelegenen Konfigurationen der (bisher aus jeweils einer Konfiguration bestehenden) Zusammenhangskomponenten von Ankunftskonfigurationen zu verbinden versucht (12). Gelingt dies, so existiert ein Weg (8 - 12 - 11 - 5) zwischen Start und Ziel, womit das Ziel der Planungsaufgabe erreicht ist.

Durch diese globale Strategie wird somit unter Verwendung eines lokalen Planers nach und nach ein immer dichter werdender Graph von Wegen über dem freien Konfigurationsraum erzeugt, wobei durch die Verwendung von Zwischenzielräumen dem lokalen Planer ein Höchstmaß an Freiheiten zur Erreichung seines Planungsziels eingeräumt wird.

6.7.3 Behandlung der kinematischen Zustände

Wie in Kap. 6.1.1 beschrieben, stellt der zusammengesetzte kinematische Zustand der beiden Manipulatoren eine Dimension des Konfigurationsraums dar. In der globalen Strategie ist die im folgenden beschriebene quasiparallele Betrachtung der Konfigurationsräume für je einen zusammengesetzten kinematischen Zustand realisiert. Hierzu werden vor Planungsbeginn diejenigen kinematischen Zustände ermittelt, in denen die Startkonfiguration und – im Falle ihrer Existenz – die Zielkonfiguration im freien Konfigurationsraum liegen. Nur für diese kinematischen Zustände ist eine parallele Planung durchzuführen. Für jeden dieser kinematischen Zustände wird von der globalen Planungsstrategie ein eigener Graph verwaltet. Um die Planung für alle kinematischen Zustände quasiparallel voranzutreiben, wird nach jeder Rückkehr aus dem lokalen Planer zyklisch am Graphen des jeweils nächsten kinematischen Zustands weitergearbeitet. Diese globale Strategie betrachtet die einzelnen kinematischen Zustände getrennt, ohne Übergänge zwischen ihnen. Dies ist durchaus ausreichend, da kinematische Einschränkungen meist durch Ausgleichsbewegungen des Kooperationspartners ausgeglichen werden können. Desweiteren entfällt das Durchfahren singulärer Stellungen, das für einen Wechsel zwischen kinematischen Zuständen erforderlich ist. Dies ist vor allem in der Praxis bei geschlossenen kinematischen Ketten problematisch, da in singulären Stellungen ein Manipulator nicht in allen Freiheitsgraden beweglich ist. Dies führt zu einer eingeschränkten Funktionsfähigkeit von Reglern, die bei der Behandlung geschlossener kinematischer Ketten erforderlich sind.

Obwohl der Wechsel des kinematischen Zustandes während der Aufgabenausführung problematisch ist und meist durch Ausgleichsbewegungen des Kooperationspartners umgangen werden kann, sollen dennoch zwei Methoden zur Einbeziehung von Übergängen zwischen kinematischen Zuständen in eine globale Strategie skizziert werden:

- Einbeziehung von Umgreifbewegungen in die Planung. Diese Methode wurde von

Koga [Kog95, KL92] beschrieben. Sie ist aber in vielen Fällen nicht praktikabel: Bei Bearbeitungsvorgängen ist eine Unterbrechung häufig unerwünscht. Bei Transportvorgängen unter dem Einfluß der Schwerkraft ist ein Lösen des Griffes ebenfalls häufig unerwünscht.

- Verwendung der Gelenkparameter *eines* Manipulators zur Darstellung des Konfigurationsraums, wenn ein Übergang zwischen kinematischen Zuständen gewünscht wird. Die globale Strategie ist dahingehend abzuändern, daß im lokalen Planer die in Kap. 6.1.1 (S. 86) beschriebenen Methoden zur Konfigurationsraumdarstellung gemischt verwendet werden können: Im Normalfall wird die Konfiguration des manipulierten Objekts verwendet. Soll ein Konfigurationsraumwechsel für einen der beiden Manipulatoren durchgeführt werden, so kommt eine Darstellung des Konfigurationsraums, die eine Rückwärtsrechnung für diesen Manipulator erforderlich macht, nicht in Betracht (s. Kap. 2.3.3, S. 12). Stattdessen ist eine Darstellung mit den Gelenkparametern dieses Manipulators zu verwenden.

6.8 Beispielaufgaben

In diesem Kapitel werden die Spezifikationen mehrerer Beispielaufgaben der Aufgabenklasse, die für die Bahnplanung betrachtet wird, beschrieben. Für diese werden von einem Bahnplanungsprototypen, der die vorgestellten Methoden realisiert, kollisionsfreie, synchrone Bewegungen geplant. Für jede Aufgabe wurden 40 Planungen durchgeführt. Zu jeder Aufgabe werden die durchschnittlichen Planungszeiten, die Anzahl der benötigten Zwischenziele, sowie der Anteil des freien Konfigurationsraums am gesamten Konfigurationsraum angegeben. Letzterer wurde durch Kollisionstests zufällig gewählter Konfigurationen ermittelt. Es wurden jeweils 10 Sätze zu je 50000 Kollisionstests durchgeführt und daraus Erwartungswert und Varianz für den prozentualen Anteil des freien Konfigurationsraums ermittelt. Da der Konfigurationsraum in den translatorischen Freiheitsgraden nicht beschränkt ist, wurde zur Bestimmung des Anteils des freien Konfigurationsraums ein in diesen Freiheitsgraden würfelförmiger Ausschnitt gewählt, der von beiden Manipulatoren erreicht werden kann.

Zu den Zeiten ist anzumerken, daß der überwiegende Anteil auf Kollisionstests und die Kommunikation zur geometrischen Datenbasis, in der die Kollisionstests ausgeführt werden, entfällt. Die Gesamtzeit pro Kollisionstest in dem für die Messungen verwendeten Prototypen liegt bei etwa 10 - 50 ms. Der Anteil des Zeitverbrauchs für Kollisionstests liegt bei etwa 90 %. Mittlerweile ist die Verfügbarkeit schnellerer Kollisionstests absehbar, die auch bei realistischen Umgebungen im Bereich von unter 1 ms liegen und somit eine Beschleunigung um mindestens den Faktor 10 bei den Kollisionstests mit sich bringen [Bag95].

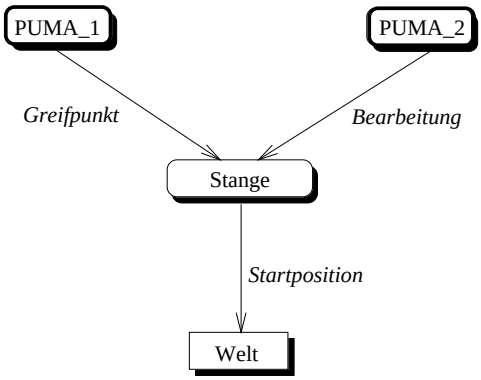
Zu jeder Aufgabe ist die entsprechende Aufgabenspezifikation angegeben. Die Umsetzung in das Eingabeformat des Planers erfolgt manuell.

6.8.1 Aufgabe: STANGE

Das Tool eines Manipulators soll von einem Ende einer langen Stange bis zum anderen Ende bewegt werden. Da die Länge der Stange den Arbeitsraum eines Manipulators überschreitet, wird die Stange von einem zweiten Manipulator kontinuierlich im Arbeitsraum des das Tool bewegenden Manipulators gehalten. In (s. Abb. 41) wird die Startstellung dargestellt.

Aufgabenspezifikation:

```
TASK: Stange
ENVIRONMENT: Zwei_Pumas
GRAPH :
```




```
RELATIONS:


Greifpunkt : (-26, 0, 0, 90, 0, 90)
Bearbeitung :
   LINEAR_INTERPOLATED :
      (26, 0, -1000, -90, 0, 90),
      (26, 0, 1000, -90, 0, 90)
   END_INTERPOLATED
Startposition :
   START : (67, -356, 1062, 86, -157, 1)
```

Der Anteil des freien Konfigurationsraums beträgt in diesem Beispiel 0.17 % (Varianz: 0.03 %). Die durchschnittliche Anzahl der generierten Zwischenzielräume beträgt 11.2 (zwischen 7 und 14), die durchschnittliche Planungszeit beträgt 235.5 s (zwischen 68 und 388 s).

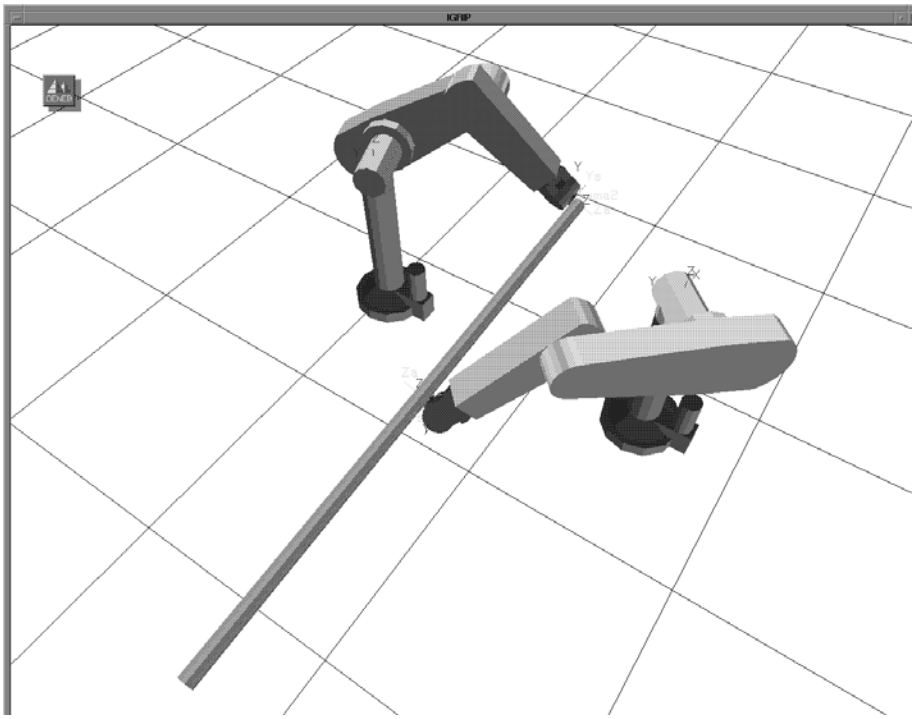


Abbildung 41: Beispiel: Aufgabe STANGE

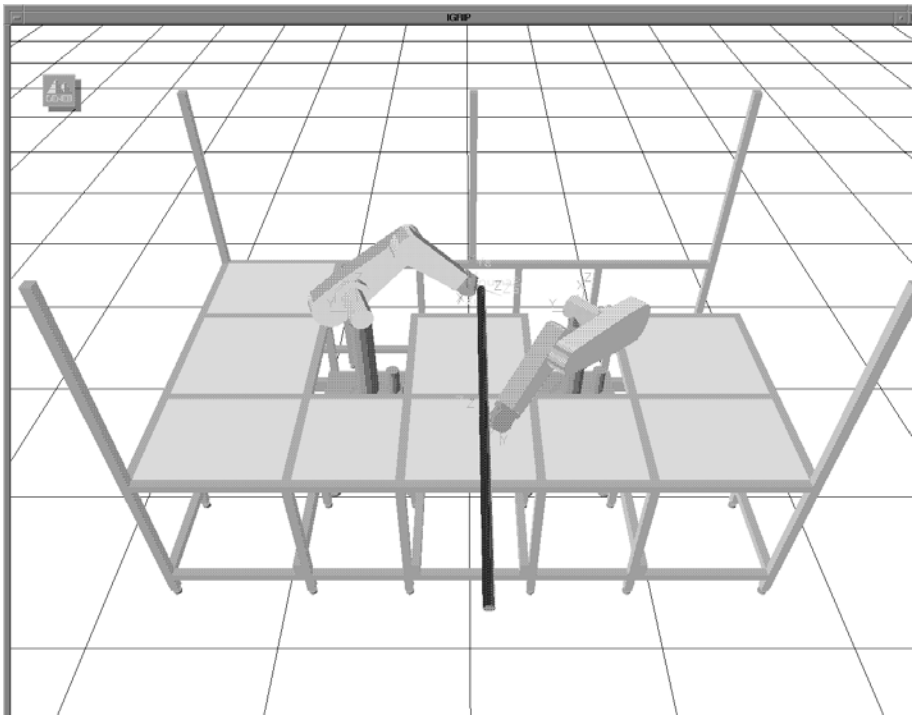


Abbildung 42: Beispiel: Aufgabe STANGE_TISCH

6.8.2 Aufgabe: STANGE_TISCH

Dies ist eine ähnliche Aufgabenstellung wie in der vorhergehenden Aufgabe. Lediglich ein Tisch wurde zur Umgebung hinzugefügt (s. Abb. 42). Die Aufgabenspezifikation entspricht somit bis auf die Umgebung der vorhergehenden Aufgabe.

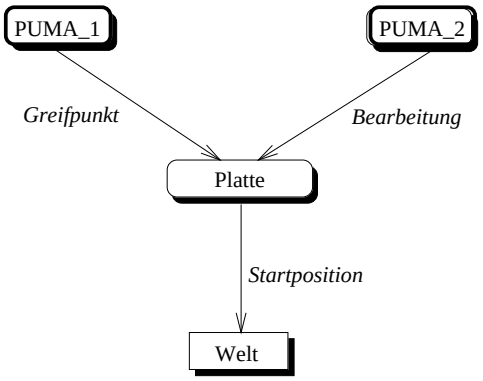
Der Anteil des freien Konfigurationsraums beträgt in diesem Beispiel 0.1 % (Varianz: 0.016 %). Die durchschnittliche Anzahl der generierten Zwischenzielräume beträgt 8,7 (zwischen 1 und 12), die durchschnittliche Planungszeit beträgt 140.1 s (zwischen 35 und 239 s). Trotz des im Vergleich zur Aufgabe STANGE verringerten Konfigurationsraums wurden für dieses Beispiel in manchen Fällen Lösungen ohne Zwischenziel gefunden, da der Tisch das 'Verhaken' der Stange zwischen Manipulatorfuß und der Schulter des Manipulators verhindert. Dies führt in der Aufgabe STANGE in jedem Fall in ein lokales Minimum.

6.8.3 Aufgabe: PLATTE

Das Tool eines Manipulators soll entlang des gesamten Rands einer Platte bewegt werden. Die Z-Achse des Tools soll dabei auf einer als Linearinterpolation spezifizierten Bahn koplanar zur Plattenfläche und senkrecht zur Kante bewegt werden. Durch die erforderlichen Orientierungsänderungen überschreitet diese Bewegung den Arbeitsraum eines Manipulators. Daher soll ein zweiter Manipulator, der die Platte in ihrer Mitte hält, diese kontinuierlich im Arbeitsraum des das Tool bewegenden Manipulators halten (s. Abb. 43).

Aufgabenspezifikation:

```
TASK: Platte
ENVIRONMENT: Zwei_Pumas
GRAPH :
```




```
RELATIONS:


Greifpunkt : (0, 0, 0, 0, 0, 0)
Bearbeitung :
   LINEAR_INTERPOLATED :
      (400,400,10,-90,0,90),
      (400,-400,10,-90,0,90),
      (400,-400,10,-90,0,0),
      (-400,-400,10,-90,0,0),
      (-400,-400,10,-90,0,-90),
      (-400,400,10,-90,0,-90),
      (-400,400,10,-90,0,-180),
      (400,400,10,-90,0,-180)
   END_INTERPOLATED
Startposition :
   START : (-98, -103, 1115, -4, -179, -52)
```

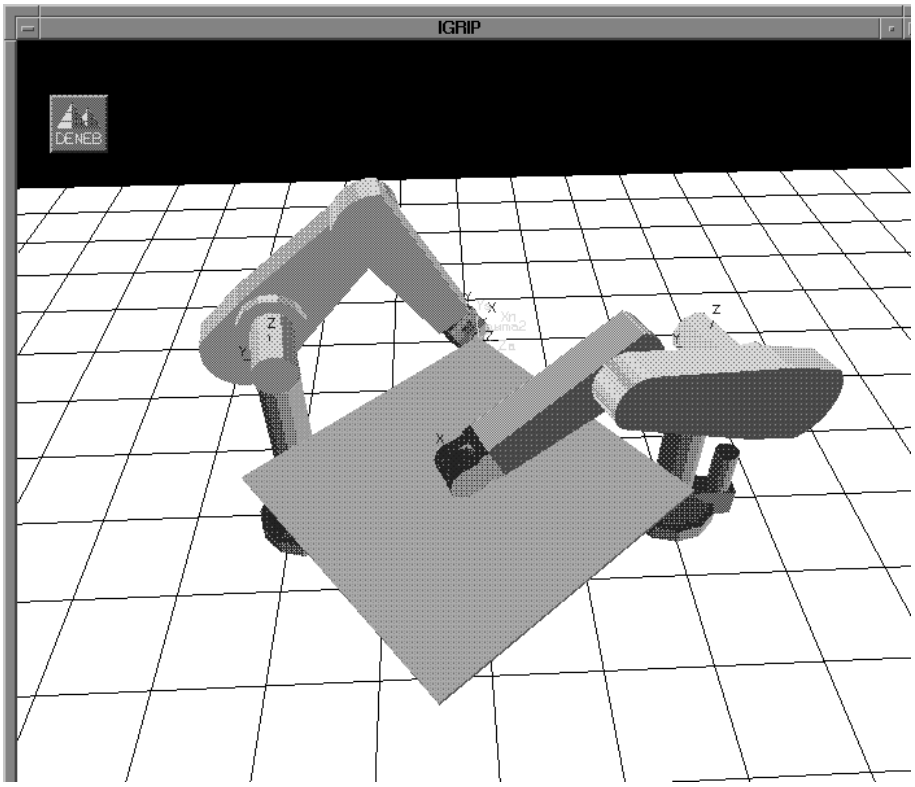


Abbildung 43: Beispiel: Aufgabe PLATTE

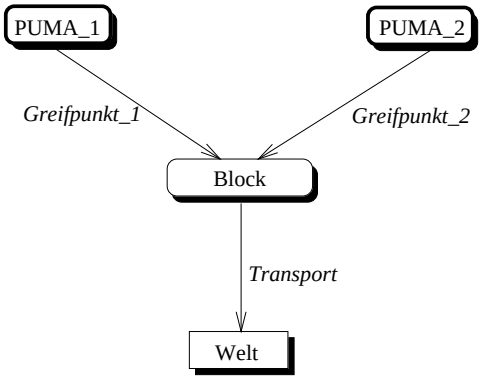
Der Anteil des freien Konfigurationsraums beträgt in diesem Beispiel 0.18 % (Varianz: 0.013 %). Die durchschnittliche Anzahl der generierten Zwischenzielräume lag bei 1.5. Die durchschnittliche Planungszeit beträgt 86.4 s (zwischen 41 und 145 s).

6.8.4 Aufgabe: TRANSPORT

Zwei Manipulatoren sollen gemeinsam einen Block aus einer Startstellung (in Abb. 44 vorne dargestellt) in eine Zielstellung (in Abb. 44 hinten transparent dargestellt) transportieren. Die Aufgabe ist zusätzlich durch geometrische Einschränkungen an die Transformation zwischen Welt und Objektkoordinatensystem spezifiziert: Das Objekt soll in einer horizontalen Position in konstanter Höhe transportiert werden. Auf dem direkten Weg befindet sich zum einen ein würfelförmiges Hindernis, zum anderen kann das Objekt nur durch eine Orientierungsänderung die Engstelle zwischen den Manipulatoren passieren. Eine vom Planer gelieferte Transportsequenz ist in Abb. 45 dargestellt.

Aufgabenspezifikation:

```
TASK: Transport
ENVIRONMENT: Zwei_Pumas_mit_Tisch
GRAPH :
```




```
RELATIONS:


Greifpunkt_1 : (320, -5, 58, 174, 1, 22)
Greifpunkt_2 : (-255, 35, 60, 174, 1, 22)
Transport :
   START : (94, -112, 1027, 0, 0, 0) AND
   GOAL : (94, 900, 1027, 0, 0, 0) AND
   TZ = 1027 AND
   RX = 0 AND
   RY = 0
```

Der Anteil des freien Konfigurationsraums beträgt in diesem Beispiel 0.14 % (Varianz: 0.016 %). Die Planung erforderte keine Generierung von Zwischenzielen, die Planungszeiten lagen zwischen 22 und 24 s, mit einem Durchschnitt von 22.97 s.

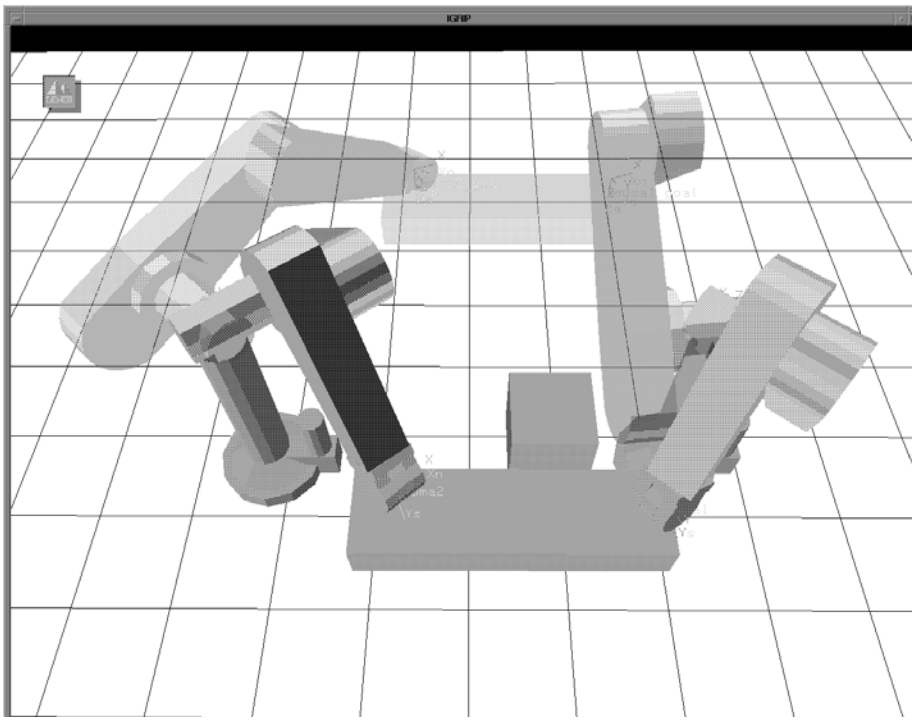


Abbildung 44: Beispiel: Aufgabe TRANSPORT

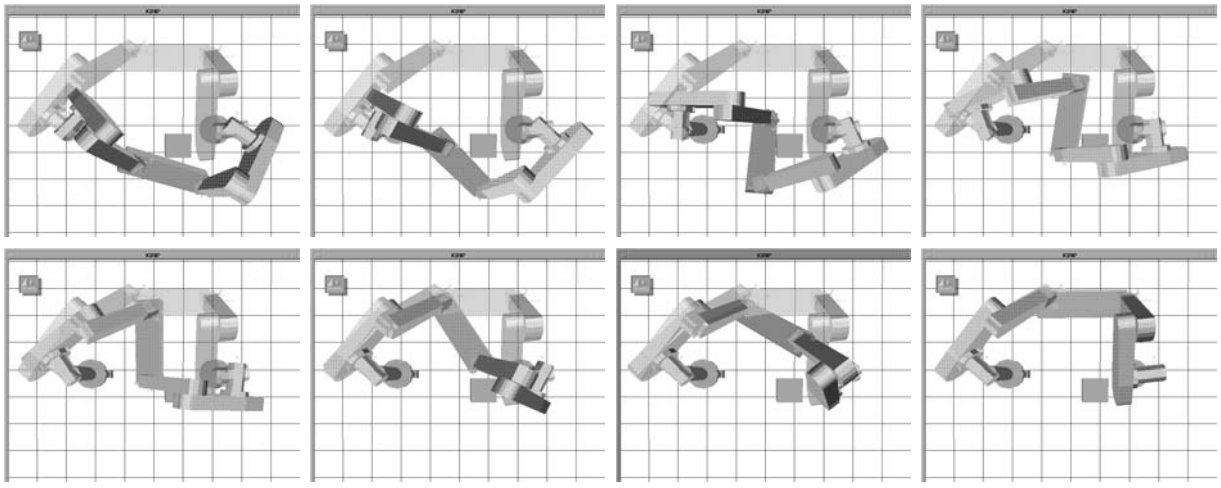


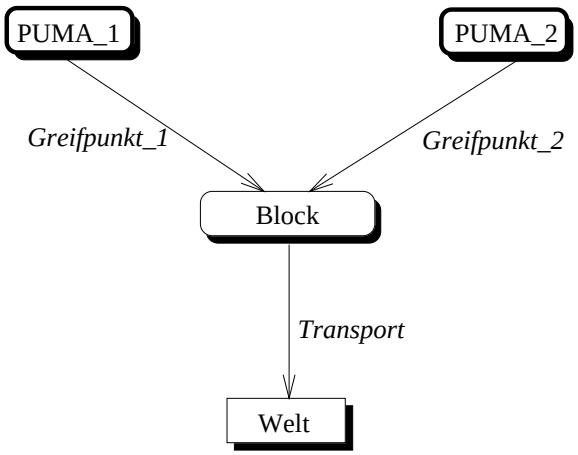
Abbildung 45: Transport eines Objekts: Ausschnitte aus der geplanten Bahn

6.8.5 Aufgabe: KISTE

Zwei Manipulatoren sollen einen Block gemeinsam aus einer Kiste heben und zu einer Zielstellung transportieren. Dabei ist eine Engstelle zwischen den Manipulatoren zu durchfahren. Die Zielstellung liegt knapp hinter einem Hindernis (s. Abb. 46).

Aufgabenspezifikation:

```
TASK: Transport_aus_Kiste
ENVIRONMENT: Zwei_Pumas_mit_Tisch_und_Kiste
GRAPH :
```




```
RELATIONS:


Greifpunkt_1 : (320, -5, 65, 174, 1, 22)
Greifpunkt_2 : (-320, -5, 65, 174, 1, 22)
Transport :
   START : (0, 880, 900, 0, 0, 0) AND
   GOAL : (94, -230, 995, 0, 0, 0)
```

Der Anteil des freien Konfigurationsraums beträgt in diesem Beispiel 0.07 % (Varianz: 0.007 %). Die durchschnittliche Anzahl der generierten Zwischenzielräume beträgt 4.69 (zwischen 2 und 6), die durchschnittliche Planungszeit beträgt 178.3 s (zwischen 53 und 245 s).

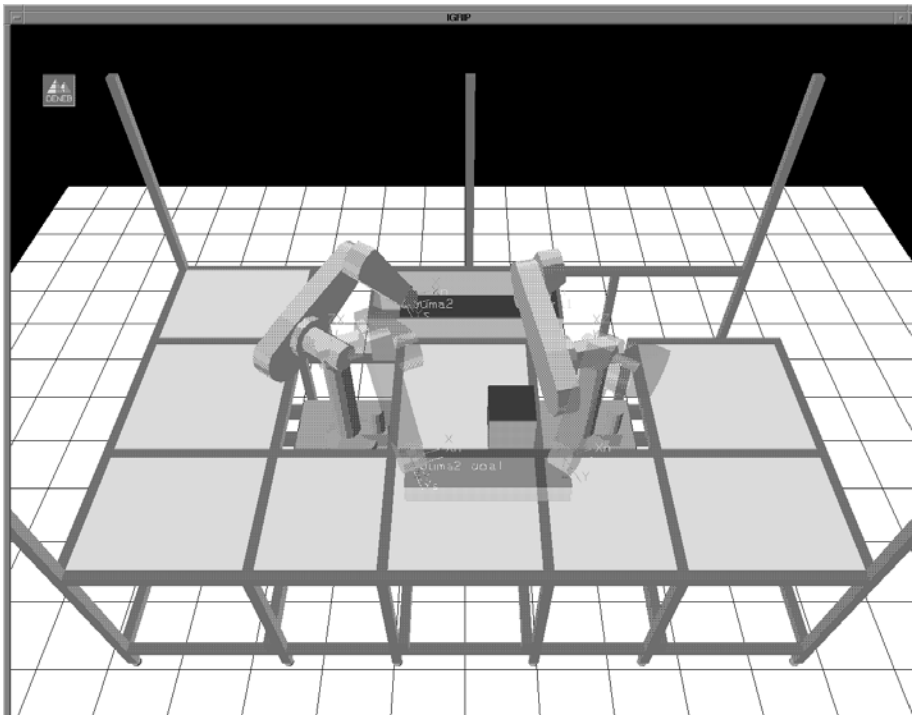


Abbildung 46: Beispiel: Aufgabe KISTE

6.9 Erweiterung der Planungsstrategien für eine umfassende Aufgabenklasse

Nachdem die Planungsstrategien anhand einer eingeschränkten Aufgabenklasse dargestellt wurden, wird im folgenden nun die Verallgemeinerung dieser Planungsstrategien auf die durch die in Kap. 5 beschriebene Sprache spezifizierbaren Aufgaben skizziert. Weiterhin werden dabei ausschließlich die für die Bahnplanung relevanten geometrischen Constraints betrachtet.

Constraints mit Werteinschränkungen Im allgemeinen Fall treten nun offene Freiheitsgrade nicht nur, wie in den bislang betrachteten Aufgaben, in *einer* Beziehung, sondern in *mehreren* Beziehungen auf. A priori beinhaltet jede Beziehung drei translatorische und drei rotatorische Freiheitsgrade. Die formale Zeit ist, wie bereits dargelegt, für alle Beziehungen global.

Betrachtet werden nun Constraints, die durch Gleichungen über reellwertigen Ausdrücken (s. Kap. 4.2) formuliert werden. Es sei gefordert, daß diese Gleichungen nach mindestens einem Freiheitsgrad aufgelöst werden können. Dies schränkt die Klasse der durch Gleichungen spezifizierbaren Constraints nur geringfügig ein. Desweiteren ist zu fordern, daß die eine Beziehung spezifizierenden Gleichungen so aufgelöst werden können, daß für jeden Freiheitsgrad höchstens eine explizite Darstellung als Gleichung existiert. Durch jedes

derartige Constraint wird die Anzahl der Freiheitsgrade in einer Beziehung um einen Freiheitsgrad erniedrigt, da die abhängigen, durch eine Gleichung beschriebenen Freiheitsgrade jeweils durch die freibleibenden Freiheitsgrade bestimmt werden.

Der Konfigurationsraum Seien für eine Aufgabe die n Beziehungen $\mathbf{B_1}$ bis $\mathbf{B_n}$ spezifiziert, wobei $\mathbf{T_{B_i}}$ die Anzahl der translatorischen, $\mathbf{R}_{B_i}$ die Anzahl der rotatorischen Freiheitsgrade der Beziehung $\mathbf{B_i}$ bezeichnet, die nach der Betrachtung obiger Constraints frei bleiben. Der Konfigurationsraum setzt sich somit aus den Freiheitsgraden der einzelnen Beziehungen, den kinematischen Zuständen der Manipulatoren und der formalen Zeit zusammen:

$$\mathbf{C}_{ges} = \mathbb{R}^{\mathbf{T_{B_1}}} \times \mathbb{R}^{\mathbf{R}_{B_1}} \cdots \mathbb{R}^{\mathbf{T_{B_n}}} \times \mathbb{R}^{\mathbf{R}_{B_n}} \cdots \times k_1 \times k_2 \times [0..1]$$

Für die Verwendung von Zielräumen als Zwischenziele und für die Gleitstrategie bildet dieser Konfigurationsraum keine Einschränkung. Es werden weiterhin diese den Konfigurationsraum aufspannenden Freiheitsgrade verwendet.

Eine Schwierigkeit ergibt sich für die Spezifikation des Zielraums der Aufgabe durch derartige Gleichungen: Zur Realisierung eines anziehenden Potentials ist eine Abstandsfunktion einer Konfiguration von dem durch die Gleichungen festgelegten Unterraum erforderlich. Werden als derartige Gleichungen lediglich lineare Gleichungen zugelassen, so entstehen als Unterräume affine Unterräume. Eine Abstandsfunktion einer Konfiguration von einem affinen Unterraum kann mit einfachen Mitteln realisiert werden. Werden allerdings beliebige Gleichungen für die Werteconstraints zugelassen, so können kompliziertere Unterräume entstehen, für die die Bestimmung und Berechnung von Abstandsfunktionen nichttrivial sind.

Constraints mit Bereichseinschränkungen Constraints, die Bereichseinschränkungen darstellen, sind Ausdrücke der Form

$$f(q') < g(q') \qquad \text{bzw.} \qquad f(q') > g(q')$$

mit reellwertigen Ausdrücken $f(q')$ und $g(q')$ (s. Kap. 4.2). Dabei beschreibt q' alle translatorischen und rotatorischen Freiheitsgrade einer Beziehung, sowie die formale Zeit. Da nur geometrische Constraints betrachtet werden, ist in q' keine Sensorinformation enthalten. Durch Verwendung obiger, abhängige Freiheitsgrade beschreibender Gleichungen können f und g als $f(q)$, $g(q)$ ($q \in \mathbf{C}_{ges}$) dargestellt werden, in denen nur die den Konfigurationsraum aufspannenden Freiheitsgrade betrachtet werden.

Diese Ungleichungen können nun in Form von Potentialen in die Schrittgenerierung mit einer Potentialfunktion einbezogen werden. Dabei sind die Potentiale so zu gestalten, daß das einem auf einen Bereich einschränkenden Constraint zugeordnete Potential zu den

Rändern des Bereichs hin stark zunimmt und bei einer Bereichsüberschreitung unendlich wird. Eine Potentialfunktion, die dies erfüllt, ist

$$\mathbf{POT}_{f(q)<g(q)}(q) = \begin{cases} \frac{1}{(f(q)-g(q))^k} & \text{falls} \quad f(q) < g(q) \\ \infty & \text{sonst} \end{cases}$$

mit $k > 0$. Analog wird die Potentialfunktion für Ausdrücke der Form $f(q) > g(q)$ gebildet.

Durch eine Summierung der den i auf Bereiche einschränkenden Constraints c_1 bis c_i einer Aufgabe zugeordneten Einzelpotentiale $\mathbf{POT}_{c_1}$ bis $\mathbf{POT}_{c_i}$ ergibt sich eine Gesamtpotentialfunktion

$$\mathbf{POT}_{BerConstr}(q) = \sum_{n=1}^{i} \mathbf{POT}_{c_n}(q),$$

die in das in Kap. 6.6.3 beschriebene abstoßende Potential $\mathbf{POT}(q, zr)$ einbezogen werden kann und somit die erzeugten Schritte von den Bereichsgrenzen der Constraints fernhält.

$$\mathbf{POT}(q, zr) = k \cdot \mathbf{POT}_{anziehend}(q, zr) + \mathbf{POT}_{abstoßend}(q) + \mathbf{POT}_{BerConstr}(q)$$

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

Da, wie in Kap. 3 gezeigt wurde, die vorhandenen Ansätze zur Programmierung kooperierender Manipulatoren mit Abhängigkeiten zwischen den Effektorkoordinatensystemen auf Bewegungsebene nicht ausreichen, wurde eine Methode zu ihrer aufgabenorientierten Programmierung vorgestellt.

In Kap. 4 wurde dazu eine formale Darstellung von Elementaraufgaben für kooperierende Manipulatoren beschrieben, aus dem in Kap. 5 eine formale Sprache zu ihrer Spezifikation abgeleitet wurde. In einem ersten Schritt werden die für eine Aufgabe relevanten Koordinatensysteme bestimmt und diese zusammen mit den zwischen ihnen herrschenden Beziehungen in einem Graphen festgelegt. Mit dem Konzept der *zeitabhängigen Transformationen* wurde ein mächtiger, einheitlicher Formalismus zur Beschreibung dieser Beziehungen zwischen den Koordinatensystemen entwickelt. Zeitabhängige Transformationen werden durch geometrische, zeitliche und auf Sensorinformation basierende Constraints beschrieben.

Mit dem vorgestellten Formalismus ist es nun möglich, eine große Klasse von Aufgaben für kooperierende Manipulatoren zu formulieren. Es können zeitliche, geometrische und auf Sensorinformation basierende Sachverhalte beschrieben werden.

Aufgaben, die durch diesen Formalismus beschrieben werden, sind durch Methoden zur Parametrierung von Regelungsgesetzen, Bahnplanungswerkzeuge und Methoden zur Behandlung der Dynamik der Manipulatoren in ausführbare Form umzusetzen. Die Erzeugung kollisionsfreier Bahnen aus der Aufgabenspezifikation durch Bahnplaner wird in Kap. 6 behandelt.

Dort wird zunächst die Darstellung des bei der Bahnplanung zu verwendenden Konfigurationsraums diskutiert. Für die betrachteten Aufgaben mit festen Abhängigkeiten zwischen den Effektorkoordinatensystemen wird eine Darstellung des Konfigurationsraums durch die Position und Orientierung eines zentralen Koordinatensystems vorgeschlagen. Gegenüber Ansätzen mit Gelenkparameterdarstellung ergeben sich dadurch für die Definition des freien Konfigurationsraums neue Bedingungen. Dies sind die Existenz von Lösungen der inversen Kinematik der verwendeten Manipulatoren und die Lage der Lösungen innerhalb der Gelenkgrenzen. Da die Gelenkparameter der verwendeten Manipulatoren aus der Lage des zentralen Koordinatensystems durch ihre inverse Kinematik bestimmt werden, diese aber mehrdeutig ist, wird zusätzlich der kinematische Zustand der verwendeten Manipulatoren für die Konfigurationsraumdarstellung angegeben. Zur Modellierung zeitlicher Abhängigkeiten zwischen den Manipulatoren stellt die formale Zeit, die den Aufgabenverlauf beschreibt und von Realzeitbedingungen trennt, einen zusätzlichen Freiheitsgrad des Konfigurationsraums dar.

Die Planungsalgorithmen sind auf effiziente Einsetzbarkeit in realistischen, dreidimensionalen Umgebungen ausgelegt, was zu der Notwendigkeit einer sparsamen Verwendung von

zeitlich teuren Kollisionstests und der Entwicklung entsprechender effizienter Heuristiken führt.

Der vorgestellte Bahnplaner realisiert ein Zusammenspiel aus einer lokalen und einer globalen Planungsstrategie. Der lokale Planer hat die Aufgabe, möglichst effizient einen kollisionsfreien Weg zwischen einer Startkonfiguration und einer Menge von Zielkonfigurationen zu finden. Die geplanten Wege ergeben sich durch eine schrittweise Konstruktion einer Folge von hinreichend nahe beieinanderliegenden, kollisionsfreien Konfigurationen, wodurch lediglich einzelne Punkte des Konfigurationsraums auf Kollision zu testen sind. Die Bestimmung der Schritte zwischen diesen Konfigurationen erfolgt mittels einer Potentialfeldsuche in einem effizient berechenbaren Potentialfeld, das sich aus einem zum Ziel hin anziehenden und einem von Konfigurationsraumhindernissen abstoßenden Potentialfeld zusammensetzt. Während die Entfernung von den Gelenkgrenzen der verwendeten Manipulatoren effizient berechnet werden kann, ist dies für die Berechnung der Entfernung von Hindernissen nicht der Fall. Stattdessen wird eine Heuristik *Entlanggleiten an Hindernissoberflächen* verwendet.

In der Aufgabenspezifikation werden als allgemeinere Planungsziele Ziel*räume* anstatt von Zielkonfigurationen eingeführt. Die vorgestellte Planungsstrategie ist in der Lage, Zielräume als Planungsziele zu verwenden. Die Verwendung von Zielräumen als Zwischenziele ermöglicht desweiteren eine wesentlich bessere Gleitleistung des lokalen Planers gegenüber der Verwendung einzelner Zielkonfigurationen, womit dieser häufiger erfolgreich ist.

Da effiziente Berechnung von Potentialen und deren Freiheit von lokalen Minima einen Widerspruch darstellen, werden zwei Methoden eingesetzt, um das Problem der auftretenden lokalen Minima zu lösen. Zum einen wurde eine Methode zur *adaptiven Gewichtung* des anziehenden Potentials entwickelt. Diese Methode paßt aufgrund des Zielfortschritts des zuletzt geplanten Schritts die Gewichtung des anziehenden Potentialfelds für den nächsten Schritt an und ist somit in der Lage, Sattelpunkte des Potentialfelds zu überwinden.

Dennoch bleiben neben Sattelpunkten echte Sackgassen des Konfigurationsraums als lokale Minima des Potentials, die durch diese Methode nicht überwunden werden können. Hierzu wird eine globale Strategie eingesetzt, die im Falle des Versagens des lokalen Planers durch die Generierung von zufälligen Zwischenzielräumen schrittweise einen immer dichter werdenden Graphen aus Zwischenzielen und diese verbindenden kollisionsfreien Wegen aufbaut. Die Suche ist beendet, wenn Start und Ziel in einer Zusammenhangskomponente dieses Graphen liegen und somit durch einen kollisionsfreien Weg verbunden werden können.

Die Bahnplanungsstrategien sind in der Lage, vorgegebene zeitabhängige Transformationen zwischen Koordinatensystemen zu berücksichtigen, die bei der über die reine Planung kollisionsfreier Bahnen hinausgehenden Planung von Manipulationsvorgängen durch mehrere Manipulatoren grundlegend sind.

Ebenfalls in die Planungsstrategien integriert ist die Möglichkeit, eine wichtige Klasse von

in der Aufgabenspezifikation spezifizierbaren geometrischen Constraints, nämlich Wert- und Bereichseinschränkungen für einzelne Freiheitsgrade der Lage eines manipulierten Objekts relativ zum Weltkoordinatensystem, zu berücksichtigen. Ebenso wurde die Behandlung allgemeiner geometrischer Constraints durch die vorgestellten Planungsverfahren skizziert.

Es wurde gezeigt, daß durch die Verwendung effizienter Heuristiken für Aufgaben für kooperierende Manipulatoren mit zeitvariabel abhängigen Effektorkoordinatensystemen unter Berücksichtigung zusätzlicher geometrischer Einschränkungen in realistischen, dreidimensionalen Umgebungen für eine Onlineplanung ausreichende Planungszeiten erzielt werden können.

7.2 Ausblick

Die aufgabenorientierte Programmierung von Manipulatoren stellt eine sehr komplexe und in den Anfangsjahren der Forschung auf diesem Gebiet stark unterschätzte Aufgabe dar. Mindestens genauso komplex stellt sich die aufgabenorientierte Programmierung kooperierender Manipulatoren dar, ein noch recht junger Forschungszweig. Nachdem in dieser Arbeit aus Komplexitätsgründen lediglich ein Kernbereich der hierzu erforderlichen Themengebiete behandelt wurde (Beschreibung der geometrischen und kinematischen Sachverhalte in einer Aufgabe; Behandlung auftretender Kräfte; Planung kollisionsfreier Bahnen), bleibt die Integration weiterer wichtiger Bereiche in Konzepte zur aufgabenorientierten Programmierung kooperierender Manipulatoren offen. Um nur einige zu nennen, handelt es sich hierbei um die Behandlung der Manipulatordynamik, die umfassende Einbeziehung von Sensorik und – aufbauend auf der in dieser Arbeit behandelten Schicht von Elementaroperationen – die Realisierung von in ein Programmiersystem eingebetteten Aufgabentransformations- und Ausführungsüberwachungssystemen, die die Spezifikation komplexer, symbolisch beschriebener Aufgaben, sowie deren Zerteilung in Elementaroperationen und deren Ausführung leisten.

In dieser Arbeit wurden zur Umsetzung durch Bahnplanungsverfahren lediglich Aufgaben betrachtet, bei denen die formale Zeit den einzigen Freiheitsgrad in der Beziehung zwischen den Effektorkoordinatensystemen darstellt. Es müssen Bahnplanungsverfahren entwickelt werden, die in der Lage sind, effizient zusätzliche Freiheitsgrade zwischen den Effektorkoordinatensystemen zu behandeln, wie sie durch den vorgestellten Spezifikationsformalismus beschrieben werden können.

Obwohl bereits sehr leistungsfähige Bahnplanungsverfahren zur Verfügung stehen, bleiben in diesem Gebiet noch viele Fragestellungen offen. Durch die Verwendung redundanter Manipulatoren zum einen und der Einbeziehung zusätzlicher Freiheitsgrade zwischen den Effektorkoordinatensystemen zum anderen werden Planungsverfahren für höherdimensionale Konfigurationsräume in realistischen Umgebungen erforderlich. Zwar wurden bereits Planungsverfahren für hochdimensionale Konfigurationsräume (31 Freiheitsgrade) vorgestellt [BL90, BL91], diese basieren aber auf für realistische Anwendungen unzulässigen

Annahmen, etwa der Verwendung redundanter Manipulatoren in lediglich zweidimensionalen Umgebungen oder der Verwendung von Linienmanipulatoren mit dem Effekt sehr schneller Kollisionstests. Dennoch bleibt bislang der Widerspruch bestehen, daß zwar einerseits durch zusätzliche Freiheitsgrade mehr Aufgaben lösbar sind bzw. einfachere Lösungen existieren, der Aufwand zum Auffinden dieser Lösungen bedingt durch den größeren Suchraum mit der Anzahl der Freiheitsgrade jedoch stark zunimmt. Ein Hauptaugenmerk liegt dabei auf der Entwicklung von Heuristiken zur Potentialauswertung, die einen in den Freiheitsgraden konstanten Aufwand erfordern. Ein Weg in diese Richtung stellt etwa der Ansatz der Potentialauswertung mittels *Simulated Annealing* dar.

In einer sehr interessanten Arbeit hat Quinlan [Qui95] eine skalierbare, schnelle Methode zur Abstandsberechnung vorgestellt. Hierdurch ist es möglich, eine Schätzung für eine untere Schranke der Entfernung zweier Objekte zu erhalten. Die Abweichung der unteren Schranke von der realen Entfernung kann dabei kontinuierlich von 0 Prozent (exakte Entfernungsberechnung) bis 100 Prozent (lediglich Kollisionstest) vorgegeben werden. Die Laufzeit ist dabei proportional zur eingestellten Abweichung. Bei angemessenen Abweichungen (etwa 20 Prozent) können dabei Zeiten erreicht werden, die lediglich um den Faktor 2 bis 3 über den Zeiten der leistungsfähigsten Kollisionstests liegen. Durch derartige Algorithmen und die weiter zunehmende Rechenleistung liegt die Einbeziehung von (gegebenenfalls vager) Abstandsinformation in Potentialfunktionen im Bereich des Machbaren.

Eine weitere Forschungsrichtung stellen Manipulationsplaner dar, die die Integration von Greifplanern, Umgreifbewegungen und den hier beschriebenen Manipulationsbewegungen leisten und dadurch in der Lage sind, auf höherer Ebene spezifizierte Manipulationsaufgaben in die Bewegungsebene umzusetzen. Neben dem Bereich der Manipulationsbewegungen, also Relativbewegungen zwischen Effektoren und manipulierten Objekten, der in dieser Arbeit betrachtet wird, sind hierbei Umgreifbewegungen, die in einer interessanten Arbeit von Koga [Kog95] behandelt werden, zu betrachten. Ein weiterer, bislang nicht betrachteter Punkt ist die erforderliche starke Integration von Greifplanern in ein Manipulationsplanungssystem, da eine Planung bzw. Auswahl von Greifpunkten abhängig von einer aktuellen Situation im Planungsvorgang erfolgen muß.

Literatur

[33194] SFB 331. *Finanzierungsantrag 1995 – 1997*, Kapitel Informationsverarbeitung in autonomen, mobilen Handhabungssystemen, Seiten 85–150. Technische Universität München, 1994.

[Asa94] S. Asami. Service Robots in Japan. Present and Future. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, San Diego, California, Mai 1994. Survey Paper.

[Bag95] B. Baginski. Das ZZZ-Verfahren zur schnellen Bahnplanung in dynamischen Umgebungen. Internes Papier, in Vorbereitung, Technische Universität München, Institut für Informatik, 1995.

[BAH94] B. Brunner, K. Arbter und G. Hirzinger. Task Directed Programming of Sensor Based Robots. In *Proceedings of the 1994 IEEE/RSI/GI International Conference on Intelligent Robots and Systems*, Seiten 1080–1087, München, September 1994.

[BHHL93] B. Brunner, J. Heindl, G. Hirzinger und K. Landzettel. Telerobotics systems using virtual environment display with visual and force display functions. In *Workshop on Force Display in Virtual Environments and its Application to Robotic Teleoperation, IEEE International Conference on Robotics and Automation*, Atlanta, Mai 1993.

[BJ83] Ch. Blume und W. Jakob. *Programmiersprachen für Industrieroboter, Konzepte und Sprachen.* Vogel-Verlag, Würzburg, 1983.

[BL90] J. Barraquand und J.-C. Latombe. A Monte-Carlo Algorithm for Path Planning With Many Degrees of Freedom. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Seiten 1712–1717, Cincinnati, Ohio, Mai 1990.

[BL91] J. Barraquand und J.-C. Latombe. Robot Motion Planning: A Distributed Representation Approach. *International Journal of Robotics Research*, 10(6):628–649, 1991.

[Boc90] S. Bocionek. *Task-Level Programming of Manipulators: A Case Study.* Technischer Bericht TUM-I9001, Technische Universität München, Institut für Informatik, 1990.

[BS87] I. N. Bronstein und K.A. Semendjajew. *Taschenbuch der Mathematik, 23. Aufl.* Verlag Harri Deutsch, 1987.

[Can88] J. F. Canny. *The Complexity of Robot Motion Planning.* MIT Press, 1988.

[CH92] P. C. Chen und Y. K. Hwang. SANDROS: A Motion Planner with Performance Proportional to Task Difficulty. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Seiten 2346–2353, Nice, France, Mai 1992.

[Den92] Deneb Robotics Inc., Auburn Hills, MI. *IGRIP Robot Simulation System Manual*, 1992.

[DKHM87] G. Duelen, U. Kirchhoff, J. Held und H. Münch. Automatische Bewegungssynthese für bahnbezogen kooperierende Industrieroboter. *Robotersysteme*, 3:107–113, 1987.

[DKS91] M. Damm, D. Kappey und J. Schloen. A Multi-sensor and Adaptive Realtime Control Architecture for an Autonomous Robot. In *Fifth International conference on advanced Robotics*, Seiten 411 – 416, Pisa, Italien, 1991.

[FH87] E. Freund und H. Hoyer. Automatische Bahnbestimmung in Echtzeit für Robotersysteme. *Robotersysteme*, 3:89–100, 1987.

[Fis80] G. Fischer. *Lineare Algebra.* Vieweg, Braunschweig, 1980.

[Fis88] K. Fischer. *Regelbasierte Synchronisation zwischen Roboter und Maschinen.* Technischer Bericht TUM-I9001, Technische Universität München, Institut für Informatik, 1988.

[Fis92] K. Fischer. *Verteiltes und kooperatives Planen in einer flexiblen Fertigungsumgebung.* Dissertation, Technische Universität München, Institut für Informatik, 1992.

[Fis94a] M. Fischer. Efficient Path Planning Strategies for Cooperating Manipulators in Environments with Obstacles. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, Seiten 2418–2423, San Diego, California, Mai 1994.

[Fis94b] M. Fischer. A Path Planner for Cooperating Robots in Realistic Environments. In Leon Žlajpah, Herausgeber, *Robotics in Alpe–Adria Region, Proceedings of the 3rd International Workshop (RAA '94)*, Seiten 181–186, Bled, Slowenien, Juli 1994. Jožef Stefan Institute.

[FK85] B. R. Fox und K. R. Kempf. Opportunistic scheduling for robot assembly. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, Seiten 880–889, St. Louis, Missouri, 1985.

[FM94] J. J. Fox und A. Maciejewski. Utilizing the Topology of Configuration Space in Real-Time Multiple Manipulator Path Planning. In *Proceedings of the 1994 IEEE/RSI/GI International Conference on Intelligent Robots and Systems*, Seiten 665–672, München, September 1994.

[FTB+75] R. Finkel, R. Taylor, R. Bolles, R. Paul und J. Feldman. An Overview of AL, a Programming System for Automation. In *Proc. of the 4th International Joint Conference on Artificial Intelligence*, Seiten 758 – 765, 1975.

[FV82] J. W. Franklin und G. J. Vanderburg. Programming Vision and Robotics Systems with RAIL. In *Proceedings of Robots VI*, Seiten 978–985, Detroit, März 1982.

[Gag93] A. Gagalowicz. Vision-Driven Home Robots. In W. Strasser und F. Wahl, Herausgeber, *Graphics & Robotics*, Dagstuhl Seminar-Report 61, Schloss Dagstuhl, April 1993.

[GJK87] E. G. Gilbert, D. W. Johnson und S. S. Keerthi. A Fast Procedure For Computing The Distance Between Complex Objects in Three-Dimensional Space. In *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, Seiten 1883–1889, Raleigh, New Carolina, 1987.

[Gla90] B. Glavina. Solving Findpath by Combination of Goal-Directed and Randomized Search. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Seiten 1718–1723, Cincinnati, Ohio, Mai 1990.

[Gla91a] B. Glavina. A Fast Motion Planner for 6-DOF Manipulators in 3-D Environments. In *Proceedings of the Fifth International Conference on Advanced Robotics*, Seiten 1176–1181, Pisa, Italy, Juni 1991.

[Gla91b] B. Glavina. *Planung kollisionsfreier Bewegungen für Manipulatoren durch Kombination von zielgerichteter Suche und zufallsgesteuerter Zwischenzielerzeugung.* Dissertation, Technische Universität München, Institut für Informatik, 1991.

[GM91] J. Graf und W. Meier. Two-Arm Coordination using Trajectory Optimisation and an Integrated Control System. In *International Symposium on Advanced Robot Technology (ISART)*, Tokio, Japan, 1991.

[GWNO87] M. P. Groover, M. Weis, R. N. Nagel und N. G. Odrey. *Robotik umfassend.* McGraw–Hill, Hamburg, New York, 1987.

[Hag92] E. Hagg. *Realisierung von Multisensoranwendungen mit vernetzten Logischen Sensoren und Aktoren.* Dissertation, Technische Universität München, Institut für Informatik, 1992.

[HBDH94] G. Hirzinger, B. Brunner, J. Dietrich und J. Heindl. ROTEX: The First Remotely Controlled Robot in Space. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, Seiten 2604–2611, San Diego, California, Mai 1994.

[HGB94] H. Hoyer, M. Gerke und U. Borgolte. Online Collision Avoidance for Industrial Robots with Six Degrees of Freedom. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, Seiten 1258–1263, San Diego, California, Mai 1994.

[HH90a] A. Hörmann und K. Hörmann. Planung kollisionsfreier Greifoperationen. *Robotersysteme*, 6:39 – 50, 1990.

[HH90b] A. Hörmann und K. Hörmann. Planung kollisionsfreier Greifoperationen – Analyse der Objektgeometrie. In *Robotersysteme 7*, Berlin, 1990. Springer-Verlag.

[HLF94] G. Hirzinger, K. Landzettel und Ch. Fagerer. Telerobotics With Large Time Delays – The ROTEX Experience. In *Proceedings of the 1994 IEEE/RSI/GI International Conference on Intelligent Robots and Systems*, Seiten 571–578, München, September 1994.

[HN92] Y. K. Hwang und A. Narendra. Gross Motion Planning – A Survey. In *ACM Computing Surveys, Vol. 24, No. 3.* ACM, September 1992.

[Hör88] K. Hörmann. *Kollisionsfreie Bahnen für Industrieroboter. Ein Planungsverfahren.* Springer Verlag, Informatik Fachberichte; 166, Berlin, 1988.

[HR91] A. Hörmann und U. Rembold. Development of an Advanced Robot for Autonomous Assembly. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Seiten 2452 – 2457, Sacramento, California, April 1991.

[KKKL94] Y. Koga, K. Kondo, J. Kuffner und J.-C. Latombe. Planning Motions with Intentions. In *Proceedings of SIGGRAPH '94*, Orlando, Florida, Juli 1994.

[KL92] Y. Koga und J.-C. Latombe. Experiments in Dual-Arm Manipulation Planning. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Seiten 2238–2245, Nice, France, Mai 1992.

[KL94a] L. Kavraki und J.-C. Latombe. Randomized Preprocessing of Configuration Space: Articulated Robots. In *Proceedings of the 1994 IEEE/RSI/GI International Conference on Intelligent Robots and Systems*, Seiten 1764–1771, München, September 1994.

[KL94b] L. Kavraki und J.-C. Latombe. Randomized Preprocessing of Configuration Space for Fast Path Planning. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, Seiten 2138–2145, San Diego, California, Mai 1994.

[Kog95] Y. Koga. *On Computing Multi-Arm Manipulation Trajectories.* Dissertation, Department of Mechanical Engineering, Stanford University, Stanford, CA, 1995.

[Kop94] P. Kopacek. Robotics Research Today and in the Future. In Leon Žlajpah, Herausgeber, *Robotics in Alpe–Adria Region, Proceedings of the 3rd International Workshop (RAA '94)*, Seiten 213–217, Bled, Slowenien, Juli 1994. Jožef Stefan Institute.

[KP94] G. Kess und P. Pérez. *Erweiterung der RKES-Schnittstelle um Koordinatentransformations und Greiferdienste und Loesen des Rubic's Cube mit zwei Robotern.* Fortgeschrittenenpraktikum, Technische Universität München, Institut für Informatik, 1994.

[Küh90] U. Kühnapfel. KISMET – 3D-Grafik zur Planung, Programmierung und Überwachung von Telerobotics-Applikationen. In *VDI-Berichte Nr. 861.3, S.71-86.* VDI-Verlag, Düsseldorf, 1990.

[Lat91] J.-C. Latombe. *Robot motion planning.* Kluwer Academic Publishers, Boston, MA, 1991.

[Lev88] P. Levi. *Planen für autonome Montageroboter.* Nummer 191 in Informatik Fachberichte. Springer-Verlag, Berlin, 1988.

[Ley91] J. Leysen. *A Knowledge Oriented Approach to the Synthesis and Monitoring of Force Controlled Robot Programs.* Dissertation, Universität Leuwen, Belgien, 1991.

[LMT91] J. Lang, W. Meier und W. Thomma. Anwendung von fortgeschrittenen Positions-Kraft-Regelungskonzepten bei einem Zweiarmrobotersystem. In *Robotersysteme 7*, Berlin, 1991. Springer-Verlag.

[LP81] T. Lozano-Pérez. Automatic Planning of Manipulator Transfer Movements. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(10):681–698, 1981.

[LP83a] T. Lozano-Pérez. Robot Programming. *IEEE Transactions on Robotics and Automation, Bd. 71, 7*, Juli 1983.

[LP83b] T. Lozano-Pérez. Spatial Planning; A Configuration Space Approach. *IEEE Transactions on Computers*, C-32(2):108–120, 1983.

[LP87] T. Lozano-Pérez. A Simple Motion-Planning Algorithm for General Robot Manipulators. *IEEE Journal of Robotics and Automation*, RA-3(3):224–238, 1987.

[LPW77] T. Lozano-Pérez und P. H. Winston. LAMA: A Language for Automatic Mechanical Assembly. In *Proc. of the 5th International Joint Conference on Artificial Intelligence*, Seiten 710–716, Cambridge, MA, 1977.

[LW77] L.I. Liebermann und M.A. Wesley. AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly. *IBM Journal of Research and Development*, 21(4):321 – 333, 1977.

[MA90] S. B. Moon und S. Ahmad. Time Scaling of Cooperative Multi-Robot Trajectories. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Seiten 506–511, Cincinnati, Ohio, Mai 1990.

[Mas82] M. T. Mason. Compliant Motion. In M. Brady et al., Herausgeber, *Robot Motion*. MIT Press Series in Artificial Intelligence, 1982.

[Pau81] R. Paul. *Robot Manipulators: Mathematics, Programming and Control*. MIT Press, Cambridge, Mass., 1981.

[PB90] G. Pritschow und M. Bauder. Steuerungsstruktur und Programmierkonzept zur On-line-Bewegungskoordinierung zweier Roboter in einer flexiblen Montagezelle. *Robotersysteme*, 6:211–217, 1990.

[PK91] G. Pritschow und T. Koch. Koordinierte Bahnführung zweier Roboter. *Robotersysteme*, 7:133–138, 1991.

[Qui95] S. Quinlan. Real-Time Modification of Collision-Free Paths. Technical Report STAN CS-TR-95-1537, Department of Computer Science, Stanford University, Stanford, CA, 1995.

[RH94] E. Ralli und G. Hirzinger. Fast Path Planning for Robot Manipulators Using Numerical Potential Fields in the Configuration Space. In *Proceedings of the 1994 IEEE/RSI/GI International Conference on Intelligent Robots and Systems*, Seiten 1922–1929, München, September 1994.

[Rie93] A. Rieder. *Entwicklung von Konzepten und Realisierung eines Bewegungsplaners für kooperierende Roboter*. Diplomarbeit, Technische Universität München, Institut für Informatik, 1993.

[Sch86] J. De Schutter. *Compliant Robot Motion: Task Formulation and Control*. Dissertation, Universität Leuwen, Belgien, 1986.

[Sch88] J. De Schutter. Compliant Robot Motion I: A Formalism for Specifying Compliant Motion Tasks. *International Journal of Robotics Research, 7/4*, Seiten 3–17, 1988.

[Sch89] A. Schweikard. *Geometrische Verfahren zur Kollisionserkennung und Wegbestimmung für kooperierende Roboter.* Dissertation, Fachbereich Informatik, Technische Universität Berlin, 1989.

[Sch94a] R. Schilwat. *Entwurf und Implementierung von Methoden zur Simulation von laserbasierten abstandsmessenden Sensoren mit Hilfe des Simulationssystems IGRIP.* Diplomarbeit, Technische Universität München, Institut für Informatik, 1994.

[Sch94b] R. Schraft. Service Robots : Opportunities, Possibilities and Potentials for the Next Decade. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, San Diego, California, Mai 1994. Survey Paper.

[SGIS85] B.E. Shimano, C.C. Geschke, C.H. Spalding III und P.G. Smith. A Robot Programming System Incorporating Real-Time and Supervisory Control: VAL-II. In K. Rathmill, Herausgeber, *Robotic Assembly*, Seiten 201 – 217. Springer-Verlag, 1985.

[SSH87] J. T. Schwartz, M. Sharir und J. Hopcroft. *Planning, Geometry and Complexity of Robot Motion.* Ablex, Norwood, NJ, 1987.

[Ste92] R. Stetter. Einbindung physikalischer Effekte in die 3D-Simulation eines Handhabungsprozesses. *Robotersysteme*, 8:134 – 138, 1992.

[Sto83] J. Stoer. *Einführung in die numerische Mathematik I, 4. Aufl.* Springer-Verlag, Berlin, 1983.

[Tay82] R. H. Taylor. AML: A Manufacturing Language. *International Journal of Robotics Research*, 1(3), 1982.

[Tec92] Tecnomatix Europe N. V. *ROBCAD Reference Manual*, 1992.

[Tsa91] C.-K. Tsai. Multiple Robot Coordination and Programming. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Seiten 978–985, Sacramento, California, April 1991.

[VMT90] P. Violero, I. Mazon und M. Taix. Automatic Planning of a Grasp for a "Pick and Place" Action. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Seiten 870–876, Cincinnati, Ohio, Mai 1990.

[Vol91] L. Volkmann. *Graphen und Digraphen – Eine Einführung in die Graphentheorie.* Springer-Verlag, Wien, 1991.

[WBK94] Ch. Woenckhaus, L. Bauer und D. Kugelmann. USIS - ein Simulationswerkzeug für Layoutplanung und Offline-Programmierung. *CIM Management*, 3(10):20 – 23, 1994.

[Wer93] G. Werling. *Produktorientierte automatische Planung von Prüfoperationen bei der robotergestützten Montage*. DISKI 46. Infix-Verlag, 1993.

[YZ92] Q. Yin und Y. Zheng. Performance Analysis of Token Bus LAN in Coordinating Multiple Robots. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Seiten 455–460, Nice, France, Mai 1992.

[ZLJ89] Y. Zheng, J. Y. S. Luh und P. F. Jia. Integrating two industrial robots in to a coordinated system. *Computers in Industry*, 12:285–298, 1989.

A Sprachbeschreibung

Im folgenden werden die Produktionen der kontextfreien Grammatik der Spezifikationssprache in erweiterter Backus-Naur-Form beschrieben. Nonterminale werden normal, Terminale fett gedruckt. Das Axiom der Grammatik ist AUFGABE. Entsprechend [McG80] werden als Ersetzungsoperator ::=, zur Metaklammerung { }, für optionale Bestandteile [,] als Variantenstrich | und als Replikationsoperatoren $\{\}^+$ und $\{\}^*$ verwendet.

AUFGABE ::= **TASK:** NAME
ENVIRONMENT: NAME
GRAPH: ⟨GRAPH⟩
RELATIONS: {BEZIEHUNG}$^+$
[**GLOBAL_CONSTRAINT:** G_CONSTRAINT]
[**TERMINATION_CONDITIONS:**]
{TERMINIER_GES_NAME : TERMINIER_GESAMT}$^+$

⟨GRAPH⟩ steht für die graphische Darstellung des die Aufgabenstruktur darstellenden Graphen.

BEZIEHUNG ::= BEZIEHUNG_NAME :
[CONSTRAINT]
[**TERMINATION** : {TERMINIER_BED}$^+$]

CONSTRAINT ::= EINF_CONSTRAINT
| (CONSTRAINT)
| CONSTRAINT **AND** CONSTRAINT
| CONSTRAINT **OR** CONSTRAINT
| **NOT** CONSTRAINT

G_CONSTRAINT ::= G_GEO_CONSTRAINT
| (G_CONSTRAINT)
| G_CONSTRAINT **AND** G_CONSTRAINT
| G_CONSTRAINT **OR** G_CONSTRAINT
| **NOT** CONSTRAINT

EINF_CONSTRAINT ::= STELLUNG

| START_SPEZ
| ZIEL_SPEZ
| INTERPOL_SPEZ
| GEO_CONSTRAINT
| REALZEIT_CONSTR
| KRAFT_CONSTR
| TRACK_CONSTR

START_SPEZ ::= **START** : STELLUNG

ZIEL_SPEZ ::= **GOAL** : STELLUNG

INTERPOL_SPEZ ::= INTERPOL_TYP :
STELLUNG{, STELLUNG}$^{+}$
END_INTERPOLATED

INTERPOL_TYP ::= **LINEAR_INTERPOLATED**
| **QUADRATIC_INTERPOLATED**
| **CUBIC_INTERPOLATED**
| **SPLINE_INTERPOLATED**

GEO_CONSTRAINT ::= REAL_EXP VERGL_OP REAL_EXP
| REAL_EXP $\in$ [REAL_EXP , REAL_EXP]

TERMINIER_BED ::= TERMINIER_BED_NAME : TERM_CONSTR

TERM_CONSTR ::= GEO_CONSTRAINT
| KRAFT_CONSTRAINT
| REALZEIT_CONSTR

TERMINIER_GESAMT ::= **T = 1**
| TERMINIER_BED_NAME
| **NOT** (TERMINIER_GESAMT)
| (TERMINIER_GESAMT) **AND** (TERMINIER_GESAMT)
| (TERMINIER_GESAMT) **OR** (TERMINIER_GESAMT)

STELLUNG ::= (REAL_KONST, REAL_KONST, REAL_KONST,
REAL_KONST, REAL_KONST, REAL_KONST)

Produktionen für globale Constraints

G_GEO_CONSTRAINT ::= G_REAL_EXP VERGL_OP G_REAL_EXP
| G_REAL_EXP ∈ [G_REAL_EXP , G_REAL_EXP]

G_REAL_EXP ::= REAL_KONST
| REAL_VARIABLE
| (G_REAL_EXP)
| G_REAL_EXP REAL_OP G_REAL_EXP
| REAL_FUNKTION (G_REAL_EXP{, G_REAL_EXP}*)
| G_REAL_SELEKT_EXP
| G_SKALARPROD
| G_MASS

G_REAL_SELEKT_EXP ::= KOORDSYS_NAME . DOF_SELEKT (TIME_EXP)

G_SKALARPROD ::= G_VEKTOR_EXP · G_VEKTOR_EXP

MASS ::= **DISTANCE** (G_FRAME_KOMP, G_FRAME_KOMP)
| **ANGLE** (G_FRAME_KOMP, G_FRAME_KOMP)

G_VEKTOR_EXP ::= VEKTOR_KONST
| KOORDSYS_NAME . VEKTOR_SELEKTOR
| (G_VEKTOR_EXP)
| G_VEKTOR_EXP VEKTOR_OP G_VEKTOR_EXP
| REAL_EXP · G_VEKTOR_EXP

G_FRAME_KOMP ::= G_VEKTOR_EXP
| KOORDSYS_NAME . KOMP_SELEKTOR

Produktionen für reellwertige Ausdrücke

REAL_EXP ::= REAL_KONST
| REAL_VARIABLE
| (REAL_EXP)
| REAL_EXP REAL_OP REAL_EXP
| REAL_FUNKTION (REAL_EXP{, REAL_EXP}*)
| REAL_SELEKT_EXP
| SKALARPROD
| MASS

REAL_OP ::= + | - | * | /

REAL_FUNKTION ::= **EXP** | **SQRT** | **SIN** | **COS** | **TAN**
| **ACOS** | **ASIN** | **ATAN** | **LOG**

REAL_SELEKT_EXP ::= DOF_SELEKT (TIME_EXP)

DOF_SELEKT ::= **TX** | **TY** | **TZ** | **RX** | **RY** | **RZ**

SKALARPROD ::= VEKTOR_EXP · VEKTOR_EXP

MASS ::= **DISTANCE** (FRAME_KOMP, FRAME_KOMP)
| **ANGLE** (FRAME_KOMP, FRAME_KOMP)

Produktionen für vektorwertige Ausdrücke

VEKTOR_EXP ::= VEKTOR_KONST
| VEKTOR_SELEKTION
| (VEKTOR_EXP)
| VEKTOR_EXP VEKTOR_OP VEKTOR_EXP
| REAL_EXP · VEKTOR_EXP

VEKTOR_SELEKTION ::= KOORDSYS_NAME . VEKTOR_SELEKTOR
| BEZIEHUNG_NAME . VEKTOR_SELEKTOR

VEKTOR_SELEKTOR ::= **ORIGIN** | **X_VECTOR**
| **Y_VECTOR** | **Z_VECTOR**

VEKTOR_OP ::= + | - | ×

VEKTOR_KONST ::= (REAL_KONST, REAL_KONST, REAL_KONST)$^{\mathbf{T}}$

FRAME_KOMP ::= VEKTOR_EXP
| KOORDSYS_NAME . KOMP_SELEKTOR

KOMP_SELEKTOR ::= **ORIGIN** | **X_AXIS** | **Y_AXIS** | **Z_AXIS**
| **XY_PLANE** | **XZ_PLANE** | **YZ_PLANE**

Produktionen für Realzeitconstraints

REALZEIT_CONSTR ::= **DURATION** VERGL_OP REAL_EXP **s**
| GESCHW_SPEZ VERGL_OP REAL_EXP {**cm/s** | **grad/s**}
| BESCHL_SPEZ VERGL_OP REAL_EXP {**cm/s²** | **grad/s²**}

GESCHW_SPEZ ::= **SPEED**
| **SPEED** (DOF_SELEKT)

BESCHL_SPEZ ::= **ACCELERATION**
| **ACCELERATION** (DOF_SELEKT)

Produktionen für kraftbezogene Constraints

KRAFT_CONSTR ::= **FORCE** (DOF_SELEKT) VERGL_OP REAL_EXP **N**
| **TORQUE** (DOF_SELEKT) VERGL_OP REAL_EXP **Nm**

TRACK_CONSTR ::= **TRACK** (DOF_SELEKT)

Allgemeine Produktionen

VERGL_OP ::= < | > | <> | =

REAL_KONST ::= [–]{ZI}$^+$[.{ZI}$^+$][**E**{ZI}$^+$]

TIME_EXP ::= **t** | **0** | **1**

REAL_VARIABLE ::= NAME

TERMINIER_GES_NAME ::= NAME

BEZIEHUNG_NAME ::= NAME

KOORDSYS_NAME ::= NAME

NAME ::= BU{BU | ZI | – | _}*

BU ::= **A** | **B** | **C** | ··· | **X** | **Y** | **Z** | **a** | **b** | **c** | ··· | **x** | **y** | **z**

ZI ::= **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

B Abstände zwischen Komponenten von Koordinatensystemen

Sei im folgenden $E_1 = (\vec{x}, \vec{n})$ eine in Punkt-Richtungsform beschriebene Ebene mit dem Ursprungsvektor $\vec{x}$ und dem normierten Normalenvektor $\vec{n}$. Sei $E_2 = (\vec{y}, \vec{m})$. Seien $G = \vec{u} + \lambda\vec{a}$ und $H = \vec{v} + \mu\vec{b}$ Geradendarstellungen ($\vec{a}$ und $\vec{b}$ normiert), $\vec{r}$ und $\vec{s}$ Ortsvektoren. Dann gilt für die Abstände bzw. Winkel zwischen diesen:

- Abstand Ebene – Ebene

$$d(E_1, E_2) = \begin{cases} |\vec{n} \circ (\vec{x} - \vec{y})| & \text{wenn } \vec{n} \times \vec{m} = 0 \quad (E_1 \parallel E_2) \\ 0 & \text{sonst} \end{cases}$$

- Abstand Ebene – Gerade

$$d(E_1, G) = \begin{cases} |\vec{n} \circ (\vec{u} - \vec{x})| & \text{wenn } \vec{a} \circ \vec{n} = 0 \quad (G \parallel E_1) \\ 0 & \text{sonst} \end{cases}$$

- Abstand Ebene – Punkt

$$d(E_1, \vec{r}) = |\vec{n} \circ (\vec{r} - \vec{x})|$$

- Abstand Gerade – Gerade

  Der Vektor $\vec{d} = \vec{a} \times \vec{b}$ steht auf beiden Geraden senkrecht, wenn diese nicht parallel liegen. Er ist somit Normalenvektor einer Ebene, die zu beiden Geraden parallel liegt. Somit ist nur noch der Abstand einer Geraden zu der Ebene, die durch $\vec{d}$ und einen Punkt der zweiten Geraden bestimmt ist, zu berechnen.

$$d(G, H) = \begin{cases} 0 & \text{wenn } \vec{a} \times \vec{b} = 0 \quad (G \parallel H) \\ |\frac{\vec{a} \times \vec{b}}{|\vec{a} \times \vec{b}|} \circ (\vec{u} - \vec{v})| & \text{sonst} \end{cases}$$

- Abstand Gerade – Punkt

$$d(G, \vec{r}) = |\vec{a} \times (\vec{u} - \vec{r})|$$

- Abstand Ursprung – Ursprung

$$d(\vec{r}, \vec{s}) = |\vec{r} - \vec{s}|$$

- Winkel Ebene – Ebene

$$\omega(E_1, E_2) = arcsin(|\vec{n} \times \vec{m}|)$$

- Winkel Ebene – Gerade

$$\omega(E_1, G) = arcsin(|\vec{n} \circ \vec{a}|)$$

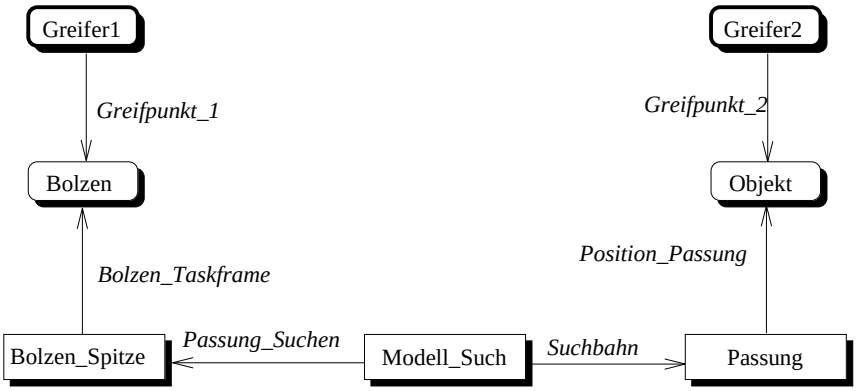
- Winkel Gerade – Gerade

$$\omega(G, H) = arcsin(|\vec{a} \times \vec{b}|)$$

C Beispiele für spezifizierte Aufgaben

Beispiel 1: Suchen einer Passung, in die ein Bolzen einzufügen ist. Der Bolzen wird dabei von einem Manipulator gehalten, das Werkstück, in das der Bolzen einzusetzen ist, von einem zweiten. Die Suche erfolgt durch spiralförmiges Tasten mit konstanter Kraft in Z-Richtung. Die spiralförmige Suchbahn ist durch die Bewegung eines Modellframes spezifiziert. Das Taskframe folgt dem Modellframe mit einer Geschwindigkeit von 3 mm/s und einem Anpreßdruck von 3 N. Der Suchvorgang ist beendet, wenn die Suchtrajektorie abgefahren wurde (erfolglos) oder wenn ein Moment in X- oder Y-Richtung darauf hindeutet, daß die Passung gefunden wurde.

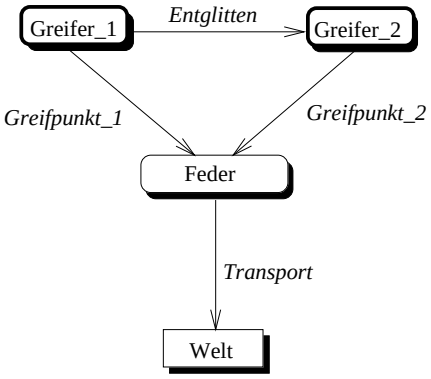
```
TASK: Passung_Suchen
ENVIRONMENT: Zwei_Pumas
```




```
RELATIONS:

Greifpunkt_1 : (0, 0, 100, 0, 0, 0)
Greifpunkt_2 : (200, 50, 100, 0, 0, 0)
Bolzen_Taskframe : (0, 0, -100, 0, 0, 0)
Suchbahn :
   LINEAR_INTERPOLATED :
      ( 0, 0, 0, 0, 0, 0 )
      ( 1, 0, 0, 0, 0, 0 )
      ( 1, 1, 0, 0, 0, 0 )
      ( -1, 1, 0, 0, 0, 0 )
      ( -1, -1, 0, 0, 0, 0 )
      ( 2, -1, 0, 0, 0, 0 )
      ( 2, 2, 0, 0, 0, 0 )
      ( -2, 2, 0, 0, 0, 0 )
      ( -2, 2, 0, 0, 0, 0 )
   END_INTERPOLATED
Passung_Suchen :
   START : (0, 0, 0, 0, 0, 0) AND
   SPEED = 0.3 cm/s AND
   TRACK ( TX ) AND
   TRACK ( TY ) AND
   FORCE ( TZ ) = 3 N AND
   TRACK ( RX ) AND
   TRACK ( RY ) AND
   TRACK ( RZ )
   TERMINATION :
      Eingerastet :
         sqrt (TORQUE ( RX ) · TORQUE ( RX )) > 10 Nm OR
         sqrt (TORQUE ( RX ) · TORQUE ( RX )) > 10 Nm

TERMINATION_CONDITIONS:
   NICHT_GEFUNDEN : T = 1
   GEFUNDEN : Eingerastet
```

Beispiel 2: Eine Feder soll von einer Startposition zu einer Zielposition transportiert werden. Die Feder soll während des Transports mit einer vorgegebenen Kraft gespannt werden. Die Aufgabe soll als fehlerhaft beendet werden, wenn die beiden Greifpunkte zu weit voneinander entfernt liegen, was auf einen Bruch bzw. ein Entgleiten der Feder hindeutet.

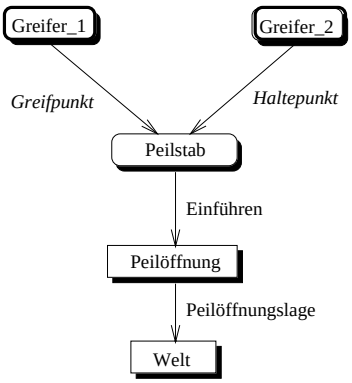
```
TASK: Transport_Feder
ENVIRONMENT: Zwei_Pumas
```




```
RELATIONS:

Greifpunkt_1 :
   START : (-200, 0, 100, 0, 0, 0) AND
      FORCE ( TX ) = - 50 N AND
      FORCE ( TY ) = 0 N AND
      FORCE ( TZ ) = 0 N AND
      TORQUE ( RX ) = 0 Nm AND
      TORQUE ( RY ) = 0 Nm AND
      TORQUE ( RZ ) = 0 Nm

Greifpunkt_2 :
   START : (200, 0, 100, 0, 0, 0) AND
      FORCE ( TX ) = 50 N AND
      FORCE ( TY ) = 0 N AND
      FORCE ( TZ ) = 0 N AND
      TORQUE ( RX ) = 0 Nm AND
      TORQUE ( RY ) = 0 Nm AND
      TORQUE ( RZ ) = 0 Nm

Transport :
   START : (100, -450, 100, 0, 0, 0) AND
   GOAL : (400, 800, 100, 0, 0, 0)
Ueberwachung :
   TERMINATION :
      Entglitten : DIST(Greifpunkt_1.ORIGIN, Greifpunkt_2.ORIGIN) > 500

TERMINATION_CONDITIONS :
   FERTIG : (T = 1)
   FEHLER : Entglitten
```

Beispiel 3: Ein flexibler Peilstab **PS** soll in einen Tank eingeführt werden. Um ein Einführen des Stabs, der wegen seiner Flexibilität zum Schwingen neigt, zu ermöglichen, wird er vom Greifer (**Greifer_2**) eines Manipulators an seiner Spitze so gegriffen, daß dessen Greifpunkt entlang des Stabs verschiebbar ist. Der Stab soll nun bis zu einem Sicherheitsabstand sa über der Peilöffnung **P** des Tanks an der Spitze gehalten werden, ab dann soll sich der Greifpunkt entlang des Stabs so verschieben, daß der Sicherheitsabstand eingehalten wird.

```
TASK: Peilstab_einführen
ENVIRONMENT: Zwei_Pumas
```




```
RELATIONS:

Greifpunkt :
   FIXED : (0, 0, 100, 90, 0, 0)
   TERMINATION :
      Blockiert_1 : FORCE ( TZ ) < -10 N
      Verbogen_1 :
         SQRT(TORQUE(RY) · TORQUE(RY) + TORQUE(RX) · TORQUE(RX)) < 3 Nm
Haltepunkt :
   START : (0, 0, -100, 90, 0, 180) AND
   TX ( t ) = 0 AND
   TY ( t ) = 0 AND
   RX ( t ) = 90 AND
   RY ( t ) = 0 AND
   RZ ( t ) = 180
   TERMINATION :
      Blockiert_2 : FORCE ( TZ ) < -10 N
      Verbogen_2 :
         SQRT(TORQUE(RY) · TORQUE(RY) + TORQUE(RX) · TORQUE(RX)) < 3 Nm
Peilöffnungslage :
   FIXED : (100, 80, 100, 0, 0, 0)
Einführen :
   LINEAR_INTERPOLATED :
      ( 0, 0, 100, 0, 0, 0 )
      ( 0, 0, -80, 0, 0, 0 )
   END_INTERPOLATED

GLOBAL_CONSTRAINT :
   ((Haltepunkt.TZ(t) - Einführen.TZ(t) > sa) AND (Haltepunkt.TZ(t) = -100)) OR
   ( Haltepunkt.TZ(t) - Einführen.TZ(t) = sa)

TERMINATION_CONDITIONS :
   FERTIG : (T = 1)
   FEHLER : Blockiert_1 OR Verbogen_1 OR Blockiert_2 OR Verbogen_2
```

D Rotatorische Gleitschritte

Entsprechend den translativen Gleitschritten stellt sich bei rotatorischen Gleitschritten die Frage nach dem maximalen Drehwinkel. Da es hier tatsächlich nur auf die Winkel ankommt, wird das Ursprungskoordinatensystem so gedreht, daß die Drehung von q nach q_m um die $x - Achse$, die Drehung von q_m nach q'' entlang der y-Achse erfolgt. Für die Drehwinkel ergibt dies keinen Unterschied, es erleichtert aber im folgenden die Rechnung erheblich.

Die Situation wird in Abbildung 47 veranschaulicht. Es bezeichnet β den Drehwinkel von q nach q_m (um die x-Achse), γ den Drehwinkel von q_m nach q'' (um die y-Achse), σ' bzw. σ'' den Drehwinkel von q nach q' bzw. q'' und δ bzw. δ' den Drehwinkel von q bzw. q'' nach $Ziel$.

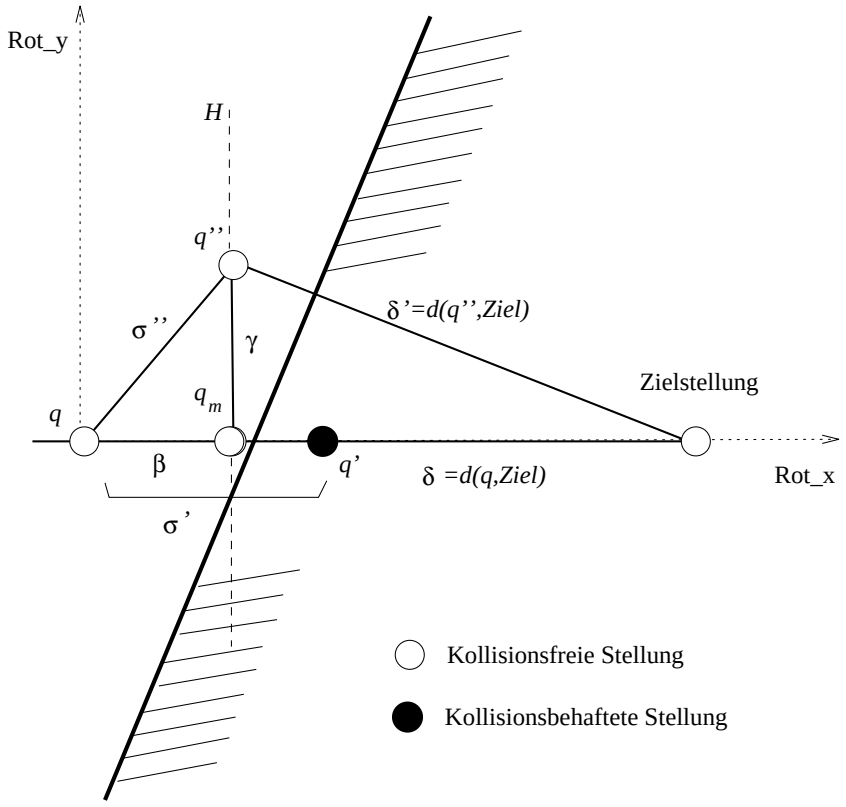


Abbildung 47: Rotatorische Gleitschritte

Wieder ergeben sich zwei Bedingungen:

1. Anders als bei translatorischen Gleitschritten, bei denen die maximale Translation direkt durch die Schutzschichtdicke bestimmt ist, ist im Fall rotatorischer Gleitschritte der maximale Drehwinkel abhängig von der Drehachse, da der am weitesten von der Drehachse entfernte Punkt sich nicht weiter als d_{schutz} bewegen darf. Daher wird als Näherungswert der Drehwinkel σ' des letzten in die Kollision führenden Schritts

verwendet und im Anschluß an die Bestimmung des rotatorischen Gleitschritts eine Skalierung des Schritts durchgeführt, sodaß die Schutzschichtbedingung erfüllt ist. Der Drehwinkel σ'' von q nach q'' sollte also nicht größer sein als der Drehwinkel σ' von q nach q'. Aus der Drehmatrix M von q nach q'' läßt sich σ'' bestimmen[1]:

$$M = \begin{pmatrix} \cos(\gamma) & \sin(\beta)\sin(\gamma) & \cos(\beta)\sin(\gamma) \\ 0 & \cos(\beta) & -\sin(\beta) \\ -\sin(\gamma) & \sin(\beta)\cos(\gamma) & \cos(\beta)\cos(\gamma) \end{pmatrix}$$

$$cos(\sigma'') = \frac{\cos(\gamma) + \cos(\beta) + \cos(\gamma)\cos(\beta)}{2}$$

$$\gamma = \arccos \frac{2\cos(\sigma'') - \cos(\beta)}{1 + \cos(\beta)}$$

$$\gamma \leq \arccos \frac{2\cos(\sigma') - \cos(\beta)}{1 + \cos(\beta)} \qquad (\text{da} \quad \sigma'' \leq \sigma')$$

2. Der Rotationswinkel zum Ziel nach Ausführung des Gleitschrittes $\delta' = d(q'', Ziel)$ darf nicht größer sein als der Rotationswinkel $\delta = d(q, Ziel)$ vorher. Wir setzen an $\delta' = \delta$ und erhalten

$$R(\mathrm{x}, \delta) = R(\vec{n}, \delta) \cdot R(\mathrm{y}, \gamma) \cdot R(\mathrm{x}, \beta)$$

Dabei bezeichnet $R(\vec{m}, \alpha)$ eine Rotation um die Achse $\vec{m}$ mit Winkel α. Die obige Gleichung läßt sich durch Multiplikation mit $R^{-1}(\mathrm{x}, \beta) \cdot R^{-1}(\mathrm{y}, \gamma) = R(\mathrm{x}, -\beta) \cdot R(\mathrm{y}, -\gamma)$ überführen in

$$R(\mathrm{x}, \delta) \cdot R(\mathrm{x}, -\beta) \cdot R(\mathrm{y}, -\gamma) = R(\mathrm{x}, \delta - \beta) \cdot R(\mathrm{y}, -\gamma) = R(\vec{n}, \delta)$$

oder in expliziter Darstellung mit $\alpha := \delta - \beta$, $C = \cos\delta$, $S = \sin\delta$ und $V = 1 - C$:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \cdot \begin{pmatrix} \cos\gamma & 0 & \sin\gamma \\ 0 & 1 & 0 \\ -\sin\gamma & 0 & \cos\gamma \end{pmatrix} = R(\vec{n}, \delta)$$

$$\begin{pmatrix} \boxed{\cos\gamma} & 0 & \sin\gamma \\ \sin\alpha\sin\gamma & \boxed{\cos\alpha} & -\sin\alpha\cos\gamma \\ -\cos\alpha\sin\gamma & \sin\alpha & \boxed{\cos\alpha\cos\gamma} \end{pmatrix} =$$

[1] Die Rechnung macht sich beim Gebrauch des *cos* und arccos die Tatsache zunutze, daß $|\sigma'|$ und $|\sigma''|$ sicherlich kleiner als $\frac{\pi}{2}$ sind.

$$= \begin{pmatrix} \boxed{n_x^2 V + C} & n_x n_y V + n_z S & n_x n_z V - n_y S \\ n_x n_y V - n_z S & \boxed{n_y^2 V + C} & n_y n_z V + n_x S \\ n_x n_z V + n_y S & n_y n_z V - n_x S & \boxed{n_z^2 V + C} \end{pmatrix}$$

Die eingerahmten Matrixelemente führen auf drei Gleichungen:

$$\begin{aligned} \cos\gamma &= n_x^2 V + C \\ \cos\alpha &= n_y^2 V + C \\ \cos\alpha\cos\gamma &= n_z^2 V + C \end{aligned}$$

Summiert man die drei Gleichungen, so erhält man:

$$\cos\gamma + \cos\alpha + \cos\gamma\cos\alpha = \underbrace{(n_x^2 + n_y^2 + n_z^2)}_{=1} V + 3C = 2C - 1$$

$$\cos\gamma = \frac{2\cos\delta + 1 - \cos\alpha}{1 + \cos\alpha}$$

$$\gamma = \arccos\frac{2\cos\delta + 1 - \cos\alpha}{1 + \cos\alpha}$$

Wir erhalten damit für den Fall, daß die rotatorischen Freiheitsgrade für den zu erreichenden Zielraum spezifiziert sind, mit γ eine weiter obere Schranke für den Rotationswinkel beim auszuführenden Gleitschritt.