Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

DIPLOMARBEIT

Navigieren eines Roboters mittels Kamera-Informationen

ausgeführt zum Zweck der Erlangung des akademischen Grades eines Diplom-Ingenieurs
unter der Leitung von

A.o. Univ.-Prof. Dipl.-Ing. Dr.techn. Markus Vincze
und
Dipl.-Ing. Wolfgang Ponweiser

E376
Institut für Automatisierungs- und Regelungstechnik

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik

von

Sascha Krasny
geb. am 25.07.1972
Matrikelnummer 9125985
Römerweg 1/2/8
A-7000 Eisenstadt

Wien, 4. Januar 2007

SaschaKrasny

Danksagung

Ich bedanke mich bei meinen Betreuern Ao. Univ. Prof. Dipl.-Ing. Dr. Markus Vincze und Dipl.-Ing. Wolfgang Ponweiser für die Unterstützung beim Erstellen meiner Diplomarbeit.

Ebenfalls bedanke ich mich bei Peter Einramhof und den restlichen Mitgliedern der RobSense Gruppe, die mir alle während der Diplomarbeit und meiner Zeit im ACIN mit Rat und Tat zur Seite standen. Ich wünsche euch allen viel Erfolg bei euren künftigen Herausforderungen!

Danke meine lieben Freunde und meine lieben Kollegen bei der Alcatel Transport Solution Austria GmbH. Eure ewigen, bohrenden Fragen nach dem aktuellen Status der Diplomarbeit haben mich bis zum Ende des Studiums begleitet. Und danke fürs Korrekturlesen, Priska!

Mein besonderer Dank gilt meiner Lebensgefährtin Silvia (die sich tapfer durch alle meine Texte über Robotik gequält hat), meinen Eltern und Sharleen für die langjährige Unterstützung und Aufmunterung während meines Studiums.

Kurzfassung

Die Gruppe *RobSense* des Instituts für Automatisierungs- und Regelungstechnik (ACIN) an der Technischen Universität Wien beschäftigt sich mit der Entwicklung von robusten Computer Vision Algorithmen für Roboter. Diese Diplomarbeit beschäftigt sich mit der Verwendung von Kameras und Computer Vision in der mobilen Robotik. Eine künftige Anwendung dafür sind *Serviceroboter*, die in Indoor Umgebungen Dienstleistungen erbringen sollen.

Serviceroboter müssen in des Menschen gestalteten Umgebungen agieren können. Das primäre Sinnesorgan von Menschen ist das Auge, daher sind unsere Umgebungen besonders für die visuelle Wahrnehmung strukturiert. Der Einsatz von visuellen Sensoren am Roboter liegt daher nahe. Die Erfahrungen mit der Integration einer Computer Vision in eine Robotersteuerung für mobile Roboter sind aber noch nicht ausreichend, um universell einsetzbare Lösungen anbieten zu können. Derzeit werden für derartige Steuerungen problembezogene Computer Vision Algorithmen entwickelt, die dann direkt mit der Steuerung verbunden werden. Diese Form der Integration stellt aber keine austauschbare, flexible Lösung dar.

Mobile Roboter benötigen Sensoren, um die für eine Navigation in dynamischen oder unbekannten Umgebungen notwendigen Informationen zu gewinnen. Dafür werden erfolgreich Sonarsensoren oder Laserscanner eingesetzt. Deren gewonnene Entfernungsdaten werden direkt für reaktive Steuerungen oder in Form von Umgebungskarten für planende Steuerungen verwendet. Die Integration der Sensorschnittstelle in die Steuerung ist hier schon weit fortgeschritten, die Sensoren können nahezu frei ausgetauscht werden, ohne das die Steuerung verändert werden muss.

Das Ziel dieser Diplomarbeit ist eine Analyse der Möglichkeiten, um vorhandene Computer Vision Algorithmen ähnlich erfolgreich als Sensoren in eine Robotersteuerung zu integrieren. Als Basis für die Integration dienen Behaviors, die Bausteine einer reaktiven Robotersteuerung mit denen Reiz-Reaktionsverbindung abgebildet werden. Als Reize für Behaviors können auch mehrdimensionale, symbolische oder semantische Daten dienen, die eine Bildverarbeitung aus einem Kamerabild extrahiert. Eine Lösung zur besseren Integration von vorhandenen Computer Vision Algorithmen ist die Verwendung einer weiteren Abstraktionsschicht zwischen Kamera und Behavior in Form eines *virtuellen Sensors*. Dadurch kann der vorhandene Computer Vision Algorithmus seine Charakteristik behalten, während gleichzeitig dem Behavior eine geeignetere Schnittstelle geboten werden kann.

Das Ergebnis der Arbeit ist eine Analyse der Möglichkeiten, um ein Sensorsystem aus Kameras und Computer Vision in eine Robotersteuerung zu integrieren. Die Resultate der Analyse sind durch praktische Demonstrationsbeispiele im S4R-Framework illustriert. Das S4R-Framework ist eine servicebasierte Applikationsplattform, die eine Reihe von Computer Vision Algorithmen und Dienste zur Ansteuerung eines mobilen Roboters anbietet. Ein weiteres Ergebnis der Arbeit sind eine Reihe von Verbesserungsvorschlägen, um das S4R-Framework noch effektiver für eine solche behavior-basierte Steuerung mit visuellen Sensoren verwenden zu können.

Abstract

The *RobSense* group of the Automation and Control Institute (ACIN) of the Technical University of Vienna develops robust computer vision algorithms for robots. This diploma thesis deals with the use of cameras and computer vision for mobile robotics. A future application for this are *service robots* that provide household services in indoor enviroments.

Service robots have to operate in enviroments designed for humans. The primary sense organ of humans is the eye and our enviroments are specifically structured for visual perception. Therefore the usage of visual sensors on a robot stands to reason. There is not enough experience with the integration of computer vision into a control system for mobile robots to provide generic solutions for this challenge. Up to now computer vision algorithms especially developed for the given task are used directly by the control system. This type of integration does not provide exchangeable and flexible solutions.

Mobile robots need sensors to extract the information necessary to navigate in dynamically changing or unknown enviroments. Today sonar range finders or laser scanners are used for this task. The range data provided by those sensors is directly used in reactive control systems or transformed into a map which is used for a route planer. The advanced integration of the sensor interface into the control system allows sensors to be exchanged almost freely without any adaption of the control system.

The goal of this diploma thesis is an analysis of the possibilities to use existing computer vision algorithms as sensors for a robot control system in a similar successful way. The integration is based on behaviors, the smallest part of reactive robot control systems. Behaviors represent stimulus-reaction connections. The stimulus for a behavior can be the multi-dimensional, symbolic or semantic data output of an image processing. A more advanced integration for existing computer vision algorithms can be achieved using a *virtual sensor*. It is another layer of abstraction between the camera and the behavior. This abstraction enables the existing computer vision algorithm to keep its characteristic while the behavior is provided with a better fitting interface.

The results of this thesis is an analysis of the options to inegrate a camera and computer vision sensor system with a reactive robot control system. The results of the analysis are illustrated with hands-on examples that have been implemented using the S4R framework. The S4R framework is a service-based application platform that provides ready to use computer vision algorithms as well as services to control a mobile robot. Another result of the thesis are suggestions for improvements for the S4R framework to create a framework for an even better reactive robot control system.

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Motivation . . . 1
1.2 Aufgabenstellung . . . 2
1.3 Lösungsweg . . . 2
1.4 Kapitelübersicht . . . 3

2 Mobile Roboter **4**
2.1 Definition eines Roboters . . . 4
2.2 Serviceroboter . . . 6
2.3 Bewegungssteuerung von mobilen Robotern . . . 8
2.3.1 Diskrete Bewegungssteuerung . . . 8
2.3.2 Kontinuierliche Bewegungssteuerung . . . 8
2.4 Steuerungskonzepte für mobile Roboter . . . 9
2.4.1 Das hierarchische Paradigma . . . 9
2.4.2 Das reaktive Paradigma . . . 10
2.4.3 Das hybride Paradigma . . . 10
2.5 Ausgewählte Architekturen . . . 11
2.5.1 Subsumption architecture . . . 11
2.5.2 Motor Schemas . . . 11
2.5.3 Saphira Architektur . . . 13
2.6 Zusammenfassung . . . 14

3 Die behavior-basierte Robotik **15**
3.1 Die Geschichte der behavior-basierten Robotik . . . 15
3.1.1 Der behavioristische Ansatz in der Psychologie (1900 - 1930) . . . 15
3.1.2 Bettwanzen und Schildkröten - Die ersten behavior-basierten Roboter (1930 - 1960) . . . 16
3.1.3 Gedankenexperimente und Rebellen - Die moderne behavior-basierte Robotik (1980 - 1990) . . . 18
3.1.4 Mit Behaviors ins 21. Jahrhundert (1990 - 2006) . . . 19
3.2 Definition eines Behaviors . . . 20
3.3 Die Reiz-Reaktionsabbildung . . . 20
3.3.1 Der Bereich der Reaktionen: R . . . 21
3.3.2 Der Bereich der Reize: S . . . 21
3.3.3 Die Abbildungsfunktion: β . . . 21
3.4 Formelle Darstellungsformen für Behaviors . . . 22
3.4.1 Reiz-Reaktionsdiagramm . . . 23
3.4.2 Funktionale Notation . . . 23

3.4.3 Endliches Akzeptor Diagramm . . . 24
3.5 Koordinatoren für Behaviors . . . 25
3.5.1 Kooperative Methoden . . . 25
3.5.2 Konkurrenzmethoden . . . 25
3.6 Kodierung eines Behaviors . . . 26
3.6.1 Die Methode der virtuellen Potentialfeld . . . 26
3.6.2 Behaviors mit einer Fuzzy Control . . . 29
3.7 Zusammenfassung . . . 30

4 Sensoren in der Robotik **31**
4.1 Gängige Sensoren und ihre Verwendung in der mobilen Robotik . . . 31
4.1.1 Taktile Sensoren . . . 32
4.1.2 Odometrie . . . 32
4.1.3 Koppelnavigation . . . 33
4.1.4 GPS und andere Systeme zur Positionsbestimmung . . . 33
4.1.5 Ultraschallsensoren . . . 34
4.1.6 Infrarotsensoren . . . 35
4.1.7 Laserscanner . . . 36
4.2 Computer Vision als Sensor für die Robotik . . . 37
4.2.1 Das digitale Bild . . . 37
4.2.2 Die Abbildung . . . 37
4.2.3 Die digitale Kamera . . . 38
4.2.4 Bildverarbeitung in Software oder Hardware . . . 39
4.2.5 Die Bildverarbeitung . . . 39
4.2.6 Machine Vision in der Robotik . . . 40
4.2.7 Computer Vision für mobile Roboter . . . 40
4.2.8 Computer Vision in der behavior-basierten Robotik . . . 41
4.2.9 Ausgewählte Beispiele . . . 42
4.3 Zusammenfassung . . . 44

5 Demonstrationsbeispiele **45**
5.1 Das Setup für die Beispiele . . . 45
5.1.1 Die verwendete Hardware . . . 45
5.1.2 Die verwendete Software . . . 46
5.2 Follow Ball Behavior . . . 48
5.2.1 Die Aufgabe . . . 48
5.2.2 Das Behavior . . . 48
5.2.3 Die Komponenten . . . 49
5.2.4 Die Analyse . . . 49
5.3 Dual Color Behavior . . . 52
5.3.1 Die Aufgabe . . . 52
5.3.2 Das Behavior . . . 52
5.3.3 Die Komponenten . . . 52
5.3.4 Die Analyse . . . 53
5.4 Triple Color Behavior . . . 55
5.4.1 Die Aufgabe . . . 55
5.4.2 Die Komponenten . . . 55
5.4.3 Die Analyse . . . 55

5.5 Multi Color Behavior . . . 57
5.5.1 Die Aufgabe . . . 57
5.5.2 Das Behavior . . . 57
5.5.3 Die Komponenten . . . 58
5.5.4 Die Analyse . . . 59
5.5.5 Der praktische Einsatz . . . 62
5.6 Zusammenfassung . . . 62

6 Bewertung **64**
6.1 Computer Vision für die mobile Robotik . . . 64
6.1.1 Kamera und Objektiv . . . 64
6.1.2 Geschwindigkeit . . . 65
6.1.3 Datenaufbereitung . . . 66
6.1.4 Erweitertes Reiz-Reaktionsdiagramm . . . 67
6.1.5 Beispiel Farb-Recogniser mit Behavior . . . 68
6.2 Das S4R-Framework . . . 70
6.2.1 Framework Struktur . . . 70
6.2.2 Framework Services . . . 71
6.2.3 Mobile Robot Services . . . 71
6.2.4 S4R Benutzeroberfläche . . . 72
6.3 Zusammenfassung . . . 74

7 Schlussbemerkung **75**

A Technische Daten Laptop „Apollo“ **77**

B Seriell/USB-Adapter Unterstützung für die Mobile Robot Services **79**

C Vorlage eines Behaviors: Follow Ball **80**

D Index **86**

Literaturverzeichnis **88**

Abbildungsverzeichnis

2.1 Funktionelle Unterteilung von Robotern. 5
2.2 Serviceroboter beim International Contest for Cleaning Robots. 7
2.3 Schema des hierarchischen Paradigmas. 9
2.4 Schema des reaktiven Paradigmas. 10
2.5 Schema des hybriden Paradigmas. 10
2.6 Die *Subsumption architecture*. 12
2.7 Die *Motor Schemas* Architektur. 12
2.8 Die *Saphira* Systemarchitektur. 13

3.1 Das Modell der Reiz-Reaktionsverbindung mit Black Box. 16
3.2 Tolmans Theorie des zielgerichteten Behaviorismus. 17
3.3 Die Schildkröten von Walter. 18
3.4 Die Braitenberg Vehicles 2a, 2b und 3c. 19
3.5 Ein einfaches Reiz-Reaktionsdiagramm. 20
3.6 Beispiele für eine diskrete und kontinuierliche Abbildungsfunktion. 22
3.7 Reiz-Reaktionsdiagramm mit drei Behaviors. 23
3.8 FSA Diagramm mit vier Zuständen. 24
3.9 Beispiele für die Koordination von Behaviors. 26
3.10 Darstellungen von virtuellen Potentialfeldern. 28
3.11 Das Schema einer Fuzzy Control. 29
3.12 Beispiel für eine Fuzzy Control 30

4.1 Beispiel für einen Winkelenkoder und Kodierscheiben. 33
4.2 Beispiel für einen Sonarsensor. 34
4.3 Beispiele für Infrarotsensoren. 35
4.4 Beispiele für Laser Entfernungsmesser. 36
4.5 Schematische Abbildung einer Camera Obscure. 38
4.6 Reiz-Reaktionsdiagramm eines Behaviors mit Computer Vision. 42
4.7 Beispiel eines mit der ROCI Plattform implementierten Behaviors. 43

5.1 Der ActivMedia Robotics Pioneer 2-DX Roboter James. 46
5.2 Das Follow Ball Behavior als Reiz-Reaktionsdiagramm. 49
5.3 Der Pioneer 2-DX Roboter mit Aufbau. 50
5.4 Momentaufnahme der S4R-Framework Benutzeroberfläche: Follow Ball. 51
5.5 FSA Diagramm zum Dual Color Behaviors. 53
5.6 Die bei den Beispielen verwendeten Markierungen und Objekte. 53
5.7 Momentaufnahme der S4R-Framework Benutzeroberfläche: Dual Color Behavior. 54
5.8 Momentaufnahme der S4R-Framework Benutzeroberfläche: Triple Color Behavior. 56
5.9 FSA Diagramm des Multi Color Behaviors. 59

5.10 Der Pioneer 2-DX Roboter mit Farbkamera und 4 mm Objektiv. 60
5.11 Momentaufnahme der S4R-Framework Benutzeroberfläche: Multi Color Behavior. 61
5.12 Aufnahmen einer typischen Testfahrt durch das ACIN RobSense Labor. 62
5.13 YO!Einstein Tag im Juni 2005. 63

6.1 Das erweiterte Reiz-Reaktionsdiagramm. 67
6.2 Eine Beispielanwendung für das erweiterte Reiz-Reaktionsdiagramm. 67
6.3 Drei Marker und 15 farbige Bälle in einem Test des Multi Color Behavior. 68
6.4 Das erweiterte Reiz-Reaktionsdiagramm zum Multi Color Behavior. 69
6.5 Die aktuelle S4R Benutzeroberfläche. 73
6.6 Vorschläge für künftige Erweiterungen der S4R Benutzeroberfläche. 74

Tabellenverzeichnis

3.1 Die Liste der Übergangsaktionen δ eines FSA Diagramms. 25

5.1 Liste der Übergangsaktionen δ zum Dual Color Behavior. 52
5.2 Liste der Übergangsaktionen δ zum Multi Color Behavior. 58

6.1 Die Ergebnisse von 13 Testfahrten mit dem Multi Color Behavior. 69

Kapitel 1

Einleitung

1.1 Motivation

Ende 1999 waren etwa 6000 mobile Serviceroboter in der Forschung, Industrie und in privaten Haushalten im Einsatz. Die Anwendungsgebiete dieser Roboter sind der medizinische Einsatz, als Unterwasser- und Weltraumroboter, als Reinigungsroboter und zum kleinen Teil auch als Serviceroboter im privaten Haushalt. Ende 2002 waren bereits 19.000 Serviceroboter installiert. Einer Studie der *United Nations Economic Commission for Europe* [UNECE03] zufolge wird die Zahl bis Ende 2006 auf 50.000 Serviceroboter steigen. Die Zahlen erinnern an den Einzug der Industrieroboter in die industrielle Fertigung in den 70er Jahren. 30 Jahre später wiederholt sich die Entwicklung und wir stehen an der Schwelle des großflächigen, kommerziellen Einsatzes von Servicerobotern.

Während mit den Industrierobotern die industriellen Prozesse erneuert und verändert wurden, werden Serviceroboter in unser Leben eingreifen und unseren Alltag verändern. Während die Gesellschaft noch dabei ist, die Schlüsseltechnologien der letzten Jahre - PC, Internet und Mobiltelefonie - in das soziale Umfeld zu integrieren, steht die nächste technologische Revolution schon vor der Haustür. Serviceroboter werden Aufgaben durchführen, die bisher Menschen vorbehalten waren. In Form von intelligenten Haushaltsgeräten wie Roboter-Staubsaugern oder Roboter-Rasenmähern werden Roboter die privaten Haushalte erobern. Medizinische Serviceroboter stellen eine Lösung für künftige medizinische und soziale Betreuungsengpässe dar. Wir werden mit den intelligenten Maschinen leben.

Einen Blick in die Zukunft kann man in Südkorea machen, dem Land mit der momentan weltweit dichtesten Integration von neuen Technologien in der Gesellschaft. Während die im Bereich der Robotik führenden Nationen Japan, USA und Europa ihren Fokus auf militärische, industrielle und humanoide Roboter legen, hat sich Südkorea die Entwicklung von Servicerobotern als nationales Ziel gesetzt. Das zuständige Ministerium für Information und Kommunikation plant den Einsatz der neuen Technologie ab 2007. Zwischen 2015 und 2020 will man einen Serviceroboter in jedem Südkoreanischen Haushalt stehen haben. Wie vor 20 Jahren der PC, soll nun der Roboter als neues Familienmitglied ein wirtschaftlicher Erfolg werden. „Robots could change how we live in a way we cant predict right now. Its like the PC. No one ever thought the PC and the Internet would transform our society the way they have.“ fasst der Direktor des südkoreanischen Zentrums für intelligente Robotik Kim Mung Sag [Oni06] die technologische Umbruchstimmung zusammen.

Eine wichtige Technologie, die Roboter für ihren Schritt in die Haushalte benötigen, ist die Fähigkeit zu sehen. Menschen sind visuell orientierte Wesen und dementsprechend ist unser privater und öffentlicher Lebensraum organisiert. Für eine erfolgreiche Integration in diesen Lebensraum muss Robotern eine robuste Form von Computer Vision zur Verfügung stehen. Das ist zweifellos eine große Herausforderung für die Forschung im Bereich der Computer Vision.

1.2 Aufgabenstellung

Die Aufgabenstellung lautet einen mobilen Roboter mit behavior-basierten Methoden und Bildinformation navigieren zu können. Die Steuerung soll für Serviceroboter geeignet sein, die in Büros oder Haushalten agieren. Das vorhandene S4R-Framework soll dabei als Softwareumgebung für eine behavior-basierte Steuerung für die im Framework eingebundenen Roboter, z.Z. ein Pioneer 2-DX der Firma ActivMedia Robotics, dienen. Das Ziel der Arbeit ist eine Analyse einer solchen Steuerung, nicht die Funktion einer konkreten Applikation.

Damit soll eine Reihe von Test- und Demonstartionsbeispielen für das S4R-Framework entwickelt werden. Anhand der Beispiele kann dann die praktische Integration von Computer Vision Modulen in eine Robotersteuerung demonstriert werden. Die Beispiele sollen die Möglichkeiten einer behavior-basierten Steuerung testen und die Erkenntnisse der Analyse verifizieren. Die Beispiele können außerdem als Vorlagen für weitere Entwicklungen dienen.

1.3 Lösungsweg

Die Einarbeitung in die Thematik begann mit einem Studium des Grundwissens der mobilen Robotik. Die Sensorik für Roboter und die verschiedenen Steuerungskonzepte wurden in Theorie und Praxis erlernt. Als Übungsobjekt diente dazu ein Pioneer 2-DX Roboter mit der originalen Saphira Steuerungssoftware.

Der zweiter Schritt widmete sich dem S4R-Framework und den darin enthaltenen Services aus den Bereichen Computer Vision und Mobile Robot. Durch praktisches Experimentieren und eigene Testbeispiele wurden sowohl die Anwenderseite als auch die Programmiererseite des Frameworks kennengelernt.

Der dritte Schritt widmete sich der behavior-basierten Robotik. Als Einarbeitung in die Thematik diente das Studium der wichtigsten Arbeiten der Pioniere in diesem speziellen Bereich. Eine Recherche im Internet und die Lektüre von aktuellen Fachzeitschriften lieferte Informationen über aktuelle Anwendungen und den Stand der Technik der behavior-basierten Robotik.

Parallel zur Analyse der Integrationsmöglichkeiten einer Computer Vision mit einer Robotersteuerung wurde mit der Entwicklung von Test- und Demonstrationsbeispielen begonnen. Die Struktur der ersten Beispiele ist einfach. Später wurde die Komplexität der Beispiele schrittweise gesteigert. Ausgewählte Beispiele wurden außerdem so geschrieben, dass sie bei Laborübungen und Roboterdemonstrationen einfach eingesetzt werden können.

1.4 Kapitelübersicht

Die Kapitel 2 - 4 widmen sich der Analyse und Beschreibung der notwendigen Grundlagen zum Lösen der Aufgabenstellung. In Kapitel 2 werden die verschiedenen Aspekte von mobilen Robotern näher beleuchtet. Im besonderen werden die verschiedenen Steuerungskonzepte für mobile Roboter vorgestellt und anhand von bekannten Steuerungsarchitekturen illustriert. Kapitel 3 konzentriert sich auf die in reaktiven Steuerungssystemen benötigten Behaviors. Ausgehend von dieser kleinsten Systemeinheit wird das Konzept einer reaktiven Steuerung bottom-up analysiert. Kapitel 4 widmet sich dem Einsatz von Sensoren in der mobilen Robotik. Mobile Roboter benötigen Sensoren, um die Umwelt wahrnehmen zu können und mit ihr interagieren zu können. Es werden kurz alle gängigen Sensoren für den mobilen Einsatz vorgestellt. Der Schwerpunkt des Kapitels widmet sich Kamera und Bildverarbeitung als Sensor für die mobile Robotik.

In Kapitel 5 werden einige der im Rahmen der Diplomarbeit ausgearbeiteten Demonstrationsbeispiele vorgestellt. Dazu wurden vorhandene Computer Vision Module mit einer Robotersteuerung verbunden. Diese Beispiele dienen zur Überprüfung der Annahmen und Testen der Möglichkeiten. Die Ergebnisse wurden direkt in den Entwicklungsprozess der nächsten Beispiele einbezogen, so dass eine tiefergehende Analyse der Problematik möglich war.

Kapitel 6 enthält eine Auswertung aller Ergebnisse. Dazu werden die verschiedenen Aspekte der Verwendung von Computer Vision in der mobilen Robotik bewertet. In einem zweiten Teil werden die Erfahrungen aus der Arbeit anhand eines praktischen Beispiels illustriert. Der dritte Teil des Kapitels widmet sich eingehend dem S4R Framework. Die Möglichkeiten das erworbene Wissen im Framework zu integrieren werden mit Vorschlägen und konkreten Beispielen präsentiert.

In der Schlussbemerkung im Kapitel 7 finden sich Vorschläge für weiterführe Experimente.

Kapitel 2

Mobile Roboter

In diesem Kapitel wird die Funktion eines Roboter im allgemeinen und von Servicerobotern im Speziellen betrachtet. Serviceroboter sind eine Untergruppe der mobilen Roboter, die sich durch die Eigenschaften Mobilität und Autonomität auszeichnen. Sie erfüllen Aufgaben aus dem Bereich der Dienstleistungen in menschlicher Umgebung oder sogar im Kontakt mit Menschen. Entsprechend sind auch die Anforderungen an Form, Benutzerschnittstelle und Sicherheit. Im weiteren werden die Steuerungsmöglichkeiten von mobilen Roboter betrachtet, wobei der Schwerpunkt auf reaktiven Systemen liegt. Reaktive Systeme erlauben es Robotern in unbekannten oder dynamischen Umgebungen effizient zu arbeiten. Reaktive Systeme basieren auf den Theorien der behavior-basierten Robotik, die im Kapitel 3 behandelt wird.

2.1 Definition eines Roboters

Die Bedeutung und damit auch die Definition eines Roboters hat sich im Laufe der Jahre stark verändert [Ark98]. Vor einem halben Jahrhundert wurden Roboter im Brockhaus noch als „künstliche Menschen, die Bewegungen nachvollziehen können“ beschrieben. 1985 definierte die *Robotic Industry Association* einen Roboter als „einen programmierbaren, multifunktionalen Manipulator, der entworfen wurde, um Material, Teile, Werkzeuge oder spezielle Geräte durch variabel programmierbare Bewegungen zu bewegen, um eine Menge von Aufgaben erledigen zu können.“ Heute sind die Aufgabengebiete und Eigenschaften von Robotern vielfältiger und die Definition ist entsprechend differenzierter. Laut *Robots and Automation Society* [RAS06] ist ein Roboter „... eine intelligente Maschine, die Arbeit erledigen kann, die für Menschen zu schmutzig, zu gefährlich, zu präzise oder zu mühsam ist.“

In einer weiteren Verfeinerung der Definition wäre ein Serviceroboter dann eine Maschine, die selbständig Dienstleistungen für Menschen erbringt. Serviceroboter sind eine Untergruppe der mobilen Roboter, die wiederum eine Untergruppe von Robotern allgemein sind. Die funktionelle Gruppierung verschiedener Robotertypen wie Industrieroboter, Serviceroboter, Forschungsroboter oder Humanoidem Roboter ist in Abbildung 2.1 dargestellt. Serviceroboter kann man weiter in ihre konkreten Aufgabengebiete unterteilen. Beispiele dafür wären Serviceroboter die Staubsaugen, Rasenmähen, Servieren oder Müll einsammeln.

Alle Roboter der Gruppe der mobilen Roboter besitzt zwei besondere Eigenschaften: Mobilität und Autonomität. Sind beide Eigenschaften vollständig erfüllt, so hätte man das Ziel der Forschung - einen vollständig mobilen, autonomen Roboter - erreicht. Die einzelnen Begriffe

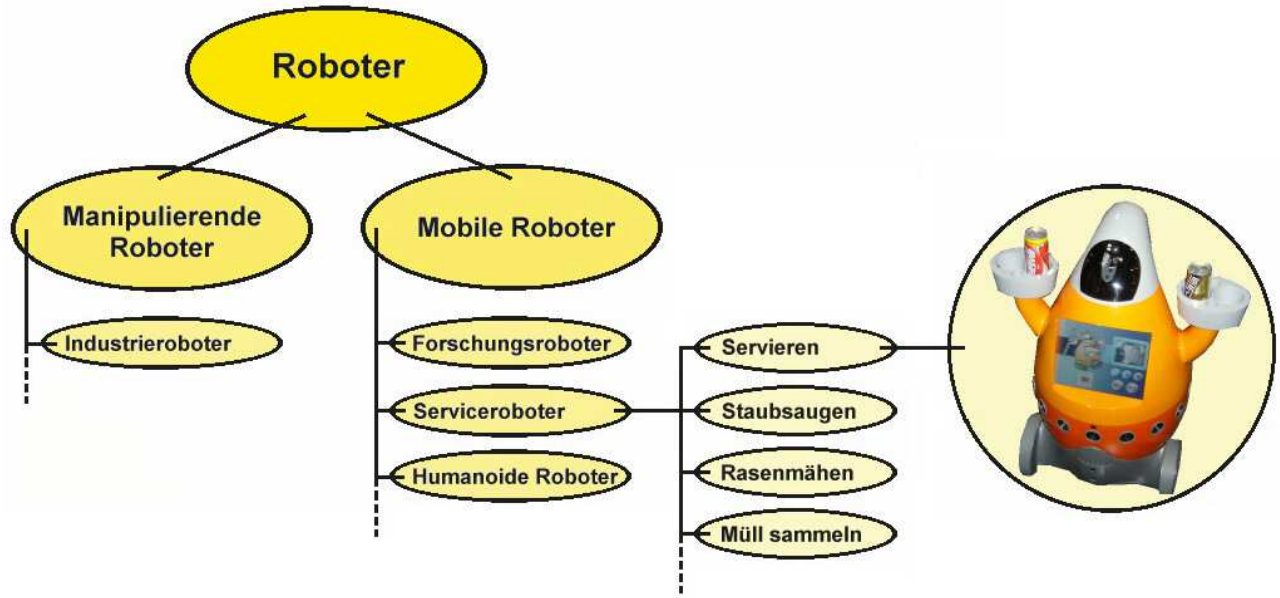

Abbildung 2.1: Funktionelle Unterteilung von Robotern. Als Beispiel für einen Serviceroboter ein Servierroboter der koreanischen Firma *Robodigm*.

werden heute von [AuRo], [Ark98] und [OlGo03] wie folgt beschrieben.

Mobiler Roboter Ein Roboter, der sich selbständig in seiner Umwelt fortbewegen kann. Dazu benötigt er einen Antrieb der den Umweltbedingungen entspricht, also Räder oder Beine für ebenes Terrain, Propeller oder Turbinen für ein flugtaugliches Modell, eine Schiffsschraube für ein schwimmendes Modell, usw. Ein mobiler Roboter besitzt außerdem ein Steuerungssystem, um den Antrieb zu kontrollieren.

Autonomer Roboter Ein Roboter, der seine Aufgaben intelligent in einer unstrukturierten Umgebung ohne menschliche Hilfestellung erledigen kann. Alle nicht dauernd ferngesteuerten Maschinen sind zu einem bestimmten Grad selbstständig, um in ihrem Arbeitsbereich agieren zu können. Der Begriff Autonomer Roboter wird aber nur für Roboter mit einem besonders hohen Grad von Autonomität verwendet. Ein besonders stabiles selbstständiges Verhalten benötigen Systeme, bei denen eine Kommunikation zwischen Roboter und Überwacher nicht immer gewährleistet ist z.B. Forschungsroboter für die Tiefsee oder Weltraumsonden.

Die beiden Kriterien lassen sich auf drei Arten kombinieren:

- Ein mobiler, aber nicht autonomer Roboter wäre beispielsweise ein unbemanntes Fahrzeug, das als ferngesteuerte Drohne in für Menschen gefährlichen Gebieten operieren kann.
- Ein nicht mobiler, aber zumindest teilweise autonomer Roboter wäre ein moderner Industrieroboter, der im Rahmen seines Arbeitsbereiches bekannte Objekte frei erkennen und manipulieren kann und der zudem in der Lage ist, auf unbekannte Fehlerzustände selbstständig zu reagieren.

- Mobile, autonome Roboter stellen ein Ziel der aktuellen Forschung dar: Maschinen, die ihre Aufgaben selbstständig in einer unstrukturierten, dynamischen Umgebung erledigen können. Ein Beispiel wäre ein Serviceroboter, der im Haushalt Reinigungsaufgaben erledigen kann.

Vollständig mobile, autonome Roboter haben die folgenden Fähigkeiten:

- Sie können sich entsprechend ihres Antriebs selbständig von einem definierten Ort zu einem anderen definierten Ort bewegen.
- Sie arbeiten über lange Zeiträume ohne menschliches Eingreifen.
- Sie sind in einem bestimmten Maß in der Lage die eigene Funktionsfähigkeit zu diagnostizieren und sich selbst zu reparieren.
- Sie gefährden oder beschädigen weder Menschen, ihre Umgebung noch sich selbst.

Ein Beispiel eines praktischen Tests für mobile, autonome Roboter ist die vom US-amerikanischen Verteidigungsministerium veranstaltete *DARPA Grand Challenge* [DARPA]. Die Aufgabenstellung des Wettbewerbs lautet eine über 200 km lange, unbekannte Strecke in weniger als zehn Stunden mit einem mobilen, autonomen Roboter zurückzulegen. Um die notwendige Mindestgeschwindigkeit für die Aufgabe zu erreichen, werden kommerzielle Fahrzeuge (PKW und Motorräder) als Basis für unbemannte Roboterfahrzeuge verwendet. Die gestellte Aufgabe konnte bei der zweiten derartigen Veranstaltung im Oktober 2005 erstmals erfolgreich gelöst werden.

2.2 Serviceroboter

Serviceroboter sind mobile Roboter mit einem hohen Maß an Autonomität, um eine spezifische Aufgabe erledigen zu können. Die Aufgaben von Servicerobotern sind Dienstleistungen, die in der Umgebung von Menschen oder sogar im direkten Kontakt mit Menschen erledigt werden müssen. Typische Arbeitsplätze sind Heim, Büro oder Gewerbe. Typische Aufgaben aus heutiger Sicht sind reinigen, rasenmähen, Akten sortieren oder servieren. Es handelt sich dabei also keineswegs um schwere oder gesundheitsschädliche Arbeiten, wie sie im industriellen Bereich vor dem Einsatz von Industrierobotern zu finden waren. Vielmehr sollen Serviceroboter unangenehme, monotone, aber notwendige Dienstleistungen erbringen. Einen Serviceroboter kann man sich als intelligente mechanische Haushalts- oder Bürohilfe vorstellen, die bestimmte Aufgaben erledigen kann.

Erste Eindrücke von den zukünftigen Haushaltshilfen lassen sich bei Roboter Konferenzen wie der International Conference on Intelligent Robots and Systems (IROS) gewinnen. So wie die DARPA Grand Challenge oder die Roboter-Fußball Weltmeisterschaft gibt es auch für Serviceroboter sportliche Wettbewerbe, um ihre Leistungen vergleichen zu können. Der *First International Contest for Cleaning Robots* fand im Rahmen der IROS 2002 statt [PHS06]. Bei dem Wettbewerb wurden Haushaltsroboter beim Reinigen eines Wohnzimmers miteinander verglichen. Die maximal 30 kg schweren Roboter mussten in 10 Minuten ein 5 x 5 m großes Zimmer (Abbildung 2.2) reinigen. Die Verschmutzung wurde mit weißem Zucker simuliert.

Serviceroboter sind keineswegs an eine bestimmte Form gebunden. Es hätte wenig Sinn, wenn man nur humanoide Serviceroboter entwickeln würde. Ein zweibeiniger instabiler Antrieb mit

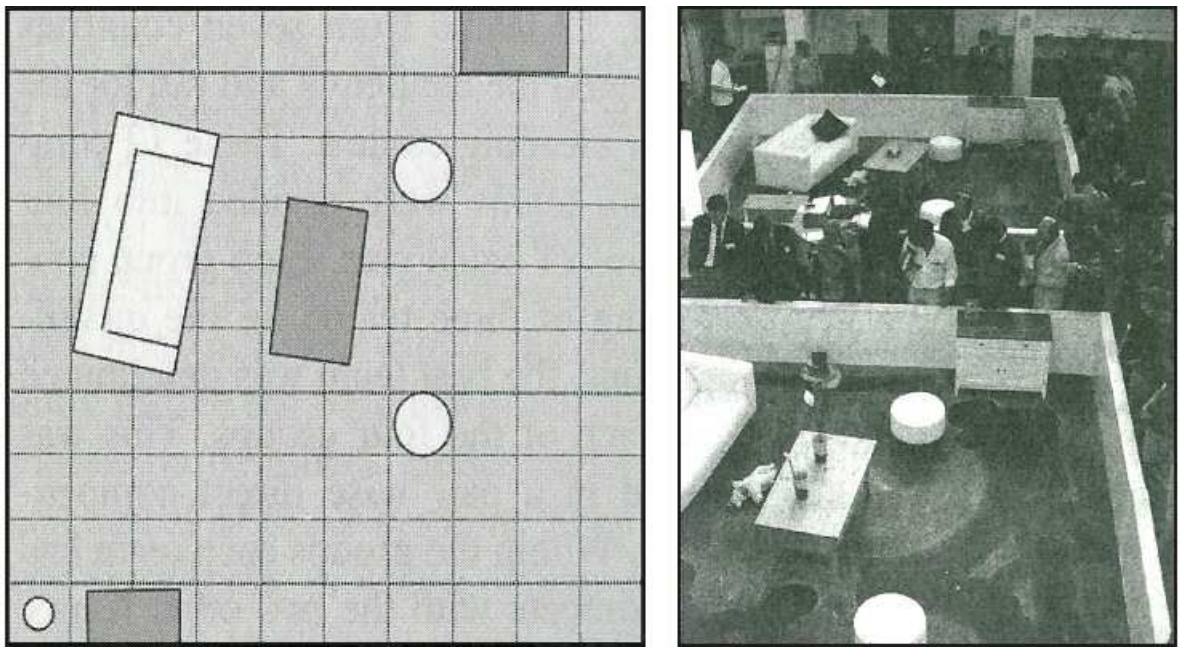

Abbildung 2.2: Serviceroboter beim International Contest for Cleaning Robots. Links eine Skizze des zu reinigenden Wohnzimmers mit Couch, Tisch, Regal, Hockern, Anrichte und Lampe. Rechts ein halbgereinigtes Zimmer mit noch teilweise weiß verschmutztem Boden. Die Abbildung wurde [PHS06] entnommen.

hohem Schwerpunkt ist beim Erledigen der meisten Serviceleistungen wenig hilfreich. Vielmehr wird die Form von Servicerobotern ihren Aufgaben angepasst sein. Ein Roboter der nur Staubsaugen kann, wird eher die Form einer flachen Box haben. Ein Serviceroboter der Aufgaben eines Zimmermädchens erledigen kann, wird lange Teleskoparme haben, um einfach ein Bett beziehen zu können. Das die Form der Aufgabe folgt, sieht man auch an dem bereits realisierten Servier-Roboter in Abbildung 2.1: Er hat einen niedrigen Schwerpunkt, um stabil fahren zu können. Die beiden Arme samt Tabletts mit Halterungen für die Getränke sind fix montiert. Die Ultraschallsensoren zur Hindernisserkennung sitzen tief unten, die Kamera zum Kontakt mit den Benutzer sitzt hoch oben im „Kopf“.

Serviceroboter benötigen eine oder mehrere Schnittstellen zur Kommunikation mit Menschen, beispielsweise Tastatur und Bildschirm oder Sprach-Ein-/Ausgabe. Der Servier-Roboter in Abbildung 2.1 besitzt dazu einen Touchscreen. Aber auch Form und Farbe des Gehäuses sind entsprechend gewählt, um eine emotionale Reaktion beim Benutzer hervorzurufen („Oh, ein herziger Roboter. Wir sollten etwas bestellen.“) und die Interaktion zu fördern. Serviceroboter werden für die Kommunikation mit Menschen eine soziale Intelligenz benötigen [Bre02], die Einfluss auf das (Navigations-)Verhalten des Roboters hat.

Die Kehrseite der Interaktion zwischen Mensch und Roboter stellt die Frage der Sicherheit dar. Unfälle können bei keinem technischen Gegenstand ausgeschlossen werden. Derzeit werden autonome Systeme nur langsam eingesetzt (z.B. automatische Bremssysteme in der Automobilbranche), da eine extrem sichere und robuste Implementierung notwendig ist und zudem eine große Rechtsunsicherheit im Fehlerfall herrscht. Mit ähnlichen Bedenken werden auch Serviceroboter konfrontiert werden.

Eine besondere Bedeutung im Design von Servicerobotern werden schließlich auch noch die Ko-

sten haben. Während Serviceroboter für Büros oder Gewerbe noch etwas teurer sein können, werden Serviceroboter für Haushalte viel härteren Preiskriterien unterliegen. Es ist davon auszugehen, dass für solche Serviceroboter eine besonders günstige Hard- und Software nötig sein wird. Schließlich müssen die Endkunden bereit sein eine völlig neue Art von Gerät zu kaufen. Wenn man bedenkt, dass heute alleine die Hardware eines entsprechenden Forschungsroboters samt rudimentärer Betriebssystem-Software die Kosten eines Kleinwagen überschreitet, dann sieht man eine Entwicklungslücke, die es noch zu füllen gilt. Wenn sich jedoch der Bedarf von Servicerobotern ähnlich dem Bedarf von Computern in den letzten 20 Jahren entwickelt, dann werden die Produkionskosten und Preise für Serviceroboter rasch sinken.

In den ersten Generationen von kommerziellen Servicerobotern werden bekannte und erprobte Techniken und Bauteile zum Einsatz kommen. Ein radgetriebener Antrieb, Abstandsensoren mit Ultraschall- oder Laser-Technik und eine oder mehrere Kameras werden für diese Geräte verwendet werden. In diesem Sinne wird in der weiteren Arbeit unter Serviceroboter in erster Linie ein derartiger Roboter verstanden. Die Ergebnisse und Überlegungen sind aber keineswegs nur auf derartige radgetriebenen Roboter beschränkt.

2.3 Bewegungssteuerung von mobilen Robotern

Ein mobiler Roboter besteht funktionell aus einer Reihe von Sensoren, Aktoren und einer Steuerung, die die Teile verbindet. Alle Teile sind in einem Chassis untergebracht. Die verwendete Steuerung hat zentrale Bedeutung, da sie bestimmt, in wie weit der Roboter in der Lage ist in seiner Umgebung zu navigieren und seine Tätigkeit selbstständig auszuführen.

2.3.1 Diskrete Bewegungssteuerung

Die einfachste Form einen mobilen Roboter zu navigieren ist die Abarbeitung einer Liste von diskreten Bewegungsbefehlen. Diese Befehle setzten sich aus Rotations- und Translationsanteilen zusammen. Bei einem radgetriebenen Roboter mit Differentialantrieb[1] wären das beispielsweise die Befehle `TURN(90.0)` und `MOVE(1.5)`, die den Roboter 90 Grad drehen bzw. 1,5 m weit fahren lassen. Jeder Befehl der Liste wird als atomare Operation abgearbeitet. Eine Unterbrechung kann durch eine Störung aus der Umwelt, z.B. ein Hindernis, erfolgen. Allerdings terminiert eine solche Störung die Abarbeitung der Bewegungsbefehle und eine weitere Steuerung des Roboters ist erst nach einer Rückkehr in einen definierten, nicht gestörten Zustand möglich. Eine Steuerung dieser Art kann nicht in einer unbekannten oder einer sich dynamisch ändernden Umgebung verwendet werden.

2.3.2 Kontinuierliche Bewegungssteuerung

Das Gegenstück zu den diskreten Bewegungsbefehlen sind die kontinuierlichen Bewegungsbefehle. Kontinuierlich bedeutet hierbei, dass sich der Roboter mit gewünschten Rotations- und Translationsgeschwindigkeiten ständig bewegt. Die kontinuierlichen Bewegungsbefehle variieren

[1] Gängige Antriebe für mobile Roboter sind der Differentialantrieb (quasi-holonom), der Synchronantrieb (holonom), der Omnidirektionalantrieb (holonom) und Antriebe mit Ackermann Lenkung (nicht holonom). Holonomes Verhalten bedeutet, dass sich der Roboter direkt in jede Richtung bewegen kann. Quasi-holonome Roboter müssen sich erst in die Fahrtrichtung drehen, verändern bei dieser Bewegung aber nicht ihre Position.

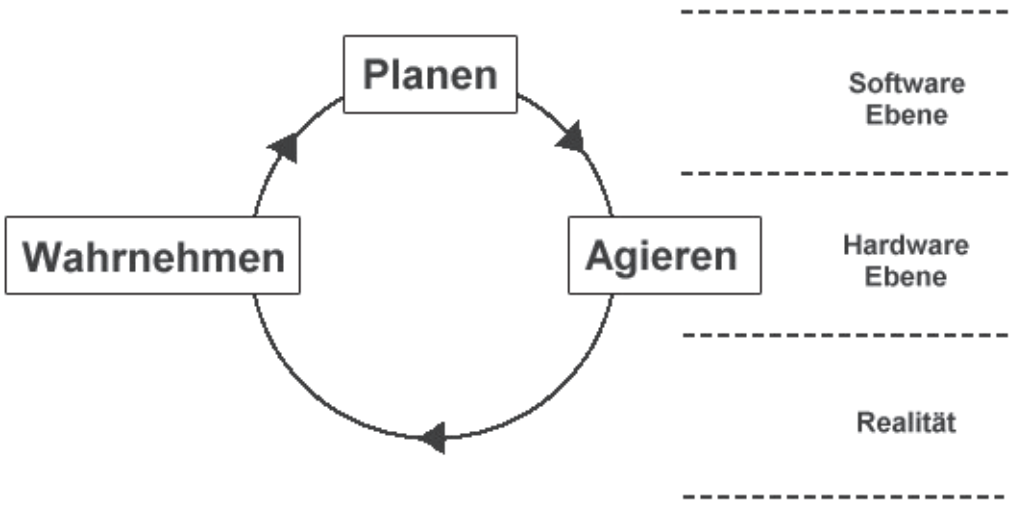

Abbildung 2.3: Schema des hierarchischen Paradigmas.

die Geschwindigkeit einer Bewegung. Bei einem radgetriebenen Roboter mit Differentialantrieb lauten die Befehle `ROTATE(180.0)`, `DRIVE(2.0)` und `STOP`, wodurch sich der Roboter mit 180°/s dreht, mit 2 m/s fährt bzw. stehen bleibt. Die Fahrbefehle können gleichzeitig ausgeführt werden, dadurch können Kurven gefahren werden.

2.4 Steuerungskonzepte für mobile Roboter

Für die Navigation eines autonomen Roboters in einer unbekannten oder nicht-statischen Umgebung benötigt man ein Steuerungskonzept, das auf einer kontinuierlichen Bewegungssteuerung basiert. Die Umwelt wird dabei mit geeigneten Sensoren wahrgenommen und der Roboter reagiert auf die Änderungen in der Umgebung. Drei Architekturen mit unterschiedlichen Beziehungen zwischen den drei Grundbausteinen *Wahrnehmen*, *Planen* und *Agieren* sind bekannt.

2.4.1 Das hierarchische Paradigma

Ein System das dem hierarchischen Paradigma folgt ist top-down strukturiert und arbeitet mit Planung. Das Schema *Wahrnehmen - Planen - Agieren* wird zyklisch durchgeführt: Die Umwelt wird mit Sensoren erfasst, der Roboter plant anhand der Daten seinen nächsten Schritt und führt entsprechende Bewegungen aus (Abbildung 2.3). Grundlage für den Planungsschritt bildet ein globales Modell der Welt, das die reale Umwelt symbolisch abbildet. Im Wahrnehmungsschritt wird das Modell anhand der Sensordaten aktualisiert.

Mit einem Ansatz dieser Art kann man theoretisch beliebig komplexe Probleme lösen, es muss nur die symbolische Abbildung der realen Umwelt entsprechend exakt sein. Diese Anforderung bereitet zwei Probleme: Zum Einen ist die korrekte symbolische Interpretation der Sensordaten in das globale Modell nur schwer, und vermutlich nie vollständig, möglich. Zum Anderen benötigen bereits einfache Aufgaben eine rechenaufwendige Modellierung der Umwelt. Der für komplexe Aufgaben notwendige Detailreichtum des globalen Modells ist mit heutigen Rechnern weder erfassbar noch abbildbar.

Abbildung 2.4: Schema des reaktiven Paradigmas.

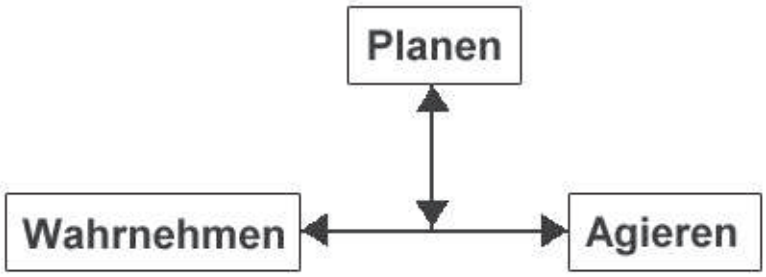

Abbildung 2.5: Schema des hybriden Paradigmas.

2.4.2 Das reaktive Paradigma

Das System basiert auf einer Menge von Reiz-Reaktionsverbindungen (Abbildung 2.4), die als parallele Prozesse arbeiten. Diese Behaviors berechnen ihre eigene Reaktion basierend auf den wahrgenommenen Sensordaten ohne andere Behaviors zu beachten. Die Kombination der Reaktionen aller Behaviors ergibt die Reaktion des Roboters.

Der Vorteil eines reaktiven Systems ist die hohe Reaktionsgeschwindigkeit mit der auch in dynamischen Umgebungen agiert werden kann. Die benötigte Rechenleistung ist zudem gering, was einem autonomen Roboter mit begrenzten Ressourcen entgegenkommt. Die Einsatzmöglichkeit von Behaviors wird aber durch ihre Intelligenz begrenzt. Analog zur Biologie können auch kombinierte Reiz-Reaktionsverbindungen ohne Verstand nur einfache Probleme effektiv lösen.

2.4.3 Das hybride Paradigma

Das System hat eine Planungsebene in der komplexe Aufgaben in Einzelschritte zerlegt werden, die dann mit einem reaktiven Ansatz lösbar sind. Diese Einzelaufgaben werden von einer ausgewählten Menge von Behaviors erledigt. Zusätzlich überwacht die Planungsebene den Fortschritt und schaltet nach jedem erfolgreich erledigten Einzelschritt zum Nächsten weiter. Schematisch ist das in Abbildung 2.5 dargestellt.

Bei einem hybriden System sollen die Vorteile von rein planenden und rein reaktiven Systemen kombiniert werden: Der Roboter kann sehr schnell reagieren und ist in dynamischen Umgebungen einsetzbar. Trotzdem bleibt sein Verhalten zielorientiert. Ineffektives oder falsches Verhalten kann von der Planungsebene erkannt und korrigiert werden. Dieser Lösungsansatzes scheint für künftige Steuersysteme vielversprechend, die konkrete Aufteilung zwischen Behaviors und Planer ist aber weiter Thema der wissenschaftlichen Diskussion.

2.5 Ausgewählte Architekturen

Im Folgenden werden drei Architekturen zur Bewegungssteuerung eines mobilen Roboters präsentiert. Die drei Architekturen wurden der Reihe nach entwickelt und zeigen deutlich die Entwicklung von einem rein reaktiven Paradigma in Richtung eines hybriden Paradigmas.

2.5.1 Subsumption architecture

Die *Subsumption architecture* von Rodney Allen Brooks [Bro86] stellt ein aus der Biologie abgeleitetes behavior-basiertes Steuerungskonzept dar. Ein gewünschtes, kompliziertes Verhaltensmuster setzt sich aus unterschiedlich abstrakten Behaviors auf verschiedenen Schichten zusammen (Abbildung 2.6). Die einzelnen Behaviors innerhalb einer Schicht funktionieren nach dem Prinzip der Reiz-Reaktionsverbindungen. Die Architektur versteht sich als Gegenteil des hierarchischen Paradigmas. Jede Schicht hat gleichermaßen Zugriff auf die Sensordaten und erzeugt einen gewünschten Output zur Motorsteuerung. Eine übergeordnete Schicht hat aber zusätzlich die Möglichkeit, das Verhalten einer untergeordneten Ebene durch Gewichtung zu verändern. Die untersten Schichten können damit schnelle Reaktionen (Reflexe) erzeugen, während die obersten Schichten eine gestellte Aufgabe zielorientiert zu lösen versuchen.

Brooks versuchte mit der Subsumption architecture folgende Anforderungen eines autonomen mobilen Roboters zu erfüllen:

- Das Erledigen von multiplen Zielen zur selben Zeit. Ein Roboter wird oft mit multiplen Forderungen konfrontiert sein, deren Priorität sich mit der Zeit verändert (z.B. in einer Gefahrensituation).
- Der Input des Roboters stammt meist von multiplen Sensoren. Jeder Sensor ist mit einem Fehler behaftet und die verschiedenen Sensorwerte können sich inkonsistent überlappen.
- Ein Roboter muss robust sein. Sensorausfälle oder drastische Änderungen der Umgebung dürfen nicht zu Fehlverhalten führen. Der Roboter sollte im Rahmen seiner Möglichkeiten weiter operativ bleiben.
- Die Architektur soll Additivität erlauben. Eine bestehenden Steuerung soll mit zusätzlichen Sensoren und zusätzlichen Aufgaben erweitert werden können.

2.5.2 Motor Schemas

Motor Schemas ist eine reaktive Steuerungsarchitektur von Ronald C. Arkin [Ark89]. Die Architektur basierte überwiegend auf Theorien der Kognitionswissenschaften. Motor Schemas stellen eine Adaption des philosophischen Modells der Schema-Theorie auf Roboter dar. Die Schema-Theorie wurde erstmals im 18. Jahrhundert von Immanuel Kant formuliert und beschreibt Schemas als ein Mittel, um sensorische Wahrnehmung zu kategorisieren und in Wissen umzuwandeln. Die Grundstruktur hinter der Schema-basierten Robotik sind behavioral primitives, eine Menge von einfachen Behaviors, die vom System zur Verfügung gestellt werden. Diese Verhaltensmuster arbeiten parallel und verteilt, um den Roboter intelligent auf Umweltreize reagieren zu lassen. Aus diesen einfachen *behavioral primitives* können durch Kombination auch komplexere Verhaltensmuster (*Perceptual Schemas*) konstruiert werden (Abbildung 2.7).

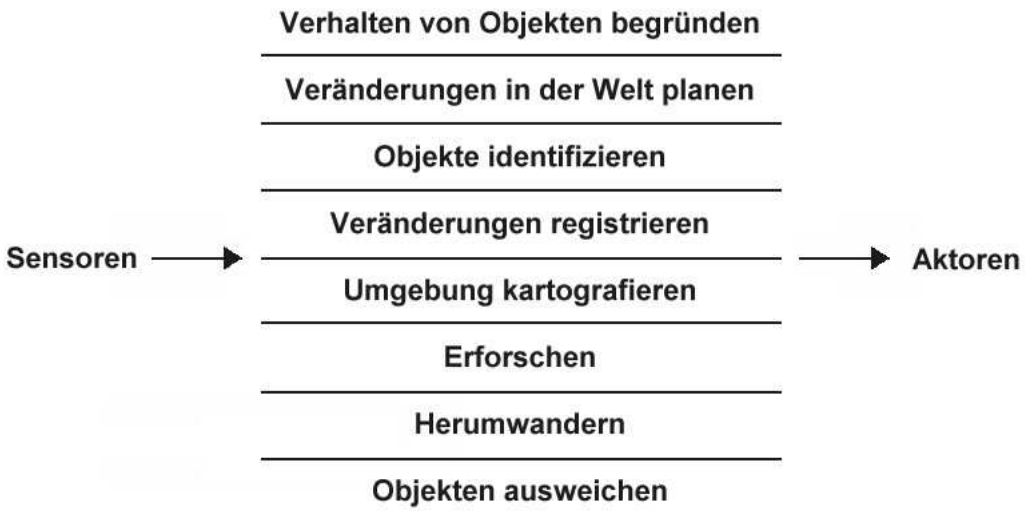

Abbildung 2.6: Die *Subsumption architecture* in einem Beispiel von Brooks [Bro86]. Jede Schicht will ihr Verhalten erfüllen und kann dazu die darunter liegenden Schichten beeinflussen.

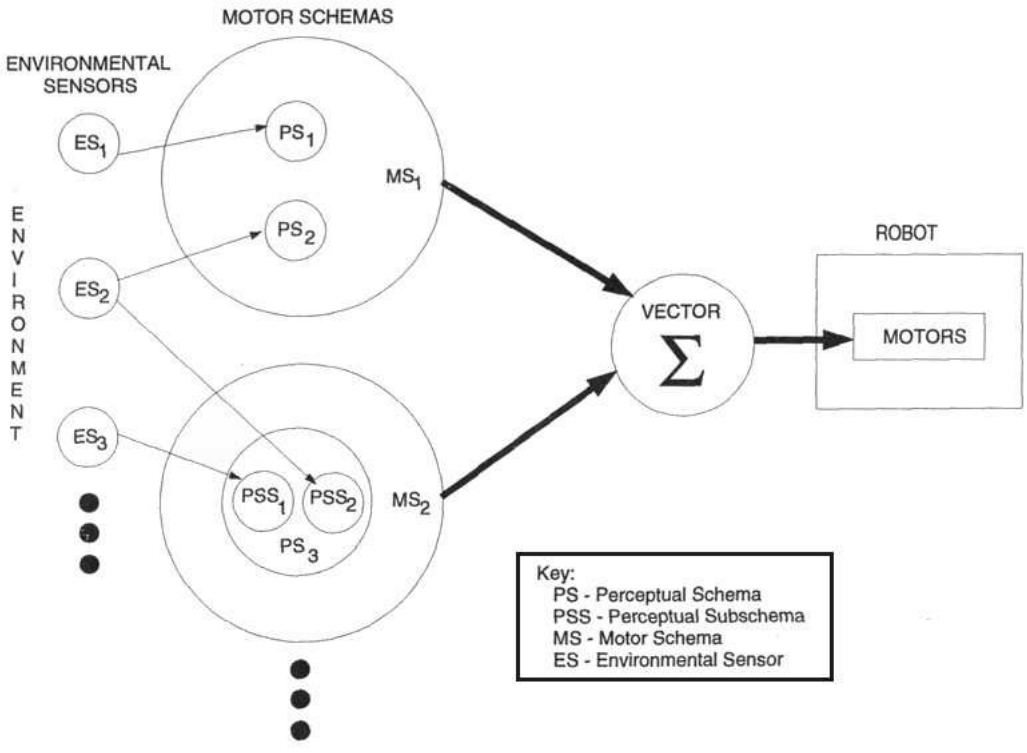

Abbildung 2.7: Eine schematische Darstellung der *Motor Schemas* Architektur von Arkin [Ark98]. Beginnend von links wird die Umgebung mit Sensoren (ESx) wahrgenommen. Die Daten werden an zwei Motor Schemas (MSx) weitergegeben. In den Motor Schemas befinden sich parallel und hierarchisch arbeitende *Perceptual Schemas* bzw. *Perceptual Subschemas* (PSx bzw. PSSx). Das Ergebnis jedes Motor Schemas ist ein Bewegungsvektor. Durch Vektoraddition wird ein Ergebnisvektor gebildet und an die Motoren weitergegeben.

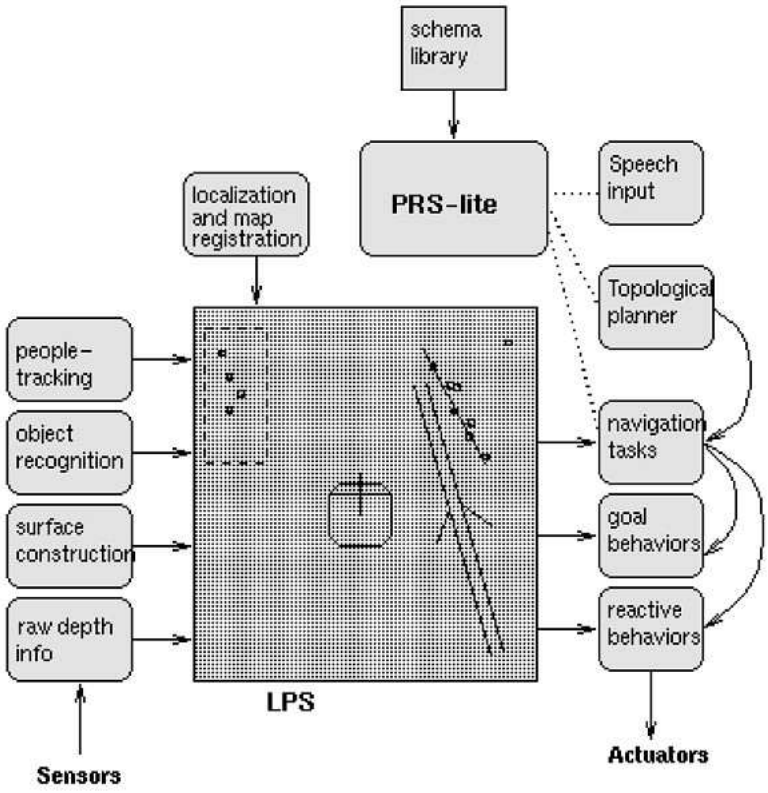

Abbildung 2.8: Die *Saphira* Systemarchitektur. Im Zentrum ist der *Local Perceptual Space* (LPS), eine geometrische Interpretation der Umwelt, dargestellt. Links sind die Input-Routinen zur Wahrnehmung der Umwelt, rechts die Action-Routinen, die die Aktoren stimulieren. Das *Procedural Reasoning System* (PRS-lite) koordiniert das Ganze. Die Abbildung wurde [GCJK97] entnommen.

Die Motor Schemas Architektur ist auf die Bewegungssteuerung eines Roboters limitiert, das Resultat ist immer ein Bewegungsvektor.

Motor Schemas unterscheiden sich speziell durch folgende Kriterien von anderen Architekturen:

- Die Antworten jedes einzelnen Verhaltensmusters ist ein Bewegungsvektors.
- Die Antworten werden durch Vektoraddition zusammengefasst.
- Es gibt keine vordefinierte Hierarchie. Zur Laufzeit werden die einzelnen Verhaltensmuster durch die Aufgabenstellung und Situation gewichtet. Dadurch entspricht die Architektur eher einem dynamischen Netzwerk als einer Schichtarchitektur.
- Innerhalb der Motor Schemas werden keine Entscheidungen getroffen, statt dessen beteiligen sich alle Behaviors mit veränderlichem Anteil am resultierenden Bewegungsvektor.

2.5.3 Saphira Architektur

Die *Saphira* Architektur von Kurt Konolige ist ein integriertes System für sensorische Wahrnehmung und zur Roboter Kontrolle [GCJK97]. Im Zentrum der Saphira Architektur sitzt der

Local Perceptual Space (LPS), eine geometrische Interpretation der Umgebung des Roboters. Als LPS dienen üblicherweise:

- Eine Karte auf Basis einer gitter-basierten Beschreibung eines *occupancy grid*. Bei dieser Technik wird die Umgebung durch ein kartographisches Gitter in kleine Teile zerlegt. Jede dieser kleinen Zellen wird entsprechend der Sensor-Informationen als besetzt oder frei markiert.
- Eine analytische Beschreibung der Umwelt in der Form, wie sie von den Sensoren wahrgenommen wird.
- Eine semantische Beschreibung der Umwelt. (Dieser Raum hat eine Tür. Hinter der Tür befinden sich ein Gang. usw.)

Der LPS wird von Sensoren über Input-Routinen mit neuen Daten versorgt und dient Action-Routinen als Grundlage für die Ansteuerung der Aktoren. Die Action-Routinen werden vom *Procedural Reasoning System* (PRS-lite) kontrolliert. In der Saphira Systemarchitektur (Abbildung 2.8) sieht man deutlich das hybride Design. Einerseits gibt es die hierarchische Kontrolle der Action-Routinen durch das PRS-lite, andererseits ermöglicht die Kette *Sensor - Input-Routinen - LPS - Action-Routinen - Aktoren* ein reaktives Verhalten.

2.6 Zusammenfassung

Serviceroboter werden in einigen Jahren Teil des Alltags werden. Über Form und Fähigkeiten dieser mechanischen Helfer können zur Zeit nur Vermutungen aufgestellt werden. Eine Reihe von besonderen Anforderungen und Einschränkungen können aber jetzt schon formuliert werden, wie etwa die Forderung nach robustem und sicherem Verhalten, entsprechenden sozialen Schnittstellen zur Kommunikation mit Menschen und die Forderung nach beschränkten Herstellungskosten, um als Massenprodukt hergestellt werden zu können. Ein Kernpunkt eines solchen Roboters wird die Bewegungssteuerung sein. Den Weg zu einer robusten Architektur zeigen die praktische Beispiele Subsumption architecture, Motor Schemas und die Saphira Architektur.

Kapitel 3

Die behavior-basierte Robotik

Die Grundbausteine jeder behavior-basierten Architektur sind die Behaviors. Während in Kapitel 2 das Thema die Analyse der reaktiven Systemarchitektur von außen war, wird Kapitel 3 ein Robotersystem aus dem Blickwinkel der Behaviors betrachten. Den Einstieg bildet eine historische Betrachtung der behavior-basierten Robotik. Der Wandel des Begriffs *Behavior* im Laufe der Jahre spiegelt die verschiedenen Aspekte von Behaviors wieder. Danach geht es in medias res: Die Definition eines Behaviors und die mathematische Beschreibung der Reiz-Reaktionsverbindung bilden die Grundlage für die behavior-basierten Robotik. Es werden die verschiedenen Darstellungsformen von Behaviors beschrieben, die im Kapitel 5 Verwendung finden. Die gängigen Koordinatoren zur Kombination von Behaviors werden vorgestellt und den Abschluss bilden die beiden gängigsten Varianten, um Behaviors zu kodieren: Die Methode der virtuellen Potentialfelder und die Implementierung mit Hilfe einer Fuzzy Control.

3.1 Die Geschichte der behavior-basierten Robotik

Die heutige behavior-basierte Robotik entwickelte sich aus behavioristischen Ansätzen der Psychologie in der ersten Hälfte des 20. Jahrhunderts. Trotz des wissenschaftliche Disziplinen übergreifenden Starts, der dem jungen Forschungszweig eine Wissensbasis in die Wiege legte, sollte es noch weitere 50 Jahre dauern, bis der behavior-basierte Ansatz in der Robotik angewendet wurde. Bei der behavior-basierten Robotik steht die Reiz-Reaktionsbeziehung mit der Umwelt im Mittelpunkt. Im Gegensatz dazu versucht die kognitive Robotik, basierend auf den Kognitionswissenschaften in der Psychologie, künstlichen Roboterintelligenzen mit abstrakten Wissen wie Zielen, Präferenzen und Annahmen arbeiten zu lassen. Es wird sich zeigen, ob die beiden widersprüchlichen Ansätze in der Robotik ihren psychologischen Vorbildern folgen werden und sich ein Ansatz finden lässt, der die behavior-basierte und kognitive Robotik verbindet.

3.1.1 Der behavioristische Ansatz in der Psychologie (1900 - 1930)

Anfang des 20. Jahrhunderts führte Ivan Petrowitsch Pawlow im Rahmen seiner Arbeit über Konditionierung seine klassischen Experimente mit Hunden durch. Er konnte nachweisen das der Speichelfluss von Hunden als Reaktion auf einen Futterreiz (ein angeborener Reflex) nach einem entsprechenden Lernvorgang auch durch eine bedingte Reaktion auf einen anderen Reiz (Glockenläuten) hervorgerufen werden kann. Einige Merkmale des Pawlowschen Reflexbegriffs

Abbildung 3.1: Das Modell der Reiz-Reaktionsverbindung mit Black Box. Es findet praktisch unverändert in der behavior-basierten Robotik seine Anwendung.

finden sich auch in der heutigen behavior-basierten Robotik wieder:

- Deterministische Natur: Jede Reaktion bedarf einer Ursache.
- Unbedingter Reflex: Automatisch auf einen Reiz folgende, unbedingte Reaktion.
- Bedingter Reflex: Individuell erworbene Verbindung zwischen Reiz und Reaktion, die erlernt und beeinflussbar ist.
- Reflexkette: Reaktionen auf einen Reiz lösen einen weiteren Reflex aus. Solche Ketten können untereinander verkoppelt sein und bilden die Grundlage des Verhaltens.

In den folgenden Jahren entstand eine Gegenthese zur klassischen Konditionierung: Der Behaviorismus. Er wollte das Bewusstsein nicht als legitimen Forschungsgegenstand der Psychologie anerkennen und behandelte die Vorgänge, die sich zwischen Reiz und Reaktion abspielen, so, als wären sie in einer Black Box verborgen. Der Organismus wurde dabei als eine Maschine angesehen, in die man nicht hineinsehen kann und deren Funktionsweise nur aus Input (Reiz) und Output (Reaktion) zu erschliessen ist. Abbildung 3.1 zeigt dieses Modell einer Reiz-Reaktionsverbindung, die sich analog in der behavior-basierten Robotik als sensomotorische Verbindung wiederfindet.

Bis zur Mitte des 20. Jahrhunderts wird der Behaviorismus beinahe zum herrschenden Paradigma, dann revolutionierte die Kognitionswissenschaft mit dem Funktionalismus die Psychologie. Zu diesem Zeitpunkt hatten aber bereits Ingenieure, Mathematiker, Philosophen und Psychologen die Übereinstimmung der Arbeitsweise von Computern und des menschliches Geistes entdeckt. In Folge entwickelte sich die Fachrichtung der Künstlichen Intelligenz und als Teil davon die behavior-basierte Robotik. In Form des Neobehaviorismus als Verquickung von Behaviorismus und Kognitivismus erlebten am Ende des 20. Jahrhunderts Teile des Behaviorismus ein Comeback innerhalb der Psychologie [Sta97].

3.1.2 Bettwanzen und Schildkröten - Die ersten behavior-basierten Roboter (1930 - 1960)

Der vermutlich erste Prototyp eines behavior-basierten Roboters stammt vom Psychologen Edward C. Tolman aus dem Jahr 1939 [Tol39]. Das als „schematische Bettwanze“ bezeichnete Modell basiert auf Tolmans Theorie des zielgerichteten Behaviorismus aus den frühen 1920er Jahren. Die schematische Bettwanze bewegt sich in ihrer Umwelt basierend auf zwei Arten von Vektoren, Orientierung und Fortschritt, welche aus mittels Lichtsensoren wahrgenommenen Umweltreizen errechnet werden. Tolmans Forschung zielte darauf ab, den Zusammenhang

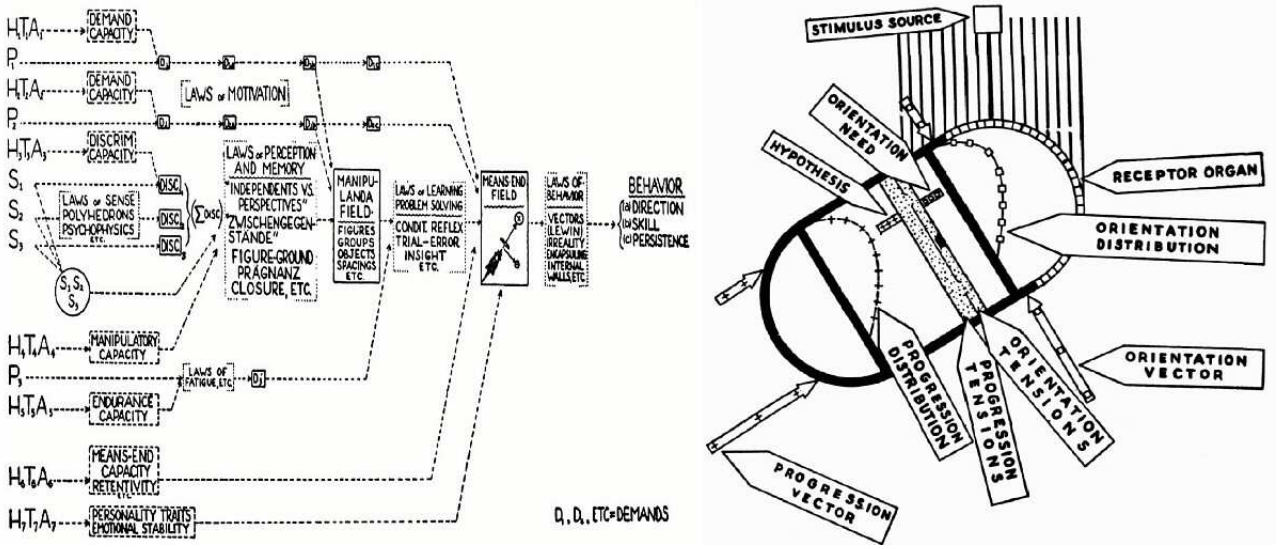

Abbildung 3.2: Links Tolmans Theorie des zielgerichteten Behaviorismus als Schaubild der schematischen Bettwanze. Rechts die schematische Bettwanze und ihre beeinflussenden Vektoren. Die Verteilung der Lichtreize wird mit Lichtsensoren wahrgenommen und in der *orientation distribution* abgebildet. Über die *orientation tension* wird die *orientation need* gebildet, aus der wiederum zwei Vektoren für die Orientierung der linken und rechten Seite berechnet werden. Analog werden zwei Vektoren für den Fortschritt berechnet.

zwischen einer Reiz-Reaktionsbeziehung und höherrangigen Faktoren wie Motivation, Wahrnehmung oder Absicht zu untersuchen. Tolman entwickelte dazu eine Formel, um den Wert eines Behaviors B aus den Umgebungsreizen S, dem physiologischem Antrieb P, einer Vererbung H, einem früheren Training T und Gemeinsamkeiten/Alter A zu berechnen:

$$B = f(S, P, H, T, A) \tag{3.1}$$

Die Formel lässt sich in einem schematischen Schaubild (Abbildung 3.2) mit den verschiedenen Faktoren in parallelen sensormotorischen Verbindungen und Knoten darstellen, worauf sich auch der Name „schematische Bettwanze“ bezieht. Dieser frühe Ansatz beachtet schon Probleme, die es auch in der modernen behavior-basierten Robotik zu lösen gilt: Wie erzielt man ein intelligentes Verhalten bei parallelen oder übereinstimmenden Reiz-Reaktionspfaden? Wie kann man den Output sinnvoll koordinieren und wie kann man mit einem zielorientieren Behavior eine Absicht implementieren? Die Funktionsfähigkeit der schematischen Bettwanze wurde von Arkin und Endo mittels Simulation und im Experiment nachgewiesen [EA01].

In den Jahren 1948-49 entwickelte der Neuropsychologe und Robotiker Walter Grey Walter ein Paar autonome Roboter, *Elmer* und *Elise*, um den Zusammenhang zwischen Speicherzellen und Verbindungen zu untersuchen [Wal53]. Die wegen ihrer Form und ihres Bewegungsmusters als Schildkröten bezeichneten Geräte arbeiteten mit Vakuumröhren-Technik und waren mit Lichtsensoren ausgestattet. Sie konnten auf Lichtreize sensormotorisch reagieren und zu ihrer Ladestation zurückfinden, wenn die Batterien schwächer wurden (Abbildung 3.3). Die Ergebnisse von Walter zeigten, dass komplexe Verhaltensmuster auch mit wenigen Speicherzellen möglich sind, wenn viele Verbindungen zwischen diesen Speicherzellen existieren. Nachfolgemodelle der ersten Schildkröten wurden der Welt beim Festival of Britain im Jahre 1951 präsentiert.

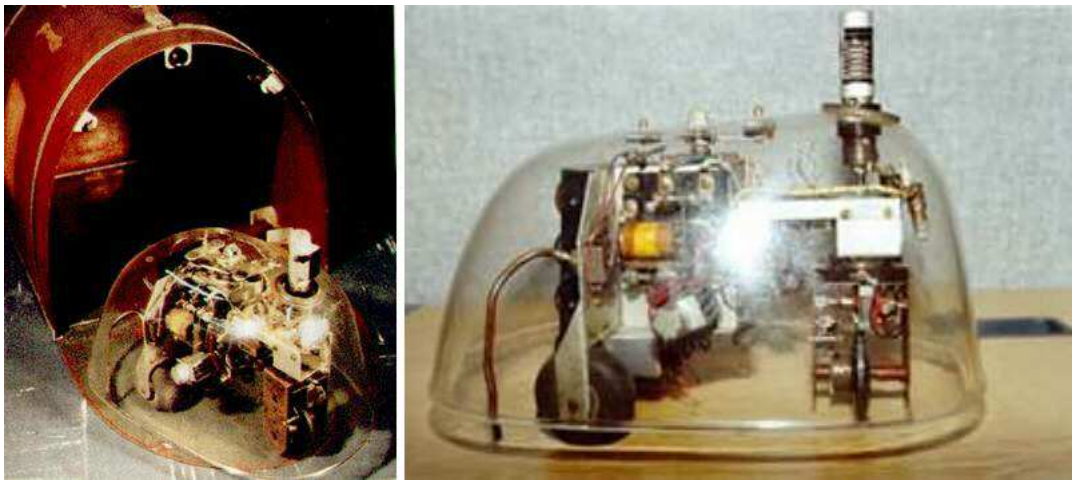

Abbildung 3.3: Eines der letzten erhaltenen Modelle von Walter. Links die „Garage“ mit Ladestation, in die der Roboter mit schwachen Batterien selbst zurückkehrt.

3.1.3 Gedankenexperimente und Rebellen - Die moderne behavior-basierte Robotik (1980 - 1990)

Anfang der 1980er Jahre erdachte der Biokybernetiker Valentino Braitenberg eine Reihe von Gedankenexperimenten, die er 1984 publizierte [Bra84]. Seine 14 hypothetischen Fahrzeuge - heute als *Braitenberg Vehicles* bekannt - bestehen aus einfachen Grundbausteinen: Sensoren, Antriebe und Verbindungen zwischen diesen (Abbildung 3.4). Trotz dieses einfachen Designs scheinen die Fahrzeuge bei ihrer Bewegung verschiedene Verhaltensmuster wie Angst, Aggression, Liebe, Voraussicht und sogar Optimismus zu zeigen. Die Fahrzeuge sind dabei nach evolutionärem Entwicklungsstadium gereiht, analog der tierischen Evolution vom Einzeller zum Primaten. Die Nachvollziehbarkeit der Gedankenexperimente inspirierte nicht nur Robotiker zu neuen Überlegungen, sondern auch Lehrende: Die Braitenberg Vehicels (im besonderen die Fahrzeuge 1 bis 4) sind heute „klassische“ Übungsbeispiele der Robotik. Die Praxistauglichkeit der Gedankenexperimente wurde in zahlreichen Simulationen und praktischen Experimenten bewiesen.

Die Arbeiten des Robotikers Rodney Allen Brooks führten schließlich zu einem fundamentalen Richtungswechsel im Forschungsbereich der Künstlichen Intelligenz. Ab 1986 publizierte er eine Reihe von Arbeiten, in denen er das Fundament der klassischen Künstlichen Intelligenz, nämlich die symbolische Abbildung der Realität, als ungeeignet für autonome Roboter darstellte [Bro90]. Laut Brooks ist „die Welt ihr eigenes bestes Modell“. Anstatt einer symbolischen Beschreibung der Realität soll die entsprechend oft und genau abgetastete reale Welt direkt als Basis für intelligente Reaktionen eines Roboters dienen. Er erhob die Einfachheit zu einer Tugend und forderte mit der Behauptung „Intelligenz liegt im Auge des Betrachters“ einen neuen Ansatz für künstliche Intelligenz. Als Ersatz für die bisherigen „horizontalen“ Steuerung von Robotern, Wahrnehmen - Planen - Exekutieren, präsentierte er einen neuen Ansatz, die *Subsumption architecture*, ein Schichtmodell mit verschiedenen Abstraktionsebenen [Bro86]. Auf Basis der Subsumption architecture wurde eine Reihe von Robotern zwecks Proof of Concept entwickelt. Diese Roboter zeigten die von Brooks erwartete Form von „Intelligenz ohne Verstand“ [Bro91].

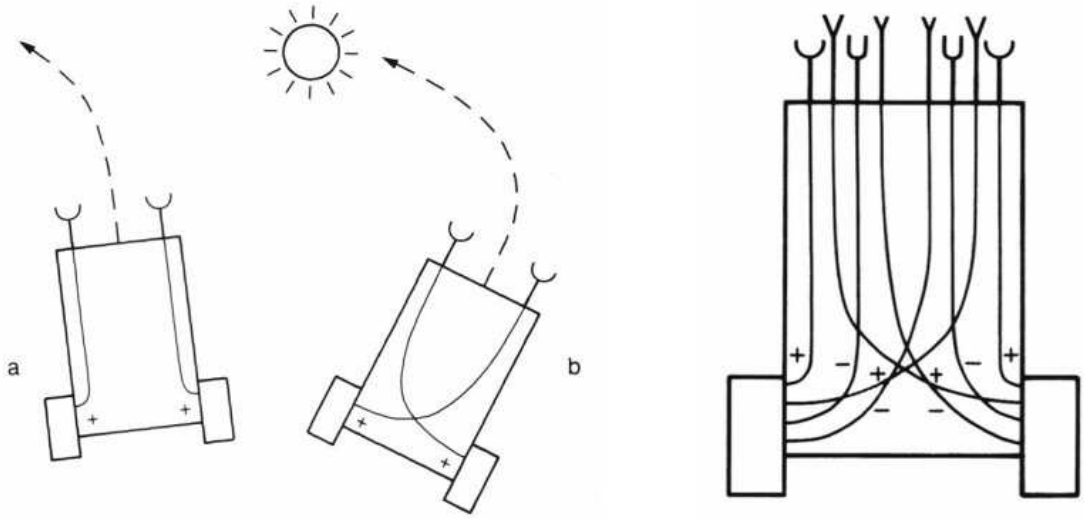

Abbildung 3.4: Links das Braitenberg Vehicle 2. Es zeigt je nach Verschaltung von Sensor zu Motor ängstliches (2a) oder aggressives (2b) Verhalten gegenüber der Reizquelle (Lampe). Rechts das Braitenberg Vehice 3c, das mit vier verschiedenen Sensortypen (Temperatur, Licht, Sauerstoffkonzentration und organisches Material) mit unterschiedlicher Verschaltung zu den Motoren ausgestattet ist. Das resultierende Verhalten suggeriert ein intelligentes Wertesystem.

3.1.4 Mit Behaviors ins 21. Jahrhundert (1990 - 2006)

In den 1990ern hatte die behavior-basierte Robotik bereits einiges vorzuweisen: Einen von der Psychologie entlehnten und von klugen Köpfen weiterentwickelten Grundgedanken, eine wissenschaftliche Rebellion gegen das symbolfixierte Establishment innerhalb des Bereichs der Künstlichen Intelligenz und eine Reihe erfolgreich implementierter Beispiele. Was immer noch fehlte war eine Basis, ein umfassender Ansatz mit mathematischer Fundierung, auf den leichter aufzubauen war, als auf eine Menge schwach vernetzter Einzelarbeiten. Diese Aufgabe fiel Ronald C. Arkin zu, dessen Arbeit sich mit einem Gesamtbild der Aspekte der behavior-basierte Robotik beschäftigt. In seinen Arbeiten wird er auch nicht müde auf die Bedeutung der Forschung an realen Robotern hinzuweisen, anstatt sich nur mit anfänglich hilfreichen, aber im Endeffekt unzulänglichen, Simulationen zu beschäftigen. Seine *Behavior-Based Robotics* [Ark98] stellen das Referenzwerk für die aktuelle Forschung im gleichnamigen Bereich dar. Ein weiteres Ergebnis seiner Arbeit stellt die Steuerungsarchitektur *Motor Schemas* dar.

Ende der 1990er Jahre lieferte Kurt Konolige mit seiner *Saphira* Architektur für mobile Roboter eine aktuelle praktische Implementierung der behavior-basierten Robotik in einer hybriden Systemarchitektur [Kon98]. Für die Saphira Architektur gibt es eine eigene Programmiersprache, *Colbert*, in der Anwender eigene Behaviors programmieren können [Kon97]. Die Programmierung und die Saphira Architektur unterstützen dabei nicht nur Entfernungssensoren (Sonarsensoren, Laserscanner), sondern auch Stereo-Kamerasysteme. Die kommerziellen Roboter der Firma *ActivMedia Robotics* (inzwischen *Mobile Robotics*) werden mit einer Steuerung auf Basis der Saphira Architektur geliefert. Ein bekanntes Robotermodell von ActivMedia Robotics sind die Geräte der Pioneer Serie, die derzeit in der dritten Generation erhältlich sind. Diese universellen mobilen Plattformen werden gerne als Forschungsroboter im universitären Bereich benutzt.

Abbildung 3.5: Ein einfaches Reiz-Reaktionsdiagramm.

3.2 Definition eines Behaviors

Ein auf dem reaktiven Paradigma basiertes Robotersystem setzt sich aus einer Menge von Behaviors zusammen (Kapitel 2.4.2). Ein Behavior in der Robotik beschreibt eine Reiz-Reaktionsverbindung, einem Reflex nicht unähnlich.

Die Form des Reiz-Reaktionsdiagramms wurde aus dem behavioristischen Ansatz in der Psychologie übernommen, anstelle der Black Box sitzt aber jetzt das Behavior (Abbildung 3.5). Die Art der Implementierung eines Behaviors obliegt dem Systemdesigner. Es gibt aber eine Reihe von Eigenschaften die Behaviors in einem reaktiven Robotersystem erfüllen müssen [Ark98]:

- Das Design von reaktiven Robotersystemen und Behaviors beruht auf Konzepten aus der Biologie und Psychologie.
- Behaviors sind die Grundbausteine für Roboteraktiviät in einem reaktiven Robotersystem.
- Ein Behavior besteht aus einer sensormotorischen Verbindung.
- Reaktive Robotersysteme verzichten auf eine symbolische Abbildung der Umwelt. Die Behaviors benötigen kein abstraktes, repräsentatives Wissen, um eine Reaktion zu berechnen.
- Behaviors verzichten außerdem auf ein zeitliches Vorwissen.
- Die reaktiven Robotersysteme sind aus der Sicht des Software-Designs inhärent modular. Der Systemdesigner kann die Fähigkeiten des Roboters durch neue Behaviors erweitern, ohne die alten Behaviors ändern oder löschen zu müssen.

Diese Eigenschaften beschreiben Behaviors eines rein reaktiven Robotersystems. In hybriden Architekturen und besonderen reaktiven Systemen werden manche der Eigenschaften aber freier ausgelegt oder sogar absichtlich ignoriert. Ein Beispiel dafür wären Behaviors mit einem „Kurzzeitgedächtnis“, um damit Deadlock-Reaktionen zu verhindern.

3.3 Die Reiz-Reaktionsabbildung

Die mathematische Beschreibung eines Behavior ist das Tripel (S, R, β). S bezeichnet dabei den Bereich der wahrnehmbaren Reize, R bezeichnet den Bereich aller möglichen Reaktionen und β bezeichnet die Abbildung $\beta : S \rightarrow R$. Die Abbildung des Reizes auf die Reaktion ist eine funktionelle Transformation des Wertes vom Reiz-Wertebereich in den Reaktions-Wertebereich.

Die Roboterreaktion teilt man üblicherweise in zwei orthogonale Komponenten: Stärke und Orientierung.

- Die Stärke gibt die Größenordnung der Reaktion an. Sie kann, muss aber nicht, von der Stärke des Reizes abhängen. Die Stärke der Reaktion kann auch beispielsweise von der Reizfrequenz oder der Intention des Roboters abhängen.
- Die Orientierung gibt die Richtung der Reaktion auf den Reiz an, also beispielsweise in Richtung eines anziehenden Reizes oder entgegen der Richtung eines abstoßenden Reizes. Die Orientierung kann, aber muss nicht, von der Stärke des Reizes abhängen.

3.3.1 Der Bereich der Reaktionen: R

Jede mögliche Reaktion r (wobei $r \in R$) eines Behaviors in einem allgemeinen reaktiven System kann als sechsdimensionaler Vektor beschrieben werden. Die Elemente des Vektors entsprechen den sechs Freiheitsgraden eines allgemeinen mobilen Roboters. Für jeden Freiheitsgrad wird die Stärke der translatorischen bzw. rotatorischen Reaktion angegeben.

$$r = [x, y, z, \theta, \phi, \psi] \tag{3.2}$$

Radgetriebene mobile Roboter haben weniger als sechs Freiheitsgrade. Roboter, die sich auf dem Boden bewegen und auf der Stelle rotieren können, haben zwei translatorische und einen rotatorischen Freiheitsgrad.

$$r = [x, y, \theta] \tag{3.3}$$

3.3.2 Der Bereich der Reize: S

Jeder wahrnehmbare Reiz s (wobei $s \in S$) eines Behaviors in einem allgemeinen reaktiven Systems kann als Tupel (p, λ) dargestellt werden, wobei p ein bestimmter Typ von Reiz und λ eine Angabe über die Stärke des Reizes darstellt. Die Menge aller Reize s beschreibt die Dinge, die ein Roboter wahrnehmen und unterscheiden kann. Die Stärke des Reizes λ kann dabei auf verschiedene Arten definiert sein, z. B. binär, kategorisch oder mit Zahlenwerten. Weiters gibt es eine Reizschwelle τ, die überschritten werden muss, damit ein Reiz eine Reaktion auslöst.

3.3.3 Die Abbildungsfunktion: β

Für jedes im System aktive Behavior gibt es eine Abbildungsfunktion β, die eine Menge von Reizen auf eine Reaktion abbildet

$$\beta(s) \rightarrow r \tag{3.4}$$

β kann beliebig definiert sein. Bei Beachtung der Reizschwelle τ ergibt sich

$$\beta : (p, \lambda) \rightarrow \{\text{für alle } \lambda < \tau \text{ sei } r = [0,0,0,0,0,0] \text{ sonst } r = \text{beliebige Funktion}\} \tag{3.5}$$

Dabei bedeutet $r = [0, 0, 0, 0, 0, 0]$ keine Reaktion auf den Reiz s. Eine Abbildungsfunktion kann zusätzlich mit einem Gewichtungsfaktor g multipliziert werden, der die Stärke der Reaktion

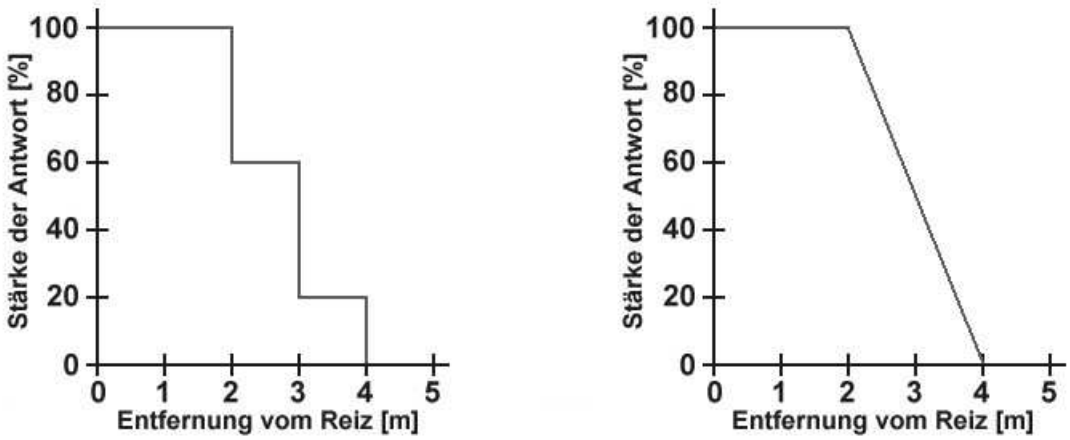

Abbildung 3.6: Beispiele für eine diskrete (links) und kontinuierliche Abbildungsfunktion. Eine praktische Anwendung für diese beiden Abbildungsfunktionen wäre ein Behavior das mit Abstandsensoren auf Objekte reagiert. Objekte näher als 2 m lösen eine volle Reaktion aus, weiter entfernte Objekte eine schwäche Reaktion. Objekte die weiter als 4 m entfernt sind werden ignoriert.

variiert. Dieser Gewichtungsfaktor kann auch auf Null gesetzt werden, um eine Reaktion zu unterdrücken.

$$r' = gr \tag{3.6}$$

Neben der Nullreaktion gibt es zwei Arten von Abbildungsfunktionen (Abbildung 3.6):

- Diskret: Die Reaktion auf einen Reiz stammt aus einer aufzählbaren Menge von Reaktionen. R besteht aus einem fixen, zählbaren Set von typischen Reaktionen.
- Kontinuierlich: Die Reaktionen zu Reizen aus dem Bereich S sind kontinuierlich auf einem Bereich R abgebildet.

In der Abbildungsfunktion steckt sehr viel der Intelligenz eines Behaviors. Bereits die Arbeiten von Walter [Wal53] und Braitenberg [Bra84] haben auf die besondere Bedeutung der Abbildungsfunktion hingewiesen. Sie haben gezeigt, das bereits mit wenigen Sensor- und Motorelementen, aber dafür vielen Verbindungen mit entsprechenden Abbildungsfunktionen zwischen den Elementen, verschiedene Verhaltensmuster konstruiert werden können.

3.4 Formelle Darstellungsformen für Behaviors

Es gibt drei gängige Methoden behavior-basierte Architekturen darzustellen [Ark98]. Jede dieser Methoden drückt die unterschiedlichen Aspekte von Behaviors besser oder schlechter aus. Die Wahl der Darstellungsform hängt daher vom gewählten Systemdesign ab. Die drei Darstellungsformen mit folgendem Beispiel illustriert: Ein Roboter soll eine Zielfahne erreichen. Dazu muss er Hindernissen ausweichen, darf aber den mit blauen Bodenbelag markierten Weg nicht verlassen.

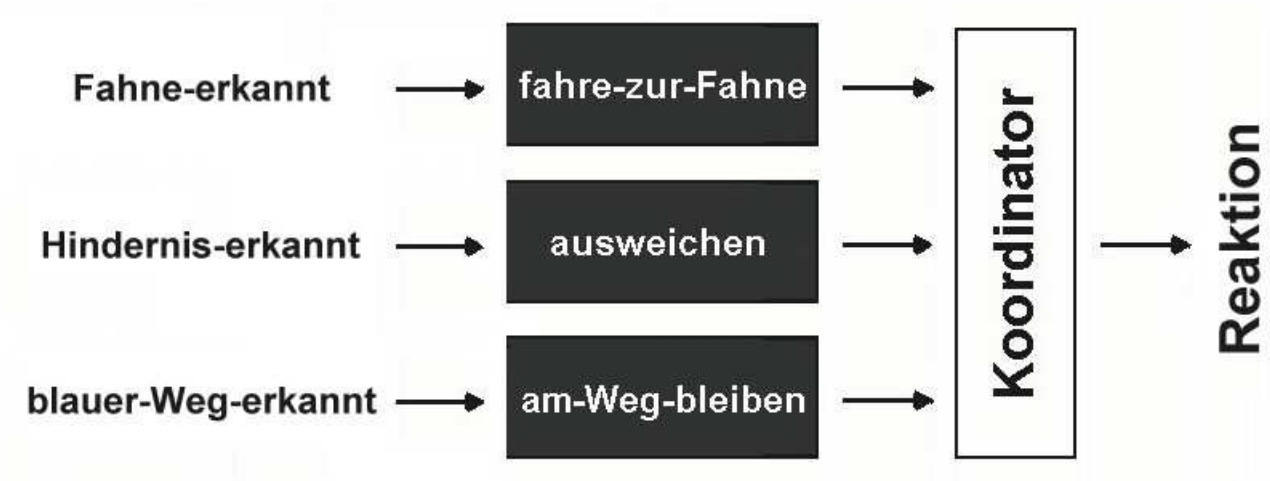

Abbildung 3.7: Reiz-Reaktionsdiagramm mit drei Behaviors, die von drei speziellen Reizen angeregt werden. Die Reaktionen der drei Behaviors werden zu einer gemeinsamen Reaktion zusammengefasst.

3.4.1 Reiz-Reaktionsdiagramm

Die Darstellungsform im Reiz-Reaktionsdiagramm ist zwar nicht sehr formell, aber dafür leicht visuell verständlich (Abbildung 3.7). Parallele Behaviors und die Verbindung zwischen diesen lassen sich leicht darstellen. Im gewählten Beispiel gibt es die drei Behaviors: `fahre-zur-Fahne`, `ausweichen` und `am-Weg-bleiben`, die von den drei speziellen Reizen `Fahne-erkannt`, `Hindernis-erkannt` und `blauer-Weg-erkannt` ausgelöst werden. Die drei Reaktionen werden von einen Koordinator zu einer gemeinsamen Reaktion summiert.

3.4.2 Funktionale Notation

Formeller als ein Reiz-Reaktionsdiagramm beschreibt die funktionale Notation die Beziehungen eines Behaviors als

$$b(s) = r \tag{3.7}$$

Das Behavior b reagiert auf den Reiz s mit der Reaktion r. In einem rein reaktiven System stellt die Zeit keinen Faktor dar. Die Reaktion erfolgt sofort auf den Reiz und das Behavior hat kein zeitliches Vorwissen. Ein Vorteil der funktionalen Notation ist, dass sie direkt in funktionelle Programmiersprachen umgesetzt werden kann. Die formelle Notation für das Beispiel der Fahrt zur Fahne lautet:

```
Fahrt-zur-Fahne-Behavior [
   fahre-zur-Fahne(Fahne-erkannt),
   ausweichen(Hindernis-erkannt),
   am-Weg-bleiben(blauer-Weg-erkannt)
] = Reaktion-des-Motors
```

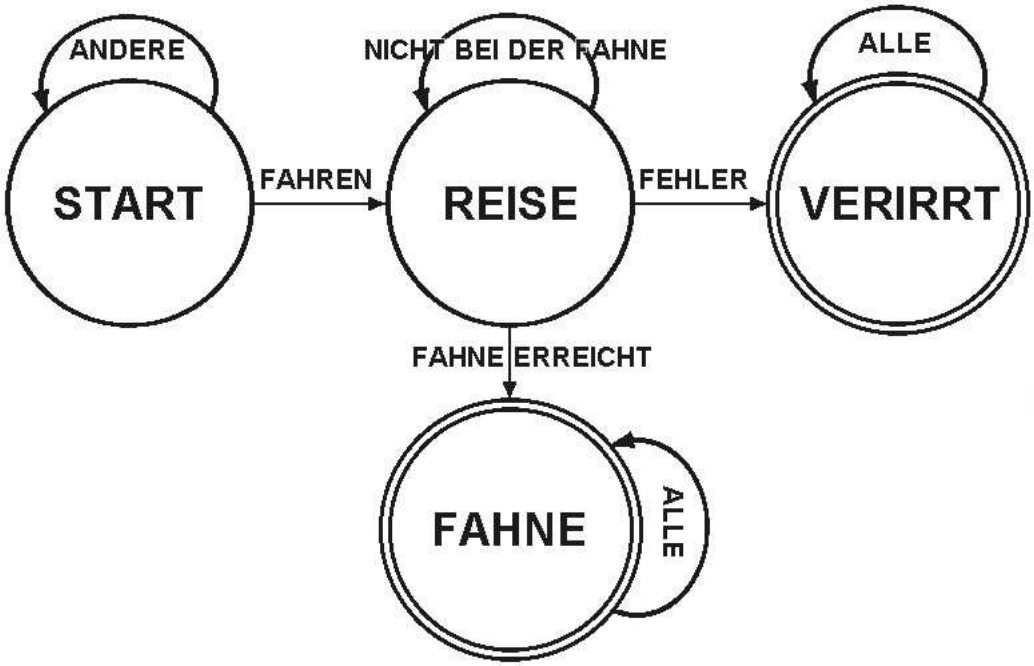

Abbildung 3.8: FSA Diagramm mit vier Zuständen. Jeder Zustand beschreibt ein Verhalten des Roboters, erzeugt durch mehrere aktive Behaviors.

3.4.3 Endliches Akzeptor Diagramm

Das Prinzip des endlichen Akzeptors (*Finit State Acceptor*, FSA) erlaubt einen anderen Blickwinkel auf die Behavior-Struktur. Ein FSA Diagramm zeigt die aktiven Behaviorgruppen (Zustände) zu jedem Zeitpunkt und die Übergänge zwischen diesen Gruppen. Ein FSA ist eine besondere Form einer Zustandsmaschine und besitzt keine Outputs. In der grafischen Darstellung sind die Zustände durch Kreise, die Endzustände durch Kreise mit Doppellinie und die Übergänge durch Pfeile dargestellt. Ein FSA wird durch ein Quadrupel (Q, δ, q_0, F) eindeutig beschrieben. Q beschreibt die Menge von erlaubten Behavior-Zuständen, δ ist eine Liste der Übergangsaktionen und q_0 gibt den Startzustand an. F ist eine Menge von Endzuständen, die ein erfolgreiches Beenden der Aufgabe bedeuten (F ist eine Teilmenge von Q). Die Liste der Übergangsaktionen δ wird zur besseren Lesbarkeit meist tabellarisch aufgelistet.

Das Beispiel mit der Fahrt zur Fahne sieht im FSA Diagramm deutlich anders aus (Abbildung 3.8). Statt der Behaviors innerhalb der Steuerung wird das resultierende Verhalten aus der Sicht eines externen Beobachters betrachtet. Es gibt vier Zustände: *Start*, *Reise*, *Verirrt* und *Fahne*. *Fahne* ist der normale Endzustand, *Verirrt* der Endzustand im Fehlerfall. Das beschreibende Quadruppel lautet:

$$M = \{\{Start, Reise, Verirrt, Fahne\}, \delta, Start, \{Verirrt, Fahne\}\} \tag{3.8}$$

Die Liste der Übergangsaktionen δ ist in Tabelle 3.1 angegeben.

δ	q	*input*	$\delta(q,input)$
	Start	fahren	Reise
	Start	andere	Start
	Reise	Fehler	Verirrt
	Reise	nicht bei der Fahne	Reise
	Reise	Fahne erreicht	Fahne
	Fahne	alle	Fahne
	Verirrt	alle	Verirrt

Tabelle 3.1: Die Liste der Übergangsaktionen δ eines FSA Diagramms.

3.5 Koordinatoren für Behaviors

Sobald in einem System mehr als ein Behavior aktiv ist, kann es zu einer Konfliktsituation zwischen den Outputs der Behaviors geben. Dieser Konflikt ist im Konzept der behavior-basierten Robotik durchaus gewünscht und wird durch Koordinatoren gelöst. Je nach Architektur bieten sich verschiedene Koordinationsmethoden an. Die verschiedenen Methoden sind aber nicht nur einzeln zu betrachten, es können auch mehrere Koordinationsverfahren gemeinsam verwendet werden.

3.5.1 Kooperative Methoden

Kooperative Koordinatoren lassen die Ergebnisse aller Behaviors in das Gesamtergebnis einfließen.

Summation Steuerungssysteme die mit der Methode des virtuellen Potentialfelds arbeiten verwenden als Koordinator für die einzelnen Behaviors eine Summation. Jedes Behavior liefert als Output einen Bewegungsvektor. Die Summe aller diese Bewegungsvektoren ergibt die gewünschte Bewegung des Roboters (Abbildung 3.9a.).

Mittelungsverfahren Koordinatoren können auch andere Mittelungsverfahren wie Mittelwert oder Median einsetzten, um die Ergebnisse aller Behaviors im Output der Steuerung zu beachten.

3.5.2 Konkurrenzmethoden

Diese Koordinatoren lassen die Behaviors miteinander im Wettbewerb konkurrieren. Hier geht es darum ein Behavior zu ermitteln, dessen Ergebnis als Systemoutput verwendet wird. Die Ergebnisse aller anderen Behavior werden ignoriert.

Wahlverfahren Die aktiven Behaviors bewerten die Ergebnisse aller anderen Behaviors. Jedes Behavior hat dazu eine vorgegebene Anzahl von Wahlstimmen, die auf die anderen Behaviors verteilt werden können. Das Behavior mit den meisten Stimmen gewinnt und sein Output wird ausgeführt (Abbildung 3.9b.).

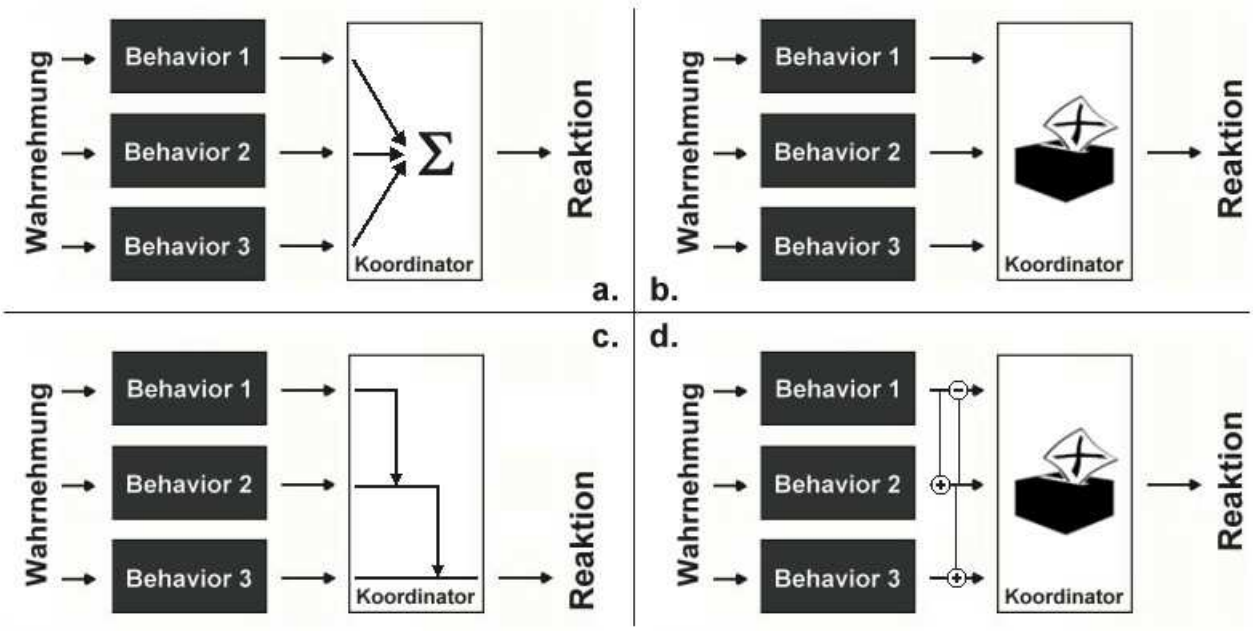

Abbildung 3.9: Beispiele für die Koordination von Behaviors: a) Summation, b) Wahlverfahren, c) Prioritäten, d) Hemmung und Verstärkung + Wahlverfahren.

Prioritäten Jedem Behavior wird ein Priorität zugeordnet. Behaviors mit höhere Priorität werden bei der Koordination Behaviors mit niedriger Priorität vorgezogen. Es gibt Systeme mit fixer Priorität wie z.B. Schichtarchitekturen bei denen jede Schicht einer Priorität entspricht. Prioritäten können aber auch im Betrieb dynamisch vergeben werden, wodurch statt einer Schichtarchitektur ein dynamisches Netzwerk entsteht (Abbildung 3.9c.).

Hemmung und Verstärkung Behaviors erhalten die Möglichkeit andere Behaviors zu hemmen oder zu verstärken. Ziel der Beeinflussung ist es, das Ergebnis eines Wahlverfahrens im Koordinator zu beeinflussen. Die Architektur bestimmt welche Behaviors von einem bestimmten Behavior beeinflusst werden können. Die Behaviors können beispielsweise gruppiert sein, damit eine Beeinflussung nur innerhalb der Gruppe erlaubt ist (Abbildung 3.9d.).

3.6 Kodierung eines Behaviors

Die konkrete Implementierung eines Behavior ist dem Systemdesigner freigestellt. Es gibt mehrere gängige Methoden, die zwei bekanntesten sind die Methode der virtuellen Potentialfelder und die Verwendung einer Fuzzy Control.

3.6.1 Die Methode der virtuellen Potentialfeld

Behaviors für Steuerungssysteme basieren häufig auf der Verwendung von virtuellen Potentialfeldern. Ähnlich wie eine Kugel entsprechend der Topologie ihrer Umgebung durch einen Raum rollt, soll sich ein Roboter entsprechend einer Topologie bewegen. Diese Topologie ist aber nicht physikalisch vorgegeben, sondern wird anhand der realen Umgebung künstlich erstellt. Diese

künstliche Abbildung ähnelt der Abbildung eines Potentialfelds, daher auch der Name der Methode. Wände und Hindernisse stellen dabei abstoßende Elemente im virtuellen Potentialfeld dar, Zielpunkte werden durch anziehende Elemente abgebildet. Dazu werden vom Systemdesigner mehrere Behaviors definiert, für jedes beschreibbare Verhalten eines. Jedes dieser Behaviors repräsentiert ein virtuelles Potentialfeld. Die Summe aller virtuellen Potentialfelder ergibt schließlich einen resultierenden Vektor, der die Bewegungsrichtung und Geschwindigkeit des Roboters ergibt.

Ein einzelnes Behavior ermittelt aus den Sensordaten einen resultierenden Bewegungsvektor. Die Richtung des Vektors hängt davon ab, ob das Behavior ein anziehendes oder abstoßendes Verhalten abbildet. Die Länge des Vektors, also die Stärke der Reaktion, hängt von einer Abbildungsfunktion ab. Die Potentialfeld-Methode unterscheidet dabei drei Bereiche: sehr weit vom Objekt, sehr nah beim Objekt und zwischen den beiden Extremen, wobei die konkreten Grenzen eine frei gewählte Designentscheidung sind. Ein `Hindernis-ausweichen` Behavior (Abbildung 3.10b) wird sehr weit entfernt von einem Hindernis noch keine Reaktion zeigen. Sehr nah vor dem Hindernis muss es einen dominanten abstoßenden Vektor ergeben. Zwischen den beiden Extremen gibt es ein graduell abgestuftes Verhalten. Ähnlich das Verhalten bei einem `Ziel-anfahren` Behavior (Abbildung 3.10a): In weiter Entfernung gibt es eine konstante Anziehung zum Ziel, kommt der Roboter näher, wird die Anziehung graduell schwächer. Eine kleiner Bereich um den Zielpunkt wird als „im Ziel“ definiert d.h. der resultierende Vektor wird in dem Bereich Null sein. Eine mathematische Beschreibung der Potentialfeld Methode bieten [Good] und [BeMc03].

Die Methode der virtuellen Potentialfelder ist sehr schnell und effizient. Die virtuellen Vektoren werden nur für den Mittelpunkt des Roboters berechnet und nicht für den ganzen betrachteten Raum. In dem Beispiel mit den drei Hindernissen zwischen zwei Gangwänden (Abbildung 3.10c) werden praktisch nur sechs Vektoren addiert, um den resultierenden Bewegungsvektor zu erhalten. Einer Vektor ist zum Ziel gerichtet, zwei Vektoren weisen weg von den Wänden und drei Vektoren weisen weg von den Hindernissen. Die erzielten Fahrbewegungen werden aber nicht immer so erfolgreich verlaufen wie in diesem Beispiel. Die Methode der virtuellen Potentialfelder hat Schwachstellen, die beachtet werden müssen:

- Lokale Minima: Orte im Raum, an denen die Summe der virtuellen Potentialfelder Null ergibt, stellen eine Falle für den Roboter dar. An einem solchen Ort hemmt der Bewegungsvektor mit Länge Null jede weitere Bewegung. Das klassische Beispiel dafür sind U-förmige Sackgassen (Abbildung 3.10d).

- Zu kleine Bewegungsvektoren: Fahrten durch Gebiete mit kleinen Bewegungsvektoren resultieren in sehr langsamer Bewegung oder gar mit Stillstand. Der Antrieb und die Räder müssen einen Widerstand überwinden bevor sie sich in Bewegung setzen, sehr langsame Geschwindigkeiten sind daher nicht möglich.

- Fahrt ohne Weitsicht: Mit dieser rein reaktiven Methode verhält sich der Roboter sehr „kurzsichtig“. Da immer nur die lokal optimale Richtung gewählt wird, werden kurzfristig nicht-optimale, aber längerfristig erfolgreichere Wege automatisch ausgeschlossen. Ein Beispiel dafür sind breite Hindernisse. Anstatt das Hindernis weiträumig zu umfahren, wird sich der Roboter scheinbar an dem Hindernis entlangtasten.

Eine Reihe von Lösungen versucht die Schwachstellen zu kompensieren. Ein zufälliges Potentialfeld in der Summierung kann genug Dynamik ins System bringen, um Probleme mit lokalen

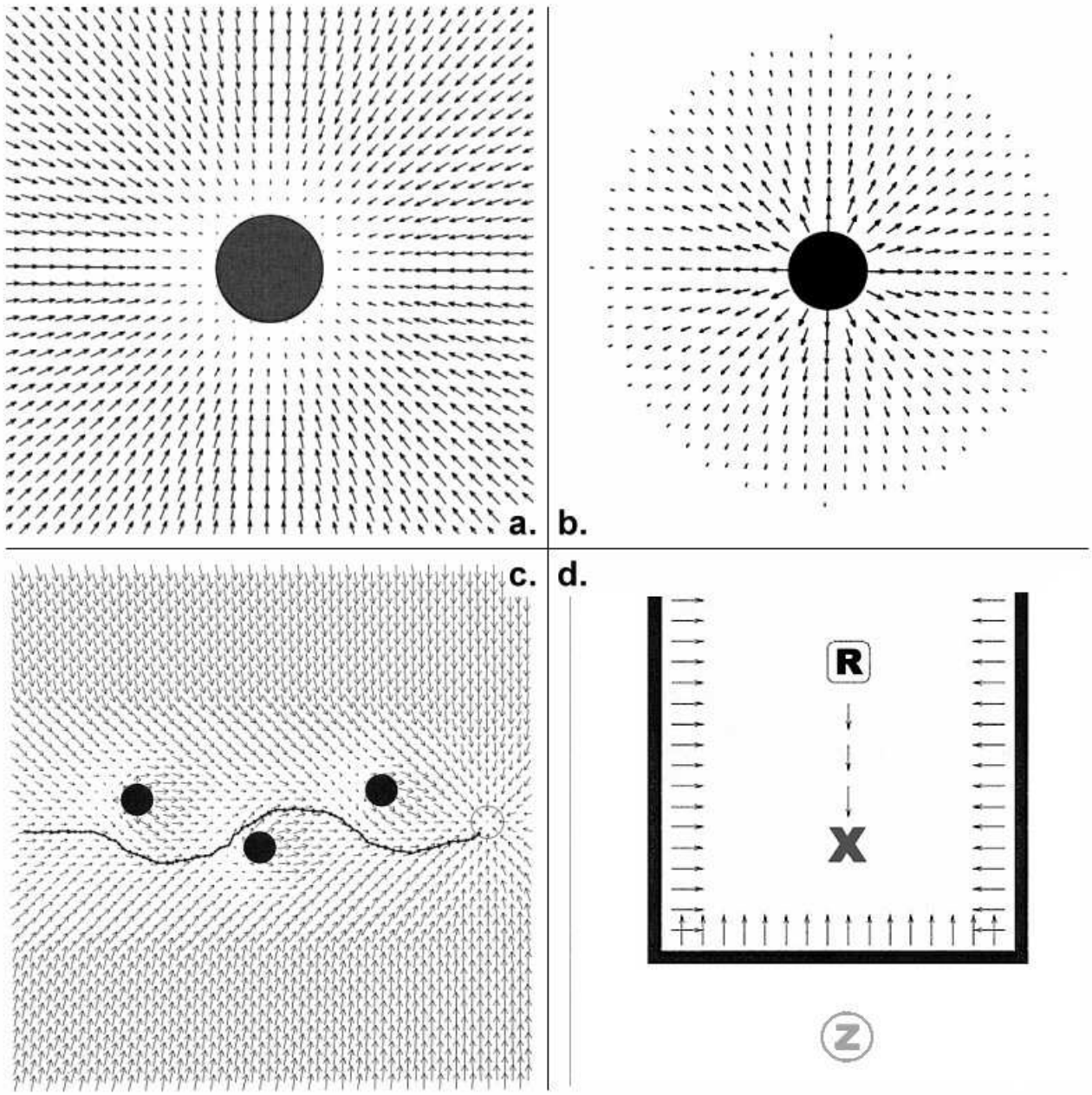

Abbildung 3.10: Darstellungen von virtuellen Potentialfeldern. a) Ein Ziel-anfahren Behavior; b) Ein Hindernis-ausweichen Behavior; c) Das summierte Potentialfeld von drei Hindernissen und einem Ziel zwischen zwei Wänden. Die Linie im rechten Bild entspricht dem Fahrweg eines von links kommenden Roboters; d) Ein Roboter fährt in eine U-förmige Sackgasse. Wenn er den Ort X erreicht, wird die Summe aller virtuellen Potentialfelder Null ergeben.

Abbildung 3.11: Das Schema einer Fuzzy Control.

Minima und zu kleinen Bewegungsvektoren zu lösen. Ein Behavior `Vermeide-die-letzten-Schritte` mit beschränktem zeitlichen Vorwissen über den zuletzt gefahrenen Weg kann den Roboter aus Sackgassen retten. Trotz ihrer Schwächen kann die Methode der virtuellen Potentialfelder erfolgreich in der reaktiven Schicht einer hybriden Steuerung eingesetzt werden.

3.6.2 Behaviors mit einer Fuzzy Control

Die „unscharfe“ Mathematik der Fuzzy Logic eignet sich sehr gut, um Behaviors mit einem natürlichen Verhalten zu kreieren. Im Unterschied zur klassischen Logik, die nur eindeutige Zustände erlaubt, arbeitet die Fuzzy Logic mit unscharfen Fuzzy Variablen, die mehr oder weniger stark einen Zustand abbilden können. Eine Fuzzy Variable wird durch eine Zahl im Bereich $[0.0, 1.0]$ dargestellt. Eine Steuerung, die mit Fuzzy Logic arbeitet, nennt man Fuzzy Control.

Eine Fuzzy Control arbeitet in drei Schritten: Fuzzyfizierung, Inferenzschritt und Defuzzyfizierung. Die gesamte Fuzzy Control kann in einem Behavior abgebildet werden (Abbildung 3.11), der Eingangsreiz wird fuzzyfiziert, läuft durch die Fuzzy Control und wird nach der Defuzzyfizierung als Reaktion weitergeleitet. Das besondere einer Fuzzy Control ist, dass sie selbst bei wenigen Fuzzy Variablen eine kontinuierliche Abbildung eines Reizes in den Reaktionsbereich erlaubt. Ein Beispiel für eine Fuzzy Control ist in Abbildung 3.12 gegeben.

A) Fuzzyfizierung Die Messwerte werden in Fuzzy Variablen abgebildet. Dazu werden die Messwerte mittels einer geeigneten Zugehörigkeitsfunktion auf die Fuzzy Variablen transformiert. Die Form der Zugehörigkeitsfunktionen kann frei gewählt werden und ist reine Designentscheidung. Mittels der Zugehörigkeitsfunktion kann nicht nur die Abbildung beeinflusst werden, sondern auch der Wertebereich der Eingangsvariablen beschnitten werden.

B) Inferenzschritt In diesem Schritt werden die Fuzzy Eingangsvariablen mit Hilfe von linguistischen wenn-dann Regeln auf Fuzzy Ausgangsvariablen abgebildet. Die linguistischen Regeln sind dabei den Regeln eines regelbasierten Systems ähnlich.

C) Defuzzyfizierung Im letzten Schritt werden die Fuzzy Werte wieder in einen realen Wert transformiert. Es gibt verschiedene Methoden dies zu tun, eine davon ist die Schwerpunktbildung der Fuzzy Ausgangsvariablen. Die Abbildung des Schwerpunktes liefert schließlich den scharfen Endwert.

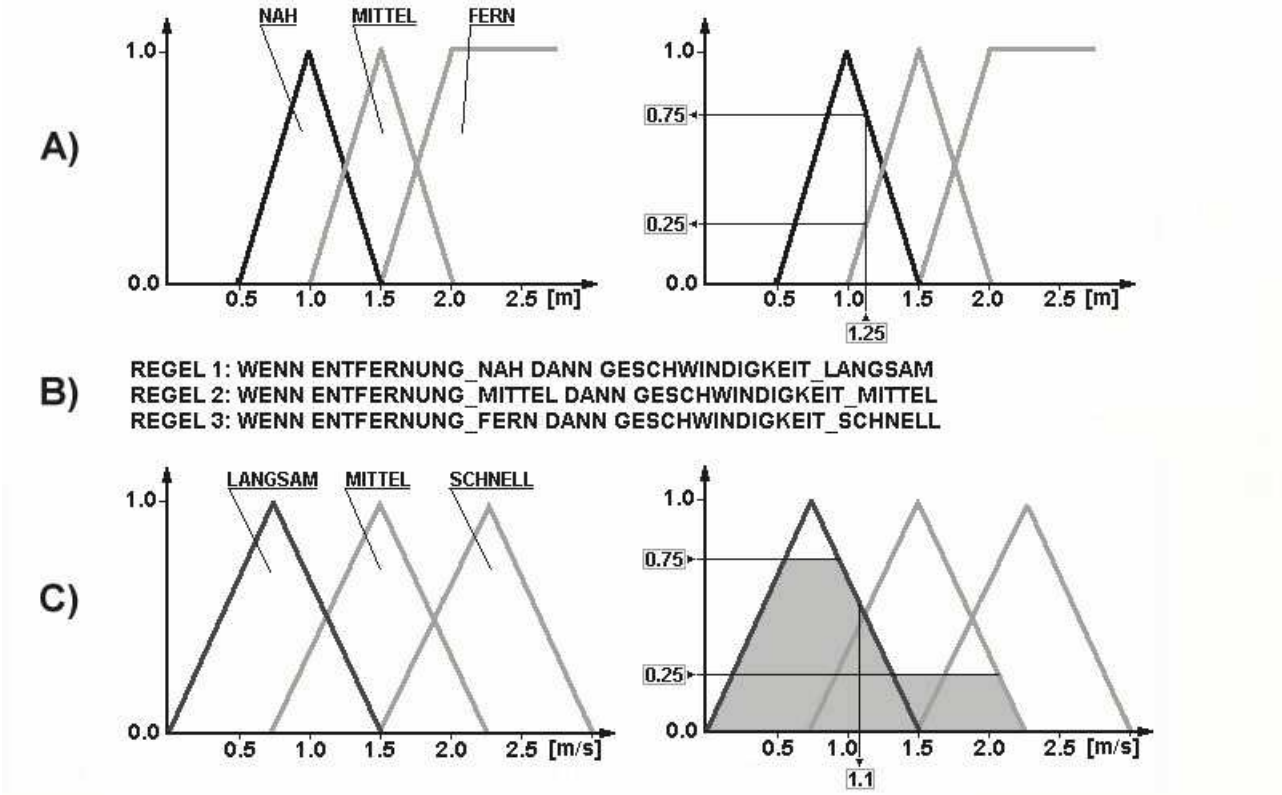

Abbildung 3.12: Beispiel für eine Fuzzy Control: Die gemessene Entfernung soll auf eine Fahrgeschwindigkeit abgebildet werden.
A) Bei der Fuzzyfizierung wird die gemessene Entfernung 1,25 m auf die drei Fuzzy Variablen nah, mittel und fern mittels der entsprechenden Zugehörigkeitsfunktionen transformiert. Die gemesse Entfernung entspricht 0,75 nah, 0,25 mittel und 0,0 fern.
B) Mit den linguistischen Regeln wird 0,75 nah auf 0,75 langsam und 0,25 mittel auf 0,25 mittel abgebildet.
C) Die drei Fuzzy Ausgangsvariablen werden mit den Zugehörigkeitsfunktionen für den Ausgang dargestellt. Der Schwerpunkt der durch 0,75 langsam und 0,25 mittel beschriebenen Fläche liefert für die Defuzzyfizierung den scharfen Endwert 1,1 m/s.

3.7 Zusammenfassung

Die Idee hinter Behaviors besteht darin, ein funktionierendes Konzept aus der Biologie in die Robotik zu übertragen. Das Verhalten von Behaviors lässt sich mathematisch beschreiben und eine Reihe von Regeln beschränkt die Randbedingungen dieses Verhaltens. Trotzdem stellt die Theorie der Behaviors nur ein Vorlage für die Steuerung eines Roboters dar. Die universell richtige Implementierung eines Behaviors für alle Anwendungen gibt es nicht. Systemdesigner haben vielmehr die Möglichkeit, das Konzept der behavior-basierten Robotik für die Aufgabenstellung eines konkreten Roboters zu verwenden. Die Implementierung der Behaviors selbst und die Kodierung der Verknüpfung zwischen den Behaviors bieten dem Entwickler dabei zahlreiche Freiräume.

Kapitel 4

Sensoren in der Robotik

Ein Roboter ohne Sensoren ist nicht mehr als ein Automat, der vordefinierte Aktionen ausführen kann. Erst die Verwendung von Sensoren erlaubt es Robotern ihre Umwelt wahrzunehmen und auf dynamische Situationen zu reagieren. Die Quantität und vor allem Qualität der Sensoren bestimmt in großen Maße die Autonomie, mit der ein Roboter seine Tätigkeit verrichten kann. Fast alle Arten von Sensoren können in der Robotik Verwendung finden. Einschränkungen stellen hauptsächlich Stromverbrauch, Größe und Gewicht dar, schließlich muss der Sensor vom Roboter transportiert werden können. Ein weiteres Argument für einen kommerziellen Einsatz stellt der Preis des Sensors dar.

In diesem Kapitel werden zuerst kurz die verschiedenen gängigen Sensorentypen vorgestellt, die in der mobilen Robotik zum Einsatz kommen. Es werden dabei die Charakteristika des jeweiligen Sensors betrachtet, im besonderen der Arbeitsbereich und die Einsatzmöglichkeiten. Danach folgt eine ausführlichere Betrachtung von Kameras mit Bildverarbeitung als visuelle Sensoren für die Robotik. Das breite Einsatzspektrum eines visuellen Sensors bietet völlig andere Möglichkeiten als die restlichen gängigen Sensorentypen in der Robotik.

4.1 Gängige Sensoren und ihre Verwendung in der mobilen Robotik

Es gibt mehrere Aufgabengebiete in der mobilen Robotik bei denen Sensoren zum Einsatz kommen.

Die eigene Ortsbestimmung des Roboters: Hier bieten sich Odometer (Winkelenkoder) und Sensoren für eine Koppelnavigation (Geschwindigkeits- und Beschleunigungsmesser) an.

Entfernungmessung zu Objekten in der Umgebung: Verschiedene Sensoren die entweder mit Schall (Sonarsensoren) oder Licht (Infrarotsensoren, Laserscanner) arbeiten werden hierfür eingesetzt. Aus den Ergebnissen der laufenden Entfernungsmessung kann auch eine Karte der Umgebung erstellt werden.

Berührungserkennung: Um zu erkennen, ob der Roboter Objekte in der Umgebung berührt, werden meist taktile Sensoren eingesetzt.

Objekterkennung: Die meisten Arten von Sensoren sind für eine Objekterkennung und -klassifizierung nur schlecht geeignet. Die besten Ergebnisse verspricht hier die Verwendung von Kameras als visuelle Sensoren.

Ein wichtiges Unterscheidungskriterium von Sensoren ist ihr Verhalten gegenüber der Umwelt. Man unterscheidet hier zwischen aktiven und passiven Sensoren. Aktive Sensoren senden zur Messung ein Signal aus (z.B. Schall oder Licht) oder sie agieren aktiv mit der Umwelt (z.B. durch Berührung). Aus der Reflexion des Signals oder durch die in der Umwelt erzeugte Veränderungen können aktive Sensoren dann Informationen über die Umwelt gewinnen. Passive Sensoren nehmen die Umwelt wahr, ohne sie aktiv zu verändern. Bei der Verwendung von mehreren Sensoren ist besonders darauf zu achten, dass aktive Sensoren die Messung anderer Sensoren nicht negativ beeinflussen. So kann etwa ein aktiver Infrarotsensor eine passive Kamera stören.

4.1.1 Taktile Sensoren

Taktile Sensoren sind aktive Sensoren, die mit mechanischer Kraft arbeiten. Findet zwischen dem Sensor und einem Gegenstand eine Berührung statt, so wird dies erkannt. Ein taktiler Sensor ist im einfachsten Fall ein mechanischer Schalter, der nur eine binäre Information liefern kann, also „Berührung“ oder „keine Berührung“. In der mobilen Robotik werden diese günstigen Sensoren verwendet, um Kollisionen zu detektieren, z.B. für den Fall, dass die berührungslosen Abstandssensoren versagen. Es gibt zwei gängige Ausführungen: Bumper und Whiskers.

- Bumper nennt man direkt am Roboter befestigte Schalter mit beweglichen Kontaktplatten. Fährt der Roboter gegen ein Hindernis, so drückt die entsprechende Kontaktplatte gegen den Schalter und die Berührung wird registriert.
- Whiskers sind eine Nachahmung der Fühler von Insekten bzw. der Schnurrhaare einer Katze. Einfache Bauformen verwenden einen dünnen, flexiblen Metalldraht, der durch einen am Robotergehäuse montierten Metallring geführt wird. Durch einen äußerlichen Krafteinfluss biegt sich der Draht, berührt den Metallring und schließt einen Stromkreis.

Die aktuelle Forschung geht weiter und versucht Whiskers zu entwickeln, die spezifischere Daten über die berührte Oberfläche und die Art der Berührung liefern können z.B. Rauheit der Oberfläche, Ausrichtung der Struktur, Geschwindigkeit und Dauer der Berührung [ABRG06]. Es ist davon auszugehen, dass künftig für die stärkere Interaktion zwischen Robotern und Menschen (Verletzungsgefahr) und Gegenständen (z.B. zerbrechliches Geschirr) entsprechend feinfühlige taktile Sensoren benötigt werden.

4.1.2 Odometrie

Mobile Roboter benötigen Sensoren, um ihre eigene Pose (Position und Orientierung im Raum), sowie die eigene Geschwindigkeit bei Bewegungen messen zu können. Die Odometrie ist die Positionsbestimmung eines radgetriebenen Fahrzeugs durch die Beobachtung seiner Räder. Dazu werden Winkelenkoder an den Achsen der Antriebsräder angebracht. Praktisch alle radgetriebenen Roboter sind mit diesem Sensortyp ausgestattet. Oft sind die Winkelenkoder schon in den Antrieben eingebaut. Winkelenkoder sind optische Zähler, die mit markierten Scheiben

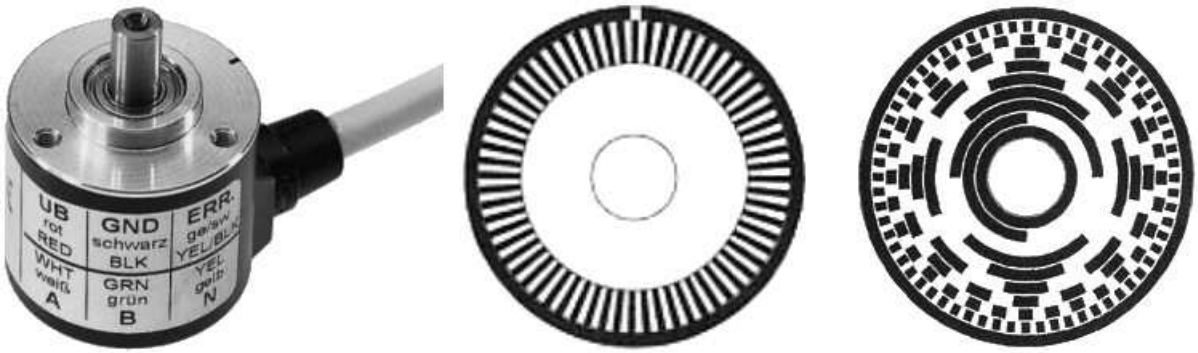

Abbildung 4.1: Links ein Beispiel für einen Winkelenkoder. Mitte und rechts zwei Beispiel für die in Winkelenkodern verwendeten Kodierscheiben. In der Mitte die Scheibe eines inkrementierenden Winkelenkoders, rechts eine absolut kodierte Scheibe mit 8-bit Gray-Code.

arbeiten. Diese Scheibe dreht sich mit der Achse mit und das Muster auf der Scheibe erzeugt über eine Lichtschranke ein Ausgangssignal. Je nach Ausführung der Scheibe spricht man von inkrementellen oder absoluten Winkelenkodern (Abbildung 4.1 zeigt Beispiele für die entsprechenden Scheiben). Trotz relativ hoher Auflösung von mehreren 10.000 Impulsen pro Achsumdrehung liefern Winkelenkoder nur auf kurzen Strecken genaue Messwerte. Bei längerer Fahrt summieren sich die Fehler bedingt durch falsche Radgeometrie (fehlerhafte Messung der Raddurchmesser, Verschleiß, verformte Räder durch Last), Fahrgestellgeometrie (Achsstellung, Achsspiel) und Bodenbeschaffenheit (Unebenheiten, Schlupf). Winkelenkoder sind günstige Sensoren. Aufgrund ihrer Fehlereigenschaften werden sie aber selten als einziges Verfahren für eine Ortsbestimmung eingesetzt.

4.1.3 Koppelnavigation

Unter dem Begriff Koppelnavigation versteht man die laufende Ortsbestimmung eines Fahrzeugs durch Messung von Richtung, Geschwindigkeit und Zeit. Der Begriff stammt ursprünglich aus der Schifffahrt, das Verfahren ist auch unter der Bezeichnung dead reckoning bekannt. Geschwindigkeits- und Beschleunigungsmesser sowie elektrische Kompasse und Gyroskope werden hierfür verwendet. Ein Roboter kann mit diesen Sensoren seinen eigenen Ort bestimmen, die Messung liefert aber einen laufenden Fehler, der sich wie bei der Odometrie aufsummiert.

4.1.4 GPS und andere Systeme zur Positionsbestimmung

Das Global Positioning System (GPS), Differential-GPS (mehrere GPS-Empfänger, die sich abgleichen), Galileo (Europäische Satellitennavigation ab 2011) und Positionsbestimmung durch Sendezellen der Mobiltelelefonie sind Beispiele für Dienste, die durch einen geeigneten Empfänger eine lokale Ortsbestimmung ermöglichen. Die Genauigkeit der verschiedenen Dienste ist unterschiedlich, mit Ausnahme von Differential-GPS und Galileo jedoch im Bereich von mehreren Metern. Die Satellitennavigation benötigt außerdem eine freie Sicht zu mehrere Satelliten. Diese Systeme sind nur für Outdoor Roboterfahrzeuge interessant.

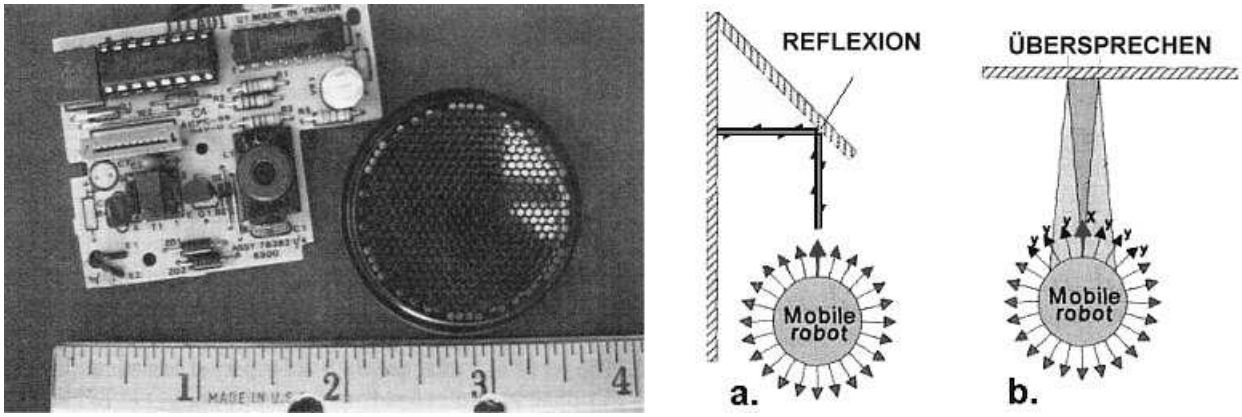

Abbildung 4.2: Links ein Beispiel für einen Sonarsensor aus [BEF96]. Das runde Gehäuse beherbergt eine Platine mit Ultraschallsender und -empfänger. Rechts zwei Beispiele für Fehlerquellen. In Beispiel a wird das Signal von einer glatten Oberfläche ähnlich einem Spiegel reflektiert. Der Signalweg verlängert sich und der Roboter „sieht“ das Hindernis zu weit vor sich. In Beispiel b wird das Signal eines Sensors von anderen Sensoren wahrgenommen, es kommt zu einem Übersprechen.

4.1.5 Ultraschallsensoren

Ultraschallsensoren ermöglichen eine aktive Entfernungsmessung mit dem Prinzip der Signallaufzeitmessung. Es wird ein kurzer Ultraschallimpuls ausgesandt und die Zeit bis zur rücklaufenden Schallreflexion gemessen. Unter Einbeziehung der Schallgeschwindigkeit lässt sich daraus die Entfernung berechnen. Durch die „langsame“ Schallgeschwindigkeit ist nur eine Zeitmessung im Millisekundenbereich notwendig und der Sensor daher günstig. Abbildung 4.2 zeigt ein Beispiel für einen Ultraschallsensor. Durch die kegelförmige Ausbreitung des ausgesendeten Schallsignals (15 - 25° bei herkömmlichen Sensoren) verringert sich die Auflösung des Sensors mit der Entfernung beträchtlich. Die Entfernungsmessung ist außerdem mit einem konstanten relativen Fehler behaftet ($< 5\%$) . Der einsetzbare Arbeitsbereich reicht von 20 cm bis 5 m.

In der mobilen Robotik werden Ultraschallsensoren häufig eingesetzt, da sie günstig sind und bereits sehr viel praktische Erfahrung mit dem Sensor vorhanden ist. Am Roboter wird meist ein Array von Sonarsensoren verwendet, um eine Halbebene oder Ebene in 15 - 20 Schritten abzutasten. Der Messzyklus eines Sensors dauert etwa 40 ms. Beim zyklischen, sequenziellen Auslesen von acht Sensoren sinkt die Datenfrequenz damit aber bereits auf 3 Hz. Zahlreiche Fehlerquellen bei der Schallausbreitung beeinflussen die Messung: Schalldämpfung oder -absorption, Mehrfachreflexion (Abbildung 4.2), Übersprechen mit anderen Sensoren (Abbildung 4.2) und veränderte Schallgeschwindigkeit (Temperatur, Feuchtigkeit) können das Messergebnis verfälschen. Neben der Entfernungsmessung können Ultraschallsensoren gemeinsam mit einer Kartierung der Umgebung auch zur Unterstützung der eigenen Positionsbestimmung verwendet werden. Gemeinsam mit einer Karte ist auch eine sehr grobe Objekterkennung von großen, unbewegten Objekten möglich.

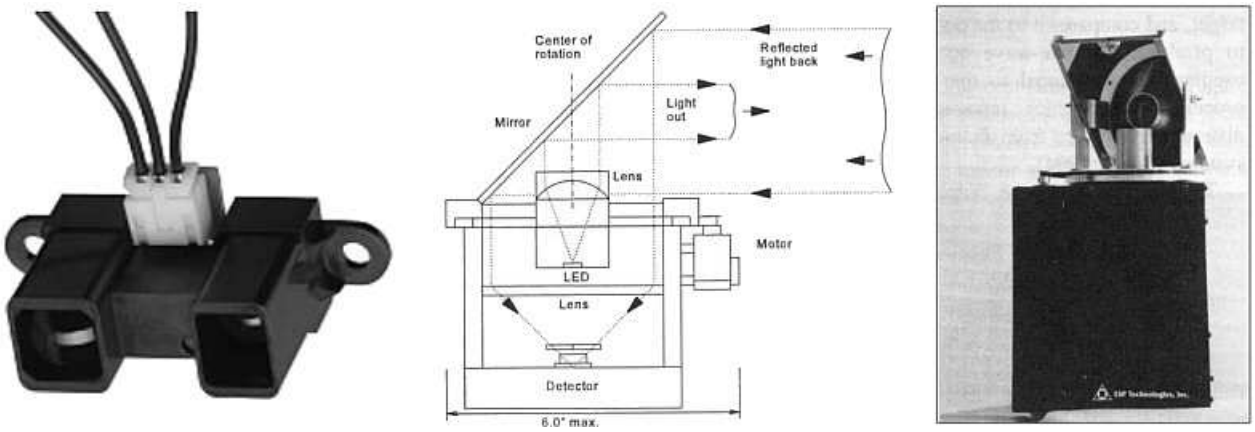

Abbildung 4.3: Links ein einfacher Infrarotsensor von Sharp. Das Gehäuse enthält Sende- und Empfangsteil. Die Kabel dienen zur Spannungsversorgung und als analoger Ausgang. Rechts ein aufwendigeres Modell mit drehbarem Spiegel aus [BEF96]. Die Skizze in der Mitte zeigt die Infrarot LED, deren Signal von einer Linse fokusiert und vom Spiegel zum Ziel gelenkt wird. Das reflektierte Signal wird vom Spiegel zurück ins Gehäuse gelenkt, wo es von einem Detektor wahrgenommen wird.

4.1.6 Infrarotsensoren

Einfache Infrarotsensoren, die ähnlich Sonarsensoren aktiv Hindernisse vor sich erkennen können, sind bereits im niedrigen Preissegment zu finden. Durch die Verwendung eines gebündelten Infrarotstrahls als Messsignal haben Infrarotsensoren gegenüber Ultraschallsensoren den Vorteil einer schnellen und genaueren Messung, aber auch den Nachteil einer geringeren Reichweite (Arbeitsbereich 20 cm - 2 m). Ein praktisches Anwendungsgebiet sind damit zeitkritische Entfernungsmessung z.B. ob der fahrenden Roboter auf eine abwärts führende Treppe zufährt.

Diese Sensoren arbeiten mittels der Methode der Triangulation. Das bedeutet, die Senderseite produziert den Infrarotstrahl, dessen Abbildung auf dem Hindernis von einem PSD-Element[1] in der Empfängerseite wahrgenommen wird. Das Messergebnis wird als analoger Spannungswert auf den Ausgang des Sensors gelegt. Als Beispiel ist ein solcher Infrarotsensor von Sharp in Abbildung 4.3 zu sehen. Ähnlich wie Sonarsensoren können auch Infrarotsensoren bei bekannter Umgebung (Karte) eine Selbstlokalisation und grobe Objekterkennung durchführen. Durch die geringe Reichweite der Sensoren sind die Möglichkeiten aber stark eingeschränkt.

Es gibt auch aufwendigere Modelle, die mit einem drehbaren Spiegel eine ganze Ebene überwachen können. Diese Infrarot Entfernungsmesser berechnen aus dem reflektierten Licht mittels der Phasenverschiebungsmethode die Entfernung zum Hindernis. Dabei wird die Phasenverschiebung der Hüllkurve zwischen dem ausgesandten modulierten Trägersignal und dem empfangenen Signal gemessen und daraus die Entfernung berechnet. Eine schematische Abbildung eines solchen Sensors ist in Abbildung 4.3 zu sehen. Aufgrund der Mechanik und Stossempfindlichkeit des drehbaren Spiegels sind solche Sensoren für die mobile Robotik aber wenig geeignet.

[1] Positionssensitive Detektoren (PSD) werden bei Sensoren verwendet, die mit dem Trigonometrischen Verfahren Abstände messen. Ein Beispiel für ein PSD-Element wäre eine Lateraleffekt-Photodiode, bei der die Differenz der abfließenden Ströme direkt einer Position am Element entspricht. Damit können kostengünstig hochgenaue Entfernungsmessungen durchgeführt werden.

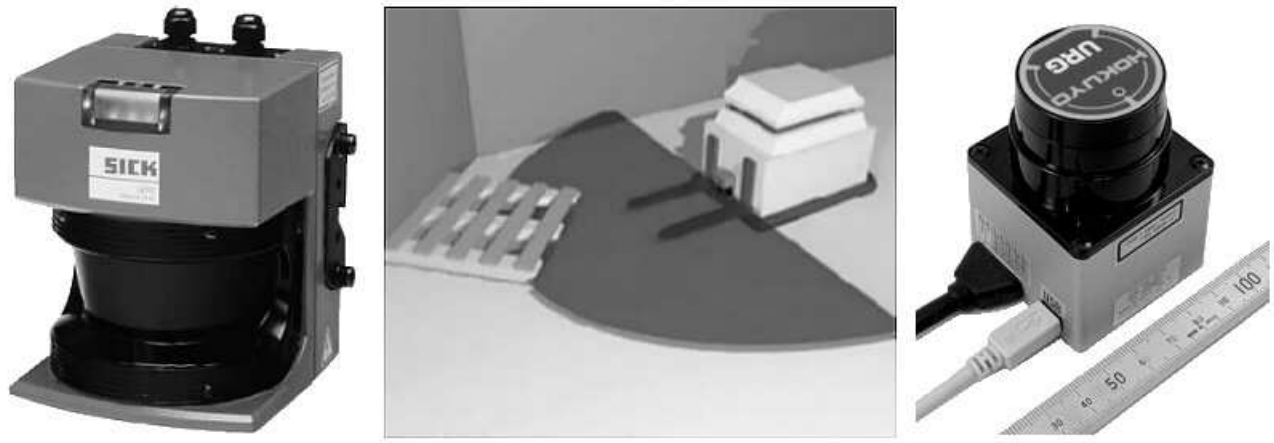

Abbildung 4.4: Links der LMS 200, ein weit verbreiteter Laser Entfernungmesser der Firma SICK, auch als „blue box“ bekannt. Das Modell arbeitet mit Pulslaufzeitmessung auf eine Entfernung bis zu 30 m mit einer Genauigkeit von +/- 15 mm. Der LMS 200 ist nicht explizit für mobile Roboter konzipiert und wiegt 4,5 kg, hat fast ein Liter Volumen und verbraucht bis zu 17,5 W Leistung. Das mittlere Bild zeigt schematisch den gescannten 180° Bereich vor einem Fahrzeug, das mit einem LMS 200 ausgestattet ist.
Rechts ein speziell für mobile Roboter entwickelter Scanning Laser Range Finder der Firma Hokuyo. Der 160 g schwere Sensor scannt einen Bereich von 240° und verbraucht dabei nur 2,5 W. Die Reichweite ist aber auf vier Meter begrenzt (Messgenauigkeit +/- 10 mm).
Die Abbildungen wurden von [Vid06] entnommen.

4.1.7 Laserscanner

Laserscanner führen Entfernungsmessungen mit einem Laserstrahl durch. Es werden unterschiedliche Messmethoden eingesetzt, aber alle haben den Vorteil einer sehr schnellen und genauen optischen Messung. Entsprechend dem gewünschten Messbereich arbeiten diese Geräte mit einer Pulslaufzeitmessung (bis zu 1000 m), der Messung der Phasenverschiebung (bis zu 100 m) oder einer optischen Triangulation (bis zu 10 m). Bei teuren Geräten sind Messgenauigkeiten bis in den Submillimeterbereich möglich. Durch die hohe Arbeitsgeschwindigkeit können Laser Entfernungsmesser ganze Bereiche scannen. In der mobilen Robotik werden oft Modelle verwendet, die eine Halbebene (den Bereich vor dem Roboter) in hoher Auflösung (z.B. 0,5° Schritten) abtasten. Neben der Entfernungmessung ist damit eine genaue Kartierung der Umgebung und basierend darauf eine Selbstlokalisation möglich. Die feine Abtastung erlaubt auch eine Interpretation der Daten für eine Objekterkennung. Im Forschungsbereich sind alle Arten von Laser Entfernungmessern erfolgreich im Einsatz. Bei einer Verwendung in einem Massenprodukt wie einem Serviceroboter müssen aber Einschränkungen beachtet werden. Zum einen müssen Größe und Gewicht des Gerätes zum Roboter passen. Die momentan verwendeten Modelle (Beispiel siehe Abbildung 4.4) mit mehreren Kilogramm Gewicht werden aber bald von geeigneteren kleineren Geräten abgelöst werden. Ein weiterer Vorteil der Miniaturisierung ist eine Reduktion des Stromverbrauchs. Ein weiteres ausschliessendes Kriterium für Laser Entfernungsmesser stellt derzeit auch oft der Preis von mehreren tausend Euro pro Stück dar.

4.2 Computer Vision als Sensor für die Robotik

Eine besondere Art von passivem Sensor stellt eine Kamera mit Bildverarbeitung dar. Statt einem Messwert je Messung, wird eine große Anzahl von Bildpunkten gleichzeitig aufgenommen. Der Vorteil liegt auf der Hand, es wird sehr viel Information auf ein Mal gewonnen. Selbst bei einer niedrigen Auflösung von 320 x 200 Pixel sind bereits 64.000 Bildpunkte je Bild verfügbar. Die große Informationsmenge ist aber auch ein Probleme, soll doch im Idealfall die Bildverarbeitung das aktuelle Bild vor der Aufnahme des nächsten Bildes komplett verarbeitet haben. Die dazu notwendige Rechenleistung hängt dabei von der Komplexität der gewünschten Bildverarbeitung ab.

In einem Sensorsystem bestehend aus Kamera und Bildverarbeitung wird die Extraktion von Information aus den Bilddaten als Computer Vision bezeichnet. Ebenfalls gebräuchlich ist die Bezeichnung Machine Vision für Computer Vision Anwendungen im Bereich der Robotik. Seltener und uneinheitlich verwendet wird der Begriff Robot Vision. Eine Verwendung ist die semantische Trennung zwischen Robot Vision für mobile Roboter und Machine Vision für Industrieroboter [Hor86].

4.2.1 Das digitale Bild

Ein digitales Bild entsteht durch die Abtastung und Quantisierung eines analogen Bildes. Ein analoges Bild kann durch eine Funktion $f(x, y)$ beschrieben werden. Jedem Punkt im Bild wird über seine Koordinaten x und y ein Maß für seine Helligkeit als Funktionswert zugeordnet. Im digitalen Bild ist dieser Helligkeitswert meist in 256 Stufen (8 Bit) quantisiert. CCD- und CMOS Sensoren erledigen die Abtastung und Quantisierung am Aufnahmechip und liefern bereits ein digitales Bild. Digitale Farbbilder enthalten für jeden Bildpunkt je nach Farbmodell mehr als nur einen Helligkeitswert. Im Falle des RGB Farbmodells wird für jeden Bildpunkt ein Farbwert für die Primärfarben Rot, Grün und Blau gespeichert. Die Mischung dieser drei Farben entspricht der Farbe des Bildpunkts im abgetasteten Bild. Ebenfalls gebräuchlich ist das YUV Farbmodell. Hier wird eine Farbe durch ein Farbdichtesignal Y und zwei Farbdifferenzsignale U und V abgebildet.

4.2.2 Die Abbildung

Das einfachste fotografische Abbildungsmodell ist eine Camera Obscura (Lochkamera). Eine solche Kamera verwendet statt einer Linse nur eine kleine Lochöffnung. Das entstehende Bild ist sehr lichtarm, aber durch die Lichtbrechung am Loch verzerrungsfrei und in allen Entfernungen scharf. Im Gegensatz dazu verwendet man Kameras mit Objektiven, bestehend aus mehrschichtigen Linsen, um viel Licht durchzulassen. Die Abbildung hinter einer Linse ist aber immer optisch verzerrt. Die mathematische Beschreibung der Abbildung einer Camera Obscura kann als erste Näherung auch für Kameras mit Objektiv benutzt werden. Bezeichnet G die Höhe des Objekts, g den Abstand zwischen Objekt und Lochscheibe, B die Höhe der Abbildung und b den Abstand zwischen Abbildung und Lochscheibe, so gilt:

$$B/G = b/g \tag{4.1}$$

Diese Gleichung ist in der geometrischen Optik als 1. Linsengleichung bekannt. Die einfachen

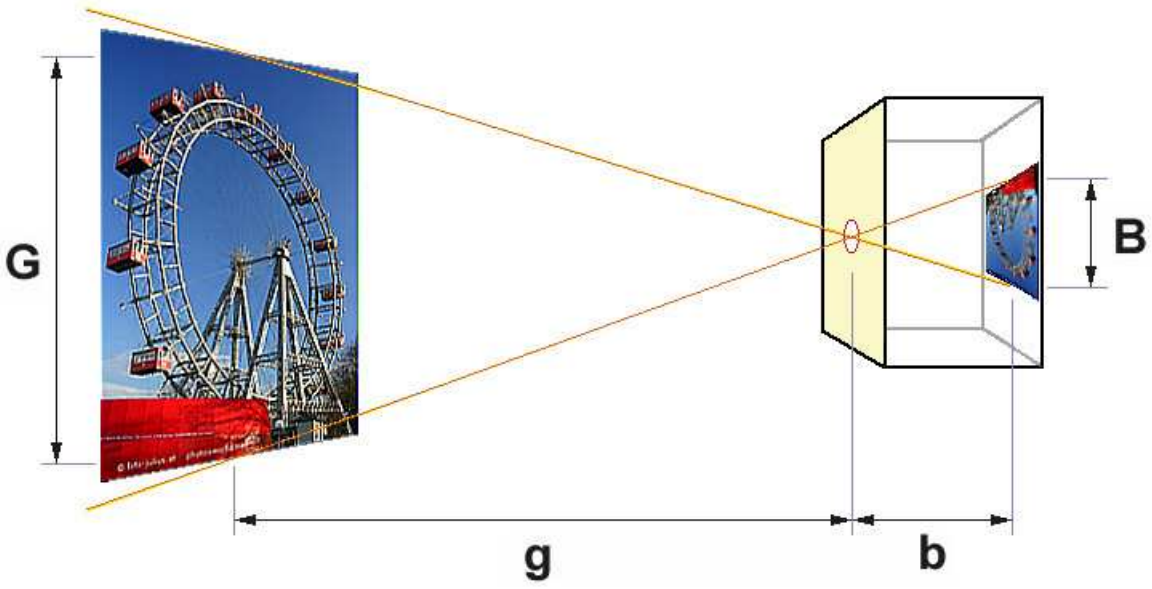

Abbildung 4.5: Schematische Abbildung einer Camera Obscure. Die verzerrungsfreie Abbildung lässt trigonometrisch einfach berechenbare Aussagen über die abgebildeten Objekte zu.

Berechnungen erlauben eine genäherte geometrische Aussage über die im Bild sichtbaren Objekte. Die Qualität dieser ersten Näherung reicht für die in der industriellen Robotik geforderte Genauigkeit allerdings nicht aus.

Eine Kamera bildet einen Raumwinkel mit rechteckigem Umriss in ein zweidimensionales Bild ab (Abbildung 4.5). Ein digitales Bild besteht aus einer Menge von Pixeln, geordnet in Zeilen und Spalten. Jedes Pixel bildet also ebenfalls einen Raumwinkel mit rechteckigem Umriss ab. Die Größe dieses Raumwinkels je Pixel ergibt sich aus dem verwendeten Objektiv und der Auflösung des Chips. Der Raumwinkel je Pixel stellt außerdem ein Maß für die minimale Größe von Objekten dar, die abgebildet werden können.

4.2.3 Die digitale Kamera

Aktuelle digitale Kameras arbeiten meist mit einem CCD(Charged Coupled Device)-Chip. Ein Vorteil dieser Technik ist, das alle Pixel gleichzeitig ausgelesen werden und ein Bild damit wirklich einer Abbildung zu einem definierten Zeitpunkt entspricht. Die Bilder von ebenfalls erhältlichen Kameras auf Basis der CMOS oder NMOS Technik können dagegen nur sequenziell als Raster ausgelesen werde. Einfache Modelle werden hauptsächlich für Spezialanwendungen z.B. Aufnahmen im Infrarotbereich verwendet. Technisch ausgereifte Ausführungen mit einem Global Shutter sind teurer CCD Kameras, bieten aber zusätzliche Features wie z.B. das Auslesen von Teilbereichen des Bildes. Die weiteren Informationen beziehen sich auf CCD-Kameras.

4.2.3.1 Videostandards

Früher arbeiteten die meisten kommerziell erhältlichen Kameras mit Videoformaten aus dem TV-Bereich, die Frequenzen von 25 bzw. 30 Hz bieten. Eine Eigenheit dieser Formate ist das Interlacing, ein früher Versuch das für das menschliche Auge sichtbare Flackern beim Bild-

wechsel zu reduzieren. Dazu wird jedes Bild aus einem Paar Halbbildern zusammengesetzt, die zeitversetzt aufgenommen werden. Kameras ohne Interlacing bzw. mit höheren Bildfrequenzen sind schon seit langem erhältlich, waren aber teuer. Mit dem CCD-Chip und den darauf basierenden günstigen USB/Firewire-Kameras konnten sich schlussendlich Bild- und Videoformate ohne Interlacing etablieren. Ein weiterer Vorteil der CCD-Kameras ist, dass man das Videobild ohne Videokarte direkt am Computer weiterverarbeiten kann. Die verwendeten Bildauflösungen haben ein 4:3 Seitenverhältnis und stammen aus dem Computerbereich: 320 x 240 Pixel, 640 x 480 Pixel u.s.w.

4.2.3.2 Bussysteme

Die hohe Datenmenge von digitalen Kameras muss über ein entsprechend breit ausgelegtes Bussystem zum Computer übertragen werden. Die weit verbreiteten und standardisierten Bussysteme USB 2.0 (Universal Serial Bus) mit 480 MBit/s [OHCIUSB] und FireWire 400 (IEEE 1394) mit 400 MBit/s bzw. FireWire800 mit 800 MBit/s [OHCI1394] bieten die notwendigen Bandbreiten für 15 und mehr Bilder pro Sekunde.

4.2.4 Bildverarbeitung in Software oder Hardware

Trotz schneller Bussysteme und schneller Rechner stellt die Bildverarbeitung immer noch einen Flaschenhals dar. Eine rein als Software laufende Bildverarbeitung kann bei komplexen Algorithmen noch mehrere Sekunden zur Verarbeitung eines Bildes benötigen. Eine Beschleunigung der Bildverarbeitung zu Echtzeit-Geschwindigkeiten (Frequenz der Bilderverarbeitung gleich der Frequenz der Kamerabilder) stellt ein wichtiges Ziel dar. Der industrielle Bereich hat dafür bereits eine praktische Lösung gefunden. Die für das jeweilige Problem passende Bildverarbeitung wird speziell optimiert und direkt als Hardware implementiert. Dazu wird sie auf ein FPGA (*Field Programmable Gate Array*) programmiert. Dieser Lösungsweg kann bei der Verwendung mit einem mobilen Roboter nicht verwendet werden. Der mobile Roboter ist beweglich, die Kameraposition und die Lichtverhältnisse - und damit die Anforderungen an die Bildverarbeitung - ändern sich dauernd. Eine teilweise Lösung des Geschwindigkeitsproblems sind Kameras mit integrierter Bildverarbeitung, die bestimmte Bildverarbeitungsoperationen bereits automatisch durchführen.

4.2.5 Die Bildverarbeitung

Das digitale Bild wird mit einer Bildverarbeitung aufbereitet, um daraus Informationen gewinnen zu können. Mit Operatoren wird der Helligkeitswert jedes Pixels vom Originalbild in ein Ausgangsbild transformiert. *Punktoperatoren* wie der Histogrammoperator oder Schwellwertoperatoren transformieren den Helligkeitswert jedes Pixels unabhängig von seiner Umgebung. *Lokale Operatoren* werden Filter genannt. Sie berücksichtigen bei der Transformation eines Pixel die Helligkeitswerte der umgebenden Pixel, beispielsweise einen 3 x 3 Pixel Bereich. Beispiele dafür sind Hochpass-, Tiefpass- und Mittelwertfilter. *Globale Operatoren* verwenden für die Transformation immer das gesamte Bild. Beispiele dafür sind der Laplace-Operator und die Fouriertransformation. Die Operatoren können mehrfach und in Kombinationen miteinander verwendet werden, um die gewünschte Bildinformation zu gewinnen. Das Ziel der Bildverarbeitung ist eine Extraktion von geometrischen Information, etwa das Ermitteln der Pose eines

abgebildeten Objekts. Wenn man diese Informationen mit Daten wie Geschwindigkeit und Pose des Roboters verbindet, dann kann die Bildinformation zur Navigation des Roboters eingesetzt werden.

4.2.6 Machine Vision in der Robotik

Die gemeinsame Geschichte von Robotik und Bildverarbeitung reicht weit zurück [Cor93]. Bereits vor zwanzig Jahren waren Kameras und integrierte Bildverarbeitungssysteme bei kommerziellen Robotern erhältlich. Während damals aber monochrome Kamerasysteme als Standard galten, erlaubt die heute verfügbare Rechnerleistung den Einsatz von Farbkameras. Der inzwischen im Vergleich zur Roboterhardware sehr geringe Preis von Kameras, insbesonders von USB-Kameras, hat den Einsatz von Computer Vision in der Robotik weiter forciert.

Die ersten Anwendungen von Bildverarbeitungen für Roboter arbeiteten hauptsächlich mit dem *look-then-move* Ansatz und waren primär für den industriellen Einsatz konzipiert. Die Steuerung dafür ist ein offener Regelkreis. Ein Bild wird aufgenommen und ausgewertet, dann wird der Roboterarm bewegt, danach wird das nächste Bild aufgenommen, u.s.w. Dieses Vorgehen war auch durch die Bildverarbeitungszeiten von bis zu einer Sekunde notwendig. Eine deutliche Verbesserung des look-then-move wurde mit der Methode des visual servoing (oder allgemeiner: visual feedback) möglich. Dabei wird die Videoinformation über die aktuelle Position für eine Feedback-Regelscheife verwendet, wodurch sich die Genauigkeit des Systems drastisch erhöht. Vorbedingung dafür ist allerdings eine entsprechend schnelle Bildverarbeitung.

Ein weiteres Verfahren ist *structure from motion*. Diese Technik versucht aus einer Sequenz von Bildern Informationen über die räumliche Gestalt eines betrachteten Objekts und die relative Bewegung zwischen dem Objekt und der Kamera zu ermitteln. Die Struktur dieses Verfahrens unterscheidet sich aber sehr von den sonst meistens bei Bildverarbeitung benutzten hierarchischen task-level Roboter Kontrollsystemen. Da industrielle Roboter meistens a priori Information über das Objekt und die Arbeitsumgebung haben, hatte structure from motion in der Industrierobotik wenig Anwendungsbereiche. Mit mobilen Robotern, die in unbekannten Umgebungen arbeiten sollen, bietet sich aber ein Einsatzgebiet für die Technik.

4.2.7 Computer Vision für mobile Roboter

Kamera und Bildverarbeitung können in allen klassischen Aufgabengebieten der mobilen Robotik eingesetzt werden:

- Für die eigene Ortsbestimmung werden Kameras selten eingesetzt, es bieten sich aber einige Möglichkeiten an. Eine Orientierung an Hand von charakteristischen Elementen im Raum wäre eine landscape navigation. Eine einfache Variante sind Positionierungs- und Führungshilfen in Form von färbigen Linien am Boden oder an der Decke.

- Für Aufgaben mit Entfernungsmessung werden meistens Stereokamerasysteme eingesetzt, die eine räumlichen Wahrnehmung ermöglichen. Es gibt bereits kommerzielle integrierte Stereokamerasysteme, die als Sensor am Roboter verwendet werden können. Der Einsatzbereich solcher Systeme wird durch den Abstand der beiden Kameras beschränkt: Je größer der Abstand, desto „tiefer“ kann das System räumlich sehen. Der Abstand kann aber nicht beliebig gross sein, da sonst die beiden Einzelbilder von der Bildverarbeitung

nicht mehr in Übereinstimmung gebracht werden können. Mit den aktuell erhältlichen Systemen ist die räumliche Wahrnehmung auf wenige Meter beschränkt. Im eingeschränkten Maß sind aber auch mit Einzelkamerasystemen Entfernungsmessungen möglich. Ist etwa die Größe eines erkannten Objekts bekannt, so kann aus der Abbildungsgröße des Objekts seine Entfernung errechnet werden. Ein anderes Beispiel wäre die Annahme, dass die erkannten Objekte auf einer gemeinsamen Ebene liegen, z.B. auf dem Boden vor dem Roboter. Das entspricht dann einer Schnittebene durch den Raumwinkel den die Kamera wahrnimmt, wodurch jede Position im Bild auf einen Abstand vor dem Roboter umgerechnet werden kann.

- Kameras werden noch nicht zur Berührungserkennung zwischen Roboter und Umwelt eingesetzt, die technische Möglichkeit besteht aber durchaus. Ein einfaches Beispiel wäre ein Servierroboter, der mit einer Kamera den eigenen Arm überwacht, um ungewollte Berührungen mit Gegenständen zu vermeiden oder zumindest zu erkennen.

- Die größte Stärke von Machine Vision ist zweifellos die Objekterkennung. Kein anderer Sensor bietet die Möglichkeit Objekte derart umfassend zu erkennen und zu unterscheiden. Als Erkennungskriterien können Form, Farbe, Struktur oder ausgewählte Merkmale benutzt werden. Eine gute Objekterkennung ist nicht nur in der Lage, das gesuchte Objekt mit hoher Wahrscheinlichkeit zu erkennen, sondern findet auch mit ebenso hoher Wahrscheinlichkeit kein Objekt, wenn das gesuchte Objekt nicht im Bild vorkommt.

4.2.8 Computer Vision in der behavior-basierten Robotik

Das Ziel von Computer Vision ist es aus den Inputdaten der Kamera eine sinnvolle symbolische oder geometrische Repräsentation der Welt zu produzieren. Dieser Ansatz liefert zwar für viele technische Probleme eine passende Lösung, beachtet aber die besonderen Anforderungen von Behaviors nicht. Die Schuld an der langsamen Entwicklung der Computer Vision für behaviorbasierte Roboter wurde meist auf die fehlende Rechenleistung oder die unzureichende Qualität der Computer Vision geschoben. Arkin sieht aber drei grundsätzliche Probleme [Ark98]:

- Die Computer Vision versteht sich zu sehr als isolierte Wahrnehmung. Statt die Bilder isoliert zu interpretieren, sollte die visuelle Wahrnehmung ein Teil der gesamten Wahrnehmung sein. Für ein System ist die Wahrnehmung ein holistischer Prozess, der sowohl mit dem kognitiven als auch mit dem Bewegungssystem verbunden ist.

- Computer Vision ist ein spezialisiertes und sehr schwieriges Fachgebiet. Sie sieht sich selbst manchmal zu elitär als beste Lösung zur Wahrnehmung. Das mag zwar oft stimmen, aber als Sensor für die Robotik ist die Computer Vision nur eine von mehreren Varianten. Das setzt auch gewisse Teamfähigkeit der Computer Vision voraus.

- Das Ziel der Computer Vision ist oft die möglichst korrekte dreidimensionale Abbildung der Umwelt. Diese Modelle entsprechen oft nicht den Bedürfnissen einer reaktiven Robotersteuerung. Die Abbildung soll weder die Realität nachmodellieren, noch menschlichen Bedürfnissen entsprechen. Es gilt eine geeignete visuelle Wahrnehmung für den Roboter zu finden.

Zu diesen Kritikpunkten lassen sich folgene Lösungsvorschläge finden:

Abbildung 4.6: Reiz-Reaktionsdiagramm eines Behaviors mit Computer Vision nach Arkin [Ark98]. Sensor und Action entsprechen Reiz und Reaktion. Im neu dazugekommenen Schritt Perception wird die Bildverarbeitung extra beachtet.

- In der behavior-basierten Robotik ist Computer Vision einer von mehreren Sensoren, um die Umwelt wahrzunehmen. Die visuellen Sensordaten sollten auch nicht a priori bevorzugt werden. Vielmehr sollen die Daten mit Information über ihre Qualität behaftet werden, beispielsweise mit einer Wahrscheinlichkeit ob es sich beim gefundenen Objekt um das gesuchte handelt. Das behavior-basierte System kann mehrere derartige Informationen verarbeiten und eine Reaktion bestimmen.
- Computer Vision ist als Sensor für Behaviors funkionell gleich jeder anderen Art von Sensor. In der praktischen Applikation stellt die aufwendige Bildverarbeitung aber eine Besonderheit dar. Diesen Umstand hat auch Arkin beachtet, in dem er das schematische Reiz-Reaktionsdiagramm für Computer Vision etwas erweitert hat (Abbildung 4.6).
- Offen bleibt die Frage nach der geeigneten visuellen Wahrnehmung für Roboter. Sieht man Computer Vision als einen von mehreren Sensoren, so verschiebt sich auch die Bedeutung der Wahrnehmung. Besitzt der Roboter einen robusten Laser Entfernungsmesser, so ist die Bedeutung von visuell wahrgenommenen Entfernungen viel weniger wichtig, was nicht bedeutet, dass man auf die redundante Information verzichten sollte. Die Stärke der Computer Vision für Behaviors liegt aber vielmehr in der Objekterkennung und relativen Ortsbestimmung (z.B. das Objekt liegt weit links aber nahe vor dem Roboter).

4.2.9 Ausgewählte Beispiele

Die Verbindung zwischen Computer Vision und mobiler Robotik ist ein offener Forschungsbereich der sich laufend ändert. Der Stand der Technik wird hier mit zwei kleinen, aber dafür aktuellen Beispielen illustriert. Die Konzepte hinter den beiden Implementierungen zeigen deutlich die aktuellen Trends.

4.2.9.1 Remote Objects Control Interface

Die *Remote Objects Control Interface* (ROCI) Plattform [CCT06] ist eine Software Plattform für mobile Roboter. Die Plattform unterstützt ein verteiltes System und ein Programmiermodell mit wiederverwendbaren Modulen. Diese wiederverwendbaren ROCI Module kapseln jeweils einen Prozess ein, der mit der restlichen Plattform nur über eine Input- und eine Outputschnittstelle kommuniziert. Jedes der Module ist eigenständig und mehrfach verwendbar, so dass man eine komplexe Aufgabe durch eine Kombination von Modulen über Input- und Outputdaten lösen kann. Eine besondere Eigenheit ist das minimalistische Design. Die Module sind so einfach wie möglich, es gibt klare Interfaces und die Eigenschaften überlappen sich nicht. Abbildung 4.7 zeigt ein Beispiel wie man aus diesen einfachen Elementen eine Anwendung zusammenstellt.

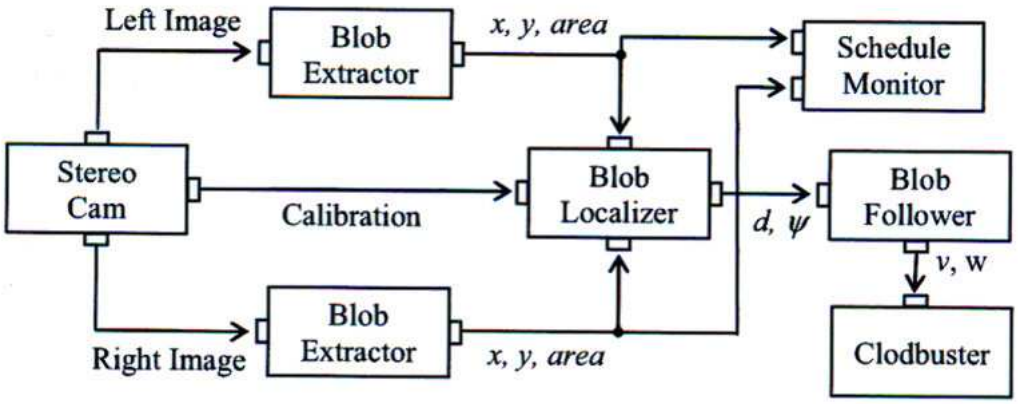

Abbildung 4.7: Beispiel eines mit der ROCI Plattform implementierten Behaviors. Das Diagramm zeigt alle Module eines *Blob following behaviors*, mit dem ein Roboter einer färbigen Fläche folgt. Beginnend links ist das Modul der Stereokamera, das zwei Bilder aufnimmt. Die Bilddaten werden an zwei Module mit Blob Extractoren weitergegeben, die nach farbigen Flächen suchen. Die gefundenen Koordinaten sowie die Kalibrierung der Stereokamera sind Inputs für den Blob Localizer, der die Koordinaten der farbigen Fläche dreidimensional bestimmt. Das Ergebnis dient dem Blob Follower zur Berechnung des gewünschte Verhaltens. Die Umsetzung des Verhaltens auf die eigentliche Roboterplattform übernimmt das Clodbuster Modul. Der nicht obligatorische Schedule Monitor dient zur Verbesserung der Bilderkennung. Abbildung aus [CCT06].

4.2.9.2 Das VIP-Framework

Das *Video Image Processing* (VIP) *Framework* [UKMK06] ist ein Software Framework für Robot Vision Applikationen in dynamischen Umgebungen, wie etwa Roboter Fussball (RoboCup). Das Ziel des Designs ist es eine Lösung für die Verwendung von aktuellen Computer Vision Algorithmen in der Robot Vision zu finden. Selbst bei einer optimierten Implementierung ist der Ressourcenbedarf dieser Algorithmen für eine schnelle Abarbeitung in Echtzeit meist zu hoch. Ein Framework bietet dazu den Vorteil, dass neben Softwareteile auch Design wiederverwendet und adaptiert werden kann.

Das Design des VIP Framework legt besonderen Wert auf den Kontroll- und den Datenfluss, um zeitkritische Module schneller mit Daten versorgen zu können, während weniger kritische Module die Ergebnisse von langsameren Computer Vision Algorithmen erhalten. Die Grundidee eines Framework ist es, dass der Systemdesigner nur die individuelle Funktionalität einer Applikation implementiert und das Framework die Ausführung entsprechend der implizierten Logik durchführt. Das unterstützt die Skalierbarkeit und die Reaktivität des Systems. Das VIP Framework organisiert den Datenfluss durch verschiedene Filter (einzelne Arbeitsschritte mit Input-Output Mapping) der Bildverarbeitung. Der Datenfluss ist dabei in einer Baumstruktur, einzelne Zweige können aber multithreaded nebeneinander abgearbeitet werden. Das Framework gewährleistet dabei, dass trotz der asynchronen Verarbeitung im gesamten Bildverarbeitungszyklus mit den Daten des selben Originalbilds gearbeitet wird. Im Framework werden außerdem eine Reihe von Standardfunktionen bereitgestellt sowie Anbindungen an Middleware (externe Programmbibliotheken) unterstützt.

Ein Beispiel für eine Anwendung des VIP Framework ist eine Steuerung für einen RoboCup

Roboter. Die Applikation besteht aus 66 Filtern mit 108 Verbindungen. Der schnellste Datenfluss ist eine farbbasierte Tor-Erkennung, die etwa 4 ms benötigt. Der langsamste Datenfluss mit etwa 20 ms ist die Klassifizierung von Robotern über ein neurales Netzwerk.

4.3 Zusammenfassung

Mobile Roboter können mit einer breiten Palette von Sensoren ausgestattet werden. Kameras stellen eine besondere Form von Sensoren dar, da zum Informationsgewinn noch ein aufwendiger Bildverarbeitungsschritt notwendig ist. Dafür bietet Machine Vision aber auch vielfältige Einsatzmöglichkeiten und erlaubt eine besonders gute Objekterkennung. Es wäre aber falsch anzunehmen, dass eine Kamera der beste Sensor für einen mobilen Roboter ist. Die beste Lösung ist vielmehr eine Menge unterschiedlicher Sensoren, die ihre Nachteile gegenseitig kompensieren und gleichzeitig redundante Daten liefern. Diese Verschmelzung der Daten verschiedener Sensoren nennt man sensor fusion. Ein derartiges holistisches System kann auf falsche Werte einzelner Sensoren stabil reagieren und ist dadurch robuster.

Kapitel 5

Demonstrationsbeispiele

Die Kapitel 2 bis 4 haben die Grundlagen von mobilen Robotern, reaktiven Steuerungen und Machine Vision behandelt. In diesem Kapitel werden die praktischen Demonstrationsbeispiele erörtert, die im Rahmen dieser Arbeit entwickelt wurden. Mit diesen Beispielen wurden die Möglichkeiten ausgetestet, einen mobilen Roboters nur mit visuellen Behaviors zu navigieren. Ausserdem wurden die Demonstrationsbeispiele so ausgewählt und umgesetzt, dass sie im Rahmen von Vorführungen oder in der Lehre einfach wiederverwendet werden können.

5.1 Das Setup für die Beispiele

Die Aufgabenstellung für die Demonstrationsbeispiele lautete, mit der im Institut selbst entwickelten Softwareumgebung S4R-Framework den radgetriebenen Roboter „James“ vom Typ Pioneer 2-DX visuell zu navigieren (Abbildung 5.1). Die folgenden Hard- und Softwarekomponenten kamen dabei zum Einsatz.

5.1.1 Die verwendete Hardware

ActivMedia Robotics Pioneer 2-DX Roboter Radgetriebener mobiler Roboter für allgemeine Anwendungen mit quasi-holonomen Differentialantrieb mittels zwei Antriebsrädern und einer Nachlaufrolle [AMR06]. Der Roboter ist mit Winkelenkoder an den Antriebsachsen für die Odometrie und acht Ultraschallsensoren in einem Sonar-Halbring ausgestattet. Die Energieversorgung erfolgt über im Chassis untergebrachte 12V/7Ah Bleiakkus. Die ursprünglichen Gummireifen wurden gegen Aluminiumräder mit einer Rundschnur als Lauffläche ausgetauscht [Ein05]. Der Roboter hat eine Transportplattform in 23,5 cm Höhe über dem Boden und kann laut Datenblatt mit einer Nutzlast von 20 kg beladen werden. Auf dieser Transportplattform wurde ein einfacher Aufbau aus Aluminiumprofilen montiert. Dieser dient als Stativ für eine Kamera und als Halterung für einen Laptop.

Fujitsu Siemens Laptop „Apollo“ mit Intel Pentium 4 Mobile CPU 1.60GHz, 512 MBytes Hauptspeicher und Suse 9.2 als Betriebssystem [Anhang A].

The Imaging Source DFK 21F04 Farbkamera mit FireWire Anschluss [TIS06]. Die Kamera ist auf der linken Seite eines Aluminiumstativs in 100 cm Höhe montiert. Die Energieversorgung erfolgt über die Akkus des Roboters durch eine zusätzliche Buchse am

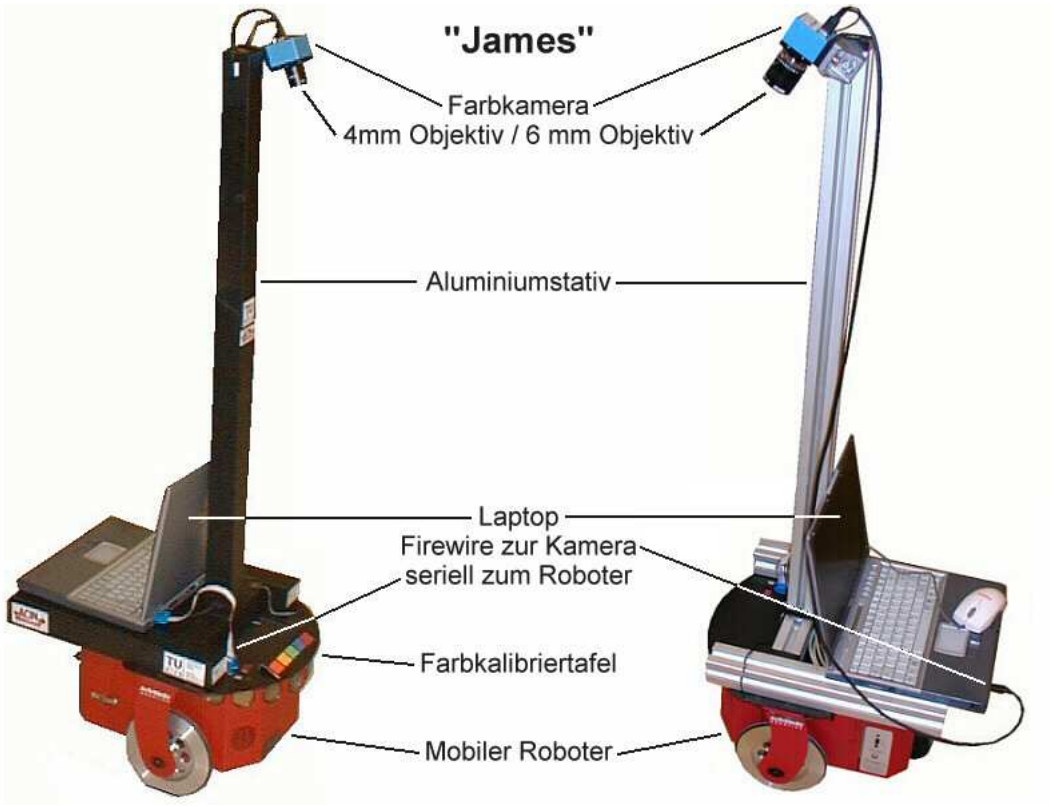

Abbildung 5.1: Zweimal der ActivMedia Robotics Pioneer 2-DX Roboter James. In beiden Fällen mit Stativ, Kamera und Laptop. Rechts ist die zuerst verwendete Konfiguration mit 6 mm Objektiv abgebildet. Links ist James mit schwarz verkleidetem Stativ und 4 mm Objektiv abgebildet.

Roboterchassis. Die Kamera ist nach unten gerichtet, um den Boden vor dem Roboter abzubilden. Der Neigungswinkel und das verwendete Objektiv variieren bei den Beispielen.

5.1.2 Die verwendete Software

Als Software wurde das am Institut entwickelte S4R-Framework eingesetzt. Das Framework ist eine Weiterentwicklung des ActIPret-Frameworks, das im Rahmen des gleichnamigen EU-Projekts *ActIPret* [ACT06] entwickelt wurde. Das Framework ist eine in der Sprache *C++* unter Linux entwickelte Plattform für ein multifunktionales Bildverarbeitungssystem, das in den Bereichen mobile Roboter, automatisierte Überwachungssysteme und kognitive Systeme einsetzbar ist. Frameworks gehören zur zweiten Generation von abstrakten Software Architekturen, die die Wiederverwendung von Programmteilen und Softwaredesigns unterstützen. Dazu bietet das Framework eine Sammlung von Algorithmen aus dem Bereich der Bildverarbeitung und Robotersteuerung, die in Form von Services angeboten werden. Anwendungen sind dabei Service Requester, die die passenden Dienste aus dem vorhandenen Pool von Services nutzen. Eine umfassendere Beschreibung der Struktur des Framework findet sich bei [Ein05]. Ein Behavior zur visuellen Navigation eines Roboters benötigt eine Reihe von solchen Services: Eine Steuerung des Roboterantriebs, eventuell noch weitere Services des Roboters (z.B. Odometrie),

ein Service zur Einbindung der Farbkamera ins System, eine oder mehrere Bildverarbeitungen und eine grafische Benutzeroberfläche zur Visualisierung und Bedienung. Im Konkreten kamen folgende Services zum Einsatz:

Mobile Robot Services Ein Servicepaket, in dem alle vom mobilen Roboter angebotenen Dienste im Framework genutzt werden können. Für die Beispiele wurden die beiden angebotenen Services zur Steuerung das Antriebs verwendet:

- **RobotContinuousMotion** arbeitet mit den drei Befehlen Fahren, Drehen und Stopp. Die Befehle Fahren und Drehen können gleichzeitig ausgeführt werden und erzeugen eine natürliche Fahrbewegung. Eine Bewegung kann nur durch den Stopp Befehl oder durch das Nullsetzten der Fahr- und Drehgeschwindigkeit beendet werden.
- **RobotDiscretMotion** arbeitet mit absoluten Fahr- und Drehbefehlen, die als atomare Aktionen sequenziell abgearbeitet werden müssen. Die Befehle verwenden die Odometriedaten des Roboters, um ein gewünschtes Ziel zu erreichen. Es wird kein zeitkritischer Stopp-Befehl zum Halten benötigt, dafür wirken die Fahrbewegungen unnatürlich.

Farbtracker CMV_Track2D Der CMV-Tracker arbeitet mit der Color Machine Vision Library der Carnegie Mellon University in Pittsburgh [CMV06]. Der Algorithmus arbeitet im YUV-Farbraum und kann Objekte, die durch einen gesuchten Farbbereich beschrieben sind, im Bild finden. Nach dem Einlesen des Bildes und einer Konvertierung in den YUV-Farbbereich werden vier Schritte durchgeführt: Bei der Segmentierung werden die einzelnen Pixel überprüft, ob sie dem gesuchten Farbbereich angehören oder nicht. Nun werden die einzelnen Pixel zu Regionen zusammengefasst. In einem dritten Schritt werden für die Regionen Fläche, Schwerpunkt und Koordinaten eines umschreibenden Rechtecks berechnet. Und zuletzt werden nebeneinander liegende Regionen, die möglicherweise zu einem Objekt gehören, zusammengefasst. Die Geschwindigkeit, mit der die Bildverarbeitung ein Bild analysiert, hängt abgesehen vom verwendeten Rechner (CPU, Speicher) hauptsächlich von der Bildgröße (640 x 480 Pixel, 320 x 200 Pixel) ab.
Das Anlernen eines neuen Objekts erfolgt mit einer eigenen Teach-In Software vor der eigentlichen Benutzung. Es werden mindestens zwei Bilder benötigt, eines vom Hintergrund und ein weiteres mit dem Objekt vor dem Hintergrund. Durch Differenzbildung kann der Hintergrund ausgeblendet werden und man erhält ein reduziertes Bild, das nur das Objekt enthält. Objektschatten und geänderte Lichtverhältnisse sollten vermieden werden. Mehrere Bilder können den Lernprozess verbessern. Aus den Farbwerten der Pixel sämtlicher Differenzbilder wird der Mittelwert und die Standardabweichung gebildet. Diese Werte repräsentieren das Objekt und werden im YUV-Farbraum kodiert und unter einem wählbaren Namen gespeichert [Zac03].

Farb-Recogniser ContinousRecognise2D Ein spezieller für die Verwendung mit dem mobilen Roboter optimierter Algorithmus, der Farbbereiche von bis zu fünf Farben in einem Bild sucht. Eine am Roboter montierte Tafel liefert die Referenzfarben, ein Anlernen mittels Trainingsprogramm ist daher nicht notwendig. Eine kurze Beschreibung der besonderen Eigenschaften dieses Farb-Recogniser findet sich in Kapitel 5.5.3.

S4R-Framework GUI Die grafische Benutzeroberfläche des Frameworks. Sie erlaubt eine Darstellung des Livebilds mit zehn Bildern pro Sekunde, wobei mit zusätzliche Markierungen im Bild (Linien, Rechtecke) gefundene Objekte hervorgehoben werden können.

5.2 Follow Ball Behavior

5.2.1 Die Aufgabe

Die Grundidee ist einfach: James soll einem Zielobjekt kontinuierlich folgen. Ähnlich einem Haustier das hinter einem Ball herläuft, soll der Roboter einem bewegten Ziel hinterherfahren. Dabei soll er immer einen „Respektabstand“ einhalten, um das Objekt im Auge zu behalten. Für die Demonstration wurde ein blauer Ball als Zielobjekt ausgewählt. Diese Art von Behavior ist als „Follow Me“ bekannt und wird gerne als Beispiel für Behaviors verwendet. Diese Variante wird zur einfacheren Kennzeichnung „Follow Ball“ genannt.

Eine Implementierung eines Follow Me Behaviors wird mit der Softwareumgebung Saphira [AMR06] für den Pioneer 2-DX Roboter mitgeliefert. Dabei wird aber keine Kamera verwendet, sondern die acht Ultraschallsensoren am Sonarring des Roboters dienen zum Wahrnehmen der Umgebung. Eine weitere Implementierung des Follow Me Behaviors mit Ultraschallsensoren findet sich auch im S4R-Framework. Im Rahmen der MobileRobot Services dient dieses Behavior als Test- und Demonstrationsprogramm für die Robotersteuerung. Trotz der unterschiedlichen Sensorarten ist das Ziel des Behaviors ein ähnliches Fahrverhalten, wodurch sich die Ergebnisse leicht vergleichen lassen. Der auffälligste Unterschied der Implementierungen ist der von den jeweiligen Sensoren überwachte Bereich. Der Sonarring kann in einer Halbebene einen Bereich von etwa 20 cm bis 7 m erfassen, wobei das Behavior diesen Bereich aber auf einen Halbkreisbogen von 40 cm bis 1,20 m beschneidet, um störende Hindernisse in grösserer Entfernung auszublenden. Beim Follow Ball ist der erfassbare Bereich eine trapezförmige Bodenfläche vor dem Roboter, die Größe der Fläche wird durch das gewählte Kameraobjektiv bestimmt.

Da dies die erste Implementierung im Rahmen dieser Arbeit war, mussten ein Reihe von grundsätzlichen Aufgaben bezüglich des Systems gelöst werden:

- Ist der Stativ-Aufbau stabil genug für eine Kamera und nicht zu schwer für den Roboter?
- Bleibt der Roboter durch den erhöhten Schwerpunkt bei Bodenunebenheiten im Gleichgewicht?
- Die Montage und Ausrichtung der Kamera. Wie kann das begrenzte Sichtfeld optimal genutzt werden?
- Ist der verwendete Bildverarbeitungsalgorithmus überhaupt geeignet, um damit einen Roboter zu navigieren?

5.2.2 Das Behavior

Das Verhalten des Roboters soll nur durch die Position des Zielobjekts bestimmt werden:

- Liegt das Objekt links oder rechts vor dem Roboter, so soll er sich zum Objekt hindrehen (Interesse).
- Liegt das Objekt weiter entfernt, so fährt er darauf zu (Anziehung).
- Liegt das Objekt zu nahe, so setzt der Roboter zurück (Abstoßung).

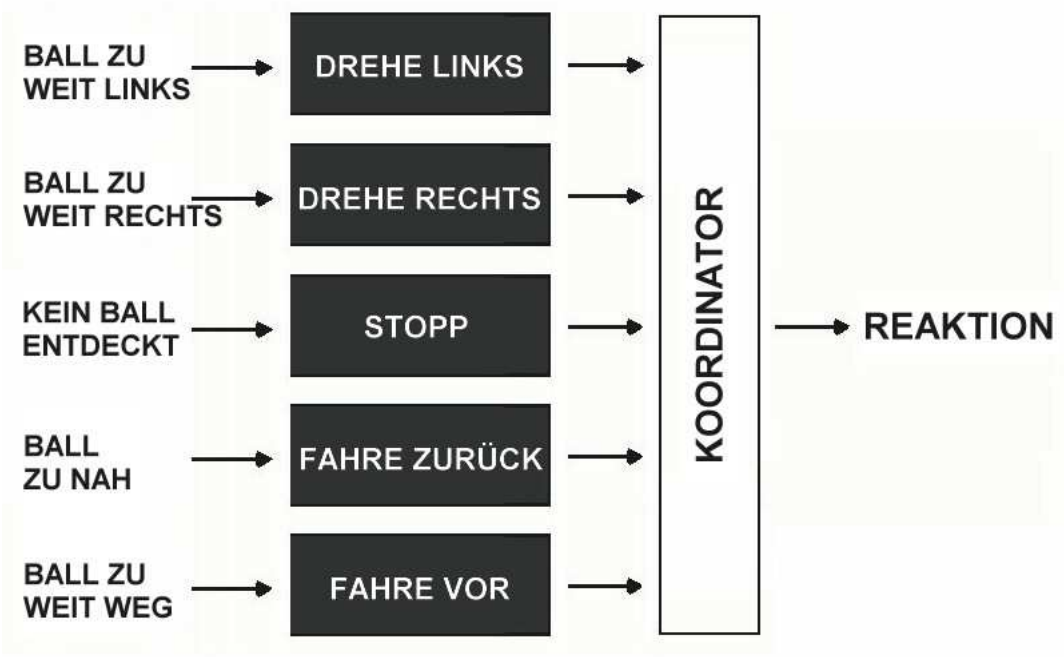

Abbildung 5.2: Das Follow Ball Behavior als Reiz-Reaktionsdiagramm. Das Behavior kennt fünf verschiedene Reize, die jeweils unterschiedliche Reaktionen hervorrufen. Die Reaktionen werden gleichzeitig ausgeführt, sofern sie sich nicht gegenseitig aufheben.

- Es gibt eine Zone vor dem Roboter in „optimaler“ Entfernung. Solange sich das Objekt in dieser Zone befindet, bewegt sich der Roboter nicht.
- Befindet sich kein Zielobjekt im Bild, steht der Roboter ebenfalls still.

Die Struktur ist als Reiz-Reaktionsdiagramm in Abbildung 5.2 dargestellt.

5.2.3 Die Komponenten

- Farbtracker CMV_Track2D: Der Farbtracker wurde auf einen blauen Ball mit 10 cm Durchmesser für unterschiedliche Lichtsituationen angelernt (siehe Abbildung 5.6).
- Die DFK 21F04 Farb-Kamera wurde mit einem 6 mm Objektiv ausgestattet und mit einer Neigung von 50° zur Horizontalen montiert. Sie erfasst damit eine Fläche von etwa 0,64 m vor dem Roboter. Der Testaufbau ist in Abbildung 5.3 dargestellt. Der Bereich direkt vor dem Roboter wurde absichtlich ausgespart, um weiter nach vorne sehen zu können. Durch die einseitige Montage am Stativ ist das Gesichtsfeld vor dem Roboter links größer als rechts.
- Die Steuerungsmethode *RobotContinuousMotion* wurde verwendet.

5.2.4 Die Analyse

Das FollowBall Behavior konnte die Aufgabenstellung erfüllen. Im direkten Vergleich mit dem auf Ultraschall basiertem Vergleichsbehavior zeigt das visuelle Behavior eine glattere und na-

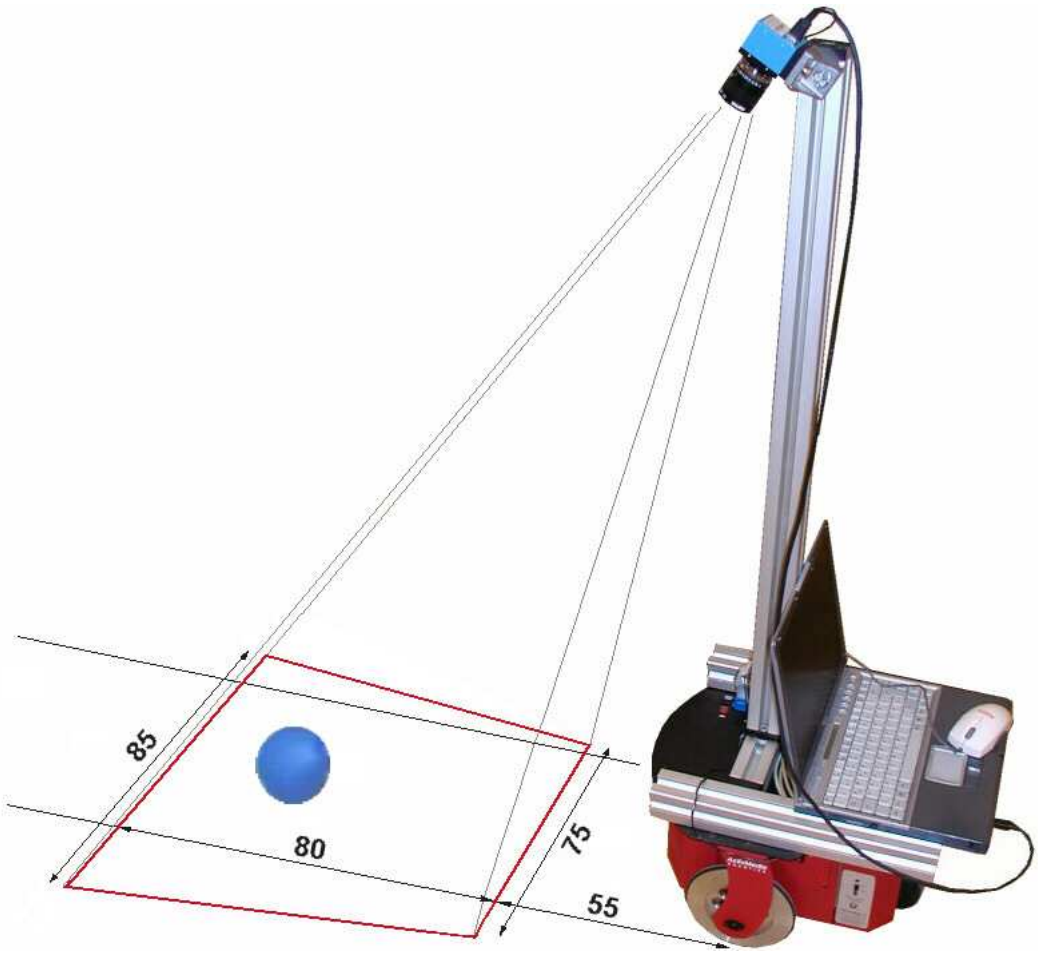

Abbildung 5.3: Der Pioneer 2-DX Roboter mit Aufbau, auf dem Farbkamera und Laptop befestigt sind. Die betrachtete Bodenfläche ist schematisch dargestellt. Alle Zahlen sind in cm angegeben.

türlichere Fahrbewegung. Ein Grund dafür ist die höhere Datenfrequenz der Kamera und Bildverarbeitung, im Beispiel waren es mehr als zehn Bilder pro Sekunde, gegenüber der konstanten Datenfrequenz der Sonarsensoren von etwa 3 Hz. Abbildung 5.4 zeigt die Sicht des Roboters wobei die Geschwindigkeit der Bildverarbeitung anhand der markierten Positionen der letzten vier Sekunden deutlich wird. Ein zweiter Grund ist die höhere Auflösung der Kamera, die den Boden mit etwa 4 Pixel/cm horizontal und etwa 3 Pixel/cm vertikal abbildet. Im Vergleich dazu arbeiten die Sonarsensoren mit einer Winkelauflösung von 20°.

- Das Gewicht des Stativ-Aufbaus und des Laptops liegt unter der angegebenen Nutzlast des Roboters von 20 kg. Trotzdem zeigt sich bereits ein Einfluss auf das Fahrverhalten. So können beispielsweise Übergangsschwellen von 1 cm Höhe nur noch schwer überquert werden.

- Die Kamera mit 6 mm Objektiv sieht eine ausreichend grosse Bodenfläche für langsam bewegte Objekte. Bei sehr schnell bewegten Objekten ist aber eine kürzere Brennweite bzw. eine größere beobachtete Bodenfläche nötig.

- Die Asymmetrie des Gesichtsfelds hat keinen negativen Einfluss auf das Behavior.

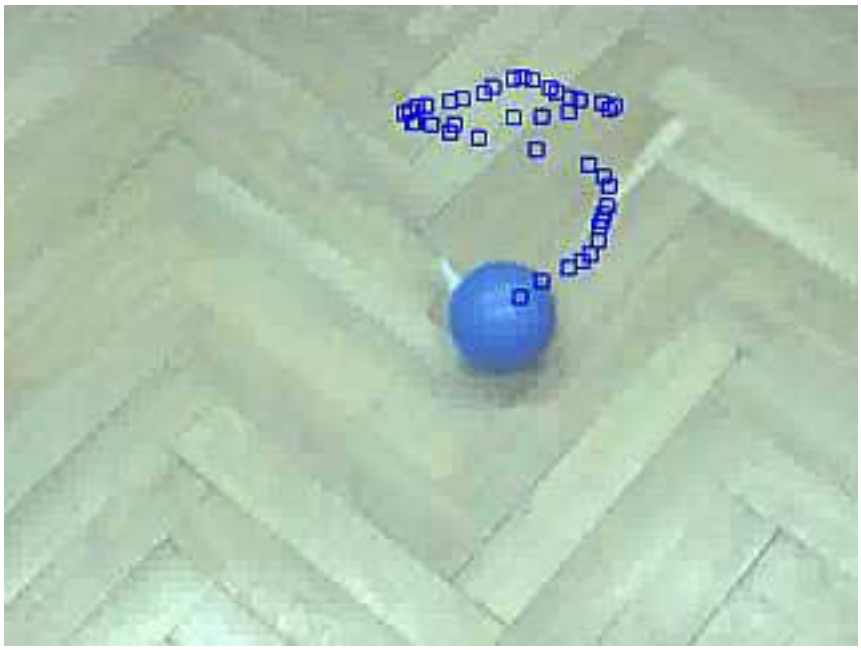

Abbildung 5.4: Die Umgebung aus der Sicht des Roboters: Momentaufnahme der S4R-Framework Benutzeroberfläche. Die kleinen Quadrate markieren die 40 detektierten Positionen des blauen Balls in den letzten vier Sekunden.

- Um die nötige Effizienz der Bildverarbeitung auf der verwendeten Hardware zu erreichen, wurde ein Subsampling des betrachteten Bildes im Farbtracker auf 320 x 200 Bildpunkte durchgeführt und eine optimierte Kompilierung des gesamten Quellcodes (Compilerflag `-O3`) durchgeführt. Damit konnte eine Verarbeitungszeit von weniger als 0,1 s je Bild erreicht werden, womit eine flüssige Fahrbewegung möglich war.

- Die Vibration des Stativs und der Kamera durch die Fahrbewegung des Roboters hat einen negativen Einfluss auf die Bildqualität und Genauigkeit der Positionsbestimmung von Objekten relativ zum Roboter. Bei einer angenommenen maximalen einseitigen Bodenunebenheit von 5 mm schwankt das Kamerastativ weniger als 2°. Der im Bild wahrgenommene Ball liegt damit maximal 3 cm neben der ermittelten Position. Dieser Fehler wird aber durch die hohe Bildfrequenz kompensiert, da der Roboter durch seine Masseträgheit nur begrenzt schnell auf falsche Positionen reagieren kann. Eine hohe Datenfrequenz ist ein sicheres Mittel, um damit einzelne falsche Messwerte auszusparen.

- Ein Problem stellt die sichere Erkennung bzw. Nicht-Erkennung eines Objektes dar. Konfiguriert man den Farbtracker so, dass der Ball mit hoher Wahrscheinlichkeit erkannt wird, so steigt auch die Wahrscheinlichkeit der Fehlinterpretation anderer Objekte wie Schatten oder Bodentexturen als blauer Ball. Das Problem ist, dass der Farbtracker die Prämisse hat, dass das Objekt sicher im Bild ist und nur noch gefunden werden muss. Die Erkennung hat damit eine höhere Priorität als die Nicht-Detektion des Objekts. Für den Roboter stellen die beiden Aussagen Objekt-gefunden und Objekt-nicht-gefunden aber gleichwertige Informationen dar. Beim Anlernen der gesuchten Farbe für das Beispiel musste der Schwellwert so gesetzt werden, dass der Roboter nicht Bodentexturen mit einem kleinen Blauanteil zu verfolgen beginnt, sobald der blaue Ball nicht mehr für die Kamera sichtbar ist.

5.3 Dual Color Behavior

5.3.1 Die Aufgabe

Die Resultate vom Follow Ball Behavior zeigten, das ein reaktives Verhalten mit der Kamera als Sensor möglich ist. Die Möglichkeiten mittels eines einfärbigen Objekts eine Navigation zu realisieren sind aber sehr begrenzt. Darum sollen im zweiten Beispiel zweifärbige Objekte erkannt werden. Die Aufgabe lautete: Der Roboter soll eine Strecke abfahren, die durch rot-blaue Marker gekennzeichnet ist.

Das besondere an zweifärbigen Markern ist, dass sie nicht nur eindeutiger identifiziert werden können als ein einfärbiger Ball, sondern das sie eine zusätzliche Information abbilden können. Ein zweifärbiger Marker kann nicht nur eine Position markieren, sondern zusätzlich noch eine Richtung angeben. Im Beispiel soll sich die Richtung normal zur Verbindungslinie des roten (links) und blauen (rechts) Farbschwerpunkts ergeben.

5.3.2 Das Behavior

Der Roboter soll einen Marker sicher erkennen, sich zum Marker drehen, ihn anfahren, sich in die durch den Marker angegebene Richtung drehen und nun mit dem nächsten Marker fortfahren. Dieses Behavior kann durch ein FSA Diagramm mit folgendem Quadrupel beschrieben werden:

$$M = \{\{Start, Suche, Fahrt\}, \delta, Start, \{\}\} \tag{5.1}$$

Die Liste der Übergangsaktionen δ ist in Tabelle 5.1 abgebildet und die grafische Darstellung des FSA Diagramms findet sich in Abbildung 5.5. Es gibt in dem Behavior keinen definierten Endzustand, daher ist die entsprechende Liste im beschreibenden Quadrupel leer.

δ	q	*input*	$\delta(q,input)$
	Start	alle	Suche
	Suche	Marker gefunden	Fahrt
	Suche	Marker nicht gefunden	Suche
	Fahrt	Fahrt beendet	Suche
	Fahrt	Fahrt nicht beendet	Fahrt

Tabelle 5.1: Liste der Übergangsaktionen δ zum Dual Color Behavior.

5.3.3 Die Komponenten

- Die Marker sind etwa 3 mal 3 cm groß und zur Hälfte rot und zur Hälfte blau (Abbildung 5.6).
- Farbtracker CMV_Track2D: Es werden zwei Instanzen des Farbtrackers verwendet, eine wurde auf die Farbe der roten Seite und eine auf die Farbe der blauen Seite der Marker angelernt.
- Die DFK 21F04 Farb-Kamera wird wie beim ersten Beispiel mit 6 mm Objektiv verwendet.

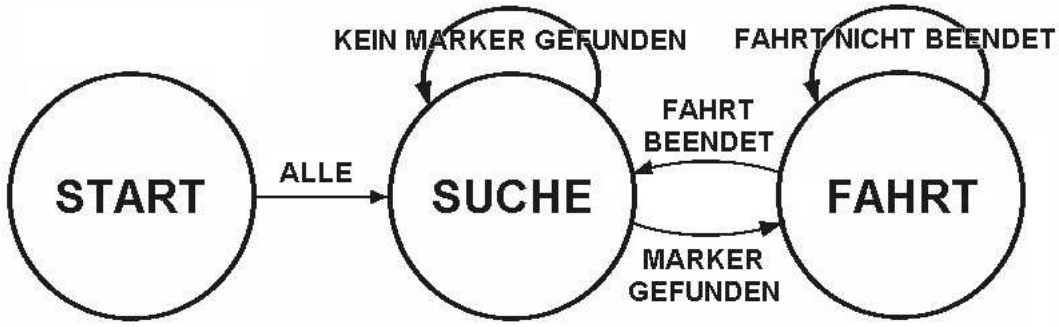

Abbildung 5.5: FSA Diagramm zum Dual Color Behaviors.

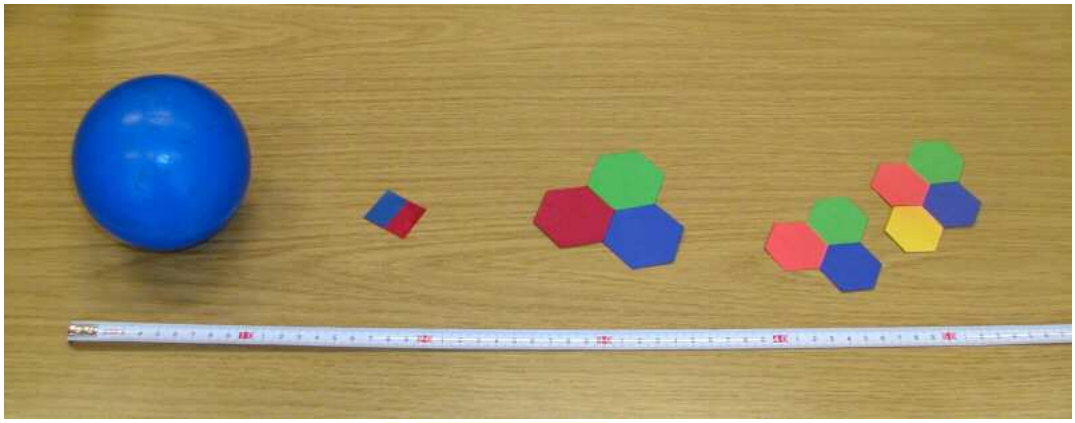

Abbildung 5.6: Die verschiedenen bei den Beispielen verwendeten Markierungen und Objekte. Von links nach rechts: blauer Ball (*Follow Ball Behavior*), zweifärbiger Marker (*Dual Color Behavior*), dreifärbiger Marker (*Triple Color Behavior*), drei- bzw. vierfärbige Marker (*Multi Color Behavior*).

- Die Steuerungsmethode RobotDiscretMotion wurden verwendet.

5.3.4 Die Analyse

Das Dual Color Behavior konnte die Aufgabenstellung erfüllen. Der Roboter folgt ohne Vorwissen über die Strecke den Markern, die auch nachträglich aufgeklebt oder verändert werden können, während der Roboter bereits fährt. Die Erkennung der Marker und der vermuteten Richtung der Marker ist in der Benutzeroberfläche des S4R-Frameworks (Abbildung 5.7) deutlich sichtbar. Ein visuell starkes Demonstartionsbeispiel erhält man, wenn der Roboter am Ende einer Markerkette stehen bleibt und auf eine leere Fläche sieht. Wenn man nun einen Marker aufklebt, kann man die sofortige Reaktion des Roboters auf den plötzlich vorhandenen Reiz beobachten.

- Die Einbeziehung von zusätzlichem Wissen in die Erkennung der Marker hat die True Positive Erkennung von Markern entscheident verbessert. Marker folgen der Regel, dass die beiden Schwerpunkte der Farbbereiche weniger als 10 Bildpunkte voneinander entfernt

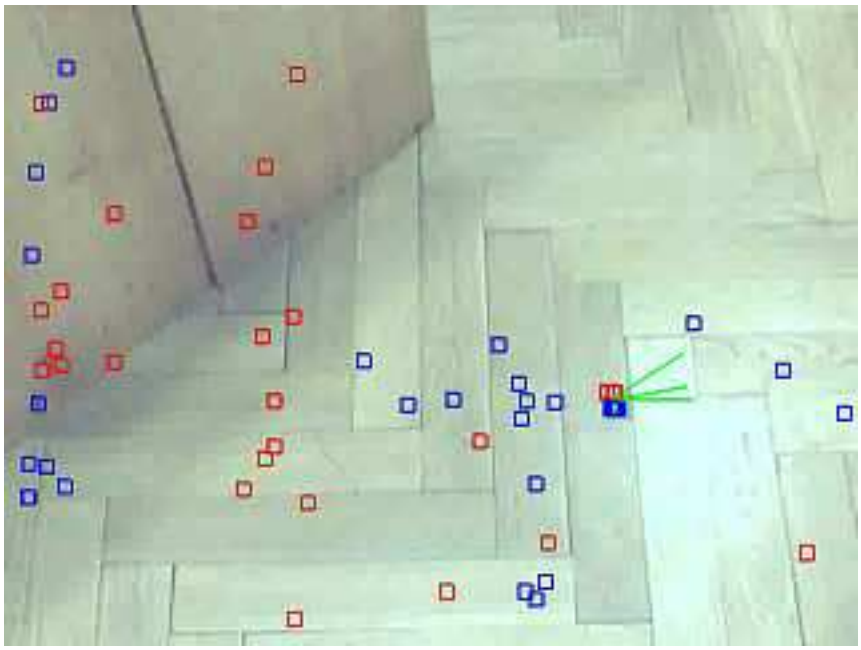

Abbildung 5.7: Momentaufnahme der S4R Benutzeroberfläche beim Dual Color Behavior. Die kleinen Quadrate markieren die 27 blauen und 27 roten detektierten Positionen für den Marker in den letzten vier Sekunden. Die meisten dieser Positionen ergeben keine korrekten rot-blau-Paare. Der Marker wurde drei Mal erkannt. Die grünen Striche zeigen die durch den Marker angegebene Richtung.

liegen müssen. Ist dies der Fall, so handelt es sich mit hoher Wahrscheinlichkeit um einen Marker.

- Keine Verbesserung bringt das Verfahren bei falsch erkannten Farbbereichen (False Positive). Bereits ein falsch erkannter Farbbereich verhindert die Detektion des Markers.
- Das resultierende Verhalten ist eine Verbesserung zum Follow Ball Behavior, bei dem der Roboter jedem Reiz - ob richtig oder falsch - direkt folgt. Der Roboter reagiert jetzt nur noch auf richtig erkannte Marker. Einfache Testläufe kann der Roboter damit erfolgreich absolvieren. Machen störende Objekte im Bild das Auffinden des nächsten Markers für den Roboter unmöglich, so bleibt er einfach stehen. Testfahrten durch allgemeine Räume sind damit noch nicht möglich.
- Durch die Verwendung von zwei Instanzen des Farbtrackers sank die Datenfrequenz auf 0,15 s je ausgewertetem Bild. Diese Geschwindigkeit erlaubt immer noch eine flüssige Bewegung des Roboters.
- Um die sichere Erkennung eines Markers zu gewährleisten, musste das Behavior den Marker öfters als nur ein Mal im Bild erkennen. Erst nach drei erfolgreichen Detektionen des Markers reagiert der Roboter auf den Reiz. Dieses Addieren des Reizes über die Zeit ist an ein biologisches Vorbild angelehnt: Erst die Senkung der Reizschwelle durch wiederholte schwache Reize erlaubt bei Nervenzellen eine Reizweiterleitung. Dadurch werden einzelne falsche oder unwichtige Reize schon vor der Auswertung unterdrückt.

- Mit der Größe der Marker konnte die Untergrenze einer für ein Behavior sinnvollen Detektion der beiden Farbbereiche gefunden werden. Im betrachteten Bodenbereich entsprach die Größe etwa zwölf mal neun Bildpunkten oder 54 Bildpunkten je Farbe.

- Ein Vorteil der kleinen Marker war die korrekten Positionen der Schwerpunkte der Farbbereiche. Die Richtung des Marker, die aus diesen beiden Positionen ermittelt wurde, konnte mit einer Winkelgenauigkeit von +/- 20° bestimmt werden. Diese Ungenauigkeit begründet sich vor allem aus der Nähe der beiden Farbbereiche zueinander. Mit der Entfernung der beiden Farben zueinander steigt auch die Genauigkeit der daraus bestimmten Richtung, aber gleichzeitig sinkt die Wahrscheinlichkeit der sicheren Detektion des Markers.

5.4 Triple Color Behavior

5.4.1 Die Aufgabe

Im dritten Beispiel soll das Follow Marker Behavior mit drei statt zwei Farben arbeiten. Mit Hilfe der dritten Farbe soll die Redundanz der vorhandenen Information die Qualität der Erkennung erhöhen. Das Verhalten des Behaviors soll gleich zum zweiten Beispiel bleiben: Ein Marker soll erkannt werden, angefahren werden und der Roboter soll sich in die vom Marker angegebene Richtung drehen. Durch das wiederholte Verhalten kann dann eine Kette von Markern abgefahren werden.

5.4.2 Die Komponenten

- Die DFK 21F04 Farb-Kamera wurde auf ein 4 mm Objektiv umgerüstet. Die betrachtete Bodenfläche bei einem Neigungswinkel von 61° zur Horizontalen beträgt etwa 1,4 m.

- Dreifarbige Marker, die sich aus drei Sechsecken in den Farben rot, blau und grün zusammensetzten (Abbildung 5.6.). Durch die neue Objektivwahl mussten die Marker größer dimensioniert werden, um im Bild einen ähnlich grossen Farbbereich darzustellen. Die Wahl fiel auf sechseckige Formen, weil sie bereits der Kreisform ähnlich sehen, die auch verzerrt gut wahrgenommen werden kann. Die gewählten Farben nutzen den RGB-Farbraum optimal aus. Die Schwerpunkte der drei Sechsecke bilden ein gleichseitiges Dreieck.

- Farbtracker CMV_Track2D: Es werden drei Instanzen des Farbtrackers verwendet, eine wird auf die rote, eine auf die blaue und eine auf die grüne Farbe der Marker angelernt.

- Die Steuerungsmethode *RobotDiscretMotion* wurde verwendet.

5.4.3 Die Analyse

Das Triple Color Behavior konnte die Aufgabenstellung nur teilweise erfüllen. Die notwendige Rechenleistung für drei parallel laufende Instanzen reduzierte die Datenfrequenz auf 0,35 s je ausgewertetem Bild. Eine sichere dreimalige Erkennung eines Markers dauert damit über eine Sekunde. In der Momentaufnahme der Markererkennung in Abbildung 5.8 sieht man deutlich, wie durch die langsamere Bildverarbeitung weniger Farbbereiche erkannt werden.

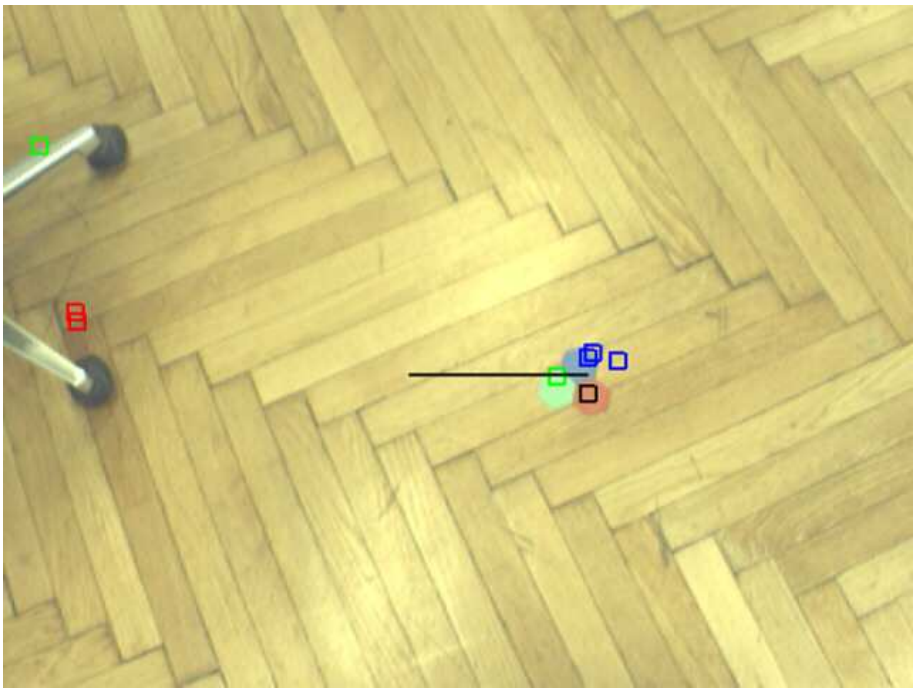

Abbildung 5.8: Momentaufnahme der S4R-Framework Benutzeroberfläche beim Triple Color Behavior. Die kleinen färbigen Quadrate markieren die blauen, roten und grünen Farberkennungen der letzten Sekunde. Im Bild ist ein Marker zu sehen. Der blaue und grüne Teil des Markers wurden erkannt, die rote Farbe wird links zwischen den Rollen des Sessels detektiert. Trotzdem wird der Marker richtig erkannt und der fehlende rote Markerteil durch einen berechneten Stützpunkt mit einem schwarzen Quadrat markiert. Die Richtung des Markers ist durch die schwarze Linie markiert.

- Die Reaktionsgeschwindigkeit des Behaviors sinkt durch die langsame Bildverarbeitung auf ein unzureichendes Mass, ein flüssiges Fahrverhalten ist nicht mehr möglich.
- Der grundsätzliche Gedanke, durch die Verwendung von drei Farben eine Redundanz zu erzeugen hat funktioniert. Die Erkennung der Marker bei realen Lichtbedingungen hat sich verbessert. Theoretisch hebt eine dritte Farbe die Erkennungsrate deutlich, da bereits zwei richtig erkannte Farbbereiche zum positiven Ergebnis führen. Wenn die drei Farben beispielsweise jeweils mit einer Wahrscheinlichkeit von 0.8 erkannt werden können, dann steigt die Wahrscheinlichkeit der richtigen Erkennung von Markern von 64% $(0,8*0,8)$ auf 90% $((0,8*0,8*0,8)+(0,8*0,8*0,2)*3)$. In realen Lichtsituationen zeigt meist eine der drei Farben eine deutlich schwächere Erkennungsrate, eine verbesserte Erkennungsrate der Marker ist trotzdem messbar.
- Eine grundsätzliche Einschränkung des verwendeten Farbtrackers ist die Inflexibilität durch das notwendige Anlernen der Farben vor der Verwendung des Trackers. Eine Neukalibrierung der Farben während des Einsatzes ist damit nicht möglich.
- Eine zweite Einschränkung stellt die einstellbare Farbtoleranz dar. Eine erlaubte Abweichung von der jeweils gesuchten Farbe ist notwendig, um die Farbe auch unter unterschiedlichen Lichtbedingungen detektieren zu können. Ist die erlaubte Abweichung aber zu gross gewählt, so werden oft falsche Farbbereiche (False Positive) erkannt. Die Farb-

toleranz muss ebenfalls vor der Verwendung des Trackers ausgewählt werden.

- Eine Randbedingung für die erfolgreiche Benutzung des Behaviors ist eine passend aufgelegte Reihe von Markern. Liegen die Marker zu weit voneinander entfernt, so kann der Roboter sie nicht wahrnehmen. Liegen Marker zu nahe beieinander, so dass zwei Marker im Bild sind, dann stört das die korrekte Erkennung.

5.5 Multi Color Behavior

5.5.1 Die Aufgabe

Das Multi Color Behavior verfolgt weiterhin das im zweiten und dritten Beispiel verwendete Verhalten. Es sollen mehrfärbige Marker, die eine Strecke markieren, sicher erkannt und abgefahren werden. Neu hinzugekommen sind vierfärbige Marker, die eine Veränderung des Verhaltens bewirken können. Im Beispiel soll ein solcher Marker verwendet werden, um den Roboter zu stoppen und um 180° zu drehen. Nach einem Befehl zum Fortfahren soll der Roboter dann die Strecke in umgekehrter Richtung zurückfahren. Dieses Verhalten simuliert beispielsweise einen Serviervorgang, bei dem der Roboter die Getränke zum Tisch des Gastes bringt, serviert und dann wieder zurückfährt.

5.5.2 Das Behavior

Das Verhalten setzt sich aus zwei getrennt voneinander betrachtbaren Behaviors zusammen. Im ersten Behavior soll der Roboter einen dreifärbigen Marker sicher erkennen, sich zum Marker drehen, ihn anfahren, sich in die durch den Marker angegebene Richtung drehen und nun mit dem nächsten erkannten Marker fortfahren. Im zweiten Behavior soll der Roboter einen vierfärbigen Marker (kurz „Stoppmarker“) sicher erkennen, sich zum Stoppmarker drehen, ihn anfahren, sich um 180° drehen und auf den Befehl zum Fortfahren warten. Der Befehl zum Fortfahren ist der Druck auf einen Button der S4R-Framework Benutzeroberfläche. Ausserdem verändert das zweite Behavior dynamisch das erste Behavior. Die Richtungsaussage der Marker wird dabei um 180° gedreht, wodurch Behavior eins einer Reihe von Markern dann in umgekehrter Richtung folgen wird. Beide Behaviors sind immer aktiv, das zweite unterdrückt das erste, wenn beide agieren wollen. Dieses Verhalten ist an die Subsumption architecture (Kapitel 2.5.1) angelehnt.

Zur weiteren Vereinfachung der Bedienung soll das Auflegen der Marker vereinfacht werden. Im Gegensatz zu den vorherigen Beispielen sollen die Marker nicht mehr in geeigneten Abständen liegen müssen, so dass immer nur ein Marker im Kamerabild zu sehen ist. Multiple Marker im Bild sollen richtig erkannt werden, ohne sich gegenseitig zu stören. Der dem Roboter in Fahrtrichtung am nächsten liegende Marker soll als nächstes Ziel angefahren werden.

Das Behavior kann durch ein FSA Diagramm mit folgendem Quadrupel beschrieben werden:

$$M = \{\{Start, Suche, Fahrt, Servieren\}, \delta, Start, \{\}\} \tag{5.2}$$

Die Liste der Übergangsaktionen δ ist in Tabelle 5.2 abgebildet und die grafische Darstellung des FSA Diagramms findet sich in Abbildung 5.9. Es gibt in dem Behavior keinen definierten Endzustand, daher ist die entsprechende Liste im beschreibenden Quadrupel leer.

δ	q	*input*	$\delta(q,input)$
	Start	alle	Suche
	Suche	Marker gefunden	Fahrt
	Suche	Stoppmarker gefunden	Servieren
	Suche	Marker nicht gefunden	Suche
	Fahrt	Fahrt beendet	Suche
	Fahrt	Fahrt nicht beendet	Fahrt
	Servieren	Befehl Fortfahren	Suche
	Servieren	Servieren nicht beendet	Servieren

Tabelle 5.2: Liste der Übergangsaktionen δ zum Multi Color Behavior.

5.5.3 Die Komponenten

- Die DFK 21F04 Farb-Kamera wurde mit einem 4 mm Objektiv verwendet und mit einer Neigung von 62° zur Horizontalen montiert. Sie sieht damit eine Fläche von etwa 1,4 m vor dem Roboter inklusive der Kalibriertafel am Bug des Roboters. Der Testaufbau ist in Abbildung 5.10 dargestellt. Die einseitige Montage am Stativ ist durch die Brennweite weitgehend kompensiert, der Roboter sieht links nur etwa 3 cm mehr als rechts.

- Die verwendete Kalibriertafel hat Platz für fünf Farben. Eine Besonderheit der Kalibriertafel ist ein Knick, der die Kalibrierfarben in zwei unterschiedlichen Beleuchtungen zeigt (Abbildung 5.10).

- Die Marker bestehen aus drei bzw. vier Sechsecken mit etwa 3 cm Durchmesser in den Farben rot, blau, grün und gelb (Abbildung 5.6). Die einzelnen Farben sind im Bild als Farbbereiche mit einer Fläche von 250 Pixeln sichtbar. Nach zahlreichen praktischen Tests wurde die zuvor verwendete rote Farbe gegen einen helleren Farbton mit weniger Blauanteil ausgetauscht.

- Der Farb-Recogniser ContinousRecognise2D kam zum Einsatz. Dieser Bildverarbeitungsalgorithmus wurde von DI Wolfgang Ponweiser speziell für den Einsatz am mobilen Roboter geschrieben. Der Algorithmus hat eine Reihe von Eigenschaften, durch die die Erkennung der farbigen Marker am Boden erleichtert wird:

  - Es reicht eine Instanz des Recognisers, um mehrere Farben im Bild zu suchen. Das spart Rechnerleistung gegenüber multiplen Instanzen eines Farb-Recognisers, der jeweils nur eine Farbe erkennt.
  - Die gesuchten Farben müssen nicht vorher angelernt werden. Der Farb-Recognisers arbeitet mit einer dauerhaft im Kamerabild sichtbaren Kalibriertafel, die am Bug des Roboters angebracht ist.
  - Der Farb-Recogniser liefert nach jeder Bildauswertung eine Liste von gefundenen Farbbereichen für jede der gesuchten Farben. Es wird keine weitere Bewertung in der Bildverarbeitung durchgeführt.
  - Jede Farbe wird zweimal im Bild gesucht: Einmal mit ihrem RGB Wert entsprechend der Kalibriertafel und ein weiteres Mal mit einem helleren Farbton. Damit können Marker, die durch Lichtreflexionen von Beleuchtungskörpern (Highlights) heller erscheinen, erkannt werden.

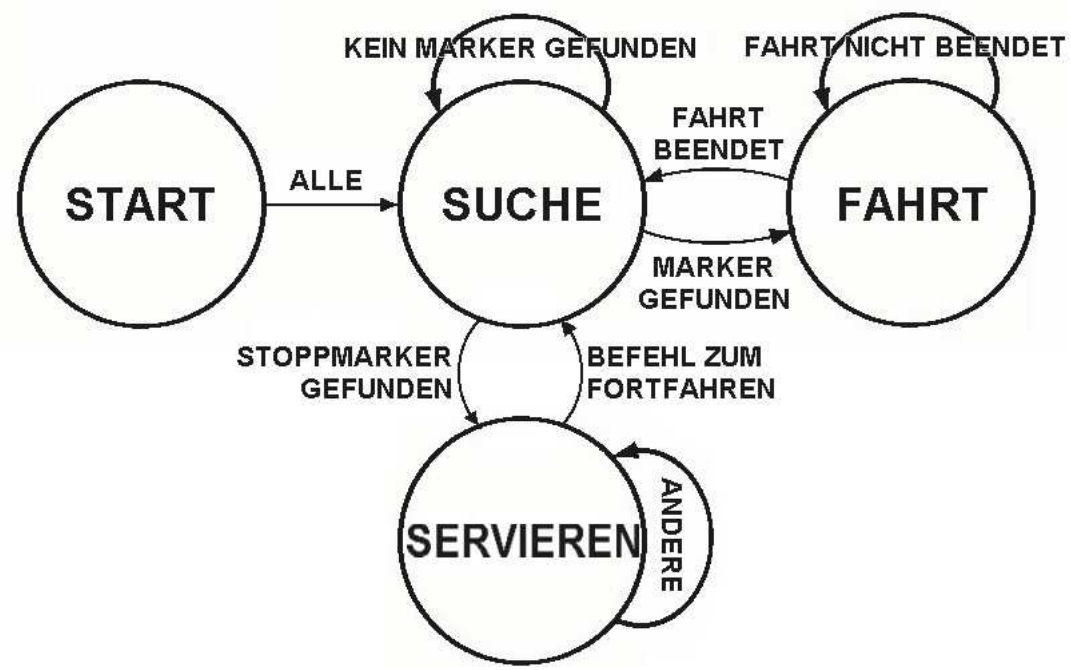

Abbildung 5.9: FSA Diagramm des Multi Color Behaviors. Ausgehend vom Zustand Suche wird je nach gefundenem Marker Fahrt oder Servieren aktiv.

  - Wird eine Farbe überhaupt nicht im Bild gefunden, so erweitert der Recogniser dynamisch die erlaubte Abweichung der gesuchten Farbe. Die Abweichung vergrößert sich weiter, bis Farbbereiche der gesuchten Farbe im Bild gefunden werden. Werden zuviele Farbbereiche gefunden wird die Abweichung wieder reduziert, um ein plausibles Ergebnis zu erzielen. Mit dieser Dynamik können Marker auch in schwierigen Lichtsituationen noch erkannt werden.

- Die Steuerungsmethode *RobotDiscreteMotion* wurden verwendet.

5.5.4 Die Analyse

Das Multi Color Behavior konnte seine Aufgabenstellung sehr gut erfüllen. Das Behavior zeigte beim wiederholten Einsatz ein sehr stabiles Verhalten. Die Bedienung ist einfach und auch im Lehrbetrieb problemlos einsetzbar. Die Momentaufnahme in Abbildung 5.11 zeigt mehrere erkannte Marker im Bild, die gerade verifiziert werden, um die sichere Erkennung zu gewährleisten.

- Der speziell für die Anwendung mit dem Roboter adaptierte Farb-Recogniser konnte im Zusammenspiel mit dem Behavior seine Stärken voll ausspielen. Die Marker konnten unter realen Bedingungen mit zahlreichen Störungen immer noch stabil erkannt werden. Künstliche Beleuchtung, reflektiertes Sonnenlicht, sich ändernde Schatten, Highlights von Beleuchtungskörpern und zahlreiche bewegte Störobjekte im Bild konnten dem robusten Verhalten nichts anhaben.

- Das Behavior wurde ausgiebig getestet (Abbildung 5.12). In 13 dokumentierten Testfahrten mit variierenden Strecken und Lichtsituationen wurden 654 Markererkennungen

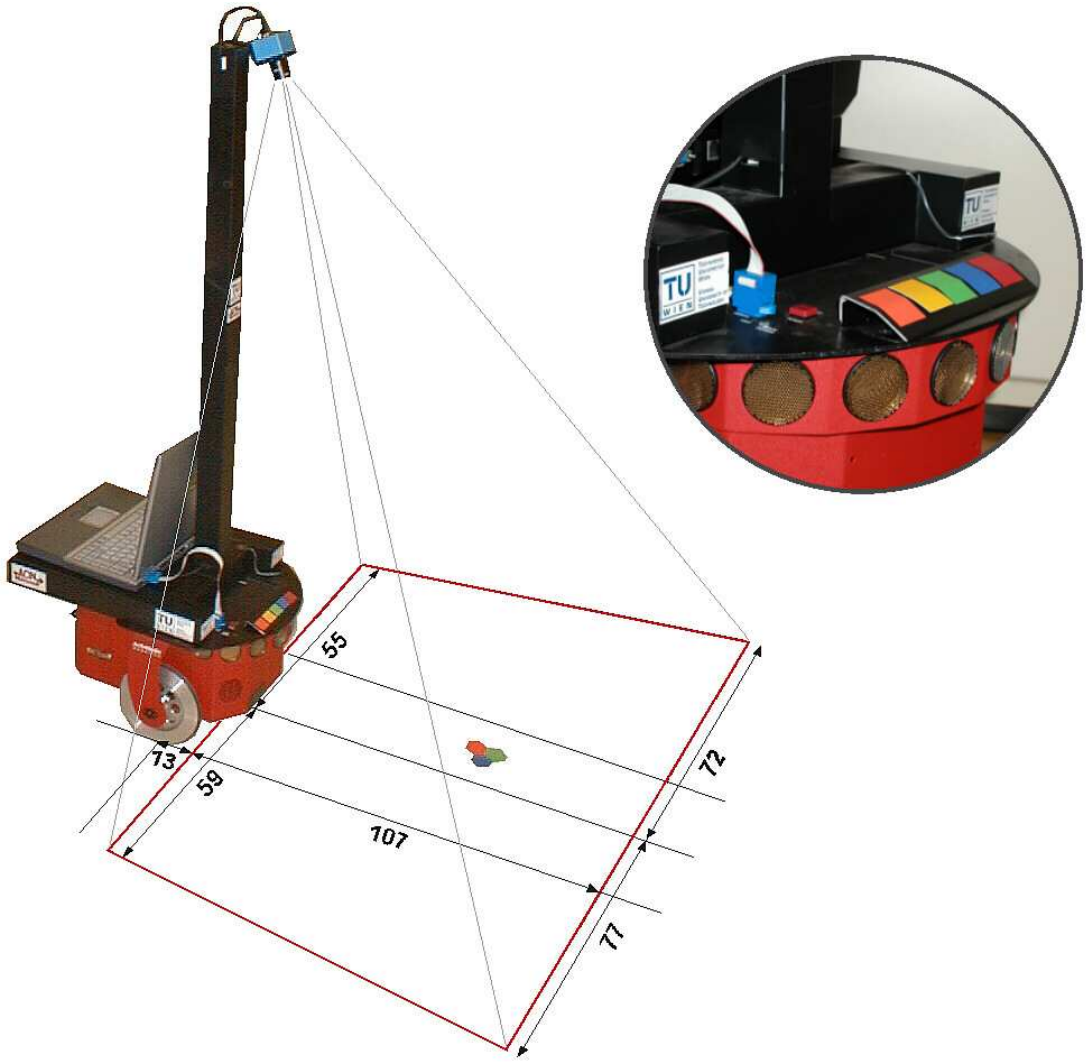

Abbildung 5.10: Der Pioneer 2-DX Roboter mit Farbkamera, Kalibriertafel und Laptop. Die betrachtete Bodenfläche ist wiederum schematisch dargestellt. Die Fläche ist größer als im vorherigen Aufbau mit dem 6 mm Objektiv. Die betrachtete Fläche hat aber auch eine deutlich stärker verzerrte Trapezform. Alle Zahlen sind in cm angegeben.

gezählt. 634 Marker wurden dabei richtig erkannt, das entspricht einer Erkennungsrate von 96,9%.

- Die etwas eigenwillige Ausrichtung der Farbkamera auf den Bereich vor dem Roboter inklusive des Bugs des Roboters auf dem die Kalibriertafel montiert ist, hat keine Nachteile mit sich gebracht. Bei der Bildbearbeitung wurden die untersten 25 Zeilen des Bildes nur für die Kalibrierung verwendet und bei der Suche nach den verschiedenen Farben ausgeblendet. Dieser Verlust an Auflösung und betrachteter Bodenfläche wurde durch die live-Kalibrierung aber mehr als kompensiert.

- Die Kombination der beiden Behaviors lieferte exakt das erwartete Verhalten. Obwohl die beiden Behaviors gegeneinander arbeiten und sich sogar gegenseitig verändern, erhält man ein stabiles System.

Abbildung 5.11: Momentaufnahme der S4R Benutzeroberfläche beim Multi Color Behavior. Vor dem Roboter liegen drei Marker am Boden. Man sieht starke Lichtreflexionen (Highlights) der Raumbeleuchtung und Verschmutzung am gemusterten Bodenbelag. Am unteren Bildrand ist der Bug des Roboters mit der Farbtafel zu sehen. Die kleinen farbigen Quadrate markieren die gefundenen Farben. Alle drei Marker im Bild werden als Marker erkannt, obwohl in keinem Fall alle drei Farben erkannt wurden. Die gelben Quadrate markieren zur Ergänzung berechnete Stützpunkte. Als nächstes wird der Marker in der Mitte angefahren, der rote Strich markiert die durch den Marker angegebene weitere Richtung.

- Der nach dem Serviervorgang notwendige Klick auf einen Button der S4R-Framework Benutzeroberfläche ist für einen Benutzer nicht optimal, da man auf dem 30 cm über dem Boden stehenden Laptop mit dem Mauscursor einen Button treffen muss. Eine praxistauglichere Schnittstelle zum Steuern wäre hilfreich.

- Umgekehrt ist die Schnittstelle vom Behavior zum Benutzer auch noch nicht optimal. Als Entwickler bekommt man ein gutes visuelles Feedback in Form des dargestellten Kamerabildes mit in der Grafik ergänzten Zusatzinformationen (Rechtecke und Linien, die Farbbereiche markieren). Mit dieser Information kann man sehr leicht einen Eindruck über die Vorgänge innerhalb der Steuererung bekommen und die Information auch weitervermitteln, was ja eines der Ziele der Demonstrationsbeispiele darstellt. Für einen darüber hinaus gehenden Einsatz wird es aber auch nötig sein, dass der mobile Roboter auf einer höheren Ebene mit dem Benutzer kommunizieren kann. Ein Benutzer wird sich nicht für die Navigation interessieren, sondern wissen wollen, ob die eigentliche Aufgabe - z.B. das Servieren - schon erledigt ist.

Abbildung 5.12: Aufnahmen einer typischen Testfahrt durch das ACIN RobSense Labor. Die Kartonschachteln simulieren zusätzliche Hindernisse. Bei verschiedenen Testläufen wurde mit Tageslicht, künstlichem Licht und gemischten Lichtsituationen getestet.

5.5.5 Der praktische Einsatz

Das Demonstrationsbeispiel Multi Color Behavior kam bei verschiedenen Anlässen zum praktischen Einsatz:

- YO!Einstein Pressenkonferenz am 21.06.2005: Butler Jamer und Dr. Vincze servieren Dr. Taschner (Leiter des math.space), Dr. Windischberger (Techwoman of the Year 2004) und Dr. Kaiser (Vizerektor der Technischen Universität Wien) am Ende der Pressekonferenz erfolgreich den Kaffee (Abbildung 5.13, linkes Bild). [YOE05]
- YO!Einstein Event am 23.06.2005: James demonstriert sein Behavior vor mehreren hundert Besuchern des YO!Einstein Schülertages (Abbildung 5.13, rechtes Bild). Das Behavior läuft dabei mehrere Stunden unter erschwerten Bedingungen: Der Roboter hat keine freie Arbeitsfläche, sondern navigiert zwischen den Besuchern. Die Beleuchtung ist unregelmäßig, der Boden reflektiert und die Besucher marschieren durch das Kamerabild und erzeugen zusätzlich bewegte Schatten. Trotzdem werden nur wenige falsche Fahrbewegungen registriert.
- Die Laborübungen Systemtechnik und Steuerungstechnik im Herbst 2005 und Frühling 2006: Das Multi Color Behavior wird von 80 Studenten in 16 Laborgruppen im Rahmen der Laborübung verwendet und dabei ausgiebig analysiert und getestet.
- YO!Einstein Event am 22.06.2006: Abermals führt James seine Behaviors vor mehreren hundert Schülern vor.

5.6 Zusammenfassung

Die in diesem Kapitel vorgestellten Demonstrationsbeispiele illustrieren die Bemühungen eine Kamera als Sensor für eine behavior-basierte Steuerung zu verwenden. Ausgangspunkt ist die Kombination eines fertigen Bildverarbeitsungsalgorithmus mit einer Robotersteuerung im Follow Ball Behavior. In den weiteren Beispielen wird das Ergebnis der Bildverarbeitung immer weiter an die Bedürfnisse der reaktiven Steuerung angepasst. Im abschließenden Multi Color

Abbildung 5.13: YO!Einstein Tag im Juni 2005. Links die Pressekonferenz. Dr. Vincze unterstützt James beim Servieren des Kaffees. Rechts der Besucherandrang beim YO!Einstein Tag. Die Schüler experimentierten mit den Markern und beobachteten das reaktive Navigationsverhalten von James.

Behavior wird schließlich eine speziell für den Einsatz mit dem Roboter konzipierte Bildverarbeitung eingesetzt. Das Ergebnis ist eine erfolgreiche Implementierung der Aufgabenstellung. Die Erfahrungen die bei den Demonstrationsbeispielen gewonnen wurden fließen in die Bewertung in Kapitel 6 ein.

Kapitel 6

Bewertung

Im diesem Kapitel werden die Ergebnisse der Analyse der Thematik Robot Vision und die Erfahrungen der praktischen Demonstrationsbeispiele (Kapitel 5) kombiniert. Der erste Teil ist eine Bewertung der Möglichkeiten, um Computer Vision als Sensor in der mobilen Robotik einzusetzen. Die Verwendung einer behavior-basierten Steuerung bietet dazu ganz eigene Möglichkeiten. Der zweite Teil des Kapitels widmet sich der Bewertung der vorhandenen Arbeitsumgebung. Wie gut eignet sich das S4R-Framework mit seinen Computer Vision Services und Mobile Robot Services als Basis für eine behavior-basierte Steuerung? Die Bewertungen sind durch Vorschläge und Beispiele ergänzt, die eine direkte praktische Umsetzung von Verbesserungen ermöglichen sollen.

6.1 Computer Vision für die mobile Robotik

Die Robotik hat langjährige Erfahrung mit Machine Vision im industriellen Bereich. Diese Erfahrungen lassen sich aber nur teilweise für eine robuste Computer Vision für mobile Robotik einsetzen. Mit einer Situationsbewertung, den Demonstrationsbeispielen und einer Reihe von Tests wurden verschiedene Komponenten und Kenngrößen für eine Robot Vision untersucht.

6.1.1 Kamera und Objektiv

Digitale Kameras sind als Sensoren für die mobile Robotik geeignet. Die folgenden Eigenschaften konnten dabei festgestellt werden:

- **Kameras sind günstige Sensoren**
  - Eine digitale Kamera als Teil einer Computer Vision Anwendung ist ein günstiger Sensor. Die verschiedenen Sensoren für mobile Robotik haben Preise im zwei- bis fünfstelligen Eurobereich. Digitale Kameras mit Optik sind preislich im dreistelligen Bereich zu finden.
  - Im Preis nicht enthalten ist die notwendige Software für die Computer Vision Anwendung. Für die Entwicklung einer passenden Bildverarbeitung für einen mobilen Roboter fallen bei einem kommerziellen Produkt nur einmalige Entwicklungskosten an.

- **Kameras sind robuste Sensoren**
  - Trotz der notwendigen Optik sind digitale Kameras robuste Sensoren, die ohne Mechanik auskommen. Durch den schon beinahe inflationären Einsatz von digitalen Kameras in mobilen Geräten (Laptops, Mobiltelefone) werden laufend bessere und robustere Kameras entwickelt. Die mobile Robotik kann sich der vorhandenen Technik bedienen und sie in die eigenen Anwendungen integrieren.
  - Beim fahrenden Roboter ist eine Kamera Vibrationen durch den Antrieb und Bodenunebenheiten ausgesetzt. Bei keinem der durchgeführten Beispiele und Tests zeigte sich ein negativer Einfluss dieser Vibrationen auf die behavior-basierte Steuerung.
- **Kameras liefern Daten für Behaviors**
  - Im Gegensatz zur hochgenauen Industrierobotik kann die behavior-basierte Robotik auch mit relativen nichtmetrischen Ortsangaben („das Objekt befindet sich links vor dem Roboter“) arbeiten. In diesem Fall ist es nicht notwendig die genauen Koordinaten eines Gegenstands zu wissen. Nichtmetrische Wertebereiche (z.B. Pixel) können direkt als Eingangsvariable von Behaviors benutzt werden, die auf einer Fuzzy Control beruhen.
  - Für die Berechnung von Positionen, Längen und Flächen im Bild kann das einfache Modell der Lochkamera herangezogen werden. Der durch den Verzicht auf die Korrektur der Objektivverzerrung entstehende Fehler ist für behavior-basierte Anwendungen meist nicht relevant.
  - Auch die bei Weitwinkelobjektiven vorhandenen starken Randverzerrungen stellen für Behaviors kein Hindernis dar. Für die Demonstrationsbeispiele haben sich Objektive im Bereich Normalobjektive bis leichtes Weitwinkelobjektiv als geeignet erwiesen.

6.1.2 Geschwindigkeit

Die Geschwindigkeit eines Bildverarbeitungsalgorithmus ist oft die wichtigste Kenngröße, vor allem wenn vom Roboter Reaktionen in Echtzeit verlangt werden. Die Demonstrationsbeispiele haben gezeigt, dass verschiedene Behaviors auf langsame Sensordaten durch eine langsame Bildverarbeitung unterschiedlich reagieren. Die Behaviors wurden dazu mit der maximal verfügbaren Datenfrequenz der Bildverarbeitung und mit künstlich verlangsamten Datenfrequenzen getestet.

- Das Follow Ball Beispiel auf Basis von kontinuierlichen Fahrbewegungen zeigt mit verlangsamter Datenfrequenz stark verschlechtertes Fahrverhalten. Die Reaktion auf ein schnell bewegtes Ziel wird ruckartig, der Roboter verfolgt scheinbar das falsche Ziel. Sinkt die Datenrate unter zwei verarbeitete Bilder/Sekunde, so kann ein schnell bewegter Ball komplett übersehen werden.
- Bei den Beispielen mit zwei- und dreifärbigen Markern war bereits die maximale Datenfrequenz mit 6 bzw. 3 Bilder/Sekunde deutlich unter den maximal vom Framework unterstützten 10 Bildern/Sekunde. Der Einfluss auf das Fahrverhalten ist deutlich sichtbar, der Roboter bewegt sich langsamer bzw. die Pausen zwischen Bewegungen werden länger. Trotzdem kommt es zu keinen falschen Fahrten.

- Analog verhält sich das Multi Color Behavior. Bei verschiedenen Tests wurde die Datenfrequenz durch das Unterdrücken von einzelnen Bildern der Kamera verlangsamt. Bei einer verlangsamten Datenfrequenz auf 3 Bilder/2 Bilder/1 Bild pro Sekunde sinkt die Geschwindigkeit der gesamten Fahrbewegung im selben Maß. Es kommt zu keinen falschen Fahrten.

Eine Bildverarbeitung in Echtzeit ist das erstrebenswerte Ziel. Solange das aber noch nicht erreicht werden kann, lassen sich aus den Ergebnissen folgende Empfehlungen ableiten:

- Das Fahrverhalten mit der Steuerung an die Qualität der Sensordaten anpassen. Mit sinkender Frequenz der Sensordaten die Fahrgeschwindigkeit reduzieren.
- Diskrete Fahrbewegungen als Alternative zu kontinuierlichen Fahrbewegungen in Betracht ziehen. Wenn der Fahrweg in kurze diskrete Bewegungen zerlegt wird, dann sieht das Ergebnis ähnlich einer kontinuierlichen Bewegung aus. Das Zeitverhalten ist aber weniger kritisch, da zwischen den Bewegungen Pausen gemacht werden können.
- Die Stärken des asynchronen Systems (multithreading) besser ausnutzen, wie z.B. beim VIP Framework (Kapitel 4.2.9.2) . Die verschiedenen Bildverarbeitungsaufgaben priorisieren, damit die wichtigen Aufgaben in minimaler Zeit erledigt werden können, während weniger zeitkritische Ergebnisse längere Verarbeitungszeiten benötigen dürfen.

6.1.3 Datenaufbereitung

Bei einer Objekterkennung findet ein mehrstufiger Datenaufbereitungsprozess statt. Die Bilddaten werden dabei durch einer Reihe von Filteroperationen auf symbolische Daten abgebildet. Im Rahmen der Bildverarbeitung wird dabei eine Menge von Meta-Daten erzeugt (mögliche Fundorte des gesuchten Objekts, Wahrscheinlichkeiten dazu, Größe des gefundenen Objekts etc.), aus denen der eigentliche Output der Bildverarbeitung gebildet wird.

- Vollständige Daten: Eine Bildverarbeitung soll die Ergebnisse nicht nur in komprimierter Form, sondern auch in kompletter Form anbieten. Beispielesweise soll ein Algorithmus, der ein rotes Objekt sucht, nicht nur die Position des Objekts liefern, sondern auch Meta-Daten, die in einer weiteren Verarbeitung verwendet werden können.
- Daten ohne Vorselektion: Die Bildverarbeitung soll keine Vorselektion der gefundenen Objekte durchführen. Ein Ergebnis der Art „wahrscheinlichster Ort des Objekts im Bildbereich“ lässt sich nur schwer weiterverwenden. Es ist zielführender die Bildverarbeitung auf ihre Hauptaufgabe zu beschränken. Hauptziel ist das Umwandeln der umfangreichen Bilddaten in symbolische Daten wie etwa Position, Größe und Wahrscheinlichkeit von gesuchten Objekten. Dieses Ergebnis, eine Menge von Datensätzen möglicher Objekte im betrachteten Bild, soll als Output weitergegeben werden.
- Recogniser vs. Tracker: Bei den Bildverarbeitungsalgorithmen zur Objekterkennung unterscheidet man zwischen Trackern und Recognisern. Zum erstmaligen Auffinden eines Objekts im Bild verwenden beide eine Suchoperation über das gesamte Bild. In Folge verwendet ein Tracker zum Auffinden des Objekts zusätzliche Informationen, die bei der Verarbeitung der vorherigen Bildern gewonnen wurden. Wenn ein Objekt dauernd im Bild zu finden ist und sich Bewegungsrichtung und Geschwindigkeit ermitteln lassen, dann ist ein Tracker besonders effizient.

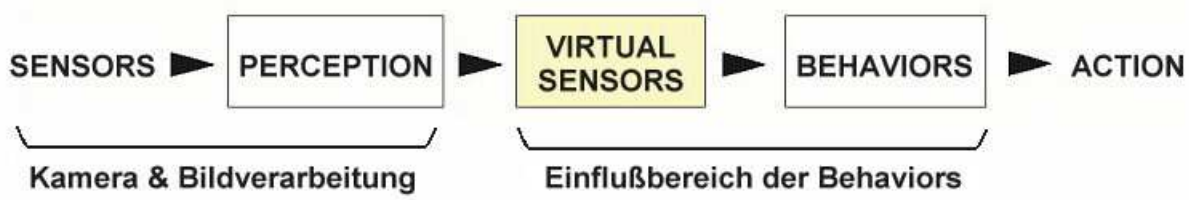

Abbildung 6.1: Das erweiterte Reiz-Reaktionsdiagramm mit dem neuen Zwischenschritt *Virtual Sensor*. Die Kamera und die Bildverarbeitung bilden die außerhalb des Einflusses des Behaviors liegende Sensoreinheit. Die virtuellen Sensoren bilden aus den Sensordaten spezifische Daten. Die virtuellen Sensoren liegen innerhalb des Einflussbereichs der Behaviors.

Abbildung 6.2: Eine Beispielanwendung für das erweiterte Reiz-Reaktionsdiagramm.

6.1.4 Erweitertes Reiz-Reaktionsdiagramm

Das Reiz-Reaktionsdiagramm für Behaviors nach Arkin (Abbildung 4.6) beachtet den Bildverarbeitungsschritt als eigenes Element. Diese Darstellung entspricht der Verwendung eines abgeschlossenen Bildverarbeitungsalgorithmus in einer behavior-basierten Steuerung. Verwendet man eine für die mobile Robotik optimierte Bildverarbeitung, so spiegelt die Darstellung die reale Struktur nur noch teilweise wieder. Durch die verbreiterte Output-Schnittstelle, die mehr Daten ohne Vorselektion überträgt, ist es möglich eine zusätzliche Verarbeitungsschicht zwischen Bildverarbeitung und Behaviors zu setzen. Diese Überlegung führt zum in Abbildung 6.1 dargestellten erweiterten Reiz-Reaktionsdiagramm.

Die neue Schicht verhält sich wie ein virtueller Sensor für einen spezifischen Reiz. Input für diesen virtuellen Sensor ist der Output der Bildverarbeitungen und der Output anderer Sensoren am Roboter. Durch Kombination der Informationen kann der virtuelle Sensor eine Aussage treffen. Output des virtuellen Sensors wäre die Meldung des spezifischen Reizes für ein Behavior. Eine weitere Eigenschaft von virtuellen Sensoren ist die Beeinflussbarkeit durch Behaviors. Kamera und Bildverarbeitung liegen dagegen außerhalb des Einflussbereichs der Behaviors.

Zur Illustration ein Beispiel (Abbildung 6.2) für eine Anwendung des erweiterten Reiz-Reaktionsdiagramms: Ein mobiler Roboter soll ein Spielzeug finden und mit einer Roboterhand hochheben. Der Roboter verfügt über eine Stereokamera als Sensor. Die Kamerabilder werden von zwei Bildverarbeitungen bearbeitet. Ein Histogramm-Recogniser kann das Spielzeug anhand seines Farb-Histogramms im Bild erkennen und liefert die Position und Größe im Bild. Ein aus beiden Kamerabildern errechnetes 3D-Tiefenbild erlaubt eine Entfernungsangabe zu jeder Koordinate im Bild. Beide Bildverarbeitungen liefern in der Beispielumgebung für sich alleine korrekte Ergebnisse.

Im virtuellen „Spielzeug“-Sensor werden die Informationen der beiden Bildverarbeitungen kombiniert und auf Plausibilität überprüft. Passen die Größe und Entfernung des Spielzeugs nicht

Abbildung 6.3: Drei Marker und 15 farbige Bälle in einem Test zum Demonstrationsbeispiel Multi Color Behavior. In der Mitte unten sieht man den Bug des Roboters mit der Farb-Referenztafel. Die farbigen Quadrate markieren die verschiedenen vom Farb-Recogniser gefundenen Farbflächen im Bild. Alle Markierungen beziehen sich auf das aktuelle Bild. Die Marker wurden in einem zweiten Bildverarbeitungsschritt (dem virtuellen Sensor) aus den vom Farb-Recogniser gefundenen Positionen berechnet. Die drei weißen Dreiecke zeigen die Position der drei im Bild platzierten Marker.

zusammen, so handelt es ich um eine falsche Erkennung (false positive). Output des virtuellen Sensors ist der spezifische Reiz Spielzeug-erkannt, der die Richtung und Entfernung zum verifizierten Spielzeug liefert. Behaviors zur Navigation des Roboters können mit dem Reiz Spielzeug-erkannt arbeiten und zum Spielzeug hinfahren.

Die Behaviors wirken auch auf den virtuellen Sensor ein, um sein Verhalten zu ändern. Das Behavior, das den Greifer auslöst, meldet seine eigene Aktivierung dem virtuellen Sensor, um damit eine falsche Objektverifizierung auf Grund der im Bild agierenden Roboterhand zu verhindern.

6.1.5 Beispiel Farb-Recogniser mit Behavior

Der im Rahmen der Demonstrationsbeispiele (Kapitel 5) entstandene Farb-Recogniser wurde bereits im Hinblick auf die Anwendung mit einem mobilen Roboter entwickelt. Die bisher gemachten Vorschläge wurden dabei beachtet. Der Farb-Recogniser kann Objekte anhand von Farbflächen erkennen. In einem Schritt werden bis zu fünf Farben gesucht. Im Beispiel in Abbildung 6.3 sind das die vier Farben Rot, Gelb, Grün und Blau, die von der Referenztafel an

Abbildung 6.4: Das erweiterte Reiz-Reaktionsdiagramm zum Multi Color Behavior Demonstrationsbeispiel.

einer konfigurierten Bildposition (unterer Bildrand, Mitte) eingelesen werden. Die Menge aller entsprechenden im Bild gefundenen Farbflächen wird komplett ohne Vorselektion als Output weitergegeben.

Der Farb-Recogniser wurde beim Multi Color Demonstrationsbeispiel (Kapitel 5.5) verwendet, in dem drei- bzw. vierfärbige Marker gesucht werden. Dazu werden die gefundenen Positionen der einzelnen Farben in einem zweiten Bildverarbeitungsschritt weiterverarbeitet. Im erweiterten Reiz-Reaktionsdiagramm entspricht das einem virtuellen Sensor, der den Reiz *Marker-gefunden* liefert. Im erweiterten Reiz-Reaktionsdiagramm (Abbildung 6.4) des Multi Color Behavior finden sich die folgenden Elemente:

1. **Die Kamera** Liefert als Output Bilddaten.
2. **Farb-Recogniser** Erkennung der gesuchten Farben im Bild. Allgemein, nicht Marker spezifisch. Liefert als Output die Position von gefundenen Farbflächen.
3. **Marker Erkennung** Der virtuelle Sensor, der aus den allgemeinen Daten des Farb-Recognisers zusammen mit Marker-spezifischen Wissen (maximaler Abstand der Farben, Reihung der Farben) die Marker im Bild erkennt. Liefert als Output die Positionen von gefundenen Markern.
4. **Behaviors** Dreifärbige Marker markieren den Weg, dem gefolgt werden soll. Vierfärbige Marker markieren einen Endpunkt, an dem der Roboter stehenbleibt und sich umdreht. Liefert als Output die Bewegungsbefehle an den Roboter.
5. **Der Roboter**

Das Multi Color Behavior wurde ausgiebig getestet und zeigt ein stabiles Fahrverhalten. Von 654 Markern wurden 634 richtig erkannt und angefahren. Die konkreten Zahlen aufgeschlüsselt nach Fehlerarten sind der Tabelle 6.1 zu entnehmen.

	Fehlerart	Anzahl	%
Marker richtig erkannt:	true positive	634	96,9%
Marker übersehen:	false negative	6	0,9%
Falscher Marker erkannt:	false positive	14	2,1%
Marker gesamt:		654	100,0%

Tabelle 6.1: Die Ergebnisse von 13 Testfahrten mit dem Multi Color Behavior.

6.2 Das S4R-Framework

Im folgenden Abschnitt werden die Vor- und Nachteile des S4R-Frameworks als Softwarebasis für eine behavior-basierte Robotersteuerung mit Computer Vision erörtert.

6.2.1 Framework Struktur

Ein Framework ist ein objektorientiertes, wiederverwendbares Design für ein Softwaresystem. Einem Entwickler wird dazu eine Grundstruktur in Form von abstrakten oder konkreten Klassen zur Verfügung gestellt. Das Framework bildet eine generelle, umfangreiche Rahmenstruktur und bestimmt damit die Softwarearchitektur. Die Verwendung eines Frameworks bringt eine Reihe von algemeinen Vorteilen:

- Softwareteile werden modular gekapselt und können wiederverwendet werden, so wie es bei Libraries möglich ist.
- Die objektorientierte Architektur fördert die Wiederverwendung und Adaptierung des gewählten Softwaredesigns.
- Die Struktur eines Frameworks unterstützt die Zusammenarbeit von mehreren Entwicklern. Ein Framework ist aber kein Masterplan der von einem Entwickler implementiert wird, sondern eine stetig wachsende Sammlung von Modulen.

Eine Framework Struktur wird auch beim Video Image Processing (VIP) Framework [UKMK06] verwendet (Kapitel 4.2.9.2). Das VIP Framework ist speziell für eine Robotersteuerung in dynamische Umgebungen entwickelt worden. Die mit dem VIP Framework erzeugte Applikation eines Steuerungssystems für Roboterfußball zeigt außerdem das Potential, dass in einer Frameworkstruktur steckt.

Die allgemeine und offene Struktur eines Frameworks kann sich auch nachteilig auswirken:

- Eine offene und allgemein gehaltene Umgebung kann nie die Leistung einer spezialisierten Anwendung bieten. Bei Computer Vision bedeutet das eine längere Verarbeitungszeit der Bildverarbeitung gegenüber der optimierten Geschwindigkeit eines Stand-Alone Algorithmus.
- Beim Entwickeln eines neuen Computer Vision Algorithmus hat dessen Anwendung in der mobilen Robotik meist geringe Bedeutung. Die Implementierung ins Framework ist im Vergleich zu eine Stand-Alone Entwicklung mit einem höheren Aufwand verbunden. Dem Entwickler soll trotzdem ein Anreiz geboten werden, seine Software ins Framework einzubringen. Ein solcher Anreiz wäre das Angebot von einfach anpassbaren Demonstrationsbeispielen, mit denen die Funktion des neuen Computer Vision Algorithmus sofort praktisch ausprobiert werden kann. Ein Beispiel dazu ist in Anhang C zu finden.
- Der Vorteil, das ein Framework die kommunikative Zusammenarbeit zwischen mehreren Entwicklern erlaubt, bringt auch immer den möglichen Nachteil von divergierenden Interessen. Ohne Fokus auf die gewünschte Zielanwendung einer Robotersteuerung wird keine - im Rahmen des Frameworks - optimale Lösung zustande kommen. Organisatorische Probleme dieser Art lassen sich durch einen lead programmer oder Software Manager lösen.

6.2.2 Framework Services

Eine Besonderheit des S4R-Frameworks stellen die Services dar. Analog den Prinzip der gelben Seiten werden Dienste im System angeboten. Ein solches Service kann angeboten (*Service Provider*) oder in Anspruch genommen werden (*Service Requester*). In der Softwarestruktur besteht ein Service aus einer oder mehreren Klassen, die einen spezifischen Dienst über eine Schnittstelle zur Verfügung stellen.

- Das Konzept der gelben Seiten lautet das ein Service Requester anhand seiner Anforderungen ein passendes Service zugeteilt bekommt. Dieses Konzept überzeugt erst, wenn es mehr als ein auf die Anforderungen passendes Service im Framework gibt. Andernfalls ergibt sich dasselbe Verhalten wie bei einer Library-Struktur, allerdings mit höherem Overhead. Solange die Anzahl von Services im Framework klein ist, wird das Potential des Frameworks nicht genutzt. Die Situation verbessert sich jedoch mit jedem neuen Service.

- Eine Funktionalität, beispielsweise ein Bildverarbeitungsalgorithmus, sollte sich nicht nur in Form eines Service anbieten. Die Bildverarbeitung kann sich über multiple Schnittstellen, in Form von mehreren Services, anbieten. Der Unterschied zwischen diesen Schnittstellen kann beispielsweise in der Qualität des Services oder umgekehrt in der Geschwindigkeit der Verarbeitung liegen. Aus der Sicht des Service Requesters wäre das Angebot vergrößert, wodurch das Service-Konzept besser realisiert wird.

6.2.3 Mobile Robot Services

Eine besondere Gruppe von Services sind die Mobile Robot Services, die die Schnittstelle zum Roboter abdecken. Die angebotenen Services dienen sowohl zur Steuerung des Roboters mittels diskreten und kontinuierlichen Bewegungsbefehlen als auch zur Abfrage der Roboterodometrie und der Sonarsensoren des Roboters.

- Erweiterung zur Unterstützung unterschiedlicher Protokolle zur Kommunikation mit dem Roboter. Derzeit wird nur das AROS Protokoll für den Pioneer 2-DX Roboter unterstützt. Durch die Implementierung einer weiteren Klasse kann aber jederzeit ein weiteres Protokoll unterstützt werden.

- Erweiterung der Services zur Unterstützung verschiedener physikalischer Schnittstellen zum Roboter. Laut [Ein05] sind als Schnittstellen RS-232 (seriell), USB und Ethernet vorgesehen, aber nur die serielle Schnittstelle wurde implementiert. Die Konfiguration zur Verwendung eines Seriell/USB-Adapters ist im Anhang B dokumentiert.

- Die beiden Bewegungsarten zur diskreten und kontinuierlichen Bewegung des Roboters sind korrekt, aber sehr restriktiv implementiert. Das RobotDiscretMotion Service könnte analog zum RobotContinuousMotion Service um die Möglichkeit erweitert werden, zwischen den Fahrbefehlen die Geschwindigkeit zu ändern. Derzeit kann die Fahrgeschwindigkeit nur über die Konfigurationsdatei eingestellt werden, d.h. Geschwindigkeit ist während einer Fahrt fix.

- Definition einer neue Bewegungsart, die eine diskrete Drehbewegung und eine diskrete Fahrtbewegung gleichzeitig erlaubt. Um den Endpunkt einer solchen Bewegung eindeutig

vorzugeben, muss zusätzlich die zeitliche Koordination der Dreh- und Fahrbewegungen definiert sein. Ein Beispiel für diese Befehle: Fahrbewegung: Fahre ab Zeitpunkt 0,0 eine Strecke von 2,0 m mit der Geschwindigkeit 0,5 m/s. Drehbewegung: Drehe ab Zeitpunkt 1,0 einen Winkel von 30° mit der Geschwindigkeit 30°/s. Das Ergebnis ist eine kombinierte Dreh und Fahrbewegung, die die Vorteile der diskreten Bewegungen (absehbare Endposition, selbstständige Terminisierung) mit den Vorteilen der kontinuierlichen Bewegungen (natürlichere Bewegung) kombiniert.

6.2.4 S4R Benutzeroberfläche

S4RGUI, die graphische Benutzeroberfläche des S4R-Frameworks, ist die Schnittstelle zwischen dem Benutzer und dem Roboter mit seiner Steuerung. Der Roboter selbst verfügt nur über sehr eingeschränkte Möglichkeiten Informationen darzustellen. Eine kleine LCD Anzeige kann beim Einschalten des Roboters benutzt werden, um den Batteriestatus abzulesen oder den Selbsttest zu starten. Alle für den Betrieb notwendigen Ein- und Ausgaben müssen daher über die Benutzeroberfläche am Laptop gemacht werden.

Die S4R Benutzeroberfläche ist einfach und für den Test- und Demonstrationsbetrieb gedacht. Kommerzielle Qualität oder eine besondere Optik sind keine Ansprüche denen gerecht werden muss. Vielmehr sollen in einer einfachen und klaren Oberfläche alle notwendigen Element geboten werden. Die Bedienung ist auf eine kleine Benutzergruppe im Institut beschränkt und nicht für einen ungeschulten Benutzer gedacht. Die folgenden Vorschläge zur Erweiterung haben das Ziel einer besseren Benutzeroberfläche für die mobile Robotik:

2D Kartographie Zur Zeit bietet das S4R-Framework zwei Arten von „Kameras“:

- Video als Live-Stream von einer oder mehreren Kameras oder aus Videodateien.
- Darstellung von 3D Bilddaten einer Stereokamera oder aus einer CAD-Datei.

Speziell in der mobilen Robotik wird oft eine lokale 2D Kartographie durchgeführt. Beim häufig verwendeten *occupancy grid* wird die Umgebung in kleine Flächen unterteilt, die anhand der Sensormessungen als besetzt oder frei markiert werden. Das Ergebnis ist eine Karte der Umgebung. Beim *Simultaneous Localisation and Mapping* (*SLAM*) Verfahren wird die lokal ermittelte Kartographie kontinuierlich mit einer früher erzeugten Karte der Umgebung abgeglichen. Für solche Anwendungen wäre eine dritte Art von Kameradarstellung, speziell für die 2D Kartographie, wünschenswert. Die Benutzeroberfläche sollte um eine solche 2D Mapping View erweitert werden.

Maus und Scratchpad Als untaugliche Eingabeinstrumente haben sich Maus und Scratchpad herausgestellt. Für die Verwendung einer Maus fehlt es am Roboter an Platz. Eine Verwendung des eigentlich für die Handballenauflage vorgesehen Bereich des Laptops ist zwar möglich, aber die Maus geht durch die Fliehkräfte bei Drehbewegungen des Roboters leicht verloren. Das am Laptop vorhandene Scratchpad ist als mobiler Ersatz für die Maus verwendbar. Es ist aber sehr unpraktisch hinter dem fahrenden Roboter herzulaufen und dabei gleichzeitig den Cursor über die Benutzeroberfläche zu navigieren. Mangels sinnvoller Eingabemöglichkeiten sollte die Benutzeroberfläche ohne Cursor benutzbar sein.

Tastenkürzel und Hotkeys Tastenkürzel für alle aktiven Eingabeelemente der Benutzeroberfläche würden eine einfache Bedienbarkeit erlauben. Neben den bereits vorhandenen

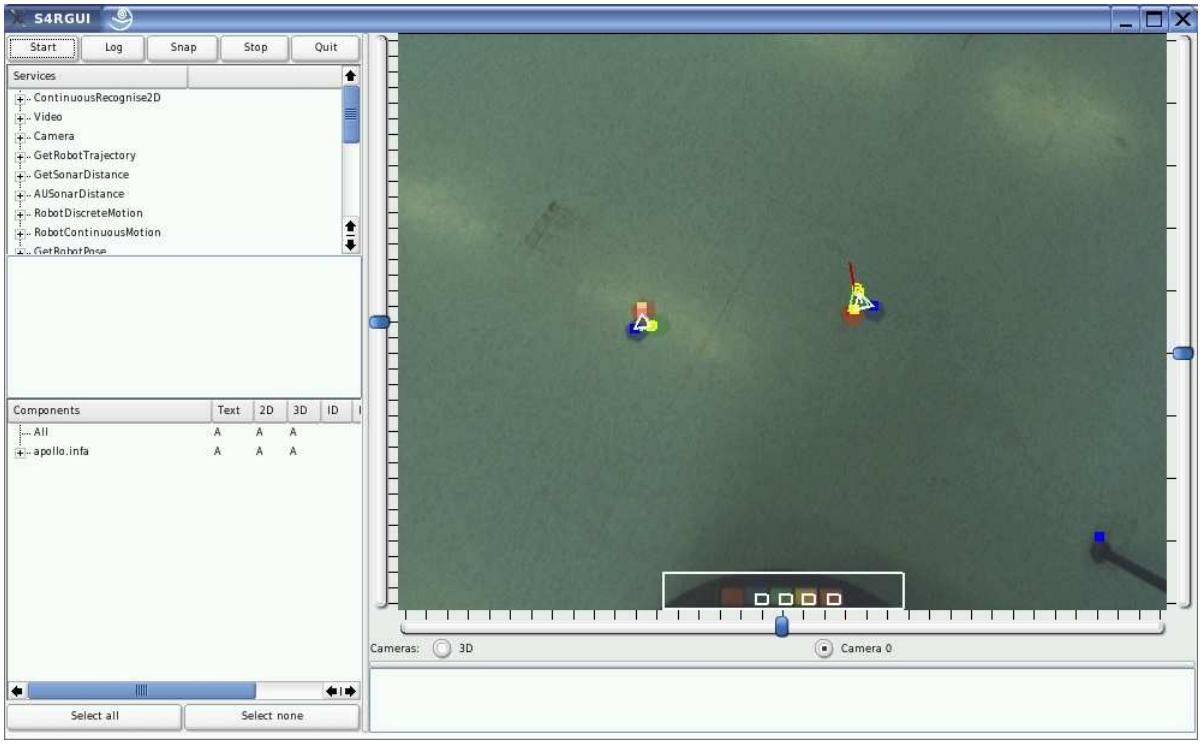

Abbildung 6.5: Die aktuelle S4R Benutzeroberfläche.

Push-Buttons wie z.B. „Start“ könnte eine Reihe weiterer, frei belegbarer Push-Buttons zur Kommunikation mit dem Roboter benutzt werden. Diese Hotkeys könnten außerhalb der Programmierung über eine Konfiguration belegbar sein. Die zugeordneten Befehle können dann beispielsweise über die Funktionstasten abrufbar sein. Ein Bedienbeispiel dazu: Der Roboter meldet die Durchführung der letzten Aufgabe und der Benutzer wählt durch Drücken der F3 Taste die nächste Aufgabe für den Roboter aus.

Sound Die Möglichkeit Informationen an den Benutzer weiterzugeben ist im Framework derzeit auf die Präsentation von Bild oder Text am Bildschirm des Laptops beschränkt. Das beschränkt die Bedienbarkeit, da beim fahrenden Roboter Informationen übersehen werden können. Eine Erweiterung der Ausgabemöglichkeiten durch das Abspielen von Audiodateien würde einen bisher ungenutzen Informationskanal zur Verfügung stellen. Eine Realisierung über eine Konfiguration, in der Befehle mit Audiodateien verknüpft werden, würde sich anbieten. Der Roboter kann dann Informationen oder Zustände wie z.B. „Ziel erreicht“, „Aufgabe X erfüllt“ oder „Räder blockieren“ akustisch wiedergeben.

Startup Configuration Der nach dem Programmstart vorhandene Zustand der Benutzeroberfläche sollte konfigurierbar sein. Folgende Einstellungen könnten berücksichtigt werden:

- Dargestellte „Kamera“ bzw. „View“ nach dem Start.
- Konfigurierbares „Autorun“, das keinen Klick auf den „Start“-Button mehr benötigt.
- Automatisches Logging vom Beginn an.

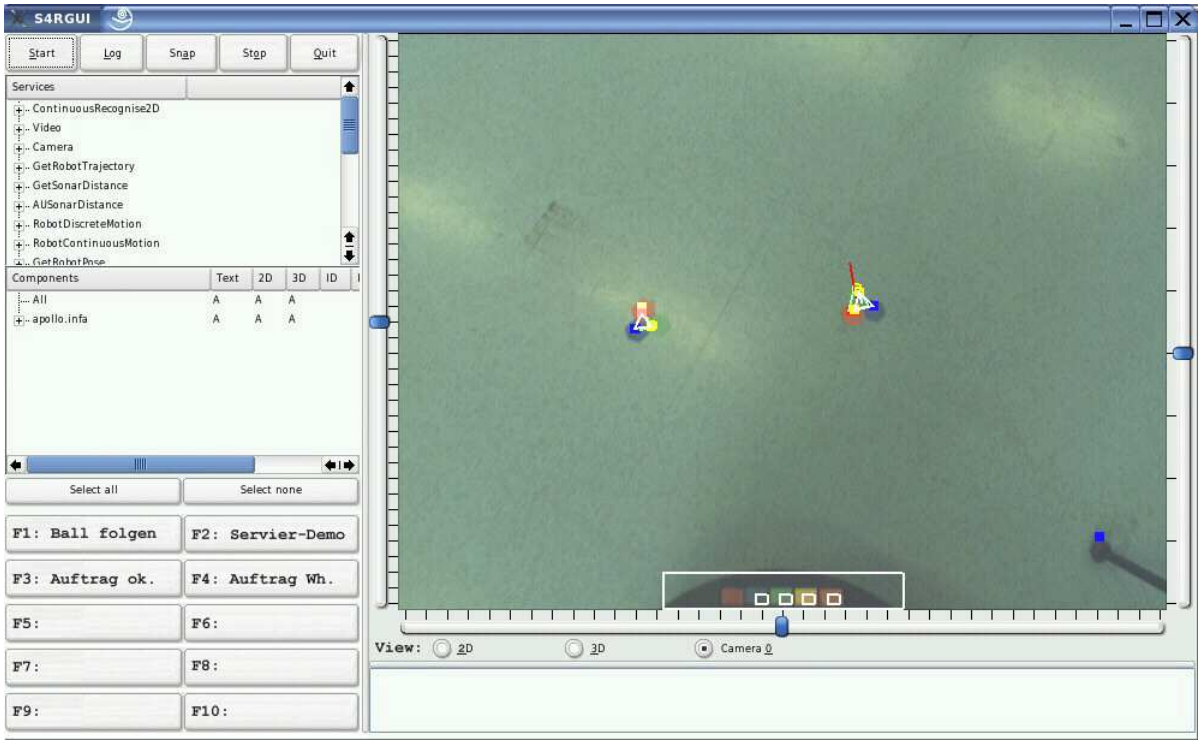

Abbildung 6.6: Vorschläge für künftige Erweiterungen der S4R Benutzeroberfläche für den Einsatz mit einem mobilen Roboter.

6.3 Zusammenfassung

Die Ergebnisse dieser Arbeit sind direkt auf die für die Steuerung des mobilen Roboters verwendete Umgebung bezogen. Die Vorschläge zum S4R-Framework sind konkret und können direkt umgesetzt werden. Die Arbeit versteht sich als Fortsetzung der im Rahmen des S4R-Frameworks (und seiner Vorgängerversionen) geleisteten Arbeit. In diesem Sinne verstehen sich die Vorschläge auch nur als ein Schritt zu einer bessern Integration von Computer Vision und mobiler Robotik im Rahmen des S4R-Frameworks. Im nächsten Kapitel finden sich konkrete Vorschläge für eine Fortsetzung dieser Arbeit. Eine Liste von Interessenschwerpunkten und weiterführenden Demonstrationsbeispielen soll einen möglichen künftigen Forschungsweg zeigen.

Kapitel 7

Schlussbemerkung

Die Ergebnisse aus Kapitel 6 lassen ein wenig vom Potential von Computer Vision für die mobile Robotik erahnen. Die Diplomarbeit ist aber nur ein erster Blick auf die komplexe Thematik der Integration der beiden Technologien. Die Forschung in diesem Bereich zur Integration von Computer Vision in die mobile Robotik im Institut für Automatisierungs- und Regelungstechnik (ACIN) sollte auf jeden Fall fortgesetzt werden. Den praktischen Applikationen der fortgeschrittenen Computer Vision wird in Fachkreisen im Moment eher wenig Aufmerksamkeit gespendet. Wenn die mobile Robotik in den nächsten Jahren aber den Einzug in die Haushalte versuchen sollte, dann wird der Bedarf nach robusten Bildverarbeitungen groß sein. Die Integration der beiden Technologien stellt damit eine Schlüsselfunktion für Serviceroboter dar.

Die ausführlich gehaltenen einführenden Kapitel dienen als Referenz und Überblick für eine weitergehende Forschung. Die im Rahmen der Arbeit implementierten Beispiele im S4R-Framework haben sich als praxistauglich erwiesen. Die im Rahmen der Arbeit gefundenen Erkenntnisse und die beim Experimentieren entdeckten Möglichkeiten der behavior-basierten Robotik bieten zudem zahlreiche Anknüpfungspunkte für weitere praktische Versuche.

Objekterkennung mit unterschiedlichen Algorithmen Das Follow Ball Demonstrationsbeispiel ist so ausgelegt, dass es mit einer Objekterkennung als Sensor arbeiten kann. Diese Eigenschaft kann mit den im S4R Framework vorhandenen Computer Vision Services, die eine Objekterkennung ermöglichen, getestet werden. Ein Beispiel dafür wären eine Reihe von vorhandenen Algorithmen, die Gegenstände an Hand ihres Farbhistogramms erkennen können.

Visuelle Kalibrierung der Odometrie Beim Multi Color Behavior Demonstrationsbeispiel erkennt der Roboter Farb-Markierungen. Man bringt in einem Raum an bekannten Koordinaten unterschiedliche eindeutige Markierungen am Boden oder an einer Wand an. Sind zwei derartige Markierungen im Bild sichtbar, kann daraus trigonometrisch die Position des Roboters berechnet werden. Diese Information kann zum Kalibrieren der Odometrie benutzt werden. Eine Navigation auf Basis der Odometrie ist damit möglich, sofern der Roboter in regelmäßigen Zeitabständen eine derartige Kalibrierung durchführen kann, um die durch die Fahrbewegung aufsummierten Fehler zu korrigieren.

Sensor Fusion von Computer Vision und Sonar Die Behavior Struktur ermöglicht eine einfache Verbindung der Daten verschiedener Sensorentypen. Eine solche *Sensor Fusion* ist erfolgreich, wenn die Vorteile der verschiedenen Sensortypen kombiniert werden können. Eine Versuchsreihe mit der Verbindung Computer Vision und Sonar Sensoren bietet

sich an. Der Pioneer 2-DX Testroboter ist mit acht Sonarsensoren ausgestattet, das S4R Framework unterstützt die Sonarsensoren und es gibt im Framework bereits Erfahrung und Testbeispiele zu den Sonarsensoren.
Das erste Beispiel dazu kann eine 20 - 30 cm große farbige Schachtel als Objekt verwenden. Ein Farbtracker kann das Objekt gut erkennen. Die Sonarsensoren funktionieren mit harten, reflektierenden Oberflächen ebenfalls gut. Die beiden Informationen über die Position im Bild und die Entfernung vom Roboter können in einem Behavior kombiniert werden. In den Tests dazu könnte dann beispielsweise ein Reiz ausfallen. Statt der Schachtel wird ein farbiges Blatt Papier verwendet. Die Farbe wird erkannt, aber die Sonarsensoren melden kein Hindernis. Weitere Beispiele können dann mit mehreren derartigen Objekten arbeiten. Das Ziel hierbei ist ein stabiles Erkennen der Objekte.

Die Verwendung einer Kamera als Sensor in der Servicerobotik ist auf Grund der erfassbaren Datenmenge, dem Preis und nicht zuletzt wegen der Sonderstellung als analog zum primären Sinnesorgan des Menschen zu favorisieren. Die Interpretation der Kameradaten ist aber sehr aufwendig und in den meisten Fällen noch nicht robust genug für den Einsatz in einer Serviceumgebung. Diese Diplomarbeit hat jedoch gezeigt, dass gerade die Domäne selbst Voraussetzungen schaffen kann, um den Interpretationsschritt zu vereinfachen. Beispiele sind der Einsatz von Kalibrierungstafeln, die Verwendung von geeigneten, künstlichen Markern oder die Integration mehrerer Sensoren. Ich hoffe diese Arbeit kann einen Beitrag zur Forschung und Entwicklung eines autonomen, mobilen Serviceroboters leisten.

Anhang A

Technische Daten Laptop „Apollo“

Auf dem Laptop ist als Betriebssystem Suse Linux 9.2 installiert. Die folgenden Informationen stammen aus der Systemdatei `/proc/cpuinfo` des Betriebssystems.

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 15
model           : 2
model name      : Intel(R) Pentium(R) 4 Mobile CPU 1.60GHz
stepping        : 4
cpu MHz         : 1196.731
cache size      : 512 KB
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 2
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 sep mtrr pge mca cmov pat
                  pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
bogomips        : 2365.44
```

Die folgenden Informationen stammen aus der Systemdatei `/proc/meminfo` des Betriebssystems.

```
MemTotal:       515124 kB
MemFree:        309404 kB
Buffers:         38912 kB
Cached:          83568 kB
SwapCached:      27156 kB
Active:         133708 kB
Inactive:        48916 kB
HighTotal:           0 kB
HighFree:            0 kB
```

```
LowTotal:       515124 kB
LowFree:        309404 kB
SwapTotal:     1028120 kB
SwapFree:       984472 kB
Dirty:               0 kB
Writeback:           0 kB
Mapped:          79280 kB
Slab:            14316 kB
Committed_AS:   186844 kB
PageTables:       1560 kB
VmallocTotal:   507896 kB
VmallocUsed:     12884 kB
VmallocChunk:   494840 kB
HugePages_Total:     0
HugePages_Free:      0
Hugepagesize:     4096 kB
```

Anhang B

Seriell/USB-Adapter Unterstützung für die Mobile Robot Services

Die Mobile Robot Services des S4R-Frameworks unterstützen zur Zeit nur die serielle Schnittstelle zur Kommunikation mit dem Roboter. Wenn keine serielle Schnittstelle am Computer vorhanden ist, kann mit der folgenden Konfigurationsänderung ein Seriell7USB-Adapter verwendet werden.

In der Konfigurationsdatei (i.a. `robot.cfg`), die beim Aufruf der Mobile Robot Services verwendet wird, muss im Abschnitt `CommunicationPort` der Eintrag `PortName` geändert werden. Der folgende Ausschnitt aus der Konfigurationsdatei zeigt die Konfiguration für den Seriell/USB-Adapter und auskommentiert die Konfiguration für die serielle Schnittstelle.

```
# ------------------------------------------------------------------
# Configuration data for the class 'cCommunicationPort'.
# ------------------------------------------------------------------
[CommunicationPort]

ID 0 # ID of the class's instance
PortType Serial # Type of the port
# PortName /dev/ttyS0 # Serial port
PortName /dev/ttyUSB0 # USB port
BaudRate 9600 # Baud rate
DataBits 8 # Number of data bits
HalfStopBits 2 # Number of HALF stop bits
UseParity false # Use parity check or not
OddParity false # Type of parity
TimeOutInMs 200 # Data reception timeout (ms)
```

Zusätzlich müssen die Benutzerberechtigungen von `ttyUSB0` umgesetzt werden:

```
su root
chmod a+rw /dev/ttyUSB0
```

Anhang C

Vorlage eines Behaviors: Follow Ball

Das Follow Ball Behavior ist ein schlichtes Beispiel für einer Robotersteuerung mit Hilfe einer Objekterkennung. Im Beispiel wurde der Farbtracker TrackObject2D verwendet. Durch Änderung der Methode `MRV_TrackObject2D_Requester` auf eine gewünschte andere Objekterkennung kann diese mit dem Follow Ball Behavior kombiniert werden.

Makefile

```
#
# $Id: Makefile,v 1.1 2006/06/30 12:18:43 sk Exp $
#
# Makefile for a component
#

# our component
COMPONENT = MRV_Follow_Ball

# services that we provide/request
SERVICES = GUI TrackObject2D RobotContinuousMotion

# our object files
COMP_OBJECTS = MRV_Follow_Ball.o

COMP_LDLIBS = -lACINBase -lS4RBase

#------ do not change below this line --------------------------------
include ../../Make.comp
```

MRV_Follow_Ball.hh

```
//-----------------------------------------------------------------
// Author: Sascha Krasny
// Last update: 30.06.2006
//-----------------------------------------------------------------

#ifndef MRV_FOLLOW_BALL_HH
```

```
#define MRV_FOLLOW_BALL_HH

#include "framework/Component.hh"
#include "services/RobotContinuousMotion.hh"
#include "services/TrackObject2D.hh"

//-------------------------------------------------------------------
class MRV_RobotContinuousMotion_Requester;
class MRV_TrackObject2D_Requester;

//-------------------------------------------------------------------
// Test component.
//-------------------------------------------------------------------
class MRV_Follow_Ball : public Component
{
  private:
    MRV_RobotContinuousMotion_Requester *rMove;
    MRV_TrackObject2D_Requester *tracker;

    Vector2I trackPos;
    int posStatus;

    virtual void Init();
    virtual void Step();
  public:
    MRV_Follow_Ball();
    ~MRV_Follow_Ball();
    void HaveObject(Vector2I position);
};

//-------------------------------------------------------------------
// Requester for a 'RobotContinuousMotion' service.
//-------------------------------------------------------------------
class MRV_RobotContinuousMotion_Requester :
  public RobotContinuousMotion_Requester
{
  private:
    virtual void Stop_Response(int Result);
    virtual void MoveConstantly_Response(int Result);
    virtual void TurnConstantly_Response(int Result);
  public:
    int MoveResult;
    int TurnResult;
    int StopResult;
};

//-------------------------------------------------------------------
// Our implementation of the Video service reqester.
```

```
//-----------------------------------------------------------------
class MRV_TrackObject2D_Requester : public TrackObject2D_Requester
{
protected:
  // methods overloaded from TrackObject2D_Requester
  virtual void TrackObject2D_Response(Vector2I position, TimeVal time);
};

#endif
```

MRV_Follow_Ball.cc

```
//-----------------------------------------------------------------
// Author: Sascha Krasny
// Last update: 30.06.2006
//-----------------------------------------------------------------
#include "MRV_Follow_Ball.hh"
//-----------------------------------------------------------------
#define TURN_SHARPLY 1.0
#define TURN_FAST 0.8
#define TURN_MODERATELY 0.65
#define TURN_SLOWLY 0.25
//-----------------------------------------------------------------
#define MOVE_SLOWLY 0.15
#define MOVE_FASTER 0.30
//-----------------------------------------------------------------
#define CAMERA_NR 0
//-----------------------------------------------------------------
#define SCREEN_X 640.0
#define SCREEN_Y 480.0


//-----------------------------------------------------------------
// Creates a new instance of the component.
//-----------------------------------------------------------------
extern "C" {
Component* NewComponent()
{
return new MRV_Follow_Ball();
}
}

//-----------------------------------------------------------------
// The constructor of the class.
//-----------------------------------------------------------------
MRV_Follow_Ball::MRV_Follow_Ball()
{
rMove = (MRV_RobotContinuousMotion_Requester *)NULL;
```

```
}

//------------------------------------------------------------------
// The destructor of the class conditionally removes the requesters.
//------------------------------------------------------------------
MRV_Follow_Ball::~MRV_Follow_Ball()
{
if (rMove != (MRV_RobotContinuousMotion_Requester *)NULL)
{
RemoveRequester(rMove);
rMove = (MRV_RobotContinuousMotion_Requester *)NULL;
}
}

//------------------------------------------------------------------
// Get a requesters for a "RobotContinuousMotion"- and a "TrackObject2D"-
// service.
//------------------------------------------------------------------
void MRV_Follow_Ball::Init()
{
// unsigned int u;
// bool WaitForAll = false;

// Robot Continous Motion Init
rMove =
(MRV_RobotContinuousMotion_Requester *)AddRequester(new
MRV_RobotContinuousMotion_Requester());
rMove->MoveResult = 1;
rMove->TurnResult = 1;
rMove->StopResult = 1;

// Object Tracker Init
tracker = new MRV_TrackObject2D_Requester();
AddRequester(tracker);
ROI pos(Vector2I(0,0),640,480);
string model("BALL");
tracker->TrackObject2D(model, pos, CAMERA_NR);

posStatus = -1;

SetRunMode(WAITING);
}

//------------------------------------------------------------------
// This function is periodically called by the framework.
//------------------------------------------------------------------
void MRV_Follow_Ball::Step()
{
```

```
if (rMove->MoveResult != 1)
{
if (rMove->MoveResult != SRCM_SUCCESS)
debug.Error("The move result was no success.\n");
rMove->MoveResult = 1;
}
if (rMove->TurnResult != 1)
{
if (rMove->TurnResult != SRCM_SUCCESS)
debug.Error("The turn result was no success.\n");
rMove->TurnResult = 1;
}
if (rMove->StopResult != 1)
{
if (rMove->StopResult != SRCM_SUCCESS)
debug.Error("The stop result was no success.\n");
rMove->StopResult = 1;
}
}


//-----------------------------------------------------------------
// Handler for a "move constantly" - response.
//-----------------------------------------------------------------
void MRV_RobotContinuousMotion_Requester::MoveConstantly_Response(int Result)
{
MoveResult = Result;
}

//-----------------------------------------------------------------
// Handler for a "turn constantly" - response.
//-----------------------------------------------------------------
void MRV_RobotContinuousMotion_Requester::TurnConstantly_Response(int Result)
{
TurnResult = Result;
}

//-----------------------------------------------------------------
// Handler for a "stop" - response.
//-----------------------------------------------------------------
void MRV_RobotContinuousMotion_Requester::Stop_Response(int Result)
{
StopResult = Result;
}

//-----------------------------------------------------------------
// This method gets called when a GetImage response arrives.
//-----------------------------------------------------------------
```

```
void MRV_Follow_Ball::HaveObject(Vector2I position)
{
double MoveSpeed = 0.0;
double TurnSpeed = 0.0;
double MoveWeight = 1.0;

// Ball liegt links
if ((position.x > 0) && (position.x < 280)) {
TurnSpeed = ((TURN_FAST-TURN_SLOWLY)/280.0*(280.0-position.x)) + TURN_SLOWLY;
}

// Ball liegt rechts
if (position.x > 360) {
TurnSpeed =
(-1.0)*(((TURN_FAST-TURN_SLOWLY)/280.0*(position.x-360.0)) + TURN_SLOWLY);
}

// Ball liegt weit
if ((position.y > 0) && (position.y < 300)) {
MoveSpeed = ((MOVE_FASTER - MOVE_SLOWLY)/300.0*position.y) + MOVE_SLOWLY;
}

// Ball liegt nah
if (position.y > 380) {
MoveSpeed = (-1.0) * MOVE_SLOWLY;
}

rMove->TurnConstantly(TurnSpeed, MoveWeight);
rMove->MoveConstantly(MoveSpeed, MoveWeight);

if ((position.x != 0)&&(position.y != 0)) {
gui->DrawBox2D(CAMERA_NR, position.x-5, position.y-5,
position.x+5, position.y+5, 0x0000ff, false);
}
}

//-----------------------------------------------------------------------
// Our implementation of the TrackObject2D_Response() method.
//-----------------------------------------------------------------------
void MRV_TrackObject2D_Requester::TrackObject2D_Response(Vector2I
position, TimeVal time)
{
((MRV_Follow_Ball*)GetComponent())->HaveObject(position);
}
```

Anhang D

Index

ActivMedia Robotics (inzwischen Mobile Robotics) .. 3.1.4, 5.1.1
Arkin, Ronald C. (Motor Schemas) .. 2.5.2, 3.1.4
Autonomer Roboter .. 2.1

Behavior, Definition .. 3.2, 3.3
Behavior, Darstellung .. 3.4
Behavior, Kodierung .. 3.6
Behavior, Koordinatoren .. 3.5
behavior-basierte Robotik, allgemein .. 2.4.2, 3
behavior-basierte Robotik *behavior-bases robotics* (Ronald C. Arkin) 3.1.4
behavior-basierte Robotik, Geschichte .. 3.1
Braitenberg, Valentino (Braitenberg Vehicles) .. 3.1.3
Braitenberg Vehicles (Valentino Braitenberg) .. 3.1.3
Brooks, Rodney Allen (Subsumption architecture) .. 2.5.1, 3.1.3

Computer Vision, allgemein .. 4.2, 4.2.1, 4.2.2, 4.2.3, 4.2.3, 4.2.4
Computer Vision, mobile Roboter .. 4.2.7, 6.1, 4.2.8
Diskrete Bewegungssteuerung *discrete motion, direct motion* .. 2.3.1

Endliches Akzeptor Diagramm *finit state acceptor* (Behavior) .. 3.4.3, 5.5.2
Erweitertes Reiz-Reaktionsdiagramm .. 6.1.4, 6.1.5

Farb-Recogniser ContinousRecognise2D .. 5.1.2, 5.5.3
Farbtracker CMV_Track2D .. 5.1.2, 5.2.3, 5.3.3, 5.4.2, 5.5.3
Funktionale Notation *functional notation* (Behavior) .. 3.4.2
Fuzzy Control (Behavior) .. 3.6.2

Hierarchisches Paradigma (Steuerungskonzept) .. 2.4.1
Hybrides Paradigma (Steuerungskonzept) .. 2.4.3

Infrarotsensoren .. 4.1.6

Kalibriertafel .. 5.5.3
Konkurrenzmethoden (Behavior) .. 3.5.2
Konolige Kurt (Saphira Architektur) .. 2.5.3, 3.1.4
Kontinuierliche Bewegungssteuerung *continuous motion* .. 2.3.2
Kooperative Methoden (Behavior) .. 3.5.1

Laserscanner .. 4.1.7

look-then-move .. 4.2.6

Machine Vision .. 4.2.6
Mobile Robot Services .. 5.1.2, 6.2.3
Mobile Robotics (früher ActivMedia Robotics) .. 3.1.4, 5.1.1
Mobiler Roboter *mobile robot* .. 2.1
Motor Schemas (Ronald C. Arkin) .. 2.5.2, 3.1.4

Odometrie .. 4.1.2

Pioneer 2-DX Roboter .. 5.1.1

Reaktives Paradigma (Steuerungskonzept) .. 2.4.2
Reiz-Reaktionsdiagramm (Behavior) .. 3.4.1, 4.2.8
Reiz-Reaktionsverbindung *stimulus-response* .. 2.4.2, 3.2
Remote Objects Control Interface .. 4.2.9.1

S4R-Framework .. 5.1.2, 6.2, 6.2.1, 6.2.2, 6.2.3, 6.2.4
Saphira Architektur (Kurt Konolige) .. 2.5.3, 3.1.4
Schematische Bettwanze *schematic sowbug* (Edward C. Tolman) .. 3.1.2
Schildkröte *tortoise* (Walter Grey Walter) .. 3.1.2
Sensoren .. 4.1
Serviceroboter .. 2.1, 2.2
Steuerungskonzept (mobile Roboter) .. 2.4
structure from motion .. 4.2.6
Subsumption architecture (Rodney Allen Brooks) .. 2.5.1, 3.1.3

Taktile Sensoren .. 4.1.1
The Imaging Source DFK 21F04 Farbkamera .. 5.1.1, 5.2.3, 5.3.3, 5.4.2, 5.5.3
Tolman, Edward C. .. (Schematische Bettwanze, Zielgerichteter Behaviorismus) .. 3.1.2

Ultraschallsensoren .. 4.1.5

VIP-Framework .. 4.2.9.2, 6.1.2
Virtuelles Potentialfeld (Behavior) .. 3.6.1
Virtueller Sensor .. 6.1.4, 6.1.5

Walter, Grey Walter (Schildkröte) .. 3.1.2

YO!Einstein .. 5.5.5

Zielgerichteter Behaviorismus *purposive behaviorism* (Edward C. Tolman) .. 3.1.2

Literaturverzeichnis

[ABRG06] Adaptive Behaviour Research Group. *Whiskerbot: A Robot Whisker System Modelled on the Rat Mystacial Vibrissae.* University of Sheffield
Verfügbar unter: http://www.shef.ac.uk/ abrg/projects/robot/robot_whiskers.shtml [15.09.2005]

[ACT06] Markus Vincze. *ActIPret - Interpreting and Understanding Activities of Expert Operators for Teaching and Education.* Institut für Automatisierung und Regelungstechnik der Technischen Universität Wien.
Verfügbar unter: http://actipret.infa.tuwien.ac.at/Sites/frameset.htm

[AMR06] ActivMedia Robotics bzw. Mobile Robots Inc. *Pioneer DX2 Roboter. Spezifikation und Dokumentation.*
Verfügbar unter: http://www.mobilerobots.com/

[Ark89] Ronald C. Arkin. *Motor Schema-Based Mobile Robot Navigation.* International Journal of Robotics Research, Vol. 8, Seiten 92 - 112, 1989

[Ark95] Ronald C. Arkin. *Reactive Robotic Systems.* In Arbib, M., editor, The handbook of brain theory and neural networks, Seiten 793 - 796. MIT press, Cambridge MA, 1995
Verfügbar unter: http://www.cc.gatech.edu/ai/robot-lab/online-publications/handbock.pdf [15.03.2006]

[Ark98] Ronald C. Arkin. *Behavior-Based Robotics.* Cambridge, MIT Press, 1998

[AuRo] G.A. Bekey (Editor-in-Chief) *Autonomous Robots.* ISSN: 0929-5593, Springer US, 2006
Verfügbar unter: http://www.kluweronline.com/issn/0929-5593 [15.05.2006]

[BEF96] J. Borenstein, H.R. Everett, L. Feng. *Navigating Mobile Robots - System and Techniques.* A K Peters Ltd., 1996

[BeMc03] Randal W. Beard, Timothy W. McLain. *Motion Planning using Potential Fields.*
Verfügbar unter: http://www.ee.byu.edu/ugrad/srprojects/robotsoccer/labs/skills/potential.pdf [30.03.2006]

[Bra84] Valentino Braitenberg. *Vehicles: Experiments in Synthetic Psycology.* Cambridge, the MIT Press, 1984.

[Bre02] Cynthia L. Breazeal. *Designing Sociable Robots.* The MIT Press. 2002

[Bro86] Rodney A. Brooks. *A Robust Layered Control System for a Mobile Robot.* IEEE Journal of Robotics and Automation. Vol. RA-2, No. 1, 1986, Seiten 14 - 23

[Bro90] Rodney A. Brooks. *Elephants dont play Chess.* Robotics and Autonomous Systems Vol. 6, 1990, Seiten 3 - 15.
Verfügbar unter: http://people.csail.mit.edu/brooks/papers/elephants.pdf [15.03.2006]

[Bro91] Rodney A. Brooks. *Intelligence without Reason.* A.I. Memo No. 1293, Massachusetts Institute of Technonolgy, Artificial Intelligence Laboratory, 1991
Verfügbar unter: http://people.csail.mit.edu/brooks/papers/AIM-1293.pdf [15.03.2006]

[CCT06] Anthony Cowley, Luiz Chaimowicz, Camillo J. Taylor. *Design Minimalsim in Robotics Programming.* International Journal of Advanced Robotic Systems, Vol. 3, No. 1, 2006

[CMV06] *The CORAL Group's Color Machine Vision Project.* Carnegie Mellon University
Verfügbar unter: http://www.cs.cmu.edu/ jbruce/cmvision/ [15.06.2006]

[Cor93] P. I. Corke. *Visual Control of Robot Manipulators - A Review.* In K. Hashimoto, editor, Visual Servoing, Robots and Automated Systems Vol. 7, pp 1 - 31. World Scientific, 1993

[DARPA] Defense Advanced Research Projects Agency. *DARPA Grand Challenge.*
Verfügbar unter: http://www.darpa.mil/GRANDCHALLENGE/ [15.03.2006]

[EA01] Yoichiro Endo, Ronald C. Arkin. *Implementing Tolmans Schematic Sowbug: Behavior-Based Robotics in the 1930s.* Proceedings of IEEE International Conference on Robotics and Automation (ICRA), Seoul, 2001, Seiten 477 - 484.
Verfügbar unter: http://www.cc.gatech.edu/aimosaic/robot-lab/research/eBug/ICRA2001-endo-arkin.pdf [15.03.2006]

[Ein05] Peter Einramhof. *Steuerung eines mobilen Roboters.* Diplomarbeit am Institut für Automatisierung und Regelungstechnik der Technischen Universität Wien. 2005

[GCJK97] D. Guzzoni, A. Cheyer, L. Julia, and K. Konolige. *Many robots make short work: Report of the SRI International mobile robot team.* AI Magazine, Seiten 55-64, 1997.

[Good] Michael A. Goodrich. *Potential Fields Tutorial.*
Verfügbar unter: students.cs.byu.edu/ cs470ta/readings/Pfields.pdf [30.03.2006]

[Hor86] Berthold Klaus Paul Horn. *Robot Vision.* The MIT Press. 1986

[Kon97] Kurt Konolige. *Colbert: A language for reactive control in sapphira.* In G. Brewka, C. Habel, and B. Nebel, editors, Advances in Artificial Intelligence, volume 1303 of Lecture Notes in Computer Science. Springer, 1997.

[Kon98] Kurt Konolige, Karen L. Myers. *The Saphira architecture for autonomous mobile robots.* Aus David Kortenkamp, R. Peter Bonasso, Robin Murphy, editors, Artificial Intelligence and Mobile Robots: case studies of successful robot systems, Seiten 211-242. MIT Press, AAAI Press, 1998. ISBN: 0-262-611376

[OHCI1394] The Promoters of the 1394. *Open HCI. Open Host Controller Interface Specification for 1394.* Release 1.1. 2000
Verfügbar unter: http://developer.intel.com/technology/1394/download/ohci_11.htm [01.06.2006]

[OHCIUSB] Compaq, Microsoft, National Semiconductor. *Open HCI. Open Host Controller Interface Specification for USB.* Release 1.0a. 1999
Verfügbar unter: http://www.mit.edu/afs/sipb/contrib/doc/specs/ic/bridge/ohci10a.pdf [01.06.2006]

[OlGo03] Dan R. Olsen, Michael A. Goodrich. *Metrics for evaluating human-robot interactions.* PERMIS 2003, September 2003

[Oni06] Norimitsu Onishi. *In a Wired South Korea, Robots Will be at Home.* Der Standard / The New York Times, 10. April 2006

[PHS06] Erwin Prassler, Martin Hägele und Roland Siegwart. *International Contest for Cleaning Robots: Fun Event or a First Step towards Benchmarking Service Robots.* S. Yuta et al. (Eds.): Field and Service Robotics, STAR 24, Seiten 447-456, 2006

[RAS06] IEEE Robotics and Automation Society.
Verfügbar unter: http://www.ncsu.edu/IEEE-RAS/RAS/AboutRAS.htm [15.03.2006]

[Sta97] Werner Stangl. *Zur Wissenschaftsmethodik in der Erziehungswissenschaft.* Werner Stangls Arbeitsblätter, 1997
Verfügbar unter: http://paedpsych.jk.uni-linz.ac.at/INTERNET/ARBEITSBLAETTERORD/default.html [15.03.2006]

[TIS06] The Imaging Source. *DFK 21F04 FireWire Farb-Kamera. Spezifikation und Dokumentation.* 2006
Verfügbar unter: http://www.1394imaging.com/de/products/cameras/firewire_color/dfk21f04/overview/ [15.04.2006]

[Tol39] Edward C. Tolman. *Prediction of Vicarious Trial and Error by Means of the Schematic Sowbug.* Psychological Review. Vol. 46, 1939, Seiten 318 - 336

[UKMK06] Hans Utz, Ulrich Kaufmann, Gerd Mayer, Gerhard K. Kraetzschmar. *VIP - A Framework-Based Approach to Robot Vision.* International Journal of Advanced Robotic Systems, Vol. 3, No. 1, 2006

[UNECE03] United Nations Economic Commission for Europe. *World Robotics 2003 - Statistics, Market Analysis, Forcasts, Case Studies and Profitability of Robot Investment.* 2003.

[Vid06] Videre Design. *Laser Rangefinder.*
Verfügbar unter: http://www.videredesign.com/robotics/LaserRanger.htm [01.06.2006]

[Wal53] Walter G. Walter. *The Living Brain.* New York, 1953

[YOE05] *„YO!Einstein“ die neue Lust an Mathe, Chemie und Physik.*
Verfügbar unter: http://www.yo-einstein.at/ [15.04.2006]

[Zac03] Gerhard Zach. *Der modulare Stereokamerakopf.* Diplomarbeit am Institut für Automatisierung und Regelungstechnik der Technischen Universität Wien. 2003