Wahrscheinlichkeitsbasierte Methoden zur autonomen Führung von Fahrzeugen in unsicherer Umgebung

Von der Fakultät für Maschinenbau
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades
Doktor-Ingenieur
genehmigte

Dissertation

von

Dipl.-Ing. Holger Blume

geboren am 23. November 1975 in Hildesheim

2008

Referent: Prof. Dr.-Ing. habil. Dr. h.c. Prof. E.H. Bodo Heimann
Korreferent: Prof. Dr.-Ing. Ludger Overmeyer
Tag der Promotion: 26. Mai 2008

Vorwort

Die vorliegende Dissertation entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Robotik (IfR) der Gottfried Wilhelm Leibniz Universität Hannover. Ich bedanke mich herzlich bei all denen, die zum Gelingen dieser Arbeit beigetragen haben.

An erster Stelle gilt mein Dank Prof. Dr.-Ing. habil. Dr. h.c. Bodo Heimann. Ihm danke ich für das hervorragende Arbeitsklima in seinem Institut, für die Freiheit bei der Ausgestaltung meiner Arbeit und das in mich gesetzte Vertrauen. Hervorheben möchte ich außerdem das gute persönliche Verhältnis.

Bei Prof. Dr.-Ing. Ludger Overmeyer bedanke ich mich für die Übernahme des Koreferats und für die konstruktiven Kritiken an meiner Dissertation.

Herrn Prof. Dr.-Ing. Eduard Reithmeier danke ich für die Übernahme des Vorsitzes im Promotionsverfahren und die zurückliegende Gelegenheit zur studentischen Mitarbeit in seinem Institut.

Für das angenehme Arbeitsklima, zahlreiche Diskussionen während und nach der Arbeitszeit, technische Hilfsstellungen und – nicht zuletzt – die Korrektur des Manuskripts bedanke ich mich herzlich bei allen Kollegen des IfR. Besonderer Dank gilt dabei meinem langjährigen Zimmerkollegen Houssem Abdellatif für die hervorragende Atmosphäre im Büro. Hervorheben möchte ich außerdem die zahlreichen Problemlösungen in Software-, Hardware- und Elektronikfragen durch Stephan Baron.

Einen wichtigen Beitrag haben Studenten als Hiwi und in Form von studentischen Arbeiten geleistet. Dafür bedanke ich mich bei allen, die für mich gearbeitet haben, insbesondere bei Artur Wiebe, Stefan Buhrs und Frank Abelbeck.

Besonders bedanke ich mich bei meiner Frau Tamara für ihre fortwährende Unterstützung während der Ausarbeitung der Dissertation und die gewährte Nachsicht bei Überstunden und Wochenendarbeit.

Schließlich bedanke ich mich bei meinen Eltern Jutta und Gerhard, die mir den akademischen Werdegang ermöglicht haben, für die stetige Unterstützung und das Interesse an meiner Arbeit.

Hannover, im Juni 2008 Holger Blume

Inhalt

Formelzeichen

Selten benutzte oder abweichende Bedeutungen von Formelzeichen werden im Text erläutert. Bei allen verwendeten Größen wird davon ausgegangen, dass sie in SI-Einheiten angegeben sind. Daher wird nicht weiter auf die Dimension eingegangen. Nur bei gemessenen Größen werden in Grafiken Einheiten zur Vermeidung von Missverständnissen mit angegeben.

Lateinische Notation

$\boldsymbol{A}$	Systemmatrix
A, B	Zufallsvariablen
a_i, b_i	x- und y-Koordinate des Radaufstandspunkts im roboterfesten Koordinatensystem
$\boldsymbol{B}$	Eingangsmatrix
$\mathcal{CO}_i$	Durch Hindernisse eingenommener Konfigurationsraum
$\mathcal{C}$	Untermenge des Zustandsraums, invariant bezüglich Dynamik
$\mathcal{C}_{\text{frei}}$	Hindernisfreier Konfigurationsraum
$\boldsymbol{C}$	Messmatrix
d_{Raster}	Kantenlänge einer Zelle im Raster
d_{ISM}	Als belegt geltender Bereich um eine Sensormessung im inversen Sensormodell
$\mathcal{F}$	Erreichbarer Raum als Untermenge des freien Raums
$\boldsymbol{F}$	Kraftvektor
$\boldsymbol{G}$	Jacobi-Matrix der nichtlinearen Systemfunktion
$g(\cdot)$	nichtlineare Zustandsgleichung
$\boldsymbol{H}$	Jacobi-Matrix der nichtlinearen Messgleichung
$h(\cdot)$	nichtlineare Messgleichung
$\boldsymbol{I}$	Einheitsmatrix
i, j	Laufindizes / Koordinaten im Bildkoordinatensystem
$J(\cdot, \cdot)$	Kostenfunktional
$\boldsymbol{K}$	Kalman-Verstärkung
k	Laufindex Abtastschritt
K_p, K_d	Parameter zur Einstellung der Fehlerdynamik in der Bahnregelung

$k_{\text{hit}}, k_{\text{short}}, k_{\text{max}}, k_{\text{rand}}$	Gewichtungsfaktoren der Wahrscheinlichkeitsverteilung
L	Zahl der Merkmale bei der Objektverfolgung
$l_0, l_{\text{free}}, l_{\text{occ}}$	Logarithmenverhältnis für eine unbekannte (0), freie (free), belegte (occ) Zelle
$l_k^{[i]}$	Logarithmenverhältnis der Zelle i im Zeitschritt k
$\mathcal{M}$	Endliche Untermenge der möglichen Trajektorien eines Roboters
$\boldsymbol{M}$	Wahrscheinlichkeitskarte
$\boldsymbol{M}_{\text{dyn}}$	Aus aufeinander folgenden Messdatensätzen erzeugt Karte, die veränderte Bereiche kennzeichnet
$\boldsymbol{M}_{\text{obj}}$	Aus den Minima in den Sensordaten erstellte Merkmalskarte
$\boldsymbol{M}_{\text{occ}}$	Aus aktuellen Sensordaten erstellte Belegtheitskarte
M	Partikelzahl
$\mathcal{N}(\mu, \sigma)$	Normalverteilung mit Mittelwert μ und Streuung σ
N	Zahl der Objekte bei der Objektverfolgung
$p(\cdot)$	Wahrscheinlichkeitsdichtefunktion
$p(A)$	Wahrscheinlichkeit für das Eintreten des Ereignisses A
$p(A \mid B)$	Wahrscheinlichkeit für das Eintreten von A bei gegebenem B
$p_0, p_{\text{free}}, p_{\text{occ}}$	Wahrscheinlichkeit für eine unbekannte (0), freie (free), belegte (occ) Zelle
p_{hit}	Verteilung zur Berücksichtigung der Streueigenschaften des Laserscanners
p_{max}	Verteilung zur Berücksichtigung einer Fehlmessung bei mangelnder/mehrfacher Reflektion
p_{rand}	Verteilung zur Berücksichtigung von über den Messbereich gleichverteilten Fehlmessungen
p_{short}	Verteilung zur Berücksichtigung von Messstörungen durch nicht kartierte Hindernisse
p_{ij}, q_{ij}	Element der Jacobi-Matrix der nichtlinearen Messgleichung der Roboterkinematik
$\mathcal{Q}_{\text{T}}$	Untermenge der möglichen Trajektorien eines Roboters, invariant gegenüber der Dynamik
$\boldsymbol{Q}$	Kovarianz des Messrauschens
$\boldsymbol{Q}_{\text{Pol}}$	Momentanpol
${}_{\text{b}_i}r_{\text{r}}$	Ursprungsabstand zur Beschreibung der Roboterposition im lokalen Polarkoordinatensystem des Splines i
$\mathcal{R}$	Menge der erreichbaren Ziele
$\boldsymbol{R}$	Kovarianz des Prozessrauschens

$\boldsymbol{r}$	Ortsvektor
r	Radius
r_{rand}	Zufallszahl
${}^{\text{wr}}\boldsymbol{R}$	Transformation aus Koordinatensystem r in das Koordinatensystem w
T	Abtastzeit
t	Zeit
$\mathcal{U}$	Raum der Stellgrößen eines Roboters
$\boldsymbol{u}, u$	Eingangsvektor, Eingang
$U\boldsymbol{q}$	Potentialfunktion
$\boldsymbol{v}$	Geschwindigkeitsvektor
v	Geschwindigkeit
$\hat{W}_{k,\langle n \rangle}$	Gleitender Durchschnitt über die Partikelgewichtsumme aller Partikel eines Objekts
$\mathcal{WO}_i$	Durch Hindernisse eingenommener Arbeitsraum
$w_{k,[i]}$	Partikelgewicht zum Zeitschritt k für Partikel i
$\mathcal{X}$	Zustandsraum des Roboters
$\boldsymbol{x}, x$	Zustandsvektor / Zustand
x, y, z	kartesische Koordinaten
X	Zufallsvariable
$\mathcal{Y}$	Untermenge des Zustandsraums, variant bezüglich Dynamik
$\boldsymbol{z}, z$	Messvektor, Messwert
$\boldsymbol{z}_k^{[j]*}$	Erwartungswert gebildet durch virtuelle Messgrößen

Griechische Notation

$\alpha_{(\cdot),(\cdot)}$	Parameter des Bewegungsmodells
β_{ISM}	Öffnungswinkel des Messstrahls im inversen Sensormodell
$\beta_{l,n}$	Zuordnungswahrscheinlichkeit von Merkmal l zu Objekt n
γ	Zahl der falschen Alarme bei der Objektverfolgung
$\boldsymbol{\delta}$	mittelwertfreies, normalverteiltes Messrauschen
$\boldsymbol{\epsilon}$	mittelwertfreies, normalverteiltes Prozessrauschen
η	Normalisierungskonstante
${}_{\text{b}_i}\vartheta_{\text{r}}$	Winkel zur Beschreibung der Roboterposition im lokalen Polarkoordinatensystem des Splines i
Θ	Menge aller Zuordnungsfälle des JPDAF
θ	Zuordnungsfall des JPDAF
λ_{short}	Parameter der Exponentialverteilung p_{short}
$\boldsymbol{\mu}, \mu$	Mittelwert einer normalverteilten Größe

π	Optimales Regelungsgesetz zur Überführung des Roboters aus dem Ausgangszustand in den Zielzustand
ρ	Richtung eines Messstrahls
σ^2	Varianz einer normalverteilten Größe
$\mathbf{\Sigma}$	Kovarianzmatrix einer normalverteilten Größe
τ	Winkelabweichung zwischen den Orientierung auf der idealen Kreisbahn und auf dem Polar-Spline
ϕ	Bewegungsrichtung eines Partikels
ϕ_r	Orientierung des Roboters
ϕ_m	Orientierung der Umgebungskarte
ϕ_s	Orientierung des Sensors
φ	Lenkwinkel
χ_k	Partikelwolke im Zeitschritt k
ω	Winkelgeschwindigkeit

Indizes und weitere Formelzeichen

$_\mathrm{m}(\cdot)$	Größe definiert im Karten-Koordinatensystem
$_\mathrm{p}(\cdot)$	Größe definiert im Bild-Koordinatensystem
$_\mathrm{r}(\cdot)$	Größe definiert im roboterfesten Koordinatensystem
$_\mathrm{b}(\cdot)$	Größe definiert im lokalen Koordinatensystem eines Bahnelements
$_\mathrm{s}(\cdot)$	Größe definiert im Sensor-Koordinatensystem
$_\mathrm{w}(\cdot)$	Größe definiert im ortsfesten Koordinatensystem
$_{\mathrm{wh}_i}(\cdot)$	Größe definiert im Koordinatensystem des Radmoduls i
$_\mathrm{w}(\mathrm{KS})$	Kennzeichnung des Koordinatensystems w
$(\cdot)^{[i]}$	i-tes Element des Zustandsvektors
$(\cdot)_{[m]}$	m-tes Element der Partikelwolke
$(\cdot)_{\langle n \rangle}$	n-tes Objekt in der Objektverfolgung
$(\cdot)_t$	Größe $\cdot$ zum Zeitpunkt t
$(\cdot)_k$	Größe $\cdot$ zum Zeitschritt k
$(\cdot)_\mathrm{r}$	Größe des Roboters
$(\cdot)_{\mathrm{wh}_i}$	Größe des Radmoduls i
$(\cdot)_\mathrm{soll}$	Sollgröße
$\tilde{(\cdot)}$	Schätzwert einer Größe
$(\cdot)^-$	a priori geschätzte Größe
$\dot{(\cdot)}$	Erste zeitliche Ableitung einer Größe
$\ddot{(\cdot)}$	Zweite zeitliche Ableitung einer Größe
$\varnothing$	Leere Menge

Kurzfassung

Seit Jahren wird der Servicerobotik ein beispielloser wirtschaftlicher Boom prognostiziert, der Verbreitungsgrad der künstlichen Diener ist jedoch bislang gering. Grund dafür ist vor allem die durch Komplexität und nicht deterministische Vorgänge geprägte Alltagsumgebung. In dieser Arbeit wird die solchen Umgebungen inhärente Unsicherheit im Rahmen der Navigationsaufgabe eines Roboters adressiert. Mit der wahrscheinlichkeitsbasierten Zustandsschätzung, insbesondere durch die wiederholte Anwendung sequentieller Monte-Carlo-Methoden, wird eine methodisch durchgängige Herangehensweise gewählt.

In die Thematik wird durch die Darstellung der Bayes'schen bedingten Wahrscheinlichkeit und deren Anwendung in Kalman-, Histogramm- und Partikelfilter eingeleitet. Die Umgebungsrepräsentation erfolgt in Form einer durch gleichzeitige Lageschätzung und Kartierung inkrementell erstellten Rasterkarte. Die Lagebestimmung in der Karte erfolgt mittels Monte-Carlo-Lokalisation. Dieses rechenintensive und dadurch langsam getaktete Verfahren wird hier durch eine signifikante Steigerung der Effizienz bei der Generierung virtueller Messgrößen, durch die Entwicklung eines Mehrratenansatzes und die Nebenläufigkeit von Prädiktion und Aktualisierung im Bayes'schen Rekursionsprozess für den direkten Einsatz in schnellen Regelungen qualifiziert. Der Wahrnehmung dynamischer Hindernisse wird durch eine Objektverfolgung Rechnung getragen, die aus den Rohdaten von Laserentfernungsmesssystemen Merkmale bewegter Objekte extrahiert und diese mittels der gemeinsamen Zuordnungswahrscheinlichkeit den durch Partikel repräsentierten Objekten zuordnet. Durch den Einsatz von Filtern für die Objektverfolgung können dabei kurzzeitige Verdeckungen und zufällige Fehlmessungen toleriert werden. Die Integration der Bestandteile erfolgt in der Entwicklung eines reaktiven Bahnplanungsalgorithmus, der unter Berücksichtigung dynamischer und nichtholonomer Zwangsbedingungen stichprobenbasiert den Arbeitsraum exploriert und aus Bewegungsprimitiven eine kollisionsfreie Bahn erzeugt. Dabei wird durch die stichprobenbasierte Generierung von Konfigurationen ein Suchbaum möglicher Trajektorien iterativ erweitert. Die Effizienz des Verfahrens wird durch verschiedene Heuristiken bei der Baumerweiterung sichergestellt. An einem Versuchsträger wird die Effektivität der vorgestellten Methoden nachgewiesen.

Schlagwörter: Lokalisation, Objektverfolgung, Wahrscheinlichkeitsbasierte Bahnplanung, Reaktive Bahnplanung

Abstract

For several years service robots have been proposed to undergo an unprecedented economic boom. Still, the spread of man-made servants is low. The main reasons are the complexity and non determinant nature of all day environments. This thesis addresses the uncertainty inherent in every day environments with respect to the navigation of wheeled mobile robots. By applying probabilistic robotics, especially the recurrent application of sequential Monte Carlo methods, an methodically continuous approach is chosen.

The introduction to the thesis includes the description of Bayes conditional probability and its application within Kalman, histogram, and particle filter. The environment representation is based on a grid map acquired incrementally through simultaneous localization and mapping. For localization with respect to the map the concept of Monte Carlo localization is used. This computationally expensive and for this reason at low clock rates driven method is sped up through a significant improvement in efficiency during generation of virtual measurements. Furthermore by means of a multi rate approach and concurrent prediction and update in Bayes recursive process this method is extended for use in high clock rate automatic feedback control. The perception of moving objects is done by an object tracking algorithm, which extracts features of moving objects from laser range measurements. It assigns these features to the objects tracked with individual particle filters by calculation of the joint assignment probability. The use of filters enables the algorithm to keep track of short time partially or fully occluded objects. The full information is integrated in an reactive path planning algorithm, which allows consideration of dynamic and nonholonomic constraints while exploring the workspace through iteratively sampling random robot configurations and connecting these through motion primitives to a search tree of possible trajectories if no collision occurs. The efficiency of the algorithms is tuned by expansion step heuristics.

Keywords: localization, object tracking, probabilistic path planning, reactive path planning

1 Einleitung

Als autonome mobile Roboter gelten solche Systeme, die sich in ihrer Umgebung bewegen, in ihr agieren und interagieren können. Kennzeichnend für diese Systeme ist, dass sie ihren Arbeitsraum mit Menschen teilen und diesen als intelligente Maschinen zur Ausführung von Aufgaben zu Diensten stehen. Es handelt sich dabei keineswegs um eine neuzeitliche Erfindung, vielmehr haben Menschen schon seit langer Zeit den Wunsch nach einem anspruchslosen Diener und haben dieses durch Mythen, Bücher und Filme dokumentiert.

Die technische Realisierung dieses Wunsches stellt große Herausforderungen an Wissenschaft und Technik und steht hinter der menschlichen Phantasie naturgemäß zurück. So weist beispielsweise die Ausschreibung *Leitinnovation Servicerobotik* des Bundesministeriums für Bildung und Forschung [BMBF, 2005] auf das Problem der Komplexität realistischer Umgebungen hin:

> Die Robotik steht als Sinnbild sowohl für mehr Wirtschaftlichkeit in der Produktion als auch für neue technische Assistenz- und Servicesysteme in Alltagsumgebungen. Sie gilt als Schlüsseltechnologie künftiger intelligenter Produkte. Ungeachtet des erreichten technisch-wissenschaftlichen Standes und der sozioökonomischen Bedeutung des Gebiets Assistenz- und Servicerobotik macht die Entwicklung neuer Produkte nur zögerliche Fortschritte. Die Ursachen hierfür liegen in hohem Maße an der fehlenden Alltagstauglichkeit der technisch-wissenschaftlichen Ergebnisse. Existierende Prototypen operieren in stark vereinfachten Welten und Aufgabenkontexten. Die Komplexität realistischer Aufgaben in Alltagsumgebungen wird nach wie vor nicht ausreichend beherrscht.

Wird weiterhin der große Umfang von Ressourcen betrachtet, die in einen von der amerikanischen Forschungsförderungseinrichtung des Militärs DARPA[1] ausgerichteten Wettbewerb, der *Grand Challenge*[2], einfließen, so kann ein Eindruck gewonnen werden, welcher Aufwand erforderlich ist, um mit den Prototypen aus Simulationen und Laboratorien in die Alltagsumgebung zu gehen.

Während die Industrieroboter im Rahmen der Automatisierung von Produktionsprozessen eine beispiellose Erfolgsgeschichte geschrieben haben, sind viele Ver-

[1] Defense Advanced Research Projects Agency
[2] www.darpa.mil/grandchallenge

suche, die Technologie in den Bereich der Servicerobotik zu übertragen, gescheitert. Der Hauptgrund für den mangelnden Erfolg der hochpräzisen Robotertechnik in der Welt der Servicerobotik liegt in der Beschaffenheit der Umgebung: Während die Umgebung von Industrierobotern auf die Anforderungen der Technik angepasst wird, muss der Serviceroboter in einer vorgegebenen Struktur zurechtkommen. Diese Umgebung ist geprägt durch eine große Vielfalt sowie durch eine Unsicherheit, die einerseits aus nicht deterministischen Vorgängen einhergehend mit einer mangelnden Vorhersagbarkeit und andererseits aus der Ermangelung einer umfassenden Wahrnehmung besteht. Als weiterer Aspekt bei der Übertragung von Methoden in die Servicerobotik ist der unterschiedliche Anforderungsschwerpunkt zu nennen: Kennzeichnend für den Industrieroboter sind wiederkehrende Tätigkeiten, die durch den Roboter mit großer Präzision und Effizienz ausgeführt werden. Der Serviceroboter bestreitet weniger wiederkehrende Aufgaben und es werden oftmals geringere Anforderungen an die Genauigkeit gestellt. Der Energieeffizienz eines mobilen Systems kommt durch die begrenzten mitzuführenden Energiespeicher eine zentrale Rolle zu. Da jedoch häufig wiederkehrende Abläufe selten sind, werden die Anforderungen an die globale Optimalität jedoch zugunsten eines geringeren Planungsaufwands und vor allem einer gesteigerten Reaktivität zurückgestellt, zumal durch die nicht vorhersagbare Umgebung die aufwändige, optimale Aufgabenlösung bereits im folgenden Schritt hinfällig sein kann. Die Beschränkung auf die Lösung von Aufgaben durch suboptimale Ergebnisse ermöglicht den Verzicht auf analytisch geschlossene Lösungen, die insbesondere bei hochdimensionalen Zustandsräumen nur mit erheblichem Aufwand für spezielle Fälle zu finden und implementieren sind, zugunsten von stichprobengestützten Suchverfahren. Hinreichend ist demnach schon die suboptimale Lösung einer Aufgabe, die sich mit angemessenem Aufwand berechnen lässt.

Ein erheblicher Teil der Funktionalität von Dienstleistungsrobotern wird durch die Informationsverarbeitung bestritten, dadurch werden in dieser derzeitig die größten Herausforderungen gestellt. Es gilt den Eigen- und Umgebungszustand mittels Sensorinformationen wahrzunehmen, um daraus Entscheidungen zu treffen und geeignete Aktionen abzuleiten. Zustände lassen sich in der Regel nicht mit lediglich einem Sensor erfassen sondern erfordern die Erfassung, Verarbeitung und Bewertung der Informationen aus mehreren Sensoren. Die der Umgebung inhärente Unsicherheit stellt die größte Herausforderung dar: Bei der Integration von Messungen ist stets zu bewerten, wie groß das Vertrauen in diese Information ist. Schließlich stellen die Algorithmen der Informationsverarbeitung durch Vereinfachungen und Modellfehler auch selbst eine Unsicherheitsquelle dar. Ein Hilfsmittel zur Berücksichtigung der Unsicherheit wird durch die Wahrscheinlichkeitsrechnung, insbesondere die auf Bayes zurückgehende bedingte Wahrscheinlichkeit, gegeben.

Die wahrscheinlichkeitsbasierten Methoden in der Wahrnehmung sind in der Lage, Modell- und Sensordaten zu integrieren. Sie zeichnen sich damit gegenüber rein modell- oder verhaltensbasierten Methoden durch eine erhöhte Robustheit bei Modell- und Sensorunsicherheiten aus. Die Anforderungen an den Detaillierungsgrad der Modelle und deren Parametrierung sind geringer, so auch die Anforderungen an die Genauigkeit der Sensoren, wodurch der Aufwand bei der Implementierung von Funktionalität bei gleicher oder besserer Leistung reduziert wird. Zusätzlicher Aufwand entsteht in der Informationsverarbeitung: Die Schätzung einer ganzen Wahrscheinlichkeitsdichtefunktion für eine Zustandsvariable statt eines einzelnen Zustands und die Verschiebung von Funktionalität in die Informationsverarbeitung ist rechenintensiv und stellt erhöhte Anforderungen an die Ressourcen zur Informationsverarbeitung. Die stetig fallenden Kosten bei rasantem Anstieg verfügbarer Rechenleistung lassen diesen Nachteil jedoch zunehmend in den Hintergrund rücken.

Diese Arbeit stellt einen Beitrag zur Berücksichtigung von Unsicherheiten beim Einsatz eines Roboters in Alltagsumgebungen dar. Zugrunde liegt die Fragestellung der Navigation eines autonomen Fahrzeugs: Wie kann ein mobiler Roboter kollisionsfrei aus einer Ausgangslage in eine Ziellage gelangen? Diese beinhaltet die Erfassung der Umgebung zwecks Erstellung einer Umgebungskarte, die Bestimmung der eigenen Position innerhalb der Karte, die Planung eines Weges aus der Ausgangslage zum Ziel, die Bahnführung auf dem geplanten Weg sowie die Erfassung sich bewegender Objekte und die reaktive Bahnplanung zur Kollisionsvermeidung. Die Lösung dieser unterschiedlichen Aspekte erfolgt durchgängig durch die Anwendung von wahrscheinlichkeitsbasierten Methoden. Die detaillierte Erläuterung der **Ziele und Gliederung dieser Arbeit** erfolgt im folgenden Kapitel.

2 Ziele und Gliederung dieser Arbeit

In der Einleitung wird motiviert, dass die Unsicherheit in der Wahrnehmung und die Unvorhersagbarkeit der Umgebung die größten Herausforderungen bei der Anwendung autonom agierender Systeme darstellen. Weiterhin wird dargelegt, das für einmalige oder geringzahlige Wiederholungen zugunsten einer gesteigerten Reaktivität auf das Auffinden einer global optimalen Lösung verzichtet werden kann. Als Lösung wird eine wahrscheinlichkeitsbasierte Zustandsschätzung für die Wahrnehmung sowie eine stichprobengestützte Suche für die Planung angeboten.

Ziel der vorliegenden Arbeit ist ein methodisch durchgängiges Konzept für die Führung autonomer Fahrzeuge in unsicherer Umgebung anzubieten, das alle für die Fahrzeugführung notwendigen Bestandteile umfasst, das ausreichend allgemein behandelt wird, um die Übertragbarkeit auf verschiedene Versuchs- und Anwendungsumgebungen zu gewährleisten, und das gleichermaßen detailliert behandelt wird, um eine konkrete Implementierung vornehmen zu können. Bei der Umsetzung wird darauf geachtet keine spezifischen strukturellen Merkmale einer Versuchsumgebung auszunutzen, sondern mit einfachen und allgemein gültigen Modellen zu arbeiten, deren Parameter mit vertretbarem Aufwand zu ermitteln sind.

Als Grundlage für die Wahrnehmung dient hier die wahrscheinlichkeitsbasierte Zustandsschätzung auf Basis der Information verschiedener Sensoren. Diese wiederkehrende Methodik findet sich in den Kernaufgaben der Navigation wieder. Im Bereich der Planung werden stichprobengestützte Suchverfahren eingesetzt, die auch in hochdimensionalen Suchräumen gute Ergebnisse erzielen und die, abhängig von der Heuristik zur Stichprobenwahl, effizient arbeiten.

Im folgenden wird die Gliederung der Arbeit mit einer kurzen inhaltlichen Zusammenfassung der einzelnen Kapitel erläutert. Ein Überblick über den Stand des Wissens mit dem Bezug zu wichtigen Literaturstellen erfolgt jeweils in den Kapiteleinführungen.

Zunächst vermittelt das Kapitel **Wahrscheinlichkeitsbasierte Zustandsschätzung** einige Grundlagen. Diese beinhalten den Wahrscheinlichkeitsbegriff und die für diese Arbeit elementare durch den Satz von Bayes ausgedrückte bedingte Wahrscheinlichkeit. Über die Regel der totalen Wahrscheinlichkeit und die Markov-Prozesse wird auf das Bayes'sche Filter übergeleitet. Dessen Ausprägungen Kalman-Filter, Histogrammfilter und Partikelfilter werden mit ihren Eigenschaften erläutert.

Bild 2.1: Der Versuchsträger verfügt über eine mobile Basis und einen Leichtbauroboterarm zur Interaktion mit der Umgebung.

Das Kapitel **Versuchsumgebung** stellt den in Bild 2.1 dargestellten Versuchsträger vor. Neben einer kurzen Einführung der Hardware wird die modulare Struktur der Software mit den Kommunikationseinrichtungen vorgestellt. Bild 4.4 gibt einen Überblick über die verschiedenen Module und deren Verknüpfung. Weiterhin werden hier elementare Funktionen wie die Schätzung der Robotergeschwindigkeit für die spezifische Kinematik sowie die Repräsentation der Bahn und die Bahnregelung dargestellt. Um den Anspruch auf Allgemeingültigkeit zu gewährleisten und die Übertragbarkeit der Methoden auf andere Anwendungen sicherzustellen, wird nur der nötigste Bezug zum Versuchsträger hergestellt und auf ausgewählte Spezifika der konkreten Versuchseinrichtung eingegangen.

Nach den vorhergehenden Grundlagenabschnitten erfolgt die Beschreibung der **Wahrnehmung der Umgebung**. Die Wahrnehmung beruht auf einem Umgebungsmodell, das in der vorliegenden Arbeit durch Rasterkarten dargestellt wird. Neben dem Umgebungsmodell wird die zur Wahrnehmung geeignete Sensorik vorgestellt. Elementar für die folgenden Abschnitte ist die Verfügbarkeit einer Karte. Um mit Hilfe des Versuchsträgers eine Karte zu erstellen, wird ausgehend von der Kartenerstellung bei bekannter Lage auf das Problem der gleichzeitigen Lageschätzung und Kartierung (*Simultaneous Localization and Mapping, SLAM*) für Rasterkarten mittels Partikelfiltern hingearbeitet.

Im Kapitel **Lokalisation mobiler Fahrzeuge** wird die derzeitig als überlegen gel-

tende MONTE-CARLO-Lokalisation für den Einsatz in Rasterkarten vorgestellt. Neben der effizienten Implementierung des auf Partikelfiltern basierten Verfahrens wird dieses vornehmlich mit langer Taktzeit arbeitende Verfahren durch einen Mehrratenansatz und die Nebenläufigkeit von Prädiktion und Korrektur für den Einsatz in schnellen Regelkreisen qualifiziert.

Mittels vorgenannter Lokalisation steht bereits die technische Möglichkeit des Abfahrens einer vorgegebenen Bahn zur Verfügung. In Kapitel **Verfolgung dynamischer Objekte** erfolgt die Erweiterung um die Funktionalität der Erfassung dynamischer Hindernisse von einem fahrenden Roboter aus. Es wird ein Verfahren vorgestellt, mit dem aus Sensorinformationen auf Lage, Richtung und Geschwindigkeit dynamischer Hindernisse im Wahrnehmungsbereich des Roboters geschlossen werden kann. Die Objektzustände werden hier durch individuelle Partikelwolken repräsentiert, deren Partikel jeweils eine Hypothese über den Aufenthaltsort des Roboters darstellen. Die Gewichtung der Partikel erfolgt durch Prüfung der Plausibilität anhand von aus Messdaten erstellten Wahrscheinlichkeitskarten. Die Zuordnung von Messung zu Objekt erfolgt durch die Schätzung einer gemeinsamen Zuordnungswahrscheinlichkeit.

Die geschätzten Objektzustände werden in Kapitel **Bahnplanung, Navigation und Hindernisvermeidung** bei der reaktiven Bahnplanung berücksichtigt. Den Kern der reaktiven Bahnplanung stellt ein Regelungsgesetz zur Überführung des Roboters in hindernisfreier Umgebung aus der aktuellen Lage in eine Ziellage dar. Mit Hilfe dieses elementaren Planers wird iterativ ein Suchbaum von möglichen Trajektorien aufgebaut, der bei erfolgreicher Bahnplanung Start- und Ziellage des Roboters beinhaltet. Durch einen Kollisionstest vor jeder Erweiterung des Suchbaums wird Kollisionsfreiheit mit statischer Umgebung und dynamischen Hindernissen gewährleistet. Kernelement für die erfolgreiche Umsetzung der Bahnplanung ist die Entkopplung von Planung und Ausführung, mit der der Planungsalgorithmus mit einer größeren Zeit getaktet werden kann, als der ausführende Bahnregler, so dass für Exploration des Arbeitsraum und Optimierung von Lösungen hinreichend Zeit zur Verfügung steht.

3 Wahrscheinlichkeitsbasierte Zustandsschätzung

Die Umgebung von autonomen Fahrzeugen ist geprägt durch einen unsicheren Arbeitsraum, der nur unzureichend bekannt ist, der darüber hinaus in mehreren Stufen dynamisch ist und der mittels Sensorik nur fehlerhaft erfasst werden kann. Die erste Stufe der Dynamik beinhaltet die Tatsache, das sich der Zustand von Objekten in der Umgebung langsam verändern kann, beispielsweise können Möbel verschoben werden, Türen können offen oder geschlossen sein und neue Objekte können im Arbeitsraum auftauchen. Diese Veränderungen sind im Umweltmodell des Roboters nicht vorhanden und müssen selbständig erfasst und hinzugefügt werden, damit der Roboter sich weiterhin sicher in der Umgebung bewegen kann. Die zweite Stufe dynamischer Objekte beinhaltet jene mit einer Bewegungsgeschwindigkeit in der Größenordnung des Roboters. Diese bewegten Objekte sind a priori nicht bekannt und müssen erfasst werden. Sie stören darüber hinaus die Messungen des Roboters gegenüber der statischen Umgebung, so dass Unsicherheiten in die Messung hereingetragen werden, die zu berücksichtigen sind.

Wesentliches Element der vorliegenden Arbeit ist die Berücksichtigung von Unsicherheiten bei der Schätzung des Eigen- und Umgebungszustands ohne eine ereignisdiskrete Sonderbehandlung. Durch die *ab ovo* modellierten Unsicherheiten wird der Prozess der Zustandsschätzung robuster und der Betrieb des Systems wird durch die intrinsisch größere Zahl der Störfälle sicherer und zuverlässiger.

Die Unsicherheit wird dadurch repräsentiert, dass statt einer Zustandsvariable selbst die Wahrscheinlichkeitsdichtefunktion bzw. die Wahrscheinlichkeitsfunktion über den gesamten Zustandsraum dieser Variable geschätzt wird. Eine Annahme über einen konkreten Zustand kann dann nach verschiedenen Kriterien (z.B. Maximum Likelihood) aus dieser Funktion abgelesen werden.

Die folgenden Abschnitte widmen sich den Grundlagen der Wahrscheinlichkeitsrechnung und definieren Zustandsräume im kontinuierlichen sowie im diskreten Fall. Erläutert wird das auf Bayes zurückgehende Prinzip der bedingten Wahrscheinlichkeiten. Mit Hilfe der Markov-Ketten wird das Bayes'sche Filter dargestellt, dessen Ausprägungen Kalman-Filter und Partikelfilter in den darauf folgenden Abschnitten genauer erläutert werden.

3.1 Zustandsräume

Systeme werden durch ihren Zustand charakterisiert. Der Zustand kann als die Summe der Informationen gedeutet werden, die einen Einfluss auf die Zukunft des Systems haben. Einige Zustandsgrößen variieren über die Zeit wie z.B. die Position von Menschen im Umfeld des Roboters, dabei handelt es sich um die dynamischen Zustandsgrößen. Andere bleiben statisch, wie z.B. der Grundriss der Umgebung im Arbeitsraum, die die statischen Zustandsgrößen darstellen. Beim Beispiel des Roboters gehören auch Variablen des Roboters selbst, wie z.B. die Lage im Raum, die Geschwindigkeit und die momentane Konfiguration der Sensoren und Aktoren zu den dynamischen Zustandsgrößen.

Im Folgenden sind typische Zustandsgrößen im Zusammenhang dieser Arbeit aufgelistet:

- Die *Pose* beinhaltet Lage und Orientierung des Roboters mit Bezug zu einem Referenzkoordinatensystem. Im kartesischen Raum bei Annahme eines Starrkörpermodells umfasst die Pose sechs Koordinaten.
- Die *Robotergeschwindigkeit* wird als zeitliche Änderung der Pose ebenfalls mittels sechs Koordinaten beschrieben.
- Die *Konfiguration* des Roboters beinhaltet mit der Beschreibung der Freiheitsgrade des Roboters interne Zustandsgrößen, z.B. die Gelenkwinkel von rotatorischen Antrieben.
- Die *Lage und Eigenschaften von Objekten in der Umgebung* stellen externe Zustandsgrößen dar. Abhängig von der Präzision des Umweltmodells ist für eine genaue Beschreibung der Umgebung ist eine sehr große Anzahl von Zustandsvariablen erforderlich.
- Ähnlich wie Lage und Geschwindigkeit des Roboters werden *Lage und Geschwindigkeit von Objekten und Menschen in der Umgebung des Roboters* mittels Zustandsgrößen beschrieben.
- Darüber hinaus gibt es eine große Anzahl von Zustandsgrößen, die Einfluss auf die Zukunft haben: Ob ein Aktor oder Sensor korrekt arbeitet, der Ladezustand der Batterien, der Abnutzungsgrad mechanischer Komponenten etc.

Ein Zustand $\boldsymbol{x}_k$ zum Zeitschritt k wird *vollständig* genannt, wenn mit diesem die Zukunft bestmöglich vorausgesagt werden kann. Daraus folgt, dass es keine weiteren Informationen in Form von Messungen, Eingängen und alten Zuständen gibt, die eine bessere Vorhersage der Zukunft ermöglichen würden. Prozesse, die

diese Bedingung erfüllen werden als MARKOV-Ketten bezeichnet (siehe auch Abschnitt 3.2.2).

In der Praxis lässt sich ein vollständiger Zustand nicht erfassen. Spätestens, wenn es darum geht, die Motivation für eine plötzliche Änderung des Bewegungszustands eines Menschen in der Umgebung des Roboters zu beschreiben, lässt sich die vollständige Zustandsbeschreibung nicht mehr bewerkstelligen. Stattdessen wird nur eine unvollständige Untermenge von Zuständen genutzt, die als *unvollständiger Zustand* beschrieben wird.

Unterschieden wird weiterhin zwischen kontinuierlichen und diskreten Zuständen. Die meisten Zustände sind kontinuierlich definiert, beispielsweise die Pose des Roboters. In einigen Fällen können auch diskrete Zustände vorhanden sein, wie z.B. der Schaltzustand eines Relais etc. Auch die zeitliche Beschreibung kann kontinuierlich oder diskret erfolgen. In der Realität ist die Zeit immer kontinuierlich, bei der Beschreibung von Systemen mittels Rechnern lässt sich eine diskrete Beschreibung der Zeit nicht vermeiden.

3.2 Grundlagen der Wahrscheinlichkeitsrechnung

Im Folgenden sollen die Grundlagen zur Anwendung der Wahrscheinlichkeitsrechnung auf die gegebene Aufgabenstellung dargestellt werden. Hierzu werden Größen wie Messungen, Eingänge und Zustände des Roboter und der Umgebung als Zufallsvariablen aufgefasst. Zufallsvariablen können gemäß der zugehörigen Wahrscheinlichkeiten verschiedene Werte annehmen. Mit Hilfe der wahrscheinlichkeitsbasierten Inferenz sollen unbekannte Zufallsvariablen von bekannten Zufallsvariablen abgeleitet werden. Zum Beispiel sei X eine Zufallsvariable und x ein spezifischer Wert, den X annehmen kann. Wenn nun der Raum der Zustandsvariable den X annehmen kann diskret ist, können wir die Wahrscheinlichkeit für $X = x$ mit

$$p(X = x) \tag{3.1}$$

ausdrücken. Die Summe aller diskreten Wahrscheinlichkeiten einer Zustandsvariable ergibt sich zu eins:

$$\sum_x p(X = x) = 1 \quad . \tag{3.2}$$

Damit ergibt sich, dass die Wahrscheinlichkeiten für diskrete Ereignisse stets kleiner oder gleich eins sein müssen. Weiterhin müssen die Wahrscheinlichkeiten stets größer oder gleich null sein: $0 \leq p(X = x) \leq 1$. Zugunsten der kompakten Schreibweise wird im Folgenden auf die Bezeichnung der Zufallsvariable verzichtet. Damit wird $p(X = x)$ zu $p(x)$.

Ein Beispiel zur Anwendung der Wahrscheinlichkeitsfunktion stellt der Münzwurf dar: Der diskrete Zustandsraum besteht aus $X = \{\text{Kopf}, \text{Zahl}\}$ mit den Wahrscheinlichkeiten $p(\text{Kopf}) = p(\text{Zahl}) = 1/2$. Die Summe der Wahrscheinlichkeiten über den Zustandsraum ergibt sich zu $p(\text{Kopf}) + p(\text{Zahl}) = 1$.

Eine Vielzahl von Zufallsvariablen ist jedoch nicht im diskreten, sondern im kontinuierlichen Raum definiert. Die Wahrscheinlichkeit, dass die Zufallsvariable einen bestimmten Wert annimmt wird über die Wahrscheinlichkeitsdichtefunktion wiedergegeben. Die Integration über die Wahrscheinlichkeitsdichtefunktion ergibt analog zur diskreten Summe der Wahrscheinlichkeiten (Gl. (3.2)) eins

$$\int p(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \mathbf{1} \quad . \tag{3.3}$$

Dennoch ist im Gegensatz zur diskreten Wahrscheinlichkeit die kontinuierliche Wahrscheinlichkeitsdichtefunktion nach oben nicht durch eins begrenzt.

Die wohl bekannteste Wahrscheinlichkeitsdichtefunktion für den eindimensionalen Fall ist die Normalverteilung mit Mittelwert μ und Varianz σ^2

$$p(x) = (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right) \quad . \tag{3.4}$$

Die Normalverteilung spielt in der Praxis eine große Rolle, viele Zufallsvariablen folgen ihr zumindest näherungsweise, die Beschreibung durch die Parameter Mittelwert und Varianz ist sehr effizient. Die Normalverteilung in der oben gezeigten Form gilt für eine skalare Größe. Die multivariate Normalverteilung in der Form

$$p(\boldsymbol{x}) = \det(2\pi\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{3.5}$$

stellt die Normalverteilung über Vektoren dar. Hier ist $\boldsymbol{\mu}$ der Mittelwertvektor und $\boldsymbol{\Sigma}$ die Kovarianzmatrix (positiv semidefinit, symmetrisch).

Nachteil ist die Beschränkung auf unimodale Zufallsvariablen: Mehrdeutigen Situationen wird die Normalverteilung nicht gerecht. Durch den Mittelwert, der sich beispielsweise bei einer bimodalen Verteilung zwischen den Maxima befindet, wird die Zustandsschätzung bei Annäherung mit der Normalverteilung stark verfälscht.

3.2.1 Bedingte Wahrscheinlichkeiten

Sind A und B Zustände in Form kontinuierlicher Zufallsvariablen und repräsentieren $p(A)$ und $p(B)$ die entsprechenden Wahrscheinlichkeiten für deren Auftreten, dann kennzeichnet $p(A|B)$ die Wahrscheinlichkeit von A bei gegebenem Zustand von B. Man spricht dabei auch von der bedingten Wahrscheinlichkeit [Grinstead

und Snell, 2003]. Daraus ergibt sich die gemeinsame Wahrscheinlichkeit für A und B zu $p(A,B) = p(A|B)p(B)$. Der Zustand A ist statistisch unabhängig von B, wenn $p(A|B) = p(A)$ gilt. Das heißt, B enthält keine weiteren Aussagen über A. Für zwei unabhängige Zustände gilt für die gemeinsame Wahrscheinlichkeit $p(A,B) = p(A)p(B)$.

Ist A statistisch abhängig von B so lässt sich im kontinuierlichen Fall die so genannte *totale Wahrscheinlichkeit* für A mit

$$p(A) = \int p(A|B) \cdot p(B) \mathrm{d}B \tag{3.6}$$

ausdrücken. Im diskreten Fall ergibt sich in

$$p(A) = \sum_B p(A|B) \cdot p(B) \tag{3.7}$$

eine Summe anstatt des Integrals.

Das Bayes-Theorem für bedingte Wahrscheinlichkeiten lautet

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)} \quad . \tag{3.8}$$

Der Nenner $p(B)$ dient dabei dazu, dass sich das Integral $\int p(A|B)\mathrm{d}A$ zu 1 ergibt. Aus diesem Grund lässt sich Gl. (3.8) auch darstellen als

$$p(A|B) = \eta p(B|A)p(A) \quad , \tag{3.9}$$

wobei η eine Normalisierungskonstante darstellt und nachträglich berechnet werden kann [Bayes, 1763].

Mit Hilfe dieser Gleichung lässt sich die Wahrscheinlichkeit $p(A|B)$ durch ihre „Inverse“ $p(B|A)$ ausdrücken, oder anders gesagt: $p(Ursache|Ereignis)$ durch $p(Ereignis|Ursache)$. Auf das praktische Problem bezogen, lässt sich dieser Zusammenhang folgendermaßen interpretieren: A sei ein Zustand, den wir kennen möchten, aber nur durch die Sensormessung B beobachten können. So kann aus der von der Messung unabhängigen a priori Zustandsvorhersage $p(A)$ und dem Sensormodell $p(B|A)$, das vom Zustand A auf das Messergebnis B schließen lässt, die gesuchte a posteriori Zustandswahrscheinlichkeit $p(A|B)$ geschlussfolgert werden. Dabei ist das Sensormodell meist einfacher herleitbar, da bei einem gegebenen Zustand die möglichen Sensormessungen leicht abgeleitet werden können.

3.2.2 Markov-Ketten

Eine Markov-Kette ist ein zeitdiskreter stochastischer Prozess, der die Markov-Eigenschaft aufweist: In einem Markov-Prozess k-ter Ordnung ist der Folgezustand

$\boldsymbol{x}_{k+1}$ nur von k vorhergehenden Zuständen abhängig und nicht von der gesamten Vorgeschichte $\boldsymbol{x}_k, \boldsymbol{x}_{k-1}, \ldots, \boldsymbol{x}_0$. Bei einem Prozess erster Ordnung hängt $\boldsymbol{x}_{k+1}$ nur von dem momentanen Zustand $\boldsymbol{x}_k$ ab. Der mathematische Ausdruck für einen Prozess erster Ordnung lautet

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k, \boldsymbol{x}_{k-1}, \boldsymbol{x}_{k-2}, \ldots, \boldsymbol{x}_0) = p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k) \quad . \tag{3.10}$$

Markov-Ketten eignen sich für die Modellierung zufälliger Zustandsänderungen, wenn diese sich gegenseitig nur über einen begrenzten Zeitraum, oder gar nicht beeinflussen. Wie bereits in Abschnitt 3.1 erwähnt, ist es bezüglich der vorliegenden Aufgabenstellung sehr aufwändig, einen Zustandsvektor zu definieren, so dass die Markov-Bedingung gilt. In der Praxis zeigt sich jedoch eine gewisse Robustheit gegenüber Verletzungen der Markov-Bedingung bei der Anwendung wahrscheinlichkeitsbasierter Methoden zur Zustandsschätzung. [Grinstead und Snell, 2003, Kapitel 11]

3.2.3 Bayes'sche Filter

Das Bayes'sche Filter berechnet die Wahrscheinlichkeit $p(\boldsymbol{x}_k|\boldsymbol{z}_{k\ldots0})$ des aktuellen Zustands $\boldsymbol{x}_k$ im Zeitschritt k , basierend auf allen bis dahin aufgenommenen Sensormessungen $\boldsymbol{z}_{k\ldots0}$. Dieses erfolgt, indem rekursiv die im vorhergehenden Schritt $k-1$ berechnete Wahrscheinlichkeit $p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{k-1\ldots0})$ unter Verwendung der aktuellen Sensormessung $\boldsymbol{z}_k$ aktualisiert wird [Doucet u. a., 2001].

Mit Hilfe des Satzes von Bayes (Gl. (3.8)) lässt sich $p(\boldsymbol{x}_k|\boldsymbol{z}_{k\ldots0})$ aufspalten

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{k\ldots0}) = \eta p(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{z}_{k-1\ldots0}) p(\boldsymbol{x}_k|\boldsymbol{z}_{k-1\ldots0}) \quad . \tag{3.11}$$

Unter der Voraussetzung, dass die Sensordaten nur auf dem jeweiligen Zustand beruhen und unabhängig von vorherigen Messungen sind, vereinfacht sich das Sensormodell zu

$$p(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{z}_{k-1\ldots0}) = p(\boldsymbol{z}_k|\boldsymbol{x}_k) \quad . \tag{3.12}$$

Durch Anwenden der Regel der totalen Wahrscheinlichkeit aus Gl. (3.6) lässt sich der zweite Teil von Gl. (3.11) ausdrücken als

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{k-1\ldots0}) = \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{z}_{k-1\ldots0}) p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{k-1\ldots0}) \mathrm{d}\boldsymbol{x}_{k-1} \quad . \tag{3.13}$$

Die Annahme, dass die Zustandsabfolge eine Markov-Kette 1. Ordnung bildet, führt zu der Vereinfachung

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{k-1\ldots0}) = \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{k-1\ldots0}) \mathrm{d}\boldsymbol{x}_{k-1} \quad . \tag{3.14}$$

Somit ergibt sich die Bayes'sche Rekursionsformel

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{k-1\ldots0}) \overset{\text{Prädiktion}}{=} \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{k-1\ldots0})\mathrm{d}\boldsymbol{x}_{k-1} \quad , \tag{3.15}$$

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{k\ldots0}) \overset{\text{Korrektur}}{=} \eta p(\boldsymbol{z}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{z}_{k-1\ldots0}) \quad . \tag{3.16}$$

Der erste Term im Integral der Prädiktionsgleichung $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ stellt die Abhängigkeit des Zustand $\boldsymbol{x}_k$ vom Zustand im vorherigen Abtastschritt $\boldsymbol{x}_{k-1}$ dar. Konkrete Implementierungen berücksichtigen dabei die Stellgröße $\boldsymbol{u}_k$, die ursächlich für Zustandsübergang ist. Damit ergibt sich die erweiterte Prädiktionsgleichung zu $p(\boldsymbol{x}_k|\boldsymbol{u}_k, \boldsymbol{x}_{k-1})$. Mit Hilfe der Gln. (3.15) und (3.16) lässt sich ein rekursiver Algorithmus zur Aktualisierung der Wahrscheinlichkeitsverteilung des Zustands $\boldsymbol{x}_k$ aufstellen, der im Wesentlichen aus zwei Teilen besteht [Thrun u. a., 2005]. Im ersten Teil (siehe Algorithmus 3.1, Zeile 3) wird mit Gl. (3.15) eine Vorhersage aufgrund

Algorithmus 3.1 Das Bayes'sche Filter in der allgemeinen Form für die i Elemente des Zustandsvektors $\boldsymbol{x}_k$.

```
1: BayesFilter(x̃_{k-1}, u_k, z_k)
2:     for all x_k^[i] do
3:         x̄_k^[i] = ∫ p(x_k^[i] | u_k, x_{k-1}^[i]) x̃_{k-1}^[i] dx_{k-1}^[i]
4:         x̃_k^[i] = η p(z_k | x_k^[i]) x̄_k^[i]
5:     end for
6:     return x̃_k
7: end
```

des Ergebnisses aus dem Schritt $k-1$ und des physikalischen Bewegungsmodells $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ getroffen. Im zweiten Teil (siehe Zeile 4) wird diese Vorhersage mit Gl. (3.16) unter Einbezug der aktuellen Sensormessung $\boldsymbol{z}_k$ und des Sensormodells $p(\boldsymbol{z}_k|\boldsymbol{x}_k)$ verbessert. Da der tatsächliche Zustand $\boldsymbol{x}_k$ in der Regel nicht bestimmt werden kann, sondern stets nur eine Schätzung erfolgt, wird die geschätzte Größe durch $\tilde{(\cdot)}$ gekennzeichnet.

Zur Initialisierung wird der Algorithmus mit einer anfänglichen Wahrscheinlichkeitsverteilung des Zustands $p(\boldsymbol{x}_0)$ gestartet. Wenn dieser Anfangszustand unbekannt ist, kann z. B. eine Gleichverteilung über den gesamten Zustandsraum angenommen werden.

3.3 KALMAN-Filter

Die wohl bekannteste Ausprägung des BAYES'schen Filters ist das KALMAN-Filter. Als erste funktionsfähige Implementierung für den kontinuierlichen Zustandsraum stellt es den geschätzten Zustand als uni- oder multivariate Normalverteilung dar, siehe Gl. (3.5):

$$p(\boldsymbol{x}) = \det(2\pi\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \quad .$$

Für diskrete und hybride Zustandsräume kann das KALMAN-Filter nicht eingesetzt werden. Die Wahrscheinlichkeitsdichtefunktion über $\boldsymbol{x}$ wird hier durch die Parameter Mittelwert $\boldsymbol{\mu}$ und die Kovarianzmatrix $\boldsymbol{\Sigma}$ repräsentiert. Die Kovarianzmatrix $\boldsymbol{\Sigma}$ ist stets symmetrisch und positiv semidefinit. Die Dimension ergibt sich aus der Dimension des Zustands $\boldsymbol{x}$ zu $\dim(\boldsymbol{x}) \times \dim(\boldsymbol{x})$. Die gewählte Form zur Beschreibung der Wahrscheinlichkeitsdichtefunktion wird auch Parametrierung mittels Momenten genannt, da der Mittelwert $\boldsymbol{\mu}$ und die Kovarianz $\boldsymbol{\Sigma}$ die ersten und zweiten Momente der Wahrscheinlichkeitsdichtefunktion darstellen. Eine alternative stellt die kanonische Parametrierung dar, welche auf das Informationsfilter führt, das zur Bereitstellung der gleichen Funktion teils andere rechnerische Anforderungen stellt und bezüglich der Berücksichtigung von Prädiktion und Korrektur orthogonale Eigenschaften aufweist. Wegen der prinzipiellen Ähnlichkeit der Ansätze und der vergleichbaren rechnerischen Effizienz soll hier auf eine Beschreibung des Informationsfilters verzichtet werden [Thrun u. a., 2005].

Die Funktionsweise des KALMAN-Filters ist wie folgt: Zum Zeitschritt k wird der geschätzte Zustand durch Mittelwert $\boldsymbol{x}_k$ und Kovarianzmatrix $\boldsymbol{\Sigma}_k$ repräsentiert. Die Posteriori-Verteilung ist normalverteilt, wenn neben der Annahme eines MARKOV-Prozesses eine lineare Übertragungsfunktion $p(\boldsymbol{x}_k|\boldsymbol{u}_k, \boldsymbol{x}_{k-1})$ mit Überlagerung normalverteilter Störung vorhanden ist. Dieses entspricht der Zustandsgleichung

$$\boldsymbol{x}_k = \boldsymbol{A}_k\boldsymbol{x}_{k-1} + \boldsymbol{B}_k\boldsymbol{u}_k + \boldsymbol{\epsilon}_k \tag{3.17}$$

mit dem Zustandsvektor $\boldsymbol{x}_k$, der Systemmatrix $\boldsymbol{A}_k$, der Eingangsmatrix $\boldsymbol{B}_k$ und dem Eingangsvektor $\boldsymbol{u}_k$. Zusätzlich werden normalverteilte Prozessstörungen $\boldsymbol{\epsilon}_k$ angenommen, die Unsicherheiten beim Zustandsübergang modellieren. Der Vektor der Störungen hat die gleiche Dimension wie der Zustandsvektor mit Mittelwert gleich null und Kovarianz $\boldsymbol{R}_k$. Wird diese Gl. (3.17) in Gl. (3.5) eingesetzt, so ergibt sich

$$p(\boldsymbol{x}_k|\boldsymbol{u}_k, \boldsymbol{x}_{k-1}) = \det(2\pi\boldsymbol{R}_k)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\left(\boldsymbol{x}_k - \boldsymbol{A}_k\boldsymbol{x}_{k-1} - \boldsymbol{B}_k\boldsymbol{u}_k\right)^{\mathrm{T}}\boldsymbol{R}_k^{-1}\left(\boldsymbol{x}_k - \boldsymbol{A}_k\boldsymbol{x}_{k-1} - \boldsymbol{B}_k\boldsymbol{u}_k\right)\right) \quad . \tag{3.18}$$

Auch die Messgleichung $p(\boldsymbol{z}_k|\boldsymbol{x}_k)$ muss linear mit Überlagerung normalverteilten Rauschens sein,

$$\boldsymbol{z}_k = \boldsymbol{C}_k\boldsymbol{x}_k + \boldsymbol{\delta}_k \quad , \tag{3.19}$$

mit dem Messvektor $\boldsymbol{z}_k$, der Messmatrix $\boldsymbol{C}_k$ und mittelwertfreiem, normalverteiltem, multivariatem Rauschanteil $\boldsymbol{\delta}_k$, der die Kovarianz $\boldsymbol{Q}_k$ hat. Die Wahrscheinlichkeitsdichtefunktion der Messung ergibt sich damit zu

$$p(\boldsymbol{z}_k|\boldsymbol{x}_k) = \det(2\pi\boldsymbol{Q}_k)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{z}_k - \boldsymbol{C}_k\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{Q}_k^{-1}(\boldsymbol{z}_k - \boldsymbol{C}_k\boldsymbol{x}_k)\right) \quad . \tag{3.20}$$

Auch die initiale Zustandsschätzung $p(\boldsymbol{x}_0)$ muss normalverteilt sein. Der Mittelwert wird mit $\boldsymbol{\mu}_0$ und die Kovarianz mit $\boldsymbol{\Sigma}_0$ bezeichnet:

$$p(\boldsymbol{x}_0) = \det(2\pi\boldsymbol{\Sigma}_0)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}_0 - \boldsymbol{\mu}_0)^{\mathrm{T}}\boldsymbol{\Sigma}_0^{-1}(\boldsymbol{x}_0 - \boldsymbol{\mu}_0)\right) \quad . \tag{3.21}$$

Bei Einhaltung der vorgenannten Eigenschaften kann davon ausgegangen werden, dass der geschätzte Zustand normalverteilt ist. Für den Beweis sei auf [Kalman, 1960] verwiesen.

Die Funktionsweise des KALMAN-Filters ist in Algorithmus 3.2 dargestellt. Die

Algorithmus 3.2 Das KALMAN-Filter für lineare GAUSS'sche Zustands- und Messgleichung.

1: KALMANFILTER($\boldsymbol{\mu}_{k-1}, \boldsymbol{\Sigma}_{k-1}, \boldsymbol{u}_k, \boldsymbol{z}_k$)
2: $\quad \bar{\boldsymbol{\mu}}_k = \boldsymbol{A}_k\boldsymbol{\mu}_{k-1} + \boldsymbol{B}_k\boldsymbol{u}_k$
3: $\quad \bar{\boldsymbol{\Sigma}}_k = \boldsymbol{A}_k\boldsymbol{\Sigma}_{k-1}\boldsymbol{A}_k^{\mathrm{T}} + \boldsymbol{R}_k$
4: $\quad \boldsymbol{K}_k = \bar{\boldsymbol{\Sigma}}_k\boldsymbol{C}_k^{\mathrm{T}}\left(\boldsymbol{C}_k\bar{\boldsymbol{\Sigma}}_k\boldsymbol{C}_k^{\mathrm{T}} + \boldsymbol{Q}_k\right)^{-1}$
5: $\quad \boldsymbol{\mu}_k = \bar{\boldsymbol{\mu}}_k + \boldsymbol{K}_k\left(\boldsymbol{z}_k - \boldsymbol{C}_k\bar{\boldsymbol{\mu}}_k\right)$
6: $\quad \boldsymbol{\Sigma}_k = \left(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{C}_k\right)\bar{\boldsymbol{\Sigma}}_k$
7: $\quad$ **return** $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$
8: **end**

Eingangsgrößen des Filters sind Mittelwert $\boldsymbol{\mu}_{k-1}$ und Kovarianz $\boldsymbol{\Sigma}_{k-1}$ des vorhergehenden Abtastschritts sowie der Eingangsvektor $\boldsymbol{u}_k$ und die Messung $\boldsymbol{z}_k$. Ausgang sind der aktualisierte Mittelwert $\boldsymbol{\mu}_k$ und die aktualisierte Kovarianz $\boldsymbol{\Sigma}_k$. In den Zeilen 2 und 3 wird die Prädiktion nach der linearen Zustandsgleichung berechnet. In Zeile 3 wird dabei die Unsicherheit durch Addition der Kovarianz des Prozessrauschens $\boldsymbol{R}_k$ in der Kovarianzmatrix $\bar{\boldsymbol{\Sigma}}_k$ berücksichtigt. In den Zeilen 4 bis 6 erfolgt schließlich die Korrektur der Prädiktion durch Einbezug der Messung $\boldsymbol{z}_k$. In

Zeile 4 wird dafür die so genannte KALMAN-Verstärkung $\boldsymbol{K}_k$ berechnet, welche in Abhängigkeit der durch die prädizierte Kovarianz $\boldsymbol{\Sigma}_k^-$ repräsentierten Zustandsunsicherheit steuert, inwieweit für die Zustandsschätzung auf die Modellgleichung in Abwägung zur Messung vertraut wird. Für die Implementierung des KALMAN-Algorithmus stellt die Zeile 4 durch die Matrizeninversion den größten informationstechnischen Aufwand dar. In Zeile 5 wird schließlich der korrigierte Mittelwert $\boldsymbol{\mu}_k$ aus dem geschätzten Mittelwert $\boldsymbol{\mu}_k^-$ und der durch die KALMAN-Verstärkung gewichteten Innovation $(\boldsymbol{z}_k - \boldsymbol{C}_k\boldsymbol{\mu}_k^-)$ berechnet. Die Berücksichtigung der Messung bei der Zustandsschätzung verringert gleichzeitig die Unsicherheit des a posteriori geschätzten Zustands. Dieses erfolgt in Zeile 6 durch die Korrektur der Kovarianzmatrix $\boldsymbol{\Sigma}_k$ mit Hilfe der KALMAN-Verstärkung.

Die Effizienz des KALMAN-Filters beruht wesentlich auf der Eigenschaft, dass die Abbildung einer normalverteilten Funktion durch eine lineare Transformation stets eine normalverteilte Funktion ergibt. Erst diese Eigenschaft ermöglicht eine geschlossene Lösung für die Berechnung der Parameter der normalverteilten Wahrscheinlichkeitsdichte des geschätzten Zustands. Dieser Zusammenhang ist in Bild 3.1a dargestellt. Aus der normalverteilten Ausgangsverteilung wird durch eine lineare Transformation eine normalverteilte Zielverteilung.

Tatsächlich gibt es die vom KALMAN-Filter geforderten Linearitätseigenschaften nur selten. Im nichtlinearen Fall werden Gl. (3.17) und Gl. (3.19) zu den nichtlinearen Ausdrücken

$$\boldsymbol{x}_k = g\left(\boldsymbol{u}_k, \boldsymbol{x}_{k-1}\right) + \boldsymbol{\epsilon}_k \quad \text{und} \tag{3.22}$$

$$\boldsymbol{z}_k = h\left(\boldsymbol{x}_k\right) + \boldsymbol{\delta}_k \tag{3.23}$$

verallgemeinert. Dieses hat Konsequenzen für die Anwendung des BAYES'schen Filters, da durch die nichtlineare Abbildung die Eigenschaft der Normalverteilung für die Posteriori Zustandswahrscheinlichkeit verloren geht. Damit lässt sich keine geschlossene Lösung für das BAYES'sche Filter finden. Das sogenannte Erweiterte KALMAN-Filter schafft hier durch die Linearisierung der Mess- und Zustandsübertragungsfunktion Gl. (3.22) und Gl. (3.23) Abhilfe. Die Linearisierung erfolgt durch Entwicklung einer TAYLOR-Reihe mit Abbruch nach dem ersten Glied. Damit ergibt sich

$$g\left(\boldsymbol{u}_k, \boldsymbol{x}_{k-1}\right) \approx \underbrace{g\left(\boldsymbol{u}_k, \boldsymbol{\mu}_{k-1}\right)}_{:=\boldsymbol{\mu}_k^-} + \underbrace{\frac{\partial g\left(\boldsymbol{u}_k, \boldsymbol{\mu}_{k-1}\right)}{\partial \boldsymbol{x}_{k-1}}}_{:=\boldsymbol{G}_k}\left(\boldsymbol{x}_{k-1} - \boldsymbol{\mu}_{k-1}\right) \quad \text{und} \tag{3.24}$$

$$h(\boldsymbol{x}_k) \approx h\left(\boldsymbol{\mu}_k^-\right) + \underbrace{\frac{\partial h\left(\boldsymbol{\mu}_k^-\right)}{\partial \boldsymbol{x}_k}}_{:=\boldsymbol{H}_k}\left(\boldsymbol{x}_k - \boldsymbol{\mu}_k^-\right) \quad . \tag{3.25}$$

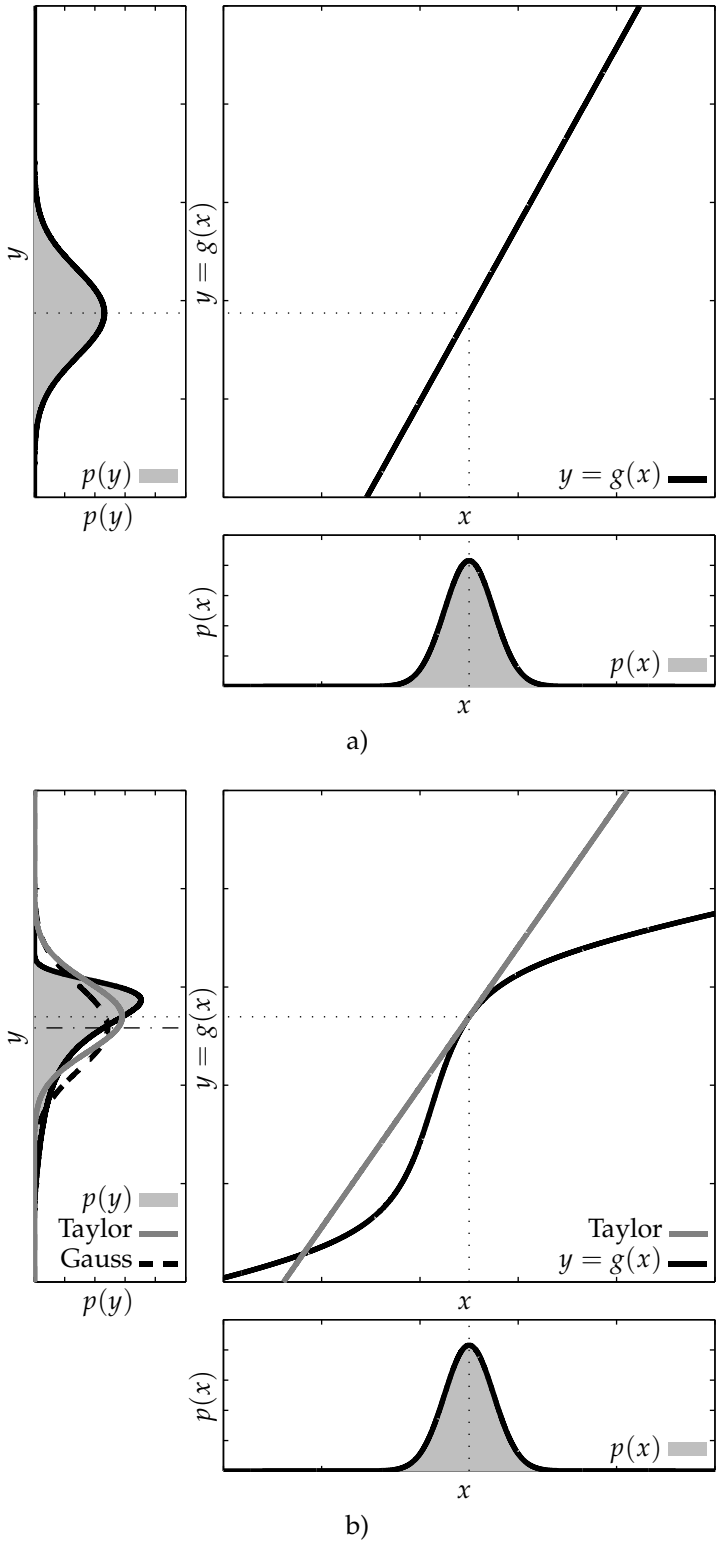


Bild 3.1: Transformation einer GAUSS'schen Zufallsvariable mit a) linearer Abbildung und b) nichtlinearer Abbildung (im Arbeitspunkt linearisiert).

In Bild 3.1b wird der Effekt einer nichtlinearen Abbildung einer normalverteilten Ausgangsverteilung deutlich. Durch die Linearisierung um den Mittelwert μ ergibt sich die mit „TAYLOR“ gekennzeichnete Abbildungsfunktion, mit deren Hilfe die Zielverteilung durch eine Normalverteilung angenähert wird. Zwar lässt sich die echte Wahrscheinlichkeitsdichtefunktion auf diese Weise nicht bestimmen, doch ergibt sich auf diese Weise wieder die ressourcenschonende Möglichkeit zur Berechnung einer geschlossenen Lösung für die Transformation der parametrischen Wahrscheinlichkeitsdichtefunktion. Zusätzlich ist hier die GAUSS'sche Verteilung skizziert, die die echt nichtlinear abgebildete Verteilung annähert.

Bei Übertragung von Gl. (3.24) und Gl. (3.25) in die Schreibweise der Wahrscheinlichkeitsdichtefunktionen ergeben sich

$$\begin{aligned} p\left(\boldsymbol{x_k}|\boldsymbol{u}_k, \boldsymbol{x}_{k-1}\right) \approx\ & \det(2\pi \boldsymbol{R}_k)^{-\frac{1}{2}} \\ & \exp\left(-\frac{1}{2}\left[\boldsymbol{x}_k - g(\boldsymbol{u}_k, \boldsymbol{\mu}_{k-1}) - \boldsymbol{G}_k(\boldsymbol{x}_{k-1} - \boldsymbol{\mu}_{k-1})\right]^{\mathrm{T}}\right. \\ & \qquad \left.\boldsymbol{R}_k^{-1}\left[\boldsymbol{x}_k - g(\boldsymbol{u}_k, \boldsymbol{\mu}_{k-1}) - \boldsymbol{G}_k(\boldsymbol{x}_{k-1} - \boldsymbol{\mu}_{k-1})\right]\right) \end{aligned} \tag{3.26}$$

und

$$\begin{aligned} p\left(\boldsymbol{z}_k|\boldsymbol{x}_k\right) \approx\ & \det(2\pi \boldsymbol{Q}_k)^{-\frac{1}{2}} \\ & \exp\left(-\frac{1}{2}\left[\boldsymbol{z}_k - h(\bar{\boldsymbol{\mu}_k}) - \boldsymbol{H}_k(\boldsymbol{x}_k - \bar{\boldsymbol{\mu}_k})\right]^{\mathrm{T}}\right. \\ & \qquad \left.\boldsymbol{Q}_k^{-1}\left[\boldsymbol{z}_k - h(\bar{\boldsymbol{\mu}_k}) - \boldsymbol{H}_k(\boldsymbol{x}_k - \bar{\boldsymbol{\mu}_k})\right]\right) \quad . \end{aligned} \tag{3.27}$$

Die aus der Linearisierung der Zustandsgleichung und Messgleichung resultierenden Matrizen $\boldsymbol{G}_k$ und $\boldsymbol{H}_k$ werden als JACOBI-Matrizen bezeichnet, sie ersetzen die Systemmatrix $\boldsymbol{A}_k$ und die Messmatrix $\boldsymbol{C}_k$. Der Algorithmus für das Erweiterte KALMAN-Filter ist in Algorithmus 3.3 dargestellt. In Zeile 2 wird die nichtlineare Zustandsprädiktion durchgeführt. Anschließend ist die Linearisierung zu berechnen, um die JACOBI-Matrix $\boldsymbol{G}_k$ zu erhalten, die analog zur Systemmatrix $\boldsymbol{A}_k$ aus Algorithmus 3.2 in Zeile 3 eingesetzt wird, um die Kovarianzmatrix für den Zustand $\bar{\boldsymbol{\mu}_k}$ zu berechnen. Weiterhin ist die JACOBI-Matrix $\boldsymbol{H}_k$ zu berechnen, die zur Berechnung der KALMAN-Verstärkung $\boldsymbol{K}_k$ in Zeile 4 benötigt wird. Die nichtlineare Messgleichung ersetzt im Korrekturschritt (Zeile 5) die lineare Messgleichung aus Algorithmus 3.2. Schließlich wird die verringerte Varianz des Zustands in der Kovarianzmatrix abgelegt.

Zusammenfassend handelt es sich beim KALMAN-Filter um eine hocheffiziente Methode, um bei normalverteiltem Zustand und für lineare Systemgleichungen

Algorithmus 3.3 Das Erweiterte KALMAN-Filter für nichtlineare Zustands- und Messgleichung.

1: ExtKalmanFilter($\mu_{k-1}, \Sigma_{k-1}, u_k, z_k$)
2: $\quad \bar{\mu}_k = g\left(u_k, \mu_{k-1}\right)$
3: $\quad \bar{\Sigma}_k = G_k \Sigma_{k-1} G_k^{\mathrm{T}} + R_k$
4: $\quad K_k = \bar{\Sigma}_k H_k^{\mathrm{T}} \left(H_k \bar{\Sigma}_k H_k^{\mathrm{T}} + Q_k\right)^{-1}$
5: $\quad \mu_k = \bar{\mu}_k + K_k \left(z_k - h_k\left(\bar{\mu}_k\right)\right)$
6: $\quad \Sigma_k = \left(I - K_k H_k\right) \bar{\Sigma}_k$
7: $\quad$ **return** μ_k, Σ_k
8: **end**

einen Folgezustand zu berechnen. Die Beschränkung auf lineare Systemgleichungen lässt sich durch Einsatz des Erweiterten KALMAN-Filters für linearisierbare Systemgleichungen aufweichen. Die Annahme von normalverteilten Zustandsvariablen ist jedoch grundlegend für die Bestimmung einer geschlossenen Lösung bei der Zustandstransformation, so dass das KALMAN-Filter stets auf unimodale Wahrscheinlichkeitsdichtefunktionen beschränkt bleibt. [Welch und Bishop, 2001] Um auch mehrdeutige Situationen einschätzen zu können, müssen andere Methoden eingesetzt werden. Im folgenden Abschnitt 3.4 wird mit dem Partikelfilter eine Methode vorgestellt, die sich auf beliebige Wahrscheinlichkeitsdichtefunktionen und Systemgleichungen anwenden lässt.

3.4 Histogrammfilter und Partikelfilter

Der in Abschnitt 3.2.3 beschriebene Ansatz des BAYES'schen Filters ermöglicht die exakte Berechnung der Dichte der Zustandswahrscheinlichkeit einer Zufallsvariable über den gesamten Zustandsraum. Die analytische Lösung dieser Gleichungen ist jedoch nur für einige Spezialfälle bekannt. So setzt das KALMAN-Filter aus Abschnitt 3.3 normalverteilte, lineare Modelle voraus. Für beliebige, z.B. multimodale Wahrscheinlichkeitsdichten, bei denen mehrere lokale Maxima existieren, stellen sich die Berechnung von Gl. (3.15) und Gl. (3.16) für praktische Zwecke als unlösbare Probleme heraus, denn der Rechenaufwand erhöht sich exponentiell mit der Größe und Dimension des Zustandsraums. Einen vielversprechenden Lösungsansatz bieten an dieser Stelle die nachfolgend dargestellten so genannten sequentiellen MONTE-CARLO-Methoden.

Eine leistungsfähige Alternative zu den KALMAN-Filtern stellen die nicht parametrischen Filter dar. Nichtparametrische Filter verzichten zur Beschreibung der Wahrscheinlichkeitsdichtefunktion auf eine vorbestimmte Funktion, deren Parameter zur

Beschreibung des Zustands angepasst werden. Sie nähern den Verlauf der Wahrscheinlichkeitsdichtefunktion durch eine begrenzte Anzahl von Werten an. Für dieses Vorgehen gibt es zwei wesentliche Ansätze. Zum einen lässt sich durch eine Diskretisierung des Zustandsraums die Wahrscheinlichkeitsdichtefunktion durch eine endliche Anzahl von Werten beschreiben, die jeweils für ein Intervall des Zustandsraums eine kumulative Wahrscheinlichkeit angeben. Die Diskretisierung kann dabei mit konstanter Intervallgröße oder mit dynamisch angepasster Intervallgröße erfolgen. Letztere ermöglicht eine bessere Auflösung in Bereichen hoher Wahrscheinlichkeiten. Das zugehörige Filter wird auch als Histogrammfilter bezeichnet. Zum zweiten lässt sich die Wahrscheinlichkeitsdichtefunktion durch eine Summe von zufällig gewählten Hypothesen (Stichproben) über den Zustand nachbilden, hierbei stellt die Dichte der Hypothesen bezüglich des Zustands ein Maß für die Wahrscheinlichkeit eines Zustands dar. Die Hypothesen werden üblicherweise als Partikel bezeichnet, das entsprechende Filter heißt Partikelfilter.

Als Beispiel für die vorteilhafte Repräsentation der Wahrscheinlichkeitsdichtefunktion durch eine endliche Anzahl von Werten wird in Bild 3.2 die Abbildung einer Partikelwolke durch eine nichtlineare Funktion dargestellt (die nicht dargestellten Abbildungseigenschaften mittels Histogramm sind dem ähnlich). Hier wird jedes einzelne Partikel gemäß der nichtlinearen Transformation in die Zielverteilung übertragen. Dadurch ergibt sich insbesondere im Vergleich zum parametrischen Vorgehen aus Abschnitt 3.3 eine gute Nachbildung der tatsächlichen Zielverteilung, deren Genauigkeit von der Anzahl der Partikel abhängt.

Für die skizzierten nichtparametrischen Methoden gilt, dass bei einer feinen Diskretisierung des Zustandsraums, bzw. durch eine hohe Anzahl von Partikeln, die tatsächliche Wahrscheinlichkeitsdichtefunktion angenähert werden kann. Der Vorteil der Bayes'schen Filter, die auf der vorgenannten Zustandsbeschreibung beruhen ist, dass keine stark einschränkenden Annahmen über die Form der Funktion getroffen werden müssen. Damit können insbesondere multimodale Verteilungen abgebildet werden, die gerade bei großen Unsicherheiten oder Mehrdeutigkeiten auftreten können. Zwar ist der rechnerische Aufwand einer Implementierung dieser Filter höher als bei parametrischen Filtern, doch bietet sich die Möglichkeit, die Zahl der Parameter (Schrittweite der Diskretisierung oder Anzahl der Partikel) an die Anforderungen der Aufgabe anzupassen. Zum einen kann bei komplexen Wahrscheinlichkeitsdichtefunktionen die Auflösung zur Laufzeit vergrößert werden, außerdem kann die Auflösung an die zur Verfügung stehenden Ressourcen (Rechenkapazität) angepasst werden.

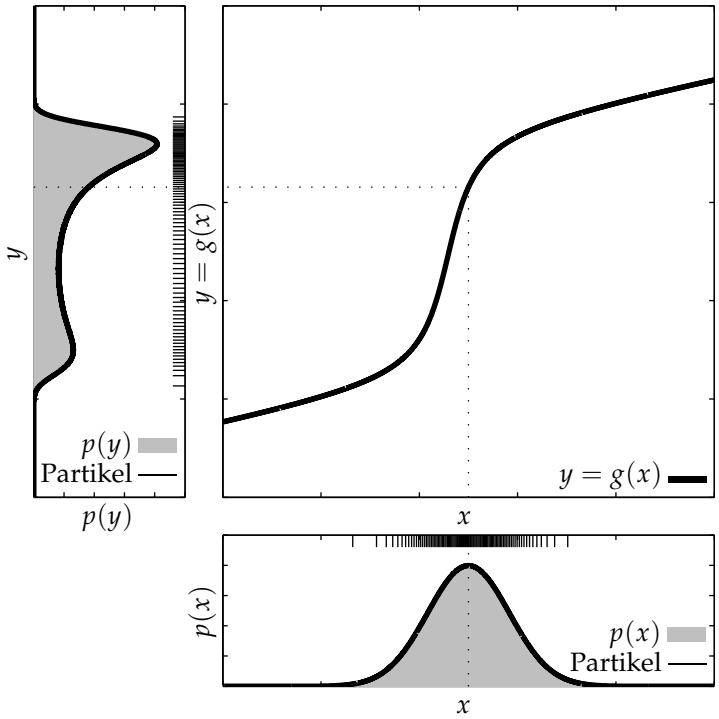


Bild 3.2: Transformation einer GAUSS'schen Zufallsvariable durch das Partikelfilter.

Das Histogrammfilter

Das oben dargestellte BAYES'sche Filter wird im Folgenden auf einen diskreten Zustandsraum angewandt. Für die Abbildung durch eine endliche Anzahl von Werten geht das Integral in Gl. (3.15) in eine Summe über

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{k-1\ldots0}) = \sum_{\boldsymbol{x}} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{k-1\ldots0}) \quad . \tag{3.28}$$

Damit lässt sich BAYES Filter in allgemeiner Form als Histogrammfilter herleiten. Wie in Algorithmus 3.4 dargestellt, lässt sich das Filter nach Übergang auf eine endliche Auflösung berechnen. Der zu schätzende Zustand wird durch eine diskrete Wahrscheinlichkeitsverteilung beschrieben. Für jeden Zustand $\boldsymbol{x}_{k,[m]}$ des Zustandsraums gibt es eine zugehörige Wahrscheinlichkeit $p_{k,[m]}$, die Menge der Wahrscheinlichkeiten $p_{k-1,[m]}$ ergibt die Wahrscheinlichkeitsdichte $\{p_{k-1,[m]}\}$ im Zeitschritt $k-1$, die mit dem Eingangsvektor $\boldsymbol{u}_k$ und der aktuellen Messung als Eingang des diskreten BAYES'schen Filters dienen. In Zeile 3 des Filters wird mittels der Systemgleichung eine Prädiktion für den diskreten Zustand $\bar{\boldsymbol{x}}_{k,[m]}$ auf Basis des Eingangsvektors berechnet. Dieser wird mit der Wahrscheinlichkeit für den Ausgangszustand gewichtet und aufsummiert. In Zeile 4 wird die Messung zur Korrektur mit einbezogen und die Wahrscheinlichkeit wird normiert.

Algorithmus 3.4 Das diskrete BAYES'sche Filter (Histogrammfilter) mit $\boldsymbol{x}_{[m]}$ und $\boldsymbol{x}_{[n]}$ als individuellen Zuständen des Zustandsraums, deren Wahrscheinlichkeit in $p_{k,[m]}$ bzw. $p_{k,[n]}$ abgelegt ist.

1: DISKRETESBAYESFILTER($\{p_{k-1,[m]}\}, \boldsymbol{u}_k, \boldsymbol{z}_k$)
2: **for all** m **do**
3: $\quad \bar{p}_{k,[m]} = \sum_n p\left(\boldsymbol{x}_{k,[m]} | \boldsymbol{u}_k, \boldsymbol{x}_{k-1,[n]}\right) p_{k-1,[n]}$
4: $\quad p_{k,[m]} = \eta p\left(\boldsymbol{z}_k | \boldsymbol{x}_{k,[m]}\right) \bar{p}_{k,[m]}$
5: **end for**
6: **return** $\{p_{k,[m]}\}$
7: **end**

Das Partikelfilter

Das Partikelfilter ist dem Histogrammfilter in der gegebenen Aufgabenstellung überlegen, da die Anpassung an die gegebenen Ressourcen hier einfach zu implementieren ist und die Abbildung der Wahrscheinlichkeitsdichtefunktion in Bereichen hoher Wahrscheinlichkeiten prinzipbedingt eine hohe Auflösung besitzt. Die Funktionsweise des Partikelfilters ist in Algorithmus 3.5 dargestellt und soll im Folgenden erläutert werden.

Die gesuchte a posteriori Wahrscheinlichkeitsdichtefunktion $p(\boldsymbol{x}_k | \boldsymbol{z}_{k...0})$ stellt das Wissen über den Zustand $\boldsymbol{x}_k$ im Zeitschritt k dar und wird im Folgenden mit $\tilde{\boldsymbol{x}}_k$ bezeichnet. Die Grundidee des Partikelfilters ist die Repräsentation von $\tilde{\boldsymbol{x}}_k$ durch eine Menge von zufälligen Zustandsstichproben, die mit der Wahrscheinlichkeit $p(\boldsymbol{x}_k | \boldsymbol{z}_{k...0})$ ausgewählt werden. Somit wird diese Wahrscheinlichkeitsdichtefunktion durch die Dichte der Zustandsstichproben approximiert (siehe Bild 3.2). Daraus leitet sich der o. g. Vorteil des Partikelfilters ab, denn die Genauigkeit, mit der $\tilde{\boldsymbol{x}}_k$ approximiert wird ist proportional zu der Wahrscheinlichkeitsdichte. Die Zustandsstichproben $\boldsymbol{x}_{k,[m]}$ sind konkrete Instanzen des Zustands $\boldsymbol{x}_k$, sie stellen Hypothesen über die möglichen Zustände eines Systems dar. Eine Partikelmenge $\boldsymbol{\chi}_k$ besteht aus einer großen Anzahl von Partikeln (typische Größenordnung $M > 1000$) $\boldsymbol{\chi}_k = \{\boldsymbol{x}_{k,[1]}, \ldots, \boldsymbol{x}_{k,[M]}\}$.

Gemäß der BAYES'schen Filterrekursion berechnet der Partikelfilteralgorithmus die Partikelmenge $\boldsymbol{\chi}_k$ aus der vorherigen Menge $\boldsymbol{\chi}_{k-1}$, dem aktuellen Eingang $\boldsymbol{u}_k$ und der aktuellen Sensormessung $\boldsymbol{z}_k$ (vergleiche Algorithmus 3.4).

In Algorithmus 3.5 ist ein Zeitschritt für das Partikelfilter dargestellt. Zunächst wird in Zeile 4 der Prädiktions-Schritt ausgeführt. Dafür wird jedes einzelne Partikel $\boldsymbol{x}_{k-1,[m]}$ der Partikelmenge $\boldsymbol{\chi}_{k-1}$ unter Anwendung des Bewegungsmodells $\boldsymbol{x}_k = g(\boldsymbol{u}_k, \boldsymbol{x}_{k-1})$ aus Gl. (3.22) in die a priori Schätzung $\bar{\tilde{\boldsymbol{x}}}_k = p(\boldsymbol{x}_k | \boldsymbol{u}_k, \boldsymbol{x}_{k-1})$ transformiert. So entsteht die neue Partikelmenge $\bar{\boldsymbol{\chi}}_k$, die die Wahrscheinlichkeitsdichte

Algorithmus 3.5 Das Partikelfilter als Variante des Bayes'schen Filters.

1: PartikelFilter(χ_{k-1}, u_k, z_k)
2: $\bar{\chi}_k = \chi_k = \emptyset$
3: **for** $m = 1$ **to** M **do**
4: sample $\bar{x}_{k,[m]} \sim p\left(x_k | u_k, x_{k-1,[m]}\right)$
5: $w_{k,[m]} = p\left(z_k | \bar{x}_{k,[m]}\right)$
6: $\bar{\chi}_k = \bar{\chi}_k + \langle \bar{x}_{k,[m]}, w_{k,[m]} \rangle$
7: **end for**
8: **for** $m = 1$ **to** M **do**
9: draw n with probability $\sim w_{k,[n]}$
10: add $x_{k,[n]}$ to χ_k
11: **end for**
12: **return** χ_k
13: **end**

der Zustandsvorhersage repräsentiert.

Im zweiten Schritt wird die Sensormessung eingearbeitet. Zu diesem Zweck werden in Zeile 5 für jede vorhergesagte Hypothese $\bar{x}_{k,[m]}$ mit Hilfe der Messgleichung (3.23) $z_k = h(x_k)$ virtuelle Messdaten und mit einem Sensormodell zugehörige Wahrscheinlichkeitsdichten für eine Sensormessung $p(z_{k,[m]}|\bar{x}_{k,[m]})$ erzeugt. Das Gewicht eines Partikels bestimmt sich nun durch Einsetzen der realen Messung in die Wahrscheinlichkeitsdichtefunktion des Partikels:

$$w_{k,[m]} = p(z_{k,[m]}|\bar{x}_{k,[m]}) \quad . \tag{3.29}$$

Wenn die Elemente der Partikelwolke $\bar{x}_{k,[m]}$ mit $w_{k,[m]}$ gewichtet werden, repräsentieren sie bereits die gesuchte Wahrscheinlichkeitsdichtefunktion $\tilde{x}_k$.

Die reine Gewichtung der Partikel führt jedoch zu einer Verarmung der Vielfalt der effektiven Partikel: Nach nur wenigen Iterationsschritten konzentriert sich das gesamte Gewicht auf wenige, bzw. nur ein Partikel. Der Effekt der Diversifikation durch eine Addition von Rauschen im Bewegungsmodell geht dann verloren und das Partikelfilter arbeitet nicht mehr korrekt. Im letzten Schritt des Filters wird daher die Approximation von $\tilde{x}_k$ durch die Partikeldichte mit gleich gewichteten Partikeln mittels eines Resampling-Schritts wiederhergestellt.

Es werden N Partikel $\bar{x}_{k,[n]}$ mit der Wahrscheinlichkeit proportional zu $w_{k,[n]}$ aus der Menge $\bar{\chi}_k$ ausgewählt, kopiert und zu der neuen Partikelmenge χ_k hinzugefügt. Dieser Algorithmus arbeitet nach dem Prinzip der *natürlichen Auslese*, denn die wahrscheinlichsten Zustandsstichproben werden häufiger ausgewählt, wogegen die von der Sensormessung nicht bestätigten Partikel aussterben. Auf diese

Weise konzentrieren sich die Partikel in der Nähe des wahren Zustands $\boldsymbol{x}_k$. Die Partikeldichte dieser neu erzeugten Menge ist somit wieder proportional zu $\tilde{\boldsymbol{x}}_k$. Als beispielhaftes Resampling-Verfahren ist in Algorithmus 3.6 das *Low Variance Resampling* dargestellt. Es wird eine Zufallszahl r_{rand} aus dem Intervall $(0; M^{-1})$ gezogen,

Algorithmus 3.6 Das *Low Variance Resampling* lässt sich effizient umsetzen. [Thrun u. a., 2005]

```
1:  LowVarianceResampler(χ_k)          ▷ Please note: χ_k = {⟨x̄_{k,[m]}, w_{k,[m]}⟩}
2:      χ̃_k = ∅
3:      r_rand = rand(0; M^{-1})
4:      c = w_{k,[1]}
5:      n = 1
6:      for m = 1 to M do
7:          U = r_rand + (m − 1) · M^{-1}
8:          while U > c do
9:              n = n + 1
10:             c = c + w_{k,[n]}
11:         end while
12:         add x̄_{k,[n]} to χ̃_k
13:     end for
14:     return χ̃_k
15: end
```

hier entspricht M der Zahl der im folgenden Abtastschritt zu berücksichtigenden Partikel. In der **for**-Schleife wird in jeder Iteration der Betrag $(m-1) \cdot M^{-1}$ auf die Zufallszahl r_{rand} addiert, womit dann genau das erste Partikel n gesucht wird, für das gilt, dass die Summe aller Partikelgewichte bis einschließlich $w_{k,[n]}$ größer oder gleich $r_{\text{rand}} + (m-1) \cdot M^{-1}$ ist. Sobald das richtige Partikel gefunden ist, wird es in Zeile 12 der neuen Partikelwolke $\tilde{\chi}_k$ hinzugefügt. Das Verfahren stellt sicher, dass jedes Partikel der Ausgangsmenge genau einmal für die neue Partikelwolke erwogen wird und dass gemäß der Partikelgewichte gesampelt wird.

Die Funktionsweise des Partikelfilters wird in Bild 3.3 verdeutlicht. Dargestellt ist eine Iterationsschleife des Filters. Gestartet wird in Bild 3.3a mit der Wahrscheinlichkeitsdichtefunktion des Zustands des vorhergehenden Iterationsschritts $p(x_{k-1})$, die hier als Wahrscheinlichkeitsdichtefunktion $p(x_{k-1})$ und als Partikelwolke $\{x_{k-1,[m]}\}$ dargestellt ist. Auf die Partikelwolke wird nun die Prädiktion angewendet. Es ergibt sich die in Bild 3.3b dargestellte neue Partikelwolke $\{\bar{x}_{k,[m]}\}$, die der Anschaulichkeit halber zusätzlich als Wahrscheinlichkeitsdichtefunktion $p(x_k|u_k, x_{k-1})$ dargestellt ist, als solche aber im Partikelfilter nicht vorhanden ist.

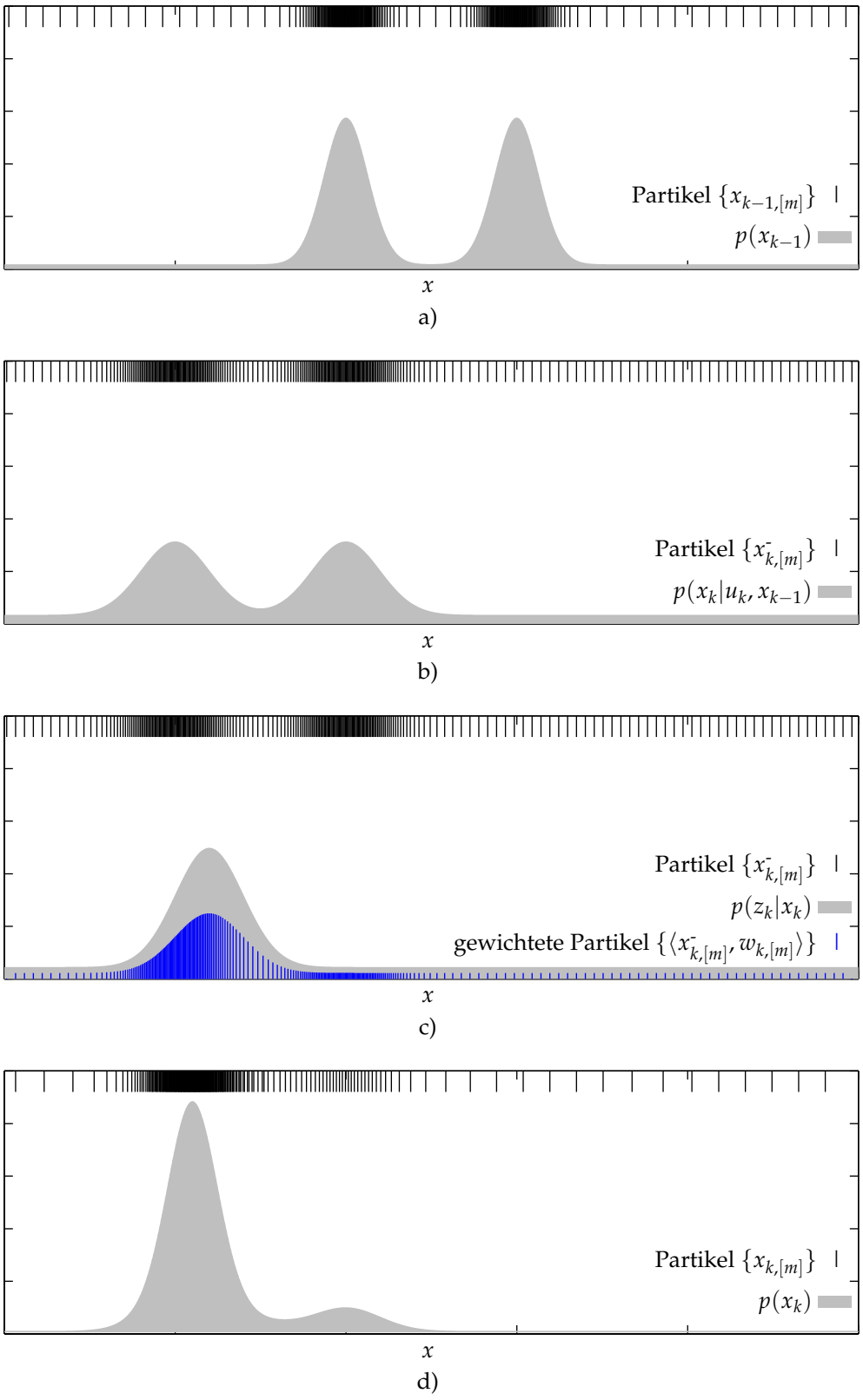


Bild 3.3: Die Funktionsweise des Partikelfilters umfasst eine Ausgangsverteilung (a), die Anwendung des Systemmodells (a → b), die Integration der Messung und Gewichtung der Partikel (c) und den Resampling-Prozess zur Wiederherstellung einer gleich gewichteten Partikelwolke (d).

Die so entstandene Partikelwolke wird mit Hilfe der Wahrscheinlichkeitsdichtefunktion der Messung $p(z_k|x_k)$ gewichtet und es entsteht die in Bild 3.3c dargestellte gewichtete Partikelwolke $\{\langle x^-_{k,[m]}, w_{k,[m]}\rangle\}$. In Bild 3.3d dargestellt ist, wie durch ein Resampling schließlich wieder eine Partikelwolke erzeugt wird, die die Wahrscheinlichkeitsdichtefunktion durch die Partikeldichte abbildet $\{x_{k,[m]}\}$. Da das Resampling auf den Partikeln der a priori geschätzten Partikel beruht und die Partikel mehrfach gezogen werden entsteht beim Betrachten zunächst nicht der Eindruck, dass die Wahrscheinlichkeitsdichtefunktion der Messung angemessen berücksichtigt wird, weil die mehrfachen Partikel exakt übereinander liegen. Erst durch die Überlagerung eines zusätzlichen Rauschens wird die Wahrscheinlichkeitsdichtefunktion in der gewählten Darstellung visualisiert.

4 Versuchsumgebung

Die experimentelle Erprobung ist elementarer Bestandteil bei der Entwicklung von Steuerungs- und Regelungsalgorithmen. Während Wahrnehmung der und Interaktion mit der Umwelt in den folgenden Kapiteln adressiert werden, wird in diesem Kapitel zunächst die Versuchsumgebung dargestellt, da die entwickelten Methoden zwar für die allgemeine Anwendung konzipiert werden, die Erprobung in Simulation und Experiment aber auf eine konkrete Versuchsplattform zurückgreifen muss. Die Steuerung und Regelung des Bewegungsmechanismus ist abhängig von der Kinematik. Weiterhin sind Informationen über den Systemzustand erforderlich, die durch die internen Sensoren bestimmt werden, dabei ergibt sich eine Ausstattungsabhängigkeit.

In der vorliegenden Arbeit kommt die in Bild 4.1 dargestellte mobile Plattform zum Einsatz. Der Fortbewegungsmechanismus basiert auf vier angetriebenen und

Bild 4.1: Die mobile Roboterplattform mit Antrieben, Energieversorgung, Mikrocontrollern, Prozessrechnern und Sensorik zur Erfassung der Umgebung.

gelenkten Radmodulen, die den Rahmen der Plattform mit den Traktionsbatterien zur Energieversorgung tragen. Die Ansteuerung und das Auslesen der mit Winkelencodern ausgestatteten Lenk- und Antriebsmotoren erfolgt durch einen Mikrocontroller. Dieser wertet neben den Radsensoren verschiedene Endlagenschalter und den Zustand einer um den Rahmen des Roboters installierten berührungsempfindlichen Leiste aus und steuert die Leistungselektronik für die Aktoren. Da die Verletzung von Stellgrößenbeschränkungen bei der gegebenen Kinematik zu einer Verletzung der durch die Redundanz hervorgerufenen nichtholonomen Zwangs-

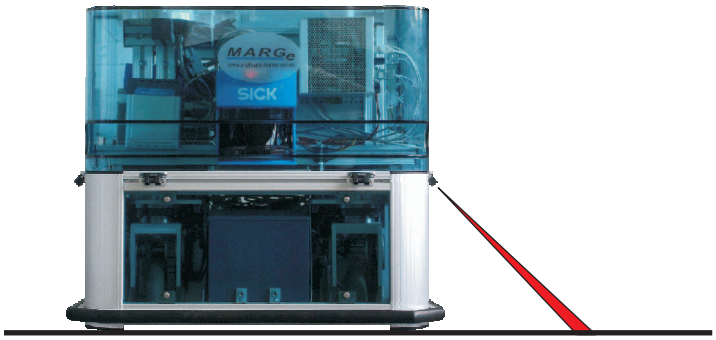


Bild 4.2: Zur Überwachung des Untergrundes im Nahbereich des Roboters ist dieser mit Infrarot-Triangulationssensoren ausgestattet. Diese erkennen negative Hindernisse wie Treppenstufen und Absätze.

bedingungen führt, überwacht der Mikrocontroller die eingehenden Steuerbefehle gemäß [Blume und Heimann, 2007] auf Ausführbarkeit und begrenzt bei Bedarf die Steuerbefehle auf zulässige Werte. Einen weiteren Sicherungsmechanismus stellen abstandsmessende Infrarot-Triangulationssensoren dar. Am Rahmen des Roboters installiert überwachen diese den Untergrund im Nahbereich des Roboters, um bei Detektion negativer Hindernisse (Absätze, Stufen, etc.) einen Nothalt auszulösen. Schließlich gibt der Mikrocontroller über ein LCD-Display Auskunft über den gegenwärtigen Zustand des Roboters. Die Kommunikation zwischen der Leitrechnerebene und dem Mikrocontroller erfolgt über den CAN-Bus. Im Fahrzustand erfolgt die Vorgabe von Sollwerten im Takt von 50 ms. Die Bedienung des Leitrechners erfolgt von einem Standard-PC aus über eine (kabellose) Netzwerkverbindung.

Zur Erfassung der Umgebung ist der Roboter mit in der Robotik weit verbreiteten Laserentfernungsmesssystemen der Fa. Sick ausgestattet. Diese tasten die Umgebung horizontal in einer Höhe von ca. 40 cm über dem Boden ab und liefern einen Fächer von Messungen mit dem jeweiligen Abstand zum nächstliegenden reflektierenden Objekt (siehe auch Abschnitt 5.2.2). Die Sensoren haben einen Messbereich von jeweils 180° und sind in entgegengesetzter Richtung ausgerichtet. Durch den Abstand zwischen den Sensoren entsteht in der Messung seitlich vom Roboter eine Lücke von ca. 50 cm, die nicht von den Sensoren überwacht werden kann (vergl. Bild 4.3).

Die mobile Plattform bietet Anschlussmöglichkeiten für einen Leichtbauroboterarm, mit dem eine koordinierte Umweltinteraktion möglich ist. [Hornung, 2007]

Wesentliche Funktionsbestandteile und Leistungsmerkmale werden durch Software bestimmt. Daher wird in diesem Kapitel zunächst die Softwarearchitektur beschrieben. Dazu gehört auch das Steuerungs- und Kommunikationskonzept für

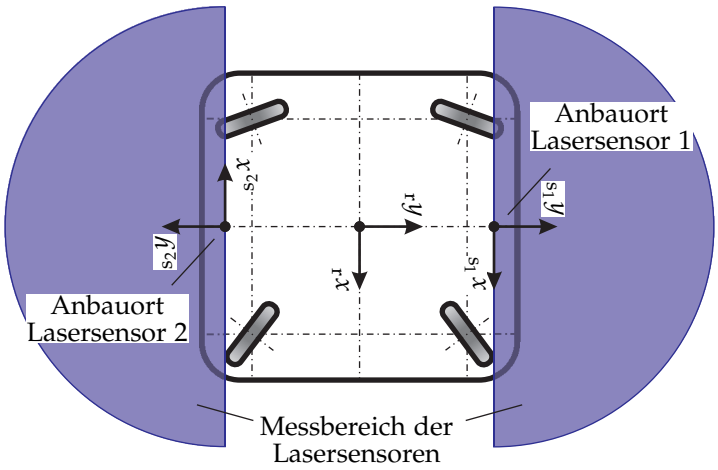


Bild 4.3: Schematische Aufsicht auf den Versuchsträger: Der Messbereich der Laserentfernungsmesssysteme beträgt jeweils 180°. Durch den Abstand zwischen den Scannern entsteht ein Bereich, der nicht eingesehen werden kann.

Regelkreise höherer Hierarchieebenen. Im Anschluss daran wird die Kinematik der mobilen Plattform dargestellt. Schließlich folgt die Beschreibung der Bahnregelung. Die Darstellung von Simulationsergebnissen und experimentellen Ergebnissen erfolgt jeweils im Anschluss an die methodischen Darstellungen der einzelnen Kapitel.

4.1 Softwarearchitektur

Ein autonom agierendes Robotersystem hat eine Vielzahl an einzelnen Aufgaben zu lösen. Hierbei werden hohe Anforderungen an die Datenverarbeitung gestellt. Um die Echtzeitfähigkeit des Gesamtsystems gewährleisten zu können, müssen zum einen Prozessrechner und Betriebssystem diese Eigenschaft zur Verfügung stellen, zum anderen muss bei der Umsetzung der Algorithmen zur Ausführung auf dem Prozessrechner auf eine geeignete Prozesssynchronisation unter den Gesichtspunkten Effizienz und Fehlertoleranz geachtet werden. Eine monolithisch aufgebaute Software hat zwar Vorteile bezüglich der Effizienz, die Komplexität und der Aufwand bei der Softwareentwicklung und Sicherstellung von Fehlertoleranz ist jedoch enorm und wächst mit steigendem Funktionsumfang überproportional an. Eine modular aufgebaute Software hilft dagegen, der Komplexität Herr zu werden, und ermöglicht zum anderen, die Softwareentwicklung mit der Möglichkeit der Absicherung von Eigenschaften einzelner Module durch entsprechende Testumgebungen zu vereinfachen. Die Modularität schafft zwar einen zusätzlichen Rechenaufwand

für die Prozesskommunikation, ermöglicht jedoch die Skalierung des Prozessrechnersystems durch den Einsatz von verteilten Systemen.

In der vorliegenden Arbeit kommt ein von Wulf u. a. [2003] entwickeltes modulares Softwarekonzept zum Einsatz. Das Softwaresystem stellt einen Mechanismus für eine paketbasierte Kommunikation über Mailboxen zur Verfügung. Das Modul ist ein Basiselement der Applikation. Jedes Modul stellt ein Objekt dar, wobei das Konzept der objektorientierten Programmierung greift. Die Kommunikation erfolgt zwischen Modulen des gleichen Prozessrechners, Modulen auf verteilten Systemen sowie nicht echtzeitfähigen Modulen ausschließlich über die Mailboxen. Die Daten der Module sind gegen den unkontrollierten Zugriff Dritter geschützt und die Interaktion erfolgt über definierte Schnittstellen. Für die Implementierung eines Moduls ist dabei unerheblich, ob das Modul des Kommunikationspartners auf dem gleichen System oder auf einem anderen System ausgeführt wird, da die Paketadressierung von einer Paket-Verwaltung aufgelöst wird. Eine Umsetzung des Softwarekonzept existiert für das Betriebssystem Linux mit RTAI[1]-Echtzeiterweiterung [RTAI, 2008; Mantegazza u. a., 2000] und für das Echtzeitbetriebssystem RTOS-UH[2] [RTOS-UH, 2008; Gerth, 1983; Wolter u. a., 2003]. Die Programmierung erfolgt für RTAI-Linux in der Programmiersprache C und C++ und für RTOS-UH unter PEARL90 [Werum und Windauer, 1978; DIN 66253-2, 1998]. Im nicht echtzeitfähigen Kontext besteht mittels der Programmiersprache Java, für die Implementierungen für eine Reihe von Betriebssystemen verfügbar sind, ebenfalls Zugang zum Kommunikationssystem.

Arbeitsumgebung

In Bild 4.4 sind die relevanten Module, die in der vorliegenden Arbeit entwickelt werden, mit ihrer Vernetzung untereinander dargestellt. Unterschieden wird hier nach Modulen im Echtzeitkontext, Modulen ohne Echtzeitanforderungen und Aktoren und Sensoren. Im Folgenden soll die Funktion der einzelnen Module kurz vorgestellt werden.

- Das Modul *Bedienerinterface* dient dem Nutzer zur Parametrierung des Systems, zur Vorgabe von Aufgaben und zur Überwachung des Systemzustands. Es stellt mittels eines graphischen Eingabe- und Anzeigebildschirms die Daten der einzelnen Module visuell aufbereitet dar.

- Das Modul *Datenlog* protokolliert die Prozessdaten ausgewählter Module, um den Prozessverlauf sichtbar machen zu können, und dient als Werkzeug bei der Funktionsentwicklung.

[1] Real Time Application Interface
[2] Real Time Operating System – University of Hannover

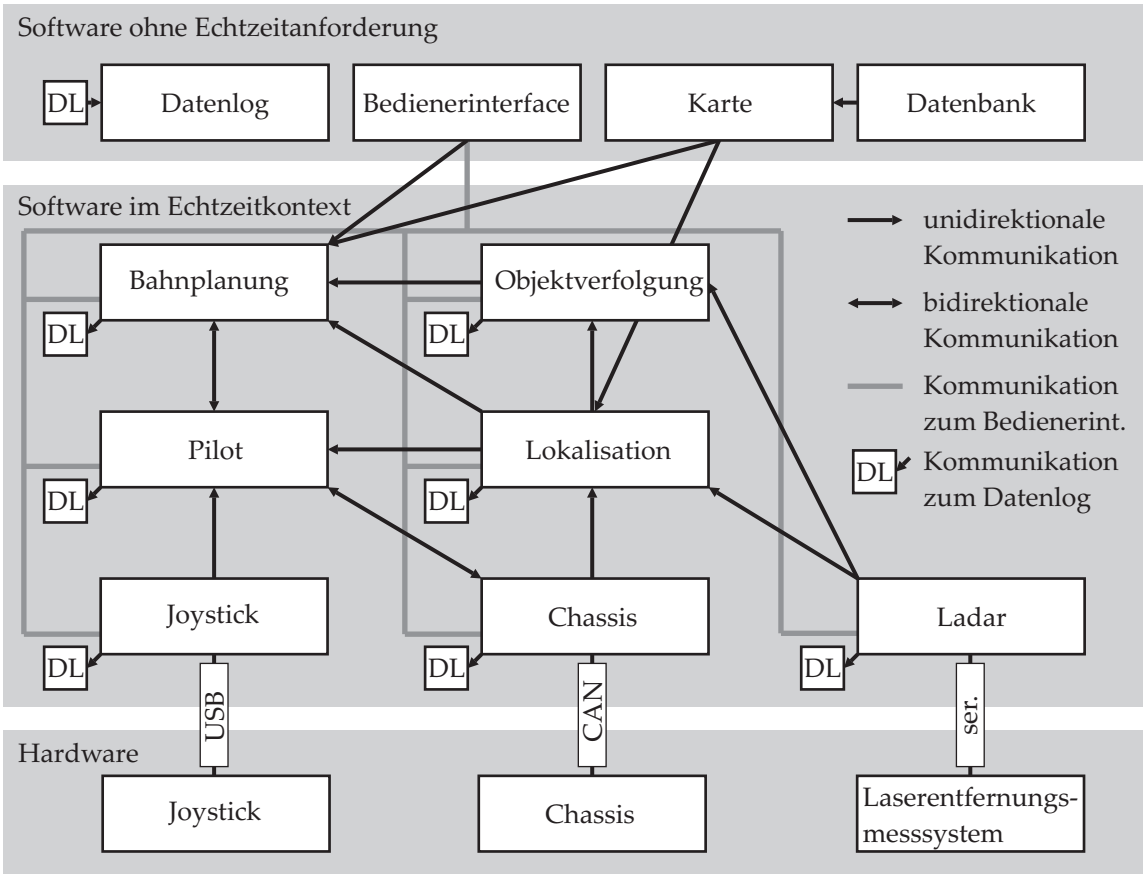


Bild 4.4: Die hier entwickelten Module sind sowohl im Echtzeitkontext, als auch im nicht echtzeitfähigen Kontext angeordnet. Für Sensoren und Aktoren gibt es jeweils ein Schnittstellenmodul, dass die Komponente in die Kommunikationsstruktur integriert.

- Das Modul *Chassis* stellt die elementaren Fahrfunktionen der mobilen Plattform bereit. Es bringt die Aktoren und Sensoren in einen fahrbereiten Zustand und leitet im Betrieb die Steuerbefehle höherer Regelkreisebenen in einem festen Takt durch den CAN-Bus zur Ausführung auf dem Mikrocontroller der Plattform weiter. Weiterhin fusioniert es die Messsignale aus der Plattform mittels eines Erweiterten KALMAN-Filters mit dem Steuerbefehl zu einem Zustandsvektor gemäß dem Unicycle-Modell (siehe Abschnitt 4.2).

- Das Modul *Ladar* parametriert das Laserentfernungsmesssystem und stellt dessen Messungen für andere Module bereit.

- Das Modul *Joystick* dient zur manuellen Steuerung des Robotersystems und stellt die Eingaben für lesende Module zur Verfügung.

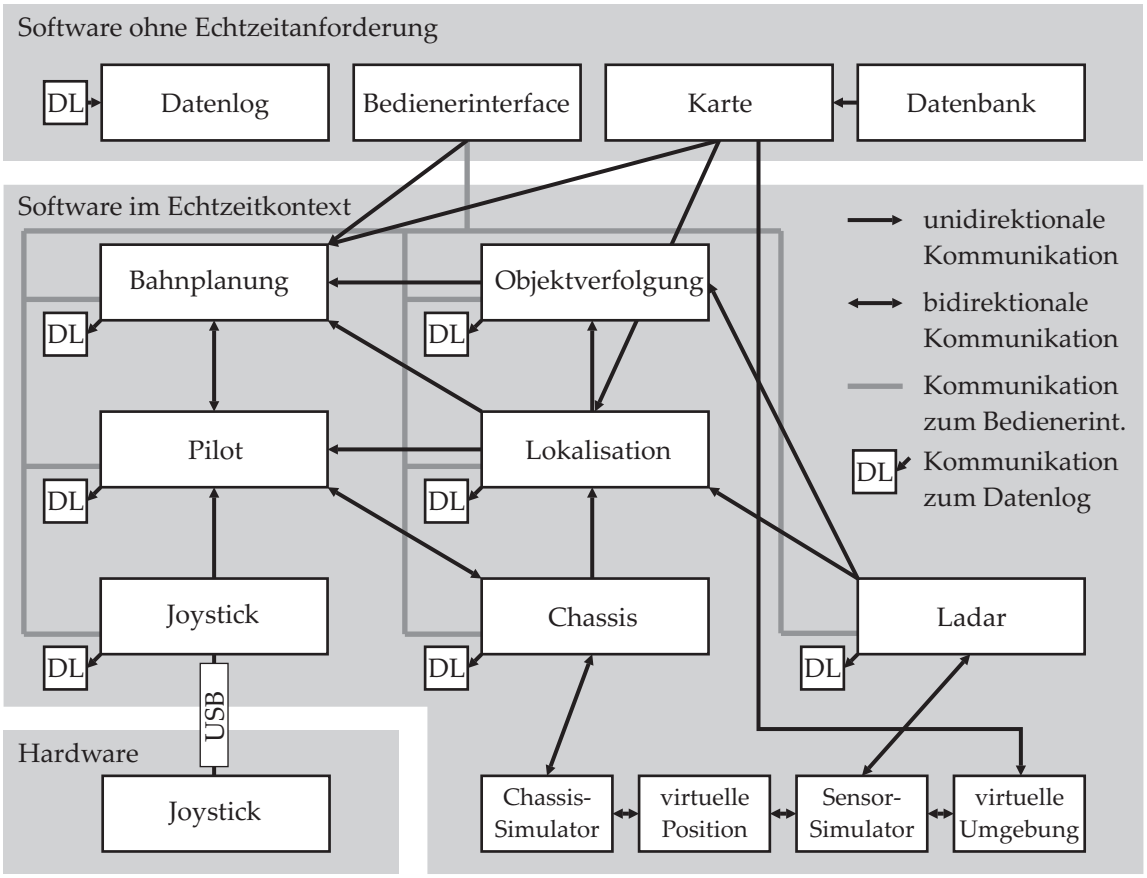


Bild 4.5: Die Steuerungsstruktur in der Simulation erfolgt durch einen Austausch der Module, die als Aktoren Einfluss auf den Zustand des Roboters haben, bzw. die als Sensoren den Zustand des Roboters wahrnehmen.

- Das Modul *Datenbank* dient zur Kommunikation mit einer MySql-Datenbank in der verschiedene Daten wie z. B. Umgebungskarten hinterlegt sind.

- Das Modul *Karte* stellt im laufenden Betrieb Umgebungskarten bereit, die entweder über eine Benutzerschnittstelle oder aus der Datenbank bezogen werden. Die Erstellung von Umgebungskarten aus Messungen wird in Abschnitt 5.3.2 dargestellt.

- Das Modul *Lokalisation* dient der Bestimmung des Aufenthaltsortes des Roboters innerhalb seiner Umgebung. Die Lokalisation erfolgt durch die Ermittlung der Position bezüglich einer Umgebungskarte, die aus dem Kartenmodul bezogen wird. Zur Bestimmung der Position wird die Monte-Carlo-Lokalisation eingesetzt (siehe Abschnitt 6.2). Als Eingangsgrößen dienen der Zustandsvektor aus dem Chassis-Modul und die Messungen des Laserentfer-

nungsmesssystems.

- Das Modul *Pilot* übernimmt die Funktion des Bahnreglers auf Grundlage der Vorgabe aus der Bahnplanung oder alternativ der manuellen Vorgabe des Joysticks. Bei der Bahnregelung wird mittels der Lokalisationsdaten die Position relativ zur Bahn bestimmt und ein Steuerbefehl für das Chassismodul generiert. Die Funktion des Bahnreglers wird in Abschnitt 4.3 beschrieben.

- Das Modul *Objektverfolgung* dient dazu, dynamische Objekte in der Umgebung des Roboters zu erkennen. Zu diesem Zweck wertet es die Daten des Laserentfernungsmesssystems aus. Um die Lage der Laserscans zueinander bestimmen zu können, greift es auf die Lokalisationsdaten zurück. Es bestimmt Merkmale die zu einem Objekt gehören und verfolgt diese mittels individueller Partikelfilter. Die Methoden zur Objektverfolgung werden in Kapitel 7 dargestellt.

- Das Modul *Bahnplanung* dient der Planung einer kollisionsfreien Bahn zwischen der Ausgangsposition und der durch den Nutzer einzugebenden Zielposition. Für die Kollisionsvermeidung mit statischen Objekten erfolgt die Suche innerhalb einer Umgebungskarte aus dem Kartenmodul. Um Kollisionsfreiheit mit dynamischen Objekten gewährleisten zu können, greift das Modul auf die Schätzung der dynamischen Objekte zurück. Da die Berechnung von Kollisionen mit dynamischen Objekten zeitabhängig ist, erfolgt über die Daten des Piloten ein stetiger Abgleich der aktuellen Position auf der Bahn. Die Beschreibung der Bahnplanung erfolgt in Kapitel 8.

Simulationsumgebung

Neben dem hohen Vernetzungsgrad stellt auch die Komplexität der Aufgabe eine Herausforderung dar. Die Bereitstellung eines Testszenarios zur Erprobung von Funktionen und Software im Versuchsbetrieb erfordert einen erheblichen Aufwand. Um diesen zu reduzieren, bietet es sich an Teile der Versuchsumgebung zu simulieren. Die modulare Softwarestruktur hilft hier, eine anwendungsnahe Simulationsumgebung zu schaffen: Es werden solche Module, die in direkter Interaktion mit der Umgebung stehen durch Simulatoren ersetzt (siehe Bild 4.5). Dieses erfolgt in der dargestellten Struktur für das Chassis-Modul und das Ladar-Modul. [Blume und Heimann, 2006]

Das Chassis-Modul leitet Steuerbefehle höherer Regelkreisebenen an die Hardware weiter, durch die Ausführung in den Aktoren erfolgt dann eine Rückwirkung auf den Roboterzustand und die Sensoren des Roboters. Der Mikrocontroller liefert gemessene Lenkwinkel und Radgeschwindigkeiten zurück. Im Simulator werden

nun aus den Steuerbefehlen virtuelle Sensorsignale generiert und als Messung zurückgegeben. Damit erfolgt die Schätzung des Plattformzustands wie in der realen Struktur mittels des Erweiterten KALMAN-Filters, so dass eine zum realen System vergleichbare Systemdynamik entsteht. Der Bahnregelkreis sowie die Regelkreise höherer Ordnung greifen auf die gleichen Schnittstellen zu, somit wird ein Funktionsablauf nahezu identisch zur realen Versuchsumgebung ermöglicht.

Um das Laserentfernungsmesssystem zu simulieren, sind zusätzliche Module erforderlich, da hier externe Merkmale gemessen werden. Das Sensorsystem gibt abhängig von der Lage im Raum einen Fächer von Messwerten aus, die den Abstand zum jeweils nächstliegenden Objekt beinhalten (siehe auch Abschnitt 5.2.2). Die Messungen sind also von der Lage im Raum und von der Beschaffenheit des Raumes abhängig. Beide Elemente müssen entsprechend simuliert werden. Die Lage im Raum wird durch eine Integration der Steuerbefehle an das Chassis-Modul berechnet. Diese virtuelle Position wird einem Simulationsmodul für die Laserentfernungsmessung zugeführt. Da sich der Roboter in der Ebene bewegt, ist eine zweidimensionale Nachbildung der Umgebung für eine Simulation ausreichend. Hier wird auf eine Rasterkarte aus dem Kartenmodul zurückgegriffen. Eine Rasterkarte beinhaltet für jede Zelle einen Wert, der die Wahrscheinlichkeit anzeigt, dass die Zelle belegt ist (vergleiche Abschnitt 5.1). Nachdem der virtuelle Sensor in der Karte positioniert wurde, wird ein virtueller Strahl vom Sensorursprung in Richtung der momentanen Sensorausrichtung ausgesandt indem eine Linie in der Rasterkarte verfolgt wird. Überstreicht diese Linie nun eine Zelle, deren Wert einen Schwellwert übersteigt, so wird eine Kollision festgestellt und die Entfernung kann bestimmt werden. In Abschnitt 6.2.2 ist eine effiziente Implementierung zur Strahlverfolgung dargestellt. Der Vorgang wird für eine der realen Sensorwinkelauflösung entsprechende Zahl von Orientierungen wiederholt, bis ein vollständiger Messdatensatz ermittelt wurde. Um auch den dynamischen Charakter der Umgebung zu berücksichtigen, sind zusätzlich zur Rasterkarte der Umgebung virtuelle Hindernisse bei der Strahlverfolgung zu berücksichtigen. Die Vorgehensweise ist analog: Eine Anzahl von Hindernisprimitiven steht zur Verfügung, diese werden gemäß ihrer Trajektorien zu den jeweiligen Zeitschritten in die Rasterkarte eingezeichnet und bei der Strahlverfolgung berücksichtigt.

Durch die effiziente Umsetzung dieser Umgebungssimulation ist der Aufbau und die beliebig häufige Wiederholung eines Testszenarios möglich. Zusätzlich stehen präzise Informationen, beispielsweise über die dynamischen Hindernisse, zur Verfügung, die in der realen Versuchsumgebung nur bedingt erfasst werden können. Abgesehen von den hier aufgeführten virtuellen Modulen, arbeitet die Simulation mit in der realen Anwendung identischen Modulen und im Echtzeitkontext.

4.2 Roboterkinematik

Der Einsatzbereich der in der vorliegenden Arbeit entwickelten Methoden ist auf radgeführte Robotersysteme abgestimmt. Die Entwicklung und Erprobung erfolgt zwar im Hinblick auf den Einsatz im Innenraum, jedoch sind sie auf den Außenbereich übertragbar. Bei radgeführten Robotersystemen im Innenraum kommen verschiedene kinematische Konzepte in Frage. Bei der Methodenentwicklung in den folgenden Abschnitten wird darauf geachtet, keine restriktiven Einschränkungen bezüglich der kinematischen Struktur zu machen. Es wird bei der Bahnplanung berücksichtigt, dass die Methoden auch für Robotersysteme mit dynamischen und nichtholonomen Zwangsbedingungen geeignet sind, da in der Realität kaum Systeme existieren, bei denen diese fehlen, bzw. vernachlässigt werden dürfen. Das Referenzkoordinatensystem ist dabei gemäß dem Unicycle-Modell definiert. (siehe Bild 4.6). Hierbei wird der Roboter durch ein Einrad definiert, dessen Zustandsvek-

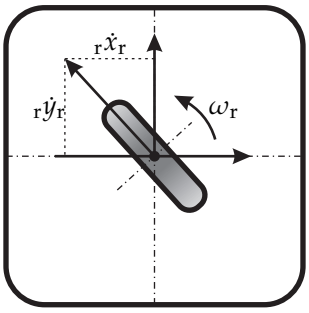


Bild 4.6: Bei der Unicycle-Kinematik wird der Roboter auf ein Einrad reduziert. Die Annahme dieser einheitlichen Schnittstelle bietet sich für die Entwicklung von Steuerungs- und Regelungskonzepten an, um die Methoden auf unterschiedliche Robotersysteme anwenden zu können.

tor aus $_{\mathrm{r}}\boldsymbol{x}_{\mathrm{r}} = [\ {}_{\mathrm{r}}\dot{x}_{\mathrm{r}} \quad {}_{\mathrm{r}}\dot{y}_{\mathrm{r}} \quad \omega_{\mathrm{r}}\ ]^{\mathrm{T}}$ besteht. Diese definierte Schnittstelle ermöglicht die Anwendung in standardisierten Systemumgebungen.

Bei dem hier eingesetzten Versuchsträger (Bild 4.1) kommt eine mobile Plattform mit redundantem Antrieb zum Einsatz. Die mobile Plattform ist mit vier identischen Radmodulen ausgestattet, die jeweils gelenkt und angetrieben sind [Hanebeck u. a., 1999]. Im Folgenden wird die Schätzung des Roboterzustands für den Versuchsträger dargestellt. Die kinematische Struktur des Versuchsträgers ist in Bild 4.7a dargestellt. Das inverse kinematische Modell liefert den Zusammenhang zwischen der Bewegung mit drei Freiheitsgraden im kartesischen Raum und den acht Aktorfreiheitsgraden. Der Zustand des Roboters ist mit Bezug zu einem plattformfesten Koordinatensystem $(KS)_{\mathrm{r}}$ im geometrischen Schwerpunkt der Plattform

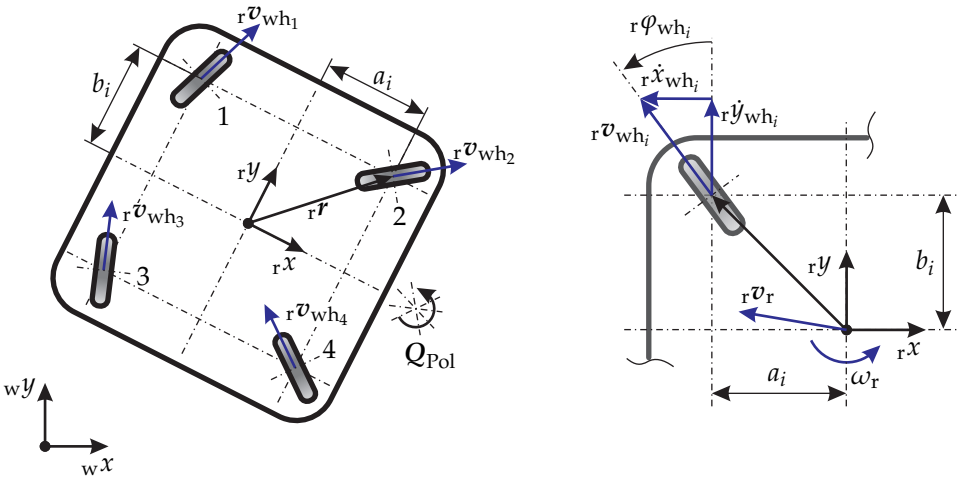


Bild 4.7: Der Versuchsträger wird mit vier gelenkten und angetriebenen Radmodulen betrieben. Dieses führt auf acht Aktorfreiheitsgrade bei drei Freiheitsgraden für die ebene Bewegung.

definiert (Bild 4.7a). Der Geschwindigkeitsvektor ${}_{\mathrm{r}}\boldsymbol{v}_{\mathrm{wh}_i}$ für jedes Rad wird aus dessen Position ${}_{\mathrm{r}}\boldsymbol{r}_{\mathrm{wh}_i}$ im Roboterkoordinatensystem gewonnen.

$$ {}_{\mathrm{r}}\boldsymbol{r}_{\mathrm{wh}_i} = \begin{bmatrix} a_i & b_i \end{bmatrix}^{\mathrm{T}} \tag{4.1} $$

$$ {}_{\mathrm{r}}\boldsymbol{v}_{\mathrm{wh}_i} = \begin{bmatrix} {}_{\mathrm{r}}\dot{x} - \omega_{\mathrm{r}} \cdot b_i \\ {}_{\mathrm{r}}\dot{y} + \omega_{\mathrm{r}} \cdot a_i \end{bmatrix} \tag{4.2} $$

Bei gegebener Plattformgeschwindigkeit können Geschwindigkeit und Lenkwinkel jedes Rades aus geometrischen Größen berechnet werden (Bild 4.7b). Exemplarisch wird die Gleichung für das erste Rad durch

$$ {}_{\mathrm{r}}\varphi_{\mathrm{wh}_1} = \arctan \frac{-{}_{\mathrm{r}}\dot{x} + \omega_{\mathrm{r}} \cdot b_1}{{}_{\mathrm{r}}\dot{y} + \omega_{\mathrm{r}} \cdot a_1} \quad \text{und} \tag{4.3} $$

$$ |{}_{\mathrm{r}}\boldsymbol{v}_{\mathrm{wh}_1}| = \sqrt{({}_{\mathrm{r}}\dot{x} - \omega_{\mathrm{r}} \cdot b_1)^2 + ({}_{\mathrm{r}}\dot{y} + \omega_{\mathrm{r}} \cdot a_1)^2} \tag{4.4} $$

gegeben, wobei ${}_{\mathrm{r}}\varphi_{\mathrm{wh}_1}$ mathematisch positiv von der y-Achse ausgehend berechnet wird. Um den Roboterzustand mittels Messungen aus Lenkwinkeln und Radgeschwindigkeiten zu bestimmen, muss die Redundanz der acht Aktorfreiheitsgrade bei drei Freiheitsgraden ${}_{\mathrm{r}}\boldsymbol{x}_{\mathrm{r}} = \begin{bmatrix} \dot{x} & \dot{y} & \omega \end{bmatrix}^{\mathrm{T}}$ für die Bewegung in der Ebene gelöst werden. Für die Zusammenführung der Daten wird ein KALMAN-Filter angewendet, welches ein zeitdiskretes kinematisches Modell im Zustandsraum erfordert. Bei Annahme eines konstant beschleunigten Roboters im Inertialkoordinatensystem wird

der diskrete Zustand durch

$$\begin{aligned} {}_{\mathrm{r}}\dot{x}_{k+1} &= {}_{\mathrm{r}}\dot{x}_k + T \cdot {}_{\mathrm{r}}\ddot{x}_k \\ {}_{\mathrm{r}}\dot{y}_{k+1} &= {}_{\mathrm{r}}\dot{y}_k + T \cdot {}_{\mathrm{r}}\ddot{y}_k \\ \omega_{k+1} &= \omega_k + T \cdot \dot{\omega}_k \end{aligned} \tag{4.5}$$

beschrieben. Zusammenfassen lässt sich das kinematische Modell mittels der folgenden linearen zeitdiskreten Zustandsgleichung mit der Systemmatrix $\boldsymbol{A}$, der Eingangsmatrix $\boldsymbol{B}$, den im Eingangsvektor zusammengefassten Beschleunigungen $\boldsymbol{u}$ und dem Prozessrauschen $\boldsymbol{\epsilon}_k$

$$\boldsymbol{x}_{k+1} = \boldsymbol{A} \cdot \boldsymbol{x}_k + \boldsymbol{B} \cdot \boldsymbol{u}_k + \boldsymbol{\epsilon}_k \tag{4.6}$$

mit

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{und} \qquad \boldsymbol{B} = \begin{bmatrix} T & 0 & 0 \\ 0 & T & 0 \\ 0 & 0 & T \end{bmatrix} \quad .$$

Für die Messung gilt

$$\boldsymbol{z}_k = \begin{bmatrix} {}_{\mathrm{r}}\varphi_{\mathrm{wh}_1} & {}_{\mathrm{r}}\varphi_{\mathrm{wh}_2} & {}_{\mathrm{r}}\varphi_{\mathrm{wh}_3} & {}_{\mathrm{r}}\varphi_{\mathrm{wh}_4} & |{}_{\mathrm{r}}\boldsymbol{v}_{\mathrm{wh}_1}| & |{}_{\mathrm{r}}\boldsymbol{v}_{\mathrm{wh}_2}| & |{}_{\mathrm{r}}\boldsymbol{v}_{\mathrm{wh}_3}| & |{}_{\mathrm{r}}\boldsymbol{v}_{\mathrm{wh}_4}| \end{bmatrix}^{\mathrm{T}}. \tag{4.7}$$

Die nichtlineare Ausgangsgleichung lässt sich aus der inversen Kinematik unter Berücksichtigung des Messrauschens δ_k gewinnen

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{\delta}_k \quad , \tag{4.8}$$

mit

$$\boldsymbol{h}(\boldsymbol{x}_k) = \begin{bmatrix} \arctan\left(\frac{-{}_{\mathrm{r}}\dot{x}+\omega\cdot b}{{}_{\mathrm{r}}\dot{y}+\omega\cdot a}\right) \\ \arctan\left(\frac{-{}_{\mathrm{r}}\dot{x}+\omega\cdot b}{{}_{\mathrm{r}}\dot{y}-\omega\cdot a}\right) \\ \arctan\left(\frac{-{}_{\mathrm{r}}\dot{x}-\omega\cdot b}{{}_{\mathrm{r}}\dot{y}-\omega\cdot a}\right) \\ \arctan\left(\frac{-{}_{\mathrm{r}}\dot{x}-\omega\cdot b}{{}_{\mathrm{r}}\dot{y}+\omega\cdot a}\right) \\ \sqrt{({}_{\mathrm{r}}\dot{x}-\omega\cdot b)^2+({}_{\mathrm{r}}\dot{y}+\omega\cdot a)^2} \\ \sqrt{({}_{\mathrm{r}}\dot{x}-\omega\cdot b)^2+({}_{\mathrm{r}}\dot{y}-\omega\cdot a)^2} \\ \sqrt{({}_{\mathrm{r}}\dot{x}+\omega\cdot b)^2+({}_{\mathrm{r}}\dot{y}-\omega\cdot a)^2} \\ \sqrt{({}_{\mathrm{r}}\dot{x}+\omega\cdot b)^2+({}_{\mathrm{r}}\dot{y}+\omega\cdot a)^2} \end{bmatrix} \quad . \tag{4.9}$$

Die Anwendung des KALMAN-Filters setzt die Bekanntheit der folgenden Größen voraus [Kalman, 1960]. $\boldsymbol{R}$ bezeichnet die Kovarianz des Prozessrauschens, während $\boldsymbol{Q}$ die Kovarianz des Messrauschens beinhaltet. Anfangswerte für den Zustand $\boldsymbol{x}_0$ und für die Kovarianz des Prädiktionsfehlers $\boldsymbol{\Sigma}_0$ werden vorausgesetzt. Die Prädiktion für die Geschwindigkeit wird auf Basis des Zustands im vorherigen Abtastschritt $\tilde{\boldsymbol{x}}_{k-1}$ mit der Systemmatrix $\boldsymbol{A}$ und der über $\boldsymbol{B}$ wirkenden Eingangsgröße $\boldsymbol{u}_{k-1}$ berechnet,

$$\tilde{\boldsymbol{x}}_k^- = \boldsymbol{A} \cdot \tilde{\boldsymbol{x}}_{k-1} + \boldsymbol{B} \cdot \boldsymbol{u}_{k-1} \quad . \tag{4.10}$$

Die Prädiktion der Fehlerkovarianz ergibt sich zusätzlich aus der Fehlerkovarianz im vorherigen Zeitschritt Σ_{k-1} unter Berücksichtigung der Prozessunsicherheit $\boldsymbol{R}$ zu

$$\boldsymbol{\Sigma}_k^- = \boldsymbol{A} \cdot \boldsymbol{\Sigma}_{k-1} \cdot \boldsymbol{A}^{\mathrm{T}} + \boldsymbol{R} \quad . \tag{4.11}$$

Bei der Variante des Erweiterten KALMAN-Filters wird das Messmodell mittels TAYLOR-Reihe linearisiert, wobei die Zustandsprädiktion $\tilde{\boldsymbol{x}}^-$ als Arbeitspunkt eingesetzt wird

$$\boldsymbol{H}_k = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}_k}(\tilde{\boldsymbol{x}}_k^-) \quad . \tag{4.12}$$

Dieses führt auf die im Folgenden exemplarisch für das erste Radmodul dargestellten Elemente der $\mathbb{R}^{8\times 3}$ Messmatrix für den Lenkwinkel

$$\varphi_1 \approx p_{11}{}_{\mathrm{r}}\tilde{x} + p_{12}{}_{\mathrm{r}}\tilde{y} + p_{13}\tilde{\omega} \quad \text{mit} \tag{4.13}$$

$$p_{11} = \frac{-\left({}_{\mathrm{r}}\tilde{y} + a\tilde{\omega}\right)}{{}_{\mathrm{r}}\tilde{x}^2 + {}_{\mathrm{r}}\tilde{y}^2 + 2\tilde{\omega}\left({}_{\mathrm{r}}\tilde{y}a - {}_{\mathrm{r}}\tilde{x}b\right) + \tilde{\omega}^2\left(a^2+b^2\right)}$$

$$p_{12} = \frac{\left({}_{\mathrm{r}}\tilde{x} - b\tilde{\omega}\right)}{{}_{\mathrm{r}}\tilde{x}^2 + {}_{\mathrm{r}}\tilde{y}^2 + 2\tilde{\omega}\left({}_{\mathrm{r}}\tilde{y}a - {}_{\mathrm{r}}\tilde{x}b\right) + \tilde{\omega}^2\left(a^2+b^2\right)}$$

$$p_{13} = \frac{\left(a_{\mathrm{r}}\tilde{x} + bv_y\right)\tilde{\omega}}{{}_{\mathrm{r}}\tilde{x}^2 + {}_{\mathrm{r}}\tilde{y}^2 + 2\tilde{\omega}\left({}_{\mathrm{r}}\tilde{y}a - {}_{\mathrm{r}}\tilde{x}b\right) + \tilde{\omega}^2\left(a^2+b^2\right)}$$

und die Radgeschwindigkeit

$$v_1 \approx q_{51}{}_{\mathrm{r}}\tilde{x} + q_{52}{}_{\mathrm{r}}\tilde{y} + q_{53}\tilde{\omega} \quad \text{mit} \tag{4.14}$$

$$q_{51} = \frac{\left({}_{\mathrm{r}}\tilde{x} - b\tilde{\omega}\right)}{\sqrt{{}_{\mathrm{r}}\tilde{x}^2 + {}_{\mathrm{r}}\tilde{y}^2 + 2\tilde{\omega}\left({}_{\mathrm{r}}\tilde{y}a - {}_{\mathrm{r}}\tilde{x}b\right) + \tilde{\omega}^2\left(a^2+b^2\right)}}$$

$$q_{52} = \frac{\left({}_{\mathrm{r}}\tilde{y} + a\tilde{\omega}\right)}{\sqrt{{}_{\mathrm{r}}\tilde{x}^2 + {}_{\mathrm{r}}\tilde{y}^2 + 2\tilde{\omega}\left({}_{\mathrm{r}}\tilde{y}a - {}_{\mathrm{r}}\tilde{x}b\right) + \tilde{\omega}^2\left(a^2+b^2\right)}}$$

$$q_{53} = \frac{\left(a_{\mathrm{r}}\tilde{y} - b_{\mathrm{r}}\tilde{x} + \tilde{\omega}\left(a^2+b^2\right)\right)}{\sqrt{{}_{\mathrm{r}}\tilde{x}^2 + {}_{\mathrm{r}}\tilde{y}^2 + 2\tilde{\omega}\left({}_{\mathrm{r}}\tilde{y}a - {}_{\mathrm{r}}\tilde{x}b\right) + \tilde{\omega}^2\left(a^2+b^2\right)}} \quad .$$

Die vollständige Messmatrix wird durch

$$\boldsymbol{H}_{k,ij} = \begin{cases} p_{ij} \text{ für } 1 \leq i \leq 4\ ,\ 1 \leq j \leq 3 \\ q_{ij} \text{ für } 5 \leq i \leq 8\ ,\ 1 \leq j \leq 3 \end{cases} \tag{4.15}$$

ausgedrückt. Die Verstärkung des KALMAN-Filters ergibt sich aus

$$\boldsymbol{K}_k = \boldsymbol{\Sigma}_k^- \cdot \boldsymbol{H}_k^{\mathrm{T}} \cdot \left(\boldsymbol{H}_k \cdot \boldsymbol{\Sigma}_k^- \cdot \boldsymbol{H}_k^{\mathrm{T}} + \boldsymbol{Q} \right)^{-1} \quad . \tag{4.16}$$

Im Korrekturschritt, in dem die aktuelle Messung eingerechnet wird, erfolgt die Aktualisierung des Zustands mittels

$$\tilde{\boldsymbol{x}}_k = \tilde{\boldsymbol{x}}_k^- + \boldsymbol{K}_k \cdot (\boldsymbol{z}_k - \boldsymbol{h}(\tilde{\boldsymbol{x}}_k^-)) \quad . \tag{4.17}$$

Der resultierende Zustand wird nun genutzt um die Fehlerkovarianzmatrix zu aktualisieren

$$\boldsymbol{\Sigma}_k = (\boldsymbol{I} - \boldsymbol{K}_k \cdot \boldsymbol{H}_k) \cdot \boldsymbol{\Sigma}_k^- \quad . \tag{4.18}$$

Durch diese Zustandsschätzung steht überlagerten Regelkreisen stets der aktuelle Zustand des mobilen Roboters zur Verfügung.

4.3 Bahnregelung

Im Folgenden wird die Bahnregelung für den mobilen Roboter beschrieben. Angestrebt wird ein einfaches und robustes Regelungsgesetz, welches auf die Unicycle-Kinematik Anwendung findet. Während der Ansatz der Verfolgung einer zeitparametrisierten Trajektorie weit verbreitet ist, wird an dieser Stelle lediglich die geometrische Bahn verfolgt, wobei die zeitliche Parametrisierung höheren Regelkreisen überlassen wird. Die Bahn wird als eine Folge von Kurven und geraden Strecken dargestellt. Um numerische Probleme mit großen Werten im kartesischen Koordinatensystem zu vermeiden, erfolgt die Regelung in den lokalen Koordinatensystemen der Geraden und Kurvenelemente (Fig. 4.8). Ein ähnlicher Ansatz wird in [Abdellatif u. a., 2002, 2003; Horn, 1997] verfolgt.

In einem überlagerten System wird die Position des Roboters im Inertialkoordinatensystem bestimmt. Mittels dieser Information kann das aktuelle Bahnelement identifiziert werden. Die Roboterposition wird in das Koordinatensystem b_i des lokalen Bahnelements i transformiert. Die Position des Roboters im lokalen (Polar-) Koordinatensystem wird durch den Abstand des Roboters zum Ursprung ${}_{\mathrm{b}}r_{\mathrm{r}}$ und

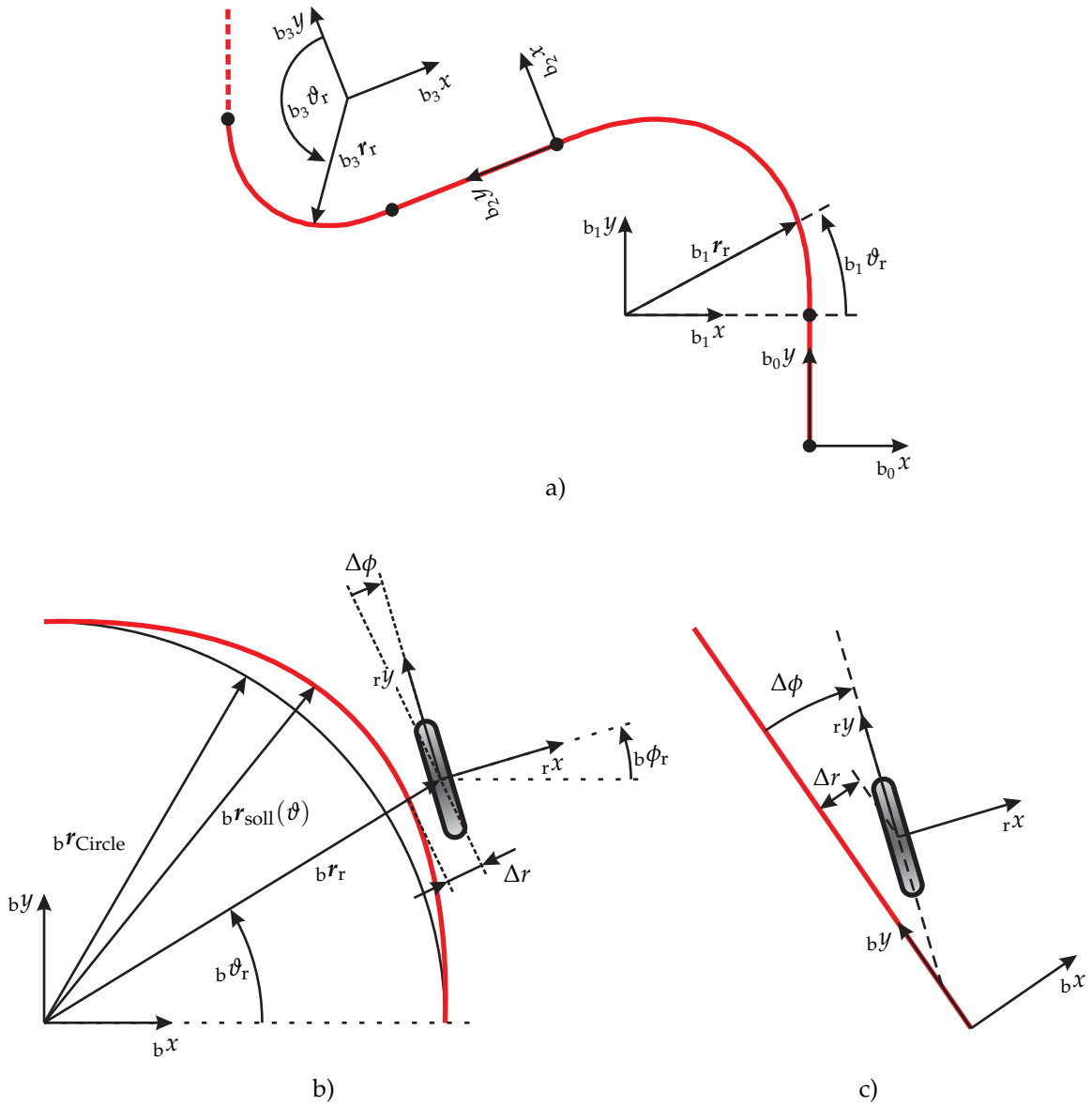


Bild 4.8: Die Bahn für den mobilen Roboter wird aus Geraden und Kurvenelementen zusammengesetzt. Die Regelung erfolgt in den lokalen Koordinatensystemen der Bahnelemente, siehe a). In b) ist die Beschreibung des Roboters in einem Kurvenelement dargestellt, hierbei kommen Polarkoordinaten zum Einsatz. In c) erfolgt analog die Beschreibung für ein Geradenelement in kartesischen Koordinaten. Die Abweichung des Roboters von der Bahn wird jeweils durch die Fehlorientierung $\Delta\phi$ und den Querversatz Δr ausgedrückt.

den Winkel ${}_{\mathrm{b}}\vartheta_{\mathrm{r}}$, gemessen von der x-Achse, ausgedrückt. Die kartesische Position des Roboters und die Ableitungen werden durch Gl. (4.19) bestimmt.

$$\begin{aligned}
{}_{\mathrm{b}}x_{\mathrm{r}} &= {}_{\mathrm{b}}r_{\mathrm{r}} \cdot \cos({}_{\mathrm{b}}\vartheta_{\mathrm{r}}) \\
{}_{\mathrm{b}}y_{\mathrm{r}} &= {}_{\mathrm{b}}r_{\mathrm{r}} \cdot \sin({}_{\mathrm{b}}\vartheta_{\mathrm{r}}) \\
{}_{\mathrm{b}}\dot{x}_{\mathrm{r}} &= -\sin({}_{\mathrm{b}}\phi_{\mathrm{r}}) \cdot v_{\mathrm{r}} \\
{}_{\mathrm{b}}\dot{y}_{\mathrm{r}} &= \cos({}_{\mathrm{b}}\phi_{\mathrm{r}}) \cdot v_{\mathrm{r}} \\
{}_{\mathrm{b}}r_{\mathrm{r}} &= \sqrt{{}_{\mathrm{b}}x_{\mathrm{r}}^2 + {}_{\mathrm{b}}y_{\mathrm{r}}^2} \\
{}_{\mathrm{b}}\vartheta_{\mathrm{r}} &= \arctan \frac{{}_{\mathrm{b}}y_{\mathrm{r}}}{{}_{\mathrm{b}}x_{\mathrm{r}}}
\end{aligned} \tag{4.19}$$

Diese dienen dann zur Berechnung der Ableitungen im Polarkoordinatensystem durch Gln. (4.20 – 4.23).

$${}_{\mathrm{b}}\dot{r}_{\mathrm{r}} = \frac{{}_{\mathrm{b}}x_{\mathrm{r}} \cdot {}_{\mathrm{b}}\dot{x}_{\mathrm{r}} + {}_{\mathrm{b}}y_{\mathrm{r}} \cdot {}_{\mathrm{b}}\dot{y}_{\mathrm{r}}}{\sqrt{{}_{\mathrm{b}}x_{\mathrm{r}}^2 + {}_{\mathrm{b}}y_{\mathrm{r}}^2}} \tag{4.20}$$

$$= v_{\mathrm{r}} \cdot \sin({}_{\mathrm{b}}\vartheta_{\mathrm{r}} - {}_{\mathrm{b}}\phi_{\mathrm{r}}) \tag{4.21}$$

$${}_{\mathrm{b}}\dot{\vartheta}_{\mathrm{r}} = \frac{{}_{\mathrm{b}}x_{\mathrm{r}} \cdot {}_{\mathrm{b}}\dot{y}_{\mathrm{r}} - {}_{\mathrm{b}}y_{\mathrm{r}} \cdot {}_{\mathrm{b}}\dot{x}_{\mathrm{r}}}{{}_{\mathrm{b}}x_{\mathrm{r}}^2 + {}_{\mathrm{b}}y_{\mathrm{r}}^2} \tag{4.22}$$

$$= \frac{v_{\mathrm{r}}}{{}_{\mathrm{b}}r_{\mathrm{r}}} \cdot \cos({}_{\mathrm{b}}\vartheta_{\mathrm{r}} - {}_{\mathrm{b}}\phi_{\mathrm{r}}) \tag{4.23}$$

Für den Differenzwinkel zwischen der Roboterposition im lokalen Koordinatensystem ${}_{\mathrm{b}}\vartheta_{\mathrm{r}}$ und der Roboterorientierung ${}_{\mathrm{b}}\phi_{\mathrm{r}}$ wird die Variable τ eingeführt,

$$\tau = {}_{\mathrm{b}}\vartheta_{\mathrm{r}} - {}_{\mathrm{b}}\phi_{\mathrm{r}} \quad , \tag{4.24}$$

womit nach Hinzufügen der Regelgröße $\dot{\phi}_{\mathrm{r,soll}}$ die folgende Zustandsgleichung entsteht:

$$\begin{bmatrix} {}_{\mathrm{b}}\dot{r}_{\mathrm{r}} \\ {}_{\mathrm{b}}\dot{\vartheta}_{\mathrm{r}} \end{bmatrix} = \begin{bmatrix} v_{\mathrm{r}} \cdot \sin(\tau) \\ \frac{v_{\mathrm{r}}}{{}_{\mathrm{b}}r_{\mathrm{r}}} \cdot \cos(\tau) \end{bmatrix} + \begin{bmatrix} 0 \\ \dot{\phi}_{\mathrm{r,soll}} \end{bmatrix} \quad . \tag{4.25}$$

In der Regelung kommt die *Dynamische Feedback-Linearisierung* zum Einsatz. Hierbei wird eine Linearisierung des nichtlinearen Systems durch die positive Rückführung des Systemausgangs über das inverse dynamische Modell erreicht. Der geschlossene Regelkreis mit der nichtlinearen Kompensation ist durch eine Transformation äquivalent zu einem linearen, regelbaren System. Die nichtlineare Kompensation wird bestimmt durch die Differentiation des Ausgangs bis der Eingang in nicht singulärer Form auftaucht [Luca u. a., 2000].

Im hier vorliegenden Fall führt die Ableitung von ${}_{\mathrm{b}}\dot{r}_{\mathrm{r}}$ auf

$$ {}_{\mathrm{b}}\ddot{r}_{\mathrm{r}} = v_{\mathrm{r}} \cdot \cos(\tau) \cdot \dot{\tau} + \dot{v}_{\mathrm{r}} \cdot \sin\tau \quad , \tag{4.26} $$

welche die Regelgröße in nicht singulärer Form innerhalb von $\dot{\tau}$ enthält. Die Annahme konstanter Geschwindigkeit vereinfacht Gl. (4.26) zu

$$ {}_{\mathrm{b}}\ddot{r}_{\mathrm{r}} = v_{\mathrm{r}} \cdot \cos(\tau) \cdot \dot{\tau} \quad . \tag{4.27} $$

Die Differentiation des Winkels τ und das Ersetzen der Zustandsgröße ${}_{\mathrm{b}}\dot{\vartheta}_{\mathrm{r}}$ ergibt

$$ \dot{\tau} = {}_{\mathrm{b}}\dot{\vartheta}_{\mathrm{r}} - {}_{\mathrm{b}}\dot{\phi}_{\mathrm{r}} \tag{4.28} $$

$$ = \frac{v_{\mathrm{r}}}{{}_{\mathrm{b}}r_{\mathrm{r}}} \cdot \cos(\tau) + \dot{\phi}_{\mathrm{r,soll}} - {}_{\mathrm{b}}\dot{\phi}_{\mathrm{r}} \quad , \tag{4.29} $$

und wird nach $\dot{\phi}_{\mathrm{r,soll}}$ aufgelöst,

$$ \dot{\phi}_{\mathrm{r,soll}} = {}_{\mathrm{b}}\dot{\phi}_{\mathrm{r}} - \frac{v_{\mathrm{r}}}{{}_{\mathrm{b}}r_{\mathrm{r}}} \cdot \cos(\tau) + \frac{{}_{\mathrm{b}}\ddot{r}_{\mathrm{r}}}{v_{\mathrm{r}} \cdot \cos(\tau)} \quad . \tag{4.30} $$

was auf ein Kompensationsglied in der Form

$$ \dot{\phi}_{\mathrm{r,soll}} = {}_{\mathrm{b}}\dot{\phi}_{\mathrm{r}} - \frac{v_{\mathrm{r}}}{{}_{\mathrm{b}}r_{\mathrm{r}}} \cdot \cos(\tau) + \frac{u}{v_{\mathrm{r}} \cdot \cos(\tau)} \tag{4.31} $$

führt, mit

$$ \Delta r = r({}_{\mathrm{b}}\vartheta_{\mathrm{r}}) - {}_{\mathrm{b}}r_{\mathrm{r}} \tag{4.32} $$

$$ u = \ddot{r}_{\mathrm{soll}} + K_d \cdot \Delta\dot{r} + K_p \cdot \Delta r \tag{4.33} $$

und der durch folgende Gleichung bestimmte Fehlerdynamik

$$ 0 = \Delta\ddot{r} + K_d \cdot \Delta\dot{r} + K_p \cdot \Delta r \quad . \tag{4.34} $$

Für die Fahrt auf gerader Strecke vereinfacht sich das Kompensationsglied auf

$$ \dot{\phi}_{\mathrm{r,soll}} = \frac{u}{v_{\mathrm{r}} \cdot \cos(\tau)} \quad . \tag{4.35} $$

Das beschriebene Bahnregelungskonzept kann auf Bahnen angewendet werden, die sich effizient in Polarkoordinaten beschreiben lassen. In dieser Arbeit werden die Kurven werden durch Polar-Splines repräsentiert, diese haben durch den kontinuierlichen Verlauf der Krümmung einen Vorteil gegenüber Kreissegmenten, so

dass der Lenkwinkel bei der Fahrt von einem Segment in ein anderes Segment kontinuierlich verläuft. Die Berechnung des idealen Radius in Abhängigkeit von ϑ und dem Öffnungswinkel des Splines γ erfolgt durch

$$r(\vartheta) = r_{Circle} \cdot \left(1 + \frac{\vartheta^2}{2} - \frac{\vartheta^3}{\gamma} + \frac{\vartheta^4}{2\gamma^2}\right) \quad , \tag{4.36}$$
$$\frac{\partial r(\vartheta)}{\partial \vartheta} = r_{Circle} \cdot \left(\vartheta - 3 \cdot \frac{\vartheta^2}{\gamma} + 2 \cdot \frac{\vartheta^3}{\gamma^2}\right) \quad .$$

Damit steht der Sollradius $r({}_{\mathrm{b}}\vartheta_{\mathrm{r}})$ analytisch zur Bildung von Δr zur Verfügung. Aus Gl. (4.21) geht hervor, dass die zeitliche Ableitung des Querversatzes lediglich vom Differenzwinkel τ und der Geschwindigkeit v_{r} abhängen. Der Differenzwinkel τ_{soll} ergibt für den Polar-Spline zu

$$\tau_{\mathrm{soll}} = \arctan\left(\frac{\partial_{\mathrm{b}} r({}_{\mathrm{b}}\vartheta_{\mathrm{r}})}{\partial_{\mathrm{b}}\vartheta_{\mathrm{r}}} \frac{1}{{}_{\mathrm{b}}r({}_{\mathrm{b}}\vartheta_{\mathrm{r}})}\right) \quad , \tag{4.37}$$

so dass mit dem gemessenen v_{r} auch die zeitliche Ableitung des Querversatzes $\Delta\dot{r}$ auf analytischem Weg bestimmt und eine numerische Differentiation vermieden werden kann.

Bei einer Taktzeit von 50 ms ergeben sich eine anhand von Bild 4.9a einzuschätzende Regelgüte. Die reine Steuerung führt erwartungsgemäß insbesondere durch die Fehlorientierung auf große Abweichungen (Bild 4.9a), während die Anwendung der oben beschriebenen Regelung nur auf kleine Bahnabweichungen in der Größenordnung $\leq 20\,mm$ führt (Bild 4.9b).

Durch die Repräsentation der Bahn mittels Kurven und geraden Strecken beschrieben aus Polar-Splines ergibt sich ein kontinuierlicher Verlauf des Lenkwinkels. Da die Lenkwinkelgeschwindigkeit jedoch nicht kontinuierlich verläuft wird die Winkelbeschleunigung durch eine Schranke begrenzt. Dieser nicht modellierte Effekt führt zu den in Bild 4.9b erkennbaren Überhöhungen bei der seitlichen Abweichung von der Bahn. Da die Nutzung von Polar-Splines eine Reihe von Vorteilen bietet und die Bahnabweichungen trotz dieser Effekte vertretbar sind, wird an dieser Form der Bewegungsprimitive festgehalten.

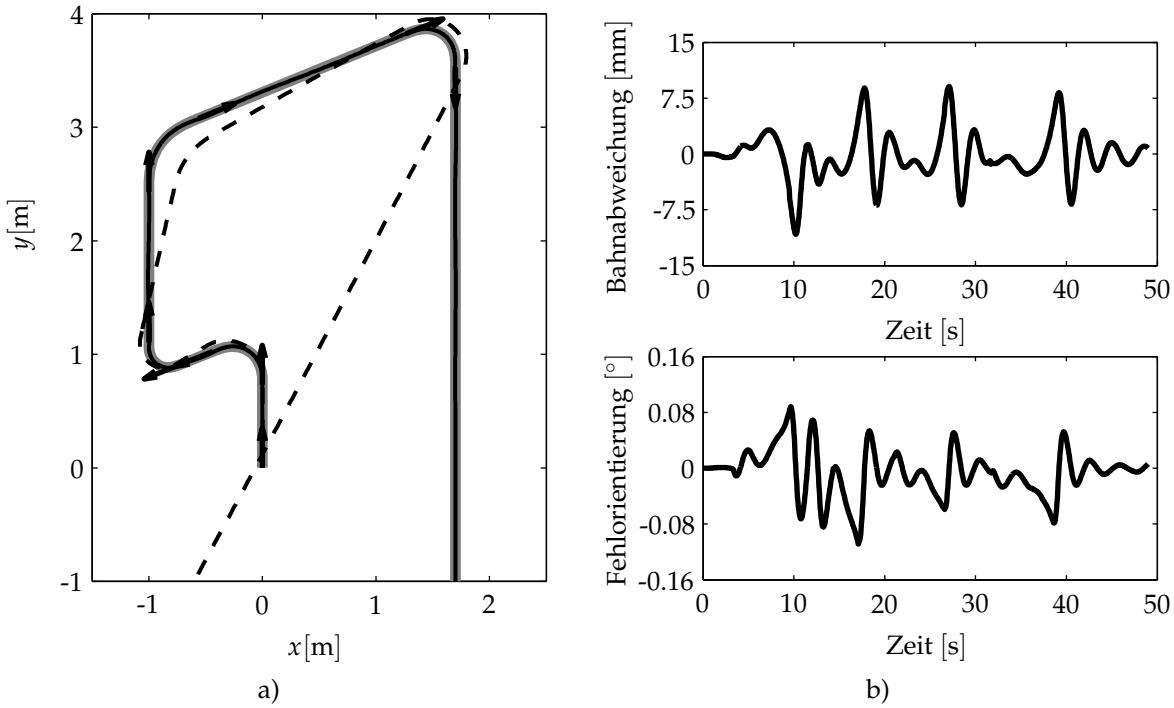


Bild 4.9: Dargestellt sind die Ergebnisse bei der Bahnregelung mittels *Feedback-Linearisierung*. In a) ist die Sollbahn durch die die graue Linie dargestellt, die in Richtung der Pfeile abgefahren werden soll. Die gestrichelte Linie zeigt die resultierende Bahn bei reiner Steuerung. Die durchgezogene Linie stellt den Verlauf mit aktiver Regelung dar. Sie verläuft nahezu exakt über der Sollbahn. In b) ist die Bahnabweichung und die Fehlorientierung für den geregelten Betrieb dargestellt.

5 Wahrnehmung der Umgebung

Während die Steuerung und Regelung des Bewegungszustands autonomer Fahrzeuge bereits gut beherrscht werden, existiert noch Forschungsbedarf bezüglich der Modellierung der Umgebung, Wahrnehmung der Umgebung und der Schätzung des Umgebungszustands. Neben Erfassung und Kontrolle des Eigenzustands ist die Wahrnehmung der zweite wesentliche Aspekt eines autonomen Fahrzeugs. Die Erfassung des Umgebungszustands ist ein ungleich größeres Problem als die Erfassung des Eigenzustands, da die Umgebung sehr vielfältig ist, die Zustandsräume vielfach unbeschränkt oder wenigstens groß sind und die Bedingungen für die Messung nur bedingt beeinflusst werden können.

Im diesem Kapitel wird als Umgebungsmodell die 2-dimensionale Rasterkarte vorgestellt, die sich gegenüber anderen Umgebungsrepräsentationen insbesondere gegenüber 3-dimensionalen Karten durch eine effiziente Handhabung und gegenüber vektorbasierten Karten durch die Unabhängigkeit von strukturellen Merkmalen auszeichnet. Neben der Art der Datenrepräsentation gehört dazu die Kartenerstellung aus Messdaten. Grundlage für diese Aufgabe sind verschiedene Sensoren, die im Folgenden mit ihren Eigenschaften kurz vorgestellt werden sollen. Hier wird vornehmlich auf die Sensoren und Sensorprinzipien eingegangen, die in der vorliegenden Arbeit zum Einsatz kommen. Eine zentrale Rolle wird dabei von Laserentfernungsmesssystemen eingenommen, neben diesen wird die Sensorik zur Erfassung des Eigenzustands beispielsweise in der Odometrie eingesetzt. Die Erläuterung zu alternativen Sensoren soll lediglich die Auswahl der Systeme begründen und einen Überblick verschaffen.

5.1 Umgebungsmodelle und Karten

Für die Wahl eines Modells zur Repräsentation der Umgebung gibt es drei wesentliche Kriterien. Zunächst steht die Genauigkeit, mit der ein autonomes System agieren kann in direktem Zusammenhang mit der Genauigkeit des Umgebungsmodells. Weiterhin muss die Art und Genauigkeit der Sensorsysteme mit der Repräsentation und Auflösung der Karte korrespondieren. Schließlich wird der Bedarf an Ressourcen bezüglich Erstellung, Bearbeitung und Benutzung von Kartenmaterial für die Lokalisation, Bahnplanung und Bahnverfolgung wesentlich durch die Komplexität der Karte geprägt. Kennzeichnend für die Umgebung der in der vor-

liegenden Arbeit behandelten autonomen Fahrzeuge ist eine Beschränkung auf die Ebene. Auch die Umgebungsmodelle zur Beschreibung der realen Welt lassen sich somit auf zwei Dimensionen reduzieren.

Für die Repräsentation der Umgebung haben sich drei Strukturen etabliert:

- Merkmalsbasierte Karten
- Topologische Karten (Roadmaps)
- Rasterkarten

Die Basis merkmalsbasierter Karten ist ein freier Raum, in dem Hindernisse durch geometrische Primitive (Merkmale, engl. Features) dargestellt werden. Die Primitive werden über Referenzpunkte in der Karte und deren geometrischen Eigenschaften beschrieben. Es handelt sich meist um Polygone. Merkmalsbasierte Karten lassen sich am ehesten mit Vektorgrafiken vergleichen. Durch die kontinuierliche Beschreibung sind beliebige Genauigkeiten möglich und die Speicherung ist sehr effizient, da die benötigten Ressourcen von der Dichte der Objekte im Raum abhängen. Ein Beispiel für eine merkmalsbasierte Karte ist in Bild 5.1a dargestellt. Bei der Darstellung einer realen Umgebung durch geometrische Primitive kann die Zahl der Objekte sehr groß werden. Hier bietet sich die Möglichkeit zur Abstraktion: je nach Einsatzzweck der Karte kann eine Beschränkung auf die Merkmale erfolgen, die von den Sensoren erfasst werden können. Besonders in Simulationen erfreuen sich merkmalsbasierte Karten großer Beliebtheit. Die Erstellung von Umgebungskarten aus Sensorinformationen gestaltet sich jedoch aufwändig, da die geometrischen Merkmale aus Sensordaten zunächst extrahiert werden müssen. In der Praxis eingesetzte, schnelle und robuste Extraktionsmethoden beschränken sich auf Verfahren zur Linienextraktion, die den Einsatz dieser Methode in einer gut strukturierten „rechtwinkligen“ Umgebung begünstigt.

Während sich die merkmalsbasierten Karten streng an die geometrische Beschreibung der Umgebung halten, zeichnen sich topologische Karten durch eine höhere Abstraktion aus. Topologische Karten verzichten auf die direkte Messung geometrischer Merkmale, sondern konzentrieren sich auf Merkmale, die der Lokalisation dienen. Die Darstellung erfolgt als Graph, in dem die Knoten Orte repräsentieren, wie z. B. Flure, Zimmer, Wegkreuzungen etc. und die Kanten eine direkte Verbindung zwischen zwei Knoten anzeigen. Die durch die Knoten repräsentierten Orte sind dabei nicht örtlich konzentriert, sondern umfassen einen Bereich, dessen Betreten bzw. Verlassen durch den Roboter als Aufenthaltsort erkannt werden kann. Ein geometrischer Aufenthaltsort im Umfeld des Knotens lässt sich nicht exakt ermitteln. Die Kanten beschreiben, wie der Weg von einem Knoten zu einem benach-

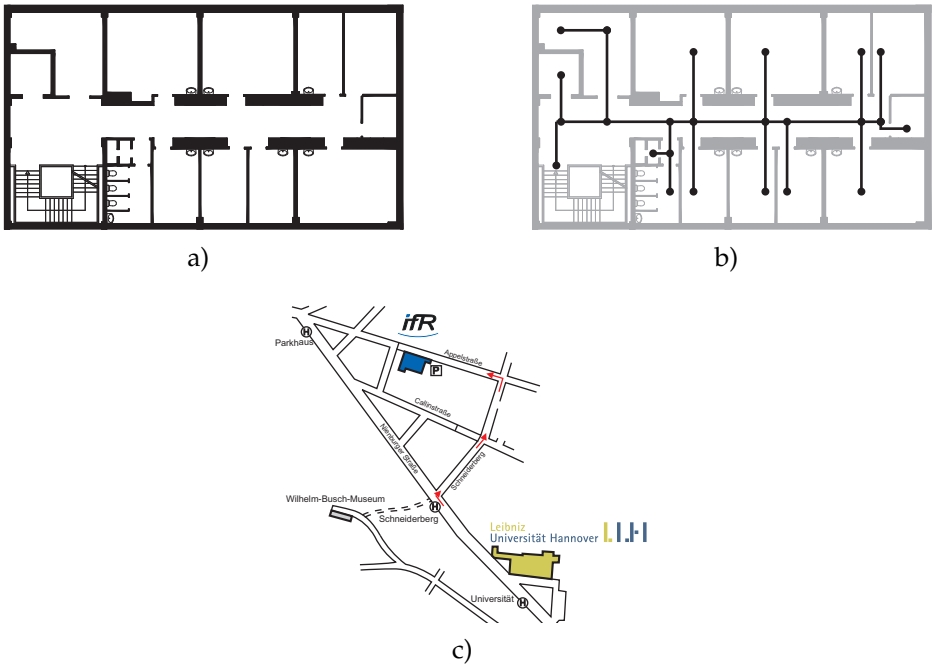


Bild 5.1: Verschiedene Kartendarstellungen: a) Grundriss eines Gebäudes, der, ähnlich einer Vektorgrafik, auf merkmalsbasierten Daten basiert; b) Beispiel für eine topologische Verknüpfung von Informationen durch Knoten und Kanten. c) zeigt eine exemplarische Straßenkarte.

barten Knoten zurückgelegt werden kann. Den Kanten kann dabei ein Gewicht zugeordnet werden, welches bei der Bahnplanung die Kosten unterschiedlicher Wege zum Ziel ins Verhältnis setzt. Besonders vorteilhaft bei der topologischen Karte ist die Möglichkeit zur Ausnutzung nicht geometrischer Informationen.

Eine Kombination von merkmalsbasierten Karten und topologischen Karten stellen Straßenkarten (engl. Roadmaps) dar. Diese verknüpfen die geometrischen Informationen der Wege mit den topologischen Informationen des Wegnetzes: Kreuzungen, Verzweigungen und Orte können als Knoten angesehen werden, während die Wege selbst als Kanten des Graphen interpretiert werden. Der Roboter nutzt das Wegnetz ähnlich wie der Mensch das Straßensystem benutzt: Am Ausgangsort muss zunächst eine Verbindung zum Netz gefunden werden, das Netz bringt den Roboter in die Nähe des Zielortes, dort verlässt der Roboter das Netz, um zum eigentlichen Ziel zu gelangen.

Neben den genannten Kartenarten existieren die Rasterkarten. Diese unterteilen den abzubildenden Raum mittels eines Rasters in einzelne Zellen. Über die Kantenlänge der einzelnen Zelle wird der Detaillierungsgrad der Karte eingestellt. Für

jede einzelne Zelle wird je nach Art der Karte eine Information gespeichert: Die Belegtheitskarten stellen mit den Zuständen *belegt* oder *frei* eine binäre Information zur Verfügung. Die Wahrscheinlichkeitskarten geben für jede einzelne Zelle die Wahrscheinlichkeit an, dass diese Zelle belegt ist. Mittels zusätzlicher Konventionen können weitere Informationen, beispielsweise über den Abstand zu Hindernissen oder eine erhöhte Kollisionsgefahr mit dynamischen Hindernissen kodiert werden. Vorteilhaft an der Rasterkarte ist die einfache informationstechnische Handhabung: Die Karte wird in einem Feld abgespeichert. Die Manipulation von Feldern mitunter beachtlicher Größen wird von nahezu allen Programmiersprachen gut beherrscht, die Daten lassen sich mittels Grafiken leicht veranschaulichen. Für die Bearbeitung der Informationen können teils effiziente Verfahren aus der Bildverarbeitung angewendet werden beispielsweise Filter zur Rauschunterdrückung oder Kantenschärfung. Generell gilt eine gute Eignung für die Anwendung numerischer Verfahren. Nachteilig ist der von der Komplexität der abzubildenden Umgebung unabhängige und insbesondere mit der Auflösung ansteigende Speicherbedarf, der für die Arbeit in unbegrenzten Umgebungen ein Arbeiten mit lokalen Karten in Kombination mit einem Kartenmanagement erforderlich macht. Weiterhin entstehen durch die Rasterung Diskretisierungsfehler (siehe auch Bild 5.2), die jedoch in realen Implementierungen der anderen Kartenarten ebenfalls unvermeidlich sind.

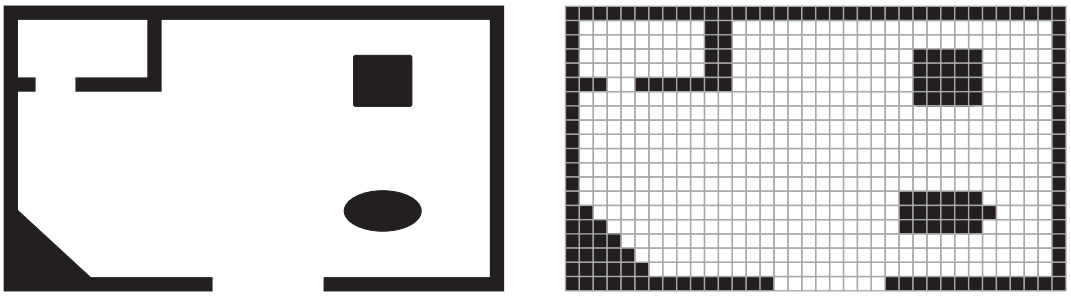


Bild 5.2: Bei der Rasterung einer realen Umgebung entstehen Diskretisierungsfehler: a) zeigt die reale Umgebung; b) zeigt den Verlust von Detailinformationen bei Rasterung.

Trotz partieller Nachteile der Rasterkarten gegenüber anderen Umgebungsrepräsentationen wird in der vorliegenden Arbeit vornehmlich mit Belegtheitskarten und Wahrscheinlichkeitskarten gearbeitet, da die Anwendung wahrscheinlichkeitsbasierter Methoden mit einfachen Sensormodellen begünstigt wird und die numerischen Methoden für die effiziente Ausführung programmiert werden können.

Untrennbar mit der Nutzung der Karten hängt ihre Erstellung und Aktualisie-

rung zusammen. Gemäß der oben beschriebenen Entscheidung für Rasterkarten werden in dieser Arbeit nur Methoden zur Erstellung genau diesen Kartentyps in Abschnitt 5.3 beschrieben.

5.2 Sensorik

Die für die Wahrnehmung der Umgebung eines Roboters geeignete Sensorik lässt sich anhand der Messprinzipien bzw. anhand der Reichweite klassifizieren, wobei diese Eigenschaften eng miteinander verwoben sind. Auch der Roboter selbst bringt eine Sensoreigenschaft mit, da davon auszugehen ist, dass der Aufenthaltsort des Roboters und die vom Roboter zurückgelegte Wegstrecke frei von Hindernissen sind.

5.2.1 Taktile Sensoren

Im unmittelbaren Nahbereich des Roboters eignen sich taktile Sensoren zur Umgebungserfassung, so verfügen viele Robotersysteme über berührungsempfindliche Leisten, die bei Umgebungskontakt, der in der Regel unbeabsichtigt ist, eine Reaktion auslösen. Aufgrund der geringen Reichweite sind die taktilen Sensoren als Basis für die Steuerung und Regelung mobiler Fahrzeuge ungeeignet, sie dienen meist lediglich zur Absicherung als letzte Instanz zum Zweck der Notabschaltung.

5.2.2 Time-of-Flight-Prinzip

Eine höhere Reichweite erreichen abstandsmessende Sensoren, die vielfach nach dem Time-of-Flight-Prinzip arbeiten. Hier wird von einem Sender ein Primärsignal ausgesendet, dieses wird durch die Umgebung reflektiert, so dass eine Teil des Signals als Sekundärsignal in Richtung des Senders zurückgeworfen wird. Ist die Laufgeschwindigkeit des Signals bekannt, so kann durch die Messung der Zeit zwischen dem Aussenden des Primärsignals und dem Empfang des Sekundärsignals auf den Abstand geschlossen werden. Die Genauigkeit bei der Messung ist abhängig von der Kenntnis der Signallaufgeschwindigkeit und deren gleichförmiger Geschwindigkeit, von der Genauigkeit bei der Zeitmessung, sowie von den Eigenschaften des reflektierenden Messobjekts. Bekannte Techniken sind das Sonar (Sound Navigation And Ranging), welches mit Schall als Testsignal arbeitet, das Radar[1] (Radio Detecting And Ranging) und das Laserentfernungsmesssystem (in

[1] Eine alternative Bauart des Radars arbeitet mit einer modulierten Trägerfrequenz, wobei Entfernung und Geschwindigkeit des beobachteten Objekts aus Phasenlage und Frequenzverschiebung ermittelt werden.

Anlehnung an das Radar wird dieses im Bereich der mobilen Robotik und teils im militärischen Kontext auch als Ladar bezeichnet: Laser Detection and Ranging) die mit elektromagnetischen Wellen arbeiten. Da Ultraschall weniger stark gebündelt werden kann und die Schallgeschwindigkeit eine nennenswerte Abhängigkeit von äußeren Einflussfaktoren aufweist, hat dieses Sensorprinzip eine geringere örtliche Auflösung und geringere Genauigkeit. Wie Stachniss [2006] zeigt, können jedoch auch mit Ultraschalltechnik mit geeigneten Verfahren detaillierte Umgebungskarten erstellt werden. Bei Radar und Laserentfernungsmessung werden aufgrund der hohen Ausbreitungsgeschwindigkeit erhöhte Anforderungen an die Zeitmessung gestellt. Das auf Laser-Licht basierte System lässt sich sehr genauer fokussieren. Jedoch wird die Lasermessung stark durch Witterungsbedingungen beeinträchtigt, während das Radar auch bei Feuchtigkeit, Schnee und Nebel robuste Messergebnisse liefert. Das Radarsignal enthält zudem neben der Abstandsmessung zusätzliche Informationen über die Geschwindigkeit eines Objekts. Für Anwendungen im Innenraum weist das Laserentfernungsmesssystem herausgehobene Eigenschaften bei der Abwägung von Reichweite und Genauigkeit auf. Ihm kommt in der vorliegenden Arbeit zentrale Bedeutung zu. Das Prinzip ist in Bild 5.3 dargestellt. Das

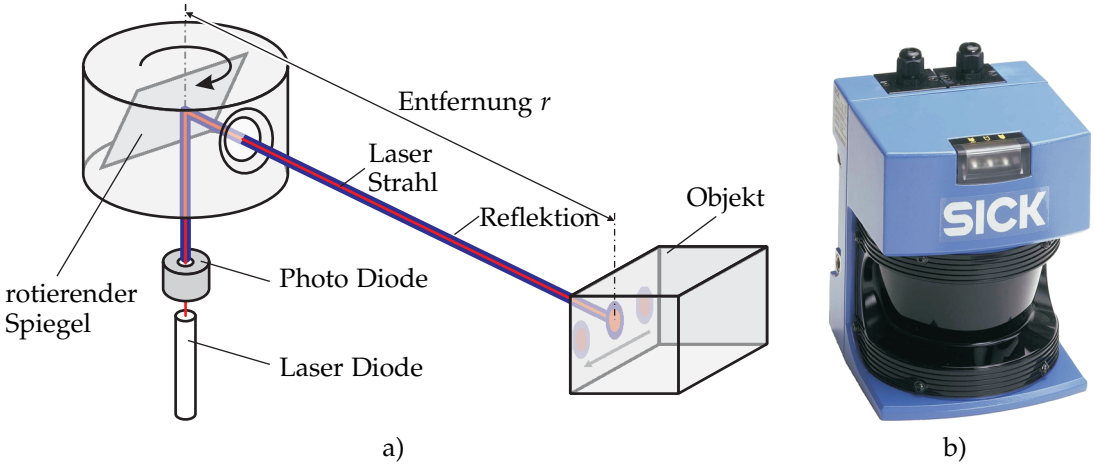


Bild 5.3: Berührungslos abtastende Lasermesssysteme der Firma Sick: a) Prinzipskizze, b) Produktfoto.

hier eingesetzte Messsystem der Firma Sick verfügt über eine Laserdiode, deren Strahl durch eine Aussparung in der Photodiode über einen rotierenden Spiegel die Umgebung abgescannt. Die Spiegellage wird über einen Winkelencoder erfasst, so dass sich die mittels einer Photodiode detektierten Messungen zur Sensorlage zuordnen lassen. Bei einer Spiegelrotation lassen sich hier 180° Messbereich in 1° Winkelauflösung erfassen. Ein detailliertes Modell des Laserstrahls wird im Zu-

sammenhang mit dem Einsatz zur MONTE-CARLO-Lokalisation dargestellt (siehe Abschnitt 6.2.3).

5.2.3 Bildverarbeitung

Von großer Bedeutung bei der Steuerung und Regelung mobiler Systeme ist die Videokamera als Sensor mit nachgelagerter Bildverarbeitung. Der breite Einsatz begründet sich in der preiswerten Hardware und einem Informationsgehalt, der der menschlichen Wahrnehmung nahekommt. Um aus Bildinformationen quantitative Informationen über die Umgebung abzuleiten, ist eine Bildverarbeitung erforderlich. Räumliche Informationen lassen sich aus der einzelnen Bildaufnahme einer einzelnen Kamera ohne Modellwissen nicht ableiten; erst wenn eine Szene aus mehreren Blickwinkeln betrachtet wird, ist die Errechnung von Tiefeninformationen möglich. Um unterschiedliche Blickwinkel zu erhalten, muss also eine monokulare Kamera räumlich bewegt werden, wobei die Berechnung der räumlichen Information durch Verfolgung von Bildmerkmalen erfolgt. Alternativ kommen sogenannte Stereokamerasysteme zum Einsatz, die über zwei Kameras an geometrisch unterschiedlichen Orten verfügen, so dass für die Bildverarbeitung stets zwei Bilder aus unterschiedlichen Blickwinkeln zur Verfügung stehen. Neuerdings werden auch monokulare Kameras mittels Spiegelsystemen zu Stereokameras aufgerüstet.

Die Herausforderungen bei der Bildverarbeitung begründen sich in der Abhängigkeit von äußeren Parametern wie Helligkeit, Witterungseinflüsse, Reflektionseigenschaften in der Umgebung etc.

5.2.4 Triangulation

Der Begriff Triangulation beinhaltet mehrere Messverfahren, wobei eines eher in den Bereich Messtechnik und das zweite eher in den Bereich der Geodäsie einzuordnen ist.

Laser-Triangulation

Die Lasertriangulation basiert auf der Fokussierung eines Laserstrahls bzw. bei geringeren Anforderungen des Lichtstrahls einer Infrarot-Diode auf ein Messobjekt, dessen Reflektion durch eine Sammellinse auf eine ortsauflösenden Fotodiode bzw. CCD[2]-Zeile gebündelt wird. Der Sensor befindet sich meist in einem Gehäuse mit der Lichtquelle. Je nach Abstand des Messobjekts vom Sensor ändert sich der Winkel, unter dem die Reflektion auf dem Empfänger registriert wird. Mit Hilfe der

[2]Charge Coupled Devices sind integrierte elektronische Bauteile, die ein zur einfallenden Lichtmenge proportionales Signal ausgeben. Durch eine Matrix-Struktur ist damit eine ortsaufgelöste Helligkeitsmessung möglich.

Winkelfunktion kann damit der Abstand vom Sensor zum Objekt bestimmt werden. Durch den rein trigonometrischen Zusammenhang kann eine kontinuierliche Messung erfolgen, auch bewegte Objekte können damit erfasst werden. Das Verfahren eignet sich für geringe Entfernungen von einigen Metern, da die Sensitivität durch den kleinen Abstand von Sensor und Empfänger mit zunehmender Entfernung abnimmt.

Geodätische Triangulation

Die Verbreitung der Triangulation in der Geodäsie begründet sich in der Tatsache, dass Winkel meist leichter zu messen sind als Abstände. Bei der Betrachtung eines Dreiecks lassen sich bei einer bekannten Seitenlänge und nach Bestimmung der Winkel zwischen den Seiten die übrigen Längen durch die Anwendung trigonometrischer Funktionen bestimmen. Um die geodätische Triangulation anwenden zu können, bedarf es eines Sensors, der den Winkel zu Landmarken bestimmen kann. Bei Anwendungen im Innenraum sind das in der Regel künstliche Landmarken, beispielsweise reflektierende Streifen, die angestrahlt durch fokussiertes Licht einen Reflektionsimpuls zurückwerfen, der mit Erfassung der Winkellage am Empfänger registriert wird. Diese Form der Messung setzt jedoch voraus, dass die Lage der Landmarken zueinander vorab bekannt ist, da sonst die obligatorische Längeninformation fehlt.

5.2.5 Weitere landmarkengestützte Systeme

Im Bereich der fahrerlosen Transportsysteme in der Produktion gibt es verschiedene Systeme die landmarkengestützt betrieben werden. Diese unterscheiden sich durch die eingesetzte Sensorik zur Erkennung der Bahn. Meist kommen hier fest verlegte Leitdrähte, optische Linien oder Magnetnägel zum Einsatz, die mit entsprechender Sensorik verfolgt werden. Diese Systeme bieten neben der Umgebungserfassung auch inhärent die Lokalisation und sogar eine Rückführungsgröße. Darüber hinaus gehen Ansätze mit Landmarken, in denen zusätzliche Informationen kodiert werden. Bei Einsatz von RFID-Tags[3] als Marke können dort über die Lageinformation hinaus die Lage anderer Transponder, semantische Informationen bzw. Informationen von anderen Robotern gespeichert werden. [Overmeyer u. a., 2006, 2007]

[3] Radio Frequency Identification

5.3 Kartenerstellung

Der Kartenerstellung kommt im Bereich der autonomen Roboter eine zentrale Rolle zu. Gerade im typischen Einsatzbereich mobiler Roboter innerhalb von Gebäuden, in Firmen, in Produktionshallen usw. gibt es oftmals keine geeignete Basis zu Erstellung von Karten aus a priori Informationen. Wenn die Grundrisszeichnungen mit den realen Gegebenheiten übereinstimmen (und auch das ist nicht selbstverständlich), so fehlen immer noch Möbel oder weitere Anlagen und Gegenstände, die das Erscheinungsbild in der Umgebung des Roboters prägen. Es ergibt sich also regelmäßig die Aufgabe, zunächst eine Umgebungskarte zu erstellen. Der Roboter, welcher mit Sensoren ausgestattet ist, um Kartenmerkmale zu sensieren, eignet sich dabei auch als Werkzeug für die Kartenerstellung. Der Aufwand für die Installation einer Robotik-Applikation vermindert sich erheblich, wenn nicht mit zusätzlichen Werkzeugen die Umgebung erfasst und eine Karte erarbeitet werden muss, sondern stattdessen eine (halb-) automatisierte Kartenerstellung mittels Roboter zur Verfügung steht. Da jedoch die Reichweite der Sensoren auf die Umgebung des Roboter beschränkt ist und in der Regel die Sicht auf Teile des Arbeitsraums versperrt ist, kann die Kartenerstellung nur inkrementell erfolgen.

Hier besteht eine besondere Wechselwirkung mit der Lokalisation, denn um eine Karte inkrementell aus Sensorinformationen von verschiedenen Orten innerhalb der Umgebung erstellen zu können, ist die präzise Information über die Relation der Aufnahmeorte zueinander erforderlich, die auch als Lage im (Karten-)Raum bezeichnet werden kann. Die Ermittlung der Lage innerhalb des Kartenraums mittels des Roboters erfordert jedoch bereits eine zur Verfügung stehende Karte. Demzufolge müssen Karte und Aufenthaltsort gleichzeitig geschätzt werden, die Methoden werden auch als *Simultaneous localization and mapping*, kurz *SLAM* bezeichnet. [Dissanayake u. a., 2001; Doucet u. a., 2000; Eliazar und Parr, 2003; Gutmann und Konolige, 1999; Hähnel u. a., 2003; Montemerlo u. a., 2002a, 2003; Murphy, 1999; Thrun, 2001; Leonard und Feder, 2000]

Das Schätzen einer Karte ist ein schwieriges Problem, da die Umgebung eines Roboters in der Regel unbeschränkt und die Dimension des Schätzproblems sehr hoch ist. Im vorliegenden Fall der Erstellung von Rasterkarten kommt schon bei kleinen Abmessungen einer Umgebung eine große Zahl von Variablen zusammen (Kartenbereich 50 m x 20 m, Kantenlänge 25 mm $\rightarrow$ 1,6 $\cdot 10^6$ Zellen). Das Problem hängt von der Größe des zu schätzenden Bereichs ab und hier insbesondere von dessen Verhältnis zur Sensorreichweite, die bedingt durch das Messprinzip, aber auch durch die Topologie der Umgebung, beschränkt sein kann. Generell leiden besonders inkrementell arbeitende Verfahren unter Messrauschen und fehlerhaften Messungen, die durch den integrierenden Charakter des Verfahrens große Schleppfehler verur-

sachen können. Voraussetzung für die erfolgreiche Anwendung von Schätzverfahren ist weiterhin, dass die Struktur der Umgebung geeignet ist, Änderungen des Aufenthaltsortes in den Sensorsignalen wiederzufinden, was bei wiederkehrenden Strukturen oder bei fehlenden Strukturen (z. B. in langen Gängen) problematisch sein kann. Bei Umgebungen mit geschlossenen Schleifen ergibt sich das Problem, dass der Roboter über eine anderen Weg zu einer bereits erfassten Stelle zurückkehrt. Der bis dahin aufsummierte Schleppfehler kann sehr groß sein, so dass die bereits bekannte Umgebung nicht wiedererkannt wird und ggf. an falscher Stelle in die Karte eingetragen wird wodurch eine inkonsistente Karte entsteht.

Bevor das gesamte SLAM-Problem dargestellt wird, soll im Folgenden zunächst die Kartierung mittels der verwendeten Sensoren, unter Annahme der präzisen Kenntnis des jeweiligen Aufenthaltsorts beschrieben werden.

5.3.1 Erstellung von Belegtheitskarten

Die inkrementelle Erstellung von Rasterkarten geht auf Moravec und Elfes [1985] zurück. Bei der Erstellung von Belegtheitskarten wird die reale Welt in ein Raster diskretisiert. Es wird dann auf Basis der Messungen $\boldsymbol{z}_{1:k}$, die zu den diskreten Zeitpunkten $1:k$ an den Orten $\boldsymbol{x}_{1:k}$ aufgenommen wurden, eine Karte

$$p(\boldsymbol{M}|\boldsymbol{z}_{1:k}, \boldsymbol{x}_{1:k}) \tag{5.1}$$

berechnet. Diese Karte $\boldsymbol{M}$ besteht aus einer Menge einzelner Zellen $\boldsymbol{M} = \{\boldsymbol{M}^{[i]}\}$. Jede einzelne Zelle beinhaltet einen Wert *belegt* oder *frei*, die Wahrscheinlichkeit, dass eine Zelle belegt ist, wird dann mit $p(\boldsymbol{M}^{[i]} = 1)$ bzw. $p(\boldsymbol{M}^{[i]} = 0)$ bestimmt. Bei Anwendung von Methoden der Wahrscheinlichkeitsrechnung auf die gesamte Karte aus Gl. (5.1) ergibt sich ein Schätzproblem sehr hoher Dimension, welches in der Regel nicht gelöst werden kann, da die Berechnung der Wahrscheinlichkeit jeder einzelnen der $2^{\text{Höhe}\cdot\text{Breite}}$ Karten zu umfangreich ist. Unter der Voraussetzung statistischer Unabhängigkeit zwischen benachbarten Zellen lässt sich das große Schätzproblem in viele kleine (binäre) Probleme unterteilen. Zwar ist die Annahme der statistischen Unabhängigkeit an dieser Stelle formal falsch, dennoch lassen sich durch diese Methode effiziente Algorithmen zur Bestimmung von Karten finden. Zu lösen ist demnach das Problem

$$p(\boldsymbol{M}^{[i]}|\boldsymbol{z}_{1:k}, \boldsymbol{x}_{1:k}) \quad , \tag{5.2}$$

welches sich unter o. g. Annahmen zur Annährung der gesuchten Karte eignet:

$$p(\boldsymbol{M}|\boldsymbol{z}_{1:k}, \boldsymbol{x}_{1:k}) = \prod_i p(\boldsymbol{M}^{[i]}|\boldsymbol{z}_{1:k}, \boldsymbol{x}_{1:k}) \quad . \tag{5.3}$$

Algorithmus 5.1 Das Bayes'sche Filter zur Erstellung von Belegtheitskarten.

```
1:  ErstelleBelegtheitskarte({l_{k-1}^[i]}, x_k, z_k)
2:      for all M^[i] do
3:          if M^[i] in Reichweite von z_k then
4:              l_k^[i] = l_{k-1}^[i] + inversesSensorModell(M^[i], x_k, z_k) − l_0
5:          else
6:              l_k^[i] = l_{k-1}^[i]
7:          end if
8:      end for
9:      return {l_k^[i]}
10: end
```

Für die Schätzung dieses binären Problems kann das Bayes'sche Filter eingesetzt werden. In Algorithmus 5.1 ist eine Implementierung des Filters dargestellt, welche zur Vermeidung von numerischen Problemen für sehr große und kleine Wahrscheinlichkeiten mit einem Logarithmen-Verhältnis der Wahrscheinlichkeiten für die Zellen der Karte arbeitet:

$$l_k^{[i]} = \log\left(\frac{p(\boldsymbol{M}^{[i]}|\boldsymbol{z}_{1:k}, \boldsymbol{x}_{1:k})}{1 - p(\boldsymbol{M}^{[i]}|\boldsymbol{z}_{1:k}, \boldsymbol{x}_{1:k})}\right) \quad . \tag{5.4}$$

Die Umrechnung dieser Darstellung in Wahrscheinlichkeiten erfolgt durch Umkehrung der obigen Gleichung:

$$p(\boldsymbol{M}^{[i]}|\boldsymbol{z}_{1:k}, \boldsymbol{x}_{1:k}) = 1 - \frac{1}{1 + \exp\left(l_k^{[i]}\right)} \quad . \tag{5.5}$$

Der Algorithmus durchläuft nun die einzelnen Zellen und aktualisiert die Wahrscheinlichkeit jeder Zelle, die in der Reichweite des Sensors liegt mittels eines inversen Sensormodells. Das inverse Sensormodell gibt an, welche Voraussetzung zu einem Messwert geführt hat. Es lässt damit einen Rückschluss aus der Messung zu, mit welcher Wahrscheinlichkeit eine Zelle belegt ist. Wie bereits erwähnt ermöglicht die Darstellung durch Logarithmenverhältnisse eine stabile Berechnung für sehr große und kleine Wahrscheinlichkeiten. Das Logarithmenverhältnis l_0 stellt gemäß

$$l_0 = \log\left(\frac{p(\boldsymbol{M}^{[i]} = 1)}{p(\boldsymbol{M}^{[i]} = 0)}\right) \tag{5.6}$$

eine Vorhersage über die Belegtheit einer bislang unbekannten Zelle dar. Die Verhältnisse l_{occ} und l_{free} stellen die Logarithmenverhältnisse für als belegt bzw. frei erkannte Zellen dar.

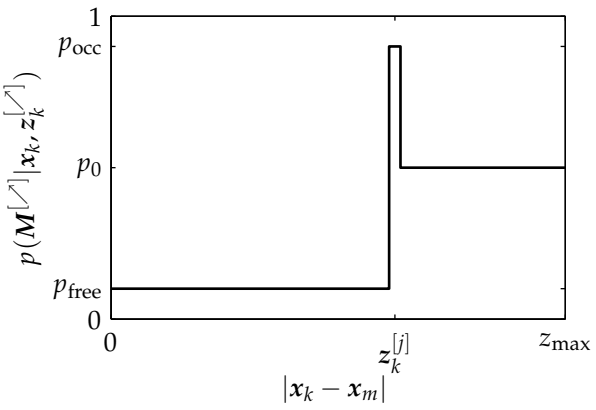


Bild 5.4: Die Belegtheitswahrscheinlichkeit ist im Bereich d_{ISM} um den Messwert hoch. Der davor liegende Bereich ist wahrscheinlich frei, über den dahinter folgenden Bereich kann keine Aussage getroffen werden.

Das inverse Sensormodell setzt die Kenntnis der Lage $\boldsymbol{x}_k = [x, y, \phi_{\text{s}}]^{\text{T}}$ des Sensorkoordinatensystems voraus. Bei Betrachtung der Zellen entlang eines Strahls ($\nearrow$), deren Position mit $\boldsymbol{x}_m$ beschrieben wird, ist die Belegtheitswahrscheinlichkeit für solche Zellen hoch, die in einem Interval d_{ISM} um den vom Sensor gemessenen Entfernungswert $\boldsymbol{z}_k^{[\nearrow]}$ liegen. Für die Zellen zwischen dem Sensorursprung und dem genannten Intervall ist die Belegtheitswahrscheinlichkeit gering und für die Zellen hinter dem Intervall kann keine zusätzliche Information gewonnen werden. Dieser Zusammenhang wird durch die Belegtheitswahrscheinlichkeit

$$p(\boldsymbol{M}^{[\nearrow]}|\boldsymbol{x}_k, \boldsymbol{z}_k^{[\nearrow]}) = \begin{cases} p_{\text{free}} & : & 0 & \leq |\boldsymbol{x}_k - \boldsymbol{x}_m| \leq & \boldsymbol{z}_k^{[\nearrow]} - \frac{d_{\text{ISM}}}{2} \\ p_{\text{occ}} & : & \boldsymbol{z}_k^{[\nearrow]} - \frac{d_{\text{ISM}}}{2} & \leq |\boldsymbol{x}_k - \boldsymbol{x}_m| \leq & \boldsymbol{z}_k^{[\nearrow]} + \frac{d_{\text{ISM}}}{2} \\ p_0 & : & \boldsymbol{z}_k^{[\nearrow]} + \frac{d_{\text{ISM}}}{2} & \leq |\boldsymbol{x}_k - \boldsymbol{x}_m| & \end{cases} \tag{5.7}$$

ausgedrückt. Diese ist in Bild 5.4 dargestellt.

Das auf Algorithmus 5.1 abgestimmte inverse Sensormodell wird in Algorithmus 5.2 beschrieben. Es geht von einem ausgedehnten Sensorstrahl mit Öffnungswinkel β_{ISM} aus. Zunächst wird das Element l des Messvektors $\boldsymbol{z}_k$ bestimmt, dessen Strahl am nächsten zur Zelle i der Karte $\boldsymbol{M}$ liegt. Dieses erfolgt durch die Berechnung des Winkels $\hat{\rho}$, der die Lage der Zelle im Sensorkoordinatensystem beschreibt (Zeile 3) und dessen Vergleich mit den Winkeln der einzelnen Strahlen $\rho_z^{[j]}$ (Zeile 4). Nach Bestimmung der zu einer Zelle gehörigen Messung, wird die Winkeldifferenz bestimmt. Liegt die Differenz innerhalb des Öffnungswinkels β_{ISM}, so wird angenommen, dass der Strahl die Zelle überstreicht. Nach Auswertung des Messwertes kann nun bestimmt werden, ob sich die Zelle innerhalb des Bereichs d_{ISM} um den Messwert befindet. In dem Fall wird angenommen, dass die

Algorithmus 5.2 Das inverse Sensormodell für Ladar-Sensoren: $\boldsymbol{x}_k = [x, y, \phi_s]^T$ kennzeichnet Lage und Orientierung des Sensors, $\boldsymbol{x}_m = [x_i, y_i]^T$ kennzeichnet den Schwerpunkt der Zelle $\boldsymbol{M}^{[i]}$, $\rho_z^{[j]}$ kennzeichnet den zu dem Element j des Messvektors $z_k = [z_k^{[0]} \dots z_k^{[I]}]^T$ gehörigen Winkel im Sensorkoordinatensystem, d_{ISM} stellt ein Interval um den Messwert $z_k^{[l]}$ dar, das als belegt gilt, β_{ISM} kennzeichnet die Aufweitung des Laserstrahls.

```
1:  InversesSensorModell(M^[i], x_k, z_k)
2:      r = sqrt((x − x_i)^2 + (y − y_i)^2)
3:      ρ̂ = atan2(y_i − y, x_i − x) − φ_s
4:      l = argmin_j |ρ̂ − ρ_z^[j]|
5:      if r > min(z_max, z_k^[l] + d_ISM/2) or |ρ̂ − ρ_z^[l]| > β_ISM/2 then
6:          return l_0
7:      else if z_k^[l] < z_max and |r − z_k^[l]| < d_ISM/2 then
8:          return l_occ
9:      else if r ≤ z_k^[l] then
10:         return l_free
11:     end if
12: end
```

Zelle belegt ist. Befindet sich die Zelle im Intervall $0 \leq z - d_{\text{ISM}}/2$ so kann sie als frei angenommen werden. Befindet sich die Zelle im Bereich $\geq z + d_{\text{ISM}}/2$ so ist sie außerhalb der Reichweite bzw. durch ein Hindernis verdeckt, hier kann keine zusätzliche Information gewonnen werden, so dass die Belegtheitswahrscheinlichkeit unverändert übernommen wird. In Bild 5.3.1 ist das inverse Sensormodell für einen einzelnen Sensorstrahl dargestellt. Wird der Algorithmus auf den Datensatz

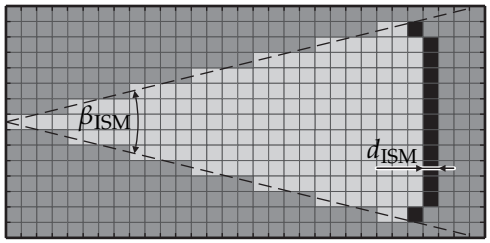


Bild 5.5: Das inverse Sensormodell für einen einzelnen Sensorstrahl mit der Strahlöffnung β_{ISM} und der Hindernistiefe d_{ISM}.

eines Laser-Scanners angewandt, so ergibt sich die Wahrscheinlichkeitsverteilung im Bereich der Sensorreichweite. Durch das Überfahren einer Fläche mit dem Sen-

sor und die Auswertung der Messdaten mit den dargestellten Algorithmen wird eine Rasterkarte der Umgebung des Roboters erstellt. Der Vorgang der Kartenerstellung ist in Bild 5.6 visualisiert: Die aus einer simulierten Fahrt durch eine virtuelle Umgebung generierten Bilder enthalten jeweils oben links den Datensatz der Sensoren, mit den Messwerten dargestellt als Punkte im kartesischen Raum, oben rechts die sich durch Anwendung des inversen Sensormodells ergebende Wahrscheinlichkeitskarte und unten die mit den Wahrscheinlichkeiten des inversen Sensormodells aktualisierte Karte. Bei den Wahrscheinlichkeitskarten wird der Wert der Zellen mit unterschiedlichen Graustufen dargestellt. Dunkle Zellen repräsentieren eine hohe Wahrscheinlichkeit der Belegtheit, helle Zellen sind mit großer Wahrscheinlichkeit frei, bei mittelgrauen Zellen ist der Zustand unsicher. Die Position des Roboters ist durch ein Symbol mit einem Kreis kleinen Durchmessers dargestellt. Die Reichweite des Sensors wird durch den großen Kreis repräsentiert.

Während bei Verwendung des ersten Datensatzes (Bild 5.6a) die Unsicherheit noch sehr groß ist (auch im von den Sensorstrahlen überstrichenen Bereichen ist noch eine deutliche graue Einfärbung zu erkennen), ist im Verlauf von Bild 5.6b bis Bild 5.6g deutlich die ansteigende Sicherheit der mehrfach durch einen Sensorstrahl überstrichenen Zellen zu erkennen. In Bild 5.6h ist schließlich die Wahrscheinlichkeitskarte zu einem fortgeschrittenen Zeitpunkt der Messung dargestellt.

Zu beachten ist hier, dass die präzise Ortsinformation des Scannerkoordinatensystems, deren Kenntnis in Algorithmus 5.1, Zeile 4 vorausgesetzt wird, üblicherweise *nicht* zur Verfügung steht. Zwar ist eine Schätzung dieser Information aus der Integration der Stellgrößen bzw. aus der Messung der zurückgelegten Wegstrecken möglich (z.B. über die Inkrementalgeber der Antriebe und Lenkung); diese Schätzung wird jedoch durch die begrenzte Genauigkeit der Messung (Schlupf an den Antriebsrädern, parametrische und systematische Unsicherheiten in der Bestimmung des kinematischen Modells, Diskretisierungsfehler) verfälscht und es bilden sich ansteigende Schleppfehler, die die geschätzte Karte unbrauchbar machen. Diese Effekte sind exemplarisch in Bild 5.7 als Vergleich zu Bild 5.6h dargestellt. Die Karte wird in beiden Fällen aus identischen Daten einer simulierten Fahrt erstellt. Während in Bild 5.6h mit den aus der Simulation verfügbaren korrekten Ortsinformationen gearbeitet wird, erfolgt in Bild 5.7 die Schätzung der Ortsinformation aus den aufintegrierten Odometriedaten. Diese werden unter Zuhilfenahme des kinematischen Robotermodells durch die Umrechnung der Steuerbefehl in virtuelle Messdaten erzeugt und anschließende Schätzung des Roboterzustands mittels eines Erweiterten KALMAN-Filters erzeugt. Da in der Simulation mit einem perfekten Modellwissen gearbeitet wird, kommen als Fehlerquellen lediglich Diskretisierungsfehler durch die zeitliche Abtastung ($\Delta T = 50\,ms$) und numerische Ungenauigkeiten in Betracht. Wie in Bild 5.7 dargestellt führen bereits geringe Fehler zur

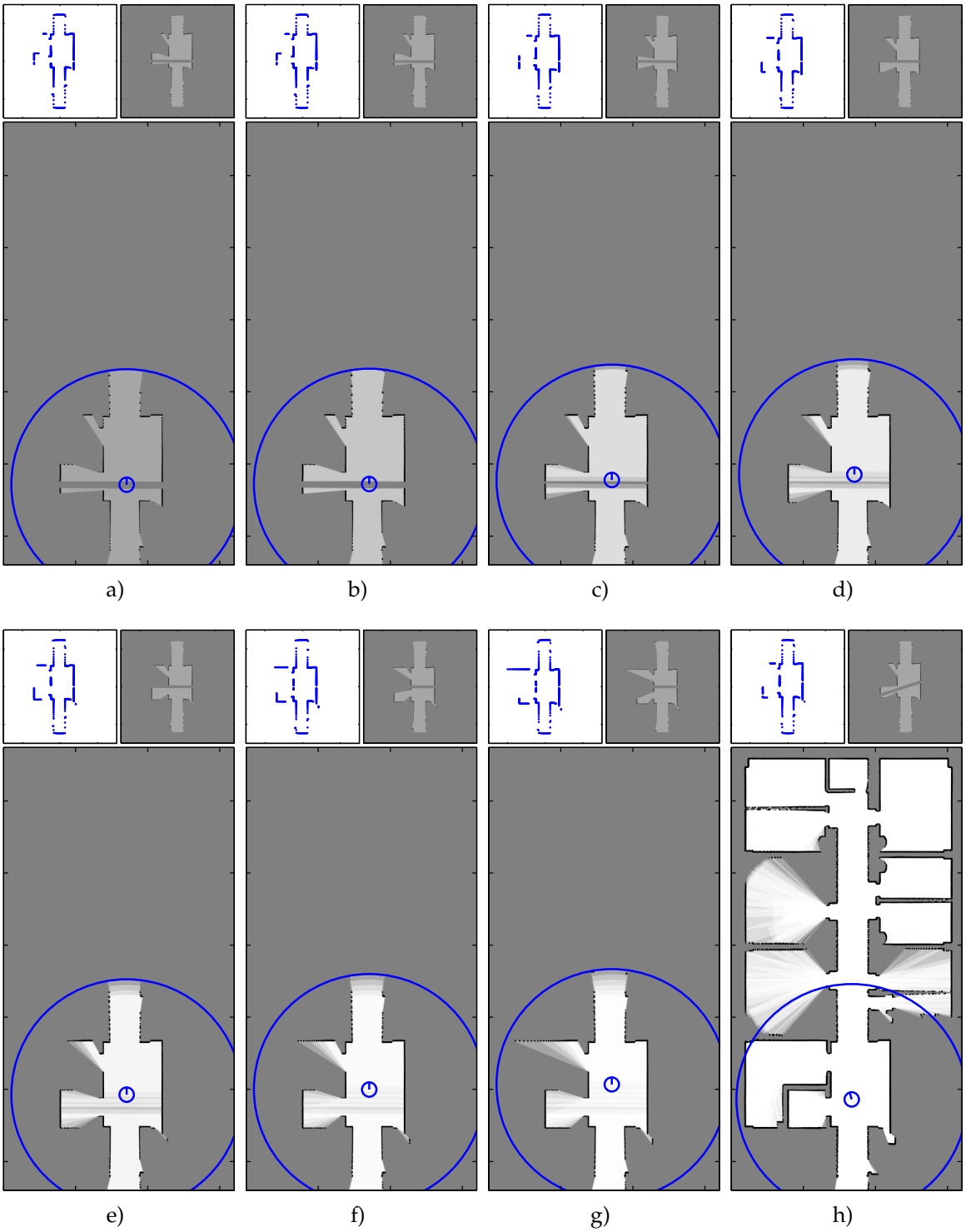


Bild 5.6: Der Prozess der Kartenerstellung verarbeitet die Sensorinformationen in Abhängigkeit der hier bekannten Roboterposition. Dargestellt sind die ersten sieben Messungen sowie die resultierende Karte unter Berücksichtigung aller Messungen.

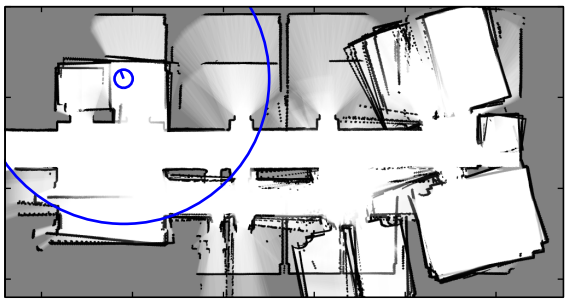


Bild 5.7: Der Prozess der Kartenerstellung aus Sensorinformationen erfordert präzise Daten über die Roboterposition. Der Versuch, diese Informationen ausschließlich durch eine Integration der Odometriedaten zu erhalten, schlägt, wie hier dargestellt, fehl.

Schätzung einer unbrauchbaren Karte.

5.3.2 Gleichzeitige Lokalisation und Kartenerstellung

Soll der Roboter mit seinen Sensoren als Werkzeug benutzt werden, um eine Umgebungskarte des Arbeitsbereichs anzufertigen, so muss gleichzeitig die Karte und die Lage des Roboters geschätzt werden. Auch hier gilt selbstverständlich, dass nicht alle möglichen Karten und alle möglichen Positionen berechnet werden können, um diese dann mittels der Sensordaten zu gewichten. Es wird vielmehr der gleiche Ansatz wie in Abschnitt 5.3.1 verwendet. Gesucht wird die Karte $\boldsymbol{M}$ sowie entweder die aktuelle Position $\boldsymbol{x}_k$ oder die zurückgelegte Bahn $\boldsymbol{x}_{1:k}$ auf Basis der Sensormessungen $\boldsymbol{z}_{1:k}$ und der Stellgrößen $\boldsymbol{u}_{1:k}$

$$p(\boldsymbol{M}, \boldsymbol{x}_{1:k} | \boldsymbol{z}_{1:k}, \boldsymbol{u}_{1:k}) \quad . \tag{5.8}$$

In der Literatur sind verschiedene Ansätze für das genannte Problem zu finden, welche jeweils auf die Repräsentation der Umgebung abgestimmt sind [Castellanos u. a., 1999; Dissanayake u. a., 2001; Leonard und Feder, 1999]. Der größte Teil dieser Ansätze arbeitet mit merkmalsbasierten Karten auf Basis von KALMAN-Filtern oder dessen Derivaten, wobei die Zahl der Merkmale beschränkt und die zulässige Nichtlinearität von Zustandsgleichung und Messgleichung harten Restriktionen unterliegen. Für die Behandlung von nichtlinearen Effekten in der Zustandsschätzung wurde in Kapitel 3 das nichtparametrische Partikelfilter eingeführt. Die Leistungsfähigkeit des Partikelfilters hängt stark von der Dimension des Zustandsraum ab. Eine direkte Anwendung auf das Kartenschätzproblem ist daher nicht möglich. Zunächst lässt sich die Unabhängigkeitsbedingung aus Gl. (5.3) auch hier anwenden, d.h. bei Annahme einer Position des Roboters im Raum lassen sich die einzelnen

Merkmale einer Karte unabhängig von einander aktualisieren. Die einzige Abhängigkeit zwischen den Kartenmerkmalen einer Karte besteht damit im Zustand des Roboters $\boldsymbol{x}_k$. Ein Derivat des Partikelfilters stellt das sogenannte Rao-Blackwellized-Partikelfilter dar. Dieses nutzt Partikel, um die Wahrscheinlichkeitsdichte für einzelne Zustände zu erfassen, sowie parametrische Beschreibungen für weitere Zustände [Doucet u. a., 2001]. Die Anwendung dieser Technik auf das Lokalisations- und Kartenerstellungsproblem geht auf Murphy [1999] zurück.

Unter Anwendung von Bayes Regel wird Gl. (5.8) zu

$$p(\boldsymbol{M}, \boldsymbol{x}_{1:k} | \boldsymbol{z}_{1:k}, \boldsymbol{u}_{1:k}) = \\ \eta \cdot p(\boldsymbol{z}_k | \boldsymbol{M}, \boldsymbol{x}_k) \cdot \int p(\boldsymbol{x}_k | \boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) p(\boldsymbol{x}_{1:k-1}, \boldsymbol{u}_{0:k-2}) \mathrm{d}\boldsymbol{x}_{1:k-1} \quad (5.9)$$

umgeformt. Nach Rao und Blackwell wird alternativ diese Wahrscheinlichkeit $p(\boldsymbol{M}, \boldsymbol{x}_{1:k} | \boldsymbol{z}_{1:k}, \boldsymbol{u}_{1:k})$ durch

$$p(\boldsymbol{M}, \boldsymbol{x}_{1:k} | \boldsymbol{z}_{1:k}, \boldsymbol{u}_{1:k}) = \underbrace{p(\boldsymbol{M} | \boldsymbol{x}_{1:k}, \boldsymbol{z}_{1:k}, \boldsymbol{u}_{0:k-1})}_{\text{Kartenschätzung}} \cdot \underbrace{p(\boldsymbol{x}_{1:k} | \boldsymbol{z}_{1:k}, \boldsymbol{u}_{0:k-1})}_{\text{Lokalisation}} \quad (5.10)$$

ausgedrückt, wobei der erste Term das Schätzproblem der Karte bei gegebenem Zustand, also ein bekanntes Problem aus Abschnitt 5.3.1 wiedergibt, während der zweite Term das Lokalisationsproblem repräsentiert. Die Roboterlage bzw. -bahn wird durch die Partikelwolke $\boldsymbol{\chi}_{1:k}$ repräsentiert, deren einzelnen Partikeln $\boldsymbol{x}_{k,[m]} = \left[[x, y, \phi]^{\mathrm{T}}, \boldsymbol{M}\right]$ jeweils eine eigene Karte $p(\boldsymbol{M} | z_{1:k}, x_{1:k})$ zugeordnet ist. Auf das Lokalisationsproblem $p(\boldsymbol{x}_{1:k} | \boldsymbol{z}_{1:k}, \boldsymbol{u}_{0:k-1})$ wird an dieser Stelle nicht näher eingegangen, da es als Gegenstand von Abschnitt 6.2 später detailliert erläutert wird.

Das Kartierungsproblem für ein Partikel ist jeweils unabhängig von dem anderer Partikel, so dass auch hier ein großes Schätzproblem in viele kleine Schätzprobleme aufgeteilt werden kann. Das Daten-Assoziationsproblem wird für jedes Partikel unabhängig gelöst. Im Gegensatz zu den etablierten SLAM-Methoden speichert das Filter damit nicht nur die wahrscheinlichste Datenassoziation, sondern über die verschiedenen Partikel und die zugeordneten Karten auch weitere.

Zusammengefasst wird die Anwendung des Partikelfilters auf die Schätzung der Wahrscheinlichkeitskarte in Algorithmus 5.3. Dargestellt ist eine Iteration des Filters. Als Eingangsgrößen dienen die Partikelwolke $\boldsymbol{\chi}_{k-1}$ mit den Hypothesen über die Roboterposition und der jeweiligen Karte aus dem vorherigen Zeitschritt, weiterhin die Stellgröße $\boldsymbol{u}_k$, die im aktuellen Zeitschritt auf den Roboter wirkt, sowie die aktuelle Sensormessung $\boldsymbol{z}_k$. In Zeile 4 wird das Bewegungsmodell des Roboters (siehe Abschnitt 6.2.1) auf die Partikel angewandt, welches die Zustandsänderung durch Einwirken der Eingangsgrößen berücksichtigt. In der Praxis werden hier bevorzugt Odometrie-Daten statt der Stellgrößen eingesetzt, die als gemessene Daten

Algorithmus 5.3 Der SLAM-Algorithmus für Wahrscheinlichkeitskarten

1: SlamAlgorithm(χ_{k-1}, u_k, z_k)
2: $\bar{\chi}_k = \chi_k = \emptyset$
3: **for** $m = 1$ **to** M **do**
4: $x_{k,[m]} = \text{sampleMotionModel}\left(u_k, x_{k-1,[m]}\right)$
5: $w_{k,[m]} = \text{measurementModelMap}\left(z_k, x_{k,[m]}, M_{k-1,[m]}\right)$
6: $M_{k,[m]} = \text{updateOccupancyGrid}\left(z_k, x_{k,[m]}, M_{k-1,[m]}\right)$
7: $\bar{\chi}_k = \bar{\chi}_k + \langle x_{k,[m]}, M_{k,[m]}, w_{k,[m]} \rangle$
8: **end for**
9: **for** $k = 1$ **to** M **do**
10: draw i with probability $\sim w_{k,[i]}$
11: add $x_{k,[i]}$ to χ_k
12: **end for**
13: **return** χ_k
14: **end**

eine präzisere Schätzung der Karte ermöglichen. In Zeile 5 wird das Messmodell angewendet, um die Plausibilität der Roboterposition in der partikeleigenen Karte zu bewerten. Diese Bewertung wird im folgenden Resampling-Prozess als Gewicht des Partikels genutzt. In Zeile 6 erfolgt die Aktualisierung der jeweiligen Karte unter der Annahme, einer korrekten Roboterposition, wiederum für jedes einzelne Partikel. In den Zeilen 9–12 erfolgt schließlich das Resampling, hier werden nun Partikel mit hohem Gewicht vervielfältigt und solche mit geringem Gewicht ersetzt.

Durch die wiederholte Anwendung der Iteration werden die Karten der Partikel aufgebaut. Der Resampling-Prozess stellt sicher, dass nur solche Partikel bestehen, bei denen die Roboterposition in der partikeleigenen Karte durch Betrachtung der Sensormessung eine hohe Wahrscheinlichkeit aufweist.

5.3.3 Experimentelle Erprobung der Kartenerstellung

Der oben genannte Algorithmus wird mit der in Kapitel 4 vorgestellten Versuchseinrichtung erprobt. Die Partikel werden mit einer identischen Roboterposition und einer leeren Wahrscheinlichkeitskarte $M^{[0\ldots I]} = l_0$ initialisiert. Im Bewegungsmodell wird statt der Stellgrößen der nach Abschnitt 4.2 geschätzte Roboterzustand eingesetzt. Die Anwendung des Bewegungsmodells erfolgt mit der Taktrate der Geschwindigkeitsschätzung von 50 ms. Die Verarbeitung der Sensormessung mit Auswertung des Messmodells, Aktualisierung der Karte und Resampling erfolgt mit einer Taktrate von 100 ms. Die Laserentfernungsmesssysteme werden dabei

mit einer Winkelauflösung von 0,5° betrieben. Der Roboter wird manuell bei einer Geschwindigkeit von ca. 250 mm/s gesteuert. Dabei wird eine Bahn innerhalb einer Büroumgebung abgefahren. Die Zeit für die Aufnahme der Messwerte beträgt ca. 3 1/2 min. Die anschließende Kartenschätzung greift auf die aufgezeichneten Daten der Laserentfernungsmesssysteme und den aus Odometriedaten geschätzten Zustand des Roboters zurück. Bei dem in dieser Arbeit vorliegenden Implementierungsstand dauert die Berechnung auf gängiger Computerhardware (Intel Pentium M mit 1,8 GHz, 2 GB Arbeitsspeicher) noch einige Stunden, wobei noch Optimierungspotential bezüglich eines effizienteren Programmcodes besteht. Die resultierende Karte ist in Bild 5.8 dargestellt. Die Darstellung zeigt die Karte in verschiedenen Entstehungsstadien. Im Ausschnitt oben links ist jeweils der aktuelle Datensatz der Laserentfernungsmesssysteme dargestellt. Im Ausschnitt oben rechts ist die konzentrierte Partikelwolke zu sehen. Der Ausschnitt unten zeigt den aktuellen Zustand der Karte wobei die Reichweite der Laserentfernungsmesssysteme durch den Kreis angedeutet ist. Die vom Roboter zurückgelegte Bahn ist in die Karte eingezeichnet.

Es ist deutlich zu erkennen, dass auch in den mehrfach aufgesuchten Orten eine konsistente Karte geschätzt wird. In einigen Bereichen wirken sich Fehlmessungen der Laserentfernungsmesssysteme aus: Die hinter den Wänden liegenden hell eingefärbten Bereiche, insbesondere unten rechts und oben links in Bild 5.8h, weisen auf gestörte Messungen hin. Im betreffenden Bereich unten rechts befinden sich Fensterscheiben mit einer rauen Oberfläche, die durch die diffuse Reflektion zu einer stark verminderten Rückstreuung der Messstrahlen in Richtung des Sensors führen und die Detektion der Reflektion erschweren. Dieses führt auf häufige Fehlmessungen, die die fälschliche Annahme von freier Fläche hinter den Scheiben verursachen. Im zweiten genannten Bereich oben links in der Karte befindet sich ein mit Stühlen und Tischen ausgestatteter Raum. Durch die geringen Abmessungen von Stuhl- und Tischbeinen können diese mittels des aufgeweiteten Laserstrahls nicht zuverlässig detektiert werden. Es kommt dort zu mehrfachen Reflektionen, die die Messung stören. Auch hier führt die falsche Entfernungsinformation zu einer fehlerhaften Annahme unbelegter Bereiche. Der Grund dafür, dass neben den fehlerhaft als unbelegt angenommenen Bereichen keine fehlerhaft als belegt dargestellte Bereiche zu erkennen sind, liegt in der Streuung der Fehlmessungen: Feste Strukturen können sich nur dann etablieren, wenn eine Zelle mehrfach als belegt erkannt wird. Wie in Bild 5.4 dargestellt ist dieser Bereich jedoch eng um den Messwert konzentriert, während mehrfache Überstreichungen mit der Information unbelegt durch den großen Bereich über der Strahllänge häufiger auftreten.

Zusammenfassend lässt sich der Prozess der Kartierung wie folgt beschreiben: In einer ersten Phase kann auch der ungeschulte Nutzer durch die handgeführte

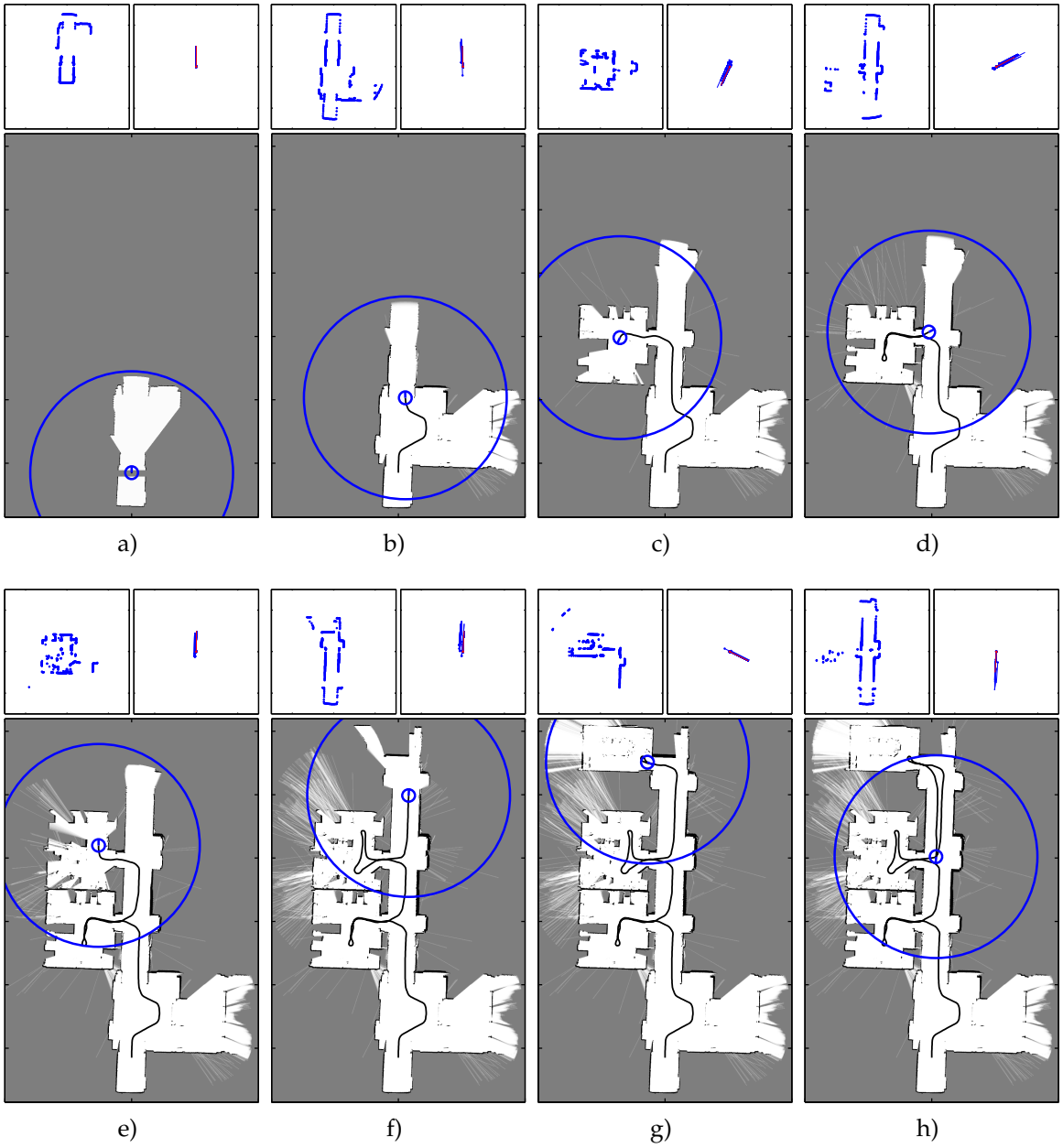


Bild 5.8: Analog zu Bild 5.6 ist hier der Prozess der Kartenerstellung bei gleichzeitiger Schätzung der Roboterposition und der Karte aus Messdaten dargestellt.

Exploration einer unbekannten Umgebung Messdaten für die Kartenerstellung aufnehmen. Dieser Prozess dauert je nach Größe der Einsatzumgebung einige Minuten. In einer zweiten Phase erfolgt die Kartenschätzung ohne weiteren Nutzereingriff. Dieser Prozess kann, je nach Effizienz der Implementierung, bis zu einigen Stunden Zeit in Anspruch nehmen. Die durch Fehlmessungen entstandenen Artefakte können nach Abschluss der Kartenschätzung eine geringfügige Nachbearbeitung erforderlich machen, bevor die Karte zum Einsatz in Bahnplanung und Lokalisation zur Verfügung steht.

6 Lokalisation mobiler Fahrzeuge

Die Lokalisation eines Roboters innerhalb bekannter oder unbekannter Umgebung ist eine der größten Herausforderungen beim Betrieb autonomer Robotersysteme. Die Qualität der Orts- und Lagebestimmung bestimmt in hohem Maß den Autonomiegrad eines mobilen Systems. Unter Lokalisation wird die Bestimmung der Transformation zwischen einem globalen ortsfesten Koordinatensystem und dem lokalen mit dem Roboter bewegten Koordinatensystem verstanden. Die Position kann in der Regel nicht direkt gemessen werden, sondern muss aus den Signalen von verschiedenen Sensoren abgeleitet werden. In den letzten Jahren wurden zahlreiche Lokalisationsverfahren auf Basis verschiedenster Prinzipien entwickelt. Wesentliche Elemente zur Klassifikation unterschiedlicher Lokalisationsverfahren sind die Eigenschaften

- a priori Informationen vorhanden/erforderlich,
- natürliche oder künstliche Landmarken erforderlich,
- globale oder lokale Positionsbestimmung,
- absolute oder relative Positionsbestimmung,
- Art der Umgebungsrepräsentation und
- Art der eingesetzten Sensoren.

Die genannten Eigenschaften beeinflussen maßgeblich die Qualität der Schätzung des aktuellen Aufenthaltsorts.

In den folgenden Abschnitten werden die grundlegenden Lokalisationsverfahren mit den entsprechenden Sensorsystemen und Umgebungsmodellen vorgestellt. Anschließend wird das im Bereich der Robotik derzeitig als überlegen geltende Verfahren der Monte-Carlo-Lokalisation vorgestellt, das im Rahmen dieser Arbeit mit einem effizienten *Raycasting* Algorithmus ausgestattet wird und so eine effiziente Lokalisation mit Hilfe von Rasterkarten ermöglicht. Schließlich wird das durch den Rechenaufwand bedingt langsam getaktete Verfahren durch eine Mehrratenansatz für den Einsatz in schnell getakteten Regelkreisen qualifiziert.

6.1 Lokalisationsverfahren

Die Notwendigkeit der Lokalisation ergibt sich nicht erst durch den Einsatz mobiler Robotersysteme, sondern spielt in den verschiedenen Bereichen des Land-, Wasser- und Luftverkehrs eine zentrale Rolle. Nicht zuletzt aus diesem Grund existieren zahlreiche Verfahren zur Lokalisation, die jeweils auf ein bestimmtes Einsatzgebiet abgestimmt sind. Im Folgenden werden die Merkmale und Eigenschaften der Verfahren kurz dargestellt. In Abschnitt 6.1.1 werden inkrementell arbeitende Verfahren dargestellt, also Verfahren, die relative Positionsänderungen auswerten und über eine Integration die aktuelle Position bestimmen. Abschnitt 6.1.2 widmet sich Verfahren, die durch die Bestimmung einer Relation zu ortsfesten, bekannten Landmarken die eigene Position errechnen. Passive Systeme nutzen zur Lokalisation Sensoren, welche nicht am Fahrzeug befestigt sind. Hier wird die Position eines Fahrzeugs von außen gemessen, die Darstellung erfolgt in Abschnitt 6.1.3.

6.1.1 Inkrementelle Verfahren

Ist die aktuelle Position bekannt, so kann nach einer Positionsänderung durch die Bestimmung der zugehörigen Transformation die neue Position bestimmt werden. Durch eine wiederholte Anwendung dieser Vorgehensweise, kann also stets die neue Position durch eine Integration über die Positionsänderungen vom Ausgangsort bestimmt werden. Für die Bestimmung der Transformation existieren verschiedene Verfahren, deren Fehlerpotential unterschiedlich ist. Bekannte Beispiele für die Anwendung sind die Koppelnavigation in der Seefahrt oder der Einsatz von Inertialplattformen in der Luftfahrt. Wesentlich für die Genauigkeit dieses Verfahrens ist die Genauigkeit, mit der die Transformation bestimmt werden kann. Eine Eigenschaft dieses Verfahrens ist, dass Fehler nicht ausgeglichen werden können, sondern mit den Positionsänderungen aufsummiert werden. Den inkrementellen Verfahren ist damit gemein, dass sie einem steigenden Fehler unterliegen, je länger das Lokalisationsverfahren läuft, ohne dass eine Korrektur vorgenommen wird.

In der Robotik sind die Antriebe und Lenkungen in der Regel mit Sensoren ausgestattet, die den Drehwinkel oder die Drehgeschwindigkeit messen und unter Zuhilfenahme des kinematischen Modells einen Rückschluss auf den Bewegungszustand des Roboters zulassen (auch Odometrie genannt). Die Fehlerquellen bei dieser Art der Positionsschätzung sind verschiedene systematische und stochastische Effekte wie Schlupf an den Rädern, Diskretisierungsfehler durch die Abtastung eines kontinuierlichen Signals, ein fehlerhaftes kinematisches Modell durch nicht berücksichtigte Effekte wie Elastizitäten im Antriebsstrang bzw. im Aufbau, parametrische Unsicherheiten wie ungenau bestimmte Raddurchmesser und Ferti-

gungstoleranzen, unbekannte Gewichtsverteilung etc.

Die bislang genannten Verfahren verwenden ausschließlich Zustandsgrößen des jeweiligen Fahrzeugs für die Positionsbestimmung. Ein Beispiel für die Nutzung extrinsischer Sensoren ist das sogenannte inkrementelle Scan-Matching. Ist ein Roboter mit einem Laserentfernungsmesssystem wie in Abschnitt 5.2.2 ausgestattet, so kann die Transformation zwischen je einem an der Ausgangsposition und an der aktuellen Position aufgenommenen Datensatz mittels eines iterativen Verfahrens bestimmt werden (z. B. ICP-Algorithmus [Besl und McKay, 1992]). Diese Transformation entspricht der gesuchten Information über die relative Positionsänderung. Eine geeignete Struktur der Umgebung vorausgesetzt, kann das Verfahren vergleichsweise genaue Schätzungen liefern. Auch wenn der Fehler hier langsamer ansteigt, bleibt die grundsätzliche Problematik der Schleppfehler bei inkrementellen Verfahren jedoch bestehen.

6.1.2 Absolut messende Verfahren

Im vorhergehenden Abschnitt wurde bereits auf den Nachteil des sich aufintegrierenden Schleppfehlers bei inkrementell arbeitenden Verfahren zur Positionsbestimmung hingewiesen. Die einzige Möglichkeit dieses Problem zu lösen ist die gelegentliche Bestimmung der eigenen Position gegenüber externen Referenzen, um so die Unsicherheit auf ein akzeptables Maß zu begrenzen. Für die Lokalisation ist eine Karte mit den darin verzeichneten Referenzen erforderlich. In diesem Abschnitt werden die Verfahren nach dem Prinzip der Bestimmung der eigenen Position gegenüber den externen Referenzen klassifiziert. Eine weitere Klassifikation ist nach Beschaffenheit der Referenzen möglich.

Als Referenzen eignen sich natürliche Elemente der Umgebung. Eine Umgebungskarte vorausgesetzt kann ein mit Laserentfernungsmesssystemen ausgestatteter Roboter mittels Scan-Matching die eigene Position innerhalb dieser Umgebungskarte durch die Maximierung der Übereinstimmung von Position und Messung ermitteln. Erforderlich ist eine geeignete Struktur der Umgebung und die Übereinstimmung der Karte mit der Realität.

Bei einem globalen Satellitennavigationssystem, z. B. GPS, dienen Satelliten als ortsfeste Referenzen. Diese übertragen per Funk fortwährend ihre präzise Position, sowie eine präzise Uhrzeit an den Empfänger. Durch Auswertung der Laufzeit des Signals kann auf die Entfernung zum Satelliten geschlossen werden. Zur Bestimmung der Position im Raum werden die Entfernungen dreier Satelliten benötigt. In der Regel wird wenigstens das Signal eines weiteren Satelliten benötigt, um eine Referenzzeit für die Laufzeitmessung zu bestimmen. Ähnliche Funktionsprinzipien werden innerhalb von Gebäuden genutzt.

Bevorzugt mit künstlichen Landmarken arbeiten Triangulationsmethoden (vergl. Abschnitt 5.2.4). Durch gleichzeitige Messung der Winkel zwischen mindestens drei Landmarken, deren Positionen bekannt sind, lässt sich die aktuelle Position bestimmen. Mittels eines rotierenden Laserstrahls und durch Auswertung der Reflektion lassen sich mit Hilfe von stark reflektierenden Materialien zuverlässig ggf. codierte Landmarken erkennen. [McGillem und Rappaport, 1989; Cohen und Koss, 1992; Betke und Gurvits, 1997]

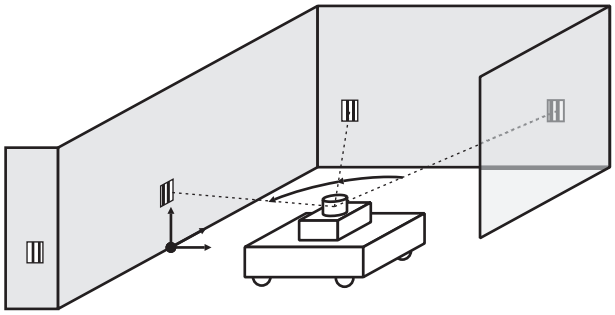


Bild 6.1: Bei bekannten Markenpositionen kann mittels geodätischer Triangulation die Position des Roboters bestimmt werden.

6.1.3 Externe Messsysteme

Bei den o. g. Systemen wird die Position jeweils auf dem Fahrzeug ermittelt. In speziell vorbereiteten Umgebungen lässt sich die Position eines Fahrzeugs auch von außen beobachten, um diese dann dem Fahrzeug auf einem Kommunikationskanal mitzuteilen. Bevorzugt zu nennen sind hier Kamerasysteme, die an der Gebäudedecke befestigt werden um die Szene von oben zu beobachten (z. B. im Roboterfußball). Durch Markierungen auf dem bzw. den Roboter(n) lassen sich Roboter mittels Bildverarbeitung im Videobild finden und unterscheiden. Für die Bewegungsanalyse bieten sich Mehrkamerasysteme an, welche mit entsprechender Bildverarbeitung, ggf. gestützt durch Marker die im Videobild gut zu verfolgen sind, zur Analyse natürlicher Bewegungsmuster eingesetzt werden. Fod u. a. [2002] setzen ortsfest aufgestellte Laserentfernungsmesssysteme ein, um bewegte Objekte und Personen zu verfolgen. Schließlich existieren hochpräzise Lasertrackingsysteme, die bevorzugt in der Fertigungsmesstechnik von außergewöhnlich großen Bauteilen im Flugzeugbau etc. eingesetzt werden [Leica, 2006].

6.2 MONTE-CARLO-Lokalisation

Aus dem Genannten wird ersichtlich, dass die konventionellen Lokalisationsverfahren für viele Anwendungen in der mobilen Robotik ungeeignet sind: Die inkrementellen Verfahren arbeiten nur über einen kurzen Zeithorizont zuverlässig, aus den Messungen lässt sich zudem kein Maß über die Qualität der Ortsschätzung ableiten. Die absolut messenden Verfahren erfordern die Installation von externen Systemen bzw. ein aufwändiges Kartierungsverfahren.

Kennzeichen des hier verfolgten Ansatzes ist, dass möglichst geringe Anforderungen an die Einsatzumgebung gestellt werden, dass die Umgebung nicht speziell für den Einsatz von Robotersystemen vorbereitet werden muss und dass der Aufwand für die Einrichtung des Systems gering ist. Auch auf a priori Wissen soll nach Möglichkeit nicht zurückgegriffen werden. Daraus lässt sich ableiten, dass externe Messsysteme weder für die Vermessung der Einsatzumgebung noch für den späteren Einsatz der Robotersysteme zur Anwendung kommen können. Um den Anforderungen in Innenraum-Umgebungen gerecht zu werden, ist der Einsatz von Satellitennavigation nicht möglich. Damit bleibt die Mitführung der erforderlichen Sensorik auf dem Roboter. Gewählt wird die Arbeit mit einer Rasterkarte, welche mit dem Roboter als Werkzeug mit den Methoden aus Abschnitt 5.3.2 erstellt werden kann. Zum Einsatz kommen dabei Laserentfernungsmesssysteme nach Bild 5.3.

Die Aufgabe der Lokalisation mit einem Roboter in der direkten Umgebung des Menschen erfordert Maßnahmen zum Umgang mit der inhärenten Unsicherheit. Aus diesem Grund werden in dieser Arbeit wahrscheinlichkeitsbasierte Verfahren verwendet. Im Folgenden soll das parametrische Filter (Partikelfilter) auf die genannte Problemstellung angewandt werden. Die Lokalisation mittels des skizzierten Ansatzes wird auch als MONTE-CARLO-Lokalisation bezeichnet [Fox u. a., 2001]. Das Verfahren verarbeitet direkt ungefilterte Sensordaten. Es gibt keinen Vorverarbeitungsschritt in dem etwa zunächst Merkmale extrahiert werden müssen. Es werden damit auch negative Informationen nutzbringend verarbeitet. Durch die nichtparametrische Arbeitsweise des Filters können multimodale Situationen abgebildet werden, so dass auch die häufig auftretenden Mehrdeutigkeiten gelöst werden können.

Die Positionsschätzung wird hier durch eine Partikelwolke χ_k repräsentiert, deren einzelne Partikel $\boldsymbol{x}_{k,[m]}$ jeweils eine Hypothese über die Position des Roboters zum Zeitpunkt k darstellen. Der Zustand beinhaltet eine Position in x-Richtung und y-Richtung bezüglich einer ortsfesten Referenzposition, sowie den Winkel ϕ_r, den die Fahrzeugachse und die x-Achse des Referenzkoordinatensystems miteinander

einschließen:

$$\boldsymbol{x}_{k,[m]} = \begin{bmatrix} x \\ y \\ \phi \end{bmatrix} \quad . \tag{6.1}$$

Die Ursache für eine Positionsänderung ist die Stellgröße $\boldsymbol{u}_k$ die im vorhergehenden Zeitschritt berechnet wurde. Die Stellgröße des Roboters besteht für ein allgemeines Fahrzeug aus den translatorischen Geschwindigkeiten im mitbewegten Koordinatensystem sowie der Winkelgeschwindigkeit um die z-Achse ω:

$$\boldsymbol{u}_k = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \omega \end{bmatrix} \quad . \tag{6.2}$$

Diese allgemeine vom kinematischen Modell des Roboters entkoppelte Schnittstelle ermöglicht die Anwendung der hier dargestellten Formalismen auf verschiedene Robotersysteme. In der Regel stehen bei Einsatz eines Robotersystems auch die zurückgelegten Wegstrecken aus der Odometrie zur Verfügung. Im vorliegenden Fall werden aus den Odometriedaten mittels eines Erweiterten KALMAN-Filters die Geschwindigkeiten im mitbewegten Koordinatensystem ermittelt (vergleiche hierzu Abschnitt 4.2). Die gemessenen Geschwindigkeiten liegen damit im gleichen Format wie die Stellgrößen vor und können statt diesen als Stellgrößen eingesetzt werden. Auch wenn dieses aus systemtheoretischer Sicht eine Unschärfe darstellt, lassen sich dadurch erheblich bessere Ergebnisse erzielen, da die Abweichung zwischen den Stellgrößen und den tatsächlich abgefahrenen Größen, abhängig von der Güte der Regelung, vom Arbeitspunkt etc., mitunter beachtlich groß werden kann. Die Zusammenfassung einer Messung des Laserentfernungsmesssystems erfolgt im Messvektor $\boldsymbol{z}_k$ der als Elemente die J fächerförmig gemessenen Entfernungen vom Ursprung des Sensors zum jeweils nächstliegenden Objekt beinhaltet:

$$\boldsymbol{z}_k = \begin{bmatrix} r_0 \\ r_1 \\ \vdots \\ r_{J-1} \end{bmatrix} \quad . \tag{6.3}$$

Je nach Ausführung und Einstellung des Sensors kann J variieren. Im vorliegenden Fall wird überwiegend mit $J = 361$ gearbeitet, daraus ergibt sich in Verbindung mit dem Winkelbereich von 180° eine Winkelauflösung von 0,5°.

Im Folgenden sollen die Grundlagen zur Anwendung des Partikelfilters auf die gegebene Aufgabenstellung dargestellt werden.

6.2.1 Bewegungsmodell

Zunächst ist ein Bewegungsmodell erforderlich, welches einen Zusammenhang zwischen der Stellgröße und dem Zustandsübergang auf den Zeitschritt k darstellt. Es ist also die Wahrscheinlichkeit

$$p(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{u}_{0:k}) \tag{6.4}$$

für $\boldsymbol{x}_k$ bei gegebener Historie des Zustands $\boldsymbol{x}_{0:k-1}$ und gegebener Stellfolge $\boldsymbol{u}_{0:k}$ gesucht. Unter der Annahme einer MARKOV-Kette 1. Ordnung gilt dabei, dass der aktuelle Zustand nur vom Zustand des vorherigen Zeitschritts abhängig ist, und damit

$$p(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{u}_{0:k}) = p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{u}_k) \quad . \tag{6.5}$$

Während für die Anwendung parametrischer Filter das Bewegungsmodell stets in geschlossener Form vorliegen muss, genügt für die Anwendung von Partikelfiltern ein Modell aus dem einzelne Stichproben gezogen werden können. Dieses kann sehr effizient erfolgen. Ein Beispiel für ein einfaches Bewegungsmodell beschreibt Thrun u. a. [2005]. In Algorithmus 6.1 wird ein auf omnidirektionale Plattformen abgestimmtes Bewegungsmodell beschrieben.

Die Funktion **sampleGaussian**(μ, σ) gibt eine normalverteilte Zufallszahl mit Mittelwert μ und Varianz σ zurück. Das Modell wird über über acht Parameter eingestellt, die jeweils auf eine Unsicherheitsquelle wirken. Die Parametergruppe α_1 stellt die Abhängigkeit der gestörten Größen von der gemessenen translatorischen Geschwindigkeit des Roboters unabhängig jeweils für die Winkelgeschwindigkeit, translatorische Geschwindigkeit in x- und y-Richtung sowie eine Störung des sich einstellenden Winkels mit dem ortsfesten Referenzkoordinatensystem dar. Die Parametergruppe α_2 stellt die Abhängigkeit der gestörten Größen von der gemessenen Winkelgeschwindigkeit dar. Für die Anwendung können die Parameter des Bewegungsmodells aus Messdaten ermittelt werden. Die praktische Erfahrung zeigt jedoch, dass die Leistungsfähigkeit des Partikelfilters steigt, wenn die Unsicherheiten leicht überschätzt werden. Der ermittelte Geschwindigkeitsvektor $[{}_{\mathrm{r}}\hat{v}_x, {}_{\mathrm{r}}\hat{v}_y]^{\mathrm{T}}$ wird aus dem Roboterkoordinatensystem ins ortsfeste Koordinatensystem transformiert. Dort wird über den Momentanpol der Zustandsübergang von $\boldsymbol{x}_{k-1}$ nach $\boldsymbol{x}_k$ berechnet. Die Wirkungsweise des Bewegungsmodells ist in Bild 6.2 dargestellt. Ausgehend von einer bekannten Position wird das Bewegungsmodell wiederholt auf die Partikelwolke $\boldsymbol{\chi}_k$ angewendet. Zu erkennen ist, wie sich der Schwerpunkt der Partikelwolke gemäß den Stellgrößen verlagert, während die Unsicherheit über den Zustand, die durch die Streuung der Partikel und damit durch den Durchmesser der Partikelwolke repräsentiert wird, stetig ansteigt.

Algorithmus 6.1 Das Bewegungsmodell für einen omnidirektionalen Roboter: $\boldsymbol{x}_k = [x, y, \phi]^{\mathrm{T}}$ kennzeichnet Lage und Orientierung des Roboters, $\boldsymbol{u}_k = [\dot{x}, \dot{y}, \dot{\omega}]^{\mathrm{T}}$ kennzeichnet die Stellgrößen, die auf den Roboter wirken. Die Unsicherheiten resultierend aus der translatorischen Geschwindigkeit werden durch die Parameter $\alpha_{1,(\cdot)}$ berücksichtigt. Die Unsicherheiten resultierend aus der Winkelgeschwindigkeit werden durch die Parameter $\alpha_{2,(\cdot)}$ berücksichtigt.

1: SAMPLEMOTIONMODELL($p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{u}_k)$)
2: $\quad {}_{\mathrm{r}}\hat{v}_x = \text{SAMPLEGAUSSIAN}(\dot{x}, \alpha_{1,x} \cdot |v| + \alpha_{2,x} \cdot |\omega|)$
3: $\quad {}_{\mathrm{r}}\hat{v}_y = \text{SAMPLEGAUSSIAN}(\dot{y}, \alpha_{1,y} \cdot |v| + \alpha_{2,y} \cdot |\omega|)$
4: $\quad \hat{\omega} = \text{SAMPLEGAUSSIAN}(\omega, \alpha_{1,\omega} \cdot |v| + \alpha_{2,\omega} \cdot |\omega|)$
5: $\quad \hat{\gamma} = \text{SAMPLEGAUSSIAN}(0, \alpha_{1,\gamma} \cdot |v| + \alpha_{2,\gamma} \cdot |\omega|)$
6: $\quad [{}_{\mathrm{w}}\hat{v}_x, {}_{\mathrm{w}}\hat{v}_y]^{\mathrm{T}} = \text{TRANSFORMTOWORLDCOORDINATES}([{}_{\mathrm{r}}\hat{v}_x, {}_{\mathrm{r}}\hat{v}_y]^{\mathrm{T}}, \boldsymbol{x}_{k-1})$
7: $\quad \boldsymbol{r}_1 = \left[-\frac{{}_{\mathrm{w}}\hat{v}_y}{\hat{\omega}}, \frac{{}_{\mathrm{w}}\hat{v}_x}{\hat{\omega}}\right]^{\mathrm{T}}$ ▷ Vektor $\boldsymbol{r}_1$ von $\boldsymbol{x}_{k-1}$ zum Momentanpol
8: $\quad \boldsymbol{r}_2 = -\boldsymbol{r}_1 \cdot \begin{bmatrix} \cos(\hat{\omega}T) & -\sin(\hat{\omega}T) \\ \sin(\hat{\omega}T) & \cos(\hat{\omega}T) \end{bmatrix}$ ▷ Vektor $\boldsymbol{r}_2$ vom Momentanpol zu $\boldsymbol{x}_k$
9: $\quad [x_k, y_k]^{\mathrm{T}} = [x_{k-1}, y_{k-1}]^{\mathrm{T}} + \boldsymbol{r}_1 + \boldsymbol{r}_2$ ▷ Bestimmung der Endposition
10: $\quad \phi_k = \phi_{k-1} + (\hat{\omega} + \hat{\gamma}) \cdot T$
11: $\quad$ **return** $\boldsymbol{x}_k$
12: **end**

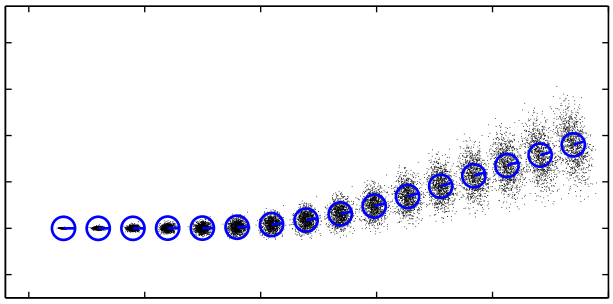


Bild 6.2: Das Bild zeigt die wiederholte Anwendung des Bewegungsmodells gemäß den Stellgrößen auf eine Partikelwolke ohne Berücksichtigung externer Messungen. Der Schwerpunkt der Partikelwolke bewegt sich gemäß den Stellgrößen, die Unsicherheit, gekennzeichnet durch die Größe der Wolke, steigt jedoch immer weiter an.

6.2.2 Raycasting

Neben dem Bewegungsmodell ist für die Anwendung eines Partikelfilters zur Positionsschätzung die Generierung von Erwartungswerten für Messungen ausgehend vom durch das Partikel repräsentierten Zustand erforderlich. Anhand der Erwartungswerte wird die Plausibilität der Partikelposition im Abgleich mit der realen Messung bewertet. Bei der Lokalisation des Roboters innerhalb einer Karte unter Einsatz von Laserentfernungsmesssystemen erfolgt die Nachbildung des realen Messsystems durch *Raycasting*.

Allgemein werden beim *Raycasting* werden Lichtstrahlen in einer Szene – einem virtuellen Aufbau von Objekten – verfolgt, um beispielsweise die Beleuchtungssituation nachzubilden. Dazu werden virtuelle Lichtstrahlen vom Standpunkt des Beobachters ausgesandt und bis zu ihrem Kontakt mit einem Objekt verfolgt.

Im hier vorliegenden Fall werden Laserstrahlen vom Standpunkt des Laserentfernungsmesssystems bis zum Auftreffen auf ein belegtes Pixel der Karte verfolgt, das *Raycasting* ist auf eine Ebene beschränkt, die Länge des Strahls entspricht dem Messwert des virtuellen Sensors an der Stelle.

Bei der Monte-Carlo-Lokalisation wird der Aufenthaltsort durch eine große Anzahl von Partikeln (Hypothesen) repräsentiert. Die Zahl der erforderlichen Partikel hängt u. a. von der Größe des Arbeitsraums ab, jedoch sind Partikelzahlen von $M \geq 1000$ üblich. Für jedes einzelne Partikel wird die Plausibilität der Sensormessung anhand von mehreren Sensorstrahlen bestimmt, wobei die virtuellen Messwerte anhand von *Raycasting* gebildet werden. Da die Zahl der benötigten virtuellen Messwerte sehr groß ist, ist eine besonders effiziente Implementierung des *Raycasting*-Verfahrens erforderlich.

Um den Vorgang des *Raycasting* veranschaulichen zu können, werden in Bild 6.3 zunächst die verschiedenen Koordinatensysteme definiert: Die Partikel repräsentieren den Roboter im Weltkoordinatensystem ${}_{\mathrm{w}}(\mathrm{KS})$. Das *Raycasting* erfolgt im Kartenkoordinatensystem ${}_{\mathrm{m}}(\mathrm{KS})$, dessen Ursprung in der linken unteren Ecke der Karte liegt. Üblicherweise werden Karten, die vom Format einer Pixelgrafik entsprechen beginnend von der oberen linken Ecke zeilenweise abgespeichert, so dass sich ein weiteres Koordinatensystem ${}_{\mathrm{p}}(\mathrm{KS})$ ergibt. Die Position des Roboters wird in Weltkoordinaten bezüglich des Schwerpunktes ${}_{\mathrm{w}}\boldsymbol{r}_{\mathrm{r}}$ definiert, der den Ursprung des Koordinatensystems ${}_{\mathrm{r}}(\mathrm{KS})$ darstellt. Die auf dem Roboter montierten Sensoren messen relativ zu den Sensorkoordinatensystemen ${}_{\mathrm{s}}(\mathrm{KS})$.

Unter Anwendung der ebenen Transformationen zur Positionierung des Sensors in der Karte erfolgt das *Raycasting* im Kartenkoordinatensystem. Die Lösung orientiert sich dabei an Anwendungen aus der Computergrafik: Der Vorgang der Strahlverfolgung in einem Raster hat große Ähnlichkeit mit dem Vorgang des Zeich-

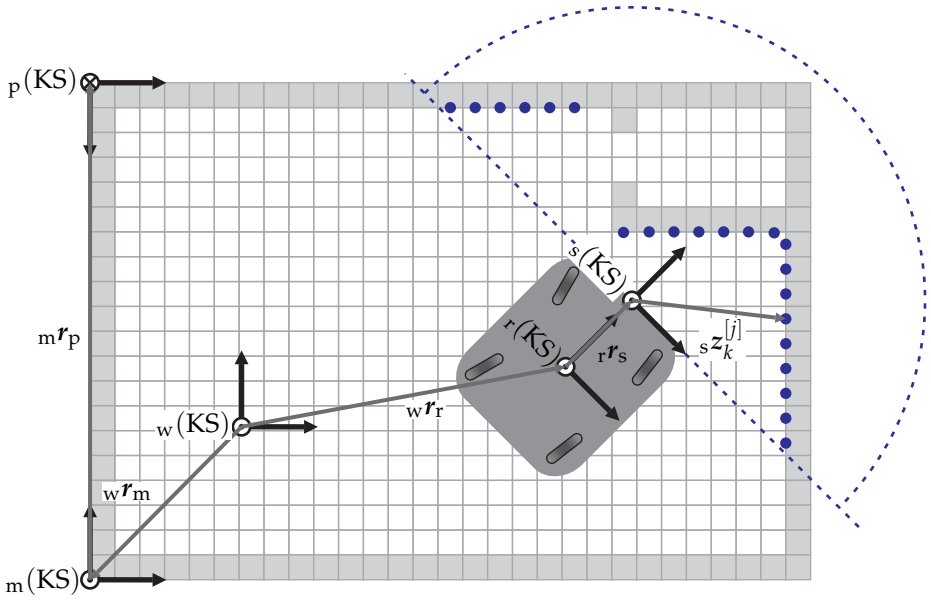


Bild 6.3: Bei der Errechnung virtueller Messwerte ist der Sensorursprung mittels der dargestellten Transformationen in der Karte zu platzieren.

nens einer Linie. Eine effiziente Lösung für dieses Problem kommt von Bresenham [1965]. Der nach ihm benannte Algorithmus wurde mit dem Ziel entwickelt, Vektoren in optisch ansprechend gerasterte Linien zu konvertieren, und beschränkt sich aus diesem Grund auf die Bestimmung aller für die Darstellung erforderlichen Rasterelemente. Der Vorteil liegt in der Arbeit mit Ganzzahlen, deren Berechnung auch auf modernen Rechnern noch immer effizienter erfolgt als die Berechnung mit Gleitkommazahlen. BRESENHAMS Ansatz war die Betrachtung von Geraden mit einer Steigung $0 \leq m_{\text{Linie}} < 1$, da in diesem Fall aufgrund der Definition der Geradensteigung die Änderung in der y-Koordinate zwischen zwei benachbarten Pixeln weniger als ein Pixel beträgt. Anhand von Bild 6.4a soll das Verfahren kurz skizziert werden. Die Werte für einen Durchlauf sind über der Skizze in Bild 6.4a aufgetragen. Beginnend bei Punkt P_0 wird eine Linie mit der Steigung $m_{\text{Linie}} = \frac{4}{7}$ gezeichnet. Jedesmal wenn x um Eins inkrementiert wird, wird die exakte y-Koordinate um m_{Linie} inkrementiert. y wird immer dann erhöht, wenn der absolute Fehler $\Delta y = |y_{exact} - y|$ über 0.5 steigt. Da der Algorithmus nicht automatisch alle überstrichenen Linien kennzeichnet, ist er unverändert nicht auf die Anwendung des *Raycastings* übertragbar. Allerdings lässt sich auf den grundlegenden Überlegungen, auf denen die Effizienz dieses Verfahrens beruht, aufbauen. Zentraler Ansatz ist es, den vom Strahl zurückzulegenden Weg zu dem jeweils nächsten Rasterelement in

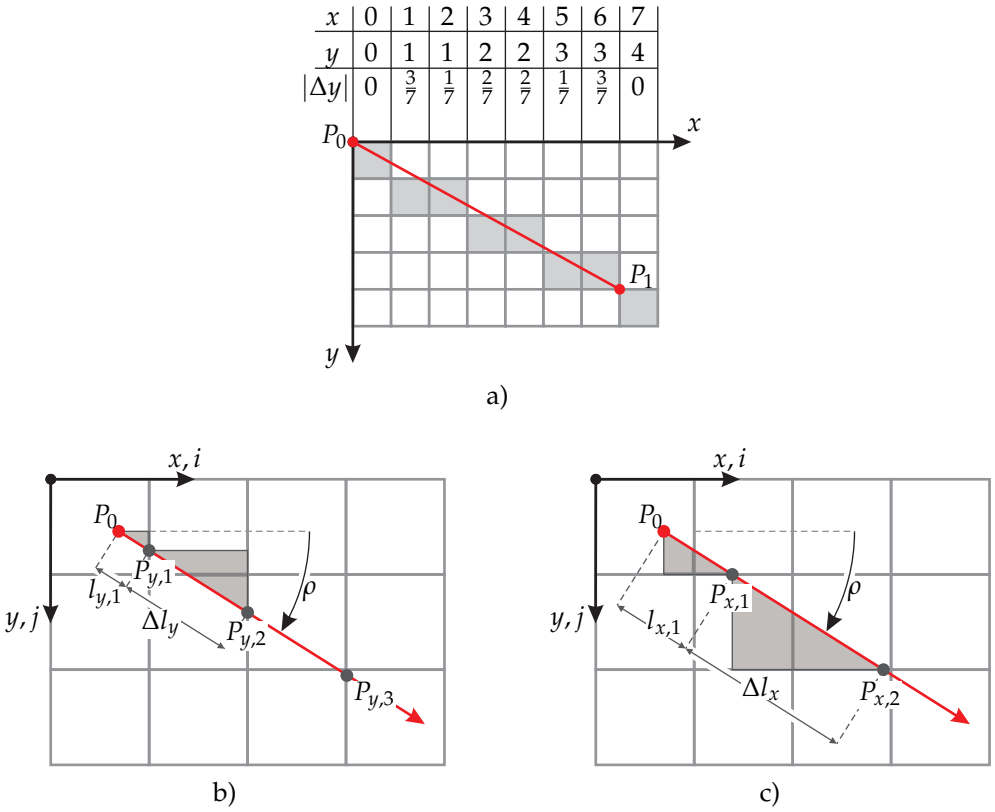


Bild 6.4: Das effiziente Zeichnen einer Linie in ein Raster basiert auf dem BRESENHAM'schen Algorithmus: In a) ist skizziert, wie mittels elementaren Rechenoperationen eine Linie in ein Raster gezeichnet werden kann; b) zeigt die Berechnung von Schnittpunkten einer Linie mit dem vertikalen Raster; c) zeigt die Berechnung der Schnittpunkte mit dem horizontalen Raster.

beiden Achsrichtungen zu vergleichen. Das als nächstes vom Strahl durchquerte Element ist stets das dichter gelegene. Wird die Entfernung in beiden Richtungen in getrennten Variablen verwaltet, dann ist bei jedem Wechsel in ein neues Rasterelement nur die Entfernung zum nächsten Element der betreffenden Richtung anzupassen. In einem regelmäßigen Raster ist der Entfernungszuwachs dabei, mit Ausnahme des ersten Elements, konstant. Die Verfolgung des Strahls kommt auf diese Weise nur mit Additions- und Vergleichsoperationen aus.

In Bild 6.4 ist die Bestimmung der Entfernung zwischen P_0 und den Kanten des Kartenrasters dargestellt. Der erste Schnitt des Strahls mit dem Raster erfolgt bei $P_{y,1}$ mit einer vertikalen Rasterlinie. Variablen, die sich auf Schnitte mit den vertikalen Linien des Rasters beziehen, werden mit y indiziert, während solche, die mit den horizontalen Achsen schneiden, durch x gekennzeichnet werden. Da die Zellen des Rasters durch Ganzzahlen adressiert werden, wird die floor-Funktion

genutzt um Fließkommazahlen abzurunden. Um den Abstand zum Schnittpunkt $P_{y,1}$ zu bestimmen, wird durch

$$i_0 = \text{floor}\left(\frac{{}_{\text{m}}x_{P0}}{d_{\text{Raster}}}\right) \tag{6.6}$$

$$j_0 = \text{floor}\left(\frac{{}_{\text{m}}y_{P0}}{d_{\text{Raster}}}\right) \tag{6.7}$$

zunächst die Ausgangsposition in Pixelkoordinaten berechnet. Mit Hilfe der Pixelkoordinaten lässt sich der Abstand $l_{y,1}$ zwischen P_0 und $P_{y,1}$ zu

$$l_{y,1} = \begin{cases} ((i_0+1)\cdot d_{\text{Raster}} - {}_{\text{m}}x_{P0})\cdot\frac{1}{\cos\rho} & \text{falls } \cos\rho > 0 \\ (i_0\cdot d_{\text{Raster}} - {}_{\text{m}}x_{P0})\cdot\frac{1}{\cos\rho} & \text{falls } \cos\rho < 0 \\ 0 & \text{sonst} \end{cases} \tag{6.8}$$

bestimmen. Durch die konstante Rasterweite gilt für alle folgenden Schnittpunkte mit der y-Achse – vorausgesetzt Strahl und y-Achse verlaufen nicht parallel – ein konstanter Abstand Δl_y

$$\Delta l_y = \begin{cases} \left|\frac{d_{\text{Raster}}}{\cos\rho}\right| & \text{falls } \cos\rho \neq 0 \\ \infty & \text{sonst} \end{cases} . \tag{6.9}$$

Die Berechnung zur Bestimmung der Distanzen zu Schnittpunkten mit der x-Achse erfolgt analog

$$l_{x,1} = \begin{cases} ((j_0+1)\cdot d_{\text{Raster}} - {}_{\text{m}}y_{P0})\cdot\frac{1}{\sin\rho} & \text{falls } \sin\rho > 0 \\ (j_0\cdot d_{\text{Raster}} - {}_{\text{m}}y_{P0})\cdot\frac{1}{\sin\rho} & \text{falls } \sin\rho < 0 \\ 0 & \text{sonst} \end{cases} , \tag{6.10}$$

$$\Delta l_x = \begin{cases} \left|\frac{d_{\text{Raster}}}{\sin\rho}\right| & \text{falls } \sin\rho \neq 0 \\ \infty & \text{sonst} \end{cases} . \tag{6.11}$$

Die Längenberechnung erfolgt durch eine iterative Addition der konstanten Längen Δl_x und Δl_y bis zum Auftreffen auf eine belegte Zelle oder bis zum Rand der Rasterkarte

$$l_{x,k} = l_{x,k-1} + \Delta l_x \quad \text{falls } l_{x,k} \leq l_{y,k} \tag{6.12}$$

$$l_{y,k} = l_{y,k-1} + \Delta l_y \quad \text{falls } l_{y,k} \leq l_{x,k} \tag{6.13}$$

wobei in jeder Iteration die Koordinate des nächsten zu prüfenden Pixels bestimmt wird. Die Entscheidung darüber, ob die i oder j-Koordinate inkrementiert wird, erfolgt durch Vergleich der Längen $l_{x,k}$ und $l_{y,k}$: Für die jeweils kürzere Länge ist der nächste Schnitt mit einer Rasterlinie zu erwarten.

Durch die iterative Addition werden sich zwar Fehler in der Bestimmung von Δl_x und Δl_y immer stärker auswirken. Die hohe Präzision der trigonometrischen Funktionen und die begrenzte Größe der Wahrscheinlichkeitskarte stellen jedoch in der Praxis die Effektivität des Algorithmus sicher. Zu beachten ist, dass auf eine explizite Bestimmung der Schnittpunktskoordinaten verzichtet wird. Zu berechnen ist nur die Addition der Längendifferenzen und die Pixelkoordinate für die jeweils folgende Iteration.

Der linearen Anordnung der Daten im Arbeitsspeicher wird durch den Übergang auf die Arbeit mit einer eindimensionalen Karte Rechnung getragen, schließlich müssen die trigonometrischen Funktionen für einen Strahl nur einmal berechnet werden, da diese über den Durchlauf einer Iteration konstant sind.

Es verbleibt ein Algorithmus, der nach Berechnung der Längendifferenzen bezüglich Winkel und Raster aus einfachen Vergleichen und Additionen besteht. Zusammenfassend erfolgt der *Raycasting*-Prozess nach dem in Algorithmus 6.2 dargestellten Verfahren.

Algorithmus 6.2 *Raycasting* zur Bestimmung der Entfernung d, die ein Strahl in einer Rasterkarte bis zum Auftreffen auf ein Hindernis zurücklegt. Die Funktion TRANSFORMTOMAPCOORDINATES transformiert die im Weltkoordinatensystem gegebene Sensorposition in das Kartenkoordinatensystem.

```
1:  RAYCASTFREEDISTANCE(x_k, M)
2:      _m[x, y]^T = TRANSFORMTOMAPCOORDINATES(x_k)
3:      _m[i, j]^T = floor(_m[x, y]^T / d_Raster)
4:      if cos ρ > 0 then
5:          l_y = ((i + 1) · d_Raster − _m x) / cos ρ;
6:      else if cos ρ < 0 then
7:          l_y = (i · d_Raster − _m x) / cos ρ;
8:      end if
9:      Δl_y = |d_Raster / cos ρ|;
10:     if sin ρ > 0 then
11:         l_x = ((j + 1) · d_Raster − _m y) / sin ρ;
12:     else if sin ρ < 0 then
13:         l_x = (j · d_Raster − _m y) / sin ρ;
14:     end if
15:     Δl_x = |d_Raster / sin ρ|;
16:     j = Bildhöhe − 1 − j;   dj = −sign(sin ρ) · Bildbreite;
17:     idx = i + j · Bildbreite;
18:     while M(idx) ist frei und d ≤ z_max do
19:         if l_y < l_x then
20:             d = l_y;
21:             idx = idx + sign(cos ρ);
22:             l_y = l_y + Δl_y;
23:         else
24:             d = l_x;
25:             idx = idx + dj;
26:             l_x = l_x + Δl_x;
27:         end if
28:     end while
29:     return d
30: end
```

6.2.3 Sensormodell

Neben dem Bewegungsmodell ist ein Sensormodell erforderlich, mit dem die Plausibilität eines Partikels berechnet werden kann. Gesucht ist also die Wahrscheinlichkeit

$$p(\boldsymbol{z}_k|\boldsymbol{x}_{0:k}) \quad , \tag{6.14}$$

wobei wiederum die Markov-Eigenschaft ausgenutzt werden kann, so dass gilt

$$p(\boldsymbol{z}_k|\boldsymbol{x}_k) = p(\boldsymbol{z}_k|\boldsymbol{x}_{0:k}) \quad . \tag{6.15}$$

Das Sensormodell hilft, für die verschiedenen Hypothesen $\boldsymbol{x}_{k,[m]}$ zugehörige Messwerte $\boldsymbol{z}_k$ zu erzeugen, mit deren Hilfe die Plausibilität eines Partikels überprüft werden kann. Die Modellierung des Sensors basiert in Teilen auf Fox u. a. [1999]. In der Messung $p(\boldsymbol{z}_k)$ sind J Messungen enthalten, die jeweils mit unterschiedlichem Winkel vom Sensorursprung ausgehend die Entfernung zum nächstliegenden Objekt beinhalten. Es wird davon ausgegangen, dass die Elemente von $p(\boldsymbol{z}_k)$ statistisch unabhängig sind. Damit lässt sich das Problem faktorisieren:

$$p(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{M}) = \prod_{j=0}^{J} p(\boldsymbol{z}_k^{[j]}|\boldsymbol{x}_k, \boldsymbol{M}) \quad . \tag{6.16}$$

Ermittelt wird im Folgenden die Wahrscheinlichkeit $p(\boldsymbol{z}_k^{[j]}|\boldsymbol{x}_k, \boldsymbol{M})$ für einen einzelnen Messstrahl $\boldsymbol{z}_k^{[j]}$, wobei sich die Wahrscheinlichkeit für den gesamten Messvektor $\boldsymbol{z}_k$ nach Gl. (6.16) ergibt.

Aus dem *Raycasting*-Prozess (vergleiche Abschnitt 6.2.2) wird zunächst der Erwartungswert $\boldsymbol{z}_k^{[j]*}$ für den hypothetischen Zustand $\boldsymbol{x}_k$ in der Karte $\boldsymbol{M}$ ermittelt. Die zugehörige Messung $\boldsymbol{z}_k^{[j]}$ wird dann anhand der aus dem Erwartungswert bestimmten Wahrscheinlichkeitsfunktion gewichtet. Das Modell setzt sich aus verschiedenen Bestandteilen zusammen: Zunächst ist davon auszugehen, dass eine Messung gestört ist. Die Streuung um den tatsächlichen Messwert stellt eine Eigenschaft des Sensors dar. Für die hier eingesetzten Laserentfernungsmesssysteme mit begrenzter Reichweite wird von einer Normalverteilung ausgegangen. Die Messwahrscheinlichkeit wird mit p_{hit} bezeichnet. Sie lautet

$$p_{\text{hit}} = \begin{cases} \eta \cdot \mathcal{N}(\boldsymbol{z}_k^{[j]*}, \sigma_{\text{hit}}) & \text{falls} \quad 0 \leq \boldsymbol{z}_k^{[j]} \leq z_{\max} \\ 0 & \text{sonst} \end{cases} \quad . \tag{6.17}$$

Hierin bezeichnet η eine Normalisierungskonstante und $\mathcal{N}(z_k^{[j]*}, \sigma_{\text{hit}})$ eine Normalverteilung mit dem Erwartungswert $z_k^{[j]*}$ und der Standardabweichung σ_{hit}:

$$\mathcal{N}(z_k^{[j]*}, \sigma_{\text{hit}}) = \frac{1}{\sigma_{\text{hit}} \cdot \sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{z_k^{[j]} - z_k^{[j]*}}{\sigma_{\text{hit}}}\right)^2} \quad . \tag{6.18}$$

Die Wahrscheinlichkeit p_{hit} ist in Bild 6.5a dargestellt.

Zufällige Hindernisse sind Objekte, die nicht in der Karte des Roboters vorhanden sind und damit Messungen unerwartet verfälschen. Mit zunehmender Entfernung eines solchen Objekts vom Roboter steigt die Wahrscheinlichkeit, dass die Messung vorzeitig durch andere Effekte, beispielsweise ein weiteres zufälliges Hindernis, unterbrochen wird. Die Wahrscheinlichkeit, dass eine Messung durch zufällige Hindernisse gestört ist, sinkt daher mit zunehmendem Messwert. Sie wird aus diesem Grund durch die Exponentialverteilung p_{short} beschrieben:

$$p_{\text{short}} = \begin{cases} \eta \cdot \lambda_{\text{short}} \cdot e^{-\lambda_{\text{short}} \cdot z_k^{[j]*}} & \text{falls} \quad 0 \leq z_k^{[j]} \leq z_k^{[j]*} \\ 0 & \text{sonst} \end{cases} \quad . \tag{6.19}$$

Die Konstante λ_{short} legt die Form der Exponentialverteilung fest, während der Faktor η wieder der Normalisierung dient. Die Wahrscheinlichkeit p_{short} ist in Bild 6.5b dargestellt.

Fehler während des Messvorgangs resultieren mit erhöhter Häufigkeit im maximalen Messwert z_{max}. Beispielsweise dann, wenn Laserstrahlen von Hindernissen absorbiert oder reflektiert werden und nicht zur Messeinrichtung zurückkehren oder wenn mehrfache Reflektionen entstehen. Dem Einfluss dieser sogenannten *Kantenschüsse* wird durch die in Bild 6.5c dargestellte Wahrscheinlichkeit p_{max} Rechnung getragen:

$$p_{\text{max}} = \begin{cases} 1 & \text{falls} \quad z_k^{[j]} \geq z_{\text{max}} \\ 0 & \text{sonst} \end{cases} \quad . \tag{6.20}$$

Darüber hinaus treten Verfälschungen der Messwerte auf, die durch keinen der zuvor genannten Effekte zu erklären sind. Vereinfachend wird angenommen, dass solche Zufallsmessungen gleichverteilt über den gesamten Wertebereich der Messgröße auftreten. Die Wahrscheinlichkeit p_{rand} ist in Bild 6.5d dargestellt:

$$p_{\text{rand}} = \begin{cases} {z_{\text{max}}}^{-1} & \text{falls} \quad 0 \leq z_k^{[j]} \leq z_{\text{max}} \\ 0 & \text{sonst} \end{cases} \quad . \tag{6.21}$$

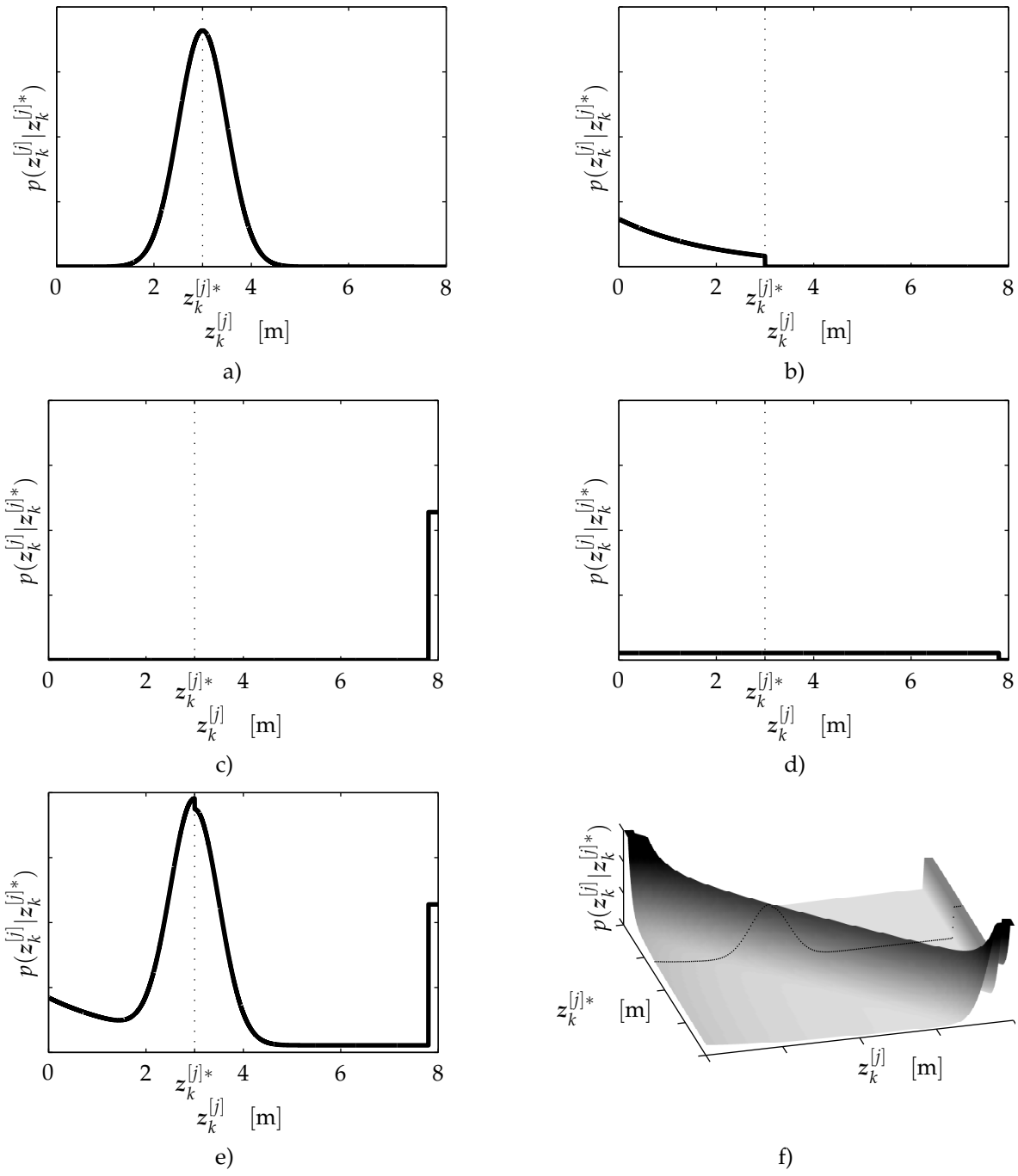


Bild 6.5: Das Sensormodell liefert die Wahrscheinlichkeitsfunktion $p(z_k^{[j]}|z_k^{[j]*})$ für eine Messung bei gegebenem Erwartungswert $z_k^{[j]*}$. Die Wahrscheinlichkeit wird aus den Bestandteilen a) p_{hit} (Messrauschen), b) p_{short} (nicht modellierte Hindernisse), c) p_{max} (Fehlmessungen, bei denen ein Echo ausbleibt) und d) p_{rand} (zufällige Fehlmessungen) zusammengesetzt. e) zeigt die gewichtete Überlagerung für den Erwartungswert $z_k^{[j]*}$, f) stellt die vollständige Wahrscheinlichkeitsfunktion $p(z_k^{[j]}|z_k^{[j]*})$ dar.

Das Sensormodell ergibt sich durch eine gewichtete Kombination dieser vier Wahrscheinlichkeitsverteilungen. Die in Gl. (6.22) enthaltenen Größen sind mit Ausnahme von $z_k^{[j]}$ und $z_k^{[j]*}$ Konstanten, die aus den Eigenschaften des Laserscanners resultieren. Ein beispielhafter Werteverlauf des Sensormodells für einen spezifischen Erwartungswert $z_k^{[j]*}$ ist in Bild 6.5e dargestellt. Bild 6.5f zeigt die vollständige Wahrscheinlichkeitsfunktion $p(z_k^{[j]}|z_k^{[j]*})$ in Abhängigkeit von $z_k^{[j]}$ und $z_k^{[j]*}$:

$$\begin{aligned} p(z_k^{[j]}|\boldsymbol{x}_k) = {} & k_{\text{hit}} \cdot p_{\text{hit}}(z_k^{[j]}|\boldsymbol{x}_k) \\ & + k_{\text{short}} \cdot p_{\text{short}}(z_k^{[j]}|\boldsymbol{x}_k) \\ & + k_{\text{max}} \cdot p_{\text{max}}(z_k^{[j]}|\boldsymbol{x}_k) \\ & + k_{\text{rand}} \cdot p_{\text{rand}}(z_k^{[j]}|\boldsymbol{x}_k) \end{aligned} \tag{6.22}$$

$$\text{mit} \qquad k_{\text{hit}} + k_{\text{short}} + k_{\text{max}} + k_{\text{rand}} = 1 \quad .$$

Zusammenfassend ist das Sensormodell in Algorithmus 6.3 dargestellt.

Algorithmus 6.3 Das Sensormodell beinhaltet die Generierung virtueller Messwerte durch *Raycasting* und die Überlagerung der Bestandteile p_{hit}, p_{short}, p_{max} und p_{rand}.

```
1: MeasurementModel(z_k, x_k, M)
2:     q = 1
3:     for j = 1 to J do
4:         z_k^[j]* = RaycastFreeDistance(x_k, M)
5:         p = k_hit · p_hit(z_k^[j] | z_k^[j]*, x_k) + k_short · p_short(z_k^[j] | z_k^[j]*, x_k) +
               k_max · p_max(z_k^[j] | z_k^[j]*, x_k) + k_rand · p_rand(z_k^[j] | z_k^[j]*, x_k)
6:         q = q · p
7:     end for
8:     return q
9: end
```

6.2.4 Partikelfilter

Wie in Algorithmus 6.4 dargestellt, ist die MONTE-CARLO-Lokalisation eine Anwendung des diskreten BAYES'schen Filters. Als Eingang des Filters dienen die Partikelwolke aus dem vorherigen Zeitschritt $\boldsymbol{\chi}_{k-1}$, die aus der Odometrie gewonnene Information über den zurückgelegten Weg $\boldsymbol{u}_k$, die aktuelle Sensormessung z_k sowie die Umgebungskarte $\boldsymbol{M}$. In der ersten Schleife wird nun zunächst ein Bewegungsmodell auf die Partikel angewendet. Dieses Modell beschreibt den Zusammenhang

Algorithmus 6.4 Der MONTE-CARLO-Lokalisations-Algorithmus basiert auf der Repräsentation des aktuellen Zustands durch eine Partikelwolke χ_k und gewichtet die einzelnen Partikel x_k gemäß den Messungen z_k.

```
1:  MclAlgorithm(χ_{k-1}, u_k, z_k, M)
2:      χ̄_k = χ_k = ∅
3:      for m = 1 to M do
4:          x_{k,[m]} = SampleMotionModel(u_k, x_{k-1,[m]})
5:          w_{k,[m]} = MeasurementModel(z_k | x_{k,[m]}, M)
6:          χ̄_k = χ̄_k + ⟨x_{k,[m]}, w_{k,[m]}⟩
7:      end for
8:      for m = 1 to M do
9:          draw i with probability ~ w_{k,[i]}
10:         add x_{k,[i]} to χ_k
11:     end for
12:     return χ_k
13: end
```

zwischen dem Zustand der Hypothese $x_{k-1,[m]}$ zum Zeitpunkt $k-1$ und dem aktuellen Zustand $x_{k,[m]}$ unter Berücksichtigung des zurückgelegten Wegs. Nach Anwendung des Bewegungsmodells wird mit dem Messmodell unter Berücksichtigung der aktuellen Messung z_k anhand der Karte M die Plausibilität der Hypothese $x_{k,[m]}$ überprüft und das Partikel entsprechend gewichtet. Das Partikel wird dann der neuen Partikelwolke $\bar{\chi}_k$ hinzugefügt. Aus der gewichteten Partikelwolke $\bar{\chi}_k$ wird in der zweiten Schleife schließlich durch Resampling wieder einer Partikelwolke χ_k mit gleich gewichteten Partikeln gemacht (vergleiche dazu Abschnitt 3.4).

Unter der Annahme, dass sich die Partikel nach einigen Iterationen im Bereich des wahren Zustands konzentrieren, kann eine Aussage über die aktuelle Position aus der Partikelwolke durch eine Überlagerung der verschiedenen Hypothesen gebildet werden:

$$x = \frac{1}{M} \cdot \sum_{m=1}^{M} x_{k,[m]}^{[x]} \tag{6.23}$$

$$y = \frac{1}{M} \cdot \sum_{m=1}^{M} x_{k,[m]}^{[y]} \tag{6.24}$$

Die Ermittlung der Orientierung erfolgt über eine Betrachtung der Richtungsvekto-

ren:

$$x_\Sigma = \sum_{m=1}^{M} \cos\left(\boldsymbol{x}_{k,[m]}^{[\phi]}\right) \tag{6.25}$$

$$y_\Sigma = \sum_{m=1}^{M} \sin\left(\boldsymbol{x}_{k,[m]}^{[\phi]}\right) \tag{6.26}$$

Der resultierende Winkel ergibt sich dann zu

$$\phi_\mathrm{r} = \mathrm{atan2}(y_\Sigma, x_\Sigma) \quad . \tag{6.27}$$

Die Varianz

$$\sigma_x^2 = \frac{1}{M} \cdot \sum_{m=1}^{M} \left(\boldsymbol{x}_{k,[m]}^{[x]} - x\right)^2 \tag{6.28}$$

$$\sigma_y^2 = \frac{1}{M} \cdot \sum_{m=1}^{M} \left(\boldsymbol{x}_{k,[m]}^{[y]} - y\right)^2 \tag{6.29}$$

$$\sigma_{\phi_\mathrm{r}}^2 = \frac{1}{M} \cdot \sum_{m=1}^{M} \left(\boldsymbol{x}_{k,[m]}^{[\phi]} - \phi_\mathrm{r}\right)^2 \tag{6.30}$$

liefert schließlich einen Hinweis auf die Unsicherheit der aktuellen Zustandsschätzung.

6.2.5 Experimentelle Erprobung der Lokalisation

Der Vorgang der Schätzung ist in Bild 6.6 dargestellt. Eingezeichnet sind die Partikel als Punkte sowie die tatsächliche Roboterposition und die geschätzte Roboterposition, dadurch gekennzeichnet, dass an dieser die im Lokalisationsprozess ausgewerteten Messstrahlen der Laserentfernungsmessung angetragen sind. Hier sind einige Iterationen des Einschwingvorgangs abgebildet. Zu Beginn befindet sich die geschätzte Roboterposition ungefähr in der Mitte der Karte. Dieses resultiert aus der Gleichverteilung der Partikel für die initiale Schätzung, bei der die Partikel über den Zustandsraum verteilt sind. Über die ersten Iterationen erfolgt eine Cluster-Bildung bis schließlich ein Cluster übrig bleibt. Da die geschätzte Roboterposition aus dem Erwartungswert gewonnen wird, ist hier erst dann eine aussagekräftige Darstellung zu erkennen, wenn sich ein dominantes Cluster gebildet hat. Anhand der eingezeichneten Messstrahlen kann die Plausibilität der Schätzung eingeordnet werden. Während die globale Lokalisation in einer Karte der dargestellten Größe in etwa 2000 Partikel erfordert, ist die Verfolgung der Position bereits mit einer deutlich kleineren Anzahl von unter 1000 Partikeln möglich.

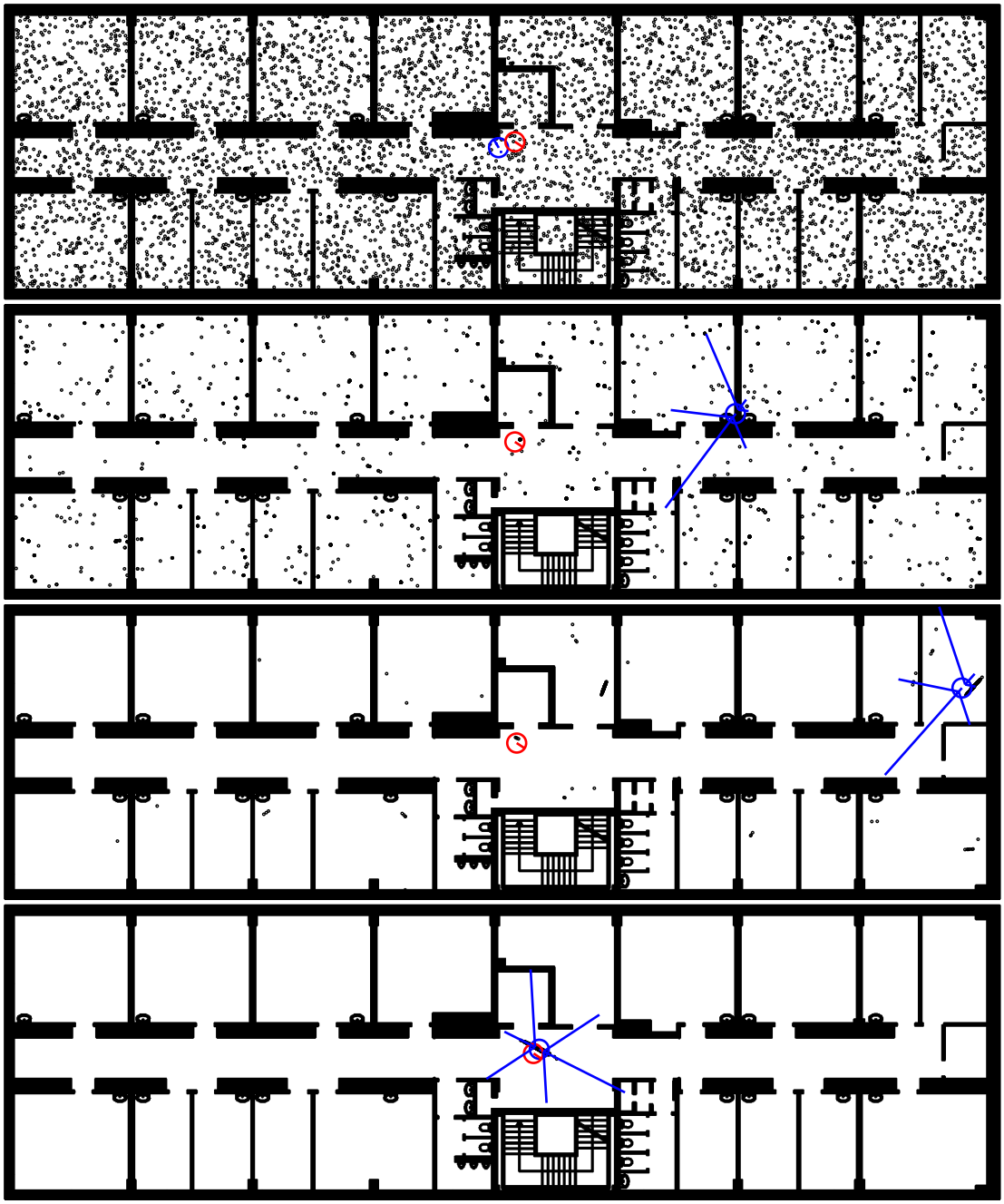


Bild 6.6: Der Einschwingvorgang der MONTE-CARLO-Lokalisation: Neben den Partikeln (dargestellt durch schwarze Punkte) sind die reale und die geschätzte Position des Roboters dargestellt. Die geschätzte Position ist jeweils durch die Darstellung der ausgewerteten Messstrahlen gekennzeichnet. Zunächst sind die Partikel gleichmäßig über den Zustandsraum verteilt (oben). Im iterativen Prozess erfolgt eine Cluster-Bildung (Mitte), bis sich die Partikel über der wahrscheinlichsten Position konzentrieren (unten).

6.2.6 Qualifizierung für den Einsatz in der Bahnregelung

Während die MONTE-CARLO-Lokalisation Vorteile bei der Robustheit und dem Einsatz in unsicheren Umgebungen zeigt, entsteht durch die große Partikelanzahl ein Rechenaufwand, der einen Einsatz in Regelkreisen mit schnellen Regeltakten verhindert. Zwar ist im vorliegenden Fall der Einsatz mit bis zu 5000 Partikeln im 50 ms Takt auf einem Prozessor mit 1,8 GHz Taktfrequenz möglich. Hier werden jedoch große Ressourcen gebunden, die dann für weitere Aufgaben des mobilen Systems nicht mehr zur Verfügung stehen. Obwohl der oben dargestellte Raycasting-Prozess sehr effizient umgesetzt ist, besteht der größte Rechenaufwand in der Generierung virtueller Messgrößen. Um die Prozessorlast zu reduzieren, wird ein Mehrratenansatz gewählt: Der alternierend zwischen Prädiktion und Korrektur schaltende Prozess wird so abgeändert, dass die Prädiktion mehrfach wiederholt wird, bis die Korrektur gestartet wird, um die steigende Unsicherheit zu begrenzen.

Um dem Korrekturprozess mehr als einen Takt Zeit zur Verfügung zu stellen, wird die Anwendung des Messmodells mit der Berechnung der Partikelgewichte in einen nebenläufigen Prozess ausgelagert, der vom Hauptprozess gestartet wird und dann parallel abläuft. Bei der nebenläufigen Bearbeitung arbeitet der Korrekturprozess der Kopie einer Partikelwolke aus dem Zeitpunkt des Anstoßens, während der Hauptprozess weiterhin das Bewegungsmodell auf die Partikel anwendet. Sind die Gewichte berechnet, werden die Informationen durch den Resampling-Prozess wieder zusammengeführt. Durch eine Indizierung der Partikel ist eine Zuordnung der Gewichte aus der Partikelwolke des Resamplingprozesses zu den bereits mehrfach aktualisierten Zuständen der Partikelwolke des Hauptprozesses möglich. Dieser Vorgang kann nur synchron mit dem Hauptprozess erfolgen. Analog zum oben beschriebenen Algorithmus wird die ressourcensparende Variante in Algorithmus 6.5 zusammengefasst. Die Folge der Auslassung von Korrekturschritten ist eine steigende Unsicherheit in der Zustandsschätzung. Dieser Effekt ist in Bild 6.7 dargestellt. In jeweils acht aufeinander folgenden Durchläufen wird das Prädiktionsmodell auf Grundlage der Odometriedaten angewendet. Der Schwerpunkt der Wolke verlagert sich und die Abmessungen der Wolke steigen an. Erfolgt die Einbeziehung von Messungen so wird die Unsicherheit reduziert. Die Steuerung der Messwertevaluation ist so zu gestalten, dass eine für nachgelagerte Regelkreise ausreichende Genauigkeit in der Zustandsschätzung erzielt wird.

Auf Basis der im Kapitel 5 erstellten Karte und unter Anwendung des Mehrratenansatzes liefert die Lokalisation bei 2000 Partikeln eine robuste Lageschätzung. Dabei erfolgt die Anwendung des Bewegungsmodells auf Basis des aus Messdaten geschätzten Roboterzustands im Takt von 50 ms. Die Einbeziehung der Messdaten aus den Laserentfernungsmesssystemen zur Gewichtung im nebenläufigen Korrek-

Algorithmus 6.5 Der Mehrraten MONTE-CARLO-Lokalisations-Algorithmus ist ressourcenschonend, indem er die Zahl der Evaluationen von Sensormessungen reduziert und in einen asynchronen, nebenläufigen Prozess auslagert. Die Prozesse (Tasks) laufen dabei gleichzeitig ab.

global $w_{[1...M]}$, $weightsAvailable = \text{false}$

Task: MULTIRATEMCLALGORITHM(χ_{k-1}, u_k, z_k, M)
  $\bar{\chi_k} = \chi_k = \emptyset$
  for $m = 1$ **to** M **do**
    $x_{k,[m]} = \text{SAMPLEMOTIONMODEL}(u_k, x_{k-1,[m]})$
    $\bar{\chi_k} = \bar{\chi_k} + \langle x_{k,[m]} \rangle$
  end for
  if Uncertainty is to high **then**
    $k_{\text{start}} = k$
    $weightsAvailable = \text{false}$
    start MULTIRATEEVALUATEMEASUREMENTS($\bar{\chi}_{k_{\text{start}}}, z_{k_{\text{start}}}, M$)
  end if
  if $weightsAvailable$ **then**
    for $m = 1$ **to** M **do**
      draw i with probability $\sim w_{k_{\text{start}},[i]}$
      add $x_{k,[i]}$ to χ_k
    end for
    return χ_k
  else
    return $\bar{\chi_k}$
  end if
end

Task: MULTIRATEEVALUATEMEASUREMENTS($\chi_{k_{\text{start}}}, z_{k_{\text{start}}}, M$)
  for $m = 1$ **to** M **do**
    $w_{k_{\text{start}},[m]} = \text{MEASUREMENTMODEL}(z_{k_{\text{start}}} | x_{k_{\text{start}},[m]}, M)$
  end for
  $weightsAvailable = \text{true}$
end

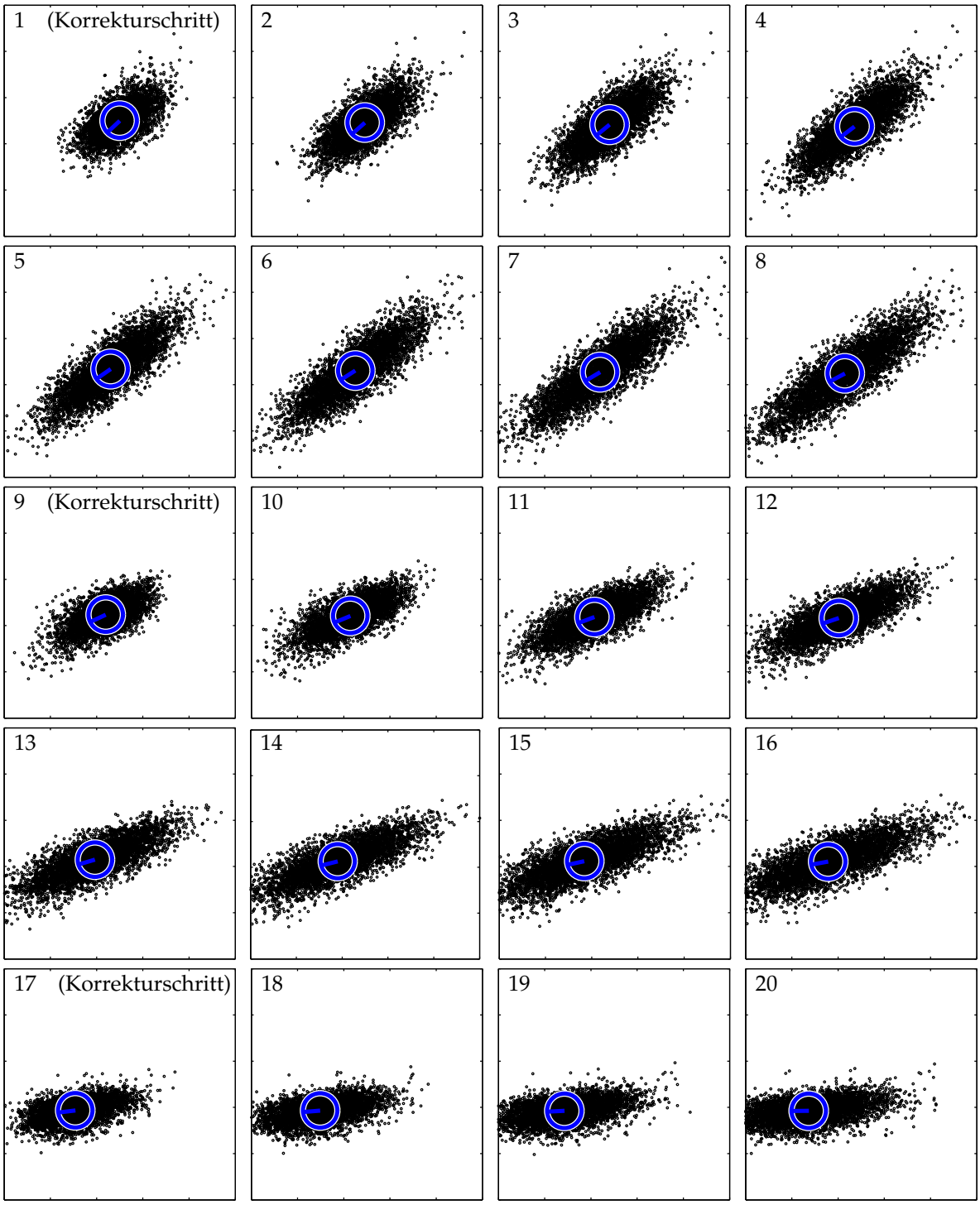


Bild 6.7: Dargestellt ist eine Partikelwolke im Lokalisationsprozess. Der Schwerpunkt stellt den Erwartungswert der Positionsschätzung dar. Durch den Mehrratenansatz ergibt sich eine steigende Unsicherheit bei wiederholter Anwendung des Bewegungsmodells, bis schließlich die Einbeziehung der Sensormessung zu einer Reduktion der Unsicherheit führt.

turprozess wird alle 400 ms angestoßen. Die Ausführungszeit des Bewegungsmodells beträgt bei durchschnittlicher Prozessorlast deutlich unter 10 ms, während der Korrekturprozess bis zu 30 ms in Anspruch nimmt. Aus diesen Ausführungszeiten lässt sich die Einsparung beim Rechenaufwand abschätzen: Statt eines Korrekturschritts von 30 ms Länge im 50 ms Takt wird dieser nur alle 400 ms ausgeführt.

Da in dieser Arbeit kein Referenzmesssystem für die Beurteilung der absoluten Messgüte zur Verfügung steht, kann eine Beurteilung der Leistungsfähigkeit der Lokalisation nur qualitativ erfolgen. Exemplarisch ist in Bild 6.8a das Abfahren einer Bahn dargestellt. Dieses erfolgt mittels der in Abschnitt 4.3 vorgestellten Bahnregelung auf Basis der hier vorgestellten Lageschätzung. Es ist zu erkennen, dass der Roboter der grau dargestellten Sollbahn mit geringen Abweichungen folgt. Bild 6.8b zeigt die durch Querversatz und Fehlorientierung repräsentierte Regelab-

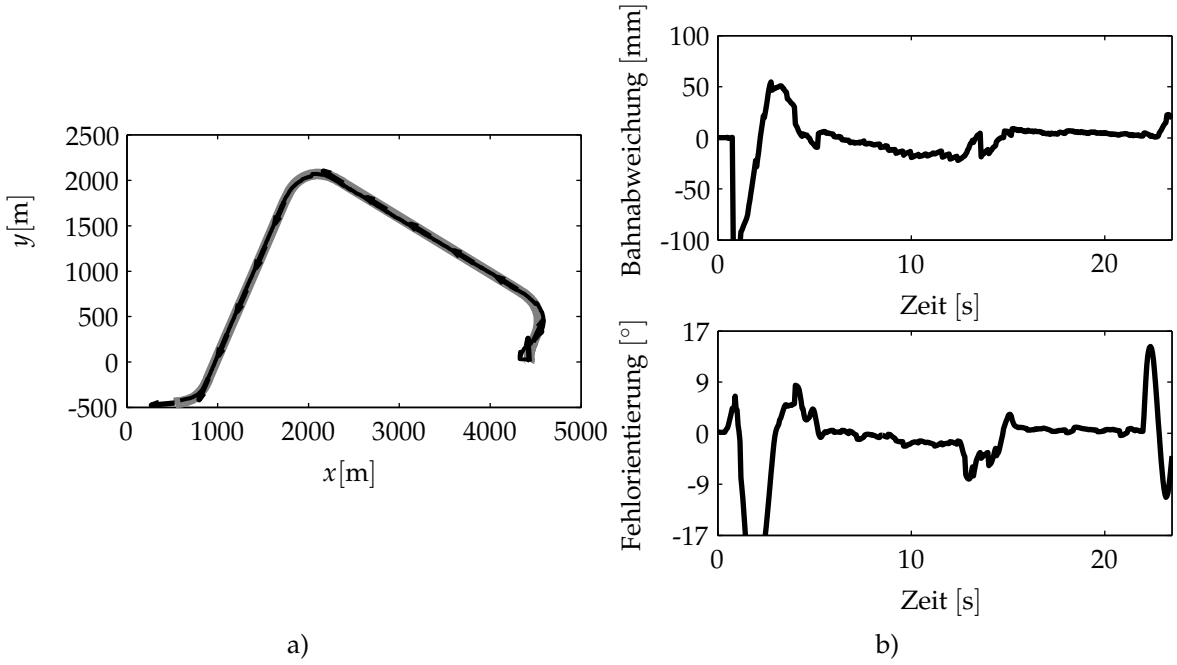


Bild 6.8: Dargestellt ist das geregelte Abfahren einer vorgegebenen Bahn; a) zeigt den Vergleich zwischen Soll- und Istbahn; b) zeigt durch Querversatz und Fehlorientierung die Regelabweichung über der Zeit.

weichung. Zu Beginn der Fahrt entsteht eine sprungförmige seitliche Abweichung durch das Einsetzen der Monte-Carlo-Lokalisation: Während des Stillstands wird die Zustandsaktualisierung ausgesetzt, da sonst in Folge der mangelnden Innovation in der Messung die Partikelvielfalt verarmt; Aus diesem Grund erfolgt die Auswertung der Messung erst nach Überschreiten eine Geschwindigkeitsschwelle,

mit dem geringfügige Sprünge in der Lageschätzung entstehen können. Im Verlauf der Fahrt erfolgt die Ausregelung des Regelfehlers und der Roboter folgt der vorgegebenen Bahn mit kleiner Regelabweichung. Erst zum Ende der Fahrt bevor der Roboter zum Stillstand kommt steigt der Regelfehler wieder geringfügig an. Dieses hängt mit dem Regelungskonzept zusammen: Der Stillstand des Roboters führt zum Verlust der Steuerbarkeit, so dass für den Übergang zum Stillstand die Regelung nicht mehr wirksam ist und ein Fehler entsteht.

Zusammenfassend lässt sich feststellen, dass die MONTE-CARLO-Lokalisation robuste Eigenschaften für die Lagebestimmung mitbringt. Die der Umgebung inhärente Unsicherheit wird durch den wahrscheinlichkeitsbasierten Ansatz berücksichtigt. Durch eine effiziente Implementierung kann der rechnerische Aufwand begrenzt werden. Der Mehrratenansatz qualifiziert die MONTE-CARLO-Lokalisation für den direkten Einsatz in schnellen Regelungen und bringt eine für die Anwendung in der Servicerobotik geeignete Genauigkeit mit.

7 Verfolgung dynamischer Objekte

Der Betrieb eines Roboters in dynamischer Umgebung, die durch sich bewegende Objekte mit einer Geschwindigkeit in der Größenordnung von der des Roboters geprägt ist, erfordert die Erfassung des Zustands dieser potentiellen sich bewegenden Kollisionspartner. Da das Verhalten der Objekte a priori unbekannt ist, muss eine Objektverfolgung zur Laufzeit unter Berücksichtigung von Echtzeitaspekten erfolgen.

7.1 Grundlagen der Objekt Verfolgung

Die zur Verfolgung von Objekten eingesetzte Sensorik deckt in der Regel einen mehrdimensionalen Messbereich ab und liefert eine in der Regel fehlerbehaftete Messung mit Informationen über die Umgebung. Der Fokus bei der Anwendung im Innenraum liegt auf der Suche nach Merkmalen bewegter Objekte (z. B. Beine) in den fächerförmig angeordneten Messungen des Abstands zum nächstliegenden reflektierenden Gegenstand aus einem Laserentfernungsmesssystem. Die ebenfalls große Zahl von Ansätzen auf Basis von Videokameras wird hier jedoch nicht diskutiert. Das Ziel der Objektverfolgung ist die Bestimmung der Objekttrajektorie. Im allgemeinen Fall steht keine Information über die Natur des Objekts zur Verfügung. Je nach Einsatzszenario können jedoch Annahmen getroffen werden, z. B. über die maximale Geschwindigkeit, oder – im Fall von Fahrzeugen – über die dynamischen Eigenschaften. Es besteht ein prinzipieller Unterschied zwischen der Verfolgung eines einzelnen Objekts und der gleichzeitigen Verfolgung mehrerer unabhängiger Objekte: Bei der Verfolgung mehrerer Objekte ist zunächst die Anzahl der Objekte zu bestimmen. Zudem besteht das Datenassoziationsproblem: Durch die Objekte werden Merkmale im Sensorsignal hervorgerufen. Es gilt dann, die einzelnen Merkmale unterschiedlichen Objekten zuzuordnen. Dabei treten Fehlmessungen auf, hervorgerufen durch Merkmale im Sensorsignal, die denen von Objekten ähnlich sind, die aber nicht von einem tatsächlich existierenden Objekt hervorgerufen wurden. Die Zahl der gemessenen Merkmale übersteigt dann die Zahl der vorhandenen Objekte. Solche Messungen werden als falsche Alarme bezeichnet. Darüber hinaus kann durch Verdeckungen oder fehlerhafte Messsignale ein vorhandenes Objekt nicht erkannt werden. Dadurch existieren weniger Messungen als Objekte. Schließlich können sich die zwei genannten Fälle überlagern. Das Datenassoziati-

onsproblem wird zusätzlich dadurch erschwert, dass in einer realen Umgebung bei begrenzter Sensorreichweite Objekte in den Messbereich eintreten können, so dass die Objektanzahl erhöht wird. Genauso können Objekte den Messbereich verlassen, was die Objektanzahl verringert.

Der am weitesten verbreitete Ansatz zur Suche nach Merkmalen ist die Suche nach Clustern im Messdatensatz, die als zusammenhängende lokale Minima erscheinen [Fod u. a., 2002; Kleinhagenbrock u. a., 2002; Scheutz u. a., 2004; Schulz u. a., 2003]. Die erkannten Merkmale können dynamischer und geometrischer Natur sein. Die Veränderung (Dynamik) in der Messung wird durch den Vergleich von zeitlich aufeinander folgenden Messungen erreicht. Bei Eigenbewegung des Roboters sind diese dafür zunächst zueinander auszurichten, beispielsweise durch ein Scan-Matching-Verfahren oder alternativ durch die Nutzung der Odometrie Information. Der Nachteil dabei ist, dass nur bewegte Objekte erkannt werden können. Bei Topp und Christensen [2005] wurde die Methode von Schulz u. a. [2003] auch auf statische Objekte übertragen, wobei sie gute Ergebnisse in typischen Einsatzszenarien zeigen konnten, während in bezogen auf das Scan-Bild unordentlichen Umgebungen Probleme auftraten. Cui u. a. [2005] verfolgen einen Multi-Sensor-Ansatz durch die Nutzung mehrerer Laserentfernungsmesssysteme und eine monokulare Kamera. Hier werden zur Beinerkennung Merkmale von 15 cm Durchmesser als potentielle Füße aufgefasst und Paare im Abstand von bis zu 50 cm als Schrittweite aufgefasst.

Neben der Merkmalsextraktion liegt die zweite Herausforderung bei der Verfolgung mehrerer Objekte in der bereits genannten Datenassoziation. Bei Kluge u. a. [2001] wird ein Algorithmus vorgestellt, der zwar das Verfolgungsproblem löst, der aber bei einer einzigen Fehlmessung oder Verdeckung versagt, da keine Modellannahme über das Objekt getroffen wird. Bei Montemerlo u. a. [2002b] und Lindström und Eklundh [2001] wird das Datenassoziationsproblem durch einen *Nearest Neighbour*-Ansatz gelöst. Eine verbreitete Methode zur Lösung des Datenassoziationsproblems ist die Schätzung der gemeinsamen Zuordnungswahrscheinlichkeit. Auf Bar-Shalom und Fortmann [1988] geht das *Joint Probabilistic Data Association Filter* (JPDAF) zurück (siehe auch [Cox, 1993]).

Zusammenfassend lassen sich die folgenden Problemstellungen formulieren: Zunächst sind aus den Messdaten Merkmale zu extrahieren, die in der Folge Objekten zugeordnet werden. Weiterhin ist die unbekannte Anzahl der Objekte zu bestimmen, wobei diese im Verlauf der Messungen variiert. Die Merkmale müssen den Objekten zugeordnet werden, dabei sind sowohl falsche Alarme als auch fehlende Merkmale zu berücksichtigen. Bei der Zustandsschätzung für die Objekte sind deren dynamische Eigenschaften in Form eines Bewegungsmodells zu berücksichtigen. Alle vorgenannten Punkte erfolgen unter den Unsicherheiten durch Messfeh-

ler, Modellfehler und fehlerhafte Modellparameter.

7.2 Objektverfolgung mittels individueller Partikelfilter

Die in den folgenden Abschnitten dargestellte wahrscheinlichkeitsbasierte Methode zur Objektverfolgung, arbeitet mit allgemein formulierten Merkmalen und beschränkt sich nicht auf eine bestimmte Klasse von Objekten. Der Ansatz ist offen für eine lernbasierte Merkmalsextraktion [Bennewitz u. a., 2005], die einen präziseren *Gating*-Vorgang[1] bei zweifelsfrei identifizierbaren Objekten ermöglicht. Damit lässt sich der Parameterbereich für die Generierung von Rasterkarten und die Anwendung des Bewegungsmodells für Objekte einschränken. Die Methode kann jedoch auch für eine unbekannte Umgebung angewendet werden, da sie eine inhärente Robustheit gegenüber Modellfehlern bietet. Verdeckungen können zu einem gewissen Grad toleriert werden. Fehlerhafte Datenassoziationen haben in der Folge keine fatalen Auswirkungen. Trotz der unbekannten bzw. nur grob bestimmbaren Modellparameter ist eine Bestimmung des Umgebungszustands, einschließlich einer Aussage über die Zuverlässigkeit, möglich. Dem hier dargestellten Vorgehen liegt der Bayes'sche Wahrscheinlichkeitsbegriff zugrunde. Ziel ist die Verfolgung einer a priori unbekannten Anzahl von Objekten mit Hilfe der auf einem mobilen Roboter mitgeführten Sensorik mittels Partikelfiltern in Anlehnung an Schulz u. a. [2003]. Es wird auf ein einfaches Bewegungsmodell zurückgegriffen, innerhalb dessen die Zustände Position und Geschwindigkeit bestimmt werden. Die Schätzung der Objektzustände erfolgt mittels Partikelfiltern. Die Objektverfolgung arbeitet mit einem Algorithmus zur Extraktion von gemessenen Merkmalen aus dem Sensorsignal. Der Schätzung der Objektanzahl liegt eine Wahrscheinlichkeitsverteilung für ihren zeitlichen Verlauf sowie für den Zusammenhang zwischen der Zahl von Merkmalen und Objekten zugrunde. Das Datenassoziationsproblem wird durch die Schätzung der gemeinsamen Wahrscheinlichkeit unterschiedlicher Zuordnungsfälle gelöst.

Die in Kapitel 4 vorgestellte Roboterplattform dient als Versuchsträger. Sie ist mit zwei einander gegenüber stehenden Laserentfernungsmesssystemen der Firma Sick ausgestattet. Diese sind in einer Höhe von ca. 40 cm angebracht, wodurch beispielsweise Personen nur aufgrund ihrer Beine in den Messungen erkannt werden können. Die zur Verfügung stehende Sensorik hat natürlich Einfluss auf den Prozessablauf und die Messqualität. Die Berücksichtigung einer konkreten Anordnung sorgt für die Praxisrelevanz des vorgestellten Ansatzes. In diesem Kapitel werden vorrangig Algorithmen und Phänomenologie beschrieben, sowie experimentelle Er-

[1] Gating bezeichnet das Eingrenzen des Messbereiches in einer Umgebung um den Erwartungswert.

gebnisse auf der realen Plattform dargestellt.

Um die folgenden Abschnitte dieses Kapitels einordnen zu können, wird zunächst der Prozessablauf skizziert. Elementar ist die Modellierung der dynamischen Objekte, deren Bewegungsmodell in Abschnitt 7.2.2 dargestellt ist. Die Zustände des n-ten Objekts $\boldsymbol{x}_{k,\langle n\rangle}$ werden durch den Vektor $[x, y, \phi, v]^{\mathrm{T}}$ repräsentiert. Dabei entsprechen x und y den Koordinaten im ortsfesten Koordinatensystem, ϕ der Orientierung der Bewegungsrichtung und v der Geschwindigkeit des Objekts. Die Repräsentation der Objektzustände erfolgt mittels Partikeln, wobei für jedes Objekt ein eigenes Partikelfilter eingesetzt wird.

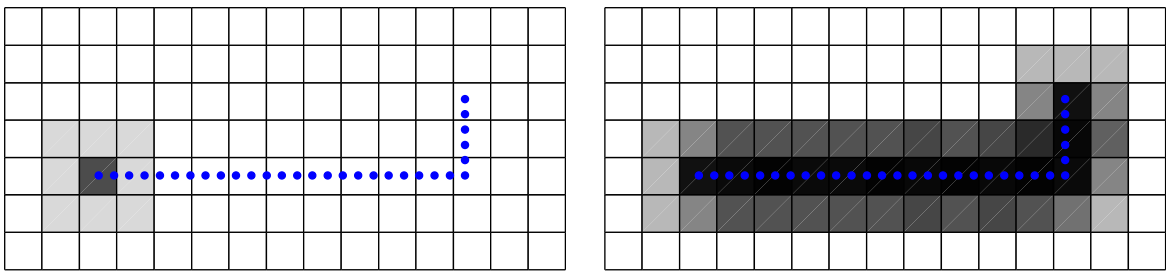
Der Prozess zur Schätzung der Zustände $\boldsymbol{x}_{k,\langle 1\rangle}, \ldots, \boldsymbol{x}_{k,\langle N\rangle}$ nimmt folgenden Verlauf. Die Daten der Laserentfernungsmesssysteme werden unter Verwendung der Roboterposition ${}_{\mathrm{w}}\boldsymbol{r}_{\mathrm{r}}$ als Punkte in das ortsfeste Koordinatensystem ${}_{\mathrm{w}}(\mathrm{KS})$ transformiert (vergleiche Bild 6.3). Aus den Daten der Laserentfernungsmesssysteme werden Wahrscheinlichkeitskarten generiert. Darunter sind eine Belegtheitskarte, eine dynamische Belegtheitskarte und eine Verdeckungskarte. Die Belegtheitskarte wird direkt aus den Messdaten erzeugt. Die Berechnung der dynamischen Belegtheitskarte erfolgt aus zwei aufeinander folgenden Messdatensätzen und kennzeichnet solche Zellen, bei denen sich bezüglich der Belegung eine Änderung ergeben hat. Die Verdeckungskarte beinhaltet den Schatten hinter dynamischen Objekten. Die Messdaten werden auf Muster untersucht, die auf Objekte innerhalb der Umgebung hindeuten. Aufgrund dieser Muster werden Merkmalskarten generiert. Die Anzahl der zu verfolgenden Objekte wird geschätzt. Die Zahl der aktiven Partikelfilter wird an die Objektanzahl angepasst. Auf die verbleibenden Filter wird das Bewegungsmodell angewendet (Prädiktionsschritt). Wenn mehr Objekte erkannt wurden als Filter aktiv sind, werden zusätzliche Partikelfilter initialisiert. Das Datenassoziationsproblem wird mittels eines *Joint Probabilistic Data Association Filter* (Datenzuordnungsfilter für gemeinsame Wahrscheinlichkeit) gelöst [Bar-Shalom und Fortmann, 1988], das von Schulz u. a. [2003] auf die Anwendung mit Partikelfiltern erweitert wurde. Die Vorhersage sowie die Merkmalskarten werden verwendet, um die Zuordnungswahrscheinlichkeiten $\beta_{l,n}$ der Merkmale $\{1 \ldots l \ldots L\}$ zu den Objekten $\{1 \ldots n \ldots N\}$ zu berechnen. Entsprechend dieser Zuordnungswahrscheinlichkeiten werden die Partikel gewichtet. Die Partikelgewichte werden im Korrekturschritt aktualisiert. Schließlich findet der Resampling-Schritt statt, um die gewichtete Partikelwolke in eine gleich gewichtete Partikelwolke umzuwandeln, in der die Wahrscheinlichkeitsdichtefunktion nur noch über die Partikeldichte repräsentiert wird. Abschließend werden die Objektzustände durch Bildung der Erwartungswerte über die Partikelwolken geschätzt, bevor der Zyklus erneut beginnt.

7.2.1 Sensormodell/Merkmalsextraktion

Das Sensormodell $p(\boldsymbol{z}_{\langle l \rangle}|\boldsymbol{x}_{\langle n \rangle})$, $l = 1 \ldots L$ soll aufgrund der Messdaten die Aussage ermöglichen, wie wahrscheinlich der Aufenthalt eines Objekts, das durch das Merkmal l repräsentiert wird, an der Position $\boldsymbol{x}_{\langle n \rangle}$ ist. Zu diesem Zweck werden in jedem Zeitschritt k verschiedene Rasterkarten erstellt. Zunächst wird eine Belegtheitskarte $p(\boldsymbol{M}_{\text{occ},k}|\boldsymbol{z}_k)$ aus dem aktuellen Messdatensatz $\boldsymbol{z}_k$ generiert, die die Umgebung des Roboters abdeckt. Zusätzlich wird für jedes im Datensatz gefundene Merkmal $\boldsymbol{z}_{\langle l \rangle}$ eine Merkmalskarte $p(\boldsymbol{M}_{\text{obj},k,\langle l \rangle}|\boldsymbol{z}_{k,\langle l \rangle})$ erstellt. Jede Zelle dieser Karte enthält die Wahrscheinlichkeit für die Belegtheit mit dem im Datensatz gefundenen Objekt l. Um statische und dynamische Hindernisse unterscheiden zu können, wird neben der statischen Belegtheitskarte auch die dynamische Belegtheitskarte $p(\boldsymbol{M}_{\text{dyn},k}|\boldsymbol{z}_k, \boldsymbol{z}_{k-1})$ erzeugt, die Veränderungen im Umfeld des Roboters beinhaltet. Durch Objekte im Nahbereich des Roboters werden Bereiche abgeschattet, die für den Sensor nicht einsehbar sind. In diesen abgeschatteten Bereichen kann kein Objekt gefunden werden. Für diesen Fall wird ein Ersatzmerkmal $p(\boldsymbol{z}_{\langle 0 \rangle}|\boldsymbol{x}_{\langle n \rangle})$ generiert, das eine Verdeckungskarte darstellt.

Erstellung der Belegtheitskarte

Bereits in Abschnitt 5.3.1 wurde ein Verfahren zur Erstellung von Belegtheitskarten vorgestellt. Für die hier vorliegende Aufgabenstellung wird jedoch ein effizienterer Ansatz verfolgt. Auf den Arbeiten von Moravec und Elfes [1985] basiert ein Verfahren, welches nur die Endpunkte einer Messung auswertet. Das Einfügen einer Messung in die Rasterkarte erzeugt in dem entsprechenden Kartenindex und in unmittelbarer Umgebung ein Wahrscheinlichkeitsdichteprofil, das der Wahrscheinlichkeitsverteilung der Messunsicherheit entspricht (siehe Bild 7.1). Weitere Scan-



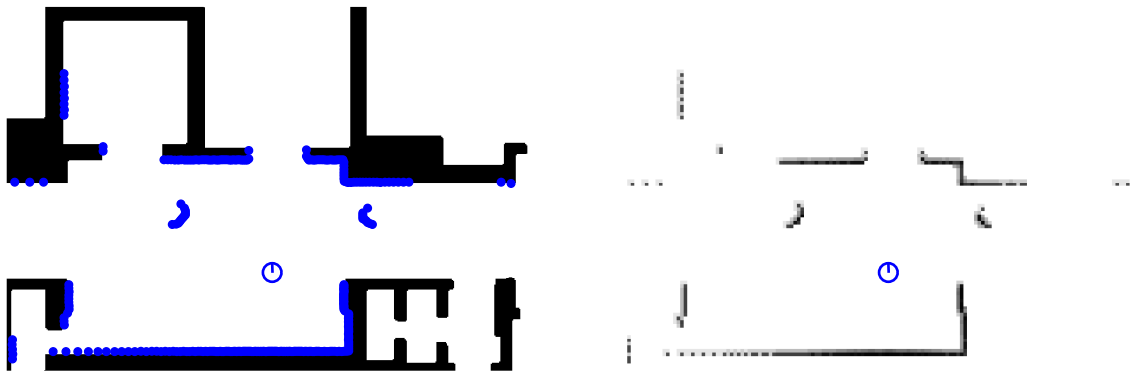
a) b)

Bild 7.1: Bei der Erstellung der Belegtheitskarte werden die Scanpunkte in eine Rasterkarte eingefügt. Bild a) zeigt das Dichteprofil der Belegtheitswahrscheinlichkeit eines Punktes. Bild b) zeigt die Überlagerung mehrerer Scanpunkte.

punkte werden in eine nicht leere Karte hinzugefügt, indem die rekursive Aktualisierungsformel

$$p\left(\boldsymbol{M}_k^{[i]}|\boldsymbol{z}_k^{[1\ldots j]}\right) = p\left(\boldsymbol{M}_k^{[i]}|\boldsymbol{z}_k^{[j-1]}\right) + p\left(\boldsymbol{M}_k^{[i]}|\boldsymbol{z}_k^{[j]}\right) - p\left(\boldsymbol{M}_k^{[i]}|\boldsymbol{z}_k^{[j-1]}\right) \cdot p\left(\boldsymbol{M}_k^{[i]}|\boldsymbol{z}_k^{[j]}\right) \quad (7.1)$$

angewendet wird. Dabei ergibt sich der jeweils neue Belegungswert einer Zelle, indem der bereits vorhandene mit neuen Sensordaten ergänzt wird. Die Belegtheitskarte wird erzeugt, indem alle Scanpunkte nacheinander in eine vorher leere Rasterkarte eingefügt werden (Bild 7.2). Der Kartenursprung im ortsfesten Koordinatensystem wird so festgelegt, dass der Roboter sich in der Kartenmitte befindet.



a) b)

Bild 7.2: Dargestellt ist die Erstellung der Belegtheitskarte. Bild a) zeigt die über der Rasterkarte dargestellten Scanpunkte, aus diesen wird die in b) über der Rasterkarte dargestellte Belegtheitskarte errechnet.

Erstellung der Merkmalskarten

Die Erstellung der Merkmalskarten unterscheidet sich von der der Belegtheitskarte nur dadurch, dass aus dem Messdatensatz z_k solche Messungen $z_k \in z_{k,\langle l \rangle}$ ausgewählt werden, die genau einem Objekt l zuzuordnen sind. Objekte in der Umgebung des Roboters erzeugen lokale Minima in den Entfernungsdaten der Scanner (Bild 7.3). Bezogen auf Personen gehören, je nach der momentanen Beinstellung, entweder ein, oder zwei nebeneinander liegende Minima zu einer Person. Eine vorangehende Filterung der Entfernungsdaten zur Erhöhung des Kontrastes erleichtert die Segmentierung. Dabei wird für jeden Entfernungswert jeweils die Distanz zum benachbarten Messwert ermittelt. Der Entfernungswert wird dann auf die Höhe

des Nachbarn mit der geringeren Distanz gesetzt. Auf diese Weise werden verschwommene Kanten stark geschärft, Ausreißer entfernt und diffuse Punktwolken zu kleinen Segmenten zusammengefasst. Für jedes aufgrund der Minima erkannte

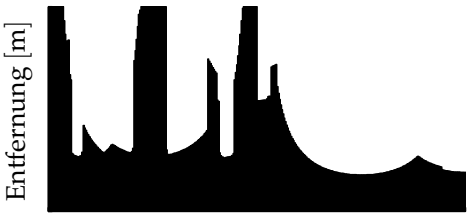


Bild 7.3: Entfernungsdiagramm eines Laserscans. In der Umgebung des Roboters befindliche Hindernisse erzeugen lokale Minima

Objekt wird eine eigene Merkmalskarte erstellt, in die dazu gehörende Messdaten eingefügt werden. Damit enthalten die Kartenzellen die Wahrscheinlichkeit für die Belegtheit durch das l-te Objekt $p(\boldsymbol{M}_{\text{obj},k,\langle l\rangle}|\boldsymbol{z}_{k,\langle l\rangle})$.

Die dynamische Belegtheitskarte

Solche lokalen Minima werden aber nicht nur von bewegten Objekten verursacht, sondern auch von statischen Hindernissen, wie z. B. von Papierkörben oder Blumentöpfen. Um diese statischen Hindernisse von den dynamischen zu unterscheiden, wird die dynamische Belegtheitskarte $p(\boldsymbol{M}_{\text{dyn},k}|\boldsymbol{z}_k, \boldsymbol{z}_{k-1})$ verwendet. Diese berechnet aus zwei auf einander folgenden Belegtheitskarten die Wahrscheinlichkeit, dass bezüglich der Belegung der Zellen eine Veränderung stattgefunden hat.

$$p\left(\boldsymbol{M}_{\text{dyn},k}|\boldsymbol{z}_k, \boldsymbol{z}_{k-1}\right) = p\left(\boldsymbol{M}_{\text{occ},k}|\boldsymbol{z}_k\right) \cdot \left(1 - p\left(\boldsymbol{M}_{\text{occ},k-1}|\boldsymbol{z}_{k-1}\right)\right) \quad . \tag{7.2}$$

Eine solche Rasterkarte ist in Bild 7.4 dargestellt. Man erkennt deutlich Maxima bei den dynamischen Objekten, während die übrigen Bereiche nur blass in Erscheinung treten.

Berechnung der Objekt-Aufenthaltswahrscheinlichkeit

Durch die Kombination der Merkmalskarten $p(\boldsymbol{M}_{\text{obj},k,\langle l\rangle}|\boldsymbol{z}_{k,\langle l\rangle})$ mit der dynamischen Belegtheitskarte $p(\boldsymbol{M}_{\text{dyn},k}|\boldsymbol{z}_k, \boldsymbol{z}_{k-1})$ kann schließlich auf die gesuchte Aufenthaltswahrscheinlichkeit eines Objekts n am Ort der Messung von Merkmal l geschlossen werden.

$$p(\boldsymbol{z}_{\langle l\rangle}|\boldsymbol{x}_{k,\langle n\rangle}) = p\left(\boldsymbol{M}_{\text{obj},k,\langle l\rangle}|\boldsymbol{z}_{k,\langle l\rangle}\right) \cdot p\left(\boldsymbol{M}_{\text{dyn},k}|\boldsymbol{z}_k, \boldsymbol{z}_{k-1}\right) \quad . \tag{7.3}$$

Die aus Gl. (7.3) resultierende Karte weist durch die mehrfache Multiplikation von Wahrscheinlichkeiten nur einen geringen Kontrast auf. Dieser kann dadurch erhöht werden, dass mittels eines Grenzwertes für die Karte aus Gl. (7.3) lediglich

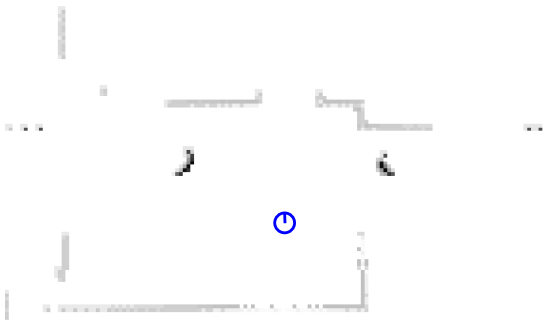


Bild 7.4: Die dynamische Belegtheitskarte: Deutlich sind die Stellen zu erkennen, an denen sich Objekte bewegt haben.

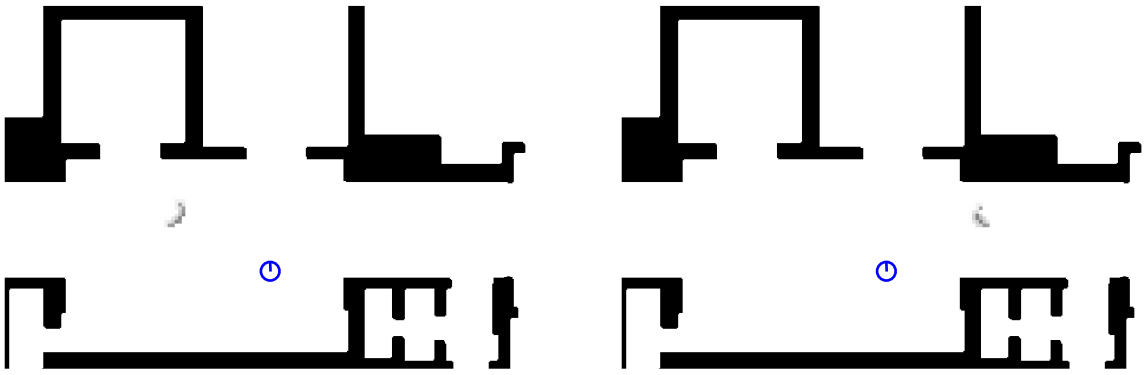


Bild 7.5: Für jedes erkannte Merkmal wird eine Merkmalskarte erstellt.

abgeleitet wird, dass das entsprechende lokale Minimum im Zusammenhang mit der dynamischen Belegtheitskarte als dynamisches Objekt gilt. Für die Erstellung der Merkmalskarte $p(z_{\langle l \rangle}|x_{k,\langle n \rangle})$ werden jedoch statt der Wahrscheinlichkeitskarte Gl. (7.3) direkt die Messdaten des Laserentfernungsmesssystems im Bereich des lokalen Minimums verwendet. Für die in Bild 7.2 dargestellte Situation ergeben sich im Datensatz des Laserentfernungsmesssystems zwei Minima. Diese bleiben auch bei Verrechnung mit der dynamischen Belegtheitskarte erhalten und es werden die in Bild 7.5 dargestellten Merkmalskarten generiert.

Erstellung der Verdeckungskarte

Die Verdeckungskarte repräsentiert die Wahrscheinlichkeit, dass ein Objekt an der Position $x_{k,\langle n \rangle}$ nicht erkannt werden kann. Das passiert entweder wegen eines Feh-

lers in der Merkmalsextraktion, oder weil die Position von einem anderen Objekt verdeckt wird. Der erste Fall wird modelliert, indem jede Zelle einen konstanten Wert $p(\neg\text{detect})$ erhält. Für den zweiten Fall wird ausgehend von jedem Laserentfernungsmesssystem der Abtaststrahl verfolgt und der Schatten, der durch dynamische Objekte verursacht wird, bestimmt. Dieser Schatten erhält dann die Wahrscheinlichkeit $p(\text{occluded})$. Zusätzlich werden statische Hindernisse in Betracht gezogen, um die Bewegung der Partikel aus dem freien Raum um den Roboter zu verhindern. Dafür wird im Schatten hinter statischen Kartenzellen der Wert für $p(\neg\text{detect})$ verkleinert. Aus der Gleichung

$$p(\boldsymbol{z}_{\langle 0\rangle}|\boldsymbol{x}_{k,\langle n\rangle}) = p(\text{occluded} \vee \neg\text{detect}) \tag{7.4}$$

ergibt sich die in Bild 7.6 dargestellte Verdeckungskarte.

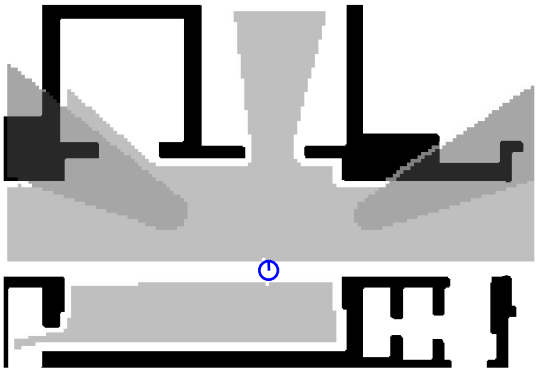


Bild 7.6: Bei der Verfolgung von Objekten ist der Fall zu berücksichtigen, dass sich Objekte gegenseitig verdecken. Dieses erfolgt durch die Berechnung der Verdeckungskarte aus dem dynamischen und statischen Teil einer Messung.

7.2.2 Bewegungsmodell

Kennzeichnend für das Bewegungsmodell sind die unbekannten Stellgrößen. Damit stehen keinerlei a priori Informationen für die Transition $p(\boldsymbol{x}_{k,\langle n\rangle}|\boldsymbol{x}_{k-1,\langle n\rangle})$ zur Verfügung. Wie bereits eingangs erwähnt, besteht die Möglichkeit durch lernbasierte Verfahren eine präzisere Objektklassifikation einzuführen, die es erlaubt die Parameter des Bewegungsmodells besser anzupassen. Für das Bewegungsmodell gilt, dass es im Gegensatz zur Anwendung parametrischer Filter nicht in geschlossener Form vorliegen muss. Für die Anwendung von Partikelfiltern genügt ein Modell, aus dem einzelne Stichproben gezogen werden können. Um die Bewegung der Objekte zu modellieren, wird angenommen, dass sich die Bewegungsgeschwindigkeit

und die Bewegungsrichtung entsprechend einer Normalverteilung ändern. So werden im Vorhersage-Schritt erst die Richtung ϕ und Geschwindigkeit v variiert und anschließend die neue Position x, y unter Einbeziehung der Schrittweite Δt berechnet. Die Wirkungsweise des Bewegungsmodells ist in Bild 7.7 für ein Objekt darge-

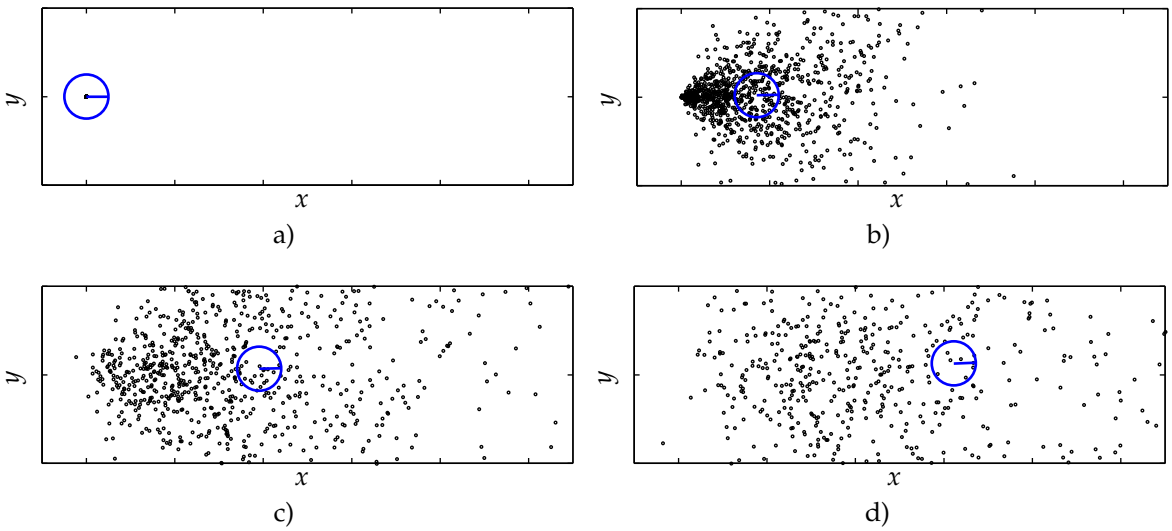


Bild 7.7: Das Bewegungsmodell für Objekte im Wahrnehmungsbereich geht aufgrund der Unkenntnis der Objektbahn von einer geradlinigen Bewegung aus, die durch einen Rauschanteil überlagert ist. Die wiederholte Anwendung des Bewegungsmodells führt zu einem Auseinanderdriften der Partikelwolke.

stellt, dessen Position, Orientierung und Geschwindigkeit zunächst exakt bekannt sind. Ausgehend von der bekannten Position wird das Bewegungsmodell wiederholt auf die Partikelwolke angewendet. Zu erkennen ist, wie sich der Schwerpunkt der Partikelwolke zunächst mit der Anfangsgeschwindigkeit verlagert, während die Unsicherheit über den Zustand, die durch die Streuung der Partikel und damit durch den Durchmesser der Partikelwolke repräsentiert wird, rapide ansteigt. Die Unsicherheit wird schließlich so groß, dass der Einfluss des Objektausgangszustands nicht mehr zu erkennen ist.

7.2.3 Korrespondenzproblem

Bei der Verfolgung mehrerer Objekte besteht das Datenassoziationsproblem. Dieses beinhaltet die Lösung der Zuordnung von Merkmalen in der Sensormessung zu den verfolgten Objekten. Zu diesem Zweck kommt hier das *Joint Probabilistic Data Association*-Filter zum Einsatz, dessen Funktionsweise im Folgenden beschrieben

wird. Das JPDA-Filter setzt voraus, dass die Objektanzahl bekannt ist. In praktischen Anwendungen variiert diese jedoch im Verlauf der Zeit. Fährt der Roboter zum Beispiel durch einen belebten Korridor, so ändert sich die Personenzahl in dessen Wahrnehmungsbereich häufig. Ein weiteres Problem besteht darin, dass nicht immer alle Objekte erkannt werden, weil sie sich gegenseitig verdecken.

Bestimmung der Objektanzahl

Um eine gute Schätzung der Anzahl zu erzielen, wird die Wahrscheinlichkeitsverteilung $p(N_k|L_{k,\ldots,0})$ über die Objektanzahl N_k verfolgt. $L_{k,\ldots,0}$ sind dabei die bisher beobachteten Anzahlen der Merkmale.

Unter Verwendung der BAYES'schen Regel ergibt sich

$$p(N_k|L_{k,\ldots,0}) = \eta p(L_k|N_k, L_{k-1,\ldots,0}) p(N_k|L_{k-1,\ldots,0}) \quad . \tag{7.5}$$

Bei Annahme eines MARKOV-Prozesses 1. Ordnung (bei gegebener Objektanzahl ist die Anzahl der beobachteten Merkmale unabhängig von der Historie) wird daraus

$$p(N_k|L_{k,\ldots,0}) = \eta p(L_k|N_k) p(N_k|L_{k-1,\ldots,0}) \quad . \tag{7.6}$$

Die Aufspaltung mit Hilfe der totalen Wahrscheinlichkeit und die weitere Annahme, dass N_k bei gegebenem N_{k-1} unabhängig von $L_{k-1,\ldots,0}$ ist, ergibt schließlich

$$p(N_k|L_{k,\ldots,0}) = \eta p(L_k|N_k) \sum_n \left[p(N_k|N_{k-1} = n) p(N_{k-1} = n|L_{k-1,\ldots,0}) \right] . \tag{7.7}$$

Das Resultat ist eine rekursive Formel für $p(N_k|L_{k,\ldots,0})$, in der die einzigen unbekannten Größen die Wahrscheinlichkeiten $p(L_k|N_k)$ und $p(N_k|N_{k-1})$ sind. Der erste Term $p(L_k|N_k)$ repräsentiert die Wahrscheinlichkeit, dass L_k Merkmale erkannt werden, wenn sich tatsächlich N_k Objekte im Wahrnehmungsbereich befinden. Diese Wahrscheinlichkeiten lassen sich über Statistiken ermitteln. In dieser Arbeit jedoch werden diese Werte durch die Binomialverteilung

$$p(L_k|N_k) = \binom{N_k}{L_k} p^{L_k} (1-p)^{(N_k - L_k)} \tag{7.8}$$

modelliert. Bei geeigneter Wahl von $p \approx 0,8$ ähnelt das in Bild 7.8a dargestellte Resultat statistisch ermittelten Ergebnissen [Schulz u. a., 2003].

Der zweite Term $p(N_k|N_{k-1})$ gibt an, wie sich die Objektanzahl über die Zeit verändert. Im Experiment mit der Modellierung durch einen POISSON-Prozess hat sich gezeigt, dass besonders bei wenigen Objekten sich das berechnete N_k zu schnell an die Merkmalsanzahl L_k anpasst. Dadurch werden die Partikelfilter kurzzeitig verdeckter Objekte schon nach wenigen Aktualisierungsschritten gelöscht und müssen nach Ende der Verdeckung wieder neu initialisiert werden. Aus diesem Grund

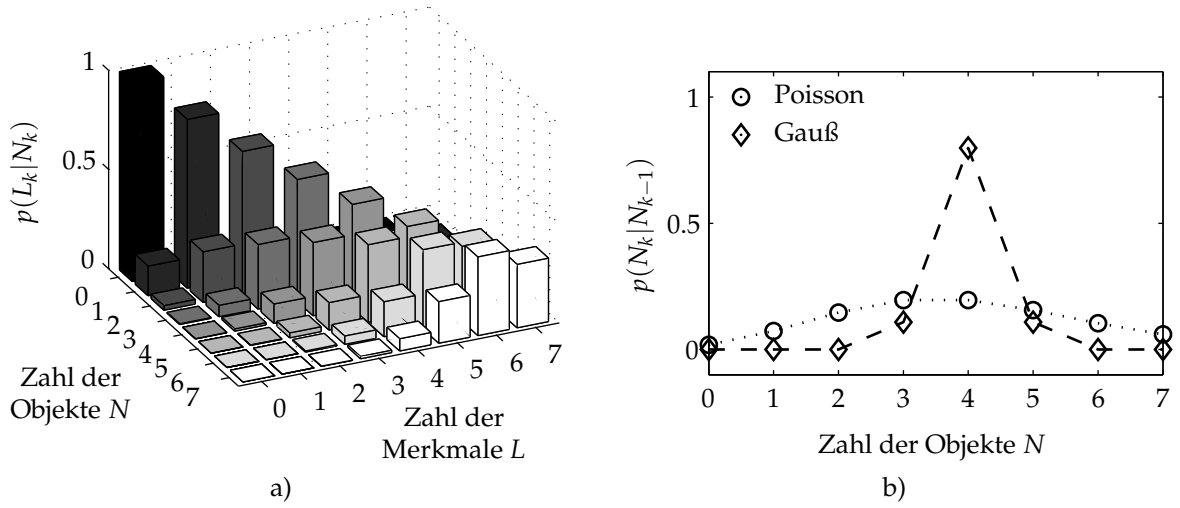


Bild 7.8: Dargestellt sind die Gesetze zur Bestimmung der Objektanzahl aus der Zahl erkannter Features (Bild a) und der Historie (Bild b).

wird in der vorliegenden Arbeit mit alternativen Verteilungen für $p(N_k|N_{k-1})$ experimentiert, unter anderem mit der Normalverteilung (Bild 7.8b). Die Objektanzahl N_k wird schließlich mittels der Maximum-Likelihood-Methode aus der Verteilung $p(N_k|L_{k,\ldots,0})$ bestimmt.

Joint Probabilistic Data Association Filter

Bei der Verfolgung von N verschiedenen Objekten nimmt $\boldsymbol{x}_k$ die Form des Vektors $\boldsymbol{x}_k = \{\boldsymbol{x}_{k,\langle 1\rangle}, \ldots, \boldsymbol{x}_{k,\langle N\rangle}\}$ an, wobei jedes $\boldsymbol{x}_{k,\langle n\rangle}$ eine Zufallsvariable über den gesamten Zustandsraum des entsprechenden Objekts darstellt. Analog bestehen die Sensordaten $\boldsymbol{z}_k$ aus L Merkmalen $\boldsymbol{z}_k = \{\boldsymbol{z}_{k,\langle 1\rangle}, \ldots, \boldsymbol{z}_{k,\langle L\rangle}\}$, die jeweils die Messungen eines Objekts oder ein fälschlicherweise erkanntes Merkmal darstellen.

Um mehrere Objekte gleichzeitig verfolgen zu können, muss die gemeinsame Wahrscheinlichkeit $p(\boldsymbol{x}_{k,\langle 1\rangle}, \boldsymbol{x}_{k,\langle 2\rangle}, \ldots, \boldsymbol{x}_{k,\langle N\rangle})$ bestimmt werden. Unter der Annahme, dass sich die Objekte unabhängig von einander bewegen, erfolgt eine Faktorisierung der Wahrscheinlichkeit. Daraus ergibt sich die Möglichkeit, die Objekte auch unabhängig von einander verfolgen zu können. Das Hauptproblem dieses Ansatzes besteht in der richtigen Zuordnung der einzelnen Merkmale $\boldsymbol{z}_{k,\langle l\rangle}$ zu den Objekten $\boldsymbol{x}_{k,\langle n\rangle}$.

Das Joint Probabilistic Data Association Filter definiert den gemeinsamen Zuordnungsfall θ, der eine Tupelmenge $(l,n) \in \{0,\ldots,L\} \times \{1,\ldots,N\}$ darstellt und jedem Objekt entweder exklusiv ein Merkmal $\boldsymbol{z}_{k,\langle 1\ldots L\rangle}$ zuweist, oder aber $\boldsymbol{z}_{k,\langle 0\rangle}$, falls das Objekt vom Sensor nicht erkannt worden ist. Ferner ist $\Theta_{l,n}$ die Menge aller solcher Zuordnungsfälle, die das Zuordnungstupel (l,n) enthalten. Das JPDA-Filter

berechnet nun die Wahrscheinlichkeit $\beta_{l,n}$, dass das Merkmal l zu dem Objekt n gehört

$$\beta_{l,n} = \sum_{\theta \in \Theta_{l,n}} p(\theta | \boldsymbol{z}_{k\ldots 0}) \quad . \tag{7.9}$$

Durch Anwenden der totalen Wahrscheinlichkeit und wiederholte Annahme eines Markov-Prozesses 1. Ordnung kann die Wahrscheinlichkeit eines einzelnen Zuordnungsfalls berechnet werden

$$p(\theta | \boldsymbol{z}_{k\ldots 0}) \overset{\text{totale Wahrscheinlichkeit}}{\overset{\downarrow}{=}} \int p(\theta | \boldsymbol{z}_{k\ldots 0}, \boldsymbol{x}_k) \cdot p(\boldsymbol{x}_k | \boldsymbol{z}_{k\ldots 0}) \mathrm{d}\boldsymbol{x}_k \tag{7.10}$$

$$\overset{\text{Markov-Prozess 1. Ord.}}{\overset{\downarrow}{=}} \int p(\theta | \boldsymbol{z}_k, \boldsymbol{x}_k) \cdot p(\boldsymbol{x}_k | \boldsymbol{z}_{k\ldots 0}) \mathrm{d}\boldsymbol{x}_k \quad . \tag{7.11}$$

Offensichtlich wird der aktuelle Zustand $p(\boldsymbol{x}_k | \boldsymbol{z}_{k\ldots 0})$ gebraucht, um die Wahrscheinlichkeit eines Zuordnungsfalls $p(\theta)$ zu berechnen, andererseits wird an anderer Stelle $p(\theta)$ gebraucht, um genau diesen Zustand zu ermitteln. Dieses Problem wird dadurch gelöst, dass anstatt des aktuellen Zustands, der durch die Gl. (3.15) vorhergesagte Zustand verwendet wird

$$p(\theta | \boldsymbol{z}_{k\ldots 0}) \approx \int p(\theta | \boldsymbol{z}_k, \boldsymbol{x}_k) \cdot p(\boldsymbol{x}_k | \boldsymbol{z}_{k-1\ldots 0}) \mathrm{d}\boldsymbol{x}_k \quad . \tag{7.12}$$

Mittels Bayes'scher Inferenz ergibt sich

$$p(\theta | \boldsymbol{z}_{k\ldots 0}) = \eta \int p(\boldsymbol{z}_k | \theta, \boldsymbol{x}_k) p(\theta | \boldsymbol{x}_k) \cdot p(\boldsymbol{x}_k | \boldsymbol{z}_{k-1\ldots 0}) \mathrm{d}\boldsymbol{x}_k \quad , \tag{7.13}$$

wobei η eine Normalisierungskonstante darstellt. Der Term $p(\theta | \boldsymbol{x}_k)$ gibt die Wahrscheinlichkeit eines Zuordnungsfalls θ bei gegebenen Objektzuständen wieder. Hier wird davon ausgegangen, dass alle Zuordnungsfälle gleich wahrscheinlich sind, so dass der Term durch eine Konstante ersetzt werden kann.

Der Term $p(\boldsymbol{z}_k | \theta, \boldsymbol{x}_k)$ bestimmt die Wahrscheinlichkeit einer Messung bei gegebenen Objektzuständen für einen konkreten Zuordnungsfall. Um diese zu berechnen, müssen die Fälle betrachtet werden, in denen gemessene Merkmale von keinem Objekt verursacht worden sind. Die Wahrscheinlichkeit dieser falschen Alarme soll γ sein. Wenn $|\theta|$ die Anzahl der Objektzuordnungen zu gemessenen Merkmalen, also ohne $\boldsymbol{z}_{k,\langle 0 \rangle}$ darstellt[2], dann berechnet sich die Wahrscheinlichkeit aller falschen Alarme des Zuordnungsfalls zu $\gamma^{(L-|\theta|)}$. Durch die Annahme, dass alle Merkmale

[2]Es sei darauf hingewiesen, dass die Zuordnung von Merkmalen zu Objekten exklusiv erfolgt.

unabhängig von einander gemessen werden, lässt sich somit folgende Gleichung aufstellen

$$p(\boldsymbol{z}_k|\theta,\boldsymbol{x}_k) = \gamma^{(L-|\theta|)} \prod_{(l,n)\in\theta} p(\boldsymbol{z}_{k,\langle l\rangle}|\boldsymbol{x}_{k,\langle n\rangle}) \quad . \tag{7.14}$$

Gleichung (3.15), die den Zustand aufgrund des Bewegungsmodells vorhersagt, kann an dieser Stelle auch für einzelne Objekte aufgestellt werden

$$p(\boldsymbol{x}_{k,\langle n\rangle}|\boldsymbol{z}_{k-1\ldots0}) = \int p(\boldsymbol{x}_{k,\langle n\rangle}|\boldsymbol{x}_{k-1,\langle n\rangle}) p(\boldsymbol{x}_{k-1,\langle n\rangle}|\boldsymbol{z}_{k-1\ldots0}) \mathrm{d}\boldsymbol{x}_{k-1,\langle n\rangle} \quad . \tag{7.15}$$

Mit dem Einsetzen von Gl. (7.14) in Gl. (7.13) und anschließend in Gl. (7.9) ergibt sich eine rekursive Formel für die Berechnung der Zuordnungswahrscheinlichkeiten

$$\beta_{l,n} = \sum_{\theta\in\Theta_{l,n}} \left(\eta\gamma^{(L-|\theta|)} \prod_{(l,n)\in\theta} \int p(\boldsymbol{z}_{k,\langle l\rangle}|\boldsymbol{x}_{k,\langle n\rangle}) p(\boldsymbol{x}_{k,\langle n\rangle}|\boldsymbol{z}_{k-1\ldots0}) \mathrm{d}\boldsymbol{x}_{k,\langle n\rangle} \right) \quad . \tag{7.16}$$

Um die benötigte Rechenzeit zu begrenzen und dadurch eine größere Objektanzahl verfolgen zu können, werden Zuordnungsfälle, die vernachlässigbare Messwahrscheinlichkeiten $p(\boldsymbol{z}_{k,\langle l\rangle}|\boldsymbol{x}_{k,\langle n\rangle})$ enthalten, ignoriert. Dieses betrifft solche Zuordnungsfälle, bei denen sehr kleine Wahrscheinlichkeiten im Produkt auftreten und die dadurch nur einen sehr kleinen Anteil zur Summe von $\beta_{l,n}$ aufweisen.

Um die Zustandsverteilungen der einzelnen Objekte berechnen zu können, ist die Integration der Merkmal-zu-Objekt-Zuordnungen in die Bayes'sche Rekursionsformel Gl. (3.16) erforderlich:

$$p(\boldsymbol{x}_{k,\langle n\rangle}|\boldsymbol{z}_{k\ldots0}) = \eta \sum_{l=0}^{L} \beta_{l,n} p(\boldsymbol{z}_{k,\langle l\rangle}|\boldsymbol{x}_{k,\langle n\rangle}) p(\boldsymbol{x}_{k,\langle n\rangle}|\boldsymbol{z}_{k-1\ldots0}) \quad . \tag{7.17}$$

Zusammenfassend lässt sich der Aktualisierungszyklus für die Objektzustände wie folgt beschreiben. Erst wird aufgrund der vorherigen Zustandsverteilung und des Bewegungsmodells mit Hilfe von Gl. (7.15) eine Vorhersage für den aktuellen Zustand getroffen. Diese wird in Gl. (7.16) eingesetzt, um die Zuordnungswahrscheinlichkeiten der Merkmale zu den Objekten zu ermitteln. Schließlich wird durch Gl. (7.17) die Wahrscheinlichkeitsverteilung über die momentanen Objektzustände aktualisiert.

Berechnung der Zuordnungswahrscheinlichkeiten

Die Berechnung der in Abschnitt 7.2.4 hergeleiteten Gl. (7.20) erweist sich als nicht trivial, sondern stellt sich als komplexe und rechenaufwändige Aufgabe heraus. Vor

diesem Hintergrund soll die Umsetzung der Berechnung hier detailliert dargestellt werden. Der Aufwand besteht darin, alle möglichen Zuordnungsfälle θ aus der Menge $\Theta_{l,n}$ zu generieren. Wie in Abschnitt 7.2.3 vorgestellt, weist θ jedem Objekt entweder exklusiv ein Merkmal $\boldsymbol{z}_{\langle 1\rangle \ldots \langle L\rangle}$ oder das Ersatzmerkmal $\boldsymbol{z}_{\langle 0\rangle}$ zu. Somit stellt θ eine Liste mit N Einträgen dar. Durch die Berechnung der Summe aus Gl. (7.20)

$$\beta_{l,n} = \sum_{\theta \in \Theta_{l,n}} \left(\eta \gamma^{(L-|\theta|)} \prod_{(l,n)\in\theta} \frac{1}{M} \sum_{m=1}^{M} p(\boldsymbol{z}_{k,\langle l\rangle} | \boldsymbol{x}_{k,\langle n\rangle,[m]}) \right)$$

werden alle Zuordnungsfälle generiert, in denen das Merkmal $\boldsymbol{z}_{\langle l\rangle}$ dem Objekt n zugeordnet ist. Der Platz n ist damit in allen θ-Listen von vornherein mit dem Eintrag $\boldsymbol{z}_{\langle l\rangle}$ belegt. Die Belegung der übrigen $N-1$ Einträge wird im Folgenden dargestellt. Die Zuordnung der Merkmale für nicht erkannte Objekte $\boldsymbol{z}_{\langle 0\rangle}$ stellt eine Besonderheit dar, denn diese können auch in mehreren Einträgen enthalten sein. Wie bereits erwähnt, ist eine präzise Bestimmung von $\beta_{l,n}$ nicht erforderlich, wenn im Lauf der Berechnung deutlich wird, dass die Wahrscheinlichkeit eine Grenze $P_{\min}$ unterschreitet. Diese ergibt sich daraus, dass die faktorisierte Wahrscheinlichkeit und die Beschränkung der diskreten Wahrscheinlichkeit auf Werte ≤ 1 eine Vergrößerung der Wahrscheinlichkeit ausschließen. Algorithmus 7.1 verdeutlicht den Ablauf.

Vor der Ausführung des Algorithmus wird eine Tabelle $\boldsymbol{P}_{\boldsymbol{l}\mapsto\boldsymbol{n}}$ erzeugt. Sie beinhaltet die häufig gebrauchten Ergebnisse der zweiten Summe von Gl. (7.20), die die Einzelzuordnungswahrscheinlichkeit der l-ten Messung $z_{k,\langle l\rangle}$ zum n-ten Objekt $\boldsymbol{x}_{k,\langle n\rangle,[m]}$ darstellt

$$\boldsymbol{P}_{\boldsymbol{l}\mapsto\boldsymbol{n}}(l,n) = \frac{1}{M} \sum_{m=1}^{M} p(\boldsymbol{z}_{k,\langle l\rangle} | \boldsymbol{x}_{k,\langle n\rangle,[m]}) \quad . \tag{7.18}$$

Die Variablen $\boldsymbol{L}_{\text{objects}}$ und $\boldsymbol{L}_{\text{features}}$ stellen Listen mit Objekten, bzw. Merkmalen dar, die noch nicht in θ enthalten sind. In Zeile 5 wird $\boldsymbol{L}_{\text{features}}$ mit allen Merkmalen außer $\boldsymbol{z}_{\langle l\rangle}$ initialisiert. Dieses wird in Zeile 6 dem Objekt n zugeordnet. Analog dazu wird $\boldsymbol{L}_{\text{objects}}$ in Zeile 12 initialisiert. Zuerst wird ermittelt, wie oft $\boldsymbol{z}_{\langle 0\rangle}$ zugeordnet werden soll (Zeile 7). Die Anzahl dieser Zuordnungen wird im Folgenden n_{z_0} genannt. Wenn mehr Objekte vorhanden, als Merkmale erkannt worden sind, d.h. $N > L$, müssen mindesten $n_{z_0} = N - L$ Einträge mit $\boldsymbol{z}_{\langle 0\rangle}$ belegt werden. Falls l bereits $\boldsymbol{z}_{\langle 0\rangle}$ ist, dann vermindert sich n_{z_0} um 1. Von der Minimalanzahl ausgehend wird n_{z_0} solange erhöht, bis alle restlichen θ-Einträge $\boldsymbol{z}_{\langle 0\rangle}$ enthalten (Zeile 8). Nun muss die ermittelte Menge der $\boldsymbol{z}_{\langle 0\rangle}$-Merkmale auf die nicht belegten Objekte verteilt werden. Dabei kommen alle Möglichkeiten in Betracht, n_{z_0} Merkmale auf

Algorithmus 7.1 Algorithmus zur Berechnung von gemeinsamen Zuordnungswahrscheinlichkeiten durch das JPDA-Filter.

```
1:  ComputeBeta(l, n)
2:      β_{l,n} = 0
3:      if P_{l↦n}(l, n) < P_min then return β_{l,n}
4:      θ(1; ...; N) = 0
5:      L_features = {1; ...; L} \ {l}
6:      θ(n) = l
7:      if l = 0 then n_{z_0} = N − L − 1 else n_{z_0} = N − L
8:      for n_{z_0} = max(n_{z_0}, 0) to N − 1 do
9:          if l = 0 then n_falseAl = L − (N − n_{z_0}) + 1 else n_falseAl = L − (N − n_{z_0})
10:         p_rek = γ^{n_falseAl}
11:         if p_rek < P_min then return β_{l,n}
12:         L_objects = {1; ...; N} \ {n_{z_0}}
13:         L_objComb = combinations(L_objects, n_{z_0})
14:         repeat[1]
15:             for all n_c ∈ L_objComb do
16:                 θ(n_c) = 0
17:                 L_objects = L_objects \ {n_c}
18:                 if P_{l↦n}(n_c, 0) < P_min then continue repeat[1]
19:             end for
20:             L_featComb = combinations(L_features, N − 1 − n_{z_0})
21:             repeat[2]
22:                 L_featPerm = permutations(L_featComb)
23:                 repeat[3]
24:                     for all n_o ∈ L_objects , l_f ∈ L_featPerm do
25:                         θ(n_o) = l_f
26:                         if P_{l↦n}(n_o, l_f) < P_min then continue repeat[3]3
27:                     end for
28:                     for all (n_t, l_t) ∈ θ do p_rek = p_rek · P_{l↦n}(n_t, l_t) end for
29:                     β_{l,n} = β_{l,n} + p_rek
30:                 until L_featPerm = ∅
31:             until L_featComb = ∅
32:         until L_objComb = ∅
33:     end for
34:     return β_{l,n}
35: end
```

Merkmal	$z_{\langle 0\rangle}$		$z_{\langle 0\rangle}$		$z_{\langle 3\rangle}$	
$\theta \in \Theta_{3,5}$	↓	↓	↓	↓	⇓	↓
Objekt	1	2	3	4	5	6

a)

Merkmal		$z_{\langle 0\rangle}$			$z_{\langle 3\rangle}$	$z_{\langle 0\rangle}$
$\theta \in \Theta_{3,5}$	↓	↓	↓	↓	⇓	↓
Objekt	1	2	3	4	5	6

b)

Bild 7.9: Zwei Beispiele für verschiedene Zuordnungen zweier $z_{\langle 0\rangle}$-Merkmale. Die Zweier-Tupel $\{1,3\}$ bzw. $\{2,6\}$ sind 2 Kombinationen aus der Menge noch freier θ-Einträge $\{1,2,3,4,6\}$. Der Eintrag 5 ist von vornherein mit z_3 belegt.

Merkmal	$z_{\langle 0\rangle}$	$z_{\langle 1\rangle}$	$z_{\langle 0\rangle}$	$z_{\langle 4\rangle}$	$z_{\langle 3\rangle}$	$z_{\langle 6\rangle}$
$\theta \in \Theta_{3,5}$	↓	↓	↓	↓	⇓	↓
Objekt	1	2	3	4	5	6

a)

Merkmal	$z_{\langle 0\rangle}$	$z_{\langle 4\rangle}$	$z_{\langle 0\rangle}$	$z_{\langle 6\rangle}$	$z_{\langle 3\rangle}$	$z_{\langle 1\rangle}$
$\theta \in \Theta_{3,5}$	↓	↓	↓	↓	⇓	↓
Objekt	1	2	3	4	5	6

b)

Bild 7.10: Das 3-Tupel $\{z_{\langle 1\rangle}, z_{\langle 4\rangle}, z_{\langle 6\rangle}\}$ ist eine Kombination aus der Menge unbenutzter Merkmale. Das Beispiel zeigt zwei Permutationen dieses Tupels.

$N-1$ Objekte zu verteilen. Die n_{z_0} Kombinationen aus der Menge nicht belegter Objekte decken alle diese Möglichkeiten ab (siehe als Beispiel Bild 7.9).

Die erste Kombinationen wird in Zeile 13 initialisiert. In der **while**[1]-Schleife werden alle Kombinationen berechnet und abgearbeitet. Die **for**-Schleife (Zeilen 15–19) dient dazu, den entsprechenden θ-Einträgen das $z_{\langle 0\rangle}$-Merkmal zuzuordnen.

Daraufhin werden $N-1-n_{z_0}$ Merkmale auf die immer noch freien θ-Einträge aufgeteilt. Dabei erfolgt wiederum die Auswahl aller Kombinationen aus der in $\boldsymbol{L}_{\text{features}}$ zur Verfügung stehenden Menge (Zeile 20). Ferner werden alle Permutationen einer Auswahl berücksichtigt (Zeile 22). Das Beispiel in Bild 7.10 zeigt eine Permutation einer Kombination. Die Vervollständigung der θ-Liste erfolgt in den Zeilen 24–27.

In der **for**-Schleife (Zeile 28) wird schließlich das Produkt eines Zuordnungsfalls berechnet und in Zeile 29 zum bisherigen $\beta_{l,n}$ addiert. Dabei ist p_{rek} bereits in Zeile 10 mit der Wahrscheinlichkeit $\gamma^{(L-|\theta|)}$ initialisiert worden. $L-|\theta|$ wird in Zeile 9 ermittelt und in n_{falseAl} gespeichert.

Der geschilderte Ablauf berechnet die vollständige Zuordnungswahrscheinlichkeit $\beta_{l,n}$. Diese vollständige Berechnung ist in der Regel zeitintensiv, denn die meisten Zuordnungsfälle θ enthalten vernachlässigbar kleine Messwahrscheinlichkeiten, die kaum zur Aufsummierung von $\beta_{l,n}$ beitragen. Aus diesem Grund sind

in dem Algorithmus an mehreren Stellen Abbruchbedingungen eingebaut, die bei Erkennung unbedeutender Zuordnungsfälle den Ablauf verkürzen. Die Konstante $P_{\min}$ bezeichnet die minimale Wahrscheinlichkeit für einen Faktor des Produkts aus Gl. (7.20). In den Zeilen 3, 18 und 26 wird überprüft, ob eine Zuordnung aus θ eine zu geringe Wahrscheinlichkeit besitzt. In Zeile 11 wird der Algorithmus abgebrochen, wenn die Anzahl falscher Alarme eine zu geringe Wahrscheinlichkeit verursacht.

7.2.4 Verfolgung mittels Partikelfilter

Der in Abschnitt 7.2.3 ausgearbeitete Ansatz ermöglicht die exakte Berechnung der Dichte der Aufenthaltswahrscheinlichkeit von dynamischen Objekten über den gesamten Zustandsraum. Die analytische Lösung dieser Gleichungen ist jedoch wiederum nur für einige Spezialfälle bekannt. Das KALMAN-Filter setzt normal verteilte, lineare Modelle voraus. Für beliebige Wahrscheinlichkeitsdichten, z. B. multimodale Wahrscheinlichkeitsdichten, stellt sich die Berechnung nach Gl. (7.17) für praktische Zwecke als ein unlösbares Problem heraus, denn der Rechenaufwand erhöht sich exponentiell mit der Größe und Dimension des Zustandsraums. Als Lösungsansatz wird hier wiederum das nichtparametrische Filter aus Abschnitt 3.4 gewählt.

Bei der Objektverfolgung stellen die Partikel Hypothesen über die möglichen Aufenthaltsorte eines Objekts dar. Eine Partikelmenge $\chi_{k,\langle n\rangle}$ besteht in der Regel aus $M > 100$ Partikeln $\chi_{k,\langle n\rangle} = \{\boldsymbol{x}_{k,\langle n\rangle,[1]}, \ldots, \boldsymbol{x}_{k,\langle n\rangle,[M]}\}$. Die Berechnung der Plausibilität eines Partikels lautet unter Berücksichtigung der Zuordnungswahrscheinlichkeit

$$w_{k,\langle n\rangle,[m]} = \eta \sum_{l=0}^{L} \beta_{l,n} \cdot p(\boldsymbol{z}_{k,\langle l\rangle} | \boldsymbol{x}_{k,\langle n\rangle,[m]}) \quad . \tag{7.19}$$

Es bleibt, die in Kapitel Abschnitt 7.2.3 hergeleiteten Zuordnungswahrscheinlichkeiten mit Hilfe des Partikelfilters zu berechnen. Da die Wahrscheinlichkeit eines Zustands durch die Häufigkeit der Partikel ausgedrückt wird, lässt sich das in Gl. (7.16) benötigte Integral hier durch eine Summe ersetzen. Die neue Gleichung lautet

$$\beta_{l,n} = \sum_{\theta \in \Theta_{l,n}} \left(\eta \gamma^{(L-|\theta|)} \prod_{(l,n)\in\theta} \frac{1}{M} \sum_{m=1}^{M} p(\boldsymbol{z}_{k,\langle l\rangle} | \boldsymbol{x}_{k,\langle n\rangle,[m]}) \right) \quad . \tag{7.20}$$

Es wird für jedes verfolgte Objekt n ein eigenes Partikelfilter eingesetzt. Die Umwandlung in eine gleich gewichtete Partikelwolke erfolgt über das Resampling-Verfahren aus Abschnitt 3.4.

Initialisierung

Sobald neue Objekte in den Wahrnehmungsbereich eintreten, werden entsprechend viele neue Filter initialisiert. Dieses wird erreicht, indem die Partikel des neuen Objekts gleichmäßig über den gesamten Einzugsbereich der Merkmalskarten verteilt werden (Bild 7.11). Auch die Orientierungen und Geschwindigkeiten der Partikel werden zufällig verteilt, wobei v im Bereich $0 \ldots 50$ $^{\mathrm{cm}}/_{\mathrm{s}}$ liegt. Die Zuordnung der neuen Filter zu den hinzugekommenen Objekten wird dem JPDA-Aktualisierungsprozess überlassen.

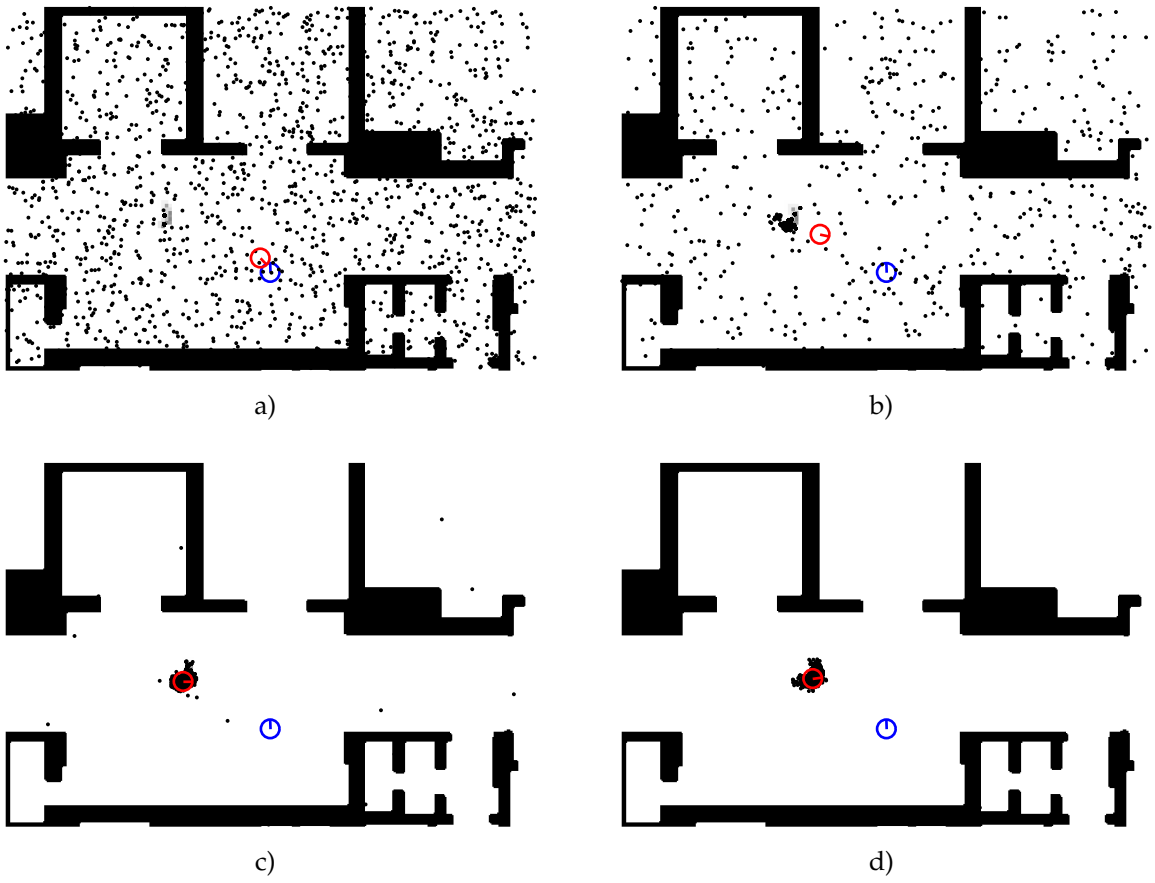


Bild 7.11: Tritt ein Objekt in den Wahrnehmungsbereich des Roboters ein, so ist die Partikelwolke des neu initialisierten Filters zunächst über den Zustandsraum verteilt. In den Iterationen des Algorithmus erfolgt eine Konzentration der Partikel über dem Objekt. Dargestellt sind aufeinander folgende Iterationen.

Erkennung überflüssiger Filter

Wenn Objekte den Wahrnehmungsbereich verlassen, müssen überflüssig gewordene Filter entfernt werden. Um diese erkennen zu können, wird für jede Partikelmenge der gleitende Durchschnitt über die Summe der Gewichte aller Partikel vor der Normalisierung berechnet

$$W_{k,\langle n\rangle} = \sum_{m=1}^{M} w_{k,\langle n\rangle,[m]} \tag{7.21}$$

$$\hat{W}_{k,\langle n\rangle} = \delta W_{k,\langle n\rangle} + (1-\delta)\hat{W}_{k-1,\langle n\rangle} \quad . \tag{7.22}$$

In Abschnitt 7.2.2 wird das Bewegungsmodell beschrieben. Es wird dargestellt, wie schnell die Unsicherheit zu einer großen Streuung der Partikelwolke führt. Der

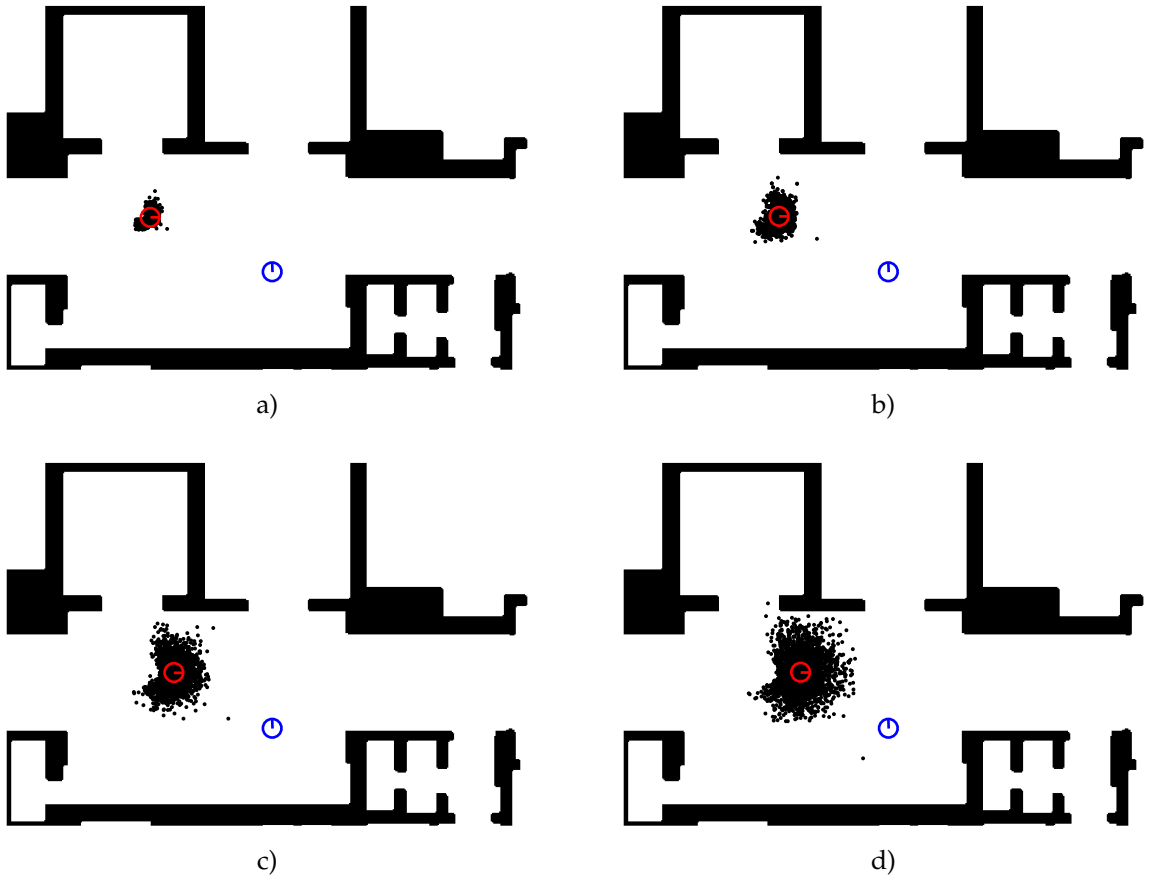


Bild 7.12: Wird für ein Objekt kein Feature in der Messung erkannt, so wird das Bewegungsmodell ohne Korrektur angewendet und die Partikelwolke driftet auseinander. Die Unsicherheit in der Partikelwolke ist ein Hinweis darauf, welches Filter ggf. zu entfernen ist.

Durchschnitt der Partikelgewichte sinkt daher sehr schnell, sobald ein Filter kein Objekt mehr verfolgt. Die Anzahl der aktiven Filter wird angepasst, indem die Partikelfilter mit dem geringsten Durchschnitt $\hat{W}_{k,\langle n\rangle}$ entfernt werden.

Schätzung der Objektzustände

Unter der Annahme, dass sich die Partikel nach einigen Iterationen im Bereich des wahren Zustands konzentrieren, kann eine Aussage über die aktuelle Position aus der Partikelwolke durch eine Überlagerung der verschiedenen Hypothesen gebildet werden. Die Zustände der Objekte $\boldsymbol{x}_{k,\langle 1\rangle}, \ldots, \boldsymbol{x}_{k,\langle N\rangle}$ werden über den Erwartungswert der zugehörigen Partikelmenge bestimmt. Da die Wahrscheinlichkeitsdichte durch die Dichte der Partikel approximiert wird, berechnet sich der Erwartungswert aus dem Durchschnitt über alle Partikel

$$x = \frac{1}{M} \cdot \sum_{m=1}^{M} \boldsymbol{x}_{k,\langle l\rangle,[m]}^{[x]} \tag{7.23}$$

$$y = \frac{1}{M} \cdot \sum_{m=1}^{M} \boldsymbol{x}_{k,\langle l\rangle,[m]}^{[y]} \tag{7.24}$$

$$v = \frac{1}{M} \cdot \sum_{m=1}^{M} \boldsymbol{x}_{k,\langle l\rangle,[m]}^{[v]} \quad . \tag{7.25}$$

Die Orientierung wird durch eine Aufsummierung der Richtungsvektoren bestimmt:

$$x_\Sigma = \sum_{m=1}^{M} \cos\left(\boldsymbol{x}_{k,\langle l\rangle,[m]}^{[\phi]}\right) \tag{7.26}$$

$$y_\Sigma = \sum_{m=1}^{M} \sin\left(\boldsymbol{x}_{k,\langle l\rangle,[m]}^{[\phi]}\right) \tag{7.27}$$

Der resultierende Winkel ergibt sich dann zu

$$\phi = \operatorname{atan2}(y_\Sigma, x_\Sigma) \quad . \tag{7.28}$$

Die Varianz

$$\sigma_x^2 = \frac{1}{M} \cdot \sum_{m=1}^{M} \left(\boldsymbol{x}_{k,\langle l\rangle,[m]}^{[x]} - x\right)^2 \tag{7.29}$$

$$\sigma_y^2 = \frac{1}{M} \cdot \sum_{m=1}^{M} \left(\boldsymbol{x}_{k,\langle l\rangle,[m]}^{[y]} - y\right)^2 \tag{7.30}$$

$$\sigma_v^2 = \frac{1}{M} \cdot \sum_{m=1}^{M} \left(\boldsymbol{x}_{k,\langle l\rangle,[m]}^{[v]} - v\right)^2 \tag{7.31}$$

$$\sigma_\phi^2 = \frac{1}{M} \cdot \sum_{m=1}^{M} \left(\boldsymbol{x}_{k,\langle l\rangle,[m]}^{[\phi]} - \phi\right)^2 \tag{7.32}$$

liefert schließlich einen Hinweis auf die Unsicherheit der aktuellen Zustandsschätzung.

7.3 Experimentelle Erprobung der Objektverfolgung

Der vorgestellte Algorithmus wird in der Simulation und auf dem Versuchsträger erprobt. Im Folgenden sind einige Ergebnisse dieser Erprobung dargestellt. Diese umfassen die Initialisierungsphase, die Ausführungszeiten und die Verfolgungsgenauigkeit.

Sobald ein Objekt in den Wahrnehmungsbereich des Roboters eintritt, wird ein neues Partikelfilter initialisiert. Dessen Partikel sind über den gesamten Wahrnehmungsbereich, der durch die internen Rasterkarten aufgespannt wird verteilt. In den folgenden Iterationen wird das Filter durch den Assoziationsprozess dem Objekt zugeordnet. Die Partikel konzentrieren sich zunehmend auf dessen wahrscheinlichsten Aufenthaltsort. Während dieser Initialisierungsphase kann der Objektzustand noch nicht zuverlässig bestimmt werden. Die Phase dauert in der Regel einige Iterationsschritte und hängt im Wesentlichen von der Größe des Wahrnehmungsbereichs und der Anzahl der Partikel ab. Der in Gl. (7.22) gebildete gleitende Durchschnitt über die Summe der Partikelgewichte ist um so höher, je mehr Partikel auf dem gemessenen Merkmal liegen. Somit ist dieser ein guter Indikator für die Clusterung der Partikel und damit für die Zuverlässigkeit der Zustandsschätzung. Dieser Wert kann genutzt werden, um das Ende der Initialisierungsphase zu erkennen. In Bild 7.13 ist der Verlauf der Initialisierung durch den gleitenden Durchschnitt und den Schätzfehler über den aufeinander folgenden Iterationen dargestellt. Die Messdaten wurden unter Verwendung der Wahrnehmungskarte mit einer Größe von 150 Pixel, bei 75 mm Kantenlänge eines Pixels, erfasst. Somit beträgt die Fläche des Wahrnehmungsbereichs 126,5 m^2. Je nach der Zahl der Partikel ist die Initialisierungsphase nach 4 bis 6 Iterationen abgeschlossen.

Der Algorithmus basiert auf der Zustandsschätzung mittels Partikelfiltern. Bei einer Zahl von $M \geq 1000$ Partikeln je Objekt ergibt sich ein nennenswerter Rechenaufwand. Zusätzlich müssen Wahrscheinlichkeitskarten erstellt werden. Je effizienter der Algorithmus arbeitet, desto höher kann die Aktualisierungsfrequenz gewählt werden, bzw. desto mehr Objekte können gleichzeitig verfolgt werden. In Tabelle 7.1 ist die Ausführungszeit von Bestandteilen des Algorithmus dargestellt. Die Zeiten in der Tabelle wurden auf der Hardware des Versuchsträgers aus Kapitel 4 ermittelt. Der Algorithmus wurde auf einem Intel Pentium 750 MHz Prozessor mit 256 MB Hauptspeicher ausgeführt. Neben dem Algorithmus wurde auf dem Prozessor zusätzlich Rechenzeit durch andere Softwaremodule beansprucht, so dass

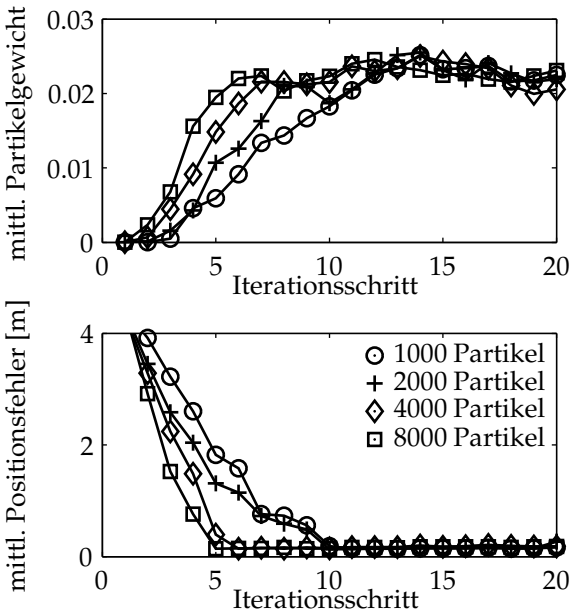


Bild 7.13: Dargestellt ist der Initialisierungsvorgang eines Partikelfilters bei unterschiedlichen Partikelzahlen über den Iterationsschritten. Oben ist das durchschnittliche Partikelgewicht als Maß für den Fortschritt des Einschwingvorgangs dargestellt. Unten ist die durchschnittliche Abweichung der Partikel von der Objektposition aufgetragen.

Tabelle 7.1: Durchschnittliche Ausführungszeiten in ms. Die Dauer des gesamten Durchlaufs und einzelner Teilaufgaben sind in Abhängigkeit von der Objektanzahl N und Partikelanzahl M dargestellt. Die Zeit für die Kartengenerierung beträgt ca. 68 ms, sie ist unabhängig von M und N

	Zuordnungs-Wahrscheinlichkeiten		Bewegungsmodell		Gewichte, Sortierung, Resampling		Gesamtmodul-durchlauf	
N \ M	2000	4000	2000	4000	2000	4000	2000	4000
0	0,0	0,0	0,0	0,0	0,0	0,0	69,8	67,7
1	3,5	12,1	6,8	10,7	10,2	29,8	92,1	127,9
2	17,7	35,7	11,6	23,1	19,7	62,0	122,7	196,3
3	32,4	68,7	15,3	34,4	31,1	86,6	152,8	265,0
4	46,9	118,2	19,9	42,7	47,1	120,5	189,5	357,8
5	74,0	141,2	25,7	55,8	64,7	147,3	238,6	424,6
6	104,0	-	34,6	-	90,5	-	304,4	-

die Ergebnisse nicht als absolute Messungen betrachtet werden können, sondern zueinander in Beziehung stehen. Für den zuverlässigen Betrieb des Moduls mit einer Taktrate von 3 Hz ist die Gesamtausführungszeit auf ca. 300 ms begrenzt. Anhand der Tabelle ist für die genannte Hardware-Konfiguration zu erkennen, dass bei Verwendung von 2000 Partikeln maximal fünf Objekte bzw. bei 4000 Partikeln maximal drei Objekte verfolgt werden können.

In Bild 7.14 ist der exemplarische Verlauf der Testbahn eines Objekts dargestellt. Es handelt sich hierbei um ein simuliertes Szenario, da andernfalls die tatsächliche Objektposition nicht zur Verfügung steht. Die graue Linie repräsentiert den tatsächlichen Verlauf der Bahn, die schwarze Linie zeigt den von der Objektverfolgung geschätzten Verlauf. Zu erkennen ist ein bleibender Positionsfehler. Dieser resultiert aus der Tatsache, dass das Laserentfernungsmesssystem nur die ihm zugewandte Oberfläche abtastet und nicht den gesamten Umfang des Objekts wahrnehmen kann. Somit entspricht der Schwerpunkt der Scanpunkte nicht dem wahren Mittelpunkt des Objekts. Weiterhin sind einige Ausreißer zu erkennen. Diese sind auf das Sensormodell zurückzuführen. Bei der Prüfung der Partikel auf Plausibilität wird ausschließlich die Position der Partikel bewertet, während Geschwindigkeit und Orientierung nur mittelbar einfließen. Erst nach mehrmaliger Anwendung des Bewegungsmodells, welches eine geradlinige Bewegung voraussagt, sterben die Partikel, die sich nicht mit dem Objekt bewegen, im Resamplingprozess aus.

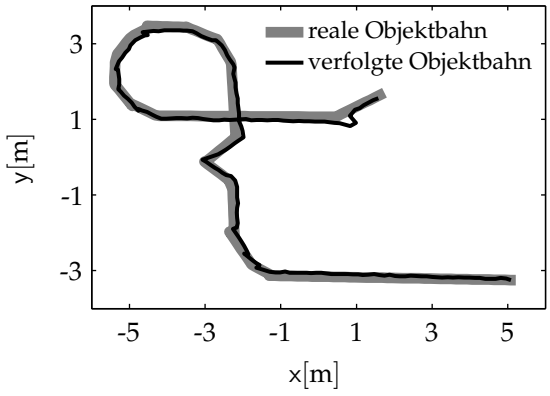


Bild 7.14: Dargestellt ist die beispielhafte Bahn eines verfolgten Objektes.

Durch den Einsatz des Filters kann bei kurzzeitigen Verdeckungen die Schätzung des Objektzustands für einige Zeitschritte aufrecht erhalten werden. Im JPDA-Filter wird dem Objekt dann das z_0 Merkmal zugewiesen, damit wird für die Zustandsschätzung zumindest noch die *negative Information* genutzt, an welcher Stelle sich das Objekt nicht befindet. Naturgemäß steigt die Unsicherheit über den Zustand jedoch an. In Bild 7.15 ist der Verlauf der Partikelwolke für zwei sich begegnende

Objekte dargestellt, wobei eine kurzzeitige Verdeckung auftritt. Es ist deutlich zu erkennen, dass im Moment der Abschattung die Unsicherheit über den Zustand des verdeckten Objekts ansteigt und, nachdem das Objekt aus dem Schatten heraustritt, wieder reduziert wird.

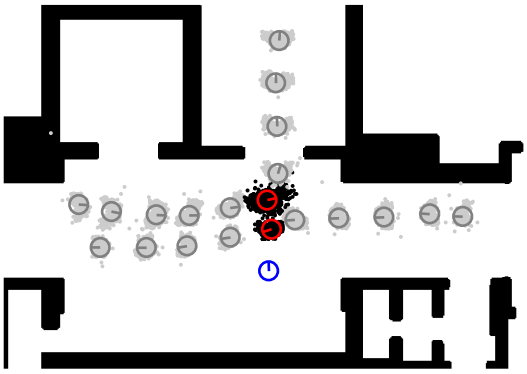
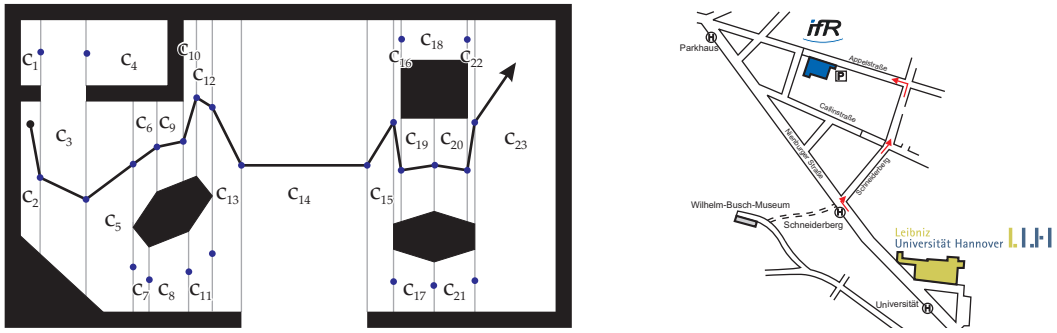


Bild 7.15: Bei Bewegung mehrerer Objekte im Wahrnehmungsbereich des Roboters kann es zu gegenseitigen Verdeckungen kommen. Bei diesem Vorgang steigt die Unsicherheit in der Positionsschätzung des verdeckten Objekts hier, dargestellt durch die vergrößerte schwarz dargestellte Partikelwolke, an.

Die Zuordnung eines neu in den Wahrnehmungsbereich des Roboters eintretenden Objekts zu dem neu initialisierten Partikelfilter geschieht in den meisten Fällen korrekt. Auch die anschließende Objektverfolgung durch den gesamten Wahrnehmungsbereich verläuft dann ohne Aussetzer. Jedoch treten in seltenen Fällen Fehler auf, die auf den im Algorithmus verankerten Nichtdeterminismus zurückzuführen sind. Es kann passieren, dass sich bei zwei vorhandenen Objekten zwei neue Partikelfilter auf nur eines der Objekte konzentrieren und dieses dann auch verfolgen. In seltenen Fällen wird nur der dynamische Schatten eines Objekts über verfolgt, bevor es korrekt erfasst wird. Schließlich kann sich ein Fehler ergeben, wenn gleichzeitig ein Objekt aus dem Wahrnehmungsbereich tritt und eine anderes an einer anderen Stelle hinzukommt. In diesem Fall wird kein neues Partikelfilter initialisiert, da die Anzahl der Objekte gleich bleibt. Das Objekt bleibt dann so lange unerkannt, bis die Partikelwolke des frei gewordenen Filters sich genügend weit ausbreitet.

8 Bahnplanung, Navigation und Hindernisvermeidung

Der Kern der Aufgabenstellung bei der Bahnplanung, Navigation und Hindernisvermeidung ist, mit einer gerichteten Bewegung von einer bekannten Ausgangsposition in eine gewünschte Zielposition zu gelangen. Dabei ist eine Bahn zu planen, die unter Vermeidung von Kollisionen mit statischen und dynamischen Hindernissen abzufahren ist. Die Bahnplanung innerhalb statischer Umgebung kann als gelöstes Problem betrachtet werden. Szenarien, in denen die Trajektorien bewegter Objekte bekannt sind, können als quasistatisch bezeichnet werden, da sich das Problem der Pfadsuche um eine Dimension erweitert, ohne Anforderungen an die Reaktivität des Systems zu stellen. Hier existieren geeignete Transformationen, die das Problem auf ein 2-dimensionales Problem zurückführen. Der Bereich der Navigation besteht aus der Kombination einer effektiven Lokalisation, einer Bahnregelung und einer reaktiven oder statischen Bahnplanung. Der Bereich der Hindernisvermeidung bezieht sich im Sinne der vorliegenden Arbeit auf die dynamischen Hindernisse, die sich mit einer Geschwindigkeit in der Größenordnung von der des Roboters im Arbeitsraum bewegen. Bei der Vermeidung dynamischer Hindernisse bedarf es eines reaktiven und schnellen Planungsalgorithmus. Dieser soll in der Lage sein, die Geschwindigkeit der Objekte und damit die zeitliche Abhängigkeit der dynamischen Umgebung bei der Wegplanung zu berücksichtigen. In den vorangegangenen Abschnitten wurde bereits geschildert, wie die Umgebung mit Hilfe von Sensoren erfasst werden kann. Weiterhin wurden Methoden aufgezeigt, um die eigene Position während einer Bewegung auch bei unsicherer Sensorinformation zu verfolgen. Im Folgenden wird daher angenommen, dass Informationen über die eigene Position zur Verfügung stehen, dass die statische Umgebung mit hinreichender Genauigkeit bekannt ist und dass auch der dynamische Teil der Umgebung erfasst wird. Zu beachten ist, dass die vollständige Information nicht a priori bekannt ist, sondern dass insbesondere die Informationen zu den dynamischen Hindernissen erst zur Laufzeit eintreffen. Im Folgenden werden zunächst die Grundlagen der Bahnplanung aufgezeigt. Daran schließt sich die Darstellung der Anforderungen an eine reaktive Bahnplanung an. Aus diesen Grundlagen wird eine spezifische Implementierung abgeleitet und mit der experimentellen Erprobung dargestellt.



a) b)

Bild 8.1: Kartenrepräsentationen: a) Durch die Zellzerlegung wird der Konfigurationsraum in freie Zellen aufgeteilt. Die Verbindung der Mittelpunkte der Zellgrenzen bietet eine Bahn, die gute Abstandseigenschaften zu Hindernissen aufweist. b) Die Straßenkarte beinhaltet sowohl semantische Informationen, als auch geometrische Informationen.

8.1 Bahnplanungsverfahren

Die in der Literatur bekannten Bahnplanungsverfahren lassen sich in die Kategorien *Zellzerlegung*, *Straßenkarten* und *Potentialfeldmethoden* einteilen [Latombe, 1999]. Weiterhin arbeiten die Verfahren üblicherweise im Konfigurationsraum eines Roboters, der über alle möglichen unterschiedlichen Stellungen (Konfigurationen) der Freiheitsgrade eines Systems aufgespannt wird. Die Zellzerlegung basiert auf der Teilung des Konfigurationsraums in eine endliche Zahl von Abschnitten, in denen sich jeweils mit einfachen Mitteln eine kollisionsfreie Bahn finden lässt (Bild 8.1a). Das Bahnplanungsproblem wird damit übertragen auf das Problem, eine Folge benachbarter Zellen zu finden, die auch Start und Ziel beinhalten [Lozano-Perez, 1981; Preparata und Shamos, 1985].

Bei den Straßenkarten wird ein Netzwerk von kollisionsfrei verbundenen Bahnen konstruiert, das den freien Konfigurationsraum aufspannt (Bild 8.1b). Dieser stellt eine durch Kollisionsfreiheit gekennzeichnete Untermenge des Konfigurationsraums dar. Das Bahnplanungsproblem beinhaltet eine Bahn in diesem Netz zu finden, die Start- und Zielkonfiguration verbindet, um dann diese Folge von Abschnitten abzufahren. Für die Konstruktion solcher Straßenkarten stehen verschiedene Verfahren zur Verfügung, z. B. Sichtbarkeitsgraphen und Voronoi-Diagramme [Lozano-Perez, 1979; Aurenhammer, 1991; Latombe, 1991].

Bei den oben genannten Verfahren erfolgt jeweils eine Diskretisierung des Arbeitsraums, aus der eine topologische Struktur generiert wird. Diese kann als Graph dargestellt werden und führt das Planungsverfahren nach dem Vorverarbeitungs-

schritt auf eine Graphensuche zurück.

Bei den Potentialfeldmethoden wird eine kollisionsfreie Trajektorie dadurch erzeugt, dass der Roboter durch virtuelle Kräfte, hervorgerufen durch den negativen Gradienten eines Potentials, vom Zielpunkt angezogen und von Hindernissen abgestoßen wird [Khatib, 1985; Hwang und Ahuja, 1992; Barraquand u. a., 1992]. Von den zuvor genannten Algorithmen grenzen sich die Potentialfeldmethoden vor allem dadurch ab, dass diese neben der Trajektorie bereits eine Stellgröße mitliefern, um den Roboter vom Start zum Ziel zu bringen. Damit sind die Potentialfeldansätze in den Bereich der *Feedback-Methoden* einzuordnen. Das Problem bei den Potentialfeldmethoden stellen lokale Minima dar, in denen der Roboter „stecken bleibt". Potentialfunktionen ohne lokale Minima heißen Navigationsfunktion. Die Erstellung der Navigationsfunktion für den allgemeinen Fall ist jedoch hoch komplex.

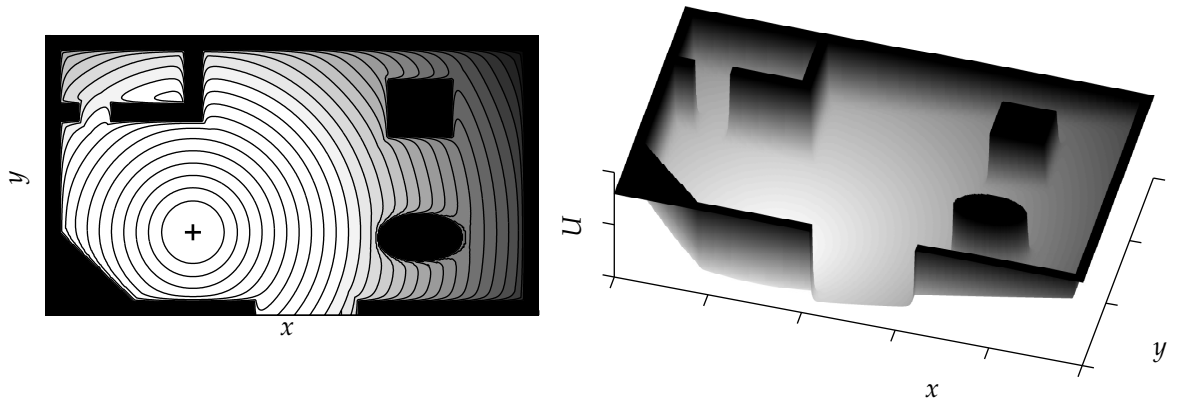


Bild 8.2: Der als Punktmasse modellierte Roboter bewegt sich durch die Einwirkung virtueller Kräfte im Konfigurationsraum. Die abstoßenden Kräfte der Hindernisse überlagern sich dabei mit den anziehenden Kräften des Zielpunkts.

Die oben genannten Verfahren dienen der reinen Bahnplanung für kinematische, holonome Systeme. Insbesondere für reaktive Systeme und für allgemeine Kinematiken müssen zusätzliche Randbedingungen in die Bahnplanung einfließen. Diese beinhalten zum einen die Dynamik sowie nichtholonome Zwangsbedingungen. Mittels geeigneter Regelung lassen sich Robotersysteme dazu bringen, näherungsweise einer rein kinematisch geplanten Bahn zu folgen. Insbesondere die real vorhandenen Stellgrößenbeschränkungen können jedoch zu erheblichen Bahnabweichungen führen, so dass die kollisionsfrei geplante Bahn durch den Regelfehler ge-

gebenenfalls dennoch in einer Kollision mündet. Wird eine mit obigen Methoden geplante Bahn in einem nachfolgenden Schritt an die dynamischen und nichtholonomen Randbedingungen angepasst, so besteht die Schwierigkeit darin, weiterhin Kollisionsfreiheit sicherzustellen.

Erstrebenswert ist demnach ein Planungsverfahren, welches Randbedingungen originär berücksichtigt, so dass das Zielsystem der generierten Trajektorie folgen kann. Derartige Verfahren werden als *kinodynamische Bahnplanung* bezeichnet [Donald u. a., 1993]. Da sich dynamische und nichtholonome Zwangsbedingungen nicht als verbotene Bereiche in den Zustandsraum eines Roboters abbilden lassen, können die oben genannten Verfahren nicht direkt angewendet werden. Ein Lösungsansatz besteht in der Formulierung des Problems in Form der optimalen Regelung [Athans und Falb, 1966; Bryson und Ho, 1975]. Für diesen wie auch für andere Ansätze gilt jedoch, dass die deterministische und vollständige Lösung des Problems rechnerisch sehr aufwändig ist.

Bei Beschränkung auf eine nicht vollständige Lösung ist eine geeignete Heuristik erforderlich, mit der eine spezielle Lösung des Problems gewonnen wird. Diese sollte zumindest in der Nähe der optimalen Lösung liegen. Vor dem Hintergrund, dass der Umgebungszustand ohnehin nur näherungsweise erfasst werden kann, und durch die nichtdeterministische Natur der dynamischen Umgebung kann in der Regel auf die aufwändige Bestimmung der optimalen Gesamtlösung zugunsten effizienterer und ressourcenschonender Verfahren verzichtet werden.

Eine neue Klasse von Bahnplanungsverfahren wurde mit den wahrscheinlichkeitsbasierten Straßenkarten – englisch: *Probabilistic RoadMaps* (PRM) – von Barraquand und Latombe [1990] eingeführt, von Overmars [1992] und Kavraki u. a. [1996a] aufgegriffen und in zahlreichen Arbeiten erweitert [Overmars und Svestka, 1996; Kavraki u. a., 1996b, 1998; Hsu u. a., 1998, 1999]. Die PRMs stellen schnelle und effiziente Algorithmen für eine geometrische Bahnplanung dar und sind auf mehrfache Suchanfragen ausgerichtet. Auf Grundlage einer Umgebungskarte wird ein Vorverarbeitungsschritt (offline) durchgeführt, in dem ein Graph gültiger Bahnen im gesamten Konfigurationsraum – die Straßenkarte – erstellt wird. Die Festlegung der Meilensteine bzw. Knoten des Graphen erfolgt in der Regel durch das Ziehen zufälliger Stichproben aus dem Konfigurationsraum, die dann nach Prüfung auf Kollisionsfreiheit mit dem Graphen verbunden werden. Bei der Anwendung (online) wird nun zunächst eine Verbindung vom Ausgangspunkt zur Straßenkarte, sowie eine Verbindung von der Straßenkarte zum Zielpunkt hergestellt. Anschließend erfolgt die Graphensuche um eine Verbindung zwischen Start und Ziel herzustellen. Dem Vorteil der Effizienz und der einfachen Implementierung stehen einige Nachteile gegenüber. Der offline durchgeführte Vorverarbeitungsschritt erfordert, dass die Umgebung bereits a priori bekannt ist. Dieses ist jedoch innerhalb

dynamischer Umgebungen nicht der Fall. Zusätzlich handelt es sich um eine rein geometrische Bahnplanung und die Dynamik des Roboters ist nicht berücksichtigt. Insbesondere der letzte Punkt stellt ein Problem dar, da die Meilensteine durch einen Zufallsprozess entstehen und die Bahn durch viele Richtungsänderungen (Knicke in der Bahn) charakterisiert ist.

Die Berücksichtigung der Dynamik sowie der dynamischen Umgebung des Roboters gelingt LaValle [1998] mit der Einführung der schnell explorierenden Zufallsbäume – englisch: *Rapidly-Exploring Random Trees* (RRT) [LaValle, 1998; LaValle und Kuffner, 1999; Kuffner und LaValle, 2000]. Mittels RRTs wird online ein Baum[1] kollisionsfreier Trajektorien aufgebaut, indem die Kanten des Baums in Richtung zufällig gewählter Punkte erweitert werden. Ein wesentlicher Unterschied zu den PRMs ist, dass der Konfigurationsraum dort in einer Vorverarbeitungsphase grundlegend untersucht wird um, ausgelegt für die mehrfache Suche, ein engmaschig verbundenes Netz zu knüpfen. Demgegenüber wird hier versucht, sich mit großen Schritten dem Ziel zu nähern. Dabei wird der Baum nur für die jeweils aktuelle Suche genutzt, es ist keine Vorverarbeitungsphase erforderlich und der Algorithmus ist damit echtzeitfähig. Die Berücksichtigung der Dynamik erfolgt durch die Methode der Baumerweiterung. Ausgehend vom nächstliegenden Knoten des Baums zu einer zufällig gewählten Konfiguration wird eine gültige Stellgröße bestimmt. Diese bringt den Folgezustand in die Nähe der zufällig gewählten Konfiguration. Die durch Aufschalten der Stellgröße erreichte Konfiguration wird dem Suchbaum hinzugefügt.

8.2 Reaktive Bahnplanung

Aufbauend auf dem Konzept der schnell explorierenden Suchbäume wird in der vorliegenden Arbeit ein Bahnplanungsalgorithmus erstellt, der nichtholonome und dynamische Zwangsbedingungen mit Reaktivität vereint. Voraussetzung dafür sind die Kenntnis über die statische Umgebung (Umgebungskarte), die Bestimmung des eigenen Aufenthaltsorts (Lokalisation) in der statischen Umgebung, die Erfassung dynamischer Objekte im Umfeld des Roboters (Objektverfolgung) sowie eine Bahnregelung, die den Roboter auf einer vorgegebenen Bahn bewegt (Pilot-Funktion).

Das Ziel der Bahnplanung ist der effiziente Aufbau eines topologischen Graphen, der als Knoten den Zustand des Roboters inkl. der Meilensteine enthält und als

[1] Ein Baum ist ein gerichteter Graph ohne Schleifen, in dem alle Knoten eine oder mehrere Kanten haben die auf einen weiteren Knoten, den Kind-Knoten, führen. Ein Kind-Knoten kann weitere Kinder haben. Alle Knoten mit Ausnahme der Wurzel haben eine eingehende Kante, die vom Eltern-Knoten kommt. Der Wurzelknoten hat keinen Elternknoten.

Kanten die Eingangsgrößen einer Bahnregelung hat, die den Roboter aus dem Startknoten in den Zielknoten überführen. Der Graph wird inkrementell aufgebaut, bis der Zielpunkt erstmalig erreicht wird. Im Anschluss daran wird weiterhin der Weg zum Ziel optimiert. Die Diskretisierung des kontinuierlichen Problems erfolgt durch eine endliche Auswahl von Bewegungszuständen, die mittels Bewegungsprimitiven verbunden werden können. Voraussetzung für die Anwendung ist die Existenz eines Regelungsgesetzes, welches den Roboter in hindernisfreier Umgebung aus einem gegebenen Zustand in den Folgezustand überführt. Dabei wird das Ergebnis über die Auswahl an Bewegungszuständen als optimal bezüglich des Optimierungskriteriums für die Bahnplanung angenommen. Mit Hilfe des Regelungsgesetzes und einer Simulation des Systems können durch Anwendung des Optimierungskriteriums die Kosten (cost-to-go) für eine Kante des Suchgraphen bestimmt werden. Die Kosten dienen neben der Steuerung der Explorationsrichtung auch der Optimierung einer vorhandenen Lösung.

8.2.1 Systembeschreibung

Der Zustandsraum des Roboters $\mathcal{X}$ setzt sich aus dem Produkt

$$\mathcal{X} = \mathcal{C} \times \mathcal{Y} \tag{8.1}$$

zusammen, so dass $\mathcal{C}$ invariant bezüglich der Dynamik ist. Der Raum $\mathcal{C}$ ist damit ein reduzierter Konfigurationsraum. Der Raum $\mathcal{Y}$ beinhaltet die verbleibenden Variablen der Konfiguration, sowie Geschwindigkeiten und höhere Ableitungen. Unter Berücksichtigung der Stellgrößen $\mathcal{U}$ kann durch $\mathcal{Y} \times \mathcal{U}$ eine Menge von Trajektorien ausgedrückt werden. Durch die Reduktion dieser zunächst unbegrenzten Menge auf eine begrenzte Auswahl

$$\mathcal{Q}_\mathrm{T} \subset \mathcal{Y} \times \mathcal{U} \quad , \tag{8.2}$$

ergibt sich die Menge der möglichen Trajektorien aus

$$\mathcal{M} = \mathcal{C} \times \mathcal{Q}_\mathrm{T} \quad . \tag{8.3}$$

Eine geeignete Auswahl der Trajektorien kann sicherstellen, dass die dynamischen Möglichkeiten des Roboters ausgenutzt werden und dass die resultierenden Trajektorien über gewünschte Eigenschaften verfügen.

Vorausgesetzt wird ein Bahnregelungssystem, dass in der Lage ist, den Roboter aus einem beliebigen Zustand $\boldsymbol{x}$ in einen Zustand $\boldsymbol{x}_\mathrm{target} \in \mathcal{T} = \mathcal{C} \times \mathcal{Y}$ zu überführen. Für spezielle Systeme gibt es dafür analytische Lösungen. Im allgemeinen ist

eine numerische Lösung mittels Minimierung eines Kostenfunktionals der Form

$$J(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)) = \int_{t_0}^{t_f} \gamma(\boldsymbol{x}_t, \boldsymbol{u}_t) \mathrm{d}t \quad \text{mit } \boldsymbol{x}_{t_f} \in \mathcal{T}(\boldsymbol{x}_{\text{target}}) \tag{8.4}$$

möglich. Die darin enthaltenen inkrementellen Kosten $\gamma(\boldsymbol{x}_t, \boldsymbol{u}_t)$ sind invariant bezüglich Änderungen in $\mathcal{C}$ und enthalten zum Beispiel die Bahnlänge, die Ausführungszeit oder den Aufwand. In Verbindung mit der o. g. Reduktion des Konfigurationsraums auf Bewegungsprimitive können bei komplexer Systemdynamik bzw. aufwändigen Optimierungskriterien die optimalen Kosten (cost-to-go) $J^*(\boldsymbol{x}, \boldsymbol{x}_{\text{target}})$ für alle $\boldsymbol{x} \in \mathcal{X}$ und alle $\boldsymbol{x}_{\text{target}} \in \mathcal{T}$ vorab berechnet und in einem Lookup-Table hinterlegt werden. Alternativ steht bei einfachen Systemen bzw. bei Beschränkung auf Zustände, die auf einfache Zustandsübergänge führen, eine analytische Berechnung der Kosten zur Verfügung, die zur Laufzeit effizient durchgeführt werden kann.

Bei bekannten Kosten $J^*(\boldsymbol{x}, \boldsymbol{x}_{\text{target}})$ für die optimale Verbindung zweier Konfigurationen, kann daraus das optimale Regelungsgesetz $\pi : \mathcal{X} \times \mathcal{C} \rightarrow \mathcal{U}$ ermittelt werden, das die Kosten für den Weg zum Ziel minimiert. Damit stellt das Regelungsgesetz π eine Funktion des Zustands $\boldsymbol{x}$ dar, die durch das Ziel $\boldsymbol{x}_{\text{target}}$ parametriert wird. Die Lösung des optimalen Regelungsproblems im hindernisfreien Raum liefert zusätzlich ein Maß für den Abstand zwischen der Ausgangskonfiguration und dem Ziel.

Bei zusätzlicher Berücksichtigung der durch statische und dynamische Hindernisse geprägten Umgebung ist es erforderlich, die Bahn auf Kollisionen zu überprüfen. Wie bereits oben erwähnt wird angenommen, dass für den Planungsprozess Lage und Geschwindigkeit von Hindernissen für einen begrenzten Planungshorizont bekannt sind. Hindernisse werden durch Zwangsbedingungen in Form von Ungleichungen

$$\boldsymbol{G}(\boldsymbol{x}, t) \leq 0 \tag{8.5}$$

formuliert. Wegen der zeitvarianten Natur der dynamischen Objekte muss die Kollisionsprüfung für Orts-/Zeitpaare $(\boldsymbol{x}, t) \in \mathcal{X} \times \mathbb{R}$ erfolgen. Die zeitvarianten Hindernisse stellen damit besondere Anforderungen an die Echtzeitfähigkeit der Bahnplanung. Da die Berechnung einer Bahn stets endliche Zeit erfordert, muss die Planung für einen Zeitpunkt (und Zustand) in der Zukunft erfolgen, damit das Ergebnis zur Ausführungszeit bekannt ist.

Durch die Berücksichtigung von Hindernissen wird der freie Raum begrenzt auf einen erreichbaren Raum

$$\mathcal{F} \subset \mathcal{X} \times \mathbb{R} \quad , \tag{8.6}$$

der alle Orts-/Zeitpaare $(\boldsymbol{x}, t)$ enthält für die keine Kollision auftritt. Bei gegebenem Zustand $(\boldsymbol{x}_0, t_0)$ ist ein Zustand $(\boldsymbol{x}_f, t_f)$ genau dann erreichbar, wenn sich eine Stellfolge finden lässt, die den Roboter mittels einer kollisionsfreien Trajektorie in den Zielzustand überführt. Die Menge der erreichbaren Ziele kann als $\mathcal{R}(\boldsymbol{x}_0, t_0) \subset \mathcal{F}$, bzw. mit $\mathcal{S} \subset \mathcal{F}$ als

$$\mathcal{R}(\mathcal{S}) = \bigcup_{(\boldsymbol{x},t)\in\mathcal{S}} \mathcal{R}(\boldsymbol{x}, t) \tag{8.7}$$

definiert werden.

Die oben genannten Definitionen vorausgeschickt, lässt sich das Bahnplanungsproblem wie folgt beschreiben: bei gegebenem Zustand $\boldsymbol{x}_0 \in \mathcal{X}$ zur Zeit t_0 und der Zielkonfiguration $\boldsymbol{x}_f \in \mathcal{C} \times \mathcal{Y}$ wird eine Stellfolge gesucht, die den Roboter von $\boldsymbol{x}_0$ nach $\boldsymbol{x}_f$ überführt. Als Nebenbedingung soll ein Kostenfunktional in Form von Gl. (8.4) minimiert werden.

8.2.2 Lösungsansatz

Die Bahnplanung in der Umgebung mit dynamischen Hindernissen basiert auf der Generierung einer Folge zufälliger Zustände $\boldsymbol{x}_{\text{random}}$ im Konfigurationsraum mit dem zugehörigen Regelungsgesetz $\pi(\cdot, \boldsymbol{x}_{\text{random}})$, das den Roboter unter Vermeidung von Hindernissen in den Zielzustand überführt. Durch die Nutzung der Lösung des optimalen Regelungsproblems im hindernisfreien Raum als virtuelle Eingangsgröße eines unterlagerten Systems wird eine Entkopplung zwischen den Regelungsebenen erreicht. Die Stellgrößen, die unmittelbar auf die Aktoren des Systems wirken, werden dabei aus der virtuellen Eingangsgröße abgeleitet. Der Ausgang der Bahnplanung ist also ein Regelungsgesetz und keine Folge von Sollzuständen oder Stellgrößen. Durch diese Entkopplung kann der Bahnplanungsalgorithmus mit einer anderen Frequenz betrieben werden als der unterlagerte Regelkreis.

Ausgehend von den Anfangsbedingungen $(\boldsymbol{x}_i, t_i)$ wird inkrementell ein Suchbaum kollisionsfreier Trajektorien aufgebaut, der in effizienter Weise den erreichbaren Raum $\mathcal{R}(\boldsymbol{x}_i, t_i)$ exploriert. Dazu wird der Baum in jedem Schritt um eine neue Kante und einen neuen Knoten erweitert. Entscheidend für die Leistungsfähigkeit ist die Frage, welcher Knoten des bestehenden Baums in welche Richtung erweitert werden soll.

LaValle und Kuffner [1999] lösen diese Fragestellung mit den RRTs durch die Wahl einer zufälligen Konfiguration $\boldsymbol{x}_{\text{random}}$ und die Ermittlung von deren nach der euklidischen Distanz nächstem Nachbarknoten im Suchbaum. Ausgehend von diesem Knoten wird für einen Zeitraum Δt eine Stellgröße aufgeschaltet, so dass

die resultierende Roboterkonfiguration möglichst nah an der zufällig gewählten liegt. Resultiert daraus eine kollisionsfreie Trajektorie, wird der Baum erweitert.

Hsu u. a. [2000] bieten eine alternative Lösung an. Gewählt wird ein zu erweiternder Knoten aus dem Baum. Dieser Knoten wird mit einer zufällig gewählten Stellgröße beaufschlagt. Die resultierende Konfiguration wird dem Baum hinzugefügt.

Ein zentrales Element dieser Arbeit ist der im Folgenden dargestellte alternative Ansatz. Nach Wahl einer zufälligen Konfiguration erfolgt der Versuch, alle Knoten des Baums in der Reihenfolge steigender Kosten gemäß $J^*(\boldsymbol{x}_i, \boldsymbol{x}_{\text{random}})$ mit der zufällig gewählten Konfiguration zu verbinden. Bei Erfolg wird die Konfiguration dem Suchbaum hinzugefügt. Der wesentliche Unterschied zu den vorgenannten Verfahren ist die Prüfung der Verbindung der Erweiterungskonfiguration mit allen Knoten, statt nur den nächsten Nachbarn zu erweitern. Der wichtigere Unterschied ist jedoch, dass die Kosten zur Erreichung des Knoten als Maß für den Abstand genommen werden.

8.2.3 Datenstrukturen

Vor der detaillierten Darstellung des Algorithmus werden die für den Suchbaum relevanten Datenstrukturen der Knoten und Kanten erläutert.

Daten der Knoten

In den Knoten ist die wichtigste Information der Zustand und die zugehörige Zeit. Diese Information ergibt sich durch Integration der Systemdynamik bei gegebenen virtuellen Stellgrößen ausgehend von der Wurzel über die Kanten des Baums. Zusätzlich werden Schätzungen über die Kosten der Trajektorie abgespeichert die den Knoten einschließen. Hier können zum Einen die akkumulierten Kosten abgeschätzt werden, die bis zu diesem Knoten auftreten. Diese Information lässt sich mittels Integration über die Kosten der Kanten bestimmen. Zum Anderen können Abschätzungen über eine obere und eine untere Schranke für die Kosten vom Knoten bis zum Ziel angegeben werden. Die Kosten für die untere Schranke werden durch die Lösung des optimalen Regelungsproblems in hindernisfreier Umgebung bestimmt. Da Hindernisse zusätzliche Zwangsbedingungen darstellen, können die Kosten unter deren Berücksichtigung nur größer sein. Stellt sich die im hindernisfreien Raum geplante Trajektorie auch in der Umgebung mit Hindernissen als kollisionsfrei heraus, so handelt es sich um eine optimale Trajektorie. Die obere Schranke für die Kosten zum Ziel ist a priori unbekannt, sowie unbekannt ist, ob überhaupt eine Lösung existiert. Sie wird demnach stets mit dem symbolischen Wert *unendlich* initialisiert. Sobald ein kollisionsfreier Weg zum Ziel ermittelt wur-

de, können die Kosten für diese Trajektorie als obere Schranke angesehen werden. Für eine effiziente Implementierung sind zusätzlich die Zahl der abgehenden Kanten, sowie Verweise auf den Elternknoten und die Kindknoten erforderlich.

Daten der Kanten

Während die Knoten den Zustand des Roboters enthalten, werden durch die Kanten die Entscheidungen des Roboters repräsentiert. Wichtigstes Datum ist damit der Zielknoten und das Regelungsgesetz, um zum Zielknoten zu gelangen. Für die Analyse der Kosten entlang des Baums werden die durch die Kante verursachten inkrementellen Kosten gespeichert.

8.2.4 Algorithmus

Im Folgenden wird die Funktionsweise des Bahnplanungsalgorithmus detailliert dargestellt. Zunächst erfolgt die Initialisierung des Baums. Anschließend findet die Exploration des Arbeitsraums statt. Die Suche erfolgt in einer Schleife mit vorgegebener Rechenzeit. Innerhalb der Schleife wird zunächst der Baum aus der vorhergehenden Iteration auf Kollisionen mit neu erfassten Hindernissen überprüft. Anschließend erfolgt eine iterative Erweiterung des Baums. Nach jeder erfolgreichen Erweiterung werden die Kosten in der Baumstruktur aktualisiert. Gegebenenfalls erfolgt ein Ausdünnen des Baums. Dies beinhaltet die Entfernung von Lösungen, die nicht den vorgegebenen Effizienzkriterien genügen. Nach Ablauf der Rechenzeit erfolgt die Auswahl der besten zur Verfügung stehenden Lösung.

Initialisierung

In der Initialisierung erfolgt die Festlegung der Baumwurzel. Diese beinhaltet den um die erwartete Zeitdauer der Bahnberechnung in die Zukunft projizierten Zustand des Roboters. Die akkumulierten Kosten an der Wurzel betragen null. Aus dem Regelungsgesetz für die hindernisfreie Umgebung zwischen Wurzelzustand und Zielzustand resultiert die untere Schranke für die Kosten zum Ziel. Die Festlegung der oberen Schranke beinhaltet die Kollisionsprüfung für diese direkte Verbindung zum Ziel. Bei erfolgreicher Prüfung sind obere und untere Schranke identisch und die optimale Lösung ist bereits gefunden. Da kein Verbesserungspotential besteht kann die Suche beendet werden. In der Regel ist das Prüfungsergebnis negativ. Eine Aussage über die obere Schranke ist damit nicht möglich. Sie wird mit dem symbolischen Wert *unendlich* initialisiert und es folgt die Einleitung des Iterationsprozesses.

Initiale Kollisionsprüfung

In aufeinander folgenden Iterationen des Bahnplanungsalgorithmus ist stets ein neuer Informationsstand bezüglich der dynamischen Hindernisse zu berücksichtigen. Vor Beginn der Iterationsschleife erfolgt daher die Prüfung, ob Elemente des bestehenden Suchbaums aus der vorherigen Iteration in Kollision mit einem der neu erfassten Hindernisse stehen. Bei Feststellung einer Kollision werden das betreffende Element sowie alle diesem nachfolgenden Elemente gelöscht.

Erweiterung des Suchbaums

Ziel ist der Aufbau eines Suchbaums kollisionsfreier Trajektorien. Die Erweiterung des Suchbaums erfolgt zunächst im Hinblick auf eine rasche Ausdehnung des explorierten Gebiets, um eine Annäherung an die Zielkonfiguration zu erzielen. Dabei wird zunächst zufällig ein Element des Konfigurationsraums gezogen. Die Erreichbarkeit lässt sich durch die Prüfung der Verbindung zu jedem Knoten des Suchbaums auf Kollisionsfreiheit ermitteln. Bei nicht erfolgreicher Prüfung ist die zufällig gewählte Konfiguration außerhalb des erreichbaren Raums und wird verworfen. Andernfalls erfolgt die Erweiterung des Baums um eine neue Kante und einen neuen Knoten.

Für die Reihenfolge, in der die einzelnen Knoten des Baums auf eine Verbindungsmöglichkeit mit dem zufällig gewählten Knoten geprüft werden, bestehen verschiedene Möglichkeiten. Die zufällige Wahl der Reihenfolge schränkt die Möglichkeit zur Einflussnahme auf die Explorationsrichtung ein. Effizienter arbeitet der Algorithmus durch die zwei im Folgenden dargestellten Heuristiken.

Solange keine Verbindung mit dem Zielzustand ermittelt ist, liegt die Priorität vor allem auf einer möglichst raschen Erweiterung des Suchgebiets in Richtung wenig explorierter Bereiche. Für die auf Zufallsprozessen basierten Bahnplanungsverfahren gilt, dass ein erheblicher Teil der Rechenzeit auf Routinen zur Kollisionsprüfung entfällt. Kann der Anteil der Kollisionsprüfungen reduziert werden, so erhöht sich gleichermaßen die Leistungsfähigkeit des Algorithmus. Da die Wahrscheinlichkeit für eine kollisionsfreie Verbindung zweier nahe aneinander liegender Knoten größer ist als die von weit entfernten Knoten, werden für die Explorationsheuristik die Knoten des Baums nach fallender Distanz zum Erweiterungsknoten sortiert. Als Maß für die Distanz wird hier wiederum das Optimierungskriterium aus Gl. (8.4) bemüht. Der Erweiterungsvorgang mit Explorationsheuristik ist in Bild 8.3a dargestellt. Der Suchbaum soll um die zufällige Konfiguration x_{random} erweitert werden. Die Verbindung zum nächstliegenden Knoten 1 führt auf eine Kollision. Auch die Verbindung zu Knoten 2 ist nicht möglich. Erst der etwas weiter entfernt liegende Knoten 3 lässt sich kollisionsfrei mit dem Baum verbinden und wird diesem hinzugefügt. Die Prüfung der Erreichbarkeit ergibt hier, dass eine direkte Verbindung zur

Zielkonfiguration möglich ist, somit ist eine Lösung für das Bahnplanungsproblem ermittelt.

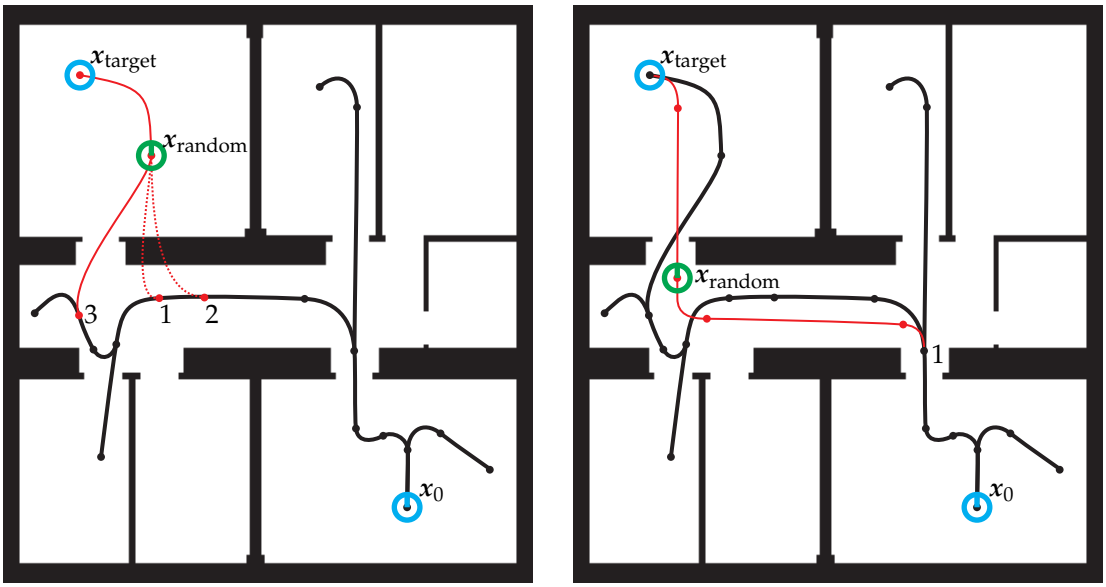


Bild 8.3: Dargestellt ist der Vorgang der Suchbaumerweiterung. a) zeigt die Explorationsheuristik. b) stellt die Optimierungsheuristik dar.

Sobald eine kollisionsfreie Trajektorie zwischen Ausgangskonfiguration und Zielkonfiguration verfügbar ist, verschiebt sich das Bahnplanungsziel von der Exploration zur Optimierung der Lösung. Eine optimale Lösung kann zwar durch die Reduktion des kontinuierlichen Lösungsraums auf eine begrenzte Anzahl von Bewegungsprimitiven nicht gefunden werden. Es kann jedoch eine Annäherung an die optimale Lösung erfolgen, indem versucht wird, die Kosten der Trajektorie nach Gl. (8.4) zu minimieren. Die Knoten des Baums werden nun nach den akkumulierten Kosten zur Erreichung der Zufallskonfiguration sortiert. Diese bestehen aus den in den Knoten gespeicherten akkumulierten Kosten, zzgl. der Kosten die durch eine Kante zwischen dem Knoten des Baums und dem Erweiterungsknoten entstehen. Auf diese Weise werden Knoten mit geringen Gesamtkosten hinzugefügt. Dieser Vorgang ist in Bild 8.3b dargestellt. Die zufällig gewählte Konfiguration x_{random} soll dem Baum hinzugefügt werden. Die niedrigsten Gesamtkosten ergeben sich bei Verbindung mit dem Knoten 1, die auf eine kürzere Strecke von der Ausgangskonfiguration x_0 bis zur Erweiterungskonfiguration x_{random} führt. Von der Erweiterungskonfiguration ist eine direkte Verbindung zur Zielkonfiguration x_{target} möglich, so dass die bisherige Lösung durch eine bessere Lösung ersetzt werden

kann.

Neben den oben beschriebenen Heuristiken zur Steigerung der Effizienz des Algorithmus liegt zusätzliches Verbesserungspotential in der Ausnutzung der Kenntnis der statischen Umgebung. Die Anwendung einer einfachen Abstandsmetrik stellt ein Maß für die Nähe einer gewählten Konfiguration zum Ziel bereit. Dies lässt sich dahingehend ausnutzen, dass bei der Ziehung einer zufälligen Konfiguration anstatt einer gleichen Wahrscheinlichkeit über den Zustandsraum solche begünstigt werden, die sich bezüglich des Abstands zwischen Ausgangskonfiguration und Ziel befinden. Zweitens lässt sich auch die Orientierung der Konfiguration an Stelle einer Gleichverteilung durch eine Ausrichtung in Richtung des Ziels begünstigen.

Wie bereits erwähnt, stellt die Kollisionsprüfung den rechenintensivsten Teil der Bahnplanung dar. Jede Suchbaumerweiterung erhöht damit die erforderliche Rechenzeit. Sobald sich eine kollisionsfreie Erweiterung des Baums ergibt, können jedoch durch die Unterteilung der hinzugefügten Trajektorie zusätzliche Knoten generiert werden. Für diese ist eine Kollisionsprüfung nicht erforderlich, da sie auf einer bereits geprüften Trajektorie liegen. Sie dienen einer gesteigerten Zahl von Anknüpfungspunkten für folgende Baumerweiterungen und begünstigen die Exploration in engen Passagen und das Auffinden von kostenoptimaleren Lösungen.

Ein Gerüst für die Erweiterung des Baums ist in Algorithmus 8.1 dargestellt.

Algorithmus 8.1 Algorithmus zur Erweiterung des Suchbaums und Überprüfung auf Kollisionsfreiheit.

```
1:  ExpandTree(tree)
2:      x_textrandom =SampleRandomConfiguration
3:      sort nodes for [random order | incremental costs | total costs]
4:      for all nodes in tree do
5:          traj = GenerateTrajectory(node.x, x_random)
6:          if CollisionCheck(traj) = collision free then
7:              return traj
8:          end if
9:      end for
10:     return error
11: end
```

Aktualisierung der Kosten

Für jeden hinzugefügten Knoten wird geprüft, ob von diesem die Zielkonfiguration direkt erreichbar ist. Bei Erfolg steht eine neue kollisionsfreie Trajektorie mit endlichen Kosten zur Verfügung. Das Problem der Bahnplanung wird damit durch

eine (suboptimale) Trajektorie gelöst. In der Folge wird das Planungsziel von der Exploration der Umgebung auf die Optimierung der gefundenen Lösung verschoben und in der Baumerweiterung kommt die Optimierungsheuristik zum Einsatz. Diese greift auf die Kosten der Trajektorie zu. Für jede kollisionsfreie Lösung lässt sich mit den akkumulierten Kosten eine obere Schranke für das Bahnplanungsproblem bestimmen. Diese betreffen neben dem Erweiterungsknoten alle Elternknoten auf dem Pfad bis zur Wurzel des Baums. Um die Kosten zu aktualisieren, wird der Baum rückwärts entlang des Pfads vom Ziel bis zur Wurzel durchlaufen. Der Vergleich der Kosten der neuen Lösung mit denen der alten Lösung zeigt das Verbesserungspotential auf. Bei einer Kostenreduktion wird der entsprechende Kostenwert ersetzt. Andernfalls existiert für den betreffenden Knoten ein Unterbaum, der eine bezüglich des Optimierungskriteriums bessere Lösung darstellt. Dieses führt auf den Abbruch des Durchlaufs. Die Funktion zur Aktualisierung der Kosten ist in Algorithmus 8.2 dargestellt.

Algorithmus 8.2 Algorithmus zur Aktualisierung der Kosten.

```
1:  UpdateCostEstimates(Tree, node, target)
2:      node.LowerBound = J*(node.x, target.x)
3:      traj = GenerateTrajectory(node.x, target.x)
4:      if CollisionCheck(traj) = collision free then
5:          node.UppBound = node.LowBound
6:          while node ≠ root do
7:              newBound = node.UppBound + node.IncomingEdge.Cost
8:              if node.Parent.UppBound > newBound then
9:                  node.Parent.UppBound = newBound
10:                 GoTo(node.Parent)
11:             else
12:                 break
13:             end if
14:         end while
15:         return success
16:     else
17:         node.UppBound = ∞
18:         return error
19:     end if
20: end
```

Ausdünnen des Baums

Durch die Erweiterung des Baums wächst die Zahl der Knoten und Kanten stetig an. Stehen mehrere Lösungen für das Bahnplanungsproblem zur Verfügung, so sind die akkumulierten Kosten der einzelnen Lösung unterschiedlich und es wird die kosteneffizienteste Lösung angestrebt. Während der Aktualisierung der Kosten können im Hinblick auf die anderen Kinder eines Elternknoten solche Unterbäume entfernt werden, deren untere Kostenschranke einen höheren Wert aufweist als die obere Kostenschranke des gerade zu aktualisierenden Knotens, da dort keine bessere Lösung resultieren kann. Der Baum wird damit um alle die Knoten und Kanten bereinigt, die keinen Beitrag zu einer „optimaleren“ Lösung leisten können. Dadurch lassen sich Ressourcen bezüglich Speicherkapazität und Rechenzeit einsparen.

Neben dieser innerhalb einer Iteration erfolgenden Bereinigung, werden nach Verlassen der Iterationsschleife und Auswahl der Bahn alle ungenutzten Knoten und Kanten verworfen. Dies begründet sich vornehmlich in der Notwendigkeit, vor Beginn der Iterationsschleife alle Elemente des Suchbaums aus der vorherigen Iteration auf Kollision mit den neu erfassten dynamischen Objekten prüfen zu müssen. Da die Kollisionsprüfung den ressourcenaufwändigsten Teil des Bahnplanungssystems darstellt, ist es vorteilhaft, ausgehend von der gewählten Bahn aus der vorherigen Iteration neu zu explorieren bzw. zu optimieren.

8.3 Implementierung

Der Ablauf lässt sich zusammenfassend anhand von Algorithmus 8.3 beschreiben. Nach Initialisierung des Baums erfolgt die iterative Berechnung der Bahn bis zur Erreichung des Ziels. Zunächst wird versucht, die Zielkonfiguration von der Ausgangskonfiguration direkt zu erreichen. Ist diese Lösung möglich, so kann der Algorithmus beendet werden, da eine gemäß der Definition optimale Lösung ermittelt wurde. Andernfalls beginnt die Ausführung der Baumerweiterung in einer inneren Schleife. Diese Schleife wird solange ausgeführt, bis die vorgesehene Berechnungszeit abgelaufen ist. Der Algorithmus kann – vorausgesetzt, das Speichern eines einzelnen Knoten und der zugehörigen Kante wird als unteilbare Sequenz erhalten – jederzeit abgebrochen werden, ohne das inkonsistente Ergebnisse entstehen.

Ist in der Iteration mindestens eine Lösung ermittelt worden, so wird die Lösung mit den geringsten Kosten gesucht und als Bahn an die unterlagerten Regelkreise weitergegeben. Bei Fehlen einer vollständigen Lösung erfolgt der Rückgriff auf eine Teillösung. Alle von der Wurzel abgehenden Kanten und Kindknoten stellen kollisionsfreie Bahnen dar, aus denen die bestmögliche gewählt wird. Besteht der

Algorithmus 8.3 Vollständiger Bahnplanungsalgorithmus.

```
1:  MotionPlanner(x_target)
2:      InitializeTree(t_0 + Δt)
3:      loop
4:          if UpdateCostEstimates(Tree, root, x_target) = success then
5:              terminate
6:          end if
7:          CollisionCheck(tree)
8:          repeat
9:              traj =ExpandTree(Tree)
10:             if traj ≠ error then
11:                 GenerateNewNodes(Tree, traj)
12:                 for all new nodes do
13:                     UpdateCostEstimates(Tree, node, x_target)
14:                 end for
15:                 Prune(Tree)
16:             end if
17:         until time is up
18:         if Tree.root.UppBound < ∞ then
19:             find path with smallest cost
20:         else if Tree.root.children ≠ ∅ then
21:             choose best temporary action
22:         else
23:             choose emergancy action
24:         end if
25:     end loop
26: end
```

Baum ausschließlich aus dem Wurzelknoten, so ist eine Ausnahmebehandlung erforderlich.

In den folgenden Abschnitten wird die konkrete Implementierung der oben hergeleiteten Algorithmen dargestellt.

Vorbereitung

Um eine kollisionsfreie Bahn planen zu können, muss die Position des Roboters bzw. die Position jedes seiner Punkte bekannt sein. Damit lässt sich der freie Konfigurationsraum $\mathcal{C}_{\text{frei}}$ genau bestimmen. In der vorliegenden Arbeit wird die Geometrie des sich in der Ebene bewegenden Roboters als kreisrund angenommen. Ist der Radius r des Roboters bekannt so kann bei Wahl des Mittelpunkts $[x, y, \phi]^{\mathrm{T}}$ als

Referenzpunkt die Konfiguration $q(x, y, \phi)$ des Roboters mittels $\mathcal{R}(q)$, d. h.

$$\mathcal{R}(x, y, \phi) = \{(x', y') | (x - x')^2 + (y - y')^2 \leq r^2\} \tag{8.8}$$

bestimmt werden. Wird der von Hindernis i eingenommene Arbeitsraum mittels $\mathcal{WO}_i$ beschrieben und ist $\mathcal{CO}_i$ dessen Abbildung in den Konfigurationsraum, so lässt sich der Teil des Konfigurationsraums, in dem Kollisionen mit Hindernis i auftreten, explizit mit

$$\mathcal{CO}_i = \{q \in \mathcal{C} | \mathcal{R}(q) \bigcap \mathcal{WO}_i \neq \emptyset\} \tag{8.9}$$

angeben. Der freie Konfigurationsraum $\mathcal{C}_{\text{frei}}$ ergibt sich damit zu

$$\mathcal{C}_{\text{frei}} = \mathcal{C} \setminus (\bigcup_i \mathcal{CO}_i) \quad . \tag{8.10}$$

Die Umgebungskarte stellt den Arbeitsraum des Roboters dar. Für die Bahnplanung ist zunächst der freie Konfigurationsraum zu ermitteln. Dieses erfolgt durch eine Aufweitung der Hindernisse in der Umgebungskarte um den Radius des Roboters. Die Abmessungen des Roboters reduzieren sich dadurch auf einen Punkt, während die Bahnplanung im aufgeweiteten Konfigurationsraum erfolgt. Die Konstruktion des Konfigurationsraums ergibt sich aus den o. g. Gleichungen: Um jeden Hindernispunkt wird ein Kreis mit dem Radius des Roboters gelegt. Die Vereinigung der durch die Kreise vereinnahmten Flächen bildet die aufgeweitete Karte. Die Implementierung in dieser Arbeit erfolgt aus Effizienzgründen, indem in einer Rasterkarte ausgehend von belegten Zellen durch Betrachtung der direkten Nachbarn iterativ der Abstand mittels *Vierer-* oder *Achternachbarschaft* bis zu einem Grenzwert hochgezählt wird. Ein Beispiel für eine aufgeweitete Karte ist in Bild 8.4 dargestellt. Die Aufweitung wird mittels dieser Methode nur für die stati-

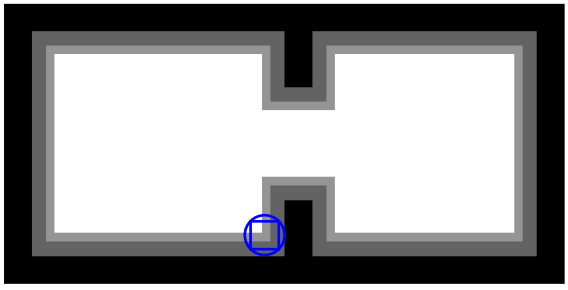
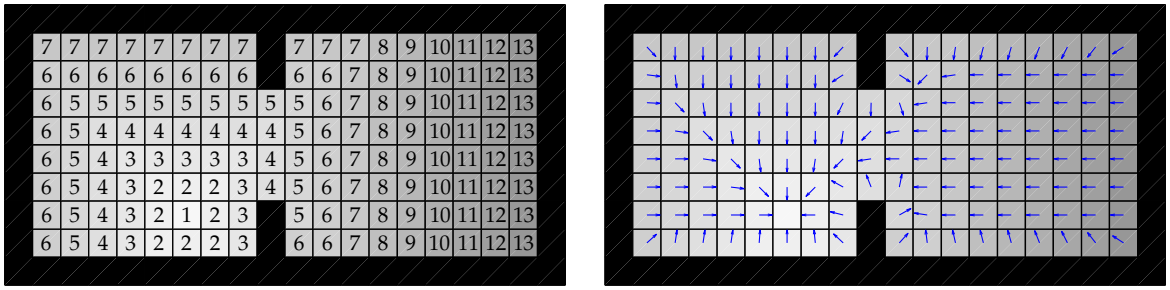


Bild 8.4: Durch das Aufweiten der Karte mit dem Radius des Roboters kann der Roboter auf einen Punkt reduziert werden. Bei rechtwinkliger Form des Roboters ergeben sich orientierungsabhängig zwei Aufweitungsradien.

sche Umgebung ermittelt. Eine Berechnung ist nur in der Phase der Initialisierung erforderlich.

Bestimmung einer Zufallskonfiguration

Neben der Aufweitung der Rasterkarte wird ebenfalls bei der Initialisierung eine Abstandsmetrik berechnet. Zwar findet die Bestimmung der Kosten gemäß Gl. (8.4) in dieser Arbeit zur Laufzeit statt. Im Folgenden wird jedoch bei der Bestimmung der Zufallskonfiguration in der Baumerweiterung von einer weiteren Metrik Gebrauch gemacht. Mittels des LEE-Algorithmus wird eine Navigationsfunktion bestimmt [Lee, 1961]. Der Algorithmus startet ausgehend vom Zielpunkt und betrach-



a) b)

Bild 8.5: Die rein zufällige Suche im Konfigurationsraum ist wenig zielgerichtet. Eine Abstandsmetrik (a) hilft solche Konfigurationen zu begünstigen, die zwischen der Roboterposition und der Zielposition liegen. Auch die Zwischenkonfigurationen können durch Gradientenbildung in Zielrichtung ausgerichtet werden (b).

tet die Nachbarn in einer *Vierer-* oder *Achternachbarschaft*. Solche Zellen die bisher unbelegt sind werden mit einem um eins inkrementierten Wert bezüglich der Ausgangszelle belegt. Diese Zelle ist dann Ausgangspunkt für die Erweiterung in der folgenden Iteration. Das Verfahren wird fortgesetzt bis alle Zellen mit einem nicht leeren Wert gefüllt sind. Die Zahlen in den Zellen repräsentieren einen Abstand zum Ziel. Bei Verfolgung des Gradienten, wird das Ziel auf dem nach dieser Metrik kürzesten Weg erreicht. In der vorliegenden Arbeit wird der auch mit Wasserfront bezeichnete Graph genutzt, um bei der Bestimmung einer zufälligen Konfiguration im Erweiterungsschritt solche Konfigurationen zu begünstigen, die zwischen Start und Ziel liegen. Außerdem kommt der Graph zum Einsatz, um durch Bestimmung des Gradienten die Orientierung bevorzugt in Richtung des Zielpunkts auszurichten. In Bild 8.5 wird die Funktionsweise skizziert. Ein Vorteil dieser Metrik gegenüber dem euklidischen Abstand ist, dass durch die Berücksichtigung der

Umgebung die Effizienz der Bahnplanung, insbesondere in mäanderförmigen Umgebungen gesteigert wird.

Ein bekanntes Problem der wahrscheinlichkeitsbasierten Bahnplanungsverfahren ist die Ausdehnung des Planungsgebiets [Choset u. a., 2005]. Die Schwierigkeiten sind dabei nicht allein durch die Größe sondern vor allem durch die Beschaffenheit bezüglich Engstellen etc. gegeben. Durch den Zufallsprozess bei der Erweiterung des Baums wird das Passieren von Engstellen erschwert. Unter dieser Einschränkung leidet die Leistungsfähigkeit des gesamten Planungsprozesses. Aus diesem Grund erfolgt eine Analyse der Umgebungskarte, um Engstellen zu erkennen und darauf aufbauend Mechanismen für deren Berücksichtigung bereitzustellen. In Engstellen ist die Einhaltung eines möglichst großen Abstands von besonderem Interesse. Ein Werkzeug um den maximalen Abstand zur Umgebung zu bestimmen, stellt das Voronoi-Diagramm dar [Aurenhammer, 1991]. Neben einem Maß für den zur Verfügung stehenden Raum liefert es zudem einen Anhaltspunkt für einen günstigen Verlauf der Bahn in Engstellen. Die Berücksichtigung im Planungsprozess erfolgt durch eine erhöhte Wahrscheinlichkeit, Zufallskonfigurationen auf den Voronoi-Linien in Engstellen zu generieren. Das Voronoi-Diagramm ist exemplarisch in Bild 8.6 dargestellt.

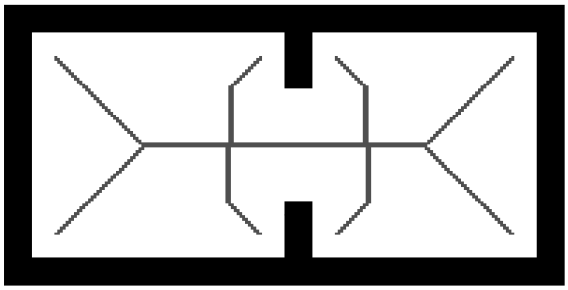


Bild 8.6: Das Voronoi-Diagramm (grau dargestellt) kennzeichnet Bereiche, die maximalen Abstand zur Umgebung aufweisen. Diese Eigenschaft kann zur Bahnplanung in Engstellen ausgenutzt werden.

Bahnplanung in hindernisfreier Umgebung

Der oben skizzierte Bahnplanungsalgorithmus setzt die Existenz eines optimalen Regelungsgesetzes für eine hindernisfreie Umgebung voraus, um mittels eines Kostenfunktionals die Güte einer Bahn beurteilen zu können. In dieser Arbeit liegt der Bahnplanung ein Unicycle-Modell zugrunde. Dieses reduziert den Roboter auf ein Einrad und bietet eine standardisierte Schnittstelle für die Steuerungs- und Regelungsalgorithmen. Bei Annahme von kinematischem Rollen des Rades unterliegt es einer nichtholonomen Zwangsbedingung, die Schlupf in Querrichtung verbietet.

Diese Zwangsbedingung wird durch

$$\dot{x}\sin\phi - \dot{y}\cos\phi = 0 \tag{8.11}$$

ausgedrückt. Bei zusätzlicher Betrachtung des dynamischen Modells wird deutlich, dass neben Gl. (8.11) die translatorische und die rotatorische Trägheit als dynamische Zwangsbedingungen zu berücksichtigen sind. Es gilt demnach, eine Trajektorie zu definieren, die bezüglich der Position und Geschwindigkeit stetig verläuft. Dieses wird durch eine geeignete Wahl von Bewegungsprimitiven erreicht. Die Repräsentation der Bahn erfolgt hier durch eine Folge von Geraden und Kurven (vergleiche Abschnitt 4.3). Die Kurven werden durch an Kreisbogensegmente angepasste Polar-Splines beschrieben. Der zeitliche Verlauf wird mittels eines Bahnparameters berücksichtigt. Durch Sicherstellung der Stetigkeit beim Übergang von Bahnsegmenten, der Differenzierbarkeit und der Stetigkeit des Krümmungsradius wird die nichtholonome Zwangsbedingung eingehalten. Eine geeignete Parametrierung stellt schließlich die Einhaltung der dynamischen Zwangsbedingungen sicher.

Das Bahnregelungssystem, das in der Lage ist, den Roboter aus einem beliebigen Zustand $\boldsymbol{x}$ in einen Zustand $\boldsymbol{x}_{\text{target}} \in \mathcal{T} = \mathcal{C} \times \mathcal{Y}$ zu überführen, wird damit durch die Repräsentation der Bahn in Verbindung mit der in Abschnitt 4.3 vorgestellten Bahnregelung gebildet. Als Optimierungskriterium dient die zurückgelegte Wegstrecke. Die Verbindung zweier Zustände wird durch eine Sequenz aus Kurve, Gerade und Kurve sichergestellt. Eine Lösung des geometrischen Problems erfolgt analytisch durch Annäherung der Kurve mittels eines Kreisbogens. Es existieren vier verschiedene Lösungen für diese Verbindung (vergl. Bild 8.7). Aus diesen wird die auf die kürzeste Wegstrecke führende Lösung gewählt.

Kollisionsprüfung

Mittels des vorgenannten Bahnplanungssystems wird die zufällig gewählte Konfiguration auf die Möglichkeit der Anbindung an den Baum geprüft. Die statische Umgebung wird durch eine Rasterkarte repräsentiert. Die Kollisionsprüfung mit dieser erfolgt mit einer an die Rasterkantenlänge angepassten Schrittweite entlang der geplanten Trajektorie. Es wird dabei jeweils geprüft, ob eine von der Bahn überstrichene Zelle belegt ist (vergleiche 6.2.2). Bei Erkennung einer Kollision wird die neue Bahn verworfen. Andernfalls ist eine bezüglich der statischen Umgebung gültige Bahn entstanden.

Für die Kollisionsprüfung mit der dynamischen Umgebung muss zusätzlich die Zeitabhängigkeit berücksichtigt werden. Die Zahl der dynamischen Objekte ist – gemessen an der Dimension der Rasterkarte – gering. Die Kollisionsprüfung mit der dynamischen Umgebung erfolgt daher individuell für jedes Objekt. Ausgewertet werden die Schnittpunkte der Bahn mit dem um den Radius des Roboters aufge-

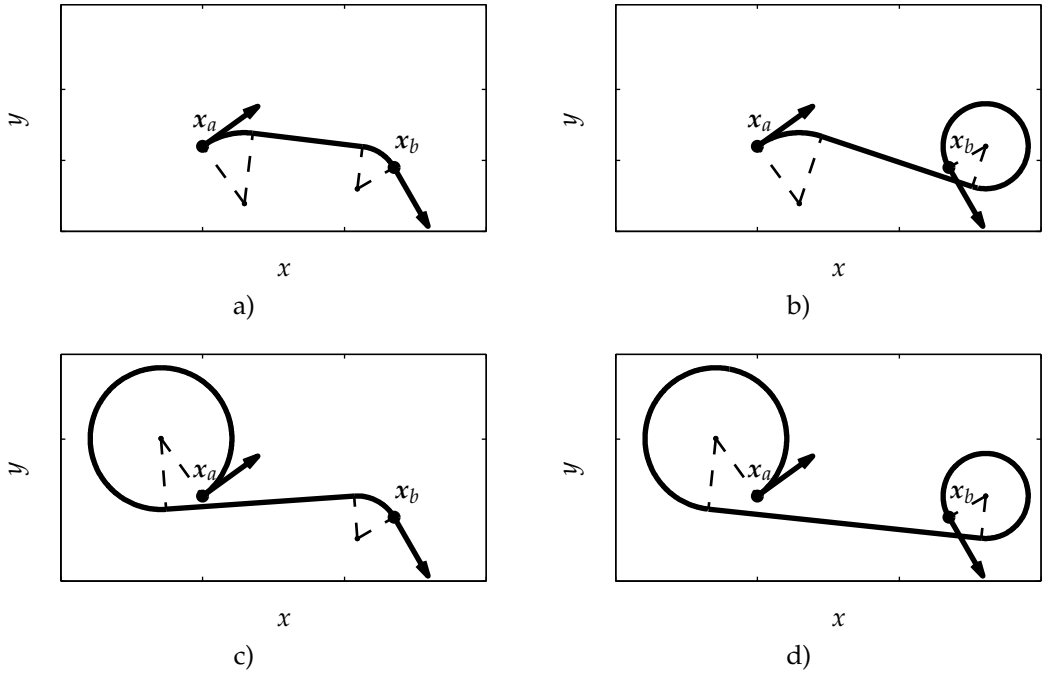


Bild 8.7: Im allgemeinen Fall gibt es zur Verbindung zweier Positionen mit zwei Kurvenelementen und einer Gerade vier verschiedene Varianten. Aus diesen wird die Variante mit der kürzesten Strecke in der Bahnplanung genutzt.

weiteten Hindernispolygon entlang der fortschreitenden Zeit mit festgelegter Zeitschrittweite. Zu diesem Zweck wird zunächst bestimmt, ob der Roboter das Bahnelement innerhalb des Vorhersagehorizonts befährt sowie ob eine Überlappung zwischen Bahnelement und dynamischem Hindernis möglich ist, um anschließend eine präzise Berechnung von Kollisionszeitpunkten zu starten.

Wird eine Kollision mit der statischen oder dynamischen Umgebung erkannt, so wird das geprüfte Bahnsegment verworfen und es folgt die Überprüfung der nächsten Baumkonfiguration.

Entkopplung von Bahnplanung und Bahnregelung

Die Entkopplung der Bahnplanung von der Bahnregelung und die damit einhergehenden unterschiedlichen Zeitskalen sind ein Kernelement des vorliegenden Ansatzes. Um die Reaktivität der Bahnplanung in dynamischer Umgebung sicherzustellen, wird ein Sicherheitsbereich um den Roboter definiert, der kollisionsfrei bleiben muss. Es handelt sich dabei um den Bereich, der während der Berechnung der Bahn für die folgende Iteration überfahren wird. Kollisionsgefahren in diesem Bereich sind durch unterlagerte Sicherungsmechanismen abzufangen. Nach

Abschluss einer Planungsiteration erfolgt die Auswahl der optimalen Bahn, die an die Bahnregelung gesendet wird. Die Bahnregelung signalisiert dem Bahnplaner jeden überfahrenen Knoten. Damit kann der Bahnplaner den Baum von bereits passierten Knoten und den davon abgehenden Pfaden bereinigen. Weiterhin kann der Bahnplaner den Sicherheitsbereich bestimmen und den Ausgangsknoten für die weitere Suche bestimmen. Die Planung für die weitere Fahrt erfolgt also stets von einem in der Zukunft liegenden Punkt auf der aktuellen Bahn. Damit ist die Stetigkeit der Trajektorie gewährleistet.

8.4 Experimentelle Erprobung der reaktiven Bahnplanung

Der reaktive Bahnplanungsalgorithmus wird auf dem in Kapitel 4 vorgestellten Versuchsträger erprobt. Für diese Zwecke ist das Zusammenwirken aller in Bild 4.4 dargestellten Module erforderlich. Bei Eingabe einer Zielkonfiguration durch den Nutzer wird zunächst die aktuelle Lage aus dem Lokalisationsmodul ermittelt. Es erfolgen dann die o. g. vorbereitenden Maßnahmen (Berechnung der Abstandsmetrik, Initialisierung des Suchbaums, Anforderung der Daten verschiedener Module, etc.) bevor die erste Schleifeniteration beginnt. Bei der Initialisierung wird im Wurzelknoten die aktuelle Lage gespeichert. In der Iterationsschleife erfolgt die Berechnung der Bahn durch Wahl einer zufälligen Konfiguration und Verbindung dieser mit dem Suchbaum.

Für die experimentelle Erprobung wird der Bahnplanungsalgorithmus mit einer Taktfrequenz von 4 Hz betrieben. Der Berechnungsweg ergibt sich dabei für eine angenommene Geschwindigkeit von $1\,\mathrm{m/s}$ zu 250 mm zuzüglich dem Bremsweg für einen eventuellen Nothalt, der als Sicherheitsbereich um den Roboter stets hindernisfrei bleiben muss. Das Objektverfolgungsmodul arbeitet mit der gleichen Taktfrequenz. Der Zeithorizont zur Berücksichtigung dynamischer Objekte beträgt 8 s. Die Länge einzelner Bahnsegmente beträgt 300 mm, diese wird durch das Einfügen von Zwischenknoten bei größeren Abständen zwischen zwei primären Knoten erreicht.

In Bild 8.8 ist der Suchbaum zunächst für einige Iterationen in hindernisfreier Umgebung jeweils nach Ablauf der Planungszeit über der Navigationsfunktion (gekennzeichnet durch den Grauwert) dargestellt. Um den Fortschritt des Roboters auf der Bahn kenntlich zu machen, ist die erste Iteration (oben links) und nachfolgend jede zehnte Iteration dargestellt. Bereits die erste Iteration liefert eine Lösung. Diese wird während des Abfahrens durch den Roboter in den folgenden Iterationen optimiert. Die „kostengünstigste Bahn“ wird jeweils in heller Farbe hervorgehoben, während die alternativen Zweige in schwarzer Farbe dargestellt sind. Farbige

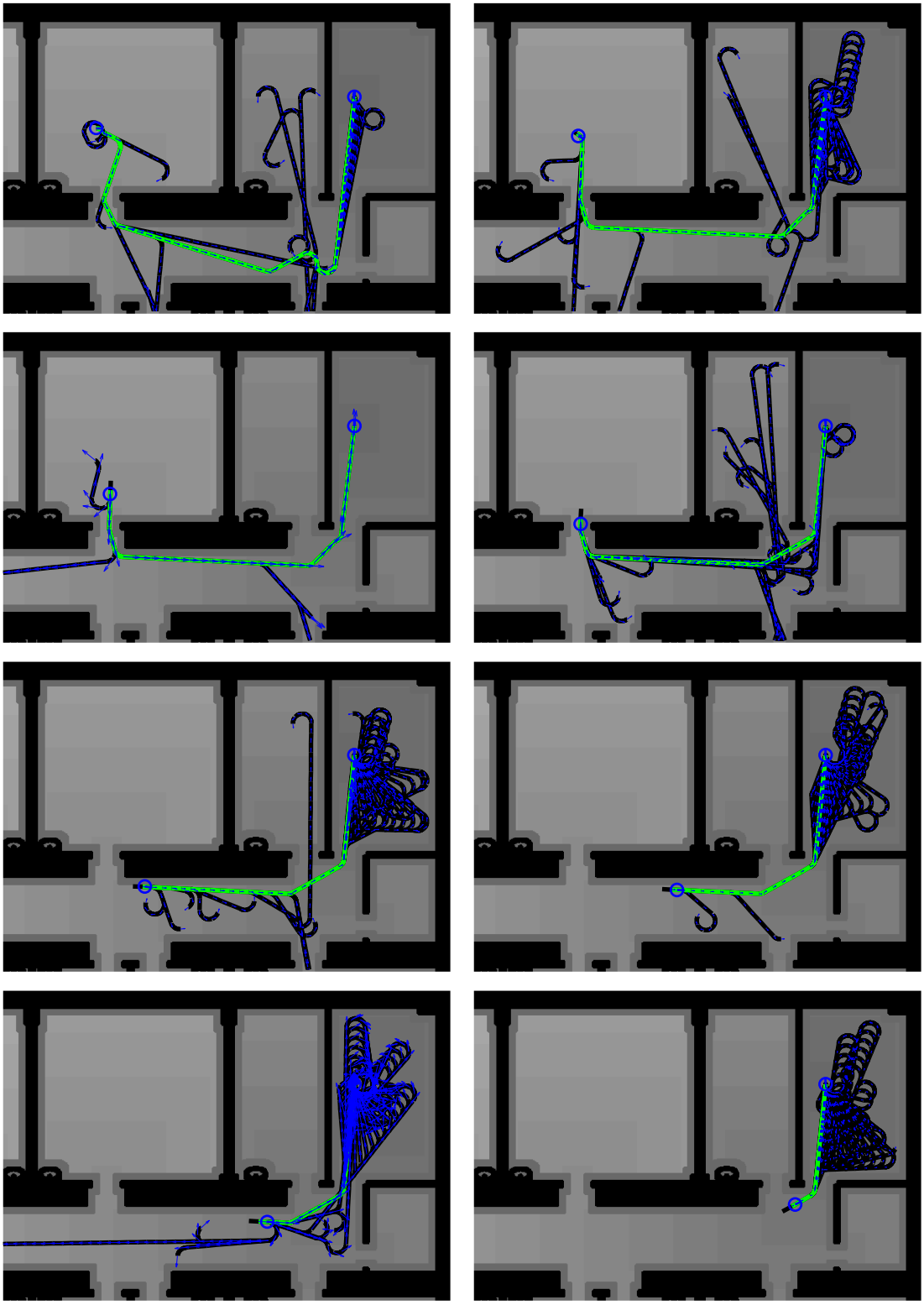


Bild 8.8: Dargestellt ist der Suchbaum im Verlauf des Abfahrens durch den Roboter, beginnend bei der ersten Iteration, gefolgt von der jeweils zehnten Iteration. Die helle Bahn kennzeichnet die „kostengünstigste“ Lösung, die farbigen Pfeile zeigen die Fahrtrichtung des Roboters an.

Pfeile deuten die Fahrtrichtung des Roboters an. Die freien Enden stellen zufällige Konfigurationen dar, um die der Suchbaum erweitert wurde. Deutlich zu erkennen ist insbesondere an Türdurchfahrten die Wirkung der sekundären Knoten: Die Anknüpfungspunkte an den Baum werden durch die kurzen Bahnsegmente vervielfältigt, so dass die Erfolgsquote bei der Baumerweiterung deutlich ansteigt und kurze Bahnen entstehen. An der Sequenz der Bilder ist zu erkennen, dass jeweils die zurückgelegten Abschnitte samt Kindern aus dem Baum gelöscht werden, so dass die Rechenkapazität für den Bereich zwischen Roboter- und Zielkonfiguration zur Verfügung steht. Naturgemäß ergibt sich bei Annäherung an das Ziel eine große Schar von Lösungen, da bei gleichbleibender Rechenkapazität der Suchbereich eingeschränkt wird.

In Bild 8.9 wird ein exemplarischer Verlauf für die Bewegung des Roboters in einer Umgebung mit dynamischen Hindernissen aufgezeigt. Das Hindernis bewegt sich diagonal über den Bildausschnitt. Der Übersichtlichkeit halber wird hier nur die als kostengünstigste Lösung gewählte Bahn eingezeichnet. Die farbigen Pfeile kennzeichnen die Fahrtrichtung. In heller Farbe dargestellt sind die von der Objektverfolgung registrierten Hindernisse. Nach den ersten drei Iterationen ist auch hier nur jede zehnte Iteration dargestellt, um den Fortschritt des Roboters auf der Bahn sichtbar zu machen. Im ersten Bild ist die ursprüngliche Bahn des Roboters dargestellt. In der Folge tritt das Objekt in den Zeithorizont für die Kollisionsprüfung ein. In der folgenden Iteration wird die Kollision des Objekts mit dem Suchbaum festgestellt. Das betroffene Element sowie alle nachfolgenden Bahnelemente werden aus dem Baum gelöscht. Ausgehend von dem letzten als kollisionsfrei eingestuften Bahnelement erfolgt nun die neue Planung innerhalb der Iteration. In der ersten Folgeiteration wird eine neue Lösung gefunden, die dann fortlaufend optimiert wird, wobei in der zweiten Folgeiteration eine geänderte Wegführung resultiert. Auch an kleinen Änderungen im Verlauf der Bildsequenz ist die fortlaufende Optimierung zu erkennen. Dem hier erkannten Hindernis wird mittels des skizzierten Bahnplanungsalgorithmus erfolgreich ausgewichen.

Zusammenfassend ist festzustellen, dass der hier entworfenen Bahnplanungsalgorithmus die Informationen aus statischer Umgebung, dynamischer Umgebung und dem Fortschritt entlang der Bahn integriert. Der Algorithmus ist in der Lage, zur Laufzeit Bahnen zu planen, bestehende Bahnen bei drohenden Kollisionen zu korrigieren und gefundene Lösungen zu optimieren. Die Planung erfolgt unter Beachtung dynamischer und nichtholonomer Zwangsbedingungen.

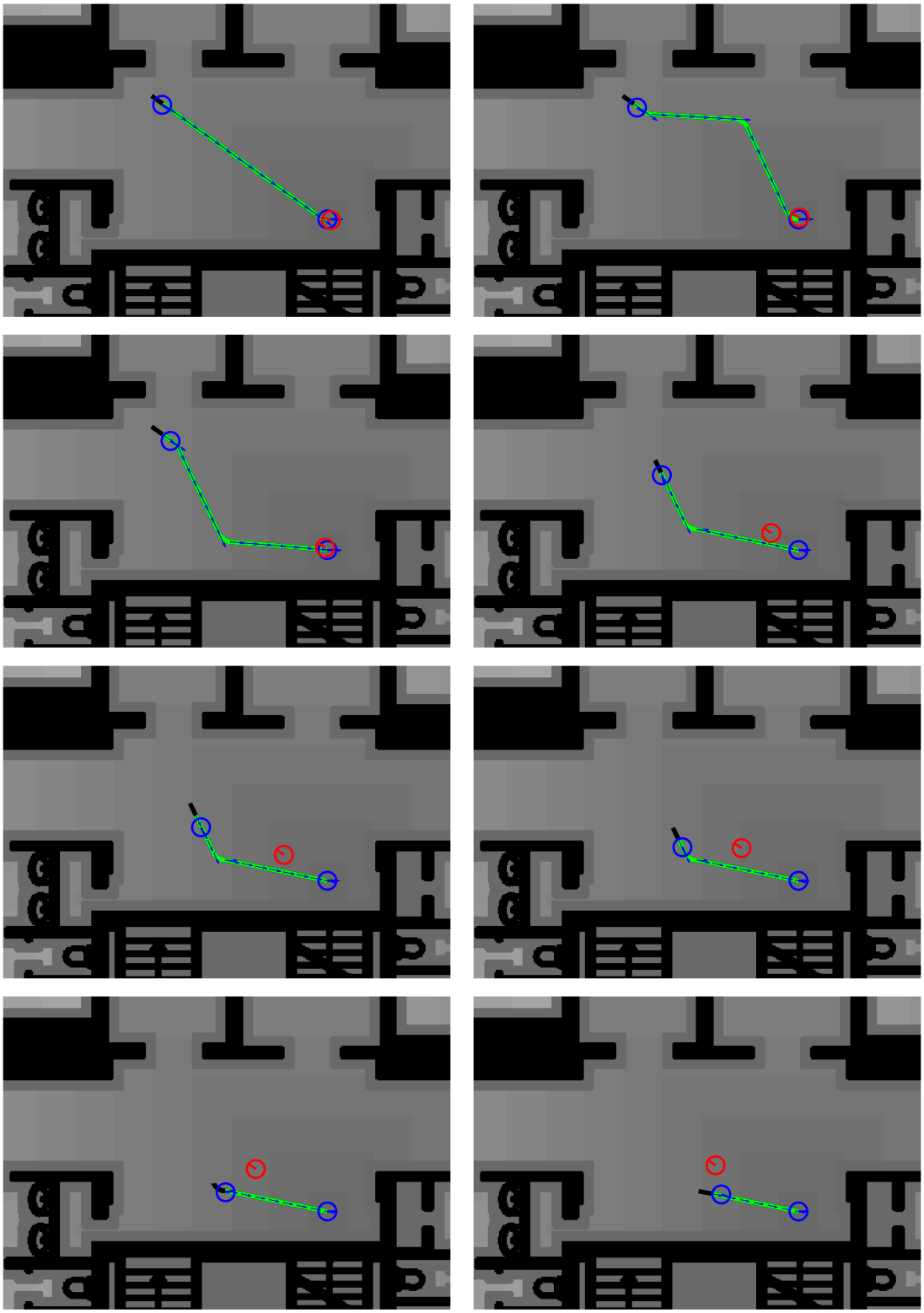


Bild 8.9: Der Roboter bewegt sich auf der dargestellten Bahn. Bei Eintritt des hell dargestellten dynamischen Hindernis in den zu überwachenden Zeithorizont erfolgt eine Neuplanung. Die Bahn wird in jeder Iteration weiter optimiert.

9 Zusammenfassung

Seit Jahren wird dem Bereich der Servicerobotik ein beispielloser wirtschaftlicher Boom prognostiziert, trotzdem ist der Verbreitungsgrad der künstlichen Diener bislang gering. Der Grund dafür ist laut Branch [2005], der einen 15 jährigen Zyklus in der Robotikeuphorie erkennt, zum einen in der Komplexität der Alltagsumgebung zu suchen sowie andererseits am Mangel einer Applikation, die den (Kosten-) Aufwand für die Installation von Servicerobotern auf breiter Front rechtfertigen würde. Es zeigt sich, dass Wissenschaft und Technik vornehmlich die Erfassung der Umgebung und die daraus abzuleitende Entscheidungsfindung bearbeiten, was Branchs These über die Komplexität stützt.

Die Herausforderung, in unstrukturierten und durch nichtdeterministische Vorgänge geprägten Umgebungen arbeiten zu können, ist groß, doch werden auf diesem Gebiet auch beachtliche Fortschritte erzielt. Nicht nur in Japan, wo mit Robotern zur Beschäftigungstherapie, Rehabilitation, Altenpflege und als Spielzeug bereits ein Markt entstanden ist und Robotik-Zeitschriften an Kiosken verkauft werden, zeigt sich das wirtschaftliche Potential dieses Bereichs. Auch der Einstieg des Massensoftwareherstellers Microsoft mit dem Microsoft Robotics Studio in den Markt der Entwicklungssoftware für Robotikanwendungen gibt einen Hinweis auf den überproportional steigenden Verbreitungsgrad.

Während also die Vorzeichen für eine größere Verbreitung der Serviceroboter und eine wachsende Entwicklergemeinde positiv stehen, gilt es Fortschritte bei der Erfassung der nichtdeterministischen Umgebung und Planung in Räumen hoher Dimension zu erzielen. In dieser Arbeit wird für die Navigationsaufgabe eines Roboters die Behandlung der inhärenten Unsicherheit innerhalb einer natürlichen Umgebung adressiert. Dabei wird mit der wahrscheinlichkeitsbasierten Zustandsschätzung insbesondere durch Anwendung der sequentiellen Monte-Carlo-Methoden [Doucet u. a., 2001] und einer stichprobenbasierten Bahnplanung [LaValle, 2006] eine methodisch durchgängige Herangehensweise an das Problem gewählt. Weiterhin wird auf die Ausnutzung struktureller Merkmale der konkreten Versuchsanordnung verzichtet, um die Übertragbarkeit der Methoden auf ähnliche Konfigurationen zu gewährleisten. Die erarbeiteten Methoden werden anhand einer konkreten Versuchsanordnung auf Praxistauglichkeit überprüft. Schließlich wird auf jegliche Vorbereitung der Umgebung verzichtet, insbesondere kommen keine künstlichen Landmarken und externen Sensorsysteme zum Einsatz. Die Navigationsaufgabe

umfasst die Bestimmung der eigenen Position innerhalb einer Umgebungskarte, die Planung des Weges zwischen der Ausgangs- und Zielposition unter Vermeidung von Kollisionen mit der statischen Umgebung, die Führung des Roboters auf dem geplanten Weg sowie die Erkennung von dynamischen Hindernissen und die Kollisionsvermeidung mit diesen.

In die Thematik wird durch die Darstellung der auf Bayes zurückgehenden bedingten Wahrscheinlichkeit, die als Grundlage für weite Teile dieser Arbeit dient, eingeleitet: Die Inversion von Ursache und Wirkung ermöglicht die Schätzung nicht messbarer Größen, indem aus einer Zustandshypothese abgeleitete erwartete Messgrößen mit der Wahrscheinlichkeit für das Eintreten dieser Hypothese gewichtet werden. Die darauf aufbauende Bayes'sche Rekursion wird in Form von Kalman-Filter, Histogrammfilter und Partikelfilter vorgestellt und kommt als elementares Werkzeug mehrfach bei der Lösung von Teilaspekten zum Einsatz.

In der Folge werden die zur kollisionsfreien Navigation zu lösenden Teilfunktionen bearbeitet. Als übergeordnete Fragestellung gilt die Form der Umgebungsrepräsentation, die für Planung und Lokalisation geeignete Eigenschaften aufweist. Die Rasterkarte zeichnet sich dabei gegenüber topologischen Karten und merkmalsbasierten Karten durch einfache Handhabbarkeit und eine effiziente Integration in die informationstechnische Verarbeitung aus. Das Problem der Kartenerstellung wird durch Einsatz der Sensorik des Roboters gelöst: Die für die Sensierung von Karteninformationen erforderliche Sensorik ist auf dem Roboter vorhanden. Durch die Exploration der Umgebung wird inkrementell eine Umgebungskarte aufgebaut. Für das Henne-Ei-Problem – die Kartierung kann nur bei Kenntnis der den Messungen zugehörigen Roboterposen erfolgen, während die Roboterposition nur anhand einer Karte ermittelt werden kann – wird ein sogenanntes SLAM[1]-Verfahren eingesetzt, dass sich durch die gleichzeitige Schätzung von Roboterlage und Umgebungskarte auszeichnet. Als Basis für die dargestellten experimentellen Arbeiten zur kollisionsfreien Navigation wird mittels eines Versuchsträgers die Einsatzumgebung kartiert.

Die Lokalisation erfolgt innerhalb der geschätzten Rasterkarte durch Einsatz der Monte-Carlo-Lokalisation. Diese ist eine Implementierung des Partikelfilters: Eine große Zahl von über die Rasterkarte verteilten Zustandshypothesen wird anhand der Konsistenz zwischen der virtuellen Messung innerhalb der Rasterkarte und der Sensormessung gewichtet. Solche Hypothesen mit hohem Gewicht kennzeichnen dann einen Ort mit erhöhter Aufenthaltswahrscheinlichkeit und werden vervielfältigt. Auf diese Weise konzentrieren sich die Hypothesen am wahrscheinlichen Aufenthaltsort und durch Bildung des Erwartungswerts wird auf die Roboter-

[1]Simultaneous Localization and Mapping

position geschlossen. Durch die große Zahl von Hypothesen und die damit einhergehende Anzahl an abzuleitenden hypothetischen Messgrößen ist das genannte Verfahren sehr rechenintensiv. Dieses führt auf eine für gewöhnlich langsame Taktung im Bereich unter 5 Hz für die Zustandsschätzung, die für den Einsatz in schnellen Regelungen unzureichend ist. Durch eine signifikante Steigerung der Effizienz bei der Generierung virtueller Messgrößen wird in dieser Arbeit die Effizienz dieses Verfahrens im Einsatz mit Rasterkarten gesteigert. Zudem wird hier durch die Entwicklung eines Mehrratenansatzes und die Nebenläufigkeit von Prädiktion und Aktualisierung im Bayes'schen Rekursionsprozess die Monte-Carlo-Lokalisation für den direkten Einsatz in schnellen Regelungen qualifiziert.

Der Wahrnehmung von dynamischen Hindernissen wird durch einen Objektverfolgungsalgorithmus Rechnung getragen, der aus den Rohdaten von scannenden Laserentfernungsmesssystemen Merkmale von bewegten Objekten extrahiert und diese mittels der Bestimmung der gemeinsamen Zuordnungswahrscheinlichkeit den durch Partikel repräsentierten Objekten zuordnet. Durch den Einsatz von Filtern für die Objektverfolgung können dabei kurzzeitige Verdeckungen und zufällige Fehlmessungen toleriert werden.

Die Integration der vorherigen Bestandteile erfolgt schließlich in der im Rahmen dieser Arbeit durchgeführten Entwicklung eines reaktiven Bahnplanungsalgorithmus, der unter Berücksichtigung dynamischer und nichtholonomer Zwangsbedingungen stichprobenbasiert den Arbeitsraum exploriert und eine Bahn, frei von Kollisionen mit statischen und dynamischen Hindernissen, erzeugt. Die Berücksichtigung der genannten Zwangsbedingungen bei der geforderten Reaktivität gelingt durch die Diskretisierung des kontinuierlichen Raums zulässiger Stellfolgen auf Bewegungsprimitive. In einem iterativen Algorithmus wird dabei ein Suchbaum aufgebaut, der durch die stichprobenbasierte Generierung von Konfigurationen erweitert wird. Die Effizienz des Verfahrens wird durch neu entwickelte Heuristiken bei der Erweiterung des Suchbaums sichergestellt: Während der Explorationsphase werden die zu erweiternden Knoten des Baums so gewählt, dass der explorierte Raum schnell erweitert wird. Ist eine Lösung gefunden so wird im Folgenden eine Optimierungsheuristik angewendet, die für die Reduktion der Kosten auf dem Weg zwischen Ausgangs- und Ziellage sorgt.

In der vorliegenden Arbeit wird das Navigationsproblem für einen mobilen Roboter innerhalb dynamischer Umgebung mittels methodisch durchgängiger Verfahren erfolgreich gelöst. Die offene Struktur der dargestellten Komponenten bietet eine Grundlage für die Erweiterung des Systems zur weiteren Leistungssteigerung, beispielsweise durch die Integration weiterer Sensorinformationen. Dargestellt ist die Erarbeitung der unterschiedlichen Teilaspekte der Navigationsaufgabe, deren Funktion durch das Zusammenwirken verschiedener Komponenten am Versuchsträger nachgewiesen wird.

Literatur

[Abdellatif u. a. 2005] Abdellatif, H. ; Blume, H. ; Heimann, B. ; Hornung, O.: Model-Based Control of Robotic Systems. In: *Proc. of the 2nd Int. Conference on Machine Intelligence, ACIDCA-ICMI'2005*, 2005

[Abdellatif u. a. 2002] Abdellatif, H. ; Weidemann, D. ; Michaelsen, A. ; Grotjahn, M. ; Heimann, B.: Control of the Omnidirectional Mobile Robot MARGe. In: *Proc. of the 14th CISM-IFToMM Symp. on the Theory and Practice of Robots and Manipulators (RoManSy)*, 2002, S. 101–108

[Abdellatif u. a. 2003] Abdellatif, Houssem ; Michaelsen, Arne ; Heimann, Bodo: Position Estimation and Control of an 8-DOF Omnidirectional Mobile Robot. In: *Proc. of Raad'03, 12th Internatinal Workshop on Robotics in Alpe-Adria-Danube Region*, 2003

[Athans und Falb 1966] Athans, M. ; Falb, P. L.: *Optimal Control*. McGraw-Hill Book Company, 1966

[Aurenhammer 1991] Aurenhammer, F.: Voronoi diagrams - a survey of a fundamental geometric data structure. In: *ACM Computing Surveys* 23 (1991), Nr. 3, S. 345–405

[Bar-Shalom und Fortmann 1988] Bar-Shalom, Yaakov ; Fortmann, Thomas E. ; Ames, William F. (Hrsg.): *Mathematics in Science and Engineering*. Bd. 179: *Tracking and Data Association*. Orlando : Academic Press, Inc., 1988. – ISBN 0-12-079760-7

[Barraquand u. a. 1992] Barraquand, J. ; Langlois, B. ; Latombe, J.-C.: Numerical potential field techniques for robot path planning. In: *IEEE Transactions on Systems, Man and Cybernetics* 22 (1992), Nr. 2, S. 224–241

[Barraquand und Latombe 1990] Barraquand, J. ; Latombe, J.: A Monte Carlo Algorithm for Path Planning with Many Degrees of Freedom. In: *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, 1990

[Bayes 1763] Bayes, Thomas: An Essay towards solving a Problem in the Doctrine of Chances. In: *Philosophical Transactions of the Royal Society of London* 53 (1763), S. 370–418

[Bennewitz u. a. 2005] Bennewitz, M. ; Burgard, W. ; Cielniak, G. ; Thrun, S.: Learning Motion Patterns of People for Compliant Motion. In: *The International Journal of Robotics Research* 24 (2005), January, Nr. 1, S. 31–48

[Besl und McKay 1992] Besl, Paul J. ; McKay, Neil D.: A Method for Registration of 3-D Shapes. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 14 (1992), Nr. 2, S. 239–256. – ISSN 0162-8828

[Betke und Gurvits 1997] Betke, Margrit ; Gurvits, Leonid: Mobile Robot Localization Using Landmarks. In: *IEEE Transactions on Robotics and Automation* 13 (1997), April, Nr. 2, S. 251

[Blume u. a. 2006] Blume, H. ; Abelbeck, F. ; Heimann, B.: A Rule Based Data Fusion Scheme for Wheeled Mobile Robot Localization. In: *Preprints of the 4th IFAC-Symposium on Mechatronic Systems, Mechatronics 2006*, 2006, S. 867–872

[Blume und Gutzeit 2004] Blume, H. ; Gutzeit, F.: Serviceroboter ermittelt Reibwerte von Gummi. In: *Technologie-Informationen niedersächsischer Hochschulen* 2004 (2004), Nr. 3, S. 6

[Blume und Heimann 2006] Blume, H. ; Heimann, B.: Simulating a Laser Range Scanner under Application Conditions. In: *Preprints of the 8th Int. IFAC Symposium on Robot Control, SYROCO*, 2006

[Blume und Heimann 2007] Blume, H. ; Heimann, B.: A geometric approach to consideration of control variable limits for wheeled mobile robots with kinematic constraints. In: *Proc. of the XII Int. Symposium on Dynamic Problems of Mechanics, XII Diname*, 2007

[Blume u. a. 2002] Blume, Holger ; Heimann, Bodo ; Lindner, Markus: Design of an Outdoor Mobile Platform for Friction Measurement on Street Surfaces. In: *Proc. of the Int. Colloquium on Autonomous and Mobile Systems*, 2002, S. 65–68

[BMBF 2005] Bundesministerium für Bildung und Forschung: *Bekanntmachung von Förderrichtlinien zum Forschungsprogramm Ïeitinnovation Servicerobotik¨*. Internet. Januar 2005. – www.bmbf.de/foerderungen/3369.php

[Borenstein und Feng 1994] Borenstein, J. ; Feng, L.: UMBmark A Method for Measuring, Comparing, and Correcting Dead-reckoning Errors in Mobile Robots / University of Michigan. Technical Report, December 1994 (UM-MEAM-94-22). – Forschungsbericht. citeseer.ist.psu.edu/borenstein95umbmark.html

[Branch 2005] Branch, Allan: Alle 15 Jahre wieder. In: *Technology Review* 3 (2005), S. 30. – www.heise.de/tr/Alle-15-Jahre-wieder–/artikel/56554/

[Bresenham 1965] Bresenham, Jack: Algorithm for computer control of a digital plotter. In: *IBM systems journal* 4 (1965), Nr. 1, S. 25–30

[Bryson und Ho 1975] Bryson, A. E. ; Ho, Y.-C.: *Applied Optimal Control*. New York : Hemisphere Publishing Corp., 1975

[Castellanos u. a. 1999] Castellanos, J. ; Montiel, J. ; Neira, J. ; Tardos, J.: The SPmap: A probabilistic framework for simultaneous localization and map building. In: *IEEE Transactions on Robotics and Automation* 15 (1999), Nr. 5, S. 948–953. – URL `citeseer.ist.psu.edu/castellanos99spmap.html`

[Castellanos und Tardós 1999] Castellanos, José A. ; Tardós, Juan D.: *Mobile Robot Localization and Map Building*. Kluwer Academic Publishers, 1999

[Choset u. a. 2005] Choset, Howie ; Lynch, Kevin M. ; Hutchinson, Seth ; Kantor, George ; Burgard, Wolfram ; Kavraki, Lydia E. ; Thrun, Sebastian: *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005

[Cohen und Koss 1992] Cohen, C. ; Koss, F.: A comprehensive study of three-object triangulation. In: *SPIE Mobile Robots VII*, 1992, S. 95–106. – citeseer.ist.psu.edu/cohen92comprehensive.html

[Cox 1993] Cox, Ingemar J.: A review of statistical data association techniques for motion correspondence. In: *International Journal of Computer Vision* 10 (1993), Nr. 1, S. 53–66

[Cui u. a. 2005] Cui, J. ; Zha, H. ; Zhao, H. ; Shibasaki, R.: Tracking multiple people using laser and vision. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005

[DIN 66253-2 1998] : *PEARL 90*. April 1998

[Dissanayake u. a. 2001] Dissanayake, Gamini ; Newman, Paul ; Clark, Steven ; Durrant-Whyte, Hugh F. ; Csorba, M.: A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. In: *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION* 17 (2001), June, Nr. 3, S. 229–241

[Donald u. a. 1993] Donald, Bruce R. ; Xavier, Patrick G. ; Canny, John F. ; Reif, John H.: Kinodynamic Motion Planning. In: *Journal of the ACM* 40 (1993), Nr. 5, S. 1048–1066. – URL `citeseer.ist.psu.edu/donald93kinodynamic.html`

[Doucet u. a. 2000] Doucet, A. ; Freitas, J.F.G. de ; Murphy, K. ; Russel, S.: Rao-Blackwellized particle Filtering for dynamic bayesian networks. In: *Proceedings of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2000, S. 176–183

[Doucet u. a. 2001] Doucet, Arnaud ; Freitas, Nando de ; Gordon, Neil ; Jordan, M. (Hrsg.) ; Lauritzen, S.L. (Hrsg.) ; J.F.Lawless (Hrsg.) ; Nair, V. (Hrsg.): *Sequential Monte Carlo Methods in Practice*. New York : Springer, 2001 (Statistics for Enngineering and Information Science). – ISBN 0-387-95146-6

[Eliazar und Parr 2003] Eliazar, A. ; Parr, R.: DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks. In: *Proceedings of the Int. Conf. on Artificial Intelligence*, 2003, S. 1135–1142

[Fod u. a. 2002] Fod, A. ; Howard, A. ; Mataric, M.J.: Laser-Based People Tracking. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* IEEE (Veranst.), 2002

[Fox u. a. 1999] Fox, Dieter ; Burgard, Wolfram ; Thrun, Sebastian: Markov Localization for Mobile Robots in Dynamic Environments. In: *Journal of Artificial Intelligence Research* 11 (1999), November, S. 391–427. – URL `http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume11/fox99a-html/jair-localize.html`

[Fox u. a. 2001] Fox, Dieter ; Thrun, Sebastian ; Burgard, Wolfram ; Dellaert, Frank: Particle Filters for Mobile Robot Localization. In: Doucet, Arnaud (Hrsg.) ; Freitas, Nando de (Hrsg.) ; Gordon, Neil (Hrsg.): *Sequential Monte Carlo Methods in Practice*. New York : Springer, 2001. – URL `citeseer.ist.psu.edu/article/fox01particle.html`

[Gerth 1983] Gerth, W.: PEARL-System für Einplatinenrechner. In: *Regelungstechnische Praxis* 25 (1983), S. 490

[Grinstead und Snell 2003] Grinstead, Charles M. ; Snell, J. L.: *Introduction to Probability*. American Mathematical Society, 2003. – URL `http://www.dartmouth.edu/~chance/teaching_aids/books_articles/probability_book/book.html`

[Gutmann und Konolige 1999] Gutmann, J.-S. ; Konolige, K.: Incremental mapping of large cyclic environments. In: *Proceedings of the IEEE Int. Symposium on Computational Intelligence in Robotics an Automation*, 1999

[Hähnel u. a. 2003] Hähnel, D. ; Burgard, W. ; Fox, D. ; Thrun, S.: An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser measurements. In: *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2003, S. 206–211

[Hanebeck u. a. 1999] Hanebeck, Uwe D. ; Saldic, Nihad ; Schmidt, Günther: A Modular Wheel System for Mobile Robot Applications. In: *Proceedings of the 1999 IEEERSJ International Conference on Intelligent Robots and Systems*, 1999

[Heimann u. a. 2007] Heimann, B. ; Gerth, W. ; Popp, Karl: *Mechatronik: Kompenenten, Methoden, Beispiele*. Muenchen, Wien : Carl Hanser Verlag, 2007

[Horn 1997] Horn, Joachim P.: *Bahnführung eines mobilen Roboters mittels absoluter Lagebestimmung durch Fusion von Entfernungsbild- und Koppelnavigationsdaten*. VDI-Verlag, Düsseldorf, Lehrstuhl für Steuerungs- und Regelungstechnik, Technische Universität München, Dissertation, 1997

[Hornung 2007] Hornung, Oliver: *Bildbasierte Bahnplanung und Regelung von Robotern für Handhabungsaufgaben*. Erlangen, Leibniz Universität Hannover, Dissertation, 2007. – VDI Fortschritt-Berichte, Band Nr.: 1116

[Hsu u. a. 1998] Hsu, D. ; Kavraki, L. ; Latombe, J. ; Motwani, R. ; Sorkin, S.: On finding narrow passages with probabilistic roadmap planners. In: *In Proceedings of the 1998 International Workshop on Algorithmic Foundations of Robotics (WAFR)*, 1998. – citeseer.ist.psu.edu/hsu98finding.html

[Hsu u. a. 1999] Hsu, D. ; Latombe, J.-C. ; Motwani, R.: Path Planning in Expansive Configuration Spaces. In: *International Journal Computational Geometry & Applications* 4 (1999), S. 495–512

[Hsu u. a. 2000] Hsu, David ; Kindel, Robert ; Latombe, Jean-Claude ; Rock, Stephen: Randomized Kinodynamic Motion Planning with Moving Obstacles. In: *Proceedings Workshop on the Algorithmic Foundations of Robotics*, 2000

[Hwang und Ahuja 1992] Hwang, Y. K. ; Ahuja, N.: A potential field approach to path planning. In: *IEEE Transactions on Robotics and Automation* 8 (1992), Nr. 1, S. 23–32

[Kalman 1960] Kalman, R. E.: A New Approach to Linear Filtering and Prediction Problems. In: *Trans. ASME, Series D, Journal of Basic Engineering* 82 (1960), S. 35–45

[Kavraki u. a. 1996a] Kavraki, L. ; Kolountzakis, M. ; Latombe, J.: Analysis of probabilistic roadmaps for path planning. In: *In Proceedings of the IEEE International Conference on Roboticsc and Automation* Bd. 4, 1996, S. 3020–3025. – citeseer.ist.psu.edu/article/kavraki98analysis.html

[Kavraki u. a. 1998] Kavraki, Lydia E. ; Latombe, Jean-Claude ; Raghavan, Prabhakar ; Motwani, Rajeev: Randomized Query Processing in Robot Path Planning. In: *Journal of Computer and System Science* 57 (1998), Nr. 1, S. 50–60

[Kavraki u. a. 1996b] Kavraki, Lydia E. ; Svestka, Petr ; Latombe, Jean-Claude ; Overmars, Mark H.: Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. In: *IEEE Transactions on Robotics and Automation* 12 (1996), Nr. 4, S. 566–580

[Khatib 1985] Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. In: *IEEE International Conference on Robotics and Automation. Proceedings* Bd. 2, 1985, S. 500–505

[Kleinhagenbrock u. a. 2002] Kleinhagenbrock, M. ; Lang, S. ; Fritsch, J. ; Lömker, F. ; Fink, G. ; Sagerer, G.: Person tracking with a mobile robot based on multi-modal anchoring. In: *IEEE International Workshop on Robot and Human Interactive Communication (ROMAN)*, 2002

[Kluge u. a. 2001] Kluge, Boris ; Köhler, Christian ; Prassler, Erwin: Fast and Robust Tracking of Multiple Moving Objects with a Laser Range Finder. In: *Proc. of International Conference on Robotics and Automation, Soul, Mai 2001* IEEE (Veranst.), 2001

[Kuffner und LaValle 2000] Kuffner, James J. ; LaValle, Steven M.: RRT-Connect: An Efficient Approach to Single-Query Path Planning. In: *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, 2000

[Latombe 1999] Latombe, J.: Motion Planning: A Journey of Robots, Molecules, Digital Actors, and Other Artifacts. In: *International Journal of Robotics Research* 18 (1999), Nr. 11, S. 1119–1128

[Latombe 1991] Latombe, Jean-Claude: *Robot Motion Planning*. Boston : Kluwer Academic Publishers, 1991

[LaValle 1998] LaValle, S. M.: Rapidly-Exploring Random Trees: A New Tool for Path Planning / Computer Science Dept., Iowa State University. October 1998 (98-11). – Forschungsbericht. citeseer.ist.psu.edu/311812.html

[LaValle und Kuffner 1999] LaValle, S. M. ; Kuffner, J. J.: Randomized Kinodynamic Planning. In: *Proceedings IEEE International Conference on Robotics and Automation*, 1999, S. 473–479

[LaValle 2006] LaValle, Steven M.: *Planning Algorithms*. Cambridge University Press, 2006

[Lee 1961] Lee, C. Y.: An Algorithm for Path Connections and Its Applications. In: *IRE Transactions on Electronic Computers* EC-10 (1961), S. 364–365

[Leica 2006] Leica Geosystems AG (Veranst.): *Leica Laser Tracker*. 1. 2006. – URL `http://www.leica-geosystems.com/metrology/de/products/laser_tracker/lgs_35317.htm`

[Leonard und Feder 1999] Leonard, J. ; Feder, H.: A computationally efficient method for large-scale concurrent mapping and localization. In: J. Hollerbach, D. K. (Hrsg.): *International Symposium on Robotics Research*. London : Springer-Verlag, 1999. – URL `citeseer.ist.psu.edu/leonard00computationally.html`

[Leonard und Feder 2000] Leonard, J.J. ; Feder, H.J.S.: A computationally efficient method for large-scale concurrent mapping and localization. In: *Proceedings of the Conf. on Neural Innformation Processing Systems*, 2000, S. 169–179

[Lindström und Eklundh 2001] Lindström, M. ; Eklundh, J.-O.: Detecting and Tracking Moving Objects from a Mobile Platform using a Laser Range Scanner. In: *Proc. IEEE International Conference on Intelligent Robots and Systems IROS'2001* Bd. 3, October 2001, S. 1364–1369

[Lozano-Perez 1979] Lozano-Perez, T.: An Algorithm for Planning Collision-free Paths Among Polyhedral Obstacles. In: *Communications of the ACM* 22 (1979), Nr. 10, S. 560–570

[Lozano-Perez 1981] Lozano-Perez, T.: Automatic Planning of Manipulator Transfer Movements. In: *IEEE Transactions on Systems, Man and Cybernetics* SMC-11 (1981), Nr. 10, S. 681–698

[Luca u. a. 2000] Luca, Alessandro D. ; Oriolo, Giuseppe ; Vendittelli, Marilena: Stabilization of the unicycle via dynamic feedback linearization. In: *Proceedings of the 6th IFAC Symp. on Robot Control*, 2000, S. 397–402

[Mantegazza u. a. 2000] Mantegazza, P. ; Bianchi, E. ; Dozio, L. ; Papacharalambous, S. ; Hughes, S. ; Beal, D.: RTAI: Real-Time Application Interface. In: *Linux Journal* (2000). – URL `www.linuxjournal.com/article/3838`

[McGillem und Rappaport 1989] McGillem, Clare D. ; Rappaport, Theodore S.: A Beacon navigation method for autonomous vehicles. In: *IEEE Transactions on Vehicular Technology* 38 (1989), August, Nr. 3, S. 132–139

[Montemerlo u. a. 2002a] Montemerlo, M. ; Thrun, S. ; Koller, D. ; Wegbreit, B.: FastSLAM: A factored solution to simultaneous localization and mapping. In: *Proceedings of the National Conf. on Artificial Intelligence*, 2002, S. 593–598

[Montemerlo u. a. 2003] Montemerlo, M. ; Thrun, S. ; Koller, D. ; Wegbreit, B.: FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In: *Proceedings of the Int. Conf. on Artificial Intelligence*, 2003, S. 1151–1156

[Montemerlo u. a. 2002b] Montemerlo, M. ; Whittaker, W. ; Thrun, S.: Conditional Particle Filters for Simultaneous Mobile Robot Localization and People-Tracking. In: *IEEE International Conference on Robotics and Automation (ICRA)* ICRA (Veranst.), 2002

[Moravec und Elfes 1985] Moravec, Hans ; Elfes, A. E.: High Resolution Maps from Wide Angle Sonar. In: *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, March 1985, S. 116–121

[Murphy 1999] Murphy, Kevin P.: Bayesian map learning in dynamic environments. In: Solla, S. A. (Hrsg.) ; Leen, T. K. (Hrsg.) ; Müller, K.-R. (Hrsg.): *Advances in Neural Information Processing Systems 12*, 1999

[Overmars 1992] Overmars, M.: A Random Approach to Motion Planning / Department of Computer Science, Utrecht University. 1992. – Forschungsbericht. Technical Report RUU-CS-92-32

[Overmars und Svestka 1996] Overmars, M. H. ; Svestka, P.: A Paradigm for Probabilistic Path Planning / Department of Computer Science, Utrecht University. 1996. – Forschungsbericht. www.cs.uu.nl/research/techreps/repo/CS-1995/1995-22.pdf

[Overmeyer u. a. 2007] Overmeyer, Ludger ; Baum, Mathias ; Niemann, Björn ; Abelbeck, Frank ; Fricke, Dirk-H.: Qualification Tests of HF RFID Foil Transponders for a Vehicle Guidance System. In: *Proceedings of the 2007 IEEE Intelligent Transportation Systems Conference*. Seattle, WA, USA : IEEE, September 2007, S. 950–955

[Overmeyer u. a. 2006] Overmeyer, Ludger ; Niemann, Björn ; Baum, Mathias: Aufbau von Fahrerlosen Transportsystemen (FTS) durch eine dezentrale Datenstruktur. In: *Logistics Journal* e-Journal (2006). – URL `http://www.elogistics-journal.de/archiv/2006/oktober/618`

[Preparata und Shamos 1985] Preparata, Franco P. ; Shamos, Michael I.: *Computational geometry: An introduction*. Springer-Verlag New York, Inc., 1985

[RTAI 2008] RTAI: *RTAI - the RealTime Application Interface for Linux from DIAPM*. 2008. – URL `www.rtai.org`

[RTOS-UH 2008] RTOS-UH: *RTOS-UH: Realtime Operating System - University of Hannover*. 2008. – URL `www.irt.uni-hannover.de/rtos/`

[Scheutz u. a. 2004] Scheutz, M. ; McRaven, J. ; Cserey, G.: Fast, reliable, adaptive, bimodal people tracking for indoor environments. In: *IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, 2004

[Schulz u. a. 2003] Schulz, D. ; Burgard, W. ; Fox, D. ; Cremers, A.: People Tracking with Mobile Robots Using Sample-based Joint Probabilistic Data Association Filters. In: *IJRR* 22 (2003), Nr. 2

[Stachniss 2006] Stachniss, Cyrill: *Exploration and Mapping with Mobile Robots*, Albert-Ludwigs-Universität Freiburg im Breisgau, Dissertation, April 2006

[Thrun 2001] Thrun, S.: An online mapping algorithm for teams of mobile robots. In: *Int. Journal of Robotics Research* 20 (2001), Nr. 5, S. 335–363

[Thrun u. a. 2005] Thrun, S. ; Burgard, W. ; Fox, D.: *Probabilistic Robotics*. MIT Press, 2005

[Topp und Christensen 2005] Topp, E. ; Christensen, H.: Tracking for following and passing persons. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005

[Welch und Bishop 2001] Welch, Greg ; Bishop, Gary: *An Introduction to the Kalman Filter*. Scriptum. 2001. – URL `http://www.cs.unc.edu/~welch/kalman/`

[Werum und Windauer 1978] Werum, W. ; Windauer, H.: *Introduction to PEARL, Process and Experiment, Automation Realtime Language*. Vieweg-Verlag, 1978

[Wolter u. a. 2003] Wolter, B. ; Albert, A. ; Gerth, W.: Distinctness of Reaction - Ein Messverfahren zur Beurteilung von Echtzeitsystemen. In: *at - automatisierungstechnik* 51 (2003), Nr. 10, S. 396–403 (Teil1), 445–452 (Teil2)

[Wulf u. a. 2003] Wulf, Oliver ; Kiszka, Jan ; Wagner, Bernardo: A Compact Software Framework for Distributed Real-Time Computing. In: *Proc. of the 5th Real-Time Linux Workshop*, 2003

Lebenslauf

Persönliches

Name:	Holger Blume
Geburtsdatum:	23. November 1975
Geburtsort:	Hildesheim
Familienstand:	verheiratet mit Tamara Blume, geb. Preußner Tochter Matilda Blume, geboren am 26. Juli 2005 Sohn Hugo Blume, geboren am 18. Juni 2007

Schulische Laufbahn

Aug. 1982 – Juli 1986	Grundschule Uslar
Aug. 1986 – Juli 1992	Heinrich-Roth-Gesamtschule, Bodenfelde
Aug. 1992 – Mai 1995	Gymnasium Uslar, Abschluss: Abitur

Studium

Okt. 1995 – Mai 2001	Studium des Maschinenbaus an der Universität Hannover, Vordiplom Sept. 1997, Abschluss: Diplom-Ingenieur
Mai 1998 – Okt. 1998	Praktikum ContiTech Continental General Tire, Inc., Farmington Hills Michigan USA
April 1999 – Feb. 2001	Wissenschaftliche Hilfskraft am Institut für Mechanik und am Institut für Mess- und Regelungstechnik der Universität Hannover

Berufstätigkeit

Juni 2001 – Jan. 2008	Wissenschaftlicher Mitarbeiter am Institut für Mechanik, später Institut für Robotik, der Leibniz Universität Hannover, Mitarbeiter im Mechatronik-Zentrum Hannover
Juni 2003 – Jan. 2008	Wahrnehmung der Funktion des Oberingenieurs für das Lehrgebiet C (Mechatronik), später Institut für Robotik