

# MobMan – Ein mobiler Manipulator für Alltagsumgebungen

Georg von Wichert, Thomas Wösch, Steffen Gutmann und Gisbert Lawitzky

Siemens AG, Zentralabteilung Technik, ZT IK 6, 81730 München

**Zusammenfassung** Der Einsatz von mobilen Servicerobotern beschränkt sich bisher auf Transport- und Reinigungsaufgaben, bei welcher Mobilität und Navigation in einer 2-dimensionalen Welt die zentrale Rolle spielen. Weitere Einsatzgebiete (z.B. im Haushalt) erfordern die Fähigkeit, Gegenstände manipulieren zu können. Inhalt dieses Beitrages ist die Beschreibung des bei der Siemens AG hierfür entwickelten Forschungsprototypen für Mobilität und Manipulation (MobMan) in Alltagsumgebungen. Der Schwerpunkt liegt auf einem Ansatz zur Steuerung von Manipulationskills in komplexen Alltagsumgebungen.

## 1 Einleitung

Der Einsatz von mobilen Servicerobotern beschränkt sich bisher meist auf Transport- und Reinigungsaufgaben, z.B. in Bahnhöfen oder Supermärkten [1, 2], bei welchen Mobilität und Navigation in einer 2-dimensionalen Welt die zentrale Rolle spielen. Sollen weitere Einsatzgebiete z.B. im Haushalt erschlossen werden, so müssen in einer 3-dimensionalen Umgebung Gegenstände manipuliert werden können. Derartige Robotersysteme agieren in, auf die Bedürfnisse des Menschen zugeschnittenen, Alltagsumgebungen, deren hohe Komplexität eine vollständige Modellierung unmöglich macht. Die jeweilige Ausprägung der Einsatzumgebung wird zum Zeitpunkt des Systementwurfs in der Regel nicht bekannt sein. Die Inbetriebnahme darf dennoch keinerlei Expertenwissen erfordern. Es muß also ein Weg gefunden werden, die Serviceroboter apriori mit einem gewissen Satz an Grundfähigkeiten perzeptiver, aktorischer und kognitiver Art auszustatten, ohne daß eine genaue Kenntnis der späteren Einsatzumgebung vorliegt.

Aufgrund der notwendigerweise unvollständigen Modellierung, ebenso wie aus Aufwands- und Komplexitätsgründen, können Systemaktionen nicht in allen Details vorausgeplant und ausgeführt werden. Sie müssen unter Nutzung aller verfügbaren Sensordaten in der Form reaktiver, sensomotorischer Skills realisiert werden. Die Einführung solcher Skills ermöglicht es, durch die in ihnen gebündelte Kompetenz im Umgang mit der Umwelt, auf übergeordneten Ebenen lediglich Grobplanungen der Systemaktionen durchzuführen. Dabei kann von den konkreten Umgebungseigenschaften abstrahiert werden. Diese Teile des Systems können ohne eine konkrete Kenntnis der spezifischen Einsatzumgebung erstellt werden, wofür nur qualitatives Wissen bezüglich der Einsatzdomäne erforderlich ist. Der Realisierung leistungsfähiger Skills kommt daher eine zentrale Bedeutung zu. In diesem Beitrag wird, nach der Vorstellung des Experimentalsystems MobMan, vor allem auf den verwendeten Ansatz zur Beschreibung

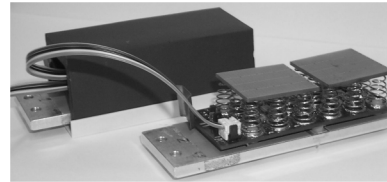
und Ausführung sensomotorischer Skills eingegangen. Die Leistungsfähigkeit des Ansatzes wird am Beispiel einer komplexen Manipulationsaufgabe demonstriert.

## 2 Manipulation in Alltagsumgebungen

Für die diesem Beitrag zugrundeliegenden Untersuchungen steht das in Abbildung 1 dargestellte Experimentalsystem MobMan zur Verfügung. Im folgenden soll neben einer Beschreibung der verwendeten Hardware auf die Steuerungsarchitektur eingegangen werden.



**Abbildung 1.** Experimentalsystem MobMan



**Abbildung 2.** Parallelbackengreifer

### 2.1 Hardware des Experimentalsystems

Das Experimentalsystem, siehe Abbildung 1, besteht aus einer mobilen Plattform mit Differentialantrieb, einem Arm mit 6 Freiheitsgraden, einem mit Taktile Sensoren ausgerüsteten Parallelbackengreifer als Endeffektor und zwei Industrie-PC's auf denen SICOMP RMOS bzw. Linux läuft. Die Navigation der Plattform erfolgt mit einer modifizierten Version des Siemens-Navigationssystems SINAS [1], das die Sensordaten eines auf der Plattform befestigten Laserscanners auswertet.

Der Greifer ist für Manipulationsaufgaben von besonderer Bedeutung. Wie bereits erwähnt, handelt es sich um einen Parallelbackengreifer, siehe Abbildung 2. Jeder Finger besteht aus zwei voneinander unabhängigen taktilen Sensoren. Jeder Sensor, bestehend aus einer Leiterplatte mit entsprechender Sensorelektronik, ist über Federn an einer gemeinsamen Bodenplatte, die die Hardware für die Sensordatenauswertung enthält, befestigt. Solange keine Zug- oder Drucklast auf die jeweilige Sensorplatte wirkt, halten die Federn den Abstand zwischen Sensorplatte und gemeinsamer Bodenplatte konstant. Unter Belastung verschiebt bzw. verdreht sich die Sensorplatte relativ zur Bodenplatte.

zur Bodenplatte, wodurch man auf die einwirkenden Kräfte bzw. Momente schließen kann. Es ist möglich, Verschiebungen in jede Achsenrichtung und auch Verdrehungen um jede Achsenrichtung zu detektieren. Weiterhin liefert der Greifer auch Informationen über die Größe und Form der Kontaktfläche. Die Leiterplatten mit Elektronik und Federn werden von einem elastischen Kunststoffmantel umgeben, der einerseits Schutz bietet, aber auch eine Haftreibung zwischen Greifer und manipuliertem Objekt bietet. Der Greifer ist mit einem 16-Bit Microprozessor ausgerüstet und enthält sein eigenes Programm zur Sensordatenauswertung. Steuerrechner und Greifer sind über eine serielle Schnittstelle miteinander verbunden.

Um die für die Manipulation wichtigen Tiefenveränderungen erfassen zu können, steht eine schräg zur Greiferebene montierte monokulare Farbkamera zur Verfügung. Zusammen mit einem in der Greiferebene aufgefächerten Laserstrahl, der auf das zu manipulierende Objekt projiziert wird, können Tiefenveränderungen, durch seitliche Verschiebung der Laserreflexion, wahrgenommen werden.

## 2.2 Steuerungsarchitektur

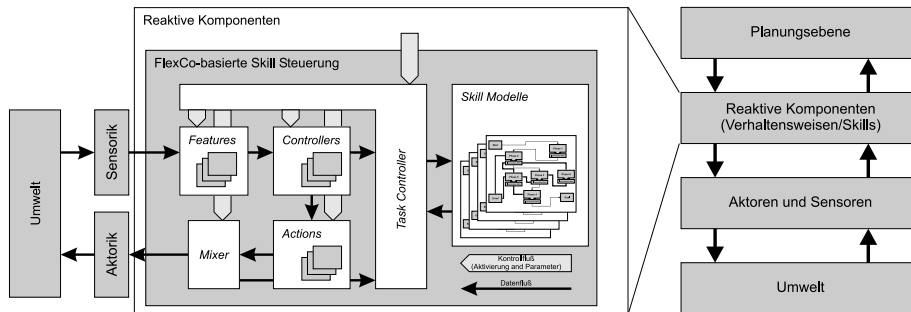
Der Einsatz in Alltagsumgebungen (bspw. in Privathaushalten) stellt an das Steuerungssystem besonders hohe Anforderungen. Diese resultieren in der Hauptsache aus der hohen Komplexität der Umgebungen. Deren jeweilige Ausprägung wird zum Zeitpunkt des Systementwurfs in der Regel nicht bekannt sein. Ein Serviceroboter sollte jedoch auch von ungeschultem Personal in Betrieb genommen werden können, so dass eine aufwendige (3D-)Modellierung der Umwelt während der Inbetriebnahme nicht erfolgen kann – sofern sie das System nicht selbsttätig vornimmt<sup>1</sup>.

Betrachtet man sich die einschlägige Literatur (siehe bspw. [3, 4]), so fällt ein gemeinsames Strukturmerkmal der Systemarchitekturen vieler autonomer Robotersysteme auf. Es handelt sich dabei um die bei allen Systemen auftretende hierarchische Kombination reaktiver Mechanismen auf einer unteren Ebene mit darüber angesiedelten (klassisch) planenden Komponenten (siehe Abbildung 3 rechts). Die Planung erfolgt dabei auf einem von der Realität abstrahierenden Niveau, weil – entsprechend dem oben gesagten – ein die Wirklichkeit genau wiedergebendes Umgebungsmodell als Basis der Planung fehlt. Diese Grobstruktur vieler Robotersteuerungen hat sich insbesondere im Bereich Mobilität/Navigation sehr bewährt, weshalb sie hier auch auf die Manipulation übertragen werden soll. Voraussetzung dafür ist die Existenz einer kompetenten reaktiven Skill-Ebene, die eine Abstraktion zum Zweck der Planung erlaubt. In den beiden nächsten Abschnitten wird das Konzept einer solchen Skillebene beschrieben und gezeigt, daß sich so komplexe Manipulationsaufgaben in realen Umgebungen lösen lassen, ohne daß vollständige Umgebungsmodelle benötigt werden.

## 3 FlexCo – Beschreibung und Ausführung sensomotorischer Skills

Im folgenden wird zunächst die in der obigen Architektur vorgesehene Komponente zur Beschreibung und Ausführung reaktiver, sensomotorischer Skills diskutiert.

<sup>1</sup> Zum gegenwärtigen Zeitpunkt erscheint ein vollständig selbstlernendes System jedoch unrealistisch.

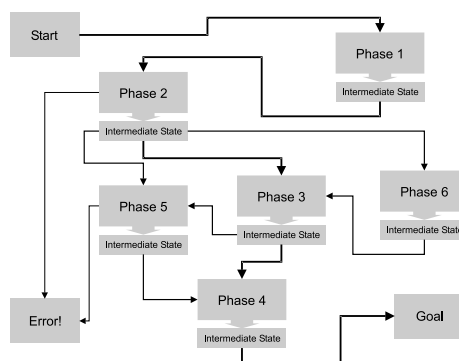


**Abbildung 3.** Grobstruktur der Steuerung des Gesamtsystems mit Ausschnittvergrößerung der reaktiven Skillebene gemäß FlexCo-Ansatz (siehe dazu Abschnitt 3)

### 3.1 Entwurfsprinzipien

Zunächst steht aufgrund der weiter oben diskutierten Eigenschaften komplexer Alltagsumgebungen fest, daß eine vollständige Modellierung der Umwelt nicht möglich ist. Deshalb wird festgelegt, daß lediglich die für die im jeweiligen Moment zu bewältigende Aufgabe notwendigen Aspekte der Umwelt modelliert werden sollen. Letztendlich wird so nicht die Umwelt, in der die Aufgabe gelöst werden soll, sondern die Lösung der Aufgabe (d.h. der Skill) modelliert.

Für diese Aufgabenmodellierung, welche die in den klassischen Systemen betriebene Umweltmodellierung ersetzt, ist es nicht erforderlich, die aufgabenrelevanten Teile der Umwelt explizit als solche zu modellieren, vielmehr genügt es, ihr sensorisches Erscheinungsbild in das Aufgabenmodell einzubeziehen. Diese Erkenntnis führt zur Methodik des *feature based servoing* (siehe dazu beispielsweise [5]). Dabei wird nicht der Zustand des Systems selbst, sondern vielmehr der sich aus der Summe der Sensordaten ergebende „sensorische Zustand“ betrachtet.



**Abbildung 4.** Aufgabenbeschreibung im FlexCo-Framework. Fette Pfeile kennzeichnen den Nominalablauf.

Zur Modellierung wird die Aufgabe in eine (nicht notwendigerweise lineare) Folge von Phasen (siehe dazu auch Abbildung 4) zerlegt, an deren Ende jeweils ein Zwischenziel steht. Dieses Zwischenziel wird durch einen sensorischen Sollzustand charakterisiert, das heißt, es werden Sollwerte für bestimmte ausgewählte und vor allem sensorisch bestimmbare Merkmale festgelegt. Dem System wird eine Reihe elementarer Verhaltensweisen zur Verfügung gestellt, mit deren Hilfe es die spezifizierten Sollwerte erreichen kann. Die Einteilung der Aufgabe in Phasen, die Auswahl der in der jeweiligen Phase interessierenden Sensormerkmale und die Auswahl geeigneter Verhaltensweisen zum Erreichen des Zwischenzieles ergeben sich aus dem qualitativen Kontext und sind weitestgehend unabhängig von der Roboterhardware und den spezifischen Eigenschaften der konkreten Einsatzumgebung. Es sollte daher möglich sein, derartige Zusammenhänge für ganze Klassen von Robotern und Einsatzumgebungen zu spezifizieren, dies wurde jedoch bisher nicht systematisch untersucht. Die eigentlichen Regelschleifen zur Implementierung der genannten Verhaltensweisen sind hingegen stark durch die jeweilige Umwelt und die Hardware des Roboters bestimmt. Ihre Auslegung erfordert quantitatives Detailwissen über den Roboter und seine Einsatzumgebung. Im folgenden Abschnitt wird die Realisierung des Regelungskonzeptes im Detail beschrieben. Im Anschluß daran präsentiert Abschnitt 4 eine beispielhafte Manipulationsaufgabe und deren Realisierung mit Hilfe des FlexCo-Frameworks.

### 3.2 Realisierung

Entsprechend den im vorhergehenden Abschnitt beschriebenen Prinzipien ergibt sich eine Aufgabenbeschreibung, wie sie beispielhaft in der Abbildung 4 dargestellt ist. Sie besteht aus einer Reihe möglicher Aufgabenphasen bzw. Teilaufgaben, die das System durchlaufen muß. Dargestellt ist ein Ablauf, bei dem das System vom Startzustand über die Phasen 1 bis 4 den Zielzustand erreichen kann. Dieser Nominalablauf wird durch die fetten Pfeile gekennzeichnet. Das FlexCo-Konzept sieht neben diesem nominalen Ablauf jedoch auch alternative Pfade vor, die unter zu spezifizierenden Randbedingungen verfolgt werden können. In der Zeichnung werden dabei die Phasen 5 oder 6 durchlaufen. Ebenso sind Fehlerzustände vorzusehen, in denen das System aufgrund schwerer Abweichungen vom Nominalablauf eine geordnete und sichere Terminierung der Aufgabe vornehmen kann.

Zur Beschreibung der sensomotorischen Skills wird vom System eine Reihe von in anderen Skills wiederverwertbaren Basiskomponenten bereitgestellt, welche die sensorischen und aktorischen Möglichkeiten des Roboters repräsentieren. In der Abbildung 3 (links) werden diese als *Features* und *Actions* bezeichnet. *Features* repräsentieren reale und logische Sensoren inklusive der zugehörigen Auswertungsverfahren, *Actions* repräsentieren die Möglichkeiten der Aktorik (z.B. vordefinierte Bewegungsmodi). Applikationsspezifisch festgelegte reaktive Mechanismen (*Controller*) werten die *Features* aus, vergleichen diese mit dem vom Entwickler des Skills spezifizierten sensorischen Zielzustand und generieren bei Bedarf nach einem vom Entwickler des *Controllers* festzulegenden Algorithmus entsprechende *Actions*, welche das System dem Zielzustand näherbringen (sollen). Eine Konvergenz dieser reaktiven Vorgehensweise kann allgemein natürlich nicht garantiert werden. Es liegt daher in der Verantwortung des Skillentwicklers dafür zu sorgen, daß die Konvergenz im Kontext der Anwendung sicher-

gestellt ist. Ein phasenspezifischer Entscheidungsprozeß filtert die von den *Controllern* generierten Systemaktionen und bestimmt anschließend die vom System im jeweiligen Augenblick tatsächlich auszuführende Aktion, diese Komponente wird in der Abbildung als Mixer bezeichnet. Das System verharrt in jeder der durchlaufenen Phasen, bis ein durch die *Controller* festgelegter Satz von Nebenbedingungen erfüllt und somit ein durch diese Nebenbedingungen festgelegter Zwischenzustand erreicht ist. In dieser Hinsicht weist der Ansatz Ähnlichkeiten mit [6] auf. Diese Bedingungen betreffen die Einhaltung bzw. das Erreichen der oben erwähnten Sollwerte für die Daten unterschiedlicher realer oder logischer Sensoren.

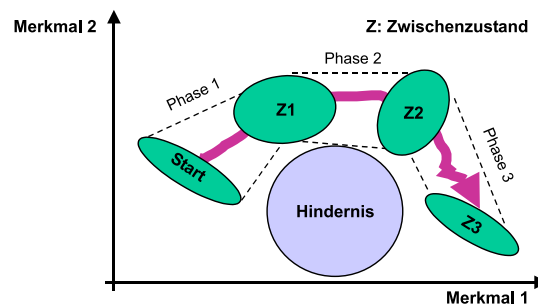


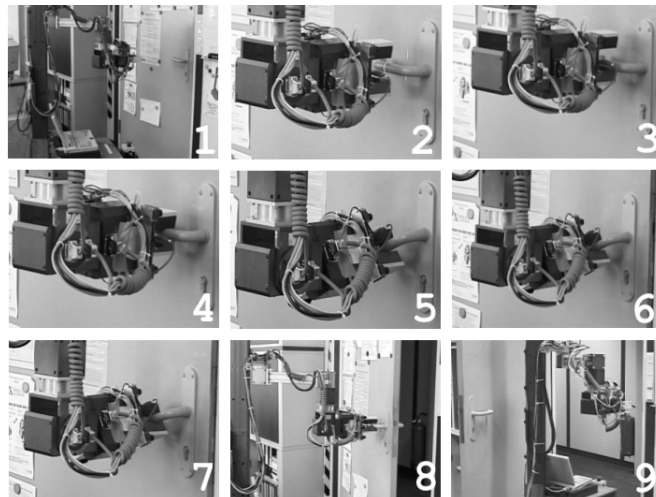
Abbildung 5. Trajektorie des Systems im Sensorraum.

Im Ergebnis ist damit in jeder Phase der Aufgabenausführung ein auf diese konkrete Phase abgestimmter Satz von Verhaltensweisen aktiv, der die Reaktionen des Systems auf die jeweils betrachteten Sensordaten bzw. daraus die abgeleiteten Merkmale bestimmt. Das Gesamtverhalten soll dazu führen, daß der durch die oben genannten Nebenbedingungen spezifizizierte Zwischenzustand sicher erreicht wird, damit die nächste Phase beginnen kann. Der erreichte Zwischenzustand muß dabei im Einzugsbereich der nächsten Phase liegen, d.h. in dem Bereich des Zustandsraums, in welchem die in der nächsten Phase aktiven Verhaltensweisen das System in den entsprechenden Zwischenzustand führen. Diese Zusammenhänge werden vereinfacht in der Abbildung 5 dargestellt. Das FlexCo-Framework besteht aus einer in C++ implementierten Klassenbibliothek, die alle benötigten Bestandteile des Aufgabenmodells sowie der Ablaufsteuerung zur Verfügung stellt. Auf dieser Basis sind die anwendungsspezifischen Komponenten (*Features*, *Actions*, *Controller*) zu programmieren. Die Aufgabenbeschreibung, d.h. der fertige Skill, gemäß der Abbildung 4 wird aus diesen Komponenten zusammengesetzt, in einer XML-konformen Definitionsdatei abgelegt und vom System eingelesen.

#### 4 Beispiel einer FlexCo-basierten Skillbeschreibung

Anhand des „Türöffnens“ soll das Prinzip einer FlexCo-basierten Skillbeschreibung veranschaulicht werden. Der Einfachheit halber beschränkt sich die Implementierung

des Skills auf jene Klassen europäischer Türen, die im geöffneten Zustand selbständig offen bleiben. Soll der mobile Manipulator nach dem Öffnen der Tür durch diese hindurchfahren, muß die Tür eine Mindestbreite und -höhe haben.



**Abbildung 6.** Nominalablauf der in FlexCo implementierten Aufgabe eines „Türöffnens“

Die in Abbildung 6 dargestellte Bildfolge präsentiert den Nominalablauf beim Öffnen einer Tür in bildlicher Form, wobei jede Sequenz einem Zustand entspricht, in dem sich die FlexCo-basierte Ablaufregelung gerade befindet. Um den Türgriff greifen zu können fährt der mobile Manipulator zu einem bestimmten Abstand vor der Tür, von wo aus er, ohne die Plattform bewegen zu müssen, die Tür öffnen kann. Dies muß gewährleistet sein, da auf eine Arm-Plattform-Koordination [7] vorerst verzichtet worden ist. Bild 1 aus Abbildung 6 zeigt den Manipulator in der Startposition. Als nächstes wird mit Hilfe der monokularen Kamera und dem aufgefächerten Laser der Türgriff „vermessen“. Die Anfangs- und Endpositionen des Türgriffes werden dabei ermittelt. Bild 2 zeigt das Justieren des vorderen Taktilsensorpaares vom Greifer in die Mitte des Türgriffes. Bevor der Griff gegriffen wird, muß der Greifer orthogonal zum Türgriff ausgerichtet werden, siehe Bild 3. Bild 4 zeigt den geschlossenen Greifer. Als nächstes wird versucht, den Griff zu rotieren. Dazu wird im „oberen“ Taktilsensor eine Kraft gefordert, die zu einer Rotation des Türgriffes führt. Der Vorgang ist in Bild 5 dargestellt. Läßt sich der Türgriff nicht mehr weiter rotieren, übersteigt die Kraft im oberen Taktilsensor des Greifers eine vorgegebene Schwelle, worauf der Rotationsvorgang beendet wird, und die Tür kann um einen „Spalt“ geöffnet werden. In Bild 6 sieht man die um ca. 2cm geöffnete Tür. Sollte ein Öffnen nicht möglich sein (weil die Tür beispielsweise verschlossen ist), wird dies über die Taktilsensoren detektiert. Eine entsprechende Fehlerbehandlung läßt den Arm dann beispielsweise wieder in die Startposition fahren. Bild 6 zeigt die erfolgreiche Türöffnung. Als nächster Schritt wird der Türgriff wie-

der in seine waagrechte Lage zurückrotiert, siehe Bild 7. Einem Öffnen der Tür steht nun nichts mehr im Weg, Bild 8 zeigt eine Momentaufnahme des Türaufschwingens. Nachdem die Tür genügend weit geöffnet werden konnte, kann der Roboter durch die geöffnete Tür fahren, siehe Bild 9 aus Abbildung 6.

## 5 Zusammenfassung und Ausblick

Der vorliegende Beitrag beschreibt ein Konzept zur Implementierung leistungsfähiger, sensomotorischer Skills. Diese bilden als elementare Fähigkeiten die Basis eines Steuerungssystems, bei dem auf die aufwendige Modellierung komplexer Alltagsumgebungen verzichtet wird. Die Skills setzen sich aus generischen Komponenten (*Features, Actions, Controller*) zusammen, die – einmal implementiert – unverändert zu anderen Skills rekombiniert werden können. Die Skillbeschreibung wird in einer XML-konformen Sprache formuliert und vom System zur Laufzeit eingelesen. Am Experiment wurde gezeigt, daß auf diese Weise komplexe Manipulationen durchgeführt werden können.

Es ist offensichtlich, daß mit den hier beschriebenen Resultaten noch kein vollständiges System zur Verfügung steht. Weitere Arbeiten werden sich daher, neben Erweiterungen auf der Hardwareseite, mit dem Aufbau einer Planungsebene beschäftigen. Die Skillebene selbst soll um Mechanismen zum (überwachten) Erlernen umgebungsspezifischer Informationen erweitert werden. In diesem Zusammenhang ist auch die insbesondere im Servicebereich große Bedeutung der Interaktion von Mensch und Maschine zu betonen, die im Rahmen des BMBF-Leitprojekts MORPHA untersucht werden soll.

Die diesem Bericht zugrundeliegenden Arbeiten wurden teilweise mit Mitteln des Bundesministeriums für Bildung und Forschung unter den Förderkennzeichen 01IN601A2 und 01IL902DO gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

## Literatur

- [1] G. Lawitzky, “Das Navigationssystem SINAS”, in *Proceedings Robotik 2000, VDI-Berichte 1582*, (Düsseldorf), S. 77–82, VDI-Verlag, 2000.
- [2] E. Prassler, J. Scholz, und M. Strobel, “MAid: Mobility Assistance for Elderly and Disabled People”, in *Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society (IECON’98)*, (Aachen, Germany), 1998.
- [3] D. Kortenkamp, R. P. Bonasso, und R. Murphy, Hrsg., *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*. The MIT Press, 1998.
- [4] R. C. Arkin, *Behavior-based Robotics*. Cambridge, USA: The MIT Press, 1998.
- [5] G. Hager und K. Toyama, “The XVision System: A General-Purpose Substrate for Portable Real-Time Vision Applications”, *Computer Vision and Image Understanding*, Bnd. 69, Nr. 1, S. 23–37, 1998.
- [6] R. R. Burridge, A. A. Rizzi, und D. E. Koditschek, “Sequential Composition of Dynamically Dexterous Robot Behaviors”, *International Journal of Robotics Research*, Bnd. 18, Nr. 6, S. 534–555, 1999.
- [7] Z. Kemény, “Design and Kinematic Motion Planning for Mobile Manipulators”, in *Proceedings of the 2000 International Conference on Intelligent Engineering Systems (INES’2000)*, (to appear).