Lernen der Handhabung von Alltagsgegenständen im Kontext eines Service-Roboters

Dissertation
zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)

vorgelegt von

Tim Baier-Löwenstein

Januar 2008

Universität Hamburg
Fakultät für Mathematik, Informatik
und Naturwissenschaften
Arbeitsbereich Technische Aspekte
Multimodaler Systeme
Vogt-Kölln-Str. 30
D–22527 Hamburg

Gutachter:	Prof. Dr. Jianwei Zhang Prof. Dr. Bernd Neumann
Vorsitzender des Prüfungsausschusses:	Prof. Dr.-Ing. Dietmar P.F. Möller
Tag der Disputation:	25.06.2008

Kurzfassung

Obwohl das Greifen von Objekten seit mehreren Jahren Gegenstand aktueller Forschung ist, stellt die Handhabung von Alltagsgegenständen auch für moderne Robotersysteme immer noch ein großes Problem dar. Dies liegt nicht zuletzt darin begründet, dass noch bis vor wenigen Jahren nur relative einfache Greifwerkzeuge für die Anbindung an einen Roboter zur Verfügung standen. Durch die zunehmende Miniaturisierung elektronischer Bauteile ist es mittlerweile möglich, Roboterhände mit einem „menschenähnlichen“ Funktionsumfang zu konstruieren, die sich durch eine geringe Größe auszeichnen und sich somit in einen mobilen Roboter integrieren lassen.

Aber auch die Entwicklung von robusten und effizienten Verfahren zur Berechnung von Griffen muss noch fortgeführt werden. Es gibt zwar eine Vielzahl unterschiedlicher Ansätze, jedoch sind die bisher entwickelten Verfahren bezüglich der Objekte, mit denen sie umgehen können, sehr eingeschränkt, oder sie sind nur in der Lage, Griffe mit einer zuvor festgelegten Anzahl von Kontaktpunkten zu berechnen. Daher werden in dieser Arbeit verschiedene Verfahren für die Berechnung von Griffen umgesetzt. Die Berechnung von Griffen mittels selbstbewertendem Lernen in einer Simulationsumgebung bildet den Kern der Arbeit. Auf der Basis von selbstbewertendem Lernen wird ein Verfahren präsentiert, das Griffe mit einer beliebigen Zahl von Kontaktpunkten für beliebige Objekte generiert. Hierfür wird ein bekanntes Verfahren adaptiert und an die Anforderungen der Griffgenerierung angepasst. Es wird gezeigt, dass das Lernverfahren bei der Generierung von Griffen besser ist als ein Verfahren, das auf einer Breitensuche basiert. Die Ergebnisse der Griffberechnung mittels Simulation werden mit dem Service-Roboter evaluiert.

Auch auf dem Gebiet der Griffanalyse besteht noch Forschungsbedarf. Die meisten der Qualitätskriterien für einen Griff beruhen auf einer Analyse der Kräfte, die während des Greifvorgangs auftreten bzw. die nach einem Griff erforderlich sind, um ein Objekt festzuhalten. Dies ist leider für einen Laien, der einen Service-Roboter benutzen will, nur schwer verständlich. Zudem wird die Funktionalität des Objekts, die Objektsemantik, nicht beachtet, so dass Griffe teilweise unbrauchbar für eine gestellte Aufgabe sind. Wenn z.B. eine Tasse nicht von der Seite, sondern von oben gegriffen wird, so kann nichts mehr eingefüllt werden. Diese Eigenschaft eines Griffs wird von den bisher eingesetzten Verfahren zur Griffanalyse nicht berücksichtigt. In dieser Arbeit wird ein Verfahren entwickelt, das die Qualität eines Griffs auf der Basis der Weiterverwendbarkeit eines gegriffenen Objekts bestimmt. An verschiedenen Beispielen wird die Anwendbarkeit des Qualitätskriteriums illustriert.

Als Demonstrationsplattform für die beschriebenen Entwicklungen dient ein Service-Roboter. Dieser mobile Service-Roboter ist mit einem Roboterarm und einer Roboterhand ausgestattet, die es ermöglichen soll, dass der Roboter in einer unstrukturierten Umgebung Objekte aus dem alltäglichen Leben greifen und manipulieren kann. Die Weiterentwicklung des Robotersystems, insbesondere die Anbindung einer künstlichen Hand, ist Teil dieser Arbeit.

Abstract

Although object grasping has been a research topic for some years, the handling of daily objects is still a very challenging problem for modern service robots. This is partly due to the fact that until a few years ago, only simple grippers could be integrated into a robot. However, the miniaturisation of electronic devices has made it possible to build small robotic hands with a humanlike operabilty that can also be integrated into a mobile robot.

In addition to this development, robust and efficient techniques for grasp computation need to be researched as well. There are various different approaches, but the methods developed so far are very restrictive concerning the objects they can handle, or they are only able to compute grasps with a fixed number of contact points. Therefore, several new kinds of methods for grasp computation are developed in this thesis. The main focus is on the computation of grasps by reinforcement learning in a simulation. A method based on reinforcement learning which is able to compute grasps for arbitray objects with an arbitray number of contacts is presented. For this purpose, an algorithm is adapted and assimilated to fit the requirements of grasp generation. It is shown that a method based on learning is more efficient than a method based on breadth-first search. The results of the grasp computation are evaluated on a service-robot.

The topic of grasp analysis also requires further research. Most quality criteria for grasps are based on the analysis of forces occuring during a grasp procedure, or the forces needed after the procedure to maintain a firm grip on the object. These criteria are hard to understand for laymen who want to use a service robot. Furthermore, the functionality of an object, the object semantics, is not taken into account by these criteria, so that a grip might be useless for a certain task. For example, if a cup is grasped from the top and not from the side, it is impossible to fill something into the cup. This attribute is not considered by any of the methods used for grasp analysis to date. In this thesis, a method is developed for computing the quality of a grasp based on the re-usability of the object. The applicability of the method is demonstrated in different examples.

The platform used for the demonstration of the described methods is a service robot. This mobile service robot is equipped with a robot-arm and a robotic hand. This setup enables the robot to grasp and manipulate everyday objects in an unstructured environment. The enhancement of the robotic system, especially the integration of the artificial hand, is part of this thesis.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich während meiner Promotion unterstützt haben.

Zunächst möchte ich mich bei meinem Arbeitsbereichsleiter Prof. Dr. Jianwei Zhang bedanken, der mich betreut und die erforderlichen Ressourcen für die Forschungsarbeit bereitgestellt hat. Des Weiteren möchte ich Prof. Dr. Bernd Neumann danken, der sich als Zweitgutachter bereit erklärt hat diese Arbeit zu betreuen.

Besonders möchte ich mich beim Team des Informatik-Rechenzentrums bedanken, insbesondere bei Herrn Friesland Köpke. Für meine Berechnungen hat mir das Team um Herrn Köpke einige Compute-Server zur Verfügung gestellt und auf meine Bedürfnisse angepasst. Ohne dies wären die Berechnungen der Lernvorgänge und die umfangreichen Experimente nicht durchführbar gewesen.

Weiterhin möchte ich mich bei allen Mitarbeitern des AB TAMS für Ihre Unterstützung in zahlreichen Diskussionen bedanken. Besonderer Dank gilt Daniel, für seine Anregungen zur Softwarearchitektur und Roblets, Markus für sein „Teaching by Demonstration“ Framework, Ole für sein Maschinenbau-Vordiplom und Lu für die Korrekturen und Anmerkungen in zahlreichen Papern. Des Weiteren möchte ich mich bei Norman, Bernd, Andreas, Hannes, Sascha, Martin, Thorsten, Kai und Christine für die wissenschaftlichen Diskussionen zu dieser Arbeit bedanken.

Mein Dank gilt auch Birte für die VRML-Modelle sowie Achim und Karola für Ihre Unterstützung und Motivation. Last but not least möchte ich mich noch bei meiner Frau Sylvia bedanken, ohne deren Unterstützung und Verzicht auf Freizeitaktivitäten diese Dissertation nie hätte geschrieben werden können.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Service-Roboter

In den Zukunftsszenarien von Forschungs- und Wirtschaftsinstituten nehmen Service-Roboter einen wichtigen Platz ein: [Infratest 2004, OECD 2004, OECD 1998] und [IFR 2006]. So sollen Service-Roboter in Zukunft dem Menschen unliebsame Aufgaben abnehmen oder ihn bei seiner Arbeit unterstützen. Das Spektrum reicht hierbei von einem relativ einfachen Fensterputzroboter bis zu humanoiden Robotern, die den Menschen als Partner sehen.

Vorläufer dieser komplexen Systeme sind zum einen die Edutainment-Roboter, die es in unzähligen Varianten bereits heute zu kaufen gibt, und zum anderen Service-Roboter, die für spezielle Aufgaben entwickelt werden. In diesem Kapitel wird eine Übersicht verschiedener Robotersysteme gegeben. Hierbei wird zunächst auf die Klasse der Entertainment-Roboter eingegangen. Anschließend werden existierende Service-Roboter sowie weitere Forschungsplattformen aus dem Bereich der Service-Robotik vorgestellt. Insbesondere wird hierbei auf die Greifer der Roboter eingegangen, da schon die Art des Greifers Aussagen über die Manipulationsfähigkeiten der Roboter erlaubt. Am Ende des Kapitels wird die Fragestellung der Arbeit motiviert.

1.1 Entertainment Roboter

In dem Bereich der Entertainment-Roboter gibt es eine Vielzahl an unterschiedlichen Systemen. Sie reichen von hochkomplexen Robotern bis zu relativ einfachen Bausätzen für Kinder und Jugendliche. Die meisten der Entertainment-Roboter können über verschiedene Programme von dem Benutzer gesteuert werden und sind bereits montiert.

Eines der populärsten Beispiele für einen Entertainment-Roboter ist der Roboterhund AIBO der Firma Sony [Fujita and Kitano 1998] [Sony 2006] (Abb. 1.1), den es mittlerweile in der dritten Generation gibt. Der Roboter ist einem kleinen Hund nachempfunden. Er hat eine Höhe von ca. 26 cm, eine Länge von ca. 32 cm und eine Breite von 18 cm (alle Angaben bezogen auf den AIBO ERS-7). Das Gewicht beträgt 1.65 kg incl. Batterien. Mit einer voll geladenen Batterie hat der AIBO eine Laufzeit von ca. 1.5 Stunden. Die Beine verfügen über je drei Freiheitsgrade. Zusätzlich kann der AIBO das Maul, die Ohren und den Schwanz bewegen. Auch die sensorische Ausstattung ist für einen „Spielzeugroboter“ sehr umfangreich. Der Roboter verfügt über eine integrierte CMOS-Digitalkamera mit einer Auflösung von 416×320 Pixeln. Zudem ist der AIBO mit je einem Beschleunigungs-, Infrarot-, Temperatur- und Vibrationssensor ausgerüstet. „Hören“ kann der AIBO mit einem Stereo-Mikrophon. Er ist mit einem Lautsprecher ausgestattet, über den Töne und Klänge wiedergegeben werden können. Seine „Stimmungen“ soll der Hund zusätzlich mit Hilfe der eingebauten LEDs nach außen sichtbar machen.

Für den AIBO kann man verschiedene Programme kaufen. So gibt es z.B. eine Software, mit welcher der Roboter E-Mails oder andere Textnachrichten vorlesen kann. Des Weiteren gibt es ein Programm,

(a) AIBO ERS-210 (b) AIBO ERS-220 (c) AIBO ERS-7

Abbildung 1.1: Die drei Modellvarianten des Sony AIBO.

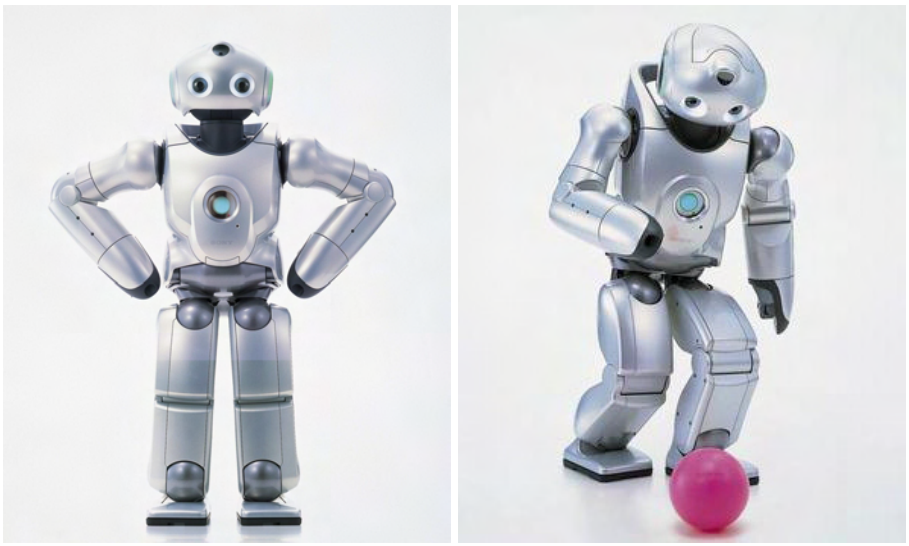


Abbildung 1.2: Der Sony QRIO.

mit dessen Hilfe der AIBO eine Art Persönlichkeitsentwicklung durchlebt. Er entwickelt sich von einem Welpen zum ausgewachsenen Hund. Die Entwicklung ist dabei jedoch nicht immer gleich, sondern wird von äußeren Faktoren beeinflusst. So wirkt sich z.B. die Anzahl der Belohnungen oder das Spielverhalten positiv auf die Entwicklung des AIBO aus, wohingegen Missachtung negativ wirkt.

Der AIBO kann auch über eine von SONY bereitgestellte C/C++ Programmierbibliothek, OPEN-R [Fujita and Kageyama 1997], durch den Benutzer selbst programmiert werden. Die Programmierung ist allerdings nicht für Anfänger geeignet, da es bisher keine Bewegungs- oder Bildverarbeitungsroutinen in OPEN-R gibt. Sony hat allerdings die Produktion und den Vertrieb des AIBO eingestellt. Der AIBO konnte in Deutschland zu einem Preis von ca. 2000 € [Sony 2006] erworben werden. Zusammen mit der Abkündigung des AIBO ist auch die Weiterentwicklung von OPEN-R eingestellt worden.

Die weitere Entwicklung der Entertainment-Roboter wird anhand eines humanoiden Roboters deutlich, der ebenfalls von der Firma Sony produziert wird. Der QRIO (Abb. 1.2) [Ishida et al. 2003] ist ca. 60 cm groß und wiegt 7 kg. Der Roboter hat 38 Freiheitsgrade (DOF: *D*egrees *o*f *F*reedom), die sich wie folgt aufteilen: vier am Hals/Nacken, zwei am Torso, fünf pro Arm, sechs pro Bein und fünf Finger pro Hand. Auch wenn dieser Roboter zur Zeit noch nicht käuflich zu erwerben ist, wird aus der Darstellung auf den

Internetseiten zu dem QRIO [Sony 2007] deutlich, dass SONY ihn nicht nur als reinen Entertainment-Roboter verstehen und vermarkten möchte, sondern als „Freund“. Aussagen über die maximale Traglast der fünffingrigen Hände oder deren Steuerung lassen sich aus den Spezifikationen nicht entnehmen. Daher ist anzunehmen, dass die Hände dem Roboter ein menschenähnliches Aussehen verleihen sollen und keine Funktionalität erfüllen.

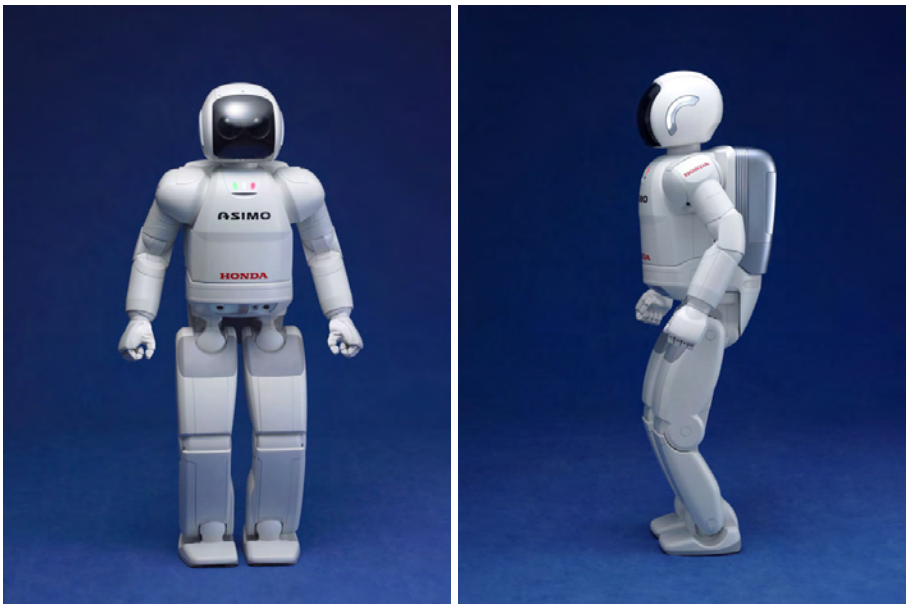


Abbildung 1.3: Der Honda ASIMO [Honda 2007a].

Auch die japanische Firma Honda entwickelt seit 1986 zweibeinige Laufmaschinen und seit 1993 humanoide Roboter. Die neueste Generation des Roboters heißt ASIMO (Abb. 1.3), der Name steht für „*A*dvanced *S*tep in *I*nnovative *M*obility“. Er ist mit 120 cm deutlich größer als der QRIO und wiegt dementsprechend mit 52 kg auch mehr. Honda stellt den Roboter auf der eigenen Web-Seite [Honda 2007a] wie folgt vor:

> „The robot’s size was chosen to allow it to operate freely in the human living space and to make it people-friendly. This size allows the robot to operate light switches and door knobs, and work at tables and work benches. Its eyes are located at the level of an adult’s eyes when the adult is sitting in a chair. A height of 120 cm makes it easy to communicate with. Honda feels that a robot height between 120 cm and that of an adult is ideal for operating in the human living space.“

Diese Aussage macht deutlich, dass der ASIMO nicht nur als Unterhaltungsroboter konzipiert wurde, sondern auch in der Lage sein soll, einfache Arbeiten zu Hause zu erledigen, wie auch aus Bildern 1.4 von der Internetseite [Honda 2007a] hervorgeht. Hierzu wurde der ASIMO auch mit Händen und Fingern, mit einer Traglast von 100 g pro Finger, ausgerüstet. Über die Manipulationsfertigkeiten sind jedoch auch in diesem Fall keine Aussagen zu finden.

Auch die Entwicklungen von anderen Firmen auf dem Gebiet der humanoiden Roboter wie z.B. Fujitsu mit dem HOAP (*H*umanoid for *O*pen *A*rchitecture *P*latform) [Fujitsu 2007] (Abb. 1.5) machen deutlich, dass es nur noch eine Frage der Zeit ist, bis eine Einführung auf dem Massenmarkt erfolgen kann. Der

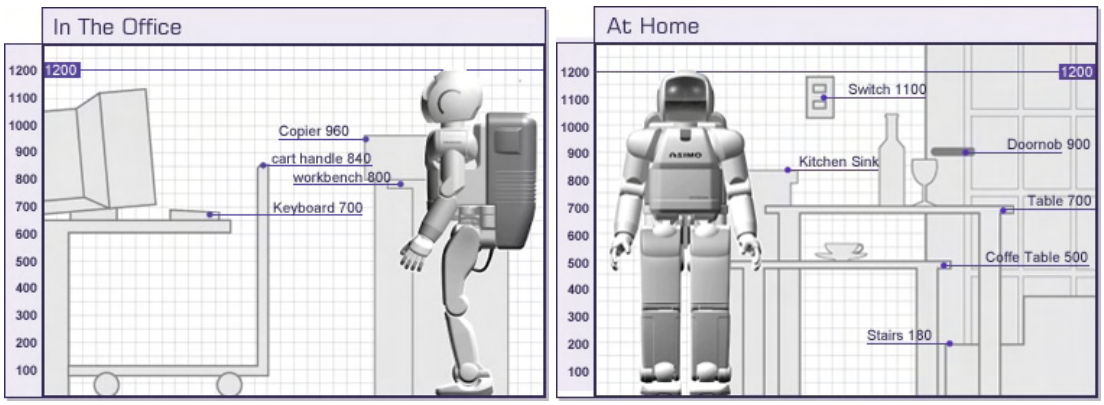


Abbildung 1.4: Szenarien für den Honda ASIMO [Honda 2007b].

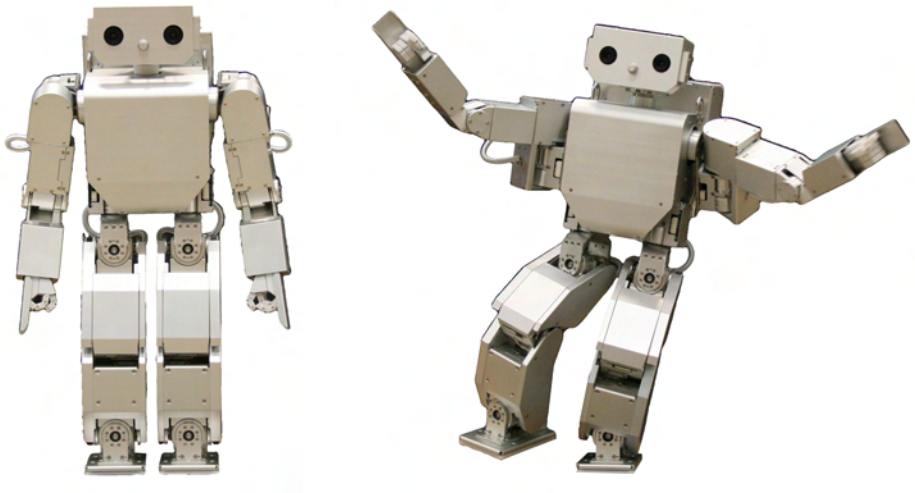


Abbildung 1.5: Der Fujitsu HOAP II.

HOAP ist zugleich auch einer der wenigen humanoiden Roboter, den man derzeit zu einem Preis von ca. 80.000 US$ erwerben kann. Er ist aber konzipiert für den Einsatz in Forschungseinrichtungen und Universitäten, die keine eigene Hardware entwickeln wollen. Die Hände des Roboters sind mit je fünf Fingern ausgestattet. Diese lassen sich entweder komplett schließen oder öffnen. Andere Gelenkstellungen sind nicht möglich. Der Aufbau der Finger ist zudem noch sehr filigran, so dass es nicht möglich ist, mehr als einen Stift zu greifen. In Tabelle 1.1 sind die Spezifikationen der humanoiden Roboter zusammengestellt.

Ein erstes Beispiel für eine kommerzielle Vermarktung eines Roboters hoher Komplexität war der AIBO. Zugegebenermaßen ist der Preis von rund 2.000 € nicht massenmarkttauglich gewesen, was wahrscheinlich auch dazu geführt hat, dass die Produktion eingestellt worden ist. Trotzdem wird deutlich, dass ein komplexer Roboter „relativ“ günstig gefertigt werden kann.

	SONY QRIO	Honda ASIMO	Fujitsu HOAP II
Maße (H × B × T mm)	580 × 270 × 190	1200 × 450 × 440	500 × 245 × 157
Gewicht (incl. Batterie)	7 kg	52 kg	7 kg
CPU	3 × 64 Bit RISC	keine Angaben	≈ PIII 700MHz
Speicher (RAM)	3 × 64 MB	keine Angaben	32 MB RAM, 128 MB Flash
Freiheitsgrade	38	26	25
Tragkraft	keine Angaben	0.5 kg je Hand	keine Angaben
Bildsensor	2 × 110.000 Pixel	keine Angaben	2 × 640 × 480 Pixeln
Greifer & Traglast	5 Finger, keine Angaben zu Traglast	5 Finger, 100 g pro Finger	5 Finger, ca. 20 g je Finger

Tabelle 1.1: Vergleich humanoider Roboter.

1.2 Service-Roboter

Die im vorhergehenden Abschnitt vorgestellten humanoiden Roboter sind zwar von den Herstellern zum Teil als Helfer für den Alltag konzipiert worden, durch ihre geringe Größe und die fehlenden Manipulationsfähigkeiten sind sie jedoch nur Entertainment-Roboter. Der ASIMO von Honda hat zwar Ambitionen, sich als Haushaltshilfe darzustellen, aber die doch relativ geringe Größe von 120 cm und die geringe Traglast von nur 500 g pro Hand stellen ein großes Handicap dar. Zwar dürften 500 g zum Holen eines Glases Wasser ausreichen, schwerere oder hoch gelegene Gegenstände sind für den Roboter aber unerreichbar.

Es gibt aber eine Vielzahl von Service-Robotern die für eine spezielle Aufgabe konzipiert wurden. In der Firmenzentrale der OPEL AG in Berlin „arbeiten“ die beiden Roboter Oskar und Mona (Abb. 1.6) . Sie sind entwickelt worden, um den Besucher durch die Ausstellung zu führen. Sie sind mit einem Touch-Screen ausgestattet, über den Besucher Informationen abrufen kann.

(a) Oskar (b) Mona

Abbildung 1.6: Die Service-Roboter Oskar und Mona.

Ein Roboter für automatisches Probenmanagement in einem Biotechnolgielabor wurde an der Universität Bielefeld entwickelt [Scherer 2004]. Dieser mobile Roboter (Abb. 1.7) kann sich autonom in einem Biotechnologielabor bewegen und die vorhandenen Geräte wie z.B. einen Eisschrank, einen Cedex und eine Zentrifuge bedienen[1]. Das System besteht aus einer mobilen Basis mit einem Roboterarm und einem

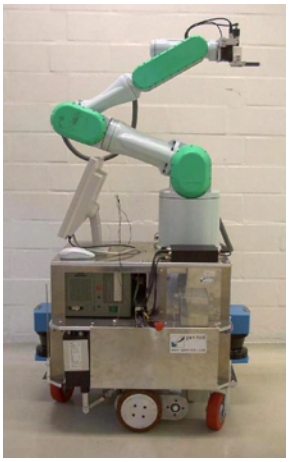


Abbildung 1.7: Roboter für automatisches Probenmanagement in einem Biotechnologielabor.

Zwei-Finger-Parallelgreifer. Dieser wurde speziell angepasst, so dass er in der Lage ist, Probenröhrchen zu greifen. Mit dem System kann über mehrere Tage eine Zellkultivierung autonom durchgeführt werden.

Es gibt aber auch Studien für den Bereich der Haushaltsroboter. Besonders für alte und schwache Menschen werden Roboter entwickelt, die bei den alltäglichen Aufgaben im Haushalt helfen können. Exemplarisch, für diese Roboterart soll hier der Care-o-bot vorgestellt werden, der im Rahmen des *MORPHA*-Projekts (*Kommunikation, Interaktion und Kooperation zwischen Menschen und intelligenten antropomorphen Assistenzsystemen*) [Morpha 2007] entwickelt worden ist. Ziel des Projekts war es, die Interaktions- und Kommunikationsfähigkeiten von Assistenzsystem zu verbessern und sie somit in der Bedienung zu vereinfachen, aber auch Bewegungs- und Griffplanung waren Teil der Forschung.

Der in diesem Projekt entwickelte Care-o-bot II (Abb. 1.8(a)) [Hans et al. 2004, Frauenhofer 2007] ist ein mobiler Roboter, der mit einem Arm, einem einfachen Greifwerkzeug, Kameras sowie einem Monitor ausgestattet ist. Er soll in der Lage sein, alte oder hilfsbedürftige Menschen zu unterstützen. Mögliche Aufgaben, die in diesem Zusammenhang von den Entwicklern genannt werden [Frauenhofer 2005], sind einfache Haushaltsaufgaben wie das Servieren von Getränken oder einfache Reinigungsaufgaben, Mobilitätsunterstützung und Kommunikationsaufgaben.

Ein weiterer Service-Roboter ist HERMES [Bischoff and Graefe 2004] (Abb. 1.8(c)), entwickelt an der Universität der Bundeswehr in München. HERMES ist ebenfalls ein mobiler Roboter mit zwei Armen und einem Stereo-Kamerasystem. Der Roboter ist mit einem Zwei-Finger-Parallelgreifer ausgestattet, um Objekte greifen zu können. Über die Manipulationsfähigkeiten der Roboter Hermes und Care-o-bot gibt es keine Aussagen. Da sie beide nur mit einem bzw. zwei Zwei-Finger-Parallelgreifern ausgestattet sind, ist davon auszugehen, dass die Systeme in der Lage sind, einfache Objekte zu greifen. Es ist aber kein Verfahren zur Generierung von Griffen mit in die Systeme integriert.

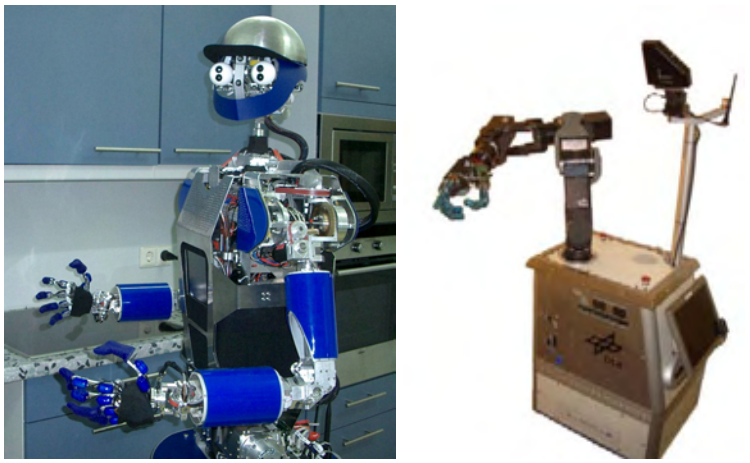
Zwei Robotersysteme die mit künstlichen Händen ausgerüstet sind, sind der Robutler (Abb. 1.9(b)) [Hilleenbrand et al. 2004, Ott et al. 2005] und der humanoide Roboter ARMAR III (Abb. 1.9(a))

[1] Ein Cedex ist ein Gerät, mit die Anzahl von Zellen in einer Probe bestimmt werden kann.

(a) Care-o-bot II. (b) Care-o-bot als Gehhilfe. (c) HERMES.

Abbildung 1.8: Die Service-Roboter Care-o-bot II und HERMES.

[Becher et al. 2004, Asfour et al. 2006a, Asfour et al. 2006b]. Der Robutler ist vom DLR und ARMAR III als Demonstrationsplattform im SFB 588 „Humanoide Roboter - Lernende und kooperierende mulitmodale Roboter“ in Karlsruhe entwickelt worden. Die beiden Roboter sind als interaktive, mobile Service-Roboter konzipiert worden, unterscheiden sich aber grundsätzlich in ihrem Aufbau. ARMAR ist ein humanoider Roboter, mit zwei Armen und Händen. Die Hände des ARMAR sind Eigenentwicklungen mit jeweils vier Fingern, die pneumatisch angetrieben werden (siehe auch Kapitel 2.3). Dieser Roboter ist mit einem System zur Griffgenerierung ausgestattet und in der Lage, Griffe für verschiedene Objekte zu generieren. Die entsprechenden Verfahren werden in Kapitel 6.1 vorgestellt. Der Robutler ist ebenfalls mit einer Vier-Finger-Hand ausgerüstet. Diese Hand ist auch eine Eigenentwicklung und wird über Motoren angetrieben. Im Gegensatz zu ARMAR ist der Robutler aber nicht nicht wie ein humanoider Roboter aufgebaut und verfügt über nur einen Arm. Auch der Robutler ist in der Lage, Griffe für verschiedene Objekte zu generieren. Auch diese Verfahren werden in Kapitel 6.1 präsentiert.



(a) ARMAR-III (b) Robutler

Abbildung 1.9: Die Service-Roboter ARMAR-III und Robutler.

Keiner der Roboter ist als kommerzielles Produkt zu erhalten aber diese Beispiele machen deutlich, welche Visionen für Roboter existieren die heute schon umsetzbar sind.

Aus den gezeigten Ansätzen wird deutlich, dass es nur noch eine Frage der Zeit ist, bis Roboter zu unserem Alltag gehören wie das Mobiltelefon und der Computer. Aber es wird auch deutlich, dass noch ein enormer Forschungsbedarf besteht. Besonders die Anforderung, sich in einer ständig ändernden Umgebung zurechtzufinden, die mit dem Menschen geteilt wird und in der somit nicht die Positionen aller Gegenstände bekannt sind, wie z.B. in einer automatisierten Fertigungstraße, stellt eine große Herausforderung dar. Weitere Schwerpunkte in der Forschung sind die Interaktion mit dem Roboter. So muss es ein Ziel sein, dass der Roboter über „natürliche“ Kommunikationskanäle, also über Sprache und Gestik, seine Befehle entgegennimmt. Es muss auch für technisch unversierte Menschen möglich sein, einen solchen Roboter zu steuern und zu programmieren.

1.3 Motivation

Da sich ein Service-Roboter in einer „natürlichen“ Umgebung[2] bewegen soll, ist es wünschenswert, dass ein Haushaltsroboter nicht nur eine bestimmte Klasse von Objekten, z.B. Gläser handhaben kann, sondern mit den verschiedensten Objekten umgehen kann. Hierbei für jede Aufgabe ein geeignetes Greifwerkzeug mitzuführen, ist aufgrund der Vielzahl der Objekte in einem durchschnittlich ausgestatteten Haushalt nicht möglich. Als möglichst universelles Werkzeug wird oftmals ein Zwei-Finger-Parallelgreifer eingesetzt. Hiermit lassen sich zwar bereits eine große Anzahl von Objekten greifen, die Möglichkeiten sind aber immer noch stark eingeschränkt.

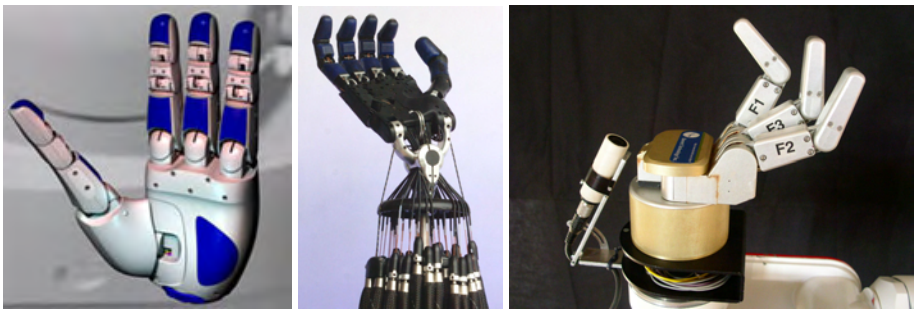


Abbildung 1.10: Roboterhände (von links nach rechts): DLR HIT-Hand, Shadow Hand, BarrettHand

Erst in den letzten drei bis vier Jahren sind mechanische Mehrfingerhände entwickelt worden, die in ihrem Aufbau an eine menschliche Hand angelehnt sind. Allerdings befindet sich der überwiegende Teil dieser Roboterhände noch in der Entwicklung, und die Forschungsinstitute wollen sie (noch) nicht als kommerzielles Produkt vermarkten. Beispiele hierfür sind die DLR II Hand (Abb. 1.10, siehe Kapitel 2.3), die zu Beginn dieser Arbeit noch nicht erhältlich war, oder die Shadow Hand (Abb. 1.10, siehe Kapitel2.3), die inklusive Controller zu groß ist, um sie auf einer mobilen Plattform zu integrieren. Die BarrettHand (Abb. 1.10, siehe Kapitel 2.3) ist die erste Multifingerhand, die als kommerzielles Produkt verfügbar ist und sich durch eine geringe Größe auszeichnet. Sicherlich ist die sensorische Ausstattung der Hand mit nur einem Sensor je Finger nicht sehr umfangreich, und mit drei Fingern sind die Fähigkeiten der Hand auch limitiert, aber sie ist deutlich flexibler einsetzbar als ein Zwei-Finger-Parallelgreifer.

[2]Natürlich im Gegensatz zu einer „künstlichen“ Roboterzelle wie sie z.B. in der Automobilindustrie eingesetzt wird und speziell auf die Bedürfnisse eines Roboters angepasst ist.

Die vorgestellten Service-Roboter sind alle mit unterschiedlichen Manipulationsfertigkeiten ausgestattet. Oskar und Mona verfügen nicht über die Möglichkeit, Objekte in irgendeiner Form aufzunehmen und zu manipulieren. Die Roboter HERMES und Car-o-bot sind in der Lage, einfache Objekte, wie z.B. ein Glas, eine Flasche oder eine Tasse, zu greifen und zu manipulieren. Diese Fertigkeiten reichen aber nicht aus, wenn sich ein Roboter autonom in einem Haushalt bewegen soll und auch komplexere Objekte aufnehmen und bedienen muss. ARMAR und Robutler sind mit ihren komplexen Greifwerkzeugen, die ähnlich der menschlichen Hand aufgebaut sind, in der Lage, nahezu beliebige Objekte zu greifen und zu manipulieren. So wird z.B. auf der Internet-Seite des Robutler in einem Video das Öffnen eines Schraubverschlusses gezeigt [DLR 2005]. Betrachtet man die Möglichkeiten der Systeme (Robutler und ARMAR), Griffe zu generieren (siehe Kapitel 6.1), so wird deutlich, dass sie nur entweder Präzisionsgriffe oder Powergriffe generieren können[3]. Es gibt noch eine Vielzahl weiterer Verfahren für die Berechnung von Griffpunkten für Mehrfingerhände, doch auch diese weisen in den meisten Fällen die bereits genannten Einschränkungen auf. Ein Ansatz, der beide Typen von Griffen generieren kann, ist erforderlich, um ein möglichst flexibles System zum Greifen von Alltagsgegenständen entwickeln zu können.

Auch die Qualitätskriterien, die für die Bewertung eines Griffs benutzt werden, beziehen sich in der Regel nur auf die Kräfte, die bei einem Griff auftreten. Dies ist für einen Laien nur schwer verständlich. Außerdem wird die Funktionalität des Objekts, die Objektsemantik, nicht mit in die Evaluation einbezogen, so dass ein Griff zwar sehr stabil sein kann, aber für eine bestimmte Aufgabe ungeeignet ist. Wenn z.B. eine Tasse nicht von der Seite, sondern von oben gegriffen wird, so kann nichts mehr eingefüllt werden. Diese Eigenschaft eines Griffs wird von den bisher eingesetzten Verfahren zur Griffanalyse nicht berücksichtigt. Dies ist jedoch gerade im Kontext eines Service-Roboters unabdingbar (siehe Kapitel 4).

Im Rahmen dieser Arbeit ist ein mobiler Service-Roboter entwickelt worden, der mit einem Roboterarm und einer Mehrfingerhand ausgestattet ist. Für die Berechnung von Griffen ist ein neues Verfahren auf der Basis von *selbstbewertenden Lernern* (*Reinforcement Lernen*) entwickelt worden. Um die Interaktion und Griffauswahl zu vereinfachen, wurde ein Qualitätsmerkmal auf Basis der Weiterverwendbarkeit eines gegriffenen Objekts entwickelt.

1.4 Struktur der Arbeit

In Kapitel 2 wird der Aufbau des Service-Roboters beschrieben, der für diese Arbeit entwickelt wurde. Es wird im Detail auf die verwendete Hardware eingegangen. Besonders ausführlich wird die BarrettHand betrachtet, die als Greifwerkzeug benutzt wird. Kapitel 3 definiert die Grundlagen zum Greifen von Objekten mit einer Roboterhand. Es werden die Definitionen für die unterschiedlichen Kontaktmodelle sowie für einen Griff gegeben. Des Weiteren wird eine Methode zur Berechnung der Kontaktkräfte präsentiert. Aufbauend auf den Definitionen für Griffe wird in Kapitel 4 eine Übersicht von Qualitätsmerkmalen für die Evaluation von Griffen abgeleitet. In diesem Kapitel wird außerdem das entwickelte Kriterium auf Basis der Weiterverwendbarkeit definiert und evaluiert.

Eine Einführung in die Theorie des selbstbewertenden Lernens wird in Kapitel 5 gegeben. Es werden verschiedene Verfahren aus dem Bereich des Reinforcement Lernens vorgestellt sowie deren Erweiterungen erörtert und bewertet. Kapitel 6 zeigt die Anwendbarkeit des Reinforcement-Lernens in der Domäne des Greifens mit einer Roboterhand. Es wird die Weiterentwicklung eines Lernverfahrens präsentiert, das

[3] Präzisionsgriffe sind Griffe mit nur einem Kontakt pro Finger und keinem Kontakt zu weiteren Teilen der Hand. Powergriffe sind Griffe mit möglichst vielen Kontaktpunkten und mindestens einem Kontakt an der Handfläche.

einen deutlichen Performancegewinn bei der Generierung von Griffen gegenüber einem Standardverfahren des Reinforcement-Lernens aufweist. Mittels einer Simulation wird in Experimenten das Verfahren zur Generierung von Griffen validiert. In 7 wird die Übertragung der in Kapitel 6 generierten Griffe auf den Service-Roboter TASER dargelegt. Es wird ein Überblick über den softwaretechnischen Aufbau des Systems zum Greifen von Alltagsgegenständen gegeben. Um die mittels Simulation generierten Griffe zu validieren, werden diese mit TASER ausgeführt und auf ihre Stabilität überprüft. Auch die in Kapitel 4 definierten Qualitätsanforderungen für einen Griff werden in diesem Zusammenhang validiert.

Das Kapitel 8 gibt eine abschließende Bewertung der eingesetzten Verfahren und einen Ausblick auf weitere Entwicklungsmöglichkeiten des Systems, sowohl in Bezug auf die Hardware als auch auf die Software.

Ein Hinweis zum Lesen

Die Arbeit ist durchgehend in deutscher Sprache verfasst. Da allerdings viele Fachbegriffe aus dem Englischen stammen und es keine adäquate Übersetzung gibt, wurden diese direkt übernommen. Gibt es eine Übersetzung ins Deutsche, so ist der englische Begriff in Klammern ebenfalls angegeben. Fachbegriffe und betonte Begriffe werden bei ihrer Einführung *betont* geschrieben. Das Literaturverzeichnis ist alphabetisch geordnet und beinhaltet keinerlei Wertung. In einigen Fällen sind HTTP-Links als Referenzen angegeben. Der Autor ist sich bewusst, dass dies nicht unproblematisch ist, da sich referenzierte Seiten ändern oder sogar komplett gelöscht werden können. Links werden in diesem Zusammenhang als ergänzende Literatur angesehen und bilden nicht die Grundlage von Aussagen oder Annahmen.

Kapitel 2

Forschungsplattform TASER

In diesem Kapitel wird der Aufbau der Service-Roboter-Plattform TASER (*T*ams *Se*rvice *R*obot) beschrieben (Abb. 2.1). Hierbei wird sowohl auf den hardwaretechnischen Aufbau, als auch auf die Softwarestruktur des Systems und die Anbindung der einzelnen Hardwarekomponenten eingegangen.

Der Roboter TASER dient als Experimentierplattform für diese Arbeit, d.h. die Untersuchungen stehen in direktem Bezug zu der Plattform und werden mit ihr evaluiert. Der Service-Roboter TASER setzt sich im Wesentlichen aus den folgenden Teilen zusammen :

- mobile Plattform
- Roboterarm
- Mehrfingerhand
- Kameras

Zunächst wird der Aufbau der mobilen Basis beschrieben und die Softwarestruktur des Gesamtsystems erläutert. Da die vorliegende Arbeit das Greifen von Gegenständen behandelt, hat die Mobilität der Plattform für die gewählte Problemstellung nur eine geringe Bedeutung. Für die Ausführung der Griffe wird davon ausgegangen, dass sich die Plattform in einer günstigen Position zum Greifen der Objekte befindet und die erforderlichen Bewegungen rein durch Bewegungen des Arms zu realisieren sind. Da die Plattform allerdings die Grundlage des gesamten Hardware- Aufbaus ist, ist sie somit ein wichtiger Bestandteil des Gesamtsystems.

Anschließend wird der Roboterarm zusammen mit der Mehrfingerhand, welche die wichtigsten Systemkomponenten für diese Arbeit darstellen, erörtert. Durch sie wird das System in der Lage sein, Alltagsgegenstände zu greifen. In Abschnitt 2.2 wird der Roboterarm und in Abschnitt 2.3 wird die Mehrfingerhand im Detail beschrieben. In Abschnitt 2.4 wird auf die Integration von Hand und Arm zu einer Einheit eingegangen, um die in der Hand integrierten Kraftsensoren für die Steuerung des Roboters zu nutzen.

Um mit dem Roboter Objekte greifen zu können, muss dieser in der Lage sein, sie zu erkennen und zu lokalisieren. Hierzu werden mehrere Kameras verwendet: eine Kamera, die als Handkamera direkt an der Hand montiert ist, sowie ein Stereokamerasystem, das Übersichtsbilder liefert. Die verwendeten Kameras werden in dem Abschnitt 2.5 beschrieben.

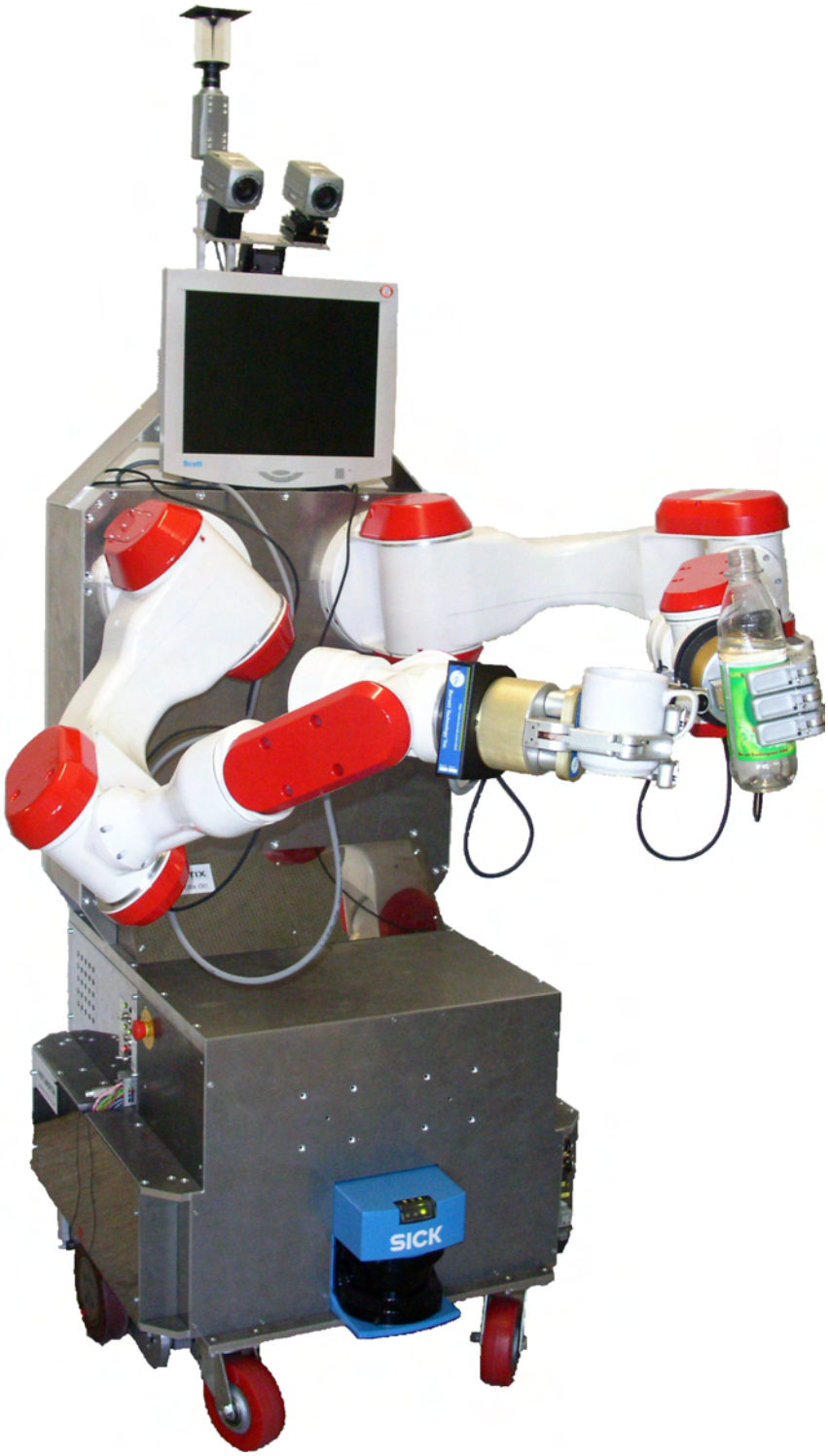


Abbildung 2.1: Der Service-Roboter TASER in der finalen Ausbaustufe mit zwei Armen.

2.1 Mobile Plattform

Die mobile Plattform stellt die Basis des gesamten Systems dar. Sie soll dem Service-Roboter TASER Mobilität verleihen und muss zusätzlich den Platz für alle weiteren Komponenten, die auf dem Roboter benötigt werden, bereitstellen. In diesem Abschnitt wird eine Übersicht über den Aufbau des Gesamtsystems sowie eine Beschreibung der Komponenten der mobilen Plattform im Besonderen gegeben.

Auf den ersten Blick lässt sich die Roboterplattform (Abb. 2.1) in drei Sektionen unterteilen:

1. die mobile Basis mit dem Turmaufbau
2. zwei Roboterarme mit je einer Dreifingerhand
3. der Aufbau mit dem Stereokameraaufbau auf der Schwenk-Neigeeinheit und dem Omnidirektionalen-Sichtsystem.

Um dem System eine menschenähnliche Operabilität zu verleihen, ist es für die Montage von zwei Roboterarmen ausgelegt. Für diese Arbeit ist das System jedoch mit nur einem Roboterarm ausgestattet. Für die Integration der zwei Arme auf dem System ist der turmförmige Aufbau auf der Basisplattform erforderlich. Um die Übersichtskameras in einer Position mit einem möglichst guten Blick vor den Roboter montieren zu können, wurde noch ein zusätzlicher Aufbau auf den Turm montiert.

2.1.1 Aufbau der mobilen Basis

Die mobile Plattform (Abb. 2.2) basiert auf der Plattform MP-L655 von Neobotix [Neobotix 2007], einer Abteilung der GPS GmbH.

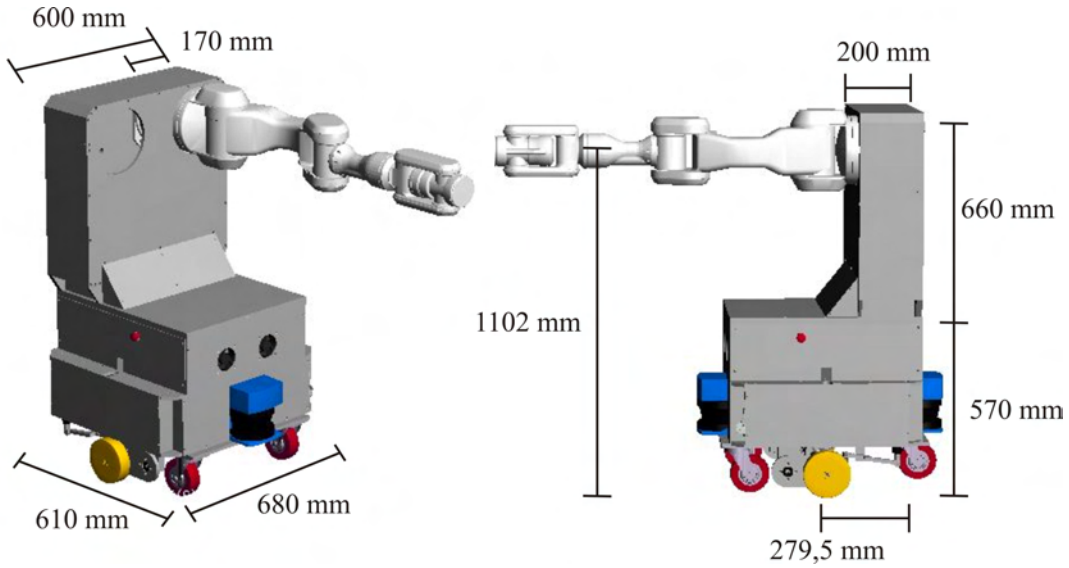


Abbildung 2.2: Die mobile Basis mit Turmaufbau, Roboteram und Bemaßung.

Die Einheit besteht aus einem Aluminiumrahmen mit einer Vielzahl von Fächern, in denen die Leistungselektronik, der Steuerungs-PC und die Kontroller der übrigen Komponenten Platz finden. An der

Bodenplatte sind die Motoren und Stützräder montiert. In dem gesamten Fach darüber befinden sich acht Blei-Gel-Akkus, in dem oberen Fach ist die Leistungselektronik für den Roboterarm sowie ein Standard-Industrie-PC untergebracht. Die Maße des Systems können Abbildung 2.2 entnommen werden. Der Arm ist in einer Höhe angebracht, die ungefähr der Schulterhöhe eines Menschen entspricht. Die Maßangaben für die Position des Arms beziehen sich auf den Ursprung des Koordinatensystems der mobilen Plattform. Dieser liegt in der Höhe Null, im Zentrum des Roboters. Das Gesamtgewicht des TASER beträgt ca. 200 kg. Weitere Details der mobilen Basis sind in Anhang A.1 beschrieben.

Die mobile Einheit ist mit einem Differentialantrieb ausgestattet. Zur Stabilisierung des Systems sind hinten ein und vorne zwei Schwingräder montiert. Diese Anordnung des Antriebs vereinfacht die Kinematik, da die Plattform auf einem Punkt rotieren kann und somit eine Richtungsänderung auch auf engem Raum möglich ist. Der Nachteil bei dieser Konstruktion besteht darin, dass durch die fünf Kontaktpunkte eine Fahrt durch unebenes Gelände nicht gewährleistet werden kann, da die Antriebsräder evtl. den Kontakt zum Boden verlieren können. Aber da sich der Roboter als Service-Einheit in einer Büroumgebung bewegen soll, kann dies ignoriert werden.

Die Hauptspannung des Roboter-Systems beträgt 48 V. Diese wird von acht Blei-Gel-Akkumulatoren bereitgestellt, die eine Gesamtkapazität von 80 Ah aufweisen. Dies ermöglicht dem System eine autonome Laufzeit von mehr als sieben Stunden.

Durch die Versorgung mit 48 V Spannung und den Einsatz eines besonderen PC-Netzteils ist es möglich, einen PC in einem Industriegehäuse, aufgebaut aus Standardkomponenten, zu verwenden. Dies ermöglicht einen einfachen und kostengünstigen Systemaufbau. Die einzelnen Komponenten des PC sind in Anhang A.1 beschrieben.

2.1.2 Software-Architektur

Im Folgenden soll kurz die Software Architektur der Roboterplattform beschrieben und ein kurzer Überblick über das Zusammenspiel der einzelnen Komponenten gegeben werden. Die Software für die Steuerung der BarrettHand wird in Abschnitt 2.3 und die des Roboterarms in Abschnitt 2.2 näher erläutert.

Die Software für die mobile Einheit wurde zu sehr großen Teilen von Torsten Scherer an der Universität Bielefeld im Rahmen seiner Promotion entwickelt [Scherer 2004]. Sie wurde jedoch an den hier verwendeten Aufbau angepasst. Die Lokalisation der mobilen Einheit wird mittels der zwei SICK-Laserscanner realisiert. Hierzu sind statische Landmarken in den Räumen, in denen sich TASER bewegt, angebracht worden. Nach einer einmaligen Initialisierung der Position werden die Landmarken mittels eines Kalmann-Filters [Schneider and Westhoff 2002] verfolgt und somit die Position ermittelt. Die Genauigkeit liegt hierbei bei ca 1 cm Positions- und bei 1° Orientierungsabweichung. Für weitere Details der Lokalisierung siehe [Scherer 2004] und [Schneider and Westhoff 2002].

Die Steuerung der mobilen Basis erfolgt über eine in C/C++ geschriebene Software. Eine Kollisionserkennung stoppt den Roboter automatisch, falls sich ein Hindernis in der Bahn des Roboters befindet. Die Software bietet die Möglichkeit, rudimentäre Bewegungsbefehle, nämlich die Änderung der Orientierung auf einen bestimmten Winkel sowie die Geradeausfahrt mit einer festgelegten Geschwindigkeit, auszuführen. Auch eine Pfadplanung oder relative Bewegung sind ebenfalls in die Implementierung integriert worden.

Diese Software-Teile sind in weiterführenden Arbeiten von Daniel Westhoff und Hagen Stanek über die genRob GmbH erweitert worden [Westhoff et al. 2004b], [Westhoff et al. 2004a] und

[Westhoff et al. 2006]. Eine ausführliche Beschreibung des genRob Projekts würde an dieser Stelle den Rahmen der Arbeit sprengen, daher soll hier nur ein sehr kurzer Einblick gegeben werden.

Die Roblet-Technologie beinhaltet als Grundgedanken die Idee, dass ein Softwareentwickler für Robotersoftware nicht immer direkt an einem Roboter entwickelt. Gewöhnlich geschieht dies an einem Arbeitsplatzrechner, von dort aus werden dann die Programme, meistens von Hand, auf den Roboter übertragen und dort manuell gestartet. Die Roblet-Technologie hat das Ziel, diese Zwischenschritte per Software auszuführen, ohne dass sich der Entwickler um die Programmierung der Netzwerkmechanismen kümmern muss. Für den Benutzer wird das Netzwerk also transparent. Daher ist ein auf Roblet-Technologie basierendes System in der Regel ein verteiltes System. Die Funktionsweise des Systems ist vergleichbar mit der von JAVA-Applets, die in Internetseiten verwendet werden, um Programme auf dem Rechner des Benutzers ausführen zu können. Bei JAVA-Applets wird ebenfalls das JAVA-Programm von einem Server auf den Rechner des Anwenders übertragen und dort in einer abgesicherten Umgebung ausgeführt. Dies ist bei Roblets prinzipiell ähnlich. Jedoch gibt es bei der Roblet Technologie im wesentlichen zwei Typen von Software: Zum einen Client-Anwendungen, die z.B. Schnittstellen für den Benutzer bereitstellen, und zum anderen Roblet-Server. Von den Client-Anwendungen werden Programme versandt, Roblets genannt, die auf den Roblet-Servern ausgeführt werden. Die Roblet-Server stellen Schnittstellen, die als Einheiten bezeichnet werden, bereit. Durch sie wird der Zugriff auf die vom Roblet-Server gekapselte Hardware realisiert. Durch diese Methode lassen sich einfach und effizient verteilte Systeme entwickeln und warten.

Da in der Roblet-Technologie Programme über ein Netzwerk versendet werden, lassen sich mit der Technologie keine echtzeitkritischen Anwendungen realisieren. Somit können Roblets nicht eingesetzt werden, um z.B. die Regelung für die Motoren der mobilen Basis zu realisieren. Der entsprechende Regler muss direkt auf dem System ausgeführt werden und „High-Level-Funktionen“ bereitstellen, die mittels Roblets ausgeführt werden. Diese „High-Level-Funktionen“ können z.B. in einer C/C++ Bibliothek gekapselt werden, die dann über das Java Native Interface (JNI) eingebunden wird.

Mittels der Roblet-Technologie wurde ein System auf dem Service-Roboter TASER installiert, das die einzelnen Hardwarekomponenten kapselt und den Benutzern zugänglich macht. Eine entsprechende Übersicht ist in Abbildung 2.3 gegeben. Die Hardware ist in der Abbildung gelb unterlegt dargestellt, die entsprechenden C/C++ Bibliotheken blau und die Roblet-Server grün. Für den Roboterarm wurde z.B. eine Funktion entwickelt, die die Bedienung eines Lichtschalters erlaubt. Diese Funktion ist in der ZRobot-Bibliothek implementiert und wird als nativer C++-Code von dem entsprechenden Roblet-Server ausgeführt. Weitere Details zu der Software Architektur sind in [Baier et al. 2006] zu finden.

2.2 Roboterarm

Hat die mobile Plattform die Aufgabe, den Roboter in die Nähe des zu greifenden Objekts zu bringen, so dient der Arm letztlich dazu, die Hand so zu positionieren, dass ein Griff ausgeführt werden kann. Somit ist der Arm einer der elementaren Bestandteile für diese Arbeit. Als Roboterarm wurde ein MHI-PA10-6C ausgewählt.

MHI-PA10-6C

Der Roboter-Arm ist ein Arm der Firma Mitsubishi Heavy Industries (MHI) [MHI 2007] vom Typ PA10-6C, wie in Abbildung 2.4 dargestellt. Der Arm hat sechs Freiheitsgrade und eine kinematische Länge von

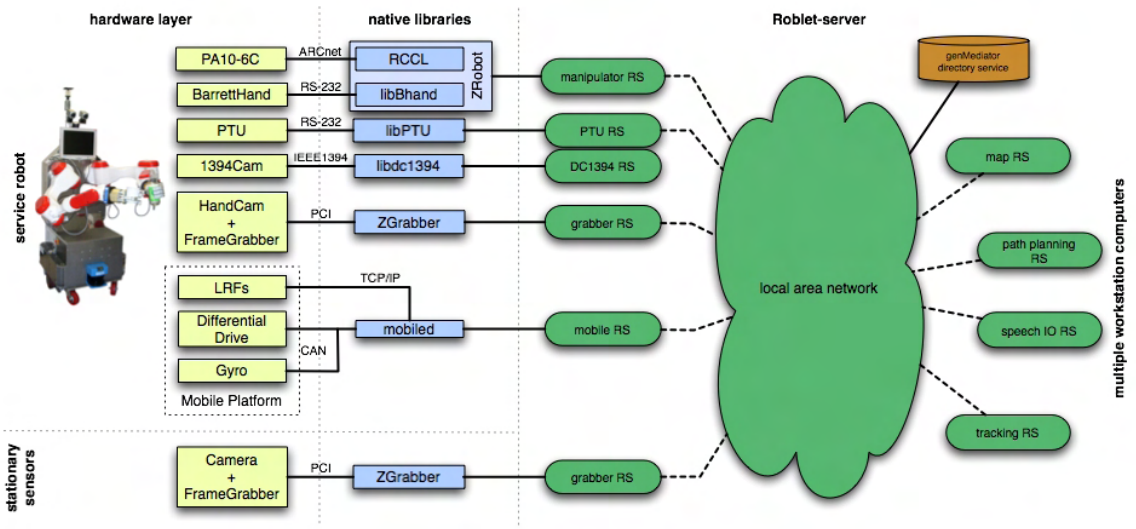


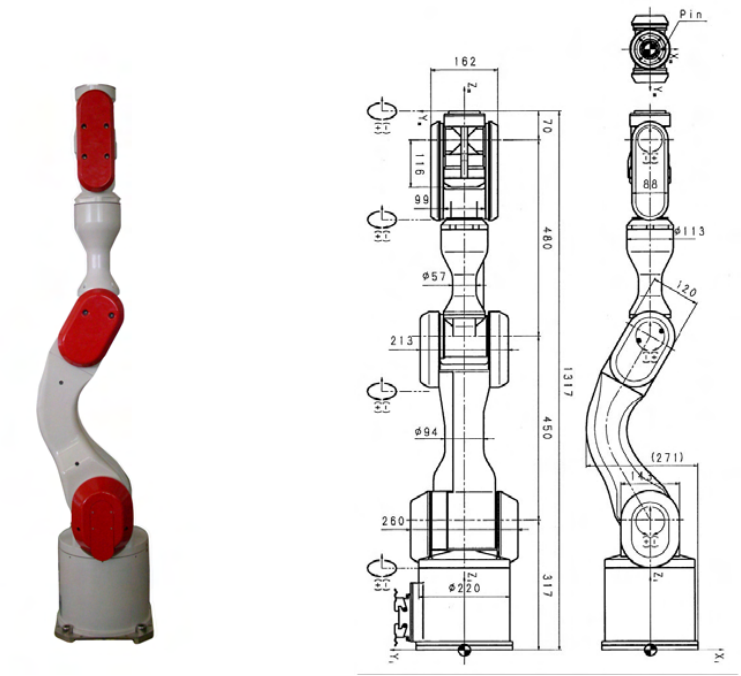
Abbildung 2.3: Die Software-Architektur des Service-Roboters TASER. Die Hardware ist in der Abbildung gelb unterlegt dargestellt, die entsprechenden C/C++ Bibliotheken blau und die Roblet-Server grün.

1317 mm. Sein Gewicht beträgt 38 kg und die maximale Traglast ist mit 10 kg angegeben. Da der Arm mit einer Spannung von 48 V anstatt 100 V betrieben wird, reduziert sich die tatsächliche maximale Traglast auf weniger als 10 kg.

Durch die Länge von 1317 mm und die sechs Freiheitsgrade hat der Arm einen Aufbau, der mit einem menschlichen Arm vergleichbar ist. Der Aufbau des Arms ist in Bild 2.4(b) dargestellt. Weitere technische Details sind in Anhang A.2 zu finden.

Ein weiterer großer Vorteil der PA10-Serie für den Aufbau eines Service-Roboters besteht darin, dass der Kontroller für den PA10 sehr klein ist und somit leicht in einen mobilen Roboter integriert werden kann. Bei Auslieferung ist die Leistungselektronik des Arms zusammen mit dem Kontrollerboard in einem 240 mm $\times$ 300 mm $\times$ 400 mm großen Gehäuse montiert (Abb. 2.5). Die Teile lassen sich getrennt in der mobilen Einheit unterbringen.

Der Kontroller des PA10 verfügt über eine ArcNet-Schnittstelle und kann somit von einem Standard-PC mit ArcNet-Interface gesteuert werden. Als Basis für die Steuerungssoftware wird RCCL (Robot-Control-C-Language) eingesetzt [LLoyd and Hayward 1992]. Sie wurde ursprünglich von John Lloyd und Vincent Hayward an der Universität British Columbia, Kanada, entwickelt und ist als freie Software erhältlich. Torsten Scherer hat RCCL um die Steuerung für den PA10-7C und PA10-6C ergänzt (siehe [Scherer 2004] für Details). Es ist auch möglich, den Arm über ein *Mitsubishi-Kontroller-Board*, das *MHI- Motion-Control-Board*, zu steuern. Wie Torsten Scherer in seiner Arbeit feststellt, wäre diese Variante jedoch mit einigen Nachteilen verbunden. So bietet das MHI-Motion-Control-Board nur unzureichende Möglichkeiten, die Trajektorie des Roboters in Echtzeit zu manipulieren. Dies ist jedoch eine unabdingbare Notwendigkeit, wenn Bewegungen anhand von Kraftmessungen gesteuert oder abgebrochen werden sollen. Ein weiterer Nachteil des MHI-Motion-Control-Boards besteht in seiner Ungenauigkeit. So hat Torsten Scherer in seiner Arbeit [Scherer 2004] eine mittlere Abweichung um 18,333 mm und eine maximale Abweichung von 46,041 mm zu der vorgegebenen, spiralförmigen Trajektorie ermit-



(a) MHI-PA10-6C (b) Maße des MHI-PA10-6C.

Abbildung 2.4: Der MHI PA10-6C.

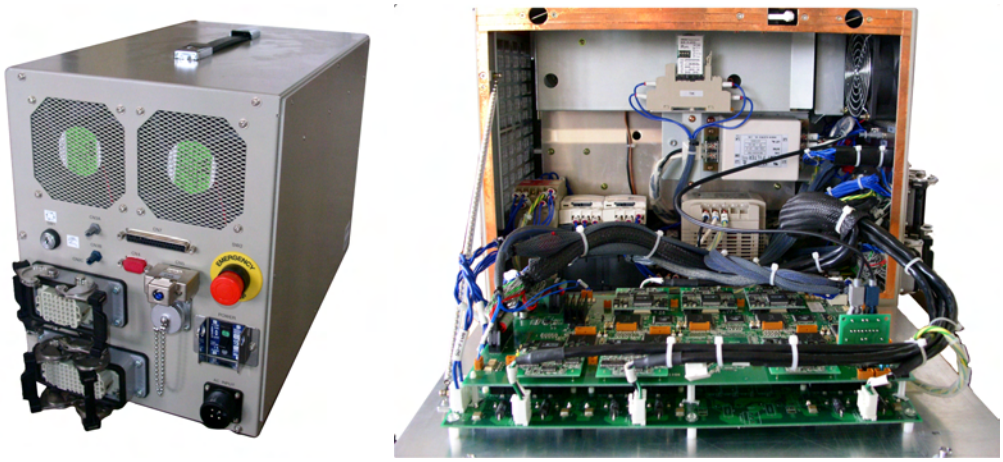


Abbildung 2.5: Gehäuse des MHI-PA10-6C-Kontroller und das geöffnete Gehäuse mit Kontrollerplatine und Leistungselektronik.

telt. Bei dem Einsatz von RCCL für die Steuerung des Roboters liegt der mittlere Fehler bei 2,376 mm und der maximale Fehler bei 4,949 mm. Somit ist RCCL im Durchschnitt um den Faktor 3,7 besser,

wenn eine spiralförmige Bewegung ausgeführt wird[1]. Basierend auf RCCL ist an der Universität Bielefeld von Markus C. Ferch und Yorck O. von Collani die Klassenbibliothek RCCL++ entwickelt worden. RCCL++ ist eine Erweiterung von RCCL um C++ Klassen. So sind in RCCL++ verschiedene Robotertypen definiert, die komplexe Bewegungen ausführen können. Hiermit ist es möglich, mittels eines Kraft-Drehmomentsensors gefügige Bewegungen – das sind Bewegungen bei denen der Roboteram aufgrund von Krafteinwirkuung nachgibt – auszuführen. In dieser Arbeit wurde kein Kraft-Drehmomentsensor verwendet, sondern die in der Multifingerhand integrierten Kraftsensoren zur Steuerung des Arms eingesetzt (siehe 2.3.2 und 2.4).

2.3 Mehrfingerhand

Als Greifwerkzeug ist eine Mehrfingerhand, die BarrettHand (Abb. 2.6), an den Roboterarm montiert [Townsend 2000].

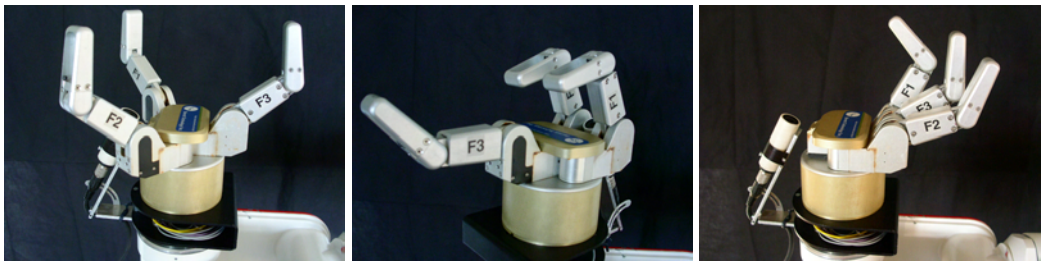


Abbildung 2.6: Die BarrettHand mit Handkamera. Der Spreizwinkel der Finger beträgt (von links nach rechts) 60°, 0° und 180°.

Mit diesem Mehrfingergreifer ist das System in der Lage, Gegenstände zu greifen und zu manipulieren. Durch die Wahl einer Mehrfingerhand soll erreicht werden, dass sich der Roboter in einer „menschlichen“ Umgebung bewegen und in dieser mit alltäglichen Gegenständen operieren kann. Dies wäre mit einem einfachen Zwei-Finger-Parallelgreifer, wie er in der Robotik oft benutzt wird, nicht in dem Umfang möglich, wie es mit einer künstlichen Hand möglich ist [Bicchi 2000, Mishra 2000].

Die BarrettHand ist die einzige Mehrfingerhand, die zum Zeitpunkt des Systemaufbaus als kommerzielles Produkt verfügbar war. Es gibt noch weitere Manipulatoren, die der menschlichen Hand nachempfunden sind, von denen eine Auswahl kurz vorgestellt wird (Abb. 2.7). Die DLR/HIT Hand [Butterfass et al. 2004, DLR 2007] die eine Weiterentwicklung der DLR Hand II in Zusammenarbeit mit dem Harbin Institute of Technology darstellt. Die Shadow Hand [Shadow 2003], die mittlerweile als kommerzielles Produkt zu erwerben ist, aber mit einer Unterarmlänge von 434 mm nur schwer auf einem mobilen Roboter integriert werden kann. Des Weiteren die ROBONAUT Hand [Lovchik and Diftler 1999, Lovchik et al. 1999, NASA 2007], entwickelt von der NASA, oder die Hand, die für den SFB-588: „Humanoide Roboter - Lernende und kooperierende multimodale Roboter“ in Karlsruhe entwickelt wird (im Folgenden SFB-Hand genannt) [Kargov et al. 2006, SFB-588 2007], die für den Roboter ARMAR (siehe Kapitel 1.2) entwickelt entwickelt wurde.

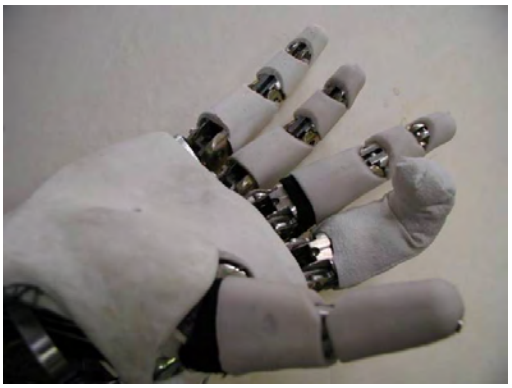
Die Antriebsarten der Hände gliedern sich dabei in zwei Klassen. Zum einen die Klasse der künstlichen Muskeln, die wie ein menschlicher Muskel funktionieren und in der Regel mit Pressluft betrieben

[1] Diese Ergebnisse gelten für das ISA Motion Control Board das T. Scherer verwendet hat. Tests für das neue MHI PCI Motion Control Board müssen noch zeigen, ob der Fehler immer noch in diesem Maße auftritt.

(a) DLR/HIT Hand

(b) Shadow Hand

(c) SFB-Hand



(d) Robonaut Hand

Abbildung 2.7: Verschiedene Roboterhände. Von links nach rechts: die DLR/Hit Hand, die Shadow Hand, die SFB-Hand, die Robonaut Hand.

werden, wie bei der Shadow Hand oder auch der SFB-Hand (die Muskeln der SFB-Hand sind zwar mit einer Flüssigkeit gefüllt, betrieben werden sie aber auch mit Pressluft). Die zweite Klasse bilden die elektrisch betriebenen Hände, bei denen die Finger durch Motoren angetrieben werden. Die BarrettHand gehört zu der zweiten Klasse und soll im Folgenden detailliert vorgestellt werden, da durch sie gewisse Einschränkungen für den praktischen Teil dieser Arbeit berücksichtigt werden müssen.

2.3.1 Aufbau

Die BarrettHand ist eine Drei-Finger-Hand. Die maximale Spannweite der Finger beträgt 317,5 mm und die Breite eines Fingers 23 mm (Abb. 2.8). Das Gewicht beträgt 1.18 kg. Die Bewegungsradien der Gelenke der Finger sind in 2.9 abgebildet. Ihre Positionen werden mittels optischer Inkremental-Encoder gemessen.

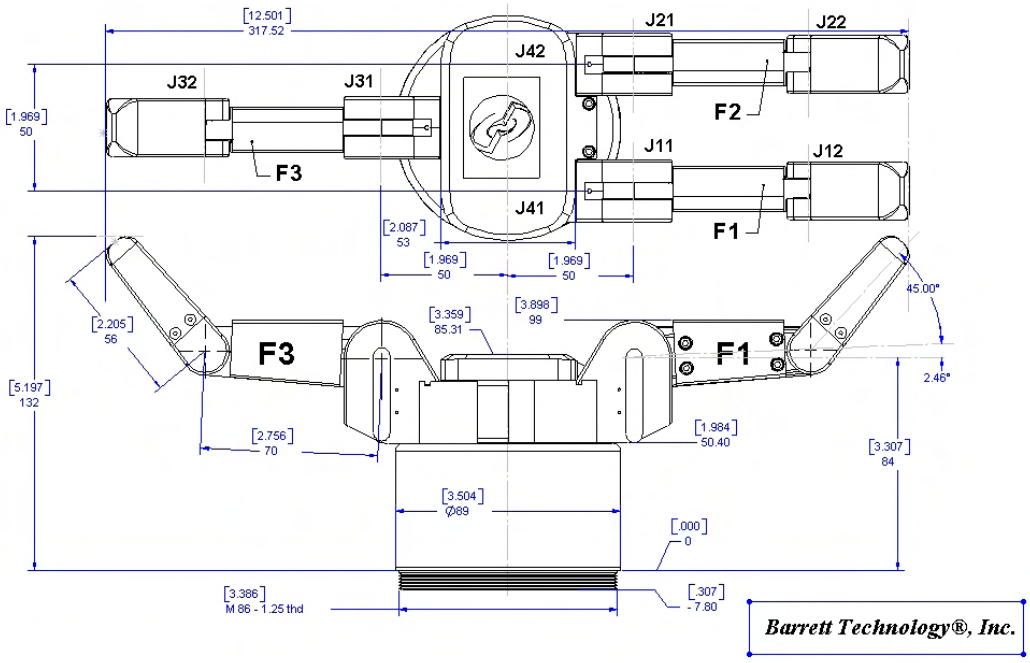


Abbildung 2.8: Dimensionen der BarrettHand (alle Angaben in mm [inch]; 1 inch = 25,4 mm) [Barrett 2007].

Die Hand wird von insgesamt vier bürstenlosen Gleichstrom-Servomotoren angetrieben. Sie sind sehr klein und leicht, bieten aber im Verhältnis dazu eine hohe Leistung. Ein weiterer Vorteil besteht darin, dass diese Motoren sehr wartungsarm sind. Die Anbindung an den Steuerrechner erfolgt über eine serielle RS232-Schnittstelle.

Jeder Finger wird über einen eigenen Motor bewegt. Die Kraftübertragung erfolgt über ein Drahtseil, das in der Basis des Fingers und im oberen Drittel des zweiten Teils jedes Fingers verankert ist. Zusätzlich treibt ein Motor über ein Getriebe die Spreiz-Gelenke (J42 und J41, Abb. 2.8) an. Dieser Aufbau führt dazu, dass sich die Gelenke eins und zwei (Jx1 und Jx2) jedes Fingers nicht unabhängig voneinander regeln lassen. Diese Funktionsweise ist der einer menschlichen Hand nachempfunden. Falls ein Kontakt

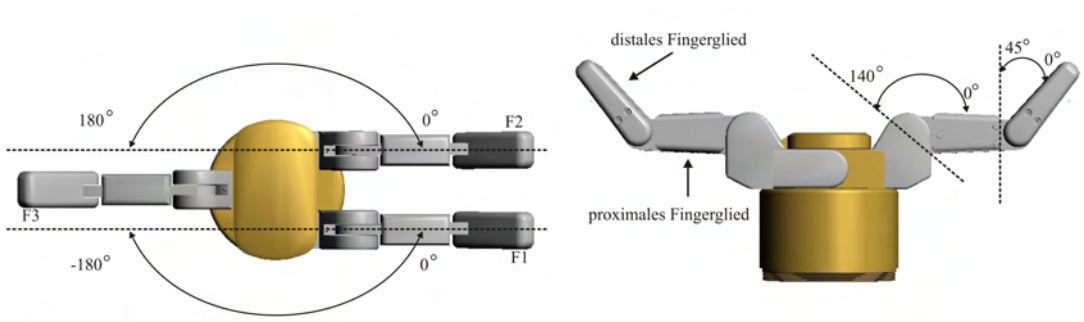


Abbildung 2.9: Gelenkwinkel der BarrettHand. „Proxmial“ (proximus = der Nächste) bedeutet zum Körper hin gelegen und „distal“ (distare = sich entfernen) vom Körper entfernt gelegen.

am proximalen Teil des Fingers eine Bewegung des gesamten Fingers verhindert, wird in der BarrettHand der so genannten *TorqueSwitch*™Mechanismus aktiv. Die Funktionsweise dieses Mechanismus ist in Abbildung 2.10 dargestellt. Diese Mechanik bewirkt, dass der distale Teil des Fingers noch „umklappen “kann, wenn der proximale Teil durch einen Kontakt bereits blockiert ist.

Die maximale Traglast pro Finger wird von Barrett Technologies mit maximal 2 kg je Finger angegeben [Barrett Manual 2002]. Somit kann die Hand Objekte mit einem maximalen Gewicht von bis zu 6 kg, bei gleichmäßiger Gewichtsverteilung auf alle drei Finger, tragen. Da die 2 kg ungefähr einer Kraft von 20 N pro Finger entsprechen, ist dies auch der maximal zulässige Kraftwert, den ein Kontakt pro Finger aufbringen kann. Mechanisch ist durch den TorqueSwitch-Mechanismus eine menschenähnliche Operabilität der Finger gegeben.

2.3.2 Kraftsensoren

Die Sensoren für die Kraftmessung sind im proximalen Teil der Finger montiert. Ein Sensor besteht aus einem Dehnungsmesstreifen, auf den die auftretenden Kräfte über den Seilzug, der den jeweiligen Finger antreibt, übertragen werden. Der Aufbau ist in Abbildung 2.11 dargestellt. Die Kraftsensoren haben eine Auflösung von acht Bit. Somit können Werte von 0 bis 255 gemessen werden. Das Rauschen eines Sensors beträgt ± 2 Werte.

Da der Sensor nicht direkt Kraftwerte in der Einheit Newton liefert, muss hier für die spätere Verwendung eine Umrechnung erfolgen. Hierfür wurde ein Kalibrationsverfahren entwickelt, mit dem sich die Faktoren für die Umrechnung der gemessen Sensorwerte in Kraftwerte der Einheit Newton berechnen lassen.

Um die erforderlichen Werte für die Umrechnung zu bestimmen, wurde mit einem Finger eine Kraft auf eine Waage ausgeübt. Hierdurch lässt sich die aufgebrachte Kraft über die Newtonsche Gleichung $F = m \cdot a$ errechnen. Hierbei ist F die resultierende Kraft, m der Wert, den die Waage misst und a die aufgebrachte Beschleunigung, die hier die Erdbeschleunigung mit einem Wert von 9.813 ms^2 ist. In Abbildung 2.12 sind die berechneten Kraftwerte in N gegen die dazu gemessenen Sensorwerte aufgetragen. Wie aus der Grafik zu sehen ist, sind die gemessenen Kraftwerte weitgehend linear zu den Sensorwerten. Somit entspricht eine Einheit des Sensorwerts ungefähr einer Kraft von 0.145 N. Wie in Abbildung 2.13

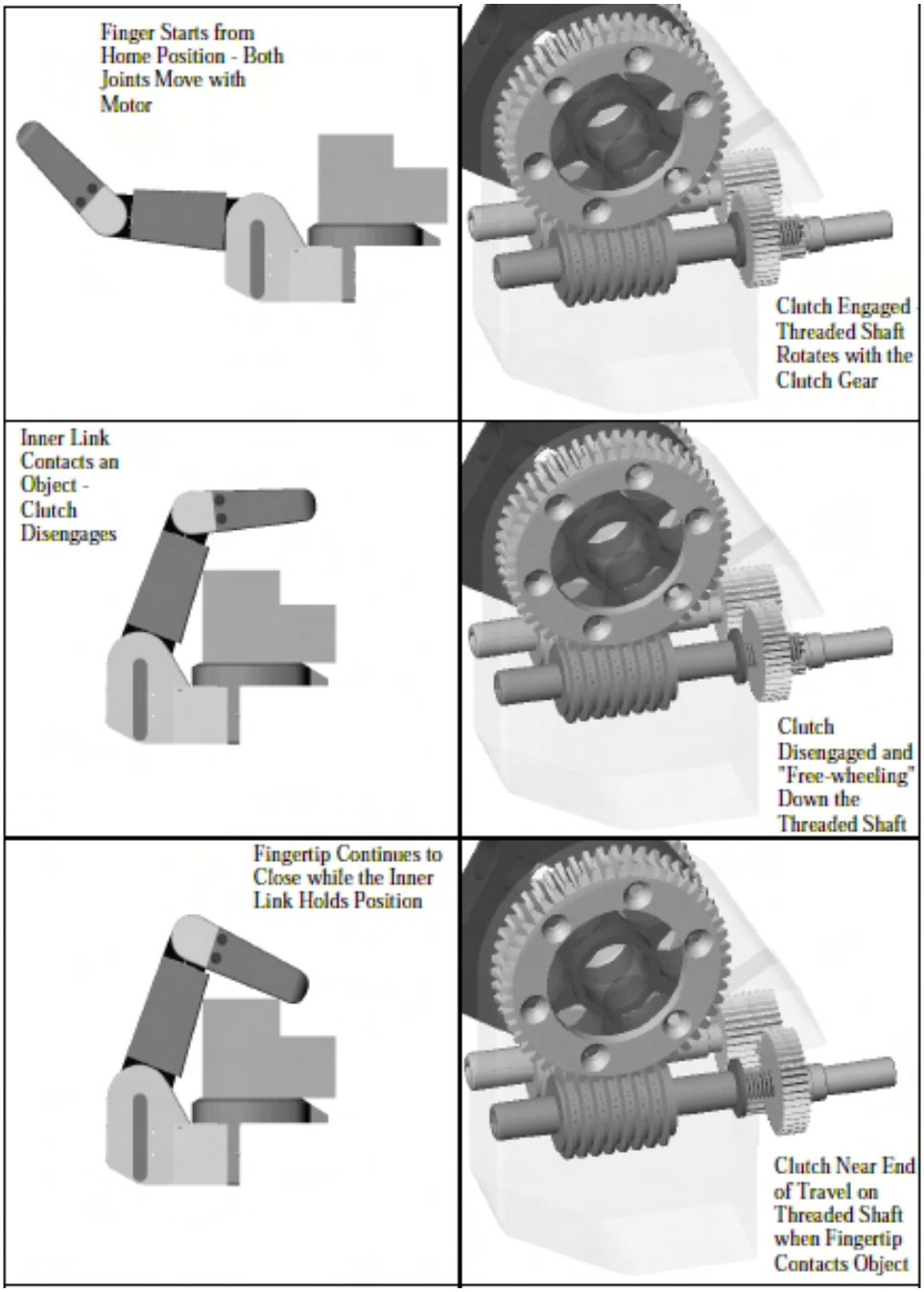


Abbildung 2.10: Funktionsweise des TorqueSwitch Mechanismus der BarrettHand [Barrett Manual 2002]. Erste Zeile: Wenn beide Fingerglieder frei laufen, treibt eine Welle den Finger über zwei Zahnräder an . Zweite Zeile: Wird der proximale Teil eines Fingers blockiert, läuft sich eines der Antriebszahnränder frei. Dritte Zeile: Während des Freilaufens bewegt sich das distale Fingerglied bis zu seinem mechanischem Limit weiter.

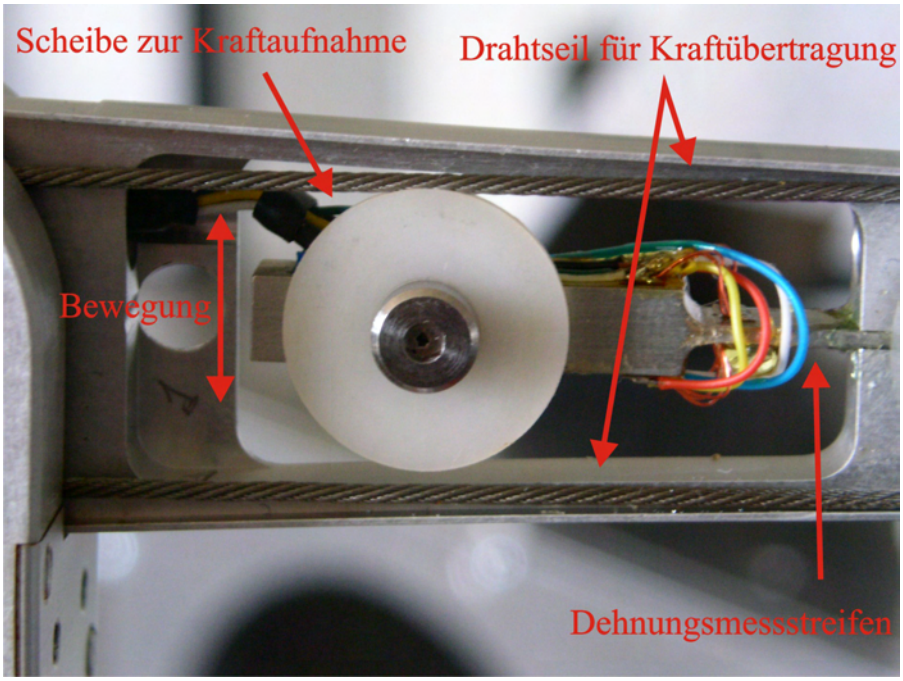


Abbildung 2.11: Kraft-Sensor an der BarrettHand.

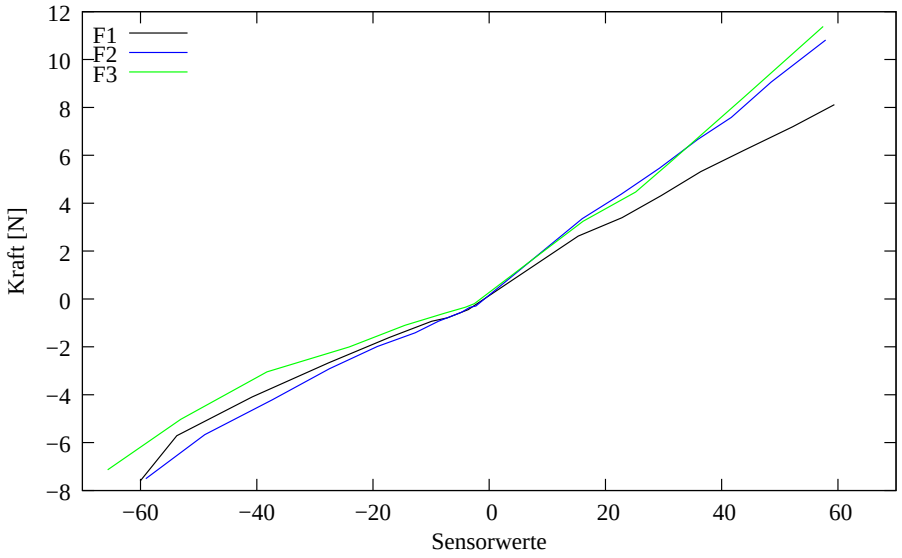


Abbildung 2.12: Die Werte der Kraftkalibrierung der Finger. Die gemessenen Sensorwerte sind (nahezu) linear zu berechneten Kraftwerten.

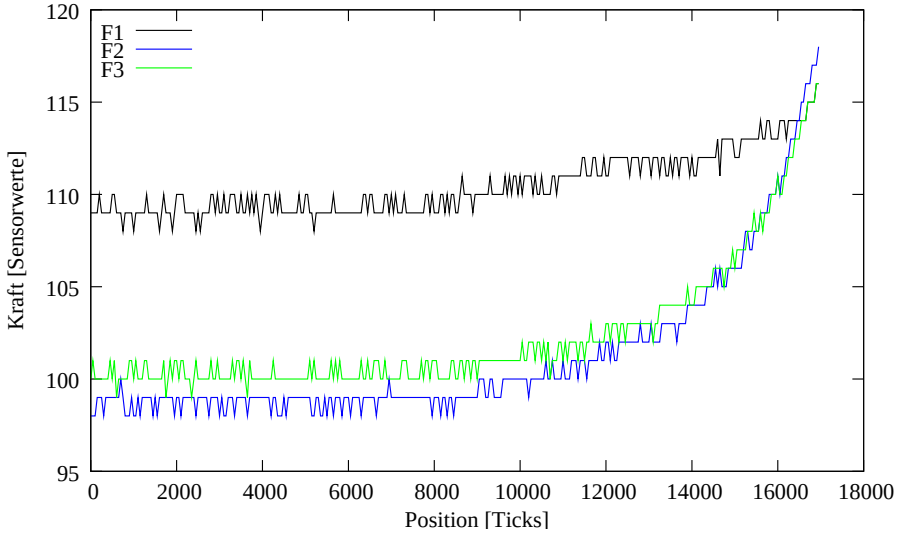


Abbildung 2.13: Die Werte der Kraftsensoren in Bezug zur Position der Finger F1, F2, F3. Bis zu einer Position von ca. 10.000 Motor-Ticks ist die Bewegung unabhängig von den gemessenen Kraftwerten.

zu sehen, ist der gemessene Kraftwert des Sensors unabhängig von der Position des Fingers bis zu einer Position von ca. 10000 Motorticks bzw. einem Winkel von $82.46°$ für das proximale Fingerglied. Somit können die Kräfte nur in einem Bereich von 0 bis 10000 Ticks unabhängig von der Position des Fingers gemessen werden.

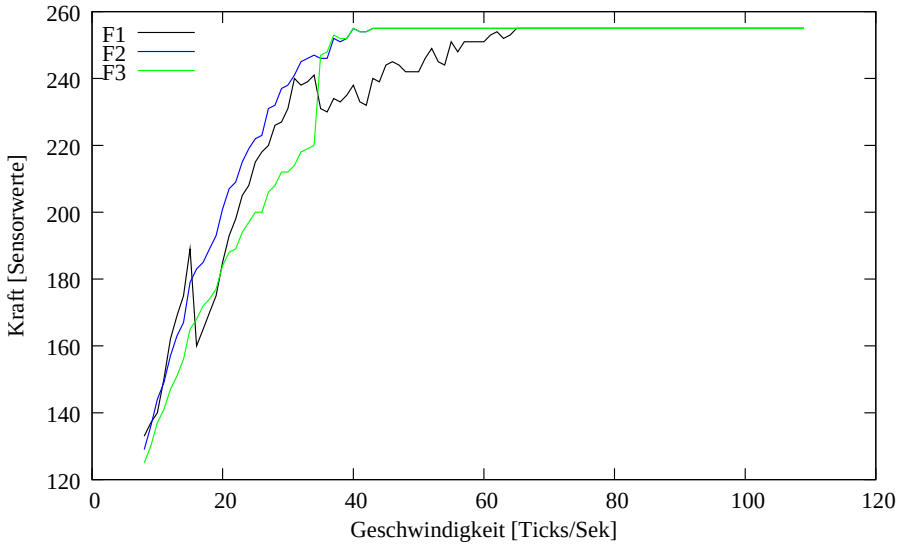


Abbildung 2.14: Die gemessen Kraftwerte aufgetragen gegen die Fingergeschwindigkeit der Finger F1, F2, F3. Mit zunehmender Geschwindigkeit wächst die Kraft die ein Finger aufbringt.

Die Kräfte, die ein Finger aufbringen kann, sind abhängig von der Bewegungsgeschwindigkeit. Somit führt eine kleine Geschwindigkeit auch zu einer kleinen Kraft an der Fingerspitze. Eine große Geschwindigkeit führt zu einer großen Kraft. Dies wird anhand von Abbildung 2.14 verdeutlicht. Für

diesen Versuch wurde ein Objekt in den Greifbereich der Finger gestellt und diese mit konstanter Geschwindigkeit geschlossen. Nach Abschluss der Bewegung wurden die auftretenden Kräfte jedes Fingers gemessen. Um einen Griff mit einer bestimmten Kraft ausführen zu können, müssen die Finger mit einer bestimmten Geschwindigkeit bewegt werden. Um die Geschwindigkeit zu ermitteln, die zum Erreichen einer bestimmten Kraft erforderlich ist, wurde ein B-Spline-Funktionsapproximator mit den Geschwindigkeits-Kraft-Werten trainiert (Details zu B-Splines siehe Anhang B). Ein ähnliches Verfahren wurde u.a. von Zhang und Knoll [Zhang and Knoll 1998, Zhang and Knoll 1997] eingesetzt, um die Regeln eines Fuzzy-Kontrollers zu lernen. Die Ausgabe des Funktionsapproximators ist in Abbildung 2.15 dargestellt.

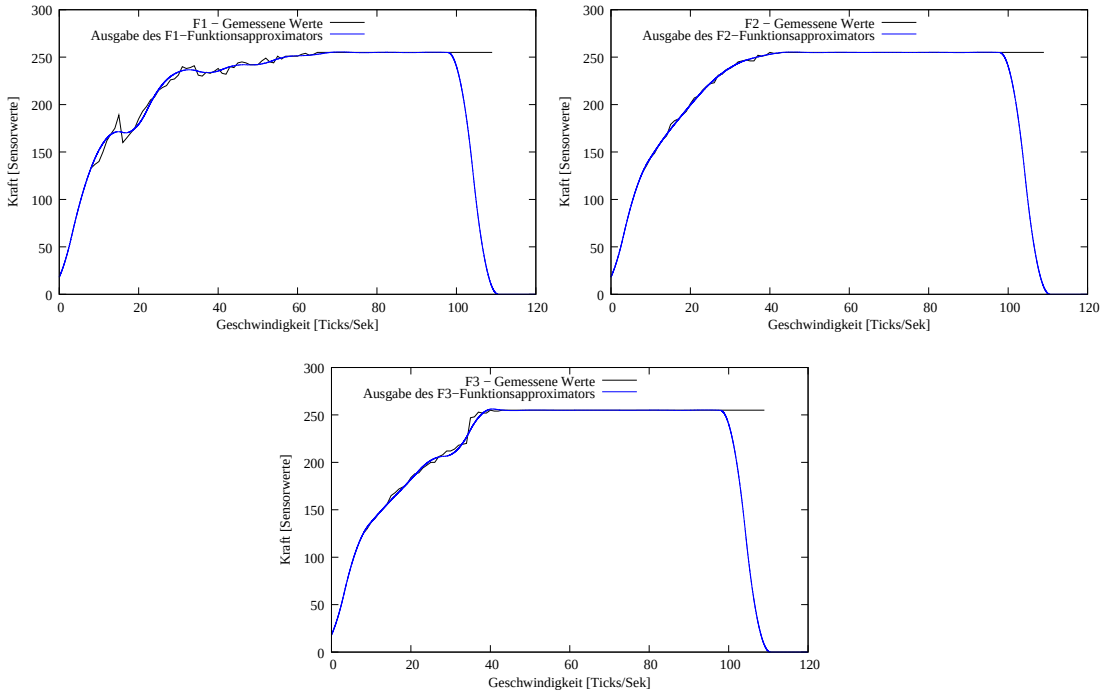


Abbildung 2.15: Gemessene und gelernte Geschwindigkeits-Kraft-Werte für die Finger der BarrettHand.

Der maximale Fehler der Approximation zu den Trainingsdatenpunkten liegt bei einem Wert von 0.01 nach durchschnittlich 86 Lernschritten bei der Verwendung von 19 B-Spline-Basisfunktionen. Durch die Approximation der Geschwindigkeit/Kraft-Kurve mittels eines B-Spline-Funktionsapproximators ist es nun möglich, bei Vorgabe einer Kraft die entsprechende Geschwindigkeit zu ermitteln.

2.3.3 Kontrollmodi

Die Befehle für die Hand werden über eine RS-232 Schnittstelle übertragen. Die maximale Geschwindigkeit beträgt 115000 Baud. Da die Befehle und auch die Antworten der Hand in Form von ASCII-Zeichenketten übertragen werden, ist die Bandbreite ausreichend.

Befehle lassen sich in zwei unterschiedlichen Modi an die BarrettHand senden:

1. High-Level Supervisory Commands

2. Low-Level Real-Time Modus

Prinzipiell lassen sich die Motoren in beiden Modi einzeln oder in beliebigen Paarungen ansteuern, sowie Parameter der Hand, wie z.B. die Bewegungsgeschwindigkeit, etc. setzen und auslesen (siehe auch Anhang A.3). Im *Supervisory* Modus akzeptiert die Hand Befehle vom Benutzerprogramm und gibt die Kontrolle erst wieder an das Programm zurück, wenn der Befehl vollständig ausgeführt wurde. Im Falles eines fehlerhaften Kommandos wird dieses mit einer entsprechenden Fehlermeldung quittiert. Die Kommandostruktur wird in Anhang A.3 genauer erläutert.

Im Gegensatz zum Supervisory Modus können im *Real-Time* Modus kontinuierlich Befehle an die BarrettHand gesendet und Feedback-Informationen erhalten werden. Die Kommunikationsbandbreite ist hierbei abhängig von der Anzahl der Parameter, die gesendet bzw. empfangen werden, sowie der gewählten Baudrate (siehe auch Anhang A.3). Die Datenblöcke, die im Real-Time Modus übertragen werden, lassen sich in zwei Kategorien einteilen: In Kontroll- und Feedback-Blöcke. Jeder Block hat als Präfix einen Kopf von einem Byte, gefolgt von den Daten des jeweiligen Blocks. Der Kopf eines Kontroll-Blocks definiert, ob Kontroll-Daten folgen oder nicht und ob als Antwort Feedback-Informationen gesendet werden sollen oder nicht. Der Kopf eines Feedback-Blocks enthält entweder eine Fehlermeldung oder eine Bestätigung, dass das Kommando verstanden und akzeptiert wurde (für die Details zu den Headern siehe [Barrett CGL Protocol 2002] und [Barrett Manual 2002]). Details zur Installation einer Low-Level Real-Time-Verbindung sind in Anhang A.3 zu finden.

Durch die Kombination von Kontroll- und Feedback-Blöcken kann im Real-Time Modus der BarrettHand ein beliebiges Verhalten anhand der ausgewählten Parameter implementiert werden. Der Nachteil hierbei besteht darin, dass der Feedback-Block als Antwort auf einen Kontroll-Block gesendet wird. Somit hängt der Regelzyklus für z.B. einen Positionsregler immer mindestens einen Schritt hinterher. Ein weiterer Nachteil besteht darin, dass die Motoren träge sind und nachlaufen bzw. eine Vorlaufzeit haben. Im Folgenden wird die BarrettHand im Real-Time Modus betrieben und somit stehen die Kraftinformationen der Finger zu jedem Zeitpunkt zur Verfügung.

2.4 Verbindung von Hand und Roboterarm

Da der Roboterarm nicht über einen zusätzlichen Kraft-Drehmomentsensor verfügt, wurde eine Softwarelösung entwickelt, die es ermöglicht, dass die Kraftwerte, die von der BarrettHand gemessen werden, auch für die Steuerung des Armes genutzt werden können. Somit ist es innerhalb gewisser Grenzen möglich, den Arm kraftgeregelt zu bewegen.

Die Einschränkungen für diese Kraftregelung ergeben sich aus den hardwaretechnischen bzw. mechanischen Details der BarrettHand und des PA10-6C sowie aus der Steuerungssoftware RCCL[2]. Das grundsätzliche Problem bei der Kraftregelung des Arms mit Hilfe der Kraftsensoren der Hand besteht in der Messweise der Sensoren. Sie können nur Kräfte messen, die (idealerweise) senkrecht auf das distale Fingerglied wirken (Abb. 2.16).

Somit ist eine Kraftmessungen bei geschlossenen Fingern nicht möglich. Kräfte die z.B. auf den Handballen, das „Handgelenk“, oder seitlich auf einen Finger wirken, können aufgrund des fehlenden Kraft-Drehmomentsensors, der z.B. wie in [Ferch 2001] oder auch in [Scherer 2004] montiert sein könnte, nicht gemessen werden. Dies ist ein großer Nachteil bei der Ausführung von Bewegungen. Da die Infor-

[2]Es wird zwar RCCL++ für die Steuerung des Arms verwendet, da jedoch RCCL die Grundlage für RCCL++ ist, definiert RCCL die Grenzen der Software.

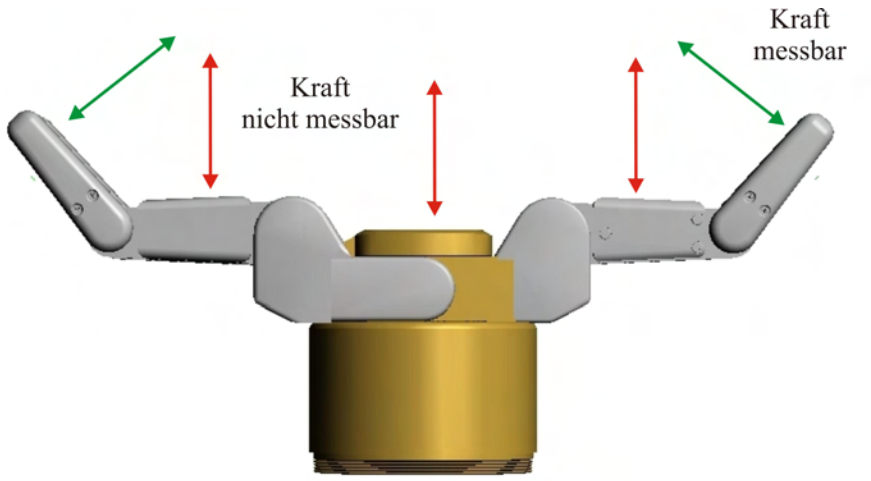


Abbildung 2.16: Die messbaren Kraftkomponenten mit der BarrettHand.

mationen über Kräfte fehlen, die nicht über die Finger gemessen werden können, müssen viele Bewegungen „blind“ ausgeführt werden. Dies muss bei der Implementierung durch den Entwickler berücksichtigt werden.

Zusätzlich zu diesen mechanischen Einschränkungen existiert noch eine weitere, die für die Kraftmessung relevant ist. Durch die Anbindung der BarrettHand über eine serielle Schnittstelle ist die Geschwindigkeit, mit der die Werte der Kraftsensoren von der Hand empfangen werden können, nicht besonders hoch. Die Zeit zwischen dem Empfang von zwei Messwerten beträgt ca. 20 ms, d.h. die Frequenz beträgt 50 Hz. Der Kontrollzyklus für den Roboterarm wird jedoch mit einer Frequenz von 100 Hz ausgeführt. Softwaretechnisch wurde die Asynchronität über eine Shared-Memory Architektur aufgelöst. Soll eine kraftgeregelte Bewegung ausgeführt werden, so wird ein Thread gestartet, der die Kraftwerte aus der BarrettHand ausliest und in den gemeinsamen Speicherbereich schreibt. Der Positionsregler des Roboterarms liest diesen Wert asynchron aus und modifiziert die Trajektorie des Arms in „Echtzeit“. Da nicht in jedem Zyklus für den Arm eine aktuelle Kraftmessung gemacht werden kann, wird die Hälfte der Armbewegungen auf der Annahme von alten Kraftmessungen durchgeführt, was natürlich besonders bei schnellen Bewegungen zu Problemen führen kann. Die Konsequenz ist, dass die Kraftregelung nur für relative langsame Bewegungen (Geschwindigkeit $<$10mm/Sek.) zuverlässig funktioniert und deshalb nicht bei schnelleren Bewegungen verwendet werden sollte.

Ein zusätzlicher Kraft-Drehmomentsensor (KMS) wurde aus drei Gründen nicht am Arm montiert: Der erste Grund ist die schlechte mechanische und elektrische Integrierbarkeit. Da der Anschluss für die BarrettHand von unten in das Gehäuse eingeführt wird, ist eine direkte Montage der Hand auf den Sensor nicht möglich, so dass der Abstand zwischen der Montageplatte des sechsten Gelenks des Roboterarms und der BarrettHand relativ groß wird. Der nächste Punkt besteht darin, dass alle Ausbauplätze des Steuer-Rechners bereits durch die anderen Erweiterungskarten belegt sind und dadurch die Ansteuerung des Kraft-Drehmomentsensors nicht gewährleistet werden kann, da ein KMS in der Regel eine Kontroller-Karte benötigt. Der dritte Grund besteht in der Kalibration des KMS. Dieser muss auf die Gewichtsverteilung der Hand kalibriert werden. Da aber die Finger der BarrettHand ein nicht zu

vernachlässigendes Gewicht haben, verschiebt sich durch die Bewegung der Finger der Massenschwerpunkt der Hand und es wäre eine Rekalibration des Sensors erforderlich. Es ist zwar prinzipiell möglich, die Gewichtsveränderung mit in die Kalibrierung des KMS einzurechnen. Dies ist allerdings nicht trivial. Als Alternative könnten die distalen Fingerglieder durch Kraft-Drehmomentsensoren ersetzt werden, wie in dem TRISK-Projekt [Trisk 2007]. Hierdurch können deutlich mehr Information über die wirkenden Kräfte gesammelt werden. Aber Kontakte an der Basis der Hand oder an den proximalen Fingergliedern sind auch mit diesem Aufbau nicht messbar. Aufgrund der bereits genannten Probleme bezüglich der Ansteuerung der Sensoren ist auch dies mit TASER nicht zu realisieren.

2.5 Kameras

In diesem Abschnitt werden die Kameras beschrieben, die auf dem Service-Roboter installiert sind. Die Kameras sind die Augen des Systems und haben eine besondere Bedeutung, da sie neben den Kraftsensoren der Finger der BarrettHand die einzigen Sensoren sind, die für das Greifen von Objekten genutzt werden. Liefern sie also schlechte Bilder und werden aufgrund von schlechten Bildern falsche Annahmen getroffen, so wirkt sich dies in einer schlechten Performanz des gesamten Systems aus. Auf der Roboter-Plattform sind drei verschiedene Kamerasysteme installiert:

1. ein omnidirektionales Sichtsystem
2. ein Stereokamerasystem auf einer Schwenk-Neige-Einheit (Pan-Tilt-Unit, PTU)
3. eine Mikrokopf-Handkamera

Von den drei Systemen ist im Wesentlichen nur die Handkamera für diese Arbeit relevant. Die beiden anderen Systeme werden in dieser Arbeit nicht verwendet, sollen aber trotzdem kurz beschrieben werden. Das omnidirektionale Sichtsystem und das Stereokamerasystem sind an einer Konstruktion über dem Turm der mobilen Basis befestigt (Abb. 2.17).

2.5.1 Omnidirektionales Sichtsystem

Die Grundlage des omnidirektionalen Sichtsystems ist eine hochauflösende Firewire-Kamera der Firma SONY vom Typ DFW-SX900 mit einer maximalen Auflösung von 1280×1024 Bildpunkten. Die Kamera ist fokussiert auf einen hyperbolischen Spiegel, so dass sie 360° Bilder aufnehmen kann. Abbildung 2.18 zeigt Bilder, die mit dem omnidirektionalen Sichtsystem aufgenommen worden sind. Das linke Bild zeigt ein Bild einer Kameraaufnahme und das Rechte ein daraus erstelltes Panoramabild.

Das omnidirektionale Sichtsystem hat für die Entwicklung von Greifalgorithmen keine Bedeutung, sondern wird für die Navigation und Lokalisation des Roboters eingesetzt. Für Details siehe [Hübner et al. 2006, Hübner 2006, Westhoff 2007].

2.5.2 Stereokamerasystem

Als Übersichtskamera wird ein Stereokamerasystem eingesetzt. Mit diesem System nimmt der Roboter Szenen im Ganzen auf und analysiert sie. Über eine Stereoanalyse können zusätzlich zu den Bildinformationen auch Positionsinformationen im dreidimensionalen Raum berechnet werden. Dies wird z.B.

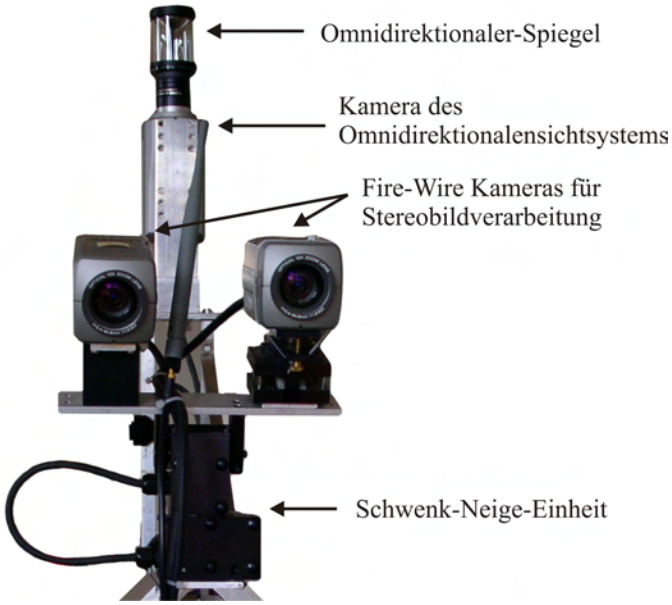


Abbildung 2.17: Zwei der installierten Kamerasysteme auf dem Roboter TASER.

Abbildung 2.18: Bilder, aufgenommen mit dem omnidirektionalen Sichtsystem. Links das Rohbild, rechts das daraus berechnete Panoramabild.

bei der Berechnung einer Greiftrajektorie eines menschlichen Instrukteurs benutzt (siehe Kapitel 6.4 und [Hüser 2008, Hüser et al. 2006, Hüser et al. 2006]).

Das Stereokamerasystem besteht aus zwei SONY DFW-VL 500 Firewire-Kameras mit 12-fachen optischen Zoom. Die Kameras sind auf einer Schwenk-Neige-Einheit montiert. Die Schwenk-Neige-Einheit vom Typ PTU-46-17.5 wird von der der Firma Directed Perception [Directed Perception 2007] produziert.

Mit Hilfe der PTU ist der Service-Roboter in der Lage, Objekte und Personen zu verfolgen und im Blickfeld zu halten, ohne dass eine Bewegung der mobilen Basis erforderlich ist. Die Geschwindigkeit der PTU ist ausreichend hoch (maximal 300°/Sek), so dass auch sich schnell bewegende Objekte bzw. Menschen verfolgt werden können. Der Abstand zwischen den optischen Achsen der Kameras

(Stereobasis) beträgt 110 mm, die maximale Auflösung der Kameras beträgt 640×480 Pixel bei einer Bildwiederholrate von 30 Hz im Bildformat YUV 411.

2.5.3 Handkamera

Da das Stereokamerasystem nur als Übersichtssystem dient und die Auflösung für eine exakte Positionsbestimmung von Objekten in einer Szene nicht ausreichend ist, wird für eine genaue Positionierung der Hand relativ zum Objekt noch ein weiterer Sensor benötigt. Da die Positionierung kontaktfrei erfolgen soll, wird auch hierfür eine Kamera eingesetzt. Hierbei eignet sich der Einsatz einer Mikrokopf-Kamera, die an der Hand bzw. am Handgelenk des Roboterams montiert ist (Abb. 2.19).

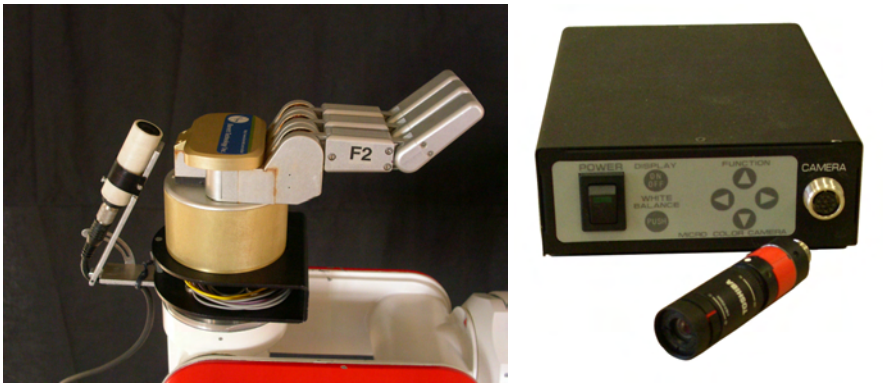


Abbildung 2.19: Die JAI CV-M2250 Mikrokopf-Farbkamera. Das linke Bild zeigt die Kamera montiert an dem Adapter zwischen Roboterarm und Hand. Rechts die Kamera mit der zugehörigen Kontrolleinheit.

In diesem Fall wird eine JAI CV-M2250 Mikrokopf-Farbkamera [JAI 2007] verwendet. Die Kamera besteht im wesentlichen aus zwei Teilen (Abb. 2.19)

- dem Kamerakopf mit Objektiv
- dem Controller

Die Länge des Kopfes beträgt 99 mm (Kamerakopf incl. Stecker) der Durchmesser beträgt 17 mm. Die Länge inklusive Objektiv hängt von der Brennweite des verwendeten Objektivs und der Einstellung des Fokuspunktes ab. Für den hier verwendeten Hardwareaufbau wird ein Objektiv mit einer Brennweite von 7.5 mm eingesetzt. Der Fokuspunkt ist so gewählt, das die Objekte in einem Abstand von ca. 2.500 mm zu der Handfläche der BarrettHand fokussiert sind. Mit dieser Einstellung hat die Kamera eine Gesamtlänge von 120 mm.

Zum Schutz der Kamera vor Beschädigungen ist sie in eine Kunststoffhülse eingefasst. Die Kamera arbeitet nach dem PAL-Standard und hat eine effektive Auflösung von 752×582 Bildpunkten[3]. Der Kamerakopf ist über ein Kabel mit der Kamera-Kontroll-Einheit verbunden, die über dem Batteriefach des TASER montiert wurde. Sie bietet die Möglichkeit, verschiedene Parameter der Kamera, wie z.B.

[3] Standard PAL Auflösung ist 768×576 Pixel.

Modell	JAI CV-M2250
TV Standard	PAL
CCD Sensor	$1/2''$ IT CCD Sensor
Effektive Pixel	752×582
Rauschabstand	$> 48\ dB$
Dimensionen Kamerakopf	17×99 mm
Masse Kamerakopf	$5,7$ g
Dimensionen Kontrolleinheit	$40 \times 110 \times 154$ mm (B×H×T)
Masse Kontrolleinheit	550 g

Tabelle 2.1: Technische Daten der JAI M2250 Mikrokopf-Farbkamera.

Weissabgleich, Blende, etc. einzustellen. Die Parameter können entweder über ein Tastenfeld an der Kontrolleinheit oder über eine serielle Schnittstelle eingestellt werden. Die Kamera benötigt eine Versorgungsspannung von 12 V. Die technischen Details der Kamera sind in Tabelle 2.1 zusammengefasst.

Die Kamera-Kontroll-Einheit ist über eine S-VHS Verbindung an einer Matrox Meteor Framegrabber Karte angeschlossen. Weitere Details der Kamera können dem zugehörigen Handbuch [JAI 2002] entnommen werden.

Kapitel 3

Greifen mit einer Roboterhand

Dieses Kapitel befasst sich mit dem Greifen von Objekten mit einer Multifinger-Roboterhand. Zunächst werden einige grundlegende Definitionen eingeführt. Anschließend wird die Definition für einen Griff gegeben. Hierbei wird ein zentraler Begriff der *Grasp-Wrench-Space* (GWS) sein. Da es für diesen Ausdruck, der den Raum der Kräfte und Drehmomente beschreibt, die einen Griff charakterisieren, keine adäquate Übersetzung ins Deutsche gibt, wird er aus dem Englischen übernommen. Ein historischer Überblick über Manipulation mittels Robotern ist in [Okamura et al. 2000] zu finden.

3.1 Definition eines Griffs

Ein Griff besteht aus einer Menge von Kontakten. Dabei ist ein Kontakt definiert als ein Berührungs*punkt* zwischen zwei Objekten. Hierbei gilt es zu beachten, dass nur von Kontaktpunkten und nicht von Kontaktflächen ausgegangen wird. Laut [Murray et al. 1994] gibt es drei unterschiedliche Arten von Kontakten:

1. Kontakte ohne Reibung
2. Kontakte mit Reibung
3. Weiche-Finger-Kontakte

Diese drei Arten von Kontakten unterscheiden sich voneinander in der Art, wie Kräfte und Drehmomente in den Kontaktpunkten gegeneinander aufgebracht werden können. Das kartesische Koordinatensystem der Kontakte wird dabei so gelegt, dass die Z-Achse parallel zu der entsprechenden Oberflächennormalen liegt. Des Weiteren werden die Koordinaten eines Kontaktpunkts relativ zum Objektursprung angegeben. Die entsprechenden Transformationen sind in Abbildung 3.1 dargestellt. W ist dabei das Welt-Koordinatensystem, O das des Objekts und C_i das des i-ten Kontakts. Die homogenen Transformationen ${}^{W}T_{O}$ und ${}^{O}T_{C_i}$ stellen den Bezug zwischen den Koordinatensystemen Welt und Objekt bzw. Objekt und Kontaktpunkt her. Der Ursprung des Objekts ist hierbei der Massenschwerpunkt des Objekts. Bei den Objekten wird von einer homogenen Massenverteilung ausgegangen. Somit entspricht der Massenschwerpunkt des Gegenstandes auch dem geometrischen Zentrum. Für die Definition der Kontakttypen ist zunächst der Begriff des Wrench zu definieren.

3.1.1 Wrench

Ein Kontakt beschreibt die Verbindung zwischen zwei Objekten und die dabei auftretenden Kräfte und Momente. Dies wird auch zu einem *Wrench* zusammengefasst. Wörtlich übersetzt bedeutet Wrench

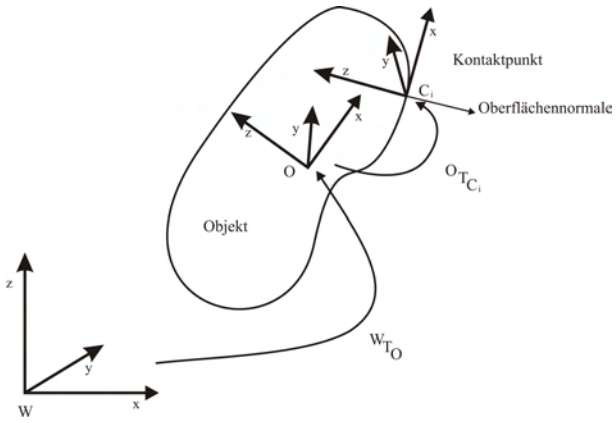


Abbildung 3.1: Die Koordinatentransformationen für einen Kontaktpunkt. W stellt das Weltkoordinatensystem dar, O das des Objekts und C_i das Koordinatensystem des Kontakts.

Schraubenschlüssel oder auch Stoß bzw. Verrenkung. Da dieser Begriff in diesem Zusammenhang allerdings eher unpassend ist, wird der Ausdruck nicht übersetzt. Ein Wrench ist definiert als:

$$w = \begin{bmatrix} f \\ \tau \end{bmatrix} \tag{3.1}$$

mit der Kraft f im Kontaktpunkt und dem Drehmoment τ. Für dreidimensionale Probleme ist $w \in \mathbb{R}^6$ mit $f = [f_x\ f_y\ f_z]^T$ und $\tau = [\tau_x\ \tau_y\ \tau_z]^T$.

Wird

$$w_{c_i} = \begin{bmatrix} f_{c_i} \\ \tau_{c_i} \end{bmatrix} \tag{3.2}$$

geschrieben, so ist der Wrench w in Bezug auf das Kontaktkoordinatensystem c_i definiert.

3.1.2 Kontakttypen

Kontakte ohne Reibung

Kontakte ohne Reibung stellen den einfachsten Typ von Kontakten dar. Über einen solchen Kontakt können nur Kräfte entlang der Kontaktnormalen aufgebracht werden. Ein entsprechendes Modell ist in Abbildung 3.2 dargestellt. Formal ist ein Kontakt ohne Reibung definiert durch:

$$w_{c_i} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} f_{c_i} = B_{c_i} f_{c_i} \text{ mit } f_{c_i} \geq 0 \tag{3.3}$$

mit f_{c_i} der Kraft für den i-ten Kontakt und B_{c_i} der Wrench-Basis. Das Ergebnis ist ein Wrench W_{c_i}, definiert in Bezug zum Kontaktkoordinatensystem. Da bei einem Kontakt ohne Reibung nur Kräfte entlang der Kontaktnormalen aufgebracht werden können ist f_{c_i} ein Skalar. Die Einschränkung $f_{c_i} \geq 0$

soll veranschaulichen, dass die Kräfte lediglich Druck auf das Objekt ausüben können, aber nicht daran ziehen können. Sie kann auch formuliert werden durch:

$$f_{c_i} \in FC_{c_i} \quad \text{mit} \quad FC_{c_i} = \{f_{c_i} \in \mathbb{R}^+\} \tag{3.4}$$

Die FC_{c_i} sind die Randbedingungen, die von f_{c_i} erfüllt sein müssen. Sie werden auch als das Modell des *Reibungskegels* bezeichnet. Kontakte ohne Reibung sind allerdings nur theoretischer Natur und dienen lediglich zur Veranschaulichung des Modells, da in der Realität immer Reibungskräfte bei einem Kontakt zwischen zwei Objekten auftreten.

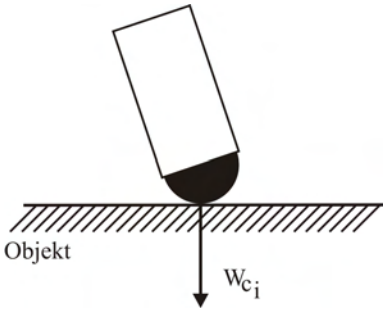


Abbildung 3.2: Das Modell für einen Kontakt ohne Reibung. Es können nur Kräfte entlang der Kontaktnormalen (w_{c_i}) aufgebracht werden.

Kontakte mit Reibung

Bei Kontakten zwischen zwei Objekten treten immer auch Reibungskräfte auf. Die tangential auftretenden Kräfte können hierbei durch das Coulomb-Reibungs-Modell bestimmt werden. Dies Modell ist definiert durch:

$$|f_t| \leq \mu f_\perp \tag{3.5}$$

wobei f_t die tangentiale Kraftkomponente, $f_\perp$ die Kraftkomponente entlang der Kontaktnormalen und $\mu > 0$ der Reibungskoeffizient ist. Bei einem dreidimensionalen Problem kann dies auch geschrieben werden als

$$\sqrt{f_x^2 + f_y^2} \leq \mu f_z \tag{3.6}$$

mit f_x und f_y als den orthogonalen Kraftkomponenten der tangentialen Ebene. Der Reibungskoeffizient μ kann empirisch bestimmt werden und ist von den Materialien des Kontakts abhängig. Eine Übersicht von Reibungskoeffizienten ist in Tabelle 3.1 gegeben. Die Werte basieren auf den Werten aus der Tabelle für Reibungskoeffizienten aus dem CRC Handbook of Physics [Lide 2005]. Da das Handbuch nicht für alle in Tabelle 3.1 aufgeführten Paarungen einen festen Wert vorgibt, sondern einen Wertebereich definiert, sind diese Werte auf den jeweiligen Mittelwert festgelegt worden. Ein Modell für einen solchen Kontakt ist in der Abbildung 3.3 dargestellt. Wie man sehen kann, bestimmt der Reibungskoeffizient den Öffnungswinkel des Kegels. Hieraus wird auch ersichtlich, dass die tangentialen Kräfte innerhalb des Kegels liegen müssen, damit der Kontakt stabil ist und nicht rutscht. Dieser Kegel wird in der Literatur als *Reibungskegel* (*friction cone*) bezeichnet. Der Öffnungswinkel dieses Kegels ist $\alpha = \arctan(\mu)$. Das

Material	Metall	Glas	Plastik	Holz	Gummi
Metall	0.2	0.2	0.2	0.3	1,0
Glas	0.2	0.2	0.2	0.3	1.0
Plastik	0.2	0.2	0.3	0.4	1.0
Holz	0.3	0.3	0.4	0.4	1.0
Gummi	1.0	1.0	1.0	1.0	2.0

Tabelle 3.1: Reibungskoeffizienten [Lide 2005].

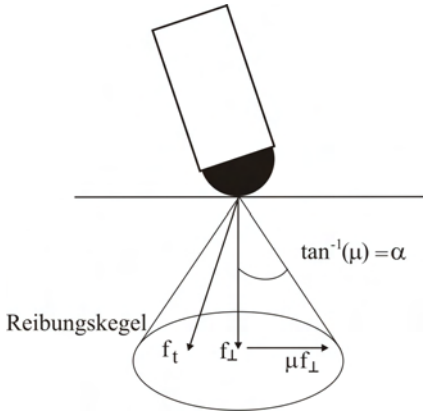


Abbildung 3.3: Das Modell für einen Kontakt mit Reibung. Zusätzlich zu den Kräften parallel zu der Kontaktnormalen $f_\perp$ können noch tangentiale Kräfte f_t aufgebracht werden. Der Kontakt beginnt zu rutschen, wenn $|f_t| > \mu f_\perp$.

Modell für einen Kontakt mit Reibung ist gegeben durch:

$$w_{c_i} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} f_{c_i} = B_{c_i} f_{c_i}, \qquad f_{c_i} \in FC_{c_i} \tag{3.7}$$

$$\text{mit } FC_{c_i} = \{f \in \mathbb{R}^3 \mid \sqrt{f_x^2 + f_y^2} \leq \mu f_z, f_z \geq 0\}. \tag{3.8}$$

mit f_x und f_y als den tangentialen Kraftkomponenten und f_z als der Kraft entlang der Kontaktnormalen. μ ist der Reibungskoeffizient.

Weiche-Finger-Kontakte

Ein erweitertes Kontaktmodell ist das Modell des *Weiche-Finger-Kontakts* (*soft-finger contact*). Hierbei ist es möglich, dass nicht nur Kräfte innerhalb eines Reibungskegels aufgebracht werden können, sondern auch Momente um die Kontaktnormale. Dies ist in Abbildung 3.4 illustriert. Zusätzlich zu der Kraft entlang der Kontaktnormalen $f_\perp$ ergibt sich ein Drehmoment f_m um $f_\perp$. Somit ist das Modell eines

solchen Kontakts definiert durch

$$w_{c_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} f_{c_i}, \qquad f_{c_i} \in FC_{c_i} \tag{3.9}$$

$$\text{mit } FC_{c_i} = \{f \in \mathbb{R}^4 \mid \sqrt{f_x^2 + f_y^2} \leq \mu f_z, f_z \geq 0, \mid f_m \mid \leq \gamma f_z\} \tag{3.10}$$

mit f_x und f_y den tangentialen Kraftkomponenten und f_z der Kraft entlang der Kontaktnormalen und f_m dem Moment um die Kontaktnormale. μ ist der Reibungskoeffizient und γ ein *Momentkoeffizient*. Dieser wird verwendet, um das Modell zu vereinfachen indem das Moment als Kraft, beschränkt durch einen Koeffizienten, eingebracht wird.

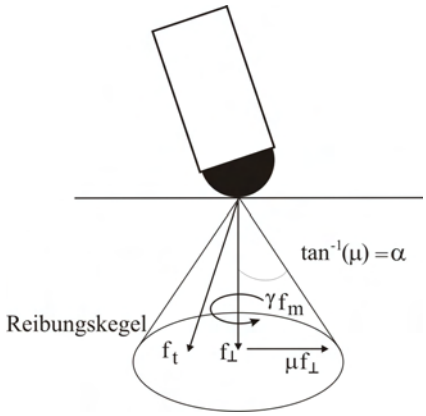


Abbildung 3.4: Modell eines Weichen-Finger-Kontakts. Zusätzlich zu den tangentialen Kräften kann noch ein Drehmoment f_m um die Kontaktnormale aufgebracht werden.

Allgemeine Definition eines Kontakts

Allgemein zusammengefasst wird ein Kontakt durch eine *Wrench-Basis* $B_{c_i} \in \mathbb{R}^{p \times m_i}$ modelliert, wobei p die Dimension des Wrench-Space und m_i die Anzahl der Kräfte, die unabhängig von einander aufgebracht werden können, sowie einen Reibungskegel FC_i modelliert. Für die Berechnungen im dreidimensionalen Raum gilt $p = 6$ und $p = 3$ bei der Berechnung von zweidimensionalen Problemen. Die Dimension der Wrench-Basis m_i zeigt dabei die Anzahl der Kräfte auf, die in einem Kontakt unabhängig voneinander aufgebracht werden können. Die Bedingungen für den Reibungskegel sind definiert durch [Murray et al. 1994]:

1. FC_{c_i} ist eine nicht leere Untermenge aus $\mathbb{R}^{m_i}$
2. $f_1, f_2 \in FC_{c_i} \Rightarrow \alpha f_1 + \beta f_2 \in FC_{c_i}$ für $\alpha, \beta > 0$

Die Menge der zulässigen Kontaktkräfte für einen Kontakt ist definiert durch:

$$w_{c_i} = B_{c_i} f_{c_i} \qquad f_{c_i} \in FC_{c_i} \tag{3.11}$$

Für diese Arbeit werden nur Kontakte mit Reibung betrachtet.

3.1.3 Griffe aus einer Menge von Kontakten

Da jeder der Kontakte in einem eigenen Koordinatensystem beschrieben wird, ist es für die Berechnung der Kräfte, die bei einem Griff auf ein Objekt wirken, erforderlich, die Kontakte in Bezug auf ein gemeinsames Koordinatensystem zu definieren. Die Kontakte lassen sich mittels einer *Grasp-Map* in ein beliebiges Koordinatensystem transformieren.

In der Regel wird hierbei der Massenschwerpunkt des zugehörigen Objekts als Ursprung für ein gemeinsames Koordinatensystem verwendet. Sei (p_{oc_i}, R_{oc_i}) (mit p_{oc_i} der Translation zwischen beiden Koordinatensystemen und R_{oc_i} der Rotation) die Konfiguration, also die Position und Orientierung, des i-ten Kontakts relativ zum Objektkoordinatensystem, dann können die Wrenches mittels der folgenden Transformation von dem Kontaktkoordinatensystem in das Objektkoordinatensystem transformiert werden:

$$w_o = AD^T_{g^{-1}_{oc_i}} w_{c_i} = \begin{bmatrix} R_{oc_i} & 0 \\ \hat{p}_{oc_i} R_{oc_i} & R_{oc_i} \end{bmatrix} B_{c_i} f_{c_i}, \qquad f_{c_i} \in FC_{c_i} \tag{3.12}$$

mit $\hat{p}$ als Kreuzprodukt von p_{oc_i} und f_{oc_i}. Dies entspricht dem Drehmoment, das erzeugt wird, wenn die Kraft f_{c_i} in Entfernung p_{oc_i} zum Referenzkoordinatensystem wirkt. Die Matrix $AD^T_{g^{-1}_{oc_i}}$ ist die Transformationsmatrix, welche die Kontakt-Wrenches in Objekt-Wrenches transformiert. Man definiert nun

$$G_i := AD^T_{g^{-1}_{oc_i}} B_{c_i} \tag{3.13}$$

$$= \begin{bmatrix} R_{oc_i} & 0 \\ \hat{p}_{oc_i} R_{oc_i} & R_{oc_i} \end{bmatrix} B_{c_i}, \tag{3.14}$$

die lineare Abbildung zwischen den Kontaktkräften in Bezug zu B_{c_i} und dem Objekt-Wrench mit $G_i \in \mathbb{R}^{p \times m_i}$, als *Contact-Map*.

Bei k Kontakten, die auf ein Objekt einwirken, ist der gesamte Wrench die Summe aller Wrenches jedes Kontakts, transformiert in das Objektkoordinatensystem. Sei die Abbildung

$$G : \mathbb{R}^m \to \mathbb{R}^p, \quad m = m_1 + \ldots + m_k. \tag{3.15}$$

definiert als *Grasp-Map*. Sie stellt die Abbildung zwischen den Kontaktkräften und der Gesamtkraft, die auf das Objekt einwirkt, dar.

Da die Contact-Maps linear sind und die Wrenches überlagert werden können, kann der Objekt-Wrench geschrieben werden als:

$$w_o = G_1 f_{c_1} + \cdots + G_k f_{c_k} = [G_1, \ldots, G_k] \begin{bmatrix} f_{c_1} \\ \vdots \\ f_{c_k} \end{bmatrix} \tag{3.16}$$

Definiert man die Grasp-Map als:

$$G = \left[AD^T_{g^{-1}_{oc_1}} B_{c_1}, \ldots, AD^T_{g^{-1}_{oc_k}} B_{c_k} \right]. \tag{3.17}$$

kann ein Griff abschließend definiert werden als Paar (G, FC). Geschrieben als

$$W_0 = G f_c \qquad \text{mit} \qquad f_c = \begin{bmatrix} f_{c_1} \\ \vdots \\ f_{c_k} \end{bmatrix} \in FC. \tag{3.18}$$

mit

$$\begin{aligned} f_c &= (f_{c_1}, \ldots, f_{c_k}) \in \mathbb{R}^m \\ FC &= (FC_{c_1} \times \ldots \times FC_{c_k}) \in \mathbb{R}^m \\ m &= m_1 + \cdots + m_k. \end{aligned}$$

G beschreibt die geometrischen Eigenschaften der i Kontakte. FC ist der generalisierte multidimensionale Reibungskegel. Somit ist ein Griff vollständig durch G und FC definiert.

3.1.4 Grifftypen

Abhängig von der Art und Anzahl der Kontakte werden zwei unterschiedliche Grifftypen definiert. Die Typen sind die *Präzisions-* und *Powergriffe*. Bei einem Präzisionsgriff besteht lediglich ein Kontakt zwischen dem Objekt und je einem Finger. Es gibt keine weiteren Kontakte zwischen Hand und Objekt. Im Gegensatz dazu verfügt ein Powergriff über beliebig viele Kontaktpunkte zwischen Hand und Objekt. Griffe mit mindestens einem Kontakt zwischen dem Objekt und der Handfläche und beliebig vielen Kontakten an den Fingern der Hand sind also Powergriffe.

3.2 Kraftschluss bei einem Griff

Eine wichtige Eigenschaft bei einem Griff ist der *Kraftschluss* (*force closure*). Kraftschluss bedeutet, dass der Griff in der Lage ist, den externen Kräften, die bei einem Griff auftreten können, entgegen zu wirken, ohne dass der Griff instabil wird. Solche externen Kräfte können z.B. durch einen Druck von außen, auf das Objekt oder durch die Schwerkraft verursacht werden. Detailliert ist dies in [Bicchi 1995] beschrieben worden.

3.2.1 Definition Kraftschluss

Ein Griff gilt also als kraftschlüssig, wenn eine Menge von Kontaktkräften, $f_c \in FC$ existiert, so dass der Griff allen externen Wrenches $F_e \in \mathbb{R}^p$ widerstehen kann.

$$Gf_c = -F_e \tag{3.19}$$

Anders lässt sich im Gegenzug auch formulieren, dass $G(FC)$ den Raum $\mathbb{R}^p$ aufspannen muss. Für den dreidimensionalen Fall mit $F_e \in \mathbb{R}^6$ gilt somit:

$$G(FC) = \mathbb{R}^6. \tag{3.20}$$

Wichtig ist in dem Zusammenhang noch zu erwähnen, dass das Kraftschluss-Kriterium nur durch interne Kräfte charakterisiert ist. Diese Kräfte haben nicht die Auswirkung eines Kräfte-Netzes auf das Objekt, im Gegensatz zu dem Netz der Kontaktkräfte. Das heißt es gilt $Gf_n = 0$ mit f_n als interne Kraft.

3.2.2 Bestimmung des Kraftschlusses

Für die Bestimmung, ob ein Griff kraftschlüssig ist oder nicht, gibt es mehrere Möglichkeiten. Sie wurden z.B. von Chen und Burdick [Chen and Burdick 1993] und Nguyen [Nguyen 1988] entwickelt. Bicchi

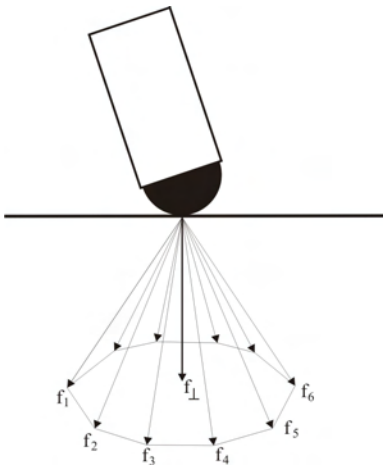


Abbildung 3.5: Modell der diskreten Approximation des Reibungskegels. Der Reibungskegel wird durch die Vektoren $f_1, \ldots, f_n$ approximiert.

beschreibt die Lösung in [Bicchi 1995] als Differentialgleichung und [Liu 1999] stellt eine geometrische Lösung für das Problem vor.

Eine der wohl am häufigsten gebrauchten Varianten ist die Berechnung der konvexen Hülle über den dreidimensionalen Reibungskegel, den *Wrench-Space* [Ferrari and Canny 1992]. Da eine kontinuierliche Berechnung des Reibungskegels nicht möglich ist, wird dieser durch eine diskrete Anzahl von Vektoren approximiert. Die Anzahl der Vektoren hat natürlich einen Einfluss auf die Genauigkeit der Analyse, da aber alle Vektoren in die Berechnung der konvexen Hülle eingehen, hat sie auch einen erheblichen Einfluss auf die Performanz des Systems. In Abbildung 3.5 ist ein Modell für die Approximation dargestellt. Mathematisch ist eine solche Approximation definiert durch:

$$f \approx \sum_{j=0}^{m-1} = \alpha_j f_j \tag{3.21}$$

$$f_j = \hat{f} \perp + \mu \cos\left(\frac{2\pi j}{m}\right) \hat{f}_x + \mu \sin\left(\frac{2\pi j}{m}\right) \hat{f}_y \tag{3.22}$$

$$\alpha_j \geq 0 \tag{3.23}$$

$$\sum_{j=0}^{m-1} \alpha_j = \|f_\perp\| \tag{3.24}$$

Somit wird jede Kontaktkraft f begrenzt durch die Summe einer finiten Menge von Kräften entlang der Grenze des Reibungskegels f_j. f_j ist dabei die Kraft entlang einer Kante der Pyramide, durch die der Kegel approximiert wird. Die gesamte Pyramide wird durch den Betrag der Kraft entlang der Normalen $f_\perp$ normiert.

3.2.3 Berechnung des Grasp-Wrench-Space

Um einen Griff im Detail analysieren zu können, sind Berechnungen erforderlich, die über die einfache Berechnung des Kraftschluss-Kriteriums hinausgehen. Solche Berechnungen können mit Hilfe eines

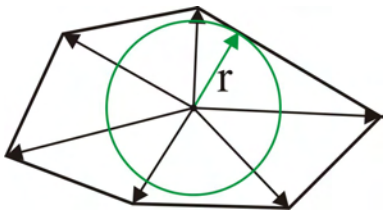


Abbildung 3.6: Ein Modell des Qualitätsmaß im zweidimensionalen Grasp-Wrench-Space. r ist der maximale Radius, so das der Kreis innerhalb des Polygons liegt, das die konvexe Hülle des Grasp-Wrench-Space beschreibt.

Grasp-Wrench-Space (GWS) durchgeführt werden. In diesem Raum können alle Kräfte und Drehmomente analysiert werden, die bei einem Griff aufgebracht werden können. Im dreidimensionalen Fall ist der GWS ein sechsdimensionaler Raum, der über die Kräfte und Drehmomente definiert ist, die bei einem Griff auftreten.

Ferrari und Canny [Ferrari and Canny 1992] stellen zwei Möglichkeiten vor, einen solchen Grasp-Wrench-Space zu berechnen. In jedem der beiden Fällen bildet die in Abschnitt 3.2.2 beschriebene konvexe Hülle die Grenzen des Raumes. Zum einen wird die maximale Kraft entlang der Normalen und zum anderen die Summe der Kräfte entlang der Normalen beschränkt. Beide Ansätze beruhen auf der linearen Approximation des Reibungskegels, um den Raum aufzuspannen, in dem alle möglichen Wrenches berechnet werden können. Sei g der generalisierte Kraftvektor, der aus den Kraftvektoren entlang der Normalen jedes Kontakts besteht:

$$g = \begin{bmatrix} \|f_{\perp_1}\| \\ \|f_{\perp_2}\| \\ \vdots \\ \|f_{\perp_n}\| \end{bmatrix}. \tag{3.25}$$

Als *lokales* Griff-Qualitätsmaß kann laut Canny et al. [Ferrari and Canny 1992] nun

$$LQ_w = \max \frac{\|w\|}{\|g\|} \tag{3.26}$$

definiert werden. Es definiert das beste Verhältnis zwischen einem Wrench und der aufgebrachten Kraft. Hierbei ist $\|w\|$, die Norm eines Wrench, definiert als:

$$\|w\| = \sqrt{\|f\|^2 + \lambda\|\tau\|^2}. \tag{3.27}$$

λ ist dabei ein Faktor, der Kraft und Drehmoment in Beziehung setzt. Während nach Canny [Ferrari and Canny 1992] die Wahl von λ mehr oder weniger beliebig erfolgen kann, hat Pollard in [Pollard 1994] gezeigt, dass $\|\tau\| \leq \|f\|$ sein muss. Dies kann garantiert werden, indem $\lambda = \frac{1}{r}$ gewählt wird, mit r als dem maximalen Abstand zu dem Ursprung des Wrench Space, was in den meisten Fällen dem Objektschwerpunkt entspricht. Somit wird auch sichergestellt, dass das Qualitätsmerkmal unabhängig von der Skalierung eines Objekts ist.

Nun definiert man ein Qualitätsmaß für einen Griff durch:

$$Q = \min_w LQ_w \tag{3.28}$$

wobei $\|w\|$ so gewählt wird, dass $\|g\| = 1$ gilt. Anhand der Definition von $\|g\|$ werden im Folgenden unterschiedliche Qualitätsmaße definiert. Die Bedingung für einen kraftschlüssigen Griff kann nun als

geometrische Interpretation von Q formuliert werden und zwar derart, dass ein Griff kraftschlüssig ist, wenn der Ursprung des Wrench-Space in der konvexen Hülle der Wrenches liegt, die den Raum aufspannen. Des Weiteren kann Q interpretiert werden als Radius der größten Kugel, die innerhalb der konvexen Hülle liegt, zentriert um den Ursprung des Wrench-Space, (Abb. 3.6). Sie definiert also den größtmöglichen Abstand zur nächsten Hyperfläche in der konvexen Hülle und garantiert somit eine untere Schranke für Störkräfte in alle Richtungen.

Anhand der Definition von $\|g\|$ lassen sich nun unterschiedliche Qualitätsmerkmale definieren. In [Ferrari and Canny 1992] werden zwei unterschiedliche Varianten vorgeschlagen. Die erste definiert $\|g\|$ über eine L_∞-Metrik. Somit entspricht $\|g\|$:

$$\|g\| = \max(g_1, \ldots, g_n). \tag{3.29}$$

Nimmt man an, dass die aufgebrachten Kräfte eigenständig und linear unabhängig sind, wenn man die maximal aufgebrachte Kraft jedes Fingers optimieren möchte, und sie durch die L_∞ Norm begrenzt sind, so kann man davon ausgehen, dass die Wrenches, die auf das Objekt ausgeübt werden können, durch $\|f_\perp\| = 1$ begrenzt sind. Somit kann die Menge aller möglichen Wrenches des i-ten Kontakts definiert werden durch:

$$W_i = \left\{ w_i | w_i = \sum_{i=0}^{m-1} \alpha_{i,j} w_{i,j},\ \alpha_{i,j} \geq 0,\ \sum_{j=0}^{m-1} \alpha_{i,j} \geq 1 \right\}. \tag{3.30}$$

Da $w = \sum_{i=0}^{n-1} w_i$ ist dies

$$W_{L_\infty} = \bigoplus_{i=0}^{n-1} W_i \tag{3.31}$$

mit $\bigoplus$ als Symbol für die Minkowski-Summe[1]. Ersetzt man nun die Minkowski-Summe durch eine Operation bezüglich der konvexen Hülle, so ergibt sich:

$$W_{L_\infty} = \text{ConvexHull}(\bigoplus_{i=0}^{n-1} w_{i,0}, \ldots, w_{i,m-1}). \tag{3.32}$$

Mit dieser Definition von $\|g\|$ wird die maximale Kraft der Finger unabhängig von einander minimiert.

Im zweiten Fall wird $\|g\| = 1$ über eine L_1 Norm definiert, dem *Einheits-Wrench-Space*. Hierbei geht es um die Minimierung der gesamten Kräfte bei einem Griff. Da die Summe der Kräfte proportional zu den Motordrehmomenten ist, kann mit diesem Qualitätsmaß die erforderliche Motorleistung minimiert werden. Die Hypothese hierbei ist, dass die obere Grenze für die Summe der Kräfte gleich 1 ist. $\|g\|$ ist also definiert als:

$$\|g\| = (g_1 + g_2 + \ldots + g_n). \tag{3.33}$$

Die Summe der Wrenches, also der „Gesamt-Wrench“ w, der auf das Objekt wirkt, ist definiert durch:

$$w = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \alpha_{i,j} w_{i,j}. \tag{3.34}$$

[1] Die Minkowski-Summe zweier Mengen A und B mit Elementen aus einem Vektorraum ist die resultierende Menge der Summen aller Elemente aus A und aller Elemente aus B: $A \bigoplus B := \{a + b \,|\, a \in A, b \in B\}$ [Berg et al. 2000].

Die Menge aller möglichen Wrenches ist gegeben durch:

$$W_{L_1} = \left\{ w_i | w_i = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \alpha_{i,j} w_{i,j},\ \alpha_{i,j} \geq 0,\ \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \alpha_{i,j} \geq 1 \right\} \tag{3.35}$$

dies entspricht

$$W_{L_1} = \text{ConvexHull}(\bigcup_{i=0}^{n-1} w_{i,0}, \ldots, w_{i,m-1}). \tag{3.36}$$

Somit ist der „Gesamt-Wrench“ definiert über die konvexe Hülle, die über die Wrenches der einzelnen Kontakte aufgespannt wird.

Abschließend gibt es noch zu bemerken, dass $W_{L_\infty} \supseteq W_{L_1}$ und $Q_\infty \geq Q_1$, da in beiden Fällen die konvexe Hülle gebildet wird, wobei die zweite Menge an Wrenches, über die die Hülle gebildet wird, Teilmenge der ersten Menge ist.

3.3 Kräfte bei einem Griff

Die Kräfte, die bei einem Griff auftreten, lassen sich nicht durch den Wrench-Space berechnen. In einen Wrench-Space können lediglich externe Kräfte als virtuelle Wrenches mit eingebracht werden, die einen Einfluss auf die Qualität des Griffs haben. Für die Berechnung und Optimierung gibt es eine Vielzahl von Lösungsmöglichkeiten, wie z.B. die in [Kerr and Roth 1986, Liu and Li 2004, Redmond et al. 2001, Liu et al. 2004a, Wöhlke 1994] und [Buss et al. 1996] vorgestellten Verfahren. In der Regel basieren die Ideen auf der Verwendung linearer und nicht-linearer Optimierung unter Berücksichtigung der Randbedingungen, die sich aus den Reibungskegeln ergeben. Eine Ausnahme bildet das in [Xia et al. 2004] vorgestellte Verfahren, das eine Optimierung der Kräfte mittels Neuronaler Netze vornimmt.

Für diese Arbeit wurde das in [Han et al. 2000] und [Miller 2001] beschriebene Verfahren angewandt. Die Grundzüge des Verfahrens sollen im Folgenden kurz vorgestellt werden und orientieren sich an den genannten Schriften.

Grundlage ist die Annahme von [Buss et al. 1996], dass die Bedingungen der Haftreibung, wie sie z.B. in Gleichung 3.8 formuliert sind, äquivalent zu einer bestimmten positiv semi-definiten Matrix sind. Diese Matrix ist für einen Kontakt mit Reibung gegeben durch:

$$P_i := \begin{bmatrix} \mu_i w_{i3} & 0 & w_{i1} \\ 0 & \mu_i w_{i3} & w_{i2} \\ w_{i1} & w_{i2} & \mu_i w_{i3} \end{bmatrix} \tag{3.37}$$

mit $w_i \in \mathbb{R}^{m_i}$ dem Wrench an Kontakt i und m_i der Dimension des Wrenches – in diesem Fall drei – und μ dem Reibungskoeffizienten. Die Bedingungen können zu einer Block-Diagonalmatrix der Form

$$P(w) = \text{Blockdiag}(P_1, \ldots, P_i, \ldots, P_k) \tag{3.38}$$

zusammengefasst werden[2]. So eine Matrix ist genau dann symmetrisch und positiv (semi) definit, wenn alle Blöcke $P_i,\ i = 1, \ldots, k$ symmetrisch und positiv (semi) definit sind. Wenn die Matrix positiv semi definit ist, so gilt, dass alle Bedingungen erfüllt sind. Da für das Reibungsmodell gilt, dass P_i linear und symmetrisch ist, kann das Problem als *LMU* (Lineare-Matrix-Ungleichheit) formuliert werden:

$$\begin{aligned} P_i &= \sum_{j=1}^{m_i} w_{ij} S_{ij} \\ &= w_{i1} S_{i1} + \cdots + w_{im_i} S_{im_i} \succeq 0 \end{aligned} \tag{3.39}$$

mit $S_{ij},\ j = 0, \ldots, m_i$ als Faktoren für die LMU in Form von symmetrischen Matrizen und $\succeq$ als Zeichen für positive Semi-Definitheit. Als Beispiel für einen Kontakt mit Reibung sei

$$\begin{aligned} S_{i1} &= E^3_{13} + E^3_{31} \\ S_{i2} &= E^3_{23} + E^3_{32} \\ S_{i3} &= \mu_i (E^3_{11} + E^3_{22} + E^3_{33}), \end{aligned}$$

wobei E^a_{bc} eine quadratische Matrix der Dimension a ist, bei der alle Einträge gleich Null sind, bis auf den Eintrag $(b, c) = 1$. Da P blockdiagonal mit den $P_i^{'}$ auf der Hauptdiagonalen ist, kann es in einer LMU zusammengefasst werden:

$$P(x) = \sum_{l=1}^{m} w_l S_l \succeq 0 \tag{3.40}$$

wobei w_{ij} durch w_l und $S_l = Blockdiag(0, \ldots, 0, S_{ij}, 0, \ldots, 0), l = 1, \ldots, m$

Um die Kräfte exakt berechnen zu können, muss auch die Geometrie des verwendeten Manipulators mit berücksichtigt werden. Geht man davon aus, dass nur Präzisionsgriffe ausgeführt werden, so sollte es durchaus möglich sein, die Kräfte an den Kontaktpunkten ohne Berücksichtigung der Manipulatorstruktur zu berechnen. Sollen aber, wie in diesem Fall, alle möglichen Typen von Griffen berechnet werden, so muss die Struktur berücksichtigt werden (siehe auch [Murray et al. 1994] und [Prattichizzo and Bicchi 1998]). Bei einem gekoppelten System beeinflussen die Kräfte, die z.B. am ersten Fingerglied wirken, auch die Kräfte, die am letzten Fingerglied wirken, und sie sind somit nicht mehr von einander unabhängig. Dies geschieht mittels der *Jacobi-Matrix*. Die Jacobi Matrix ist eine Matrix von partiellen Ableitungen, die genutzt wird, um Gelenkgeschwindigkeiten in Beziehung zu der Geschwindigkeit des Endeffektors eines Roboters zu setzen. Die Transponierte kann aber auch benutzt werden, um die Beziehung zwischen den Kräften, die am Endeffektor wirken, in Beziehung zu den Drehmomenten der Gelenke zu setzen. Mehr Details zu der Jacobi Matrix sind in [Murray et al. 1994] und [Craig 2005] zu finden.

Analog zu den Bedingungen für die Reibung aus Gleichung 3.8 kann auch eine Bedingung für die Wechselwirkung zwischen Kontakt-Wrench und Manipulator laut [Han et al. 2000] definiert werden:

$$\begin{aligned} J^T w + g_{ext} - \tau^L &\geq 0 \\ -J^T w - g_{ext} + \tau^U &\geq 0 \end{aligned} \tag{3.41}$$

[2] Eine Block-Diagonalmatrix ist eine quadratische Matrix der Form $A = \begin{bmatrix} B_1 & 0 & \cdots & 0 \\ 0 & B_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B_n \end{bmatrix}$ mit B_i als quadratischer Matrix.

mit J^T der transponierten Jacobi-Matrix, w dem entsprechenden Wrench, g_{ext} einer externen Kraft, die ausgeglichen werden muss und τ^L, τ^U dem unteren bzw. oberen Limit für die Drehmomente. Dies kann auch als

$$\mathcal{T} = \left\{ w \in \mathbb{R}^m | \tau^L \leq J_h^T w + \tau_{ext} \leq \tau^U \right\} \tag{3.42}$$

geschrieben werden. Die LMUs zu Gleichung 3.41 sind gegeben durch

$$T^L(w) = \operatorname{diag}(J_h^T w + g_{ext} - \tau^L) = T_0^L + \sum_{l=1}^{m} T_l^L w_l \succeq 0 \tag{3.43}$$

$$T^U(w) = \operatorname{diag}(-J_h^T w - g_{ext} + \tau^U) = T_0^U + \sum_{l=1}^{m} T_l^U w_l \succeq 0. \tag{3.44}$$

Diese können wieder zu einer LMU zusammengefasst werden:

$$T(w) = \text{Blockdiag}\left(T^L(w), T^U(w)\right) = T_0 + \sum_{l=1}^{m} T_l w_l \succeq 0 \tag{3.45}$$

mit $T_l = \text{Blockdiag}(T_l^L, T_l^U),\ l = 0, \ldots, m.$

Kraftoptimierung

Ist nun ein Griff G mit den Reibungsbedingungen FC, den Randbedingungen $\mathcal{T}$ für die Gelenkdrehmomente und ein Objekt-Wrench W gegeben, so kann eine Lösung für die optimalen Griffkräfte x gefunden werden durch

$$\mathcal{A}_x = \{x \in \mathbb{R}^m | P(x) \succeq 0, T(x) \succeq 0, Gx = W\}. \tag{3.46}$$

Das Optimalitätskriterium ist definiert als

$$\Psi(x) = f_n^T x + \log\det P^{-1}(x) \tag{3.47}$$

mit $f_n = [f_{n_1}^T \cdots f_{n_i}^T \cdots f_{n_k}^T]^T \in \mathbb{R}^m$ als Vektor der Normalen-Komponenten der Griffkraft x. n_i kann beispielsweise für einen Kontakt mit Reibung definiert werden als $f_{n_i} = [0\ 0\ d_i]^T$ mit $d_i \geq 0$. Der erste Term wird genutzt, um die Kräfte entlang der Kontakt-Normalen zu kontrollieren und der zweite Term wächst gegen Unendlich, falls eine der Kontaktkräfte die Grenzen des Reibungskegels erreicht. Das Optimierungsproblem kann nun geschrieben werden als

$$\arg\min_{x \in \mathcal{A}_x} \Psi(x). \tag{3.48}$$

Da das System noch in Abhängigkeit von x definiert ist, müssen noch einige Umformungen vorgenommen werden, um die internen Kräfte λ optimieren zu können. Die Transformation des LMU Problems ist

$$\tilde{P}(\lambda) := P(N\lambda) = \sum_{l=1}^{m-6} \lambda_l \tilde{S}_l \succ 0 \tag{3.49}$$

$$\tilde{S}_l = \sum_{i=1}^{m} n_{il} S_i \tag{3.50}$$

mit n_{il} als Element des Null-Vektorraum von N in Zeile i und Spalte l[3]. Die Menge der Lösungen für λ ist gegeben durch

$$\mathcal{A}_\lambda = \{\lambda \in \mathbb{R}^{m-r} | \tilde{P}(\lambda) \succeq 0, \tilde{T}(\lambda) \succeq 0\}, \tag{3.51}$$

mit Optimalitätsmaß

$$\tilde{\Psi}(\lambda) := \Psi(G^+ W + N\lambda) \tag{3.52}$$

wird das Problem zu

$$\arg \min_{\lambda \in \mathcal{A}_\lambda} \tilde{\Psi}(\lambda) \tag{3.53}$$

Dies ist ein Optimierungsproblem, dass sich z.B. mit dem *maxdet* Software-Packet [Wu et al. 1996] lösen lässt.

[3]Der Null-Vektorraum (engl. null space) ist der Raum der alle Lösungen für v enthält so dass $Av = 0$ gilt.

Kapitel 4

Evaluation von Griffen

Die Qualität eines Griffs ist abhängig von den Eigenschaften des Greifers, der Art des Griffs und des Objekts. Aus diesen Eigenschaften lassen sich Qualitätsmaße erstellen, die für die Evaluation von Griffen geeignet sind. Hierzu gibt es in der Literatur verschiedene Möglichkeiten. Diese werden in diesem Kapitel zunächst dargestellt und erläutert. Wie gezeigt wird, weisen die vorgestellten Verfahren den Mangel auf, dass sie nicht auf die Semantik des zu greifenden Objekts, das heißt dessen Verwendungszweck, eingehen. Daher wird ein neues Qualitätsmaß für Griffe entwickelt. Abschließend wird dieses Qualitätsmaß mit einer Auswahl von Griffen evaluiert.Wie die Griffe generiert werden, wird in Kapitel 6 beschrieben.

4.1 Qualitätsmaße zur Evaluation von Griffen

Eines der grundlegenden Maße für die Evaluation von Griffen ist das Kraftschluss-Kriterium für einen Griff, das bereits in Kapitel 3.2 beschrieben wurde. Der Definition aus [Nguyen 1988] folgend, erfüllt ein Griff dieses Kriterium, wenn die Kräfte, die an einem der Kontaktpunkte auftreten, ausbalanciert werden können. Des Weiteren ändern die Finger des Manipulators ihre Position bei Krafteinwirkung nicht. Es existieren verschiedene Möglichkeiten, einen Griff auf Kraftschluss zu testen.

Das bekannteste Verfahren für die Evaluation ist das in [Ferrari and Canny 1992] vorgestellte und in Kapitel 3.2.3 erörterte Verfahren. Hierbei wird die konvexe Hülle über dem Grasp-Wrench-Space (GWS) aufgespannt und analysiert. Der GWS ist ein sechsdimensionaler Raum, der über die Kräfte und Drehmomente definiert ist, die bei einem Griff auftreten. Ferrari und Canny beschreiben zwei Qualitätsmaße, die mit einem solchen GWS gemessen werden können. Zum einen kann die maximale Kraft an einem Finger (unabhängig) minimiert werden, und zum anderen die Summe der Fingerkräfte. Letzteres ist auch in [Kirkpatrick et al. 1990] als *Quantitative Seinitz's Theorem* untersucht worden. Als Qualitätsmaß wird dabei der Radius der größtmöglichen L_2-Kugel definiert, die in der konvexen Hülle der Kontakt-Wrenches eingeschlossen werden kann. Dies wird in [Mishra 1995a] beschrieben (siehe auch Kapitel 3.2.3)[1].

In [Zhu et al. 2003, Zhu and Wang 2003] beschreiben die Autoren eine Möglichkeit, das Kraftschluss-Kriterium sowohl für planare als auch dreidimensionale Griffe ohne Reibung über die Lösung eines linearen Gleichungssystems zu bestimmen. Für dreidimensionale Griffe mit Reibung wird eine Lösung über ein nichtlineares System beschrieben, ohne das hierfür, wie bei vielen anderen Verfahren, die Approximation des Reibungskegels erforderlich ist. Zusätzlich wird ein neues Qualitätsmaß eingeführt, das aus dem in [Mishra 1995a] vorgestellten Verfahrens abgeleitet ist. Dabei wird L_2-Metrik durch eine Pseudometrik ersetzt, die sich durch eine effiziente Berechenbarkeit auszeichnet und einen Ansatz

[1] Eine L_2- Kugel ist eine Kugel die über die L_2-Norm definiert ist. Analog dazu ist der Begriff der L_2-Metrik definiert.

zur einfachen Griffgenerierung bietet. Ein entsprechender Algorithmus, der alle Griffe, die über einem definierten Schwellwert des Kriteriums liegen, berechnet, wird ebenfalls in [Zhu et al. 2003] definiert.

In [Liu et al. 2004b] werden drei Problemdefinitionen für die Qualitätsbestimmung eines Griffs eingeführt. Zunächst wird die Grasp-Map als Transferfunktion verwendet, um die Kräfte der Finger in Wrenches in der Domäne der Reibungskegel zu übertragen. Somit können optimale Kontaktpunkte für einen möglichst stabilen Griff über die Optimierung der Transferfunktion gefunden werden. Das zweite Problem besteht darin, die maximal möglichen Kräfte an den Kontakt-Normalen zu finden, die durch eine Störung schlimmstenfalls aufgebracht werden dürfen. Das Problem des *analytischen Zentrums* ist der dritte Teil der vorgeschlagenen Funktionen. Mit Hilfe der Grasp-Map können Punkte gefunden werden, die möglichst weit von den Grenzen der Reibungskegel entfernt liegen und somit Stabilität garantieren. Die Autoren beschränken sich auf die Anwendung des Verfahrens auf runde bzw. ellipsoide Objekte.

In [Pollard 2004] und [Pollard 1994] wird die Qualität eines Griffs als die

> „ ...reziproke Summe der Beträge der Kräfte entlang der Kontakt-Normalen [...], die schlimmstenfalls bei einer Aufgabe auftreten können ...“

definiert. Eine *Aufgabe* wird dabei als eine Menge von Wrenches definiert, also eine Kombination aus Kräften und Drehmomenten, die bei einem Griff auftreten. Weiterhin werden Kriterien für *teilweise kraftschlüssige Griffe* gegeben. Diese sind nicht mehr in der Lage, allen auftretenden Kräften zu widerstehen, bieten aber dennoch eine hinreichende Stabilität, so dass viele Operationen mit dem Griff ausgeführt werden können.

In [Kim et al. 2004b] wird von den Autoren eine Kombination aus verschiedenen Indikatoren als Qualitätsmaß für einen Griff vorgeschlagen. Die Indikatoren sind ein Stabilitäts-Index, der als quantitatives Maß die Widerstandsfähigkeit gegenüber externen Kräften bewertet. Des Weiteren ein Unsicherheits-Index, der über den Abstand der Greifpunkte zum Mittelpunkt der jeweiligen Kontaktfläche definiert ist. Ein weiterer Index, der die maximal mögliche Kraftübertragung mit einem Griff beschreibt. Ein Aufgaben-Isotropie-Index, der einen Zusammenhang zwischen der Konfiguration jedes Fingers und den Manipulationsmöglichkeiten des Objekts herstellt und einem Index, der die Steifigkeit bzw. die Nachgiebigkeit eines Griffs beschreibt. Da alle Indizes eine unterschiedliche physikalische Bedeutung haben, werden sie nach einer Normalisierung zu einem *undimensionierten* Index zusammengefasst. Die Maximierung dieses Index führt laut den Autoren zu einem bestmöglichen Griff.

Qualitätsmaße auf Basis von Task-Wrench-Spaces

Ebenfalls von Pollard wird in [Pollard 1994] eine Methode für die Berechnung eines *Objekt-Wrench-Space* (OWS) vorgeschlagen. Der OWS beschreibt dabei den bestmöglichen Griff, der für ein Objekt gefunden werden kann. Zur Berechnung bezieht die Methode die Objektgeometrie mit ein, da Kräfte und Drehmomente in der Regel an der Objektoberfläche wirken. Der Objekt-Wrench-Space ist die Vereinigung aller Grasp-Wrench-Spaces des Objekts. Der Faktor, der den Objekt-Wrench-Space skaliert, so dass er innerhalb des L_1-Wrench-Space eines Griffs liegt, wird von Pollard dabei als Qualitätsmaß vorgeschlagen (für Details zum L_1-Wrench-Space siehe Kapitel 3.2.3).

Li und Sastry schlagen vor, einen *Task-Wrench-Space* in Form von Ellipsoiden zu definieren [Li and Sastry 1987]. Ein Task-Wrench-Space ist dabei ein Wrench-Space, der über die Wrenches definiert wird, welche die Anforderungen an einen Griff definieren. Die Qualität eines Griffs kann über die Parameter des Ellipsoiden beschrieben werden. Das Kriterium minimiert hierzu die Summe der Beträge

der Kräfte, die erforderlich sind, um den schlimmstmöglichen *Task-Wrench* auszugleichen, der innerhalb des Ellipsoiden liegt.

Da die Qualitätsmaße auf der Basis von Grasp-Wrench-Spaces aufgrund von Objekt-Skalierung, Referenzpunkt und Skalierung der Drehmoment-Achsen nach [Borst et al. 2004] nicht miteinander vergleichbar sind, stellen die Autoren ein Verfahren vor, das ohne eine Diskretisierung der Reibungskegel auskommt. Denn auch hierbei wird laut Borst et al. eine Fehlerquelle mit in die Berechnung eingebracht. Ist die mit dem Griff auszuführende Aufgabe unbekannt, wird normalerweise der größtmögliche Radius einer Kugel, die noch innerhalb der konvexen Hülle des Wrench-Space liegt, als Qualitätsmaß genommen. Dies ist aber laut den Autoren nachteilig, da es hierfür keine physikalische Interpretation gibt, da Kräfte und Drehmomente an den Objektgrenzen auftreten und selten über das gesamte Objekt gleich verteilt sind, wie es die Verwendung einer L_2-Kugel als Qualitätsmaß impliziert. Daher kombinieren die Autoren den Object-Wrench-Space, wie er in [Pollard 1994] definiert ist, mit dem in [Li and Sastry 1987] beschriebenen Ansatz. Der Grasp-Wrench-Space repräsentiert dabei die Möglichkeiten eines Griffs, externen Kräften zu widerstehen, während ein Objekt-Wrench-Space die Kräfte, die möglicherweise bei einem Griff auftreten können, beschreibt. Die Qualität für einen Griff beschreibt der größtmögliche Faktor, mit dem der Objekt-Wrench-Space skaliert werden kann, so dass er noch in den Grenzen des Grasp-Wrench-Space liegt. Somit ist das Maß laut der Autoren unabhängig vom gewählten Referenzpunkt und der Größe der Objekte.

[Haschke et. al. 2005] schlagen einen Ansatz für ein aufgabenorientiertes Qualitätsmaß vor. Auch hierbei werden die Anforderungen an den Griff über einen Wrench (Task-Wrench) definiert. Der Griff wird jedoch nicht auf Kraftschluss geprüft, sondern darauf, ob er in der Lage ist, den Task-Wrench auszugleichen. Dieser Wrench wird dabei als virtueller Kontakt mit in die Berechnung des GWS eingebracht. Das Problem wird schließlich als lineare Matrix-Ungleichung aufgefasst und gelöst. Zwar lässt sich eine Aufgabe oftmals über einen Wrench definieren, jedoch werden hierdurch noch Störungen, in Form von weiteren Kräften, ausgeübt, die natürlich Einfluss auf den Griff haben und zur Instabilität führen können. Daher wird von den Autoren noch eine Erweiterung des Task-Wrench-Space für konvexe Polytope angegeben[2].

Verfahren für die Berechnung von Griffkräften

Buss et al. haben in [Buss et al. 1996] festgestellt, dass die Randbedingungen, die sich durch die Reibungskegel an den Kontaktpunkten ergeben und die Randbedingungen für den Kräfteausgleich bei einem Griff äquivalent zu einer positiv definiten Matrix mit linearen Bedingungen sind[3]. Mit dieser Arbeit haben sie den Grundstein für eine Reihe weiterer Entwicklungen in dem Bereich der Kraftoptimierung gelegt. Die Optimierung der Griffkräfte erfolgt über ein Gradienten-Abstiegsverfahren.

Die in [Han et al. 2000] beschriebene Arbeit basiert auf den Grundlagen aus [Buss et al. 1996]. Das Problem der Griffevaluation wird dabei in drei Teilaspekte zerlegt: erstens in die Bestimmung des Kraftschlusses, zweitens die Bestimmung der Kräfte für den Griff und drittens in die Optimierung der Kräfte. Für die Lösung des zweiten Problems müssen die Kinematik der Hand und die Grenzen der Aktuatoren mit in Betracht gezogen werden. Eine Lösung für die drei vorgenannten Probleme wird in Form von konvexen Optimierungsproblemen unter Zuhilfenahme von *Linearen-Matrix-Ungleichungen* (*LMU*) gelöst. Hierzu werden die Randbedingungen, die sich durch die Reibungskegel ergeben, in LMUs ge-

[2] Der Begriff Polytop bezeichnet in der Geometrie ein verallgemeinertes Polygon in beliebiger Dimension.

[3] Ein Definitheitskriterium lässt sich über die Eigenwerte definieren. Demzufolge ist eine Matrix positiv definit, wenn alle Eigenwerte größer als Null sind.

fasst. Im Gegensatz zu der Arbeit von Buss et al. muss somit für die Lösung des Problems keine Kraft für einen Griff vorgegeben werden, die die Bedingungen der Reibungskegel erfüllt und einen entsprechenden Objekt-Wrench definiert.

4.2 Qualitätsmaß für Griffe auf Basis der Weiterverwendbarkeit

Die in dem vorhergehenden Abschnitt vorgestellten Qualitätsmaße für die Evaluation von Griffen basieren auf Kräften. So werden z.B. die Kräfte, die erforderlich sind, um ein Objekt mit einem bestimmten Griff zu halten, berechnet und bezüglich einer erforderlichen Minimalkraft oder eines Kräftegleichgewichts optimiert. In anderen Fällen wird ein Task-Wrench-Space definiert, der Anforderungen, die an einen Griff gestellt werden, in Form eines Wrench-Spaces definiert. Ein weiterer Ansatz besteht darin, Anforderungen nicht als einen Wrench-Space, sondern als eine Kraft, die den Greifkräften entgegenwirkt, zu definieren. Aber die Nutzung von Kräften nicht erfolgt intuitiv und ist für einen Laien mit geringen Kenntnissen in Physik und Mechanik nur schwer verständlich. Im Kontext der Service-Robotik sollte jeder, der mit einem Roboter interagiert, in der Lage sein, die Anforderungen an einen Griff zu formulieren, den dieser Roboter ausführen soll. Dies muss in einer Art und Weise geschehen können, die jedem, auch technisch nicht versierten Menschen, möglich ist.

Dass die *Semantik* eines Griffs nicht beachtet wird, sondern nur die Kräfte optimiert werden, ist ein weiterer Nachteil, den die bisher aufgeführten Verfahren aufweisen. So ist z.B. der Griff eines Bechers von oben, wie in Abbildung 4.1 dargestellt, möglicherweise sehr stabil und auch gut geeignet, um den Becher an Position *A* aufzunehmen und an Position *B* abzusetzen. Möchte man jedoch etwas in den Becher einschenken oder etwas ausgießen, so ist dieser Griff gänzlich unbrauchbar. Dem Griff fehlt eine Semantik in Bezug auf eine mögliche *Folgeoperation* , also in Bezug auf die *Weiterverwendbarkeit* des gegriffenen Objekts.

Ein weiteres Beispiel ist der Griff eines Bechers von der Seite, wie in Abbildung 4.2 dargestellt. Hierbei kann im Gegensatz zum Griff aus Abbildung 4.1 noch etwas in den Becher eingeschenkt werden. Weiterhin ist es möglich, den Becher aufzunehmen und umzusetzen, aber soll der Becher übergeben werden, schränken sich die Angreifpunkte für eine Übernahme stark ein. Hierbei wäre es praktisch nur möglich, die Tasse von unten zu greifen. Somit hat der Griff von der Seite in Bezug auf die Weiterverwendbarkeit eine andere Semantik als der Griff von oben. Es wird deutlich, dass die Griffe, je nachdem zu welchem Zweck die Objekte gegriffen worden sind, unterschiedlich bewertet werden müssen.

Der Forderung folgend, dass jeder Mensch in der Lage sein soll, einen Service-Roboter zu bedienen, muss daher ein Qualitätsmaß entwickelt werden, dass einen Griff nach seiner Weiterverwendbarkeit bewertet. Basis für die Bewertung der Weiterverwendbarkeit ist eine vom Benutzer anzugebende *Folgeoperation* bzw. eine Menge von Operationen, die nach dem Greifen eines Objektes ausgeführt werden soll. Hierbei ist es sinnvoll, nicht dem Benutzer die Definition der nachfolgenden Operationen als Gesamtes zu überlassen, sondern sie durch generische Primitive zu beschreiben, die der Benutzer seinen Bedürfnissen entsprechend anpassen kann.

4.2.1 Definition von Folgeoperationen

Die gebräuchlichsten Operationen für Gegenstände in einer Büroumgebung, für die der Service-Roboter konzipiert ist, sind: etwas in eine Tasse oder Becher einschenken, etwas aus einer Tasse, Flasche oder einem Becher ausschütten, das Aufnehmen eines Objektes, um es an jemanden zu übergeben sowie eine

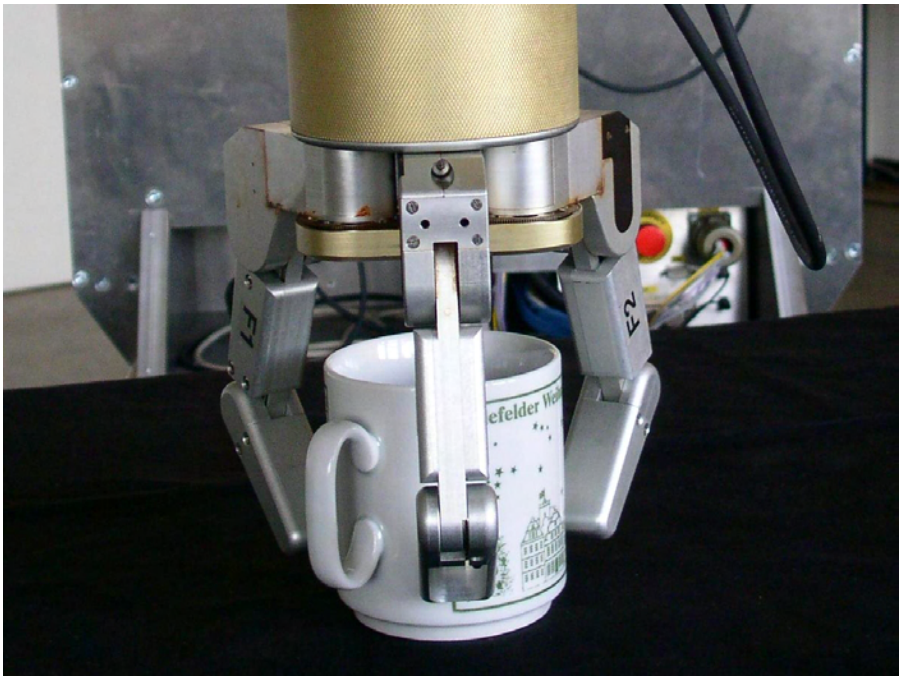


Abbildung 4.1: Der Griff einer Tasse von oben mit der BarrettHand.

bestimmte Bewegung mit einem Objekt ausführen. Diese Operationen können vom Benutzer für einen Griff priorisiert werden und das System muss in der Lage sein, anhand dieser Priorisierung einen entsprechenden Griff zu bestimmen. Die einzige Operation, die hierbei weitere Benutzereingaben erforderlich macht, ist die Operation *Bewegung*: Da ein System kaum in der Lage ist, a priori die Bewegung, die mit einem Objekt ausgeführt werden soll, zu bestimmen – mit Ausnahme eines Werkzeugs vielleicht – muss hierbei der Benutzer zwingend weitere Angaben machen. Diese erfolgen jedoch auch in Form von Bewegungsprimitiven, die lediglich vom Benutzer modifiziert werden müssen. Komplexe Bewegungen lassen sich ebenfalls über diesen Mechanismus generieren, indem einfache Bewegungen überlagert werden können.

Für die Berechnung der möglichen Folgeoperationen müssen einige Details zum jeweiligen Objekt a priori bekannt sein. So ist es z.B. nicht möglich, Angreifpunkte für eine mögliche Übergabeoperation zu bestimmen, wenn die geometrische Form eines Objekts völlig unbekannt ist. Ähnliches gilt für die Berechnung, ob in ein Objekt etwas eingefüllt werden kann. Hierfür muss das Volumen des Objektes bzw. das aufzunehmende Gewicht bekannt sein. Es müssen also je nach Art der zu berechnenden Operationen verschiedene Objektmerkmale bekannt sein. Diese Objektmerkmale werden in einer entsprechenden Objektstruktur gespeichert und einer Simulation zur Verfügung gestellt. Mittels dieser werden die entsprechenden Berechnungen für die jeweiligen Operationen durchführt und visualisiert. Details zu der Objektstruktur, der Modellierung der Objekte sowie der Simulation sind in Kapitel 6.2 beschrieben.

Es ergeben sich vier *Folgeoperationen*, die dem Benutzer als generische Primitive zur Verfügung stehen:

- Einschenken
- Ausgießen

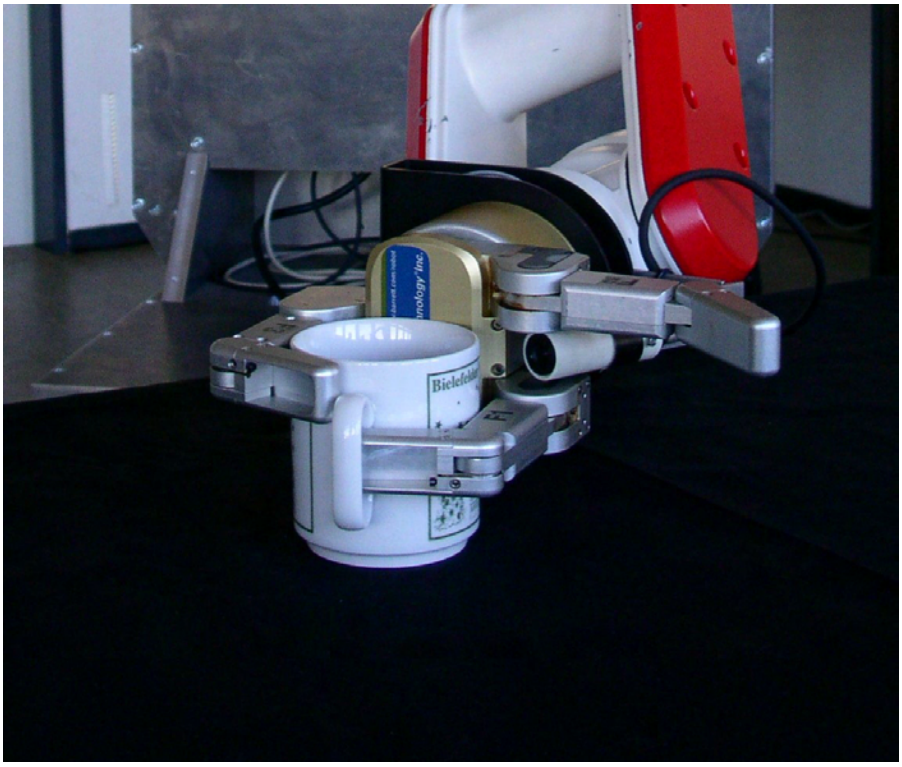


Abbildung 4.2: Der Griff einer Tasse von der Seite mit der BarrettHand.

- Übergabe
- Bewegung

Sie werden im Folgenden definiert und ihre Berechnungsvorschriften festgelegt. In diesem Zusammenhang werden auch die Berechnungsverfahren der einzelnen Operationen erörtert. Alle Operationen haben gemein, dass der Griff stabil sein muss, damit eine Operation als durchführbar gewertet wird.

4.2.2 Folgeoperation Einschenken

Die Folgeoperation *Einschenken* ist im Wesentlichen durch die folgenden zwei Eigenschaften definiert:

1. Die Einfüllöffnung des Gefäßes muss frei sein, d.h. sie darf nicht durch Teile der Hand verdeckt sein.
2. Der Griff muss in der Lage sein, der zusätzlichen Gewichtskraft, die durch das Einfüllen auftritt, zu widerstehen.

Die Einfüllöffnung muss dabei nicht extra definiert werden, sondern es wird davon ausgegangen, dass sie am äußeren Rand entlang der Z-Achse des Objekts liegt (Abb. 4.2.2). Es können aber auch andere Positionen für die Öffnung mit in das Objektmodell eingebracht werden (siehe auch Kapitel 6.2.2.1). Die Evaluation der Bedingungen für die Operation teilt sich in die folgenden Schritte auf. Zunächst wird

überprüft, ob der Griff in der Lage ist, dem zusätzlichen Gewicht standzuhalten. Dies erfolgt, indem zu der Gewichtskraft des Objektes die Gewichtskraft des Nutzvolumens des Objektes addiert wird. Hierbei wird die Annahme getroffen, dass in der Regel Flüssigkeiten mit einer Dichte von 1.0 g/cm^3 in ein Objekt eingefüllt werden[4]. Soll ein Stoff mit einer deutlich geringeren oder deutlich größeren Dichte eingefüllt werden, so muss dies bei der Berechnung mit berücksichtigt werden. Mit dieser neuen Gewichtskraft wird anschließend der Grasp-Wrench-Space neu berechnet und die Kräfte an den einzelnen Kontaktpunkten bestimmt. Ist die Berechnung des GWS erfolgreich und liegen die Kräfte innerhalb der Toleranzen des Objekts, erfolgt die Berechnung der geometrischen Randbedingungen. Ist die Berechnung nicht erfolgreich, so ist der Griff nicht stabil genug und somit das Einfüllen nicht möglich. Ist der Griff stabil genug, um das zusätzliche Gewicht der eingefüllten Flüssigkeit abzufangen, so wird anschließend geprüft, ob die Einfüllöffnung des Objekts nicht durch Teile der Hand verdeckt ist.

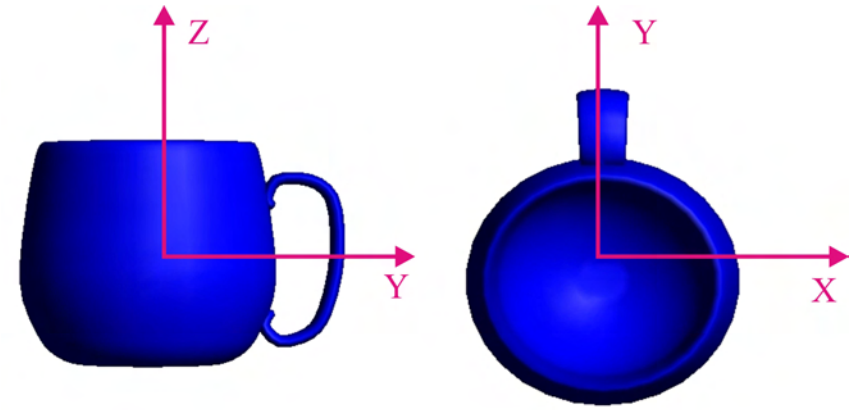


Abbildung 4.3: Koordinatensystem der Objekte am Beispiel einer Tasse: die Z-Achse entlang der Objekt-Längsachse, die Y-Achse entlang des Griffs, die X-Achse entsprechend eines rechtshändigen Koordinatensystems.

Dies wird in der Simulation getestet, indem ein Zylinder mit dem Durchmesser des Gegenstandes um die Einfüllöffnung gelegt wird. Kollidiert dieser hierbei mit Teilen der Hand, so wird angenommen, dass zumindest ein Teil der Öffnung verdeckt ist und der Griff wird als unbrauchbar zurückgewiesen.

Formal kann dies mit Hilfe von Topologien beschrieben werden: Sei $\mathcal{O}$ der von dem Objekt okkupierte Raum, $\mathcal{T} \subseteq \mathcal{O}$ sei eine Teilmenge von $\mathcal{O}$, die die Fläche der Einfüllöffnung inklusive eines Sicherheitsabstandes definiert. Sei $\mathcal{X}$ das Volumen, das erforderlich ist, um den Raum um die Einfüllöffnung zu bedecken und $\mathcal{H}$ das Volumen, das von dem Manipulator (in diesem Fall der BarrettHand) eingenommen wird. Dann kann der Freiraum um die Einfüllöffnung definiert werden als:

$$\mathcal{X} \cap \mathcal{O} = \mathcal{T} \ \wedge \ (\mathcal{X} \cap \mathcal{H} = \emptyset). \tag{4.1}$$

Anschaulich ist dies in Abbildung 4.2.2 illustriert. $\mathcal{H}$ ist der Raum, den die BarrettHand einnimmt, $\mathcal{O}$ beschreibt das gesamte Objekt und $\mathcal{T}$ definiert die Fläche der Einfüllöffnung mit einem zusätzlichen Sicherheitsabstand, entspricht also bei dem Becher einem Kreis. Da $\mathcal{X}$ das Volumen ist, das erforderlich ist, um den Raum um die Einfüllöffnung zu bedecken, ist es bei der Tasse ein Zylinder. Die Höhe des Zylinders ist in diesem Fall auf 80 mm festgelegt worden.

[4] $1.0g/cm^3$ oder auch $1.0g/ml$ entspricht der Dichte von Wasser bei $3.98°$C und stellt somit eine gute Näherung für wässrige Flüssigkeiten dar.

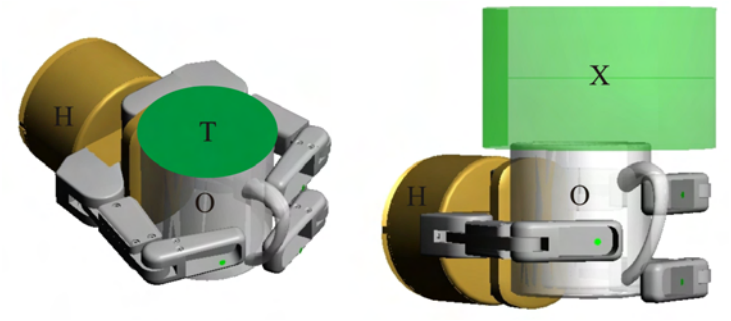


Abbildung 4.4: Visualisierung der Bedingungen der Folgeoperation Einschenken. $\mathcal{H}$ ist der Raum, den die BarrettHandeinnimmt, $\mathcal{O}$ beschreibt den Becher und $\mathcal{T}$ definiert die Fläche der Einfüllöffnung. $\mathcal{X}$ definiert das Volumen, dass erforderlich ist, um den Raum um die Einfüllöffnung zu bedecken.

Die zusätzliche Gewichtskraft, die durch die eingefüllte Flüssigkeit aufgebracht wird, wird in die Berechnung der Kräfte an den Kontaktpunkten mit eingebracht. Dies erfolgt, indem sie zu der Gewichtskraft des Objekts $\vec{g}_{ext}$ in Formel 3.41 addiert wird. $\vec{g}_{ext}$ ist dann definiert als:

$$\vec{g}_{ext} = R \cdot (-m \cdot \vec{g} + V \cdot \rho\vec{g}) \tag{4.2}$$

mit m der Masse des Objekts, $\vec{g}$ der Erdbeschleunigung (9.81 m/s^2) entlang der Welt-Z-Achse, V dem Volumen und ρ der Dichte des eingefüllten Stoffs (1 g/cm^3). R definiert die Rotation des Objekts in Bezug zum Weltkoordinatensystem.

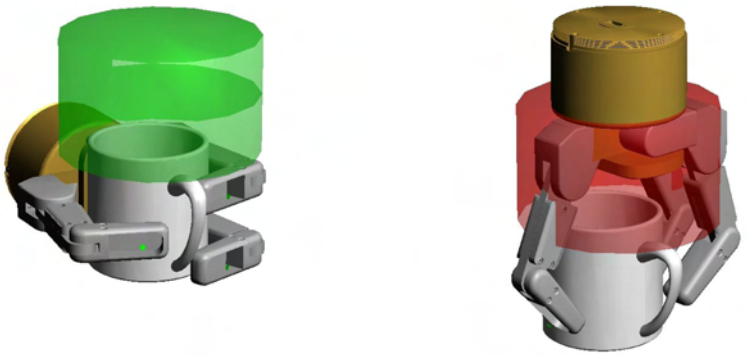
Beispiele für entsprechende Griffe und ihre Visualisierung sind in Abbildung 4.5 dargestellt. Bei einem Griff der Tasse von der Seite (Abb. 4.5(a)) ist die Einfüllöffnung nicht verdeckt, was durch den grün gefärbten Zylinder visualisiert wird. Bei einem Griff von oben ist die Öffnung verdeckt. Dies wird mittels eines rot gefärbten Zylinders dargestellt.

In Tabelle 4.1 sind die Kräfte, die an den Kontaktpunkten entlang der Z-Achse wirken, für die in Abbildung 4.5 gezeigten Griffe aufgeführt. Die Kräfte wurden mittels einer Simulation berechnet, da, wie in Kapitel 2.3 gezeigt wurde, die Sensoren der BarrettHand nicht ausreichend sind um, die Kräfte aller Kontakte eines Griffs berechnen zu können. Detailliert ist die Simulation in Kapitel 6.2 beschrieben. Wie zu sehen ist, steigen die Kräfte deutlich an, wenn bei der Evaluation von einer gefüllten Tasse ausgegangen wird. Der F2-Finger der BarrettHand (siehe Kapitel 2.3) und die Basis müssen dabei die größte Kraft ausgleichen, da zum einen der F2-Finger den „tiefsten“ Kontaktpunkt stellt und somit der größte Teil des Gewichts auf ihm liegt, zum anderen, da er den Gegenpunkt zu allen anderen Kontakten bildet.

Kontaktpunkt	Basis	F1 distal	F1 proximal	F2 distal	F2 proximal	F3 proximal
Tasse leer	3,80 N	3,83 N	4,04 N	2,00 N	8,06 N	1,10 N
Tasse gefüllt	6,31 N	6,34 N	5,46 N	2,23 N	12,44 N	2,19 N

Tabelle 4.1: Kräfte entlang der Z-Achse bei einem Griff von der Seite.

Bei der Evaluation der Folgeoperation Einschenken wird davon ausgegangen, dass die Gewichtszunahme gleichmäßig um den Massenschwerpunkt des Objekts stattfindet, so dass sich dessen Lage durch



(a) Griff einer Tasse von der Seite. (b) Griff einer Tasse von oben.

Abbildung 4.5: Simulation der Operation Einfüllen. Das Einfüllen ist möglich, wenn die Tasse von der Seite gegriffen wird (a) und schlägt bei einem Griff von oben fehl (b).

das Einfüllen nicht verändert. Ebenso wird davon ausgegangen, dass der Einfüllvorgang als solcher keinen Einfluss auf den Massenschwerpunkt des Objekts hat. Dies ist in der Realität nicht der Fall, da das Einfüllen ein kontinuierlicher Vorgang ist, bei dem sich ein Gefäß nicht instantan füllt. Dieses zu simulieren würde die Simulation von Flüssigkeiten in Bewegung bedeuten und wäre mit einem System für die Berechnung von komplexen Dynamiken möglich, wird aber aufgrund der Komplexität hier nicht angewendet.

4.2.3 Folgeoperation Ausgießen

Ausgießen ist eine weitere Folgeoperationen, die mit einem Objekt ausgeführt werden kann. Auch diese Operation ist im wesentlichen durch zwei Merkmale gekennzeichnet:

1. Die Öffnung des Objektes muss frei sein.
2. Eine Rotationsbewegung, die letztendlich zum Ausgießen führt, muss durchführbar sein.

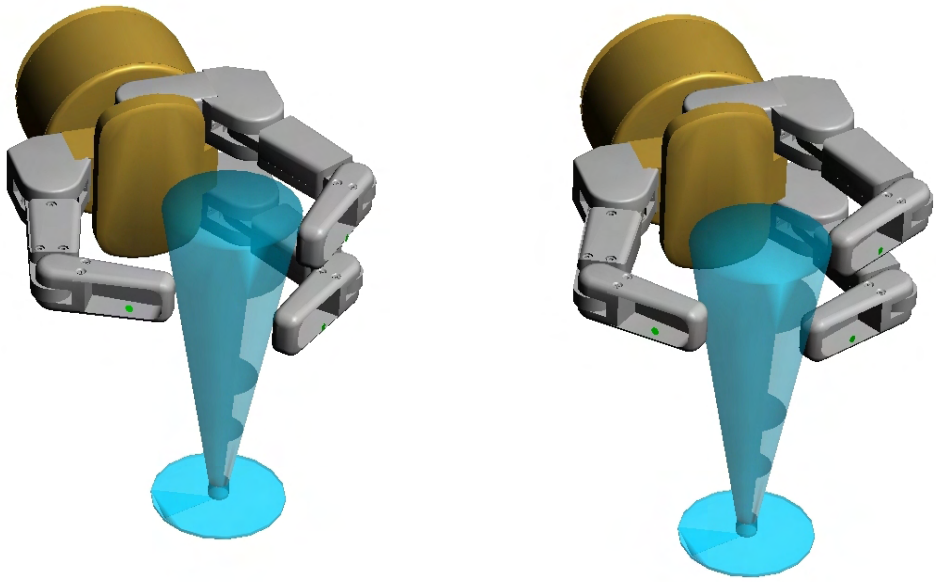
Formal kann die erste Bedingung wie bei der Operation Einschenken, also wie Gleichung 4.1, definiert werden. Für die zweite Bedingung muss die Gewichtsverteilung entsprechend der Rotationsbewegung angepasst werden. Somit ergibt sich Gleichung 4.2 zu:

$$\vec{g}_{ext} = B\ R\ (-m \cdot \vec{g} + V \cdot \rho \cdot \vec{g}) \tag{4.3}$$

wobei B die Rotation des Objekts relativ zum Weltkoordinatensystem beschreibt, die die Ausgießbewegung definiert. Dabei ist m die Masse des Objekts, $\vec{g}$ die Erdbeschleunigung (9.81 m/s^2), V das Volumen und ρ die Dichte des eingefüllten Stoffs (1 g/cm^3). R definiert die Rotation des Objekts in Bezug zum Weltkoordinatensystem. Die Rotation der Bewegung B kann beliebig definiert werden. In der Regel wird aber die Rotationsachse für die Bewegung orthogonal zur Normalen der BarrettHand definiert (also um

die Basis der BarrettHand), also bei Verwendung von TASER parallel zur Achse des sechsten Gelenks (vergleiche Abschnitt 2.2). Diese Bewegung ist analog zu der eines Menschen, die durch die Rotation des Handgelenks ausgeführt wird. Darüber hinaus ist diese Bewegung mit einem Roboterarm einfach zu realisieren und in den meisten Fällen (es sei denn, das Gelenk ist anderweitig blockiert) auch vom Roboter auszuführen.

Es wird, wie bei der Folgeoperation Einschenken, davon ausgegangen, dass die Gewichtsabnahme den Massenschwerpunkt des Objekts nicht beeinflusst, ebenso wie das Ausschütten als solches. Somit kann bei der Berechnung der Folgeoperation davon ausgegangen werden, dass die Gewichtsabnahme des Objektes in jedem Fall vom Griff kompensiert werden kann und somit für die Berechnung irrelevant ist.



(a) Griff 1. (b) Griff 2.

Abbildung 4.6: Griffe vor der Rotation zu den in Tabelle 4.2 aufgeführten Kräften. Die Griffe unterscheiden sich durch eine geringe Positionsänderung der Finger F1 und F2 der BarrettHand. Bei Griff 2 liegen die Kontaktpunkte näher an der Mittelachse des Objekts als bei Griff 1.

In Abbildung 4.6 sind zwei Präzisionsgriffe, Griffe mit nur einem Kontakt je Finger und keinem Kontakt an der Handbasis, für ein Sektglas dargestellt. Es wurde die Operation Ausschütten berechnet, mit einer Rotation entlang der Y-Achse der Hand um einen Winkel von $90°$. Hierbei wurden die Kräfte, die an den Kontaktpunkten erforderlich sind, vor und nach der Bewegung berechnet. Die Ergebnisse der Berechnung sind in Tabelle 4.2 zusammengefasst. Wie zu sehen ist, sind beide Griffe in beiden Fällen stabil. Obwohl die Griffe relativ ähnlich sind und die berechneten Kraftwerte für den Normalfall relativ ähnlich zueinander sind, führt die Evaluation zu unterschiedlichen Ergebnissen. Für Griff 2 (Abb. 4.6(b)) werden die maximal zulässigen Kräfte des Objekts deutlich überschritten (wenn man von einer maximal zulässigen Kraft von 8 N ausgeht). Somit wäre die Operation Ausgießen für Griff 1 (Abb. 4.6(a)) durchführbar und für Griff 2 (Abb. 4.6(b)) nicht durchführbar.

Kontaktpunkt	F1 proximal	F2 proximal	F3 proximal
Griff 1 (Abb. 4.6(a)) vor Rot.	3,60 N	1,90 N	5,23 N
Griff 1 (Abb. 4.6(a)) nach Rot.	7,80 N	1,37 N	7,25 N
Griff 2 (Abb. 4.6(b)) vor Rot.	2,58 N	2,56 N	4,94 N
Griff 2 (Abb. 4.6(b)) nach Rot.	16,82 N	15,74 N	30,12 N

Tabelle 4.2: Kräfte entlang der Z-Achse zu den Griffen aus Abbildung 4.6 vor und nach der Ausgieß-Bewegung.

4.2.4 Folgeoperation Übergabe

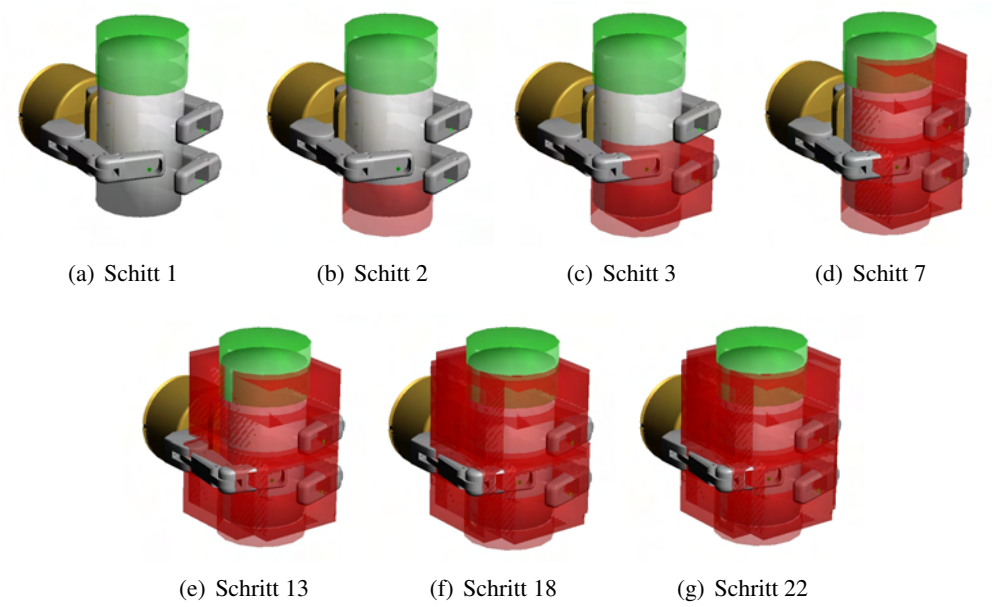
Die Folgeoperation *Übergabe* ist wahrscheinlich eine der am meisten gebrauchten Operationen im Zusammenhang mit einem Service-Roboter, da der Transport von Objekten von A nach B eine der klassischen Service-Aufgaben ist (Beispiel: Kaffee holen). Hierbei ist es natürlich unbedingt erforderlich, dass der Roboter die Objekte derart greift, dass er sie entweder geeignet abstellen oder übergeben kann. Für die Übergabe muss ein entsprechender Port bzw. Freiraum am Objekt vorhanden sein, so dass das Objekt dem Roboter abgenommen werden kann. Somit ist die Operation Übergabe durch nur ein Merkmal bestimmt (außer dass der Griff selbstverständlich stabil sein muss):

1. Freiraum am Objekt für eine Übergabe.

Es gibt zwei unterschiedliche Arten von Freiräumen, die evaluiert werden (Abb. 4.2.4): ein Zylinder wird benutzt, um den Freiraum oben und unten am Objekt zu definieren und eine Box für Griffe von der Seite. Damit ein Mensch noch die Möglichkeit hat, das Objekt vom Roboter entgegen zu nehmen, muss die Box eine bestimmte Mindestgröße aufweisen. Nimmt man an, dass die durchschnittliche Handschuhgröße 8 ist und diese einen Umfang von 203 mm aufweist (laut [HVBG 1995]), so kann man von einer durchschnittlichen Handbreite von 80 mm ausgehen. Wird außerdem angenommen, dass eine Hand im Durchschnitt ca. 35 mm dick ist, muss die Box mindestens eine Höhe von 95 mm aufweisen, damit eine menschliche Hand problemlos zugreifen kann. Die Breite der Box entspricht der Objektbreite, damit es immer vollständig umschlossen werden kann. Die Tiefe der Box ist auf 30 mm festgelegt worden, da davon ausgegangen wird, dass ein Angreifen mit der Fingerkuppe ausreichend ist. Der Zylinder, der den Freiraum für Griffe von oben bzw. von unten definiert, ist über den Durchmesser des Objekts und eine Mindesthöhe von 80 mm definiert.

Um die möglichen Angreifpunkte zu bestimmen, wird das Objekt segmentweise, mittels einer Breitensuche [Russel und Norvig 2003], nach Freiräumen in entsprechender Größe abgesucht. Kollidieren Teile der Umgebung oder der Hand mit dem Freiraum, so wird er gesperrt und das nächste Intervall untersucht. Um Schäden am Roboter und Verletzungen von Menschen ausschließen zu können, sind die Volumina, die für eine Übergabe erforderlich sind, relativ groß definiert worden. Wegen dieser Sicherheitsüberlegungen werden auch keine verzahnten Positionen, also Positionen bei, denen jemand zwischen die Finger der BarrettHand greift, evaluiert.

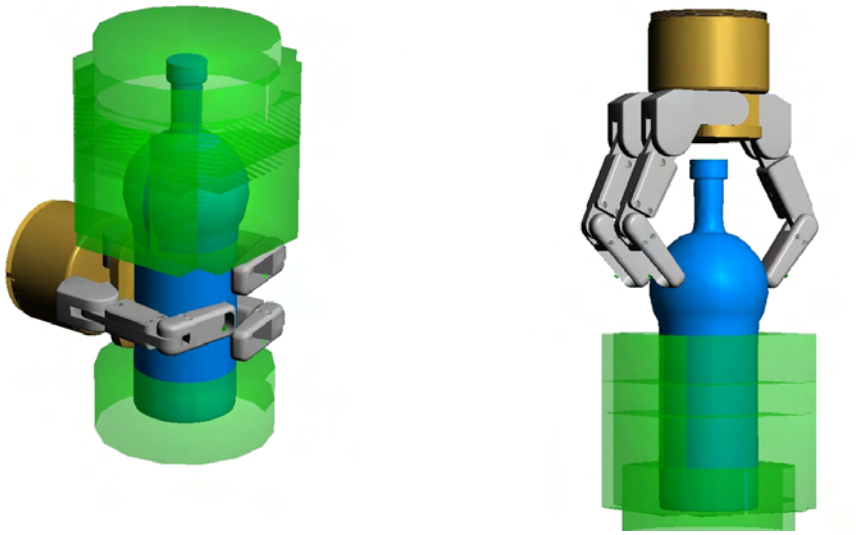
In Abbildung 4.8 sind zwei Beispiele für eine Übergabeoperation dargestellt. Bei einem Griff der Flasche von der Seite (Abb. 4.8(a)) ergeben sich sieben, bei einem Griff von oben (Abb. 4.8(b)) acht mögliche Positionen, an denen ein Partner die Flasche greifen kann. Die möglichen Angreifpunkte sind durch grün gefärbte Volumina gekennzeichnet. Die roten Markierungen an den Fingern sind Teile der Reibungsgkegel die ebenfalls in der Simulation visualisiert werden.



(a) Schitt 1 (b) Schitt 2 (c) Schitt 3 (d) Schitt 7

(e) Schritt 13 (f) Schritt 18 (g) Schritt 22

Abbildung 4.7: Die Berechnung der Folgeoperation Übergabe. Die Berechnung der möglichen Positionen erfolgt sequentiell. Zunächst werden die Übergabemöglichkeiten am oberen und unteren Ende des Objekts evaluiert. Anschließend werden die Übergabemöglichkeiten an den Flanken berechnet.



(a) Griff einer Flasche von der Seite. (b) Griff einer Flasche von oben.

Abbildung 4.8: Simulation der Operation Übergabe. Bei einem Griff von der Seite ergeben sich sieben mögliche Positionen für eine Übergabe (a), bei einem Griff von oben acht (b).

Eine Übergabe sollte auch möglich sein, indem das Objekt von oben bzw. von unten gegriffen wird. Dies wird ebenfalls über das Freiraumkriterium geprüft. Hierfür wird ein Zylinder mit dem Durchmesser des Objekts sowie einem Überstand von 10 mm und einer Höhe von 5 mm mit einer Überlappung von 30 mm am Kopfteil und am Ende des Objekts angesetzt. Anschließend wird über eine Kollisionserkennung ermittelt, ob der erforderliche Freiraum bereits okkupiert ist.

Sei $\mathcal{V}$ das Volumen, welches erforderlich ist, damit ein Gegenüber das Objekt greifen kann und sei $\mathcal{O}$ der von dem Objekt okkupierte Raum mit $\mathcal{T} \subseteq \mathcal{O}$ einer Teilmenge des Objekts und $\mathcal{H}$ der Manipulator. Formal ist die Folgeoperation Übergabe dann definiert durch:

$$\mathcal{T}_i \cap \mathcal{V}_i = \mathcal{V}_i \wedge (\mathcal{V}_i \cap \mathcal{H} = \emptyset), \tag{4.4}$$

da es möglicherweise mehrere Positionen für eine Übergabe gibt, existieren möglicherweise auch mehrere $\mathcal{H}_i$ und $\mathcal{T}_i$. Es gibt in diesem Fall keine rein binäre Entscheidung, ob die Folgeoperation durchgeführt werden kann oder nicht, sondern es wird darüber hinaus eine Aussage getroffen, an wie vielen unterschiedlichen Position das Objekt übergeben werden kann.

4.2.5 Folgeoperation Bewegung

Die Folgeoperation *Bewegung* wurde bereits als Bestandteil der Folgeoperation Ausgießen eingeführt. Zusätzlich zu der bereits möglichen Ausgießbewegung soll es dem Benutzer auch möglich sein, besondere Bewegungen für Objekte zu definieren, um überprüfen zu können, ob ein Griff den dabei auftretenden Kräften widerstehen kann. Diese Operation erfordert jedoch erheblich mehr Benutzereingaben bzw. Benutzerinteraktion als die bereits beschriebenen Kriterien, da genaue Angaben über die Bewegungstransformation erforderlich sind, um möglichst viele Aspekte einer Bewegung berücksichtigen zu können.

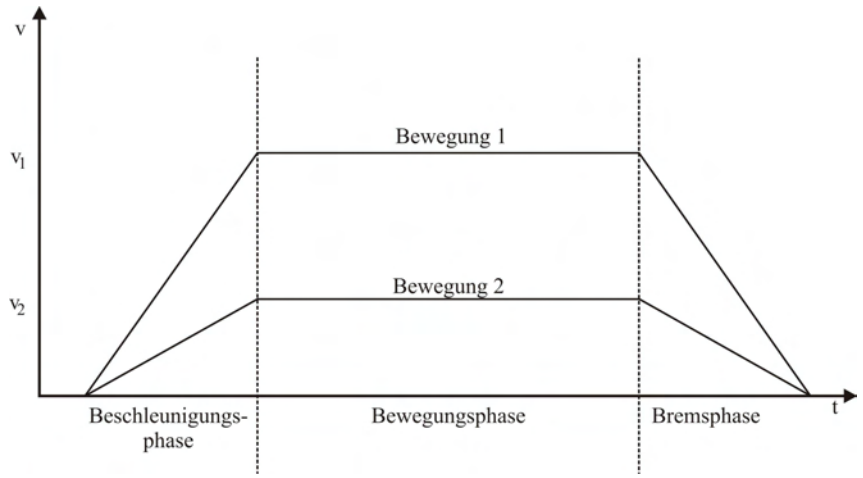


Abbildung 4.9: Modell einer gleichförmigen Bewegung.

Eine Bewegung wird dabei in drei Phasen eingeteilt (Abb. 4.9): die Beschleunigungsphase, die Bewegungsphase und die Bremsphase. Bei der Beschleunigungs- und Bremsphase wird von einer linearen Geschwindigkeitsänderung ausgegangen, wie in [LLoyd 1998]. Außerdem wird von einer konstanten Zeit bis zum Erreichen der Bewegungsgeschwindigkeit ausgegangen.

Bei der Analyse der Bewegung werden vor allen Dingen die Kräfte betrachtet, die in der Beschleunigungs- und Bremsphase der Bewegung auftreten. Während der Bewegungsphase wird von einer gleichförmigen Bewegung ausgegangen, in der keine weiteren Kräfte auftreten. Um die resultierenden Kräfte aus der Beschleunigungs- und Bremsphase berechnen zu können, kann der Benutzer eine Bewegungsgeschwindigkeit festlegen. Die Eingabe erfolgt nicht in Form von absoluten Geschwindigkeiten, sondern in Form von linguistischen Termen[5]. Hierbei resultiert eine schnelle Bewegung (Bewegung 1 in Abb. 4.9) in einer großen Beschleunigung, da die Zeit, die zum Erreichen der maximalen Geschwindigkeit zulässig ist, konstant ist. Analog dazu führt eine langsame Bewegung zu einer geringen Beschleunigung (Bewegung 2 in Abb. 4.9). Es ist möglich, die Beschleunigungs- und Bremszeit anzupassen und somit auch geringe Beschleunigungskräfte für Bewegungen mit hoher Bewegungsgeschwindigkeit zu erreichen. Die Form der auszuführenden Bewegung muss durch eine Eingabe festgelegt werden. Hierbei gibt es die folgenden zwei Möglichkeiten:

1. Die Bewegung wird über Bewegungsprimitive (rechts, links, auf, ab, Drehung um X-Achse, Drehung um Z-Achse etc.) definiert.

2. Die Bewegung wird über eine Start- und Zieltransformation definiert. Die Starttransformation beschreibt dabei die Position und Orientierung des Objekts relativ zum Weltkoordinatensystem, vor Beginn der Bewegung. Die Zieltransformation definiert die Position und Orientierung nach Abschluss der Bewegung. Die Bewegung entspricht dabei der Differenz zwischen den beiden Transformationen.

Wird die Bewegung über Primitive definiert, so können mehrere Primitive zu einer komplexen Bewegung kombiniert werden. Diese Menge von Primitiven wird in eine Transformation umgerechnet, die die Bewegung vollständig beschreibt. Somit ist eine Bewegung definiert über die folgenden Eigenschaften:

- Die Bewegungsgeschwindigkeit
- Die Bewegungstransformation

sowie optional

- die Beschleunigungszeit.

Der relevante Faktor für die Berechnung der Kräfte während der Bewegung ist die Beschleunigung. Die Kraft kann mit in Gleichung 3.41 eingebracht werden. Da die Kräfte jeweils zu Beginn und zum Abschluss der Bewegung berechnet werden müssen, ergeben sich zwei Gleichungen:

$$\vec{g}_{ext_{start}} = R(-m \cdot \vec{g} + V \cdot \rho \cdot \vec{g}) + B(m \cdot \vec{a}) \tag{4.5}$$

$$\vec{g}_{ext_{end}} = BR(-m \cdot \vec{g} + V \cdot \rho \cdot \vec{g}) + B(m \cdot -\vec{a}) \tag{4.6}$$

mit R als Orientierung des Objekts und B als Matrix der Rotation der Bewegung, relativ zum Weltkoordinatensystem. BR beschreibt somit die Orientierung des Objekts in der Zielposition relativ zum

[5] Ein linguistischer Term ist ein Begriff aus der Fuzzy-Logik. Mit linguistischen Termen können unscharfe Mengen beschrieben werden. Somit lassen sich z.B. Geschwindigkeitsangaben nicht in Form von reellen Zahlen, sondern durch Begriffe wie „langsam, schnell, sehr schnell“ etc. modellieren [Kruse et al. 1995].

Weltkoordinatensystem. m, g, V, ρ sind analog zu Gleichung 4.2 die Masse des Objekts, die Erdbeschleunigung, das Volumen und die Dichte des Füllstoffs. $\vec{a}$ beschreibt die Beschleunigung der Bewegung, in allen Achsen. Sie kann über die Bewegungsgeschwindigkeit berechnet werden:

$$\vec{a}(t) = \dot{\vec{v}}(t). \tag{4.7}$$

Somit ergibt sich die Beschleunigung aus der gewählten Geschwindigkeit und der Beschleunigungszeit. Ein entsprechendes Geschwindigkeitsprofil wird mit dem in [LLoyd 1998] beschriebenen Ansatz berechnet.

Bei einer solchen Bewegungsberechnung können Ausnahmefälle, wie z.B. die Bewegung eines Hammerschlags, nicht berücksichtigt werden. Die Berechnung der auftretenden Kräfte in der Beschleunigungsphase wären noch möglich, das Auftreffen auf einen Gegenstand allerdings entspricht nicht mehr einem „normalen“ Abbremsen der Bewegung, sondern einem Stoß. Die Art des Stoßes hängt von den Materialien ab, die am Stoß beteiligt sind, z.B. vom Hammer. Dies ist somit nur schwer und höchstens als spezielle Bewegung zu modellieren, hat mit der hier vorgestellten Definition einer Bewegung nur wenig zu tun und wird deshalb nicht weiter untersucht.

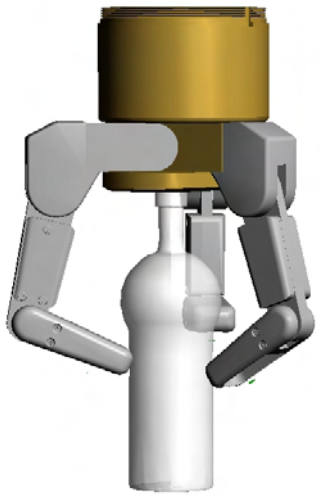


Abbildung 4.10: Griff einer Plastikflasche von oben (für eine bessere Visualisierung ist das Objekt halbtransparent dargestellt).

Als Beispiel ist der in Abbildung 4.10 dargestellte Griff bezüglich des Bewegungskriteriums evaluiert worden. Die Kräfte, die beim Anheben der Flasche auftreten, sind in Tabelle 4.3 zusammengefasst. Wie aus Tabelle 4.3 zu entnehmen ist, ist der Griff in allen Fällen stabil. Wird die Flasche jedoch zu schnell angehoben, ist also die Beschleunigung zu groß, so übersteigen die erforderlichen Kräfte die maximal zulässigen Kräfte von 20 N für die Plastikflasche. Wie ebenfalls aus den berechneten Daten in Tabelle 4.3 ersichtlich wird, stimmt das Modell der Simulation zur Berechnung der auftretenden Kräfte mit der Realität überein. Eine Beschleunigung von $1\frac{\mathrm{m}}{\mathrm{s}^2}$ hat nur einen relativ geringen Einfluss auf die Kräfte. Bei einer Beschleunigung von $10\frac{\mathrm{m}}{\mathrm{s}^2}$ ist die erforderliche Kraft fast doppelt so groß wie im Ruhezustand. Somit ist die aus der Bewegung resultierende Kraft, die sich aus $\vec{f} = m \cdot \vec{g} + m \cdot \vec{a}$ (mit $\vec{a}$ und $\vec{g}$ als Vektoren, welche die Beschleunigung entlang aller Achsen relativ zum Weltkoordinatensystem angeben) ergibt, auch ungefähr doppelt so groß, wenn g und a in etwa gleich groß sind.

Kontaktpunkt	Basis	F1 proximal	F2 proximal	F3 proximal
Ruhezustand	13,96 N	12,37 N	2,65 N	7,87 N
Start der Bewegung 1 m/s^2	15,14 N	13,44 N	2,87 N	8,59 N
Stopp der Bewegung -1 m/s^2	12,78 N	11,31 N	2,43 N	7,17 N
Start der Bewegung 10 m/s^2	25,93 N	23,18 N	4,91 N	15,14 N
Stopp der Bewegung -10 m/s^2	4,49 N	3,83 N	1,01 N	2,47 N

Tabelle 4.3: Berechnete Kräfte entlang der Z-Achse zu dem Griff aus Abbildung 4.10. Bei einer Bewegung in Richtung der positiven Welt-Z-Achse mit einer Beschleunigung von 1 m/s^2 und 10 m/s^2.

4.2.6 Objekt-Ports

Für die drei Folgeoperationen Einschenken, Ausgießen und Übergabe sind Bereiche definiert worden, die nicht mit dem Greifer penetriert werden dürfen. Als allgemeiner Fall von Bereichen, die nicht vom Greifer penetriert werden dürfen, sind die *Ports* für Objekte definiert. Hierbei werden Punkte oder Volumina an einem Objekt definiert, die als unbedingter Freiraum behandelt werden. Somit kann sichergestellt werden, dass bestimmte Bereiche an einem Objekt garantiert freigehalten werden und bei einem Griff nicht durch Teile des Manipulators verdeckt werden. Diese Ports können z.B. das Gewinde einer Schraube, die Öffnung einer Schraubenmutter oder der Henkel einer Tasse sein. In jedem Fall kann sichergestellt werden, dass die Funktionalität des Objekts nicht durch den Griff beeinträchtigt wird. Über diese Ports lassen sich somit Spezialfälle von Griffen realisieren. Dies erfordert allerdings die Interaktion mit einem Operator, der die Ports definiert. Die Ports werden durch geometrische Primitive beschrieben, die der Operateur entsprechend platzieren muss. Diese Primitive werden anschließend mit in das Objektmodell eingebracht. Formal entspricht die Definition der Objektports der des Übergabe Kriteriums aus Gleichung 4.4.

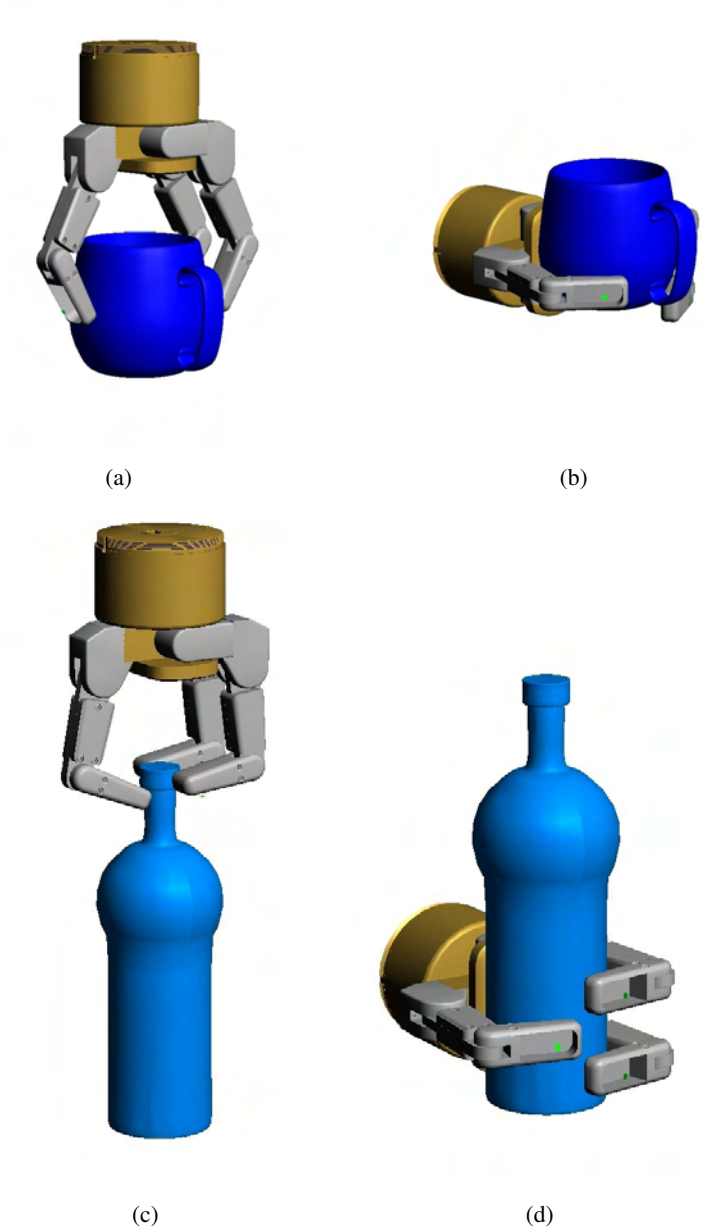
4.3 Semantisches Qualitätsmaß für Griffe

Aus den in den vorhergegangen Abschnitten vorgestellten Maßen kann nun ein *Umfassendes Qualitätsmaß* Q_G definiert werden, das alle anderen Maße beinhaltet. Dieses Maß für einen Griff G sei wie folgt definiert:

$$Q_G = \sum_{i=1}^{N} \lambda_i p_i \qquad \text{mit } p_i \in \mathcal{P} \tag{4.8}$$

wobei λ_i ein Gewichtsfaktor für eine Folgeoperation und p_i die korrespondierende Operation aus der Menge aller Folgeoperationen $\mathcal{P}$ ist. N ist die Kardinalität von $\mathcal{P}$. Über die Wahl von λ_i lässt sich der semantische Bezug eines Griffs zum gegriffenen Objekt variieren. Dies soll im Folgenden an einem kleinen Beispiel illustriert werden. Eine ausführliche Evaluation der Qualitätsmaße und der entsprechenden semantischen Bedeutung folgt in Abschnitt 4.4.

Als Beispiel für die Berechnung einer Semantik ist die Folgeoperation der Übergabe für die vier Griffe aus Abbildung 4.11 evaluiert worden. Die Gewichte für Gleichung 4.8 sind in Tabelle 4.4 aufgeführt. Sie sind so eingestellt worden, dass das bevorzugte Kriterium gegenüber dem nicht bevorzugten doppelt gewichtet wird.



(a) (b)

(c) (d)

Abbildung 4.11: Evaluation des Übergabe-Kriteriums bei einer Tasse und einer Flasche: (a) und (b) zeigen die berechneten Griffe für eine Tasse; (c) und (d) zeigen die berechneten Griffe für eine Flasche.

Die Ergebnisse der Evaluation sind in Abbildung 4.12 dargestellt. Die numerischen Ergebnisse für die einzelnen Operationen sind in Tabelle 4.5 zusammengefasst. Die Spalte „Übergabe“ beschreibt, an wie vielen Stellen des Objekts eine Übergabe-Operation durchführbar ist. Die Spalte Einschenken zeigt, ob die entsprechende Operation durchführbar ist oder nicht.

Griff	Gewichtung	
	Übergabe	Einschenken
Übergabe bevorzugt	1,0	0,5
Einschenken bevorzugt	0,5	1,0
Kein Bevorzugung	1,0	1,0

Tabelle 4.4: Gewichte für eine Griff-Evaluation. Die Werte in der Spalte „Übergabe“ definieren die Faktoren (λ_i aus Gl. 4.8) für die Operation Übergabe. Analog hierzu definieren die Werte der Spalte „Einschenken“ die Faktoren für die Operation Einschenken.

Griff	Werte für jede Operation	
	Übergabe	Einschenken
Tasse von oben	4	nicht möglich
Tasse von der Seite	1	möglich
Flasche von oben	22	nicht möglich
Flasche von der Seite	14	möglich

Tabelle 4.5: Ergebnisse der Evaluation für die Operationen „Einschenken“ und „Übergabe“. Die Werte in der Spalte Übergabe beschreiben die Anzahl der berechneten möglichen Positionen für eine Übergabe. Die Werte in der Spalte Einschenken geben an, ob die Folgeoperation Einschenken ausführbar ist.

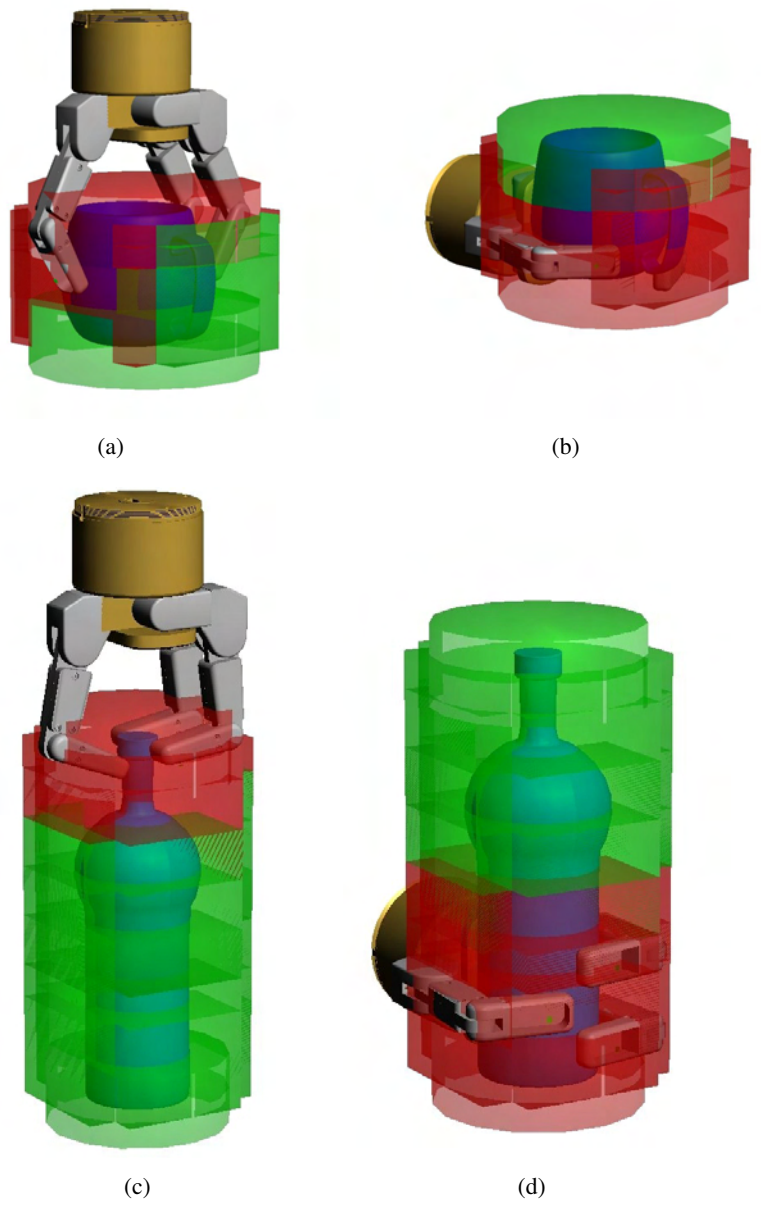
Tabelle 4.6 zeigt die nach Gleichung 4.8, berechneten Ergebnisse für die in Tabelle 4.4 aufgeführten Griffe. Um die Ergebnisse für die Operation Übergabe für verschiedene Objekte miteinander vergleichen zu können, sind diese noch mit der Objektgröße skaliert worden.

Griff	Gewichtete Summe		
	Übergabe bevorzugt	Einschenken bevorzugt	Keine Präferenz
Tasse von oben	0,20	0,10	0,20
Tasse von der Seite	0,55	1,03	1,05
Flasche von oben	0,56	0,28	0,56
Flasche von der Seite	0,86	1,18	1,36

Tabelle 4.6: Ergebnisse der gewichteten Summe nach Gleichung 4.8.

Die Ergebnisse zeigen, dass die Griffe von oben eine bessere Bewertung bezüglich der Übergabe-Operation erreichen, als die Griffe von der Seite, da sie mehrere Möglichkeiten bieten, das Objekt zu übergeben. Dagegen sind die Griffe unbrauchbar, wenn eine Einschenk-Operation ausgeführt werden soll, da die Hand die Öffnung der Tasse verdeckt. Der Griff für die Flasche von der Seite ist der am höchsten bewertete Griff, da er zum einen die Möglichkeit bietet, etwas in die Flasche einzufüllen und zum anderen eine Vielzahl von Übergabe-Positionen vorhanden ist. Dies trifft nicht auf den Griff von der Seite für die Tasse zu, da nur eine Position für eine mögliche Übergabe existiert.

Um sicherzustellen, dass mit einem Griff eine bestimmte Folgeoperation in jedem Fall durchführbar ist, muss diese mit dem Faktor 1 und alle anderen mit dem Faktor 0 gewichtet werden. In dem Fall wird allerdings nur diese eine Operation eines Griffs bewertet und alle anderen vernachlässigt. Um dies



(a) (b)

(c) (d)

Abbildung 4.12: Evaluation des Übergabe-Kriteriums : (a) und (b) zeigen die evaluierten Übergabeoperationen für eine Tasse; (c) und (d) zeigen die möglichen Positionen für eine Übergabe bei einer Flasche. Mögliche Angreifpunkte für eine Übergabe sind grün eingefärbt, gesperrte Bereiche rot.

zu verhindern, können verschiedene Kombinationen von *gewichteten Folgeoperationsmengen* zu einem Qualitätsmaß vereinigt werden. Dies ist definiert durch:

$$Q_G = \prod_{i=1}^{n} Q_i \tag{4.9}$$

mit Q_i, entsprechend zu Gleichung 4.8, als

$$Q_i = \sum_{j=1}^{N} \lambda_j^i p_j^i \qquad \text{mit } p_j^i \in \mathcal{P} \tag{4.10}$$

mit λ_j^i als einem Gewichtsfaktor für die jeweilige Operation p_j^i. Somit lassen sich Kombinationen von Operationen zu einer Gesamtoperation fusionieren. Ist $Q_G > 0$, so erfüllt der Griff alle gewünschten Eigenschaften.

4.4 Evaluation

In den bisher gezeigten Beispielen wurde die Anwendbarkeit der neu entwickelten Qualitätsmaße gezeigt. Im Folgenden soll dies vertieft werden, indem mehrere Griffe auf die Anwendbarkeit von unterschiedlichen Folgeoperationen untersucht werden. Des Weiteren werden Gewichtungen für verschiedene Semantiken entwickelt.

Die Griffe, die für die Evaluation verwendet werden, sind in Abbildung 4.13 und 4.14 auf den Seiten 68 und 69 dargestellt. Die exemplarische Auswahl der Beispiele erfolgte zum einen nach der Stabilität und zum anderen nach den Ergebnissen der Evaluation. Die Griffe werden indiziert und im Folgenden über die Indizes referenziert. Für die Auswertung werden die folgenden Qualitätsmaße betrachtet:

- Operation Einschenken
- Operation Ausgießen
- Operation Übergabe
- Operation Bewegung (mit jeweils zwei unterschiedlichen Geschwindigkeiten)
  - Bewegung entlang der Welt-Z-Achse (Bewegung 1)
  - Bewegung entlang der Welt-X-Achse (Bewegung 2)
  - Rotation um die Welt-Y-Achse (Bewegung 3)
  - Rotation um Welt-Z-Achse (Bewegung 4)
  - Kombinierte Bewegung (Bewegung 5)

Diese Bewegungen entsprechen dabei den Bewegungen, die erforderlich sind, um ein Objekt aufzunehmen (Bewegung 1), es nach rechts oder links zu bewegen (Bewegung 2), einer Rotation, die z.B. erforderlich ist, um ein Objekt von der Längs- auf die Querseite zu stellen (Bewegung 3) und einer Rotation, mit der sich z.B. die Orientierung eines Griffs eines Objekts verändern lässt (Bewegung 4). Bewegung 5 stellt eine Kombination aus Bewegung 1 und 3 dar, entspricht also einer Ausgießbewegung in Kombination mit einer Aufwärtsbewegung. Sie ist gewählt worden, um eine Bewegung um mehr als eine Achse und Richtung zu evaluieren. Die Bewegungs-Operationen werden mit zwei unterschiedlichen Geschwindigkeiten evaluiert. Zum einen mit einer geringen Geschwindigkeit und zum anderen mit einer großen Geschwindigkeit. Die sich daraus ergebende Beschleunigung beträgt 1 m/s und 10 m/s.

Die Gewichtungen, die für die Berechnung der Griffsemantik verwendet werden, sind in Tabelle 4.7 dargestellt. Sie sind so eingestellt, dass die Summe der Gewichte aus Gleichung 4.8 immer gleich 1 ist, also

$$\sum_{i=1}^{N} \lambda_i = 1. \tag{4.11}$$

Des Weiteren wird immer genau eine Aktion bevorzugt bewertet. Die bevorzugte Aktion wird mit 52% und alle anderen Aktionen mit 3% gewichtet. Da die Bewegungs-Operationen mit zwei unterschiedlichen Geschwindigkeiten evaluiert werden, sind diese aus Gründen der Übersichtlichkeit in Tabelle 4.7 zu einem Wert zusammengefasst. Hierdurch ergibt sich die Summe von 1 in jeder Zeile der Tabelle 4.7.

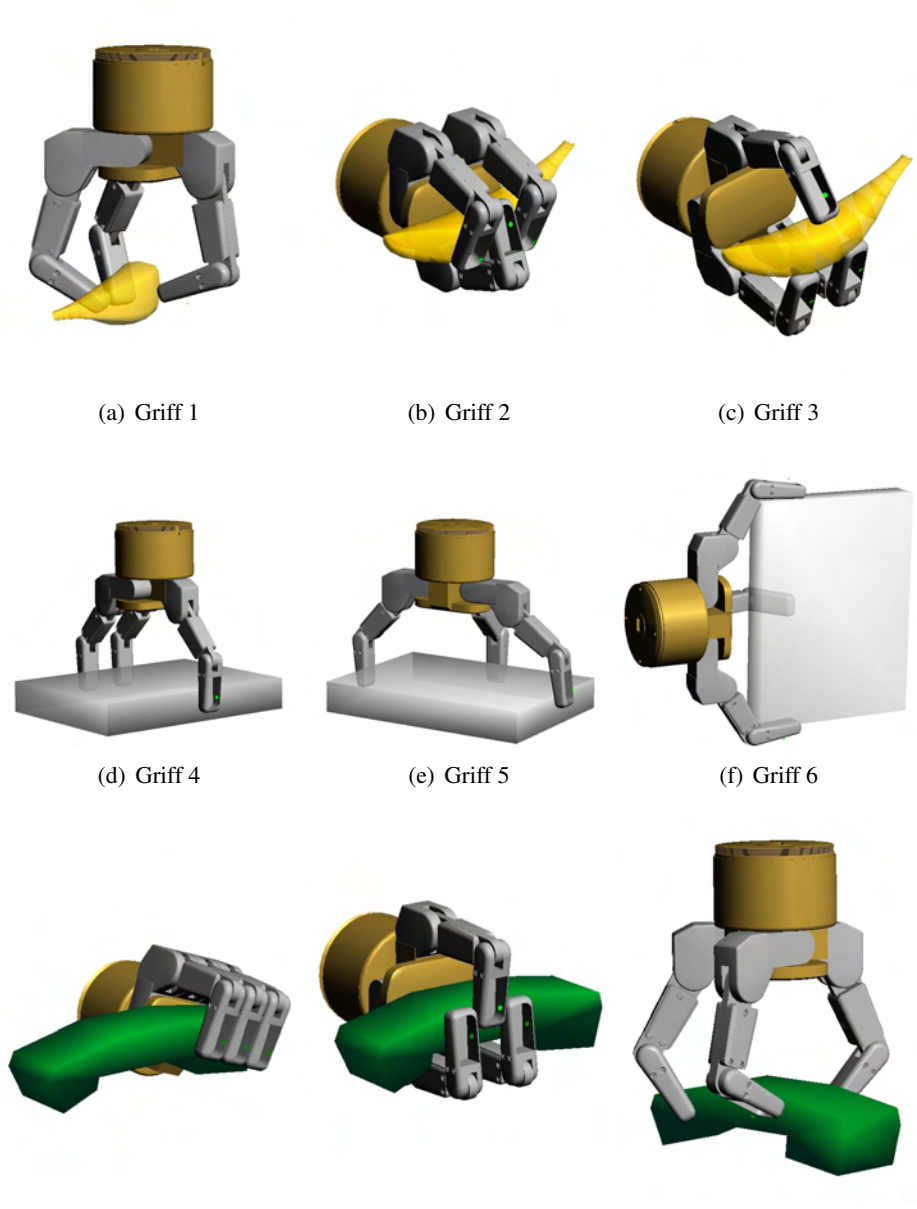
Durch die Gewichtung von 0,03 werden die nicht bevorzugten Aktionen nicht völlig ignoriert, aber ein Wert ≥ 0 für eine Aktion bedeutet nicht automatisch, dass die Aktion auch ausführbar ist. In diesem Fall müsste das Gewicht für die jeweilige Aktion 1 und das der anderen Aktionen gleich 0 sein. Beispielsweise ist die Operation Ausgießen für Griff 18 als nicht ausführbar evaluiert worden (siehe Tabelle 4.8), trotzdem ist der Griff, wenn die Operation Ausgießen bevorzugt gewichtet wird, insgesamt mit 0,284 bewertet (siehe Tabelle 4.9). Dieser Wert ergibt sich, da die übrigen Operationen sowie die Stabilität des Griffs mit in die Bewertung eingehen. Da die meisten der übrigen Operationen durchführbar sind, ergibt sich der berechnete Wert.

Für Bewegungen wird geprüft, ob sie mit zwei unterschiedlichen Geschwindigkeiten bzw. Beschleunigungen (langsam und schnell) ausgeführt werden können. Hierbei wird die Summe beider Ergebnisse als Gesamtergebnis gewertet. Ähnliches gilt für die Operation Ausgießen. Sie wird mit zwei unterschiedlichen Drehrichtungen evaluiert. Ein Operation wird dabei als nicht durchführbar bewertet, wenn die entsprechenden Bedingungen nicht erfüllt sind. Zwar sind alle Griffe als grundsätzlich stabil bewertet worden, es kann jedoch sein, dass die Kräfte, die für eine Operation erforderlich sind, die maximal zulässigen Kräfte für das jeweilige Objekt überschreiten. Ist dies der Fall, wird die Operation als „nicht ausführbar“ gewertet.

Die Ergebnisse der Evaluation der Griffe sind in Tabelle 4.8 zusammengefasst. Die Werte in der Spalte „Übergabe“ geben an, wie viele mögliche Positionen für eine Übergabe-Operation ermittelt worden sind. Die Werte in der Spalte „Stabilität“ sind mittels des L_1-Grasp-Wrench-Space (siehe Kapitel 3.2.3) berechnet worden. Je größer der entsprechende Wert ist, um so stabiler ist der Griff. Eine 1 in der Spalte „Eingießen“ bedeutet, dass die Operation Eingießen mit dem Griff durchführbar ist, eine 0 dagegen bedeutet, dass die Operation nicht ausgeführt werden kann. Da die Operation Ausgießen und die fünf Bewegungen in je zwei unterschiedlichen Konfigurationen ausgeführt worden sind, bedeutet ein Wert „0/2“ in diesem Fall, dass die Operation in keiner Konfiguration durchführbar ist, „1/2“ bedeutet, eine Konfiguration ist ausführbar und „2/2“ bedeutet, dass alle Konfigurationen mit dem jeweiligen Griff durchführbar sind.

Die Ergebnisse der Gewichtung der einzelnen Operationen, also die evaluierten Operationen aus Tabelle 4.8, multipliziert mit den Gewichten aus Tabelle 4.7, sind in in Tabelle 4.9 dargestellt.

Im Folgenden werden anhand von Beispielen die Ergebnisse der Evaluation aus Tabelle 4.9 veranschaulicht. Die Ergebnisse lassen sich anhand der Objekteigenschaften in zwei Gruppen einteilen: zum einen die Griffe von Gefäßen, also die Objekte, in die etwas eingefüllt werden kann, und zum anderen die Griffe von Objekten, die keine Gefäße sind. Für die Objekte, die keine Gefäße sind, also die Banane, das Buch und der Telefonhörer, bestimmten im Wesentlichen zwei Eigenschaften die Bewertung der Griffe: Zum einen, ob sich der Griff für Übergabeoperationen eignet, und zum anderen wie stabil er in Bezug auf die gewählten Bewegungen ist. Schaut man sich z.B. die analysierten Griffe für die Banane an (Griff

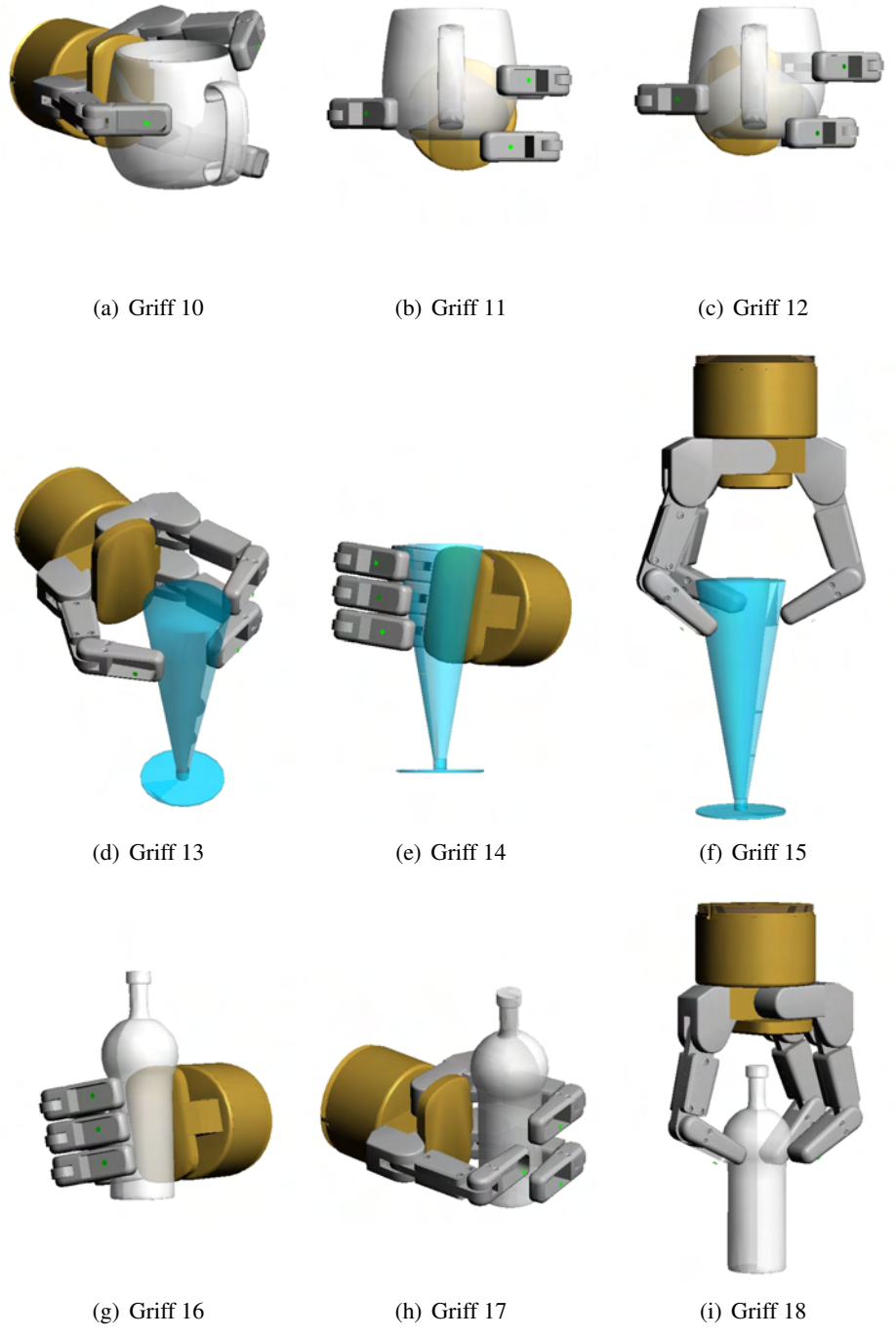


(a) Griff 1 (b) Griff 2 (c) Griff 3

(d) Griff 4 (e) Griff 5 (f) Griff 6

(g) Griff 7 (h) Griff 8 (i) Griff 9

Abbildung 4.13: Griffe für die Evaluation der Qualitätsmaße. Die Griffe sind alle mit der gleichen Orientierung in Bezug zum Weltkoordinatensystem dargestellt. Zeile 1: Griffe für eine Banane; Zeile 2: Griffe für ein Buch; Zeile 3: Griffe für einen Telefonhörer.



(a) Griff 10 (b) Griff 11 (c) Griff 12

(d) Griff 13 (e) Griff 14 (f) Griff 15

(g) Griff 16 (h) Griff 17 (i) Griff 18

Abbildung 4.14: Griffe für die Evaluation der Qualitätsmaße. Die Griffe sind alle mit der gleichen Orientierung in Bezug zum Weltkoordinatensystem dargestellt. Zeile 1: Griffe für eine Tasse; Zeile 2: Griffe für ein Sektglas; Zeile 3: Griffe für eine Flasche

Bevorzugte Operation	Gewichte für Operationen: Übergabe	Eingießen	Ausgießen	Bewegung 1	Bewegung 2	Bewegung 3	Bewegung 4	Bewegung 5	Stabilität
Übergabe	**0,52**	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06
Eingießen	0,06	**0.52**	0,06	0,06	0,06	0,06	0,06	0,06	0,06
Ausgießen	0,06	0,06	**0.52**	0,06	0,06	0,06	0,06	0,06	0,06
Bewegung 1	0,06	0,06	0,06	**0.52**	0,06	0,06	0,06	0,06	0,06
Bewegung 2	0,06	0,06	0,06	0,06	**0.52**	0,06	0,06	0,06	0,06
Bewegung 3	0,06	0,06	0,06	0,06	0,06	**0.52**	0,06	0,06	0,06
Bewegung 4	0,06	0,06	0,06	0,06	0,06	0,06	**0.52**	0,06	0,06
Bewegung 5	0,06	0,06	0,06	0,06	0,06	0,06	0,06	**0.52**	0,06
Stabilität	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	**0.52**
Keine Präferenz	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11	0,11

Tabelle 4.7: Gewichte der einzelnen Operationen für die Evaluation.

1-3), so eignen sich alle für eine Übergabeoperation. Lediglich Griff 2 bietet weniger Möglichkeiten für eine Übergabe als die beiden anderen Griffe. Griff 1 und 3 sind ähnlich stabil, nur Griff 2 ist nicht in der Lage, den Kräften einer der Bewegungen entgegenzuwirken, und wird somit leicht abgewertet. Wie aus Tabelle 4.9 zu sehen ist, sind Griff 1 und 3 nahezu gleichwertig. Griff 2 wird aufgrund einer nicht ausführbaren Bewegung und dem Mangel an Übergabemöglichkeiten leicht abgewertet. Griff 2 ist dabei allerdings der Griff, der laut GWS-Berechnung (siehe Kapitel 3.2.3) am stabilsten von den drei Griffen sein müsste. An diesem Beispiel wird deutlich, dass eine gute Stabilitätsbewertung durch den GWS nicht automatisch bedeutet, dass alle Bewegungen mit diesem Griff ausgeführt werden können, sondern dass die Bewegungen immer im Einzelfall zu überprüfen sind.

Gleiches gilt für die Griffe des Buches (Griff 4-6). Hierbei ist Griff 4 der Griff, der bezüglich der Stabilität die höchste Bewertung erhalten hat (siehe Tabelle 4.8). Es ist jedoch nicht möglich, mit ihm alle Bewegungen auszuführen.

Aus der Analyse der Griffe für den Telefonhörer (Griff 7-9) wird ersichtlich, dass eine besonders gut bewertete Eigenschaft den Griff nicht automatisch aufwertet. Griff 7 ist im Verhältnis zu den beiden anderen besonders stabil, Griff 8 bietet besonders viele Übergabemöglichkeiten und Griff 9 weist keine besonderen Eigenschaften auf. Obwohl Griff 8 doppelt so viele Übergabemöglichkeiten bietet wie Griff 7, wird dieser Nachteil von Griff 7 durch die deutlich bessere Stabilität des Griffs ausgeglichen. Zwar ist Griff 8 bei Präferenz der Übergabe-Operation am besten bewertet, die Bewertung entspricht aber nicht dem Verhältnis, das sich aus der Anzahl der Übergabemöglichkeiten zwischen den beiden Griffen ergibt. Betrachtet man nur das Verhältnis der Übergabemöglichkeiten zwischen Griff 7 und 8, so müsste Griff 8 ungefähr dreimal so gut bewertet werden wie Griff 7. Das Verhältnis der gewichteten Summe aus Tabelle 4.9 beträgt aber nur 1,1. Betrachtet man die Stabilität als die zu bevorzugende Komponente, so ist Griff 7 am besten bewertet, aber der Abstand zu den beiden anderen Griffen entspricht - wie zuvor - nicht dem Faktor, der sich aus der ausschließlichen Bewertung anhand der GWS-Berechnung ergibt. Da auch mit Griff 7 Übergabeoperationen möglich sind und er der stabilste Griff ist, erhält er die höchste Bewertung, wenn keine Operation bevorzugt wird. Darauf folgt Griff 8, der viele Übergabemöglichkei-

	Griff	Ergebnisse der Evaluation für Operation: Übergabe	Eingießen	Ausgießen	Bewegung 1	Bewegung 2	Bewegung 3	Bewegung 4	Bewegung 5	Stabilität
Banane	1	5	-	-	2/2	2/2	2/2	2/2	2/2	0,42
	2	2	-	-	2/2	1/2	2/2	2/2	2/2	0,46
	3	4	-	-	2/2	2/2	2/2	2/2	2/2	0,44
Buch	4	13	-	-	1/2	2/2	2/2	0/2	0/2	0,09
	5	15	-	-	1/2	1/2	2/2	0/2	0/2	0,07
	6	14	-	-	1/2	0/2	1/2	0/2	0/2	0,10
Telefon	7	3	-	-	2/2	2/2	2/2	2/2	1/2	0,24
	8	10	-	-	1/2	2/2	2/2	0/2	0/2	0,05
	9	4	-	-	1/2	2/2	2/2	0/2	0/2	0,03
Tasse	10	0	0	0/2	2/2	2/2	2/2	2/2	2/2	0,003
	11	1	1	1/2	2/2	2/2	2/2	2/2	2/2	0,27
	12	0	1	2/2	2/2	2/2	2/2	2/2	2/2	0,40
Sektglas	13	10	0	1/2	2/2	2/2	2/2	1/2	0/2	0,02
	14	6	1	0/2	1/2	2/2	2/2	0/2	0/2	0,15
	15	13	0	0/2	1/2	2/2	2/2	0/2	0/2	0,07
Flasche	16	5	1	2/2	2/2	2/2	2/2	2/2	2/2	0,25
	17	8	1	2/2	2/2	2/2	2/2	2/2	2/2	0,83
	18	6	0	0/2	2/2	2/2	2/2	2/2	1/2	0,05

Tabelle 4.8: Ergebnisse der Evaluation der Folgeoperationen für die Griffe aus Abbildung 4.13 und 4.14. Ein Wert von 0 bedeutet, die Operation ist nicht ausführbar, ein Wert von 1 bezeichnet eine durchführbare Operation. Da die Operation Ausgießen mit zwei unterschiedlichen Winkeln und die Operationen Bewegung mit zwei unterschiedlichen Geschwindigkeiten berechnet worden sind, bedeutet 1/2, dass nur eine von beiden Möglichkeiten ausgeführt werden kann und 2/2, dass beide möglich sind. Die Werte in der Spalte „Übergabe“ geben die Anzahl der evaluierten möglichen Übergabepositionen wieder. Die Spalte „Stabilität“ gibt den mittels eines GWS berechneten Wert für die Stabilität an.

ten bietet, aber nicht sonderlich stabil ist. Den letzten Platz in diesem Trio belegt Griff 9, der sich weder durch Stabilität noch durch viele Übergabemöglichkeiten auszeichnet.

Bei den Objekten, die sich als Container gebrauchen lassen, müssen noch die Operationen Einfüllen und Ausgießen mit berücksichtigt werden. Ein- und Ausgießen wird jedoch nur als durchführbar bewertet, wenn *kein* Teil des Manipulators die Öffnung des Objekts verdeckt. Zum Beispiel verdeckt der Finger bei Griff 10 die Öffnung nicht, aber er reicht doch in den Bereich der Öffnung hinein (Abb. 4.14 Seite 69). Um sicherzustellen, dass der Roboter nicht durch Flüssigkeitsspritzer beschädigt wird, werden in einem solchen Fall die Operationen Einschenken und Ausgießen als nicht durchführbar bewertet. Dies führt auch zu einer deutlichen Abwertung von Griff 10 gegenüber den beiden anderen Griffen (Griff 11 und 12). Griff 11 und 12 sind in ihrer Bewertung einander ähnlich. Griff 12 ist etwas stabiler als Griff 11 und bietet die Möglichkeit, die Tasse sowohl durch eine Rotation im als auch entgegen des Uhrzeigersinns auszugießen. Griff 11 bietet dafür eine Übergabemöglichkeit. Zwar scheint der Platz bei Griff 10 auch ausreichend zu sein, um die Tasse aus der Hand zu übernehmen, um aber Schäden am Roboter und Ver-

	Griff	Ergebnisse der Evaluation für Operation: Übergabe	Eingießen	Ausgießen	Bewegung 1	Bewegung 2	Bewegung 3	Bewegung 4	Bewegung 5	Stabilität	Keine Präferenz
Banane	1	0,46	-	-	1,65	1,65	1,65	1,65	1,65	0,63	0,63
	2	0,35	-	-	1,58	1,09	1,58	1,58	1,58	0,51	0,56
	3	0,43	-	-	1,65	1,65	1,65	1,65	1,65	0,54	0,63
Buch	4	0,38	-	-	0,83	1,32	1,32	0,34	0,34	0,21	0,33
	5	0,38	-	-	0,77	0,77	1,26	0,28	0,28	0,17	0,28
	6	0,30	-	-	0,87	0,16	1,14	0,16	0,16	0,13	0,17
Telefon	7	0,36	-	-	1,56	1,56	1,56	1,56	1,07	0,40	0,54
	8	0,40	-	-	0,82	1,31	1,31	0,33	0,33	0,19	0,33
	9	0,26	-	-	0,80	1,29	1,29	0,31	0,31	0,17	0,30
Tasse	10	0,30	0,30	0,60	1,58	1,58	1,58	1,58	1,58	0,30	0,55
	11	0,46	0,90	1,86	1,86	1,86	1,86	1,86	1,86	0,56	0,81
	12	0,44	0,90	1,87	1,87	1,87	1,87	1,87	1,87	0,63	0,82
Sektglas	13	0,42	0,17	7,07	1,31	1,31	0,82	0,33	0,33	0,17	0,33
	14	0,38	0,69	0,46	0,95	1,44	1,44	0,46	0,46	0,30	0,44
	15	0,50	0,18	0,35	0,84	1,33	1,33	0,35	0,35	0,21	0,36
Flasche	16	0,59	0,90	1,87	1,87	1,87	1,87	1,87	1,87	0,56	0,83
	17	0,71	0,94	1,95	1,95	1,95	1,95	1,95	1,95	0,87	0,92
	18	0,46	0,28	0,57	1,55	1,55	1,55	1,55	1,06	0,31	0,54

Tabelle 4.9: Die Werte der Tabelle sind die berechneten Werte für Q_G nach Gleichung 4.8. Sie wurden mit den Gewichten aus Tabelle 4.7 und den evaluierten Operationen aus Tabelle 4.8 berechnet. Je größer der Wert einer Operation, umso besser ist der Griff für die jeweilige Operation geeignet.

letzungen von Menschen ausschließen zu können, sind die Volumina, die für eine Übergabe erforderlich sind, relativ groß definiert worden. Wegen dieser Sicherheitsüberlegungen sind keine verzahnten Positionen, also Positionen, bei denen jemand zwischen die Finger der BarrettHand greift, überprüft worden. Somit bieten sich für die Griffe nur wenig Übergabemöglichkeiten.

Keiner der Griffe für das Sektglas (Griffe 13-15) ist in der Lage, den Kräften, die bei einer Ausgießbewegung auftreten, zu widerstehen. Lediglich der *Powergriff*, Griff 14, kann das zusätzliche Gewicht beim Einfüllen von Flüssigkeiten ausbalancieren. Er ist auch der stabilste Griff für das Sektglas. Er bietet aber wiederum weniger Varianten für eine Übergabe. Hier zeigt sich auch wieder, dass die Stabilität von Griffen nicht generell als Maß genutzt werden kann, wenn spezielle Bewegungen ausgeführt werden sollen. Die Ausgießbewegung (Bewegung 4) ist, mit geringer Beschleunigung, nur mit Griff 13 durchführbar, der laut GWS-Berechnung der Schwächste der drei Griffe für das Sektglas ist.

Bei den Griffen für die 0,5 l Flasche (Griff 16-18) sticht Griff 17 durch seine Eigenschaften hervor. Er ist der stabilste Griff, alle Bewegungen können ausgeführt werden und er bietet die meisten Möglichkeiten für eine Übergabe. Griff 16 ist der am zweitbesten bewertete Griff für die Flasche, egal welche Operation bevorzugt bewertet wird. Dies liegt daran, dass er nur eine Übergabemöglichkeit weniger bietet als Griff 18, dafür aber deutlich stabiler ist.

4.5 Bewertung der Ergebnisse der Evaluation von Griffen

In diesem Kapitel wurde ein Qualitätsmaß für Griffe auf Basis der Weiterverwendbarkeit von Objekten entwickelt. Dieses neue Maß bezieht die Objektsemantik mit in die Bewertung eines Griffs ein. Diese Semantik kann von dem Benutzer anhand von vier unterschiedlichen Folgeoperationen definiert werden. Somit lässt sich ein Griff bezüglich seiner Eigenschaften bewerten und nicht nur anhand seiner Stabilität, wie bei den in Abschnitt 4.1 vorgestellten Verfahren. Wie die Ergebnisse der Evaluation zeigen, zeichnen sich die verwendeten Folgeoperationen durch eine gute Anwendbarkeit für die Berechnung von Griffen aus.

Da die mittels der Semantik aus Gleichung 4.8 berechneten Werte linear zu den evaluierten Folgeoperationen sind, ist es möglich, eine Rangordnung anhand der berechneten Werte für die Griffe zu definieren. Diese Rangordnung beschreibt die Eignung eines Griffs für eine bestimmte Operation. Je größer der Wert, um so besser ist der Griff geeignet, die Folgeoperation auszuführen. Soll eine bestimmte Operation unbedingt ausführbar sein, so kann dies durch die gewichteten Folgeoperationsmengen (Gleichung 4.9) erreicht werden. Da nicht nur die evaluierten Folgeoperationen in die Berechnung des Qualitätsmaßes eingehen, sondern auch die Stabilität eines Griffs mit einbezogen wird, ist die Stabilität auch ein Teil des berechneten Qualitätsmaßes. Somit ist das entwickelte Qualitätsmaß den in Abschnitt 4.1 beschriebenen Qualitätskriterien überlegen. Dies zeigen die zuvor beschriebenen Beispiele. Wie z.B. bei den Griffen für das Sektglas oder die Banane gezeigt wurde, ist der Griff mit der besten Stabilitätsbewertung nicht immer auch der Griff, mit dem alle möglichen Bewegungen ausgeführt werden können. Auch die evaluierten Griffe für den Telefonhörer zeigen, dass mit dem Qualitätsmaß ein guter Ausgleich zwischen den einzelnen Operationen erfolgen kann. Möchte man z.B. für den Telefonhörer einen Griff anwenden, der möglichst viele Übergabemöglichkeiten bietet, so wird bei einer Analyse der Übergabemöglichkeiten Griff 8 ausgewählt. Sollen aber noch andere Operationen mit in Betracht gezogen werden (wird also keine Operation bevorzugt), so ist Griff 9 der am besten bewertete Griff. Diese Beispiele zeigen, dass sich mit dem Qualitätsmaß Griffe für unterschiedliche Anwendungen durch die entsprechende Wahl der Gewichtungsfaktoren definieren lassen. Eine Anwendung dieser Selektion wird in Kapitel 7 genutzt, um Griffe mit einer bestimmten Eigenschaft zu selektieren. Zudem ist es mit den vorgestellten Folgeoperationen jedermann möglich, eine Semantik für einen Griff zu erstellen, auf deren Basis eine entsprechende Rangordnung für Griffe definiert werden kann.

Kapitel 5

Grundlagen des Verstärkungslernens

Verstärkungslernen oder auch *Reinforcement-Lernen* ist eine Form des maschinellen Lernens. Verstärkungslernen liegt zwischen überwachtem und unüberwachtem Lernen. Reinforcement-Lernen erfolgt nach dem „Belohnen und Bestrafen“ Prinzip. Es wird also, wie beim überwachten Lernen, ein Signal gegeben, das die ausgeführten Aktionen bewertet. Dieses Signal wird aber mittels Beobachtung der Umwelt generiert. Laut Definition ist das Signal ein Teil der Umwelt. Wie das Signal generiert wird, ist dem System nicht notwendigerweise bekannt (daher unüberwacht). Verstärkungslernen soll also einen Algorithmus in die Lage versetzen, sein Ziel zu erreichen, ohne genau zu spezifizieren, mit welchen Mitteln dies geschehen kann. Der Algorithmus kann aus einer Reihe von Aktionen (mehr oder weniger) zufällig Aktionen auswählen. Nach der Ausführung der Aktion gibt es das oben erwähnte Signal zur Belohnung oder Bestrafung. Das Lernen besteht somit in einer Maximierung der Belohnungen über einen gewissen Zeitraum (der nicht begrenzt sein muss).

In diesem Kapitel sollen die Grundlagen des Verstärkungslernens vermittelt werden. Aufbauend auf den Grundlagendefinitionen werden einige Lösungsmethoden und Algorithmen aus dem Bereich des Reinforcement-Lernens vorstellt. Ein zentraler Begriff ist hierbei der des *Markov-Entscheidungs-Prozesses* (*MDP*). Das in Kapitel 6.6 verwendete Verfahren zur Generierung von Griffen, ist eine Variante des TD-λ Verfahrens, das ebenfalls in diesem Kapitel definiert wird. Abschließend wird noch ein Ausblick auf hierarchische Verfahren des Reinforcement-Lernens gegeben

Einführungen in die Grundlagen des Verstärkungslernens sind auch in [Kaelbling et al 1996], [Glorennec 2000] und [Keerthi and Ravindran] zu finden. Ausführlich wird das Thema in [Sutton and Barto 1998] behandelt.

Wenn im Folgenden von *Agenten* gesprochen wird, so ist damit eine Software gemeint, die als eigenständige Einheit in der Lage ist Wissen, zu akquirieren und auszuwerten. Der Begriff *Agent* bezeichnet also eine lernende Software.

5.1 Definition Verstärkungslernen

Die Basis für das Verstärkungslernen bildet der Markovsche-Entscheidungsprozess (*Markov Decision Process, MDP*) . Ein solcher Prozess ist durch die folgenden Eigenschaften charakterisiert:

- $s \in \mathcal{S}$ einer diskreten Menge von *Weltzuständen*
  $\mathcal{S}$ umfasst die Menge aller möglichen Weltzustände, die der Agent während des Lernens einnehmen kann.

- $a \in \mathcal{A}$ einer Menge von *Aktionen*
  Sie stellen den „Methodensatz“ des Agenten dar, der ihm zur Verfügung steht, um sein Ziel zu erreichen.

- $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ einer *Bewertungsfunktion* (*Reward*)
  Die Bewertungsfunktion verknüpft die Zustände und Aktionen mit einem reellen Wert. Dieser Wert ist die Grundlage des Lernens für den Agenten. Ziel des Agenten ist es, die Rewards über einen bestimmten Zeitraum zu maximieren. Die Bewertungsfunktion ist Teil der Umwelt des Agenten und ihm somit nicht bekannt.

- $\delta : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ einer *Zustandsübergangsfunktion*
  Die Zustandsübergangsfunktion ist ebenfalls Teil der Umwelt und somit dem Agenten nicht bekannt. Sie sorgt dafür, dass der Agent nach einer Aktionsauswahl und -ausführung einen Zustand übermittelt bekommt. $s \in \mathcal{S}$ geht also durch die Ausführung von $a \in \mathcal{A}$ in $s^{'} \in \mathcal{S}$ über. Der Zustand $s^{'} \in \mathcal{S}$ muss nicht notwendigerweise verschieden von seinem Vorgängerzustand $s \in \mathcal{S}$ sein. Die Zustandsübergangsfunktion muss keine deterministische Funktion sein.

- $\pi : \mathcal{S} \rightarrow \mathcal{A}$ einer *Strategie* (*Policy*)
  Die Strategie ist eine Funktion, die Zustände mit Aktionen verbindet. Da die Aktionen, oder besser deren Ausführung durch den Agenten, zu einer Bewertung führen, sollte eine optimale Strategie so ausgerichtet sein, dass Aktionen, die eine gute Bewertung zur Folge haben, bevorzugt behandelt werden.Die Strategie ist verantwortlich für das Verhalten des Agenten.

- $V^{\pi} : \mathcal{S} \rightarrow \mathbb{R}$ einer *Wertefunktion*
  Die Zustands-Wertefunktion verknüpft die einzelnen Weltzustände $s \in \mathcal{S}$ unter der Verwendung einer bestimmten Strategie π mit einem reellen Wert. Dieser Wert ergibt sich aus der Summe der zu erwartenden Rewards die sich nach Lösen der *Bellmann-Gleichung* für den jeweiligen Zustand ergeben. Die Bellmann-Gleichung und die verschiedenen Varianten der Wertefunktion werden im Folgenden näher erläutert.

Nach der Definition besteht der Lernprozess in einer Maximierung der Rewards, also in der Optimierung der Zustands-Wertefunktion V^{π} und einer Optimierung der Policy π mittels V^{π}.

Die Markov-Eigenschaft eines solchen Prozesses besteht darin, dass die Entscheidung welche Aktion als nächstes auszuführen ist, ausschließlich anhand des aktuellen Zustandes gefällt werden kann. Im allgemeinen Fall hängt die Reaktion für ein System zum Zeitpunkt $t+1$ nach einer Aktionsauswahl zum Zeitpunkt t von allen vorhergegangen Zuständen, Aktionen und Rewards ab. Dies kann formal definiert werden durch die folgende Wahrscheinlichkeitsverteilung[1]:

$$Pr\{s_{t+1} = s^{'}, r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, \ldots, s_0, a_0\}, \quad (5.1)$$
$$\forall s^{'}, r, \text{ und alle Möglichkeiten für } s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, \ldots, s_0, a_0$$

Ist der Prozess ein Markov-Prozess, so hängt die Reaktion zum Zeitpunkt $t+1$ nur von dem aktuellen Zustand (t) ab. Somit kann die Dynamik definiert werden durch:

$$Pr\{s_{t+1} = s^{'}, r_{t+1} = r | s_t, a_t\}, \quad \forall s^{'}, r, s_t, a_t \quad (5.2)$$

[1] Um das Problem zu vereinfachen, wird von einem endlichen Zustandsraum und einer endlichen Menge von Bewertungen ausgegangen.

Somit muss bei einem Markov-Prozess gelten, dass Gleichung 5.1 gleich Gleichung 5.2 für alle $s^{'}, r$, ist. Hieraus folgt auch, dass der Agent kein Wissen über vorhergegangene Aktionen und Weltzustände haben muss bzw. hat, um eine Aktionsauswahl auszuführen.

5.1.1 Wertefunktionen

Nahezu alle Verfahren aus dem Bereich des selbstbewertenden Lernens basieren auf der Berechnung oder Schätzung einer Wertefunktion. Es gibt verschiedene Möglichkeiten, diese Wertefunktionen zu berechnen. Im Folgenden sollen drei Varianten für die Berechnung von V^{π} vorgestellt werden. Sei t der aktuelle Zeitpunkt mit dem Zustand s_t, dann kann V^{π} wie folgt definiert werden:

Endlicher Horizont (*Finite Horizon*)

$$V^{\pi}(s_t) = E\left\{\sum_{d=0}^{h} r_{t+d}\right\} \tag{5.3}$$

Bei dieser Variante werden die zu *erwartenden* Rewards der nächsten h Schritte aufsummiert. h ist also der endliche Horizont, bis zu dem das System rechnet. Da in einer nicht deterministischen Umgebung nicht vorhersagbar ist, ob die Ausführung einer Aktion den Agenten in den Zustand $s^{'} \in \mathcal{S}$ oder in den Zustand $s^{''} \in \mathcal{S}$ mit $s^{'} \neq s^{''}$ überführt, wird hier mit dem Erwartungswert $E\left[\sum_{d=0}^{h} r_{t+d}\right]$ und nicht dem reellen Wert der Akkumulierung $\sum_{d=0}^{h} r_{t+d}$ gerechnet. Dies gilt natürlich auch für die folgenden Varianten.

Durchschnittlicher Reward (*Average Reward*)

$$V^{\pi}(s_t) = \lim_{h\to\infty} \frac{1}{h} E\left\{\sum_{d=0}^{h} r_{t+d}\right\} \tag{5.4}$$

Hierbei soll der im Mittel zu erwartende Langzeit-Reward maximiert werden. Es wird also im Mittel immer die bestmögliche Aktion ausgewählt werden.

Verzögerter Reward (*Delayed Reward*)

$$V^{\pi}(s_t) = E\left\{\sum_{d=0}^{\infty} \gamma^d r_{t+d}\right\} \text{ mit } 0 \leq \gamma < 1 \tag{5.5}$$

Der Delayed Reward ist die am meisten gebrauchte Form des Rewards. Hierbei wird eine Gewichtung eingeführt, d.h. die zu erwartenden Rewards werden abgeschwächt. Durch die Verwendung von γ^d, mit $0 \leq \gamma < 1$, entsteht ein endlicher Horizont. Wählt man $\gamma = 0$, so hat der Agent keine „Weitsicht“ und maximiert den zu erwartenden Reward im nächsten Schritt. Mit $\lim \gamma \rightarrow 1$ hingegen werden auch die noch zu erwartenden Rewards mit in die Berechnung einbezogen, und der Agent bekommt ein gewisses Maß an „Weitsicht“. Die obere Schranke für den verzögerten Reward ist der durchschnittliche Reward aus Gleichung 5.4. γ wird auch als *Discount-Faktor* bezeichnet.

Die Wertefunktion $V^{\pi}(s)$ kann durch die Berechnung der Bellmann-Gleichung gelöst werden:

$$
\begin{aligned}
V^{\pi}(s) &= E_{\pi}\left\{R_t|s_t = s\right\} \\
&= E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s\right\} \\
&= E_{\pi}\left\{r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}|s_t = s\right\} \\
&= \sum_a \pi(s,a)\sum_{s'}\mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_{t+1} = s'\right\}\right] \\
&= \sum_a \pi(s,a)\sum_{s'}\mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s')\right]
\end{aligned}
\tag{5.6}
$$

mit $\mathcal{P}^a_{ss'} = Pr\{s_{t+1} = s'|s_t = s, a_t = a\}$ der Wahrscheinlichkeit von Zustand s' als Folgezustand bei aktuellem Zustand s und Aktion a und $\mathcal{R}^a_{ss'} = E\{r_{t+1}|s_t = s, a_t = a, s_{t+1} = s'\}$ dem zu erwartenden Reward.

5.1.2 Optimale Wertefunktion

Die optimale Wertefunktion, repräsentiert die bestmögliche Lösung eines Reinforcement-Agenten für ein Problem. Sie stellt das Ergebnis der optimalen Aktionsauswahl dar. Da V^{π} direkt von π abhängt, wird also die optimale Strategie benötigt, um die optimale Wertefunktion berechnen zu können. Eine Strategie π ist besser als eine zweite Strategie π', wenn der zu erwartende Return, also der Erwartungswert der Summe aller folgenden Rewards, in jedem Zustand größer ist, als der von π'. Die optimale Strategie $\pi \geq \pi'$ ist besser oder gleichwertig, wenn $V^{\pi} \geq V^{\pi'}$ für alle $s \in \mathcal{S}$ gilt. Mathematisch ist der Ausdruck $\pi \geq \pi'$ nicht ganz korrekt, da eine Strategie keinen reellen Wert hat. Es lässt sich aber eine partielle Ordnung verschiedener Strategien über die zugehörigen Wertefunktionen definieren. Definiert man den Operator $\geq$ über diese Ordnung, so ist die Aussage $\pi \geq \pi'$ zulässig. Die optimale Strategie wird auch mit π^*, und korrespondierend hierzu, die optimale Wertefunktion mit V^* bezeichnet. Sie ist definiert durch:

$$
\begin{aligned}
V^*(s) &= \max_{\pi} V^{\pi}(s) && (5.7)\\
&= \max_{\pi} E\left\{r_t + \gamma V^*(s_{t+1})\right\} && (5.8)\\
&= \max_{a\in\mathcal{A}}\left[r(s_t,a) + \gamma\sum_{s'\in\mathcal{S}}\delta(s_t,a)V^*(s')\right],\ \forall s_t \in \mathcal{S} && (5.9)
\end{aligned}
$$

Somit ergibt sich die optimale Strategie zu:

$$
\begin{aligned}
\pi^* &= \arg\max_{\pi} V^{\pi}(s)\ \ \forall s \in \mathcal{S} && (5.10)\\
&= \arg\max_{a\in\mathcal{A}}\left[r(s,a) + \gamma\sum_{s'\in\mathcal{S}}\delta(s_t,a)V^*(s')\right] && (5.11)
\end{aligned}
$$

5.1.3 Q-Lernen

Eine Alternative zu der Berechnung der vorgestellten Wertefunktion V^π stellt die Berechnung der Aktions-Wertefunktion $Q^\pi(s,a)$, auch *Q-Funktion*, dar. Sie verknüpft eine Aktion a in Zustand s mit dem Erwartungswert bei einer Strategie π. *Q-Lernen* wurde von Watkins [Watkins 1989] und Watkins und Dayan [Watkins and Dayan 1992] vorgestellt. Die Q-Funktion ist definiert durch:

$$Q^\pi(s,a) = E_\pi\{R_t|s_t=s,a_t=a\} \tag{5.12}$$

$$= E_\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t=s,a_t=a\right\} \tag{5.13}$$

Im Vergleich zur Zustands-Wertefunktion bietet die Q-Funktion einen entscheidenden Vorteil. Um bei der Zustands-Wertefunktion die nächste, bestmögliche Aktion auswählen zu können, muss eine Suche über alle Folgezustände ausgeführt werden, die den jetzigen Zustand in den mit der höchsten Bewertung überführt. Bei der Q-Funktion ist eine solche Suche nicht erforderlich, da die Aktionen direkt bewertet werden. Somit wird der Aufwand für die Aktionsauswahl erheblich reduziert.

Die optimale Q-Funktion ist definiert durch:

$$Q^*(s,a) = E\left\{r_{t+1}+\gamma\max_{a'}Q^*(s_{t+1},a')|s_t=s,a_t=a\right\} \tag{5.14}$$

$$= \sum_{s'}\mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'}+\gamma\max_{a'}Q^*(s',a')\right]. \tag{5.15}$$

Die Gleichungen 5.14 und 5.15 werden auch als Bellmann Gleichungen für die optimale Q-Funktion bezeichnet.

Die Wertefunktion $V^\pi(s)$ und $Q^\pi(s,a)$ sind voneinander ableitbar. So kann $Q^*(s,a)$ mittels $V^*(s)$ ausgedrückt werden:

$$Q^*(s,a) = \max_a E\{r_{t+1}+\gamma V^*(s_{t+1})|s_t=s,a_t=a\} \tag{5.16}$$

$$= \max_a\sum_{s'}\mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'}+\gamma V^*(s')\right] \tag{5.17}$$

oder anders

$$V^*(s)=\max_{a\in\mathcal{A}}Q^{\pi^*}(s,a). \tag{5.18}$$

5.1.4 Explorationstrategien

Eines der Probleme bei diesem Lernverfahren ist der Kompromiss zwischen Exploration und Evaluierung einer Strategie. Der Agent soll den Reward über einen möglichst langen Zeitraum maximieren. Hierfür muss er Aktionen, die zu einen hohen Reward führen, immer wieder ausführen. Auf der anderen Seite führt die permanente Wahl der gleichen Aktion möglicherweise dazu, dass eine bessere Lösung nicht gefunden – exploriert – wird. Diesen potentiellen Nachteil vermeidet ein so genanntes ϵ-*greedy* Verfahren für die Aktionsauswahl. Hierbei wird im Normalfall die Aktion mit dem besten Reward gewählt, und mit einer Wahrscheinlichkeit von ϵ zufällig eine der schlechter bewerteten Aktionen.

Eine andere Möglichkeit zur Aktionsauswahl bietet die die Boltzmann-Exploration. Hierbei werden den einzelnen Aktionen Wahrscheinlichkeiten anhand ihrer Q-Werte zugeordnet. Je größer der Q-Wert, desto

größer ist die Wahrscheinlichkeit für die Aktion. Die Aktionswahl wird dann durch die Wahrscheinlichkeit beeinflusst. Die Wahrscheinlichkeitsfunktion $P(s,a)$ ist definiert durch:

$$P(s,a) = \frac{e^{\frac{Q(s,a)}{T}}}{\sum_{a'} \left(e^{\frac{Q(s,a')}{T}} \right)} \tag{5.19}$$

mit T als Explorationstemperatur. Für kleine Werte von T werden Aktionen mit einem großen Q-Wert favorisiert, bei einem großen T wird die Auswahl immer zufälliger.

5.2 Lernverfahren

Ein direktes Lösen der Bellmann-Gleichung ist in der Regel nicht möglich. Dies liegt zum einen in der rekursiven Definition der Gleichung, deren Berechnung extrem kostenintensiv ist. Ein weiteres Problem besteht darin, dass die Anzahl der Zustände, die nötig sind um ein Modell komplett zu beschreiben, in der Regel zu groß ist. Somit ist das Modell nicht berechenbar, da sich nicht alle möglichen Zustände darstellen lassen. Darüber hinaus geht, wegen der Rekursion, die Anzahl der Zustände exponentiell in die Berechnung der Bellmann-Gleichung ein. Somit müssen Methoden entwickelt werden, die eine iterative Berechnung der Zustands-Wertefunktion bzw. Aktions-Wertefunktion ermöglichen, oder Berechnungen auf Schätzungen der Funktionen anstellen können. In den folgenden Abschnitten werden einige Varianten des selbstbewertenden Lernens vorgestellt, die eine solche Berechnung ermöglichen. Generell wird für die folgenden Betrachtungen von einem endlichen MDP ausgegangen. Damit ist verbunden, dass $\mathcal{S}$ und $\mathcal{A}(s),\ \forall s \in \mathcal{S}$ ebenfalls begrenzt sind.

5.2.1 Dynamische Programmierung

Unter dem Begriff *dynamische Programmierung* werden Algorithmen zusammengefasst, die die Lösung eines Problems beschleunigen können [Russel und Norvig 2003]. Bei Problemen, bei denen sich die Lösung in die Lösung von unabhängigen Teilproblemen aufteilen lässt, kann die Lösung des Gesamtsystems möglicherweise durch eine Partitionierung beschleunigt werden. Es gibt eine Vielzahl von Beispielen aus den unterschiedlichsten Bereichen der Informatik, in denen dynamische Programmierung mit Erfolg eingesetzt wird. In der Bildverarbeitung lassen sich Berechnungen mittels dynamischer Programmierung beschleunigen, wie z.B. in [Hübner 2006] und [van Meerbergen et al 2002] beschrieben. Oder auch die in [Sedgewick 2002] vorgestellten Verfahren zur effektiven Berechnung von *Fibonacci-Zahlen* wenden Methoden der dynamischen Programmierung an[2].

Die Berechnung der optimalen Policy π^* kann mittels dynamischer Programmierung gelöst werden, indem das Problem in zwei Teile zerlegt wird:

1. Evaluierung einer Policy
2. Verbesserung der Policy

Diese Schritte müssen iterativ wiederholt werden, um eine optimale Strategie zu berechnen:

$$\pi_0 \rightarrow V^{\pi_0} \rightarrow \pi_1 \rightarrow V^{\pi_1} \ldots \rightarrow \pi^* \rightarrow V^{\pi^*}$$

[2]Die Fibonacci-Folge $(f_0, f_1, \ldots)$ ist durch das rekursive Bildungsgesetz $f_n = f_{n-1} + f_{n-2}$ für $n \geq 2$ mit den Anfangswerten $f_0 = 0$ und $f_1 = 1$ definiert. Die Werte dieser Folge werden Fibonacci-Zahlen genannt.

Die iterative Update-Regel für die Zustands-Wertefunktion ist wie folgt definiert:

$$V_{k+1} = \sum_{a \in \mathcal{A}} \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V_k(s') \right] \qquad \forall s \in \mathcal{S}. \tag{5.20}$$

Die Verbesserung der Strategie erfolgt durch:

$$\pi_{k+1} = \arg\max_{a \in \mathcal{A}} \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V_k(s') \right] \qquad \forall s \in \mathcal{S}. \tag{5.21}$$

Wahrscheinlich ist jedoch bei der Strategie-Evaluierung mehr als ein Durchlauf erforderlich, um V_{k+1} aus V_k rekursiv zu berechnen. Somit kann bereits an dieser Stelle ein enormer Rechenaufwand erforderlich sein. Das größte Problem besteht jedoch darin, dass ein komplettes Welt-Modell benötigt wird, um die erforderlichen Berechnungen durchführen zu können.

Analog zu der oben vorgestellten Zustands-Wertefunktion lässt sich auch eine Update-Regel für die Aktions-Wertefunktion definieren:

$$Q(s,a) \leftarrow r + \gamma \max_{a \in \mathcal{A}} Q(s', a'). \tag{5.22}$$

Hierbei muss jedoch mit Schätzwerten für $Q(s,a)$ gerechnet werden. Dynamische Programmierung in Verbindung mit selbstbewertendem Lernen ist in [Barto et al. 1995] ausführlich untersucht worden.

Q-Lernen und Varianten davon sind die häufig eingesetzten Verfahren im Bereich des Reinforcement-Lernens. Q-Lernen ist ein Verfahren aus dem Bereich der *Monte-Carlo-Methoden* [Sutton and Barto 1998, Fishman 1999].

5.2.2 Monte-Carlo-Verfahren

Monte-Carlo-Verfahren [Sutton and Barto 1998, Fishman 1999] zeichnen sich durch die folgenden Eigenschaften aus:

- Es ist kein komplettes Welt-Modell für die Berechnung erforderlich
- Das Lernen erfolgt in Episoden

Episoden sind dabei Durchläufe, also eine Menge von Zuständen, Aktionen und Rewards, die in einem terminalen, oder auch absorbierenden, Zustand enden. Gelangt der Agent in einen solchen Zustand, so führt jede weitere Aktionsauswahl nur wieder in den selben Zustand, d.h. es finden keine weiteren Zustandsübergänge statt. Um die Episoden berechnen zu können, ist kein komplettes Welt-Modell mehr erforderlich, sondern es muss nur der Teil modelliert werden, der für die Berechnung der Episoden erforderlich ist. Dies ist in vielen Fällen möglich, in denen eine komplette Modellierung unmöglich ist.

Das Lernen besteht also aus einem Lernen aus Erfahrungen. Die Erfahrungen spiegeln sich dabei in den *Returns*, der Summe der Rewards, aus den einzelnen Episoden wieder. Beim *First-Visit* Monte-Carlo-Ansatz wird $V^\pi(s)$ berechnet, indem der Durchschnitt über die akkumulierten Returns nach dem ersten Auftreten (Visit) von s in einer Episode gebildet wird. Das Verfahren konvergiert zu $V^\pi(s)$, wenn die Anzahl der Visits von s gegen unendlich geht.

Zwei unterschiedliche Varianten von Monte-Carlo-Verfahren können unterschieden werden:

- *On-Policy*-Verfahren
- *Off-Policy*-Verfahren.

Bei den On-Policy-Verfahren wird eine Strategie verfolgt und diese gleichzeitig evaluiert. Bei den Off-Policy-Verfahren besteht die Möglichkeit, eine Strategie zu verfolgen, und eine Zweite zu evaluieren. Somit kann beispielsweise eine Zufallsstrategie verfolgt werden, während eine optimale Strategie evaluiert wird.

5.2.3 Temporal-Difference Lernen

Temporal-Difference Lernen (TD-Lernen) [Sutton 1988] beinhaltet eine der zentralen Ideen des selbstbewertenden Lernens. Es verbindet die Vorteile der Monte-Carlo-Methoden mit denen des dynamischen Programmierens. Diese Methoden lernen direkt aus den einzelnen Rewards und müssen nicht bis zu einem finalen Ergebnis warten. Dieses Verhalten wird auch als *Bootstrapping* bezeichnet. Außerdem ist kein Modell für die Dynamik der Welt mehr erforderlich, da sie durch das Bootstrapping aus den direkten Erfahrungswerten lernen können.

Die Update-Regel bei den Monte-Carlo-Methoden ist durch

$$V(s_t) \leftarrow V(s_t) + \alpha \left[R_t - V(s_t)\right] \tag{5.23}$$

mit α als einen konstanten Schrittweiten-Parameter, gegeben. Somit muss auf den Return R_t nach Zeitschritt t gewartet werden, der erst am Ende einer Episode berechnet werden kann. Im Gegensatz dazu ist die Update-Regel für TD-Lernen gegeben durch:

$$V(s_t) \leftarrow V(s_t) + \alpha \left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right], \tag{5.24}$$

mit γ als Discount-Faktor. Da R_t durch $r_{t+1} + \gamma V(s_{t+1})$ definiert ist, muss nicht mehr bis zum Abschluss einer Episode gewartet werden, sondern das Update kann bereits nach einem Schritt erfolgen. Diese Variante ist auch die einfachste TD-Methode, TD(0).

Wendet man dies auf das in Abschnitt 5.1.3 vorgestellte Q-Lernen an, so ergibt sich die Update-Regel nach einigen einfachen Umformungen zu:

$$Q^*(s,a) := Q^*(s,a) + \alpha \left[r(s^{'},a) + \gamma \max_{a^{'}} Q^*(s^{'},a^{'}) - Q^*(s,a)\right] \tag{5.25}$$

Dies ist die Definition für eine Update-Regel für ein Off-Policy-Lernverfahren. Off-Policy daher, da mit $\max_{a'} Q^*(s^{'},a^{'})$ gerechnet wird und nicht mit $Q^*(s^{'},a^{'})$ wie im folgenden Fall. Die Regel

$$Q^*(s,a) := Q^*(s,a) + \alpha \left[r(s^{'},a) + \gamma Q^*(s^{'},a^{'}) - Q^*(s,a)\right] \tag{5.26}$$

ist eine On-Policy-Variante des Q-Lernens und wurde von [Rummery and Niranjan 1994] entwickelt. Der Name des Verfahrens ist *Sarsa* und wurde von Sutton in [Sutton 1996] eingeführt, da der Algorithmus das Quintupel $(s, a, r(s^{'},a), s^{'}, a')$ beinhaltet.

Konvergenz von Q-Lernen

Die Frage die sich nun stellt ist: Kann die Konvergenz des Verfahrens, also die Entwicklung von $Q^{\pi} \rightarrow Q^{*}$ garantiert werden? Problematisch ist dies in einer nicht-deterministischen Umgebung, da es hierbei nicht zwangsläufig für jeden Übergang (s, a) auch immer den gleichen Reward ergibt. Somit ergibt sich immer eine „Regelabweichung“, also ein Fehler, der in die Update-Funktion mit einfließt und somit die Konvergenz zu einem festen Wert verhindert. Um dieses zu umgehen wird ein dynamischer Faktor eingeführt, der die Konvergenz ermöglicht. Somit ergibt sich Gleichung 5.26 zu

$$Q_n(s,a) \leftarrow Q_{n-1}(s,a) + \alpha_n \left[r(s^{'},a) + \gamma Q_{n-1}(s^{'},a^{'}) - Q_{n-1}(s,a) \right] \tag{5.27}$$

mit

$$\alpha_n = \frac{1}{1 + \text{Visit}_n(s,a)} \tag{5.28}$$

wobei $\text{Visit}_n(s,a)$ die Gesamtzahl der Besuche von (s,a) ist. Somit sinkt der Anteil des „Fehlers“ bei einem Update mit jedem Durchlauf, und eine Konvergenz kann unter den folgenden Bedingungen garantiert werden:

- Es gibt eine obere Schranke für die Rewards: $\forall s, a : |r(s,a)| \leq c$, mit c einer positiven Konstanten
- $Q(s,a)$ wird beliebig (endlich) initialisiert
- Der Discount-Faktor ist definiert durch : $0 \leq \gamma < 1$
- Die Lernrate ist definiert durch : $0 \leq \alpha < 1$
- Jedes Zustands-Aktions-Paar (s,a) wird unendlich oft besucht, $0 \leq \alpha_n < 1$ und $\sum_{i=1}^{\infty} \alpha_{n(i,s,a)} = \infty$, $\sum_{i=1}^{\infty} \left[\alpha_{n(i,s,a)}\right]^2 < \infty$ mit $n(i,s,a)$ als Variable für die i-te Iteration in der Aktion a in Zustand s ausgeführt wird.

Die Konvergenz des Verfahrens wurde zuerst in [Watkins 1989] und [Watkins and Dayan 1992] gezeigt und später von [Jaakkola et al. 1994] unter einem anderen Aspekt erneut bestätigt.

5.2.4 TD (λ)-Verfahren

Die bisher vorgestellten Algorithmen hatten nur eine „kleine“ Weitsicht, d.h. sie haben nur einen Schritt in die Zukunft geschaut. Daher werden diese Varianten des Temporal-Difference Lernens auch $TD(0)$ Methoden genannt. Es kann aber von Vorteil sein, wenn man weiter in die Zukunft schauen kann als nur einen Schritt. Dies ist mit den $TD(\lambda)$ Algorithmen möglich. $TD(\lambda)$-Methoden verwenden *Eligibility Traces* um mehr Weitsicht zu bekommen, als $TD(0)$ Varianten. Eligibility Trace heißt wörtlich übersetzt „Eignungs-Spur“.

$TD(\lambda)$ stellt eine generelle Form des vorgestellten Temporal-Difference-Lernens dar. Der Sarsa-Algorithmus (Gleichung 5.26) ist ein $TD(0)$ Algorithmus. Die verallgemeinerte Update-Regel für $TD(\lambda)$ ist definiert durch:

$$Q(s,a) := Q_{n-1}(s,a) + \alpha_n \left[r(s^{'},a) + \gamma Q_{n-1}(s^{'},a^{'}) - Q_{n-1}(s,a) \right] e(s) \tag{5.29}$$

Es wird eine Eignung als ein „Kurzeitgedächtnis“ für jeden Zustand berechnet. Eine Möglichkeit die Eignung eines Zustandes zu berechnen ist gegeben durch:

$$e(s) = \sum_{k=1}^{t} (\lambda\gamma)^{t-k} \delta_{s,s_k}, \text{ mit } \delta_{s,s_k} = \begin{cases} 1 & \text{falls } s = s_k \\ 0 & \text{sonst} \end{cases} \tag{5.30}$$

Für $\lambda = 1$ ergibt sich somit:

$$e(s) := \begin{cases} \gamma\lambda e(s) + 1 & \text{falls } s = \text{ aktueller Zustand} \\ \gamma\lambda e(s) & \text{sonst} \end{cases} \tag{5.31}$$

Sarsa λ

Die λ-Variante des in Abschnitt 5.2.3 vorgestellten Algorithmus mit Gleichung 5.26 ergibt sich zu:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t e_t(s,a), \qquad \forall s, a \tag{5.32}$$

mit

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \tag{5.33}$$

und

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 & \text{wenn } s = s_t \text{ und } a = a_t \\ \gamma\lambda e_{t-1}(s,a) & \text{sonst} \end{cases}, \qquad \forall s, a. \tag{5.34}$$

5.2.5 Hierarchisches Lernen

Die Methoden des *hierarchischen Lernens* versuchen, durch eine Zerlegung des Gesamtproblems in mehrere kleinere Einheiten eine effizientere und somit schnellere Konvergenz des Lernens zu erzielen. Hierbei gibt es verschiedene Ansätze, von denen zwei hervorstechende Varianten im Folgenden dargestellt werden sollen.

Für die Ansätze ist jedoch zunächst die Erweiterung des Markovschen-Entscheidungsprozesses erforderlich. Viele der folgenden Verfahren beruhen auf einem *Semi-Markov-Entscheidungs-Prozess* (*SMDP*). In Ergänzung zu dem in Abschnitt 5.1 vorgestellten MDP, wird bei einem SMDP auch die Zeit, die zwischen zwei, aufeinander folgenden Entscheidungen liegt, mit als Zufallsvariable in den Prozess einbezogen. Dass heißt, die Definition des MDP wird um die Funktion $f : s \times a \rightarrow \mathbb{R}$ erweitert, die die Wahrscheinlichkeitsverteilung $\mathcal{F}(t|s,a)$ definiert. $\mathcal{F}$ definiert, dass die nächste Entscheidung in t Zeiteinheiten erfolgt, unter der Voraussetzung, das Aktion a in Zustand s ausgeführt wurde. Somit muss eine neue Übergangsfunktion definiert werden, die sich aus $\mathcal{F}$ und $\mathcal{P}$ (aus Gleichung 5.6) zusammen setzt:

$$\mathcal{N}(t, s^{'}|s,a) = \mathcal{P}(s^{'}|s,a)\mathcal{F}(t|s,a). \tag{5.35}$$

$\mathcal{N}$ beschreibt also die Wahrscheinlichkeit, dass sich der Agent spätestens nach t Zeiteinheiten in Zustand $s^{'}$ befindet, nachdem in Zustand s Aktion a ausgewählt wurde. Die Reward-Funktion für einen SMDP besteht aus einem fixen Reward $r(s,a)$, der sich über die Aktionsauswahl ergibt und einem akkumulierten Reward $c(s, s^{'}, a)$ für die Zeit, die der Agent in Zustand s zwischen zwei Entscheidungs-Perioden verbleibt. Die Bellmann-Gleichung (Gleichung 5.6) ändert sich zu

$$V^*(s) = \max_{a \in \mathcal{A}} \left[r(s,a) + \sum_{s^{'},t} \gamma^t \mathcal{P}(s^{'}|s,a) V^*(s^{'}) \right], \qquad \forall s \in \mathcal{S} \tag{5.36}$$

und

$$Q^*(s,a) = r(s,a) + \sum_{s',t} \gamma^t \mathcal{P}(s'|s,a) \max_{a'} Q^*(s',a'), \qquad \forall s \in \mathcal{S} \text{ und } a \in \mathcal{A}. \tag{5.37}$$

Alle bisher vorgestellten Verfahren lassen sich in ihrer Anwendung von MDPs auf SMDPs übertragen.

Ein Überblick über einen Teil der im Folgenden vorgestellten Varianten ist in [Barto and Mahadevan 2003] zu finden.

Dekomposition

Eine Vielzahl von Methoden versucht, dass ursprüngliche Problem in kleine Sub-Probleme aufzuteilen, wozu es verschiedene Ansätze gibt. Eine Möglichkeit besteht darin, das Problem in immer feiner werdenden Granularitäten auf mehrere Lerner zu verteilen, wie z.B. in [Dayan and Hinton 1993] oder auch [Dean and Lin 1995].

Auch der Ansatz der *hierarchisch abstrakten Maschinen* (hierarchical abstract machines, HAMs) [Parr and Russell 1997] teilt ein Problem in Teilprobleme auf, die von mehreren Agenten gelöst werden, und deren Ergebnis anschließend fusioniert wird. Eine weitere Möglichkeit besteht in der Definition von Zwischen-Zielen, die sich in der entsprechenden Domäne leichter berechnen lassen, da nicht mehr alle Zustandsvariablen berücksichtigt werden müssen. Dies ist z.B. bei dem *MAXQ*-Algorithmus [Dietterich 2000, Dietterich 1998] der Fall. Hier wird das Problem, mit einem Taxi, einen Passagier aufzunehmen und ihn an einem Zielort abzusetzen in die Schritte (1) Passagier finden und Aufnehmen, und (2) zum Ziel fahren und Passagier absetzen, zerlegt. Diese können unabhängig voneinander gelernt werden.

Eine andere Variante besteht in der Identifikation von Zwischenzielen, die einen „Flaschenhals“ für den Agenten darstellen. Oftmals wird hierfür das Beispiel einer Tür zwischen zwei Räumen angeführt [Menache et al. 2002, Şimşek et al. 2005, McGovern and Barto 2001a] und [McGovern and Sutton 1998]. Um von einem Raum in den anderen zu gelangen, muss der Pfad des Agenten notwendigerweise durch die Tür verlaufen. Der Zustand „Tür“ wird also deutlich häufiger in den Episoden auftreten als alle anderen Zustände und stellt somit einen „Flaschenhals“ dar. Speichert man nun die Historie der Episoden in einem Graphen ab, so lassen sich mittels des *Q-Cut* Algorithmus aus [Menache et al. 2002] diese Zustände über einen *Max-Flow/Min-Cut* Algorithmus bestimmen[3]. Ein ebenfalls graphenbasierter Ansatz wird in [Şimşek et al. 2005] vorgestellt. Der *L-Cut* Algorithmus beruht auf einem lokalen Übergangsgraphen, der sich nur auf die letzten Erfahrungen des Agenten bezieht. Dieses spiegelt sich somit natürlich auch in einer Beschränktheit des Graphen wieder, in dem die Schnitte für die Unter-Ziele ermittelt werden. Dies soll laut den Autoren einen Vorteil gegenüber dem Q-Cut Verfahren bringen, da die (lokalen) Zielzustände nur in einer Subdomäne ermittelt werden müssen und nicht in Bezug auf den gesamten Zustandsraum. Diese, als „Engstellen“ bezeichneten Zustände lassen sich nach [McGovern and Barto 2001a] und [McGovern and Barto 2001b] über *Diverse Density* bestimmen. Diverse Density Verfahren sind Methoden aus dem Bereich des *multiple instance learning*. Hierbei wird versucht anhand von positiven und negativen Beispielen einen Algorithmus zu trainieren. Es wird also versucht, aus verschiedenen Instanzen zu lernen.

[3] Der Max-Flow/Min-Cut Algorithmus ist ein Verfahren aus der Graphentheorie [Ahuja et al. 1993]

Makro-Aktionen

Ein weitere Variante mit der versucht wird, das Lernen zu beschleunigen, besteht in der Verwendung von *Makro-Aktionen*. Hierbei werden mehrere elementare Aktionen, zu einem Aktionsblock zusammengefasst. Ein solcher Block ist ein Makro und wird als neue Aktion in dem entsprechenden Zustand eingefügt. Hierbei findet die Zeit, die als zusätzliche Komponente in den SMDP eingeführt wurde, Beachtung. Makros oder auch *Options*, wie sie von [Sutton et al. 1999] benutzt werden, werden beschrieben als

> „. . . , we use the term options for our generalization of primive actions to include temporally extended courses of action."

Makros sind also eine Generalisierung einer Menge von primitiven Aktionen, die zu einer temporal extendierten Operation zusammengefasst werden. Die Optionen bestehen aus den den folgenden drei Komponenten:

Einer Policy $\pi_O : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$

Einer Bedingung für die Terminierung $\beta : \mathcal{S}^+ \rightarrow [0, 1]$

Einer Menge von Initialzuständen $\mathcal{I} \subseteq \mathcal{S}$

mit $\mathcal{S}^+$ als Menge aller Zustände, inklusive der Terminalen-Zustände.

Eine Option ist somit das Tripel $\langle \mathcal{I}, \pi_O, \beta \rangle$. Sie ist im Zustand s_t verfügbar, falls $s_t \in \mathcal{I}$. Wenn die Option gewählt wird, werden die Aktionen korrespondierend zu der Strategie π_O der Option ausgewählt, bis die Option gemäß β terminiert[4]. In [Sutton et al. 1999] werden z.B. Makro-Aktionen benutzt um einen Zielzustand zu finden, der sich in einer Gitternetzumgebung befindet, die aus vier miteinander verbunden Räumen besteht. Die Autoren zeigen, dass der Einsatz von Makro-Operationen hier einen Geschwindigkeitszuwachs gegenüber dem Einsatz von rein elementaren Operationen bietet.

Dieser Ansatz wurde auch in [McGovern and Sutton 1998] untersucht. Zwei Fälle wurden dabei unterschieden. Zum einen die Beschleunigung des Lernens (die Konvergenz von V^π zu V^*) durch die Benutzung von Makro-Operationen, und zum anderen das explorative Verhalten des Agenten. Die Änderungen gegenüber einem Standardverfahren waren in beiden Fällen signifikant. Jedoch war der Effekt in Bezug auf die Beschleunigung des Lernens deutlich größer, als der in Bezug auf die Exploration. Für die Analyse wurde zum einen ein Szenario gewählt, bei dem ein Agent in einer Gitternetzwelt einen bestimmten Zielzustand erreichen muss. Hierfür wurde bereits in [McGovern et al. 1997] gezeigt, dass sich der Lernvorgang mittels Makro-Aktionen beschleunigen lässt. Bei einem zweiten Szenario musste ein Roboter zwischen zwei durch eine Tür verbundenen Räumen navigieren. In beiden Fällen waren ungefähr die gleichen Effekte der Beschleunigung und des explorativen Verhaltens zu beobachten.

Eine Methode, Makro-Aktionen automatisch zu generieren, wird in [Pickett and Barto 2002] vorgestellt. Hierbei wird nach Gemeinsamkeiten in verschieden Policies für eine Aufgabe gesucht, um aus den Gemeinsamkeiten eine Makro-Operation zu generieren. Auch in diesem Fall belegen die Autoren, u.a. an einem Gridworld-Beispiel, dass sich durch die Anwendung ihres Algorithmus der akkumulierte durchschnittliche Reward gegenüber anderen Verfahren deutlich steigern lässt.

[4] Da β eine Wahrscheinlichkeitsverteilung darstellt, terminiert ein Makro mit der Wahrscheinlichkeit $\beta(s_t)$.

5.3 Anwendbarkeit des selbstbewertenden Lernens auf das Lernen von Griffen

Im folgenden Abschnitt soll kurz auf die Anwendbarkeit von Reinforcement-Lernen auf das Problem der Generierung von Griffen eingegangen werden. Es soll nur gezeigt werden, dass eine Anwendbarkeit prinzipiell möglich ist und welche Einschränkungen gelten. Eine detaillierte qualitative und quantitative Untersuchung wird in Abschnitt 6.6 gegeben.

Das Lernen von Griffen lässt sich sehr gut mittels eines MDP beschreiben. Die einzelnen Zustände lassen sich durch die Position der Hand relativ zum Objekt, den Spreizwinkel der Hand (siehe Kapitel 2.3) und die Winkel der Fingergelenke, definieren. Eine Episode wird dabei als abgeschlossen definiert, wenn ein Greifversuch durchgeführt wurde. Somit stellt ein Griff einen absorbierenden Zustand dar. Aus den Definitionen folgt, dass Zustandsübergänge durch eine Veränderung der Position der Hand, durch eine Änderung des Spreizwinkels oder durch Zugreifen erfolgen können.

Methoden aus dem Bereich des hierarchischen Lernens sind prinzipiell anwendbar. Da der Suchraum, der durch die Größe des Objekts definiert ist, relativ groß sein kann, ist es sogar sinnvoll, solche Methoden einzusetzen. So wäre z.B. denkbar, dass man über den Ansatz der Dekomposition, zunächst grob ermittelt, in welchen Regionen eines Objekts Griffe zu finden sind, um anschließend diese Regionen genauer zu untersuchen. Auch die Anwendung von Makro-Aktionen in diesen Regionen ist möglich, um Griffe zu generieren. Automatisches Auffinden von Zuständen die einen „Flaschenhals“ im Zustandsraum darstellen, dürfte jedoch schwierig sein.

Wie in Kapitel 6.8 gezeigt wird, gibt es teilweise größere Regionen, in denen kein stabiler Griff gefunden werden kann. Somit gibt es nicht *einen* Zustand, wie z.B. die Tür in den oben beschriebenen Szenarien, der einen Flaschenhals darstellt, sondern eine relativ große Menge von Zuständen. Auch der Einsatz von Makro-Operationen, welche die Position deutlich verändern, kann problematisch sein, da hierdurch möglicherweise Regionen, in der viele stabile Griffe gefunden werden können, übersprungen werden.

Kapitel 6

Generierung von Griffen

Dieses Kapitel befasst sich mit der Berechnung von Griffen. Es wird zunächst eine Übersicht über in der Literatur beschriebene Verfahren gegeben. Da in dieser Arbeit Griffe mit Hilfe einer Simulation berechnet werden, wird im zweiten Abschnitt die entwickelte Simulationsumgebung beschrieben. Anschließend werden zwei, im Rahmen dieser Arbeit entwickelte Verfahren, vorgestellt. Hierbei liegt der Fokus auf der automatischen Berechnung von Griffen mittels selbstbewertendem Lernen. Für eine Auswahl von Objekten wird das Verfahren zur Griffgenerierung mittels Reinforcement-Lernen evaluiert. Abschließend erfolgt eine Bewertung der Ergebnisse.

6.1 Verfahren für die Generierung von Griffen

In der Literatur werden die Verfahren für die Berechnung und Evaluation von Griffen häufig miteinander gekoppelt dargestellt. Einige Verfahren für die Generierung von Griffen wurden deshalb bereits in Kapitel 4.1 vorgestellt. Der folgende Abschnitt befasst sich daher nur mit Verfahren, die sich ausschließlich mit der Berechnung von Griffen auseinander setzen. Ein Überblick über Greifen mit Roboterhänden ist in [Bicchi and Kumar 2000] zu finden.

Eine Vielzahl von Verfahren konzentriert sich ausschließlich auf das Finden von Kontaktpunkten, die einen stabilen Griff ermöglichen. Die frühen Arbeiten von Nguyen [Nguyen 1986, Nguyen 1988] beschreiben das Problem noch mit Hilfe von geometrischen Bedingungen. Spezielle Verfahren wurden zum Finden von Griffen mit vier Kontaktpunkten entwickelt, wie z.B. in [Borst et al. 1999]. Hier wird der erste Kontaktpunkt bei einem Objekt beliebig gewählt, der zweite Kontakt gegenüber dem ersten gesetzt und Kontaktpunkt drei und vier werden über den Mittelpunkt der beiden ersten Punkte errechnet. Die Punkte werden anschließend in Bezug auf bestimmte Kriterien, wie z.B. Erreichbarkeit durch die Roboterhand evaluiert und verworfen oder anschließend bezüglich ihrer Stabilität begutachtet. Durch Experimente konnten die Autoren zeigen, dass das Verfahren in der Lage ist, effizient Griffkandidaten zu generieren. Ähnliches gilt für das in [Niparnan and Sudsang 2004] vorgestellte Verfahren. Hierbei wird nach vier Kontaktpunkten bei einem Polygon-Modell des zu greifenden Objekts gesucht. Laut den Autoren begrenzt der Detailgrad des Modells oft die Berechenbarkeit. Diese Limitierung soll mit dem vorgestellten Verfahren aufgehoben werden. Die Suche wird hierzu durch die Aufteilung des Suchraumes effizient gestaltet, so dass eine schnelle Berechnung von Griffen möglich ist. Eine Methode für die Berechnung von Griffen mit n-Fingern oder Kontaktpunkten wird in [Liu and Lam 2003] beschrieben. Auch hierbei wird das Problem durch eine effiziente Suche, allerdings im Grasp-Wrench-Space, gelöst. Mit dieser Methode können Griffe mit und ohne Reibung berechnet werden.

Weitere Ansätze für die Berechnung von Griffpunkten sind in [Cheong and van der Stappen] und [Mirtich and Canny 1994] zu finden. Konkrete Einschränkungen, die sich z.B. aus dem Aufbau des ein-

gesetzten Manipulators ergeben, werden hierbei jedoch nicht berücksichtigt. Im Folgenden werden einige Ansätze vorgestellt, die sich durch besondere Ansätze, wie z.B. eine intelligente Griffplanung, Berechnung durch Lernen, etc., hervorheben.

Miller et al. beschreiben in [Miller et al. 2003] eine Variante für die Berechnung von Griffen auf Basis von Form-Primitiven. Um das Dimensionsproblem bei der Griffplanung zu reduzieren, werden Objekte als Aggregate aus geometrischen Primitiven (Quader, Kugel, Kegel, Zylinder) definiert. Für jedes der Primitive wird eine initiale Handkonfiguration gewählt, von denen für die BarrettHand laut der Autoren vier existieren. Die Startpositionen für die Greifversuche werden auf der Basis von Regeln gewählt. Kollidiert die Hand bereits in der Startkonfiguration mit dem Objekt, wird der Griff als unbrauchbar verworfen. Andernfalls wird die Distanz zwischen Hand und Objekt so lange reduziert, bis eine Kollision erfolgt. Die Finger schließen sich um das Aggregat und der Griff wird evaluiert. Mittels dieser Heuristik war es möglich, in relativ kurzer Zeit eine Vielzahl von Griffen für Objekte mit unterschiedlicher Komplexität zu generieren und zu evaluieren.

In Verbindung mit dem Humanoiden-Roboter AMAR [Asfour et al. 2006a] wird in [Morales et al. 2006a] und [Morales et al. 2006b] die Berechnung von Griffen beschrieben. Die Autoren entwickeln keinen neuen Ansatz, sondern verwenden das in [Miller et al. 2003] vorgestellte Verfahren und adaptieren es für AMAR. Lediglich die Integration in ein Gesamtsystem mit Objektlokalisierung, Objekterkennung, Griffgenerierung und Griffdatenbank war in dieser Form bisher nicht beschrieben worden.

Von Pollard [Pollard 1994, Pollard 1996] wurde ein Verfahren entwickelt, das in der Lage ist, Griffe für neue Objekte aus prototypischen Griffen zu generieren. Hierbei wird ein Griff der speziell für eine Aufgabe generalisiert, so dass er anschließend als Prototyp für eine Menge von Griffen stehen kann. Die Generalisierung erfolgt über den Grasp-Wrench-Space des Griffs, der nur durch einen Satz von Kontaktpunkten repräsentiert wird. Unter Beachtung verschiedener Randbedingungen kann der Griff zu einem generalisierten Prototyp expandiert werden. Da es möglich ist, diese Expansion offline durchzuführen, kann das Verfahren eingesetzt werden, um neue Griffe on-line zu generieren.

Ein Ansatz für die Planung von Pinzetten- und Präzisions-Griffen wird in [Borst et al. 2002] vorgestellt. Der Algorithmus läuft dabei in zwei Phasen ab. In der der ersten Phase werden Sätze von Kontaktpunkten berechnet, die einen stabilen Griff gewährleisten. In einer zweiten Phase wird versucht, eine Position und Konfiguration auf der Basis eines kinematischen Modells einer Roboterhand für einen Satz von Kontaktpunkten, zu finden. Das Verfahren beruht auf einem Optimierungsproblem, das die Kinematik der Roboterhand als Randbedingung berücksichtigt.

Ein Verfahren für die Bestimmung von Griffen für eine Übergabe-Operation wird in [Kim et al. 2004a] beschrieben. Die Objekte werden mit einem Zwei-Finger-Parallelgreifer aufgenommen und übergeben. Für die Evaluation der Übergabe-Operation wird der Griff des Objekts mittels eines geometrischen Modells betrachtet. Ebenfalls mit Hilfe des Modells werden die möglichen Griffpositionen für einen Abnehmer evaluiert. Eine Bewertung des Griffs erfolgt nicht. Nur einige Randbedingungen, wie z.B. ein Sicherheitsabstand zwischen Roboter- und menschlicher Hand müssen von dem System beachtet werden. Das System wurde mittels einer Simulation getestet und war in der Lage, Griffe für Übergabe-Operationen zu erzeugen. Eine quantitative oder qualitative Analyse der erzeugten Griffe nahmen die Autoren nicht vor.

Pelossof et al. [Pelossof et al. 2004] beschreiben ein Verfahren für die Berechnung von Griffen mit Hilfe einer *Support Vector Machine* (*SVM*). Die SVM ist mittels eines überwachten Lernverfahrens in der Lage, Aussagen über die Qualität von Griffen für einen mathematisch beschreibbaren, geometrischen Körper

zu treffen. In dem konkreten Fall werden verschiedene Superellipsoide eingesetzt. Als Greifer kommt eine BarrettHand zum Einsatz. Das Problem der Griffberechnung wird von den Autoren als zehndimensionales Problem aufgefasst. Hierbei entsprechen sechs Dimensionen der Translation und Orientierung der Hand und vier Dimensionen den Motorwerten der BarrettHand. Das Problem wird jedoch auf vier Dimensionen reduziert, da sich einige vereinfachende Annahmen treffen lassen. So liegt die Orientierung der Handfläche immer parallel zu der Oberfläche des Objekts. Die Hand nähert sich dem Objekt bis eine Kollision erfolgt, anschließend werden die Finger zu einem Griff um das Objekt geschlossen. Somit sind nur noch der Spreizwinkel der Hand, der Startpunkt der Hand, sowie der Roll-Winkel der Hand, bzw. die Orientierung des „Daumens“ für die Beschreibung relevant.

In [Rössler 2001] und [Rössler et al. 2002] beschreiben die Autoren einen Ansatz, bei dem ein Robotersystem mittels selbstbewertendem Lernen Griffe für Objekte lernt. Die Objekte bestehen aus Baufix-Bauteilen. Der Roboter ist dabei nur mit einem Zwei-Finger-Parallelgreifer ausgestattet. Für die Berechnung von Griffkandidaten wird nur der Umriss des Objektes betrachtet und somit das Problem auf ein zweidimensionales reduziert. Es werden zwei unterschiedliche Lerner trainiert. Zum einen, ein Lerner für *globale* Kriterien, der in der Lage ist, die optimale Position für einen Griff (Translation entlang der X- und Y-Achse) zu finden, zum anderen ein Lerner für *lokale* Kriterien. Der zweite lernt die Orientierung der Hand (Roll-Winkel) relativ zum Objekt. Die Kräfte, die beim Anheben des Objekts gemessen werden, dienen dabei als Reinforcement-Signal für den Positions-Lerner. Eine Lageveränderung des Baufix-Aggregates nach dem Ablegen des gegriffenen Objekts dient als Eingabesignal für den lokalen Lerner. Die durchgeführten Experimente zeigen, dass das entwickelte Verfahren in der Lage ist, Griffe für unbekannte Objekte zu lernen und dieses Wissen auch über Objekte zu generalisieren.

Auch in [Dehm 2006] werden mittels einer BarrettHand und Reinforcement-Lernen Griffe für Objekte generiert. Hierbei wurde, in Anlehnung an die Arbeit von [Rössler 2001], das Lernen auf zwei Lerner aufgeteilt. Zum einen wird ein Lerner für die Orientierung und den Spreiz-Winkel (SPROT) eingesetzt, und zum anderen einen Lerner für die Position. Das System wurde mit dem Service-Roboter TASER evaluiert. Die Rückkopplung für die Lerner erfolgt visuell, über eine Auswertung der Bilder einer Handkamera. Es konnte mit der Arbeit gezeigt werden, dass das Verfahren prinzipiell anwendbar ist, dass aber durch die schlechte Sensorik der BarrettHand und das Fehlen eines Kraftdrehmomentsensors starke Einschränkungen bei der Berechnung von Griffen zu beachten sind.

Gorce und Rezzoug [Gorce and Rezzoug 2004] beschreiben einen anderen Ansatz. Mit Ihrer Methode ist es möglich, zu einem Satz von Kontaktpunkten (zwei bis fünf Punkte) eine gültige Handkonfiguration (mit bis zu fünf Fingern) und Position zu finden. Es gelten allerdings Einschränkungen für die Menge der Kontaktpunkte. Es darf nur ein Kontakt pro Finger existieren, über die Menge der Kontakte ist ein stabiler Griff definiert und jeder Kontakt ist bereits mit einem Finger assoziiert. Wie aus den Restriktionen schnell ersichtlich wird, können somit nur Präzisionsgriffe berechnet werden. Die Berechnung erfolgt über ein zweistufiges Modell, bei dem in einem ersten Schritt die inverse Kinematik der Hand gelernt wird. In einem zweiten Schritt wird, unter der Berücksichtigung von Rauschen in den gegebenen Informationen, die Position der Finger optimiert und die Position der Hand bestimmt. In beiden Phasen wird ein neuronales Netz eingesetzt, um die entsprechenden Parameter zu generieren.

Alle der bisher vorgestellten Verfahren haben einen Nachteil: Sie sind nur in der Lage, jeweils eine spezielle Art von Griffen zu berechnen. Dies sind in den meisten Fällen entweder die so genannten *Power*griffe, mit möglichst vielen Kontaktpunkten zwischen Hand und Objekt, oder die Präzisionsgriffe, bei denen jeder Finger maximal einem Kontaktpunkt zum Objekt hat. Es ist kein Verfahren dabei, dass die Möglichkeit bietet, alle Arten von Griffen zu berechnen. Zudem erfolgt die Auswahl der Griffpositionen zumeist über ein Suchverfahren, das in der Lage ist, entsprechende Kontaktpunkte zu generieren. Es gibt

nur wenige Verfahren, die mit Methoden aus dem Bereich des maschinellen Lernens arbeiten, um Griffe zu generieren.

6.2 Simulation

Mit Hilfe einer Simulation ist es möglich, Griffe für Objekte zu berechnen und zu validieren. Sie kann zu einem enormen Geschwindigkeitszuwachs für die Entwicklung neuer Modelle – gegenüber Experimenten mit realen Robotern – führen, da keine Rücksicht auf die Geschwindigkeit des realen Systems oder sonstige Einschränkungen (mit Ausnahme der Kinematik) genommen werden muss. Ein weiterer Vorteil besteht darin, dass in der Simulation verschiedene Greifwerkzeuge miteinander vergleichbar sind. Es muss lediglich ein geometrisches Modell mit einer entsprechenden Vorwärtskinematik in der Simulation kombiniert werden. Auch stellen die Eigenschaften der Objekte keine Einschränkungen dar, da es höchstens zu „virtuellen Scherben“ kommen kann, falls die erforderlichen Kräfte außerhalb der maximal zulässigen Kraft für ein Objekt liegen.

Die hier vorgestellte Simulationsumgebung ist eine Eigenentwicklung. Sie wurde unter den Gesichtspunkten der einfachen Wart- und Bedienbarkeit, sowie einer generischen Architektur entwickelt, die ein leichtes Hinzufügen von weiteren Objekten ermöglicht. Es wurde bewusst auf eine Simulation der Dynamik zwischen den Objekten verzichtet. Es wurde lediglich die Gewichtskraft des Objekts als zusätzliche, statische Komponente zu den Kräften, welche die Finger ausüben, in die Berechnung einbezogen.

Der folgende Abschnitt gibt zunächst eine Übersicht einer Auswahl existierender Software zur Simulation von Robotern sowie von Systemen, mit denen sich eine Simulation aufbauen lässt. Anschließend wird die für diese Arbeit entwickelte Simulationsumgebung vorgestellt.

6.2.1 Existierende Simulationsumgebungen

Es existieren einige Simulationsumgebungen für Roboter. Diese weisen jedoch Defizite auf. Sie beschränken sich meist auf die Simulation von Arbeitsabläufen oder auf die Simulation von Roboterzellen, wie sie in der Industrie eingesetzt werden. Einige dieser Systeme bieten die Möglichkeit, Industrieroboter in der Simulation zu programmieren (oft auch als Offline-Programmierung bezeichnet) und anschließend das Ergebnis auf den realen Roboter zu übertragen. Bei vielen Produkten können physikalische Rahmenbedingungen, wie z.B. Schwerkraft, in die Simulation einfließen. Diese Modelle reichen in der Regel jedoch nicht aus um die Kräfte, die bei einem Griff auftreten können, exakt zu bestimmen.

Die Software IGRIP der Firma DELMIA [Delmia 2007], ROPSIM von CAMELOT Robotics [Camelot 2007], Actin von ENERGID [Energid 2007], Flow Software Technologies Workspace5 [Flow Software Technologies 2007] oder auch EASY-ROB von dem gleichnamigen Unternehmen [Easy Rob 2007] sind Beispiele für derartige Produkte. Sie sind vornehmlich für die oben genannten Szenarien konzipiert. Es können zwar eigene Roboter und Programme definiert werden, dies ist aber mit einem entsprechend hohen Aufwand verbunden.

Diese Programme sind kommerzieller Natur und die Lizenzen kostspielig. Microsoft bietet mit dem *Microsoft Robotics Studio* [Microsoft 2007] eine frei verfügbare Software für die Simulation von Robotern an. Hierbei können eigene Modelle integriert, sowie die Physik in einer entsprechenden Umwelt simuliert werden. Für die Beschleunigung der Physiksimulation bietet Microsoft die Option an, auf eine Hardwarebeschleunigung mittels *PhysX* der Firma Ageia Technologies zurückzugreifen. RoboSiM, ein in Java

geschrieben Simulator für Roboter [Speck und Klaeren 1999] und eine Matlab Toolbox [Corke 1996] sind Beispiele für freie Software, die für die Simulation von Robotern und Dynamik geeignet sind.

Eine weitere Möglichkeit besteht in der Verwendung von Systemen, die für die Entwicklung von Computerspielen gedacht sind. Hierbei existieren viele kommerzielle Produkte. Eine Übersicht ist bei [Devmaster 2007] zu finden. Daneben gibt es auch viele kostenlose Varianten, die einen enormen Funktionsumfang bieten. So ermöglicht z.B. die WildMagic Bibliothek [Eberly 2004, Eberly 2001, Geometrictools 2007] eine Simulation der Dynamik für feste Körper. Eine Kollisionserkennung ist in die Bibliothek integriert. Mittels der Dynamik und der Kollisionserkennung sowie eines physikalischen Modells kann ein elastischer Stoß zwischen Objekten berechnet werden. Auch kinematische Ketten lassen sich als Modelle in die Berechnung einbringen.

Multifinger-Roboterhände sind in keiner der genannten Anwendungen vorhanden. Die entsprechenden Modelle und Kinematiken müssen vom Benutzer in die Software integriert werden. Dies ist aber durch eine relativ gute Dokumentation mit akzeptablem Aufwand durchführbar.

Das als freie Software von Andrew Miller entwickelte Programm „GraspIt!“ [Miller and Allen 2000, Miller 2001, Miller 2005] ist als Einziges speziell für die Simulation von Roboterhänden und die Berechnung von Griffen konzipiert. Es lassen sich aber nicht nur künstliche Hände, sondern alle Arten von Robotern simulieren. Die Software bietet sowohl Unterstützung für die Berechnung von allgemeinen dynamischen Kräften, die während einer Kollision von Objekten auftreten, als auch speziell für die Berechnung von Kräften, die bei einem Griff auftreten. Die Modelle der Roboter können als 3D-Modell im OpenInventor-Format angegeben werden[1]. Die kinematische Beschreibung erfolgt über eine Textdatei, die von dem System gelesen und in ein entsprechendes kinematisches Modell umgesetzt wird. Leider ist die Positionierung der Objekte für den Benutzer nicht intuitiv verständlich und somit schwer zu realisieren. Da sie außerdem über ein grafisches Interface erfolgt, ist eine genaue Positionierung kaum möglich. Ein Verfahren für die automatische, intelligente Berechnung von Griffen ist nicht in das System integriert. Ein Vergleich von Griffen hinsichtlich ihrer Qualität fehlt ebenfalls. Die komplexe Simulation der Dynamik macht das System fehleranfällig und schwer zu bedienen.

Durch die Berechnung von Griffen in einer Simulation können unterschiedliche Probleme auftreten: Möglicherweise sind die Ergebnisse aus der Simulation nicht direkt auf den realen Roboter übertragbar oder enthalten Berechnungsfehler, die aus numerischen Problemen resultieren. Ein weiterer Nachteil ergibt sich aus der Art und Weise, wie die Kontaktpunkte berechnet werden. Da sie über eine Kollisionserkennung der geometrischen Modelle erfolgt [Eberly 2002], steigt der Aufwand mit der Anzahl, der für die Modelle verwendeten Polygone. Je komplexer und detaillierter die Modelle sind, umso länger dauert die Berechnung der Kontaktpunkte. Dies lässt sich zu Lasten der Genauigkeit umgehen, indem bei der Modellierung auf Details verzichtet wird. Dies führt aber unter Umständen dazu, dass sich die Ergebnisse der Simulation nicht auf die Realität übertragen lassen. Weitere Modellfehler, wie z.B. eine fehlerhafte Berechnung der Massenverteilung, können in der Realität zu Problemen bei der Umsetzung der Griffe führen. Auch numerische Probleme, z.B. bei der Berechnung der Kontaktpunkte, oder der Analyse der erforderlichen Griffkräfte, können zu Schwierigkeiten bei der Umsetzung der Simulationsergebnisse führen.

[1] OpenInventor ist ein Dateiformat dass von SGI für die Grafikworkstations entwickelt worden ist [SGI 1994, SGI 2007].

6.2.2 Simulationsumgebung für die Berechnung von Griffen

Die für diese Arbeit entwickelte Simulationsumgebung und das entsprechende Bedienkonzept sind eine Eigenentwicklung. Die Software baut aber auf verschiedenen Softwarebibliotheken auf.

Eine vereinfachte Übersicht über die Softwarearchitektur der Simulationsumgebung ist in Abbildung 6.1 dargestellt.

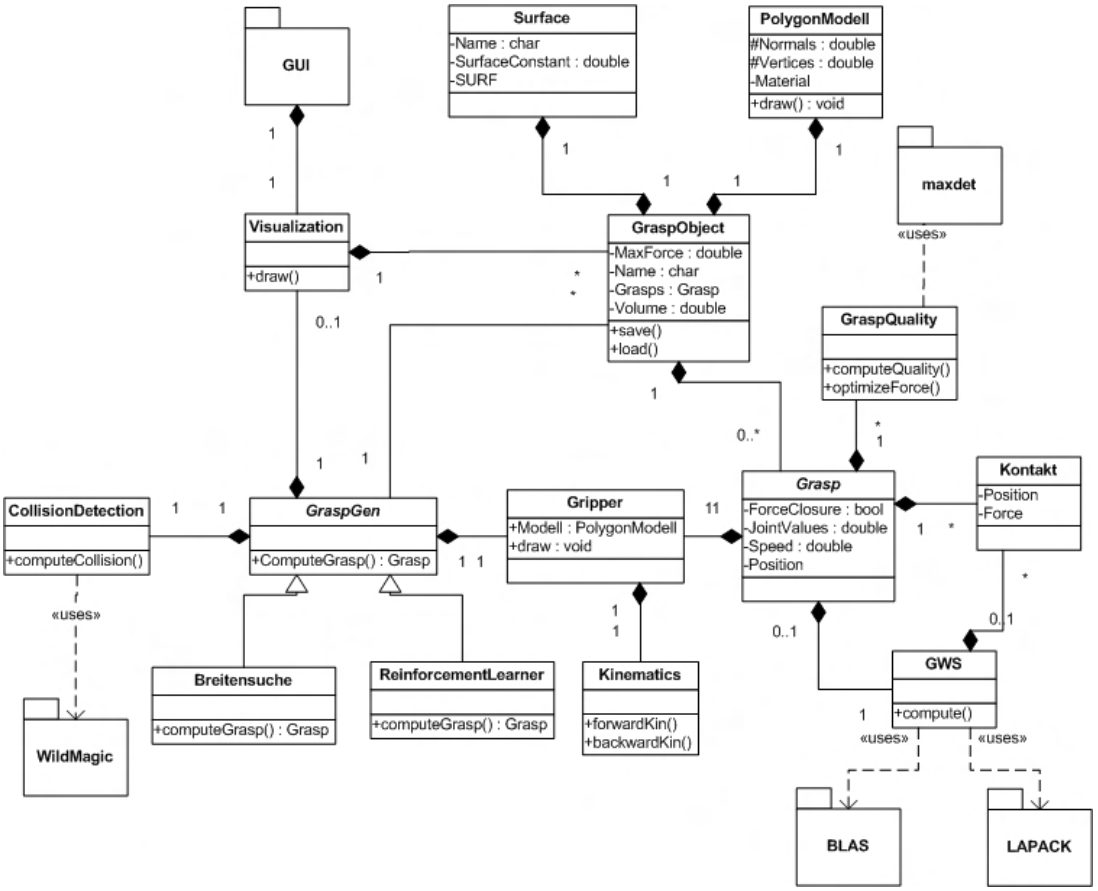


Abbildung 6.1: Vereinfachte Softwarearchitektur der Simulationsumgebung.

Kern der Simulation ist das Objekt, das gegriffen wird (GraspObject). Ihm wird eine Oberfläche und ein Polygonmodell zugeordnet. Jedem Objekt kann eine beliebige Anzahl von Griffen zugeordnet werden. Ein Griff besteht aus einer Menge von Kontakten, die zu einem Grasp-Wrench-Space (GWS) zusammengefasst werden. Die Kontakte werden über die Kollisionserkennung (CollisionDetection) berechnet, welche die WildMagic-Bibliothek [Eberly 2004, Eberly 2001, Geometrictools 2007] benutzt. Die Berechnung des GWS erfolgt mit Hilfe der BLAS- [Blackford et al. 2002, Dongarra 2002] und LAPACK-Bibliothek [Anderson et al. 1999]. Die Qualität eines Griffs berechnet sich aus dem in Kapitel 4 entwickelten Qualitätsmerkmal. Die Berechnung der Griffkräfte erfolgt mit Hilfe der maxdet-Bibliothek [Wu et al. 1996]. Für die Visualisierung ist eine Graphische-Benutzerschnittstelle entwickelt worden (GUI). Sie wird in den Prozess der Griffberechnung über ein Modul zur Visualisierung eingebunden. Somit es auch möglich Griffe ohne eine GUI, also ohne eine entsprechende Visualisierung, zu berechnen. Für die Berechnung der Griffe sind verschiedene Algorithmen entwickelt worden, von denen zwei in

der Abbildung exemplarisch dargestellt sind. Zum einen ein Lernverfahren und zum anderen ein Verfahren auf der Basis einer Breitensuche. Sie implementieren das GraspGen-Interface und sind somit einfach anzubinden. Die genauen Definitionen der Objekte, der Greifer und der Kollisionserkennung, werden in den folgenden Abschnitten gegeben.

6.2.2.1 Definition von Objekten

Die entwickelte Simulationsumgebung beschränkt sich auf die Betrachtung von starren Körpern. Elastische Körper können in der Simulation bis zu einem gewissen Grad betrachtet werden. Dies wirkt sich jedoch, wie in Abschnitt 7.2.8 gezeigt wird, nachteilig auf die Umsetzbarkeit der Ergebnisse mit dem realen Service-Roboter aus. Die Simulation von Flüssigkeiten ist nicht möglich.

Prinzipiell muss ein Polygonmodell für ein Objekt definiert sein, um es in die Simulation einbinden zu können. Eigenschaften, die über das Polygonmodell hinaus für eine korrekte Berechnung von Griffen festgelegt sein müssen, sind:

- Gewicht
- Maximal zulässige Kraft
- Oberfläche

Angaben zur Oberflächenbeschaffenheit sind erforderlich, um den entsprechenden Reibungskoeffizienten zwischen Objekt und Greifer zu bestimmen (für Details siehe Kapitel 3.1.2). Handelt es sich bei dem Objekt um ein Gefäß, so muss zusätzlich das Nutzvolumen angegeben werden. Es wird von der Annahme ausgegangen, dass sich die Öffnung des Gefäßes entlang der Z-Achse des Objekts, am äußersten Punkt befindet. Liegt die Öffnung an einer anderen Stelle, so kann sie für das Modell über eine Benutzereingabe definiert werden. Das Objektkoordinatensystem, das als Referenz für einen Griff dient, ist wie folgt definiert: Die Z-Achse liegt entlang der vertikalen Achse des Objekts, dementsprechend bei einer Flasche oder einer Tasse entlang der Achse des Zylinders, der die geometrische Form beschreibt. Ist das Objekt rotationsinvariant, so ist die Orientierung der Y-Achse beliebig. Ist das Objekt dies nicht, wie z.B. eine Tasse, so definiert der Henkel der Tasse die Richtung der Y-Achse. Die X-Achse wird entsprechend einem rechtshändigen Koordinatensystem angeordnet. Somit ist die Position der Henkel bei Tassen implizit immer bekannt. Die Einrichtung des Objekt-Koordinatensystems ist in Abbildung 6.2 illustriert.

Optional kann ein Name für das Objekt angegeben werden, um es hinterher leichter von anderen Objekten unterscheiden zu können. Auch Objekt-Ports, also Volumina, die bei der Griffberechnung besonders beachtet werden sollen, können in Form von geometrischen Primitiven definiert werden (siehe auch Kapitel 4.2.6). Die Angabe einer Position und Orientierung des Objekts sind obligatorisch. Die Position ist allerdings kein Teil des Objektmodells, sondern wird für die Berechnung der Griffe benötigt.

Die Objekte, die im Folgenden für die Experimente verwendet werden, sind in Abbildung 6.3 dargestellt. Es sind (von links nach rechts): eine Tasse, ein Becher, ein Schokoriegel, ein Apfel, ein Akkuschrauber, ein Buch, ein Ball, eine 0,5 l Flasche sowie ein Sektglas und eine Banane.

Die Objekte repräsentieren eine Vielzahl von Alltagsgegenständen. So weisen z.B. der Ball, der Schokoriegel und das Buch eine einfache Geometrie auf. Der Akkuschrauber und das Sektglas haben komplexere Formen. Sowohl künstliche Formen wie die Flasche oder der Becher und die Tasse, als auch natürliche Formen, wie z.B. die Banane und der Apfel sind unter den Objekten. Die Auswahl der Objekte erfolgte

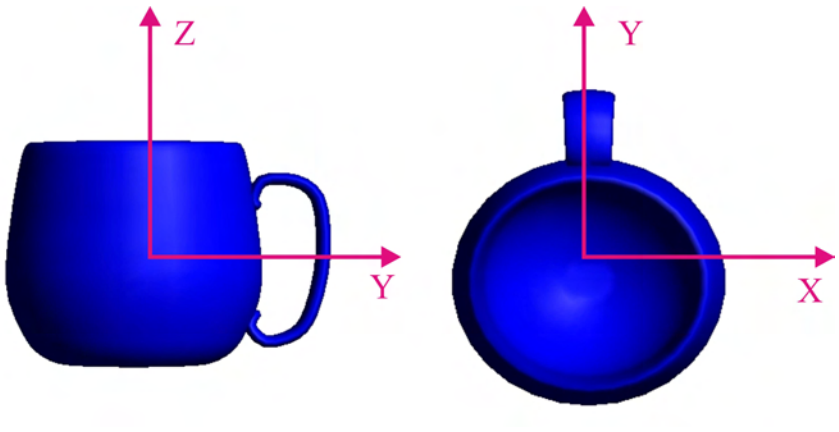


Abbildung 6.2: Koordinatensystem der Objekte am Beispiel einer Tasse: die Z-Achse entlang der Objekt-Längsachse, die Y-Achse entlang des Griffs, die X-Achse entsprechend eines rechtshändigen Koordinatensystems.

Abbildung 6.3: Für die Berechnung von Griffen verwendete Objekte.

auch in Bezug auf die maximal zulässigen Griffkräfte der Objekte. Bei einem Griff des Sektglases und des Schokoriegels dürfen nur relative geringe Kräfte aufgebracht werden. Dagegen können die Flasche, der Becher und der Akkuschrauber mit einer großen Kraft gegriffen werden.

6.2.2.2 Definition von Greifern

Um die Simulation von verschiedenen Greifern zu ermöglichen, müssen zwei Eigenschaften für diesen definiert werden. Zum einen ein geometrisches Modell, das die Form des Greifers beschreibt, und zum anderen die Kinematik des Greifers. Zur Zeit ist nur das Modell und die Kinematik der BarrettHand in

das System eingebunden. Es ist aber möglich, weitere Greifer zu integrieren. Das Koordinatensystem der Greifer muss dabei wie in Abbildung 6.4 definiert sein.

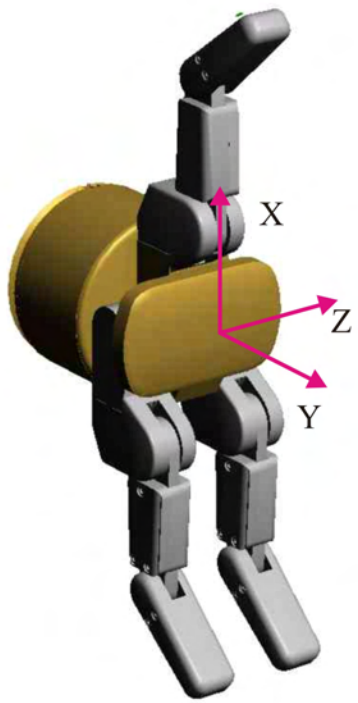


Abbildung 6.4: Koordinatensystem eines Greifers in der Simulation. Die Y-Achse liegt orthogonal zur Handfläche, die X-Achse entlang der Achse des Daumens, und die Z-Achse entsprechend eines rechtshändigen Koordinatensystems.

6.2.2.3 Definition von Griffen

Griffe sind in der Simulation entsprechend den in Kapitel 3 erörterten Grundlagen aufgebaut. Die Griffe sind immer in Bezug zu einem bestimmten Objekt definiert. Durch die folgenden Eigenschaften ist ein Griff beschrieben:

- Position der Hand relativ zum Objekt,
- Winkel der Gelenke der Hand,
- Kraft, die für den Griff aufgebracht werden muss.

Die Position der Hand ist durch eine homogene Transformation relativ zum Objekt definiert. Durch eine Berechnung der kinematischen Kette ist es möglich, den Griff jederzeit wieder umzusetzen, egal in welcher Position, relativ zum Welt-Koordinatensystem, sich das Objekt befindet. Eine entsprechende Transformationskette ist in Abbildung 6.5 dargestellt. Diese ist definiert durch:

$$^{W}T_{O} = {}^{W}T_{A}\,{}^{A}T_{H}\,{}^{H}T_{O}, \tag{6.1}$$

nach Umformung ergibt sich die Position von Objekt zu Hand zu:

$$ {}^{H}T_O = ({}^{W}T_A \, {}^{A}T_H)^{-1} \, {}^{W}T_O \tag{6.2} $$

wobei ${}^{W}T_O$ für die Transformation von Weltkoordinaten zu Objektkoordinaten und ${}^{H}T_O$ für die Transformation von Handkoordinaten zu Objektkoordinaten steht. Die weiteren Transformationen beschreiben die Transformation von Weltkoordinaten zur Basis des Roboterarms (${}^{W}T_A$) und die Transformation von der Basis des Roboterarms zur Basis der Roboterhand (${}^{A}T_H$) .

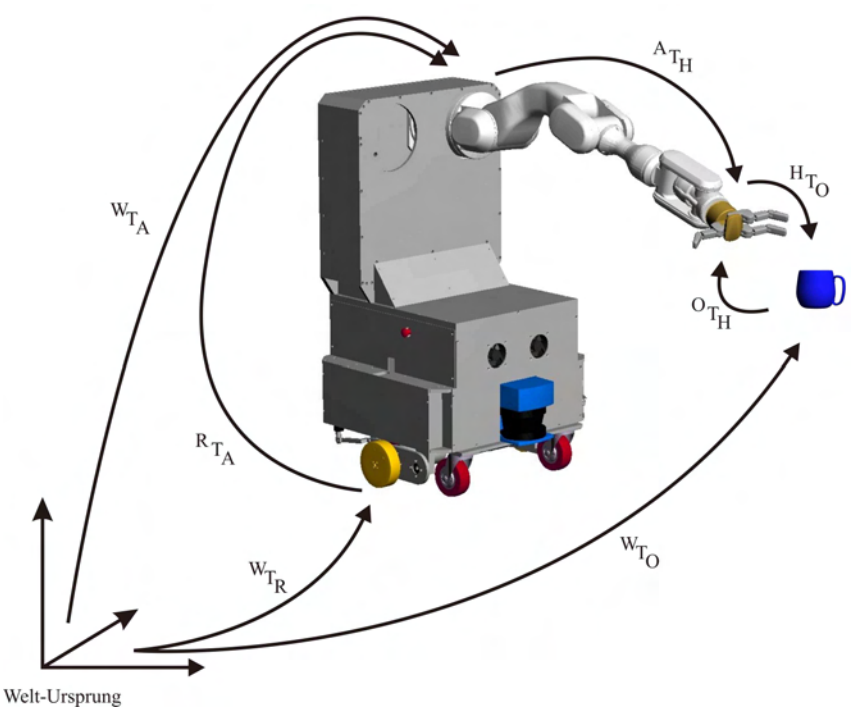


Abbildung 6.5: Die kinematische Kette zur Berechnung der Griffposition. W steht für Welt, O für Objekt, R, für Robotersystem, A für Arm und H für Hand. ${}^{W}T_O$ ist also die Transformation vom Welt- ins Objektkoordinatensystem, ${}^{W}T_A$ ist die Transformation vom Weltkoordinatensystem zur Basis des Arms, ${}^{A}T_H$ beschreibt die Transformation von der Basis des Arms zur Hand, etc.

6.2.2.4 Berechnung der Kontaktpunkte

Um die Kontaktpunkte zwischen Hand und Objekt bestimmen zu können, wird eine Kollisionserkennung berechnet. Diese erfolgt mittels *Oriented Bounding Box Trees* (*OBBT*), welche die geometrische Struktur des Objekts widerspiegeln und mittels eines hierarchischen Aufbaus eine effiziente Berechnung erlauben. Für Details der Kollisionserkennung mittels OBBT siehe auch [Gottschalk et al. 1996], [Eberly 2002, Eberly 2004] und [Eberly 2001]. Liegt für ein Objekt aus Dreiecken bestehendes ein Polygonmodell vor, so ist es möglich, die Objektstruktur in einen solchen OBBT zu überführen. Dies ermöglicht eine Kollisionsberechnung für beliebige Objekte. In der, im Rahmen dieser Arbeit entwickelten Anwendung können die Polygonstrukturen für einfache geometrische Primitive (Quader, Zylinder,

Kugel) durch die Definition der entsprechenden Parameter generiert werden. Für komplexe Objekte besteht die Möglichkeit, ein solches Modell über eine Datei zu laden. Das Dateiformat ist hierbei an den VRML-Standard angelehnt[2]. In diesem Dateiformat findet sich bereits eine Vielzahl von unterschiedlichen Objekten im Internet, so dass nahezu für jedes Objekt ein entsprechendes Modell gefunden werden kann.

Auch die Finger und die Handfläche der BarrettHand werden als Polygonmodell in die Berechnung eingebracht. Die Kollisionserkennung ist ein rechenaufwendiger Prozess, mit der maximalen Komplexität $O(n^2)$ mit n als Anzahl der Polygone. Je komplexer ein Modell ist, um so langsamer ist die Berechnung der Kollisionserkennung, und um so langsamer die eines Griffs. Da die Simulation möglichst genaue Resultate liefern soll, wurde im vorliegenden Fall ein sehr detailliertes Modell mit über 4000 Polygonen pro Finger benutzt, das bereits zu einem hohen Berechnungsaufwand führt. Je feiner und detaillierter bzw. je komplexer die Modelle werden, um so aufwendiger wird die Berechnung der Kollisionen zwischen Hand und Objekt.

Für die Implementierung der Kollisionserkennung wurde die WildMagic-Bibliothek [Geometrictools 2007] benutzt. Sie zeichnet sich durch eine sehr gute Dokumentation und überdurchschnittliche Funktionalität aus, zudem wird sie kontinuierlich weiterentwickelt und gepflegt.

6.3 Generierung von Griffen durch Definition

In den folgenden Abschnitten sollen vier Verfahren für die Berechnung von Griffen vorgestellt werden. Die Verfahren sind: Definition von Griffen durch Benutzereingabe, Generierung von Griffen durch Demonstration, Berechnung von Griffen über ein Suchverfahren und die Generierung von Griffen mittels eines Lernverfahrens. Der Schwerpunkt liegt dabei auf der Berechnung von Griffen mittels eines Lernverfahrens. Hierzu wird eine Erweiterung für das TD(λ)-Verfahren entwickelt, das in Abschnitt 5.2.4 vorgestellt wurde.

Die Generierung von Griffen mittels Definition stellt die einfachste Art dar, Griffe zu erzeugen. Hierbei kann der Benutzer einen Griff definieren, der von dem System evaluiert wird. Die Definition für Griffe kann auf unterschiedliche Weisen erfolgen. Zum einen kann der Benutzer über ein Kommando-Interface die entsprechenden Parameter für die Translation und Orientierung der Roboterhand sowie der Gelenkwinkel der Finger eingegeben. Diese Methode hat jedoch gravierende Nachteile. Sie ist umständlich und nicht anschaulich genug. So müssen für die Definition sechs Parameter für die Position der Hand und vier Parameter für die Definition der Gelenkwinkel eingegeben werden. Dies kann durch den Einsatz eines grafischen Benutzerinterface vereinfacht werden. Mit diesem Interface kann der Benutzer die Position der Hand zum Objekt in einer Simulation manuell festlegen. Die Gelenkwinkel der Finger werden von dem System automatisch bestimmt. Lediglich der Spreizwinkel muss von dem Benutzer vorgegeben werden. Die definierten Griffe können mittels der Simulation direkt evaluiert werden. Hierdurch kann ausgeschlossen werden, dass der Benutzer instabile Griffe oder Griffe, die mehr Krafteinsatz erfordern, als für das Objekt zulässig ist, ausgeführt. Insgesamt ist es eine eher umständliche Methode, Griffe für eine Roboterhand zu generieren. Weiterhin ist davon auszugehen, dass nicht alle Benutzer über die not-

[2] VRML steht für „Virtual Reality Markup Language“ und ist entwickelt worden um dreidimensionale, interaktive Vektor-Graphiken darstellen zu können. Sie wurde auch speziell im Hinblick mit Verwendung im Internet entwickelt. Weitere Details sind zu finden in [Cary and Bell 1997] und [Web3d 2007].

wendige Erfahrung im Umgang mit 3D-Visualisierungen verfügen, so dass die Bedienung für einige Benutzer schwierig ist.

6.4 Generierung durch Demonstration

Die Generierung von Griffen durch Demonstration beruht auf dem Prinzip des *Vormachen und Nachahmen* oder auch *Learning by Demonstration* (*LBD*). In Kooperation mit Markus Hüser [Hüser et al. 2006] wurde ein System entwickelt, das in der Lage ist über eine multimodale Schnittstelle mit dem Benutzer zu interagieren und dessen Bewegungen zu verfolgen und zu imitieren, um diesen Ansatz auf dem Service-Roboter TASER umzusetzen. Der von dem Benutzer ausgeführte Griff wird von dem System analysiert und bei Instabilität entsprechend optimiert. Da das Verfolgen der menschlichen Hand in einer natürlichen Umgebung ein sehr komplexes Thema ist, wird es hier nur in gebotenen Kürze vorgestellt. Weitere Details sowie eine genaue Analyse der Verfahren sind in [Hüser 2008] zu finden.

(a) (b)

Abbildung 6.6: Szenario einer Demonstration: (a) der Roboter wird von einem Benutzer instruiert, (b) der Roboter führt den demonstrierten Griff aus.

6.4.1 Learning by Demonstration Szenario

Ein typischer Aufbau für das LBD-Szenario ist in Abbildung 6.6 dargestellt. Der Benutzer steht dem Roboter gegenüber, der die Handbewegung mittels des Stereokamerasystems verfolgt und aufzeichnet. Der Beginn der Bewegung zum Objekt und das Ende der Bewegung wird dem Roboter über eine Schnittstelle zur Spracheingabe angezeigt. Durch Verwendung eines kalibrierten Stereo-Kamerasystems und eine Korrespondenzanalyse der Stereobilder kann eine Tiefeninformation aus den aufgezeichneten Bildern errechnet werden, so dass die räumliche Position der Hand berechnet werden kann. Die Qualität der Tiefeninformation hängt dabei von verschiedenen Faktoren ab. Zum einen von der Qualität der Kameras, der Qualität der Kalibrierung und von der Entfernung des Instrukteurs zur Kamera. Je größer die Distanz zwischen beiden ist, umso schlechter ist die Tiefenauflösung. Um diese Messfehler des Verfahrens zu minimieren und eine möglichst glatte Trajektorie der Bewegung berechnen zu können ist eine mehrfache

Wiederholung der Bewegung durch den Instrukteur erforderlich. Anschließend ist der Roboter in der Lage, die generalisierte Trajektorie eigenständig nachzufahren und den entsprechenden Griff auszuführen.

6.4.2 Generierung der Trajektorie

Die Positionserkennung und Verfolgung der Hand des Instrukteurs in den Bildsequenzen erfolgt in zwei miteinander gekoppelten Stufen. In der ersten Stufe berechnet ein *Expectation-Maximization-Like-Algorithmus* [Zivkovic and Kröse 2004] aus der aktuellen Position und Bewegungsgeschwindigkeit der Hand Vorhersagen über den Aufenthaltsort der Hand im nächsten Zeitschritt bzw. Bild. Diese Schätzung wird als Grundlage für ein Suchfenster gewählt. In diesem dann, in der zweiten Stufe, basierend auf einem kombinierten Textur und Farbmodell der menschlichen Haut, eine Segmentierung durchgeführt wird, die eine genauere Verfolgung von hautfarbenen Objekten ermöglicht.

6.4.3 Generierung von Griffen

Um einen Griff generieren zu können, muss das System in der Lage sein, die Postur der Hand, also die Position und Orientierung der Hand und die Stellung der Finger, des Benutzers während des Greifvorgangs zu analysieren und reproduzieren. Dies erfordert einen enormen Aufwand, da die menschliche Hand über 27 Freiheitsgrade verfügt [Rehg and Kanade 1994]. Zudem sind meist Teile der Hand verdeckt, so dass eine vollständige Rekonstruktion der Postur aufgrund fehlender visueller Informationen in der Regel nicht möglich ist. Dies ist in diesem Fall aber auch nicht erforderlich. Da nur eine Übertragung des Griffs von der menschlichen auf die Roboterhand erfolgen muss, ist es ausreichend, diese Transferfunktion zu definieren.

Die Bestimmung der Position und Orientierung der Hand erfolgt mittels Daten der Handverfolgung aus den Kamerabildern. Die Übertragung der Handkonfiguration auf die BarrettHand erfolgt mittels einer *Hauptkomponentenanalyse* (*Prinicipal Component Analysis, PCA*). Details zur PCA sind in Anhang C zu finden. Als Eingabe für die PCA dient ein segmentiertes Bild der Hand des Instrukteurs. Die Segmentierung trennt das Bild in Vorder- und Hintergrund und ermöglicht somit eine genaue Klassifikation der Handkonfiguration. Beispiele für den Transfer der Konfiguration einer menschlichen Hand auf die BarrettHand des TASER sind in Abbildung 6.7 dargestellt.

Die Transferfunktion ist in diesem Fall über drei unterschiedliche Grifftypen definiert worden. Die Grifftypen (Abb. 6.8) entsprechen den in [Miller et al. 2003] definierten Grifftypen. Eine abschließende Analyse evaluiert den Griff. Sollte er aufgrund kinematischer oder sonstiger Probleme nicht ausführbar sein, wird die Position der Hand leicht variiert, so dass eine Ausführung letztendlich möglich ist.

6.4.4 Experimente

Verschiedene Objekte wurden in den LBD-Experimenten gegriffen. Von diesen soll der Griff für ein Buch exemplarisch vorgestellt werden (für weitere Experimente siehe [Hüser et al. 2006]). Unter Beobachtung des Roboters wurde das Buch zehn mal von einem Instrukteur gegriffen. Aus den Sequenzen wurden über 1600 Weltkoordinaten extrahiert und als Eingabedaten für die Berechnung der Trajektorie genutzt.

Die Trajektorie für das Buch ist in Abbildung 6.9 dargestellt. Zusätzlich ist der Pfad auf die XY-Ebene projiziert worden. Wie in der Abbildung deutlich wird ist die Trajektorie in Bezug auf die Länge der

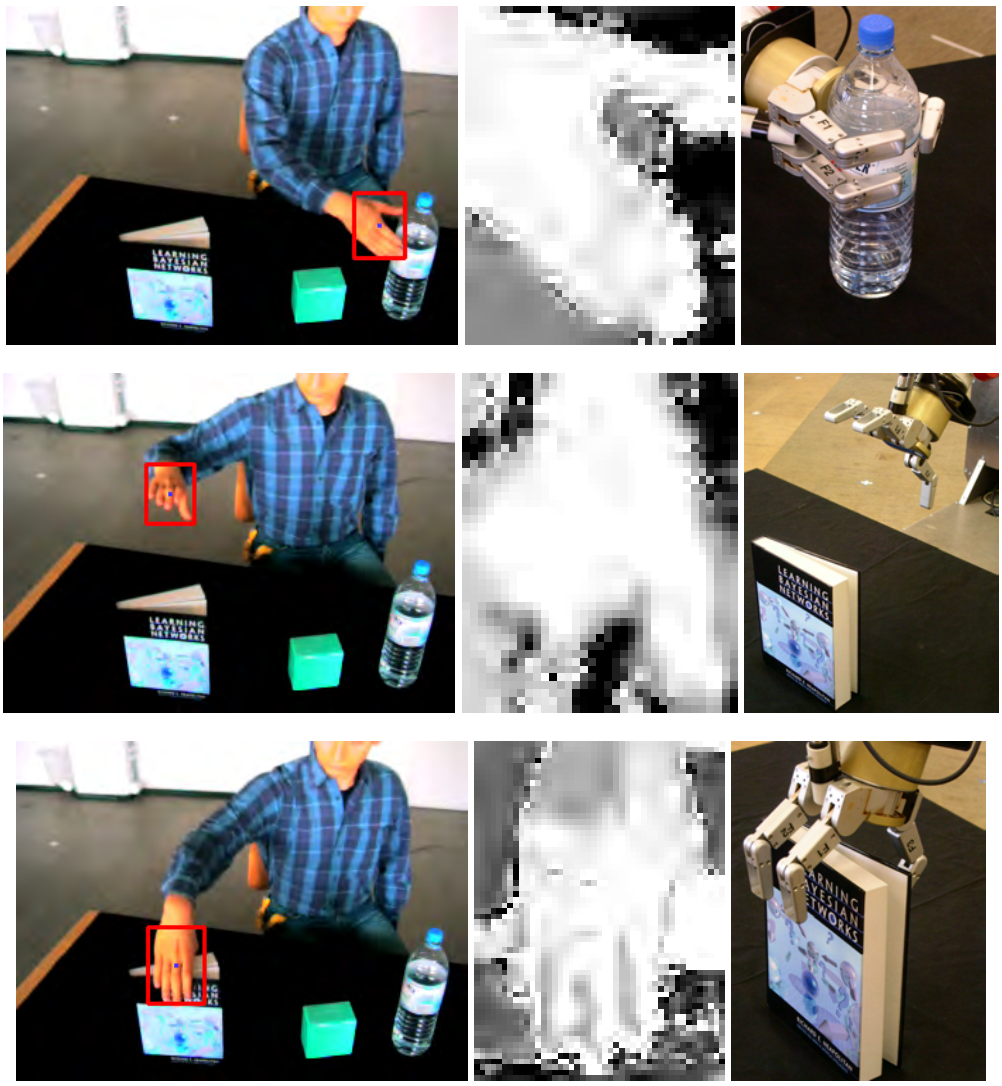


Abbildung 6.7: Übertragung der Handpostur während des Greifens auf eine BarrettHand: links original Bild, mitte segmentierte Hand, rechts Ergebnis mit der BarrettHand.

Bewegung und auf die Anzahl der besuchten Punkte optimal. Der berechnete Griff wurde anschließend mit dem Service-Roboter TASER ausgeführt und erwies sich dabei als stabil.

6.5 Generierung von Griffen durch Breitensuche

Bei der Generierung von Griffen mittels der *Breitensuche* [Russel und Norvig 2003] werden an allen möglichen Positionen Griffe berechnet. Dabei muss eine Diskretisierung der betrachteten Positionen vorgenommen werden. Die Art der Diskretisierung hat allerdings einen nicht zu vernachlässigenden Einfluss auf den erforderlichen Rechenaufwand, denn je feiner das Positionsraster gewählt wird, um

(a) Spread = 0.0° (b) Spread = 45.0° (c) Spread = 180.0°

Abbildung 6.8: Grifftypen der Transferfunktion.

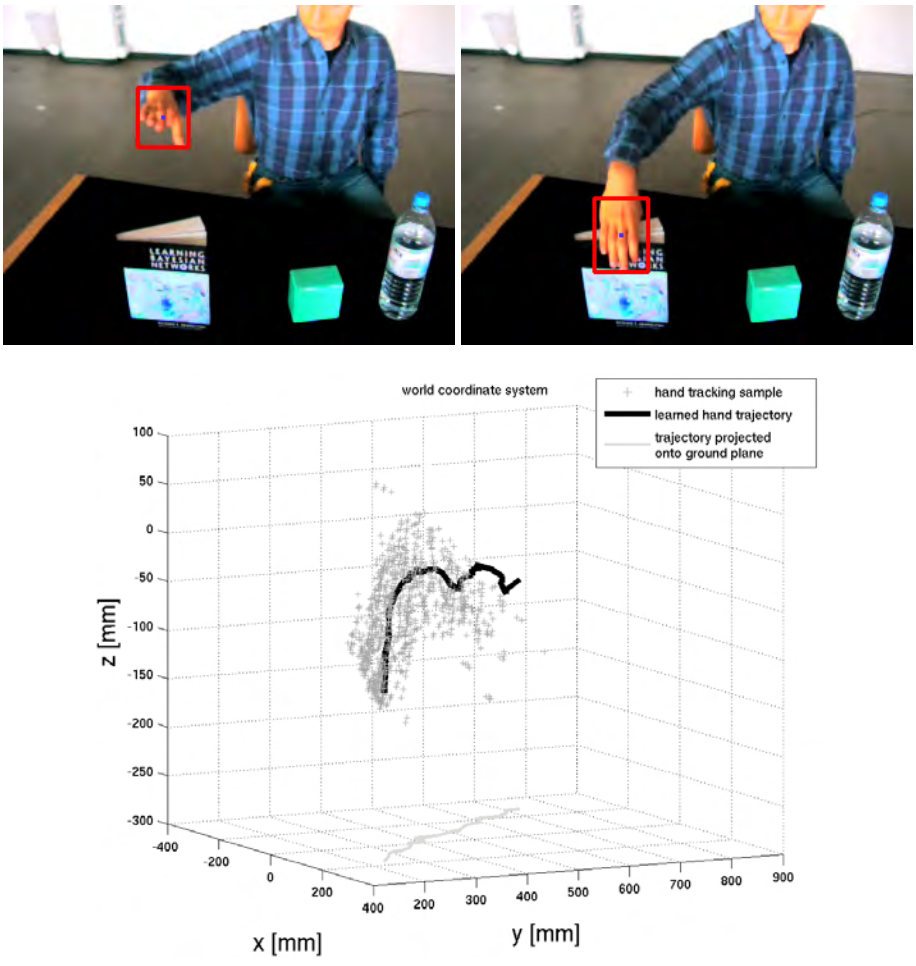


Abbildung 6.9: Ergebniss der Trajektoriengenerierung für ein Buch.

so mehr Griffe müssen an eben diesen Positionen berechnet und evaluiert werden. Allerdings ist das System durch ein feines Raster in der Lage, möglichst viele Griffe zu bestimmen. Bei einem zu groben

Raster werden möglicherweise einige qualitativ gute Griffe nicht evaluiert. Bei kleinen Objekten kann ein grobes Raster auch dazu führen, dass kein stabiler Griff gefunden werden kann. Da bei dem Greifen mittels eines Service-Roboters in einer realen Umgebung viele Griffe aufgrund kinematischer Probleme nicht ausführbar sind, ist es erforderlich, ein relativ feines Raster für die Positionsänderung zu wählen.

Als Beispiel für die Wahl der Diskretisierung wurden in dieser Arbeit ein Positionsraster von 5 mm mit einem Raster von 10 mm verglichen. Es wurden mit beiden Rastern für verschiedene Objekte, Griffe von der Seite der Objekte berechnet, und die Anzahl der gefundenen Griffe miteinander verglichen. In Abbildung 6.10 sind zwei Beispiele für einen solchen Griff von der Seite dargestellt. Bei einem Raster von 5 mm werden zwar deutlich mehr Griffe gefunden, dies rechtfertigt aber den erheblich gesteigerten Aufwand nicht, wie es in Tabelle 6.1 zu sehen ist. Beispielsweise werden bei einem Quader der Größe 40 mm $\times$ 50 mm $\times$ 120 mm, bei einem Raster von 5 mm im Vergleich zum 10 mm Raster 13939 mehr Positionen berechnet, aber das Verhältnis von stabilen Griffen pro Position bleibt ungefähr gleich (0.02 zu 0.019)[3]. Bei der Box werden bei einem Positionsraster von 5 mm ungefähr sechs mal so viele Griffe wie bei einem Positionsraster von 10 mm gefunden, aber da aber 51 Griffe mehr als vier Griffen pro Zentimeter entsprechen (bezogen auf die Längsachse des Objekts), ist dies bereits mehr als ausreichend (siehe Tabelle 6.1 1. Zeile).

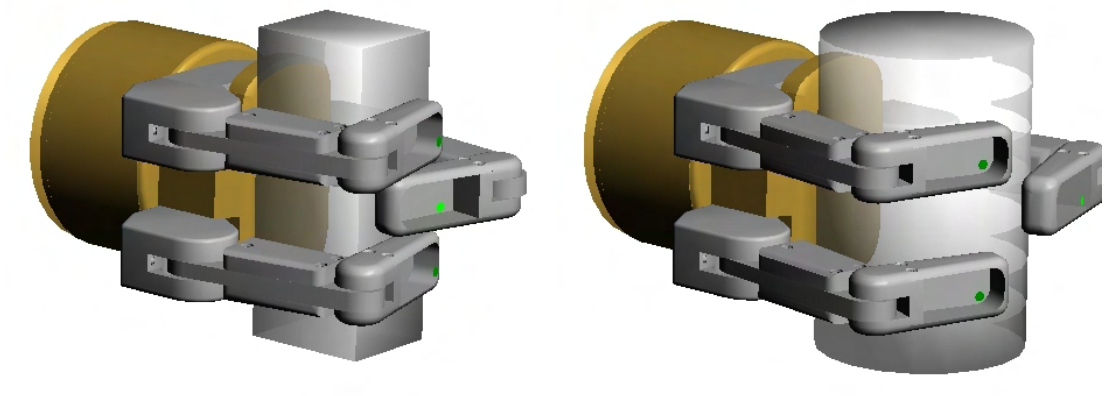


Abbildung 6.10: Griff von der Seite bei einer Box und einem Zylinder.

Der Vorteil der Breitensuche liegt darin, dass innerhalb des gewählten Rasters alle möglichen Griffe gefunden werden. Der große Nachteil besteht darin, dass an vielen Positionen Griffe berechnet werden, die nicht stabil sind. Eine intelligente Auswahl von Positionen, an denen stabile Griffe generiert werden können, ist also wünschenswert. Hierdurch könnte der Rechenaufwand deutlich reduziert werden. Ein solches Verfahren wird im nächsten Abschnitt beschrieben. Hierbei werden die Griffe mittels maschinellem Lernen, (selbstbewertendem Lernen) berechnet. Die Ergebnisse der Generierung von Griffen mittels Breitensuche soll an dieser Stelle nicht näher erörtert werden, sie dienen lediglich als Referenz

[3] Die Griffe werden nicht nur direkt über dem Quader in einem Bereich von 40 mm $\times$ 50 mm $\times$ 120 mm berechnet, sondern es wird auch nach Griffen im Randbereich des Objekts gesucht. Hierdurch ergibt sich eine Suchraumgröße von rund 80 mm $\times$ 100 mm $\times$ 240 mm. Zusätzlich kommt noch hinzu, dass die Position bezüglich des Handmittelpunkts und nicht zu der Handfläche berechnet wird, somit ergeben sich 16.468 mögliche Position für einen Griff bei einem Raster von 5 mm.

	Diskretisierung					
	5 mm			10 mm		
Objekt	Positionen	Griffe	Griffe/Positionen	Positionen	Griffe	Griffe/Positionen
Box 40 x 50 x 120 mm	16.468	311	0,019	2.529	51	0,020
Zylinder 20 x 150 mm	29.227	399	0,014	3.873	40	0,010
Zylinder 40 x 150 mm	24.935	1.076	0,043	3.329	184	0,055
Kugel 60 mm	12.401	373	0,03	1.942	40	0,021
Kugel 40 mm	5.991	174	0,029	984	31	0,032

Tabelle 6.1: Vergleich der Positionsdiskretisierung von 5 mm und 10 mm bei Griffen von der Seite. Die Spalte „Positionen“ stellt die Anzahl der Greifversuche dar, „Griffe“ die Anzahl der gefunden stabilen Griffe, die weniger als 20 N Kraft je Kontakt erfordern, und „Griffe/Positionen“ ist der Quotient aus beiden.

für die Generierung von Griffen mittels maschinellem Lernen und werden in diesem Zusammenhang in Abschnitt 6.8 analysiert.

6.6 Generierung von Griffen durch Lernen

Die automatische Berechnung von stabilen Griffen für jegliche Art von Objekten ist eines der Ziele in der modernen Servicerobotik. Hierdurch wird ein Roboter in die Lage versetzt, eigenständig Griffe für ihm bisher unbekannte Objekte zu erzeugen, zu analysieren und letztendlich auch anzuwenden. Dies verleiht dem Roboter mehr Autonomie, da Griffe nicht erst von einem Operator eingegeben oder demonstriert werden müssen. Die automatische Generierung setzt allerdings Wissen des Systems über das Objekt voraus. So muss das System in der Lage sein, die Form des Objekts zu bestimmen. Aus diesen Daten müssen Parameter wie z.B. das Gewicht berechnet werden. Auch die Oberflächenbeschaffenheit spielt in Bezug auf die Berechnung von Griffen eine enorme Rolle, wie in Kapitel 3.1.2 gezeigt wurde. Hinzu kommt noch, das auch die Kräfte bekannt sein müssen, die maximal an einem Punkt auf das Objekt einwirken dürfen, damit das Objekt nicht zerstört oder beschädigt wird. Für die im Folgenden betrachteten Verfahren werden die erforderlichen Eigenschaften als gegeben betrachtet. Zwar gibt es Verfahren, welche die dreidimensionale Form von Objekten anhand von Kamerabildern berechnen können, wie z.B. in [Pollefeys et al. 2004] vorgestellt, diese Verfahren sind jedoch sehr komplex und werden daher im Rahmen dieser Arbeit nicht eingesetzt.

Das Finden von stabilen Griffen für ein Objekt, für das bisher keine Griffe generiert worden sind, ist Gegenstand aktueller Forschung. Die in Abschnitt 6.1 beschrieben Ansätze stellen in vielen Fällen eine nur unzureichende Lösung dar. Für die Definition eines Griffs, wie in Abschnitt 6.3 beschrieben, muss der Benutzer in der Lage sein, die Stabilität eines Griffs richtig einzuschätzen. Diese Einschätzung kann unter Umständen fehlerhaft sein und zu Problemen, z.B. zur Instabilität des Griffs, führen. Bei der Berechnung

von Griffen nach einer Demonstrationsphase liegt das Problem in der Demonstration als solche: Es kann hierbei nicht garantiert werden, dass die Demonstration in ihrer Gesamtheit vollständig erfasst werden kann.

Sowohl die Generierung von Griffen durch Definition als auch die Generierung von Griffen durch Demonstration haben den Nachteil, dass immer nur ein Griff generiert wird. Für die Entwicklung von mehreren Griffen sind eine Vielzahl von Demonstrationen oder Definitionen erforderlich. Soll ein neues Objekt gegriffen werden, oder ist aufgrund technischer Probleme keiner der bekannten Griffe anwendbar, so muss der Benutzer einen neuen Griff definieren.

Hieraus folgt, dass die automatische Generierung von Griffen eine nahezu unerlässliche Fähigkeit eines Service-Roboters sein muss. Die automatische Berechnung über eine Breitensuche erscheint möglich, ist aber enorm aufwendig. Wie in Abschnitt 6.1 erläutert wird, existieren bereits einige solche Ansätze, die alle jedoch die oben beschriebenen Nachteile aufweisen.

Im Folgenden wird daher ein Verfahren für die Berechnung von Griffen mittels selbstbewertendem Lernen entwickelt. Hierzu ist zunächst die Übertragung des Problems auf einen Markovschen-Entscheidungs-Prozess (siehe Kapitel 5.1) erforderlich. Wird im Folgenden von Greifern und Fingern gesprochen, so sind damit die Greifer und Finger des Service-Roboters gemeint.

Übertragung auf einen Markov-Entscheidungs-Prozess

Wie in Kapitel 5.1 dargelegt, wird ein Markov-Entscheidungs-Prozess (MDP) durch das Tupel $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}^a_{s,s'}, \mathcal{R} \rangle$ beschrieben. Hierbei ist $\mathcal{S}$ die Menge aller möglichen Zustände, die das System einnehmen kann. $\mathcal{A}$ die Menge aller möglichen Aktionen. $\mathcal{P}^a_{s,s'}$ beschreibt die Zustands-Übergangswahrscheinlichkeiten. $\mathcal{R}$ beschreibt die Rewardfunktion. Im Folgenden werden die Definitionen der Komponenten des MDP in der Domäne des Greifenlernens gegeben. Zunächst müssen hierfür die Zustände und Aktionen für ein MDP definiert werden.

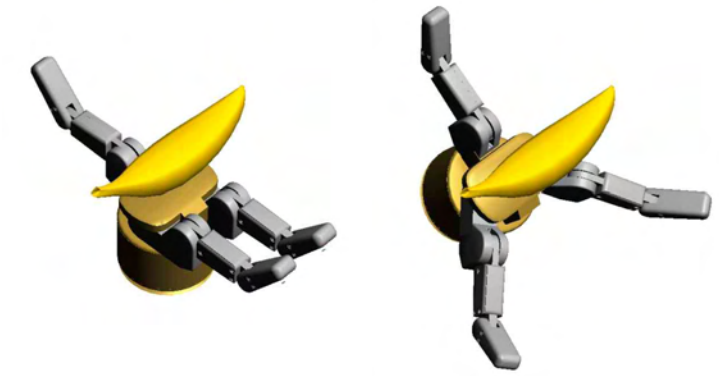
Zustand

Die Definition eines Zustandes für ein System, das in der Lage sein soll Greifen zu lernen, ergibt sich aus den Merkmalen, die einen solchen Zustand charakterisieren. Dies sind:

- Die Position der Hand relativ zum Objekt
- Der Spreizwinkel der Hand
- Die Information, ob ein Griff ausgeführt wurde oder nicht

Die Gelenkwinkel der Finger sind nicht explizit Teil eines Zustandes. Es wird davon ausgegangen, dass die Hand zu Beginn eines Greifversuchs geöffnet ist und sich durch das Zugreifen schließt. Wenn die Hand geschlossen wird und sich dabei ein stabiler Griff bildet, werden die Gelenkwinkel der Finger zusammen mit weiteren Parametern als Griff gespeichert. Somit sind in der Information ob, ein Griff ausgeführt wurde oder nicht, implizit die Gelenkwinkel enthalten.

Die Position der Hand relativ zum Objekt wird in Form einer homogenen Transformation angegeben. Hieraus lassen sich sowohl der translatorische Versatz zwischen Hand und Objekt, als auch die Rotation



(a) Zustand 1 (b) Zustand 2

(c) Zustand 3 (d) Zustand 4

Abbildung 6.11: Beispiele für Zustände, die bei der Berechnung von Griffen auftreten können. Die Parameter der Zustände können Tabelle 6.2 entnommen werden.

Zustand	1	2	3	4
Translation x, y, z in mm	0, -75, 0	-55, -75, 0	-8.5, 0, 0	-8.5, 0, 0
Rotation Roll-, Pitch-, Yaw-Winkel	0°, 0°, 0°	0°, 0°, 40°	0°, 0°, 0°	0°, 0°, 0°
Spreizwinkel	0°	40°	0°	180°
Griff	nein	nein	ja	ja

Tabelle 6.2: Parameter der in Abbildung 6.11 dargestellten Zustände. Die Einheiten sind Millimeter bzw. Grad.

relativ zum Objektkoordinatensystem ableiten. Für die Berechnung von Positionsänderungen bei Zustandsübergängen wird die homogene Transformation in die Komponenten Translation und *Roll-Pitch-Yaw*-Winkel oder auch *X-Y-Z fixe* Winkel [Craig 2005], zerlegt. In Abbildung 6.11 sind vier Beispiele für

unterschiedliche Zustände dargestellt. Die entsprechenden Parameter der Zustände sind in Tabelle 6.2 aufgeführt.

Aktionen

Die Aktionen, mittels denen Greifen gelernt werden soll, sind die Folgenden:

- Translation entlang der Y-Achse
- Translation entlang der X-Achse
- Translation entlang der Z-Achse
- Rotation um die X-Achse
- Rotation um die Y-Achse
- Rotation um die Z-Achse
- Änderung des Spreizwinkels
- Ausführung eines Griffs (Schließen der Finger)

Alle Aktionen sind in Bezug zum Koordinatensystem des Greifers definiert. Hierbei liegt die Y-Achse senkrecht zur Handfläche, stellt also den Annäherungsvektor dar. Die Translation entlang der Y-Achse wird eingeschränkt, da von der Annahme ausgegangen wird, dass sich die Hand zunächst außerhalb der Reichweite des Objekts befindet. Somit ist eine Bewegung, die die Hand noch weiter vom Objekt entfernt, nicht sinnvoll und führt nur zu einer unnötigen Vergrößerung des Zustandsraumes.

Die Aktionen werden in Abbildung 6.12 anhand eines Beispiels dargestellt. Die Pfeile stehen dabei für Bewegungsaktionen, die Kugeln für Griffe. Die Länge der Pfeile bzw. der Durchmesser der Kugel stellen dabei den Wert der entsprechenden Aktion dar. Positiv bewertete Aktionen sind grün, negativ bewertet Aktionen sind rot eingefärbt. Rotationsaktionen und Aktionen, die den Spreizwinkel der Hand ändern, sind aus Gründen der Übersichtlichkeit von der Visualisierung ausgenommen. In der Abbildung sind vier Zustände dargestellt: die beiden Kugeln stehen für je einen Zustand, in dem ein Griff ausgeführt wurde. Die beiden anderen Zustände liegen jeweils unterhalb der Kugel, im Zentrum der „Pfeile“. Sie stellen die Bewertung der Bewegungsaktionen dar.

Die Übergangswahrscheinlichkeiten durch eine Aktionsausführung, also $\mathcal{P}^{a}_{s,s'}$, werden alle zu 1 definiert. Diese Möglichkeit besteht, da bei einer realen Anwendung des Verfahrens die Hand an einem Roboterarm montiert ist und dieser Bewegungsbefehle immer umsetzt. Gleiches gilt für die Änderung des Spreizwinkels oder das Schließen der Finger der Roboterhand.

Aktionsauswahl

Die Aktionsauswahl erfolgt nach dem Epsilon-Greedy Verfahren (siehe Kapitel 5.1.4). In der Literatur ist umstritten, welche Strategie die bessere Explorationsstrategie ist [Sutton and Barto 1998]. Es lässt sich zeigen, dass die Explorationstemperatur T der Boltzman-Exploration mit dem ϵ des Epsilon-Greedy Verfahren vergleichbar ist. Zudem hat sich in verschiedenen durchgeführten Experimenten herausgestellt, dass die Epsilon-Greedy Aktionsauswahl in diesem Fall im Durchschnitt die besseren Ergebnisse liefert.

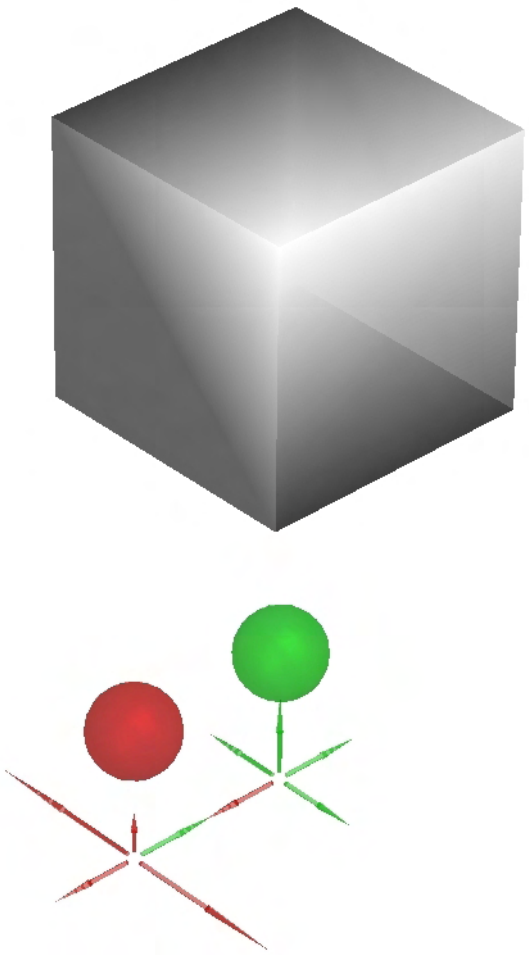


Abbildung 6.12: Visualisierung der Aktionen. Positiv bewertete Aktionen werden grün dargestellt, negativ bewertete rot. Die Pfeile markieren Bewegungsaktionen, die Kugeln markieren Griffe. Je länger ein Pfeil bzw. je größer eine Kugel, desto größer ist der entsprechende Wert.

Rewardfunktion

Die Rewardfunktion sollte so eingestellt sein, dass sie bei „guten“ Aktionen einen positiven Reward gibt und bei „schlechten“ Aktionen einen negativen. Was die „guten“ und „schlechten“ Aktionen sind, hängt von dem Kontext ab, in dem gelernt wird.

Beim Greifen von Objekten ist ein „guter“ Griff ein Griff, der möglichst stabil ist. Im Gegensatz dazu ist ein schlechter Griff, ein Griff der (a) entweder nicht stabil ist oder aber (b) die erforderlichen Kräfte, die für die Ausführung des Griffs aufgebracht werden müssen, außerhalb der Toleranzen des zu greifenden Objekts liegen, oder (c) ein Griff, bei dem höchstens ein oder gar kein Kontakt zu dem Objekt besteht. Der erste Fall ist der wahrscheinlichste. Aber auch die zweite Variante ist denkbar, da der Startpunkt für das Lernen so gewählt wurde, dass sich das Objekt zunächst nicht in Reichweite der Hand befindet, da

keine vordefinierten Grenzen entlang der Objektachsen definiert sind, und so Greifversuche außerhalb der Reichweite erfolgen können.

Somit orientiert sich die Bewertungsfunktion an der Stabilität eines Griffs. Diese wird mit dem Kriterium nach [Ferrari and Canny 1992], das in Kapitel 3.2.3 vorgestellt worden ist, berechnet. Da das Verfahren unabhängig von der Objektgröße ist, lassen sich so auch Griffe von verschiedenen Objekten miteinander vergleichen. Ein instabiler Griff wird mit -1 bewertet, falls es mehr als einen Kontaktpunkt zwischen Hand und Objekt gibt und mit -100 falls es einen oder keinen Kontaktpunkt gibt. Die Bewertung -100 scheint auf den ersten Blick sehr groß gewählt, hat sich aber in den Experimenten bewährt: Griffe die einen oder keinen Kontaktpunkt mit dem Objekt haben, liegen sehr weit am Rand, und markieren somit auch die Grenzen des Zustandsraumes, in dem es sinnvoll ist, nach stabilen Griffen zu suchen. Da diese Grenzen a-priori nicht definiert werden, müssen sie über andere Methoden impliziert werden. Wird eine Aktion, die in so einen Grenzzustand führt, stark negativ bewertet, sinkt die Wahrscheinlichkeit, dass sie noch einmal ausgewählt wird, drastisch.

Die Rewardfunktion $r(s, a)$ ist also definiert durch:

$$r(s,a) = \begin{cases} -100 & \text{falls } 0 \leq N_g \leq 1 \\ -1 & \text{falls } GWS(g) = 0 \\ GWS(g) & \text{sonst.} \end{cases} \tag{6.3}$$

Dabei ist N_g die Anzahl der Kontakte zwischen Hand und Objekt und $GWS(g)$ das Ergebnis der Berechnung des Grasp-Wrench-Space des Griffs g.

6.7 Adaption eines Lernverfahrens

Als Lernverfahren wird das in Abschnitt 5.2.4 beschriebene TD(λ) Verfahren eingesetzt. Da sich jedoch das Lernen mit dem realen Roboter TASER wegen Sicherheitsfaktoren als zu aufwendig und langwierig erwiesen hat, und zudem die Sensoren der BarrettHand nur unzureichende Informationen liefern (siehe Abschnitt 2.3), um daraus genaue Rückschlüsse über die Qualität eines Griffs zu ziehen, wurde eine Simulationsumgebung für die Berechnung von Griffen entwickelt (siehe Abschnitt 6.2.2). Da Verfahren aus dem Bereich des Temporal-Difference-Lernens angewendet werden, findet das Lernen in Episoden statt. Eine Episode terminiert, wenn ein Greifversuch durchgeführt wird. Das Ergebnis einer Episode ist nicht in jedem Fall ein stabiler Griff, da es vorkommen kann, dass es keine Kontaktpunkte zwischen Hand und Objekt gibt, oder die Kontaktpunkte keinen stabilen Griff formen. Da es bei der Berechnung der Griffe in der Regel viele Zustände (mehr als 100000) gibt, wird der Zustandsraum dynamisch aufgespannt. Nach dem in [Ferch 2001] beschriebenen Ansatz, werden nur die ausgeführten Aktionen mitsamt den zugehörigen Zuständen gespeichert. Dies ist deutlich speichereffizienter, als wenn der gesamte Zustands-Aktionsraum zu Beginn der Berechnungen aufgespannt wird. Aufgrund der Eigenschaften des Lernverfahrens ist die dynamische Bildung des Zustands-Aktionsraum zulässig und wirkt sich nicht nachteilig auf das Verfahren aus.

6.7.1 (Zu) Viele Terminalzustände

Bereits bei den ersten Experimenten zeigte sich ein Problem bei dem Erlernen von Griffen. Die Verfahren aus dem Bereich des Reinforcement-Lernens sind für Probleme mit wenigen Terminalzuständen ausgelegt. Bei der Berechnung von Griffen gibt es deutlich mehr Terminalzustände, da jeder Zustand in einen

Terminalzustand überführt werden kann. Das Ende einer Episode ist definiert durch die Ausführung eines Greifversuchs. Da dies in jedem Zustand möglich ist, kann jeder Zustand in einen Terminalzustand überführt werden. Das Problem bei den zahlreichen Terminalzuständen ist jedoch, dass nach dem Ende einer Episode kein Lernen mehr stattfindet. Somit ist es möglich, dass sich ein lokales Minimum bildet und der Agent nicht alle Lösungen findet. Um dies zu veranschaulichen wurden Reinforcement-Algorithmen nach dem TD(λ)-Verfahren trainiert, um über 1.500 Episoden Griffe für verschiedene Objekte zu generieren. Die Ergebnisse der Griffgenerierung sind in Abbildung 6.13 dargestellt, es sind jeweils die Mittelwerte von drei Lernvorgängen.

Wie in Abbildung 6.13 zu sehen ist, werden zwar ab Episode 1.000 noch Griffe für die Objekte gefunden, es sind aber bei weitem nicht mehr so viele wie zu Beginn des Lernens. Es ist zwar durchaus normal, dass das Lernen nach einer gewissen Zeit „langsamer“ wird, wie aber im Folgenden gezeigt wird, liegt es im vorliegen Fall daran, dass sich der Agent in einem lokalen Minimum befindet.

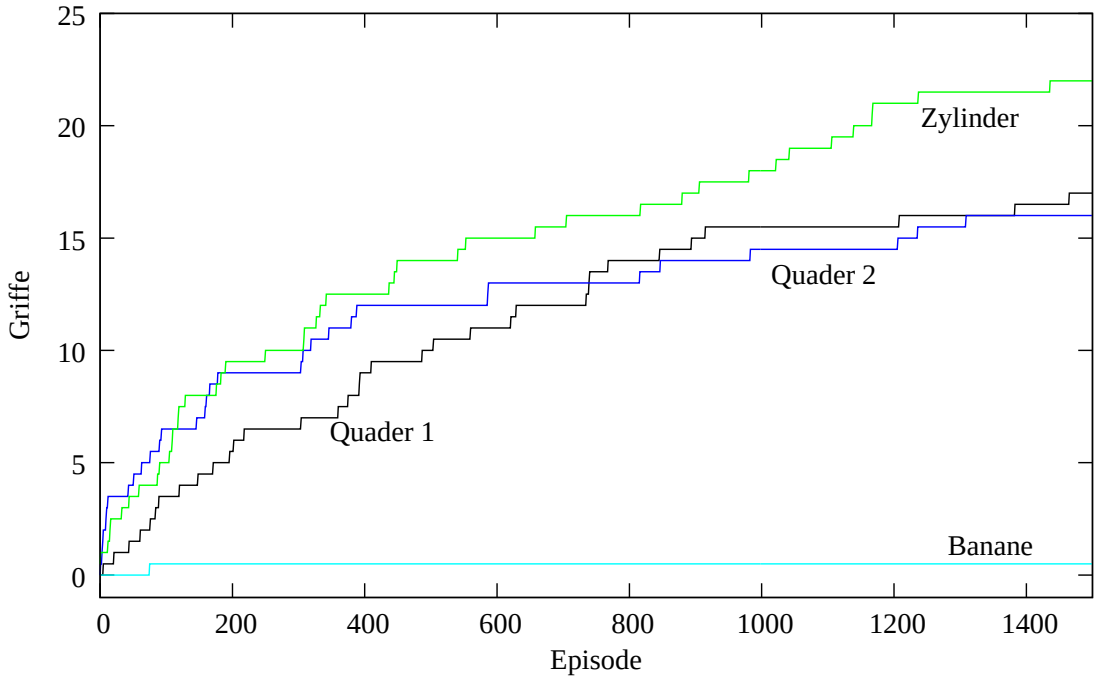


Abbildung 6.13: Lernen von Griffen mit einem Standard-TD(λ)-Lernverfahren. Die Anzahl der stabilen Griffe ist gegen die Anzahl der berechneten Episoden aufgetragen. Die Objekte sind: ein Quader der Größe 40 mm $\times$ 50 mm $\times$ 120 mm (Quader 1), ein Quader der Größe 30 mm $\times$ 180 mm $\times$ 245 mm (Quader 2), ein Zylinder der Größe 40 mm$\times$ 140 mm und eine Banane.

Selbst wenn man es schafft, die Parameter so einzustellen, dass ein solches lokales Minimum nach einer gewissen Zeit verlassen wird, ist der Lernvorgang sehr kostenintensiv. Ein weiteres Problem besteht darin, dass aufgrund einer teilweise zufälligen Aktionsauswahl auch die Zustände, die zu einem instabilen Griff führen, immer wieder ausgewählt werden. Dies führt zu einer zusätzlichen Verzögerung des Lernvorgangs. Erhöht man den Wert für eine zufällige Aktionsauswahl, findet zwar eine größere Exploration statt, aber das Lernen dauert deutlich länger, da der Einfluss der positiv bewerteten Griffe verloren geht. Da aber von der Annahme ausgegangen wird, dass es Regionen innerhalb des Suchraums gibt, in denen

besonders viele stabile Griffe zu finden sind, gelangt der Agent aufgrund der zufälligen Aktionsauswahl nur mit einer deutlich geringeren Wahrscheinlichkeit in diese Regionen, um sie weiter zu untersuchen.

Zusätzlich zu den Terminal-Zuständen, die zu einem instabilen Griff führen, gibt es eine Vielzahl von Zuständen, die für einen stabilen Griff stehen. Somit kann es sein, dass trotz einer hohen Explorationsrate in einer Region, in der sich viele positiv bewertete Zustände befinden, immer genau diese, bereits explorierten Zustände, ausgewählt werden. Somit verzögern auch diese Zustände den Lernprozess. Die entwickelte Lösung für das Problem besteht in einer dynamischen Aktionsbeschränkung, im Folgenden automatische Aktionsreduzierung genannt.

Automatische Aktionsreduzierung

Durch die automatische Aktionsreduzierung werden Aktions-Wertepaare, die zu einem instabilen Griff führen, aus dem dynamischen Zustands-Aktionsgraphen gelöscht. Somit werden sie nach einmaliger Exploration nicht wieder ausgewählt, und das Lernen wird dadurch beschleunigt. Dies stellt keinen Nachteil dar: da die Griffgenerierung in einer statischen Umgebung erfolgt, ist nicht davon auszugehen, dass bei einer erneuten Exploration der Griff stabil ist. Das Entfernen eines solchen Aktions-Wertepaares aus der Q-Tabelle hat Einfluss auf die Verteilung der Gewichtung. Da aber ein TD(λ)-Verfahren benutzt wird, verteilen sich die Gewichtungen bereits während des Lernvorgangs über die Zustände einer Episode. Somit geht der Einfluss des Zustandes nicht verloren.

Das Löschen von Zuständen und Aktionen aus denen ein stabiler Griff resultiert, ist nicht ohne Weiteres möglich. Zwar verteilen sich auch in diesem Fall die Gewichte über die Zustände einer Episode. Da aber die positiv bewerteten Aktionen ihre Gewichte weiter im Netz verteilen sollen, damit auch Nachbarzustände exploriert werden, werden diese erst nach dem Erreichen von 95% des Rewards entfernt. Durch diese Wahl der oberen Schranke, die sich in den Experimenten bewährt hat, in Relation zum Reward wird gewährleistet, dass der Zustand hinreichend oft besucht worden ist und somit seine positive Gewichtung im Netz verteilt werden konnte.

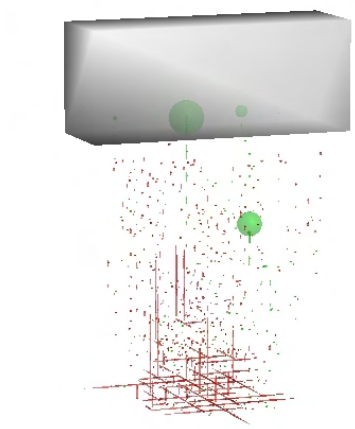
Somit wird die Update-Regel für alle Terminal-Zustände zu:

$$Q(s,a) \leftarrow \begin{cases} Q(s,a) + \alpha \left[r + \gamma Q(s^{'}, a^{'}) - Q(s,a) \right] & \text{falls } 0 \leq Q(s,a) < r * 0.95 \\ \text{Entferne } Q(s,a) \text{ aus } Q & \text{sonst} \end{cases} \quad \forall Q(s,a) \in S_T \tag{6.4}$$

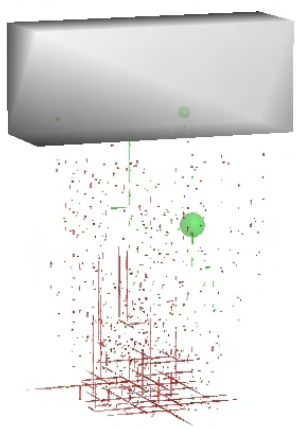
mit S_T als Menge aller Terminal-Zustände.

Ein exemplarisches Beispiel für Lernen ohne und mit automatischer Aktionsreduzierung ist in Abbildung 6.14 und 6.15 dargestellt. Abbildung 6.14 zeigt Ausschnitte aus einem Lernvorgang, bei dem das Greifen eines Quaders gelernt wird. Der Zustand mit dem positiv bewerteten Griff in der Nähe des Objekts wird beim Übergang von Episode 101 zu Episode 102 gelöscht. Wie zu sehen ist, verteilen sich die Gewichte trotzdem über das Positionsraster. Die Gewichte für Griffe werden nach einiger Zeit gelöscht. Hierdurch ergibt sich eine Exploration über den gesamten Suchraum.

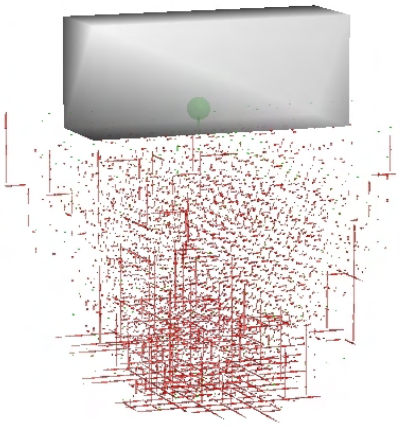
Ein Beispiel für die Berechnung von Griffen ohne automatische Aktionsreduzierung ist in Abbildung 6.15 dargestellt. Die Gewichte der Aktionen wachsen, und die positiven Griffe werden wiederholt von dem Algorithmus angelaufen. Somit findet keine ausreichende Exploration mehr statt, und es wird nur ein Teil des Suchraums exploriert. Dies ist in den Bildern zu Zustand 101 und 102 zu sehen. Hier findet



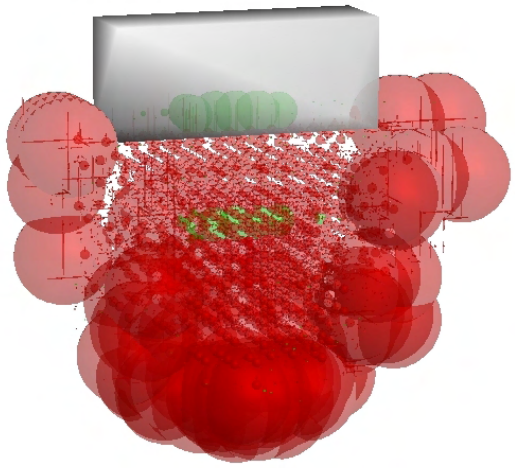
(a) Zustandsraum nach Episode 101



(b) Zustandsraum nach Episode 102

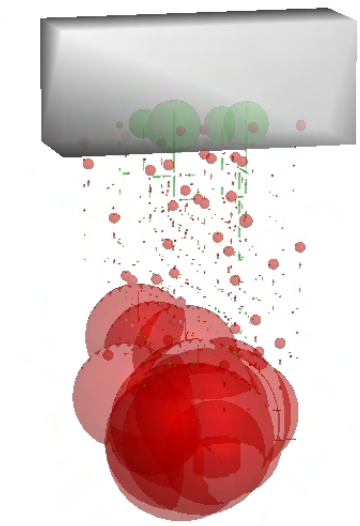


(c) Zustandsraum nach Episode 750

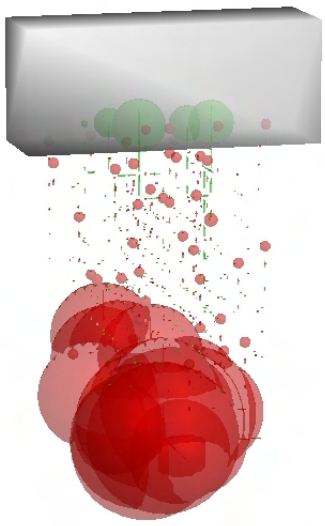


(d) Zustandsraum nach Episode 1.500

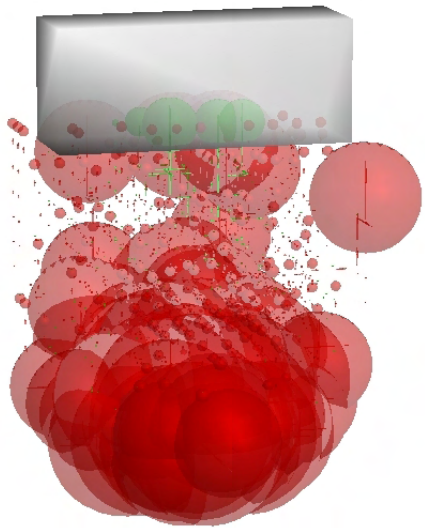
Abbildung 6.14: Visualisierung des Zustands-Aktionsraumes bei einem Lernvorgang mit automatischer Aktionsreduzierung: nach Zustand 101 werden zwei positiv bewertete Zustände für einen Griff angezeigt (grüne Kugeln), nach Zustand 102 ist eine der Aktionen für einen Griff aus dem Aktionsraum entfernt worden. Nach Zustand 750 wird ein positiv bewerteter Griff im Zustands-Aktionsraum angezeigt. Das Bild nach Episode 1.500 zeigt den gesamten Zustands-Aktionsraum, alle Aktionen die zwischenzeitlich entfernt wurden, sind wieder eingefügt worden.



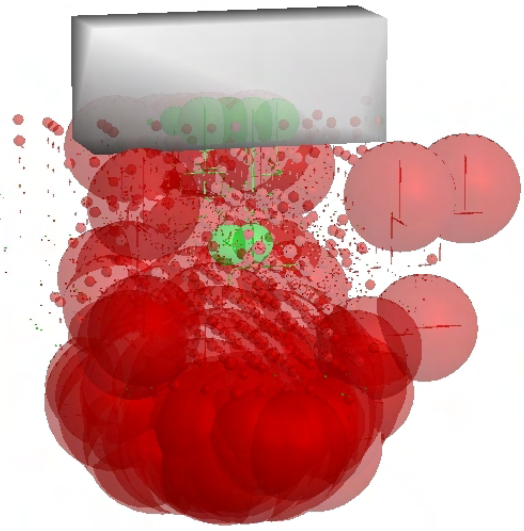
(a) Zustandsraum nach Episode 101



(b) Zustandsraum nach Episode 102



(c) Zustandsraum nach Episode 750



(d) Zustandsraum nach Episode 1.500

Abbildung 6.15: Visualisierung des Zustands-Aktionsraumes bei einem Lernvorgang ohne automatische Aktionsreduzierung: Nach Episode 101 sind vier positiv bewertete Aktionen für einen Griff zu sehen, sowie einige negativ bewertet Aktionen für Griffe. Eine Veränderung zwischen dem Zustandsraum nach Episode 101 und 102 ist kaum wahrnehmbar. Wird der Zustandsraum nach Episode 1.500 mit dem Zustandsraum mit Aktionsreduzierung nach Episode 1.500 verglichen, so sieht man, dass durch den Einsatz der Aktionsreduzierung deutlich mehr Zustände exploriert worden sind.

so gut wie keine weitere Exploration statt, im Gegensatz zu dem Lernvorgang mit Aktionsreduzierung (Abb. 6.14). Besonders auffällig ist dies vom Übergang von Episode 750 zu Episode 1.500.

Die quantitativen Einflüsse des Effekts wurden in mehreren Versuchen untersucht. Hierzu wurden in Versuchsserien (drei Serien pro Objekt) Griffe für ein Objekt gelernt. Die Ergebnisse der Analyse sind in Abbildung 6.16 graphisch dargestellt. Wie anhand der Beispiele zu sehen ist, werden mit einem Algorithmus mit automatischer Aktionsreduzierung deutlich mehr stabile Griffe gefunden, als bei einem Standard-Algorithmus. Beide Algorithmen wurden über 1.500 Episoden trainiert, und die Anzahl der Episoden gegen die Anzahl der generierten Griffe aufgetragen. Besonders deutlich wird der Vorteil der automatischen Aktionsreduzierung bei der Betrachtung der Graphen für den Zylinder. Ohne Aktionsreduzierung werden im Durchschnitt 23,5 stabile Griffe gefunden, mit Aktionsreduzierung sind es 68,5 Griffe, also ungefähr dreimal so viele. Bei den Boxen sind es ungefähr doppelt so viele, und bei einem asymmetrischen Objekt, wie in diesem Fall der Banane ca. viermal so viele.

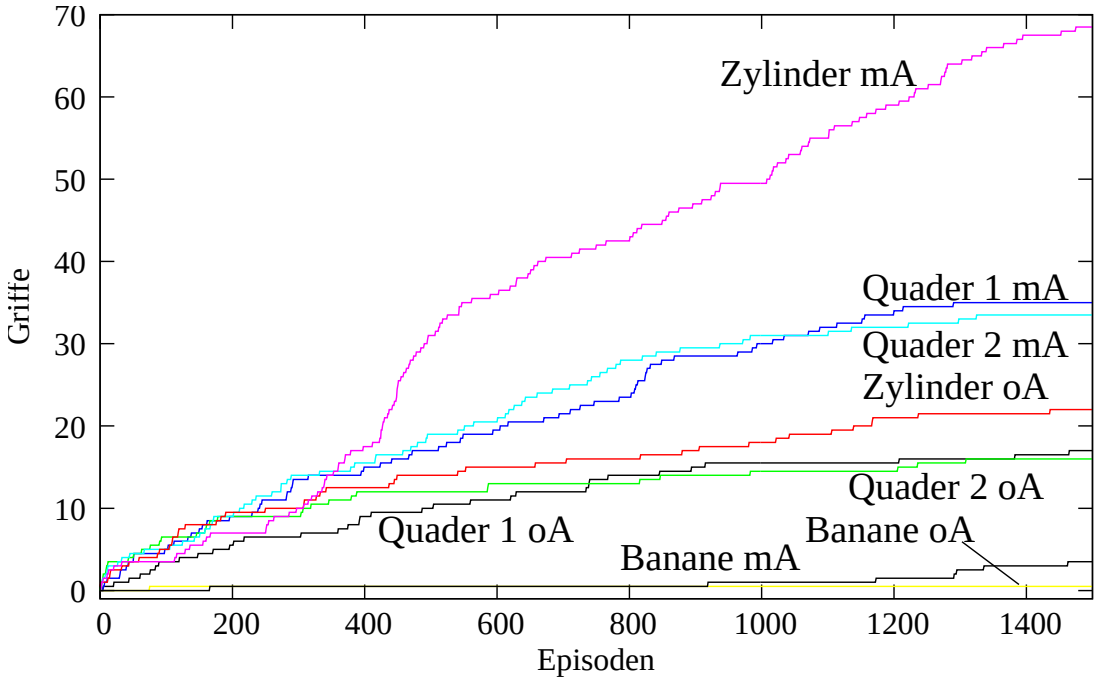


Abbildung 6.16: Vergleich des Algorithmus mit automatischer Aktionsreduzierung (mA) mit einem Verfahren ohne Aktionsreduzierung (oA).

6.7.2 Generalisierung zwischen Objekten

Aufgrund der automatischen Aktionsreduzierung ist es nicht mehr ohne Weiteres möglich, einen bereits trainierten Lernalgorithmus zu verwenden, um für ein weiteres Objekt Griffe zu generieren. Die Übertragung ist zwar theoretisch möglich, allerdings sind die Gewichte des Zustands-Aktionsraum so eingestellt, das sie „in Richtung“ der zuletzt positiv bewerten Zustände „zeigen“. Die zuletzt positiv bewerteten Zustände sind die Greifversuche, die zu einem stabilen Griff geführt haben. Somit „zeigen“ die Gewichte des Zustand-Aktionsraumes in die Region, in der die letzten stabilen Griffe gefunden wurden. Die Gewichte und Zustände für die zuvor gefundenen und positiv bewerteten Griffe sind schließlich

durch die automatische Aktionsreduzierung in den meisten Fällen aus dem Zustandsraum entfernt worden. Werden die Gewichtungen nun genutzt, um Griffe für ein anderes Objekt zu generieren, so werden zunächst die Regionen untersucht, in denen bei dem vorherigen Objekt die letzten stabilen Griffe gefunden worden sind. Dies kann zwar zu einer Beschleunigung des Lernens führen, aber der Effekt ist nur auf diese Region begrenzt. Die Gewichte für den gesamten Zustandsraum müssten erneut berechnet werden.

Um dies zu verhindern, wird eine besondere Form des *Experience-Replay* [Sutton and Barto 1998] eingesetzt. Hierbei werden die Episoden mit den Aktionen und der abschließenden Bewertung zwischengespeichert. Allerdings werden nur jene Episoden endgültig gespeichert, die in einem Zustand terminieren, der zu einem stabilen Griff führt. Nachdem der gesamte Lernvorgang abgeschlossen wurde, werden diese Episoden vom Ziel- zum Startzustand erneut ausgeführt. Die Aktualisierungsregel für die Zustände ist definiert durch:

$$Q(s,a) \leftarrow Q(s,a) + \left(\gamma Q(s',a') - Q(s,a)\right). \tag{6.5}$$

Analog zu Gleichung 6.4, ist der Zustand $Q(s',a')$ in der Episode der Folgezustand von $Q(s,a)$. In diesem Fall beginnt die Updatefunktion aber in einem Terminalzustand, also einem $Q(s',a')$ und aktualisiert die Gewichte für $Q(s,a)$. Durch dieses Episodische-Experience-Replay werden die Gewichte, die zu einem stabilen Griff „führen“, im Zustands-Aktionsraum erneut aufgebaut. Somit ist der gesamte Zustands-Aktionsraum wieder hergestellt, und der Nachteil, der für die Generalisierung durch die automatische Aktionsreduzierung entstand, aufgehoben.

6.7.3 Terminierung des Lernens

Es gibt verschiedene Möglichkeiten, den Lernalgorithmus terminieren zu lassen, die jeweils Vor- und Nachteile aufweisen. Im Folgenden werden einige Kriterien miteinander verglichen:

Maximale Anzahl von Episoden

Bei dieser Variante wird eine bestimmte Anzahl von Episoden berechnet. Die Menge von Griffen, die innerhalb der Episoden gefunden wurden, stellen das Ergebnis dar. Das Problem liegt darin, dass normalerweise nicht abgeschätzt werden kann, nach wie vielen Episoden eine bestimmte Anzahl von Griffen berechnet worden ist, die bestimmte Kriterien bezüglich ihrer Anwendbarkeit oder Stabilität erfüllen.

Maximale Zeit

Hierbei wird der Lernvorgang nach einer bestimmten Zeit abgebrochen. Auch hierbei treten die gleichen Probleme auf, wie bei einer Berechnung über eine bestimmte Anzahl von Episoden. Zusätzlich wächst die benötigte Zeit mit der Anzahl der Polygone der Modelle, so dass z.B. bei einem Modell in einer Zeiteinheit 20, bei einem anderen Modell aber nur 5 Episoden berechnet werden können. Somit ist es schwer abzuschätzen, wie viel Zeit erforderlich ist, um für ein Objekt ausreichend viele Griffe zu berechnen.

Minimale Anzahl von Griffen

Nach der Bestimmung einer zuvor festgelegten Anzahl N von Griffen wird der Algorithmus beendet. Wird N jedoch zu klein gewählt so besteht die Gefahr, dass die Griffe den später gestellten Anforderungen nicht genügen. Wird N zu groß gewählt, läuft die Berechnung möglicherweise endlos, da sich nicht genügend Griffe finden lassen. Dies kann auch der Fall sein, wenn N gleich der Anzahl von maximal möglichen Griffen ist. Da über die teilweise zufällige Aktionsauswahl ein Zufallsprozess mit in das System einfließt, kann nicht garantiert werden, dass immer alle möglichen Griffe gefunden werden.

Minimale Anzahl von Griffen mit bestimmter Qualität

Mit dieser Methode werden Griffe bestimmt, bis eine gewisse Anzahl N_q von Griffen einem Qualitätsmaß genügt. Die Probleme hierbei sind jedoch die gleichen, wie in dem letzt genannten Beispiel.

Maximale Anzahl von Episoden, in denen kein neuer Griff gefunden wurde

Diese Methode stellt eine Variante dar, um möglichst viele Griffe mit einer minimalen Anzahl von Episoden zu bestimmen. Hierbei muss lediglich ein Schwellwert S angegeben werden, der den Lernvorgang terminiert, wenn über S Episoden kein neuer Griff gefunden wurde. Wenn S nicht deutlich zu klein gewählt wurde, kann davon ausgegangen werden, dass entweder alle oder zumindest fast alle Griffe für ein Objekt gefunden worden sind und weitere erst mit einem enormen zusätzlichen Aufwand gefunden werden können.

6.8 Experimente

In diesem Abschnitt werden die Experimente, die für die Evaluation des Lernvorgangs durchgeführt worden sind, beschrieben. Hierbei wird zunächst die Anwendbarkeit an geometrischen Grundformen gezeigt und anschließend auf eine Reihe von Alltagsgegenständen erweitert. Um bereits in dieser Phase die Generalisierungseigenschaften des Lernens auszunutzen, werden die Objekte in Bezug auf die dominierende geometrische Grundform grob klassifiziert, und dann Griffe mit einem bereits entsprechend vortrainierten Algorithmus berechnet.

Die im Folgenden gezeigten Graphen zur Veranschaulichung des Lernvorgangs beruhen auf den Mittelwerten mehrerer Lern-Durchläufe. Hieraus ergibt sich zum einen eine glatte Kurve, und zum anderen verliert der Zufallsprozess, der in den Lernvorgang mit eingeht, an Gewicht. Die Berechnung von Griffen erfolgte über 40.000 Episoden. Somit ist zum einen sichergestellt, dass eine ausreichende Zahl von Lernschritten durchgeführt worden ist, und zum anderen, dass ausreichend viele Griffe für jedes Objekt gefunden werden. Zwar ist es in den meisten Fällen noch möglich, auch nach mehr als 40.000 Episoden noch weitere stabile Griffe zu finden, aber bereits bei 40.000 Episoden werden ausreichend viele Griffe gefunden. Wie die Experimente zeigen, sind in der Regel auch bereits bei 20.000 Episoden ausreichend viele Griffe gefunden worden, aber mit 40.000 Episoden kann gezeigt werden, dass das eingesetzte Verfahren nicht nur kurzfristig bessere Ergebnisse liefert als die Breitensuche, sondern langfristig erfolgsversprechender ist.

In der Regel werden die Objekte aus zwei unterschiedlichen Richtungen gegriffen. Zum einen von der Seite und zum anderen von oben. Von der Seite sind Griffe, bei denen die Handfläche parallel zur Z-Y-Ebene des Objekts liegt. Bei Griffen von oben liegt die Handfläche parallel zur X-Y-Ebene des Objekts. Ein Griff von oben und ein Griff von der Seite für eine Tasse sind in Abbildung 6.17 dargestellt.

Die generierten Griffe werden in diesem Abschnitt nicht demonstriert. Eine Auswahl der Griffe für jedes der Objekte wird in Kapitel 7.2 mit dem Service-Roboter TASER umgesetzt. Hierbei wird auch gezeigt, in wie weit sich die Simulationsergebnisse auf die Realität übertragen lassen.

Im Folgenden wird ein Reinforcement-Algorithmus, der zuvor beschriebene TD(λ)-Algorithmus, als Lerner bezeichnet. Ein trainierter Lerner ist dabei ein Algorithmus, der bereits einige Episoden lang gelernt hat, Griffe zu generieren. Der untrainierte Lerner hingegen verfügt über kein „Vorwissen“, hat also noch keine Episode gelernt. Als „Breitensuche“ oder auch „Verfahren auf Basis der Breitensuche“ wird der Algorithmus bezeichnet, der wie in Abschnitt 6.5 beschrieben in der Lage ist, Griffe zu berechnen.

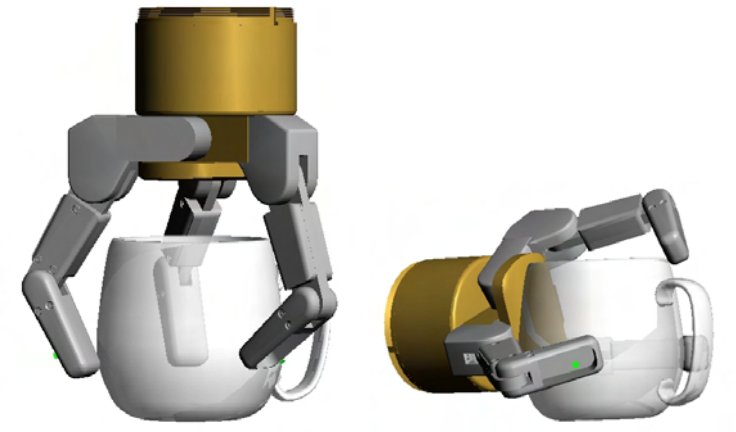


Abbildung 6.17: Griffe für eine Tasse von oben und von der Seite.

Die Vergleiche, die zwischen dem Reinforcement-Lerner und der Breitensuche angestellt werden, beziehen sich auf den trainierten Lerner, falls entsprechende Daten vorhanden sind. Eine Differenzierung zu dem untrainierten Lerner erfolgt in diesem Fall nicht.

6.8.1 Aktionsmenge

Der einfachste Fall besteht in dem Greifen eines Objekts von einer Seite ohne Orientierungsänderung und ohne Änderung des Spreizwinkels. Die Aktionen, die für den Lernvorgang ausgewählt worden sind, sind in Tabelle 6.3 aufgeführt. Wie bereits zuvor erwähnt, ist die Aktion „Translation -Y“ mit dem Faktor 0.25 initial leicht stärker gewichtet als die übrigen Aktionen, da davon ausgegangen wird, dass sich die stabilsten Griffe in Positionen ergeben, bei denen ein Kontakt zwischen der Handfläche und dem Objekt besteht (Powergriff). Dies bedeutet wiederum, dass sich die Hand möglichst nah am Objekt befinden muss. Die in Tabelle 6.3 aufgeführten Aktionen werden im Folgenden auch als *Aktionsmenge 1* oder *Aktionen 1* bezeichnet. Als Erweiterung der Aktionsmenge 1 gibt es noch einen weiteren Satz an Aktionen, die *Aktionsmenge 2*. Dieser ist in Tabelle 6.4 dargestellt.

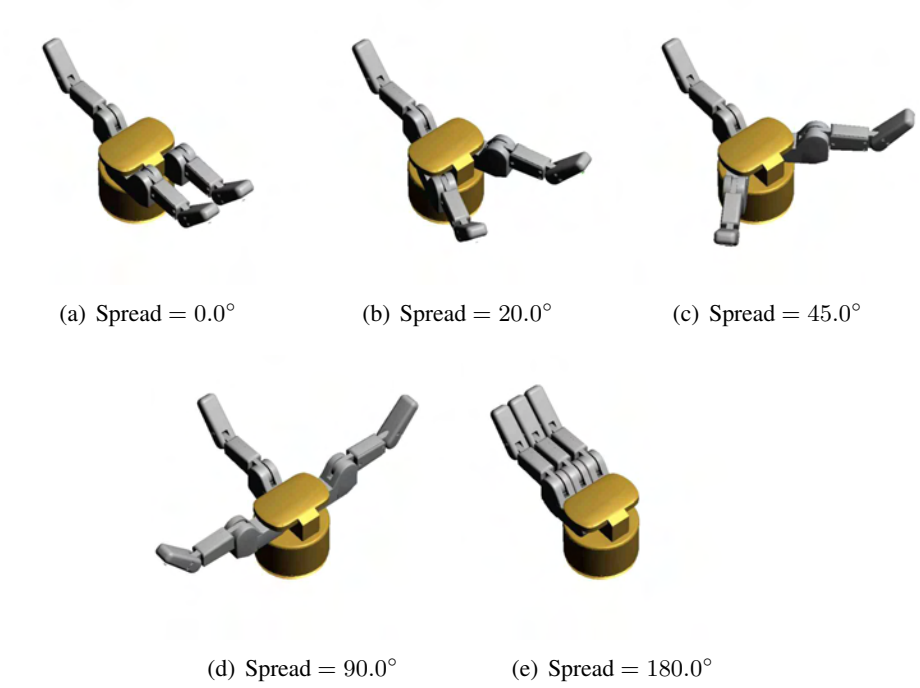
Aktion	initiale Gewichtung
Translation ±X,±Z	0.0
Translation -Y	0.25
Griff	0.0

Tabelle 6.3: Aktionsmenge 1 für die Experimente.

Diese Aktionsmenge beinhaltet zusätzlich zu den in Aktionsmenge 1 aufgeführten Aktionen, eine Orientierungsänderung entlang der Y-Achse. Anschaulich kann dies als Orientierung des Daumens, bzw. des F3-Fingers der BarrettHand aufgefasst werden. Außerdem sind Änderungen des Spreizwinkels der Hand möglich. Die möglichen Werte für den Spreizwinkel sind ebenfalls in Tabelle 6.4 gegeben. Sie stellen eine Erweiterung der in [Miller et al. 2003] vorgeschlagenen vier Griffformen der BarrettHand dar. Die Formen Sphärischer-, Zylindrischer-, Präzisions- und Haken-Griff wurden um

Aktion	initiale Gewichtung
Translation ±X,±Z	0.0
Translation -Y	0.25
Rotation $\pm Y$	0.0
Spreizwinkel 0.0	0.0
Spreizwinkel 20.0	0.0
Spreizwinkel 45.0	0.0
Spreizwinkel 90.0	0.0
Spreizwinkel 180.0	0.0
Griff	0.0

Tabelle 6.4: Aktionsmenge 2 für die Experimente.



(a) Spread = 0.0° (b) Spread = 20.0° (c) Spread = 45.0°

(d) Spread = 90.0° (e) Spread = 180.0°

Abbildung 6.18: Die möglichen Spreizwinkel der BarrettHand. Die Griffformen (von links nach rechts) sind Zylindrischer-, Präzisions-, Sphärischer-, Parallel und Haken-Griff.

die Form des Parallel-Griffs ergänzt. Dieser kann als eine Art Zangen-Griff eingesetzt werden. Diese fünf Griffformen sind in Abbildung 6.18 dargestellt. Dies entspricht auch der von Cutkosky und Wright in [Cutkosky and Wright 1986] vorgeschlagenen Taxonomie der Griffe, unter Berücksichtigung der Möglichkeiten die sich mit einer BarrettHand ergeben, die in Abbildung 6.19 dargestellt ist.

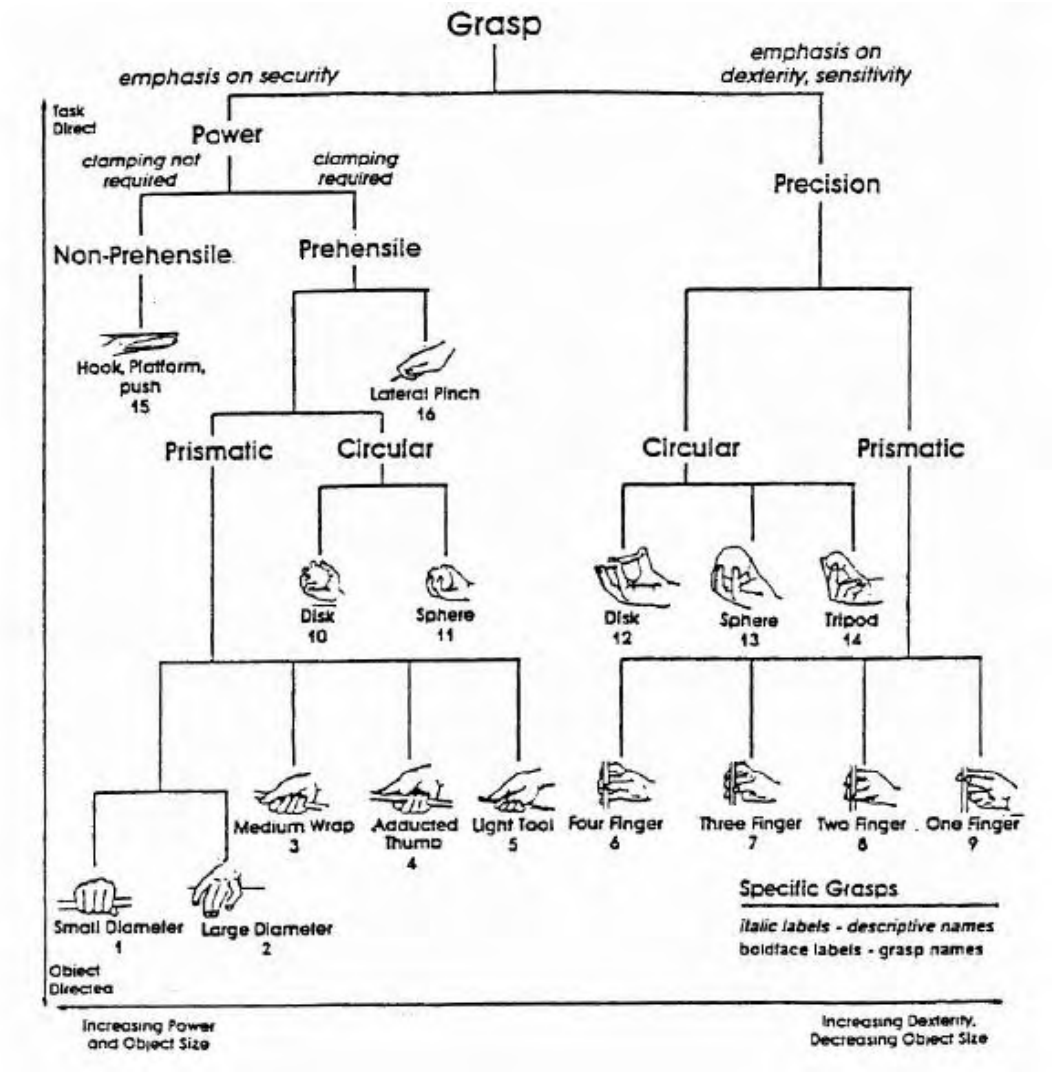


Abbildung 6.19: Griff-Taxonomie nach [Cutkosky and Wright 1986].

Die Orientierungsänderung des Daumens und die gleichzeitige Änderung des Spreizwinkels der Hand kann dazu führen, dass unterschiedliche Konfigurationen in den gleichen Kontaktpunkten am Objekt resultieren. Somit bietet sich unter Umständen die Möglichkeit, dass eine Konfiguration aufgrund von Einschränkungen nicht ausführbar ist, die andere aber dennoch angewendet werden kann. Diese Eigenschaft des Systems wurde bewusst gewählt; die Zustände werden nicht zu einem Zustand fusioniert. Der dadurch entstehende Mehraufwand zur Datenspeicherung ist vernachlässigbar, zieht man die hinzugewonnene Flexibilität in die Betrachtung ein. Da die Aktionsmenge 2 die Aktionsmenge 1 beinhaltet, werden die Experimente mit Aktionsmenge 2 durchgeführt. Die Berechnung von Griffen ohne Orien-

tierungsänderung der Hand und ohne eine Änderung des Spreizwinkels entspricht ebenfalls nur einer unvollständigen Analyse des Problems.

6.8.2 Greifen eines Quaders

Ein Quader ist eine der einfachsten geometrischen Grundformen. Mit seiner Hilfe lassen sich aber bereits eine ganze Menge von Gegenständen in unserer Umwelt beschreiben. Die Beschreibung des Quaders, der für die folgenden Experimente ausgewählt wurde, ist in Tabelle 6.5 gegeben.

Breite	120 mm
Tiefe	40 mm
Höhe	50 mm
Gewicht	100 g
Max. zulässige Kraft	20 N

Tabelle 6.5: Quader für Experimente

Die Ergebnisse des Lernvorgangs werden mit denen der Griffgenerierung durch Breitensuche verglichen. Für die beidem Algorithmen ist der gleiche Startpunkt gewählt worden. Das Lernverfahren wurde dabei über 20.000 Episoden trainiert. Bei der Breitensuche werden die ersten 20.000 Greifversuche für die Evaluation verwendet. Um ein repräsentatives Ergebnis zu erzielen, sind die dargestellten Werte des Lernens Mittelwerte von vier Lernvorgängen. Dies gilt auch für die folgenden Experimente, falls nicht anders angegeben.

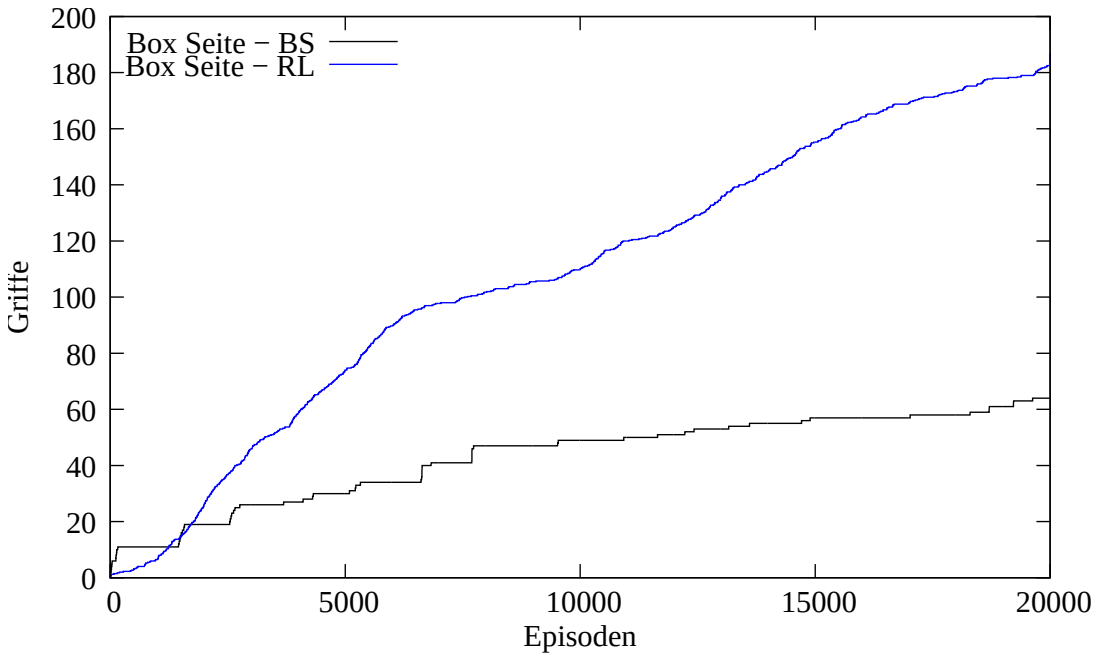


Abbildung 6.20: Ergebnisse der Griffgenerierung für einen Quader. „BS“ steht für Breitensuche und „RL“ für Reinforcement-Lerner

Die Ergebnisse der Griffgenerierung sind in Abbildung 6.20 dargestellt. Hierbei steht „BS“ als Abkürzung für Breitensuche und „RL“ für Reinforcement-Lernen. Wie aus der Grafik ersichtlich wird, ist das Lernverfahren bereits nach ca. 1.750 Episoden der Breitensuche deutlich überlegen. Im Mittel wurden für den Quader nach 20.000 Episoden 142,2 Griffe mittels Lernen und 64 Griffe mittels Breitensuche generiert. Das Lernverfahren war also in der Lage, mit der gleichen Anzahl an Greifversuchen 122,2% mehr stabile Griffe zu berechnen als die Breitensuche.

Einer der Lerner ist anschließend benutzt worden, um Griffe für weitere Objekte, die einen Quader als Grundform haben, zu generieren.

6.8.2.1 Greifen eines Schokoriegels

Als konkretes Beispiel für einen Quader wurde ein Schokoriegel ausgewählt. Dieser ein relativ kleines Objekt mit den Dimensionen von ca 20 mm $\times$ 30 mm $\times$ 100 mm und einem Gewicht zwischen 50 g und 90 g. Er eignet sich somit hervorragend für alle möglichen Griffe, die mit einer BarrettHand ausgeführt werden können. Deshalb wird Aktionsmenge 2 eingesetzt, um entsprechende Griffe zu generieren.

Da die Objektbreite ungefähr der Objekthöhe entspricht, sind auch für dieses Objekt nur Griffe von der Seite generiert worden. Unter Berücksichtigung der Symmetrieeigenschaften des Objekts sind alle Griffe, die von der Seite berechnet worden sind, auch auf Griffe von oben auf das Objekt übertragbar. Die Ergebnisse der Griffgenerierung sind in Abbildung 6.22 dargestellt. Wie in der Abbildung zu sehen ist, ist die Griffgenerierung mittels der Lernverfahren zwischen den Episoden 3.033 und 20.095 besser als die Generierung mittels Breitensuche. Auch wird deutlich, dass das Lernverfahren, das auf einem bereits vortrainierten Lerner aufsetzt, in der Lage ist mehr Griffe in der gleichen Anzahl von Episoden zu generieren, als der untrainierte Lerner. Im Mittel ist der untrainierte Lerner 13,1 Griffe schlechter als der trainierte. Die Breitensuche ist im Mittel 2,77% oder 1,82 Griffe besser als der Lernalgorithmus mit „Vorwissen“. Die Griffgenerierung mittels Lernen ist in diesem Fall schlechter als die Generierung über die Breitensuche, da die Griffe, die den Sprung in dem Graph für die Breitensuche bei ca. 35.000 Episoden auslösen, sehr nah am Rand des Objekts liegen. Dies ist aber ein Bereich, der von dem Lernverfahren aufgrund der Gewichtung nur sehr schwache Beachtung findet und somit nicht ausreichend exploriert wird. Dies wirkt sich bei einem relativ kleinen Objekt nachteilig aus. Die maximale Differenz zwischen den beiden Verfahren beträgt im Durchschnitt allerdings nur 33,5 Griffe.

(a) reales Objekt (b) simuliertes Objekt

Abbildung 6.21: Ein Schokoriegel und ein enstprechendes Simulations-Modell.

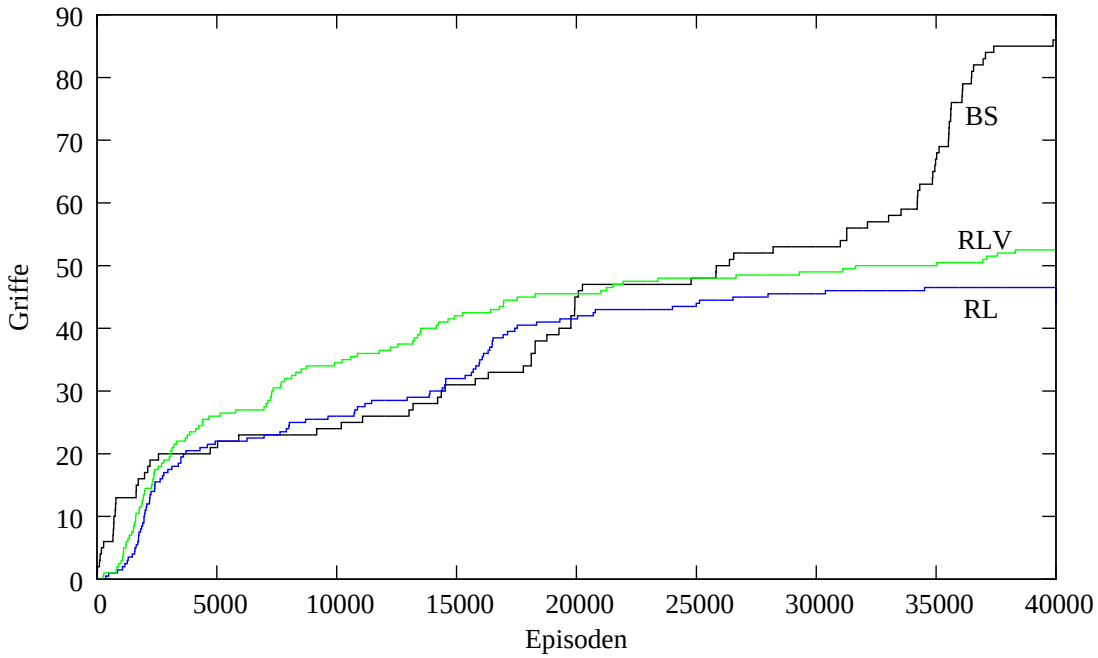
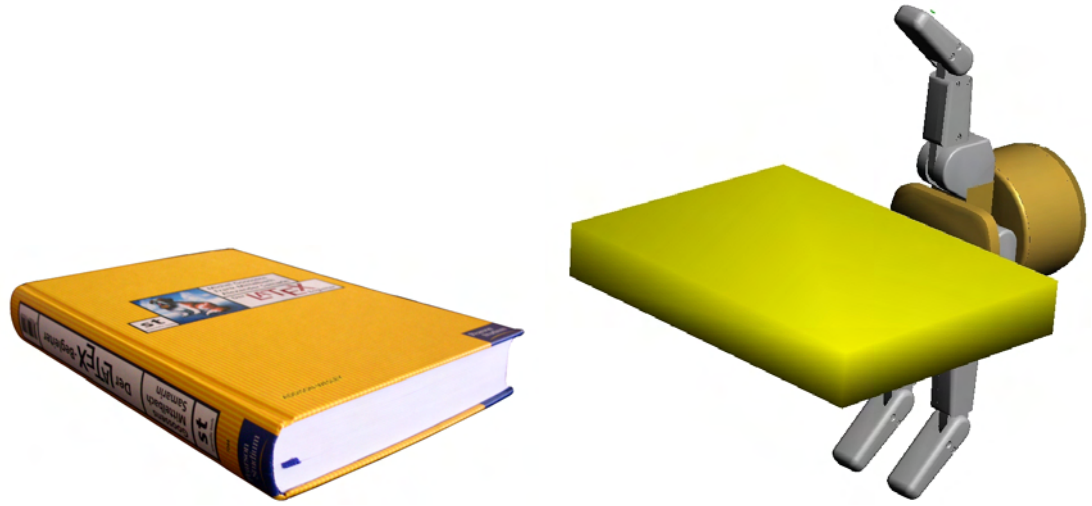


Abbildung 6.22: Ergebnisse der Griffgenerierung für einen Schokoriegel von der Seite. Die Graphen entsprechen der Griffberechnung mittels Breitensuche („BS“), Reinforcement-Lernen („RL“) und vortrainiertem Reinforcement-Lernen („RLV“). Die Breitensuche findet ab ca. Episode 30.000 zusätzliche Griffe am Rand des Objekts während diese bei Reinforcement-Lernen durch eine geringe Gewichtung unterdrückt werden.

6.8.2.2 Greifen eines Buchs

Ein weiteres mögliches Greif-Objekt ist ein Buch. Dieses kann sehr gut durch einen Quader approximiert werden. Das hier ausgewählte Exemplar (Abb. 6.23) hat die Maße 35 mm $\times$ 180 mm $\times$ 245 mm und ein Gewicht von 1100 g. Hierbei werden zwei Griffe an zwei unterschiedlichen Positionen des Buches generiert. Zum einen Griffe von der Seite, die einem Griff aus einem Regal entsprechen, und zum anderen Griffe von vorne, die dem Greifen eines liegenden Buches entsprechen. Da das Buch zu groß ist, um es bei einem Griff von der Seite komplett zu umschließen, werden hierbei die Griffe nur mittels Aktionsmenge 1 generiert. Bei einem Griff von vorne ist es durchaus möglich, das Buch in unterschiedlichen Orientierungen zu greifen. Es kann zwar nach wie vor nicht umschlossen werden, aber durch die Orientierungsänderung kann es durchaus sinnvoll sein, Griffe auch mit unterschiedlichen Spreizwinkeln zu generieren. Um Griffe von vorne zu generieren, wird Aktionsmenge 2 eingesetzt.

Die Ergebnisse der Griffgenerierung von der Seite mit Aktionsmenge 1 sind in Abbildung 6.24 dargestellt. Bei einem Greifversuch von der Seite können maximal 35 unterschiedliche Griffe gefunden werden. Dies ist ein sehr geringer Wert, und alle Verfahren sind in der Lage, die entsprechenden Griffe in weniger als 1.500 Episoden zu berechnen. Auch in diesem Fall ist die Generierung durch die Breitensuche im Mittel 0,85 Griffe besser (7,7%), als die Generierung mittels selbstbewertendem Lernen. Betrachtet man die Ergebnisse für die Generierung mittels Lernen, so ist der vortrainierte Lerner im Durchschnitt 1,24 Griffe (12,3%) besser, als der untrainierte Lerner. Der Effekt des Trainings wirkt sich angesichts der geringen Anzahl von Episoden die zur Berechnung der Griffe erforderlich sind nur sehr



(a) reales Buch (b) simuliertes Buch

Abbildung 6.23: Ein reales Buch und das entsprechende Modell aus der Simulation.

schwach aus. Ab Episode 982 ist der Lerner mit „Vorwissen“ sogar schlechter, als der ohne „Vorwissen“. Dies liegt daran, dass durch die Initialisierung auch Aktionen positiv bewertet werden, die nicht in einem stabilen Griff resultieren. Bei der geringen Zahl von Griffen wirkt sich dies negativ aus.

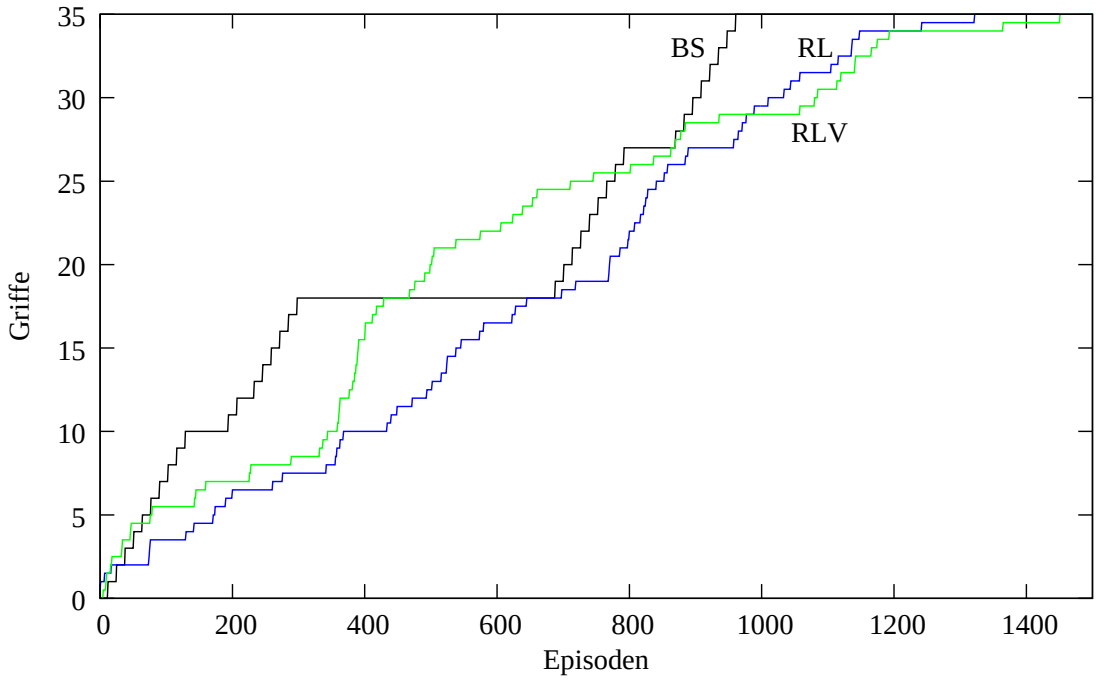


Abbildung 6.24: Ergebnisse der Griffgenerierung für ein Buch für Griffe von der Seite. Es können maximal 35 stabile Griffe gefunden werden.

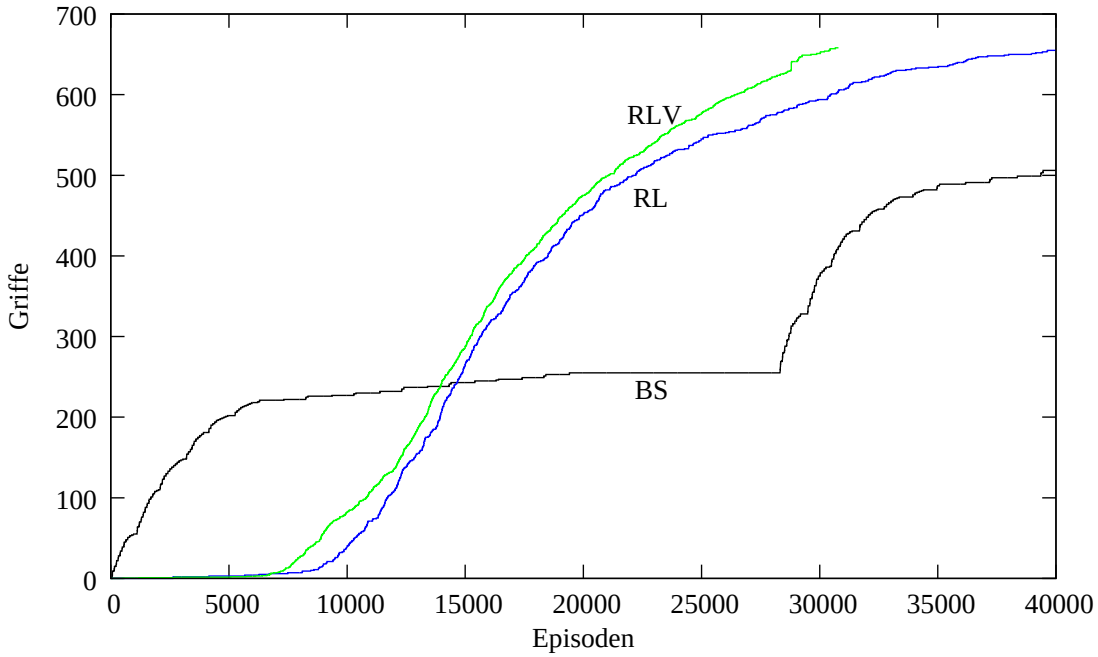


Abbildung 6.25: Ergebnisse der Griffgenerierung für ein Buch.

Die Ergebnisse für die Griffgenerierung für das Buch von vorne sind in Abbildung 6.25 dargestellt. Es wird deutlich, dass die Generierung mittels Reinforcement-Lernen in diesem Fall überlegen ist. Im Mittel ist die Griffgenerierung mittels Lernen um 68,3 Griffe bzw. 13,8% besser als das Verfahren auf Basis der Breitensuche. Die Breitensuche ist im Bereich von ca. Episode 19.200 bis Episode ca. 28.200 nicht in der Lage neue Griffe zu finden, da die Orientierung der BarrettHand in einem Bereich befindet (zwischen 20° und 70°) in dem keine stabilen Griffe gefunden werden können. Der vortrainierte Lerner ist besser als der untrainierte. Durchschnittlich ist der trainierte Lerner in der Lage 25,3 Griffe oder 14,8% besser als der untrainierte Lerner.

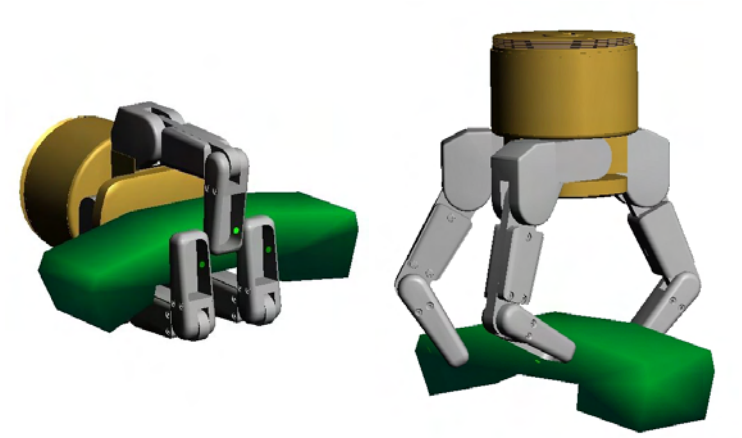
6.8.2.3 Greifen eines Telefonhörers

Die Form eines Telefonhörers kann, wenn man sich das verwendete Modell aus Abbildung 6.26(b) ansieht, ebenfalls von einem Quader abgeleitet werden. In Abbildung 6.26 ist ein realer Telefonhörer und ein entsprechendes vereinfachtes Modell, das für die Simulation verwendet wurde, dargestellt. Einige der Rundungen wurden also nicht modelliert. Die Auswirkungen bei Experimenten mit dem Roboter sind in Abschnitt 7.2.3 beschrieben.

Es wurden Griffe für zwei unterschiedliche Annäherungsrichtungen für den Telefonhörer generiert. Zum einen Griffe von der Seite (Abb. 6.27(a)) und zum anderen von oben (Abb. 6.27(b)). Wie in Grafik 6.28 zu sehen ist, ist die Griffgenerierung mittels Reinforcement-Lernen für Griffe von der Seite ab ca. 8800 Episoden effizienter, als die Generierung mittels Breitensuche. Im Mittel werden mit dem Lernverfahren 56,8% mehr Griffe gefunden, als mit dem Verfahren auf Basis der Breitensuche. Das entspricht im Durchschnitt ca. 66 Griffen bei 40.000 Episoden. Bei der Generierung der Griffe von oben ist dagegen die Breitensuche besser, als die Generierung mittels Reinforcement-Lernen. Hierbei liegt die gleiche Situation vor, wie bei der Generierung der Griffe für das Buch von oben. Durchschnittlich ist die Brei-

(a) realer Telefonhörer (b) simulierter Telefonhörer

Abbildung 6.26: Ein Telefonhörer und ein vereinfachtes Modell aus der Simulation.



(a) Griff von der Seite (b) Griff von oben

Abbildung 6.27: Griffbeispiele für einen Telefonhörer.

tensuche 196,56% bzw. 20,7 Griffe besser als das Lernverfahren. Der trainierte Lerner ist 5,7 Griffe bzw 6,6% besser bei der Generierung von Griffen von der Seite und 2,8 Griffe bzw. 48,3% besser bei der Generierung von Griffen von oben, als der untrainierte Lerner.

6.8.3 Greifen eines Zylinders

Eine weitere geometrische Grundform ist der Zylinder. Die Form des Zylinders ist definiert durch einen Radius r und eine Höhe h. Eine Vielzahl von Objekten, wie Tassen, Flaschen, etc. lassen sich von der Form eines Zylinders abstrahieren. Der Algorithmus wurde zunächst mit einem Zylinder mit einer Höhe von 120 mm und einem Radius von 40 mm über 20.000 Episoden trainiert. Die Ergebnisse des Lernvorgangs und der Breitensuche sind in Abbildung 6.29 dargestellt. Es wurden Griffe für zwei unterschiedliche Orientierungen berechnet. Zum einen Griffe von der Seite, und zum anderen Griffe von oben. Wie in der Grafik zu sehen ist, ist die Griffgenerierung mittels Reinforcement-Lernen in beiden Fällen besser, als die Generierung mittels Breitensuche, bei der Berechnung der Griffe von oben im Durchschnitt um 245,57% (241,5 Griffe) und bei der Berechnung der Griffe von der Seite um 26,77% (74,9 Griffe).

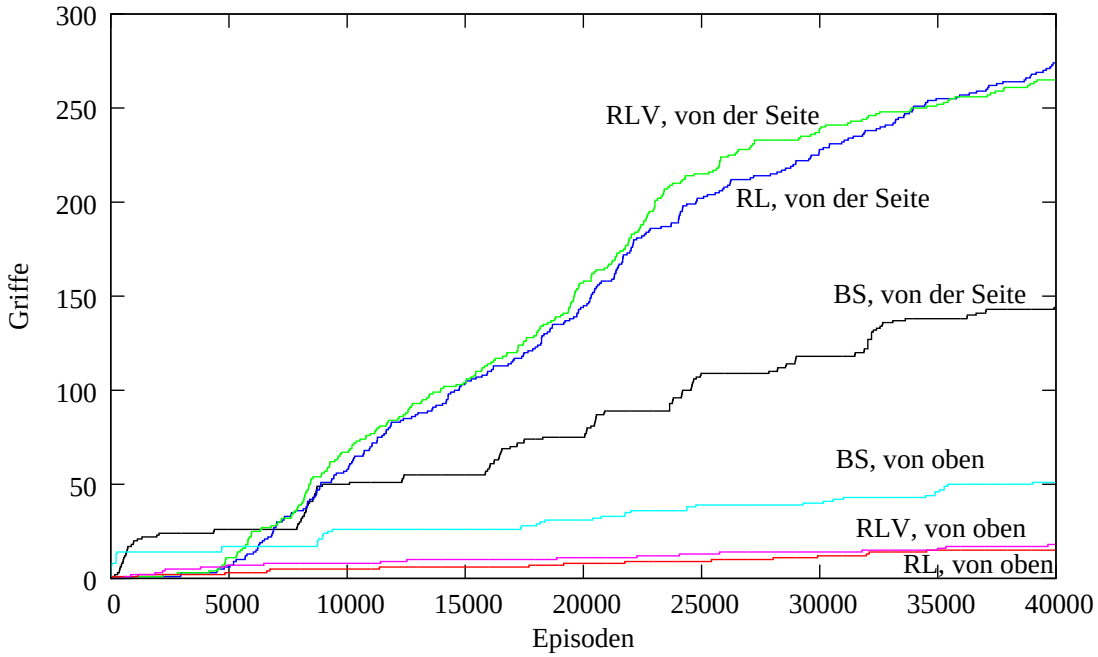


Abbildung 6.28: Ergebnisse der Griffgenerierung für einen Telefonhörer.

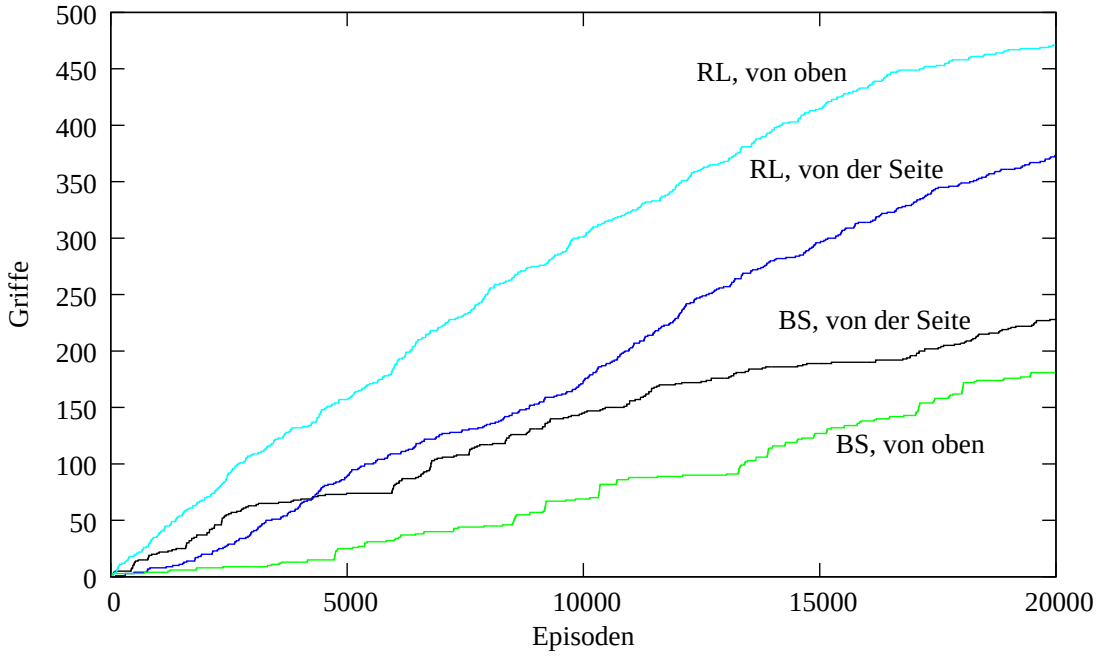


Abbildung 6.29: Ergebnisse der Griffgenerierung für einen Zylinder mit den Parametern h=120 mm und r=40 mm.

(a) reale 0.5 l Flasche (b) simulierte Flasche

Abbildung 6.30: Eine 0.5 l PET-Flasche und ein entsprechendes Modell aus der Simulation.

6.8.3.1 Greifen einer Flasche

Die Form einer Flasche entspricht, bis auf den Flaschenhals, oft der eines idealen Zylinders. In diesem Beispiel werden Griffe für eine 0.5 l PET Flasche generiert. Da die Flasche aus Plastik und somit recht robust ist, wird eine maximale Kraft von 20 N zugelassen. Als Eigengewicht wird eine Masse von 60 g angenommen. Eine Flasche und ein entsprechendes Modell sind in Abbildung 6.30 dargestellt. Für das entsprechende Modell einer Flasche wurden Griffe über 40.000 Episoden generiert. Im Durchschnitt ist das Lernverfahren um 53,1% (68,8 Griffe) effizienter als die Breitensuche bei Griffen von der Seite und um 127,2% (83,6 Griffe) bei Griffen von oben. Wie außerdem noch zu sehen ist, ist der vortrainierte Lerner bei der Berechnung von Griffen der Seite bis ca. Episode 8.000 dem untrainierten Lerner überlegen (129,1% oder 3,3 Griffe im Mittel). Im Mittel, über 40.000 Episoden, ist der trainierte Lerner 4,3% (1,8 Griffe) besser als der untrainierte Lerner. Bei der Griffberechnung von oben ist der vortrainierte Lerner ca. ab Episode 12.000 permanent besser als der nicht-trainierte Lerner (104% im Mittel). Durchschnittlich ist er 3,8 Griffe (2,8%) besser als der untrainierte Lerner (über 40.000 Episoden).

6.8.3.2 Greifen eines Bechers

Die Form eines Bechers ähnelt der eines Zylinders. Lediglich der Griff des Bechers, der „Henkel“, unterbricht die Symmetrie des Zylinders. In Abbildung 6.32 sind ein Becher und das entsprechende Modell aus der Simulation abgebildet. Wie in Grafik 6.33 zu sehen ist, ist die Griffgenerierung mittels Lernen (RLV) bei der Generierung von Griffen von oben deutlich besser als die Breitensuche (BS). Im Mittel ist das Lernverfahren um 76,8% (181,2 Griffe) besser als die Breitensuche. Zu Beginn der Berechnung (bis Episode 5.000) ist das Lernverfahren in der Lage 109,3% oder 17,2 mehr Griffe zu berechnen als die Breitensuche. Im Maximum ist das Lernverfahren um 500,0% (406 Griffe) besser als die Breitensuche. Die Differenz zwischen dem trainierten und dem untrainierten Lerner beträgt in diesem Fall nur 0,5 Griffe. Dieser insignifikante Unterschied läßt darauf schließen, dass es in diesem Fall keine Differenz zwischen den beiden Verfahren gibt. Im Vergleich zur Flasche können für den Becher mehr Griffe von oben gefunden, da hierbei auch Griffe am oberen Rand des Objekts stabil sind. Bei der Flasche können im Bereich des Flaschenhals nur wenig stabile Griffe gefunden werden, daher ergibt sich der relativ große Unterschied zwischen den Objekten.

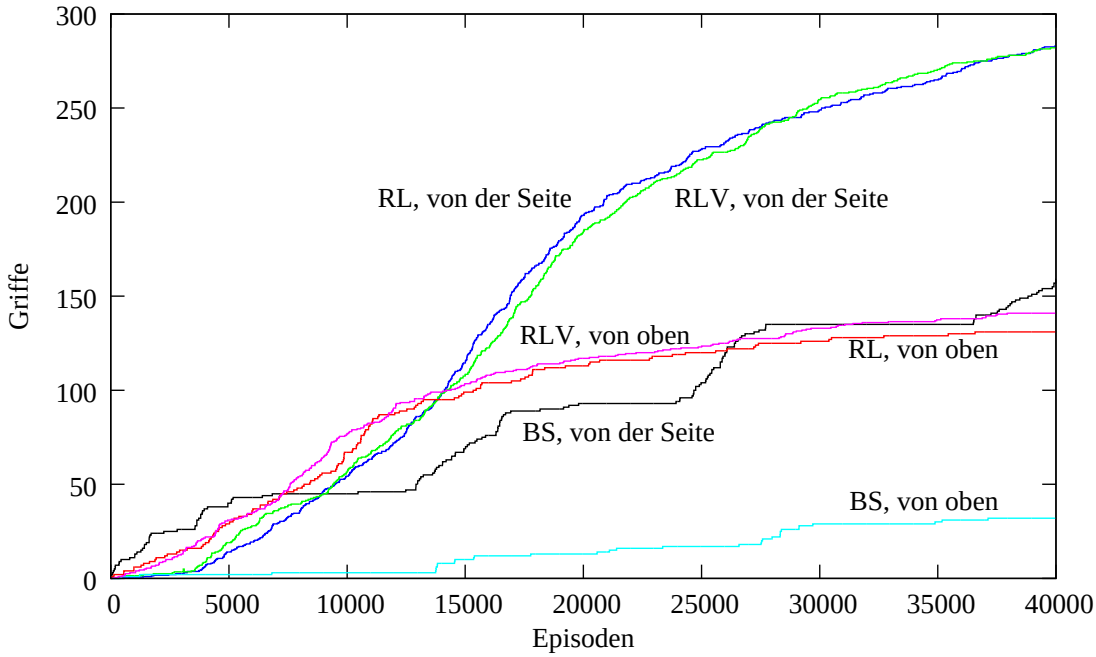


Abbildung 6.31: Ergebnisse der Griffgenerierung für eine 0,5 l PET-Flasche.

(a) realer Becher (b) simulierter Becher

Abbildung 6.32: Ein gewöhnlicher Becher und das entsprechende Modell aus der Simulation.

Bei der Berechnung für Griffe von der Seite ist das Verfahren auf Basis der Breitensuche dem Lernverfahren überlegen. Im Mittel ist die Breitensuche um 73,0% (73,8 Griffe) besser als der trainierte Reinforcement-Lerner. Bei den zuvor gezeigten Experimenten der Griffberechnung von der Seite für den Zylinder und die Flasche, war das Lernverfahren dagegen besser als die Breitensuche. Die Gründe hierfür sind bisher unklar und lassen sich nicht aus den gewonnenen Daten ableiten. Der trainierte Lerner ist 8,4 Griffe bzw. 10,7% besser bei der Generierung von Griffen von der Seite, als der untrainierte Lerner.

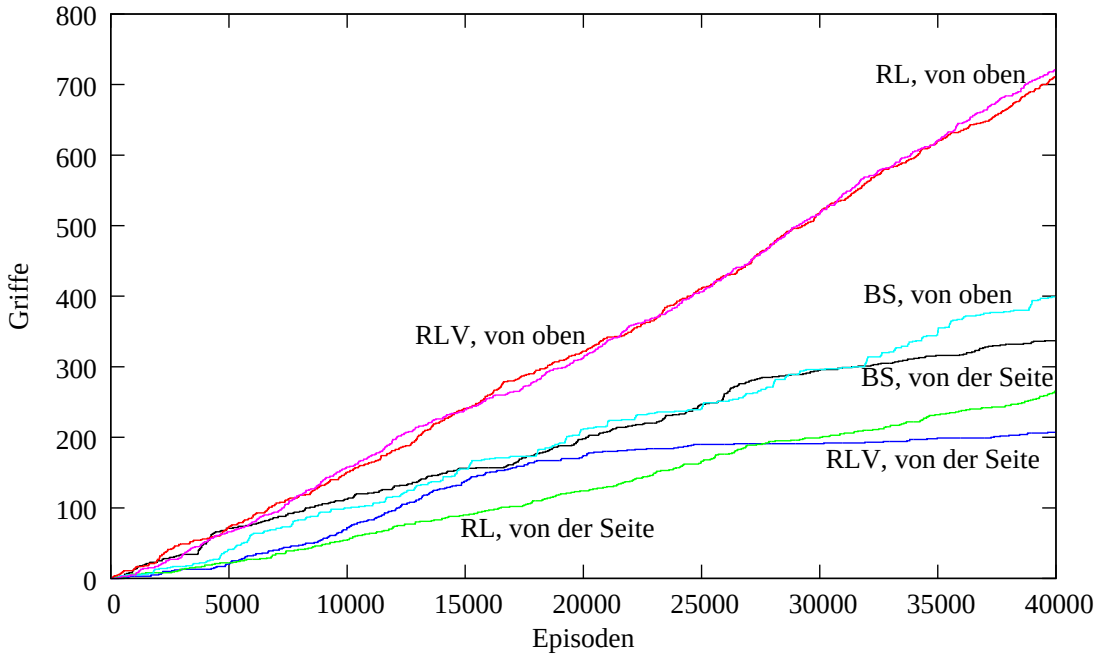


Abbildung 6.33: Ergebnisse der Griffgenerierung für einen Becher.

6.8.3.3 Greifen einer Tasse

Die Form einer Tasse ist ebenfalls von der eines Zylinders abzuleiten. Im Gegensatz zu dem zuvor untersuchten Becher ist die Tasse jedoch bauchiger. Das Gewicht der Tasse aus Abbildung 6.34(a) beträgt 190 g und das Volumen 200 ml.

(a) reale Tasse (b) simulierte Tasse

Abbildung 6.34: Eine Tasse und ein entsprechendes Modell aus der Simulation.

Die Ergebnisse der Griffgenerierung sind Abbildung 6.35 dargestellt. Wie zu sehen ist, ist die Breitensuche zu Beginn der Berechnung, bis Episode 3.432 bei Griffen von der Seite und bis zu Episode 5.772 bei Griffen von oben, effizienter als die Berechnung durch ein Lernverfahren. Danach, also bis Episode 40.000, ist das Lernverfahren aber deutlich effizienter als die Griffgenerierung mittels Breitensuche. Im

Durchschnitt ist der Lerner bei der Generierung von Griffen von oben bis Episode 5.772 41,2% (19,0 Griffe) schlechter als die Breitensuche und von Episode 5.773 bis Episode 40.000 60,8%, oder 217,3 Griffe durchschnittlich besser als die Breitensuche. Analog hierzu ist die Breitensuche, bei der Berechnung von Griffen von der Seite, bis Episode 3.423 18,2%, oder 2,8 Griffe, besser und von Episode 3.424 bis Episode 40.000 um 40,2% (112,9 Griffe im Durchschnitt) schlechter als das TD(λ)-Verfahren.

Der vortrainierte Reinforcement-Lerner ist in der Lage mehr Griffe zu berechnen als der untrainierte Lerner. Der Effizienzgewinn ist mit durchschnittlich 9,9% (17,9 Griffe) für die Berechnung für Griffe von der Seite und 2,3% (13,0 Griffe) für die Berechnung für Griffe von oben nicht sehr groß. Allerdings ist der trainierte Algorithmus in der Lage, schneller stabile Griffe zu finden als der untrainierte. Wird die Tasse von der Seite gegriffen, so findet der trainierte Lerner den ersten Griff im Mittel nach 50,7 Episoden und der untrainierte erst nach 544,3 Episoden.

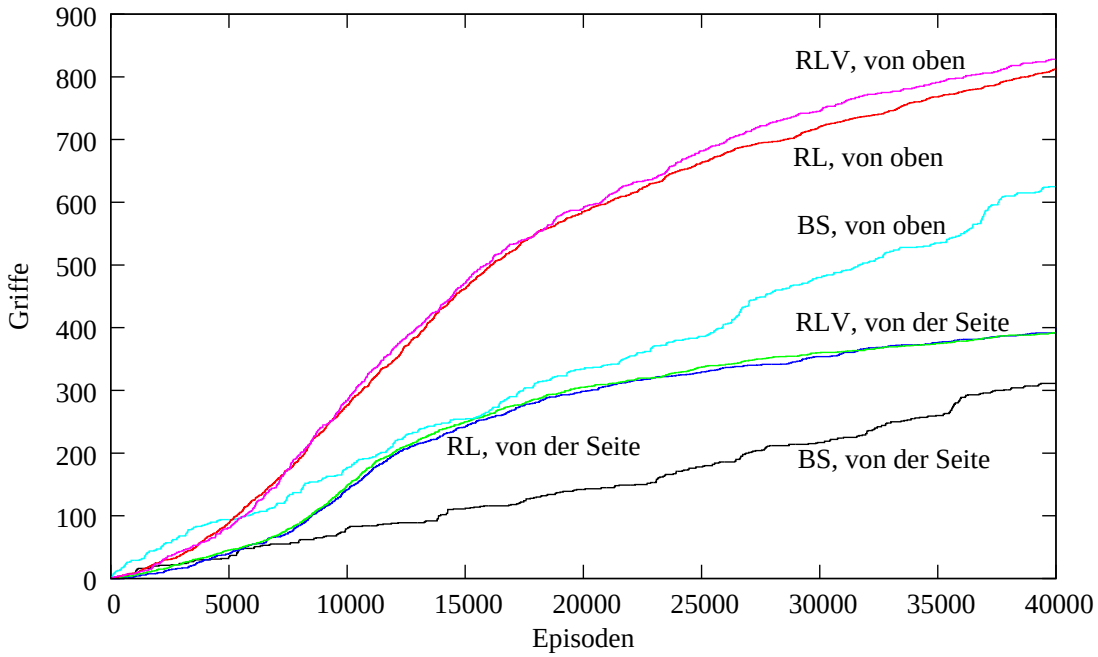
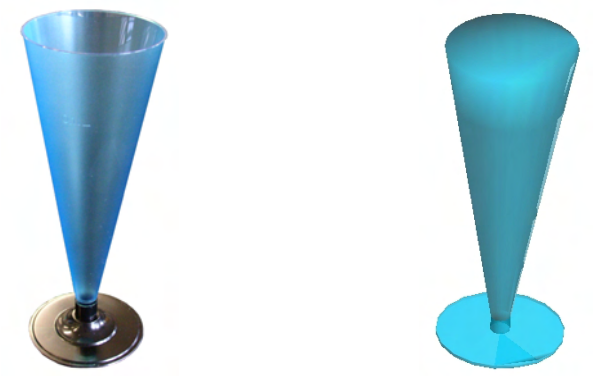


Abbildung 6.35: Ergebnisse der Griffgenerierung für eine Tasse.

6.8.3.4 Greifen eines Sektglases

Die Form des Sektglas leitet sich zwar eher von der Form eines Kegelstumpfs ab, kann aber dennoch von der Form eines Zylinders abstrahiert werden. Wird die Form des Sektglas von der Form eines Zylinders abstrahiert, kann mit dem vortrainierten Lerner die Generalisierungsfähigkeit validiert werden. Das Sektglas und ein entsprechendes Modell sind in Abbildung 6.36 dargestellt.

Die Ergebnisse der Griffgenerierung sind in Abbildung 6.37 dargestellt. Auch in diesem Fall wird deutlich, dass die Griffgenerierung mittels selbstbewertendem Lernen der Generierung mittels Breitensuche überlegen ist. Werden Griffe von der Seite für das Objekt berechnet, so ist das Lernverfahren im Durchschnitt 126,3% besser (entspricht 19,7 Griffen) als die Breitensuche. Bei Griffen von oben ist das Lernverfahren durchschnittlich 452,5% besser (entspricht 100,9 Griffen). Maximal ist das Lernverfah-



(a) reales Sektglas

(b) simuliertes Sektglas

Abbildung 6.36: Ein Sektglas und ein entsprechendes Modell aus der Simulation.

ren bei Greifversuchen von der Seite 178,3% besser, und bei Griffen von oben 1.100% besser, als die Breitensuche.

Vergleicht man den Lernvorgang des trainierten Lerners mit dem des untrainierten, so fällt ein ähnlicher Verlauf wie zuvor bei der Tasse auf. Bei der Griffberechnung von der Seite gibt es nur einen zu vernachlässigenden Unterschied zwischen beiden Varianten. Werden Griffe von oben für das Sektglas berechnet, ist der trainierte Lerner leicht besser (im Mittel 7,46 Griffe oder 105%) als der untrainierte Lerner.

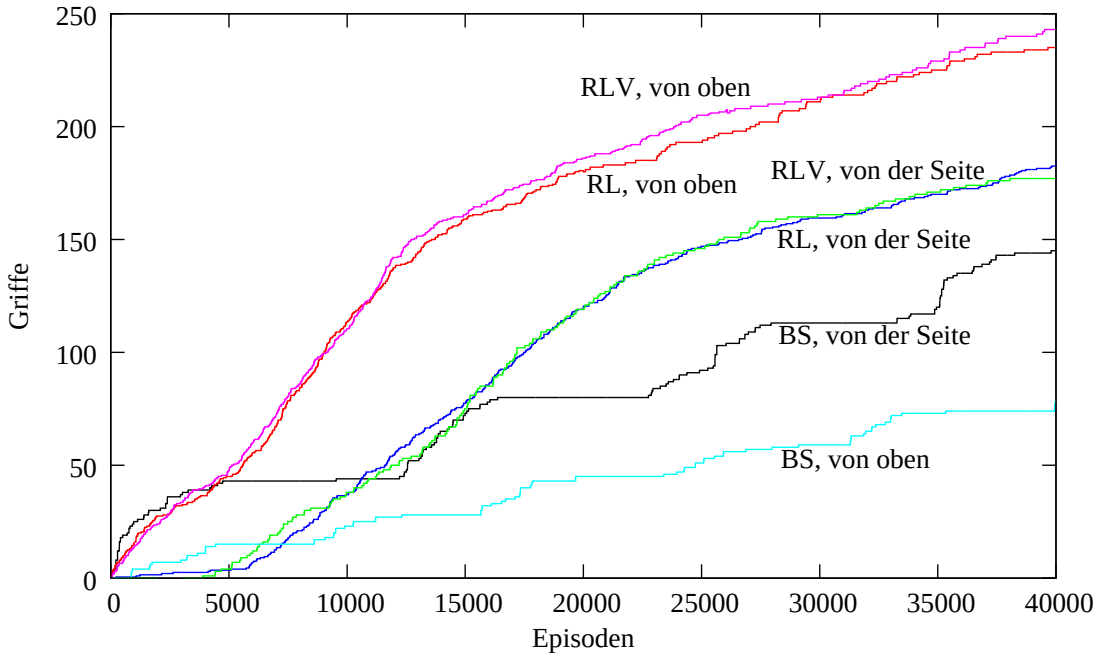


Abbildung 6.37: Ergebnisse der Griffgenerierung für ein Sektglas.

6.8.4 Greifen einer Kugel

Eine Kugel ist eine der geometrischen Grundformen. Für die Evaluation wurde eine Kugel mit einem Durchmesser von 80 mm mit einem Gewicht von 100 g und einer metallischen Oberfläche betrachtet. Mittels des Lernverfahrens wurden Griffe über 20.000 Episoden für die Kugel berechnet. Durch die Invarianz einer Kugel bezüglich jeder Rotation, ist die Berechnung von Griffen aus unterschiedlichen Richtungen überflüssig. Die Ergebnisse der Berechnung sind in Abbildung 6.38 dargestellt.

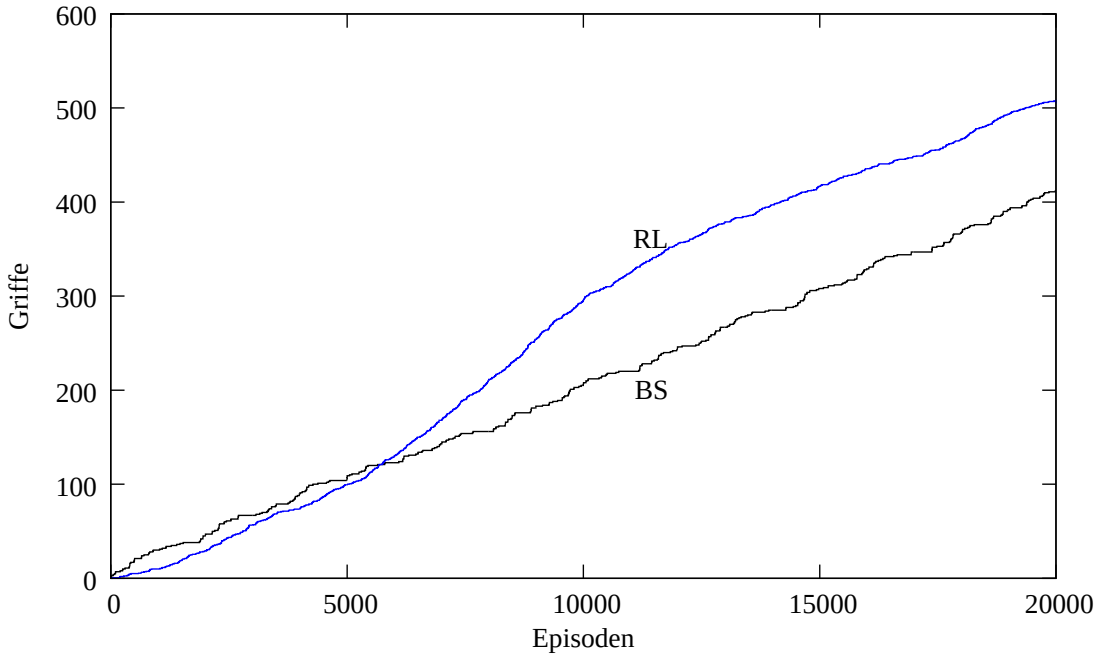


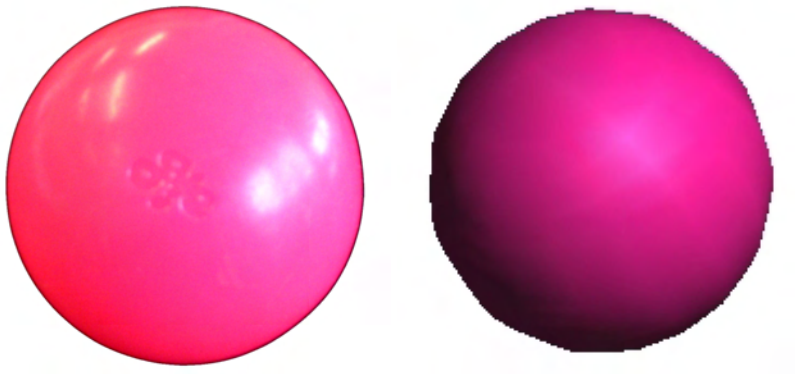
Abbildung 6.38: Ergebnisse der Griffgenerierung für eine Kugel mit einem Durchmesser von 80 mm.

Die Abbildung verdeutlicht, dass das Lernverfahren bei der Griffgenerierung für die Kugel, im Durchschnitt besser ist als die Breitensuche. Im Mittel ist das Lernverfahren um 111,33% bzw. 48,1 Griffe besser als die Breitensuche. Einer der zur Griffgenerierung benutzen Lerner, wird im Folgenden als vortrainierter Lerner benutzt.

6.8.4.1 Greifen eines Balls

Ein Ball stellt die ideale Form einer Kugel dar. In diesem Experiment wurden Griffe für einen Plastikball mit einem Gewicht von 60 g und einem Durchmesser von 70 mm berechnet. Der reale Ball und ein entsprechendes Modell sind in Abbildung 6.39 dargestellt.

Auch in diesem Experiment wurden Griffe über 40.000 Episoden generiert. Die Ergebnisse sind in Abbildung 6.40 dargestellt. Vergleicht man die Graphen für das Reinforcement-Lernverfahren mit dem Verfahren der Breitensuche, so fällt auf, dass das Lernverfahren bis Episode 4.469 weniger Griffe generiert, als die Breitensuche. Statistisch ist das Lernverfahren durchschnittlich 19,6% oder 7,0 Griffe schlechter, als die Breitensuche. In dem Intervall von Episode 4.470 bis Episode 28.000 ist das Lernverfahren besser (um 24,5%) als die Breitensuche und von Episode 28.000 bis Episode 40.000 ist wiederum die Breitensuche besser (um 8,7%). Insgesamt, über 40.000 Episoden, ist das Lernverfahren durchschnittlich



(a) realer Ball (b) simulierter Ball

Abbildung 6.39: Ein Ball mit 70 mm Durchmesser und ein entsprechendes Modell aus der Simulation.

109,6% (21,1 Griffe) besser als das Verfahren auf Basis der Breitensuche. Der trainierte Lerner ist im Durchschnitt 103,35% besser als der untrainierte Lerner. Dies entspricht im Mittel 7,51 Griffen.

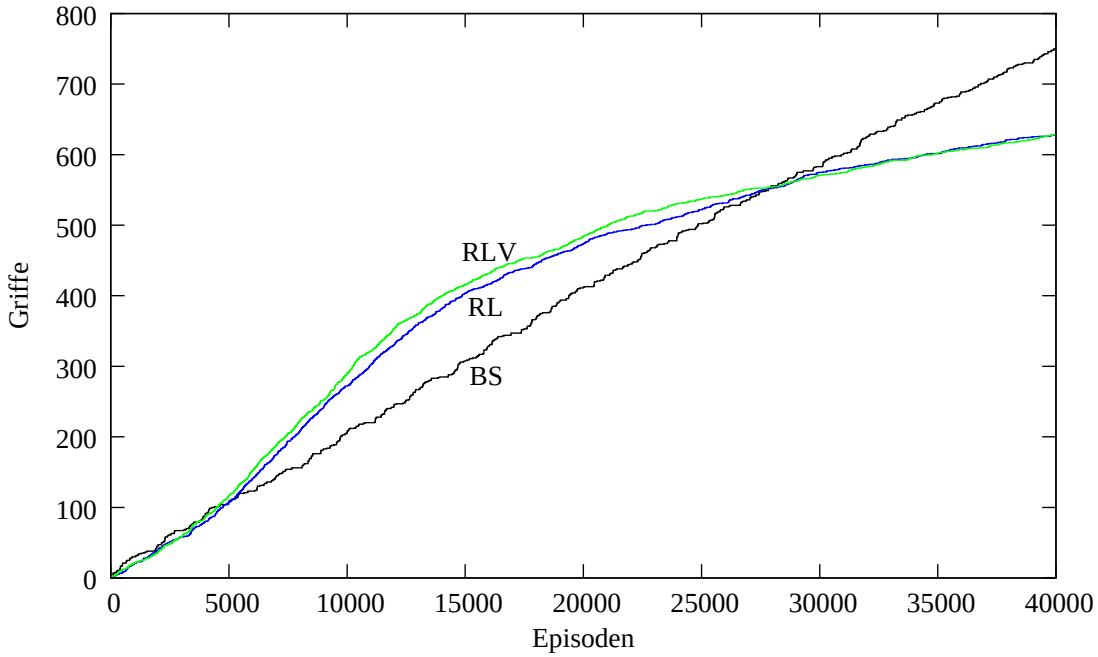
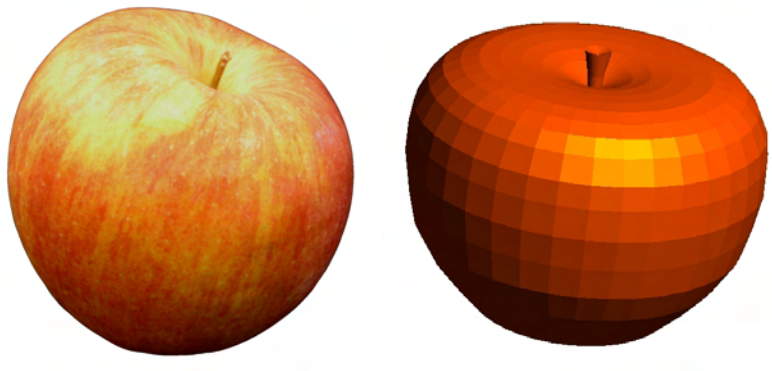


Abbildung 6.40: Ergebnisse der Griffgenerierung für einen Plastikball mit einem Durchmesser von 70 mm.

6.8.4.2 Greifen eines Apfels

Ein Apfel entspricht nicht der Form einer idealen Kugel, die Differenz zwischen beiden Formen ist aber in der Regel nicht sehr groß. In Abbildung 6.41 ist ein Apfel und das entsprechende Modell aus der

Simulation abgebildet. Für die Simulation wurde das Gewicht des Apfels mit 180 g definiert. Da der Apfel durch einen Griff nicht zerdrückt werden soll, beträgt die maximal zulässige Kraft 10 N.



(a) realer Apfel (b) simulierter Apfel

Abbildung 6.41: Ein Apfel und ein entsprechendes Modell aus der Simulation.

Die Ergebnisse der Griffberechnung für das Experiment mit dem Apfel sind in Abbildung 6.42 dargestellt. Wie aus der Grafik ersichtlich wird, sind beide Reinforcement-Lerner deutlich besser als die Breitensuche. Der trainierte Lerner ist im Durchschnitt 162,8% (350.0 Griffe) und der der untrainierte Lerner 38,2% (95,1 Griffe) besser als die Breitensuche. In diesem Fall ist der trainierte Lerner auch deutlich besser als der untrainierte. Durchschnittlich ist der trainierte Lerner 85,33% (246,4 Griffe) besser, als der untrainierte Lerner. Im Vergleich zur Griffberechnung für den Ball ist die Breitensuche für den Apfel deutlich schlechter. Der Ball ist symmetrisch, so dass es keinen signifikaten Unterschied macht, an welcher Stelle der Ball gegriffen wird. Der Apfel weist dagegen eine Asymmetrie auf, so dass es im Gegensatz zum Ball einen deutlichen Unterschied in den Positionen der Griffe gibt.

6.8.5 Greifen verschiedener Objekte

Es ist auch möglich, Griffe für Objekte zu lernen, die sich nicht direkt von einer geometrischen Grundform ableiten lassen. Als typische Beispiele für derartige, komplexe Objekte wurden für die Experimente eine Banane und ein Akkuschrauber gewählt. Diese Objekte lassen sich auch durch eine Aggregation von geometrischen Formen beschreiben, für die jeweils ein eigener Lerner trainiert werden könnte. So könnten z.B nach dem in [Miller et al. 2003] beschriebenen Verfahren, Griffe generiert werden. Hierzu wäre allerdings eine Beschränkung des Lernens auf die unterschiedlichen Objektteile erforderlich. Dieser Ansatz wurde in der vorliegenden Arbeit nicht verfolgt, sondern es wurde ein Lerner benutzt, um Griffe für das gesamte Objekt zu generieren.

6.8.5.1 Greifen einer Banane

Die Form einer Banane kann nicht, bzw. nur sehr schwer, von einer der geometrischen Grundformen (Kugel, Zylinder, Quader) abgeleitet werden. Ist die Krümmung der Banane sehr gering, ähnelt sie in ihrer Form zwar der eines Zylinders, der Unterschied zwischen beiden Formen ist aber immer noch zu

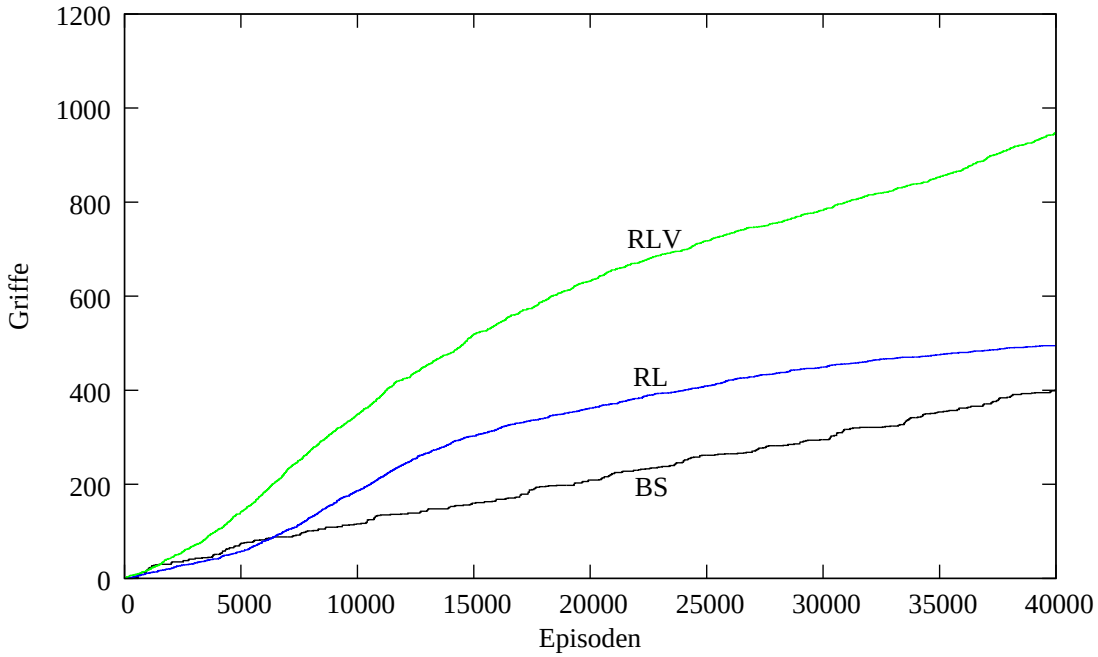
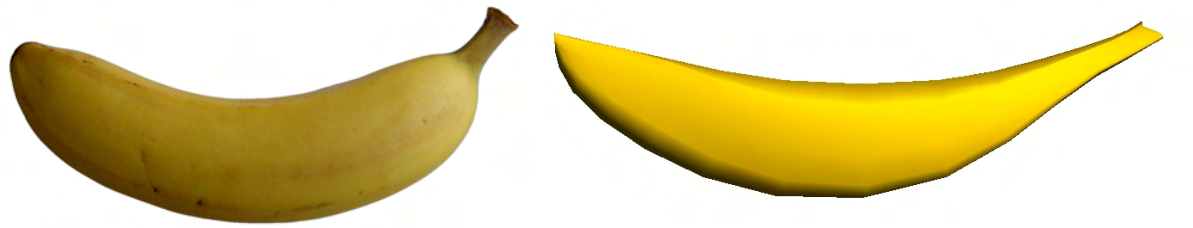


Abbildung 6.42: Ergebnisse der Griffgenerierung für einen Apfel.

groß, um sie direkt voneinander ableiten zu können. Eine Banane und ein entsprechendes Modell (mit 512 Polygonen) sind in Abbildung 6.43 dargestellt.



(a) reale Banane (b) simulierte Banane

Abbildung 6.43: Eine Banane und ein entsprechendes Modell aus der Simulation.

Die Ergebnisse der Griffberechnung sind in Abbildung 6.44 dargestellt. Es ist deutlich zu sehen, dass das Lernverfahren bei der Generierung von Griffen von der Seite der Breitensuche überlegen ist. Zwar ist die Breitensuche in zwischen Episode 0 und 9.298 besser (um durchschnittlich 670% oder 115,75 Griffe) als das Lernverfahren, im Mittel ist aber das Lernverfahren mit 212%, das entspricht 1.037,02 Griffen, besser als die Breitensuche (über 40.000 Episoden).

Obwohl sich die Formen nicht voneinander ableiten lassen, wurde trotzdem versucht, mit einem Lernverfahren, das zuvor zum Greifen eines Zylinders verwendet worden ist, Griffe für die Banane zu generieren. Wie in Abbildung 6.44 zu sehen ist, ergibt sich nur eine sehr geringe Abweichung zwischen dem trainierten und dem untrainierten Lerner. Die Abweichung beträgt nur 2,68% zwischen dem trainierten und dem untrainierten Lerner.

Es ist zwar sehr unwahrscheinlich, dass eine Banane so auf einem Tisch steht, dass sie mit einem Griff von oben aufgenommen werden kann, aber Griffe von oben sind z.B. für Übergabeoperationen durchaus geeignet. Bei der Generierung von Griffen von oben ist die Breitensuche dem Lernverfahren überlegen. Durchschnittlich ist hierbei die Breitensuche 113,67 Griffe, oder 260,76% besser als das Lernverfahren. Der Unterschied beträgt maximal 247 Griffe zwischen beiden Verfahren.

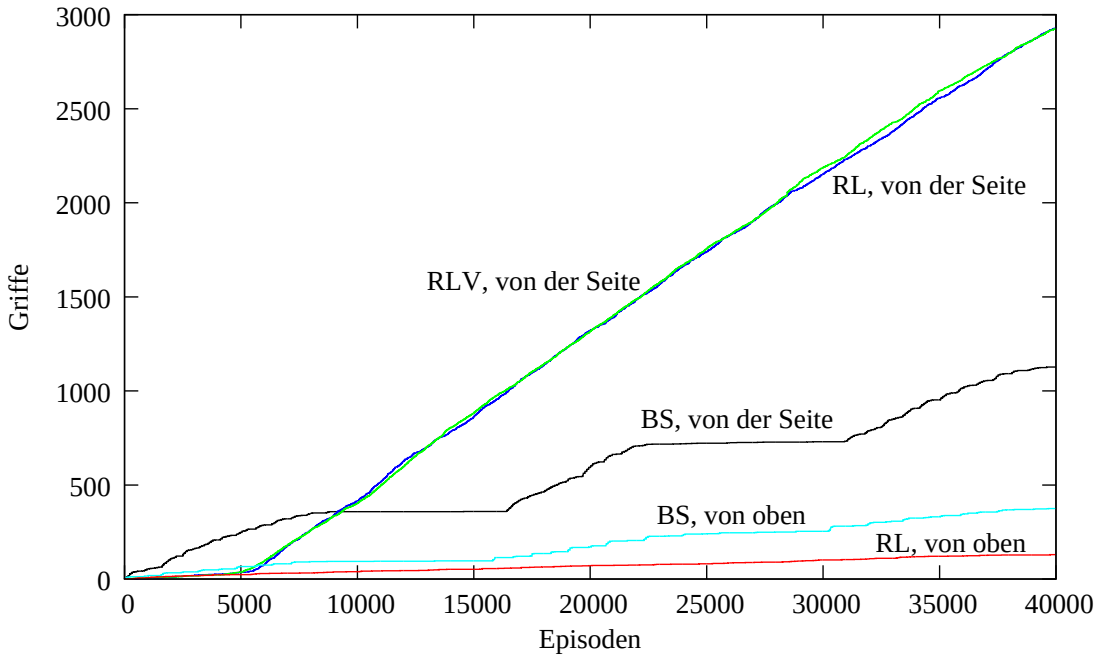


Abbildung 6.44: Ergebnisse der Griffgenerierung für eine Banane.

6.8.5.2 Greifen eines Akkuschraubers

Ein Akkuschrauber ist im Vergleich zu den bisher untersuchten Objekten ein relativ großes Objekt. Er ist mit einem Gewicht von ca. 3 kg deutlich schwerer (größer) als z.B. der Schokoriegel (50 g). Durch diese Eigenschaften ist ein Akkuschrauber auch für ein menschliche Hände unhandlich. Da nicht der gesamte Akkuschrauber für die Berechnung relevant ist, sondern nur der Griff und das Motorengehäuse, wurde auch nur dieser Teil in der Simulation betrachtet (Abb. 6.45(b)).

Für den Akkuschrauber wurden Griffe von der Seite und Pistolengriffe betrachtet. Die Ergebnisse der Griffgenerierung sind in Abbildung 6.46 zusammengefasst. Die Graphen für die Generierung der Griffe von der Seite verlaufen analog zu den Graphen der Griffberechnung für die Banane von der Seite. Zu Beginn der Berechnungen ist das Verfahren auf Basis der Breitensuche besser als das Lernverfahren. Insgesamt ist aber das Lernverfahren im Durchschnitt um 117,17%, oder 18,85 Griffe besser als die Breitensuche.

Bei der Berechnung der Pistolengriffe ist das Lernverfahren im Durchschnitt 48,2% oder 34,3 Griffe besser als die Breitensuche.

(a) realer Akkuschrauber (b) Modell eines Akkuschraubers

Abbildung 6.45: Ein Akkuschraubendreher und ein entsprechendes Modell aus der Simulation.

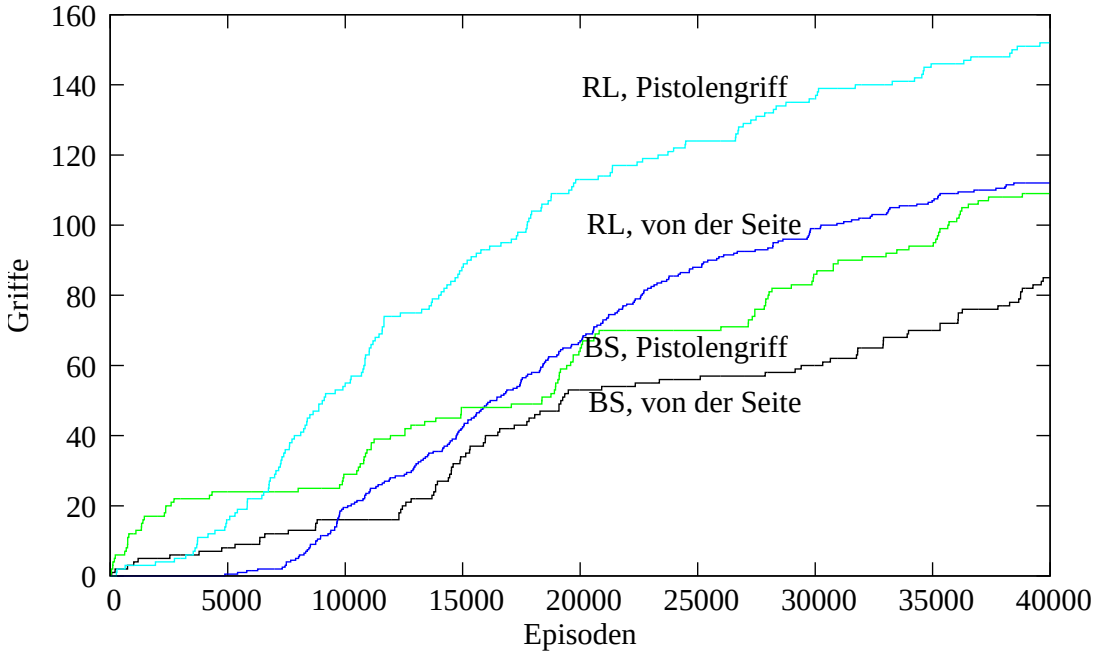


Abbildung 6.46: Ergebnisse der Griffgenerierung für einen Akkuschrauber.

6.9 Auswertung der Experimente

Die durchgeführten Experimente zeigen, dass die Generierung von Griffen mittels Reinforcement-Lernen möglich ist, und dass für jedes Objekt eine Vielzahl von Griffen gefunden werden kann. Insgesamt wurden über 1.000.000 Episoden und mehr als 16.000 Griffe durch das Lernverfahrens berechnet. Die Effizienz des Lernens ist in einigen Fällen schlechter als die des Breitensuche-Ansatzes, insgesamt betrachtet aber deutlich besser. Statistisch betrachtet ist das Lernverfahren im Durchschnitt über alle betrachteten Objekte um 148,8% besser als die Breitensuche. Dies entspricht im Mittel 167,1 Griffen. Im Vergleich zwischen dem trainierten und untrainierten Lernverfahren ist der trainierte Lerner durchschnittlich in der Lage, 122,7% mehr Griffe (bzw. 68,7 Griffe) mehr zu generieren. Dies zeigt, dass sich durch einen bereits trainierten Lerner ein deutlicher Effizienzgewinn erreichen lässt. Da der Lerner nur einmal für eine Objektform trainert werden muss, ist der Aufwand hierfür zu vernchlässigen.

Im Vergleich zu den in Abschnitt 6.1 vorgestellten Ansätzen für die Generierung von Griffen hebt sich das entwickelte Verfahren besonders dadurch ab, dass sowohl Power- als auch Präzisionsgriffe berechnet werden können.

Ein Vergleich der Performanz der vorgestellten Verfahren ist schwierig. So geben z.B. die Autoren in [Miller et al. 2003] an, dass mit dem beschriebenen Verfahren 68 Griffe in 248 Sekunden für eine Tasse getestet worden sind. Ergebnisse beziehen sich dabei auf Experimente, die mit einem Pentium IV 1 GHz durchgeführt worden sind. Mit dem in dieser Arbeit entwickelten Verfahren können mit einem AMD Athlon 64 X2 Dual Core 3800+ (2,0GHz) für ein ähnliches Modell in der Zeit von 248 Sekunden rund 12 Griffe berechnet werden. Die Performanz in Bezug auf die reine Rechenleistung ist bei dem entwickelten Verfahren also schlechter. Die liegt daran, dass das entwickelte Verfahren mit der Anzahl der Polygone, die für ein Objekt verwendet werden, skaliert. Können im Durchschnitt noch 162,5 Griffe in 60 Sekunden für einen einfachen Quader mit 12 Polygonen berechnet werden, so sind es bei einem komplexen Objekt wie z.B. dem Griff des Akkuschraubers mit 5.161 Polygonen noch rund 14,6 Griffe in 60 Sekunden. Wie sich in den Experimenten gezeigt hat, ist die Kollisionserkennung die Komponente des Algorithmus, welche den größten Anteil der Rechenzeit verbraucht. Es ist aber möglich, diese Berechnung auf den Prozessor der Grafikkarte auszulagern und somit zu beschleunigen. Wie aus den Beispielen zu erkennen ist, ist ein Vergleich der Performanz eines Algorithmus in Bezug auf die Rechenzeit nicht sehr aussagekräftig. Zum einen werden nach Moores Gesetz [Moore 1965] in Zukunft immer schnellere Prozessoren die Rechenzeit immer weiter verkürzen und zum anderen werden die Griffe in den vorgestellten Verfahren in einer Simulationsumgebung berechnet. Somit können die Griffe offline generiert werden, und der Roboter kann bei der Griffauswahl auf eine entsprechende Datenbank, in der die Griffe gespeichert sind, zugreifen. Hierdurch ist eine on-line Berechnung der Griffe nicht erforderlich und die Berechnungszeit irrelevant.

Entscheidend ist, dass für jedes Objekte jede Art von Griff mit einem einzigen Verfahren generiert werden kann. Dies leistet das entwickelte Verfahren.

Erweiterbarkeit

Bei den durchgeführten Experimenten wurde von der Annahme ausgegangen, dass es eine Vorgabe gibt, von welcher Seite das Objekt gegriffen werden soll, also von oben, von vorne oder von der Seite, etc. Möchte man nun automatische Griffe von allen Seiten mit allen mögliche Orientierungen und Spreizwinkeln generieren lassen, so muss die Aktionsmenge um weitere Aktionen ergänzt werden. Die Rotationen um die X- und die Z-Achse des Objekts müssen zu den Aktionen aus Aktionsmenge 2 hinzugefügt wer-

den. Die daraus resultierende *Aktionsmenge 3* ist in Tabelle 6.6 dargestellt. Da der Suchraum mit dem

Aktion	initiale Gewichtung
Translation ±X,±Z	0.0
Translation -Y	0.25
Rotation $\pm X$	0.0
Rotation $\pm Y$	0.0
Rotation $\pm Z$	0.0
Spread 0.0	0.0
Spread 20.0	0.0
Spread 45.0	0.0
Spread 90.0	0.0
Spread 180.0	0.0
Griff	0.0

Tabelle 6.6: Aktionsmenge 3 für Experimente

Hinzufügen einer neuen Dimension entsprechend wächst, ist nun eine deutlich längere Lernphase erforderlich, um in jeder Orientierung ausreichend viele Griffe zu generieren. Hierdurch ist die Performanz geringfügig schlechter, wenn man die absolute Anzahl der gefundenen Griffe betrachtet, als die Griffberechnung mit nur einer Orientierung.

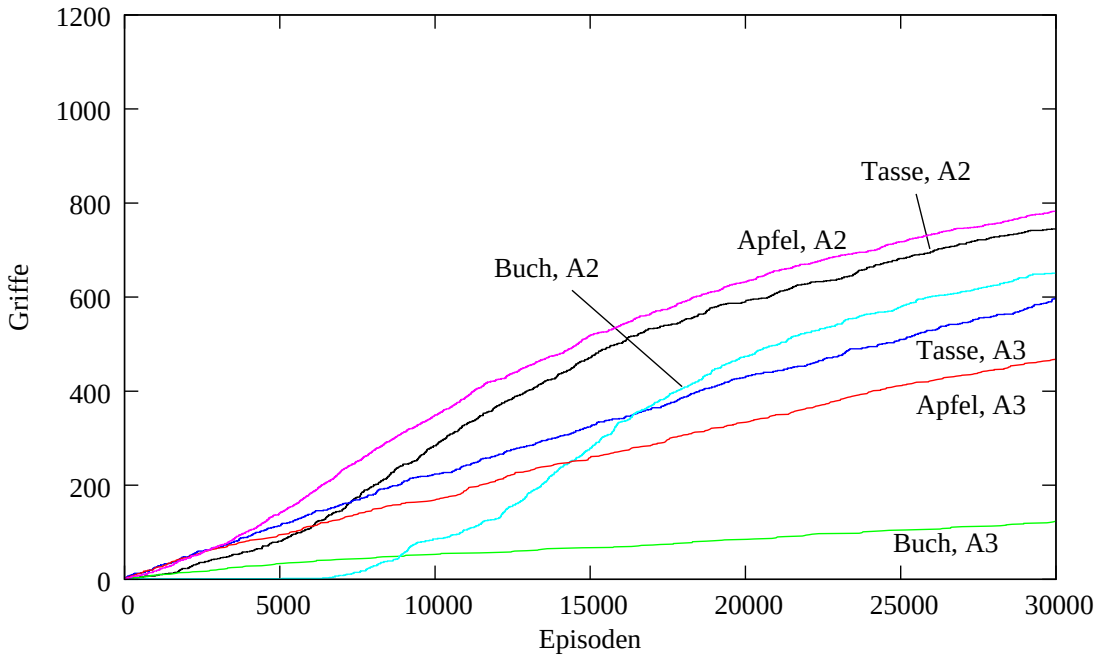


Abbildung 6.47: Ergebnisse der Griffgenerierung ohne Orientierungsbeschränkung.

Die Ergebnisse der Griffgenerierung mit Aktionsmenge 3 sind in Abbildung 6.47 dargestellt. Bei den durchgeführten Experimenten wurden exemplarische Griffe für ein Buch und einen Apfel über 40.000 Episoden generiert und die Ergebnisse denen der Griffgenerierung mit Aktionsmenge 2 gegenüberge-

stellt. Bei dem Buch entsprechen die Griffe der Aktionsmenge 2 einem Griff von vorne. Wie zu sehen ist, werden mit Aktionsmenge 3 in der gleichen Anzahl von Episoden weniger Griffe gefunden als mit Aktionsmenge 3.

Kapitel 7

Greifen von Alltagsgegenständen

In diesem Kapitel wird das entwickelte Qualitätsmaß für Griffe (Kapitel 4) mit dem Service-Roboter TASER (Kapitel 2) eingesetzt, um eine Auswahl der in Kapitel 6 berechneten Griffe zu validieren. In Abschnitt 7.1 wird zunächst der Systemaufbau beschrieben. Hierbei wird auf die verwendete Objekterkennung und die Griffauswahl und -ausführung eingegangen. In Abschnitt 7.2 werden die Experimente vorgestellt, die mit dem Service-Roboter TASER durchgeführt worden sind. Hierfür ist eine Auswahl von Griffen aus der Simulation mit TASER evaluiert worden. Abschließend erfolgt eine Bewertung des Systems zum Greifen von Alltagsgegenständen.

7.1 Systemaufbau

Das gesamte System zum Greifen von Alltagsgegenständen ist in einer objektzentierten Sichtweise aufgebaut. Als ein Objekt wird dabei ein greifbarer Gegenstand aufgefasst. Die Modellierung enthält dabei alle relevanten Daten, wie die Oberflächenbeschaffenheit, das Gewicht, das geometrische Modell, etc. (siehe auch Kapitel 6.2.2.1). Auch die Griffe werden objektbezogen generiert, wobei eine Generalisierung zwischen Objekten mit ähnlicher Form möglich ist, wie die Ergebnisse der Simulation aus Kapitel 6.8 zeigen.

Das Gesamtsystem setzt sich, neben der Griffgenerierung, die in Kapitel 6.2.2 vorgestellt wurde, aus den Teilen Objekterkennung, Griffauswahl und Griffausführung zusammen. Hierbei beinhaltet die Objekterkennung auch eine Positionserkennung. Eine Übersicht über die Systemkomponenten ist in Abbildung 7.1 dargestellt. Die einzelnen Teile des Systems werden im Folgenden kurz vorgestellt.

7.1.1 Objekterkennung

Wenn ein Service-Roboter in der Lage sein soll, autonom Objekte auf der Basis von zuvor in einer Simulation berechneten Griffen zu greifen, ist eine Objekterkennung obligatorisch. Eine Alternative wäre die on-line Berechnung von Griffen. Da dies in der Regel ein aufwendiger Prozess und die Griffe jedes mal erneut berechnet werden müssten, ist diese Vorgehensweise höchst ineffizient. Der Roboter muss also in der Lage sein Objekte wieder zu erkennen, um einen bereits berechneten Griff für das Objekt auswählen und anwenden zu können.

Für die Objekterkennung wird ein Verfahren aus dem Bereich des unüberwachten, maschinellen Lernens eingesetzt, die Hauptkomponentenanalyse (*Prinicipal Component Analysis, PCA*). Bei einer PCA wird versucht, die Bilder in einen Subraum zu transformieren, im dem sich die Bilder von einander unterscheiden lassen. Als Grundlage für die Transformation dient dabei die Varianz zwischen den Bildern. Mittels einer Eigenvektor Berechnung wird ein mehrdimensionaler Subraum aufgespannt, in dem sich

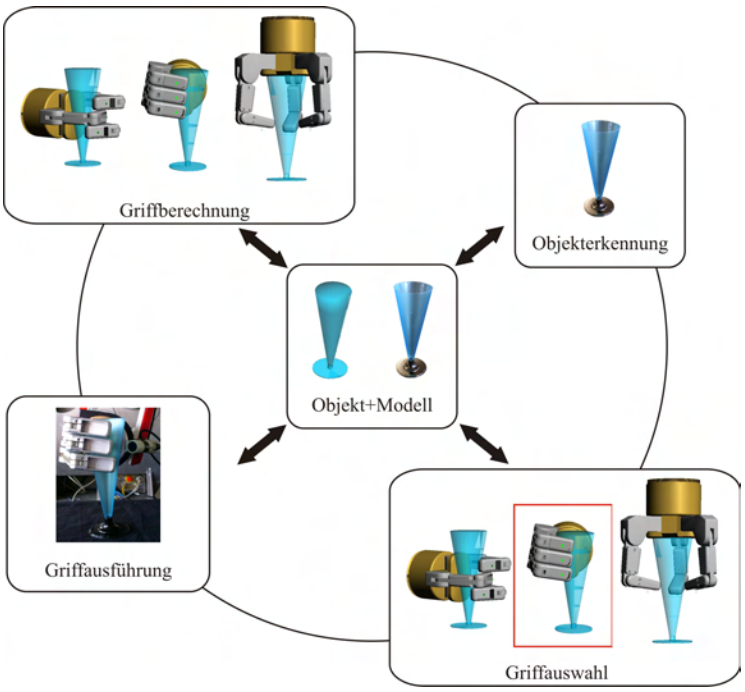


Abbildung 7.1: Die Systemkomponenten des Systems zum Greifen von Alltagsgegenständen. Das System ist in einer objektzentierten Sichtweise aufgebaut.

die Achsen entlang der maximalen Varianzen ausrichten. Wenn die Varianzen der Bilder die entsprechenden Merkmalsunterschiede repräsentieren, können Objekte durch eine Projektion in den Subraum klassifiziert werden. Detailliert ist das Verfahren der PCA in Anhang C beschrieben.

In dieser Arbeit wurde das oben beschriebene Verfahren auf die 12 Objekte angewandt, die für Generierung von Griffen verwendet worden sind. Dabei konnte eine Erkennungsrate von 90% erreicht werden. Hierfür sind für jedes Objekt fünf Testbilder aufgenommen. Von jedem dieser Bildern wurden 90 Rotationsbilder mit unterschiedlichen Orientierungen berechnet. Somit ist das System mit insgesamt 5.400 Testbildern evaluiert worden.

7.1.2 Griffauswahl

Die Auswahl eines Griffs erfolgt unter Berücksichtigung verschiedener Kriterien. Ist eine Qualitätsanforderung in Bezug auf die Weiterverwendbarkeit des Objekts vorgegeben, so wird eine entsprechende Rangordnung der, für das Objekt zur Verfügung stehenden Griffe erstellt. Anschließend werden die Griffe bezüglich ihrer Umsetzbarkeit evaluiert. Hierbei wird geprüft, ob die entsprechende Position für die Ausführung eines Griffs von dem Roboterarm eingenommen werden kann. Die Position des Service-Roboter TASER ist dabei statisch. Das heißt es findet keine Bewegung der mobilen Plattform statt, falls ein Griff aufgrund kinematischer Probleme nicht ausführbar ist.

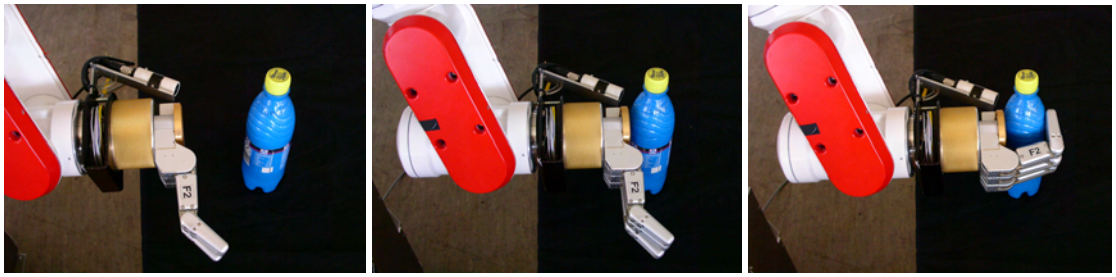
Kann der Roboterarm die Position für den Griff nicht erreichen, so wird der in der Rangordnung folgende Griff auf seine Anwendbarkeit überprüft. Dies erfolgt so lange, bis entweder ein Griff gefunden wurde, der den Qualitätsanforderungen genügt und ausführbar ist, oder ohne einen Griff auszuführen terminiert

das System. Im zweiten Fall wäre es denkbar, dass der Roboter das Objekt mit einem Griff aufnimmt, der den Anforderungen nicht genügt und anschließend umgreift. Alternativ bewegt sich die gesamte Plattform in eine Position, von der aus die Ausführung des Griffs möglich ist. Dies ist allerdings eine Erweiterung, die in dieser Arbeit nicht mehr realisiert wurde und als Ergänzung vorgesehen ist (siehe auch Kapitel 8).

7.1.3 Ausführung von Griffen

Wird ein adäquater Griff gefunden, wird der Roboterarm bis auf eine Annäherungsposition an den Gegenstand bewegt. Diese Position entspricht in ihrer Orientierung bereits der finalen Position, lediglich die Translation entlang des Näherungsvektors weist noch einen Versatz auf. Abbildung 7.2 zeigt die Annäherungsposition und die Position, in der der Griff ausgeführt wird. Von der Annäherungsposition erfolgt eine langsame, kraftgeregelte Bewegung bis zur finalen Position. Anschließend wird der Griff mit der BarrettHand ausgeführt und der Arm in eine Parkposition gefahren, die dynamisch berechnet wird. Der Service-Roboter TASER wird während des gesamten Ablaufs nicht bewegt.

Abbildung 7.3 stellt noch einmal den Ablauf eines Greifvorgangs dar.



(a) Annäherungsposition (b) Griffposition (c) Griff

Abbildung 7.2: Die Annäherung der Hand an das Objekt.

7.2 Experimente und Ergebnisse

In den folgenden Abschnitten wird das Greifen von Alltagsgegenständen mit dem Service-Roboter TASER unter realen Bedingungen analysiert werden. Hierbei werden die aus der Simulation (siehe Kapitel 6.8) gewonnenen Griffe mit dem realen Robotersystem validiert. Da es nicht praktikabel ist, alle der mehr als 16.000 generierten Griffe auszuprobieren, wurde für jedes Objekt eine Auswahl von Griffen getestet. Die Auswahl mittels den in Kapitel 4.2 definierten Qualitätsmaßen.

Dabei werden verschiedene Anforderungen an die Griffe gestellt. Diese sind:

- Der Griff soll möglichst stabil sein.
- Der Griff soll möglichst viele Übergabeoperationen ermöglichen.
- Keine der definierten Operation wird bevorzugt.

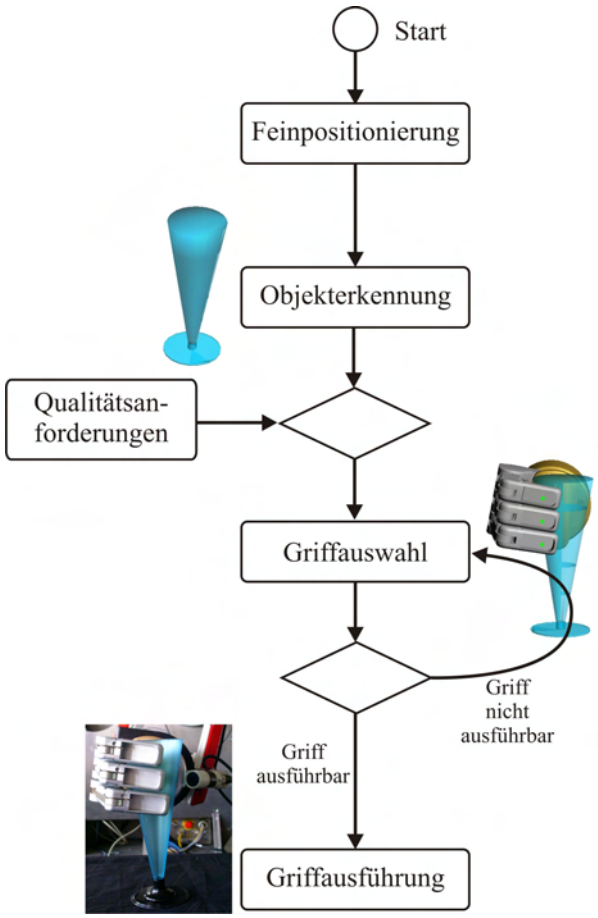


Abbildung 7.3: Prozess der Griffausführung.

Handelt es sich bei dem Objekt um einen Container, so wird ein zusätzlicher Griff ausgewählt der ein Einfüllen bzw. Ausgießen ermöglicht. Wurden für ein Objekt Griffe generiert, die sich durch eine besondere Griffgeometrie auszeichnen, so wurden diese ebenfalls evaluiert. Die Definition der Qualitätsmaße erfolgt analog zu den Definitionen aus Tabelle 4.7. Ist es aufgrund der Griffkonfiguration nicht möglich, das Objekt direkt vom Tisch aufzunehmen, so wird es dem Roboter in entsprechender Position übergeben.

Leider ist die Sensorik der BarrettHand mangelhaft, so dass nur eine quantitative Aussage über die Anwendbarkeit der Griffe getroffen werden kann. Eine exakte Auswertung, ob die auftretenden Kräfte denen entsprechen, die in der Simulation berechnet worden sind, ist leider nicht möglich. Hierfür wären deutlich mehr Sensoren an den Fingern und Gliedmaßen der BarrettHand erforderlich. Die Griffkraft wird bei den Experimenten aber jeweils so eingestellt, dass sie den Kraftwerten entspricht, die in der Simulation berechnet worden sind. Um die Griffe trotzdem qualitativ bewerten zu können, wird jeder der Griffe von dem Roboter drei mal ausgeführt. Gelingt der Griff dabei mindestens zweimal, so wird die Annahme getroffen, dass die Ergebnisse der Simulation korrekt sind und sich auf die Realität übertragen lassen. Andernfalls wird der Griff als unbrauchbar zurückgewiesen. Hierbei wurde für Objekte, bei denen die Operation „Ausgießen“ möglich sein soll, eine entsprechende Bewegung ausgeführt. Sie

wurde mittels einer Rotation des sechsten Gelenks des Roboterarms realisiert. Momentaufnahmen einer solchen Bewegung sind in Abbildung 7.4 auf Seite 147 festgehalten.

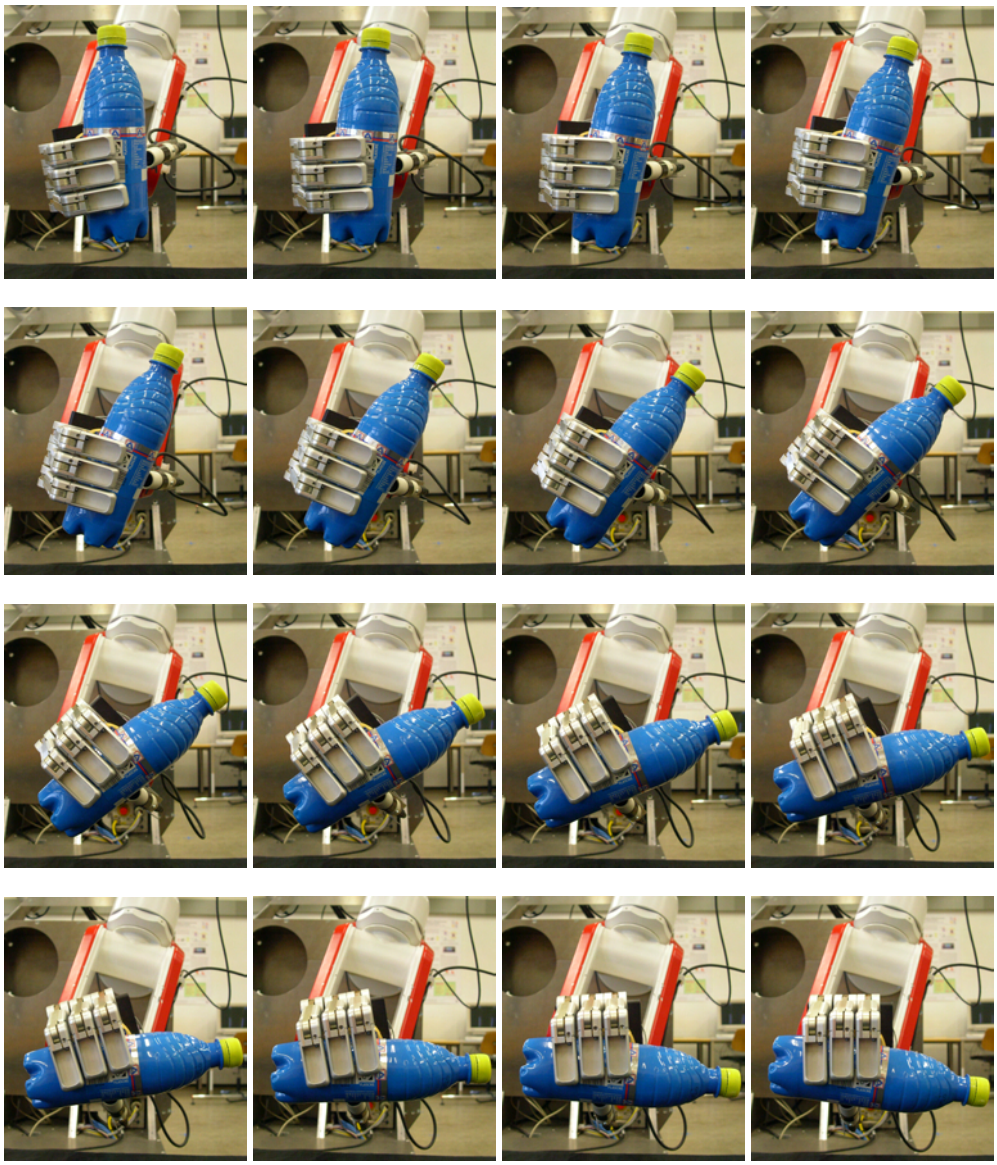


Abbildung 7.4: Ausführung einer Ausgießbewegung.

7.2.1 Griffe für einen Schokoriegel

Ein handelsüblicher Schokoriegel, mit einem Gewicht von ca. 60 g und Ausmaßen von ungefähr 20 mm $\times$ 30 mm $\times$ 100 mm, ist ein relativ kleines Objekt. In der Simulation gelang es, viele stabile Griffe für ein solches Objekt zu finden. Bei der Durchführung der Experimente hat sich jedoch gezeigt, dass keiner der Griffe geeignet ist, um den getesteten Schokoriegel aufzunehmen. Die evaluierten Griffe, sind in der oberen Zeile von Abbildung 7.5 dargestellt. Die untere Zeile der Abbildung zeigt die Ausführung der

Griffe mit dem Service-Roboter TASER. Jeder der fünf dargestellten Griffe ist ausführbar und stabil.

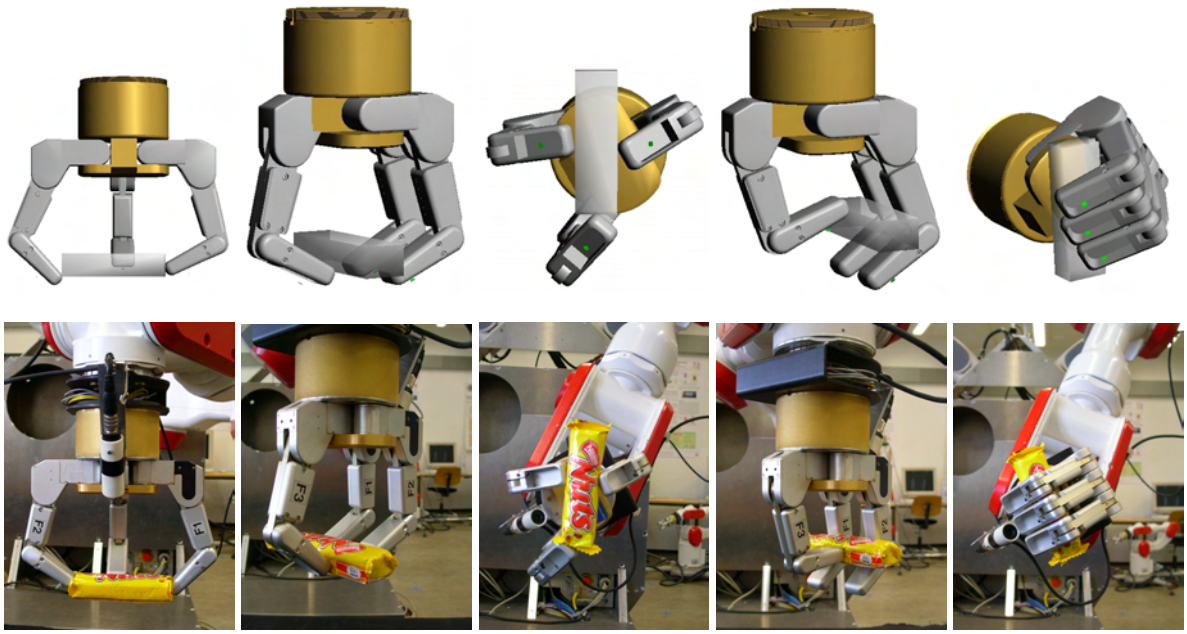


Abbildung 7.5: Griffe für einen Schokoriegel.

Allerdings eignet sich keiner der Griffe, um den Schokoriegel vom Tisch aufzunehmen. Zwar erwecken die evaluierten Präzisionsgriffe (Griff 1-3) den Anschein, dass dies Aktion möglich ist, der tiefste Punkt der Kreisbahn, auf der sich die Fingerspitze beim Schließen bewegt, liegt jedoch tiefer als die Greifpunkte an dem Objekt (siehe Abb. 7.6), so dass diese Griffe nicht ausführbar sind. Die bisher implementierte Steuersoftware erlaubt es nicht, die Handposition während der Fingerbewegung zu verändern. Somit ist es nicht möglich, das Objekt an den berechneten Kontaktpunkten aufzunehmen.

Entsprechend den gestellten Qualitätsanforderungen bieten die Griffe 1 und 2 die beste Stabilität. Griff 3 bietet die meisten Möglichkeiten für eine Übergabe und Griff 4 ist der „Universalgriff“, der sich aus der Rangordnung der Griffe ergibt, wenn keine Operation bevorzugt wird. Der letzte Griff zeichnet sich durch seine besondere Geometrie aus.

7.2.2 Griffe für ein Buch

Für ein Buch der Größe 35 mm $\times$ 180 mm $\times$ 240 mm und einem Gewicht von ca. 2 kg wurden in der Simulation 35 verschiedene Griffe von der Seite und 287 Griffe von vorne berechnet. Bei der Evaluation der Griffe zeigte sich, dass die Differenz zwischen dem Modell und dem real existierenden Buch zu groß ist. Viele der Griffe lassen sich nur unter besonderen Bedingungen bzw. unter besonderer Berücksichtigung der Orientierung umsetzen. Die evaluierten Griffe, sind in Abbildung 7.7 dargestellt. Griff 1 und 2 sind die Griffe, die die beste Stabilität bieten. Griff 2 erhält die beste Bewertung, wenn keine Operation bevorzugt wird. Griff 3 ist der Griff, der die meisten Übergabemöglichkeiten bietet.

Der letzte Griff aus Abbildung 7.7 ist nur in den gezeigten Konfigurationen stabil. Ändert man die Orientierung, so ist die Haftreibung nicht mehr ausreichend, um das Buch stabil zu halten, und es rutscht,

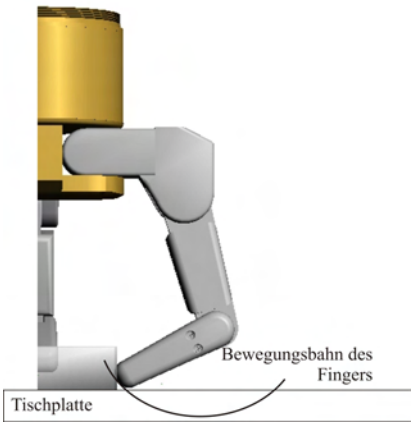


Abbildung 7.6: Bewegung eines Fingers.

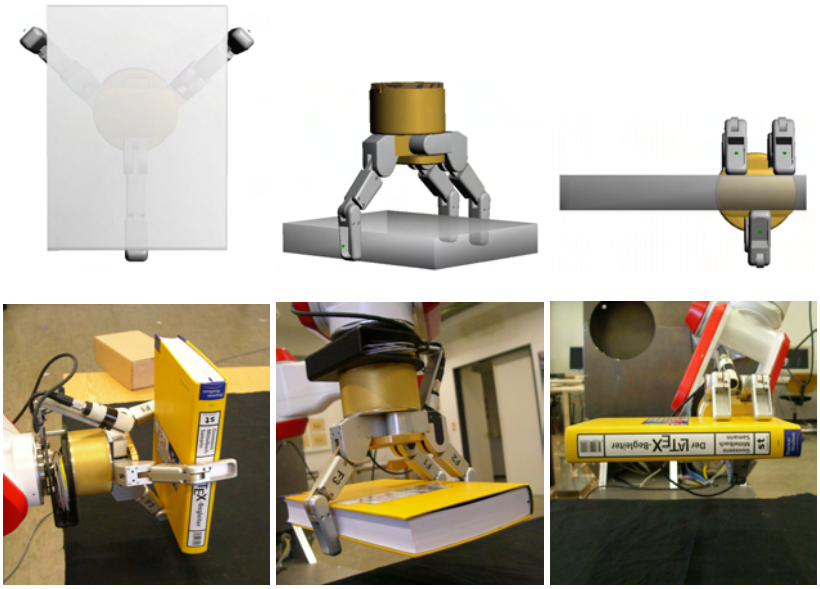
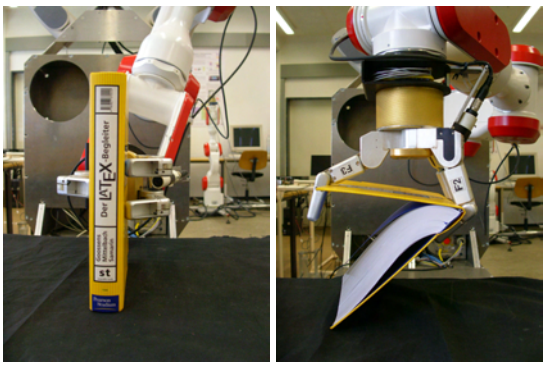


Abbildung 7.7: Griffe für ein Buch.

wie in Abbildung 7.8(a) illustriert, aus den Fingern der Hand. In einem zweiten Beispiel (Abb. 7.8(b)) ist ein Modellfehler die Ursache für die Instabilität. Die Finger umschließen die Buchdeckel nicht komplett und es klappt während des Anhebens auf. Da das Buch in der Simulation lediglich als Box mit entsprechender Größe simuliert worden ist, konnte dieses Verhalten nicht voraus berechnet werden. Aufgrund dieses Verhaltens sind nur ca. 60% aller für das Buch berechneten Griffe stabil.



(a) (b)

Abbildung 7.8: Instabile Griffe für ein Buch.

7.2.3 Griffe für einen Telefonhörer

Die Griffe für einen Telefonhörer wurden in der Simulation mit einem relativ groben Modell berechnet, das sich lediglich aus Quadern zusammen setzt. Wie in Abbildung 7.9 zu sehen ist, lassen sich die Griffe dennoch auf das reale Objekt übertragen.

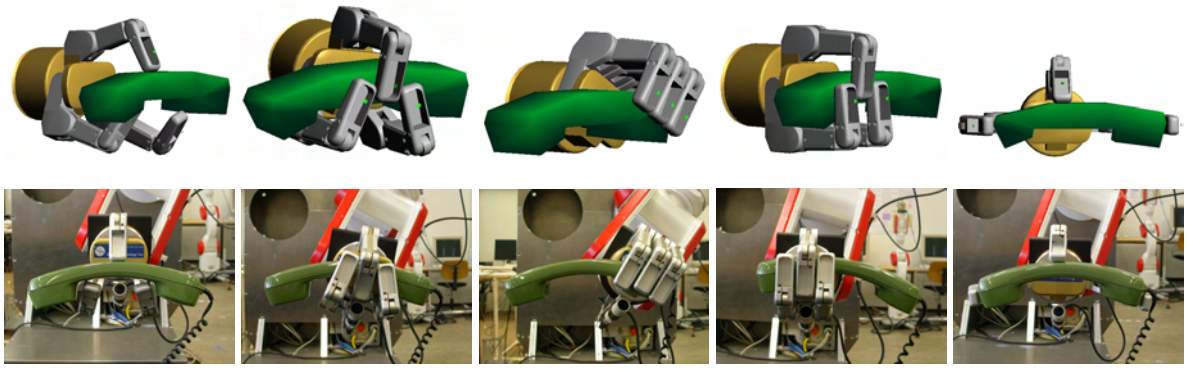


Abbildung 7.9: Griffe für einen Telefonhörer.

Die Griffe 1 und 2 sind die, mit der am besten bewerteten Stabilität und Griff 3 bietet die meisten Übergabemöglichkeiten. Griff 4 ist der „Universalgriff“, wenn keine Operation bevorzugt wird. Griff 5 zeichnet sich durch eine interessante Geometrie aus. In der Simulation wurden auch Griffe von oben für den Telefonhörer berechnet, wie in Abbildung 7.10 dargestellt. Von diesen Griffen ließ sich keiner in der Realität umsetzten. Hierfür ist ebenfalls ein Modellfehler verantwortlich. Das Modell, das für die Simulation verwendet worden ist, ist nur eine grobe Approximation des realen Telefonhörers. Im Gegensatz zu dem Modell, hat der reale Hörer abgerundete und keine scharfen Kanten. Somit ist die Berechnung der Kontaktnormalen, übertragen auf den realen Hörer fehlerhaft. In der Realität treffen immer zwei runde Kontaktflächen aufeinander. Werden nun die Kontaktnormalen nicht entsprechend berechnet, können die Kräfte nicht wie berechnet wirken und die Kontakte sind instabil. Daher können nur Griffe von der Seite mit dem realen Telefonhörer ausgeführt werden. Diese Griffe haben allerdings

den Nachteil, dass sie nicht geeignet sind den Telefonhörer aufzunehmen. Griff 3 erweckt zwar den Anschein, dass ein Aufheben möglich ist, jedoch kollidieren Teile der Handbasis bzw. des Roboterarms bei einem entsprechenden Versuch mit dem Tisch. Die Ergebnisse der Evaluation zeigen in diesem Fall, dass die Differenz zwischen dem Modell der Simulation und dem realen Objekt zu groß ist, um dieses mit Präzisionsgriffen aufnehmen zu können.

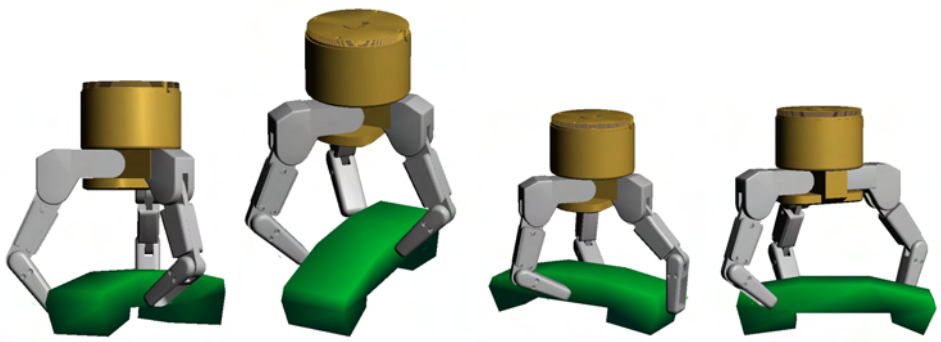


Abbildung 7.10: Nicht ausführbare Griffe für einen Telefonhörer.

7.2.4 Griffe für eine Flasche

Für eine 0,5 l PET-Flasche wurden in der Simulation 438 Griffe (156 von der Seite und 282 von oben) berechnet. Von diesen wurden fünf Griffe ausgewählt, die in Abbildung 7.11 dargestellt sind. Griff 1 entspricht dabei dem Griff mit den meisten Übergabemöglichkeiten, ist aber äußerst instabil. Daher bietet es sich in diesem Fall an, auf Griff 2 auszuweichen. Griff 3 ist der stabilste Griff, wenn die Flasche von oben gegriffen wird. Griff 4 ist insgesamt der stabilste Griff, der sich auch am besten für die Operationen Einfüllen und Ausgießen eignet. Griff 5 zeichnet sich durch seine besondere Geometrie aus.

Obwohl das Modell, das in der Simulation benutzt worden ist von dem realen Objekt abweicht, sind alle der evaluierten Griffe stabil. Dies gilt sowohl für eine leere als auch für eine gefüllte Flasche. Auch die berechneten optimalen Kontaktkräfte, die in Tabelle 7.1 zusammengefasst sind, liegen deutlich unter den zulässigen Maximalwerten von 20 N je Kontakt.

Kontaktpunkt	Basis	F1 distal	F2 distal	F3 distal
Griff 1	-	7,22 N	7,64 N	7,29 N
Griff 2	-	4,21 N	4,22 N	1,83 N
Griff 3	1,00 N	5,10 N	0,56 N	5,99 N
Griff 4	2,43 N	3,45 N	3,42 N	2,60 N
Griff 5	2,43 N	3,45 N	3,42 N	2,60 N

Tabelle 7.1: Kräfte entlang der Z-Achse der Kontaktpunkte zu den Griffen aus Abbildung 7.11.

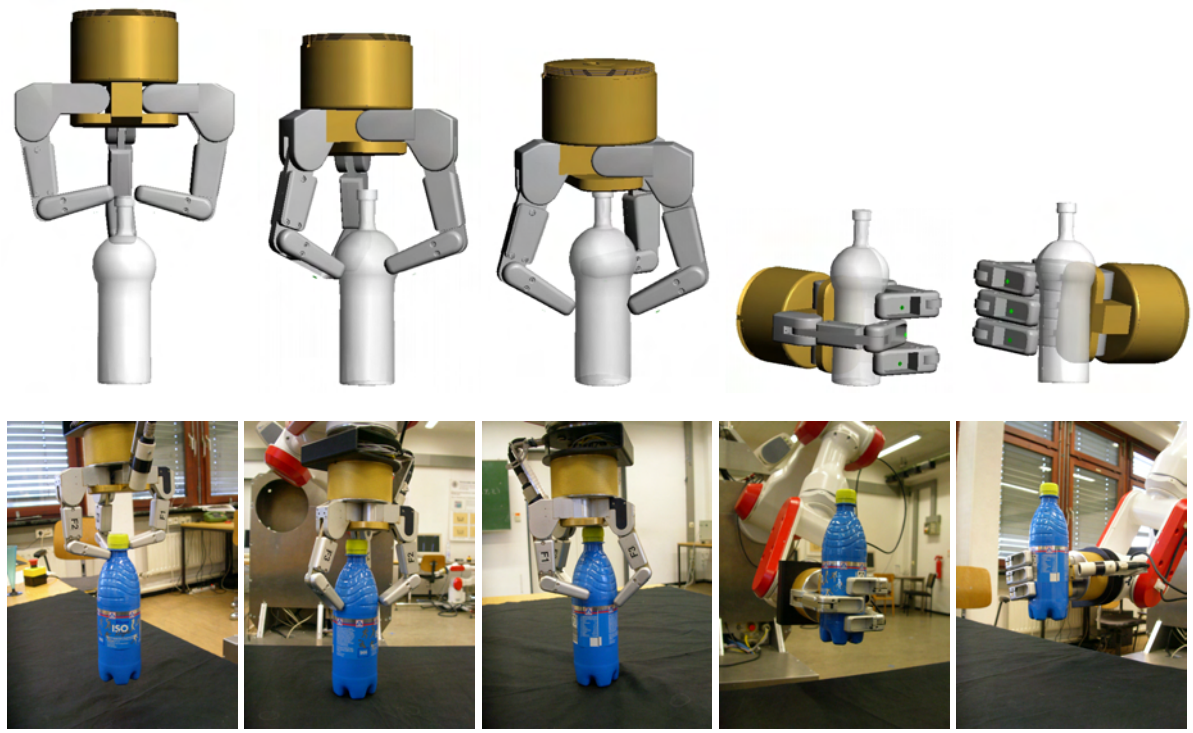


Abbildung 7.11: Griffe für eine 0,5 l Flasche.

7.2.5 Griffe für einen Becher

In Abbildung 7.12 sind fünf Griffe dargestellt, die mit dem realen Roboter umgesetzt werden konnten. Alle Griffe sind stabil und erfüllen die gestellten Anforderungen. Griff 1 ist der stabilste Griff, Griff 2 der mit den meisten Übergabemöglichkeiten und Griff 3 ist der am besten bewertete Griff, wenn alle Operationen gleich gewichtet werden. Mit Griff 4 können die Operationen Ein- und Ausgießen ausgeführt werden. Griff 5 zeichnet sich durch seine besondere Geometrie aus.

Allerdings wurden mittels der Simulation auch Griffe berechnet, die nur sehr schwer bzw. gar nicht mit der realen BarrettHand umgesetzt werden können. Beispiele für solche Griffe sind in Abbildung 7.13 dargestellt. Bei diesen Griffen liegt ein ähnliches Problem wie bei den Griffen für den Telefonhörer vor. Hierbei treffen ebenfalls zwei runde Kontaktflächen aufeinander. Zwar ist das Modell im Fall des Bechers deutlich genauer, aber trotzdem müssen die Kontaktpunkte genau getroffen werden. Schon eine geringe Abweichung von wenigen Millimetern führt dazu, dass die Kräfte nicht wie in der Simulation berechnet wirken und der Griff instabil wird.

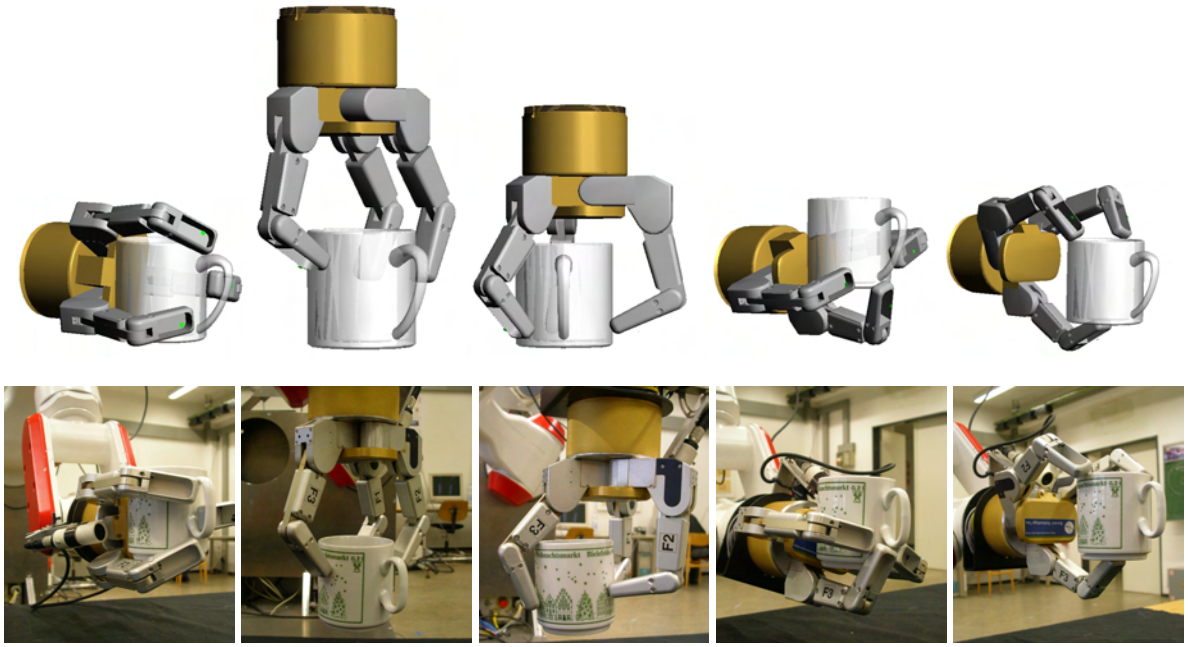


Abbildung 7.12: Griffe für einen 0,3 l Becher.

Abbildung 7.13: Nicht ausführbare Griffe für einen 0,3 l Becher.

7.2.6 Griffe für eine Tasse

Die Form einer Tasse ist mit der eines Bechers vergleichbar. Sie ist lediglich etwas bauchiger und hat gegenüber dem Becher einen abgerundeten Boden. Da das Modell in diesem Fall etwas größer als das reale Objekt ist, kann an diesem Beispiel die Robustheit des Verfahrens gezeigt werden. Sowohl beim Greifen des Bechers, als auch bei der Tasse gab es Schwierigkeiten. Trotzdem sind alle der berechneten Griffe umsetzbar und stabil. Fünf Beispiele hierfür sind in Abbildung 7.14 dargestellt. So sind sowohl die berechneten Präzisionsgriffe (siehe Abb. 7.14 Spalte 3–4) als auch die Powergriffe stabil und können ohne Weiteres eingesetzt werden.

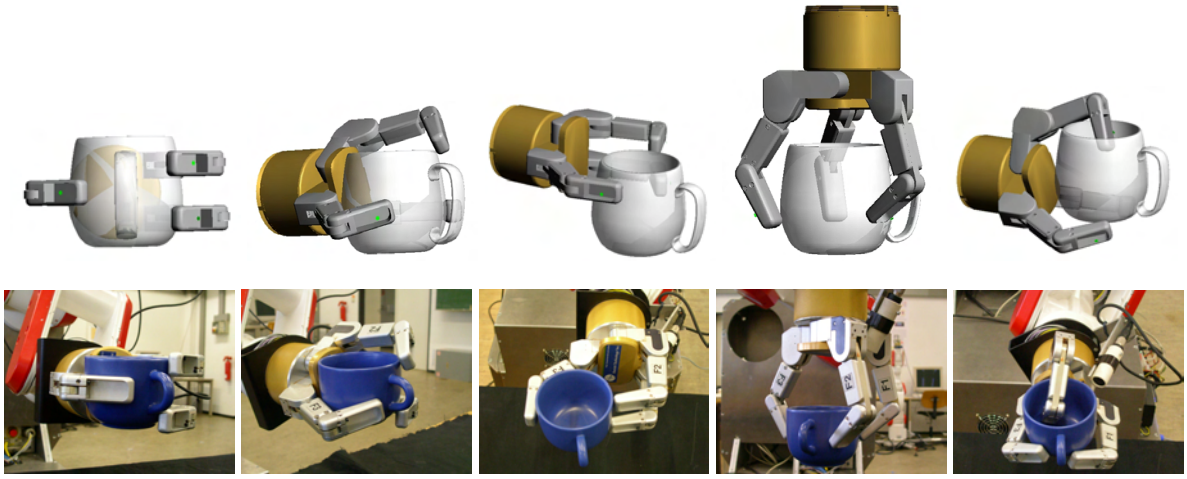


Abbildung 7.14: Griffe für eine Tasse.

7.2.7 Griffe für ein Sektglas

Das Sektglas ist das fragilste Objekt, das im Rahmen der Experimente mit der BarrettHand gegriffen wurde. Es ist mit einer Masse von 50 g ein sehr leichtes Objekt. Auf das Sektglas darf nur eine maximale Kraft von 7 N ausgeübt werden. Trotzdem war es möglich, mittels der Simulation eine Vielzahl von Griffen zu berechnen. Eine Auswahl von Griffen, die mit dem realen Robotersystem umgesetzt worden sind, sind in Abbildung 7.15 dargestellt. Das System war in der Lage, sowohl Präzisions- als auch Powergriffe zu berechnen. Insgesamt waren alle fünf der aus den Simulationsergebnissen ausgewählten Griffe mit TASER ausführbar. Wie in Tabelle 7.2 zu sehen ist, liegen die erforderlichen Griffkräfte innerhalb der Objektoleranzen. Dies hat sich auch bei der Ausführung der Griffe mit TASER bestätigt. Entsprechend den gestellten Qualitätsanforderungen ist Griff 1 der stabilste Griff und Griff 2 der beste, wenn alle Operationen gleich bewertet werden. Griff 3 ist der stabilste Griff, wenn das Glas von oben gegriffen werden soll. Griff 4 bietet die meisten Möglichkeiten für eine Übergabe. Griff 5 zeichnet sich durch seine besondere Griffgeometrie aus.

Kontaktpunkt	Basis	F1 distal	F2 distal	F3 proximal	F3 distal
Griff 1	0,52 N	5,17 N	4,80 N	0,61 N	-
Griff 2	3,03 N	3,75 N	3,18 N	1,94 N	3,74 N
Griff 3	1,92 N	4,15 N	1,78 N	-	4,46 N
Griff 4	-	4,24 N	4,46 N	-	2,16 N
Griff 5	6,29 N	6,55 N	1,10 N	-	1,23 N

Tabelle 7.2: Kräfte entlang der Z-Achse der Kontaktpunkte zu den Griffen aus Abbildung 7.15.

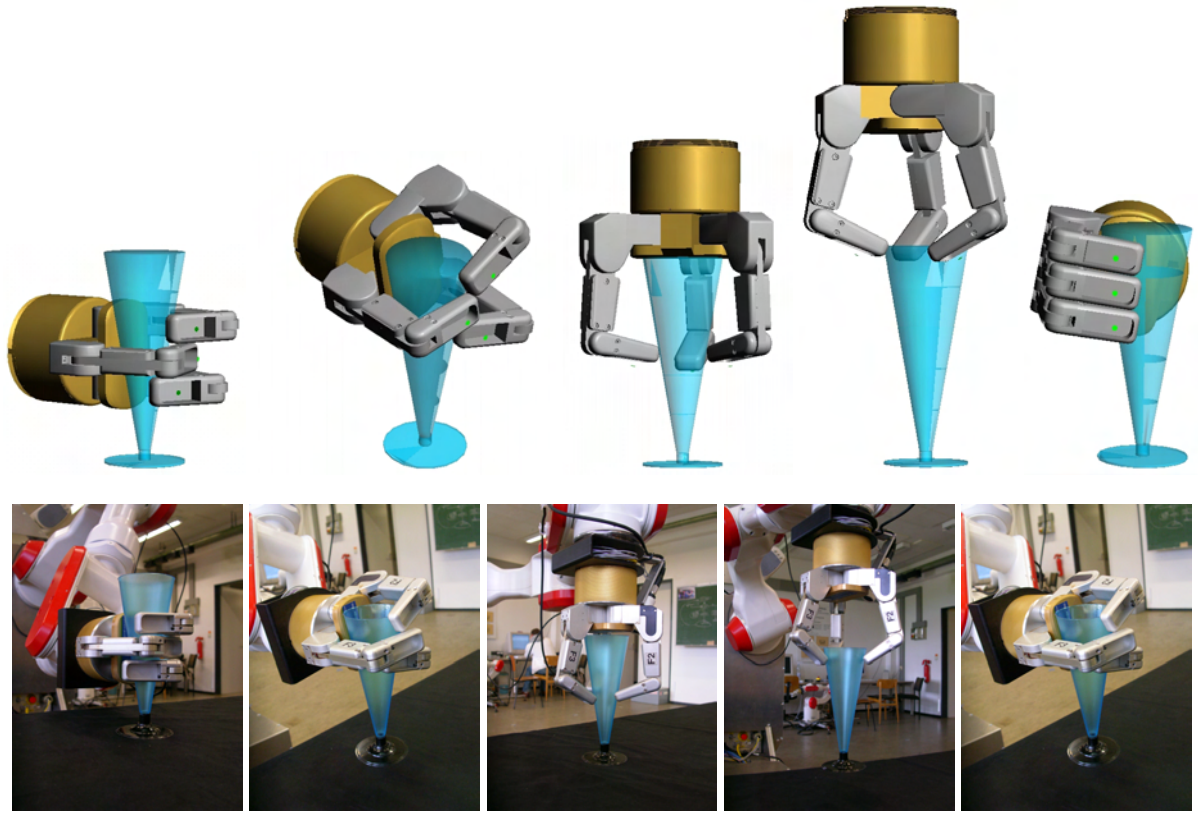


Abbildung 7.15: Griffe für ein Sektglas.

7.2.8 Griffe für eine Banane

Bananen ähneln sich zwar, sind aber in Bezug auf Form und Gewicht nicht identisch. In der Simulation wurden insgesamt über 4.000 Griffe für eine Banane berechnet. Eine Auswahl der Griffe, die mit dem Roboter TASER evaluiert worden ist, ist in Abbildung 7.16 dargestellt. Die abgebildeten Griffe ließen sich alle mit dem Robotersystem umsetzen und waren, entsprechend den Ergebnissen aus der Simulation, stabil. Allerdings gab es auch bei den Griffen, die für die Banane berechnet worden sind, instabile Griffe. Dieses ist, ähnlich wie bei den Griffen für das Buch, auf Ungenauigkeiten des Modells zurückzuführen. Eine Banane ist innerhalb gewisser Grenzen elastisch und gibt unter Druck nach. Durch diese Eigenschaft ist es möglich, dass sich Kontaktpunkte verschieben. Hierdurch verschiebt sich die gesamte geometrische Anordnung der Kontaktpunkte und der Griff wird eventuell instabil. Von den mehr als 4.000 berechneten Griffen besteht nach einer Schätzung bei ca. 20% der Griffe die Gefahr, dass sie durch ein entsprechendes Verhalten instabil werden. Beispiele für solche Griffe sind die Griffe 4 und 5 aus Abbildung 7.16. Sieht man von den Griffen ab, die aufgrund des Modellfehlers nicht ausführbar bzw. instabil sind, so zeigt die gute Umsetzbarkeit der restlichen Griffe dennoch, dass eine Generalisierbarkeit vorhanden ist. Von den evaluierten Griffen ist Griff 1 derjenige mit der am besten bewerteten Stabilität. Griff 2 zeichnet sich durch viele Übergabemöglichkeiten aus, und Griff 3 ist zu bevorzugen, wenn alle Operationen mit der gleichen Gewichtung bewertet werden. Die Griffe 4 und 5 zeichnen sich durch ihre Geometrie aus. Griff 5 ist dabei ein reiner Zwei-Finger-Griff.

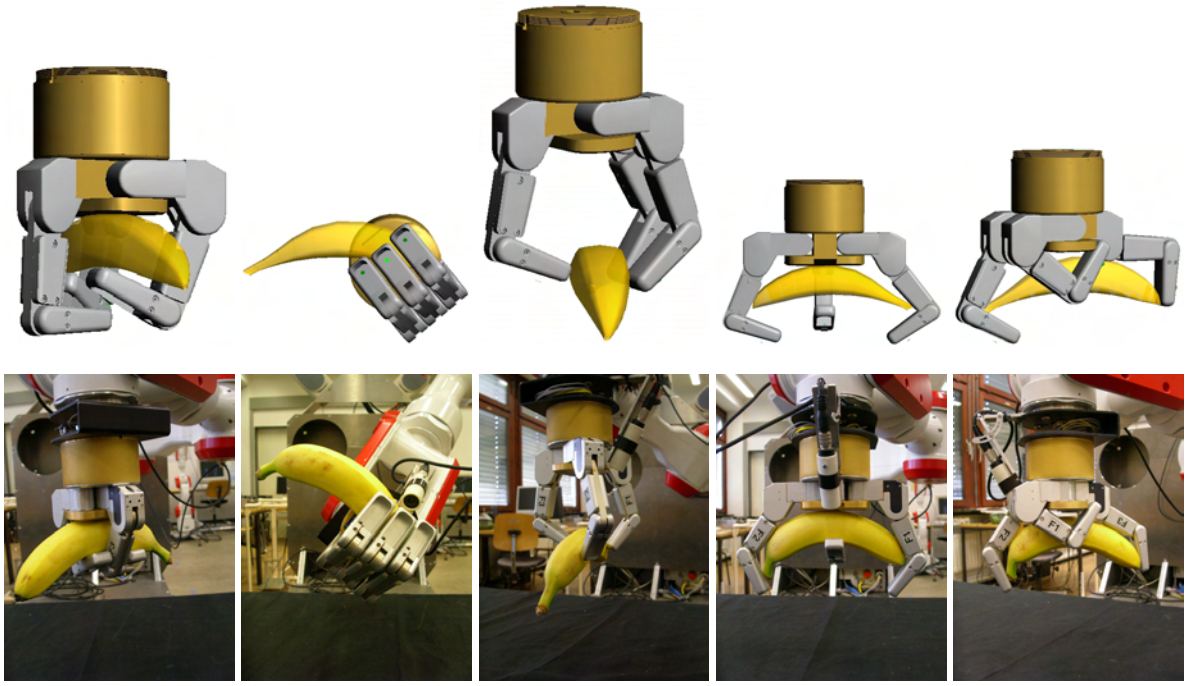


Abbildung 7.16: Griffe für eine Banane.

7.2.9 Griffe für einen Ball

Für einen Ball mit 35 mm Radius war das System in der Lage, mehr als 750 Griffe zu berechnen. Allerdings haben sich auf Grund der Symmetrie des Balls nur drei unterschiedliche Grifftypen herausgebildet. Die weiteren Griffe unterscheiden sich nur in Nuancen von den drei Grundtypen. Die drei Grifftypen sind in Abbildung 7.17 dargestellt. Die Typen unterscheiden sich durch die Anzahl der Kontakte und durch den Winkel des Spreizgelenks. Griff 1 ist ein Präzisionsgriff mit genau drei Kontakten an den Fingerspitzen. Griff 2 ist ein Powergriff mit einem Spreizwinkel von 90° und einem zusätzlichen Kontakt an einem proximalen Fingerglied (in diesem Fall F3) sowie den Kontakten an den Fingerkuppen und der Basis der BarrettHand. Der dritte und letzte Typ zeichnet sich durch einen Spreizwinkel aus, der kleiner als 90° ist und fünf Kontaktpunkte hat. Von diesen Kontaktpunkten liegen drei an den Fingerkuppen, einer am proximalen Teil eines Fingerglieds und einer an der Basis der Hand. Auch in diesem Fall liegen die berechneten Kontaktkräfte innerhalb der Objekttoleranz. Die Kräfte für die Griffe aus Abbildung 7.17 sind in Tabelle 7.3 zusammengefasst. Wie zu sehen ist, sind die berechneten Kräfte bei allen Grifftypen relativ identisch und auch relativ gleichmäßig auf alle Kontaktpunkte verteilt. Lediglich der Präzisionsgriff (Griff 1) bildet hierbei eine Ausnahme. Der F2-Finger (der obere rechte Finger) übt bei dem Griff nur eine relativ geringe Kraft auf das Objekt aus. Dabei zeichnet sich Grifftyp 1 durch die meisten Übergabemöglichkeiten aus. Griff 2 ist besonders stabil und Griff 3 ist der „Universalgriff“, wenn keine Operation bevorzugt wird.

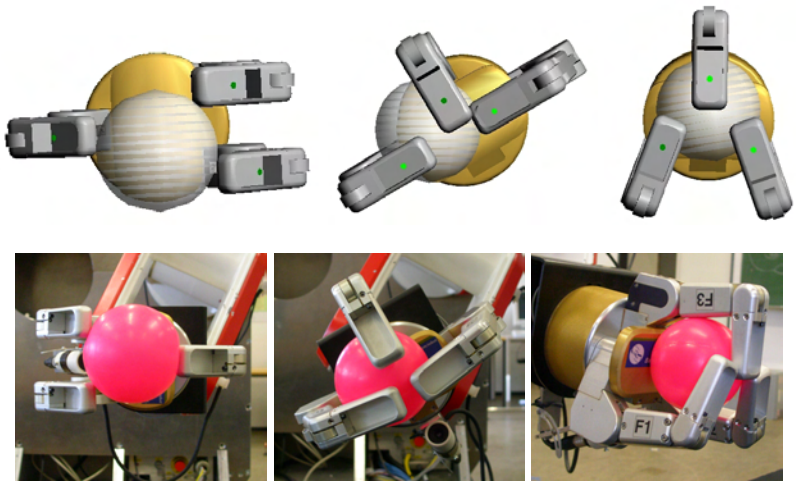


Abbildung 7.17: Griffe für einen Ball.

Kontaktpunkt	Basis	F1 distal	F2 distal	F3 proximal	F3 distal
Griff 1	-	3,75 N	0,41 N	-	3,66 N
Griff 2	4,49 N	2,46 N	2,58 N	3,97 N	1,38 N
Griff 3	2,77 N	3,93 N	3,26 N	2,37 N	2,31 N

Tabelle 7.3: Kräfte entlang der Z-Achse der Kontaktpunkte zu den Griffen aus Abbildung 7.17.

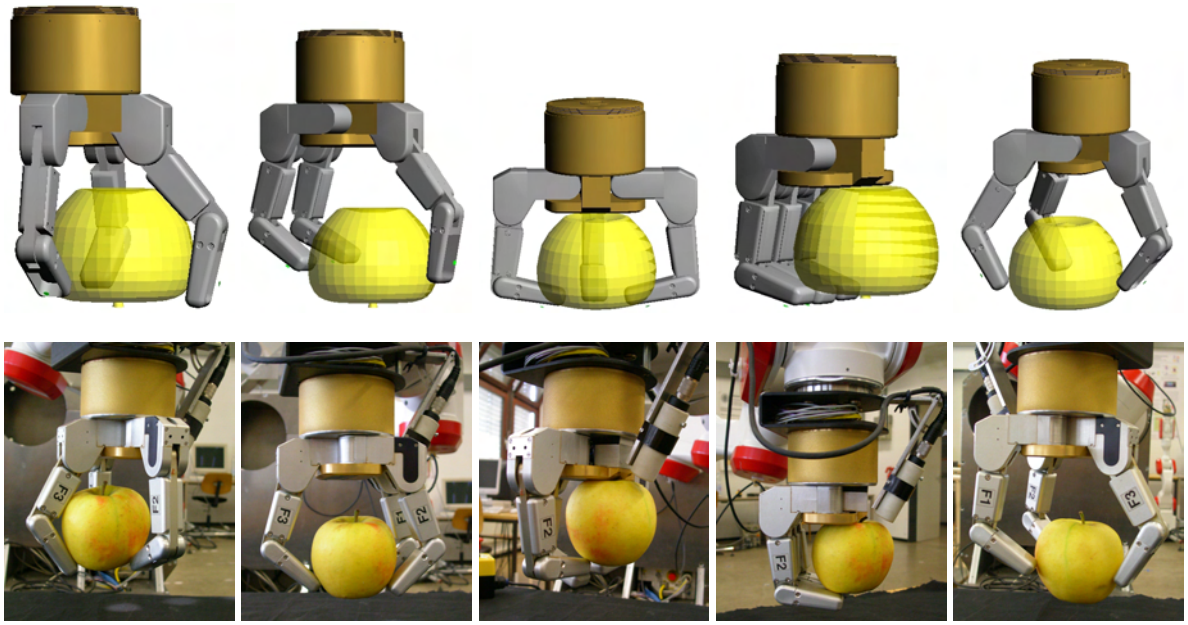


Abbildung 7.18: Griffe für einen Apfel.

7.2.10 Griffe für einen Apfel

Ein Apfel ähnelt in seiner Form einer Kugel. Allerdings gelten für einen Apfel die gleichen Bedingungen wie für eine Banane: es gibt keine zwei gleichen Äpfel. Somit gibt es auch keinen Apfel, der exakt mit dem Modell der Simulation übereinstimmt. Nichtsdestotrotz konnten alle Griffe, die für den Apfel mittels der Simulation berechnet worden sind, mit dem realen Service-Roboter ausgeführt werden. Die entsprechenden Griffe sind in Abbildung 7.18 dargestellt. Die in der Simulation berechneten Kräfte liegen auch in diesem Fall innerhalb der Toleranzen des Objekts von 10 N, wie in Tabelle 7.4 zu sehen ist.

Kontaktpunkt	Basis	F1 distal	F2 distal	F3 proximal	F3 distal
Griff 1	-	4,33 N	4,14 N	3,95 N	1,46 N
Griff 2	-	6,33 N	1,09 N	-	7,36 N
Griff 3	7,48 N	7,50 N	7,50 N	-	0,96 N
Griff 4	8,21 N	8,88 N	8,28 N	-	8,23 N
Griff 5	-	9,91 N	5,01 N	-	9,65 N

Tabelle 7.4: Kräfte entlang der Z-Achse der Kontaktpunkte zu den Griffen aus Abbildung 7.18.

7.2.11 Griffe für einen Akkuschrauber

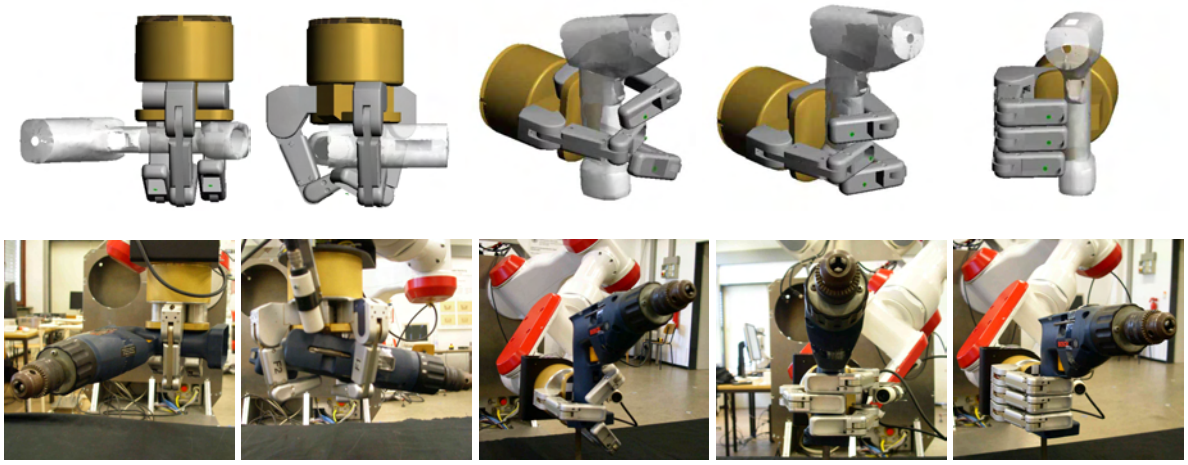
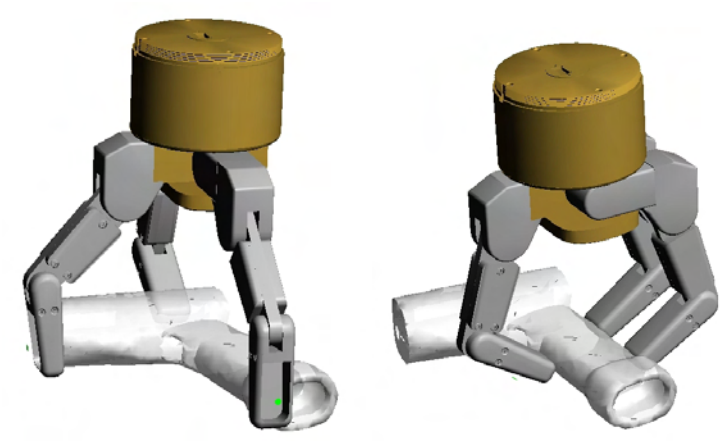


Abbildung 7.19: Griffe für einen Akkuschrauber.

Der Akkuschrauber stellt mit einem Gewicht von ca. 3 kg und seiner Größe die obere Schranke dessen dar, was mit einer BarrettHand gegriffen werden kann. Die Mehrzahl der Griffe, die in der Simulation für den Akkuschrauber berechnet worden sind, sind allerdings nicht in der Realität umsetzbar. In der Simulation wurde nur der Griff des Werkzeugs betrachtet. Somit sind auch Griffe an Positionen berechnet worden, an denen z.B. der Bohrkopf sitzt, wie in Abbildung 7.20(a) dargestellt. Die berechneten Präzisionsgriffe sind nicht stabil. Ein Beispiel für einen solchen Griff ist in Abbildung 7.20(b) dargestellt. Bei diesem Griff liegt das gleiche Problem vor, wie es schon bei den Griffen für den Becher beschrieben

worden ist. Es ist sehr schwer, exakt die in der Simulation berechneten Punkte für den Griff zu treffen. Allerdings führen kleine Abweichungen bereits zur Instabilität des Griffs. Daher bleiben zwei Grifftypen übrig, die mit dem Service-Roboter evaluiert worden sind. Sie sind in Abbildung 7.19 dargestellt.



(a) Nicht ausführbarer Griff aufgrund einer Modellungenauigkeit.

(b) Nicht ausführbarer Griff aufgrund von Berechnungsfehlern.

Abbildung 7.20: Nicht ausführbare Griffe für einen Akkuschrauber.

7.3 Zusammenfassung der Experimente

Von den 70 für die Experimente ausgewählten Griffen ließen sich 49 ausführen und ergaben einen stabilen Griff. Es sind also ungefähr 70% der Griffe direkt von der Simulation auf die reale Anwendung übertragbar. Im Wesentlichen gab es zwei Gründe, dass Griffe, die in der Simulation als stabil berechnet worden sind, nicht auf das reale System übertragbar waren: Der erste besteht in einer zu großen Abweichung zwischen dem Modell, das in der Simulation benutzt worden ist, und dem realen Objekt. Dies war z.B. beim Telefonhörer und Buch der Fall. Dies führte dazu, dass die Kontakte und die entsprechenden Kräfte falsch berechnet wurden und sich nicht in die Realiät übertragen ließen. Bei dem Buch ist die „Dynamik“ des Objekts nicht mit in die Berechnung eingeflossen. Somit konnte das Verhalten im Fall des Anhebens nicht korrekt simuliert werden. Ein weiteres Problem liegt in der Mechanik der BarrettHand. Damit der TorqueSwitch-Mechanismus ausgelöst wird, muss eine Mindestkraft aufgebracht werden. Diese Kraft wird aber nicht in jedem Fall erreicht, oder der Mechanismus löst aufgrund mechanischer Probleme nicht aus. Dies war z.B. bei einigen Griffen für das Sektglas der Fall, bei denen der TorqueSwitch nicht immer ausgelößt worden ist. Somit ergibt sich nicht die zuvor berechnete Griffkonfiguration, und der Griff ist instabil.

Schließt man aber nun die Griffe aus, die sich aufgrund der Vereinfachung bei der Modellierung als instabil erwiesen (12 Griffe), so sind 82,9% der in der Simulation berechneten Griffe auch in der Realität stabil. Ein Fehler von ca. 17% ist ein durchaus vertretbarer Wert, wenn man bedenkt, dass die Modelle, die für die Simulation verwendet worden sind, nicht exakt mit dem realen Objekt übereinstimmen.

Auch die Anwendbarkeit des Qualitätsmaß wurde gezeigt. Die automatisch von dem System gewählten Griffe entsprechen den gestellten Anforderungen. Sind die Griffe auch in der Realität stabil, so lassen sich auch die Folgeoperationen ausführen. Da mit dem Verfahren zur Griffberechnung sowohl stabile Power-

Griffe, als auch stabile Präzisionsgriffe generiert werden können, hebt es sich deutlich von den bisher bekannten Verfahren ab. Zudem können Griffe für beliebige Objekte berechnet und evaluiert werden.

Der Vergleich mit anderen Verfahren für die Berechnung von Griffen und deren Umsetzung mit einem Roboter-System ist nicht möglich. Zwar beschreiben die Autoren z.B. in [Asfour et al. 2006a, Morales et al. 2006b, Kim et al. 2004a, Hilleenbrand et al. 2004, Miller 2001] und [Kragic et al. 2001], dass es möglich war, die geplanten Griffe mit dem jeweiligen Roboter-System umzusetzen, jedoch erfolgen keine weiteren Angaben. Somit ist weder ein quantitativer noch ein qualitativer Vergleich dieser Arbeit mit anderen Arbeiten möglich.

Kapitel 8

Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Service-Roboter sowie die zugehörigen Software-Komponenten entwickelt. Darauf aufbauend ist ein System zum Lernen und zur Ausführung von Griffen erarbeitet worden. Die folgenden Problemstellungen wurden bearbeitet:

- Aufbau eines Service-Roboters für den autonomen Einsatz in einer Büroumgebung.
- Softwaretechnische Integration mehrerer Hardwarekomponenten zu einem Gesamtsystem (Integration von Hand- und Armsteuerung in einer Einheit).
- Entwicklung eines neuen Qualitätsmaßes für die Evaluation von Griffen auf Basis der Weiterverwendbarkeit des gegriffenen Objekts.
- Entwicklung eines Verfahrens zum Lernen von Griffen mittels selbstbewertendem Lernen.
- Realisierung eines Systems zum Greifen von Alltagsgegenständen auf Basis des entwickelten Qualitätsmaßes unter Verwendung der automatisch generierten Griffe.

Wie die Ergebnisse und Analysen der einzelnen Teile bereits gezeigt haben (siehe Kapitel 4.4, 6.8, 7.2), verfügt das System insgesamt über eine gute Performanz. Das Qualitätsmaß wurde auf Basis der Weiterverwendbarkeit der gegriffenen Objekte entwickelt. Diese wird anhand von vier typischen Aufgaben definiert: Einfüllen, Ausgießen, Übergabe und Bewegung. Die Operationen können mittels eines Objektmodells evaluiert werden. Somit erfolgt eine Verknüpfung von Griffen mit der Objektsemantik. Das entwickelte Qualitätsmaß stellt eine enorme Vereinfachung bei der Benutzung des Service-Roboters dar, da hierdurch auch technische Laien in der Lage sind, ihre Anforderungen an einen Griff zu formulieren. Dies is bei rein kraftbasierten Kriterien nur schwer möglich.

Um mittels Reinforcement-Lernen Griffe berechnen zu können, war eine Modifikation des Lernverfahrens erforderlich. Die Standard-Verfahren sind nur für eine geringe Zahl von Terminal-Zuständen ausgelegt. Bei dem Greifen von Objekten kann jedoch jeder Zustand durch das Schließen der Finger in einen Terminal-Zustand überführt werden. Zudem kann es bei einer teilweise zufälligen Aktionsauswahl nicht verhindert werden, dass auch fehlgeschlagene Greifversuche erneut ausgeführt werden. Daher wurde eine dynamische Aktionsreduzierung eingeführt. Durch eine modifizierte Aktualisierungsregel werden so Greifaktionen die in einem instabilen Griff resultieren, aus dem Aktionsraum des entsprechenden Zustandes gelöscht. Auch Greifaktionen, die in einem stabilen Griff resultieren werden mit dieser neu entwickelten Regel nach einer ausreichenden Exploration gelöscht.

Die automatische Generierung von Griffen wurde mittels einer eigens entwickelten Simulation evaluiert. Sie ist einem Verfahren auf Basis der Breitensuche deutlich überlegen. Auch in Bezug auf die Griffart, die mittels des Lernverfahrens generiert werden kann, weist die Methode des selbstbewertenden Lernens

Vorteile auf. So erlaubt sie nicht nur die Berechnung von Griffen für beliebige Objekte, sondern auch die Berechnung von Griffen mit beliebig vielen Kontaktpunkten. Das Zusammenspiel aller Komponenten wurde in einem System zum Greifen von Alltagsgegenständen mit dem realen Service-Roboter evaluiert. Hierbei zeigte sich, dass ca. 70% der in der Simulation berechneten Griffe auch mit dem realen Robotersystem TASER ausgeführt werden können. Bei den meisten Griffen, die sich mit dem Service-Roboter als instabil erwiesen, waren Modellfehler die Ursache. Vernachlässigt man diese Griffe, so können ungefähr 83% der simulierten Griffe in der Realität umgesetzt werden.

Trotzdem bestehen noch einige Verbesserungs- und Erweiterungsmöglichkeiten. Die Ansätze zur Weiterentwicklung lassen sich dabei in drei Klassen einteilen. Zum einen sind Verbesserungen an der Hardware möglich, als zweites kommen Erweiterungsmöglichkeiten für die Generierung von Griffen in Betracht, und als drittes gibt es Möglichkeiten, die Performanz des Gesamtsystems zu steigern.

Hardwarespezifische Verbesserungsmöglichkeiten

Wie sich bereits im Kapitel 2.3 gezeigt hat und im Kapitel 7.2 durch die durchgeführten Experimente bestätigt wurde, ist die Sensorik der BarrettHand nicht ausreichend. Es können nur Kräfte, die senkrecht auf die Fingerspitzen treffen, gemessen werden. Es besteht auch keine Möglichkeit, über die Sensorik einen Kontakt an der Handfläche festzustellen. Der Einsatz von Kraftmomentsensoren anstelle des letzten Fingergliedes, wie es z.B. im TRISK Projekt am MIT [Trisk 2007] erfolgt, stellt eine gute Alternative dar. Auch die Montage einer „Kontaktfolie“ auf weiteren Flächen der Hand würde die Möglichkeit bieten, Kontakte und evtl. sogar die damit verbunden Kräfte zu messen. Hierdurch ergeben sich auch Möglichkeiten, Griffe durch eine Analyse der auftretenden Kräfte on-line zu optimieren.

Eine weitere Verbesserung könnte mit der Montage eines Kraft-Drehmomentsensors am Roboterarm erreicht werden. Mit einem zusätzlichen Kraftmomentsensor im Handgelenk des Roboters wäre es möglich, die auftretenden Drehmomente beim Anheben eines Objekts festzustellen um, ähnlich wie in der Arbeit von Rössler [Rössler 2001], weitere Aussagen über die Qualität eines Griffs zu treffen.

Verbesserungsmöglichkeiten bei der Berechnung von Griffen

Eine Analyse der Objekte bezüglich ihrer Symmetrieeigenschaften könnte den Lernvorgang beschleunigen. Viele Griffe ließen sich so anhand der Symmetrieachsen an eine andere Position des Objekts übertragen, ohne erneut bezüglich ihrer Stabilität evaluiert werden zu müssen.

Eine Hierachisierung des Lernens bzw. eine dynamische Aufteilung des Suchraums, wie in den Ansätzen in Kapitel 5.2.5 beschrieben, kann genutzt werden, um den Suchraum effektiver zu analysieren. So könnten Bereiche, in denen sich bei einem groben Suchraster kein stabiler Griff gebildet hat, bei der erneuten Evaluation mit einem feineren Raster ausgespart werden. In diesem Zusammenhang kann auch zu einer kontinuierlichen Zustandsraumbeschreibung übergegangen werden. Bisher wird eine Diskretisierung des Zustandsraumes vorgenommen. Wenn ein Mensch ein Objekt greift, besteht in den meisten Fällen aber kein Unterschied zwischen Griffen, die sich nur in der Translation entlang einer Achse um wenige Millimeter unterscheiden. Es wird also eine Menge von Griffen zu einem für eine bestimmte Objektregion definierten Griff, zusammengefasst. Dies lässt sich auch auf den Bereich des Reinforcement-Lernens übertragen.

Grundsätzlich besteht die Möglichkeit, die Simulation zu verbessern. So bietet sich zunächst die Ergänzung der Kontaktmodelle um das Modell der Weichen-Finger-Kontakte an. Auch die Simulati-

on von elastischen Objekten kann eine weitere Verbesserung des Verfahrens bewirken. Eine generelle Simulation der Dynamik kann verwendet werden, um z.B. das Verhalten der Objekte beim Anheben zu berechnen. Dies kann, wie z.B. bei dem Greifen eines Buchs (siehe Kapitel 7.2.2), für eine detaillierte Analyse benutzt werden. Es ist aber zu bedenken, dass eine Berechnung der Dynamik sehr aufwendig ist. Die Evaluation der Griffe mit TASER hat gezeigt, dass sie in den meisten Fällen auch nicht erforderlich ist.

Auch eine Verfeinerung der Polygonmodelle der Objekte ist denkbar. Steigt die Rechenleistung der Computer weiter so wie bisher, stellt die schnelle Berechnung der Kollisionserkennung mit einigen Millionen Polygonen in naher Zukunft kein Problem mehr dar. Es können also mehr Objektdetails mit in die Polygonmodelle der Objekte eingebracht werden, ohne das hierdurch ein Nachteil entsteht. Auch eine Modellierung der Arbeitsfläche kann integriert werden, um so z.B. nur Griffe zu generieren, die geeignet sind, ein Objekt vom Tisch aufzunehmen.

Bei der Modellierung der Objekte wurde von einer homogenen Massenverteilung innerhalb der Objekte ausgegangen. Somit entspricht das geometrische Zentrum des Objekts auch dem Massenschwerpunkt. Für einige Objekte, wie z.B. einen Hammer oder einen Akkuschrauber, ist dies nicht korrekt. Die Folge hiervon ist, dass Griffe, die in der Simulation noch als stabil evaluiert worden sind, in der Realität instabil sind. Eine Modellierung der Massenverteilung bei einem Objekt könnte genutzt werden, um diesen Fehler zu korrigieren.

Verbesserungsmöglichkeiten des Gesamtsystems

Um die Performanz des Gesamtsystems zu steigern, sind einige Verbesserungen denkbar. Besonders bei der Objekterkennung bietet sich die Verwendung anderer Verfahren an. Sie sind zwar in der Regel mit deutlich mehr Rechenaufwand verbunden, bieten aber auch eine deutlich bessere Performanz. Hier sind insbesondere Verfahren interessant, die zusätzlich zur Objekterkennung die Erkennung der Orientierung des Objekts ermöglichen. Das von Lowe [Lowe 2004] entwickelte Verfahren bietet sich für Objekte an, die eine entsprechend stark texturierte Oberfläche haben. Bei der einfachen blauen Tasse (siehe Kapitel 6.8.3.3) ist das Verfahren nur sehr schwer anwendbar. Für die dreidimensionale Erkennung von Objekten kann auch das in [Krüger and Peters 2000] vorgestellte Verfahren eingesetzt werden. Besonders bei Szenen mit Verdeckung von Objekten ist der von Graf gewählte Ansatz auf Basis von Fuzzy-Invarianten [Graf 2000] interessant.

Dies gilt in ähnlicher Form auch für die Positionserkennung der Objekte. Bei einer Trennung von Objekt- und Positionserkennung bietet sich die Verwendung anderer Verfahren für die Berechnung der Objektorientierung an. Eine Möglichkeit bietet z.B. das in [Peters 2004] vorgestellte Verfahren auf der Basis von Gabor-Wavelets.

Bisher ist das System nur in der Lage, Objekte automatisch zu greifen, die zuvor in der Simulation trainiert wurden. Dies wiederum setzt voraus, dass ein (Polygon-) Modell von dem Objekt existiert. Ist diese nicht vorhanden und dass Objekt somit dem System unbekannt, sollte das System in der Lage sein, automatisch ein Modell zu erstellen. Wie z.B. in [Pollefeys et al. 2004] gezeigt wurde, ist dies mit einem monokularen Visionsystem, wie der Handkamera, möglich.

Um die Flexibilität des Systems zu steigern, ist eine weiterreichende Aktionsplanung sinnvoll. Falls ein Objekt aufgrund seiner aktuellen Position nicht entsprechend der geforderten Qualität gegriffen werden kann, sollte das System in der Lage sein, das Objekt aufzunehmen und umzugreifen. Eine Möglichkeit,

eine solche Aktionsplanung zu realisieren, besteht in der Adaption des von Markus Ferch entwickelten Verfahrens zum Erlernen von Montagesequenzen [Ferch 2001].

In Verbindung mit einer derartigen Planung kann eine Umgebungsmodellierung mit in Betracht gezogen werden. So wäre es z.B. möglich, die gesamte Objektumgebung in einem dreidimensionalen Modell zu erfassen, um weitere Aussagen über die Erreichbarkeit von Objekten treffen zu können. So kann es z.B. vorkommen, dass ein Objekt nicht erreichbar ist, weil es von einem anderen Objekt verdeckt wird. In diesem Fall müsste das System in der Lage sein, eine Szene erst „aufzuräumen“, bevor bestimmte Aktionen ausführbar sind. Auch dies könnte mit dem von Markus Ferch vorgeschlagenen Ansatz erfolgen. Eine Alternative besteht in der Verwendung des von Yorck von Collani [von Collani 2001] vorgestellten Verfahrens zur automatischen Sequenzgenerierung. Ein entsprechender Ansatz zur Szenenrekonstruktion mit einer monokularen Handkamera ist von [Jockel et al. 2007] entwickelt worden.

Anhang A

Hardwaredetails

Im Folgenden werden weitere Details zur Hardware des Service-Roboters TASER dargestellt. Zunächst werden weitere technische Details zur mobilen Basis des Systems gegeben. Anschließend werden weitere Details zum MHI-PA10-6C Roboterarm aufgeführt. Die Darstellung schließt mit Details zur BarrettHand.

A.1 Ergänzende technische Details zur mobilen Basis

Der Controller für die Schwenk-Neige-Einheit (Pan-Tilt-Unit [PTU]) und der Controller für die Handkamera sind in der mobilen Einheit in dem Fach über den Akkumulatoren montiert. Der Turm beinhaltet die Steuerungselektronik für den Arm, sowie einen Spannungswandler für die BarrettHand (Abb. A.1).

Batterien	8 × 12 V, 40 Ah Blei-Gel
Hauptspannung	48 V (4 × 12 V in Reihe)
Gesamtkapazität	80 Ah ((2 × 40 Ah parallel)
Gesamtleistung	3,84 kWh
Zusätzliche Spannungen	+24, +12 und +5 V

Tabelle A.1: Daten der Elektrik der MP-L655 Plattform.

Die Daten der Elektrik sind Tabelle A.1 aufgeführt. Die Stromversorgung wird durch acht Blei-Gel-Akkus realisiert, die jeweils zu viert in Reihe geschaltet sind. Somit ergibt sich eine Spannung von 48 V und eine Gesamtkapazität von 80 Ah. Die 3,84 kWh betragende Gesamtleistung ermöglicht dem System eine Laufzeit von mehr als sieben Stunden. Zusätzlich zu der Hauptspannung von 48 V, mit der der PC, die Motoren und der Arm betrieben werden, stehen die Spannungen 24 V, 12 V und 5 V für weitere Komponenten zur Verfügung.

Die Maße des Systems sind in Tabelle A.2 zusammengefasst. Die „Position des Arms“ beschreibt den Versatz des Roboterams relativ zum Ursprung der Basis. Dieser befindet sich ebenerdig im Zentrum der mobilen Einheit.

Da der PC die Steuerung der gesamten Hardware der Roboterplattform übernehmen muss, ist er mit den folgenden Komponenten ausgerüstet:

- Ein *ICP Industrial Computer Products*[ICP 2007] „ROCKY-4784EV“ CPU Mainboard mit einem 2.4 GHz Intel Pentium 4 Prozessor, 512 Mb RAM, einer 2 1/2 Zoll Notebook Festplatte mit einer Kapazität von 40 GB, On-Board Grafikkarte und Ethernet-Schnittstelle

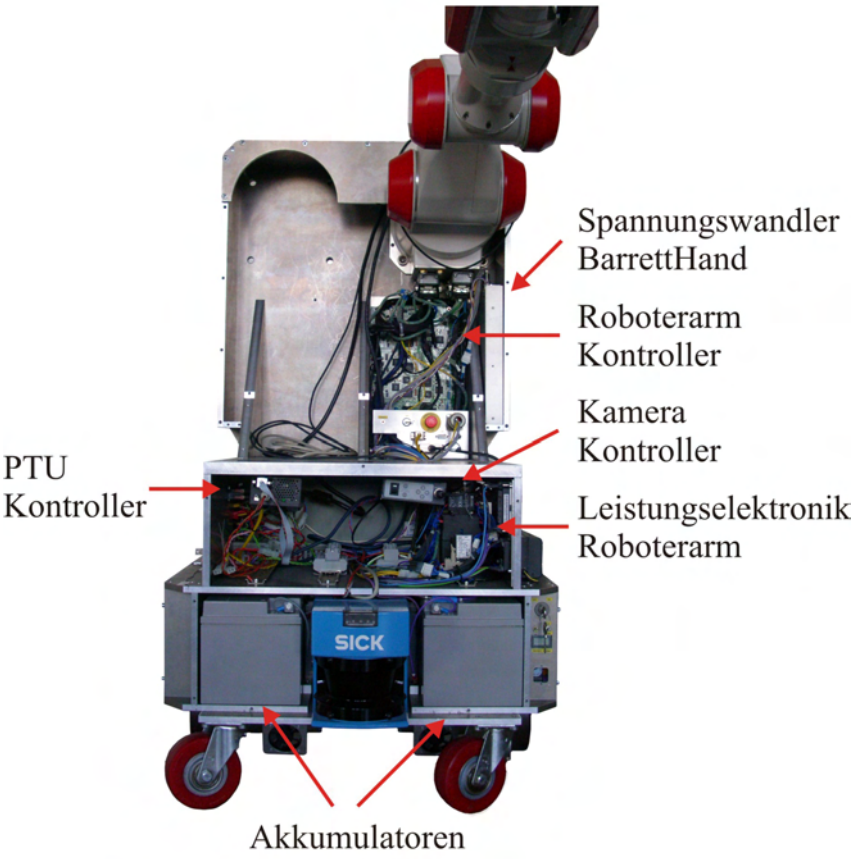


Abbildung A.1: Aufbau des Roboters TASER.

Breite	680 mm
Tiefe	610 mm
Höhe Basis	570 mm
Breite Turm	600 mm
Höhe Turm	660 mm
Position Arm	170 × -279,5 × 1102 mm
Gesamtgewicht	ca. 200 kg

Tabelle A.2: Maße der mobilen Plattform.

- eine *Matrox*[Matrox 2007] „Meteor“ PCI Framegrabber Karte (siehe Abschnitt 2.5 für Details)
- eine (modifizierte) *Contempory Controls*[CC 2007] „PCI22-485X“ PCI Arcnet Karte und
- eine *D-Link*[D-Link 2007] 54Mbit PCI-Wireless-Lan-Karte.

Die Laserscanner der mobilen Plattform sind über ein Ethernet-Interface angebunden. Hierfür wurde mittels eines *Rabbit Powercore* ein Adapter für RS-422 – Ethernet entwickelt [Bistry et al. 2007, Bistry und Pöhlsen 2006].

Die Steuerung der Motoren der mobilen Einheit erfolgt über ein CAN-Bus Interface, angeschlossen an den Parallel-Port des Rechners. Die Controller für die Motoren der Plattform sind, ebenso wie die Getriebe, mit in die Motorengehäuse integriert (Abb. A.2).

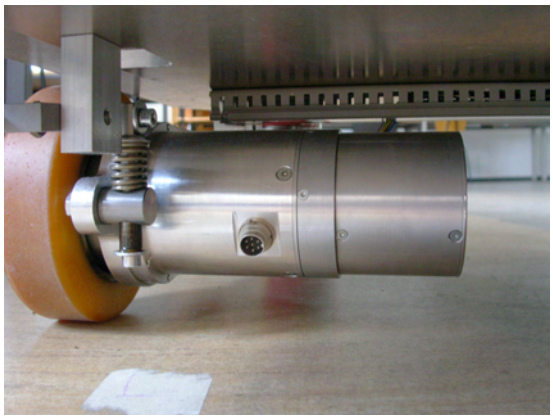


Abbildung A.2: Ein Motor inklusive Kontroller und Getriebe.

Mit den Motoren sind

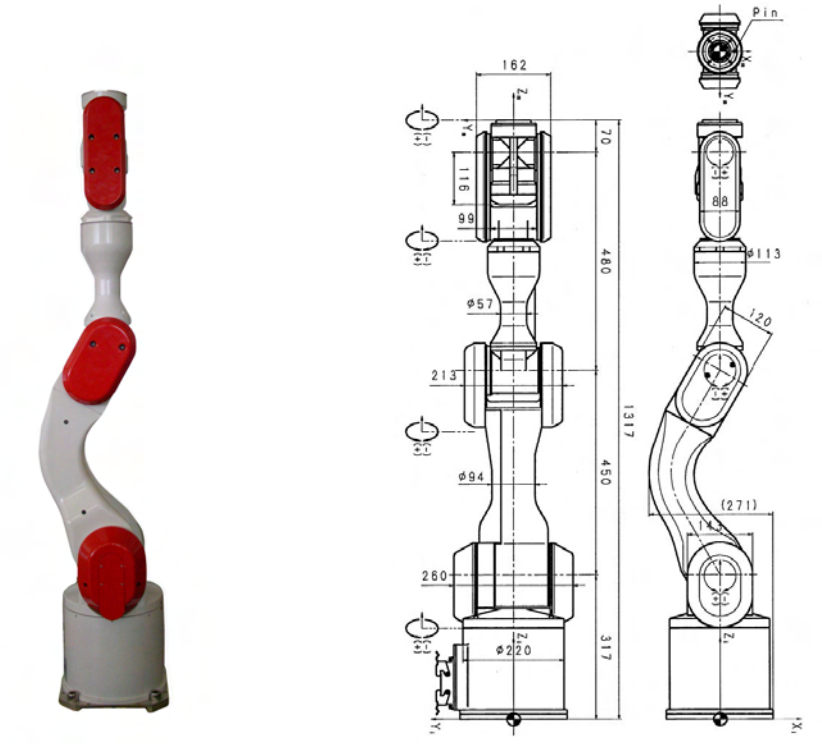
- 1 m/s translatorische Bewegung
- 90 °/s rotatorische Bewegung

möglich.

Die Sensoren der Plattform sind ebenfalls über das CAN-Bus Interface ansprechbar. Als Sensoren stehen ein Kompass, die Odometriesensoren, ein Spannungsmesser und ein Temperaturfühler zur Verfügung. Der Kompass misst die Rotation der Plattform relativ zu einer festen Position (GYRO). Die Odometriesensoren liefern den Winkel, um den sich ein Rad bewegt. Der Spannungsmesser ermittelt die aktuelle Spannung der Batterien und der Temperaturfühler misst der die Motor-Temperaturen. Die Ergebnisse der Odometriemessung werden jedoch durch Schlupf und Bodenunebenheiten beeinflusst, und sind somit fehlerhaft.

A.2 Ergänzende technische Details zum MHI-PA10-6C Roboterarm

Abbildung A.3 zeigt einen Roboterarm des Typs PA10-6C sowie seine Maße.



(a) (b) Maße des MHI-PA10-6C.

Abbildung A.3: Der MHI PA10-6C [MHI 2004].

Die Anordnung und Bezeichnung der Achsen ist in Abbildung A.4 dargestellt. Die Grenzwerte und technischen Daten der einzelnen Gelenke können Tabelle A.3 entnommen werden. Der Arbeitsraum des Roboters ist in Grafik A.5 abgebildet.

Der Roboter wird über wartungsarme, bürstenlose Gleichstrommotoren angetrieben und jedes Gelenk ist mit einem bürstenlosen Resolver ausgestattet. Bei einer Versorgungsspannung von 100 V ist der Roboter in der Lage 10 kg zu tragen. Da die BarrettHand ca. 1,2 kg wiegt und der Arm am TASER nur mit 48 V betrieben wird, liegt die maximale Belastbarkeit bei ca. 4 kg. Dies ist noch ausreichend, um vielen Aufgaben eines Service-Roboters zu genügen.

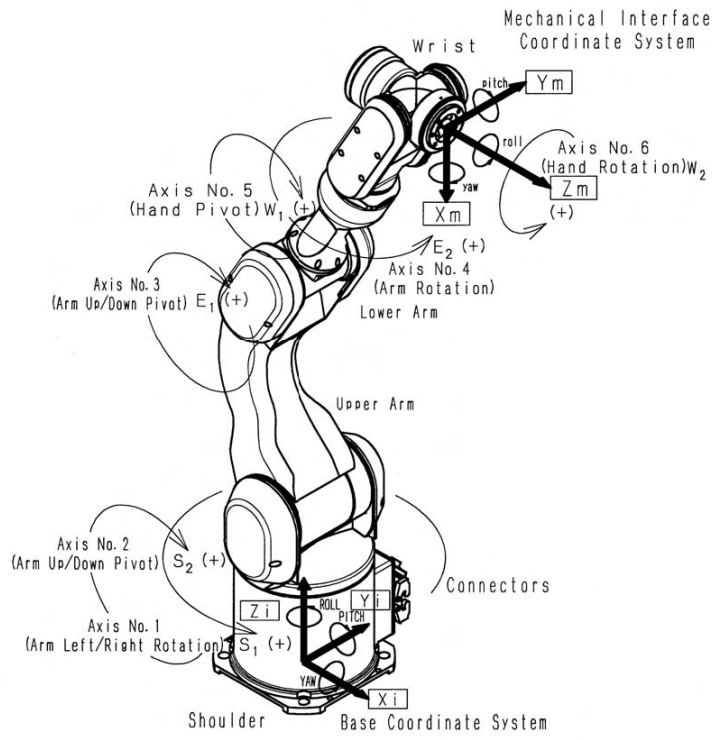


Abbildung A.4: Anordnung und Bezeichnung der Achsen des MHI-PA10-6C [MHI 2004].

Name	Grenzen			maximale Geschwindigkeit (rad/sek)
	Mechanische Grenze	Kontroller Grenze	Software Grenze	
S1 (Rotation)	$\pm 180°$	$\pm 178°$	$\pm 177°$	± 1
S2 (Swing)	$+127°, -67°$	$+125°, -65°$	$+124°, 64°$	± 1
E1 (Swing)	$+164°, -113°$	$+159°, -108°$	$+158°, -107°$	± 2
E2 (Rotation)	$\pm 270°$	$\pm 256°$	$\pm 255°$	$\pm 2\pi$
W1 (Swing)	$\pm 180°$	$\pm 166°$	$\pm 165°$	$\pm 2\pi$
W2 (Rotation)	$\pm 270°$	$\pm 256°$	$\pm 255°$	$\pm 2\pi$

Tabelle A.3: Gelenkwerte für den MHI-PA10-6C.

A.3 Ergänzende technische Details zur BarrettHand

Pro Finger können 17500 verschiedene Positionen gemessen werden, so genannte (Motor-) Ticks. Bei einem maximalen Winkel von $140°$ entspricht eine Einheit einem Winkel von $0.008°$. Die Genauigkeit liegt bei der Standardeinstellung, bei ca. 25 Ticks. Sie kann per Software festgelegt werden, bei zu kleinen Toleranzen wird der Positionsregler jedoch instabil. Somit liegt die Positionsgenauigkeit der Finger bei $\sim 0.2°$ (bei 25 Ticks Genauigkeit). Das Spreizgelenk hat eine Auflösung von 3150 Positionen. Die

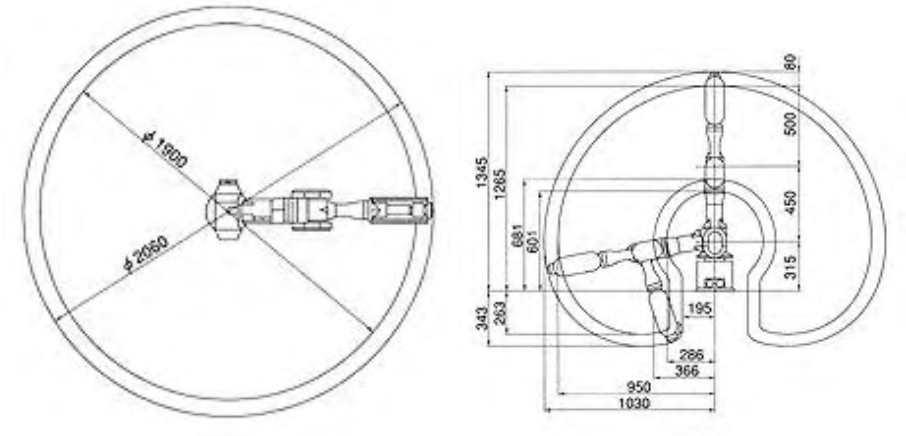


Abbildung A.5: Arbeitsbereich des MHI-PA10-6C [MHI 2004].

Differenz zwischen zwei Positionen entspricht einem Winkel von $0.06°$. Für das Spreizgelenk liegt die Positionsgenauigkeit ebenfalls bei 25 Ticks und somit die Positionsgenauigkeit bei $\sim 1.5°$.

Die maximale Geschwindigkeit für einen Finger der BarrettHand liegt bei $140°$ pro Sekunde. Für das Spreiz-Gelenk beträgt die maximale Geschwindigkeit $420°$ pro Sekunde, d.h. innerhalb von 0.3 Sekunden kann das Spreiz-Gelenk von Position $0°$ zu Position $180°$ bewegt werden.

Die Versorgungsspannung der Hand beträgt 24 V. Somit ist ein zusätzliches Netzteil erforderlich. Dieses wurde in den Turmaufbau der mobilen Einheit integriert (Abb. A.1).

Die Low-Level Motorsteuerung wird hardwareseitig von einem HCTL-1100 Kontroller [Agilent Technologies] übernommen. Die Anbindung an den Steuerrechner erfolgt über eine serielle RS232-Schnittstelle.

Im Supervisory Modus akzeptiert die BarrettHand Befehle vom Benutzerprogramm. Die die Kontrolle wird erst wieder an das Programm zurückgegeben, wenn der Befehl vollständig ausgeführt wurde. Die Kommandostruktur ist dabei `<Motor> <Kommando> <Parameter> <Wert>`. Die möglichen Werte für die Motoren sind in Tabelle A.4 aufgeführt.

Wert	Motor
1	Finger F1
2	Finger F2
3	Finger F3
4	Spread
G	Finger F1, Finger F2 und Finger F3
S	Spread
	Alle Motoren

Tabelle A.4: Motor-Präfixe.

Beispiele für Kommandos sind:

- 12C : Schließt die Finger 1 und 2

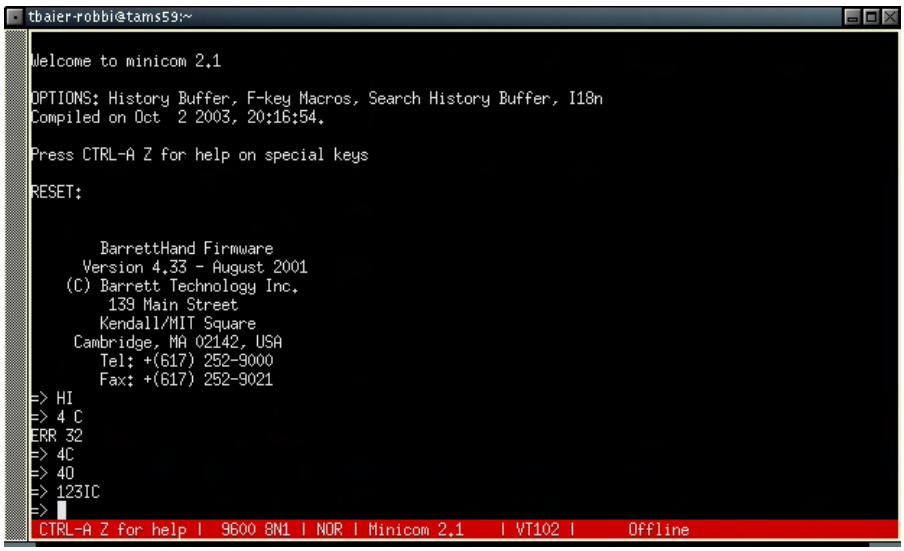


Abbildung A.6: Ein ASCII-Terminal verbunden mit einer BarrettHand.

- 4O : Öffnet das Spread-Gelenk (Gelenk 4)
- FGET TEMP : Liest die Temperatur der Hand aus
- FGET MSG : Liest den Wert für die maximal zulässige Kraft aus.

Für weitere Details der Kommandostruktur und mögliche Kommandos siehe [Barrett Manual 2002]. In Abbildung A.6 ist ein Beispiel zu sehen, bei dem die Finger über ein ASCII Terminal-Programm interaktiv bewegt wurden.

Bevor mit der Hand im Low-Level Real-Time Modus Daten ausgetauscht werden können muss definiert werden, welche Daten die Hand empfangen bzw. zurück liefern soll. Die Variablen, die sich hierbei einstellen lassen, sind in Tabelle A.5 aufgeführt .

Somit können im Real-Time Modus jederzeit Geschwindigkeitskommandos bzw. eine Änderung des Proportionalfaktors des Geschwindigkeitsreglers an die Hand gesendet werden. Diese antwortet mit den zuvor festgelegten Werten. Der Befehl `23FSET LCV 1 LCVC 1 LCPG 0 LFV 0 LFS 1 LFAP 1 LFDP 0` z.B. setzt die Parameter für den Real-Time Modus für die Finger F2 und F3. Die Finger erhalten ein Geschwindigkeitskommando und melden den gemessenen Kraftwert sowie die Absolut-Position zurück. Alle relevanten Koeffizienten wurden auf 1 gesetzt.

Dass der Feedback-Block im Real-Time Modus als Antwort auf einen Control-Block gesendet wird, wirkt sich nachteilig auf den Regelzyklus der Hand aus. Dadurch hängt z.B. der Regelzyklus für einen Positionsregler immer mindestens einen Zyklus hinterher. Ein weiterer Nachteil besteht darin, dass die Motoren träge sind und nachlaufen bzw. eine Vorlaufzeit haben. Wird z.B. in einem Control-Block die Geschwindigkeit 40 an die Hand gesendet, so dauert es mindestens 6 Zyklen, bis der Motor die Geschwindigkeit eingestellt hat, wenn er zuvor nicht bewegt worden ist.

Parameter	Name	Type	Funktion	Größe innerhalb eines Blocks
LCV	Loop Control Velocity	Flag	Falls wahr enthält der Control Block einen Geschwindigkeitswert	1 Byte
LCVC	Loop Control Velocity Coefficent	Variable (integer)	LCV wird mit LCVC multipliziert, um die Geschwindigkeit zu bestimmen	Nicht bekannt
LCPG	Loop Control Proportional Gain	Flag	Falls wahr, enthält der Control-Block einen proportionalen Verstärkungsfaktor	1 unsigned Byte
LFV	Loop Feedback Velocity	Flag	Falls wahr, enthält der Feedback-Block die Feedback Geschwindigkeit	1 signed Byte
LFVC	Loop Feedback Velocity Coefficent	Variable (integer)	Die aktuelle Geschwindigkeit kann berechnet werden aus $LVF = \frac{Velo}{LFVC}$	unbekannt
LFS	Loop Feedback Strain	Flag	Falls wahr, enthält der Feedback-Block ein KS-Wert	1 unsigned Byte
LFAP	Loop Feedback absolute Position	Flag	Fall wahr, enthält der Feedback-Block einen Wert für die Absolute Position	2 unsigned Bytes
LFDP	Loop Feedback Delta Position	Flag	Fall wahr, enthält der Feedback-Block einen Wert für die delta Position	1 signed Byte
LFDPC	Loop Feedback Delta Position Coefficent	Variable (integer)	Die aktuelle Delta Position kann berechnet werden aus $LFDP = \frac{Delta}{LFDPC}$	unbekannt

Tabelle A.5: Echtzeit-Parameter der BarrettHand

Paramater	Name	Type	Funktion	Größe innerhalb eines Blocks
LFT	Loop Feedback Temperature	Flag	Falls wahr enthält der Feedback-Block einen Temperaturwert	2 signed Byte

Tabelle A.6: Globale Echtzeitparameter der BarrettHand.

Anhang B

B-Splines

In diesem Kapitel werden zunächst die Grundlagen für B-Splines, die B-Spline Basisfunktionen, definiert. Darauf aufbauend wird die Funktionsapproximation mit Basisfunktionen erörtert.

B.1 Grundlagen

B-Splines und B-Spline-Funktionen sind ein bekanntes Teilgebiet der Numerik. Durch ihre günstigen Eigenschaften bei der Approximation von Kurven werden sie im Bereich der Computergrafik häufig eingesetzt (siehe [Cohen et al. 2001, Eberly 2004, Foley et al. 1996]).

B.1.1 B-Spline Basisfunktionen

B-Spline Funktionen ergeben sich durch Linearkombination der B-Spline Basisfunktionen. Sei B eine nicht uniforme B-Spline Basisfunktionen mit der Ordnung k und

$$S = (s_0, s_1, \ldots, s_{N_{i-1}})$$

ein Vektor von N_i Knotenpunkten (auch Knotenvektor), wobei $s_j \leq s_{j+1}$ für $j = 0 \ldots N_{i-1}$, dann ist die j-te B-Spline Basisfunktion definiert durch:

$$B_j^k(s) = \begin{cases} 1 & \text{falls} \quad s_j \leq s < s_{j+1} \\ 0 & \text{sonst} \end{cases} \tag{B.1}$$

für $k = 1$.

$$B_j^k(s) = \frac{s - s_j}{s_{j+k-1}} B_j^{k-1}(s) + \frac{s_{j+k} - s}{s_{j+k} - s_{j+1}} B_{j-1}^{k-1}(s) \tag{B.2}$$

für $k > 1$.
Uniforme B-Spline Funktionen ergeben sich für $s_j = j$. D.h., dass alle Knotenpunkte gleich verteilt sind, wodurch jede Basisfunktion die gleiche Form hat.

Beispiele für B-Spline Basisfunktionen sind in den Abbildungen B.1 und B.2 mit den Knotenvektoren

$$S_{nonuni} = (0.0,\ 0.0,\ 0.0,\ 1.0,\ 2.0,\ 3.0,\ 4.0,\ 5.0,\ 6.0,\ 6.0,\ 6.0)$$

für die nicht uniformen und

$$S_{uni} = (-2.0,\ -1.0,\ 0.0,\ 1.0,\ 2.0,\ 3.0,\ 4.0,\ 5.0,\ 6.0, 7.0,\ 8.0)$$

für die uniformen Basisfunktionen.

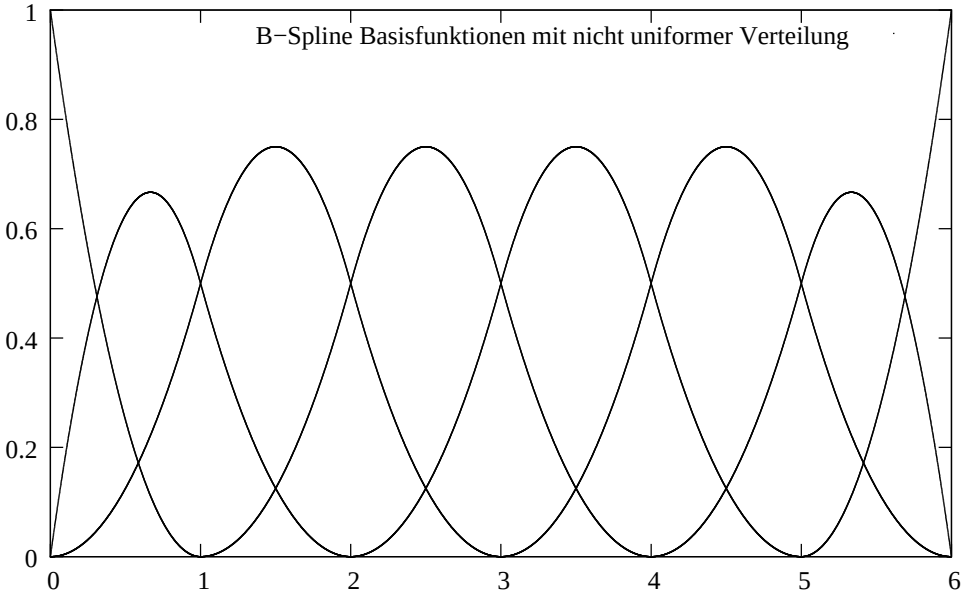


Abbildung B.1: Nicht-uniforme B-Spline Basisfunktionen der Ordnung 3.

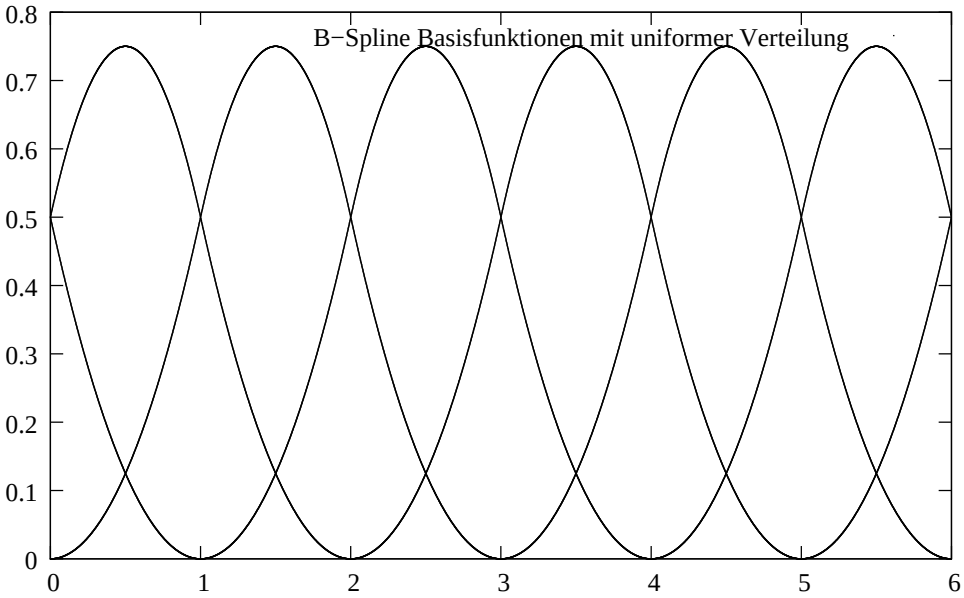


Abbildung B.2: Uniforme B-Spline Basisfunktionen der Ordnung 3.

Die Basisfunktionen besitzen die folgenden Eigenschaften:

- Summe der 1: $\sum_{j=0}^{N_{i-1}} B_j^k(s) = 1$
- Lokaler Support: $B_j^k(s) = 0 \quad \forall s \notin [s_j < s < s_{j+1})$
- Positivität: $B_j^k(s) \geq 0$

- C^{k-2} - Stetig: Wenn alle Kontrollpunkte paarweise verschieden sind, so ist eine B-Spline-Basisfunktion $k-2$ stetig.

Durch multiple Knoten $s_i = s_{i+1} = \ldots$ lässt sich die Stetigkeit gezielt herabsetzen, so dass die Glattheit der Kurve beeinflusst werden kann. Hierdurch wird ebenfalls die Stetigkeit der durch die Basisfunktionen approximierten Funktion in einem Punkt herabgesetzt, so dass sich an dieser Stelle Funktionssprünge oder auch Unstetigkeiten bilden können.

Bild B.3 zeigt Basisfunktionen mit einem multiplen Knoten bei dem Wert 3. Der Knotenvektor zu diesen Basisfunktionen ist gegeben durch:

$$S_{nonuni-break} = (0.0,\ 0.0,\ 0.0,\ 1.0,\ 2.0,\ 3.0,\ 3.0,\ 4.0,\ 5.0,\ 6.0,\ 6.0,\ 6.0)$$

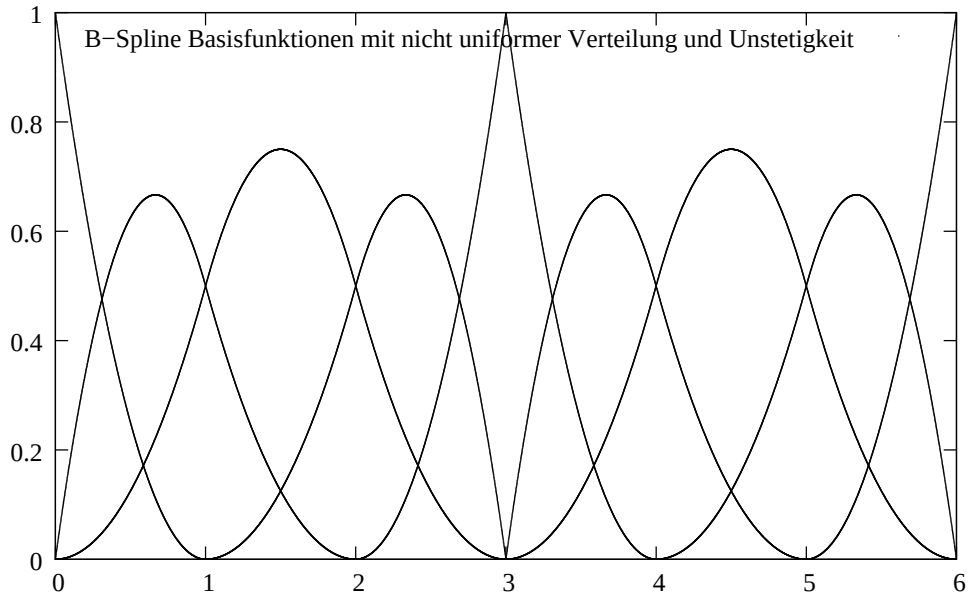


Abbildung B.3: Nicht-uniforme B-Spline Basisfunktionen mit einem doppelten Knoten bei $s = 3$.

B.1.2 B-Spline Funktionen

B-Spline Funktionen setzen sich linear aus den Basisfunktionen zusammen. Für den eindimensionalen Fall ist das:

$$x(s) = \sum_{j=0}^{N_k} q_j \cdot B_j^k(s) \tag{B.3}$$

mit q_j als *Kontrollpunkten* (auch *deBoor* Punkte). Über die deBoor Punkte lässt sich die Form der Kurve festlegen, sie stimmen jedoch nur bei B-Splines der Ordnung zwei mit Punkten auf der zu approximierenden Funktion überein. Diese rekursiv zu berechnen, wie es die Definition aus Gleichung B.1 und B.2 naheliegt, ist sehr aufwendig. Ferch schlägt in [Ferch 2001] vor, die Funktionen bis zur Ordnung drei fest zu kodieren, so dass eine schnelle Berechnung möglich ist.

Im mehrdimensionalen Fall kann eine B-Spline Funktion für Eingabevektor $\vec{s} = (s_1, \ldots, s_n)^T$ durch

$$x(\vec{s}) = \frac{\sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} \left(q_{i_1,\ldots,i_n} \prod_{j=1}^{n} B_{i_j,j}^{k_j}(s_j) \right)}{\sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} \left(\prod_{j=1}^{n} B_{i_j,j}^{k_j}(s_j) \right)}$$

definiert werden. Da der Nenner gleich eins ist gilt

$$= \sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} \left(q_{i_1,\ldots,i_n} \prod_{j=1}^{n} B_{i_j,j}^{k_j}(s_j) \right) \tag{B.4}$$

mit: j als Dimensionsbezeichner, k_j der Ordnung des j-ten B-Splines, m_j der Anzahl der für j definierten Basisfunktionen, i_j dem Index des i-ten B-Spline der Dimension j, $B_{i_j,j}^{k_j}$ der entsprechenden Basisfunktion und $q_{i_1,\ldots,i_n}$ den entsprechenden deBoor Punkten.

B.2 B-Splines als Funktionsapproximator

Ist die Zielfunktion nicht bekannt, und sind nur einige Datenpunkte der Zielfunktion gegeben (wie in Kapitel 2.3), so können B-Splines eingesetzt werden, um diese unbekannte Funktion zu approximieren. Für die Lösung gibt es zwei Möglichkeiten. Die eine besteht in der Berechnung eines Gradientenabstiegs, die andere in einer schrittweisen Optimierung der deBoor Punkte auf der Basis einer Minimierung des quadratischen Fehlers:

$$E = \frac{1}{2}(y_r - y_d)^2. \tag{B.5}$$

Dabei ist y_r der aktuelle Ausgabewert der B-Splinefunktion und y_d der entsprechende Wert der Zielfunktion. Die deBoor-Punkte müssen nun so angepasst werden, dass der Fehler ein Minimum annimmt. Die Änderung der deBoor-Punkte q_j ergibt sich aus der Ableitung der Kostenfunktion nach allen Kontrollpunkten:

$$\delta q_j = \epsilon \frac{\partial E}{\partial q_j} = \epsilon (y_r - y_d) \prod_{j=1}^{n} B_{i,j}^{k}(x_j) \tag{B.6}$$

Beim Einsatz dieser Kostenfunktion ist das Finden eines lokalen Minimums garantiert, da die zweite partielle Ableitung nach q_j grundsätzlich größer Null ist [Zhang and Knoll 1997].

Anhang C

Hauptkomponentenanalyse

C.1 Problem der Objekterkennung

Eine ansichtenbasierte Objekterkennung aus Kamerabildern an Hand von Referenzbildern ist nur schwer möglich. Daher haben sich in der Praxis Verfahren durchgesetzt, bei denen dieses Problem durch eine Transformation der Bilder in einen Subraum gelöst wird. Dieser hat normalerweise eine deutlich geringere Dimensionalität als die Dimension des Bildes, auf dem die Erkennung durchgeführt werden soll. Eines der Verfahren, mit dem eine solche Transformation durchführbar ist die *Karhunen-Loève-* oder auch *Hauptachsentransformation* (*Principal Component Analysis*). Eine Einführung ist in [Smith 2002] zu finden. Alternative Verfahren benutzen z.B. skalierungsunabhängige Merkmale, auch *Scale Invariant Features* (*SIFT*) genannt [Lowe 2004]. Dieses Verfahren hat jedoch den Nachteil, dass ein Objekt ein gewisses Mindestmaß an Textur aufweisen muss, um zuverlässig erkannt zu werden. Dies ist z.B. bei einfarbigen Tassen nicht der Fall. Somit lässt sich das Verfahren für den Anwendungsbereich dieser Arbeit nicht einsetzen, zumindest nicht ohne die Ergebnisse durch ein weiteres Verfahren abzusichern.

Die für diese Arbeit verwendete Objekterkennung stützt sich auf die Hauptachsentransformation. Die Hauptachsentransformation ist ein stochastisches Verfahren. Hierbei wird versucht, die Korrelation bei mehrdimensionalen Daten durch eine Überführung in einen Subraum zu minimieren. Kategorisiert man das Verfahren im Kontext von Lernverfahren, so ist es eine Variante des unüberwachten Lernens. Der große Nachteil des Verfahrens liegt darin, dass es sich nicht generalisieren lässt, sondern für jeden Datensatz neu berechnet werden muss. Dies kann bei großen Datensätzen zu einem erheblichen Aufwand führen.

C.1.1 Hauptachsentransformation

Für eine Hauptachsentransformation werden zunächst einige repräsentative Daten benötigt. Diese werden mittelwertbereinigt und es wird ihre Kovarianzmatrix aufgestellt. Von dieser Matrix werden die Eigenvektoren und die dazugehörigen Eigenwerte bestimmt. Mit Hilfe der Eigenvektoren kann nun einen Subraum aufgespannt werden. Dabei bestimmt die Anzahl der gewählten Eigenvektoren die Dimensionalität des Subraums. Da normalerweise die Anzahl der gewählten Eigenvektoren deutlich kleiner ist als die Dimension der Eingabevektoren, wird eine deutliche Komprimierung des Datenvolumens erzielt. Die Anzahl der Eigenvektoren kann, abhängig von der Problemstellung, frei gewählt werden. Es wird aber oftmals vorgeschlagen, dass man die Anzahl so wählt, dass der Subraum mind. 97% der Informationen der ursprünglichen Daten enthält [Smith 2002].

Sei x_i eines der Eingabebilder über die der Subraum aufgespannt wird, dann wird der Mittelwert über alle Bilder gebildet durch:

$$c = \frac{1}{n} \sum_{i=0}^{n-1} x_i \tag{C.1}$$

Die Kovarianzmatrix Q wird dann aus den mittelwertbereinigten Bildern wie folgt aufgebaut:

$$\begin{aligned} Q &= XX^T \qquad \text{mit} \\ X &= \{x_0 - c, x_1 - c, \ldots, x_{n-1} - c\} \end{aligned} \tag{C.2}$$

Die Eigenwerte der Kovarianzmatrix lassen sich durch folgende Gleichung bestimmen:

$$\begin{aligned} \lambda_i e_i &= Qe_i \qquad \text{mit} \\ \lambda_i &\qquad i = 1 \ldots m \qquad \text{Eigenwerten und} \\ e_i &\qquad i = 1 \ldots m \qquad \text{Eigenvektoren} \end{aligned} \tag{C.3}$$

Da zu jedem Eingabevektor auch ein Eigenvektor aus der Kovarianzmatrix berechnet werden kann, werden die Eigenvektoren nach der Größe der zugehörigen Eigenwerte sortiert, so dass gilt:

$$\lambda_1 > \lambda_2 > \ldots > \lambda_m$$

Somit nimmt der Informationsgehalt mit Index i ab. Um eine Dimensionsreduktion zu erzielen, werden nur die ersten n Eigenvektoren benutzt ($n < m$). Mit m der Dimension eines Eingabevektors, da in ihnen die meiste Information kodiert ist. Die Wahl von n ist wesentlich für eine gute Funktionalität des Systems. Wird n groß gewählt, führt dies zu einem geringen Informationsverlust durch die Projektion in den Subraum, es wird aber auch keine große Dimensionsreduktion erzielt. Wird andererseits n klein gewählt, führt dies unter Umständen zu einem großen Informationsverlust, aber auch zu einer erheblichen Dimensionsreduzierung. Somit werden die nachfolgende Operationen „vereinfacht“, d.h. sie werden in ihrem Berechnungsaufwand reduziert. Um n in einer adequaten Größe zu wählen, gibt es den Vorschlag, dass 97% der Informationen noch nach der Transformation in den Subraum erhalten sein sollen. Dies erfolgt über den folgenden Ansatz:

$$\frac{\sum_{i=1}^{n} \lambda_i}{\sum_{i=1}^{m} \lambda_i} \geq T \tag{C.4}$$

T ist dabei der prozentuale Informationsgehalt des Subraums zum Bildraum ist. Aus den n gewählten Eigenvektoren lässt sich somit eine Transformationsmatrix erstellen, die ein Bild in den Subraum transformiert. Die Transformationsmatrix A ist definiert durch:

$$A = (e_1 \ldots e_n)^T \tag{C.5}$$

Die Transformation in den Eigenraum ist dann definiert als:

$$p_i = A \cdot x_i \qquad dim(p_i) = n \tag{C.6}$$

Natürlich können auch Vektoren aus dem Subraum zurück in den Bildraum transformiert werden. Da A quadratisch und orthogonal ist, gilt:

$$A^{-1} = A^T \tag{C.7}$$

Somit ergibt sich die Rücktransformation durch:

$$x_i = A^{-1} p_i = A^T p_i \tag{C.8}$$

C.1.1.1 Implizite Kovarianz

Die Berechnung der Kovarianzmatrix ist nicht möglich, da Ihre Dimension zu einem enormen Speicherbedarf führt, der auch für einen Rechner mit mehr als 1GB Hauptspeicher nicht zu bewältigen ist. Ein Beispiel: bei der Bildung der Kovarianzmatrix aus Bildern mit der Größe von 384×288 Pixel (1/2 Pal) hat Bildvektor die Dimension t von 110592. Somit wäre die Kovarianzmatrix definiert als:

$$\begin{aligned} Q &= PP^T \qquad Q \in M_{t\times t} \\ &\Rightarrow \quad 110592^2 \approx 12.23 * 10^9 \text{ Elemente} \\ &\Rightarrow \quad \text{bei einem Byte pro Element} \approx 11.4\text{GB} \end{aligned} \tag{C.9}$$

Normalerweise stehen aber nur $M \ll dim(x_i)$ Eingangsdaten zur Verfügung und somit können auch höchstens M unterschiedliche Eigenvektoren gefunden werden. Daher ist es möglich die Berechnungen über die implizite Kovarianzmatrix durchzuführen. Sie ist definiert durch:

$$\tilde{Q} = P^T P \qquad \tilde{Q} \in M_{M\times M} \tag{C.10}$$

Hierbei hängt die Dimension von $\tilde{Q}$ nicht mehr von der Größe der Bilder, sondern von der Anzahl der Eingabedaten ab.

Bei einem Beispiel mit 100 Bildern als Eingabe gilt somit:

$$\begin{aligned} \tilde{Q} &= P^T P \qquad Q \in M_{100\times 100} \\ &\Rightarrow \quad 100^2 = 10000 \text{ Elemente} \\ &\Rightarrow \quad \text{bei einem Byte pro Element} \approx 9,77\text{kB} \end{aligned} \tag{C.11}$$

Hierdurch wird der Speicherbedarf enorm reduziert.

C.1.1.2 Eigenwerte und Eigenvektoren der impliziten Kovarianzmatrix

Die Eigenwerte von Q und ihre korrespondierenden Eigenvektoren lassen sich aus den Eigenwerten und -vektoren von $\tilde{Q}$ berechnen:

$$\begin{aligned} \lambda_i &= \tilde{\lambda}_i \\ e_i &= \tilde{\lambda}_i^{-\frac{1}{2}} P\tilde{e}_i \end{aligned}$$

C.1.2 Klassifikation mittels PCA

Die Klassifikation der Objekte geschieht nach einer Methode, die von Turk und Pentland [Turk and Pentland 1991] für die Klassifikation von Gesichtern entwickelt wurde. Sie haben dafür die Gesichter von 16 unterschiedlichen Personen aufgenommen, diese auf eine Größe von 256×256 Pixel skaliert und anschließend über eine PCA trainiert. Der für die Klassifikation eingesetzte Algorithmus operierte nach dem folgenden Schema:

Zunächst werden die Merkmalsvektoren einer Klasse in den Eigenraum projiziert:

$$\Omega_c = A^T(x_c - \mu) \qquad c = 1, \ldots, k. \tag{C.12}$$

Danach wird der maximale Abstand zwischen zwei Klassen im Eigenraum bestimmt:

$$\theta_l = \frac{1}{2} \max_{j,k} \{\|\Omega_j - \Omega_i\|\} \qquad j, i = 1, \ldots, k. \tag{C.13}$$

Die Klassifikation eines Objektes aus einem Bild (x) erfolgt in drei Schritten

1. Projektion in den Eigenraum:
$$\Omega = A^T(x - \mu) \tag{C.14}$$

2. Klassenabstand bestimmen:
$$\epsilon_c = \|\Omega - \Omega_c\| \tag{C.15}$$

3. Bestimmung des Abstands zwischen Eingabe und Rückprojektion:
$$\epsilon = \|x - x_r\| \qquad \text{mit} \tag{C.16}$$
$$x_r = A\Omega + \mu \tag{C.17}$$

Die Klassifikation der Handkonfiguration aus Kapitel 6.4.3 erfolgt nach dem folgenden Muster:

- Falls $\epsilon \geq \theta_l$
  Das Eingabebild enthält keine Hand.

- Falls $\epsilon < \theta_l$ und $\forall c, \epsilon_c \geq \theta_l$
  Das Eingabebild enthält eine unbekannte Handkonfiguration.

- Falls $\epsilon < \theta_l$ und $\epsilon_{c^*} = \min_c\{\epsilon_c\} < \theta_l$
  Das Eingabebild entspricht Handkonfiguration c^*

Somit lassen sich die Objekte von einander unterscheiden.

Literaturverzeichnis

[Ageia Technologies 2007]
Ageia Technologies
Ageia Technologies Internetseite.
www.ageia.com letzter Aufruf 04.04.2007

[Agilent Technologies]
Agilent Technologies
General Purpose Motion Control IC's - Technical Data HCTL-1100 Series.
http://literature.agilent.com/litweb/pdf/5965-5893E.pdf letzter Aufruf 01.03.2007

[Ahuja et al. 1993]
Ahuja, R.A.; Magnanti, T.L.; Orlin J.B.
Network Flows: Theory, Algorithms, and Applications.
Prentice Hall, 1993

[Anderson et al. 1999]
Anderson, E.; Bai, Z.; Bischof, C.; Blackford, S.; Demmel,J.; Dongarra, J.; Du Croz, J.; Greenbaum, A.; Hammarling, S.; McKenney, A.; Sorensen, D.
LAPACK Users' Guide.
3.rd Edition, Society for Industrial and Applied Mathematics, 1999

[Asfour et al. 2006a]
Asfour, T.; Regenstein, K.; Azad, P.; Schröder, J.; Dillmann, R.
ARMAR-III: A Humanoid Platform for Perception-Action Integration.
International Workshop on Human-Centered Robotic Systems, Munich, 2006

[Asfour et al. 2006b]
Asfour, T.; Regenstein, K.; Azad, P.; Schröder, J.; Bierbaum, A.; Vahrenkamp, N.; Dillmann, R.,
ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control.
International Conference on Humanoid Robots - HUMANOIDS, Genoa, Italy, 2006

[Baier et al. 2006]
Baier, T.; Hüser, M.; Westhoff, D.; Zhang, J.
A flexible software architecture for multi-modal service robots.
Multiconference on Computational Engineering in Systems Applications, Beijing, China, 2006

[Baird and Klopf 1993]
Baird L.; Klopf, A.
Reinforcement learning with high-dimensional, continuous actions.
Technical Report WL-TR-93-1147, Wright-Patterson Air Force Base, Ohio 1993

[Barrett CGL Protocol 2002]
Barrett Technologies Inc.
Grasper Control Language (GCL) Protocol.
Manual, Barrett Technologies Inc., 2002

[Barrett Manual 2002]
Barrett Technologies Inc.
BarrettHand^{TM}- BH8-Series User Manual Firmware Version 4.3x.
Manual, Barrett Technologies Inc., 2002

[Barrett 2007]
Barrett Technologies Inc.
BarrettHandspecifications.
http://www.barrett.com/robot/products/hand/HandDime4.gif letzter Aufruf 06.06.2007

[Barto and Mahadevan 2003]
Barto, A.; Mahadevan, S.
Recent advances in hierarchical reinforcement learning.
Discrete Event Dynamic Systems: Theorie and Applications, vol. 13, pp. 41 – 77, 2003

[Barto et al. 1995]
Barto, A.; Bradke, S.; Singh, S.
Learning to act using real-time dynamic programming.
Artificial Intelligence, Special volume: Computational research on interaction and agency, vol. 72, pp. 81 – 138, 1995

[Becher et al. 2004]
Becer, R.; Seinhaus, P.; Dillmann, R.
Humanoid Robots - Learning and Cooperating Multimodal Robots.
International Journal on Humanoid Robotics, vol. 1, pp. 429 – 448, Karlsruhe, Germany, 2004

[Berg et al. 2000]
de Berg, M.; van Kreveld, M.; Overmars, M.; Schwarzkopf, O.
Compututional Geometry, Algorithms and Applications.
2.nd Edition, Springer Verlag, 2000

[Bertsekas and Tsitsiklis 1996]
Bertsekas, D.P.; Tsitsiklis, J.N.
Neuro-Dynamic Programming.
Athena Scientific, 1996

[Bicchi 1995]
Bicchi, A.
On the closure properties of robotic grasping.
International Journal of Robotics Research, vol. 14, pp. 319 – 334, 1995

[Bicchi 2000]
Bicchi, A.
Hands for Dexterous Manipulation and Powerful Grasping: A Difficult Road Towards Simplicity.
IEEE Transactions on Robotics and Automation, vol. 16, pp. 652 – 662, 2000

[Bicchi and Kumar 2000]
Bicchi, A.; Kumar, V.
Robotic grasping and contact: A review.
IEEE International Conference on Robotics and Automation, San Francisco, California, USA, 2000

[Bischoff and Graefe 2004]
Bischoff, R.; Graefe, V.
HERMES - a Versatile Personal Assistant Robot.
IEEE - Special Issue on Human Interactiv Robots for Psychological Enrichment, pp. 1759 – 1779, 2004

[Bischoff and Gräfe 2002]
Bischoff, R.; Graefe, V
Demonstrating the Humanoid Robot HERMES at an Exhibition: A Long-Term Dependability Test.
IEEE/RSJ International Conference on Intelligent Robots and Systems, Workshop on Robots at Exhibitions, Lausanne, Swiss, 2002

[Bistry et al. 2007]
Bistry, H.; Pöhlsen, S.; Westhoff, D.; Zhang, J.
Development of a Smart Laser Range Finder for an Autonomous Servide Robot.
IEEE International Conference on Integration Technology, Shenzhen, China, 2007

[Bistry und Pöhlsen 2006]
Bistry, H.; Pöhlsen, S.
Entwicklung eines Eingebetteten-Systems zur ressourcenschonenden und plattformunabhängigen Anbindung von SICK-Lasermesssystemen.
Diplomarbeit, Universität Hamburg, Fakultät für Mathematik, Informatik und Naturwissenschaften; Deparment Informatik, Arbeitsbereich Technische Aspekte Multimodaler Systeme, Hamburg, Germany, 2006

[Blackford et al. 2002]
Blackford, L.S.; Demmel, J.; Dongarra, J.; Duff, I.; Hammarling, S.; Henry, G.; Heroux, M.; Kaufman, L.; Lumsdaine, A.; Petitet, A.; Pozo, R.; Remington, K.; Whaley, R.C.
An Updated Set of Basic Linear Algebra Subprograms (BLAS).
ACM Transactions on Mathematical Software, vol. 28, pp. 135 – 151, 2002

[Boutilier et al 1995]
Boutilier, C.; Dearden, R.; Goldszmidt, M.
Exploiting Structure in Policy Construction.
International Joint Conference on Artificial Intelligence, pp. 1104 – 1111, Montreal, Quebec, Canada, 1995

[Borst et al. 2004]
Borst, Ch.; Fischer, M.; Hirzinger, G.
Grasp Planning: How to Choose a Suitable Task Wrench Space.
IEEE International Conference on Robotics and Automation, pp. 319 – 325, New Orleans, LA, USA, 2004

[Borst et al. 2002]
Borst, Ch.; Fischer, M.; Hirzinger, G.
Calculating hand configurations for precision and pinch grasps.
IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, pp. 1553 – 1559, EPFL, Lausanne, Switzerland, 2002

[Borst et al. 1999]
Borst, Ch.; Fischer, M.; Hirzinger, G.
A fast and robust grasp planner for arbitrary 3d objects.
IEEE International Conference on Robotics and Automation, pp. 1890 – 1896, Detroit, Michigan, 1999

[Brafman and Tennenholtz]
Brafman, R.I.; Tennenholtz, M.
R-max - a general polynomial time algorithm for near-optimal reinforcement learning.
Journal of Machine Learning Research, vol. 3, pp. 213 – 231, 2003

[Brown and Lowe 2005]
Brown, M.; Lowe, D.G.
Unsupervised 3D object recognition and reconstruction in unordered datasets.
International Conference on 3-D Digital Imaging and Modeling, pp. 56 – 63, Ottawa, Canada, 2005.

[Buss et al. 1996]
Buss, M.; Hashimoto, H.; Moore, J.B.
Dextrous hand grasping force optimization.
IEEE Transactions on Robotics and Automation, vol. 12, pp. 406 – 418, 1996

[Butterfass et al. 2004]
Butterfass, J.; Fischer, M.; Grebenstein, M.; Haidacher, S.; Hirzinger, G.
Design and Experiences with DLR Hand II.
World Automation Congress, vol. 15, pp. 105 – 110, Seville, Spain, 2004

[Butterfass et al. 2001]
Butterfaß, J.; Grebenstein, M.; Liu, H.; Hirzinger, G.
DLR-Hand II: Next Generation of a Dextrous Robot Hand.
IEEE International Conference on Robotics and Automation, pp. 109 – 114, Seoul, Korea, 2001

[Besl and McKay 1992]
Besl, P.J.; McKay, N.D.
A method for registration of 3-d shapes.
IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, pp. 239 – 256, 1992

[Cary and Bell 1997]
Carey, R.; Bell, G.
The Annotated VRML 2.0 Reference Manual (OpenGL).
Addison-Wesley Professional, 1997

[Camelot 2007]
Camelot Inc.
Camelot Internetseite
www.camelot.dk letzter Aufruf 04.04.2007

[Chen and Burdick 1993]
Chen, I.-M.; Burdick, J.W.
A qualitative test for N-finger force closure grasps on planar objects with applications to manipulation and finger gaits.
IEEE International Conference on Robotics and Automation, pp. 814 – 820, Atlanta, Georgia, USA, 1993

[Cheong and van der Stappen]
Jae-Sook C.; van der Stappen, A.F.
Output-Sensitive Computation of All Form-Closure Grasps of a Semi-Algebraic Set.
IEEE International Conference on Robotics and Automation, pp. 772 – 778, Barcelona, Spain, 2005

[Cohen et al. 2001]
Cohen, E.; Riesenfeld, R.F.; Elber, G.
Geometric Modeling with Splines: An Introduction. B&T, 2001

[von Collani 2001]
von Collani, Y.O.
Repräsentation und Generalisierung von diskreten Ereignisabläufen in Abhängigkeit von Multisensormustern.
Dissertation, Technische Fakultät, Universität Bielefeld, 2001

[CC 2007]
Contemporary Controls GmbH,
Webseite.
http://www.ctrlink.de letzter Aufruf 06.06.2007

[Corke 1996]
Corke, P.L.
A Robotics Toolbox for MATLAB.
IEEE Robotics and Automation Magazine, no. 1, vol. 3, pp. 24 – 32, 1996

[Craig 2005]
Craig, J.J.
Introduction to Robotics: Mechanics and Control.
3.rd Edition, Pearson Education, 2005

[Cutkosky and Wright 1986]
Cutkosky, M.R.; Wright, P.K.
Modeling Manufacturing Grips and Correlations with Design of Robotic Hands,
IEEE International Conference on Robotics and Automation, vol. 3, pp. 1533 – 1539, San Francisco, California, USA, 1986

[Dayan and Hinton 1993]
Dayan, P.; Hinton, G.E.
Feudal Reinforcement Learning.
International Conference on Advances in Neural Information Processing Systems, pp. 271 – 278, Denver, USA, 1993

[Dean et al. 1998]
Dean, T.; Givan, R.; Kim, K.-E.
Solving Stochastic Planning Problems with Large State and Action Spaces.
International Conference on Artificial Intelligence Planning Systems, pp. 102 – 110, Pittsburgh Pennsylvania, USA, 1998

[Dean and Lin 1995]
Dean, T.; Lin, S.-H.
Decomposition Techniques for Planning in Stochastic Domains.
International Joint Conference on Artificial Intelligence, pp. 1121 – 1129, San Francisco, California, 1995

[Dehm 2006]
Dehm, S.H.
Using self-valuing to learn grasping of everyday objects with a multi-fingered hand.
Diploma Thesis, University of Hamburg, Faculty of Mathematics, Informatics and Natural Sciences (MIN); Group of Technical Aspects of Multimodal Systems, Hamburg, Germany, 2006

[D-Link 2007]
D-Link GmbH
D-Link Internetseite.
www.dlink.de letzter Aufruf 06.06.2007

[Delmia 2007]
Delmia GmbH
Delmia Internetseite.
www.delmia.de letzter Aufruf 04.04.2007

[Devmaster 2007]
Devmaster Project
Engine Database.
www.devmaster.net/engines/list.php, letzter Aufruf 04.04.2007

[Dietterich 2000]
Dietterich, T.G.
An Overview of MAXQ Hierarchical Reinforcement Learning.
Lecture Notes in Computer Science, vol. 1864, 2000

[Dietterich 1998]
Dietterich, T.G.
The MAXQ method for hierarchical reinforcement learning.
International Conference on Machine Learning, San Francisco, California, pp. 118 – 126, 1998

[DLR 2005]
DLR
ICRA05 Video Proceedings.
http://www.dlr.de/rm/en/Portaldata/3/Resources/downloads/robutler/dlr-robutler-icra05-wm8-700kbp.wmv letzter Aufruf 06.06.2007

[DLR 2007]
DLR
Internetseite zur DRL HIT Hand.
http://www.dlr.de/rm/en/desktopdefault.aspx/tabid-398/ letzter Aufruf 01.04.2007

[Directed Perception 2007]
Directed Perception Inc.
Internetseite von Directed Perception Inc.
http://www.dperception.com letzter Aufruf 01.04.2007

[Dongarra 2002]
Dongarra, J.
Basic Linear Algebra Subprograms Technical Forum Standard.
International Journal of High Performance Applications and Supercomputing, vol. 16, pp. 1–111, and International Journal of High Performance Applications and Supercomputing, vol. 16, pp. 115–199, 2002

[Easy Rob 2007]
Easy Rob Inc.
Easy Rob Internetseite.
www.easy-rob.com letzter Aufruf 04.04.2007

[Eberly 2004]
Eberly, D.H.
3D Game Engine Architecture.
Morgan Kaufmann Publishers, 2004

[Eberly 2001]
Eberly, D.H.
3D game engine design : a practical approach to real-time computer graphics.
Morgan Kaufmann Publishers, December 2001

[Eberly 2003]
Eberly, D.H.
Game Physics.
Morgan Kaufmann Publishers, 2003

[Eberly 2002]
Eberly, D.H.
Dynamic Collision Detection using Oriented Bounding Boxes.
www.geometrictools.com, December 2002

[Energid 2007]
Energid Inc.
Energid Internetseite.
www.energid.com letzter Aufruf 04.04.2007

[Ferch 2001]
Ferch, M.C.
Lernen von Montagestrategien in einer verteilten Multiroboterumgebung.
Dissertation, Universität Bielefeld, Technische Fakultät, Bielefeld, Germany, 2001

[Ferrari and Canny 1992]
Ferrari, C.; Canny, J.
Planning Optimal Grasps.
IEEE International Conference on Robotics and Automation, pp. 2290 – 2295, Nice, France, 1992

[Fishman 1999]
Fishman, G.S.
Monte Carlo : concepts, algorithms, and applications.
Corrected 3. printing, Springer, 1999

[Flow Software Technologies 2007]
Flow Software Technologies Inc.
Worksapce 5 Internetseite.
www.workspace5.com letzter Aufruf 04.04.2007

[Foley et al. 1996]
Foley, J.D.; van Dam, A.; Feiner, S.K.; Hughes, J.F.
Computer Graphics - Principles and Practice. 2.nd Edition, Addison Wesley, 1996

[Frauenhofer 2005]
Fraunhofer-Institut für Produktionstechnik und Automatisierung
Care-O-Bot - Produktdatenblatt.
http://www.care-o-bot.de/Produktblaetter/Produktblatt_Care-O-bot.pdf letzter Aufruf 05.06.2007

[Frauenhofer 2007]
Fraunhofer-Institut für Produktionstechnik und Automatisierung
Care-O-Bot.
http://www.care-o-bot.de letzter Aufruf 05.06.2007

[Fujita and Kageyama 1997]
Fujita M.; Kageyama, K.
An Open Architecture for Robot Entertainment.
International Conference on Autonomous Agents, pp.435 – 442, Marina Del Rey, California, USA, 1997

[Fujita and Kitano 1998]
Fujita, M.; Kitano, H.
Development of an Autonomous Quadruped Robot for Robot Entertainment.
Autonomous Robots, 5, pp. 7 – 20, Springer, Boston, USA, 1998

[Fujita et al. 2003a]
Fujita, M.; Kuroki, Y.; Ishida, T.; Doi, T. T.
SDR-4X II: A Small Humanoid as an Entertainer in Home Environment.
International Symposium of Robotics Research, Siena, Italy, 2003

[Fujitsu 2007]
Fujitsu Corporation
Humanoid Robots.
http://jp.fujitsu.com/group/automation/en/services/humanoid-robot/ letzter Aufruf 22.3.2007

[genRob 2007]
genRob GmbH
genRob Internetseite.
http://www.genrob.com letzter Aufruf 06.06.2007

[Geometrictools 2007]
Geometrictools Inc.
Geometrictools Internetseite.
www.geometrictools.com letzter Aufruf 23.04.2007

[Glorennec 2000]
Glorennec, P.Y.
Reinforcement Learning: an Overview.
European Symposium on Intelligent Techniques, pp. 17 – 35, Aachen, Germany, 2000

[Gorce and Rezzoug 2004]
Gorce, P.; Rezzoug, N.
A method to learn hand grasping posture from noisy sensing information.
Robotica, vol. 22, issue 3, pp. 309 – 318, Pavilhao Rosa Mota, Spain, 2004

[Gottschalk et al. 1996]
Gottschalk, S.; Lin, M. C.; Manocha, D.
OBB-Tree: A Hierarchical Structure for Rapid Interference Detection.
Technical report TR96-013, Department of Computer Science, University of N. Carolina, Chapel Hill, USA, 1996

[Graf 2000]
Graf, T.
Flexible object recognition based on invariant theory and agent technology,
Dissertation, Technische Fakultät, Universität Bielefeld, Bielefeld, Germany, 2000

[Han et al. 2000]
Han, L.; Trinkle, J.C.; Li, Z.
Grasp Analysis as Linear Matrix Inequality Problems.
IEEE Transactions on Robotics and Automation, vol. 16, pp. 663 – 674, 2000

[Hans et al. 2004]
Hans, M.; Graf, B.
Robotic Home Assistant Care-O-bot II.
Advances in Human-Robot Interaction. Series : Springer Tracts in Advanced Robotics (Eds. Prassler, E.; Lawitzky, G.; Stopp, A.; Grunwald, G.; Hägele, M.; Dillmann, R.; Iossifidis, I.), vol. 14, pp. 371 – 384, 2004

[Haschke et. al. 2005]
Haschke, R.; Steil, J.J.; Steuwer, I.; Ritter, H.
Task-Oriented Quality Measures for Dextrous Grasping.
IEEE International Conference on Computational Intelligence in Robotics and Automation, Espoo, Finland, 2005

[Hauskrecht et al. 1998]
Hauskrecht, M.; Meuleau, N.; Kaelbling, L.P.; Dean, T.; Boutilier, C.
Hierarchical Solution of Markov Decision Processes using Macro-actions.
International Conference on Uncertainty in Artificial Intelligence, pp. 220 – 229, San Francisco, California, 1998

[Hilleenbrand et al. 2004]
Hillenbrand, U.; Ott, C.; Brunner, B.; Borst, C.; Hirzinger, G.:
Towards Service Robots for the Human Environment: the Robutler.
IEEE International Mechatronics & Robotics Conference, Aachen, Germany, 2004

[Honda 2007a]
Honda Croporation
ASIMO - The Honda humanoid Robot.
http://world.honda.com/ASIMO/ letzer Aufruf 22.3.2007.

[Honda 2007b]
Honda Croporation
ASIMO - Technology.
http://world.honda.com/ASIMO/technology/concept.html letzter Aufruf 22.3.2007

[Hübner et al. 2006]
Hübner, K.; Westhoff, D.; Zhang, J.
A Comparison of Regional Feature Detectors in Panoramic Images.
IEEE International Conference on Information Aquisition, pp. 666 – 671, Weihai, China, 2006

[Hübner 2006]
Kübner, K.
Symmetriesignaturen für bildbasierte Anwendungen in der Robotik.
Dissertation, Universität Bremen, Fachbereich Mathematik und Informatik, Bremen, Germany, 2006

[Hüser 2008]
Hüser, M.
Programmierung von Manipulationsfertigkeiten durch natürliche Demonstration für interaktive Serviceroboter.
Dissertation, Universität Hamburg, Fachbereich für Mathematik, Informatik und Naturwissenschaften, Hamburg, Germany, noch nicht veröffentlicht

[Hüser et al. 2006]
Hüser, M.; Baier, T.; Zhang, J.
Learning of demonstrated Grasping Skills by stereoscoopic tracking of human hand configuration.
IEEE International Conference on Robotics and Automation, Orlando, Florida, USA, 2006

[Hüser et al. 2006]
Hüser, M.; Baier, T.; Westhoff, D.; Zhang, J.
Multimodal learning of demonstrated grasping skills for flexibly handling grasped objects.
ISR/Robotik 2006, VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, Munich, Germany, 2006

[HVBG 1995]
Hauptverband der gewerblichen Berufsgenossenschaften
BGR 195 - Einsatz von Schutzhandschuhen.
Hauptverband der gewerblichen Berufsgenossenschaften, Berufsgenossenschaftliche Regeln für Sicherheit und Gesundheit bei der Arbeit, aktualisierte Fassung, 1995

[ICP 2007]
Industrial Computer Products Deutschland
Industrial Computer Products Deutschland Internetseite.
http://www.icp-deutschland.de letzter Aufruf 06.06.2007

[IFR 2006]
IFR - Statistical Department
World Robotics 2006 - Executive Summary.
http://www.worldrobotics-online.org/downloads/2006_Executive_Summary(1).pdf letzter Aufruf 06.06.2007

[Infratest 2004]
TNS Infratest
Horizons 2020 - Ein Szenario als Denkanstoß für die Zukunft.
TNS Infratest, 2004
http://w3.siemens.de/horizons2020/index.htm letzter Aufruf 06.06.2007

[Ishida et al. 2003]
Ishida, T.; Kuroki, Y.; Yamaguchi, J.
Development of Mechanical System for a Small Biped Entertainment Robot.
IEEE International Workshop on Robot and Human Interactive Communication, pp. 297 – 302, 2003

[Jaakkola et al. 1994]
Jaakkola, T.S.; Jordan, M.I.; Singh, S.P.
Convergence of Stochastic Iterative Dynamic Programming Algorithms.
Advances in Neural Information Processing Systems, vol. 6, pp. 703 – 710, 1994

[JAI 2007]
JAI A/S
JAI Internetseite.
http://www.jai.com letzter Aufruf 02.03.2007

[JAI 2002]
JAI A/S
Micro-head Color Camera Series - CV-M2000 Series Operation Manual.
Manual, Denmark 2002

[Jockel et al. 2007]
Jockel, S.; Baier-Löwenstein, T.; Zhang, J.
Three-Dimensional Monocular Scene Reconstruction for Service-Robots: An Application.
International Conference on Computer Vision Theory and Applications, Barcelona, Spain, 2007

[Jonsson and Barto 2005]
Jonsson, A.; Barto, A.
A causal approach to hierarchical decomposition of factored MDPs.
International Conference on Machine learning, pp. 401 – 408, Bonn, Germany, 2005

[Kaebling et al. 1998]
Kaelbling, L.P.; Littman, M.L.; Cassandra, A.R.
Planning and Acting in Partially Observable Stochastic Domains.
Artificial Intelligence, vol. 101, pp. 99 – 134, 1998

[Kaelbling et al 1996]
Kaelbling, L.P.; Littman, M.L.; Cassandra, A.R.
Reinforcement Learning: A Survey.
Journal of Artificial Intelligence Research, vol. 4, pp. 237 – 285, 1996

[Kargov et al. 2006]
Kargov, A.; Pylatiuk, C.; Schulz, S.; Bretthauer, G.
Modularly designed lightweight anthropomorphic robot hand.
International Conference on Multisensor Fusion and Integration for Intelligent Systems, Heidelberg, Germany, pp.155 – 159, 2006

[Keerthi and Ravindran]
Keerthi, S.; Ravindran, B.
A tutorial survey of reinforcement learning.
Sadhana (published by the Indian Academy of Sciences), 1995

[Kerr and Roth 1986]
Kerr, J.; Roth, B.
Analysis of Multifingered Hands.
International Journal of Robotics Research, vol. 4, no. 4, pp. 3 – 17, 1986

[Kim et al. 2004a]
Kim, J.; Park, J.; Hwang, Y.; Lee, M.
Advanced Grasp Planning for Handover Operation Between Human and Robot: Three Handover Methods in Esteem Etiquettes Using Dual Arms and Hands of Home Service Robot.
International Conference on Autonomous Robots and Agents, pp. 34 – 39, Palmerston North, New Zeeland, 2004

[Kim et al. 2004b]
Kim, B.-H.; Yi, B.-J.; Oh, S.-R.; Suh, I.H.
Non-dimensionalized performance indices based optimal grasping for multi-fingered hands.
Mechatronics, vol. 14, no. 3, pp. 255 – 280, 2004

[Kirkpatrick et al. 1990]
Kirkpatrick, D. G.; Mishra, B.; Yap, C.K.
Quantitative Steinitz's theorems with applications to multifingered grasping.
ACM symposium on Theory of computing, pp. 341 – 351, , Baltimore, Maryland, USA, 1990

[Kohonen 1997]
Kohonen, T.
Self-Organizing Maps.
Springer, Berlin, Germany, 1997

[Kragic et al. 2001]
Kragic, D.; Miller, A.T.; Allen, P.K.
Real-time tracking meets online grasp planning.
International Conference on Robotics and Automation, pp. 2460 – 2465, Seoul, Republic of Korea, 2001

[Krüger and Peters 2000]
Krüger, N.; Peters, G.
ORASSYL: Object Recognition with Autonomously Learned and Sparse Symbolic Representations Based on Metrically Organized Local Line Detectors.
Computer Vision and Image Understanding, vol. 77, no. 1, pp. 48 – 77, 2000

[Kruse et al. 1995]
Kruse, R.; Gebhardt, J.; Klawonn, F.
Fuzzy-Systeme.
Serie Leitfäden und Monographien der Informatik. Teubner Verlag, 2. Auflage, Stuttgart, 1995
http://fuzzy.cs.uni-magdeburg.de/studium/fuzzy/txt/fsbook.pdf,
letzter Aufruf 10.06.2007

[Li and Sastry 1987]
Li, Z.; Sastry, S.
Task Oriented Optimal Grasping by Multifingered Robot Hands.
IEEE Conference on Robotics and Automation, pp. 389 – 394, Raleig, North Carolina, 1987

[Lide 2005]
Lide, D.R. (Editor-in-Chief)
CRC Handbook of Chemistry and Physics.
Chemical Rubber Company, Cleveland, 86 edition, Ohio, 2005-2006

[Liu 1999]
Liu, Y.H.
Qualitative test and force optimization of 3-D frictional form-closure grasps using linear programming.
IEEE Transaction on Robotics and Automation, vol. 15, pp. 163 – 173, 1999

[Liu and Lam 2003]
Lui, Y.; Lam, M.
Searching 3-D Form Closure Grasps in Discrete Domain.
IEEE International Conference on Intelligent Robots and System, vol. 4, pp. 3711 – 3716 , Las Vegas, Nevada, 2003

[Liu and Li 2004]
Liu, G.; Li, Z.
Real-time grasping-force optimization for multifingered manipulation: theory and experiments.
IEEE/ASME Transactions on Mechatronics, vol. 9, no. 1, pp. 65 – 77, 2004

[Liu et al. 2004a]
Liu, G.; Xu, J.; Li, Z.
On Geometric Algorithms for Real-Time Grasping Force Optimization.
IEEE Transactions on Control Systems Technology, vol. 12, no. 6, pp. 843 – 859, 2004

[Liu et al. 2004b]
Liu, G.; Xu, J.; Wang, X.; Li, Z.
On Quality Functions for Grasp Synthesis, Fixture Planning, and Coordinated Manipulation.
IEEE Transactions on Automation Science and Engineering, vol. 1, no. 2, pp. 146 – 162, 2004

[LLoyd 1998]
Lloyd, J.E.
Trajectory Generation Implemented as a Non-linear Filter.
Technical Report 98-11, Department of Computer Science, University of British Columbia Vancouver, B.C., Canada, 1998

[LLoyd and Hayward 1992]
Lloyd, J.E.; Hayward, V.
Multi-RCCL User's Guide.
Montreal, Qu'ebec, Canada, 1992
http://www.cs.ubc.ca/~lloyd/rccl/rcclGuide.ps.Z letzter Aufruf 01.03.2007

[Lovchik and Diftler 1999]
Lovchik, C.; Diftler, M.
The Robonaut Hand: A Dexterous Robot Hand For Space.
IEEE International Conference on Automation and Robotics, vol. 2, pp. 907 – 912, Detroit, Michigan, USA, 1999

[Lovchik et al. 1999]
Lovchik, C.; Aldridge, H.; Diftler, M.
Design of the NASA Robonaut Hand.
ASME Dynamics and Control Division, DSC-Vol. 67, pp. 813 – 830, Nashville, Tennessee, USA, 1999

[Lowe 2004]
Lowe, D.G.
Distinctive image features from scale-invariant keypoints
International Journal of Computer Vision, vol. 60, no. 2, pp. 91 – 110, 2004

[Makar et al. 2001]
Makar, R.; Mahadevan, S.; Ghavamzadeh, M.
Hierarchical multi-agent reinforcement learning.
International Conference on Autonomous Agents, pp. 246 – 253, Montreal, Canada, 2001

[Mason et al.]
Woo, M.; Nieder, J.; Davis, T.; Shreiner, D.
OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.4 .
4.th edition, Addison-Wesley Publishing Company, 2004

[Matrox 2007]
Matrox Electronic Systems GmbH
Matrox Internetseite.
http://www.matrox.com letzter Aufruf 06.06.2007

[McGovern and Barto 2001a] **McGovern, A.; Barto, A.G.**
Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density.
International Conference on Machine Learning, pp. 361 – 368, San Francisco, California, USA, 2001

[McGovern and Barto 2001b]
McGovern, A.; Barto, A.G.
Accelerating Reinforcement Learning through the Discovery of Useful Subgoals.
International Symposium on Artificial Intelligence, Robotics and Automation in Space: i-SAIRAS, Montreal, Canada, 2001

[McGovern et al. 1998]
McGovern, A.; Precup, D.; Ravindran, B.; Singh, S.; Sutton, R.S.
Hierarchical Optimal Control of MDPs.
Yale Workshop on Adaptive and Learning Systems, pp. 186 – 191, 1998

[McGovern and Sutton 1998]
McGovern, A.; Sutton, R.S.
Macro-Actions in Reinforcement Learning: An Emperical Analysis.
University of Massachusetts, Amherst, Technical Report Number 98-70, 1998

[McGovern et al. 1997]
McGovern, A.; Sutton, R.S.; Fagg, A.H.
Roles of macro-actions in accelerating reinforcement learning.
Grace Hopper Celebration of Women in Computing, pp. 13 – 17, San Jose, California, 1997

[Menache et al. 2002]
Menache, I.; Mannor, S.; Shimkin, N.
Q-Cut - Dynamic Discovery of Sub-Goals in Reinforcement Learning.
Lecture Notes In Computer Science, vol. 2430, European Conference on Machine Learning, pp. 295 – 306, Helsinki, Finland, 2002

[MHI 2007]
Mitsubishi Heavy Industries
PA 10 Internetseite.
http://www.mhi.co.jp/kobe/mhikobe-e/products/mechatronic/e_index.html letzter Aufruf 06.06.2007

[MHI 2004]
Mitsubishi Heavy Industries
PA 10 Series Manual.
Mitsubishi Heavy Industries, 2004

[Microsoft 2007]
Microsoft Corporation
Microsoft Robotics Internetseite.
msdn.microsoft.com/robotics/ letzter Aufruf 04.04.2007

[Miller 2005]
Miller, A.; Allen, P.; Santos, V.; Valero-Cuevas, F.
From robotic hands to human hands: a visualization and simulation engine for grasping research .
Industrial Robot, Intelligent manipulation and grasping, vol. 32, no. 1, pp. 55 – 63, 2005

[Miller 2001]
Miller, A.T.
GraspIt!: A Versatile Simulator for Robotic Grasping.
Ph.D. Thesis, Department of Computer Science, Columbia University, Columbia, USA, 2001

[Miller and Allen 2000]
Miller, A.T.; Allen, P.K.
GraspIt!: A Versatile Simulator for Grasp Analysis.
ASME International Mechanical Engineering Congress & Exposition, Orlando, Florida, USA, pp. 1251 – 1258, 2000

[Miller and Allen 1999]
Miller, A.T.; Allen, P.K.
Examples of 3D Grasp Quality Computations.
IEEE International Conference on Robotics and Automation, pp. 1240 – 1246, Detroit, Michigan, USA, 1999

[Miller et al. 2003]
Miller, A.T.; Knoop, S.; Allen, P.K.; Christensen, H.I.
Automatic Grasp Planning Using Shape Primitives.
IEEE International Conference on Robotics and Automation, pp. 1824 – 1829, Taipei, Taiwan, 2003

[Mirtich and Canny 1994]
Mirtich, B.; Canny, J.
Easily Computable Optimum Grasps in 2-D and 3-D.
IEEE International Conference on Robotics and Automation, pp. 739 – 747, San Diego, California, USA, 1994

[Mishra 1995a]
Mishra, B.
Grasp Metrics: Optimality and Complexity.
Algorithmic Foundations of Robotics, pp. 137–166, Wellesley, Massachusetts, USA, 1995

[Mishra 2000]
Mishra, B.
On the Other Hands: Geometric Ideas in Robotics.
Geometry at Work: Papers in Applied Geometry, (Eds. C.A. Gorini et al.), vol. 53, pp. 105 – 117, Cambridge, England, 2000

[Moore 1965]
Moore, G.E.
Cramming More Components Onto Integrated Circuits.
Electronics, 1965

[Morales et al. 2006a]
Morales, A.; Azad, P.; Asfour, T.; Kraft, D.; Knoop, S.; Dillmann, R.; Kargov, A.; Pylatiuk, Ch.; Schulz, S.
An antropomorphic grasping approach for an assistant humanoid robot.
International Symposium of Robotics, Munich, Germany, 2006

[Morales et al. 2006b]
Morales, A.; Asfour, T.; Azad, P.; Knoop, S.; Dillmann, R.
Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands.
International Conference on Intelligent Robots and Systems, pp. 5663 – 5668, Beijing, China, 2006

[Morpha 2007]
MORPHA Konsortium
MORPHA - Kommunikation, Interaktion und Kooperation zwischen Menschen und intelligenten antropomorphen Assistenzsystemen.
www.morpha.de letzter Aufruf 04.05.2007

[Murase and Nayar 1993]
Murase, H.; Nayar, S.K.
Learning and recognition of 3D objects from appearance.
IEEE Qualitative Vision Workshop, pp. 39 – 50, New York, New York, USA, 1993

[Murray et al. 1994]
Murray, R.M.; Li, Z.; Sastry, S.S.
A Mathematical Introduction to Robotic Manipulation.
CRC Press, 2004.

[NASA 2007]
NASA
Internetseiten zur Robonaut Hand
http://robonaut.jsc.nasa.gov/hands.htm letzter Aufruf 01.04.2007

[Neobotix 2007]
Neobotix GmbH
Neobotix Internetseite
http://www.neobotix.de letzter Aufruf 06.06.2007

[Nguyen 1988]
Nguyen, V.-D.
Constructing force-closure grasps
International Journal of Robotics Research, vol. 7, no. 3, pp. 3 – 16, 1988

[Nguyen 1986]
Nguyen, V.-D.
Constructing stable force-closure grasps.
ACM Fall joint computer conference, pp. 129 – 137, Dallas, Texas, USA, 1986

[Niparnan and Sudsang 2004]
Niparnan, N.; Sudsang, A.
Fast Computation of 4-Fingered Force-Closure Grasps from Surface Points.
IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 4, pp. 3692 – 3697, Sendai, Japan, 2004

[OECD 2004]
OECD
TECHNOLOGY TRENDS - PREMIMINARY REPORTS; Project on The Commercialisation of Space and the Development of Space Infrastructure: The Role of Public and Private Actors.
OCED, SG/AU/SPA(2004)2, 2004

[OECD 1998]
OECD
21st Century Technologies - Promises and Perils of a Dynamic Future
OECD, 1998
http://www.oecd.org/dataoecd/41/16/35391210.pdf letzter Aufruf 24.03.2007

[Okamura et al. 2000]
Okamura, A.M.; Smaby, N.; Cutkosky, M.R.
An Overview of Dexterous Manipulation.
IEEE International Conference on Robotics and Automation, Symposium on Dexterous Manipulation, Vol. 1, pp. 255 – 262, San Francisco, California, USA, 2000

[Ott et al. 2005]
Ott, C.; Borst, C.; Hillenbrand, U.; Brunner, B.; Bäuml, B.; Hirzinger, G.,
The Robutler: Towards Service Robots for the Human Environment.
IEEE International Conference on Robotics and Automation, pp. 1399 – 1405, Barcelona, Spain, 2005

[Parr and Russell 1997]
Parr, R.; Russell, S.
Reinforcement Learning with Hierarchies of Machines.
Advances in Neural Information Processing Systems, vol. 10, The MIT Press, 1997

[Pelossof et al. 2004]
Pelossof, R.; Miller, A.; Allen, P.; Jebara, T.
An SVM learning approach to robotic grasping.
IEEE International Conference on Robotics and Automation, pp. 3215 – 3218, New Orleans, LA, USA, 2004

[Peters 2004]
Peters, G.,
Efficient Pose Estimation Using View-Based Object Representations.
Machine Vision and Applications, vol. 16, no. 1, pp. 59 – 63, 2004

[Pickett and Barto 2002]
Pickett, M.; Barto, A.G.
PolicyBlocks: An Algortithm for Creating Useful Macro-Actions in Reinforcement Learning.
International Conference of Machine Learning, pp. 506 – 513, Sydney, Australia, m2002

[Pollard 2004]
Pollard, N.
Closure and Quality Equivalence for Efficient Synthesis of Grasps from Examples.
International Journal of Robotics Research, vol. 23, no. 6, pp. 595 – 614, 2004

[Pollard 1997]
Pollard, N.
Parallel algorithms for synthesis of whole hand grasps.
IEEE International Conference on Robotics and Automation, vol 1, pp. 373 – 378, Albuquerque, New Mexico, USA, 1997

[Pollard 1996]
Pollard, N.
Synthesizing grasps from generalized prototypes.
IEEE International Conference on Robotics and Automation, vol.3, pp. 2124 – 2130, Minneapolis, Minnesota, USA, 1996

[Pollard 1994]
Pollard, N.
Parallel Methods for Synthesizing Whole-Hand Grasps from Generalized Prototypes.
MIT Artificial Intelligence Laboratory, Technical Report AI-TR 1464, 1994

[Pollard and Wolf 2005]
Pollard, N.; Wolf, A.
Grasp Synthesis from Example: Tuning the Example to a Task or Object.
Multi-point Interaction with Real and Virtual Objects, (Eds. Federico Barbagli, Domenico Prattichizzo, and Kenneth Salisbury), Springer Tracts in Advanced Robotics, pp. 77 – 90, 2005

[Pollefeys et al. 2004]
Pollefeys, M.; Van Gool, L.; Vergauwen, M.; Verbiest, F.; Cornelis, K.; Tops, J.; Koch, R.
Visual modeling with a hand-held camera,
International Journal of Computer Vision, vol. 59, no. 3, pp. 207 – 232, 2004

[Prattichizzo and Bicchi 1998]
Prattichizzo, D.; Bicchi, A.
Dynamic analysis of mobility and graspability of general manipulation systems.
IEEE Transactions on Robotics and Automation, vol. 14, no. 2, pp. 241 – 257, 1998

[Precup 200]
Precup, D.
Temporal abstraction in reinforcement learning.
Ph.D. Thesis, University of Massachusetts, Massachusetts, Amherst, USA, 2000

[Puterman 1994]
Puterman, M.L.
Markov Decision Processes: Discrete Stochastic Dynamic Programming.
John Wiley and Sons, New York, NY, 1994

[Redmond et al. 2001]
Remond, C.; Perdereau, V.; Drouin, M.
A Multi-fingered Hand Control Structure with On-line Grasping Force Optimization.
IEEE/ASME International Conference on Advanced Intelligent Mechatronics, vol.2, pp. 804 – 809, Como, Italy, 2001

[Rehg and Kanade 1994]
Rehg, J.M.; Kanade, T.
Visual Tracking of High DOF Articulated Structures: an Application to Human Hand Tracking.
European Conference on Computer Vision, pp. 35 – 46, Stockholm, Sweden, 1994

[Rössler et al. 2002]
Rössler, B.; Zhang, J.; Knoll, A.
Visual guided grasping of aggregates using self-valuing learning.
IEEE International Conference on Robotics and Automation, pp. 3912 – 3917, Washington, DC, USA, 2002

[Rössler 2001]
Rössler, B.
Visual Guided Grasping of Aggregates using Self-Valuing Learning.
Diplomarbeit, Universität Bielefeld, Technische Fakultät, 2001

[Rummery and Niranjan 1994]
Rummery, G.; Niranjan, M.
On-line q-learning using connectionist systems.
Technical Report CUED/F-INFENG/TR 166, Cambridge University, Engineering Department, 1994

[Russel und Norvig 2003]
Russel, S.; Norvig, P.
Künstliche Intelligenz - Ein moderner Ansatz.
2.Auflage, Pearson Studium, 2003

[Sallans and Hinton 2004]
Sallans, B.; Hinton, G.E.
Reinforcement Learning with Factored States and Actions.
Journal of Machine Learning Research, vol. 5, pp. 1063 – 1088, 2004

[Salomon 2006]
Salomon, D.
Curves and Surfaces for Computer Graphics.
Springer Science+Business Media Inc., 2006

[Scherer 2004]
Scherer, T.
A Mobile Service Robot for Automatisation of Sample Taking and Sample Management in a Biotechnological Pilot Laboratory.
Ph.D. Thesis, University of Bielefeld, Faculty of Technology, Bielefeld, Germany, 2004

[Schneider and Westhoff 2002]
Schneider, A.; Westhoff, D.
Autonomous Navigation and Control of a Mobile Robot in a Cell Culture Laboratory.
Diplomarbeit, Universität Bielefeld, Technische Fakultät, Bielefeld, Germany, 2002

[Sedgewick 2002]
Sedgewick, R.
Algorithmen in C++.
3. Auflage, Pearson Studium, 2002

[SFB-588 2007]
SFB-588 Humanoide Roboter - Lernende und kooperierende multimodale Roboter.
SFB-588 Internetseite
SFB-588, 2007
http://www.sfb588.uni-karlsruhe.de/index.html letzter Aufruf 01.04.2007

[Shadow 2003]
Shadow Company
Design of a Dextrous Hand for advanced CLAWAR applications.
Shadow Robot Company, 2003

[Shadow 2007]
Shadow Company
Shadow Hand Internetseite.
http://www.shadowrobot.com/hand/ letzter Aufruf 01.04.2007

[Shreiner 2004]
Shreiner, D.
OpenGL Reference Manual: The Official Reference Document to OpenGL, Version 1.4.
4.th edition, Addison-Wesley Publishing Company, 2004

[SGI 2007]
Silicon Graphics
OpenInventor Internetseite.
http://oss.sgi.com/projects/inventor/ letzter Aufruf 10.06.2007

[SGI 1994]
Silicon Graphics
Open Inventor: How to Write an Open Inventor File Translator.
SGI, Document number : 007-2468-001, 1994
http://techpubs.sgi.com/library/manuals/2000/007-2468-001/pdf/007-2468-001.pdf letzter Aufruf 10.06.2007

[Şimşek et al. 2005]
Şimşek, Ö.; Wolfe, A.P.; Barto, A.G.
Identifying useful subgoals in reinforcement learning by local graph partitioning.
International conference on machine learning, pp. 816 – 823, Bonn, Germany, 2005

[Şimşek and Barto 2004]
Şimşek, Ö.; Barto, A.
Using relative novelty to identify useful temporal abstractions in reinforcement learning.
International conference on Machine learning, pp. 95 – 103, Banff, Alberta, Canada, 2004

[Smith 2002]
Smith, L.I.
A tutorial on Principal Component Analysis.
Lindsay I. Smith, 2002
http://csnet.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf letzter Aufruf 01.05.2007

[Sony 2004]
SONY Corporation
OPEN-R SDK - Model Informatoin for ERS-7.
SONY Corporation 2004
http://www.openr.aibo.com letzter Aufruf 04.05.2006

[Sony 2006]
SONY Corporation
Aibo Europe.
http://www.aibo-europe.com letzter Aufruf 30.06.2006.

[Sony 2007]
SONY Corporation
QRIO.
http://www.sony.net/SonyInfo/QRIO/ letzter Aufruf 22.3.2007.

[Speck und Klaeren 1999]
Speck, A.; Klaeren, H.
RoboSiM: Java 3D Robot Visualization.
Conference of the IEEE Industrial Electronics Society, pp. 821 – 826, Aachen, Germany, 1999

[Sutton 1996]
Sutton, R.S.
Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding“.
Advances in Neural Information Processing Systems, vol. 8, pp. 1038–1044, 1996

[Sutton 1988]
Sutton, R.S.
Learning to Predict by the Methods of Temporal Differences.
Machine Learning, vol. 3, pp. 9 – 44, 1988

[Sutton and Barto 1998]
Sutton, R.S.; Barto, A.G.
Reinforcement Learning: An Introduction.
MIT Press, Cambridge, MA, 1998

[Sutton et al. 1999]
Sutton, R.S.; Precup, D.; Singh, S.
Between MDPs and semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning.
Artificial Intelligence 112, pp. 181 – 211, 1999

[Thrun 1992]
Thrun, S.B.
Efficient Exploration in Reinforcement Learning.
Technical Report MU-CS-92-102, Pittsburgh, Pennsylvania, USA, 1992

[Thrun and Möller 1992]
Thrun, S.B.; Möller, K.
Active Exploration in Dynamic Environments.
Advances in Neural Information Processing Systems, vol. 4, pp. 531 – 538, 1992

[Townsend 2000]
Townsend, W.T.
The BarrettHand grasper - programmably flexible part handling and assembly.
Industrial Robot: An International Journal, vol. 27, no. 3, pp. 181 – 188, 2000

[Trisk 2007]
TRISK Project
TRISK Project Internetseite.
http://www.media.mit.edu/cogmac/projects/trisk.html letzter Aufruf 02.04.2007

[Turk and Pentland 1991]
Turk, M.; Pentland, A.
Eigenfaces for recognition.
Journal of Cognitive Neuroscience, vol. 3, no. 1, pp. 71 – 86, 1991

[Unvin et al. 2006]
Unvin, A.; Theus, M.; Hoffman, H.
Graphics of Large Datasets-Visualizing a Million
Springer, 2006

[van den Bergen 2003]
van den Bergen, G.
Collision Detection in Interactive 3D Environments.
Playlogic Game Factory, Breda, The Netherlands, Morgan Kaufmann Publishers, 2003

[van Meerbergen et al 2002]
G. Van Meerbergen, M. Vergauwen, M. Pollefeys and L. Van Gool
A Hierarchical Symmetric Stereo Algorithm Using Dynamic Programming.
International Journal of Computer Vision, vol. 47, no. 1-3, pp. 275 – 285, 2002

[Wang and Mahadevan 1999]
Wang, G.; Mahadevan, S.
Hierarchical optimization of policy-coupled semi-Markov decision processes.
International Conference on Machine Learning, pp. 464 – 473, Bled, Slovenia, 1999

[Watkins 1989]
Watkins, C.
Learning from delayed rewards.
Ph.D. Thesis, University of Cambridge, England, 1989

[Watkins and Dayan 1992]
Watkins, C.; Dayan, P.
Q-learning.
Machine Learning, vol. 8, no. 3, pp. 279 – 292, 1992

[Wheeler et al. 2002]
Wheeler, D.S.; Fagg, A.H.; Grupen, R.A.
Learning Prospective Pick and Place Behavior.
IEEE/RSJ International Conference on Development and Learning, pp. 197 – 203, Cambridge, Massachusetts USA, 2002

[Web3d 2007]
Web3d Project
Web3d Internetseite.
http://www.web3d.org/ letzter Aufruf 20.04.2007

[Westhoff 2007]
Westhoff, D.
Hierarchische multi-sensorgestütze Selbst-Lokalisierung mobiler Roboter
Dissertation, Fakultät für Mathematik, Informatik und Naturwissenschaften, Universtät Hamburg, Hamburg, Germany, noch nicht veröffentlicht

[Westhoff et al. 2006]
D. Westhoff, H. Stanek, J. Zhang
Distributed Applications for Robotic Systems using Roblet®-Technology,
ISR/Robotik 2006, VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, Munich, Germany, 2006

[Westhoff et al. 2004a]
Westhoff, D.; Stanek, H.; Scherer, T.; Zhang, J.; Knoll, A.
Mobile Manipulatoren und ihre aufgabenorientierte Programmierung,
atp - Automatisierungstechnische Praxis 10/2004, pp. 87 – 93, 2004

[Westhoff et al. 2004b]
Westhoff, D.; Stanek, H.; Scherer, T.; Zhang, J.; Knoll, A.
A flexible framwork for task-oriented programming of service robots.
Robotik 2004, VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, VDI-Berichte, vol. 1841, pp. 737 – 744, Munich, Germany, 2004

[Wöhlke 1994]
Wöhlke, G.
Grasp Planning & Force Computation For Dextrous Object Manipulation With Multi-Finger Robot Hands,
Technical Report TR-94-018, International Computer Science Institute, Berkley, California, USA, 1994

[Wu et al. 1996]
Wu, S.-P.; Vandenberghe, L.; Boyd, S.
MAXDET: software for determinant maximization problems.
User's Guide, Alpha version, Stanford April 1996,
http://www.stanford.edu/~boyd/MAXDET.html, letzter Aufruf 09.06.2007.

[Xia et al. 2004]
Xia, Y.; Wang, J.; Fok, L.-M.,
Grasping force optimization of multi-fingered robotic hands using a recurrent neural network.
IEEE Transactions on Robotics and Automation, vol. 20, no. 3, pp. 549 – 554, 2004

[Zhang and Knoll 1998]
Zhang, J.; Knoll, A.
Constructing fuzzy controllers with B-spline models - principles and applications.
International Journal of Intelligent Systems, vol. 13, no. 2/3, pp. 257 – 285, 1998

[Zhang and Knoll 1997]
Zhang, J.; Knoll, A.
Incrementally Constructing Sensor-Based Behaviors with B-Spline Basis Functions.
International Confrerence on Robotics and Automation, Albuquerque, New Mexico, USA, 1997

[Zhu and Wang 2003]
Zhu, X.; Wang, J.
Synthesis of force-closure grasps on 3-D objects based on the Q-Distance.
IEEE Transactions Robotics and Automation, vol.19, no.4, pp. 669 – 679, 2003

[Zhu et al. 2001]
Zhu, X.; Ding, H.; Li, H.
A quantitative measure for multi-fingered grasps
IEEE/ASME International Conference on Advanced Intelligent Mechatronics, vol. 1, pp. 213 – 219, Como, Italy, 2001

[Zhu et al. 2003]
Zhu, X.; Ding, H.; Wang, J.
Grasp analysis and synthesis based on a new quantitative measure.
IEEE Transactions on Robotics and Automation, vol. 19, no. 6, pp. 942 – 953, 2003

[Zivkovic and Kröse 2004]
Zivkovic, Z.; Kröse, B.
An EM-like algorithm for color-histogram-based object tracking.
IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 798 – 803, Washington, DC, USA, 2004

Index