14. Fachgespräch Autonome Mobile Systeme (AMS '98), Karlsruhe, Germany, 30. Nov. - 1. Dez. 1998, © Springer-Verlag, Reihe "Informatik Aktuell", pp 180-187.

Ein verhaltensorientierter Ansatz zum flächendeckenden Fahren in a priori unbekannter Umgebung

Michael Kasper, Dirk Müller, Ewald von Puttkamer

Universität Kaiserslautern • Fachbereich Informatik
AG Robotik und Prozeßrechentechnik • Postfach 3049 • D-67653 Kaiserslautern
kasper@informatik.uni-kl.de • http://ag-vp-www.informatik.uni-kl.de

Kurzfassung

In diesem Aufsatz wird ein Verfahren zum flächendeckenden Fahren in zunächst unbekannter Umgebung beschrieben, wie es z.B. für Reinigungsanwendungen im Heimbereich benötigt wird. Parallel zur Durchführung der Reinigungsaufgabe wird dabei die Umgebung exploriert und kartiert.

Der verhaltensorientierte Ansatz ermöglicht eine robuste, zielgerichtete und dennoch ressourcenschonende Implementierung und gestattet es, einzelne Verhaltensweisen leicht durch verbesserte oder auch speziell erlernte Versionen auszutauschen. Das vorgestellte Verfahren wurde simulativ getestet und wird in Kürze auf einem realen Roboter erprobt.

1. Einleitung

Servicerobotern wird als "Leitprodukten der Zukunft" in den kommenden Jahren ein großes Wachstumspotential vorausgesagt [Schraft und Schmierer 98]. Einen relativ hohen Anteil an diesen auf den Dienstleistungssektor spezialisierten Robotern werden dabei verschiedenste Reinigungsmaschinen einnehmen. Neben spezialisierten Robotern zur Fassaden- oder Fahrzeugreinigung kommt hier der Bodenreinigung eine Schlüsselrolle zu. Während es bereits seit einigen Jahren (teil-)autonome Bodenreinigungsroboter für den kommerziellen Einsatz auf Flughäfen, Bahnhöfen oder in Supermärkten gibt (z.B. [Hako 94], [Kent 98], [Lawitzky et al. 98]), existieren für den Heimbereich bisher nur Prototypen, wie die Roboter der Firmen Electrolux oder Minolta.

Der vorliegende Aufsatz konzentriert sich auf die Steuerungskomponente eines solchen autonomen Bodenreinigungsroboters für den Privatbereich. Nach der Diskussion der Anforderungen an diese Systeme wird ein verhaltensorientierter Ansatz vorgestellt, der diesen entgegenkommt. Die ersten experimentellen Ergebnisse werden präsentiert und eine Zusammenfassung, verbunden mit einem Ausblick auf das in der Entwicklung befindliche reale System, beschließt den Beitrag.

2. Anforderungen und Voraussetzungen

Autonomous Home Cleaning Robots müssen leicht und klein genug sein, um zwischen Stuhl- und Tischbeinen reinigen zu können. Sie benötigen eine robuste und feh-

lertolerante Steuerung, die auch mit dynamischen Objekten umgehen kann und den Roboter zielgerichtet und effizient steuert. - Der Prototyp der Firma Electrolux [Electrolux 97] arbeitet z.B. rein indeterministisch und somit nicht zielgerichtet.

Da für die sich häufig ändernden Einsatzumgebungen (Veränderung innerhalb einer Umgebung, Einsatz in verschiedenen Umgebungen) der Aufwand einer a priori Kartenerstellung nicht gerechtfertigt erscheint, sollte der Roboter parallel zur Reinigungsaufgabe seine Umgebung explorieren und kartieren. Es sind nur wenige Ansätze bekannt, die dies berücksichtigen (z.B. [Burhanpukar 94] oder [Furuhashi et al. 92]), wobei aber entweder die Algorithmen nicht veröffentlicht sind, oder der Nachweis der praktischen Einsetzbarkeit fehlt.

Schließlich sollten autonome Reinigungsroboter sowohl einfach in der Handhabung als auch preiswert sein. Letzteres stellt aus heutiger Sicht neben der begrenzten Batterietechnologie die größte Einschränkung dar. Die technologischen Anforderungen lassen sich mit entsprechender Ausstattung prinzipiell lösen. Durch die finanziellen Restriktionen, die noch stärker als im industriellen Bereich greifen, und den daraus resultierenden Beschränkungen in der Sensorik und der Rechenleistung scheiden jedoch viele klassische Lösungsansätze von vornherein aus.

Auf der anderen Seite kann die Realisierung durch einige Annahmen vereinfacht werden, die sich auf ein einzelnes reales System im Heimbereich übertragen lassen: Um die Navigation zu erleichtern, kann ein runder Roboter mit der Fähigkeit zu Punktdrehungen vorausgesetzt werden. Ferner werden in der Einsatzumgebung keine "aggressiven" dynamischen Hindernisse auftreten, bei denen ein einfaches Ausweichen oder Stehenbleiben des Roboters zur Kollisionsvermeidung nicht ausreichen würde.

Eine weitergehende Vereinfachung ergibt sich aus der Annahme, daß ein Localizer-Modul existiert, welches aus den Sensordaten (ggf. eines speziellen Sensors) die aktuelle Position des Roboters relativ zur Startposition bestimmt. Arbeiten wie [Weiß et al. 94], [Lu und Milios 97] oder [Thrun et al. 98] zeigen Möglichkeiten auf, wie z.B. durch Korrelation von Sensordaten die Roboterposition ausreichend genau bestimmt werden kann.

Eine Alternative stellen spezielle Sensorsysteme zur Positionsbestimmung dar, die sich auf künstliche Landmarken stützen. Sofern diese Systeme den finanziellen Voraussetzungen genügen und mit nur geringen Umweltveränderungen auskommen, erscheint ihr Einsatz durchaus gerechtfertigt. Ein entsprechendes System wird derzeit in der Arbeitsgruppe der Autoren entwickelt, soll hier jedoch nicht näher diskutiert werden.

3. Systemstruktur

Im folgenden wird eine verallgemeinerte verhaltensorientierte Architektur vorgestellt, welche in ihren Wurzeln auf Arbeiten von Brooks zurückgeht [Brooks 86]. Durch ihren modularen Aufbau, die kurzen Reaktionszeiten und die hohe Fehlertoleranz sowie durch den relativ geringen Ressourcenbedarf wird die verhaltensorientierte Kontrollstruktur den genannten Anforderungen in idealer Weise gerecht. Sie bietet zudem den Vorteil, daß die einzelnen Module weitgehend unabhängig voneinander implementiert und getestet sowie bei Bedarf zu einem späteren Zeitpunkt durch verbesserte Versionen ersetzt werden können.

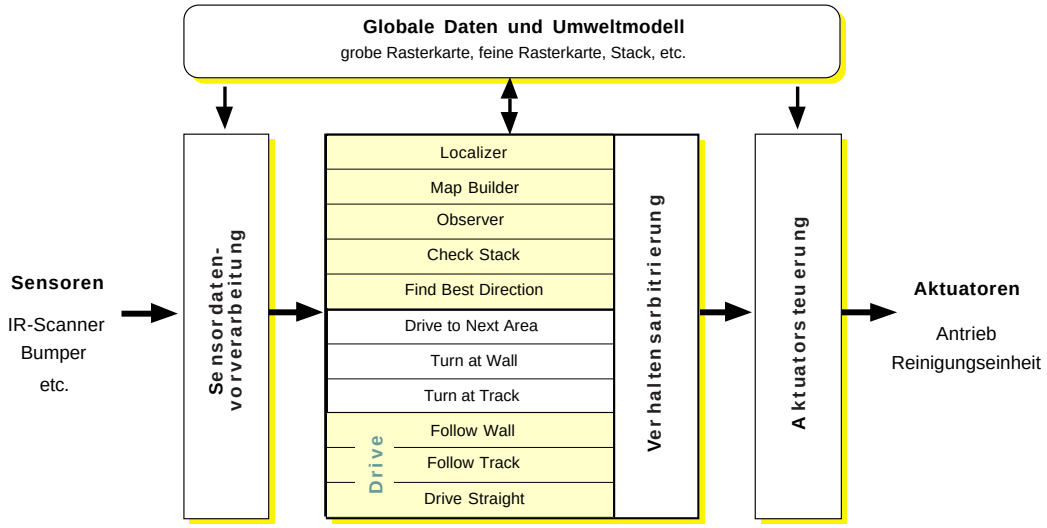

Abbildung 1: verallgemeinerte verhaltensorientierte Kontrollstruktur

Im Gegensatz zu der von Brooks vorgeschlagenen *subsumption architecture* kommt bei dieser Arbeit jedoch eine allgemeinere Arbitrierungseinheit (vgl. [Rosenblatt 97], [Arkin 98]) zur Verhaltensauswahl zum Einsatz. Darüber hinaus werden globale Daten und Zustände zugelassen, die Voraussetzung für die Umweltmodellierung sind.

Neben konkurrierenden, auf verschiedene Aufgaben spezialisierten *aktuatorischen Verhaltensschichten* (ähnlich den Bewegungsmakros in [Hofner 97]) wurden verdeckte, im Hintergrund laufende *background behaviors* implementiert, welche lediglich globale Daten verändern bzw. die Arbitrierung beeinflussen. Das Gesamtverhalten des Roboters wird durch das emergente Zusammenspiel der einzelnen Verhaltensweisen bestimmt. Abbildung 1 zeigt die generelle Systemsstruktur.

Die Verhaltensmodule greifen im wesentlichen auf drei Datenstrukturen zurück: eine hochauflösende Rasterkarte zur Umweltmodellierung (Repräsentation von freien, bereits abgearbeiteten bzw. von Hindernissen besetzten Flächenstücken), eine grobe Rasterkarte zur Navigation (s.u.) sowie auf einen Stack, auf dem noch abzufahrende Anfangsstrecken abgelegt werden. Die flächendeckende Fahrweise ergibt sich durch sukzessives Abarbeiten der auf dem Stack befindlichen Streckensegmente, was letztlich einer backtracking-Strategie entspricht.

Der zugrundeliegende Algorithmus läßt sich folgendermaßen skizzieren:

1. Fahre geradeaus bzw. bündig entlang der eigenen Spur bis zum Auftreffen auf ein Hindernis. (Unter einer *Spur* wird eine vom Roboter bereits abgefahrene Bahn verstanden.) Beobachte dabei die benachbarten Bahnen und speichere freie, noch nicht befahrene Streckenabschnitte auf einem Stack ab.
2. Solange der Stack noch Einträge enthält, fahre zu der entsprechenden Position und beginne von vorne.
3. Ende

Dieser Algorithmus wird durch die einzelnen, verdeckten und aktuatorischen Verhaltensweisen realisiert.

3.1 Verdeckte Verhaltensweisen

Neben dem eingangs erwähnten *Localizer-Modul* wurden folgende verdeckte Verhaltensweisen implementiert:

Map Builder: Der *Map Builder* ist für das Erstellen und Verwalten der beiden Rasterkarten verantwortlich. Während die hochauflösende Karte zur Umweltmodellierung von den meisten Verhaltensweisen genutzt wird, wird die gröbere Karte aus Performancegründen zur Navigation mittels eines Potentialfeld-Ansatzes von dem Modul *Drive to Next Area* benötigt. Der in der Simulation eingesetzte *Map Builder* geht stets von korrekten Sensordaten aus. In dem realen System wird ein probabilistischer Ansatz, z.B. nach [Elfes 89], [Jörg 94] oder [Thrun et al. 98], Verwendung finden.

Observer: Die Aufgabe des *Observers* ist die Beobachtung der zur aktuellen Fahrbahn benachbarten Parallelbahnen im Verlauf der Geradeausfahrt. Wann immer ein befahrbarer, noch nicht bearbeiteter Bereich rechts oder links erkannt wird, werden die Start- und Endposition des entsprechenden Streckenabschnitts der Nachbarbahn auf dem Stack vermerkt.

Check Stack: Während der *Observer* dem Stack Einträge hinzufügt, dient *Check Stack* dazu, überflüssige Einträge wieder zu löschen. Ein Eintrag wird überflüssig, wenn die zugehörige Bahn zwischenzeitlich vom Roboter gekreuzt wurde. Die Flächendeckung bleibt auch nach dem Löschen gewährleistet, da der *Observer* automatisch neue Einträge parallel zur Fahrtrichtung generiert.

Find Best Direction: Der Algorithmus hat das Bestreben, alle Bahnen möglichst parallel zur Ausgangsbahn zu legen, wobei die Hauptrichtung im wesentlichen beibehalten wird. Da dies z.B. bei engen Passagen zu einer wenig optimalen Bahnplanung führt, versucht das Modul *Find Best Direction* bei Unterschreiten einer definierten Mindestlänge eines Stackeintrages, die nachfolgenden Bahnen günstiger im Raum anzuordnen. Unter einer optimalen Richtung wird diejenige verstanden, bei deren Einhaltung der Roboter die wenigsten Drehungen durchführen muß und gleichzeitig im Mittel die längsten Bahnen fahren kann. Da sich diese Fragestellung unter der gegebenen Sensorkonfiguration nicht optimal lösen läßt, wurde zunächst eine Heuristik definiert, welche die Richtung derart bestimmt, daß die Länge der nächsten abzufahrenden Bahn maximiert wird.

3.2 Aktuatorische Verhaltensweisen

Drive: Das komplexe Verhalten *Drive* dient der Abstraktion von den drei Verhaltensweisen *Drive Straight*, *Follow Track* und *Follow Wall*. Es realisiert eine möglichst geradlinige Fahrt an der eigenen Spur entlang bis zum Auftreffen auf ein Hindernis.

Die Fahrt wird in zwei Phasen unterteilt: Während der ersten Phase, in der sich der Roboter zwischen Anfangs- und Endpunkt der aktuellen Strecke befindet, sind die Komponenten *Follow Track* und, falls ein Hindernis passiert wird, *Follow Wall* aktiv. Sie garantieren ein nahtloses Aneinanderfügen der Reinigungsbahnen, indem der Abstand zur bisherigen Spur bzw. zu einem in die Bahn ragenden Hindernis mit einem klassischen P-Regler geregelt wird.

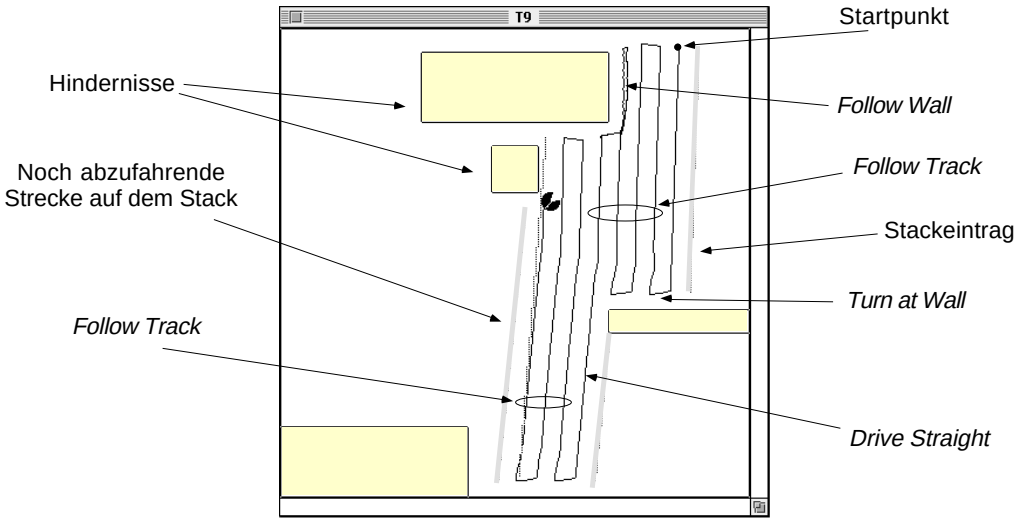

Abbildung 2: Einige der aktuatorischen Verhaltensweisen

Wurde der Endpunkt der aktuellen Strecke passiert, so wird auf die zweite Phase mit *Drive Straight* umgeschaltet. Der Roboter fährt nun solange geradeaus, bis er auf ein Hindernis stößt, wodurch gleichzeitig das Verhalten *Drive* beendet wird. Befindet sich *Drive* in dieser Phase, so kann es von den Verhalten *Turn at Wall*, *Turn at Track* oder *Drive to Next Area* unterbrochen werden.

Turn at Wall übernimmt die Kontrolle genau dann, wenn ein Hindernis den Weg blockiert und eine noch nicht befahrene Nachbarbahn existiert. Es veranlaßt den Roboter, der Kontur des Hindernisses in Richtung der Nachbarbahn zu folgen und sich bei Erreichen dieser entlang ihr auszurichten.

Turn at Track erfüllt eine ähnliche Funktion, falls der Roboter auf bereits bearbeitetes Gebiet trifft. Dabei handelt es sich nicht wirklich um ein aktuatorisches Verhalten, sondern um einen Watchdog, der auf *Drive to Next Area* umschaltet, sobald der Roboter in seiner ganzen Breite auf eine bereits bearbeitete Fläche stößt.

Drive to Next Area: Nachdem ein Teilbereich komplett abgefahren wurde, und falls keine direkte Möglichkeit zur Weiterfahrt existiert, muß ein neues Ziel vom Stack geholt und angefahren werden. Diese Aufgabe übernimmt *Drive to Next Area*, indem es sowohl die Bahnplanung als auch die Bewegungssteuerung durchführt.

Die Wegeplanung mittels Gradientenabstieg auf einem Distance Transform-Feld erfolgt ausschließlich auf bereits befahrenem Gebiet, da das zugrundeliegende Umgebungsmodell in der Regel noch unvollständig ist. Nur so läßt sich in einer statischen Umgebung die Erreichbarkeit garantieren. Darüber hinaus berücksichtigt das Verfahren dynamische Hindernisse, die sich auf bereits befahrenem Gebiet befinden, indem es diese während der Fahrt als temporäre Hindernisse in der Navigationskarte vermerkt. Versperren solche Hindernisse den Zielpfad, so erfolgt eine Neuplanung unter Berücksichtigung der veränderten Situation. Mehrfache erfolglose Anfahrtsversuche führen dazu, daß die entsprechende Zielstrecke an das Ende des Stacks zurückgelegt wird. Unter Umständen läßt sich das Ziel zu einem späteren Zeitpunkt wieder erreichen. Weitere Fehlversuche führen zum Löschen des Eintrages, so daß die Terminierung gewährleistet bleibt.

Durch das Zusammenspiel dieser einfachen Verhaltensweisen wird eine lokal-optimale Aktionssteuerung realisiert. Ein übergeordneter Planer, der ein global-optimales Verhalten garantieren würde, kann aufgrund der erst während der Task-Ausführung gewonnenen Umweltinformationen prinzipiell nicht zum Einsatz kommen. Bild 2 erläutert die Verhaltensweisen.

4. Experimentelle Ergebnisse

Das Verfahren wurde in einer Simulationsumgebung auf seine prinzipielle Anwendbarkeit hin untersucht. Es wurde ein Roboter mit differentiellem Antrieb und einem Durchmesser von 40 cm in einem maximal 20 x 20 m^2 großen Einsatzgebiet betrachtet. Die Auflösung der Rasterkarten betrug 14 cm für die Navigationskarte respektive 1 cm für das Umweltmodell. Neben einem taktilen Sensor (Bumper) wurde ein IR-Sensorsystem emuliert, welches einen 210°-Scan mit beschränkter Reichweite (1-2 m) und einer Winkelauflösung von 10° durchführt.

Eine definierte, gewollte Überlappung der Reinigungsbahnen braucht für den Algorithmus nicht spezifiziert zu werden. Es reicht aus, den überfahrenen Bereich etwas kleiner als den tatsächlich Roboterradius anzunehmen.

Die Simulation hat gezeigt, daß der Algorithmus bereits in dieser ersten, einfachen Version gute Ergebnisse liefert. So wird in dem in Abb. 3 gezeigten Szenario eine Flächendeckung von ca. 89% erreicht; die übrigen 11% werden im wesentlichen aufgrund der einstellbaren Sicherheitsabstände ausgespart. Von der insgesamt befahrenen Fläche wurden 85.8% genau einmal, 12.4% genau zweimal und die verbleibenden 1.8% drei- oder mehrmals befahren. Die Überdeckungen sind einerseits auf die backtracking Strategie und andererseits auf das Befahren von Randbereichen sowie auf die Ausweichmanöver bei Hindernissen zurückzuführen. Die beiden letzten Punkte lassen sich auch bei anderen Bewegungsstrategien nicht vermeiden. Abbildungen 4 und 5 zeigen weitere Beispielfahrten mit ähnlichen Ergebnissen.

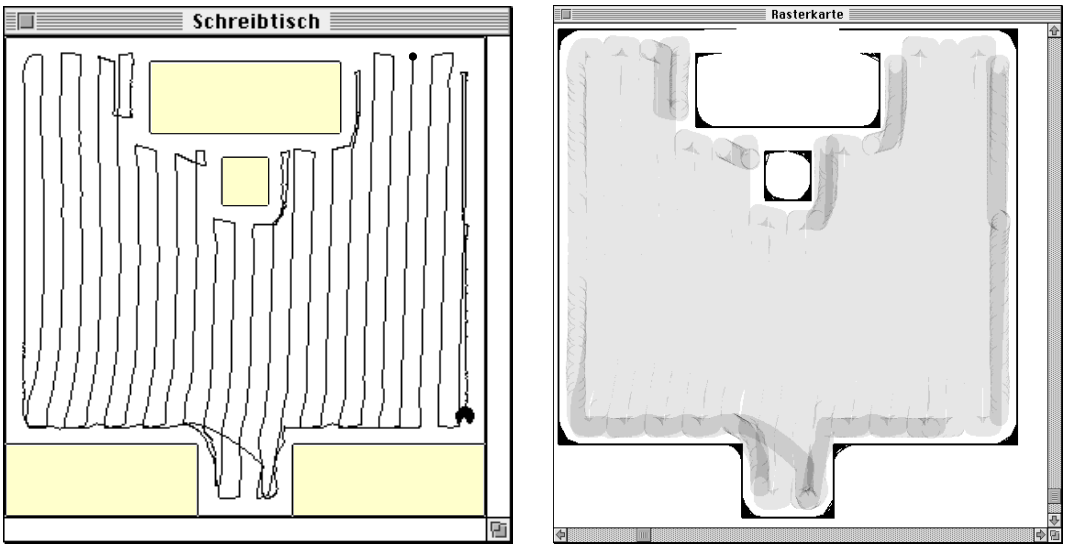

Abbildung 3: Ein einfaches Szenario. Links die gefahrene Trajektorie, rechts die Rasterkarte des Umweltmodells

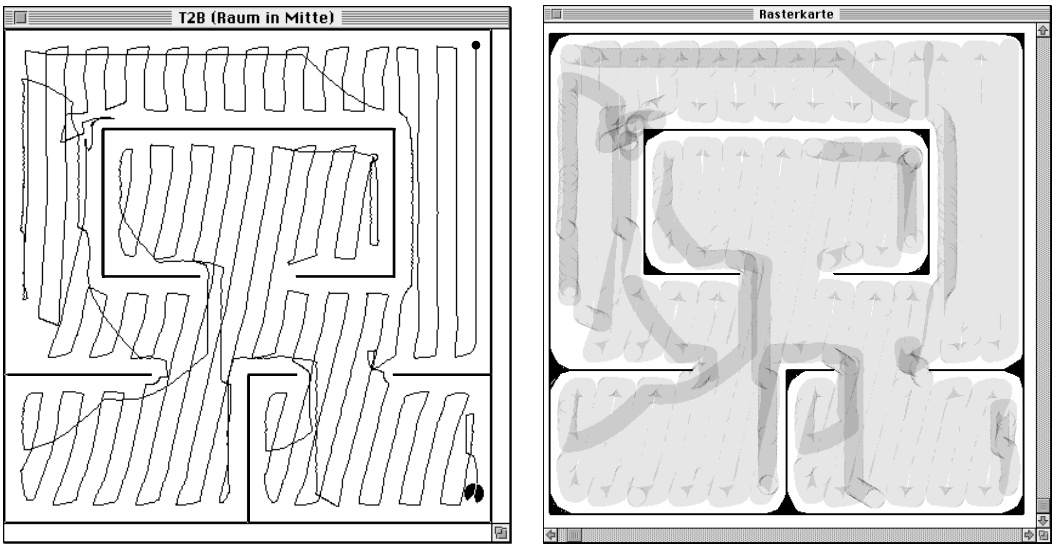

Abbildung 4: Ein komplexeres Szenario

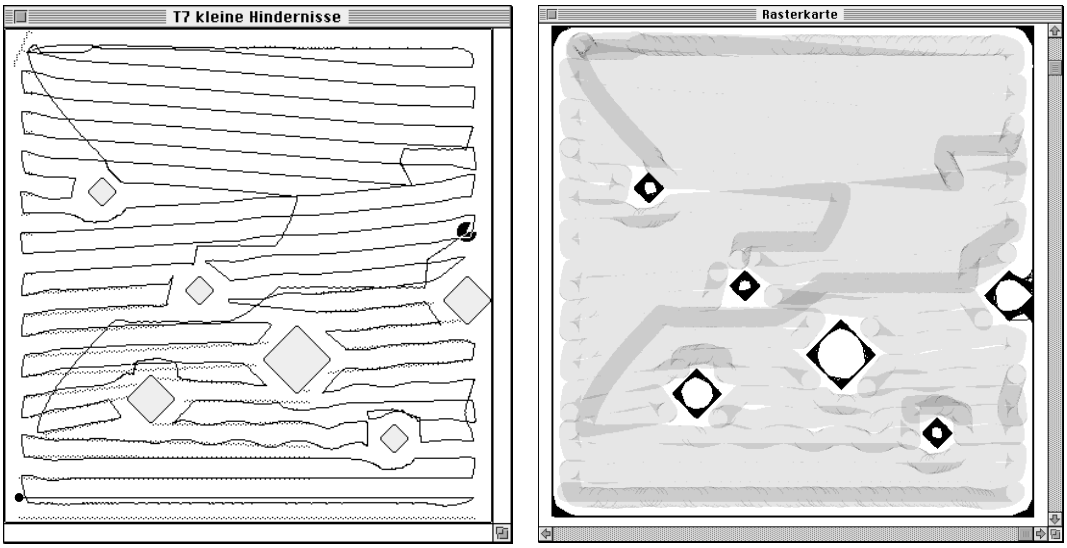

Abbildung 5: Szenario mit mehreren kleinen Hindernissen

Da eine erfolgreiche Simulation allenfalls eine notwendige, jedoch keinesfalls hinreichende Bedingung für einen erfolgreichen realen Einsatz darstellt, wird die abschließende Bewertung des Verfahrens mit einem realen Fahrzeug erfolgen.

5. Zusammenfassung und Ausblick

Es wurde eine erfolgversprechende, verhaltensorientierte Systemstruktur zum flächendeckenden Fahren in a priori unbekannter Umgebung vorgestellt. Bis zur Validierung der Ergebnisse durch ein reales Experimentalsystem - welches sich bereits in der Implementierungsphase befindet - sind noch einige Detailfragen zu klären. Insbesondere die Einbindung eines probabilistischen Umweltmodells ist für den tatsächlichen Einsatz unerläßlich.

Dem kommt der modulare Aufbau des verhaltensorientierten Ansatzes entgegen, der das gezielte Austauschen der entsprechenden Verhaltenskomponente erlaubt. Gleichzeitig bietet er das Potential für Verbesserungen durch die Erweiterung bestehender bzw. die Hinzunahme neuer Verhaltensschichten ohne Beeinträchtigung der bisher existierenden Module. In der Regel muß lediglich die Verhaltensauswahl an die neue Komponente angepaßt werden.

Neben der expliziten Programmierung stellt das Erlernen von Verhaltensmodulen eine interessante Alternative dar, welche insbesondere bzgl. der aktuatorischen Verhaltensweisen in zukünftige Entwicklungen miteinbezogen werden soll.

Literatur

[Arkin 98] Arkin, R.: *Behavior-Based Robotics (Intelligent Robots and Autonomous Agents)*, MIT-Press, 1998

[Brooks 86] Brooks, R. A.: *A Robust Layered Control System for a Mobile Robot*; IEEE Journal of Robotics & Automation, 2(1), 1986

[Burhanpurkar 94] Burhanpurkar, V.P.: *Real World Application of a Low-Cost High-Performance Sensor System for Autonomous Mobile Robots*; Proceedings of the International Conference on Intelligent Robots and Systems 1994 (IROS), München, Germany, 1994

[Electrolux 97] Pressemitteilung der Firma Electrolux zur Vorstellung eines autonomen Bodenreinigungsroboters; *http://www.electrolux.se/corporate/pressnotes/274a_4b2.htm*; siehe auch *http://www.electrolux.se/robot/;* Dezember 1997

[Furuhashi et al. 92] Furuhashi, H. et al: *Development of an Autonomous Cleaning Robot with an External Power Cable*; Proceedings of the International Conference on Motion and Vibration Control, Yokohama, Japan, 1992

[Hako 94] Hako -Werke GmbH & Co.: *Vollautomatischer Reinigungsroboter Hako-Robomatic 80 für die Reinigung großflächiger Hartböden*, Bad Oldesloe, 1994

[Hofner 97] Hofner, C.: *Automatische Kursplanung und Fahrzeugführung für mobile Roboter bei flächendeckenden Bearbeitungsaufgaben*; Dissertation am Lehrstuhl für Steuerungs- und Regelungstechnik, TU München, 1997

[Jörg 94] Jörg, K.-W.: *Echtzeitfähige Multisensorintegration für autonome mobile Roboter*; BI-Wissenschaftsverlag, Mannheim, 1994

[Kent 98] The Kent Company: Spezifikationen zum RoboKent ScrubberVac und SweeperVac; Elkhart, 1998; siehe auch *http://www.robokent.com/*

[Lawitzky et al. 98] Endres, H.; Feiten, W., Lawitzky,G.: *Field Test of a Navigation System: Autonomous Cleaning in Supermarkets*; Proceedings of the 1998 IEEE International Conference on Robotics & Automation (ICRA), Leuven, Belgium, May 1998

[Lu und Milios 97] Lu, F.; Milios, E.: *Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans*; Journal of Intelligent and Robotic Systems. vol. 18, pp. 249-275, 1997

[Rosenblatt 97] Rosenblatt, J.: *A Distributed Architecture for Mobile Navigation;* Journal of Experimental and Theoretical Artificial Intelligence, vol. 9, no. 2/3, pp. 339-360, April-September 1997

[Schraft und Schmierer 98] Schraft, R. D.; Schmierer, G.: *Serviceroboter: Produkte, Szenarien, Visionen*; Springer, 1998

[Thrun et al. 98] Thrun, S.; Fox, D.; Burgard, W.: *Probabilistic Mapping Of An Environment By A Mobile Robot*; Proceedings of the 1998 IEEE International Conference on Robotics & Automation (ICRA), Leuven, Belgium, May 1998

[Weiß et al. 94] Weiß, G.; Wetzler, C.; von Puttkamer, E.: *Positions- und Orientierungsbestimmung von bewegten Systemen in Gebäuden durch Korrelation von Laserradardaten*; Autonome Mobile Systeme, 10. Fachgespräch Stuttgart, 13.-14. Oktober 1994