Universität Bremen

Fachbereich 03
Mathematik/Informatik

Masterarbeit

Aufbau, Ansteuerung und Simulation eines interaktiven Ballspielroboters

Construction, Control and Simulation of an interactive ballgame robot

Autor
Tobias Hammer

Erstprüfer: Prof. Dr.-Ing. Udo Frese
Zweitprüfer: Dr.-Ing. Dipl.-Inform. Thomas Röfer

Bremen, September 2011

Erklärung

Bisher habe ich keine Informatik-Masterprüfung in dem gleichen oder einem fachlich entsprechenden Studiengang an einer wissenschaftlichen Hochschule in der Bundesrepublik Deutschland endgültig bestanden oder nicht bestanden oder befinde mich in einem entsprechendem Prüfungsverfahren.
Außerdem habe ich nicht den Prüfungsanspruch durch endgültiges Nichtbestehen einer Prüfung verloren.

Bremen, den 19. September 2011

_____

Tobias Hammer

Inhaltsverzeichnis

1 Einleitung

Spielen ist eine der ältesten und natürlichsten Verhaltensweisen höher Lebewesen. Auf diese Weise findet ein intensiver Prozess des miteinander Messens und Lernens statt. Es dient zum Erhöhen der Geschicklichkeit, dem Erfahren sozialen Kontakts und auch der Unterhaltung. Beim Menschen fehlt der Druck, seine Fähigkeiten durch spielerischen Wettstreit zu verbessern, dort liegt der soziale und unterhaltende Aspekt im Vordergrund.

Auf der Suche nach immer neuen Möglichkeiten und Herausforderungen ist es nicht verwunderlich, dass Menschen von Anfang an versucht haben, Maschinen in ihre spielerische Unterhaltung einzubeziehen. Auch als dies technisch noch kaum möglich war, wurde die Faszination für das Messen mit der Maschine zur Vermehrung von Ruhm und Geld genutzt. Ein Beispiel, welches das Potential und die Anziehungskraft solcher Ideen belegt, ist der „Schachtürke“ [Mra01]. Dies war ein schachspielender Roboter, der um 1769 gebaut wurde, sich jedoch als Betrug herausstellte.

Heutzutage ist es technisch problemlos möglich gegen eine Maschine in Form eines Computers anzutreten. Diese sind fähig zu einer großen Zahl von unterschiedlichsten Spielen und dem Menschen in vielen Aspekten weit überlegen. Einzig die Möglichkeiten der Interaktion zwischen Mensch und Maschine sind seit Jahren kaum verändert und orientieren sich noch immer weit mehr an den technischen Gegebenheiten, denn an Menschlichen. Bis vor wenigen Jahren waren die Interaktionsmöglichkeiten im Massenmarkt sehr begrenzt, so stellten Controller, Mäuse und Tastaturen den Standard dar. Nur im kommerziellen Unterhaltungsbereich existierten schon weitaus länger spezialisierte Eingabegeräte, die einen höheren Realismus und ein intuitiveres Verhalten ermöglichen sollten. So reichte dort die Spannbreite von Plastikwaffen als Eingabegerät bis zu vollständig nachgebauten Fahr- oder Flugzeug-Cockpits.

In den letzten Jahren hat eine Veränderung begonnen bei der die Interaktion mit der Maschine physischer wird. Den Anfang im Massenmarkt machte Nintendos Wii-Konsole [Nin11], bei der statt dem Drücken von Knöpfen ein Contoller im Raum bewegt werden muss woraufhin diese Bewegungen im Spiel mehr oder weniger ähnlich umgesetzt werden. Es folgten Sony mit Move, einer Kopie des Wii-Controllers und, mit einer echten Neuerung Microsoft. Deren Kinect [Mic11] ermöglichte zum ersten Mal im Massenmarkt eine Interaktion vollständig ohne Controller. Es reichte aus, sich vor einer Kamera zu bewegen. Diese filmt die spielenden Personen, extrahiert ihre Bewegungen und ermöglicht eine Verwendung dieser Informationen im Spiel.

Mit dieser Entwicklung hat sich die Eingabeseite hin zu einer natürlichen, physischen Ausprägung bewegt. Der Mensch muss sich weitaus weniger den technischen Gegebenheiten anpassen und kann umso natürlicher agieren. Vom Realismus her ist dies jedoch noch immer weit von den kommerziellen Spielhallen- und Jahrmarkt-Geräten entfernt. Es fehlt weiterhin ein echter Kontakt mit Gegenständen aus dem Spiel. So können zwar die aktuellen Spielkonsolen die Bewegungen für ein virtuelles Tennisspiel exakt einfangen und umsetzen, das Gefühl des Schlägers und der Aufprall des Balls fehlen dagegen vollständig. Doch auch die kommerziellen, nicht für den Endverbraucher-Markt entwickelte Geräte haben in diesem Bereich große Defizite. So naturgetreu die Eingabegeräte auch sein mögen, so wenig realistisch ist die darauf folgende Reaktion des Spiels. Ein Feedback beschränkt sich bisher fast ausschließlich auf eine sich verändernde Darstellung am Bildschirm. Dieses Bild wird zwar mit der aufkommenden 3D-Technik wirklichkeitsgetreuer, ist aber von einem echten, physischen Feedback noch weit entfernt.

Der logische nächste Schritt, nachdem die Eingabemethoden einen solchen Entwicklungssprung hin zu einer natürlichen, physischen Interaktion vollzogen haben, wäre es nun, die Gegenseite anzugehen. Dies bedeutet, dass die Reaktion der Maschine sich von der limitierten, visuellen Dar-

Abbildung 1.1: Der im Rahmen dieser Arbeit entwickelte und aufgebaute Roboter Piggy

stellung löst und in den physischen Raum verschoben wird. Erst dadurch würde eine realistische, wirklichkeitsgetreue Interaktion möglich werden. Stellt sich die Frage, warum diese Entwicklung erst langsam beginnt, wo die Vorteile so signifikant sind. Die Schwierigkeiten sind vielfältig und begründen sich in der noch immer starken Limitierung der technischen und mechanischen Möglichkeiten. So existieren zwar simple Ansätze für eine haptische *Virtual Reality*, bei denen einzelne Gegenstände erfühlt werden können. Die Forschung für eine ganzkörperliche, physische Interaktion steht hingegen noch am Anfang. Mangels Möglichkeiten zum praktischen Testen und Entwickeln bestehen auch auf algorithmischer Seite Defizite.

Eine von mehreren Möglichkeiten, der Gegenseite eine physische Interaktion zu ermöglichen, ist der Einsatz von Robotern. So können mit diesen, trotz der vorhandenen Defizite bereits jetzt Spiele gespielt werden, wenn entsprechende Einschränkungen akzeptiert werden. So verwendet man beispielsweise reguläre Spiele, die normalerweise zwischen Menschen ausgetragen werden, welche aber nur eingeschränkte Eingriffsmöglichkeiten in das Spiel bieten. Schach wäre ein solches Spiel, das ein Roboter relativ einfach auch physisch beherrschen könnte, Tischfussball ein anderes, schwierigeres. Es lässt sich sagen, dass ein Spielkonzept umso einfacher als Spiel mit Menschen und Robotern umsetzbar ist, wenn es mehreren Anforderungen genügt. Dies ist zum Einen eine geringe physische Interaktion, die keine harten Zeitkriterien besitzt und zum Anderen, wenn der Roboter nur über wenige Eingriffmöglichkeiten mit wenigen Freiheitsgraden am Spiel teilnehmen kann.

Die im Rahmen dieser Arbeit betrachteten Ballspiele entsprechen kaum diesen Anforderungen. Sie sind meist, je nach Spielkonzept, mit physischem Kontakt verbunden, mindestens mit dem Ball im Extremfall auch mit den menschlichen Spielern. Ballspiele sind sehr dynamisch und schnell und die Interaktionen müssen harten Zeitkriterien genügen, da sonst ein Eingriff in das Spiel, beispielsweise zum kontrollierten Treffen eines Balls unmöglich ist. Letztendlich sind die Fähigkeiten eines

menschliches Spielers der Standard, an dem sich ein Roboter messen muss. Nur wenn sein Spielniveau hoch genug ist, können anspruchsvolle Spielsituationen geschaffen werden, die langfristige Motivation und Herausforderung versprechen.

Es gibt dennoch einen Weg, die Anforderungen zu entschärfen, der hier gewählt wurde. Statt den Roboter immer als Gegenspieler zu positionieren, besteht die Möglichkeit eines kooperativen Spiels. Bei diesem tritt der Roboter einem Team menschlicher Spieler bei, um gegen andere Menschen antreten zu können. Je nach Spielkonzept lässt sich so erreichen, dass die Mitspieler die Schwächen des Systems erkennen und umgehen anstatt, wie es in einer rein konkurrierenden Situation der Fall wäre, auszunutzen. Als positive Begleiterscheinung besteht bei einem kooperativen Spiel, das prinzipbedingt mindestens zwei menschliche Spieler enthält, ein sozialer Aspekt, der verhindert, dass das Spiel mangels zwischenmenschlichen Verhaltensweisen an Reiz verliert.

Bei dem Einsatz von Robotern muss immer der Sicherheitsaspekt im Vordergrund stehen. Im industriellen Umfeld lässt dieser sich durch Ausschluss von Menschen leicht sicherstellen, im spielerischen Umfeld ist dies kaum möglich. Besonders Ballspiele, die oft auf physische Nähe ausgelegt sind und ein schnelles, dynamisches Verhalten notwendig machen bedürfen besonderer Maßnahmen zum Schutz der beteiligten Menschen. Nur wenn diese gegeben sind, ist eine kommerzielle Nutzung im Unterhaltungsbereich erst möglich.

1.1 Das Spiel

Zur Umsetzung mit einem Roboter wurde das Kinderspiel *Schweinchen in der Mitte* (engl: *Piggy in the middle*) ausgewählt. Es wird mit mindestens drei Personen gespielt, von denen eine das *Schweinchen* darstellt und in der Mitte steht (Abbildung 1.2(a)). Die anderen Spieler spielen zusammen gegen die Person in der Mitte und müssen sich den Ball so zuwerfen, das der Spieler in der Mitte ihn nicht fängt. Geschieht dies doch, vertauschen sich die Rollen und der Werfer muss in die Mitte[1].

Eine direkte Ersetzung eines Spielers mit einem Roboter wäre schwer umsetzbar. Der Roboter müsste beide Spielpositionen beherrschen, um erfolgreich teilnehmen zu können. Dafür müsste er sehr mobil sein, damit nicht um ihn herum gespielt werden kann und er müsste in der Lage sein, einen Ball kontrolliert zu fangen sowie diesen gezielt und schnell weiter zu werfen. Alle diese Eigenschaften wären möglich, würden jedoch den Aufbau des Systems sehr komplex machen. Auch die Sicherheitsfrage einer solchen Umsetzung wäre noch zu klären.

Das grundlegende Spielprinzip kann jedoch so variiert werden, dass es weitaus einfacher kooperativ mit einem Roboter gespielt werden kann. Dafür wird der Spielaufbau in Abbildung 1.2(b) gewählt, bei dem der Roboter in der Mitte positioniert ist und sich die menschlichen Spieler in zwei Kreisen um diesen herum anordnen. Die Spieler im äußeren Kreis gehören zum Team des Roboters, die im inneren sind die Gegenspieler. Der Spielablauf wird so modifiziert, dass die Mitspieler den Ball zum Roboter werfen und dieser ihn weiter zu einem Mitspieler passt, ohne das ein Gegenspieler ihn fangen kann. Sollte der Ball des Spielers doch gefangen werden, muss der Werfer nach innen[2].

Diese Abwandlung bietet eine Reihe von Vorteilen. So muss der Roboter nicht mobil sein und den Ball auch nicht fangen und festhalten können. Dies würde zwar mehr taktische Möglichkeiten bieten, ist für die Teilnahme am Spiel jedoch nicht zwingend notwendig. Ein weiterer Vorteil ist das kooperative Spielkonzept. Die Mitspieler müssen, und werden, versuchen, den Ball so zu werfen, dass der Roboter ihn bestmöglich spielen kann. Nur so verhindern sie eine Versetzung in das andere Team. Dadurch braucht der Roboter nicht mit allen Tricks und Finessen eines menschlichen Werfers umgehen können.

Ein weiterer großer Vorteil dieser Umsetzung ist das Konzept der Kreise. Auf diese Weise lässt sich einfach ein Sicherheitsabstand um den Roboter herum definieren, wodurch ungewollte Kontakte vermieden werden. Das Konzept ermöglich jedoch auch eine sehr einfache Klassifizierung der Spieler in Mit- und Gegenspieler. So muss nur die Entfernung eines Spielers zum Roboter bekannt sein, um ihn eindeutig einer der beiden Gruppen zuordnen zu können.

Aus dem bisher beschriebenen Ablauf lassen sich eine Reihe von Anforderungen generieren, die ein solches Robotersystem beherrschen muss. Es muss in der Lage sein, einen zugeworfenen Ball

[1] Es existieren viele Variationen der Regeln, von denen hier exemplarisch eine ausgewählt wurde

[2] Entweder die Positionen werden getauscht oder der Spieler hat gewonnen, der als Letztes noch aussen steht

an einen Mitspieler weiter zu passen. Dafür muss es den Ball zeitnah erkennen, verfolgen und seine Flugbahn mit einem Manipulator verändern können. Zum gezielten Passen muss die Flugbahn kontrolliert geändert werden und es müssen die Positionen der Mit- und Gegenspieler vorliegen und einbezogen werden.

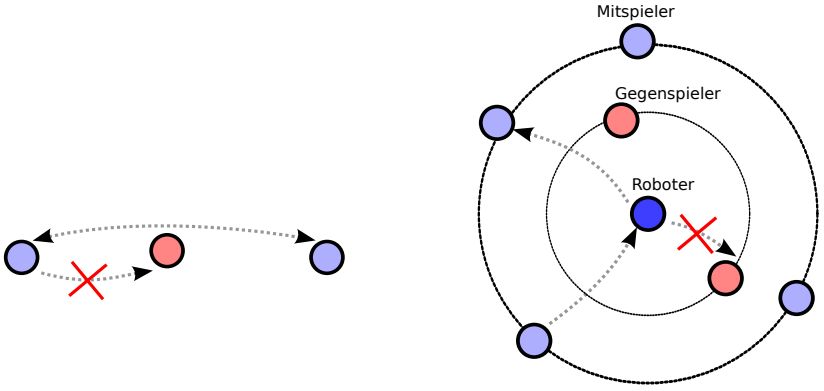


Abbildung 1.2: *Schweinchen in der Mitte* für (a)Menschen und (b)Menschen und Roboter

1.2 Zielsetzung

Das Fernziel ist demnach die Entwicklung eines Roboters für die vorgestellte Abwandlung des Spiels *Schweinchen in der Mitte*. Dem Roboter wurde der Name *Piggy*[3] verliehen, in Anlehnung an das Spiel, dass er zukünfig spielen soll. Piggy soll als Event-Modul eingesetzt werden, d.h. es soll von einer Event-Agentur für verschiedene Veranstaltungen vermietet werden und dort der Unterhaltung der Teilnehmer dienen.

Aus diesem Fernziel ergeben sich eine Reihe von Anforderungen, die bereits im frühen Prototypenstadium beachtet und eingearbeitet werden müssen.

- Sicherheit: Auch wenn das Spielkonzept bereits Vorkehrungen zur Vermeidung eines Mensch-Roboter-Kontakts enthält, so muss das System bei deren Versagen sicher sein. Der Roboter darf unter keinen Umständen Menschen verletzen oder gefährden.
- Performance: Der Roboter muss unter verschiedensten Umgebungsbedingungen arbeiten und unabhängig davon ein flexibles und dynamisches Spiel bieten können. Nur so kann ein spannendes und kurzweiliges Erlebnis geboten werden.
- Kosten: Das System muss, da es kommerziell eingesetzt werden soll, gewinnbringend sein. Dafür dürfen weder die Herstellungskosten noch der Betrieb den Rahmen von bereits existierenden Event-Modulen sprengen.

Die vorliegende Arbeit ist ein erster Schritt auf dem Weg zur Entwicklung eines sicheren und vertriebsfertigen Robotersystems für das vorgestellte Spiel. Der eigene Beitrag der Arbeit besteht dabei aus den im Folgenden beschriebenen Punkten.

- Konzeptionierung einer geeigneten Hardware und Ermittlung der Parameter für die Auslegung unter Berücksichtigung der Anforderungen.
- Mechanischer und elektrischer Aufbau eines Prototypen.
- Entwicklung einer Software zur Ansteuerung und Abstraktion der Hardware.
- Planung und Umsetzung eines einfachen Algorithmus zum Treffen eines zugeworfenen Balls.
- Experimente zur Tauglichkeit von Konstruktion und Software. Er- und Einarbeitung von Verbesserungen.

[3] **P**hysikalisch **i**nteraktives Verg**n**ügun**g**s-Roboters**y**stem

- Entwicklung von Methoden und Software zum kontrollierten Passen eines zugeworfenen Balls.

Das entwickelte System (Abbildung 1.1) ist im aktuellen Stadium bereits in der Lage Bälle zurückzuspielen, die ihm zugeworfen werden. Es wurden jedoch auf Grund des frühen Entwicklungsstadiums Vereinfachungen gegenüber einem System gemacht, welches das vollständige Schweinchen-Spiel beherrscht. So ist noch kein Spiel mit mehreren Personen möglich und auch die Lokalisation und Klassifikation von umstehenden Personen liegt nicht im Fokus dieser Arbeit.

Das der bisherige Entwicklungsstand bereits eine Reihe interessanter Neuerungen und Konzepte enthält zeigt sich in der Einreichung der zugehörigen Publikation [LBHF12] zur ICRA 2012. Das Video zur Einreichung kann unter `http://www.der-hammer.info/piggy/piggy.mp4` abgerufen werden. Die praktische Tauglichkeit hingegen wird durch das rege Interesse der Spieler bei eine Vielzahl von Vorführungen belegt (s. Abschnitt 5.7).

2 Stand der Technik

In diesem Kapitel wird ein Überblick über derzeit erhältliche und in Entwicklung befindliche Roboter gegeben. Dabei wird im ersten Abschnitt der Hauptschwerpunkt auf den Bereich der spielerischen Mensch-Roboter-Interaktion gelegt. Ein Großteil der im Folgenden vorgestellten Beispielprojekte beschäftigt sich mit Robotern, die Ball-Spiele mit oder gegen den Menschen ausführen können. Entsprechend handelt der zweite Abschnitt von den Gemeinsamkeiten ballspielender Roboter und den Grundlagen der im Rahmen dieser Arbeit eingesetzten Ballerkennungs- und Verfolgungs-Algorithmen.

2.1 Spielende Roboter

Im Folgenden werden einige spielend mit dem Menschen interagierenden Roboter kurz beschrieben. Dabei liegt der Fokus besonders, aber nicht nur, auf Robotern, die Ball-Spiele beherrschen. Die ausgewählten Exemplare werden unter mehreren Gesichtspunkten betrachtet, die im Rahmen dieser Arbeit als relevant erscheinen. Zum einen sind dies das Spielkonzept und die damit einhergehenden Einschränkungen, die der Mensch in Kauf nehmen muss, um an diesem Spiel zu partizipieren. Zum anderen wird die Praktikabilität des Konzepts für einen Einsatz unter realen Bedingungen, d.h. außerhalb von kontrollierten Laborbedingungen betrachtet. Auch die Herausforderung und somit der Spielspaß für den Menschen sind neben den Gesamtkosten zu beachten.

Aibo und Paro

Im der Kategorie Spielzeug bzw. künstliche Tiere sind Sonys Roboterhund Aibo [FK98] wie auch die zu therapeutischen Zwecken eingesetzte Babyrobbe Paro [Shi01] angesiedelt. Auch wenn bei Ersterem im Rahmen des Robocups [The99] seine Fähigkeiten drastisch weiterentwickelt wurden, ist er auch dort nur im Spiel gegen Artgenossen erfolgreich. Der Reiz dieser Spielzeuge liegt mehr im Hervorrufen von Verhaltensweisen und Reaktionen des Roboters als im gleichberechtigten Spiel mit dem Menschen. Somit muss dieser sich vollkommen den Gegebenheiten des Spielzeugs anpassen und sich selbstständig eine Langzeitmotivation schaffen. Andererseits sind solche Systeme günstig, robust und direkt funktionsbereit zu vermarkten.

Basketball-Robbe

Ein vollkommen anderes Ziel wird in [Hu10] verfolgt. Dort wurde ein Roboterarm im Plüsch-Robben-Überzug, ausgestattet mit einem Stereokamera-System, entwickelt, der einen Basketballkorb treffen kann. Der Korb kann sich an unterschiedlichen Positionen befinden und wird aus wenigen Metern Entfernung mit einer Wahrscheinlichkeit von ca. 99% getroffen. Trotz der mit einem Menschen vergleichbaren Trefferquote läßt sich hier nur schwerlich ein kompetitives Spielprinzip entdecken. Auch der stark vorgegebene Aufbau, bestehend aus einem einfarbigen, dunklen Hintergrund und exakt definierter Ballablage läßt kaum einen Einsatz, außer zur simplen Demonstration des einprogrammierten Ablaufs, zu.

JediBot

Mit einem Leichtbau-Roboterarm ausgestattet und mit Microsofts Kinect als Bild- und Tiefenkamera bestückt haben Studenten der Stanford-University den JediBot [Sta11] entwickelt. Der Arm

ist mit einem Schaumstoff-Stab als Lichtschwert bestückt und kann simple Angriffs- und Abwehbewegungen in Bezug auf ein menschliches Gegenüber ausführen. Bisher führt der Arm nur relativ langsame Bewegungen aus, legt dem auch mit einem Schaumstoff-Stab bewaffneten Menschen jedoch kaum Einschränkungen auf. Auch der extrem simple und robuste Aufbau würde einen Einsatz unter realistischen Bedingungen zulassen. Nur eine garantierte Sicherheit für den Menschen dürfte bei diesem, auf direkten Kontakt ausgelegten Konzept, besonders wenn der Roboter schnell und dynamisch reagieren soll, schwierig zu realisieren sein.

Justin

Einen vollständigen Roboter mit Fahrgestell, zwei Armen und Händen mit mehreren Freiheitsgraden stellt Justin dar. Er wurde am DLR[1] entworfen und gebaut. Der Roboter kann bis zu zwei zugeworfene Bälle mit seinen Händen fangen [BFB11]. Dazu besitzt er in seinem Kopf ein Stereokamera-System. Ein Teil der Rechenkapazität wurde auf ein Cluster ausgelagert, sonst ist er jedoch ein abgeschlossen funktionierendes System. Von seinen Freiheiten her würde er sich als Spiele-Roboter bestens eignen, da er dem Menschen ähnlich ist und diesem damit nur sehr wenige Einschränkungen in der Art des Spiels auferlegen würde. Die Ausrichtung dieses Prototypen-Projekts geht jedoch mehr in Richtung Service-Robotik. Dort ist Flexibilität wichtiger als schnelles, reaktives Verhalten, wie es für Spiele wichtig ist. Vor allem der Preis, sowie die Fragilität der Konstruktion würde einen praktikablen Einsatz im angepeilten Segment bisher ohnehin wenig attraktiv machen.

RoboKeeper

Der RoboKeeper [4at11] ist ein Torwart-Roboter. Er besteht aus einer Torwart-Silhouette vor einem Tor, die mittels eines Freiheitsgrades gedreht werden und so nahezu beliebige Positionen des Tors vor zugespielten Bällen abschirmen kann. Die Bälle werden durch Stereo-Kameras erkannt und verfolgt. Das System kann Bälle von Profifußballern halten und bietet somit ausreichend Herausforderung für jeden menschlichen Gegner. Durch Aufgreifen eines simplen aber bekannten Spielprinzips muss der Mensch sich nicht an das System anpassen und gewöhnen. Der Roboter kann als Event-Modul gemietet werden und ist somit vollkommen praxistauglich und erschwinglich. Allerdings muss für den Einsatz eine größere Konstruktion um den eigentlichen Torwart-Roboter herum von mehreren Helfern aufgebaut werden.

KiRo

Ein weiteres praxistaugliches und im Einsatz befindliches System ist KiRo [WN03], ein Tischfußball-Roboter. Das System bewegt die Stangen des Tischfußballspiels mittels Aktoren und beobachtet den Ball mit einer über dem Spielfeld angebrachten Kamera. Auch dieses System ist dem Menschen ebenbürtig und greift ein bekanntes Spielprinzip auf, d.h. es reduziert die Freiheit des menschlichen Spielers nicht weiter indem es ihm die vom Spielprinzip vorhandenen vielseitigen Steuerungsmöglichkeiten lässt. Zusätzlich ist der Aufbaubedarf sehr gering und die Sicherheit des Menschen kann garantiert werden.

Erkenntnisse

Bei diesem Überblick über spielende oder spieletaugliche Roboter-Systeme wurde versucht einen möglichst repräsentativen Querschnitt über aktuelle Entwicklungen zu erhalten. So fallen einige der Systeme in die Kategorie Spielzeug, andere sind Forschungsprototypen oder neue Ansätze, während einige Systeme bereits praxistauglich sind und sich auf dem Markt befinden.

Es fällt auf, dass die auf dem Markt befindlichen, mit dem Menschen konkurrierenden Systeme noch keine große Verbreitung gefunden haben oder nur mit Personal gemietet werden können. Das zeigt, dass die Kosten zur Anschaffung noch nicht Endverbraucher-kompatibel sind oder der Betrieb des Systems soviel Aufwand bzw. Erfahrung benötigt, sodass ein Betrieb nur mit Personal möglich ist. Beides sind Einschränkungen, die mittel- bis langfristig überwunden werden können.

[1] Deutsches Zentrum für Luft- und Raumfahrt e.V.

Ein weiterer markanter Punkt ist, dass es Schwierigkeiten zu bereiten scheint, mit dem Menschen in Punkto Geschwindigkeit, Flexibilität und Anpassbarkeit zu konkurrieren. Doch letztendlich ist der Mensch der Standard, an dem sich jedes spielende System messen muss, das kommerziell vermarktet und Menschen mit herausfordernden Spielen unterhalten soll.

Bis diese Schwierigkeiten für beliebige spielerische Mensch-Roboter-Interaktionen gelöst sind, besteht bereits jetzt, wie die kommerziellen Systeme zeigen, die Möglichkeit den Menschen auf ein extrem einfaches (RoboKeeper) oder in sich extrem gut abschlossenes System (KiRo) festzulegen. In diesem Rahmen können spielende Roboter bereits heute dem Menschen ebenbürtig bis überlegen sein.

2.2 Ballspiele und Ballerkennung

Wie an den Beispielen im vorherigen Abschnitt ersichtlich wurde, existieren sehr unterschiedliche Ansätze zu Ballspielen mit Robotern und Menschen. Neben den dort meist betrachteten kompetitiven Spielprinzipien, bei denen der (eine) Roboter gegen den oder die Menschen spielt, sind auch kooperative Spielkonzepte denkbar. Allen gemeinsam ist jedoch eine sehr grundlegende wie entscheidende Technik: Die Erkennung und Verfolgung eines Balls. Aufgrund der Vielseitigkeit und breiten Verfügbarkeit hat sich für diese Aufgabe der Einsatz von einer oder mehreren Kameras durchgesetzt.

Wieviele Kameras notwendig sind, um das Objekt des Interesses, hier meist ein Ball, im Raum lokalisieren zu können hängt von der Einsatzumgebung ab. So reicht, wenn der Ball sich hauptsächlich in zwei Dimensionen bewegen kann, wie es etwas bei *KiRo* der Fall ist, eine Kamera theoretisch aus. Bei dreidimensionalen Ballbewegungen läßt sich die Position im Raum erst mit zwei Kameras eindeutig berechnen. Mehr Kameras sind für eine höhere Genauigkeit immer möglich, benötigen jedoch mehr Rechenleistung bei der Auswertung.

Die Schwierigkeit, die sich durch das betrachtete Szenario ergibt ist, dass die Reaktionen des Roboters in Echtzeit erfolgen müssen. Somit müssen die eingehenden Bilddaten vor Eintreffen des nächsten Kamerabildes ausgewertet worden sein. Wenn die Kameras beispielsweise mit den üblichen 50 Hz betrieben werden, bleiben pro Bild nur 20 ms.

Ballerkennung und -verfolgung

Bei der hier betrachteten, und dem aktuellen Stand entsprechenden Technik, werden die Eingangsdaten der Kameras in zwei Schritten bearbeitet. Der erste Schritt extrahiert die zu suchenden Formen unabhängig voneinander aus jedem Kamerabild. Bei Bällen ist die gesuchte Form meist (annähernd) kreisförmig. Demnach ist es das Ziel, genau solche Kreise vor dem Hintergrund zu erkennen, die Bälle repräsentieren.

Da bei diesem Schritt viele Fehlerkennungen auftreten können, etwa durch weitere, sich im Blickfeld der Kamera befindende kreisförmige Objekte, müssen die Ergebnisse anschließend gefiltert werden. Auch können aus einer einzelnen Momentaufnahme noch nicht alle benötigten Informationen gezogen werden. Damit der Roboter auf den Ball reagieren kann, ist neben der Position zusätzlich die aktuelle Bewegungsrichtung wichtig. Dafür müssen die Informationen aus mehreren aufeinander folgenden Bildern fusioniert werden.

Dieser Schritt wird heutzutage meist mittels probabilistischer Algorithmen durchgeführt [TBF05]. Diese gehen davon aus, dass sich die Bewegung des Balls in der Zukunft aus seinem aktuellen Zustand heraus berechnen lässt. Dafür wird ein Modell des Balls erstellt, aus dem zusammen mit dem bereits genannten Zustand ein neuer, zukünftiger Zustand berechnet werden kann. Ein Zustand kann beispielsweise Position, Radius und Geschwindigkeit enthalten. Aber auch weitere Parameter, wie der Spin können je nach Anforderung und Einsatzzweck enthalten sein. Das Modell beschreibt das Verhalten des Balls, beispielsweise die zweidimensionale Bewegung mit Reflexion an den Feld-Rändern bei *KiRo* oder die Schwerkraft-bestimmte Flugbahn der Bälle bei *Justin*.

Gegeben einen Zustand und ein Modell kann nun berechnet werden, wie wahrscheinlich ein im Bild erkannter Ball der internen Vorhersage entspricht. Die erkannten Kreise können dafür entweder mittels der bekannten Kamerakalibrierung in eine 3D-Position umgerechnet werden oder einzeln

für jede Kamera integriert werden. Aus der Übereinstimmung mit der Vorhersage kann entschieden werden, ob ein erkannter Kreis zum internen Zustand passt und dieser gegebenenfalls anhand der neuen Informationen korrigiert werden muss. Dieses Vorgehen verwirft implizit bereits einen großen Teil der fehlerkannten Kreise.

Bisher wurd jedoch bei dieser grundlegenden Beschreibung der Faktor Mensch vollständig ignoriert. Wenn der Mensch mit dem Ball interagiert, ihn z.B. schießt oder wirft, verhält der Ball sich während dieser Zeit unvorhersehbar und nicht dem Modell entsprechend[2]. Erst wenn der Mensch seinen Ballkontakt beendet hat, beginnt eine Bewegungsphase, in der der Ball sich dem Modell entsprechend verhält. Ein für diesen Einsatzzweck geeigneter Algorithmus muss diesen Zeitpunkt erkennen, ab dem eine Vorhersage möglich ist und dann so schnell wie möglich, also bestenfalls nach zwei Frames, einen internen Zustand des Balls erstellt haben. Um die Genauigkeit der Erkennung dieses Zeitpunkts weiter zu erhöhen, kann bekanntes Wissen über den Start einer beschreibbaren Ballbewegung integriert werden. Beispielsweise Wissen, wo solch ein Start mit welcher Wahrscheinlichkeit stattfindet und welchen Geschwindigkeitsvektor er mit welcher Wahrscheinlichkeit dort besitzt.

Das eingesetzte Verfahren

Die Ballerkennung und -verfolgung, die im Rahmen dieser Arbeit eingesetzt wird, wurde für ein anderes Projekt entwickelt und von dort portiert. Originär läuft sie auf *Rollin' Justin* des DLR (s. Abschnitt 2.1 und Abbildung 2.1) und wurde in [BF09, BBF11] vorgestellt. Dort erkennt diese Programmkomponente mittels Stereo-Kameras beliebig viele zugeworfene Bälle und ermöglicht dem Roboter über eine Vorhersage der Flugbahn bis zu zwei Bälle gleichzeitig mit seinen Händen zu fangen.

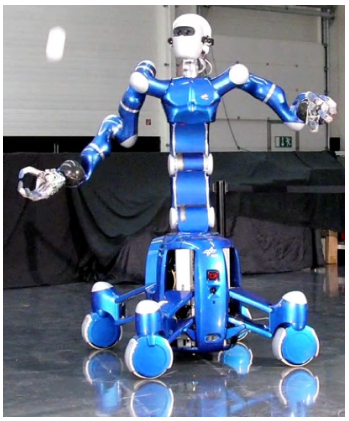


Abbildung 2.1: Rollin' Justin beim Ballfangen (Bild: DLR)

Der Algorithmus verwendet den im vorherigen Abschnitt beschriebenen, zweistufigen Ansatz. Im Kreiserkenner arbeitet eine effiziente Implementierung der Hough-Transformation für Kreise. Sie arbeitet auf Graustufen-Bildern, ist beleuchtungsinvariant und benötigt somit keine Kalibrierung. Besonders diese Stufe muss einen hohen Pixeldurchsatz erreichen und ist folglich stark auf Geschwindigkeit optimiert. Sie nutzt dazu die Mehrkern- und SIMD[3]-Fähigkeiten moderner Prozessoren. Die Kreiserkennung wird für jedes Bild von beiden Kameras einzeln ausgeführt und liefert die besten Kreise mit Subpixel-Genauigkeit an die nächste Stufe.

Die zweite Stufe besteht aus einem *Multi Hypothesis Tracker* (MHT) [CH96] der den im vorherigen Abschnitt grob beschriebenen probabilistischen Ansatz verwendet. Der MHT verwaltet intern einen Zustand zu jeder erkannten Ballflugbahn, der unter anderem die Position und Geschwindigkeit enthält. Der Zustand wird dabei mit einem *Unscented Kalman Filter* [TBF05] als Mittelwert und Covarianz geschätzt. Das zugehörige Modell bezieht den Einfluss der Schwerkraft

[2] Es wird davon ausgegangen, dass der menschliche Einfluß nicht modelliert werden kann
[3] Single Instruction Multiple Data

auf den Ball ebenso ein, wie das Abbremsen durch die Luftreibung. Für den Start von neuen Ballflugbahnen (Tracks) werden bekannte Informationen über die wahrscheinliche Abwurfposition und Geschwindigkeit wiederum als Mittelwert und Covarianz einbezogen.

Genauere Informationen zum internen Aufbau dieser Komponente, zur Gewinnung der Modellparameter und der vorhandenen Schnittstellen finden sich in Abschnitt 4.1.2.

3 Hardware

In diesem Kapitel wird die Hardware des Robotersystems genauer betrachtet. Zuerst wird in Abschnitt 3.1 das zu Grunde liegende Konzept des Roboters beschrieben, wie der prinzipielle Aufbau gestaltet ist und wie mit dem Ball interagiert wird. Danach werden die Anforderungen an die Hardware untersucht und physikalische Effekte und Richtwerte hergeleitet. Anschließend beschäftigt sich Abschnitt 3.3 mit der konkreten Umsetzung unter Einhaltung der zuvor aufgestellten Rahmenbedingungen.

3.1 Konzept

Der Roboter soll, wie bereits in der Einleitung beschrieben einer Reihe von Anforderungen genügen. So schränkt die Forderung nach einem minimalistischen Aufbau sowie die nach einem möglichst preiswerten System die Freiheiten beim Entwurf drastisch ein. Ein üblicher Industrieroboter, wie er millionenfach im Einsatz ist, würde beispielsweise beiden Forderungen zuwider laufen. Gleiches gilt für einen komplexen Eigenentwurf.

Besonders der preisliche Aspekt begrenzt die Anzahl und Stärke der Antriebe, die das System bewegen. Um dennoch die geforderte hohe Performanz zu erreichen wird eine simple, wie clevere Konstruktion benötigt. Nur so können alle drei Anforderungen bestmöglich erfüllt werden.

Es wurde das Hardwarekonzept gewählt, wie es in Abbildung 3.1 dargestellt ist. Der bewegliche und damit spielrelevante Teil des Roboter soll demnach aus einer Stange mit einer Kugel am oberen und einem Gelenk am anderen Ende bestehen. Dieser Schläger dient zum interagieren mit dem anfliegenden Ball, indem seine Kugel mit ihm kollidiert. Das Gelenk wäre optimalerweise ein Kugelgelenk, das beliebig durch Aktoren bewegt werden kann.

Diese Aktoren müssen direkt am Gelenk angebracht werden, um es bewegen zu können. Es wird hier allerdings davon ausgegangen, dass die Aktoren nur Rotationsbewegungen verursachen können und in Reihe arbeiten, d.h. ein Aktor auf den nächsten montiert ist. So muss zwar ein unterer das Gewicht des auf ihm befestigten Aktors tragen, jedoch ist durch den geringen räumlichen Versatz der Hebelarm und damit das entstehende Drehmoment gering. Bei einer Montage direkt auf der Achse des vorangehenden Aktors würden sich die Drehmomente sogar auslöschen und nur die Trägheit des Systems als Folge der höheren Masse ansteigen.

Es lässt sich somit festhalten, dass idealerweise möglichst wenige und leichte Aktoren verwendet werden und diese möglichst nah beieinander montiert werden sollten. Optimal wäre es, wenn sich die aus der Massenverteilung ergebende Symmetrieachse des oberen Aktors direkt auf der Drehachse des tieferliegenden befinden würde.

Weitere Details zur genauen Umsetzung dieser Anforderungen sowie weitere sich ergebende Einschränkungen und Vorgaben an die Hardware finden sich im weiteren Verlauf dieses Kapitels.

Verfügbare Freiheitsgrade

Der Roboter soll mit dem ihm zugeworfenen Ball interagieren. Da seine Konstruktion es ihm nicht erlaubt, den Ball zu halten besteht seine einzige mögliche Einflußnahme darin, die Flugbahn des Balls direkt zu verändern. Dementsprechend muss die Geschwindigkeit verändert werden, mit der der zugeworfene Ball wieder abgespielt wird. Da dies eine beliebige Raumrichtung sein soll, läßt sich diese Größe durch drei Freiheitsgrade beschreiben, beispielsweise einen Vektor in $\mathbb{R}^3$.

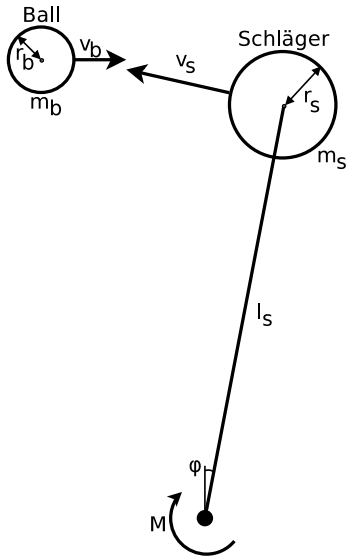


Abbildung 3.1: Schematischer Aufbau des Robotersystems

Es werden demnach mindestens drei Freiheitsgrade roboterseitig benötigt, um den Ball geeignet beeinflußen zu können. Zwei davon können aus dem Gelenk des Roboters kommen. Dessen Konfiguration läßt sich beispielsweise durch zwei Winkel oder die Position des Schlägers auf dem zweidimensionalen, (näherungsweise) kugelförmigen Arbeitsbereich beschreiben. Da hier jedoch die Ballgeschwindigkeit beeinflußt werden soll, sind die zwei durch das Gelenk bestimmten Freiheiten die Geschwindigkeiten, mit denen sich das Gelenk und damit der Schläger bewegt. Die Ausrichtung des Endeffektors, also der Kugel am andere Ende des Stabs, hat keinen weiteren Einfluß, da dieser vollständig rotationssymetrisch ist.

Bisher wurde angenommen, dass Ball und Schläger einen zentralen Stoß durchführen, also die Ballflugbahn mittig durch die Schläger-Kugel führt. In diesem Fall würde, wenn der Roboter sonst unbewegt ist, die Flugrichtung des Balls umgekehrt werden und dieser damit zum Werfer zurück fliegen. Seine Geschwindigkeit würde durch Reibungsverluste abnehmen, jedoch wäre seine Richtung unverändert.

Wenn jetzt die beschriebene Bewegung des Gelenks hinzugenommen wird, hat der Roboter während dieses Stoßvorgangs, also der Zeit, in der sich der Ball zuerst verformt und dann wieder seine Ausgangsform zurück erlangt, Einfluß. In dieser Zeitspanne kann durch die Reibung zwischen Ball und Kugel eine Geschwindigkeit auf den Ball übertragen werden. Jedoch kann der Ball nur maximal tangential zur Kugel bzw. orthogonal zu seiner ursprünglichen Flugbahn beschleunigt werden.

Da der Ball die Kugel jedoch nicht zwangsläufig zentral treffen muss, bieten sich hier weitere Größen, auf die Einfluß genommen werden kann. Der Auftreffpunkt ist zweidimensional beschreibbar und fügt dementsprechend zwei Freiheitsgrade hinzu. Ein solcher nicht-zentraler Stoß ist in Abbildung 3.2 dargestellt. Dort ist, zur besseren Veranschaulichung eindimensional, die Auswirkung eines Versatzes des Auftreffpunkts auf der Kugel gezeigt. Wenn die Kugel beispielsweise nach rechts versetzt wird prallt der Ball nach links ab. Analog führt ein Versatz nach links zu einer Geschwindigkeitsänderung nach rechts. Hier wurde angenommen, dass der Einfalls- gleich dem Reflexionswinkel ist, was stark vereinfacht ist aber das grundlegende Verhalten hinreichend beschreibt.

Das System besitzt demnach vier Freiheitsgrade, zwei aus den Gelenkgeschwindigkeiten und zwei aus dem Auftreffpunkt des Balls. Da nur drei Freiheitsgrade benötigt werden ist es redundant, wodurch nicht jede Ausgangsgeschwindigkeit nur durch exakt eine Kombination der Eingangsparameter erreicht werden kann. Da dies jedoch keine benötigte Anforderung ist, hat die Redundanz keine negativen Auswirkungen. Stattdessen ermöglicht sie es teilweise, aus mehreren möglichen Konfigurationen zu wählen und dementsprechend auch, die bessere von mehreren auszuwählen.

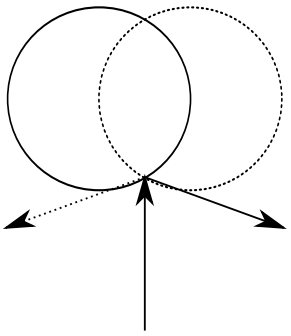


Abbildung 3.2: Zusätzliche DOF durch Auftreffpunkt des Balls auf dem Schläger

Sicherheit

Die Sicherheit eines solchen Robotersystems ist von entscheidender Bedeutung. Wie in der Einleitung geschildert, soll es nach seiner Fertigstellung als Event-Modul eingesetzt werden. Dort soll es in direktem Kontakt mit Menschen an dynamischen und schnellen Ballspielen teilnehmen. Es ist somit nicht auszuschließen, dass ein Mensch sich absichtlich oder unabsichtlicht in den Arbeitsbereich des Roboters bewegt während dieser versucht den Ball zu treffen. Dadurch ist nicht auszuschließen, dass der Roboter einen Menschen mit maximaler Geschwindigkeit treffen kann.

Eine physikalische, räumliche Abgrenzung zum Menschen wäre eine Lösung, würde aber das Spielerlebnis stark einschränken. Der bessere Weg ist es, solche Treffer einzuplanen und den Roboter so auszulegen, dass auch unter diesen Umständen keine Verletzungen auftreten. Dieser Punkt verbietet es, dass der gesamte Roboter mobil ausgelegt ist und sich beispielsweise auf Rollen durch den Raum bewegen kann. Müsste sich das Gesamtsystem mit Batterien, Rechner und weiteren Komponenten hinreichend schnell bewegen, um ein dynamisches Spiel zu garantieren, wäre die Sicherheit kaum mehr zu gewährleisten.

Um dieses Problem zu vermeiden ist der Roboter selbst statisch an einer Position im Raum aufgestellt. Nur sein Schläger, bestehend aus Stange und Kugel ist beweglich ausgelegt. Um hier Verletzungen zu vermeiden, müssen diese Komponenten möglichst leicht und, wenn möglich, weich sein. Wie diese Anforderung konkretisiert und umgesetzt wurde, wird unter anderem im weiteren Verlauf dieses Kapitels beschrieben.

3.2 Anforderungen

In diesem Abschnitt werden die Anforderungen an das Robotersystem genauer untersucht. Neben den grundlegenden Anforderungen an den Schläger sowie den Ball werden die beim Design auftretenden Konflikte beschrieben. Um eine geeignete Auslegung abzuleiten, werden zusätzlich die auf das System wirkenden physikalischen Effekte beschrieben. Im nächsten Abschnitt wird auf die konkrete Auslegung genauer eingegangen.

3.2.1 Schlägerparameter und -kräfte

Der Schläger mit seinem kugelförmigen Ende ist die wichtigste Komponente des gesamten Robotersystems. Mit ihm findet die Interaktion mit der Umwelt statt, er hat Einfluss darauf, welche Bälle der Roboter wie spielen kann. Bei der Suche nach einer optimalen Konstruktion dieser Komponente müssen mehrere, gegenläufige Parameter abgewogen werden.

So sollte die Länge des Schlägers (l_s) möglichst groß sein. Dadurch ergibt sich ein größtmöglicher Arbeitsraum und damit eine größere Chance, den anfliegenden Ball zu erreichen. Auch der Kugel-Radius (r_s) sollte möglichst groß sein, um den Ball präziser spielen zu können (s. 3.2.3). Dem gegenüber steht die Forderung nach einem leichten und kleinen System. Dieses stellt durch die geringeren auftretenden Kräfte weniger Anforderungen an die Aktoren. So ergibt sich das maximales Halte-Drehmoment M_{grav} bei $\varphi = 90°$ entsprechend (3.1).

$$M_{\text{grav}} = l_s m_s g \tag{3.1}$$

Dabei wurde der Schläger als Punktmasse an einem masselosen Stab der Länge l_s modelliert. Auch das Drehmoment, welches durch beschleunigen und abbremsen des Schlägers hervorgerufen wird, reduziert sich mit geringerem l_s und m_s. Es kann abhängig von der Winkelbeschleunigung α beschrieben werden (3.4).

$$J_{bat} = \frac{2}{5} m_s r_s \qquad \text{Trägheitsmoment massive Kugel} \tag{3.2}$$

$$J = J_{bat} + m_s l_s^2 \qquad \text{Verschiebung nach Satz von Steiner} \tag{3.3}$$

$$M_{\text{acc}} = J\alpha \tag{3.4}$$

Des weiteren verdient die Auslegung des Schläger-Stabs Aufmerksamkeit. Neben dem geringen Gewicht ist vor allem seine Steifigkeit ein wichtiger Parameter. Sie beschreibt, wie stark sich der Stab biegt, wenn er an einem Ende fixiert wird und eine Kraft orthogonal auf das freie Ende wirkt. Die zu beobachtende Auslenkung wächst proportional zur Länge l_s und zur aufgewendeten Kraft.

Eine hohe Steifigkeit verspricht eine größere Positions- und Treffgenauigkeit des Systems, da durch Beschleunigungsvorgänge ausgelöste Schwingungen verringert werden. Dudurch ist der Zustand des Gesamtsystems zu jedem Zeitpunkt exakter bekannt. Zusätzlich ist die Auslenkung bei Kollision mit dem Ball geringer, was auch hier eine bessere Vorhersagbarkeit vespricht. Andererseit wird durch ein Nachgeben des Stabs beim Kollisionsvorgang Energie aufgenommen, sodass das Drehmoment geringer ausfällt und die Aktoren weniger belastet werden.

3.2.2 Anforderungen an den Ball

Neben dem Schläger als Komponente des Roboters ergeben sich auch an den geworfenen Ball gewisse Anforderungen. Er sollte möglichst leicht sein, um geringe Kräfte bei der Kollision hervorzurufen. Dem gleichen Ziel ist die Forderung geschuldet, dass er Ball möglichst weich sein sollte. Dadurch ist die Verformung beim Aufprall größer und die hervorgerufenen Kräfte auf die Gegenseite wiederum geringer. Dem zuwider läuft die Forderung nach einem möglichst geringen Energieverlust beim Abprall. Hier wäre das Optimum ein vollkommen elastischer Stoßvorgang, denn wenn weniger Energie in nicht nutzbare Formen umgesetzt wird, verringert sich die Energie, die das System auf den Ball übertragen muss, um ihn entgegen seiner ursprünglichen Flugbahn zu beschleunigen.

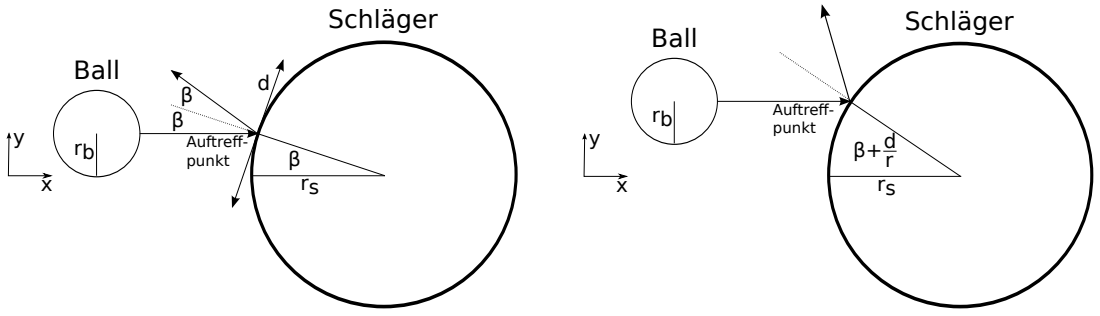
Weitere relevante Parameter sind die Oberflächenbeschaffenheit und das Flugverhalten. Ersterer sollte so gewählt werden, dass der Ball einen möglichst hohen Reibungskoeffizienten besitzt und demnach wenig rutscht. Beim Flugverhalten ist es essentiell, dass dieses möglichst stabil ist. Nur wenn externe, nicht abzuschätzende Einflüsse gering sind, kann die Flugbahn vorhergesagt werden. Neben der Forderung nach einer möglichst optimalen Kugelform spielt das Gewicht des Balls eine wichtige Rolle (s. auch 3.2.4). Externe Einflüsse, meist in Form von Luftströmungen (Wind, Zug etc.) verursachen eine Kraft auf den Ball, welche sich bei geringerem Gewicht in einer stärkeren, schwer abzuschätzenden Beschleunigung äußert (3.5).

$$a_{ball} = F_{extern}/m_b \tag{3.5}$$

Des weiteren muss der Ball gut durch einen Menschen zu handhaben und zu werfen sein. Dementsprechend muss er eine angenehme Größe und Beschaffenheit besitzen.

3.2.3 Spielgenauigkeit in Abhängigkeit des Ball- und Schlägerradius

Maßgeblich für die Spielgenauigkeit des Robotersystems, das heißt die Präzision, mit der ein zugeworfener Ball in eine bestimmt Richtung weiter gespielt werden kann, sind mehrere Größen. Neben der Software und den Aktoren, auf die im weiteren Verlauf der Arbeit noch genauer eingegangen wird, ist die Dimensionierung des Schlägers entscheidend. Um den Einfluß besser abschätzen zu können, bietet es sich an, den möglichen maximalen Fehler des Reflexionswinkels des Balls in Abhänigkeit von der Positionierungsungenauigkeit des Schlägers zu betrachten. Hierzu wird der Soll-Reflexionswinkel einer Schläger-Ball-Kollision mittels der Annahme Einfallswinkel gleich Reflexionswinkel berechnet. Effekte wie Rotation, Reibung und nicht vollständig ellastisches Verhalten des Balls werden dabei ignoriert.



(a) Ausgangssituation mit den benötigten Größen

(b) Auftreffpunkt tangential zur Oberfläche um d Meter verschoben

Abbildung 3.3: Testkonfiguration zur Bestimmung der Treffgenauigkeit

Der Ball fliegt in diesem Szenario parallel zur x-Achse auf den Schläger zu. Dort trifft er unter einem Winkel von β auf der Oberfläche auf und prallt mit 2β von dieser ab (Abbildung 3.4(a))

Die Positionierungsungenauigkeit d wird tangential zur Schlägeroberfläche am Auftreffpunkt angewendet, da dort ihr Einfluß maximal ist. Dann lässt sich mittels Gleichung (3.6) die maximale Winkelabweichung σ vom Sollwinkel für einen gegebenen Radius berechnen. Der Reflexionswinkel ist in diesem Modell immer das doppelte des Einfallswinkels und die Positionierungsungenauigkeit lässt sich durch Normierung auf einen Einheitskreis in einen Winkel überführen (Abbildung 3.4(b)). Der in (3.6) angesetzte Radius r entspricht dabei nicht dem Schlägerradius sondern der Summe beider Radien (3.7).

$$\sigma = |2(\beta + \frac{d}{r}) - 2\beta| \tag{3.6}$$

$$r = r_b + r_s \tag{3.7}$$

In Abbildung 3.4 ist der entstehende Winkelfehler über die Radien-Summe für verschiedene Positionierungsungenauigkeiten aufgetragen. Aus dem Plot lässt sich die Erkenntnis ziehen, dass die Spielgenauigkeit ein großes Problem darstellt. Eine bessere Positionierungsgenauigkeit als 10 mm wird kaum zu erreichen sein, da dieser Wert bei einem Meter Schlägerlänge bereits einem maximal erlbauten Winkelfehler am Gelenk von nur 0.57° entspricht. Auch eine Radien-Summe von mehr als 0.2 m ist schwierig zu realisieren. Die mit einem größeren Radius verbundenen höheren Gewichte von Ball und Schläger werden, wie der nächsten Abschnitt genauer erläutert, schnell zu einem Problem. Unter diesen optimistischen Annahmen liegt der, mittels der beschriebenen Vorgehensweise, ermittelte Fehler für den Reflexionswinkel des Balls bereits bei 5.7°.

3.2.4 Betrachtung der Kollisionen

Kräfte treten im Roboter nicht nur durch die statische Masse des Schlägers auf, sondern zusätzlich durch dynamische Vorgänge. Neben der bereits beschriebenen Eigenbewegung des Schlägers ist die zweite Quelle von Kräften der Kollisionsvorgang zwischen Schläger und Ball. Dieser Kontakt verursacht ein Drehmoment im Gelenk des Schlägers, welches auf die Aktoren wirkt. Da die Belastung der Aktoren so gering wie möglich ausfallen sollte, muss versucht werden, die auf sie wirkenden Kräfte durch geeignete konstruktive Maßnahmen zu reduzieren.

Das Drehmoment, das bei einer Kollision des Balls mit dem Schläger am Gelenk auftritt hängt von mehreren Faktoren ab. Auf der Seite des Balls ist die Geschwindigkeit und die Masse relevant. Beim Schläger hängt das auftretende Drehmoment ebenso von der Masse ab, aber zusätzlich von der Länge des Schläger-Stabs sowie von dessen Steifigkeit. Diese Steifigkeit verursacht, sobald der Ball durch seinen Aufprall eine Auslenkung des Schlägers hervorruft, eine Kraft, die diesen in seine Ausgangsposition zurück treibt.

Um den Einfluss der Schläger- sowie Ballmasse abschätzen zu können, wurde das Kollisionsverhalten zeitdiskret simuliert. Diese Simulation muss nur in einer Raumdimension stattfinden und

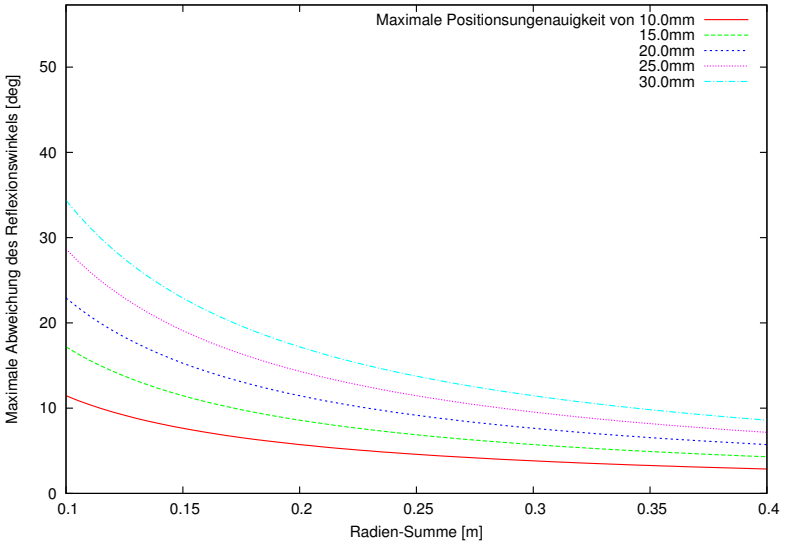


Abbildung 3.4: Vergleich der Abweichungen des Reflexionswinkels für verschiedene Radien-Summen und Positionierungsfehler

beinhaltet zwei wirkende Kräfte. Die erste Kraft entsteht, wenn der Ball beim Auftreffen auf den Schläger zusammengedrückt wird. Das Bestreben des Balls, sich wieder auf seine ursprüngliche Form auszudehenen treibt den Ball und den Schläger auseinander. Die zweite Kraft $F_{\text{schläger}}$ bewirkt die beschriebene Bewegung des Schlägers zurück in seine Ausgangsposition. Sie verhält sich dabei wie eine Feder proportional zur Auslenkung d mit einer Federkonstante c (3.8).

$$F_{\text{schläger}} = cd \tag{3.8}$$

Der genaue Ablauf zur Durchführung der Simulation findet sich ebenso wie die exakten Formeln für die auftretenden Kräfte in Abschnitt 4.4.2. Für die Einordnung der Ergebnisse ist deren Kenntnis jedoch nicht notwendig.

Die Simulationen wurden mit variierenden Annahmen für die Ballmasse und die Federkonstante durchgeführt. Für die Ballmasse wurde dabei ein Bereich von 1 bis 300 g betrachtet, für die Federkonstante zwischen 10 und $100\frac{N}{m}$. Das Gewicht des Schlägers wurde fest mit 0.2 kg angenommen und seine Länge auf 1 m festgesetzt. Der Plot in Abbildung 3.5 zeigt jeweils das maximale, während eines Simulationslaufs aufgetretene Drehmoment in Abhängigkeit der gegebenen Größen am Schlägergelenk an.

Es ist zu erkennen, dass das maximale Drehmoment mit höherer Federkonstante ebenso wie mit der Ballmasse ansteigt. Da das Drehmoment direkt proportional zur auf den Schläger wirkenden Kraft und zur Schlägerlänge ist, steigt dieser Wert mit der Kraft an. Eine höhere Federkonstante verursache demnach eine höhere Kraft, ebenso wie ein schwererer Ball, der eine stärkere Auslenkung des Schlägers von seiner Ausgangsposition hervorruft.

Die Schlägermasse kann mit einem festen Wert angenommen werden, da das resultierende Drehmoment direkt mit den beiden Massen skaliert. Würde beispielsweise die Schläger- sowie Ballmasse verdoppelt werden, hätte auch das Drehmoment den doppelten Betrag. Somit ist nur das Verhältnis beider Werte relevant und es lassen sich aus den Daten problemlos Werte für andere Schlägermassen ableiten.

Es wird deutlich, dass die Schlägermasse ein Vielfaches der Ballmasse betragen sollte, um das auftretende Drehmoment gering zu halten und somit die Aktoren zu schonen. So sollte die Ballmasse im Plot, unabhängig von der konkreten Federkonstante, möglichst vor Beginn des nahezu linearen Verhaltens des Maximaldrehmoments liegen. Daraus resultiert, dass in diesem Beispiel das ungefähre Ballgewicht höchstens 50 g betragen sollte. Unter Berücksichtigung des Skalierungsverhaltens des Drehmoments lässt sich somit ein allgemeingültiger Richtwert festlegen. Dieser besagt, dass das Schlägergewicht idealerweise mindestens das 4-fache des Ballgewichts betragen sollte.

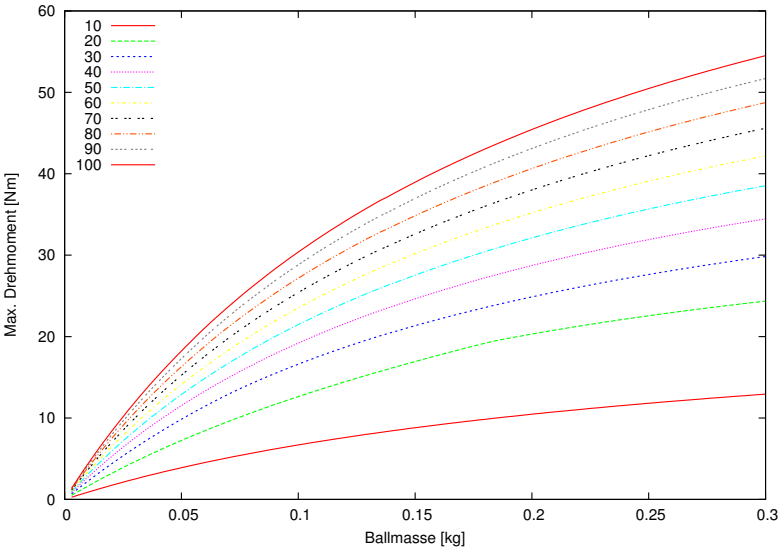


Abbildung 3.5: Maximale Drehmomente bei Ball-Schläger-Kollision für verschiedene Federkonstanten. Schlägermasse 0.2 kg, Schlägerlänge 1 m

3.2.5 Bewertung der Anforderungen

Aus den hier vorgestellten Anforderung und Berechnungen lassen sich nicht auf direktem Weg Werte für alle offenen Parameter ableiten. Dafür ist deren Anzahl zu vielfältig und die Anforderungen oft widersprüchlich. So ist ein längerer Stab zwar von der Endgeschwindigkeit und dem Arbeitsbereich her wünschenswert, verursacht aber eine höhere Belastung der Aktoren. Ähnlich verhält es sich für die Schläger-Kugel. Auch dieser sollte möglichst groß sein, um eine höhere Abspielgenauigkeit zu erreichen und möglichst das 4-fache des Balls oder mehr wiegen, jedoch wiederum unter Beachtung des dadurch entstehenden mehr an Drehmoment für die Aktoren. Der Ball sollte hingegen möglichst klein und leicht ausfallen, aber nicht seine guten Flugeigenschaften verlieren. Andererseits sollte die Radien-Summe von Ball und Schläger wiederum möglichst groß sein, bestensfalls 0.2 m oder mehr, um eine höhere Spielgenauigkeit zu erreichen.

Letztendlich hängen jedoch viele Parameter direkt oder indirekt von den verwendeten Aktoren ab. Sie sollten, unabhängig von anderen Größen, möglichst genau sein, um eine hohe Positionierungsgenauigkeit zu erhalten. Ihr Drehmoment ist ausschlaggebend für viele konkrete Auslegungsentscheidungen. Es bestimmt den Arbeitsbereich und die Schläger-Dimensionierung und damit auch indirekt die Auslegung des Balls. Andere Parameter, wie etwa die Oberflächenbeschaffenheit des Balls oder die Festigkeit des Schlägers lassen sich mit den hier hergeleiteten Anforderungen bereits eng eingrenzen.

Um eine gewählte Auslegung zu überprüfen, können die hier hergeleiteten Kräfte berechnet und mit den Daten der Aktoren abgeglichen werden, sobald diese vorliegen. So ist es möglich, für eine konkrete Dimensionierung der Komponenten das Zusammenspiel und die Tauglichkeit zu überprüfen. Dieses Vorgehen wurde im weiteren Verlauf gewählt.

3.3 Aufbau

Dieser Abschnitt beschreibt die gewählte Auslegung des Robotersystems. Es werden der real verwendete Gehäuseaufbau, die Aktoren und Sensoren beschrieben und aufgezeigt, warum diese Komponenten gewählt wurden. Dafür werden die theoretischen Überlegungen aus Abschnitt 3.2 angewandt. Auch auf den Rechner als zentrale Ausführungseinheit und die Gesamtkosten des Systems wird kurz eingegangen.

3.3.1 Dimensionierung und Aufbau

Der Hardwareaufbau besteht aus einem aktiven Element, dem Schläger sowie dem Unterbau, auf dem dieser zusammen mit den Kameras und weiteren Bauteilen montiert ist. Auch der Ball, den der Roboter spielen soll, lässt sich der Hardware zuordnen.

Schläger

Die Komponente zum Schlagen des Balls besteht aus der Kugel, dem Stab und der Befestigung zur Montage an die Aktoren (s. Abbildung 3.6(a)). Als Material für die Kugel wurde Polystrol[1] gewählt. Es ist leicht und stabil, d.h. verformt sich nur unter massiver Krafteinwirkung, wie sie im normalen Betrieb nicht auftreten sollte. Der Außenradius der Kugel beträgt 0.125 m. Konstruktionsbedingt und zur Erhöhung des Gewichts wurde das Innere mit PU-Schaum ausgespritzt. Insgesamt liegt das Gewicht der Konstruktion bei ca. 230 g.

Der Stab, der die Verbindung zwischen Kugel und Aktoren herstellt besteht aus torsionsstabilem CFK[2]-Rohr [Car11]. Es hat einen Außendurchmesser von 14 mm und einen Innendurchmesser von 11 mm. Das Gewicht beträgt 92 g/m.

Der Aufbau und die relevanten Maße sind Abbildung 3.6(a) zu entnehmen. Die Gesamtlänge des CFK-Rohres l_{rohr} beträgt 0.9 m, der effektive Teil, in dem es sich frei biegen kann jedoch nur $l_{eff} = 0.575$ m. Die Befestigung des Stabs, die zur Montage an die Aktoren dient, misst $l_{bef} = 0.14$ m. Die Länge l_s, die für alle weiteren Berechnungen relevant ist erstreckt sich vom unteren Ende der Befestigung bis zur Mitte der Kugel und beträgt 0.84 m.

Der freiliegende Teil des CFK-Rohres ist von synthetischem Kautschuk[3] mit einer Dicke von 15 mm umhüllt. Dies dient dem Schutz von Personen, die sich während des Betriebs des Roboters unerlaubt in seinem Arbeitsbereich aufhalten.

Ball

Der Ball ist neben dem Schläger die zweite wichtige und aktive Komponente. Die an ihn gestellten Anforderungen bezüglich Gewicht, Elastizität und Handhabbarkeit (Abschnitt 3.2.2) werden am besten durch einen simplen Gummiball erfüllt. Das möglichst geringe Gewicht wird durch die Luftfüllung und den geringen Radius von nur $r_b = 0.0605$ m (60.5 mm) erreicht. Es beträgt für den ausgewählten Ball nur 53 g, was sehr gut mit der externen Komponente zur Erkennung der Ballflugbahnen zusammenspielt. Sie ist für einen Ball mit ähnlichem Radius und 60 g Gewicht ausgelegt und kann somit mit den Störeinflüssen, wie beispielsweise Wind umgehen, die auf einen solchen Ball wirken. Die Gummioberfläche des ausgewählten Balls sorgt für einen ausreichend hohen Reibungskoeffizienten und die Luftfüllung für ein nahezu vollelastisches Stoßverhalten.

Bei physikalischen Berechnungen wird im linearen Fall die Masse $m = 53$ g verwendet, im Falle von Rotationsbewegungen ein Trägheitstensor. Dieser entspricht näherungsweise dem einer Hohlkugel und wird entsprechend Gleichung (3.9) definiert.

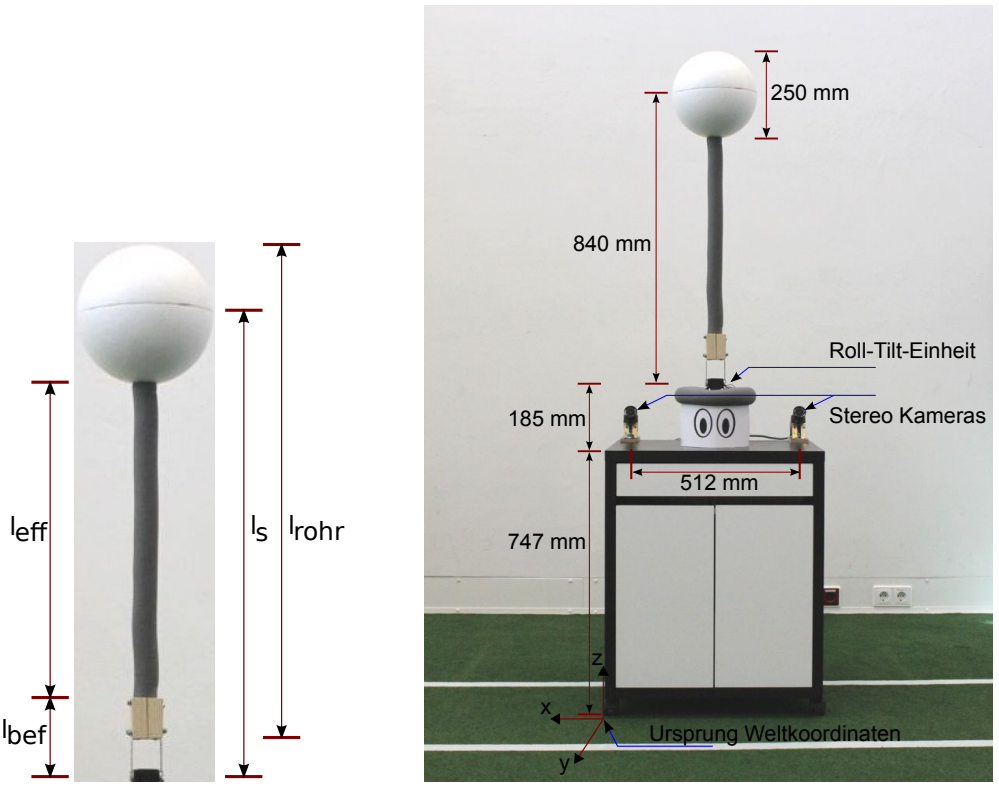
$$I_{ball} = \begin{pmatrix} \frac{2}{3} m_b r_b^2 & 0 & 0 \\ 0 & \frac{2}{3} m_b r_b^2 & 0 \\ 0 & 0 & \frac{2}{3} m_b r_b^2 \end{pmatrix} \tag{3.9}$$

Unterbau

Alle Komponenten des Roboters sind auf oder innerhalb des Unterbaus montiert. So ist der Schläger beispielsweise nahezu mittig auf diesem angebracht. Das Aussehen und die Abmessungen des Gesamtsystems in der aktuellen Ausführung sind Abbildung 3.6(b) zu entnehmen.

Die Schlageinheit ist über die Aktoren (Roll-Tilt-Einheit) an das Gehäuse montiert. Diese sind von einem gepolsterten Schutz umgeben, der ein Greifen in die Mechanik erschwert und zusätzlich einen ansprechenderen Sichtschutz darstellt. Auch oben auf dem Gehäuse befinden sich die beiden Kameras, auf die im weiteren Verlauf des Textes noch genauer eingegangen wird. Um bei der

[1] besser bekannt unter dem Markennamen Styropor
[2] Carbon-faserverstärkter Kunststoff
[3] allg. verwendet zur Isolation von Warmwasserleitungen



(a) (b)

Abbildung 3.6: Aufbau und Abmessungen: (a) des Schlägers und (b) des gesamten Robotersystems

Aufstellung des Systems die Ausrichtung zum Boden reproduzieren zu können, befindet sich auf dem Gehäuse eine 2D-Wasserwaage. Die möglichst waagerechte Ausrichtung zum Boden ist für alle darauf aufbauenden kinematischen Berechnungen essentiell.

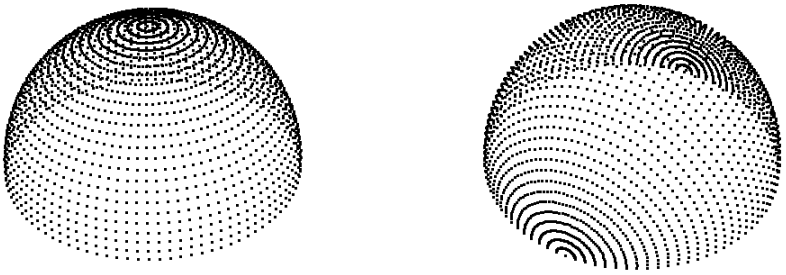
Innerhalb des Gehäuses sind der Rechner, die Stromversorgung für die Aktoren und die gesamte Verkabelung untergebracht. Nur ein Stromkabel muss nach außen geführt werden. Das Gewicht der innenliegenden Komponenten, allen voran des Rechner sorgen für einen stabilen Stand auch bei ruckartigen Bewegungen und Stößen.

Achsenanordnung

Beim Design des Roboters stellt sich die Frage, wie die Gelenkeinheit am besten aufgebaut werden sollte. Die Anforderung besteht darin, dass ein näherungsweise halbkugel-förmiger Arbeitsbereich abgedeckt werden soll. Die optimale Lösung wäre ein Kugelgelenk, das durch einen Motor frei in jede Position bewegt werden kann. Doch konstruktionsbedingt ist ein Aufbau mit zwei in Reihe geschalteten Motoren einfacher umzusetzen und damit für dieses System attraktiver.

Auch bei der Festlegung auf zwei Motoren besteht noch Auswahl zwischen verschiedenen Anordnungen, die ihre spezifischen Vor- und Nachteile haben. So ist hier vor allem die Lage sogenannter Singularitäten interessant. Abbildung 3.7 vergleicht den Arbeitsbereich einer Pan-Tilt-Einheit (Drehen, Kippen) mit dem einer Roll-Tilt-Einheit (Rollen, Kippen). Die Abbildungen wurden mit festen Winkelschritten für beide Gelenke erstellt.

Wie ersichtlich ist, hat die Pan-Tilt-Einheit ihre Singularität mittig oben im Arbeitsbereich, dementsprechend müssen dort sehr große Wege im Drehgelenk zurückgelegt werden, um eine kurze Strecke in der Welt zu überbrücken. Verglichen damit hat die Roll-Tilt-Einheit ihre Singularitäten vorne und hinten im Arbeitsbereich. Dort ist analog ein weiter Weg im Roll-Gelenk für kleine Strecken in der Welt nötig. Da die Singularitäten bei der Roll-Tilt-Einheit am Rand des Arbeitsbereichs liegen, anstatt mittig darin wie bei der anderen Einheit, wurde diese für die Konstruktion gewählt.



(a) (b)

Abbildung 3.7: Vergleich zwischen (a) Pan-Tilt- und (b) Roll-Tilt-Gelenk

Koordinatensysteme

Der Roboter verwendet intern verschiedene Koordinatensysteme, die an wichtigen Positionen des Systems angedockt sind (Abbildung 3.8). Das Weltkoordinatensystem dient als einheitliche Basis zur Ausgabe von Werten und zum Austausch von Daten innerhalb des Systems. Sein Ursprung liegt an der vorderen, rechten Ecke des Systems auf Bodenhöhe (s. Abbildung 3.6(b)).

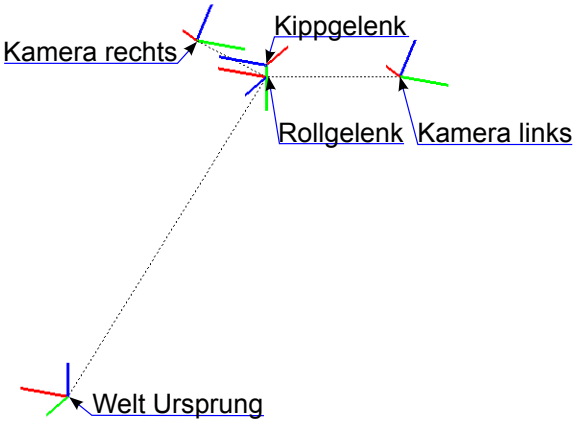


Abbildung 3.8: Lage der verschiedenen Koordinatensysteme (x:rot, y:grün, z:blau)

Zwei weitere Koordinatensysteme liegen auf den Motorachsen. Das Erste befindet sich auf der Achse des Roll-Gelenks, es stellt gleichzeitig die Basis für alle Roboter-bezogenen Koordinatensysteme dar, da es unbewegt ist und relativ zum Weltkoordinatensystem zwar gespiegelte und vertauschte Achsen besitzt aber sonst noch immer achsenparallel ist. Die Anordnung ist darin begründet, dass bei beiden Motorkoordinatensystemen die z-Achse parallel zur Motorachse ausgerichtet wurde.

Das zweite Motorkoordinatensystem liegt auf der Achse des Tilt-Gelenks. Es ist nicht fix gegenüber der Welt, da es sich bei Bewegungen des Rollgelenks mitbewegt. Konstruktionsbedingt ist es gegenüber der Rollachse verschoben (Abbildung 3.9), wodurch sich mehrere Probleme ergeben. Zum einen verliert der Arbeitsbereich seine perfekte Halbkugelform und stellt stattdessen einen halben Ellipsoid dar. Dadurch verkompliziert sich die Kinematik, worauf in Abschnitt 4.2 eingegangen wird. Ausserdem treten Mehrdeutigkeiten auf, wenn der Arbeitsbereich nicht auf einen halben Ellipsoiden[4] beschränkt wird. So wäre es in der Nähe der Singularitäten möglich, eine Weltposition mit verschiedenen Roboterposen, exakt auf der Singularität sogar mit unendlich vielen zu erreichen.

Zwei weitere Koordinatensysteme liegen in den Kameras und sind so ausgerichtet, dass die z-Achse in Sichtrichtung zeigt. Sie haben im Rahmen dieser Arbeit kaum Relevanz, da die Verar-

[4] Der Arbeitsbereich darf auch die Singularitäten nicht enthalten

beitung der visuellen Information innerhalb eines anderen, bereits vorhandenen Moduls erfolgt.

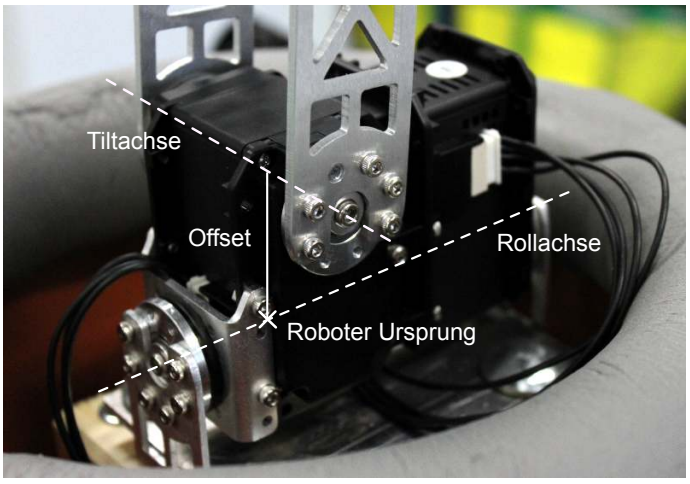


Abbildung 3.9: Detailansicht der Motoren und Gelenkachsen

3.3.2 Aktoren

Bei den beiden eingesetzten Motoren handelt es sich um Dynamixel EX106+ Servos [Rob11a], je einen für die Roll- und Tilt-Achse. Die wichtigsten technischen Daten sind Tabelle 3.1 zu entnehmen. Die Anordnung der beiden Servos zueinander und der daraus resultierende Versatz der Servoachsen wurde bereits in Abbildung 3.9 veranschaulicht.

Maße	40.1 mm x 65.3 mm x 50.1 mm
Gewicht	154 g
Auflösung	0.06°
Getriebe	184 :1, vollständig aus Metall
Leerlaufgeschwindigkeit	91 rpm = $546°/s$
Haltedrehmoment	107 kgf.cm ≈ 10.49 Nm
Max. Haltestrom	7A bei 18.5 V
Betriebsspannung	12 – 18.5 V, 14.8 V empfohlen und verwendet
Anschlussmöglichkeiten	2x RS-485 zum durchschleifen Synchronisation von zwei Servos Versorgungsspannung

Tabelle 3.1: Technische Daten zum Dynamixel EX106+ [Rob11a]

Der Anschluss der Motoren erfolgt über die angebotene RS-485-Schnittstelle. Dabei handelt es sich um einen seriellen Bus, der Differenzspannungspegel verwendet. Er arbeitet als sogenannter Multi-Drop-Bus, d.h. ein Bus mit einem Master, der die Kommunikation anstößt und leitet und einer Vielzahl an Empfängern (Slaves), die nur auf Aufforderung reagieren. Auf der Halbduplex-Kommunikation setzt ein simples Protokoll mit Start- und Stopbits auf, welches die Übertragung von einzelnen Bytes ermöglicht. Ein typischer Vorgang zum Auslesen von Daten vom Slave (S) durch den Master (M) sieht folgendermaßen aus:

`M Anfrage` → `S empfängt` → `S verarbeitet Anfrage` → `S sendet Antwort` → `M empfängt Antwort`

Es wird deutlich, dass bei diesem Vorgang Latenzen sowohl durch die Kommunikation wie auch die Verarbeitung im Slave entstehen. Diese limitieren die mögliche Abfragefrequenz, das sequentielle Auslesen von mehreren Slaves verursacht eine weitere Reduktion der möglichen Abfragen pro Sekunde.

Den Anschluss der Servo-Motoren an den Rechner erledigt ein RS-485 zu USB-Konverter. Hier kommt ein USB2Dynamixel [Rob11b] zum Einsatz, welcher speziell zum Einsatz mit den Servos dieses Herstellers konzipiert ist.

Die Stromversorgung der Motoren übernimmt ein Voltcraft HPS-13015 Labornetzgerät [Vol11]. Es liefert eine einstellbare Spannung von $1 - 30\,\mathrm{V}$ bei maximal $15\,\mathrm{A}$ und ermöglicht demnach die Versorgung der Motoren unter maximaler Last von 2x $7\,\mathrm{A}$.

Die Servos bieten die Möglichkeit verschiedene Werte zum aktuellen Betriebszustand auszulesen. Hier sind vor allem Daten zur Position, Geschwindigkeit, Temperatur und Spannung interessant. Die Auslesefrequenz ist, wie beschrieben, durch die Anfrage-Antwort-Latenz limitiert, jedoch ändern sich nur die Positionsangaben bei jeder Anfrage. Alle anderen Werte werden Servo-Intern nur alle $100\,\mathrm{ms}$ aktualisiert und ändern sich folglich nur in diesem Intervall. Das mag für Spannungs- und Temperaturwerte akzeptabel sein, jedoch nicht für die Bewegungsgeschwindkeit. Diese und die Beschleunigung lassen sich jedoch problemlos wie in (3.10) gezeigt aus den Positions- und Zeitdifferenzen zwischen zwei Auslesevorgängen berechnen.

$$v_t = \frac{p_t - p_{t-1}}{t_t - t_{t-1}} \qquad a_t = \frac{v_t - v_{t-1}}{t_t - t_{t-1}} \tag{3.10}$$

Zwei verschiedene Modi können zur Ansteuerung der Motoren verwendet werden. Sie haben beim Einsatz im hier beschriebenen Robotersystem unterschiedliche Vor- und Nachteile, die im folgenden beschrieben werden.

Positionsregler

In diesem Modus wird der Servo als in sich geschlossener Positionsregler betrieben. Von außen wird ihm eine Sollposition und Maximalgeschwindigkeit vorgegeben und der Regler innerhalb des Servos fährt diese Position an. Ist sie erreicht, hält der Regler die Position eigenständig, bis ein neues Kommando eintrifft.

Der Bewegungbereich ist hier auf $0 - 251°$ beschränkt, es können demnach keine Mehrdeutigkeiten entstehen. Auch die ausgegebenen Positionsangaben sind dementsprechend eindeutig. Die aktuelle Position im Bereich von $0 - 4095$ kann über eine Mittelstellung und einen Skalenfaktor problemlos in Radiant umgerechnet werden, wie in (3.11) dargestellt ist. Diese beiden Werte können durch Kalibrierung ermittelt werden (siehe Kapitel 5) und ermöglichen eine von der Hardware abstrahierte Ansteuerung um eine selbstdefinierte Neutral- oder Nullstellung.

$$p_{\mathrm{SI}} = (p_{\mathrm{Servo}} - p_{\mathrm{middle}})scale \tag{3.11}$$

Dieser einfachen Verwendung steht der gravierende Nachteil der Unflexibilität gegenüber. So lassen sich die internen Reglerparameter nicht beeinflußen und demnach nicht an den Anwendungsfall anpassen. Wenn die Vorgaben, wie es hier der Fall ist, mit der schwingenden Masse des Schlägers Probleme haben, besteht die einzige praktikable Lösung in einer Reduktion der maximalen Bewegungsgeschwindigkeit. Dies läuft jedoch dem Ziel eines schnellen, interaktiven Spiels zuwider.

Drehmomentsteuerung

Die Verwendung der Drehmomentsteuerung (auch Torque-Control oder Wheel Mode genannt) erfordert mehr Aufwand. Hier hat man die Möglichkeit, die Regelschleife aus dem Servo heraus bis in den Steuerrechner zu verlängern. Der Rechner erhält nur noch einen Ist-Positionswert und führt darauf zusammen mit der ihm bekannten Sollposition einen beliebigen Regelalgorithmus aus. Dieser berechnet einen Drehmoment-Wert, der an den Servo zurückgesendet und von diesem umgesetzt wird.

Der Name Wheel Mode kommt daher, dass der Servo keine Begrenzung der Position mehr besitzt, d.h. er seine Achse frei und beliebig drehen kann. Es ist nicht mehr möglich eine absolute Position zu erhalten und es besteht die Möglichkeit zu Mehrdeutigkeiten nach einer vollständigen Drehung. Als Position erhält man nur noch die Differenz des internen Drehencoders zwischen dem letzten und dem aktuellen Auslesen des Wertes. Für den Betrieb im entworfenen Roboter ist jedoch eine exakte Ansteuerung einer absoluten Position nötig. Da der Arbeitsbereich des Roboters

nur 180° beträgt, lassen sich Mehrdeutigkeiten durch vollständige Umdrehungen rein mechanisch effektiv ausschließen.

Bleibt das Problem bestehen, dass die Position des Servos nach dem Anlegen der Versorgungsspannung unbekannt ist. Da dieser jedoch immer im Positionsregler-Modus startet, kann dort initial die absolute Position ausgelesen und nach einer Umschaltung auf Drehmomentsteuerung über die relativen Angaben weitergezählt werden. (3.12).

$$p_{\mathrm{SI}} = p_{\mathrm{reference}} + 0.06 \cdot p_{\mathrm{relative}} \tag{3.12}$$

Der Vorteil dieser Lösung ist eindeutig die Möglichkeit, einen beliebigen, passenden Regler für den jeweiligen Anwendungsfall zu wählen und zu optimieren. Zu Problemen führt die hohe Latenz des Regelkreises durch die Übertragung der Daten zum Rechner und zurück zum Servo. Dadurch können leicht Effekte wie Schwingen auftreten, wie später noch beschrieben wird. Ein weiterer, nicht zu unterschätzender Nachteil ist das Verhalten bei Ausbleiben von Steuerkommandos. Solch ein Fall kann auftreten, wenn das Steuerprogramm auf dem Rechner sich unsachgemäß beendet. Dann dreht der Servomotor weiter, bis er in die mechanische Limitierung fährt was schlimmstenfalls zu Schäden der Hardware führen kann. Menschliche Schäden sollten durch einen geeigneten Aufbau und den Einsatz von weichen und flexiblen Materialien auch dann ausgeschlossen sein.

3.3.3 Kameras

Für die optische Erfassung der Ballflugbahn sind auf dem Roboter zwei AVT F-046C Kameras von Allied Vision Technologies verantwortlich [All11]. Sie nehmen in der eingesetzten Konfiguration Bilder mit einer Auflösung von 780 x 580 Punkten und 50 Hz auf. Die Bilder werden als 8-Bit Graustufenbild per Firewire an den Rechner gesendet. Dort sind zum Bewältigen der Datenmenge zwei getrennte Firewire-Controller für den Empfang verantwortlich. Damit die Bilder möglichst exakt zum gleichen Zeitpunkt aufgenommen werden und somit die Korrelation von Bildinformationen erleichtert wird, sind die Kameras mit einem speziellen Trigger-Kabel verbunden. Es ermöglicht einer der Kameras, der anderen den Start einer Bildaufnahme mitzuteilen, damit diese zeitgleich starten kann.

Auf den Kameras sind Objektive mit einer Brennweite von 4.8 mm montiert. Dadurch beträgt der horizontale Öffnungswinkel ca. 57°. Durch die Anbringung auf dem Robotergehäuse mit ca. 35° Neigung nach oben können die Kameras den Ball möglichst lange im Flug beobachten. Da auf dem Boden zumeist nichts für das Spielkonzept interessantes passiert, ist es ideal, den unteren Rand des Sichtbereichs nahezu parallel zum Boden auszurichten. Der Abstand der Kameras zueinander, die Baseline, ist mit 0.52 m relativ groß, was eine bessere Gewinnung von Tiefeninformationen ermöglicht. Die exakte Position zueinander sowie die Transformation zwischen Kameras und Roboter-Basis-Koordinatensystem wird mittels Kalibrierung ermittelt. Der genaue Ablauf wird in Abschnitt 5.1 beschrieben.

Im Rahmen dieser Arbeit wird das Verfahren zur Rekonstruktion von Informationen aus den aufgenommenen Bildern keine weitere Rolle spielen. Diese Verarbeitung erledigt eine eigenständige Komponente, auf die, wenn nötig zugegriffen wird.

3.3.4 Kräfte bei gewählter Auslegung

Die ausgewählte Hardware kann nun mit den in Abschnitt 3.2 hergeleiteten Formeln auf ein Zusammenspiel innerhalb der Toleranzen überprüft werden. Damit lässt sich rechnerisch abschätzen, ob das Design fehlerfrei und ohne Schäden zu nehmen arbeiten kann. Da bereits beim Herleiten der hier verwendeten Formeln viele Vereinfachungen eingebracht wurden, dürfen die folgenden Angaben nur als grober Richtwert genommen werden.

Zuerst wird das Haltedrehmoment überprüft. Dazu wird (3.1) verwendet und die in diesem Abschnitt angegebenen Werte eingesetzt um (3.13) zu erhalten. Für das Gewicht wird das Gesamtgewicht von Befestigung, Stange und Schläger verwendet, um eine Abschätzung nach oben zu erhalten. Der so errechnete Wert liegt weit unter dem maximalen Haltedrehmoment des Motors

von 10.94 Nm.

$$\begin{aligned} M &= l_s m_s g \\ &= 0,84 \cdot 0,335 \cdot 9,81 \\ &= 2.761\,\mathrm{Nm} \end{aligned} \tag{3.13}$$

Auch das Drehmoment, das aus Beschleunigungen heraus entsteht kann mit den Motordaten abgeglichen werden. Dazu wird (3.4) herangezogen und die entsprechenden Werte eingesetzt. Als maximale Winkelbeschleunigung wird $1800°/s^2$ $(= 31,42\frac{rad}{s^2})$ angesetzt. Der Wert ist durch Messungen am realen System ermittelt worden. Das aus (3.14) resultierende Drehmoment von 7.953 Nm liegt immer noch deutlich unter dem Maximum, das für die Motoren angegeben ist. Allerdings kann die Winkelbeschleunigung auch kurzzeitig höher liegen, jedoch nur wenige Milli- bis Zehn-Millisekunden. In diesem Fall ist anzunehmen, dass die Trägheit des Schlägers zusammen mit der leichten Flexibilität des Stabes einen Teil der Beschleunigung abfängt. Dadurch würde nicht das volle Gewicht des Schlägers wirken und dementsprechend das Drehmoment reduziert werden.

$$\begin{aligned} M &= J\alpha \\ &= (\frac{2}{5} m_s r_s + m_s l_s^2)\alpha \\ &= (\frac{2}{5} 0.335 \cdot 0.125 + 0.335 \cdot 0.84^2) \cdot 31.42 \\ &= 7.953\,\mathrm{Nm} \end{aligned} \tag{3.14}$$

Das Füllen der in Abschnitt 3.2.3 vorgestellten Berechnungen zur Treffgenauigkeit mit Werten ist hier leider nicht möglich. Es fehlen die Angaben zur Positionierungsgenauigkeit sowie eine Abschätzung des durch Schwingungen der Konstruktion verursachten maximalen Fehlers. Allerdings geben die dort erstellten Graphen auch in der derzeitigen Form eine gute Richtlinie zum Einschätzen des Systemverhaltens. Zusätzlich kann jetzt der Wert für die Summe der beiden Radien $r = 0.125 + 0.0605 = 0.1855\,\mathrm{m}$ genauer im Plot in Abbildung 3.4 in Betracht gezogen werden.

Für das Kollisionsverhalten zwischen Schläger und Ball müssen die Massen sowie ihr Verhältnis zueinander betrachtet werden. Der Ball hat bei der gewählten Auslegung das 4.3-fache der Schlägermasse und liegt damit über der aufgestellten Grenze von vier. Je nach Stärke der angenommenen Federkonstante liegt das durch den Auprall verursachte Drehmoment zwischen 4 und 19 Nm. Diese Werte liegen teilweise über dem Maximaldrehmoment der Motoren. Aufgrund der extremen Kürze des Kontakts kann jedoch angenommen werden, dass dieser Stoßvorgang keine Probleme verursachen wird.

3.3.5 Rechner

Die vollständige Steuerungssoftware läuft auf einem Standardrechner, der im Robotergehäuse untergebracht ist. Er besitzt als CPU einen Intel Core i7 860 Processor mit 4 Kernen zu je 2.80 GHz. Die RAM-Ausstattung beträgt 4 GB. Ausserdem besitzt er die besagten zwei Firewire-Anschlüße.

Als Betriebssystem kommt Ubuntu 10.04 LTS zum Einsatz. Der Rechner kann regulär mit Bildschirm und Eingabegeräten betrieben werden, aber auch komplett ohne diese Ein- und Ausgabegeräte, was für den angedachten Einsatzzweck erforderlich ist. Auch eine Administration über Netzwerk ist möglich, etwa zur Fehlersuche oder Ferndiagnose.

Die weiteren Details zur Ausstattung und Konfiguration des Systems sind hier nicht weiter relevant, da das System weitgehend unabhängig von der im Detail eingesetzten Hard- und Softwarebasis ist.

3.3.6 Kosten des Systems

Die Kosten des Gesamtsystems müssen betrachtet werden, da der Roboter für einen Einsatz als Event-Modul entworfen ist. Dort wäre es nicht rentabel und damit inakzeptabel, wenn das System zu teuer wird. Die exakte Grenze für einen rentablen Einsatz müsste unter Betrachtung des

Anschaffungspreises, der Wartungs- und Personalkosten sowie der zu erwartenden Einnahmen bestimmt werden. Aufgrund der Komplexität und der vielen Unwägbarkeiten dieser Abschätzung wurde das System mit einer Obergrenze von ca. 10 000 € entwickelt.

Die reellen Kosten finden sich grob abgeschätzt in Tabelle 3.2. Es handelt sich dabei nur um Richtwerte, da die eingesetzten Komponenten noch in gewissem Maße variieren können. Da die so errechneten Gesamtkosten weit unterhalb der selbst gesteckten Grenze liegen, soll diese Abschätzung hier ausreichend sein.

Komponente	**ca. Kosten**
Rechner	1 000 €
Sensoren	1 500 €
Aktoren	1 000 €
Sonstige Hardware	500 €
Summe	**4 000 €**

Tabelle 3.2: Kostenabschätzung zur eingesetzten Hardware

Neben den reinen Kosten für die Hardware-Komponenten müssen auch die Betriebskosten möglichst niedrig gehalten werden. Das wird erreicht, indem am Aufstellungsort keine weiteren Modifikationen vorgenommen werden müssen. Die Umgebung des Roboters muss weder speziell ausgeleuchtet, mit Geräten bestückt noch anderweitig angepasst werden. Das System ist in sich abgeschlossen und muss nur ebenerdig in einem geeignet großen Raum aufgestellt und an die Stromversorgung angeschlossen werden. Die Aufstellung ist so problemlos in kurzer Zeit möglich und sollte Aufgrund der Einfachheit dieses Vorgangs von der im laufenden Betrieb ohnehin nötigen betreuenden Person zu erledigen sein.

4 Steuerung

Dieses Kapitel erläutert die Software, die auf dem Roboter Verwendung findet. Nach der Beschreibung der verwendeten Fremd-Software folgt eine Einführung in die Kinematik des Systems. Anschließend wird die grundlegende Bewegungssteuerung, die mit der Hardware kommuniziert und diese ansprechbar macht vorgestellt. Auf die Entscheidungsfindung, wie ein Ball optimal gespielt werden kann, geht der Abschnitt zur Bewegungsplanung detailliert ein. Eine Gesamtübersicht über das Zusammenspiel der Softwarekomponenten folgt im letzten Abschnitt dieses Kapitels.

4.1 Softwarebasis

Die Software des Roboters basiert zum Teil auf bereits existierenden Bibliotheken und Frameworks. Die zwei wichtigsten, nicht im Rahmen dieser Arbeit entwickelten sind das *Robot Operating System* und der BallTracker. Beide werden im Folgenden vorgestellt. Zusätzlich wurde QT [Nok11] für alle GU-Interfaces und das Boost-Unittest-Framework [Gen11] für automatisierte Tests verwendet.

4.1.1 Robot Operating System

Das *Robot Operation System* (ROS) [Wil11] wird von Willow Garage, der Stanford-Universität und einer aktiven Entwicklergemeinde erweitert und gepflegt. Es dient zum Aufbau von modularen Systemen innerhalb eines Rechners oder über Rechnergrenzen hinweg. Die Vielzahl an fertigen und freien Modulen aus dem Umfeld der Robotik machen es besonders für diesen Bereich attraktiv. Der Einsatz in den verschiedensten kleinen und großen Projekten macht es zum Quasi-Standard für die transparente Kopplung von Modulen in Robotersystemen.

Grundlagen

ROS unterstützt neben C++ und Python weitere Programmiersprachen, jedoch ist hier alleinig C++ relevant, da alle Programmteile in dieser Sprache erstellt wurden. Auch liegen dem Software-Paket eine Reihe weiterer Programme zum Debugging, Visualisieren oder zur Vereinfachung von Routinearbeiten bei. Auch diese spielen im Rahmen dieser Arbeit keine weitere Rolle.

Die Kommunikation der einzelnen Module, im ROS-Jargon Knoten genannt läuft über Nachrichten. Das sind im Textformat definierte Strukturen aus beliebig geschachtelten elementaren Datentypen. Diese werden zur Kommunikation der Knoten untereinander mit den anwendungsspezifischen Daten gefüllt, in einen binären Datenstrom serialisiert und anschließend über Sockets an den Bestimmungsort verschickt. Damit sich einzelne Knoten anhand von symbolischen Bezeichnern finden können, muss jedes auf ROS basierende System einen Kommunikations-Master (*roscore*) besitzen, der die Zuordnung von Namen zu Rechnern und darauf laufenden Prozessen herstellt.

Kommunikation

Zur Kommunikation von Knoten untereinander bietet ROS zwei Modi. Der eine besteht aus simplen Remote-Procedure-Calls (RPCs), bei denen ein Kommunikationsteilnehmer eine nach aussen sichtbare Funktion genau eines anderen Kommunikationsteilnehmers aufruft und von diesem synchron das Resultat erhält.

Der zweite Kommunikationsmechanismus dient zur Verbreitung von Daten von einer Quelle an beliebig viele Empfänger. Die Daten müssen nicht periodisch anfallen, allerdings ist dies häufig der Fall. Die Quelle muss die Empfänger nicht kennen, sondern ROS kümmert sich vollständig und transparent um die Verteilung an alle angemeldeten Interessenten. Dieses Schema ist üblicherweise als Publisher-Subscriber-Modell bekannt. In ROS wird dies so umgesetzt, dass ein Empfänger sich für ein Nachrichten-*Thema* anmeldet, dies intern[1] über den Master der Quelle mitgeteilt wird und diese fortan die Nachrichten über die beschriebenen Mechanismen zusätzlich an den neuen Empfänger übermittelt.

Ein ROS-Programm muss auf asynchron eintreffende Nachrichten einer abonnierten Quelle oder auf eingehende RPCs möglichst zeitnah reagieren. Zu diesem Zweck besteht ein ROS-Programm üblicherweise aus einer Hauptschleife, in der regelmäßig auf neue Ereignisse abgefragt wird und andere wiederkehrende Aufgaben bearbeitet werden können.

Vor- und Nachteile

ROS ist nicht ohne Grund zum Quasi-Standard avanciert. Es ist einfach zu verwenden und nimmt dem Entwickler viel Arbeit ab. Durch seine Abstraktion der Kommunikation muss der Entwickler sich keine Gedanken mehr darüber machen, mit welchen Knoten auf welchen Rechnern ein Programm Daten austauscht. Auch die aktive Entwicklergemeinde und die Vielzahl an daraus hervorgegangenen freien Modulen aus dem Robotik-Bereich haben sicherlich zur Verbreitung beigetragen. Die Geschwindigkeit, zumindest der C++-Implementierung ist durch Vorkompilierung der Nachrichtenbeschreibungen und die eingesetzte Binär-Serialisierung relativ schnell.

Doch ROS hat als sehr generischer Ansatz auch mit Problemen zu kämpfen. So ist vor allem das Zeitverhalten nicht vorhersagbar. Für harte Echtzeit-Kommunikation ist es nicht vorgesehen und kann somit auch keine Garantien für Latenzen geben. Zusätzlich wird beispielsweise durch den Einsatz von Sockets auch zur Rechner-internen Kommunikation viel Performance gegenüber einer Shared-Memory-Implementierung verschenkt. Ein weiterer Nachteil ist die Teils knappe bis nicht vorhandene Dokumentation der Schnittstellen und vor allem der internen Arbeitsweise.

4.1.2 BallTracker

Die folgenden Informationen zum BallTracker sind als Ergänzung zu Abschnitt 2.2 zu verstehen. Hier werden Details erläutert, die die Implementierung, Verwendung und den praktischen Einsatz betreffen. Der interne Aufbau des Balltrackers ist in Abbildung 4.1 dargestellt. Die Komponente besteht aus mehreren, unabhängigen ROS-Knoten, die über Nachrichten miteinander kommunizieren. So werden die Bilder der Kameras nach dem Auslesen und Konvertieren an den Kreiserkenner gesendet. Dieser wiederum schickt die Liste der pro Frame erkannten Kreise an den MHT. Dort werden die erkannten Kreise beider Kameras fusioniert, mit den internen Zuständen abgeglichen und die erkannten Tracks an die Robotersteuerung gesendet. Alle diese Vorgänge werden mit 50 Hz ausgeführt, was der Framerate der Kameras entspricht. Der rückgerichtete Pfeil des MHTs zu den Kreiserkennern dient zum Verfeinern der Suche an den Stellen, an denen laut Modell Kreise vermutet werden.

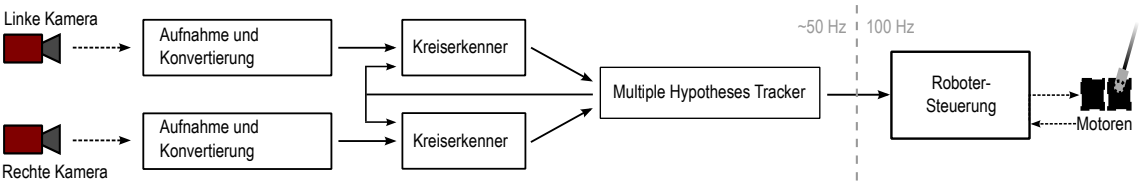


Abbildung 4.1: Schematischer Aufbau des Balltrackers mit angrenzender Hard- und Software

[1] mittels XMLRPC

Kalibrierung

Wie bereits erläutert, verwendet die Komponente intern eine Reihe von Parametern, die an den jeweiligen Anwendungsfall angepasst werden müssen. Die Kameras werden nach dem üblichen Verfahren mittels Schachbrett-Muster kalibriert, um die exakte Transformation zueinander sowie die intrinsischen Kameraparameter zu erhalten.

Die Modellparameter werden nicht fest vorgegeben, sondern gelernt. Die gesuchten Werte sind die exakte Ausrichtung des Schwerkraftvektors relativ zur Kamera sowie der Bremsfaktor des Balls durch die Luftreibung. Beide Werte können je nach Aufstellungsort oder verwendetem Ball variieren und müssen demnach leicht zu aktualisieren sein. Die Kalibrierung geschieht durch Füllen des Modells mit Schätzwerten und anschließendes mehrmaliges Werfen des Balls in Richtung der Kameras. Aus den aufgezeichneten Tracks können die exakten Parameter berechnet werden.

Unter Verwendung der so gewonnenen Flugbahnen lässt sich zusätzlich die Hand-Auge- bzw. Servo-Kalibrierung durchführen. Bei dieser wird die Transformation zwischen Kamera- und Rollgelenk-Koordinatensystem ermittelt. Dadurch lassen sich Ballflugbahnen, die in Kamerakoordinaten vorliegen, in Bezug zur Aktorik des Roboters setzen. Zusätzlich werden bei dieser Kalibrierung die Parameter der Servomotoren bestimmt. Diese bestehen aus der Position der Mittelstellung sowie einem Skalierungsfaktor. Damit lassen sich Servopositionen entsprechend (3.11) aus der internen Darstellung in SI-Einheiten umrechnen.

Ein weiterer, zu kalibrierender Parametersatz ist das Vorwissen über die Startpositionen und -geschwindigkeiten von Tracks. Auch diese Informationen können aus repräsentativen Würfen gewonnen werden. Dafür ist es nur nötig, mit variierenden Geschwindigkeiten von verschiedenen erlaubten Positionen Bälle Richtung Kamera zu werfen. Dieses so gewonnene Vorwissen enthält über den Start-Geschwindigkeitsvektor auch die Wurfrichtung, wodurch mögliche Tracks vermieden werden, die nicht Richtung Kameras orientiert sind und somit irrelevant wären.

Schnittstellen

Wie Abbildung 4.1 zu entnehmen ist, muss die Robotersteuerung keine Daten in Richtung Tracking-Komponente senden. Sie bekommt in regelmäßigen Abständen die erkannten Tracks mittels des ROS Publisher-Subscriber-Mechanismus geliefert und kann diese verwerten. Solch ein Track besteht aus einer eindeutigen Identifikationsnummer, um ihn auch über mehrere empfangene Nachrichten hinweg identifizieren zu können sowie einer Liste mit Informationen zur Flugbahn. Dort sind jeweils in festen Zeitschritten die Position und Geschwindigkeit zusammen mit den jeweiligen Standardabweichungen eingetragen.

In der Abbildungssequenz 4.2 ist zu erkennen, dass die Genauigkeit der Tracks zunimmt, je länger der Ball verfolgt werden kann. So ist das Ende der in 4.2(a) grün eingezeichneten Flugbahn weit von den Enden der darauffolgenden Bilder entfernt. Erst in Abbildung 4.2(c)-(d) verändert sich die Endposition kaum mehr. Die Steuerungssoftware sollte entsprechend sinnvoll mit diesen Schwankungen umgehen.

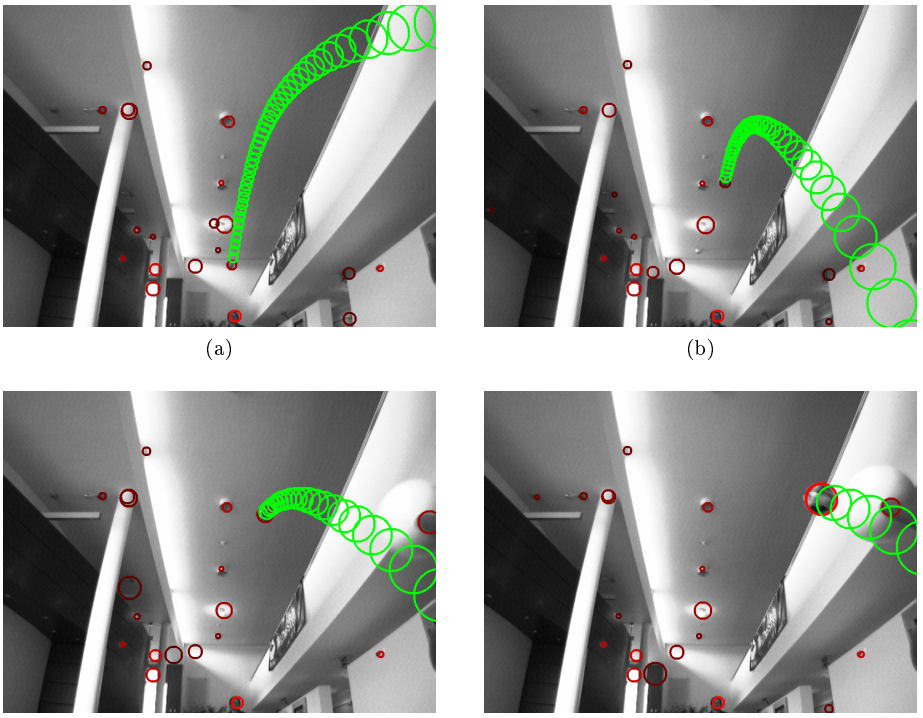
4.2 Kinematik

Die Kinematik des Roboters gliedert sich in Vorwärtskinematik und inverse Kinematik. Mit ersterer wird bei bekannten Gelenkwinkeln ein Punkt aus einem internen Koordinatensystem in Weltkoordinaten transformiert. Die inverse Kinematik hat die umgekehrte Aufgabe. Mit ihr wird ein Punkt in Weltkoordinaten auf die Gelenkwinkel zurückgeführt, die den Roboter-Endeffektor, also die Schlägermitte an diese Position bringen.

Eine weitere, die Kinematik betreffende Aufgabe, ist das Zerlegen von auf den Schläger wirkenden Kräften auf die einzelnen Gelenke. Auch diese Operation ist von den Gelenkwinkeln abhängig und wird im weiteren Verlauf von der Robotersteuerung benötigt.

4.2.1 Vorwärtskinematik

Die Vorwärtskinematik arbeitet mit verketteten Transformationsmatrizen, die jeweils von einem Koordinatensystem in ein direkt angeschlossenes umwandeln. Die relevanten Koordinatensysteme



(a) (b) (c) (d)

Abbildung 4.2: Ballflugbahn aus Kamerasicht. Rot: erkannte Kreise, grün: Vorhersage

wurden bereits in Abbildung 3.8 dargestellt. Der räumliche Bezug des Weltkoordinatensystems wird aus Abbildung 3.6(b) deutlich. Die einzelnen Transformationen sind im Folgenden angegeben und erläutert.

Die Matrix $M_{\text{rollInWorld}}$ (4.1) dient zur Umwandlung von Koordinaten im System des Roll-Gelenks in Weltkoordinaten. Hierfür müssen die Achsen entsprechend vertauscht werden, da die Gelenkkoordinatensysteme per Konvention mit ihrer z-Achse entlang der Gelenkachse orientiert sind. Ausserdem enthält sie die Translation zwischen beiden Systemen.

$$M_{\text{rollInWorld}} = \begin{pmatrix} 1 & 0 & 0 & -0.305 \\ 0 & 0 & 1 & -0.23 \\ 0 & -1 & 0 & 0.898 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.1}$$

Ein einzelnes Gelenk wird durch die Matrix $M_{\text{joint}}(\theta)$ (4.2) beschrieben. Sie hat als Parameter den Gelenkwinkel θ. Es kann für beide Gelenke die gleiche Transformation verwendet werden, da beide Koordinatensysteme immer gleich am Gelenk orientiert sind und die Rotation und Translation zwischen den Gelenksytemen in eine eigene Matrix ausgelagert wurde. So ergibt sich, da die z-Achse immer der Rotationsachse entspricht, eine reguläre Rotationsmatrix.

$$M_{\text{joint}}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.2}$$

Die Umwandlung von Koordinaten aus dem Tilt- ins Roll-System übernimmt $M_{\text{tiltInRoll}}$ (4.3). Sie dreht die Achsen entsprechend und enthält auch den Versatz l_{off} zwischen beiden Gelenkachsen.

$$M_{\text{tiltInRoll}} = \begin{pmatrix} \cos\frac{\pi}{2} & 0 & \sin\frac{\pi}{2} & 0 \\ 0 & 1 & 0 & l_{\text{off}} \\ -\sin\frac{\pi}{2} & 0 & \cos\frac{\pi}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.3}$$

Um nun einen Punkt aus dem Tilt-Koordinatensystem, in dem sich beispielsweise die Schlägerposition simpel und statisch beschreiben läßt, in Weltkoordinaten umzuwandeln ist folgende Verkettung (4.4) der vorgestellten Transformationsmatrizen nötig.

$$\begin{aligned} \vec{p}_{\text{world}} &= M_{\text{rollInWorld}} \cdot M_{\text{joint}}(\theta_{\text{roll}}) \cdot M_{\text{tiltInRoll}} \cdot M_{\text{joint}}(\theta_{\text{tilt}}) \cdot \vec{p}_{\text{tilt}} \\ &= M_{\text{tiltInWorld}}(\theta_{\text{roll}}, \theta_{\text{tilt}}) \cdot \vec{p}_{\text{tilt}} \end{aligned} \tag{4.4}$$

4.2.2 Inverse Kinematik

Die inverse Kinematik wandelt eine Weltposition in Gelenkwinkel um. Hier bietet sich eine analytische Lösung, die bei beliebigem Abstand der gegebenen Koordinate zum Arbeitsbereich passende Winkel für beide Gelenkte liefert. Passend bedeutet hier, dass der Schläger mit diesen Winkel nächstmöglich an den gesuchten Punkt gebracht werden kann. Die Problematik bei der Erstellung einer solchen Lösung liegt im Versatz der beiden Gelenkachsen. Ohne diesen wäre die vom Schläger erreichbare Fläche eine Kugel und die Berechnung könnte mittels der Formeln für Kugelkoordinaten erfolgen. Unter Berücksichtigung des Versatzes entspricht sie der Oberfläche eines Ellipsoiden, wodurch aufwendigere Berechnungen notwendig werden.

Zu beachten ist, dass die analytische Lösung aufgrund der Mehrdeutigkeiten um die Singularitäten herum nicht für den vollständigen ellipsoiden Arbeitsbereich definiert ist. Dies ist jedoch unkritisch, da der interessante Bereich sich nur bei Tilt-Winkeln von $(-90°, 90°)$ befindet und damit die Singularitäten nicht enthält.

Der erste Schritt der Berechnung besteht darin, die Position in Weltkoordinaten $\vec{p}_{\text{world}}$ auf den Nullpunkt des Roll-Koordinatensystems zu beziehen (4.5), jedoch nur unter Anwendung der Translation, nicht der Rotation. Aus diesem Punkt $\vec{p}_{\text{base}}$ können im Folgenden die beiden gesuchten Winkel berechnet werden.

Der Roll-Winkel θ_{roll} wird entsprechend (4.6) bestimmt. Abbildung 4.3(a) verdeutlicht den Vorgang. Wie dort gezeigt, kann der Winkel direkt aus der x- und z-Komponente des Punktes berechnet werden, also indem der Punkt auf eben diese x-z-Ebene projiziert wird.

Die Berechnung von θ_{tilt} (4.7) ist komplexer, da sich hier die Projektionsebene mit dem Roll-Winkel dreht. Um den gesuchten Winkel zu erhalten, wird wie in Abbildung 4.3(b) gezeigt der Punkt auf die durch die Roll-Achse und die Orthogonale zur Tilt-Achse aufgespannten Ebene projiziert. Die projizierte Länge l_{proj} muss noch um den Offset der Achsen zueinander l_{off} bereinigt werden, da der Punkt in Base-Koordinaten diesen Offset noch beinhaltet. Anschließend kann der gesuchte Winkel direkt berechnet werden.

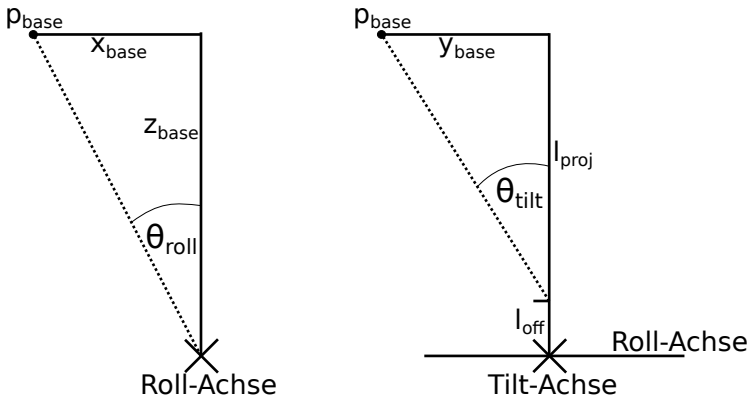
$$\vec{p}_{\text{base}} = \vec{p}_{\text{world}} - M_{\bullet,4_{\text{rollInWorld}}} \tag{4.5}$$

$$\theta_{\text{roll}} = \text{atan2}(x_{\text{base}}, z_{\text{base}}) \quad \text{für } z_{\text{base}} > 0 \tag{4.6}$$

$$\begin{aligned} \theta_{\text{tilt}} &= -\arctan\frac{y_{\text{base}}}{l_{\text{proj}} - l_{\text{off}}} \\ &= -\arctan\frac{y_{\text{base}}}{\sqrt{x_{\text{base}}^2 + z_{\text{base}}^2} - l_{\text{off}}} \end{aligned} \tag{4.7}$$

4.2.3 Kräfteverteilung

Für die im nächsten Abschnitt beschriebene Motorregelung ist es wichtig, welche Drehmomente alleine durch das Gewicht des Schlägers auf die Motoren wirkt. Hierbei geht es alleinig um die statischen, d.h. im unbewegten Zustand wirkenden Kräfte. Die bei Bewegung von der Trägheit hervorgerufenen Belastungen sind aussen vor. Erschwert wird die Berechnung durch die Anordnung



(a) Roll-Winkel (b) Tilt-Winkel

Abbildung 4.3: Berechnung der Gelenkwinkel mittels inverser Kinematik

als Roll-Tilt-Gelenk, da hierbei die Gelenkwinkel des einen Motors die Kräfte auf den jeweils anderen beeinflussen.

Die Berechnung findet auf dem vereinfachten Modell einer Punktmasse an einem masselosen Stab statt. Diese Annahme ist aufgrund des Roboteraufbaus gerechtfertigt. In einem solchen Fall berechnet sich das Drehmoment nach (4.8) mit l als der Stablänge, F der an seinem Ende wirkenden Kraft und θ als dem Winkel zwischen Kraft und Stab.

$$M = lF \sin\theta \tag{4.8}$$

Das Drehmoment am Roll-Gelenk (4.9) wird berechnet, indem die effektive Länge des Stabes mittels θ_{tilt} korrigiert und anschließend der Offset zwischen beiden Achsen addiert wird. Dies ist notwendig, da die Masse sich mit größerem Tilt-Winkel der Roll-Achse annähert, bis zu l_{off} bei 90° tilt.

Ähnlich verhält es sich bei der Drehmoment-Berechnung des Tilt-Gelenks (4.10). Hier wird die Kraft F mittels des Roll-Winkels korrigiert, da mit steigendem Roll-Winkel die effektive Kraft auf das Tilt-Gelenk abnimmt.

$$M_{\text{comp}_{\text{roll}}} = (l_s \cos\theta_{\text{tilt}} + l_{\text{off}}) F \sin\theta_{\text{roll}} \tag{4.9}$$

$$M_{\text{comp}_{\text{tilt}}} = l_s F \cos\theta_{\text{roll}} \sin\theta_{\text{tilt}} \tag{4.10}$$

4.3 Bewegungssteuerung

Die Bewegungssteuerung bietet eine einfache und kompakte Schnittstelle zur Ansteuerung der Motoren. Sie besteht aus mehreren Komponenten, die ihrerseits über klare Schnittstellen miteinander kommunizieren. Der schematische Aufbau ist in Abbildung 4.4 veranschaulicht.

Wie dort zu sehen ist, werden die Motordetails direkt abstrahiert, um darauf einen eigenen Regler zu implementieren. Diesen kann man direkt über die Schnittstelle ansteuern, oder optional auf einen Mechanismus zur Generierung von Bewegungsmustern zurückgreifen. Eine detaillierte Beschreibung der einzelenen Bestandteile wird im Folgenden gegeben.

4.3.1 Hardwareabstraktion

Es ist wichtig, die Eigenheiten der Hardware, hier konkret die der Motoren möglichst früh zu verstecken und einen einheitlichen Zugriff auf diese Ressource zu bieten. So können mit minimalem Aufwand Parameter der Hardware verändert werden, etwa im Zuge einer Kalibrierung oder sogar die Hardware vollständig durch eine ähnliche ersetzt werden. Diese Aufgabe übernimmt innerhalb der Bewegungssteuerung die direkt auf den Motoren aufsetzende Hardwareabstraktion.

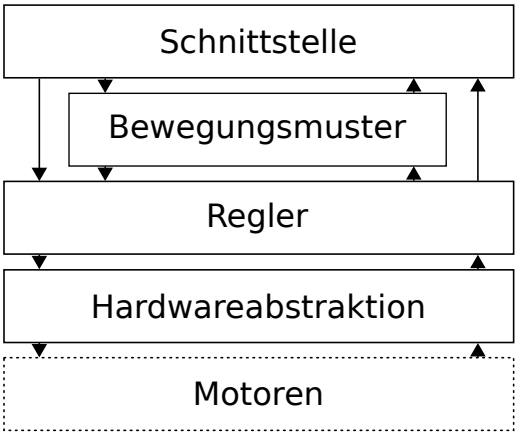


Abbildung 4.4: Schematischer Aufbau der Bewegungssteuerung

Sie ist unter anderem dafür zuständig, die USB-Verbindung zum USB2Dynamixel-Adapter zu verwalten, über den auf die Servos zugegriffen wird. Auch die Initialisierung der Servos, d.h. das Überprüfen und gegebenenfalls Einstellen von benötigten Parametern ist hier angesiedelt. Wenn zu Testzwecken keine echten Servomotoren benötigt werden, erlaubt es diese Komponente auch, simulierte Servos einzubinden, die sich ausgenommen der Motoreigenheiten wie echte Servos verhalten.

Da die Motoren im Modus zur Drehmoment-Ansteuerung laufen sollen (s. Abschnitt 3.3.2), liefern sie bei einer Bewegung immer nur eine relative Angabe in Form der Positionsänderung gegenüber dem letzten Auslesen. Um dennoch mit absoluten Positionen arbeiten zu können, muss erst ein Referenzwert ermittelt werden. Dafür wird im Positionsregler-Modus die aktuelle Gelenkstellung ausgelesen, mittels (4.11) in Radiant umgerechnet und anschließend als Ausgangswert verwendet. j_{off} und j_{scale} beschreiben die Mittelstellung respektive den Skalierungsfaktor und entstammen der Servo-Kalibrierung (s. Abschnitt 4.1.2). Daraufhin kann mit Hilfe von (3.12) die jeweilige Position berechnet werden und daraus die zugehörige Geschwindigkeit und Beschleunigung bestimmt werden (4.12).

$$\theta_t = (p_{\text{servo}} - j_{\text{off}}) \cdot j_{\text{scale}} \tag{4.11}$$

$$\omega_t = \frac{\theta_t - \theta_{t-1}}{\Delta t}, \quad \alpha_t = \frac{\omega_t - \omega_{t-1}}{\Delta t} \tag{4.12}$$

Auch kann hier unter Zuhilfenahme der Kinematik die Position des Schlägers berechnet werden. Somit liegen alle Größen in hardwareunabhängingen SI-Einheiten bzw. in Weltkoordinaten vor. Das Versenden von Werten an die Servos, beispielsweise um Drehmoment-Werte zu setzen wird in äquivalenter Weise abstrahiert.

4.3.2 Regler

Der Regler setzt auf der Hardwareabstraktion auf und ermöglicht es, die Servos an eine vorgegebene Position fahren zu lassen. Da die Motoren mit Drehmomentsteuerung arbeiten, können keine festen Positionen vorgegeben werden, sondern nur gewünschte Drehmoment-Werte. Somit muss die Regelschleife zum Anfahren und Halten einer Position ausserhalb der Motoren implementiert werden.

Hierfür bietet sich je ein regulärer PD-Regler pro Motor an, der mit 100 Hz läuft. Höhere Regelfrequenzen sind aufgrund der Kommunikations-Latenzen nicht möglich. Die verwendete Gleichung (4.13) berechnet das benötigte Drehmoment, um die gewünschte Position zu erreichen. Dabei sind k_p und k_d die Faktoren für den p- bzw. d-Anteil und e_t die Regelabweichung des jeweiligen Servos zum Zeitpunkt t. Die weiteren Größen werden im Folgenden beschrieben.

$$M_{\text{out}} = k_p\, e_t + k_d\, d_t - M_{\text{comp}} \tag{4.13}$$

D-Filter

Der Parameter d_t wäre in einem regulären PD-Regler die zeitliche Ableitung von e_t. Hier wird zusätzlich ein Tiefpassfilter auf diese Ableitung angewandt, um Frequenzen oberhalb einer gewissen Grenzfrequenz abschwächen zu können. Dies ist nötig, um Schwingungen auszufiltern, die vom Getriebespiel verursacht werden. Entsprechend definiert (4.14) einen Tiefpassfilter auf die zeitliche Ableitung von e_t. Die Filterfrequenz f wird mit 20 Hz unterhalb der experimentell ermittelten Schwingungsfrequenz von ca. 24 Hz angesetzt.

$$d_t = e^{-2\pi f \Delta t} \cdot d_{t-1} + (1 - e^{-2\pi f \Delta t)}) \cdot \frac{e_t - e_{t-1}}{\Delta t} \tag{4.14}$$

Die genaue Ursache für das Getriebespiel, auch Backlash genannt, wird detailliert in Abschnitt 5.3 untersucht.

Schwerkraftkompensation

Es fällt auf, dass der Regler keinen I-Anteil besitzt obwohl durch das sehr variable Drehmoment, welches durch das Gewicht des Schlägers hervorgerufen wird, dauerhafte Regelabweichungen auftreten können. Dieser Einfluss lässt sich jedoch, wie in Abschnitt 4.2.3 dargestellt, für den statischen Fall näherungsweise beschreiben. Dem Schwerkraft-Einfluss lässt sich somit durch Subtraktion des Kompensationsterm M_{comp} entgegenwirken. Je nach Gelenk, für das der Regler die Drehmomente berechnet, muss hierfür (4.9) bzw. (4.10) verwendet werden.

Mit diesem Vorgehen errreicht man weit bessere Ergebnisse als mit einem I-Anteil, da die dort benötigte Zeit zur Anpassung an eine Abweichung wegfällt. Wie in Abschnitt 5.3 noch gezeigt wird, reicht diese Maßnahme aus, um dauerhafte Regelabweichungen nahezu vollständig zu reduzieren.

4.3.3 Idealisierte Bewegungsmuster

Um dem Roboter seine volle Flexibilität zu geben muss die Bewegung der Gelenke anhand möglichst vieler Parameter planbar sein. Dafür bietet sich eine Abwandlung der Trapezinterpolation an [SK08]. Neben den offensichtlichen, vorgegebenen Größen, wie etwa, dass er nach der zu planenden Bewegung an einer bestimmten Position ist oder dass diese Position nach einer bestimmten Zeitspanne erreicht wird, existieren weitere von aussen vorgegebene Parameter. Diese sind im Folgenden aufgelistet und in Abbildung 4.5 dargestellt.

- v_{start}: Geschwindigkeit des Gelenks zum Start einer Bewegung
- v_{max}: Die maximal mögliche Geschwindigkeit eines Servos unter realistischen Lastbedingungen
- v_h: Mit welcher Geschwindigkeit soll das Gelenk eine bestimmte Position überfahren
- t_h: Wie lange vor Erreichen des Zielpunktes soll die Zielgeschwindigkeit konstant sein

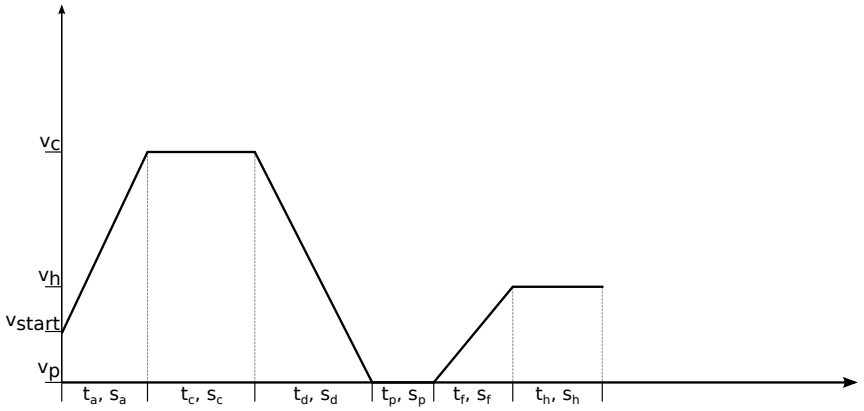


Abbildung 4.5: Aufbau und Größen der Trapez-Interpolation

Interpolations-Gleichungen

Unter der Annahme einer konstanten Beschleunigung a können zwei Gleichungen aufgestellt werden, welche die Beziehungen der in der Abbildung eingezeichneten Größen beschreiben. Zum einen ist dies die Gesamtzeit t_{ges}, die zum Erreichen der Zielbedingung benötigt wird (4.15). Die berechnung der Variablen t_c und t_p kann nicht angegeben werden, da ihre Werte entweder vorgegeben sind oder zu den gesuchten Größen gehören.

$$\begin{aligned} t_a &= \frac{|v_c - v_{\text{start}}|}{a} \\ t_c &= ? \\ t_d &= \frac{|v_p - v_c|}{-a} \\ t_p &= ? \\ t_f &= \frac{|v_h - v_p|}{a} \\ t_h &= const \\ t_{\text{ges}} &= t_a + t_c + t_d + t_p + t_f + t_h \end{aligned} \tag{4.15}$$

Die zweite Gleichung beschreibt den mit dieser Bewegungssequenz insgesamt zurückgelegten Winkel s_{ges} (4.16).

$$\begin{aligned} s_a &= \frac{v_{\text{start}} + v_c}{2} t_a \\ s_c &= v_c t_c \\ s_d &= \frac{v_c + v_p}{2} t_d \\ s_f &= \frac{v_p + v_h}{2} t_f \\ s_h &= v_h t_h \\ s_{\text{ges}} &= s_a + s_c + s_d + s_f + s_h \end{aligned} \tag{4.16}$$

Berechnung mit Zeitbedingung

Durch diese zwei Gleichungen lassen sich demnach zwei unbekannte Größen ermitteln. Da aber die Gleichungen mehr Unbekannte besitzen, müssen diese durch sinnvolle und geeignete Annahmen festgesetzt werden. Die folgende Liste von festgelegten Größen ermöglicht ein Lösen der Gleichungen. Die Annahmen sind so ausgelegt, dass sie in der angegebenen Reihenfolge angewendet werden können um ein optimales Ergebnis zu erhalten. Optimal bedeutet hier, dass die resultierende Bewegungssequenz möglichst schnell möglichst nah an die Zielposition führt. Nur die Beschleunigung auf die Zielgeschwindigkeit v_h kann erst kurz vor dem gewünschten Zeitpunkt zum Erreichen der Zielposition ausgeführt werden. Dieses Vorgehen ist möglichst tolerant gegenüber Schwankungen des Zielzeitpunktes. Sollte während der Ausführung einer Bewegung festgestellt werden, dass die Zielposition früher als geplant erreicht werden muss, kann diese Veränderung so bestmöglich berücksichtigt werden.

1. Gegeben: $v_c = v_{\max}$, $v_p = 0$
   Gesucht: t_c, t_p
2. Gegeben: $v_c = v_{\max}$, $t_p = 0$
   Gesucht: t_c, v_p
3. Gegeben: $v_p = v_h$, $t_p = 0$
   Gesucht: v_c, t_c
4. Gegeben: $t_c = 0$, $v_p = v_h$, $t_p = 0$
   Gesucht: v_c

Vereinfachte Berechnung ohne Zeitbedingung

Wird nicht das Erreichen einer Zielposition nach einer vorgegebenen Zeit benötigt, sondern das schnellstmögliche Erreichen der Zielposition, fällt die Gleichung zu t_{ges} weg. Somit läßt sich nur noch eine Unbekannte berechnen, was aber in diesem Fall ausreichend ist. Dann führen die folgenden zwei Annahmen zum Ziel.

1. Gegeben: $v_c = v_{\max},\ v_h = v_p$
   Gesucht: t_c
2. Gegeben: $t_c = 0,\ v_p = v_h$
   Gesucht: v_c

Spline-Interpolation

Die mittels des vorgestellten Schemas berechneten Bewegungen enthalten abrupte Geschwindigkeitsänderungen. Diese sind für einen Betrachter eindeutig als ruckartiges Verhalten sichtbar und sollten somit vermieden werden. Diese Geschwindigkeitssprünge verursachen auch ein Nachschwingen des Schlägers und beeinträchtigen somit die Performanz des Systems. Schwingen bedeutet eine höhere Motorbelastung und eine schlechtere Kontrolle über das System und sollte demnach vermieden werden.

Eine naheliegende Lösung ist die Verwendung von Splines, um einen kontinuierlichen Übergang zwischen den Geschwindigkeiten zu erhalten. Damit wäre es für jeden Abschnitt des Bewegungsmusters möglich, zwischen gegebenen Start- und Endgeschwindigkeiten einen weichen Übergang zu erhalten. In diesem Fall wurden Bézierkurven dritten Grades [Len04] gewählt. Sie verlangen die Angabe von vier Kontrollpunkten, einem Start- und Endpunkt durch welche die Kurve verläuft sowie zwei Kontrollpunkten, die die Richtung an diesen Punkten definieren.

Die Kurven werden für jedes lineare Segment S separat berechnet, das die Trapez-Interpolation ausgibt. Die Start- und Endpunkte sind die Start- und Endgeschwindigkeiten des jeweiligen Segments. Es wird von einer konstanten Geschwindigkeit an diesen Punkten ausgegangen was durch eine Initialisierung der Kontrollpunkte entsprechend (4.17) erreicht wird.

$$P_0 = P_1 = S_{\text{velocity}_i}, \qquad P_2 = P_3 = S_{\text{velocity}_{i+1}} \tag{4.17}$$

Die Berechnung der kubischen Bézierkurven erfolgt nach (4.18). Der Parameter $t \in [0, 1]$ bestimmt den Punkt, an dem der interpolierte Wert berechnet werden soll. Dieser lässt sich aus den Start- und Endzeiten des aktuellen Segments sowie der aktuellen Zeit t_{cur} bestimmen (4.19).

$$v(t) = \sum_{i=0}^{3} \binom{3}{i} t^i (1-t)^{3-i} P_i \tag{4.18}$$

$$t = \frac{t_{\text{cur}} - S_{\text{time}_i}}{S_{\text{time}_{i+1}} - S_{\text{time}_i}} \tag{4.19}$$

Bei diesem Verfahren entspricht die mittlere Beschleunigung der für das Segment geplanten Beschleunigung während die konkreten Werte über das Intervall in beide Richtungen schwanken. Wenn in der Trapezberechnung bereits die maximal mögliche Beschleunigung angesetzt wurde, überschreiten die interpolierten Werte dieses Maximum. Von daher muss dieser Wert geringer angesetzt werden. Experimentell hat sich ein Faktor im Bereich von $0.85 - 0.95$ als sinnvoll erwiesen.

4.3.4 Schnittstelle

Die Schnittstellen-Komponente bietet den Zugangspunkt zur Ansteuerung der Motoren für alle anderen Teile des Systems. Sie bietet die Wahl zwischen direkter, reaktiver Regler-Bewegung und der Ansteuerung mittels der vorgestellten trapezförmigen Bewegungsmuster. In welcher Situation sich welche Art der Ansteuerung anbietet bzw. zwingend notwendig ist wird genauer in Abschnitt 5.3 anhand von Experimenten untersucht.

Zum Schutz der Hardware wird in dieser Komponente die Bewegung auf den gültigen Arbeitsbereich beschränkt. Dafür wird die aktuelle Position sowie der gewünschte Zielpunkt per Vorwärtskinematik in Weltkoordinaten transformiert. Sollte die Zielposition gültig sein, ist keine weitere Aktion nötig, andernfalls wird die Bewegung aufgesplittet, der Mittelpunkt auf den Arbeitsbereich projiziert und wiederum auf Gültigkeit getestet. Der Vorgang wird entsprechend in der Hälfte wiederholt, die den Bereich verlässt, solang der betrachtete Bewegungsabschnitt eine gewisse Länge übersteigt.

Mess- und Beobachtungsschnittstellen

Es existiert in der Schnittstelle die vor allem zu Entwicklungs- und Testzwecken nützliche Möglichkeit, alle relevanten internen Daten wie etwa Soll- und Ist-Positionen oder Regelgrößen in jedem Zeitschritt in einer Datei abzulegen. Das ermöglicht die nachträgliche Analyse und Optimierung der Ansteuerung.

4.4 Bewegungsplanung

Im Folgenden wird beschrieben, wie die Bewegungssteuerung mit der Ballverfolgung gekoppelt wird. Intelligente Reaktionen im Sinne eines Spiels sind dabei nicht im Fokus, es geht viel mehr darum aus den zugeworfenen Bällen mögliche Handlungsoptionen für den Roboter zu extrahieren.

4.4.1 Zielsetzung

Für die Gestaltung dieser Komponente, die die Ballerkennung mit der Bewegungssteuerung verbindet, lohnt es sich die Problemstellung genauer zu betrachten. Wenn ein Ball angeflogen kommt und vom BallTracker erkannt wird liegt von ihm eine zeitdiskrete Flugbahn vor. Diese Größe ist für den Roboter ein unveränderlicher Eingangswert. Auch die Entscheidung in welche Richtung der Ball nach dem Kontakt mit dem Roboter fliegen soll obliegt nicht dieser Komponente. Diese Entscheidung ist vom angestrebten Spielprinzip abhängig und dort beispielsweise wiederum von der Position der Mit- und Gegenspieler. Somit ist auch die Richtung, bzw. die möglichen Richtungen, in die der Ball gespielt werden soll ein Eingangswert.

Einfluss hat die Bewegungsplanung nur auf die bereits in Abschnitt 3.1 gelisteten Freiheitsgrade: Die Position des Schlägers und seine Geschwindigkeit. Das Ziel der Bewegungsplanung ist es somit, zu einer gegebenen Flugbahn des Balles eine Kombination der Roboterfreiheiten zu finden, die zu einer gegebenen Flugrichtung nach dem Schlägerkontakt führen.

Die naheliegende Lösung besteht darin, aus der Ballflugbahn vor, und der Richtung nach dem Auftreffen zu berechnen, wie der Schläger angesteuert werden muss. Ein großes Problem ist dabei jedoch der Schläger-Ball-Kontakt. Um hier sinnvolle Resultate zu erhalten, kann dieser nicht als zeitloser Stoßvorgang mit Einfalls- gleich Reflexionswinkel angenommen werden (Abbildung 4.6(a)). Stattdessen wird der Ball verformt und gewinnt Rotationsenergie je nach Auftreffwinkel auf den Schläger (Abbildung 4.6(b)). Diese Interaktion ist nicht oder nur mit massivem Aufwand analytisch lösbar, jedoch mit relativ wenig Aufwand simulierbar.

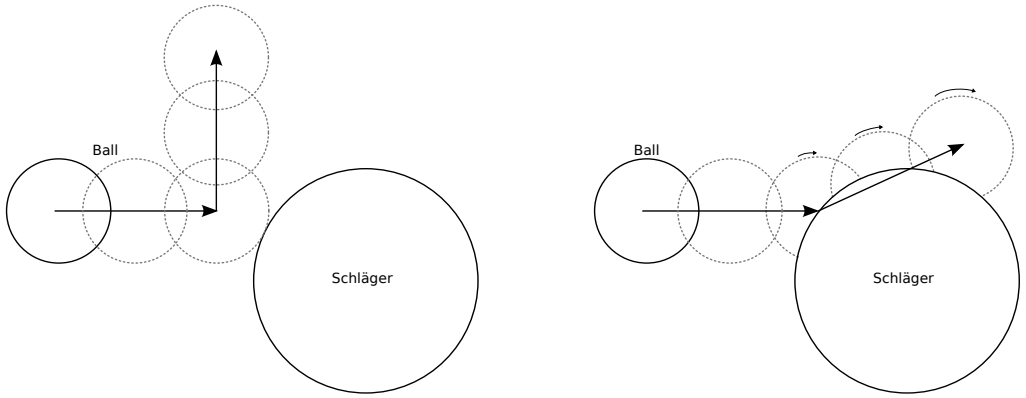
Um trotz der zeitdiskreten Simulation des Schläger-Ball-Kontakts in Echtzeit zu einer Lösung zu gelangen, bietet es sich an, das Ballverhalten vorzuberechnen und nur noch in diesen vorberechneten Ergebnissen zu suchen. Damit ist jegliche Abhängigkeit von der Zeitkomplexität der Simulation aufgehoben. Doch eine direkte, naive Simulation aller möglichen Kombinationen verbietet sich auf Grund der hohen Dimensionalität ebenso wie die Suche in den so generierten Ergebnissen.

Für die Erzeugung der benötigten Simulationsergebnisse müsste für jede Ballposition und -geschwindigkeit (3D + 3D) sowie jede Schlägerposition und -geschwindigkeit (2D + 2D) ein Simulationslauf durchgeführt werden, um die resultierende Ballgeschwindigkeit zu erhalten. Das vollständige Durchlaufen dieser 10-dimensionalen Eingangswerte verbietet sich von der Zeitkomplexität ebenso wie das Abspeichern der Ergebnisse von der Platzkomplexität her.

Unter der Annahme, dass eine vollständige Simulation aller Kombinationen möglich wäre, verbietet sich dennoch ein direktes Nachschalgen innerhalb dieser Ergebnisse. Eine dafür erstellte Lookup-table (LUT) wäre zwar insofern optimal, dass sie für jede Ballgeschwindigkeit vor und

nach dem Roboterkontakt (3D + 3D) sowie jede Ballposition (3D) im Arbeitsraum des Roboters eine ideale Lösung in Form der Schlägerposition und -geschwindigkeit liefern würde, der Platzbedarf wäre jedoch enorm. So würde der Speicherverbrauch einer solchen 9-dimensionalen Datenstruktur den Arbeitsspeicher des zur Verfügung stehenden Steuerrechners bei weitem übersteigen.

Um dennoch ein optimales Roboterverhalten zu einer gegebenen Ballflugbahn vor und nach dem Aufreffen auf den Schläger ermitteln zu können muss sowohl für die Simulation wie auch für die darauf aufsetzende Suche die Dimensionalität des jeweiligen Problems drastisch reduziert werden.



(a) Zeitloser Abprall (b) Realistischerer Abprall

Abbildung 4.6: Vergleich von Modellen zum Aufprall eines Balles auf den Schläger

4.4.2 Physiksimulation zur Referenzwertermittlung

Im Folgenden wird die Physiksimulation beschrieben, die zur realitätsnäheren Ermittlung des Ballverhaltens beim Kontakt des Balls mit dem Schläger des Roboters dient. Es werden die Grundlagen, die Simulation selbst sowie die Größen, die sich daraus für den weiteren Ablauf der Bewegungsplanung ergeben erläutert. Da das Problem einfach genug ist wurde, statt auf eine eine bestehende Programm-Bibliothek zu setzen, der Weg einer Eigenentwicklung gewählt. So bestand mehr Flexibilität in der Auswahl, welche Eigenschaften simuliert werden und an welchen Stellen geeignete Vereinfachungen gemacht werden können. Auch liefern viele Physik-Engines, da sie für den Bereich der Spieleprogrammierung konzipiert wurden, nur realistisch aussehende Resultate für die, zur Steigerung der Berechnungsgeschwindigkeit von der dahinter liegen Physik abstrahiert wurde.

Grundlagen

Die Simulation verwendet ein einfaches, zeitdiskretes Schema. In jedem Zeitschritt wird der Zustand der simulierten Objekte verändert. Dies geschieht durch Integration von Zustandsänderungen, die durch Umgebungseinflüsse oder Interaktion der Objekte untereinander entstehen. Die Objekte sind im konkreten Fall Starrkörper, es handelt sich folglich um eine Starrkörpersimulation [Sto05]. Die verwendeten Zustände und ihre Zustandsänderungen sind in Tabelle 4.1 gelistet.

Bei der linearen Bewegung bzw. der Translation wird die Position und die Geschwindigkeit mittels einer auf den Körper wirkenden Beschleunigung geändert. Dabei sind alle Größen durch Vektoren darstellbar. Gleichung (4.20) und (4.21) geben die Zustandsaktualisierung mittels des Euler-Verfahrens an.

$$\vec{p}_t = \vec{p}_{t-1} + \vec{v}_{t-1}\Delta t + \frac{1}{2}\vec{a}\Delta t^2 \tag{4.20}$$

$$\vec{v}_t = \vec{v}_{t-1} + \vec{a}\Delta t \tag{4.21}$$

Kategorie	Größe	Formelzeichen	Einheit
Zustand	Position	$\vec{p}$	m
	Lineargeschwindigkeit	$\vec{v}$	$\frac{m}{s}$
	Rotation	$\vec{\phi}$ bzw. q_ϕ	rad
	Winkelgeschwindigkeit	$\vec{\omega}$	$\frac{rad}{s}$
Änderung	Linearbeschleunigung	$\vec{a}$	$\frac{m}{s^2}$
	Winkelbeschleunigung	$\vec{\alpha}$	$\frac{rad}{s^2}$

Tabelle 4.1: Zustände und Zustandsänderungen in der Starrkörpersimulation

Bei der Rotation gibt es verschiedene Darstellungsmöglichkeiten. Vektoren bzw. Euler-Rotationen haben den gravierenden Nachteil, dass sie zu Singularitäten führen können[2] und scheiden somit aus. Quaternionen haben gegenüber Matrizen den Vorteil, dass sie kompakter und einfacher zu normalisieren sind. Somit wurden Quaternionen zur Repräsentation von Rotationen gewählt. Winkelgeschwindigkeiten und Winkelbeschleunigungen sind jedoch weiterhin Vektoren. Ihre Richtung gibt die Rotationsachse und die Länge die jeweilige Geschwindigkeit bzw. Beschleunigung an. Gleichungen (4.22) und (4.23) geben die entsprechenden Berechnungen eines neuen Zustandes bei gegebener Winkelbeschleunigung an. Dabei ist q_ϕ das Rotationsquaternion und $q(\vec{v}) = \exp(\frac{\vec{v}}{2})$ eine Funktion, die eine Drehung um $\vec{v}$ mit dem Rotationswinkel $|v|$ als Quaternion darstellt.

$$q_{\phi_t} = q_{\phi_{t-1}}\, q(\vec{\omega}_{t-1}\Delta t + \frac{1}{2}\vec{\alpha}\Delta t^2) \tag{4.22}$$

$$\vec{\omega}_t = \vec{\omega}_{t-1} + \vec{\alpha}\Delta t \tag{4.23}$$

Oft ist es nötig, die Bewegungsgeschwindigkeit eines Körperpunktes $\vec{p}_{ges}$ herauszufinden. Die Geschwindigkeit im Schwerpunkt $\vec{p}$ entspricht dabei der linearen Geschwindigkeit, da dieser Punkt nicht durch Rotation bewegt wird. An jedem Punkt des Körpers, der nicht auf der Rotationsachse liegt, hat die Rotationsgeschwindigkeit einen Anteil an der Bewegung. Wie Linear- und Winkelgeschwindigkeit addiert werden müssen, um die Geschwindigkeit in einem Punkt $\vec{p}_{ges}$ zu erhalten zeigt (4.24).

$$\vec{p}_{ges} = \vec{v} + \vec{\omega} \times (\vec{p}_{ges} - \vec{p}) \tag{4.24}$$

Kräfte auf Körper

Die Körper der Simulation interagieren nur über Kräfte miteinander. Wenn beispielsweise der Ball den Schläger berührt, wird der Ball eingedrückt und erzeugt damit über die Kompression der eingeschlossenen Luft eine Kraft, die ihn wieder in seine Ausgangsform zurückbringen will. Diese Kraft wirkt dann auf den Schläger und den Ball mit gleichem Betrag aber entgegengesetzter Richtung.

Kräfte werden in der beschriebenen Simulation durch Schwerkraft, Reibung und die Verformung des Balls hervorgerufen. Wenn eine Kraft zentral, d.h. in Richtung des Schwerpunkts eines Körpers wirkt, verursacht sie letztendlich eine lineare Beschleunigung. Nichtzentrale Kräfte verursachen ausserdem ein Drehmoment und damit zusätzlich eine Winkelbeschleunigung. Um die Beziehung zwischen Kraft und Beschleunigung herzustellen, muss die Masse m des Körpers einbezogen werden. Da ein schwerer Körper träger ist, verursacht eine Kraft dort weniger Beschleunigung. Den gleichen Effekt hat der Trägheitstensor I für die Beziehung zwischen Drehmoment und Winkelbeschleunigung. (4.25) und (4.26) verdeutlichen die Zusammenhänge für Kräfte $\vec{F}_i$, die jeweils am Punkt $\vec{p}_{w_i}$ wirken.

$$\vec{a} = \frac{\vec{F}}{m} = \frac{1}{m}\sum \vec{F}_i \tag{4.25}$$

$$\vec{\alpha} = I^{-1}\vec{M} = I^{-1}\sum(\vec{p}_{w_i} - \vec{p}) \times \vec{F}_i \tag{4.26}$$

[2] s. z.B. Gimbal Lock

Die Schwerkraft stellt die einfachste Quelle einer Kraft da. Sie wirkt immer zentral auf den Körper und verursache eine Beschleunigung $\vec{g}$. Eine Kraft kann entsprechend (4.27) aus ihr berechnet werden.

$$\vec{F}_g = m\vec{g} \tag{4.27}$$

Reibungskräfte entstehen, wenn zwei Körper mit einer Kraft aneinander gedrückt werden, d.h. die Kraftkomponente in Richtung der Oberflächennormale im Kontaktpunkt nicht null ist. Üblicherweise wird zwischen Haft- und Gleitreibung unterschieden. Erstere liegt vor, wenn keine relative Bewegung im Kontaktpunkt herrscht. Wenn allerdings eine relative Bewegung $\vec{v}_c$ im Berührungspunkt vorliegt spricht man von Gleitreibung. Hier wird dieser Unterschied im Folgenden vernachlässigt und immer von Gleitreibung ausgegangen.

Reibungskräfte sind immer abhängig von der Kraft in Normalenrichtung F_n und dem Reibungskoeffizienten μ. μ leitet sich aus den Materialeigenschaften der beiden beteiligten Oberflächen ab. Die Richtung der entstehenden Kraft $\vec{F}_f$ ist tangential zur Oberfläche im Kontaktpunkt und wirkt der Bewegung in diesem Punkt entgegen. Dieser Zusammenhang ist bildlich in 4.7(a) und formal in (4.28) dargestellt.

$$\vec{F}_f = -\mu|\vec{F}_n|\frac{\vec{v}_c}{|\vec{v}_c|} \tag{4.28}$$

Die letzte der drei Quellen von Kräften innerhalb der beschriebenen Simulation entsteht durch die Kompression, d.h. die Volumenänderung des Balls. Wenn dieser auf den Schläger trifft verformt er sich, wodurch der Druck der Luft im Inneren steigt. Der Druck wirkt auf die gesamte Innenseite des Balls, es wird jedoch davon ausgegangen, dass der Ball so stabil aufgebaut ist, dass er sich nicht insgesamt ausdehnen kann. Somit kann der innere Druck nur auf der eingedrückten Fläche als Kraft wirken.

Um nun die Kompressionskraft zu berechnen muss somit die Volumenänderung des Balls bestimmt werden, um aus ihr die Druckänderung im Inneren zu erhalten. Die Kraft ergibt sich dann aus dem Druck und der eingedrückten, und demnach wirksamen Fläche.

Für die Volumenänderung wird angenommen, dass nur der Ball sich verformen kann und der Schläger vollkommen starr ist. Ball und Schläger überlappen sich demnach entsprechend Abbildung 4.7(b). Das Volumen, das sich beide Körper teilen ist unter diesen vereinfachten Bedingungen die gesuchte Volumenänderung des Balls. Die Berechnung wurde [Wei11] entnommen und ist in (4.29) wiedergegeben.

$$V_{\text{int}} = \frac{\pi(r_s + r_b - d)^2(d^2 + 2dr_b - 3r_b^2 + 2dr_s + 6r_b r_s - 3r_s^2)}{12d} \tag{4.29}$$

$$d = ||\vec{p}_b - \vec{p}_s|| \tag{4.30}$$

Um die entstehende Druckänderung zu berechnen, kann ein ideales Gas in einem isothermen Prozess angenommen werden. In diesem Fall ist die Druckänderung antiproportional zur Volumenänderung (4.31). Da die Berechnung auf den Veränderungen gegenüber den Ausgangswerten basiert, müssen diese bestimmt werden. Der Ausgangsdruck P_{b_0} des Balls läßt sich direkt an den verwendeten Bällen mit Hilfe eines Manometers messen. Für das Ausgangsvolumen V_{b_0} kann die Formel zur Berechnung eines Kugelvolumens verwendet werden (4.32).

$$P_b = \frac{P_{b_0} V_{b_0}}{V_{b_0} - V_{\text{int}}} \tag{4.31}$$

$$V_{b_0} = \frac{4}{3}\pi r_b^3 \tag{4.32}$$

Aus dem Druck lässt sich die daraus resultierende Kraft ermitteln, indem man ihn mit der Fläche multipliziert, auf die er wirkt. Die wirksame Fläche ist in diesem Fall die Schnittebene zwischen Ball und Schläger. Da die entstehende Kraft immer orthogonal zur Fläche wirkt, kann als wirkende Fläche nicht die Oberfläche des Schlägers innerhalb des Balls genommen werden. Hier heben sich die nicht parallel wirkenden Kräfte teilweise auf.

Um die gesuchte Fläche zu erhalten, wird der Radius h der kreisförmigen Schnittfläche zwischen beiden Körpern in (4.33) entsprechend [Wei11] bestimmt. Daraus lässt sich direkt die benötigte Fläche berechnen (4.34).

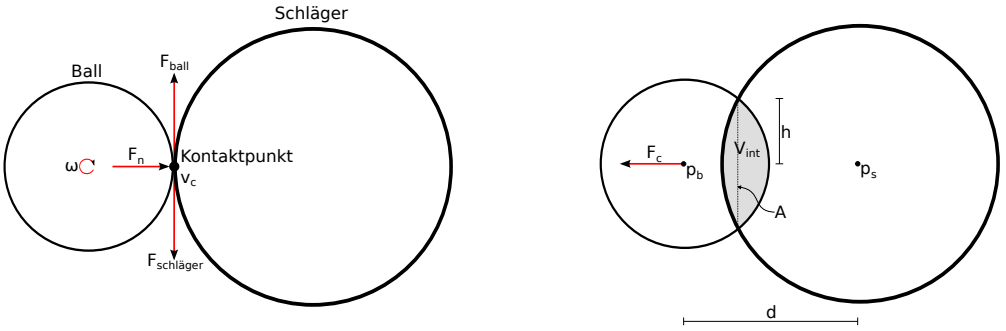
$$h = \frac{1}{2d}\sqrt{4d^2r_s^2 - (d^2 - r_b^2 + r_s^2)^2} \tag{4.33}$$

$$A = \pi h^2 \tag{4.34}$$

Die letztendlich gesuchte Kraft $\vec{F}_c$, die durch Kompression des Balls entsteht ergibt sich aus dem Differenzdruck gegenüber dem Ausgangsdruck auf die soeben berechnete Fläche (4.35). Sie ist orthogonal zur Oberfläche des Balls und zeigt in Richtung seines Mittelpunkts, drückt ihn somit von der Kontaktstelle weg.

$$\vec{F}_c = \frac{\vec{p}_b - \vec{p}_s}{|\vec{p}_b - \vec{p}_s|}(P_b - P_{b_0})A \tag{4.35}$$

Der Verlauf des Betrags dieser Kraft ist Abbildung 4.8 dargestellt. Für die Berechnung wurden die korrekten Parameter für den Ball- und Schlägerdurchmesser verwendet. Der Plot visualisert die resultierende Kraft für eine Eindringtiefe zwischen 0, d.h. keinem Überlappen zwischen Schläger und Ball, und dem 1.5-fachen des Ballradius. Solch eine extreme Überlappung der beiden Körper sollte in der Realität nicht auftreten, da vorher der entstehende Druck die Oberfläche des Balls massiv verformen bzw. zerstören würde. In beiden Fällen gelten die vereinfachten Formeln nicht mehr.



(a) (b)

Abbildung 4.7: Veranschaulichung der (a)Reibungs- und (b)Kompressionskraft

Ablauf der Simulation

Die implementierte Simulation arbeitet zur Vereinfachung der Berechnungen und der physikalischen Zusammenhänge mit einer Reihe von Annahmen. Ein Großteil dieser Vereinfachungen ergibt sich aus der Verwendung einer Starrkörpersimulation. Diese ist, wie der Name bereits andeutet, nicht für flexible, sich mit der Zeit verändernde Körper gedacht. Deshalb kommen zu den bereits genannten Annahmen die Folgenden hinzu.

Sowohl der Schläger wie auch der Ball werden vereinfacht modelliert. Auf den Schläger wirken zwar die Kräfte, die der Ball auf ihn ausübt jedoch keine Kräfte, die von seinem Gelenk her wirken. Unter realistischen Bedingungen wäre nur eine minimale Bewegung in Richtung der starren Schläger-Stange möglich und der fest mit dem Boden verbundene Unterbau würde Kräfte in diese Richtung aufnehmen. Auch würde die feste Verbindung mit dem Gelenk den Schläger auf eine Kreisbahn zwingen. Tangential zum Arbeitsbereich könnten zwar Kräfte auf den Schläger wirken,

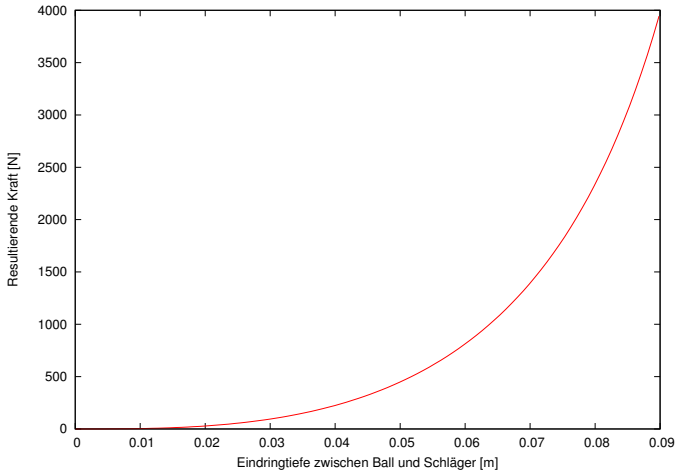


Abbildung 4.8: Kompressionskraft $|\vec{F}_c|$ für eine Überlappung des Balls mit dem Schläger zwischen 0 und $\frac{3}{4}$ des Balldurchmessers

jedoch würde ihnen die rücktreibende Kraft der Stange entgegen stehen, die hier wie eine Feder wirkt. Beide Effekte werden vernachlässigt und die Schläger-Kugel als frei im Raum fliegend angenommen. Durch die vielfach höhere Masse des Schlägers gegenüber dem Ball entsteht durch diese Annahme nur ein geringer und damit tolerierbarer Fehler.

Auch der Ball wird stark vereinfacht modelliert. So hat die Verformung des Balls nur Auswirkungen auf die entstehende Kompressionskraft, wohingegen sich in der Realität auch der Trägheitstensor ändern würde. Auch der Ball-Schläger-Kontakt wurde vereinfacht. Obwohl eindeutig kein Punktkontakt mehr zwischen den beiden Körpern besteht wurde das Rollverhalten des Balls auf dem Schläger mit einem solchen modelliert. Zusätzlich wurde die Haftreibung ignoriert und die Reibungsfläche vereinfacht abgebildet.

Ein einzelner Simulationslauf beginnt immer mit der Initialisierung der Parameter. Es müssen die Eigenschaften des Balls sowie des Schlägers gesetzt werden, etwa ihre lineare und Winkelgeschwindigkeit. Auch Masse und Radius werden benötigt. Daraus lässt sich unter Vereinfachung beider Körper als Hohlkugel nach (3.9) der zugehörige Trägheitstensor bestimmen. Zusätzlich muss für den Schläger der Auftreffpunkt des Balls auf diesem bekannt sein.

Daraufhin kann die zeitdiskrete Berechnung mit fixen Zeitschritten erfolgen. Es werden $\vec{F}_c$, $\vec{F}_f$ und $\vec{F}_g$ berechnet, die an den beschriebenen Punkten auf den Ball-Körper wirken und somit Kräfte und Drehmomente auf diesen ausüben. Die Kräfte $\vec{F}_c$ und $\vec{F}_f$ wirken mit gleichem Betrag aber entgegengesetzter Richtung auch auf den Schläger. Die Gravitationskraft wird hier auf Grund der starren Befestigung ignoriert.

Nach dem Anwenden der Kräfte kann der Zeitschritt abgeschlossen werden, indem der Zustand der beiden Körper anhand der neuen Zustandsänderungen und des vorherigen Zustands aktualisiert wird. Dies kann sowohl zu Positions wie auch zu Geschwindigkeitsänderungen der Körper führen.

Das Ende des Simulationslaufs ist erreicht, wenn Ball und Schläger den Kontakt verlieren. Ab diesem Zeitpunkt hat der Schläger keinen Einfluss mehr auf den Ball, wodurch dieser sich rein Schwerkraft-getrieben und analytisch beschreibbar weiterbewegt. Demnach ist ab diesem Zeitpunkt eine Simulation nicht mehr nötig.

4.4.3 Vereinfachung der Problemstellung

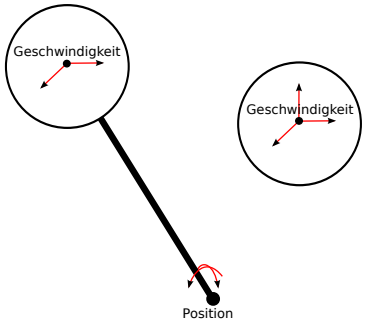
Wie dargelegt ist die Dimensionalität des Suchproblems zu hoch um es auf direktem Wege effizient lösen zu können. Es lassen sich jedoch Vereinfachungen finden, die Eigenheiten, insbesondere Symmetrien des Problems ausnutzen. So muss die Simulation des Roboter-Ball-Kontakts nicht für jede mögliche Pose des Roboters, jede Ein- und Ausgangsgeschwindigkeit des Balles sowie jeden Auftreffpunkt durchgeführt werden. Im Folgenden sind die zwei Ansätze beschrieben, mit denen die benötige Dimensionalität reduziert werden kann.

Zum einen ist es möglich, von der Position des Schlägers zu abstrahieren. Der Schläger kann sich in Hardware, unabhängig von der Position, nur tangential zum Arbeitsbereich bewegen. Wenn die Position vernachlässigt wird, müssen als mögliche Bewegungsrichtungen demnach alle drei Raumdimensionen angenommen werden (Abbildung 4.9(a)).

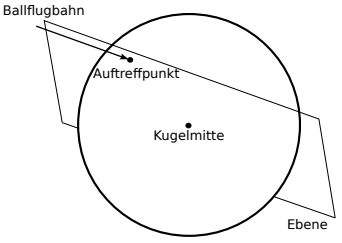
Die andere Verallgemeinerung nutzt die Symmetrie der Schläger-Kugel. Nach aktuellem Stand muss, besonders nach der ersten Vereinfachung, die Ball Ein- und Ausgangsgeschwindigkeit mit 3-Dimensionen und der Auftreffpunkt auf dem Ball mit 2-Dimensionen angenommen werden. Da eine Kugel jedoch vollkommen symmetrisch aufgebaut ist, interessieren nur die Relationen der Vektoren zueinander. Es lässt sich nun eine Ebene konstruieren, die durch den Kugelmittelpunkt verläuft und von der Flugrichtung des Balls sowie dem Vektor Auftreffpunkt-Kugelmitte aufgespannt wird. Diese Ebene bildet zusammen mit ihrer Normalen die Achsen des Koordinatensystems, wobei eine Achse der Ebene so orientiert ist, dass sie parallel zur Anflugrichtung des Balls ist.

Folglich ist die Anflugrichtung nur noch Eindimensional. Der Auftreffpunkt befindet sich auf der Ebene und auf der Kugel und ist somit ein eindimensionaler Kreisbogen. Abbildung 4.9(c) veranschaulicht die Kombination der bisher beschriebenen Vereinfachungen.

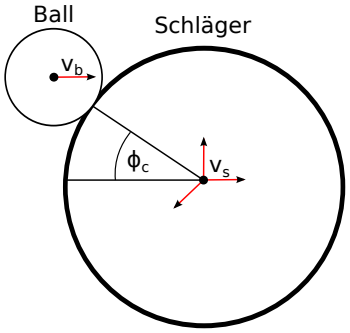
Bei einem zeitlosen Abprall des Balls wäre die Reflexionsrichtung auf der Ebene und folglich nur zweidimensional. Da jedoch ein realistisches Abprallverhalten simuliert wird, wirkt für die Zeitspanne der Berührung zwischen Ball und Schläger eine Reibungskraft. Diese ist dafür verantwortlich, dass der Schläger den Ball auch orthogonal zur aufgespannten Ebene beschleunigen kann. Somit kann eine Reflexion des Balls in alle drei Raumrichtungen erfolgen.



(a) Abstraktion von der Roboterposition



(b) Konstruktion des neuen Koordinatensystems



(c) Kombination beider Vereinfachungen

Abbildung 4.9: Ansätze zur Dimensionsreduktion

Verwendung innerhalb der Physiksimulation

Die Ergebnisse der Physiksimulation, die benötigt werden, um in ihnen Spielposen und -geschwindigkeiten zu gegebenem Ballverhalten nachzuschlagen werden mit den genannten Vereinfachungen simuliert. Die in jedem Simulationslauf generierten Größen werden zusammen mit

den jeweils variierenden Eingangsparametern gespeichert, um in ihnen, wie im weiteren Verlauf beschrieben, suchen zu können. Durch die Dimensionsreduktion wird dabei eine massive Effizienzsteigerung erreicht.

Da von der echten Schlägerposition abstrahiert wurde, muss darauf geachtet werden, die Simulationen ohne Einfluß der Schwerkraft durchzuführen. Diese würde zu falschen Ergebnissen führen, da ihre Richtung, nach rückgängig machen der Vereinfachungen, noch nicht bekannt ist. Aufgrund der Kürze des Ball-Schläger-Kontakts ist der dadurch entstehende Fehler vertretbar. Die Eingangswerte für die Simulationsläufe sind, zusammen mit ihren Besonderheiten, im Folgenden gelistet.

- **Ballgeschwindigkeit** $\vec{v}_{b_{in}}$ (1D): $v_{b_{in}} > 0$, $(v_{b_{in}}, 0, 0)^T$
- **Auftreffwinkel** ϕ_c (1D): $\phi_c \in [-\frac{3}{4}\pi, \frac{3}{4}\pi]$
- **Schlägergeschwindigkeit** $\vec{v}_s$ (3D)

Der Parameter $v_{b_{in}}$ besteht entsprechend den Vereinfachungen aus einer einzelnen Größe, die die Ballgeschwindigkeit entlang der x-Achse angibt. Die folgenden Größen werden pro Simulationslauf generiert.

- **Reflexionsgeschwindigkeit** $\vec{v}_{b_{out}}$ (3D)
- **Ballversatz**: Differenz der Ballposition zwischen erstem und letztem Schlägerkontakt
- **Bewertung** r_{sim}: $r_{sim} \in [0, 1]$

Die Bewertung ist ein Maß für die Qualität des Simulationslaufs. Sie wird bisher nicht verwendet, könnte jedoch angeben, wie stark die Ausgangsparameter auf leichte Veränderungen der Eingangswerte reagieren. Auch die Stabilität der Werte auf verschiedene Rotationsgeschwindigkeiten des Balls ließen sich dort kodieren.

4.4.4 Lookup-Tables zur Suche im simulierten Verhalten

Um innerhalb der, mittels Simulation erstellten, Referenzwerte zu suchen, müssen diese nach gegebenen Parametern gefiltern werden. Direkt vorgegebene Größen sind die Ball- und Reflexionsgeschwindigkeiten. Doch auch die Schlägergeschwindigkeit, eigentlich eine gesuchte Größe, muss gefiltert werden. Durch Abstraktion von der Schlägerposition wurde die Schlägergeschwindigkeit von nur zwei auf drei Dimensionen vergrößert.

Demnach reicht es nicht aus, alle Simulationsergebnisse zu den beiden vorgegebenen Geschwindigkeiten zu finden. Dabei würden viele Einträge herauskommen, die eine Schlägerbewegung beschreiben, die am Schnittpunkt von Ballflugbahn und Arbeitsbereich des Roboters unmöglich wäre. Durch die Bauweise des Systems ist immer nur eine zweidimensionale Bewegung tangential zum Arbeitsbereich möglich. Es ist somit nötig die Schlägergeschwindigkeiten danach zu filtern, ob sie in oder nah an der erlaubten Ebene liegen.

Da eine einzelne Tabelle zum Nachschlagen der gesuchten Einträge auch mit den beschriebenen Vereinfachungen noch zu viele Dimensionen hätte, wurde ein zweistufiger Ansatz gewählt. Mit diesem wird in einem ersten Schritt nach Ball- und Reflexionsgeschwindigkeit gefiltert und anschließend in den Resultaten nach der Ausrichtung der Ballgeschwindigkeit gesucht. Beide Filterstufen sind in den folgenden Abschnitten beschrieben.

Der vollständige Ablauf der Bewegungsplanung ist in Abbildung 4.10 schematisch dargestellt. Die einzelnen Stufen, in denen das Filtern und Auswählen der Simulationsergebnisse stattfindet, sind in den jeweils angegebenen Abschnitten beschrieben.

4.4.5 Lookup-Table für Geschwindigkeiten

Der erste Filter zum Eingrenzen der relevanten Ergebnisse der Physiksimulation arbeitet auf den vorgegebenen Ball- und Reflexionsgeschwindigkeiten. Das Eingrenzen soll mit Hilfe einer Lookup-Table durchgeführt werden. Die in diesem Fall gewählte Ausprägung der Datenstruktur hat ebenso viele Dimensionen wie die gegebenen Größen zusammengenommen. Jede Dimension wird äquidistant in einzelne Elemente diskretisiert, sodass jeder reelle Wert der zugehörigen Eingangsgröße exakt auf ein Element abgebildet werden kann. Die Größe der Datenstruktur steigt mit der Anzahl

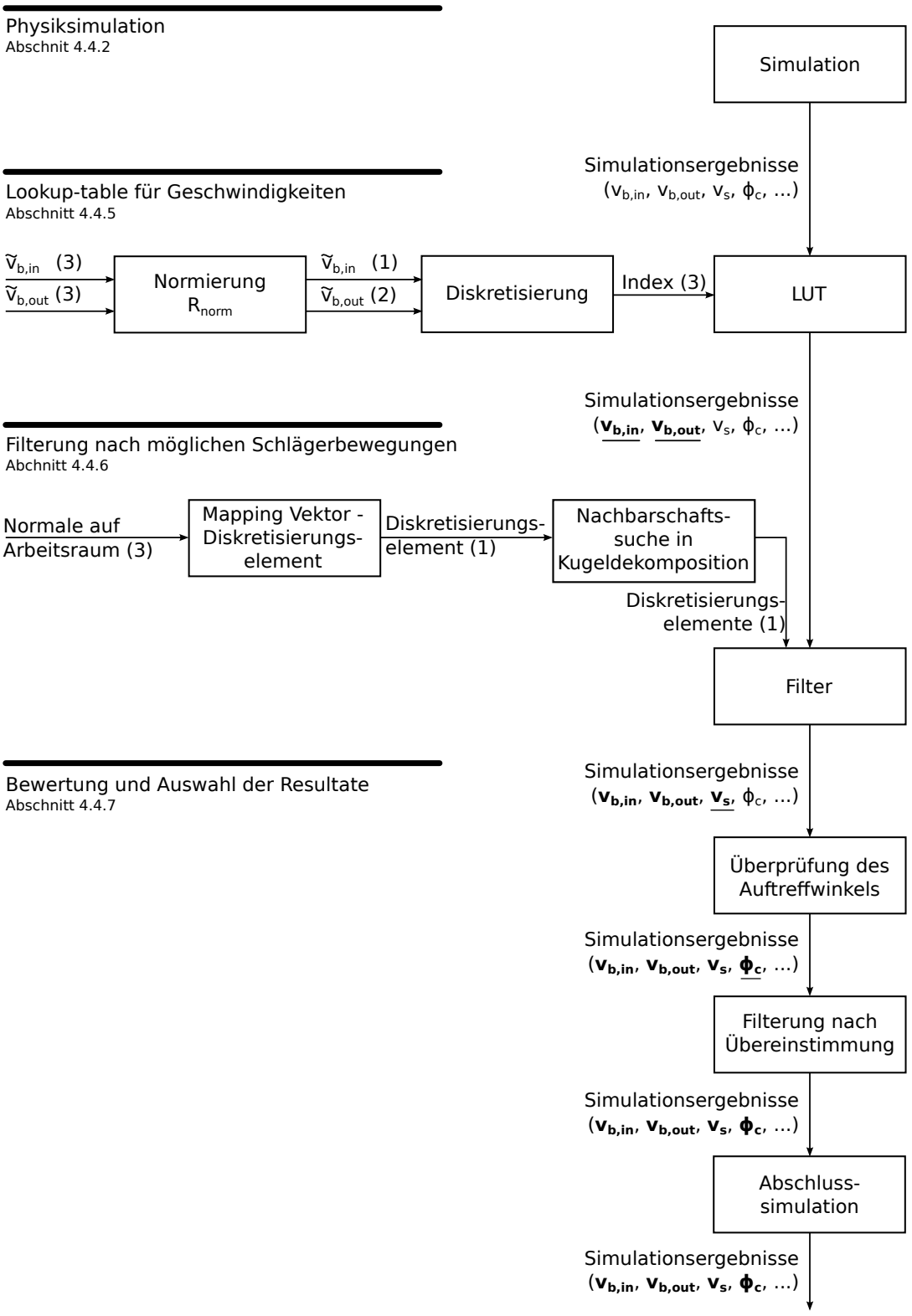


Abbildung 4.10: Schematischer Ablauf der Bewegungplanung mit den einzelnen Stufen zur Filterung und Auswahl der Simulationsergebnisse. In jedem Bereich, der im angegebenen Abschnitt beschrieben ist, werden einzelne Größen der Ergebnisse mit den Soll-Werten abgeglichen (unterschrichen). Die Dimensionen der ausgetauschten Daten stehen in Klammern. Zuerst werden die Simulationsergebnisse entsprechend der geforderten ein- und ausgehenden Ballgeschwindigkeiten ($\tilde{\vec{v}}_{b_{in}}$, $\tilde{\vec{v}}_{b_{out}}$) gefiltern, anschließend anhand der Schlägergeschwindigkeit $\vec{v}_s$. In der letzten Stufe wird der Auftreffwinkel ϕ_c zwischen Ball und Schläger überprüft und die verbleibenden Ergebnisse gewertet sowie sortiert.

der Dimensionen und der Elemente pro Dimension. Da eine Verringerung der Elemente jedoch mehr Werte auf ein Element abbilden würde, würde dies eine gröbere Filterung verursachen. Eine Reduktion der Dimensionen ist, wenn möglich, der bessere Weg zum Verkleinern der Struktur.

Die Lookup-Table wäre, mit den in der Simulation verwendeten Vereinfachungen, noch immer vierdimensional. Da jedoch, im Unterschied zur Simulation, die Reflexionsgeschwindigkeit bereits bekannt ist, sind weitere Vereinfachungen möglich. Ein Einsparungspotential, und damit eine effizientere Normalisierung, ergibt sich daraus, dass sich zwei bekannte Vektoren im selben Raum immer in zwei Dimensionen beschreiben lassen. Das Koordinatensystem, in dem sich die beiden Vektoren, Ball- und Refexionsgeschwindigkeit, befinden kann so ausgerichtet werden, dass zum einen beide Vektoren in einer Ebene parallel zu einer Achse liegen. Zum anderen kann einer der Vektor parallel zur zweiten Achse ausgerichtet werden, woraufhin sich beide Vektoren durch insgesamt drei Größen beschreiben lassen.

Hilfsfunktion für Vektor-zu-Vektor-Rotationen

Zum Konstruieren der benötigten Koordinatentranformation wird eine Funktion benötigt, welche die Rotationsmatrix erstellt, um einen Vektor $\vec{v}_{\text{from}}$ parallel zu einem beliebigen anderen Vektor $\vec{v}_{\text{to}}$ zu drehen. In (4.36) bis (4.40) sind die Schritte angeben, um zuerst die Rotationsachse und den zugehörigen Rotationswinkel zu bestimmen. Anschließend wird mittels der Rodrigues-Formel [Bel11] die Rotationsmatrix konstruiert. Sollten die beiden Vektoren bereits parallel sein ist keine Rotation nötig und die Funktion liefert die Einheitsmatrix I.

$$\vec{a}' = \vec{v}_{\text{from}} \times \vec{v}_{\text{to}} \tag{4.36}$$

$$\vec{a} = \frac{\vec{a}'}{|\vec{a}'|} \tag{4.37}$$

$$\theta = \arccos(\vec{v}_{\text{from}} \cdot \vec{v}_{\text{to}}) \tag{4.38}$$

$$M = \begin{pmatrix} \cos\theta + a_x^2(1-\cos\theta) & a_x a_y(1-\cos\theta) - a_z\sin\theta & a_y\sin\theta + a_x a_z(1-\cos\theta) \\ a_z\sin\theta + a_x a_y(1-\cos\theta) & \cos\theta + a_y^2(1-\cos\theta) & -a_x\sin\theta + a_y a_z(1-\cos\theta) \\ -a_y\sin\theta + a_x a_z(1-\cos\theta) & a_x\sin\theta + a_y a_z(1-\cos\theta) & \cos\theta + a_z^2(1-\cos\theta) \end{pmatrix} \tag{4.39}$$

$$Rot(\vec{v}_{\text{from}}, \vec{v}_{\text{to}}) = \begin{cases} I & \text{für } \vec{a}' = 0 \\ M & \text{sonst} \end{cases} \tag{4.40}$$

Konstruktion der Koordindatentransformation

Um die gesuchte Transformationsmatrix zu konstruieren, wird zuerst die Normale der Ebene der beiden Vektoren zusammen mit dem Koordinatenursprung bestimmt (4.41). Die erste Rotation ergibt sich, indem man den Normalenvektor parallel zur y-Achse rotiert (4.42). Anschließend wird die Eingangsgeschwindigkeit parallel zur x-Achse gedreht (4.43) und beide Matrizen verknüpft.

$$\begin{aligned} \vec{n}'_{\text{plane}} &= \vec{v}_{b_{in}} \times \vec{v}_{b_{out}} \\ \vec{n}_{\text{plane}} &= \frac{\vec{n}'_{\text{plane}}}{|\vec{n}'_{\text{plane}}|} \end{aligned} \tag{4.41}$$

$$R_y = Rot(\vec{n}_{\text{plane}}, (0,1,0)^T) \tag{4.42}$$

$$R_x = Rot(R_y\vec{v}_{b_{in}}, (1,0,0)^T) \tag{4.43}$$

$$R_{yx} = R_x R_y \tag{4.44}$$

Um den Wertebereich der Vektoren weiter einzuschränken wird, wenn die z-Komponente von $\vec{v}_{b_{out}}$ negativ ist, noch eine Rotation um die x-Achse von 180° angefügt. Diese kann vereinfacht durch negieren der y- und z-Komponente einer Einheitsmatrix erfolgen (4.45). Die finale Transformationsmatrix ergibt sich demnach entsprechend (4.46). Die Umkehrung der Transformation kann, da es sich um reine Rotationen ohne Translation handelt, durch Transponieren der Matrix erfolgen.

$$R_z = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \tag{4.45}$$

$$R_{\text{norm}} = \begin{cases} R_z R_{yx} & \text{für} \quad (R_{yx}\vec{v}_{b_{out}})_z < 0 \\ R_{yx} & \text{sonst} \end{cases} \tag{4.46}$$

Aufbau der Datenstruktur

Die Datenstruktur der ersten LUT muss die eindimensionale Ballgeschwindigkeit und die zweidimensionale Reflexionsgeschwindigkeit aufnehmen. Sie ist demnach ein 3D-Gitter, das für jede Kombination der Eingangswerte die Simulationsergebnisse enthält, die zu diesen passen.

Um auf eine Zelle des Gitters zugreifen zu können muss ihr Index aus den beiden Vektoren berechnet werden. Dafür müssen diese zuerst mit R_{norm} multipliziert werden (4.47)-(4.48). Die verbleibenden Komponenten der Vektoren, d.h. solche, die $\neq 0$ sind, werden zum reellwertigen Index zusammengefügt (4.49). Unter Zuhilfenahme der gegebenen Minimal- und Maximalwerte in jeder Dimension sowie der gewünschten Anzahl an Unterteilungen kann die Schrittgröße berechnet werden (4.50). Daraus kann letztendlich der gesuchte ganzzahlige Index bestimmt werden (4.51), der zum Zugriff auf die Datenstruktur benötigt wird.

$$\vec{b} = \vec{v}_{b_{in},N} = R_{\text{norm}}\vec{v}_{b_{in}} \tag{4.47}$$

$$\vec{c} = \vec{v}_{b_{out},N} = R_{\text{norm}}\vec{v}_{b_{out}} \tag{4.48}$$

$$\vec{i}_{\text{real}} = (b_x, c_x, c_z)^T \tag{4.49}$$

$$\vec{stepssize} = \frac{\vec{max} - \vec{min}}{\vec{steps} - (1,1,1)^T} \tag{4.50}$$

$$\vec{i} = \lfloor \frac{(\vec{i}_{\text{real}} - \vec{min})}{\vec{stepsize}} + (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})^T \rfloor \tag{4.51}$$

Das Befüllen der Lookup-table geschieht demnach nach folgendem Schema. Für jedes Simulationsergebnis werden die Ball- und Reflexionsgeschwindigkeiten transformiert und diskretisiert Die Daten werden anschließend in die so erhaltene Gitterzelle eingefügt.

Die Abfrage geschieht ähnlich. Auch hier werden zuerst die angefragten Ball- und Reflexionsgeschwindigkeiten transformiert und normalisiert. Anschließend werden alle in der so addressierten Gitterzelle hinterlegten Simulationsergebnisse zugänglich gemacht.

4.4.6 Filterung nach möglichen Schlägerbewegungen

Der Schläger des Roboters kann sich immer nur tangential zur Arbeitsfläche bewegen. Wie diese mögliche Bewegungsebene im Raum orientiert ist, hängt von seiner Position ab. Da die vorherige Filterstufe die Simulationsergebnisse nur nach Eingangs- und Reflexiongeschhwindigkeit auswählt, finden sich so viele Einträge, in denen die Bewegungsebene stark von der hardwaremäßig Vorgegebenen abweicht. Es muss demnach eine weitere Auswahl anhand dieses Kriteriums getroffen werden.

Die Simulationsergebnisse enthalten jeweils nur den Schlägergeschwindigkeitsvektor. Um alle passenden Einträge anhand dieser Geschwindigkeitsvektoren zusammenfassen zu können bietet sich der Normalenvektor auf dem Arbeitsraum an. Wie in Abbildung 4.11 zu sehen, müssen die Geschwindigkeitsvektoren immer einen 90°-Winkel zur Normalen zuzüglich einer gewissen Toleranz haben.

Natürlich könnten die Ergebnisse der ersten Filterstufe linear durchsucht werden, um alle passenden Einträge herauszufiltern. Da dieser Vorgang jedoch zu lange dauert, um den Echtzeitanforderungen des Systems gerecht zu werden, muss auch hier eine effizientere Lösung gefunden werden. Die Wahl fiel auf eine weitere, zweite LUT-Stufe.

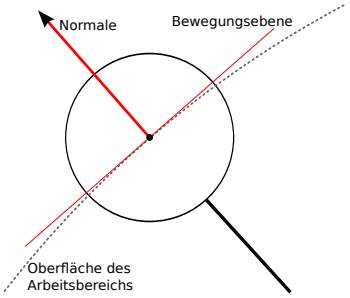


Abbildung 4.11: Bewegungsmöglichkeiten in Abhängigkeit von der Schlägerposition

Anforderungen und Lösungsansatz

Die benötigte LUT hat eine einfach zu beschreibende Aufgabe. Sie muss für eine Menge an Simulationsergebnissen diejenigen zurückliefern, deren Schlägergeschwindigkeit ungefähr orthogonal zu einem gegebenen Vektor ist.

Natürlich ließe sich hier ähnlich wie in der ersten Filterstufe ein 3D-Gitter verwenden, auf das per Normalenvektor zugegriffen wird. Da diese Struktur jedoch für jede Zelle aus der ersten Stufe einmal vorhanden sein müsste, sollte hier sehr sparsam mit Speicherplatz umgegangen werden. Dafür lohnt es sich auszunutzen, dass der Normalenvektor auf dem Arbeitsbereich ohne Informationsverlust ein Einheitsvektor sein kann. Dieser lässt sich mit nur noch zwei Größen beschreiben. Demnach wird zum Erstellen einer Lookup-table eine sinnvolle Diskretisierung dieser zwei Größen benötigt.

Da alle Normaleneinheitsvektoren auf der Oberfläche einer Einheitskugel enden, liegt die Möglichkeit nah, diese Kugeloberfläche zu diskretisieren. Gängige Ansätze zur Beschreibung einer Position auf einer Kugel mittels zweier Winkel sind dabei ungeeignet. Sie haben den gravierenden Nachteil, dass sie keine äquidistante Unterteilung liefern. Es gibt immer zwei gegenüberliegende Singularitäten, an denen eine große Winkeländerung eine kleine bis gar keine Auswirkung auf die beschriebene Position hat.

Im Folgenden wird jedoch beschrieben, wie eine solche, nahezu äquidistante Zerlegung einer Kugel aussehen kann. Diese Zerlegung liefert einzelne Elemente auf der Kugeloberfläche, die über Nachbarschaftsbeziehungen miteinander verknüpft sind. Jedes dieser Elemente hat eine eindeutige Identifikationsnummer, die genutzt werden kann, um ihr die zur Normalen gehörenden Schlägergeschwindigkeitsvektoren zuzuordnen.

Bisher fehlt jedoch noch die Zuordnung zwischen dem gegebenen Einheitsvektor und einem Element aus der Kugeldiskretisierung. Da kein einfacher, mathematisch beschreibbarer Zusammenhang zwischen beiden Größen besteht, wird hier auf eine weitere LUT zurückgegriffen. Dieses 3D-Gitter kann mit dem Normaleneinheitsvektor angefragt werden und liefert das zugehörige Element aus der Kugelzerlegung.

Diese beiden Datenstrukturen haben keinen signifikanten Einfluß auf den Speicherverbrauch, da sie nur einmalig im gesamten System vorliegen müssen. Das ist möglich, da beide vollkommen unabhängig von den Ballgeschwindigkeiten oder sonstigen Parametern aus der Simulation sind.

Zur Veranschaulichung der komplexen Folge von Abfragen sind die einzelnen Schritte in Abbildung 4.12 schematisch angegeben. Eine detailierte Beschreibung, wie die Umsetzung im konkreten aussieht, folgt.

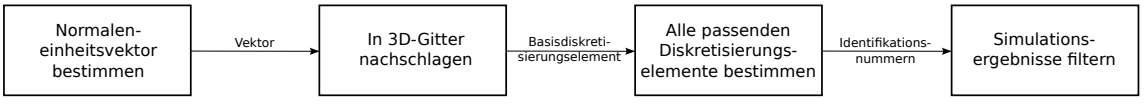


Abbildung 4.12: Schematischer Ablauf der Schlägergeschwindigkeits-Filterung

Konstruktion

Die äquidistante Diskretisierung einer Kugel basiert auf der rekursiven Zerlegung eines Ikosaeders. Ein Ikosaeder ist ein Körper, der aus 20 Dreiecken gleicher Größe besteht (s. Abbildung 4.14(a)). Die Konstruktion des benötigten Basis-Ikosaeders mit der Kantenlänge 2 kann mittels der in (4.52) angegebenen Eckpunkte geschehen, die [Wik11] entnommen sind.

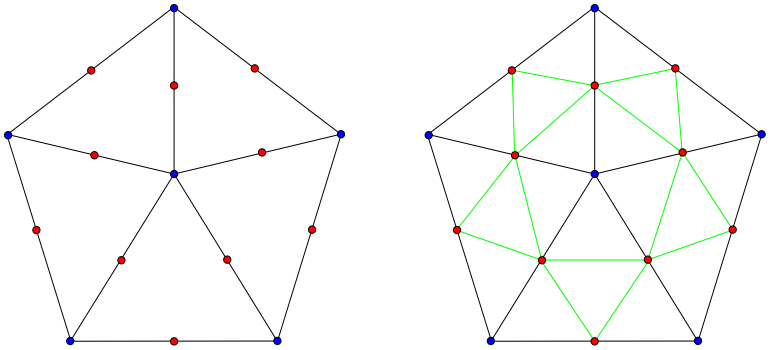
$$\begin{array}{llll} k_0 = (0, 1, \varphi) & k_1 = (0, 1, -\varphi) & k_2 = (0, -1, \varphi) & k_3 = (0, -1, -\varphi) \\ k_4 = (1, \varphi, 0) & k_5 = (1, -\varphi, 0) & k_6 = (-1, \varphi, 0) & k_7 = (-1, -\varphi, 0) \\ k_8 = (\varphi, 0, 1) & k_9 = (-\varphi, 0, 1) & k_{10} = (\varphi, 0, -1) & k_{11} = (-\varphi, 0, -1) \end{array} \tag{4.52}$$

$$\varphi = \frac{1}{2}(1 + \sqrt{5}) \tag{4.53}$$

Die initiale Verknüpfung der Eckpunkte (Knoten) mittels Nachbarschaften kann über die bekannte Kantenlänge erfolgen. Alle Knoten mit einem Abstand von 2 fügen sich gegenseitig zu ihren Nachbarn hinzu. Anschließend werden die Knotenpositionen auf Länge eins normiert, d.h. die Knoten auf eine Einheitskugel projiziert.

Der rekursive Algorithmus zum Verfeinern der Gitterstruktur wurde aus [HR95] abgeleitet. Der Ablauf eines Schrittes ist in Abbildung 4.13 veranschaulicht. Zuerst werden entsprechend Abbildung 4.13(a) zwischen allen alten Knoten (blau) neue Knoten eingefügt (rot). Diese liegen von ihrer Position her exakt zwischen den alten Knoten und müssen folglich auf die Oberfläche der Einheitskugel projiziert werden. Die Nachbarschaften der alten Knoten werden gelöst und auf die neu eingefügten Knoten aktualisiert. Die neuen Knoten bekommen entsprechend die Knoten, zwischen denen sie eingefügt wurden als Nachbarn.

Anschließend können in einem zweiten Schritt, der in Abbildung 4.13(b) dargestellt ist, die Nachbarn der neuen Knoten untereinander aktualisiert werden. Hierfür werden die vier nächstgelegenen neuen Knoten zu den eigenen Nachbarn hinzugefügt.

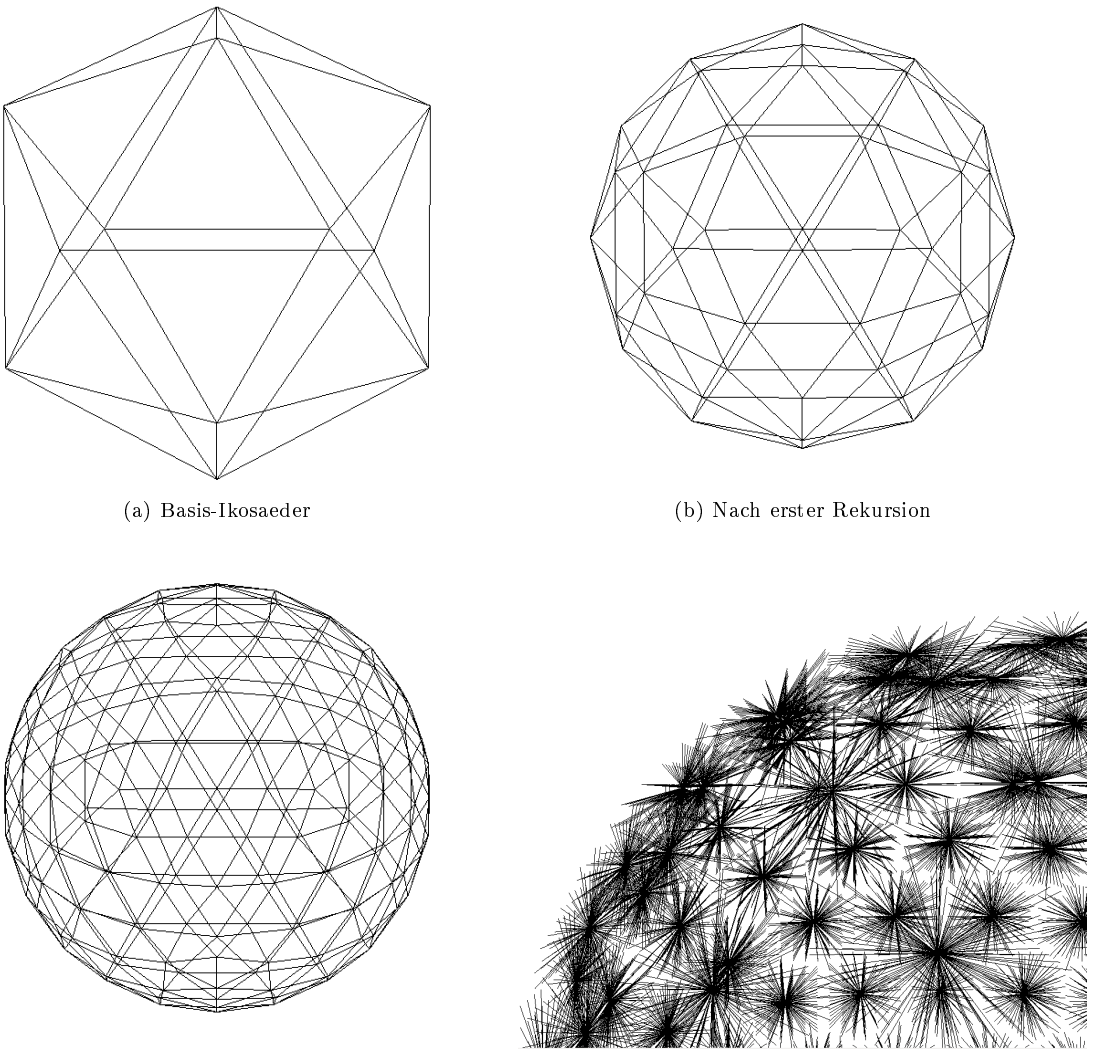


(a) Neue Knoten einfügen (b) Knoten verknüpfen

Abbildung 4.13: Ablauf der rekursiven Zerlegung

Das Basis-Ikosaeder gefolgt von zwei Rekursionen zur Verfeinerung der Gitterstruktur ist in Abbildung 4.14(a) – (c) zu sehen. Die in (c) dargestellte Zerlegung nach zwei Iterationen mit insgesamt 126 Knoten wurde im weiteren Verlauf verwendet, da der Abstand der Knoten zueinander sich experimentell als gut geeignet erwies. Die Zerlegung ist nicht exakt äquidistant, da um die Ursprungsknoten des Ikosaeders herum immer 5-Ecke bestehen bleiben, wohingegen alle neu hinzugefügten Strukturen 6-Ecke sind. Da jedoch die Ausdehnung der 5- wie auch der 6-Ecke nahezu identisch ist, kann dieser Umstand toleriert werden.

Zur Zuordnung zwischen einem gegebenen Einheitsvektor und dem zugehörigen Knoten wird zusätzlich noch das angesprochene 3D-Gitter benötigt. Es ist in jeder Dimension in 50 Einheiten unterteilt. Jede dieser Einheiten ist, wenn sie sich in der Nähe der Kugeloberfläche befindet, dem nächstgelegenen Knoten zugeordnet. Die Zuordnung ist bildlich in 4.14(d) dargestellt. Der Anfang der Linien befindet sich immer in einer Zelle des Gitters, das Ende an der Position des zugeordneten, nächstgelegenen Knoten.



(a) Basis-Ikosaeder

(b) Nach erster Rekursion

(c) Nach zweiter Rekursion

(d) Linien zwischen Gitterzellen und zugehörigem Knoten

Abbildung 4.14: (a) – (c) Rekursive Zerlegung eines Ikosaeders und (d) Zuordnung zwischen 3D-Gitter und Kugel-Diskretisierungs-Elementen

Zugriff

Der Zugriff auf die Datenstruktur erfolgt mittels eines gegebenen Einheitsvektors. Zuerst wird dieser im 3D-Gitter nachgeschlagen. Der Zugriff über Diskretisierung der einzelnen Dimensionen erfolgt analog zur ersten LUT aus Abschnitt 4.4.5. Diese Abfrage liefert den entsprechenden Knoten aus der Ikosaeder-Dekomposition. Da eine Abfrage letztendlich jedoch nicht darauf zielt, exakt einen Knoten zurückzuliefern sondern alle Knoten, die in einen gewissen Toleranzbereich um den gegebenen Einheitsvektor hinein reichen, müssen diese weiteren Knoten bestimmt werden.

Die Ausdehnung eines Knotens bestimmt sich aus dem maximalen Abstand zu einem Nachbarknoten. Dadurch wird der Abstand überschätzt, es garantiert allerdings, dass sich die Einzugsbereiche der Knoten ausreichend und überall überlappen. Da die Toleranz nicht als Entfernung, sondern als Winkel gegeben wird, muss auch der Knotenabstand als Winkel vorliegen. Dafür wird aus der Entfernung d, die einer Kreissehne entspricht, der entsprechende Winkel bestimmt (4.54).

$$\alpha_r = 2 \arcsin \frac{d}{2} \tag{4.54}$$

Um nun vom Basis-Knoten alle innerhalb der Toleranz liegenden Knoten zu finden, werden

rekursiv alle Nachbarknoten untersucht und als relevant markiert, solange diese sich innerhalb des Toleranzbereichs befinden. Dieses Kriterium bestimmt sich aus dem Winkel zwischen Einheitsvektor und untersuchtem Knoten α_d, der Ausdehnung des untersuchten Knotens α_r sowie der Toleranz α_t (4.55).

$$\alpha_d \leq \alpha_r + \alpha_t \tag{4.55}$$

Als Ergebnis dieser Operation erhält man alle relevanten Knoten und damit die zugehörigen Knoten-Identifikationsnummern. Diese Nummern werden für das Einsortieren und Filtern der Simulationsergebnisse benötigt.

Verwendung

Da diese Stufe der ersten LUT nachgeschaltet ist, operiert sie nur auf den Simulationsergebnissen, die der entsprechenden Zelle mit passender Ball- und Reflexionsgeschwindigkeit zugeordnet sind.

Für das Einfügen wird herausgefunden, welchen Knotennummern einem Simulationsergebnis zugeordnet werden müssen. Dafür werden die Nummer aller Knoten angefragt, die auf der Ebene orthogonal zur Schlägergeschwindigkeit liegen. Die Toleranz kann hierfür mit 0° angesetzt werden. Mit der Zuordnung zu diesen Knotennummern erreicht man, dass bei einer Anfrage mit einem Normalenvektor auf dem Arbeitsbereich alle Simulationsergebnisse gefunden werden, deren Schlägergeschwindigkeit orthogonal zur angefragten Normalen sind. Also genau die Bewegungen, die in dieser Schlägerposition von der Hardware her möglich sind.

Eine Besonderheit sind Schlägergeschwindigkeiten mit einem Betrag nahe null. Es wäre fehlerhaft, diese nur einem Teil der Simulationsergebnisse zuzuordnen. Vielmehr ist die Angabe, dass keine Schlägergeschwindigkeit nötig ist, vollkommen positionsunabhängig. Deshalb müssen solche Einträge unabhängig von der Zuordnung zu einer regulären Knotennummer gesammelt und behandelt werden. Dies geschieht im entwickelten System, indem sie einer speziellen, unbenutzten Nummer zugewiesen werden.

Nach dem Einsortieren kann die Lookup-Table verwendet werden. Ein Filtern der Simulationsergebnisse beginnt mit dem Beziehen der Knotennummern bei gegebener Toleranz. Hierzu wird die Sondernummer für keine Schlägerbewegung hinzugefügt. Anschließend müssen alle Simulationsergebnisse ausgewählt werden, die einer der relevanten Knotennummern zugeordnet sind. Da dabei leicht Simulationsergebnis-Dubletten auftreten können, muss das Ergebnis um diese bereinigt werden.

Mit Anwendung dieser beiden bisher beschriebenen Filterstufen wurde erreicht, dass die Simulationsergebnisse ausgewählt wurden, deren Ball- und Reflexionsgeschwindigkeiten passend zur Anfrage sind und deren Schlägergeschwindigkeiten sich innerhalb der erlaubten Ebene tangential zum Arbeitsraum bewegen.

4.4.7 Bewertung und Auswahl der Resultate

Auf den Ergebnissen, die nach der bisher durchgeführten Eingrenzung übrig bleiben, werden weitere Filteroperationen durchgeführt. Das Ziel ist es, nur noch wenige, möglichst optimal passende Vorschläge zu bieten, an welcher Stelle der Schläger den Ball mit welcher Geschwindigkeit treffen soll. Da die Menge der Kandidaten immer weiter zusammenschrumpft, sind immer aufwendigere Tests durchführbar.

Überprüfung der Auftreffpunkts

Der folgende Test definiert ein hartes Ausschlusskrierium, d.h. es muss zwingend erfüllt sein, damit ein Simulationsergebnis weiter betrachtet wird.

Da jedes Simulationsergebnis einen Auftreffwinkel zwischen Ball und Schläger enthält, muss überprüft werden, ob die Ballflugbahn geeignet verläuft, um den Ball an eben dieser Stelle treffen zu können. Der Auftreffwinkel lässt sich zuerst in eine Position und anschließend in Weltkoordinaten umrechnen. Nun kann überprüft werden, ob der Schläger passend positioniert werden kann, damit die Ballflugbahn durch eben diesen Punkt verläuft.

Zuerst wird der Auftreffpunkt auf die Arbeitsflächennormale projiziert. Dadurch können alle weiteren Tests eindimensional ablaufen, da nur noch der Abstand von der Arbeitsfläche als Größe interessant ist. Daraufhin wird für jeden Abschnitt der Ballflugbahn der Bereich berechnet, in dem sich der Auftreffpunkt befinden darf. Abbildung 4.15 veranschaulicht dies. Dort ist eine Ballflugbahn (rot) eingezeichnet, die sich durch den Bereich der möglichen Auftreffpunkte bewegt. An jeder beispielhaften Ballposition ist angegeben, welcher Bereich des Balls zum Treffen prinzipiell geeignet ist (grün).

Der gesamte Treffbereich für einen Abschnitt der Flugbahn ergibt sich aus der Vereinigung der Werte des Start- und Endpunkts eines einzelnen Abschnitts (blau). Nur wenn sich der in Weltkoordinaten umgerechnete und auf die Normale projizierte Auftreffpunkt aus dem Simulationsergebnis innerhalb dieses blauen Bereichs befindet, gehört er zu einer validen Lösung.

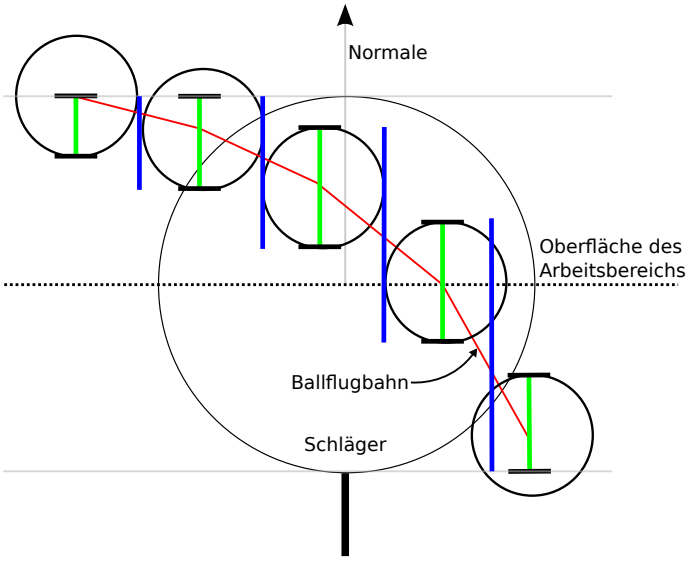


Abbildung 4.15: Veranschaulichung der möglichen Auftreffpunkt-Bereiche (blau) für eine Ballflugbahn (rot)

Filterung nach Übereinstimmung

Der nächste Filter bestimmt die Abweichung der Größen eines Simulationsergebnisses von ihren Sollwerten. Es wird sowohl für Ball- und Reflexionsgeschwindigkeit überprüft, wie stark sie vom geforderten Soll abweichen. Bei der Schlägergeschwindigkeit kann nicht der exakte Wert überprüft werden, da dieser gerade das Suchergebnis ist und damit nicht vorgegeben wurde. Die Abweichung der Schlägergeschwindigkeit von der hardwaredeterminierten Bewegungsebene kann jedoch überprüft werden. Für alle Größen gibt es immer ein hartes Limit, bei dessen Überschreitung diese Lösung verworfen wird. Unterhalb dieses Limits wird eine Bewertung, ein Rating, erstellt, dass ein Maß für die stärke der Abweichung darstellt.

Die Berechnung des Ratings für die Ballgeschwindigkeit setzt sich aus der Berechnung der Abweichung (4.56) und des Rating selbst (4.57) zusammen. $\vec{v}_{b_{in}}$ gibt dabei die gefundene und in Weltkoordinaten transformierte Lösung an und $\tilde{\vec{v}}_{b_{in}}$ die sich aus der Ballflugbahn ergebende Vorgabe. Zusammen mit der maximal erlaubten Abweichung $v_{b_{in},limit}$ wird das Rating r_{in} für diese Größe berechnet. Es bewegt sich immer im Bereich $r_{in} \in [0,1]$ wobei größere Werte besser sind. Die Berechnung von r_{out} für die Reflexionsgeschwindigkeit verläuft analog.

$$v_{b_{in},norm} = |\vec{v}_{b_{in}} - \tilde{\vec{v}}_{b_{in}}| \tag{4.56}$$

$$r_{in} = \begin{cases} 1 - \frac{v_{b_{in},norm}}{v_{b_{in},limit}} & \text{für} \quad v_{b_{in},norm} \leq v_{b_{in},limit} \\ 0 & \text{sonst} \end{cases} \tag{4.57}$$

Bei der Schlägergeschwindigkeit kann nur überprüft werden, wie weit der Vektor von der geforderten Bewegungsebene abweicht. Dafür wird in (4.58) der Winkel zwischen der in dieser Lösung vorgeschlagenen Schlägergeschwindigkeit $\vec{v}_s$ und der Normalen auf der Arbeitsfläche $\vec{n}$ bestimmt. Da dieser Winkel im Idealfall 90° beträgt, wird dieser Wert subtrahiert. Der zweite Teil der Berechnung in (4.59) verläuft analog zur Ballgeschwindigkeit. Auch hier wird auf einen Maximalwert $\phi_{diff,limit}$ begrenzt und ein Rating bestimmt.

$$\phi_{diff} = |\arccos(\frac{\vec{v}_s}{|\vec{v}_s|} \cdot \vec{n}) - \frac{\pi}{2}| \tag{4.58}$$

$$r_s = \begin{cases} 1 - \frac{\phi_{diff}}{\phi_{diff,limit}} & \text{für} \quad \phi_{diff} \leq \phi_{diff,limit} \\ 0 & \text{sonst} \end{cases} \tag{4.59}$$

Aus den Bewertungen kann, zusammen mit der internen Bewertung des Simulationsergebnisses r_{sim}, eine Gesamtbewertung erstellt werden (4.60). Es werden alle Lösungen verworfen, die eine Teilbewertung haben, die ausserhalb des Limits war und somit 0 ist. Von den übrigen Lösungen werden die 100 Bestbewerteten gespeichert und an die nächste Filterstufe übergeben. Das Sammeln dieser Lösungen geschieht dabei mittels einer Heapdatenstruktur, die sich in Tests als effizienteste Umsetzung herausgestellt hat.

$$r = r_{sim}\ r_{in}\ r_{out}\ r_s \tag{4.60}$$

Abschlusssimulation

Die finale Filterstufe hat zum Ziel, das Ballverhalten erneut zu simulieren und zwar mit möglichst vielen der vorgegebenen Größen. Damit lässt sich überprüfen, wie sich der Ball nicht nur in den näherungsweise passenden, gefunden Simulationsergebnissen verhält, sondern unter möglichst realistischen Bedingungen.

Um zu diesem Ziel zu gelangen, muss zuerst der Auftreffpunkt und die Schlägerposition möglichst exakt bestimmt werden. Die Schwierigkeit besteht hier in der leicht ellipsoiden Form des Arbeitsbereichs, sodass die Positionen näherungsweise bestimmt werden. Dafür wird am bisherigen, ungenauen Auftreffpunkt der exakte Arbeitsbereichsradius bestimmt. Damit lässt dieser sich in dem kleinen Bereich, der hier relevant ist sehr gut über eine Kugel annähern. Der Schnitt zwischen dieser Kugel und dem Segment der Ballflugbahn gibt uns einen exakteren Auftreffpunkt. Aus diesem wird die Schlägerposition berechnet. Diese muss, vorgegeben durch die Hardware auf dem Arbeitsbereich liegen, und kann somit per Projektion auf den Arbeitsbereich exakt auf diesen abgebildet werden. Dies liefert die finale Schlägerposition zusammen mit der finalen Normalen auf dem Arbeitsbereich. Hieraus läßt sich wiederum der finale Auftreffpunkt berechnen.

Die finale Normale erlaubt es, die Schlägergeschwindigkeit exakt orthogonal zu dieser auszurichten, denn nur diese Bewegung kann der Roboter in dieser Position durchführen. Dafür wird der Geschwindigkeitsvektor längenerhaltend auf die durch die Normale definierte Ebene projiziert.

Mit den nun zur Verfügung stehenden Größen, der Ballgeschwindigkeit, dem Auftreffpunkt sowie der Schlägergeschwindigkeit kann eine Simulation des Ballverhaltens durchgeführt werden. Einzelne Simulationsläufe sind schnell genug, um diese in Echtzeit auszuführen. Von daher ist es möglich, die Reflexionsgeschwindigkeit mit den vorliegenden exakteren Werten erneut zu ermitteln. Die Übereinstimmung dieses Ergebniswerts der Simulation mit der Vorgabe ist entscheidend. Um auch hier eine Einordnung der Ergebnisse vornehmen zu können, wird erneut eine Bewertung erstellt, die sich nur aus der Reflexionsgeschwindigkeitsabweichung sowie dem Rating des zugrunde liegenden Simulationsergebnisses zusammensetzt. Die Berechnung erfolgt analog zu (4.56), (4.57) und (4.60).

Anhand dieser Bewertung kann eine endgültige Sortierung der verbleibenden Lösungsvorschläge erstellt werden. Damit ist die Suche von möglichen Schlägerpositionen und -geschwindigkeiten für ein gegebenes Segment der Ballflugbahn vor dem Kontakt mit dem Roboter sowie der Reflexionsgeschwindigkeit nach dem Kontakt, beendet.

4.5 Gesamtablauf

Die einzelnen Komponenten der Steuerung sind für sich alleine genommen nicht in der Lage dem Roboter ein kontrolliertes Verhalten zu geben. Dafür müssen sie sinnvoll zusammen arbeiten. Dieses Zusammenspiel ist im Folgenden beschrieben. Dabei ist zu beachten, dass die Bewegungsplanung (Abschnitt 4.4) bisher nur in einer simulierten Umgebung Verwendung findet. Die aktuelle Roboterhardware ist den Anforderungen an die Genauigkeit, die das gezielte Zurückspielen stellt noch nicht gewachsen. In der Bescheibung wird deshalb einmal der konkret in Hardware verwendete Ablauf und zusätzlich der geplante aber bisher noch nicht auf der Hardware stattfindende Ablauf beschrieben.

Das Verhalten beginnt immer damit, dass der Roboter in eine Ausgangsposition fährt und darauf wartet, dass ein Ball auf ihn zugeworfen wird. Ist dies der Fall, liefert der BallTracker (Abschnitt 4.1.2) die erkannten Flugbahnen. Diese werden jeden Frame, d.h. mit ca. 50 Hz aktualisiert und enthalten die veränderten aktuellen Ballposition sowie eine genauer werdende Schätzung der verbleibenden Bewegung.

An dieser Stelle beginnen der aktuelle und der geplante Ablauf sich zu unterscheiden. Derzeit wird der Schnitt zwischen Ballflugbahn und der Oberfläche des Arbeitsraums gesucht. Diese Position wird der Bewegungssteuerung (Abschnitt 4.3) mitgeteilt, um den Schläger dorthin zu bewegen. Das führt dazu, dass der Ball im Idealfall exakt orthogonal zur stillstehenden Schlägeroberfläche auftrifft und, nur durch den Energieverlust verlangsamt, zurück zum Werfer reflektiert wird. Anschließend fährt der Roboter den Schläger zurück in die Ausgangsposition und wartet auf den nächsten Ball.

Der geplante, aber bisher nur in Simulation erprobte Ansatz bindet beim Eingehen einer Ballflugbahn die Bewegungsplanung (Abschnitt 4.4) ein. Diese benötigt jedoch neben der Ballflugbahn auch eine gewünschte Reflexionsrichtung. Bisher existiert noch keine Komponente im System, die eine solche generieren könnte. Es ist jedoch ersichtlich, dass diese vom derzeit aktiven Spiel abhängig wäre. So müssten beispielsweise Mit- und Gegenspielerpositionen ausgewertet werden, um eine für das Spiel und den Spielspaß optimale Flugbahn des Balls nach dem Roboterkontakt zu bestimmen. Da diese Flugbahn rein schwerkraftgetrieben wäre, läge damit auch der benötigte Reflexionsgeschwindigkeits-Vektor beim Roboter-Ball-Kontakt vor.

In der Bewegungsplanung wird mit Hilfe der Lookup-tables für jeden Abschnitt der Flugbahn versucht, aus den vorsimulierten Ergebnissen diejenigen auszuwählen, die die gestellten Anforderungen am Besten erfüllen. Anschließend wird über alle Flugbahnsegmente hinweg das beste Ergebnis ausgewählt. Die ermittelte Schlägergeschwindigkeit wird an die Bewegungssteuerung (Abschnitt 4.3) übermittelt, die diese in Gelenkgeschwindigkeiten überführt. Der PD-Regler (Abschnitt 4.3.2) generiert daraufhin die Drehmoment-Werte für die Motoren.

Nachdem der Schläger den Ball getroffen hat, oder wenn der Ball nicht mehr erreicht werden kann, fährt der Roboter seinen Schläger in die Ausgangsposition zurück und wartet auf den nächsten Ball.

5 Experimente

Diese Kapitel befasst sich mit der Evaluierung der entwickelten Komponenten und Konzepte. Es wird auf die Kalibrierung eingegangen, die notwendig ist, bevor Experimente mit der Hardware durchgeführt werden. Anschließend werden relevante, nicht per Unittest abdeckbare, Verhaltensweisen der Komponenten analysiert und bewertet. Eine Einordnung der rein simuliert betrachteten Bewegungsplanung gefolgt von den bisherigen realen Einsätzen des Systems schließen das Kapitel ab.

5.1 Kalibrierung

Bevor der Roboter sinnvoll verwendet werden kann, müssen die internen Parameter kalibriert werden. Da das System aus einzelnen Komponenten besteht, müssen deren Parameter einzeln, mit speziellen Vorgehensweisen auf sinnvolle Werte gesetzt werden. Wie dies geschieht, wurde bereits kurz in Abschnitt 4.1.2 beschrieben. Die zu kalibrierenden Bestandteile sind die Kameras, die dynamischen Modellparameter des MHT sowie die Servomotoren. Es ist jedoch nicht nötig, diese Prozedur vor jedem Systemstart durchzuführen, sondern nur, wenn sich am System etwas verändert hat. Dies kann beispielsweise nach dem Austausch einzelner Bestandteile, wie der Motoren oder der Kameras, nötig werden.

Die Algorithmen und Programme zur Durchführung der Kalibrierung waren nicht Teil dieser Arbeit. Für diese Aufgabe wurde auf bereits bestehende, für den BallTracker entwickelte Vorgehensweisen zurückgegriffen.

Kamerakalibrierung

Bei der Kalibrierung der Kameras werden deren Ausrichtung zueinander sowie die intrinsischen Kameraparameter bestimmt. Dazu zählen beispielsweise die Verschiebung des Bildmittelpunkts, die Brennweite oder die Linsenverzerrung. Dieser Vorgang wird mit Hilfe des üblichen Schachbrettmusters vorgenommen.

Nach der Aufnahme von 9 Schachbrettmustern, jeweils unterschiedlich gekippt und gedreht, konnte ein mittlerer quadrischer Fehler von nur noch $(0.62, 0.88)$ Pixeln erreicht werden [LBHF12]. Dieser Vorgang muss immer wiederholt werden, nachdem Kameras ausgetauscht oder ausgebaut werden. Aber auch ein heftiger Stoß gegen eine Kamera kann dazu führen, dass diese sich leicht verschiebt und damit die Kalibrierung an Genauigkeit verliert.

Modell-Parameter

Die Modellparameter werden innerhalb des MHT verwendet, um das Verhalten von fliegenden Bällen vorherzusagen. Da Bälle hauptsächlich von der Schwerkraft und der Luftreibung beeinflußt werden, müssen diese Parameter entsprechend den externen Gegebenheiten eingestellt werden. Dies geschieht, wie in Abschnitt 4.1.2 erläutert, indem Bälle in Richtung des Roboters geworfen und aus diesen Ballflugbahnen die Parameter gewonnen werden.

Dieser Vorgang sollte mit verschiedenen Bällen durchgeführt werden, wenn unterschiedliche Bälle im Spiel verwendet werden. Natürlich muss er erneut ausgeführt werden, wenn die Bälle ausgetauscht werden. Die Schwerkraft muss, auch wenn der Roboter neu aufgebaut wurde, nicht unbedingt neu vermessen werden. Durch die aufmontierte 2D-Wasserwaage kann auch unter diesen

Bedingungen sichergestellt werden, dass die Ausrichtung des Systems zum Boden unverändert ist und sich somit der Schwerkraftvektor nicht relativ zum Roboter gedreht hat.

Die Kalibrierung des Vorwissens über die möglichen Ballabwurfpositionen und -geschwindigkeiten wurde auch bereits im besagten Abschnitt erwähnt. Diese Parameter müssen, wenn sie korrekt und mit genügend Varianz ermittelt wurden, nicht neu eingestellt werden. Da der Bereich, aus dem die verwendeten Bälle gut geworfen werden können nur gering schwankt ist hier kaum mit starken Veränderungen zu rechnen.

Servomotoren

Bei der Kalibrierung der Motoren werden mehreren Parameter ermittelt. Zum einen sind dies die Mittelstellungen und Skalierungsfaktoren der Servos selbst. Diese beiden Größen, j_{off} und j_{scale} werden jeweils für beide Servos ermittelt und entsprechend der Umrechnung in Abschnitt 4.3.1 verwendet. Dort dienen sie zur Ermittlung der hardwareunabhängigen Positionen in Radiant aus den servoeigenen Positionsangaben.

Der andere Parameter, der mittels der Servomotoren bestimmt werden kann, ist die Hand-Auge-Kalibrierung. Bei dieser wird die Umrechnung von Kamera- in Roboter-Basis-Koordinaten ermittelt. Erst mit einer solchen Transformation ist es möglich, eine Beziehnung zwischen Ballflugbahnen, die in Kamerakoordinaten erstellt werden und dem Schläger herzustellen.

Als Eingangswerte benötigt der verwendete Algorithmus eine Menge an Roboterposen und die zugehörigen Bilder beider Kameras. In der aktuellen, für dieses System entwickelten, Implementierung fährt der Roboter insgesamt 25 Posen ab, die in einem 5x5-Raster angeordnet sind 5.1(c). Es ist darauf zu achten, dass jeweils in beiden Kamerabildern die vollständige Schläger-Kugel sichtbar ist. Aus der Position des Schlägers im Bild (Abbildung 5.1(a) – 5.1(b)), den Servoposition und der Vorwärtskinematik können die gesuchten Parameter und Transformationen bestimmt werden.

Die mittlere quadratische Abweichung zwischen dem Schläger im Bild und der, mit den gewonnenen Parametern geschätzten Position des ins Bild projiziteren Schlägers liegt bei $(3.8120, 2.2419)$ Pixel. Die Schätzung für den Radius hat eine Genauigkeit von 0.7307 Pixel. [LBHF12]

Dieser Kalibriervorgang muss nur erneut durchgeführt werden, wenn der Schläger abmontiert oder die Motoren getauscht wurden. Im normalen Betrieb sollten sich die ermittelten Werte nicht verändern, da diese Komponenten des Roboters fest miteinander verbunden sind.

5.2 Bewertung der Eingangsdaten

Damit das Gesamtsystem eine hohe Performance liefern kann, ist es wichtig zu untersuchen, welche Qualität die Daten haben, die von anderen Komponenten in das System hinein gelangen. Da Entscheidungen auf Grund dieser eingehenden Informationen getroffen werden, können Fehler oder Abweichungen dort Auswirkungen auf das gesamte, im Rahmen dieser Arbeit implementierte System haben. Als Hauptquelle für externe Daten werden die Ballflugbahnen des BallTrackers sowie die Informationen der Servomotoren genauer betrachtet.

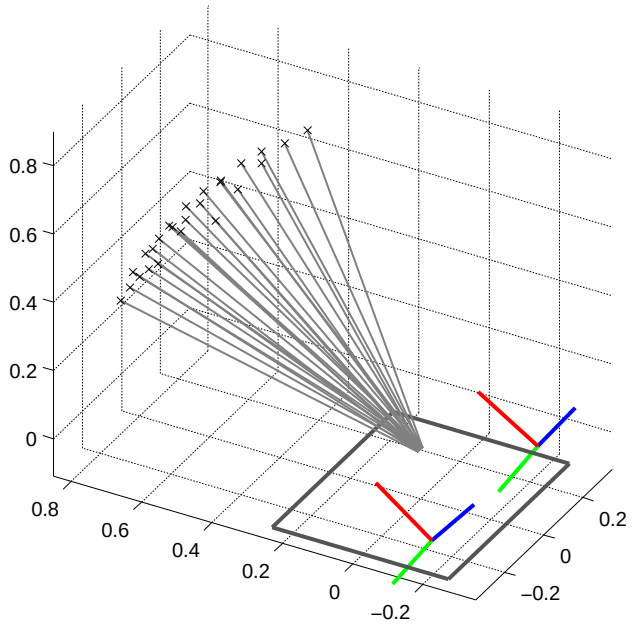
BallTracker

Der BallTracker liefert die Ballflugbahnen, sobald eine solche aus den erkannten Kreisen rekonstruiert werden kann. Oft ist dies schon nach wenigen Frames möglich. Die Schätzung der Ballposition ist dabei zu diesem Zeitpunkt bereits recht genau, die der Geschwindigkeit weniger. So wird der Geschwindigkeitsvektor meist innerhalb der ersten Frames zu flach angesetzt und im weiteren Verlauf dann so korrigiert, dass der Ball in einem höheren Bogen fliegt. Die Flugweite nimmt gleichzeitig entsprechend ab.

Der Fehler tritt kaum orthogonal zu den Sichtachsen der Kameras auf, sondern ist am stärksten parallel zu diesen ausgeprägt. Das lässt sich damit begründen, dass in orthogonaler Richtung die Informationen direkt aus den erkannten Kreispositionen abgeleitet werden können. Die Tiefeninformationen parallel zur Sichtachse müssen aufwendiger rekonstruiert werden. Sie sind nur indirekt in der Ballgröße und dem Versatz des Balles in den beiden gleichzeitig aufgenommenen Kamerabildern enthalten und damit weitaus fehleranfälliger. Abbildung 5.2(a) und (b) zeigen beispielhafte Flugbahnen, an denen sich das Beschriebene erkennen lässt.

(a) Linke Kamera

(b) Rechte Kamera



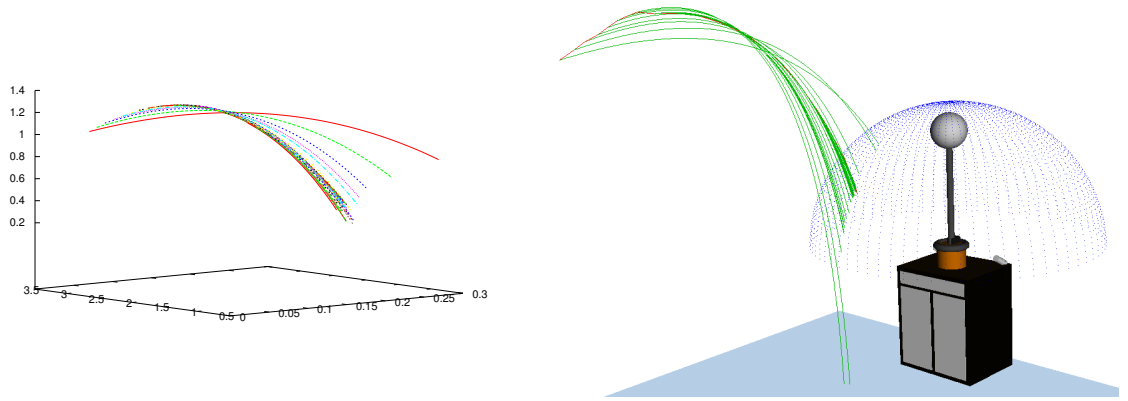
(c) Kalibrierungsposen

Abbildung 5.1: (a) – (b) Sicht der linken und rechten Kamera bei der Kalibrierung mit hervorgehobenem Schläger (c) Roboterposen bei der Kalibrierung, Kreuze markieren Schlägermitte [LBHF12]

Dieses Verhalten wurde durch die Auswertung von 11 exemplarischen Ballflugbahnen verifiziert. Dafür wurde jeweils ein Ball geworfen und die Flugbahn mit den zugehörigen Vorhersagen des Verhaltens aufgezeichnet. Für die erste und letzte Schätzung werden daraus die Positionen ermittelt, an denen der Ball die x-y-Ebene auf Höhe des Roll-Gelenks schneidet. Aus diesen beiden Werten kann der Versatz in x- und y-Richtung bestimmt werden. Die Mittelwerte und Varianzen über alle 11 Würfe sind in Tabelle 5.1 angegeben. Entlang der x-Achse, die ungefähr orthogonal zu den Kamera-Sichtachsen verläuft ist der Mittelwert und die Varianz sehr gering. Parallel zur y-Achse liegt der Mittelwert hingegen um ca. den Faktor 12 höher. Dieser Wert liegt nah am Radius des Arbeitsraums von 0.84 m. Auch die Varianz ist sehr viel höher und belegt eindeutig, dass die Schätzung der Ballgeschwindigkeit in diese Richtung nicht nur wesentlich ungenauer sondern auch mit starken Schwankungen in der Genauigkeit behaftet ist.

Achse	**Mittelwert** $[m]$	**Varianz** $[m^2]$
X	0.0592	0.0031
Y	0.7415	0.0872

Tabelle 5.1: Mittelwert und Varianz der Abweichung zwischen erster und letzter Track-Schätzung



(a) Einzelne Vorhersagen eines Tracks

(b) Positionen und Vorhersagen in Bezug zum Roboter und Arbeitsbereich

Abbildung 5.2: Beispiele für vom Balltracker erkannte Ballpositionen und Vorhersagen zum weiteren Verhalten

Servo-Motoren

Die beiden Motoren für das Roll- und Tilt-Gelenk liefern verschiedene Daten zurück an die Bewegungssteuerung. Wie dort bereits erwähnt, werden diese Werte, bis auf die Positionsangaben nur mit 10 Hz aktualisiert. Besonders Geschwindigkeitsangaben, die nur jeden 10. Regelzyklus angepasst werden sind wertlos und können demnach nicht verwendet werden. Daten, wie die Betriebsspannung oder die interne Temperatur liefern zwar Informationen über Fehlerzustände des Systems, sind aber für die reguläre Funktion nicht nötig. Somit ist dort die langsame Aktualisierungsfrequenz unkritisch und hinnehmbar.

Die Positionsangaben sind demnach die einzige direkte Eingangsgröße der Motoren, die für den PD-Regler verwendet werden können. Die Genauigkeit dieser Angaben ist, nach der beschriebenen Kalibrierung, hoch genug. Jedoch müssen Geschwindigkeits- und, wenn benötigt, Beschleunigungsangaben aus den Positionsdaten berechnet werden (Abschnitt 4.3.1). Die servointernen Positionsangaben werden, im Drehmoment-Modus der Motoren, in 0.06°-Schritten ausgegeben. Diese Rasterung führt, wie jegliche Diskretisierung von analogen Werten, zu Rundungsfehlern. Wenn der echte, analoge Wert nah an der Grenze zwischen zwei diskreten Werten liegt, können minimale Veränderungen dazu führen, dass der diskrete Wert zwischen zwei Schritten schwankt.

Das dieser Effekt bei den Servomotoren auftritt verdeutlicht Abbildung 5.3. Dort ist die gemessene Position eines eigentlich unbewegten Motors dargestellt. Das Signal wandert in dem Plot, vermutlich durch externe Einflüße, von $-0.06°$ nach $0.06°$. Besonders an den Übergängen von einer Diskretisierungsstufe zur nächsten wird das Springen zwischen diesen sehr deutlich.

In den Positionsangaben ist dieses Verhalten noch zu tolerieren, in den daraus berechneten Geschwindigkeiten jedoch nicht. Die Sprünge finden von einem Auslesen zum nächsten statt, der Positionswert ändert sich somit bei 100 Hz Auslesefrequenz innerhalb von 0.01 s um 0.06°. Das entspricht einer berechneten Winkelgeschwindigkeit von $6°/s$. Eine solche Fehlmessung kann zu Problemen im Regler führen, der sich auf diese Werte verlässt. Da die Frequenz dieser Sprünge jedoch zumeist weit über 20 Hz liegt, sollte dieser Effekt mittels des Tiefpassfilters eliminiert werden können.

In Anbetracht dieses Verhaltens wird deutlich, das Beschleunigungswerte, die aus den Positionsangaben berechnet wurden noch massivere Fehler enthalten. Da die Schwankungen meist einzelnen Spitzen entsprechen, dass heißt, ein Wechsel auf die nächste Diskretisierungsstufe und sofort wieder zurück stattfindet, verdoppelt sich der Fehler in den Beschleunigungswerten. So wäre die Beschleunigung, die sich aus einer solchen Spitze ergibt $\frac{6-(-6)}{0.01} = 1200°/s^2$. Dieser Wert liegt bei über 50 % der gemessenen maximalen Beschleunigung des Systems.

Ein weiterer wichtiger Punkt, der die Motoren betrifft, sind die Angaben zu möglichen und echten Geschwindigkeiten und Beschleunigungen. So ist im Datenblatt des Motors zwar die Leer-

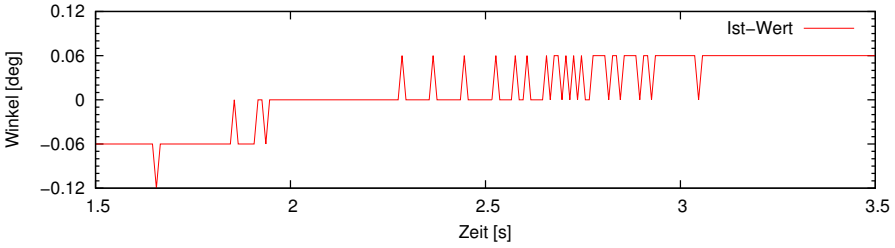


Abbildung 5.3: Diskretisierungsfehler eines Servomotors

laufdrehzahl von $546°/s$ angegeben, jedoch wird diese nicht erreicht, sobald der Motor mit einer Last verbunden ist. Es fehlen jegliche Angaben dazu, in welchem Verhältnis Motordrehzahl und -last stehen. Somit ist die einzige Methode, um eine realistische Maximalgeschwindigkeit zu erhalten, diese zu messen. Das System kommt dabei durchschnittlich auf einen Wert von $180°/s$ bei 70 % des Maximaldrehmoments. Da die Last auf den Motor jedoch mit der Position des Schlägers stark variiert, ist dieser Wert nur als grobe Schätzung zu verstehen. Erst mit einer vollständigen Messreihe ließe sich ein korrektes Modell des Motors aufstellen, anhand dem man berechnen könnte, wie schnell dieser sich in bestimmten Situationen, gegeben einem bestimmten Soll-Drehmoment bewegen kann. Ein solches Modell würde eine wesentlich effektivere Regelung der Geschwindigkeit ermöglichen, da sich durch Kompensationsrechnungen ein lineares Verhalten der Regelstrecke auf die Regelgröße erreichen ließ.

Zur erreichbaren Beschleunigung finden sich im Datenblatt des Motors keine Angaben. Dieser Wert läßt sich somit nur durch Messungen bestimmen. Im aktuellen Aufbau wird als grober Richtwert $1800°/s^2$ erreicht. Für eine Modellierung des Motorverhaltens würde auch hier ein genauerer, lastabhängiger Wert benötigt.

5.3 Bewegungssteuerung

Die Bewegungssteuerung enthält die grundlegende Ansteuerung der Motoren. Von ihrem effizienten und korrekten Funktionieren sind alle darauf aufbauenden Bestandteile des Systems abhängig. Für die Hardwareabstraktion sind keine weiteren Experimente nötig. Sie hat ein deterministisches, durch Unittests überprüfbares Verhalten, das im Rahmen der Erwartungen liegt. Der Regler hingegen hat ein sehr dynamisches Verhalten, das kaum mit einfachen, regelbasierten Tests kontrolliert werden kann. Hier werden im Folgenden einzelne Verhaltensweisen und Reaktionen betrachtet und analysiert. Anschließend werden die trapezförmigen Bewegungsmuster genauer untersucht und ihre Vor- und Nachteile herausgestellt. Die Schnittstellen-Komponente wiederum bedarf keiner weiteren Experimente oder Analysen.

Reglerverhalten

Das normale Verhalten des PD-Reglers ist in Abbildung 5.4 dargestellt. Dort sind zwei Sprungantworten für das Tilt-Gelenk zu sehen, an denen sich mehrere Eigenschaften des Systems und des Reglers ablesen lassen. So zeigt sich, dass die Stellgröße, also das vorgegebene Drehmoment nach dem Sprung sofort nachzieht. Beim Erreichen des Soll-Winkels ist ein unruhiges Verhalten der Stellgröße zu erkennen. Dieses resultiert aus dem Nachschwingen des Schlägers, sichtbar im Ist-Winkel, welches der Regler versucht zu kompensieren. Auch der Backlash ist eindeutig sichbar und zwar als der kurze Bereich nach dem Sprung, in dem die Ist-Änderungsrate sehr hoch ist, um danach wieder auf nahe null abzusinken und erneut langsamer anzusteigen. Dieses Verhalten lässt sich damit erklären, dass die erste, schnelle Bewegung rein innerhalb des Getriebes stattfinden und das Abbremsen durch das Erreichen der Grenzen des internen Spiels verursacht wird. Anschließend beschleunigt der Servo mit Kontakt zur externen Last erneut.

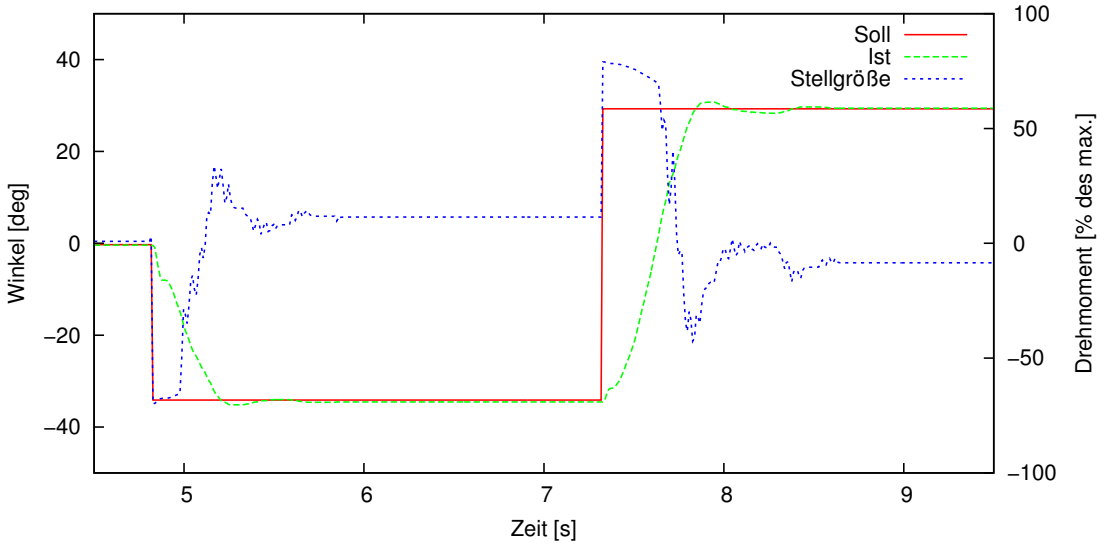


Abbildung 5.4: Verhalten des PD-Reglers im normalen Betrieb

Schwerkraftkompensation im Regler

Wie in Abschnitt 4.3.2 erläutert, wird ein PD-Regler verwendet. Statt einem I-Anteil wird auf die direkte Kompensation des Hauptauslösers für solche Fehler gesetzt. Mittels dieser Schwerkraftkompensation kann, zumindst theoretisch, eine höhere Performance erreicht werden, da nicht erst auf eine vorhandene Abweichung reagiert sondern diese bereits im voraus einkalkuliert wird. Das dieser Effekt auch in der Praxis zu beobachten ist, kann anhand von beobachtetem Regler- und Roboterverhalten belegt werden. So zeigen Abbildung 5.5(a) und (b) die relevanten Größen für das Roll- respektive Tilt-Gelenk.

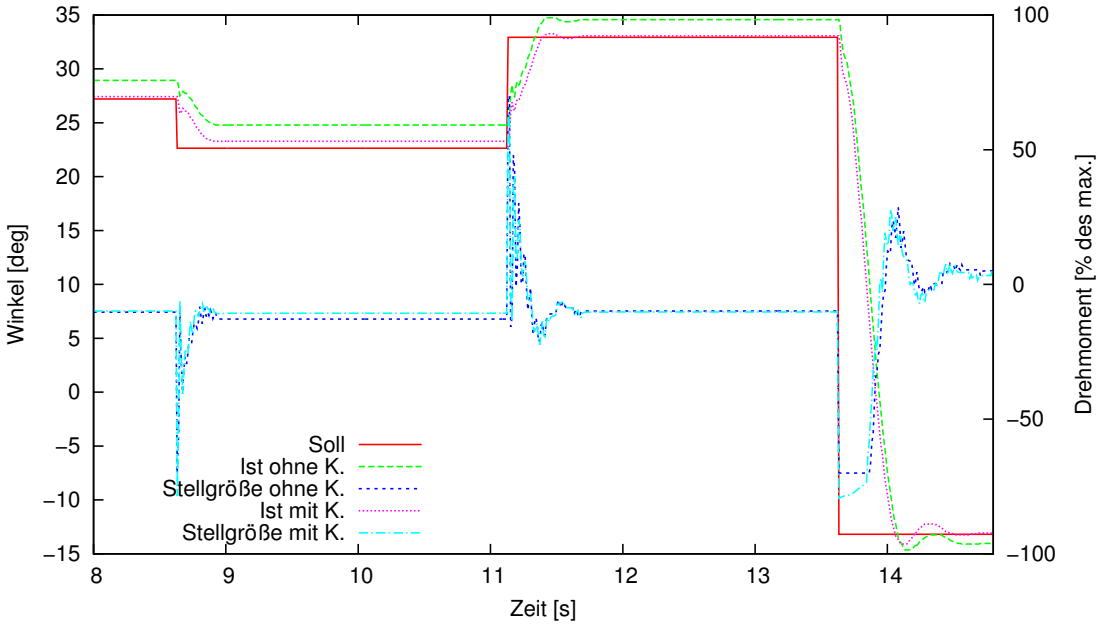
Es ist deutlich zu erkennen, dass die Ist-Werte ohne Kompensation wesentlich weiter vom Soll entfernt sind als mit aktiver Korrektur. Für das Rollgelenk ist die Abweichung statt 2.16° / 1.65° nur 0.65° / 0.15° bei 10 bzw. 13 s. Für das Tilt-Gelenk liegen die Werte bei 0.23° / 0.27° gegenüber 0.95° / 1.15°. Der Fehler konnte somit um mehrere Größenordnungen verringert werden. Besonders mit steigendem Soll-Winkel wird der Effekt umso sichtbarer, da der Einfluß der Schlägermasse zunimmt.

Aus dem Fehlen des I-Anteils resultiert jedoch, dass die geringe, verbleibende Abweichung nicht korrigiert werden kann. Auch die idealisierte Betrachtung des Schlägers als Punktmasse am masselosen Stab sowie die Missachtung von dynamischen Effekten führen zu einem geringen Fehler. Dennoch ist die Endgenauigkeit des Reglers für die aktuelle Hardware vollkommen ausreichend und wird weitaus stärker durch Effekte wie Nachschwingen und Backlash beeinflusst.

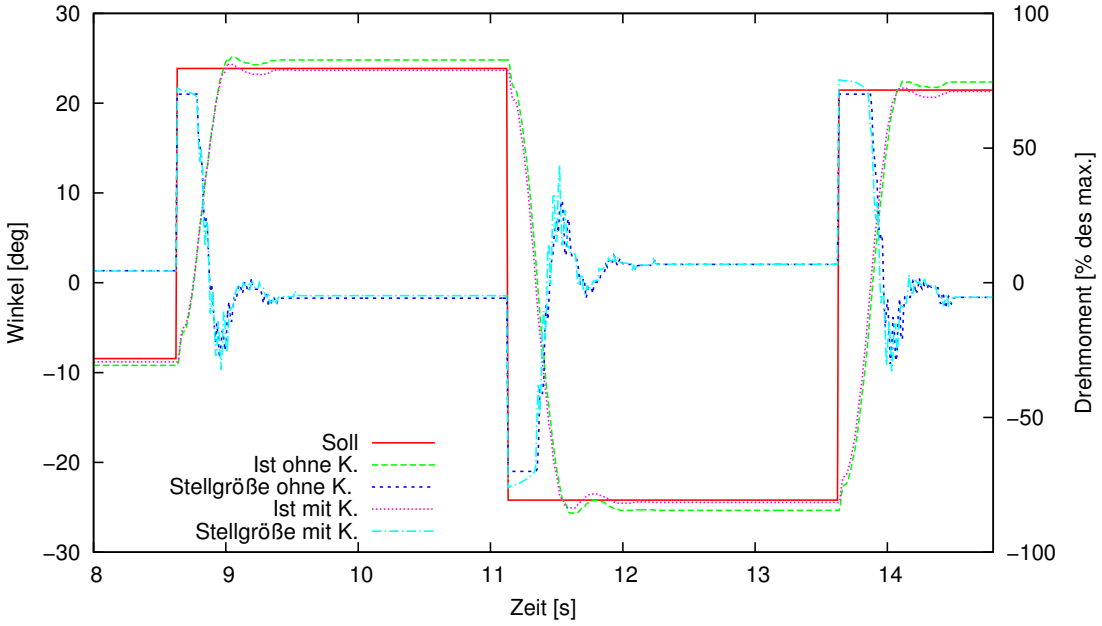
Servo-Backlash

Als Backlash bezeichnet man in der Mechanik das Spiel zwischen zwei beweglichen Komponenten. Im Robotersystem tritt es innerhalb des Getriebes der Servomotoren auf. Da die dort verbauten Zahnräder nicht exakt ineinander greifen kann beispielsweise ein Zahnrad minimal bewegt werden, ohne das ein angrenzendes sich mitbewegt. Bei der Hintereinanderschaltung mehrerer Zahnräder, wie es in einem kompakten Getriebe mit einer Übersetzung von 184:1 zwingend notwendig ist, vervielfacht sich das Spiel mit der Übersetzung. So wäre ein minimales Spiel am Zahnrad antriebsseitig an der Abtriebsseite um den Faktor 184 größer.

Dieser Effekt führt dazu, dass eine Bewegung an einer Seite des Getriebes nicht an der anderen Seite ankommen muss. Es existiert immer ein Bewegungsbereich, in dem die vollständige Bewegung auf der einen Getriebeseite im Spiel der Zahnräder verloren geht. Erst wenn eine Rotation den Randanschlag des Spiels erreicht hat, wird eine Bewegung an die gegenüberliegenden Seite übertragen. In dieser Stellung haben alle Zähne der Zahnräder direkten Kontakt zum nächsen Zahnrad.



(a) Roll-Gelenk



(b) Tilt-Gelenk

Abbildung 5.5: Vergleich zwischen Soll- und Ist-Werten mit und ohne Kompensation des Schwerkrafteinflusses

Wenn die Rotationsbewegung jedoch umgekehrt wird, muss erst der vollständige Backlash-Bereich durchfahren werden bis auch diese Rotation an der Gegenseite ankommt.

Der Backlash-Effekt führt zu massiven Problemen bei der Implementierung des Reglers zur Ansteuerung der Motoren. Er limitiert die sinnvollen Werte für die beiden Parameter k_p und k_d, macht den Einsatz des Tiefpassfilters zwingend notwendig und verursacht dennoch Schwingungen und Ungenauigkeiten.

Die Schwingungen treten auf, da die Antriebsseite innerhalb des Spiels bewegt werden kann, ohne das diese Bewegung abtriebsseitig abgebildet wird. Da die Bewegung nicht von einer ab-

triebsseitig vorhandenen Last gebremst wird, sondern ihr nur das Trägheitsmoment des Getriebes entgegenwirkt, ist diese Bewegung nahezu ohne Energieeinsatz ausführbar. Sie wird jedoch vom antriebsseitigen Drehencoder erkannt und demnach vom Regler beachtet. Das kann, wenn es nicht verhindert wird, zu einem Aufschwingen innerhalb des Getriebes führen. Im konkreten Fall, d.h. für die verwendeten Motoren, Regler und Regelfrequenzen liegt die Amplitude ohne den Filter bei ca. 3° und die Frequenz bei ca. 24 Hz.

Dieser Bereich, in dem An- und Abtiebsseite entkopplt sind, hat Auswirkungen darauf, wie sich der Regler einstellen lässt. Normalerweise würde man, um eine schnellere Reaktion des Reglers auf Änderungen des Soll-Wertes zu erreichen, den P-Anteil weiter erhöhen. Dies führt jedoch dazu, dass der Tiefpassfilter des Reglers nicht mehr ausreichend ist, um ein Aufschwingen zu verhindern. Das Resultat eines solchen Verhaltens ist in Abbildung 5.6 eindeutig zu erkennen. Wie dort zu sehen ist, verstärkt der Regler in einem solchen Fall eine anfängliche, minimale Regelabweichung schnell zu einem Schwingen zwischen extremen Drehmoment-Steuerwerten. Der Ausschlag des Ist-Wertes findet hier vollständig innerhalb des Backlash-Bereichs statt, beinahe ohne das die externe Last in Form des Schlägers sich mitbewegt. Der Effekt ist nur in den Ist-Werten sichtbar, da diese antriebsseitig gemessen werden. Die Amplitude der Schwingung beträgt in diesem Fall ca. 1°.

Bei einer höheren Regelfrequenz wäre die Backlash-Schwingung entsprechend höher und ließe sich leichter filtern. Hardwareseitig würde ein Drehmomentsensor abtriebsseitig das Erkennen und vollständige Verhindern dieses Verhaltens erlauben. Da beide Möglichkeiten durch die vorhandene Hardware nicht umsetzbar sind, bleibt nur das Unterdrücken der Schwingungsfrequenz innerhalb des Reglers mittels des Tiefpassfilters auf dem D-Anteil.

Das Backlash wirkt jedoch nicht nur, wenn antriebsseitig am Getriebe eine Bewegung erfolgt. Auch der Fall, dass abtriebsseitig eine Bewegung erfolgt, die keine Auswirkungen auf die Antriebsseite hat, tritt auf. Eine solche Bewegung verursacht keine Bewegung am antriebsseitig montierten Drehencoder und führt somit zu keiner Veränderung des Ist-Wertes. Es ist dadurch möglich, den Schläger im Bereich von ca. ±3° vollkommen frei zu bewegen, ohne dass der Regler diese Abweichung als Regelfehler wahrnimmt. Der Winkel entspricht auf Höhe der Schlägermitte einem Spiel und damit einer nicht messbaren Ungenauigkeit von ca. 0.12 m.

Die Auswirkungen des Backlashs können unter gewissen Bedingungen vorhergesagt werden. Somit ist es möglich die exakte Position des Schlägers herauszufinden. Dafür wird das Wissen verwendet, dass eine direkte Kraftübertragung ohne jegliches Spiel stattfindet, sobald der Randanschlag des Spiels erreicht wurde, d.h. alle Zähne direkten Kontakt haben. Dies deckt sich mit der Beobachtung, dass die stärksten Schwingungen hauptsächlich im Bereich um 0° des jeweiligen Gelenks auftreten. In diesem Bereich ist die Wirkung der Schwerkraft auf das jeweilige Gelenk minimal. Auch während einer gleichmäßigen Bewegungen treten keine Schwingungen auf. Nur zu deren Beginn und Ende kann eine Bewegung innerhalb des Spiels beobachtet werden. Die Erklärung dafür ist, dass, wenn ein Drehmoment auf das Gelenk wirkt, dieses an einen der beiden Außenanschläge des Spiels gedrückt wird und somit eine direkte Verbindung zwischen An- und Abtriebsseite besteht. Dadurch kann das Verhalten des Getriebes ohne Spiel angenommen werden und ist somit wesentlich einfacher beschreibbar. Erst wenn das Drehmoment nachlässt, können wieder Bewegungen innerhalb des Getriebes auftreten. In den beiden beschriebenen Beispielen wird das Drehmoment durch das Eigengewicht des Schlägers bzw. durch die Trägheit bei der Bewegung erzeugt.

Diese Effekt könnte ausgenutzt werden, indem beispielsweise versucht wird, den Ball immer in Bewegung zu treffen. Die Geschwindigkeit und das durch die Kreisbewegung ausgelöste Drehmoment müsste hoch genug sein, um den Schläger immer an einem Anschlag des Spiels zu halten, auch wenn die Gravitation dem entgegenwirkt. Dadurch würde sich theoretisch eine höhere Genauigkeit erreichen lassen. Bisher wird dieser Effekt noch nicht genutzt, da eine neue Revision der Antriebseinheit das Problem direkt an der Ursache, dem Getriebe, beheben könnte.

Idealisierte Bewegungsmuster

Die idealisierten Bewegungsmuster ermöglichen eine Ansteuerung über die Vorgabe von Zielzeit, -position und -geschwindigkeit. Zum Vermeiden von Sprüngen in der Geschwindigkeit wird auf die so generierten Segmente mit konstanter Geschwindigkeit eine Interpolation mittels Bézier-Kurven angewendet. Abbildung 5.7 zeigt die Soll- und Ist-Werte beider Gelenke für zufällig angefahrene

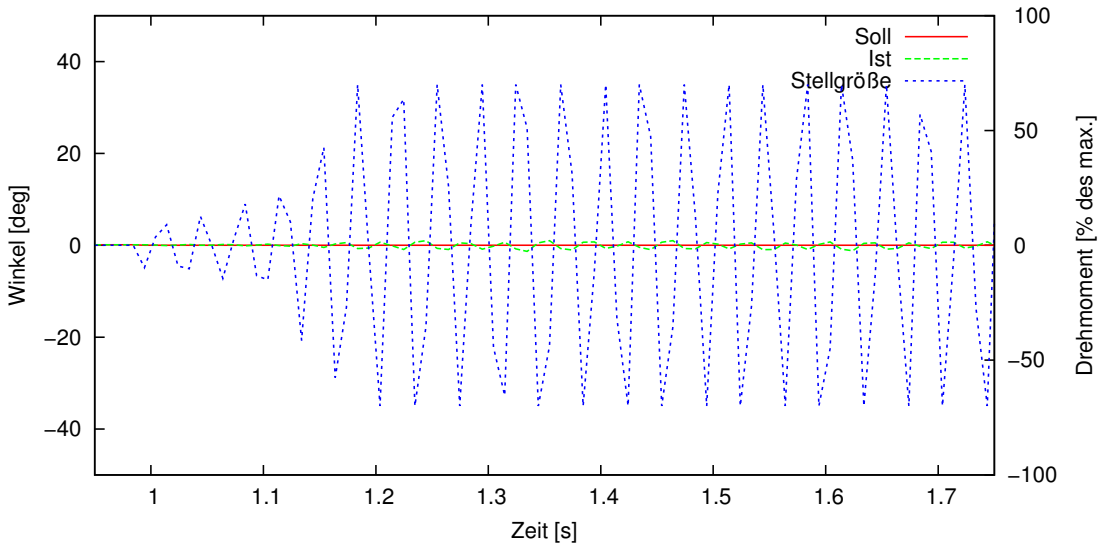


Abbildung 5.6: Schwingen des Reglers ausgelöst durch Backlash im Getriebe

Positionen unter Verwendung dieses Verfahrens.

Wie dort zu sehen ist, sind die Bewegungen sehr gleichmässig und gehen fließend ineinander über. Dieser Effekt ist der Bézier-Interpolation geschuldet. Auch zeigt sich kaum Nachschwingen beim Beschleunigen oder beim Erreichen der jeweiligen Endposition. Nachteilig ist hier jedoch, dass nur eine vergleichsweise langsame Bewegung möglich ist. Da dieses Verfahren auf die volle Kontrolle der Gelenk-Position angewiesen ist, muss sichergestellt sein, dass Soll- und Ist-Wert möglichst nicht auseinander driften. Andernfalls würde es dadurch wieder zu einem ungleichmässigen Geschwindigkeitsverlauf kommen. Ein Auseinanderdriften lässt sich effektiv vermeiden, wenn nur Geschwindigkeiten vorgegeben werden, denen die Servomotoren noch problemlos folgen können. Das führt dazu, dass die Motoren meist nicht mit ihren maximal möglichen Geschwindigkeiten und Beschleunigungen betrieben werden.

Die Langsamkeit wird deutlich, wenn man in der Abbildung die Bewegung des Tilt-Gelenks betrachtet. Es benötigt für einen Winkel von 46° eine Zeit von ca. 1 s. Mittels der direkten verwendung des PD-Reglers ließe sich dieser Winkel in weniger als 0.4 s überbrücken, wobei aber wesentlich massivere Nachschwingungen auftreten würden.

Somit ist der Einsatzzweck dieses Verfahrens vorgegeben. Wenn das Erreichen der Zielposition extrem zeitkritisch ist, kann es nicht verwendet werden. In diesem Fall ermöglicht nur die direkte Verwendung des Reglers eine ausreichend schnelle Bewegung. Ist hingegen das Erreichen des Ziels mit minimalem Schwingen oder ohne extreme Limitierung der zur Verfügung stehenden Zeit gefordert, sind die trapezförmigen Bewegungen zusammen mit der Bézier-Interpolation das optimale Mittel.

5.4 Performance

Bei dem aufgebauten Roboter handelt es sich um ein System, das Aufgaben in Echtzeit erfüllen muss. Die Kamerainformationen müssen bei jedem neuen Bild ausgewertet werden und auch die darauf aufsetztende Ansteuerung der Aktoren muss zeitnah erfolgen, um eine schnelle Reaktion zu ermöglichen. Aus diesem Grund sollte die zur Verfügung stehende Rechenzeit des Systems und ihr Verbrauch durch die Softwarekomponenten beachtet werden. Nur wenn dort noch ausreichende Reserven vorliegen, können andere Programme auf dem Rechner die zeitkritischen Komponenten nicht ausbremsen oder behindern.

Der Verbrauch der Ressourcen Rechenzeit und Speicher durch die einzelnen Komponenten ist in Tabelle 5.2 gelistet. Dort ist zu sehen, dass besonders die beiden Kreiserkenner zusammen mit dem MHT einen Großteil der CPU-Leistung beanspruchen. Die Bewegungsplanung benötigt

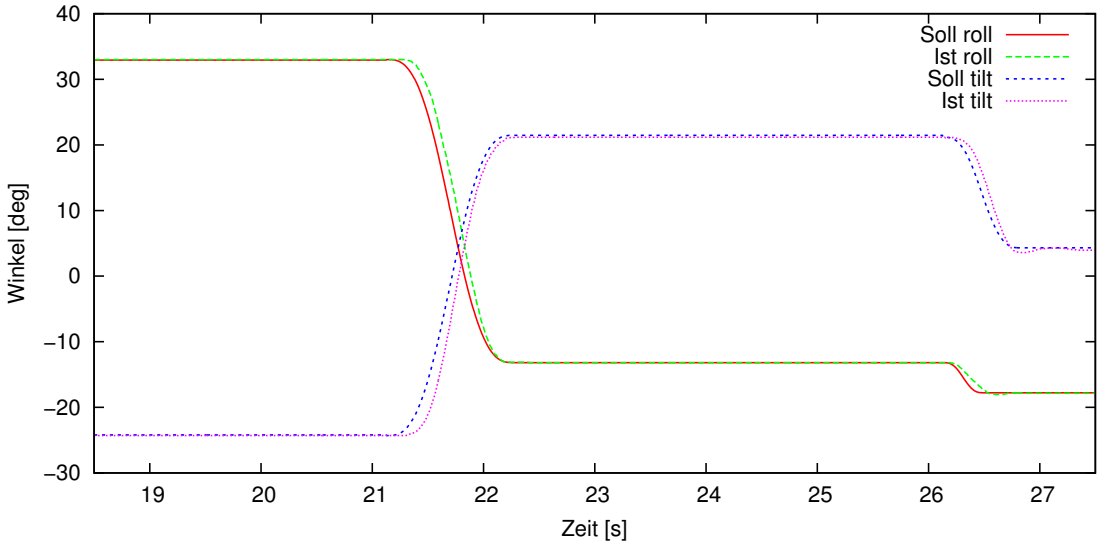


Abbildung 5.7: Verhalten der Steuerung mittels Bewegungsmustern und Bézier-Interpolation bei zufälligen Bewegungen

keine feste Rechenzeit, sondern ihr Verbrauch hängt von den erkannten Ballflugbahnen und den im Arbeitsraum des Roboters liegenden Segmenten ab. Die Bewegungssteuerung ist nicht gelistet, da ihr Verbrauch zu vernachlässigen ist. Die meiste Zeit verbringt diese Komponente mit dem ressourcenschonenden Warten auf eine Antwort der Servomotoren. Rechnet man den Verbrauch zusammen, ergibt sich, dass mehr als die Hälfte er Rechenzeit verwendet wird. Es bleibt jedoch noch mehr als ein vollständiger CPU-Kern frei, sei es für die Bewegungsplanung, das Betriebssystem oder weitere laufende Programme.

Auch der Arbeitsspeicher ist im System eine begrenzte Ressource. Wie anhand der Tabelle deutlich wird, benötigt jedoch nur die Bewegungsplanung signifikante Mengen. Dieser Verbrauch geht auf die platzraubenden Lookup-Table-Datenstrukturen zurück. Ein verbleibender Arbeitsspeicher von mehr als 3 GB zeigt, dass im System noch genügend Reserven vorhanden sind und diese Ressource nach aktuellem Stand nicht zu einem Flaschenhals werden wird.

Komponente	**Anzahl**	**CPU (4x 100 %)**	**RAM (4 GB)**
Aufnahme	2	5 %	13 MB
Kreiserkenner	2	80 %	16 MB
MHT	1	118 %	8 MB
Bewegungsplanung	1		400 MB
Gesamt		288 %	466 MB
Reserve		112 %	$>$ 3 GB

Tabelle 5.2: Benötigte Prozessor- und Arbeitsspeicher-Ressourcen auf dem Steuerrechner

Zwar ist der Rechenzeitverbrauch der Bewegungsplanung sehr variabel, es lohnt dennoch, ihn genauer zu betrachten. Diese Komponente benötigt zum einen Zeit für die aufwendige Initialisierung der Datenstrukturen und zum anderen zum Zugriff auf eben diese. Die Initialisierung setzt sich aus dem Durchführen der Simulationen und dem Einsortieren der Ergebnisse in die Datenstrukturen zusammen. Dieser Vorgang benötigt für ca. 400 000 Simulationsläufe 6 s.

Eine Anfrage in die Datenstruktur besteht aus den in Abschnitt 4.4.4ff beschriebenen Schritten zum Filtern und Bewerten der Simulationsergebnisse. Die dafür benötigte Zeit liegt auf dem Steuerrechner, während parallel alle anderen Komponenten laufen, bei 2 μs. Dieser Vorgang muss für jede Ballflugbahn und jedes Segment dieser Flugbahn, das den Arbeitsraum schneidet durchgeführt werden. Da jedes Kamerabild eine neue Flugbahnvorhersage eintreffen kann, müssen bei einem erkannten Ball maximal 50 Vorhersagen pro Sekunde bearbeitet werden. Verrechnet man dies mit

der benötigten Zeit pro Segment, so darf der zu untersuchende Teil der Flugbahn aus maximal 10 000 Segmenten bestehen. Dieser Wert ist hoch genug, sodass dort mit keinen Zeitproblemen zu rechnen ist.

5.5 Tests

Im Folgenden wird eine Reihe von Tests und Experimenten mit dem System vorgestellt. Es wird die Bewgungsgeschwindigkeit und -genauigkeit gemessen, und der Vorgang des Schläger-Ball-Kontakts analysiert. Auch das Systemverhalten im Spiel wird betrachtet. In einer quantitativen Auswertung wird anschließend untersucht, wie sich das System im Einsatz bewährt und ob ein grundlegendes Richtungsspiel möglich ist. Bei den Tests aufgefallene, grundlegende Probleme werden im letzten Abschnitt angesprochen.

Bewegungsgeschwindigkeit

Die Messung der Bewegungsgeschwindigkeit- und genauigkeit kann nicht für eine einzige Bewegung erfolgen. Vielmehr muss, von einer Ausgangsposition ausgehend, die Zeit gemessen werden, die der Schläger benötigt, um eine Zielposition zu erreichen. Dadurch lässt sich eine Aussage treffen, ob das System schnell genug einen großen Bereich abdecken kann und sich somit für den angedachten Einsatzzweck eignet.

Stellt sich die Frage, wie ein Erreichen der Zielposition definiert werden kann. Dabei spielen zwei Faktoren eine Rolle. Zum einen schwingt der Schläger beim Erreichen des Ziels nach, sodass es nicht ausreicht, das erste Überfahren dieser Position zu erfassen. Zum anderen hat der Schläger eine, wenn auch kleine, bleibende Abweichung von der Zielposition.

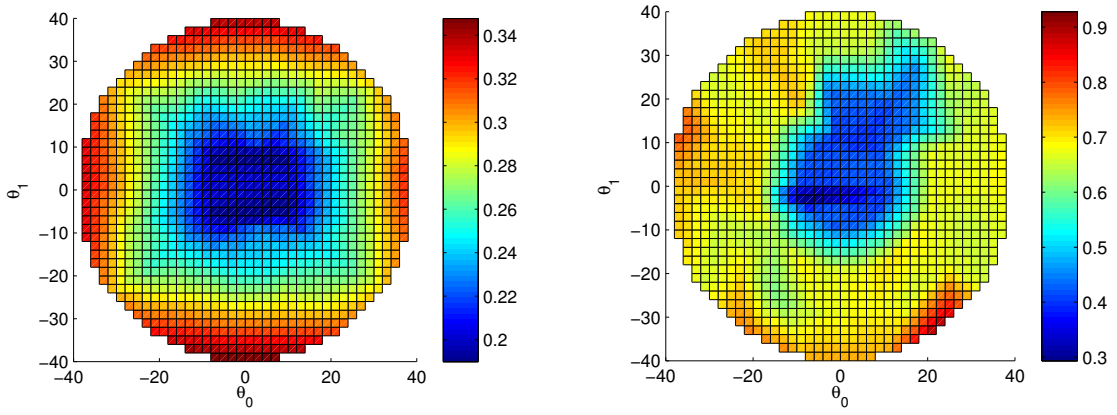
Die Zeit bis zum Erreichen der Zielposition kann nun so bestimmt werden, dass gemessen wird, wie lange der Schläger benötigt, bis er nur noch mit einer bestimmten, maximalen Amplitude um die bleibende Abweichung herum schwingt. Im konkreten Fall wurde, ausgehend von der Initialposition 0° roll und 0° tilt, die Zeit gemessen, bis der Schläger nur noch $\leq 2°$ schwingt bzw. nur noch $\leq 0.5°$ schwingt. Im ersten Fall, dargestellt in Abbildung 5.8(a), ist zu erkennen, dass das System selbst für die am weitesten entfernt liegenden Randbereiche nie länger als 0.34 s benötigt. Im zweiten Fall, zu sehen in Abbildung 5.8(b), hingegen fällt auf, dass das System sehr lange benötigt um mit weniger als 0.5° Amplitude zu schwingen. Dort sind es für fast alle Position mehr als 0.4 s, im Durchschnitt sogar um 0.7 s. Dieser hohe Wert wird zu einem großen Teil im Backlash der Servos begründet sein. Bei diesen kleinen Bewegungen ist es mit dem enormen Spiel zwischen An- und Abtriebsseite der Motoren kaum mehr möglich gegen zu regeln.

Der bereits erwähnte, bleibende Fehler liegt bei allen gemessenen Positionen unter 0.4°. Der Wert ist für dieses System sehr gering und untermauert die Behauptung, dass ein PD-Regler mit Kompensation der Schwerkraft für eine hohe Genauigkeit vollkommen ausreichend ist.

Aus diesem Experiment lässt sich Schlussfolgern, dass das System schnell genug zum Erreichen der zugeworfenen Bälle ist. Zwar liegt deren Flugzeit in ähnlichen Großenordnungen wie die maximale Zeit zum Erreichen der äußeren Positionen, jedoch werden nur wenige Bälle ein Anfahren eben dieser Positionen erfordern. Die Zeiten bis zum Erreichen einer nur noch geringen Schwingung müssen hingegen drastisch verbessert werden. Es ist allerdings zu vermuten, dass dieses Problem mit der bestehenden Hardware nicht gelöst werden kann.

Systemverhalten bei Ballkontakt

In Abbildung 5.9 sind für eine Spielbewegung die Soll- und Ist-Werte der Tilt-Achse aufgezeichnet. An Position 1748 s ist der Aufprall des Balls auf den Schläger zu beobachten. Die dadurch verursachte Auslenkung beträgt nur 1.87°. Es ist jedoch nicht eindeutig, welcher Teil des Systems die Energie des Balls aufnimmt, da die Messung der Ist-Position erst antriebsseitig im Servomotor erfolgt. Möglich wären die Trägheit des Schlägers, die Flexibilität der Stange oder auch das Getriebe des Servos. Wenn ein Großteil auf die ersten beiden Optionen entfallen würde, spräche dies für eine gute Auslegung des Systems. Dann würde der Ball nur ein geringes Drehmoment auf



(a) $\theta_{err} \leq 2°$ (b) $\theta_{err} \leq 0.5°$

Abbildung 5.8: Benötigte Zeit zur Bewegung von der Position $(0°, 0°)$ bis zur Zielposition (θ_0, θ_1) und Erreichen einer Schwingungsamplitude von höchstens θ_{err} um den bleibenden Fehler

das Servogetriebe ausüben und dieses nicht übermäßig stark belasten. Eine exaktere Messung wäre jedoch nur mit weiteren Sensoren[1] möglich, die derzeit nicht vorhanden sind.

In den meisten Fällen ist es kaum möglich, den Auftreffzeitpunkt und seine Auswirkungen so eindeutig in den Messwerten abzulesen. Da der Ball innerhalb der Zeit auftrifft, in der der Schläger nachschwingt, überlagern sich die wenigen Grad Auslenkungen und die reguläre Schwingung. Im Beispiel ist die Bewegung des Tilt-Gelenks mit nur 13° so gering und somit so schnell ausgeführt, dass es zu keiner solchen Überlagung kommt.

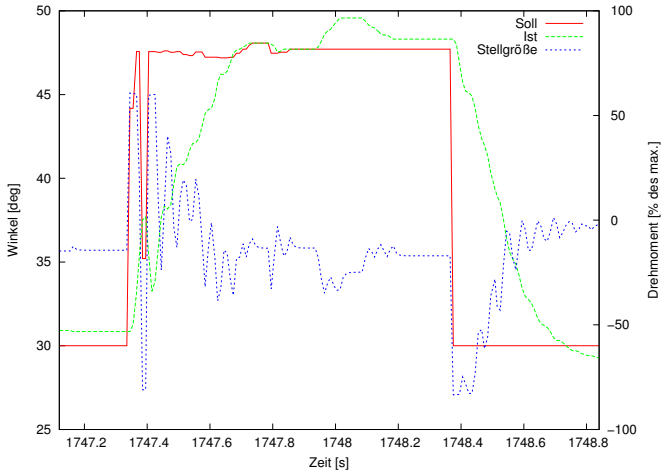


Abbildung 5.9: Auftreffen des Balls auf den Schläger bei 1748 s und Plot der Auswirkung auf den Ist-Wert der Tilt-Achse

Der Plot in Abbildung 5.10 zeigt die Soll- und Ist-Werte der Motoren für eine Folge schnell zugeworfener Bälle. Eindeutig erkennbar ist die derzeit gewählte Ausgangsposition des Schlägers. Diese ist bei 0° Roll-Winkel und um 30° nach hinten geneigt. In diese kehrt der Schläger nach jedem zugeworfenen Ball zurück. Auch lässt sich erkennen, dass die Ist- den Sollwerten nur langsam folgen. Dies liegt in der Trägheit des Systems begründet, in dessen Folge die Motoren den Schläger nicht so schnell beschleunigen können, wie es nötig wäre. Allerdings ist die Geschwindigkeit noch ausreichend zum Treffen des Balls, viel Spielraum existiert dort jedoch nicht mehr.

[1] z.B. Drehmomentsensoren abtriebsseitig

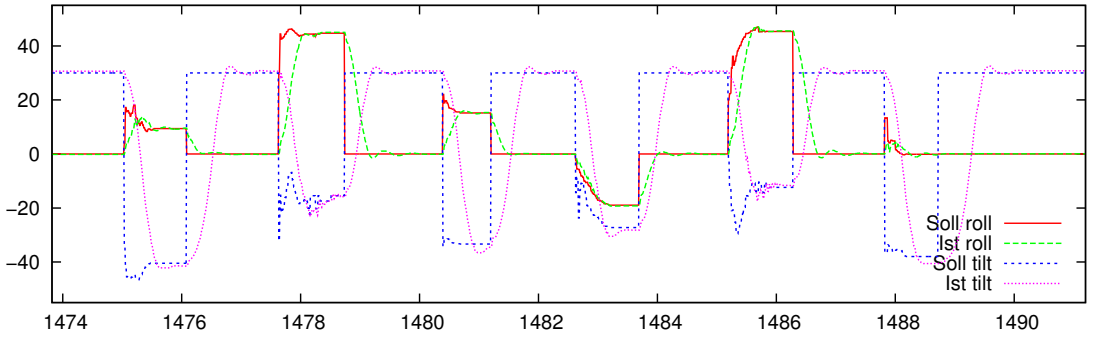


Abbildung 5.10: Soll-und Ist-Werte beider Gelenke für eine Folge zugeworfener Bälle

Wurfexperiment

Um die Performance des Systems im praktischen Einsatz zu erproben wurde ein Wurfexperiment [LBHF12] durchgeführt. Dafür wurden dem Roboter Bälle zugeworfen und gezählt, ob der Roboter diese erreicht hat oder warum nicht. Das Experiment wurde an zwei unterschiedlichen Orten vorgenommen. Zum einen in einem hohen Innenraum mit weißen Wänden und Deckenbeleuchtung. Der andere Ort war im Freien auf einer Wiese. Der Himmel war dabei größtenteils klar mit wenigen Wolken und es wehte ein leichter Wind.

Es wurde gezählt, wieviele Würfe der Roboter insgesamt getroffen hat und wieviele er von denen getroffen hat, die seinen Arbeitsraum durchflogen haben und er demnach hätte treffen können. Bei den nicht getroffenen Bällen wurde der Grund unterschieden. Es wurde gezählt, wieviele den Arbeitsraum verfehlt hatten und bei wievielen keine Ballflugbahn vom Balltracker erkannt wurde. Der dritte Grund, den Ball nicht zu treffen ist, dass der Roboter ihn schlicht verfehlt hat, meist weil er nicht schnell genug mit seinem Schläger an der zum Treffen nötigen Position angelang war.

Das Resultat dieses Experiments ist überaus positiv. So wurden 84.0% aller Bälle getroffen, die der Roboter hätte erreichen können. Mit 95.1% sind die Treffer im Arbeitsraum im Aussenbereich sogar noch höher. Wenn man die Gründe betrachtet, warum der Ball nicht getroffen wurde fallen weitere Besonderheiten auf. So wurde unter freiem Himmel in 26.8% der Fälle der Arbeitsraum verfehlt, im Innenraum dagegen nur zu 9.2%. Dort ist der dominante Grund für Fehler, dass keine Tracks generiert wurden, weil der Balltracker aus den Kamerabildern keine Ballflugbahn erstellen konnte.

Der Grund für die Probleme im Freien wird im Wind begründet sein. So ist es mit dem nur 53 g leichten Ball selbst bei schwachem Wind schwierig, auf 5 – 6 Meter Entfernung den Arbeitsraum verläßlich zu treffen. Der Balltracker hatte in dieser Umgebung erstaunlich wenige Probleme. Die Beleuchtungsinvarianz des Kreiserkenners sorgte dafür, dass die Bälle auch unter den dort herrschenden Beleuchtungsbedingungen zuverlässig erkannt wurden. Da sich im Himmel normalerweise keine kreisförmigen Objekte befinden, hatte auch der MHT keine größeren Schwierigkeiten.

Das dominante Problem im Innenraum, dass keine Tracks generiert wurden, liegt darin begründet, dass die Sichtfelder der Kameras sehr viele Störkreise enthalten haben (Abbildungssequenz 4.2). Die runden Deckenlampen wurden, zusammen mit weitere Objekten, zuverlässig als Kreise erkannt und ließen den Kreiserkenner teils die gesuchten Bälle nicht erkennen. Auch den MHT stellten die vielen Störungen vor Probleme, da durch diese der Start von neuen Ballflugbahnen oft nicht korrekt erkannt wurde.

	Würfe	ges. Treffer	AR verfehlt	AR erreicht		
				Treffer	Kein Track	Verfehlt
Inneraum	196	145 (74.0%)	9.2%	81.5%	14.6%	3.9%
Außenbereich	56	39 (69.6%)	26.8%	95.1%	4.9%	0%
Gesamt	252	184 (73.0%)	13.1%	84.0%	12.8%	3.2%

Tabelle 5.3: Ergebnis der durchgeführten Wurfexperimente (AR: Arbeitsraum) [LBHF12]

Richtungsspiel

Um zu überprüfen, ob es mit der gewählten Hardware möglich ist, die Abspielrichtung des Balls geeignet zu beinflussen, wurde ein weiteres Experiment durchgeführt. Der Test sollte herausstellen, ob ein simples, aber gezieltes Abprallen nach Links oder Rechts möglich ist. Damit dies funktioniert, muss das System die Ballflugbahn genau genug erkennen und den Schläger ausreichend schnell und genau an die benötigte Position fahren. Der exakte Abprallwinkel ist hierfür nicht relevant gewesen, weshalb auch keine aufwendige Berechnung des Auftreffpunkts zwischen Ball und Schläger durchgeführt wurde, sondern der Schläger nur relativ zur Ballflugbahn verschoben wurde. Ein Versatz nach links verursacht dabei entsprechend Abschnitt 3.1 einen Ballflug nach rechts und umgekehrt.

Das Resultat ist auch hier eindeutig, wie positiv. In über 90% der Fälle wurde der Ball in die mittels des Versatzes vorgegebene Richtung gespielt. Die wenigen Probleme traten vermehrt am Rand des Arbeitsraums bei großen Rollwinkeln auf. Hier war die Positionierung des Schlägers nicht schnell bzw. nicht präzise genug für diese Aufgabe. Dies verwundet nicht, da an diesen Positionen systembedingt ein präziser Versatz nach links oder recht wesentlich schwieriger ist als für Rollwinkel nahe der Mittelstellung.

Probleme im praktischen Einsatz

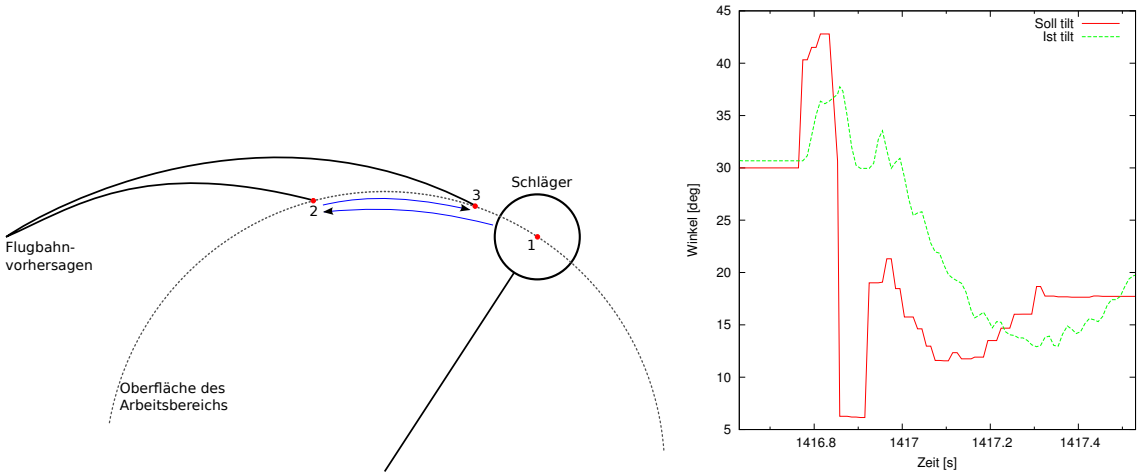
Bei der Entwicklung und Durchführung der Experimente traten Probleme mit der gewählten Hardware zu Tage. Besonders die Servomotoren erwiesen sich als fehleranfällig und den Anforderungen nicht immer gewachsen. Es zeigte sich, dass die Getriebe einen Hauptschwachpunkt darstellen. Die verbauten Zahnräder bestehen zwar aus Metall, sind aber auf Grund der geringen Baugröße und der hohen Übersetzung von 184:1 relativ filigran. Besonders das größte, abtriebsseitige Zahnrad zeigte zum Teil starke Abnutzungserscheinungen, die bis zur Unbrauchbarkeit führten.

Dabei war, wider erwarten, weniger das Zusammentreffen zwischen Schläger und Ball das Problem. Zwar werden dort Kräfte auf den Schläger ausgeübt, die zu einem Drehmoment im Gelenk führen, diese sind jedoch durch die niedrige Masse des Balls gering und werden noch weiter abgeschwächt. So verursacht das höhere Gewicht des Schlägers mit seiner Trägheit eine verminderte Beschleunigung und auch der Stab nimmt einen geringen Teil der Energie auf, indem er sich verformt.

Einer der Hauptgründe für die Belastung scheinen schnelle, ruckartige Geschwindigkeitswechsel antriebsseitig zu sein. Diese treten verstärkt dadurch auf, dass die geschätzten Ballflugbahnen zu anfang sehr ungenau sind und stark von der realen, später geschätzten Flugbahn abweichen. So kann, wenn Flugbahn und Arbeitsraum sich ungünstig schneiden, leicht ein Fall wie in Abbildung 5.11(a) auftreten. Dort befindet sich der Schläger anfänglich in seiner Ausgangsposition (1). Tritt nun eine erste, in diesem Fall zu kurze Schätzung auf, die der Roboter mit seinem Schläger an Position (2) treffen will, startet er eine Bewegung an diese Position. Wird die Ballflugbahn nach mehreren Bildern genauer geschätzt, beispielsweise so, dass sie den Arbeitsbereich des Schlägers an Position (3) schneidet, muss der Roboter die Schlägergeschwindigkeit umkehren und in die entgegen gesetzte Richtung beschleunigen. Dieser massive Wechsel, bei dem die Trägheit des Schlägers diesen weiter Richtung (2) treibt, der Motor intern jedoch in die entgegen gesetzte Richtung steuert, kann zu Schäden im Getriebe führen. Hier spielt zusätzlich das Spiel des Getriebes eine Rolle, das es erlaubt, dass die Antriebsseite für eine kurze Zeit ungehindert in die Gegenrichtung beschleunigen kann. Sind die Grenzen dieses Spiels erreicht kollidieren die Zähne der Zahnräder mit großer Wucht und verformen sich im schlimmsten Fall.

Ein solcher abrupter Geschwindigkeitswechsel ist in Abbildung 5.11(b) zu sehen. Dort wird von der Initialstellung bei 30° das Kommando gegeben, den Schläger nach 42.8° zu fahren. Nur 0.03 s später ist die neue Zielposition des Schlägers bei 6.2°.

Natürlich könnten die Motoren geschont werden, indem die ersten Schätzungen der Ballflugbahn ignoriert werden, der Schläger langsamer dort hin bewegt wird oder der Geschwindigkeitswechsel mit geringeren Beschleunigungen durchgeführt wird. Da jedoch keine großen Zeitreserven zum Treffen des Balls vorhanden sind, würde mit diesen Ansätzen die Trefferquote sinken. Erst mit schnelleren, und wenn möglich robusteren Motoren ließen sich diese Probleme beseitigen.



(a) Situationsablauf mit zwei weit auseinandergehenden Schätzungen einer Ballflugbahn

(b) Plot mit starkem Sprung im Sollwert

Abbildung 5.11: Ursache für hohe Geschwindigkeitsveränderungen, bei denen Schäden an den Motoren auftreten können

5.6 Bewegungsplanung

Die Bewegungsplanung kann bisher nur virtuell getestet werden, da die Aktoren des Roboters die an sie gestellten Genauigkeitsanforderungen nicht erfüllen. Es ist jedoch problemlos möglich, das korrekte Funktionieren dieser Komponente systematisch in Software zu überprüfen.

Bevor jegliche Tests durchgeführt werden können, müssen die Lookup-tables mit Simulationsergebnissen befüllt werden. Dafür wird der Physik-Simulator mit den in Tabelle 5.4 angegebenen Parametern initialisiert. Sie orientieren sich, wo immer möglich, an den realen Gegebenheiten. Der Balldruck wurde mit dem 1.1-fachen des normalen Luftdrucks angenommen. Dieser Wert ist eine grobe Schätzung, da das verwendete Manometer beim Messen des realen Balldrucks keinen Ausschlag zeigte. Darauf hin wurde ein Wert Nahe der Messgenauigkeit des Geräts als geeignete Annahme erachtet. Für den Reibunskoeffizienten zwischen Gummi und Styropor, der sich im Intervall $[0, 1]$ befinden muss, wurde mangels verlässlicher Daten der Mittelwert angenommen.

Parameter	**Formelzeichen**	**Wert**
Ballradius	r_b	0.0605 m
Ballmasse	m_b	0.053 kg
Balldruck	P_{b_0}	111457.5 hPa
Schlägerradius	r_s	0.125 m
Schlägermasse	m_s	0.230 kg
Reibungskoeffizient	μ	0.5
Zeitschritt	Δt	0.001 s

Tabelle 5.4: Simulationsparameter zum Testen der Bewegungsplanung

Nach dem Erstellen der Simulationsergebnisse werden diese in die Datenstrukturen eingefügt, um in ihnen gezielt suchen zu können. Zur Feststellung der korrekten Funktion wird wiederum die Simulation benötigt. Mit ihr werden zufällige, aber gültige Testfälle generiert, die beim Nachschlagen in der Datenstruktur zu einem Ergebnis innerhalb der Toleranzen führen müssen.

Der erste Test beschränkt sich auf die Verwendung der zwei Lookup-tables sowie der Bewertung und Filterung mittels Simulation (Abschnitt 4.4.5 – 4.4.7). Als Eingangsdaten dienen hier die Ballgeschwindigkeiten vor und nach dem Schlägerkontakt sowie die Normale auf dem Arbeitsraum. Bei diesem Testszenario werden für alle zufällig generierten Anfragen korrekte Ergebnisse gefunden.

Der zweite, komplexere Testfall überprüft die Abfrage mittels Ballflugbahnen. Hier wird nicht bereits durch den Normalenvektor eine Position auf dem Arbeitsraum vorgegeben, sondern dem Algorithmus die Ermittlung einer optimalen Schlägerposition überlassen. Somit bestehen die Eingangsdaten aus einer Ballflugbahn und zugehöriger -geschwindigkeit sowie einer per Simulation generierten, möglichen Ballgeschwindigkeit nach dem Schlägerkontakt.

Aufgrund von Ungenauigkeiten gelingt es der Bewegungsplanung hier nicht, für jeden Testfall ein gültiges Ergebnis zu ermitteln. Die Erfolgsquote liegt jedoch bei allen Testläufen deutlich über 90%. In den verbleibenden maximal 10% der Fälle scheitert die Suche immer daran, eine Schlägerposition zu ermitteln, bei der die Ballflugbahn möglichst exakt durch den Auftreffpunkt zwischen Schläger und Ball verläuft. Dieses Verhalten ist darin begründet, dass der Auftreffpunkt durch die hinterlegten Simulationsergebnisse vorgegeben ist und, unter Beachtung, dass der Schläger sich immer auf dem Arbeitsbereich befinden muss, nicht frei positioniert werden kann. Erschwerend kommt hinzu, dass dieser Suchvorgang durch den ellipsoiden Arbeitsbereich nur näherungsweise erfolgen kann (s. Abschnitt 4.4.7). Es ist auf Grund der Struktur des Problems anzunehmen, dass sich mit einem exakt kugelförmigen Arbeitsbereich, wie er ohne Achsenversatz entstehen würde und mit entsprechend angepassten Algorithmen, eine weitaus höhere Erfolgsquote realisieren ließe.

5.7 Vorführung

Ein System, das für den praktischen Einsatz konzipiert wurde, wird am Besten im direkten Kontakt mit Menschen in einer realistischen Umgebung erprobt. Auch wenn die bisher vorgestellten Einzeltests die prinzipielle Eignung von einzelnen Komponenten des Roboters belegen können, muss sich das Gesamtsystem als Verbund aller seiner Hard- und Software-Bestandteilen bewähren. Aus diesem Grund wurde es so oft wie möglich, so realistisch wie möglich erprobt.

Schon während der Entwicklung standen regelmässige Tests im Innen- und Außenbereich an, welche die Eignung unter den verschiedenen Umgebungsbedingungen sicherstellen sollten. Da ein realistischer, unvoreingenommener Test jedoch nicht von dem oder den Entwicklern durchführbar ist, wurde innerhalb des letzten halben Jahres, seit dem das System als Gesamtes funktionsfähig ist jede Gelegenheit genutzt, den Roboter in neuen Umgebungen und mit Außenstehenden zu erproben.

So wurde der Roboter innerhalb der Arbeitsgruppe[2] auch von nicht direkt involvierten Mitarbeitern getestet. Genauso hatten Teilnehmer anderer Projekte und Gruppen diese Möglichkeit. Auf zwei größeren Vorführungen, der Veranstaltung zu *3 Jahre Echtzeitbildverarbeitung* und dem *DFKI*[3] *Workshop Lokalisation*, stand das System allen Besuchern zum Ausprobieren zur Verfügung. Es zeigte sich, dass es bereits in der vorgestellten Ausbaustufe für reges Interesse sorgt und mit seinen Fähigkeiten begeistern konnte. So wurde der Roboter immer von einer kleineren oder größeren Gruppe Menschen umringt.

Der Nachweis, dass ein solches Konzept nicht nur technophile Menschen fasziniert, wurde mit Schülern einer 11. Klasse erbracht. Auch diese nahmen das Konzept gut an und konnten sich dafür begeistern, seine Möglichkeiten und auch Grenzen auszutesten. Es zeigte sich, dass der Roboter auch weitaus schneller zugeworfene Bälle erreichen und spielen konnte, als dies zuvor unter Laborbedingungen getestet worden war.

Doch nicht nur Gruppen hatten die Möglichkeit, den Roboter in Aktion zu erleben. Vielen Besuchern der AG wurde das System vorgeführt. Auf Grund seiner Anschaulichkeit eignet es sich sehr gut zur Vorführung von Bildverarbeitungskonzepten unter Echtzeitbedingungen. Unter dem Gesichtspunkt des Einsatzes als Event-Modul wurde es von einem Mitarbeiter einer Event-Agentur begutachtet. Dieser lieferte wertvolle Ideen und Anregungen, was in diesem Umfeld beachtet werden muss. Unter anderem berichtete er, welchen Ansprüchen an die Robustheit das Sytem genügen sollte, um den praktischen, kommerziellen Einsatz schadfrei zu überstehen.

[2] AG Echtzeitbildverarbeitung der Universität Bremen
[3] Deutsche Forschungszentrum Künstliche Intelligenz

6 Fazit und Ausblick

Das Ziel dieser Masterarbeit war es, einen Roboter-Prototypen für eine Abwandlung des Spiels *Schweinchen in der Mitte* zu entwickeln. Dieser sollte als Forschungsobjekt dienen, um Methoden, Auslegung und Konzepte für eine spätere kommerzielle Umsetzung zu evaluieren. Der Prototyp soll die bekannten Anforderungen berücksichtigen, allen voran die Sicherheit des Menschen, eine für das Spiel ausreichende Performance und geringe Kosten.

Zur Umsetzung dieser Anforderungen in ein funktionsfähiges System wurde geeignete Hardware ausgewählt und zusammengefügt. Anfängliche Schwächen der Konstruktion, vor allem in Bezug auf die Gesamtstabilität, wurden erkannt und durch konstruktive Maßnahmen behoben. Neben dem mechanischen lag zusätzlich der elektrische Aufbau im Fokus dieser Arbeit. Diese bestand zum Großteil aus der Verkabelung, teils aus der Herstellung geeigneter Kabel.

Softwareseitig wurde ein Framework zur Ansteuerung des Systems konzipiert und erstellt. Es ermöglicht eine Abstraktion von der gegebenen Hardware und eine intuitive Ansteuerung der Aktoren. Diese Bewegungssteuerung stellt den Grundstein für jede höhere Steuerung dar. Auch weitere Datenquellen, wie der Ballflugbahnen liefernde Balltracker wurden geeignet angepasst und in die Software integriet.

Darauf aufbauend wurde ein System entwickelt, mit dem ein gezieltes Spielen des Balls möglich ist. Diese Bewegungsplanung verwendet intern eine Physiksimulation zur Beschreibung des Ballverhaltens beim Schlägerkontakt. In dieser Datenbasis werden mittels zeit- und speichereffizienter Datenstrukturen und Algorithmen für einen anfliegenden Ball und eine gegebene Flugbahn nach dem Kontakt mit dem Roboter die optimale Schlägerposition und Geschwindigkeit ermittelt. Damit ist es möglich, die zur Verfügung stehenden Freiheitsgrade, in Form der Schlägergeschwindigkeit und des Auftreffpunkts des Balls auf dem Schläger zu nutzen. So erhält man ein System, dass mit nur zwei Aktoren die Flugbahn eines Balls in allen drei Raumdimensionen manipulieren kann.

Zur Analyse des Systemverhaltens sowie zur Evaluation der Eignung für den gegebenen Anwendungszweck wurde eine Vielzahl an Experimenten durchgeführt. Diese beschränkten sich zum Teil auf einzelne Komponenten, um kritische Systembestandteile besonders zu prüfen. Andere sollten die Vor- und Nachteile sowie noch vorhandene Schwächen des Gesamtsystems kenntlich machen. So stellte sich heraus, dass die Gesamtkonstuktion bereits gute Resultate liefert. Dennoch enthält sowohl die Hard- wie auch die Software noch viele Möglichkeiten zur Verbesserung.

Einen der Hauptansatzpunkte für Verbesserungen stellen die Motoren dar. So ist vor allem ihr Spiel im Getriebe, der Backlash, ein gravierendes Problem. Durch diesen Effekt ist es möglich, die Motoren ohne Kontakt zwischen An- und Abtriebsseite mehrere Grad zu bewegen. Dies führt unter anderem zu massiven Problemen im darauf aufsetzenden Regler. Er neigt zum Schwingen im Backlash-Bereich, da den Motor dort nur die Trägheit des Getriebes, aber nicht die externe Last bremst. Dieses Verhalten ist ohne weitere, externe Sensorik oder Einbußen in der Performance kaum zu beheben. Auch führt der Backlash zu einer Schwingungsneigung der externen Last. Da eine externe Bewegung im Backlash-Bereich antriebsseitig weder mess- noch korrigierbar ist, existiert auch hier mit gegebener Hardware keine Lösung.

Weitaus schwerer wiegt jedoch, dass dieses Systemverhalten zu einer Beschädigung der Servomotoren führen kann. Die kurzzeitig mögliche gegenläufige Bewegung von Last und Antrieb innerhalb des Getriebespiels kann im Servo zu einem Aufprall der Zahnräder aufeinander führen. Dadurch, zusammen mit einem duch die massiven wirkenden Kräfte ausgelösten Verziehen des Motorgehäuses, können die Zahnräder sich aufeinander drücken und ihre Zähne verformen.

Da dieses Verhalten der Servomotoren auf Grund der integrierten Bauweise nicht behebbar ist,

besteht die einzige Möglichkeit zur dauerhaften Lösung des Problems darin, die Motoren gegen ein anderes Modell auszutauschen. Dabei sollte darauf geachtet werden, dass diese ein mindestens ebenbürtiges Drehmoment besitzen und keine für Backlash anfällige Übersetzung. Optimal wäre, um Schäden zu vermeiden eine Übersetzung ohne Spiel und mit nur geringer Eigenträgheit.

Die Probleme des auf den Motoren aufsetzenden Reglers sind hauptsächlich in diesem Getriebespiel begründet. Durch dieses ist es nicht möglich den Regler auf gewohnte Weise einzustellen. Auch führt das Backlash zu der beschriebenen Schwingungsneigung, die zum Teil, jedoch nicht vollständig ausgefiltert werden kann. Ein weiteres Problem stellt die geringe Auslese- und damit Regelfrequenz dar. Sie begründet sich in der großen Kommunikationslatenz zwischen Steuerungsrechner und Servomotoren. Optimal, jedoch kaum ohne Echtzeitbetriebssystem mit geringem Jitter erreichbar, wäre eine um den Faktor 10 höhere Regelfrequenz. Auch die verwendete USB-Schnittstelle steht dieser Steigerung im Weg.

Als sehr guter Ansatz hat sich die verwendete Schwerkraftkompensation erwiesen. Sie führt dazu, dass auch ohne I-Anteil im Regler nur minimale bleibende Fehler zu beobachten sind. Mit einem genaueren Modell der Motoren, das auch die dynamischen Kräfte einbezieht, sollten sich noch bessere Ergebnisse erzielen lassen.

Es hat sich in praktischen Tests gezeigt, dass die Bewegungssteuerung und der Regler im speziellen, gut für den Einsatz im Roboter geeignet sind. Nur die nicht softwareseitig behebbaren Schwächen der Hardware stehen einem optimalen Funktionieren im Weg. Diese ließen sich durch ein Antriebssystem mit den bereits beschriebenen Eigenschaften beheben.

Die durchgeführten Experimente mit dem Roboter zeigen ein durchaus positives Ergebnis. So hat sich gezeigt, dass das System nicht nur prinzipiell schnell genug ist, sondern es zugeworfene Bälle problemlos treffen kann. In den Fällen, in denen der Roboter einen Ball verfehlt, liegt es selten an den im Rahmen dieser Arbeit entwickelten Softwarekomponenten. Oft verhindern externe Einflüsse, wie Wind, ein Treffen des Arbeitsraums. In Innenräumen hat besonders der Balltracker bestehend aus Kreiserkenner und MHT Probleme. Bei zu vielen Störobjekten sinkt seine Erkennungsrate merklich. Zusätzlich verhindert die ungenaue Vorhersage der Ballflugbahn zu Beginn eines Tracks teils die optimale Bewegung zum Auftreffpunkt.

Die Fähigkeit, einen zugeworfenen Ball gezielt in eine Richtung zu lenken, konnte nur grundlegend evaluiert werden. So war es bereits möglich, die Spielfeldhälfte, in die der Ball abprallt mit hoher Zuverlässigkeit vorzugeben. Eine genauere Kontrolle der ausgehenden Ballflugbahn würde eine exakte Positionierung des Schlägers mit einer Genauigkeit im Zentimeter-Bereich erfordern. Auch müsste die Geschwindigkeit zum Auftreffzeitpunkt exakt einstellbar sein. Dieses hohe Maß an Kontrolle ist mit den verwendeten, ungenauen Aktoren und der geringen Regelfrequenz nicht möglich. Erst eine neue Hardwarerevision würde hier weitere Tests erlauben.

Die Grundlagen für ein gezieltes Richtungsspiel wurden jedoch bereits mit der Bewegungsplanung geschaffen. Sie erlaubt es, zu einem beliebigen, anfliegenden Ball und einer gewünschten Flugbahn nach dem Auftreffen auf den Schläger eine geeignete Schlägerposition und -geschwindigkeit zu ermitteln. Eine praktische Evaluation war auch hier auf Grund der ungenauen Hardware nicht möglich, die Ergebnisse der rein in Software durchgeführten Tests zeigen jedoch eine hohe Zuverlässigkeit der Suche. So konnte zu beinahe jeder gestellten Anfrage ein valides und geeignetes Ergebnis generiert werden.

Die Faszination des Roboters und des Spielkonzepts wurde während der vielen Gelegenheiten zur Vorführung belegt. Obwohl das System nur ein rudimentäres Spielerlebnis erlaubte, ließ sich bereits Spaß und Interesse der Mitspieler erkennen. Leider zeigte sich, dass die Hardware noch nicht vollständig für den praktischen Einsatz geeignet ist. Zwar erlitt der Schläger keinerlei Schäden, aber die Servomotoren neigen zu Ausfällen. Der Analyse der Probleme nach zu urteilen ist die Ursache für die Schäden jedoch nicht in zu schnell geworfenen Bällen zu suchen, sondern in den abrupten Geschwindigkeitswechseln in Kombination mit dem Getriebespiel.

Das im Rahmen dieser Arbeit erstellte Gesamtsystem aus Hard- und Software hat sich als wertvoller Prototyp erwiesen. So konnte mit ihm eindrucksvoll aufgeführt werden, dass die grundlegenden Konzepte und Annahmen korrekt und für den praktischen Einsatz geeignet sind. Es zeigten sich jedoch die genannten Schwächen, vor allem der Hardware. Es ist dennoch positiv, dass diese Probleme so früh im Entwicklungsprozess auftraten und genau analysiert werden konnten. So ist es möglich, die Schwächen in einer zweiten Hardwarerevision anzugehen und zu vermeiden. Das entwickelte Software-Framework sollte mächtig und flexibel genug sein, um auf ihm das vollständi-

ge *Schweinchen in der Mitte*-Spiel aufbauen zu können. Natürlich sind noch Erweiterungen nötig, vor allem um den konkreten Spielablauf vorzugeben, doch diese können auf eine intensiv getestete und erprobte Basis aufsetzen.

Für die Weiterentwicklung des Systems wurden Fördergelder im Rahmen des VIP[1]-Programms [Bun11] des Bundesministeriums für Bildung und Forschung beantragt. Mit diesen soll das System innerhalb weniger Jahre bis zur Marktreife gebracht werden.

Bei der fortscheitenden Entwicklung hin zu einem marktreifen Produdukt müssen eine Reihe von Erweiterungen und Verbesserungen in das System eingebracht werden. Andere Eigenschaften sind nicht zwingend notwendig, würden jedoch Vorteile bringen. Im Folgenden sind die derzeit absehbaren zukünftigen Entwicklungsziele gelistet, es ist jedoch anzunehmen, dass im Laufe des Projekts weitere, derzeit noch unbekannte Ziele hinzu kommen werden.

Neue Hardwarerevision – Die Schwächen der aktuellen Hardware, insbesondere der Motoren, sind bei den durchgeführten Experimenten deutlich geworden. Eine neue Hardwarerevision sollte eine neu Motoreinheit enthalten, die ein stabileres, trägheits- und reibungsärmeres Getriebe mit weniger Backlash besitzen. Hiefür würde sich eine Übersetzung aus Zahnrädern und Kette anbieten.

Eine weitere praktische Ergänzung wäre die Einführung eines weiteren Pan- bzw. Dreh-Freiheitsgrades. Dieser würde jegliche Singularitäten vermeiden und schnellere Bewegungen erlauben, jedoch unter Verlust der eindeutigen inversen Kinematik. Eine solche Bauweise würde es ermöglichen, die Kameras an der Pan-Achse zu montieren. Dadurch würden sie sich mitdrehen und es dem Roboter ermöglichen, Bälle aus beliebigen Richtungen zu erkennen und anzunehmen. Bei einer Neukonstruktion sollte zusätzlich der Achsenversatz vermieden werden. Dadurch würde sich die Kinematik und viele Berechnungen massiv vereinfachen

Ein weiterer wichtiger Punkt wäre es, den Schläger und seine Befestigung stabiler auszulegen. Die Konstruktion sollte so starr und schwingungsarm wie möglich sein, um ein exaktes Positionieren zu ermöglichen.

Neue Motoren würden Anpassungen in der Bewegungssteuerung notwendig machen. Innerhalb der Hardwareabstraktion müsste das neue Kommunikationsprotokoll sowie die geänderte Umrechnung zwischen Motorpositionen und SI-Einheiten implementiert werden. Auch die Regelparameter werden eine Anpassung benötigen, und im Idealfall wird eine höhere Regelfrequenz möglich.

Evaluierung und Anpassung der Physiksimulation – Die bisherige Physiksimulation bietet nur ein grobes Abbild der Realität. Um ihre Wirklichkeitstreue zu erhöhen würde sich die Durchführung einer Versuchsreihe anbieten. Dadurch könnten die bisher geschätzten Parameter der Simulation genauer ermittelt werden. Vor allem das Reibungs- und Abprallverhalten verdient eine genauere Untersuchung. Zusätzlich könnten diese Tests heraustellen, ob weniger Vereinfachungen eine signifikant größere Genauigkeit ergeben.

Experimente zum Richtungsspiel – Mit einer neuen und genaueren Hardware könnten erste echte Tests für ein gezieltes, punktgenaues Richtungsspiel durchgeführt werden. Mittels fest vorgegebener Auftreffpunkte zwischen Ball und Schläger ließen sich die bisherigen Theorien überprüfen, um bei ihrer Korrektheit mittels der Bewegungsplanung eine automatische Berechnung des optimalen Spielverhaltens vornehmen zu lassen.

Implementierung einer Spielintelligenz – Bisher existiert noch keine Logik, die den Spielablauf für *Schweinchen in der Mitte* umsetzen kann. Für eine solche wäre es nötig, dass der Roboter die Positionen der Mit- und Gegenspieler erkennen kann. Dies könnte kosteneffizient mittels der bestehenden Kameras oder durch weitere Sensorik erfolgen. Diese Daten würden es dem System ermöglichen, ein dem Spielprinzip nach optimales Verhalten zu planen und mittels der Bewegungsplanung und -steuerung umzusetzen.

Um auf einfache Weise weitere Variationen des Spiels umzusetzen oder auch vollständig neue, mit der gegebenen Hardware mögliche, Spielkonzepte zu implementieren, würde sich eine gesonderte Beschreibungssprache anbieten. Mittels einer solchen Skriptsprache und einfacher

[1] Validierung des Innovationspotenzials wissenschaftlicher Forschung

Schnittstellen zum Ansprechen der darunter liegenden Sensoren und Aktoren ließen sich neue Verhaltensweisen um ein vielfaches einfacher sowie zeit- und ressourcenschonender umsetzen.

Kundenfreundliches Aussehen – Das bisherige Aussehen des Roboters ist größtenteils den mechanischen und technischen Gegebenheiten geschuldet. Für ein kommerzielles Produkt wäre hingegen ein kundenfreundliches Design angebrachter. Dieses sollte wenn möglich die Zielgruppe ansprechen und zu einer Teilnahme am Spiel motivieren. Auch ein optisches Feedback des Roboters wäre denkbar. So könnte dieser mitteilen, wie gut er einen zugeworfenen Ball erreichen konnte, d.h, wie gut er ihm *gefallen* hat.

Die genauere Ausarbeitung des Designs und der zusätzlichen Interaktionsmöglichkeiten mit dem Spieler sollten von einem Fachmann aus dem künstlerischen Bereich durchgeführt werden.

Einfache Bedienung – Die Bedienung des Systems muss letztendlich auch ohne Experten möglich sein. Sie sollte, nach einer Unterweisung, auch von einem Amateur durchgeführt werden können. Dafür wird ein allgemein verständliches Interface benötigt, das die Steuerung über wenige, intuitive Bedienelemente erlaubt.

Die bisher nötigen komplexen Vorgänge, wie beispielsweise das Kalibrieren mittels Ballflugbahnen müssen in einfach nachvollziehbare, schrittweise Dialoge ausgegliedert werden. Andere Aktionen, wie das Starten der einzelnen Systemkomponenten müssen vollständig vor dem Nutzer versteckt werden.

Garantierte Sicherheit – Auch wenn das System ebenso wie das Spielkonzept bereits auf größtmögliche Sicherheit für die beteiligten Menschen ausgelegt ist, wurde diese Eigenschaft bisher nicht nachgewiesen. Dafür müsste sichergestellt sein, dass auch im Fehlerfall oder wenn sich ein Mensch in den Arbeitsbereich begibt, keine Verletzungen verursacht werden können. Dafür müssen insbesondere die Auswirkungen eines Treffers des Schlägers mit voller Wucht gegen empfindliche Köperstellen, wie den Kopf einkalkuliert werden.

Formell sollte die Eignung vor einer Markteinführung von einer externen Prüfstelle begutachtet und bescheinigt werden.

Zusammenfassend lässt sich sagen, dass der im Rahmen dieser Arbeit entwickelte Roboter Piggy die an ihn gestellten Anforderungen erfüllt. So ermöglicht er in seiner Rolle als Prototyp die erfolgreiche Evaluierung der Konzepte sowie die frühzeitige Entdeckung und Behebung von Schwächen der Konstuktion. Das System kann bereits in seiner jetzigen Ausbaustufe mit hoher Zuverlässigkeit zugeworfene Bälle unter verschiedensten Umgebungsbedingungen erfolgreich treffen. Auch der wichtigsten Aufgabe eines angehenden Unterhaltungssystems, die Menschen zu faszinieren und zum Spielen zu motivieren, ist es eindeutig gewachsen.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[4at11] 4attention GmbH: *RoboKeeper Internetauftritt*, 2011. `http://www.robokeeper.com`, Gesichtet 01.09.2011.

[All11] Allied Vision Technologies: *AVT Marlin Technical Manual v2.4.0*, 2011. `http://www.alliedvisiontec.com/fileadmin/content/PDF/Products/Technical_Manual/Marlin/Marlin_TechMan_V2.4.0_en.pdf`, Gesichtet 02.09.2011.

[BBF11] Birbach, O., B. Bäuml und U. Frese: *Realtime Perception for Catching a Flying Ball with a Mobile Humanoid.* In: *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[Bel11] Belongie, Serge: *Wolfram MathWorld: Rodrigues' Rotation Formula*, 2011. `http://mathworld.wolfram.com/RodriguesRotationFormula.html`, Gesichtet 02.09.2011.

[BF09] Birbach, O. und U. Frese: *A Multiple Hypothesis Approach for a Ball Tracking System.* In: *Proceedings of the Computer Vision Systems (ICVS)*, Band 5815, Seiten 435–444, 2009.

[BFB11] Birbach, O., U. Frese und B. Bäuml: *Realtime Perception for Catching a Flying Ball with a Mobile Humanoid.* In: *Proc. IEEE Int. Conf. on Robotics and Automation*, 2011.

[Bun11] Bundesministerium für Bildung und Forschung: *Validierung des Innovationspotenzials wissenschaftlicher Forschung*, 2011. `http://www.bmbf.de/de/2391.php`, Gesichtet 11.09.2011.

[Car11] Carbon Team Germany GmbH: *Glasfaser Rohr 14,0 x 11,0 x 2000mm GFK*, 2011. `http://www.carbon-team.de/cosmoshop/pix/a/media/rp201411/technische%20Daten%20Carbon-Rohr%20technical%20data%20carbontube%20pullwinded.pdf`, Gesichtet 02.09.2011.

[CH96] Cox, I. J. und S. L. Hingorani: *An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and Its Evaluation for the Purpose of Visual Tracking.* IEEE Trans. on Pattern Analysis and Machine Intelligence, 18(2):138–150, 1996.

[FK98] Fujita, M. und H. Kitano: *Development of an Autonomous Quadruped Robot for Robot Entertainment.* Autonomous Robots, 5(1):7–18, 1998.

[Gen11] Gennadiy Rozental: *Boost Test Library*, 2011. `http://www.boost.org/doc/libs/release/libs/test/`, Gesichtet 01.09.2011.

[HR95] Heikes, R. und D. A. Randall: *Numerical integration of the shallow-water equations of a twisted icosahedral grid. Part I: basic design and results of tests.* Monthly Weather Review, 123:1862–1880, 1995.

[Hu10] Hu, Jwu-Sheng: *A ball-throwing robot with visual feedback.* 2010.

[LBHF12] Laue, T., O. Birbach, T. Hammer und U. Frese: *An Entertainment Robot for Playing Interactive Ball Games.* In: *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA 2012)*, St. Paul, MN, USA, 2012. **Eingereicht**.

[Len04] Lengyel, Eric: *Mathematics for 3D Game Programming and Computer Graphics, 2nd edition*. Cengage Learning Emea, 2004.

[Mic11] Microsoft: *Microsoft Kinect für Xbox 360*, 2011.

[Mra01] Mraček, E. Wenzel: *Simulatum Corpus. Vom künstlichen zum virtuellen Menschen*, 2001. `http://www.chess.at/geschichte/kempelen.htm`, Gesichtet 14.09.2011.

[Nin11] Nintendo: *Nintendo Wii*, 2011.

[Nok11] Nokia Corporation: *Qt – Cross-platform application and UI framework*, 2011. `http://qt.nokia.com/`, Gesichtet 01.09.2011.

[Rob11a] Robotis: *ROBOTIS e-Manual v1.05.01 EX-106+*, 2011. `www.crustcrawler.com/motors/EX106/docs/EX106.pdf`, Gesichtet 02.09.2011.

[Rob11b] Robotis: *USB2Dynamixel User's Manual v1.2*, 2011. `http://www.trossenrobotics.com/images/productdownloads/Robotis_USB2Dynamixel(english).pdf`, Gesichtet 02.09.2011.

[Shi01] Shibata, T.: *Paro Therapeutic Robot*, 2001. `http://www.parorobots.com/`, Gesichtet 01.09.2011.

[SK08] Siciliano, B. und O. Khatib (Herausgeber): *Springer Handbook of Robotics*. Springer-Verlag Berlin Heidelberg, 2008.

[Sta11] Stanford University: *JediBot*, 2011. `http://spectrum.ieee.org/automaton/robotics/diy/stanford-robots-flip-burgers-play-jedi-make-your-life-complete`, Gesichtet 01.09.2011.

[Sto05] Stoecker, Prof. Dr. Horst: *Taschenbuch der Physik, 5. Auflage*. Wissenschaftlicher Verlag Harri Deutsch, 2005.

[TBF05] Thrun, S., W. Burgard und D. Fox: *Probabilistic Robotics*. MIT Press, Cambridge, 2005.

[The99] The Robocup Federation: *RoboCup Four-Legged League*, 1999. `http://www.robocup.org/robocup-soccer/`, Gesichtet 01.09.2011.

[Vol11] Voltcraft: *Voltcraft HPS-13015 Labornetzgerät Manual*, 2011. `http://www.produktinfo.conrad.com/datenblaetter/500000-524999/512321-an-01-ml-Labornetzgeraet_HPS13015_de_en_fr_nl.pdf`, Gesichtet 02.09.2011.

[Wei11] Weisstein, E. W.: *Wolfram MathWorld: Sphere-Sphere-Intersection*, 2011. `http://mathworld.wolfram.com/Sphere-SphereIntersection.html`, Gesichtet 02.09.2011.

[Wik11] Wikipedia: *Icosahedron — Wikipedia, The Free Encyclopedia*, 2011. [Online; accessed 2-September-2011].

[Wil11] Willow Garage: *ROS – Robot Operating System*, 2011. `http://www.ros.org/`, Gesichtet 01.09.2011.

[WN03] Weigel, T. und B. Nebel: *KiRo - An Autonomous Table Soccer Player*. In: Kaminka, Gal A., Pedro Lima und Raul Rojas (Herausgeber): *RoboCup 2002: Robot Soccer World Cup VI*, Band 2752 der Reihe *LNCS*, Seiten 384–392. Springer, 2003.