Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Diplomarbeit

Visuomotorische Bewegungskoordination für mobile Roboter

vorgelegt von
Oliver Köckritz
am 25. August 2005

Studiengang Softwaretechnik

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr. Gunter Klemke

Fachbereich Elektrotechnik und Informatik
Department of Electrical Engineering and Computer Science

Oliver Köckritz

Visuomotorische Bewegungskoordination für mobile Roboter

Diplomarbeit eingereicht im Rahmen der Diplomprüfung
im Studiengang Softwaretechnik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Kai von Luck
Zweitgutachter : Prof. Dr. Gunter Klemke

Abgegeben am 25. August 2005

Oliver Köckritz

Thema der Diplomarbeit

Visuomotorische Bewegungskoordination für mobile Roboter

Stichworte

Subsumption, kameragesteuert, Roboterfußball, mobile Roboter, Verhaltensfusion, verhaltensgesteuert, Bewegungskoordination

Kurzzusammenfassung

In dieser Arbeit wird ein neues Konzept verhaltensgesteuerter Bewegungskoordination für mobile Roboter entworfen und implementiert. Die Aspekte visuell gesteuert, freibeweglich und autonom sind für dieses Konzept bestimmend. Es wird die Nutzung einer Kamera und die daraus folgende Visuomotorik in das entstandene Projekt integriert. Grundlage der Arbeit ist eine mechanische, omnidirektionale Plattform und die RCube-Plattform, ein Embedded System zur Ansteuerung von Aktoren und Sensoren, sowie der Möglichkeit einer Videobildverarbeitung. Das Konzept der Bewegungskoordination beruht auf der Annahme, dass sich verschiedene Verhaltensausprägungen nicht widersprechen müssen. Die Bewegungskoordination wird durch die Verschmelzung der einzelnen Aktorenausgaben verschiedener Verhaltensausprägungen durchgeführt. Durch diese Verschmelzungsbereiche wird der Relevanz und der Funktionsweise unterschiedlicher Verhaltensweisen Rechnung getragen.

Oliver Köckritz

Title of the paper

Visuomotorik movement-coordination for mobile robots

Keywords

Subsumption, camera-controlled, Robot Soccer, mobil Robots, behaviour asumption, behaviour-controlled, movement-coordination

Abstract

In this paper a new concept of behaviour-controlled movement-coordination for mobile robots is outlined and implemented. The aspects visually-controlled, free-movable and autonomous are decisive for this concept. The use of a camera and the subsequent visuomotorik of it are being integrated into the developed project. The basis of this work is a mechanical, omnidirectional platform and the RCube-platform, an embedded system made to steer actuators and sensors as well as having the possibility of videopicture-processing. The idea of movement-coordination is based on the assumption that different kinds of behaviour are not contradictory. The movement-coordination is realized by melting the output for the actuators created by the different behaviours. Using different melting areas accommodates different significances and functionalities of the behaviours.

Danksagung

In erster Linie möchte ich meinem Betreuer Prof. Dr. Kai von Luck danken, der mich während meines Studiums unterstützt hat und mir viele interessante Erkenntnisse ermöglicht hat.

Weiterhin bedanke ich mich bei Eske Schlüters für die Korrektur und Hendrik Simon für die Einführung in Roboterfußball.

Gedankt sei auch Sabine, Tanja, Gunter, Toby, Hase, Christina, Nina und allen Anderen.

Hinweis zu Markennamen

Alle in dieser Arbeit verwendeten Firmen- und/oder Produktnamen sind Warenzeichen und/oder eingetragene Warenzeichen ihrer jeweiligen Hersteller in ihren Märkten und/oder Ländern.

Inhaltsverzeichnis

1 Einleitung

Roboter spielen heutzutage in unserem Leben eine nicht wegzudenkende Rolle. Sie laufen zwar nicht wie in einigen Science Fiktion Geschichten wie Menschen durch die Straßen, wohl aber sind mit jedem unserer Blicke Produkte ihrer Arbeit sichtbar. In der Produktion, wie z.B. in der Automobilindustrie, sind Roboter so stark beteiligt, dass mehr Arbeitnehmer damit beschäftigt sind die Maschinen für die Produktion in Gang zu halten, als damit, selber Autos zu produzieren. In der Unterhaltungsindustrie ist ihre Präsenz ebenfalls spürbar. Sei es als Roboter-Gladiator im Fernsehen oder als Spielzeug für Kinder in Form eines Furby[1], Robosapiens[2] oder Aibos[3]. Im Haushalt helfen heute Roboter wie der Roomba beim Staubsaugen oder der Automower beim Rasenmähen. 1997 schaffte es sogar der Roboter Sojourner auf den Mars, machte dort Fotos und führte chemische Analysen durch.

Spielzeugroboter oder Haushaltsroboter sollen mit der realen Welt, in der wir leben, interagieren. Wenn mehr als ein Roboter im Haushalt hilft, müssen eventuell auch diese miteinander kooperieren. Ein Roboter selber besteht häufig auch aus verschiedenen Subsystemen, die miteinander interagieren und als Gesamtsystem das selbe Ziel verfolgen. Bei Aufgaben, die sich in viele Teilaufgaben separieren lassen, können verschiedene Roboter diese übernehmen, um die Gesamtaufgabe

---

[1] Ein Plüschtier mit Augenaufschlag, das über Künstliche Intelligenz verfügt und seinen Gemütszustand verbal und über Gestik mitteilen kann. Es wurden über 40 Millionen Exemplare verkauft [Ichbiah05].

[2] Ein humanoider Roboter mit 6 Freiheitsgraden, der laufen und greifen kann. Er ist 35 cm groß und kann programmiert werden.

[3] Ein Spielzeughund für Erwachsene, der programmiert werden kann, läuft und bellt. Dabei orientiert er sich mit einer Kamera, die im Kopf eingebaut ist. Der Aibo spielt in der Four-Legged-League des RoboCups Fußball und kostet über 1000 Euro.

lösen zu können. Roboterfußball z.B. ist eine Welt zur Entwicklung von autonomen, sehenden und kooperierenden Systemen im Bereich der Künstlichen Intelligenz.

1.1 Zielsetzung dieser Diplomarbeit

Diese Diplomarbeit baut auf die im Roboterlabor der HAW Hamburg vorhandenen materiellen und immateriellen Voraussetzungen auf und hat zum Ziel, diese zu erweitern. Es wird die von Michael Ziener[4] neu entwickelte Roboterplattform mit omnidirektionalem Antrieb mechanisch und sensorisch für die in dieser Diplomarbeit gestellten Fragen und Lösungen in Bezug auf Roboterfußball erweitert, damit eine solide, leicht erweiterbare Experimentierplattform zur Verfügung steht. Dazu soll eine Bewegungskontrollstruktur etabliert werden, die es ermöglicht, Sensor- und Verhaltensbausteine in das vorhandene System einfach zu integrieren.

Die Rahmenbedingungen dieser Diplomarbeit sind Roboterfußball und das Roboterlabor an der HAW Hamburg. Die Aspekte autonom, visuell gesteuert und freibeweglich sind bei der Lösung bestimmend. Da bei einem Fußballspiel dynamische, freibewegliche Bewegungsprofile sowohl spielerische Vorteile bieten, als auch dem Vorbild Mensch entsprechen, wird ein omnidirektionaler Antrieb verwendet.

Ein weiterer Eckpunkt dieser Arbeit ist es, eine Kamera als einen variablen und vielvermögenden Sensor, der dem menschlichen Auge ähnelt, für den visuellen Input in das vorhandene System zu integrieren. Dabei ist es notwendig, diese Kamera lokal zu integrieren, damit die Autonomie des Roboters gewährleistet ist.

Zusammengefasst soll ein Robotersystem an der HAW Hamburg etabliert werden, welches visuelle Reize in Bewegungen umsetzt und im Sinne von sogenannten Verhaltensweisen leicht erweitert werden kann, damit andere Studenten diesen

[4] Titel und Quelle der Diplomarbeit von Michael Ziener findet sich unter [Ziener05].

Roboter als Grundlage für Experimente nutzen oder gegebenenfalls grundsätzlich erweitern können.

1.2 Vorgehensweise

Ausgehend vom gegenwärtig gespielten Roboterfußball innerhalb des RoboCup-Projekts und an der HAW Hamburg, welche ich in der Einleitung beschreibe, beginne ich in Kapitel 2 mit den für diese Diplomarbeit relevanten Grundlagen eines reaktiven Robotersystems. So gehe ich in Kapitel 2.1 zunächst auf die Grundlagen der Computer-Vison in Bezug auf das Konzept von Lars Brandt[5] ein. Darüber hinaus gebe ich dort einen Überblick über verschiedene Kamerasysteme für Roboter. In Kapitel 2.2 erläutere ich die Anwendung von digitalen Regelkreisen, die in dieser Arbeit, sowie häufig in der Robotertechnik im allgemeinen, eingesetzt werden. Kapitel 2.3 schließt den Grundlagenabschnitt ab und beschreibt die Möglichkeiten und Funktionsweise omnidirektionaler Antriebsarten in Bezug auf die hier verwendete mechanische Plattform. Diskussionsgrundlagen zum Verständnis der im folgenden entwickelten Bewegungskoordination liefert Kapitel 3, welches die Programmierung von Robotern beschreibt. Dieses Kapitel teilt sich in die drei Teile Parameterprogrammierung, Strukturierung von Roboterprogrammierung und Strukturierung von Roboterprogrammierung in Bezugnahme auf ressourcenarme Roboter.

In den beiden darauffolgenden Kapiteln 4 und 5 werden die materiellen und immateriellen Voraussetzungen für diese Diplomarbeit dargelegt. Kapitel 4 geht auf die hier relevante im Roboterlabor vorhandene Hardware ein und bewertet diese. Unter dem selben Fokus illustriert Kapitel 5 die beiden parallel laufenden Diplomprojekte von Lars Brandt und Michael Ziener, deren praktische Ergebnisse in dieser Diplomarbeit Anwendung finden und dadurch verbunden werden.

[5] Titel und Quelle der Diplomarbeit von Lars Brandt findet sich unter [Brandt05].

Die Konzepte der vorgenommenen Erweiterungen werden in Kapitel 6 beschrieben und deren Vor- und Nachteile abgewogen. Um auf einen Pool von koordinierbaren Verhaltensweisen zurück greifen zu können, werden neben der Bewegungskoordination auch Aspekte wie Schussmechanismus, visuelle Objektpositionsbestimmung, haptische Eigenschaften und manuelle Manipulation des Roboters bei der Erweiterung betrachtet. Die Implementation dieser Konzepte wird in Kapitel 7 beschrieben und die Bewertung der Erweiterungen sowie eine Beschreibung weiterer Möglichkeiten folgt letztlich in Kapitel 8.

1.3 Roboterfußball

So wie es beim menschlichen Fußball nicht nur darum geht, das Runde ins Eckige zu bekommen, sondern auch um finanzielle, unterhaltungstechnische und soziale Aspekte, so geht es beim Roboterfußball auch um den Forschungs- und Bildungsaspekt in der Robotik und damit im Bereich der Künstlichen Intelligenz (KI).

Bis 1997 gab es in der KI-Forschung mit ähnlichen Motiven das Ziel, den Menschen im Schach mit einem Computer zu schlagen. Nachdem 1997 der amtierende Schachweltmeister Garri Kasparow von Deep Blue, einem Schachcomputer, geschlagen wurde, war klar, dass zukünftig Schach nicht mehr einer der großen Schwerpunkte der KI-Forschung sein wird. Da sich bereits in den 80er Jahren abzeichnete, dass die Schachproblematik wohl bald gelöst sein würde, begannen Forscher aus dem Bereich Künstliche Intelligenz nach neuen Herausforderungen zu suchen.

Einen Wettbewerb in Form von Roboterfußball erdachten sich 1993 Manuela Veloso und Peter Stone aus den USA, Alan Mackworth aus Kanada und Minoru Asada, Yasuo Kuniyoshi, Hiroaki Kitano und Itsuki Noda aus Japan. Den ersten internationalen RoboCup gab es schließlich im August 1997 in Nagoya (Japan), drei Monate nach dem Sieg von Deep Blue. Seitdem wird jedes Jahr ein internationaler

RoboCup von der „RoboCup-Federation“ veranstaltet. Der RoboCup ist heute einer der größten wissenschaftlichen Wettbewerbe der Welt.

Ein weiterer Grund für den Wechsel des Forschungsschwerpunkts besteht darin, dass Schach eine relativ einfach berechenbare künstliche Welt ist, deren Komplexität endlich ist. Dagegen findet Roboterfußball unter anderem in der realen Welt statt, die nicht diskret berechenbar und deren Komplexität wohl auch nicht endlich ist. Roboterfußball schafft damit eine ganz neue Bandbreite von Problemen und daraus resultierender Lösungsansätze. Eines der Problemschwerpunkte ist die Interaktion mit der realen Welt, sowohl das Erkennen dieser als auch deren Manipulation: Wie erkenne ich den Ball, wie das Tor und die Gegenspieler, wie stelle ich intern (also im Computer) die reale Welt dar, wie setze ich die gewünschte Handlungsweise auf dem realen Spielfeld um, wie kann ich die Verhaltensweisen von mehreren Robotern zu einem Ziel verbinden und wie kann ich verschiedene Verhaltensweisen innerhalb eines einzigen Roboters miteinander verbinden? Diese Fragen werden wohl aufgrund der Komplexität nie in Gänze beantwortet werden können, was die Versuche, Antworten und Lösungen zu finden, zu einer noch größeren Herausforderung werden lässt.

Das Interesse an Roboterfußball in Forschung und Ausbildung im allgemeinen wie auch in dieser Diplomarbeit liegt einerseits in seinem spielerischen Charakter, andererseits auch darin, dass Lösungen im Roboterfußball auch Lösungsansätze von Problemen in der realen Welt bieten können. Leider ist nicht auszuschließen, dass auch militärische oder repressive Organe der Gesellschaft von Forschungsergebnissen des Roboterfußballs profitieren. Da jedoch im Forschungsfeld von Roboterfußball mit relativ begrenzten finanziellen Mitteln gearbeitet wird und der Forschungsaustausch international und nicht konkurrent angelegt ist, kann man davon ausgehen, dass keine militärischen Ziele mit dieser Technik unterstützt werden, sondern tendenziell eher soziale.

1.4 RoboCup-Projekt

Das Ziel des RoboCup-Projekts ist es, 2050 nach FIFA-Regeln gegen den amtierenden Fußballweltmeister antreten und gewinnen zu können. Dazu wurden verschiedene RoboCup-Ligen mit unterschiedlichen Problemschwerpunkten und den damit verbundenen parallel laufenden Entwicklungsstrategien gegründet. Heutzutage wird in acht Ligen gespielt: Simulation-League, Small-Size-League, Middle-Size-League, Humanoid-League, Rescue-League, Rescue-Simulations-League, Junior-League und Four-Legged-Robot-League.

Die jeweiligen Ligen fokussieren verschiedene Aspekte, damit es eine parallele Forschungsentwicklung geben kann, deren Ergebnisse sich über die Jahre hinweg verbinden können, um sie schließlich in Form von Synergieeffekten nutzen zu können. Aus vielen Ligen werden sukzessive wenige, bis nur noch eine Liga existiert, die dann 2050 gegen den „menschlichen“ Weltmeister antreten kann.

Ein weiterer aber nicht unbedeutender Grund für das Aufteilen von Forschungsschwerpunkten auf die Ligen ist die schon erwähnte Komplexität sowie das eingeschränkte Budget der Forschungsinstitutionen. So erlaubt diese Art der „Arbeitsteilung“ eine übersichtlichere, themenspezifische und damit handhabbarere Herangehensweise für Forschende, Auszubildende und unabhängig Organisierte.

1.5 Vier RoboCup-Fußball-Ligen

Als Referenz- und Anhaltspunkt zu dieser Diplomarbeit stelle ich im folgenden Teilaspekte der vier RoboCup-Ligen Middle-Size-League, Four-Legged-League, Humanoid-League und Junior-Soccer-League vor. In den ersten drei Ligen sollen die Roboter vollkommen autonom spielen: Alle Roboter haben eigene Sensoren, ein lokales Kamerasystem und Rechenleistung bei sich, können sich omnidirektional bewegen und dürfen über Funk mit den anderen Robotern und einem externen Rechner kommunizieren. Art und Umfang der Kommunikation sind nicht definiert,

ansonsten kommen diese Rahmenbedingungen einem Fußballspiel in der menschlichen Welt sehr nahe. Die letzte der vier vorgestellten Ligen, die Junior-Soccer-League, ist die Referenz für den zur Zeit an der HAW Hamburg gespielten Roboterfußball.

In der Middle-Size-League hat jeder der bis zu vier Roboter eines Teams eine maximale Grundfläche von 2000 cm^2, und es wird auf einem Spielfeld von mindestens 5x10 Metern gespielt. Am Anfang waren die Roboter schwer, behäbig und sehr teuer. Mittlerweile wiegen Roboter, wie die der FU-Fighter aus Berlin, nur noch 10 kg und sind mit einem omnidirektionalen Antrieb ausgestattet. Zur Zeit setzt sich der Trend durch, den Differentialantrieb durch den omnidirektionalen Antrieb zu ersetzen. Auch die Gewinner der German Open 2005, die Brainstormers-Tribots aus Osnabrück [Tribots05], sind mit einem omnidirektionalen Antrieb ausgestattet. Jeder Spieler muss sich anhand von farbigen Markierungen und Toren zurechtfinden.

Die Four-Legged-Robot-League enstand 1999, nachdem Sony den AIBO, einen lenkbaren Roboter-Spielzeughund mit schwenkbarer Kamera und Funk, der programmiert werden kann, herausbrachte. Die Plattform bei dieser Liga ist für alle beteiligten Teams gleich. Somit ist nicht die Geschicklichkeit in der Konstruktion eines Roboters ausschlaggebend für Sieg oder Niederlage, sondern allein das programmierte Verhalten des AIBOs. Gespielt wird mit bis zu vier AIBOs pro Team auf einem mit Farben markiertem 3x5 Meter großem Fußballfeld. Da Sony die Roboter dieser Liga herstellt und vertreibt, ist die Entwicklung in dieser Liga somit auch von Sony abhängig.

Die Humanoid-Robot-League, welche seit 2002 existiert, kommt in Form und Bewegung der Roboter den menschlichen Fußballspielern am nächsten. Es gibt verschiedene Größenklassen und vorgeschriebene Proportionen müssen eingehalten werden. Zur Zeit wird in dieser Liga eins gegen eins oder zwei gegen zwei gespielt. Die Humanoid-League ist mittlerweile die Liga mit der größten Anziehungskraft. Sie wird letztlich alle anderen Fußball-Ligen vereinen und gegen das menschliche Pendant antreten.

Die Junior-Soccer-League dient nicht primär der Forschung, sondern der Ausbildung. Hier werden autonome Roboter aus handelsüblichen Bausätzen, wie z.B. Lego-Minestorm, von Schülerinnen und Schülern zusammengebaut. In den lokalen Meisterschaften treten diese eins gegen eins oder zwei gegen zwei gegeneinander an.

Nationalkomitees gibt es seit 1997 in Japan, seit 2001 in Deutschland, seit 2002 in Australien und seit 2003 in den USA. Die Junior-Soccer-League wird auf einem Feld von 87x119 cm bei eins gegen eins und 122x183 cm bei zwei gegen zwei gespielt. Der Boden ist mit einer matten Grauskala bedruckt. Das gesamte Spielfeld ist mit einer 14 cm hohen mattschwarzen Wand umgeben. Die Tore sind mattweiß und jeweils 29 cm oder 45 cm breit. Die Roboter dürfen einen Durchmesser von 18 cm oder 22 cm haben. Der Ball hat einen Durchmesser von 8 cm und sendet ein Infrarotlicht aus.

1.6 Roboterfußball an der HAW Hamburg

Roboterfußball im Roboterlabor der HAW Hamburg wird ähnlich wie in der RoboCup-Junior-Soccer-League gespielt. Allerdings ist das Environment um Infrarotbarken in den Toren und einen Filz am Boden erweitert worden, da das Setting der Junior-League nicht ausreichend war, um gewünschte Verhaltensmuster bei den Spielern hervorrufen zu können.

Abbildung 1.1: Roboter-Fußball-Contest im Sommer 2004 an der HAW Hamburg.

In der HAW Hamburg wird auf Basis von Lego-Mindstorm auf einem Fußballfeld gespielt, welches dem Junior-League Feld mit eins gegen eins Spielern gleicht, gespielt. Jedes Tor hat ein Leuchtfeuer und kann dadurch als Freund- bzw. Feind-Tor identifiziert werden. Der Boden besteht aus einfarbigem Filz, damit der Ball nicht so weit rollt und zufällige Eigentore weitgehend vermieden werden können. Die Leuchtfeuer ermöglichen eine genauere Zielanvisierung sowie die Entscheidung aus dem Stand, welches das richtige Tor ist. Der Ball sendet Infrarotlicht aus und kann damit durch Infrarotsensoren erkannt und angepeilt werden.

Um das Niveau, auf dem an der HAW Hamburg Roboterfußball gespielt wird, anzuheben, werden in Studien- und Diplomarbeiten neue Plattformen und Software entwickelt. Diese sollen bei Wettkampftauglichkeit zukünftig in Roboterfußballturnieren zum Einsatz kommen.

2 Sehen Erkennen Handeln

In diesem Kapitel beschreibe ich Grundlagen eines reaktiven Roboters, der Fundament für ein System sein soll, welches mit umfangreicheren Analysen von Situationen in der Lage ist, komplexere Aufgaben zu lösen. Das abstrakte Programm „while(alive) act(reason(sense));“ ist die Grundlage für viele Roboterprogramme. In dieser Diplomarbeit steht unter anderem die visuelle Erhebung der Umgebungsgegebenheiten für die Sensorik. Die Interpretation von Sensordaten erfolgt am unmittelbarsten mit Hilfe von Reglern und die Ergebnisse werden mit dem omnidirektionalen Antrieb in die Tat umgesetzt.

2.1 Computer Vision

Bilder im Sinne einer Computer Vision sind zweidimensionale Projektionen von dreidimensionalen Geschehnissen. Zwei essentielle Teile der Computer Vision sind die Kamera, welche die zu interpretierende Szene digitalisiert und die Algorithmen, welche die digitalisierte Information interpretierbar machen. Wie ich in Kapitel 2.1.4 beschreiben werde, kann auch die Kamera selber schon einen Filter auf die Szene legen, so dass den Algorithmen ein schon bewertetes Bild zur Verfügung steht.

Bei der Computer Vision müssen wir zwischen der Interpretation von stehenden und bewegten Bildern unterscheiden. Die zwei Dimensionen des stehenden Bildes erweitern sich bei bewegten Bildern um die Dimension der Zeit. Bei stehenden Bildern geht es darum, das Geschehen anhand von Symbolen und Strukturen innerhalb des Bildes zu erkennen. Bei bewegten Bildern können die zuvor genannten Kriterien ebenfalls nützlich sein, außerdem können auch Veränderungen der Bilder in der Zeit Grundlage der Umgebungsinterpretation sein. Je komplexer

die Analyse des Einzelbildes, desto mehr Zeit wird dafür beansprucht. Dadurch bedingt ändert sich die Schärfe bzw. Auflösung der zeitlichen Dimension.

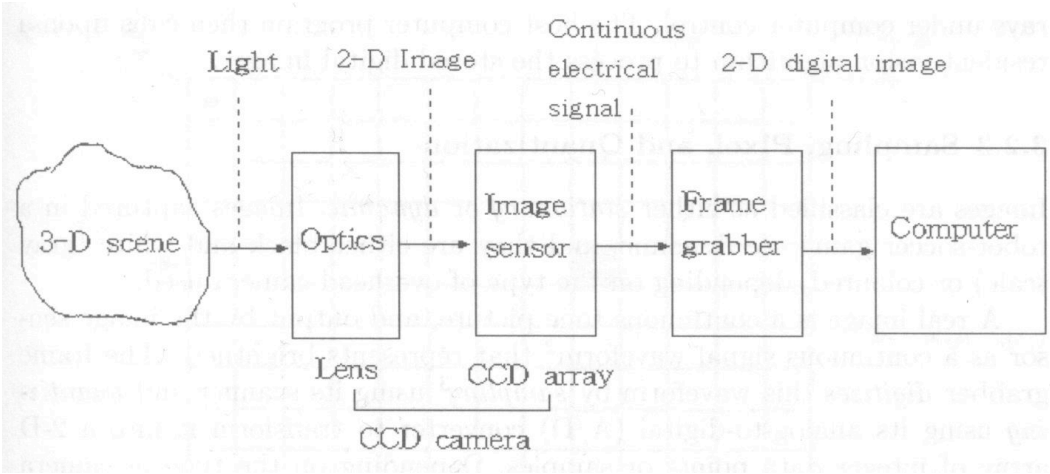

Abbildung 2.1: Grundarchitektur von einem Computer Vision System, [JongHwan04, S. 73].

2.1.1 Grundlagen Sehen

Bilder und ihre Bedeutung sind, da von Menschen gesehen, auch durch Menschen definiert. Das Auge ist das Organ des Menschen, welches Bilder oder auch seine Umgebung visuell wahrnehmen kann. Farbe ist ein über das Auge erzeugter Sinneseindruck. Die Farbwahrnehmung erfolgt aufgrund eines im Auge erzeugten Farbreizes, der wiederum durch Lichtstrahlung bzw. elektromagnetische Wellen mit einer bestimmten spektralen Zusammensetzung ausgelöst wird. Das menschliche Auge hat weit über 100 Millionen Stäbchen zur Wahrnehmung von hell und dunkel und über sechs Millionen Zapfen für die Farbwahrnehmung [vgl. Burkhard03]. Im Auge gibt es drei verschiedene Arten von Zapfen mit unterschiedlicher spektraler Empfindlichkeit, nämlich für den kurzwelligen blauen, den mittleren grünen und den langwelligen roten sichtbaren Spektralbereich. Die wahrgenommenen Farbwerte ergeben zusammen den Farbeindruck, so wie es im RGB-Modell beschrieben wird.

Der Frequenzbereich der vom menschlichen Auge wahrnehmbaren elektromagnetischen Wellen beschränkt sich wie in Abbildung 2.2 gezeigt auf ein kleines Band innerhalb der gesamten Bandbreite elektromagnetischer Wellen. Wir können nur diesen Bereich mit unseren Sinnen wahrnehmen und damit auch nur einen kleinen Teil der Informationen über unsere Umwelt. In Superhelden-Comics oder in Science Fiktion Geschichten können die Helden auch andere Spektren der elektromagnetischen Strahlung wahrnehmen und damit das in der Geschichte gestellte Rätsel lösen. In diesem Zusammenhang gibt es wenige Menschen, die sich über ein vermindertes Wahrnehmungsvermögen beschweren. Der Grund darin liegt wohl in der Tatsache, dass sich unsere Aufgaben und Lösungen meistens dadurch bedingen, dass die wahrgenommene Welt die Welt ist, die für uns real existiert und uns diese Probleme stellt. So stellt das Leben eben die Probleme, die subjektiv eine Bedeutung haben.

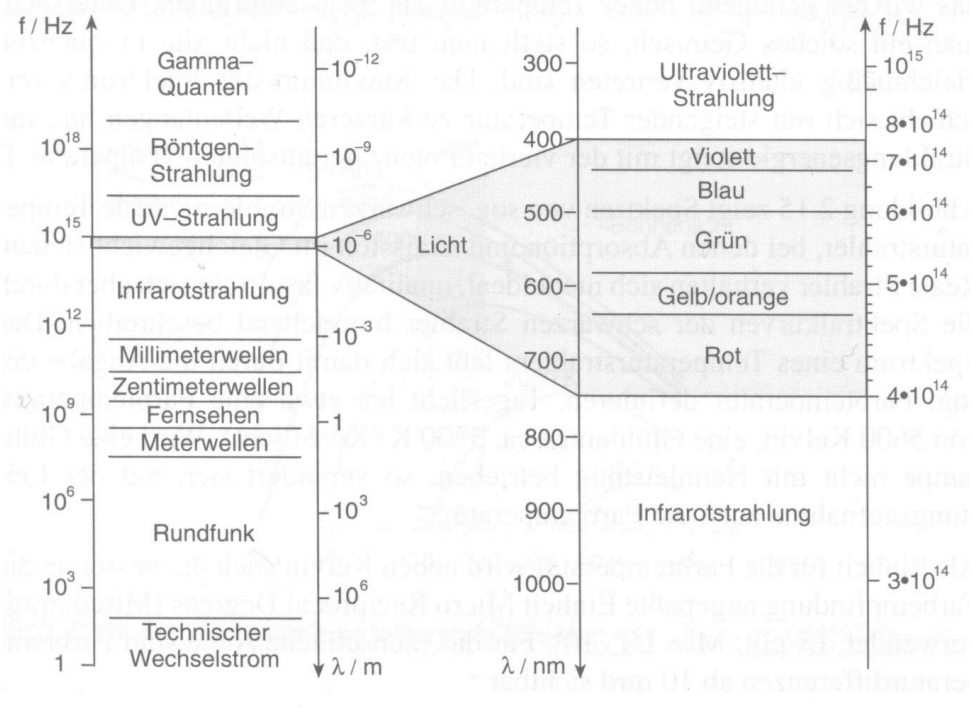

Abbildung 2.2: Spektrum elektromagnetischer Wellen [Schmidt00].

2.1.2 Digitale Bilder

Bilder, die ein Robotersystem mit einer einfachen Kamera sieht, werden intern als zweidimensionale Matrix dargestellt und bestehen ähnlich wie ein Knüpfteppich aus vielen einzelnen Punkten, die Pixel[6] genannt werden. Ein Fernsehbild in Deutschland (PAL) hat ungefähr 720x576 sichtbare Pixel.

Jedes Pixel beinhaltet die jeweiligen Farbwerte für einen Punkt. Bei einem Schwarz-Weiß Bild wäre das pro Pixel ein Bit, nämlich 0 für Schwarz und 1 für Weiß. Bei einem Graustufenbild mit 256 Graustufen würde jedes Pixel ein Byte beinhalten. Bei einem RGB-Farbbild mit einer Farbtiefe von 24 Bit kann ein Pixel 2^{24}, also über 16 Millionen Farben annehmen. Jedes Pixel beinhaltet je nach Farbmodell die entsprechende Information für die eigene „Farbe“.

2.1.3 Farbmodelle

Das bekannteste und am meisten genutzte Farbmodell ist das RGB-Modell, da dieses der Funktionsweise unseres Auges am nächsten kommt, ein Fernsehbild aus RGB-Werten zusammen gesetzt ist und auch Computermonitore nach diesem Prinzip funktionieren.

Das CMY-Modell ist für alle Layouter und Drucker von Druckerzeugnissen interessant, da dieses Modell auf Reflexion basiert. Die „Farben“ werden also ohne eigene Leuchtkraft erzeugt.

Das HSV-Modell ähnelt der menschlichen Art der Bedeutungszumessung von „Farben“. Hier werden die Helligkeit und der Farbton direkt zugänglich gemacht.

Das YUV-Modell ist wie das RGB-Modell ebenfalls ein verbreitetes System für die Videodatenübertragung.

[6] „Dieses Kunstwort steht für „Picture Element“ und kennzeichnet die kleinste, sichtbare Einheit eines digitalen Bildes.“[Balzer02].

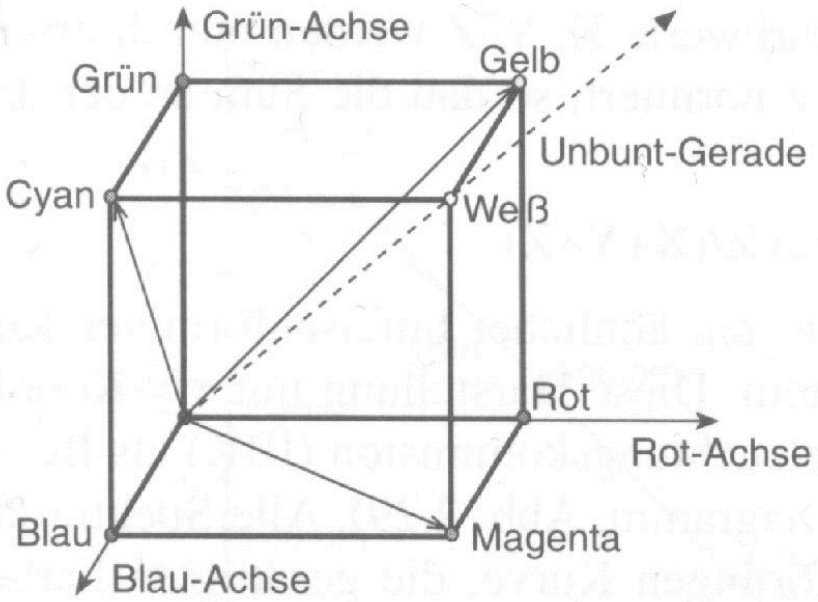

Abbildung 2.3: Dreidimensionaler Farbraum für RGB und CMY aus [Schmidt00].

RGB-Farbmodell

Im RGB-Farbraum wird die sogenannte Farbvalenz durch die Farbwerte Rot, Grün und Blau dargestellt. Die Grundfarben Rot, Grün und Blau der additiven Farbmischung werden Primärvalenzen genannt. Die Darstellung der Primärvalenzen als Vektoren ergibt einen Würfel, wie in Abbildung 2.3 gezeigt. Der Endpunkt eines Vektors innerhalb dieses Farbraumes stellt eine Farbvalenz dar. Eine Farbvalenz ist also eine dreidimensionale Größe, die einen Farbreiz kennzeichnet. Die Farbvalenz kann als additive Mischung der drei Farbwerte RGB ausgegeben werden.

Rot	Grün	Blau	Entstehende „Farbe“
0	0	0	Schwarz
255	0	0	Rot, gesättigt
255	60	60	Rot, geringer gesättigt
255	255	0	Gelb, gesättigt
255	255	255	Weiß

Abbildung 2.4: Farbbeispiele RGB.

Aus einer additiven Mischung mit R=G=B=255[7] ergibt sich Weiß, was sich dadurch erklärt, dass bei dieser Mischung die Lichtquelle den Farbwerten entspricht.

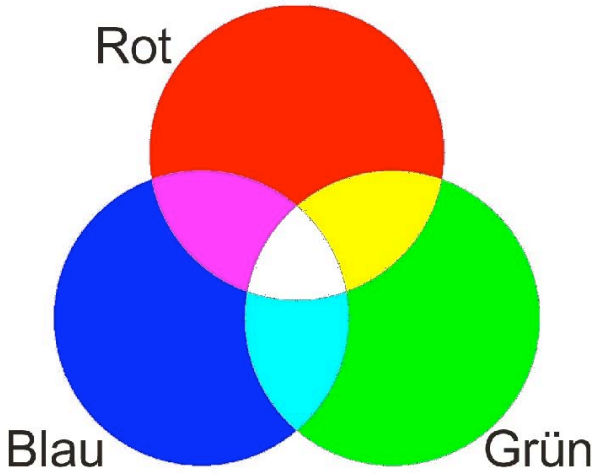

Abbildung 2.5: Additive Mischung [Schmidt00].

[7] Normalerweise wird von R=G=B=1 gesprochen, wobei in unserem Fall 1 für 255 steht. Um Irrtümer zu vermeiden, habe ich hier den konkreten praktischen Wert und nicht den theoretischen für eine hundertprozentige Ausprägung gewählt.

CMY-Farbmodell

Beim CMY-Modell sprechen wir von einer subtraktiven Mischung, da hier davon ausgegangen wird, dass das Licht reflektiert wird. Die Farbwerte Cyan, Magenta und Gelb (Yellow) mischen sich nicht zu Weiß, wie beim RGB-Modell, sondern zu Schwarz, wie in Abbildung 2.6 gezeigt wird. Wir müssen bei diesem Modell davon ausgehen, dass die zu reflektierende Lichtquelle eine weiße Farbvalenz von sich gibt, da wir sonst eine Verfälschung des Ergebnisses bekommen würden. Ansonsten ist auch hier der Farbraum dreidimensional und stellt sich als Würfel dar, wie in Abbildung 2.3 gezeigt.

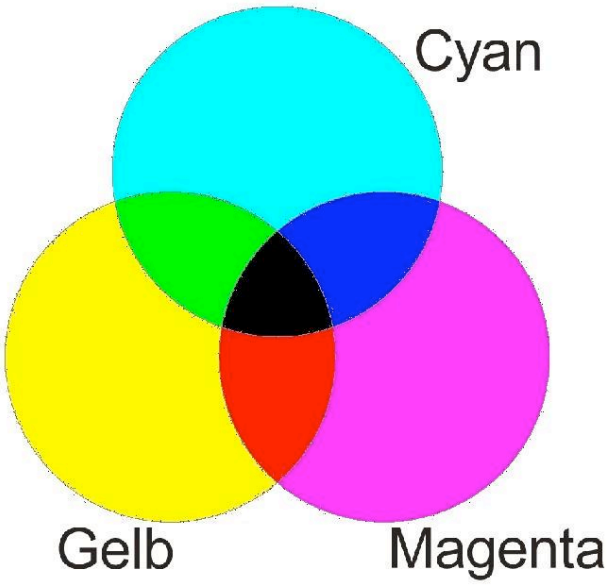

Abbildung 2.6: Subtraktive Mischung [Schmidt00].

HSV-Farbmodell

HSV steht für Hue (Farbton), Saturation (Sättigung) und Value (Helligkeit) und ist an die Deutung von Farben durch den Menschen angelehnt. Auch hier gibt es drei

Werte, welche die Farbvalenz beschreiben. Wir müssen uns hier den Farbraum (siehe Abbildung 2.7) als Kegel vorstellen. Der Winkel im Kegel beschreibt den Farbton, die Vektorlänge die Sättigung und die Höhe innerhalb des Kegels die Helligkeit. Eine Höhe von 0 bedeutet Schwarz, eine Länge von 0 bedeutet Farblos (bezeichnet also einen Grauwert) und ein Winkel von 0° bedeutet Rot.

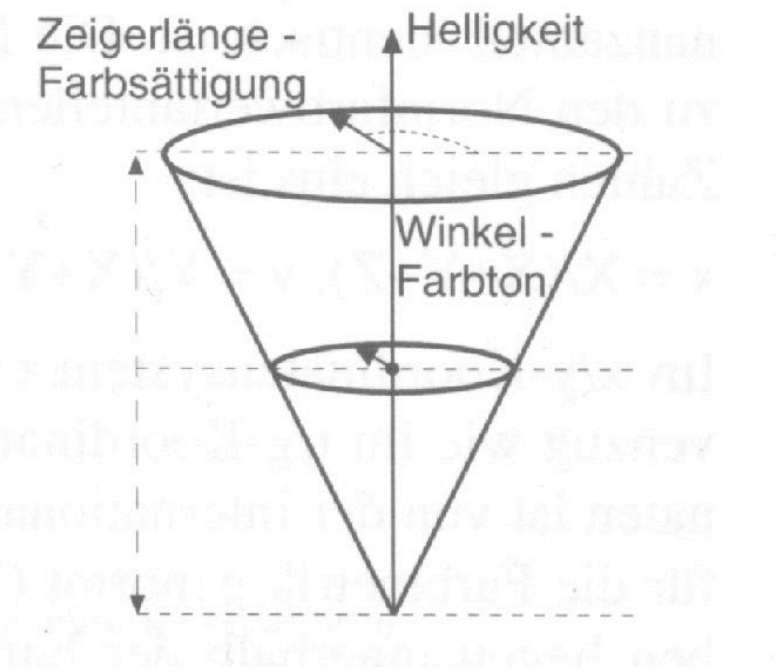

Abbildung 2.7: Farbtüte des HSV-Farbmodells [Schmidt00].

YUV-Farbmodell

Das YUV-Farbmodell entstand als das Farbfernsehen eingeführt wurde, da eine Kompatibilität zum Schwarz-Weiß Fernsehen möglich sein sollte. Der Y-Wert ist der Wert für die Luminanz (Leuchtdichte), welcher direkt im S/W Fernsehen verwendet werden konnte. U und V berechnen sich aus der Differenz von Y und jeweils dem Rot-Wert und dem Blau-Wert. Da sich Y aus allen drei RGB-Werten berechnet und in U und V enthalten ist, steckt die Information von Grün in dem Verhältnis von Y, U

und V. Der Umstand, dass Firewire-Kameras die im Roboterfußball häufig genutzt werden, ein YUV-Signal liefern, kann es notwendig machen, sich auch mit diesem Modell zu beschäftigen.

2.1.4 Kameras

Für Roboter gibt es verschiedene Möglichkeiten die Umwelt zu entdecken. Berührungssensoren, Infrarot-Geber-Nehmer Kombinationen, Ultraschall, Laserscanner und Kameras sind wohl die am meisten eingesetzten Sensoren. Da ich mein Augenmerk in dieser Diplomarbeit auf eine visuelle Wahrnehmung des Roboters lege, folgt hier eine Übersicht über einige Arten von Kameras zur elektronischen visuellen Umgebungserfassung.

Standardkameras

Die einfachsten Kameras sind diejenigen, die aus einem dreidimensionalen Geschehen eine zweidimensionale Projektion machen. Sie bestehen aus einer Optik und einem zweidimensionalen Bildsensor. Die Unterschiede der Kameras liegen in den Bildsensoren, welche die einfallende Lichtintensität messen. Die beiden gängigsten Bildsensoren sind: CCD (Charge Coupled Device) und CMOS (Complementary Metal Oxide Silicon). Der CCD-Bildsensor ist in fast jeder digitalen Kamera zu finden. Jede Fotozelle auf diesem Chip misst die Intensität des einfallenden Lichtes. Drei Farbfilter, je einen für Rot, Grün und Blau, zerlegen das einfallende Licht in die Grundfarben[8]. Für CMOS-Chips, die ähnlich wie CCD-Chips arbeiten, ist anzumerken, dass jede einzelne Fotozelle gesondert angesteuert

[8] Die genaue Funktionsweise wird in [Schmidt00], Kapitel 5 beschrieben.

werden kann, so dass es möglich ist, für jedes Pixel eine eigene Belichtungszeit einzustellen damit das sogenannte Blooming vermieden werden kann.[9]

Kameras für dreidimensionale Bilder

Es gibt auch Kamerasysteme, die aus einem dreidimensionalen Geschehen eine dreidimensionale Projektion erzeugen können. So bietet eine Stereokamera[10] zwei Bilder aus unterschiedlichen Perspektiven an, aus denen dann über Differenzwerte ein dreidimensionales Bild errechnet werden kann. Bei einer 3D-Kamera, die mit Hilfe von Streifenprojektion[11] arbeitet, werden dreidimensionale Größen ebenfalls anhand der Perspektive errechnet. Der Unterschied zu einer Stereokamera besteht darin, dass die zweite Perspektive kein Licht empfängt, sondern Licht in Form eines definierten Musters ausstrahlt. Ein modulationslaufzeitbasiertes 3D-Sichtsystem[12], welches keine zusätzliche Rechenleistung benötigt, misst die Laufzeit von Laserlicht. Bei dieser 3D-Kamera wird die Entfernung (siehe Abbildung 2.8) anhand der durch die Laufzeit bedingten Phasenverschiebung gemessen und jeweils den dazugehörigen Bildpunkten zugeordnet.

[9] „Bei extremer Helligkeit kann der Fall auftreten, dass die hellen Bildstellen vergrößert erscheinen, quasi aufblühen (Blooming Effect)“, [Schmidt00], S. 213.

[10] Wie mit Hilfe einer Stereokamera eine dreidimensionale Projektion generiert werden kann, wird in [Klette96], Kapitel 5 beschrieben.

[11] Die Funktion einer 3D-Kamera mit Hilfe von Streifenprojektion wird in [Klette96], Kapitel 9 beschrieben.

[12] Dieses 3D-Sichtsystem wird in [Heinol01] entwickelt und beschrieben.

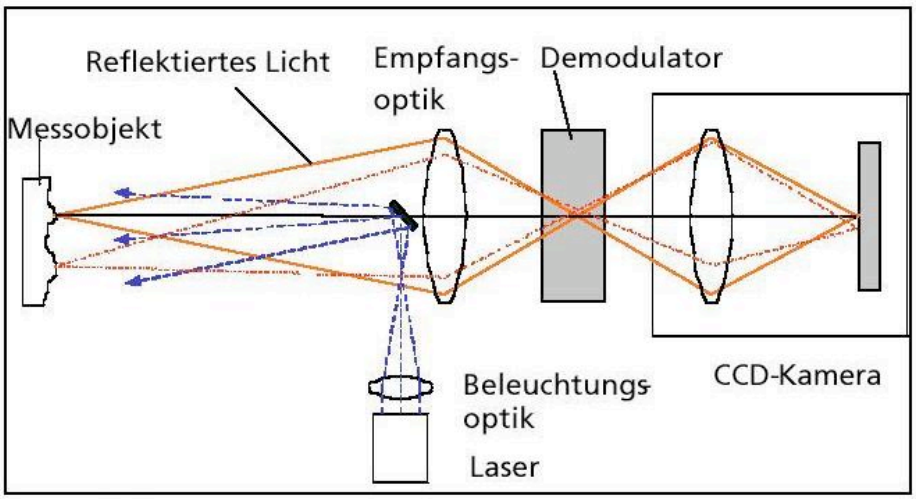

Abbildung 2.8: Funktionsweise eines modulationslaufzeitbasierten 3D-Sichtsystems [3DKamera].

Eine weitere Methode zur Erfassung von Entfernungen in Bezug auf ein Kamerabild basiert ebenfalls auf einer Laufzeitmessung. Bei der CMOS-Entfernungskamera[13] werden neben dem eigentlichen zweidimensionalen Bild noch zwei weitere Bilder mit einer Infrarotlichtquelle aufgenommen. Eins der Infrarotbilder ist ein Referenzbild mit kompletter Infrarotbelichtung. Beim zweiten Bild ist die Belichtungszeit und die Beleuchtungszeit so klein, dass die Laufzeit des Lichtes einen Einfluss auf die Belichtung hat und somit Rückschlüsse auf die Entfernung des reflektierenden Bildteiles gezogen werden können.

Kamera mit Vorverarbeitung

Ein Kameramodul mit eigener FPGA zur Bildvorverarbeitung, die CMUcam2[14], welche von der Carnegie-Mellon-Universität in Pittsburgh entwickelt wurde, stellt

[13] Einen Eindruck, wie eine CMOS-Entfernungskamera mit Hilfe von Infrarotlaufzeiten funktioniert, kann man unter [CMOSEntfer] bekommen.

[14] Die 135,- Euro teure CMUcam2-Platine kann unter [RoboTeile] bestellt werden. Der genaue Funktionsumfang wird dort auch beschrieben.

nicht ein zweidimensionales Bild mit dreidimensionalen Informationen zur Verfügung, sondern interpretiert das aufgenommene Bild und liefert über eine serielle Schnittstelle die vorverarbeiteten Informationen aus dem Videodatenstrom. Es können unter anderem Verfolgungen von benutzerdefinierten Farbobjekten, Erkennung von Bewegungen durch Bildvergleich und horizontale Kantenerkennung durchgeführt werden.

Omnidirektionale Kameras

Eine omnidirektionale Kamera nach dem catadioptrischen Prinzip ermöglicht die Aufnahme eines 360 Grad Bildes um die Blickrichtungsachse der Kamera. In den meisten Fällen besteht eine omnidirektionale Kamera aus einer senkrecht nach oben schauenden Kamera und einem Spiegel, der über der Kamera montiert wird. Die üblichen Spiegelformen wie parabolisch, hyperbolisch, sphärisch, elliptisch oder kegelförmig spiegeln die gesamte Umgebung zur Kamera [vgl. Petrov04].

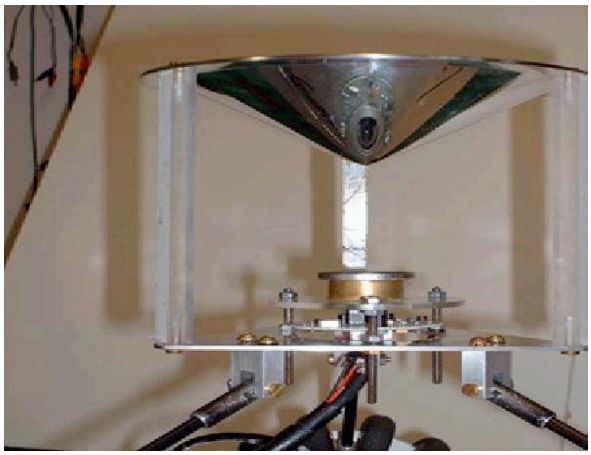

Abbildung 2.9: Omnivision Spiegel und Kamera der FU-Fighters [Petrov04].

Diese Konstruktion führt dazu, dass mit einer omnidirektionalen Kamera die Umgebung mit einem 360 Grad Sichtfeld wahrgenommen werden kann. Das Bild

wird je nach verwendetem Spiegel sehr verzerrt aufgenommen, und die Verzerrungen müssen später wieder herausgerechnet werden. Die Qualität des Bildes leidet ebenfalls. In den RoboCup-Ligen, in denen sich eine lokale Kamera auf den Robotern befindet, wird eine solche Kamera meistens zur Selbst- und Objektlokalisation verwendet.

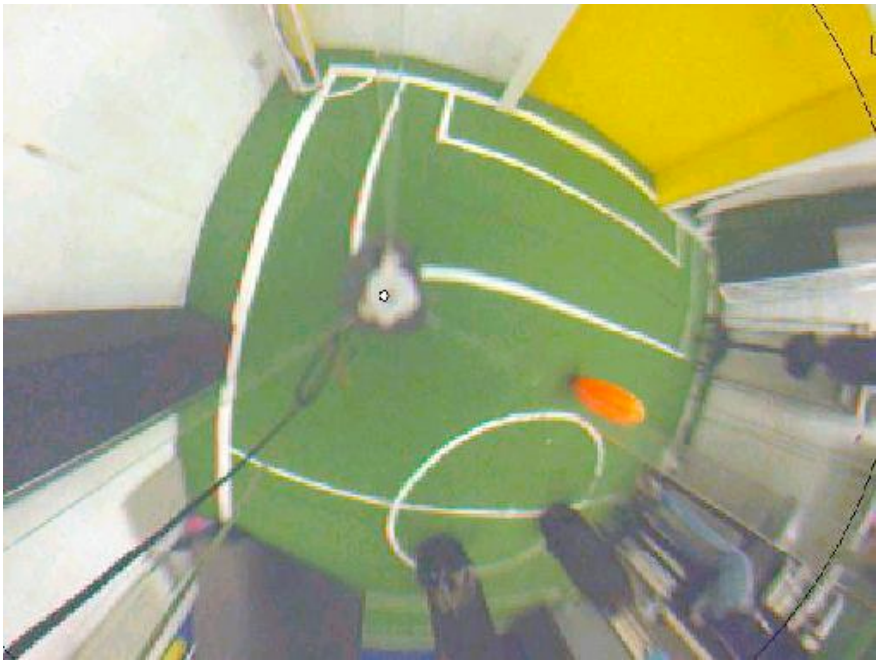

Abbildung 2.10: Umgebungsbild von einer omnidirektionalen Kamera aufgenommen [Petrov04].

2.2 Digitale Regelkreise

2.2.1 Steuerung

„Steuern ist ein Vorgang, der in einem System Größen beeinflusst. Steuerungen haben eine kettenförmige Struktur. Das bedeutet, dass das Signal von Glied zu Glied wandert, ohne auf das vorherige Glied einzuwirken. Dies nennt man einen offenen Wirkungsablauf.“ [Wikipedia]

Es wird also der Stellwert beeinflusst, ohne auf den Istwert zu achten. (Ein Beispiel ist eine Heizung, bei der die Warmwasser Durchflussmenge eingestellt werden kann. Es fließt dann immer gleich viel warmes Wasser durch den Heizkörper, egal wie hoch die Umgebungstemperatur ist.)

Abbildung 2.11: Schematische Darstellung einer Steuerung.

2.2.2 Regelung

Im Gegensatz zur Steuerung weist eine Regelung einen geschlossenen Wirkungsweg auf, über den die gesteuerte Größe nach einem Soll-Istwert-Vergleich auf sich selbst zurück wirkt.

Das heißt, dass der Stellwert abhängig von Istwert und Sollwert ist. Bei einer Heizung bedeutet das, dass wenn die gewünschte Temperatur erreicht ist, die Warmwasser-Durchflussmenge gedrosselt wird.

Ein geregeltes System mit den richtigen Parametern ist sehr stabil und zuverlässig, da es sich innerhalb seiner Parameter an die wirkliche Situation anpasst. Deshalb sind Regler bei Roboterbau und Konzeption nicht wegzudenken, da ein Roboter mit seiner Umwelt agiert und die Umwelt aus „Sicht“ des Roboters kein deterministisches System ist.

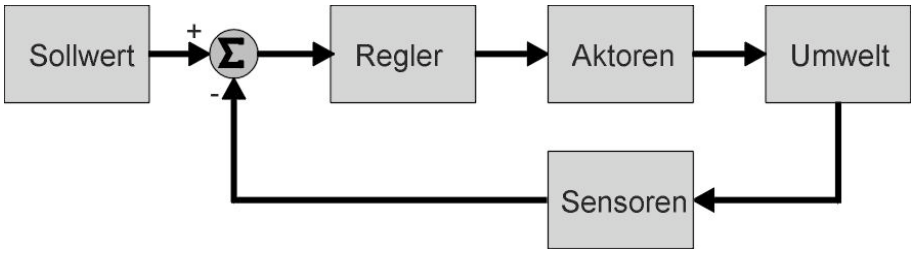

Abbildung 2.12: Schematische Darstellung eines Reglers.

2.2.3 PID-Regler

Der PID-Regler ist aus drei Teilen zusammengesetzt: Proportionalteil, Integralteil und Differentialteil, wie in Abbildung 2.13 gezeigt. PID-Regler oder Teile davon sind die am häufigsten eingesetzten Regler für die Robotersteuerung. Je nach Anwendung und Parametern wird der komplette PID-Regler oder Submengen davon eingesetzt.

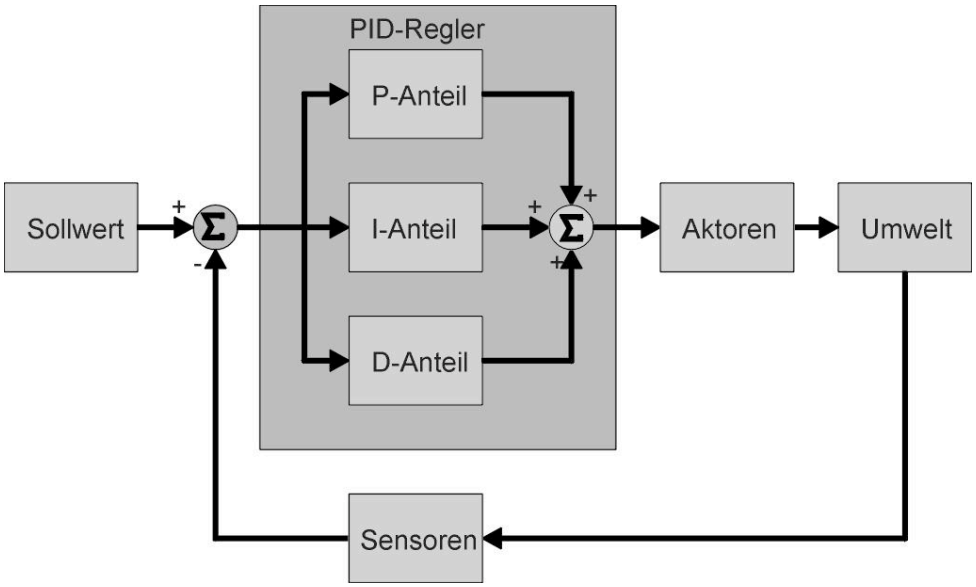

Abbildung 2.13: Schematische Darstellung eines PID-Reglers.

Proportional-Anteil

Der Proportional-Anteil (P-Regler) errechnet sich aus der Differenz zwischen Soll- und Istwert bzw. Führungsgröße und Regelgröße multipliziert mit einem sogenannten Proportionalitätsfaktor. Der Proportional-Anteil ist bei einem P-Regler dann die Stellgröße. Ein reiner P-Regler ist allerdings sehr anfällig gegen Störungen oder große Regeldifferenzen und neigt dann zum Übersteuern und damit zum Aufschwingen. Weiterhin kann ein P-Regler einen systemischen Fehler in der Konstruktion des Roboters, welcher zum Aufschwingen führt, nicht ausgleichen.

Programmbeispiel: panteil=proportionalfaktor*(fuehrungsgroesse-regelgroesse);

Integral-Anteil

Der Integral-Anteil errechnet sich durch die Summe der zeitlich vergangenen Regeldifferenzen in einem bestimmten Zeitintervall. Dieser Teil glättet Störungen

und sprunghafte Regeldifferenzen, so dass eine Regelung mit einem Integral-Anteil ruhig und weich nachregelt.

Programmbeispiel:
```
temp=ianteil;
ianteil=integralfaktor(oldianteil+inanteil);
oldianteil=temp;
```

Dieses Programmbeispiel hat nur eine zeitliche Tiefe von 2.

Differential-Anteil

Der Differential-Anteil errechnet sich durch die Differenz der aktuellen Regeldifferenz und der vergangenen Regeldifferenz. Das bedeutet bei einer positiven Regelgröße, dass der D-Anteil positiv ist, wenn die Regeldifferenz steigt und negativ ist, wenn die Regeldifferenz sinkt. Der Differential-Anteil regelt bei einem Ansteigen der Regeldifferenz verstärkend und bei einem Absinken der Regeldifferenz abschwächend. Der D-Anteil ist sehr anfällig gegen Störungen und führt dann schnell zum Übersteuern. Mit einem angemessenen D-Anteil kann sehr elegant ein Aufschwingen des Regelkreises verhindert werden.

Programmbeispiel:
```
temp=danteil;
danteil=differentialfaktor(danteil-olddanteil);
olddanteil=temp;
```

2.3 Omnidirektionaler Antrieb

Ein Roboter mit omnidirektionalem Antrieb kann sich in alle Richtungen bewegen ohne seine räumliche Ausrichtung zu verändern. Die ersten beiden Beispiele in Abbildung 2.14 sollen dies veranschaulichen.

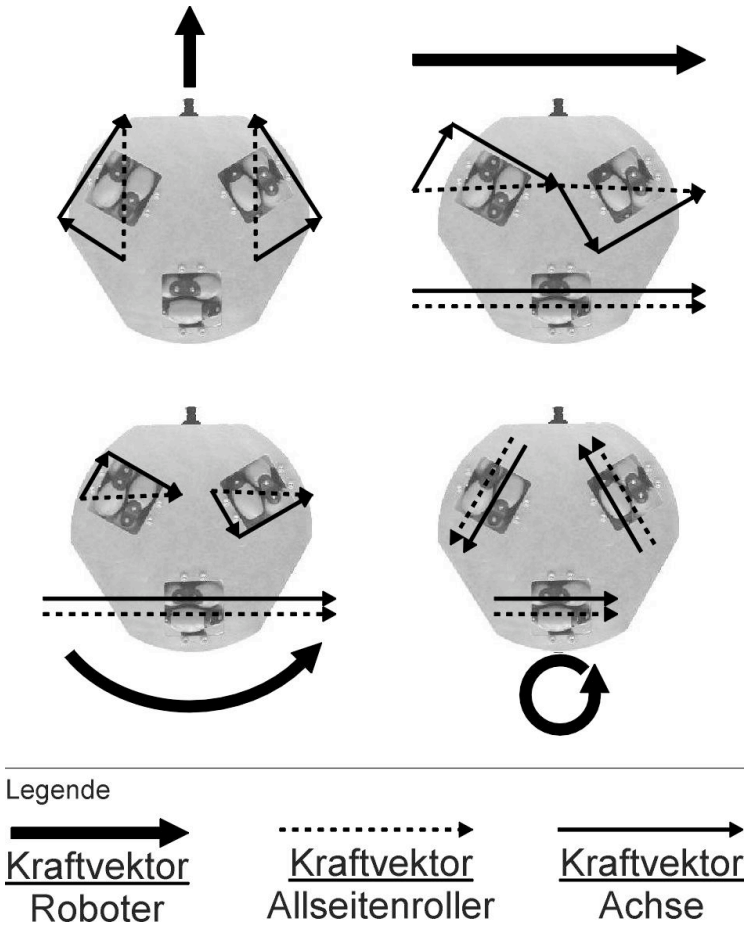

Abbildung 2.14: Beispiele verschiedener Bewegungsmöglichkeiten eines omnidirektionalen Antriebes.

Das beste Beispiel, welches wir kennen, ist der Mensch, denn die meisten Menschen können vorwärts, seitwärts und rückwärts laufen. Da wir keinen omnidirektionalen Blick haben, müssen wir uns allerdings beim Rückwärtslaufen umschauen. In der Welt des Roboterfußballs gibt es zwei grundsätzlich verschiedene omnidirektionale Antriebsarten. In der Humanoid-League und in der Aibo-League[15] erfolgt die Fortbewegung über den Einsatz von Beinen. Die Small-Size-League und die Middle-Size-League verwenden Allseitenräder für den Antrieb. In der Small-Size-League hat sich der omnidirektionale Antrieb schon seit längerem durchgesetzt, während er sich in der Middle-Size-League erst in diesem Jahr endgültig durchgesetzt hat. Das liegt wohl an dem großen Gewicht der Roboter und dem damit einhergehenden größeren Verschleiß der Räder.

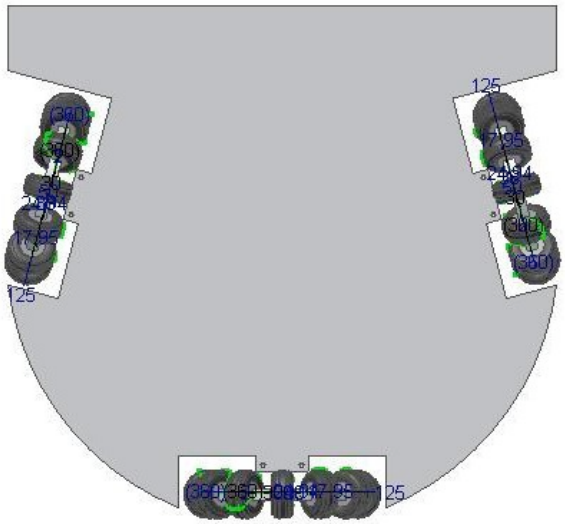

Abbildung 2.15: Schematische Ansicht von unten des omnidirektionalen Antriebes der Fu-Fighter aus Berlin, die Räder stehen hier 150, 105 und 105 Grad zueinander [FuFighters].

[15] Eine Diplomarbeit über vierbeiniges Laufen - Modellierung und Optimierung von Roboterbewegungen findet sich unter [Düffer04].

Bei den meisten omnidirektionalen Antriebssystemen mit Rädern handelt es sich um einen Drei-Achsen-Antrieb mit Allseitenrollen (oder auch Allseitenräder, Omniwheels bzw. Omniräder). Meistens sind die Räder in einem Winkel von 120 Grad gleichmäßig verteilt. Die Allseitenrollen werden aktiv über den jeweiligen Motor angetrieben und können passiv orthogonal zur Antriebsachse über die kleineren Querrollen abrollen.

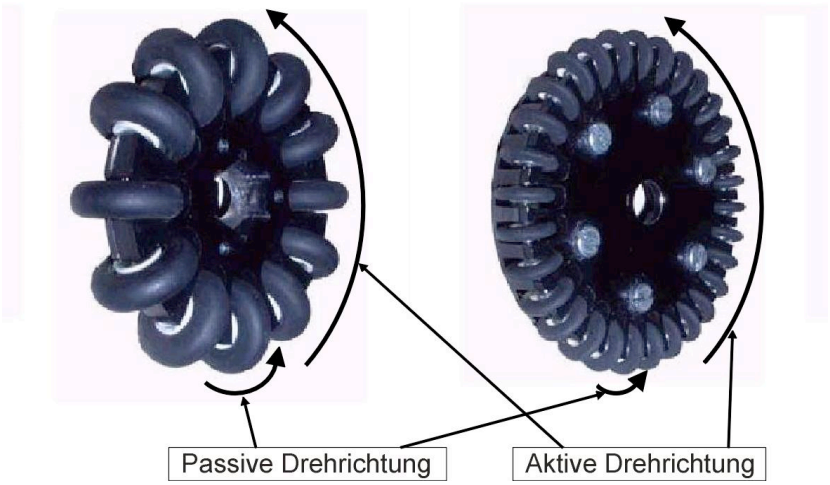

Abbildung 2.16: Zwei Omniräder der Fu-Fighters, links aus 2002 mit 12 orthogonalen Rädchen und rechts aus 2004 mit 30 orthogonalen Rädchen. Die kleineren Rädchen haben zwar einen höheren Verschleiß sind aber bei einem omnidirektionalen Antrieb mit vier Omnirädern nötig, da der Roboter sonst kippelt. [Bild: Gloye05].

Auf der Plattform von Michael Ziener sind die Allseitenroller im Winkel von 120 Grad zueinander angebracht. Dies ermöglicht einen sehr variablen Einsatz der Plattform in alle Richtungen. Die Fu-Fighter haben eine Plattform, bei der der Öffnungswinkel der vorderen zwei Allseitenroller 150 Grad beträgt. Dies macht den Roboter zwar theoretisch nicht schneller, praktisch aber stabiler bei Vorwärts- bzw. Rückwärtsbewegungen, jedoch instabiler bei einer Seitwärtsbewegung. [vgl. Gloye05, Kapitel 4.10.1]. Das Fahrverhalten der Fu-Fighter kommt damit dem Laufverhalten eines Menschen, nämlich vorwärts besser laufen zu können als seitwärts, näher als bei gleichmäßiger Verteilung der Allseitenroller.

3 Programmierung von Robotern

Das Verhalten eines Roboters wird normalerweise durch ein Programm dargestellt. Es gibt verschiedene Arten von Programmiersprachen und verschiedene Ansätze von Programmstrukturen für Roboter, welche sich oft gegenseitig bedingen. Typische Programmiersprachen hierfür sind Maschinencode oder C. Da diese Sprachen sehr hardwarenah und ressourcenschonend sind, können hiermit entwickelte Programme auf einem Mikrokontroller oder einer extra entwickelten Platine für Roboter eingesetzt werden. Weil ein Roboter ein Echtzeitsystem ist und deshalb auf möglichst kleine Latenzzeiten angewiesen ist, werden auch häufig auf ressourcenreichen Systemen hardwarenahe Programmiersprachen eingesetzt. Außerdem wird der Roboter von Spezialisten programmiert und soll ohne fremde Hilfe arbeiten, weswegen keine benutzerfreundliche Umgebung notwendig ist.

Es gibt auch spezielle Roboter-Programmiersprachen, welche meistens in der Industrie ihre Anwendung finden. Eine dieser Programmierumgebungen für Roboter, wenn auch eine sehr alte, ist die SPS (speicherprogrammierbare Steuerung). Bei der SPS handelt es sich programmtechnisch um eine Endlosschleife, in der logische Operationen wie „und“, „oder“ und „nicht“ mit Eingangsvariablen von Sensoren durchgeführt werden, aus denen dann Ausgangsvariablen für Aktoren oder Variablen für andere logische Operationen resultieren. Diese Endlosschleife läuft meistens auf einer speziellen Rechnerarchitektur, die auf SPS-Operationen optimiert ist und einen möglichst kurzen Schleifenzyklus für Echtzeitbedingungen garantiert. Die SPS-Anlage ist meistens stationär in eine industrielle Fertigungsanlage integriert. Deshalb wurde bei der Konzeption nicht auf eine Mobilität einer solchen Anlage Wert gelegt. Folglich ist eine SPS-Anlage in der existierenden Form nicht zur Programmierung von mobilen Robotern geeignet. Interessant ist jedoch, dass durch die permanent laufende Programmschleife alle Regeln praktisch parallel abgearbeitet werden, was zur Folge hat, dass der

Programmierer alle Situationen und Eventualitäten gleichzeitig verarbeitbar machen muss. Dies setzt eine bestimmte Herangehensweise an Probleme voraus und bestimmt damit die Problemlösungsstruktur. Diese Art der Problemlösung beeinflusst, wie später zu sehen sein wird, auch meine Herangehensweise zur verhaltensbasierten Bewegungskoordination.

3.1 Programmierung der Parameter

In allen Fällen der Programmierung braucht das Roboterprogramm Parameter für die Sensorauswertung und für die Aktorenansteuerung. Diese Parameter können z.B. durch Analyse, Ausprobieren oder Intuition herausgefunden werden und direkt oder in einer zentralen Datei in das Programm eingefügt werden. Hierbei wird „off-line“ am Programm gearbeitet und der Roboter wird nur zum Testen benötigt. Wenn der Roboter die Parameter lernen soll, dann wird der Roboter während der Programmierphase benötigt, da die Parameter in diesem Fall „on-line“ programmiert werden [vgl. Siegert96].

3.2 „on-line“ Programmierung durch Beispiele

Eine Möglichkeit, den Roboter durch Beispiele zu programmieren ist, den Roboter mit der Hand, also mechanisch, in die jeweiligen Zustände zu bringen, in denen er sich befinden soll. Der Roboter registriert hier die Parameter für die Zielzustände sowie auch die der Zustände auf dem Weg zum Ziel und kann die Bewegungen dann einfach wiederholen. Diese Methode funktioniert allerdings nur, wenn eine ausreichende Odometrie vorhanden ist und es keine Veränderungen im Verhältnis der Sensoren, Umgebung und Aktoren geben kann, wie es zum Teil bei Industrierobotern der Fall sein kann. Die Zustandsveränderungen, die der Roboter vollziehen soll, kann auch durch einen Joystick, eine Maus oder verschiedene dreidimensionale Eingabegeräte erfolgen. Auch in diesen Fällen wird dem Roboter

die gewünschte Veränderung vorgeführt, der Roboter „merkt“ sich die Zielzustände und die Wegpunkte zu diesen. Wenn der Roboter nicht erreichbar oder zu groß bzw. zu klein ist, spricht man von einer „Teleoperation“ oder „Master-Slave-Programmierung“[16]. Bei der „Master-Slave-Programmierung“ wird ein Pendant des eigentlichen Roboters manipuliert. Bei der „Teleoperation“ kann einem Roboter in gefährlicher Umgebung oder in weiterer Entfernung ein Beispiel gegeben werden. Die Manipulation selber erfolgt z.B. über einen Joystick und die Überprüfung der Manipulation erfolgt dann meistens über ein Kamerabild der wirklichen Szene. Problem bei der Teleoperation ist die Zeitverzögerung, welche durch Vorhersagen vermindert werden kann. Für diese Art der Roboterprogrammierung sind keine Programmierkenntnisse erforderlich und die konstruktive Ungenauigkeit wird gleich mit der Programmierung ausgeglichen.

3.3 „on-line“ Programmierung durch Training

Bei der „on-line“ Programmierung durch Training wird dem Roboter zunächst eine reale Aktion vorgeführt oder eine Reihe von abstrakten Sensordaten einer Aktion zum Lernen vorgegeben. Die Programmierung besteht nun darin, dass der Roboter immer wieder versucht, sein eigenes Verhalten den vorgegebenen Daten anzupassen bis sein eigenes Verhalten den Vorgaben in Geschwindigkeit oder Genauigkeit so stark gleicht, dass es der Qualitätsvorgabe des Programmierers genügt. Zur Messung der Genauigkeit werden Sensoren benötigt, die dem Roboter die Möglichkeit geben, die Ist- und Sollwerte von jedem Schritt zu vergleichen. Diese Art der Programmierung wird häufig zur Bilderkennung oder zur Sensorintegration verwendet. Neuronale Netze sind in diesem Zusammenhang wohl wichtigstes Schlagwort und werden im Folgenden kurz erläutert.

[16] [vgl. Siegert96, S.91]

3.3.1 Neuronale Netze

Neuronale Netze sind eine vereinfachte Abstraktion von Vorgängen des menschlichen Gehirns, wie sie heutzutage von der Wissenschaft dargestellt werden [siehe Zell94]. Die Knoten des Netzes sollen dabei die Neuronen darstellen und die Kanten die Synapsen. Alle eingehenden Synapsen werden gewichtet und aufsummiert. Wird ein Schwellwert überschritten, so feuert das Neuron, also der Knoten, und legt ein Signal auf den Ausgang. Dadurch entsteht aus einem Eingangsmuster ein Ausgangsmuster, welches durch die Gewichtungen und die Schwellwerte definiert ist. Beim Training des neuronalen Netzes wird ein Ausgangsmuster und ein Eingangsmuster vorgegeben, und die Knoten im Netz passen ihre Gewichte und Schwellwerte den jeweiligen Vorgaben des Trainings so an, dass später aus den jeweiligen Eingangsmustern ein nahezu identisches Ausgangsmuster reproduzierbar ist. Vor den 90er Jahren wurden neuronale Netze hauptsächlich zur Bild- und Sprachverarbeitung eingesetzt und seit den 90ern auch zur Manipulation von Robotern.[17] Neuronale Netze sind meistens unempfindlich gegen Störungen, können ihre Parameter während des Betriebs an eine sich ändernde Umwelt anpassen und sind gut im Zusammenhang mit einfachen parallelen Systemen einsetzbar und damit echtzeitfähig.

[17] [vgl. Bruske97].

3.4 Roboterorientierte Programmierung

Roboterorientierte Programmierung baut auf einer Programmiersprache wie beispielweise C auf. Hier werden roboterspezifische Aktionen wie Bewegungsbefehle oder Greifbefehle, die der Roboter ausführen kann, z.B. in C definiert. Diese Definition ist dann entweder in einer eigenen roboterorientierten Entwicklungsumgebung oder als Bibliothek benutzbar. Der Programmierer braucht sich nicht mehr darum zu kümmern wie genau die Motoren angesteuert werden, es reicht die Mitteilung sich vorwärts bewegen zu wollen.[18]

3.5 Aufgabenorientierte Programmierung

Bei der aufgabenorientierten Programmierung geht es nicht mehr darum, dem Roboter die einzelnen Schritte mitzuteilen, die notwendig sind, um eine Tasse Kaffee zu holen, sondern nur noch darum, ihm mitzuteilen, dass er eine Tasse Kaffee holen soll.[19] Um diese Aufgabe lösen zu können, braucht die aufgabenorientierte Programmierumgebung folgendes:

[18] [vgl. Siegert96, S. 105].

[19] [vgl. Siegert96, S. 197].

- ein Weltmodell
- das Wissen darüber, in welche Einzelschritte eine Aufgabe zerlegt werden kann
- Planungsalgorithmen für die einzelnen Teilaufgaben
- Synchronisationsmuster zur Koordination der Tätigkeiten des Roboters mit der Umwelt
- Sensordatenfusion

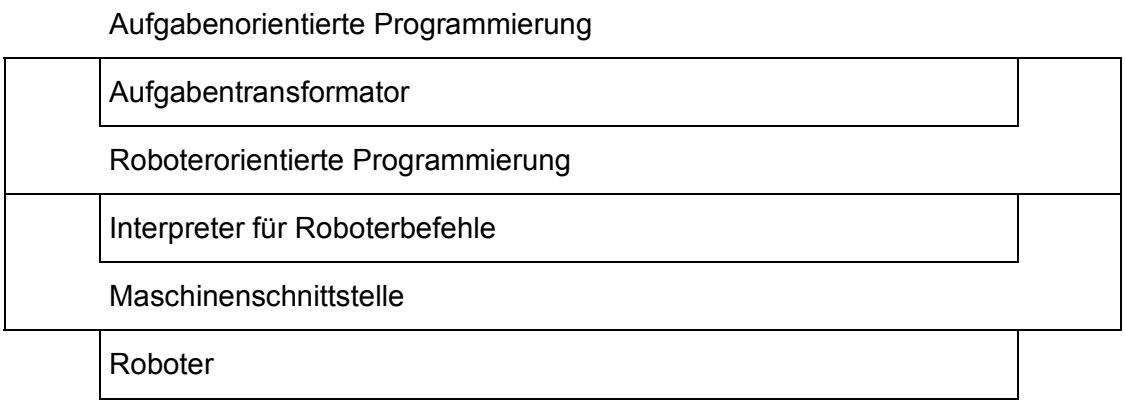

Abbildung 3.1: Schichtenmodell beim Aufgabentransformator [vgl. Siegert96, S. 199]

3.6 Programmierung mobiler Roboter mit beschränkten Ressourcen

Die Programmierung eines Roboters anhand eines Weltmodells, wie in Abbildung 3.2 gezeigt, stellt uns, vorausgesetzt der Roboter agiert in einer stark wechselnden Umgebung und sein Programm läuft auf einem Mikrokontroller oder Laptop, vor das

Problem, dass der Roboter sehr viel Zeit braucht, um die Sensordaten und die daraus folgenden Konklusionen in dieser Programmarchitektur zu verarbeiten.

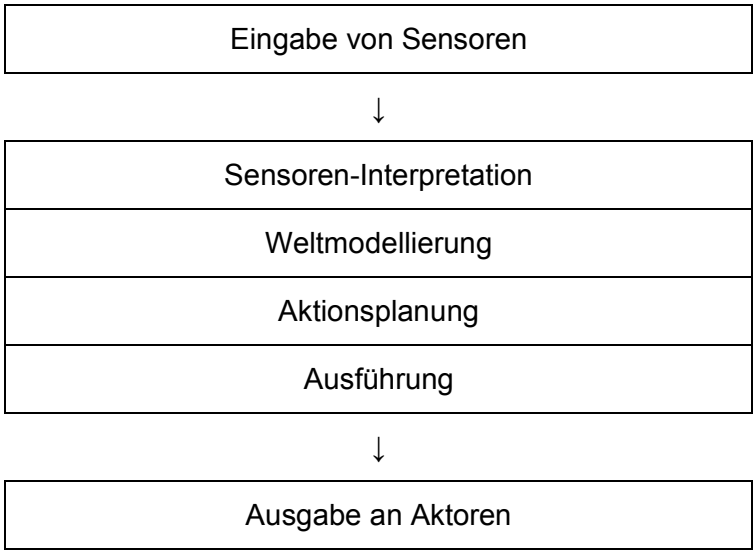

Abbildung 3.2: „Ein Roboterprogramm nach dem Paradigma der Weltmodellierung und Aktionsplanung besteht aus einer Reihe von Schritten. Diese Funktionseinheiten setzen die zu einem Zeitpunkt aufgenommenen Sensordaten (einen Sensorschnappschuss) in eine Reihe von Handlungen mit einem bestimmten Ziel um.“ [Jones96, S. 242]

3.6.1 Subsumtionsmethode

Es ist zwar ein Vorteil, wenn ein genaues Weltmodell existiert, denn es können klarere Entscheidungen getroffen und das Geschehen besser interpretiert werden, aber es ist sehr viel Aufwand, möglichst die ganze „Welt“ mit Sensoren intern darzustellen. Der Subsumtionsansatz von Professor Rodney Brooks und der Arbeitsgruppe für mobile Roboter am MIT Artificial Intelligence Laboratory bietet hier eine Alternative. Bei dieser Methode wird nicht anhand aller Sensoren entschieden, sondern die Entscheidungen hängen von Teilaspekten und damit von wenigen Sensoren ab. Wird nun eine Abhängigkeit von hoher Priorität entdeckt, werden die Interpretationen der restlichen Sensoren nicht mehr benötigt. Umgangsprachlich

würden wir hier von einem „Tunnelblick“ in einer Stresssituation sprechen. Bei der Subsumtionsarchitektur findet also nicht wie beim Weltmodell eine globale Sensorfusion statt, sondern eher eine Verhaltensfusion. Zur Veranschaulichung wird in Abbildung 3.3 ein einfaches Verhaltensmodell eines mobilen Roboters mit Subsumtionsarchitektur dargestellt. Der Roboter in diesem Beispiel folgt einer vorhandenen Lichtquelle - solange die IR-Sensoren und Bumper[20] kein verwertbares Signal liefern - mit den Fotozellen. Wenn nun ein Hindernis in Kontakt mit dem Roboter tritt, verändert der Roboter sein Verhalten und versucht entweder zu entkommen oder es zu umfahren, je nachdem ob die Bumper oder die Infrarotsensoren ansprechen.

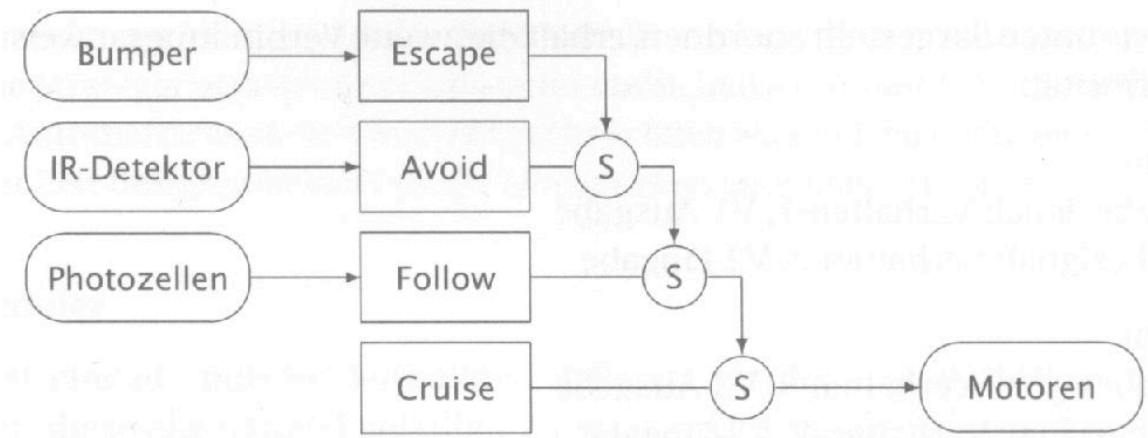

Abbildung 3.3: Einfaches Beispiel für eine Subsumtionsarchitektur [Jones96, S. 258].

Das Beispiel in Abbildung 3.3 veranschaulicht die Funktionweise der Subsumtionsarchitektur, allerdings sind die Verhaltensmuster im Beispiel sehr vereinfacht und können durch wesentlich komplexere Verhaltensmuster ausgetauscht werden.

[20] Bumper sind Berührungssensoren, die anzeigen, dass der Roboter physikalischen Kontakt mit einem Hindernis hat.

3.6.2 Dual Dynamics Ansatz

Verhaltensmodule, die aus zwei Teilen bestehen sind die Grundlage von Dual Dynamics. Ein Teil berechnet, ob das Verhaltensmodul in der jeweiligen Situation aktiviert werden soll und damit das Gesamtverhalten beeinflussen soll. Ein anderer Teil berechnet, welche Befehle an die Motoren geschickt werden sollen. Die Abstraktion einer Sequenz von Situationen ist ein Aktivierungsmuster für ein Verhalten. Deshalb werden, wie bei der Subsumtionsmethode, nur die Sensoreninformationen ausgewertet, die relevant für die jeweilige Situation sind. Diese Abstraktion verringert die Komplexität der jeweiligen Analyse und erhöht damit auch die Performance des Gesamtsystems. Der Roboter muss nicht das „big picture“, also den Gesamtsachverhalt, erkennen, sondern nur die momentane Situation und zeitlich nahe Abhängigkeiten [vgl. jaeger96].

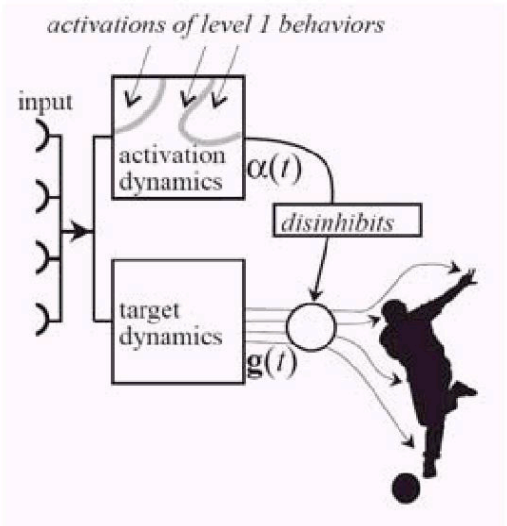

Abbildung 3.4: „Der Dual Dynamics Ansatz von Jaeger. Jede Ebene des Verhaltens besteht aus einer Aktivierungsdynamik, die entscheidet, ob das Verhalten die Aktuatoren beeinflussen darf, und einer Zieldynamik, welche die Aktuatoren des Roboters tatsächlich beeinflusst.“ [Gloye05] [Abbildung aus Breden99].

4 Roboterlabor der HAW Hamburg

Bei der Betrachtung des zur Verfügung stehenden Materials im Roboterlabor der HAW Hamburg werde ich mich auf die Roboterbauteile oder Plattformen beschränken, die variabel einsetzbar sind und den in der Zielsetzung genannten Kriterien autonom, visuell gesteuert, erweiterbar und freibeweglich, entsprechen. Diese Kriterien schränken die zur Verfügung stehende Hardware ein.

4.1 Fahrbare mechanische Plattformen

Die fahrbaren mechanischen Roboterplattformen des Roboterlabors sind: der Staubsaugerroboter Roomba, ad hoc aus Lego gebaute Plattformen, aus Spielzeug umgebaute kettenbetriebene Fahrzeuge, verschiedene Pioneer-Varianten, eine Eigenkonstruktion von Michael Manger mit omnidirektionalem Antrieb und eine sehr stabile Eigenkonstruktion von Michael Ziener, ebenfalls mit omnidirektionalem Antrieb. Zur Verwendung werden Lego, Roomba und die Kettenfahrzeuge nicht in Betracht gezogen, da sie entweder nicht stabil und haltbar genug sind oder eine Weiterentwicklung wie beim Roomba eher einen Kompromiss darstellen würde, weil der Roomba ein auf einen bestimmten Zweck hin konstruiertes und abgeschlossenes System ist und nicht zur Weiterentwicklung konstruiert wurde.

Die Pioneer-Roboter sind sehr stabil gebaut und zuverlässig. Sie wurden in der Vergangenheit auch in der Middle-Size-League eingesetzt und haben u.a. bereits ein Kameramodul und Ultraschall eingebaut. Über eine serielle Schnittstelle könnte ein Pioneer-Roboter z.B. mit einem leistungsstarken Laptop mit Webcam im Huckepack kommunizieren, um komplexere Berechnungen und Verhaltensweisen zu unterstützen. Die Plattform von Michael Manger ist zwar mit einem

omnidirektionalen Antrieb ausgestattet, allerdings nicht sehr stabil und hat nur sehr leistungsschwache Servomotoren für den Antrieb. Die zur Zeit neu entwickelte Plattform von Michael Ziener ist eine sehr stabile, gut erweiterbare Plattform mit omnidirektionalem Antrieb ohne festgelegtes Kontrollerboard.

Da der Pioneer mit einem Differentialantrieb und einem integrierten Kontrollerboard ausgestattet ist und damit weder so gut erweiterbar, noch so flexibel einsetzbar ist wie die Plattform von Michael Ziener, entschied ich mich für die Plattform mit omnidirektionalem Antrieb.

4.2 Kontrollerboards

An der HAW Hamburg sind zwei verschiedene, für diese Diplomarbeit geeignete, variable Kontrollerboards in Benutzung. Die Wahl des für diese Diplomarbeit eingesetzten Kontrollerboards ist bereits durch die Entscheidung von Lars Brandt in seiner Diplomarbeit für den RCube und damit für das AkSen-Board vorbestimmt. Der Vollständigkeit halber werde ich hier die beiden Kontrollerboards vorstellen. Danach werde ich noch den RCube erläutern, der von Lars Brandt benutzt wurde und der eins der beiden vorgestellten Kontrollerboards beinhaltet.

4.2.1 MIT 6.270-Board mit Expansionsboard

Das MIT 6.270-Board[21], welches mit einem Motorola 68HC11 8 Bit Mikrokontroller mit 8Mhz betrieben wird, entstand im Rahmen der Dissertation zu alternativen Lernformen von Fred Martin am Massachusetts Institute of Technology.

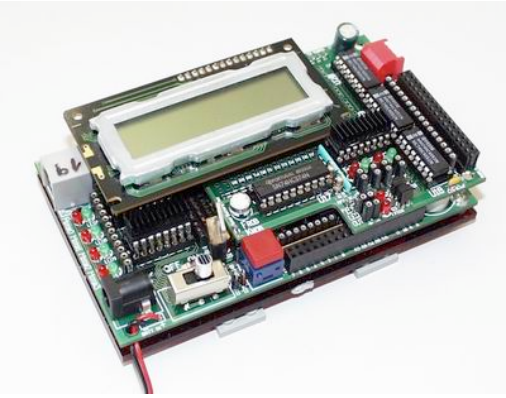

Abbildung 4.1: MIT 6.270-Board.

Das Roboterboard hat folgende Eigenschaften: 32 KB RAM, sechs bidirektionale phasenmodulierte Motortreiber, zwei LED-Ausgänge, einen Servo-Anschluss, einen modulierten Infrarotausgang, 8 digitale Eingänge, 16 analoge Eingänge, eine serielle Schnittstelle und ein LC-Display. Das Board, welches mit einem 8 x schnelleren C64 vergleichbar ist, wird in „interactive C“ programmiert und ist multitaskingfähig.

Dieses Board würde sich als Kontrollerboard für rudimentäre Programmstrukturen sowie zum Einlesen von Analog- und Digital-Sensoren und Ansteuerung von den Aktoren gut eignen. Eine Verbindung über die serielle Schnittstelle mit einem Laptop

[21] Eine genaue Beschreibung und ein Tutorial findet sich in [Kraetz94].

oder einem Palmtop ist hier gut möglich und wurde auch schon an der HAW Hamburg praktiziert [Gerling03].

4.2.2 AkSen-Board

Das AkSen-Board[22] wurde an der FH-Brandenburg als Nachfolge vom 6.270-Board entwickelt. Die Möglichkeiten dieses Boards sind umfangreicher und es ist durch ein Sandwichprinzip und einen CAN-Bus einfach und kompakt mit anderen Boards kombinierbar.

Es hat 15 analoge Eingänge, 16 digitale, frei konfigurierbare Ein- und Ausgänge, vier bidirektionale phasenmodulierte Motortreiber, vier LED-Treiber, drei Servo-Ausgänge, einen modulierten Infrarotausgang, drei Encoder-Eingänge, eine serielle Schnittstelle, ein CAN-Bus Interface, 64 KB Flash-RAM und ein LC-Display. Getrieben wird das Board von einem 8 Bit SAB80C515A Microkontroller mit 12 MHz, der unter Windows oder Linux mit nativem GPL-C programmiert wird.

Das AkSen-Board stellt zu diesem Zeitpunkt wohl die im Preis-Leistungsverhältnis beste Lösung zur programmgesteuerten Aktoren- und Sensorenverwaltung dar.

Abbildung 4.2: AkSen-Board.

[22] AkSen steht für Aktoren-Sensoren. Diese und weitere Informationen wurden aus [LaborKIFHB05] entnommen.

Da das AkSen-Board ein Bestandteil des RCubes ist und über einen CAN-Bus mit den weiteren Boards, welche eine Kameraunterstützung ermöglichen, verbunden werden kann, außerdem ebenfalls leicht beschaffbar ist und es einen direkten Kontakt zu den Projekt betreuenden Personen gibt, ist das AkSen-Board mit der Benutzung des RCubes eine gute Wahl für dieses Projekt.

4.3 RCube-Plattform

Der RCube[23] ist ein Roboterboard, welches aus den drei Komponenten CPU-Board, Videoboard und dem AkSen-Board besteht. Die Verbindung zu den üblichen Aktoren und Sensoren übernimmt das AkSen-Board. An das Videoboard können bis zu vier Kameras[24] angeschlossen werden und ein Videoausgang macht das Debuggen einfacher. Das Herzstück ist eine CPU-Karte mit einem 200 MHz StrongARM Prozessor und 32 MB RAM auf der ARM-Linux läuft. Das alles befindet sich auf einer Grundfläche von 7x10 cm und kommuniziert über einen CAN-Bus miteinander.

[23] Diese und weitere Informationen zum RCube finden sich unter [RCube05].

[24] Diese vier Videoeingänge können nicht gleichzeitig verwendet werden, da sie multigeplexed werden.

Abbildung 4.3: RCube-Roboterplattform.

Der RCube wurde an der FH-Brandenburg entwickelt und stellt eine "preisgünstige" Roboterplattform für sehende intelligente autonome Systeme dar.

Weil das Videoboard den Arbeitsspeicher des CPU-Boards nutzt, ist die Belastung des Busses sehr hoch. Bei der Nutzung eines Videokanals im Debugmodus, also mit Videoein und –ausgabe, wird der Bus schon alleine durch das Videoboard mit ca. 62 MB/s von 100 MB/s (720 x 576 Pixel (PAL) x 3 Byte (RGB) x 25 Bilder/Sekunde x 2 (VIO)) belastet. Dieser Nachteil ist leider unumgänglich, da er konstruktionsbedingt ist. Folglich ist ein schonender Umgang mit den Ressourcen erforderlich.

5 Projekte in Bearbeitung

An der HAW Hamburg sind zur Zeit zwei Diplomarbeiten in Arbeit, die direkt mit dieser Diplomarbeit verknüpft sind. Lars Brandt beschäftigt sich in seiner Diplomarbeit damit, einen farblich markierten Ball mit Hilfe des RCubes und einer Kamera zu erkennen. Michael Ziener arbeitet an der Programmierung und Konstruktion einer stabilen Roboterplattform mit omnidirektionalem Antieb.

5.1 Objekttracking mit Hilfe einer Kamera

Mit dem von Lars Brandt entwickelten Objekttracker können Objekte anhand eines Farbwerte- und Sättigungsbereiches innerhalb eines durch eine Kamera aufgenommenen zweidimensionalen Bildes lokalisiert werden [siehe Brandt05].

Der Objekttracker wurde zuerst auf einem PC in Java entwickelt und nach Abschluss der Entwicklungsphase auf dem RCube in der Programmiersprache C implementiert. Der eigens entwickelte Algorithmus zur Erkennung von Objekten, welcher auf knappe zeitliche Ressourcen ausgelegt ist, verspricht, auch auf einem Embedded System wie dem RCube in passabler Zeit zu laufen.

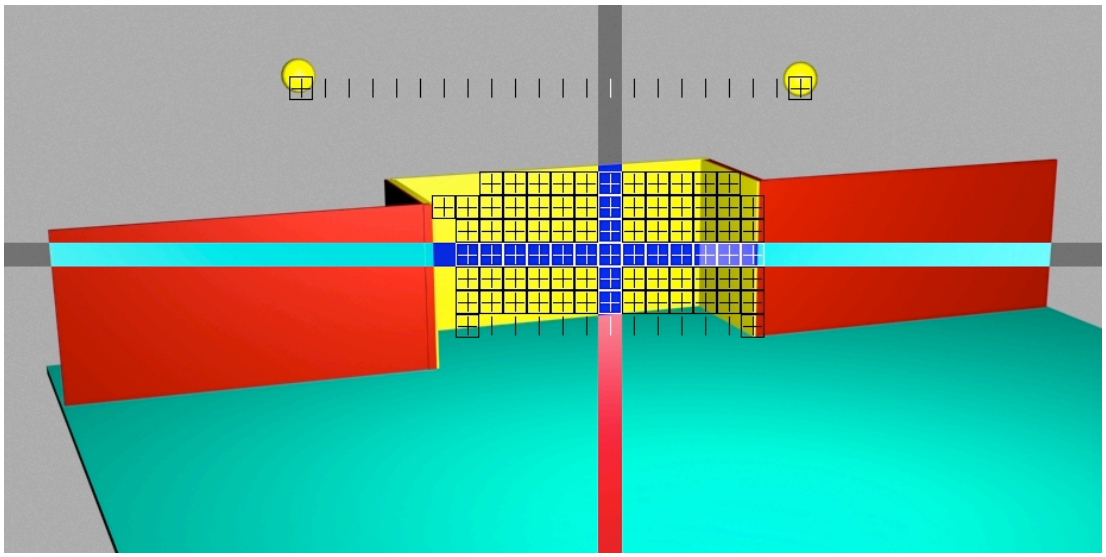

Abbildung 5.1: Der Objekttracker liefert die X-/Y-Koordinaten des gelben Objektes [Brandt05].

Bei der Erkennung von Objekten wird ausschließlich mit Farbwerten und nicht mit der Form, Größe oder anderen Charakteristika von Objekten gearbeitet. Es werden nicht alle Pixel verwertet, folglich gibt es immer eine Schatteninformation. Allerdings kann die gewünschte Auflösung eingestellt werden, so dass je nach Problematik der Objekttracker entweder auf Geschwindigkeit oder auf Genauigkeit getrimmt werden kann.

Der Rückgabewert des Objekttrackers besteht aus einer X- und Y-Koordinate, die den Median aller vermuteten Farbflecken im Bild wiedergeben. Das Bild wird vom Algorithmus für jede Zeile von links und gegebenenfalls von rechts nach dem gesuchten Farbwert durchsucht. Der Suchvorgang wird für die jeweilige Richtung und Zeile gestoppt, wenn der gesuchte Farbvalenzbereich gefunden wurde. Es wird davon ausgegangen, dass Objekte geschlossen und stetig sind, so dass der Zwischenraum von zwei gefundenen Punkten innerhalb einer Zeile automatisch als erkannt gilt.

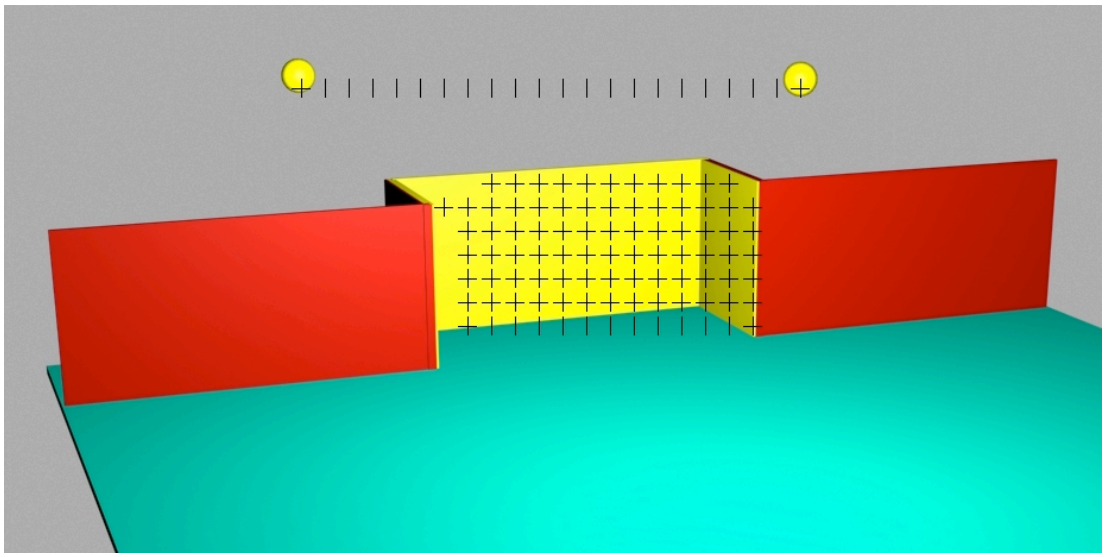

Abbildung 5.2: Es wird in horizontaler und vertikaler Richtung nach gelben Blöcken gesucht [Brandt05].

Da es aber nicht nur das gesuchte Objekt in diesem Farbvalenzbereich geben kann, sondern auch „Farbschmutz“ oder Störungen, würde dies zu einer falschen Vergrößerung des Objekts führen. Deshalb wird der gleiche Vorgang für die Zeilen nochmal für die Spalten wiederholt und nur die Punkte ausgewählt, die in beiden Scan-Vorgängen ausgewählt wurden. Das Tor wird markiert und der Zwischenraum zwischen dem „Schmutz“ wird nicht markiert wie in Abbildung 5.2 gezeigt wird.

Der Rückgabewert des Objekttrackers ist der Median, wie es Abbildung 5.1 veranschaulicht. Dadurch, dass der Median und nicht der Mittelwert zurückgegeben wird, ist sichergestellt, dass auch bei einem durch ein Hindernis geteiltes Objekt,

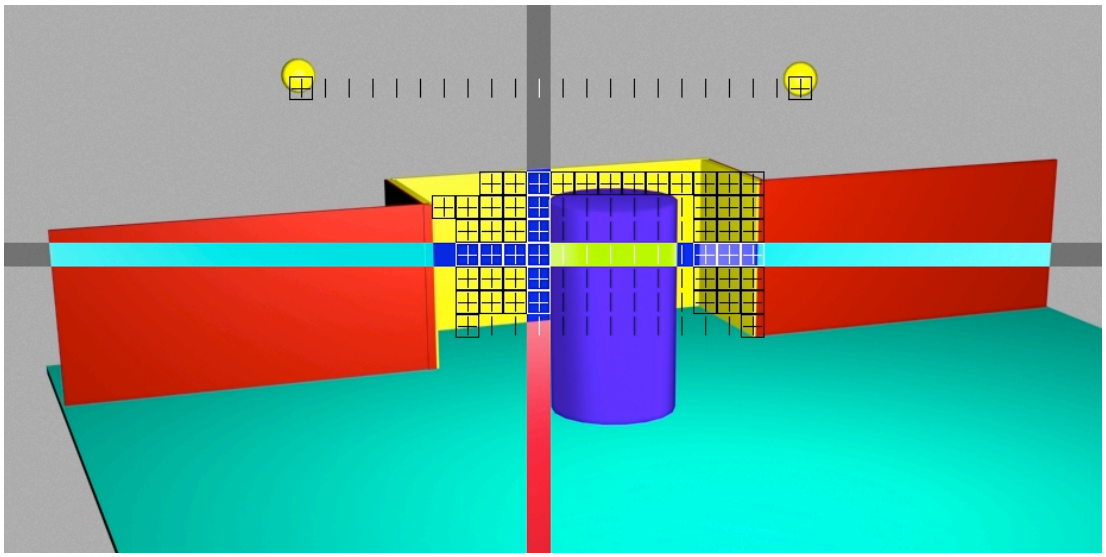

Abbildung 5.3: Der Median des Objektes (gelbes Tor) liegt neben dem Hindernis [Brandt05].

wie Abbildung 5.3 zeigt, die X-/Y-Koordinaten des Rückgabewertes auf jeden Fall auf einem Teil des Objektes liegt. Dabei wird der größere Teil des Objektes anvisiert.

Der Objekttracker liegt als C-Code vor und kann so in das eigene Programm integriert und gegebenenfalls erweitert oder modifiziert werden.

Abbildung 5.4: Die richtigen Parameter zu finden gestaltet sich bei automatischem Weißabgleich der Kamera und leichten Beleuchtungsveränderungen schwierig.

5.2 Roboterplattform mit omnidirektionalem Antrieb

Michael Ziener arbeitet zur Zeit in seiner Diplomarbeit an einer mechanischen Roboterplattform inklusive Funktionalitäten zur Ansteuerung des Antriebes. Die Antriebsform der Roboterplattform ist omnidirektional [siehe Ziener05].

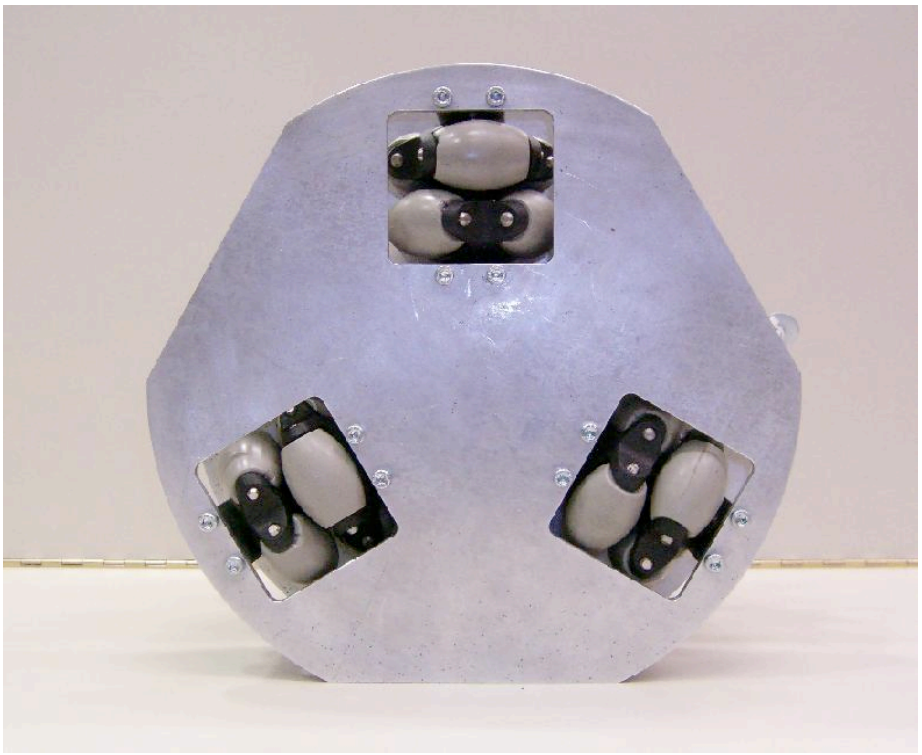

Abbildung 5.5: Omnidirektionaler Antrieb von Michael Ziener (Ansicht von unten).

Die Allseitenräder stehen, wie in Abbildung 5.5 gezeigt, gleichmäßig in einem Winkel von 120 Grad zueinander. Dadurch gibt es keine bevorzugte Fahrtrichtung, was für eine Experimentierplattform von Vorteil ist. Die Allseitenräder sind von der Firma Interroll, haben einen Durchmesser von 80 mm und eine Tragkraft von 250N. Da die Lauffläche, der Allseitenräder glatt ist und aus hartem Kunststoff besteht,

haben diese Räder auf den meisten Untergründen ein wenig Schlupf, welches die Genauigkeit der Roboterbewegung beeinträchtigt. Weicheres Gummi und Struktur in der Lauffläche wie bei den gezeigten Allseitenräder in Abbildung 5.6, würden hier zu einem genaueren Fahrverhalten des Roboters führen.

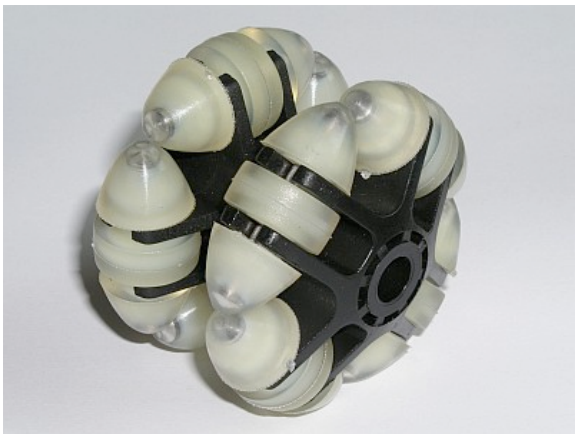

Abbildung 5.6: Ein Allseitenroller mit Grip [RoboTeile].

Getrieben werden die Allseitenräder über einen Zahnriemen mit einem Übersetzungsverhältnis von ungefähr 1 zu 2 von RB-35 11-Type Motoren mit Planetengetriebe. Das Drehmoment und die hohe Übersetzung machen es möglich, den Roboter auf verschiedenen Untergründen einzusetzen und auch die Pulsweitenmodulation lässt sich dadurch in einem Bereich von 60 % gut und relativ linear einsetzen, so dass die Ansteuerung der omnidirektionalen Antriebsräder nicht unbedingt eine Regelung und damit verbundene Odometrie benötigt. Um nicht auf die jeweiligen Eigenschaften eines Motortreibers auf einem Kontrollerboard angewiesen zu sein, wurde von Michael Ziener eine separate Motortreibereinheit auf der Roboterplattform angebracht, wodurch die Anzahl der nutzbaren Kontrollerboards steigt.

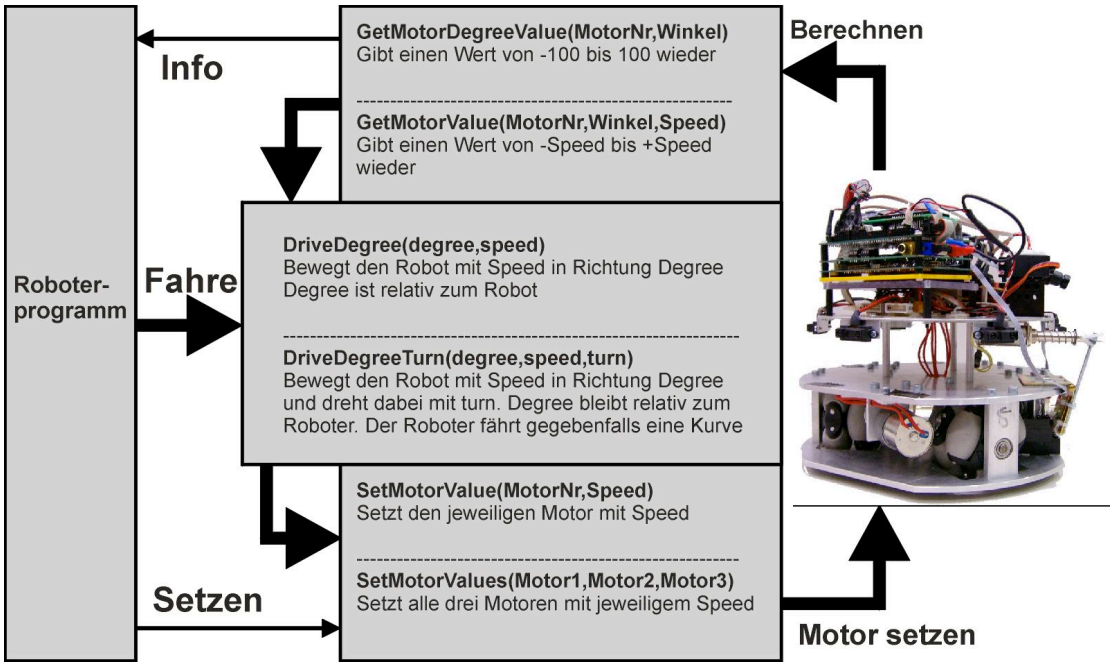

Abbildung 5.7: Schnittstelle zur Ansteuerung des omnidirektionalen Antriebes.

Die Schnittstelle für die Ansteuerung des Antriebs kann ich, da sie noch in der Entwicklungsphase ist, nicht in Gänze darstellen, aber ich werde hier eine vorläufige Version in Abbildung 5.7 darlegen.

6 Erweiterungen

Die in diesem Kapitel vorgeschlagenen Hard- und Softwareerweiterungen für die von Michael Ziener entwickelte Plattform können auch auf anderer Hardware zur Anwendung kommen und werden erst einmal plattformunabhängig dargestellt. Die Erweiterungen bestehen aus zwei Hardwareerweiterungen für eine Roboterfußballplattform, einer Erweiterung zur einfachen visuellen Positionsbestimmung eines Objektes, einer Erweiterung zum Testen und Trainieren von Robotersystemen und einem Konzept für eine Verschmelzung von Aktorwerten, um verschiedene Aufgaben leicht miteinander verbinden zu können und dadurch mit einfachen Mitteln ein komplexes Gesamtverhalten zu erlangen.

6.1 Visuelle Objektpositionsbestimmung

In diesem Kapitel werde ich zwei verschiedene Ansätze für eine Entfernungseinschätzung darstellen. Diese beiden Ansätze können, da sie die gleiche Grundlage an Sensordaten haben und damit kein Mehraufwand besteht, auch miteinander verbunden werden, um eine größere Genauigkeit zu erhalten.

6.1.1 Bestimmung der Entfernung durch die Y-Koordinate

Voraussetzung zur Bestimmung der Entfernung durch die Y-Koordinate innerhalb eines Kamerabildes ist, dass Objekt und Subjekt auf einer zweidimensionalen Ebene stehen und die Kamera vom Subjekt aus in einem nicht allzu kleinen Winkel auf die Ebene schaut, auf der sich das Objekt befindet. Die Ausrichtung der Kamera in Y-Richtung muss entweder fixiert oder bekannt sein. Bei der Projektion von einem

dreidimensionalen Sachverhalt auf ein zweidimensionales Kamerabild erhöht sich mit der Entfernung des Objektes vom Subjekt – durch den Blickwinkel bestimmt – der Wert der Y-Koordinate in der zweidimensionalen Darstellung.

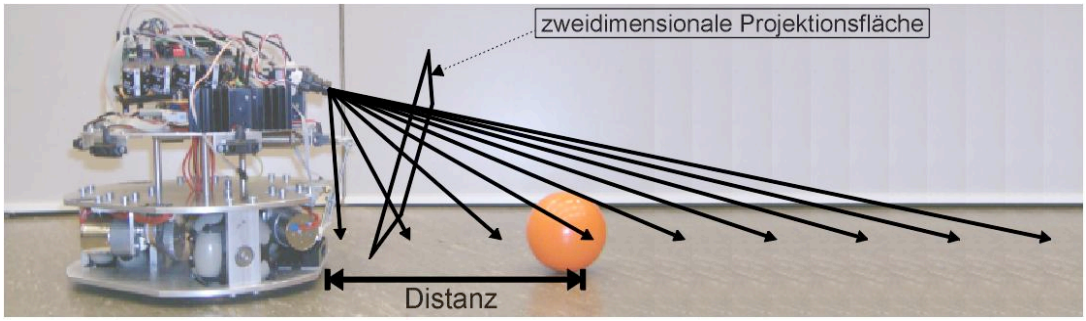

Abbildung 6.1: Entfernungsbestimmung eines Objektes anhand der Y-Koordinaten.

Wie in Abbildung 6.1 zu sehen ist nimmt die Bildauflösung in Bezug auf das Objekt mit der Distanz ab. Dies ist ein großer Nachteil dieser Methode, da die Entfernungseinschätzung bei großen Distanzen sehr unpräzise wird. Weiter entfernte Objekte können dann nur noch als entfernt klassifiziert werden, eine genauere Bestimmung ist hier nicht mehr möglich.

6.1.2 Bestimmung der Entfernung durch die Größe eines Objekts

Für diese Methode muss die eigentliche Größe des Objekts oder zumindest ein relatives Maß bekannt sein. Beim Roboterfußball wären dies z.B. der Ball oder das Tor. Beide dieser Größen sind bestimmt und damit bekannt. Nun können von der Erscheinung eines Objekts und der damit verbundenen Größe Rückschlüsse gezogen werden. Auch hier gilt, wie bei der Entfernungsbestimmung durch die Y-Koordinate, dass größere Entfernungen nicht mehr genauer bestimmt werden können, da die zweidimensionalen Größenveränderungen des Objektes mit der Entfernung soweit abnehmen, dass sie kaum noch unterscheidbar sind.

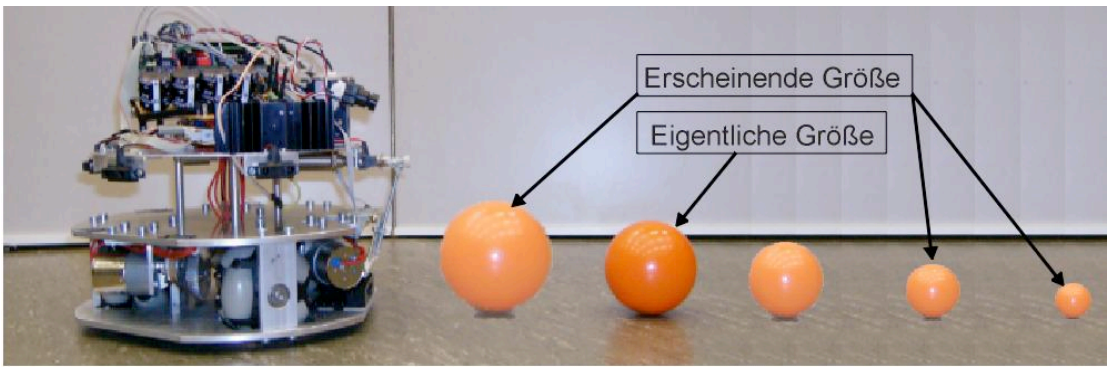

Abbildung 6.2: Entfernungsbestimmung durch Erscheinungsgröße.

6.2 Schießen des Balls

Das Beschleunigen des Balls in Richtung eines Ziels ist eine notwendige Voraussetzung, um den Ball ins Tor zu bekommen. Die einfachsten Möglichkeiten sind, den Ball mit dem Roboter ins Tor zu schieben oder ihn mit dem Roboter anzustoßen, so dass er ins Tor rollt. Um aber einen Lösungsweg zu beschreiten, der es möglich macht, den Wirkungsbereich, in dem der Ball bewegt werden kann zu vergrößern, bedarf es einer Schussvorrichtung, die den Ball wirklich beschleunigt.

Zwei schon im Einsatz gesehene und mir bekannte Schussprinzipien werde ich hier kurz aufzeigen und bewerten. Ein häufig genutztes Prinzip basiert auf einem Elektromagneten und einer Wippe. Ein weiteres auf der Drehbewegung eines Motors.

6.2.1 Schussmechanismus mit Wippe

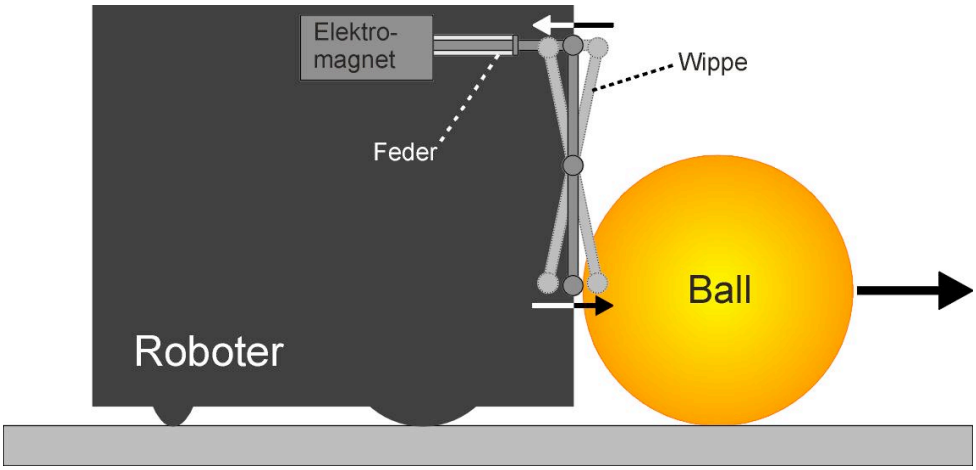

Abbildung 6.3: Schematische Darstellung einer Schussvorrichtung mit Wippe.

Bei der ersten der beiden Varianten wird im Moment des Schusses ein Metallkörper zu einem Elektromagneten beschleunigt. An diesem Metallkörper ist eine Wippe angebracht, die die entstehende Bewegung so umlenkt, dass das andere Ende der Wippe nach vorne stößt und damit den Ball beschleunigen kann. Diese Variante mit Wippe ist sehr vielversprechend, da sie innerhalb der Roboterproportionen angebracht werden kann und eine wirkliche Schussdynamik verspricht.

6.2.2 „Schussmechanismus“ mit Walze

Die zweite Variante besteht aus einem Motor und einer Gummiwalze. Hier befindet sich der Ball in direktem Kontakt mit der Walze. Die Walze wird dann plötzlich durch einen Elektromotor gedreht, so dass der Ball vom Roboter wegbeschleunigt wird. Diese Variante schlägt zwei Fliegen mit einer Klappe, denn sie lässt sich nicht nur zum Wegbeschleunigen des Balls vom Roboter verwenden, sondern auch zum Heranziehen. Dies geschieht durch die Umkehrung der Drehrichtung der Walze, so

dass der Ball auf den Roboter zu rollt. Dieser Mechanismus kann dann auch zum Dribbeln genutzt werden, denn der Ball klebt regelrecht am Roboter.

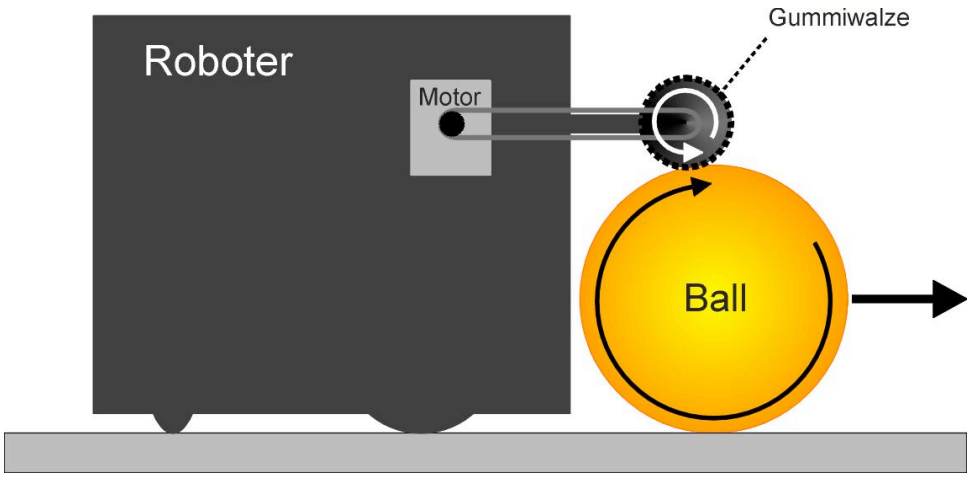

Abbildung 6.4: Schematische Darstellung einer Schussvorrichtung mit Gummiwalze.

Bei der Gummiwalzenvariante findet leider keine Beschleunigung durch einen Aufprall statt, so dass hier nicht von einem Schuss, sondern eher von einem Wegschieben die Rede sein kann. Dementsprechend fällt die Beschleunigung auch eher gering aus. Die Walze zum Dribbeln zu verwenden, ist allerdings ein sehr großer Nutzen. Wegen dieses großen Vorteils wurden in der Small-Size-League letztes Jahr die Regeln so abgeändert, dass ein dauernder Einsatz der Walze verboten wurde.

6.3 „Taktiler Fuß“

Die Schwierigkeit festzustellen, ob der Ball in der richtigen Position zum Abschuss liegt, lässt sich durch einen Kontaktsensor oder eine Lichtschränke lösen. Eine Lichtschränke müsste links und rechts von der Abschussvorrichtung angebracht werden und würde damit den Vorbau des Roboters stark beeinflussen. Der

Kontaktsensor benötigt einen kinetischen Impuls, welcher von einem leichten Ball nicht hervorgerufen werden würde und auch nur den Fall des direkten Kontakts abdeckt.

Eine Infrarot-Geber-Nehmer-Kombination[25] ermöglicht hier eine kontaktfreie und von der Innenseite des Roboters ausgehende Ermittlung der richtigen Abschussposition. Da die Farbe und der Reflexionswert des Balls sich im Groben nicht ändert, stehen hier sogar die Parameter für die Messung des reflektierten Infrarotlichts fest, und es braucht keine Anpassung stattfinden. Auch eine Referenzmessung[26] mit und ohne Infrarot-Geber muss nicht erfolgen, da es vom Gesamtsystem die Annahme darüber gibt, dass der Ball sich ungefähr in der Abschussposition befindet und er in diesem Bereich das Sichtfeld des Infrarot-Geber-Nehmer-Sensors komplett einnimmt und damit externe Lichtquellen im Groben verdeckt.

Erweitern wir diese Strategie um einen weiteren Infrarot-Geber-Nehmer, dann bekommen wir nicht nur einen eindimensionalen Wert, der die Entfernung des Balls von einem Punkt beschreibt, sondern einen zweidimensionalen Wert, der auch die Position innerhalb des gesamten Messbereichs wiedergibt. Dieser Wert kann dann mit einem PID-Regler für eine genaue Ausrichtung des Roboters am Ball oder für ein Ansaugverhalten des Roboters an den Ball genutzt werden.

[25] Ein Infrarot-Geber-Nehmer besteht aus einer Infrarotlichtquelle und einem Infrarotlichtsensor. Die Intensität des reflektierten Infrarotlichtes gibt Rückschlüsse darauf, wie weit ein Objekt entfernt ist. Hierbei sind die Reflexionseigenschaften des Objektes, wie Farbe und Oberflächenstruktur, zusätzlich ausschlaggebend für den Messwert.

[26] Bei einer Referenzmessung wird mit eingeschaltetem und ausgeschaltetem Infrarot-Geber gemessen. Die Differenz dieser beiden Messwerte ergibt dann den gewünschten Messwert. Mit diesem Verfahren werden weitestgehend Störeinflüsse durch andere Lichtquellen ausgeschlossen.

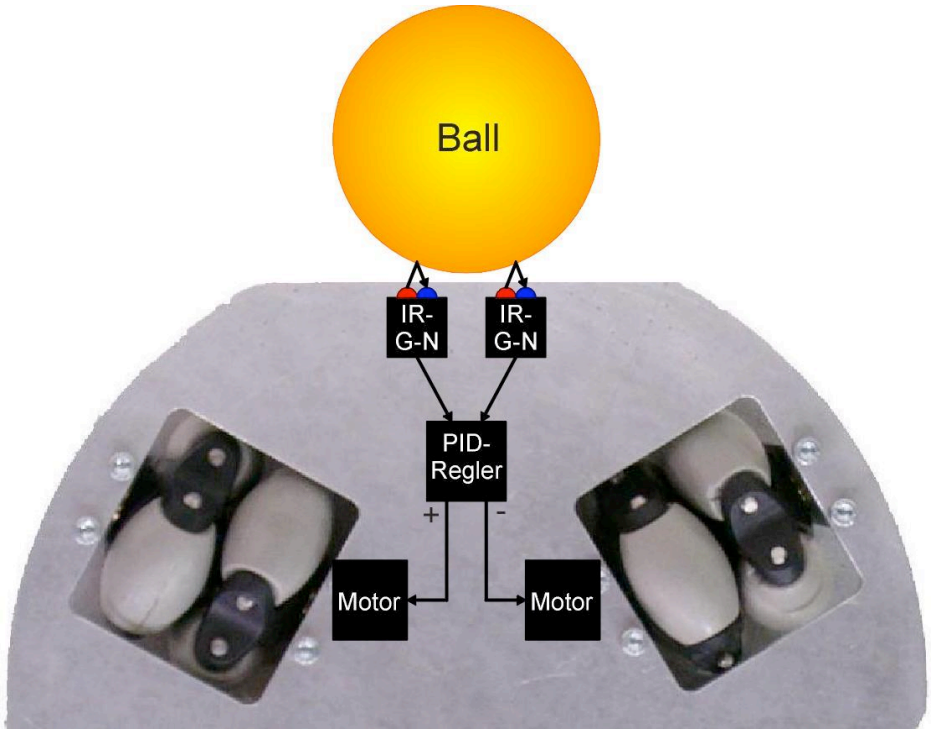

Abbildung 6.5: Schema für eine Implementierung der Ausrichtung des Roboters an einem Ball.

6.4 Manuelle Manipulation des Roboters

Um dem Programmierer in Bezug auf die Programmierung von Robotern die Möglichkeit zu geben, ein theoretisches Verhaltensmuster zu überprüfen oder innerhalb des Roboterverhaltens eine Manipulation der Rahmenbedingungen zu ermöglichen, ist es sinnvoll, einen manuellen Manipulator in das vorhandene System zu integrieren. Diese Integration hilft zusätzlich beim „Extremprogramming“, gibt dem Programmierer die Möglichkeit sich in das Verhalten des Roboter besser „einfühlen“ zu können und ermöglicht das Programmieren des Roboters durch Beispiele und Training.

Eigentlich können hier alle aus der „on-Line“ Programmierung bekannten Manipulatoren zum Einsatz kommen. Roboterspezifisch soll der Manipulator primär

die verschiedenen Bewegungsmuster des omnidirektionalen Antriebes nachstellen können. Weiterhin sind verschiedene auslösende Merkmale gefordert, und analoge Sensoren sollen simulierbar sein. Eine Manipulation des Kamerabildes ist hier nicht nötig, da es in einem Szenario durch den Programmierer real nachgestellt werden kann. Eine Testumgebung für die Entwicklung von Algorithmen im Bereich Computer Vision sollte ohnehin aus Effizienzgründen in einer komfortablen Entwicklungsumgebung und einer künstlichen Testumgebung stattfinden.

Eine Maus mit Rad entspricht den zuvor genannten Kriterien und ist sehr günstig und leicht zu beschaffen. Sie hat eine definierte Schnittstelle und kann über einen seriellen Port angeschlossen werden. Wenn die Maus direkt mit dem Roboter verbunden werden soll, ist allerdings ein Treiber für die Entschlüsselung der von der Maus gesendeten Daten nötig. Wenn der Roboter über Funk mit einem PC in Verbindung steht, können hier die schon aufgearbeiteten Daten der Maus einfach weiter geleitet werden.

Ein Joystick mit mehreren Freiheitsgraden entspricht ebenfalls den oben genannten Kriterien und kann mit einem PC verbunden werden. Da bei den meisten analogen Joysticks die analogen Signale direkt weiter geführt werden, kann der Joystick auch direkt mit den jeweiligen Analog- und Digitaleingängen des Kontrollers verbunden werden. Eine einfache Funktion zur Projektion der Eingangswerte ist in diesem Fall auf dem Kontroller zu implementieren.

6.5 Bewegungskoordinaton

In diesem Abschnitt werde ich eine Verfeinerung der Subsumtionsmethode und damit eine Parallelentwicklung zum Dual Dynamics Ansatz vorstellen. Da sich das Verhalten eines Roboters in unserem Fall durch Bewegung äußert, werde ich hier von einer Bewegungskoordination bzw. Bewegungsfusion sprechen und nicht von einer Verhaltenskoordination bzw. Verhaltensfusion wie sonst üblich. Das Wort Verhalten wird dann nur noch im abstrakten und nicht im konkreten Sinn benutzt.

Der hier gemachte Vorschlag basiert auf der Verbindung von Aktorwerten und der direkt damit verbundenen Bewegung bzw. dem Verhalten.

Die Ausprägung des Gesamtverhaltens eines Roboters basiert auf den Eingabewerten. Die Verhaltenausprägung selber wird über die Ausgabe an die Aktoren von der virtuellen in die reale Welt transformiert (siehe Abbildung 6.6).

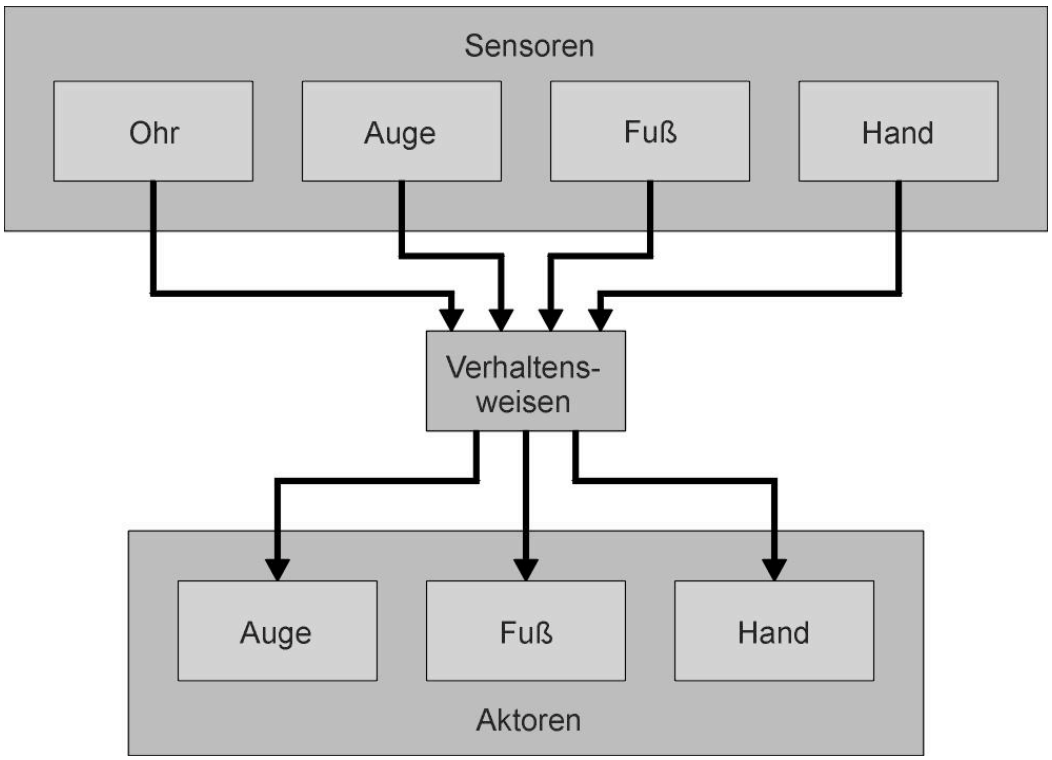

Abbildung 6.6: Die Umgebung bestimmt die Verhaltensweisen, welche wiederum die Aktion in der Umgebung bestimmt.

Wie bereits in Kapitel 3.6 erläutert, greift die Subsumtionsmethode nicht auf ein Weltmodell zurück, um das Verhaltensmuster des Roboters zu bestimmen, sondern passt das Verhaltensmuster und deren Berechnungen nur den unmittelbaren Ereignissen an. Die Subsumtionsmethode als Grundlage zu nutzen, soll hier aber keine Abgrenzung zum Weltmodellansatz bedeuten oder dass kein Weltmodell mehr benutzt werden soll, sondern nur, dass ein Weltmodell nicht mehr notwendige

Bedingung für ein Handeln darstellt. Ich beschränke mich hier auf den reaktiven Teil, der nicht ausschließt, dass auch, wie bei „Dual Dynamics“, die Interpretation eines Weltmodells einen Beitrag liefern kann. Die aus dem Weltmodell resultierenden Verhaltensmuster sollen sogar, wenn auch nicht in dieser Diplomarbeit, später in die Bewegungskoordination mit einfließen können.

Um eine Variabilität und Handhabbarkeit sicher zu stellen, trenne ich das Gesamtsystem der Verhaltensweisen wie in den „Dual Dynamics“ und der „Hybrid Control Architecture“[27] in zwei Teile. Der erste Teil, die Verhaltenskonkretisierung, beinhaltet die Auswertung der Sensoren von den dazu gehörigen Verhaltensmustern und die daraus resultierenden Ausgangsparameter. Der zweite Teil, die Bewegungssynthese, soll die Menge der in Teil Eins erzeugten Ausgangsparameter miteinander verbinden und ausgeben.

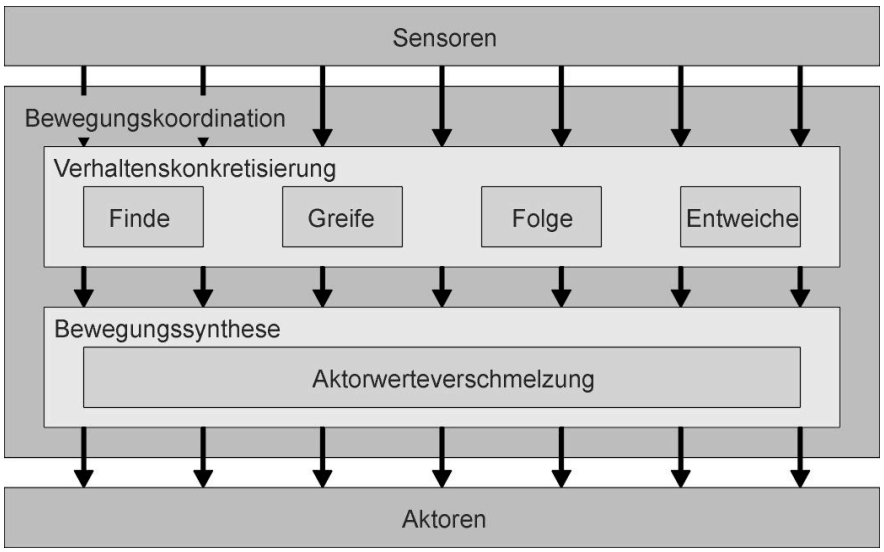

Abbildung 6.7: Die Bewegungskoordination teilt sich in Verhaltenskonkretisierung und Bewegungssynthese.

[27] „Hybrid Control Architecture“ wird in [JongHwan04] in Kapitel 4.2 beschrieben und hat ähnliche Merkmale wie „Dual Dynamics“.

6.5.1 Verhaltenskonkretisierung

Bei der Verhaltenskonkretisierung kann, wie bei der Subsumtionsmethode, jedes Verhaltensmuster erst einmal für sich selbst betrachtet, entwickelt und genutzt werden. Um Ressourcen im Gesamtkontext zu schonen, muss jedes Verhaltensmuster anhand seiner Eingangsvariablen selber entscheiden, ob es aktiv sein soll oder nicht. Die Eingangsvariablen können hier aus den Sensorwerten und aus Werten, die durch andere Verhaltensmuster erzeugt werden, bestehen. Dies ermöglicht das Teilen von Ressourcen durch eine Kommunikation über die Ereignisvariablen zwischen den Verhaltensmustern und auch eine Strukturierung von Verhaltensmustern und Submustern. Damit ist ein Netzwerk von Verhaltensmustern möglich.

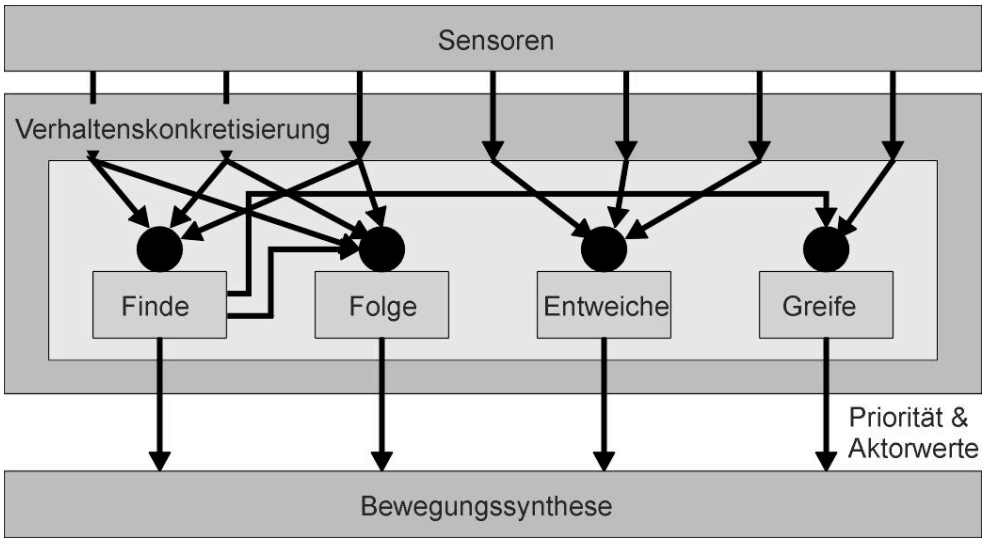

Abbildung 6.8: Die Verhaltenskonkretisierung beinhaltet die verschiedenen Verhalten.

Die Ausgangsparameter der Verhaltenskonkretisierung beinhalten neben den Parametern für die Aktoren und den Ereignisvariablen auch die Prioritätswerte, die für die Gewichtung der Bewegungsparameter wichtig sind. Das heißt, dass jedem Verhalten die Möglichkeit gegeben wird, selber zu entscheiden, wie wichtig das

eigene Verhalten innerhalb des Gesamtverhaltens ist. Die Möglichkeit der Selbsteinschätzung kann zu einer Selbstkalibrierung des Gesamtsystems führen.

Es gibt zwei Fälle, in dem sich das Prioritätsgefüge des Systems bis zum Ende aufschaukelt. Erstens: Wenn die Situation nicht oder schlecht lösbar ist, werden wegen des ständigen Misserfolges die einzelnen Verhalten sukzessive ihr eigenes Prioritätsniveau steigern, bis alle das höchste Niveau erreicht haben. Zweitens: Wenn z.B. zwei Verhalten so miteinander konkurrieren, dass das Heraufsetzen der Priorität des einen dazu führt, dass sich das andere auch erhöht und umgekehrt. Um diese beiden Fälle auszuschließen ist es ratsam, den jeweils wählbaren Prioritätsbereich sinnvoll einzuschränken. Ein weiterer und wesentlich wichtigerer Grund für das Einführen eines Prioritätenfensters ist, dass nicht jede Verhaltenausgabe in der Bewegungssynthese in allen Prioritätsstufen sinnvoll verwendbar ist.

6.5.2 Bewegungssynthese

Bei der Subsumtionsmethode würde die aus der Verhaltenskonkretisierung entstandene Verhaltensausprägung mit der höchsten Prioritätsstufe vom Arbitrator ausgewählt und ausgeführt werden, und alle anderen Verhaltensausprägungen würden einfach ignoriert werden. Die These, dass zwei konkurrierende Verhaltensweisen sich nicht ausschließen müssen, führt zu der Idee einer sogenannten Verhaltensverschmelzung. Da sich in unserem Fall das Verhalten durch eine Bewegung ausprägt, erfolgt im Konkreten bei der Bewegungssynthese eine Bewegungsmusterverschmelzung. Ich spreche hier nicht mehr vom Arbitrator, der entscheidet, sondern vom Moderator, der verbindet. Da verschiedene Situationen unterschiedliche Bewegungsmusterverschmelzungen benötigen, wird eine in mehrere Bereiche eingeteilte Prioritätenliste konstruiert. Die Prioritätsbereiche bestimmen die Art und Weise der Bewegungssynthese. Der Prioritätswert bestimmt die Dominanz des jeweiligen Bewegungsmusters innerhalb des Bereiches.

Mein Vorschlag ist, eine Aufteilung der Prioritätenliste in die fünf Teile vorzunehmen: Null-Bereich, Aspekt-Exklusiv-Bereich, Aspekt-Additiv-Bereich, Additiv-Bereich, Exklusiv-Bereich.

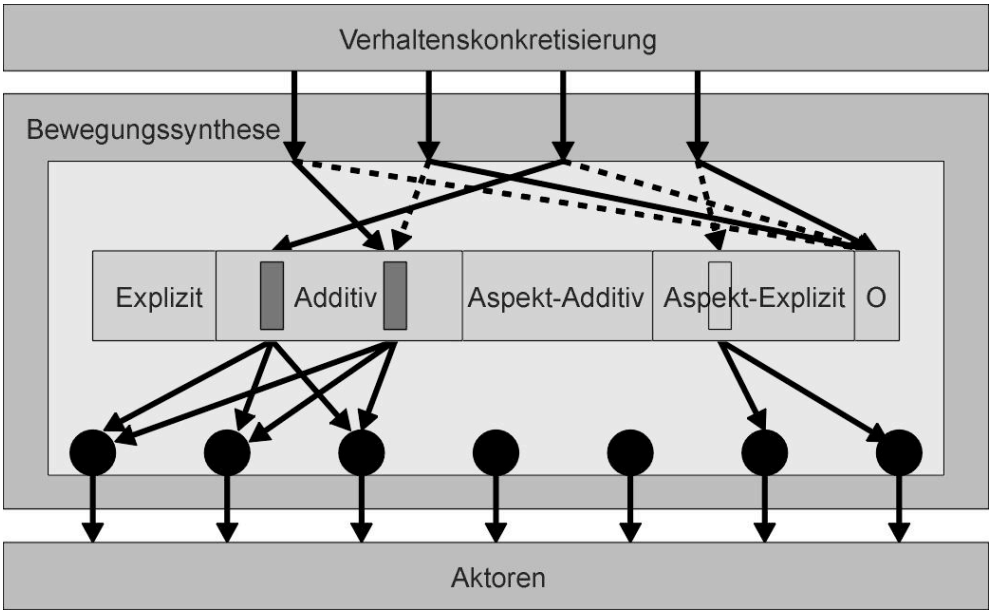

Abbildung 6.9: Die Bewegungssynthese arbeitet mit verschiedenen Prioritätsbereichen.

Der Exklusiv-Bereich wird in einer Notsituation benötigt oder von einem Verhalten, welches nur alleine, also exklusiv, eingesetzt werden kann. Dieser Bereich könnte auch Tunnel-Blick-Bereich genannt werden, da nur das Verhalten zum Tragen kommt, welches die höchste Priorität hat. Alle anderen Verhalten dieses und anderer Prioritätsbereiche werden in diesem Fall komplett ignoriert.

Der Additiv-Bereich ist wohl der interessanteste, da hier jedes aktive Verhalten einen anteiligen Beitrag zum Gesamtverhalten beisteuert und damit die größte Verhaltensdynamik und -komplexität entsteht. Realisiert werden kann diese Synthese durch eine zur Priorität proportionale Anteilsvergabe der jeweiligen Verhaltensausgabe.

Abbildung 6.10 zeigt ein Beispiel zur Demonstration der großen Bedeutung des Additiv-Bereichs. Gegeben sei, dass ein Verhalten anstrebt, mit einer konstanten Geschwindigkeit zum Ziel zu gelangen. Ein anderes Verhalten ist bestrebt einem Hindernis zu entkommen. Das sich bei Annährung an das Hindernis steigernde Fluchtverhalten kann entweder durch Ändern der Fluchtgeschwindigkeit oder durch Steigern der Priorität erreicht werden. Nähert sich der Roboter auf dem Weg zum Ziel einem Hindernis, so wird durch die anteilige Addition der Aktorwerte - und damit der Addition der Zielvektoren - der Zielvektor automatisch so verändert, dass sich der Roboter im Bereich des Hindernisses automatisch einen Weg um das Hindernis bahnt. Das Hindernis wirkt hier wie ein negativer Magnet, der den Roboter abstößt.

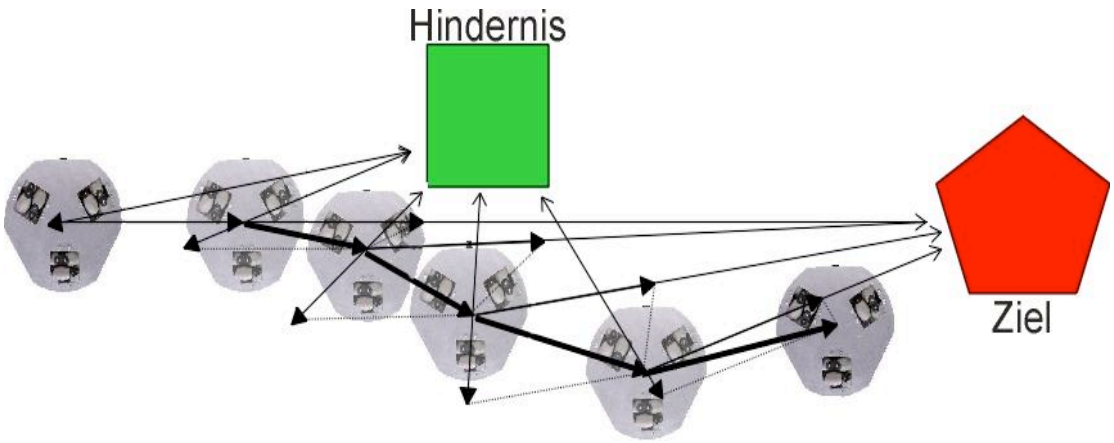

Abbildung 6.10: Durch die Addition des Fluchtvektors und des Zielvektors entsteht ein Ausweichverhalten.

Der Aspekt-Additiv-Bereich funktioniert genauso wie der Additiv-Bereich, nur dass sich die Addition der Aktorwerte nicht auf alle Aktorwerte bezieht, sondern nur auf ausgewählte. Ausgewählt wird mit einer Maske, die beschreibt, auf welche Aktoren sich das Verhalten auswirken soll. Als Beispiel nehmen wir einen Humanoid-Roboter, bei dem das Verhalten „WerfeBall“ aktiviert wurde. Nun soll sich die Ausgabe dieses Verhaltens allerdings nur auf den linken Arm beziehen und das

Laufverhalten soll explizit nicht beeinflusst werden. Für diesen Fall soll der Aspekt-Additiv-Bereich die gewünschte Funktionalität bieten.

Der Aspekt-Exklusiv-Bereich ist eine Mischung aus den beiden zuerst genannten Bereichen. Hier soll allerdings nur ein Teil des Roboters explizit manipuliert werden. Wieder mit einer Maske werden die zu manipulierenden Aktoren ausgewählt. Ein Beispiel wäre, dass der Roboter auf ein Tor schießen will. Nun ist es wichtig, dass dies genau in dem Augenblick passiert, in dem das Verhalten „Schießen“ dies wünscht und dieser Vorgang nicht von einem anderen Verhalten verzögert oder abgeschwächt wird.

Der Null-Bereich ist der Pause-Bereich der Verhalten. Die sich hier verortenden Verhalten wollen keinen Einfluss auf das Gesamtverhalten nehmen und geben dies mit der Priorität Null an.

Der Moderator hat nun die Aufgabe, möglichst in zeitlich kleinen Zyklen immer wieder die Liste der Verhaltensweisen abzuarbeiten und die jeweiligen Aktorwerte zuzuordnen und gegebenenfalls zu verbinden. Diese Programmkomplexität kann bei geschickter Programmierung auch auf einem Mikrokontroller laufen, wie sie auf dem AkSen- oder 6.270-Board zu finden sind. Die Komplexität besteht aus genau O(N)=N*Aktoren*Verhaltensweisen. Wenn nun ein Verhalten eine möglichst kleine Zykluszeit benötigt, muss dieses Verhalten mit dem Explizit-Bereiche die anderen unterdrücken und die Zykluszeit damit abkürzen.

Verhalten, die zuvor in einer Subsumtionsarchitektur zum Einsatz kamen, können modifiziert weiter genutzt werden. Diese müssen die Aktorenausgabe und zusätzlich die eigene Priorität an den Moderator weiterleiten und nicht mehr an den Arbitrator. Die hier vorgeschlagene Bewegungssynthese ist sozusagen abwärtskompatibel zur Subsumtionsmethode.

Ein weiterer sehr interessanter Aspekt ist, dass sehr einfach und effektiv eine dynamische Verhaltensanpassung über das Ändern der Priorität durch das Verhalten selbst erfolgen kann. Das heißt, wenn die Parameter für jedes einzelne Verhalten gut funktionieren, kann sich das Verhalten ein Referenzmuster generieren. Weicht nun das eigentliche Muster von dem Referenzmuster ab, so

kann davon ausgegangen werden, dass andere Verhalten das Gesamtverhalten beeinflussen. Überschreitet nun die Differenz der beiden Muster eine vorher definierte Grenze, so kann das Verhalten seine Priorität innerhalb des eigenen Prioritätsfensters erhöhen und damit mehr Einfluss auf das Gesamtverhalten erlangen, um die Musterdifferenz wieder in einen annehmbaren Bereich zu bringen. Wenn die Musterdifferenz sehr klein ist, so kann ein Verhalten auch seine Priorität innerhalb des Fensters verringern, so dass anderen Verhalten mehr Einfluss auf das Gesamtverhalten gegeben werden kann. Wie bei einer digitalen Regelung muss ein Überschwingen oder ein Aufschwingen durch eine Dämpfung verhindert werden. Auch Resonanzen zwischen den Verhalten müssen hier beachtet werden.

7 Implementierung der Erweiterungen

Bei der Implementierung der zuvor vorgeschlagenen Erweiterungen handelt es sich um Prototypen. Diese können, so wie sie implementiert wurden, genutzt und weiterentwickelt werden. Das Ziel bei der Implementierung war eine Erweiterung und Verbindung der vorhandenen Teile der Diplomarbeiten von Lars Brandt und Michael Ziener. Die Auswahl der Erweiterungen war bedingt durch die Möglichkeit, das eigene Konzept und das der beiden anderen Diplomarbeiten zu bestätigen und das Roboterlabor der HAW Hamburger einen Schritt weiter in Richtung RoboCup-Fußball zu bringen.

7.1 Schussmechanismus

Bei der Implementierung entschied ich mich für die vorgeschlagene Variante mit Elektromagneten. Bei dieser Variante konnte ich fast destruktionsfrei mit der Plattform von Michael Ziener umgehen und die Außengrenzen der Plattform weitgehend bewahren. Der Fakt, dass diese Plattform parallel von Michael Ziener entwickelt wurde, bestärkte mich in der Entscheidung für minimale, wieder rückgängig zu machende, Änderungen an der entstehenden Plattform.

Der Schussmechanismus besteht aus einem Scharnier, einer Gewindestange, einem Elektromagnet-Schießmechanismus, einigen Muttern und einer Kugelschreiberhülse. Für die Schusselektronik konnten die Motortreiber des AkSen-Boards nicht genutzt werden, da sie theoretisch zwar ca. 5 V bei 1 Ampere

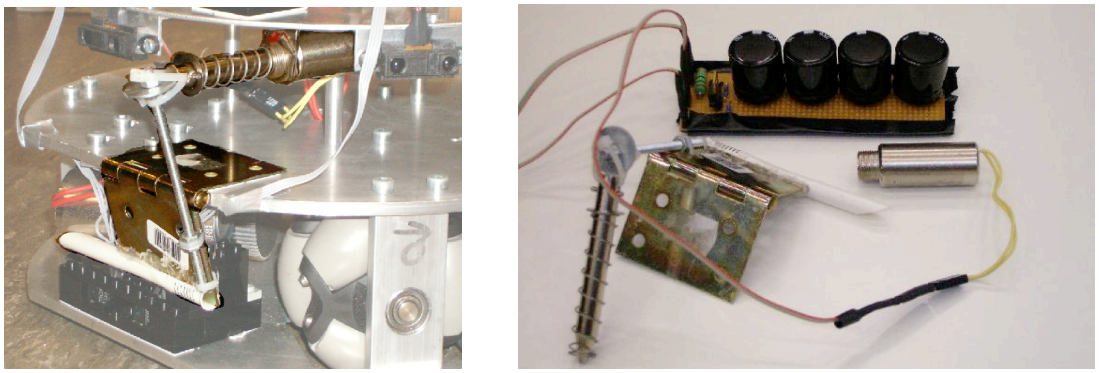

Abbildung 7.1: Links: Der am Roboter installierte Schussmechanismus. Rechts: Die Einzelteile inklusive der Platine.

lieferten, praktisch jedoch wesentlich weniger. Die neu entwickelte Schaltung beinhaltet eine Reihe von Kondensatoren, die mit einer höheren Spannung als 6 Volt über einen Vorwiderstand geladen werden. Die geladene Leistung wird dann über einen Aktorenausgang des AkSen-Boards sowie einen Leistungstreiber an den Elektromagneten freigegeben. Bei der Dimensionierung der Bauteile wurde darauf geachtet, dass eine leistungsstarke Ladung nach spätestens 2 Sekunden erfolgt und abrufbar ist. Um Kosten zu sparen, wurden wenn möglich elektronische Sonderposten und Baumarktartikel verwendet. Die Kosten für Wippe und Schaltung liegen um und bei 15,- Euro.

7.2 „Taktiler Fuß“

Der „taktile Fuß“ wurde grundsätzlich wie vorgeschlagen implementiert. Der Abstand der beiden Infrarot-Geber-Nehmer-Kombinationen wurde so angepasst, dass bei der Differenz der beiden Messwerte kein Sattel entsteht, also bei der Funktion der ersten Ableitung der Funktion: F(IR1,IR2)=IR1-IR2 nur Werte größer als Null generiert werden. Da sich der Messbereich im Abstand von null bis drei cm vor dem Sensor befindet, kommen als IR-G-N-K leistungsschwache Optoreflexkoppler (CNY 70) zum Einsatz. Als Regler wurde ein PD-Regler gewählt. Da diese Regelschleife

durch die programmtechnisch bedingte Verzögerung und die mechanische Trägheit des Roboters zum Überschwingen neigt und keine Störungen im Messergebnis zu erwarten sind, wurde auf den I-Anteil im PID-Regler verzichtet. Der D-Anteil wurde beibehalten, um das bereits erwähnte Überschwingen zu reduzieren.

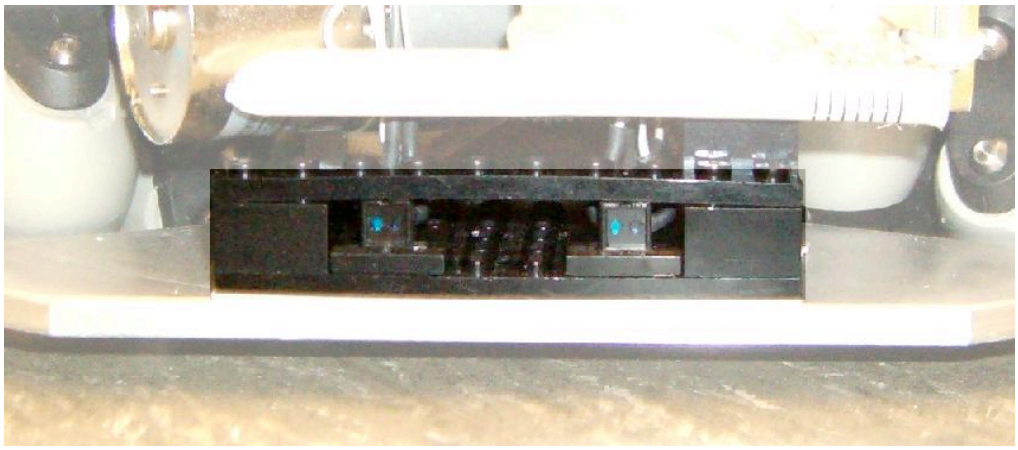

Abbildung 7.2: Zwei Optokoppler ermitteln beim „taktilen Fuß“ die Position des Balls in unmittelbarer Nähe.

7.3 Manuelle Manipulation

Zum Testen der Plattform und des Zusammenspiels der verschiedenen Verhaltensweisen war es notwendig, einen manuellen Manipulator zu implementieren. Die Wahl eines Joysticks war naheliegend, weil nur die jeweiligen Ausgänge des Joysticks mit den Analog- oder Digitaleingängen des AkSen-Boards verbunden werden mussten. Die Implementierung des Treibers für den Joystick beinhaltet das Auslesen von vier Digital-Buttons und drei Analog-Achsen

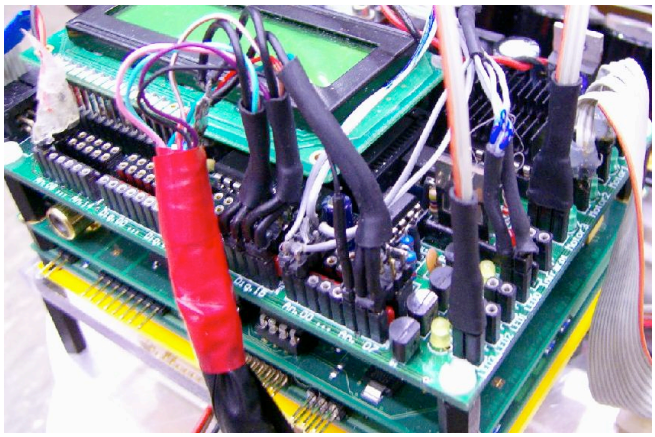

Abbildung 7.3: Die digitalen und analogen Ausgänge des PC-Joysticks werden einfach mit den Sensoreingängen des AkSen-Boards verbunden.

(X, Y, Drehen). Angeschlossen werden kann jeder PC-Joystick, da nicht der Joystick selber adaptiert wurde, sondern ein PC-Joystick-Anschlusskabel.

Die Schnittstelle zum Joystick besteht aus drei Umrechnungsfunktionen. Die Eingabewerte für diese Funktionen sind die vom Joystick gelieferten Analogwerte.

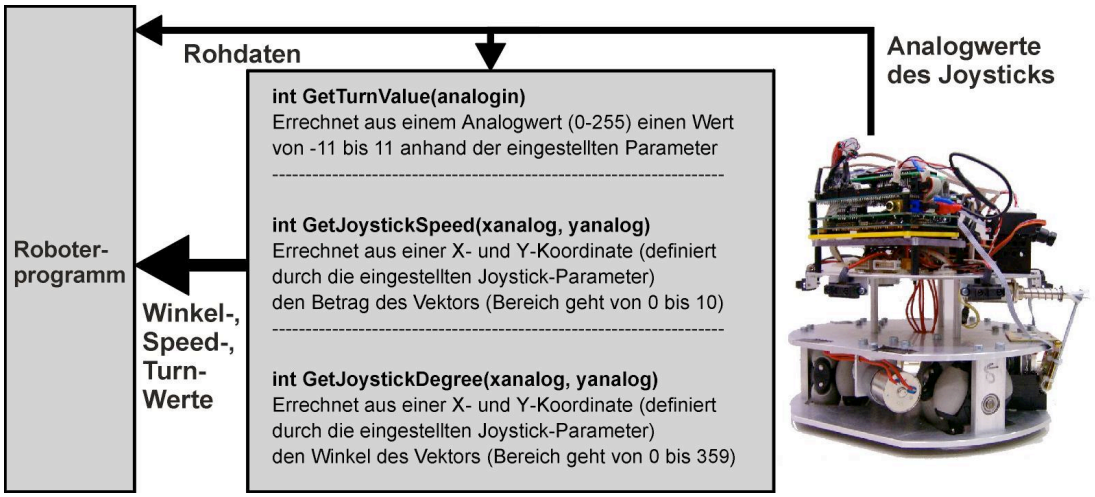

Abbildung 7.4: Schnittstelle der Joystickfunktionen.

7.4 Visuelle Objektpositionsbestimmung

Weil die Schnittstelle von Lars Brandt nur eine X- und Y-Koordinate liefert und nicht die Größe eines Objektes, und es derzeitig keine definierte Schnittstelle zur Erweiterbarkeit der Funktionalitäten der Serviceschicht gibt, habe ich mich auf die Methode der Entfernungseinschätzung durch die Y-Koordinate beschränkt.

Die Kamera wurde hier, wie in Abbildung 7.5 gezeigt, an dem Roboter angebracht. Die obere Sichtfeldkante soll tendenziell am Horizont enden, damit Störungen durch nicht relevante Umgebungsereignisse vermieden werden und die untere Sichtfeldkante möglichst dicht an den Roboter geholt wird. Dadurch, dass der Roboter mit einem omnidirektionalen Antrieb ausgestattet ist, ist es nicht nötig, die Kamera horizontal ausrichten zu können. Der Roboter richtet sich anhand der X-Koordinaten an dem Objekt aus und ist dann weiterhin frei beweglich.

Die Entfernungswerte werden anhand von empirischen Daten von der Y-Koordinate auf einen ungefähren Entfernungswert projiziert.

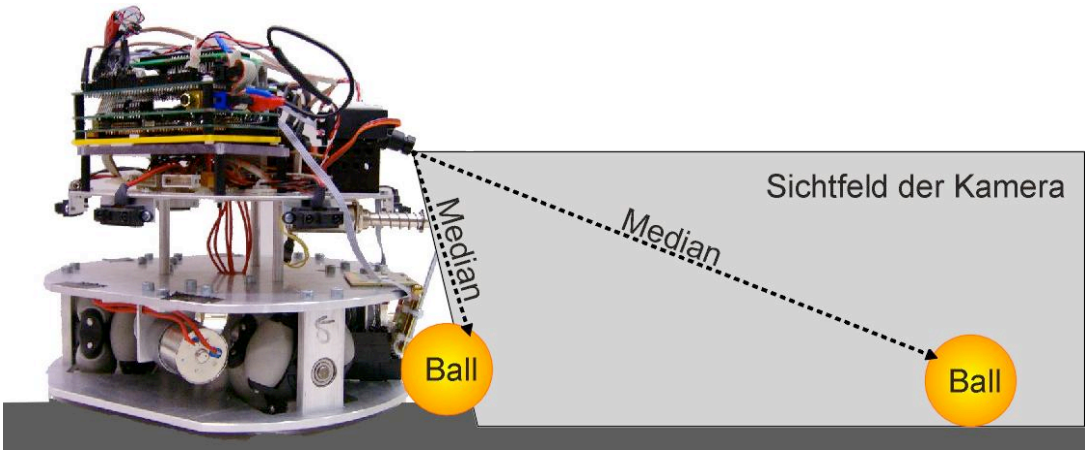

Abbildung 7.5: Installation der Kamera.

7.5 Bewegungskoordination

Bei der Implementierung der Bewegungskoordination habe ich darauf geachtet, das Konzept in seiner Idee beizubehalten, aber die Umsetzung möglichst schlicht und einfach zu gestalten. Grund dafür ist, dass ich es als wichtig erachte, dass der Prototyp durchschaubar und verstehbar bleibt, und folglich ein Aufsetzen durch andere Studenten einfach umzusetzen ist und diese nicht zu einer Neuimplementation verleitet werden. Außerdem läuft ein schlichtes Programm stabiler und zeigt ein besseres Laufzeitverhalten.

Die Implementierung der Bewegungskoordination teilt sich, wie im Konzept beschrieben, in die Teile Verhaltenskonkretisierung und Bewegungssynthese auf. Bei der Verhaltenskonkretisierung habe ich die Verhalten oder Subverhalten implementiert, die die jeweiligen Hardwareerweiterungen nutzen und gleichzeitig ausreichend sind, um den in der Bewegungssynthese implementierten Moderator und das dazugehörige Konzept zu testen.

7.5.1 Verhaltenskonkretisierung

Ich habe fünf Verhaltenskonkretisierer implementiert. Drei davon wurden auf dem AkSen-Board implementiert, da die beanspruchten Rechnerressourcen sehr gering sind und die an das AkSen-Board angeschlossenen Sensoren für die Verhaltenskonkretisierung benötigt werden. Ein Übertragen der Sensordaten über den CAN-Bus zum Lart-Board würde in diesem Fall keine Rechnerleistung schonen. Zwei von ihnen sind Subverhalten und werden nur aktiv, wenn ein Verhalten auf dem Lart-Board diese aktiviert.

Die zwei Verhaltenskonkretisierer auf dem Lart-Board benutzen hauptsächlich die Kamera, um Entscheidungen über Bedingungen für andere Verhalten zu treffen und visuomotorische Eigenschaften des Roboters zu konkretisieren.

Das „Ballfinder“ Verhalten nutzt die Funktion „FindeTor“ von Lars Brandt mit den Farbvalenz-Parametern des Balles. Die X- und Y-Rückgabewerte führen zu einer Bewegung zum Ball, und wenn der Ball erreicht ist, zum Setzen der Ereignisvariable „HabeBall“ für die Verhaltenskonkretisierer „TorFinder“ und „TaktileBall“. Das „TorFinder“ Verhalten versucht sich unter Berücksichtigung der veränderten Robotereigenschaften und mit Hilfe der Kamera, an dem Tor auszurichten. Ist dies der Fall, wird die Ereignisvariable „ZielTor“ gesetzt.

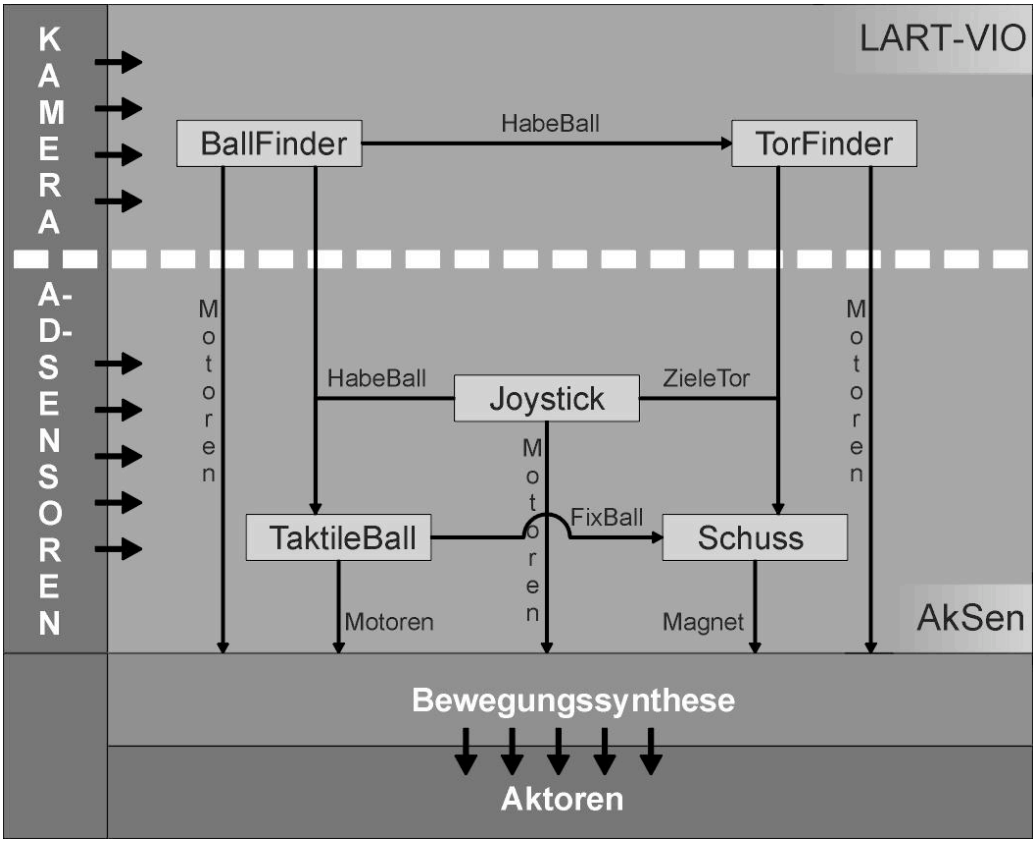

Abbildung 7.6: Zusammenspiel der implementierten Verhaltenskonkretisierer.

Die Verhalten „Schuss“ und „TaktileBall“ werden nur aktiv, wenn die jeweiligen Ereignisvariablen „HabeBall“, „ZielTor“ oder „FixBall“ gesetzt sind. Das Verhalten „Joystick“ konkretisiert die jeweiligen Bewegungswerte und setzt gegebenenfalls die

entsprechenden Ereignisvariablen für andere Verhaltensweisen. Es ist also das Verhalten, welches sich durch den Programmierer direkt manipulieren lässt.

7.5.2 Bewegungssynthese

Die Bewegungssynthese habe ich exemplarisch implementiert. Der Moderator unterscheidet nur die Bereiche „Explizit-Bereich“, „Additiv-Bereich“ und „Null-Bereich“. Wenn die Umsetzung der Verhalten diffiziler gestaltet werden soll, muss der Moderator um die beiden anderen Bereiche erweitert werden. Die beiden Aspekt-Bereiche können auch durch eine Prioritätsverschiebung oder zur Not auch mit dem Explizit-Bereich nachgebildet werden.

Ansonsten habe ich mich genau an die zuvor dargestellte Funktionsweise gehalten. Die implementierten Verhalten siedeln sich alle im Additiv-Bereich an, bis auf das Schuss-Verhalten, welches eigentlich im Explizit-Additiv-Bereich angesiedelt werden würde. Da dieser nicht implementiert wurde, habe ich mich entschieden, dieses Verhalten in den Explizit-Bereich zu setzen, da es entweder Schießen oder nicht Schießen heißt und dieser Vorgang weniger als eine Sekunde beansprucht.

8 Fazit

8.1 Bewertung der Ergebnisse

Bei dieser Arbeit ist ein Joystickanschluss, ein „taktiler Fuß“, eine Schussvorrichtung und eine verhaltensbasierte Bewegungskoordination für mobile Roboter entstanden. Diese Innovationen habe ich mit den Ergebnissen der beiden Diplomarbeiten von Lars Brandt und Michael Ziener verbunden. Das Ergebnis ist eine Roboterplattform mit visuomotorischen und omnidirektionalen Bewegungseigenschaften, auf dem leicht neue Verhaltensweisen in das Gesamtsystem integriert werden können.

Der Treiber für den Joystickanschluss, an den verschiedene PC-Joysticks angeschlossen werden können, läuft auf dem AkSen-Board und ist durch das Errechnen der Vektoren auf der Basis von Integerzahlen nicht wirklich präzise. Diese Methode ist jedoch präzise genug, um den Roboter galant in alle Richtungen bewegen zu können. Durch Nutzung eines Joysticks zum Testen von Programmen oder Robotereigenschaften kann auf einfache Weise verhindert werden, dass der Programmierer auf Grundlage von vagen Hypothesen sein Programm konstruiert oder verifiziert. Durch die Implementierung eines manuellen Manipulators ist es nun auch möglich, den Roboter (z.B. mit Hilfe von neuronalen Netzen) zu trainieren.

Der „taktile Fuß“ ist eine sehr gute Erweiterung für einen Fußball spielenden Roboter und zeigt gleichzeitig, wie einfach mit zwei Sensoren und einem PD-Regler große Probleme gelöst werden können. In der Vergangenheit war häufig die Frage „wie halte ich den Ball am Roboter“ und „ist der Ball in der richtigen Schussposition“. Die Lösungen waren Arme oder Ringe zum Festhalten des Balles bzw. eine rückwärts drehende Gummiwalze, die den Ball Richtung Roboter beschleunigt. Diese Methoden wurden jedoch nach und nach in den Regeln des RoboCups

verboten oder eingeschränkt. Außerdem würde kein Mensch diese beiden Verfahren mit dem Wort „Ballgefühl“ assoziieren. Neuerdings gibt es Teams, die eine Art „Ballgefühl“ implementiert haben, aber es sind noch sehr wenige. Wir können hier also, wenn auch mit einfachen Mitteln implementiert, von „State of the Art“ oder Innovation sprechen.

Die vorgenommene Implementierung der Schussvorrichtung ist wohl nicht mehr wirklich aktuell,[28] auch wenn diese immer noch von vielen Teams mit ähnlicher Robotergrundkonstruktion im RoboCup benutzt wird. Sie ist günstig und einfach zu implementieren und erzeugt ein dynamisches Schussverhalten. Wirft man einen Blick auf die Small-Size-League, so stellt man fest, dass dort mittlerweile so hart geschossen wird, wie es mit dieser Methode nicht umsetzbar wäre.

Die Funktionalität der in dieser Diplomarbeit konzipierten und teils implementierten Bewegungskoordination ist positiv bestätigt worden. Die hier angewendete Verbindung von Bewegungsmustern in der Bewegungssynthese ist sehr einfach und schnell zu implementieren sowie ressourcenschonend. Einfach ist ebenfalls die Implementierung weniger Verhaltenskonkretisierungen. Komplexer wird die Implementierung und Verifizierung von Verhaltenskonkretisierungen bei einem Ansteigen der Anzahl der Konkretisierer. Dadurch, dass die Wechselwirkungen und Überlagerungen vieler Konkretisierungen die Komplexität des Gesamtsystems stark steigert, müsste bei einer komplexen Nutzung der Bewegungskoordination eine Methode zur Vereinfachung und Strukturierung eingeführt werden. Ansonsten hat die Bewegungskoordination gezeigt, dass dividierbare Verhaltensweisen leicht und redundanzfrei implementierbar sind.

Ebenfalls hat sich gezeigt, dass die Plattform von Michael Ziener stabil und in ihren Parametern ausgewogen konzipiert ist. Eine Odometrie zur Ansteuerungsverbesserung der Motoren ist nicht implementiert, aber auch nicht

[28] Aktuell bedeutet in diesem Zusammenhang, dass diese Idee wohl älter als 3 Jahre ist. Jedes Jahr gibt es beim RoboCup Innovationen, die häufig dazu führen, dass das innovativere Team gewinnt. Sinnvolle Innovationen werden innerhalb von 1 bis 2 Jahren von anderen Teams assimiliert.

unbedingt nötig, da das Ist- und Sollverhalten der mechanischen Plattform so gut übereinstimmt, dass genügend Spielraum für kleine Fehler bleibt. Der Objekttracker von Lars Brandt hat sich unter den richtigen Bedingungen und mit den richtigen Parametern als sehr dynamisch und leicht integrierbar erwiesen. Während der Entwicklungsphase fehlte leider eine gut definierte Schnittstelle zur Erweiterung des Objekttrackers, so dass dieser nur wie übergeben zur Anwendung kam. Ein einfaches Programm zur Ermittlung der Farbparameter wurde jedoch implementiert.

Ob nun die mechanische Plattform, die visuelle Interpretation wie auch die Bewegungssynthese und die Verhaltensmodule genutzt und weiterentwickelt werden, wird sich in den nächsten Semestern zeigen.

8.2 Ausblick

Bisher wurde an der HAW Hamburg Roboterfußball ähnlich wie in der Junior-Soccer-League gespielt. Die neue Plattform ermöglicht den Studenten nun, einen Roboter in kleinen Schritten im Hinblick auf die Middle-Size-League weiterzuentwickeln und die Strukturen und Ansätze dieser Liga zu nutzen. Dadurch entsteht die Möglichkeit, sich mit eben dieser Liga zu messen.

Konkret sollten die Omniräder wie vorgeschlagen ausgetauscht und besser fixiert werden, um eine präzisere Umsetzung von Motoransteuerung und wirklicher Bewegung zu erreichen.

In Bezug auf die Kameranutzung stellte sich heraus, dass leicht veränderte Lichtverhältnisse eine Änderung der Toleranzbereiche der Farbparameter erforderten. Um nicht mühsam immer wieder die Farbparameter zu eruieren und neu einzugeben, wäre eine automatische Kalibrierung dieser Werte wünschenswert. Dies könnte z.B. mit einer Analyse des Bildes auf existierende Farbebenen oder Farbplateaus und einer Kalibrierung an diesen geschehen. Damit könnten auch die Toleranzbereiche und die Fehleranfälligkeit verringert werden.

Für die visuellen Eigenschaften sollten klare Schnittstellen im Sinne eines Schichtenmodells für die verschiedenen Ebenen der Computer Vision definiert und implementiert werden. Damit ist eine parallele und sukzessive Erweiterung des an der HAW Hamburg entstandenen „Know-Hows“ in diesem Bereich möglich.

Auch wenn die implementierte Bewegungssynthese sinnvoll nutzbar ist, sollte diese um die im Konzept vorgeschlagenen Prioritätsbereiche erweitert werden, um komplexere Verhaltensgraphen unterstützen zu können. Leider ist die hier entwickelte Bewegungskoordination nur für radgetriebene Plattformen bei einer einfachen und direkten Umsetzung der Bewegungssynthese nutzbar. Für einen Humanoid-Roboter müsste bei der Bewegungssynthese entweder bei der Antriebsansteuerung oder bei der Synthese selber eine Abstraktionsebene eingeführt werden. Diese Steigerung der Komplexität in der untersten Ebene würde aber eventuell die Ressourcen eines Kontrollerboards überfordern, so dass zudem eine Auslagerung der Synthese nötig ist.

Die Denkweise in dieser Diplomarbeit, Probleme parallel und gleichberechtigt zu betrachten und die Lösungen den Umständen entsprechend zu verschmelzen, kann auch in anderen Bereichen zu neuen unerwarteten Ergebnissen führen.

9 Inhalt der CD

Die beiliegende CD beinhaltet neben einem PDF der Diplomarbeit und dem entstandenen Programm auch eine Ansammlung von Texten, die ich im Rahmen der Diplomarbeit aus dem Internet heruntergeladen habe.

Verzeichnis:

Root:

Bilder	verwendete Bilddateien
Diplom	PDF-File der Diplomarbeit
Filme	Filme zur Veranschaulichung
HAW-Roboter	Genutzte Texte, Scripte, RCube-CD, AkSen-CD
Projekt\ballinstor	
Aksen	Programmcode für das AkSen-Board:
	-Can: Canbus -Joystick: Joysticktreiber -Mathok: Mathebibliothek für Int -Movementcontrol: Moderatorbibliothek -Omnidrivedegree: Antriebsansteuerung -Omnidrivemovementcontrol: Moderator Programm
Lart	Programmcode für das Lart-Board -div. Programmcode von Lars Brandt (Torerkennung) -app Programm mit zwei Behaviours (Ball und Tor)
WWWTexte	Genutzte Texte aus dem Netz

10 Literaturverzeichnis

10.1 Bücher, Dissertationen, Studien- und Diplomarbeiten

[Balzer02] Rainer Balzerowski: Realisierung eines webcambasierten Kamera-Systems für mobile Roboter, Diplomarbeit von 2002, http://users.informatik.haw-hamburg.de/~lego/Projekte/Balzerowski/diplomarbeit-www.pdf (Zugriffsdatum: 27.4.05).

[Brandt05] Lars Brandt: Entwicklung eines Objekttrackers für Embedded Systems zur Steuerung mobiler Roboter, Diplomarbeit von 2005, http://users.informatik.haw-hamburg.de/~kvl/brandt/diplom.pdf (Noch nicht veröffentlicht: 12.8.05).

[Breden99] Ansgar Bredenfeld, Thomas Christaller, Wolf Göhring, Horst Günther, Herbert Jaeger, Hans-Ulrich Kobialka, Paul-Gerhard Plöger, Peter Schöll, Andrea Siegberg, Arend Streit, Christian Verbeek, Jörg Wilberg: Behavior engineering with "dual dynamics" models and design tools, IJCAI-99 RoboCup Workshop, Linköping Electronic Articles in Computer and Information Science, vol. 4, no. 006/22, Linköping, Sweden, 1999.

[Bruske97] Jörg Bruske, Erika Abraham-Mumm, Gerald Sommer: Visuomotorische Koordination eines Roboterarmes mit Kohonen-Karten, neuronalem Gas und dynamischen Zellstrukturen – ein

Vergleich, http://www.ks.informatik.uni-kiel.de/modules.php/name+Downloads,d_op+getit,lid+1332 (Zugriffsdatum: 2.5.05).

[Burkhard03] Hans-Dieter Burkhard und Hans-Arthur Marsiske: Endspiel 2050 - Wie Roboter Fußball spielen lernen, Hannover, Heise Zeitschriften Verlag, 2003, ISBN:3-936931-02-X.

[CorLem03] Dietmar Cordes, Gunther Lemm: Konzeption von I/O-Erweiterungen für Lego Mindstorm, Studienarbeit von 2003, http://users.informatik.haw-hamburg.de/~kvl/lepomux/lepomux.pdf (Zugriffsdatum: 28.4.05).

[CorLem04] Dietmar Cordes, Gunther Lemm: Entwicklung einer mikroprozessorbasierten Sensor- / Aktorerweiterung für das LEGO-Mindstorm System, Diplomarbeit von 2004, http://users.informatik.haw-hamburg.de/~kvl/cordes/diplom.pdf (Zugriffsdatum: 28.4.05).

[Dieckm03] Arne Dieckmann: Verbesserung visueller Objekterkennung für mobile Roboter, Diplomarbeit von 2003, http://users.informatik.haw-hamburg.de/~kvl/diekmann/diplom.pdf (Zugriffsdatum: 8.6.05).

[Düffer04] Uwe Düffert: Vierbeiniges Laufen - Modellierung und Optimierung von Roboterbewegungen, Diplomarbeit von 2004 an der Humboldt-Universität zu Berlin, http://www.uwe-dueffert.de/publication/dueffert04_diplom.pdf (Zugriffsdatum: 23.7.05).

[Egorov04] Anna Egorova: MAAT - Multi Agent Authoring Tool for Programming Autonomous Mobile Robots, Diplomarbeit, Institut für Informatik, Freie Universität Berlin, 2004, http://robocup.mi.fu-berlin.de/docs/anna_diplom_arbeit.pdf (Zugriffsdatum: 2.5.05).

[Gerling03] Mirco Gerling: PDA-gestützte Robotersteuerung mit funkbasierter Serveranbindung, Diplomarbeit von 2003, http://users.informatik.haw-hamburg.de/~kvl/gerling/diplom.pdf (Zugriffsdatum: 20.7.05).

[Gloye05] Alexander Gloye: Lernmethoden für autonome mobile Roboter, Fachbereich Mathematik u. Informatik, Freie Universität Berlin, Dissertation von 2005, http://www.diss.fu-berlin.de/2005/36/ (Zugriffsdatum: 26.6.05).

[Görz03] G. Görz, C.-R. Rollinger, J. Schneeberger: Handbuch der Künstlichen Intelligenz, 4., korrigierte Auflage, München, Oldenbourg Verlag, 2003, ISBN: 3-486-27212-8.

[Hauck99] A. Hauck, M. Sorg, T. Schenk, and G. Färber: What can be Learned from Human Reach-To-Grasp Movements for the Design of Robotic Hand-Eye Systems?, ftp://ftp.lpr.e-technik.tu-muenchen.de/pub/papers/rovi/icra99-ah+ms+ts.ps.gz (Zugriffsdatum: 2.5.05).

[Heinol01] Horst G. Heinol: Untersuchung und Entwicklung von modulationslaufzeitbasierten 3D-Sichtsystemen, Fachbereich Elektrotechnik und Informatik, Universität Siegen, Dissertation von 2001, http://www.zess.uni-siegen.de/cms/diss/2001/heinol/heinol.pdf (Zugriffsdatum: 15.7.05).

[Hundel04] Felix von Hundelshausen: Computersehen für autonome mobile Roboter, Fachbereich Mathematik u. Informatik, Freie Universität Berlin, Dissertation von 2004, http://www.diss.fu-berlin.de/2004/243/ (Zugriffsdatum: 26.6.05).

[Ichbiah05] Daniel Ichbiah: Roboter – Geschichte_Technik_Entwicklung, München, Knesebeck, 2005.

[Jaeger96] Herbert Jaeger: The Dual Dynamics Design Scheme for Behavior-Based Robots-A Tutorial, Arbeitspapiere der GMD 966, GMD-Forschungszentrum Informationstechnik GmbH, 1996.

[Jensen04] Björn Jensen: Konzeptionierung eines Labors für Künstliche Intelligenz und Robotik, Studienarbeit von 2004, http://users.informatik.haw-hamburg.de/~kvl/jensen/studien.pdf (Zugriffsdatum: 27.4.05).

[Jones96] Joseph L. Jones und Anita M. Flynn: Mobile Roboter - Von der Idee zur Implementierung, Bonn, Addison-Wesley, 1996, ISBN: 3-89319-855-5.

[JongHwan04] Jong-Hwan Kim, Dong-Han Kim, Yong-Jae Kim und Kiam-Tian Seow: Soccer Robotics, Heidelberg, Springer, 2004, ISBN: 3-540-21859-9.

[Klette96] Reinhard Klette, Andreas Koschan und Karsten Schlüns: Computer Vision – Räumliche Information aus digitalen Bildern, Braunschweig/Wiesbaden, Vieweg Technik, 1996, ISBN: 3-528-06625-3.

[Koch03] Birgit Koch: Einsatz von Robotikbaukästen in der universitären Informatikausbildung am Fallbeispiel "Hamburger RoboCup: Mobile autonome Roboter spielen Fussball", Diplomarbeit, Fachbereich Informatik, Universität Hamburg, Februar 2003, http://www.kochcompany.de/kc/ftp/diplomarbeit_koch.pdf (Zugriffsdatum: 27.4.05).

[Köckri05] Oliver Köckritz: Roboter Fußball im Robot-Lab der HAW Hamburg, Studienarbeit von 2005, http://users.informatik.haw-hamburg.de/~kvl/koeckritz/studien.pdf (Zugriffsdatum: 23.7.05).

[Kraetz94] Gerhard K. Kraetzschmar mit Beträgen von Henry Hexmoor, Jeffrey Graham and Willie Lim: Robot Building Laboratory - Tutorial and Jump Start Session, Tutorial Notes for AAAI-94 Robot Building Lab (RBL-94), AAAI, Seattle, WA, USA, August 1994.

[Manger03] Michael Manger: Design und Realisierung von „kostengünstigen“ fußballspielenden Robotern, Studienarbeit von 2003, http://users.informatik.haw-hamburg.de/~kvl/manger/studienarbeit.pdf (Zugriffsdatum: 27.4.05).

[Manger04] Michael Manger: Design und Realisierung einer experimentellen Plattform für Roboterfußball, Diplomarbeit von 2004, http://users.informatik.haw-hamburg.de/~kvl/manger/diplom.pdf (Zugriffsdatum: 27.4.05).

[Mann00] Heinz Mann, Horst Schiffelgen, Rainer Froriep: Einführung in die Regelungstechnik – Analoge und digitale Regelung, Fuzzy-Regler,

Regler-Realisierung, Software, 8. Auflage, Hanser Lehrbuch, 2000, ISBN: 3-446-21516-6.

[Petrov04] Slav Petrov: Computer Vision, Sensorfusion und Verhaltenssteuerung für Fußball-Roboter, Diplomarbeit, Institut für Informatik, Freie Universität Berlin, 2004, http://robocup.mi.fu-berlin.de/docs/slav_diplom_arbeit.pdf.

[Schmidt00] Ulrich Schmidt: Professionelle Videotechnik, Heidelberg, Springer-Verlag, 2000, ISBN:3-540-66854-3.

[Siegert96] Hans-Jürgen Siegert und Siegfried Bocionek: Robotik: Programmierung intelligenter Roboter, Heidelberg, Springer-Verlag, 1996, ISBN:3-540-60665-3.

[Sorg03] Michael Sorg: Visual Tracking and Grasping of a Dynamic Object: From the Human Example to an Autonomous Robotic System, Dissertation, Technische Universität München, 2003, http://tumb1.biblio.tu-muenchen.de/publ/diss/ei/2003/sorg.html (Zugriffsdatum: 26.6.05).

[Zell94] Andreas Zell: Simulation neuronaler Netze, Bonn, Addison-Wesley, 1994.

[Ziener05] Michael Ziener: Konstruktion einer programmierbaren, omnidirektionalen Roboterplattform, Diplomarbeit von 2005, http://users.informatik.haw-hamburg.de/~kvl/ziener/diplom.pdf (Noch nicht veröffentlicht: 12.8.05).

10.2 Internetseiten

[3DKamera] Laserbasierte parallele 3D-Formerfassung für Robotik, http://www.ipm.fraunhofer.de/fhg/Images/paraform_d_tcm91-16310.pdf (Zugriffsdatum: 15.7.2005).

[AkSenshop05] AkSen Shop, http://www.AkSen-roboter.de/stage/index.html (Zugriffsdatum: 18.4.05).

[CIDTEC] CID-Kamera Firma, http://www.cosyco.de/produkte/kameras/zz_pdf/spezial/CID-Technologie_Uebersicht.pdf.

[CMOSEntfer] http:// www.hhi.fraunhofer.de /german/im/projects/sp/sp2/html/spims_de.html.

[FiraÖst03] Fira Weltmeisterschaftsseite aus Österreich, http://www.ihrt.tuwien.ac.at/FIRAWM03/german/default.html (Zugriffsdatum: 18.4.05).

[FuFighters] Homepage Fu-Fighters aus Berlin, http://robocup.mi.fu-berlin.de (Zugriffsdatum: 29.4.05).

[interroll] Interroll Hollding AG, http://www.interroll.de (Zugriffsdatum: 28.7.05).

[IKS96] Homepage des Projekts, „Integration Kognitiver Systeme“ an der HAW Hamburg, http://users.informatik.haw-hamburg.de/~kvl (Zugriffsdatum: 27.4.05).

[LaborKIFHB05] Labor für Künstliche Intelligenz an der Fachhochschule Brandenburg, http://ots.fh-brandenburg.de/index.php (Zugriffsdatum: 18.4.05).

[RCube05] RCube, http://ots.fh-brandenburg.de/RCube (Zugriffsdatum: 27.4.05).

[RJRules05] Regelwerk RoboCup-Junior-Soccer, http://alex.ais.fraunhofer.de/zeno/web/Regelwerk_RoboCup_Junior_Soccer_League_2005.pdf?action=openattachment&id=16654&attachmentid=5744&rootid=15465 (Zugriffsdatum: 23.7.05).

[RoboTeile] Roboterteile.de, Technik-Shop für Hobby-Robotiker http://www.roboter-teile.de/Shop/ (Zugriffsdatum: 23.7.05).

[SHARPIR12] Sharp Sensor Information zum Typ GP2D12, http://www.acroname.com/robotics/parts/R48-IR12.html (Zugriffsdatum: 27.4.05).

[Tribots05] Homepage Brainstormers Tribots aus Osnabrück, http://amy.informatik.uos.de/tribots/ (Zugriffsdatum: 29.4.05).

[Wikipedia] Anwendergesteuerte Enzyklopädie im World Wide Web, http://www.wikipedia.de.

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(4) bzw. §25(4) ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 25. August 2005

_____

Oliver Köckritz