Theoretische Grundlagen
und
Algorithmen
für die Positionierung eines Multirobotersystems
zur Wartung von Hochspannungfreileitungen

Autor
Christian Müller

e-mail: christian.mueller@ifib.uni-karlsruhe.de

Karlsruhe, November 1997

Abbildungsverzeichnis

0. Zusammenfassung

Ziel der Betrachtungen der vorliegenden Arbeit ist die Bestimmung theoretischer Grundlagen und Algorithmen, um eine geeignete Positionierung eines Multirobotersystems zur Wartung von Hochspannungsfreileitungen zu bestimmen.
Da es sich um ein System mit zwei Robotern handelt, analysiert man zunächst die relative Aufstellung beider Roboter zueinander. Diese wird nicht wie allgemein üblich experimentell ermittelt, sondern man geht hier den Weg einer analytischen Formulierung einiger Gesichtspunkte, die für die koordinierte Ausführung der Arbeiten relevant sind. Schlüsseldaten zur Lösung des Problems sind die Beschreibung des Arbeitsraumes und ihre qualitative und quantitative Interpretation. Anfangs bestimmt man anhand allgemeiner Vorüberlegungen, spezieller Voraussetzungen und geometrischer Betrachtungen, welches die Aufstellung charakterisierenden variablen Größen sind. Durch die mathematische Formulierung von vier Bewertungsgrößen und ihrer anschließenden Normierung und Zusammenfassung erhält man ein Gütemaß, von dem ausgehend eine quantitative Aussage über die Güte der Aufstellung möglich ist. Diese Methode eignet sich durch ihre schnelle und exakte Findung einer für die gegebenen Parameter guten Aufstellung auch besonders als Entscheidungshilfe für den Einsatz vor Ort, um das System an die gestellte Aufgabe anzupassen.
Das Robotersystem ist auf der Plattform eines Mobilkrans montiert, die zur Durchführung der Arbeiten entsprechend vor dem Mast wird. Im zweiten Teil der vorliegenden Diplomarbeit entwickelt man einen Algorithmus zur Suche der besten Positionierung der Plattform. Aufgrund der Vielzahl der zu bearbeitenden Objekte und der starken Einschränkung der jeweiligen Arbeitsräume ist die manuelle Auswahl einer Plazierung unmöglich.
Wieder ist die Beschreibung des Arbeitsraumes der Schlüssel zur Lösung des Problems. Allerdings reicht die vom Hersteller gegebene Arbeitsraumbeschreibung hier nicht aus, da verschiedene Greiforientierungen für die Objekte zu berücksichtigen sind. Somit muß eine neue Strategie zur volumenorientierten, orientierungsabhängigen Beschreibung der Arbeitsräume entwickelt werden. Der eigentliche Algorithmus beruht auf einer sequentiellen Suche nach allen gültigen Plazierungen, d.h. solchen, von denen aus die Ausführung der Aufgabe

(Manipulation aller definierten Objekte) möglich ist. Hat man alle diese gefunden, so muß man aus dieser Menge die guten Positionierungen bestimmen. Hier werden, ähnlich wie im ersten Teil, Kriterien formuliert, die die Berechnung der Güte jeder einzelnen Plazierung ermöglichen. Hierzu versucht man Erfahrungswerte aus der bisherigen manuellen Arbeit zu nutzen und entsprechend aufzubereiten.

Ein Problem bei einer dreidimensionalen sequentiellen Suche ist das starke Anwachsen der zu überprüfenden Positionierungen und damit auch der Suchzeit. Abgesehen von einer diskreten Beschreibung der Arbeitsräume bieten geeignete Anfangsplazierungen und ein Ausschließen von Raumbereichen, die zu keiner Lösung führen können, ein großes Potential zur Zeitersparnis. Weiterhin werden die Eingaben des Bedieners erst einem Plausibilitätstest unterworfen, der über sukzessive Verfeinerungen der Überprüfung nicht sinnvolle Aufgabendefinitionen zum frühestmöglichen Zeitpunkt erkennt. In diesem Falle muß man erst gar nicht in den zeitintensiven Algorithmus eintreten und kann sofort eine neue Aufgabendefinition vom Bediener anfordern.

Bei der Lösung der Aufgabenstellung konnte auf keine schon existierenden Methoden zurückgegriffen werden, und aufgrund des Umfangs der Arbeit und der zeitlichen Begrenzung war eine komplette Programmierung der Algorithmen nicht möglich. Dennoch wurden alle wichtigen Teile des Algorithmus jeweils exemplarisch anhand kleiner (meist zweidimensionaler) Beispiele verifiziert. Alle mathematischen Berechnungen und die Programmierung der Beispiele wurden mit Hilfe von MAPLE durchgeführt. Für die eigentliche Robotersimulation konnte auf das am Institut entwickeltes Simulationsprogramm TOROS [9] zurückgegriffen werden.

Keywords:

Telemanipulation, Multirobot, Workspace, Coordination, Automatic Positioning

1. Einleitung

In modernen Industriestaaten ist bezüglich des Gesamtenergiebedarfs von einem stetig wachsenden Anteil an elektrischer Energie auszugehen. Die ausreichende Versorgung von Fabriken, Computerzentren, Kommunikations-einrichtungen, öffentlichem Verkehr und nicht zuletzt der privaten Haushalte mit Strom ist durch ein hochentwickeltes System an Energieerzeugung und -verteilung gewährleistet. Da komplexe Produktionsprozesse nicht einfach ab- und wieder angestellt werden können, Rechnerstillzeiten und nicht funktionierende Kommunikation enorme Kosten verursachen, ist zudem auch besonders auf die *kontinuierliche* Versorgung Wert zu legen. Dies bedeutet, daß Arbeiten an Versorgungseinrichtungen, wenn möglich unter Spannung ausgeführt werden. Hierfür existieren heutzutage schon Methoden und Werkzeuge wie z.B. die Installation eines Bypass, das Arbeiten mit isolieren Stangen, isolierte Hebebühnen. All diese Arbeiten werden jedoch noch überwiegend von Hand ausgeführt, sind gefährlich und zeitintensiv, u.a. auch bedingt durch den erhöhten Sicherheitsaufwand.

Unter diesem Aspekt riefen IBERDROLA und DISAM ein Forschungsprojekt namens ROBTET ins Leben, dessen Ziel die Entwicklung eines marktreifen Robotersystems ist, das Wartungs- und Reparaturarbeiten an elektrischen Versorgungsleitungen bis 20kV durchführen kann. Ähnliche Systeme werden auch in den USA und in Japan entwickelt.

Dortige Untersuchungen haben ergeben, daß man die Zeiten zur Ausführung bestimmter Arbeiten an Hochspannungsleitungen unter Zuhilfenahme eines roboterisierten Systems drastisch verringern kann.

Um die Entwicklungszeit möglichst effektiv zu nutzen und möglichst schnell ein einsetzbares System zur Verfügung zu haben, wurde ROBTET zeitlich und logisch so gegliedert, daß nach jeder Entwicklungsstufe das schon vorhandene System um eine weitere Komponente ergänzt wird. Die Gliederung des Projektes wird im folgenden kurz erläutert, um die vorliegende Arbeit entsprechend in den Gesamtkontext einordnen zu können.

Am Anfang des Projektes steht die Festlegung, die Auswahl und die anschließende Integration der Einzelkomponenten zu einem funktionierenden

System *(Systemintegration)*. Es beinhaltet schon die komplette Hardware (Mobilkran, Roboter, Versorgungseinrichtungen, …) und ist sofort einsetzbar, wenn auch nicht im vollem Funktionsumfang.

Im Unterschied zu den amerikanischen und japanischen Systemen, die an bedienerunterstützenden Einrichtungen lediglich Kamera und Monitor bieten, soll hier ein Multimediasystem als hochentwickeltes *Mensch-Maschine-Interface* geschaffen werden. Hierbei wird die Umgebung mittels Sensoren und Kameras erfaßt und im Rechner in geeigneter Form modelliert. Mit diesen Daten und ihrer intelligenten graphischen Darstellung kann dem Bediener umfangreiche Unterstützung bei der Steuerung der Roboter gegeben werden *(Telepräsenz)*.

Die Weiterentwicklung und Ergänzung der Telepräsenz in Richtung einer aktiven und intelligenten Hilfestellung läßt sich als *kontrollierte Teleoperation* bezeichnen. Es wird hier beispielsweise überprüft, ob der Bediener mit den Roboterarmen Punkte verschiedenen Potentialen verbindet oder ob eine Kollision mit Objekten der Umgebung erfolgen würde. Zur Vermeidung solcher Kollisionen oder zur Empfehlung von Zugriffen auf Objekte nutzt man spezielle Algorithmen aus den Bereichen Bahnplanung und Kollisionsüberprüfung.

Aktionen, die immer wieder in einer standardisierbaren Reihenfolge auftreten (Routinearbeiten), sollen unabhängig vom Bediener in einer vorher festgelegten Art und Weise selbst vom System ausgeführt werden *(Automatisierung der Operationen)*. Der Bediener kann währenddessen schon Vorbereitungen für die nächste Aufgabe treffen. Da es sich um ein Multirobotersystem handelt, muß zusätzlich die koordinierte Manipulation der beiden Roboter implementiert werden.

Bisher wurden hardwareseitig nur Standardkomponenten verwendet, die zwar sehr flexibel und vielseitig einsetzbar sind, aber oft einen entscheidenden Mehraufwand an Programmierung und Planung erforderten (sequentielle Zerlegung komplexer Aktionen). Diese Manipulationen können durch die *Entwicklung spezieller Werkzeuge*, einschließlich eines automatischen Werkzeugwechslers, durch optimale Anpassung und Parallelisierung erheblich vereinfacht werden. Neben diesen Hardwareerweiterungen werden zusätzlich softwareseitig Applikationen zur Lösung spezifischer Probleme, die bei der roboterisierten Ausführung der Aufgabe auftreten, entwickelt.

Der letzte zeitlichen Abschnitt des Projektes sieht die Entwicklung zusätzlicher Hardware mit den entsprechenden Schnittstellen zum bisherigen System vor, welche bei Bedarf als Erweiterung dem System hinzugefügt wird *(Zubehör)*.

2. Aufgabenstellung

Die vorliegende Diplomarbeit gliedert sich in zwei Teile, die man in die Gliederungsabschnitte *Systemintegration* und *Entwicklung von speziellen Werkzeugen und Applikationen* einordnen kann. Das die beiden Teilaufgaben verbindende Element stellt die Betrachtung und Formulierung des 'Workspace' des Roboters und die spätere Auswertung und Nutzung der gewonnenen Erkenntnisse dar.

Im Gegensatz zu einzelnen freistehenden Robotern ist bei einem kooperierenden Multirobotersystem die relative Aufstellung zueinander kein triviales Problem mehr. Bei einem Einzelrobotersystem kann man jeden Roboter getrennt in seiner Umgebung betrachten und muß höchstens Kollisionsfreiheit mit dem benachbarten System gewährleisten. Hier aber werden beide Roboter durch den für die koordinierten Manipulationen so wichtigen gemeinsamen Arbeitsraum räumlich einander zugeordnet. Aufgabe war es nun, unter Berücksichtigung der spezifischen Anforderungen und der Systemstruktur, die *beste relative Aufstellung* zu finden. Hier wurde bewußt der analytischen Formulierung des Problems gegenüber der experimentellen Suche nach der optimalen relativen Positionierung der Vorzug gegeben, um eine später schnell modifizierbare und allgemeingültige Aussage zu bekommen.

Ausgehend von ständig wechselnden Szenarien, bedingt durch die Vielzahl von Masten und auszuführenden Arbeiten, kann man keine allgemeingültige zufriedenstellende *Positionierung des Multirobotersystems* bezüglich des Mastes angeben. Durch die weite räumliche Verteilung der Objekte wird die manuelle Positionierung aufgrund der Dreidimensionalität schwierig. Eine zeitsparende Ausführung der Arbeiten ist, auch im Hinblick auf die automatische Ausführung mit der zeitintensiven Festlegung des *Baseframes* , nur bei wenigen Positionswechseln möglich. Ziel der Betrachtungen ist es deshalb, Struktur und Algorithmen für eine Software zu entwickeln, die eine schnelle Positionierung unter Berücksichtigung der zu manipulierenden Objekte und spezieller Parameter wie Mastgeometrie, Sicherheitsabstand usw. gewährleistet. Der Einsatz dieser Software soll direkt vor Ort, sowie auch a priori als Planungswerkzeug unter Zuhilfenahme eines Robotersimulationssystem erfolgen. Deshalb schließt die Aufgabenstellung auch

noch die Modellierung des Systems in der Robotersimulationssoftware TOROS [9] ein.

Bei keiner der beiden Aufgabenstellungen konnte auf schon existierende Standardlösungen zurückgegriffen werden. Die Beschreibung und Formulierung der Arbeitsräume von Robotern ist bisher nicht generell gültig für alle Anwendungen angebbar. Das Studium vorhandener Literatur gestaltete sich als schwierig, da jeder der Autoren bezüglich seiner speziellen Aufgabenstellung erhebliche Einschränkungen der Betrachtungen macht. Deshalb wurden für beide Aufgabenstellungen eigene Lösungsansätze entwickelt. Aufgrund des zeitlich beschränkten Rahmens dieser Diplomarbeit, konnte die Software nicht mehr als Ganzes implementiert werden. Dennoch wurden wichtige Ansätze und Algorithmen anhand kleiner Beispiele in der Programmiersprache MAPLE verifiziert und visualisiert.

3. Systemstruktur

3.1. Aufgaben / Anforderungen

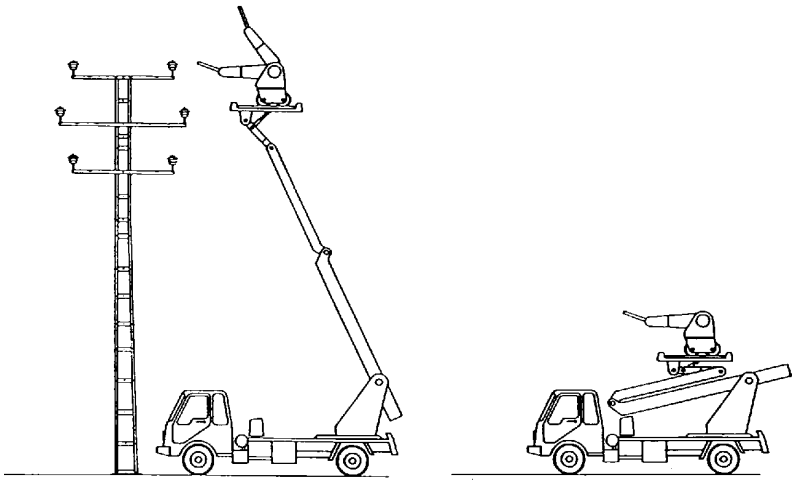
Dieses System soll speziell für Wartung, Überwachung und Reparatur von elektrischen Versorgungsleitungen eingesetzt werden. Vorgesehen ist der Einsatz hauptsächlich in Spanien, dessen spezielle Struktur des Versorgungsnetzes den Einsatz besonders einfach und wirtschaftlich erscheinen läßt. Geringe Höhe der Masten und relativ niedrige Spannungen ergeben Einschränkungen an das System, die die Komplexität in Bezug auf Isolierung und Sicherheitsmaßnahmen erniedrigen.
Dennoch ist aufgrund der Vielzahl unterschiedlicher Formen der Strommasten und der auch daraus resultierenden nötigen Arbeitsschritten die Spezifikation nicht geschlossen in wenigen Kriterien festschreibbar. Man kann weder alle Arten von Schäden an den Versorungsleitungen oder Masten vorhersagen noch alle Einzelheiten der Ausführung im Voraus planen. Man muß also versuchen, die Systemintegration möglichst flexibel zu gestalten, um gegebenfalls später durch Simulation oder durch Erfahrungswerte möglichst viele Freiheitsgrade zur Anpassung des System zur Verfügung zu haben.

3.2. Systematisierung der auszuführenden Operationen

In der ersten Phase des Projektes wurden die Vorgänge bei Arbeiten unter Spannung untersucht und eine hierarchische Strukturierung vorgenommen. Die Einteilung erfolgte nach Verfahren, die sich auf die zu bearbeitenden Elemente (Isolierung, Ableitungen, Verbinder, ...) bezog. Diese wurden wiederum in einfache spezifische Aktionen (verbinden, trennen, zapfen, ...) und die dazugehörigen Objekte (Isolierung, Kabel, Klemmen, ...) zerlegt. Wegen der noch vorhandenen Komplexität einzelner Aktionen erfolgte schließlich noch eine Einteilung in *Primitive* d.h. ganz einfache Befehle in der Programmiersprache des Robotersystems, die direkt vom System ausgeführt werden können (greifen, bewegen, drehen, ...).

3.3. Komponenten des Systems

Zur ersten Definition der Komponenten orientierte man sich an der zuvor erfolgten Systematisierung und legte aufgrund der einzelnen auszuführenden Aktionen die vorläufige Konfiguration fest. Diese sieht einen Mobilkran mit zwei oben auf einer Bühne angebrachten Robotern vor, die eine geringe Tragkraft aber eine hohe Positioniergenauigkeit haben sollten. Für schwere Lasten gibt es zusätzlich einen Arm oder ein zweiter Mobilkran mit großer Reichweite und Tragkraft. Das ganze soll mit einem entsprechenden Sensorsystem und einer zentralen Steuerung ausgestattet werden, die sowohl Teleoperation, als auch automatische Operationen und Überwachungsaufgaben zuläßt. Die folgende Abbildung zeigt eine grobe Skizzierung des Systems in aus- und eingefahrenem Zustand.



Skizze des Gesamtsystems

3.4. Zweiarm-System

3.4.1. Gründe für den Einsatz eines Zweiarm-Systems

Einarm-Robotersysteme haben sich in der Praxis im industriellen Bereich längst etabliert. Man setzt sie hauptsächlich im der Produktion für Schweiß-, Montage-, oder Bestückungsarbeiten ein. Fast alle am Markt erhältlichen Modelle weisen einen sehr hohen technischen Stand an Genauigkeit, Schnelligkeit und Programmierkomfort auf. Leider bleiben solchen Einarm-Systemen Einsatzgebiete verschlossen, die auch der Mensch nur mit Hilfe beider Arme lösen kann. Deshalb werden in dem Gebiet der industriellen Automatisierung, aber auch in der Weltraumforschung und in der Nukleartechnik immer häufiger Mehrarm-Systeme entwickelt und eingesetzt.

3.4.1.1. Umgreifproblematik

Ein Einarm-System kann ein aufgenommenes Objekt nicht ohne die Hilfe einer zusätzlichen Einrichtung umgreifen. Dabei kann diese Vorrichtung von einer einfachen Halterung bis hin zur komplexen Umsetzeinheit alles darstellen. Dies erfordert eine stark angepaßte und damit spezielle Systemumgebung des Roboters, die in unserem Falle bei der Vielzahl der zu manipulierenden Objekte und der knappen Platzverhältnisse nicht installiert werden kann. Eine einfachere und elegantere Art des Umgreifens an Objekten läßt sich mit Hilfe von zwei Armen bewältigen, bei dem die eine Hand das Objekt in nahezu beliebig viele Positionen bringen kann und sich somit für die andere Hand eine optimale Greiferposition ergibt.

3.4.1.2. Komlexere Bewegungen

Aufgrund der bei schweren Objekten auftretenden großen Kräften auf Gelenke und Greifer des Roboters sind bei Einarmrobotern mit solchen Objekten nur sehr eingeschränkte Manipulationen möglich. Mit zwei Armen dagegen kann der eine Greifer auch noch stützend Hilfestellung bei der Manipulation des Objekts geben. Desweiteren sind Aufgaben wie fügen, gegeneinander verdrehen und auseinanderziehen für diese Konfiguration, natürlich immer unter Berücksichtigung der Genauigkeit, kein Problem.

3.4.1.3. Größerer erreichbarer Arbeitsraum

Obwohl dieser Punkt eigentlich jedem einleuchtend erscheint, muß man hier bedenken, daß die Möglichkeit zur gegenseitigen Behinderung besteht, was unter Umständen den Arbeitsraum des einzelnen Roboters stark eingrenzen kann. Dennoch läßt sich der Arbeitsraum bei entsprechender relativer Lage der beiden Roboter zueinander entscheidend vergrößern und auch die maximale Reichweite erhöht sich in gewisse Richtungen enorm.

3.4.1.4. Nähe zur menschlichen Arbeitsweise

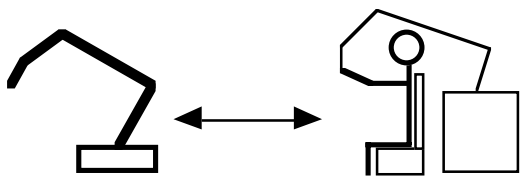
Bei der Entwicklung von einfachen Manipulatoren hin zu komplexen Robotern, orientiert man sich gerne am menschlichen Körper, der an Beweglichkeit und Flexibilität, betreffend die Manipulation von Objekten, den Maßstab schlechthin setzt. Ein weiterer entscheidender Punkt ist aber auch die Bedienung dieser Manipulatoren, die, je mehr sich ein System am Mensch orientiert, desto schneller, leichter und präziser erlernt werden kann.

3.5. Telerobotiksystem

Gerade in schwierigen, unzugänglichen Umgebungen, sowie dort wo sich direkte Risiken für den Bediener ergeben, leisten telemanipulierte Robotersysteme schon seit geraumer Zeit wichtige Hilfestellung zur Ausführung spezieller Aufgaben. Hier sind besonders Einsatzgebiete im Weltraum, in der Nukleartechnik und Unterwassereinsätze zu nennen. Dabei ist es sinnvoll, eine Einteilung der Telerobotersysteme vorzunehmen, um die Entwicklungsstufen und die verschiedenen Systemkomponenten zu verdeutlichen [4].

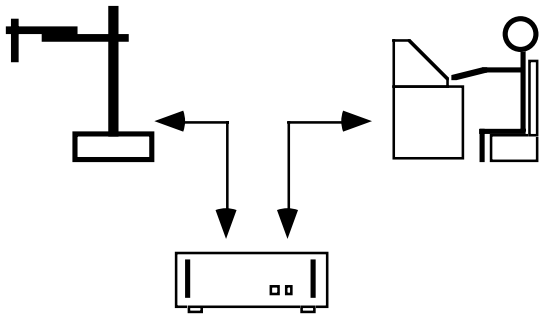
Telepräsenz

Der Mensch steuert den Manipulator *(slave)* mit einem auf seine Größe angepaßten Nachbau desselben *(master)* **vollständig**. Über die **Rückführung** von **visuellen** und **taktilen Signalen** hat der Bediener das Gefühl, die Aufgabe direkt auszuführen.



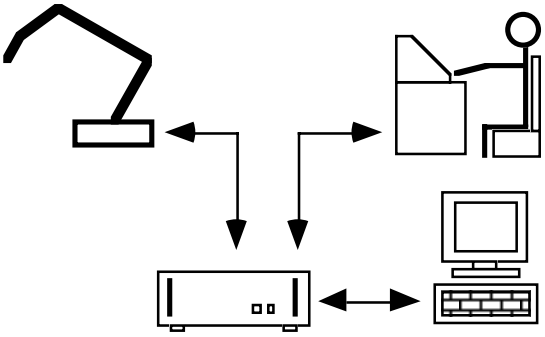
Teleoperation

Der Mensch übernimmt hier ebenfalls **vollständig** die Steuerung. Die Manipulatoren sind nicht mehr menschenähnlich gestaltet, und auch die Bedieneinrichtung muß keine Ähnlichkeit zum Manipulator aufweisen. Die notwendigen **Koordinatentransformationen** werden von einem Prozeßrechnersystem ausgeführt. Zudem besitzt dieses System die Möglichkeit Erfahrungswerte abzulegen, die später für gleichartige Aufgaben wieder abgerufen werden können.



Teleroboter

Der Mensch steuert hier weiterhin den Manipulator im Rahmen der ihm gestellten Aufgabe. Im **Robotersteuerungsmodus** können jedoch kleinere 'primitive' Aufgaben schon selbständig durchgeführt werden. Nach Beendigung der automatischen Arbeitsphase kehrt das System wieder in den **Teleoperationsmodus** zurück. Wegen des hybriden Charakters werden diese Systeme als Teleroboter bezeichnet.



Die genaue Zuordnung zu einem dieser drei Klassen ist heute kaum mehr exakt möglich. Fast jedes handelsübliche teleoperiertes System stellt eine Mischform aus zwei oder drei dieser Grundtypen dar.

3.5.1. Gründe für den Einsatz eines Teleroboters

Im folgenden werden einige der Gründe aufgeführt, wieso hier kein vollautomatisches System (Industrieroboter) sondern einen Teleroboter eingesetzt wird, der aber die Option für eine spätere Automatisierung offenhält.

3.5.1.1. Ständig wechselnde Aufgaben/Umgebungen

Aufgrund der Vielzahl von unterschiedlichen Arbeitsumgebungen und Aufgabenstellungen ist es nur unter sehr großem Zeit-, und Programmieraufwand möglich, einzelne Tasks d.h. programmierte Bewegungsabläufe im voraus zu planen.

3.5.1.2. Reaktion auf unvorhergesehene Ereignisse

Ein Ziel der Entwicklung des Systems ist auch, Wartungs- und Reparaturarbeiten an Tagen auszuführen, die witterungsbedingt für die 'Handmontage' nicht geeignet sind. Starker Wind zum Beispiel setzt die Leitungen in Bewegung und dies stellt den Roboter vor eine schier unlösbare Aufgabe, da die Umgebung nicht mehr mit seinem eingespeicherten Weltbild übereinstimmt.

3.5.1.3. Schwierige Festlegung des World-Base-Frames

Beide Roboter werden auf eine Plattform montiert, die von einem Mobilkran an die erforderliche Position gebracht wird. Aufgrund der geringen Präzision und der ebenfalls vorhandenen Möglichkeit des Schwingens des Kranarms, ist eine einmalige genaue Positionierung relativ zum zu bearbeitenden Objekt fast unmöglich. Nötig wäre hier die ständige Anpassung des Umweltmodells an die reale Umgebung, was außer aufwendiger Sensortechnik auch noch die Simulation der Tasks für die neue Aufgabenbeschreibung erforderte. Diese Simulation würde den Zeitvorteil, das als feste Anforderung an dieses System gestellt wird, wieder zunichte machen und somit die Wirtschaftlichkeit in Frage stellen. Denkbar ist noch die Möglichkeit der festen Verankerung des Kranarms am Strommasten, wobei hier wieder zusätzlicher Aufwand für Isolierung und Mechanik zu erbringen wäre.

3.5.1.4. Vollautomatisches System ist teuer und anfällig

Damit die Roboter ihre Aufgaben vollständig automatisch ausführen, müßte man ein sehr aufwendiges und umgebungsunabhängiges Sensorsystem entwickeln. Bildverarbeitungsysteme unterliegen aber heute noch zu großen Einschränkungen, denn z.B. Feuchtigkeit und Schmutz auf Linsen sowie Schatten durch sich ständig ändernde Beleuchtung sind Faktoren, die eine korrekte Objekterkennung sehr erschweren wenn nicht sogar verhindern. Abgesehen von Lasermeßsystemen, die auch heute schon z.B. im Baubetrieb eingesetzt werden, ist für alle anderen die EMV- Verträglichkeit und Robustheit nur schwer zu realisieren.

Zu guter Letzt muß man immer im Auge behalten, daß die Robotersteuerung, der Prozeßrechner zur Bildverarbeitung und die ganze dazu notwendige Peripherie immer mitgeführt und auch gegen Witterungseinflüsse, wie Feuchtigkeit und Schmutz, gesichert werden muß.

All dies erhöht die Komplexität des Systems beträchtlich und bei Ausfall einer Komponente wird das Gesamtsystem in seiner Funktion sehr eingeschränkt, wenn nicht sogar lahmgelegt.

3.5.1.5. Leichte Bedien- und Erlernbarkeit des Systems

Da Personalkosten einen immer größer werdenden Anteil an den Gesamtkosten ausmachen, ist es für den wirtschaftlichen Einsatz dieses Systems wichtig, daß das System nicht nur von hochbezahlten Spezialisten, sondern auch von den schon jetzt damit betreuten Mitarbeitern bedient werden kann. So können lange und kostenintensive Schulungs- und Einarbeitungsphasen einspart werden.
Außerdem gilt es zu bedenken, daß heutzutage jede Entscheidung über die Automatisierung einer Tätigkeit, die bisher 'von Hand', also von Menschen, durchgeführt worden ist, nicht nur eine finanzielle und technologische Aspekte, sondern auch eine soziale einbeziehen muß.

3.6. Eingesetztes System

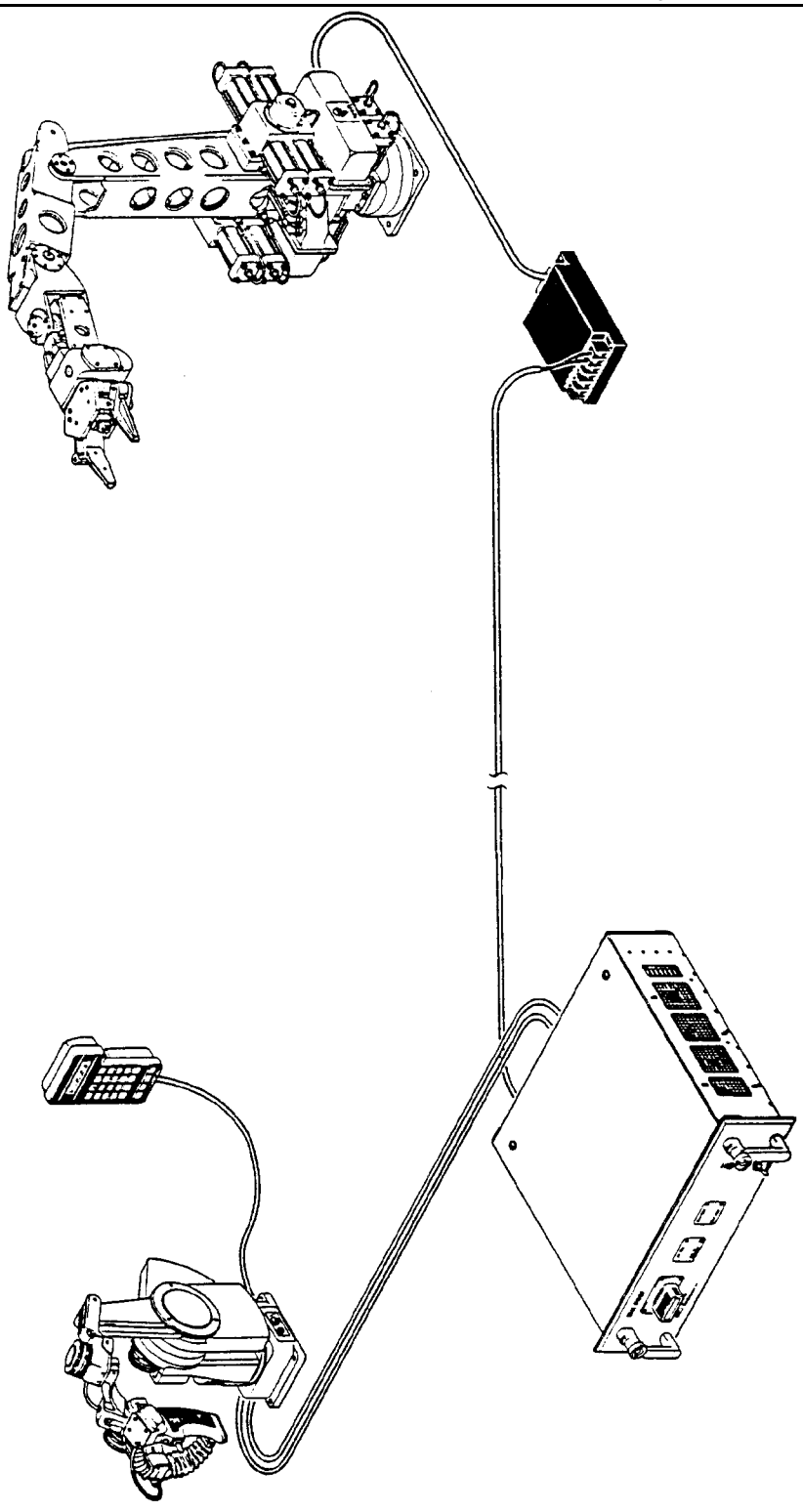
3.6.1. GRIPS von Kraft Telerobotics

Das GRIPS Telerobotiksystem läßt ich wie oben schon angedeutet, nicht genau in eine der vorgestellten Gruppen einordnen. Das System besteht aus einem elektrohydraulisch angetriebenen Roboterarm *(slave)*, mit großer Tragkraft, Genauigkeit und Schnelligkeit. Durch den Hydraulikantrieb ist das System leicht und sehr robust. Versuche zur Manipulation an Hochspannungsleitungen bis 300kV wurden im Rahmen des amerikanischen Projekts TOMCAT auch im Hinblick auf die EMV-Verträglichkeit erfolgreich durchgeführt. Dieses Projekt sieht ebenfalls den Einsatz des Roboters GRIPS vor. Sein fest montierter Greifer scheint besonders geeignet, da Schließgeschwindigkeit und die Greifkraft einzeln und unmittelbar beeinflußt werden können. Außerdem existieren zwei Betriebsmodi des Greifers. Im *slaved wrist* - Mode wird der Greifer direkt durch den Master und nach den Bewegungen des Bedieners gesteuert. Im *continious wrist* - Mode dagegen kann eine geschwindigkeitsgesteuerte Drehung des Greifers bewirkt werden, die zum Schrauben oder zum Aufwickeln benutzt werden kann. Ein Masterarm, der aufgrund seiner Ergonomie und seiner *force-feedback*-Technik ein komfortables Mensch-

Maschine-Interface darstellt, bietet alle Qualitäten einer intuitiven Bedienung. Die Steuerelektronik ist modular aufgebaut, bestehend aus einer zentralen CPU- Einheit und einer hermetisch abgeschirmten *remote*-Einheit direkt am Slave. Beide Module können durch *twisted-pair*-, *coax*- oder Glasfaserkabel verbunden werden.
Weiterhin bietet die Steuerung:

- **scaled motion** Skalierung der Bewegungsverhältnisse von *master-slave* z.B. für präzise Endpositionierung
- **indexed motion** Offset des *master* relativ zum *slave*, um den Bedienerkomfort zu erhöhen und den *master* an den Bediener anzupassen
- **constrainted motion** Softwaregesteuerte Bereichs- und Grenzfestlegung für den gesamten Roboter, sowie wie für jede einzelne Achse
- **teach/playback** 'Erlernen' von Ausführungssequenzen, die dann abgespeichert und wieder abgerufen werden können. Diese *tasks* werden mit jeweils eigener Indentifikationsnummer in einer Bibliothek abgespeichert und können vorwärts sowie auch rückwärts in verschiedenen Geschwindigkeiten ausgeführt werden.
- **halt/resume** 'Einfrieren' der aktuellen Achsstellungen durch Anhalten zu jedem beliebigen Zeitpunkt
- **axis lockout** Blockieren einzelner Achsen

Kraft
TeleRobotics, Inc.



Teleroboter GRIPS mit master, slave und CPU-Einheit

4. Relative Plazierung und Ausrichtung der beiden Einzelroboter zueinander

4.1. Vorgehensweise

Prinzipiell kann man zur relativen Positionsbestimmung der Roboter drei verschiedene Wege einschlagen, wobei hier die heuristische Aufstellung nach Gefühl bewußt nicht berücksichtigt wird.

4.1.1. Maßgerechte Kopie eines existierenden Systems

Diese Möglichkeit ist weder sehr fortschrittlich, noch ist sie aufgrund der wenigen schon implementierten Systeme gleichen Aufbaus sinnvoll, da man so automatisch zahlreiche Einschränkungen übernimmt und sich neuartigen Konzepten verschließt. Außerdem erhofft man sich von einer eigenen Betrachtung weitergehende und neuartige Erkenntnisse, die auch in andere Teilbereiche des Projektes einfließen können.

4.1.2. Reale Versuche im Experimentierfeld / Simulation

Dieser Weg, der beim japanischen System eingeschlagen wurde, sieht den Aufbau eines Experimentierfeldes vor, das zum Teil auch die orginalgetreue Nachbildung einer realen Umgebung einschließt (Plattform, Leitungen, Masten, Roboter,…). Sowohl dieses Vorgehen, als auch eine Simulation erfordert ein enormes Maß an Zeit und Geld und eine sehr detaillierte Spezifikation der Manipulationen, Werkzeuge und Masten im Vorfeld. Dies führt zu einer für die Voraussetzungen gültigen Lösung, die aber wenig allgemeingültig und schlecht auf Änderungen anpaßbar ist.

4.1.3. Analytischer Ansatz

Dieses Vorgehen bietet, im Gegensatz zu den vorherigen folgende Vorteile:

- sehr kostengünstig, da auf den realen Aufbau im Vorfeld verzichtet werden kann und man nur wenig Zeit benötigt (Einsparung von Arbeitszeit).
- führt im allgemeinen auf eine exakte und eindeutige Lösung.

- durch Nichtbeschränkung auf spezielle Arbeitsgänge und Mastgeometrien erhält man eine allgemeingültigere Lösung, die die Freiheit zur Integration neuer Bearbeitungstechniken, Werkzeugen und Masttypen zuläßt.
- Denkbar ist weiterhin auch noch die Implementierung auf einem Rechner zur interaktiven Nutzung direkt vor Ort, um bei veränderbarer Aufstellung das System schnell neuen Gegebenheiten anzupassen.

Natürlich -es wäre auch zu schön gewesen- besitzt dieser Ansatz einige Nachteile, die zum großen Teil in der schwierigen mathematischen Formulierung der für die Zusammenarbeit der Roboter bestimmenden Größen liegen. Weiterhin muß man, wie man im folgenden sehen wird, nicht unerhebliche Voraussetzungen und vereinfachte Annahmen machen um auf einem erträglichen Rechenaufwand zu kommen.

4.2. Bezeichnungen

S_0 ..... Bezugskoordinatensystem
S_L ..... Koordinatensystem des linken Robotersystem (positive x-Richtung)
S_R ..... Koordinatensystem des rechten Robotersystem (positive x-Richtung)
d_R ..... Abstand zwischen linkem und rechtem Roboter ($|S_R - S_L|$)
a..... Winkel um den die Roboter nach außen gedreht werden
r_{coord} ..... maximaler Arbeitsradius für koordinierte Aktionen
d_{byp}..... maximale Greifweite im Abstand r_{coord}
F_{coord}..... xy-Projektion des gemeinsamen Arbeitsraum
F_{tot}..... xy-Projektion des gesamten Arbeitsraum
$r_{coordnorm}$ ..... genormte Größe von r_{coord}
$d_{bypnorm}$ ..... genormte Größe von d_{byp}
$F_{coordnorm}$ ..... genormte Größe von F_{coord}
$F_{totnorm}$ ..... genormte Größe von F_{tot}
d_{bypeff}..... für Optimierung modifizierte Form von $d_{bypnorm}$ ($d_{bypnorm}$ x $r_{coordnorm}$)
k_1 ..... Gewichtungsfaktor d_{bypeff}
k_2 ..... Gewichtungsfaktor $r_{coordnorm}$
k_3 ..... Gewichtungsfaktor $F_{totnorm}$
k_4 ..... Gewichtungsfaktor $F_{coordnorm}$

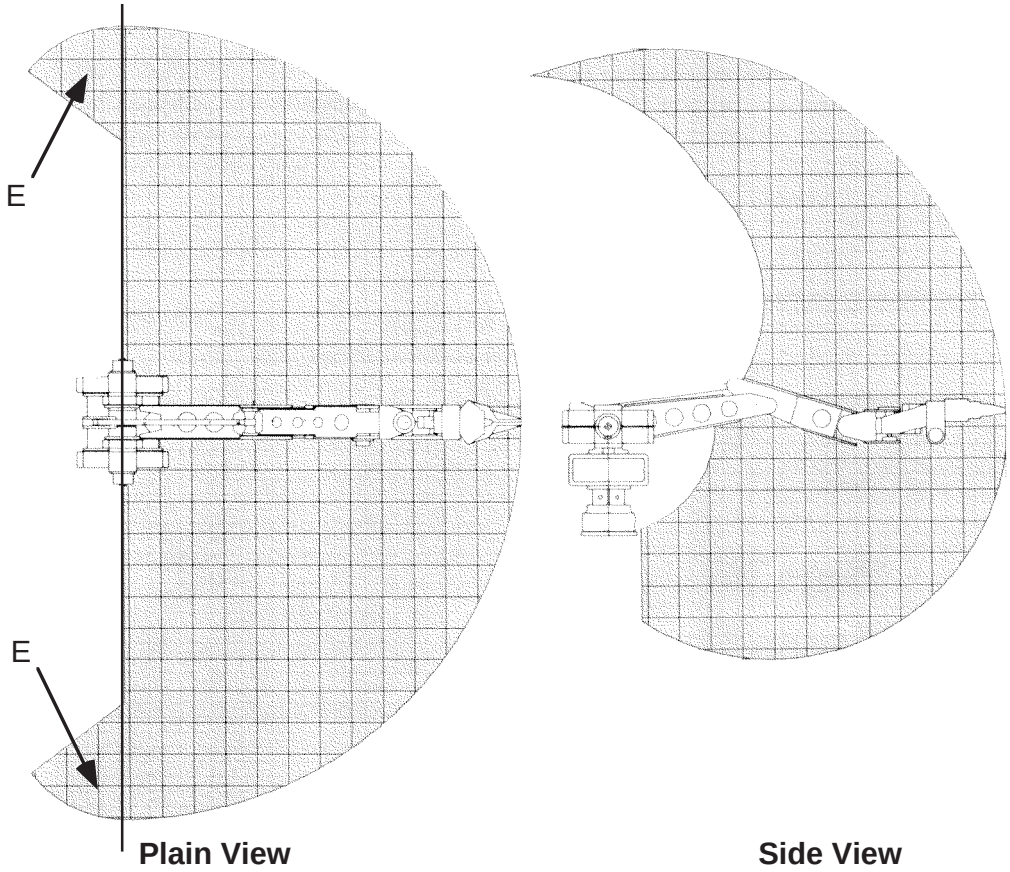
4.3. Vorausetzungen und vereinfachende Annahmen

4.3.1. Arbeitsraum

Die vielfältigen Einsatzmöglichkeiten eines Roboters und seine Mächtigkeit, in seiner Umgebung Manipulationen vornehmen zu können, hängt sehr stark von der Größe seines Arbeitsraumes ab. Man muß also erst eine Aussage über den Arbeitsraum machen, bevor man sich über Standort und Programmierung des Roboters Gedanken macht.

4.3.1.1. Gegebene Daten

In der technischen Beschreibung des einzusetzenden Roboters GRIPS findet sich ein Datenblatt mit den Senkrechtschnitten des Arbeitsraumes in zwei verschiedenen Raumebenen [Abb.].



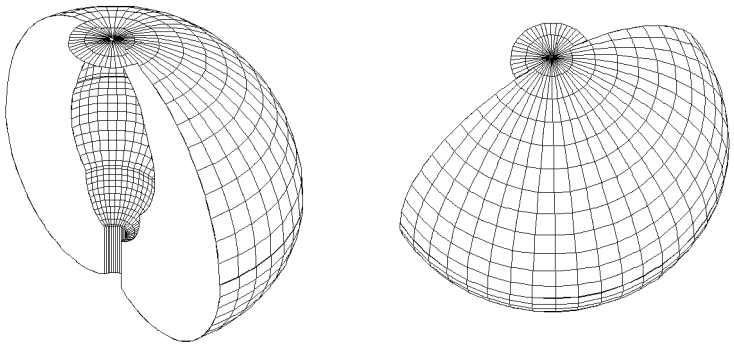
Darstellung des Arbeitsraumes in zwei Projektionen

Um daraus den erreichbaren Arbeitsraum zu bestimmen, muß man zuerst die gegebene Fläche der Seitenansicht zu einem Rotationskörper erweitern und diesen dann mit dem Zylinder, der durch Erweiterung der Draufsicht in Richtung seiner Raumachse entsteht, schneiden. Dies ist mit CAD-Programmen, die über Solidbody-Design verfügen machbar. Der Arbeitsraum kann somit graphisch dargestellt werden und ist somit auch Volumenberechnungen und Vermessungen zugänglich.

4.3.1.2. Dreidimensionale Visualisierung des Arbeitsraumes eines Roboters

Folgende Visualisierung des Arbeitsraumes erfolgte mit Hilfe des CAD-Programms AutoCad und soll eine qualitative Vorstellung des Vorgehens (Rotationskörper) und

des Resultats geben [Abb.]. Dabei wurde der Roboter in der Rotationsebene seiner ersten Achse als 'planar' angenommen und somit die beiden Ecken in der Draufsicht (E) vernachlässigt.



Dreidimensionale Visualisierung des Arbeitsraumes des Roboters GRIPS

Uns stellt sich aber das Problem der Positionsbestimmung der beiden Roboter relativ zueinander, wobei uns die graphische Darstellung des Arbeitsraumes allenfalls eine Anschauung geben wird. Vielmehr ist es hier wichtig Bewertungsgrößen in Abhängigkeit der die Positionierung charakterisierenden Maße zu suchen, die die Situation der Zusammenarbeit beider Roboter für unsere spezielle Aufgabe beschreiben.

4.3.2. Ausrichtung im Bezugskoordinatensystem

Für die nächste Betrachtung ist es notwendig die Arbeitsraumbeschreibung etwas zu detaillieren, wobei eine Einteilung nach Becquet[5] gemacht wird. Seine allgemeine allgemeinen Definition des Arbeitsraumes lautet:

- Der Arbeitsraum ist der Lösungsraum aller möglichen Positionen und Orientierungen des Tool-Center-Points (TCP), für die das inverse kinematische Problem eine Lösung hat.

Ausgehend von dieser Definition macht man folgende Unterscheidung:

4.3.2.1. Erreichbarer Volumenarbeitsraum

Hier sind nur alle möglichen Positionen des TCP von Interesse, ohne dessen Orientierung miteinzubeziehen. Das bedeutet, daß man auch Punkte in den Arbeitsraum mit aufnimmt, die nur in einer bestimmten Orientierung anzufahren sind, und somit für praktische Manipulationen nicht zur Verfügung stehen.

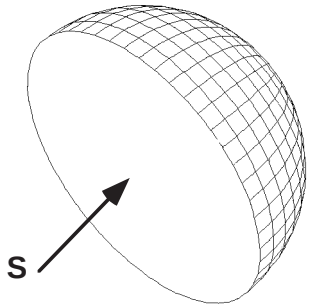
4.3.2.2. Praktischer Arbeitsraum

Hier werden nur Punkte in den Arbeitsraum mitaufgenommen, die in allen Orientierungen angesteuert werden können. Dieser Raum ist natürlich kleiner, aber für Manipulationen besonders wertvoll.

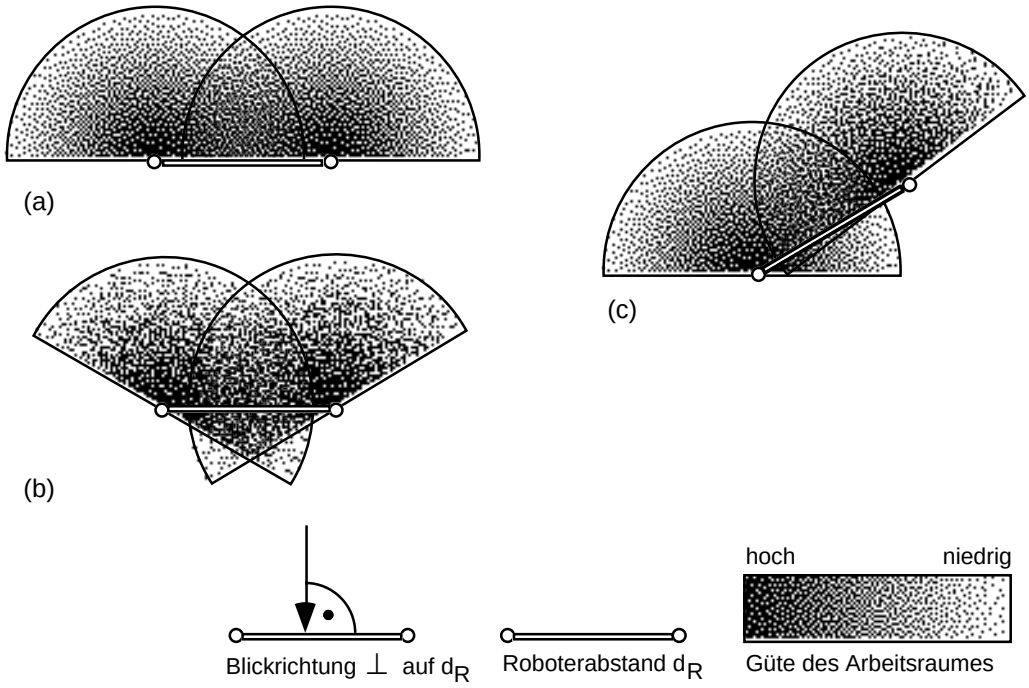
Der erreichbare Volumenarbeitsraum stellt für grobe Abschätzungen ein Maximum und der praktischen Arbeitsraum ein Minimum an Raumgröße dar. Die 'Güte' des Arbeitsraums verhält sich natürlich genau antiproportional. Sie wird sicherlich zu seinen Rändern hin abnehmen, da hier die Raumpunkte in immer weniger Orientierungen angefahren werden können.

4.3.2.3. Festlegung der die Aufstellung der Roboter bestimmenden variablen Größen

Geht man also vereinfacht davon aus, daß die Güte in einem halbkugelförmigen Arbeitsraum [Abb.] um den Roboter radial abnimmt, ist die Plazierung mit sich teilweise überdeckenden Schnittflächen S sicher insoweit optimal, als daß sich, bei gleichem absoluten Abstand d_R der Roboter, ein Maximum an gemeinsamen nutzbarem Arbeitsraum ergibt [Abb.(a)]. Der Begriff 'nutzbar' bezieht sich auf die Tatsache, daß der Raum zwischen und hinter den Robotern für unsere Anwendung nur geringen Wert hat, da die zu manipulierenden Objekte nicht von außen in diesen Raum eingeführt werden, sondern in möglichst großer Reichweite von vorne vom Roboter erreicht werden müssen. Die anderen Plazierungsmöglicheiten mit gleichgroßem oder sogar größeren gemeinsamen Arbeitsraum, die für andere Aufgabenstellungen (zwei kooperierende Roboter in einer Montagezelle) Vorteile bieten, sind hier deshalb nicht sinnvoll. So erkennt man z.B. bei Plazierung nach Abb. (b), daß ein großer Teil des Arbeitsraumes hinter der direkten Verbindung der Roboter liegt und nach Abb. (c), daß der vordere Roboter dem hinteren den freien Zugang zu den zu manipulierenden Objekten versperrt und somit ein großer Teil seines Operationsgebietes ungenutzt bleibt. Außerdem würde man mit einer unsymetrischen Aufstellung der beiden Roboter eine Seite mehr betonen, was wegen der überwiegend symetrischen Anordnung der Masten und aus Gründen der universellen Einsetzbarkeit nicht zu empfehlen ist.



Approximation des Arbeitsraumes durch Halbkugel



Draufsicht auf verschiedene Plazierungsmöglichkeiten

Aufgrund dieser Überlegungen und der Tatsache, daß die Roboter aufgrund ihrer Mechanik lotrecht (z-Achsen von S_L, S_R und S_0 sind parallel) montiert werden müssen, mache ich folgende vorläufige Festlegungen:

- Die Grundorientierung der beiden Roboter ist identisch d.h. in ihrer Homeposition sind sie armparallel zur y-Achse von S_0.
- Beide Roboter befinden sich auf der x-Achse von S_0 im Abstand d_R.

Diese Ausrichtung bietet den größtmöglichen gemeinsamen Arbeitsraum mit hoher 'Güte'. Sicherlich ist diese Positionierung der Roboter auch die am intuitivsten

bedienbare Aufstellung, die am ehesten der Anschauung des Menschen entspricht. [Abb. (a)]

Somit ist d_R die einzige variable Größe, die die relative Position der Roboter bestimmt. Wie oben schon erwähnt, kann der Zwischenraum der beiden Robotern praktisch nicht genutzt werden. Deshalb läßt man die Voraussetzung der Armparallelität fallen und führt noch einen Winkel a ein, um den beide Roboter nach außen gedreht werden. Man verliert dabei zwar gemeinsamen , gewinnt aber seitlich an gesamten Arbeitsraum, der z.B. zum Werkzeugwechsel oder zum Aufgreifen von Objekten genutzt werden könnte.[Abb. (b)]

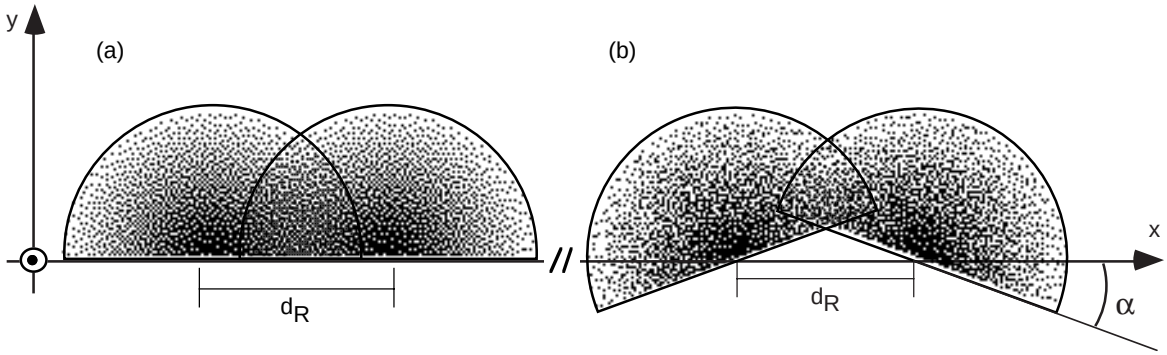
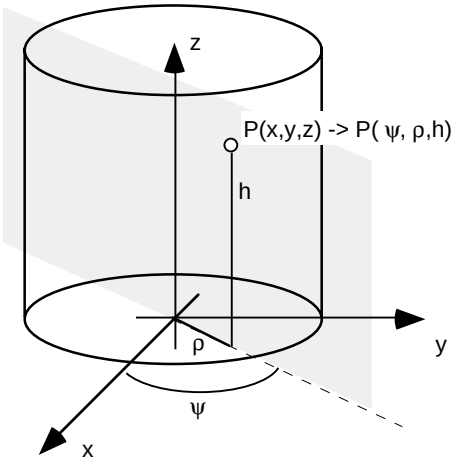


Abbildung der die Aufstellung bestimmenden Parameter d_R und a

4.3.3. Reduzierung der Komplexität der Berechnungen

Aufgrund der schwierigen mathematischen Beschreibbarkeit des dreidimensionalen Arbeitsraumes betrachtet man nur die Draufsicht auf die xy-Ebene , um das dreidimensionale Problem in ein zweidimensionales zu überführen. Das dies eine gute Näherung ist, zeigen die folgenden Überlegungen:

Zerlegt man den dreidimensionalen Arbeitsraum in die zwei Ansichten xy und yz, so muß ein beliebiger für den Roboter erreichbarer Raumpunkt ein notwendiges und ein hinreichendes Kriterium erfüllen. Der Punkt muß zwingend in der Projektion des Arbeitsraumes auf die xy-Ebene (S_L/S_R) liegen (notwendig) und danach muß mittels Transformation in Zylinderkoordinaten überprüft werden, ob der Punkt auch im gültigen Gebiet der rh-Ebene (S_L/S_R) liegt (hinreichend). Hinreichend bedeutet hier soviel, daß ein Punkt innerhalb der yz-Projektion des Arbeitsraumes in der gültigen rh-Ebene liegen kann, aber nicht liegen muß.



Zylinderkoordinatensystem

Beide Variablen d_R und a sind nur in der xy-Ebene definiert und ändern die Orientierung des Arbeitsraumes in z-Richtung nicht (Normalenvektor der Projektion bleibt parallel zur z-Achse). Es ist also eine gute Näherung, Betrachtungen bezüglich d_R und a nur in der xy-Ebene anzustellen, da die Größe des Volumen von der definierten Fläche bestimmt wird.

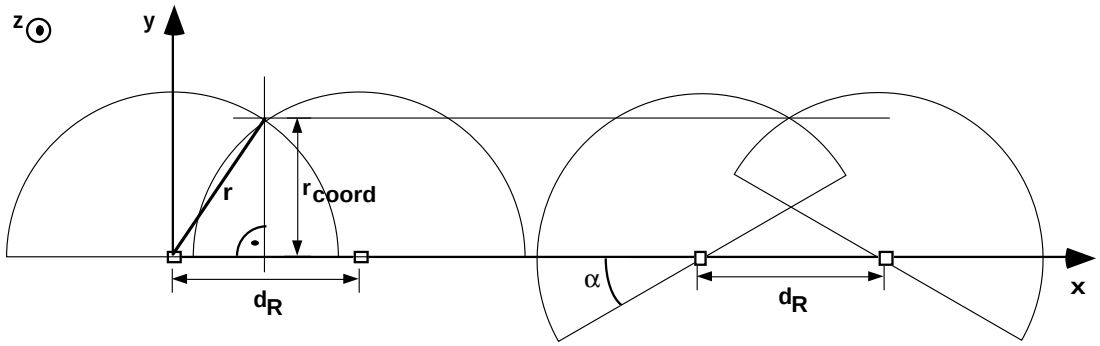
4.4. Einführung von Bewertungsgrößen

Nachdem jetzt eine Reihe von Festlegungen und Vereinfachungen gemacht wurden, muß man jetzt ein Gütemaß definieren, das vom Abstand d_R und a abhängt und nach dem dann im folgenden eine analytische Aussage über ein optimales d_R und optimales a möglich ist. Man könnte hier einzelne einfache spezifische Aktionen (verbinden, trennen, zapfen, ...) betrachten und daraus aussagekräftige Größen ableiten. Wegen der Vielzahl an Aktionen und der Tatsache, daß bei Änderung der Bearbeitungstechnik oder bei neu eingesetzten Werkzeugen die Gültigkeit der Analyse nicht mehr gewährleistet wäre, wird versucht, die Bewertungsgrößen nicht auf spezielle Aufgabenstellungen zu beziehen, um später eine möglichst allgemeingültige Aussage über die relative Position der Roboter zueinander zu bekommen. Dies gewährleistet dann einen flexiblen Einsatz des Systems.

4.4.1. Maximaler Arbeitsradius r_{coord} für koordinierte Zweiarmoperationen

Die Mächtigkeit eines Zweiarmsystems beruht zu einem großen Teil darauf, daß es in der Lage ist, mit einem Arm das Objekt zu manipulieren während der andere Arm Hilfestellung leistet d.h. das Objekt fixiert oder dreht u.s.w . Wir nehmen hier an, daß

das Objekt ungefähr dieselben Dimensionen hat wie der Greifer, und nicht etwa in spezielle Richtungen große Längen aufweist (z.B. Stangen). Da diese Manipulationen noch in möglichst großem Abstand von der Plattform (Sicherheitsabstand, ungünstige Geometrie des Mastes) durchführbar sein sollen, ist es wichtig, den Abstand zu dem Punkt, den beide Roboterarme noch gleichzeitig erreichen können, zu maximieren. Aufgrund der Darstellung des yz-Schnittes des Arbeitsraumes als Halbkreisflächen, lassen sich folgende einfache geometrische Betrachtungen anstellen [Abb.].



Maximaler Arbeitsradius r_{coord}

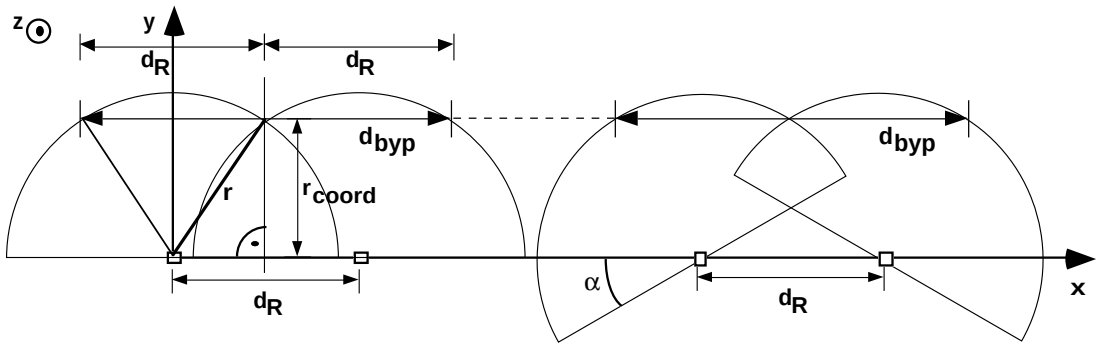
Man sieht, daß r_{coord} nicht von a abhängt. Die mathematische Abhängigkeit von d_R ist gegeben durch

$$r_{coord}(d_R) = \sqrt{r^2 - (d_R/2)^2}$$

4.4.2. Maximale Greifweite bei maximalem Arbeitsradius r_{coord}

Bei der Definition der Größe r_{coord} wurde für die zu manipulierenden Objekte die Einschränkung einer kleinen geometrischen Ausdehnung gemacht (kompakt und gleiche Größenordnung wie Greifer). Für manche Aufgaben (Bypass, Ableitungen, Isolierungen anbringen,…) muß man aber mit großen und langen Objekten arbeiten. Ausgehend davon führe ich eine Größe ein, deren Bedeutung am deutlichsten durch graphische Anschauung wird [Abb.]. Auch hier sollte man die Arbeiten an stromführenden Leitungen wegen der Konstruktion der Strommasten und wegen dem einzuhaltenden Sicherheitsabstand möglichst weit entfernt durchführen können. Der Unterschied zu r_{coord} liegt hier aber in der gleichzeitigen Forderung nach maximaler Greifweite bei dieser Entfernung. Weiterhin erhält man so einen

größtmöglichen Raum um das Objekt aufzugreifen und entsprechend zu positionieren.



Maximale Greifweite d_{byp}

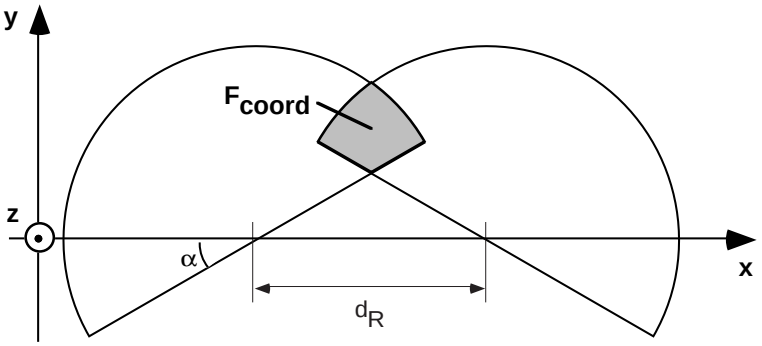
Die mathematische Abhängigkeit ergibt sich aus der geometrischen Betrachtung sehr einfach zu:

$$d_{byp}(d_R) = 2 \cdot d_R$$

Anschaulich stellt d_{byp} die Breite des maximalen Arbeitsraumes mit der Tiefe r_{coord} dar.

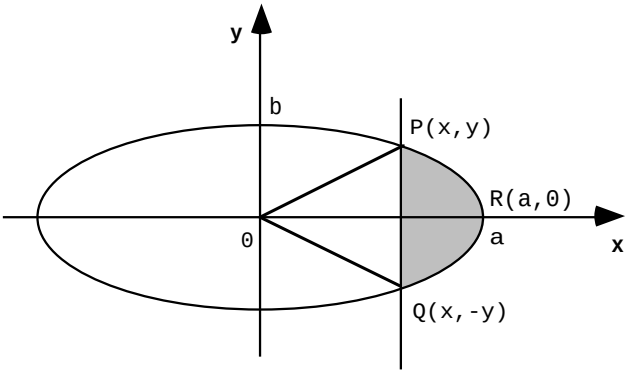
4.4.3. Maximaler gemeinsamer Arbeitsraum

Für Umgreifoperationen und Aufgabenstellungen, bei denen beide Arme am Objekt (geringe Dimension!) angreifen, ist der für beide gleichzeitig erreichbare Arbeitsraum F_{coord} eine wichtige Größe. Betrachtet man den Schnitt in der xy-Ebene, so läßt sich die Fläche als Schnittfläche der jeweiligen Einzelarbeitsflächen der beiden Roboter interpretieren [Abb.].



Fläche F_{coord} repräsentiert gemeinsamen Arbeitsraum

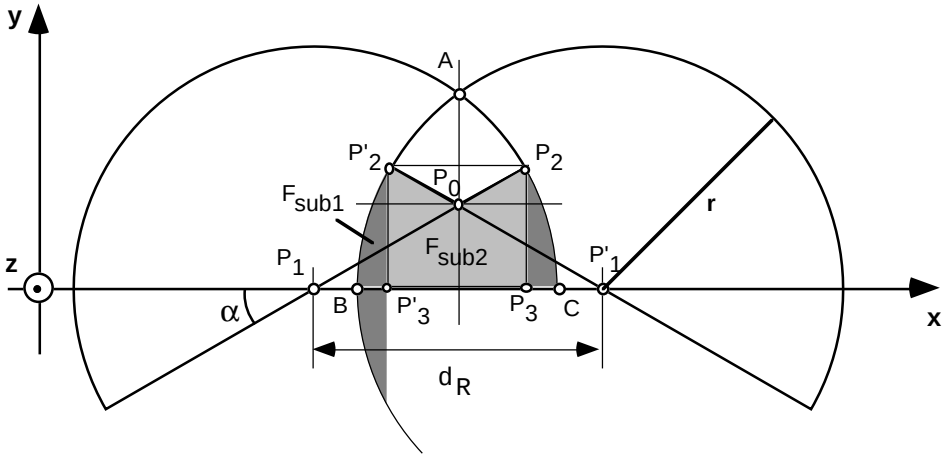
Mathematisch kann man die Schnittfläche unter Zuhilfenahme einer Formel zur Sektorberechnung bei Ellipsen errechnen [7].



Skizze zur Sektorberechnung bei Ellipsen

$$\text{Flächeninhalt} \quad S_{PQR} = ab \cdot \arccos\left(\frac{x}{a}\right) - xy$$

Da der Kreis einen Spezialfall einer Ellipse (a=b) ist, können die Eigenschaften und Formeln sinngemäß übertragen werden. Berücksichtigt man auch noch die speziellen geometrischen Verhältnisse [Abb.], so kann man für die Fläche in Abhängigkeit von d_R folgende Formel angeben.



Skizze zur Flächenberechnung von F_{coord}

$$F_{coord}(d_R,\alpha) = S_{ABC} - F_{sub1} - F_{sub2} = (r^2 \cdot \arccos\left(\frac{d_R}{2r}\right) - \frac{d_R}{2} \cdot r_{coord}) - F_{sub1} - F_{sub2}$$

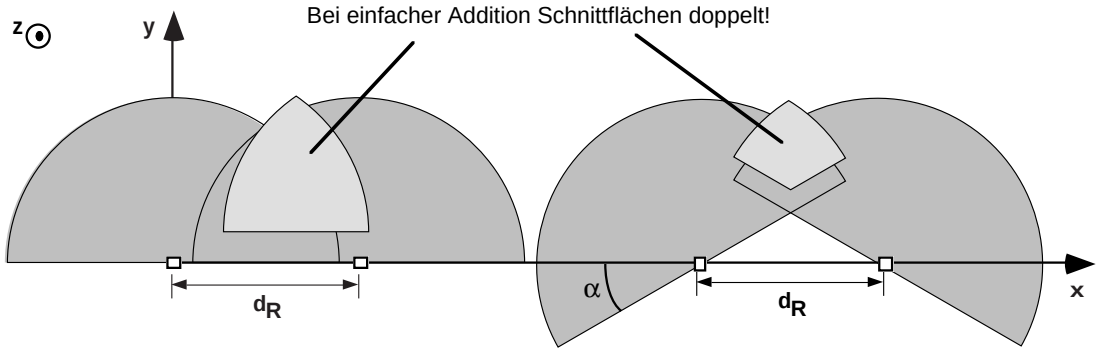
mit

$$F_{sub1}(\alpha) = 2(S_{P_1P_2C} - \Delta_{P_1P_2P_3}) = r^2 \cdot (\alpha - \sin\alpha \cdot \cos\alpha)$$

$$F_{sub2}(d_R,\alpha) = S_{P_2P'_2P_3P'_3} - \Delta_{P_2P'_2P_0} = 0,5 \cdot (2r\cos\alpha - d_R)^2 \cdot \tan\alpha$$

4.4.4. Gesamtarbeitsraum

Die wohl am einleuchtenste Größe, die es zu maximieren gilt, ist der Gesamtarbeitsraum F_{tot}. Er bestimmt wesentlich, wie flexibel, schnell und elegant die Arbeiten auszuführen sind.(z.B. ohne Änderung der Kranposition und ohne Zwischenablegen). Berechnen kann man ihn, in dem man beide Arbeitsflächen der Roboter addiert und dann die Schnittfläche subtrahiert [Abb.]. Diese Schnittfläche wurde schon als Größe F_{coord} formelmäßig angegeben und kann deshalb hier sofort verwendet werden.



Fläche F_{tot} repräsentiert gesamten Arbeitsraum

Aufgrund der geometrischen Verhältnisse (Arbeitsflächen der beiden Roboter addieren sich zur Fläche eines Vollkreises) ergibt sich dann folgende formelmäßige Abhängigkeit:

$$F_{tot}(d_R,\alpha) = F_{Vollkreis} - F_{coord}(d_R,\alpha)$$

$$\Rightarrow F_{tot}(d_R,\alpha) = \pi \cdot r^2 - F_{coord}(d_R,\alpha)$$

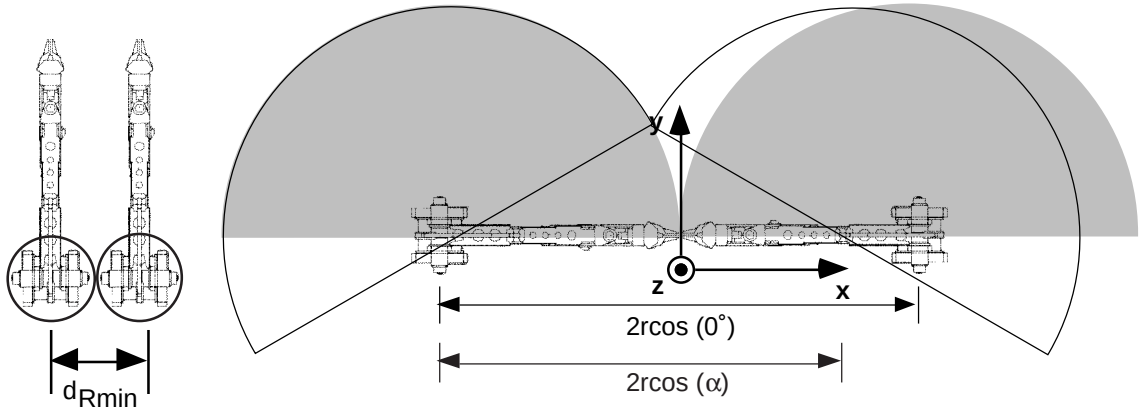
4.5. Klassifizierung der gefundenen Bewertungsgrößen

An dieser Stelle erfolgt ein kleiner Rückblick auf die gefundenen Größen und ein Versuch ihre Eigenschaften zu klassifizieren. Man hat je zwei ein- und

zweidimensionale Größen gefunden, die bei näherer Betrachtung nicht vollkommen unterschiedlich getrennte quantitative Aussagen machen. Ein großes r_{coord} impliziert eine große Fläche F_{coord} und ebenso verhält es sich mit den Größen d_{byp} und F_{tot}. Man kann also feststellen, daß es zwei Paare von jeweils ein- und zweidimensionalen Größen gibt, die d_R tendenziell in gleicher Richtung beeinflußen. r_{coord} und F_{coord} betonen in starkem Maße die Kooperation der beiden Roboter (kleines d_R und a), während d_{byp} und F_{tot} die Selbständigkeit der beiden Systeme (großes d_R und a) mehr gewichten. Die geometrischen Abmessungen der beidhändig manipulierenden Objekte unter größtmöglicher Entfernung werden von den eindimensionalen Größen r_{coord} und d_{byp} berücksichtigt, während die beiden Flächengrößen F_{tot} und F_{coord} ein Maß für den allgemeinen und flexiblen Einsatz darstellen. Man hat also eine Beschreibungsmethode gefunden, die genügend Freiheit zur Integration von verschiedensten Bearbeitungstechniken läßt, gleichzeitig aber doch eine quantitative und allgemeingültige Aussage über Eigenschaften des vorliegenden Zweiarmsystems ermöglicht.

4.6. Minimaler und maximaler Abstand der Roboter

Aufgrund der Mechanik der Roboter und der Tatsache, daß die Roboter zusammenarbeiten sollen, ergeben sich Grenzen, außerhalb derer die Betrachtung sinnlos ist und auch im Fall der Ellipsensegmentberechnung mathematisch kein korrektes Ergebnis ergeben würde. Der minimale Abstand ergibt sich aus den Hardware-Abmessungen der Roboter und wird mit d_{Rmin} bezeichnet. Bei Unterschreiten dieses Minimums stoßen die Roboter zusammen. Der maximale Abstand ist der Grenzfall, daß sich beide Roboter gerade noch berühren (d_{Rmax}=2r cos a). Das Intervall innerhalb dessen sich die Werte a bewegen wird auf $\Delta\alpha \in [0, \pi/8]$ festgelegt. Größere Werte von a sind nicht mehr sinnvoll. Wie die Anschauung zeigt, nimmt dann der Arbeitsraum in y-Richtung auf Kosten des seitlichen in x-Richtung zu stark ab.



Gültigkeitsintervall von d_R

4.7. Optimaler Abstand und Winkel unter Berücksichtigung aller Bewertungsgrößen

4.7.1. Zusammenfassen aller Bewertungsgrößen

Ziel der Betrachtung ist es, den Abstand der Roboter zu finden, so daß alle vier Bewertungsgrößen möglichst große Werte annehmen. Um trotz verschiedener Dimensionen und Funktionswertintervalle alle Größen in einem einzigen Koordinatensystem übersichtlich darstellen zu können, ist es nötig, alle auf ihr Maximum zu beziehen. Die jeweils dimensionslosen und normierten Gleichungen (Funktionswerte im Intervall 0…1) lauten:

$$d_{bypnorm}(d_R) = \frac{1}{C_{ndbyp}} \cdot 2 \cdot d_R$$

$$r_{coordnorm}(d_R) = \frac{1}{C_{nrcoord}} \sqrt{r_R{}^2 - (d_R/2)^2}$$

$$F_{coordnorm}(d_R,\alpha) = \frac{1}{C_{nFcoord}} (r^2 \cdot \arccos\left(\frac{d_R}{2r}\right) - \frac{d_R}{2} \cdot r_{coord} - r^2 \cdot (\alpha - \sin\alpha \cdot \cos\alpha) - 0{,}5 \cdot (2r\cos\alpha - d_R)^2 \cdot \tan\alpha$$

$$F_{totnorm}(d_R,\alpha) = \frac{\pi r^2 - F_{coord}}{\pi r^2} = 1 - \frac{F_{coord}}{\pi r^2}$$

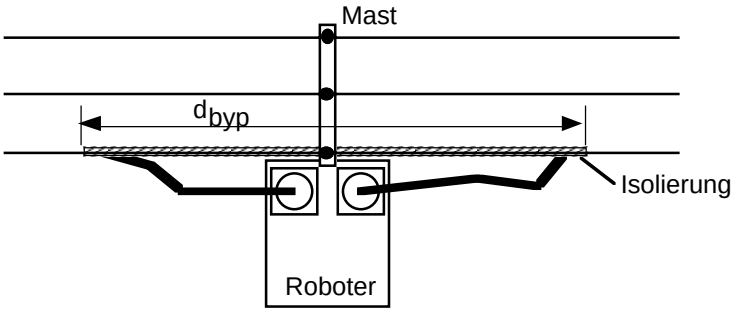
mit

$$C_{ndbyp} = d_{byp}(d_{r\,max})$$

$$C_{nrcoord} = r_{coord}(d_{r\,min})$$

$$C_{nFcoord} = F_{coord}(d_{r\,min}, \alpha_{min})$$

Die Größe $d_{bypnorm}$ besitzt eine lineare Abhängigkeit von d_R. Sie hat aber sicher nicht ihr qualitatives Optimum bei d_R=2r, da sie in Wirklichkeit stark an Aussagekraft verliert, je weiter r_{coord} gegen 0 geht. Manipulationen mit größeren Objekten sind aufgrund der mechanischen und räumlichen Gegebenheiten kaum mehr sinnvoll. Folgende Abbildung (Anbringen eines Isolationsmantels) zeigt daß bei nach links und rechts ausgestreckten Armen kaum Freiheitsgrade zur Manipulation bleiben und außerdem der Abstand des Robotersystems zu Mast und Leitungen sehr gering wird.

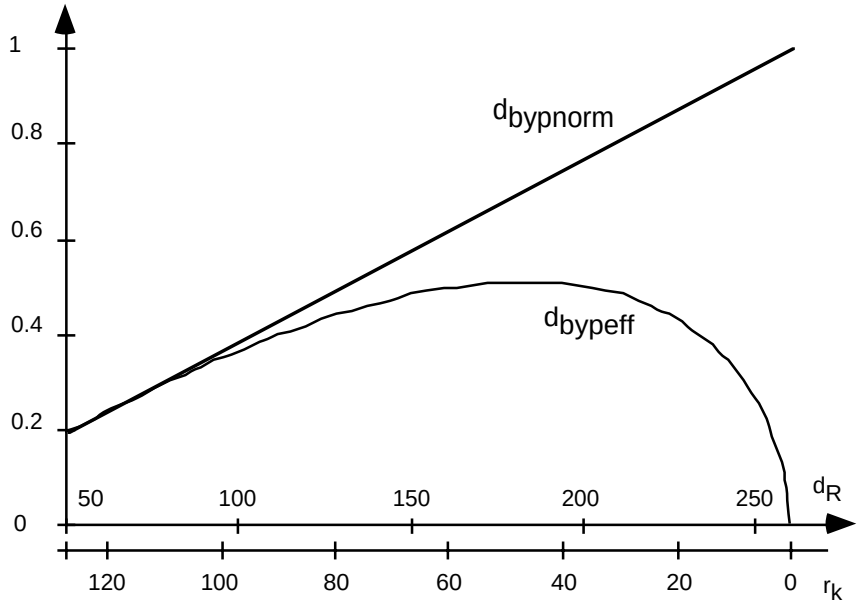


Manipulation mit d_{byp} in der Nähe seines Maximums

Um diesem gerecht zu werden, wird die Funktion mit $r_{coordnorm}$ multipliziert. Dies ist zudem sinnvoll, da d_{byp} direkt über die Größe r_{coord} definiert ist. Die formelmäßige Abhängigkeit von d_{bypeff}, d.h. die in die Optimierung miteinzubeziehende Größe von d_{byp}, läßt sich dann folgendermaßen darstellen.

$$d_{bypeff}(d_R) = \frac{1}{C_{ndbyp}} \cdot 2 \cdot d_R \cdot r_{coordnorm}$$

Folgende Abbildung zeigt den Verlaufs von d_{byp} vor und nach der Multiplikation mit r_{coord}. Die zusätzliche Einteilung der x-Achse nach r_{coord} verdeutlicht die starke Abnahme von d_{bypeff} in der Nähe von r_{coord}=0.



Graphische Darstellung von $d_{bypnorm}$ und d_{bypeff}

4.7.2. Graphische Darstellung

Die graphische Darstellung aller vier Größen in Abhängigkeit von d_R und a sowie die spätere Bestimmung von d_{Ropt} erfolgt mit Hilfe von MAPLE [Listing siehe Anhang A], einer Sprache zur rechnerunterstützten symbolischen oder numerischen Lösung von mathematischen Problemen. Alle vier Bewertungsgrößen sind im Anhang A in Form von dreidimensionalen Funktionsplots dargestellt.

4.7.3. Berechnung des optimalen Abstandes und Winkels

Es muß also nun ein Abstand d_R und ein Winkel a gefunden werden, bei denen alle zuvor gefundenen Größen möglichst hohe Werte annehmen. Dies erreicht man indem man alle Größen durch Multiplikation zu einer Funktion zusammenfaßt und von dieser dann das Maximum sucht.

$$val(d_R,\alpha) = d_{bypeff}(d_R)\cdot r_{coordnorm}(d_R)\cdot F_{totnorm}(d_R,\alpha)\cdot F_{coordnorm}(d_R,\alpha)$$

In der jetzigen Form geht jede Größe mit der selben 'Wichtigkeit' in die Berechnung des Maximums ein. Denkbar wäre jedoch, daß die verschiedenen einzelnen Tätigkeiten so ungleich verteilt sind, daß es sinnvoll wäre, die dafür spezifische Größe stärker hervorzuheben. Besteht z.B. die Aufgabe hauptsächlich darin, Verbinder anzubringen oder zu überprüfen, so ist wegen der zwingenden Zweiarmoperation, die Größe F_{coord} d.h. das Gebiet möglicher koordinierter

Aktionen (von kleinen Objekten), stärker zu gewichten. Um dies mathematisch zu formulieren, führt man die Exponenten k_1 bis k_4 ein. Durch Variation dieser Exponenten kann man dann eine Anpassung an die verschiedenartigen Aufgabenstellungen durchführen. Hierbei sei aber anzumerken, daß dies nur dann sinnvoll ist, wenn der Abstand d_R leicht und am Einsatzort des Systems variiert werden kann. Bei einer einmaligen festen Montage der Roboter mit einer nicht exakt definierbaren Verteilung der Aktionen, sind die Exponenten zu 1 zu wählen, da die Konfiguration die Lösung möglichst flexible Aufgabenstellungen gewährleisten sollte. Man erhält damit folgende Gleichung:

$$val_{\Pi}(d_R,\alpha) = d_{bypeff}{}^{\kappa_1}(d_R) \cdot r_{coordnorm}{}^{\kappa_2}(d_R) \cdot F_{totnorm}{}^{\kappa_3}(d_R,\alpha) \cdot F_{coordnorm}{}^{\kappa_4}(d_R,\alpha)$$

Eine mathematisch wesentlich einfachere Zusammenfassung der Größen wäre eine Summation mit den Gewichten als Faktoren gewesen. Die Gütefunktion hätte somit folgendes Aussehen:

$$val_{\Sigma}(d_R,\alpha) = \kappa_1 \cdot d_{bypeff}(d_R) + \kappa_2 \cdot r_{coordnorm}(d_R) + \kappa_3 \cdot F_{totnorm}(d_R,\alpha) + \kappa_4 \cdot F_{coordnorm}(d_R,\alpha)$$

Diese Funktion zeigt bei der Gewichtung ($k_1 = 2$; $k_i = 1$; $i = 2\ldots4$) ein der Multiplikation vergleichbares Maximum. Es zeigt sich aber, daß bei schon relativ geringer Variation der Gewichtungen (Dk=1) keine sinnvollen Maxima mehr auftreten, da Bewertungsgrößen mit niedrigem Gütemaß die Gesamtgüte kaum mehr beeinflußen. Man kann sie zwar durch die Gewichtungsfaktoren stärker berücksichtigen, aber diese Anpassung verliert bei Änderung der anderen Gewichte an Gültigkeit und ändert prinzipiell nichts an der Tatsache des Vernachläßigen der 'schlechten' Bewertungen. Man sieht hier, daß eine lineare Gewichtung scheitert, da niedrigere Bewertungen einer Größe entsprechend stärker zu berücksichtigen sind als hohe. Genau diese nichtlineare Anpassung gewährleistet die multiplikative Zusammenfassung. Um dies zu verdeutlichen, macht man folgende Betrachtung. Um die Übersichtlichkeit zu steigern, schreiben wir die Gütefunktion als

$$val_{\Pi}(p) = \prod_{i=1}^{4} x_i{}^{v_i}(p)$$

Weiterhin definieren wir als Gütefunktion $val^*(p)$ den Logarithmus der Gütefunktion val(p). Da der Logarithmus im Bereich von 0…1 streng monoton steigt, werden die Maxima von val(p) und $val^*(p)$ an der selben Stelle liegen. Die hier auftretende

Änderung der Funktionscharakeristik kann man außer Acht lassen, da wir nur an der Lage der Maxima interessiert sind.

$$\mathrm{val}^*(p) = \ln\left(\prod_{i=1}^{4} x_i{}^{\nu_i}(p)\right)$$

Durch Ausnutzen der Eigenschaften des Logarithmus läßt sich das Produkt in eine Summe umformen.

$$\mathrm{val}^*(p) = \sum_{i=1}^{4} \nu_i \ln(x_i(p))$$

Um das Maximum zu bestimmen, müßen wir die Funktion nach p ableiten und zu Null setzen.

$$\left(\mathrm{val}^*(p)\right)' = \sum_{i=1}^{4} \nu_i \cdot \frac{1}{x_i(p)} \cdot x_i'(p) = 0$$

Man sieht sehr schön, daß eine Bewertungsgröße bei kleinem Funktionswert $x_i(p)$ durch den Faktor $(x_i(p))^{-1}$ stärker einfließt, als bei großem. Somit hat man eine nichtlineare und vom Wert der Bewertungsgröße abhängige Gewichtung erhalten.
Nach diesen Erklärungen kehren wir nun wieder zu unserer ursprünglichen Gleichung $\mathrm{val}(d_R, \alpha)$ zurück. Das Lage des Maximums dieser Funktion kann man entweder graphisch oder rechnerisch mittels

$$\mathrm{opt}(d_R, \alpha) = \frac{\partial}{\partial(d_R, \alpha)}\left(\mathrm{val}(d_R, \alpha)\right) = 0 \qquad ;d_R = d_{R\,min} \ldots d_{R\,max}$$

$$\alpha = \alpha_{min} \ldots \alpha_{max}$$

ermitteln. Dabei ist zu beachten, daß Funktionen mit mehreren Veränderlichen, die einen eingeschränkten Definitionsbereich besitzen, zwei verschiedene Arten von Extrema geben kann:

- Extrema, die mit den Nullstellen der ersten Ableitungen berechnet werden können.
- Randwerte. Der Rand eines n-dimensionalen Definitionsbereiches ist ein (n-1)-dimensionaler Bereich, dessen Extremwerte wieder in 2 Typen relative Extrema und Randwerte) unterteilt werden können.

So kann man iterativ alle möglichen Kandidaten für die Extrema in allen Dimensionen (von n bis 0) berechnen und die Funktionswerte vergleichen.

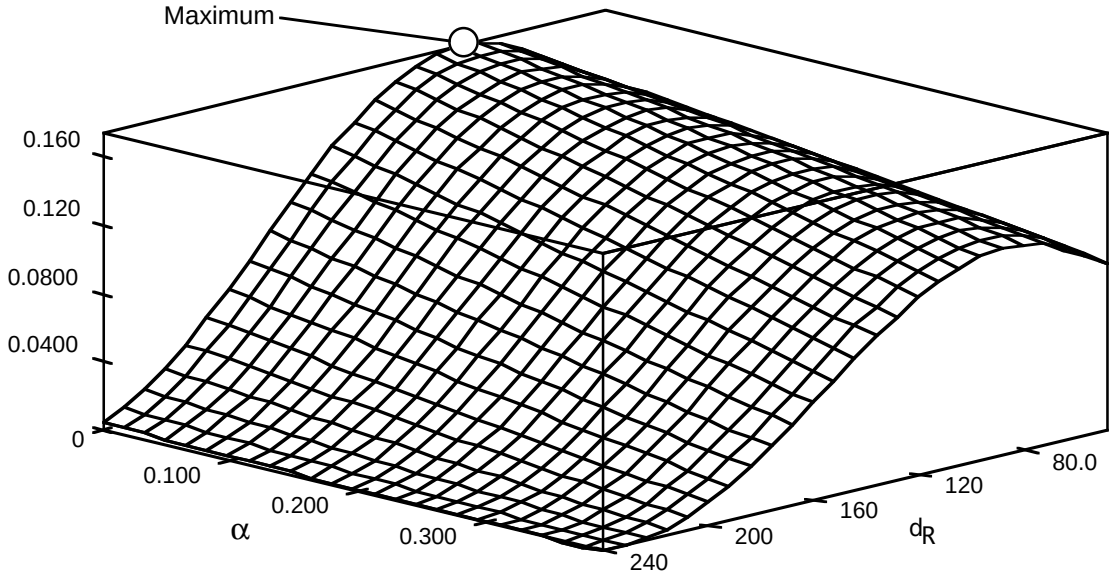
Die oben beschriebene Möglichkeit zur Bestimmung der Extrema ist im allgemeinen analytisch wegen der Komplexität der auftretenden Funktionen nicht mehr durchzuführen. Vielfach führt auch der Versuch einer numerischen Lösung zu keiner Lösung, da die verschiedenen Iterationsverfahren nicht konvergieren. Daher empfiehlt man hier einen relativ pragmatischen Ansatz indem man graphisch erst den Wert von a beim Maximum bestimmt und dann so die zweidimensionale Funktion in eine eindimensionale überführt. Die resultierende Funktion kann dann leicht abgeleitet und d_R durch die Lösung der Gleichung

$$\mathrm{opt}(d_R, \alpha) = \frac{\partial}{\partial d_R}\left(\mathrm{val}\left(d_R, \alpha = \mathrm{const.}\right)\right) = 0$$

bestimmt werden. Das Ablesen des Wertes a beim Maximum, die resultierende eindimensionale Funktion und ihre Ableitung sind im Anhang für die beiden Fälle k_3=1 und k_3=8 skizziert.
Hier kann man auch sehr schön erkennen, daß nur bei sehr starker Gewichtung des Gesamtarbeitsraumes ($k_3 \geq 7$) sich ein a ungleich 0 ergibt. Aber selbst im Fall von $a \neq 0$ ist das Maximum so wenig ausgeprägt, daß auch hier näherungsweise a=0 angenommen werden kann, ohne das Ergebnis merklich zu verfälschen.

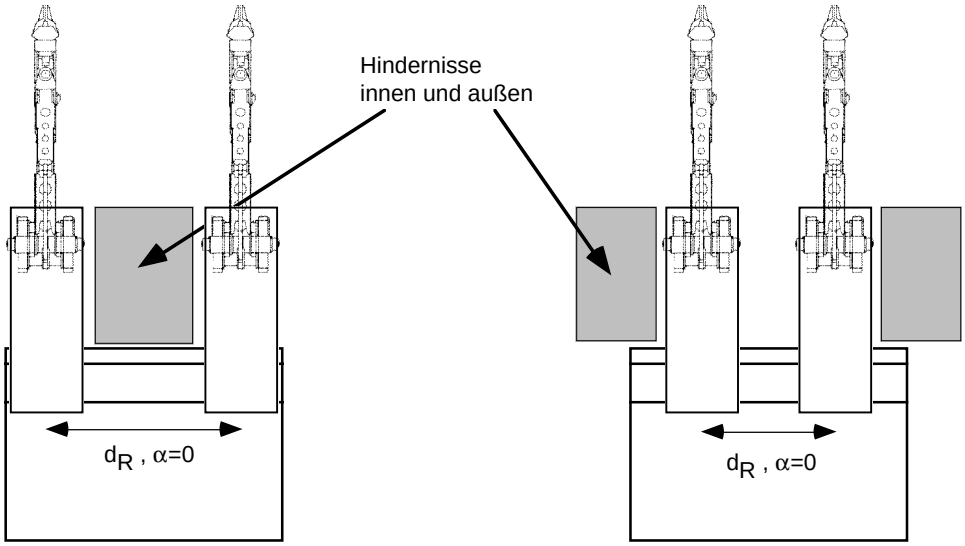
In der graphische Darstellung der Funktion val(d_R,a) für k_i = 1 ; i = 1…4 erkennt man das Maximum bei a=0 und $d_R \approx$100cm.
Der genaue Wert ergibt sich mit r=130cm zu

$$\alpha = 0 \quad ; \quad d_{Ropt} = 104{,}2\,\mathrm{cm}$$

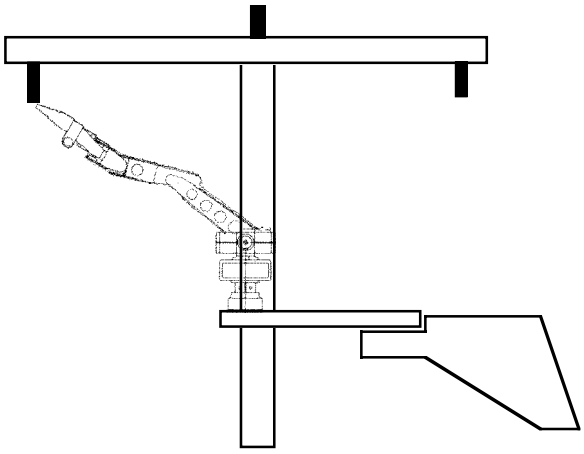


Dreidimensionale Darstellung der Qualitätsfunktion der Aufstellung

Da die Funktion in der Umgebung des Maximums nicht so stark abfällt, wäre hier noch eine Anpassung nach speziellen zusätzlichen Gesichtspunkten möglich. Dies könnte z.B. ein Aufbau mit variablem d_R sein, um in Hindernisse einzufahren oder zu umgehen [Abb.].



Ansicht des Systems in xy-Ebene



Ansicht des Systems in yz-Ebene

Folgende Tabelle soll zeigen, wie die Exponenten in die Größe d_R mit einfließen. Die einzelnen Bewertungsgrößen und d_{Ropt} sind für Gleichgewichtung (k_i=1;i=1…4) und für unterschiedliche Gewichtung angegeben. Für alle Fälle ergibt sich a=0. Die jeweils hervorzuhebende Größe ist mit k_i = 2 gewichtet. Längen sind jeweils in Zentimeter und Flächen in Quadratmeter angegeben.

Kommentar	**k_1**	**k_2**	**k_3**	**k_4**	**d_{Ropt}**	**r_{coord}**	**d_{byp}**	**F_{coord}**	**F_{tot}**
gleichmäßige Gewichtung	1	1	1	1	104,3	119,1	208,5	1,34	3,97
stärkere Gewichtung d_{byp}	2	1	1	1	128,4	113	256,8	1,06	4,25
stärkere Gewichtung r_{coord}	1	2	1	1	96,76	120,7	193,5	1,43	3,88
stärkere Gewichtung F_{tot}	1	1	2	1	117,2	116	234,5	1,18	4,13
stärkere Gewichtung F_{coord}	1	1	1	2	73,93	124,6	147,9	1,71	3,6
stärkere Gewichtung r_{coord} **und** F_{coord}	1	2	1	2	70,55	125,1	141,1	1,75	3,56

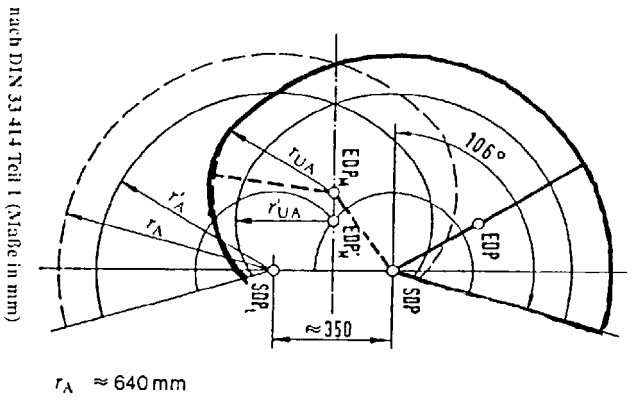
Außerdem zeigt sich deutlich, wie die Exponenten k_i die jeweils gewichtete Größe beeinflussen, und auch, daß die Größen d_{byp} und F_{tot} sowie r_{coord} und F_{coord} bei Änderung von d_R paarweise in gleicher Richtung beeinflußt werden.

4.8. Vergleich mit antropometrischen Daten des Menschen

Interessant ist an dieser Stelle noch ein Vergleich der gefundenen Größen mit Erkenntnissen aus der Anthropometrie. Dieser Vergleich scheint sinnvoll und interessant aus folgenden Gründen:

- Die Ausführung der Arbeiten erfolgt wegen der intuitiven Bedienung und der Ergonomie des Systems ähnlich einer manuellen Ausführung.
- Kinematik und Konstruktion der einzelnen Roboter ähneln nicht unerheblich dem menschlichen Arm und der menschlichen Hand.
- Die Positionierung der beiden Systeme auf der Plattform ist analog zum Schulter/Arm System.

Folgende Abbildung zeigt das Greiffeld mit einigen typische Größen für den 5.Perzentil-Mann [12].



Greiffeld des 5. Perzentil-Mannes

Das Verhältnis Schulterbreite zu Armlänge beträgt hier 0,546. Demnach müßte der Abstand der Roboter unter Beachtung der menschlichen Proportionen 70,98 cm betragen. Die Frage ist nun, welche Werte von a_i den Verhältnissen beim Menschen am nähesten kommen. Beim Menschen liegt die Häufigkeit und die Gewichtung von Tätigkeiten mit den Händen sicherlich auf der Kooperation beider Arme und Hände. Betrachtet man deshalb in vorheriger Tabelle die Funktion mit doppelt gewichtetem r_{coord} und F_{coord}, also den Größen, die von der Definition her die Kooperation repräsentieren, so findet man einen Abstand von 70,55 cm. Ein quantitativ genauer Vergleich ist auf Grund der vereinfachenden Annahmen, der auf die Aufgaben

bezogenen Bewertungsgrößen und der noch vorhandenen Unterschiede zwischen technischem System und Mensch natürlich nicht möglich. Dennoch scheinen die gefundenen Ergebnisse und Formeln durch den Vergleich Mensch/technisches System eine gewisse Bestätigung gefunden zu haben.

4.9. Ausblick und Erweiterbarkeit

Man hat hier ein Verfahren gefunden, um auf analytischem Wege die die Aufstellung bestimmenden variablen Größen eines Multirobotersytems festzulegen. Aufgrund der Komplexität wurde die eigentlich dreidimensionale auf eine zweidimensionale Betrachtungsweise zurückgeführt. Sollte diese Methode auch vor Ort eingesetzt werden, um die Aufstellungsparameter aufgabenabhängig zu verändern, ist die Schnelligkeit des Verfahrens eine wesentliche Voraussetzung. Dazu muß diese Ungenauigkeit der Betrachtung hier in Kauf nehmen.

Eine Erweiterbarkeit der vorgestellten Methode ist im wesentlichen durch eine weitere Einführung und Formulierung von Bewertungsgrößen, die die jeweiligen speziellen Aufgabebereiche charakterisieren, möglich. Im Falle einer dreidimensionale Betrachtung, wo die Angabe der Kriterien in geschlossenen Formeln nicht möglich ist, könnte die Genauigkeit der Berechnungen mittels numerischer Methoden gesteigert werden.

5. Algorithmen zur computerunterstützten Positionierung des Multirobotersystems relativ zur Hochspannungsfreileitung

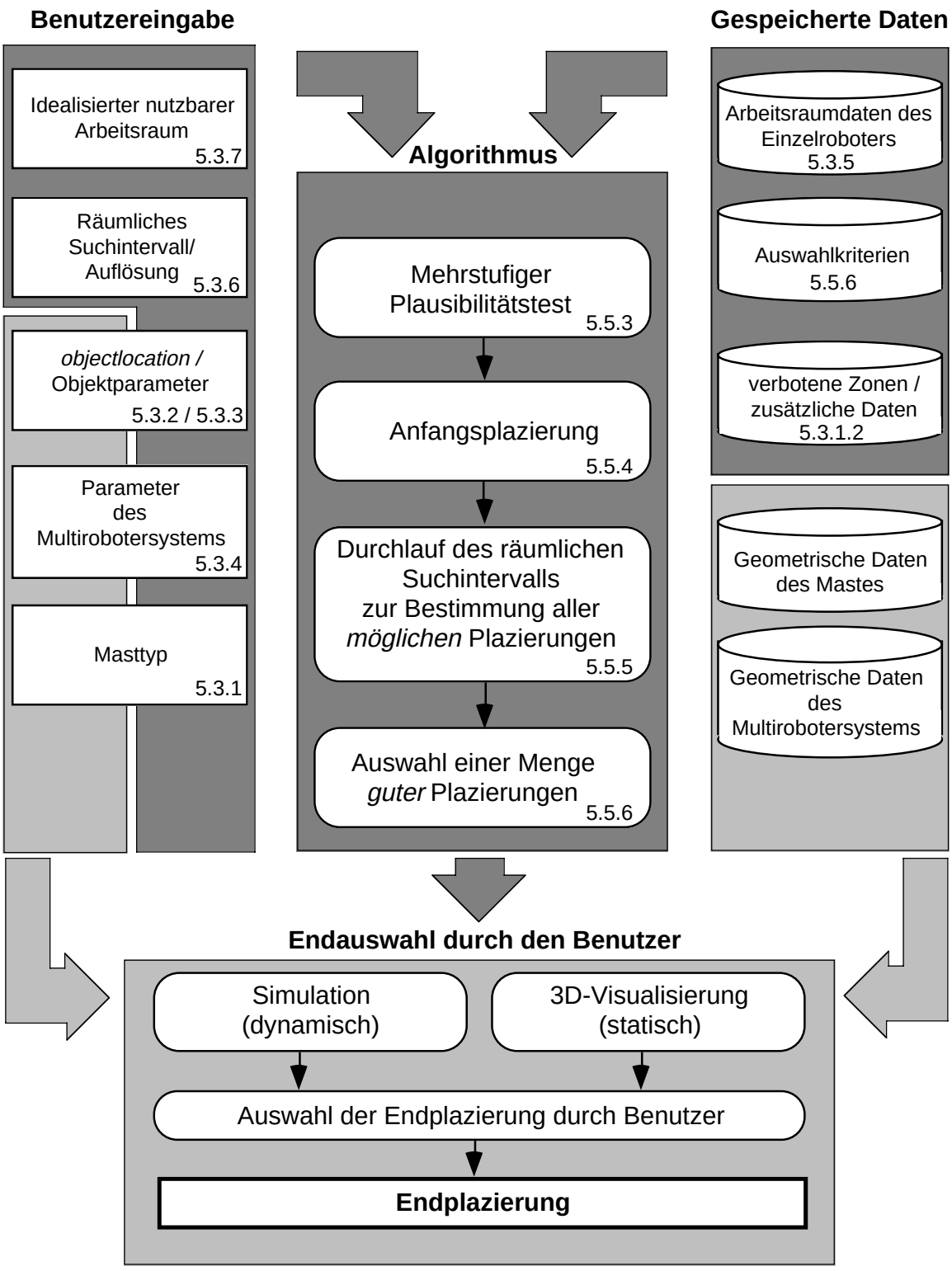
Für die Plazierung eines Robotersystem in einer festinstallierten industriellen Umgebung, wobei es hier unerheblich ist, ob es sich um ein Einzelroboter- oder Multirobotersystem handelt, hat man zusätzlich zur freien Positionierung des Roboters noch die Möglichkeit die Zuführung der zu manipulierenden Objekte zu gestalten. Zusammen mit der im allgemeinen recht beschränkte Anzahl von verschiedenartigen Manipulationen kann man oft eine sehr einfache Entscheidung über die Positionierung treffen. In unserem Falle aber ist die Umgebung fest vorgegeben und die Anpassung erfolgt allein roboterseitig. Durch die Vielzahl verschiedener Arbeitsvorgänge an vielen unterschiedlichen Typen von Strommasten, kommt man nicht umhin, zur effektiven Ausführung der Arbeiten die für die jeweiligen Aktionen nötige Positionierung des Systems im Vorfeld zu planen. Durch dieses Vorgehen erreicht man nicht nur eine schnellere Bewältigung der Aufgaben vor Ort, sondern vermeidet auch ein Herumprobieren, das leicht durch Falschpositionieren oder Schwenken des Mobilkrans zu Beschädigungen und zur Gefährdung des Bedienpersonals führen kann. Dies heißt jedoch nicht, daß bisherige Erfahrungen aus der manuellen Arbeit nicht einfließen können. Vielmehr hängt eine gute Planung entscheidend von sinnvollen Eingangsgrößen und der erfahrenen manuellen Nachbearbeitung ab. Dieses Tool soll den Planer von lästigem Probieren entledigen und ihm ein graphisches Modell seiner Arbeitsumgebung mit allen für ihn wichtigen Komponenten zur Verfügung stellen. Hierbei kann auf das bei DISAM eigens entwickeltes Robotersimulationssystem TOROS [9] zurückgegriffen werden.

Nachstehende Abbildung zeigt die grobe Struktur der Software. Auf der linken Seite sind die vom Benutzer direkt einzugebenden Größen aufgezeigt während sich auf der rechten Seite die Daten befinden, die a priori vom Benutzer als Auswahlkriterien oder Arbeitsraumbeschreibungen zur Verfügung gestellt werden. Die Arbeitsraumbeschreibung befindet sich schon im einen geeigneten Format auf einem Massenspeicher, da die Berechnung innerhalb des Plazierungsalgorithmus viel zu lange Zeit in Anspruch nehmen würde. Man löst rechenintensive

Problemstellungen zu einem früheren Zeitpunkt, speichert eine Menge von Lösungen ab und wählt sich dann nur noch die entsprechende Lösung aus.

Der eigentliche Algorithmus stellt dem Benutzer dann eine überschaubare Menge von guten Plazierungen zur Verfügung, aus denen er dann mittels echter Simulation [Anhang B.10] oder nur einer Visualisierung der vorgeschlagenen Plazierungen dann die endgültige auswählt.

Zum besseren Übersicht sind in folgender Zeichnung jeweils die Kapitelnummern in der rechten unteren Ecke angegeben. Weiterhin wurde durch helle und dunklere Unterlegungen die Zugehörigkeit der Daten zum Algorithmus und zur Visualisierung deutlich gemacht.



Struktur der Software

5.1. Bezeichnungen und Begriffe

5.1.1. Begriffe

erreichbarer Arbeitsraum..... Arbeitsraum, der alle Punkte umfaßt, die in irgendeiner beliebigen Orientierungen erreicht werden können

praktischer Arbeitsraum..... Arbeitsraum, der alle Punkte umfaßt, die in allen Orientierungen erreicht werden können

idealisierter nutzbarer Arbeitsraum *(INA)*..... derjenige Teil des Arbeitsraumes des Roboters, der für die Positionierung berücksichtigt wird. Die Festlegung erfolgt durch einen quaderförmigen Raum der Maße h_{IAR}, b_{IAR} und t_{IAR} relativ zum Punkt P_{IAR}.

nutzbarer Arbeitsraum*(NA)* ..... Schnittmenge des idealisierten nutzbaren Arbeitsraumes mit echtem Arbeitsraum

frame ..... siehe Endeffektorframe

Endeffektorframe ..... 4x4 Matrix, die Position, Orientierung und Skalierungs- und Perspektivinformation enthält

objectlocation..... Position und Orientierung eines punktförmigen Objektes

EAO ..... Einarmobjekt - Objekt, das nur von einem Roboter manipuliert wird

ZAO ..... Zweiarmobjekt - Objekt wird koordiniert durch beide Roboter manipuliert

COI ..... Kodierung des Orientierungsintervalls

COT ..... Kodierung des Objekttyps (*EAO*, *ZAO*)

5.1.2. Bezeichnungen

d	diskrete Auflösung des vom Benutzter zu definierenden Suchintervalls
$(\vec{n},\vec{o},\vec{a})$	Koordinatensystem des Greifers mit den Vektoren $\vec{n}$ *(normal)* $\vec{o}$ *(orientation)* $\vec{a}$ *(approach)*
n_{INT}	Anzahl an Orientierungsintervallen
n_S	Anzahl der diskreten Positionen im benutzerdefinierten räumlichen Suchintervall
n_{aut}	Anzahl der diskreten Positionen im optimalen räumlichen Suchintervall (berechnet durch Programm)
n_V	Anzahl der Volumenelemente, in die der Arbeitsraum des Roboters zerlegt wird.
res	diskrete Auflösung der abgespeicherten Arbeitsräume
P_{IAR}	Referenzpunkt zur Definition des idealisierten nutzbaren Arbeitsraumes
b_{IAR}, h_{IAR}, t_{IAR}	Breite, Höhe und Tiefe des idealisierten nutzbaren Arbeitsraumes
v_x, v_y, v_z	Dimensionen in x-, y- und z-Richtung des Suchintervalls in der Einheit d
Db, Dh, Dt	Dimensionen des alle Objekte *(EAOs* und *ZAOs)*umschreibenden quaderförmigen Volumenkörpers
Db_{coord}, Dh_{coord}, Dt_{coord}	Dimensionen des alle koordiniert zu manipulierenden Objekte *(ZAOs)*umschreibenden quaderförmigen Volumenkörpers
S_O, S_{AR}	Schwerpunkte der Objekte und des Arbeitsraumes
S_L, S_R	Koordinatensysteme des linken und rechten Einzelroboters
S_S	Koordinatensystem des Multirobotersystems
S_K	Koordinatensystem des Mobilkrans
R_L, R_R	Kennzeichnung für linken und rechten Einzelroboter

t_x, t_y, t_z	Vektor zur Translation eines Objektes (zusätzlicher Objektparameter)
r_j, r_J	Winkel zur Angabe der Rotation eines Objektes (zusätzlicher Objektparameter)
q_1, q_2, q_3	Parameter der Kriterien zur Bewertung einer möglichen Plazierung
$G_n(q_1, q_2, q_3)$	Gütefunktion der n-ten Plazierung - abhängig von den Parametern q_i

5.2. Voraussetzungen und vereinfachende Annahmen

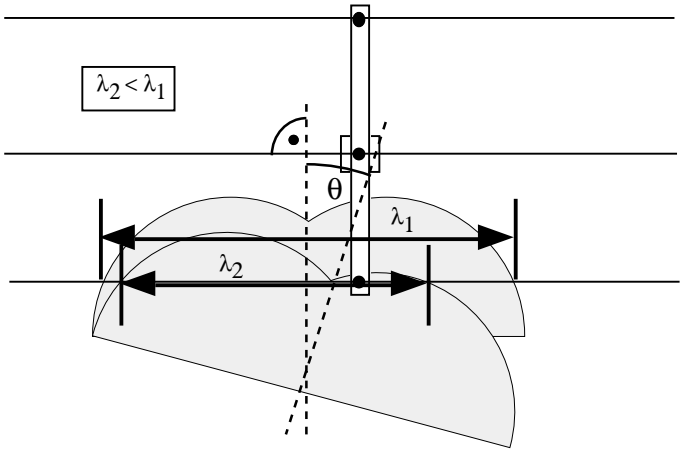
5.2.1. Ausrichtung des Mastes

Um eine spätere Systematisierung zur Entwicklung von Positionieralgorithmen zu vereinfachen, legt man die Ausrichtung des Masten bezüglich des Weltkoordinatensystems so fest, daß die Leitungen parallel zur x-Achse verlaufen.

5.2.2. Ausrichtung der Roboter bezüglich des Mastes

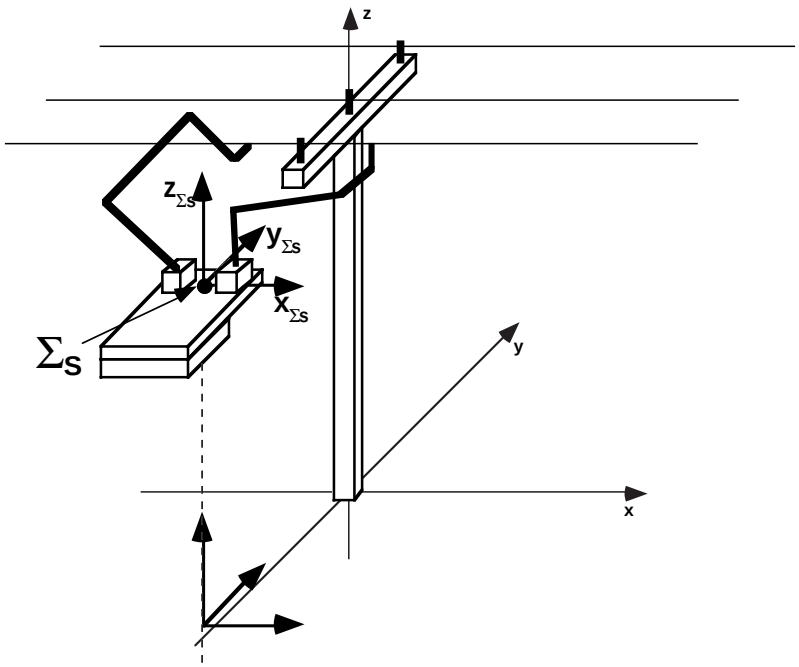
Es erscheint sinnvoll, die Fülle von denkbaren Plazierungen des Systems bezüglich des Mastes ebenfalls einzuschränken um eine Verringerung der Komplexität zu erreichen. Man macht dazu folgende Überlegungen:

- Ein Plazierung des Systems so, daß die Armausrichtung der Roboter mit der Laufrichtung der Leitungen übereinstimmt, macht wenig Sinn, da die Leitungen dann nur eine Positionierung weit unterhalb von ihnen zulassen würden. Diese würde den Arbeitsraum bis auf ein Minimum reduzieren und wäre zudem auch für den Mobilkran nur schwer durchzuführen. Als durchführbare Möglichkeit bliebe deshalb nur die Möglichkeit einer Positionierung links oder rechts der Leitungen, was ebenfalls wenig sinnvoll wäre.
- Weiterhin bietet nur die Plazierung senkrecht zu den Leitungen einen möglichst flexiblen Zugriff auf die zu manipulierenden Objekte. Jedes Wegdrehen aus der Senkrechten verkleinert die Reichweite für Manipulationen[Abb].



Senkrechte Positionierung des Systems bezüglich der Leitungen

Aus diesen Überlegungen resultiert die folgende in der Zeichnung dargestellte Plazierung, die im folgenden immer vorausgesetzt wird. Das hier eingezeichnete Bezugskoordinatensystem des System S_S wird genau in die Mitte der direkten Verbindung der beiden Roboterkoordinatensystem (S_L und S_R) gelegt.



Orientierung des Systems bezüglich des Mastes

5.2.3. Keine Positionsveränderung während der Ausführung

Weiterhin wird festgelegt, daß nur Aktionen geplant werden können, die keine Veränderung der Plazierung während der Ausführung verlangen. Komplexere und

großräumigere Vorgänge sollen in kleinere zerlegt werden, wobei das Abarbeiten der einzelnen Teile sequentiell mit entsprechender Umplazierung erfolgt. Diese einzelnen Subvorgänge werden so geplant, als seien sie vollständig abgeschlossene Aktionen.

5.3. Benötigte Daten

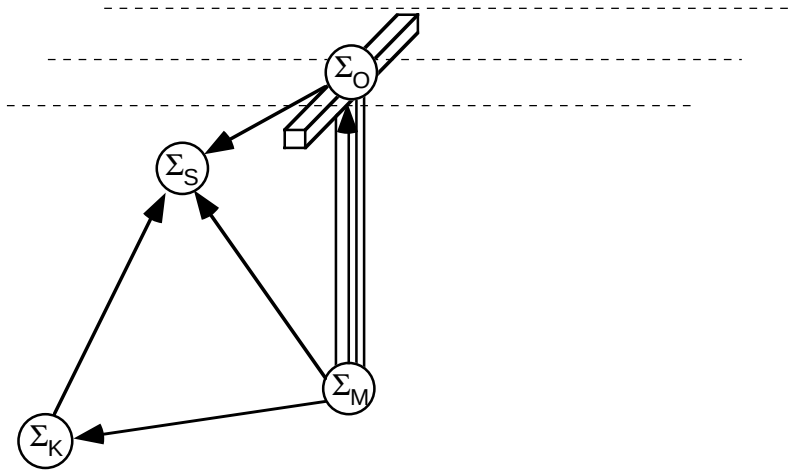
Im folgenden werden die für den Algorithmus wichtigen Größen erklärt. Einige dieser Daten liegen schon in geeigneter Form auf einem Massenspeicher bereit, während andere jeweils vom Bediener für die jeweilige Planung eingegeben werden müssen.

5.3.1. Masttyp

Es ist offensichtlich, daß für die Positionierung des Systems der Mast mit dessen Mastgeometrie eine wichtige Rolle spielt. Die zur Bestimmung einer geeigneten Plazierung und zur späteren Simulation benötigten Daten lassen sich in drei Arten unterteilen. Zur Speicherung bietet sich eine Datenbank an, die einen leichten Zugriff auf die unterschiedlichen Mastdaten ermöglicht, und die sehr leicht um neue Mastgeometrien erweitert werden kann.

5.3.1.1. Bezugskoordinatensysteme

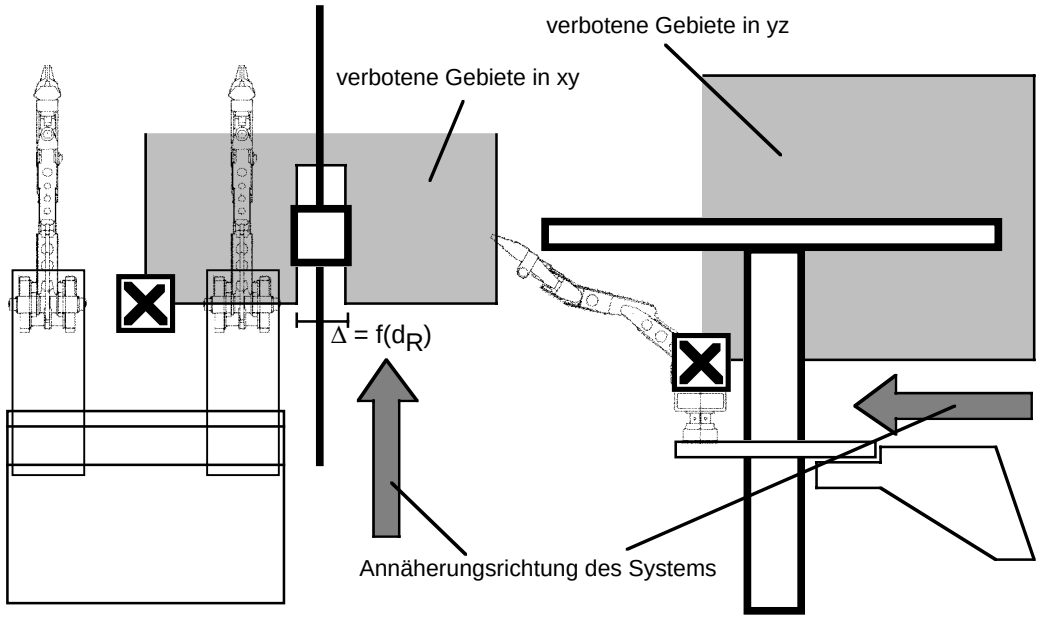
Der Mast bietet das feste Bezugskoordinatensystem in der Umgebung, bezüglich dessen alle zu manipulierenden Objekte und auch das System definiert werden. Im allgemeinen legt man die Basis in den Fußpunkt des Mastes mit der schon festgelegten Orientierung. Zusätzlich kann man aber zur Vereinfachung der Objektdefinition ein zweites Basiskoordinatensystem S_O definieren, daß im allgemeinen durch eine Translation von S_M entsteht und die Geometrien der einzelnen Masten berücksichtigt. Berücksichtigt man, daß die Koordinatensysteme S_M, S_O und S_S aufgrund der Voraussetzungen die gleiche Orientierung haben, so ergeben sich mit dem Koordinatensystem des Mobilkrans S_K folgende vektorielle Beziehungen.



Lage der verschiedenen Bezugskoordinatensysteme

5.3.1.2. Für Plazierung gesperrte Raumbereiche

Durch den teilweise doch recht komplexen Aufbau der Masten und der Größe der Plattform, ergeben sich an der Mastgeometrie orientierten Raumbereiche, in der eine Positionierung des Systems zu gefährlich (Sicherheitsabstand) oder einfach aufgrund von Kollisionen nicht möglich ist. Diese Bereiche werden in geeigneter Form relativ zu S_M beschrieben und stehen dann für die Bewertung der gefundenen Lösungsmenge der Plazierungen zur Verfügung. Diese Raumbereiche hängen aber auch vom physikalischen Aufbau der Plattform ab. Bei sich verändernden Plattformgeometrien muß man deshalb die Definition dieser verbotenen Zonen nicht absolut, sondern von den sich ändernden Plattformparametern angeben (z.B. d_R). Die nachfolgende Skizze zeigt ein Beispiel zur Festlegung der gesperrten Zonen in Form von zwei Ansichten.



Qualitative Skizze zur Definition der für die Plazierung verbotenen Gebiete

Auf die Frage, ob diese Beschreibung sinnvoll ist oder ob man hier besser eine volumenorientierte Definition bevorzugt soll hier nicht mehr eingegangen werden, da dies für das Verständnis des Plazierungsalgorithmus nicht mehr von Interesse ist.

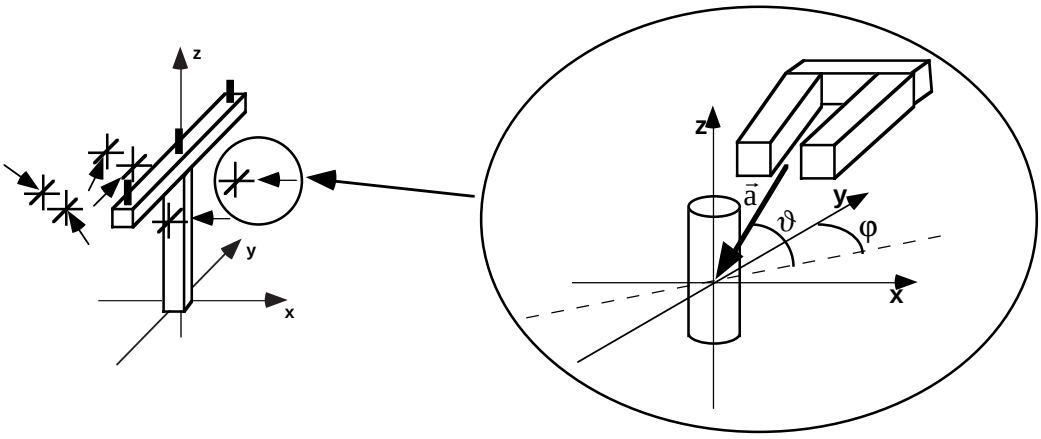
5.3.1.3. Geometrische Daten des Mastes

Für die spätere Simulation in TOROS oder die 3D-Visualisierung sind die geometrische Daten des Mastes im entsprechenden Eingabeformat erforderlich. Diese Daten werden aber für den eigentlichen Algorithmus nicht benötigt, da die für die Plazierung wichtige Information schon in der Form der Beschreibung der verbotenen Gebiete und der Definition der Koordinatensysteme enthalten ist.

5.3.2. Objektposition und -orientierung

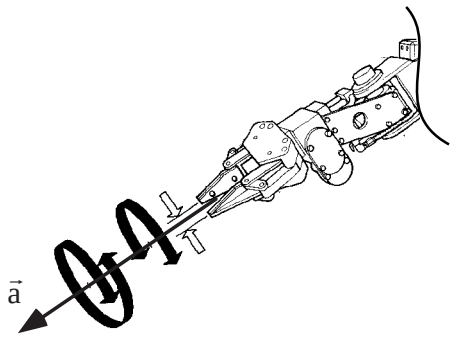
Man nimmt hier erst einmal an, daß es für die Plazierung der beiden Roboter nicht von Bedeutung ist, welche Manipulationen an den Objekten durchgeführt werden, sondern nur, wo sie sich befinden und aus welcher Raumrichtung das Objekt gegriffen wird. Die Raumrichtung wird repräsentiert durch die Orientierung des Objektkoordinatensystems, da dieses zum Greifen mit dem Koordinatensystem des Greifers zur Deckung gebracht werden muß. Es wird also für jedes Objekt ein sogenanntes *objectlocation* definiert, daß aus der Position und der räumlichen Ausrichtung des Objektkoordinatensystems besteht. Die räumliche Ausdehnung der

Objekte wird nicht betrachtet, so daß das Objekt durch die Angabe eines Punktes und dessen Ausrichtung im Raum vollständig beschrieben ist [Abb.].



Beschreibung der Objekte durch 'objectlocation'

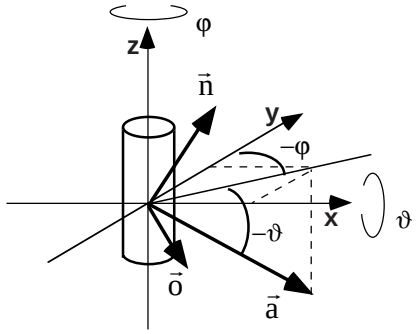
Zur Angabe der Greiforientierung ist es aufgrund der speziellen Greifermechanik des Roboters GRIPS völlig ausreichend, den Vektor $\vec{a}$ (Vektor in Richtung der Annäherung, *approach*) zu spezifizieren. Der Roboter GRIPS ist nämlich in der Lage, seinen Greifer ohne Winkelbegrenzung um den Vektor $\vec{a}$ zu drehen (*continous mode*). Man wird aber auch später anhand der inversen kinematischen Gleichungen sehen, daß man nur die Orientierungsparameter a_x, a_y, a_z und die Positionsparameter p_x, p_y, p_z zur Berechnung der Arbeitsräume abhängig von der Greiforientierung braucht.



' continous - mode' des Roboters GRIPS

Da das greifereigene Koordinatensystem $(\vec{n},\vec{o},\vec{a})$ den selben Ursprung wie das Objektkoordinatensystem besitzt kann es für unsere Zwecke durch zwei Rotationen

mit den Winkeln J (um x-Achse) und j (um z-Achse) ohne Translation überführt werden.



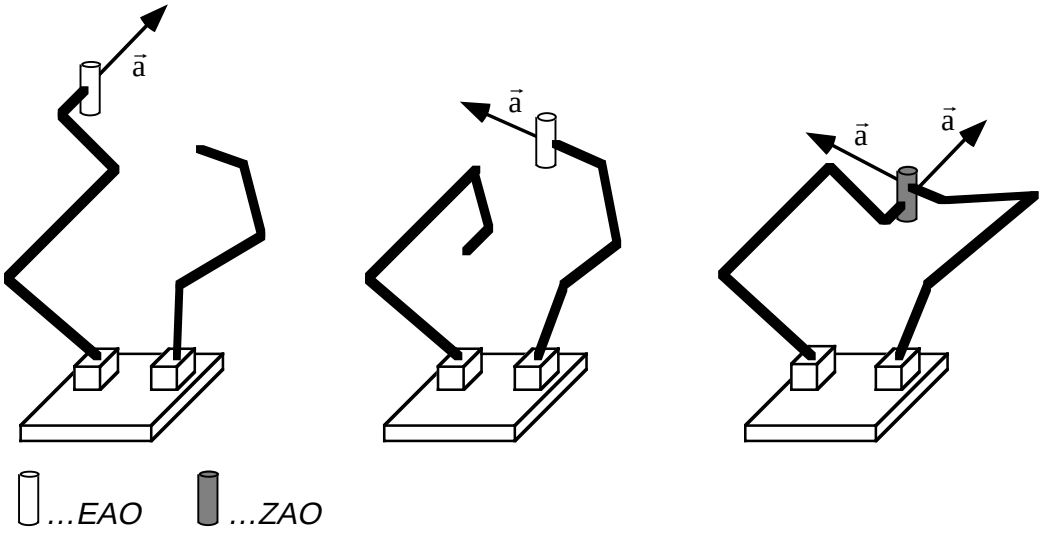
Rotation des Greiferkoordinatensystems um x- und z-Achse

Es wird hier darauf hingewiesen, daß durch die Rotationen um die x- und die z-Achse nur der Vektor $\vec{a}$ eindeutig definiert ist. Für eine eindeutige Beschreibung der Orientierung aller drei Vektoren $(\vec{n},\vec{o},\vec{a})$ bedarf es drei Winkel .

Wird in folgenden von Orientierung im Raum gesprochen, kann man davon ausgehen, daß diese in unserem speziellen Fall die Greiforientierung in Richtung des Vektors $\vec{a}$ ist. Diese Orientierung von nur durch die zwei Winkeln J und j bestimmt, obwohl für den allgemeinen Fall natürlich drei Raumwinkel notwendig sind.

5.3.3. Zusätzliche Objektparameter

Da wir hier ein System zweier Roboter einsetzen, ist es für die Erreichbarkeit eines Objektes von entscheidender Bedeutung, von welchem Roboter es manipuliert werden soll. Man unterscheidet zwischen Objekten, die nur von jeweils einem der beiden Roboter gegriffen werden *(EAO)* und solchen, die beidhändig manipuliert werden *(ZAO)*. Bei der Definition eines Objektes für die Zweiarmmanipulation wird für jeden der beiden Roboter die Greifrichtung getrennt angegeben. Die Information ob es sich um ein Einarm- oder Zweiarmobjekt handelt ist für nachfolgende Betrachtung der Anfangsplazierung von entscheidender Bedeutung, da der gemeinsame Arbeitsraum durch seine Form und seine Größe die Anzahl der Plazierungsmöglichkeiten erheblich einschränkt. Die Codierung des Objekttyps (*EAO* oder *ZAO*) sowie die Zuordnung der Objekte zu den einzelnen Robotern, werden als Zusatzinformation dem *objectlocation* angehängt.



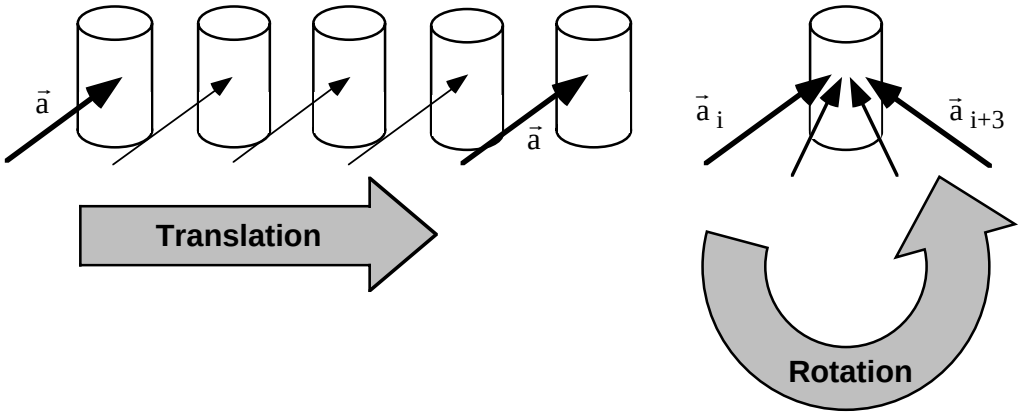
Einteilung der Objekte in EAOs und ZAOs

Alle bisherigen relevanten Eingabegrößen werden in nachstehender Tabelle noch einmal zusammengefaßt.

Objekttyp	Position			Orient.		Roboter	
COT	x	y	z	j	J	R_L	R_R
EAO beliebig	-	-	-	-	-	*1*	*1*
EAO	-	-	-	-	-	*0*	*1*
EAO	-	-	-	-	-	*1*	*0*
ZAO	-	-	-	-	-	*0*	*1*
				-	-	*1*	*0*

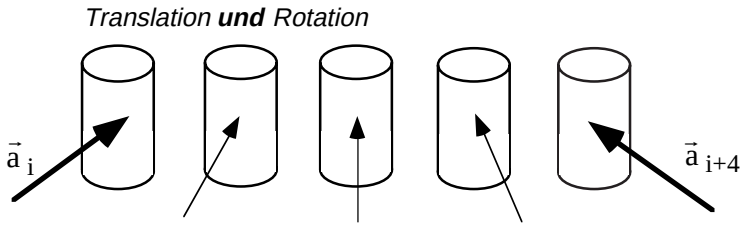
Die Manipulation eines Objektes ist durch dessen Erreichbarkeit in einer einzigen Greiforientierung natürlich noch keineswegs definiert. Vielmehr ist es erforderlich, dynamische Manipulationen wie z.B. schrauben, zapfen oder schieben eines Objektes in die Positionierungsbetrachtung mitaufzunehmen. Alle diese komplexen Operationen lassen sich jedoch in die beiden Grundbewegungsarten Rotation und Translation zerlegen.

Die Idee beruht nun für diese Angaben auf der Annahme, daß man für eine Drehung endlich viele Objekte an der selben Position, aber mit Orientierungen zwischen Anfangs- und Endorientierung definiert. Ebenso verfährt man bei einer translatorischen Verschiebung des Objekts.



Zerlegung einer Translation oder Rotation in Einzelobjekte

Ebenso kann man sich auch eine Kombination beider Grundbewegungsarten vorstellen, die man dann z.B. entsprechend Abb. zerlegen könnte.



Diese Zerlegung einer kontinuierlichen dynamischen Bewegung in statische Objekte diskreter Orientierung ist aber natürlich im allgemeinen nicht gültig, da zwischen zwei Objekten aufgrund der Achsenbeschränkungen ein Konfigurationswechsel des Roboters (lefty, righty, above, below,...) stattfinden kann. Da wir aber bei der Formulierung des Arbeitsraumes nur eine Konfiguration betrachten, ist dieser pragmatische Ansatz hier sinnvoll. Man muß also noch zusätzlich den Translationsvektor (t_X, t_Y, t_Z) und die Rotationswinkel (rj, r_J) in der Objekttabelle angeben.

Wird ein Objekt sehr intensiv bearbeitet, ist es wichtig, daß es bei der Plazierung entsprechend berücksichtigt wird. Um dies zu gewährleisten, führt man noch eine Gewicht g_i ein, daß die Wichtigkeit des Objektes i betont. Man erhält somit als Eingabegrößen:

COT	x	y	z	t_X	t_Y	t_Z	j	J	rj	r_J	g	R_L	R_R
EAO beliebig	-	-	-	-	-	-	-	-	-	-	-	*1*	*1*
EAO	-	-	-	-	-	-	-	-	-	-	-	*0*	*1*

EAO	-	-	-	-	-	-	-	-	-	-	-	*1*	*0*
ZAO	-	-	-	-	-	-	-	-	-	-		*0*	*1*
				-	-	-	-	-	-	-		*1*	*0*

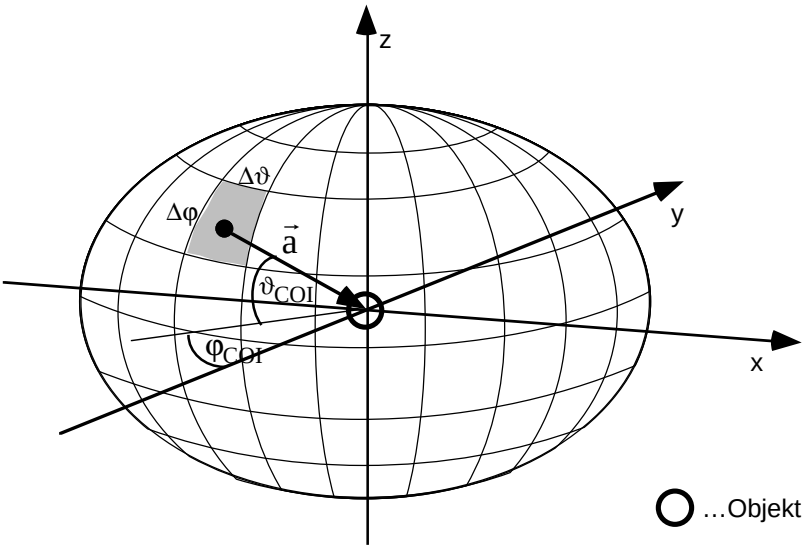
5.3.4. Geometrische Daten des Multirobotersystems

Da das System aus Plattform und zwei variabel aufstellbaren Robotern besteht, müssen hier die die Aufstellung bestimmenden Parameter eingegeben werden. Dies ist in unserem Falle nur der Roboterabstand d_R. Alle anderen für die Aufstellung wichtigen geometrischen Daten fließen schon in andere Bereiche automatisch mit ein (vgl. verbotene Zonen / Masttyp).

5.3.5. Arbeitsraumdaten

Die Beschreibung des Arbeitsraumes, die vom Hersteller zur Verfügung gestellt wird, berücksichtigt alle Punkte im Raum, die in irgendeiner beliebigen Orientierung angefahren werden können. Will man aber komplexere Manipulationen mit Hilfe des Roboters durchführen, ist es wenig hilfreich, das Objekt nur in einer beliebigen Orientierung zu erreichen. Wie man schon an der Definition der Objekte durch Position und Greifrichtung sehen konnte, muß man das Objekt in den Teil des Arbeitsraumes legen, der vom Roboter in der definierten Orientierung erreicht werden kann. Becquet [5] unterschied nur zwischen praktischen und erreichbaren Arbeitsraum . Dies stellt die beiden Extrema der Erreichbarkeit in irgendeiner oder in allen möglichen Orientierungen dar. Der praktische Arbeitsraum existiert aber bei den meisten Robotern gar nicht (oder ist verschwindend klein), da die Kinematik der Roboter durch den mechanischen Aufbau sehr stark beschränkt ist. Man sieht also, daß diese Unterscheidung für unser Positionierungsproblem nicht geeignet ist.
Nötig wäre also die Formulierung des Arbeitsraumes für jede beliebige Raumorientierung. Eine analytische Beschreibung in Form von vertretbar komplexen Lösungen ist hier aber nicht möglich, da alle bisher hierzu bekannten Verfahren erhebliche Einschränkungen bezüglich der Orientierung machen.
Es bleibt deshalb nur die Möglichkeit einer volumenorientierten diskreten Zerlegung des Arbeitsraumes und dessen Berechnung für verschiedene Orientierungen mit anschließender Abspeicherung in einer geeigneten Form. Zur Realisation verwendet man einen Scanning-Algorithmus, der den gesamten Raum entlang fest definierten

Kurven durchläuft. Man kann dies wegen der Unendlichkeit natürlich nicht in allen Orientierungen machen, sondern zerlegt den Raum in n_{INT} Intervalle (DJ,Dj) , die jeweils durch eine diskrete räumliche Orientierung (J,j) des Vektors $\vec{a}$ repräsentiert (und approximiert) werden.



Einteilung der Orientierung in Raumintervalle

Die Genauigkeit dieser Approximation hängt natürlich entscheidend von der Größe der Intervalle (DJ,Dj) ab und man muß im folgenden einen vertretbaren Kompromiß zwischen Genauigkeit und Anzahl der abzuspeichernden Lösungsmöglichkeiten finden. Die Anzahl der zu betrachteten Intervalle n_{INT} kann noch verringert werden, indem man nur die für uns interessanten Zugriffe mit negativer y-Komponenete des Vektors $\vec{a}$ berücksichtigt (kein Zugriff auf Objekte von hinten).

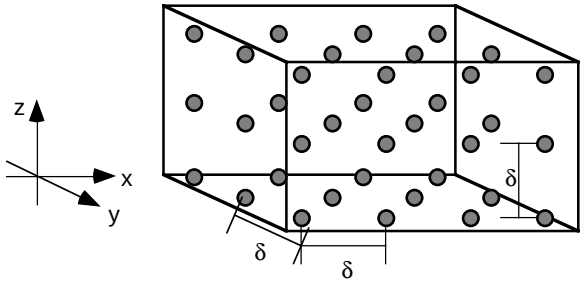
$$\forall \ \vec{a} = (a_x, a_y, a_z)$$
$$\text{mit} \quad a_y \leq 0$$

Die Raumintervalle werden dann codiert *(COI)* um später eine leichte Zuordnung Objekt-Arbeitsraumdaten treffen zu können.

5.3.6. Festlegung des räumlichen Suchintervalls und der Auflösung

Die eigentliche Plazierungsalgorithmus sucht um eine geeignete Anfangsplazierung herum in einem festgelegten Raumintervall die Anzahl aller möglichen Aufstellungen, die bestimmte Kriterien erfüllen. Zu diesem Zweck wird der Raum als

umschreibendes Volumen äquidistanter diskreter Punkte repräsentiert. Der Abstand wird bestimmt durch die Auflösung d [Abb.].



Suchintervall repräsentiert durch diskrete Raumpunkte

Das Suchintervall wird in x-, y-, und z-Richtung durch ganzzahlige Vielfache der Auflösung d angegeben (v_x, v_y, v_z). Damit ergibt sich z.B. für obige Skizze

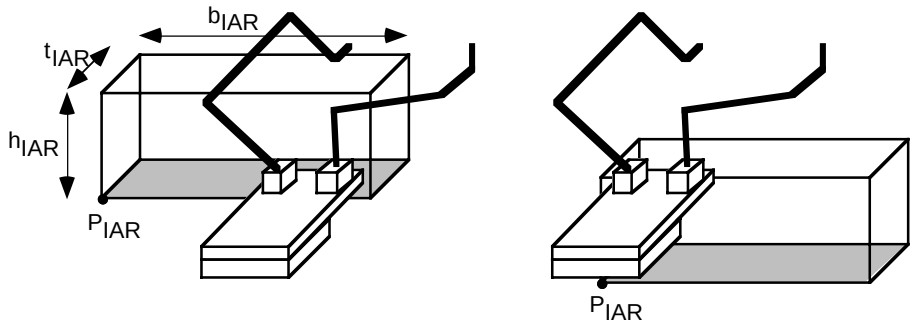
$$v_x = 3 \; ; \; v_y = 2 \; ; \; v_z = 2$$

Es ist leicht ersichtlich, daß durch entsprechende Wahl diese beiden Angaben die Geschwindigkeit der Lösungsfindung erheblich beeinflußen. Die minimale Auflösung d des Suchintervalls entspricht der Auflösung res der abgespeicherten Arbeitsräume und größere d müssen immer ganzzahlige Vielfache von res sein.
Die Spezifikation dieses Suchintervalls durch den Benutzer ist aber nicht unbedingt erforderlich. Das Programm berechnet selbst automatisch ein optimales räumliches Suchintervall und vergleicht dies mit dem des benutzerdefinierten.

5.3.7. Definition des zu berücksichtigenden Teils des Gesamtarbeitsraums des Systems

Je nach Aufgabenstellung und Einsatzort des Systems kann es sinnvoll sein, nur gewisse Teile des Arbeitsraumes als benutzbar zu kennzeichnen. Dies ist z.B. dann notwendig, wenn man Objekte in einem gewissen Mindestabstand vom System bearbeiten will oder wenn man gewisse Bereiche, die mit einer Kamera nicht gut einzusehen sind ausschließen will. Man definiert den zu betrachtenden Arbeitsraum als Schnitt des Gesamtarbeitsraumes mit einem quaderförmigen Volumenkörper *(nutzbarer Arbeitsraum, NA)*. Dieser Quader wird durch Angabe eines Bezugspunktes P_{IAR} und durch Höhe h_{IAR}, Breite b_{IAR} und Tiefe t_{IAR} charakterisiert und im folgenden als *idealisierter nutzbarer Arbeitsraum* (INA)bezeichnet. Jedes Objekt, daß außerhalb dieses nutzbaren Arbeitsraumes liegt, gilt dann als nicht

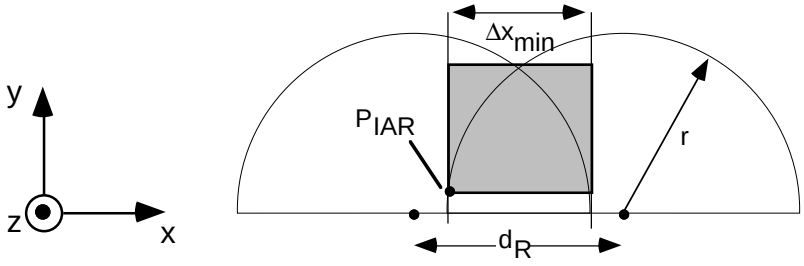
erreichbar, obwohl es vielleicht unter Einbeziehung des Gesamtarbeitsraumes zu manipulieren wäre. Nachfolgende Abbildung zeigt zwei Fälle der Nutzung des linken oberen und des rechten unteren Teils des Gesamtarbeitsraumes.



Beispiel zur Definition des idealisierten nutzbaren Arbeitsraumes des Systems

Aufgrund der Wichtigkeit des gemeinsamen Arbeitsraumes macht man hier die Vorgabe, daß die maximale Ausdehnung der gemeinsamen Arbeitsraumes in x-Richtung nicht eingeschränkt werden darf. Man erhält deshalb für die Größe des idealisierten gemeinsam nutzbaren Arbeitsraumes die Maße (relativ zu Punkt P_{IAR}):

$$\Delta x = \Delta x_{min} = (2r - d_R)$$
$$\Delta y = t_{IAR}$$
$$\Delta z = h_{IAR}$$



Minimale Breite des idealisierten nutzbaren Arbeitsraumes

Wird hier keine Beschränkung definiert, so entspricht der idealisierte nutzbare Arbeitsraum dem umschreibenden quaderförmigen Volumenkörper des Gesamtarbeitsraumes.

5.4. Arbeitsraumbeschreibung unter Berücksichtigung von Orientierungen

5.4.1. Kinematikgleichungen des Roboters

Fast jeder Roboter besteht aus einer Reihe von hintereinandergeschalteten Translations- und Rotationsachsen, die durch eine offene kinematische Kette repräsentiert werden können. Durch die determinierten Bewegungsbereich jeder Achse ergibt sich der für den Roboter erreichbare Arbeitsraum. Die Beschreibung des Arbeitsraumes im besonderen unter Berücksichtigung von Orientierungen führt also über die mathematische Formulierung der kinematischen Kette. Die Komplexität steigt aber sehr rasch mit der Anzahl der Freiheitsgrade an, so daß eine Beschreibung nur mit Hilfe der Vektor- und Matrizenalgebra möglich ist. Eine der häufigst eingesetzten Methoden ist die Matrixmethode von Denavit-Hartenberg [2][4] (DH-Methode). Sie basiert auf den Bewegungsachsen zugeordneten Bezugskoordinatensystemen, wobei zwei benachbarte Bezugssysteme durch eine vorgeschriebene Folge von mindestens zwei Rotationen und zwei Translationen ineinander überführt werden können [11]. Bei Verwendung von homogenen Koordinaten im euklidischen dreidimensionalen Raum lassen sich 4x4 Transformationsmatrizen aufstellen, die die Rotations-, die Translations- sowie die Skalierungs- und Perspektivtransformation enthalten. Eine Transformationsmatrix bildet dann einen Vektor in homogenen Koordinaten von einem Koordinatensystem in ein anderes ab.

Eine 4x4 DH-Transformationsmatrix setzt sich dann aus folgenden Untermatrizen zusammen (Abbildung eines Vektors von Koordinatensystem n nach n+1):

$$DH_{n,n+1} = \begin{bmatrix} R & T \\ P & S \end{bmatrix}$$

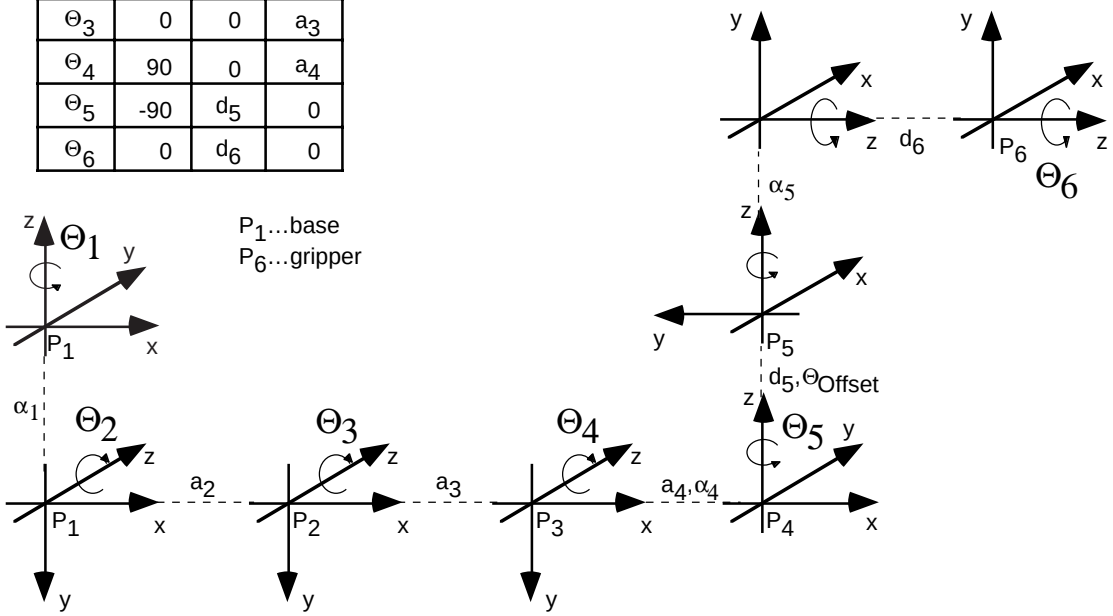
$$\begin{aligned} R &\ldots 3 \times 3 - \text{Rotationsmatrix} \\ T &\ldots 3 \times 1 - \text{Translationsmatrix} \\ P &\ldots 1 \times 3 - \text{Perspektivtransformation} \\ S &\ldots 1 \times 1 - \text{Skalierungsfaktor} \end{aligned}$$

Für tiefergehende Betrachtungen verweise ich auf entsprechende Literatur [4].

5.4.1.1. Vorwärtstransformation

Die Modellierung der kinematische Kette nach Denavit-Hartenberg erfolgt nach gewissen Konventionen durch das Festlegen von Koordinatensystemen in den Bewegungsachsen.

Θ_i	α_i	d_i	a_i
Θ_1	-90	0	0
Θ_2	0	0	a_2
Θ_3	0	0	a_3
Θ_4	90	0	a_4
Θ_5	-90	d_5	0
Θ_6	0	d_6	0

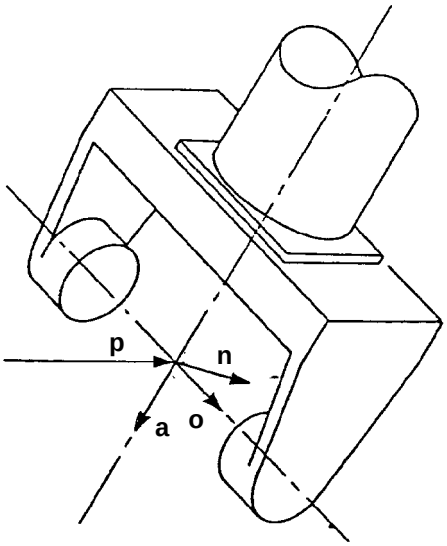


Kinematische Kette des Roboters GRIPS in der DH-Konvemtion

Nach Festlegen der entsprechenden DH-Parameter (a_i, d_i, a_i) und durch Multiplikation der Matrizen erhält man wieder eine 4x4-Matrix *(Endeffektorframe)*, die Position und Orientierung des Greifers in Weltkoordinaten (Bezugssystem) enthält.

$$E = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Der Vektor (p_x,p_y,p_z) repräsentiert die Position und die n-, o-, und a-Parameter die Orientierung. Dabei steht n für *normal*, o für *orientation* und a für *approach*. Diese Bezeichnung werden an folgender Abbildung sehr gut deutlich.



Greifereigenes Koordinatensystem mit Vektoren n, o, a

Es wurde jetzt eine mathematische Methode gefunden, von bekannten Gelenkwinkeln des Roboters auf die Position und Orientierung des Greifers zu schließen. Dies ist die sogenannte Vorwärtstransformation. Was uns aber vielmehr interessiert ist das inverse kinematische Problem (Rücktransformation), d.h. von einer gegeben Position in kartesischen Koordinaten auf die Winkelstellungen der einzelnen Achsen zu schließen. Dies ist im allgemeinen ungleich viel komplizierter, da die Lösung durch Inversion der Matrix mit anschließender Lösung des Gleichungssystems bei 6 und mehr Freiheitsgraden unmöglich oder zumindest in annehmbarer Zeit nicht durchführbar ist (z.B. transzendentes Gleichungssystem mit 12 Gleichungen und 6 Unbekannten).

5.4.1.2. Rückwärtstransformation

Um zu einer geschlossenen Formulierung des inversen Problems zu kommen, die im übrigen auch für die Modellierung des Roboters im Simulationsprogramm benötigt wird, wendet man ein Verfahren an, daß 1979 von Richard P. Paul [6] ausgehend von der Denavit-Hartenberg Formulierung vorgeschlagen wurde. Das Verfahren beruht auf einer sukzessiven Multiplikation mit den inversen DH-Matrizen und anschließender Bestimmung von Winkel oder Winkelsummen ausgehend von den bekannten Werten des Endeffektorframes. Mit den Werten des Endeffektorframes ergibt sich wie folgt:

$$\Theta_1 = \arctan\left(\frac{p_y}{p_x}\right)$$

$$(\Theta_2 + \Theta_3 + \Theta_4) = \arctan\left(\frac{a_z}{C_1 a_x + S_1 a_y}\right)$$

$$\Theta_3 = \arccos\left(\frac{(C_1 p_x + S_1 p_y - a_4 C_{234} - d_5 S_{234})^2 + (-p_z - a_4 S_{234} + d_5 C_{234})^2 - a_2{}^2 - a_3{}^2}{2 a_2 a_3}\right)$$

$$\Theta_{23} = -\arctan\left(\frac{e}{f}\right) + \arctan\left(\frac{d}{-\sqrt{e^2 + f^2 - d^2}}\right)$$

e, f, d … siehe ausführliche Rechnung im Anhang

$$\Theta_4 = \Theta_{234} - \Theta_{23}$$

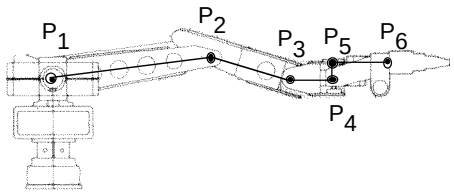
$$\Theta_2 = \Theta_{23} - \Theta_3$$

$$\Theta_6 = \arctan\left(\frac{C_4(C_{23}[C_1 o_x S_1 o_y] + S_{23}[-o_z]) + S_4(-S_{23}[C_1 o_x S_1 a_y] + C_{23}[-o_z])}{C_4(C_{23}[C_1 n_x S_1 n_y] + S_{23}[-n_z]) + S_4(-S_{23}[C_1 n_x S_1 n_y] + C_{23}[-n_z])}\right)$$

Das ausführlich durchgerechnete Verfahren findet man im Anhang B. Ein besondere Schwierigkeit bei diesem Verfahren ist die Vermeidung der Division durch einen Sinus oder Cosinus, da bei entsprechenden Argumenten den Funktionswert Null annehmen. Auch mußte man im weiteren einige Fallunterscheidungen machen, um entsprechende Vorzeichenkorrekturen vorzunehmen, da einige trigonometrischen Funktionen nur in einem bestimmten Intervall eindeutig Gültigkkeit besitzen.

5.4.1.3. Test durch Modellierung im Robotersimulationssystem TOROS

Um diese doch schon recht aufwendigen Rechnungen zu überprüfen, nutzte man außer des Mathematikprogramms MAPLE noch die Möglichkeit der physischen sowie kinematischen Modellierung des Roboters im Simulationssystem TOROS [9]. Dieses Simulationssystem ist auf eine geschlossene mathematische Formulierung der Inverse angewiesen, führt aber die Vorwärtstransformation mittels Matrizenmultiplikation selbständig durch (Multiplikation der DH-Matrizen). So kann man direkt durch Auslesen der Parameter des Endeffektorframes die Richtigkeit der gefundenen Gleichungen der Vorwärtstransformation sowie auch durch Trajektorieneingabe in kartesischen Koordinaten die Rücktransformation überprüfen. Voraussetzung ist hier allerdings, daß die Modellierung der kinematischen Kette unter den schon festgelegten Denavit-Hartenberg-Konventionen erfolgt. Um eine Vorstellung von der Lage der *frames* in den Achsen des Roboters zu geben, sind sie hier einmal in Bezug zum Roboter aufgezeigt. Die Definition der einzelnen mechanischen Teile des Roboters in Form von Polyedern, die aus in einer der Achsenrichtungen wachsenden Polygonen bestehen, findet man ebenso wie die in C codierten inversen Kinematikgleichungen in den Anhängen (B.2/B.3). Mit diesen Daten erzeugt TOROS eine dreidimensionale Representation des Roboters, die man in Anhang B.9 sieht.



Lage der 'frames' des Roboters GRIPS

5.4.2. Repräsentation des Raumes als 3D-Matrix

Im folgenden interessiert uns eine brauchbare Formulierung des Arbeitsraumes, die eine möglichst schnelle Aussage zuläßt, ob ein *objectlocation* erreicht werden kann oder nicht. Die Beschreibung des Arbeitsraumes hängt eng mit den Lösungen des inversen kinematischen Problems zusammen, da diese nur innerhalb des Arbeitsraumes des Roboters Gültigkeit besitzen [5].

Eine geschlossene Formulierung des Raumes durch ein Volumenintegral ist im allgemeinen nur für sehr einfache Kinematiken oder mit erheblichen Einschränkungen angebbar, zumal hier keine Ansätze existieren, verschiedene Orientierungen einzubeziehen.

Es bleibt hier also nur die Möglichkeit die Parameter jedes *objectlocation* in die inversen kinematischen Gleichungen einzusetzen und zu prüfen, ob sie die Gleichungen im Gelenkintervall erfüllen. Das Problem dieser Methode liegt darin, daß es relativ lange Zeit in Anspruch nimmt die komplexen trigonometrischen Gleichungen zu berechnen. Da der Positionieralgorithmus den Raum sequentiell in diskreten Abständen d nach einer geeigneten Positionierung absucht, müßte man z.B. für ein Suchintervall mit $v_x = v_y = v_z = 20$ für 30 Objekte ≈240000-mal diese Gleichungen berechnen. Man kann sich vorstellen, daß dies einen nicht mehr zu tolerierenden Zeitbedarf darstellen würde. Um dies zu vermeiden wird die folgende Methode angewandt:

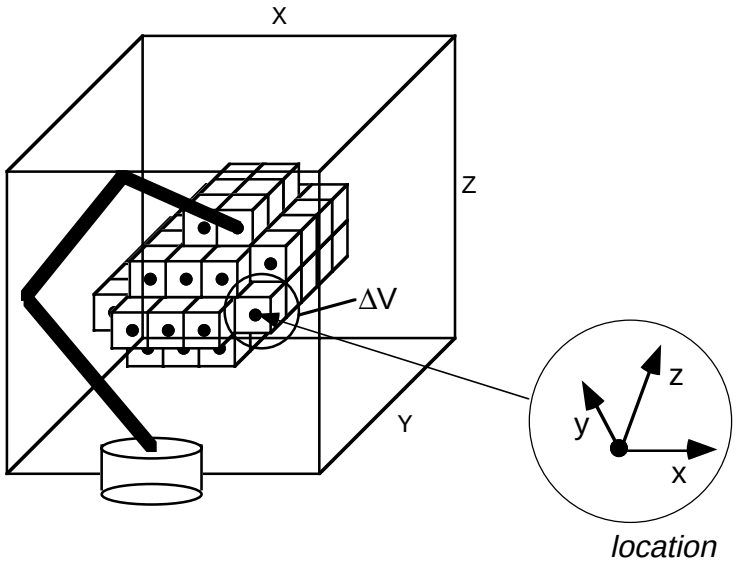
- Diskretisierung des Arbeitsraumes in Volumenelemente DV, die durch ein *location* im Mittelpunkt repräsentiert werden. Wenn dieses *location* erreichbar ist, gilt auch jeder andere Raumpunkt in DV unter gleicher Orientierung als erreichbar
- a priori-Berechnung von Arbeitsräumen unter verschiedenen Orientierungen mit anschließender platzoptimaler Speicherung.

Die Abstände der Mittelpunkte sind in x-, y- und z-Richtung äquidistant. Mit Hilfe der Auflösungen res_i kann man es wie folgt formulieren

$$res_x = res_y = res_z = res$$

Mit den Dimensionen X, Y und Z des Arbeitsraumes, erhält man somit n_V Volumenelemente.

$$n_V = \frac{X}{res_x} \cdot \frac{Y}{res_y} \cdot \frac{Z}{res_z} = res^{-3} \cdot XYZ$$



Diskretisierung des Arbeitsraumes in Volumenelemente DV

Wenn man Orientierungen bei der Beschreibung des Arbeitsraumes berücksichtigt, erhält man im allgemeinen Körper, die nicht mehr rotationssymetrisch und stark konvex sind. Eine Sequentialisierung des Problems durch eine Zerlegung des Arbeitsraumes in zwei zweidimensionale Flächen ist deshalb hier wegen der sich ergebenden enormen Ungenauigkeiten nicht mehr möglich. Es bleibt hier also nur eine volumenorientierte Beschreibung des Arbeitsraumes.

Diese Art der Speicherung bringt natürlich den Nachteil mit sich, daß der Speicherbedarf bei steigender Auflösung res schnell sehr groß wird. Man wird sich also Schnelligkeit und geringen Speicherbedarf immer mit einer relativ großen Diskretisierung und damit geringerer Genauigkeit erkaufen müssen. Dennoch scheint es, wie auch überschlägige Berechnungen in den nächsten Kapiteln zeigen, daß man durch geeignete Speichertechniken mit wenigen kbyte Speicherplatz pro orientierungsbezogenem Arbeitsraum auskommt, was bei heutiger Hauptspeicherausstattung von mehreren Megabytes (Tendenz steigend) keinerlei Einschränkung mehr darstellt.

Diese Speicherung in einer 3D-Matrix könnte dann auch zur Visualisierung des Arbeitsraumes genutzt werden (ähnlich der Gehirntomographie).

In unserem speziellen Fall sind, durch die endliche Aufteilung in diskrete Orientierungen, alle Parameter des Endeffektorframes außer der Position bestimmt.

Die eigentliche Berechnung beruht auf einem sukzessivem Einsetzen der Position aller Mittelpunkt der n_V verschiedenen Volumenelemente. Damit sind dann alle

Parameter des Endeffektorframes bekannt und man kann dann das entsprechende *location* auf Erreichbarkeit überprüfen. Zur Speicherung des Ergebnisses genügen die diskreten Werte 0 (nicht erreichbar) oder 1(erreichbar). Folgende Skizze zeigt einen Schnitt durch einen diskretisierten Arbeitsraum.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0
0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0
0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0
0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0
0	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	**1**	**1**	**1**	**1**	0
1	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	**1**	**1**	**1**	**1**
1	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	**1**	**1**	**1**	**1**
1	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0	**1**	**1**	**1**	**1**
1	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	**1**	**1**	**1**
1	**1**	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	1	0
0	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0
0	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Schnitt durch einen diskretisierten Arbeitsraum

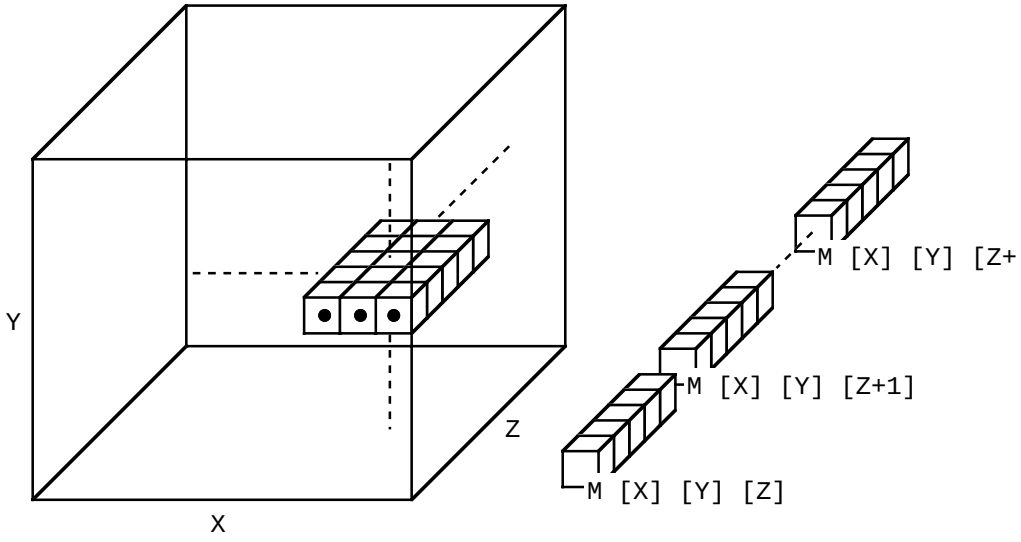
Man erhält dann als Ergebnis eine Arbeitsraumbeschreibung für jede der n_{INT} Orientierungen. Eine solche Zerlegung eines Raumes wird in der Bildverarbeitung auch als Raumbeschreibung durch VOXEL *(volume pixel)* genannt, die ähnlich zu der bekannten Zerlegung eines 2-dimensionalen Bildes in PIXEL, diese Definition in die dritte Dimension ausdehnt.

Im Anhang B.8 sind einige Arbeitsräume mittels einer Beschreibung durch VOXEL visualisiert (AutoCAD). Die Berechnung erfolgte mit einem MAPLE-Programm, dessen Programmcode sich ebenfalls im Anhang (B.7) befindet.

5.4.3. Speichertechniken

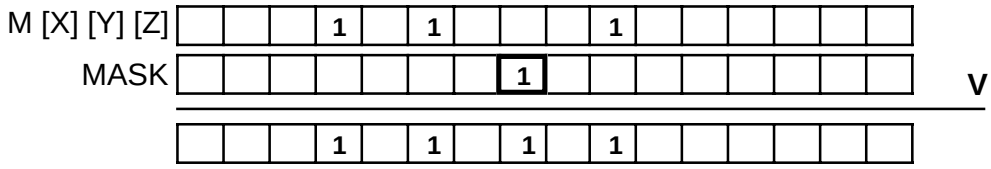
5.4.3.1. Bitweise Speicherung

Da der Arbeitsraum pro Volumenelement nur durch die diskreten Werte 0 und 1 beschrieben wird, scheint es wenig sinnvoll die Werte 1 und 0 jeweils als ganze Zahlen durch zwei oder sogar vier Byte zu speichern. Es liegt daher nahe, durch bitweises Einschreiben den Speicher besser zu nutzen und so den Gesamtspeicherbedarf erheblich zu reduzieren. Im folgenden geht man davon aus, daß x-, y-, und z-Variablen mit jeweils 4 Byte (32 bit) sind. Man definiert hierfür eine dreidimensionale Matrix M mit den Dimensionen X, Y und Z, die von der Größe des Gesamtarbeitsraumes und von der Größe der Volumenelemente DV abhängen. Man kann nun sehr einfach jedes der 32 bit einer Dimension der Matrix für die diskrete Beschreibung nutzen, indem man mit einer entsprechend definierten Maske an der richtigen Stelle die diskreten Werte einschreibt oder ausliest. Nachfolgende Abbildung soll die Nutzung der einzelnen Bits in der Dimensionsvariablen z verdeutlichen.



Bitweise Speicherung

Die Maske ist, bis auf die gewünschte Stelle, nur mit Nullen belegt. Man kann so durch einfache boolsche Operationen ein Bit setzten (SET) oder rücksetzen (RESET). Die SET Operation durch eine Oder-Verknüpfung ist in untenstehender Abbildung graphisch verdeutlicht (leere Felder entsprechen logischen Nullen).



Ein Beispiel für die Realisierung von SET, RESET und TEST zeigt folgender C-Code (kein vollständige Programmstruktur)

```
/*************************************************************/
short int x,y ;
long int z ;
int a,b ;
boolean VOXEL ;

a = z / 32;
b = z % 32;

Mask = 1 << b ;
W = M [x][y][a] ;

/*** SET ***/
W |= Mask ;

/*** Reset ***/
W &= Mask ;

/*** TEST if VOXEL is true***/

if ((W & Mask) != 0)

/*************************************************************/
```

Eine andere Möglichkeit diese Struktur in C zu realisieren ist die Definition von sogenannten Bitfeldern. Der Vorteil liegt hier in dem einfachen Zugriff auf die gewünschten Bits.

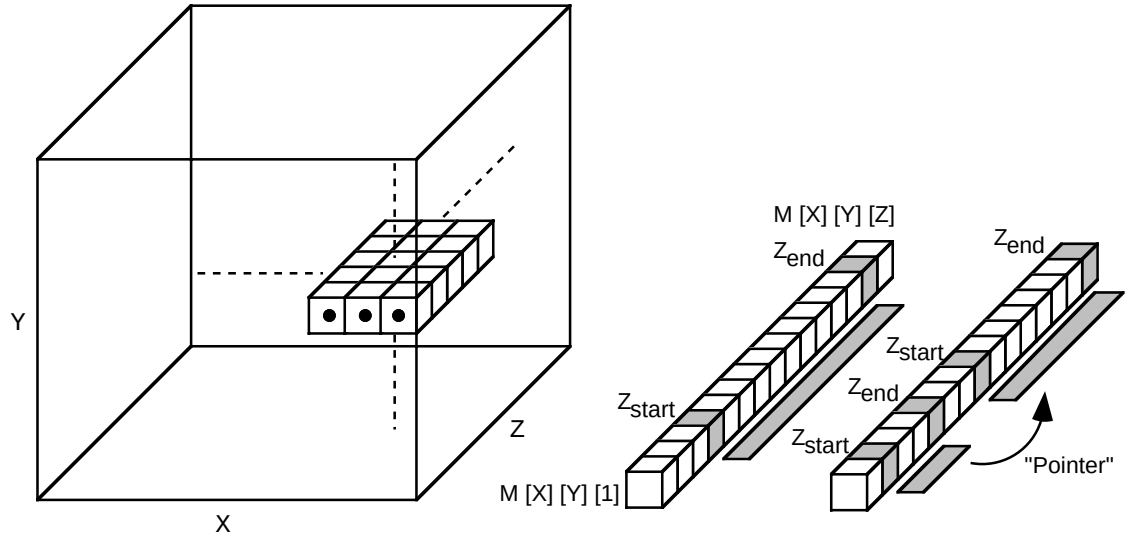
```
/*********************************************************/
struct BIT {
                        unsigned bit_1 : 1;
                        unsigned bit_2 : 1;
                        unsigned bit_3 : 1;
                        …
                        unsigned bit_32 : 1;
        };

struct BIT M[x_max][y_max][z_max]

/*** SET Bit_n ***/
M[x][y][a].bit_n = 1 ;
/*** RESET Bit_n ***/
M[x][y][a].bit_n = 0 ;
/*********************************************************/
```

Mit einer Auflösung von 5cm benötigt man unter Verwendung der bitweisen Speichertechnik für den Arbeitsraum des Roboters GRIPS ungefähr 5 kbyte Speicherplatz.

5.4.3.2. Speicherung in Pointerstruktur

Der Arbeitsraum einer Orientierung wird aufgrund der Kinematik der Roboter ein zusammenhängender Raum sein. Dies bedeutet erreichbare Volumenelemente nie vereinzelt auftreten und mindestens in eine der Raumrichtungen wieder erreichbare Nachbarvolumenelemente besitzen. Man kann dieses häufige Auftreten von benachbarten Einsen dadurch ausnutzen, daß man nur Anfang und die Ende dieser Reihe von Einsen speichert und annimmt, daß innerhalb dieses Intervalls alle Volumenelemente erreichbar sind. Zur Speicherung eignet sich hier besonders ein Array von Pointerstrukturen [Abb.]. Hier belegt jeder Arbeitsraum (res=5cm) näherungsweise 20 kbyte Speicherplatz. Obwohl hier der Speicherbedarf gegenüber

der bitweisen Speicherung größer ist, kann die Pointersruktur bei höherer Auflösung und bei einer ausgeprägten z-Dimension Vorteile bieten.



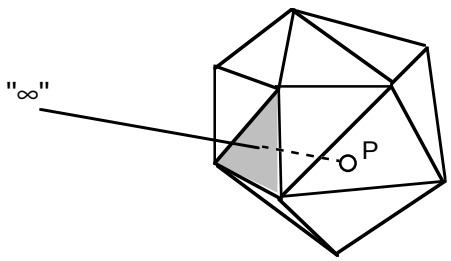
Speicherung in Pointerstruktur

Ein Beispiel einer Implementierung dieser Struktur zeigt folgender C-Code (kein vollständige Programmstruktur)

```
/*********************************************************/
boolean first_element
typedef struct pnode {
                    int Z_start, Z_end ;
                    struct pnode *next ;
                    } PNODE ;
first_element = 1;
PNODE *M[X][Y] ; *aux ; * aux_old

for (x=x_min; x<x_max;x++) {
          for (y=y_min; y<y_max;y++) {
          aux_old = malloc(sizeof (PNODE) )
            for (z=z_min; z<z_max;z++) {
                                        if (start_reachable(x,y,z) == 1)
                                                { aux=malloc( sizeof (PNODE))
```

```
                                        aux -> Z_start = z
                                        if (first_element)
                { first_element = 0;
                                                aux_old = M[x][y] -> next = aux ;
                                                }
                                        else
                                                aux_old -> next = aux  }
                                if (end_reachable(x,y,z) == 1)
                                        aux -> Z_end = z ;
                                        }
                                }
                        }
/*********************************************************/
```

Eine weitere denkbare Möglichkeit wäre die Speicherung der Einhüllenden. Da diese Art der Arbeitsraumbeschreibung nicht mehr volumen- sondern oberflächenorientiert ist, muß man in Verfahren finden um zu prüfen, ob ein Punkt innerhalb oder außerhalb der Einhüllenden liegt. Folgende Skizze zeigt eine Einhüllende beschrieben durch Dreiecksflächen (*triangulation* - sehr häufig angewandte Methode)



Einhüllende in Form von Dreiecksflächen

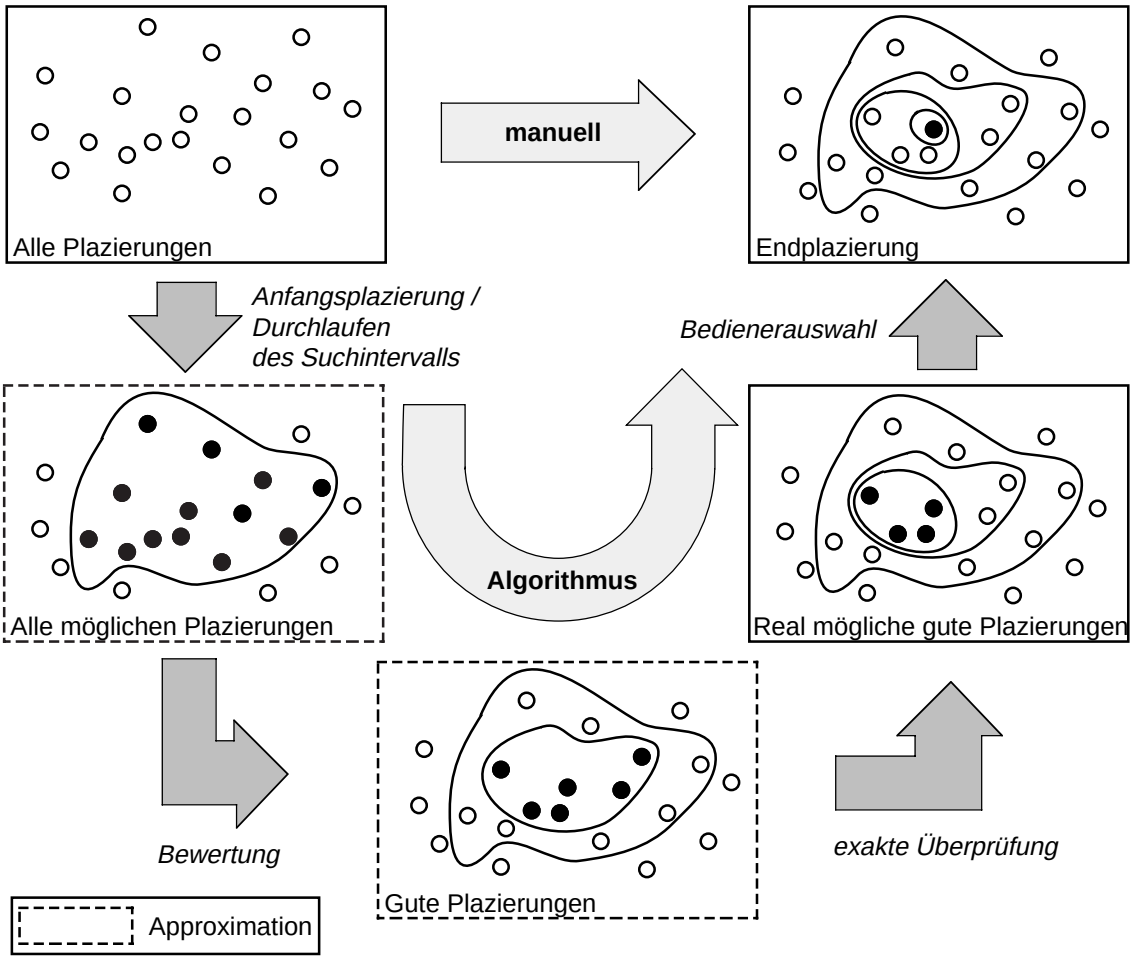
Ein Verfahren zur Überprüfung ob ein Punkt innerhalb oder außerhalb des beschriebenen Raumes liegt wäre die Bestimmung der Anzahl, wie oft eine Gerade von einem sehr weit entfernten Punkt zum Testpunkt ein Flächenelement schneidet. Bei ungeradzahliger Anzahl von Schnittpunkten liegt der Punkt innerhalb und bei geradzahliger Anzahl außerhalb.
Einige Gründe, wieso dieses Vefahren hier nicht geeignet ist, sind:

- Exakte Verfahren zur Berechnung der Hülle (*contour tracing* fi *boundary detection* fi *triangulation*) sind kompliziert und rechenaufwendig
- Der Test, ob ein Punkt innerhalb oder außerhalb der Einhüllenden liegt ist relativ zeitintensiv, da jedes der Flächenelemente auf Schnitt überprüft werden muß
- Keine Einsparung an Speicherplatz bezüglich Pointerlösung

5.5. Positionieralgorithmus

5.5.1. Struktur

Der Algorithmus hat die Aufgabe, dem Benutzer eine Reihe von sinnvollen Plazierungen für das System vor dem Mast vorzuschlagen. Eine manuelle Plazierung des Systems durch den Menschen erfolgt auf direktem Wege unter Einbeziehung von Erfahrung und eingeschränktem Ausprobieren und ist in den meisten Fällen suboptimal. Wenn man eine mögliche Positionierung findet, kann man aber nur sehr schwer einschätzen, ob es weitere, vielleicht sogar bessere Lösungen gibt. Diese Nachteile vermeidet man durch eine sequentielle hierarchische Strukturierung des Problems.
Schematisch läßt sich also die Findung einer geeigneten Endpositionierung folgendermaßen darstellen:



Struktur des Positionieralgorithmus

Durch sukzessive Variation der Position des Systems in einem bestimmten Suchintervall sucht man zuerst alle möglichen Positionierungen. Ausgehend davon schränkt man aufgrund bestimmter Kriterien diese Lösungsmenge immer mehr ein, bis schließlich eine für den Menschen überschaubare und bewertbare Menge von Vorschlägen übrig bleibt.

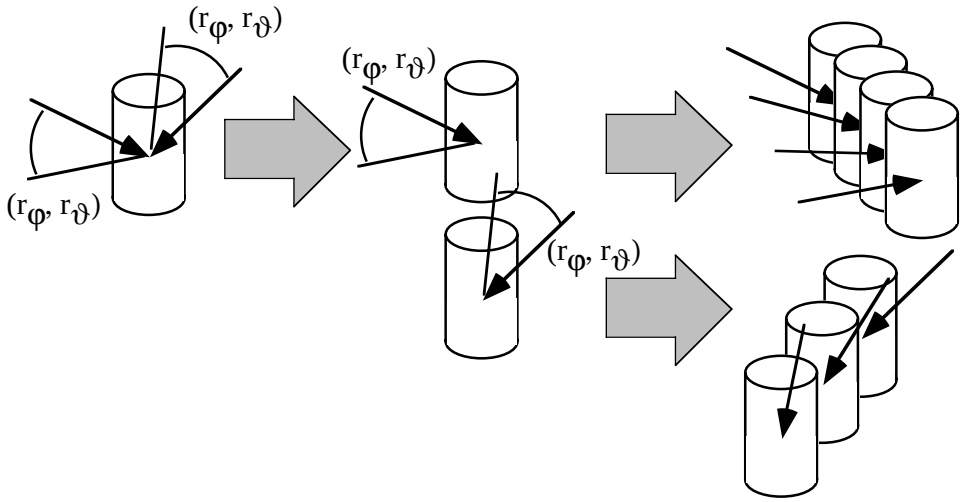
5.5.2. Aufbereiten der Daten

5.5.2.1. Zerlegung in Einzelobjekte

Der eigentliche Plazierungsalgorithmus verarbeitet nur die Daten von Einzelobjekten mit eindeutiger Position, Orientierung, Roboterzuordnung, Gewichtung und Objektart. Dies bedeutet, daß im ersten Schritt alle Objektdaten vom bisherigen Eingabeformat in Einzelobjekte mit den oben genannten Parametern überführt werden. Folgende Zerlegungen sind nötig:

- Zerlegung eines *ZAO* in zwei *ZAOs* an derselben Position, aber mit fester Zuordnung zu einem Roboter
- Auflösen der Objektparameter Translation und Rotation in eine diskrete Anzahl von Objekten mit jeweils eindeutiger Position und Orientierung

Wie schon am Anfang angedeutet, soll hierauf aber nicht weiter eingegangen werden. Folgende Skizze zeigt schematisch die Zerlegung eines *ZAOs* in zwei *ZAOs* jeweils für einen Roboter und danach die Zerlegung der Rotation in eine diskrete Anzahl von Einzelobjekten mit eindeutiger Greiforientierung. Da hier im Beispiel kein Translationsvektor berücksichtigt wird, haben alle Objekte dieselben Raumpositionen.



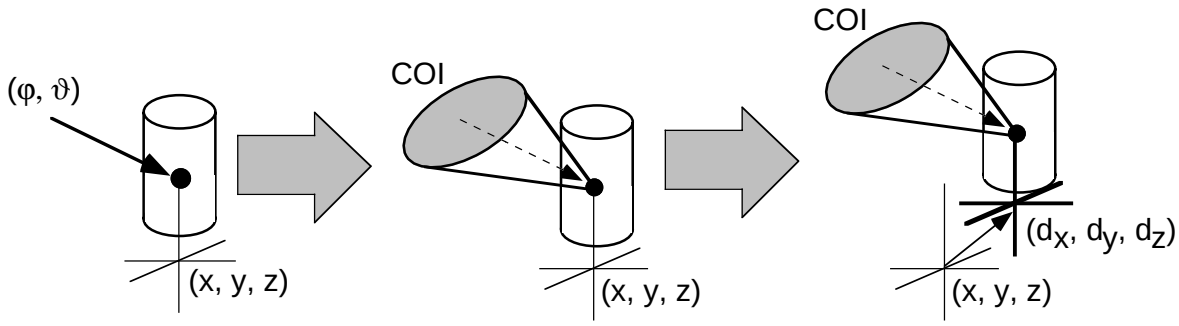
Zerlegung eines ZAO in Einzelobjekte mit eindeutiger Greiforientierung

5.5.2.2. Einteilung der Objekte in das Positionsraster und in die diskreten Raumintervalle

Da die Speicherung des Arbeitsraumes diskret in der Auflösung res erfolgte, müssen natürlich auch die Objekte für einen schnellen Ablauf des Algorithmus in dieses Raster einzuordnen sein. Dazu wird der Gesamtraum ebenfalls in Volumenelemente der Auflösung res zerlegt und jedes *objectlocation* erhält als zusätzliche Objektparameter die Raumdaten (dx, dy, dz) des das Volumenelement repräsentierenden Mittelpunktes. Der Test, in welchem Volumenelement des Arbeitsraumes das Objekt liegt, könnte natürlich auch jedesmal neu im Algorithmus

durchgeführt werden. Diese Möglichkeit würde aber die Geschwindigkeit des Algorithmus erheblich reduzieren.

Bei der Beschreibung des Arbeitsraumes wurde, um die abzuspeichernde Datenmenge erträglich zu halten, die Vereinfachung gemacht, daß ein Intervall von Raumorientierungen durch eine Orientierung repräsentiert wird. Es ist also im folgenden zusätzlich erforderlich, jedes Objekt entsprechend dem Raumintervall zuzuordnen, in welchem sich die Objektorientierung befindet. Diese Information wird dann entsprechend kodiert *(COI)* als Objektparameter mit aufgenommen.



Einteilung des Objekts in ein Orientierungsintervall und auf Rasterposition

Aus den genannten Aufbereitungen der Objektdaten erfolgt eine Menge an Objekten, die durch die folgenden Parameter beschrieben werden:

exakte Pos.			diskrete Pos.			exakte Orient.		g	COT	COI	Roboter	
x	y	z	d_X	d_Y	d_Z	j	J				R_L	R_R
-	-	-	-	-	-	-	-	-	*EAO*	-	*1*	*1*
-	-	-	-	-	-	-	-	-	*EAO*	-	*0*	*1*
-	-	-	-	-	-	-	-	-	*EAO*	-	*1*	*0*
-	-	-	-	-	-	-	-	-	*ZAO*	-	*0*	*1*
-	-	-	-	-	-	-	-	-	*ZAO*	-	*1*	*0*

5.5.2.3. Laden der entsprechenden Arbeitsraumdaten in den Hauptspeicher

Da die Suche einer Plazierung nicht nur auf sehr leistungsfähigen Rechnern im Labor, sondern auch am Einsatzort erfolgen soll, ist dies bei der Entwicklung der Algorithmen zu berücksichtigen. Da die Arbeitsraumdaten bitweise in Form von

VOXELN vorliegen, bietet sich hier besonders die Möglichkeit schneller Speicheroperationen besonders an. Dabei wird nur eine bestimmte Speicherstelle auf 0 oder 1 überprüft. Bei genügend großer Hauptspeicherausstattung werden außerdem alle benötigten Arbeitsräume vom entsprechenden Speichermedium geladen, um während der eigentlichen Plazierungssuche das langsame Nachladen zu verhindern.

5.5.3. Mehrstufiger Plausibilitätstest

Bevor mit dem eigentlichen Plazierungsalgorithmus begonnen wird, sollte erst einmal überprüft werden, ob die gewählte Eingabe der Objekte überhaupt zu einer Lösung des Plazierungsproblems führen kann. Dies bringt einen erheblichen Zeitvorteil, da sinnlose Positionseingaben sofort erkannt und vom System nicht akzeptiert werden. Der Benutzer wird dann zur erneuten Eingabe aufgefordert. Zur Überprüfung betrachtet man nur die Position der eingegebenen Objekte getrennt nach *EAO* und *ZAO* und leitet dazu zwei notwendige Kriterien zur möglichen Plazierung her. Wurden bei der Formulierung des ersten Kriteriums noch erhebliche vereinfachende Voraussetzungen gemacht, so liegt dem zweiten Kriterium schon eine ziemlich genaue Beschreibung des Arbeitsraumes zugrunde (sukzessive Verfeinerung der Betrachtungen). Somit erkennt man das Scheitern einer Plazierung für die gegebenen Eingangsdaten zum frühestmöglichen Zeitpunkt.
Aus dem erfolgreichen Test der Benutzereingabe läßt sich natürlich keine zwingende Existenz einer geeigneten Plazierung ableiten, da man die Greifrichtung der Objekte nicht in Betracht zieht. Dennoch eliminiert man Eingaben, die zu keiner Lösung führen können, schon vor dem zeitintensiven Durchlaufen des Plazierungsalgorithmus.

5.5.3.1. Objekte passen nicht in den INA (erstes notwendiges Kriterium)

Die Objektpositionen spannen einen Raum auf, der in seiner Größe in erster Linie von seinen Extrempunkten in x, y und z-Richtung charakterisiert wird. Diese Punkte sind für die x-Richtung mit K_{min}, K_{max} , für die y-Richtung mit L_{min}, L_{max} und für die z-Richtung mit M_{min}, M_{max} bezeichnet. Entsprechend heißen sie für den Zweiarm-Objekttyp K_coord_{min}, K_coord_{max}, L_coord_{min}, L_coord_{max}, M_coord_{min} und M_coord_{max} [Abb.]

Für den ersten Test, ob die Punkte überhaupt ohne Verfahren des Systems erreicht werden können, macht man folgende idealisierte Annahmen. Die Objektpositionen werden, wie auch der Arbeitsraum des Roboters, durch kubische Volumenkörper umschrieben. Ausgehend von diesen Annahmen läßt sich für die Erreichbarkeit der Objete folgendes notwendiges Kriterium ableiten:

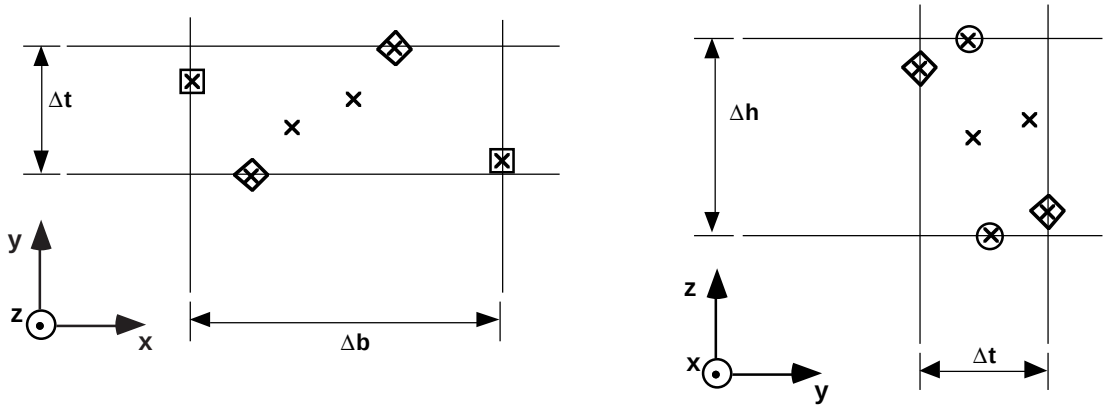
- Die Dimensionen des die Objekte umschreibenden Volumenkörpers (Db,Dh,Dt) müssen kleiner sein als die des idealisierten nutzbaren Arbeitsraumes.

1. $(\Delta b \le b_{IAR}) \,\&\, (\Delta t \le t_{IAR}) \,\&\, (\Delta h \le h_{IAR})$
2. $(\Delta b_{coord} \le (2r - d_R)) \,\&\, (\Delta t_{coord} \le t_{IAR}) \,\&\, (\Delta h_{coord} \le h_{IAR})$

Ausgehend von den oben bestimmten Extrempunkten ist es sehr einfach Breite, Tiefe und Höhe zu bestimmen.

$$\Delta b = K_{max} - K_{min} \quad \text{bzw.} \quad \Delta b_{coord} = (K\_coord_{max} - K\_coord_{min})$$
$$\Delta t = L_{max} - L_{min} \qquad \Delta t_{coord} = (L\_coord_{max} - L\_coord_{min})$$
$$\Delta h = M_{max} - M_{min} \qquad \Delta h_{coord} = (M\_coord_{max} - M\_coord_{min})$$

Extrempunkte, Breite, Höhe und Tiefe sind in nachfolgender Abbildung noch einmal verdeutlicht, die zwei Senkrechtprojektionen des betrachteten Raumbereichs darstellt.



⊠ ... K_{min} , K_{max} ⊗ ... L_{min} , L_{max} ⊗ ... M_{min} , M_{max}

Existiert nur ein oder kein Objekt für eine Zweiarmmanipulation *(ZAO)*, so sind die Größen Db_{coord}, Dh_{coord}, Dt_{coord} null oder nicht definiert und man kann diese deshalb nicht mehr in den Algorithmus einbeziehen. Diese beiden Fälle müssen

deshalb immer dort getrennt betrachtet werden, wo der Algorithmus auf diese Größen aufbaut.

5.5.3.2. Objekte passen nicht in den NA (zweites Kriterium)

Die Betrachtung für das erste Kriterium ist aufgrund der Beschreibung des eigentlich rotationssymetrischen Arbeitsraumes des Roboters als Quader recht idealisiert. Für das zweite Kriterium legt man deshalb die Beschreibung des Arbeitsraumes zugrunde, die vom Hersteller des Roboters in Form zweier Projektionsansichten des Arbeitsraumes angegeben wird. Aufgrund der Rotationssymetrie muß man hier eine Transformation in Zylinderkoordinaten durchführen. Man kann deshalb das Kriterium folgendermaßen angeben:

- Alle Objekte müssen in der xy- *und* in der rh-Projektionsebene liegen

Die Überprüfung läßt sich entsprechend der beiden Projektionsebenen sequentialisieren, d.h. man sucht erst eine Plazierung aller (projezierten) Objektpositionen in der xy- (Variation in Richtung x und y) und dann in der rh-Projektionsebene (Variation in Richtung z). Natürlich muß man wiederum prüfen, ob die Objekte auch im definierten idealisiert nutzbaren Arbeitsraum liegen.

Im folgenden soll kurz der Algorithmus erläutert werden. Die gegebene Arbeitsraumbeschreibung in zwei Ebenen wird hierzu analytisch durch Kreis und Geradengleichungen beschrieben. Diese Kreis- und Geradendefinitionen kann man dem Anhang B.4 entnehmen.

Für die Anfangsplazierung unterscheidet man jeweils folgende drei Fälle mit den Bezeichnungen:

1.) $\Delta b_{coord} \neq 0$ .....mehrere *ZAOs*

2.) $\Delta b_{coord} = 0$ .....ein Objekt *ZAO*

3.) Δb_{coord} nicht definiert .....kein Objekt *ZAO*

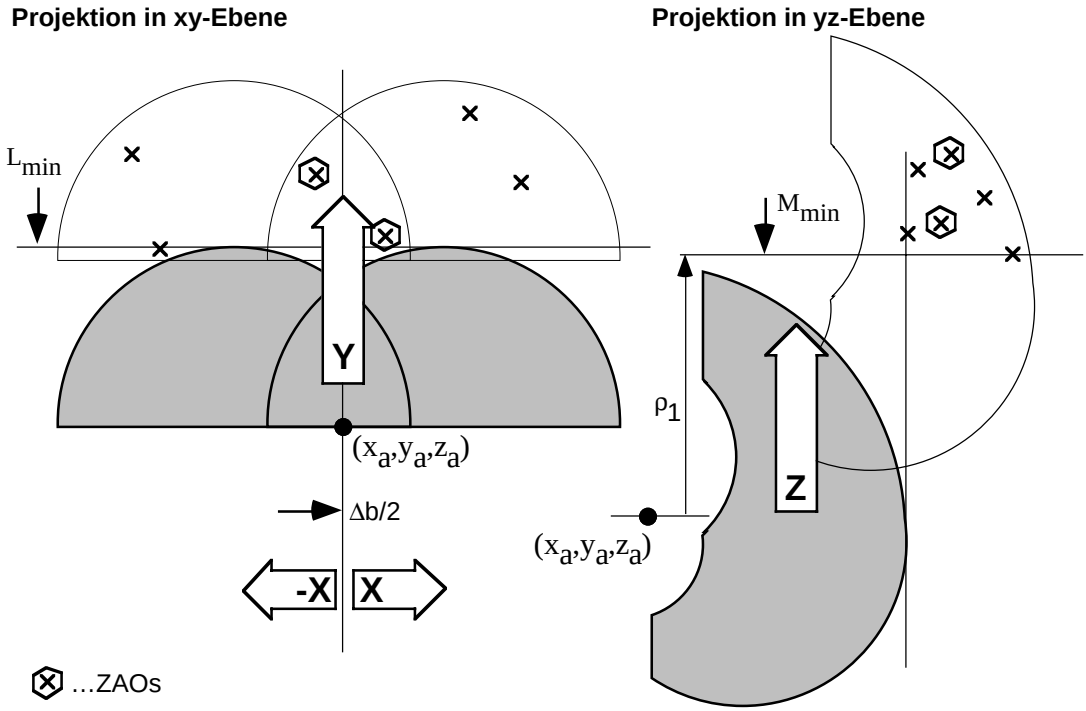
- Anfangsplazierung bei keiner Einschränkung des Gesamtarbeitsraumes

$\Delta b_{coord} \neq 0$ [Abb.]	$x_a = x_{K\_coord\,max} - \Delta b_{coord} / 2$ $y_a = y_{L\,min} - \rho_1$ $z_a = z_{M\,min} - \rho_1$
$\Delta b_{coord} = 0$	$x_a = x_{coordObj}$ $y_a = y_{L\,min} - \rho_1$ $z_a = z_{M\,min} - \rho_1$
Δb_{coord} nicht definiert	$x_a = x_{K\,max} - \Delta b / 2$ $y_a = y_{L\,min} - \rho_1$ $z_a = z_{M\,min} - \rho_1$

- Anfangsplazierung bei einer Einschränkung des Gesamtarbeitsraumes (INA definiert)

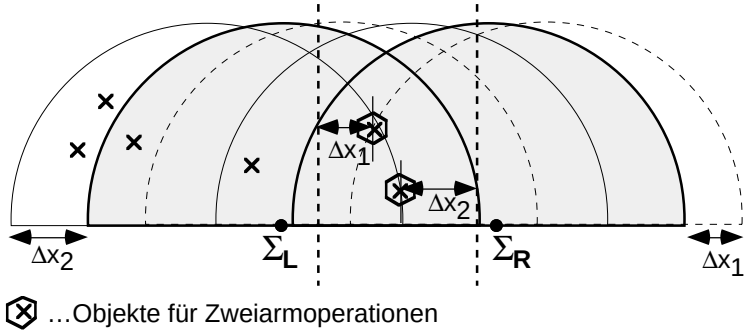
$\Delta b_{coord} \neq 0$	$x_a = x_{Kcoord\,max} - \Delta b_{coord} / 2$ $y_a = y_{L\,min} - (y_{PIAR} + t_{IAR})$ $z_a = z_{M\,min} - (z_{PIAR} + h_{IAR})$
$\Delta b_{coord} = 0$	$x_a = x_{coordObj}$ $y_a = y_{L\,min} - (y_{PIAR} + t_{IAR})$ $z_a = z_{M\,min} - (z_{PIAR} + h_{IAR})$
Δb_{coord} nicht definiert	$x_a = (x_{K\,max} - \Delta b_{coord} / 2) + (x_{PIAR} - b_{IAR} / 2)$ $y_a = y_{L\,min} - (y_{PIAR} + t_{IAR})$ $z_a = z_{M\,min} - (z_{PIAR} + h_{IAR})$

Die Koordinate x_a wird bei Existenz von mehr als einem Zweiarmobjekt (Fall 1) in die Mitte ihrer beiden Extrempositionen gelegt, da der gemeinsame Arbeitsraum spiegelsymetrisch ist, und somit für die Objekte eine gute Plazierungschance innerhalb des gemeinsamen Arbeitsraumes besteht. Erst wenn die Plazierung der Objekte mit diesem festem x_a scheitert, wird x um Dx und -Dx variiert und der Algorithmus noch einmal mit verändertem x durchlaufen. Für die anderen recht selten auftretenden Fälle 2.) und 3.) sind für x_a leicht modifizierte Ansätze zu machen, die hier aber nicht mehr genauer erklärt werden. Erklärende Skizzen findet man im Anhang B.5.



Anfangsplazierung für den Fall mehrerer ZAOs

Mit einer geschickten Festlegung von x_a erhofft man sich, in den meisten Fällen nur y und z variieren zu müssen, um das Kriterium positiv zu erfüllen. Es ist hier nicht entscheidend, alle (!) möglichen Plazierungen zu finden, sondern man kann nach der ersten möglichen Plazierung die Plausibilitätsüberprüfung abbrechen. Muß man aber doch noch x variieren, so kann man durch geschicktes Festlegen von Dx in beide Richtungen die Suchzeit auch hier klein halten. Ein qualitatives Beispiel sei in folgender Abbildung skizziert.

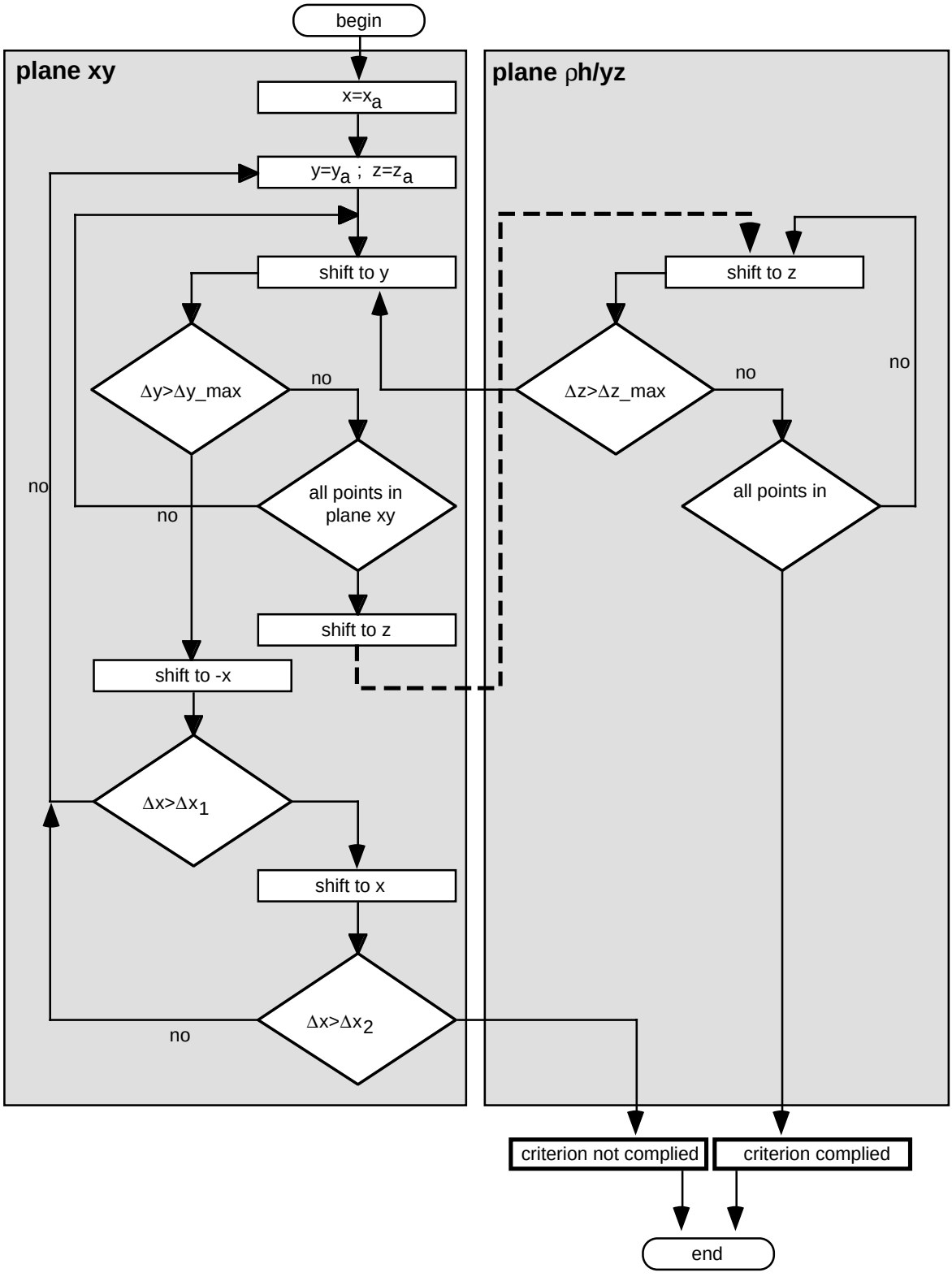


Bestimmung von Dx

Der Algorithmus stellt sich im Ablaufdiagramm mit den folgenden Größen dar:

x_a, y_a, z_a	Anfangsplazierung des Systems. Diese variieren je nach Fallunterscheidung
Dy_{max}, Dz_{max}	Maximales Variationsintervall in den Dimensionen y und z. Für Dimension x macht man eine gesonderte Betrachtung
Dx_1, Dx_2	Vorher bestimmtes maximales Variationsintervall in x-Richtung (Dx_2)und (-x)-Richtung(Dx_1)

Man erkennt sehr gut die serielle Zerlegung der dreidimensionalen Betrachtung in zwei zweidimensionale. Zur Überprüfung, ob der Punkt auch im Projektionsgebiet rh liegt, kommt man nur bei vorherigem erfolgreichen Test in der xy-Ebene. Dies kann unter Umständen eine enorme Geschwindigkeitssteigerung der Überprüfung bedeuten, denn wenn man keine gültige Plazierung in xy findet, kann man sofort mit der Aussage der Nichtplazierbarkeit abbrechen und spart sich die Variationen in der dritten Dimension.



Flußdiagramm des zweistufigen Plausibilitätstests

Die Frage, ob ein Punkt im xy-Projektionsgebiet liegt, läßt sich analytisch sehr leicht formulieren.

Zur Notation der Hilfskoordinatensysteme und der bezogenen x-, y- und z-Komponenten sei noch folgendes bemerkt:

Koordinaten des Punktes P relativ zu {S}:

$$\begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix}_{\{\Sigma\}} = \begin{pmatrix} x_{p,\{\Sigma\}} \\ y_{p,\{\Sigma\}} \\ z_{p,\{\Sigma\}} \end{pmatrix} = \vec{p} - \vec{\Sigma}$$

$\vec{p}$…Positionsvektor

$\vec{\Sigma}$…Vektor auf Koordinatensystem Σ

Definition der Hilfskoordinatensysteme erfolgt mittels Translation relativ zum Ursprungskoordinatensystems S (keine Änderung der Orientierung).

$$\vec{\Omega} = \vec{\Sigma} + \vec{t}$$

$\vec{t}$…Translationsvektor

$\vec{\Omega}$…Vektor auf Hilfskoordinatensystem Ω

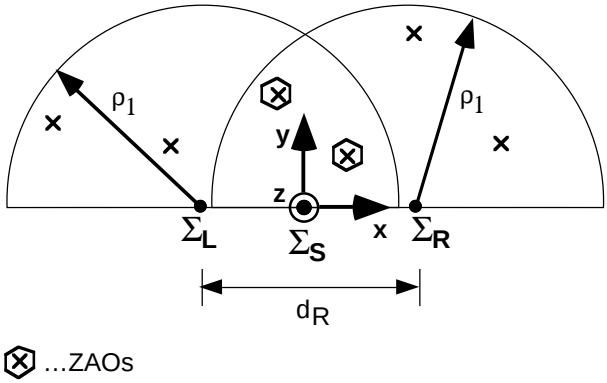
Ausgehend von der Definition des Arbeitsgebietes als zwei sich schneidende Halbkreise [Abb.] ist mathematisch sehr einfach nachzuweisen, ob sich ein Punkt (Projektion des Objektes in die entsprechende Ansichtsebene) in dem Gebiet befindet. Man stellt hier mit Hilfe der Kreisgleichung folgende Ungleichungen auf, die der Punkt O erfüllen muß:

- Objekt für Einarmmanipulation *(EAO)* muß in Halbkreis 1 *oder* Halbkreis 2 liegen.

$$\left(x_{O,\{\Sigma_L\}}^2 + y_{O,\{\Sigma_L\}}^2 \le \rho_1^2\right) \quad \text{V} \quad \left(x_{O,\{\Sigma_R\}}^2 + y_{O,\{\Sigma_R\}}^2 \le \rho_1^2\right)$$

- Objekt für koordinierte Manipulation *(ZAO)* muß in Halbkreis 1 *und* Halbkreis 2 liegen.

$$\left(x_{O,\{\Sigma_L\}}^2 + y_{O,\{\Sigma_L\}}^2 \le \rho_1^2\right) \quad \& \quad \left(x_{O,\{\Sigma_R\}}^2 + y_{O,\{\Sigma_R\}}^2 \le \rho_1^2\right)$$



Nachweis ob Punkt in xy-Ebene

Dasselbe Prinzip wie in der xy-Ebene wird auch in der rh-Ebene verwendet. Leider ist die Geometrie hier nicht ganz so einfach, und man muß zuerst feststellen, ob ein Punkt oberhalb oder unterhalb der Ebene liegt, die durch die x- und y-Achsen des Hilfsframes S_H aufgespannt wird. Ist diese Entscheidung getroffen, so hat man als Flächenbegrenzungen jeweils zwei Kreissegmente und eine Gerade. Macht man noch die Transformation in Zylinderkoordinaten mit

$$r_{\{\Sigma\}} = \sqrt{x_{O,\{\Sigma\}}{}^2 + y_{O,\{\Sigma\}}{}^2} \quad \text{und} \quad h = z_{O,\{\Sigma\}}$$

so lassen sich wieder Ungleichungen formulieren:

- Punkt liegt unterhalb der Fläche F(x,y,0)

$$\left(r_{\{\Sigma_{L/R}\}} < \varepsilon\right) \;\&\; \left(h_{\{\Sigma_{L/R}\}}{}^2 + r_{\{\Sigma_{L/R}\}}{}^2 \geq \rho_3{}^2\right) \;\&\; \left(h_{\{\Omega_2\}}{}^2 + r_{\{\Omega_2\}}{}^2 \leq \rho_2{}^2\right)$$

- Punkt liegt oberhalb der Fläche F(x,y,0)

$$\left(r_{\{\Sigma_{L/R}\}} < \rho_3\right) \;\&\; \left(h_{\{\Sigma_{L/R}\}}{}^2 + r_{\{\Sigma_{L/R}\}}{}^2 \leq \rho_1{}^2\right) \;\&\; \left(h_{\{\Omega_1\}}{}^2 + r_{\{\Omega_1\}}{}^2 \geq \rho_2{}^2\right)$$

Diese Überprüfung muß man natürlich wieder für Einarm- und Zweiarmobjekte getrennt mit den entsprechenden boolschen Verknüpfungen durchführen.
Um zu bestimmen, ob die Objekte sich auch in dem zur Nutzung vorgesehenen Raum befinden, geht man denselben Weg wie oben und zerlegt in xy- und yz-Projektion. Die Gleichungen sind hier viel einfacher, da man nur Geraden als Begrenzungslinien hat, keine Tansformation in Zylinderkoordinaten und auch keine Unterscheidung zwischen Ein- und Zweiarmobjekten machen muß. Die Ungleichungen lauten hier dann:

1. $(x_{O,\{P_{IAR}\}} \geq 0)$ & $(x_{O,\{P_{IAR}\}} \leq b_{IAR})$
2. $(y_{O,\{P_{IAR}\}} \geq 0)$ & $(y_{O,\{P_{IAR}\}} \leq t_{IAR})$
3. $(z_{O,\{P_{IAR}\}} \geq 0)$ & $(z_{O,\{P_{IAR}\}} \leq h_{IAR})$

5.5.4. Anfangsplazierung

Die Wahl der Ausgangssituation für einen Algorithmus ist von entscheidender Bedeutung, denn von ihr hängt wesentlich die Geschwindigkeit der Lösungsfindung ab. Es sollte also eine Anfangsplazierung gefunden werden, die gewährleistet, daß die Objekte hinreichend gut im zur Verfügung stehenden Arbeitsraum plaziert werden, so daß schon eine Suche in einem kleinen Raumintervall um diesen Anfang eine möglichst große Zahl von gültigen Plazierungen bringt.

5.5.4.1. Schwerpunkt der Objekte

Betrachtet man Arbeitsraum und den die Objekte umschreibenden Raum als Volumenkörper, so sollte man die beiden Körper so plazieren, daß die Schnittmenge zwischen beiden möglichst groß ist. Außerdem sollten die Objekte möglichst im Volumeninneren (große Orientierungsvielfalt) und nicht nahe des Randes des Arbeitsraumes liegen, d.h. daß die zentralen Bereiche beider Arbeitsräume zusammenfallen. Da der Schwerpunkt das Zentrum eines Volumenkörpers repräsentiert, scheint es eine gute Approximation obiger Forderung zu sein, Schwerpunkt der Objekte und Schwerpunkt des Arbeitsraumes zur Deckung zu bringen. Betrachtet man die Objekte als Massepunkte mit dem Gewicht g, so liegen Objekte je nach Gewichtung auch in qualitativ besseren oder schlechteren Gebieten. Liegen die Objekte mit den Massen g_{Oi} an den Stellen $o_i = (x_i, y_i, z_i)$, so berechnet sich der Schwerpunkt S_O zu

$$\vec{S}_O = \frac{\sum g_i \vec{o}_i}{\sum g_i}$$

$\vec{o}_i$…Positionsvektor auf Massepunkt i mit Masse g_i

Zur Berechnung des Schwerpunktes des Arbeitsraumes und zur anschließenden Plazierung mit

$$\vec{S}_{AR} = \vec{S}_O$$

muß man zuerst definieren, welchen Teil des Arbeitsraumes und welche Objekte man hierfür berücksichtigt. Man unterscheidet zwei Fälle:

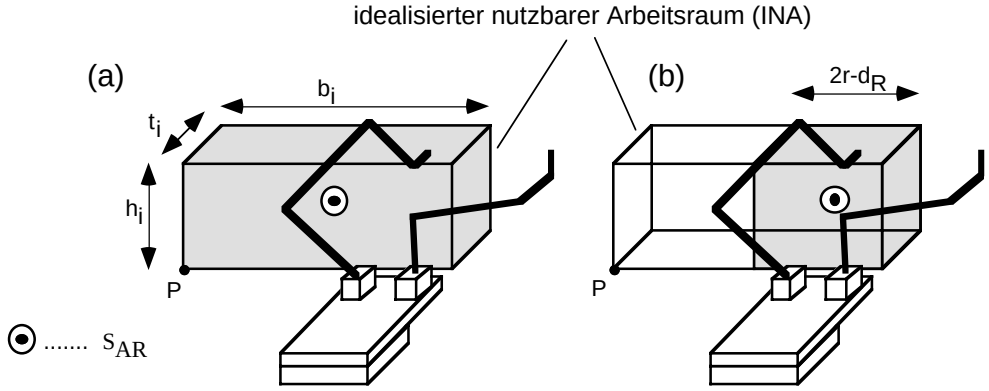
- Existiert mindestens ein Zweiarmobjekt, so sind für die Plazierung nur diese Objekte und der idealisierte gemeinsam benutzbare Arbeitsraum zu berücksichtigen. Der Schwerpunkt S_{AR} ergibt sich zu

$$\vec{S}_{AR} = \begin{pmatrix} x_{S_{AR}} \\ y_{S_{AR}} \\ z_{S_{AR}} \end{pmatrix}_{\{P_{IAR}\}} = \frac{1}{2} \cdot \begin{pmatrix} 2b_{IAR} - (2r - d_R) \\ t_{IAR} \\ h_{IAR} \end{pmatrix}$$

- Existiert kein Zweiarmobjekt, so legt man den ganzen idealisierten nutzbaren Arbeitsraum und alle Objekte zugrunde. S_{AR} ergibt sich dann zu

$$\vec{S}_{AR} = \begin{pmatrix} x_{S_{AR}} \\ y_{S_{AR}} \\ z_{S_{AR}} \end{pmatrix}_{\{P_{IAR}\}} = \frac{1}{2} \cdot \begin{pmatrix} b_{IAR} \\ t_{IAR} \\ h_{IAR} \end{pmatrix}.$$

Der Grund für diese Unterscheidung liegt zum einen in der wesentlich kleineren Größe des gemeinsamen Arbeitsraumes (wenig Spielraum) und zum anderen darin, daß koordiniert manipulierte Objekte in der Regel wesentlich höhere Anforderungen an die Orientierungsvielfalt stellen und sich deshalb in Kernbereichen des Arbeitsraumes befinden müssen.



Position des Schwerpunktes für die Fälle keiner ZAOs und mehrerer ZAOs

5.5.5. Durchlaufen des Suchintervalls

Nachdem man jetzt eine geeignete Anfangsplazierung gefunden hat, muß man nun die Systemposition im definierten Suchintervall variieren. Alle möglichen Plazierungen werden dann gespeichert um sie im nächsten Schritt einer Bewertung zu unterziehen, die dann eine geeignete Auswahl einer Endplazierung aus der Menge der möglichen zu erhalten.

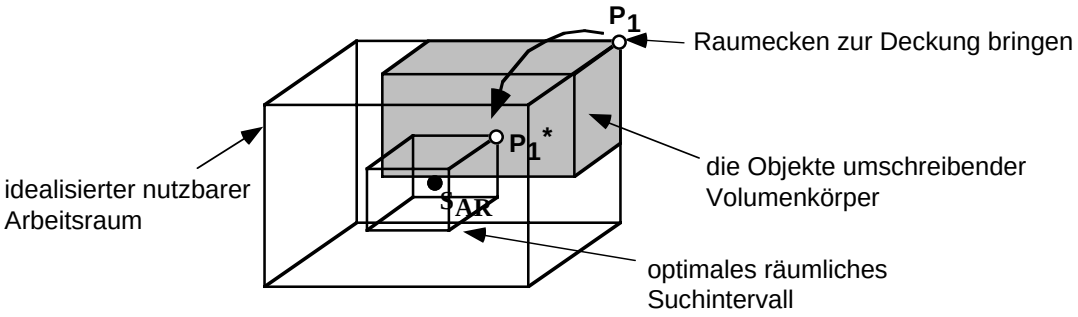
5.5.5.1. Sukzessives Durchlaufen des Gesamtsuchraumes

Die einfachste Möglichkeit ist das sukzessive Durchlaufen des gesamten angegebenen Raumintervalls. Das Suchintervall wird dazu geeignet in diskrete äquidistante Punkte zerlegt, die alle auf eine mögliche Plazierung überprüft werden. Die Anzahl n der jeweils zu überprüfenden Punkte beträgt also

$$n_S = v_x \cdot v_y \cdot v_z$$

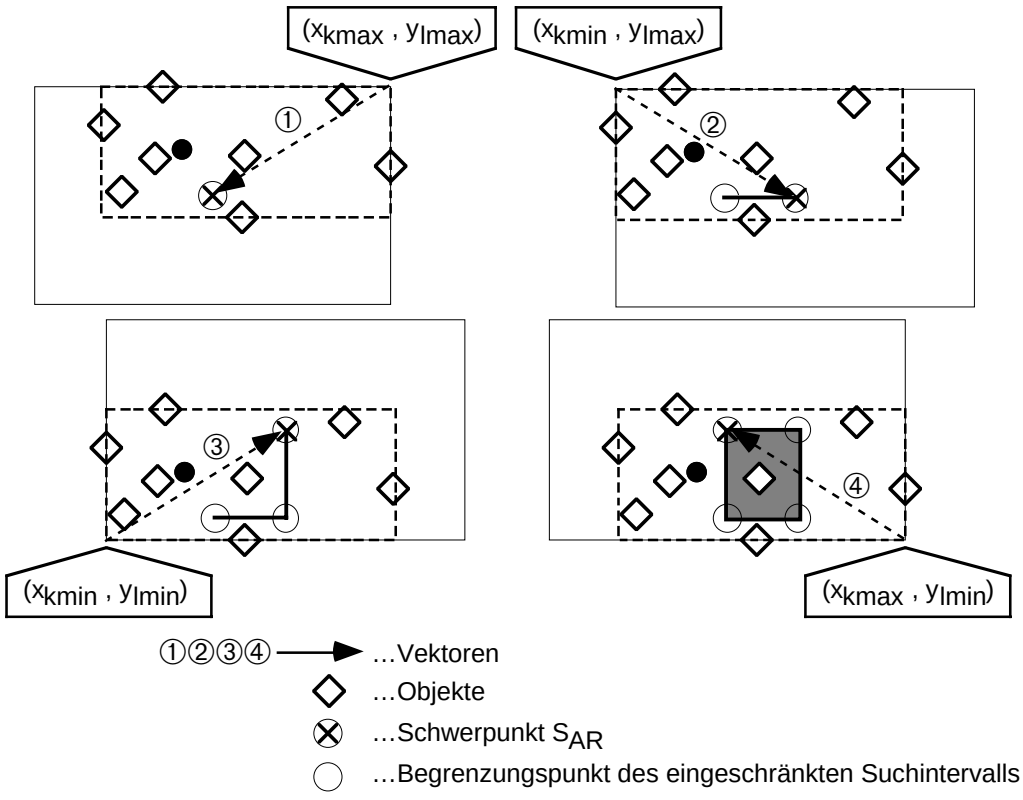
5.5.5.2. Automatische Bestimmung des optimalen Suchintervalls

Das Problem beim sukzessiven Durchlaufen ist die stark anwachsende Suchzeit bei Erhöhung der Auflösung und bei großen Suchintervallen. Vor allem eine höhere Auflösung ist für ein starkes Anwachsen der Suchzeit verantwortlich.
Nehmen wir z.B. ein festes Suchintervall das durch einen kubischen Volumenkörper mit 0,5m Kantenlänge repräsentiert wird. Dann muß man bei einer Auflösung von d = 10cm 125, bei d = 5cm 1000 und bei d=2,5cm 8000 Plazierungen überprüfen. Bei Halbierung der Auflösung wächst die Suchzeit also auf das Achtfache an.
Man versucht deshalb, durch einfache geometrische Betrachtungen das Suchintervall so zu verkleinern, daß man Plazierungsversuche, die nicht zu einer Lösung führen können, von Anfang an ausschließt. Hier z.B. nutzt man die Tatsache, daß der die Objekte umschreibende kubische Volumenkörper immer im idealisiert nutzbaren Arbeitsraum liegen muß. Das resultierende eingeschränkte Suchintervall besitzt dann als Eckpunkte jeweils die Positionen des Schwerpunkts des idealisierten Arbeitsraumes, wenn man seine Raumecken mit den Raumecken des die Objekte umschreibenden Kubus zur Deckung bringt. Folgende Abbildung zeigt das Vorgehen für eine Raumecke (rechts oben)



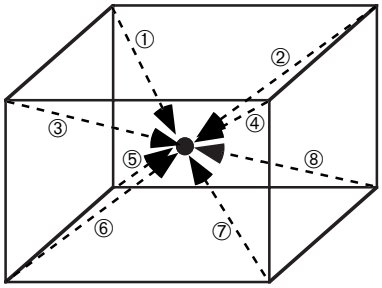
Punkt P_1 bestimmt P_1^* des optimalen Suchintervalls

Da die Koordinaten der Raumecken des Objektkörpers durch die entsprechenden x-, y- und z-Koordinaten von K_{min}, K_{max}, L_{min}, ... gegeben sind, ist die Bestimmung des eingeschränkten Suchintervalls einfach. Zu diesen Eckpunkten addiert man jeweils den entsprechenden Vektor, der von der Ecke auf den Schwerpunkt S_{AR} zeigt. Der Übersicht halber ist das Vorgehen für den zweidimensionalen Fall skizziert. Es kann aber sehr leicht um eine Dimension erweitert werden.



Zweidimensionales Beispiel zur Bestimmung des optimalen Suchintervalls

Man unterscheidet abhängig von der Eckenanzahl vier verschiedene Vektoren und muß dann entsprechend beim dreidimensionalen Fall acht Vektoren (wegen der acht Ecken) unterscheiden.

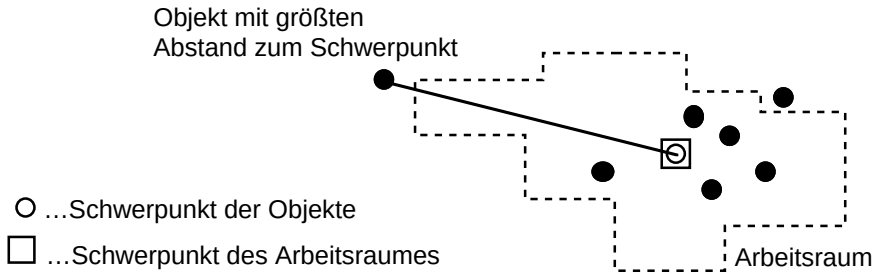


Die Anzahl der diskreten Positionen in diesem optimal beschränkten Intervall, die als mögliche Plazierungen in Frage kommen, bezeichnen wir mit n_{aut}.
Ausgehend von der Anzahl der zu überprüfenden Plazierungen (n_{aut} oder n_S), kann man entscheiden, welches Intervall man in Betracht zieht. Die mathematisch formulierte Bedingung lautet:

$$n = \min(n_{aut}, n_S)$$

5.5.5.3. Überprüfung auf Erreichbarkeit der Objekte

Für jede der n Plazierungen im Suchintervall muß man die Erreichbarkeit der Objekte unter ihrer Greiforientierung verifizieren. Jedoch ist es in der Mehrzahl der Fälle nicht nötig, die Erreichbarkeit jedes Objektes nachzuweisen. Eine mögliche Plazierung scheitert schon bei einem nicht zugänglichen Objekt, und es kann deshalb auf die Überprüfung der anderen verzichtet werden. Diesen Umstand kann man zur Verringerung des Zeitbedarfs des Algorithmus ausnutzen. Es ist zum Beispiel sinnvoll, die weit vom Schwerpunkt des Arbeitsraumes entfernt liegenden Objekte zuerst zu prüfen, da es wahrscheinlicher ist, daß sie außerhalb des erreichbaren Arbeitsraumes liegen, als diejenigen in der Nähe des Schwerpunktes. Je nach Form und Größe der Arbeitsräume kann man die Überprüfung schon vor der Betrachtung aller Objekte abbrechen. Das folgende Beispiel zeigt die Anfangsplazierung (Schwerpunkte stimmen überein). Wenn man in diesem Beispiel das am weitesten entfernteste Objekt zuerst überprüfen würde, würde die Plazierung schon nach einem betrachteten Objekt scheitern.



Überprüfung des am weitesten entfernten Objektes

5.5.6. Bewertung der Plazierung

In diesem Abschnitt beschäftigt man sich mit der Frage, wie aus der Menge der möglichen Plazierungen die für die spezielle Aufgabenstellung optimale ausgewählt werden kann. Der Begriff „optimal" ist natürlich nicht global und unabhängig von Umgebung und Aufgabenstellung angebbar, sondern seine Definition orientiert sich stark an den vorher durch die Programmierung festgelegten Bewertungskriterien. Dies ist die Stelle, an der Erfahrungen aus der manuellen Ausführung in Form von geeigneten Kriterien und Regeln festgelegt werden. Man sollte deshalb darauf achten, die Struktur des Programms so anzulegen, daß es leicht modifizierbar und erweiterbar ist.

Als Resultat einer solchen Bewertung erhält man dann die möglichen Plazierungen, die in einem gewissen Intervall um das Maximum der Bewertungfunktion liegen. Von der Größe dieses Intervalls hängt natürlich die Anzahl der Lösungen ab, aus der dann entweder zufällig oder durch den Benutzer die Endplazierung festgelegt wird. Im folgenden unterscheidet man zwischen mastabhängigen und mastunabhängigen Kriterien, die sich nur dadurch unterscheiden, daß sich die Parameter bei den mastabhängigen Kriterien je nach Masttyp ändern. Man muß dies dann bei der Implementierung entsprechend berücksichtigen.

Alle Kriterien legen als einzigste Variablen die Position im Raum d.h. nur die x-, y- und z-Koordinaten zugrunde. Die Güte einer Positionierung wird durch Zahlenwerte angegeben, wobei höhere Werte eine höhere Güte repräsentieren. Anfangs weist man allen möglichen Positionierungen den Wert 1 zu, der dann kriterienabhängig durch Multiplikation oder Addition verändert wird. Man erhält so allgemein eine Gütefunktion $G_i(q_j)$ für jede der n Positionierungen (i=0…n), die von den Parametern q_j und ihren dazugehörigen Gewichtungsfaktoren υ_i abhängig sind.

$$G_n(q_j) = \prod (q_j^{\nu_j})$$

Anfangs gilt für alle Kriterien:

$$q_j = 1 \quad \text{für} \quad j = 1 \ldots \text{Anzahl der Kriterien}$$

Die Parameter q_j in der Gütefunktion werden oft als Summe definiert und zwar dann, wenn die Definition eines Kriteriums mehrere Referenzpunkte berücksichtigt. Dann erhält man für das j-te Kriterium mit k Referenzpunkten

$$q_j = \sum_k q_{jk}$$

Im folgenden werden beispielhaft drei Kriterien formuliert, die sich in die zwei Klassen mastunabhängig und mastabhängig einordnen lassen. Ihre Definition soll Idee und Vorgehensweise bei der Formulierung der Kriterien vermitteln.

5.5.6.1. Mastunabhängige Kriterien

Aufgrund des Sicherheitsabstandes und der Kollisionsgefahr sollen die Objekte in möglichst großer Distanz zur Roboterbasis manipuliert werden. Stellt z.B. jedes Objekt einen Referenzpunkt dar, so kann man den Parameter q_{1k} des ersten Kriteriums über den absoluten Abstand der Roboterbasis zum k-ten Objekt definieren. Summiert man über alle k Objekte erhält man q_1 zu

$$q_j = \sum_k q_{jk} = \sum_k \sqrt{(x_n - x_k)^2 + (y_n - y_k)^2 + (z_n - z_k)^2}$$

mit

$(x_n, y_n, z_n) \ldots$ Position der n - ten Systempositionierung

$(x_k, y_k, z_k) \ldots$ Position des k - ten Objektes

Einer etwas komplexeren Bewertung liegt die Definition von sogenannten Vorzugsgebieten zugrunde. Einige Gründe für diese zu bevorzugenden Raumzonen sind:

- ergonomische Anforderungen, da der Operator im Teleoperationsbetrieb mit ausgestreckten Armen weniger bequem und exakt manipulieren kann.
- Manche Raumzonen sind aufgrund einer guten Kamerapositionierung in mehreren Ansichten und in einer guten Auflösung darstellbar, während andere Raumbereiche schlechter eingesehen werden können. Beim Betrieb mit direktem Sichtkontakt durch den Bedieners gibt es Bereiche, die abhängig von seinem

Sichtfeld und von dem mechanischen Aufbau des Systems gut oder schlecht eingesehen werden können.

Man kann diese Vorzugsgebiete näherungsweise durch einfache geometrische Körper (Kubus, Kugel, Kegel, …) oder komplexer mit Hilfe von Raumkurven definieren, wobei jede Raumzone eine Güte in Form einer Zahl besitzt. Wird wiederum diese Güte über alle Objekte summiert, erhält man

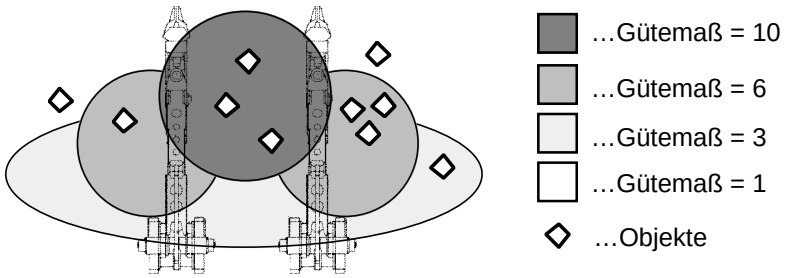
$$q_2 = \sum_k g_k \cdot q_{2k}$$

mit

q_{2k}…Güte der Raumzone, in der sich Objekt k befindet

g_k …Gewicht des k – ten Objekts

Folgendes kleines Beispiel zeigt die Draufsicht auf Vorzugsgebiete in Form von sich schneidenden Kugeln und Ellipsoiden. Außerhalb dieser Gebiete ist das Gütemaß eins.



Mögliche Vorzugsgebiete des Robotersystems

Der Zahlenwert für q_2 (alle Objekte haben das Gewicht 1)errechnet sich wie folgt:

$$q_2 = \sum_{k=1}^{10} q_{2k}$$
$$= 3 \cdot 10 + 4 \cdot 6 + 1 \cdot 3 + 2 \cdot 1 = 59$$

5.5.6.2. Mastabhängige Kriterien

Das wichtigste mastabhängige Kriterium ist eine Plazierung außerhalb der verbotenen Gebiete. Deren geometrische Daten werden abhängig vom Masttyp aus einer Datenbank oder Datei geladen und stehen dann für die Überprüfung zur Verfügung. Die mathematische Formulierung des Kriteriums gestaltet sich einfach, weil es sich um ein für die Plazierung notwendiges Kriterium handelt. Da für dieses Kriterium alle Plazierungen außerhalb der verbotenen Gebiete gleichwertig sind, kann man folgende Definition für das Kriterium q_3 angeben.

- Plazierung innerhalb verbotener Zonen

$$q_3 = 0$$

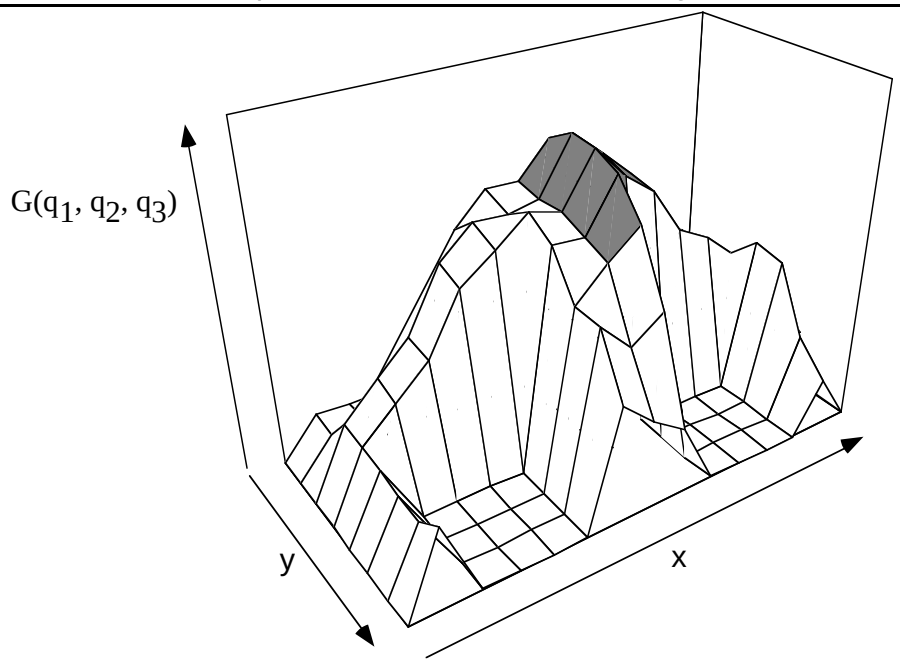
- Plazierung außerhalb verbotener Zonen

$$q_3 = 1$$

Somit hat dieses Bewertungkriterium die absolute Priorität, denn bei einer Güte von Null scheidet eine Plazierung aus. Man sieht hier, daß es notwendige und hinreichende Kriterien gibt. Daß die Aufteilung mit der Zuordnung zu mastabhängig/mastunabhängig übereinstimmt ist, bei der geringen Anzahl von betrachteten Kriterien rein zufällig.

5.5.7. Plazierung

Nach der Berechnung der Güte jeder einzelnen Plazierung ist die Auswahl denkbar einfach. Man bestimmt die maximale Güte und erhält so die *beste* Positionierung. Aufgrund der approximativen Formulierung der Kriterien kann es aber sinnvoll sein, ein gewisses Güteintervall zu definieren, in dem alle Plazierungen gleichberechtigt favorisiert sind. Dann kann dem Benutzer eine gewisse Anzahl an Systemplazierungen vorschlagen werden, aus der er dann manuell die ihm geeignet Erscheinende auswählt. Nachfolgende Graphik zeigt die Gütefunktion des Beispiels aus Anhang B.6. Das Güteintervall, in dem alle Plazierungen als gleichwertig gelten, ist grau unterlegt.



Güte der möglichen Plazierungen (x,y)

Die einzigen Approximationen, die bezüglich des *objektlocation* gemacht wurden, waren die Zuordnung der Objektorientierung in ein Orientierungsintervall, repräsentiert durch einen Vektor $\vec{a}$, und die Abbildung der Objektposition auf ein Raster. Die jetzt noch verbleibende Menge an Positionierungen kann man unter vertretbarem Zeitaufwand mittels exakter Angabe von Position und Orientierung durch Einbeziehung der kinematischen Transformationsgleichungen auf Realisierbarkeit überprüfen.

Die Auswahl der Endplazierung aus dieser Menge an realisierbaren Aufstellungen erfolgt schließlich durch den Benutzer selbst. Um ihm diese Aufgabe zu erleichtern, wird ihm entweder ein Simulationsprogramm (TOROS) oder eine einfache 3D-Visualisierung zur Verfügung gestellt.

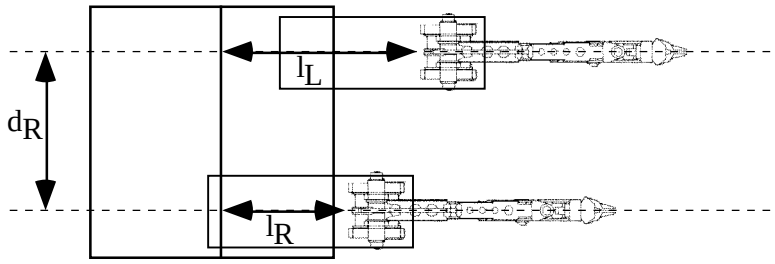
Mit Hilfe der Simulation kann er dynamisch die Manipulationen durchführen und somit nicht nur die Endplazierung bestimmen, sondern auch gleich die Manipulationen mit Kollisionsüberprüfung und Trajektorienbestimmung planen und eventuell sogar für den automatischen Betrieb abspeichern. Diese Möglichkeit ist aber aufgrund des Zeitbedarfs der vorherigen Planung dem Labor vorbehalten.

Im Gegensatz dazu bietet die statische 3D-Visualisierung ein einfaches und auch auf kleineren Rechnern zu implementierendes Werkzeug, um die Plazierung des

Systems vor dem Mast zu veranschaulichen. Die Entscheidung trifft der Bediener dann aufgrund seiner bisherigen Erfahrung mit dem System.

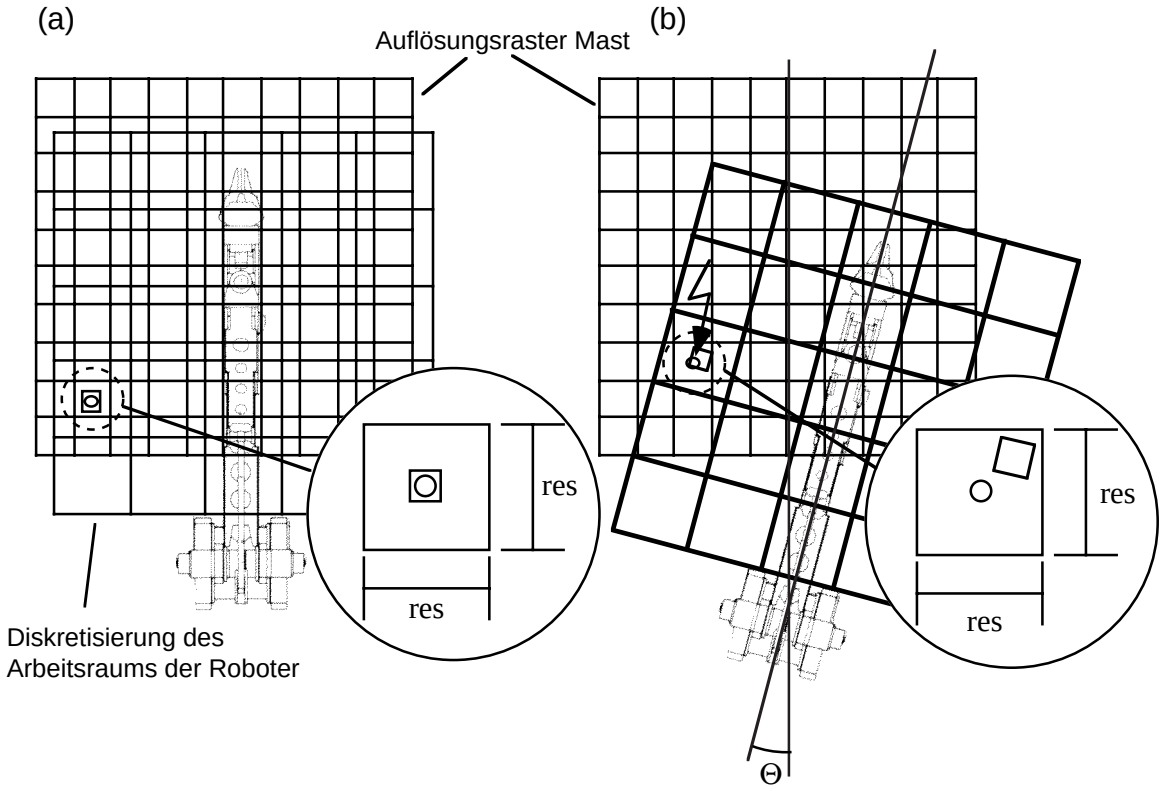
5.6. Erweiterungsmöglichkeiten

Da der Algorithmus auf einer sequentiellen Suche nach möglichen und guten Plazierungen basiert, ergibt sich natürlich ein erhebliches Potential zur Geschwindigkeitssteigerung im Ausschluß von sinnlosen Positionierversuchen, die zu keinem Ergebnis führen können. Ein Überlegung dazu wurde schon in der Beschränkung des Suchintervalls aufgezeigt, denkbar sind aber noch viele weitere.
Da man bei der Arbeitsraumbetrachtung die Roboter immer getrennt betrachtet, kann man zusätzlich zu dem variablen Parameter d_R auch noch weitere variable Aufstellungsparameter des Systems betrachten. Realisiert man z.B. das System so, daß die Roboter nicht mehr wie bisher angenommen auf der x-Achse montiert sind, sondern variabel in y-Richtung verschoben werden können (l_L, l_R), ist eine solche Aufstellungsänderung sehr leicht zu realisieren.



Beispiel für mögliche Plattformparameter

Man kann im allgemeinen davon ausgehen, daß jede Translation der Roboter (im Raster der diskreten Auflösung res), d.h. ohne Änderung ihrer Grundorientierung [Abb.(a)], in den Algorithmus implementiert werden kann. Drehungen allerdings sind nur unter größerem Aufwand berücksichtigbar, da die Auflösungraster sich nicht mehr durch ganzzahlige Multiplikation ineinander überführen lassen [Abb.(b)].



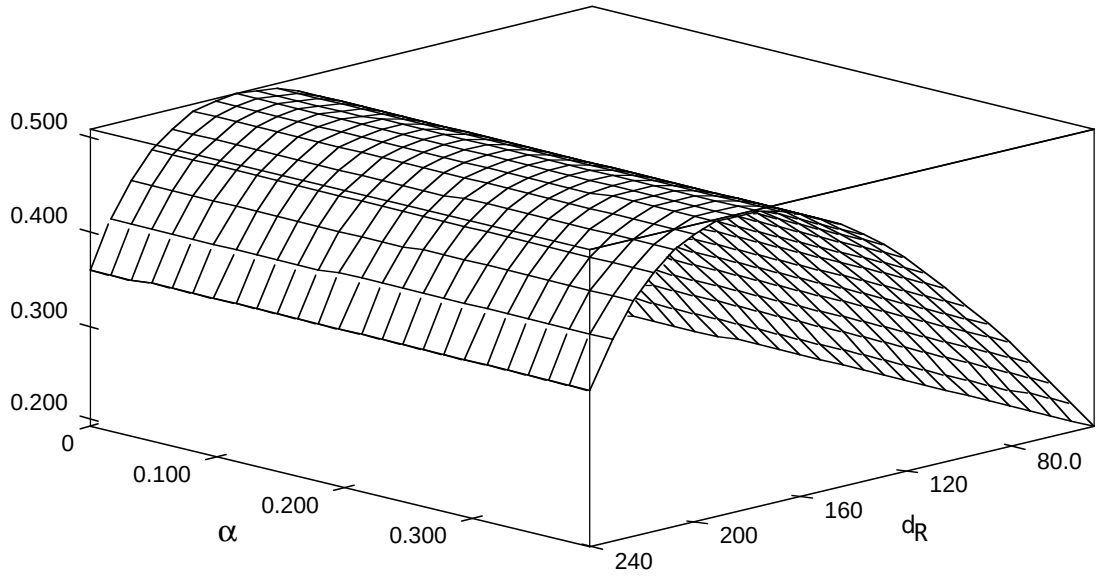
Änderung der Grundorientierung des Roboters um den Winkel Q

Im Moment besitzt die Software nur die Möglichkeit, eine Plazierung für die spezifizierten Objekte zu ermitteln, wenn diese möglich ist. Das bedeutet, daß der Benutzer bei Nichterreichen der Objekte unter einer sich nicht ändernden Plazierung deren Lage oder Zusammenstellung verändern muß. Die Software selber macht ihm aber in dieser Beziehung keinerlei Vorschläge. Durch die Schnelligkeit der Überprüfung auf eine mögliche Plazierung ist es denkbar, daß man einen Algorithmus implementiert, der eine Aufteilung der gesamten Manipulationen in Subaktionen so findet, daß man so wenig wie möglich Positionswechsel des Systems hat.

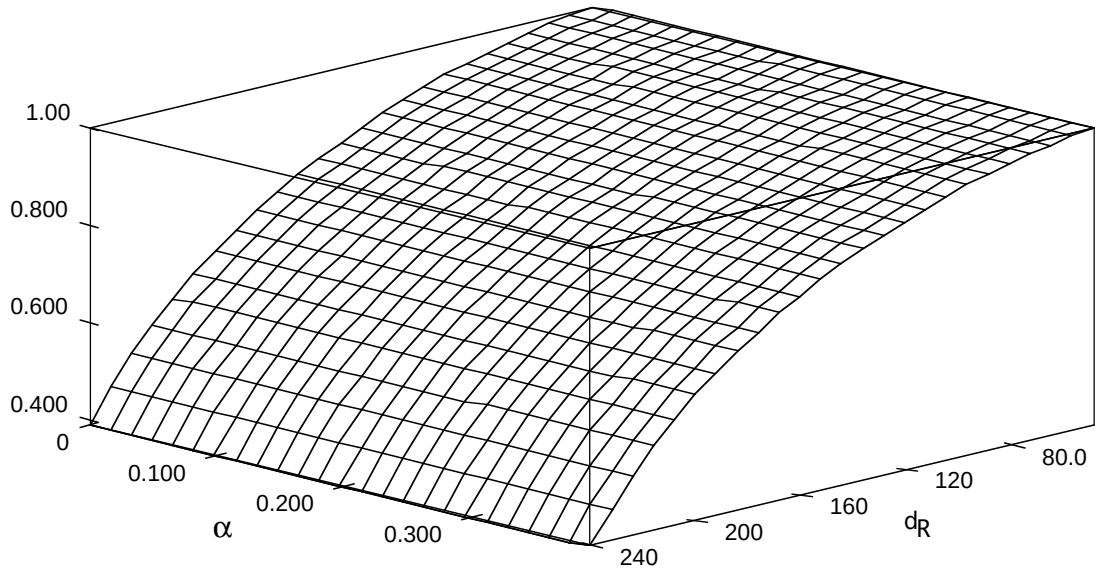
Anhang A

A.1. Zweidimensionale graphische Darstellung der Bewertungsgrößen in Abhängigkeit der variablen Größen d_R und a

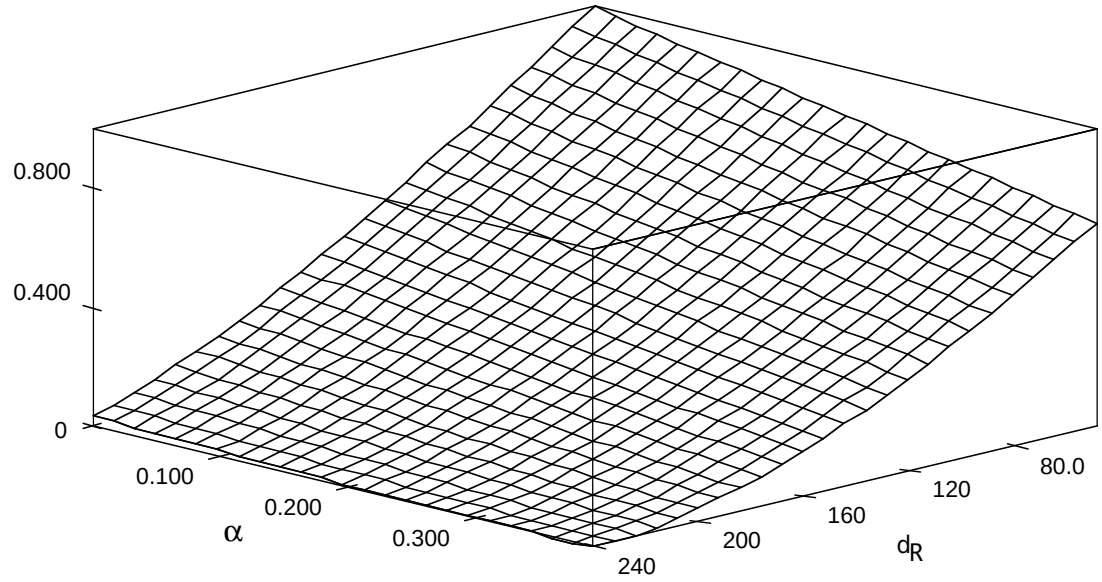
A.1.1. Maximale Greifweite $d_{bypeff}(d_R,a)$ im Abstand r_{coord}



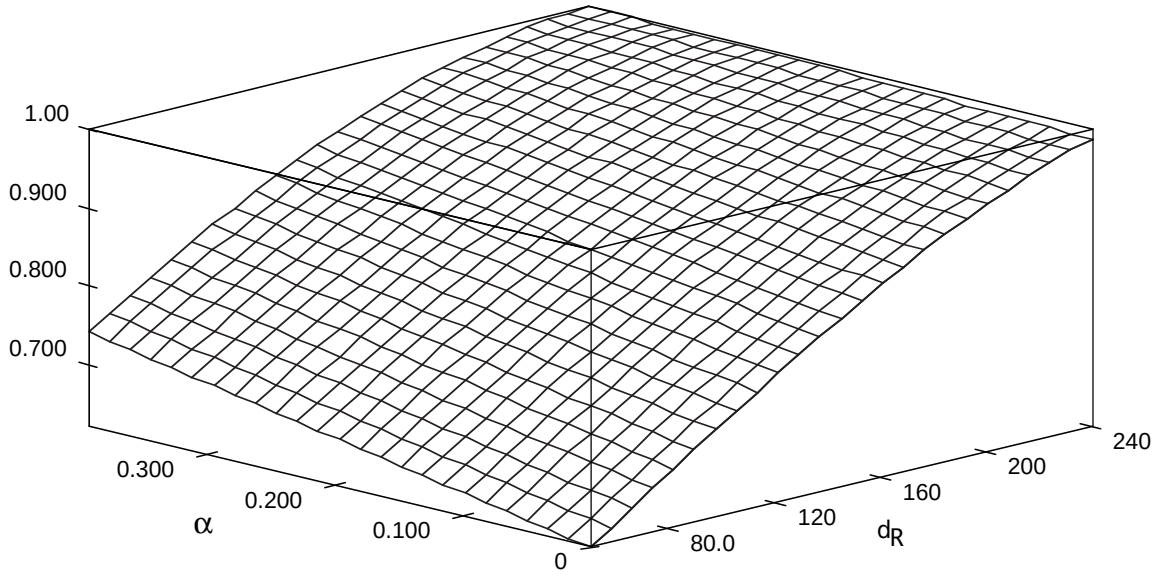
A.1.2. Maximaler Arbeitsradius für koordinierte Aktionen $r_{coordnorm}(d_R,a)$



A.1.3. xy-Projektion des gemeinsamen Arbeitsraumes $F_{coordnorm}(d_R,a)$



A.1.4. xy-Projektion des gesamten Arbeitsraumes $F_{totnorm}(d_R,a)$



A.2. Maplecode zur Bestimmung der Parameter d_R und a

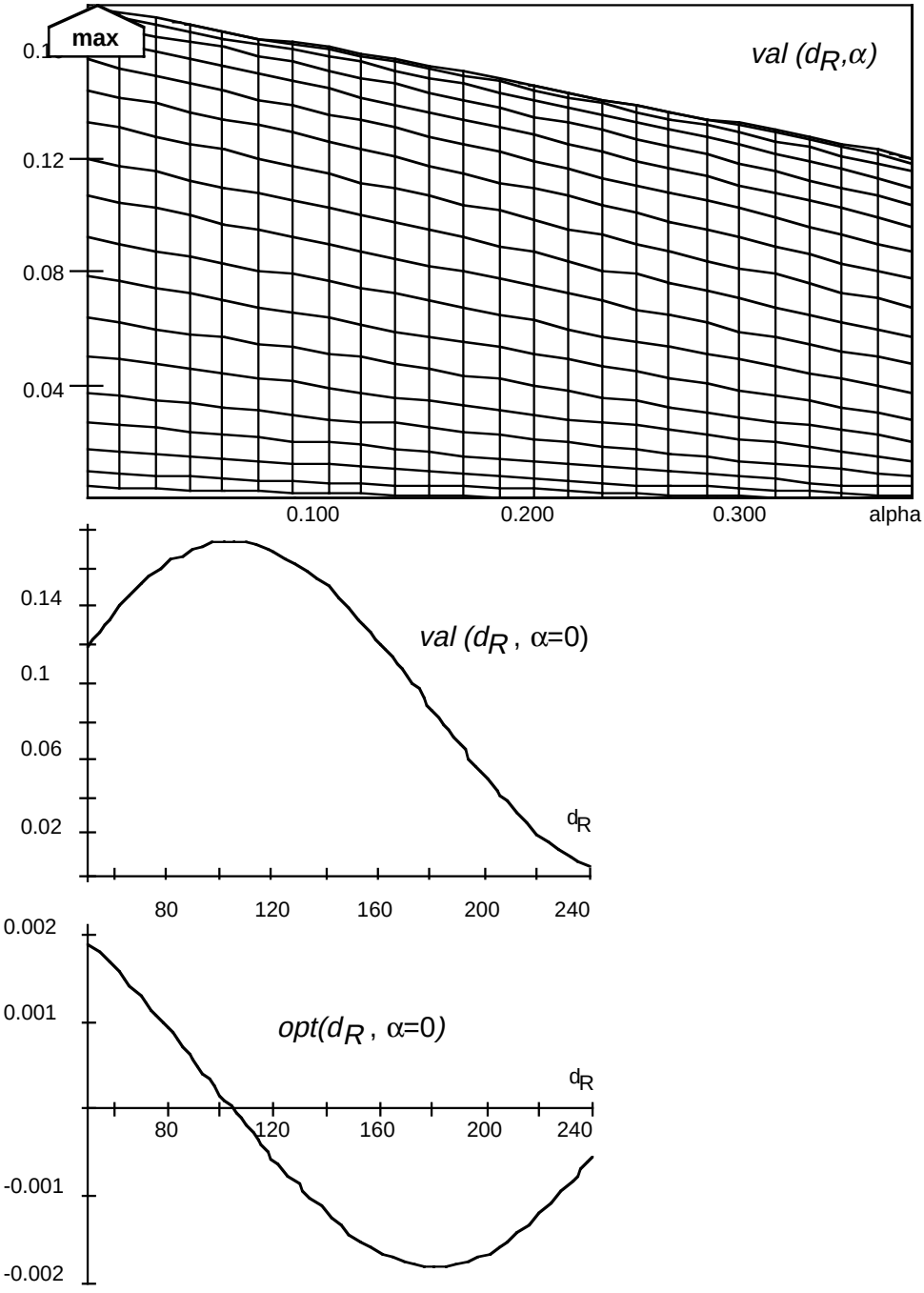
```
with(plots):
printlevel:=1:
r := 130: drmin := 50:
drmax := 260:
alpha_max := Pi/8:

a1:=1: ## Gewichtung d_byp
a2:=1: ## Gewichtung r_coord
a3:=1: ## Gewichtung F_tot
a4:=1: ## Gewichtung F_coord

        ## Gültigkeitsintervall der Formeln
        drmax_plot := evalf(2*r*cos(alpha_max)):

        ## rcoord normiert auf rcoord bei drmin
        r_coord := sqrt(r^2-(dr/2)^2):
        c_nrcoord := evalf(subs(dr=drmin,r_coord)):
        r_coordnorm := r_coord/c_nrcoord:

        ## dbyp normiert auf
        d_byp := 2*dr:
        c_ndbyp := 2*drmax:
        d_bypnorm := d_byp/c_ndbyp:
        d_bypeff := d_bypnorm*r_coordnorm:

        ## Gemeinsam überdeckte Fläche normiert auf F_12 bei
        ## alpha=0 und dr = drmin
        F_alpha0 := r^2*arccos(dr/(2*r)) - (dr/2)*r_coord:
        F_sub1 := r^2*(alpha - cos(alpha)*sin(alpha)):
        F_sub2 := (2*r*cos(alpha)-dr)*r*sin(alpha)-
    (tan(alpha)*(0.5*(2*r*cos(alpha)-dr))^2):
        F_coord := F_alpha0 - F_sub1 - F_sub2:
```

```
cnF_coord := evalf(subs(dr=drmin,F_alpha0)):
F_coordnorm := F_coord/cnF_coord:

## Gesamtfläche normiert auf Maximum (F_12=0)
F_kreis := evalf(Pi) * r^2:
F_tot := F_kreis - F_coord:
F_totnorm := F_tot/F_kreis:

## Zusammenfassen der Größen/Funktion der Qualität
## der Aufstellung
val := d_bypeff^a1 * r_coordnorm^a2 * F_totnorm^a3 *
F_coordnorm^a4:

## Dreidimensionaler Plot der Funktion
plot3d(val,dr=drmin..drmax_plot,alpha=0..alpha_max,axes=BOXED);

## Einsetzen des graphisch abzulesenden Wertes für alpha bei ##
Maximum (default = 0)
## mit zweidimensionalem Plot
alpha_fixed := 0:
val_alpha := evalf(subs(alpha=alpha_fixed,val)):
plot(val_alpha,dr=drmin..drmax_plot);

## Differentiation der Gütefunktion nach d_R
opt := diff(val_alpha,dr):
plot(opt,dr=drmin..drmax_plot);

## Auflösung der Gleichung opt(d_R) = 0
fsolve((opt=0),dr,50..200);
```

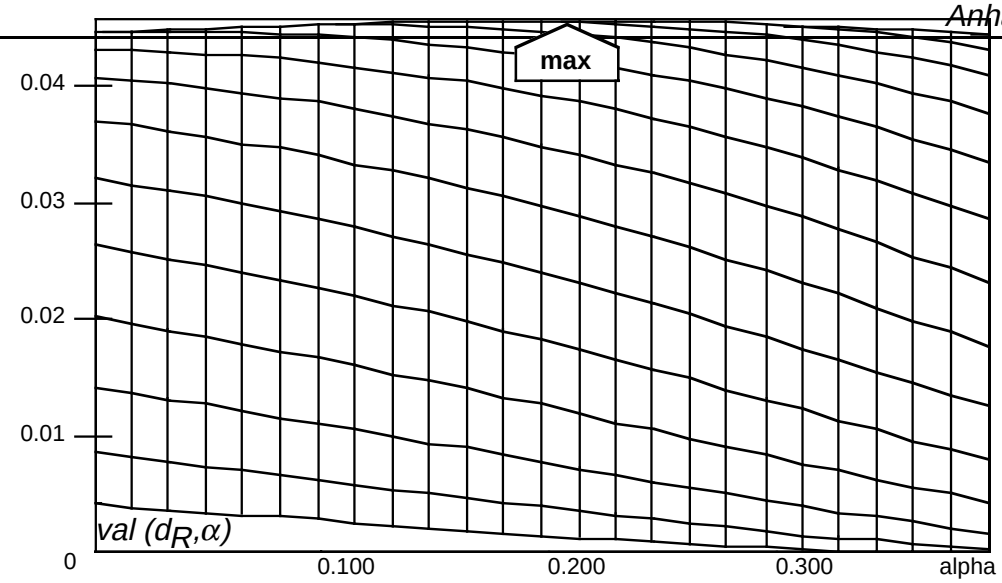
A.3. Graphische Illustration zur Bestimmung von d_R und a

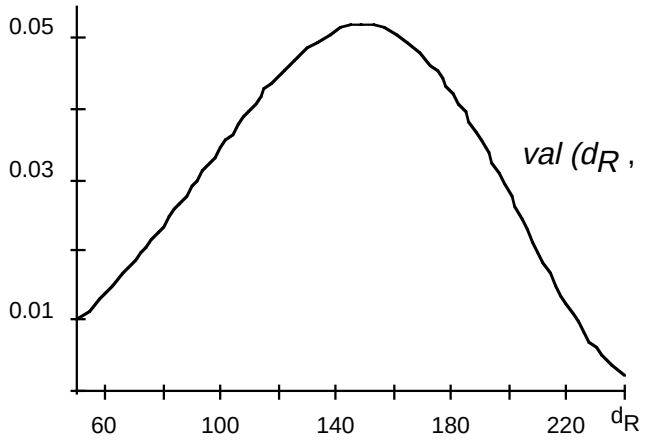
Pragmatisches Vorgehen durch Reduktion der zweidimensionalen auf eine eindimensionale Funktion durch Ersetzen von a durch festen Wertes beim Maximum.

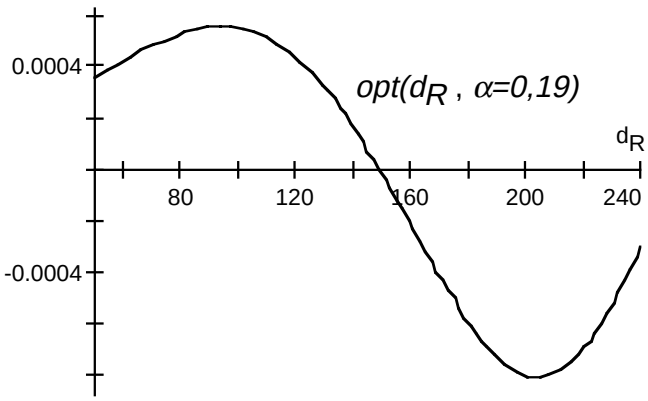
A.3.1. Gewichtungsfaktor k_3 = 1



A.3.2. Gewichtungsfaktor k_3 = 8


max
0.04
0.03
0.02
0.01
0
val (d_R,α)
0.100
0.200
0.300
alpha


0.05
0.03
0.01
val (d_R , α=0,19)
60
100
140
180
220
d_R


0.0004
-0.0004
opt(d_R , α=0,19)
d_R
80
120
160
200
240

Anhang B

B.1. Verfahren nach R.P.Paul zur Lösung des inversen kinematischen Problems

Denavit- Hartenberg- Matrizen und ihre Inversen für jede der sechs Achsen. S_n steht für sin Q_n und C_n für cos Q_n

$$A_1 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_1^{-1} = \begin{bmatrix} C_1 & S_1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -S_1 & C_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & C_2 a_2 \\ S_2 & C_2 & 0 & S_2 a_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_2^{-1} = \begin{bmatrix} C_2 & S_2 & 0 & -a_2 \\ -S_2 & C_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} C_3 & -S_3 & 0 & C_3 a_3 \\ S_3 & C_3 & 0 & S_3 a_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_3^{-1} = \begin{bmatrix} C_3 & S_3 & 0 & a_3 \\ -S_3 & C_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} C_4 & 0 & S_4 & C_4 a_4 \\ S_4 & 0 & -C_4 & S_4 a_4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_4^{-1} = \begin{bmatrix} C_4 & S_4 & 0 & -a_4 \\ 0 & 0 & 1 & 0 \\ S_4 & -C_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} C_5 & 0 & -S_5 & 0 \\ S_5 & 0 & C_5 & 0 \\ 0 & -1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_5^{-1} = \begin{bmatrix} C_5 & S_5 & 0 & 0 \\ 0 & 0 & -1 & d_5 \\ -S_5 & C_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_6 = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_6^{-1} = \begin{bmatrix} C_6 & S_6 & 0 & 0 \\ -S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Vorwärtstransformation durch Multiplikation der Matrizen A_1 bis A_6.

$$U_6 = A_6$$

$$U_5 = A_5U_6 = \begin{bmatrix} C_5C_6 & -C_5S_6 & -S_5 & 0 \\ S_5C_6 & -S_5S_6 & C_5 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$U_4 = A_4U_5 = \begin{bmatrix} C_4C_5C_6 - S_4S_6 & -C_4C_5S_6 - S_4C_6 & -C_4S_5 & S_4d_5 + C_4a_4 \\ S_4C_5C_6 + C_4S_6 & -S_4C_5S_6 + C_4C_6 & -S_4S_5 & -C_4d_5 + S_4a_4 \\ S_5C_6 & -S_5S_6 & C_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$U_3 = A_3U_4 = \begin{bmatrix} X_1C_3 + X_2S_3 & X_3C_3 + X_4S_3 & -C_3C_4S_5 + S_3S_4S_5 & C_3(a_4C_4 + S_4d_5 + a_3) + S_3(C_4d_5 - a_4S_4) \\ -X_2C_3 + X_1S_3 & -X_4C_3 + X_3S_3 & -S_3C_4S_5 - C_3S_4S_5 & C_3(-C_4d_5 + a_4S_4) + S_3(a_4C_4 + S_4d_5 + a_3) \\ S_5C_6 & -S_5S_6 & C_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$U_2 = A_2U_3 = \begin{bmatrix} X_2S_{23} + X_1C_{23} & X_4S_{23} + X_3C_{23} & S_{23}S_4S_5 - C_{23}C_4S_5 & C_2a_2 + C_{23}(a_4C_4 + S_4d_5 + a_3) + S_{23}(C_4d_5 - a_4S_4) \\ X_1S_{23} - X_2C_{23} & X_3S_{23} - X_4C_{23} & -S_{23}S_5C_4 - C_{23}S_4S_5 & S_2a_2 + S_{23}(a_4C_4 + S_4d_5 + a_3) + C_{23}(-C_4d_5 + a_4S_4) \\ S_5C_6 & -S_5S_6 & C_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$U_1 = A_1U_2 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

mit

$$X_1 = C_4C_5C_6 - S_4S_6$$
$$X_2 = -S_4C_5C_6 - C_4S_6$$
$$X_3 = -C_4C_5S_6 - C_4C_6$$
$$X_4 = S_4C_5S_6 - C_4C_6$$

und mit

$$n_x = C_1(X_2S_{23} + X_1C_{23}) - S_1S_5C_6$$
$$n_y = S_1(X_2S_{23} + X_1C_{23}) + C_1S_5C_6$$
$$n_z = -X_1S_{23} + X_2C_{23}$$
$$o_x = C_1(X_4S_{23} + X_3C_{23}) + S_1S_5S_6$$
$$o_y = S_1(X_4S_{23} + X_3C_{23}) - C_1S_5S_6$$
$$o_z = -X_3S_{23} + X_4C_{23}$$
$$a_x = C_1(S_{23}S_4S_5 - C_{23}C_4S_5) - S_1C_5$$
$$a_y = S_1(S_{23}S_4S_5 - C_{23}C_4S_5) + C_1C_5$$
$$a_z = S_{23}S_5C_4 + C_{23}S_4S_5$$
$$p_x = C_1(C_2a_2 + C_{23}a_4C_4 + C_{23}S_4d_5 + C_{23}a_3 + S_{23}C_4d_5 - S_{23}a_4S_4)$$
$$p_y = S_1(C_2a_2 + C_{23}a_4C_4 + C_{23}S_4d_5 + C_{23}a_3 + S_{23}C_4d_5 - S_{23}a_4S_4)$$
$$p_z = -S_2a_2 - S_{23}a_4C_4 - S_{23}S_4d_5 - S_{23}a_3 + C_{23}C_4d_5 - C_{23}a_4S_4$$

Bestimmung der inversen kinemarischen Gleichungen durch sukzessives Multiplizieren mit den inversen A-Matrizen und Determinierung der Winkel oder Winkelsummen unter Verwendung der vorher bestimmten Winkel.

$$U_2 = A_1^{-1}T_6 = \begin{bmatrix} f_{11}(n) & f_{11}(0) & f_{11}(a) & f_{11}(p) \\ f_{12}(n) & f_{12}(0) & f_{12}(a) & f_{12}(p) \\ f_{13}(n) & f_{13}(0) & f_{13}(a) & f_{13}(p) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

mit

$f_{11} = C_1x + S_1y$

$f_{12} = -z$

$f_{13} = -S_1x + C_1y$

$f_{13}(p) = 0$

$\Rightarrow -S_1p_x + C_1p_y = 0$

mit $p_x = r\cos\phi; p_y = r\sin\phi$

$\Rightarrow \phi = \arctan\left(\frac{p_y}{p_x}\right); r = \sqrt{p_x^2 + p_y^2}$

$\sin(\phi - \Theta_1) = 0$

$\Rightarrow \Theta_1 = \arctan\left(\frac{p_y}{p_x}\right)$

$f_{11}(a) = C_1a_x + S_1a_y = S_5(S_{23}S_4 - C_{23}C_4) = S_5(-C_{234})$

$f_{12}(a) = -a_z = S_5(-S_{23}C_4 - C_{23}S_4) = S_5(-S_{234})$

$\Rightarrow \tan(\Theta_2 + \Theta_3 + \Theta_4) = \frac{f_{12}(a)}{f_{11}(a)} = \frac{-a_z}{C_1a_x + S_1a_y}$

$\Rightarrow (\Theta_2 + \Theta_3 + \Theta_4) = \arctan\left(\frac{a_z}{C_1a_x + S_1a_y}\right)$

$$f_{11}(p) = C_1 p_x + S_1 p_y$$
$$= C_2 a_2 + C_{23}(a_4 C_4 + S_4 d_5 + a_3) + S_{23}(C_4 d_5 - a_4 S_4)$$
$$f_{12}(p) = -p_z$$
$$= S_2 a_2 + S_{23}(a_4 C_4 + S_4 d_5 + a_3) + C_{23}(-C_4 d_5 + a_4 S_4)$$

$\Rightarrow$

$$f_{11}(p) = C_2 a_2 + a_4 C_{234} + d_5 S_{234} + a_3 C_{23}$$
$$f_{12}(p) = S_2 a_2 + a_4 S_{234} - d_5 C_{234} + a_3 S_{23}$$

mit

$$u = a_4 C_{234} + d_5 S_{234} = \text{const.}$$
$$v = a_4 S_{234} - d_5 C_{234} = \text{const.}$$

$\Rightarrow$

$$f_{11p} = C_2 a_2 + a_3 C_{23} + u$$
$$f_{12p} = S_2 a_2 + a_3 S_{23} + v$$

quadrieren

$\Rightarrow$

$$(C_2 a_2)^2 + 2 a_2 a_3 C_2 C_{23} + (a_3 C_{23})^2 = (f_{11p} - u)^2$$
$$(S_2 a_2)^2 + 2 a_2 a_3 S_2 S_{23} + (a_3 S_{23})^2 = (f_{12p} - v)^2$$

addieren

$\Rightarrow$

$$a_2{}^2 + a_3{}^2 + 2 a_2 a_3 (C_{23} C_2 + S_{23} S_2) = (f_{11p} - u)^2 + (f_{12p} - v)^2$$

mit

$$C_{23} C_2 + S_{23} S_2 = C_2 C_3 C_2 - S_2 S_3 C_2 + S_2 C_3 S_2 + C_2 S_3 S_2 = C_3 (C_2{}^2 + S_2{}^2)$$
$$= C_3$$

$$\Rightarrow \Theta_3 = \arccos\left(\frac{(C_1 p_x + S_1 p_y - a_4 C_{234} - d_5 S_{234})^2 + (-p_z - a_4 S_{234} + d_5 C_{234})^2 - a_2{}^2 - a_3{}^2}{2 a_2 a_3} \right)$$

$$U_4 = A_3^{-1}A_2^{-1}A_1^{-1}T_6 = \begin{bmatrix} f_{31}(n) & f_{31}(0) & f_{31}(a) & f_{31}(p) - a_2C_3 - a_3 \\ f_{32}(n) & f_{32}(0) & f_{32}(a) & f_{32}(p) + a_2S_3 \\ f_{33}(n) & f_{33}(0) & f_{33}(a) & f_{33}(p) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

mit

$f_{31} = C_{23}f_{11} + S_{23}f_{12}$

$f_{32} = -S_{23}f_{11} + C_{23}f_{12}$

$f_{33} = f_{13}$

$S_4d_5 + C_4a_4 = C_{23}f_{11p} + S_{23}f_{12p} - a_2C_3 - a_2$

$-C_4d_5 + S_4a_4 = C_{23}f_{12p} - S_{23}f_{11p} + a_2S_3$

mit

$k = a_2C_3 + a_3$

$l = a_2S_3$

quadrieren

$(S_4d_5)^2 + 2S_4d_5C_4a_4 + (C_4a_4)^2 = (C_{23}f_{11p} + S_{23}f_{12p} - k)^2$

$(C_4d_5)^2 - 2C_4d_5S_4a_4 + (S_4a_4)^2 = (C_{23}f_{12p} - S_{23}f_{11p} + l)^2$

addieren

$$\begin{aligned} a_4^2 + d_5^2 &= [(C_{23}f_{11p})^2 + 2C_{23}S_{23}f_{11p}f_{12p} + (S_{23}f_{12p})^2] - 2k(C_{23}f_{11p} + S_{23}f_{12p}) + k^2 \\ &\quad + [(C_{23}f_{12p})^2 - 2C_{23}S_{23}f_{11p}f_{12p} + (S_{23}f_{11p})^2] + 2l(C_{23}f_{12p} - S_{23}f_{11p}) + l^2 \\ &= f_{11p}^2 + f_{12p}^2 + k^2 + l^2 + C_{23}(2lf_{12p} - 2kf_{11p}) + S_{23}(-2lf_{11p} - 2kf_{12p}) \end{aligned}$$

mit

$e = 2lf_{12p} - 2kf_{11p}$

$f = -2lf_{11p} - 2kf_{12p}$

$d = a_4^2 + d_5^2 - (f_{11p}^2 + f_{12p}^2 + k^2 + l^2)$

$\Rightarrow d = eC_{23} + fS_{23}$

mit $\quad f = r\cos\phi \quad ; \quad e = r\sin\phi$

$$\phi = \arctan\left(\frac{e}{f}\right); r = \sqrt{e^2 + f^2}$$

$$\sin(\phi + \Theta_{23}) = \frac{d}{r} \qquad \cos(\phi + \Theta_{23}) = \pm\sqrt{1 - (\frac{d}{r})^2}$$

$$\Rightarrow \Theta_{23} = -\arctan\left(\frac{e}{f}\right) + \arctan\left(\frac{d}{-\sqrt{e^2 + f^2 - d^2}}\right)$$

$$\Rightarrow \Theta_4 = \Theta_{234} - \Theta_{23}$$
$$\Rightarrow \Theta_2 = \Theta_{23} - \Theta_3$$

$$U_5 = A_4^{-1}A_3^{-1}A_2^{-1}A_1^{-1}T_6 = \begin{bmatrix} f_{41}(n) & f_{41}(0) & f_{41}(a) & f_{41}(p) + a_2(C_{34} - C_4) - a_4 \\ f_{42}(n) & f_{42}(0) & f_{42}(a) & f_{42}(p) \\ f_{43}(n) & f_{43}(0) & f_{43}(a) & f_{43}(p) + a_2(S_{34} - S_4) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

mit

$$f_{41} = C_4 f_{31} + S_4 f_{32}$$
$$f_{42} = f_{33}$$
$$f_{43} = S_4 f_{31} - C_4 f_{32}$$

$$f_{41a} = -S_5$$
$$f_{42a} = C_5$$
$$\Rightarrow \tan\Theta_5 = \frac{-f_{41a}}{f_{42a}} = -\frac{C_4 f_{31a} + S_4 f_{32a}}{f_{33a}}$$
$$= -\frac{C_4(C_{23}f_{11a} + S_{23}f_{12a}) + S_4(-S_{23}f_{11a} + C_{23}f_{12a})}{f_{13a}}$$

$$\Rightarrow$$
$$\Theta_5 = \arctan\left(\frac{C_4(C_{23}[C_1 a_x S_1 a_y] + S_{23}[-a_z]) + S_4(-S_{23}[C_1 a_x S_1 a_y] + C_{23}[-a_z])}{-S_1 a_x + C_1 a_y}\right)$$

$$f_{41n} = C_5 C_6$$
$$f_{41o} = -C_5 S_6$$
$$\Rightarrow \tan\Theta_6 = \frac{-f_{41o}}{f_{41n}} = -\frac{C_4 f_{31o} + S_4 f_{32o}}{C_4 f_{31n} + S_4 f_{32n}}$$

$$\Rightarrow \Theta_6 = \arctan\left(\frac{C_4(C_{23}[C_1 o_x S_1 o_y] + S_{23}[-o_z]) + S_4(-S_{23}[C_1 o_x S_1 a_y] + C_{23}[-o_z])}{C_4(C_{23}[C_1 n_x S_1 n_y] + S_{23}[-n_z]) + S_4(-S_{23}[C_1 n_x S_1 n_y] + C_{23}[-n_z])}\right)$$

B.2. Implementierung der inversen kinematischen Gleichungen für das Robotersimulationsprogramm TOROS

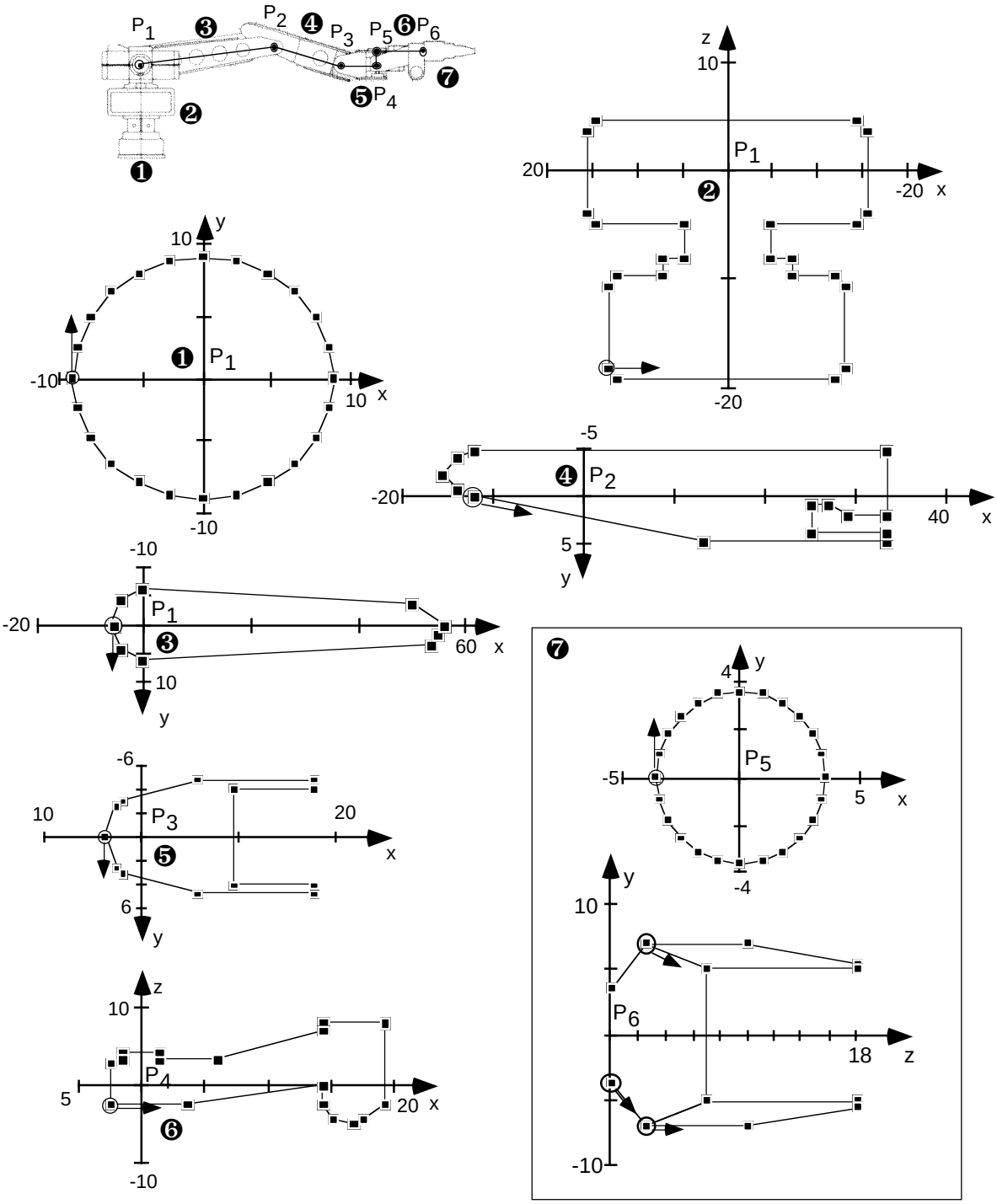
```
/*********************************************************************/
cinInvKraft(r, pos, or, conf, velFlg)
Robot *r;
punto3d pos, or;
int conf, velFlg;
{/*********************************************************************/
int i;
float DH[4][4], ang[6];
double f11a, f11p, f12p, f12a, f13a, f13n, f13o, f31a, f32a, f41a, f42a,
    f11o, f11n, f12o, f12n, f31o, f31n, f32o, f32n, f41o, f41n,
    u, v, w, k, l, d, e, f, ang23, ang234, ang23_1, ang23_2, t_val;
static int inic=0;
static float a2, a3, a4, d5;

if (!inic) {
        a2=r->modCin[1][3][0];
        a3=r->modCin[2][3][0];
        a4=r->modCin[3][3][0];
        d5=r->modCin[4][3][2];
        inic=1;
}
getDH(r, pos, or, DH);

/****************** Teta 1 ********************/
if (DH[3][0] <= 0)
        ang[0]=-arcTag(DH[3][1], -DH[3][0]);
else
        ang[0]=arcTag(DH[3][1], DH[3][0]);

if (verifLim(r, 0, ang, velFlg) != PACA_OK)
```

```
        return NO_SOLUCION;

f11a=cos(ang[0])*DH[2][0]+sin(ang[0])*DH[2][1];
f12a= -DH[2][2];
f13a= -sin(ang[0])*DH[2][0]+cos(ang[0])*DH[2][1];
f11p=cos(ang[0])*DH[3][0]+sin(ang[0])*DH[3][1];
f12p= -DH[3][2];
f11o=cos(ang[0])*DH[1][0]+sin(ang[0])*DH[1][1];
f11n=cos(ang[0])*DH[0][0]+sin(ang[0])*DH[0][1];
f12o= -DH[1][2];
f12n= -DH[0][2];

ang234=arcTag(f12a,f11a);

u=a4*cos(ang234)+d5*sin(ang234);
v=a4*sin(ang234)-d5*cos(ang234);

/****************** Teta 3 ********************/

w = ((SQR(f11p-u)+SQR(f12p-v))-SQR(a2)-SQR(a3))/(2*a2*a3);
if (fabs(w) > 1)
        w = 1.0;
ang[2]=arcTag(sqrt(1-SQR(w)),w);

if (verifLim(r, 2, ang, velFlg) != PACA_OK)
        return NO_SOLUCION;

k=a2*cos(ang[2])+a3;
l=a2*sin(ang[2]);
e=2*(l*f12p-k*f11p);
f= -2*(l*f11p+k*f12p);
d=SQR(a4)+SQR(d5)-(SQR(f11p)+SQR(f12p)+SQR(k)+SQR(l));
```

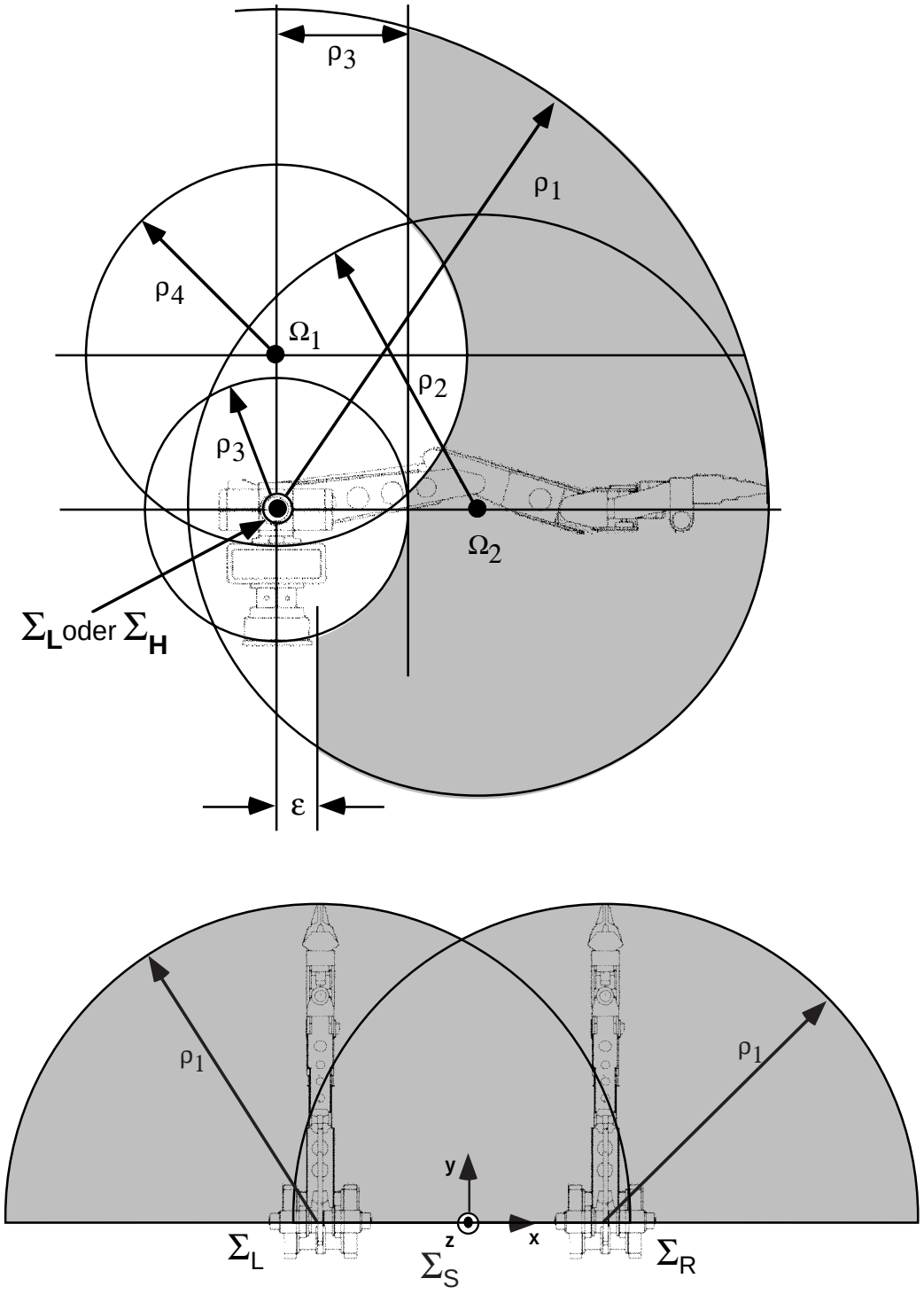
```
if (SQR(e)+SQR(f) < SQR(d)) {
        sprintf(torosError, "GRIPS: Punto fuera del alcance del robot\n");
        return NO_SOLUCION;
}

ang23_1=-arcTag(e,f)+arcTag(d, -sqrt(SQR(e)+SQR(f)-SQR(d)));
ang23_2=arcTag(e,-f)-arcTag(d, -sqrt(SQR(e)+SQR(f)-SQR(d)));

t_val=cos(ang23_1-ang[2])*a2+a4*cos(ang234)+d5*sin(ang234)+a3*cos(ang23_1);

if (IGUAL(t_val, f11p))
                ang23 = ang23_1;
        else
                ang23 = ang23_2;

/****************** Teta 4 ********************/
ang[3]=ang234 - ang23;

if (verifLim(r, 3, ang, velFlg) != PACA_OK)
        return NO_SOLUCION;

/****************** Teta 2 ********************/
ang[1]=ang23 - ang[2];

if (verifLim(r, 1, ang, velFlg) != PACA_OK)
        return NO_SOLUCION;

f32a= -sin(ang23)*f11a + cos(ang23)*f12a;
f31a=cos(ang23)*f11a + sin(ang23)*f12a;
f42a=f13a;
f41a=cos(ang[3])*f31a + sin(ang[3])*f32a;

/****************** Teta 5 ********************/
```

```
ang[4]=arcTag(-f41a,f42a);

if (verifLim(r, 4, ang, velFlg) != PACA_OK)
        return NO_SOLUCION;

/****************** Teta 6 ******************/

f31o=cos(ang23)*f11o + sin(ang23)*f12o;
f31n=cos(ang23)*f11n + sin(ang23)*f12n;
f32o= -sin(ang23)*f11o + cos(ang23)*f12o;
f32n= -sin(ang23)*f11n + cos(ang23)*f12n;
f41o=cos(ang[3])*f31o + sin(ang[3])*f32o;
f41n=cos(ang[3])*f31n + sin(ang[3])*f32n;

if (IGUAL(cos(ang[4]), 0.0))
        ang[5]=r->valArt[5]*r->relArt[5][0]+r->relArt[5][1];
else if (cos(ang[4]) > 0)
        ang[5]=arcTag(-f41o,f41n);
else if (cos(ang[4]) < 0)
        ang[5]=arcTag(f41o,-f41n);

if (verifLim(r, 5, ang, velFlg) != PACA_OK)
        return NO_SOLUCION;

for (i=0; i < 6; i++) {
        r->valArt[i]=(ang[i]-r->relArt[i][1])/r->relArt[i][0];
/*
        printf("ang %d = %f\n", i+1, valArt[i] rad);
*/
}
return PACA_OK;
```

B.3. Definition der mechanischen Teile des Roboters GRIPS in TOROS

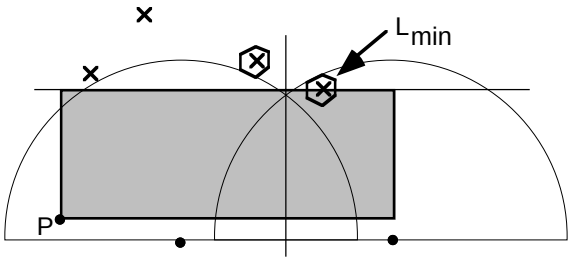
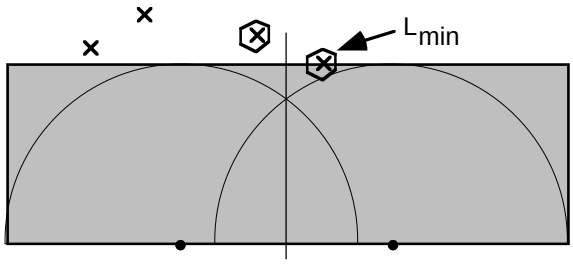
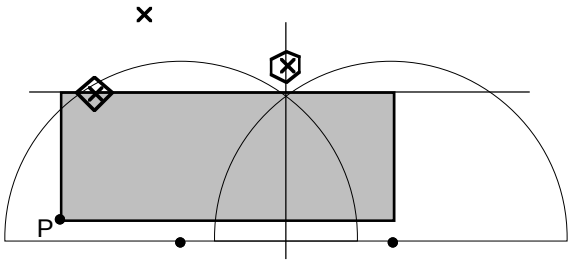
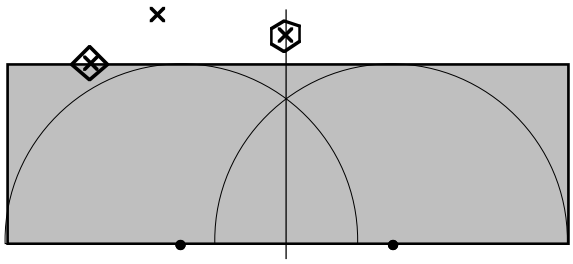
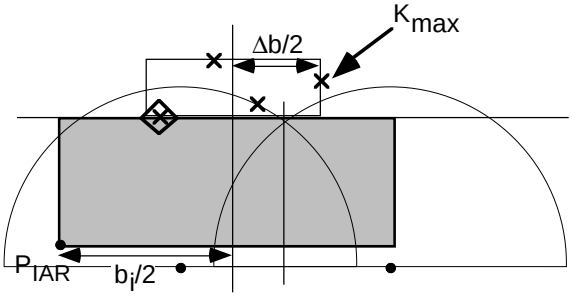
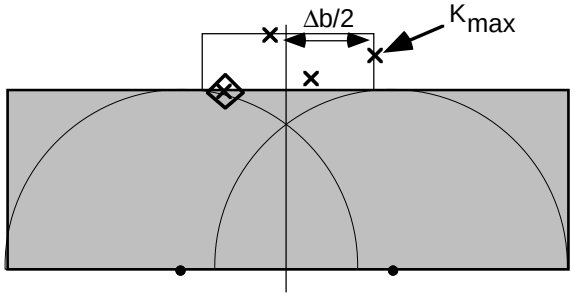

P1
P2
P3
P4
P5
P6
❶
❷
❸
❹
❺
❻
❼
z
10
20
-20
x
y
-10
10
-5
40
5
-6
6
-4
18

B.4. Begrenzung der Arbeitsraumprojektionen durch Kreise und Geraden



B.5. Qualitative Skizzen zur Bestimmung von x_a und y_a

Die folgende Tabelle zeigt anschaulich die Überlegungen, die man in den beiden Fällen der Beschränkung und Nichtbeschränkung des Arbeitsraumes zur Bestimmung der Anfangsplazierung für den Plausibilitätstest macht.

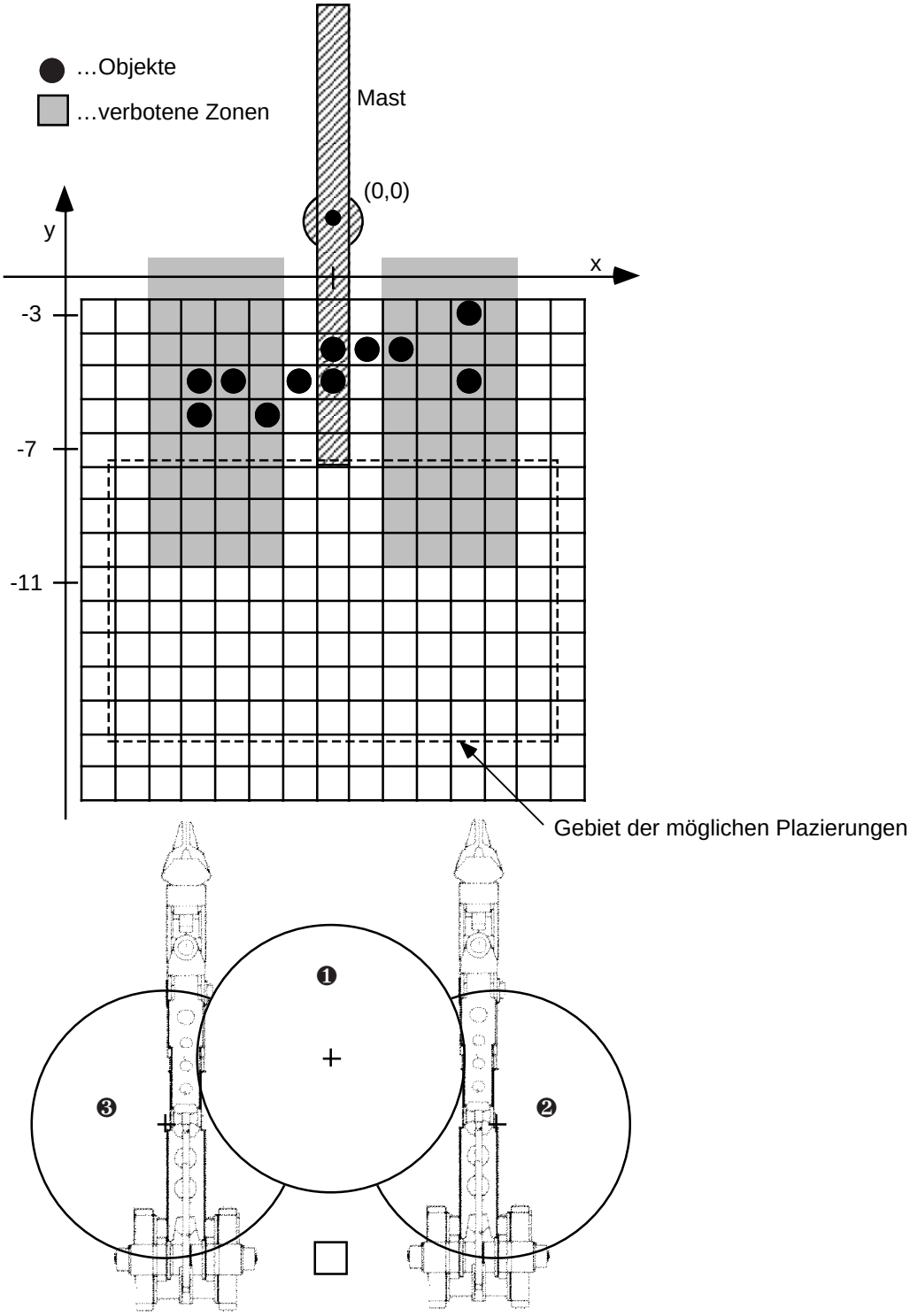
Arbeitsraumbeschränkung	**Keine Arbeitsraumbeschränkung**
	
mehrere Objekte für Zweiarmmanipulationen	
	
ein Objekt für Zweiarmmanipulationen	
	
kein Objekt für Zweiarmmanipulationen	

…ZAO

…L_{min}

B.6. Beispiel zur Bewertung der Plazierung

B.6.1. Definition der Objekte sowie der Vorzugsgebiete des Systems



B.6.2. MAPLE-Programmcode zur Bewertung der Plazierung für ein gegebenes Beispiel

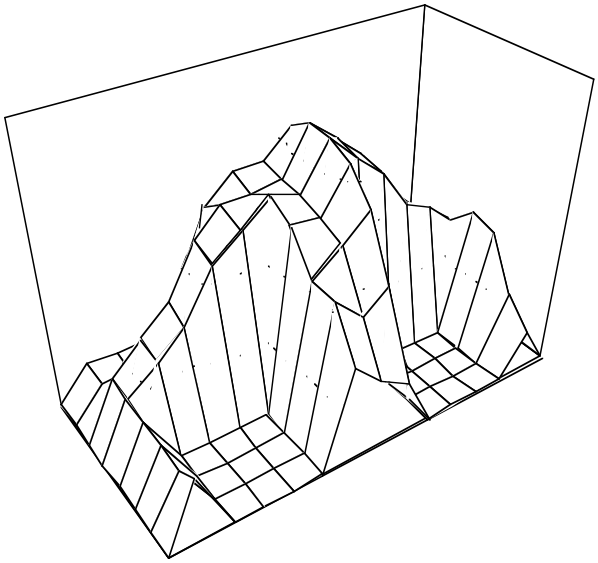
```
with(plots):
readlib(write):
open(out.dat):

## Definition der Parameter ##
num_obj := 11:
nu1 := 0.5:
nu2 := 1:
g1 := 5:
g2 := 5:
g3 := 5:

## Definition der Objekte ##
O[1] := -4,-6:
O[2] := -4,-5:
O[3] := -3,-5:
O[4] := -2,-6:
O[5] := -1,-5:
O[6] := 0,-5:
O[7] := 0,-4:
O[8] := 1,-4:
O[9] := 2,-4:
O[10] := 4,-3:
O[11] := 4,-5:
plotlist := NULL:

  for y from -15 to -7 do
        excellist := NULL:
        for x from -6 to 6 do
```

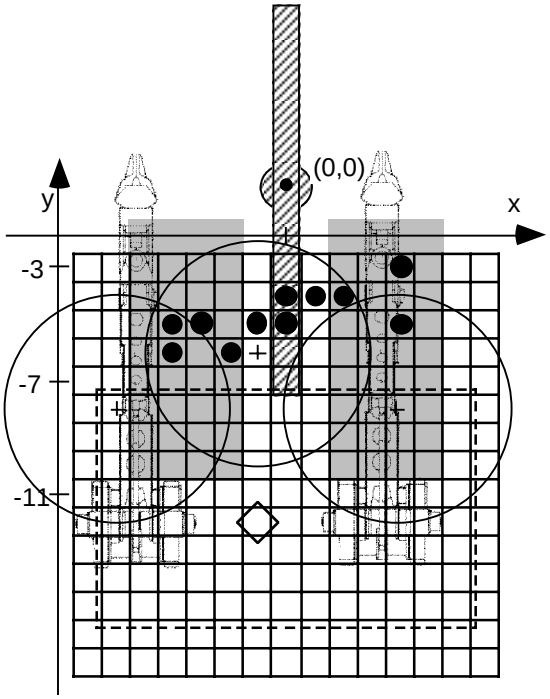
```
######## Test verbotene Gebiete (Kriterium 3) #########
q3 := 1:

### Gebiet 1 ###
if (x>(-6)) and (x<(-1)) and (y>(-11)) then q3 := 0  fi;

### Gebiet 2 ###
if (x<6) and (x>1) and (y>(-11)) then q3 := 0  fi;

#############################################################
######### Kriterium 1 (Abstand der Objekte)  ##########
q1 := 0;

for o from 1 to num_obj do
  q1h := q1:
  q1 := q1h + sqrt( (x - O[o][1])^2 + (y - O[o][2])^2 ):
od;

#############################################################
########### Kriterium 2 (Vorzugsgebiete) ###############
q2 := 1;
## Gebiet mit Güte g1 (Kreis mit Radius 4 MP1(0,6)) ##
a := 0 : b := 6:
rad1 := 4:
for o from 1 to num_obj do
        q2h := q2;
        r1 := sqrt( ((x+a) - O[o][1])^2 + ((y+b) - O[o][2])^2 ):
        if (evalf(r1) < rad1) then q2 := q2h + g1  fi:
od;

## Gebiet mit Güte g2 (Kreise mit Radius 4 MP2(6,3)) ##
a := 6 : b := 3:
rad2 := 4:
```

```
        for o from 1 to num_obj do
                q2h := q2;
                r2 := sqrt( ((x+a) - O[o][1])^2 + ((y+b) - O[o][2])^2 ):
                if (evalf(r2) < rad2) and not(evalf(r1) < rad1)
                                then q2 := q2h + q2  fi:
        od;

        ## Gebiet mit Güte g3(Kreise mit Radius 4 ,MP3(-6,3)) ##
        a := -6 : b := 3:
        rad3 := 4:
        for o from 1 to num_obj do
                q2h := q2;
                r3 := sqrt( ((x+a) - O[o][1])^2 + ((y+b) - O[o][2])^2 ):
                if (evalf(r3) < rad3) and not(evalf(r1) < rad1)
                                then q2 := q2h + g3  fi:
        od;

        G := evalf(q1^nu1 * q2^nu2 * q3);

        ############################################################
        if (G<>0) then plotlist := plotlist,[x,y,G] fi;
        excellist := excellist,trunc(G):

        od;
        writeln(excellist):
 od;
## Visualisierung der Daten ##
pointplot({plotlist},axes=BOXED);

close(out.dat):
```

B.6.3. Alternative 1

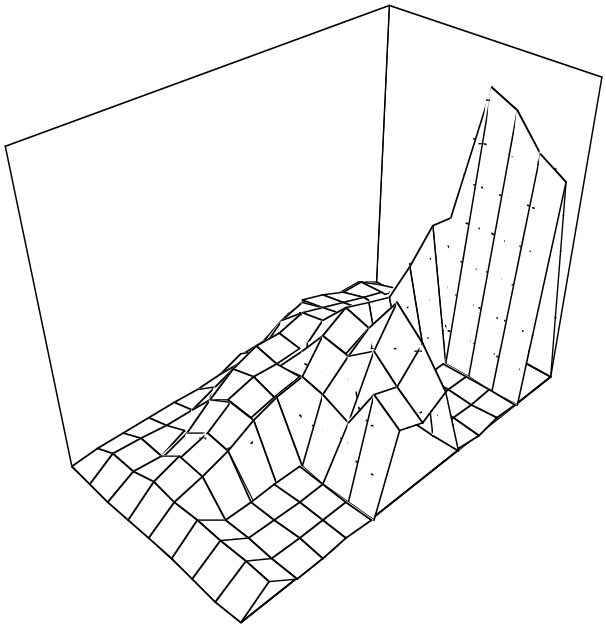
Gewichtung der Kriterien		Güte der Vorzugsgebiete g_i			beste Positionierung
n_1	n_1	g_1	g_2	g_3	(x,y)
0,5	1	20	5	5	(1,-12)

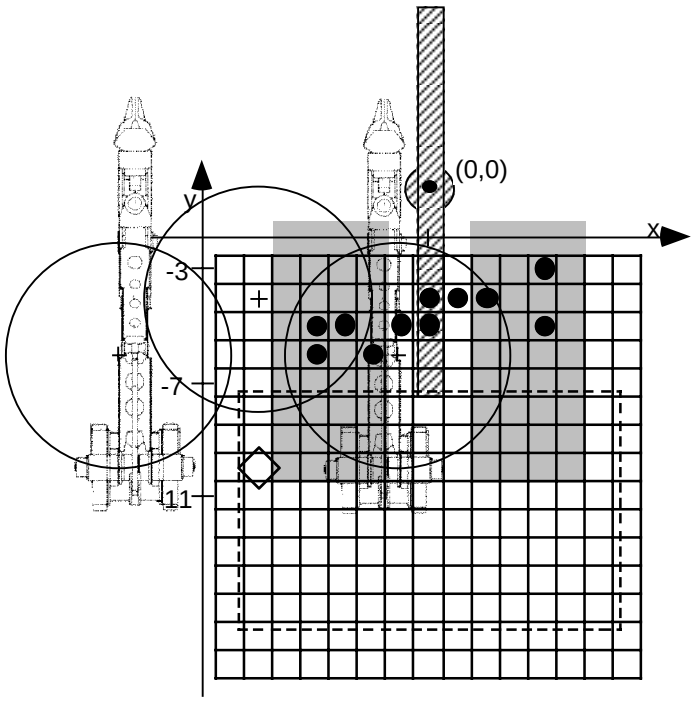




B.6.4. Alternative 2

Gewichtung der Kriterien		Güte der Vorzugsgebiete g_i			beste Positionierung
n_1	n_1	g_1	g_2	g_3	(x,y)
0,5	1	5	20	1	(-6,-10)





B.7. Programmcode in MAPLE zur Berechnung diskreter Arbeitsraumrepresentationen

```
wsnum := proc():

    ## -- BERECHNUNG DES ARBEITSRAUMES FUER ACAD VISUALISIERUNG --
    ## --- librarys lesen ---
    readlib(write):
    with(plots):

    ## --- datenfile oeffnen ---
    open(daten_workspace):

    ## --- Orietierung des vektors a eingeben ----
    ang_phi := args[1]:
    ang_theta := args[2]:
    phi := evalf(ang_phi):
    theta :=evalf(ang_theta):

    alpha2 := -1:
    alpha5 := -1:
    alpha6 := -1:
    offset2 := -0.1396:
    offset3 := Pi/6:
    offset4 := 0:
    offset5 := -Pi/2:
    offset6 := Pi/2:
    TCPoffset := 18:

    a2 := 54:a3 := 32.4:a4 := 12:d5 := 4.8:
    d6 := 21.2+TCPoffset:

    xmin := 0: xmax := 130:
    ymin := -130: ymax := 130:
```

```
zmin := 0: zmax := 0:

delta :=10:
EPS := 0.0001:

## -------- deklaration der achsenlimits -----------
t1min := -Pi/2 : t1max:= Pi/2 :
t2min := alpha2*0 + offset2 : t2max:= alpha2*2*Pi/3 + offset2:
t3min := 0 + offset3 : t3max:= 1.92 + offset3:
t4min := -0.873 + offset4 : t4max:= 0.873 + offset4:
t5min := alpha5*(-0.917) + offset5 : t5max:= alpha5*0.917 + offset5:
t6min := alpha6*(-Pi) + offset6 : t6max:= alpha6*Pi + offset6:

## ------ parameter fuer orientierung ---------
ax := cos(phi)*cos(theta): ay := sin(phi)*cos(theta):
az :=-sin(theta):

## ------- initialierung der seqences -----------
plotlist := NULL:

for z from zmin by delta to zmax do
        i := 0:
        for x from xmin by delta to xmax do
                j := i: i := j + 1:
                for y from ymin by delta to ymax do

## -------- DHcin = DHreal * inv(Trans_d6) ----------
px := -ax*d6 + x:
py := -ay*d6 + y:
pz := -az*d6 + z:

## -------  Teta 1 ---------
if (px >= 0) then
```

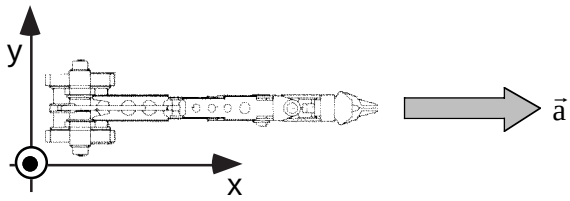
```
                    t1 := arctan(py,px):
        else
                    t1 := -arctan(py,-px):
        fi:

        if (evalf(t1) > evalf(t1max)) then next
        elif (evalf(t1) < evalf(t1min)) then next
        fi:

        f11a := evalf(cos(t1)*ax + sin(t1)*ay):
        f12a := -az:
        f13a := evalf(-sin(t1)*ax + cos(t1)*ay):
        f11p := evalf( cos(t1)*px + sin(t1)*py):
        f12p := -pz:

        ## -------  Teta 234 ---------
        t234 := arctan(f12a,f11a):

        u := evalf(a4*cos(t234) + d5*sin(t234)):
        v := evalf(a4*sin(t234) - d5*cos(t234)):

        ## Abfangen der Ungenauigkeiten ##
        w := evalf((evalf((f11p-u)^2) + evalf((f12p-v)^2) - evalf(a2^2) -
evalf(a3^2))/evalf(2*a2*a3)):

        if (w>1) then w := 1
        elif (w<0) then next
        fi:

        ## ------ Teta 3 ----------
        t3 := evalf(arctan(sqrt(1-w^2),w)):

        if (evalf(abs(t3-evalf(offset3))) < 0.00001) then
```

```
        t3 := evalf(offset3)
fi:

if (evalf(t3) > evalf(t3max)) then next
elif (evalf(t3) < evalf(t3min)) then next
fi:

k := evalf(a2*cos(t3) + a3):
l := evalf(a2*sin(t3)):

e := evalf(2*l*f12p - 2*k*f11p):
f := evalf(-2*l*f11p - 2*k*f12p):
d := evalf(a4^2 + d5^2 - (f11p^2 + f12p^2 + k^2 + l^2)):

## -------  Teta 23 ---------
t23_1 := -arctan(e,f) + arctan(d,-sqrt(e^2 +f^2 - d^2)):
t23_2 := arctan(e,-f) - arctan(d,-sqrt(e^2 +f^2 - d^2)):

t_val := cos(t23_1-t3)*a2+a4*cos(t234)+d5*sin(t234)+a3*cos(t23_1);

if ((t_val<(f11p+EPS)) and (t_val>(f11p-EPS))) then t23 := t23_1
else t23 := t23_2
fi:

## -------  Teta 4 ---------
t4 := evalf(t234 - t23):

if (evalf(t4) > evalf(t4max)) then next
elif (evalf(t4) < evalf(t4min)) then next
fi:

## -------  Teta 2 ---------
t2 := evalf(t23 - t3):
```

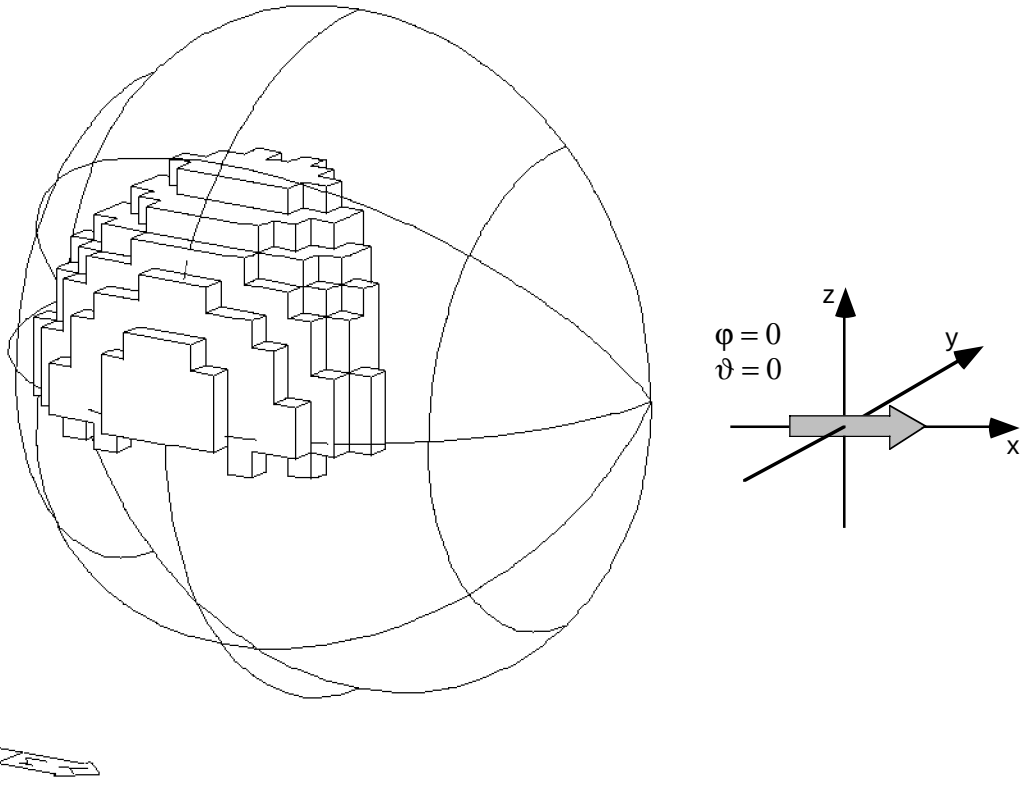
```
        if (evalf(t2) < evalf(t2max)) then next
        elif (evalf(t2) > evalf(t2min)) then next
        fi:

        f33a := f13a:
        f32a := evalf(-sin(t23)*f11a + cos(t23)*f12a):
        f31a := evalf(cos(t23)* f11a + sin(t23)*f12a):
        f42a := f33a:
        f41a := evalf(cos(t4)*f31a + sin(t4)*f32a):

        ## -------  Teta 5 ---------
        t5 := arctan(-f41a,f42a):

        if (evalf(t5) < evalf(t5max)) then next
        elif (evalf(t5) > evalf(t5min)) then next
        fi:
        ## ----------- eintrag des erreichbaren Punktes -----------
        ## ---- in datei fuer ACAD ---
        writeln(x,y,z);
        ## ------------------------------------------------------
           od:  ## ende for y
         od:   ## ende for x
        od:    ## ende for z

        close(daten_workspace);

        ## ---------- ENDE ----------

end;
```

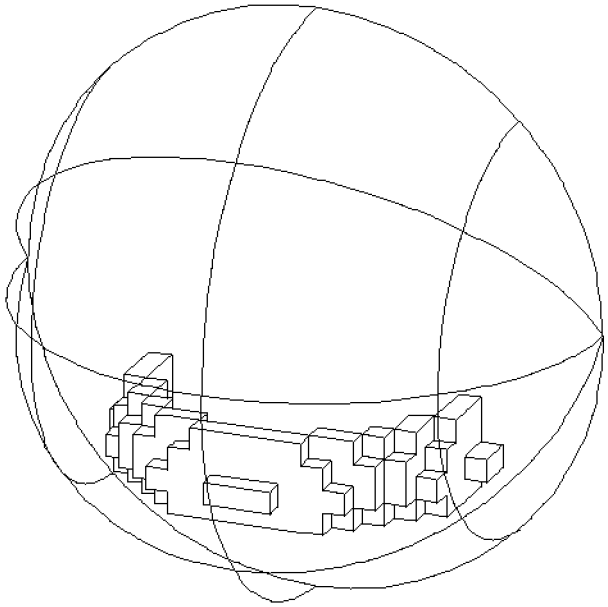
B.8. Visualisierung von Arbeitsräumen unter Berücksichtigung verschiedener Orientierungen

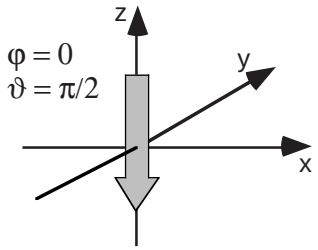
Nachfolgende Abbildung zeigt die Ruheorientierung des Roboters, auf der die Berechnungen der Arbeitsräume basieren. Die skizzierte Halbkugel stellt näherungsweise den Arbeitsraum ohne Berücksichtigung von Orientierungen dar.

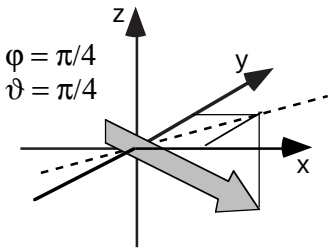


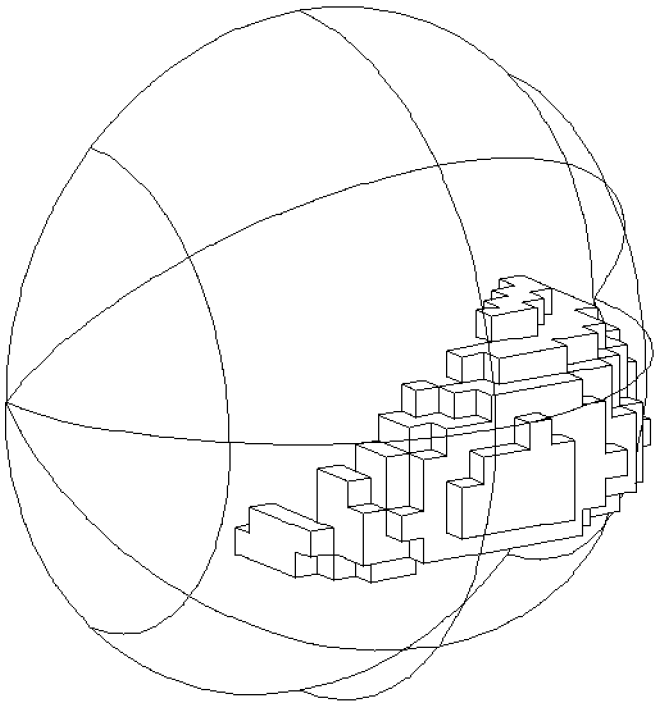
Ruheorientierung des Roboters GRIPS






z
φ = 0
ϑ = π/2
y
x


z
φ = π/4
ϑ = π/4
y
x



B.9. Repräsentation des Roboters GRIPS in TOROS

B.10. Repräsentation des Gesamtsystems in TOROS

Literaturverzeichnis

[1] Borrel, P., Leigois, A.
A Study of Manipulators Inverse Kinematic Solutions with Applications toTrayectory Planning and Workspace Determination
IEEE '86, International Conference on Robotics and Automation, pages 1180-1185
San Fransisco, C.A. April 1986

[2] Paul, P.
Robot Manipulators: Mathematics, Programming and Control
MIT Press, Cambrige, Mass. 1981

[3] Alameldin, T. K.
Threedimensional Workspace Determination for redundant articulated Chains.
Thesis
University of Pennsylvania, 1991

[4] Dillmann, R., Huck, M.
Informationsverarbeitung in der Robotik
Springer, Berlin 1991

[5] Becquet, M.
Computing of Robot´s Workspace
ICAR ´85, 1985, pages 289-295

[6] Paul, R.P., Shimano, B., Mayer, G.E.
Kinematic Control Equations for Simple Maniplators
IEEE Transactions on Systems, Man and Cybernetics, 6. June 1981

[7] Bronstein, Semendjajew
Taschenbuch der Mathematik
Teubner, Leipzig, 1979

[8] Kraft Telerobotics Inc.
11667 West 90th Street, Overland Park, KS 66214, USA
Description of GRIPS Force Feedback Manipulator System

[9] Balaguer, C., Rodríguez, F.
TOROS. Graphical Toolbox for Robot Simulation
International Workshop on Graphics & Robotics
Schloß Dagstuhl, Germany, 19.-22. April 1993

[10] Char, B.W., Geddes, K.O.
MAPLE V Language Reference Manual
Springer Verlag, New York, 1991

[11] McCloy, D., Harris, D.M.J.
Robotertechnik; Einführung
VCH - Verlag, Weinheim, 1989

[12] Geiser,G.
Mensch-Maschine-Kommunikation
R. Oldenbourg Verlag, München, 1990