3D-Umweltmodellierung zur Navigationsunterstützung von Laufmaschinen im Gelände

Diplomarbeit

Gruppe Interaktive Diagnose- und Servicesysteme
Forschungszentrum Informatik
an der
Universität Karlsruhe (TH)

von

Lutz Frommberger

Tag der Ausgabe : 15. Mai 2002
Tag der Abgabe : 14. November 2002

Betreuer : Dipl.-Inform. Bernd Gaßmann
Referent : Prof. Dr.-Ing. Rüdiger Dillmann
Koreferent : Prof. Dr.-Ing. Detlef Schmid

Ich erkläre hiermit, die vorliegende Diplomarbeit selbstständig verfasst zu haben. Die verwendeten Quellen und Hilfsmittel sind im Text kenntlich gemacht und im Literaturverzeichnis vollständig aufgeführt.

Karlsruhe, November 2002

Danksagung

An dieser Stelle möchte ich kurz ein paar Worte an diejenigen verlieren, ohne die diese Arbeit jetzt nicht in dieser Form vorliegen würde. Mein Dank gilt zunächst meinen Eltern und Großeltern für die nicht nur finanzielle Unterstützung in den langen Jahren meines Studiums, meinen Kommilitoninnen und Kommilitonen für alle Anregungen, Diskussionen und Korrekturgänge, meinen Mitbewohnern und vor allem meiner Freundin Michaela für ihr Verständnis für meine Schrulligkeit in den letzten Wochen vor der Abgabe, allen Mitarbeitern und Studierenden am IDS für die stets angenehme und produktive Atmosphäre sowie insbesondere meinem Betreuer Bernd Gaßmann für seine mitunter auf Kosten des Familienlebens gehende Unterstützung. Danke schön.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Mit Konstruktion von Laufmaschinen befassen sich Forscher in allen Teilen der Welt schon seit sehr langer Zeit. Bereits im 4. Jahrhundert wurde Überlieferungen zufolge in China das Konzept einer laufenden Maschine für den Lastentransport im Gelände angedacht, und auch die meisten der Entwürfe für Roboter in den letzten 300 Jahren zeigten laufende Maschinen. Ungeachtet ihrer Popularität stellen Laufmaschinen in der heutigen Robotik nur einen Randbereich dar: Ihre Entwicklungskosten sind hoch, ihre Mechanik aufwändig und die Steuerung komplex. Dennoch, oder gerade deshalb, stellen sich Wissenschaftler in den letzten Jahren verstärkt der Aufgabe der Forschung an dieser Art von Robotern.

Die Gruppe Interaktive Diagnose- und Servicesysteme (IDS) am Forschungszentrum Informatik (FZI) in Karlsruhe beschäftigt sich seit mehr als 10 Jahren mit der Entwicklung von Laufmaschinen. Eine der ersten Entwicklungen, der der Stabheuschrecke nachempfundene sechsbeinige Laufroboter LAURON, wurde in dieser Zeit sowohl mechanisch als auch steuerungstechnisch stetig weiter entwickelt und stellt heute eine sehr robuste und flexible Roboterplattform dar. Er ist in der Lage, sich in unstrukturiertem Gelände ohne allzu große Probleme fortzubewegen und kann auch größere Hindernisse sicher überwinden. Um so mehr verwundert es, dass all dies geschieht, ohne dass LAURON die ihn umgebende Welt nennenswert wahrnimmt: Die auf seinem Kopf montierten Kameras dienen nur Visualierungszwecken, allein ein Infrarot-Distanzsensor an seinem Rumpf wird zur Regelung der Körperhöhe herangezogen; eine weiter gehende Erfassung der Umwelt oder gar die Erstellung eines Umweltmodells findet nicht statt. Hindernisse überwindet der Roboter, indem er auf Kollisionen „wartet“ und sich dann förmlich durch das Gelände tastet.

Die Lebewesen, denen Laufmaschinen nachempfunden sind, passen ihr Laufverhalten im Allgemeinen den Gegebenheiten der Natur an, sie besitzen ein Modell der Umwelt, die sie durch ihre Sinnesorgane wahrnehmen. Es steht außer Zweifel, dass man auch bei Laufmaschinen in der Lage sein sollte, durch geeignete Erfassung, Speicherung und Auswertung von Sensorinformationen das Laufverhalten der Maschine signifikant zu verbessern. Auch steht mit einem Weltmodell auch die Möglichkeit zur Lösung weiter gehender Probleme wie der Pfadplanung offen.

Für Umweltmodelle gibt es bereits eine Vielzahl von Konzeptionen und Realisierungen, nur sind sie zumeist für die Arbeit mit radgetriebenen Robotern bestimmt. Umfassende Ansätze zur Weltmodellierung für Laufmaschinen gibt es bislang überhaupt nicht. Dabei sind die Anforderungen, die ein beingetriebener Roboter an ein Weltmodell hat, komplexerer Natur: Laufmaschinen sollen Hindernisse nicht nur vermeiden, sondern auch überwinden können. Zudem ist die Planung und Ausführung jedes einzelnen Schritts bereits von der Beschaffenheit der Umgebung abhängig. Dies sind Probleme, die in den bisherigen Ansätzen zur Umweltmodellierung nicht oder nur sehr rudimentär beachtet worden sind.

In einer Bewertung der unterschiedlichen Teilprobleme bei der Entwicklung von Laufmaschinen hat Karsten Berns den Punkt „Sensorauswertung für die Erfassung der Umwelt“ als einzigen mit dem höchsten Schwierigkeitsgrad („weitgehend ungelöst“) versehen (Berns, 1997/98). Die Entwicklung einer Umweltmodellierung, die sich diesem Problem stellt und für die Verwendung mit Laufmaschinen geeignet ist, ist also eine große Herausforderung, die aber für die Steuerung dieser Roboter einen erheblichen Nutzen verspricht. Die vorliegende Arbeit möchte sich dieser Aufgabe stellen.

1.2 Ziel der Arbeit

Ziel der Arbeit ist es, eine Umweltmodellierung zu entwerfen, die für die Unterstützung der Navigation von Laufmaschinen im Gelände verwendbar ist. Dazu soll unter kritischer Betrachtung der existierenden Ansätze zu diesem Thema eine Repräsentation der Umwelt gefunden werden, die den speziellen Anforderungen gerecht wird, die ein Laufroboter an sie stellt. Insbesondere ist dabei das Problem zu beachten, dass die Maschine jeden einzelnen Schritt geplant setzen muss und dass ihr Einsatzgebiet unstrukturiertes Gelände ist. Die zu erarbeitende Realisierung soll zudem so effizient und Speicher sparend arbeiten, dass sie unter Einhaltung der Echtzeitbedingung auf Kleincomputern laufen kann, die auf dem Roboter angebracht sind. Das Weltmodell soll dabei in die am IDS verwendete Steuerungsarchitektur MCA2 eingebunden und so entworfen werden, dass es möglichst unabhängig von einem konkreten Robotertyp funktioniert, um eine leichte Anwendung in unterschiedlichen Maschinensteuerungen zu ermöglichen. Zudem soll es mögliche Erweiterungen bereits in die Planung einbeziehen.

Für den Abgleich neu gewonnener Sensordaten mit dem schon existierenden Wissen soll ein geeignetes Verfahren gefunden werden. Mit dessen Hilfe soll die Umweltinformation effizient gespeichert und Karten der Umgebung mit verschiedenem Abstraktionsgrad erstellt werden können. Zudem soll ein Weg gefunden werden, das erworbene Wissen geeignet auszuwerten und so das Laufverhalten des Roboters zu verbessern, indem die Planung der Schritte mit Hilfe des Wissens über die Umgebung modifiziert wird. Die Praxiseignung der gefundenen Lösungen soll durch Anpassung der Steuerung an das Weltmodell an der Laufmaschine LAURON nachgewiesen werden.

1.3 Aufbau der Arbeit

In Kapitel 2 wird zunächst die Arbeit in einen größeren Kontext eingeordnet. Dazu werden kurz die Begriffe Wahrnehmung und Bewusstsein aus philosophischer Sicht beleuchtet und in einen Bezug zu Robotern und ihrer Umwelterfassung gesetzt. Danach wird allgemein auf autonome Systeme, insbesondere Laufmaschinen, und die besondere Bedeutung der Wahrnehmung der Welt bei ihnen eingegangen.

Kapitel 3 gibt zunächst eine Definition des Begriffs „Weltmodell“ und betrachtet die Anforderungen, die an ein solches gestellt werden. Es werden verschiedene Ansätze der Repräsentation von Umweltwissen vorgestellt und verglichen. Nach der Formulierung weiterer Anforderungen, die sich durch die Verwendung von Laufmaschinen im Gelände ergeben, wird unter deren Berücksichtigung ein geeigneter Repräsentationsansatz ausgewählt. Abschließend werden exemplarisch einige Realisierungen von Weltmodellen vorgestellt.

Kapitel 4 erläutert nun die Konzeption des erstellten 3D-Umweltmodells. Zunächst wird ein mehrschichtiger Aufbau beschrieben, mit dem maschinenunabhängig verschieden dimensionierte Karten der Umwelt verwaltet werden können. Mit dem *Erweiterten Inferenz-Gitter* wird eine Repräsentation vorgestellt, die den genannten Anforderungen entspricht. Zudem wird eine Datenstruktur zur Speicherung der Umweltdaten beschrieben. Im Anschluss daran wird ein auf Fuzzy-Logik basierendes Verfahren zum Einfügen von Sensordaten in den bestehenden Datenbestand eingeführt.

In Kapitel 5 wird zunächst die Laufmaschine LAURON beschrieben, auf der das entworfene Weltmodell zunächst angewendet wird. Es wird erläutert, wie mit Hilfe der entwickelten Methoden Sensordaten in die Umweltrepräsentation übernommen werden. Im Anschluss daran wird ein Verfahren vorgestellt, mit dem durch Erzeugung lokaler Umgebungskarten Fußaufsetzpunkte für die Maschine gefunden und unzureichend kartografierte Bereiche ermittelt werden können. Weiterhin wird aufgezeigt, auf welche Weise gröbere Karten aus dem Datenbestand erzeugt werden. Mit einer Beschreibung der Einbindung des Modells in die Steuerungsarchitektur MCA2 und der verwendeten Visualisierungsmechanismen endet dieses Kapitel.

Die Praxistauglichkeit der vorgestellten Verfahrens wird durch einige Experimente in Kapitel 6 untersucht. Kapitel 7 schließt die Arbeit mit einer Zusammenfassung der Ergebnisse und einem Ausblick ab.

Kapitel 2

Einordnung der Arbeit

Dieses Kapitel soll als Hinführung zum Thema dieser Arbeit dienen. Es befasst sich mit Robotern, deren Bezug zur Umwelt und den daraus erwachsenen Konsequenzen und stellt das Thema der Arbeit damit in einen allgemeineren Kontext.

2.1 Roboter, Bewusstsein und Wahrnehmung

Roboter kennt sprichwörtlich jedes Kind: In der Science-Fiction-Literatur haben sie ihren festen Platz und sind aus dieser nicht mehr wegzudenken. Dort hat der Begriff „Roboter“ auch seinen Ursprung: Er entstammt der Übersetzung des Theaterstücks „Rossum's Universal Robot“ von Karel Capek. Seitdem beflügeln diese Maschinen die Fantasie von Schriftstellern, Filmemachern und deren Publikum. Diese fiktiven Roboter erlangen eine Berühmtheit, die an die menschlicher literarischer Figuren zumindest heran reicht: die Roboter-Maria aus Fritz Langs Stummfilm „Metropolis“, die tollpatschigen C-3PO und R2-D2 aus „Star Wars“, der deprimierte Marvin aus „Per Anhalter durch die Galaxis“ oder der Bier trinkende Bender aus der Fernsehserie „Futurama“ – alle sind sie quasi zu Weltruhm gekommen.

Die hier genannten Roboterfiguren sind stets bewusst handelnde, intelligente Individuen, die sich nur wenig vom Menschen unterscheiden. Von der Entwicklung einer Maschine, der man ein Bewusstsein zusprechen könnte, ist die reale Forschung allerdings noch weit entfernt. Nicht zuletzt liegt das auch an der weitestgehend ungeklärten Frage, was nun Denken und Bewusstsein wirklich ausmacht. Es gibt in der Philosophie keine objektiv gültige Definition von Bewusstsein, im Gegenteil wird zumeist ausgeschlossen, dass es überhaupt möglich ist, eine solche zu finden. Vielfach wird gar die These vertreten, die Frage nach dem Wesen des Bewusstseins, das sog. „Leib-Seele-Problem“, sei nur ein Scheinproblem (vgl. Gierer (1989)).

Relative Einigkeit herrscht allerdings in der Sichtweise, dass „echte“ Intelligenz auch Bewusstsein voraussetzt (Gierer, 1989; Penrose, 1991). Was sich in den Erklärungsversuchen auch stets finden lässt, ist die Auffassung, dass für Bewusstsein zumindest Interaktion mit der Umwelt notwendig ist. Ernst Pöppel formuliert bspw.: „Bewusstsein steht [...] immer in einem sozialen Rahmen. Ohne andere gibt es kein Bewusstsein.“ (Pöppel, 1985)

Abbildung 2.1: Die Roboter-Maria aus Fritz Langs „Metropolis“

Aus diesem Minimalkonsens kann die Informatik zumindest die Information gewinnen, dass als Voraussetzung auf dem Weg zu einer bewussten Maschine mit Sicherheit eine geeignete Wahrnehmung der Umwelt notwendig ist. Wo die Umwelt nicht wahrgenommen werden kann, kann auch keine Interaktion mit ihr stattfinden. Kommunikation benötigt immer zwei Kanäle, insbesondere also auch einen Eingangskanal.

Inwieweit diese Überlegungen zum Bewusstsein für die Architektur von Steuerungssystemen für Roboter relevant sind, ist allerdings fraglich. Laut dem bekannten Turing-Test kann man eine Maschine dann als denkend betrachten, wenn ein menschlicher Beobachter nicht zwischen ihren Reaktionen und denen eines Menschen unterscheiden kann (Turing, 1950). So macht es letztendlich in der Betrachtung der Maschine keinen Unterschied, ob ihre „Intelligenz“ nun echtem Bewusstsein entspringt oder ob es nur so aussieht. Genau dies bringt Marvin Minsky mit seiner bissigen Definition von Künstlicher Intelligenz auf den Punkt:

> „Artificial Intelligence is the science of making machines do things that would require intelligence if done by people.“[1] (Minsky, 1968)

Ob eine geeignete Wahrnehmung der Umwelt jemals dazu beitragen kann, eine wirklich bewusst denkende Maschine zu konstruieren, ist mehr als zweifelhaft, auch wenn es sicher ist, dass umgekehrt dieses Ziel ohne die Erfüllung dieser Voraussetzung niemals erreicht werden wird. Die Fähigkeit der Maschine allerdings, wenigstens so zu wirken, als hätte sie Bewusstsein und würde daraus ihre Entscheidungen treffen, sich also souverän und angemessen in ihrem Umfeld zu bewegen, wird auf jeden Fall von der Modellierung ihrer Umweltwahrnehmung abhängen. Diese Herausforderung ist für die

[1] Künstliche Intelligenz ist die Wissenschaft, in der es darum geht, Maschinen dazu zu bringen, dass sie Dinge tun, für deren Ausführung ein Mensch Intelligenz benötigen würde.

Informatik schon groß genug. Der Satz, der das zehnbändige „Handbook of Perception“ eröffnet und der diese Betrachtung abschließen soll, gilt sicherlich für die menschliche Wahrnehmungsforschung genauso wie für die Robotik:

> „Perception is a rich, diverse and difficult field.“[2] (Carterette und Friedman, 1978)

2.2 Autonomes Agieren und Bezug zur Umwelt

Wenn der viel zitierte Mann von der Straße an einen Roboter denkt, wird er im Normalfall nicht an einen industriellen Fertigungsroboter oder ein ferngesteuertes Vehikel denken, sondern an eine einem Lebewesen, meist einem Menschen, ähnelnde Maschine, die sich ohne räumliche Begrenzungen und ohne Einwirkung von Menschen frei in beliebiger Umgebung bewegen kann. Auch wenn dies im Allgemeinen als der Normalfall eines Roboters angesehen wird, stellen derartige Konstruktionen nur einen Teilbereich innerhalb der Robotik dar: die Autonomen Mobilen Systeme. Dies ist die Klasse von Robotern, mit denen sich diese Arbeit befassen wird. Im Folgenden soll stets davon ausgegangen werden, dass die besprochenen Roboter nicht stationär sind.

Autonome Maschinen dringen langsam aber sicher in das alltägliche Leben der Menschen vor. Besonders der Bereich der Service-Roboter wächst stetig an. Zwar befinden sich Menschheitsträume wie der mechanische Butler noch im Experimentalstadium, aber autonome Putz-Roboter bspw. stehen kurz vor der Serienreife und werden in Bürokomplexen und Hotels bereits erfolgreich eingesetzt.

Vor allem haben Autonome Systeme nicht haushaltsnahe Nischen besetzt. Eine wichtige Anwendung ist das Agieren an Orten, die für den Menschen nicht erreichbar sind. Klassische Beispiele sind bspw. das Erkunden von Schwachstellen in Kanalsystemen oder die Erforschung von Schiffswracks in großen Meerestiefen (Lågstad und Auran, 1996). Für militärische Zwecke sind solche Systeme bspw. als unbemannte Minenräum-Fahrzeuge interessant. Andere Fälle sind noch spektakulärer: Der hermetisch abgedichtete Nuklearreaktor von Tschernobyl wird mit Hilfe eines Autonomen Roboters auf Schwachstellen in der Versiegelung untersucht (Maimone et al., 1998). Und erst kürzlich erregte der (wenn auch nur eingeschränkt autonom agierende) Roboter „Pyramid Rover“ weltweit Aufmerksamkeit, als er, live vom Fernsehen übertragen und dennoch ohne Ergebnis, einen schmalen Schacht in der Cheops-Pyramide erforschte (Traufetter, 2002).

Den spektakulärsten Auftritt eines autonomen Roboters in den letzten Jahre hatte wahrscheinlich die mobile Marssonde „Sojourner“ der Pathfinder-Mars-Mission der NASA, die auf der Oberfläche des Planeten Gesteinsproben nahm und Fotos schoss (Stone, 1996). Zugleich rückte diese Mission auch einige übersteigerte Erwartungen zurecht. Wer den euphorischen Prophezeiungen in den unwissenschaftlichen Medien geglaubt hatte, wurde bitter enttäuscht: Statt Marsmenschen zu entdecken, kämpfte sich die Sonde mühsam von einem Stein zum andern. Aber auch viele realistische Erwartungen konnte Sojourner, zumindest aus Sicht der Robotik, nicht erfüllen: Sein sechsrädriger Antrieb erwies sich für die zerklüftete Marslandschaft nur als bedingt

[2] Wahrnehmung ist ein ergiebiges, mannigfaltiges und schwieriges Feld.

Paß gut auf!

Millionen zittern mit dem kleinen Marsroboter

Von Gero von Randow

So also sieht ein Abgesandter der Menschheit aus: wie eine Kreuzung aus Faxgerät und Solarmobil. Der Kundschafter namens *Sojourner* zeigte sich erst ein wenig verklemmt, doch nach einem Tribut an die goldene Regel der technischen Informatik („ausschalten, einschalten, funktioniert wieder") setzte sich das Marsfahrzeug in Bewegung. Das Schauspiel hatte am vergangenen Wochenende sein Weltpublikum: Ein nicht unbeträchtlicher Teil der Menschheit saß vor den Fernsehgeräten, um die Abenteuer eines anderen Geräts zu verfolgen. Wie zuvor bei *Deep Blue,* dem Schachcomputer.

Abbildung 2.2: Links: Ein Bild aus der Pathfinder-Mission: Der radgetriebene Roboter *Sojourner* kämpft mit der zerklüfteten Marslandschaft. Rechts: „Eine Kreuzung aus Faxgerät und Solarmobil": Die Hamburger Wochenzeitung „Die ZEIT" macht sich auf der Titelseite über das „kleine Mars-Tamagochi" lustig (von Randow, 1997).

geeignet, sein Aktionsradius umfasste nur wenige Meter pro Tag, und obwohl er wunderschöne, hoch aufgelöste Bilder der Umgebung auf die Erde funkte, fiel er mitunter tagelang aus, weil er dennoch einen herumliegenden Stein „übersehen" hatte und es zu einer Kollision kam. Die Presse quittierte diese augenscheinlichen Unbeholfenheiten mit sanftem Spott (Abbildung 2.2).

2.2.1 Laufmaschinen – autonome Roboter für den Einsatz in unstrukturiertem Gelände

Gerade der Anwendungsbereich von Sojourner ist ein Paradebeispiel für das Einsatzgebiet einer speziellen Gruppe autonomer Roboter, der Laufmaschinen, die sich nicht auf Rädern, sondern auf Beinen fortbewegen. Diese Art von Robotern ist für den Einsatz in zerklüfteten, unstrukturierten Landschaften besonders geeignet, da sie keine planen Flächen zur Fortbewegung benötigt. Konkret erhofft man sich folgende Vorteile durch die Verwendung von Laufmaschinen (Brandl, 1999):

- Größere Mobilität, vor allem im Gelände
- Geringere Geländeabhängigkeit und -zerstörung
- Bessere Entkopplung der Fortbewegung vom Gelände und damit gleichmäßigere Bewegung
- Höhere Geschwindigkeit und Effizienz in unstrukturiertem Terrain

Abbildung 2.3: Die Laufmaschine LAURON im Einsatz in einem Waldgebiet

Die meisten frühen Entwürfe und Konstruktionen von Robotern zeigen Laufmaschinen, ungeachtet der Tatsache, dass das Konzept „Laufen“ für den Menschen zwar leicht ausführbar, aber sehr schwer modellierbar ist (für einen Überblick über die historische Entwicklung siehe Berns und Fiegert (1991)). So befinden sich Laufroboter auch bei aktuellem Entwicklungsstand noch mehr oder weniger im Forschungsstadium und sind wegen ihrer hohen Entwicklungs- und Hardwarekosten noch recht selten und nur in wenigen Nischenbereichen im Praxiseinsatz. Dennoch ist es gerade die hervorragende Adaptionsfähigkeit an die Gegebenheiten der Umgebung, die Laufmaschinen für die Forschung weiter attraktiv macht. Einen umfassenden Überblick über den Stand der Technik bietet der Laufmaschinenkatalog von Karsten Berns (Berns, 2000). Eine kurze Darstellung der Laufmaschine LAURON (Abbildung 2.3), die im Rahmen dieser Arbeit verwendet wurde, findet sich in Abschnitt 5.1.

2.2.2 Umweltwahrnehmung als Voraussetzung autonomen Agierens

In Dillmann und Huck (1991) wird folgende Definition eines Autonomen Systems gegeben:

> „Ein Autonomes System ist ein informationsverarbeitendes System, welches die Fähigkeit hat, eine ihm gestellte Aufgabe auch unter Einwirkung nicht explizit vorgesehener Ereignisse und Umweltzustände erfolgreich durchzuführen.“

Die Einwirkung von Umweltzuständen ist also ein notwendiges Kriterium für das Vorhandenseins eines Autonomen Systems. Wie sollte ein Roboter auch selbstständ-

dig in seiner Umgebung operieren können, ohne sie in irgendeiner Weise wahrzunehmen? Autonome Systeme müssen also mit einer Sensorik als Schnittstelle zu ihrer Umwelt ausgestattet sein. Die Möglichkeiten sind vielfältig: Entfernungssensoren wie Ultraschall-, Infrarot-Sensoren oder Laser-Scanner, Kamera-Systeme, Kraftsensoren, Funk-Positionierungssysteme, Magnetfeldmesser ... die Liste der Möglichkeiten ist lang. Eine ausführliche Übersicht über Sensorik in der Robotik findet sich in Borenstein et al. (1996).

Mit der einfachen Aufnahme durch die Sensorik ist es jedoch nicht getan. Ein Kamerabild bspw. scheint für den Menschen eine erschöpfende Fülle von Informationen zu liefern – jedoch erstmal nur für den Menschen, der über eine beeindruckende Fähigkeit zur Echtzeit-Extraktion von Information aus visuellen Daten verfügt, über deren Mechanismen sich die Wahrnehmungsforschung immer noch den Kopf zerbricht. Merkmalsextraktion aus Kamerabildern ist für Computer dagegen immer noch eine komplizierte und rechenzeitaufwändige Aufgabe.

Welche der gewonnenen Daten sind relevant? In welcher Beziehung stehen sie zu anderen Messungen? Was sagen sie aus? Wie sind sie zeitlich einzuordnen? Und wo findet sich die Maschine in der Datenmenge wieder? Diese Fragen sind zu stellen, wenn ein Modell erzeugt werden soll, dass die Umwelt und ihre Beziehung zum Roboter repräsentiert.

Kapitel 3

Umweltmodellierung - Stand der Technik

> „Given the robot's position and a set of measurements, what are the sensors seeing?“[1] (Rencken, 1993)

Mit dieser Frage gibt Rencken eine Definition für ein Weltmodell. Dabei spielt er geschickt mit der Bedeutung des Wortes „sehen“ in der Alltagssprache. Wenn ein Mensch sagt „Ich sehe eine Tür“, dann gibt er damit keine reine Datenaufzeichnung seiner Sensorik wieder, sondern hat diese bereits ausgewertet und interpretiert. Die von den Augen aufgenommenen Lichtwellen wurden letztendlich mit dem bekannten Objekt Tür identifiziert. Wie auch immer der Mensch diese Daten im Gehirn aufbereitet und modelliert, er ist in der Lage, aus dieser Repräsentation Schlüsse über seinen Zustand und Pläne für sein weiteres Vorgehen abzuleiten. Aus der Erkenntnis „Ich sehe eine Tür“ kann er bspw. sofort die Aussage „Es gibt eine Möglichkeit, diesen Raum zu verlassen“ folgern.

Folgende Kernsätze zur Charakterisierung eines Umweltmodells werden in Dillmann und Huck (1991) formuliert:

- Ein informationsverarbeitendes System muss neben der Akquisition von Information durch Sensoren auch über die Möglichkeit zur Speicherung von Wissen verfügen.
- Sensoren dienen als Bindeglied zwischen den abstrakten Modellen einer Wissensbasis und der realen Umwelt, in welcher ein autonomes System operiert.
- Das Ziel ist, mit Hilfe des gespeicherten Wissens zu jedem Zeitpunkt ein genügend detailliertes Bild der realen Umwelt zu erhalten.

Es bleibt die Frage, welcher Art die Speicherung sein soll und auf welche Weise die Sensordaten in diese überführt werden. Diese Problematik ist Gegenstand der folgenden Abschnitte.

[1] Wenn die Position des Roboters und eine Anzahl Messdaten vorliegen, was sehen dann die Sensoren?

3.1 Grundlagen der Umweltmodellierung

In diesem Abschnitt sollen grundlegende Betrachtungen zu Ansätzen für Weltmodelle angestellt werden. Dabei soll jeweils der Bezug zur aktuellen Anwendung hergestellt werden. Am Ende soll die Wahl eines für diese Zwecke geeigneten Ansatzes stehen.

3.1.1 Anforderungen an ein Weltmodell

Um zu einer Speicherung von Wissen zu gelangen, das auf der durch die Sensorik ermittelten Information beruht, ist es zunächst nötig, sich über die Vorgehensweise dieser Datenverarbeitung klar zu werden. Drei Hauptschritte für die Verarbeitung von Sensordaten zu verwertbaren Karten werden in Hoppen et al. (1990) zusammengestellt:

1. Merkmalsextraktion aus den rohen Sensordaten
2. Fusion der Daten der verschiedenen Sensortypen
3. Automatische Generierung eines Umweltmodells mit verschiedenen Abstraktionsebenen

Da die Sensordaten nicht alle gleichzeitig ankommen, muss das Umweltmodell ständig mit den neuesten Informationen aus der Sensorik abgeglichen werden. Für die Aktualisierung einer bestehenden Datenbasis nennt Crowley (1989) ebenfalls ein dreischrittiges Vorgehen:

1. Aufbau einer abstrakten Beschreibung der neuesten Sensordaten (Sensormodell)
2. Ermittlung der Übereinstimmung zwischen dem neuesten Sensormodell und dem gegenwärtigen Inhalt des kombinierten lokalen Weltmodells
3. Veränderung der Komponenten des kombinierten lokalen Modells und Verstärkung oder Abschwächung von Vertraulichkeitswerten, um die Übereinstimmungen wiederzugeben

Mit den konkreten Verfahren zum Modifizieren der Weltmodelldaten und insbesondere der Modellierung von Vertraulichkeitswerten werden sich spätere Kapitel dieser Arbeit befassen. Zunächst ist die Frage zu klären, welche Art der Repräsentation für das Weltmodell gewählt werden soll und welche Faktoren dabei zu berücksichtigen sind.

3.1.2 Randbedingungen und Einschränkungen

Bei der Erstellung von Umweltmodellen unterliegt man einer Reihe von Einschränkungen (vgl. auch Thrun (1998)). Die Problemfelder lassen sich dabei in drei Klassen einteilen:

Wahrnehmungsschwierigkeiten Unter diese Klasse fallen alle Probleme, die sich bei der Erfassung der realen Welt ergeben. Dazu gehören sowohl Störrauschen und technische Begrenzungen der Sensoren als auch die Problematik, aus den Eingangsdaten genau die Information herauszulesen, die man braucht; oftmals

messen Sensoren eine Vielzahl von Werten, die das Erforderliche nicht unmittelbar liefern und erst nachbearbeitet werden müssen. Weiterhin umfasst diese Klasse Außeneinflüsse, die überhaupt nicht oder nur unvollkommen erfasst werden können. Als Beispiel sei hier rutschiger Untergrund genannt: Diese Störgröße hat starken Einfluss auf die Genauigkeit der Odometrie des Systems, ist aber schwer messbar und fällt als unmittelbar wahrnehmbare Größe aus.

Abstraktionsschwierigkeiten Hierunter lassen sich die Probleme subsummieren, die sich aus der Umwandlung der Sensordaten in ein Modell der Umwelt ergeben. Jede Wahl einer Repräsentation der Umwelt, jede Aufbereitung der Daten wird auch zu Verlust von Wissen führen; nicht jede Form der Abbildung wird dem zu lösenden Problem gerecht. Eine wirklich exakte und von der aktuellen Fragestellung unabhängige Projektion der realen Welt ist quasi unmöglich. Auf diese Problemklasse geht Kapitel 3.1.3 ausführlicher ein.

Verarbeitungsschwierigkeiten Übrig bleiben die Probleme, die beim Verarbeiten der gesammelten Daten entstehen. Es müssen Methoden gefunden werden, um die Sensordaten in die gewählte Repräsentation zu überführen, genauso müssen aus den aufbereiteten Daten Strategien zur Steuerung der Maschine gewonnen werden. All diese Schritte unterliegen der Einschränkung, in Echtzeit ablaufen zu müssen, um ein angemessenes Verhalten des Roboters ermöglichen zu können.

Diese auftretenden Schwierigkeiten pflanzen sich in die anderen Problemebenen fort: Je schlechter die Eingangsdaten sind, desto mehr Aufwand muss in ihre Aufbereitung gelegt werden und desto schwieriger wird die Wahl eines geeigneten Modells; je unglücklicher die Repräsentation gewählt ist, desto aufwändiger wird das Gewinnen von Strategien aus dem Datenbestand – und je mehr Aufwand jeweils betrieben werden muss, desto schwieriger ist das Einhalten der Echtzeitbedingung.

Vor allem die Folgen ungenauer oder gar falscher Sensordaten sind bis in die höchste Ebene wahrnehmbar: Filter müssen hinter die Sensorik geschaltet werden, die Repräsentation muss mit Fehlerhaftigkeit umgehen können und die Auswertung des Wissens muss stets die Möglichkeit von Fehlinformation betrachten – all diese Maßnahmen verbrauchen Rechenzeit. Es empfiehlt sich also insbesondere, in die Sensorik zu investieren, um ihre negativen Folgen möglichst gering zu halten.

3.1.3 Geometrische und Topologische Karten

Es gibt prinzipiell zwei verschiedene Möglichkeiten zur Erstellung von Umweltkarten: den geometrischen und den topologischen Ansatz (Engelson und McDermott, 1992).

Der *geometrische Ansatz* versucht, die Geometrie der Welt weitestgehend genauso zu beschreiben, wie sie durch die Sensorik wahrgenommen wird, indem sie auf geometrische Primitive herunter gebrochen wird: Eine wahrgenommene Kante wird bspw. durch eine Kante repräsentiert. Im Fall idealer, allumfassender Sensorik wird eine geometrische Karte eine relativ exakte visuelle Wiedergabe der realen Welt sein – genau das ist auch der große Vorteil dieses Ansatzes: Karte und Welt stehen in einer unmittelbaren, wohldefinierten Beziehung. Weiterhin sind auch Position und Orientierung des Roboters direkt in die Karte einfügbar. Ein Nachteil ist, dass es schwierig ist, eine zufrieden

stellende Repräsentation zu finden, die komplexe Geometrien gut abbildet und auch mit den Unsicherheiten und Einschränkungen der Sensorik in geeigneter Weise umgehen kann. Desweiteren ist der Nutzen der gesammelten Information nicht unmittelbar einsichtig, die geometrische Karte abstrahiert von der für das System aus ihr erwachsenden Anwendung. Zudem ist der geometrische Ansatz anfällig für Fehler, vor allem Ungenauigkeiten in der Positionsbestimmung des Roboters können verheerende Effekte verursachen. Nicht zuletzt stellen geometrische Modelle hohe Anforderungen an Speicher und Rechenzeit: Die Umwelt wird in all ihrer Detailhaltigkeit wie wahrgenommen abgebildet, unabhängig vom wahren Informationsgehalt dieser Einzelheiten.

Der *topologische Ansatz*, der bereits 1978 von B. Kuipers eingeführt wurde (Kuipers und Byun, 1991), stellt die gewonnenen Beobachtungen in direkte Beziehung zu den Aktionen des Roboters. Die Welt wird durch einen Graph dargestellt, dessen Knoten Orte der wahrgenommenen Welt sind, die mit Kanten verbunden sind, die mögliche Aktionen des Roboters darstellen. Dies eröffnet, „eine phänomenologische Repräsentation der möglichen Interaktionen des Roboters mit der Welt“ (Engelson und McDermott, 1992). Antworten auf weitergehende Fragestellungen, wie bspw. Routenplanung, lassen sich aus topologischen Karten direkt ableiten. Dadurch bleibt der Bedarf an Rechenzeit gering, zudem sind topologische Modelle aufgrund des hohen Abstraktionsgrades sparsam im Speicherverbrauch. Zur Speicherung werden vielfach Hidden-Markov-Modelle (HMMs) verwendet, die sich sehr effektiv und effizient auswerten lassen. Weiterhin sind topologische Modelle vergleichsweise resistent gegen Schwächen in der Odometrie des Systems, da sie statt metrischen Informationen nur Relationen zwischen Objekten der Umwelt halten. Das ist zugleich auch eine große Schwäche: Gewöhnliche Probleme wie das des Finden des „kürzesten Pfades“ lassen sich nicht ohne Weiteres lösen. Auch lässt sich die Position des Roboters nur dann aus der Karte ableiten, wenn er sich an einer „signifikanten Stelle“ befindet.

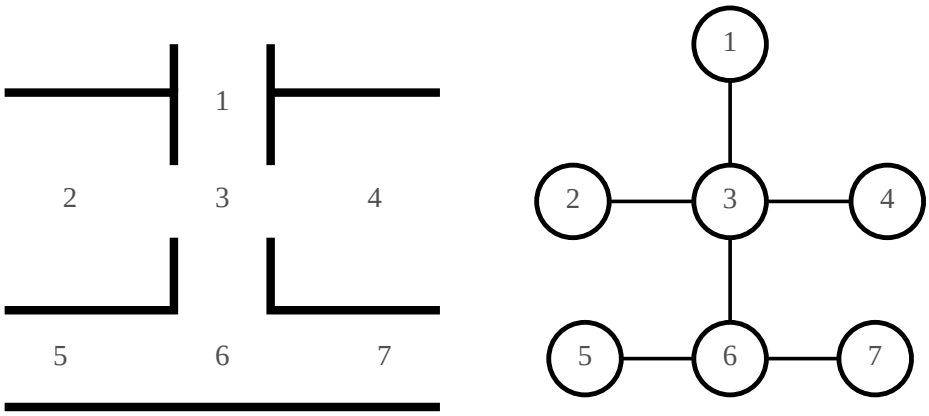


Abbildung 3.1: Beispiel einer geometrischen (links) und einer topologischen Karte (rechts): Die mit Nummern markierten Räume in der geometrischen Karte entsprechen den Knoten in der topologischen Darstellung, mögliche Wege sind durch Kanten repräsentiert.

Engelson und McDermott erweitern den topologischen Ansatz allerdings noch dahin gehend zur *diktiometrischen Repräsentation*, indem sie zu den in den Kanten gehaltenen Beziehungen zwischen den Orten geometrische Informationen hinzufügen. Die

Kombination topologischer Karten und odometrischer Daten hat sich als sehr gewinnbringend heraus gestellt, auch unter der Randbedingung schwacher Odometrie (Shatkay und Kaelbling, 1997).

Es ist leicht ersichtlich, dass zur Erstellung topologischer oder gar diktiometrischer Karten ein hohes Maß an Wissen über die Umwelt erforderlich ist: ein Hindernis, das in eine solche Karte eingetragen wird, muss erst einmal aus den Sensordaten und der bisher vorhandenen Information als ein solches klassifiziert werden, Zusammengehörigkeiten von verschiedenen Messungen erkannt werden. Diese Ansätze eignen sich daher für leicht erfassbares, insbesondere aber auch für im Vorfeld bekanntes Terrain, aus dem a-priori-Karten erstellt werden können. Für die Aufgabe der Umweltwahrnehmung eines mobilen autonomen Roboters ohne oder mit nicht detailgenauem a-priori-Wissen sind sie weniger geeignet, da die erforderliche Menge und Genauigkeit an Wissen über die Umgebung, vor allem unter Einsatz schwacher Sensorik, nicht erworben werden kann.

Für das geometrische Paradigma gibt es allerdings, außer den bereits genannten, auch solche Einschränkungen, die Alberto Elfes als die „Lücke zwischen zwei Informationsebenen“ bezeichnet (Elfes, 1989). Einerseits existiert die Ebene der unpräzisen und limitierten Sensordaten, andererseits die Ebene des abstrakten symbolischen und geometrischen Weltmodells, auf dem operiert wird und aus dem Information für das weitere Verhalten des autonomen Systems gewonnen wird. Elfes folgert daraus, dass geometrische Ansätze vor allem in gut strukturierten Umgebungen sinnvoll sind, für komplexere Szenarien aber nur beschränkte Eignung aufweisen. Topologische Karten haben damit allerdings auch Probleme, nur auf einer anderen Ebene: Mit der Komplexität der Umgebung steigt auch die der Karte und damit auch ihr Erstellungs- und Auswertungsaufwand.

Die wichtigsten Eigenschaften beider Ansätze bzgl. verschiedener Kriterien sind in Tabelle 3.2 zusammengefasst. Offensichtlich stehen die Vor- und Nachteile der Methoden jeweils orthogonal zueinander. Es empfiehlt sich also, beide Ansätze zu kombinieren, um die jeweiligen Stärken ausnutzen zu können. Eine nahe liegende Möglichkeit ist, auf der Grundlage einer geometrische Karte eine topologische Karte zu erzeugen, indem man erstere nach kohärenten Regionen durchsucht und daraus eine Topologie generiert (Thrun und Bücken, 1996). Simhons und Dudek erstellen topologische Karten, indem sie lokale geometrische Karten („islands of reliability“) in Relation zueinander setzen (Simhons und Dudek, 1998). Solche Vorgehensweisen lassen sich unter dem Slogan *„first metric, then topological“* zusammenfassen. Auch eine umgekehrte Vorgehensweise (*„first topological, then metric“*) wird praktiziert: Aus topologischen Karten kann mit Hilfe des EM-Algorithmus die wahrscheinlichste geometrische Repräsentation abgleitet werden (Thrun et al., 1998). Beide Ansätze erreichen das gleiche Ziel: Durch die gleichzeitige Verwendung zweier Karten kann sichergestellt werden, dass für jede auftauchende Problemstellung eine geeignete Umweltrepräsentation verfügbar ist.

Ein weiterer Ansatz der Kombination von topologischen und geometrischen Informationen, die *Variable Resolution Maps* (Arleo et al., 1999), wird in Abschnitt 4.4.2 behandelt.

	Geometrische Karten	Topologische Karten
Generierung	leicht zu erstellen und aktualisieren, Sensordaten sofort nach Messung eintragbar	viele Sensordaten und hohes Maß an Datenverarbeitung notwendig, Schwierigkeiten bei sehr großen Umgebungen
Speicherbedarf	extrem hoch	eher gering, abhängig von Komplexität der Umgebung
a-priori-Wissen	nützlich, aber nicht notwendig	nur bei sehr leistungsstarker Sensorik verzichtbar
komplexe Welt	Kartengröße unabhängig von Komplexität, aber Erkenntnisgewinn aus Karte aufwändig	Erzeugung sehr komplexer Karten
Roboterposition	hohe Genauigkeit bei der Erfassung und ständige Korrektur notwendig	nicht nötig ⇒ robust gegen Störungen wie Rutschen etc.
Wiedererkennung	absolute Position des Roboters kann direkt in Karte eingeordnet werden, Orte so leicht wiedererkannt werden	Wiedererkennung von Orten schwierig, vor allem, wenn sie auf verschiedenen Pfaden erreicht wurden; Probleme mit mobilen Objekten
Pfadplanung	sehr hoher Verarbeitungsaufwand, kürzester Pfad aber leicht ermittelbar	Pfadplanung direkt aus Karte ableitbar, aber evtl. suboptimale Wege durch fehlende metrische Information

Tabelle 3.1: Gegenüberstellung der Eigenschaften Geometrischer und Topologischer Karten

3.1.4 Büroflur oder Urwald? Zusätzliche Herausforderungen in unstrukturiertem Gelände

Folgt man Alberto Elfes in seiner im letzten Kapitel skizzierten Argumentation (Elfes, 1989), so gibt es überhaupt keinen geeigneten Ansatz, unstrukturiertes Gelände zu repräsentieren. Schon die sprachlichen Bezeichnungen spiegeln die Schwierigkeit wieder: Wie bekomme ich Unstrukturiertes in eine Daten*struktur*? Auch ein Blick durch die Veröffentlichungen der letzten Jahre über Umweltmodellierungen für autonome Roboter scheint die Problematik zu stützen: Das Anschauungs- und Beispielobjekt ist in fast allen Fällen der Gang vor dem eigenen Büro. Es scheint eine Tatsache zu sein, dass (vielleicht abgesehen vom Datenbestand der NASA und US Army) wohl nichts auf der Welt so detailgenau vermessen und kartografiert worden ist wie die Büroflure von Robotik-Instituten; eine andere Tatsache ist aber, dass die meisten Anwendungen für mobile Roboter eben nicht in selbigem Büroflur ihren Platz finden, sondern in Umgebungen, die sich nicht durch gerade Kanten und plane Flächen auszeichnen.

Um in einer solchen Umgebung bestehen zu können, sollte ein Umweltmodellierungsverfahren einige zusätzliche Anforderungen erfüllen:

Unabhängigkeit von Kanten Die Repräsentation der Umgebung sollte nicht auf der Detektion von Kanten oder vergleichbarer Strukturen beruhen. Zufrieden stellende Karten müssen generiert werden können, auch ohne dass strukturelle Informationen aus der Umgebung gewonnen werden, da nicht erwartet werden kann, dass diese überhaupt existieren.

Weitestgehende Unabhängigkeit von a-priori-Karten Vergleicht man ein Schotterfeld mit einem Büroflur, wird deutlich, wo das Problem bei unstrukturiertem Gelände liegt: Eine a-priori-Karte des Schotterfelds wird kaum so fein granuliert sein können, als dass die Informationen über die Wegbarkeit des Geländes ausreichend berücksichtigt werden können; das Gelände ist nicht homogen und damit nicht effizient kartierbar. Zudem können Steine leicht ihre Position ändern, während der Roboter sich dort bewegt. Als zusätzliche Informationsquelle können a-priori-Karten nützlich sein, aber das System sollte auch ohne sie gut funktionieren.

Robustheit gegen schwache Odometrie In natürlichen Umgebungen ist mit mehr Störeinflüssen zu rechnen als in geschlossenen Räumen: Rutschiger oder nachgebender Boden erschwert eine korrekte Odometrie. Die Umweltmodellierung sollte also nicht zu stark von ihrer Genauigkeit abhängen.

Behutsame Hindernisklassifikation Vieles, was von der Sensorik wahrgenommen wird, muss kein wirkliches Hindernis sein. Ein großer Grasbüschel bspw. kann von einer Laufmaschine mit Leichtigkeit überwunden werden. Wünschenswert wäre also ein skeptischer Umgang mit der Klassifikation „Hindernis“.

Zusätzlich wäre, zumindest für Karten in größerem Maßstab, eine wünschenswerte Anforderung:

Information über Wegbarkeit Während der Fortbewegung des Roboters soll Information darüber gewonnen werden, wie gut das derzeitige Gelände begehbar ist. Dies kann durch Auswertung von Sensordaten erkannt werden oder auch dadurch, dass die Prognose des Weltmodells nicht wie gewünscht umsetzbar war. Solches Wissen um „unsichere Stellen“ sollte in die Kartierung der Umwelt übernommen werden.

Wenn sich ein geeigneter Ansatz also nicht finden lässt, stellt sich aber die Frage nach dem am wenigsten ungeeigneten Ansatz. Elfes selbst führt als Antwort auf die von ihm benannten Einschränkungen geometrischer Weltmodelle in o. g. Artikel das *Belegtheitsgitter* ein.

3.1.5 Belegtheitsgitter

Die Idee des Belegtheitsgitters (*Occupancy Grid*, seltener auch *Certainty Grid*) wurde Ende der 80er Jahre des letzten Jahrhunderts von Elfes und Moravec propagiert (Elfes, 1989; Moravec, 1988). Es wurde als Repräsentation der Umwelt vor allem für unsichere und lückenhafte Sensorik vorgestellt. Die reale Welt wird dabei in ein zwei- oder dreidimensionales Gitter eingeteilt, in dem jede Gitterzelle alle in ihr enthaltenen Punkte im Raum repräsentiert. Die kontinuierliche Umgebung wird also diskretisiert, jedem wirklichen Punkt wird eine Gitterzelle zugeordnet, die eine probabilistische Aussage über den Zustand der ihr zugehörigen realen Region speichert. Dieser Zustand der Zelle entspricht in der vorgestellten Form ihrer Belegtheitsinformation, die entweder die binäre Information „frei oder belegt“ sein kann oder, wie üblicherweise verwendet, ein Wert $P(C_i = occ) \in [0, 1]$, der die Wahrscheinlichkeit angibt, mit der die Zelle belegt (*occupied*) ist. Gitter, die noch weitergehende Daten als den Belegtheitsstatus halten, nennt Elfes *Inferenzgitter*.

Bei der Aktualisierung der Zellinhalte wird aus einer Anzahl von Sensormessungen eine jeweils neue Belegtheitswahrscheinlichkeit nach dem Satz von Bayes errechnet. Ausgehend von einer Reihe von Sensormessungen $\{r\}_t = \{r_0, r_1, \ldots, r_t\}$ und der daraus geschätzten Belegtheitswahrscheinlichkeit für die Zelle C_i, $P(s(C_i) = occ|\{r\}_t)$, kann eine neue Schätzung $P(s(C_i) = occ|\{r\}_{t+1})$ nach einer weiteren Messung r_{t+1} rekursiv gewonnen werden:

$$P(s(C_i) = occ|\{r\}_{t+1}) \quad = \quad \frac{p(r_{t+1}|s(C_i) = occ)P(s(C_i) = occ|\{r\}_t)}{\sum\limits_{s(C_i)} p(r_{t+1}|s(C_i))P(s(C_i)|\{r\}_t)}$$

Die Wahrscheinlichkeitsverteilung $p(r_{t+1}|s(C_i))$ ergibt sich aus dem Sensormodell, der Prior $P(s(C_i)|\{r\}_t)$ kann direkt aus dem Belegtheitsgitter abgelesen werden. Betrachtet werden dabei alle Zellen C_i im gesamt möglichen Erfassungsbereich der jeweiligen Messung. Dieser wahrscheinlichkeitstheoretische Ansatz erklärt sich auch aus den zur Zeit der Einführung der Belegtheitsgitter vorwiegend verwendeten Ultraschallsensoren. Diese weisen einen hohen Grad an Unsicherheit auf, zu gewinnen sind nur probabilistische Informationen über eventuelle Hindernisse. Dafür ist der Ansatz mit der Formel von Bayes sehr gut geeignet: Die aus dem Sensormodell hervorgehende Wahrscheinlichkeitsinformation der Sensormessung kann sofort in die Gitterrepräsentation eingehen.

Diese Methode ist relativ rechenzeitaufwändig und erfordert zudem, dass der Roboter zum Aufnehmen der Messungen stationär bleibt. Borenstein und Koren stellten daher als Alternative dazu das *Histogramm-Gitter* (*Histogram Grid*) vor (Borenstein und Koren, 1991). Dabei wird für jede Sensormessung nur genau eine (die „anscheinend getroffene" Zelle) aktualisiert, indem ein Belegtheitswert $b \in \mathbb{N}$ hochgezählt wird. Dabei ergibt sich eine „Pseudo-Wahrscheinlichkeitsverteilung" durch schnell aufeinander folgende Messungen, während der Agent sich kontinuierlich weiter bewegt. Diese Parallelisierung von Roboterbewegung, Umweltmodellierung und Planung, die die starre Abfolge „Stehenbleiben – Wahrnehmen – Planen" ersetzt, ist neben der primitiven Arithmetik der große Vorteil dieser Methode. Es konnte sogar nachgewiesen werden, dass diese augenscheinlich simplifizierte Herangehensweise exaktere Karten produziert als der wahrscheinlichkeitstheoretische Ansatz (Rascke und Borenstein, 1990).

Die Histogramm-Gitter haben in den 90er Jahren gegenüber den stochastischen Belegtheitsgittern zunehmend an Bedeutung gewonnen, vor allem für Sensorik mit einem relativ niedrigen Grad an Unsicherheit wie Laser-Scanner und Infrarotsensoren. Auch in dieser Arbeit wird eine Variante dieses Verfahrens verwendet (siehe dazu Abschnitt 4.4.1).

Ein entscheidender Vorteil der Idee der Belegtheitsgitter und dessen Varianten ist, dass sich die Daten verschiedener Sensoren sehr leicht in einer Repräsentation vereinen lassen. Ist die Position eines Punktes im Raum (bzw. eine diesbezügliche Wahrscheinlichkeitsverteilung) bekannt, können mehrere solche Informationseinheiten mittels einfacher Arithmetik zusammengefasst werden.

3.1.6 Spezielle Anforderungen an Weltmodelle für Laufmaschinen

Zunächst soll noch betrachtet werden, welche zusätzlichen Anforderung an ein Weltmodell gestellt werden müssen, das für eine Laufmaschine geeignet ist. Dazu ist es notwendig, sich kurz die Rahmenbedingungen vor Augen zu führen, die bei der Arbeit mit dieser Art von Robotern auftreten.

Ein fundamentaler Unterschied in der Steuerung einer Laufmaschine zu bspw. radgetriebenen Robotern ist, dass nicht nur eine Lenkachse und ein Motor angesteuert werden muss, sondern eine Vielzahl von einzelnen Beinen. Genügt bei einem radgetriebenen Gefährt prinzipiell die Route, auf der es einen Zielpunkt erreichen muss, sind für eine Laufmaschine Steuerungsentscheidungen für jeden einzelnen Schritt notwendig. Jeder neue Fußaufsetzpunkt setzt im Optimalfall eine separate Regelungsentscheidung voraus. Die jeweils unmittelbar bevorstehende Herausforderung einer Laufmaschine ist stets ihr nächster Schritt. Ausgehend von dieser Feststellung lassen sich folgende Anforderungen für Umweltmodelle für Laufmaschinen benennen:

Detailgenauigkeit im Nahfeld Um eine optimale Steuerung anzustreben, ist es nötig, dass Fußaufsetzpunkte des nächsten Laufzyklus' möglichst exakt bestimmt werden können. Dazu ist eine gute Kenntnis der in Frage kommenden Umgebung von Nöten. Die unmittelbare Umgebung muss also in einer Auflösung im Zentimeter-Bereich erfasst werden können. Insbesondere bedeutet das, dass eine sehr detaillierte Erfassung des Untergrunds erforderlich ist.

Gezielte Nachmessungsmöglichkeit Erwartet man realistischerweise, dass nicht der gesamte in Bewegungsrichtung der Maschine liegende Raum vollständig und exakt kartiert werden konnte, wenn die Entscheidung des nächsten Schritts ansteht, wäre es wünschenswert, die in Frage kommenden Bereiche gezielt sensorisch untersuchen zu können, wenn dies nötig erscheint. Das Weltmodell sollte dazu Mechanismen zur Verfügung stellen, solche kritischen Bereiche zu erkennen und gezielt nacherfassen zu können.

Exakte Hinderniskartierung Laufmaschinen weisen ein hohes Maß an Flexibilität in der Bewegung und damit auch in der Bewältigung von Hindernissen auf. Barrieren können überstiegen werden oder durch Ducken untergangen werden. Deshalb muss die Geometrie des Hindernisses möglichst genau erfasst und gespeichert werden, um eine geeignete Reaktion des Roboters wählen zu können.

Leichte Generierbarkeit unter schwacher Sensorik Einschränkungen bestehen bei Laufmaschinen bezüglich der Sensorik. Aufgrund der Bauweise ist es nicht möglich, beliebige Sensoren auf der Maschine unterzubringen, da die Sensoren zu groß und schwer sind, als dass sich die Maschine noch normal bewegen könnte. Ein 2D-Laser-Scanner nach derzeitigem Stand der Technik bspw. wird für viele Laufroboter sowohl zu groß als auch zu schwer sein – zumindest für solche mit einem Gewicht von unter 20 kg, mit denen sich in dieser Arbeit befasst werden soll. Zudem besitzen Laufmaschinen sehr viele Freiheitsgrade, so dass Messungen der eigenen Lage sehr störanfällig sind. Das Weltmodell sollte dem Rechnung tragen: Seine Erstellung sollte auch unter der Einschränkung schwacher und lückenhafter Sensorik leicht erstellbar sein.

Extraktion metrischer Information Da grundlegende Parameter der Steuerung von Laufmaschinen wie die Länge der Schritte metrischer Natur sind, müssen so geartete Informationen aus dem Weltmodell extrahierbar sein.

Eine detaillierte Beschreibung des hier verwendeten Roboters LAURON findet sich in Abschnitt 5.1. Da das zu entwerfende Weltmodell für beliebige Laufmaschinen geeignet sein soll, sind diese Details erst einmal nicht relevant. Für die folgenden Ausführungen ist es jedoch wichtig, die genannten physikalischen Beschränkungen geeignet zu berücksichtigen.

3.1.7 Wahl eines geeigneten Repräsentations-Ansatzes

Nachdem in den letzten Abschnitten mehrere Anforderungen an das zu entwerfende System gestellt wurden, kann man nun bereits eine erste Design-Entscheidung treffen. Offensichtlich ist der topologische Kartierungsansatz trotz einiger deutlicher Vorteile für das in dieser Arbeit gesteckte Ziel nicht geeignet. Daher wird sich im Folgenden nur noch mit geometrischen Kartierungsmethoden befasst werden. Im Einzelnen sprechen folgende Gründe für diese Entscheidung:

1. Die Generierung eines zufrieden stellenden topologischen Modells wird mit der verfügbaren Sensorik nicht möglich sein. Eine geeignete Klassifizierung beobach-

teter Strukturen setzt eine ausreichende Dichte an Sensordaten voraus. Der Roboter soll sich aber auch auf seine Weltmodellinformation verlassen können, wenn nur vereinzelte Sensormessungen verfügbar sind. Zum Beispiel kann ein Mittel- oder Hinterbein der Maschine aus den von der Sensorik des Vorderbeines gewonnenen Daten ausreichend Information für seinen nächsten Schritt ziehen, auch wenn diese nur einen sehr geringen Bereich der Welt umfasst. Eine verlässliche topologische Repräsentation ist in diesem Fall kaum zu gewinnen.

2. Wenn dem System a-priori-Informationen über die Umwelt vorliegen würden, wäre es möglich, diese Daten durch Abgleich mit den einkommenden Sensordaten zu verwerten und so die oben genannte Schwäche zu umgehen. Das System soll aber ohne a-priori-Wissen auskommen.

3. Die Ansteuerung des Roboters erfolgt über exakt messbare Parameter wie Schrittlänge oder Gelenkwinkel. Zu ihrer Ermittlung ist es unverzichtbar, regelmäßig metrische Information aus dem Weltmodell zu gewinnen. Reine topologische Ansätze bieten diese nicht, bestenfalls kämen diktiometrische Karten in Betracht.

4. Eine detaillierte Kartierung des Bodens wird mit topologischer Repräsentation nicht sinnvoll möglich sein, wenn dieser nicht überwiegend plan ist. Ansonsten müsste jede Bodenunebenheit durch einen eigenen Knoten repräsentiert und in Bezug zu anderen Unebenheiten gesetzt werden, was im Endeffekt zu einer quasi-geometrischen Darstellung (nur mit ungleich höherem Aufwand) führen würde. Gerade das Finden von geeigneten Fußaufsetzpunkten ist aber eine Kernaufgabe für Laufmaschinen, so dass auf eine hoch aufgelöste Bodenrepräsentation nicht verzichtet werden kann.

Tabelle 3.2 versucht anhand der in den Abschnitten 3.1.4 und 3.1.6 genannten Anforderungen die Vor- und Nachteile der vorgestellten Ansätze in Form einer Checkliste darzustellen. Alles in allem scheint es unter Berücksichtigung dieser Punkte – vor allem bei den spezifischen Problemen von Laufmaschinen – so gut wie unmöglich, mit dem topologischen Ansatz zu zufrieden stellenden Ergebnissen zu kommen. Die Problemfelder, die geometrische Karten aufweisen (wie bspw. der massive Speicherverbrauch), sind nicht derart weitreichend und können mit geeigneten Methoden entschärft werden. Auf Möglichkeiten hierfür wird im weiteren Verlauf der Arbeit eingegangen werden. Vor allem die Vorteile bei der Verwendung schwacher und lückenhafter Sensorik sprechen bei der Wahl einer Repräsentation eindeutig für das Belegtheitsgitter.

Um die definitiv vorhandenen Vorteile topologischer Kartierungen, insbesondere die eleganten Methoden zur Pfadplanung nicht vollends zu verwerfen, wird für die Umweltmodellierung perspektivisch die in Abschnitt 3.1.3 erwähnte Möglichkeit der nachträglichen Generierung topologischer Karten aus den geometrischen Daten verfolgt. Dies ist allerdings nicht Thema dieser Arbeit und sollte Gegenstand weiterer Forschungen sein.

	Geometrische Karten		Topologische Karten	
	allgemein	Belegtheitsgitter	allgemein	diktiometrisch
Unstrukturiertes Gelände:				
Unabhängigkeit von Kanten	−	○	−	−
Unabhängigkeit von a-priori-Karten	○	○	−	−
Schwache Odometrie	−	−	+	○
Behutsame Hindernisklassifikation	○	+	−	−
Laufmaschinen:				
Detailgenauigkeit im Nahfeld	+	+	−−	−−
Nachmessmöglichkeit	+	++	−	○
Exakte Hindernisklass.	+	+	−−	−−
Schwache Sensorik	○	++	−	−
Metrische Information	++	+	−−	+

++: gut geeignet, +: geeignet, ○: mäßig geeignet, −: wenig geeignet, −−: ungeeignet

Tabelle 3.2: Zusammenstellung der Eignung der vorgestellten Kartierungsmodelle bzgl. der Anforderungen für Laufmaschinen in unstrukturiertem Gelände

3.1.8 Zwei, zweieinhalb oder drei Dimensionen?

In Abschnitt 3.1.7 fiel die Wahl auf eine geometrische Repräsentation der Umwelt. Nun muss abschließend noch geklärt werden, ob zwei Dimensionen für die Speicherung ausreichen oder ob eine dreidimensionale Darstellung, also ein Volumenmodell, notwendig ist. Letzteres würde insbesondere eine immense Zunahme des Speicherbedarfs und damit der Verarbeitungskomplexität bedeuten, so dass eine dreidimensionale Darstellung nur gewählt werden sollte, wenn es unbedingt angezeigt ist.

In der Robotik finden sich zumeist zweidimensionale Kartierungsansätze. In der Regel reicht dies auch vollkommen aus: Für einen autonomen Putzroboter genügt ein Grundriss seines Einsatzgebiets, um die dort eingezeichneten Hindernisse umgehen zu können. Informationen über die dritte Dimension sind hier ohne Belang: Es ist für seine Zwecke nicht relevant, wie hoch ein im Weg stehender Schrank ist oder ob der Fußboden leicht uneben ist. Entscheidend ist, ob er die Position erreichen kann oder nicht. Die Information „Hindernis oder nicht“ ist ausreichend.

Für Laufmaschinen ist dies aber nicht der Fall. Hindernisse sollen nicht nur vermieden, sondern möglichst überwunden werden. In Abschnitt 3.1.6 wurden zwei Anforderungen genannt, die ein zweidimensionales Umweltmodell für die Arbeit mit Laufmaschinen ausschließen:

1. Die Forderung nach exakter Hinderniskartierung zieht die Notwendigkeit einer möglichst exakten Wiedergabe der Geometrie des Hindernisses nach sich. Höheninformationen müssen zwingend gespeichert werden, damit die Maschine entscheiden kann, ob sie eine Barriere übersteigt oder nicht.

2. Detailgenauigkeit im Nahfeld bedeutet insbesondere, dass der Untergrund so erfasst ist, dass der nächste Fußaufsetzpunkt möglichst exakt ermittelt werden kann. Auch dazu ist die Speicherung von Höheninformationen notwendig.

Eine weit verbreitete Methode ist es, in einem zweidimensionalen Gitter für jede Zelle eine Höheninformation und ggf. weitere Daten wie Belegtheitswahrscheinlichkeit zu speichern (vgl. bspw. Murphy et al. (2002), Abschnitt 3.2.6). Solche Ansätze werden oft mit der etwas unkorrekten Bezeichnung $2\frac{1}{2}$*D-Modelle* versehen. Ihre Verwendung würde allerdings einerseits voraussetzen, dass die Höheninformation jeweils mit großer Sicherheit bestimmbar ist, was auf Grund der zu verwendenden Sensorik nicht erwartet werden kann. Andererseits können auf diese Weise Hindernisse nicht einschließlich ihrer geometrischen Information als Überhang oder einragendes Objekt (wie bspw. ein Ast) beschrieben werden. Insofern ist die Verwendung eines „echten“ 3D-Modells in der Ausprägung eines Belegtheitsgitters der für die Steuerung von autonomen Laufmaschinen geeignetste Ansatz und soll in dieser Arbeit verwendet werden.

3.2 Exemplarische Zusammenstellung verschiedener Weltmodellentwürfe

Abschließend sollen an dieser Stelle einige konkrete Umweltmodellierungen für autonome mobile Roboter vorgestellt werden. Das Hauptaugenmerk liegt dabei nicht auf Vollständigkeit, vielmehr wurde versucht, eine Auswahl von Beispielen zusammenzustellen, die Gemeinsamkeiten zur Stoßrichtung dieser Arbeit aufweisen. Aufgeführt werden hierbei nur Arbeiten, die eine geometrische Modellierung der Umwelt verfolgen. Einige der aufgezeigten Ansätze sind auch in den vorliegenden Weltmodellentwurf eingeflossen, andere wurden verworfen. Dies soll jeweils am konkreten Beispiel diskutiert werden. In erster Linie soll dieser Abschnitt aber einen Überblick über unterschiedliche Vorgehensweisen unter verschiedenen Randbedingungen bieten.

3.2.1 Kruse, Gutsche und Wahl (1995)

Einen interessanten Ansatz zur gezielten und effizienten Vermessung einer unbekannten Umwelt durch autonome Roboter mittels einer dreidimensionalen Umweltbeschreibung stellen Kruse, Gutsche und Wahl in Kruse et al. (1995) vor. Ihr Schwerpunkt liegt dabei darin, mittels einer systematischen Durchführung von Messungen zur Weltmodellerstellung die Abhängigkeit von a-priori-Wissen zu reduzieren. Als Repräsentation für die Umgebung wählten sie ein dreidimensionales Raster, für das empfohlen wird, den Ansatz des Belegtheitsgitters von Elfes (Elfes, 1989) ins Dreidimensionale zu übertragen. In dieses Raster werden Hindernisse eingetragen, die die Sensorik durch Abstandswerte liefert. Ziel des Systems ist es, einen Zustand zu erreichen, in dem eine maximal große

Anzahl neuer Informationen akquiriert werden kann. Um dies zu gewährleisten, führen die Autoren eine *Bewertungsfunktion* für die Systemzustände ein, die maximiert wird. Eine Teilfunktion, die in die Bewertung einfließt, beschreibt die Güte der Sensormessung: Messbereiche, die mit hoher Genauigkeit erfasst werden können, sollen stärker einfließen (so werden bspw. Messwerte im Schwerpunkt der Sichtpyramide eines Sensors stärker gewichtet). Das System beschreibt zur Erreichung seines Ziels dabei einen Messzyklus mit den Phasen *Planen*, *Messen* und *Aktualisieren*.

Gerade die Idee einer Gütefunktion für Messungen eignet sich besonders für das vorliegende Problem: Die Sensorik ist recht unterschiedlich und nicht allzu umfassend, zudem ist ihre Qualität vom Zustand der Maschine abhängig – nicht jede Sensormessung wird also gleich zu gewichten sein. Um dem gerecht zu werden, wird dieser Ansatz einer Gütefunktion in Abschnitt 4.6 erneut aufgegriffen werden.

3.2.2 Lågstad und Auran (1996)

Lågstad und Auran stellen hier ein Verfahren vor, um in Echtzeit mittels Kombination eines Sonars und eines Kamerasystems unter Wasser zu einem 3D-Modell der Umwelt zu kommen. Sowohl das spezielle Aufgabengebiet als auch die verwendete Sensorik unterscheiden sich stark von den Anforderungen dieser Arbeit. Interessant ist aber die Herangehensweise an das Problem, dass die Datenmengen der Kamerabilder zu groß sind, um sie in Echtzeit geeignet verarbeiten und in Bezug zur Sonarinformation setzen zu können: Dazu wird das Originalbild mehrfach hintereinander verkleinert, indem ein Tiefpassfilter über die Bildinformation gelegt wird. Es entsteht so eine Kaskade von Karten mit gröberer Auflösung (*scale space pyramid*). Zu diesem Zweck werden hier Filter vorgestellt, die trotz geringer Rechenzeit Merkmale wie Kanten gut erhalten.

Ein ähnliches Problem entsteht in dieser Arbeit, wenn, wie in Abschnitt 3.1.1 gefordert, Karten mit verschiedenen Abstraktionsebenen gehalten werden sollen. Dazu bietet dieses Filterverfahren einen guten Ansatz.

3.2.3 Weckesser und Dillmann (1997)

In dieser Veröffentlichung wird ein Umweltmodell für den experimentellen, radgetriebenen Roboter Priamos vorgestellt. Für die Kartierung wird hierbei eine dreidimensionale geometrische Repräsentation gewählt, in die als geometrische Primitive Liniensegmente eingetragen werden. Sie werden durch Mittelpunkt, Richtungsvektor und Länge beschrieben. Die Segmente werden mit Hilfe des *Iterative-End-Point-Fit*-Algorithmus aus den Sensordaten gewonnen. Mit dieser Vorgehensweise können sehr exakte Karten unbekannter Umgebungen erstellt werden, die sich gut für weitergehende Aufgaben wie Positionskontrolle und Ableitung topologischer Karten aus dem Modell eignet.

Für unsere Zwecke ist dieses Verfahren allerdings aus zwei Gründen nicht verwendbar: Einerseits widerspricht es der in Abschnitt 3.1.4 geforderten Unabhängigkeit von Kantendetektion. Andererseits – und vor allem deshalb wird dieses Beispiel hier angeführt – zeigt sich die begrenzte Eignung dieses Verfahrens beim Blick auf die eingesetzte Sensorik: Verwendet werden für die Kantensegmentextraktion u. a. ein triokulares Kamerasystem und ein 2D-Laserscanner. Zumindest letzterer ist für den Einsatz auf einer

Laufmaschine wie LAURON beim derzeitigen Stand der Technik noch viel zu groß und schwer.

3.2.4 Lallement, Siadat, Dufaut und Husson (1998)

Das in Lallement, Siadat, Dufaut und Husson (1998) vorgestellte Umweltmodell verwendet weitestgehend den gleichen Ansatz wie das zuletzt vorgestellte. Hier werden die Sensordaten aus einem Monokular-Kamerasystem und einem 2D-Laserscanner ebenfalls mit dem Iterative-End-Point-Fit-Algorithmus in Liniensegmente transformiert, allerdings nur in einer zweidimensionales Repräsentation. In der Veröffentlichung werden explizit als Nachteile des gewählten Verfahrens die notwendige Datenvor- und -nachbearbeitung sowie die Notwendigkeit, dass die gesamte Datenreihe zum Verarbeitungszeitpunkt vollständig vorliegen muss, genannt. Die Einschränkungen bzgl. der Sensorik gelten wie oben erwähnt.

3.2.5 Stuck, Manz, Green und Elgazzar (1994)

Die in Stuck et al. (1994) beschriebene Methode zur Umweltkartierung verwendet Histogrammgitter (Borenstein und Koren, 1991) als Repräsentation. Die Belegtheitswerte werden zwischen 0 und 14 definiert, wobei alle Zellen mit dem Wert 7 initialisiert werden. In einem fünfschrittigen Zyklus wird nun zunächst die Position und Orientierung des Roboters bestimmt. Danach wird in das Gitter eine Linie zwischen der Roboterposition und dem gemessenen Hindernispunkt projiziert sowie ein Toleranzbereich berechnet, der ungefähr die halbe Ausdehnung einer Gitterzelle umfasst; diese Linie beschreibt definitiv freie Bereiche. Im vierten Schritt werden die Belegtheitswerte aller Zellen im Toleranzbereich des Endpunktes erhöht und schließlich die der verbliebenen Zellen in der Projektion erniedrigt, da dieser Bereich augenscheinlich frei ist. Belegtheitsinformation wird dabei stärker gewichtet, erhöht wird jeweils um den Wert 3, erniedrigt um 1. Mit dieser Herangehensweise konnten gute Resultate in der Echtzeit-Routenplanung für einen radgetriebenen Roboter erzielt werden.

Einiges aus diesem Ansatz wird in der vorliegenden Arbeit berücksichtigt: Die Errechnung einer Linie zwischen Sensor und Hindernis und die Markierung dieser Zellen als frei ist ein Ansatz, der für dieses Projekt aufgegriffen wird (siehe Abschnitt 5.2.2). Auch wird eine Variante eines Histogramm-Gitters verwendet. Die Repräsentation durch 15 verschiedene Belegtheitswerte wird allerdings den speziellen Anforderungen für Laufmaschinen in unstrukturiertem Gelände nicht gerecht, so dass ein feineres Repräsentationsmodell gewählt wird; dies wird in Abschnitt 4.4 ausführlich erläutert.

3.2.6 Murphy, Abrams, Balakirsky, Chang, Hong, Lacaze und Legowik (2002)

In Murphy et al. (2002) wird eine Umweltmodellierungsarchitektur beschrieben, um ein so genanntes HMMWV (High Mobile Multipurpose Wheeled Vehicle) zu steuern, eine Art autonomer Geländewagen. Dazu wird ein vierschichtiger Regler verwendet: die unterste Schicht (Servo) erledigt die Verarbeitung der Sensordaten, die zweite Schicht (Prim) Hinderniserkennung und Nahfeldkartierung, die dritte Schicht (Autonomous

Mobility, AM) weitergehende Kartierung und Ansteuerung von Motoren und Lenkung, die letzte Schicht (Vehicle) dient schließlich der Großkartierung und Routenplanung, hier werden die Daten auch mit einer a-priori-Karte abgeglichen. Für die Schichten 2 bis 4 werden jeweils Karten mit verschiedenen Auflösungen und Ausdehnungen generiert, mit zunehmendem Level nimmt der Abstraktionsgrad sowohl von Kartierung als auch von Planung und Steuerung zu. Die Karten werden bei der Bewegung des Fahrzeugs in Echtzeit mitgescrollt. Repräsentiert wird die Umwelt als ein zweidimensionales Gitter, es handelt sich demnach um ein $2\frac{1}{2}$D-Modell. Der Höhenwert ergibt sich als gewichteter Durchschnitt von alten und aktuellen Messungen, wobei der Anteil alter Beiträge exponentiell mit der Zeit abnimmt. Weiterhin wird jede Zelle nach einem relativ einfachen System entweder als Boden, Hindernis, Überhang oder Unbekannt klassifiziert, dazu wird sowohl die Höhe des detektierten Objekts als auch die Neigung an dieser Stelle betrachtet. Das ist möglich, da als Sensor ein 3D-Bereichslaser verwendet wird und eine entsprechend große Anzahl an zusammenhängenden Eingangsdaten zur Verfügung steht.

Die erstklassige Sensorik, die auf einem Geländewagen leicht unterzubringen ist, ist einer der wichtigen Unterschiede zu unserer Aufgabenstellung: Es ist eine hohe Dichte und Genauigkeit an Umweltdaten zu erwarten, die ihre Verarbeitung erleichtern. Zudem ist die Größe der Roboter sehr verschieden, Murphy et al. arbeiten auf der untersten Ebene mit einer Auflösung von 40 × 40 cm, Sensorrauschen fällt hierbei nicht so stark ins Gewicht. Und zuletzt ist für ein radgetriebenes System eine zweidimensionale Kartierung aufgrund seiner eingeschränkten Bewegungsmöglichkeiten vollkommen ausreichend: Entweder, es kann eine Position anfahren, oder es kann nicht. Der Ansatz der mehrschichtigen Kartierung wird aber auch in dieser Arbeit noch aufgegriffen werden (Abschnitt 4.1).

3.2.7 Bai und Low (2002)

Diese Arbeit ist eine der sehr wenigen Veröffentlichungen, die sich überhaupt mit Weltmodellierung explizit für Laufmaschinen beschäftigen. Vorgestellt wird ein Verfahren zur Pfadplanung, das insbesondere auch berücksichtigt, dass während der Fortbewegung des Roboters stets gute Aufsetzpunkte für die Füße gefunden werden müssen. Die Welt wird hier durch ein geometrisches $2\frac{1}{2}$D-Modell repräsentiert. Ausgehend von der Höheninformation jeder Zelle wird diese als frei bzw. positives oder negatives Hindernis klassifiziert; negative Hindernisse sind solche, die niedriger als die maximale Schritthöhe des Roboters sind und somit überwunden werden können. Eingeführt wird nun eine Bewertung A_c, die aussagt, ob eine Zelle als Fußaufsetzpunkt geeignet ist; dies ist dann der Fall, wenn sich in einer definierten Umgebung um diese nicht ausschließlich positive Hindernisse befinden. Aus diesen Bewertungen werden nun weitere Klassifikationen für größere Umgebungen errechnet, die zur Pfadplanung herangezogen werden. Der Dreidimensionalität der Landschaft wird dadurch Rechnung getragen, dass um positive Hindernisse herum eine Einflussumgebung von einer halben Maschinenlänge definiert wird, die als unbegehbar klassifiziert wird. Dies soll ein Anstoßen des Roboters verhindern.

Das Hauptaugenmerk dieser Veröffentlichung liegt auf der Pfadplanung, einer eher fortgeschrittenen Ausprägung der Wissensextraktion aus einem Weltmodell. Mit dessen

Erstellung wird sich nicht befasst: Alle Ergebnisse entstammen einer Simulation, die Karte wird a-priori vorgegeben und entbehrt jeglicher Störeinflüsse. In der vorliegenden Arbeit sind Erstellung und Repräsentation des Weltmodells allerdings die Kernaufgaben, so dass der Ansatz von Bai und Low (2002) eher für Folgearbeiten interessant sein wird. Zudem erscheint die Umsetzung ins Dreidimensionale halbherzig, da die Einteilung der Hindernisse in positive und negative ausschließlich auf der Grundlage ihrer absoluten Höhe erfolgt und offensichtlich nicht berücksichtigt, dass die Maschine auf negativen Hindernissen stehen kann und somit evtl. in der Lage wäre, auch positive Hindernisse zu überwinden.

Interessant ist allerdings die Bewertung von guten Fußaufsetzpunkten, die auch freie Stellen in der Umgebung berücksichtigt. Ein ähnlicher Denkansatz findet sich in dieser Arbeit bei der Erstellung lokaler Karten in der unmittelbaren Umgebung der Füße (vgl. Abschnitt 5.3.2).

Kapitel 4

Konzeption des Weltmodells

An dieser Stelle soll ausgehend von den Überlegungen des letzten Kapitels der grundlegende Aufbau des in dieser Arbeit behandelten Weltmodell-Entwurfs diskutiert und erläutert werden. Dabei sollen insbesondere softwaretechnische Design-Entscheidungen begründet werden.

4.1 Mehrstufige Kartierung

In Abschnitt 3.1.8 wurde die Notwendigkeit einer dreidimensionalen Repräsentation der Umwelt und somit der Verwendung eines Volumenmodells dargestellt. Solche Ansätze weisen allerdings eine hohe Komplexität auf und erfordern ein hohes Maß an Speicherplatz. Eine Speicherung größerer Umgebungen würde auf diese Weise jeden Rahmen sprengen. Sie ist allerdings auch nicht notwendig: Für Aufgaben, die weit reichende Karten benötigen (z. B. Routenplanung), sind auch für Laufmaschinen zweidimensionale Modelle mit geeigneten Wegbarkeitsinformationen vollkommen ausreichend. 3D-Daten benötigt man in erster Linie für das Nahfeld, sie sind entscheidend, um die richtigen Schritte setzen zu können. Es bietet sich demnach an, die unmittelbare Umgebung hoch aufgelöst in 3D und alles weitere in geringerer Auflösung in 2D zu speichern. Damit wird auch der in Abschnitt 3.1.1 genannten Forderung von Hoppen et al. (1990) entsprochen, ein Modell mit verschiedenen Abstraktionsebenen zu generieren. Die einzelnen Karten besitzen den Detaillierungsgrad, der für die jeweilige Verarbeitungsaufgabe notwendig ist. Die dreidimensionale Repräsentation soll im Folgenden als *Urmodell* oder *Urkarte* bezeichnet werden.

Diese Arbeit greift diesbezüglich einen Ansatz von Murphy et al. (2002) auf: Die unmittelbare Umgebung wird (in 3D) in hoher Auflösung gehalten und diese Karte mit der Bewegung des Roboters mitgescrollt, so dass sich die Maschine immer ungefähr in der Mitte der Karte befindet. Gleichzeitig werden daraus Modelle mit höherem Abstraktionsgrad und niedrigerer Auflösung generiert (Abbildung 4.1). Diese Erzeugung der Karten kann ggf. in einem Hintergrundprozess mit niedriger Priorität ablaufen, da unmittelbare Aktualität hier nicht erforderlich ist. Damit behindert die Generierung von Großkarten nicht die Steuerung der Roboters.

Der Speicherverbrauch solcher Übersichts-Karten ist unproblematisch: Wegen ihrer Beschränkung auf zwei Dimensionen und die gröbere Granulierung der Zellen fällt sie

gegen das 3D-Modell nicht ins Gewicht. Es können also mehrere Karten in verschiedenen Auflösungen generiert und gehalten werden.

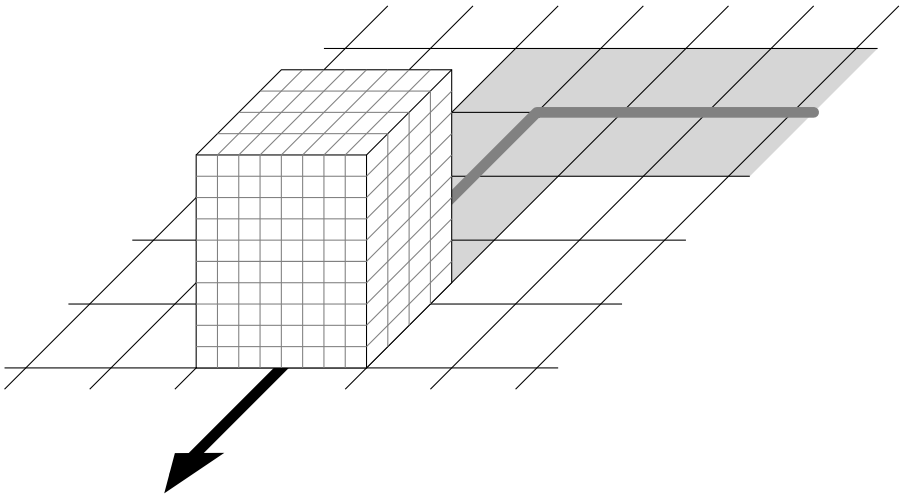


Abbildung 4.1: Schematische Darstellung: Scrollen der 3D-Umgebung und Erzeugung von 2D-Karten mit gröberer Auflösung. Die grauen Bereiche wurden aus der sich mit dem Roboter „bewegenden“ Urkarte generiert.

Das Scrollen der unmittelbaren Umgebungskarte bedeutet natürlich, dass das erworbene Wissen in den herausgeschobenen Bereichen wieder verloren geht. Dies ist aber vertretbar: Der Wert detailgenauer Informationen nimmt mit der Zeit durch die auftretenden Ungenauigkeiten in der Odometrie der Maschine stetig ab, so dass ausgiebige Neuvermessungen ohnehin notwendig sein werden, wenn der Roboter nach längerer Zeit an eine bereits bekannte Stelle zurückkehrt. Der Speicheraufwand des Vorhaltens derart alter Daten steht in keinem Verhältnis zum erwartbaren Nutzen. Wenn man die detaillierten Umweltinformationen dennoch nicht verlieren möchte, wäre es eine Option, sie auf einen externen Datenträger wie die Festplatte auszulagern.

Eine Besonderheit dieses Modell ist, dass zusätzlich noch weitere Kleinkarten gehalten werden, die die unmittelbare Umgebung der nächsten Aufsetzpunkte aller Füße des Roboters beschreiben. Diese Karten werden vor jedem Schritt aus dem 3D-Datenbestand generiert. Für sie bietet sich ebenfalls eine 2D- bzw. $2\frac{1}{2}$D-Repräsentation an, da diese Karten einem klar umrissenen Ziel (dem Finden eines geeigneten Fußaufsetzpunktes) dienen und keine neuen Sensorinformationen erhalten, sondern sich zu einem singulären Erzeugungszeitpunkt auf den Datenbestand der Urkarte stützen.

4.2 Schichtaufbau des Umweltmodells

Für die Repräsentation der Umwelt wird in diesem Abschnitt ein Schichtenmodell vorgestellt, das für alle in diesem Weltmodell verwendeten Karten, einschließlich der Fußumgebungs-Kleinkarten, angewendet wird. Der Schichtenansatz folgt dem objekt-

orientierten Paradigma der Kapselung, wonach alle Details eines Objektes versteckt werden sollen, die nur seine inneren Vorgänge betreffen[1].

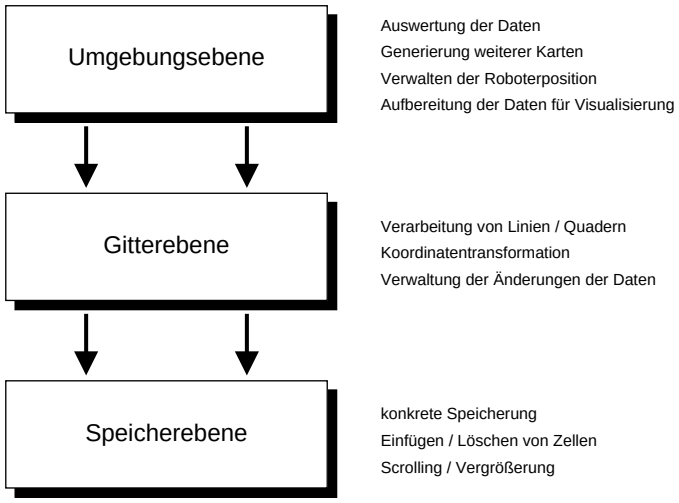


Abbildung 4.2: Schichtweiser Aufbau der Karten des Umweltmodells mit den entsprechenden Aufgaben. Feste Schnittstellen zwischen den Ebenen sorgen für leichte Austauschbarkeit.

Danach besteht jede der verwendeten Karten aus drei Ebenen, die durch fest definierte Schnittstellen miteinander verbunden sind. Den Aufbau und die jeweiligen Aufgaben zeigt Abbildung 4.2. Im einzelnen sind dies, beginnend mit der untersten Schicht:

Speicherebene In dieser Schicht erfolgt die konkrete Speicherung der Umweltdaten in einer Datenstruktur. Hier sind die Operationen definiert, die direkt den Datenbestand des Umweltmodells betreffen: Einfügen, Löschen und Auslesen der Zellen, Vergrößern des Definitionsraums und Scrollen der Karte.

Gitterebene Diese Schicht liefert eine Sicht auf das Belegtheitsgitter. Reale Koordinaten werden in das interne Koordinatensystem der Speicherebene transformiert und Methoden zur Verarbeitung größerer geometrischer Primitive wie Linien und Quader werden zur Verfügung gestellt. Zudem erfolgt hier eine Verwaltung der letzten Änderungen am Datenbestand; dies wird bspw. für Visualisierungszwecke verwendet (siehe Abschnitt 5.6).

Umgebungsebene Dies ist die höchste Schicht im Umweltmodell. Neben der Verwaltung der Position des Roboters in der Karte werden hier die Funktionen zur Auswertung des Weltmodells bereitgestellt, bspw. Generierung von Karten anderer Abstraktionsebenen, Finden von Fußaufsetzpunkten etc. Zudem ist die Umgebungsebene für die Aufbereitung der Visualierungsdaten und deren Transport zur grafischen Oberfläche verantwortlich.

Die Umgebungsebene ist diejenige, die den Unterschied zwischen den im Modell verfügbaren Karten ausmacht, ihre Ausprägung ist auf die jeweilige Charakteristik ab-

[1] „Encapsulation is the process of hiding all of the details of an object that do not contribute to its essential characteristics.“ (Booch, 1991)

gestimmt. Gitter- und Speicherebene dagegen sind für alle Abstraktionsebenen gleich, ihre Funktionalität wird von allen Karten benötigt.

Die Vorteile dieser Architektur im praktischen Einsatz liegen auf der Hand. Durch die Kapselung und die Definition fester Schnittstellen kann jede Schicht leicht ausgetauscht werden. Es ist also bspw. problemlos möglich, die Realisierung der Speicherebene durch eine effizientere Variante zu ersetzen, ohne am Rest der Implementierung Änderungen vornehmen zu müssen. Zudem garantiert dieser Ansatz extrem hohe Wiederverwertbarkeit des Programmcodes: Die Programm-Klassen für Gitter- und Speicherebene werden in der vorliegenden Umsetzung des Weltmodells nicht nur für alle Versionen von Karten, sondern auch gleichermaßen für die Realisierung der Visualisierungssoftware verwendet. Die Schnittstellen der einzelnen Programmklassen finden sich in Anhang B.1.

Allen Karten übergeordnet ist die *Steuerungsebene*. Sie gibt die aufbereiteten Sensordaten an die 3D-Urkarte (und nur an diese) weiter, empfängt aber Auswertungen der Umweltdaten von allen aus der Urkarte generierten Karten und verarbeitet diese entsprechend (siehe Abbildung 4.3).

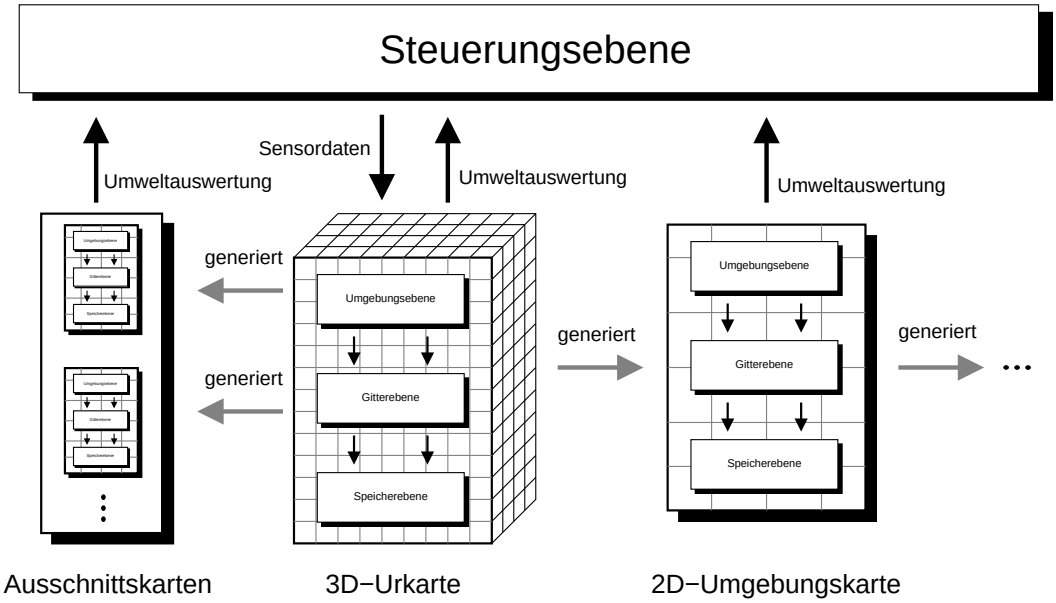


Abbildung 4.3: Interaktion von Steuerungsebene und Karten verschiedener Abstraktionsstufen: Die Steuerungsebene sendet Daten nur zur Urkarte, empfängt aber Auswertungen von allen Kartenstufen. Die Erzeugung von Unterkarten erfolgt in den Karten selbst.

4.3 Unabhängigkeit von der Maschine

Eine wesentliche Anforderung an den Entwurf dieses Weltmodells ist eine weitest gehende Maschinenunabhängigkeit. So soll die Realisierung der Umweltmodellierung nicht nur für die Steuerung der für diese Arbeit verwendeten sechsbeinigen Laufmaschine verwendet werden können, sondern auch für andere Laufmaschinen, die mit der gleichen

Steuerungsarchitektur betrieben werden. Es ist als also notwendig, dass das Software-Konzept so weit wie möglich von den physikalischen Eigenschaften des jeweiligen Roboters abstrahiert. Dies ist natürlich nur bis zu einem gewissen Punkt möglich: Bei einem sechsbeinigen Roboter müssen auch konkret sechs Beine angesteuert und deren Umwelt modelliert werden, bei einem Vierbeiner hingegen nur vier. Ebenso verwenden alle Maschinen unterschiedliche Sensorik und bieten so einen differenzierten Blick auf die Umgebung, der entsprechend gespeichert werden muss. Dem muss die Modellarchitektur Rechnung tragen, indem sie klar zwischen allgemeinen und maschinenspezifischen Konzepten trennt. Dazu werden für den Software-Entwurf des Weltmodells zwei Forderungen formuliert:

- Die Modellierung sämtlicher Spezifika der Roboter bzgl. Topologie und Sensorik beschränkt sich auf die Architektur der Gitterzelle im Belegtheitsgitter.
- Methoden, die auf maschinenspezifische Elemente der Gitterzelle zugreifen, müssen strikt vom restlichen Code abgetrennt sein.

In der vorliegenden, in C++ programmierten Implementierung, werden diese Forderungen dadurch umgesetzt, dass die im vorigen Abschnitt genannten Abstraktionsebenen (Speicher-, Gitter- und Umweltebene) jeweils durch Container-Klassen realisiert werden; die Steuerungsebene ist stets maschinenspezifisch. Die Klasse der Speicherebene operiert dabei auf Instanzen einer für jede Roboterarchitektur separat zu definierenden Gitterzellen-Klasse, deren übergeordnete, maschinenunabhängige Funktionalität und einheitliche öffentliche Schnittstelle von einer Basisklasse geerbt wird. Zusätzlich finden alle konkreten Details der Robotertopologie bzgl. der Umweltwahrnehmung in der Gitterzelle als kleinster Informationseinheit des Modells ihren Niederschlag. Operationen auf den Gitterzellendaten, wie bspw. die Ausgabe des gesamten Zellinhalts, werden für jede Roboterarchitektur separat implementiert und sind Bestandteil der Gitterzellenklasse selbst. Eine Teilmenge der Schnittstelle aller Gitterzellenklassen ist durch das Erben von einer gemeinsamen Basisklasse einheitlich.

Methoden, die direkt auf die Gitterzelle zugreifen (bspw. das Einfügen neuer Sensordaten auf der Steuerungsebene oder die Aufbereitung der Umweltinformation für die Visualisierung auf der Umweltebene), werden in anwendungsspezifischen Tochterklassen der Klassen für die jeweilige Ebene definiert. Eine einheitliche Schnittstelle ist dabei von der Basisklasse vorgegeben, die entsprechenden Funktionen müssen nur überladen werden.

Durch diese Architektur ist die Portierung des Weltmodells auf eine andere Maschinenarchitektur denkbar einfach. Es müssen dafür nur die Repräsentation der Gitterzelle angepasst und die wenigen direkt auf diese zugreifenden Funktionen entsprechend neu definiert werden. Die generelle Funktionalität der Weltmodellsoftware bleibt davon unberührt.

4.4 Repräsentation der Umgebung

In diesem Abschnitt soll nun das Teil-Modell beschrieben werden, das die Umgebung der Maschine repräsentiert. Zunächst wird diskutiert, welche Information sinnvollerweise gehalten werden soll, die gewählte Repräsentation erläutert und eine Methode

zum geeigneten Einfügen von Sensor-Informationen in das bestehende Modell vorgestellt. In Anschluss daran wird aufgezeigt, auf welche Weise Wissen aus dem Modell gewonnen werden kann und wie diese Erkenntnisse in die Steuerung des Roboters einfließen können. Abschließend wird anhand von Experimenten an der realen Maschine die Tauglichkeit der vorgestellten Verfahren aufgezeigt werden.

Wie in Kapitel 4 beschrieben, wird die Nahfeldumgebung durch ein 3D-Modell beschrieben. Als Repräsentation wurde eine Variante eines Belegtheitsgitters gewählt (für eine Übersicht siehe Abschnitt 3.1.5).

4.4.1 Das erweiterte Inferenz-Gitter

An dieser Stelle soll eine Variante eines Belegtheitsgitters vorgestellt werden, dass als *Erweitertes Inferenz-Gitter* bezeichnet werden soll. *Inferenz-Gitter* hat Alberto Elfes solche Belegtheitsgitter genannt, die mehr Information enthalten als nur die Belegtheitswahrscheinlichkeit (Elfes, 1989). Das hier verwendete Gitter verwendet als Inhalt der Zellen eine genau auf den jeweiligen Roboter und seine Sensorik zugeschnittene Datenstruktur, die neben der Wahrscheinlichkeit eines Hindernisses auch Information über die Entstehung der Messungen speichert. Die in dieser Arbeit benutzte Zellstruktur wird in Abschnitt 4.5 behandelt.

Die Erweiterung besteht nun im Vorgehen bei der Zellinhalt-Modifikation. Das erweiterte Inferenz-Gitter verwendet prinzipiell den Ansatz des Histogramm-Gitters: Für jede Sensormessung wird nur eine einzige Zelle modifiziert (Borenstein und Koren, 1991). Hier wird jedoch nicht einfach ein Wert um einen festen Betrag erhöht oder vermindert, sondern die neue Messung geht mit einem Faktor ein, der sich sowohl nach einer Schätzung der Güte der aktuellen Messung als auch nach der Historie der Zusammensetzung des bestehenden Zellinhalts richtet. Damit soll insbesondere der Tatsache Rechnung getragen werden, dass die eine Zelle betreffenden Informationen zu sehr unterschiedlichen Zeitpunkten von unterschiedlichen Sensoren und unter unterschiedlichen Rahmenbedingungen akquiriert wurden. Das in dieser Arbeit entwickelte Modifikationsverfahren beschreibt Abschnitt 4.6.

4.4.2 Verwendung von Octrees und Quadtrees zur Unterteilung des Raumes

Eines der Hauptprobleme bei geometrischen Ansätzen wie dem Belegtheitsgitter ist der ausufernde Speicherbedarf. Dieser Umstand wird noch dadurch erschwert, dass die Implementierung des Weltmodells auf einem auf dem Roboter befindlichen Kleinrechner laufen soll, dessen Ressourcen begrenzter sind als die einer großen Workstation. Von einer Speicherung der Weltmodell-Daten in einem herkömmlichen Array ist daher abzusehen.

Eine verbreitete Methode zur Platz sparenden Speicherung von Volumendaten ist die Verwendung von *Octrees* (Achterbäumen). Diese Datenstruktur stellt eine rekursive Unterteilung eines (kubischen) Raumes in jeweils acht Oktanten dar, die durch einen Baum mit acht Söhnen für jeden Knoten repräsentiert wird (Jackins und Tanimoto, 1980). Die Oktanten werden dabei sukzessive solange unterteilt, bis ihr Inhalt

homogen ist. Somit eignen sich Octrees insbesondere zur Speicherung spärlich verteilter 3D-Daten, da in diesem Fall homogene leere Regionen vorherrschen, wenig Unterteilungen nötig sind und wenig Speicher belegt werden muss. Dies ist für unsere Zwecke sehr geeignet, da die empfangenen Sensordaten im Normalfall räumlich begrenzt sind und weite Teile der realen Welt nicht betrachtet werden (bspw. alles unterhalb des wahrgenommenen Bodens). Das Analogon eines Octrees für den zweidimensionalen Fall ist ein *Quadtree* (Viererbaum) mit jeweils vier Söhnen. Quadtrees werden für die Speicherung der zu generierenden zweidimensionalen Karten verwendet.

Die Knoten eines herkömmlichen Octrees kennen drei Zustände: belegt, frei und gemischt. Die ersten beiden Arten von Knoten repräsentieren homogene Regionen und sind somit stets Blätter, gemischte Knoten haben jeweils wieder acht Söhne. Abbildung 4.4 veranschaulicht dieses Konzept an einem Quadtree.

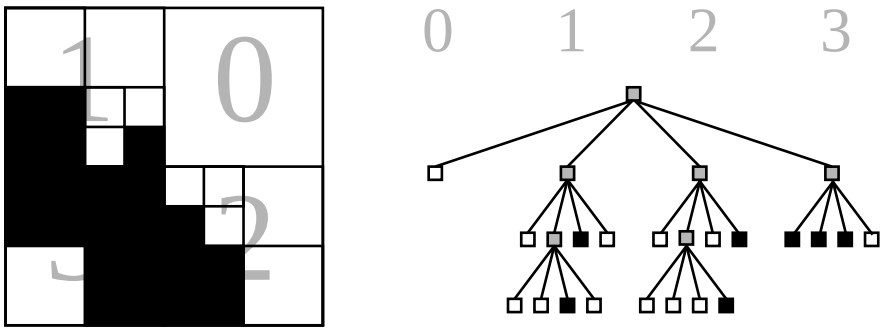


Abbildung 4.4: Repräsentation einer Unterteilung einer Fläche in einem Quadtree. Belegte Knoten sind schwarz, freie weiß und gemischte grau dargestellt.

In der Umsetzung für diese Arbeit steht an der Stelle eines belegten Knotens wie bereits dargelegt nicht nur die Information „belegt“, sondern eine Reihe von Daten, die die so repräsentierte Gitterzelle betreffen. Zwei Gitterzellen werden im Allgemeinen nie den gleichen Inhalt haben, so dass es keine homogenen belegten Gebiete geben wird. Die Information, sofern es welche gibt, betrifft stets eine Einheit des maximalen Granulierungsgrades und befindet sich somit stets in Blättern auf der Ebene der maximalen Tiefe des Octrees.

Die Unterteilung des Raumes findet bei einem Octree stets in der Mitte der Koordinatenachse jedes Oktanten statt, unabhängig von der Ausprägung der zu repräsentierenden Umgebung. So hängt die Anzahl der nötigen Unterteilungen und somit der Speicherbedarf von der willkürlichen Wahl des Ursprungs des Octrees ab (vergleiche Abbildung 4.5). Eine flexiblere Unterteilung der Umwelt liefert der *Parti-Game-Algorithmus* (Moore und Atkeson, 1995). Nachteilig daran ist, dass die Gesamtgröße der Umgebung im Vorfeld bekannt sein muss und das Verfahren auf Grund seines hohen Rechenaufwands der Echtzeit-Bedingung nicht genügt. Diese Einschränkungen bringt ein anderer Ansatz, die Verwendung von *Variable Resolution Maps* (Arleo et al., 1999), nicht mit sich. Diese unterteilen die räumliche Repräsentation anhand detektierter Kanten, um eine optimale Ausnutzung von Homogenitäten der Umgebung zu erhalten. Dies stellt eine Variante des in Abschnitt 3.1.3 erwähnten Ansatzes der Generierung topologischer Karten aus geometrischen Informationen („first metric, then

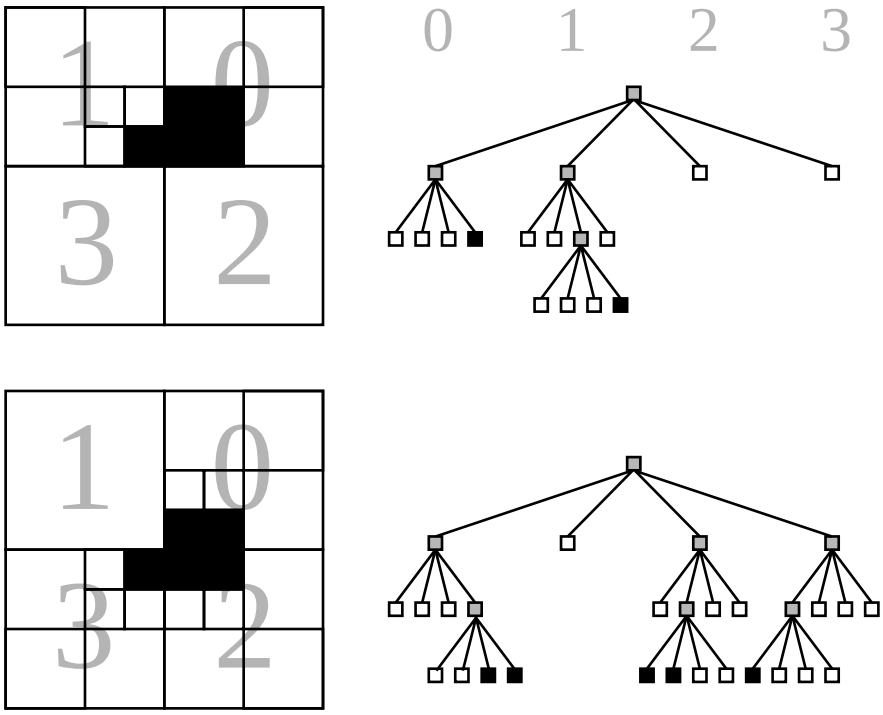


Abbildung 4.5: Speicherbedarf einer Szene in einem Quadtree in Abhängigkeit zur Wahl des Ursprungs: Eine identische Szene benötigt im oberen Beispiel 17 Knoten, im unteren 29.

topological“) (Thrun und Bücken, 1996) dar, da die erzeugte Repräsentation in einem Beziehungsgraph zwischen homogenen Regionen resultiert. Allerdings geschieht dies hier in Echtzeit während der Fortbewegung des Roboters, und die Ursprungsinformation wird sofort nach Verarbeitung wieder verworfen. Das größte Problem dieses Ansatzes ist seine Beschränkung auf isoorientierte Kanten, so dass das Verfahren bislang nur für Indoor-Navigation verwendet wird, bspw. in Cho et al. (2001).

Zur Repräsentation eines Weltmodells für Laufmaschinen sind diese Ansätze nicht geeignet. Die genannten Einschränkungen von Octrees bzgl. der Speicherungeffizienz müssen deshalb hingenommen werden.

4.4.3 Komplexitätsbetrachtungen für die Octree-Repräsentation

Die Speicherersparnis bei der Verwendung von Octrees ist mit einem erhöhten Aufwand an Rechenzeit verbunden. Da die Steuerung des Roboters in Echtzeit erfolgen muss, soll hier die Zeitkomplexität der verwendeten Operationen auf der Octree-Datenstruktur betrachtet werden. Dazu wird auch kurz auf die Realisierung der Funktionalitäten eingegangen.

Die Höhe $h(n)$ eines Octrees, der einen Raum mit einer Ausdehnung von n Zellen auf jeder Achse, also insgesamt n^3 Zellen, beschreibt, ist

$$h(n) = \lceil \log_8 n^3 \rceil = \lceil 3 \log_8 n \rceil = O(\log n). \tag{4.1}$$

Operation	Aufwand
Initialisierung	$O(1)$
Einfügen	$O(\log n)$
Report	$O(\log n)$
Löschen	$O(\log n)$
Vergrößern	$O(1)$
Scrolling	$O(1)$
Generierung Fußumgebungskarten	$O(\log n)$
Generierung globaler Karten	$O(k \log n)$

Tabelle 4.1: Aufwand der Operationen auf dem Octree mit k belegten Zellen und einer Ausdehnung von n für jede Dimension.

Daraus ergeben sich für die Basis-Operationen auf der Datenstruktur folgende Aufwandsabschätzungen (siehe auch Tabelle 4.1):

Initialisierung Es muss nur ein Wurzelknoten angelegt werden, der Aufwand ist $O(1)$.

Einfügen einer Zelle Hierzu muss höchstens auf jeder Ebene des Baumes eine Unterteilung durchgeführt werden. Da diese in $O(1)$ durchführbar ist, ergibt sich ein Aufwand für die Einfügeoperation von $O(\log n)$.

Report einer Zelle Es fallen $h(n)$ Vergleichsoperationen an, um den jeweils richtigen Unterbaum zu ermitteln, der Aufwand beträgt also $O(\log n)$.

Löschen einer Zelle Die Zelle ist in $O(\log n)$ ermittelt. Nach dem Entfernen der Daten muss der Unterbaum entfernt werden ($O(1)$), falls er keine Information mehr enthält, dies muss ggf. $(h(n)-1)$-mal bis hin zur Wurzel wiederholt werden. Der Gesamtaufwand ist demnach $O(\log n)$.

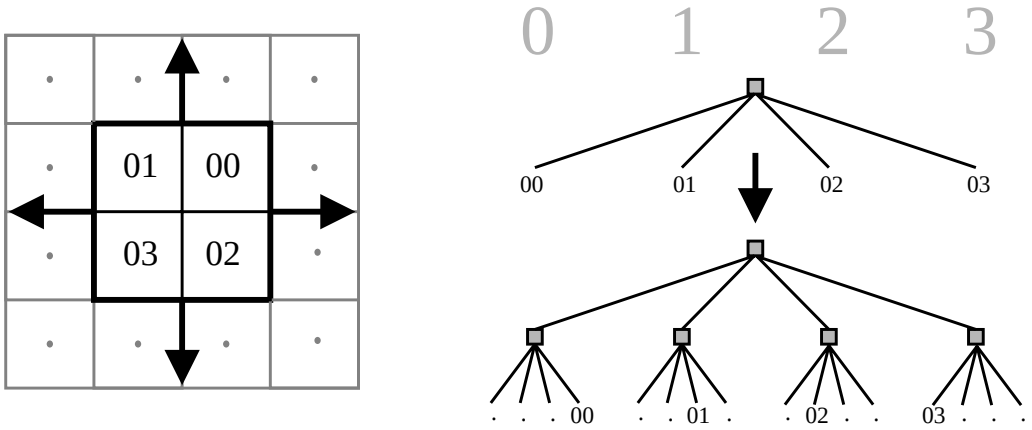


Abbildung 4.6: Vergrößern eines Quadtrees: Durch Einziehen einer weiteren Ebene ist dies mit konstantem Aufwand möglich.

Vergrößern der Karte Um nicht unnötig Hauptspeicher und Rechenzeit zu verbrauchen, empfiehlt es sich, die Umweltmodellierung stets mit einer pessimistischen Annahme über die Ausdehnung des zu kartografierenden Bereichs zu beginnen (also im Zweifelsfall zu klein) und die Größe der Karte erst bei Bedarf anzupassen. Soll eine Zelle außerhalb des Definitionsraums eingefügt werden, werden einfach die Ausdehnungen aller Achsen verdoppelt. Der neue Octree ergibt sich, indem man zwischen Wurzel und erster Ebene eine weitere einfügt und die acht Teilbäume der früheren ersten Ebene geeignet an die neuen Knoten „anhängt“ (Abbildung 4.6). Das Vergrößern der Karte hat also nur einen Aufwand von $O(1)$.

Scrollen Für die Bewegung von in Octrees gespeicherten Volumenmodellen gibt es eine Reihe eleganter und effizienter Verfahren, die auch in Echtzeit funktionieren (vergleiche bspw. Smith et al. (1994)); dennoch führt dies weiterhin zu einem großen Rechenaufwand – es ist nicht zu vergessen, dass die Speicherebene des Weltmodells nicht der einzige Prozess ist, der der Echtzeitbedingung genügen muss. Ein „weiches“, zellengenaues Scrolling ist allerdings für die hier verfolgten Zwecke gar nicht notwendig. Stattdessen wird, wenn der Aufnahmebereich des Roboters die Grenze des Definitionsraums des Gitters erreicht, auf der gegenüberliegenden Seite ein Viertel des Raumes „abgeschnitten“ und ein leeres Stück entsprechender Größe dort angefügt. Dies lässt sich, ähnlich wie die Vergrößerung, durch einfaches Umzeigern der Unterbäume realisieren (siehe Abbildung 4.7). Der Aufwand der Scrollfunktion in dieser Ausprägung ist also ebenfalls $O(1)$.

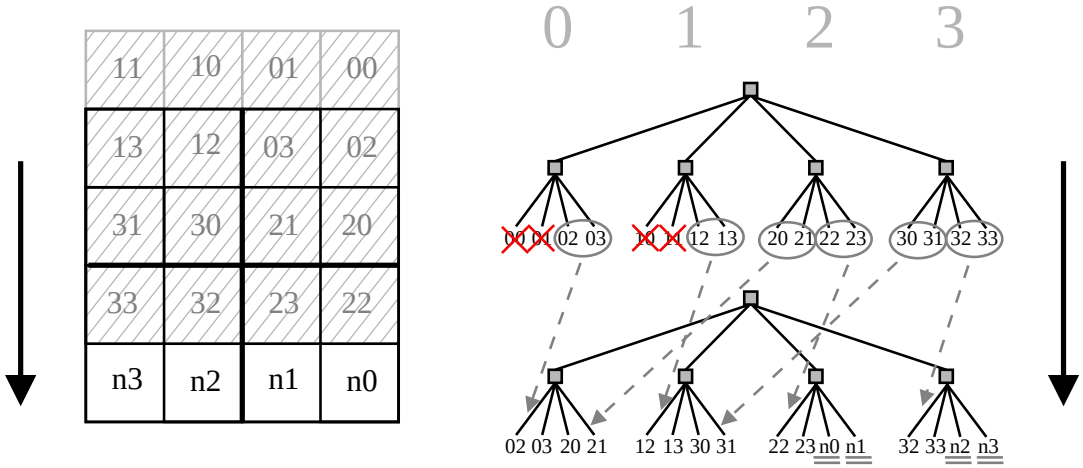


Abbildung 4.7: Schematische Darstellung des Scrollverfahren an einem Quadtree: Die Karte scrollt um ein Viertel der Ausdehnung nach unten; die Ursprungskarte ist grau schraffiert, die neue Karte ist die dick umrandete. Die Unterbäume 00 bis 11 werden verworfen bzw. ausgelagert, neue (leere) Bäume $n0$ bis $n3$ werden angehängt, die restlichen Unterbäume entsprechend neu verzeigert.

Generierung von Fußumgebungskarten Für die Erzeugung dieser Karten wird eine Anzahl von Zellen in der Umgebung der den geplanten Fußaufsetzpunkt repräsentierenden Zelle betrachtet. Da die Anzahl dieser Nachbarpositionen für

alle Dimensionen konstant ist, ergibt sich durch den Reportingaufwand nur ein Gesamtaufwand von $O(\log n)$.

Generierung von 2D-Karten Der naive Algorithmus, der alle Zellen (x, y, z) mit maximalem z sucht und in einer neuen Karte speichert, benötigt n^3 Anfragen und hat somit einen Aufwand von $O(n^3 \log n)$. In Abschnitt 5.4 ist allerdings ein Algorithmus angegeben, der diese Aufgabe unter Ausnutzung der Tiefensuche mit einem Aufwand von $O(k \log n)$ bewältigt (siehe dort), wobei k die Anzahl des tatsächlich im Octree gespeicherten Zellen ist und im schlimmsten, aber in der Realität nie erreichten Fall, natürlich auch n^3 sein kann.

4.4.4 Speicherung auf einem externen Datenträger

Obwohl die Repräsentation eines Raumes durch einen Octree eine sehr große Platzersparnis bringt, ist der Speicherverbrauch letztendlich immer noch hoch. Es gibt daher einige Ansätze zur Optimierung der Datenhaltung, einer davon ist die *DF-Repräsentation* (Kawaguchi und Endo, 1980). Sie stellt eine lineare Codierung des Octrees (bzw. Quadtrees) dar: Ein belegter Knoten wird mit „B“ (black) ein freier mit „W“ (white) und ein gemischter Knoten, also ein Nicht-Blatt, mit „(“ codiert. Verwendet wird eine Präfix-Notation ausgehend von der Wurzel, so dass sich die Reihenfolge der Buchstaben aus einer Tiefensuche ergibt. Der Quadtree aus Abbildung 4.4 hätte demnach die Repräsentation

```
(W((W(WWBWBW(W(WWWBWB(BBBW
```

Da dieses Alphabet nur aus drei Zeichen besteht, genügen 2 bit für jeden der k Knoten des Baumes, um dessen gesamte Struktur zu speichern. Die zu den belegten Knoten gehörigen Daten lassen sich in einer linearen Liste abspeichern. Der Aufwand für die Erstellung dieser Repräsentation ist $O(k \log n)$, da dies auch für die Tiefensuche benötigt wird, weil ein Octree mit k belegten Zellen $O(k \log n)$ Knoten hat. Die gilt allerdings nur für sehr spärlich besetzte Bäume, im realistischen Normalfall ist die Beziehung zwischen k und der Anzahl der Knoten linear.

In Abschnitt 4.1 wurde auf die Möglichkeit hingewiesen, aus dem Bereich der Urkarte „herausgescrollte“ Bereiche auf einen externen Datenträger auszulagern. Genau hier bietet sich die *DF-Repräsentation* an, da die Daten auch auf einer Festplatte sequenziell hintereinander abgelegt werden. Es ist mit dieser Darstellung also möglich, das Speichervolumen und damit auch den Datentransfer-Aufwand auf ein Minimum zu begrenzen und die Daten sowohl zumeist in linearer Zeit zu speichern als auch wieder auszulesen und den Octree neu aufzubauen.

4.5 Die Gitterzelle

Die Gitterzelle ist die Datenzentrale des 3D-Weltmodells. In diesem Kapitel wird zu diskutieren sein, wie sie ausgestaltet und dimensioniert werden sollte und welche Daten sinnvollerweise gehalten werden. Wie in Abschnitt 4.3 gefordert, ist die Gitterzelle

auch der Ort, der alle Information aufnimmt, die unmittelbar mit der physikalischen Ausprägung der Maschine zusammenhängen.

4.5.1 Dimensionierung der Zellen

Zunächst stellt sich die Frage nach der Auflösung des über die Umwelt gelegten Gitters, also der Größe der Gitterzellen. Eine sinnvolle Wahl muss natürlich in erster Linie die Abmessungen des Roboters berücksichtigen. Unabhängig davon ist es sinnvoll, das Gitter bzgl. der Höhenachse (z-Achse) feiner zu granulieren. Auch der Mensch nimmt Höheninformationen empfindlicher wahr als Ausdehnungen in xy-Richtung. Für den Roboter sind Ausdehnungen in z-Richtung gerade für das zielgerichtete Aufsetzen relevant, da sie die erforderliche Schritthöhe bestimmen. Die Granulierung in der xy-Ebene muss dagegen nicht so fein gewählt werden. Hier bietet sich die Ausdehnung der Füße der Maschine als Maß an. Damit würde jede Zelle genau einen möglichen Fußaufsetzpunkt beschreiben.

In dieser Arbeit wurde daher ausgehend von der Größe der Füße von LAURON eine Gitterzellengröße von $5 \times 5 \times 1\,\text{cm}$ gewählt.

4.5.2 Information und Metainformation – der Inhalt der Gitterzelle

In jeder Gitterzelle wird sowohl gespeichert, was an Wissen über diesen Bereich der Umwelt relevant erscheint, als auch „Metadaten“, die zur Bewertung und Einordnung der Information herangezogen werden können. Als solche Metadaten kann bspw. Wissen darüber gehalten werden, auf welche Weise die in der Zelle gespeicherten Informationen akquiriert wurden.

Die wichtigste Information in einem Belegtheitsgitter ist, was über die durch die Zelle repräsentierte wirkliche Welt ausgesagt werden kann. An solchen unmittelbaren Informationen wird in der 3D-Gitterzelle Folgendes gespeichert:

Belegtheitswahrscheinlichkeit $\text{occ}(x) \in [0, 1]$ bezeichnet die Sicherheit, mit der die Zelle x belegt ist.

Zuverlässigkeit $\text{cred}(x) \in [0, 1]$ gibt eine Wahrscheinlichkeit an, mit der die für diese Zelle gespeicherten Daten glaubwürdig sind.

Es stellt sich die Frage, wieso mit $\text{cred}(x)$ neben $\text{occ}(x)$ noch ein weiterer stochastischer Wert gehalten wird und wo der Unterschied liegt. Als Hinleitung zu dieser Entscheidung mag das Beispiel einer Tischkante dienen, die genau zur Hälfte in die zu betrachtende Gitterzelle „hinein ragt“; dazu nehmen wir an, man würde nur die Belegtheitswahrscheinlichkeit speichern. Ideale Sensorwerte vorausgesetzt würde man $\text{occ}(x) = 0{,}5$ erwarten, da die Hälfte der Messungen ein Hindernis erfassen würde, die andere Hälfte nicht. Die 0,5 würden also genau das Richtige aussagen: dass die Zelle zur Hälfte belegt ist. Fällt aber die Voraussetzung idealer Sensorik weg, lässt sich diese Aussage nicht mehr halten, da genauso gut eine komplett belegte Zelle und 50% Fehlmessungen vorliegen könnten. Der Schwachpunkt der Messungen ist also nicht

ersichtlich – es kann sowohl an schwacher Sensorleistung als auch an der nicht gittertauglichen realen Welt liegen.

Die Aufspaltung in eine Belegtheitswahrscheinlichkeit und Zuverlässigkeit behebt dieses Manko. $\mathrm{occ}(x)$ liefert eine Beschreibung der Umwelt am Ort x, so gut es die Sensorik eben hergibt, $\mathrm{cred}(x)$ die Information, wie sehr dieser Beschreibung zu trauen ist. Damit erhält man bspw. auch die Möglichkeit, auf der Weltmodellierung basierende Entscheidungen mit verschieden hoher Risikobereitschaft zu treffen. Um auf die Tischkante zurückzukommen: $\mathrm{occ}(x) = 0{,}5$ und $\mathrm{cred}(x) = 0{,}9$ würde mit großer Sicherheit uneinheitliches Terrain anzeigen, das man besser nicht betritt. Werte von $\mathrm{occ}(x) = 0{,}9$ und $\mathrm{cred}(x) = 0{,}5$ deuten, wenn auch nicht mit allzu großer Sicherheit, auf eine einheitliche Landschaft hin. Im zweiten Fall wäre die Konsequenz, durch zusätzliche Messungen mehr Information zu akquirieren, im ersten Fall würde das wenig Sinn machen.

Zusätzlich zu den genannten Datenfeldern ist es sinnvoll, eine Reihe von *Metadaten* in jeder Zelle zu halten, die zusätzliche Information über die gespeicherten unmittelbaren Daten liefern. Für Metadaten hat die Zelle folgende Felder:

Anzahl vorheriger Messungen Zunächst wird gespeichert, wie oft sich eine Messung auf diese Zelle ausgewirkt hat. Dies ist ein wichtiges Indiz für die Zuverlässigkeit der gespeicherten Werte. Da die Anzahl der Messungen monoton steigend ist, wäre zu überlegen, ob dieser Wert nicht ggf. zurückgesetzt werden sollte. Eine Möglichkeit wäre, dies zu tun, wenn die letzte Messung eine gewisse Altersgrenze überschreitet.

Alter der letzten Messung Die Aktualität der in der Zelle gehaltenen Daten ist ein wichtiger Indikator, wie stark sich auf diese verlassen werden kann. Dazu wird bei jedem Schreibzugriff auf die Zelle, auch wenn sie nicht modifiziert werden sollte, ein aktueller Zeitwert gespeichert. Der Aussagewert dieser Altersangabe ist allerdings leicht eingeschränkt, da nur das Alter der letzten Messung ablesbar ist, die Information der Zelle aber aus sehr vielen, eventuell sehr viel älteren Messungen, gebildet wurde. Will man die Auswirkungen dieser Einschränkung verringern, kann man statt eines Zeitwertes eine Reihe der letzten n Messungen speichern und dann deren arithmetisches Mittel betrachten. Für kleine n lohnt dies allerdings nicht, da üblicherweise mehrere kurz aufeinander folgende Messungen ein und dieselbe Zelle betreffen, ein zu großer Wert n treibt den Speicherverbrauch der Datenstruktur unzulässig in die Höhe. Auch der Ansatz der *Temporal Occupancy Grids* (Arbuckle et al., 2002), der Belegtheitsinformationen in verschiedenen Zeitskalen speichert, hat den Nachteil eines immens hohen Speicherverbrauchs. In dieser Arbeit wurde es deshalb so gehalten, ausschließlich den Zeitwert des letzten Zugriffs zu speichern und bei der Bewertung des Alters die Anzahl der vorherigen Messwerte geeignet zu berücksichtigen.

Sensordichte Die Sensordichte gibt an, wie viele verschiedene Quellen an der in der Zelle gespeicherten Information beteiligt waren. Diese Quellen sind in erster Linie Sensoren, können aber auch mittelbare andere Informationsquellen sein. Beispielsweise ist das Wissen, dass an der gegenwärtigen Position des Roboterkörpers

kein Hindernis sein kann, eine sehr verlässliche Information, die nicht direkt aus der Sensorik abgeleitet wird.

4.6 Unscharfe Regelung zur Bewertung der Qualität von Sensormessungen

In diesem Abschnitt wird ein auf unscharfer Logik basierendes Verfahren vorgestellt, das eine qualitative Bewertung der eingehenden Sensordaten ermöglicht.

4.6.1 Motivation des Bewertungsvorgangs

Wenn neue Sensordaten vorliegen, müssen diese in das bestehende Weltmodell integriert werden. Es stellt sich die Frage, wie man vorgeht, um die in der betreffenden Zelle des Belegtheitsgitters gespeicherten Daten auf Grund der neu gewonnenen Informationen zu aktualisieren. Dass die alten Daten nicht einfach überschrieben werden sollen, steht außer Frage.

Eine geeignete Möglichkeit, die bspw. auch in Murphy et al. (2002) verwendet wird, ist den Einfluss alter Daten exponentiell mit dem Alter abnehmen zu lassen. Dies berücksichtigt jedoch nur einen einzigen Parameter. Kruse et al. (1995) führen eine allgemeinere Gütefunktion ein, die den Einfluss der Messungen aufgrund der angenommenen Qualität der Messung ermittelt. Hier soll nun eine solche Funktion definiert werden, die Parameter für die Aktualisierung der Zelldaten liefert. Dabei soll nach Möglichkeit alles an Wissen einbezogen werden, was sowohl über die bereits gespeicherten Daten als auch über die aktuelle Messung an Rahmenparametern bekannt ist. Aufgrund der Ausgabe dieser Gütefunktion soll dann der Einfluss der neu einkommenden Daten gewichtet werden.

4.6.2 Weitere Einflussgrößen für die Bewertung

Im letzten Abschnitt wurden bereits einige Größen genannt, die zur Bewertung der alten und neuen Daten herangezogen werden können. Einige weitere sollen in diesem Abschnitt eingeführt werden:

Sensorqualität Die Sensorqualität ist ein Wert zwischen 0 und 1, der eine generelle Aussage darüber trifft, wie sehr den Daten des aktuell messenden Sensor zu trauen ist. Diese Bewertung ist Expertenwissen und muss von außen vorgegeben werden.

Nachbarschaftsbeziehung Bei der Betrachtung der Nachbarschaftsbeziehung geht man von der simplen Einsicht aus, dass Hindernisse meist zusammenhängend sind und einzelne Punkte selten frei im Raum schweben. Eine Messung einer belegten Zelle in Nachbarschaft anderer Zellen lässt im Normalfall nicht auf einen Messfehler schließen. Es macht also Sinn, ein Maß für den Belegtheitsstatus der unmittelbaren Nachbarschaft der Zelle als zusätzliche Einflussgröße zu betrachten.

Sei D_x die Menge aller Zellen in Nachbarschaft von einer Zelle x, für die schon Beobachtungen vorliegen. Dann ist für einen für x beobachtbaren Belegtheitswert $s \in \{0, 1\}$

$$n(x, s) = \frac{\sum_{y \in D_x} |d(x, y)(s - \text{occ}(y)\text{cred}(y))|}{|D_x|}$$

ein Maß für die Ähnlichkeit der Nachbarschaft von x. $d(x, y)$ ist hierbei ein Gewichtungsfaktor, für den sich bei größer gewählten Nachbarschaften die Entfernung von y zu x anbietet, um nähere Nachbarzellen stärker zu gewichten. Für diese Arbeit wurden stets nur die unmittelbar angrenzenden Zellen betrachtet und als Abstandsmaß $d(x, y) \equiv 1$ verwendet.

Bewegungsstatus des Roboters Es ist beobachtbar, dass es vor allem dann zu ungenauen Messungen kommt, wenn der Roboter in starker Bewegung ist. Vor allem liegt das daran, dass einige Sensorwerte einen recht starken Median- und Durchschnitts-Filter durchlaufen. Durch deren integrativen Effekt reagieren die Sensoren auf Veränderungen durch Positionsverlagerungen langsamer als die Sensoren in den Gelenken, die diese Filter nicht durchlaufen. In einem relativen Ruhezustand der Maschine sind also verlässlichere Sensorwerte erwartbar.

Zusammenfassend dienen somit als Eingangsgrößen:

- Zuverlässigkeit der bestehenden Daten
- Alter der letzten Messung
- Anzahl vorheriger Messungen
- Sensordichte
- Sensorqualität
- Nachbarschaftsbeziehung
- Bewegungsstatus des Roboters

4.6.3 Aktualisierung des Zellinhalts

Mit Hilfe dieser Eingangsgrößen soll die aufzustellende Gütefunktion zwei Werte als Ausgabe liefern:

- Einen *Zuverlässigkeitsparameter* k_{cred}, der eine Aussage über die Verlässlichkeit der aktuellen Messung liefert. Zu dessen Berechnung werden keine Daten aus dem bisherigen Datenbestand der Zelle herangezogen.
- Einen *Einflussparameter* k_{in}, der dazu dient, zu bestimmen, wie stark die aktuelle Messung (und auch die gleichzeitig geschätzte Zuverlässigkeit) in den Datenbestand eingehen soll.

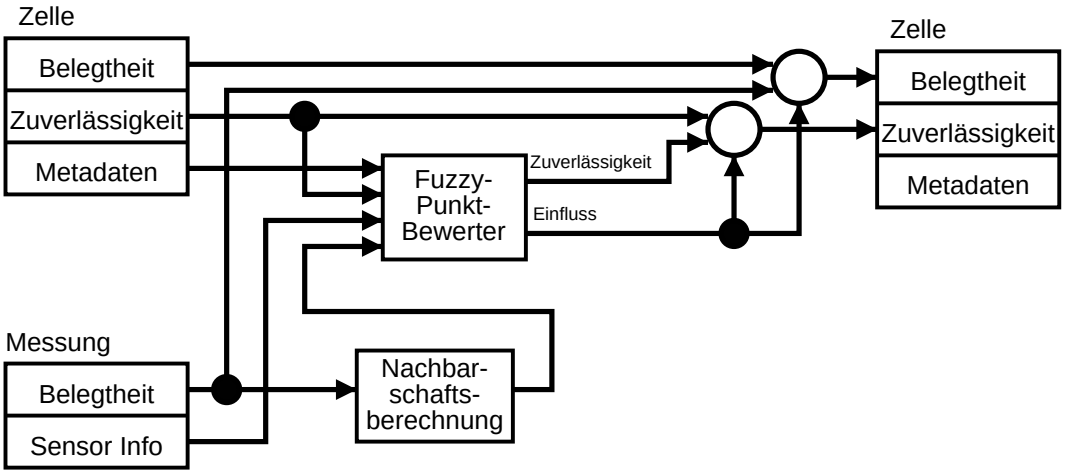


Abbildung 4.8: Schematische Darstellung des Datenflusses bei der Berechnung der neuen Werte einer Gitterzelle

Die zu betrachtende Zelle x hält zu einem Zeitpunkt t die Werte $\mathrm{cred}_t(x)$ und $\mathrm{occ}_t(x)$. Eine neue Sensormessung zum Zeitpunkt $t+1$ liefert für die Zelle einen neuen Belegtheitszustand $occ_{t+1}(x) \in \{0,1\}$ – entsprechend, ob ein Hindernis erkannt wurde oder nicht. Mit den aus der Gütefunktion gewonnenen Parametern k_{cred} und k_{in} aktualisiert sich der Zellinhalt nun wie folgt:

$$\begin{aligned} \mathrm{occ}_{t+1}(x) &= (1-k_{in})\mathrm{occ}_t(x) + k_{in}\mathrm{occ}_{t+1}(x) \\ \mathrm{cred}_{t+1}(x) &= (1-k_{in})\mathrm{cred}_t(x) + k_{in}k_{cred} \end{aligned}$$

Abbildung 4.8 stellt die Ein- und Ausgänge und ihre gegenseitige Einflussnahme noch einmal schematisch dar. Für die Realisierung der Gütefunktion wurde eine Bewertung realisiert, die mit Methoden der Fuzzy-Logik arbeitet.

4.6.4 Einführung in die Fuzzy-Methodiken

> „Je komplexer ein System wird, desto weniger sind wir in der Lage, präzise und gleichwohl bedeutsame Aussagen über sein Verhalten zu machen, bis ab einem bestimmten Punkt sich Präzision und Bedeutsamkeit wechselseitig ausschließen.“ (Zadeh, 1973)

Die Regelung komplexer Prozesse mit Hilfe von Unscharfer Logik (*Fuzzy Logic*), die 1965 von Lotfi A. Zadeh vorgestellt wurde (Zadeh, 1965), ist im Laufe der Zeit zu einer festen Größe in der Regelungstechnik geworden. Zunächst noch mit wenig positiver Resonanz bedacht, konnten in der zweiten Hälfte der 70er Jahre des letzten Jahrhunderts mehrere unscharfe Regelsysteme erfolgreich in der Industrie eingesetzt werden. Ende der 80er Jahre bescherten vor allem japanische Entwicklungen der Unscharfen Logik einen regelrechten Boom, der sich bis in die halb- und unwissenschaftliche Presselandschaft ausbreitete. Ungeachtet der genauso euphorischen wie unrealistischen Erwartungen dieser Zeit, spielen Fuzzy-Systeme, zumeist in Kombination mit anderen

Methodiken, weiterhin eine unbestritten bedeutende Rolle in wissenschaftlichen und industriellen Projekten.

In diesem Abschnitt sollen zunächst in gebotener Kürze die benötigten Grundlagen erläutert werden. Für eine ausführliche praxisrelevante Darstellung von Fuzzy-Methoden sei auf Kahlert (1995) verwiesen, eine sehr gute und ausführliche theoretische Aufbereitung des Themas „Unschärfe“ findet sich in Biewer (1997).

4.6.4.1 Fuzzy-Mengen und linguistische Terme

Die Unscharfe Logik bietet die Möglichkeit, umgangssprachlich formulierte Sachverhalte mathematisch zu repräsentieren und weiter zu verarbeiten. So kann empirisch gewonnenes, nicht in exakter Form vorliegendes Modellwissen als Grundlage komplexer Regelprozesse dienen.

Eine *Fuzzy-Menge* A über einem Wertebereich X ist definiert als eine geordnete Menge von Paaren

$$A := \{x, \mu_A(x) \,|\, x \in X\}$$

wobei die Abbildung

$$\mu_A : X \to [0,1], \;\; \mu_A \in \mathbb{R}$$

jedem Element von X einen *Zugehörigkeitsgrad* $\mu_A(x)$ zuordnet. Diese Funktion heißt *Zugehörigkeitsfunktion* von F. Da eine Fuzzy-Menge durch ihre Zugehörigkeitsfunktion vollständig und eindeutig definiert ist, wird der Einfachheit halber gerne μ_A als Bezeichner für eine Fuzzy-Menge verwendet.

Eine *linguistische Variable* ist eine natursprachliche Größe, bspw. „Geschwindigkeit“. Sie wird im Allgemeinen durch eine Reihe so genannter *linguistischer Terme* beschrieben, die Zustände bzgl. der linguistischen Variable umgangssprachlich beschreiben (bspw. „langsam“, „schnell“, „sehr schnell“), und denen jeweils eine Fuzzy-Menge zugeordnet ist. Die Gesamtheit der Fuzzy-Mengen überdeckt den gesamten Wertebereich der Variablen[2].

Bildet man einen *scharfen*, also numerisch wohldefinierten, Wert einer linguistischen Variable (in unserem Beispiel könnte das 50 km/h sein) auf die linguistischen Terme ab, indem man für den scharfen Wert den jeweiligen *Zugehörigkeitsgrad* zu jeder Fuzzy-Menge ermittelt, spricht man von *Fuzzyfizierung*. Ein Beispiel dazu wird in Abbildung 4.12 erläutert.

4.6.4.2 Operatoren auf Fuzzy-Mengen

Auf Fuzzy-Mengen kann man, wie auch in klassischen Logiken, logische Operatoren definieren. Es existiert eine Vielzahl unterschiedlicher Möglichkeiten, dies zu tun. Hier sollen, wie auch in den folgenden Abschnitten, nur jeweils diejenigen aufgezeigt werden, die auch im vorliegenden System verwendet wurden. Die Vereinigung $\mu_{A \cup B}$ zweier Fuzzy-Mengen A und B mit den entsprechenden Zugehörigkeitsfunktionen μ_A und μ_B wird definiert als:

$$\mu_{A \cup B}(x) := \max(\mu_A(x), \mu_B(x)).$$

[2] In den Beispielen in diesem Abschnitt ist dies aus Gründen der Übersicht nicht immer so. Man denke sich zur vollständigen Überdeckung des Wertebereichs einfach noch ein paar Fuzzy-Mengen hinzu.

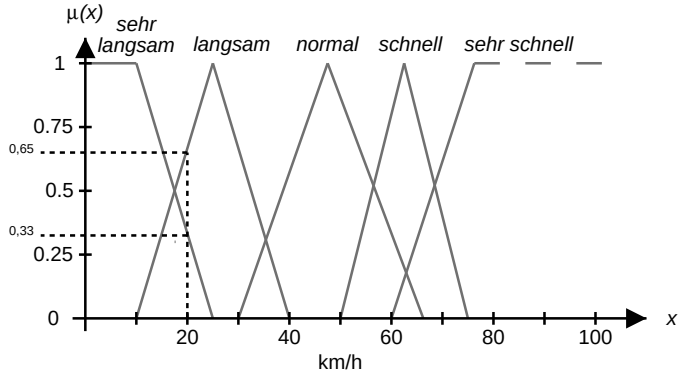


Abbildung 4.9: Fuzzy-Mengen am Beispiel der linguistischen Variable X = „Geschwindigkeit im Ortsbereich“: Auf X sind fünf Fuzzy-Mengen definiert, denen die linguistischen Termen *sehr langsam*, *langsam*, *normal*, *schnell* und *sehr schnell* zugeordnet sind. Die Fuzzy-Menge *langsam* bspw. überdeckt dabei den Wertebereich von 10 bis 40 km/h. Für den Wert 20 km/h ist eine Fuzzyfizierung skizziert: Sein Zugehörigkeitsgrad zur Fuzzy-Menge *sehr langsam* ist ungefähr 0,33, der zur Menge *langsam* 0,65. Für alle anderen Mengen ist der Zugehörigkeitsgrad 0. Für alle zur linguistischen Variablen gehörenden Fuzzy-Mengen ergibt sich also ein Zugehörigkeitstupel $\mu(20\,\mathrm{km/h}) = (0,33\,;\,0,65\,;\,0\,;\,0\,;\,0)$.

Dies entspricht dem ODER-Operator. Dem UND-Operator entspricht die Schnittmenge $\mu_{A\cap B}$; sie ist definiert als:

$$\mu_{A\cap B}(x) := \min(\mu_A(x), \mu_B(x)).$$

Die UND- und ODER-Operatoren veranschaulicht Abbildung 4.10.

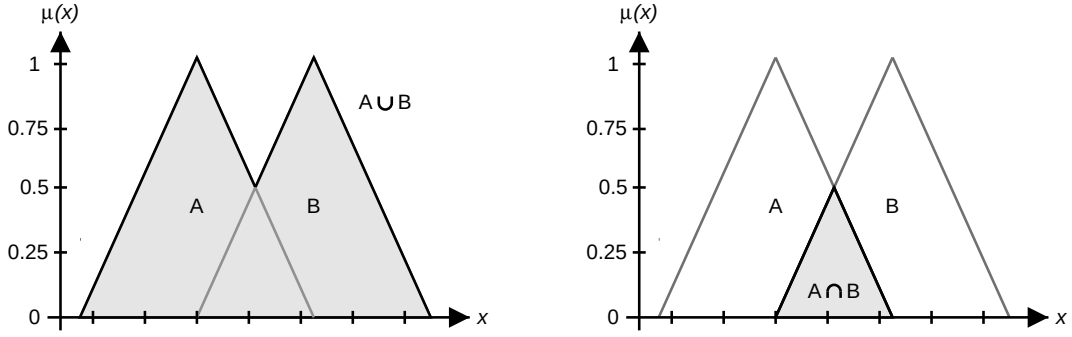


Abbildung 4.10: Links: Vereinigung zweier Fuzzy-Mengen (UND-Operator). Rechts: Schnitt zweier Fuzzy-Mengen (ODER-Operator).

4.6.4.3 Fuzzy-Implikation und Fuzzy-Inferenz

Will man empirisches Systemwissen mit unscharfen Methoden modellieren, kann man auf die *Fuzzy-Implikation* zurückgreifen, um mit linguistischen Variablen Wenn-Dann-Regeln in der Form

$$\text{WENN } x = A \text{ DANN } y = B$$

aufzustellen. Für die natursprachliche Regel „Wenn die Geschwindigkeit langsam ist, dann ist die Gefahr gering“ wäre die Eingangsgröße x die linguistische Variable „Geschwindigkeit“, die einen linguistischen Term „niedrig“ erfüllen muss, damit die Ausgangsgröße y (die Variable „Gefahr“) durch den Term „niedrig“ charakterisiert wird. Im Gegensatz zur klassischen Logik ist die Prämisse nicht entweder wahr oder falsch, sondern kann beliebige *Erfüllungsgrade* annehmen; entsprechend ist auch die Conclusio ein unscharfer Wert. Der Erfüllungsgrad einer Prämisse entspricht im einfachsten Fall dem Zugehörigkeitsgrad der entsprechenden Fuzzy-Menge.

Die Fuzzy-Implikation $\mu_{A\Rightarrow B}(x, y)$ definieren wir als:

$$\mu_{A\Rightarrow B}(x, y) := \min(\mu_A(x), \mu_B(y)).$$

Diese Definition (die so genannte *Mamdani-Implikation*) hat insbesondere die wünschenswerte Eigenschaft, das der Wahrheitsgehalt einer Schlussfolgerung nie höher sein kann als der der Prämisse. Anschaulich bedeutet dies, dass die Ausgangs-Fuzzy-Menge am Erfüllungsgrad der Eingangs-Menge „abgeschnitten“ wird. Die Auswertung eines Satzes von Regeln bezeichnet man als *Inferenz*. Eine grafische Zusammenstellung eines Inferenzvorganges zeigt Abbildung 4.11.

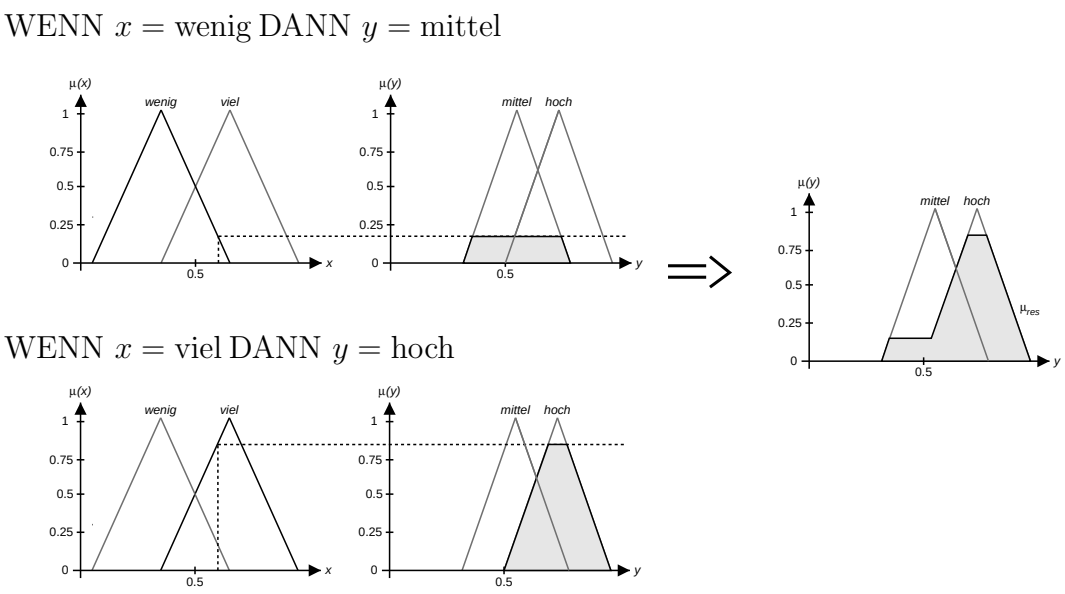


Abbildung 4.11: Fuzzy-Inferenz mit zwei Regeln über einer Eingangsbereich X mit einem Ergebnisbereich Y. Die zugehörige Ergebnismenge wird auf der Höhe des jeweiligen Zugehörigkeitsgrades der Eingangsmenge abgeschnitten. Die letztendlich resultierende Fuzzy-Menge ergibt sich durch Anwendung des Vereinigungsoperators auf die beiden Ergebnismengen.

Sind für eine Ausgangsgröße nach dem Inferenzvorgang mehrere Fuzzy-Mengen *aktiv* (d. h. der Zugehörigkeitsgrad des Eingangswerts ist größer als 0), werden diese mittels des ODER-Operators vereinigt. Die so entstandene Ergebnismenge μ_{res} muss nun noch *defuzzifiziert* werden, um einen scharfen Ausgabewert zu erhalten. Eine gebräuchliche Methode hierfür ist die so genannte *Schwerpunkt-Methode*, bei der sich der Ergebniswert aus der Projektion des Schwerpunktes der durch die Fuzzy-Menge

beschriebenen Fläche auf die Achse der Wertebereichs ergibt. Der Schwerpunkt y_{res} errechnet sich dabei aus

$$y_{res} = \frac{\int y\mu_{res}(y)\mathrm{d}y}{\int \mu_{res}(y)\mathrm{d}y}$$

4.6.5 Warum unscharfe Regeln?

Die Verarbeitung einer relativ großen Anzahl von Eingangsgrößen, wie sie in Abschnitt 4.6.2 zusammengefasst sind, macht es schwierig, eine einfache mathematische Form zu finden, die die gewünschten Verläufe beschreibt. Die Arbeit mit Unscharfer Logik ermöglicht es dagegen, auf einfache Weise, mathematisch nicht exakt fassbares Modellwissen zu verarbeiten.

Man mag anmerken, dass Fuzzy-Methoden nicht mit der Exaktheit arbeiten, die eine geschlossene mathematische Form bieten würde. Das ist natürlich richtig. Es ist dem aber entgegenzuhalten, dass die Exaktheit im Modellwissen auch gar nicht existiert. Regeln wie „Wenn die bestehenden Daten sehr alt sind, dann gewichte die neuen relativ hoch“ beschreiben insofern recht genau die simple dahinter stehende Heuristik. Eine mathematische Formel wäre im Bemühen um eine wirklichkeitsnahe Modellierung genauso willkürlich aus diesen wenig exakten Beobachtungen erstellt, es ist daher mehr als zweifelhaft, ob die Resultate, die sie liefert, der Realität eher gerecht werden.

Für die Wahl der Realisierung der Gütebewertung mit Fuzzy-Logik sprechen insbesondere folgende Gründe:

- Nichtlineare Zusammenhänge lassen sich leicht formulieren.
- Die Regeln lassen sich leicht und verständlich erstellen. Implikationen wie „Wenn A, dann B“ lassen sich, im Gegensatz zu einer parametrisierten mathematischen Gleichung, direkt so aufschreiben und verarbeiten, wie sie wahrgenommen werden. Auch sind die Zusammenhänge sofort aus den Regeln ablesbar.
- Der Regelsatz lässt sich auf einfache Weise erweitern, indem man weitere Regeln hinzufügt. Dabei muss man insbesondere nicht befürchten, dass die Ausgangsgrößen ihren Definitionsbereich verlassen.
- Die Regeln sind problemlos auf andere Roboterarchitekturen oder andere Sensorik anpassbar. Dazu müsste es meist ausreichen, wenn nur die entsprechenden Eingangs-Fuzzy-Mengen angepasst werden.
- Nicht zuletzt schützen Fuzzy-Methoden auch vor unerwünschten Auswirkungen nicht perfekter Modelle. Nicht immer wird man in der Lage sein, vollkommen konsistente Regelsätze zu erstellen. Die Fuzzy-Methoden stören sich daran wenig: Widersprüchliche Regeln werden toleriert, ihre Effekte heben sich auf.

4.6.6 Konkrete Umsetzung

Um nun zu einer Bewertung der Sensordaten zu kommen, werden für alle in Abschnitt 4.6.2 genannten Eingangsgrößen auf ihrem jeweiligen Wertebereich mehrere Fuzzy-Mengen definiert, die qualitativen Bewertungen der Größe entsprechen. Für das Alter

werden dies bspw. Terme wie „sehr alt“, „alt“ und „neu“ sein, für die Sensorqualität „hoch“ oder „gering“. Als Beispiel findet sich der gesamte verwendete Satz von Fuzzy-Mengen für die Eingangsgröße „Anzahl der Messungen“ in Abbildung 4.12.

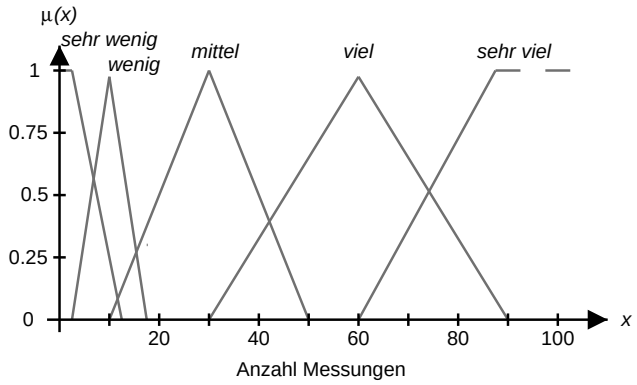


Abbildung 4.12: Fuzzy-Mengen für die Eingangsgröße „Anzahl der Messungen“. Die Größe der Mengen ist unterschiedlich, Messanzahlen im sehr geringen Bereich können so im Regelsatz gesondert berücksichtigt werden, hier fallen auch kleinere Unterschiede stark ins Gewicht.

Genauso werden auch für die beiden Ausgangsgrößen „Einfluss“ und „Zuverlässigkeit“ Fuzzy-Mengen definiert. Danach können Implikationsregeln in der Form

WENN Nachbarschaft = mittel	DANN k_{cred} = hoch
WENN Sensorqualität = hoch	DANN k_{in} = hoch
WENN Messanzahl = viel UND Alter = veraltet	DANN k_{in} = sehr hoch

aufgestellt werden. Für jede Sensormessung wird nun ein Inferenzschritt durchgeführt und so die Parameter k_{in} und k_{cred} ermittelt. Abbildung 4.13 zeigt an zwei Beispielen das Ergebnis der Inferenz: Es sind die Werte der beiden Ausgabeparameter aufgetragen, wenn man jeweils eine Eingabegröße (hier Alter der letzten Messung und Anzahl der Messungen) ihren gesamten Definitionsraum durchlaufen lässt. Der komplette verwendete Regelsatz findet sich im Anhang B.3.

Die für diese Arbeit verwendete Steuerungsarchitektur MCA2 (vgl. Abschnitt 5.1.3) stellt bereits Klassen für die Fuzzy-Inferenz zur Verfügung. Mit Hilfe eines Perl-Scripts kann der erstellte Regelsatz in maschinenlesbare Form gebracht werden: Es kann entweder eine statische C++-Klasse erzeugt werden, die zum Compilierzeitpunkt fest in den Programmcode eingebunden wird, oder es wird eine Regeldatei erzeugt, die zur Laufzeit von dem entsprechenden Modul eingelesen wird. Letztere Methode ermöglicht es insbesondere, während der Verlaufs einer Steuerung die Strategie des Einfügens der Sensordaten zu ändern, indem man einen neuen Regelsatz einliest. So könnte auf Änderungen der Rahmenbedingungen reagiert werden und bspw. etwas „vorsichtigere“ Regeln verwendet werden, wenn sich die Maschine in erkennbar schwierigeres Terrain bewegt.

Die konkrete Implementierung der verwendeten Fuzzy-Control-Klassen ist in Luksch (2001) beschrieben.

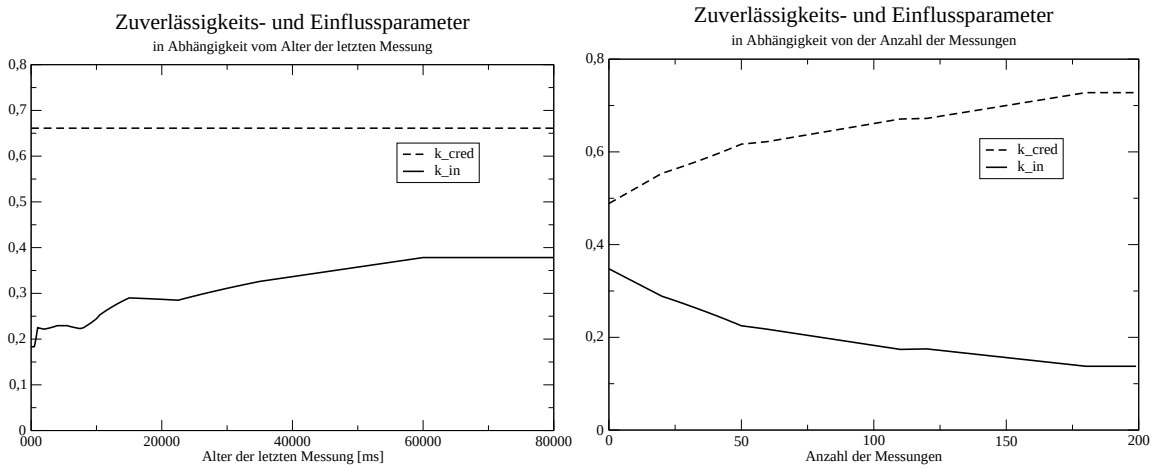


Abbildung 4.13: Verlauf der Ausgabeparameter bei unterschiedlichen Eingabewerten: Für beide Diagramme wurden bis auf eine alle Eingangsgrößen konstant gehalten und eine ihren gesamten Definitionsraum durchlaufen lassen (links: Alter der letzten Messung, rechts: Anzahl der Messungen). Im linken Beispiel ist k_{cred} konstant, da dieser Parameter nicht vom Alter der letzten Messung abhängt.

Kapitel 5

Realisierung des Weltmodells

5.1 Die Laufmaschine Lauron

Die Arbeit am Urmodell LAURON I (Cordes et al., 1993) begann zunächst in einer Simulationsumgebung. 1992 entstand ein erstes Holzmodell, und 1994 wurde die sechsbeinige Maschine dann auf der CeBIT in Hannover erstmals einer breiten Öffentlichkeit vorgestellt. Die Geometrie des Roboters orientiert sich am biologischen Vorbild der Stabheuschrecke. Die Steuerung der Maschine bzgl. Beinsteuerung, Beinkoordination und reaktiver Elemente erfolgt unter Zuhilfenahme von neuronalen Methoden, woraus sich auch der Name LAURON (LAUfender ROboter, neuronal gesteuert) erklärt. Mittlerweile ist LAURON I ein Museumsstück, ein Exemplar findet sich im Deutschen Museum in Bonn.

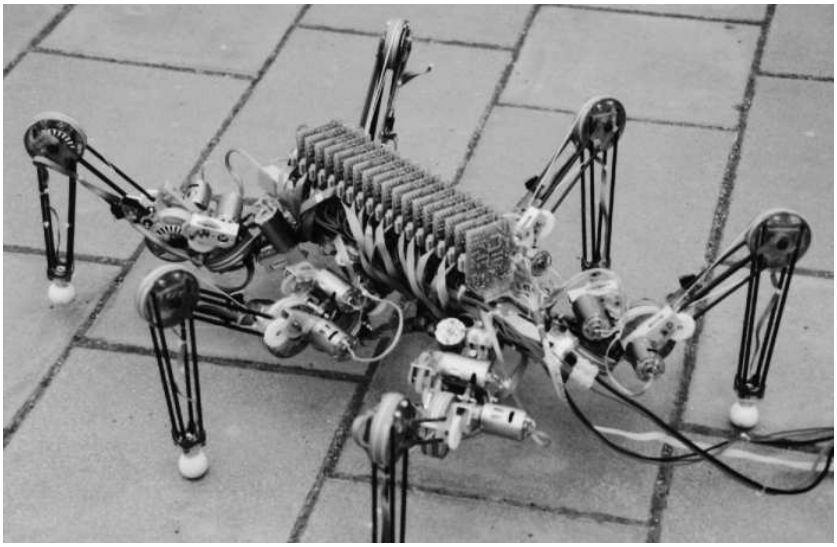


Abbildung 5.1: LAURON I

Beim Nachfolger LAURON II (Kepplin und Berns, 1999) wurden in erster Linie Schwachstellen in der Mechanik behoben, außerdem wurde die Maschine mit einer Reihe von verschiedenen Sensoren bestückt. Aus der kontinuierlichen Weiterentwicklung dieses Modells entstand dann die aktuelle Version LAURON III (Gaßmann et al.,

2001). Wenn in der vorliegenden Arbeit einfach von LAURON die Rede sein wird, wird immer die aktuelle Version III gemeint sein.

Abbildung 5.2: Einsatz von LAURON III in unstrukturiertem Gelände

5.1.1 Technische Daten

LAURON besteht aus einem Hauptkörper, sechs (bis auf Symmetrie) gleichförmigen Beinen und einem bewegbaren Kopf. Wenn man den Abstand zwischen den Beinen betrachtet, hat der Roboter eine Länge und Breite von jeweils ca. 70 cm. Sein Gewicht liegt bei ungefähr 18 kg, maximal können weitere 10 kg zugeladen werden. Jedes der sechs Beine hat drei Gelenke mit Maximalwinkelauslenkungen von 80°, 100° und 140°, die mit Gleichstrom-Motoren betrieben werden; für den Körper existieren also 18 Freiheitsgrade. Die technische Maximalgeschwindigkeit für LAURON wird mit $0{,}5\,\mathrm{ms}^{-1}$ angegeben, liegt aber durch Beschränkungen in der Steuerung im praktischen Einsatz ungefähr bei $0{,}3\,\mathrm{ms}^{-1}$ (Gaßmann, 2000).

Die Stromversorgung erfolgt entweder extern oder durch 20 Nickel-Cadmium-Batterien (24 V, 2,5 Ah) im Hauptkörper der Maschine. Im letzteren Fall kann LAURON ca. 45 Minuten autonom betrieben werden, seine durchschnittliche Leistungsaufnahme liegt bei 80 Watt.

Jedes Bein und der Kopf werden mit einem Siemens 80C167-Mikrocontroller gesteuert, die per CAN-Bus untereinander und mit einem an Bord der Maschine befindlichen Klein-PC verbunden sind. Der PC ist derzeit ein PC/104 mit einem Pentium-II-400-Prozessor und läuft unter dem Echtzeitbetriebssystem Real-Time-Linux. Mittels Wireless LAN kann der PC/104 via TCP/IP mit einem externen (Desktop-)Rechner kommunizieren, auf dem ausgelagerte Aufgaben wie Visualisierung und Bedienoberflächen ablaufen können.

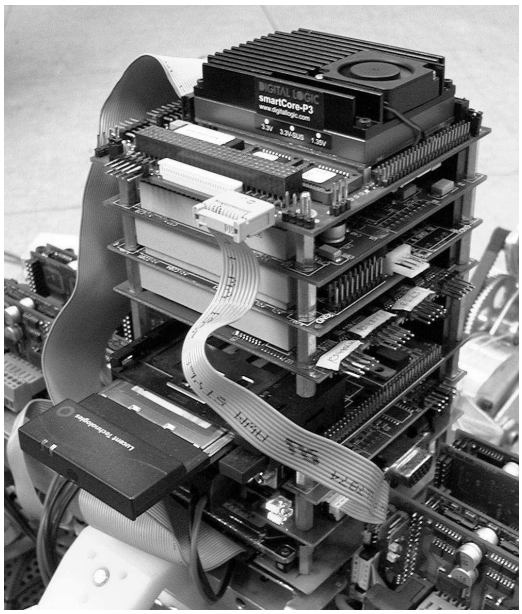


Abbildung 5.3: Auf LAURON montierter PC/104

5.1.2 Sensorik

Da eine Laufmaschine eine Vielzahl an Freiheitsgraden aufzuweisen hat, ist LAURON mit einer Reihe von Sensoren bestückt. Im einzelnen sind dies:

Positionssensoren Diese sind durch absolut messende optische Winkelgeber realisiert, die an den Gelenken angebracht sind und deren Winkel bestimmen. Bei der Überstreichung eines Winkels auftretende Impulse werden per Hardware gezählt und mittels einer Linearfunktion in Winkelwerte umgerechnet.

Neigungssensoren Diese messen die absolute Neigung des Maschinenkörpers durch Projektion der Erdbeschleunigung auf zwei orthogonal aufeinander stehende Achsen. Zur Verminderung des Rauschens ist diesen Sensoren ein Durchschnitts- und ein Median-Filter nachgeschaltet.

Fußkraftsensoren Die an den Füßen des Roboters angebrachten Sensoren messen die Signale, die von Stahlbauteilen mit definierten Biegestellen und darauf aufgebackenen Dehnungsstreifen ermittelt werden. Diese werden im Anschluss durch einen Differenzverstärker geleitet.

Motorstromsensoren Der Stromverbrauch der einzelnen Motoren wird durch Messung des Spannungsabfalls an einem Widerstand ermittelt und danach einer Spannungsverstärkung unterzogen.

Distanzsensor Zur Zeit ist nur ein einziger Distanzsensor auf LAURON installiert. Es handelt sich um einen eindimensionalen Laser-Abstandssensor. Seine technischen Daten finden sich in Anhang A.

Die A/D-Wandlung ermittelter Sensorwerte von Fußkraft- und Motorstromsensoren wird jeweils von den C167-Mikrocontrollern übernommen. Für eine detailliertere Aufstellung der verwendeten Sensorik und ihrer Auswertung sei auf Gaßmann (2000) verwiesen.

5.1.3 Die Steuerungsarchitektur MCA2

Alle am IDS entwickelten Roboter werden mittlerweile mit der im eigenen Haus entwickelten Steuerungsarchitektur MCA2 (Modular Controller Architecture) betrieben (Scholl et al., 2001). MCA2 ist in C++ implementiert und steht unter der *GNU Public Licence* (GPL).

MCA2 stellt eine mehrschichtige Architektur zur Verfügung. In ihr gibt es zwei Datenflüsse: Ein Sensordatenfluss leitet Sensordaten von der Maschine bis zur Benutzerschnittstelle, ein Kontrolldatenfluss transportiert Steuerungsdaten auf entgegengesetztem Weg nach unten. Dabei durchlaufen beide Flüsse eine Vielzahl von *Modulen*. Dies sind die kleinsten Einheiten des Systems, es handelt sich dabei um wiederverwendbare Code-Stücke mit fester Schnittstelle, die alle von einer gemeinsamen Basisklasse erben und die Datenflüsse modifizieren. Module können zu *Modulgruppen* zusammengefasst werden, die sich nach außen hin als ein normales Modul darstellen. Jedes Modul (und jede Modulgruppe) verfügt über jeweils zwei Ein- und Ausgänge für Sensor- und Kontrolldaten. Ein Beispiel für eine Modulgruppe wäre die Hintereinanderschaltung zweier Module, in denen ein Durchschnitts- und ein Medianfilter implementiert sind. Module und Modulgruppen können frei miteinander kombiniert werden, indem man sie durch *Kanten* miteinander verbindet. Es entsteht so ein gerichteter Graph für jeden Datenstrom.

Die Schichten von MCA2 sind hierarchisch aufgebaut: Auf der untersten Ebene liegen die Hardware-nahen Programmteile, die auf den Mikrocontrollern laufen. Diese sind per CAN-Bus mit einem PC und der auf ihm laufenden darüber liegenden Schicht verbunden, die unter dem Echtzeitbetriebssystem RT-Linux (Real-Time-Linux) läuft und alle echtzeitkritischen Bereiche steuert. Die Kommunikation mit der höchsten Schicht erfolgt per RT-FIFOs. MCA2 bietet dabei volle Netzwerktransparenz, es bleibt dem Benutzer überlassen, welche Programmteile er in welcher Schicht unterbringen möchte.

Via Wave-LAN und TCP/IP kann der die Maschine steuernde PC mit einem zweiten Rechner verbunden werden, auf dem Visualisierungswerkzeuge und Benutzerschnittstellen laufen können. MCA2 bietet mit MCAadmin ein mächtiges Instrument, um die Modul-Graphen zu visualisieren und Steuerungs-Parameter direkt anpassen zu können. MCAGUI ist eine auf dem Toolkit *Qt* basierende Benutzeroberfläche, die es erlaubt, alle Ein- und Ausgänge am höchsten Punkt der Gesamtsteuerung sowohl mit Kontrollelementen wie Schiebereglern oder Knöpfen (für die Kontrolleingänge) als auch mit Überwachungs-Widgets wie Anzeigeelemente, Graphen oder auch 3D-Simulationen für die Sensorausgänge zu verbinden.

+

5.2 Eintragen der Sensordaten

Dieser Abschnitt erläutert den Weg von der konkreten Sensormessung zur Modifikation einer Gitterzelle.

5.2.1 Ermittlung der Gitterzelle

Um Informationen über die Umgebung zu erlangen, kann man bei Laufmaschinen wie der in dieser Arbeit verwendeten auf mehrere Möglichkeiten zurückgreifen:

Abstandssensorik Hierunter fallen Infrarot-Sensoren, Laser-Scanner oder auch Kamera-Vision-Systeme. Sie liefern den Abstandswert eines Hindernisses zum Sensor zurück.

Fußkraftsensoren Diese messen die Kraft, die auf ein Bein wirkt. Insbesondere kann man daraus ableiten, ob ein Bein auf dem Boden steht oder nicht.

Lagesensoren Unter dieser Bezeichnung sind alle Sensoren zusammengefasst, die Aussagen über die Lage der Maschine geben, also Neigungssensoren oder Positionssensoren, die bspw. die Gelenkwinkel messen. Hieraus lässt sich insbesondere die Position der Füße ablesen.

Odometrie Durch Messung und Aufzeichnung der zurückliegenden Bewegungen kann die Positionsveränderung des Roboters gemessen werden.

Ein Überblick über die Sensorik der für diese Arbeit verwendeten Laufmaschine Lauron findet sich in Abschnitt 5.1.2.

Um einen Punkt in das Weltmodell eintragen zu können, muss er zunächst aus dem lokalen Koordinatensystem (LKS) des Roboters in ein globales Koordinatensystem (GKS) transformiert werden. Das LKS hat seinen Ursprung in der Mitte des Roboterkörpers, x- (vorwärts) und y-Achse (seitwärts) sind die Bewegungsachsen, die z-Achse die Höhe (Abbildung 5.4).

Aus der Odometrie erhält man einen Translationsvektor t für die Verschiebung zwischen den Ursprüngen der beiden Koordinatensysteme. Für die notwendigen Drehungen lassen sich zwei Drehwinkel (Rotation_x und Rotation_y) aus den Neigungssensoren ermitteln, den dritten Winkel (Rotation_z) erhält man wiederum aus der Odometrie. Damit ergibt sich ein Punkt $w^{(i)}$ im GKS aus einem Punkt $v^{(i)}$ im LKS aus der Hintereinanderschaltung dreier Drehungen und einer Translation (der besseren Übersicht wegen stehen hier abgekürzt α, β und γ für die Drehwinkel, $\mathrm{c}\alpha$ für $\cos\alpha$ und $\mathrm{s}\alpha$ für $\sin\alpha$):

$$w = \mathrm{rot}_\alpha \mathrm{rot}_\beta \mathrm{rot}_\gamma (v^{(i)} - t) \tag{5.1}$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}\alpha & -\mathrm{s}\alpha \\ 0 & \mathrm{s}\alpha & \mathrm{c}\alpha \end{pmatrix} \begin{pmatrix} \mathrm{c}\beta & 0 & -\mathrm{s}\beta \\ 0 & 1 & 0 \\ \mathrm{s}\beta & 0 & \mathrm{c}\beta \end{pmatrix} \begin{pmatrix} \mathrm{c}\gamma & -\mathrm{s}\gamma & 0 \\ \mathrm{s}\gamma & \mathrm{c}\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x^{(i)} - t_x \\ v_y^{(i)} - t_y \\ v_z^{(i)} - t_z \end{pmatrix} \tag{5.2}$$

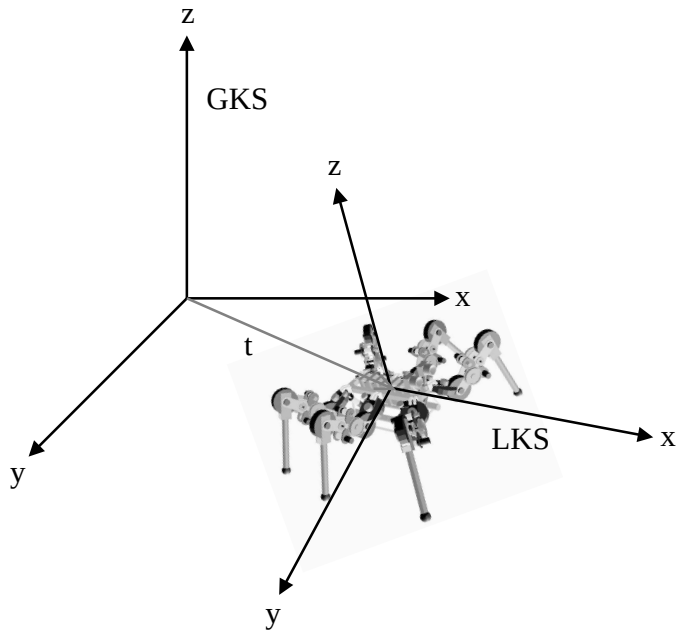


Abbildung 5.4: Lokales und globales Koordinatensystem (LKS und GKS). Eingezeichnet ist der Translationsvektor t, der die Nullpunkte ineinander überführt.

Für den so gewonnenen globalen Positionsvektor w lassen sich nun leicht die dazugehörigen Gitterzellenkoordinaten w^{grid} ermitteln. Deren x-Komponente w_x^{grid} (und analog w_y^{grid} und w_z^{grid}) berechnet sich aus

$$w_x^{grid} = \frac{(w_x - \frac{1}{2}(w_x^{min} + w_y^{max}))}{\Delta w_x} \tag{5.3}$$

wobei w_x^{min} und w_x^{max} den Wertebereich des Gitters und Δw_x die Ausdehnung der Gitterzelle auf der jeweiligen Achse bezeichnet.

Eingetragen in das Weltmodell werden nun die Daten aus zwei unterschiedlichen Quellen: die Entfernungswerte der Abstandssensoren und die „ertasteten“ Stellen, an denen die Füße der Maschine den Boden berühren. Diese Positionen sind direkt aus der inversen Kinematik der Steuerungssoftware als LKS-Koordinaten ermittelbar und können nach der Transformation mit Hilfe von Formel (5.1) ins GKS sofort im Belegtheitsgitter eingetragen werden. Dies wird genau dann getan, wenn sich aus einem Fußkraftsensor ein fester Stand ablesen lässt. Das Erfassen leichter Hindernisberührungen erwies sich als zu störanfällig und führte zu einer Vielzahl von Fehleintragungen, weshalb darauf ganz verzichtet wurde.

Abstandssensoren liefern Entfernungen eines aufgenommenen Hindernisses zurück. Daraus muss bei nicht fest angebrachten Sensoren (wie bspw. den auf dem Roboterkopf montierten) zunächst aus der direkten Kinematik eine Position im LKS ermittelt werden. Für fix angebrachte Sensoren genügt dazu eine Translation und ggf. Rotationen um konstante Winkel.

5.2.2 Belegte und freie Gebiete

Nicht nur die durch die Sensorik gefundenen belegten Gebiete sind Informationen über die Umwelt. Aus den Messungen geht hervor, an welchen Stellen Hindernisse bzw. Boden wahrgenommen wird – aber auch, wo keine Hindernisse sind. So kann man davon ausgehen, dass sich dort, wo sich der Roboter befindet, keine belegten Gebiete befinden. Ebenso kann man mit Sicherheit darauf schließen, dass die Verbindungsstrecke zwischen ermitteltem Hindernis und dem Sensor frei ist. All dies sollte sinnvollerweise in das Weltmodell eingetragen werden.

Wie in Abschnitt 4.2 beschrieben, stehen auf der Gitterebene Methoden zur Verfügung, um grafische Primitive wie Linien einfügen zu können. Dies kann hier genutzt werden, um auf effiziente Weise sämtliche Zellen in der Speicherschicht, die zwischen der zur Sensorposition gehörenden Zelle und der des Hindernisses mit der Information „frei“ zu versehen. Dazu werden in der Gitterebene alle Zellen ermittelt, die die Verbindungslinie schneiden und entsprechende Koordinaten der Umgebungsebene zur Verfügung gestellt. Dort werden diese Punkte genau so behandelt, als wären sie gerade von einem Sensor gemessen worden: die Aktualisierungsparameter k_{in} und k_{cred} werden ermittelt und der jeweilige Zellinhalt entsprechend modifiziert (vgl. Abschnitt 4.6.3). Ähnlich wird bei den Fußaufsetzpunkten vorgegangen: Hier werden einige Zentimeter oberhalb des gemessenen Punktes als frei klassifiziert, da sich dort ja das Bein befindet. Dies mag man als relativ ungenau und wenig sauber empfinden, es erfüllt aber seinen wichtigsten Zweck: Fehlmessungen unmittelbar oberhalb des realen Bodens werden so korrigiert. Der Aufwand, die gesamte Ausdehnung des Roboters einschließlich aller Beinauslenkungen zu errechnen, lohnt im Ergebnis nicht entsprechend. Abbildung 5.5 stellt das Einfügen dieser freien Zellen dar.

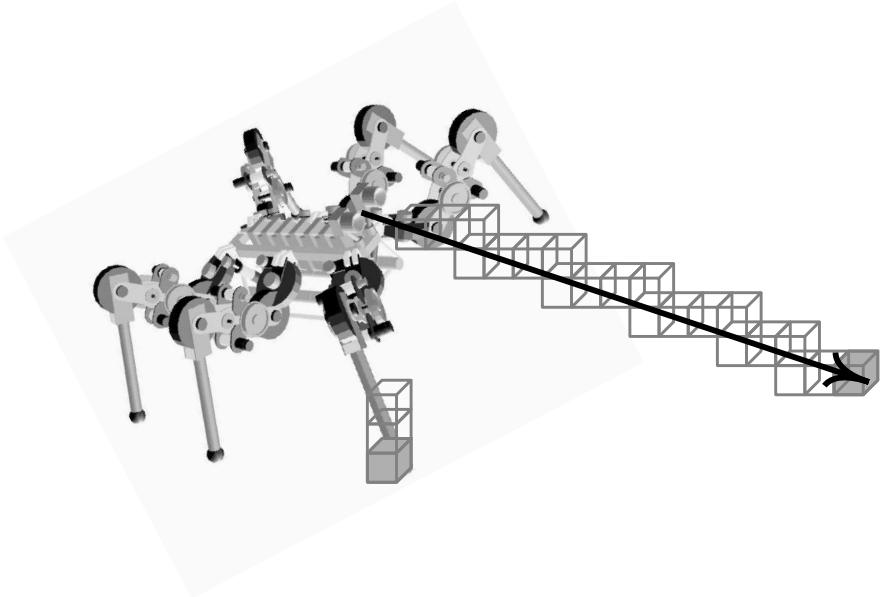


Abbildung 5.5: Einfügen von freien Zellen am Beispiel eines Abstands- und eines Fußsensors: Die grau unterlegten Zellen werden direkt gemessen und als belegt klassifiziert, die transparenten Zellen werden als frei betrachtet und entsprechend eingetragen.

Eine Eingangsgröße für die Ermittlung der Aktualisierungsparameter ist die Sensorqualität, die von außen für jeden verwendeten Sensor festgelegt wird (Abschnitt 4.6.2). Für alle wie oben beschrieben als frei klassifizierten Zellen wird ein so genannter *virtueller Sensor* eingeführt, der mit einer Sensorqualität versehen wird und dem alle so gewonnenen Daten zugeschlagen werden. Das gewählte Qualitätsmaß sollte auf Grund der relativ grob ermittelten Linie nicht allzu hoch gewählt werden.

Für LAURON bedeutet dies, dass es derzeit acht verschiedene Eingangsquellen gibt: sechs Fußsensoren, der bewegbare Laser-Sensor am Kopf und der virtuelle Sensor, der die als frei gemessenen Gebiete aller anderen Sensoren umfasst.

5.3 Auswertung der gespeicherten Umweltdaten

An dieser Stelle soll nun aufgezeigt werden, wie aus dem vorgestellten Umweltmodell Wissen extrahiert wird und in welcher Weise es zur Steuerung der Maschine angewendet werden kann.

5.3.1 Laufverhalten von Lauron

Es ist zunächst nötig, kurz auf das Laufverhalten der Maschine einzugehen, mit der bei diesem Projekt gearbeitet wurde. Die Ausführungen sind bewusst kurz gehalten und auf die Auswertung der Umweltinformation relevante Einzelheiten beschränkt. Details zur Steuerung finden sich in Gaßmann (2000).

Die Bewegung eines einzelnen Beins der Laufmaschine beschreibt einen zweiteiligen Zyklus aus *Stemmphase* und *Schwingphase*. In der Stemmphase steht das Bein auf dem Boden und stützt den Körper des Roboters. Dabei führt es für den Fall des Vorwärtsgehens eine Bewegung nach hinten aus, um den Rumpf der Maschine so nach vorne zu schieben. In der folgenden Schwingphase wird das Bein angehoben und schwingt in der Luft nach vorne zum nächsten Aufsetzpunkt, wo erneut die Stemmphase eingeleitet wird. Für die Ausprägung dieses Bewegungsablaufs sind für unsere Zwecke in erster Linie vier Parameter von Interesse: die Schrittlänge vorwärts (Sv), die Schrittlänge seitwärts (Ss), die Schritthöhe (Sh) und der Drehwinkel (Sd).

LAURON beherrscht verschiedene Gangarten, die durch einen Belastungsfaktor $\beta \in [0 \ldots 1)$ charakterisiert werden, der das Verhältnis der Dauer der Stemmphasen zu den Schwingphasen beschreibt. In den Spezialfällen $\beta = \frac{1}{2}$ (Tripod), $\beta = \frac{2}{3}$ (Tetrapod) und $\beta = \frac{5}{6}$ (Pentapod) ist dies der Anteil der Beine, die sich gerade in der Stemmphase befinden. Der Tripod ist dabei die schnellste Art der Fortbewegung. Die anderen Gangarten werden dann sinnvoll, wenn die Bewältigung des Untergrunds schwieriger fällt: Je mehr Beine den Körper stützen, desto sicherer steht der Roboter.

Bislang gestaltet sich die Steuerung von Lauron grob dargestellt derart, dass die Gangart sowie die Parameter Sv, Ss und Sd von außen vorgegeben werden und so das Laufverhalten von LAURON bestimmen. Andere Werte des Systems wie die Schritthöhe Sh oder die Körperhöhe werden von der Steuerungssoftware automatisch den Gegebenheiten angepasst. Wenn der Roboter eine Kollision eines Beines feststellt, wird bspw. versucht, durch eine größere Schritthöhe das gefundene Hindernis zu überwinden. Auch wird stets durch tastende Bewegungen eine gute Fußaufsetzposition gesucht. Auch ein

sehr kleiner Ansatz von Umweltmodellierung findet sich in der Steuerung bereits: Wenn durch den am Rumpf montierten Infrarot-Sensor ein Hindernis in der Zentralkörpermitte festgestellt wird, wird der Körper entsprechend lange angehoben. Die notwendige Anzahl von Zyklen, in denen so reagiert wird, errechnet sich aus dem Quotienten aus Körperlänge und Geschwindigkeit der Maschine.

5.3.2 Finden geeigneter Fußaufsetzpunkte

Ziel ist es nun, das Verhalten der Maschine bzgl. ihres Ganges mittels der durch das Weltmodell gewonnenen Information zu modifizieren. Dabei soll explizit eine Schwäche der bisherigen Steuerung angegangen werden: das wenig zielgerichtete Vorgehen in der Bewegung. Um ein Beispiel zu nennen: Beim Überqueren eines schmalen Grabens hat die Maschine derzeit noch Probleme, weil sie weder genau genug an den Rand der Vertiefung herantreten noch zielgerichtet mit großen Schritten das Hindernis überwinden kann – für beides fehlt ihr das Wissen, wo dieser Rand und wie breit der Graben ist. Solche Informationen sollen nun aus dem Weltmodell gewonnen und in konkrete Vorgaben für Fußaufsetzpunkte umgewandelt werden (siehe Abbildung 5.6).

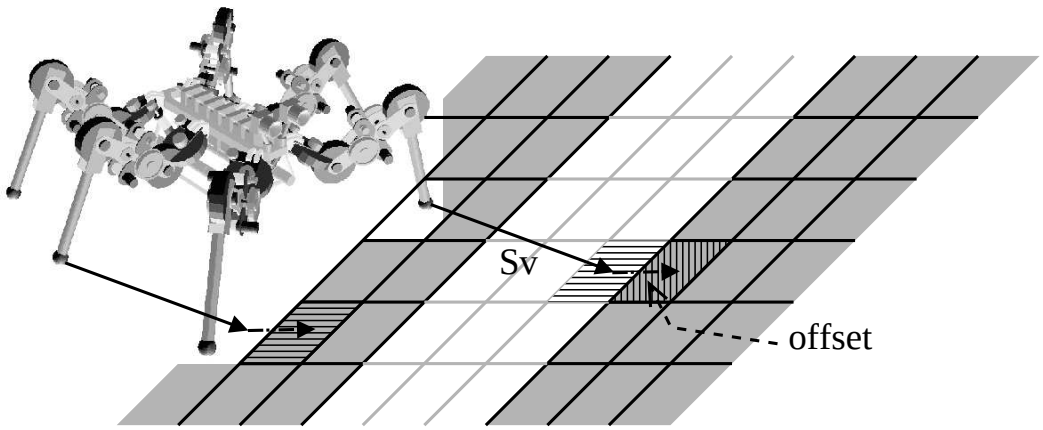


Abbildung 5.6: Finden eines geeigneten Fußaufsetzpunktes: Schematisch dargestellt ist ein Graben (weiß), der voraussichtliche neue Fußaufsetzpunkt für das linke Vorderbein liegt in selbigem (horizontal schraffiert). Aus der Umweltinformation erweist sich offensichtlich die rechts daneben liegende Zelle als geeignetere Aufsetzstelle. Die daraus resultierende Verschiebung ist der Offset, der zur voraussichtlichen Aufsetzstelle des Mittelbeins addiert wird: Für dieses Bein wird die Suche nach einem geeigneten Punkt an dieser Stelle beginnen (ebenfalls horizontal schraffiert).

Algorithmus 1 zeigt das Verfahren auf, das vor dem Beginn der Schwingphasen der Beine jeweils angewandt wird. Zunächst wird bestimmt, welche Beine in die Schwingphase eintreten werden. Nun wird, mit dem Vorderbein beginnend, zunächst der Punkt $P_v = (x_v, y_v)$ ermittelt, an dem unter Berücksichtigung des gegebenen Parametersatzes der Fuß wahrscheinlich aufsetzen wird, wenn es nicht zu Komplikationen kommt. Dabei handelt es sich um einen Punkt auf der xy-Ebene, der sich durch

$$P_v = (x_v, y_v) = (x_a, y_a) + (Ss_v, Sv_v) \tag{5.4}$$

Algorithmus 1 Extrahieren geeigneter Fußaufsetzpunkte

Require: alle Schwingphasen beendet
offset = $(x_o, y_o) := (0, 0)$
for all Beine, die in Schwingphase eintreten **do**
ermittle voraussichtlichen Fußaufsetzpunkt $P_v = (x_v, y_v)$
$P_{v'} = P_v +$ offset
generiere lokale Umgebungskarte $K(P_{v'})$ um $P_{v'}$
ermittle Karten-Vertrauenswürdigkeit $c(K(P_{v'}))$
if $c(K(P_{v'})) < c_{min}$ **then** {Vertrauenswürdigkeit zu niedrig}
führe Nachmessung um P_v durch
generiere lokale Umgebungskarte $K(P_{v'})$ um $P_{v'}$
end if
finde geeignetsten Fußaufsetzpunkt $P_p = (x_p, y_p)$ aus $K(P_{v'})$
ermittle Höheninformation $h(P_p)$ aus Karte $K(P_{v'})$
ermittle Parameter, um das Bein zum Punkt $(x_p, y_p, h(x_p, y_p))$ zu steuern
offset $:= P_p - P_v$
end for
übergebe gefundene Parameter an Steuerung

aus dem aktuellen Fußpunkt $P_a = (x_a, y_a)$ berechnet, wobei Ss_v und Sv_v die geplanten, voraussichtlichen Schrittlängen sind und (x_a, y_a) die aktuelle Position des Fußes ist.

Um diesen Punkt wird nun eine der bereits in Abschnitt 4.1 erwähnten kleinen, lokalen Fußumgebungskarten erstellt (siehe dazu Abschnitt 5.3.2.1). Aus ihr wird ermittelt, wie sicher die Informationen bzgl. dieses Ausschnitts der Umgebung sind, im Zweifelsfall wird veranlasst, dass an dieser Stelle weitere Sensordaten erfasst werden (Abschnitt 5.3.3) und danach die Fußumgebungskarte neu erstellt wird. Nun wird ausgehend vom geplanten Fußaufsetzpunkt der unter Einbeziehung der Umweltinformation geeignetste Ort P_p bestimmt, auf den das Bein aufsetzen soll (Abschnitt 5.3.2.2). Dazu wird die Höheninformation $h(P_p)$ aus der Karte bestimmt, um aus der Differenz $h(P_p) - h(P_a)$ eine geeignete neue Schritthöhe Sh_p bestimmen zu können. Die angepassten Schrittlängen Ss_v und Sv_v berechnen sich analog zu Gleichung (5.4).

Da die Bewegungen der Beine untereinander koordiniert werden müssen, muss dafür Sorge getragen werden, dass die gewählten Fußaufsetzpunkte für gemeinsam bewegte Beine nicht dazu führen, dass der Roboter instabil wird. Insbesondere dürfen sie nicht allzu stark in verschiedene Richtungen weisen. Um dies zu verhindern, wird die Bewegung des Vorderbeins als Referenz für die Bewegung der Maschine genommen. Die durch die Weltmodellauswertung erfolgende Abweichung von der geplanten Route, $P_p - P_v$, wird als *Offset* genommen und zu den geplanten Fußaufsetzpunkten des folgenden Beins addiert, um sicherzustellen, dass es der Richtung der Vorderbeine folgt. Aus dem gleichen Grund werden die Fußumgebungskarten der Mittel- und Hinterbeine auch etwas kleiner gewählt – der ermittelte Fußaufsetzpunkt muss mit der neu errechneten Verschiebung noch erreichbar sein. Für die Hinterbeine addieren sich die Offsets der ersten beiden Beine.

5.3.2.1 Generierung lokaler Fußumgebungskarten

Die lokalen Fußumgebungskarten werden aus einer Projektion der 3D-Information der Urkarte ins Zweidimensionale gewonnen. Sie haben eine Ausdehnung von n_L Zellen auf der xy-Ebene um den voraussichtlichen Fußaufsetzpunkt P_v. Für jede Rasterzelle innerhalb dieses Bereiches wird nun von oben kommend innerhalb eines Höhentoleranzbereiches die höchste Zelle C gesucht, deren Belegungswahrscheinlichkeit $occ(C)$ über einem Schwellwert occ_{min} liegt. occ_{min} ist dabei die Grenze, ab der eine Zelle als ausreichend belegt angesehen wird, um nicht als Fehlmessung klassifiziert zu werden. Deren z-Koordinate wird als Höheninformation mit dem entsprechenden Zuverlässigkeitswert in die Umgebungskarte eingetragen. Wenn bereits die oberste Zelle belegt ist oder keine Zelle mit ausreichender Belegung gefunden werden kann, wird diese Position als unbegehbar markiert, da sie keinen geeigneten Fußaufsetzpunkt bietet. Algorithmus 3 beschreibt das Generieren der lokalen 2D-Fußumgebungskarte L aus einer 3D-Urkarte U.

Algorithmus 2 Generieren der lokalen Fußumgebungskarte L aus Urkarte U

for all Zellen $L(x, y)$ im Bereich $n_L \times n_L$ um P_v **do**
  wähle Zelle $U(x, y, z)$ mit maximaler z-Koordinate
  z_h = Höhe der Gitterzelle
  while $occ(U(x, y, z)) < occ_{min}$ **and** $z \geq$ minimale z-Koordinate **do**
    {Finde höchste ausreichend belegte Zelle}
    $z = z - z_h$
  end while
  if z = maximale z-Koordinate **or** z = minimale z-Koordinate **then**
    {kein Aufsetzpunkt gefunden}
    Höhe$(L(x, y))$ = undefiniert
    $cred(L(x, y)) = 0$ {Vertraulichkeit auf 0}
  else
    Höhe$(L(x, y)) = z$
    $cred(L(x, y)) = cred(U(x, y, z))$ {Vertraulichkeit wie 3D-Zelle}
  end if
end for

Da die Granulierung der Zellen in z-Richtung im Zentimeter-Bereich liegt, ist die Wahrscheinlichkeit ungenauer Messeinträge durch die Diskretisierung der Umwelt relativ hoch. Fällt eine gemessene Hindernisposition ungefähr auf die Grenze zweier Zellen, genügen minimale Messfehler, um das Hindernis mal der einen, mal der anderen Zelle zuzuordnen. Um dies in der Erzeugung der lokalen Karte zu berücksichtigen, wird statt der Belegtheitswahrscheinlichkeit $occ(U(x, y, z))$ ein Wert $occ^*(U(x, y, z))$ betrachtet, der zur Hälfte die Zellen ober- und unterhalb mit einrechnet:

$$occ^*(U(x,y,z)) = \frac{1}{2}occ(U(x,y,z)) + \frac{1}{4}occ(U(x,y,z-z_h)) + \frac{1}{4}occ(U(x,y,z+z_h)) \quad (5.5)$$

Als Vertraulichkeitswert $\text{cred}^*(L(x,y))$ für die ermittelte Position wird dementsprechend

$$cred^*(L(x,y)) = \frac{1}{2}cred(U(x,y,z)) + \frac{1}{4}cred(U(x,y,z-z_h)) + \frac{1}{4}cred(U(x,y,z+z_h)) \tag{5.6}$$

eingetragen.

5.3.2.2 Ermittlung des besten Aufsetzpunktes

Um nun einen geeigneten Fußaufsetzpunkt zu bestimmen, muss man zunächst klären, was ein *guter* Fußaufsetzpunkt ist. Dies ist dann der Fall, wenn die gefundene Zielposition

1. begehbar ist
2. nahe an dem voraussichtlichen Fußaufsetzpunkt P_v liegt
3. mit hinreichender Vertraulichkeit bekannt ist.

Da für nicht begehbare Punkte die Vertraulichkeit bei der Kartengenerierung auf 0 gesetzt wurde, muss man nun also nur zwischen Vertraulichkeit und Nähe zum Vorgabepunkt abwägen. Weiter entfernte Punkte benötigen eine höhere Vertraulichkeit, um genauso bewertet zu werden wie nahe. Dazu wird mit zunehmender Entfernung von dem voraussichtlichen Aufsetzpunkt die Vertraulichkeit vermindert und in diesen Werten dann das Maximum gesucht:

$$P_p = (x_p, y_p) = \underset{x,y}{\text{argmax}}(cred(L(x,y)) \cdot (1 - k_L(x,y)\|(x_v,y_v) - (x,y)\|)) \tag{5.7}$$

Dabei bezeichnet $\|\ldots\|$ die euklidische Norm. Der Gewichtungsfaktor $k_L(x,y)$ bestimmt in dieser Formel, wie stark Vorgabennähe und große Vertraulichkeit gegeneinander gewichtet werden sollen. Ein sinnvoller Wert wäre bspw.

$$k_L(x,y) = \left(\frac{\|(x_v,y_v) - (x,y)\|}{\max\limits_{x,y} \|(x_v,y_v) - (x,y)\|} + 1 \right)^{-1} \tag{5.8}$$

Diese Wahl bewirkt, dass Punkte an den Rändern der Karte eine doppelt so hohe Vertraulichkeit aufweisen müssen wie der Mittelpunkt, um berücksichtigt zu werden.

5.3.2.3 Parameterübergabe an die Steuerung

Um nun zu dem derart gewonnenen Zielpunkt P_p zu gelangen, müssen die das Laufverhalten steuernden Parameter Schrittlängen und -höhe (Sv, Ss und Sh) entsprechend angepasst werden. Die neuen Parameter Sv_p, Ss_p und Sh_p ergeben sich auf einfache Weise aus

$$Sv_p = Sv_v + (x_p - x_v) \tag{5.9}$$
$$Ss_p = Ss_v + (y_p - y_v) \tag{5.10}$$
$$Sh_p = Sh_v + (h(x_p,y_p) - h(x_a,y_a)) \tag{5.11}$$

$h(x, y)$ ist hierbei die Höheninformation zu (x, y). Zu beachten ist hierbei, dass diese Parameter für jedes Bein separat berechnet werden. Die Vorgabeparameter Sv, Ss und Sh gelten dagegen für alle Beine. Eine Einschränkung muss gemacht werden: Ist die Differenz $Sv_p - Sv_v$ (bzw. $Ss_p - Ss_v$) kleiner als die jeweilige Ausdehnung der Gitterzelle des Weltmodells, wird die ermittelte Abweichung verworfen, da sie auch nur aus der verlustbehafteten Granulierung des Weltmodells resultieren könnte und auch keinen großen Einfluss auf das Bewegungsverhalten hätte. Dadurch wird vermieden, dass die Maschine von ihrem Kurs abweicht, ohne dass wirklich eine Notwendigkeit dazu besteht.

Diese modifizierten Parameter werden an die Beinsteuerung übergeben und so die Laufbewegung entsprechend der aus dem Weltmodell gewonnenen Information an die Umwelt angepasst, ohne dass in die eigentliche Steuerung des Laufvorgangs eingegriffen werden muss. Dies ist natürlich eine sehr grobe Modifikation und noch nicht optimal; eine andere Möglichkeit lässt die Steuerungssoftware von LAURON allerdings derzeit noch nicht zu. Dieses Verfahren zu verfeinern wird eine Aufgabe für die Zukunft sein. Dennoch reicht dieser Ansatz vollkommen aus, um das Laufverhalten des Roboters nachhaltig zu verbessern (vgl. dazu die durchgeführten Experimente in Kapitel 6).

5.3.3 Lokales Nachmessen unsicherer Gebiete

Wie in Algorithmus 1 auf Seite 60 aufgezeigt, ist es vorgesehen, dass durch Nachmessen zusätzliche Sensorinformation gezielt akquiriert werden kann. Dies erfüllt die in Abschnitt 3.1.6 formulierte Anforderung nach „gezielter Nachmessungsmöglichkeit“. Umgesetzt wird dies derart, dass der auf dem Kopf befindliche Sensor zielgerichtet die Umgebung der unsicheren Stelle erneut abscannt und dadurch eine Verbesserung der Qualität der Daten zu erwarten ist. Es versteht sich, dass dies derzeit nur für den Fall der beiden Vorderbeine realisiert werden kann – die voraussichtlichen Aufsetzpunkte von Mittel- und Hinterbeinen liegen außerhalb des Sichtbereiches des Kopfsensors.

Die Notwendigkeit der Nachmessung ist gegeben, wenn mindestens eins der folgenden beiden Kriterien erfüllt ist:

1. Die durchschnittliche Vertraulichkeit aller begehbaren Zellen der Karte liegt unter einem Schwellwert c_{min}.

2. Zu viele einzelne Zellen unterschreiten den Schwellwert c_{min}.

Dies ist jeweils bereits bei der Generierung der Karte ermittelbar. Die Zellen, die als nicht begehbar eingestuft wurden und deren Vertraulichkeit auf 0 gesetzt wurde, werden dabei nicht berücksichtigt.

Der Schwellwert c_{min} regelt dabei, wie risikobereit sich die Maschine verhält. In als unproblematisch anzusehendem Terrain kann er entsprechend niedriger gewählt werden als in schwierigem Gelände, denn die Nachmessung erfordert, dass das Bein stehen bleibt. Das simuliert in gewisser Weise das natürliche Verhalten eines Lebewesens: Der Roboter ist unsicher, stutzt, bleibt stehen, schaut sich die problematische Stelle noch einmal an und geht dann erst weiter. Die Nachmessung sollte dennoch ein Ausnahmefall für besonders schwierige Szenen bleiben, damit die kontinuierliche Laufbewegung nicht unnötig unterbrochen wird.

Durchführen einer Nachmessung bedeutet hier, dass die Strategie geändert wird, mit der der Sensor die Umgebung abfährt (Scan-Strategie). Eine Scan-Strategie ist durch einen Kopfreferenzpunkt, eine Zykluszeit und Auslenkungen für die drei Dimensionen realisiert. Der Kopf des Roboters wird so ausgerichtet, dass der Sensorstrahl auf den Kopfrefenrenzpunkt, einem Punkt im GKS, zeigt. Er wird dann so weiterbewegt, dass innerhalb der angegebenen Zykluszeit die jeweilige Auslenkung abgefahren wird. Auf diese Weise können verschiedene Ausprägungen von Scan-Linien realisiert werden (siehe Abbildung 5.7). Für die lokalen Nachmessungen wird man kurze Zykluszeiten und geringe Auslenkungen wählen, als Referenzpunkt bietet sich der geplante nächste Fußaufsetzpunkt an.

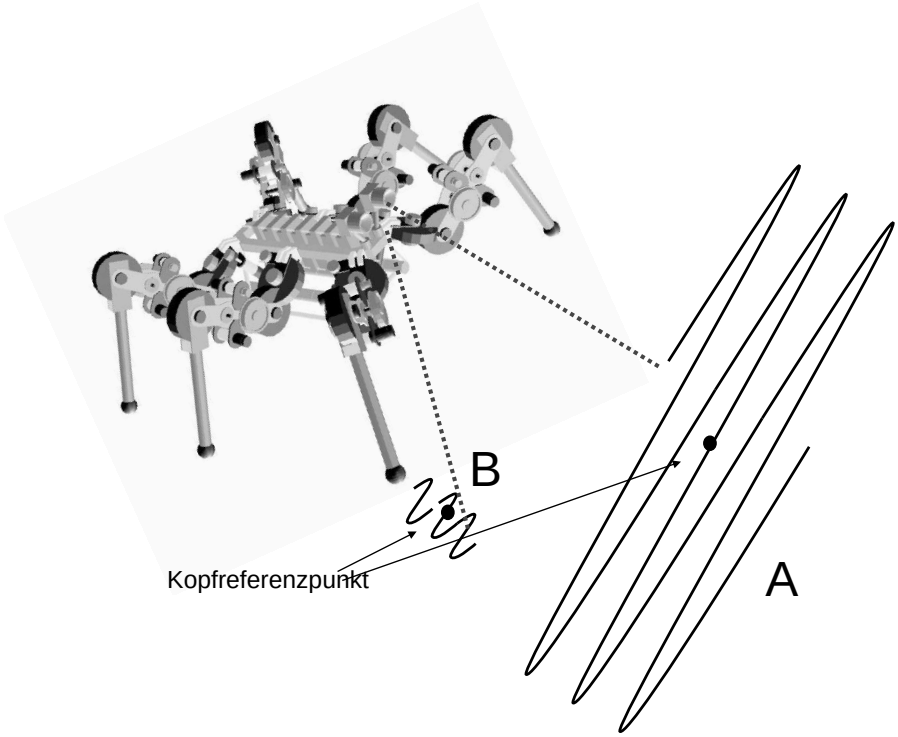


Abbildung 5.7: Nachmessen unsicherer Gebiete: Verschiedene Scan-Strategien des Kopfes. In Fall A fährt der Sensor ein breites Gebiet ab, um eine möglichst großflächige Umweltwahrnehmung zu erreichen, Fall B zeigt das lokale Nachmessen im Umfeld eines möglichen Fußaufsetzpunktes.

Abbildung 5.8 zeigt die Kommunikation zwischen Steuerung der Maschine und Weltmodell. Es handelt sich dabei um eine synchrone Kommunikation. Die Steuerung stellt fest, dass ein Bein in die Schwingphase eintreten sollte und fordert beim Weltmodell die entsprechenden Parameter an. Dieses erzeugt eine lokale Fußaufsetzkarte und ermittelt deren Vertraulichkeit. Ist diese zu niedrig, wird in der Steuerung eine Nachmessung initiiert, deren Beendigung dem Weltmodell mitgeteilt wird. Nun wird der Fußaufsetzpunkt errechnet. Die Übergabe der Parameter an die Steuerung ist gleichzeitig das Signal für diese, mit der Schwingphase zu beginnen.

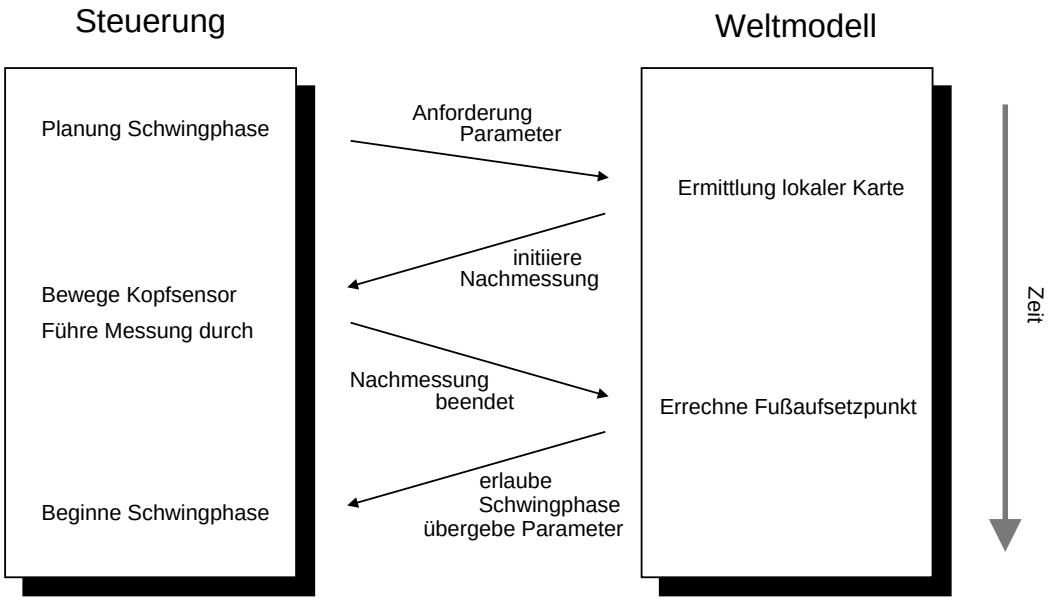


Abbildung 5.8: Synchrone Handshake-Kommunikation zwischen Steuerung und Weltmodell: Die Steuerung fordert Parameter an, um die Schwingtrajektorie für ein Bein zu berechnen. Erst wenn die Zuverlässigkeit der ermittelten lokalen Karte hoch genug ist, wird die Erlaubnis zur Schwingphase erteilt, ansonsten muss zuvor eine Nachmessung durchgeführt werden.

5.4 Erzeugung globaler Umgebungskarten

In Abschnitt 4.1 wird beschrieben, dass im Modell mehrere Karten in verschiedenen Abstraktionsstufen gehalten werden. Dabei handelt es sich bis auf die dreidimensionale Urkarte stets um zweidimensionale Karten. An dieser Stelle soll beschrieben werden, wie diese aussehen und wie sie aus den Daten der zu Grunde liegenden Karten erzeugt werden.

Die gröberen Karten haben dieselbe dreischichtige Struktur wie die 3D-Urkarte (vgl. Abschnitt 4.2). Das bedeutet insbesondere, dass für sie auch die gleichen Programmklassen verwendet werden können wie für die dreidimensionale Repräsentation und somit sofort auch deren gesamte Funktionalität wie bspw. Visualisierungsmechanismen zur Verfügung haben. Die unterste Ebene, die Speicherebene, unterscheidet sich nur marginal: Hier werden einerseits Quadtrees statt des Octrees verwendet, andererseits operieren die gröberen Karten auf unterschiedlichen Gitterzellen, da die Sensorinformation nicht direkt in diese eingetragen wird und sie somit die gesamte Metainformation der 3D-Gitterzelle nicht benötigen. In einer Zelle einer globalen 2D-Karte wird Folgendes abgelegt:

Höheninformation Zu jeder xy-Koordinate wird der entsprechend ermittelte absolute Höhenwert gespeichert. Sind mehrere Zellen an der Generierung der Zelle beteiligt, ergibt er sich aus dem arithmetischen Mittel der einzelnen Höheninformationen.

Zuverlässigkeit Diese ergibt sich direkt aus dem arithmetischen Mittel der Zuverlässigkeiten der entsprechenden Zellen der generierenden Karte.

Geländegängigkeit Zusätzlich wird eine Größe abgelegt, die beschreibt, wie einfach das Gelände, das durch die Zelle repräsentiert wird, für die Maschine zu begehen ist. Dies ist bspw. für die Routenplanung von Belang. Aus dem Höhenwert allein lässt sich nicht ablesen, ob es sich um ein homogenes oder zerklüftetes Gelände handelt. Eine Maß für die Geländegängigkeit einer zusammenhängenden Umgebung wäre bspw. die Summe aller paarweisen Differenzen der Höhenwerte. Perspektivisch könnten hier die in Abschnitt 3.1.4 angesprochenen Informationen über die Wegbarkeit ihren Platz finden.

Zur Erstellung einer gröberen Karte muss der gesamte Datenbestand der generierenden Karte durchlaufen werden. Ein Verfahren zur Erstellung einer globalen 2D-Karte aus der dreidimensionalen Urkarte findet sich in Algorithmus 3. Es basiert auf der Tiefensuche durch den Octree, die zweimal durchgeführt wird: Einmal, um zu jeder xy-Koordinate die Zelle mit der maximalen Höheninformation zu finden, ein weiteres mal, um die so gefundenen Zellen zu identifizieren und in die neue Karte einzutragen. Sei k nun die Anzahl der im Baum gespeicherten Zellen. Ein Octree hat im schlimmsten Fall (wenn er extrem spärlich besetzt ist) $O(k \log n)$ Knoten. Da bei einer Tiefensuche jeder Knoten maximal achtmal besucht wird, hat die Tiefensuche einen Aufwand von $O(k \log n)$. Jede gefundene Zelle muss nun noch mit einem Aufwand von $O(\log n)$ in den Quadtree der neuen 2D-Karte eingetragen werden, so dass sich für den Algorithmus ein Gesamtaufwand von $O(k \log n)$ ergibt.

Algorithmus 3 Generieren einer globalen 2D-Karte G aus Urkarte U

```
Lege 2D-Array an
Führe Tiefensuche durch:
for all Knoten des Baumes U mit Koordinaten (x, y, z) do
  if z-Koordinate der Zelle größer als in Wert im Array an (x, y) then
    Speichere z and Stelle (x, y) in 2D-Array
  end if
end for
Lege Quadtree G an
Führe erneut Tiefensuche durch:
for all Knoten des Baumes U mit Koordinaten (x, y, z) do
  if Koordinaten identisch mit den im 2D-Array gespeicherten then
    Speichere z für (x, y) im Quadtree G
  end if
end for
```

5.5 Einbindung des Weltmodells in MCA2

Die Steuerung von LAURON ist in mehrere Schichten unterteilt, die unterschiedliche Aufgaben erfüllen. In MCA2 sind diese Schichten durch Modulgruppen realisiert. Ab-

bildung 5.9 zeigt die Schichtung der einzelnen Steuerungsebenen in einer Darstellung des Visualisierungswerkzeugs *MCAadmin*. Im einzelnen sind die Schichten:

C167-Remote-Part Hier findet die Steuerung der C167-Mikrocontroller statt.

Signal Processing Group Alle Aufgaben der Signalverarbeitung wie zum Beispiel Filterung der Eingangsdaten sind Bestandteil dieser Gruppe.

Behaviour Group Diese Schicht steuert das Verhalten der Laufmaschine, berechnet die Beintrajektorien, kontrolliert die Gangarten, berechnet die Kinematiken etc.

Environment Model Group Die Umweltmodellierung ist die oberste Schicht. In ihr findet alles seinen Platz, was in dieser Arbeit beschrieben wird.

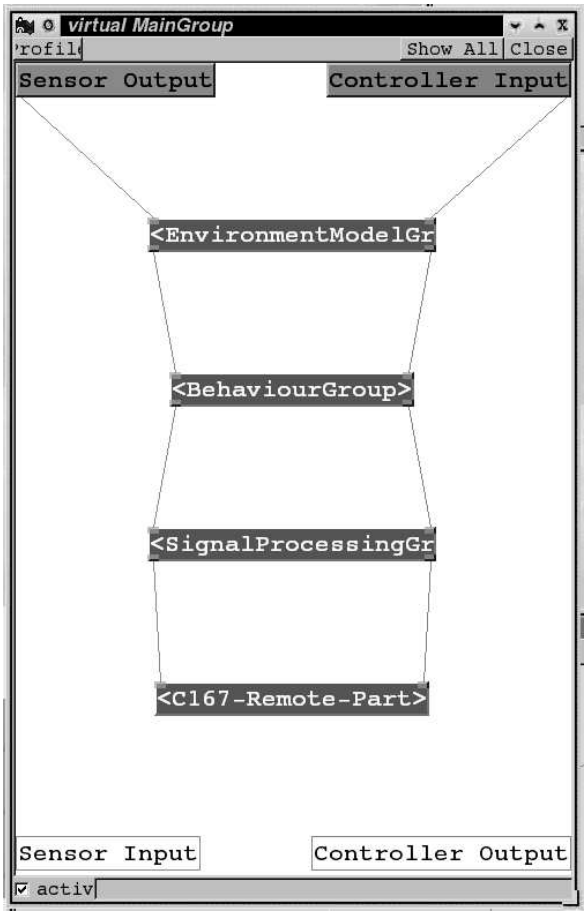


Abbildung 5.9: Realisierung der Steuerung unter Einschluss des Weltmodells in der Steuerungsarchitektur MCA2: Eine Schichtung eigenständiger Prozesse. Verbunden sind die einzelnen Ebenen durch Sensor- und Kontrolldatenflüsse (hier mittels Linien dargestellt).

Signalverarbeitung, Verhaltenssteuerung und Umweltmodell laufen dabei als eigenständige Prozesse nebeneinander. Wird der Weltmodell-Prozess nicht gestartet, funktioniert die Verhaltenssteuerung dennoch; sie besitzt dann ihre ursprüngliche Funktionalität.

Die Umweltmodell-Gruppe ist durch einen Kontroll- und Sensordatenfluss mit den benachbarten Ebenen verbunden. Die Schnittstelle zur obersten Ebene der Steuerung

besteht nur aus Kommunikationsleitungen zur Visualisierung der grafischen Benutzeroberfläche, interessant ist vielmehr die Schnittstelle zur darunter liegenden Verhaltenssteuerung:

Sensoreingänge Von der Verhaltenssteuerung werden geliefert: die Positionen der sechs Beine in Maschinenkoordinaten, Bodenkontaktstatus der Beine, Beginn und Ende des Distanzsensor-Strahls in Maschinenkoordinaten, absolute Position der Maschine relativ zur Position zum Beginn der Steuerung, die drei Neigungswinkel der Maschine sowie Anforderungen zum Eintritt in die Schwingphase und Meldungen bzgl. der Nachmessungen (vgl. Abbildung 5.8)

Kontrollausgänge Zur Verhaltenssteuerung gehen folgende Daten: die Erlaubnis für jedes Bein zum Eintritt in die Schwingphase und die veränderten Schwingphasen-Parameter, der Zielpunkt für das Scannen mit dem Distanzsensor sowie die gewählte Scan-Strategie.

Der interne Aufbau der Modulgruppe, die das Weltmodell beschreibt, ist in Abbildung 5.10 dargestellt. Der Kern des Weltmodells liegt dabei in dem Modul *SenseEnvironment*. Dieses umfasst die gesamte in Kapitel 4 und 5 beschriebene Funktionalität: Verarbeitung und Speicherung der Sensorinformation, Verwaltung aller Karten, Auswertung des vorhandenen Wissens und Funktionen zur Visualisierung.

Die Module *GlobalCoordinates* und *GlobalRotation* sind für die Transformation der von der Verhaltenskontrolle lokalen Koordinaten in das GKS verantwortlich; dabei unterscheiden sie sich dadurch, dass *GlobalRotation* ausschließlich die Rotation vornimmt, *GlobalCoordinates* addiert noch einen Translationsvektor. *LocalHeadTarget* berechnet bei Bedarf den Punkt, auf den der Kopfsensor der Maschine bei der lokalen Nachmessung ausgerichtet werden soll. *Multiplexer* und *ReplyValues* dienen nur der Kontrolle des Datenflusses.

5.6 Visualisierung

Um den Inhalt des Weltmodells während dessen Akquirierung sichtbar zu machen, wurde ein Visualisierungswidget für die grafische Oberfläche von MCA, MCAGUI, entwickelt (siehe Abbildung 5.11). Es handelt sich dabei um eine mit Hilfe der Bibliothek *Open Inventor* realisierte dreidimensionale Darstellung aller Gitterzellen der Urkarte. Diese Darstellung kann mit Hilfe der Maus frei inspiziert werden, sie kann gedreht, verschoben, vergrößert und verkleinert werden.

Die grafische Oberfläche läuft auf einem anderen Rechner als die Steuerungssoftware. Um die Daten auf diesen Computer zu übertragen, wird der MCA2-interne *Blackboard-Mechanismus* verwendet, eine Methode, um Datensätze netzwerktransparent zu transportieren. Dies geschieht in unserem Fall via Wireless LAN.

Um die zu transferierende Datenmenge nicht ausufern zu lassen, werden die Visualisierungsinformationen nur auf Anforderung der GUI gesendet. Übertragen werden nur die Daten von Zellen, die seit der letzten Anforderung modifiziert wurden. Dies schont sowohl die Auslastung der Netzwerkverbindung als auch den Prozessor des die Grafik darstellenden Rechners. Wie bereits in Abschnitt 4.2 auf Seite 30 erläutert, sind

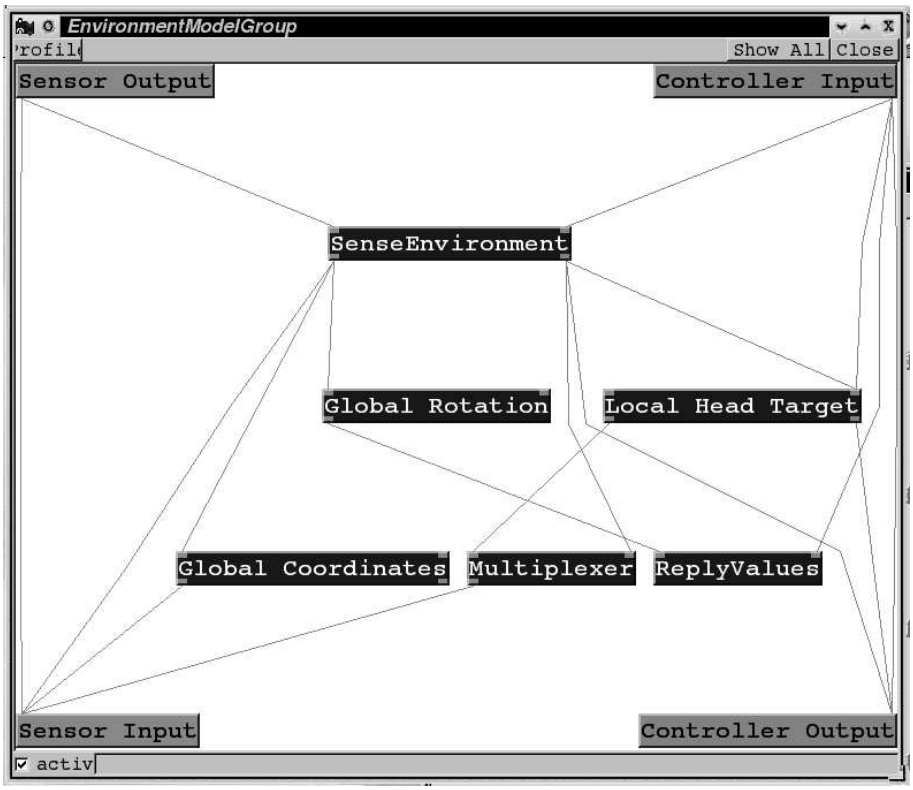


Abbildung 5.10: Interner Aufbau der Umweltmodellierung: Alle Karten und die entsprechenden Auswertungen werden im Modul *SenseEnvironment* durchgeführt.

für die Aufbereitung der Visualisierungsdaten zwei Ebenen des Weltmodells zuständig: die Gitter- und die Umgebungsebene.

Die Gitterebene verwaltet hierbei die letzten Änderungen am Datenbestand des Belegtheitsgitters. Ist die Visualisierungsfunktionalität eingeschaltet, werden bei jedem Schreibzugriff auf der Urkarte die Koordinaten der entsprechenden Zelle gespeichert. Als Datenstruktur dient hierbei eine *Menge* (Set). Dies stellt sicher, dass jede Zelle nur einmal in dieser Struktur gehalten wird. Die so ermittelten Änderungen werden bei Anforderung an die Umgebungsebene weitergegeben, wo aus den Koordinaten und den dazu gespeicherten Daten abhängig von den Spezifika des Roboters Grafikdaten erzeugt und via Blackboard an die Visualisierung gesendet werden. Rahmendaten, die sich nur selten oder gar nicht ändern, wie bspw. die Dimensionierung der Gitterzellen oder die gewählte Grundfarbe, werden auf einem getrennten Datenkanal in einem *grafischen Kontext* übergeben. Ein solcher wird nur gesendet oder angefordert, wenn sich an den Rahmenbedingungen etwas geändert hat.

Da die Kapazität des Netzes begrenzt ist, kann es vorkommen, dass nicht alle Änderungsdaten innerhalb eines Zyklus' von der Steuerung an die Visualisierung geschickt werden können. Deshalb wird in der Gitterschicht zusätzlich eine *Warteschlange* (Queue) bereit gestellt, in die bei Anforderung von Änderungsdaten der Inhalt der Menge kopiert wird. Sie wird danach der Umgebungsschicht übergeben, die diese dann sukzessive leert und die Daten an den darstellenden Rechner schickt. Bei erneuter An-

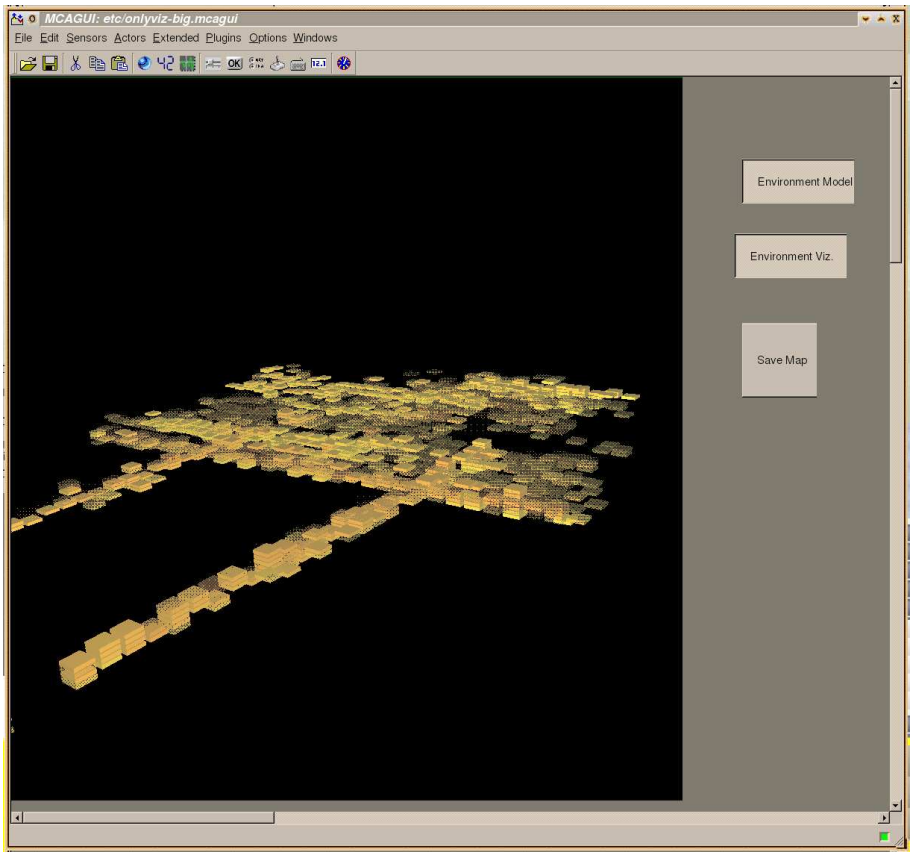


Abbildung 5.11: Visualierungswidget der grafischen Oberfläche MCAGUI zur Sichtbarmachung der Weltmodelldaten. Belegtheitsinformation ist durch Transparenz, Zuverlässigkeit durch Farbabstufungen dargestellt. Das Modell kann durch Mausbedienung frei bewegt werden.

forderung von Seiten der grafischen Oberfläche wird wiederum der Inhalt der Menge in eine Warteschlange kopiert und an die Umgebungsschicht übergeben.

Auf Seiten der Visualisierung wird der Bestand an Grafikdaten ebenfalls in einer Datenstruktur gehalten. Da es sich hierbei prinzipiell auch um eine 3D-Karte handelt, kann dazu dieselbe mehrschichtige Struktur und auch die selben Programmklassen verwendet werden wie für das Weltmodell selbst (zur Struktur siehe Abschnitt 4.2). Als Gitterzelle fungiert hier eine Datenstruktur, die ausschließlich einen Farb- und einen Transparenzwert enthält.

Kapitel 6

Experimente

In diesem Kapitel sollen anhand einiger Versuche mit der Laufmaschine LAURON die in den letzten Kapiteln vorgestellten Konzepte auf Tauglichkeit untersucht werden. Dazu sollen sowohl die Fähigkeit des Weltmodells zur geeigneten Aufnahme von Umweltdaten als auch die Möglichkeit der Extraktion von Wissen aus diesem experimentell nachgewiesen werden.

6.1 Rahmenbedingungen bei der Durchführung der Versuche

Im Laufe der Anfertigung dieser Arbeit zeigte sich, dass der ursprünglich auf dem Kopf von LAURON montierte Infrarot-Sensor Sharp GP 2DR für den vorliegenden Einsatzzweck vollkommen ungeeignet ist. Seine maximale Reichweite von 80 cm genügt nicht, um Umweltdaten in der Umgebung vor der Maschine in einem Ausmaß zu sammeln, dass sie sinnvoll auswertbar wären. Insbesondere liegt die Entfernung zwischen Kopf und Füßen von LAURON je nachdem, wie hoch sich der Roboter aufstellt, schon im absoluten Grenzbereich des Sensors. Zudem waren die Daten sehr ungenau und verrauscht, so dass ab einer Entfernung von 50 cm kaum noch zwischen Hindernis und Freiraum unterschieden werden konnte, so dass noch nicht einmal eine ungefähre Kartierung des Bodens möglich war. Aus diesem Grund wurde der Infrarot-Sensor im Verlauf der Arbeit durch einen Laser-Sensor der Firma Leuze ersetzt, der in Anhang A näher beschrieben ist. Zum Test mit dieser Sensorik blieben nur wenige Tage, so dass sowohl die Kinematik des neuen, vorerst nur provisorisch montierten Kamerakopfes als auch die Kalibrierung des Sensors noch nicht optimal ausgetestet werden konnten. Ein weiterer, am Körper des Roboters fest angebrachter Infrarotsensor, der den Bodenabstand misst, konnte im Zuge der Montage des Laser-Sensors aus technischen Gründen nicht mehr in Betrieb genommen werden; die Relevanz der von ihm gelieferten Daten war allerdings eher stets gering.

Eine weitere Einschränkung musste bezüglich der Odometrie der Maschine gemacht werden, die derzeit Gegenstand eines weiteren Projekts ist, dessen Ergebnisse aber zum Zeitpunkt der Erstellung dieser Arbeit noch nicht vorliegen. So musste mit einer provisorischen Odometrie gearbeitet werden, deren Ausgaben mitunter noch Exaktheit vermissen ließen. Insbesondere ist die Drehung der Maschine um die Höhenachse

nicht ermittelbar, so dass es bei der Durchführung der Versuche nicht möglich war, die Maschine nach rechts oder links drehen zu lassen und nur Geradeaus-Bewegungen durchgeführt werden konnten, wenn man verwertbare Umweltdaten erhalten wollte.

6.2 Kartierung der Umwelt

In einem ersten Experiment sollte untersucht werden, wie es um die Qualität der verwendeten Kartierungsmethode, auch im Vergleich mit einem anderen Verfahren, bestellt ist. Dazu wurde aus Paletten ein kleines Szenario aufgebaut, dessen geometrische Eigenschaften sich möglichst genau in der Umweltrepräsentation des Weltmodells wiederfinden sollten (siehe Abbildung 6.1).

Abbildung 6.1: Test-Szenario für den Versuch zur Kartierung: Eine Sackgasse mit Stufen in verschiedenen Höhen, in die LAURON hinein laufen soll.

LAURON wurde nun mehrfach von einer festgelegten Position aus in diese Sackgasse hinein laufen gelassen (Abbildung 6.2), und zwar so, dass stets die rechten Beine über die Stufen steigen mussten. Dabei wurde der Boden in einer Entfernung von 45 cm vor der Maschine linear von links nach rechts abgetastet. Für die Breite eines solchen Scans wurden 140 cm gewählt, die Zykluszeit eines Scandurchgangs betrug 4 Sekunden[1]. Mit diesem Experiment soll nachgewiesen werden, dass

1. die Repräsentation der Umwelt, wie sie bei der Aufnahme gewonnen wird, die reale Welt geeignet abbildet und Merkmale extrahierbar sind

[1] Es ist zu beachten, dass der Wert von 140 cm sich auf die Länge bezieht, die der Scan abfährt, wenn er auf der ursprünglichen Höhe der Füße auf den Boden trifft. Bei höheren Bodenerhebungen ist er entsprechend kürzer.

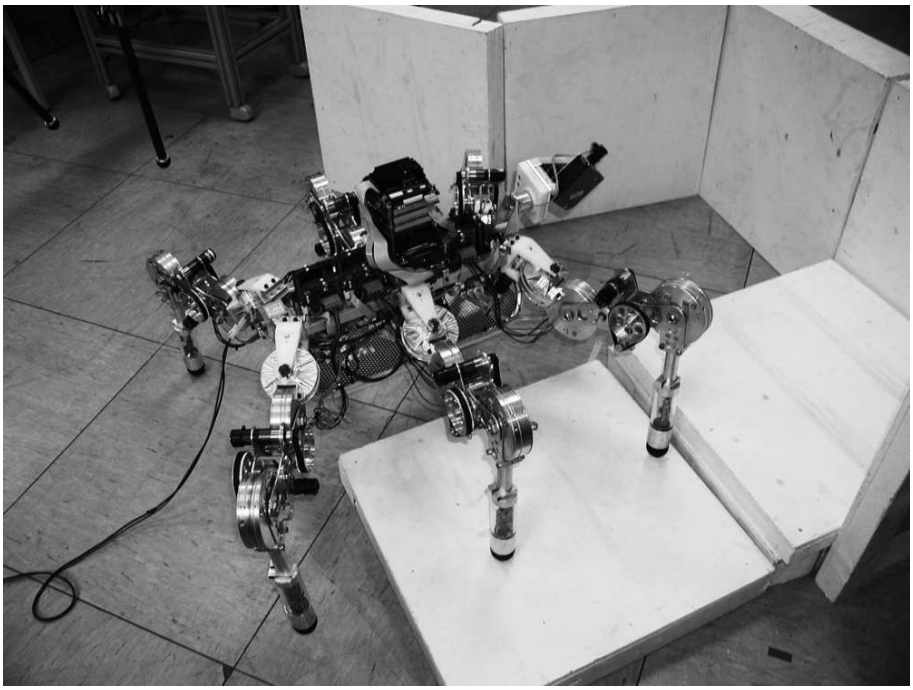


Abbildung 6.2: LAURON beim Einscannen des Beispiel-Szenarios

2. die Ergebnisse reproduzierbar sind, dass also eine wiederholte Erfassung der Umwelt zu vergleichbaren Kartierungsergebnissen führt

3. die in Abschnitt 4.6 beschriebene Methode zur Bewertung der Sensormessung zu hochwertigen Ergebnissen führt.

6.2.1 Genauigkeit der Abbildung

Abbildung 6.3 zeigt zwölf aufeinander folgende Momentaufnahmen der Urkarte, die mit dem Visualierungsplugin für die grafische Bedien-Oberfläche von MCA2 (vgl. Abschnitt 5.6) aufgezeichnet wurden. Der Standpunkt der Maschine lässt sich an den im ersten Bild erkennbaren drei einzelnen Fußaufsetzpunkten ablesen, die sich dort abzeichnende Kante ist die erste Scanlinie. Die Bewegung des Roboters erfolgte dann orthogonal zu dieser Linie.

In dieser Visualisierung der 3D-Urkarte sind für jede Zelle x sowohl die Belegtheitswahrscheinlichkeit $\mathrm{occ}(x)$ als auch die Zuverlässigkeit $\mathrm{cred}(x)$ sichtbar. Jeder der gezeigten Quader entspricht einer Zelle des Belegtheitsgitters mit den Abmessungen $5 \times 5 \times 1$ cm. Die Belegtheitswahrscheinlichkeit wird durch den Füllgrad des Quaders dargestellt, die Zuverlässigkeit durch den Farbwert: Gelb steht für einen sehr geringen, rot für einen sehr hohen Wert von $\mathrm{cred}(x)$. Es ist ablesbar, dass die Zuverlässigkeit der Messungen im Laufe des Versuchs durch die immer größere Anzahl von Messdaten stetig zugenommen hat (auf die Zuverlässigkeit der Kartierung wird im weiteren Verlauf dieses Abschnitts noch eingegangen werden). Gut ersichtlich ist auch die Fusion der Sensormessungen: Die von den Fußsensoren aufgenommenen Daten „verschmelzen“ in

Folge der Bewegung der Maschine mit denen, die der Abstandssensor aufgenommen hat.

Sobald Lauron das Ende der Sackgasse erreichte, wurde er angehalten und eine globale Karte des bisher erfassten Ausschnitts der Umwelt erzeugt. Abbildung 6.4 zeigt links diese geometrische Karte. Die Höheninformation ist farblich dargestellt: Die tief dunkelblauen Stellen sind undefiniert, an diesen Positionen liegen keine Messdaten vor. Blau repräsentiert niedrige Höhenwerte, höhere sind zunächst in Grün- und Gelbtönen dargestellt. Dies geht bis hin zu den größten gemessenen z-Koordinaten, die durch rote Farbe repräsentiert sind. Die Ausdehnung eines Kästchens auf der xy-Ebene entspricht der Auflösung einer Gitterzelle der Urkarte, $5 \times 5\,\mathrm{cm}$.

Das aufgebaute Szenario ist dabei gut wiederzuerkennen: Die Fußspuren sind klar erkennbar, ebenso zeigt sich auf dem Boden eine relativ homogene Struktur mit nur geringen Abweichungen. Sehr exakt sind die beiden Treppenstufen erfasst, die ebenfalls gleichmäßige Höhenwerte aufweisen. Auch die Erhebung der Wand am Ende der Sackgasse wurde sehr genau kartografiert.

Nicht in die Karte eingetragen wurde die Wand auf der rechten Seite – sie wurde vom Kopfsensor während des Experiments gar nicht erfasst, die Umgebung wird dementsprechend als undefiniert klassifiziert. Ebenfalls finden sich in der gesamten Karte verteilt immer wieder nicht definierte Zellen. Hier wurde bei der Kartengenerierung keine ausreichend sichere Belegtheitsinformation gefunden, um eine Aussage über diese Zelle treffen zu können. Eine optimale Erfassung der Umwelt durch die Sensorik war auch nicht zu erwarten, die unbekannten Stellen wurden also durchaus richtig eingetragen.

Auffällig ist die rechte Beinspur: Die auf ihr liegenden Zellen wurden durchgängig leicht höher klassifiziert als die Umgebung. Dies ist offensichtlich nicht korrekt und liegt in der Erfassung der Fußpositionen durch die Fußsensorik begründet. Auch in der Visualisierung der Urkarte in Abbildung 6.3 ist zu erkennen, dass oftmals kleine „Stapel“ von Zellen auf den Fußspuren eingetragen wurden. Diese entstehen, wenn das entsprechende Bein in die Schwingphase eintritt und den Punkt verlässt – eine Schwachstelle in der Sensordatenerfassung, die leider auch in anderen Experimenten stets zu beobachten war.

Auf der rechten Seite von Abbildung 6.4 findet sich eine Darstellung der Zuverlässigkeitswerte der gleichen Karte. Hellere Werte bedeuten dort größere Zuverlässigkeit. Deutlich zu sehen ist hier, dass es mehrere homogene Gebiete mit höheren Werten von $\mathrm{cred}(x)$ gibt. Einerseits sind dies die Aufsetzpunkte der Beine, da diese natürlich längere Zeit an einem Ort stehen und so eine Vielzahl von Messungen liefern. Ebenso ist die Sicherheit an Rückwand und der Startlinie groß: An beiden Stellen hatte Lauron mehrere Sekunden im Stand Daten gesammelt. Geringe Zuverlässigkeitswerte finden sich dagegen vor allem an der linken Seitenwand, hier schlägt zu Buche, dass dort die Bewertung der (unbekannten) Nachbarschaft ungünstig ausfällt. Die schwachen Sicherheitswerte an dieser Hürde setzen die Anforderung aus Abschnitt 3.1.6 nach behutsamer Hindernisklassifizierung um: Im Zweifelsfall sollte an möglichen Hindernissen noch einmal nachgemessen werden, ob sie nicht doch bewältigbar sind.

Die Qualität der erzeugten Karte ist alles in allem sehr hoch: Alle im erfassbaren Bereich liegenden Merkmale der realen Welt wurden detailgetreu erfasst. Eine Aus-

Abbildung 6.3: Sukzessiver Aufbau der Urkarte des Weltmodells: Die Belegtheitswahrscheinlichkeit wird durch den Füllgrad der Kästchen symbolisiert, der Farbwert markiert die Zuverlässigkeit – je höher der Rotanteil, desto höher. Gut sichtbar ist die Spur der Füße.

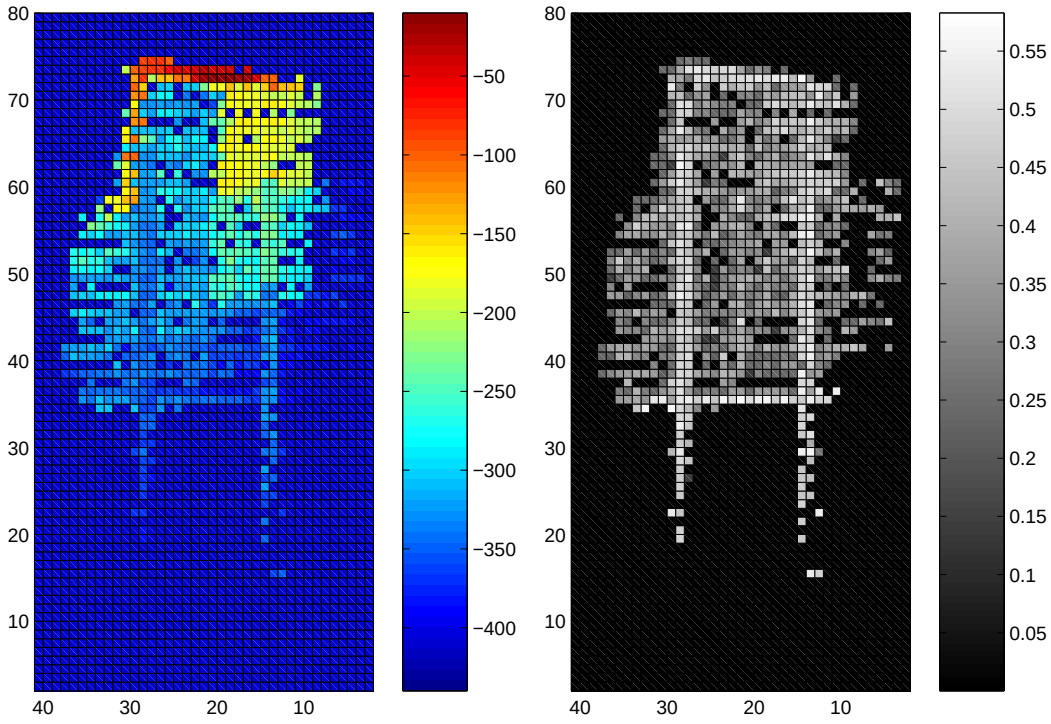


Abbildung 6.4: Belegtheits- (links) und Zuverlässigkeitskarte (rechts) des vermessenen Szenarios. Ein Kästchen stellt hierbei eine Gitterzelle von 5 × 5 cm dar. Die Höhenwerte für die linke Karte sind in mm angegeben.

wertung einer solchen Karte ist mit wenig Aufwand möglich; so könnten aus den hier ermittelten Darstellungen problemlos alle enthaltenen Kanten extrahiert werden.

6.2.2 Reproduzierbarkeit des Ergebnisses

Um aufzuzeigen, dass die Ergebnisse der Kartierung reproduzierbar sind, wurde das Experiment mehrmals unter möglichst gleichen Rahmenbedingungen wiederholt. Abbildung 6.5 stellt die resultierenden Karten aus zwei aufeinander folgenden Testläufen gegenüber. Die Unterschiede zwischen den Ergebnissen sind denkbar gering, Abweichungen sind nur in kleinen Details erkennbar. So wurde im zweiten Lauf die schräge Wand zur Linken besser erfasst, weil der Roboter etwas in diese Richtung abgedriftet war. Auch sind die Höhenwerte insgesamt in der rechten Karte leicht niedriger. Dies lässt sich auf die nicht perfekte Odometrie der Maschine zurück führen (vgl. dazu auch Abbildung 6.12). Wichtig ist hier die Erkenntnis, dass auch bei solchen Abweichungen seitens der Odometrie homogene Gebiete als solche erkannt werden und sich die Störungen nicht lokal bemerkbar machen. Zudem liegen die Unterschiede im Bereich von maximal 2 cm, was als ausreichend genau betrachtet werden kann, da es viel wichtiger ist, dass die Karten in sich konsistent sind. Eine Abweichung in der Absoluthöhe ist relativ unerheblich, da die ermittelte Position des Roboters die gleiche Abweichung aufweisen wird. Insgesamt lässt sich festhalten, dass die vorliegende Umweltmodellierung reproduzierbare Daten liefert.

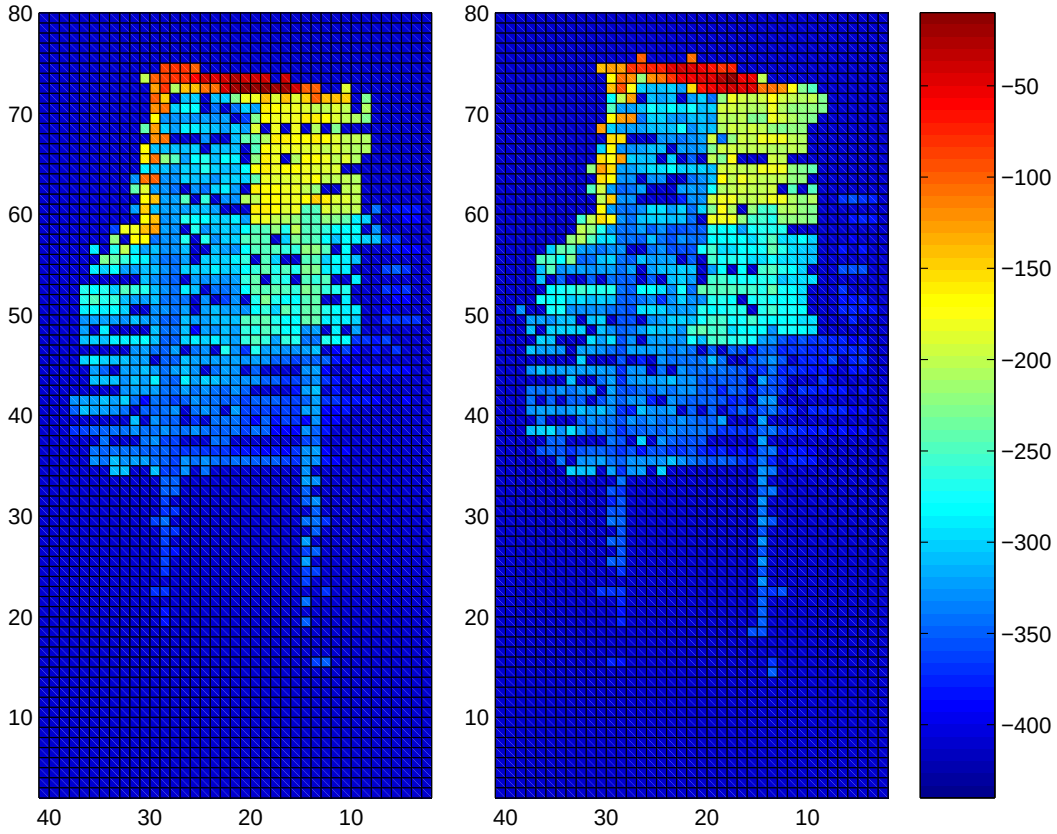


Abbildung 6.5: Gegenüberstellung der resultierenden Karten zweier verschiedener Testläufe: Die Repräsentationen unterscheiden sich kaum. Höhenangaben in mm.

6.2.3 Erneutes Besuchen bereits erfasster Gebiete

Es sollte nun ausgetestet werden, wie die Repräsentation der Umwelt beeinflusst wird, wenn die Maschine an Stellen zurückkehrt, die sie bereits in das Weltmodell eingetragen hatte. Wegen der eingangs dieses Kapitels erwähnten Einschränkung, dass Drehungen in der Ebene von der Odometrie nicht erfasst werden, konnte dies nur dadurch realisiert werden, dass LAURON den gleichen Weg, den er gekommen war, rückwärts wieder zurück ging.

Abbildung 6.6 stellt sowohl die Geometrie- (oben) als auch die Zuverlässigkeits-Karten (unten) gegenüber. Links ist jeweils die Karte, die beim Erreichen der Rückwand der Sackgasse erzeugt wurde, die rechte wurde erstellt, nachdem der Roboter den gleichen Weg wieder zurück gegangen war. Der Vergleich zeigt, dass beide Karten sich generell in ihrer Aussage nicht unterscheiden. Die aus Hin- und Rückweg erzeugte Darstellung weist allerdings deutlich weniger undefinierte Stellen auf, auch sind die Plateaus der Abstufungen als einheitlichere Flächen erkannt. Die Qualität der Umwelt-Repräsentation hat sich durchweg verbessert. Dies stützt auch ein Blick auf die Zuverlässigkeitskarten: Nach dem Rückweg wird eine viel höhere und einheitlichere Sicherheit für die Kartendaten ausgewiesen. Deutlich sichtbar ist auch, an welcher Stelle die Rückwärtsgang der Maschine abgebrochen wurde: Ungefähr bei Zelle 50 geht die Vertraulichkeit sichtbar zurück.

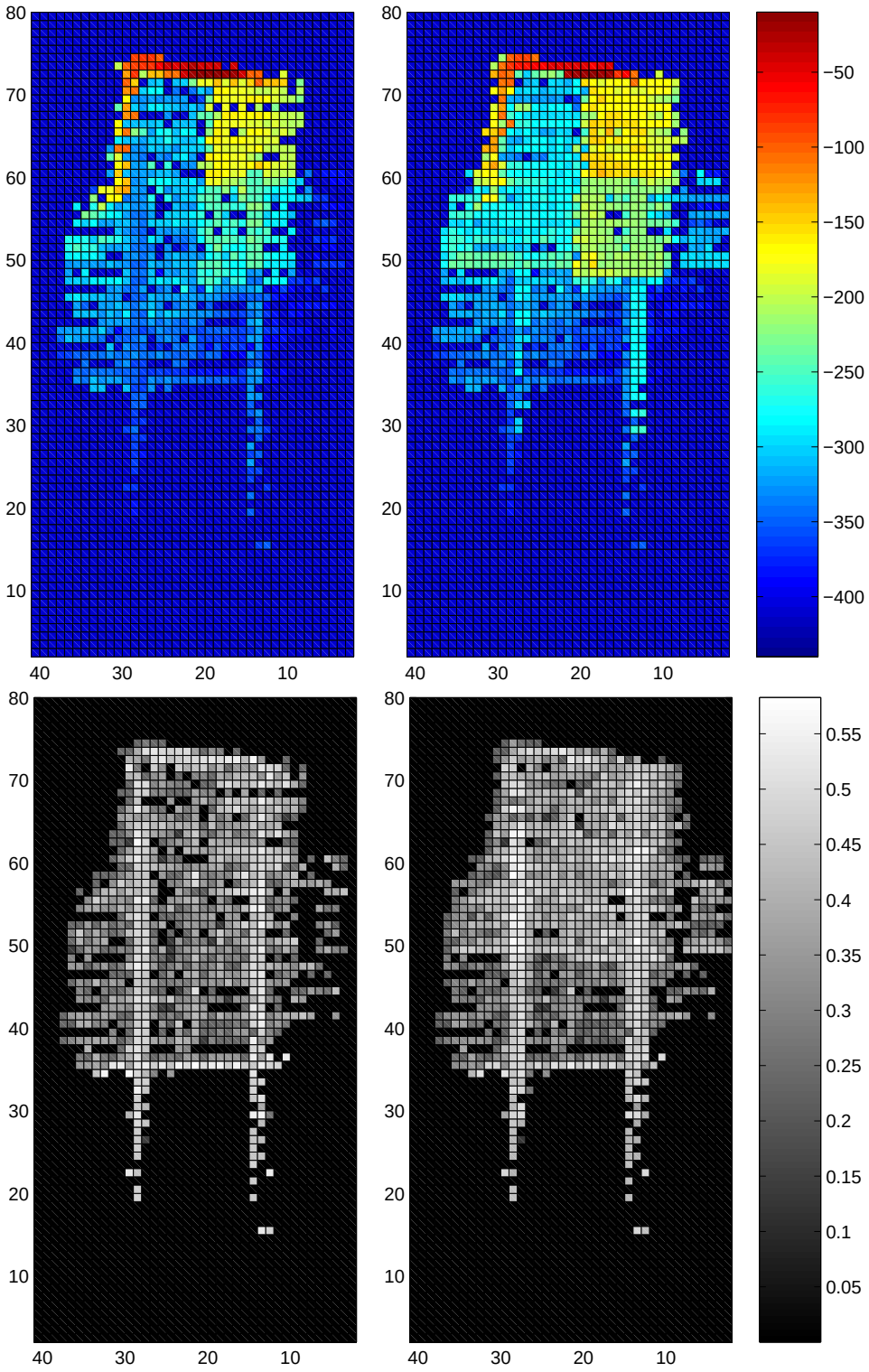


Abbildung 6.6: Einfache und doppelte Kartierung desselben Gebiets: Links die Karten nach dem Hin-, recht nach Hin- und Rückweg; die oberen Karten zeigen Belegtheits-, die unteren Zuverlässigkeitswerte. Die Qualität der Karten hat sich durch das Begehen des Rückwegs deutlich verbessert.

6.2.4 Vergleich mit einem einfachen Histogrammgitter

Um die Qualität des verwendeten Ansatzes des Erweiterten Inferenz-Gitters (siehe Abschnitt 4.4) zu überprüfen, wurde zusätzlich das Experiment mit einem anderen, verbreiteten Verfahren zur Umweltrepräsentation durchgeführt. Dafür wurde die Umgebung mittels ein Histogramm-Gitters gespeichert (siehe Abschnitt 3.1.5). Jede Zelle kann hier Belegtheitszustände zwischen 0 und 13 annehmen, wobei dieser Wert bei jedem erfassten Hindernis um 1 erhöht, ansonsten um 1 vermindert wird. Dazu wurde wie im vorigen Abschnitt beschrieben, der Roboter bis zur Wand und wieder zurück laufen gelassen. Abbildung 6.7 stellt das Ergebnis dem bislang getesteten Verfahren gegenüber. Es zeigt sich, dass dieser Ansatz zwar auch ein recht detailgenaues Bild der

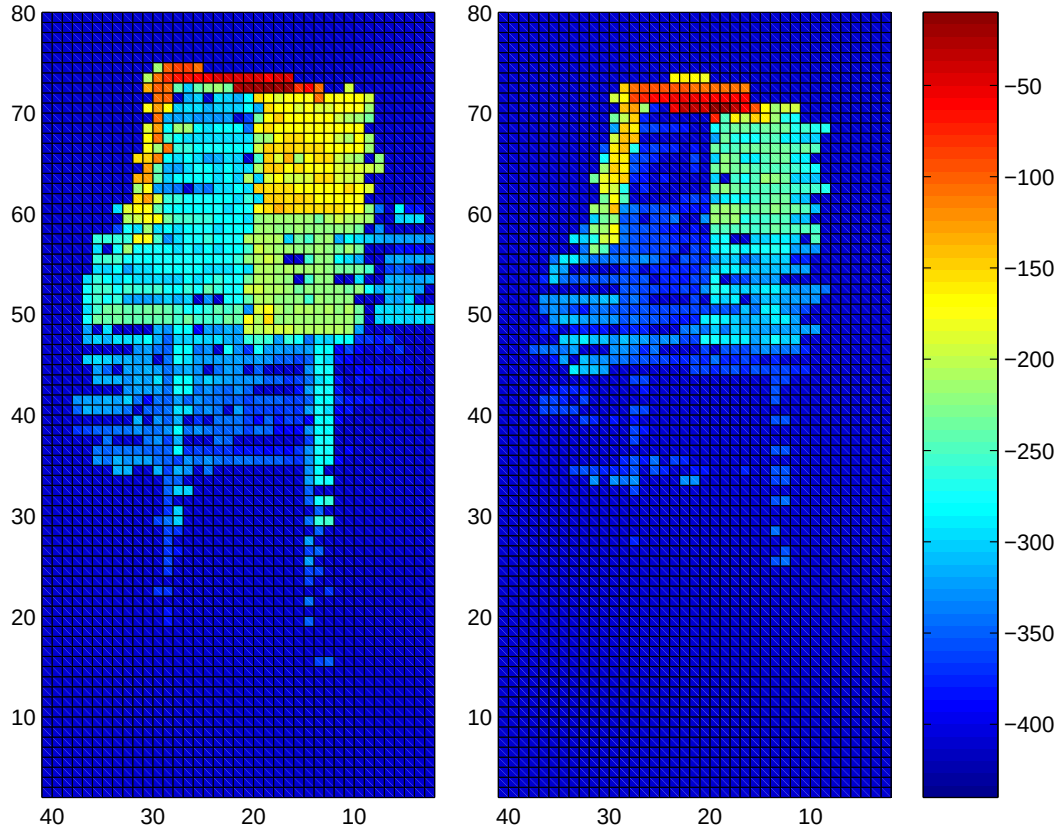


Abbildung 6.7: Vergleich des Erweiterten Inferenz-Gitters (links) mit einem Histogramm-Gitter (rechts): Das Histogramm-Gitter erfasst in der gleichen Zeit wesentlich weniger und ungenauere Information.

realen Umwelt erzeugt, jedoch deutlich weniger mit ausreichender Sicherheit erkannte Zellen in gleicher Zeit zurück liefert wie das in dieser Arbeit entwickelte Verfahren. Zudem ist das Ergebnis stärker verrauscht und der Boden wurde durchweg etwas zu niedrig erfasst. Ein deutlicher Unterschied zeigt sich in der Erfassung der Rückwand: Der Histogramm-Gitter-Ansatz klassifiziert dort zu viele Zellen als Hindernis, die Wand ist überall scheinbar breiter, als es der tatsächlichen Messung entsprechen sollte. Der in dieser Arbeit benutzte Ansatz des Erweiterten Inferenz-Gitters erstellt ein exakteres und homogeneres Abbild der Umgebung und stellt zudem durch die zusätzliche Zuver-

lässigkeitsinformation, die beim Histogramm-Gitter lediglich implizit in der geometrischen Darstellung enthalten ist, eine wichtige weitere Quelle für die Wissensextraktion aus dem Umweltmodell zur Verfügung.

Es lässt sich zusammenfassend sagen, dass die Kartierungsfähigkeiten der getesteten Weltmodellarchitektur den gestellten Anforderungen voll entsprechen. Die Umwelt wird detailgenau abgebildet, die Ergebnisse sind reproduzierbar, und die Bewertungsmethode für die Sensordaten führt zu guten Ergebnissen.

6.3 Wissensextraktion aus dem Weltmodell

In einem weiteren Versuch sollte nun gezeigt werden, dass die Steuerung des Roboters in der Lage ist, aus den Umweltdaten gewonnenes Wissen geeignet zu verarbeiten und das Laufverhalten der Maschine entsprechend anzupassen. Dazu wurde aus Paletten ein kleiner Hindernisparcours erstellt, den der Roboter überwinden sollte (Abbildung 6.8).

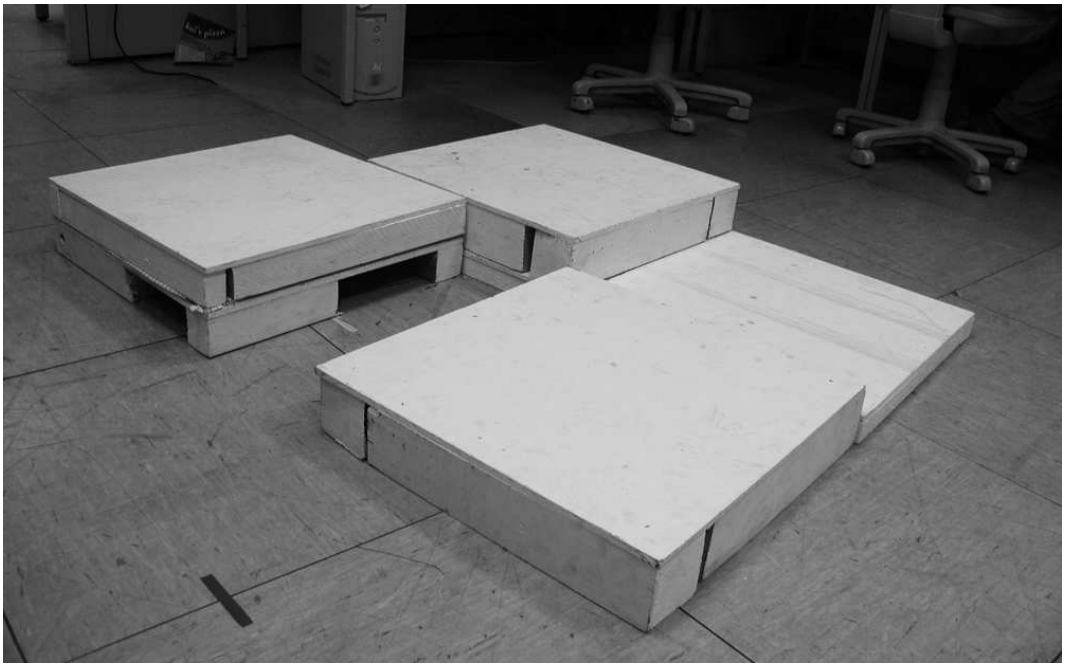


Abbildung 6.8: Hindernisparcours, der von LAURON überwunden werden soll

Auch mit der bisherigen Steuerung konnte LAURON solche Umgebungen meistern. Allerdings konnte er bislang Hindernisse erst erkennen, wenn er mit dem Fuß gegen eines gestoßen war; danach wurde auf Verdacht sukzessive die Schritthöhe vergrößert, bis der Fuß einen Aufsetzpunkt gefunden hatte. Ähnlich war beim Herabsteigen von Stufen vorgegangen worden: Wenn der Fuß in der erwarteten Höhe keinen Bodenkontakt erreichte, wurde sich mit kreiselnden Bewegungen nach unten getastet, bis ein sicherer Stand erreicht war. Es soll nun gezeigt werden, dass dieses Verhalten durch die Umweltmodellierung im Optimalfall nicht mehr benötigt wird und nur noch in Ausnahmefällen eingesetzt werden muss: Einerseits soll die Maschine ihre Füße zielgerichtet durch einen großen Schritt auf die Hindernisse platzieren, andererseits soll sich beim

Herabsteigen der Zielpunkt der Schwingtrajektorie des Fußes auf dem niedrigeren Niveau befinden und so die Suchbewegungen durch einen sicheren Schritt abwärts ersetzt werden.

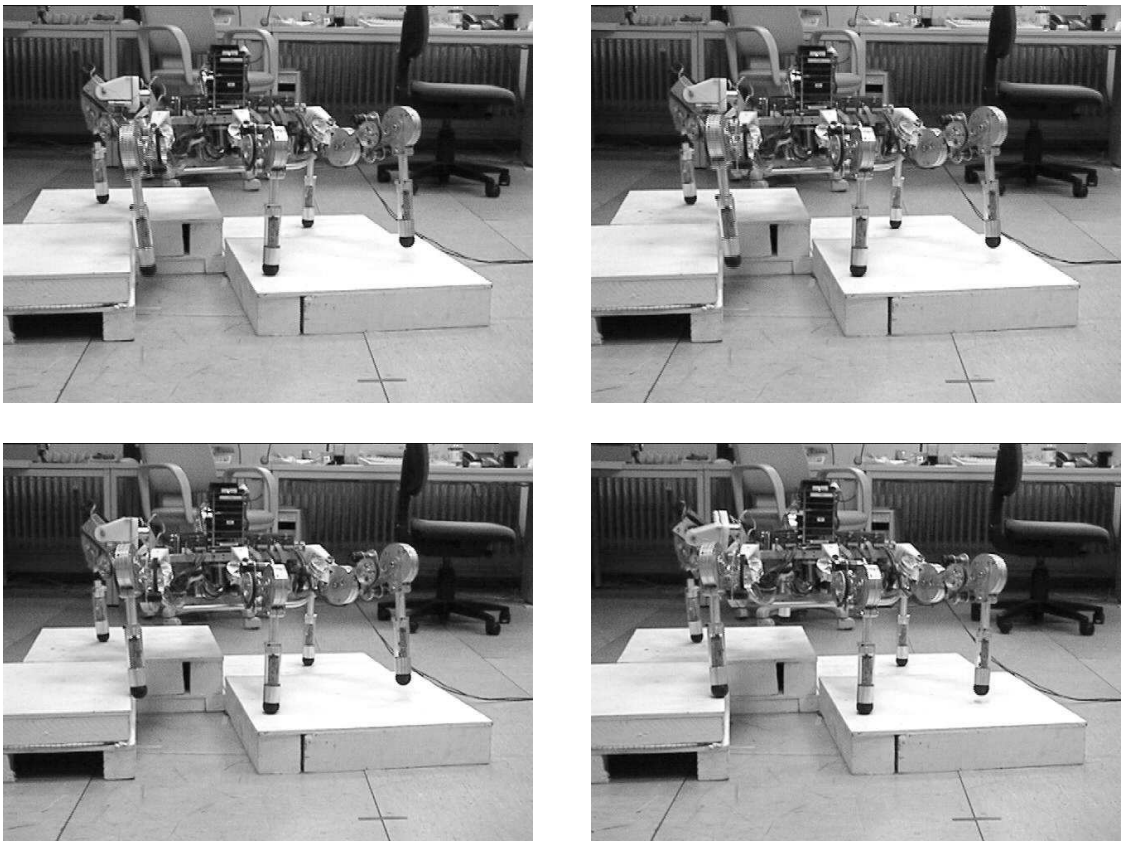


Abbildung 6.9: LAURON beim Überqueren des Hindernisses. Das linke Vorderbein wird zielsicher und kollisionsfrei auf die Stufe positioniert.

Beim Überqueren der Hindernisse war zu beobachten, dass die oben genannten Erwartungen erfüllt werden konnten: In den meisten Fällen konnte die Maschine ihre Schritthöhe derart anpassen, dass sie die Füße ohne Anstoßen direkt auf die Treppenstufe setzen konnte, insbesondere den Mittel- und Hinterbeinen gelang dies sogar nahezu jedesmal. Auch beim Heruntersteigen von Stufen wurde der Fuß meist direkt auf das niedrigere Niveau manövriert. Im Gegensatz zur bisherigen Steuerung war also eine deutliche Verbesserung zu erkennen. An Fehlverhalten zeigte sich lediglich, dass die Maschine des öfteren zu früh mit den hohen Schritten begann. Eine Erklärung dafür lässt sich aus Abbildung 6.11 ablesen, die die aufgenommene Umgebungskarte dieses Experiments zeigt: Es ist zu sehen, dass die Fußspuren deutlich über die Grenzen der Hindernisse hinausgehen – der Roboter wähnte sich also schon weiter vorne, als er wirklich gegangen war. Dieser Effekt trat um so stärker auf, desto schwieriger das Gelände war.

Erwähnenswert ist, dass vor allem die Mittel- und Hinterbeine ausreichend Information aus dem Weltmodell extrahieren konnten, um dem Gelände angemessen zu reagieren. Dieser Qualitätsgewinn für die hinteren Beinpaare ist durchaus erwartbar: Sie können zusätzlich auf durch die Vorderbeine gewonnene Umweltinformation zurück-

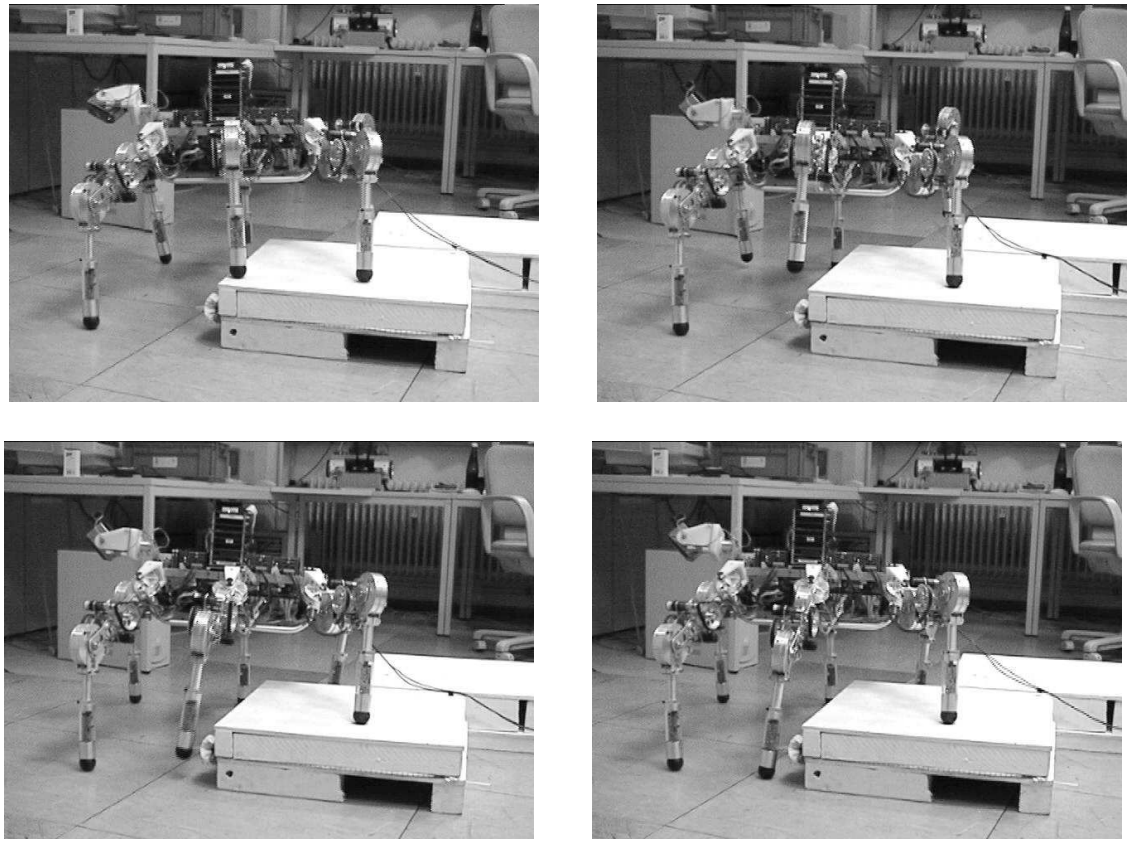


Abbildung 6.10: LAURON steigt ein Hindernis herab. Der Fuß in der Mitte links wird ohne Zögern sofort auf den Boden platziert.

greifen. Die für diese Beine generierten Fußumgebungskarten werden also eine größere Gesamtsicherheit aufweisen und die Wahl geeigneterer Fußaufsetzpunkte ermöglichen können.

Ein Grund für das nicht optimale Laufverhalten ist sicherlich in der nur provisorisch realisierten Odometrie zu finden. Ein Blick auf die während dieses Testlaufs aufgezeichneten Odometrie-Daten stützt diese Einschätzung: Abbildung 6.12 zeigt den Verlauf der ermittelten absoluten Körperhöhe im Laufe des Experiments, eine Größe, an der Ungenauigkeiten gut ablesbar sind. Obwohl LAURON zu Beginn und Ende des Testlaufs auf gleichem Höhenniveau stand, differiert der ermittelte Höhenwert deutlich: Er lag nach Abschluss des Experiment um ca. 7,5 cm höher als zu Anfang. Erkennbar ist auch, dass er auch nach dem Überwinden der Hindernisse beim Gehen auf ebenem Untergrund stetig weiter anstieg.

Insgesamt lässt sich feststellen, dass trotz der ungenauen odometrischen Daten das Laufverhalten von LAURON deutlich verbessert und an die Umgebung angepasst werden konnte. Das im Weltmodell gespeicherte Wissen lässt sich geeignet für die Anpassung der Steuerung verwenden.

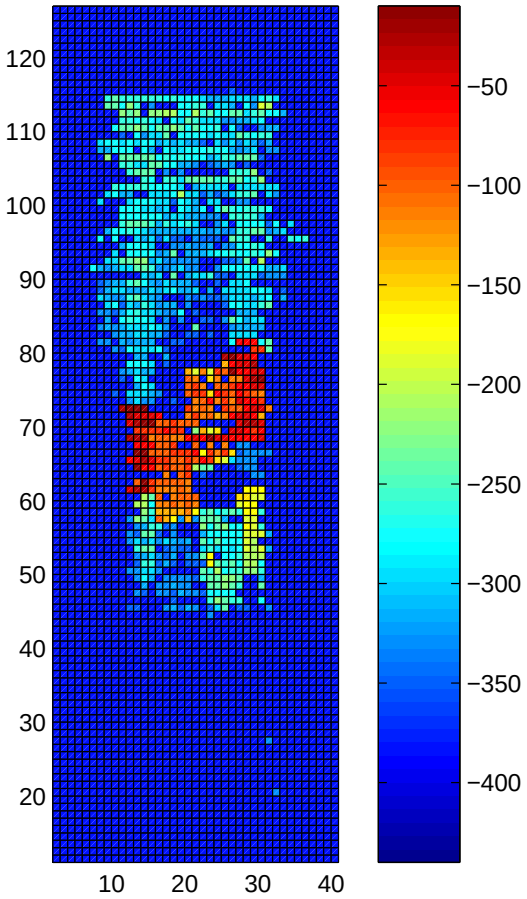


Abbildung 6.11: Karte, die während des Überqueren des Hindernisparcours' aufgenommen wurde. Deutlich erkennbar ist, dass die von den Fußsensoren erfassten Werte fälschlicherweise stets deutlich über das Hindernis herausragen.

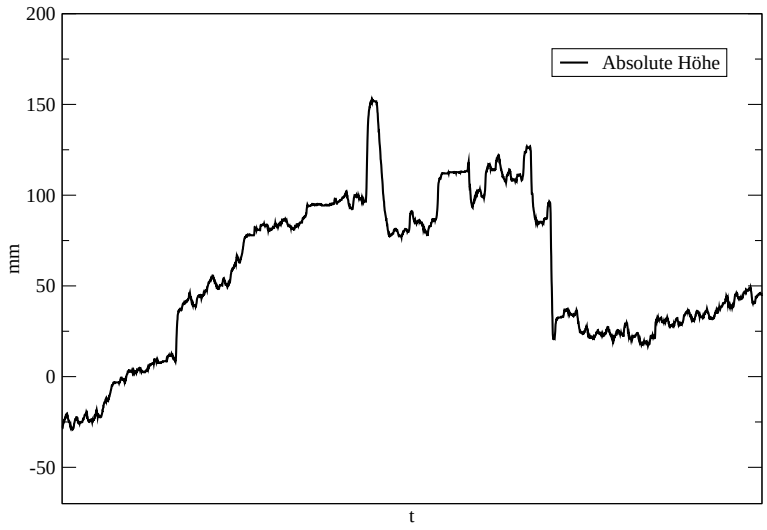


Abbildung 6.12: Verlauf der von der Odometrie gelieferten absoluten Körperhöhe der Maschine: Die z-Koordinate steigt stetig an, auch wenn sich der Roboter nicht nach oben bewegt.

6.4 Zielgerichtetes Laufverhalten

In einem letzten Experiment sollte überprüft werden, ob die Anpassung des Laufverhaltens durch die aus dem Weltmodell extrahierte Information ausreichen würde, um LAURON über eine Spalte steigen zu lassen (siehe Abbildung 6.13), eine Herausforderung, die mit der bisherigen Steuerung nicht zu bewältigen war. Dazu sollte in diesem Experiment auch die Kopfsteuerung und das zielgerichtete Nachmessen unsicherer Gebiete untersucht werden, indem der Roboter in Richtung einer aus Tischen erbauten Spalte laufen gelassen wurde, in der Hoffnung, er möge ihn überschreiten. Dies ge-

Abbildung 6.13: LAURON beim Versuch, eine Spalte zwischen zwei Tischen zu überqueren

lang allerdings auch mit der an die Umweltmodellierung angepasste Steuerung nicht: LAURON setzte seine Füße zwar nahe an den Rand des Spalts, trat danach allerdings dennoch hinein. Der Grund war in der Visualisierung deutlich erkennbar: Der Spalt war zwar zunächst als solcher erkannt, später aber wieder geschlossen worden. Die Gegend im Bereich der Tischkanten war zudem extrem verrauscht. Hier zeigt sich, dass die Qualität der eingehenden Sensordaten und des odometrischen Systems dringend noch verbessert werden muss, um solche Aufgaben, in denen es um zentimetergenaue Positionierung geht, lösen zu können.

Dem Test des zielgerichteten Nachmessens konnte dagegen Erfolg beschieden werden: Zu Beginn des Testlaufs, beim Erreichen des Grabens und vereinzelt auch dazwischen, also immer dann, wenn die erfassten Daten keine ausreichende Genauigkeit aufwiesen, änderte LAURON die Scan-Strategie des Kopfes und führte lokale Nachmessungen unmittelbar vor den Vorderfüßen durch.

Auch konnte beobachtet werden, wie sich die Maschine verhält, wenn die Umweltinformation extrem spärlich ist. Zu Beginn des Experiments war die Umgebung der Mittel- und Hinterbeine vollkommen unbekannt. Da für diesen Test die Größe der zu erstellenden Fußumgebungskarten so groß gewählt worden war, dass ein Überschreiten

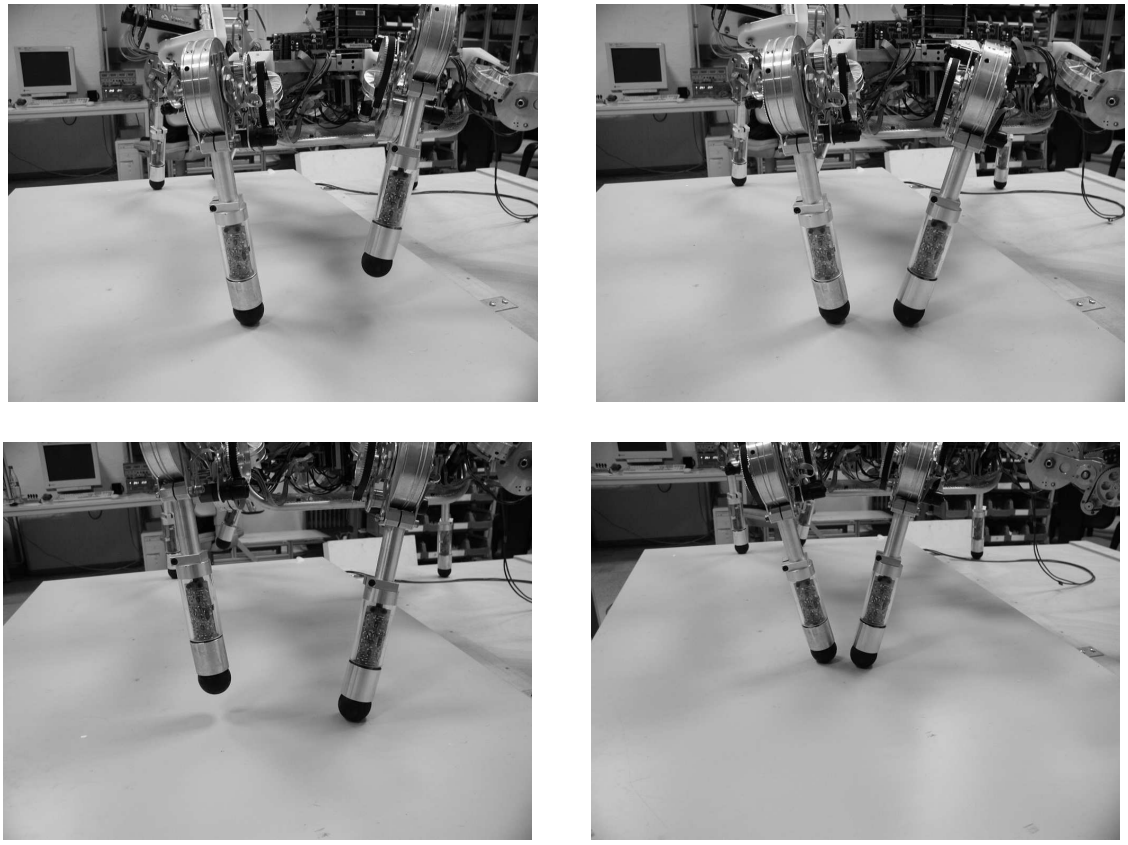


Abbildung 6.14: „Follow the leader"-Strategie bei LAURON: Weil keine anderen Stellen der Umgebung bekannt sind, setzt die Maschine zielsicher ihre Füße auf die bekannte Position des Vorgängers.

des Spalts möglich gewesen wäre, gab es in jeder Karte dann doch eine Stelle, für die Informationen vorlagen: die Position des vorderen Beins. Diese wurde als einzig möglicher Fußaufsetzpunkt für die nachfolgenden Beine ermittelt und folgerichtig setzte LAURON den Fuß mit einem großen Schritt genau auf die Position seines Vorgängers. Die Bildserie in Abbildung 6.4 verdeutlicht dieses Verhalten. Die Steuerung setzte also, ohne explizit dazu angehalten worden zu sein, eine so genannte *„Follow-the-leader"-Strategie* durch, wie sie auch im Tierreich häufig zu finden ist.

Kapitel 7

Zusammenfassung und Ausblick

In dieser Arbeit wurde der Entwurf einer Umweltmodellierung vorgestellt, um die Navigation von Laufmaschinen im Gelände zu ermöglichen. Es wurde ein Verfahren entwickelt, mit dem es möglich ist, sensorisches Wissen über die Umgebung eines Roboters in geeigneter Form aufzunehmen und zu speichern, so dass eine Auswertung möglich ist, die das Laufverhalten der Maschine deutlich verbessert.

Das vorgestellte Modell verwaltet Kartierungen der Umwelt in verschiedenen Abstraktionsstufen, indem es aus einer scrollenden 3D-Urkarte globale 2D-Karten und lokale Abbildungen der Fußumgebungen erzeugt und Detailinformation vor allem für das unmittelbare Umfeld der Maschine speichert. Damit kann für jede aus dem Weltmodell erwachsende Aufgabe eine dafür geeignete Karte bereit gestellt werden. Insbesondere hält dies den Speicherverbrauch in einem Rahmen, der es ermöglicht, auch sehr große Umgebungen zu modellieren. Durch die gleiche mehrschichtige Architektur für alle Arten von Karten konnte Unabhängigkeit von der konkreten Implementierung erreicht werden, da einzelne Komponenten durch feste Schnittstellen leicht austauschbar sind. Zudem ermöglicht die strikte Trennung zwischen maschinenspezifischen und -unspezifischen Elementen eine leichte Portierung des Weltmodells auf andere Roboterarchitekturen.

Bei der Wahl einer Repräsentation zeigte sich, dass geometrische Karten die gestellten Anforderungen am besten zu erfüllen vermögen. Vor allem die Robustheit gegenüber schwacher und lückenhafter Sensorik spricht hierbei für eine Realisierung mit einem Belegtheitsgitter. Um der großen Flexibilität der Navigation von Laufmaschinen gerecht zu werden, ist es notwendig, dass dieses dreidimensional ausgeprägt ist.

Mit dem Erweiterten Inferenzgitter wurde eine Variante des Belegtheitsgitters eingeführt, die die separate Speicherung von Belegtheitswahrscheinlichkeit und eines Zuverlässigkeitsmaßes für jede Zelle durchführt und damit eine flexiblere Auswertung des erworbenen Wissens ermöglicht. Zudem wird diese zusätzliche Information auch in dem auf Fuzzy-Logik basierenden Verfahren zur Zellinhaltsmodifikation berücksichtigt. Mit diesem konnte erreicht werden, dass die Sensordaten mit hoher Qualität mit den bereits vorhandenen Informationen abgeglichen werden und gute, weiterverarbeitbare Karten erzeugen. Zur konkreten Speicherung der Daten erwiesen sich Octrees und Quadtrees als speicherschonende und ausreichend performante Datenstrukturen.

Die generierten globalen Karten weisen eine hohe Detailgenauigkeit auf und sind für weitere Auswertungen einsetzbar. Die Erstellung lokaler Fußaufsetzkarten ermög-

licht es der Steuerung auf effiziente Weise, vor jedem Schritt eine geeignete Fußaufsetzposition zu bestimmen. So kann das Laufverhalten der Maschine erstmals dahin gehend beinflusst werden, dass Hindernisse im Vorfeld erkannt und kollisionsfrei bewältigt werden können. Eine deutliche Verbesserung des Laufverhaltens des Roboters LAURON durch die Verwendung der Weltmodellinformationen konnte im Experiment nachgewiesen werden.

Auf der Grundlage der hier vorgestellten Ergebnisse sollte nun weiter gearbeitet werden, um die in den Tests noch aufgetretenen Schwächen zu beheben. Dringlichste Aufgabe ist hierbei die Verbesserung der Kinematik des neu installierten Roboterkopfes und eine sorgfältige Kalibrierung des Lasersensors. Zudem muss unbedingt eine verlässliche Odometrie implementiert werden, die insbesondere auch Drehungen um die z-Achse registriert. Weiterhin sollten die Kartierungsfähigkeiten danach auch in unstrukturiertem Gelände getestet und die Fuzzy-Regelsätze und Parameter daran angepasst werden. Eine weitere Aufgabe liegt darin, zusätzlich zu den bisherigen Sensordaten auch Distanzinformationen in das Modell zu integrieren, die durch die Auswertung von Stereo-Kamerabildern gewonnen werden.

Die in dieser Arbeit verwendete Anbindung an die Steuerung von LAURON konnte zwar aufzeigen, dass eine Verbesserung des Laufverhaltens realisiert werden kann, ist aber für weiter gehende Aufgaben nicht geeignet. Vielmehr müsste die Steuerung derart neu gestaltet werden, dass nicht nur die Bewegung jedes Beines einzeln, sondern die aller Beine auf der Grundlage des Weltmodells gleichzeitig betrachtet wird. So sollte auch nicht nur der nächste, sondern zumindest die nächsten zwei oder drei Schritte in eine mittelfristige Planung mit einbezogen werden. Die Bewegungen der Beine müssen im Kontext der Gesamtbewegung der Maschine betrachtet und untereinander koordiniert werden, um einen flüssigen und vorausschauenden Gang zu erreichen. Die Steuerungsarchitektur des ebenfalls am IDS entwickelten Vierbeiners BISAM (Albiez et al., 2002) bietet für diesen Zweck bereits eine gute Grundlage. Eine Anbindung dieser Steuerung an die vorliegende Umweltmodellierung wäre daher ein weiteres Ziel und auf Grund des maschinenunabhängigen Entwurfs des Modells auch ohne allzu große Anstrengungen realisierbar.

Die generierten globalen Karten weisen eine Qualität auf, die es ermöglicht, mit ihrer Hilfe weiter gehende Aufgaben anzugehen. Sinnvoll wäre es hier, auf ihrer Grundlage ein Routenplanungsverfahren umzusetzen. Eine Möglichkeit wäre, den Ansatz von Bai und Low (2002) zu verfolgen (vgl. auch Abschnitt 3.2.7). Eine andere Möglichkeit, die bereits in Abschnitt 3.1.3 angesprochen wurde, ist die Erzeugung topologischer Karten aus den vorliegenden geometrischen Informationen (Simhons und Dudek, 1998), auf denen sehr elegante Pfadplanungsverfahren umgesetzt werden können (siehe bspw. Castejón et al. (2001)). Zudem sollte versucht werden, Verfahren zu implementieren, um die Position der Maschine im Weltmodell zu korrigieren und so Schwächen der Odometrie zu beheben (siehe bspw. Burgard et al. (1998)). All diese Ansätze sollten dazu führen, dass sich der Roboter auch in weitläufigen Umgebungen für einen längeren Zeitraum souverän und vorausschauend bewegen kann.

Langfristig soll dies erreicht, dass sich die Maschine auch tage- oder wochenlang ohne jeden Einfluss von außen in ihrer Umwelt zurechtfindet, deren Ausprägungen in die eigenen Entscheidungen einbezieht und auch mit kurzfristig auftauchenden Umwelteinflüssen wie bewegten Objekten problemlos interagieren kann.

Anhang A

Technische Daten des Distanzsensors

Bei dem auf LAURON montierten Abstandssensor handelt es sich um den Laser-Distanz-Sensor ODS 96 der Firma *Leuze electronic* (Abbildung A.1). Im Folgenden sind dessen technische Daten laut Datenblatt in Auszügen wiedergegeben.

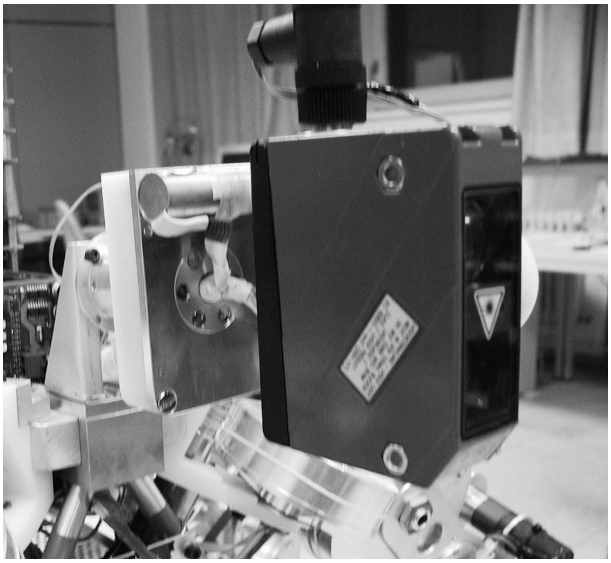


Abbildung A.1: Der auf LAURON montierte Abstandssensor ODS 96

Optische Daten	
Messbereich	200 ... 5000 mm
Auflösung	$\leq$ 5 mm bis 2000 mm
	$\leq$ 10 mm bis 3000 mm
	$\leq$ 30 mm bis 5000 mm
Lichtquelle	Laser (Wechsellicht)
Wellenlänge	660 nm (sichtbares Rotlicht)
Lichtfleckdurchmesser	divergent, $3 \times 8\,\text{mm}^2$ bei 120 mm (Laser)

Fehlergrenzen

Absolutmessgenauigkeit	bei 5 m ± 10 %
	bei 3 m ± 5 %
Wiederholgenauigkeit	bei 5 m ± 5 %
	bei 3 m ± 2,5 %

Zeitverhalten

Schaltfrequenz	10 ... 100 Hz
Ansprechzeit	≤ 100 ms
Bereitschaftsverzögerung	≤ 300 ms

Elektrische Daten

Betriebsspannung U_B	18 ... 30 VDC (inkl. Restwelligkeit)
Restwelligkeit	≤ 15% von U_B
Leerlaufstrom	≤ 150 mA
Schaltausgang	PNP-Transistor, high-aktiv
Signalspannung high/low	≥ (U_B − 2 V) / ≤ 2 V
Analogausgang	R_L ≥ 2 kΩ (Spannung)
	R_L ≤ 500 kΩ (Strom)

Mechanische Daten

Gehäuse	Zink-Druckguss
Optikabdeckung	Glas
Gewicht	380 g

Anhang B

Details zur Implementierung

In diesem Anhang findet sich zunächst eine Übersicht über die entstandenen Klassen sowie der Fuzzy-Regelsatz für die Einfüge-Bewertung der Sensordaten.

B.1 Dokumentation der Programmklassen

Dieser Abschnitt enthält die mit dem Dokumentierungswerkzeug*Doxygen* erzeugte Übersicht über die wichtigsten der erstellten Programmklassen. Dabei repräsentiert *t3DLocalEnvironment* die Umgebungsebene, *3DLocalGrid* die Gitterebene und *tOctree* die Speicherebene. Die mit *Viz* beginnenden Klassen enthalten die Visualierungsmechanismen, die mit *Lauron* beginnenden die maschinenspezifischen Codestücke. *SenseEnvironment* ist das MCA-Modul für das Weltmodel.

B.2 Klassenübersicht

B.2.1 t2DFootEnvironment Class Reference

Handles the concrete environment of a foot position.

Inheritance diagram for t2DFootEnvironment:

Collaboration diagram for t2DFootEnvironment:

Public Methods

- **t2DFootEnvironment** (**t2DLocalMap** ∗local_map)

Contructor.

- ~**t2DFootEnvironment** (void)

  Destructor.

- tVec3 **FindGoodFootLocation** (void)

  Returns a proposal for next foot position.

- float **GetMapCredibility** (void)

  Returns the credibility rating of the whole map.

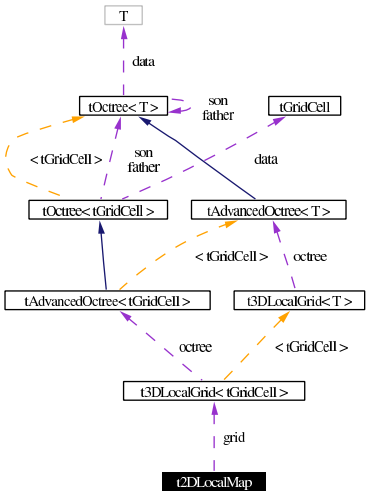
- float **GetMaximumCredibility** (void)

  Returns the maximum credibility value of a map.

B.2.2 t2DLocalMap Class Reference

A twodimensional environment map.

Collaboration diagram for t2DLocalMap:



Public Methods

- **t2DLocalMap** (void)

  Constructor.

- ~**t2DLocalMap** (void)

  Destructor.

- void **Init** (float x, float y, float dx, float dy, float granu_x, float granu_y)

  Initializes the map.

- void **SetPoint** (float x, float y, float value, float cred)

  Sets a point in the map to a specified value.

- float **GetHeight** (float x, float y)

  Returns the height value of a given position (x,y).

- float **GetCredibility** (float x, float y)

  Returns the credibility of a given position.

- tVec2 **GetMinima** (void)

  Returns the minima.

- tVec2 **GetMaxima** (void)

  Returns the maxima.

- tVec2 **GetGranularity** (void)

  Returns the granularity.

- bool **DataExistent** (float x, float y)

  Tells if data is existent.

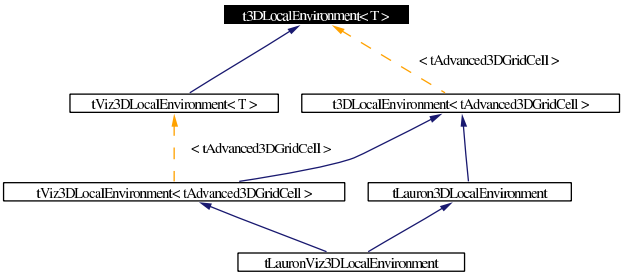
- void **SaveWholeMap** (char *filename, int mode=0)

  Saves the map to a file.

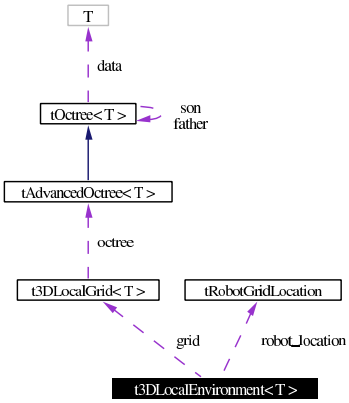
B.2.3 t3DLocalEnvironment< T > Class Template Reference

Basis class for 3D Local Environment Models.

Inheritance diagram for t3DLocalEnvironment< T >:



Collaboration diagram for t3DLocalEnvironment< T >:



Public Methods

- **t3DLocalEnvironment** (void)

  Constructor.

- virtual ~**t3DLocalEnvironment** (void)

  Destructor.

- void **SetEnvironmentSize** (tVec3 min, tVec3 max, tVec3 granu)

  Sets size of the environment.

- tVec3 **GetGranularity** (void)

  Returns the granularity information.

- tVec3 **GetMinima** (void)

  Returns the minima of the environment.

- tVec3 **GetMaxima** (void)

  Returns the maxima of the environment.

- float **GetNeighbourhoodRating** (float x, float y, float z, float sollwert)

  Returns a rating of the neighbourhood relationship.

- void **SetLine** (tVec3 start, tVec3 end, int sensor_nr=0, float occupied=1)

  Sets a line of points to the value of "occupied".

- void **UnsetLine** (tVec3 start, tVec3 end, int sensor_nr=0)

Sets a line of points to 0.

- void **GenerateLocalMap** (**t2DLocalMap** *map, float x, float y, float z, float dx, float dy, float dz)

  Generates a local 2D map.

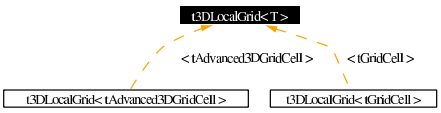
- virtual void **SetPoint** (float x, float y, float z, int sensor_nr=0, float occupied=1)

  Virtual member, must be defined in inherited class.

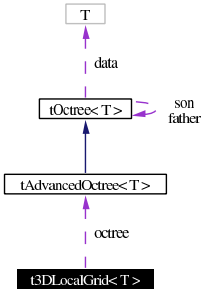
B.2.4 t3DLocalGrid< T > Class Template Reference

A 3-dimensional grid basing on an octree.

Inheritance diagram for t3DLocalGrid< T >:



Collaboration diagram for t3DLocalGrid< T >:



Public Methods

- **t3DLocalGrid** (void)

  Constructor.

- **~t3DLocalGrid** (void)

  Destructor.

- void **Init** (vector< float > min, vector< float > max, vector< float > granu)

  Initializes the grid and creates the octree.

- void **Reset** (void)

  Destroy and reinitializes the grid, which means: the octree.

- vector< float > **GetMinima** (void)

  Returns the minima.

- vector< float > **GetMaxima** (void)

  Returns the maxima.

- vector< float > **GetGranularity** (void)

  Returns the granularity.

- void **Enhance** (void)

  Enhances the octree (doubles its size).

- void **SetPoint** (float x, float y, float z, T object)

  Inserts a single point into the grid.

- T **GetPoint** (float x, float y, float z)

  Returns the data stored in a single point.

- void **DeletePoint** (float x, float y, float z)

  Deletes a point (cell) from the grid.

- bool **DataExistent** (float x, float y, float z)

  Checks if data is existent.

- tVec3 **Neighbour** (float x, float y, float z, int dx, int dy, int dz)

  The coordintates of a neighboar cell.

- queue< **tGridDataContainer**< T > > **GetLastChanges** (int max_queue_size=100000, bool force_clear_changes=false)

  Returns all lately changed data.

- void **ClearChanges** (void)

  Clears the set of last recent changes.

- void **AllDataToChanges** (void)

  Copies all octree data to the set of changes.

- queue< **tGridDataContainer**< T > > **FindLine** (tVec3 start, tVec3 end)

  Returns a queue containing all cells on a line.

- int **Size** (void)

  Returns the size of the grid.

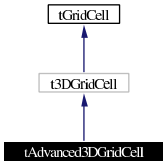
- void **GatherChanges** (bool do_it=true)

  Tells the grid to gather changes or not.

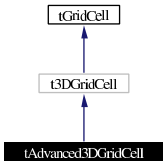
B.2.5 tAdvanced3DGridCell Class Reference

A grid cell for environment modelling with more features.

Inheritance diagram for tAdvanced3DGridCell:



Collaboration diagram for tAdvanced3DGridCell:



Public Methods

- **tAdvanced3DGridCell** (int _max_sensors=16)

  Constructor.

- ~**tAdvanced3DGridCell** (void)

  Destructor.

- void **IncreaseMeasurements** (void)

  Increases the number of measurments.

- int **GetNumberOfMeasurements** (void)

Returns the number of measurements.

- void **AdaptSensorDensity** (int sensor_nr)

  Adapt the sensor density.

- int **GetSensorDensity** (void)

  Returns sensor density.

B.2.6 tFuzzyCellRating Class Reference

A decision package to calculate parameters for using the environment model.

Public Methods

- **tFuzzyCellRating** (int in, int out, int sensor_dim, float *sensor_creds)

  Constructor.

- **~tFuzzyCellRating** (void)

  Destructor.

- void **ReadRules** (char *filename)

  Read the rule set from a file.

- void **InputCredibility** (float _cred)

  Sets the (prior) credibility which comes as an input.

- void **InputSensorDensity** (int _density)

  Sets sensor density information as an input.

- void **InputDataAge** (long _age)

  Sets the timestamp as an input.

- void **InputNumberOfMeasurements** (unsigned int _measurements)

  Inputs the number of prior measurements.

- void **InputMachineStatus** (float _machine_status)

  Sets the machine movement status as input.

- void **InputSensorCredibility** (float _sensor_cred)

  Sets the sensor credibility as input.

- void **InputNeighbourhood** (float _neighbourhood)

  Sets a neighbourhood value (horizontal) as an input.

- void **InputSensorNr** (int _sensor_nr)

  Sets a sensor credibility according to sensor number.

- void **CalculateValues** (void)

  Take all input values and calculate proposals.

- void **ProcessGridCell** (**tAdvanced3DGridCell** cell)

  Processes the whole data of a grid cell to the inputs.

- float **ProposeCredibility** (void)

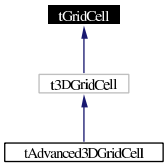
  Returns the proposal for credibility of the current insertion process.

- float **ProposeInfluence** (void)

  Returns the proposed influence of this measurement.

B.2.7 tGridCell Class Reference

A class representing one grid cell of the 3D world.
Inheritance diagram for tGridCell:



Public Methods

- **tGridCell** (void)

  The Constructor.

- **~tGridCell** (void)

  The Destructor.

- float **GetOccupation** (void)

  Returns the occupancy status.

- bool **IsOccupied** (void)

  Tells if there is something or not.

- void **SetOccupation** (float prob)

  Sets an occupation probability.

- void **SetCredibility** (float cred)

  Sets the overall credibility of this data set.

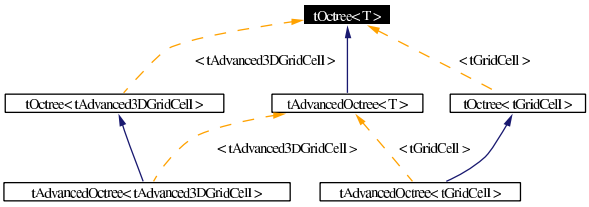
- float **GetCredibility** (void)

  Returns overall credibility.

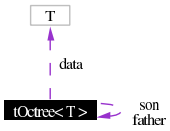
B.2.8 tOctree< T > Class Template Reference

An Octree.

Inheritance diagram for tOctree< T >:



Collaboration diagram for tOctree< T >:



Public Methods

- **tOctree** (void)

  The Constructor.

- **~tOctree** (void)

  The Destructor.

- void **DestroyTree** (void)

Destroy the whole tree.

- void **Init** (int size)

  Initializes a new octree from scratch.

- int **GetAxisMax** (int ax) throw (tXOctreeUninitialized)

  Returns the maximum coordinate for all axis'.

- int **GetAxisMin** (int ax) throw (tXOctreeUninitialized)

  Returns the minimum coordinate for all axis'.

- bool **Initialized** (void)

  Returns if the octree is initialized or not.

- void **InsertCell** (int x, int y, int z, T object) throw (tXOctreeUninitialized,tXOutOfBoundaries)

  Inserts an object into the tree at position (x,y,z).

- void **DeleteCell** (int x, int y, int z) throw (tXOctreeUninitialized, tXDataUndefined)

  Removes a grid cell from the octree.

- T **GetData** (int x, int y, int z) throw (tXOctreeUninitialized,tXDataUndefined)

  Returns the data stored in a cell.

- bool **DataExistent** (int x, int y, int z) throw (tXOutOfBoundaries)

  Checks if there is data in (x,y,z).

- void **Enhance** (void)

  Enhances the octree by doubling each axis.

- void **Scroll** (unsigned short direction)

  Scrolls the octree data.

- int **Size** (int param=1)

  Returns the size of the octree.

- set< vector< int > > **AllDataPositions** (void)

  Returns a set consisting all octree data.

- void **OutputTree** (char *filename)

Prints the whole tree to a file.

- void **OutputData** (char *filename)

  Prints data to a file.

B.2.9 tRobotGridLocation Class Reference

A rudimentary class to store information about the robot's location.

Public Methods

- **tRobotGridLocation** (void)

  Constructor.

- **~tRobotGridLocation** (void)

  Destructor.

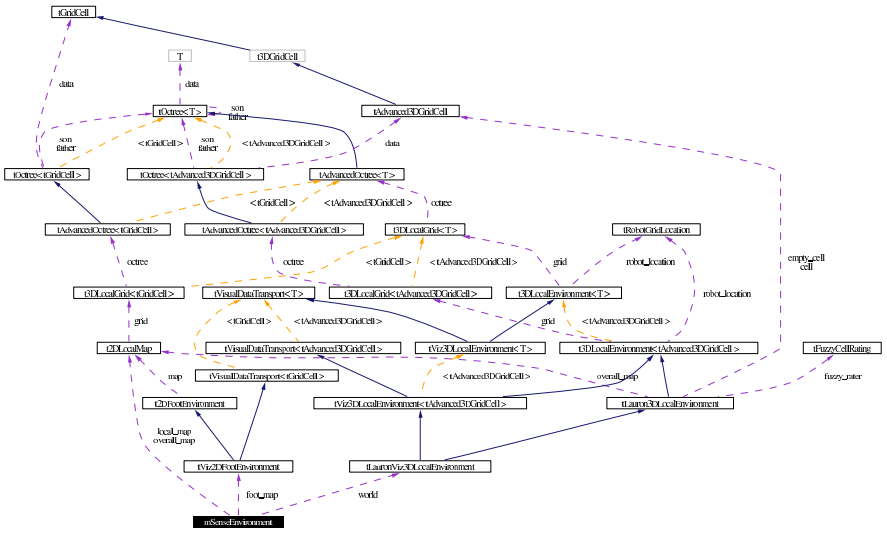
- float **GetPosition** (**tAxis::tAx** axis)

  Returns robot location of one axis.

B.2.10 mSenseEnvironment Class Reference

Inserts sensor data into local 3D environment model.

Collaboration diagram for mSenseEnvironment:



Public Methods

- **mSenseEnvironment** (tParent *parent, tDescription name="SenseEnvironment")

  Constructor.

- ~**mSenseEnvironment** ()

  Destructor.

- void **Control** ()

  Conrol loop.

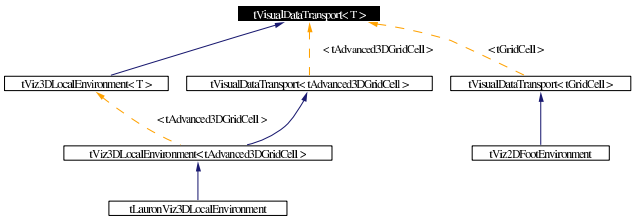
- void **Sense** ()

  Sense loop.

B.2.11 tVisualDataTransport< T > Class Template Reference

Visualization data transport mechanisms for environment models.

Inheritance diagram for tVisualDataTransport< T >:



Public Methods

- **tVisualDataTransport** (tModule *_module, char *name, unsigned structs_per_line, unsigned number_of_lines)

  Constructor.

- virtual ~**tVisualDataTransport** (void)

  Destructor.

- short int **DataToGUI** (**t3DLocalGrid**< T > *grid)

  Transports last changes in the environment towards the GUI.

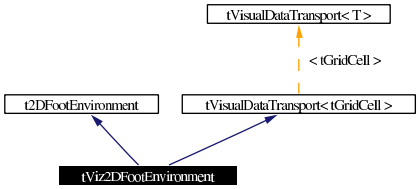
- void **TransferContextInformation** (tVec3 minima, tVec3 maxima, tVec3 granularity)

Sends a graphical context to a blackboard.

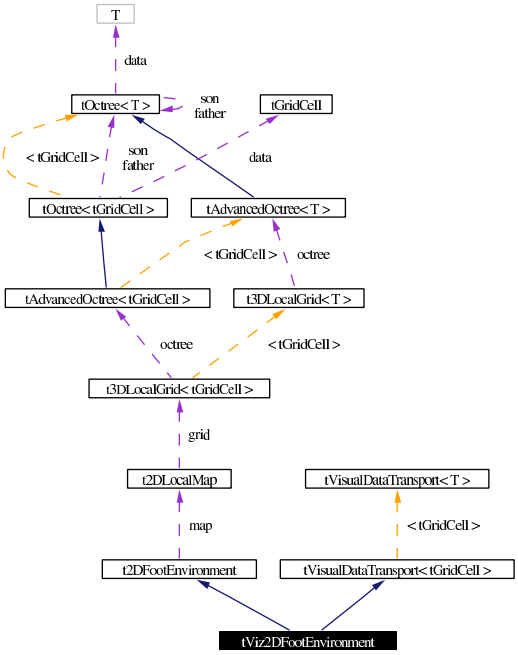
B.2.12 tViz2DFootEnvironment Class Reference

Visualization extension to the 2D Foot Environment.

Inheritance diagram for tViz2DFootEnvironment:



Collaboration diagram for tViz2DFootEnvironment:



Public Methods

- **tViz2DFootEnvironment** (tModule *module, **t2DLocalMap** *local_map, char *name, unsigned structs_per_line, unsigned number_of_lines)

  Constructor.

- ~**tViz2DFootEnvironment** (void)

Destructor.

- t3DVisualizationData **ProcessVisualData** (tGridDataContainer< **tGridCell** > env_data)

  Transforms grid data to visualization data.

- short int **WriteChangedDataToGUI** (void)

  Transports last changes in the environment towards the GUI.

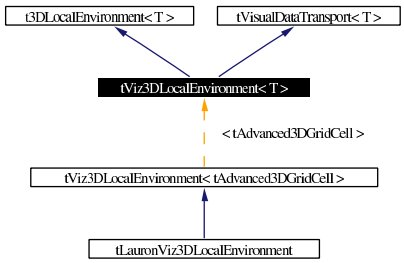
- void **WriteContext** (void)

  Forces writing context information to GUI.

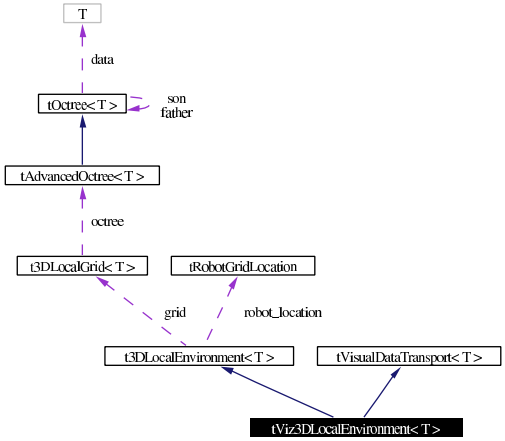
B.2.13 tViz3DLocalEnvironment< T > Class Template Reference

Visualization extension for the 3D environment model.

Inheritance diagram for tViz3DLocalEnvironment< T >:



Collaboration diagram for tViz3DLocalEnvironment< T >:



Public Methods

- **tViz3DLocalEnvironment** (tModule *module, char *name, unsigned structs_per_line, unsigned number_of_lines)

  Constructor.

- **~tViz3DLocalEnvironment** (void)

  Destructor.

- short int **WriteChangedDataToGUI** (void)

  Transports last changes in the environment towards the GUI.

- void **EnableVisualization** (bool viz_on=true)

  Enables or disables visualization.

- void **WriteContext** (void)

  Sends a graphical context to a blackboard.

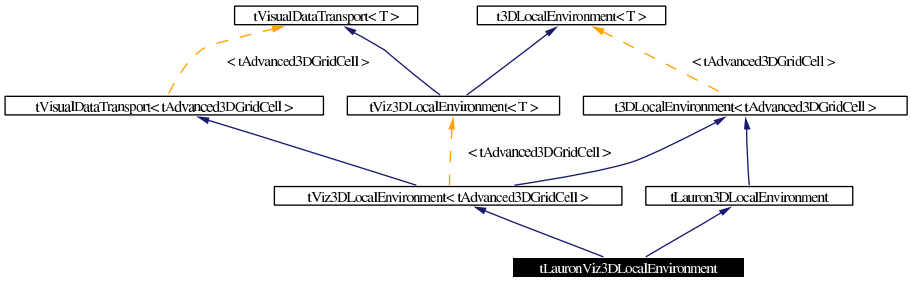
- void **RedrawWorld** (void)

  Forces the grid to send all stored data for performing a redraw.

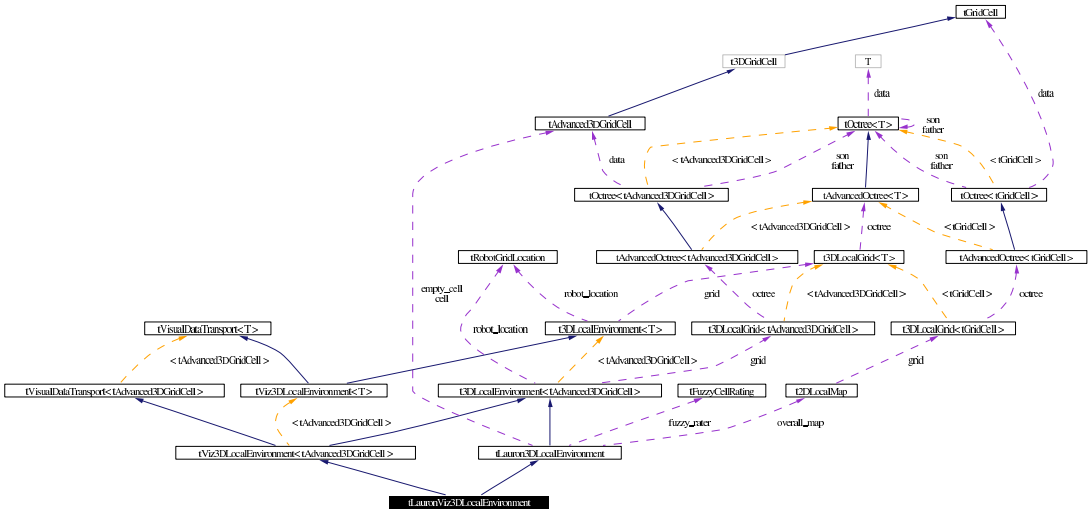
B.2.14 tLauronViz3DLocalEnvironment Class Reference

World model for Lauron including visualization.

Inheritance diagram for tLauronViz3DLocalEnvironment:



Collaboration diagram for tLauronViz3DLocalEnvironment:



Public Methods

- **tLauronViz3DLocalEnvironment** (tModule ∗module, char ∗name, unsigned structs_per_line, unsigned number_of_lines)

  Constructor.

- **∼tLauronViz3DLocalEnvironment** (void)

  Destructor.

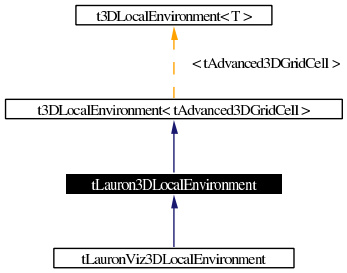
- t3DVisualizationData **ProcessVisualData** (tGridDataContainer< **tAdvanced3DGridCell** >)

  Transforms grid data to visualization data.

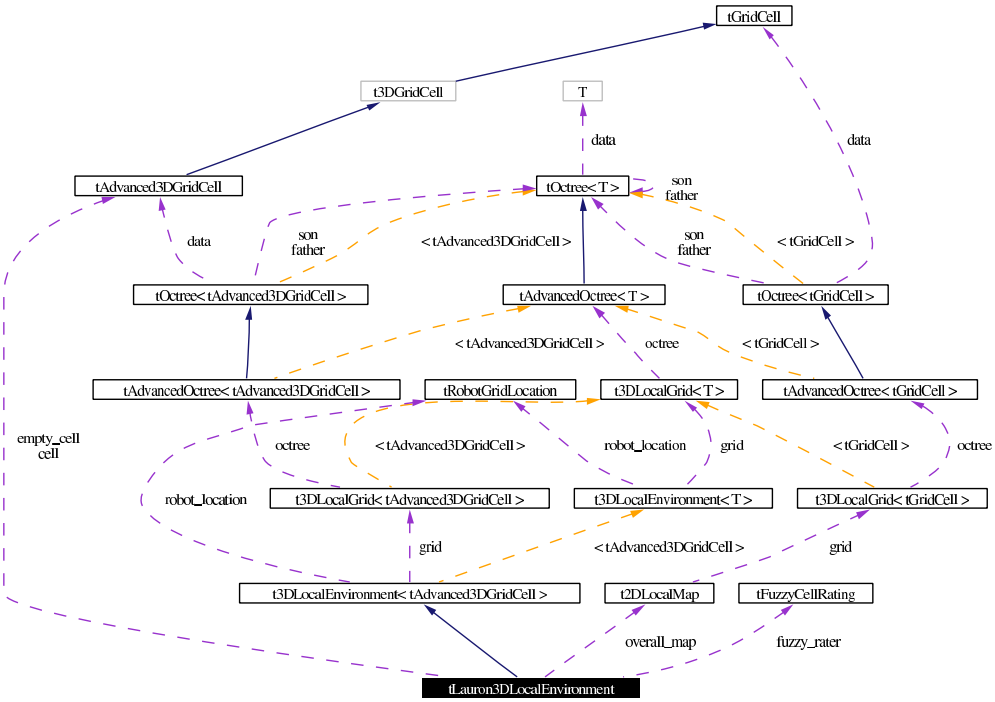
B.2.15 tLauron3DLocalEnvironment Class Reference

World model for Lauron.

Inheritance diagram for tLauron3DLocalEnvironment:



Collaboration diagram for tLauron3DLocalEnvironment:



Public Types

- enum { **eSIMPLE_INCREASE**, **eFUZZY_JUDGING** }

  Enumeration naming the various point insertion variants.

Public Methods

- **tLauron3DLocalEnvironment** (void)

  Constructor.

- **~tLauron3DLocalEnvironment** (void)

  Destructor.

- void **tLauron3DLocalEnvironment::SetPoint** (float x, float y, float z, int sensor_nr=0, float occupied=1)

  Handles point insertion.

- void **UnsetPoint** (float x, float y, float z, int sensor_nr)

  Handles point insertion (unsetting points),

See also:
SetPoint (p. 95).

- void **SaveMap** (void)

B.3 Regelsatz für die Punkt-Einfüge-Bewertung

Im Folgenden findet sich der Regelsatz, wie er in den Tests zu der vorliegenden Arbeit verwendet wurde. Aus dieser Form können die vom Programm einlesbaren Regeldateien erstellt werden (vgl. Abschnitt 4.6.6).

```
IN 6
OUT 2

LING_VAR Measurements 0 10000
FUZZY_SET MVeryFew EDGE_LEFT 0 40
FUZZY_SET MFew TRIANGLE 20 60
FUZZY_SET MMedium TRIANGLE 50 120
FUZZY_SET MMore TRIANGLE 120 120
FUZZY_SET MMany EDGE_RIGHT 180 120

LING_VAR CredIn 0 1
FUZZY_SET CIVeryLow EDGE_LEFT 0.05 0.2
FUZZY_SET CILow TRIANGLE 0.1 0.4
FUZZY_SET CIMedium TRIANGLE 0.4 0.4
FUZZY_SET CIHigh TRIANGLE 0.6 0.4
FUZZY_SET CIVeryHigh EDGE_RIGHT 0.8 0.4

LING_VAR Age 0 1000000
FUZZY_SET AVeryNew EDGE_LEFT 500 1000
FUZZY_SET ANew TRIANGLE 2000 4000
FUZZY_SET AMedium TRIANGLE 8000 5000
FUZZY_SET AOld TRIANGLE 15000 15000
FUZZY_SET AVeryOld TRIANGLE 35000 50000
FUZZY_SET AOutdated EDGE_RIGHT 60000 50000

LING_VAR SensorCred 0 1
FUZZY_SET SCLow EDGE_LEFT 0.2 0.6
FUZZY_SET SCMedium TRIANGLE 0.5 0.6
FUZZY_SET SCHigh EDGE_RIGHT 0.8 0.6

LING_VAR Neighbourhood 0 1
FUZZY_SET NVeryLow EDGE_LEFT 0.05 0.2
FUZZY_SET NLow TRIANGLE 0.15 0.4
FUZZY_SET NMedium TRIANGLE 0.3 0.5
FUZZY_SET NHigh EDGE_RIGHT 0.6 0.4

LING_VAR SensorDens 0 7
FUZZY_SET SDLow EDGE_LEFT 1 2
FUZZY_SET SDMedium TRIANGLE 2 4
```

```
FUZZY_SET SDHigh EDGE_RIGHT 4 4

LING_VAR CredOut 0 1
FUZZY_SET CVeryLow EDGE_LEFT 0.1 0.4
FUZZY_SET CLow TRIANGLE 0.3 0.4
FUZZY_SET CMedium TRIANGLE 0.5 0.4
FUZZY_SET CHigh TRIANGLE 0.7 0.4
FUZZY_SET CVeryHigh EDGE_RIGHT 0.9 0.4

LING_VAR Influence 0 1
FUZZY_SET IVeryLow EDGE_LEFT 0.05 0.2
FUZZY_SET ILow TRIANGLE 0.2 0.4
FUZZY_SET IMedium TRIANGLE 0.4 0.4
FUZZY_SET IHigh TRIANGLE 0.6 0.4
FUZZY_SET IVeryHigh EDGE_RIGHT 0.8 0.4



RULES

### INFLUENCE RULES

IF Measurements = MMany THEN Influence = IVeryLow
IF Measurements = MMore THEN Influence = ILow
IF Measurements = MMedium THEN Influence = IMedium
IF Measurements = MFew THEN Influence = IHigh
IF Measurements = MVeryFew THEN Influence = IVeryHigh

IF CredIn = CIVeryLow THEN Influence = IHigh
IF CredIn = CILow THEN Influence = IMedium
IF CredIn = CIMedium THEN Influence = ILow
IF CredIn = CIHigh THEN Influence = IVeryLow
IF CredIn = CIVeryHigh THEN Influence = IVeryLow

IF Age = AVeryNew THEN Influence = IVeryLow
IF Age = ANew THEN Influence = ILow
IF Age = AOld THEN Influence = IMedium
IF Age = AMedium THEN Influence = ILow
IF Age = AVeryOld THEN Influence = IHigh
IF Age = AOutdated THEN Influence = IVeryHigh

IF SensorCred = SCLow THEN Influence = IVeryLow
IF SensorCred = SCMedium THEN Influence = IMedium
IF SensorCred = SCHigh THEN Influence = IHigh
```

```
IF Measurements = MMany AND Age = AVeryOld THEN Influence = IHigh
IF Measurements = MMany AND Age = AOutdated THEN Influence = IVeryHigh

IF Measurements = MMany AND CredIn = CILow THEN Influence = IVeryHigh


### CREDIBILITY RULES

IF Neighbourhood = NVeryLow THEN CredOut = CVeryLow
IF Neighbourhood = NLow THEN CredOut = CLow
IF Neighbourhood = NMedium THEN CredOut = CHigh
IF Neighbourhood = NHigh THEN CredOut = CVeryHigh

IF SensorCred = SCLow THEN CredOut = CLow
IF SensorCred = SCMedium THEN CredOut = CMedium
IF SensorCred = SCHigh THEN CredOut = CVeryHigh

IF SensorDens = SDLow THEN CredOut = CMedium
IF SensorDens = SDMedium THEN CredOut = CHigh
IF SensorDens = SDHigh THEN CredOut = CVeryHigh

IF Measurements = MVeryFew THEN CredOut = CVeryLow
IF Measurements = MFew THEN CredOut = CLow
IF Measurements = MMedium THEN CredOut = CMedium
IF Measurements = MMore THEN CredOut = CHigh
IF Measurements = MMany THEN CredOut = CVeryHigh
```

Literaturverzeichnis

[Albiez et al. 2002] Albiez, Jan ; Luksch, Tobias ; Berns, Karsten ; Dillmann, Rüdiger: An Activation Based Behaviour Control Architecture for Walking Machines. In: *Proceedings of 7th International Conference on Simulation of Adaptive Behaviour (SAB), Edinburgh*, 2002, S. 118–121

[Arbuckle et al. 2002] Arbuckle, Daniel ; Howard, Andrew ; Matarić, Maja J.: Temporal Occupancy Grids: a Method for Classifying the Spatio-Temporal Properties of the Environment. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* EPFL Switzerland (Veranst.), September 2002

[Arleo et al. 1999] Arleo, Angelo ; R. Millán, José del ; Floreano, Dario: Efficient Learning of Variable-Resolution Cognitive Maps for Autonomous Indoor Navigation. In: *IEEE Transactions on Robotics and Automation* 15 (1999), Dezember, Nr. 6

[Bai und Low 2002] Bai, Shaoping ; Low, K. H.: Path Generation of Walking Machines in 3D Terrain. In: *Proceedings of the IEEE Conference on Robotics and Automation (ICRA)*, 2002. – to be published

[Berns 1997/98] Berns, Karsten: *Grundlagen für die Steuerung von Laufmaschinen.* Skriptum zur Vorlesung. Wintersemester 1997/98. – Universität Karlsruhe

[Berns 2000] Berns, Karsten: *The Walking Machine Catalogue.* 2000. – `http://www.fzi.de/WMC.html`

[Berns und Fiegert 1991] Berns, Karsten ; Fiegert, M.: Laufmaschinen – von der Fiktion zur Realität. In: *Technische Rundschau Schweiz* 46 (1991), S. 4–51

[Biewer 1997] Biewer, Benno: *Fuzzy-Methoden.* Springer-Verlag Berlin Heidelberg, 1997

[Booch 1991] Booch, Grady: *Object Oriented Design.* Benjamin/Cummings Pub. Co., 1991 (The Benjamin/Cummings series in Ada and software engineering)

[Borenstein et al. 1996] Borenstein, Johann ; Everett, H. R. ; Feng, Liqiang: *Where am I? Sensors and Mehtods for Mobile Robot Positioning.* University of Michigan, April 1996

[Borenstein und Koren 1991] Borenstein, Johann ; Koren, Yoram: The Vector Field Histogram – Fast Obstacle Avoidance for Mobile Robots. In: *IEEE Transactions on Robotics and Automation* 7 (1991), Juni, Nr. 3, S. 278–288

[Brandl 1999] Brandl, Florian: *Entwurf einer modularen Steuerung für eine sechsbeinige Laufmaschine*. Gruppe Interaktive Diagnose und Service-Systeme, Forschungszentrum Informatik, Universität Karlsruhe, Studienarbeit, November 1999

[Burgard et al. 1998] Burgard, Wolfram ; Derr, Andreas ; Fox, Dieter ; Cremers, Armin B.: Integrating Global Position Estimation and Position Tracking for Mobile Robots: The Dynamic Markov Localization Approach. In: *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems, Victoria B. C., Canada*, Oktober 1998

[Carterette und Friedman 1978] Carterette, Edward C. (Hrsg.) ; Friedman, Morten E. (Hrsg.): *Handbook of Perception*. Bd. 1. Academic Press, New York, 1978

[Castejón et al. 2001] Castejón, C. ; Moreno, L. ; Salichs, M. A.: Traversability Modeling in 3D Environments. In: *Proceedings of the International Conference on Field and Service Robotics*, 2001

[Cho et al. 2001] Cho, Hye-Kyung ; Cho, Young-Jo ; You, Bum-Jae: Integration of Schema-Based Behaviours and Variable-Resolution Cognitive Maps for Stable Indoor Navigation. In: *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, Mai 2001

[Cordes et al. 1993] Cordes, Stefan ; Berns, Karsten ; Dillmann, Rüdiger: Steuerungsarchitektur der sechsbeinigen Laufmaschine Lauron. In: Schmidt, Günther (Hrsg.): *Autonome Mobile Systeme, 9. Fachgespräch an der Technischen Universität München*. Lehrstuhl für Steuerungs- und Regelungstechnik, TU München, 1993, S. 205–213

[Crowley 1989] Crowley, J.: World Modeling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging. In: *Proceedings of IEEE International Conference on Robotics and Automation, Scottsdale, AZ*, Mai 1989, S. 674–680

[Dillmann und Huck 1991] Dillmann, Rüdiger ; Huck, Martin: *Informationsverarbeitung in der Robotik*. Springer-Verlag Berlin Heidelberg, 1991

[Elfes 1989] Elfes, Alberto: Using Occupancy Grids for Mobile Robot Perception and Navigation. In: *Computer* 22 (1989), Nr. 6, S. 46–57

[Engelson und McDermott 1992] Engelson, Sean P. ; McDermott, Drew V.: Error Correction in Mobile Robot Map Learning. In: *Proceedings of the 1992 IEEE International Conference on Robotics and Automation, Nice, France*, Mai 1992

[Gaßmann 2000] Gassmann, Bernd: *Erweiterung einer modularen Laufmaschinensteuerung für unstrukturiertes Gelände*. Gruppe Interaktive Diagnose und Service-Systeme, Forschungszentrum Informatik, Universität Karlsruhe, Diplomarbeit, September 2000

[Gaßmann et al. 2001] Gassmann, Bernd ; Scholl, Kay-Ulrich ; Berns, Karsten: Locomotion of LAURON III in Rough Terrain. In: *Proceedings of the International Conference on Advanced Mechatronics, Como, Italy*, Juli 2001

[Gierer 1989] Gierer, Alfred: Überlegungen zur Leib-Seele-Beziehung. In: Pöppel, Ernst (Hrsg.): *Gehirn und Bewusstsein*. VCH Verlagsgesellschaft mbH Weinheim, 1989

[Hoppen et al. 1990] Hoppen, P. ; Knieriemen, P. ; Puttkamer, E.: Laser-Radar Based Mapping and Navigation for an Autonomous Mobile Robot. In: *Proceedings of IEEE International Conference on Robotics and Automation, Cincinnati, OH*, Mai 1990, S. 948–953

[Jackins und Tanimoto 1980] Jackins, C. L. ; Tanimoto, S. L.: Octtrees and their Use in Representing Three-Dimensional Objects. In: *Computer Graphics and Image Processing* 14 (1980), S. 249–270

[Kahlert 1995] Kahlert, Jörg: *Fuzzy Control für Ingenieure*. Friedrich Viehweg & Sohn, Braunschweig, 1995

[Kawaguchi und Endo 1980] Kawaguchi, E. ; Endo, T.: On a Method of Binary Picture Representation and its Application to Picture Compression. In: *IEEE Transactions on Pattern Analysis an Machine Intelligence* 2 (1980), Nr. 1, S. 27–35

[Kepplin und Berns 1999] Kepplin, Volker ; Berns, Karsten: A Concept for Walking Behaviour in Rough Terrain. In: *Climbing and Walking Robots and the Support Technologies for Mobile Machines (Clawar 99)*, September 1999, S. 509–516

[Kruse et al. 1995] Kruse, E. ; Gutsche, R. ; Wahl, F. M.: Inkrementelle sensorbasierte Erzeugung eines Umweltmodells mit Hilfe von Bewertungsfunktionen im Konfigurationsraum. In: Dillmann, Rüdiger (Hrsg.) ; Rembold, Ulrich (Hrsg.) ; Lüth, Tim (Hrsg.): *Autonome Mobile Systeme 1995*. Springer Verlag Heidelberg Berlin, 1995 (Informatik aktuell), S. 120–131

[Kuipers und Byun 1991] Kuipers, B. J. ; Byun, Y.-T.: A Robot Exploration and Mapping Strategy based on a Semantic Hierarchy of Spatial Representations. In: *Journal of Robotics and Autonomous Systems* 8 (1991), S. 47–63

[Lågstad und Auran 1996] Lågstad, Petter ; Auran, Per G.: Real Time Sensor Fusion for Autonomous Underwater Imaging in 3D. In: *Proceedings of Oceans* IEEE (Veranst.), 1996

[Lallement et al. 1998] Lallement, Alex ; Siadat, Ali ; Dufaut, Michel ; Husson, René: Laser-Vision Cooperation for Map Building. In: *Proceedings of the 1998 IEEE ISC/CIRA/ISAS Joint Conference, Gaithersburg, MD*, September 1998, S. 387–392

[Luksch 2001] Luksch, Tobias: *Entwicklung einer Fuzzy-Regelbasis zur Haltungskontrolle einer vierbeinigen Laufmaschine*. Gruppe Interaktive Diagnose und Service-Systeme, Forschungszentrum Informatik, Universität Karlsruhe, Studienarbeit, Juni 2001

[Maimone et al. 1998] Maimone, M. ; Matthies, L. ; Osborn, J. ; Rollins, E. ; Teza, J. ; Thayer, S.: A Photo-Realistic 3-D Mapping System for Extreme Nuclear Environments: Chornobyl. In: *Proceedings of the 1998 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, Oktober 1998

[Minsky 1968] Minsky, Marvin L.: *Semantic Information Processing.* MIT Press, Cambridge, MA, 1968

[Moore und Atkeson 1995] Moore, Andrew W. ; Atkeson, Christopher G.: The Parti-game Algorithm for Variable Resolution Reinforcement Learning in Multidimensional State-spaces. In: *Machine Learning* 21 (1995), Dezember

[Moravec 1988] Moravec, Bruno P.: Certainty Grids for Sensor Fusion in Mobile Robots. In: *AI Magazine* 9 (1988), Nr. 2, S. 61–77

[Murphy et al. 2002] Murphy, Karl ; Abrams, Marylin ; Balakirsky, Stephen ; Chang, Tommy ; Hong, Tsai ; Lacaze, Alberto ; Legowik, Steven: Intelligent Control for Off-Road Driving. In: Nahavandi, Saeid (Hrsg.): *First International NAISO Congress on Autonomous Intelligent System* Natural and Artificial Intelligence Systems Organization (Veranst.), Februar 2002

[Penrose 1991] Penrose, Roger: *Computerdenken: die Debatte um Künstliche Intelligenz, Bewusstsein und die Gesetze der Physik.* Spektrum der Wiessenschaft Verlagsgesellschaft mbH Heidelberg, 1991

[Pöppel 1985] Pöppel, Ernst: *Grenzen des Bewusstseins: Über Wirklichkeit und Welterfahrung.* Deutsche Verlagsanstalt Stuttgart, 1985

[von Randow 1997] Randow, Gero von: Paß gut auf! In: Die Zeit (1997), 11. Juli, S. 1

[Rascke und Borenstein 1990] Rascke, Ulrich ; Borenstein, Johann: A Comparison of Grid-type Map-building techniques by Index of Performance. In: *Proceedings of IEEE International Conference on Robotics and Automation, Cincinnati, OH*, Mai 1990, S. 1021–1026

[Rencken 1993] Rencken, W. D.: Concurrent Localization and Map Building for Mobile Robots Using Ultrasonsic Sensors. In: *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robotics and Systems, Yokohama, Japan*, Juli 1993, S. 2192–2197

[Scholl et al. 2001] Scholl, Kay-Ullrich ; Albiez, Jan ; Gassmann, Bernd: MCA - An Expandable Modular Controller Architecture. In: *3rd Real-Time Linux Workshop, Milano, Italy*, 2001. – `http://mca2.sourceforge.net`

[Shatkay und Kaelbling 1997] Shatkay, Hagit ; Kaelbling, Leslie P.: Learning Topological Maps with Weak Local Odometric Information. In: *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAJ-97)* IJCAJ, Inc. (Veranst.), Morgan Kaufman Publishers, San Francisco, August 1997, S. 920–927

[Simhons und Dudek 1998] Simhons, Saul ; Dudek, Gregory: A Global Topological Map formed by Local Metric Maps. In: *International Conference on Intelligent Robots and Systems, Victoria, Canada*, 1998

[Smith et al. 1994] Smith, Andrew ; Kitamura, Yoshifumi ; Kishino, Fumio: Efficient Algorithms for Octree Motion. In: *IAPR Workshop on Machine Vision Applications, Kawasaki, Japan*, Dezember 1994, S. 172–177

[Stone 1996] Stone, Henry W.: Mars Pathfinder Microrover – A small Low-Cost, Low-Power Spacecraft. In: *Proceedings of the 1996 AIAA Forum on Advanced Development in Space Robotics*, August 1996

[Stuck et al. 1994] Stuck, Elizabeth ; Manz, Allan ; Green, David A. ; Elgazzar, Shadia: Map Updating and Path Planning for Real-time Mobile Robot Navigation. In: *Proceedings of IEEE International Conference on Intelligent Robots and Systems, Munich, Germany*, September 1994, S. 753–760

[Thrun 1998] Thrun, Sebastian: Learning Maps for Indoor Mobile Robot Navigation. In: *Artificial Intelligence* 99 (1998), S. 21–71

[Thrun et al. 1998] Thrun, Sebastian ; Gutmann, Jens-Steffen ; Fox, Dieter ; Burgard, Wolfram ; Kuipers, Benjamin J.: Integrating Topological and Metric Maps for Mobile Robot Navigation: A Statistical Approach. In: *Proceedings of the AAAI Fifteenth National Conference on Artificial Intelligence (AAAI-98), Madison, WI*, AAAI/MIT Press, 1998, S. 989–995

[Thrun und Bücken 1996] Thrun, Sebatian ; Bücken, Arno: Integrating Grid-Based and Topological Maps for Mobile Robot Navigation. In: *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96), Portland Oregon*, August 1996

[Traufetter 2002] Traufetter, Gerald: Liveshow aus Nekropolis. In: Der Spiegel (2002), Nr. 38, S. 170–172

[Turing 1950] Turing, Alan M.: Computing Machinery and Intelligence. In: *Mind* 59 (1950). – Nachdruck in: Douglas R. Hofstadter und. Daniel C. Dennett, Einsicht ins Ich, Klett-Cotta, 1986

[Weckesser und Dillmann 1997] Weckesser, Peter ; Dillmann, Rüdiger: Navigating a Mobile Service-Robot in a Natural Environment Using Sensor Fusion Techniques. In: *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems, Grenoble, France*, 1997, S. 1423–1428

[Zadeh 1965] Zadeh, Lotfi A.: Fuzzy Sets. In: *Information and Control* 8 (1965), S. 338–353. – Nachdruck in: Yager, Ovchinnikov, Tong und Nguyen, Fuzzy Sets and Applications: Selected Papers by L. A. Zadeh, Wiley, New York, 1987

[Zadeh 1973] Zadeh, Lotfi A.: Outline of a New Approach to The Analysis of Complex Systems And Decision Processes. In: *IEEE Transactions on Systems, Man,*

and Cybernetics 2 (1973), S. 28–44. – Nachdruck in: Yager, Ovchinnikov, Tong und Nguyen, Fuzzy Sets and Applications: Selected Papers by L. A. Zadeh, Wiley, New York, 1987