Interaktive Verhaltenssteuerung für Robot Companions

Marcus Kleinehagenbrock

Dipl.-Inform. Marcus Kleinehagenbrock
AG Angewandte Informatik
Technische Fakultät
Universität Bielefeld
email: mkleineh@techfak.uni-bielefeld.de

Abdruck der genehmigten Dissertation zur Erlangung
des akademischen Grades Doktor-Ingenieur (Dr.-Ing.).
Der Technischen Fakultät der Universität Bielefeld
am 15.12.2004 vorgelegt von Marcus Kleinehagenbrock,
am 27.04.2005 verteidigt und genehmigt.

Gutachter:

Dr. Jannik Fritsch, Universität Bielefeld
Prof. Dr. Erwin Prassler, Fachhochschule Bonn-Rhein-Sieg

Prüfungsausschuss:

Prof. Dr. Helge Ritter, Universität Bielefeld
Prof. Dr. Gerhard Sagerer, Universität Bielefeld
Dr. Jannik Fritsch, Universität Bielefeld
Dr. Stefan Kopp, Universität Bielefeld

Gedruckt auf alterungsbeständigem Papier nach ISO 9706

Interaktive Verhaltenssteuerung

für Robot Companions

Der Technischen Fakultät der Universität Bielefeld

zur Erlangung des Grades

Doktor-Ingenieur

vorgelegt von

Marcus Kleinehagenbrock

Bielefeld – Dezember 2004

Danksagung

Als erstes bedanke ich mich bei meinem Betreuer Dr. Jannik Fritsch für seine vorbildliche Unterstützung. Sein persönliches Interesse an meiner Arbeit und die daraus resultierten Ratschläge und Anregungen waren überaus motivierend für mich und haben sehr zum Gelingen dieser Dissertation beigetragen. Ich danke ihm außerdem dafür, dass er sich dazu bereit erklärt hat, diese Ausarbeitung zu begutachten. Ebenso bedanke ich mich bei Prof. Dr. Erwin Prassler für seine Bereitschaft, als Zweitgutachter einzutreten.

Als nächstes möchte ich Prof. Dr. Gerhard Sagerer dafür bedanken, dass er mich für ein Promotionsstudium begeistern konnte und es mir schließlich auch ermöglicht hat. In diesem Kontext danke ich auch Michael Pfeiffer, der in mir überhaupt erst den Gedanken geweckt hat, eine Promotion anzustreben. Ein besonderer Dank geht außerdem an meinen Bürokollegen Sebastian Lang, der jederzeit für technische und fachliche Diskussionen zur Verfügung stand und dabei viele Ideen, Anregungen und überaus konstruktive Kritik lieferte. Die damit verbundene, sehr angenehme, produktive und nicht selten auch lustige Büroatmosphäre habe ich stets zu schätzen gewusst. Bei Axel Haasch bedanke ich mich dafür, dass er mich in den letzten Monaten vor der Fertigstellung dieses Dokuments bei diversen technischen Arbeiten entlastet und somit viele Mühen auf sich genommen hat. Besonders bedanken möchte ich mich auch bei meiner Freundin Shuyin Li, die mich durch ihren „multimodalen Beistand“ nicht selten davor bewahrt hat, dass ich mich beim Erstellen dieser Niederschrift hoffnungslos überarbeitet habe.

Nicht zuletzt möchte ich mich noch bei allen übrigen Mitgliedern der AG „Angewandte Informatik“ bedanken, die ebenfalls dazu beigetragen haben, eine sehr angenehme Arbeitsatmosphäre zu schaffen. Das Zusammengehörigkeitsgefühl innerhalb dieser Gruppe ist wirklich außergewöhnlich. Leute, bleibt so!

Inhaltsverzeichnis

1. Einleitung

Wäre es nicht großartig, wenn jeder Mensch in Zukunft einen eigenen Roboter besitzen könnte, mit dem man ähnlich wie mit einem Menschen kommunizieren kann? Ein Roboter, der einem zuhause bei der Hausarbeit helfen kann, indem er beispielsweise Pflanzen gießt oder den Esstisch abräumt. Ein Roboter, der einem Gesellschaft leistet und einen unterhalten kann. Also ein Roboter, der einen richtigen Partner im Haushalt darstellt.

Solch ein Roboter wäre insbesondere für Menschen praktisch, die nicht selbstständig alle zuhause anfallenden Aufgaben ausführen können, wie z.B. behinderte oder ältere Menschen. Hinzu kommt, dass die Anzahl älterer Menschen, die einer Betreuung bedürfen, stetig ansteigt. Schätzungen zufolge wird es in Deutschland im Jahr 2030 mehr als doppelt so viele 65-jährige Menschen geben wie heute. Entsprechende Statistiken besagen, dass im Jahr 2040 etwa 3.5% der deutschen Bevölkerung gepflegt werden müssen, was einer Steigerung von 66% im Vergleich zu heute darstellt [Sch99]. Ähnliche Studien bestätigen diesen Trend auch für andere Länder, wie z.B. für die Vereinigten Staaten [Roy00]. Der mit dieser Situation verbundene erhöhte Bedarf an Personal zur Pflege und zur sozialen Betreuung dieser Menschen könnte aus Kostengründen zu einem schweren gesellschaftlichen Problem führen. Eine Lösungsmöglichkeit stellen entsprechende Roboter dar, die es älteren Menschen ermöglichen, dass sie zuhause in ihrer gewohnten Umgebung (wie z.B. in Abbildung 1.1 dargestellt) eigenständig leben können.

Abbildung 1.1.: Ein Roboter beim Servieren eines Getränks (aus [Gra04]).

Eine Voraussetzung für den breiten Einsatz von Robotern im Privatbereich ist, dass sie möglichst natürlich und wenig technisch wirken. Dies bedeutet, dass sie einfach und intuitiv zu bedienen sein müssen [Aga00]. Aus diesem Grund ist die Erforschung fortgeschrittener Schnittstellen zur Mensch-Roboter-Interaktion für solche persönlichen Roboter unerlässlich.

Aktuellste Forschungsaktivitäten im Bereich der persönlichen Robotik streben eine Verknüpfung von Fähigkeiten zur Autonomie und zur Interaktion an. Ein entsprechender Roboter soll sowohl Aufgaben autonom ausführen als auch mit Menschen interagieren können. Die Interaktion zwischen dem Menschen und dem Roboter dient dabei nicht nur zur Instruktion des Systems, sondern auch zur Unterhaltung des Menschen. Allerdings ist die Voraussetzung für entsprechende Interaktionsfähigkeiten, dass der Roboter Menschen wahrnehmen und fokussieren kann. Die Realisierung dieser Eigenschaften ist ein Schwerpunkt in dieser Arbeit.

Der Bedarf an fortgeschrittenen Interaktionsfähigkeiten begründet sich damit, dass sich die persönliche Robotik an Menschen richtet, die nicht zwingend auf die Hilfe solcher Systeme angewiesen sind. Daher ist es entscheidend, dass ein Benutzer einen solchen Roboter „mag“ [Dau04], um ihn gewissermaßen als eine Art Partner bei sich zuhause akzeptieren zu können. Folglich muss die Interaktion mit dem Roboter möglichst intuitiv, angenehm und natürlich wirken. Dazu sind eine Menge von entsprechenden Verhaltensweisen in das Gesamtsystem zu integrieren. Außerdem müssen zusätzlich umfangreiche funktionale Fähigkeiten im Robotersystem untergebracht werden. Diese werden unter anderem für die Realisierung einer gewissen Autonomie benötigt, so dass der Roboter möglichst universell einsetzbar ist. Die Entwicklung einer Gesamtarchitektur, in der all diese Funktionalitäten integriert werden können, stellt den anderen Schwerpunkt dieser Arbeit dar.

1.1. Natürliche Interaktion mit Robotern

Die Fähigkeit zur natürlichen, d.h. menschenähnlichen Interaktion ist die Voraussetzung, um es potentiellen Besitzern komplexer Robotersysteme zu ermöglichen, alle vorhandenen Fähigkeiten effektiv nutzen zu können. Der Hauptgrund dafür ist, dass nicht jeder Mensch, der einen solchen Roboter besitzen möchte, ein Experte für Robotik ist. Folglich muss der Roboter über grundlegende soziale Fähigkeiten verfügen, welche die Basis für eine natürliche Interaktion darstellen, damit ein solches System von einem Benutzer einfacher akzeptiert wird.

Auch wenn bereits seit längerer Zeit humanoide Roboter erforscht werden und in Bezug auf die Mechatronik bereits sehr leistungsfähige humanoide Systeme existieren (siehe z.B. [Has02, Hir98, Kan04, Lim00, Tev00]), so stellt die Integration sozialer Fähigkeiten noch immer ein großes Problem dar. Wie bereits erwähnt, ist eine Bedingung für die Akzeptanz von Robotern für den persönlichen Einsatz, dass sie durch einen Benutzer leicht bedienbar sind. Um eine entsprechend einfache Benutzung zu ermöglichen, sollte der Roboter ein für den Menschen nachvollziehbares Gesamtverhalten aufweisen. Je besser es gelingt, den Roboter mit Verhaltensweisen auszustatten, die ein Benutzer auch von einem menschlichen Interaktionspartner erwarten

würde, desto weniger Berührungsängste und Zweifel wird er gegenüber einem solchen System haben. Nach Fong et al. [Fon03a] muss der Roboter dazu gewisse soziale Kompetenzen aufweisen. In diesem Kontext definieren Fong et al. was unter solchen *„Socially Interactive Robots“* zu verstehen ist wie folgt:

> *Socially Interactive Robots* sind verkörperte Agenten, die einen Teil einer heterogenen Gruppe, d.h. einer Gesellschaft von Robotern oder Menschen, darstellen. Sie sind in der Lage, sich gegenseitig wahrzunehmen und in soziale Interaktion zueinander zu treten. Sie besitzen ein Gedächtnis, so dass sie basierend auf ihrer eigenen Erfahrung die Umwelt wahrnehmen und interpretieren können. Außerdem kommunizieren sie auf explizite Weise miteinander und sind in der Lage voneinander zu lernen (übersetzt aus [Fon03a]).

Als besonders wichtig werden also Fähigkeiten zum sozialen Lernen gesehen. Um diese Eigenschaft zu realisieren, sind Kommunikationsfähigkeiten basierend auf Gesten und natürlicher Sprache in einem Robotersystem zu integrieren. Für die Umsetzung des Gedächtnisses muss die Software-Architektur des Roboters die durch die Kommunikation akquirierten Informationen speichern und auf Abruf auch wieder bereitstellen können. Beispielsweise kann dieser Mechanismus dazu genutzt werden, bereits bekannte Interaktionspartner wieder zu erkennen. Gleichzeitig sollten auch Emotionen berücksichtigt werden, da sie die Basis zur Realisierung einer Persönlichkeit für den Roboter darstellen.

Um zu ermitteln, welche weiteren sozialen Faktoren für eine anspruchsvolle Interaktion mit einem Roboter wichtig sind, hat Kidd [Kid03] entsprechende Benutzerexperimente durchgeführt und die Antworten der Testpersonen ausgewertet. Einzelne Aspekte solcher Evaluationen können ebenfalls dazu beitragen, Roboter zu bauen, die auf effektive Weise mit Menschen interagieren und in einer breiten Palette von Anwendungen eingesetzt werden können.

1.2. Robot Companions: Roboter für Zuhause

Die Entwicklung von fortgeschrittenen Mensch-Roboter-Schnittstellen, die über eine einfache dialogbasierte Interaktion zwischen Benutzer und Roboter hinausgehen, stellt eine große Herausforderung dar. Aufgrund der Beschaffenheit von mobilen Systemen ist es außerdem notwendig, dass nur Sensoren verwendet werden, die an Bord des Roboters mitgeführt werden können, um eine Mensch-Roboter-Interaktion zu realisieren. Zusätzlich müssen die Messtechniken unaufdringlich sein. Dies bedeutet, dass es einem Menschen erlaubt sein muss, mit dem Roboter zu interagieren, ohne dabei spezielle Gerätschaften tragen zu müssen, wie z.B. Marker oder farbige Handschuhe. Solche Hilfsmittel würden die Interaktion wenig natürlich gestalten.

Erst durch die Berücksichtigung der zuvor genannten Voraussetzungen für eine natürliche Interaktion ist es vorstellbar, dass ein Roboter in der Zukunft möglicherweise zu einer Art Kameraden für einen Menschen, d.h. zu einem so genannten *Robot Companion*, werden kann. Folglich darf

es nicht sein, dass ein solcher Roboter wie ein Computer programmiert oder konfiguriert werden muss. Ein *Robot Companion* muss ein aktives System darstellen, welches selbstständig in der Lage ist, Modelle seiner Umgebung aufzubauen. Außerdem muss dieser Roboter mit Fähigkeiten ausgestattet sein, die es ihm erlauben, vom Benutzer instruiert zu werden und neue Fähigkeiten zu lernen. Somit kann sich das Robotersystem nicht nur an eine individuelle Umgebung, sondern auch an die Lebensgewohnheiten des Besitzers anpassen, um seinen Ansprüchen gerecht zu werden. Mit der Phase des Lernens einer neuen Umgebung beschäftigt sich z.B. das so genannte *Home Tour Scenario*.

Die Grundidee des *Home Tour Scenarios* ist es, dass ein Benutzer einen *Robot Companion* in einem entsprechenden Geschäft erwirbt, ihn zu sich nach Hause bringt, dort auspackt und zum ersten Mal aktiviert. Anschließend beginnt der Benutzer damit, dem Roboter alle Räume und Gegenstände beizubringen, die für eine spätere Interaktion relevant sind. Dazu gibt der Mensch dem *Robot Companion* sprachliche Informationen und zeigt gegebenenfalls auf entsprechende Objekte. Der Roboter kann dabei einen Dialog mit dem Benutzer führen, falls Informationen fehlen oder mehrdeutig sind. Der *Robot Companion* lernt auf diese Weise die private Wohnung von Grund auf kennen und baut ein Verständnis für diese Umgebung und die darin enthaltenen Gegenstände auf. Sobald der *Robot Companion* allerdings hinreichend in seine neue Umgebung eingeführt wurde, ist er in der Lage, Aufgaben autonom auszuführen.

Das Lernen neuer Umgebungen durch die Interaktion mit einem Menschen stellt allerdings eine große Herausforderungen dar. Zum einen muss der *Robot Companion* über entsprechende Interaktionsfähigkeiten verfügen. Zum anderen muss es sich bei dem Lernvorgang um einen kontinuierlichen, niemals endenden Prozess handeln, damit der Roboter eine hinreichende Lernfähigkeit erreichen kann. Die Art dieses Lernvorgangs wird üblicherweise als *„open-ended“* bezeichnet. Dies bedeutet, dass der Roboter solange lernt, wie er neuen Situationen gegenübersteht. Die Offenheit bei diesem Prozess ist dabei allerdings nicht nur als quantitativ, sondern auch als qualitativ zu verstehen. Beispielsweise muss der Roboter nicht nur beliebig viele Instanzen gewisser Objekttypen lernen können, sondern auch in der Lage sein, Repräsentationen für neuartige bzw. unbekannte Objekte aufzubauen. Dies ist notwendig, da dem Roboter nicht alle Gegenstände, denen er jemals begegnen mag, im Voraus bekannt sein können.

Nachdem ein *Robot Companion* im Rahmen des *Home Tour Scenarios* instruiert wurde, können die gelernten Informationen schließlich für unterschiedlichste Aufgaben genutzt werden. Einige potentielle Aufgaben wären z.B. die folgenden:

- Suchaufgaben (wie z.B. das Finden von Schlüsseln)
- Transportaufgaben (wie z.B. das Bringen von Kaffee [Kri01])
- Überwachungsaufgaben (wie z.B. das Leeren eines Aschenbechers, wenn er voll ist)
- Routineaufgaben (wie z.B. das regelmäßige Gießen von Pflanzen [Sch99])
- Kooperationsaufgaben (wie z.B. das gemeinsame Tragen von Objekten [Asf00, Kan04])

Insgesamt ist es also das Ziel, dass ein *Robot Companion* täglich zahlreiche Service-Aufgaben in einer privaten Wohnung erfüllen kann. Da eine solche Umgebung unstrukturiert und dynamisch ist, muss der Roboter entsprechend flexibel sein und sich anpassen können. Die Voraussetzung, um entsprechende Informationen zu erhalten, stellt die Fähigkeit zur natürlichen Interaktion mit einem Menschen dar. Dabei ist insbesondere eine aktive Wahrnehmung der Umwelt notwendig. Beispielsweise müssen Personen selbstständig erfasst und identifiziert werden können. Außerdem muss das System robust sein, da einerseits Menschen nicht zu Schaden kommen dürfen und andererseits der Roboter zuverlässig sein muss, um von Menschen als nützlich akzeptiert zu werden. Des Weiteren sollen seine Kommunikations- und Interaktionsfähigkeiten auch zur Unterhaltung von Menschen dienen können. Das ultimative Ziel bei der Entwicklung eines *Robot Companion* ist es also, einen Roboter zu schaffen, der nicht nur helfen, sondern auch unterhalten und Gesellschaft leisten kann. Durch diese Flexibilität eignet sich ein *Robot Companion* folglich auch zur Betreuung von älteren Menschen.

1.3. Zielsetzung der Arbeit

Im Rahmen dieser Arbeit ist eine Software-Architektur für einen *Robot Companion* zu entwerfen und zu realisieren, die es einem Menschen ermöglicht, auf intuitive, natürliche und angenehme Weise mit dem Roboter zu interagieren. Als Grundlage zur Mensch-Roboter-Interaktion dient die Verarbeitung zahlreicher Modalitäten, um unterschiedliche Personen möglichst robust wahrnehmen zu können. Die Realisierung eines solchen Personen-Trackings stellt die Voraussetzung für eine effektive Interaktion zwischen dem Roboter und einem menschlichen Benutzer dar. Allerdings muss das Robotersystem dazu noch mit weiteren entsprechenden Verhaltensweisen ausgestattet werden. Zum Beispiel muss der Roboter einer Person zu einem bestimmten Ort folgen können. Es wäre sehr unnatürlich, wenn der Mensch den Roboter stattdessen mit Kommandos wie z.B. „Fahre gerade aus“ oder „Fahre links“ durch seine Wohnung manövrieren müsste. Außerdem muss der Roboter allen anwesenden Personen seine Absichten anzeigen können, damit diese in der Lage sind, das Verhalten des Systems zu verstehen. Um diese Fähigkeiten umzusetzen, ist unter anderem eine spezielle Aufmerksamkeitssteuerung für Personen im Gesamtsystem zu integrieren.

Zur Realisierung der natürlichen Interaktion zwischen Mensch und Roboter wird folglich eine Vielzahl an Modulen benötigt. Um die Integration dieser Module und der darin enthaltenen Funktionalitäten auf einfache Weise zu ermöglichen, ist die Entwicklung einer entsprechenden Gesamtarchitektur notwendig. Diese Architektur soll außerdem den Entwicklungsprozess des *Robot Companions* unterstützen und die Möglichkeit bieten, das Robotersystem nachträglich zu modifizieren und um zusätzliche Fähigkeiten zu erweitern. Da die zu entwickelnde Architektur auf einem realen Roboter zum Einsatz kommen soll, müssen auch entsprechende Module zur Ansteuerung der Hardware integriert werden. Erst durch den Entwurf einer Architektur, welche diese Aspekte berücksichtigt, ist es möglich, ein sinnvolles und lauffähiges Gesamtsystem zu entwickeln.

1.4. Gliederung der Arbeit

Im Folgenden wird in Kapitel 2 zuerst die Entwicklung im Bereich der Robotik zusammenfassend dargestellt. Anschließend wird eine Einführung in Architekturen für mobile Robotersysteme gegeben. Dabei werden die in den letzten Jahrzehnten zur Kontrolle von Robotern entwickelten Paradigmen ausführlich beschrieben.

Anschließend werden in Kapitel 3 Anforderungen an eine zu entwickelnde Architektur für einen *Robot Companion* formuliert. Der Schwerpunkt liegt dabei einerseits auf den gewünschten Interaktionsfähigkeiten des Zielsystems und andererseits auf strukturellen Notwendigkeiten, um diese Fähigkeiten entwickeln und ausbauen zu können. Basierend auf diesen Informationen und den Erkenntnissen aus dem 2. Kapitel wird schließlich ein entsprechendes Architekturkonzept vorgestellt.

Danach wird in Kapitel 4 der mobile Roboter BIRON, für den das im vorhergehenden Kapitel vorgestellte allgemeine Architekturkonzept eines *Robot Companions* umgesetzt werden soll, präsentiert.

In Kapitel 5 wird die ebenfalls für BIRON entwickelte Personenerkennung erläutert. Diese Komponente stellt für den Roboter die Grundlage zur Interaktion mit menschlichen Benutzern dar.

Weitere, zur Interaktion benötigte und in das Gesamtsystem zu integrierende Module werden in Kapitel 6 beschrieben. Da diese Module im Rahmen anderer Promotionsprojekte in der AG „Angewandte Informatik“ entwickelt wurden, fällt ihre Darstellung etwas weniger ausführlich aus. Es handelt sich dabei um die beiden Aufmerksamkeitssteuerungen für Personen und Objekte und um die Dialogsteuerung.

Die Integration aller zuvor präsentierten Komponenten in eine Gesamtarchitektur wird in Kapitel 7 beschrieben. Dazu wird das in Abschnitt 3.3 vorgestellte Architekturkonzept für *Robot Companions* auf BIRON umgesetzt. Dabei wird insbesondere auf die Implementierung der zentralen Komponente der Architektur, dem so genannten *Execution Supervisor*, eingegangen. Außerdem wird die resultierende Funktionalität des Gesamtsystems erläutert.

Kapitel 8 beschäftigt sich mit der Evaluation von BIRON. Hier werden Ergebnisse von verschiedenen Experimenten mit dem Roboter präsentiert. Die Experimente richten sich zum einen an das Personen-Tracking und zum anderen an die Interaktionsfähigkeiten des Gesamtsystems. Zusätzlich wird der erfolgreiche Einsatz des Systems auf einer Messe beschrieben.

Die Arbeit schließt mit einer Zusammenfassung in Kapitel 9.

2. Stand der Forschung

In diesem Kapitel wird zunächst kurz die Entwicklung im Bereich der Robotik zusammengefasst. Diese zeigt, dass in den letzten Jahren, ausgehend von der industriellen Anwendung, immer fortgeschrittenere Robotersysteme entwickelt wurden, von denen die neusten Exemplare bereits sogar Einzug in das private Wohnzimmer gefunden haben. Da jene Systeme zur gleichen Zeit stetig komplexer geworden sind, wurden spezielle Kontrollparadigmen entwickelt, um diese Komplexität beherrschen zu können. Die Kontrollparadigmen bilden wiederum die Basis entsprechender Software-Architekturen zur Steuerung autonomer Robotersysteme und werden daher im Anschluss dargestellt.

2.1. Bisherige Entwicklungsschritte in der Robotik

Erfreulicherweise hat die Entwicklung in der Robotik (siehe Abb. 2.1) über die letzten Jahrzehnte zu einer gestiegenen Verfügbarkeit von mobilen Roboterplattformen mit guten Navigationsfähigkeiten geführt, welche die Basis für die Erforschung hoch entwickelter Mensch-Roboter-Schnittstellen darstellen. Ausgehend von der industriellen Robotik, über den Bereich der Service-Robotik, hat sich die persönliche Robotik entwickelt.

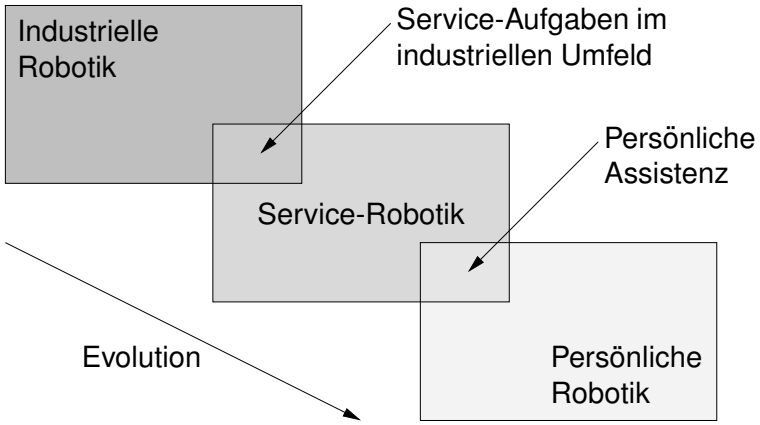


Abbildung 2.1.: Ein Schema der Evolution in der Robotik mit drei Hauptbereichen als grundlegende Evolutionsschritte (aus [Gug99]).

Beginnend bei der industriellen Robotik liegen klassischen Anwendungen üblicherweise Umgebungen zugrunde, welche basierend auf den Fähigkeiten des Roboters und den Anforderungen der Anwendung entsprechend strukturiert sind. Industrieroboter werden durch einen Fachmann programmiert und es ist in der Regel keine weitere Interaktion mit Menschen vorgesehen. Somit entsprechen diese Roboter vielmehr komplexen Werkzeugen und dienen speziellen Aufgaben in speziellen Umgebungen, wie z.B. in Montage-Straßen für Personenkraftwagen. In anspruchsvolleren Service-Anwendungen kann die Umgebung aber auch teilweise unstrukturiert sein. Bei Service-Aufgaben im industriellen Umfeld handelt es sich dabei zumeist um Transport- oder Überwachungsaufgaben. Zum Bereich der Überwachung zählen auch spezielle Sicherheitsroboter, die allerdings in der Regel teleoperiert sind (siehe z.B. [Cic99]). Eine weitere Art von Robotern für den industriellen Einsatz stellen so genannte Fabrikationsassistenten dar, die unter Anleitung eines Instrukteurs beispielsweise Werkstücke auf spezielle Weise bearbeiten können (siehe z.B. [Sto01a]).

Das umfassende Gebiet der Service-Robotik stellt hingegen viel schwerwiegendere technische Anforderungen an die Robotik, da hier üblicherweise flexible operative Fähigkeiten in unstrukturierten Umgebungen gefragt sind. Der Anspruch besteht dabei darin, es mobilen Robotern zu ermöglichen, auch in nicht geschlossenen oder gar unkontrollierten Umgebungen selbstständig zu agieren. Die eigentliche Herausforderung ergibt sich dadurch, dass diese Umgebungen im Allgemeinen auf menschliche Bedürfnisse zugeschnitten sind und selten für Service-Roboter entsprechend strukturiert werden können. Des Weiteren muss berücksichtigt werden, dass in solchen Umgebungen Menschen anwesend sind. Doch selbst wenn Menschen primär als Hindernisse zu interpretieren sind und daher nur selten den Arbeitsbereich eines Service-Roboters teilen sollten, ergibt sich das Problem der Realisierung einer gefahrlosen Interaktion zwischen Mensch und Roboter.

Eine prototypische Anwendung für solche Service-Roboter ist ihre Verwendung als so genannte Tour-Führer in wissenschaftlichen Laboren oder Museen. Solche Systeme sind intensiv erforscht worden und haben den Weg zur Erforschung von komplexeren interaktiven Robotern geebnet. Als Beispiele sind Sage [Nou99], RHINO [Bur99], Minerva [Thr00], RoboX [Tom02] oder Jinny [Kim04a] zu erwähnen. Der Schwerpunkt bei der Entwicklung eines Tour-Führers liegt in der Selbstlokalisation, der Navigation und dem Ausweichen von Hindernissen. Diese Fähigkeiten müssen sehr robust sein, damit einerseits menschlicher Eingriff nur in Ausnahmen nötig ist und andererseits Kollisionen mit Menschen oder anderen Objekten auszuschließen sind. Ein Vorteil bei dieser Anwendung besteht darin, dass die Umgebung bekannt ist und mit künstlichen Markierungen versehen werden kann, wie z.B. in [Nou99]. Ein Tour-Führer muss aber auch Menschen detektieren können, um mit ihnen zu interagieren bzw. um ihnen in Form von multimedialen Präsentationen Wissen zu vermitteln, wie z.B. in [Bur99]. Da die Benutzer eines solchen Roboters in der Regel nur kurz mit dem System interagieren, werden zur einfachen Bedienung fast ausschließlich Eingabeknöpfe verwendet. Die Reaktionen eines Tour-Führers können sowohl akustisch als auch visuell sein. Im letzteren Fall wird oft ein Gesicht auf einem Display dargestellt. Dies hat den Vorteil, dass vom Roboter simulierte Emotionen visualisiert werden können, die sich je nach Verhalten der Besucher ändern. Beispielsweise kann ein Tour-Führer verärgert reagieren, wenn ihm eine Person den Weg versperrt, wie z.B. in [Thr00].

Der Bereich der persönlichen Assistenz beinhaltet spezielle Service-Roboter, die eine individuellere Interaktion zwischen Mensch und Roboter ermöglichen. In der Regel erlauben solche Roboter, dass ein Benutzer vollständige Dialoge mit dem System führen kann. Der Schwerpunkt liegt dabei meist auf der Verarbeitung von Eingaben in natürlicher Sprache. Dadurch ergibt sich nicht nur eine viel komplexere Mensch-Roboter-Interaktion, entsprechende Roboter sind auch intuitiver zu bedienen. Ein Beispiel für einen solchen Roboter ist Perses [Böh03], ein interaktiver mobiler Service-Roboter, der in einem Baumarkt als Einkaufsassistent fungiert. Auch Roboter in Form von Boten für Post, wie z.B. MOPS [Tsc01] oder für Gepäck, wie z.B. MILVA [Böh98] zählen zu dieser Kategorie. Die meisten Roboter aus dem Bereich der persönlichen Assistenz sind allerdings für so genannte *Fetch-And-Carry*-Aufgaben konstruiert. Dies bedeutet, dass sie Gegenstände aufnehmen und an einen bestimmten Ort bringen können. In der Regel verfügen solche Roboter daher zusätzlich über mindestens einen Greifarm, wie z.B. *HERMES* [Bis02a], PSR [Kim03] oder PSR-II [Kim04b].

Die persönliche Robotik leitet sich aus der Service-Robotik ab und stellt somit aktuell den letzten Schritt der in Abbildung 2.1 gezeigten Evolution dar (siehe beispielsweise das BMBF-Leitprojekt MORPHA [Lay01]). In diesem Bereich der Robotik steht der persönliche Gebrauch eines Roboters, der für viele spezifische Aufgaben eingesetzt werden kann, im Vordergrund. Zu dieser Kategorie gehören auch die bereits zu Beginn erwähnten Roboter, die als Hilfe für behinderte und ältere Menschen dienen können. Gerade diese Roboter stellen zur Zeit einen großen Schwerpunkt in der Robotikforschung dar und es sind bereits etliche solcher Systeme entwickelt worden, wie z.B. Do-U-Mi [Par01], Flo [Roy00], Pearl [Mon02], Care-O-bot [Han01] oder Care-O-bot II [Gra04]. Die Aufgaben dieser Roboter sind sowohl kommunikativer als auch sozialer Art und umfassen beispielsweise das Vorlesen von Email, Videotelefonie, die Überwachung des Gesundheitszustands des Besitzers, die interaktive Kommunikation mit ihm und dessen Unterhaltung. Dazu ist eine benutzerfreundliche Schnittstelle nötig, so dass Eingaben in der Regel durch natürliche Sprache oder über ein Touch-Screen-Display erfolgen. Des Weiteren sind diese Roboter in der Lage, einem Menschen beim Gehen zu helfen oder *Fetch-And-Carry*-Aufgaben ausführen, um z.B. das Essen aus der Küche holen zu können oder einen Tisch abzuräumen.

Neben den gerade genannten Systemen, die durchaus eine Notwendigkeit für Menschen darstellen können, gehören auch Roboter, die allein zur Unterhaltung dienen, zu dem Bereich der persönlichen Robotik. Den Großteil solcher Systeme stellen Spielzeugroboter, wie z.B. Sony's AIBO®, Hasbro's My-Real-Baby® oder Nec's PaPeRo®, dar [Bar01]. Da ihr Schwerpunkt auf der Interaktion mit Menschen beruht, ist neben dem Verstehen und Erzeugen von natürlicher Sprache für diese Systeme auch das Wahrnehmen und Äußern von Emotionen sehr wichtig. Spielzeugähnliche Systeme eignen sich daher auch für therapeutische Maßnahmen [Shi03]. Des Weiteren dienen solche Systeme, wie z.B. AIBO® [Kap00], zur Erforschung fortgeschrittener Mensch-Roboter-Schnittstellen, die ebenfalls für den aufgabenorientierten Einsatz in privaten Haushalten gedacht sind. Für solche Umgebungen werden auch spezielle Robotersysteme erforscht, die mit weiteren technischen Geräten im Haushalt in Verbindung stehen, um diese ansteuern zu können. Diese Roboter stellen somit die Schnittstelle für ein so genanntes „intelligentes Zuhause" dar. Beispiele solcher *Robotic User Interfaces* sind durch die Roboter Lino [Krö03] und iCat [vB04] gegeben.

Ein Forschungsbereich, der sich neben der persönlichen Robotik ebenfalls mit der Realisierung fortgeschrittener Mensch-Maschine-Schnittstellen beschäftigt, basiert auf dem Studium von so genannten „virtuellen Agenten“. Virtuelle Agenten haben gegenüber realen Robotern den Vorteil, dass sie nur innerhalb eines Computersystems existieren und auf einem Monitor visualisiert werden. Da es sich dabei in der Regel um fest installierte Systeme handelt, besteht keine Beschränkung in der Wahl der Sensoren, wie es bei einem mobilen Roboter der Fall ist. Außerdem müssen keine Aktuatoren angesteuert werden. Der Schwerpunkt bei der Realisierung solcher Systeme liegt allein im Führen von anspruchsvollen multimodalen Dialogen mit einem Benutzer. Bisher sind eine ganze Reihe von beeindruckenden virtuellen Systemen entwickelt worden, wie z.B. Gandalf [Thó02], der Fragen über unser Sonnensystem beantworten kann. REA [Cas00] übernimmt hingegen die Rolle einer Immobilienmaklerin und interagiert mit einem Benutzer, um dessen Bedürfnisse zu ermitteln. Sie führt ihn in virtuellen Immobilien herum und versucht dem Benutzer schließlich ein Haus zu verkaufen. Ein weiteres Beispiel ist Max [Kop03], ein anthropomorpher künstlicher Agent, der einem Benutzer bei virtuellen Konstruktionsaufgaben mit baufix®, einem Konstruktionsspiel aus Holzbauteilen, assistieren kann.

Die oben genannten Ansätze sind gewöhnlich kognitionswissenschaftlich motiviert. Dabei geht häufig mit dem Führen von multimodalen Dialogen die Abstimmung von verbalen und nicht-verbalen Verhaltensweisen einher. Zur Realisierung solcher Systeme werden oft Implementierungen des so genannten *Belief-Desire-Intention*-Modells (BDI) verwendet, wie z.B. JAM [Hub99], die dazu dienen, menschliche Denkprozesse nachzubilden. In Bezug auf reale Robotersysteme beeinflussen solche Ansätze für virtuelle Agenten allerdings nicht direkt den Entwurf einer entsprechenden Gesamtarchitektur, da sie in der Regel Punkte, wie z.B. Reaktivität und unbrauchbare Sensordaten, nicht berücksichtigen müssen. Sie stellen eher eine mögliche Grundlage für ein Interaktionsmodul in einem Robotersystem dar. Dennoch sind Erkenntnisse aus Evaluationen solcher Systeme wichtig für die Realisierung eines *Robot Companions*, da auch dieser gewisse kognitive Fähigkeiten zur Realisierung eines natürlichen Gesamtverhaltens aufweisen muss.

2.2. Architekturen für mobile Robotersysteme

Der Begriff „Architektur“ bezüglich eines Robotersystems verweist in erster Linie auf das Software- oder Hardware-Gerüst, das zur Ansteuerung des Roboters dient. In dieser Arbeit bezieht sich der Begriff durchgehend auf die Software-Architektur eines Roboters. Unter der Software-Architektur eines Robotersystems ist die Art der Organisation der im System enthaltenen Software-Komponenten zu verstehen. Da innerhalb des Gesamtsystem zwischen gewissen Komponenten auch Daten ausgetauscht müssen, sind entsprechende Werkzeuge zur Modulkommunikation ebenfalls ein Bestandteil einer Architektur.

Die Notwendigkeit zur Entwicklung einer ausgefeilten Roboterarchitektur begründet sich in der ständig steigenden Verarbeitungsgeschwindigkeit von integrierbaren Rechnersystemen. Schnellere Rechner ermöglichen die Konstruktion von immer komplexeren Robotersystemen, welche stetig mehr Fähigkeiten integrieren können. Zu diesen Fähigkeiten zählt auch die natürliche Inter-

aktion mit Menschen. Gerade weil Robotersysteme immer komplexer werden, ist es wichtig, dass sie möglichst intuitiv zu bedienen sind, um Menschen den einfachen Gebrauch eines Roboters zu erlauben. Außerdem sind stetig mehr Sensoren zu integrieren und Aktuatoren anzusteuern, von denen die meisten, insbesondere bei humanoiden Robotern [Has02, Hir98, Kan04, Lim00, Tev00], viele Freiheitsgrade besitzen. In solchen Robotern existieren einerseits Komponenten, die harten Realzeitbedingungen unterliegen, wie z.B. motorische Systeme für Arme und Beine. Andererseits gibt es Komponenten, die keine Realzeitgrenzen einhalten können. Diese finden sich in der Regel auf höheren Verarbeitungsebenen wieder und führen komplexe Berechnungen zur Planung von Aktionen durch. All diese Komponenten müssen für ein korrektes Gesamtverhalten eines Roboters in Einklang gebracht werden. Als Folge davon wird die Entwicklung eines gesamten Robotersystems immer anspruchsvoller.

Entsprechende Architekturformen sind allerdings oftmals spezifisch bezüglich ihrer Einsatzdomäne sowie den zugehörigen Aufgaben, und sind daher in der Regel unangemessen für eine breite Spanne von Anwendungen. Zum Beispiel ist eine spezielle Architekturform, die sehr geeignet für direkte Teleoperation ist, nicht unbedingt angemessen für autonome Zwecke. Generell ist bei der Wahl einer geeigneten Architekturform Folgendes zu beachten: Je übersichtlicher die Organisation eines Systems mit einer gewissen Architekturform ausfällt, desto spezifischer, aber auch einfacher zu entwickeln, ist folglich das Endsystem. Allerdings werden dadurch gleichzeitig auch die Möglichkeiten beschränkt, mit denen bestimmte Probleme gelöst werden können.

Aus der Sichtweise der Entwickler eines Robotersystems muss eine Architektur eine entsprechende Struktur vorgeben, um die Konstruktion des Gesamtsystems auf technischer Ebene beherrschen zu können. Eine Gliederung in Subsysteme und Komponenten ermöglicht einen modularen Aufbau, welcher wiederum die Grundlage für ein wartbares und erweiterbares Gesamtsystem darstellt. Allerdings müssen dazu entsprechende Dienste von der Architektur den Subsystemen und Komponenten zur Verfügung gestellt werden, wie beispielsweise Werkzeuge zur Kommunikation, um verteilte Berechnungen zu ermöglichen.

Aus der Sichtweise des Systems besteht die Aufgabe einer Roboterarchitektur darin, die Kontrolle eines Roboters zu strukturieren. Diese Aufgabe gliedert sich in die in die Fragestellungen, wie Informationen über die Umgebung mit Hilfe der Sensoren des Roboters zu akquirieren sind; wie gesammelte Informationen nach Bedarf zu verarbeiten sind, um entscheiden zu können, wie gehandelt werden soll; und schließlich wie Aktionen in der Umgebung auszuführen sind.

Dieses Kapitel beschreibt die wichtigsten Ansätze zur Strukturierung der Kontrolle in einem Robotersystem. Die Angemessenheit und Leistungsfähigkeit eines solchen Paradigmas zur Steuerung eines mobilen Roboters hängt dabei allerdings primär vom Einsatzzweck des Roboters ab. Die vier Hauptparadigmen, die in den letzten Jahrzehnten entwickelt worden sind, werden als deliberative, reaktive, hybride und Behavior-basierte Kontrolle bezeichnet. Matarić fasst diese Kontrollparadigmen wie folgt prägnant zusammen [Mat01a]:

- Deliberative Kontrolle: *„Think hard, then act.“*
- Reaktive Kontrolle: *„Don’t think, act.“*

- Hybride Kontrolle: *„Think and act independently, in parallel.“*
- Behavior-basierte Kontrolle: *„Think the way you act.“*

Eine Einführung in diese verschiedenen Paradigmen für autonome mobile Roboter wird in den folgenden Abschnitten gegeben. Eine kurze Zusammenfassung über die historische Entstehung der Kontrollparadigmen ist in [Mat98] zu finden.

2.2.1. Deliberative Ansätze

Deliberative Kontrolle, welche auch gelegentlich als „planungsbasierte“ Kontrolle bezeichnet wird, macht Gebrauch von der so genannten *„Sense-Plan-Act“*-Zerlegung (siehe Abb. 2.2), um ein intelligentes System zu begründen [Nil82]. Dieses Paradigma entwickelte sich aus der traditionellen künstlichen Intelligenz (KI) und stützt sich auf eine zentrale, robuste und aktionsunabhängige Repräsentation der Umgebung. Dieses Weltmodell bildet die Basis für die Integration von Sensorinformationen und der Generierung von in der Welt auszuführenden Aktionen. Das wohl am besten bekannte Beispiel für einen mit deliberativer Kontrolle ausgestatteten Roboter stellt Shakey [Nil84] dar.

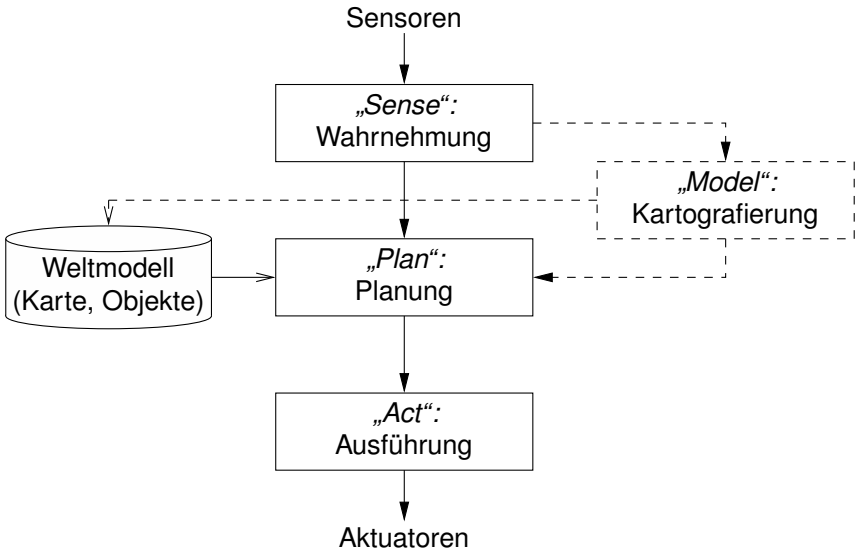


Abbildung 2.2.: Konzeptdiagramm einer deliberativen Kontrollarchitektur nach dem Schema *Sense-Plan-Act* bzw. *Sense-Model-Plan-Act* (gestrichelt).

In einem System mit deliberativer Kontrolle werden die Schritte *„Sense“*, *„Plan“* und *„Act“* sequentiell ausgeführt (siehe Abb. 2.2). In dem ersten Schritt nimmt ein entsprechend gesteuerter Roboter alle vorhandenen Sensorinformationen auf. Diese Informationen werden dann mit dem intern gespeicherten Wissen verglichen, welches vom Entwickler des Systems fest vorgegeben ist. Ziel dieses Vergleichs ist es, zu entscheiden, welche Aktionen vom Roboter auszuführen sind.

Diese Entscheidung wird üblicherweise durch so genanntes „Planen“ getroffen. Dessen Aufgabe ist es, verschiedenste Sequenzen von Aktionen, so genannte „Aktionspläne“, zu analysieren, um die viel versprechendste Sequenz zum Erreichen eines spezifischen Ziels zu finden. Schließlich werden die einzelnen Aktionen des aussichtsreichsten Aktionsplans ausgeführt, um das entsprechende Ziel zu erreichen.

Der Nachteil des *Sense-Plan-Act*-Schemas ergibt sich durch die Verwendung eines statischen Weltmodells. Falls dieses nicht exakt mit der realen Welt übereinstimmen sollte, sind Fehler im Verhalten des Roboters vorprogrammiert. Eine Erweiterung des oben angeführten Vorgehens verwendet daher die so genannte *„Sense-Model-Plan-Act“*-Zerlegung (siehe z.B. [Bro91]). In diesem Fall wird zwischen der Akquisition von Sensordaten und der Planung ein zusätzlicher Schritt eingefügt (siehe Abb. 2.2). In diesem Schritt wird die Umgebung des Roboters analysiert und gegebenenfalls, beim Auftreten von Inkonsistenzen zwischen wahrgenommenen Informationen und gespeichertem Wissen, das Weltmodell aktualisiert. Der Vorteil dieser Vorgehensweise ist, dass der Roboter nun dynamische Veränderungen an seiner Umwelt wahrnehmen und berücksichtigen kann. Im optimalen Fall müsste der Roboter nicht einmal mit einem initialen Weltmodell ausgestattet werden, sondern könnte sich dieses komplett selbst erstellen. Allerdings wäre der Roboter dabei natürlich nicht sofort einsatzbereit, da er zuerst die für ihn unbekannte Umgebung erfassen müsste. Außerdem ist das gleichzeitige Kartografieren und Lokalisieren (genannt: SLAM – *„Simultaneous Localization And Mapping“*) keine einfache Aufgabe und daher immer noch ein aktives Forschungsgebiet (siehe z.B. [Lis03]).

Architekturen mit deliberativer Kontrolle besitzen zumeist eine hierarchische Struktur, in welcher der Kontrollfluss und die Kommunikation festen Mustern unterliegen. Beispielsweise findet der Austausch entsprechender Informationen zwar zwischen den einzelnen Ebenen statt, nicht aber zwischen den Komponenten innerhalb einer Ebene. In einer solchen Hierarchie versorgen Module von höheren Ebenen üblicherweise Module aus darunter gelegenen Ebenen mit entsprechenden Teilzielen [Ark98].

Der Kernpunkt deliberativer Architekturen stellt das Weltmodell dar. Falls das Weltmodell hinreichende Informationen enthält und dem Roboter ausreichend Zeit zur Verfügung steht, um Pläne zu generieren, so ist ein deliberativer Ansatz eine sehr effektive Methode, um strategische Aktionen hervorzubringen. In diesem Fall ist es dem Roboter möglich, eine Aufgabe erfolgreich und oftmals sogar auf optimale Weise zu erfüllen.

Dennoch ist der Suchraum für Systeme, die in der realen Welt operieren, wie es bei mobilen Robotern der Fall ist, typischerweise groß und in der Regel nur partiell wahrnehmbar, so dass der Prozess der Planung sehr rechenintensiv ausfallen kann. Da Sensoren gewissen Einschränkungen unterliegen, sind Informationen über die Umwelt des Roboters in der Regel verrauscht und somit unsicher. Dies kann es dem Roboter erschweren, seine Ziele zu erreichen. Wenn zusätzlich dynamische Veränderungen in der Umgebung berücksichtigt werden müssen, so kann es daher vorkommen, dass eine Neuplanung sehr oft vorgenommen werden muss. Des Weiteren sind deliberative Ansätze wegen ihrer schlechten Skalierbarkeit kritisiert worden (siehe z.B. [Bro90, Bro91]). Je komplexer ein zu lösendes Problem ausfällt, um so schwieriger ist es, einen Roboter in Realzeit reagieren zu lassen.

2.2.2. Reaktive Ansätze

Während deliberative Ansätze ein hohes Maß an Intelligenz durch die Fähigkeit zur Planung mitbringen können, so versagen sie jedoch in inkorrekt repräsentierten oder unbekannten Umgebungen. Dabei stellt das zentrale Problem die Verwendung eines globalen Weltmodells dar [Gra98]. Sollte die Abbildung von aktuellen Sensordaten auf das Weltmodell fehlschlagen, so scheitert die gesamte Steuerung des Systems. Aus diesem Grund wurden reaktive Kontrollsysteme vorschlagen, welche kein Weltmodell verwenden und sich nur auf aktuelle Sensordaten stützen; treu nach der Devise: „*The world is its own best model*“ [Bro90].

Die Abkehr vom *Sense-Plan-Act*-Paradigma zu reaktiven Systemen ist oft inspiriert bzw. motiviert durch das Interesse daran, wie sich Intelligenz in biologischen Systemen manifestiert. Entsprechende Ansätze implementieren in der Regel diverse zielorientierte Verhaltensweisen, welche auf einem Roboter in Form von Modulen parallel ausgeführt werden (siehe Abb. 2.3). Als Resultat ergeben sich für das reaktive Paradigma direktere Verbindungen zwischen Sensoren und Aktuatoren [Ark98]. Dieses führt allerdings zu einem Komplexitätsverlust, da die Fähigkeit zur deliberativen Planung zu Gunsten einer schnelleren Reaktionszeit weicht. Eine schnellere Reaktionfähigkeit ermöglicht es dem Roboter aber, in kürzester Zeit zu handeln und auf eine sich verändernde und unstrukturierte Umgebung auf angemessene Weise zu reagieren.

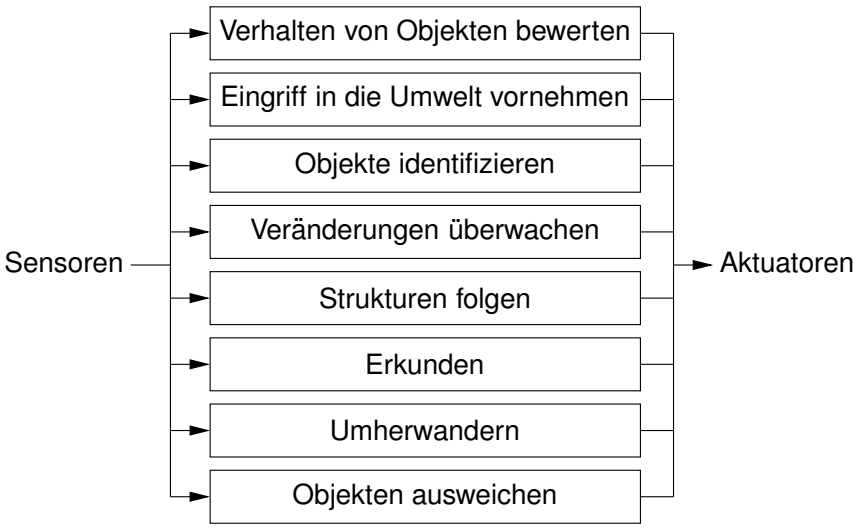


Abbildung 2.3.: Konzeptdiagramm einer reaktiven Kontrollarchitektur [Bro86].

Da rein reaktive Systeme keinerlei interne Repräsentation der Umgebung verwenden, führen sie auch keine Suche durch, um Lösungen bzw. Aktionspläne zu finden. Sie reagieren einzig und allein auf die aktuell vorliegenden Sensorinformationen. Dabei arbeiten diese Systeme gewöhnlich nach dem biologischen „Stimulus-Antwort-Prinzip“. Dementsprechend werden reaktive Kontrollstrategien üblicherweise dadurch realisiert, dass eine Kollektion von vordefinierten Regeln, so genannten „Bedingungs-Aktions-Paaren“, spezifiziert wird. Diese Regeln bilden bestimmte Sensorinformationen, auch Situationen genannt, auf zugehörige Aktionen ab. Somit stellt die

Ansteuerung reaktiver Systeme eine direkte Abbildung von den Sensoren zu den Aktuatoren des Roboters dar, wobei keine oder nur minimale Zustandsinformationen verwendet werden.

Oftmals ist es nicht möglich alle potentiellen Situationen, in die ein mobiler Roboter geraten kann, auf oben genannte Weise zu repräsentieren oder zumindest wäre der Aufwand dazu unverhältnismäßig hoch. Im Regelfall wird daher keine vollständige Abbildung in handgefertigten reaktiven Systemen verwendet. Gewöhnlich bewirken spezifische Situationen entsprechende Aktionen und alle anderen Fälle, die auftreten können, werden durch Standardaktionen behandelt. Entwickler von reaktiven Kontrollsystemen können somit effektiv den Sensorraum auf wesentliche Eingaben bzw. Situationen beschränken und somit das System stark vereinfachen.

Ein sehr bekanntes Beispiel für eine reaktive Kontrollarchitektur ist durch die *Subsumption*-Architektur gegeben [Bro86]. Die grundlegenden Komponenten stellen dabei spezielle zielorientierte Module dar (vgl. z.B. Abb. 2.3), welche durch endliche Automaten kontrolliert werden. Eine Priorisierung der Module wird dadurch realisiert, dass ein höherwertiges Modul die Operation eines niederwertigeren Moduls steuern kann. Dazu kann das erstere Modul entweder die normale Eingabe des anderen Moduls überschreiben oder die Ausgabe des Moduls unterdrücken, indem sie mit eigenen Werten überschrieben wird. Eine stärkere Form der Hierarchisierung wird erreicht, indem Module in verschiedenen Ebenen, so genannten *„Levels of Competence“*, angeordnet werden (siehe Abb. 2.4). Dabei arbeiten alle Ebenen parallel und können auch auf verschiedenen Prozessoren laufen. Eine inkrementelle Entwicklung des Systems wird dadurch erreicht, dass neue Ebenen über allen bestehenden eingefügt werden können. Jede Ebene kann dabei aus mehreren Modulen bestehen und verkörpert eine einzelne komplexere Fähigkeit, die es dem Roboter z.B. ermöglicht, Hindernissen auszuweichen.

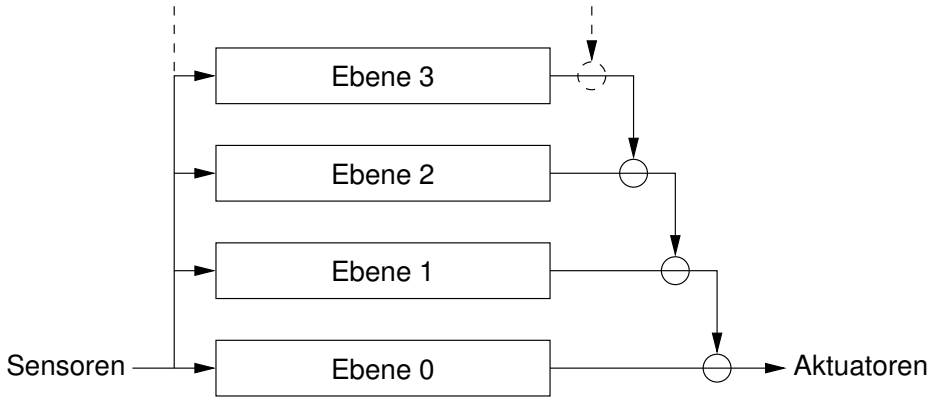


Abbildung 2.4.: In Ebenen unterteilte Kontrolle: Die *Subsumption*-Architektur [Bro86].

Die *Subsumption*-Architektur schränkt den Kontrollfluss im System allerdings stark ein. Die Koordinierung zwischen verschiedenen Modulen geschieht durch eine feste Priorisierung, welche eine kooperative Interaktion zwischen den Modulen ausschließt. Die Architektur ist anfällig dafür, extrem unhandlich zu werden, wenn neue Ebenen einzuführen sind. Außerdem ist es schwierig, sie von Programmierfehlern zu bereinigen. Dies liegt insbesondere daran, dass keine Einschränkungen bestehen, auf welche Weise höhere Ebenen mit tiefer gelegenen Ebenen interagie-

ren dürfen [Mac96]. Somit kann es bei einer Umstrukturierung der untersten Ebene notwendig sein, das gesamte Kontrollsystem neu zu implementieren.

Ein weiteres, beeindruckendes Beispiel für einen reaktiven Roboter ist Polly [Hor93]. Dabei handelt es sich um einen mobilen Roboter, der Führungen in einem Bürogebäude geben kann. Die Architektur von Polly ähnelt der *Subsumption*-Architektur und ermöglicht es dem System in Realzeit zu arbeiten. Allerdings ist der Roboter fest auf eine Umgebung zugeschnitten. Beispielsweise werden bestimmte Eigenheiten der Umgebung für die Navigation ausgenutzt. Zu Beginn nähert sich der Roboter einem Menschen, der sich in der Nähe befindet, wobei dieser anhand seiner Beine erkannt wird. Nachdem sich Polly vorgestellt hat, kann der Mensch durch Bewegen eines Beines eine Tour durch das Gebäude starten. Geht die Person verloren oder entfernt sie sich, sucht der Roboter erneut nach einen Menschen, um seine Dienste anbieten zu können.

Ein anderer Ansatz, der ebenfalls die *Subsumption*-Architektur als Ausgangsbasis hat, ist auf dem Roboter LiSA [Ami01] umgesetzt worden. Hier wurde versucht, ein gewisses Maß an deliberativer Kontrolle in die reaktive Architektur zu integrieren. Dazu wurden mathematische Modelle aus dem Bereich der Logik in zusätzlich eingefügten Ebenen untergebracht. Insgesamt wurde es dem Roboter damit ermöglicht, zu verschiedenen Räumen eines Gebäudes zu navigieren. Allerdings sank auch die Verarbeitungsgeschwindigkeit, so dass das System etwas zu langsam für ein wirklich reaktives System wurde. LiSA verfügt allerdings nicht über die Fähigkeit zur Interaktion mit einem Menschen, da der Schwerpunkt bei diesem System allein die Navigation darstellt.

Falls die Umgebung eines mobilen Roboters schlicht und spezifisch ist, so ist es zur Design-Zeit möglich, die meisten Ereignisse vorherzusagen, die eintreten könnten. Somit können entsprechende Antworten implementiert werden, die den Roboter auf die Weise ansteuern, die es ihm effektiv ermöglicht, seine Ziele zu erreichen. Es wurde gezeigt, dass eine rein reaktive Kontrolle eine Reihe von Problemen lösen kann, die zur Design-Zeit klar definiert sind (siehe z.B. [Mil92]). Allerdings ist eine reaktive Kontrolle nicht in der Lage Annahmen über zukünftige Situationen zu formulieren, da sie zu jedem Zeitpunkt nur den aktuellen Systemzustand berücksichtigt, ohne hypothetische Zustände der Umwelt zu betrachten. Des Weiteren ist es mit einer reaktiven Kontrolle nicht möglich, Ziele oder Sequenzen möglicher Aktionen zu kodieren. Reaktiv gesteuerte Systeme sind außerdem unflexibel zur Laufzeit, da sie keine Informationen dynamisch speichern können [Mat92]. Somit kann auch keine Repräsentation der Umwelt erstellt werden und es besteht für entsprechende Systeme keine Möglichkeit, über die Zeit zu lernen. Insgesamt ist eine rein reaktive Kontrolle nur für Roboter mit rudimentärer Intelligenz angemessen, da sie nicht gut skaliert. Es ist schwierig von einfachen Aufgaben, wie z.B. der Navigation in einem Raum, auf komplexeres Verhalten, wie z.B. einen Plan zu generieren und auszuführen, zu abstrahieren.

2.2.3. Hybride Ansätze

Um die Einschränkungen von deliberativer und reaktiver Kontrolle zu überwinden, entstand die Idee, die besten Aspekte beider Paradigmen in einer so genannten hybriden Kontrolle zu kombinieren. Eine der ersten hybriden Architekturen ist AuRA[Ark86], die einen hierarchischen Planer

und eine Hardware-Ansteuerung basierend auf so genannten „*Motor Schemas*“ [Ark89] verwendet. Im Allgemeinen ist es das Ziel der hybriden Kontrolle, ein Ansprechverhalten in Realzeit zu erreichen und mit der Fähigkeit, komplexe Aufgaben lösen zu können, in einem System zu kombinieren. Die übliche Strategie empfiehlt, dazu zwei voneinander getrennte Komponenten zu verwenden. Auf der einen Seite ermöglicht es eine reaktive Komponente, dass der Roboter ein bestimmtes Verhalten ununterbrochen ausführen kann. Ein solches Verhalten ermöglicht ausreichend schnelle Antworten in dynamischen Umgebungen. Auf der anderen Seite stellt eine deliberative Komponente die Fähigkeit, komplexe Aufgaben zu erfüllen, bereit. Dieses geschieht in der Regel durch Planung, welche für die jeweils aktuelle Situation entsprechende Pläne erstellt. Diese zwei Komponenten laufen zwar unabhängig voneinander, aber dennoch müssen sie aufeinander abgestimmt werden. Beispielsweise muss die reaktive Kontrolle die deliberative Kontrolle aufheben können, wenn die Umgebung eine plötzliche Herausforderung präsentiert. Andersherum muss die deliberative Kontrolle die reaktive Kontrolle instruieren dürfen, so dass der Roboter dazu gebracht werden kann, nach effizienten oder sogar optimalen Strategien vorzugehen, um ein spezifisches Ziel zu erreichen (siehe Abb. 2.5a).

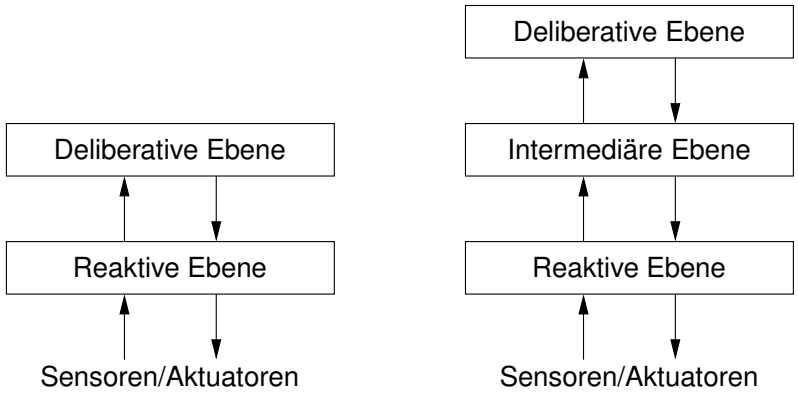


Abbildung 2.5.: Konzeptdiagramme hybrider Kontrollarchitekturen: (a) klassischer Ansatz, (b) Drei-Ebenen-Architektur.

Die Steuerung der reaktiven Komponente durch die deliberative Ebene erlaubt es einer hybriden Roboterarchitektur, ein komplexeres Verhalten auszuführen. Wo rein reaktive Kontrolle nur unabhängige Aktionen, wie z.B. das Folgen von Wänden oder das Ausweichen von Hindernissen ermöglicht, erlauben hybride Systeme dem Roboterkontrollsystem eine Serie von Verhaltensweisen nacheinander auszuführen. Eine typische Sequenz um einen Roboter in einen bestimmten Raum zu navigieren könnte es beispielsweise sein, einer Wand zu folgen und dabei einem unerwarteten Hindernis auszuweichen, an einer Tür zu halten, dann sich um 90° zur Tür zu drehen und schließlich durch die Tür in das Büro zu fahren. Diese Art der Sequenzierung von Aktionen wäre nur schwierig in einem rein reaktiven System umzusetzen. Auch eine rein deliberative Steuerung würde hier scheitern, da das dynamische Hindernis ein Problem darstellt. In hybriden Systemen können hingegen durch die Kombination von reaktiver und deliberativer Kontrolle einzelne Verhaltensweisen aktiviert und deaktiviert werden, um komplexere Ziele zu erreichen.

Allerdings stellt gerade die Verbindung der reaktiven und der deliberativen Komponente die Herausforderung bei diesem Ansatz dar. Die beiden Komponenten müssen miteinander kommunizieren, um Konflikte zwischen einander auflösen zu können. Diese Interaktion bedarf einer ausgefeilten Ablaufkoordinierung, da reaktive Kontrolle hauptsächlich Sensorinformationen verarbeitet und folglich mit relativ hoher Frequenz operiert, während deliberative Kontrolle eher längeren Arbeitszyklen unterliegt. In AuRA werden die bereits erwähnten *Motor Schemas* der reaktiven Komponente durch Anweisungen der deliberativen Komponente instanziiert. Ein so genannter „*Schema Manager*“ kontrolliert und überwacht die entsprechenden Prozesse bzw. Verhaltensweisen während deren Ausführung und sendet resultierende Statusinformationen an die deliberative Komponente zurück.

Oft wird zur Steuerung der Interaktion zwischen reaktiven und deliberativen Komponenten eine zusätzliche intermediäre Ebene eingefügt. Die Entwicklung einer solchen Vermittlerkomponente stellt gewöhnlich die größte Herausforderung in einem hybriden System dar [Mat01a]. Resultierende Architekturen werden üblicherweise als „Drei-Ebenen-Architekturen“ bezeichnet und bestehen aus einer reaktiven, einer intermediären und einer deliberativen Ebene (siehe Abb. 2.5b).

Auch wenn alle hybriden Architekturen der oben genannten Philosophie folgen, so unterscheiden sie sich jedoch gewöhnlich in ihrer Gesamtstruktur, in ihren Kontrollmethoden, in ihrem Schwerpunkt und in den Schnittstellen zwischen deliberativen und reaktiven Komponenten. Arkin [Ark98] gibt einen kurzen Überblick über individuelle Ausprägungen hybrider Kontrolle und beschreibt, auf welche Weisen Planung realisiert werden kann.

Neben der bereits erwähnten Architektur AuRA sind weitere hybride Systeme für mobile Roboter zahlreich in der Literatur vertreten. Dies zeigt, dass die Kombination von reaktiver und deliberativer Kontrolle sich sehr gut dazu eignet, ein autonomes Robotersystem zu kontrollieren. Ein Beispiel stellt das *Planner-Reactor*-System [Lyo92] dar. Dabei handelt es sich bei dem so genannten „*Reactor*“ um eine Komponente, welche einen Verbund von Sensor-Aktuator-Verknüpfungen enthält. Diese Verknüpfungen werden wiederum als „*Reactions*“ bezeichnet. Im Gegensatz zu einer gewöhnlichen Planausführungskomponente, welche eine Aktion nur dann ausführen kann, wenn die vorherige abgeschlossen wurde, kann der *Reactor* Aktionen zu jedem Zeitpunkt erzeugen. Außerdem arbeitet diese Komponente in Realzeit. Sie wird zu Beginn mit einer gewissen Anzahl an allgemein „sinnvollen“ Verhaltensweisen initialisiert. Der „*Planner*“ ist komplett unabhängig von und nebenläufig zum *Reactor*. Er arbeitet ebenfalls kontinuierlich und überprüft dabei ständig, ob das Verhalten des *Reactors* abhängig vom aktuellen Zustand der Umgebung noch mit seinen Zielen konform ist. Wenn der *Planner* feststellt, dass der *Reactor* diesen Zielen zuwider handelt, so kann er inkrementelle Änderungen an der Konfiguration des *Reactors* vornehmen. Somit ist gewährleistet, dass der Roboter seine Aufgabe entsprechend erfüllen kann. Insgesamt existieren zwei Verbindungen zwischen den beiden Komponenten. Zum einen werden bestimmte Sensordaten, so genannte Perzepte, die durch den *Reactor* erfasst werden, zum *Planner* gesendet. Zum anderen werden vom *Planner* spezifizierte Konfigurationsänderungen an den *Reactor* geschickt, um das Verhalten der reaktiven Komponente zu modifizieren.

Ein weiteres Beispiel für eine hybride Architektur ist durch ATLANTIS [Gat92] gegeben. Sie wurde unter anderem dazu eingesetzt, ein Marsfahrzeug zu steuern. Die Asynchronität zwischen

Planungskomponente und Reaktionskomponente wird hier durch so genannte heterogene Architekturelemente erzeugt. Analog zum *Planner-Reactor*-System wird auch hier das Ergebnis des Planers nicht dazu verwendet den Roboter direkt zu steuern, sondern dazu, ihn entsprechend anzuleiten. ATLANTIS verwendet ein so genanntes „kontinuierliches“ Aktionsmodell, welches sich spezieller Operatoren bedient. Dabei benötigt die Ausführung eines solchen Operators nur eine vernachlässigbar kleine Zeitspanne. Dieses wird dadurch erreicht, dass die Operatoren selbst keine Veränderungen an der Umwelt vornehmen, sondern stattdessen entsprechende Prozesse initiieren, welche dann die Veränderungen verursachen. Diese Prozesse werden als *„Activities“* bezeichnet und die Operatoren, die sie initiieren, werden *„Decisions“* genannt. Erstere können Rechenprozesse enthalten, welche wiederum weitere *Activities* initiieren können. Unter der Annahme, dass keine Zyklen unter den Initiierungen dieser Aktivitäten existieren, ergibt sich eine Hierarchie, in die man sie einordnen kann. *Activities*, die in der Hierarchie höher stehen, enthalten Rechenprozesse, welche tiefer gelegene Aktivitäten initiieren können. Die Hierarchie endet nach unten in primitiven *Activities*, die keine Rechenprozesse mehr enthalten, sondern nur noch sensomotorische Reaktionen ausführen. Um einen mobilen Roboter zu steuern, werden in ATLANTIS drei Komponenten verwendet: Ein Mechanismus um die *Activities* zu kontrollieren, ein Mechanismus um Rechenprozesse ausführen zu können und eine Ablaufsteuerung um die Interaktion zwischen den zwei genannten Mechanismen zu kontrollieren. Somit ergibt sich eine Drei-Ebenen-Architektur für ATLANTIS.

Im Gegensatz zu ATLANTIS handelt es sich bei der ebenfalls hybriden *Task Control Architecture* (TCA) [Sim94] um eine flexiblere Architektur, obwohl sie nur aus zwei Ebenen besteht. Sie ermöglicht es einer Vielzahl von unterschiedlichen autonomen Robotern, in komplexen und dynamischen Umgebungen zu operieren. Die Voraussetzung dazu ist, dass ein Roboter in der Lage ist, seine beschränkten physischen Ressourcen, wie z.B. Sensoren, und seine Rechenkapazität effektiv zu koordinieren und zu nutzen. Insbesondere wegen der ständig steigenden Komplexität von Robotersystemen wird es in [Sim94] als notwendig angesehen, dass der Ausführung von Planung, Wahrnehmung und Aktionen explizite Einschränkungen auferlegt werden. Dieses Vorgehen soll sicherstellen, dass keine unerwünschten Interaktionen zwischen verschiedenen Verhaltensweisen auftreten. Für die entsprechende Umsetzung stellt TCA einen integrierten Satz von Kontrollkonstrukten zu Verfügung, welche die Basis darstellen, um deliberative und reaktive Verhaltensweisen zu implementieren. Des Weiteren stellt TCA Mechanismen zur verteilten Kommunikation, zur hierarchischen Zerlegung von Aufgaben, zur Verwaltung von Ressourcen, zur Überwachung der Prozessausführung und zur Fehlerbehandlung bereit.

Ein weiteres Beispiel für eine hybride Architektur ist 3T [Bon97], deren Name sich von dem Begriff *„3 Tiers“* ableitet und somit die Verwendung von drei Abstraktionsebenen innerhalb der Architektur anzeigt. 3T verwendet eine spezielle Beschreibungssprache, welche zwischen den einzelnen Ebenen kompatibel ist. Dieser Aufbau vereinfacht hier die Koordinierung zwischen planungsbasierten Aktivitäten und Verhaltensweisen, die realzeitfähig sein müssen, um in dynamischen Umgebungen operieren zu können. Die reaktive Ebene besteht aus einem Satz von reaktiven *„Skills“* welche durch einen so genannten *„Skill Manager“* koordiniert werden. Die intermediäre Ebene ist für die Sequenzierung verantwortlich und arbeitet unter Verwendung von RAPs [Fir89]. Ein Planer auf der deliberativen Ebene entscheidet über die Ziele des Roboters,

die Verwendung von Ressourcen und die Verteilung von Zeitbeschränkungen. Im Unterschied zu anderen hybriden Architekturen sind für 3T eine Menge an Software-Werkzeugen entwickelt worden. Diese haben es ermöglicht, dass die Architektur auf mehreren realen Robotersystemen implementiert werden konnte, welche mit unterschiedlichster Hardware ausgestattet waren. Dazu zählt ein mobiler Roboter, der in einem Gebäude navigieren kann, und einer, der Abfall einsammeln kann. 3T wurde auch für einen Roboter verwendet, der Personen identifiziert. Dazu nähert sich das System zuerst einer Person mit einer bestimmten Bekleidungsfarbe an, um anschließend ihr Gesicht untersuchen zu können. Für diese Aufgabe wurde die deliberative Ebene allerdings nicht benötigt.

Das Ziel bei dem Design der Architektur des hybriden Roboterkontrollsystems ARCoS [Nev98] war es, sie so generisch wie möglich zu gestalten. Die Architektur sollte angemessen für einen mobilen Roboter sein, der eine Vielzahl von Aufgaben in verschiedensten Umgebungen bewältigen soll. ARCoS ist ein agentenbasiertes System und unterstützt somit verteilte Kontrolle. Dabei bilden die Agenten der reaktiven Ebene den Schwerpunkt des Roboterkontrollsystems. Die Agenten der deliberativen Ebene weisen wiederum die reaktive Agenten an, so dass der Roboter eine Art von gerichteter Reaktionsfähigkeit aufweist. Zusätzlich ist ARCoS mit der Fähigkeit zum Lernen ausgestattet. Diese erlaubt es dem Roboter, sich an die Umwelt anzupassen, um somit unterschiedliche und auch neue Situationen meistern zu können.

Eine etwas andere Art der hybriden Kontrolle findet sich in der Architektur von Jensen und Veloso [Jen98] zur Steuerung von RoboCup-Agenten. Es handelt sich dabei zwar um eine in zwei Ebenen unterteilte Architektur, jedoch steuern hier deliberative und reaktive Komponenten einen Roboter nicht gleichzeitig, sondern abwechselnd. Die vorgeschlagene Architektur ist außerdem für das Zusammenspiel mehrerer Roboteragenten ausgelegt, aber die grundlegende Idee, dass Pläne in Realzeit ermittelt und ausgeführt werden müssen, gilt dennoch für jeden einzelnen Roboter. Daher kann, wenn es die Situation erfordert oder bei der Ausführung eines Plans ein Konflikt auftritt, die reaktive Komponente die Steuerung des Roboters solange übernehmen, bis ein neuer Plan von der deliberativen Komponente erzeugt wurde.

Ebenfalls außergewöhnlich ist der Kontrollfluss in der als hybrid zu klassifizierenden *ARTIS Agent Architecture* [Sol00]. In dieser Architektur berechnet die reaktive Komponente in jedem Durchlauf eine Antwort. Danach entscheidet der Roboteragent allerdings, ob noch ausreichend Zeit zur Verfügung steht, um bei der deliberativen Komponente eine Anfrage nach einer eventuell geeigneteren Aktion zu stellen. Dieses ist möglich, da neben der reaktiven Komponente auch die deliberative Komponente hart realzeitfähig ist. Sollte allerdings nicht genug Zeit für die Planung zur Verfügung stehen oder die deliberative Komponente ihre Rechenzeit überschreiten, so wird die bereits von der reaktiven Komponente bestimmte Aktion zur Ansteuerung des autonomen mobilen Roboters verwendet.

Da der Trend bei der Entwicklung von autonomen mobilen Robotern dahin geht, immer mehr Fähigkeiten zur Interaktion mit Menschen zu integrieren, werden diese Roboter auch immer menschenähnlicher. Daher erwarten Benutzer eines solchen Systems in der Regel auch, dass der Roboter sowohl Emotionen besitzt als auch durch Motivationen angetrieben ist. Aus diesem Grund wird in [Sto01b] eine entsprechend spezielle hybride Roboterarchitektur vorgeschlagen.

Diese enthält zwischen deliberativen und reaktiven Komponenten eine zusätzliche Motivationskomponente, um Emotionen wie z.B. Verdruss, Frustration, Heimweh und Neugier nachzubilden. Durch solche Emotionen soll es dem Benutzer einfacher möglich sein, die Motivationen des Roboters zu verstehen, da die im System umgesetzten Emotionen an das menschliche Vorbild angelehnt sind.

Es wird allgemein angenommen, dass hybride Architekturen auch in Zukunft unter autonomen mobilen Robotern den größten Anteil ausmachen, da sie auch bei ständig steigender Komplexität gut skalieren. Bezüglich des zu erwartenden Umfangs der zu integrierenden Fähigkeiten müssen allerdings für zukünftige Roboterarchitekturen entsprechende Design-Kriterien bestimmt werden. Aus diesem Grund werden in [Ore03] drei aktuelle hybride Architekturen untersucht: Saphira, TeamBots und BERRA [Lin00]. Das Ergebnis dieser Untersuchung ist vielfältig. Neben dem Bedarf für einheitlichere Schnittstellen zur Hardware, um eine höhere Portabilität zu gewährleisten, wird die Entwicklung eines hochmodularen Systems als erstrebenswert erachtet. Nach Möglichkeit sollten keine unterschiedlichen Funktionalitäten in einem Modul vereint werden, um diese einfacher austauschen oder ersetzen zu können. Dies ist in den meisten hybriden Architekturen bisher nur schwer oder gar nicht möglich. Auch die Repräsentation von Daten sollte in Zukunft strukturierter ausfallen, damit diese Daten durch unterschiedliche Module möglichst einfach verarbeitet werden können. Des weiteren sind die meisten der bestehenden Architekturen nicht für interaktive Aufgaben ausgelegt. Die Möglichkeit, mit Menschen natürlich zu interagieren, wird auf Robotern bisher weitgehend ignoriert. Doch insbesondere für Service-Roboter sind neben Fähigkeiten zur flexiblen Ausführung von autonomen Aufgaben auch Fähigkeiten zur Interaktion mit Menschen essentiell.

2.2.4. Behavior-basierte Ansätze

Behavior-basierte Kontrolle, gelegentlich auch als *„Reactive Subsumption-Style Control“* bezeichnet, ist inspiriert und motiviert durch die Biologie. Dabei sind Behavior-basierte Systeme nach ihren grundlegenden Komponenten benannt, den so genannten „Behaviors“. Entsprechende Aktivitäten dieser Komponenten spiegeln sich in beobachtbaren Verhaltensmustern wider, welche aus der Interaktion zwischen dem Roboter und seiner Umgebung entstehen [Mat01a]. Behavior-basierte Kontrolle sollte allerdings nicht mit rein reaktiver Kontrolle verwechselt werden. Sie unterscheidet sich von der letzteren durch die Tatsache, dass sowohl komplementäre als auch sich widersprechende Aktionen vom System parallel zueinander erzeugt werden. Diese Aktionen werden durch einen oder mehrere Koordinierungsmechanismen verarbeitet um eine endgültige Aktion zu erhalten, die schließlich vom Roboter ausgeführt wird (siehe Abb. 2.6). Das resultierende sichtbare Verhalten des Systems wird auch als *„Emergent Behavior“* bezeichnet [Ste94], welches sich durch die Verknüpfung der einzelnen Behaviors ergibt.

Zusätzlich können Behavior-basierte Systeme, im Gegensatz zu reaktiven Systemen, ihre Umwelt repräsentieren. Die Hauptcharakteristik dabei ist aber, dass Behavior-basierte Ansätze Informationen nicht wie bei deliberativen Systemen zentralisiert speichern. Stattdessen werden entsprechende Repräsentationen lokal durch die Behaviors verwaltet. Die Verwendung interner

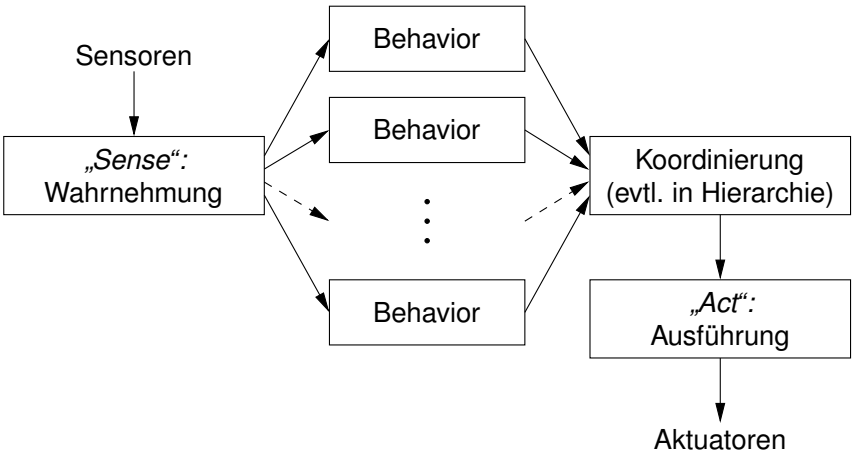


Abbildung 2.6.: Konzeptdiagramm einer Behavior-basierten Kontrollarchitektur.

Zustände ermöglicht es einem Behavior-basierten System somit nicht nur, deliberative Prozesse auszuführen, sondern auch zu lernen.

Falls ein durch Behaviors gesteuerter Roboter Kontrolle Planung betreiben soll, so kann es sein, dass dazu benötigte Informationen in mehreren Behaviors verteilt vorliegen. In diesem Fall müssen die Behaviors in der Lage sein, miteinander zu kommunizieren und Daten auszutauschen. Daher wird die Planung durch ein Netzwerk von miteinander interagierenden Behaviors vorgenommen anstelle durch einen zentralisierten Planer. Dieser Umstand muss insbesondere dann berücksichtigt werden, wenn Leistungsfähigkeit und Effizienz des Behavior-basierten Systems wichtige Schwerpunkte im Design darstellen.

Auch wenn jedes Behavior die Möglichkeit zu komplexen Operationen hat, kann ein Behavior-basiertes System sehr wohl rein reaktive Komponenten enthalten. Nicht jedes Behavior muss zwangsläufig in Prozessen zur Erstellung von Repräsentationen involviert sein. Eigentlich verwenden viele Behavior-basierte Architekturen überhaupt keine komplexen Repräsentationen. Im Gegensatz zu rein reaktiven Systemen wird hier die Ansteuerung des Roboters aber nicht durch eine zentrale Ansammlung von einfachen Regeln begründet. In Behavior-basierten Systemen ist für jede Verhaltensweise des Roboters ein bestimmtes Behavior verantwortlich. Der Vorteil gegenüber zentralisierten Architekturen ist, dass in einem Behavior-basierten System einzelne Behaviors kein vollständiges Wissen über die Umwelt benötigen. Sie erhalten ausschließlich die Sensordaten, die unmittelbar für den jeweiligen Prozess der Entscheidungsfindung maßgeblich sind und erzeugen aus den gewonnenen Informationen entsprechende Ausgaben. In Behavior-basierten Ansätzen ist die Modularisierung des Kontrollsystems somit aufgabenorientiert. Auf diese Weise ist jedes einzelne Behavior frei in der Wahl einer individuellen internen Repräsentation und kann diejenige verwenden, die am angemessensten für den Einsatzzweck des Behaviors erachtet wird. Da außerdem keine Notwendigkeit besteht, sämtliche vorhandenen Daten individueller Behaviors in einem kohärenten Weltmodell zu fusionieren bevor ein einzelnes Behavior seine Antwort ausgeben kann, ergeben sich im Gesamtsystem für die einzelnen Verhaltensweisen diesbezüglich weniger Querabhängigkeiten und somit eine bessere Wartbarkeit des Systems.

Da mehrere Behaviors spezielle Aufgaben parallel durchführen können, um bestimmte Ziele zu erreichen, vermeiden Behavior-basierte Architekturen erfolgreich den Flaschenhals von Sensordatenaufnahme und Planung, der in zentralisierten Systemen auftritt. Dadurch ergibt sich ein wichtiger Vorteil im Punkt der Reaktivität. Da ein Behavior allerdings ein in sich geschlossenes Modul ist, welches individuelle Eingabedaten auf spezifische Aktionen abbildet, beziehen sich dessen Ausgaben jeweils nur auf einen spezifischen Aspekt bei der Ansteuerung eines Roboters, um gewisse Ziele zu erreichen. Daher ergibt sich für ein Behavior-basiertes Roboterkontrollsystem die Anforderung, dass es in der Lage ist, mehrere eventuell gegenläufige Ziele zu koordinieren. Folglich ist es in einem solchen System erforderlich, entweder eine Auswahl aus den von den einzelnen Behaviors vorgeschlagenen Aktionen zu treffen oder sie miteinander zu kombinieren. Das Ziel einer solchen Koordinierung ist es, dass eine Aktion bestimmt wird, die den Erfordernissen des gesamten Systems genügt.

Die Art und Weise, in der eine Koordinierungskomponente die Ausgabe aller Behaviors vereinen soll, wird in der Entwicklungsphase des Systems bestimmt. Entweder wird dabei eine „Arbitrierung“ vorgenommen, d.h. es wird genau eine gewünschte Aktion ausgewählt, oder alle Aktionen werden miteinander fusioniert (siehe Abb. 2.7). Zusätzlich werden gewöhnlich unterschiedliche Gewichte an die Ausgaben verschiedener Behaviors geknüpft um die Flexibilität der Koordinierung zu erhöhen. Allerdings ist es selten trivial, solche Abstimmungen zu optimieren. Darüber hinaus ergibt sich beim Erweitern von großen Behavior-Netzwerken das Problem, dass bestehende Behaviors unnötig oder unerwünscht werden können. Entsprechende Konflikte zwischen Behaviors sind allerdings nicht immer sofort offensichtlich, so dass die Wartung eines Behavior-basierten Systems aus diesem Grund oft eine große Schwierigkeit darstellt.

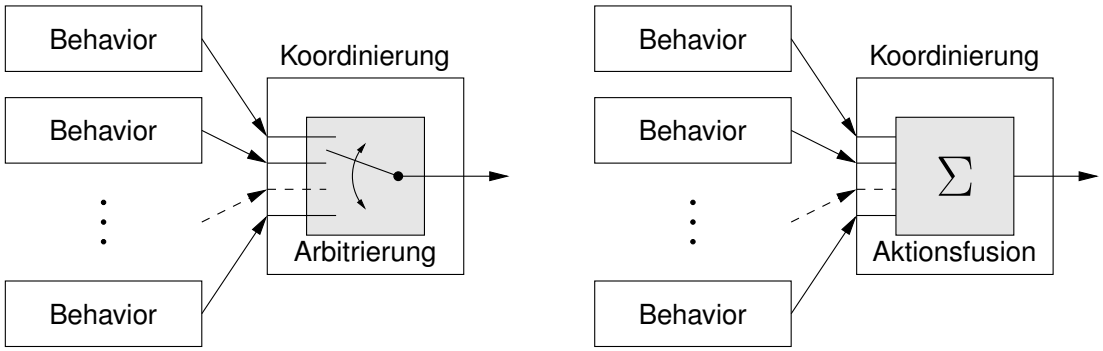


Abbildung 2.7.: Methoden zu Behavior-Koordinierung: Arbitrierung und Aktionsfusion.

Eine der ersten Methoden zur Arbitrierung ist in der reaktiven *Subsumption*-Architektur von Brooks eingesetzt worden, wobei es hier einem Behavior erlaubt ist, die Ausgabe eines anderen Behaviors komplett zu überschreiben [Bro86]. Obwohl dies ein effektives Schema ist, um unter unvereinbaren Anweisungen eine passende auszuwählen, so stellt es kein angemessenes Mittel dar, um mehrere Ziele zu bewältigen, welche gleichzeitig erfüllt werden können und auch werden sollen. Das Unvermögen einen entsprechenden Kompromiss zu schließen entspringt aus der Tatsache, dass in einer solchen prioritätsbasierten Arbitrierung das gesamte Wissen ignoriert wird, welches innerhalb eines jeden Behaviors verwendet wurde, um die Ausgabe des Behaviors zu

begründen. Diese Informationen und das individuelle Wissen, welches durch das entsprechende Behavior repräsentiert wird, geht bei dieser Art von Koordinierung komplett verloren.

Um das oben genannte Problem der Arbitrierung zu lösen, entstand die Strategie, die Ausgaben von verschiedenen Behaviors miteinander zu kombinieren bzw. eine Fusion der entsprechenden Aktionen durchzuführen. Das bedeutet, dass bei dieser Art der Koordinierung Entscheidungen aufgrund mehrerer Betrachtungsweisen getroffen werden können, während die Modularität und die Reaktivität eines verteilten Systems erhalten bleibt. Des Weiteren kann ein Robotersystem dynamischer als unter Verwendung einer Arbitrierungsmethode auf seine Umgebung reagieren, ohne dabei die Probleme zu erleiden, die mit der Fusion von Sensordaten einhergehen [Ros95]. Allerdings stellt der Fusionsprozess wiederum einen Flaschenhals im System dar. Dafür besteht hier im Gegensatz zur Arbitrierung nicht die Gefahr, dass wertvolle Informationen, die für den Entscheidungsprozess wesentlich sind, verloren gehen.

Insgesamt muss bei der Entwicklung eines Koordinierungsprozesses sehr sorgfältig ein Gleichgewicht zwischen Vollständigkeit und Optimalität auf der einen Seite und Modularität und Effizienz auf der anderen Seite ermittelt werden. Die am häufigsten verwendeten Methoden zu Koordinierung sind im Folgenden zusammengefasst (vgl. [Ros95, Pir99]):

- „*Prioritization*“: Diese Art der Arbitrierung wird auch als „*Suppression*“ bezeichnet. Es wird eine strikte Hierarchie von Behaviors definiert, wobei sich das am höchsten in der Hierarchie befindliche aktive Behavior gegenüber den anderen Behaviors durchsetzt. Beispielsweise kann das Behavior „Weiche dem Hindernis aus“ das Behavior „Gehe zum Zielpunkt“ übersteuern, wenn der Roboter sich auf ein Hindernis zu bewegt. Dieser Mechanismus zur Arbitrierung wurde ursprünglich von Brooks für seine *Subsumption*-Architektur entwickelt [Bro86], allerdings ergibt sich dort die Arbitrierung durch eine feste Verdrahtung der Behaviors miteinander.

- „*Selection*“: Bei dieser Arbitrierungsvariante ergibt sich eine variable Hierarchie von Behaviors. Dabei bestimmen neben Sensordaten zusätzliche Einflüsse, wie z.B. die Ziele oder die Intentionen des Systems, welches Behavior schließlich die Kontrolle über den Roboter erhält. Dieser Ansatz ist zwar dynamischer als die Verwendung von *Prioritization*, allerdings wird auch hier eine Aktion nur durch genau ein Behavior bestimmt. Zu dieser Methode zählt auch das so genannte „*Activation Spreading*“, bei dem die einzelnen Behaviors untereinander bestimmen, welches Behavior sich gegenüber den anderen Behaviors durchsetzt [Mae89].

- „*Voting*“: In diesem Fall wird eine kooperative Form der Arbitrierung vorgenommen, die es allen Behaviors ermöglicht, Stimmen für ihre bevorzugte Aktion abzugeben. Dabei kann ebenfalls die Anzahl der Stimmen je nach Priorität des entsprechenden Behaviors variieren. Die Aktion, welche die meisten Stimmen erhält, wird schließlich ausgeführt. Somit können sich z.B. mehrere Behaviors gemeinsam gegen ein anderes Behavior durchsetzen. Allerdings muss sichergestellt sein, das alle Behaviors regelmäßig ihre Stimme abgeben, damit sich eine ausgewogene Abstimmung ergibt. Ansonsten muss die Dynamik des Abstimmungsverhaltens berücksichtigt werden.

- „*Superposition*“: Hierbei handelt es sich um eine Fusionsmethode, die auch „*Vector Summation*“ genannt wird. Die grundlegende Idee dieser Methode ist es, dass die Ausgaben aller Behaviors Vektoren darstellen. Diese Vektoren können beispielsweise zum Bewegen des Roboters Geschwindigkeiten oder Beschleunigungswerte kodieren. Sie werden von der Koordinierungskomponente aufsummiert und in manchen Implementierungen auch normalisiert, um eine entsprechende Ansteuerung der Motoren zu erreichen. Dieser Art von Linearkombination gewünschter Aktionen kann z.B. durch Verwendung von Potentialfeldern oder *Motor Schemas* [Ark89] erreicht werden.

- „*Fuzzy Control*“: Diese Art der Aktionsfusion basiert auf der Verwendung von *Fuzzy Logic* (siehe z.B. [Lee90]). Sie bietet eine einfache Art, um eine eindeutige Entscheidung basierend auf vagen, mehrdeutigen, ungenauen, verrauschten oder fehlenden Eingabeinformationen zu erzielen [Yen95]. Dieses ist gerade bei den Sensordaten, die von einem mobilen Roboter aus aufgenommen werden, sehr zuträglich. Allerdings leidet diese Fusionsmethode an Problemen, die mit dem Auflösen von Konflikten zwischen Behaviors einhergeht [Pir99].

Eine weitere Herausforderung bei der Entwicklung eines Behavior-basierten Systems ist es, die richtige Zusammensetzung von Behaviors zu wählen. Dabei ist die richtige Kombination von reflexorientierten und planungsorientierten Behaviors zu ermitteln, um die notwendigen Verhaltensweisen des Roboters, die *Emergent Behaviors*, zu erhalten. Beispielsweise könnte ein Roboter, dessen Aufgabe es ist, Büros zu reinigen, nicht über ein spezifisches Behavior „Reinige den Raum“ verfügen, aber dennoch diese Aufgabe erfolgreich erfüllen. Er könnte z.B. simple Behaviors wie „Fahre herum“, „Weiche dem Hindernis aus“, „Finde Müll“ und „Hebe Müll auf“ besitzen. Diese grundlegenden Behaviors würden dann dazu führen, dass der Roboter durch die Interaktion mit seiner Umwelt den Raum reinigt, in dem er sich befindet. Wenn ein Roboter in der Lage sein soll, unterschiedliche Aufgaben zu erfüllen, so sorgt eine entsprechende Anzahl von *Emergent Behaviors* letztlich für das gewünschte Gesamtverhalten des Roboters.

Schließlich müssen die einzelnen Behaviors in einer Behavior-basierten Architektur nicht strikt parallel laufen. Diese können auch in Hierarchien angeordnet sein. In so einem Fall bildet in der Regel eine Kollektion von einfachen Behaviors die Basis des Systems. Dabei arbeiten die Behaviors aus der Basis wie gewöhnlich, d.h. sie bilden direkt Sensoreingaben auf Roboteraktionen ab. Auf höheren Ebenen in der Hierarchie können Behaviors auch die Ausgaben der einfacheren Behaviors als Eingabe nutzen um komplexere Fähigkeiten zur Verfügung zu stellen. Die Interaktion zwischen all diesen Behaviors erlaubt es dem Roboter schließlich, umfangreiche Reaktionen in seiner Umgebung hervorzubringen. Ein weiterer Vorteil der Hierarchisierung von Behaviors ist, dass gewisse Grundreflexe, die eventuell in mehreren Behaviors benötigt werden, in separate Behaviors ausgelagert werden können. Somit kann eine mehrfache Implementierung von bestimmten Verhaltensweisen vermieden werden.

Ein Beispiel für eine Behavior-basierte Architektur ist mit der *Distributed Architecture For Mobile Navigation* (DAMN) [Ros95] gegeben. DAMN ist als eine Sammlung von verteilten Behaviors implementiert, die mit einem zentralen Arbitrierer kommunizieren. Dabei sendet jedes Behavior

sein Votum an den Arbitrierer. Dieses Votum fällt zu Gunsten einer bestimmten Aktion aus, welche das Erreichen des Ziels des jeweiligen Behaviors sicherstellt und gegen jene Aktionen, die dieses nicht gewährleisten. Der Arbitrierer weist dann den Eingaben von jedem Behavior ein Gewicht zu. Allerdings können in DAMN auch höher gelegene Module zur Aktionsplanung in das System integriert werden, wodurch sich eine so genannte „heterogene Kontrolle“ ergibt. Diese Module können auf der Meta-Ebene Kontrolle über das System ausüben, indem sie die Gewichte, die der Arbitrierer an die einzelnen Behaviors knüpft, modifizieren können. Somit kann der Grad, mit dem jedes Behavior Einfluss auf den Entscheidungsprozess im System nehmen kann, kontrolliert werden. Dies wirkt sich natürlich auch unmittelbar auf die Aktionen des Roboters aus. Insgesamt sorgt der zentrale Arbitrierer für ein kohärentes und zielgerichtetes Verhalten des Roboters, während die Behaviors ein Ansprechverhalten bezüglich der physischen Umgebung in Realzeit sicherstellen.

Eine Behavior-basierte Architektur, die ganz ohne Behavior-Koordinierung auskommt wird von Ishiguro et al. [Ish99] beschrieben. Das System wird dazu mit Behaviors ausgestattet, welche sequentiell ausgeführt werden. Die Reihenfolge, in der die Behaviors abgearbeitet werden, wird allerdings nicht vorherbestimmt, sondern automatisch zur Laufzeit ermittelt. Um dieses zu ermöglichen wird jedes Behavior mit entsprechenden *„Preconditions“* versehen, die erfüllt sein müssen, bevor das zugehörige Behavior ausgeführt werden darf. Da sich bei dieser Strategie Abhängigkeiten zwischen den Behaviors in der Regel nicht schon während der Entwicklung dem Programmierer gegenüber offenbaren, erlaubt ein spezieller Mechanismus eine so genannte progressive Programmierung des Systems. Dies bedeutet, dass wenn das System im Einsatz vor einem unlösbaren Problem stehen sollte, d.h. kein passendes Behavior vorhanden sein sollte, der Entwickler zur Laufzeit ein entsprechendes neues Modul implementieren und dem System hinzufügen kann. Anschließend kann das System den Einsatz fortsetzen. Durch diese Art der dynamischen Entwicklung können Ausnahmen sofort behandelt werden, wodurch das System robuster wird. Sollten zu einem Zeitpunkt mehrere Behaviors für eine bestimmte Situation ausführbar sein, so wird das Behavior mit der besten Bewertung ausgewählt. Ein Modul wird dabei positiver bewertet, wenn es erfolgreich ausgeführt werden konnte, und negativer bei einem Fehlschlag. Behaviors, deren Bewertung einen bestimmten Schwellwert unterschreitet, werden automatisch aus dem System entfernt.

Nicolescu und Matarić [Nic02] stellen heraus, dass Behavior-basierte Systeme zwar erfolgreich für eine Vielzahl von Anwendungen eingesetzt wurden, aber aufgrund der eingeschränkten Verwendbarkeit von abstrakten Repräsentationen selten für komplexere Probleme verwendet wurden. Diese Probleme umschließen z.B. die Verwaltung von zeitlichen Abläufen oder die Repräsentation von hierarchisch strukturierbaren Aufgaben. Um das Potential des Behavior-basierten Paradigmas besser nutzbar zu machen, widmen sich Nicolescu und Matarić zwei speziellen Einschränkungen. Erstens entbehren Behaviors abstrakte bzw. symbolische Repräsentationen, die es ihnen ermöglichen würden, zur Kontrolle auf einer höheren Ebene im System eingesetzt zu werden, wie es z.B. bei Operatoren in einem Plan der Fall ist. Zweitens wird die breite Mehrheit von Behavior-basierten Systemen noch immer von Hand für spezielle Aufgaben entworfen. Dabei wird eine Wiederverwendung von Behaviors für andere Aufgaben nicht berücksichtigt, was eine automatische Erzeugung von Behavior-basierten Systemen ausschließt. Diese Probleme werden

durch die Einführung von so genannten „abstrakten“ und „primitiven“ Behaviors behandelt, welche als Verkapselung für gewöhnliche Behaviors mit einer einheitlichen Schnittstelle verstanden werden können. Außerdem werden die Behaviors mit einer Repräsentation ihrer Ziele erweitert, wodurch sie parametrisierbar und somit generischer werden. Jedes Ziel des Gesamtsystems wird wiederum durch ein Netzwerk von Behaviors repräsentiert, wobei jedes Behavior-Netzwerk eine Hierarchie von abstrakten und primitiven Behaviors darstellt. Eine solche Anordnung erlaubt es, komplexe zeitliche Abläufe auszuführen, wobei die Behaviors, ähnlich zu denen von Ishiguro, auf so genannten *Preconditions* basieren, welche zwischen den einzelnen Behaviors übermittelt werden. Im Gegensatz zu hybriden Architekturen läuft dieser Ansatz nicht der Natur von Behavior-basierten Systemen zuwider: Er erlaubt es, komplexe Kontrolle in Form von echten Behaviors zu spezifizieren, welche dieselben Möglichkeiten zum Aufbau von Repräsentation besitzen und auf der gleichen Zeitskala arbeiten.

Im Gegensatz zu Nicolescu und Matarić wird von anderen Forschern, die ebenfalls kritisieren, dass Behavior-basierte Architekturen Punkte wie Planung, Lernen, Ziel-Management und Kommunikation nicht ausreichend unterstützen, eine heterogene Architektur bevorzugt. Zum Beispiel verwendet Michaud [Mic99] neben Behaviors zusätzliche deliberative Komponenten. Hier besteht das grundlegende Prinzip zur Verknüpfung aller Komponenten darin, dass Ergebnisse der auf vorhandenem Weltwissen beruhenden Planung genutzt werden können, um die Behaviors entsprechend zu rekonfigurieren. Insgesamt existieren neben einem *„Behavior Level“* noch zwei weitere Ebenen: ein so genannter *„Recommendation Level“*, welcher äußere und innere Einflüsse des Systems berücksichtigt und ein so genannter *„Motivation Level“*, welcher die Ziele des Systems repräsentiert. Die Ausgaben dieser zwei zusätzlichen Ebenen sorgen für eine entsprechende Konfiguration der Behavior-Ebene. Dieser Ansatz ist ebenfalls biologisch motiviert und ähnlich zu dem von BDI-Architekturen (vgl. z.B. JAM [Hub99]), deren *„Beliefs“* zu den *„Motives“* auf dem *Motivation Level* vergleichbar sind. Allerdings werden bei Michaud im Gegensatz zu BDI-Architekturen an Stelle der direkten Ausführung von ermittelten Plänen die Behaviors konfiguriert. Michauds Ansatz ist bezüglich der Verwendung seiner *Motives* außerdem ähnlich zu dem Ansatz der hybriden Architektur von Stoytchev und Arkin [Sto01b]. Hier werden ebenfalls Motivationen verwendet, um zu verhindern, dass der Roboter in aussichtslose Situationen gerät. Allerdings ist das Problem, solche Situationen zu detektieren, in Behavior-basierten Systemen kritischer. In planungsbasierten Systemen ist es in der Regel einfacher, ein Fehlverhalten des Roboters zu erkennen.

Ein guter Überblick über weitere Beispiele zu Behavior-basierten mobilen Robotern findet sich in [Mat97]. Die dort präsentierten Roboter werden allerdings nur für einzelne kleine Aufgaben eingesetzt, wie z.B. zur Navigation und um einfaches Gruppenverhalten zu lernen. Für komplexere Aufgaben, welche z.B. die Interaktion mit einem Menschen beinhalten, werden rein Behavior-basierte Systeme nur selten verwendet. Die Randbedingungen für Behavior-basierte Systeme erlauben viel Interpretationsfreiraum, so dass bisher diverse neuartige und effiziente Ansätze entwickelt werden konnten. Jedoch wird dieser Freiraum oft als ein Mangel an Struktur angesehen, der die Entwicklung eines komplexen Systems erschweren kann. Da verschiedene Behavior-basierte Systeme sehr unterschiedlich konstruiert sein können, ist es außerdem schwierig, sie im Rahmen einer Evaluation miteinander zu vergleichen.

2.2.5. Vergleich der Ansätze

Die Angemessenheit und Leistungsfähigkeit einer Kontrollarchitektur für einen mobilen Roboter hängt sehr stark von der geplanten Anwendung ab. Des Weiteren ist die Entscheidung, welches Kontrollparadigma vorzuziehen ist, auch vom jeweiligen Robotersystem abhängig. Rein deliberative Kontrolle wird gewöhnlich nur für sehr spezielle Zwecke eingesetzt, da sie nur innerhalb kontrollierter Arbeitsbereiche angemessen arbeitet. Rein reaktive Kontrollsysteme kommen üblicherweise nur für niedere Aufgaben und in hochdynamischen Umgebungen zum Einsatz. Sie haben den Nachteil, dass sie keinen internen Zustand pflegen können und dass ihre Komplexität unvertretbar stark ansteigen kann, wenn neue Fähigkeiten hinzugefügt werden. In Situationen, in denen die Umwelt exakt modelliert werden kann und eine Garantie existiert, dass sich während der Ausführung die Umgebung nicht verändert, wie z.B. in Einsatzbereichen von Montage-Robotern, werden deliberative Methoden bevorzugt, da dort Pläne effektiv ausgeführt werden können. In der realen Welt hingegen, in der ein mobiler Roboter arbeiten soll, sind die Voraussetzungen, die für rein deliberative Systeme gegeben sein müssen, nicht gegeben. Daher sind hier reaktive Grundfunktionalitäten unerlässlich.

Im Gegensatz zu deliberativen und reaktiven Systemen sind hybride und Behavior-basierte Systeme ausdrucksstärker und leistungsfähiger. Des Weiteren sind sie in ihren Fähigkeiten in etwa miteinander vergleichbar. Beide Paradigmen können Repräsentationen ihrer Umwelt aufbauen und Vorhersagen über die Zukunft treffen, aber jedes Paradigma verwendet dazu unterschiedliche Mechanismen. Oft wird behauptet, dass hybride Systeme, die in der Lage sind, deliberatives Schlussfolgern und reaktives Handeln miteinander zu verbinden, benötigt werden, um das volle Potential eines mobilen Roboters ausnutzen zu können [Ark95]. Behavior-basierte Ansätze sind zwar eine Erweiterung reaktiver Systeme, finden allerdings auch zwischen rein reaktiven Systemen und planungsbasierten Extremen ihren Platz [Bro86]. Behavior-basierte Strategien sind eben deswegen mächtiger als rein reaktive Systeme, da sie nicht den strengen Beschränkungen bezüglich des internen Zustands unterliegen. Behavior-basierte Systeme sind durch eine lose Koppelung zwischen Sensoren und Aktuatoren und durch eine aufgabenorientierte Modularisierung des Gesamtsystems charakterisiert. Hybride Systeme sind vorherrschend in Domänen in denen einzelne Roboter verwendet werden, da eine hybride Kontrolle sehr gut bezüglich der Anforderungen von Service-Robotern und auch humanoiden Robotern skaliert. Im Gegensatz dazu sind in Domänen, in denen mehrere Roboter zugleich eingesetzt werden, Behavior-basierte Systeme üblicher, da ihre Anpassungsfähigkeit es den Robotern auf flexibelste Weise erlaubt, ein gewünschtes Gruppenverhalten zu zeigen [Mat97]. Behavior-basierte Kontrolle wird außerdem oft im Unterhaltungsbereich für so genannte Haustierroboter eingesetzt [Ark03].

2.3. Zusammenfassung

In diesem Kapitel wurden die vier Hauptparadigmen diskutiert, die in den letzten Jahrzehnten entwickelt worden sind, um unter anderem mobile Roboter anzusteuern. Im Hinblick auf die

Wahl einer angemessenen Architektur für ein interaktives Robotersystem, qualifiziert sich für entsprechenden Aufgaben allerdings nur eine hybride oder eine Behavior-basierte Kontrolle. Auch wenn dem Paradigma der hybriden Kontrolle bei der Entwicklung eines Service-Roboters am häufigsten gefolgt wird, so sollte die Verwendung einer Behavior-basierten Kontrolle nicht im voraus ausgeschlossen werden. Gerade mit der Erweiterung um zusätzliche deliberative Elemente, also in Form einer heterogenen Architektur, verspricht das Behavior-basierte Paradigma auch komplexere Interaktionsfähigkeiten realisieren zu können. Welche Architekturform für die Entwicklung eines *Robot Companions* schließlich am geeignetsten ist, hängt aber nicht zuletzt von der Art der Eigenschaften ab, die das Gesamtsystem am Ende besitzen soll. Bevor die Entscheidung für eine bestimmte Kontrollarchitektur getroffen wird, werden daher im folgenden Kapitel zuerst entsprechende Anforderungen formuliert.

3. Anforderungen an eine zu entwickelnde Architektur

In diesem Kapitel werden die Anforderungen an eine Architektur für ein Robotersystem diskutiert, welches zur Steuerung eines *Robot Companions* im Rahmen des *Home Tour Scenarios* eingesetzt werden soll. Robotersysteme unterscheiden sich von anderen Software-Anwendungen auf viele Weisen [Cos00]: Zuerst besteht für ein Robotersystem die Notwendigkeit, komplexe Ziele erreichen zu können. Eventuell müssen auch mehrere Ziele zur gleichen Zeit verfolgt werden. Dabei muss es mit komplexen und dynamischen Umgebungen interagieren und auf unerwartete Veränderungen entsprechend reagieren können. Gleichzeitig muss das System in der Lage sein, verrauschte und unsichere Daten zu bewältigen. Diese Notwendigkeiten beeinflussen die Entwicklung von Robotersystemen und bestimmen maßgeblich, wie sie schließlich in ihrer Umwelt agieren und inwiefern sie validiert werden können. Zusätzlich zählen viele Roboter zur Klasse der kritischen Systeme. Dazu gehören insbesondere diejenigen, die zur Interaktion mit Menschen dienen. In solchen Systemen können Fehler, die während des Betriebs auftreten, erhebliche Konsequenzen, wie z.B. die Verletzung einer Person, nach sich ziehen. Somit steht auch die Funktionssicherheit des Systems im Vordergrund.

Bezüglich der Entwicklung des autonomen Service-Roboters PSR haben Kim et al. entsprechende Anforderungen an ihr System formuliert [Kim03]. PSR soll dabei Aufgaben in Bürogebäuden und Krankenhäusern übernehmen können, zu denen er von einem Benutzer angewiesen wird. Die genannten Anforderungen, welche auch bei der Entwicklung eines *Robot Companions* nicht unerheblich sind, haben schließlich zur Entwicklung einer Drei-Ebenen-Architektur für PSR geführt. Sie lauten:

1. Im Robotersystem müssen diverse Software-Module untergebracht werden, wobei jedes Modul mit mehreren anderen Modulen kommunizieren können muss. Dazu wird ein entsprechend flexibles Kommunikationssystem benötigt.

2. Die verschiedenen Software-Module unterliegen unterschiedlichen Prioritäten und Anforderungen bezüglich ihres Verarbeitungsablaufs. Manche Prozesse müssen in Realzeit betrieben werden, während sich für komplexe Berechnungen allenfalls untere Zeitschranken angeben lassen. Daher muss die Ausführung aller Prozesse entsprechend koordiniert werden können.

3. Die Architektur soll die Autonomie des Robotersystems unterstützen. Wenn das System vom Benutzer instruiert worden ist, so muss es zur Lösung der Aufgabe nicht nur entsprechende Pläne generieren und ausführen, sondern auch dabei auftretende Fehler erkennen und beheben können. Somit müssen neben deliberativen Komponenten auch entsprechende reaktive Module existieren, um Herausforderungen dynamischer Umgebungen bewältigen zu können.

4. Damit der Roboter bestimmte Aufgaben erfüllen kann, müssen diverse Hardware-Komponenten im System integriert werden. Dazu muss einerseits die Roboterbasis so konstruiert sein, dass alle benötigten Sensoren und Manipulatoren auf ihr Platz finden. Andererseits müssen diese Komponenten entsprechend den zu erfüllenden Aufgaben organisiert sein.

5. Da an der Implementierung der Software-Komponenten mehrere Entwickler beteiligt sind, müssen sie in der Lage sein, unabhängig voneinander bestehende Komponenten zu überarbeiten und neue Module hinzuzufügen. Aus diesen Gründen sollte das zu entwickelnde System modular sein und die Wiederverwendbarkeit von bestehenden Komponenten ermöglichen.

Die oben genannten Anforderungen sind allerdings eher allgemeiner Natur und beziehen sich im Wesentlichen auf technische bzw. strukturelle Aspekte eines Robotersystems. Für ein spezifisches Robotersystem, wie z.B. einen *Robot Companion*, der für den Einsatz im *Home Tour Scenario* entwickelt werden soll, lassen sich die Anforderungen jedoch in funktionale und strukturelle Anforderungen unterteilen. Erstere beschreiben die Fähigkeiten, die das Robotersystem besitzen soll, während letztere eher die dazu notwendigen Hilfsmittel darstellen. Diese Hilfsmittel sind aber in Bezug auf die Entwicklung eines Robotersystems in der Regel sehr zahlreich und stellen hohe Anforderungen an entsprechende Software-Techniken [Ore04].

Bezüglich der im Rahmen dieser Arbeit zu entwickelnden Architektur werden im Weiteren zuerst die funktionalen Anforderungen für einen *Robot Companion* in Abschnitt 3.1 präsentiert. Danach werden die strukturellen Anforderungen in Abschnitt 3.2 beschrieben. Schließlich wird das resultierende Architekturkonzept in Abschnitt 3.3 vorgestellt, das die Grundlage für die Entwicklung eines *Robot Companions* darstellt.

3.1. Funktionale Anforderungen

Da der Schwerpunkt im *Home Tour Scenario*, wie in Abschnitt 1.2 beschrieben, auf der Interaktion zwischen Roboter und Benutzer liegt, ergeben sich dadurch die Hauptanforderungen an das Gesamtsystem. Der Roboter muss in erster Linie in der Lage sein, Menschen wahrzunehmen. Da sich mehrere Menschen im Wahrnehmungsbereich des Roboters aufhalten können, sollte das System die Individuen zwecks selektiver Interaktion auseinander halten können. Ein *Robot Companion* sollte natürlich auch in der Lage sein, seinen Besitzer bzw. Personen, die er bereits kennen gelernt hat, zu identifizieren, um sie beispielsweise mit ihrem Namen anreden zu können. Auf

diese Anforderungen wird daher in Kapitel 5 näher eingegangen. Des Weiteren muss der Roboter gewisse Aktionen eines Menschen deuten können, um festzustellen, ob dieser eventuell mit dem Roboter in Interaktion treten möchte, also einen Interaktionspartner darstellt.

Damit der Roboter eine anwesende Person als Interaktionspartner identifizieren kann, muss er die Fähigkeit besitzen, seine Aufmerksamkeit auf sie zu richten. Anschließend müssen bestimmte Merkmale der betrachteten Person untersucht werden, um feststellen zu können, ob sie wirklich mit dem Roboter zu interagieren wünscht. Für den Fall, dass sich mehrere Personen im Wahrnehmungsbereich des Roboters befinden sollten, muss der Roboter seine Aufmerksamkeit abwechselnd auf diese potentiellen Interaktionspartner richten können. Dabei sollte das System in der Lage sein, diejenige Person zu bevorzugen, die am wahrscheinlichsten den Roboter instruieren möchte. Wird schließlich anhand der Merkmale einer Person ein Interaktionspartner gefunden, so sollte sich der Roboter nicht mehr durch andere Personen ablenken lassen. Der Interaktionspartner ist dann die einzige Person, die dem Roboter Anweisungen geben kann. Dies gilt solange, bis der Interaktionspartner die Interaktion mit dem Roboter beendet. Des Weiteren sollte der Roboter in der Lage sein, seine Aufmerksamkeit dem Benutzer gegenüber anzeigen zu können, damit dieser verifizieren kann, ob sich der Roboter wie gewünscht verhält. Auf die Realisierung dieser Verhaltensweisen wird in Abschnitt 6.1 eingegangen.

Der Roboter soll neben der Interaktion mit einem Interaktionspartner auch Gegenstände lernen können, die ihm vom Benutzer beigebracht werden. Damit der Interaktionspartner in der Lage ist, dem Roboter Gegenstände an verschiedenen Orten zu zeigen, muss er den Roboter zu diesen Objekten führen können. Somit muss der Roboter die Fähigkeit besitzen, dieser Person zu folgen. Dabei ist zu beachten, dass der Roboter, anders als *BUGNOID* [Doi02], nicht nur der nächstbesten Person folgen darf. Das System muss sicherstellen, dass es tatsächlich dem aktuellen Interaktionspartner folgt und diesen nicht mit einer anderen anwesenden Person verwechselt. Für das Folgen einer Person sind außerdem entsprechende Navigationsfähigkeiten notwendig. Beispielsweise muss der Roboter dabei in der Lage sein, unerwarteten Hindernissen auszuweichen.

Um schließlich ein vom Interaktionspartner referenziertes Objekt wahrnehmen zu können, muss der Roboter seine Aufmerksamkeit vom Interaktionspartner auf das Objekt verlagern. Dazu müssen neben der Verarbeitung von sprachlichen Anweisungen auch Zeigegesten erkannt werden können. Hat der Roboter den Ort lokalisiert, an dem sich der zu lernende Gegenstand befindet, so kann dieses Objekt vom System detektiert werden. Da es allerdings nicht möglich ist, dass einem System alle denkbaren Objekte im voraus bekannt sind, muss der Roboter auch unbekannte Objekte lernen können. Dazu ist wiederum eine entsprechende Interaktion mit dem Interaktionspartner nötig. Neben der visuellen Erfassung des zu lernenden Objektes muss der Roboter auch zugehörige sprachliche Informationen akquirieren können. Eine Phrase wie z.B. „Dieses ist meine Lieblingstasse“ gibt nämlich Aufschluss über die Funktion und den Eigentümer des entsprechenden Objektes.

Nachdem der Roboter ein vom Benutzer gezeigtes Objekt und eventuell zugehörige sprachliche Informationen akquiriert hat, muss er dieses Wissen auch speichern können. Dieses Wissen muss dabei strukturiert verwaltet werden, um mögliche Anfragen bezüglich der gespeicherten Objekte

zu ermöglichen. Dem Interaktionspartner muss es beispielsweise möglich sein, den Roboter über dessen Wissen bezüglich eines speziellen Objektes, wie z.B. seiner Tasse, befragen zu können. Das System muss dann in der Lage sein, mit den gegebenen Informationen das entsprechende Objekt unter allen gespeicherten Daten wiederzufinden. Die Fähigkeiten, die sich auf die Verarbeitung von Objekten beziehen, werden in Abschnitt 6.2 behandelt.

Wie bereits erwähnt, ist es für eine natürliche Interaktion unerlässlich, dass der Roboter vom Benutzer gegebene sprachliche Informationen verarbeiten kann. Dazu muss das System Sprache erkennen und verstehen können. Die Sprache sollte dabei von Mikrofonen aufgenommen werden, die sich auf dem Roboter befinden, da die Verwendung eines Headsets nicht sehr intuitiv ist. Auch wenn Sprache die wichtigste Modalität darstellt, um das Verhalten des Roboters durch den Interaktionspartner zu verändern, darf eine sprachliche Anweisung nicht einfach in eine Instruktion umgewandelt werden. Gerade wegen der über einige Entfernung hinweg aufgenommenen Sprache sind die vom Roboter verstandenen Informationen in der Regel mit Unsicherheiten versehen. Der Hauptgrund dafür ist, dass neben einer Anweisung auch eine Vielzahl von Störgeräuschen mit aufgenommen wird. Außerdem ist nicht auszuschließen, dass ein Benutzer mehrdeutige Informationen gibt. Aus diesen Gründen muss der Roboter in der Lage sein, Dialoge mit dem Interaktionspartner zu führen. Bei eventuellen Unsicherheiten oder Uneindeutigkeiten kann das System somit diskriminative Rückfragen stellen. Beispielsweise könnte ein Benutzer den Roboter nach seiner „Tasse“ fragen. Sind daraufhin die Begriffe „Tasse“ und „Tasche“ für den Roboter ungefähr gleich wahrscheinlich, so sollte er den Interaktionspartner fragen, welcher dieser zwei Gegenstände gemeint war. Des Weiteren könnte es sein, dass der Benutzer zwei Tassen besitzt, die sich aber im Wesentlichen durch ihre Farbe voneinander unterscheiden. Wenn dem Roboter beide Tassen bekannt sind, so sollte er den Interaktionspartner z.B. fragen: „Meinst du deine gelbe oder deine rote Tasse?“ Zusätzlich sollten für einen Dialog multimodale Informationen berücksichtigt werden. Befindet sich ein Objekt, welches Gegenstand eines Dialogs ist, im Sichtbereich des Roboters, so können beispielsweise Objekterkennungsergebnisse in den Sprachverstehensprozess einfließen. Insgesamt sollen die Dialoge eine für den Benutzer möglichst intuitive, angenehme und flüssige Interaktion mit dem Roboter erlauben. Auf die zugehörige Realisierung einer solchen Dialogsteuerung wird in Abschnitt 6.3 eingegangen.

Neben den oben genannten Interaktionsfähigkeiten muss der Roboter allerdings auch gewisse Aufgaben autonom erfüllen können. Beispielsweise sollte er nach entsprechender Instruktion in der Lage sein, von einem Raum aus einen weiteren ihm bekannten Raum selbstständig zu erreichen. Dazu werden unter anderem gewisse Navigationsfähigkeiten benötigt, wie z.B. das Vermögen zur Planung eines geeigneten Weges. Die Realisierung solcher Fähigkeiten ist allerdings nicht Bestandteil dieser Arbeit, dennoch sollten sie im System integrierbar sein.

Schließlich müssen die oben genannten Anforderungen nicht nur umgesetzt, sondern auch in einem Gesamtsystem integriert werden. Damit geht auch die Koordinierung aller Funktionalitäten einher. Zusätzlich müssen die mit einer Integration verbundenen strukturellen Aspekte berücksichtigt werden. Dies geschieht daher im folgenden Abschnitt. In Abschnitt 3.3 wird dann schließlich das auf Basis der funktionalen und strukturellen Anforderungen entwickelte Architekturkonzept vorgestellt.

3.2. Strukturelle Anforderungen

Neben den im vorherigen Abschnitt beschriebenen funktionalen Anforderungen müssen auch diverse strukturelle Aspekte, die mit der Software-Entwicklung für einen autonomen mobilen Roboter einhergehen, beim Entwurf einer entsprechenden Architektur berücksichtigt werden.

Eine Architektur sollte nach der Beurteilung von Coste-Manière und Simmons [Cos00] die Entwicklung von Robotersystemen erleichtern, indem das Design und die Implementierung bezüglich der gewünschten Anwendung nützlichen und förderlichen Einschränkungen unterworfen wird. Allerdings dürfen diese Einschränkungen nicht zu restriktiv sein. Als die wichtigsten Gesichtspunkte beim Entwurf werden zum einen die Architekturstruktur und zum anderen der Architekturstil gesehen. Die Architekturstruktur bezieht sich darauf, wie ein Gesamtsystem in Subsysteme unterteilt ist und wie diese Subsysteme miteinander interagieren. Der Architekturstil beschreibt hingegen die Datenverarbeitungskonzepte denen das Gesamtsystem unterliegt. Beispielsweise kann ein System Nachrichten nach dem *Publisher-Subscriber*-Prinzip versenden, während ein anderes System zu Kommunikation eine synchrone *Client-Server*-Variante verwendet [Tan02].

Nach Rosenblatt [Ros95] sollte eine Systemarchitektur auch die Koordinierung und Integration von Subsystemen ermöglichen, die unabhängig voneinander entwickelt werden. Somit dürfen nur minimale Einschränkungen bezüglich der Beschaffenheit von Daten, Repräsentationen und Algorithmen bestehen, die von diesen Subsystemen verwendet werden, um die Aufgaben zu erfüllen, für die sie entwickelt wurden. Des Weiteren arbeiten in einem Robotersystem verschiedene Sensoren inklusive der Komponenten, welche die zugehörigen Daten verarbeiten, mit unterschiedlichen Geschwindigkeiten. Aus diesem Grund sollte es den einzelnen Komponenten ermöglicht werden, asynchron zu operieren, um den Durchsatz und somit auch die Ansprechempfindlichkeit des Gesamtsystems zu maximieren. Weitere entscheidende Gesichtspunkte bei der Entwicklung einer Architektur für einen mobilen Roboter stellen Mechanismen dar, die das Entwickeln, Testen, Fehlerbereinigen und Validieren des Systems erleichtern.

Aus den oben genannten strukturellen Anforderungen an eine Architektur für ein Robotersystem und den Anforderungen, die in der Einleitung dieses Kapitels beschrieben wurden, lassen sich folgende Punkte, die für die Konstruktion eines *Robot Companions* von Bedeutung sind, ableiten:

- **Modularität:** Bei der Entwicklung eines umfangreichen Systems muss berücksichtigt werden, dass in der Regel mehrere Entwickler bzw. Entwickler-Teams beteiligt sind. Ein einzelner Entwickler bzw. jedes einzelne Team beschäftigt sich dabei gewöhnlich nur mit bestimmten Teilaspekten des Gesamtsystems. Entsprechende Komponenten müssen daher im System unabhängig voneinander entwickelt und getestet werden können. Somit muss das System hinreichend modular sein, wobei die individuellen Module in Form von autonomen Agenten nebeneinander existieren müssen.

- **Systemkontrolle:** Wenn die einzelnen Module in einem System parallel und unabhängig voneinander arbeiten sollen, so muss es eine Form von Kontrolle im Gesamtsystem ge-

ben. Anderenfalls wäre es nicht möglich, dass der Roboter letztendlich ein zielgerichtetes Gesamtverhalten aufweist. Dabei ist zu entscheiden, ob eine zentrale oder eine verteilte Kontrollform verwendet werden soll. Es muss berücksichtigt werden, dass sowohl reaktive als auch deliberative Module zu integrieren sind und dass insbesondere der Zugriff auf Hardware-Komponenten präzise koordiniert werden muss.

- **Kommunikation:** Zur Realisierung eines Gesamtsystems müssen die enthaltenen Module miteinander kommunizieren können, um Informationen auszutauschen. Dies muss auch über verschiedene Rechner hinweg ermöglicht werden. Die Kommunikation sollte dabei ausreichend flexibel sein, so dass verschiedenste Daten und Datentypen auf unterschiedliche Weisen (z.B. synchron oder asynchron) transportiert werden können. Zusätzlich sollten einheitliche Schnittstellen die Transparenz der Kommunikation unterstützen.

- **Fehlertoleranz:** Wenn ein komplexes System wie ein *Robot Companion* in einer dynamischen Umgebung agiert, kann das Auftreten von Ausnahmen oder sogar Fehlern nicht ausgeschlossen werden. Daher sollte der Roboter mit solchen Situationen umgehen können und gegebenenfalls den Benutzer über eine bestehende Problematik informieren. Auch für den Fall, dass ein Modul seinen Dienst versagen sollte, muss das System diesen Ausfall kompensieren können, ohne selbst komplett auszufallen.

- **Lastverteilung:** Ein autonomes System kann nur eine begrenzte Menge an Hardware beherbergen. Dadurch sind auch die Rechenkapazitäten des Roboters beschränkt. Berechnungsintensive Module des Systems sollten daher je nach Bedarf aktiviert bzw. deaktiviert werden können. Dieses Vorgehen ermöglicht es, die für einen *Robot Companion* benötigte Menge an Funktionalitäten in einem Gesamtsystem zu integrieren, auch wenn diese nicht alle gleichzeitig ausgeführt werden können.

Die oben genannten Gesichtspunkte werden in den folgenden fünf Unterabschnitten im Einzelnen diskutiert.

3.2.1. Modularität

Analog zu dem bereits genannten Service-Roboter PSR, welcher von Kim et al. [Kim03] entwickelt wurde, sind an der Implementierung eines *Robot Companions* in der Regel mehrere Entwickler beteiligt. Diese müssen unabhängig voneinander in der Lage sein, sowohl bestehende Komponenten zu bearbeiten und auszutauschen als auch neue Komponenten hinzuzufügen. Des Weiteren ist es gewöhnlich nicht möglich, alle geplanten Funktionalitäten des Gesamtsystems gleichzeitig fertig zu stellen. Die Architektur des Roboters muss somit für eine inkrementelle Entwicklung vorgesehen sein und das schrittweise Einbinden von neuen Modulen ermöglichen. Dabei sollte das System auch in soweit skalieren, dass eventuell nachträglich identifizierte Funktionalitäten noch integriert werden können. Trotzdem müssen während des Entwicklungsprozesses einzelne Module im Gesamtsystem getestet werden können, um ihre Funktionsfähigkeit zu

verifizieren. Dazu müssen nicht existente Module simuliert werden können. Insgesamt empfiehlt es sich somit, alle zu entwickelnden Module als autonome Agenten zu konzipieren.

Der Begriff „Agent“ wird im Kontext von autonomen Agenten sehr vielfältig verwendet [Fra97]. Beispielsweise kann ein autonomes Robotersystem in seiner Gesamtheit als ein so genannter Roboteragent betrachtet werden. Aber auch die einzelnen Module in diesem Gesamtsystem können wiederum Agenten, so genannte Software-Agenten, darstellen. Und genau um diese Agenten handelt es sich bei der Entwicklung unabhängiger Module für das Gesamtsystem eines Roboters. Um die Unterschiede zwischen Software-Agenten und „einfachen“ Programmen zu identifizieren, wird in [Fra97] nach der Analyse diverser Agentensysteme folgende Definition entwickelt:

> Ein autonomer Agent ist ein System, welches sich in einer Umgebung befindet und gleichzeitig ein Teil dieser Umgebung ist. Dieses System nimmt über die Zeit seine Umgebung wahr und agiert in ihr, wobei es seine eigenen Ziele verfolgt. Infolgedessen beeinflusst der Agent, was er in Zukunft wahrnimmt.

Auch wenn diese Definition eines autonomen Agenten weniger formal ist als z.B. die Definition, die sich in [Goo95] findet, so ist sie doch hinreichend, um die Selbstständigkeit als einen Kernpunkt solcher Agenten zu identifizieren. Offensichtlich lässt sich diese am angemessensten realisieren, indem jeder Agent im System auf einen eigenen Prozess abgebildet wird.

Nachdem nun die Anforderungen an die Funktionsweise der zu entwickelnden Module gestellt wurden, verbleibt die Frage über deren Granularität. Dieser Gesichtspunkt beeinflusst insbesondere die Skalierbarkeit des Gesamtsystems. Nach Orebäck und Christensen [Ore03] sollte eine mittlere Granularität verwendet werden. Sind nämlich die Module bezüglich des Umfangs der zu realisierenden Funktionalitäten zu klein, so könnte die Menge der Module so groß werden, dass die Wartbarkeit des Systems darunter leidet. Sind die Module hingegen zu umfangreich konzipiert, so besteht wiederum die Gefahr, dass die Module selbst zu unübersichtlich werden. Als grober Richtwert ist eine Funktionalität pro Modul wünschenswert, da dieses Vorgehen das Austauschen von einzelnen Funktionalitäten auf einfache Weise und unabhängig voneinander ermöglicht. Bezüglich der im vorherigen Kapitel betrachteten Kontrollparadigmen lässt sich dieses Konzept in einer hybriden Architektur am einfachsten realisieren. Insbesondere in Behavior-basierten Architekturen fällt die Granularität oft feiner aus, da sich bestimmte Funktionalitäten erst aus der Interaktion mehrerer Behaviors ergeben.

3.2.2. Systemkontrolle

Damit der Roboter in der Lage ist, seine Aufgaben zu erfüllen, müssen alle Module des Systems einer Kontrolle unterliegen. Nur wenn das System als Gesamtheit koordiniert werden kann, ist es dem Roboter auch möglich, bestimmte Ziele zu verfolgen. Die primär zu fällende Entscheidung ist, ob die resultierende Kontrolle zentralisiert oder verteilt in der Roboterarchitektur repräsentiert sein soll. Dabei ist zu beachten, dass sich eine zentralisierte Kontrolle am angemessensten in

einem hybriden System integrieren lässt, während eine verteilte Kontrolle mehr einem Behavior-basierten System entspricht.

Die Frage nach der geeigneten Kontrollform für einen autonomen Roboter wurde ebenfalls von Rosenblatt [Ros95] diskutiert. Seiner Meinung nach müssen bei der Wahl der Kontrollform für ein Robotersystem, wie in jedem komplexen System, Kompromisse geschlossen werden. Auf der einen Seite stehen dabei die Kohärenz, Korrektheit und relative Einfachheit eines zentralisierten Systems, auf der anderen Seite die Ansprechempfindlichkeit, Robustheit und Flexibilität eines verteilten Systems.

Rosenblatt sieht in zentralisierten Architekturen die Fähigkeit zur kohärenten Koordinierung von mehreren Ziele und Bedingungen in einer komplexen Umgebung. Allerdings ist eine streng zentralisierte Architektur nicht angemessen für Realzeitsysteme, die vorwiegend in dynamischen und unsicheren Umgebungen operieren. Durch den Flaschenhals, den eine zentrale Kontrolle darstellt, wird einerseits die Ansprechempfindlichkeit des Systems gehemmt. Andererseits leiden zentralisierte Architekturen darunter, dass sie öfter schwerwiegenden Fehlern unterliegen als verteilte Systeme. Falls nämlich eine Komponente eines zentralisierten Systems ausfällt, so quittiert in der Regel das gesamte System seinen Dienst. Rein zentralisierten Architekturen mangelt es oft an Modularität, so dass diese anspruchsvoller in Entwicklung und Wartung sind. Außerdem ist dadurch ihre Wiederverwendbarkeit stark eingeschränkt.

Dezentralisierte Architekturen bieten nach Rosenblatt Vorteile in den Punkten Reaktionsfähigkeit, Flexibilität und Robustheit. Jedoch hat auch ein komplett verteiltes System gewisse Nachteile. Im Allgemeinen spiegeln die vom System erzeugten Verhaltensweisen nicht die Gesamtheit der Ziele und Bedingungen wieder, denen das System zu jedem Zeitpunkt unterliegt, da kein übergreifender Kontrollmechanismus vorhanden ist. Somit führen die erzeugten Verhaltensweisen gewöhnlich zu einem stark suboptimalen Leistungsverhalten des Gesamtsystems. Zusätzlich sind die Interaktionen sowohl zwischen den einzelnen Modulen als auch zwischen dem Gesamtsystem und seiner Umgebung weniger vorhersagbar und daher auch schwieriger zu verstehen und zu modifizieren.

Nach der obigen Diskussion wird deutlich, dass in einem so umfangreichen System wie einem *Robot Companion* die Koordinierung der Interaktion zwischen den zu integrierenden Modulen eine besondere Herausforderung darstellt. Es scheint, dass zur Realisierung des Gesamtsystems eine zentrale Kontrolle erforderlich ist. Diese Form der Kontrolle würde auch eine angemessene Repräsentation aller Ziele des Systems erlauben. Allerdings darf es sich um keine rein zentralisierte Architektur handeln, da diese nicht ausreichend reaktiv ist. Eine schnelle Reaktionsfähigkeit wird insbesondere deshalb benötigt, da sich ein *Robot Companion* in dynamischen Umgebungen bewegen soll, in denen sich auch Menschen befinden. Somit wäre es wünschenswert, wenn die zwei oben genannten Kontrollformen miteinander kombiniert werden könnten, um die Vorteile beider Ansätze nutzen zu können. Als Ergebnis bietet sich eine Architekturform an, die zwar einer zentralen Kontrolle untersteht, aber den einzelnen Modulen ausreichend Eigenständigkeit gewährt, so dass diese, ähnlich wie Behaviors, eine gewisse Autonomie besitzen. Die Aufgabe der zentralen Kontrollinstanz besteht dann ausschließlich in der Überwachung und der Parametrisierung der übrigen Module.

Schließlich sollte die Kontrolle über die Hardware des Robotersystems mittels einer abstrakten Schnittstelle erfolgen. Dies ermöglicht es dem System, unabhängig von der Roboterbasis und den verwendeten Sensoren zu sein. Des Weiteren würde eine entsprechend standardisierte Schnittstelle nicht nur den Austausch von einzelnen Hardware-Komponenten erleichtern, sondern auch die Portabilität des Gesamtsystems auf andere Hardware-Plattformen erhöhen.

3.2.3. Kommunikation

In einem Gesamtsystem, welches aufgrund seiner hohen Komplexität eine entsprechende Anzahl von Modulen enthält, ist auch der Kommunikationsaufwand zwischen den Modulen nicht unerheblich. Aus diesem Grund sollte die Kommunikation transparent und einheitlich für unterschiedliche Datentypen sein, so dass deren Verwendung so einfach wie möglich ist. Außerdem ist so sichergestellt, dass Modifikationen und Erweiterungen im System nur einen geringen Aufwand bedeuten. Um diese Anforderungen zu erfüllen wird in den meisten aktuellen Architekturen eine Kombination aus synchronem und asynchronem Kontroll- und Datenfluss verwendet. Asynchrone Prozesse sind dadurch charakterisiert, dass sie lose miteinander verbunden sind und durch Ereignisse gesteuert werden, ohne dabei strengen Ausführungsfristen zu unterliegen. Synchrone Prozesse sind hingegen sehr eng aneinander gekoppelt. Sie sind in der Regel gleich getaktet und unterliegen häufig harten Realzeitbedingungen.

Ein Sachverhalt, der nicht unberücksichtigt bleiben darf, betrifft die Art der Kommunikation zwischen den Modulen im System. Diesen Aspekt diskutiert auch Rosenblatt in [Ros95]. Er stellt fest, dass eine direkte Kommunikation zwischen den Modulen dem Systementwickler ein hohes Maß an Kontrolle über den Arbeitsablauf des Systems bietet. Dieses kann gewünscht sein, falls Module speziell dazu konstruiert sind, sehr eng miteinander zu interagieren, wie es z.B. in der *Subsumption*-Architektur [Bro86] der Fall ist. Eine direkte Kommunikation erschwert jedoch die Erweiterung und die Modifikation eines Systems und beschränkt das Ausmaß, zu dem das System oder darin enthaltene Komponenten wiederverwendet werden können. Eine indirekte Form der Kommunikation lässt sich durch gewisse Hilfsmittel, wie z.B. ein *Blackboard* [Hay85] oder einen *Broadcast*-Mechanismus, realisieren. Solche Instrumente bieten eine Abstraktionsschicht zwischen den Modulen im System und vereinfachen die Aufgabe, Module im System auszutauschen oder neue Module dem System hinzuzufügen. Diese Art der Flexibilität birgt aber einen erhöhten Aufwand, der sich in einem reduzierten Durchsatz und somit auch in einer geringeren Effizienz des Systems widerspiegelt.

Speziell unter Behavior-basierten Architekturen existiert noch eine weitere Form der indirekten Kommunikation [Ros95]. Hierbei werden überhaupt keine Informationen innerhalb des Systems, also von einem Behavior zu einem anderen Behavior, verschickt. Stattdessen verwenden solche Architekturen eine Befehlsfusion, wie z.B. *Motor Schemas* [Ark89] oder *Fuzzy Control* [Yen95]. Die einzelnen Behaviors kommunizieren dabei ihre Intentionen ausschließlich zu einem zentralen Koordinator, so dass diese Module nicht nur vollständig voneinander getrennt entwickelt, sondern auch ausgeführt werden können. Dennoch stellt in einem solchen System das zentrale Koordinierungsmodul wiederum einen Flaschenhals dar. Außerdem könnten in anderen Syste-

men gerade Vorteile dadurch erlangt werden, dass sich Behaviors gegenseitig mit Informationen versorgen. Dies kann allerdings nicht ohne direkte Modulkommunikation realisiert werden.

Das Design des Kommunikationssystems in einer Architektur beeinflusst, neben der Art der Modularisierung (vgl. Abschnitt 3.2.1), auch die Validierbarkeit eines Gesamtsystems. Die Validierung eines Systems beinhaltet nach Coste-Maniere und Simmons [Cos00] sowohl das Testen von einzelnen Modulen als auch das Testen des Systems in seiner Gesamtheit. Bevor ein komplexes Robotersystem allerdings fertig gestellt und als Einheit getestet werden kann, müssen seine Komponenten separat getestet werden können. Jedoch hängt die Antwort einer Komponente üblicherweise von dem Verhalten anderer Komponenten im System ab. Daher müssen Tester in der Lage sein, die anderen Komponenten durch Module mit identischer Funktionalität zu ersetzen, welche aber wesentlich einfacher konstruiert sind. Erst dann kann geprüft werden, ob das zu testende Modul im Kontext des Systems korrekt reagiert. Um diese Art der Validierung durch die Roboterarchitektur zu unterstützen, sollte die Kommunikation daher eine Form von anonymem Datentransfer, wie z.B. das *Publisher-Subscriber*-Modell [Tan02], verwenden.

Eine weitere fundamentale Notwendigkeit für ein Kommunikationssystem einer Roboterarchitektur ist die Fähigkeit, den Austausch von Daten auch zwischen Modulen zu ermöglichen, die über verschiedene Rechner verteilt sind, um schnellere Antwortzeiten des Gesamtsystems zu ermöglichen. Da jedoch die meisten Forscher aus dem Bereich der Robotik keine Experten für *Middleware* sind, ist eine Verwendung von nativen, wie z.B. auf CORBA basierende Lösungen, ausgeschlossen. Aus diesem Grund wird in Kommunikationssystemen, welche auf CORBA aufbauen [Blu03, Kno04], versucht, die Komplexität dieser *Middleware* mit Hilfe von domänenspezifischen Klassenbibliotheken zu kapseln, was aber deren Verwendung in anderen Systemarchitekturen oftmals erschwert. Daher empfiehlt es sich, ein möglichst generisches Kommunikationssystem zu verwenden, welches es den Entwicklern eines Robotersystems auf unkomplizierte Weise ermöglicht, ihre Module in einer verteilten Architektur zu integrieren. Dieses Kommunikationssystem muss folglich möglichst einfach zu verwenden sein und weitgehend auf standardisierte Techniken zurückgreifen.

Neben dem Transferieren von Daten könnte das Kommunikationssystem weitere Aufgaben übernehmen. Eine Möglichkeit wäre, die Gültigkeit von zu versendenden Daten zu überprüfen. Dieses bietet sich an dieser Stelle an, da so weder der Sender, noch der Empfänger eine Validierung der Daten modulintern durchführen muss. Allerdings müssen dazu die Regeln, die zur Überprüfung dieser Daten herangezogen werden, von allen beteiligten Entwicklern einsehbar und auch modifizierbar sein, damit der Kontrollmechanismus ausreichend transparent ist. Diese Form der zentralen Validierung bietet nicht nur mehr Übersicht über die Kommunikationsdaten, sondern verhindert auch eine versehentliche doppelte und damit eventuell widersprüchliche Überprüfung dieser Informationen. Außerdem ist es möglich, einen im Kommunikationssystem integrierten Kontrollmechanismus global zu verwalten. Beispielsweise kann die Datenvalidierung für Testzwecke aktiviert sein, aber im Normalbetrieb deaktiviert werden, um Rechenzeit zu sparen.

Des Weiteren kann das Kommunikationssystem das Testen des Gesamtsystems zusätzlich aktiv unterstützen, indem entsprechende Werkzeuge bereitgestellt werden, die z.B. die Protokollierung des Netzwerkverkehrs zur Systemlaufzeit erlauben.

3.2.4. Fehlertoleranz

Ein *Robot Companion* sollte so robust wie möglich funktionieren und muss somit möglichst fehlertolerant arbeiten, da gewisse Probleme nicht ausgeschlossen werden können. Einerseits kann sich ein Roboter mit umgebungsbedingten Ausnahmen konfrontiert sehen, andererseits können Fehler auf Seiten des Systems auftreten. Diese Probleme sollten nach Möglichkeit vom System behoben werden können.

Ausnahmebehandlung

Eine Ausnahme ist allgemein betrachtet ein Zustand, welcher einen Roboter bei der Ausführung seiner eigentlichen Aufgabe behindert. Die Detektion und die Behandlung von Ausnahmezuständen ist daher ein wichtiger Bestandteil in einem Robotersystem [Cos00]. Ausnahmen können während des Betriebs in nahezu jeder Situation und zu jedem Zeitpunkt auftreten. Allerdings geschieht dies nur mit geringer Wahrscheinlichkeit. Eine Architektur kann dazu beitragen, angemessene Strategien zur Reaktion auf solche Ausnahmen zu spezifizieren, indem normales von ausnahmebedingtem Verhalten unterschieden wird. Dazu muss es dem Entwickler möglich sein, korrektes und inkorrektes Verhalten auf modulare und inkrementelle Weise zu definieren.

In einigen Architekturen, wie z.B. in der *Subsumption*-Architektur [Bro86], existieren keine Ausnahmen, so dass jeder Zustand auf die gleiche Weise behandelt wird. In anderen Architekturen werden Ausnahmen speziell behandelt und sind daher auf andere Weise spezifiziert als normales Verhalten. Zur eigentlichen Behandlung der Ausnahmen werden oft besondere Routinen verwendet. Diese Routinen werden im Ausnahmefall angesprungen und kehren nach ihrer Ausführung wieder zurück, so dass anschließend der unterbrochene Programmablauf wieder aufgenommen werden kann. Beispiele für solche Architekturen sind TCA [Sim94] und ORCCAD [Bor98]. In diesen Architekturen sind Ausnahmen durch Namen definiert und können hierarchisch angeordnet werden. Dies erlaubt es dem Entwickler, sowohl individuelle als auch generelle Ausnahmebehandlungen zu spezifizieren. Dabei können individuelle Ausnahmebehandlungen nur auf eingeschränkte Situationen angewendet werden. Beispielsweise kann eine spezielle Bewegung wiederholt werden, weil sie bei der ersten Ausführung nicht den gewünschten Effekt erzielt hat. Generelle Ausnahmebehandlungen haben hingegen einen breiteren Anwendungsbereich und erlauben z.B. den Abbruch der aktuellen Tätigkeit.

Detektion von technischen Fehlern

Neben den oben genannten Ausnahmezuständen können auch technische Probleme in einem Robotersystem auftreten. Speziell in einem modular gestalteten System ergeben sich primär zwei Arten von Problemen: Entweder fällt ein Modul (oder mehrere) aus oder die Kommunikation zwischen zwei Modulen (oder mehreren) bricht zusammen. Der letztere Fall sollte allerdings vom Kommunikationssystem selbstständig festgestellt und über die zugehörige Schnittstelle von

den betroffenen Modulen wahrgenommen werden können. Somit wären die einzelnen Module in der Lage, selbstständig auf die Problematik zu reagieren.

Ein Modul kann hingegen auf unterschiedliche Weise ausfallen. Die einfachste Variante ist ein Totalausfall, d.h. der mit dem Modul assoziierte Prozess beendet sich auf unvorhergesehene Weise. In diesem Fall würden andere Module, die mit dem ausgefallenen Modul kommunizieren, keine Antworten mehr erhalten. Idealerweise würde das Kommunikationssystem auch diesen Fall abfangen können, da er vom System erkannt werden kann. Falls ein Modul allerdings nur bedingt ausfällt, weil es sich z.B. in einer Endlosschleife befindet, so kann das Kommunikationssystem diesen Ausfall in der Regel nicht feststellen. Dennoch würden auch hier die angebundenen Module keine Antworten mehr erhalten. Um einen solchen Ausfall zu detektieren gibt es verschiedene Möglichkeiten. Eine Möglichkeit ist, dass das empfangene Modul einen „Wächter“ instanziiert, der die Kommunikation beobachtet und bei einer Unregelmäßigkeit Alarm gibt.

Um eine Unregelmäßigkeit in der Kommunikation bemerken zu können, müssen Informationen über den Datenfluss vorliegen. Solche Informationen können z.B. durch die Überwachung eines Kommunikationskanals erworben werden [Kaw00a]. Diese Art des Lernens ist möglich, da angenommen werden kann, dass sich in einem Multiagentensystem gewisse Datenflussmuster ergeben. Aus diesen Mustern lassen sich zeitliche Charakteristika ableiten, die für die korrekte Funktionsweise eingehalten werden müssen. Dabei sind insbesondere die Verzögerungen zwischen zwei aufeinander folgenden Nachrichten interessant. Solche Verzögerungen können statistisch erfasst und in einem Histogramm dargestellt werden. Nachdem das System auf diese Weise trainiert wurde, können Fehler in der Kommunikation detektiert werden. Im einfachsten Fall wird angenommen, dass ein Problem vorliegt, wenn das Maximum aller zuvor gemessenen Verzögerungen übertroffen wird. Allerdings ist diese Strategie nicht in allen Fällen erfolgreich [Kaw00b]. Problematisch ist z.B. die Überwachung von Kommunikationskanälen, über die Nachrichten mit nur geringer Frequenz versendet werden, da hier ein Ausfall der Kommunikation erst mit einiger Verzögerung entdeckt werden kann. Dies ist besonders kritisch, wenn die angebundenen Module für die Ansteuerung von Aktuatoren auf dem Roboter verantwortlich sind.

Eine sicherere und gleichzeitig einfacher zu realisierende Methode, um ein Kommunikationsproblem zu detektieren, ist dadurch gegeben, dass ein Sender in regelmäßigen Abständen Nachrichten verschickt. Einerseits kann es sich bei solchen Nachrichten um einen so genannten *„Ping“* handeln, anhand dessen der Empfänger verifizieren kann, ob der Sender noch korrekt arbeitet oder nicht. Wichtig hierbei ist, dass der *Ping* nicht in der Kommunikationsschicht eines Moduls erzeugt, sondern von dessen Hauptschleife initiiert wird. Nur in diesem Fall kann ein Ausfall des Senders von dem Empfänger detektiert werden. Andererseits kann die Kommunikation zwischen zwei Modulen auch so strukturiert werden, dass ständig informationstragende Nachrichten versendet werden. Dazu werden anstelle von Nachrichten der Art „Starte Aktion X“ und „Beende Aktion X“ ständig Nachrichten in der Form „Führe Aktion X aus“ versendet. Nur wenn und solange diese Nachrichten regelmäßig vom Empfänger wahrgenommen werden, wird die entsprechende Aktion ausgeführt. Das ständige Senden dieser Art von Nachrichten wird als *„Continual Messaging“* [Nic02] bezeichnet. Allerdings ist zu berücksichtigen, dass beide Varianten des regelmäßigen Versendens von Nachrichten den Netzwerkverkehr erhöhen.

Neben der oben genannten Problematik, dass aufgrund eines Fehlers ein Empfänger keine Daten mehr erhält, kann es vorkommen, dass ein Sender ungültige Ergebnisse erzeugt. Sollten die zu versendenden Daten außerhalb entsprechender Definitionsbereiche liegen, so sind auch solche Fehler detektierbar. Dies kann wie bereits in Abschnitt 3.2.3 diskutiert, durch das Kommunikationssystem geleistet werden. Durch diesen Mechanismus kann somit verhindert werden, dass sich ein Fehler aufgrund von ungültigen Daten im System fortpflanzt und zu einem schwerwiegenden Ausfall führt. Ein Modul kann aber auch auf eine solche Weise ausfallen, dass es semantisch falsche Ergebnisse erzeugt, die allerdings alle syntaktischen Richtlinien erfüllen. Beispielsweise könnte ein perzeptuelles System, welches Messwerte eines Laser-Entfernungsmessers bereitstellt, für alle Winkel ein und denselben Wert liefern, was offensichtlich nicht korrekt sein kann. Auch wenn dieser spezielle Fall einfach abgefangen werden könnte, so kann doch im Allgemeinen nicht automatisch verifiziert werden, ob ein derartiger Fehler vorliegt.

Behandlung von technischen Fehlern

Sollte ein technischer Fehler auftreten, so wäre es wünschenswert, wenn der Roboter nicht komplett seinen Dienst versagen würde, sondern in irgendeiner Form weiterarbeiten könnte. Dies bedeutet, dass alle noch intakten Module im System mit dem vorliegenden Problem umgehen können müssen.

Um es Modulen zu erleichtern, Ausfälle der Kommunikation oder anderer Module zu bewältigen, kann ein spezieller Mechanismus verwendet werden. Dazu werden zu versendende Nachrichten, auf welche ein Modul eine Antwort erwartet, mit einer Gültigkeitsdauer versehen. Dieses Vorgehen verhindert, dass sich bei einem temporären Problem, wie z.B. bei einem vorübergehenden Leistungseinbruch, größere Mengen von Daten bei dem betroffenen Modul anhäufen. Nachdem das Problem behoben worden ist, können dann nämlich veraltete Nachrichten identifiziert und verworfen werden. Somit werden Daten nicht unnötigerweise verarbeitet und weitere Leistungseinbußen verhindert.

Falls nun ein Modul bis zum Gültigkeitsende der versendeten Anfrage keine zugehörige Antwort erhalten haben sollte, so gilt diese als verloren und es können entsprechende Maßnahmen ergriffen werden. Beispielsweise könnte die Komponente des Systems, die für das Führen von Dialogen mit einem Benutzer verantwortlich ist, eine Anfrage an das Modul stellen, welches die Wissensbasis des Roboters verwaltet. Würde die Dialogkomponente nun keine Antwort vom System erhalten, so könnte sie den Benutzer über die Verzögerung informieren, bevor z.B. die Anfrage an die Wissensbasis wiederholt wird.

Sollte das aktuelle Problem des Systems persistent sein und ein Modul identifiziert werden können, welches dafür verantwortlich ist, so sollte es im Idealfall beendet und neu gestartet werden können, um den Normalzustand im Gesamtsystem wiederherzustellen. Ist das Problem allerdings nicht auf diese Weise zu lösen, so sollte der Benutzer soviel Informationen wie möglich erhalten, um bei der Lösung des Problems behilflich sein zu können. Im schlimmsten Fall sollte der Roboter den Benutzer bitten können, einen Techniker zu rufen.

3.2.5. Lastverteilung

Insbesondere auf einem autonomen Robotersystem sind die Rechenressourcen sehr beschränkt. Bei der Entwicklung einer entsprechenden Architektur muss deshalb darauf geachtet werden, dass diese Ressourcen möglichst optimal ausgeschöpft werden.

In der Regel müssen nicht alle zu integrierenden Module auf einem Roboter zur gleichen Zeit laufen. Beispielsweise darf ein Modul deaktiviert sein, wenn die von dem Modul produzierten Daten im System nicht benötigt werden. Aus diesem Grund werden in der zur Roboteransteuerung entwickelten Architektur DCA [Pet01] zu Beginn zwar alle vorhandenen Module gestartet, aber nicht aktiviert. Dies geschieht erst, wenn sie tatsächlich benötigt werden. Sollten die Daten eines Moduls keine Verwendung mehr finden, so kann es, auch zeitweise, wieder deaktiviert werden. Die Entscheidung, wann ein Modul aktiv sein muss bzw. inaktiv sein sollte, hängt einerseits von der Funktionalität des Moduls ab und andererseits auch von dem Kontext, in dem sich der Roboter zum jeweiligen Zeitpunkt befindet [Top04a]. Beispielsweise muss ein Modul zur Gestenerkennung erst dann aktiviert werden, wenn der Benutzer dem Roboter etwas zeigen möchte. Eine noch dynamischere Variante würde es dem System sogar ermöglichen, den Rechenzeitbedarf einzelner Module zu regeln, um die Rechenkapazität auf alle aktiven Module optimal bezüglich der vom Roboter zu erfüllenden Aufgabe verteilen zu können.

Falls das System des Roboters über mehrere Rechner verfügen sollte, so müssen zusätzlich die einzelnen Module auch auf diesen Rechnern verteilt werden. Dabei sind eventuelle Abhängigkeiten zwischen den individuellen Modulen zu berücksichtigen. Allerdings sind solche Abhängigkeiten in der Regel bereits zum Entwurfszeitpunkt bekannt. Beispielsweise können zwei Module, die nicht für den gleichzeitigen Betrieb vorgesehen sind, auf einem Rechner untergebracht werden, da sie sich nicht gegenseitig stören können. Es muss bei der Verteilung der Module aber auch der Datenfluss zwischen den einzelnen Modulen berücksichtigt werden. Beispielsweise kann es zwei Module geben, die gewöhnlich zur gleichen Zeit aktiv sind und währenddessen intensiv Daten austauschen. In diesem Fall ist es üblicherweise erforderlich, dass diese Module trotzdem auf einem Rechner untergebracht werden, obwohl sie immer zeitgleich laufen. Ansonsten kann die Kommunikation zwischen den Modulen dazu führen, dass nicht nur die Module selbst, sondern auch das gesamte System aufgrund des erhöhten Netzwerkverkehrs ausgebremst wird. Aus demselben Grund ist es in der Regel notwendig, dass ein Modul, welches die Daten eines Sensors verarbeitet, auch auf dem Rechner verortet wird, an dem der entsprechende Sensor angeschlossen ist. Allerdings kann bei dem Entwurf des Roboters die Zuordnung der Sensoren zu den vorhandenen Rechnern berücksichtigt werden, um eine ausgewogene Verteilung solcher perzeptuellen Systeme im Gesamtsystem zu gewährleisten. Für alle verbleibenden Module kann in der Regel einfach auf empirische Weise bestimmt werden, auf welchem Rechner sie sich so harmonisch wie möglich bezüglich des Gesamtsystems verhalten.

Die oben beschriebene manuelle Zuteilung der zu integrierenden Module auf die vorhandenen Rechner ist umso schwieriger, je mehr Rechner und Module auf einem Roboter verwendet werden. Sie kann daher auch zu stark suboptimalen Ergebnissen führen. Ein weiterer Grund ist, dass in einem umfangreichen Gesamtsystem die Anzahl der aktiven Module pro Rechner je nach

Kontext stark variieren kann. Dadurch kann es sich ergeben, dass es sinnvoller ist, bestimmte Module in bestimmten Situationen auf einem anderen Rechner als dem sonst üblichen zu starten. Um diese Art der dynamischen Lastverteilung zu automatisieren, wurden so genannte „mobile Agenten“ entwickelt [Abe98]. Diese Agenten erweitern die Vorteile von gewöhnlichen Agenten, wie z.B. Autonomie, Flexibilität, Wartbarkeit und Fehlertoleranz, um die Eigenschaft der Mobilität. Darunter ist zu verstehen, dass mobile Agenten ihren Dienst auf einem Rechner einstellen, sich über das Netzwerk auf einen anderen Rechner bewegen und dort ihren Dienst wieder aufnehmen können. Zu entscheiden, welcher Agent zu welchen Zeitpunkt auf welchem Rechner laufen soll, ist allerdings sehr aufwändig, da neben der Auslastung der einzelnen Rechner auch hier der Datenfluss zwischen den Agenten betrachtet werden muss. Da sich auf autonomen mobilen Robotern in der Regel nur eine relativ überschaubare Anzahl von Rechnern und Modulen befindet, lohnt sich dieser Aufwand zur Laufzeit nur sehr selten. Dann kann aber in den meisten Fällen von einer kontextabhängige Analyse des Systems zur Entwicklungszeit profitiert werden.

3.3. Flexibles Architekturkonzept für Robot Companions

Nach dem Studium bestehender Roboterarchitekturen sowie der Auswertung funktionaler und struktureller Anforderungen an einen *Robot Companion*, fiel die Wahl einer entsprechenden Kontrollarchitektur auf eine hybride Variante. Eine solche Architektur stellt die beste Methode dar, ein System, das autonome Kontrolle und Fähigkeiten zur Mensch-Roboter-Interaktion miteinander verbindet, auf flexible Weise zu organisieren [Kno04]. Außerdem bietet eine hybride Architektur im Vergleich zu einer Behavior-basierten Architektur eine höhere Flexibilität bezüglich der Granularität zu integrierender Module und schließlich kann sie auch am einfachsten um zusätzliche Module und Funktionalitäten erweitert werden. Somit werden die Anforderungen bezüglich der Modularität (vgl. Abschnitt 3.2.1) berücksichtigt.

Das Konzept der im Rahmen dieser Arbeit entwickelten Architektur [Kle04] basiert im Prinzip auf einer Drei-Ebenen-Architektur. Diese Architekturform besteht im Wesentlichen aus drei Mechanismen: einem Mechanismus zur Ausführung von zeitintensiven deliberativen Berechnungen, einem reaktiven Planausführungsmechanismus und einem reaktiven Feedback-Kontrollmechanismus [Gat98]. Einzelne Komponenten dieser Mechanismen werden gewöhnlich in Form von separaten Prozessen betrieben und lassen sich daher optimal als autonome Agenten implementieren, um eine agentenbasierte Architektur zu realisieren. Damit die Zuordnung dieser Komponenten zu ihren Mechanismen auf übersichtliche Weise dargestellt werden kann, gliedert sich die Architekturform entsprechend in eine deliberative, eine intermediäre und eine reaktive Ebene. Ein Gesamtüberblick über das entwickelte Architekturkonzept ist in Abbildung 3.1 gegeben.

Der wesentlichste Bestandteil bezüglich der Struktur des hier vorgestellten Architekturkonzeptes ist eine zentrale Koordinierungskomponente, der so genannte „Execution Supervisor“ (siehe Abschnitt 3.3.1). Diese Komponente repräsentiert den reaktiven Planausführungsmechanismus

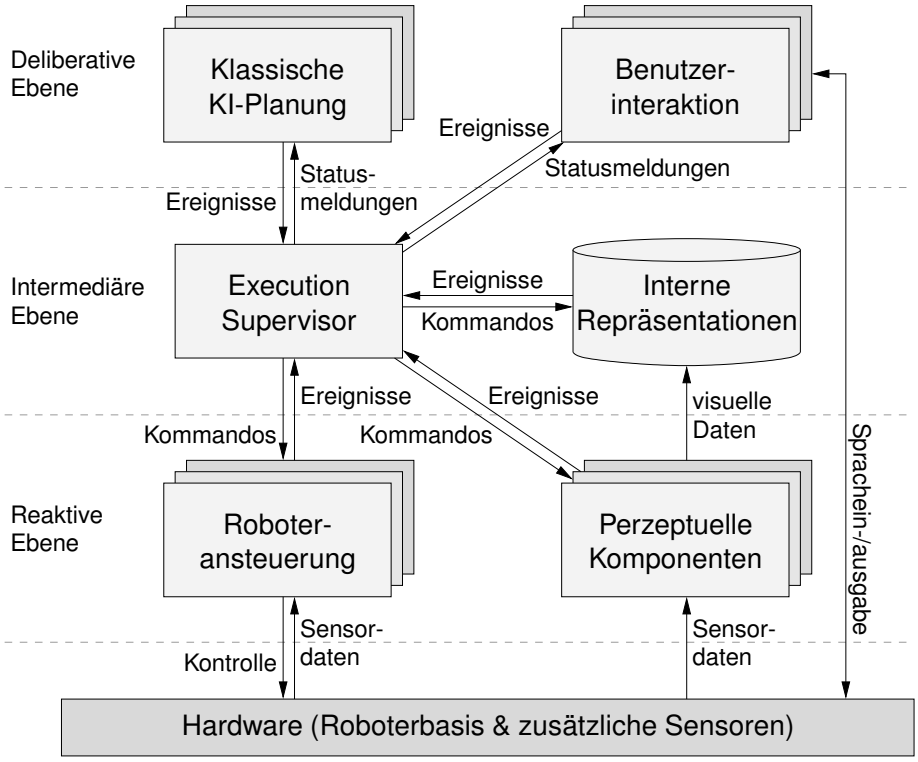


Abbildung 3.1.: Konzeptuelles Modell für die Architektur eines *Robot Companions*.

der Drei-Ebenen-Architektur und ist daher auf der intermediären Ebene vorzufinden, wo sie für die Koordinierung aller angebundenen Module sorgt. Komponenten, die zur Roboteransteuerung dienen, repräsentieren den reaktiven Feedback-Kontrollmechanismus und sind somit in der reaktiven Ebene platziert. Das Design der Roboteransteuerung hängt von den gewünschten Fähigkeiten des Gesamtsystems ab und kann z.B. auch auf der Verwendung von Behaviors beruhen. Die Roboteransteuerung hat direkten Zugriff auf die Hardware des Roboters und sollte durch den Execution Supervisor konfigurierbar sein, um verschiedene Verhaltensweisen zur Erfüllung der gewünschten Aufgaben realisieren zu können. Im Gegensatz zu den reaktiven Modulen befinden sich Komponenten, die klassische Planungsaufgaben übernehmen, auf der deliberativen Ebene, da diese Module den Mechanismus zur Ausführung von zeitintensiven deliberativen Berechnungen bilden. Die Ausgaben der Planungskomponenten stellen gewöhnlich Aktionsfolgen dar, die vom System abgearbeitet werden müssen. Die entsprechenden Aufgaben, für die Pläne generiert werden müssen, werden in der Regel durch die Interaktion mit einem Benutzer initiiert.

Zur Erfassung von Instruktionen, die durch einen Interaktionspartner gegeben werden, befinden sich neben den Planungskomponenten entsprechende Module zur Benutzerinteraktion auf der deliberativen Ebene. Diese Module sind ebenfalls auf der obersten Ebene verortet, da sie, ähnlich wie Komponenten zu Planung, langen Arbeitszyklen unterliegen. Falls sich durch die Interaktion mit einem Benutzer der Bedarf zur Planung ergibt, erhält die entsprechende Komponente die dazu notwendigen Informationen nicht auf direktem Wege, sondern über den Execution Super-

visor. Auf diese Weise kann der Execution Supervisor sicherstellen, dass sich die Instruktionen des Benutzers auch mit dem Zustand des Gesamtsystems vereinen lassen. Zu den Benutzerinteraktionsmodulen zählt z.B. eine Komponente, die für das Führen von Dialogen mit einem Interaktionspartner verantwortlich ist. Dabei ist es das Ziel, dass alle Informationen vom Benutzer akquiriert werden, die notwendig sind, um eine gewünschte Aufgabe erfüllen zu können. Dazu muss dieses Modul Spracheingaben verarbeiten und entsprechende sprachliche Ausgaben erzeugen. Da dieses Benutzerinteraktionsmodul direkt mit dem Execution Supervisor verbunden ist, kann es auch dazu genutzt werden, um Probleme, die in Modulen der reaktiven Ebene auftreten mögen, durch einen Dialog mit dem menschlichen Interaktionspartner aufzulösen. Zu diesem Zweck werden entsprechende Anfragen von der reaktiven Ebene über den Execution Supervisor an dieses Benutzerinteraktionsmodul weitergeleitet. Ein zugehöriges Diskussionsergebnis würde schließlich den umgekehrten Weg zu dem Modul finden, welches die ursprüngliche Anfrage gestellt hat.

Das Architekturkonzept wird schließlich durch perzeptuelle Komponenten in der untersten Ebene und durch ein Modul zur Speicherung interner Repräsentationen in der intermediären Ebene vervollständigt. Die Aufgabe der perzeptuellen Komponenten ist es, Sensordaten zu verarbeiten, um Informationen über die Umwelt des Roboters zu gewinnen. Diese Informationen können zusammen mit zusätzlichen Daten, welche sich z.B. aus einem Dialog mit einem Benutzer ergeben können, als Wissen über die Umgebung vom System gespeichert werden. Damit diese Daten schließlich für alle anderen Module im System zur Verfügung stehen, ist das Modul zur Speicherung interner Repräsentationen auch an den Execution Supervisor angebunden.

Wie in Abbildung 3.1 angedeutet, können zur Erfüllung unterschiedlicher Aufgaben auf der deliberativen und der reaktiven Ebene verschiedenste Komponenten integriert werden. Um ein neues Modul in das gewünschte Zielsystem einbinden zu können, muss in der Regel nur eine Anbindung zum Execution Supervisor erstellt werden. Somit kann auch ein bereits bestehendes System, welches auf dem hier präsentierten Architekturkonzept aufbaut, nachträglich auf einfache Weise um zusätzliche Funktionalitäten erweitert werden. Abgesehen von den schon vorhandenen Modulen, die aufgrund einer neuen Funktionalität angepasst werden müssen, ist dabei bezüglich der Architektur lediglich der Execution Supervisor zu aktualisieren. Um diese Aufgabe leicht erfüllen zu können, ist auch das Konzept des Execution Supervisors auf Erweiterbarkeit ausgelegt.

Im Folgenden wird zunächst das Konzept des Execution Supervisors zur Koordinierung des Systems beschrieben (Abschnitt 3.3.1). Damit der Execution Supervisor letztendlich in der Lage ist, die an ihn angebundenen Module entsprechend zu steuern, muss er mit diesen Komponenten kommunizieren können. Auf welche Weise dies in dem hier beschriebenen Architekturkonzept zu erfolgen hat, wird anschließend dargelegt (Abschnitt 3.3.2).

3.3.1. Zentrale Systemkontrolle

Wie bereits erwähnt, ist es die Aufgabe des Execution Supervisors, alle an ihn angebundenen Module zu koordinieren, um zu jedem Zeitpunkt ein einheitliches und zielorientiertes Verhalten des

resultierenden Gesamtsystems gewährleisten zu können. Die Funktionalität des Execution Supervisors ist dazu ähnlich zu der des so genannten *Sequencers* ist, welcher in ATLANTIS [Gat92] verwendet wird. Dieser *Sequencer* koordiniert, andersherum als sonst üblich, die Arbeitsweise der Module, die für deliberative Berechnungen verantwortlich sind. Dies ist gegensätzlich zu der Funktionsweise der meisten hybriden Architekturen, in denen der reaktive Planausführungsmechanismus lediglich sicherzustellen hat, dass ein Plan ausgeführt worden ist, bevor ein neuer Plan empfangen wird (siehe z.B. [Kul02]). In diesem Fall ist nämlich eine deliberative Komponente gewöhnlich nur für die kontinuierlichen Erzeugung von Plänen verantwortlich, ohne Feedback von reaktiven Systemen berücksichtigen zu müssen.

Das Design des Execution Supervisors ist außerdem durch die Architektur des Roboters ROBITA inspiriert [Kim02]. Dort sind alle Komponenten des Systems durch einen so genannten *„Priority Based Coordination Mechanism“* miteinander verbunden. Grundlage dieses Mechanismus' stellt ein zentrales *Blackboard* [Hay85] dar. Das *Blackboard* wird dabei von einem so genannten *„Situation Observation Server“* überwacht, welcher den einzelnen Systemkomponenten aufgrund der aktuellen Situation eine entsprechende Priorität zuweisen kann. Die Komponente mit der höchsten Priorität erhält dann die Kontrolle im System. Sollte diese Komponente aufgrund interner Bedingungen nicht ausgeführt werden können, so wird die Komponente mit der zweithöchsten Priorität aktiviert. Der Vorteil dieses Ansatzes ist, dass die Programmierer einzelner Komponenten nicht mit dem gesamten System vertraut sein müssen. Allein der Verwalter des *Situation Observation Servers* muss über die Fähigkeiten aller Komponenten informiert sein, um die entsprechenden Prioritäten definieren zu können, die in diesem System fest vergeben werden. Aus diesem Grund ist auch das hier beschriebene Konzept des Execution Supervisors, abgesehen von der Verwendung der Prioritäten, an den Entwurf des *Situation Observation Servers* angelehnt.

Um das Gesamtsystem kontinuierlich kontrollieren zu können, sollte der Execution Supervisor nur Berechnungen durchführen, die einen relativ geringen Zeitbedarf haben. Eine solche Dauer ist relativ zu der Geschwindigkeit zu betrachten, mit der Veränderungen durch den reaktiven Feedback-Kontrollmechanismus in der Umgebung wahrgenommen werden. Aus diesem Grund sind neben der Fähigkeit zur Systemkontrolle im Execution Supervisor keine zusätzlichen Kompetenzen zu integrieren. Folglich müssen sich letztere allesamt in anderen Modulen befinden, die mit dem Execution Supervisor verbundenen sind. Diese Strategie ist beispielsweise gegensätzlich zu der, welche der in [Kia04] vorgestellten Architektur zugrunde liegt, wo die zentrale Komponente unter anderem auch deliberative Aufgaben übernimmt [Amp04]. Aus demselben Grund ist im Execution Supervisor, anders als bei dem oben genannten Roboter ROBITA, auch kein internes *Blackboard* integriert. Für entsprechende Aufgaben existiert das Modul zur Speicherung interner Repräsentationen parallel zum Execution Supervisor. Durch diese Auslagerung nicht kontrollbezogener Fähigkeiten resultiert das in dieser Arbeit präsentierte Architekturkonzept folglich auch nicht in einer rein zentralisierten Architektur, wodurch ein darauf aufgebautes Gesamtsystem auch nicht den in [Ros95] genannten Schwierigkeiten unterliegen kann (vgl. Abschnitt 3.2.2). Außerdem wird durch die Beschränkung der Verantwortlichkeiten des Execution Supervisors erreicht, dass dieser in Form einer kompakten Komponente realisiert werden kann, welche einfacher erweiterbar und weniger fehleranfällig ist als eine komplexere Steuerungskom-

ponente. Somit werden auch die Anforderungen bezüglich der Kontrolle eines komplexen Robotersystems erfüllt (vgl. Abschnitt 3.2.2).

Die Aufgaben, die dem Execution Supervisors schließlich noch zur Ausführung verbleiben, sind die folgenden: Einerseits muss der Execution Supervisor die Funktionalitäten anderer Module konfigurieren, indem er sie entsprechend parametrisiert. Andererseits ist es seine Aufgabe, die Aktivität der anderen Module zu steuern, um die Rechenkapazitäten des Gesamtsystems zu schonen. Dazu müssen entsprechende Module während der Laufzeit aktiviert und deaktiviert werden können. Da auf einem mobilen Roboter in der Regel nur sehr wenige Rechner eingesetzt werden und das vorgestellte Architekturkonzept eine sehr überschaubare Anzahl an Modulen beinhaltet, sind allerdings keine großen Anstrengungen zur automatischen Lastverteilung notwendig. Das Laufzeitverhalten der einzelnen Komponenten kann empirisch ermittelt werden, um die individuellen Module auf die verschiedenen Rechner zu verteilen. Insgesamt werden durch das Konzept des Execution Supervisors somit auch gewisse Anforderungen bezüglich der Verteilung von Rechenlast berücksichtigt (vgl. Abschnitt 3.2.5).

3.3.2. XML-basierte Kommunikation

Die Kommunikation zwischen dem Execution Supervisor und allen mit ihm verbundenen Komponenten basiert auf der Verwendung von Nachrichten, die in XML kodiert sind. Die Beschreibungssprache XML eignet sich insbesondere als Basis für den Informationsaustausch in dem hier präsentierten Architekturkonzept, da sie sehr flexibel, einfach zu lernen und angemessen zur Beschreibung von abstrakten Konzepten ist. Wichtig ist hierbei, dass XML nicht nur als Protokoll für den Datentransfer verwendet wird, wie es in Lösungen der Fall ist, die auf XML-RPC basieren [Kia04]. Solche Ansätze produzieren gewöhnlich einen zusätzlichen Aufwand durch die verwendeten Regeln zur Parameterkodierung und durch die textbasierte Repräsentation von Binärdaten. Stattdessen ist es die Idee, dass für das zu implementierende Zielsystem eine XML-Sprache zur Repräsentation von symbolischen Daten, wie z.B. Zuständen, Ereignissen oder Objekten, entworfen wird. Instanziierte XML-Dokumente enthalten dabei dann direkt die zugehörigen semantischen Informationen, auf die mittels standardisierter XQuery/XPath-Mechanismen komfortabel zugegriffen werden kann.

Der deklarative Zugriff auf XML-Daten durch XPath-Ausdrücke hat diverse Vorteile. Einerseits wird durch ihn die Erstellung und Erweiterung von im Zielsystem benötigten Datentypen unterstützt, da ein deklarativer Zugriff nicht zu Instabilitäten im System führt, falls auf eine modifizierte Datenstruktur zugegriffen wird. Andererseits kann auf deklarative Weise eine lose Koppelung zwischen den verschiedenen Modulen realisiert werden. Somit können beispielsweise bestimmte Module im Gesamtsystem problemlos durch andere ersetzt werden. XML unterstützt daher die Ambition, dass alle Module in der entwickelten Architektur als unabhängige Agenten miteinander interagieren, wodurch zusätzlich die Übersichtlichkeit und die Flexibilität der Architektur gefördert wird. Folglich werden dadurch die Anforderungen bezüglich der Systemkommunikation erfüllt (vgl. Abschnitt 3.2.3).

Für XML stehen außerdem Mechanismen zur Verfügung, mit denen sich zu allen XML-Strukturen Meta-Informationen, wie z.B. über die darin enthaltenen Datentypen, spezifizieren lassen. Diese Informationen werden allerdings nicht mit in dem jeweiligen XML-Dokument integriert, sondern in einer separaten Datei, einer so genannten XML-Schemadatei, abgelegt. Die Spezifikation von Datentypen mittels XML-Schemata hat diverse Vorteile gegenüber traditionellen Programmiersprachenkonstrukten. Zunächst sind diese Datentypen unabhängig von bestimmten Programmiersprachen. Trotzdem sind entsprechende, zur Verarbeitung benötigte Werkzeuge für nahezu jede Plattform erhältlich. Außerdem können mit XML-Schemata Inhalte von Datenstrukturen und gültige Wertebereiche der darin enthaltenen Variablen sehr detailliert spezifiziert werden. Indem ferner Deklarationen von grundlegenden Datenstrukturen semantisch gruppiert und in separaten Schemadateien bereitgestellt werden, können wiederverwendbare Bibliotheken von Datentypen erstellt werden. Komplexe Schemata für individuelle Module können dann auf einfache Weise definieren werden, indem aus den grundlegenden Datentypen dieser Bibliotheken spezifische komplexe Datentypen zusammengesetzt werden. Auch für die Erweiterung von bestehenden Schemata sind entsprechende Mechanismen gegeben. Selbst komplexe Grammatiken können für Komponenten, die von verschiedenen Modulen des Systems stammende XML-Dokumente interpretieren und validieren müssen, auf einfache und überschaubare Weise durch eine sorgfältig konstruierte Schemahierarchie erstellt werden. Durch dessen Verwendung ist es schließlich möglich, alle Datenstrukturen, die zur Kommunikation in einem komplexen Gesamtsystem verwendet werden, zentral und auf strukturierte Weise zu verwalten, so dass widersprüchliche Definitionen verhindert werden. Diese Eigenschaft ist insbesondere dann entscheidend, wenn mehrere Entwickler bzw. Entwickler-Teams an der Implementierung des Systems beteiligt sein sollten.

Für den eigentlichen Datenfluss in dem Architekturkonzept ist schließlich folgende Struktur vorgesehen (siehe Abb. 3.1): Jedes Modul sendet Nachrichten in Form von Ereignissen, welche Konfigurationsdaten oder Informationen für andere Module enthalten, an den Execution Supervisor. Als Folge der Verarbeitung eines Ereignisses versendet der Execution Supervisor Kommandos und Statusmeldungen, welche die benötigten Parameter oder Informationen erhalten, die zusammen mit dem Ereignis empfangen wurden. Kommandos werden an Module in der intermediären und der reaktiven Ebene gesendet, um das System entsprechend zu konfigurieren. Statusmeldungen werden hingegen an Module in der deliberativen Ebene verschickt und informieren diese Komponenten über den internen Status des Gesamtsystems. Diese Form der Kommunikation spiegelt die hierarchische Struktur des hier vorgestellten Architekturkonzeptes wider: Kommandos werden nach „unten“ versendet, während Statusmeldungen nach „oben“ gerichtet sind.

Um des Weiteren bestimmten Sicherheitsanforderungen des Gesamtsystems gerecht zu werden, sollte jedes Modul für andere Module auf wahrnehmbare Weise versagen, da im realen Betrieb Ausfälle nicht ausgeschlossen werden können. Diese Anforderung wird durch die Verwendung spezieller Nachrichten erfüllt, welche durch die Hauptschleifen der einzelnen Module zu initiiert sind. Dabei werden diese Nachrichten in festen Intervallen an alle Module versendet, die mit dem jeweiligen Sender in Verbindung stehen. Falls nun ein Modul solche Nachrichten nicht mehr erhalten sollte, so kann es feststellen, dass der entsprechende Sender nicht mehr korrekt funktioniert. In diesem Fall können Aktionen zur Fehlerbehebung eingeleitet werden, um die

Funktionsfähigkeit des Gesamtsystems wiederherzustellen. Falls dies nicht möglich sein sollte, so wäre der Roboter zumindest in der Lage, den Benutzer um technische Unterstützung zu bitten. Durch diesen Mechanismus werden daher auch Anforderungen bezüglich der Fehlertoleranz des Zielsystems berücksichtigt (vgl. Abschnitt 3.2.4). Insgesamt sind somit durch das hier präsentierte Architekturkonzept alle strukturellen Anforderungen abgedeckt, welche in Abschnitt 3.2 angeführt wurden.

3.4. Zusammenfassung

Nachdem im vorigen Kapitel Paradigmen zur Ansteuerung von Robotern diskutiert wurden, sind in diesem Kapitel sowohl funktionale als auch strukturelle Anforderungen an einen *Robot Companion* gestellt worden. Während sich die funktionalen Aspekte auf die Fähigkeiten beziehen, welche der Roboter für seinen Einsatz im Rahmen des *Home Tour Scenarios* besitzen soll, dienen die strukturellen Aspekte hauptsächlich dem Entwicklungsprozess des Robotersystems. Hierbei wurden insbesondere die Punkte Modularität, Systemkontrolle, Kommunikation, Fehlertoleranz und Lastverteilung als Schwerpunkte für das Konzept der zu entwickelnden Roboterarchitektur identifiziert. Erst die Umsetzung dieser nicht-funktionalen Anforderungen ermöglicht es, die Entwicklung des *Robot Companions* insoweit zu strukturieren, dass ein effizientes Gesamtsystems geschaffen werden kann, welches außerdem einfach zu warten und leicht um neue Funktionalitäten zu erweitern ist.

Schließlich ist das Architekturkonzept vorgestellt worden, welches ausgehend von den zuvor beschriebenen Anforderungen im Rahmen dieser Arbeit entwickelt wurde. Es basiert auf einer Drei-Ebenen-Architektur, welche durch einen zentralen Execution Supervisor koordiniert wird. Die Kommunikation zwischen dem einzelnen Modulen der Architektur basiert auf der Verwendung von XML. Folglich ermöglicht das Konzept eine einfache Integration von Fähigkeiten, die zur anspruchsvollen Mensch-Roboter-Interaktion benötigt werden.

Bevor allerdings die Umsetzung dieses Architekturkonzeptes für den Roboter BIRON in Kapitel 7 vorgestellt wird, erfolgt in den nächsten Kapiteln zuerst die Vorstellung des Robotersystems. Diese gliedert sich einerseits in die Beschreibung der Hardware-Plattform des Roboters und andererseits in die Präsentation alle Software-Komponenten, die zur Realisierung der gewünschten Interaktionsfähigkeiten benötigt werden.

4. Eingesetztes Robotersystem: BIRON

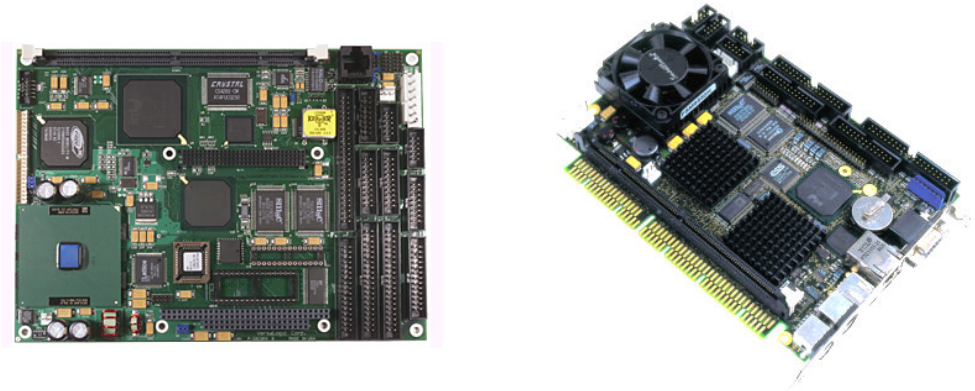
Bei der Hardware-Plattform des Roboters BIRON handelt es sich um einen *Pioneer PeopleBot* der Firma ActivMedia. Das Robotersystem wurde allerdings um zusätzliche Hardware-Komponenten erweitert, um den Ansprüchen an einen *Robot Companion*, welcher im *Home Tour Scenario* eingesetzt werden soll, zu genügen. Zum einen wurde ein Touch-Screen-Display zur Interaktion mit einem Benutzer auf dem Roboter installiert, zum anderen wurden mehrere Sensoren in den *PeopleBot* integriert. Das resultierende Robotersystem ist in Abbildung 4.1 dargestellt. Im Folgenden wird zunächst der *PeopleBot* in Abschnitt 4.1 und anschließend die zusätzlich verwendeten Sensoren in Abschnitt 4.2 beschrieben.

Abbildung 4.1.: Der Roboter BIRON.

4.1. Grundausstattung des Robotersystems

Das von ActivMedia vertriebene Robotersystem *Pioneer PeopleBot*, welches die Hardware-Plattform von BIRON bildet, ist bereits mit grundlegenden Komponenten ausgestattet, die dem Roboter Mobilität und Autonomie verleihen. Diese Komponenten umfassen sowohl einen Satz Hochleistungsakkus, zwei Antriebsmotoren und deren zugehörige Antriebsräder, Wegaufnehmer zur Berechnung von Position und Geschwindigkeit des Roboters als auch mehrere Sonar- und Anstoßsensoren zu Wahrnehmung von Hindernissen (siehe Abb. 4.1). Der Zugriff auf diese Komponenten erfolgt mittels eines Mikro-Controllers, der ebenfalls im Roboter integriert ist.

Der *PeopleBot* bietet zahlreiche Erweiterungsmöglichkeiten für zusätzliche Ausstattungen. Unter anderem existieren an dem internen Mikro-Controller diverse Anschlüsse. Darunter befindet sich ein systemspezifischer Bus und weitere digitale und analoge Anschlussmöglichkeiten. Der Mikro-Controller wird inklusive aller an ihm angeschlossenen Komponenten über eine serielle Schnittstelle angesprochen, welche mit einem im Roboter installierten Rechner verbunden ist. Bei dem Rechner handelt es sich um ein Intel® Pentium®-III-System mit 850 MHz Prozessortakt und 256 MB Hauptspeicher (siehe Abb. 4.2a). Dieses System ist in der Basis des *PeopleBots* integriert und bietet vier serielle Schnittstellen und einen Bus für bis zu fünf PC104plus-Karten. Hier ist eine Sound-Karte angeschlossen, die unter anderem zur Ansteuerung von zwei Lautsprechern im Turm des Roboters dient. Des Weiteren ist eine Funk-Ethernet-Karte installiert, um einen drahtlosen Zugriff auf das Robotersystem zu ermöglichen. Ferner ist ein Fast-Ethernet-Anschluss vorhanden.



(a) (b)

Abbildung 4.2.: Bei den integrierten Rechnern handelt es sich um Single-Board-Systeme: (a) Basis-PC (VersaLogic VSBC-8), (b) Turm-PC (Kontron *cool*MONSTER/P3).

Um ausreichend Rechenkapazitäten für die Verarbeitung der Daten der zusätzlich integrierten Sensoren (siehe Abschnitt 4.2) zu erhalten, wurde ein zusätzlicher Rechner im Turm des Roboters installiert (siehe Abb. 4.2b). Dabei handelt es sich um ein Intel® Pentium®-III-System mit 500 MHz Prozessortakt und 256 MB Hauptspeicher. Dieser Rechner besitzt unter anderem einen PCI-Bus um den Anschluss einer Steckkarte zur Aufnahme von Kamerabildern zu ermöglichen.

Außerdem verfügt er ebenfalls über einen Fast-Ethernet-Anschluss. Sowohl der hier beschriebene Turm-PC, als auch der zuvor genannte Basis-PC werden unter Linux betrieben und sind über das 100-Mbit-Ethernet miteinander verbunden. Ein zwischengeschalteter Switch bietet die Möglichkeit, weitere Rechner, wie z.B. einen Laptop, einzubinden.

Damit der Roboter auch über Artikulationsfähigkeiten verfügt, wurde er mit einem 12 Zoll großen Touch-Screen-Display ausgestattet. Da der Roboter aus Gewichtsgründen nicht mit Manipulatoren ausgerüstet werden konnte, ist das Display für die Interaktion mit einem Benutzer vorgesehen. Neben dem Erfassen von Benutzereingaben kann das Display zur Simulation von Zeigegesten dienen. Dazu blendet der Roboter ein bestimmtes Objekt aus seiner Umgebung, welches er mit seiner Kamera erfasst hat, auf dem Display ein, anstatt auf diesen Gegenstand zu zeigen. Außerdem kann ein Gesicht dargestellt werden, welches dem Roboter mehr Persönlichkeit verleihen kann und die Möglichkeit bietet, Systemzustände in Form von Emotionen zu visualisieren. Zusätzlich kann über die Lautsprecher im Turm des Roboters, unter Verwendung eines Moduls zur Sprachausgabe, der aktuelle Interaktionsstatus des Roboters gegenüber dem Interaktionspartner sprachlich artikuliert werden. Im Gegensatz zu grafischen Ausgaben auf dem Display hat dies den Vorteil, dass die Äußerungen des Roboters weitgehend unabhängig von der aktuellen Position des Benutzers wahrgenommen werden können.

4.2. Zusätzliche Sensorausstattung

Der Roboter BIRON wurde zusätzlich zu der im vorherigen Abschnitt beschriebenen Grundausstattung mit weiteren Sensoren zur Erfassung unterschiedlicher Modalitäten ausgestattet. Dabei handelt es sich um eine Pan-Tilt-Kamera, eine Stereokamera, ein Mikrofonpaar und ein Laser-Entfernungsmesser. Ein Überblick über die Anordnung dieser Sensoren auf dem Roboter ist in Abbildung 4.3 gegeben.

Bei der Pan-Tilt-Kamera handelt es sich um eine Sony EVI-D31 (siehe Abb. 4.4a). Sie wurde ganz oben auf dem Roboter in einer Höhe von ungefähr 142 cm montiert, um Bilder von der oberen Körperhälfte des Menschen, der mit dem Roboter interagiert, aufzunehmen. Die Kamera ist motorgesteuert, d.h. sie lässt sich um jeweils 100° nach links und rechts drehen und um jeweils 25° neigen und senken. Dadurch ist es möglich, sie auf den Interaktionspartner auszurichten, um ihn einerseits optimal verfolgen zu können und andererseits, um ihm ein Feedback bezüglich der Aufmerksamkeit des Roboters zu geben. Die Ansteuerung der Kamera geschieht über eine serielle Schnittstelle und wird von dem PC im Turm des Roboters übernommen. Dieser Rechner ist ebenfalls für die Aufnahme der zugehörigen Kamerabilder verantwortlich und ist dazu mit einer handelsüblichen TV-Karte für den Einsatz in PCs (siehe Abb. 4.4b) ausgestattet.

Da die Sony-Kamera mit einem Öffnungswinkel von etwa $48{,}8° \times 37{,}6°$ ein sehr eingeschränktes Sichtfeld hat, wurde zusätzlich eine Stereokamera (siehe Abb. 4.4c) auf dem Roboter auf einer Höhe von ungefähr 95 cm installiert. Diese Kamera bietet nicht nur einen größeren Öffnungswinkel von etwa $95{,}3° \times 78{,}9°$, sondern kann auch, ähnlich zum menschlichen Augenpaar, das

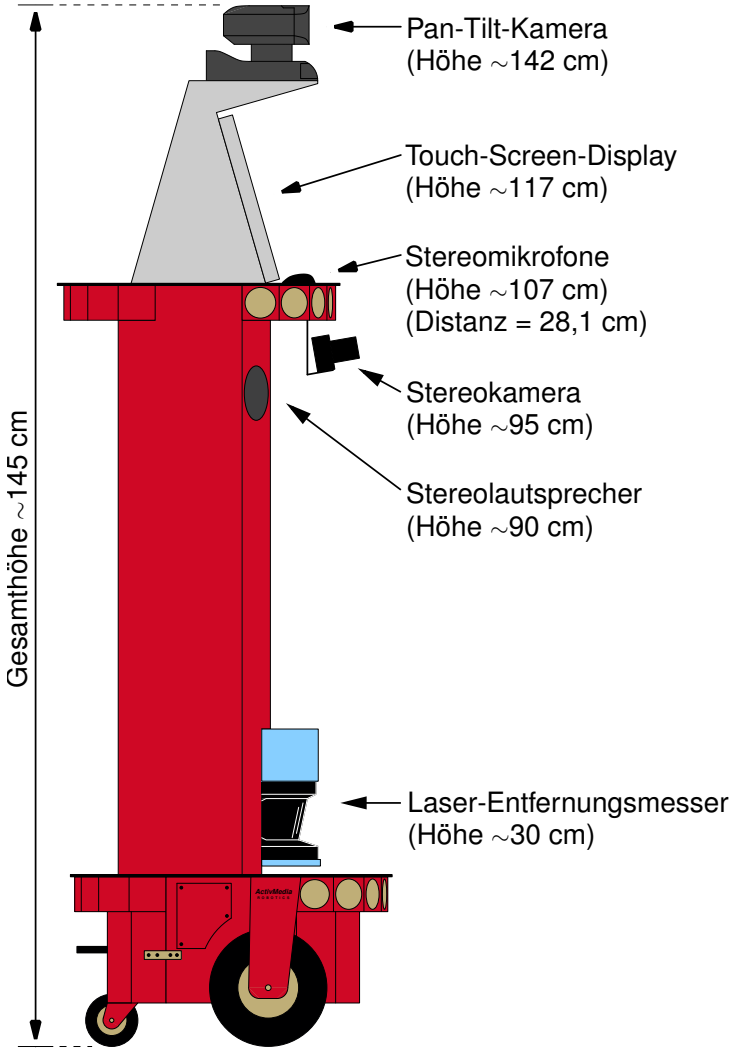


Abbildung 4.3.: Seitenansicht des Roboters BIRON.

Tiefenprofil der Umgebung erfassen. Im Gegensatz zu z.B. dem Roboter *BUGNOID* [Doi02] wird keine omni-direktionale Kamera eingesetzt, da dies nicht kognitiv adäquat ist. Das Ziel ist es, durch einen dem Menschen ähnlichen Wahrnehmungsbereich ein für den Benutzer nachvollziehbares Verhalten des Roboters realisieren zu können, so dass der Roboter möglichst intuitiv genutzt werden kann. Die Verarbeitung der Stereobilder findet auf einem separaten Laptop statt, der allerdings im Turm des Roboters integrierbar ist.

Bei den verwendeten Mikrofonen handelt es sich um zwei AKG-Grenzflächenmikrofone (siehe Abb. 4.5a), die beispielsweise in Freisprecheinrichtungen für Telefone eingesetzt werden, da ihr Frequenzbereich zur Aufnahme von Sprache optimiert ist. Die Mikrofone sind an der Vorderseite des Roboters, direkt unter dem Touch-Screen-Display, auf einer Höhe von ungefähr 107 cm angebracht. Der Abstand zwischen den Mikrofonen beträgt 28,1 cm. Da für die im Roboter integrierten Rechnersysteme keine systemkompatiblen Sound-Karten mit Stereomikrofoneingängen

(a) (b) (c)

Abbildung 4.4.: (a) Die Pan-Tilt-Kamera (Sony EVI-D31), (b) die TV-Karte (Pinnacle PCTV PRO), (c) die Stereokamera (Videre Design STH-MDCS).

erhältlich sind, wird das Signal über den Line-In-Eingang in die Soundkarte des PCs in der Roboterbasis gespeist. Daher ist die Verwendung eines Vorverstärkers (siehe Abb. 4.5b) nötig, der außerdem modifiziert wurde, um die von den Mikrofonen benötigte Phantomspannung zu liefern.

(a) (b)

Abbildung 4.5.: (a) Der verwendetet Mikrofontyp (AKG C 400 BL), (b) der Mikrofonvorverstärker (Vivanco MA222).

Der Laser-Entfernungsmesser der Firma SICK (siehe Abb. 4.6) ist an der Vorderseite des Roboters auf einer Höhe von etwa 30 cm montiert. Die verwendete Konfiguration erlaubt eine Auflösung im Millimeterbereich und liefert dabei Messwerte bis zu 32 m Entfernung. Außerdem wurde die feinste Winkelauflösung von 0,5° gewählt, so dass aus dem Messbereich von 180° pro Messung 361 Messwerte resultieren. Der Sensor ist über eine serielle Schnittstelle an den Basis-PC des Roboters angeschlossen, wodurch die Frequenz der Messungen auf etwa 4,7 Hz beschränkt ist.

Abbildung 4.6.: Der Laser-Entfernungsmesser (SICK LMS 200).

Insgesamt ergibt sich auf BIRON durch die Integration der vorgestellten Hardware-Komponenten die in Abbildung 4.7 dargestellte Verschaltung aller Rechner, Sensoren und Aktuatoren.

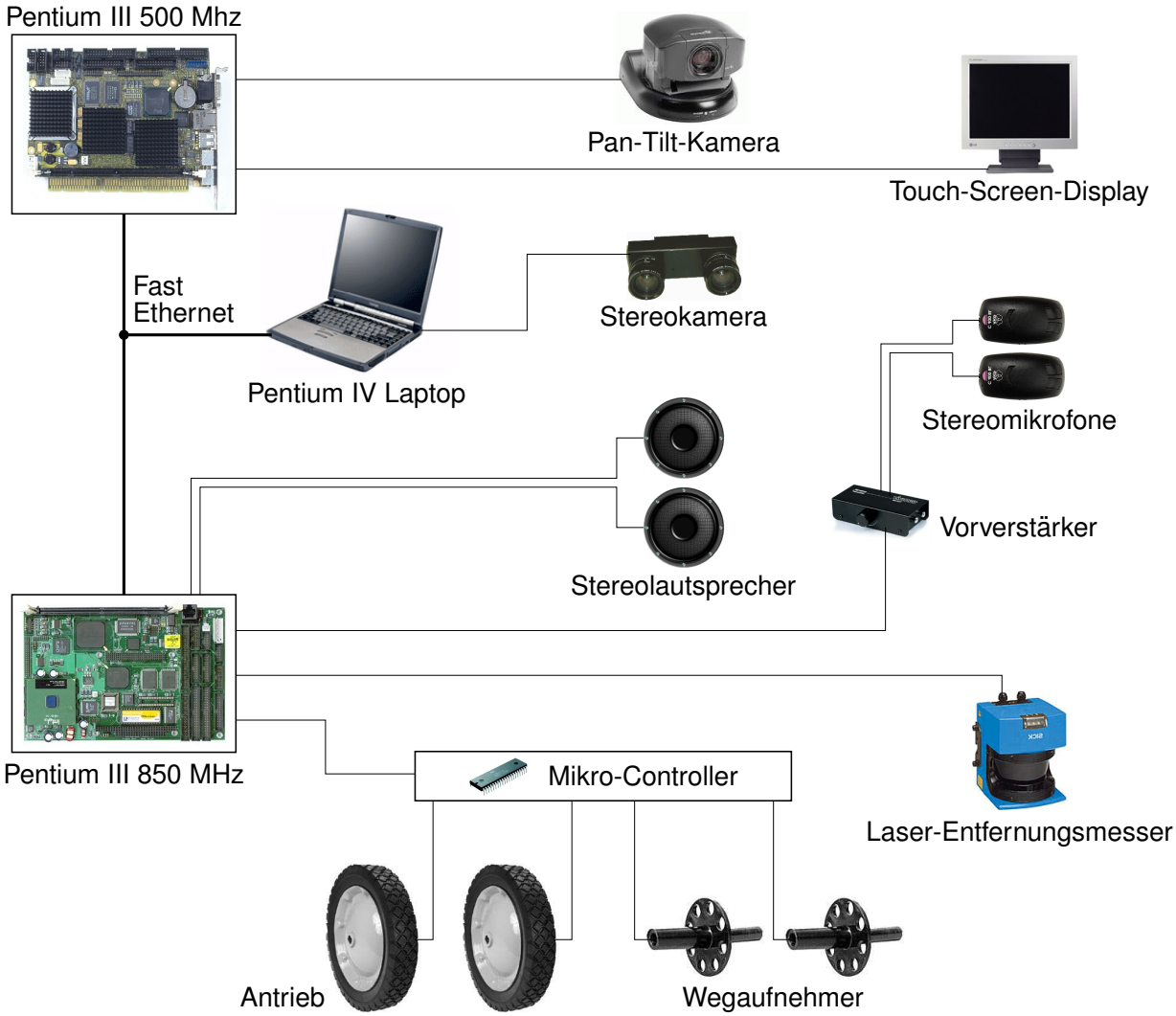


Abbildung 4.7.: Integrationsschema aller Hardware-Komponenten auf BIRON.

4.3. Zusammenfassung

In diesem Kapitel wurde der Roboter BIRON vorgestellt, welcher die Hardware-Grundlage zur Umsetzung der im Rahmen dieser Arbeit entworfenen Software-Architektur bildet. Bei dem Roboter handelt es sich um einen *Pioneer PeopleBot* der Firma ActivMedia, der mit diversen zusätzlichen Sensoren und Rechnern ausgestattet wurde, um den Anforderungen an einen *Robot Companion* gerecht zu werden. Im Folgenden werden die Software-Module vorgestellt, welche auf die im Roboter integrierten Hardware-Komponenten zugreifen.

5. Personen-Tracking als Basis für Interaktion

Die Erkennung von Personen stellt für einen Roboter eine grundlegende Voraussetzung dar, um mit ihnen interagieren zu können (vgl. z.B. [Has02, Wil02]). Da zu jedem Zeitpunkt mehrere Personen vor dem Roboter anwesend sein können, muss das System in der Lage sein, zwischen verschiedenen Personen zu unterscheiden. Ansonsten kann nicht gewährleistet werden, dass es bei einer Interaktion zu keinen Verwechslungen kommt. Somit ist es nicht nur wichtig, Personen zu erkennen, sondern sie auch über die Zeit verfolgen zu können.

Das Verfolgen von Personen, auch „Tracking“ genannt, ist von einem mobilen Roboter aus allerdings eine hochdynamische Aufgabe. In der Regel bewegen sich nicht nur die Personen vor dem Roboter, sondern auch der Roboter selbst kann sich bewegen. Diese Eigenschaft stellt eine Herausforderung für die Entwicklung eines robusten Trackings für Personen dar. Hinzu kommt noch, dass komplexe Objekte, wie etwa Menschen, in der Regel nicht nur durch einen einzigen Sensor, der sich auf dem mobilen Roboter befindet, erfassbar sind. Das Problem kann im Allgemeinen auch nicht gelöst werden, indem ein spezieller Sensor mehrfach verwendet wird. Werden beispielsweise statt nur einer Kamera mehrere Kameras verwendet, so kann es trotzdem vorkommen, dass der Roboter bei ungünstigen Lichtverhältnissen praktisch blind ist. Folglich bietet sich ein multimodales Tracking an. Daraus ergibt sich wiederum das Problem der Datenfusion: Unterschiedliche Daten von verschiedenen Sensoren müssen integriert werden, um Aufschluss über die Position einer Person vor dem Roboter zu bekommen. Da, wie bereits erwähnt, mehrere Personen anwesend sein können, geht mit der Datenfusion das so genannte *„Data Association Problem“* [Ros03] einher. Dabei ist zu klären, welches von einem Sensor wahrgenommene Muster zu welcher Instanz eines verfolgten Objektes gehört.

Um die angeführten Probleme bezüglich des Erkennens und des Trackings von Personen zu lösen, wurde die Arbeit von Coradeschi und Saffiotti [Cor00, Cor01, Cor03] erweitert (siehe Abschnitt 5.1 bzw. 5.2). Vor der Präsentation des im Rahmen dieser Arbeit realisierten Systems werden im Folgenden verwandte Techniken zur Assoziation und Fusion von Daten vorgestellt, da diese Ähnlichkeiten zum präsentierten Ansatz aufweisen.

Um eine Kategorisierung von Techniken zur Assoziation und Fusion von Daten vorzunehmen, definieren Bar-Shalom und Li in [Bar95], Kapitel 8.2, verschiedene Typen von Konfigurationen für Multisensor-Tracking. Die so genannte *Typ-I*-Konfiguration bezeichnet dabei ein gewöhnli-

ches Tracking-System, welches nur einen Sensor verwendet. Die *Typ-II*-Konfiguration führt ein *Typ-I*-Tracking für mehrere Sensoren durch und fusioniert anschließend die individuellen Datenströme. Im Gegensatz dazu führt die *Typ-III*-Konfiguration eine direkte, synchrone Sensordatenfusion über mehrere Sensoren hinweg durch, bevor das Tracking auf die fusionierten Daten angewendet wird. Die *Typ-IV*-Konfiguration führt hingegen nur eine lokale Datenassoziation für die einzelnen Sensoren durch, verwendet aber dennoch ein globales Tracking. Der Vorteil dieser Konfiguration liegt im Vergleich zur *Typ-III*-Konfiguration darin, dass keine synchronen Sensordaten benötigt werden.

Für das Tracking von Personen, welches sich unterschiedlicher Sensormodalitäten bedient, ist bereits eine große Vielfalt von Ansätzen und Methoden zur Datenfusion entwickelt worden. Darrell et al. [Dar00] verwenden eine *Typ-II*-Konfiguration zur Integration von Tiefeninformationen, Farbsegmentierungsresultaten und Gesichtserkennungsergebnissen. Die einzelnen Datenströme werden schließlich unter Verwendung von einfachen Regeln fusioniert, wobei ein statistisches Modell für die Zuordnung von Erkennungsergebnissen zu verschiedenen Personen sorgt. Gleichermaßen bedienen sich Okuno et al. [Oku01] einer *Typ-II*-Konfiguration, um auditive und visuelle Informationen von sprechenden Personen miteinander zu fusionieren. Die Fusion der Datenströme geschieht ebenfalls regelbasiert, aber im Gegensatz zu [Dar00] werden zusätzlich Schwellwerte verwendet, um Daten, die aus unterschiedlichen Richtungen aufgenommen werden, besser differenzieren zu können. Somit ist sichergestellt, dass nur Informationen miteinander fusioniert werden, die von derselben Person stammen.

Eine *Typ-III*-Konfiguration wird von Feyrer und Zell [Fey00] verwendet, um Personen auf Basis von visuellen Daten und Laser-Entfernungsdaten zu verfolgen. Dabei werden die zwei Arten von Sensorinformationen in einer Repräsentation basierend auf einem Potentialfeld fusioniert. In dieser Repräsentation wird jede Position, an der sich potentiell eine Person befindet, durch eine zweidimensionale Gauß-Funktion beschrieben. Nach der Auswahl der Person, die es zu verfolgen gilt, wird eine weitere Gauß-Verteilung in dem Potentialfeld hinzugefügt. Diese Funktion wird auf die durch einen Kalman-Filter bestimmte Schätzung der Personenposition angewendet, um eine zeitliche Kohärenz zu gewährleisten.

Im Gegensatz zu den oben präsentierten Verfahren, die eine parallele Fusion von Sensordaten vornehmen, welche von unterschiedlichen Sensortypen stammen, führen *Typ-IV*-Konfigurationen eine sequenzielle Verarbeitung der individuellen Sensorinformationen aus. Der Vorteil einer solche Konfiguration besteht darin, dass eine hierarchische Verarbeitung entsprechender Daten möglich ist. In der Regel wird dazu nach der Ausführung grober Positionsbestimmungen ein kleinerer Suchraum verwendet, um präzisere Sensordaten verarbeiten zu können. Beispielsweise schlagen Schlegel et al. [Sch98] ein visuelles Tracking für Personen vor, welches im ersten Schritt Farbinformationen verwendet, um den Suchbereich in einem Kamerabild einzuschränken. Im zweiten Schritt wird der bestimmte Bereich schließlich nach der Kontur eines Menschen abgesucht. Wilhelm et al. [Wil02] verwenden die Merkmale Hautfarbe und Kopf-Schulter-Silhouette, um Personen visuell zu detektieren. Diese Merkmale werden durch einen *Fuzzy-Min-Max-Operator* fusioniert, welcher zusätzlich durch Sonarmessungen gewichtet wird. Obwohl die Sonarmessungen nicht sehr genau sind, können sie schnell verarbeitet werden, um freie Bereiche

vor dem Roboter ausfindig zu machen, welche folglich nicht visuell untersucht werden müssen. Somit sorgen die Sonarmessungen für eine schnellere und robustere Lokalisation von Personen. Eine verfeinerte Methode, um eine sequentielle Reduktion des Suchraumes zu realisieren, wird von Vermaak et al. [Ver01] empfohlen. In diesem Ansatz werden auditive und visuelle Informationen durch Techniken, welche auf *Particle Filtering* basieren, fusioniert. Ein verwandter probabilistischer Ansatz, der grafische Modelle zur Kombination von auditiven und visuellen Daten verwendet, wird von Beal et al. [Bea03] vorgestellt.

Alle oben präsentierten Verfahren zum Verfolgen von Personen verwenden entweder einfache Regeln oder probabilistische Zuordnungen, die sowohl vorgegeben als auch gelernt sein können, um verschiedene Arten von Daten zu fusionieren. Das im Folgenden vorgestellte multimodale System, welches in Kooperation mit S. Lang [Lan05] entwickelt wurde, beruht auf einem strukturierten Ansatz, um gleichzeitig mehrere Personen, basierend auf ihren individuellen „Bestandteilen“, zu verfolgen. Ein Vorteil dieser Vorgehensweise ist, dass einige der erfassten Sensordaten zusätzlich auch für speziellere Aufgaben eingesetzt werden können. Beispielsweise kann anhand von Kamerabildern festgestellt werden, ob ein menschlicher Instrukteur gerade mit dem Roboter interagiert oder sich einer anderen Person in der Nähe zuwendet (siehe Abschnitt 6.1).

In Szenarios, in denen sich ein oder mehrere Menschen in der Nähe eines Roboters befinden können, wird das Tracking von Personen oft durch die Verwendung der Daten eines Laser-Entfernungsmessers erreicht. Wird der Sensor auf niedriger Höhe eingesetzt, so enthalten die aufgenommenen Daten charakteristische Muster, welche von den Beinen der umgebenen Personen stammen (siehe z.B. [Fey00, Fri03]). Obwohl diese Informationen unter anderem hilfreich für die Bestimmung einer angemessenen Geschwindigkeit beim Folgen einer bestimmten Person sind, können Beine oftmals durch am Boden befindliche Hindernisse, wie z.B. durch einen Papierkorb, verdeckt und somit für den Roboter nicht sichtbar sein. Allerdings stellen neben den Beinen auch das Gesicht, der Oberkörper und die Stimme markante Hinweise für die Anwesenheit einer Person dar, die für das Tracking genutzt werden können. Gesichter und die Bekleidung von Oberkörpern können mit Hilfe von Bilddaten detektiert werden, welche von einer Videokamera geliefert werden. Auf analoge Weise erlaubt ein Paar an Mikrofonen Geräuschquellen zu lokalisieren, wie z.B. die Stimme einer sprechenden Person. Diese beiden Informationsquellen repräsentieren zusätzliche Schätzungen für die Positionen von Personen, während sie zugleich zur Steuerung der Aufmerksamkeit des Roboters eingesetzt werden können (siehe Abschnitt 6.1).

Für das in dieser Arbeit beschriebene Verfahren zur Erkennung und Verfolgung von Personen wird eine Kamera, ein Mikrofonpaar und ein Laser-Entfernungsmesser verwendet (siehe Abschnitt 5.3.1). Alle diese Sensoren befinden sich zwar auf dem Roboter, aber aufgrund der unterschiedlichen Verarbeitungsgeschwindigkeiten der entsprechenden Daten muss die Fusion asynchron geschehen. Somit muss das Tracking von Personen in einer *Typ-IV*-Konfiguration durchgeführt werden. Im Gegensatz zu den in [Bar95] vorgeschlagenen Methoden zur Datenfusion wurde ein modellbasiertes, modulares Integrationsschema entwickelt (siehe Abschnitt 5.2). Dieses so genannte „multimodale Anchoring“ erweitert das von Coradeschi und Saffiotti präsentierte „Anchoring“-Schema (siehe folgenden Abschnitt). Multimodales Anchoring beherrscht nicht nur klassisches Tracking mit unterschiedlichen Sensoren, die sich an verschiedenen Positionen

befinden, sondern ist auch in der Lage, Repräsentationen für temporär nicht erfassbare Objekte zu pflegen. Des Weiteren stellt es Mechanismen bereit, um diese Objekte wiederzufinden. Daher kann multimodales Anchoring als eine Erweiterung zu klassischen Tracking-Ansätzen verstanden werden, welches ein Schema vorgibt, um mit ausbleibenden Sensordaten in strukturierter Weise umzugehen. Es ist außerdem einfach zu implementieren, besitzt eine transparente Struktur und weist eine geringe Komplexität auf, so dass die Fusion von Sensordaten auf effiziente Weise möglich ist.

5.1. Anchoring zur Fusion von Sensordaten

Das Problem, Objekte zu erkennen, indem aus Sensordaten extrahierte Merkmale mit einer internen symbolischen Repräsentation verknüpft werden, ist besonders prominent in autonomen Systemen, deren Umgebung sich ständig ändert. Solch ein System muss Verknüpfungen zwischen Prozessen, die mit abstrakten Repräsentationen von Objekten in der Umwelt arbeiten (symbolische Ebene) und Prozessen, die für die physische Wahrnehmung dieser Objekte verantwortlich sind (sensorische Ebene), herstellen. Diese Verknüpfungen müssen dynamisch sein, da ein bestimmtes Symbol jedes Mal an neue Sensordaten geknüpft werden muss, wenn neue Beobachtungen von dem entsprechenden Objekt gewonnen werden. Um diese Aufgabe in strukturierter Weise zu lösen, wurde das so genannte „Anchoring“ entwickelt. Anchoring weist dabei auf konzeptueller Ebene gewisse Parallelen zur Mustererkennung auf [Blo01].

Wir folgen der Definition von Anchoring, die von Coradeschi und Saffiotti in [Cor01] vorgestellt wird. Dort wird Anchoring als das Problem der Erzeugung und der zeitlichen Aufrechterhaltung einer Zuordnung zwischen Symbolen und Sensordaten, die sich auf dasselbe physische Objekt beziehen, beschrieben. Im Wesentlichen beinhaltet Anchoring ein „Symbolsystem“ und ein „perzeptuelles System“. Diese beiden Systeme werden durch einen so genannten „Anchor“ miteinander verbunden (siehe Abb. 5.1 am Beispiel des Anchorings von Gesichtern). Das Symbolsystem enthält eine Menge von individuellen „Symbolen“ und einen Satz von unären „Prädikatsymbolen“. Jedes individuelle Symbol besitzt dabei eine „symbolische Beschreibung“, die aus einem Satz von Prädikatsymbolen besteht. Das perzeptuelle System enthält hingegen eine Menge von „Perzepten“ und einen Satz von „Attributen“. Ein Perzept ist dabei eine strukturierte Kollektion von Messwerten, von denen angenommen wird, dass sie von dem gleichen physischen Objekt stammen. Ein Attribut stellt dabei eine messbare Eigenschaft eines Perzeptes dar. Die Menge der Attribut-Messwert-Paare eines Perzeptes wird als „perzeptuelle Signatur“ bezeichnet.

Die Aufgabe des Anchorings ist es nun, eine Zuordnung zwischen einem Symbol, welches benötigt wird um ein Objekt im Symbolsystem zu bezeichnen, und einem Perzept, welches im perzeptuellen System durch dasselbe Objekt hervorgerufen wurde, herzustellen. Dies wird dadurch erreicht, dass die symbolische Beschreibung mit der perzeptuellen Signatur über eine so genannte *„Predicate Grounding Relation“*[1], bezeichnet als g, verglichen wird. Diese Relation

[1] in Anlehnung an *Symbol Grounding*

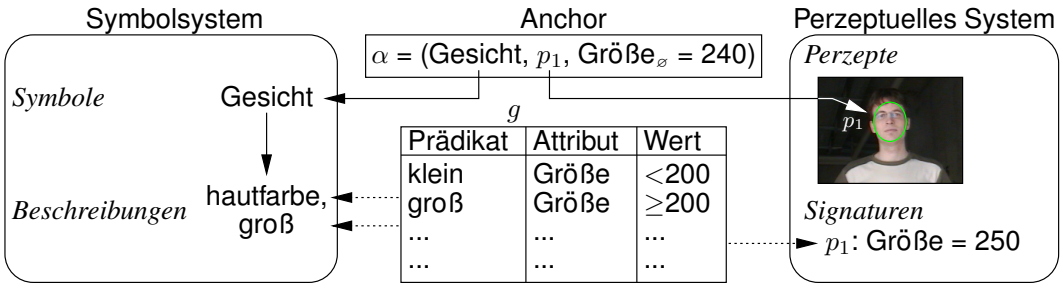


Abbildung 5.1.: Verknüpfung eines Symbols mit Sensordaten durch einen Anchor.

stellt eine Zuordnung von unären Prädikaten zu ermittelten Messwerten von messbaren Eigenschaften eines Objektes dar. Beispielsweise könnte g spezifizieren, dass ein Symbol mit dem Prädikat „groß“ einem bestimmten Perzept entspricht, falls der Messwert des Attributs „Größe“ über einem bestimmten Schwellwert liegt. Die Relation g kann entweder vom Entwickler vorgegeben oder vom System gelernt werden. Mit ihr kann entschieden werden, ob eine gegebene perzeptuelle Signatur mit einer gegebenen symbolischen Beschreibung konsistent ist oder nicht. Die Zuordnung von Symbol zu Perzept wird in einer internen Datenstruktur α repräsentiert, die als „Anchor“ bezeichnet wird. Da neue Perzepte fortlaufend von dem zugehörigen perzeptuellen System erzeugt werden, ist diese Relation durch die Zeit indiziert.

Zu jedem Zeitpunkt t enthält der Anchor $\alpha(t)$ drei Elemente: ein Symbol, welches ein Objekt innerhalb des Symbolsystems bezeichnet; ein Perzept, welches innerhalb des perzeptuellen Systems erzeugt wird, indem ein Objekt wahrgenommen wird; und eine Signatur, welche eine Schätzung für die Messwerte liefert, die den beobachtbaren Eigenschaften des Objektes entsprechen. Der Anchor α wird als *„grounded“* zum Zeitpunkt t bezeichnet, falls er ein Perzept, welches zum Zeitpunkt t wahrgenommen wurde, und die zugehörige aktuelle Signatur enthält. Falls das Objekt zum Zeitpunkt t nicht wahrgenommen werden konnte, so wird der Anchor als *„ungrounded“* bezeichnet. In diesem Fall wird kein Perzept in dem Anchor gespeichert, die Signatur stellt aber immer noch die bestmögliche Schätzung für die erwartete perzeptuelle Signatur dar.

Um das Anchoring-Problem für ein gegebenes Symbol x in einer dynamischen Umgebung zu lösen, werden folgende drei Hauptfunktionalitäten in [Cor00, Cor01, Cor03] eingeführt:

- **Find:** Diese Funktion ist dafür verantwortlich, einen Anchor initial in den Zustand *grounded* zu versetzen, nachdem das durch x bezeichnete Objekt zum ersten Mal wahrgenommen wurde. Dabei wird sichergestellt, dass die symbolische Beschreibung des Symbolsystems mit der perzeptuellen Signatur des perzeptuellen Systems vereinbar ist. Für den Fall, dass sich mehrere Perzepte als passend erweisen, muss entweder in der *Find*-Funktion selbst oder im Symbolsystem eine Auswahl vorgenommen werden.

- **Track:** Diese Funktion aktualisiert kontinuierlich einen Anchor, wenn das zugehörige Objekt in aufeinander folgenden Verarbeitungsschritten wahrgenommen wird. In diesem Fall

werden die Werte für die Attribute in der Signatur des Anchors geschätzt. Die auf diese Weise geschätzte Signatur wird dann mit den wahrgenommenen Attributwerten verglichen. Ziel des Vergleichs ist es, ein Perzept zu finden, das bezüglich seiner Attribute mit dem Perzept kompatibel ist, welches im vorherigen Schritt mit dem entsprechenden Symbol durch den Anchor verknüpft wurde. Falls es mehrere passende Perzepte geben sollte, muss ähnlich wie im *Find*-Fall, ein Perzept ausgewählt werden.

- **Reacquire:** Durch diese Funktion wird ein Anchor aktualisiert, falls das entsprechende Objekt zeitweilig nicht beobachtet werden konnte, d.h. der Anchor *ungrounded* ist. Diese Funktionalität sorgt für eine erneute Lokalisation eines Objektes, wenn zuvor bereits entsprechende perzeptuelle Informationen erworben wurden. Diese Informationen werden verwendet, um eine neue Signatur für den Anchor zu schätzen, welche dann mit einem neu akquirierten Perzept verglichen wird. In diesem Fall ist die Schätzung generell komplexer als in dem *Track*-Fall. Wenn ein Perzept sowohl kompatibel mit der geschätzten Signatur als auch mit der symbolischen Beschreibung ist, so wird die Signatur des Anchors aktualisiert. Falls wiederum mehrere passende Perzepte vorliegen sollten, muss auch hier ein Perzept für die Aktualisierung ausgewählt werden.

Die Relation zwischen diesen drei Hauptfunktionalitäten ist in Abbildung 5.2 visualisiert. Ein Anchoring-Prozess startet im *Find* und wechselt ins *Track*, sobald das entsprechende Objekt erfasst wurde. Falls es während des Trackings zeitweilig nicht erfasst werden kann, so wechselt der Anchoring-Prozess für diese Zeit in das *Reacquire*, bis wieder ein entsprechendes Perzept erfolgreich zugeordnet werden kann. Um zu berücksichtigen, dass ein zu beobachtendes Objekt dauerhaft aus dem Wahrnehmungsbereich des Systems entfernt werden könnte, sollte ein Anchoring-Vorgang abbrechbar sein. Zu diesem Zweck wird der zugehörige Anchor aufgelöst, falls der Anchoring-Prozess das *Reacquire* nach einer bestimmten Zeit noch nicht verlassen haben sollte.

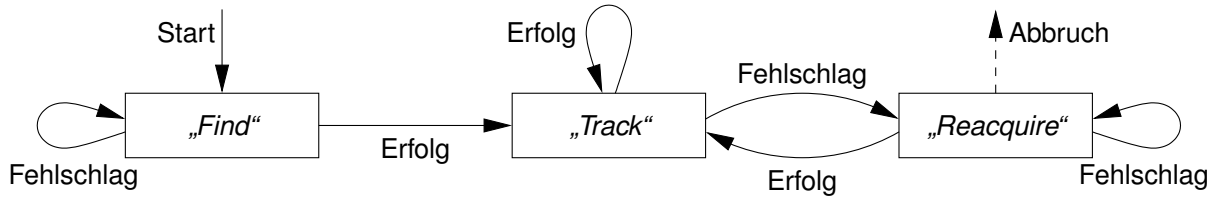


Abbildung 5.2.: Die drei Hauptfunktionalitäten des Anchorings und deren Relation zueinander.

Ein zum Anchoring verwandtes Verfahren verwendet so genannte *„Marker“* um das absolute Sichtfeld eines Systems zu einem effektiven Sichtfeld zu erweitern [Was97b]. Diese *Marker* entsprechen den in diesem Kapitel vorgestellten Anchors und bilden das Gedächtnis für die reaktive Komponente des in [Was97b] präsentierten Systems. Das System ist ebenfalls in der Lage Objekte zu lokalisieren und sie zu verfolgen. Ein *Marker* dient dabei zum Speichern von Objekten, die entweder den physischen Sichtbereich des Systems verlassen haben oder verdeckt sind. Die Verwendung von *Markern* wird auch in Bezug auf die Gesamtarchitektur eines Robotersystems

betrachtet [Was97a], wo sie zwischen den Ebenen einer hybriden Architektur ausgetauscht werden können (vgl. Abschnitt 2.2). Dabei kann der Planer, je nach Aufgabe, *Marker* verarbeiten oder erzeugen. In [Was99] werden diese Konstrukte auch dazu verwendet, um mehrere Personen zu detektieren, die der Roboter dann abwechselnd betrachten kann.

5.2. Multimodales Anchoring

Das im vorigen Abschnitt vorgestellte Anchoring wurde bisher in der Literatur nur für den speziellen Fall betrachtet, in dem genau ein Symbol mit den Perzepten von einem Sensor verknüpft wird. Die reale Welt enthält jedoch Objekte, die nicht von einem Sensor allein komplett erfasst werden können. Daher bietet es sich an, nicht nur mehrere, sondern auch unterschiedliche Sensoren zu verwenden. Infolgedessen muss die symbolische Beschreibung eines zu beobachtenden Objektes an mehrere verschiedene Arten von Perzepten geknüpft werden können, die aus unterschiedlichen Modalitäten akquiriert werden.

Eine Möglichkeit, mehrere Sensoren zu berücksichtigen, stellt die Erweiterung der Anchoring-Definition dar, so dass mehrere Arten von Perzepten an ein einzelnes Symbol gebunden werden können. Da verschiedene Perzeptarten üblicherweise unterschiedlichen Verarbeitungszeiten unterliegen, erfordert dieser Ansatz, dass die individuellen Perzepte auf asynchrone Weise im Anchoring-Prozess verarbeitet werden. Hinzu kommt, dass die verschiedenen Sensoren in der Regel dazu eingesetzt werden, um Informationen über verschiedene Eigenschaften des zu beobachtenden Objektes zu erfassen. Beispielsweise kann ein Mensch verfolgt werden, indem sein Gesicht, seine Stimme und seine Beine wahrgenommen werden. In diesem Fall müssen gegebenenfalls die räumlichen Relationen zwischen den Objektbestandteilen innerhalb der *Predicate Grounding Relation* berücksichtigt werden, um konsistente Ergebnisse zu erhalten. Ist darüber hinaus das Gesamtobjekt in sich nicht statisch, so muss zusätzlich die Dynamik in der Konfiguration der Objektbestandteile berücksichtigt werden. Folglich würde der resultierende Algorithmus für diese Lösung aus Sicht der Implementierung sehr komplex ausfallen.

Aus den oben angeführten Gründen ist schließlich im Rahmen dieser Arbeit in Kooperation mit S. Lang [Lan05] ein modularer Ansatz entwickelt worden: das so genannte „multimodale Anchoring“ (siehe Abb. 5.3). Dieser Ansatz erlaubt es, ein Symbol eines Gesamtobjektes mit verschiedenen Perzeptarten zu verknüpfen, indem die zugehörigen Objektbestandteile mit entsprechenden Perzepten, die aus unterschiedlichen Modalitäten stammen, dezentralisiert verknüpft werden. Bei erfolgreicher Verknüpfung von Symbol und Perzept auf Ebene dieser monomodalen Anchoring-Prozesse werden die Attributwerte der entsprechenden perzeptuellen Systeme auch an einen speziellen multimodalen Anchoring-Prozess zwecks zentraler Integration weitergeleitet. Dort werden die individuellen Daten schließlich miteinander kombiniert und wiederum in einer Anchor-Struktur abgelegt, dem so genannten „multimodalen Anchor“. Multimodales Anchoring stellt somit eine strukturierte Methode zur Integration mehrerer monomodaler Anchoring-Prozesse dar und ermöglicht dabei eine parallele Verarbeitung der verschiedenen Perzeptarten.

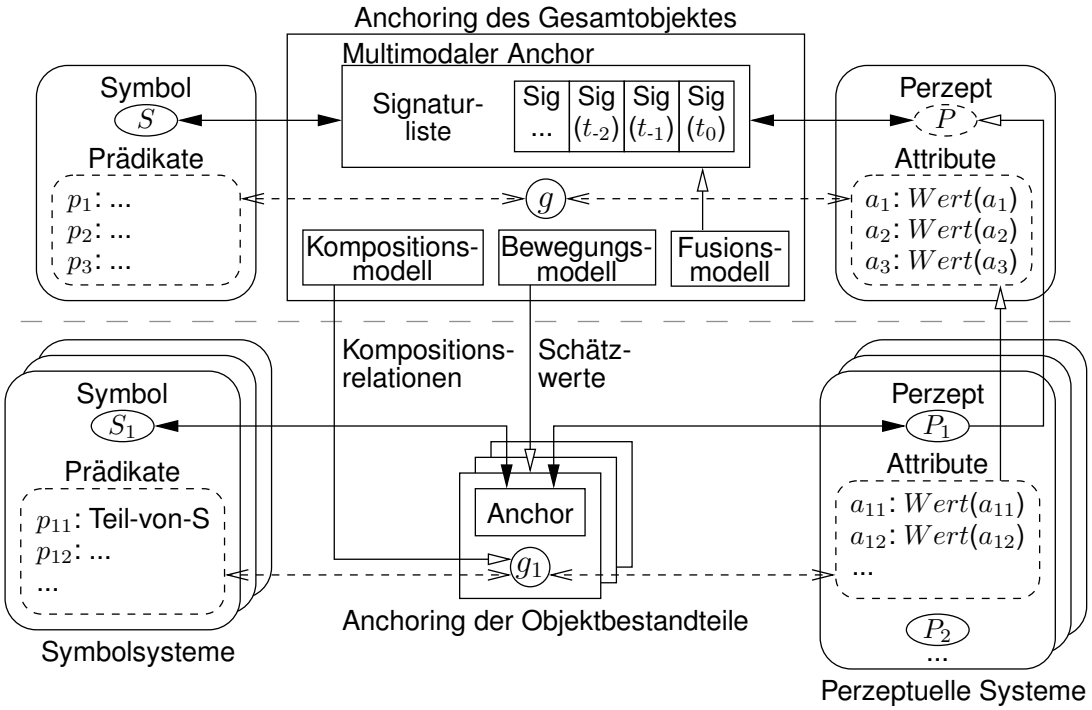


Abbildung 5.3.: Multimodales Anchoring.

Der Hauptunterschied zum ursprünglichen Anchoring nach Coradeschi und Saffiotti besteht auf Ebene des multimodalen Anchoring-Prozesses darin, dass das Symbol, welches dem Gesamtobjekt entspricht, kein direktes perzeptuelles Gegenstück besitzt. Jedes Mal, wenn einer der monomodalen Anchoring-Prozesse ein neues Perzept zur Aktualisierung seines Anchors ausgewählt hat, wird dieses Perzept ebenfalls an das Symbol des Gesamtobjektes geknüpft (siehe Abb. 5.3). Der multimodale Anchoring-Prozess berechnet dann seine eigene perzeptuelle Signatur auf Basis der vorliegenden Attributwerte des monomodalen Anchoring-Prozesses. Gewöhnlich kann die perzeptuelle Signatur eines monomodalen Anchoring-Prozesses allerdings nur dazu verwendet werden, eine Teilmenge der in dem multimodalen Anchoring-Prozess verwendeten Attribute zu aktualisieren. Das liegt daran, dass die zugeordneten Perzepte aus den perzeptuellen Systemen der Objektbestandteile stammen und sich nicht auf das Gesamtobjekt beziehen.

Des Weiteren existieren für den multimodalen Anchoring-Prozess keine individuellen Instanzen der drei Hauptfunktionalitäten *Find*, *Track* und *Reacquire*, wie es im ursprünglichen Anchoring der Fall ist. Diese Funktionalitäten werden von den monomodalen Anchoring-Prozessen ausgeführt, die auch die Aktualisierung des multimodalen Anchors initiieren. Das Gesamtobjekt befindet sich im Zustand *grounded*, falls sich wenigstens ein zugehöriger Objektbestandteil im gleichen Zustand befindet. Analog ist das Gesamtobjekt im Zustand *ungrounded*, falls sich alle zugehörigen Objektbestandteile in diesem Zustand befinden. Weil jeder monomodale Anchoring-Prozess individuelle Prädikatsymbole besitzt, enthält er auch seine eigene *Predicate Grounding Relation*. Die *Predicate Grounding Relation* des multimodalen Anchoring-Prozesses beschreibt

hingegen die Zuordnung von den Prädikaten, die das Gesamtobjekt betreffen, zu den Attributwerten, die aus den perzeptuellen Signaturen der verschiedenen monomodalen Anchoring-Prozesse berechnet werden.

Schließlich müssen alle monomodalen Anchoring-Prozesse koordiniert werden, um sicherzustellen, dass die verschiedenen Sensoren auch alle dasselbe Gesamtobjekt wahrnehmen. Daher müssen einerseits die monomodalen Anchoring-Prozesse mit Schätzungen für die Position des Gesamtobjektes versorgt werden. Andererseits muss der multimodale Anchoring-Prozess die Informationen, die von den monomodalen Anchoring-Prozessen geliefert werden, miteinander fusionieren. Aus diesen Gründen wird im multimodalen Anchoring ein „Kompositionsmodell“, ein „Bewegungsmodell“ und ein „Fusionsmodell“ zur Verfügung gestellt. Das erste Modell berücksichtigt räumliche Relationen zwischen den einzelnen Objektbestandteilen, das zweite Modell beschreibt die Dynamik des Gesamtobjektes, und das dritte Modell dient schließlich zur Kombination der Daten, die von den verschiedenen Objektbestandteilen stammen.

5.2.1. Das Kompositionsmodell

Das Kompositionsmodell enthält die räumlichen Beziehungen zwischen dem Gesamtobjekt und seinen Objektbestandteilen. Es stellt sicher, dass die monomodalen Anchoring-Prozesse nur die Perzepte auswählen, die zu dem Gesamtobjekt kompatibel sind. Zu Beginn des multimodalen Anchoring-Vorgangs baut ein monomodaler Anchoring-Prozess einen Anchor mit dem Zustand *grounded* auf, wenn seine symbolische Beschreibung mit der perzeptuellen Signatur übereinstimmt. Allerdings können als erstes nur die monomodalen Anchors aufgebaut werden, die dazu keine Zusatzinformationen von dem multimodalen Anchor benötigen. Beispielsweise muss ein monomodaler Anchoring-Prozess zuerst eine bestimmte Eigenschaft über das zugehörige Gesamtobjekt detektieren, bevor eine anderer monomodaler Anchoring-Prozess starten darf. Durch den ersten monomodalen Anchor wird auch der multimodale Anchor initialisiert und stellt von nun an Daten über das Gesamtobjekt allen zugehörigen monomodalen Anchoring-Prozessen zur Verfügung. Neben der Berücksichtigung dieser Daten müssen die einzelnen monomodalen Anchoring-Prozesse ebenfalls sicherstellen, dass den Relationen zwischen den Objektbestandteilen, die durch das Kompositionsmodell vorgegeben werden, entsprochen wird. Dazu enthält die symbolische Beschreibung eines jeden monomodalen Anchors das Prädikat „Teil-Von-S“, wobei *S* das Symbol des entsprechenden Gesamtobjektes darstellt. Nachdem ein monomodaler Anchoring-Prozess seinen Anchor aktualisiert hat, kann der multimodale Anchoring-Prozess schließlich überprüfen, ob auch die symbolische Beschreibung des Gesamtobjektes zu der entsprechenden perzeptuellen Signatur des gerade verarbeiteten Perzeptes passt.

5.2.2. Das Bewegungsmodell

Das Bewegungsmodell beschreibt das Bewegungsverhalten des Gesamtobjektes und die Bewegungsdynamik seiner einzelnen Objektbestandteile zueinander. Zusammen mit den räumlichen

Relationen, die durch das Kompositionsmodell bereitgestellt werden, erlaubt dieses Modell eine Schätzung der Positionen der einzelnen Objektbestandteile. Diese Informationen sind insbesondere für steuerbare Sensoren wertvoll, welche es erlauben, das benötigte Sichtfeld auszuwählen. In diesem Fall kann ein entsprechender Sensor in die Richtung gesteuert werden, aus der ein Perzept zu erwarten ist, um einen zugehörigen monomodalen Anchor aufbauen zu können.

5.2.3. Das Fusionsmodell

Das Fusionsmodell wird benötigt, um die verschiedenen perzeptuellen Signaturen der monomodalen Anchoring-Prozesse in dem multimodalen Anchor zu integrieren. Jedes Mal, wenn ein monomodaler Anchoring-Prozess ein neues Perzept verarbeitet hat, sendet er die zugehörige perzeptuelle Signatur an den multimodalen Anchoring-Prozess. Diese Signatur bezieht sich auf den Zeitpunkt, an dem die entsprechenden Sensordaten akquiriert wurden. Da die verschiedenen perzeptuellen Systeme unterschiedlichen Verarbeitungsgeschwindigkeiten unterliegen, erhält der multimodale Anchoring-Prozess nicht immer die Attributdaten der verschiedenen monomodalen Anchoring-Prozesse in der korrekten zeitlichen Reihenfolge. Um sicherzustellen, dass die Attributdaten zum korrekten Zeitpunkt mit der Signatur des multimodalen Anchors fusioniert werden, verwaltet der multimodale Anchoring-Prozess eine Liste aller empfangenen Signaturen, die zeitlich sortiert ist. Neue Attributdaten werden in die Liste eingefügt und die Signatur des multimodalen Anchors wird unter Verwendung des Fusionsmodells für den entsprechenden Zeitpunkt aktualisiert. Falls die Liste schon Einträge enthalten sollte, die aktueller sind als ein neu einzufügender Eintrag, so wird die Fusion der zugehörigen Signaturen des multimodalen Anchors für alle nachfolgenden Zeitpunkte, zu denen eine Signatur bereits verarbeitet wurde, wiederholt. Dies ist notwendig, um den Einfluss der nachträglich eingefügten perzeptuellen Signatur in den zeitlich darauf folgenden Fusionsschritten korrekt zu berücksichtigen. Die zu Grunde liegende Spezifizierung der Fusion selbst ist allerdings domänenabhängig.

5.2.4. Erweiterung auf mehrere komplexe Objekte

Gewöhnlich muss nicht nur ein Objekt, sondern mehrere Objekte gleichzeitig verfolgt werden. Dazu müssen mehrere multimodale Anchoring-Prozesse parallel betrieben werden, um die verschiedenen Objekte beobachten zu können. In diesem Fall kann es beim multimodalen Anchoring, wie es bisher beschrieben wurde, zu folgenden Konflikten zwischen den individuellen multimodalen Anchoring-Prozessen kommen:

- Ein Perzept wird von mehr als einem multimodalen Anchoring-Prozess durch die zugehörigen monomodalen Anchoring-Prozesse ausgewählt.
- Die multimodalen Anchoring-Prozesse versuchen einen steuerbaren Sensor zur gleichen Zeit auf widersprüchliche Weise anzusteuern.

Um diese zwei Probleme zu lösen, wird eine zusätzliche Komponente verwendet, welche alle multimodalen Anchoring-Prozesse verwaltet. Sie koordiniert die Auswahl von Perzepten und regelt den Zugriff auf steuerbare Sensoren und die Roboterbasis. Diese Komponente erlaubt den Zugriff auf die Hardware nur den multimodalen Anchoring-Prozessen, die einen so genannten *„Anchor Of Interest“* verwalten. Die Entscheidung, welcher Anchor ein *Anchor Of Interest* ist, hängt von der beabsichtigten Anwendung ab. Beispielsweise könnte dies der multimodale Anchor sein, der die meisten monomodalen Anchors besitzt, welche sich im Zustand *grounded* befinden, oder diese Entscheidung wird von einer höheren Instanz getroffen. Dabei kann es zu jedem Zeitpunkt entweder nur einen *Anchor Of Interest* geben, dessen multimodaler Anchoring-Prozess exklusiven Zugriff auf die gesamte Hardware bekommt, oder es gibt mehrere *Anchors Of Interest*. Im letzteren Fall können die beteiligten Anchoring-Prozesse allerdings nur Zugriff auf unterschiedliche Hardware-Komponenten erhalten.

Um das Auswählen der Perzepte zu koordinieren, muss die Weise, in der die individuellen monomodalen Anchoring-Prozesse ihre Auswahl treffen, modifiziert werden. Diese so genannten *Select*-Funktionalitäten wählen nun die Perzepte nicht mehr selbstständig aus. Stattdessen bewerten sie jedes Perzept, indem sie entsprechende Punktzahlen vergeben. Ein Perzept wird dabei um so höher bewertet, je besser es zum jeweiligen Anchor passt. Basierend auf dieser Bewertung kann eine übergreifende Auswahl durchgeführt werden (siehe Abb. 5.4). Jedes mögliche Ergebnis dieser Auswahl kann durch eine Liste von Zuweisungen beschrieben werden, wobei der n-te Eintrag der Liste die Nummer des Perzeptes enthält, welches für den n-ten Anchor ausgewählt wurde. Die Einträge in der Liste müssen natürlich paarweise verschieden sein, um ein konsistentes Ergebnis darzustellen. Die Gesamtpunktzahl der übergreifenden Auswahl wird als die Summe der Punkte aller entsprechenden Zuweisungen definiert. Das Ziel ist es nun, das optimale Ergebnis zu finden, welches die Auswahl darstellt, die die Gesamtpunktzahl maximiert. Die entsprechende Suche ist durch die Verwendung eines Suchbaumes realisiert: Die Wurzel des Baumes beschreibt die leere Liste von Zuweisungen, deren Einträge alle undefiniert sind. Für jeden nachfolgenden Knoten verringert sich die Anzahl der undefinierten Einträge um eins. Die Blätter des Baumes enthalten somit alle möglichen Ergebnisse einer übergreifenden Auswahl. Da das Maximum aller Punktzahlen, die von den einzelnen monomodalen Anchoring-Prozessen bestimmt werden, bekannt ist, kann die maximal mögliche Gesamtpunktzahl einer erst teilweise definierten Liste berechnet werden. Somit kann die Suche in dem Baum effizient unter Verwendung des A*-Algorithmus’ [Nil82] realisiert werden.

Wie in Abbildung 5.4 zu sehen ist, muss die Anzahl der wahrgenommenen Perzepte nicht notwendigerweise mit der Anzahl der verwalteten multimodalen Anchoring-Prozesse übereinstimmen. Falls es bezüglich eines perzeptuellen Systems mehr multimodale Anchoring-Prozesse als Perzepte gibt, dann wird nicht jedem zugehörigen monomodalen Anchor ein Perzept zugewiesen und in dem Fall auch der multimodale Anchor nicht aktualisiert. Falls es mehr Perzepte als multimodale Anchors geben sollte, so wird nicht jedes Perzept einem entsprechenden monomodalen Anchor zugewiesen. Verbleibende Perzepte werden aber dazu verwendet, neue monomodale und multimodale Anchors zu erzeugen. Umgekehrt werden monomodale Anchors, die eine bestimmte Zeit lang nicht mehr aktualisiert wurden, aus einem multimodalen Anchoring-Prozess entfernt. Falls ein multimodaler Anchoring-Prozess keine zugehörigen monomodalen

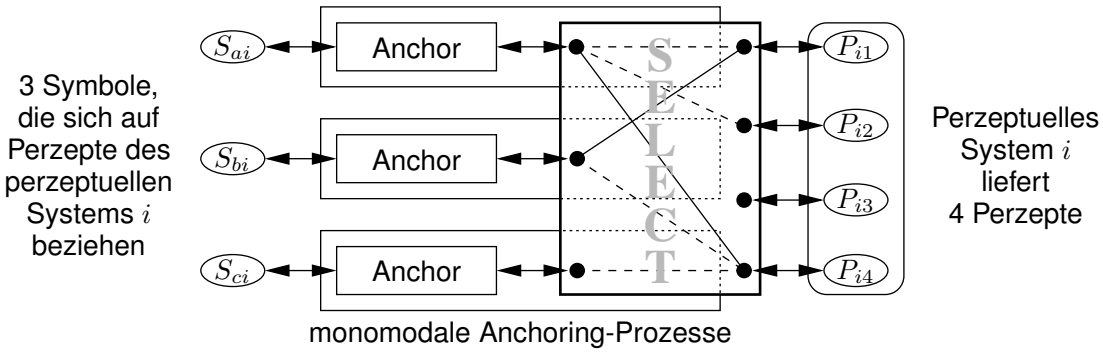


Abbildung 5.4.: Die Modifikation der *Select*-Funktionalitäten in den monomodalen Anchoring-Prozessen am Beispiel für 3 Symbole und 4 Perzepte.

Anchors mehr besitzen sollte, so wird auch dieser von der Komponente, die alle multimodalen Anchoring-Prozesse verwaltet, aufgelöst.

5.3. Multimodales Anchoring für Personen

In diesem Abschnitt wird beschrieben, wie das zuvor präsentierte multimodale Anchoring für das Tracking von Personen eingesetzt wird. Dazu werden zuerst die perzeptuellen Systeme dargelegt, welche die perzeptuellen Daten für die individuellen monomodalen Anchoring-Prozesse liefern. Danach werden die Besonderheiten der monomodalen Anchoring-Prozesse bezüglich der Personenbestandteile beschrieben. Schließlich wird die Verwaltung einer ganzen Person mit Hilfe des multimodalen Anchorings präsentiert und die Erweiterung auf mehrere Personen dargestellt.

5.3.1. Perzeptuelle Systeme

Im Folgenden werden die perzeptuellen Systeme für das Tracking von Personen vorgestellt. Es handelt sich dabei um eine Gesichtserkennung, eine Sprecherlokalisation, eine Oberkörperdetektion und eine Beindetektion.

Gesichtserkennung

Für Mensch-Roboter-Schnittstellen ist es nahezu unerlässlich, dass das Gesicht des Benutzers vom System beobachtet wird. Es bietet Informationen über die Identität des Benutzers, seinen emotionalen Zustand und seine Absichten. Allerdings muss jedes visuelle System, welches Gesichter verarbeitet, zuerst die Orte in den Kamerabildern detektieren, an denen sich Gesichter befinden. Jedoch stellt die Detektion von Gesichtern aufgrund von Variationen in Größe und Position innerhalb eines Bildes eine Herausforderung dar. Zusätzlich muss die Detektion robust ge-

genüber unterschiedlichen Beleuchtungssituationen und verschiedenen Gesichtsausdrücken sein. Bisher ist eine Vielzahl an Verfahren entwickelt worden, beispielsweise unter Verwendung von neuronalen Netzen [Row98], so genannten *„Deformable Templates“* [Yui92], Hautfarbensegmentierungen [Yan96] oder Hauptkomponentenanalysen. Letztere Verfahren stellen die so genannten *„Eigenface“*-Methoden dar [Tur91]. Ein sehr guter Überblick über entsprechende Ansätze findet sich in [Hje01, Yan02].

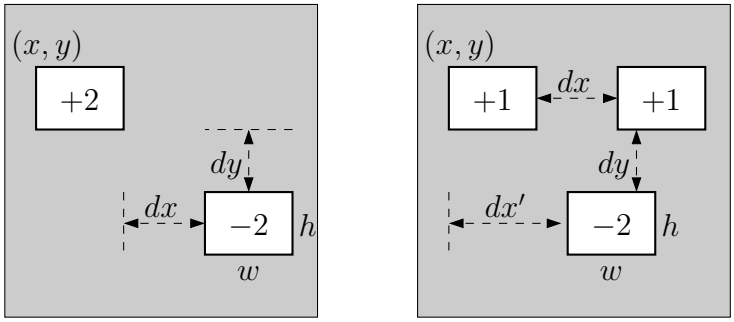
Es existieren auch Ansätze, um nicht nur das Gesicht, sondern den ganzen Kopf eines Menschen von einem Roboter aus zu erkennen. Dieses ist insbesondere hilfreich, wenn sich eine Person nicht ständig mit ihrem Gesicht dem Roboter zuwendet. Beispielsweise verwenden Ghidary et al. [Ghi00a] dazu mehrere Merkmale: Farbe, Form, Bewegung und Temperatur eines Menschen. Die Richtung, in die der Kopf orientiert ist, kann allerdings nicht bestimmt werden. Die Kopfdetektion stellt hier ebenfalls die Grundlage dafür dar, dass der Roboter seine Aufmerksamkeit auf einen Menschen fokussieren kann [Ghi00b] (vgl. Abschnitt 6.1). Das von Ghidary et al. vorgestellte Verfahren liefert allerdings nicht den Abstand zu einer Person. Dieser muss durch ein separates Verfahren bestimmt werden, damit der Roboter einen angemessenen Abstand zum Interaktionspartner einnehmen kann.

In einer ersten Implementierung einer Gesichtsdetektion für den Roboter BIRON [Kle02, Fri03] wurde eine Methode realisiert, die eine adaptive Hautfarbensegmentierung mit einem auf *Eigenfaces* basierenden Verfahren kombiniert [Fri02]. Die Segmentierung vermindert dabei die Größe des Suchraumes, so dass nur die Teile eines Kamerabildes mit der *Eigenface*-Methode untersucht werden müssen, die hautfarbene Bereiche enthalten. Um unterschiedliche Beleuchtungsbedingungen zu berücksichtigen, wird das Gesichtsfarbenmodell kontinuierlich mit den Bildpunkten, die aus den erkannten Gesichtern extrahiert werden, aktualisiert. Dieser zyklische Prozess benötigt allerdings eine Initialisierung, da zu Beginn kein passendes Modell für Hautfarbe vorhanden ist. Dazu wird die *Eigenface*-Detektion auf das gesamte Kamerabild angewendet. Diese Methode hat zwei schwerwiegende Nachteile: Einerseits reagiert sie sehr empfindlich auf falsch positiv detektierte Gesichter, da sich dadurch das Hautfarbenmodell an eine falsche Farbe anpasst. Andererseits ist die Initialisierung sehr rechenaufwändig.

In der aktuellen Implementierung, die im Folgenden beschrieben wird, beruht die Detektion von Gesichtern in frontaler Ansicht auf einer Methode die von Viola und Jones [Vio01] vorgeschlagen wurde. Diese Methode erlaubt es, Bilder sehr schnell und mit einer hohen Erkennungsrate nach Gesichtern zu durchsuchen. Daher wird weder eine zeitraubende Initialisierung benötigt, noch ist eine Beschränkung des Suchraumes durch ein Hautfarbenmodell notwendig.

Die Detektion beruht auf zwei Merkmalstypen, welche auch in [Zha02] verwendet werden (siehe Abb. 5.5). Ein Merkmal ist ein skalarer Wert, welcher dadurch berechnet wird, dass die gewichtete Summe aller Bildpunktintensitäten in rechteckigen Bildbereichen ermittelt wird. Die Berechnung kann sehr effizient durch die Verwendung von so genannten Integralbildern vorgenommen werden [Vio01]. Die Merkmale haben sechs Freiheitsgrade (x, y, w, h, dx, dy), wenn es sich um Zwei-Block-Merkmale (siehe Abb. 5.5a) handelt, und sieben Freiheitsgrade $(x, y, w, h, dx, dy, dx')$, wenn es sich um Drei-Block-Merkmale (siehe Abb. 5.5b) handelt. Durch Beschränkung der Rechteckgrößen und der Abstände zueinander ergeben sich ungefähr 300.000

verschiedene Merkmale für Teilbilder der Größe 20×20 Bildpunkte. Klassifikatoren werden dadurch konstruiert, dass eine kleine Anzahl ausschlaggebender Merkmale unter Verwendung von *„AdaBoost“* [Fre97] ausgewählt wird. Eine Kaskade von Klassifikatoren $(C_1, \ldots, C_n)$ mit steigender Anzahl von Merkmalen stellt den gesamten Gesichtsdetektor dar (siehe Abb. 5.6). Zur Gesichtsdetektion wird ein Kamerabild durchlaufen, indem jedes Teilbild mit dem ersten Klassifikator C_1 der Kaskade klassifiziert wird. Wenn das Teilbild nicht als Gesicht klassifiziert wird, so fährt der Prozess mit dem nächsten Teilbild fort. Ansonsten wird das aktuelle Teilbild an den nächsten Klassifikator (C_2) weitergereicht, usw.



(a) (b)

Abbildung 5.5.: Zwei Merkmalskonfigurationen zur Gesichtsdetektion. (a) Zwei-Block-Merkmal, (b) Drei-Block-Merkmal. Jedes Merkmal nimmt einen Wert an, der die gewichtete Summe aller Bildpunkte in dem Rechteck darstellt.

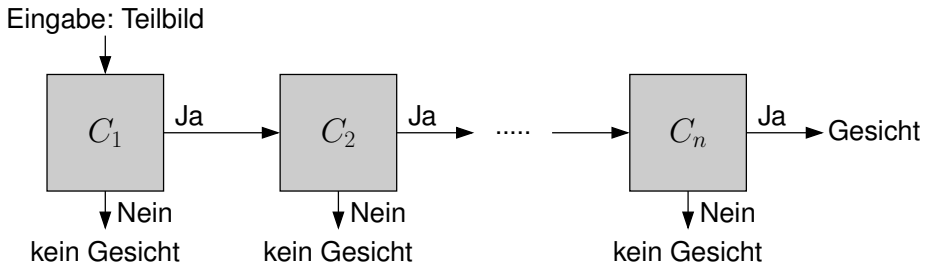


Abbildung 5.6.: Eine Kaskade von n Klassifikatoren mit wachsender Komplexität zur schnellen Gesichtsdetektion.

Die Kaskade wurde mit mehreren tausend Bildern automatisch trainiert, besteht aus 16 Klassifikatoren und verwendet insgesamt 1327 Merkmale. Der erste Klassifikator der Kaskade beruht nur auf zwei Merkmalen, weist aber bereits ungefähr 75 % aller Teilbilder zurück. Nur wenn die gesamte Kaskade erfolgreich durchlaufen wird, ist ein Gesicht gefunden. Somit ist der Detektionsprozess sehr schnell.

Um den multimodalen Anchoring-Prozess aktualisieren zu können, wird die Position eines detektierten Gesichts ermittelt: Unter Berücksichtigung der Ausrichtung der Pan-Tilt-Kamera kann der Winkel des Gesichts relativ zum Roboter bestimmt werden. Die Größe des detektierten Gesichts wird dazu verwendet, den Abstand zur Person zu schätzen. Unter der Annahme, dass die

Größe von erwachsenen Menschen sich nicht zu stark unterscheiden, ist der Abstand reziprok proportional zu der Größe des Gesichts. Die Höhe des Gesichts relativ zum Boden wird ebenfalls ermittelt, indem die Distanz und die Kameraposition einbezogen werden.

Da der präsentierte Ansatz zur Detektion von Gesichtern keine Identifikation von Personen erlaubt, wird eine zusätzliche Berechnung benötigt. Dazu wird eine erweiterte Variante der *Eigenface*-Methode [Tur91] verwendet. Jedes Individuum wird in einem Gesichtsraum durch eine Mischung von mehreren Gauß-Funktionen mit diagonaler Kovarianz repräsentiert. Praktische Experimente haben gezeigt, dass vier bis sechs Gauß-Verteilungen eine hinreichende Genauigkeit ergeben, um eine kleine Menge von bekannten Personen ($\sim$10) unterscheiden zu können.

Sprecherlokalisation

Eine Sprecherlokalisation erlaubt es einem mobilen Roboter nicht nur Personen zu detektieren, sie gibt auch Auskunft über die Interaktionsabsichten einer Person (vgl. Abschnitt 6.1). Für die Lokalisation werden Mikrofone verwendet, die auf dem Roboter montiert sind. In aktuellen Publikationen zur Geräusch- und Sprecherlokalisation werden weitgehend Mikrofonanordnungen mit mindestens drei Mikrofonen verwendet. Lediglich einige wenige Ansätze begnügen sich nur mit einem Paar an Mikrofonen. Schnelle und robuste Techniken zur Geräusch- und Sprecherlokalisation sind z.B. die *„Generalized Cross-Correlation Method“* [Kna76] und die *„Cross-Powerspectrum Phase Analysis“* [Giu94], welche beide sowohl für Mikrofonfelder als auch für nur ein Mikrofonpaar eingesetzt werden können. Komplexere Algorithmen zur Sprecherlokalisation, wie z.B. *„Spectral Separation and Measurement Fusion“* [Ber02] und *„Linear-Correction Least-Squares“* [Hua01], sind ebenfalls sehr robust und können zusätzlich die Distanz zum Sprecher und dessen Größe schätzen. Des Weiteren können mehrere Geräuschquellen unterschieden werden. Diese Algorithmen benötigen allerdings mehr als nur ein Paar an Mikrofonen, um angemessen zu funktionieren und verbrauchen auch beträchtlich mehr Rechenzeit als die zwei zuerst genannten Verfahren.

Für autonome Roboter mit beschränkten Rechenressourcen stellt die Verwendung von nur einem Mikrofonpaar einen guten Kompromiss dar, da andere Modalitäten einer zu beobachtenden Person vom Roboter aus wahrgenommen werden können. Aus diesem Grund wurde eine Sprecherlokalisation auf Basis der *Cross-Powerspectrum Phase Analysis* für ein Mikrofonpaar realisiert [Hoh03, Fin04]. Eine ausführliche Evaluation dieses Verfahrens [Lan03] hat gezeigt, dass die Verwendung von nur zwei Mikrofonen ausreichend für eine robuste Sprecherlokalisation im Rahmen des multimodalen Anchoring-Systems ist. Ein ähnlicher Ansatz wurde für den Roboter SIG verwendet. Allerdings sind dort zusätzlich zwei Mikrofone im Inneren des Roboters installiert worden, um Störgeräusche des Roboters besser herausfiltern zu können [Nak02].

Ziel der akustischen Lokalisation ist es, die Position einer Schallquelle relativ zum Roboter zu berechnen. Für eine Schallquelle s im dreidimensionalen Raum gilt, dass sich die Entfernungen d_1 und d_2 zwischen s und den beiden Mikrofonen m_1 und m_2 um den Abstand $\Delta d := d_2 - d_1$ unterscheiden (siehe Abb. 5.7). Dieser Abstand Δd resultiert unter Berücksichtigung der Schall-

geschwindigkeit c_s in einer zeitlichen Verzögerung $\delta = \Delta d / c_s$ beim Signalempfang zwischen rechtem und linkem Mikrofon. Unter Verwendung der *Cross-Powerspectrum Phase Analysis* wird zuerst ein Maß für die spektrale Korrelation berechnet:

$$K(\tau) = FT^{-1}\left(\frac{\hat{S}_L(f)\,\hat{S}_R^*(f)}{|\hat{S}_L(f)|\,|\hat{S}_R(f)|}\right) \tag{5.1}$$

Hierbei stellen $\hat{S}_L(f)$ und $\hat{S}_R(f)$ die Kurzzeitspektren des linken bzw. rechten Kanals dar. Ist nur eine einzige Schallquelle vorhanden, so ergibt sich die Laufzeitdifferenz δ durch das Argument τ, welches das Maß für die spektrale Korrelation $K(\tau)$ maximiert:

$$\delta = \arg\max_{\tau}\ K(\tau) \tag{5.2}$$

Werden in Gleichung (5.1) zusätzlich lokale Maxima berücksichtigt, so können verschiedene Schallquellen gleichzeitig lokalisiert werden.

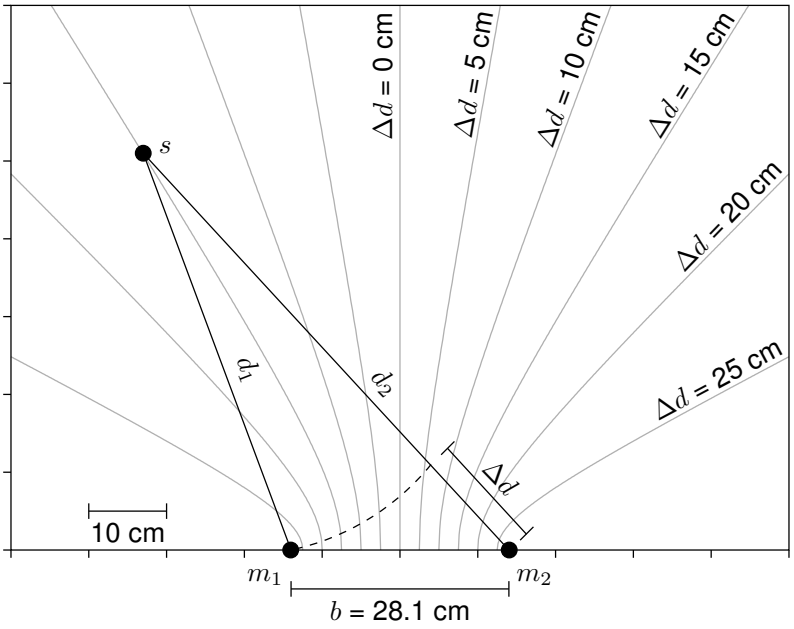


Abbildung 5.7.: Die Abstände d_1 und d_2 zwischen der Schallquelle s und den zwei Mikrofonen m_1 und m_2 unterscheiden sich um Δd. Alle Geräuschquellen mit gleichem Abstand Δd liegen auf einer Hälfte eines zweischaligen Hyperboloids (grau).

Allerdings kann selbst im zweidimensionalen Fall, in dem sich alle Geräuschquellen in der gleichen Ebene wie die Mikrofone befinden, die Position von s nur bestimmt werden, wenn die Distanz zu den Mikrofonen bekannt ist oder zusätzliche Annahmen aufgestellt werden. Daher wird in einer vereinfachten Geometrie angenommen, dass der Abstand der Mikrofone b im Vergleich zu den Distanzen d_1 und d_2 vernachlässigbar gering ausfällt. Somit kann der Einfallswinkel des Signals an den beiden Mikrofonen als nahezu gleich angenommen werden und direkt

unter Verwendung von Δd berechnet werden. Im dreidimensionalen Fall hängt die wahrgenommene Zeitdifferenz allerdings nicht nur von der Richtung und der Entfernung der Schallquelle ab, sondern auch von deren Höhe relativ zu den Mikrofonen. Folglich ist das das Problem unterbestimmt, wenn nur der Abstand Δd gegeben ist. Im Einzelnen liegen alle Geräuschquellen, die in dem gleichen Δd resultieren, auf einer Hälfte eines zweischaligen Hyperboloids, welcher wie folgt beschrieben ist:

$$\frac{s_x^2 + s_z^2}{\frac{1}{4}\left(b^2 - (\Delta d)^2\right)} - \frac{s_y^2}{\frac{1}{4}(\Delta d)^2} = -1 \tag{5.3}$$

Hier stellt (s_x, s_y, s_z) die Position der Schallquelle in kartesischen Koordinaten da. Die Symmetrieachse des Hyperboloids fällt mit der Achse, auf der sich die Mikrofone befinden, zusammen. Abbildung 5.7 zeigt den Schnitt zwischen der Ebene, die durch s, m_1 und m_2 aufgespannt wird, und einigen Hyperboloiden mit jeweils unterschiedlichem Δd. Folglich benötigt die Lokalisation von Schallquellen im dreidimensionalen Raum unter Verwendung von nur zwei Mikrofonen zusätzliche Informationen.

Da im gegebenen Szenario die zu betrachtenden Schallquellen sprechende Personen darstellen, können die zusätzlich benötigten räumlichen Informationen von den übrigen perzeptuellen Systemen des multimodalen Anchoring-Systems bezogen werden. Beine, Oberkörper und Gesicht liefern Informationen über die Richtung und die Entfernung einer Person relativ zum Roboter, letzteres Körperteil auch über ihre Größe. Falls allerdings kein Gesicht detektiert werden kann, so kann die Größe der Person zumindest mit einer Durchschnittsgröße für erwachsene Menschen geschätzt werden.

Um zu entscheiden, ob ein Sprachperzept einer bereits detektierten Person zugeordnet werden kann, muss die zugehörige Schallquelle im dreidimensionalen Raum bestimmt werden. Dazu wird angenommen, dass der Mund der Person denselben Abstand zum Roboter wie die Person selbst und dieselbe Höhe relativ zum Boden wie ihr Gesicht hat. Nun kann die entsprechende Richtung der Schallquelle mit Hilfe der Gleichung (5.3) in Zylinderkoordinaten berechnet werden. Wenn diese Richtung zu der Richtung passt, in der sich die Person befindet, so kann das Sprachperzept dem entsprechenden Anchor der Person zugeordnet werden. Dazu muss die Differenz zwischen den beiden Richtungen entsprechend dem Kompositionsmodell klein genug sein. Schließlich wird die Richtung, aus der das Sprachperzept stammt, mit der Position der Person fusioniert. Hervorzuheben ist, dass aufgrund der Notwendigkeit von positionsbestimmenden Attributen einer Person, die Lokalisation von Sprechern erst möglich ist, wenn für Beine, Gesicht oder Oberkörper einer Person bereits ein Anchor aufgebaut wurde.

Oberkörperdetektion

Auch wenn das Gesicht einer Person schon auf visuelle Weise erfasst wird, so ist es nur detektierbar, wenn sich die Person dem Roboter zuwendet. Möchte die Person den Roboter beispielsweise an einen anderen Ort führen, so ist ihr Gesicht für den Roboter nicht sichtbar, falls sich die Person dabei vorwärts bewegt. Daher ist es hilfreich, weitere visuelle Merkmale einer Person

zu betrachten. Hier bietet sich die Oberkörperbekleidung an, da diese oft von allen Seiten das gleiche Erscheinungsbild aufweist. Beispielsweise wird auf dem Roboter AMELIA [Wal00] ein farbbasiertes Tracking von Gesicht und Oberkörper einer Person durchgeführt. Diese Merkmale werden allerdings nur in Kombination betrachtet, so dass die Person verloren geht, wenn sie sich umdreht. Das Tracking stellt hier die Basis dar, um den Roboter sowohl mit statischen als auch mit dynamischen Gesten dazu zu instruieren, Abfall auf dem Boden einzusammeln. Für den Roboter *BUGNOID* [Doi01] wird neben farbbasierter Gesichtsdetektion eine histogrammbasierte Oberkörpererkennung eingesetzt, um Personen folgen zu können. Die zugehörigen Berechnungen werden allerdings nicht auf dem Roboter ausgeführt, sondern auf externen Rechnern, und Instruktionen können nur durch Nicken und Kopfschütteln gegeben werden.

Um das Anchoring-System mit Informationen über die Position des Oberkörpers einer betrachteten Person zu versorgen, werden 256×192 Bildpunkte große Kamerabilder untersucht, um den Bildbereich zu finden, der zu einem zuvor gelernten Farbmodell der Oberkörperbekleidung passt. Ist ein solcher Bereich gefunden, kann die Richtung der Person relativ zum Roboter bestimmt werden, indem die Ausrichtung der Kamera und die Position des Oberkörpers im Bild berücksichtigt werden. Eine Kombination von mehreren Gauß-Funktionen wird verwendet, um die Farbverteilung der Oberkörperbekleidung zu repräsentieren, da solch ein Parametermodell eine angemessen effiziente Modellierung einer zuvor bekannten Farbverteilung darstellt. Bisherige Anwendungen, die einen Satz von Gauß-Verteilungen verwenden, um z.B. eine Getränkedose [McK99] oder Gesichter [Oli00] unter sich verändernden Lichtverhältnissen zu verfolgen, haben die Flexibilität eines solchen Parametermodells gezeigt.

Als Farbrepräsentation wurde der LUV-Farbraum [Wys82] gewählt, da dieser Raum die menschliche Wahrnehmung am besten widerspiegelt. Dieses Vorgehen erlaubt bei der Berechnung, die bestimmt, ob die beobachteten Farben mit denen im Farbmodell übereinstimmen, die Verwendung eines homogenen Abstandsmaßes. Die Initialisierung des Farbmodells wird durchgeführt, nachdem das Gesicht der zu verfolgenden Person das erste Mal detektiert wurde. Da angenommen wird, dass sich der Oberkörperbereich 35 cm unterhalb des Gesichts befindet, kann unter Verwendung des Anchoring-Systems der entsprechende Bereich im Kamerabild bestimmt werden. An dieser Position im Kamerabild wird ein elliptischer Bildbereich ausgewählt, um das initiale Farbmodell zu erstellen. Die Parameter der einzelnen Gauß-Funktionen werden unter Verwendung eines *Cluster*-Algorithmus' (*k-means*) aus dem ESMERALDA-System [Fin99] bestimmt. Um die Farbverteilung einer typischen Oberbekleidung zu modellieren, reicht die Kombination von drei Gauß-Funktionen in der Regel aus, um gute Resultate zu erzielen [Fri04]. Die Anzahl der Gauß-Verteilungen muss allerdings entsprechend erhöht werden, wenn Personen sehr bunte Kleidung tragen.

Beindetektion

Auf mobilen Robotern werden 2D-Laser-Entfernungsmesser zwar sehr häufig eingesetzt, allerdings hauptsächlich zur Lokalisation des Roboters und zur Detektion von Hindernissen. Ein solcher Entfernungsmesser kann aber auch zur Detektion von Personen verwendet werden. Ge-

wöhnlich werden diese Geräte in mobilen Robotern aus Gewichts- und Sicherheitsgründen im unteren Bereich des Systems installiert. In diesem Fall befinden sich Beine von anwesenden Personen im Sichtbereich des Lasers. In Entfernungsmessungen ergeben Beinpaare charakteristische Muster, die einfach detektiert werden können [Kle02, Fri03]. Aus den Detektionsergebnissen können der Abstand und die Richtung einer Person relativ zum Roboter bestimmt werden.

Das Erkennen von Beinen mit Hilfe von Laser-Entfernungsmessern wurde bereits für mobile Systeme betrachtet. Im Ansatz von Schraft et al. [Sch01a] werden für jedes Objekt, das in einer Messung wahrgenommen wird, Merkmale wie Durchmesser, Form und Abstand bestimmt. Mit Hilfe von *Fuzzy Logic* wird dann entschieden, bei welchen Objekten es sich um Beinpaare handelt. Schulz et al. [Sch01b] nehmen an, dass Beine lokale Minima in den Entfernungsmessungen erzeugen. Da aber auch andere Objekte, wie z.B. Tischbeine, solche Muster erzeugen, werden zusätzlich sich bewegende Objekte von statischen Objekten unterschieden.

Der Laser-Entfernungsmesser auf BIRON ist auf einer Höhe von 30 cm montiert und nimmt Messwerte in einem 180°-Sichtbereich vor dem Roboter auf. Die Winkelauflösung beträgt 0.5°, wodurch sich 361 Messwerte pro Messung ergeben. Abbildung 5.8 zeigt eine Beispielmessung in der sich eine Person direkt vor dem Roboter befindet. Generell kann eine Person durch zwei nah beieinander liegende Segmente lokalisiert werden. Ein Segment in einer Entfernungsmessung besteht aus Messpunkten, die ungefähr denselben Abstand zum Messgerät aufweisen. Solche Segmente stammen in der Regel von ebenen Oberflächen eines einzelnen Objektes. Große Abstände zwischen zwei benachbarten Messwerten ergeben sich hingegen durch Kanten oder Verdeckungen. Daher wird ein einzelnes Bein gewöhnlich als ein Segment wahrgenommen.

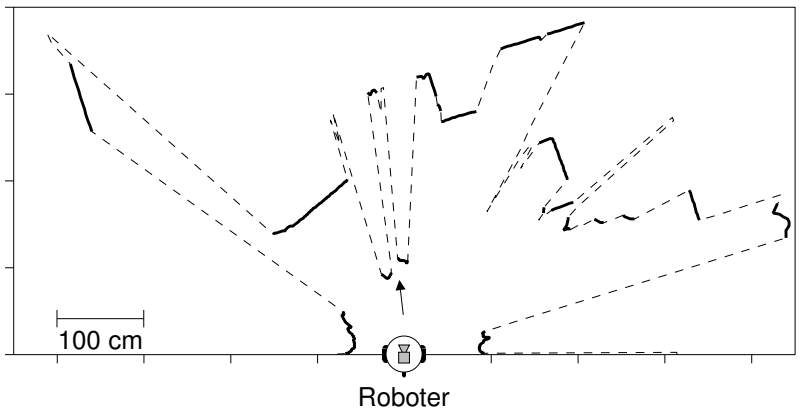


Abbildung 5.8.: Ein Beispiel einer Laser-Entfernungsmessung. Der Pfeil markiert ein Beinpaar.

Im Einzelnen geschieht die Detektion von Beinpaaren in drei Schritten: Zuerst wird eine „Segmentierung“, dann eine „Klassifikation“ und zuletzt eine „Gruppierung“ durchgeführt. Im ersten Schritt wird eine Laser-Messung in Segmente unterteilt. Jedes Segment besteht aus einer maximalen Anzahl von aufeinander folgenden Messwerten, wobei sich zwei benachbarte Werte um höchstens 75 mm unterscheiden dürfen. Im anschließenden Schritt wird für jedes Segment bestimmt, ob es sich um ein Bein handelt oder nicht. Dazu werden folgende Merkmale betrachtet: die Anzahl (n), die mittlere Entfernung (μ) und die Standardabweichung (σ) der im Segment

enthaltenen Messwerte, die Breite (b) des Segmentes senkrecht zum Laserstrahl und die Abstände (d_1 und d_2) zu den direkt benachbarten Segmenten. Gute Ergebnisse um ein Segment korrekt als Bein zu klassifizieren haben sich mit der folgenden Bedingung erzielen lassen:

$$\begin{aligned}(n > 4) \quad &\wedge \quad (\mu < 3000\,mm) \wedge (\sigma < 40\,mm) \wedge (50\,mm < b < 250\,mm) \\ &\wedge \quad (\max(d_1, d_2) > 250\,mm) \wedge (\min(d_1, d_2) > -50\,mm)\end{aligned}$$

Im letzten Schritt werden einzelne Beine abhängig von ihren Abständen zueinander zu Beinpaaren gruppiert. Dabei ergeben zwei Beine ein Paar, falls ihr Abstand unter 500 mm liegt.

In bestimmten Fällen kann ein Bein einer Person durch ihr anderes Bein verdeckt sein, so dass auch nur ein Bein detektiert werden kann. Um diese Information trotzdem nutzen zu können, beschreiben die durch dieses perzeptuelle System generierten Perzepte sowohl alle detektierten Beinpaare als auch alle verbleibenden einzelnen Beine. Daher wird neben den Attributen, welche die Distanz und die Richtung einer Person relativ zum Roboter angeben, noch ein weiteres Attribut verwendet, welches anzeigt, ob das jeweilige Perzept ein Bein oder ein Beinpaar beschreibt.

5.3.2. Anchoring von Bestandteilen einer Person

Die Besonderheiten der einzelnen Anchoring-Prozesse für die Personenteile Gesicht, Sprache, Oberkörper und Beine spiegeln sich in ihren drei Hauptfunktionalitäten wider. Die *Find*-Funktionen der Anchoring-Prozesse für Gesicht und Beine akzeptieren nur Perzepte, deren Distanz zum Roboter weniger als 3 m beträgt. Zusätzlich muss das ausgewählte Beinperzept mit dem Prädikat *Ist-Paar* übereinstimmen, d.h. es müssen beide Beine der Person wahrgenommen werden. Falls der Gesichts-Anchoring-Prozess zum *Anchor Of Interest* gehört, so wird in der *Find*-Funktion zuerst geprüft, ob sich das Sichtfeld der Kamera und die Personenposition, welche durch den *Anchor Of Interest* bereitgestellt wird, überschneiden. Ist dies nicht der Fall, so wird die Kamera in die Richtung bewegt, aus der das Gesichtsperzept erwartet wird. Die *Find*-Funktion des Sprach-Anchoring-Prozesses kann erst dann einen *grounded* Anchor aufbauen, wenn der Abstand zur Person und deren Größe bekannt ist. Die Größe einer Person wird allerdings geschätzt, wenn sie nicht bekannt ist. Der Oberkörper kann nur gefunden werden, wenn das Gesicht bereits verfolgt wird. Daher erstellt das *Find* des Oberkörper-Anchoring-Prozesses ein entsprechendes Farbmodell erst, nachdem das Gesicht zum ersten Mal erkannt wurde.

Die Funktionalitäten *Track* und *Reacquire* der einzelnen monomodalen Anchoring-Prozesse sind sehr ähnlich zueinander. Sie versuchen alle jeweils ein Perzept zu akquirieren, welches sich am nächsten zu der geschätzten Position des beobachteten Objektes befindet. Gleichzeitig berücksichtigen diese Funktionen die Einschränkungen, die durch das Kompositionsmodell für Personen vorgegeben sind. Dabei schätzen die *Track*-Funktionen die Position des zu erwartenden Perzeptes basierend auf der Position des zuletzt wahrgenommenen Perzeptes. Für die Beine einer Person gilt, dass nicht unbedingt beide Beine sichtbar sein müssen, d.h. das Prädikat *Ist-Paar* wird hier nicht berücksichtigt. Im Gegensatz zum *Track* basieren die Schätzungen der *Reacquire*-Funktionen auf der Position der Person, die vom multimodalen Anchor bereitgestellt wird. Falls

sich der Gesichts- oder der Oberkörper-Anchoring-Prozess der so genannten *„Person Of Interest“* im *Track* oder im *Reacquire* befindet, so steuert die jeweilige Funktionen die Kamera des Roboters. Durch das Ausrichten der Kamera wird sichergestellt, dass sich die geschätzte Position des zu erwartenden Perzeptes nicht aus dem Sichtfeld der Kamera bewegt.

5.3.3. Anchoring einer gesamten Person

Der Personen-Anchoring-Prozess erhält individuelle Signaturen, welche von den monomodalen Anchoring-Prozessen für Gesicht, Sprache, Oberkörper und Beine geliefert werden. Wichtig ist hierbei, dass diese Daten asynchron durch den multimodalen Anchoring-Prozess verarbeitet werden. Abbildung 5.9 zeigt das multimodale Anchoring des Gesamtobjektes Person. Das verwendete Kompositionsmodell beschreibt die empirisch bestimmten Relationen der Bestandteile einer Person. Abbildung 5.10 zeigt die Relationen für Gesicht und Beine bezogen auf die Person selbst.

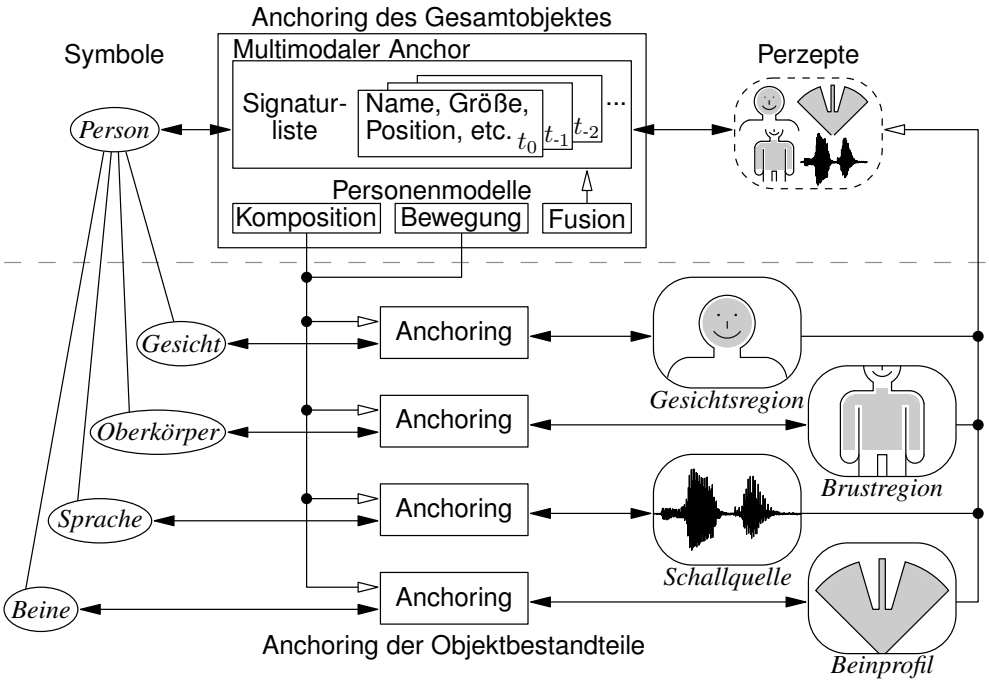


Abbildung 5.9.: Anchoring einer gesamten Person basierend auf ihren Komponenten Gesicht, Oberkörper, Sprache und Beine.

Alle Attribute des multimodalen Anchorings von Personen, die sich auf räumliche Positionen beziehen, sind nicht durch skalare Werte, sondern durch Gauß-Verteilungen beschrieben. Dies erlaubt es, Positionsungenauigkeiten zu modellieren. Für diese Attribute von Perzepten kann die Varianz der Gauß-Verteilungen bestimmt werden, indem die Ungenauigkeit der entsprechenden Sensoren gemessen wird. Das Bewegungsmodell definiert, wie eine Position für den Zeitpunkt $t(i+1)$ geschätzt werden kann, basierend auf der bekannten Position zum Zeitpunkt $t(i)$: Der

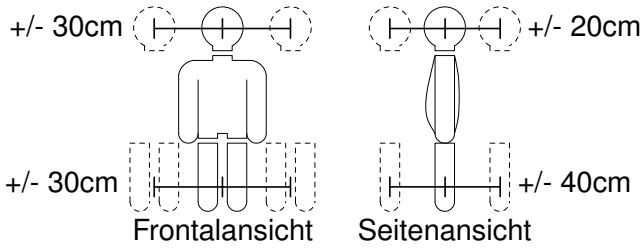


Abbildung 5.10.: Das Kompositionsmodell zur Zuordnung von Perzepten am Beispiel für Gesicht und Beine.

Erwartungswert bleibt dabei gleich, d.h. es wird keine Geschwindigkeit angenommen, während die Varianz linear mit der Zeit ansteigt, was die wachsende Unsicherheit ausdrückt.

Die Attributwerte, die in der Signaturliste des Personen-Anchoring-Prozesses enthalten sind, werden aktualisiert, indem die Gauß-Verteilungen eines jeden Attributwertes mit den Gauß-Repräsentationen der entsprechenden Attributwerte von den neuen Perzepten multipliziert werden. Dies resultiert in den folgenden Aktualisierungsformeln, die durch das Fusionsmodell berechnet werden:

$$\mu_{t(i)} = \frac{\mu_{t(i-1)}\sigma_p + \mu_p\sigma_{t(i-1)}}{\sigma_{t(i-1)} + \sigma_p} \qquad \sigma_{t(i)} = \frac{\sigma_{t(i-1)}\sigma_p}{\sigma_{t(i-1)} + \sigma_p}$$

Der resultierende Erwartungswert $\mu_{t(i)}$ ist eine gewichtete Summe des alten Erwartungswertes $\mu_{t(i-1)}$ und des Erwartungswertes des Perzeptes μ_p. Die zugehörigen Gewichte sind durch die Varianz der alten Position in der Signaturliste bzw. durch die des neuen Perzeptes gegeben. Somit hat der Erwartungswert, der zu der geringeren Varianz gehört, einen höheren Einfluss.

Die Attributwerte der Person, die mit den Signaturen der sich im Zustand *grounded* befindenden monomodalen Anchors aktualisiert werden, sind der Winkel ϕ_p und der Abstand d_p relativ zum Roboter, die Höhe h_p des Gesichts relativ zum Boden und der Name N_p der Person. Die Initialisierung der Werte ϕ_p und d_p wird ausgeführt, wenn ein entsprechender monomodaler Anchor zum ersten Mal in den Zustand *grounded* übergeht. Die Attributwerte h_p und N_p können erst initialisiert werden, nachdem die erste Signatur vom Gesichts-Anchoring-Prozess empfangen wurde. Während des multimodalen Trackings sorgt das Personenfusionsmodell dafür, dass die Personenposition ohne große Sprünge durch die Positionsdaten der Personenbestandteile aktualisiert wird. Im Gegensatz dazu können h_p und N_p nur aktualisiert werden, wenn Gesichtssignaturen verarbeitet werden. Da es sich bei dem Namen der Person um keine räumliche Position handelt, wird N_p nicht durch eine Gauß-Verteilung beschrieben. Die Aktualisierung von N_p entspricht daher einer Ersetzung mit dem neu ermittelten Wert.

Um das Konzept des multimodalen Anchorings zu veranschaulichen, ist in Abbildung 5.11 ein schematisches Beispiel für die Verfolgung einer einzelnen Person dargestellt. Aus Gründen der Übersichtlichkeit werden nur zwei Merkmale betrachtet: das Gesicht und die Beine der Person. Der Personen-Anchor ist hier durch den Torso der Person symbolisiert. Die Abbildung stellt sechs aufeinander folgende Zeitschritte zu Beginn eines Anchoring-Vorgangs dar:

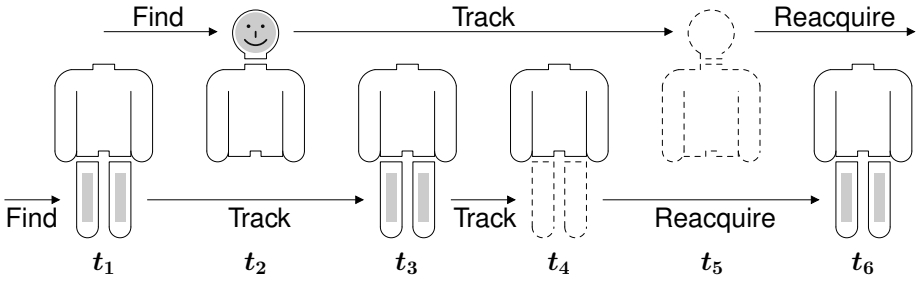


Abbildung 5.11.: Ein schematisches Beispiel für das Anchoring einer einzelnen Person anhand von Gesicht und Beinen (Perzepte: grau, Anchors: schwarze Umrandungen).

t_1: Das multimodale Personen-Anchoring wird gestartet und alle monomodalen Anchoring-Prozesse führen ihr *Find* aus. Die Beindetektion erzeugt ein Beinpaarperzept und die Beine werden zum ersten Mal mit dem monomodalen Anchor verknüpft. Der Bein-Anchoring-Prozess wechselt vom *Find* zum *Track*. Anschließend wird die Personenposition, welche im multimodalen Anchoring-Prozess enthalten ist, initialisiert und der Personen-Anchor wechselt somit in den Zustand *grounded*. Nun ist die *Find*-Funktion des Gesichts-Anchoring-Prozesses in der Lage, die Kamera in die gewünschte Richtung zu steuern.

t_2: Die Gesichtsdetektion erzeugt ein Gesichtsperzept und der Gesichts-Anchor wechselt in den Zustand *grounded*. Der Gesichts-Anchoring-Prozess wechselt ebenfalls vom *Find* zum *Track* und der Personen-Anchor wird entsprechend aktualisiert.

t_3: Die Beindetektion erzeugt erneut ein Beinperzept. Unter Verwendung der *Track*-Funktion wird der Bein-Anchor sowie der Personen-Anchor aktualisiert.

t_4: In diesem Zeitschritt werden neue Laser-Daten verarbeitet, aber es wird kein passendes Perzept von dem Bein-Anchoring-Prozess gefunden. Daher wechselt er vom *Track* zum *Reacquire*. Der Personen-Anchor wird folglich auch nicht aktualisiert.

t_5: Ein neues Kamerabild wird verarbeitet, aber es wird kein Gesichtsperzept, das zu der Schätzung der Gesichtsposition passt, gefunden. Daher wechselt der Gesichts-Anchoring-Prozess ebenfalls vom *Track* zum *Reacquire*. Nun nimmt auch der Personen-Anchor den Zustand *ungrounded* ein, da sich weder die Beine noch das Gesicht im Zustand *grounded* befinden.

t_6: In den neu akquirierten Laser-Daten wird ein Bein-Perzept gefunden, welches mit der geschätzten Position der Person übereinstimmt. Nun wechseln sowohl der Zustand des Bein-Anchors als auch der Zustand des Personen-Anchors zu *grounded* zurück.

5.3.4. Erweiterung auf mehrere Personen

Durch die Verwendung von mehreren multimodalen Anchoring-Prozessen können zwar mehrere Personen passiv erfasst, aber nur eine Person aktiv verfolgt werden. Aus diesem Grund wird in

der Komponente, die alle multimodalen Anchoring-Prozesse verwaltet, zu jedem Zeitpunkt immer nur genau eine Person als *Person Of Interest* definiert. Auf Grundlage des zu dieser Person gehörenden multimodalen Anchoring-Prozesses wird dann die Kamera bzw. auch die Roboterbasis entsprechend ausgerichtet. Die Auswahl, welche Person die *Person Of Interest* darstellt, wird durch die Aufmerksamkeitssteuerung für Personen bestimmt, die in Abschnitt 6.1 beschrieben wird.

5.4. Zusammenfassung

In diesem Kapitel wurde ein im Rahmen dieser Arbeit entwickeltes, multimodales System zur Integration von Daten, die von unterschiedlichen Sensoren auf einem mobilen Roboter wahrgenommen werden, beschrieben. Dieses System basiert auf multimodalem Anchoring, welches eine Erweiterung des Anchoring-Verfahrens von Coradeschi und Saffiotti [Cor01] darstellt. Mit multimodalem Anchoring ist es möglich, gleichzeitig mehrere komplexe Objekte, wie z.B. Menschen, mit mehreren unterschiedlichen Sensoren über die Zeit zu verfolgen. Dabei berücksichtigt das System, dass nicht zu jedem Zeitpunkt alle Modalitäten wahrgenommen werden können. Beispielsweise sind die Beine einer Person nicht sichtbar, wenn diese durch einen Papierkorb verdeckt werden, oder die Stimme einer Person nicht wahrnehmbar, wenn diese gerade nicht spricht. Sollten entsprechende Perzepte nach einer solchen Unterbrechung wieder erfasst werden, so können sie auch wieder der entsprechenden Person zugeordnet werden. Da das System in der Lage ist, mehrere Personen parallel zu beobachten, können diese auch differenziert werden. Dies ist wichtig, damit der Roboter nicht Gefahr läuft, unterschiedliche Personen zu verwechseln. Somit ermöglicht multimodales Anchoring ein robustes Tracking von Personen. Zusätzlich können steuerbare Sensoren und auch die Basis des Roboters auf eine ausgewählte Person ausgerichtet werden, um die Aufmerksamkeit des Roboters auf diesen Menschen fokussieren zu können. Multimodales Anchoring stellt daher die wichtigste Grundlage für eine anspruchsvolle Interaktion zwischen dem Roboter und einem Menschen dar.

6. Interaktionsmodule für BIRON

Neben dem Personen-Tracking, das im vorigen Kapitel beschrieben wurde, war es ein Ziel dieser Arbeit, noch weitere Komponenten in das Gesamtsystem zu integrieren, um die grundlegenden Funktionalitäten für einen *Robot Companion* zu erhalten. Diese Komponenten, die zur Realisierung einer natürlichen Interaktion mit Menschen benötigt werden, sind im Rahmen weiterer Promotionsvorhaben entstanden. Sie werden im Folgenden kurz beschrieben, um ihre Aufgaben im Gesamtsystem zu erläutern. Dazu zählen die Aufmerksamkeitssteuerung für Personen (Abschnitt 6.1), die Aufmerksamkeitssteuerung für Objekte (Abschnitt 6.2) und die Dialogsteuerung, die für die sprachliche Interaktion mit dem Benutzer verantwortlich ist (Abschnitt 6.3).

6.1. Die Aufmerksamkeitssteuerung für Personen

Das verwendete Personen-Tracking ist in der Lage, Menschen auf multimodale Weise zu verfolgen und bietet somit eine solide Basis für eine natürliche Interaktion zwischen Mensch und Roboter. Allerdings haben die Sensoren des Roboters einen eingeschränkten Wahrnehmungsbereich, so dass nicht immer alle Personen, die sich in der Nähe des Roboters befinden, zur gleichen Zeit vollständig erfasst werden können. Dieser Umstand erschwert die Aufgabe, unter diesen Personen einen Interaktionspartner zu finden, der mit dem Roboter interagieren möchte [Lan02].

Bei Systemen, die einen statischen Aufbau darstellen, kann die Detektion eines Interaktionspartners relativ einfach erreicht werden. Beispielsweise muss ein potentieller Benutzer eines intelligenten Informationskiosks [Pav00a] oder eines interaktiven Kasinokiosks [Pav00b] zur Interaktion einen vorgegebenen Bereich vor dem entsprechenden Gerät einnehmen. In intelligenten Räumen werden Sensoren wiederum so platziert, dass alle anwesenden Personen gleichzeitig erfasst werden können (siehe z.B. [Sti01]) und eine bestimmte Person zur Interaktion nur noch ausgewählt werden muss.

Im Gegensatz zu den gerade genannten Szenarien agiert ein mobiler Roboter jedoch nicht in einer solchen geschlossenen bzw. kontrollierten Umgebung. Folglich muss jeder Mensch, der sich dem Roboter nähert oder an ihm vorübergeht, als potentieller Interaktionspartner angesehen werden. Zusätzlich ist es möglich, dass mehrere Personen anwesend sind, wodurch die Detektion eines Interaktionspartners erheblich erschwert wird. Beispielsweise müssen sprachliche Äußerungen einer Person nicht unbedingt an den Roboter gerichtet sein, sondern können auch einen anderen

Menschen als Ziel haben. Um dieses Problem zu umgehen, wurden bereits diverse Lösungen entwickelt. Damit z.B. eine Interaktion mit dem Tour-Führer RHINO [Bur98] begonnen werden kann, muss der Benutzer einen Knopf auf dem Roboter betätigen. Um den Roboter *iSHA* [Suz03] auf sich aufmerksam zu machen, muss eine Person in die Hände klatschen. Erst dann nimmt der Roboter auch sprachliche Kommandos von diesem Benutzer entgegen [Suz02]. Der Roboter Jijo-2 [Aso01] erwartet wiederum ein spezielles Schlüsselwort, bevor er eine Person als Interaktionspartner betrachtet.

Um ein wirklich natürliches Interaktionsverhalten zu realisieren, sollte ein Roboter allerdings selbstständig erkennen können, welche Person in seiner Umgebung einen potentiellen Interaktionspartner darstellt. Des Weiteren sollte er seine Sensoren auf diese Person ausrichten können, um sie besser wahrzunehmen. Dieser Mechanismus wird „selektive Aufmerksamkeit“ genannt und stellt die Grundlage für eine entsprechende Aufmerksamkeitssteuerung eines Roboters dar. Ein solches System ist auch ein Bestandteil in der Realisierung der humanoiden Robotersysteme SIG [Nak01], ROBITA [Mat99b] und ISAC [Kaw00a]. Jeder dieser drei Roboter verwendet eine Kombination aus visueller Gesichtserkennung und Geräuschquellenlokalisation, um einen potentiellen Interaktionspartner zu detektieren. Bei SIG handelt es sich um einen humanoiden Torso, der als Empfangsdame eingesetzt wird [Oku02]. Der Fokus der Aufmerksamkeit wird immer auf die Person ausgerichtet, die gerade spricht und sich dabei dem Roboter nähert oder bereits in seiner Nähe befindet. ROBITA kann neben der Detektion von Sprechern auch den Empfänger einer gesprochenen Äußerung bestimmen. Somit kann er unterscheiden, ob eine Äußerung an ihn oder an eine andere anwesende Person gerichtet ist. ROBITA kann beispielsweise an Gesprächen über das Thema *Baseball* teilnehmen [Mat01b]. Der Roboter bezieht dazu Informationen über ausgetragene Spiele aus dem Internet und kann somit einen menschlichen Gesprächspartner korrigieren, falls er ein Spielergebnis falsch wiedergibt. ISAC fokussiert ebenfalls eine Person in seiner Nähe, wenn sie zu sprechen beginnt. Dazu richtet der Roboter seine Stereokamera auf sie aus. Diese Person kann dann zusätzlich anhand ihres Gesichts und ihrer Hände visuell verfolgt werden, auch wenn sie sich bewegt und dabei nicht mehr sprechen sollte.

Neben dem Ausrichten der Aufmerksamkeit ist es allerdings ebenfalls äußerst wichtig für einen Roboter, den Personen in seiner Umgebung anzuzeigen, dass sie aktiv wahrgenommen werden. Beispielsweise kann jeder der drei oben genannten Roboter Rückmeldung darüber geben, welche anwesende Person gerade als aktueller Interaktionspartner betrachtet wird. Dazu dreht SIG immer seinen kompletten Körper in die Richtung des entsprechenden Menschen. ROBITA kann hingegen seinen Körper, seinen Kopf und seinen Blick individuell ausrichten, um verschiedene Stadien der aktuellen Kommunikation anzuzeigen. Außerdem werden Augenbrauen verwendet, um dem Roboter mehr Ausdrucksfähigkeit zu verleihen. Schließlich besitzt ROBITA einen einfachen Arm, um auf Objekte in seiner Umgebung zeigen zu können [Toj00]. Dies hilft bei der Auflösung von Mehrdeutigkeiten, falls in einer Interaktion nicht klar sein sollte, auf welchen Gegenstand sich der Roboter bezieht. ISAC kann hingegen seine Hand ausstrecken, um die Person zu begrüßen, die sich in seiner unmittelbaren Nähe befindet.

Häufig wird auch die Verwendung eines Gesichts als ein wichtiger Feedback-Mechanismus für einen Roboter angesehen, da es dazu genutzt werden kann, um die soziale Intelligenz des Sy-

stems zu visualisieren. Der Roboter Kismet kann beispielsweise die Emotionen Ärger, Ermüdung, Angst, Abneigung, Begeisterung, Freude, Interesse, Trauer und Überraschung darstellen, indem er verschiedene Gesichtsausdrücke zeigt [Bre99]. Diese Emotionen können unter anderem genutzt werden, um einen Benutzer des Roboters so zu beeinflussen, dass dieser sich dem Roboter gegenüber möglichst kooperativ verhält. Dieses Vorgehen wirkt sich daher in der Regel positiv auf die Interaktion aus [Bro99]. Beispielsweise reagiert Kismet interessiert und fröhlich, falls das Gesicht des Benutzers dem Roboter zugewandt ist und sich der Mensch nur langsam bewegt. Sollte sich der Benutzer zu schnell bewegen, so wirkt der Roboter abgeneigt oder frustriert, da es in diesem Fall für das System schwieriger ist, den Benutzer zu verfolgen. Außerdem verfügt Kismet über eine so genannte *Comfort Zone*: Sollte ein Benutzer zu nah oder zu weit entfernt sein, so kann ihn Kismet bitten, eine für die Sensoren des Roboters günstigere Position einzunehmen [Bre00]. Ein weiterer Roboter, der einen entsprechenden Satz an Emotionen besitzt, ist Pong [Har01]. Die Reaktionen dieses Systems basieren ähnlich wie bei Kismet auf einer Kombination aus Spracheingabe, Sprecherlokalisation und Gesichtsdetektion.

Die für BIRON entwickelte Aufmerksamkeitssteuerung für Personen wurde von S. Lang entwickelt und realisiert ebenfalls den Mechanismus der selektiven Aufmerksamkeit [Lan03]. Sie unterscheidet sich allerdings von den oben genannten Systemen, da sie im Prinzip in zwei Phasen unterteilt ist. In der ersten Phase sucht der Roboter unter allen Personen, die sich in seiner Nähe befinden, nach potentiellen Interaktionspartnern. Dazu wird immer die Person betrachtet, die mit der höchsten Wahrscheinlichkeit mit dem Roboter interagieren möchte. Um dies zu entscheiden, werden entsprechende Informationen von der Personenverfolgung verarbeitet, auf die die Aufmerksamkeitssteuerung aufsetzt (siehe Abb. 6.1). Somit ergibt sich eine reflexive Aufmerksamkeit, die auch bottom-up gesteuerte Aufmerksamkeit genannt wird, da sie nur durch Sensordaten beeinflusst wird. Wenn ein potentieller Interaktionspartner in der ersten Phase eine Interaktion mit dem Roboter initiiert, so wechselt die Aufmerksamkeitssteuerung in die zweite Phase. Auf diese Weise wird der potentielle Interaktionspartner zum echten Interaktionspartner. Folglich betrachtet die Aufmerksamkeitssteuerung nur noch diese Person als relevant und nimmt daher auch nur noch von ihr Instruktionen entgegen. Diese Phase realisiert also eine bewusste Aufmerksamkeit, die daher auch als top-down gesteuerte Aufmerksamkeit bezeichnet wird. Beide Phasen der Aufmerksamkeitssteuerung haben die Aufgabe, die Sensoren des Roboters so auszurichten, dass die Wahrnehmung des Systems für die jeweilige Situation möglichst optimal ist. Des Weiteren sind sie dafür zuständig, dem Benutzer ein entsprechendes Feedback über die Aufmerksamkeit des Roboters zu vermitteln. Um verschiedene Aufmerksamkeitszustände zu realisieren, basiert die Aufmerksamkeitssteuerung auf einem endlichen Automaten [Haa04].

Wenn die bottom-up gesteuerte Aufmerksamkeit aktiv ist, so ist keine bestimmte Person als Interaktionspartner für den Roboter ausgewählt. Welche der anwesenden Personen in dieser Phase die so genannte *Person Of Interest* darstellt (vgl. Abschnitt 5.3.4), hängt allein von den Informationen ab, die von der Personenverfolgung geliefert werden. Um die *Person Of Interest* auszuwählen, werden alle betrachteten Personen in drei Kategorien eingeteilt. Die erste Kategorie enthält alle Personen, die nicht sprechen und daher am wenigsten interessant für den Roboter sind. Die zweite Kategorie besteht aus allen Personen, die zwar sprechen, aber deren Gesichter nicht von der Kamera des Roboters erfasst werden. Dies kann zwei Gründe haben: Entweder

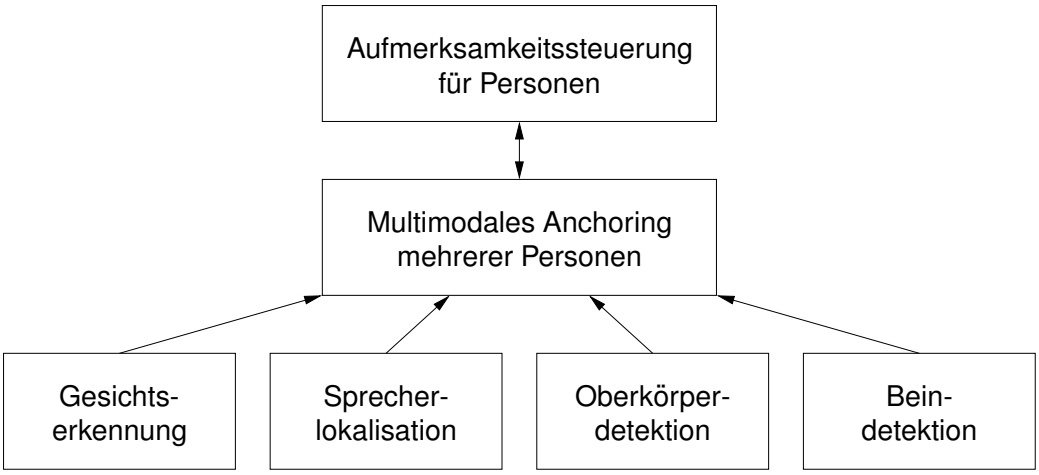


Abbildung 6.1.: Die Aufmerksamkeitssteuerung für Personen basiert auf der multimodalen Personenverfolgung und nutzt somit ebenfalls multimodale Informationen.

schaut die betroffene Person nicht in Richtung der Kamera oder die Person befindet sich gar nicht in deren Sichtbereich. Die Personen, die sich in der dritten Kategorie befinden, sind schließlich am interessantesten für den Roboter, da sie sprechen und gleichzeitig in seine Richtung schauen. In diesem Fall nimmt der Roboter an, dass er angesprochen wird und betrachtet daher diese Personen als potentielle Interaktionspartner. Wenn eine Person der dritten Kategorie zugeordnet ist, so wird sie augenblicklich zur *Person Of Interest* und bleibt dies auch solange, bis sie in eine der anderen Kategorien wechselt, weil sie z.B. aufhört zu sprechen oder in eine andere Richtung schaut. Falls keine Person den Status des potentiellen Interaktionspartners besitzt, so wählt die Aufmerksamkeitssteuerung immer die Person mit der höchsten Relevanz aus, d.h. eine Person der zweiten Kategorie wird einer Person der ersten Kategorie bevorzugt. Falls sich in einer Kategorie gleichzeitig mehrere Personen befinden sollten, so entscheidet sich die Aufmerksamkeitssteuerung für die Person, die am längsten nicht mehr betrachtet wurde. Somit wird jede anwesende Person irgendwann einmal angeschaut. Des Weiteren werden diese Personen abwechselnd fokussiert, um zusätzliche Informationen, wie z.B. deren Identität zu erfassen. Dabei wird eine Person immer nur für eine bestimmte Zeit ausgewählt. Danach wird sie für eine Weile ignoriert, um die anderen Personen betrachten zu können. Eine Person wird fokussiert, indem die Kamera des Roboters auf sie ausgerichtet wird. Zusätzlich wird die Roboterbasis so orientiert, dass der Laser-Entfernungsmesser möglichst alle anwesenden Personen erfassen kann.

Die top-down gesteuerte Aufmerksamkeit wird aktiviert, indem ein potentieller Interaktionspartner den Roboter mit einer Begrüßung oder einer Anweisung anspricht. Dadurch wird diese Person zum Interaktionspartner und der Roboter beginnt mit ihr zu interagieren. Während der Interaktionsphase bleibt der Aufmerksamkeitsfokus des Roboters auf diese Person gerichtet, auch wenn sie nicht mehr sprechen sollte. Im Gegensatz zur bottom-up gesteuerten Aufmerksamkeit wird in dieser Phase ein Wechsel des Aufmerksamkeitszustands ausschließlich durch einen Dialog mit dem Interaktionspartner ausgelöst. Entsprechende Anweisungen empfängt die Aufmerksamkeitssteuerung daher nicht von der Personenverfolgung, sondern über den Execution

Supervisor (siehe Kapitel 7). Mit dem Wechsel des Aufmerksamkeitszustands kann sich auch das Verhalten des Roboters ändern. Dies äußert sich darin, auf welche Weise die Kamera und die Basis des Roboters orientiert werden. Die Kamera wird im Regelfall auf das Gesicht des Interaktionspartners ausgerichtet. In gewissen Interaktionsphasen wird jedoch von diesem Vorgehen abgewichen (vgl. Abschnitt 7.3). Beispielsweise wird die Kamera etwas gesenkt, wenn der Roboter dem Interaktionspartner hinterherfahren soll, um neben dem Gesicht auch den Oberkörper der Person erfassen zu können (siehe Abb. 6.2). Somit ist der Roboter nicht darauf angewiesen, dem Interaktionspartner, ähnlich wie in [Sid99], nur auf Grundlage dessen Gesichts zu verfolgen. Stattdessen nutzt er die multimodalen Informationen der Personenverfolgung, was der Robustheit des Systems dient. Außerdem kann die Kamera auf die Hände des Benutzers ausgerichtet werden, wenn der Roboter eine Zeigegeste erwartet. Deutet der Interaktionspartner dann auf ein Objekt, so wird die Kontrolle über die Hardware an die Aufmerksamkeitssteuerung für Objekte abgegeben (siehe Abschnitt 6.2), bis diese das Objekt akquiriert hat. Im Gegensatz zur Kamera wird die Basis des Roboters immer direkt auf den Interaktionspartner ausgerichtet. Somit können die Mikrofone des Roboters ein möglichst optimales Signal aufzunehmen, was die Spracherkennung erleichtert. Vorwärts bewegt wird die Basis des Roboters allerdings nur, wenn der Roboter dem Interaktionspartner hinterherfahren soll.

Abbildung 6.2.: BIRON folgt einem bestimmten Interaktionspartner, während sich im Hintergrund zwei weitere Personen unterhalten.

Neben dem Ausrichten der Kamera und der Roboterbasis wird von der Aufmerksamkeitssteuerung für Personen zusätzlich ein zweidimensionales Gesicht auf dem Touch-Screen-Display des Roboters angesteuert (siehe Abb. 6.3). Es dient dazu, den Aufmerksamkeitsfokus des Roboters

auf eine für den Menschen intuitive Weise zu visualisieren. Dieses Gesicht verfügt über Augen, Augenbrauen und einen Mund, da diese Merkmale die größte Bedeutung in der Kommunikation zwischen Menschen haben [Eli02]. Beispielsweise kann durch das Ausrichten der Pupillen der für Menschen wichtige Blickkontakt hergestellt werden. Auch wenn es recht primitiv wirken mag, beeinträchtigt das lediglich zweidimensionale Gesicht in der Regel nicht die Qualität der Interaktion. Ein Mensch interagiert selbst mit dem einfachsten künstlichen Gesicht nahezu wie mit einem echten Gesicht eines Menschen. Aus diesem Grund wurde auch für den Roboterkellner José [Eli02] ein solches Gesicht mir Erfolg eingesetzt. Des Weiteren benötigt es zur Darstellung auf der Hardware des Roboters weniger Rechenzeit als ein dreidimensionales Gesicht.

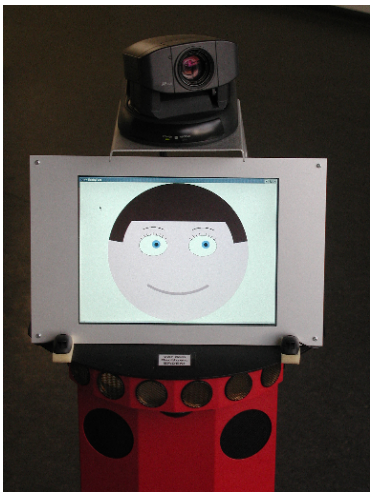


Abbildung 6.3.: Visualisierung der Aufmerksamkeit des Roboters über ein Gesicht, um interne Systemzustände in Form von Emotionen darstellen zu können.

6.2. Die Aufmerksamkeitssteuerung für Objekte

Neben dem Interaktionspartner muss ein *Robot Companion* auch Objekte wahrnehmen können, die er zwecks späterer Interaktion lernen soll. Für die Erfassung von Objekten von einem mobilen Roboter aus bieten sich insbesondere steuerbare Kameras an. Der Grund dafür ist, dass solche Sensoren sich auch gut dazu eignen, um einen menschlichen Instrukteur zu beobachten. Kameras erlauben, dass die Position des Menschen verfolgt und seine auf Gesten basierenden Instruktionen erkannt werden können [Kah96, Kor96, Böh98, Guo99, Per99, Wal00, Ghi02, Rog02]. Ausgehend von einer erkannten Zeigegeste kann somit schließlich in einem Kamerabild der Bereich lokalisiert werden, der ein zu lernendes Objekt enthalten sollte.

In der Regel geht die Erfassung eines Objektes mit einem Dialog zwischen dem Roboter und dem Benutzer einher. Daher steht z.B. auch die Kombination von Sprache und Gestik auf dem Roboter ISAC [Kaw00a] im Vordergrund, um den Roboter anzuweisen. ISAC besitzt dazu so genannte „Object Agents“, die für die Wahrnehmung von Objekten sorgen, auf die sich der Be-

nutzer bezieht [Pet99]. Zeigegesten geben primär Aufschluss über die Position eines Objektes. Sie können einem Roboter aber auch zusätzliche Hinweise bieten. Dem humanoiden Roboter WABIAN kann z.B. gezeigt werden, an welcher Stelle ein bestimmtes Objekt gegriffen werden muss [Has02]. Sprachliche Instruktionen werden eher dazu genutzt, einem Roboter nicht sichtbare Informationen zu vermitteln. Beispielsweise kann dem Roboter ARMAR mitgeteilt werden, wie mit einem bestimmten Objekt zu verfahren ist [Asf00].

Die Interaktion mit den oben genannten Systemen wirkt allerdings häufig sehr starr. Mit einer natürlicheren Form des Lernens einer unbekannten Umgebung beschäftigt sich z.B. das Cog-Projekt [Bro99]. Hier ist es das Ziel, den Lernvorgang möglichst einfach, aber dennoch robust zu gestalten. Dieses Vorgehen erfordert nach Ansicht der Autoren die Berücksichtigung der so genannten *„Joint Attention“*, welche im System des Roboters Cog integriert ist, um die Basis für eine fortgeschrittene Mensch-Roboter-Interaktion zu formen [Sca00]. Die *Joint Attention* beschreibt das Verhalten zwischen Menschen, die sich in einer Unterhaltung gegenüberstehen. Wenn sie sich dabei auf Gegenstände und Situationen in ihrem Sichtbereich beziehen, nutzen sie effiziente Beschreibungen in ihren Äußerungen und lassen sogar Wörter aus. Um diese Form der menschlichen Kommunikation verstehen zu können, muss ein Roboter folglich in der Lage sein, die für den aktuellen Kontext relevanten Sensorinformationen selbstständig zu erfassen [Ima01].

BIRON soll ebenfalls auf intuitive Weise instruiert werden können. Dazu wird von A. Haasch zur Zeit eine spezielle Aufmerksamkeitssteuerung für Objekte entwickelt, deren Aufgabe es ist, neu zu lernende Gegenstände in den Fokus des Roboters zu bringen [Haa05, Möl05]. Um ein solches Objekt lokalisieren zu können, muss der Benutzer allerdings darauf zeigen. Die Zeigegeste wird dann durch ein entsprechendes Verfahren detektiert, das von N. Hofemann entwickelt wurde [Hof04]. Zwar kann die Stereokamera des Roboters zur Erkennung von Zeigegesten im dreidimensionalen Raum eingesetzt werden [Sch04], allerdings ist die Auflösung in den zugehörigen Bilder zu gering, um anschließend eine robuste Objektdetektion gewährleisten zu können. Daher fokussiert die Aufmerksamkeitssteuerung Gegenstände, indem sie die auf dem Roboter montierte Pan-Tilt-Kamera auf den vom Benutzer gezeigten Bereich ausrichtet.

Nach dem Ausrichten der Pan-Tilt-Kamera muss die Aufmerksamkeitssteuerung für Objekte außerdem dafür sorgen, dass der fokussierte Gegenstand detektiert wird und dessen Eigenschaften akquiriert werden. Diese Aufgabe wird von einer so genannten Objektdetektion übernommen (Abschnitt 6.2.1). Die dabei erhaltenen Informationen müssen vom Robotersystem anschließend auch entsprechend gespeichert werden, so dass auf dieses Wissen zu einem späteren Zeitpunkt zugegriffen werden kann. Zur Verwaltung dieser Objektdaten ist ein spezielles Szenemodell (Abschnitt 6.2.2) vorgesehen. Sowohl die Objektdetektion als auch das Szenemodell werden neben der Aufmerksamkeitssteuerung für Objekte von A. Haasch für BIRON entwickelt.

6.2.1. Die Detektion von Objekten

Die Aufgabe einer Objektdetektion ist es, Sensordaten mit Modellinformation über Objekte zu vergleichen. Die üblicherweise zur Sensordateninterpretation eingesetzten Modellierungstech-

niken lassen sich prinzipiell in zwei Klassen einteilen: geometrische und erscheinungsbasierte Modelle [Heb95, Pon96]. Typische geometrische Repräsentationsformen sind 3D-Drahtmodelle, welche sich für eine kantenbasierte Bildinterpretation eignen. Erscheinungsbasierte Modelle repräsentieren Objekte und Umgebung hingegen durch einen Satz von echten Sensoransichten. Sie können sich stark bezüglich der Speichermethoden und der Art der berücksichtigten Sensorinformation unterscheiden. Diese Methoden reichen von in Graphen organisierten geometrischen Merkmalen und ihren räumlichen Relationen [Löm04] bis hin zur so genannten *Eigenspace*-Repräsentation auf Bildpunktbasis [Mur95]. Erscheinungsbasierte Modelle vereinfachen zwar den Vergleich zwischen Sensordaten und dem Modell, das Aufbauen und Warten großer Modelldatenbanken sowie die Behandlung generischer Objektklassen ist allerdings schwierig. Des Weiteren hängt die Wahl der Repräsentation auch davon ab, ob das Wissen über entsprechende Objekttypen im Voraus bekannt ist oder erst zur Laufzeit vom System erworben werden kann, da es sich um initial unbekannte Objekttypen handelt.

Bekannte Objekttypen

Einem Robotersystem ist ein Objekttyp bekannt, falls es über einen entsprechenden Objekterkenner verfügt. Sollte der Roboter mehrere Objekttypen kennen, so kann er per Dialog angewiesen werden, welches Objekt in einer von ihm wahrgenommenen Szene gefunden werden soll.

In [Miu03] wird beispielsweise ein Service-Roboter präsentiert, der speziell für die Aufgabe entwickelt wurde, Getränkedosen aus einem Kühlschrank zu holen. Da somit die zu erkennende Objektart bekannt ist, wird das System im Voraus mit mehreren Ansichten einer Dose trainiert. Der Roboter kann dann per Computer angewiesen werden, dem Benutzer ein Getränk zu bringen. Sollte der Roboter dabei auf Mehrdeutigkeiten stoßen, so präsentiert er dem Menschen ein Bild vom Inhalt des Kühlschranks. Der Benutzer wird dann gebeten, das Problem durch eine Anweisung in natürlicher Sprache zu lösen. In dem Fall, dass der Roboter mehrere passende Objekte findet, kann der Benutzer durch eine diskriminative Ortsangabe eines der hervorgehobenen Objekte auswählen. Sollte der Roboter wegen einer teilweisen Überlappung das gewünschte Objekt nicht finden, so kann der Mensch dem Roboter auf ähnliche Weise Hilfestellung geben.

Für BIRON wird ebenfalls ein Objekterkenner verwendet, der im Voraus auf mehrere Objekttypen trainierbar ist, um eine grundlegende Wissensbasis für den Roboter zu realisieren. Das implementierte Verfahren führt ein *Template Matching* durch und basiert auf einer normalisierten Kreuzkorrelation. Dennoch muss davon ausgegangen werden, dass Fehler bei der Erkennung auftreten können. In solchen Fällen muss der Roboter eine entsprechende Rückmeldung geben. Ist beispielsweise ein Objekt vom Roboter aus nicht sichtbar, so muss der Benutzer gebeten werden können für eine bessere Sicht auf das Objekt zu sorgen, indem er z.B. den Roboter repositioniert. Sollte der Roboter nicht genug Informationen besitzen, um das gewünschte Objekt unter mehreren Gegenständen herauszufinden, so sollte der Mensch nach zusätzlichen Anhaltspunkten, wie z.B. nach der Farbe oder der relativen Position des Objektes, gefragt werden können. Die Voraussetzung zur Verwendung dieses Verfahrens ist allerdings, dass die zur Interaktion vorgesehenen Objekte im Voraus bekannt sind, um das initiale Training durchführen zu können.

Unbekannte Objekttypen

Ein Objekttyp ist für ein Robotersystem unbekannt, falls das System zum entsprechenden Zeitpunkt über keinen Objekterkenner verfügt, der den Objekttypen detektieren kann. In diesem Fall muss das Objekterkennungssystem in der Lage sein, das unbekannte Objekt zu lernen. Entsprechende Verfahren sind daher ansichtsbasiert und werden in der Regel fortlaufend durch neue Beispielansichten trainiert. Das Problem besteht allerdings darin, es dem System zur ermöglichen, eine Ansicht des Objektes korrekt zu erfassen. Die Ausgangssituation sieht normalerweise so aus, dass der Benutzer dafür gesorgt hat, dass sich das Objekt im Sichtbereich des Roboters befindet. Dies kann z.B. dadurch geschehen, dass der Benutzer in die Richtung zeigt, in der sich das Objekt befindet, so dass der Roboter seine Sensoren entsprechend ausrichten kann. Da jedoch kein Wissen über das unbekannte Objekt vorliegt, kann der Roboter es nicht ohne weitere Informationen aus der von ihm wahrgenommenen Szene extrahieren.

Um dieses Problem für einen mobilen Roboter zu lösen, dessen Einsatzgebiet eine Büroumgebung in einem Service-Szenario darstellt, kann ein auf dem Roboter montiertes Touch-Screen-Display verwendet werden [Kri01]. Dazu wird auf dem Display die vom Roboter mit einer Kamera wahrgenommene Szene präsentiert. Die Szene ist hier in Segmente unterteilt, von denen der Benutzer durch Antippen diejenigen Segmente auswählen muss, die zum Objekt gehören. Auch in [Ghi02] ist es das Ziel, einem mobilen Roboter neue Objekte beizubringen, damit sie in einer topologischen Karte gespeichert werden können. Hier wird allerdings nicht versucht, das Objekt exakt aus der Szene zu extrahieren. Stattdessen wird durch eine Zeigegeste vom Benutzer der Bildbereich bestimmt, in dem sich das Objekt befindet. Jede Ansicht eines Objektes wird dabei allerdings vom System als quadratisch angenommen, so dass längliche Objekte nicht angemessen repräsentiert werden können. Die Größe des Objektes wird verbal und qualitativ spezifiziert, z.B. indem der Benutzer Begriffe wie „groß“ und „klein“ verwendet.

Die gerade genannten Verfahren bedienen sich allerdings einer nicht sehr natürlichen Form der Interaktion mit Menschen und erlauben auch nur die Erstellung von relativ groben Modellen unbekannter Objekte. Daher wird für BIRON im Rahmen des Promotionsvorhabens von A. Haasch eine Objektdetektion realisiert, die auf eine direktere Weise entsprechende Ansichten eines Objektes erfassen kann. Für diese Aufgabe soll ein Verfahren entwickelt werden, welches auf so genannten Aufmerksamkeitskarten [Haa03] basiert. Eine Aufmerksamkeitskarte repräsentiert dabei ein spezielles Bildmerkmal, wie z.B. gewisse Strukturinformationen oder bestimmte Farbwerte. In Kombination mit sprachlichen Informationen, welche vom Benutzer gegeben werden, können verschiedene Aufmerksamkeitskarten entsprechend kombiniert werden. Würde der Roboter beispielsweise die Phrase „Das ist meine rote Tasse“ erhalten und dabei nicht wissen, was eine Tasse ist, so könnte er im Optimalfall den Bildbereich, in dem sich das Objekt befindet, anhand des vom Menschen spezifizierten Farbmerkmals lokalisieren.

Ein System zur Extraktion von unbekannten Objekten aus einem Kamerabild, welches vom Roboter aus aufgenommen wurde, ist bereits für BIRON entwickelt worden [Haa03]. Dieses System bedient sich allerdings eines zusätzlichen Laser-Moduls, welches auf der Pan-Tilt-Kamera des Roboters montiert wurde, um die Distanz zu einem von der Kamera fokussierten Punkt mes-

sen zu können. Mit Hilfe der vom Laser-Modul gelieferten Daten kann ein Tiefenprofil erstellt werden, um die Entfernung zu einem Objekt und dessen Ausdehnung zu bestimmen. Durch das Kombinieren dieser Entfernungsdaten mit Informationen aus dem zugehörigen Kamerabild kann schließlich das Objekt aus einer Szene extrahiert werden. Dieses Verfahren ist allerdings bisher nicht im aktuellen Robotersystem integriert worden, da der Vorgang der Objekterfassung aufgrund der zahlreichen Bewegungen des Laser-Moduls bzw. der Kamera zu lange für eine natürliche Interaktion dauert. Aus diesem Grund wird angestrebt, auf die vom Laser-Modul gelieferten Daten zu verzichten. Stattdessen bieten sich die Tiefeninformationen an, die aus den Bildern der nachträglich auf dem Roboter montierten Stereokamera ermittelt werden können.

6.2.2. Das Szenemodell

Das Szenemodell dient schließlich zur Speicherung von Informationen über Objekte, die ein Benutzer dem Roboter beigebracht hat, um sich auf sie in weiteren Interaktionen beziehen zu können. Diese Informationen umfassen Attribute wie Position, Größe und visuelle Informationen von Objekten, die durch die Objektdetektion geliefert werden. Außerdem sind Informationen, die durch den Benutzer gegeben werden, in dem Szenemodell zu speichern. Dabei handelt es sich oft um abstrakte Informationen über das entsprechende Objekt. Beispielsweise können aus der Phrase „Dieses ist meine Lieblingstasse“ Benutzer und Verwendungszweck des referenzierten Objektes extrahiert werden. Insgesamt müssen also in dem Szenemodell, das als eine Art Langzeitgedächtnis des Roboters dient, Informationen unterschiedlicher Modalitäten abgelegt werden. Da diese Daten auch auf unterschiedliche Weise repräsentiert werden müssen, stellt die Verwaltung des Szenemodells, welches als eine multimodale Datenbank betrachtet werden kann, eine große Herausforderung dar. Wichtig ist dabei außerdem, dass das Szenemodell unabhängig von z.B. topologischen Karten für die Navigation des Roboters ist, damit die Modularität des Gesamtsystems erhalten bleibt.

Zur Realisierung der Datenverwaltung im Szenemodell ist die Verwendung eines *Active Memories* [Wre04c] vorgesehen. Dieser Mechanismus basiert zwar auf einer XML-Datenbank, allerdings ist darauf eine spezielle Server-Architektur aufgebaut, die es ermöglicht, nicht nur XML-Daten, sondern auch damit verknüpfte binäre Daten zu verwalten. Die Server-Architektur erlaubt es, dass auf diese heterogenen Daten von verschiedene Modulen des Robotersystems parallel zugegriffen werden kann. Für diese beiden Datenarten stehen entsprechende standardisierte DBMS-Methoden, wie z.B. *insert*, *update*, *remove* und *query* zur Verfügung. Auf spezifische Daten kann außerdem mit XPath-Ausdrücken [Wre04c] zugegriffen werden. An die Wissensrepräsentation wurde ein so genanntes *Subscription*-Modell geknüpft, wodurch eingetragene Prozesse benachrichtigen können, wenn sie an bestimmten Daten oder Speicheraktionen interessiert sind. Zusätzlich enthält das *Active Memory* einen Mechanismus, welcher zu alte oder ungültige Informationen automatisch entfernt und somit eine Art Vergessensprozess repräsentiert. Diese Funktionalität ist für einen *Robot Companion* essentiell, da sich seine Umgebung kontinuierlich ändert. Beispielsweise ist das Wissen über die Position einer Tasse, die einen Monat zuvor erfasst wurde, in der Regel nicht mehr relevant.

Die Funktionalität des *Active Memories* wird für den Einsatz auf BIRON allerdings erweitert, um multimodale Informationen über ein Objekt handhaben zu können. Dazu werden dieselben Informationen, die aus unterschiedlichen Modalitäten erworben wurden, in unterschiedlichen Formaten gespeichert. Auch wenn dieses Vorgehen auf dem ersten Blick redundant erscheint, so ist es erforderlich, was leicht an den folgenden Beispielen ersichtlich wird: Der Benutzer gibt beispielsweise an, dass ein Objekt die Farbe „blau“ besitzt. Die Aufmerksamkeitssteuerung für Objekte verarbeitet Farbwerte jedoch auf Basis des HSV-Farbmodells, so dass hier Farben nur in Form von dreidimensionalen Vektoren repräsentiert sind. Die Aufmerksamkeitssteuerung kann daher die Information „blau“ nicht nutzen, um ein Objekt in einem Kamerabild ausfindig zu machen. Ein ähnliches Problem ergibt sich für die Position eines Objektes: Der Benutzer spezifiziert einen Ort z.B. mit der Phrase „auf dem Tisch“, während die Aufmerksamkeitssteuerung für Objekte hingegen lediglich Raumkoordinaten erfasst. Diese Koordinaten, welche die Position eines Objektes bezüglich eines internen Koordinatensystems beschreiben, sind aber für den Benutzer nutzlos, falls dieser fragt sollte: „Wo ist meine Tasse?“

Um die gerade beschriebene Problematik zu lösen, erhält das Szenemodell eine spezielle Komponente, die Informationen unterschiedlicher Formate ineinander überführen kann. Dieser so genannte Modalitätenkonverter bedient sich einer Nachschlagetabelle, welche drei Spalten besitzt: „Eigenschaft des Objektes“, „symbolische Beschreibung“ und „perzeptuelle Repräsentation“. Unter einer Objekteigenschaft sind Begriffe wie z.B. „Farbe“ oder „Relation“ zu verstehen. Eine symbolischen Beschreibung einer Farbe ist z.B. „blau“. Da die HSV-Werte für „blau“ allerdings variieren können, findet sich unter der perzeptuellen Repräsentation einer Farbe gewöhnlich nicht nur ein einzelner Farbwert, sondern ein Intervall von Farbwerten. Falls nun ein Modul im Gesamtsystem eine Anfrage bezüglich eines Objektes an das Szenemodell stellt, so kann unter Verwendung des Modalitätenkonverters die gewünschte Beschreibung geliefert werden. Diese Komponente kann außerdem nicht nur vordefinierte Zuordnungen nutzen, sondern auch neue Beziehungen zwischen symbolischen Beschreibungen und visuellen Repräsentationen lernen.

6.3. Die Dialogsteuerung

Eine Dialogsteuerung bildet die Grundlage für einen autonomen Roboter, um auf natürliche Weise mit Menschen kommunizieren zu können. Für die Natürlichkeit ist es allerdings wichtig, dass nicht nur sprachliche Eingaben berücksichtigt werden, sondern auch multimodale Informationen verarbeitet werden können. Da Menschen beispielsweise auf Objekte zeigen, die sie in einem Dialog erwähnen, sollte die Dialogsteuerung Informationen über entsprechende Zeigegesten beim Führen eines Dialogs mit einbeziehen können.

Ein Beispiel für einen Roboter, der einen entsprechenden multimodalen Dialog führen kann, ist WABIAN [Has02]. Er kann sprachlich in Form von ganzen Sätzen und durch statische Gesten instruiert werden. Neben dem Erfassen von Instruktionen ist ein Dialog allerdings auch wichtig, um fehlende Informationen zu erfragen. Außerdem können entsprechende Rückfragen gestellt werden, wenn der Roboter offensichtlich etwas falsch verstanden hat. Der in [Per99] vorgestellte

Roboter kann beispielsweise Nachfragen formulieren, falls bestimmte Informationen zum Lösen einer Aufgabe vom Benutzer nicht spezifiziert wurden. Diese Informationen können sowohl sprachlicher Natur sein als auch Gesten darstellen. Einen noch komplexeren Dialog erlaubt der Roboter *HERMES* [Bis99]. Dieser Roboter kann nicht nur nach fehlenden Informationen fragen, sondern auch Mehrdeutigkeiten unter den gegebenen Informationen per Dialog auflösen. Um den Dialog so natürlich wie möglich zu gestalten, wird hier der aktuelle Kontext bzw. die Situation in der sich der Roboter befindet, berücksichtigt. Dadurch kann die Grammatik der Spracherkennungskomponente vereinfacht werden, was zu besseren Ergebnissen in der Erkennung führt. Außerdem ist es dem Benutzer erlaubt, *HERMES* gegenüber Gegenstände und Orte frei zu benennen. Sollte der Roboter einen Begriff nicht kennen, so kann ihm dieses Wort durch Buchstabieren beigebracht werden. Schließlich kann *HERMES* den Benutzer auch über seine Fähigkeiten und sein Wissen informieren [Bis02b]. Im Vergleich zu *HERMES* kann der Roboter Jijo-2 auch einen Dialog mit einem Menschen initiieren, wenn er bestimmte visuelle Informationen wahrnimmt [Mat99a]. Beispielsweise kann Jijo-2 von einer Person über einen unbekannten Flur geführt werden. Sollte der Roboter dabei eine Tür detektieren, so fragt er den Menschen, wessen Büro sich hinter der Tür verbirgt. Dieser Ort kann dann von der Person natürlichsprachlich benannt werden, wie z.B. durch die Phrase „Dies ist das Büro von Person X“. Der Roboter speichert diese Information und kann den Ort später selbstständig wiederfinden, um beispielsweise eine Person dorthin zu führen.

Interessant ist auch die Realisierung des Dialogs zwischen einem Menschen und dem in [Fon01] verwendeten Roboter, der mit einer auf Kooperation beruhenden Kontrolle ausgestattet ist. Auch wenn es sich dabei um eine Art der Teleoperation des Roboters handelt, bei der keine direkte Interaktion zwischen Mensch und Roboter stattfindet, bestehen dennoch gewisse Parallelen zum Dialog im *Home Tour Scenario*. Es wird herausgestellt, dass ein Roboter zwar strukturiertes Planen zuverlässig realisieren kann, sich aber in der Regel weniger gut bei der Erkennung von Objekten und beim Beurteilen von Situationen verhält. Aus diesem Grund verfügt der verwendete Roboter über die Fähigkeit, aktiv Dialoge mit dem Benutzer zu initiieren, um so Unklarheiten, Widersprüche und Mehrdeutigkeiten auf progressive Weise auflösen zu können. Diese Fähigkeit wird durch eine spezielle Dialogkomponente umgesetzt, welche Fragen an den Benutzer formulieren, Informationen aus den erhaltenen Antworten extrahieren und die Qualität dieser Antworten beurteilen kann. Dabei ist die Idee, dass sich der Roboter eher auf die Antworten von einer ihm bekannten Person verlässt, als auf Antworten, die von Personen stammen, welche den Roboter zum ersten Mal bedienen. Eine Evaluation des teleoperierten Roboters hat ergeben, dass es für den Roboter sehr hilfreich ist, wenn er dem Benutzer viele Fragen stellen kann [Fon03b]. Diese Vorgehensweise hilft allerdings auch dem Benutzer, da er so die Probleme des Roboters besser verstehen kann. Dennoch sollte der Mensch vom Robotersystem als limitierte Ressource betrachtet werden, da die Qualität eines Dialogs leidet, wenn der Benutzer mit einer Vielzahl von Fragen bombardiert wird. Aus diesem Grund soll auch BIRON nur Fragen stellen, wenn unbedingt benötigte Informationen fehlen sollten. Dies muss somit von der Dialogsteuerung des Roboters gewährleistet werden.

Das Modell der Dialogsteuerung für BIRON wurde von I. Toptsis entwickelt [Top04b]. Es basiert auf einer Reihe von endlichen Automaten und ist modular aufgebaut. Dazu sind alle Dialoge,

über welche die Dialogsteuerung verfügt, in mehrere Teildialoge untergliedert. Jeder einzelne Teildialog ist schließlich für eine bestimmte Aufgabe verantwortlich und daher durch einen individuellen endlichen Automaten repräsentiert. Diese endlichen Automaten besitzen jeweils einen Zielzustand, welcher die Erfüllung der zugehörigen Aufgabe kennzeichnet. Außerdem sind sie mit der Fähigkeit versehen, rekursiv weitere endliche Automaten zu aktivieren und in jedem Zustand eine Aktion auszuführen. Die Aktionen, die in bestimmten Zuständen ausgeführt werden können, sind in der so genannten *„Dialog Policy“* der Dialogsteuerung spezifiziert. Diese Aktionen beinhalten das Erzeugen von Sprachausgaben und das Senden von Anweisungen und Anfragen in Form von Ereignissen an den Execution Supervisor (siehe Abschnitt 7.2). Die *„Dialog Strategy“* basiert auf der so genannten *„Slot-Filling“*-Methode [Sou00]. Dabei stellt ein *Slot* ein Informationsfeld dar, für das ein Eintrag benötigt wird. Der Status eines solchen *Slots* kann somit entweder „leer“, „mit einem Attribut gefüllt“ oder im Falle eines binären Eintrags „wahr“ oder „falsch“ sein. Für jeden endlichen Automaten existiert ein Satz von *Slots*, die in einem so genannten *„Dialog Frame“* organisiert sind. Jede unterschiedliche Statuskombination der *Slots* in einem *Frame* definiert einen Zustand in dem zugehörigen endlichen Automaten. Die Aufgabe der Dialogsteuerung ist es nun, genügend *Slots* zu füllen, um das aktuelle Dialogziel zu erreichen, welches als ein Zielzustand in dem entsprechenden endlichen Automaten definiert ist. Die *Slots* werden dabei mit Informationen gefüllt, die vom Benutzer und anderen Komponenten des Robotersystems stammen.

Die Dialogsteuerung basiert auf der Verarbeitung von Ereignissen, wobei der Wechsel zwischen den einzelnen Dialogzuständen nicht unter Verwendung eines Transitionsmodells vollzogen wird, sondern von der Statuskomposition aller *Slots* im aktuellen *Dialog Frame* abhängt. Verschiedene Eingabeereignisse, wie z.B. Äußerungen vom Benutzer oder Informationen von anderen Komponenten des Robotersystems, verändern den Status der *Slots*. Während ein Dialog geführt wird, vergleicht die Dialogsteuerung jedes Mal wenn ein *Dialog Frame* aktualisiert wird die enthaltenen *Slots* mit denen im endlichen Automaten, um den neuen Zustand des Modells zu finden. Dabei beruht der Vergleich der einzelnen *Slots* darauf, dass nur deren Status und nicht der in ihnen gespeicherte Inhalt betrachtet wird. Nachdem schließlich der Zustandswechsel im endlichen Automaten erfolgt ist, wird die zugehörige Aktion ausgeführt, die durch die *„Dialog Policy“* bestimmt ist. Anschließend wartet die Dialogsteuerung auf neue Eingaben, die sie einerseits von der Sprachverstehenskomponente erhält, die von S. Hüwel entwickelt wurde [Haa04]. Andererseits können zusätzliche Informationen durch das Stellen von Anfragen an das Szenemodell (siehe Abschnitt 6.2.2) erlangt werden. Allerdings werden sowohl die Anfragen als auch deren Antworten über den Execution Supervisor (siehe Abschnitt 7.2) geleitet. Insgesamt ist die Dialogsteuerung somit in der Lage, multimodale Information zu berücksichtigen.

Die verwendete *„Slot-Filling“*-Technik ist für sich allein genommen allerdings nicht leistungsstark genug, um die komplexen Interaktionsszenarien in der Roboterdomäne ausreichend unterstützen zu können [Lem01]. Das liegt daran, dass die Intentionen des Benutzers in solchen Szenarien selten vorhersagbar sind. Oft wechselt ein Mensch in einem Gespräch den Kontext, um Bezug auf bereits kommunizierte Informationen zu nehmen. Um dieses Problem zu bewältigen, kann die Abarbeitung eines jeden Teildialogs bei Bedarf durch weitere Teildialoge unterbrochen werden, so dass eine alternierende Verarbeitung von Instruktionen entsprechend dem aktuellen

Kontext möglich ist. Die individuellen Dialoge sind unter Verwendung einer deklarativen Definitionssprache spezifiziert und in XML auf modulare Weise kodiert. Dieses Vorgehen erlaubt eine einfache Konfiguration und Erweiterung der definierten Dialoge.

6.4. Zusammenfassung

In diesem Kapitel wurden drei weitere Software-Komponenten vorgestellt, die zur natürlichen Interaktion mit einem Menschen im Rahmen dieser Arbeit in das Gesamtsystem zu integrieren waren. Diese Module sind im Rahmen anderer Promotionsprojekte implementiert worden bzw. werden zum Teil auch in Zukunft noch weiterentwickelt. Bei der zuerst präsentierten Komponente handelt es sich um die Aufmerksamkeitssteuerung für Personen, deren Aufgabe es ist, potentielle Interaktionspartner unter den in der Nähe des Roboters anwesenden Menschen zu detektieren. Ist schließlich ein Interaktionspartner gefunden, so kann die Aufmerksamkeitssteuerung für Personen die Aufmerksamkeit des Roboters auf diese Person fokussieren und ihr dabei ein entsprechendes Feedback vermitteln. Anschließend wurde die Aufmerksamkeitssteuerung für Objekte vorgestellt, die in der Lage ist, Objekte zu fokussieren, auf die der Interaktionspartner verweist. Sie bildet die Grundlage zur visuellen Erfassung von Gegenständen, die der Roboter lernen soll und die somit im Szenemodell des Robotersystems abgelegt werden müssen. Da der Roboter während einer solchen Interaktion in der Lage sein muss, vom Benutzer entsprechende Instruktionen entgegenzunehmen, benötigt er die Fähigkeit zum Führen von Dialogen. Diese Fertigkeit wird durch eine Dialogsteuerung gewährleistet, welche als drittes in diesem Kapitel präsentiert wurde. Die Dialogsteuerung ermöglicht es dem Roboter, mit einem Menschen mittels natürlicher Sprache zu kommunizieren. Dabei wurde insbesondere auf einen modularen Aufbau Wert gelegt, um die Realisierung möglichst flexibler Dialoge zu erlauben. Im Folgenden wird nun die Integration dieser drei Interaktionskomponenten in ein Gesamtsystem, unter Verwendung des in Abschnitt 3.3 vorgestellten Architekturkonzeptes, beschrieben.

7. Die Gesamtarchitektur für BIRON

Nachdem in den vorherigen Kapiteln alle Software-Komponenten für den Roboter BIRON vorgestellt wurden, folgt in diesem Kapitel ihre Integration in einer Gesamtarchitektur. Die Integration wird nur zu oft als ein notwendiger „Zusammenschluss“ verwendeter Komponenten bezeichnet, obwohl eine sorgfältig entwickelte Architektur diese Aufgabe nicht nur sehr erleichtern kann, sondern auch eine flexible, multimodale und somit natürliche Interaktion mit dem System überhaupt erst ermöglicht. Um diese Fähigkeit im Rahmen dieser Arbeit für den Roboter BIRON zu realisieren, wurde das in Abschnitt 3.3 vorgestellte hybride Architekturkonzept umgesetzt. Im Folgenden wird daher zuerst die Architektur von BIRON vorgestellt (Abschnitt 7.1). Anschließend wird die darin verwendete Implementierung des Execution Supervisors beschrieben (Abschnitt 7.2) und schließlich werden die resultierenden Interaktionsfähigkeiten zusammengefasst, die BIRON in Bezug auf das *Home Tour Scenario* besitzt (Abschnitt 7.3).

7.1. Umsetzung des Architekturkonzeptes

In diesem Abschnitt wird beschrieben, wie das in Abschnitt 3.3 vorgestellte hybride Architekturkonzept für *Robot Companions* auf BIRON umgesetzt wurde, um Interaktionen mit Benutzern, wie sie im Rahmen des *Home Tour Scenarios* stattfinden, zu ermöglichen [Kle04, Fri05]. Ein Überblick über die entwickelte Drei-Ebenen-Architektur ist in Abbildung 7.1 gegeben.

Die Module der deliberativen Ebene sind die Dialogsteuerung und die Sprachverstehenskomponente. Ein Planer, der in erster Linie für die Erzeugung von Plänen für Navigationsaufgaben verantwortlich ist, ist ebenfalls auf dieser Ebene vorgesehen. Er kann aber auch um zusätzliche Fähigkeiten zur Planung erweitert werden, welche benötigt werden könnten, um beispielsweise autonome Aktionen ohne die Hilfe eines Menschen auszuführen. Die Integration solcher Fähigkeiten befindet sich allerdings nicht im Fokus dieser Arbeit, da sie sich auf Aktionen stützen, welche erst von Interesse sind, nachdem der Roboter ausreichend instruiert wurde.

Die Dialogsteuerung ist für das Führen von Dialogen zuständig (vgl. Abschnitt 6.3). Das Ziel dieser Dialoge ist es, Instruktionen von einem menschlichen Interaktionspartner zu erfassen. Die Dialogsteuerung ist in der Lage, Interaktionsprobleme zu lösen und Mehrdeutigkeiten zu eliminieren, indem der Benutzer zu Rate gezogen wird. Eingaben erhält die Dialogsteuerung von der Sprachverstehenskomponente, die ebenfalls in der deliberativen Ebene verortet ist, um multimo-

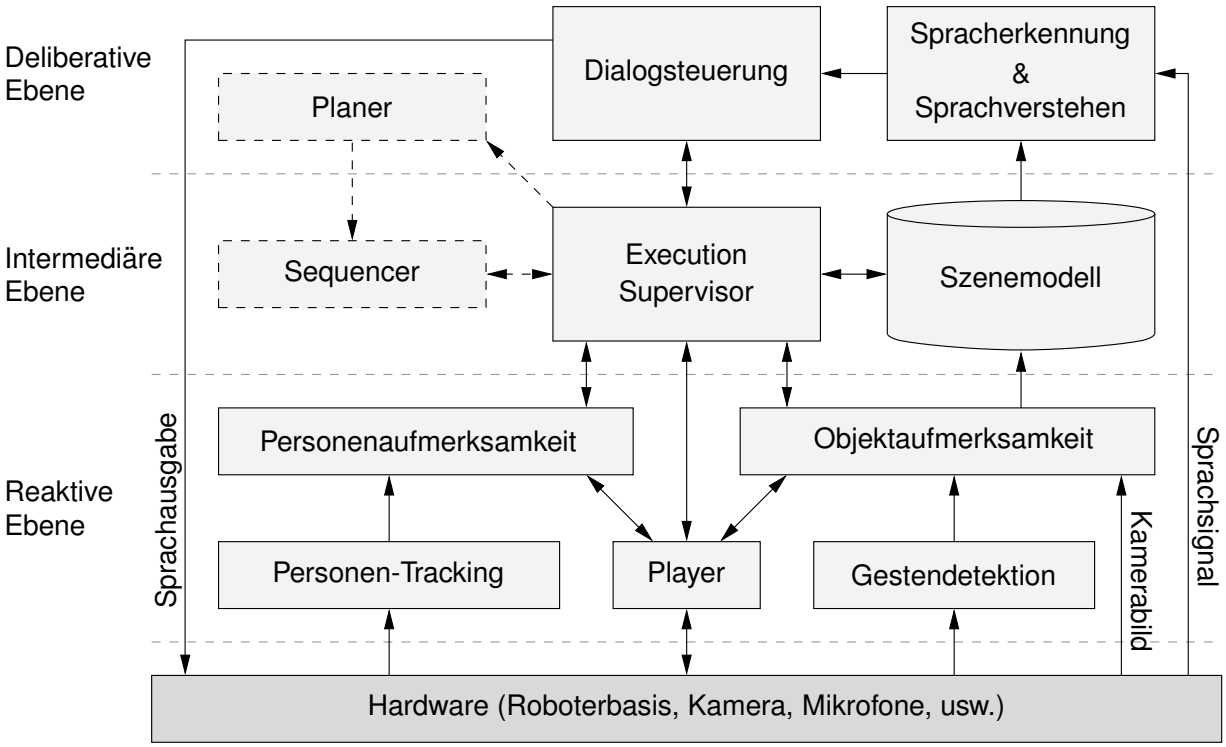


Abbildung 7.1.: Umsetzung des entwickelten Architekturmodells für BIRON.

dale Informationen von Modulen aus den darunter gelegenen Ebenen, wie z.B. vom Szenemodell, verarbeiten zu können. Die Dialogsteuerung sendet gültige Instruktionen an den Execution Supervisor (siehe Abschnitt 7.2), welcher auf der intermediären Ebene der Architektur zu finden ist. Ein *Sequencer* ist ebenfalls in der intermediären Ebene vorgesehen. Da der Execution Supervisor nur einzelne Kommandos verarbeiten kann, wäre dieser *Sequencer* für die Zerlegung von Plänen, die von einem Planer geliefert werden würden, in einzelne Schritte verantwortlich.

Die Aufmerksamkeitssteuerung für Personen befindet sich in der reaktiven Ebene der Architektur. Sie erhält Informationen über Personen, die sich in der Nähe des Roboters befinden, von der Personen-Tracking-Komponente, um potentielle Interaktionspartner für BIRON detektieren zu können (vgl. Abschnitt 6.1). Die Aufmerksamkeitssteuerung wird durch den Execution Supervisor konfiguriert, um verschiedene Verhaltensweisen annehmen zu können. Beispielsweise kann sie dafür sorgen, dass der Roboter entweder alle Personen in seiner Umgebung abwechselnd betrachtet oder einen speziellen Interaktionspartner fokussiert. Die Aufmerksamkeitssteuerung für Personen kontrolliert allerdings nicht direkt die Hardware des Roboters. Diese Aufgabe übernimmt die verwendete *Player/Stage*-Software [Ger03, Vau03]. Dabei stellt die *Player*-Komponente dieser Software eine übersichtliche und einfache Schnittstelle zu den Sensoren und Aktuatoren des Roboters bereit. Obwohl diese Komponente dazu verwendet wird, die Hardware direkt anzusteuern, kann sie leicht durch eine komplexere Hardware-Ansteuerung ersetzt werden, welche z.B. auf der Verwendung von Behaviors basiert.

Eine komplexere Hardware-Ansteuerung wurde auf BIRON auch bereits eingesetzt, allerdings wurde aus ressourcentechnischen Gründen auf eine weitere Verwendung verzichtet. Bei dieser Ansteuerung handelte es sich um die Behavior-basierte Komponente der ISR-Software [And99], wobei die Aufmerksamkeitssteuerung für Personen entschieden hat, welche Behaviors zu gewissen Zeitpunkten aktiviert wurden. Somit konnte beispielsweise bei der Navigation gleichzeitig ein Behavior zum Ausweichen von Hindernissen verwendet werden. Die komplette ISR-Software stellt selbst ein flexibles und skalierbares Navigationssystem dar, welches Aufgaben wie das Ausliefern von Post in einer Büroumgebung übernehmen kann. Die verschiedenen Navigationsfunktionen sind dabei durch Sätze entsprechender Behaviors realisiert. Die Ausgaben der einzelnen Behaviors werden wiederum mittels *Superposition* (vgl. Abschnitt 2.2.4) miteinander kombiniert. Um die Ausführung einer Aufgabe zu kontrollieren, enthält ISR einen *„State Manager“* und einen *„Planner“* und stellt so ein eigenständiges Kontrollsystem dar. Da der interne Zustand von BIRON durch den Execution Supervisor bzw. durch die Aufmerksamkeitssteuerungen verwaltet wird, wurde nur die Behavior-basierte Komponente von ISR in unsere Architektur eingebunden. Die Parametrisierung dieser Komponente sorgte unter Verwendung einer zugehörigen Zuordnungstabelle für die Aktivierung eines entsprechenden Netzwerks an Behaviors innerhalb der ISR-Software.

Parallel zu der Aufmerksamkeitssteuerung für Personen existiert auf der reaktiven Ebene auch die Aufmerksamkeitssteuerung für Objekte. Der Execution Supervisor kann die Kontrolle des Roboters von der Aufmerksamkeitssteuerung für Personen zu der Aufmerksamkeitssteuerung für Objekte verlagern. Um die Rechenkapazitäten des Gesamtsystems zu schonen, können bei diesem Vorgang entsprechende Module zur Bildverarbeitung während der Laufzeit aktiviert bzw. deaktiviert werden (vgl. Konzept in Abschnitt 3.3.1). Die Grundlage dazu bildet das Programm „iceWing“ [Löm04], welches alle bildverarbeitenden Prozesse im Gesamtsystem koordiniert und es ermöglicht, sie von außen zu kontrollieren. Wenn die Aufmerksamkeitssteuerung für Objekte schließlich die Kontrolle über die Hardware erhalten hat, so ist sie in der Lage, Objekte zu fokussieren, die durch den Benutzer referenziert werden (vgl. Abschnitt 6.2). Die Aufmerksamkeitssteuerung für Objekte wird dabei durch die Komponente zur Detektion von Zeigegesten unterstützt. Die Kombination einer sprachlichen Instruktion und einer Zeigegeste erlaubt es, visuelle Informationen eines referenzierten Objektes zu akquirieren. Diese Informationen werden letztendlich von der Aufmerksamkeitssteuerung für Objekte an das Szenemodell in der intermediären Ebene der Architektur geschickt.

Das Szenemodell speichert Informationen über Objekte, die dem Roboter für spätere Interaktionen vom Benutzer beigebracht werden (vgl. Abschnitt 6.2.2). Diese Informationen enthalten Attribute wie Position, Größe und visuelle Daten über das Objekt, welche von der Aufmerksamkeitssteuerung für Objekte geliefert werden. Außerdem werden zusätzliche Informationen, die durch den Benutzer gegeben sind, in dem Szenemodell gespeichert. Beispielsweise gibt eine Aussage wie „Dieses ist meine Kaffeetasse“ Aufschluss über den Benutzer und die Verwendung des gelernten Objektes. Anfragen an das im Szenemodell gespeicherte Wissen werden in der Regel durch die Dialogsteuerung initiiert und durch den Execution Supervisor an diese Komponente weitergeleitet. Entsprechende Antworten gelangen schließlich über den gleichen Weg, aber in umgekehrter Richtung zurück an die Dialogsteuerung.

Die Kommunikation zwischen allen oben genannten Modulen basiert gemäß dem verwendeten Architekturkonzept auf der Verwendung von Nachrichten, die in XML kodiert sind. Ursprünglich wurde für den Austausch dieser Nachrichten das Kommunikationssystem DACS [Fin95] verwendet. Dieses ist jedoch nicht auf die Übertragung von XML-Nachrichten ausgerichtet, so dass es daher gegen XCF, dem *XML-enabled Communication Framework* [Wre04b], ausgewechselt wurde. Im Gegensatz zu DACS ist XCF nämlich auch direkt in der Lage, die XML-Strukturen zu transportierender Nachrichten zu validieren, insofern entsprechende XML-Schemata für sie spezifiziert wurden (vgl. Konzept in Abschnitt 3.3.2).

Neben der Verwendung von XML zur Repräsentation von Wissen und Schemata zur Spezifikation von Datentypen wurde die *Internet Communication Engine* (ICE) als technische Basis zur Realisierung der kommunikativen Aufgaben in XCF ausgewählt. ICE bietet eine ähnliche Funktionalität wie CORBA an, stellt aber einen erheblich leichteren Ansatz dar. Infolgedessen bildet XCF eine einfach zu verwendende *Middleware*, die es in einer verteilten Architektur auf effiziente Weise ermöglicht, XML-Dokumente und darin referenzierte Binärdaten, wie z.B. Bilder, auszutauschen. Dazu werden die Binärdatenstrukturen allerdings nicht mit in der XML-Nachricht integriert, sondern als binäre Daten parallel verschickt. Diese Lösung kombiniert somit die Flexibilität von XML mit der Effizienz von systemnahen Kommunikationssemantiken, die sich für große Mengen an Binärdaten eignen. Der XCF-Kern selbst ist auf so genannten Entwurfsmustern aufgebaut und bietet diverse Kommunikationssemantiken, unter denen sich sowohl *Publisher/Subscriber*-Mechanismen, synchrone und asynchrone Funktionsaufrufe als auch *Event Channels* befinden. Zusätzlich können alle XCF-Objekte und bereitgestellten Methoden dynamisch zu Laufzeit im System registriert werden. Da alle transportierten Datentypen durch XML-Schemata spezifizierbar sind, kann auch die Laufzeitsicherheit eines Systems garantiert werden. Außerdem wird eine Selbstprüfung des Systems direkt unterstützt, da das so genannte *Interceptor Pattern* [Sch00] in XCF umgesetzt wurde. Dieser Mechanismus unterstützt insbesondere die Fehlerbereinigung und die Überwachung eines laufenden verteilten Systems.

Insgesamt stellt XCF eine einfach zu verwendende und transparente Basis für verteiltes Rechnen auf asynchrone und entkoppelte Weise dar. Von dieser Fähigkeit profitiert insbesondere der Execution Supervisor, da ihm nicht bekannt sein muss, auf welchem Rechner des Roboters sich ein gewisses Modul befindet. Er adressiert bestimmte Daten einfach mit dem Namen des Zielmoduls, welche dann über XCF an das entsprechende Komponente, die sich im System unter diesem Namen angemeldet hat, transportiert werden. Auf welche Weise schließlich Nachrichten, die an den Execution Supervisor gerichtet sind, von ihm verarbeitet werden, um daraufhin Daten zur Konfiguration einzelner Module im System zu versenden, wird im Folgenden beschriebenen.

7.2. Realisierung der Execution Supervisors

Der Execution Supervisor, als das zentrale Modul der Architektur, ist so konzipiert, dass es so generisch wie möglich ist (vgl. Abschnitt 3.3.1). Um diese Anforderung zu erfüllen, interpretiert er, ähnlich wie das zentrale Kontrollmodul in TCA [Sim94], keinerlei Daten. Stattdessen

konfiguriert der Execution Supervisor entweder andere Module des Systems zur Laufzeit, basierend auf empfangenen Nachrichten, welche dazu notwendige Parameter enthalten, oder er leitet spezifische Daten zu den Komponenten, die für deren Verarbeitung verantwortlich sind. Da alle Informationen in Form von XML-Dokumenten transportiert werden, braucht der Execution Supervisor nicht einmal zwischen den verschiedenen Datenstrukturen zu unterscheiden, die er empfängt. Zusätzlich können verschiedene Daten zwischengespeichert und kombiniert werden, bevor sie an den entsprechenden Empfänger weitergeleitet werden.

Die Koordinierung zwischen dem Execution Supervisor und allen an ihn angebundenen Komponenten ist dabei ähnlich wie bei dem Roboter ISAC [Kaw00b]. Dessen hybride Architektur integriert neben so genannten *„Low-level Agents“* einen *„Human Agent“* und einen *„Self Agent“*, welche die höher stehenden Agenten im System darstellen. Der *Human Agent* ist für die Wahrnehmung von Personen verantwortlich, während der *Self Agent* den internen Zustand und die kognitiven Fähigkeiten des Systems verwaltet. Zur Koordinierung des Gesamtsystems verwenden alle Agenten endliche Automaten, die miteinander synchronisiert werden. Dabei übernehmen die höher stehenden Agenten die Sequenzierung für die Aktivierung und Deaktivierung der *Low-level Agents* [Pet99].

Im Gegensatz zu den bei ISAC verwendeten endlichen Automaten, die in jedem Zustand eine Aktion initiieren können, wird der Execution Supervisor durch einen „erweiterten endlichen Automaten“ (EEA) gesteuert, der Aktionen auslöst, sobald eine Transition ausgeführt wird. Da der Execution Supervisor nicht dafür ausgelegt ist, Daten zu interpretieren, ist der EEA nicht sehr komplex. Aus diesem Grund ist die Struktur des EEA ebenfalls in XML spezifiziert. Somit bedarf es keiner speziellen Programmiersprache, wie z.B. RAPs [Fir89], um den Execution Supervisor zu programmieren. Dieser Ansatz ist effizient, da die entsprechende Konfigurationsdatei nur einmal zum Zeitpunkt des Systemstarts geladen werden muss. Außerdem ergibt sich für den EEA durch dessen Definition in XML eine übersichtliche Repräsentation. Diese erlaubt es, den Execution Supervisor schnell zu restrukturieren und zu erweitern, ohne dass er neu kompiliert werden muss. Dieses Konzept erlaubt es ebenfalls, den Execution Supervisor zur Laufzeit zu modifizieren, um das Lernen neuer Modulinteraktionen über die Zeit zu ermöglichen. Da neue XML-Dokumente direkt verarbeitet werden können, ist deren Integration ohne Weiteres möglich.

Die Struktur zur Spezifizierung einer Konfigurationsdatei für den Execution Supervisors ist in Abbildung 7.2 in Form eines UML-Klassendiagramms zu sehen. In dieser Datei werden die Namen aller im System vorhandenen „Module“, die mit dem Execution Supervisor verbunden sind, spezifiziert. Außerdem sind die Namen der „Datentypen“ anzugeben, die zwischen dem Execution Supervisor und den spezifizierten Modulen ausgetauscht werden. Sowohl die Namen der Module als auch der Datentypen bilden Bestandteile der im System versendeten „Ereignisse“, „Kommandos“ und „Statusmeldungen“. Während Ereignisse an den Execution Supervisor gerichtet sind, werden Kommandos und Statusmeldungen von ihm an die anderen Module versendet (vgl. Abschnitt 3.3.2). Da zur Rekonfiguration des Systems eventuell mehrere spezifische Nachrichten zur gleichen Zeit an verschiedene Module verschickt werden müssen, sind entsprechende Kommandos und Statusmeldungen zu so genannten „Operationen“ zusammengefasst. Diese Operationen werden wiederum durch „Transitionen“ des im Execution Su-

pervisor vorhanden EEA ausgelöst. Eine Transitionen wird ausgeführt, wenn ein entsprechendes Ereignisse beim Execution Supervisor eintrifft. Außerdem benötigt der EEA „Zustände“, die zur eventuellen Synchronisation von Ereignissen wiederum zu so genannten „Kategorien“ zusammengefasst sind.

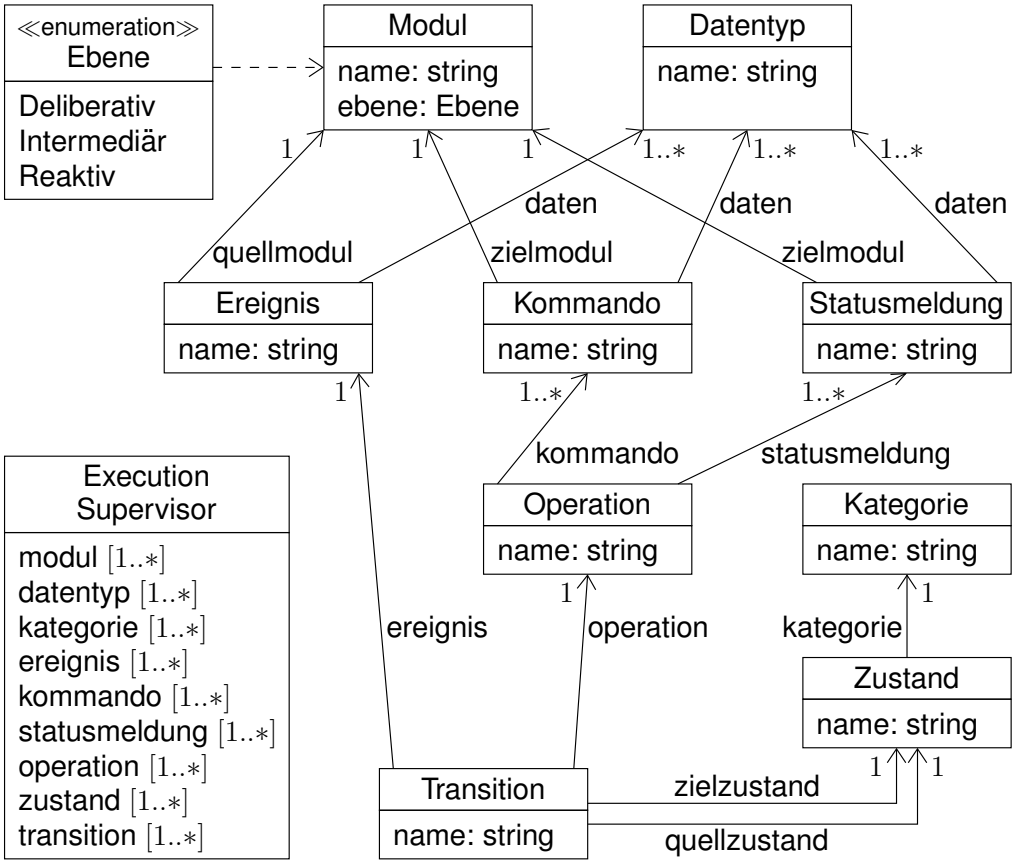


Abbildung 7.2.: UML-Klassendiagramm der internen Struktur des Execution Supervisors.

Die Erweiterung des Execution Supervisors ist somit auch einfach möglich, da dazu nur die zugehörige Konfigurationsdatei angepasst werden muss. Wenn z.B. ein neues Modul zu integrieren ist, so wird es lediglich durch Angabe eines Namens spezifiziert. Dieser Name wird dann vom Execution Supervisor verwendet, um die für ihn benötigten Kommunikationskanäle zu der neuen Komponente automatisch aufzubauen. Falls dieses Modul zusätzliche Funktionalitäten in das System einbringt, so sind eventuell auch neue Zustände für den Execution Supervisor zu definieren. Des Weiteren müssen nur noch die Nachrichten spezifiziert werden, die das hinzugefügte Modul einerseits an die zentrale Komponenten sendet und andererseits von ihr erwartet. Letztere Nachrichten werden gewöhnlich durch bereits bestehende Komponenten des Systems initiiert, so dass zur Einbindung des neuen Moduls natürlich auch an deren Kommunikationsverhalten Anpassungen vorzunehmen sind. Die eigentliche Funktionsweise des Execution Supervisors zur Steuerung aller auf diese Weise spezifizierten Module und die letztendlich durch ihn realisierte Verhaltenssteuerung des Roboters werden in den folgenden zwei Abschnitten beschrieben.

7.2.1. Funktionsweise des Execution Supervisors

Das Kernelement des Execution Supervisors stellt der bereits oben genannte EEA dar. Wie ebenfalls zuvor angedeutet, ist dieser endliche Automat insofern erweitert, dass er zusammen mit jeder Transition eine spezielle Operation ausführt (vgl. Abb. 7.2). Diese Operationen sorgen für die Konfiguration des Gesamtsystems, indem zwei spezielle Arten von Nachrichten versendet werden: Kommandos und Statusmeldungen. Im vorgestellten System werden Kommandos durch den Execution Supervisor zum Szenemodell, zu einem potentiell verfügbaren *Sequencer*, zu beiden Aufmerksamkeitssteuerungen und zur *Player*-Software versendet, während Statusmeldungen an die Dialogsteuerung und an einen eventuell vorhandenen Planer gerichtet sind (vgl. Abb. 7.1).

Statusmeldungen informieren die Dialogsteuerung über den internen Status des Robotersystems. Dazu gehören z.B. Informationen über den aktuellen Interaktionspartner. Auf gleiche Weise würde der Planer eine Navigationsaufgabe in Form einer Statusmeldung erhalten, um einen entsprechenden Plan zu generieren. Im Gegensatz dazu enthalten Kommandos, welche an die Aufmerksamkeitssteuerungen gesendet werden, gewisse Konfigurationsdaten, um die Verhaltensweisen dieser Module zu modifizieren. Die *Player*-Software erhält hingegen Kommandos zur direkten Kontrolle der Hardware. Kommandos, die an das Szenemodell gesendet werden, veranlassen es entweder dazu, mitgelieferte Daten zu speichern, oder bereits gespeicherte Daten bereitzustellen. Im letzteren Fall erzeugt das Szenemodell eine Liste von Objekten, die konform mit der entsprechenden Anfrage sind. Anschließend sendet das Szenemodell diese Liste zurück an den Execution Supervisor, der sie dann an das anfragende Modul weiterleitet. Falls ein Plan ausgeführt werden sollte, so würde der *Sequencer* solange Anfragen zu neuen Planschritten erhalten, bis die entsprechende Aufgabe erfüllt ist.

Neben dem Versenden von Kommandos und Statusmeldungen erhält der Execution Supervisor ebenfalls Daten in Form von Ereignissen von allen Modulen, die mit ihm verbunden sind. Diese Ereignisse initiieren schließlich das Verschicken von Kommandos und Statusmeldungen. Da der Execution Supervisor die Ereignisse auf asynchrone Weise von den anderen Modulen empfängt, enthält er neben dem EEA außerdem eine so genannte „Ereigniswarteschlange“. Alle eintreffenden Ereignisse enthalten einen Zeitstempel, an dem ersichtlich ist, wann das jeweilige Ereignis erzeugt wurde. Alle Ereignisse werden in die Warteschlange sortiert eingefügt, wobei die Sortierung anhand der Zeitstempel der Ereignisse vorgenommen wird. Die Ereignisse in der Warteschlange werden der Reihe nach abgearbeitet, wobei mit dem ältesten Eintrag begonnen wird. Jedes Ereignis entspricht einer Transition in dem EEA. Somit werden Transitionen, die von einem Zustand in einen anderen führen, durch Ereignisse ausgelöst. Wenn eine Transition ausgeführt wird, so wird das zugehörige Ereignis aus der Warteschlange gelöscht.

Die Ereigniswarteschlange wird auch dazu verwendet, um gewisse Ereignisse zu synchronisieren. Beispielsweise kann eine Person nur der aktuelle Interaktionspartner des Roboters werden, wenn einerseits die Aufmerksamkeitssteuerung für Personen signalisiert, dass sie einen potentiellen Interaktionspartner detektiert hat, und andererseits die Dialogsteuerung dem Execution Supervisor mitteilt, dass sie eine entsprechende Spracheingabe erhalten hat. Nur wenn diese beiden Ereignisse in einem bestimmten Zeitintervall bei dem Execution Supervisor eintreffen,

wird angenommen, dass diese Ereignisse zusammengehören. Folglich wechselt der Execution Supervisor in einen anderen Zustand, wobei die Aufmerksamkeitssteuerung für Personen dazu veranlasst wird, sich auf die entsprechende Person zu fokussieren.

Eine Möglichkeit, Ereignisse zu synchronisieren, wäre dadurch gegeben, wenn sie nur in Kombination eine Transition auslösen könnten. Dennoch sind alle Transitionen in dem EEA des Execution Supervisors nur mit einzelnen Ereignissen beschriftet, wodurch folglich alle Ereignisse sequentiell behandelt werden. Diese Vorgehensweise wurde gewählt, da es notwendig sein kann, bestimmte Ereignisse unverzüglich zu verarbeiten, obwohl die verbleibenden, zur Synchronisation benötigten Ereignisse noch nicht empfangen wurden. Um an das obige Beispiel anzuknüpfen: Das System muss zur korrekten Konfiguration der Aufmerksamkeitssteuerung für Personen unterscheiden können, ob ein potentieller Interaktionspartner detektiert wurde oder nicht, und zwar unabhängig davon, ob die Person den Roboter begrüßt. Um Ereignisse in der Ereigniswarteschlange speichern zu können, bis andere Ereignisse eintreffen, die zuerst abgearbeitet werden müssen, sind einige Zustände des EEA zu Kategorien zusammengefasst.

Neben Synchronisationsaufgaben muss der Execution Supervisor auch Latenzen zwischen eintreffenden Ereignissen berücksichtigen, da alle Module im Gesamtsystem asynchron zueinander betrieben werden. Dies ist durch einen Lebensdauereintrag in den entsprechenden Ereignissen realisiert, welcher beschreibt, wie lange ein Ereignis gültig ist. Somit kann es vorkommen, dass gewisse Ereignisse nicht mehr verarbeitet werden. Insgesamt gibt es zwei Fälle, in denen Ereignisse eine solche Ausnahme verursachen. Ein Ereignis wird nicht behandelt, falls:

1) es veraltet ist, was an dem Lebensdauereintrag ersichtlich ist. In diesem Fall wird das Ereignis aus der Ereigniswarteschlange gelöscht und verworfen. Falls das Ereignis von einem Modul aus der deliberativen Ebene stammt, so wird es an den Sender zurückgeschickt, inklusive dem Grund für die Zurückweisung. Dies erlaubt es beispielsweise der Dialogsteuerung, das aufgetretene Problem dem Benutzer gegenüber mitzuteilen.

2) es nicht auf den aktuellen Zustand des EEA angewendet werden kann. Falls die entsprechende Transition zu der Zustandskategorie gehört, in der sich der EEA gerade befindet, so wird dieses Ereignis übersprungen und es wird das folgende Ereignis behandelt. Das ausgelassene Ereignis verbleibt solange in der Ereigniswarteschlange, bis es entweder auf einen Zustand der aktuellen Kategorie angewendet werden kann oder es veraltet ist, wonach wiederum Fall 1 vorliegt. In allen anderen Fällen wird das Ereignis aus der Warteschlange gelöscht und verworfen bzw. an den Sender zurückgeleitet.

Der oben beschriebene Mechanismus stellt sicher, dass bestimmte Ereignisse von verschiedenen Modulen innerhalb eines bestimmten Zeitintervalls bei dem Execution Supervisor eintreffen müssen, bevor dieser in einen bestimmten Zustand wechseln kann. In dieser Vorgehensweise spiegelt sich wider, dass der Execution Supervisor die Kontrolle über das Gesamtsystem hat. Die Dialogsteuerung kann zwar Anweisungen geben, aber falls die Komponenten der übrigen Ebenen nicht entsprechende Informationen liefern, so werden die Anweisungen zurückgewiesen. Beispielsweise könnte der Roboter Instruktionen von einem Radio erhalten, würde aber keine

Interaktion mit dem Gerät beginnen, da kein potentieller Interaktionspartner von der Aufmerksamkeitssteuerung für Personen detektiert werden kann. Somit sorgt die Synchronisation von Ereignissen, die von Modulen aus verschiedenen Ebenen der Architektur stammen können, für mehr Robustheit im Gesamtsystem.

Eine Visualisierung des EEA, wie er in der aktuellen Version des Execution Supervisors von BIRON verwendet wird, ist in Abbildung 7.3 zu sehen. Der graue Balken links oben deutet eine Zustandskategorie an. Beschriftungen in Anführungszeichen bezeichnen Transitionen, die ausgeführt werden, wenn Ereignisse verarbeitet werden, die von der Dialogsteuerung stammen. Im Gegensatz dazu beziehen sich kursive Beschriftungen auf Ereignisse, die von der Aufmerksamkeitssteuerung für Personen empfangen werden. Die verbleibenden, geklammerten Beschriftungen verweisen auf die übrigen Module, die mit dem Execution Supervisor verbunden sind. Wie dieser EEA nun für die Steuerung des Verhaltens von BIRON sorgt, wird im folgenden Abschnitt beschrieben.

7.2.2. Resultierende Verhaltenssteuerung

Die zuvor in den Kapiteln 5 und 6 beschriebenen Interaktionskomponenten bilden die Grundlage verschiedener Verhaltensweisen für den Roboter BIRON. Gewünschte Wechsel zwischen diesen Verhaltensweisen werden durch Zustandswechsel des EEA des Execution Supervisors erreicht. Initial startet der EEA in dem Zustand *«Beobachte (KeinKP)»* (siehe Abb. 7.3). Gleichzeitig wird bei dessen Start die Aufmerksamkeitssteuerung für Personen so konfiguriert, dass alle vor dem Roboter anwesenden Personen abwechselnd betrachtet werden. Dieser Zustand wird auch immer dann eingenommen, wenn keine Person mehr detektiert werden kann, die mit dem Roboter interagieren möchte. Sollte die Aufmerksamkeitssteuerung für Personen einen potentiellen Interaktionspartner finden, so wechselt der EEA in den Zustand *«Beobachte (pot. KP)»*. Entsprechend Abschnitt 6.1 stellt eine Person einen potentiellen Interaktionspartner dar, wenn sie sich dem Roboter zuwendet und gleichzeitig spricht. In dem Zustand *«Beobachte (pot. KP)»* fokussiert der Roboter diese Person solange, wie sich keine interessantere Person findet. Sollte beispielsweise der aktuelle potentielle Interaktionspartner aufhören zu sprechen, so würde er vorerst noch der potentielle Interaktionspartner bleiben. Wenn aber anschließend eine andere Person damit beginnen würde, in die Richtung des Roboters zu sprechen, so würde die Aufmerksamkeitssteuerung für Personen den potentiellen Interaktionspartner wechseln. Würde ein (potentieller) Interaktionspartner identifiziert werden, was im Prinzip in jedem Zustand des EEA möglich ist, so würde das zwar nichts an der Konfiguration des Gesamtsystems ändern, aber die Dialogsteuerung würde über den Namen dieser Person informiert werden.

Sollte der aktuelle potentielle Interaktionspartner nun dem Roboter seine Absicht zur Interaktion mitteilen, indem er z.B. den Roboter entsprechend begrüßt, so erhält der Execution Supervisor diese Information von der Dialogsteuerung. Folglich wechselt der EEA in den Zustand *«Fokussiere Person»* und der Roboter akzeptiert den potentiellen Interaktionspartner fortan als tatsächlichen Interaktionspartner. Wenn sich der EEA also in diesem Zustand befindet, kann der Roboter nicht mehr durch andere Personen abgelenkt werden. Eine Person bleibt solange der Interakti-

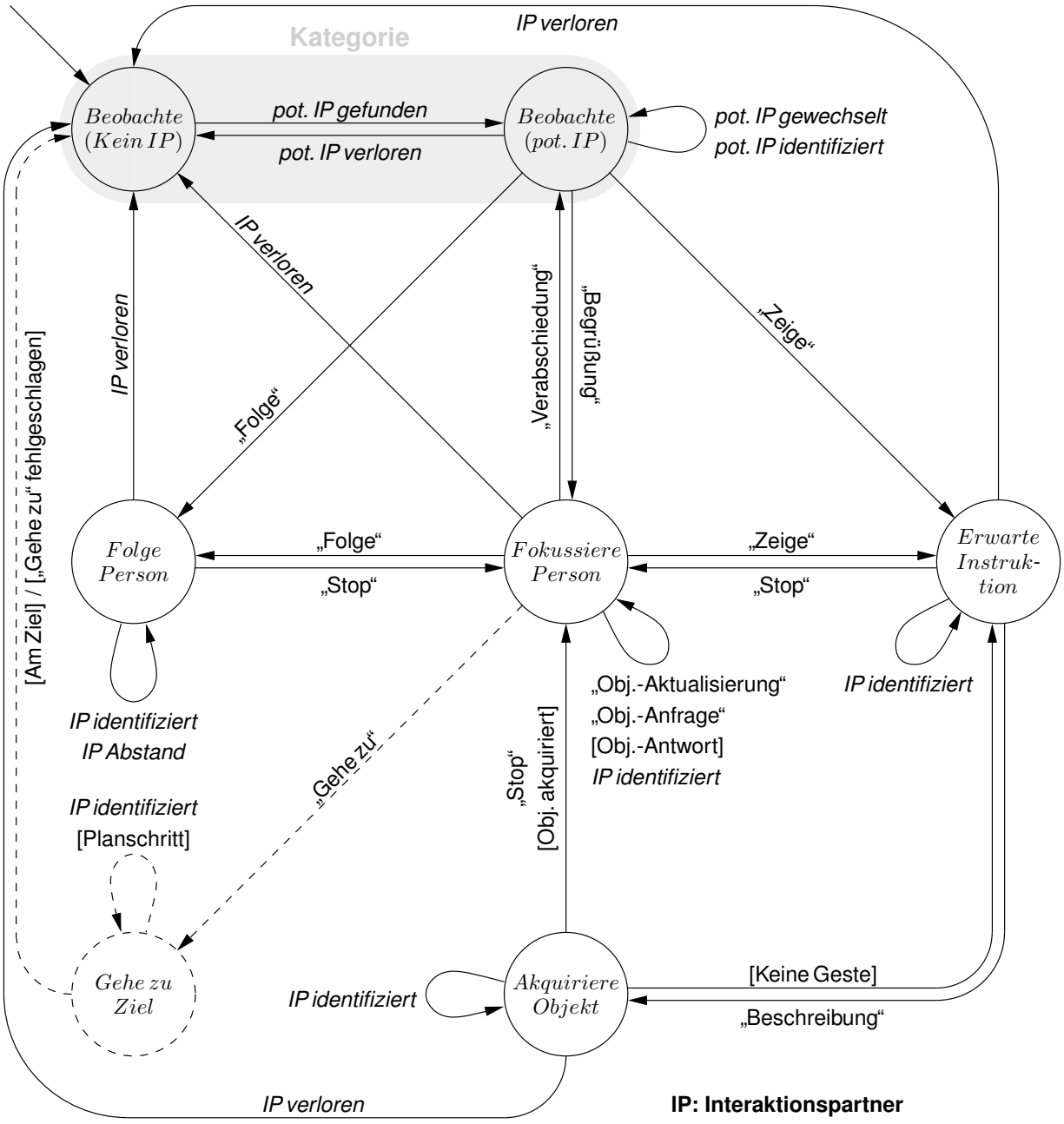


Abbildung 7.3.: Der erweiterte endliche Automat des Execution Supervisors.

onspartner, bis sie sich vom Roboter entsprechend verabschiedet oder sich aus dem Sichtbereich des Roboters bewegt. Dadurch würde der EEA wieder in den Zustand *«Beobachte (pot. KP)»* bzw. in den Zustand *«Beobachte (KeinKP)»* gelangen.

Wenn sich der EEA im Zustand *«Fokussiere Person»* befindet, so kann der Roboter auf mehrere sprachliche Instruktionen des Interaktionspartners reagieren. Dabei handelt es sich entweder

um Anweisungen, die das Wissen des Systems betreffen, oder um Aktionsanweisungen. Folglich beziehen sich erstere Anweisungen auf das Szenemodell und führen nicht zu einer Änderung in der Konfiguration des EEA bzw. des Gesamtsystems. Das Szenemodell erhält dabei vom Execution Supervisor entsprechende Anfragen über ein Objekt oder Zusatzinformationen zu einem Objekt, welches bereits gespeichert wurde. Antworten vom Szenemodell werden wiederum über den Execution Supervisor zur Dialogsteuerung propagiert, ohne dass dabei der Zustand des EEA beeinflusst wird. Bei den Aktionen, zu denen der Roboter angewiesen werden kann, handelt es sich um das Folgen einer Person, das Lernen eines neuen Objektes und das Erreichen eines bestimmten Ortes.

Gibt der Interaktionspartner dem Roboter die Anweisung ihm zu folgen, während der EEA im Zustand *«Fokussiere Person»* ist, so wechselt der EEA in den Zustand *«Folge Person»*. Die Aufmerksamkeitssteuerung für Personen wird dabei so konfiguriert, dass sie versucht, den Abstand zwischen dem Roboter und der Person konstant zu halten. Die Aufmerksamkeitssteuerung sendet dabei regelmäßig den aktuellen Abstand zwischen Roboter und Interaktionspartner an den Execution Supervisor. Diese Information wird an die Dialogsteuerung weitergeleitet, um den Benutzter informieren zu können, falls sich dieser zu schnell bewegen sollte. Auch bei dieser Weiterleitung ändert sich der Zustand des EEA nicht. Der Roboter folgt dem Interaktionspartner schließlich solange, bis dieser den Roboter anweist, dieses Verhalten zu beenden. Danach erwartet der Roboter im Zustand *«Fokussiere Person»* weitere Anweisungen.

Teilt der Benutzer dem Roboter im EEA-Zustand *«Fokussiere Person»* mit, dass er ihm ein neues Objekt zeigen möchte, so wird vom EEA der Zustand *«Erwarte Instruktion»* eingenommen. In diesem Fall wird die Kontrolle über den Roboter von der Aufmerksamkeitssteuerung für Personen an die Aufmerksamkeitssteuerung für Objekte übergeben. Letztere versucht nun, eine Zeigegeste des Interaktionspartners zu detektieren, um das von der Person referenzierte Objekt zu erfassen. Dazu wird die Kamera des Roboters und eventuell auch dessen Basis bewegt. Sollte keine Geste vom System erkannt werden, so wird dem Interaktionspartner dieses mitgeteilt. Daraufhin kann der Benutzer einen weiteren Versuch unternehmen oder den Vorgang abbrechen, um beispielsweise den Roboter in eine bessere Position zu führen. Verläuft die Datenerfassung hingegen erfolgreich, so werden die entsprechenden Objektdaten im Szenemodell gespeichert und der EEA gelangt erneut in den Zustand *«Fokussiere Person»*.

Für den Fall, dass entsprechende Navigationsfähigkeiten im Robotersystem integriert werden, ist im EEA der Zustand *«Gehe zu Ziel»* vorgesehen. Würde der Roboter im Zustand *«Fokussiere Person»* dann vom Interaktionspartner die Anweisung bekommen, sich an einen bekannten Ort zu begeben, so würde der EEA in den Zustand *«Gehe zu Ziel»* wechseln. Im Unterschied zu den zwei zuvor beschriebenen Aktionen gibt es bei dieser Aktion keine Interaktion mit dem Benutzer. Sie stellt eine autonom auszuführende Handlung dar, bei der der Interaktionspartner in der Regel verloren geht. Diese Aktion benötigt einen Plan zur Navigation, der von einem Planer zu liefern ist. Die zugehörigen Planschritte würden schließlich vom Execution Supervisor über einen zugehörigen *Sequencer* empfangenen und an die Roboteransteuerung weitergeleitet werden. Sollte der Roboter sein Ziel schließlich erreichen, so würde vom EEA der Zustand *«Beobachte (KeinKP)»* eingenommen werden, um erneut potentielle Interaktionspartner erfassen zu

können. In diesen Zustand würde der EEA auch dann wechseln, wenn der Roboter sein Ziel nicht erreichen sollte. In diesem Fall könnte das System z.B. eine detektierte Person um Hilfe bitten, da der Dialog über das Fehlschlagen der Aktion informiert werden würde.

7.3. Interaktionsfähigkeiten des Systems

Im Folgenden werden die Interaktionsfähigkeiten von BIRON, über die der Roboter in der aktuellen Implementierung verfügt, in Bezug auf das *Home Tour Scenario* dargestellt. Zu Beginn beobachtet BIRON seine Umgebung. Wenn in dieser Phase Personen in der Nähe des Roboters anwesend sind, so versucht er seine Basis so auszurichten, dass er möglichst alle Personen mit seinen Sensoren betrachten kann. Außerdem fokussiert er die interessanteste Person unter ihnen (vgl. Abschnitt 6.1). Eine Person ist für BIRON interessant, wenn sie zum Roboter schaut oder spricht. Dabei bevorzugt der Roboter eine Person, die gleichzeitig spricht während sie ihn anschaut und somit einen potentiellen Interaktionspartner darstellt. Der Roboter fokussiert eine Person, indem er seine Kamera auf sie ausrichtet. Des Weiteren werden die Augen des auf dem Touch-Screen-Display präsentierten Gesichts auf diese Person gerichtet, um Blickkontakt herzustellen. Ein potentieller Interaktionspartner kann eine Interaktion mit dem Roboter beginnen, indem er ihn z.B. mit „Hallo BIRON“ begrüßt. Damit wird die Person zum Interaktionspartner des Roboters. Infolgedessen konzentriert sich BIRON nur noch auf diese Person, d.h. er lässt sich nicht mehr durch andere Personen vom Interaktionspartner ablenken.

In dieser Phase richtet BIRON neben der Kamera auch seine Basis auf den Interaktionspartner aus, um bei der Aufnahme von Sprache die Qualität des Signals zu erhöhen. Danach kann der Benutzer den Roboter bitten, ihm an einen anderen Ort zu folgen, um dem Roboter neue Objekte beizubringen. Während BIRON der Person folgt, versucht der Roboter immer einen konstanten Abstand zu ihr einzuhalten. Sollte sie sich zu weit entfernen, so kann der Roboter die Person darauf hinweisen, dass sie sich zu schnell bewegt. Wenn BIRON die gewünschte Position erreicht hat, kann der Interaktionspartner ihn bitten, den Vorgang des Folgens zu beenden. Dann kann der Benutzer den Roboter dazu anweisen, ein neues Objekt zu lernen. In diesem Fall fokussiert der Roboter mit der Kamera den Oberkörper der Person, um ihre Hände in sein Blickfeld zu bekommen. Nachdem der Interaktionspartner eine Zeigegeste ausgeführt und eine sprachliche Äußerung, wie z.B. „Dieses ist meine Lieblingstasse“, gegeben hat, fokussiert die Aufmerksamkeitssteuerung für Objekte den referenzierten Gegenstand. Wenn dann dieses Objekt detektiert worden ist, werden alle akquirierten Informationen in dem Szenemodell des Systems gespeichert. Anschließend kann der Benutzer dem Roboter auf gleiche Weise weitere Objekte beibringen. Anfragen an das gesammelte Wissen von BIRON können in der aktuellen Implementierung allerdings noch nicht gestellt werden, da der entsprechende Anfragemechanismus für das Szenemodell (vgl. Abschnitt 6.2.2) noch nicht umgesetzt wurde. Wenn sich der Interaktionspartner hingegen mit „Tschüss BIRON“ vom Roboter verabschiedet oder sich einfach entfernt, während ihm der Roboter nicht folgt, nimmt BIRON an, dass die Interaktion beendet ist und sucht nach neuen potentiellen Interaktionspartnern.

7.4. Zusammenfassung

In diesem Kapitel wurde die Umsetzung des in Abschnitt 3.3 entwickelten Architekturkonzeptes einer hybriden Architektur für *Robot Companions* auf dem Roboter BIRON beschrieben. In die resultierende Architektur wurden dabei alle zuvor vorgestellten Software-Komponenten, die für den Roboter BIRON implementiert wurden, integriert. Insgesamt ergibt sich ein modular organisiertes und erweiterbares System, nicht zuletzt durch die Verwendung der zentralen, generischen Kontrollkomponente, dem Execution Supervisor, dessen Implementierung ebenfalls in diesem Kapitel vorgestellt wurde. Das Konzept dieses Execution Supervisors skaliert nicht nur für zusätzlich zu integrierende Fähigkeiten, sondern auch, wenn das System um neue Komponenten erweitert wird. Im Allgemeinen wird nur ein neuer Zustand in dem EEA des Execution Supervisors benötigt, wenn eine neue Funktionalität oder ein neues Modul zum System hinzugefügt wird. Um dies zu erreichen, muss lediglich die XML-basierte Konfigurationsdatei des Execution Supervisors angepasst werden.

Ein Kritikpunkt an der vorgestellten Integrationsmethode ist, dass das Resultat eine zentralisierte Architektur ist. Sollte der Execution Supervisor ausfallen, so würde das gesamte System versagen. Daher wird oft für die Entwicklung verteilter Architekturen argumentiert, um weniger anfällig für den Ausfall einer einzelnen Komponente zu sein.

Allerdings kann der Execution Supervisor einfach als ein weiteres kritisches System des Roboters betrachtet werden, wie es beispielsweise die Stromversorgung des Roboters ist. Darüber hinaus kann auch in einem Behavior-basierten System ein bestimmtes Behavior ausfallen, so dass sich ein unvorhersehbares *Emergent Behavior* ergibt, welches gegebenenfalls schädliche Auswirkungen auf das System oder dessen Umwelt haben kann. Des Weiteren stellt der Execution Supervisor kein sehr komplexes Modul dar, und dessen Funktionsweise verändert sich aufgrund der Generizität auch dann nicht, wenn das System erweitert wird. Da die Funktionsweise des Execution Supervisors deterministisch ist, lässt sich die Ausfallwahrscheinlichkeit des Moduls bereits im Entwicklungsprozess drastisch minimieren.

Die wahrscheinlichsten Fehler, die im Gesamtsystem auftreten können, basieren auf fehlerhaften Nachrichten. Da im System verschickte Nachrichten in XML formuliert sind, kann eine syntaktische Analyse der Daten bereits durch die Kommunikations-Software unter der Verwendung von XML-Schemas durchgeführt werden. Da der Execution Supervisor erhaltene Ereignisse nicht interpretiert, ist er auch gegen semantische Fehler in den Nachrichten immun. Sollten fehlerhafte Daten vom System detektiert werden, so wird immer das verursachende Modul benachrichtigt, welches sich dann deren Behandlung widmen muss.

Schließlich ergibt sich durch die Verwendung von XML für die Konfiguration des Execution Supervisors und der Kommunikation zwischen den einzelnen Modulen des Systems ein hohes Maß an Wartbarkeit. Insgesamt ist eine mächtige und einfach erweiterbare Architektur für den Roboter BIRON entstanden, in der entsprechende Fähigkeiten zur Interaktion zwischen Mensch und Roboter auf einfachste Weise integriert werden können. Die aktuellen Interaktionsfähigkeiten von BIRON, die durch die Integration der in den vorangegangenen Kapiteln präsentierten

Komponenten erreicht wurden, sind zum Abschluss dieses Kapitels ebenfalls präsentiert worden. Um deren Leistungsfähigkeit im Kontext des Gesamtsystems belegen zu können, werden im folgenden Kapitel die Ergebnisse entsprechender Evaluationen präsentiert.

8. Evaluation des Gesamtsystems

In diesem Kapitel werden diverse Experimente beschrieben, die zur Evaluation des Robotersystems BIRON durchgeführt wurden. Zuerst werden die Ergebnisse von drei unterschiedlichen Experimenten zum Personen-Tracking vorgestellt, die in verschiedenen Entwicklungsphasen des Systems vorgenommen wurden. Anschließend werden Ergebnisse einer Benutzerevaluation präsentiert, um die Qualität der Interaktion zwischen unbefangenen Personen und dem Roboter als Gesamtsystem bewerten zu können. Da eine wichtige Voraussetzung für ein natürliches Gesamtverhalten des Roboters eine flüssige Interaktion ist, werden außerdem Ergebnisse einer quantitativen Evaluation präsentiert. In dieser Auswertung wurde die Geschwindigkeit der Datenverarbeitung in der Architektur bestimmt, um die Qualität des Ansatzes zu bewerten. Schließlich wird die Präsentation des Systems auf einer dreitägigen Ausstellung beschrieben, auf der zwei BIRON-Roboter ihre Interaktionsfähigkeiten erfolgreich den Besuchern dargeboten haben.

8.1. Personen-Tracking-Experimente

Die Experimente zum Personen-Tracking wurden, wie auch die Entwicklung des Personen-Trackings selbst, in Kooperation mit S. Lang durchgeführt. Da die Implementierung der Tracking-Komponente nur einen der zwei Schwerpunkte dieser Arbeit darstellt, werden die Ergebnisse der zugehörigen Experimente hier lediglich zusammenfassend dargestellt. Eine ausführlichere Beschreibung findet sich in der Dissertationsschrift von S. Lang [Lan05].

Da die drei Experimente, die im Folgenden beschrieben werden, zu verschiedenen Zeitpunkten durchgeführt wurden, befand sich das Personen-Tracking auch jeweils in einem unterschiedlichen Entwicklungsstadium. Im ersten Experiment standen dem multimodalen Anchoring lediglich zwei, im zweiten Experiment drei und im dritten Experiment vier monomodale Anchoring-Prozesse zur Verfügung. D.h., zum Zeitpunkt des ersten Experiment waren nur die Detektionskomponenten für Beine und Gesichter im Tracking-System integriert. Das zweite Experiment wurde dann durchgeführt, nachdem die Sprecherlokalisation zum System hinzugekommen ist. In der letzten Ausbaustufe wurde das Personen-Tracking schließlich um die Oberkörperdetektion erweitert, so dass es von diesem Zeitpunkt an der Form entsprach, wie sie in Abschnitt 5.3 beschrieben ist. Daher war es im dritten Experiment das Ziel, zu evaluieren, wie diese zuletzt eingefügte Komponente mit den zuvor integrierten perzeptuellen Systemen zusammenspielt.

Folgen einer Person

Im ersten Experiment sollte die Leistungsfähigkeit des Personen-Trackings in einem dynamischen Szenario untersucht werden [Fri03]. Zum Zeitpunkt des Experiments standen nur die monomodalen Anchoring-Prozesse für Beine und Gesichter zu Verfügung. Außerdem wurde zur Generierung von Gesichtsperzepten auf die weniger leistungsfähige Variante zurückgegriffen, welche die Gesichtsdetektion über eine adaptive Hautfarbensegmentierung, kombiniert mit der *Eigenface*-Methode, realisiert (vgl. Abschnitt 5.3.1). Das Experiment wurde mit zwei unterschiedlichen Versuchsanordnungen durchgeführt. Diese sind in Abbildung 8.1 dargestellt.

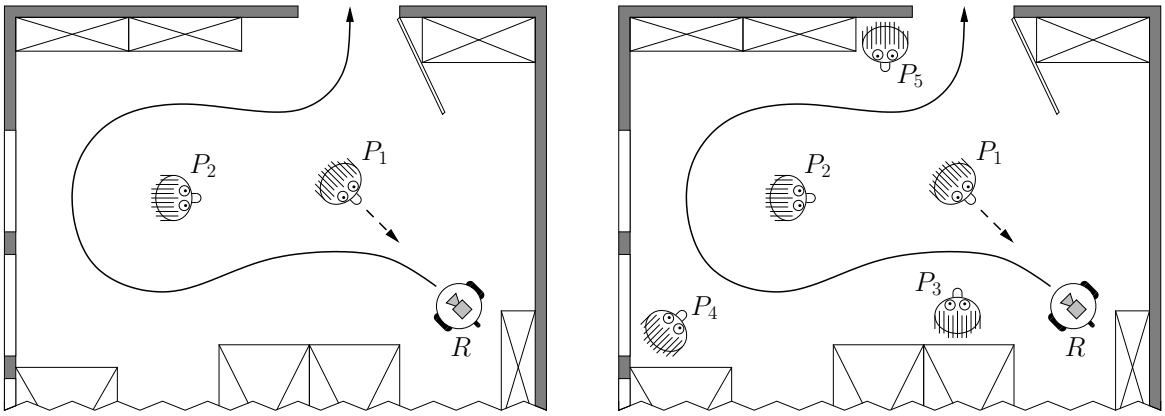


Abbildung 8.1.: Verwendete Versuchsanordnungen: (a) mit zwei Personen, (b) mit fünf Personen. Die Aufgabe von Person P_1 war es, den Roboter R den eingezeichneten Pfad entlang und schließlich aus dem Raum zu führen.

Das Experiment wurde in einem Teil eines Büroraumes durchgeführt. Die Fläche dieses Bereichs betrug ungefähr 4,6 m$\times$3,4 m. Der Raum war mit hellen Holzmöbeln ausgestattet, was eine Herausforderung für die Gesichtsdetektion darstellt, da Holz und Haut eine ähnliche Farbverteilung besitzen. Im ersten Aufbau (siehe Abb. 8.1a) waren zwei Personen anwesend: P_2 befand sich etwa in der Mitte des Raumes und hat sich während des Experiments nicht bewegt. Die Aufgabe von P_1 war es, den Roboter über einen festgelegten Pfad aus den Raum zu führen. Im zweiten Aufbau (siehe Abb. 8.1b) wurden zu der ersten Versuchsanordnung lediglich drei zusätzlichen Personen (P_3–P_5) hinzugefügt, die sich ebenfalls wie P_2 nicht bewegt haben.

Damit der Roboter von P_1 aus dem Raum geführt werden konnte, wurde er so konfiguriert, dass er einer Person folgt, sobald diese sich dem Roboter auf weniger als 1 m annähert. Während des Folgens hat der Roboter dann versucht einen konstanten Abstand zu der Person zu halten. In dieser Phase sollte Person P_1 außerdem möglichst durchgängig zum Roboter blicken, so dass ihr Gesicht vom Roboter detektiert werden konnte. Außerdem war es die Aufgabe von P_1, den Vorgang des Folgens erneut zu initiieren, falls der Roboter die Person während eines Durchlaufs verloren haben sollte, so dass der Lauf fortgesetzt werden konnte. Falls dies nicht möglich war,

weil der Roboter versuchte, einer anderen Personen zu folgen, die dazu weniger als 1 m weit entfernt sein musste, wurde der Durchlauf erfolglos abgebrochen. Für beide Versuchsanordnungen (siehe Abb. 8.1) wurden jeweils zehn Durchläufe mit verschiedenen Testpersonen durchgeführt.

Im ersten Aufbau (siehe Tab. 8.1) wurde die Aufgabe im Schnitt in 55 s erfüllt. In drei Durchläufen hat der Roboter P_1 jeweils einmal verloren. Die Versuchspersonen waren jedoch jedes Mal in der Lage, die Verfolgung erneut zu aktivieren, um den Durchlauf erfolgreich zu beenden. Der multimodale Anchor für P_1 war pro Durchlauf im Schnitt zu 95,3% der Zeit im Zustand *grounded*. Dabei konnten im monomodalen Anchoring-Prozess für die Beine entsprechende Perzepte zu 92,1% der Zeit zugeordnet werden, für die Gesichtern war dies nur zu 42,1% der Zeit möglich. Die unterschiedlichen Zuordnungsraten ergaben sich, da für die Beinerkennung im Mittel in jedem Arbeitszyklus 1,76 Perzepte erzeugt wurden, für die Gesichtserkennung aber nur 0,60 Perzepte. Dies lag unter anderem daran, dass P_1 nicht durchgängig zur Kamera geschaut hat.

Lauf	t (s)	$v_\varnothing$ (m/s)	wie oft verloren	*grounded* (%)			Perzepte/Arbeitszyklus	
				Person	Beine	Gesicht	Beine	Gesicht
1	39	0,19	0	99,7	98,9	63,1	1,78	0,76
2	62	0,12	0	96,6	93,8	36,4	1,71	0,90
3	52	0,14	1	95,4	83,5	51,0	1,72	0,57
4	56	0,13	0	99,3	93,8	54,1	1,79	0,59
5	81	0,09	1	96,4	95,7	34,7	1,63	0,40
6	32	0,23	0	99,2	98,7	51,1	1,87	0,78
7	90	0,08	1	80,9	73,2	22,0	1,94	0,49
8	51	0,15	0	99,2	98,8	56,5	1,75	0,70
9	42	0,18	0	98,0	97,9	35,7	1,79	0,45
10	44	0,17	0	88,6	87,1	16,3	1,60	0,39
∅	55	0,14	–	95,3	92,1	42,1	1,76	0,60

Tabelle 8.1.: Ergebnisse aus dem ersten Aufbau mit zwei Personen

Im zweiten Aufbau (siehe Tab. 8.2) benötigten die Versuchspersonen im Schnitt nur zwei Sekunden länger als in der ersten Anordnung, um die Aufgabe zu bewältigen, obwohl hier drei Personen mehr anwesend waren. Aus diesem Grund wurden im zweiten Aufbau vom System auch durchgängig mehr als zwei Beinperzepte erzeugt. Gesichter wurden hingegen weniger oft detektiert, was einerseits daran lag, dass die Personen P_3 – P_5 nicht in die Kamera geblickt haben. Andererseits haben sich die Versuchspersonen P_1 öfters umgeschaut, um sicherzustellen, dass sie nicht mit einer der anderen Personen kollidieren. Folglich blickten sie seltener zum Roboter, was zu einer entsprechend niedrigeren Erkennungsrate bei der Gesichtsdetektion führte. Im Schnitt war der monomodale Anchor für das Gesicht der Person P_1 zu 32,8% der Zeit im Zustand *grounded*. Die Zuordnungsraten für den monomodalen Bein-Anchor und für den multimodalen Personen-Anchor waren mit 92,7% bzw. 95,3% ähnlich zu denen des ersten Experiments. Die Durchläufe 3 und 6 schlugen fehl, da der Roboter, direkt nachdem er P_1 verloren hatte, versuchte einer anderen Person zu folgen, die in dem Moment weniger als 1 m weit entfernt war.

Lauf	t (s)	$v_{\varnothing}$ (m/s)	wie oft verloren	*grounded* (%)			Perzepte/Arbeitszyklus	
				Person	Beine	Gesicht	Beine	Gesicht
1	60	0,13	2	93,6	91,5	27,7	2,63	0,41
2	43	0,17	0	96,7	95,0	20,7	2,61	0,32
3	Der Roboter hat P_1 verloren und versuchte P_3 zu folgen.							
4	51	0,15	0	98,7	90,4	66,0	2,49	0,74
5	47	0,16	0	96,2	94,5	7,1	2,52	0,20
6	Der Roboter hat P_1 verloren und versuchte P_2 zu folgen.							
7	77	0,10	0	99,8	97,5	72,0	2,59	0,85
8	74	0,10	0	93,4	92,6	20,3	2,63	0,22
9	61	0,12	0	97,7	96,1	36,4	2,55	0,56
10	42	0,18	0	86,1	84,2	11,9	2,73	0,26
$\varnothing$	57	0,13	–	95,3	92,7	32,8	2,59	0,45

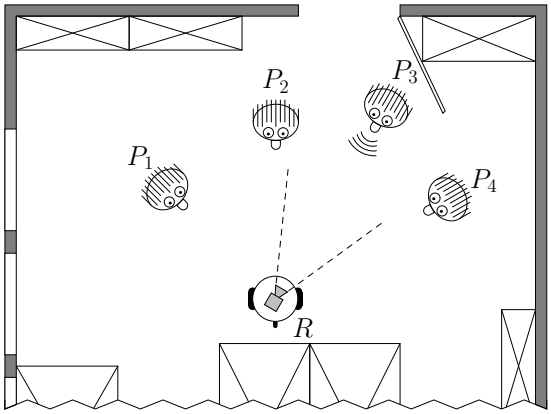
Tabelle 8.2.: Ergebnisse aus dem zweiten Aufbau mit fünf Personen

Insgesamt hat das oben beschriebene Experiment gezeigt, dass das multimodale Anchoring zum Verfolgen von Personen in einem dynamischen Szenario sehr geeignet ist. Die zwei verwendeten Modalitäten haben sich dabei gut ergänzt, d.h. die Kombination aus Bein- und Gesichtsdetektion hat zu einem robusteren Personen-Tracking geführt.

Sprecherlokalisation

In einem zweiten Experiment war es das Ziel, das Personen-Tracking zu testen, nachdem die Sprecherlokalisation eingebunden wurde [Lan03]. Wie in Abschnitt 5.3.1 beschrieben, kann die Position eines Sprechers unter Verwendung nur eines Mikrofonpaares, wie es auf dem Roboter BIRON zur Verfügung steht, lediglich auf eine Hälfte eines zweischaligen Hyperboloids eingeschränkt werden. Erst unter Berücksichtigung zusätzlicher Informationen über die Person lässt sich der Sprecher eindeutig lokalisieren. Diese Daten werden von dem multimodalen Anchoring geliefert: Gefundene Beine liefern die Position einer Person und ein detektiertes Gesicht gibt außerdem Aufschluss über die Größe der Person. Zur Gesichtserkennung wurde hier bereits das in Abschnitt 5.3.1 beschriebene, effiziente Verfahren nach Viola und Jones eingesetzt.

Die Sprecherlokalisation wurde zwar in Verbindung mit dem multimodalen Gesamtverfahren untersucht, der zugehörige monomodale Anchoring-Prozess bildete dabei jedoch den Schwerpunkt der Evaluation. Ein Sprach-Anchor wurde erzeugt, sobald einer Person ein Sprachperzept zugeordnet werden konnte. Wenn für diese Person für mehr als zwei Sekunden lang kein weiteres Sprachperzept akquiriert werden konnte, so wurde der Sprach-Anchor wieder aus dem Gesamt-Anchoring-Prozess entfernt. Der Roboter wurde so konfiguriert, dass er seine Kamera immer auf die Person ausrichtete, die einen neuen Sprach-Anchor erhalten hat. Diese Person wurde dabei so lange fokussiert, wie der monomodale Anchor vorhanden war. Der Roboter selbst hat sich



(a)

Person \ Schritt	1	2	3	4	5	6	7	8	9	10	11	12
P_1	●					●			●			
P_2		●					●					●
P_3			●		●						●	
P_4				●				●		●		

(b)

Abbildung 8.2.: Versuchsaufbau zur Evaluation der Sprecherlokalisation: (a) Aufsicht, (b) Reihenfolge der Sprecher. In den Zeitschritten 1–4 und 9–12 wurde der Roboter angesprochen, in den Zeitschritten 5–8 eine benachbarte Person.

allerdings nicht bewegt. Das Experiment wurde in dem Büro durchgeführt, in dem auch schon das erste Experiment stattgefunden hatte. Der zugehörige Aufbau ist in Abbildung 8.2a zu sehen.

In diesem Experiment wurden vier Personen um den Roboter herum an vorgegebenen Positionen (vgl. Abb. 8.2a) stationiert. Die Probanden hatten die Aufgabe, nacheinander und jeweils 10 Sekunden lang zu sprechen. Sie sollten dabei entweder zum Roboter oder zu einer der anderen Versuchspersonen reden, indem sie sich in die entsprechende Richtung gedreht haben. Die Reihenfolge der Sprecher und der jeweilige Adressat waren festgelegt und sind in Abbildung 8.2b zu sehen. Dieses Experiment wurde dreimal durchgeführt, wobei insgesamt neun verschiedene Testpersonen beteiligt waren. Die zugehörigen Ergebnisse sind in Abbildung 8.3 dargestellt.

Der Roboter war in jedem Durchlauf in der Lage, innerhalb der jeweils 10 Sekunden lang andauernden Sprechphase den Sprecher zu bestimmen und mit der Kamera zu fokussieren. Jedoch wurde in manchen Situationen entweder der Fokus zu lange auf dem vorangegangenen Sprecher belassen oder es wurde für kurze Zeit eine falsche Person selektiert. Ein solches fehlerhaftes Verhalten trat in 6 der insgesamt 36 Phasen auf. In vier Fällen fokussierte der Roboter kurz eine Person, die sich in dem Moment still verhielt (siehe rote Pfeile in Abbildung 8.3). In den zwei übrigen Fällen wurde zwar die richtige Person vom Roboter fokussiert, jedoch geschah dies erst nach einer längeren Verzögerung (siehe blaue Pfeile in Abbildung 8.3). In allen anderen Phasen betrug die Verzögerung beim Fokuswechsel nur ungefähr zwei Sekunden. Diese Dauer ergab sich dadurch, dass ein Sprach-Anchor, wie oben beschrieben, erst dann aus dem multimodalen Anchoring-Prozess entfernt wurde, nachdem zwei Sekunden lang kein Sprachperzept zugeordnet werden konnte. Folglich konnte auch erst danach ein neuer Sprecher fokussiert werden.

Es fällt auf, dass in allen Situationen in denen ein fehlerhaftes Verhalten aufgetreten ist, die Person P_2 als Sprecher erkannt wurde, obwohl sie zu dem Zeitpunkt nicht gesprochen hat. Darüber

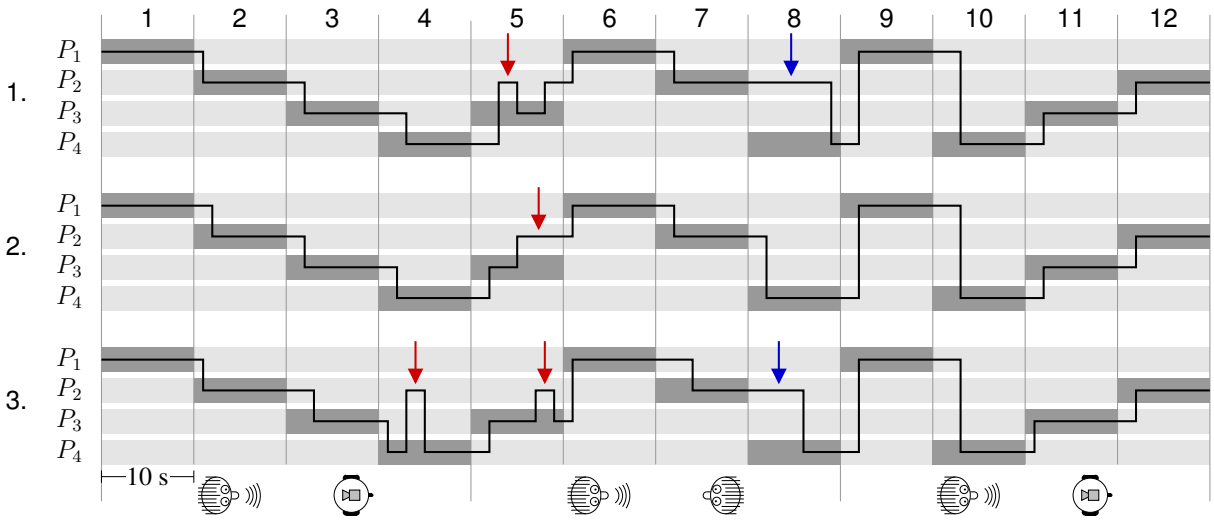


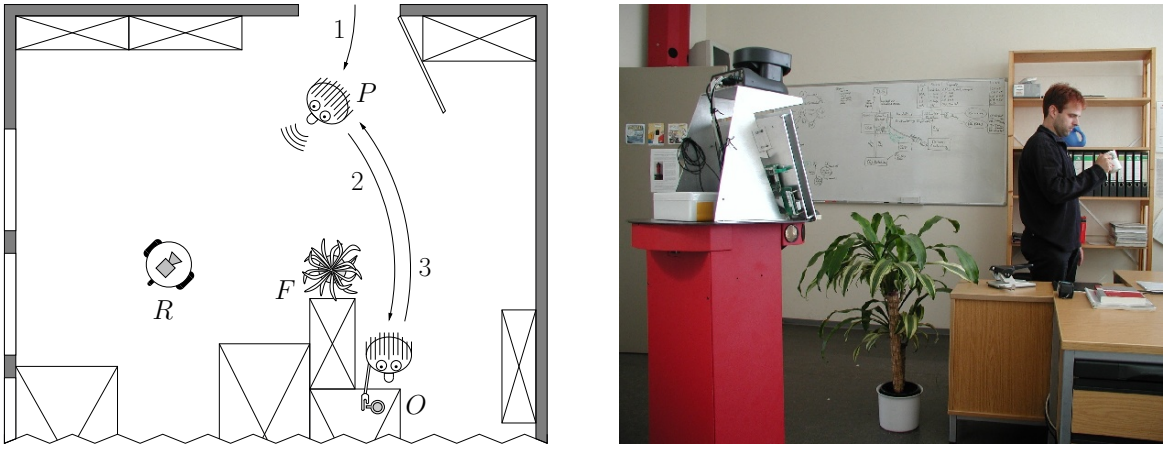
Abbildung 8.3.: Ergebnisse zur Evaluation der Sprecherlokalisation: Jeder Person ist ein horizontaler Balken zugeordnet, der in den Phasen dunkel gefärbt ist, in denen die Person gesprochen hat. Die drei schwarzen Linien geben an, auf wen sich der Roboter jeweils ausgerichtet hat. Die farbigen Pfeile kennzeichnen fehlerhaftes Verhalten des Roboters.

hinaus ist in fünf der sechs genannten Fälle nicht der Roboter, sondern eine der anderen Personen vom Sprecher adressiert worden. Alle Fehler traten auf, weil eine Geräuschquelle frontal vor dem Roboter lokalisiert wurde, d.h. an der Stelle wo sich Person P_2 befunden hat. Die Ursache dafür waren die Eigengeräusche des Roboters, da diese vom System als eine Geräuschquelle aus der entsprechenden Richtung interpretiert wurden.

Insgesamt hat dieses Experiment gezeigt, dass es mit nur einem Mikrofonpaar möglich ist, sprechende Personen von einem Roboter aus zu lokalisieren, indem auf Informationen anderer Modalitäten zurückgegriffen wird. Die durch Störgeräusche des Roboters verursachten Fehler könnten aber noch verringert werden, indem ein Verfahren zur Unterscheidung von Sprache und anderen Geräuschen eingesetzt werden würde. Ein anderer, jedoch auch technisch aufwändigerer Ansatz wird von Nakadai et al. [Nak00] verwendet, bei dem ein zusätzliches Mikrofonpaar im inneren des Roboters montiert ist. Diese Mikrofone nehmen die Eigengeräusche des Systems auf, so dass diese in dem anderen Signal eliminiert werden können.

Verfolgen des Oberkörpers

Im dritten Experiment wurde das vollständige multimodale Anchoring-System verwendet, wie es in Abschnitt 5.3 beschrieben ist. Es sollte dabei insbesondere die Erweiterung zur Lokalisation von Oberkörpern untersucht werden [Fri04]. Der dazu verwendete Versuchsaufbau ist in Abbildung 8.4 dargestellt.



(a) (b)

Abbildung 8.4.: Versuchsaufbau zum Verfolgen eines Oberkörpers: (a) Aufsicht, (b) Sicht aus einer Position hinter dem Roboter.

Zu Beginn befand sich der Roboter R an einer vorgegebenen Position in dem Versuchsraum, der auch schon in den anderen Experimenten verwendet wurde, und war auf die Tür ausgerichtet. Daraufhin hat ein Proband P den Raum betreten. Damit der Roboter begonnen hat, diese Person zu verfolgen, musste sie den Roboter ansprechen. Das Verfolgen der Versuchsperson wurde dann durch Ausrichtung der Kamera und Rotation der Roboterbasis erreicht. Der Proband hat sich dann zu einem bestimmten Schreibtisch begeben, um dort mit einem vorhandenen Objekt O zu interagieren. Schließlich ist die Person wieder an die Position zurückgekehrt, an welcher der Roboter damit begonnen hat, sie zu verfolgen.

Die Herausforderung für das Personen-Tracking bestand bei diesem Experiment darin, die Testperson zu verfolgen, nachdem sie die auf dem Boden stehende Pflanze F passiert hatte. Von dem Zeitpunkt an war nämlich nur noch der Oberkörper der Person für den Roboter sichtbar. Das lag zum einen daran, dass die Beine des Probanden durch die Pflanze oder den daneben stehenden Schrank verdeckt waren, und zum anderen, dass die Person ihr Gesicht vom Roboter abgewendet hat. Außerdem hat die Versuchsperson nicht gesprochen. Die Person konnte also nur dann bis zum Ende eines Durchlaufs verfolgt werden, wenn das System in der Lage war, die Person erfolgreich anhand des Oberkörpers zu verfolgen.

Insgesamt wurden 25 Durchläufe mit mehreren Versuchspersonen durchgeführt, die Oberkörperbekleidungen mit verschiedenen Farben und Mustern trugen. Es waren dabei 80% der Durchläufe erfolgreich, d.h. in diesen Fällen konnte die jeweilige Versuchsperson während des gesamten Durchlaufs verfolgt werden. In der Phase, in der nur der Oberkörper der Person für den Roboter sichtbar war, hat die Person auch mit dem Objekt interagiert. Diese Phase dauerte im Schnitt 5 bis 10 Sekunden und alle erfolglosen Durchläufe sind währenddessen fehlgeschlagenen. Ohne die Oberkörperdetektion wäre allerdings kein einziger Durchlauf erfolgreich gewesen, so dass das Personen-Tracking durch diese Komponente entschieden verbessert werden konnte.

8.2. BIRON aus Sicht des Benutzers

Um das Verhalten des integrierten Systems und die Interaktionsfähigkeiten von BIRON bewerten zu können, wurden entsprechende Versuche mit 21 Testpersonen durchgeführt [Wre04a, Li04]. Außerdem sollte die generelle Einstellung der Benutzer gegenüber einem solchen *Robot Companion* bestimmt werden. Die Experimente wurden in einem ausreichend großen Raum durchgeführt, so dass sich der Roboter frei bewegen konnte, ohne mit Hindernissen zu kollidieren (siehe Abb. 8.5). Die Probanden waren zwischen 22 und 54 Jahren alt, wobei die berufliche Spanne vom Landwirt bis hin zum Computerwissenschaftler reichte. Dennoch hatten die meisten Testpersonen einen technischen Hintergrund.

Jede Testperson wurde instruiert, die folgende Prozedur unter Verwendung von sprachlichen Anweisungen zu absolvieren: Zuerst sollte sie sich dem Roboter nähern und ihn durch eine Begrüßung auf sich aufmerksam machen. Anschließend sollte sie den Roboter auffordern, ihr zu folgen. Nachdem der Roboter ein Stück durch den Raum geführt wurde, war der Roboter anzuhalten, um ihm ein Objekt zeigen zu können. Schließlich sollte die Testperson die Interaktion beenden, indem sie sich vom Roboter verabschiedet. Jede Testperson benötigte für diese Interaktion etwa 3 bis 5 Minuten. Danach wurden die Probanden gebeten, einen Fragebogen auszufüllen.

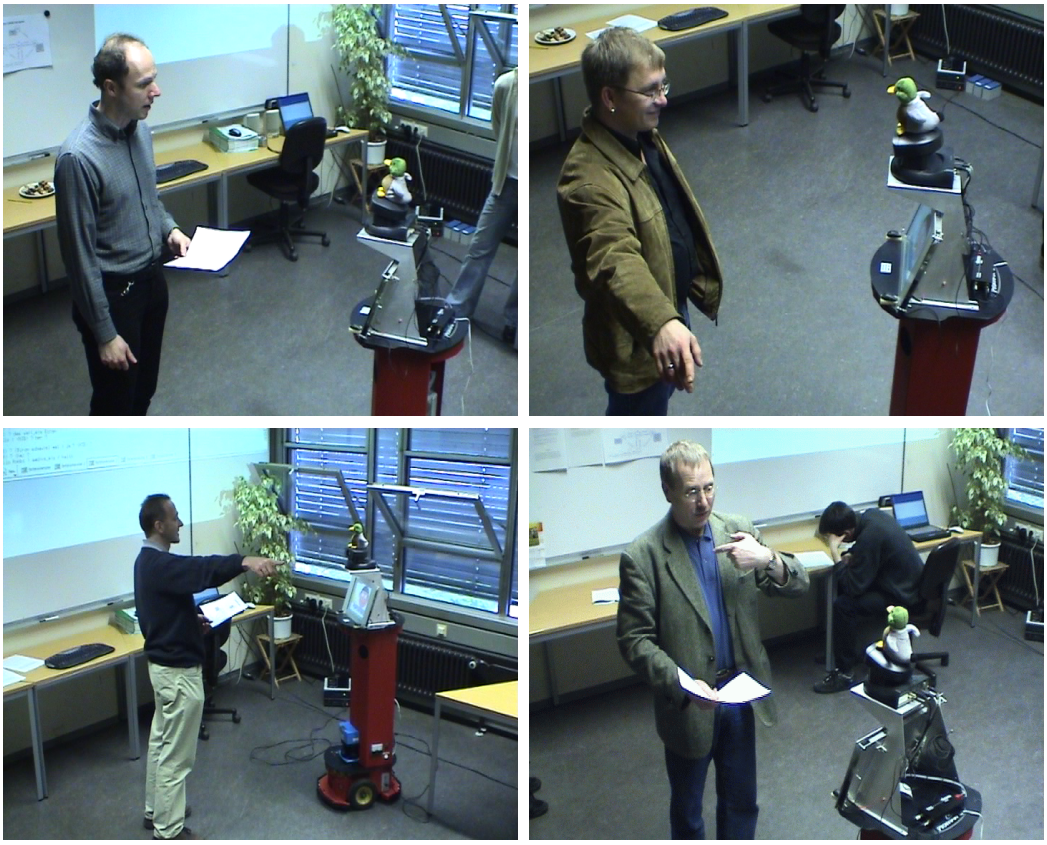


Abbildung 8.5.: Verschiedene Benutzer bei ihrer ersten Interaktion mit BIRON.

8.2.1. Evaluation individueller Systemkomponenten

Um das Systemverhalten auf Grundlage der individuellen Komponenten qualitativ bewerten zu können, wurden den Testpersonen folgende zwei Fragen gestellt: „Welche Fähigkeiten des Roboters fanden Sie besonders interessant?“ und „Was hat Ihnen an BIRON am wenigsten gefallen?“ Bei beiden Fragen waren Mehrfachnennungen erlaubt. Das Ergebnis der Befragung ist in den Abbildungen 8.6 und 8.7 dargestellt und wird im Folgenden diskutiert.

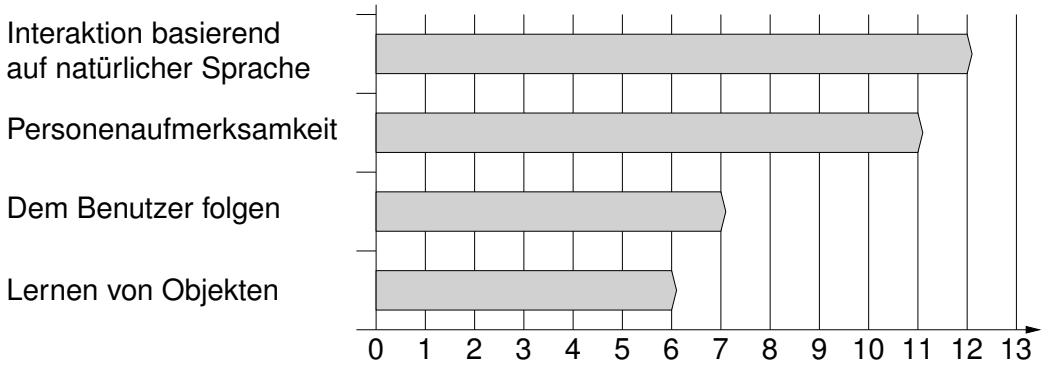


Abbildung 8.6.: Histogramm der Antworten auf die Frage „Welche Fähigkeiten des Roboters fanden Sie besonders interessant?“ (Mehrfachnennungen waren erlaubt.)

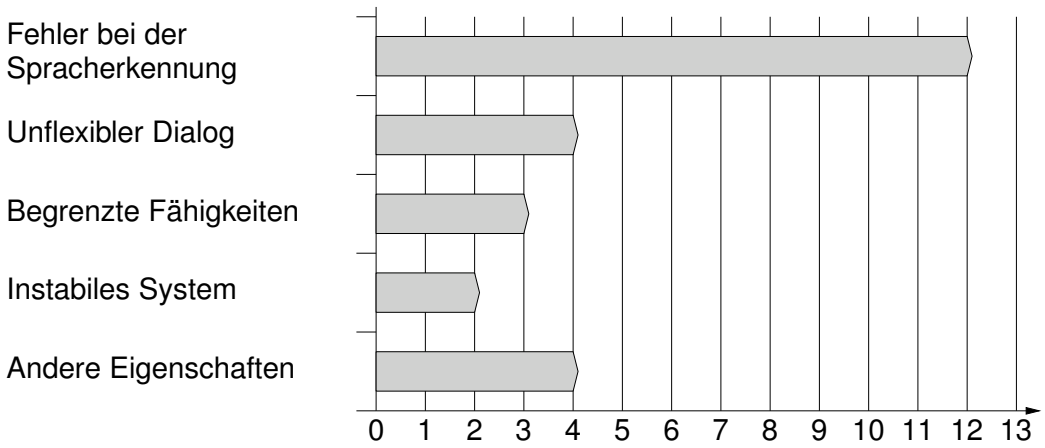


Abbildung 8.7.: Histogramm der Antworten auf die Frage „Was hat Ihnen an BIRON am wenigsten gefallen?“ (Mehrfachnennungen waren erlaubt.)

Bezüglich der Aufmerksamkeitssteuerung für Personen gab es die meisten positiven Antworten. Von den 21 Testpersonen empfanden 11 die Fähigkeit des Roboters interessant, dass er sie fokussieren konnte, während 7 dadurch beeindruckt waren, dass er ihnen hinterherfahren konnte (siehe Abb. 8.6). Keiner der Probanden hat in der zweiten Frage eine Unzufriedenheit gegenüber diesen Fähigkeiten geäußert. Daraus kann geschlossen werden, dass sie intuitiv nachvollziehbar sind und relativ natürlich wirken.

Im Hinblick auf die Spracherkennung ist interessant, dass die Antworten der ersten Frage nicht mit denen der zweiten Frage korrelieren. Es war zwar für 12 Testpersonen interessant, dass die Interaktion mit dem Roboter auf natürlicher Sprache beruht (siehe Abb. 8.6), aber ebenso viele Probanden bemängelten auch die zu geringe Leistungsfähigkeit der Spracherkennung (siehe Abb. 8.7). Dieses Ergebnis impliziert, dass die Verwendung von natürlicher Sprache die Attraktivität eines *Robot Companions* erhöht. Gleichzeitig ist aber auch ersichtlich, dass die Leistungsfähigkeit der Spracherkennung ausschlaggebend für die Akzeptanz eines solchen Roboters ist.

Da die natürliche Sprache so wichtig ist, spielt auch die Dialogsteuerung eine entscheidende Rolle bei der Interaktion mit dem Benutzer. Die aktuelle Implementierung erlaubt zwar die grundlegende Kommunikation mit dem Menschen, allerdings scheint sie nicht flexibel genug zu sein, da sie die Freiheit des Benutzers bezüglich seiner zu stark Wortwahl einschränkt. Dieses spiegelt sich in den Verbesserungsvorschlägen wider (siehe Abb. 8.7), da sich dort 4 Probanden für ein flexibleres Dialogschema ausgesprochen haben.

Des Weiteren wurde angenommen, dass eine Testperson neben der Sprachausgabe des Roboters auch an seinem internen Zustand interessiert sein könnte. Daher sind den Testpersonen während des Experiments die Ergebnisse der Spracherkennung und der jeweils aktuelle Zustand des Execution Supervisors (vgl. Abb. 7.3) präsentiert worden. Allerdings wurden die Probanden in zwei Gruppen eingeteilt, wobei nur die 11 Personen aus der ersten Gruppe die Spracherkennungsergebnisse erhalten haben. Sie wurden nach dem Experiment gefragt, ob diese Informationen hilfreich waren, worauf 6 von ihnen zustimmten. Den 10 Personen der zweiten Gruppe wurden diese Informationen vorenthalten. Daraufhin bemerkten 3 von ihnen, dass Kenntnisse über die Ergebnisse der Spracherkennung aufschlussreich gewesen wären. Von allen 21 Testpersonen waren 14 der Meinung, dass das Wissen darüber, in welchem Zustand sich der Roboter während der Interaktion jeweils befunden hat, hilfreich war. Da die Sprachverarbeitung auf dem Roboter zum Zeitpunkt der Experimente (Mai 2004) ungefähr 2 bis 3 Sekunden länger dauerte als bei einem Menschen, war diese Art der Rückmeldung extrem wichtig für die Probanden. Allerdings haben auch einige Testpersonen angemerkt, dass die Präsentation der internen Informationen zu technisch für eine natürliche Interaktion ist. Eine Alternative wäre, die Spracherkennungsergebnisse mit den Gesichtsausdrücken des animierten Gesichts, das auf dem Display des Roboters angezeigt wird, zu koppeln. Wenn das Gesicht des Roboters sozusagen den mentalen Status des Gesamtsystems widerspiegeln würde, so könnte ein Benutzer erkennen, ob der Roboter noch nachdenkt oder ihn bereits verstanden hat. Allerdings unterstützt das Gesicht in der aktuellen Implementierung dazu noch nicht genügend Ausdrucksmöglichkeiten. Des Weiteren wird es gegenwärtig auch nicht durch die Spracherkennung, sondern nur durch den Status der Aufmerksamkeitssteuerung für Personen beeinflusst.

8.2.2. Auswertung allgemeiner Benutzerantworten

Bei der Entwicklung von Mensch-Roboter-Schnittstellen sollten die Vorlieben und die Einstellung von potentiellen Nutzern dem Roboter gegenüber berücksichtigt werden. Diese Punkte werden allerdings in vielen Studien außer Acht gelassen. Daher wurden den Testpersonen einige

Fragen gestellt, um ihre generelle Haltung gegenüber Robotern und damit verbundenen Technologien zu erfassen.

Auf die Frage: „Wieviel Erfahrung haben Sie im Umgang mit Spracherkennungssystemen?“ haben überraschenderweise alle Testpersonen geantwortet, dass sie ein solches System zumindest einmal ausprobiert haben. Dies kann darauf zurückgeführt werden, dass sich Sprachtechnologien im Alltag immer weiter durchsetzen, wie z.B. bei telefonischen Informationsdiensten. Daher ist anzunehmen, dass wenigstens technisch interessierte Personen mehr oder weniger auf das Interagieren mit einem *Robot Companion* in natürlicher Sprache vorbereitet sind. Somit bietet sich Sprache als Hauptmodalität für die Interaktion mit einem Roboter nicht nur wegen ihrer Natürlichkeit an, sondern auch weil unter den potentiellen Benutzern die Vertrautheit gegenüber dieser Technik ansteigt.

Trotz der eingeschränkten Interaktionsfähigkeiten des Systems haben alle Testpersonen auf die Frage: „Wieviel Spaß hat Ihnen die Interaktion gemacht?“ eine positive Antwort gegeben. Dieses Ergebnis impliziert zwar eine generelle Offenheit unter den Probanden gegenüber dieser Art von neuer Technologie, aber dennoch sollte der Effekt der Neugier berücksichtigt werden. Die meisten Personen haben noch niemals zuvor mit einem echten Robotersystem interagiert und somit war es für sie interessant, es einmal auszuprobieren. Es ist aber wahrscheinlich, dass nach einiger Zeit das Interesse nachlässt, wenn ein Roboter nur über begrenzte Interaktionsmöglichkeiten verfügt und nur eine überschaubare Anzahl an Aufgaben erfüllen kann. Da ein Roboter initial nur mit einer festen Anzahl an Fähigkeiten ausgestattet werden kann, sollte ein *Robot Companion* in der Lage sein, neue Fähigkeiten zu lernen. Dies würde definitiv einen positiven Einfluss auf die Interaktionsqualität zwischen Mensch und Roboter haben.

Die Frage: „Hätten Sie gerne einen Roboter zu Hause?“ haben 14 der 21 Testpersonen positiv beantwortet. Die 14 Personen wurden auch gefragt, welche Fähigkeiten er in diesem Fall besitzen müsste. Darauf antworteten 9, dass sie ihn am liebsten für Haushaltsaufgaben einsetzen würden, 4 hätten ihn gerne zu Unterhaltungszwecken und einer würde sich gerne vom Roboter an Termine erinnern lassen. Basierend auf diesen Antworten ist ersichtlich, dass die meisten Menschen eine relativ traditionelle Sichtweise auf die Rolle eines Roboters im Haushalt haben. Sie erwarten eher, dass ein Roboter ihnen bei der täglichen Arbeit helfen kann, als dass sie in ihm einen dem Menschen ähnlichen Kameraden sehen würden. Daraus kann geschlossen werden, dass ein *Robot Companion* mit entscheidend mehr sozialen Fähigkeiten ausgestattet werden muss, als es bei aktuellen Service-Robotersystemen der Fall ist.

8.3. Reaktionszeit des Gesamtsystems

Um zu den qualitativen Aussagen aus dem vorigen Abschnitt auch quantitative Ergebnisse des Systemverhaltens zu erlangen, wurden entsprechende Zeitmessungen im Gesamtsystem vorgenommen [Fri05]. Für diese Auswertung wurde ein zusätzlicher Laptop (Pentium IV, 2,53 GHz, 512 Mb) über eine Funkverbindung (11 Mbit) an den Switch angebunden, der die beiden Rechner

auf dem Roboter miteinander verbindet. Auf diesen Laptop wurde die Sprachverarbeitung ausgelagert, da sie im Vergleich zu anderen Komponenten einem relativ hohen Bedarf an Rechenzeit unterliegt. Die Messergebnisse dieser Evaluation wurden anhand einer Beispielinteraktion zwischen BIRON und einem Benutzer aufgenommen und sind in Abbildung 8.8 in Form eines Sequenzdiagramms dargestellt. Alle Interaktionen mit BIRON laufen im Allgemeinen nach diesem Muster ab, allerdings können die Zeiten der Sprachverarbeitung variieren. Ein Grund dafür ist, dass die Verarbeitungsdauer einer Spracheingabe von der Länge der Äußerungen abhängig ist. Es wirkt sich allerdings auf diese Laufzeit auch aus, wie gut eine Äußerung zum Modell des Spracherkenners passt.

In der ersten Phase der Interaktion (siehe Abb. 8.8) wird das System gestartet. Die Aufmerksamkeitssteuerung für Personen ist initial für die Steuerung der Hardware verantwortlich und versendet entsprechende Befehle im Schnitt alle 50 ms. Nachdem der Execution Supervisor gestartet ist, wird die Aufmerksamkeitssteuerung für Personen automatisch so konfiguriert, dass sie nach einem potentiellen Interaktionspartner in der Nähe des Roboters sucht. Zusätzlich wird die Dialogsteuerung über den Zustand des Gesamtsystems informiert, indem sie regelmäßige Statusinformationen vom Execution Supervisor erhält. Dies geschieht ungefähr alle 500 ms. Alle Nachrichten, die im System regelmäßig versendet werden, sind in Abbildung 8.8 durch die heller werdenden Pfeile angedeutet.

In der zweiten Phase nähert sich eine Person dem Roboter und begrüßt ihn. Folglich wird sie von der Aufmerksamkeitssteuerung für Personen als potentieller Interaktionspartner erkannt, da sie zu BIRON spricht. Dieses wird dem Execution Supervisor mitgeteilt, so dass dieser von nun an alle von der Dialogsteuerung empfangenen Instruktionen als gültig akzeptiert. Außerdem wird die Komponente zur Spracherkennung darüber informiert, dass gerade eine Äußerung an den Roboter gerichtet ist. Erst daraufhin wird der Spracherkennungsprozess angestoßen. Durch diesen Mechanismus wird sichergestellt, dass nur Äußerungen verarbeitet werden, die tatsächlich an BIRON gerichtet sind. Somit werden Äußerungen, die z.B. aus einem Gespräch zwischen zwei sich in der Nähe des Roboters unterhaltenden Personen stammen, nicht berücksichtigt. Eine Schwierigkeit ergibt sich allerdings dadurch, dass die Äußerung, anhand derer die Aufmerksamkeitssteuerung für Personen einen Menschen als mit BIRON redend wahrnimmt, von Anfang an verarbeitet werden muss. Bis jedoch die Person als potentieller Interaktionspartner eingestuft wird, ist ein Teil der zugehörigen Äußerung bereits verstrichen. Aus diesem Grund werden die letzten zwei Sekunden aller sprachlichen Eingaben zwischengespeichert, so dass es der Spracherkennung möglich ist, auf den Anfang einer aktuellen Äußerung zuzugreifen. Das Ende einer Äußerung wird dadurch bestimmt, dass die Aufmerksamkeitssteuerung für Personen wahrnimmt, dass der Interaktionspartner aufgehört hat zu sprechen. Dieses Ereignis wird der Spracherkennung ebenfalls mitgeteilt.

Eine übliche Äußerung zur Instruktion von BIRON dauert in der Regel ein bis zwei Sekunden. Der Erkennungsprozess wird gestartet, sobald der Spracherkennung bekannt ist, dass der Benutzer zu sprechen begonnen hat. Nachdem bekannt ist, dass die Person ihre Äußerung beendet hat, wird allerdings noch eine Nachlaufzeit von etwa 200 bis 500 ms benötigt, bis der Spracherkennungsprozess abgeschlossen ist. Anschließend findet die Verarbeitung des Spracherkennungs-

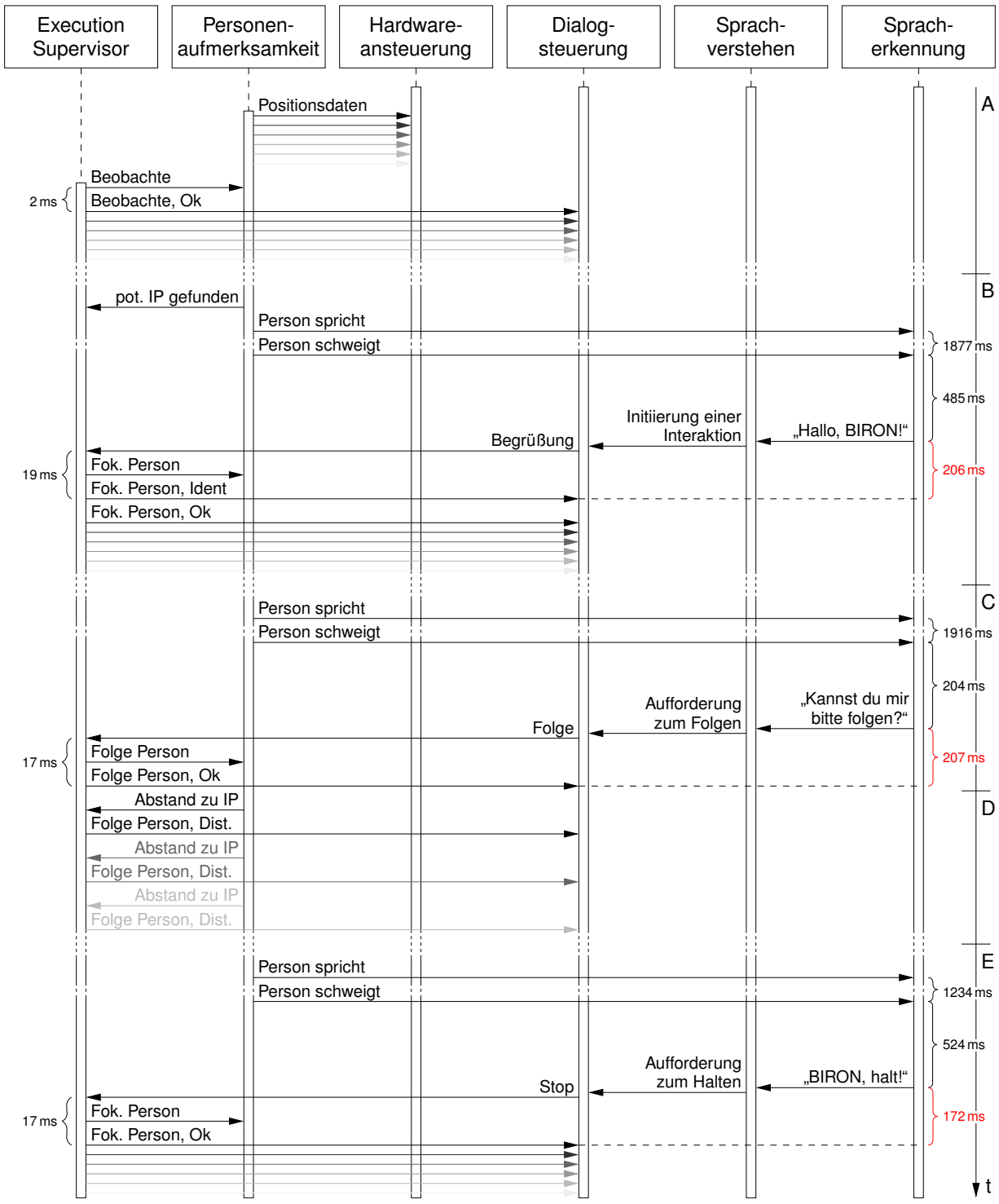


Abbildung 8.8.: Qualitative Darstellung des zeitlichen Verlaufs der Modulkommunikation auf BIRON anhand eines Interaktionsbeispiels. A: Start des Systems. B: Eine Person nähert sich BIRON und begrüßt ihn. C: Die Person bittet BIRON, ihr zu folgen. D: BIRON folgt der Person. E: Die Person bittet BIRON, anzuhalten.

ergebnisses durch die Sprachverstehenskomponente und die Dialogsteuerung statt. Die Person wird daraufhin als Interaktionspartner akzeptiert und der Execution Supervisor rekonfiguriert das Gesamtsystem entsprechend. Die Dialogsteuerung wird durch die regelmäßigen Statusmeldungen über den neuen Zustand des Systems informiert und sorgt schließlich dafür, dass der Roboter eine Sprachausgabe produziert. Dieser ganze Vorgang, d.h. ab dem Zeitpunkt, zu dem das Ergebnis der Spracherkennung bekannt ist, bis hin zur sprachlichen Reaktion des Roboters, benötigt insgesamt nur etwa 200 ms (rote Zeitwerte in Abb. 8.8). In diesem Zeitintervall wird neben der Verarbeitung der Daten auch die Kommunikation zwischen den beteiligten Modulen abgewickelt, die zum Teil über das WLAN verläuft.

In der dritten Phase bittet der Interaktionspartner den Roboter, ihm zu folgen. Die Modulkommunikation verläuft hier ähnlich wie in der vorherigen Phase. Nachdem der Execution Supervisor am Ende dieser Phase das System rekonfiguriert hat, folgt BIRON der Person, wenn sie sich in Bewegung setzt. Dieser Vorgang stellt die vierte Phase in Abbildung 8.8 dar. Hier sendet die Aufmerksamkeitssteuerung für Personen Ereignisse an den Execution Supervisor, die den Abstand zwischen Roboter und Person enthalten. Diese Information wird dann an die Dialogsteuerung weitergereicht, so dass der Benutzer informiert werden kann, falls er sich zu schnell bewegen sollte und daher Gefahr läuft, vom System verloren zu werden.

In der fünften Phase fordert der Interaktionspartner BIRON auf, den Folgevorgang abzubrechen. Die daraus resultierenden Vorgänge im System sind nahezu identisch zu denen in der zweiten Phase des Interaktionsbeispiels. Die Aufmerksamkeitssteuerung für Personen wird auch hier so konfiguriert, dass der Interaktionspartner vom System fokussiert wird, um weitere Instruktionen entgegen nehmen zu können. Die Dialogsteuerung wird natürlich auch über den neuen Systemstatus informiert, um dem Benutzer unter anderem mitteilen zu können, dass BIRON der Person nicht weiter folgen wird.

Anhand der in Abbildung 8.8 angegebenen Zeiten für die wichtigsten Verarbeitungsphasen ist erkennbar, dass die Spracherkennung die meiste Rechenzeit beansprucht. Danach benötigt das System üblicherweise nur noch etwa 200 ms, bis es dem Benutzer antwortet. In Bezug auf diesen Bedarf an Rechenzeit benötigt der Execution Supervisor mit weniger als 20 ms zur Rekonfiguration des Systems nur einen geringen Zeitaufwand. Auch die Zeiten für die Modulkommunikation sind sehr gering und zeigen somit, dass der Vorteil einer erhöhten Rechenkapazität in dem verteilten System den zusätzlichen Kommunikationsaufwand überwiegt. Eine Nachricht von einem Modul zu einem anderen Modul zu übermitteln dauert im Schnitt nur wenige Millisekunden. Eine genauere Aufschlüsselung ist allerdings schwierig, da durch die asynchrone Kommunikation gewisse Verzögerungen bei der Verarbeitung von Nachrichten in den einzelnen Modulen auftreten. Anzumerken ist außerdem, dass die beschriebenen Zeitmessungen das Gesamtsystem zusätzlich gebremst haben, so dass die Verarbeitungsvorgänge im System im Normalfall schneller ablaufen. Insgesamt zeigen diese Werte aber, dass die auf BIRON implementierte Architektur die Interaktion zwischen allen Komponenten im System auf effiziente Weise ermöglicht und somit äußerst reaktionsschnell ist. Folglich könnte eine Beschleunigung des Systems am effektivsten erreicht werden, indem das Ende einer Äußerung durch die Aufmerksamkeitssteuerung für Personen exakter bestimmt wird oder indem der Spracherkennungsprozess schneller arbeitet.

8.4. BIRON auf dem IST-Event 2004

Das IST-Event (Information Society Technologies Event) fand vom 15. bis zum 17. November 2004 in Den Haag in den Niederlanden statt und stand unter dem Motto *„Participate in your Future“*. Dieses Event bot den ungefähr 4000 Teilnehmern eine Konferenz mit 30 Sessions, eine Ausstellung zur Präsentation von konkreten Ergebnissen aus dem Forschungsbereich sowie spezielle Organisationsveranstaltungen zur Bildung von interdisziplinären und multinationalen Forschungs-Teams in Europa. Die übergreifenden Themen waren dabei „Menschen“ und „Wirtschaftlichkeit“. Insgesamt richtete sich das IST-Event an Personen aus folgenden Gruppen:

- Industrie
- Unternehmen
- Entscheidungsträger im öffentlichen Sektor
- Wissenschaftliche Gemeinschaft und Akademiker
- Interessierte Bürger, einschließlich junger Menschen und Studenten

Die AG „Angewandte Informatik“ der Universität Bielefeld nahm mit ihren zwei BIRON-Robotern an der Ausstellung des IST-Events teil (siehe Abb. 8.9). Die Ausstellung erstreckte sich über eine Fläche von etwa 8500 m^2 und bestand aus ungefähr 130 Ständen. Die Grundidee dieser Veranstaltung bestand darin, zu vermitteln, auf welche Weise Informations- und Kommunikationstechnologien neue Wege im Leben der Menschen eröffnen. Dazu zählte, Möglichkeiten zu präsentieren, wie Menschen in Zukunft miteinander leben, arbeiten sowie sich unterhalten können. Ein weiterer Aspekt betraf die Fragestellung, wie sichergestellt werden kann, dass jeder Bürger von der Informationsgesellschaft profitiert, indem z.B. unangenehme Arbeiten zunehmend durch intelligente Maschinen erledigt werden. Die Ausstellung erlaubte es somit den Besuchern, den Stand der Technik in spezifischen Bereichen der Forschung zu besichtigen und mit Forschern über die Ergebnisse ihrer Arbeit zu diskutieren.

Die zwei Roboter wurden der Öffentlichkeit an den drei Ausstellungstagen für eine Dauer von zweimal 9 Stunden und einmal 6 Stunden präsentiert. Instruiert wurden die Roboter allerdings nicht über die auf den Geräten montierten Mikrofone, sondern durch Headsets, die per Funk mit jeweils einem Pentium IV Laptop verbunden waren. Diese Lösung wurde gewählt, um die Vielzahl von Störgeräuschen zu kompensieren, welche durch die enorm laute Umgebung der Ausstellung hervorgerufen wurden. Diese Geräuschkulisse hätte ansonsten eine zu hohe Herausforderung an die Sprachverarbeitung gestellt, da die auf dem Roboter angeordneten Mikrofone, neben Spracheingaben des Benutzers, auch alle in der Nähe befindlichen Geräuschquellen mit aufnehmen. Folglich wurde auch die Sprachverarbeitung auf den Laptops durchgeführt, wobei jeder der Laptops wiederum per Funk (11 Mbit) an den WLAN-Switch des jeweiligen Roboters

Abbildung 8.9.: BIRON auf der Ausstellung des IST-Events 2004 in Den Haag.

angeschlossen war (vgl. Abschnitt 4.1). Somit konnten die Roboter bis auf ein Stromkabel autonom betrieben werden. Eine Verbindung zum Stromnetz war notwendig, da die in den Robotern enthaltenen Akkus keinen Betrieb über mehrere Stunden ermöglicht hätten.

Die beiden Systeme liefen, bis auf kleinere Ausfälle in der Hardware, an allen drei Tagen durchgängig und robust. Neben ein paar Wackelkontakten an den Mikrofonen auf einem der Roboter ergab sich das größte Problem dadurch, dass es auf einem Kanal im Funknetz offensichtlich zu Kollisionen mit den Geräten eines anderen Ausstellers kam. Daraufhin wurde der zur Sprachverarbeitung verwendete Laptop über ein ausreichend langes Netzwerkkabel an den Switch des zweiten Roboters angeschlossen, wodurch das Problem gelöst werden konnte. Somit waren beide BIRON-Roboter in der Lage, den Besuchern der Ausstellung die in Abschnitt 7.3 beschriebenen Fähigkeiten zu präsentieren.

Im Verlauf der Ausstellung haben anwesende Personen insgesamt weit über 15000 mal den Wahrnehmungsbereich der Roboter betreten und wurden somit von dem jeweiligen System registriert. Am zweiten Nachmittag des IST-Events hat einer der Roboter über 5000 Personen detektieren können, da der Andrang entsprechend hoch war. Anzumerken ist dabei allerdings, dass eine bestimmte Person mehrfach gezählt wurde, wenn sie sich aus dem Sichtbereich des Roboters entfernt hat und danach wieder zurückgekehrt ist. Unter Verwendung der bereits genannten Headsets funktionierte die Instruktion der Roboter sehr zuverlässig. Einige Besucher haben sogar selbst versucht, den Roboter zu instruieren. Auch dieses funktionierte nach einer kurzen

Einweisung zur Freude der Gäste, die sich an einer Interaktion versucht haben, erstaunlich gut. Insgesamt wurde die Vorführung der BIRON-Systeme von den Besuchern mit großem Interesse verfolgt und die Darbietung der Roboter durchweg als überaus beeindruckend bezeichnet.

8.5. Zusammenfassung

In diesem Kapitel wurden die Ergebnisse zu unterschiedlichen Experimenten mit dem Roboter BIRON präsentiert. Zuerst wurden drei verschiedene Evaluationen zum Personen-Tracking, welches auf dem im Rahmen dieser Arbeit entwickelten multimodalen Anchoring beruht, beschrieben. Diese Untersuchungen haben die Leistungsfähigkeit des multimodalen Ansatzes unterstrichen. Anschließend wurde der Roboter BIRON auch als Gesamtsystem betrachtet. Dazu wurden einerseits Ergebnisse eines Experiments mit Benutzern des Roboters vorgestellt, um eine qualitative Bewertung der Interaktionsfähigkeiten zu präsentieren. Bei dieser Evaluation hat sich gezeigt, dass die Fähigkeiten des Roboters, obwohl sie noch sehr beschränkt sind, den richtigen Ansatz darstellen, um den Roboter zu instruieren. Andererseits wurde eine Auswertung der Leistungsfähigkeit der entwickelten Roboterarchitektur vorgenommen. Die zugehörigen Ergebnisse zeigen, dass die Architektur eine schnelle Interaktion zwischen den enthaltenen Modulen erlaubt und somit Rekonfigurationen des Systemverhaltens ohne nennenswerte Verzögerungen ermöglicht, was die Voraussetzung für eine flüssige Interaktion mit dem System darstellt. Schließlich wurde das System auf dem IST-Event 2004 in Den Haag präsentiert, wo es auf zwei BIRON-Robotern insgesamt mehr als 24 Stunden lang im Einsatz war. Die Demonstration des Systems war überaus erfolgreich: Die Roboter funktionierten sehr zuverlässig und boten den Besuchern der Messe eine beeindruckende Vorstellung.

9. Zusammenfassung

Das Bestreben in der Robotikforschung, Robotersysteme zu entwickeln, die dem Menschen gewisse Dienste erweisen, ist nach wie vor ungebrochen. Ausgehend von industriellen Systemen, die in Montage-Anlagen nur für ganz spezielle Aufgaben eingesetzt werden können, ist bereits eine Vielzahl von Robotersystemen entstanden, die sowohl mobil sind als auch über gewisse interaktive Fähigkeiten verfügen. Beispielsweise wurden Roboter entwickelt, die Besuchern eines Museums Führungen anbieten können, oder die spezielle Service-Aufgaben, wie z.B. Hol- und Bringdienste, in Büroumgebungen übernehmen können.

Die aktuelle Entwicklung im Bereich der autonomen Robotik konzentriert sich zunehmend auf Systeme für den privaten Gebrauch. Das Bestreben dieser Forschungsrichtung ist es, persönliche Roboter zu entwickeln, die jeder Mensch besitzen kann. Diese Roboter sollen in Zukunft in der Lage sein, mit Menschen, einem Kameraden ähnlich, das Zuhause zu teilen. Ein solcher, so genannter *Robot Companion*, könnte insbesondere für ältere oder behinderte Menschen sehr wertvoll sein, da er ihnen unter anderem bei täglichen Aufgaben zur Hand gehen kann. Somit muss der Roboter über umfangreiche Fähigkeiten verfügen, wobei die Interaktion mit Menschen die wichtigste Fähigkeit darstellt. Allerdings richtet sich ein *Robot Companion* nicht nur an Menschen, die auf Hilfe angewiesen sind, sondern auch an jene Personen, die einfach gerne einen Roboter besitzen möchten. Folglich muss ein solcher Roboter für Menschen einfach zugänglich sein, damit diese dazu geneigt sind, sich einen elektronischen Gefährten zuzulegen.

Aus der gerade genannten Charakteristik eines *Robot Companions* ergibt sich ein besonders hoher Anspruch an die Robotik. Einerseits ist die Realisierung einzelner Fähigkeiten, wie z.B. einen Tisch abräumen oder Blumen gießen zu können, zu berücksichtigen. Andererseits ist zu bedenken, dass die wenigsten potentiellen Besitzer eines *Robot Companions* Experten für Robotik sind. Daher ist es unabdingbar, dass jede Person in der Lage ist, ein solches System auf intuitive Weise zu bedienen. Nur dadurch ist es möglich, dass ein natürlicher Umgang zwischen einem Menschen und einem *Robot Companion* entsteht. Folglich muss der Roboter mit dem Benutzer Dialoge in natürlicher Sprache führen können und zusätzlich Zeigegesten erkennen, falls der Mensch dem *Robot Companion* etwas zeigen möchte. Da entsprechende Interaktionen mit einem solchen Roboter in der Privatwohnung eines Menschen stattfinden sollen, kommt weiter hinzu, dass der Roboter diese Umgebung nicht im Voraus kennen kann. Ebenso ist es nicht möglich, dass das System entsprechendes Wissen über alle erdenklichen Gegenstände, die sich in einer Wohnung befinden können, von Anfang an besitzt. Infolgedessen muss ein *Robot Companion* insbesondere in der Lage sein, all diese Informationen zu lernen. Das erworbene Wissen muss

außerdem auf angemessene Weise verwaltet werden, so dass der *Robot Companion* diese Daten in weiteren Interaktionen entsprechend nutzen kann.

Um die angeführten Herausforderungen lösen zu können, war es ein Ziel dieser Arbeit, eine Software-Architektur für einen *Robot Companion* zu entwickeln. Das Konzept dieser Architektur sollte die Integration aller zur natürlichen bzw. intuitiven Interaktion benötigten Fähigkeiten erlauben und dabei möglichst flexibel und erweiterbar sein, um zusätzliche Fähigkeiten berücksichtigen zu können. Als weiteres Ziel dieser Arbeit sollte die Grundlage zur Interaktion mit Menschen geschaffen werden. Dazu wurde eine neuartiges multimodales Personen-Tracking entwickelt, welches schließlich mit weiteren Interaktionskomponenten in dem realisierten Architekturkonzept zu integrieren war.

Für die Entwicklung der Software-Architektur wurde die bisherige Entwicklung unter den Architekturen für mobile Robotersysteme untersucht. Da die Wahl zu Gunsten einer Architekturform auch zum Großteil von dem Einsatzzweck eines Roboters abhängt, wurden zusätzlich funktionale und strukturelle Anforderungen in Bezug auf einen *Robot Companion* formuliert. Basierend auf diesen Informationen wurde daraufhin das Architekturkonzept für *Robot Companions* entworfen. Dabei handelt es sich um eine besonders flexible Form einer so genannten Drei-Ebenen-Architektur. Diese hybride Architektur ist mit einer zentralen Komponente, dem Execution Supervisor, ausgestattet. Dennoch handelt es sich um keine rein zentralisierte Architektur, da das zentrale Kontrollmodul nur für die Koordinierung des gesamten Systems zuständig ist. Alle Komponenten in der Architektur, die mit dem Execution Supervisor verbunden sind, stellen autonome Agenten dar, die ausreichend Selbstständigkeit besitzen müssen, um auf reaktive Weise in einem umfangreichen Gesamtsystem agieren zu können.

Das im Rahmen dieser Arbeit entwickelte Personen-Tracking ist multimodal, da es die Daten von drei verschiedenen Sensorsystemen verarbeitet, die auf dem Roboter montiert sind. Nur auf diese Weise ist es möglich, Personen in der Nähe des Roboters robust verfolgen zu können, was die wichtigste Voraussetzung für eine Mensch-Roboter-Interaktion darstellt. Zur Fusion der Sensordaten wurde ein spezielles Verfahren, das so genannte multimodale Anchoring, entwickelt. Dieser neuartige Ansatz erlaubt es nicht nur, gleichzeitig mehrere Personen aufgrund ihrer Gesichter, Oberkörper, Stimmen und Beine zu verfolgen, sondern sie auch voneinander zu unterscheiden. Somit ist es möglich, ein einzelnes Individuum bevorzugt zu betrachten, indem z.B. die beweglichen Sensoren des Roboters auf diese spezielle Person ausgerichtet werden. Die verschiedenen, während der Entwicklung des Personen-Trackings durchgeführten Experimente wurden schließlich im 8. Kapitel dargelegt. Durch sie konnte die hohe Leistungsfähigkeit des multimodalen Verfahrens belegt werden.

Neben dem Personen-Tracking wurden jeweils eine Aufmerksamkeitssteuerung für Personen und Objekte und eine Dialogsteuerung vorgestellt. Diese Komponenten sind im Rahmen anderer Promotionsvorhaben entwickelt worden und es galt, sie ebenfalls im Gesamtsystem zu integrieren, da sie zur Interaktion benötigt werden. Die Aufmerksamkeitssteuerung für Personen setzt auf den Daten des multimodalen Personen-Trackings auf. Sie bestimmt, welche Person für den Roboter den Interaktionspartner darstellt und daher von den Sensoren des Roboters fokussiert wird. Zur sprachlichen Kommunikation mit einem solchen Interaktionspartner ist die Dialog-

steuerung verantwortlich. Das Ziel dieses Moduls ist es, durch einen Dialog mit dem Benutzer eine Instruktion zu erhalten, die z.B. die Anweisung darstellt, ein bestimmtes Objekt zu lernen. Für das Erfassen von Gegenständen dient schließlich die Aufmerksamkeitssteuerung für Objekte. Sie kombiniert sprachliche und gestische Informationen, damit das entsprechende Objekt vom Roboter gelernt werden kann.

Nicht zuletzt stand die Umsetzung des präsentierten Architekturkonzeptes für den Roboter BIRON im Vordergrund, um die Tauglichkeit dieses Ansatzes unter Beweis stellen zu können. Dabei wurde insbesondere auf die Implementierung des zentralen Execution Supervisors eingegangen und beschrieben, wie er mit allen vorgestellten Komponenten interagiert. Der Execution Supervisor ist dazu hochgenerisch ausgelegt, so dass bei einer Modifikation oder Erweiterung des Systems nur eine entsprechende Konfigurationsdatei angepasst werden muss, die in XML spezifiziert ist. Der Execution Supervisor empfängt von allen mit ihm verbundenen Komponenten so genannte Ereignisse, die gewisse Aktionen auslösen. Dazu gehört das Weiterleiten von Informationen an entsprechende Module und das Rekonfigurieren des Gesamtsystems. Die gesamte Kommunikation basiert dabei auf dem *XML enabled Communication Framework* und ist somit im höchsten Maße transparent.

Um die Qualität der entworfenen Architektur bewerten zu können, wurden schließlich auch Ergebnisse von Benutzerexperimenten dargelegt, in denen der Roboter BIRON von Testpersonen instruiert wurde. Obwohl bei der Instruktion gelegentlich Fehler im Spracherkennungsprozess aufgetreten sind, so waren die Resultate durchweg positiv. Die Interaktionsfähigkeiten des Roboters wurden als relativ natürlich eingestuft und allen Probanden hat es Spass gemacht, mit dem System zu interagieren. Um neben diesen qualitativen Aussagen auch ein quantitatives Ergebnis zu erhalten, wurde eine weitere Evaluation des Systems durchgeführt. Da für eine flüssige und somit auch natürliche Interaktion das System nicht zu langsam reagieren darf, wurden entsprechende Zeitmessungen vorgenommen. Dabei wurden Verzögerungen, die durch die Sprachverarbeitung, den Execution Supervisor und das Kommunikationssystem in der Architektur entstehen, dem Rechenzeitbedarf der Spracherkennung gegenüberstellt. Diese Messungen zeigen, dass der zusätzliche Aufwand, der durch die Struktur der Architektur verursacht wird, im Vergleich zu den Berechnungen der Spracherkennung sehr gering ist und folglich eine flüssige Interaktion erlaubt. Das System wurde außerdem auf dem IST-Event 2004 in Den Haag präsentiert. Dabei waren zwei BIRON-Roboter an drei Tagen insgesamt 24 Stunden lang im Einsatz und haben den Besuchern eine erfolgreiche Demonstration geliefert.

Insgesamt bietet das präsentierte Architekturkonzept eine hervorragende Basis zur Entwicklung eines *Robot Companions*. Zusammen mit der Integration der vorgestellten Komponenten ergibt sich bereits ein grundlegendes System zur natürlichen Interaktion zwischen Mensch und Roboter. Da die Architektur auf Erweiterbarkeit ausgelegt ist, können zusätzliche Funktionalitäten und Module auf einfache Weise dem System hinzugefügt werden. Dabei bedarf es zur Anbindung an den Execution Supervisor keiner Modifikation bestehender Komponenten, da allein die Konfigurationsdatei des zentralen Moduls angepasst werden muss. Die Generizität der Architektur wird insbesondere durch die Verwendung von XML ermöglicht. Diese Beschreibungssprache wird sowohl zur Konfiguration des Execution Supervisors als auch für die Kommunikation mit

allen angebundenen Modulen verwendet. Nur auf diese Weise ist es möglich, dass das bereits umfangreiche System auch in Zukunft noch um zahlreiche Komponenten erweitert werden kann.

Literaturverzeichnis

[Abe98] S. Abeck, A. Köppel, J. Seitz: *A Management Architecture for Multi-Agent Systems*, in *Proc. IEEE Int. Workshop on Systems Management*, Newport, Rhode Island, April 1998, S. 133–138.

[Aga00] A. Agah: *Human Interactions with Intelligent Systems: Research Taxonomy*, *Computers and Electrical Engineering*, Bd. 27, Nr. 1, Nov. 2000, S. 71–107.

[Ami01] E. Amir, P. Maynard-Reid II: *LiSA: A Robot Driven by Logical Subsumption*, in *Proc. Symp. on Logical Formalizations of Commonsense Reasoning (CommonSense)*, New York, NY, Mai 2001.

[Amp04] V. Ampornaramveth, P. Kiatisevi, H. Ueno: *SPAK: Software Platform for Agents and Knowledge Management in Symbiotic Robots*, *IEICE Trans. on Information and Systems*, Bd. E87-D, Nr. 4, April 2004, S. 886–895.

[And99] M. Andersson, A. Orebäck, M. Lindstrom, H. I. Christensen: *ISR: An Intelligent Service Robot*, in H. I. Christensen, H. Bunke, H. Noltmeier (Hrsg.): *Sensor Based Intelligent Robots; International Workshop Dagstuhl Castle, Germany, September 28 – October 2, 1998. Selected Papers*, Bd. 1724 von *Lecture Notes in Computer Science*, Springer-Verlag, New York, NY, 1999, S. 287–310.

[Ark86] R. C. Arkin: *Path Planning for a Vision-based Autonomous Robot*, in *Proc. SPIE Conf. on Mobile Robots*, Cambridge, MA, Okt. 1986, S. 240–249.

[Ark89] R. C. Arkin: *Motor Schema-Based Mobile Robot Navigation*, *Int. Journal of Robotics Research*, Bd. 8, Nr. 4, Aug. 1989, S. 92–112.

[Ark95] R. C. Arkin: *Reactive Robotic Systems*, in M. A. Arbib (Hrsg.): *The Handbook of Brain Theory and Neural Networks*, MIT Press, Cambridge, MA, 1995, S. 793–796.

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

[Ark03] R. C. Arkin, M. Fujita, T. T. R. Hasegawa: *An Ethological and Emotional Basis for Human-Robot Interaction*, *Robotics and Autonomous Systems*, Bd. 42, Nr. 3–4, März 2003, S. 191–201.

[Asf00] T. Asfour, K. Berns, R. Dillmann: *The Humanoid Robot ARMAR: Design and Control*, in *Proc. IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, MIT Press, Boston, MA, Sep. 2000.

[Aso01] H. Asoh, Y. Motomura, F. Asano, I. Hara, S. Hayamizu, K. Itou, T. Kurita, T. Matsui, N. Vlassis, R. Bunschoten, B. Kröse: *Jijo-2: An Office Robot that Communicates and Learns*, *IEEE Intelligent Systems*, Bd. 16, Nr. 5, Sep./Okt. 2001, S. 46–55.

[Bar95] Y. Bar-Shalom, X. Li: *Multitarget-Multisensor Tracking: Principles and Techniques*, YBS Publishing, 1.. Ausg., Jan. 1995.

[Bar01] C. Bartneck, M. Okada: *Robotic User Interfaces*, in *Proc. Int. Conf. on Human and Computer*, Aizu, Japan, Sep. 2001, S. 130–140.

[Bea03] M. J. Beal, N. Jojic, H. Attias: *A Graphical Model for Audiovisual Object Tracking*, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Bd. 25, Nr. 7, Juli 2003, S. 828–836.

[Ber02] B. Berdugo, J. Rosenhouse, H. Azhari: *Speakers' Direction Finding Using Estimated Time Delays in the Frequency Domain*, *Signal Processing*, Bd. 82, Nr. 1, Jan. 2002, S. 19–30.

[Bis99] R. Bischoff, T. Jain: *Natural Communication and Interaction with Humanoid Robots*, in *Proc. Int. Symp. on Humanoid Robots (HURO)*, Tokyo, Japan, Okt. 1999, S. 121–128.

[Bis02a] R. Bischoff, V. Graefe: *Demonstrating the Humanoid Robot HERMES at an Exhibition: A Long-Term Dependability Test*, in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems; Workshop on Robots at Exhibitions*, Lausanne, Switzerland, Sep./Okt. 2002.

[Bis02b] R. Bischoff, V. Graefe: *Dependable Multimodal Communication and Interaction with Robotic Assistants*, in *Proc. IEEE Int. Workshop on Robot-Human Interactive Communication (ROMAN)*, IEEE Press, Berlin, Germany, Sep. 2002, S. 300–305.

[Blo01] I. Bloch, A. Saffiotti: *Some Similarities Between Anchoring and Pattern Recognition Concepts*, in *Proc. AAAI Fall Symposium on Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems*, AAAI Fall Symposium Series, AAAI Press, North Falmouth, MA, Nov. 2001, S. 75–78.

[Blu03] S. A. Blum: *From a CORBA-Based Software Framework to a Component-Based System Architecture for Controlling a Mobile Robot*, in J. L. Crowley, J. H. Piater, M. Vincze, L. Paletta (Hrsg.): *Computer Vision Systems; Third International Conference, ICVS 2003, Graz, Austria, April 1–3, 2003. Proceedings*, Bd. 2626 von *Lecture Notes in Computer Science*, Springer-Verlag, Heidelberg, Germany, 2003, S. 333–344.

[Böh98] H.-J. Böhme, U.-D. Braumann, A. Brakensiek, A. Corradini, M. Krabbes, H.-M. Gross: *User Localisation for Visually-based Human-Machine-Interaction*, in *Proc. IEEE Int. Conf. on Automatic Face and Gesture Recognition (FG)*, IEEE Press, Nara, Japan, April 1998, S. 486–491.

[Böh03] H.-J. Böhme, T. Wilhelm, J. Key, C. Schauer, C. Schröter, H.-M. Groß, T. Hempel: *An Approach to Multi-modal Human-Machine Interaction for Intelligent Service Robots*, *Robotics and Autonomous Systems*, Bd. 44, Nr. 1, Juli 2003, S. 83–96.

[Bon97] R. P. Bonasso, R. J. Firby, E. Gat, D. Kortenkamp, D. P. Miller, M. G. Slack: *Experiences with an Architecture for Intelligent, Reactive Agents*, *Journal of Experimental and Theoretical Artificial Intelligence, Special issue on Software Architectures for Physical Agents*, Bd. 9, Nr. 2–3, April 1997, S. 237–256.

[Bor98] J. J. Borrelly, È. Coste-Manière, B. Espiau, K. Kapellos, R. Pissard-Gibollet, D. Simon, N. Turro: *The Orccad Architecture*, *Int. Journal of Robotics Research*, Bd. 17, Nr. 4, April 1998, S. 338–359.

[Bre99] C. Breazeal, B. Scassellati: *A Context-Dependent Attention System for a Social Robot*, in T. Dean (Hrsg.): *Proc. Int. Joint Conf. on Artificial Intelligence*, Bd. 2, Morgan Kaufmann Publishers Inc., Stockholm, Sweden, Juli/Aug. 1999, S. 1146—1151.

[Bre00] C. Breazeal, A. Edsinger, P. Fitzpatrick, B. Scassellati, P. Varchavskaia: *Social Constraints on Animate Vision*, *IEEE Intelligent Systems*, Bd. 15, Nr. 4, Juli/Aug. 2000, S. 32–37.

[Bro86] R. A. Brooks: *A Robust Layered Control System for a Mobile Robot*, *IEEE Journal of Robotics and Automation*, Bd. 2, Nr. 1, März 1986, S. 14–23.

[Bro90] R. A. Brooks: *Elephants Don't Play Chess*, *Robotics and Autonomous Systems*, Bd. 6, Nr. 1–2, Juni 1990, S. 3–15.

[Bro91] R. A. Brooks: *Intelligence Without Reason*, in J. Myopoulos, R. Reiter (Hrsg.): *Proc. Int. Joint Conf. on Artificial Intelligence*, Bd. 1, Morgan Kaufmann Publishers Inc., Sydney, Australia, Aug. 1991, S. 569–595.

[Bro99] R. A. Brooks, C. Breazeal, M. Marjanović, B. Scassellati, M. M. Williamson: *The Cog Project: Building a Humanoid Robot*, in C. L. Nehaniv (Hrsg.): *Computation for Metaphors, Analogy, and Agents*, Bd. 1562 von *Lecture Notes in Computer Science*, Springer-Verlag, New York, NY, 1999, S. 52–87.

[Bur98] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, S. Thrun: *The Interactive Museum Tour-Guide Robot*, in *Proc. Nat. Conf. on Artificial Intelligence (AAAI)*, AAAI/MIT Press, Madison, WI, Juli 1998, S. 11–18.

[Bur99] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, S. Thrun: *Experiences with an Interactive Museum Tour-Guide Robot, Artificial Intelligence*, Bd. 114, Nr. 1–2, Okt. 1999, S. 3–55.

[Cas00] J. Cassell, T. Bickmore, L. Campbell, H. Vilhjálmsson, H. Yan: *Human Conversation as a System Framework: Designing Embodied Conversational Agents*, in J. Cassell, J. Sullivan, S. Prevost, E. Churchill (Hrsg.): *Embodied Conversational Agents*, Kap. 2, MIT Press, Cambridge, MA, 2000, S. 29–63.

[Cic99] D. A. Ciccimaro, H. R. Everett, M. H. Bruch, C. B. Phillips: *A Supervised Autonomous Security Response Robot*, in *Proc. American Nuclear Society Int. Topical Meeting on Robotics and Remote Systems*, Pittsburgh, PA, April 1999.

[Cor00] S. Coradeschi, A. Saffiotti: *Anchoring Symbols to Sensor Data: preliminary report*, in L. Getoor, D. Jensen (Hrsg.): *Proc. Nat. Conf. on Artificial Intelligence (AAAI)*, AAAI/MIT Press, Austin, Texas, Juli 2000, S. 129–135.

[Cor01] S. Coradeschi, A. Saffiotti: *Perceptual Anchoring of Symbols for Action*, in B. Nebel (Hrsg.): *Proc. Int. Joint Conf. on Artificial Intelligence*, Bd. 1, Morgan Kaufmann Publishers Inc., Seattle, WA, Aug. 2001, S. 407–412.

[Cor03] S. Coradeschi, A. Saffiotti: *An Introduction to the Anchoring Problem, Robotics and Autonomous Systems, Special issue on Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems*, Bd. 43, Nr. 2–3, Mai 2003, S. 85–96.

[Cos00] È. Coste-Manière, R. G. Simmons: *Architecture, the Backbone of Robotic Systems*, in *Proc. IEEE Int. Conf. on Robotics and Automation*, Bd. 1, IEEE Press, San Franciso, CA, April 2000, S. 67–72.

[Dar00] T. Darrell, G. Gordon, M. Harville, J. Woodfill: *Integrated Person Tracking Using Stereo, Color, and Pattern Detection*, *Int. Journal of Computer Vision*, Bd. 37, Nr. 2, Juni 2000, S. 175–185.

[Dau04] K. Dautenhahn: *Robots We Like to Live With?! – A Developmental Perspective on a Personalized, Life-Long Robot Companion*, in *Proc. IEEE Int. Workshop on Robot-Human Interactive Communication (ROMAN)*, IEEE Press, Kurashiki, Okayama Japan, Sep. 2004, S. 17–22.

[Doi01] M. Doi, M. Nakakita, Y. Aoki, S. Hashimoto: *Real-time Vision System for Autonomous Mobile Robot*, in *Proc. IEEE Int. Workshop on Robot-Human Interactive Communication (ROMAN)*, IEEE Press, Bordeaux/Paris, France, Sep. 2001, S. 442–449.

[Doi02] M. Doi, K. Suzuki, S. Hashimoto: *Integrated Communicative Robot "BUGNOID"*, in *Proc. IEEE Int. Workshop on Robot-Human Interactive Communication (ROMAN)*, IEEE Press, Berlin, Germany, Sep. 2002, S. 259–265.

[Eli02] P. Elinas, J. Hoey, D. Lahey, J. Montgomery, D. Murray, S. Se, J. J. Little: *Waiting with José, a vision-based mobile robot*, in *Proc. IEEE Int. Conf. on Robotics and Automation*, Bd. 4, IEEE Press, Washington DC, USA, Mai 2002, S. 3698–3705.

[Fey00] S. Feyrer, A. Zell: *Robust Real-Time Pursuit of Persons with a Mobile Robot Using Multisensor Fusion*, in E. Pagello, F. Groen, T. Arai, R. Dillman, A. Stentz (Hrsg.): *Proc. Int. Conf. on Intelligent Autonomous Systems*, IOS Press, Venice, Italy, Juli 2000, S. 710–715.

[Fin95] G. A. Fink, N. Jungclaus, H. Ritter, G. Sagerer: *A Communication Framework for Heterogeneous Distributed Pattern Analysis*, in V. L. Narasimhan (Hrsg.): *Proc. Int. Conf. on Algorithms and Architectures for Parallel Processing*, IEEE Press, Brisbane, Australia, April 1995, S. 881–890.

[Fin99] G. A. Fink: *Developing HMM-based Recognizers with ESMERALDA*, in V. Matoušek, P. Mautner, J. Ocelíková, P. Sojka (Hrsg.): *Text, Speech and Dialogue; Second International Workshop, TSD'99, Plzen, Czech Republic, September 1999. Proceedings*, Bd. 1692 von *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Heidelberg, Germany, Sep. 1999, S. 229–234.

[Fin04] G. A. Fink, J. Fritsch, S. Hohenner, M. Kleinehagenbrock, S. Lang, G. Sagerer: *Towards Multi-Modal Interaction with a Mobile Robot*, *Pattern Recognition and Image Analysis*, Bd. 14, Nr. 2, 2004, S. 173–184.

[Fir89] R. J. Firby: *Adaptive Execution in Complex Dynamic Worlds*, Technical Report, YALEU/CSD/RR #672, Yale University, New Haven, CT, Jan. 1989.

[Fon01] T. W. Fong, C. Thorpe, C. Baur: *Collaboration, Dialogue, and Human-Robot Interaction*, in *Proc. Int. Symp. on Robotics Research*, Springer-Verlag, Lorne, Victoria, Australia, Nov. 2001.

[Fon03a] T. W. Fong, I. Nourbakhsh, K. Dautenhahn: *A survey of socially interactive robots*, *Robotics and Autonomous Systems*, Bd. 42, Nr. 3–4, März 2003, S. 143–166.

[Fon03b] T. W. Fong, C. Thorpe, C. Baur: *Robot, Asker of Questions*, *Robotics and Autonomous Systems*, Bd. 42, Nr. 3–4, März 2003, S. 235–243.

[Fra97] S. Franklin, A. Graesser: *Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents*, in J. P. Müller, M. Wooldridge, N. R. Jennings (Hrsg.): *Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages, August 12 - 13, 1996*, Bd. 1193 von *Lecture Notes in Computer Science*, Springer-Verlag, London, UK, 1997, S. 21–35.

[Fre97] Y. Freund, R. E. Schapire: *A decision-theoretic generalization of on-line learning and an application to boosting*, *Journal of Computer and System Sciences, Special issue: 26th annual ACM symposium on the theory of computing & STOC'94, May 23–25*,

1994, and second annual Europe an conference on computational learning theory (EuroCOLT'95), March 13–15, 1995, Bd. 55, Nr. 1, Aug. 1997, S. 119–139.

[Fri02] J. Fritsch, S. Lang, M. Kleinehagenbrock, G. A. Fink, G. Sagerer: *Improving Adaptive Skin Color Segmentation by Incorporating Results from Face Detection,* in *Proc. IEEE Int. Workshop on Robot-Human Interactive Communication (ROMAN)*, IEEE Press, Berlin, Germany, Sep. 2002, S. 337–343.

[Fri03] J. Fritsch, M. Kleinehagenbrock, S. Lang, T. Plötz, G. A. Fink, G. Sagerer: *Multi-Modal Anchoring for Human-Robot-Interaction, Robotics and Autonomous Systems, Special issue on Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems*, Bd. 43, Nr. 2–3, Mai 2003, S. 133–147.

[Fri04] J. Fritsch, M. Kleinehagenbrock, S. Lang, G. A. Fink, G. Sagerer: *Audiovisual Person Tracking with a Mobile Robot,* in F. Groen, N. Amato, A. Bonarini, E. Yoshida, B. Kröse (Hrsg.): *Proc. Int. Conf. on Intelligent Autonomous Systems*, IOS Press, Amsterdam, The Netherlands, März 2004, S. 898–906.

[Fri05] J. Fritsch, M. Kleinehagenbrock, A. Haasch, S. Wrede, G. Sagerer: *A Flexible Infrastructure for the Development of a Robot Companion with Extensible HRI-Capabilities*, in *Proc. IEEE Int. Conf. on Robotics and Automation*, IEEE Press, Barcelona, Spain, April 2005, S. 3419–3425.

[Gat92] E. Gat: *Integrating Planning and Reacting in a Heterogenous Asynchronous Architecture for Controlling Real-World Mobile Robots,* in W. R. Swartout (Hrsg.): *Proc. Nat. Conf. on Artificial Intelligence (AAAI)*, AAAI/MIT Press, San Jose, CA, Juli 1992, S. 809–815.

[Gat98] E. Gat: *On Three-Layer Architectures,* in D. KortenKamp, R. P. Bonasso, R. Murphy (Hrsg.): *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems,* Kap. 8, MIT Press, Cambridge, MA, 1998, S. 195–210.

[Ger03] B. P. Gerkey, R. T. Vaughan, A. Howard: *The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems,* in *Proc. Int. Conf. on Advanced Robotics*, Coimbra, Portugal, Juni/Juli 2003, S. 317–323.

[Ghi00a] S. S. Ghidary, Y. Nakata, T. Takamori, M. Hattori: *Head and Face Detection at Indoor Environment by Home Robot,* in *Proc. Iranian Conference on Electrical Engineering*, Isfahan, Iran, Mai 2000.

[Ghi00b] S. S. Ghidary, Y. Nakata, T. Takamori, M. Hattori: *Human Detection and Localization at Indoor Environment by Home Robot,* in *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics*, Nashville, Tennessee, Okt. 2000, S. 1360–1365.

[Ghi02] S. S. Ghidary, Y. Nakata, H. Saito, M. Hattori, T. Takamori: *Multi-Modal Interaction of Human and Home Robot in the Context of Room Map Generation, Autonomous Robots*, Bd. 13, Nr. 2, Sep. 2002, S. 169–184.

[Giu94] D. Giuliani, M. Omologo, P. Svaizer: *Talker Localization and Speech Recognition using a Microphone Array and a Cross-Powerspectrum Phase Analysis*, in *Proc. Int. Conf. on Spoken Language Processing*, Bd. 3, Yokohama, Japan, Sep. 1994, S. 1243–1246.

[Goo95] R. Goodwin: *Formalizing Properties of Agents*, *Journal of Logic and Computation*, Bd. 5, Nr. 6, Dez. 1995, S. 763–781.

[Gra98] V. Graefe: *Perception and Situation Assessment for Behavior-Based Robot Control*, in Y. Kakazu, M. Wada, T. Sato (Hrsg.): *Proc. Int. Conf. on Intelligent Autonomous Systems*, IOS Press, Sapporo, Japan, Juni 1998, S. 376–383.

[Gra04] B. Graf, M. Hans, R. D. Schraft: *Care-O-bot II—Development of a Next Generation Robotic Home Assistant*, *Autonomous Robots*, Bd. 16, Nr. 2, März 2004, S. 193–205.

[Gug99] E. Guglielmelli, C. Laschi, P. Dario: *Robots for Personal Use: Humanoids vs. Distributed Systems*, in *Proc. Int. Symp. on Humanoid Robots (HURO)*, Tokyo, Japan, Okt. 1999.

[Guo99] D. Guo, X. M. Yin, Y. Jin, M. Xie: *Efficient Gesture Interpretation for Gesture-based Human-Service Robot Interaction*, in *Proc. Int. Conf. on Field and Service Robotics*, Pittsburgh, PA, Aug. 1999.

[Haa03] A. Haasch: *Extraktion von 3D-Objektinformationen aus korrelierten Kamera- und Laser-Daten*, Diplomarbeit, Universität Bielefeld, Technische Fakultät, Angewandte Informatik, Bielefeld, Deutschland, Aug. 2003.

[Haa04] A. Haasch, S. Hohenner, S. Hüwel, M. Kleinehagenbrock, S. Lang, I. Toptsis, G. A. Fink, J. Fritsch, B. Wrede, G. Sagerer: *BIRON – The Bielefeld Robot Companion*, in E. Prassler, G. Lawitzky, P. Fiorini, M. Hägele (Hrsg.): *Proc. Int. Workshop on Advances in Service Robotics*, Fraunhofer IRB Verlag, Stuttgart, Germany, Mai 2004, S. 27–32.

[Haa05] A. Haasch, N. Hofemann, J. Fritsch, G. Sagerer: *A Multi-Modal Object Attention System for a Mobile Robot*, in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Edmonton, AB, Aug. 2005, in Vorbereitung.

[Han01] M. Hans, W. Baum: *Concept of a Hybrid Architecture for Care-O-bot*, in *Proc. IEEE Int. Workshop on Robot-Human Interactive Communication (ROMAN)*, IEEE Press, Bordeaux/Paris, France, Sep. 2001, S. 407–411.

[Har01] I. Haritaoglu, A. Cozzi, D. Koons, M. Flickner, D. Zotkin, Y. Yacoob: *Attentive Toys*, in *Proc. IEEE Int. Conf. on Multimedia and Expo*, IEEE Press, Tokyo, Japan, Aug. 2001, S. 1124–1127.

[Has02] S. Hashimoto, S. Narita, H. Kasahara, K. Shirai, T. Kobayashi, A. Takanishi, S. Sugano, J. Yamaguchi, H. Sawada, H. Takanobu, K. Shibuya, T. Morita, T. Kurata, N. Onoe,

K. Ouchi, T. Noguchi, Y. Niwa, S. Nagayama, H. Tabayashi, I. Matsui, M. Obata, H. Matsuzaki, A. Murasugi, T. Kobayashi, S. Haruyama, T. Okada, Y. Hidaki, Y. Taguchi, K. Hoashi, E. Morikawa, Y. Iwano, D. Araki, J. Suzuki, M. Yokoyama, I. Dawa, D. Nishino, S. Inoue, T. Hirano, E. Soga, S. Gen, T. Yanada, K. Kato, S. Sakamoto, Y. Ishii, S. Matsuo, Y. Yamamoto, K. Sato, T. Hagiwara, T. Ueda, N. Honda, K. Hashimoto, T. Hanamoto, S. Kayaba, T. Kojima, H. Iwata, H. Kubodera, R. Matsuki, T. Nakajima, K. Nitto, D. Yamamoto, Y. Kamizaki, S. Nagaike, Y. Kunitake, S. Morita: *Humanoid Robots in Waseda University—Hadaly-2 and WABIAN*, *Autonomous Robots*, Bd. 12, Nr. 1, Jan. 2002, S. 25–38.

[Hay85] B. Hayes-Roth: *A Blackboard Architecture for Control*, *Artificial Intelligence*, Bd. 26, Nr. 3, Juli 1985, S. 251–321.

[Heb95] M. Hebert, J. Ponce, T. E. Boult, A. D. Gross (Hrsg.): *Object Representation in Computer Vision; International NSF-ARPA Workshop, New York City, NY, USA, December 5–7, 1994. Proceedings*, Bd. 994 von *Lecture Notes in Computer Science*, Springer-Verlag, Heidelberg, Germany, Okt. 1995.

[Hir98] K. Hirai, M. Hirose, Y. Haikawa, T. Takenaka: *The Development of Honda Humanoid Robot*, in *Proc. IEEE Int. Conf. on Robotics and Automation*, Bd. 2, IEEE Press, Leuven, Belgium, Mai 1998, S. 1321–1326.

[Hje01] E. Hjelmås, B. K. Low: *Face Detection: A Survey*, *Computer Vision and Image Understanding*, Bd. 83, Nr. 3, Sep. 2001, S. 236–274.

[Hof04] N. Hofemann, J. Fritsch, G. Sagerer: *Recognition of Deictic Gestures with Context*, in C. E. Rasmussen, H. H. Bülthoff, M. A. Giese, B. Schölkopf (Hrsg.): *Pattern Recognition; 26th DAGM Symposium, Tübingen, Germany, August/September 2004. Proceedings*, Bd. 3175 von *Lecture Notes in Computer Science*, Springer-Verlag, Heidelberg, Germany, 2004, S. 334–341.

[Hoh03] S. Hohenner, S. Lang, M. Kleinehagenbrock, G. A. Fink, F. Kummert: *Multimodale Sprecherlokalisation für Mensch-Roboter-Interaktionen in einer Multi-Personen-Umgebung*, in K. Kroschel (Hrsg.): *Elektronische Sprachsignalverarbeitung: Tagungsband der 14. Konferenz Karlsruhe, 24. bis 26. September 2003*, Bd. 28 von *Studientexte zur Sprachkommunikation*, Karlsruhe, Deutschland, Sep. 2003, S. 162–169.

[Hor93] I. Horswill: *Polly: A Vision-Based Artificial Agent*, in *Proc. Nat. Conf. on Artificial Intelligence (AAAI)*, AAAI/MIT Press, Washington DC, USA, Juli 1993, S. 824–829.

[Hua01] Y. Huang, J. Benesty, G. W. Elko, R. M. Mersereau: *Real-Time Passive Source Localization: A Practical Linear-Correction Least-Squares Approach*, *IEEE Trans. on Speech and Audio Processing*, Bd. 9, Nr. 8, Nov. 2001, S. 943–956.

[Hub99] M. J. Huber: *JAM: A BDI-Theoretic Mobile Agent Architecture*, in *Proc. Int. Conf. on Autonomous Agents (Agents)*, Seattle, WA, Mai 1999, S. 236–243.

[Ima01] M. Imai, T. Ono, H. Ishiguro: *Physical Relation and Expression: Joint Attention for Human-Robot Interaction*, in *Proc. IEEE Int. Workshop on Robot-Human Interactive Communication (ROMAN)*, IEEE Press, Bordeaux/Paris, France, Sep. 2001, S. 512–517.

[Ish99] H. Ishiguro, T. Kanda, K. Kimoto, T. Ishida: *A Robot Architecture Based on Situated Modules*, in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Bd. 3, Kyongju, Korea, Okt. 1999, S. 1617–1623.

[Jen98] R. M. Jensen, M. M. Veloso: *Interleaving Deliberative and Reactive Planning in Dynamic Multi-Agent Domains*, in *Proc. AAAI Symp. on Integrated Planning for Autonomous Agent Architectures*, AAAI Fall Symposium Series, AAAI/MIT Press, Orlando, FL, Okt. 1998.

[Kah96] R. E. Kahn, M. J. Swain, P. N. Prokopowicz, R. J. Firby: *Gesture Recognition Using the Perseus Architecture*, in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, IEEE Computer Society, San Francisco, CA, Juni 1996, S. 734–741.

[Kan04] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, T. Isozumi: *Humanoid Robot HRP-2*, in *Proc. IEEE Int. Conf. on Robotics and Automation*, Bd. 2, IEEE Press, New Orleans, LA, April/Mai 2004, S. 1083–1090.

[Kap00] F. Kaplan: *Talking AIBO: First Experimentation of Verbal Interactions with an Autonomous Four-legged Robot*, in A. Nijholt, D. Heylen, K. Jokinen (Hrsg.): *Proc. of the CELE-Twente Workshop on Interacting Agents*, Workshop series on (Human-)Agent Interaction and Agent Learning, Enschede, The Netherlands, Okt. 2000, S. 57–63.

[Kaw00a] K. Kawamura, W. A. Alford, K. Hambuchen, D. M. Wilkes: *Towards a Unified Framework for Human-Humanoid Interaction*, in *Proc. IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, MIT Press, Boston, MA, Sep. 2000.

[Kaw00b] K. Kawamura, R. A. Peters II, D. M. Wilkes, W. A. Alford, T. E. Rogers: *ISAC: Foundations in Human-Humanoid Interaction*, *IEEE Intelligent Systems*, Bd. 15, Nr. 4, Juli/Aug. 2000, S. 38–45.

[Kia04] P. Kiatisevi, V. Ampornaramveth, H. Ueno: *A Distributed Architecture for Knowledge-Based Interactive Robots*, in *Proc. Int. Conf. on Information Technology for Application (ICITA)*, Harbin, China, Jan. 2004, S. 256–261.

[Kid03] C. D. Kidd: *Sociable Robots: The Role of Presence and Task in Human-Robot Interaction*, Diplomarbeit, Massachusetts Institute of Technology, The Media Laboratory, Cambridge, MA, Juni 2003.

[Kim02] K. Kim, Y. Matsusaka, T. Kobayashi: *Inter-Module Cooperation Architecture for Interactive Robot*, in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Bd. 3, Lausanne, Switzerland, Sep./Okt. 2002, S. 2286–2291.

[Kim03] G. Kim, W. Chung, M. Kim, C. Lee: *Tripodal Schematic Design of the Control Architecture for the Service Robot PSR,* in *Proc. IEEE Int. Conf. on Robotics and Automation*, Bd. 2, IEEE Press, Taipei, Taiwan, Sep. 2003, S. 2792–2797.

[Kim04a] G. Kim, W. Chung, S. Han, K.-R. Kim, M. Kim, R. H. Shinn: *The Autonomous Tour-Guide Robot Jinny*, in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Bd. 3, Sendai, Japan, Sep./Okt. 2004, S. 3450–3455.

[Kim04b] G. Kim, W. Chung, M. Kim, C. Lee: *Implementation of Multi-Functional Service Robots Using Tripodal Schematic Control Architecture*, in *Proc. IEEE Int. Conf. on Robotics and Automation*, Bd. 4, New Orleans, LA, April 2004, S. 4005–4010.

[Kle02] M. Kleinehagenbrock, S. Lang, J. Fritsch, F. Lömker, G. A. Fink, G. Sagerer: *Person Tracking with a Mobile Robot based on Multi-Modal Anchoring*, in *Proc. IEEE Int. Workshop on Robot-Human Interactive Communication (ROMAN)*, IEEE Press, Berlin, Germany, Sep. 2002, S. 423–429.

[Kle04] M. Kleinehagenbrock, J. Fritsch, G. Sagerer: *Supporting Advanced Interaction Capabilities on a Mobile Robot with a Flexible Control System*, in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Bd. 3, Sendai, Japan, Sep./Okt. 2004, S. 3649–3655.

[Kna76] C. H. Knapp, G. C. Carter: *The Generalized Correlation Method for Estimation of Time Delay*, *IEEE Trans. on Acoustics, Speech and Signal Processing*, Bd. 24, Nr. 4, Aug. 1976, S. 320–327.

[Kno04] S. Knoop, S. Vacek, R. Zöllner, C. Au, R. Dillmann: *A CORBA-Based Distributed Software Architecture for Control of Service Robots*, in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Bd. 3, Sendai, Japan, Sep./Okt. 2004, S. 3656–3661.

[Kop03] S. Kopp, B. Jung, N. Leßmann, I. Wachsmuth: *Max – A Multimodal Assistant in Virtual Reality Construction*, *KI - Künstliche Intelligenz, Special issue on Embodied Conversational Agents*, Bd. 17, Nr. 4, 2003, S. 11–17.

[Kor96] D. Kortenkamp, E. Huber, R. P. Bonasso: *Recognizing and Interpreting Gestures on a Mobile Robot*, in *Proc. Nat. Conf. on Artificial Intelligence (AAAI)*, Bd. 2, AAAI/MIT Press, Portland, OR, Aug. 1996, S. 915–921.

[Kri01] S. Kristensen, S. Horstmann, J. Klandt, F. Lohnert, A. Stopp: *Human-Friendly Interaction for Learning and Cooperation*, in *Proc. IEEE Int. Conf. on Robotics and Automation*, Bd. 3, IEEE Press, Seoul, Korea, Mai 2001, S. 2590–2595.

[Krö03] B. J. A. Kröse, J. M. Porta, A. J. N. van Breemen, K. Crucq, M. Nuttin, E. Demeester: *Lino, the User-Interface Robot*, in E. H. L. Aarts, R. Collier, E. van Loenen, B. E. R. de Ruyter (Hrsg.): *Ambient Intelligence; First European Symposium, EUSAI 2003, Veldhoven, The Netherlands, November 3–4, 2003. Proceedings*, Bd. 2875 von *Lecture Notes in Computer Science*, Springer-Verlag, Heidelberg, Germany, 2003, S. 264–274.

[Kul02] V. Kulyukin, A. Steele: *Instruction and Action in the Three-Tiered Robot Architecture*, in *Proc. Int. Symp. on Robotics and Automation*, IEEE Press, Toluca, Mexico, Sep. 2002, S. 578–583.

[Lan02] S. Lang, M. Kleinehagenbrock, J. Fritsch, G. A. Fink, G. Sagerer: *Detection of Communication Partners from a Mobile Robot*, in R. P. Würtz, M. Lappe (Hrsg.): *Proc. Workshop on Dynamic Perception*, Akademische Verlagsgesellschaft Aka GmbH, Berlin, Bochum, Germany, Nov. 2002, S. 183–188.

[Lan03] S. Lang, M. Kleinehagenbrock, S. Hohenner, J. Fritsch, G. A. Fink, G. Sagerer: *Providing the Basis for Human-Robot-Interaction: A Multi-Modal Attention System for a Mobile Robot*, in *Proc. Int. Conf. on Multimodal Interfaces*, ACM Press, Vancouver, Canada, Nov. 2003, S. 28–35.

[Lan05] S. Lang: *Multimodale Aufmerksamkeitssteuerung für mobile Systeme*, Dissertation, Universität Bielefeld, Technische Fakultät, Angewandte Informatik, Bielefeld, Deutschland, Mai 2005.

[Lay01] K. Lay, E. Prassler, R. Dillmann, G. Grunwald, M. Hägele, G. Lawitzky, A. Stopp, W. von Seelen: *MORPHA: Communication and Interaction with Intelligent, Anthropomorphic Robot Assistants*, in *Proc. Int. Status Conf. of the German Lead Projects in Human-Computer-Interaction*, Saarbrücken, Germany, Okt. 2001, S. 67–77.

[Lee90] C. C. Lee: *Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Parts I & II*, *IEEE Trans. on Systems, Man, and Cybernetics*, Bd. 20, Nr. 2, März/April 1990, S. 404–435.

[Lem01] O. Lemon, A. Bracy, A. Gruenstein, S. Peters: *The WITAS Multi-Modal Dialogue System I*, in *Proc. Europ. Conf. on Speech Communication and Technology (Eurospeech)*, Aalborg, Denmark, Sep. 2001, S. 1559–1562.

[Li04] S. Li, M. Kleinehagenbrock, J. Fritsch, B. Wrede, G. Sagerer: *"BIRON, let me show you something": Evaluating the Interaction with a Robot Companion*, in W. Thissen, P. Wieringa, M. Pantic, M. Ludema (Hrsg.): *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics, Special Session on Human-Robot Interaction*, IEEE, The Hague, The Netherlands, Okt. 2004, S. 2827–2834.

[Lim00] H.-O. Lim, A. Takanishi: *Waseda Biped Humanoid Robots Realizing Human-like Motion*, in *Proc. Int. Workshop on Advanced Motion Control*, IEEE Press, Nagoya, Japan, März/April 2000, S. 525–530.

[Lin00] M. Lindström, A. Orebäck, H. I. Christensen: *BERRA: A Research Architecture for Service Robots*, in *Proc. IEEE Int. Conf. on Robotics and Automation*, Bd. 4, IEEE Press, San Francisco, CA, April 2000, S. 3278–3283.

[Lis03] B. Lisien, D. Morales, D. Silver, G. Kantor, I. M. Rekleitis, H. Choset: *Hierarchical Simultaneous Localization and Mapping*, in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Bd. 1, Las Vegas, NV, Okt. 2003, S. 448–453.

[Löm04] F. Lömker: *Lernen von Objektbenennungen mit visuellen Prozessen*, Dissertation, Universität Bielefeld, Technische Fakultät, Angewandte Informatik, Bielefeld, Deutschland, Mai 2004.

[Lyo92] D. M. Lyons, A. J. Hendriks: *Planning for Reactive Robot Behavior*, in *Proc. IEEE Int. Conf. on Robotics and Automation*, IEEE Press, Nice, France, Mai 1992, S. 2675–2680.

[Mac96] D. C. MacKenzie: *A Design Methodology for the Configuration of Behavior-Based Mobile Robots*, Dissertation, Georgia Institute of Technology, College of Computing, Atlanta, GA, 1996.

[Mae89] P. Maes: *How to Do the Right Thing*, *Journal of Connection Science, Special issue on Hybrid Systems*, Bd. 1, Nr. 3, Dez. 1989, S. 291–323.

[Mat92] M. J. Matarić: *Behavior-Based Systems: Main Properties and Implications*, in *Proc. IEEE Int. Conf. on Robotics and Automation; Workshop on Architectures for Intelligent Control System*, IEEE Press, Nice, France, Mai 1992, S. 46–54.

[Mat97] M. J. Matarić: *Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior*, *Journal of Experimental and Theoretical Artificial Intelligence, Special issue on Software Architectures for Physical Agents*, Bd. 9, Nr. 2–3, April 1997, S. 323–336.

[Mat98] V. Matellán, D. Borrajo: *Combining Classical and Reactive Planning: the ABC^2 Model*, in R. Bergmann, A. Kott (Hrsg.): *Proc. Int. Conf. on Artificial Intelligence in Planning Systems; Workshop on Integrating Planning, Scheduling and Execution in Dynamic and Uncertain Environments*, AAAI/MIT Press, Pittsburgh, PA, Juni 1998, S. 121–126.

[Mat99a] T. Matsui, H. Asoh, J. Fry, Y. Motomura, F. Asano, T. Kurita, I. Hara, N. Otsu: *Integrated Natural Spoken Dialogue System of Jijo-2 Mobile Robot for Office Services*, in *Proc. Nat. Conf. on Artificial Intelligence (AAAI)*, AAAI/MIT Press, Orlando, FL, Juli 1999, S. 621–627.

[Mat99b] Y. Matsusaka, T. Tojo, S. Kubota, K. Furukawa, D. Tamiya, K. Hayata, Y. Nakano, T. Kobayashi: *Multi-person Conversation via Multi-modal Interface — A Robot who Communicate with Multi-user —*, in *Proc. Europ. Conf. on Speech Communication and Technology (Eurospeech)*, Bd. 4, Budapest, Hungary, Sep. 1999, S. 1723–1726.

[Mat01a] M. J. Matarić: *Learning in Behavior-Based Multi-Robot Systems: Policies, Models, and Other Agents*, *Cognitive Systems Research, Special issue on Multi-disciplinary studies of multi-agent learning*, Bd. 2, Nr. 1, April 2001, S. 81–93.

[Mat01b] Y. Matsusaka, S. Fujie, T. Kobayashi: *Modeling of Conversational Strategy for the Robot Participating in the Group Conversation*, in P. Dalsgaard, B. Lindberg, H. Benner, Z. Tan (Hrsg.): *Proc. Europ. Conf. on Speech Communication and Technology (Eurospeech)*, Bd. 3, Aalborg, Denmark, Sep. 2001, S. 2173–2176.

[McK99] S. J. McKenna, Y. Raja, S. Gong: *Tracking Colour Objects using Adaptive Mixture Models, Image and Vision Computing*, Bd. 17, Nr. 3–4, März 1999, S. 225–231.

[Mic99] F. Michaud: *Emergent behavior for real world robots,* in *Proc. IIIS/IFSR/IEEE World Multiconf. on Systemics, Cybernetics and Informatics*, IIIS, Orlando, FL, Juli 1999, S. 402–408.

[Mil92] D. P. Miller, R. S. Desai, E. Gat, R. Ivlev, J. Loch: *Reactive Navigation Through Rough Terrain: Experimental Results,* in *Proc. Nat. Conf. on Artificial Intelligence (AAAI)*, AAAI Press, San Jose, CA, Juli 1992, S. 823–828.

[Miu03] J. Miura, Y. Shirai, N. Shimada, Y. Makihara, M. Takizawa, Y. Yano: *Development of a Personal Service Robot with User-Friendly Interfaces,* in *Proc. Int. Conf. on Field and Service Robotics*, Lake Yamanaka, Japan, Juli 2003, S. 293–298.

[Möl05] B. Möller, S. Posch, A. Haasch, J. Fritsch, G. Sagerer: *Interactive Object Learning for Robot Companions using Mosaic Images,* in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Edmonton, AB, Aug. 2005, in Vorbereitung.

[Mon02] M. Montemerlo, J. Pineau, N. Roy, S. Thrun, V. Verma: *Experiences with a Mobile Robotic Guide for the Elderly,* in *Proc. Nat. Conf. on Artificial Intelligence (AAAI)*, AAAI/MIT Press, Edmonton, AB, Juli 2002, S. 587–592.

[Mur95] H. Murase, S. K. Nayar: *Visual Learning and Recognition of 3D Objects from Appearance, Int. Journal of Computer Vision*, Bd. 14, Nr. 1, Jan. 1995, S. 5–24.

[Nak00] K. Nakadai, T. Lourens, H. G. Okuno, H. Kitano: *Active Audition for Humanoid,* in *Proc. Nat. Conf. on Artificial Intelligence (AAAI)*, Austin, Texas, Aug. 2000, S. 832–839.

[Nak01] K. Nakadai, K. Hidai, H. Mizoguchi, H. G. Okuno, H. Kitano: *Real-Time Auditory and Visual Multiple-Object Tracking for Humanoids,* in B. Nebel (Hrsg.): *Proc. Int. Joint Conf. on Artificial Intelligence*, Bd. 2, Morgan Kaufmann Publishers Inc., Seattle, WA, Aug. 2001, S. 1425–1432.

[Nak02] K. Nakadai, H. G. Okuno, H. Kitano: *Real-Time Sound Source Localization and Separation for Robot Audition,* in *Proc. Int. Conf. on Spoken Language Processing*, Denver, CO, Sep. 2002, S. 193–196.

[Nev98] M. C. Neves, E. Oliveira: *ARCoS - An Autonomous Mobile Robot Control System,* in *Proc. ICSC Int. Symp. on Engineering of Intelligent Systems*, ISCS Academic Press, Tenerife, Spain, Feb. 1998, S. 220–227.

[Nic02] M. Nicolescu, M. J. Matarić: *A Hierarchical Architecture for Behavior-Based Robots,* in *Proc. Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, ACM Press, Bologna, Italy, Juli 2002, S. 227–233.

[Nil82] N. J. Nilsson: *Principles of Artificial Intelligence*, Springer-Verlag, Berlin, Germany, 1982.

[Nil84] N. J. Nilsson: *Shakey the Robot*, Technical Note 323, AI Center, SRI International, Menlo Park, CA, 1984.

[Nou99] I. Nourbakhsh, J. Bobenage, S. Grange, R. Lutz, R. Meyer, A. Soto: *An Affective Mobile Educator with a Full-time Job*, *Artificial Intelligence*, Bd. 114, Nr. 1–2, Okt. 1999, S. 95–124.

[Oku01] H. G. Okuno, K. Nakadai, K. Hidai, H. Mizoguchi, H. Kitano: *Human-Robot Interaction through Real-Time Auditory and Visual Multiple-Talker Tracking*, in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Bd. 3, Maui, Hawaii, Okt./Nov. 2001, S. 1402–1409.

[Oku02] H. G. Okuno, K. Nakadai, H. Kitano: *Social Interaction of Humanoid Robot Based on Audio-Visual Tracking*, in T. Hendtlass, M. Ali (Hrsg.): *Proc. Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE)*, Bd. 2358 von *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Cairns, Australia, Juni 2002, S. 725–734.

[Oli00] N. Oliver, A. Pentland, F. Bérard: *LAFTER: a real-time face and lips tracker with facial expression recognition*, *Pattern Recognition*, Bd. 33, Nr. 8, Aug. 2000, S. 1369–1382.

[Ore03] A. Orebäck, H. I. Christensen: *Evaluation of Architectures for Mobile Robotics*, *Autonomous Robots*, Bd. 14, Nr. 1, Jan. 2003, S. 33–49.

[Ore04] A. Orebäck: *A Component Framework for Autonomous Mobile Robots*, Dissertation, KTH Stockholm, Stockholm, Sweden, Nov. 2004.

[Par01] H. K. Park, H. S. Hong, H. J. Kwon, M. J. Chung: *A Nursing Robot System for the Elderly and the Disabled*, in *Proc. Int. Workshop on Human-friendly Welfare Robotic Systems*, Daejeon, Korea, Jan. 2001, S. 122–126.

[Pav00a] V. Pavlović, A. Garg, J. Rehg, T. Huang: *Multimodal Speaker Detection using Error Feedback Dynamic Bayesian Networks*, in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Bd. 2, IEEE Press, Hilton Head Island, SC, Juni 2000, S. 34–41.

[Pav00b] V. Pavlović, A. Garg, J. M. Rehg: *Multimodal Speaker Detection Using Input/Output Dynamic Bayesian Networks*, in T. Tan, Y. Shi, W. Gao (Hrsg.): *Advances in Multimodal Interfaces - ICMI 2000; Third International Conference, Beijing, China, October 2000. Proceedings*, Bd. 1948 von *Lecture Notes in Computer Science*, Springer-Verlag, Heidelberg, Germany, 2000, S. 308–316.

[Per99] D. Perzanowski, A. C. Schultz, W. Adams, E. Marsh: *Goal Tracking in a Natural Language Interface: Towards Achieving Adjustable Autonomy*, in *Proc. IEEE Int. Symp.*

on Computational Intelligence in Robotics and Automation (CIRA), IEEE Press, Monterey, CA, Nov. 1999, S. 208–213.

[Pet99] R. A. Peters II, D. M. Wilkes, D. M. Gaines, K. Kawamura: *A Software Agent Based Control System for Human-Robot Interaction*, in *Proc. Int. Symp. on Humanoid Robots (HURO)*, Tokyo, Japan, Okt. 1999.

[Pet01] L. Petersson, D. J. Austin, H. I. Christensen: *DCA: A Distributed Control Architecture for Robotics*, in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Bd. 3, Maui, Hawaii, Okt./Nov. 2001, S. 2361–2368.

[Pir99] P. Pirjanian, M. J. Matarić: *A Decision-Theoretic Approach to Fuzzy Behavior Coordination*, in *Proc. IEEE Int. Symp. on Computational Intelligence in Robotics and Automation (CIRA)*, IEEE Press, Monteray, CA, Nov. 1999, S. 101–106.

[Pon96] J. Ponce, A. Zisserman, M. Hebert (Hrsg.): *Object Representation in Computer Vision II; ECCV '96 International Workshop, Cambridge, UK, April 13–14, 1996. Proceedings*, Bd. 1144 von *Lecture Notes in Computer Science*, Springer-Verlag, Heidelberg, Germany, Sep. 1996.

[Rog02] O. Rogalla, M. Ehrenmann, R. Zöllner, R. Becher, R. Dillmann: *Using Gesture and Speech Control for Command a Robot Assistant*, in *Proc. IEEE Int. Workshop on Robot-Human Interactive Communication (ROMAN)*, IEEE Press, Berlin, Germany, Sep. 2002, S. 454–459.

[Ros95] J. K. Rosenblatt: *DAMN: A Distributed Architecture for Mobile Navigation*, in H. Hexmoor, D. Kortenkamp (Hrsg.): *Proc. AAAI Spring Symp. on Lessons Learned from Implememted Software Architectures for Physical Agents*, AAAI/MIT Press, Stanford, CA, März 1995, S. 167–178.

[Ros03] M. Rosencrantz, G. Gordon, S. Thrun: *Locating Moving Entities in Indoor Environments with Teams of Mobile Robots*, in *Proc. Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems*, ACM Press, Melbourne, Australia, Juli 2003, S. 233–240.

[Row98] H. A. Rowley, S. Baluja, T. Kanade: *Neural Network-Based Face Detection*, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Bd. 20, Nr. 1, Jan. 1998, S. 23–38.

[Roy00] N. Roy, G. Baltus, D. Fox, F. Gemperle, J. Goetz, T. Hirsch, D. Magaritis, M. Montemerlo, J. Pineau, J. Schulte, S. Thrun: *Towards Personal Service Robots for the Elderly*, in *Proc. Int. Workshop on Interactive Robotics and Entertainment*, Pittsburgh, PA, April/Mai 2000.

[Sca00] B. Scassellati: *Investigating Models of Social Development Using a Humanoid Robot*, in B. Webb, T. Consi (Hrsg.): *Biorobotics*, MIT Press, Cambridge, MA, 2000.

[Sch98] C. Schlegel, J. Illmann, H. Jaberg, M. Schuster, R. Wörz: *Vision Based Person Tracking with a Mobile Robot,* in P. H. Lewis, M. S. Nixon (Hrsg.): *Proc. British Machine Vision Conference*, Southampton, UK, Sep. 1998, S. 418–427.

[Sch99] C. Schaeffer, T. May: *Care-O-bot^{TM}: A System for Assisting Elderly or Disabled Persons in Home Environments,* in C. Bühler, H. Knops (Hrsg.): *Assistive Technology on the Threshold of the New Millennium; AAATE 99, 5th European Conference for the Advancement of Assistive Technology*, Bd. 6 von *Assistive Technology Research Series*, IOS Press, Amsterdam, 1999, S. 340–345.

[Sch00] D. C. Schmidt, M. Stal, H. Rohnert, F. Buschmann: *Pattern-Oriented Software Architecture*, Bd. 2: Patterns for Concurrent and Networked Objects, John Wiley & Sons Ltd., 2000.

[Sch01a] R. D. Schraft, B. Graf, A. Traub, D. John: *A Mobile Robot Platform for Assistance and Entertainment, Industrial Robot: An International Journal*, Bd. 28, Nr. 1, Jan. 2001, S. 29–35.

[Sch01b] D. Schulz, W. Burgard, D. Fox, A. B. Cremers: *Tracking Multiple Moving Objects with a Mobile Robot,* in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Bd. 1, IEEE Press, Kauai, Hawaii, Dez. 2001, S. 371–377.

[Sch04] J. Schmidt: *Inkrementelle Adaption eines 3D-Körpermodells aus Bildsequenzdaten*, Diplomarbeit, Universität Bielefeld, Technische Fakultät, Angewandte Informatik, Bielefeld, Deutschland, Aug. 2004.

[Shi03] T. Shibata, K. Wada, K. Tanie: *Statistical Analysis and Comparison of Questionnaire Results of Subjective Evaluations of Seal Robot in Japan and UK,* in *Proc. IEEE Int. Conf. on Robotics and Automation*, Bd. 3, IEEE Press, Taipei, Taiwan, Sep. 2003, S. 3152–3157.

[Sid99] H. Sidenbladh, D. Kragić, H. I. Christensen: *A Person Following Behaviour for a Mobile Robot,* in *Proc. IEEE Int. Conf. on Robotics and Automation*, Bd. 1, IEEE Press, Detroit, Michigan, Mai 1999, S. 670–675.

[Sim94] R. G. Simmons: *Structured Control for Autonomous Robots, IEEE Trans. on Robotics and Automation*, Bd. 10, Nr. 1, Feb. 1994, S. 34–43.

[Sol00] J. Soler, V. Julián, C. Carrascosa, V. J. Botti: *Applying the ARTIS Agent Architecture to Mobile Robot Control,* in M. C. Monard, J. S. Sichman (Hrsg.): *Advances in Artificial Intelligence; International Joint Conference, 7th Ibero-American Conference on AI, 15th Brazilian Symposium on AI, IBERAMIA-SBIA 2000, Atibaia, SP, Brazil, November 2000. Proceedings*, Bd. 1952 von *Lecture Notes in Computer Science*, Springer-Verlag, Heidelberg, Germany, 2000, S. 359–368.

[Sou00] B. Souvignier, A. Kellner, B. Rueber, H. Schramm, F. Seide: *The Thoughtful Elephant: Strategies for Spoken Dialog Systems*, in *IEEE Trans. on Speech and Audio Processing*, Bd. 8, Jan. 2000, S. 51–62.

[Ste94] L. Steels: *The Artificial Life Roots of Artificial Intelligence*, *Artificial Life*, Bd. 1, Nr. 1, 1994, S. 76–110.

[Sti01] R. Stiefelhagen, J. Yang, A. Waibel: *Estimating Focus of Attention based on Gaze and Sound*, in *Proc. Workshop on Perceptive User Interfaces*, ACM Press, Orlando, FL, Nov. 2001.

[Sto01a] A. Stopp, S. Horstmann, S. Kristensen, F. Lohnert: *Towards Interactive Learning for Manufacturing Assistants*, in *Proc. IEEE Int. Workshop on Robot-Human Interactive Communication (ROMAN)*, IEEE Press, Bordeaux/Paris, France, Sep. 2001, S. 338–342.

[Sto01b] A. Stoytchev, R. C. Arkin: *Combining Deliberation, Reactivity and Motivation in the Context of a Behavior-Based Robot Architecture*, in *Proc. IEEE Int. Symp. on Computational Intelligence in Robotics and Automation (CIRA)*, Banff, Canada, Juli/Aug. 2001, S. 290–295.

[Suz02] K. Suzuki, R. Hikiji, S. Hashimoto: *Development of an Autonomous Humanoid Robot, iSHA, for Harmonized Human-Machine Environment*, *Journal of Robotics and Mechatronics, Special Issue on Human Robot Interaction*, Bd. 14, Nr. 5, Okt. 2002, S. 324–332.

[Suz03] K. Suzuki, S. Hashimoto: *A Multi-Layered Hierarchical Architecture for a Humanoid Robot*, in V. Palade, R. J. Howlett, L. C. Jain (Hrsg.): *Proc. Int. Conf. on Knowledge-Based Intelligent Information and Engineering Systems (KES)*, Bd. 2774 von *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Heidelberg, Germany, Sep. 2003, S. 592–599.

[Tan02] A. S. Tanenbaum, M. van Steen: *Distributed Systems: Principles and Paradigms*, Prentice Hall, 1.. Ausg., 2002.

[Tev00] G. Tevatia, S. Schaal: *Inverse Kinematics for Humanoid Robots*, in *Proc. IEEE Int. Conf. on Robotics and Automation*, Bd. 3, IEEE Press, San Francisco, CA, April 2000, S. 294–299.

[Thó02] K. R. Thórisson: *Machine Perception of Multimodal Natural Dialogue*, in P. McKevitt, S. Ó. Nualláin, C. Mulvihill (Hrsg.): *Language, Vision and Music; Selected papers from the 8th International Workshop on the Cognitive Science of Natural Language Processing, Galway, 1999*, Advances in Consciousness Research, John Benjamins Publishing Company, Amsterdam, The Netherlands, 2002, S. 97–115.

[Thr00] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, D. Schulz: *Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva*, *Int. Journal of Robotics Research, Special Issue on Field and Service Robotics*, Bd. 19, Nr. 11, Nov. 2000, S. 972–999.

[Toj00] T. Tojo, Y. Matsusaka, T. Ishii, T. Kobayashi: *A Conversational Robot Utilizing Facial and Body Expressions*, in *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics*, Nashville, TN, Okt. 2000, S. 858–863.

[Tom02] N. Tomatis, R. Philippsen, B. Jensen, K. O. Arras, G. Terrien, R. Piguet, R. Siegwart: *Building a Fully Autonomous Tour Guide Robot: Where Academic Research Meets Industry*, in *Proc. Int. Symp. on Robotics*, Stockholm, Sweden, Okt. 2002.

[Top04a] E. A. Topp, D. Kragic, P. Jensfelt, H. I. Christensen: *An Interactive Interface for Service Robots*, in *Proc. IEEE Int. Conf. on Robotics and Automation*, Bd. 4, IEEE Press, New Orleans, LA, April 2004, S. 3469–3474.

[Top04b] I. Toptsis, S. Li, B. Wrede, G. A. Fink: *A Multi-modal Dialog System for a Mobile Robot*, in *Proc. Int. Conf. on Spoken Language Processing*, Bd. 1, Jeju, Korea, Okt. 2004, S. 273–276.

[Tsc01] N. Tschichold, S. Vestli, G. Schweitzer: *The Service Robot MOPS: First Operating Experiences*, *Robotics and Autonomous Systems*, Bd. 34, Nr. 2–3, Feb. 2001, S. 165–173.

[Tur91] M. Turk, A. Pentland: *Eigenfaces for Recognition*, *Journal of Cognitive Neuroscience*, Bd. 3, Nr. 1, 1991, S. 71–86.

[Vau03] R. T. Vaughan, B. P. Gerkey, A. Howard: *On Device Abstractions For Portable, Reusable Robot Code*, in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Bd. 3, Las Vegas, NV, Okt. 2003, S. 2421–2427.

[vB04] A. J. N. van Breemen: *Animation Engine for Believable Interactive User-Interface Robots*, in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Bd. 3, Sendai, Japan, Sep./Okt. 2004, S. 2873–2879.

[Ver01] J. Vermaak, A. Blake, M. Gangnet, P. Perez: *Sequential Monte Carlo fusion of sound and vision for speaker tracking*, in *Proc. IEEE Int. Conf. on Computer Vision*, Bd. 1, IEEE Computer Society, Vancouver, Canada, Juli 2001, S. 741–746.

[Vio01] P. Viola, M. Jones: *Robust Real-time Object Detection*, in *Proc. IEEE Int. Workshop on Statistical and Computational Theories of Vision*, Vancouver, Canada, Juli 2001.

[Wal00] S. Waldherr, R. A. F. Romero, S. Thrun: *A Gesture Based Interface for Human-Robot Interaction*, *Autonomous Robots*, Bd. 9, Nr. 2, Sep. 2000, S. 151–173.

[Was97a] G. S. Wasson, G. J. Ferrer, W. N. Martin: *Perception, Action and Effective Representation in Multi-Layered Systems,* in *Proc. Joint Conf. on Computer Graphics, Human-Computer Interaction, Vision and Image Processing*, Kelowna, Canada, Mai 1997, S. 73–80.

[Was97b] G. S. Wasson, G. J. Ferrer, W. N. Martin: *Systems for Perception, Action and Effective Representation,* in *Proc. of the Florida Artificial Intelligence Research Symposium (FLAIRS), Special Track on Real-Time Planning and Reacting*, Daytona Beach, FL, Mai 1997, S. 352–356.

[Was99] G. S. Wasson, D. Kortenkamp, E. Huber: *Integrating Active Perception with an Autonomous Robot Architecture, Robotics and Autonomous Systems*, Bd. 29, Nr. 2–3, Nov. 1999, S. 175–186.

[Wil02] T. Wilhelm, H.-J. Böhme, H.-M. Groß: *Sensor Fusion for Visual and Sonar Based People Tracking on a Mobile Service Robot,* in *Proc. Int. Workshop on Dynamic Perception*, IOS Press, Infix, Bochum, Germany, Nov. 2002, S. 315–320.

[Wre04a] B. Wrede, A. Haasch, N. Hofemann, S. Hohenner, S. Hüwel, M. Kleinehagenbrock, S. Lang, S. Li, I. Toptsis, G. A. Fink, J. Fritsch, G. Sagerer: *Research Issues for Designing Robot Companions: BIRON as a Case Study,* in P. Drews (Hrsg.): *Proc. IEEE Conf. on Mechatronics & Robotics*, Bd. 4, Eysoldt-Verlag, Aachen, Aachen, Germany, Sep. 2004, S. 1491–1496.

[Wre04b] S. Wrede, J. Fritsch, C. Bauckhage, G. Sagerer: *An XML Based Framework for Cognitive Vision Architectures,* in *Proc. Int. Conf. on Pattern Recognition*, Bd. 1, Cambridge, UK, Aug. 2004, S. 757–760.

[Wre04c] S. Wrede, M. Hanheide, C. Bauckhage, G. Sagerer: *An Active Memory as a Model for Information Fusion,* in *Proc. Int. Conf. on Information Fusion*, Bd. 1, Stockholm, Sweden, Juni/Juli 2004, S. 198–205.

[Wys82] G. Wyszecki, W. S. Stiles: *Color Science: Concepts and Methods, Quantitative Data and Formulae,* John Wiley and Sons, New York, NY, 1982.

[Yan96] J. Yang, A. Waibel: *A Real-time Face Tracker*, in *Proc. IEEE Workshop on Applications of Computer Vision*, Sarasota, FL, Dez. 1996, S. 142–147.

[Yan02] M.-H. Yang, D. J. Kriegman, N. Ahuja: *Detecting Faces in Images: A Survey, IEEE Trans. on Pattern Analysis and Machine Intelligence*, Bd. 24, Nr. 1, Jan. 2002, S. 34–58.

[Yen95] J. Yen, N. Pfluger: *A Fuzzy Logic Based Extension to Payton and Rosenblatt's Command Fusion Method for Mobile Robot Navigation, IEEE Trans. on Systems, Man, and Cybernetics*, Bd. 25, Nr. 6, Juni 1995, S. 971–978.

[Yui92] A. L. Yuille, P. W. Hallinan, D. S. Cohen: *Feature Extraction from Faces Using Deformable Templates*, *Int. Journal of Computer Vision*, Bd. 8, Nr. 2, Aug. 1992, S. 99–111.

[Zha02] Z. Zhang, L. Zhu, S. Z. Li, H. Zhang: *Real-Time Multi-View Face Detection*, in *Proc. IEEE Int. Conf. on Automatic Face and Gesture Recognition (FG)*, IEEE Computer Society, Washington DC, USA, Mai 2002, S. 149–154.

Index