ALBERT-LUDWIGS-UNIVERSITÄT
FREIBURG
INSTITUT FÜR INFORMATIK

Lehrstuhl für Autonome Intelligente Systeme
Prof. Dr. Wolfram Burgard

Robustes Lernen von Umgebungskarten durch Integration verschiedener Repräsentationen

Diplomarbeit

Kai M. Wurm

Erstgutachter:	Prof. Dr. Wolfram Burgard
Zweitgutachter:	Prof. Dr. Bernhard Nebel
Betreuer:	Dr. Cyrill Stachniss
Abgabedatum:	19. Juni 2007

Kurzfassung

Die Erstellung von Umgebungskarten aus Roboterdaten wird entscheidend von der gewählten Umweltrepräsentation beeinflusst. Häufig werden hierzu eine Menge von Landmarken oder eine Rasterkarte der Umgebung gewählt. Beide Formen der Modellierung bieten abhängig von der Beschaffenheit der Umwelt verschiedene Vor- und Nachteile. In dieser Arbeit wird ein Ansatz vorstellt, der eine Kombination aus Landmarken und Rasterkarten verwendet. So können während der Kartierung die Roboterpositionen und Umgebungskarten anhand der jeweils am besten geeigneten Modellierung korrigiert werden. Die Auswahl zwischen beiden Modellen wird anhand der aktuellen Sensordaten getroffen. Wir nutzen ein Lernverfahren, um diese Auswahl anhand von Beispielszenarien zu erlernen. Experimente in Simulation und mit echten Roboterdaten zeigen, dass der kombinierte Ansatz gegenüber den Einzelverfahren zu einer robusteren Kartierung führt.

Abstract

The designer of a mapping system for mobile robots has to choose how to represent the environment. Popular models are feature maps and grid maps. Depending on the structure of the environment, each representation has certain advantages. In this thesis, we present an approach that maintains feature maps as well as grid maps of the environment. This allows a robot to update its pose and map estimate based on the representation that models the surrounding of the robot in the best way. The model selection procedure is obtained by reinforcement learning and takes a decision based on the current observation. This allows a robot to learn accurate maps in a more robust way than approaches using pure feature maps or grid representations, as is illustrated by experiments in simulation as well as with real robot data.

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbstständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Freiburg, den 19. Juni 2007

Kai M. Wurm

Danksagung

An dieser Stelle möchte ich mich bei allen Personen bedanken, die mir bei der Erstellung dieser Arbeit geholfen haben. An erster Stelle möchte ich Prof. Dr. Wolfram Burgard danken, der diese Arbeit durch seine Unterstützung ermöglicht hat. Ich danke weiterhin Prof. Dr. Bernhard Nebel dafür, dass er sich als Zweitgutachter zur Verfügung gestellt hat. Mein besonderer Dank gilt Dr. Cyrill Stachniss für die exzellent Betreuung während der gesamten Arbeit. Ich bedanke mich bei allen Mitarbeitern der Arbeitsgruppe AIS für das gute Arbeitsklima, sowie die vielen guten Ideen und Diskussionen. Besonders möchte ich hier Dr. Giorgio Grisetti danken. Ich danke insbesondere auch Axel Rottmann für seine wertvollen Hinweise zu maschinellem Lernen und seine Hilfe bei der Durchführung der Experimente. Des weiteren danke ich der Badischen Zeitung für die Bereitstellung künstlicher Landmarken für meine Experimente. Ich danke Andrea Büscher für ihre Unterstützung und Geduld. Nicht zuletzt danke ich meinen Eltern, die mir dieses Studium ermöglicht haben.

Inhaltsverzeichnis

1. Einleitung

Die moderne Gesellschaft kennt unzählige Automaten und Geräte. Seit hunderten von Jahren konstruieren Wissenschaftler und Erfinder Maschinen, die dem Menschen monotone oder schwere Arbeit abnehmen oder aber seiner Unterhaltung dienen. Die Automatisierung hat inzwischen in fast jeden Bereich des Lebens Einzug gehalten: von der Nahrungsproduktion über die Arbeit im Haushalt bis hin zur medizinischen Versorgung.
Viele dieser Maschinen können nur eine bestimmte Funktion erfüllen und sind zudem an einen festen Ort gebunden. Oftmals müssen sie außerdem von Menschen gesteuert oder überwacht werden. Die mobile Robotik beschäftigt sich hingegen mit Maschinen, die sich selbst bewegen und Aufgaben selbstständig erfüllen können. Solche Aufgaben umfassen den Transport von Gütern aller Art, die Überwachung von Gebäuden, die selbstständige Erkundung unzugänglicher oder gefährlicher Umgebungen, die automatische Wartung von Rohrsystemen oder Industrieanlagen und die Unterstützung des Menschen im Haushalt.
All diese Tätigkeiten erfordern eine geeignete Repräsentation der Umgebung, in der sich ein Robotersystem aufhält. Um in der Umwelt sicher und effizient navigieren zu können, muss ein mobiler Roboter in der Lage sein, seine Position in der Umgebung zu bestimmen. Dazu stehen in der Regel Umweltsensoren wie Kameras oder Abstandssensoren zur Verfügung. Sofern die Umgebung nicht mit für den Roboter lesbaren Positionsmarkierungen verändert werden soll, kommen universelle Umgebungskarten zum Einsatz. In einfach strukturierte Umgebungen, wie beispielsweise Lagerhallen für Güter mit genormter Größe und vordefinierter Position, können unter Umständen manuell erstellte Karten verwendet werden. In typischen Büroumgebungen mit vielen verschiedenen Hindernissen, wie Wänden, Schränken, Tischen, Stühlen usw. ist eine solche Form der Kartierung jedoch nicht einsetzbar, da insbesondere jede Änderung der Umgebung eine manuelle Aktualisierung der Karte erfordern würde. Die automatische Erstellung einer Umgebungskarte aus den Sensorinformation des Roboters ist daher im Allgemeinen die effizienteste Methode der Kartierung.

1.1. Grundproblem Odometrie

Falls die Startposition des Roboters bekannt wäre und der Roboter alle Bewegungsbefehle fehlerfrei ausführen könnte, wäre die Kartierung denkbar einfach. Nach jedem Fahrkommando ließe sich die aktuelle Position des Roboters berechnen und die entsprechenden Sensormessun-

gen in eine Karte eintragen. Auf diese Weise würde schließlich eine komplette Umgebungskarte aufgebaut.

Leider ist dies in der Realität nicht möglich, da selbst ein einfaches Fahrkommando an den Roboter wie "`fahre 1 m vorwärts`" in vielen verschiedenen Zielpositionen resultieren kann. Abhängig von der Genauigkeit der Roboterhardware und der Bodenbeschaffenheit fährt der Roboter vielleicht etwas zu weit, nicht weit genug oder fährt eine leichte Kurve. Gewöhnlich wird die gefahrene Distanz mithilfe von Kodierscheiben an den Rädern des Roboters gemessen. Diese Art von Messungen werden Odometrie genannt. Tatsächlich wird so jedoch lediglich die Rotation der Räder gemessen - vor allem in Kurvenfahrten oder auf unebenem Untergrund entspricht diese nicht der real gefahrenen Distanz. Solche kleinen Fehler haben in Summe dramatische Auswirkungen auf das Kartierungsergebnis, wie Abbildung 1.1 veranschaulicht.

Es ist daher unerlässlich, Odometriefehler zu korrigieren, um eine exakte Karte der Umgebung zu erstellen. Eine solche Korrektur könnte durch die das Wissen über die Umgebung mithilfe der Sensoren des Roboters geschehen. Idealerweise würden die aktuellen Sensormessungen mit einer Karte der Umgebung verglichen, um so die tatsächliche Position zu schätzen. Zum Zeitpunkt der Kartierung steht allerdings noch keine vollständige Umgebungskarte zur Verfügung. Gewissermassen bedingen sich also die Kenntnis der Position des Roboters und die Erstellung einer Umgebungskarte gegenseitig.

In der Vergangenheit wurden verschiedene Verfahren vorgestellt, um dieses Grundproblem der mobilen Robotik zu lösen. Von entscheidender Bedeutung ist dabei die Art und Weise, wie die Umwelt in der Karte repräsentiert wird. Viele Methoden verwenden eine Menge von markanten Punkten in der Umgebung, um die Position des Roboters zu bestimmen. Diese naheliegende Art der Positionsbestimmung wird seit Jahrhunderten erfolgreich in der Navigation zu Land, zu Wasser und in der Luft verwendet. Daneben werden in der Robotik häufig Rasterkarten verwendet, die die Umwelt in kleine Bereiche aufteilt und diese dann als belegt oder frei markiert. Besonders bei der Verwendung von Abstandssensoren führt diese Art der Umweltrepräsentation zu einer sehr genauen Positionsbestimmung. Beide Umweltmodell haben Vor- und Nachteile. Diese Arbeit strebt daher eine Kombination beider Repräsentation an, um deren Vorteile zu nutzen.

Abbildung 1.1.: Auswirkung der Odometriefehler. Die obere Abbildung zeigt eine Karte des Versuchfelds des Fraunhofer IPA (Stuttgart). Sie basiert auf den durch die Odometrie berechneten Roboterpositionen. Die untere Abbildung zeigt eine Karte der selben Umgebung, die anhand von korrigierten Odometriemessungen erstellt wurde. Der geschätzte Pfad des Roboters ist in grün bzw. rot zu sehen.

1.2. Zielsetzung

In dieser Arbeit wird ein System zur robusten Kartierung anhand von Sensormessungen eines Roboters vorgestellt. Das System soll in der Lage sein, sowohl in Umgebungen mit vielen Hindernissen (beispielsweise dem Inneren von Gebäuden), als auch auf weitläufigen Freiflächen gute Kartierungsergebnisse zu liefern. Dies wird durch die Kombination zweier verschiedener Umgebungsrepräsentationen erreicht. Das kombinierte Verfahren soll sich die Vorteile beider Umgebungsrepräsentationen zunutze machen und so ein besseres Ergebnis erreichen, als jeweils mit nur einem der beiden Modelle möglich gewesen wäre.

1.3. Aufbau der Arbeit

In Kapitel 2 wird diese Arbeit zunächst in einen größeren Kontext wissenschaftlicher Arbeiten eingeordnet. Die Grundlagen dieser Arbeit werden in Kapitel 3 erläutert. Ausgehend von allgemeinen Techniken zur Zustandsschätzung werden die grundlegenden Verfahren zur Kartierung und Lokalisierung in der Robotik, sowie die verwendeten Umgebungsmodelle vorgestellt. In Kapitel 4 wird das in dieser Arbeit beschriebene Verfahren im Detail erläutert. Der Schwerpunkt des Abschnitts liegt auf der Auswahlstrategie, nach der entschieden wird, welche der beiden Umweltrepräsentationen zur Positionskorrektur verwendet wird. Wir stellen ein Lernverfahren vor, um diese Entscheidung anhand von Beispielumgebungen zu erlernen. Das entwickelte System wird in einer Reihe von Experimenten in Simulation und mit echten Roboterdaten evaluiert. Diese Experimente und deren Ergebnisse werden in Kapitel 5 vorgestellt. Kapitel 6 enthält eine abschließende Zusammenfassung der Ergebnisse und einen Ausblick über mögliche weitere Arbeiten.

1.4. Publikation

Teile dieser Arbeit werden auf der *3rd European Conference on Mobile Robots (ECMR)* vorgestellt [Wurm u. a., 2007].

2. Verwandte Arbeiten

Das Erstellen von Umgebungskarten aus Daten eines mobilen Roboters wird allgemein als schwierige Aufgabe angesehen [Smith u. a., 1990; Dissanayake u. a., 2000; Doucet u. a., 2000; Eliazar u. Parr, 2003; Gutmann u. Konolige, 1999; Hähnel u. a., 2003; Montemerlo u. a., 2003, 2002b; Murphy, 1999; Thrun, 2001; Grisetti u. a., 2007a]. Der Grund hierfür liegt in der Tatsache, dass zur korrekten Kartierung der Umgebung die genaue Position des Roboters bekannt sein muss. Andererseits wird zur Lokalisierung des Roboters bereits eine Karte benötigt. Das zugrundeliegende Problem wird im Englischen *Simultanious Localization and Mapping (SLAM)* genannt. Kartierungsverfahren für mobile Roboter werden seit mehr als zwanzig Jahren untersucht. Es wurde eine Vielzahl von Methoden vorgestellt, die sich grob nach der verwendeten Methode zur Zustandsschätzung und nach der zugrundeliegenden Umweltrepräsentation einteilen lassen.

Die bekannteste Form der Kartierung beruht auf der Extraktion von definierten Landmarken aus Sensordaten. Landmarken haben den Vorteil, dass sie beispielsweise durch geometrische Beschreibungen oder Vektoren kompakt repräsentiert werden können. Allerdings müssen die erwarteten Landmarken bereits im Vorhinein definiert werden, was voraussetzt, dass die Struktur der Umgebung bekannt ist.

Eine weitere häufig verwendete Repräsentation der Umgebung sind Rasterverfahren, die den kontinuierlichen Raum in gleich große, diskrete Teile aufteilen und diesen eine Belegtheitswahrscheinlichkeit zuweist. Diese Art der Modellierung wurde erstmals von Moravec und Elfes vorgestellt [Moravec u. Elfes, 1985]. Sie bietet den Vorteil, beliebige Hindernisse abbilden zu können, weist jedoch einen hohen Ressourcenbedarf auf [Stachniss, 2006].

Neben dem verwendeten Kartentyp ist die Methode der Positionsschätzung oder allgemein der Zustandsschätzung von entscheidender Bedeutung für ein Kartierungsverfahren. Häufig verwendete Verfahren umfassen Gausssche Filter wie den Erweiterten Kalmanfilter und parameterfreie Filter wie den Partikelfilter. Gausssche Filter modellieren den Zustandsvektor durch mehrdimensionale Normalverteilungen, während parameterfreie Filter beliebige Verteilungen durch eine endliche Menge von Stichproben darstellen [Thrun u. a., 2005].

Eines der am besten untersuchten Kartierungsverfahren verwendet einen Erweiterten Kal-

manfilter (EKF) in Verbindung mit Landmarken. Die Leistungsfähigkeit dieses Ansatzes ist in der Tatsache begründet, dass eine vollständig korrelierte Verteilung von Landmarken und Roboterpositionen geschätzt wird [Leonard u. Durrant-Whyte, 1991; Smith u. a., 1990]. Das Verfahren beruht allerdings auf starken Annahmen bezüglich der Bewegung eines Roboters und der Beschaffenheit der Sensoren. Falls diese Annahmen verletzt werden, ist es möglich, dass der Filter divergiert und die Kartierung fehlschlägt [Frese, 2006a; Julier u. a., 1995; Uhlmann, 1995]. Darüber hinaus gehen einige Verfahren davon aus, dass individuelle Landmarken eindeutig identifiziert werden können. Es existieren jedoch Techniken, die mit Unsicherheiten in der Assoziierung von gemessenen Landmarken und solchen in der Karte umgehen können [Neira u. Tardós, 2001].

Während Kalmanfilter eine Normalverteilung durch einen mehrdimensionalen Mittelwert und eine Kovarianzmatrix repräsentieren, verwendet der Informationsfilter die inverse Darstellung der Kovarianzmatrix, die auch Informationsmatrix genannt wird. Im Kontext des SLAM Problems, wurde ein Erweiterter Informationfilter erstmalig von Nettleton u. a. [2000a,b] verwendet. Diese Technik bietet Laufzeitvorteile gegenüber dem EKF, besitzt aber die gleiche Mächtigkeit. Auch Thrun u. a. [2004] verwenden die Informationsmatrix zur Darstellung der Unsicherheit. Diese ändert sich bei der Integrierung neuer Daten nur an wenigen Stellen, während sich die Mehrzahl der Werte der Kovarianzmatrix ändert. Diese Eigenschaft zusammen mit weiteren Approximationen ermöglichen es Thrun u. a. das Aktualisieren des Filters und die Schätzung des Zustands in amortisiert konstanter Zeit durchzuführen. Eustice u. a. [2005] haben eine Erweiterung dieses Verfahrens vorgestellt, dass alle Roboterpositionen mitschätzt und eine konservative Unsicherheitsmodellierung erlaubt. Eine ähnliche, alternative Repräsentation der geschätzten Wahrscheinlichkeitsverteilung, die eine Baumstruktur verwendet, wurden von Paskin [2003] und Frese [2006b] vorgestellt. Die Effizienz dieses Verfahrens beruht auf dem Ignorieren kleiner Zustandskorrelationen.

Eine weitere Klasse von Algorithmen betrachtet das SLAM-Problem als Optimierungsproblem. Roboterpositonen werden hier als Knoten eines Graphs verwaltet. Durch Sensormessungen und Odometriewerte ergeben sich Beziehungen zwischen den Positionen, die als Kanten des Graphs repräsentiert werden. Der wahrscheinlichste Roboterpfad wird mithilfe eines Optimierungsverfahren bestimmt. Aus dem Pfad und den gespeicherten Messungen kann dann anschließend eine korrigierte Karte konstruiert werden. Dieser Ansatz wurde zuerst von Lu u. Milios [1997] vorgestellt. Die Optimierung wird hier für die gesamte Karte durch eine Reihe von aufwändigen Matrixoperationen durchgeführt, was sich in der Praxis als sehr zeitaufwändig herausgestellt hat. Aus diesem Grund verwenden eine Reihe von späteren Arbeiten iterative Optimierungsverfahren [Howard u. a., 2001; Duckett u. a., 2002; Frese u. a., 2005; Olson u. a., 2006; Grisetti u. a., 2007b]. Dieser unterscheiden sich im verwendeten Optimierungs-

verfahren. Howard und Duckett verwenden Gauss-Seidel Relaxation um den quadratischen Fehler zu minimieren. Frese u. a. konnten die Qualität der Korrektur deutlich verbessern, in dem er eine hierarchische Struktur in den Graphen eingeführt hat. Im Gegensatz dazu verwenden die Verfahren von Olson u. a. und Grisetti u. a. Varianten des Gradientenabstiegs. Olsons Algorithmus, sowie die Verfahren von Frese und Grisetti zählen heute zu den bestem Graphoptimierern im Kontext des SLAM-Problems.

In Arbeiten von Murphy, Doucet und anderen Forschern [Doucet u. a., 2000; Murphy, 1999] wurden Rao-Blackwellized Partikelfilter (RBPF) vorgestellt, um das SLAM-Problem zu lösen. Partikelfilter repräsentieren Wahrscheinlichkeitsverteilungen durch eine endliche Anzahl von Stichproben. Jede Stichprobe in einem RBPF repräsentiert einen möglichen Roboterpfad und eine zugehörige Umgebungskarte. Auf dieser Grundlage wurde von Montemerlo u. a. [2003, 2002b] ein SLAM-Ansatz mit Landmarken vorgestellt. Dabei wird eine gausssche Darstellung der Landmarken und Roboterbewegung verwendet. Um akkurate metrische Karten zu lernen, wurden RBPFs von Eliazar u. Parr [2003] sowie Hähnel u. a. [2003] in Verbindung mit Rasterkarten verwendet. Die erste Arbeit stellt eine effiziente Kartenrepräsentation vor. Sie basiert darauf, dass verschiedene Partikel Kartenabschnitte gemeinsam verwenden können. Dies führt zu einem deutlich reduzierten Resourcenbedarf. Ähnliche Techniken finden sich auch in [Grisetti u. a., 2007c]. Der Ansatz von Hähnel u. a. integriert ein Scanmatching-Verfahren in die Berechnung der Vorhersageverteilung. Dieses Vorgehen erlaubt es, die Anzahl der Stichproben und damit den Ressourcenbedarf zu reduzieren. Eine aktuelle Arbeit von Grisetti u. a. [2007a] beschreibt eine weitere Verbesserung dieses Verfahrens, das Ideen des FastSLAM2-Algorithmus von Montemerlo u. a. [2003] verwendet. Es berechnet eine optimierte Vorhersageverteilung, die die aktuelle Messung berücksichtigt.

Bisher existieren nur sehr wenige Methoden, die eine Kombination von Landmarken und Rasterkarten anstreben. Eines dieser Verfahren ist der *Hybrid Metric Map (HYMM)* Ansatz von Nieto u. a. [2004]. Basis des Verfahrens ist die Schätzung von Landmarkenpositionen. Aus diesen wird eine Triangulierung berechnet, die eine Segmentierung der Umgebung darstellt. Für jedes der Segmente werden detaillierte Karten *(dense maps)* erstellt, die beliebige Informationen über die Areale beinhalten können, insbesondere können hier auch Rasterkarten verwendet werden. Ändert sich die Position einer Landmarke, werden die detaillierten Karten entsprechend transformiert. Das Verfahren ist jedoch in jedem Fall auf die zuverlässige Extrahierung von Landmarken aus der Umgebung angewiesen. Informationen der detaillierten Karten werden nicht zur Zustandsschätzung genutzt, der Ansatz ist daher nicht so robust, wie die in dieser Arbeit vorstellte Technik.

2.1. Beitrag der Arbeit

Der Beitrag der vorliegenden Arbeit ist die Entwicklung eines kombinierten SLAM-Verfahrens, welches sowohl Rasterkarten, als auch Landmarken verwendet und so „das beste aus zwei Welten“ kombiniert. Beide Umweltrepräsentationen werden parallel verwaltet. Durch die Kombination von Umgebungsmodellen mit verschiedenen Charakteristiken wird der robuste Einsatz in unterschiedlichen Umgebungen ermöglicht. Das erlaubt unserem Verfahren, Modelle von Umgebungen zu erstellen, die durch bisherige Verfahren, die entweder nur Grid- oder Landmarkenkarten verwenden, nicht konsistent modelliert werden können. Zur Zustandsschätzung wird in jedem Zeitschritt das jeweils am besten geeignete Modell der Umgebung gewählt. Die Auswahl erfolgt nach einer mittels *Reinforcment-Learning* gelernten Strategie. Das Verfahren basiert auf einem Rao-Blackwellized Partikelfilter und baut auf dem Ansatz von Grisetti u. a. [2007a] auf. Es ist unabhängig von der Wahl der konkreten Landmarkendefinition.

3. Grundlagen

Typischerweise werden in der mobilen Robotik Aktionen in Abhängigkeit bestimmter gemessener oder errechneter Kenngrößen ausgeführt. Angenommen, ein Roboter habe die Aufgabe, ein Paket in ein bestimmtes Büro zu bringen. Ein Teilziel könnte dann darin bestehen, die Tür zu öffnen, wenn der Roboter vor dem entsprechenden Büro steht. Nehmen wir weiter an, dass diese Tür eine von vielen gleichartigen Türen sei und es keine direkte Möglichkeit gäbe, sie von den anderen zu unterscheiden. Um die Aufgabe erfüllen zu können, benötigt der Roboter nun mehrere Informationen, die nicht direkt beobachtbar sind: eine Karte des Gebäudeinneren, seine aktuelle Position relativ zu der Gebäudekarte und die Information, wo sich der Eingang zum gesuchten Büro befindet. Diese Art von Informationen über den Roboter selbst und seine Umgebung nennen wir *Zustand*. Der Zustand ändert sich in der Regel im Laufe der Zeit, zum Beispiel wenn sich der Roboter bewegt. Eine wichtige Aufgabe eines autonomen Systems ist daher die Schätzung des aktuellen Zustandes. Diese Schätzung basiert einerseits auf den Bewegungskommandos des Roboters und andererseits auf den Daten der am Roboter angebrachten Sensoren.

Da der Zustand nicht vollständig beobachtbar ist, kann er nur mit einer gewissen Wahrscheinlichkeit angegeben werden. Die Verwendung probabilistischer Verfahren ist daher notwendig. Im Folgenden werden einige häufig verwendete Verfahren zur Zustandsschätzung vorgestellt, die auch in dem hier vorgestellten System zur Anwendung kommen. Die formalen Beschreibungen stammen in weiten Teilen aus [Thrun u. a., 2005]. Die hier verwendeten mathematischen Sätze und Zusammenhänge aus dem Bereich der Wahrscheinlichkeitsrechnung finden sich im Anhang.

3.1. Bayesfilter

Den meisten Robotern stehen zur Schätzung des aktuellen Zustands verschiedene Daten zur Verfügung. Einerseits erlauben die vom Roboter durchgeführten Aktionen eine Vorhersage des aus ihnen resultierenden Zustands, andererseits können Sensormessungen verwendet werden, um diese Vorhersage zu korrigieren. Es ist üblich, die Schätzung eines Zustands x durch bedingte Warscheinlichkeitsverteilungen $P(x \mid z_{1..t}, u_{1..t})$ anzugeben. Dabei bezeichnet u_1 die erste Aktion, z_1 die erste Messung, u_2 die zweite Aktion und so weiter.

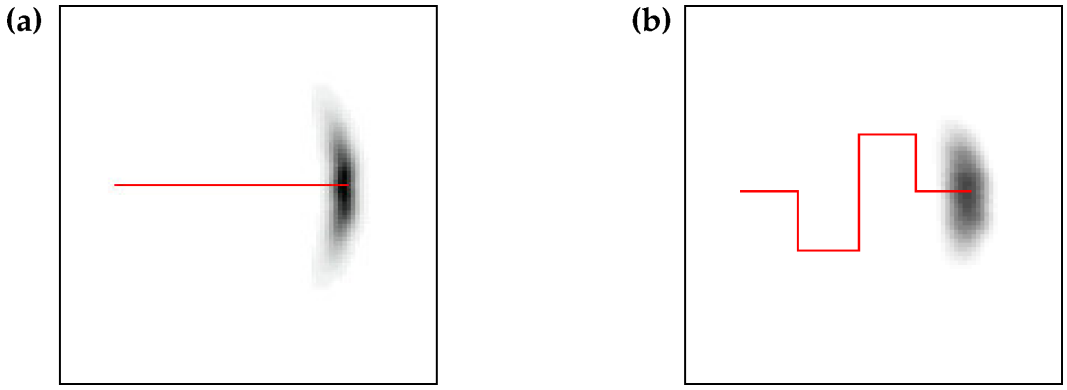


Abbildung 3.1.: Mögliche Roboterpositionen nach der Ausführung von Fahrkommandos unter Annahme eines normalverteilten Fehlers in den Bewegungskomponenten. Die roten Linien veranschaulichen die Kommandosequenzen. Je dunkler eine Position dargestellt ist, desto wahrscheinlicher ist sie (aus [Thrun u. a., 2005]).

Wie eingangs erläutert, birgt die Ausführung einer Aktion immer eine gewisse Unsicherheit. Diese Unsicherheit lässt sich durch ein *Aktionsmodell* $P(x \mid u, x')$ formalisieren. Es gibt die Übergangswahrscheinlichkeit von Zustand x' zu x bei der Durchführung von Aktion u an. Beispielsweise nimmt man bei fahrenden Robotern typischerweise einen normalverteilten Fehler in der Ausführung von Bewegungskommandos an. Genauer gesagt werden Fehler in den einzelnen Bewegungskomponenten wie Drehung und Geradeausfahrt durch Normalverteilungen modelliert. Abbildung 3.1 illustriert ein solches Aktionsmodell und die resultierende typische Zustandsverteilung.

Integriert man über alle möglichen Vorgängerzustände x', so erlaubt das Aktionsmodell eine Schätzung der Wahrscheinlichkeit eines Zustands x nachdem Aktion u ausgeführt wurde:

$$P(x \mid u) = \int P(x \mid u, x')\ P(x')\ dx' \tag{3.1}$$

Sofern ein System über Umweltsensoren verfügt, können diese verwendet werden, um die Positionsschätzung zu verbessern[1]. Die grundlegende Frage ist dabei, wie wahrscheinlich eine Messung z in einem Zustand x ist. Diese Wahrscheinlichkeit wird durch ein *Sensormodell* $P(z \mid x)$ angegeben, welches vom verwendeten Sensor abhängt. In dieser Arbeit werden vor allem Robotersysteme mit Laserscannern untersucht. Dabei handelt es sich um einen Sensor, der mithilfe eines Infrarotlasers sehr genaue Abstandsmessungen durchführen kann.

Mithilfe des Aktions- und Sensormodells ist es möglich, den Zustand eines Systems nach der

[1] Odometriemessungen werden allgemein nicht als Messungen sondern als Fahrkommandos interpretiert, da sie mit diesen im direkten Zusammenhang stehen.

Algorithmus 1 Bayesfilter

Eingabe:
$Bel(x_{t-1})$ bisherige Schätzung
d aktuelle Daten
Ausgabe:
$Bel(x_t)$ aktualisierte Schätzung

// Vorhersage des Zustands
if d ist Aktion **then**
  for all x_t **do**
    $Bel(x_t) = \int P(x \mid u, x_{t-1})\ Bel(x_{t-1})\ dx_{t-1}$
  end for
end if

// Korrektur der Schätzung
if d ist Messung **then**
  $\eta = 0$
  for all x_t **do**
    $Bel(x_t) = P(z_t \mid x_t)\ Bel(x_{t-1})$
    $\eta = \eta + Bel(x_t)$
  end for
  for all x_t **do**
    $Bel(x_t) = \eta^{-1}\ Bel(x_t)$
  end for
end if

Durchführung von Aktionen u_1 bis u_t und nach Beobachtung von Messungen z_1 bis z_t zu schätzen. Der in Algorithmus 1 beschriebene *Bayesfilter* bildet den formalen Rahmen für die in diesem Kapitel beschriebenen Algorithmen zur Zustandsschätzung. Er basiert auf folgender rekursiver Gleichung:

$$\begin{aligned} Bel(x_t) &:= P(x_t \mid z_{1..t}, u_{1..t}) && (3.2)\\ &= \eta\ P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1})\ Bel(x_{t-1})\ dx_{t-1} && (3.3) \end{aligned}$$

Oftmals werden Aktionen und Messungen wie in Algorithmus 1 getrennt integriert. Aus einer Aktion u und der bisherigen Zustandsschätzung $Bel(x_{t-1})$ wird zunächst eine Zustandsvorhersage errechnet:

$$\overline{Bel}(x_t) = \int P(x_t \mid u_t, x_{t-1})\ Bel(x_{t-1})\ dx_{t-1} \tag{3.4}$$

Diese wird dann anhand einer Messung z_t korrigiert:

$$Bel(x_t) = \eta\ P(z_t \mid x_t)\ \overline{Bel}(x_t) \tag{3.5}$$

Der Normalisierungsfaktor η ergibt sich dabei aus der Anwendung des Satzes von Bayes.

Die Herleitung von Gleichung (3.3) verwendet zwei Unabhängigkeitsannahmen. Erstens wird angenommen, dass der Nachfolgezustand des Systems nur von der aktuellen Aktion und dem aktuellen Zustand beeinflusst wird (*Markov-Annahme für Aktionen*):

$$P(x_t \mid x_{0..t-1}, z_{1..t-1}, u_{1..t}) = P(x_t \mid x_{t-1}, u_t) \tag{3.6}$$

Außerdem wird angenommen, dass die aktuelle Messung z_t unabhängig von den vorigen Messungen, Aktionen und Zuständen ist, falls der aktuelle Zustand x_t bekannt ist (*Markov-Annahme für Messungen*):

$$P(z_t \mid x_{0..t}, z_{1..t-1}, u_{1..t}) = P(z_t \mid x_t) \tag{3.7}$$

Damit ergibt sich:

$$\begin{aligned}
Bel(x_t) &= P(x_t \mid z_{1..t}, u_{1..t}) && (3.8)\\
&\overset{\text{Bayes}}{=} \eta\, P(z_t \mid x_t, z_{1..t-1}, u_{1..t})\, P(x_t \mid z_{1..t-1}, u_{1..t}) && (3.9)\\
&\overset{(3.7)}{=} \eta\, P(z_t \mid x_t) \qquad P(x_t \mid z_{1..t-1}, u_{1..t}) && (3.10)\\
&\overset{\text{total. Ws.}}{=} \eta\, P(z_t \mid x_t) \int P(x_t \mid x_{t-1}, z_{1..t-1}, u_{1..t}) \\
&\qquad\qquad P(x_{t-1} \mid z_{1..t-1}, u_{1..t})\, dx_{t-1} && (3.11)\\
&\overset{(3.6)}{=} \eta\, P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \\
&\qquad\qquad P(x_{t-1} \mid z_{1..t-1}, u_{1..t})\, dx_{t-1} && (3.12)\\
&= \eta\, P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1})\, Bel(x_{t-1})\, dx_{t-1} && (3.13)
\end{aligned}$$

Das Hauptproblem bei der Implementierung des Bayesfilters ist, dass $Bel(x_t)$ bis auf wenige Spezialfälle nicht exakt bestimmt werden kann [Thrun u. a., 2005]. Im Folgenden werden einige bekannte Implementierungen angegeben, die auf verschiedenen Näherungen beruhen.

3.2. Kalmanfilter

Der Kalmanfilter [Kalman, 1960] ist eine der bekanntesten Implementierungen des Bayesfilter. Zustandshypothesen werden durch eine Normalverteilung[2] $N(\mu_t; \Sigma_t)$ repräsentiert, wobei der Mittelwert μ_t dem geschätzten Zustand entspricht und die Varianz Σ_t die Unsicherheit in der Schätzung ausdrückt. Der Filter wurde für *Lineare Gausssche Systeme* entwickelt, in denen folgende Bedingungen gelten müssen [Thrun u. a., 2005]:

1. Die Zustandsübergänge des Systems lassen sich durch eine lineare Funktion beschreiben:

$$x_t = A_t \, x_{t-1} + B_t \, u_t + \epsilon_t \tag{3.14}$$

   A_t und B_t sind Matrizen. A_t drückt die Zustandsänderung ohne Aktionen oder Rauschen aus und B_t gibt die Zustandsänderung durch Aktion u_t an. ϵ_t modelliert die Unsicherheit des Aktionsmodells. Sie wird durch eine Normalverteilung $N(0; R_t)$ angeben. Aus den im Anhang aufgeführten Eigenschaften der Normalverteilung ergibt sich folgendes Aktionsmodell:

$$P(x_t \mid u_t, x_{t-1}) \sim N(A_t \, x_{t-1} + B_t \, u_t; R_t) \tag{3.15}$$

2. Messungen können durch eine lineare Funktion in Abhängigkeit des Zustands angegeben werden:

$$z_t = C_t \, x_t + \delta_t \tag{3.16}$$

   C_t ist eine Matrix und bildet Zustände auf Messungen ab. δ_t beschreibt die Unsicherheit des Sensormodells und wird durch eine Normalverteilung $N(0; Q_t)$ ausgedrückt. Daraus folgt ein normalverteiltes Sensormodell:

$$P(z_t \mid x_t) \sim N(C_t \, x_t; Q_t) \tag{3.17}$$

3. Die initiale Zustandsschätzung ist normalverteilt:

$$x_0 \sim N(\mu_0; \Sigma_0) \tag{3.18}$$

Die Herleitung der in Algorithmus 2 angegebenen Berechnungsschritte beruht auf diesen Voraussetzungen und den oben erwähnten Markov-Annahmen (3.6) und (3.7). Es kann gezeigt werden, dass die Zustandsschätzung zu jeder Zeit einer Normalverteilung entspricht.

[2] Definition und Eigenschaften der Normalverteilung finden sich im Anhang.

Algorithmus 2 Kalmanfilter

Eingabe:

μ_{t-1}, Σ_{t-1} bisherige Schätzung
u_t, z_t aktuelle Daten

Ausgabe:

μ_t, Σ_t aktualisierte Schätzung

1: $\overline{\mu}_t = A_t\ \mu_{t-1} + B_t\ u_t$
2: $\overline{\Sigma}_t = A_t\ \Sigma_{t-1}\ A_t^T + R_t$

3: $K_t = \overline{\Sigma}_t\ C_t^T (C_t\ \overline{\Sigma}_t\ C_t^T + Q_t)^{-1}$
4: $\mu_t = \overline{\mu}_t + K_t\ (z_t - C_t\ \overline{\mu}_t)$
5: $\Sigma_t = (I - K_t\ C_t)\ \overline{\Sigma}_t$

Der Algorithmus gliedert sich grob in zwei Teile. Zunächst wird basierend auf der aktuellen Aktion u_t eine Vorhersage $\langle \overline{\mu}_t, \overline{\Sigma}_t \rangle$ berechnet. Diese wird dann im zweiten Schritt anhand der aktuellen Messung z_t korrigiert.

Die Komplexität des Algorithmus wird im Wesentlichen von der Matrixinversion in Zeile 3 dominiert. Insgesamt ergibt sich eine Komplexität von $O(k^{2.4} + n^2)$, wobei k die Dimension der Messungen und n die Dimension des Zustandsvektors angibt.

In der Praxis sind Zustandsübergänge und Messungen selten linear. Sobald sich beispielsweise ein Roboter auf einer Kreisbahn bewegt, kann das Aktionsmodell nicht mehr durch eine lineare Funktion beschrieben werden. Aus diesem Grund wurden verschiedene Varianten des Kalmanfilters entwickelt, von denen der Erweiterte Kalmanfilter eine der bekanntesten ist.

3.2.1. Erweiterter Kalmanfilter (EKF)

Der Erweiterte Kalmanfilter erlaubt die Verwendung nicht-linearer Übergangs- und Messungsfunktionen g und h:

$$x_t = g(u_t, x_{t-1}) + \epsilon_t \tag{3.19}$$

$$z_t = h(x_t) + \delta_t \tag{3.20}$$

Da die Herleitung des Kalmanfilters auf der Annahme linearer Übergänge und Messungen beruht, müssen g und h linearisiert werden. Der EKF erreicht dies durch Taylor-Entwicklung an der Stelle μ_{t-1}. Im Allgemeinen handelt es sich dabei um eine Näherung der tatsächlichen Funktionen. Andere Methoden der Linearisierung sind ebenfalls möglich und liefern unter Umständen bessere Ergebnisse [Thrun u. a., 2005].

Algorithmus 3 Erweiterter Kalmanfilter

Eingabe:

μ_{t-1}, Σ_{t-1}	bisherige Schätzung
u_t, z_t	aktuelle Daten

Ausgabe:

μ_t, Σ_t	aktualisierte Schätzung

1: $\overline{\mu}_t = g(u_t, \mu_{t-1})$
2: $\overline{\Sigma}_t = G_t \; \Sigma_{t-1} \; G_t^T + R_t$

3: $K_t = \overline{\Sigma}_t \; H_t^T (H_t \; \overline{\Sigma}_t \; H_t^T + Q_t)^{-1}$
4: $\mu_t = \overline{\mu}_t + K_t \; (z_t - h(\overline{\mu}_t))$
5: $\Sigma_t = (I - K_t \; H_t) \; \overline{\Sigma}_t$

Linearisierung des Aktionsmodells durch Taylorentwicklung:

$$g'(u_t, x_{t-1}) = \frac{\partial g(u_t, x_{t-1})}{\partial x_{t-1}} \tag{3.21}$$

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \underbrace{g'(u_t, \mu_{t-1})}_{=:G_t} (x_{t-1} - \mu_{t-1}) \tag{3.22}$$

$$\approx g(u_t, \mu_{t-1}) + G_t \; (x_{t-1} - \mu_{t-1}) \tag{3.23}$$

Linearisierung des Sensormodells:

$$h'(x_t) = \frac{\partial h(x_t)}{\partial x_t} \tag{3.24}$$

$$h(x_t) \approx h(\overline{\mu_t}) + \underbrace{h'(\overline{\mu_t})}_{=:H_t} (x_t - \overline{\mu_t}) \tag{3.25}$$

$$\approx h(\overline{\mu_t}) + H_t \; (x_t - \overline{\mu_t}) \tag{3.26}$$

Durch die Linearisierung bleiben Zustandsübergänge und Messvorhersagen normalverteilt:

$$P(x_t \mid u_t, x_{t-1}) \sim N(g(u_t, \mu_{t-1}) + G_t \; (x_{t-1} - \mu_{t-1}); R_t) \tag{3.27}$$

$$P(z_t \mid x_t) \sim N(\mu_t + H_t(x_t - \overline{\mu_t}); Q_t) \tag{3.28}$$

Algorithmus 3 fasst die Berechnungschritte zusammen. Die Funktionsweise des Erweiterten Kalmanfilters soll im Folgenden durch ein Beispiel illustriert werden.

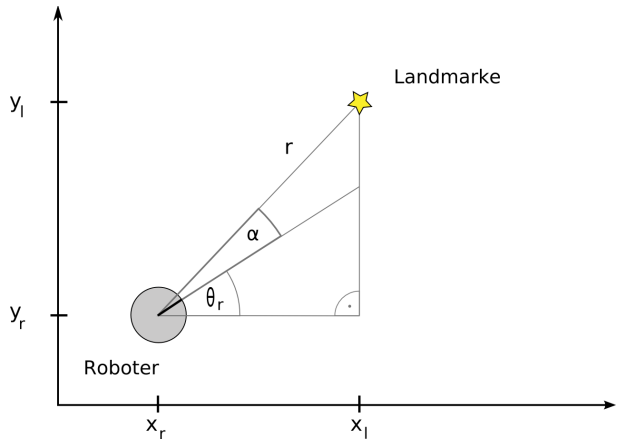


Abbildung 3.2.: Eine Landmarke (als Stern dargestellt) wird in Entfernung r und unter Winkel α erkannt. Ihre Position $\langle x_l, y_l \rangle$ soll relativ zur Roboterposition $\langle x_r, y_r, \theta_r \rangle$ geschätzt werden.

3.2.2. Beispiel: Landmarken-Sensor

Angenommen, ein Roboter verfüge über einen Sensor, der bestimmte markante Punkte der Umgebung erkennen und deren Position relativ zu sich angeben kann. Solche Punkte werden Landmarken genannt und können verschiedener Art sein (wir werden in Abschnitt 3.4 darauf zurückkommen). Es ist üblich, die relative Position einer erkannten Landmarke durch den beobachteten Winkel α zur Vorausrichtung des Roboters und durch die gemessene Entfernung r zum Roboter anzugeben (siehe Abbildung 3.2). Da solche Messungen immer mit einer Unsicherheit behaftet sind, kann die tatsächliche Position einer Landmarke nur geschätzt werden. Zu diesem Zweck kann ein Erweiterter Kalmanfilter verwendet werden. Im Beispiel soll die Position $l = \langle x_l, y_l \rangle$ einer einzelnen Landmarke geschätzt werden.

Aktionen des Roboters haben keine Auswirkung auf die Position der Landmarke, daher muss nur das Sensormodell durch eine Messfunktion $h(l_t)$ und eine entsprechende Fehlervariable δ_t (bzw. deren Varianz Q_t) angegeben werden. Die Position des Roboters $\langle x_r, y_r, \theta_r \rangle$ wird in diesem Beispiel als konstant und bekannt angenommen. In Abschnitt 4.1 werden wir Techniken zur gleichzeitigen Schätzung von Landmarken- und Roboterposition vorstellen.

Die Messfunktion $h(l_t)$ gibt die erwartete Messung in Abhängigkeit der aktuellen Schätzung zurück. Die erwartete Distanz r_t entspricht im Beispiel der Euklidischen Distanz zwischen der geschätzten Landmarkenposition und der bekannten Roboterposition. Der erwartete Winkel α_t lässt sich trigonometrisch ermitteln, wie aus Abbildung 3.2 hervorgeht:

$$h(l_t) = \begin{pmatrix} \sqrt{(x_l - x_r)^2 + (y_l - y_r)^2} \\ \arctan(\frac{y_l - y_r}{x_l - x_r}) - \theta_r \end{pmatrix} = \begin{pmatrix} r_t \\ \alpha_t \end{pmatrix} \tag{3.29}$$

Des weiteren wird die Jacobi-Matrix von h benötigt:

$$H(l_t) = \begin{pmatrix} \frac{\partial h_1(l_t)}{\partial x_l} & \frac{\partial h_1(l_t)}{\partial y_l} \\ \\ \frac{\partial h_2(l_t)}{\partial x_l} & \frac{\partial h_2(l_t)}{\partial y_l} \end{pmatrix} = \begin{pmatrix} \frac{x_l - x_r}{\sqrt{(x_l - x_r)^2 + (y_l - y_r)^2}} & \frac{y_l - y_r}{\sqrt{(x_l - x_r)^2 + (y_l - y_r)^2}} \\ \\ \frac{-(y_l - y_r)}{\left(\left(\frac{y_l - y_r}{x_l - x_r}\right)^2 + 1\right)(x_l - x_r)^2} & \frac{1}{\left(\left(\frac{y_l - y_r}{x_l - x_r}\right)^2 + 1\right)(x_l - x_r)} \end{pmatrix} \tag{3.30}$$

Die Fehlermatrix Q_t sollte die Ungenauigkeit des Landmarkensensors wiederspiegeln. Eine höhere Ungenauigkeit in der Entfernungsmessung könnte beispielsweise folgendermassen modelliert werden:

$$Q_t = \begin{pmatrix} 0.2 & 0 \\ 0 & 0.1 \end{pmatrix}$$

Damit sind alle Komponenten definiert, die für die Positionsschätzung mittels eines Erweiterten Kalmanfilters benötigt werden. Da in diesem Beispiel keine Aktionen durchgeführt werden, werden Zeilen 1 und 2 aus Algorithmus 3 nicht benötigt und in Zeilen 3-6 direkt mit der bisherigen Schätzung l_{t-1} gerechnet.

Sei nun die wahre Position der Landmarke $l = \langle 3.0, 2.5 \rangle$, die bisherige Schätzung der Landmarke $\mu_0 = \begin{pmatrix} 3.0 \\ 3.0 \end{pmatrix}, \Sigma_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ und die Position des Roboters $\langle 1.0, 1.0, 0.0 \rangle$.

Es werden zwei verrauschte Messungen der Landmarke integriert:

$$z_1 = \begin{pmatrix} 2.3m \\ 37^\circ \end{pmatrix} \text{ und } z_2 = \begin{pmatrix} 2.5m \\ 36^\circ \end{pmatrix}$$

Die aktualisierte Positionsschätzung nach der Integration von z_1 lautet:

$$\mu_1 = \begin{pmatrix} 2.85 \\ 2.52 \end{pmatrix}, \Sigma_1 = \begin{pmatrix} 0.3 & -0.14 \\ -0.14 & 0.3 \end{pmatrix}$$

Nach Integration von z_2 entspricht die Schätzung etwa der wahren Position und die Unsicherheit hat sich stark reduziert:

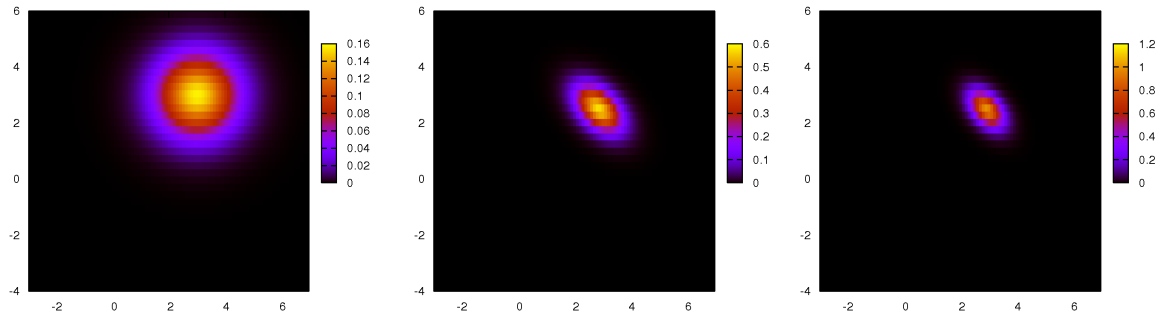


Abbildung 3.3.: Schätzung einer Landmarkenposition. Je heller ein Bereich ist, desto wahrscheinlicher liegt dort die wahre Position der Landmarke. Links: Ausgangsschätzung, Mitte: Schätzung nach Integration einer Messung, rechts: Schätzung nach Integration von zwei Messungen.

$$\mu_2 = \begin{pmatrix} 2.93 \\ 2.5 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 0.16 & -0.08 \\ -0.08 & 0.18 \end{pmatrix}$$

Abbildung 3.3 veranschaulicht diese Reduktion der Unsicherheit in dem Bereiche gleicher Wahrscheinlichkeit farblich dargestellt werden.

3.3. Partikelfilter

Der Partikelfilter ist eine weitere Implementierung des Bayesfilters. Während der Kalmanfilter die gesuchte Wahrscheinlichkeitsverteilung $Bel(x_t)$ durch eine Normalverteilung modelliert, repräsentiert der Partikelfilter die Verteilung durch eine endliche Menge von Stichproben des Zustandsraums:

$$\mathcal{X}_t = \{x_t^{(1)}, x_t^{(2)}, ..., x_t^{(N)}\} \tag{3.31}$$

Jede Stichprobe $x_t^{(i)}$ entspricht einer Zustandshypothese zum Zeitpunkt t und wird *Partikel* genannt. Der Vorteil dieser Methode ist einerseits, dass dadurch die Approximierung verschiedenster Verteilungen ermöglicht wird und andererseits, dass nicht-lineare Transformationen ohne Linearisierung verwendet werden können. Die Wahrscheinlichkeit, mit der eine Zustandshyptothese x_t in die Partikelmenge aufgenommen wird, sollte proportional zu $Bel(x_t)$ sein [Thrun u. a., 2005]:

$$x_t^{(i)} \sim P(x_t \mid z_{1..t}, u_{1..t}) \tag{3.32}$$

Intuitiv folgt daraus, dass der wahre Zustand des Systems desto wahrscheinlicher in einer Region des Zustandsraums liegt, je mehr Partikel aus dieser Region in die Partikelmenge auf-

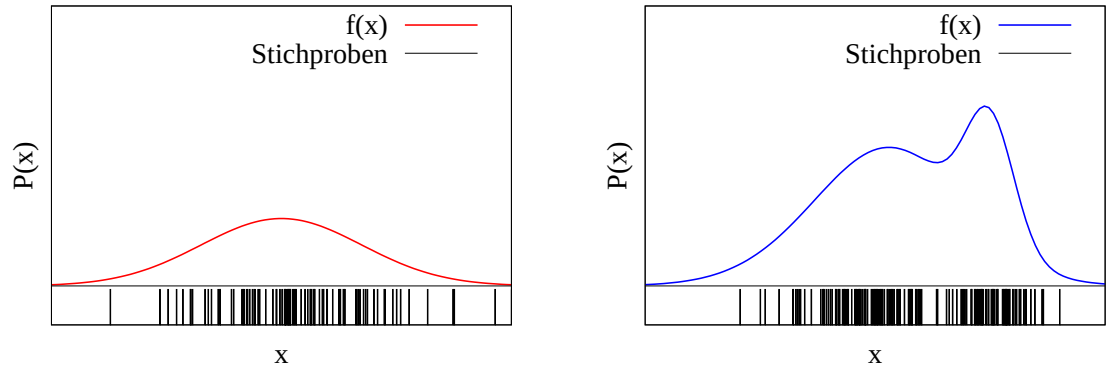


Abbildung 3.4.: Zwei Funktionen und ihre Approximierung durch Stichproben. Die Stichproben werden durch vertikale Striche dargestellt (aus [Stachniss, 2006]).

genommen wurden. Abbildung 3.4 veranschaulicht das Prinzip.

Das in Algorithmus 4 dargestellte Verfahren umfasst drei grundlegende Schritte:

1. **Stichproben ziehen** (Zeile 3):
   Für jedes Partikel wird ein möglicher Nachfolgezustand aus dem Aktionsmodell gezogen. Diese Nachfolgehypothesen bilden eine temporäre neue Partikelmenge $\overline{\mathcal{X}_t}$.

2. **Gewichtung der Stichproben** (Zeile 4):
   Jedem Partikel $x_t^{(i)} \in \overline{\mathcal{X}_t}$ wird ein Gewicht $w_t^{(i)}$ zugeordnet, dass der Wahrscheinlichkeit der aktuellen Messung z_t unter der durch $x_t^{(i)}$ gegebenen Zustandshypothese entspricht.

3. **Konzentration der Stichprobenmenge, *Resampling*** (Zeile 7-10):
   Die neue Partikelmenge $\mathcal{X}_t$ wird aus der temporären Menge $\overline{\mathcal{X}_t}$ erzeugt. Ein Partikel $x_t^{(i)}$ aus $\overline{\mathcal{X}_t}$ wird mit einer zu $w_t^{(i)}$ proportionalen Wahrscheinlichkeit zu $\mathcal{X}_t$ hinzugefügt. Es wird „mit Zurücklegen“ gezogen, $\mathcal{X}_t$ kann das gleiche Partikel also mehrmals beinhalten. Dieser Schritt hat zur Folge, dass Partikel mit einem geringen Gewicht häufig durch solche mit einem hohen Gewicht ersetzt werden. Die Stichprobenmenge konzentriert sich auf diese Weise auf die wahrscheinlichsten Bereiche des Zustandsraums.

Der Resampling-Schritt wird benötigt, da zur Zustandsschätzung nur eine endliche Menge von Stichproben verwendet wird. Ohne den abschließenden Resampling-Schritt würden die meisten Stichproben nach einer Weile Zustandshypothesen mit geringer Wahrscheinlichkeit repräsentieren. Die Bedeutung des Gewichtungs- und Resamplingschritts wird in der folgenden mathematischen Herleitung verdeutlicht (entnommen aus [Stachniss, 2006]).

Gewöhnlich ist das Aktionsmodell nur eine grobe Annäherung der wahren Zustandsübergangsfunktion. Sei zur Verdeutlichung $\pi(x_{0..t} \mid z_{1..t}, u_{1..t})$ die Wahrscheinlichkeitsverteilung,

Algorithmus 4 Partikelfilter

Eingabe:

$\mathcal{X}_{t-1}$	bisherige Partikelmenge
u_t, z_t	aktuelle Daten

Ausgabe:

$\mathcal{X}_t$	neue Partikelmenge

1: $\overline{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
2: **for** $i = 1$ to N **do**
3: ziehe $x_t^{(i)}$ aus $P(x_t \mid u_t, x_{t-1}^{(i)})$
4: $w_t^{(i)} = P(z_t \mid x_t^{(i)})$
5: $\overline{\mathcal{X}}_t = \overline{\mathcal{X}}_t \cup \langle x_t^{(i)}, w_t^{(i)} \rangle$
6: **end for**

7: **for** $i = 1$ to N **do**
8: ziehe m mit Wahrscheinlichkeit proportional zu $w_t^{(m)}$
9: $\mathcal{X}_t = \mathcal{X}_t \cup \langle x_t^{(m)}, \frac{1}{N} \rangle$
10: **end for**

aus der die temporäre Partikelmenge in Schritt 1 gezogen wird. Sei $P(x_{0..t} \mid z_{1..t}, u_{1..t})$ dagegen die zu schätzende wahre Übergangsfunktion.

Die Partikelgewichte dienen allgemein dazu, den Unterschied zwischen diesen beiden Verteilungen auszugleichen (Herleitung siehe [Stachniss, 2006]):

$$w_t = \frac{P(x_t)}{\pi(x_t)} \tag{3.33}$$

Falls die oben erwähnten Markov-Annahmen (3.6) und (3.7) gelten, lässt sich die vollständige Übergangswahrscheinlichkeit $P(x_{0..t} \mid z_{1..t}, u_{1..t})$ in eine rekursive Form überführen:

$$\begin{aligned}
P(x_{0..t} \mid z_{1..t}, u_{1..t}) &\overset{\text{Bayes}}{=} \eta\, P(z_t \mid x_{0..t}, z_{1..t-1}, u_{1..t}) \\
&\quad P(x_{0..t} \mid z_{1..t-1}, u_{1..t}) && (3.34) \\
&\overset{(3.7)}{=} \eta\, P(z_t \mid x_t) \\
&\quad P(x_{0..t} \mid z_{1..t-1}, u_{1..t}) && (3.35) \\
&\overset{\text{Produktregel}}{=} \eta\, P(z_t \mid x_t) \\
&\quad P(x_t \mid x_{0..t-1}, z_{1..t-1}, u_{1..t}) \\
&\quad P(x_{0..t-1} \mid z_{1..t-1}, u_{1..t}) && (3.36) \\
&\overset{(3.6)}{=} \eta\, P(z_t \mid x_t)\, P(x_t \mid x_{t-1}, u_t) \\
&\quad P(x_{0..t-1} \mid z_{1..t-1}, u_{1..t-1}) && (3.37)
\end{aligned}$$

Dabei ist η ein Normalisierungsfaktor, der aus der Anwendung des Satz von Bayes resultiert. Unter der Markov-Annahme gilt außerdem:

$$\pi(x_{0..t} \mid z_{1..t}, u_{1..t}) = \pi(x_t \mid x_{t-1}, z_t, u_t)\, \pi(x_{0..t-1} \mid z_{1..t-1}, u_{1..t-1}) \tag{3.38}$$

Es folgt:

$$w_t = \frac{P(x_{0..t} \mid z_{1..t}, u_{1..t})}{\pi(x_{0..t} \mid z_{1..t}, u_{1..t})} \tag{3.39}$$

$$\overset{(3.37)}{=} \frac{\eta\, P(z_t \mid x_t)\, P(x_t \mid x_{t-1}, u_t)}{\pi(x_{0..t} \mid z_{1..t}, u_{1..t})}\, P(x_{0..t-1} \mid z_{1..t-1}, u_{1..t-1}) \tag{3.40}$$

$$\overset{(3.38)}{=} \frac{\eta\, P(z_t \mid x_t)\, P(x_t \mid x_{t-1}, u_t)}{\pi(x_t \mid x_{t-1}, z_t, u_t)}\, \underbrace{\frac{P(x_{0..t-1} \mid z_{1..t-1}, u_{1..t-1})}{\pi(x_{0..t-1} \mid z_{1..t-1}, u_{1..t-1})}}_{w_{t-1}} \tag{3.41}$$

$$= \frac{\eta\, P(z_t \mid x_t)\, P(x_t \mid x_{t-1}, u_t)}{\pi(x_t \mid x_{t-1}, z_t, u_t)}\, w_{t-1} \tag{3.42}$$

Falls die Stichproben aus dem Aktionsmodell gezogen werden, ergibt sich für jedes Partikel:

$$w_t^{(i)} = \frac{\eta\, P(z_t \mid x_t^{(i)})\, P(x_t \mid x_{t-1}^{(i)}, u_t)}{P(x_t \mid x_{t-1}^{(i)}, u_t)}\, w_{t-1}^{(i)} \tag{3.43}$$

$$= \eta\, P(z_t \mid x_t^{(i)})\, w_{t-1}^{(i)} \tag{3.44}$$

$$\propto P(z_t \mid x_t^{(i)})\, w_{t-1}^{(i)} \tag{3.45}$$

Da der Resampling-Schritt die Partikelgewichte auf $\frac{1}{N}$ zurücksetzt (Zeile 9 in Alg. 4), können die Gewichte des vorigen Zeitschritts ignoriert werden:

$$w_t^{(i)} \propto P(z_t \mid x_t^{(i)}) \tag{3.46}$$

Die Herleitung zeigt, dass zur Berechnung der Gewichte das Sensormodell $P(z_t \mid x_t)$ verwendet werden muss, sofern die Stichproben aus dem Aktionsmodell gezogen werden [Stachniss, 2006].

3.4. Kartentypen

3.4.1. Landmarken-Karten

In der mobilen Robotik werden verschiedene Repräsentationen der Umwelt verwendet. Eine weit verbreitete Methode der Kartierung beruht auf der Erkennung definierter Punkte oder Bereiche in der Umgebung. Diese *Landmarken* sollten mit den zur Verfügung stehenden Sensoren zuverlässig erkennbar sein und in ausreichender Dichte vorkommen. Ihre Po-

sition in der Umwelt wird gewöhnlich als konstant angenommen. Beispiele für solche Landmarken sind Ecken, Kanten und Türrahmen in Gebäuden, Bäume und Laternenmasten im Freien, sowie eine Reihe künstlicher Landmarken wie Strichcodes, Farbmarkierungen, Laser-Reflexionsmarken oder RFID-Tags.

Abbildung 3.5(a) zeigt eine Karte des Intel Research Labs in Seattle[3]. In die Karte wurden Landmarken (Ecken, Türen, Fenster) eingetragen, die nach einfachen linienbasierten Verfahren aus den Abstandsmessungen eines Laserscanners extrahiert wurden.

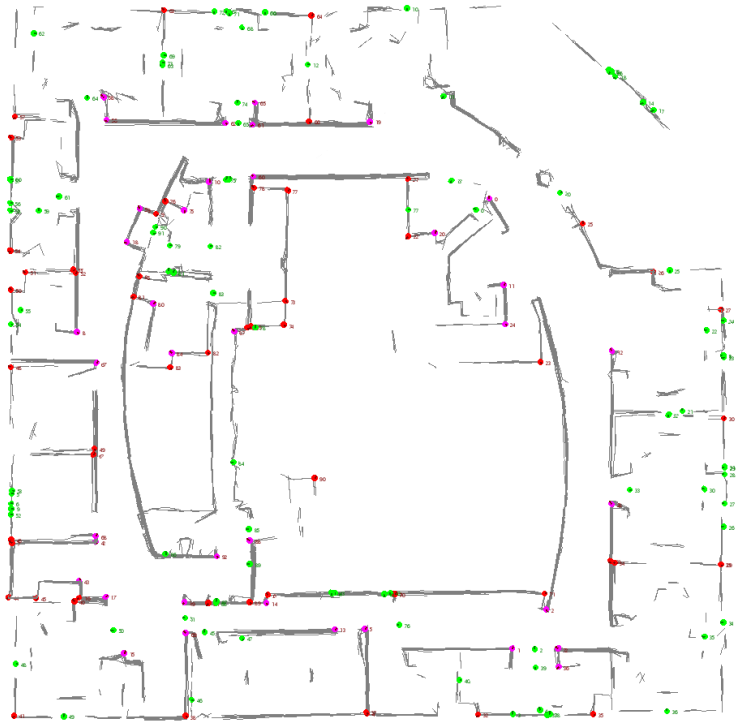
Ein Hauptproblem bei der Verwendung von Landmarken ist die Zuordnung einer gemessenen Landmarke zu einer solchen in der Karte. Dieser Vorgang wird *Datenassoziation* genannt. Sofern die Landmarken nicht von sich aus unterscheidbar sind, muss die Zuordnung aufgrund von Informationen über die Roboterposition oder Umgebung getroffen werden (siehe z.B. [Neira u. Tardós, 2001]). Der Hauptvorteil in der Verwendung von Landmarken ist, dass diese durch ihre kompakte Darstellung effizient gespeichert und verarbeitet werden können.

3.4.2. Gridkarten

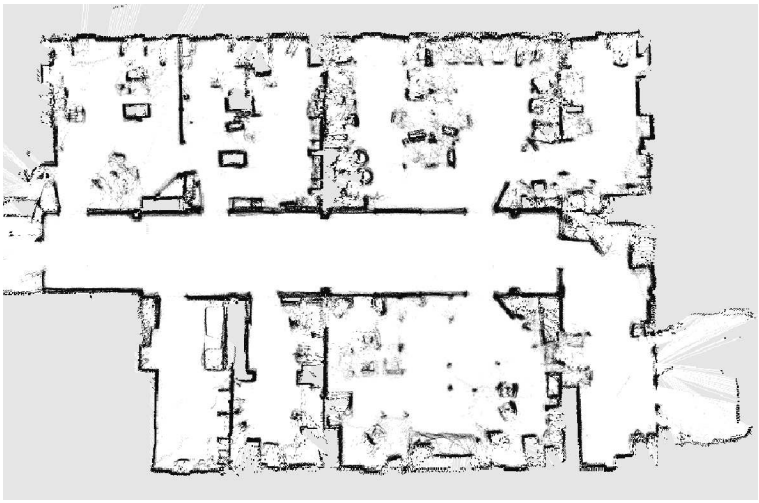
Falls ein Roboter mit Sensoren zur Abstandsmessung wie Laserscannern oder Sonarsensoren ausgestattet ist, können Rasterkarten oder Gridkarten (vom englischen *grid maps*) verwendet werden. Diese Karten unterteilen die Umgebung in gleich große diskrete Bereiche, die als belegt oder frei markiert werden [Moravec u. Elfes, 1985]. Die Kartierung mit Abstandssensoren beruht auf folgender Idee: misst der Abstandssensor ein Hindernis in einer bestimmten Entfernung, kann mit einer gewissen Wahrscheinlichkeit davon ausgegangen werden, dass der Bereich zwischen Hindernis und Sensor frei ist. Zusammen mit der Position und Ausrichtung des Roboters kann die Position des gemessenen Hindernisses und des freien Bereichs ermittelt und in der Gridkarte registriert werden. Durch Integration vieler Messungen an verschiedenen Orten entsteht eine Repräsentation der Umwelt.

Abbildung 3.5(b) zeigt ein Beispiel einer Gridkarte. Es handelt sich um eine zweidimensionale Karte von einem Teil eines Gebäudes der Universität Freiburg und wurde mit Hilfe eines Roboters erstellt, der mit einem Laserscanner ausgestattet ist. Deutlich zu erkennen sind Wände, Schränke, Türen und andere Hindernisse.

[3] Der Datensatz stammt von Dirk Hähnel und kann über das *Robotics Data Set Repository* (Radish) bezogen werden [Howard u. Roy, 2003].



(a) Aus Laserscans erkannte Landmarken im Intel Research Lab, Seattle. Eingezeichnet sind Ecken (rot, magenta), Kanten (grau) und Durchgänge wie Türen und Fenster (grün).



(b) Gridkarte eines Teiles von Geb. 79 der Fakultät für Angewandte Wissenschaften in Freiburg. Weiße Bereiche sind frei, je dunkler eine Bereich ist, desto wahrscheinlicher befindet sich dort ein Hindernis.

Abbildung 3.5.: Beispiele verschiedener Umgebungsrepräsentationen.

Gridkarten sind in der Regel speicher- und rechenintensiver als Landmarken-Karten. Sie bieten allerdings den Vorteil, dass sie ohne vordefinierte Landmarken auskommen und so in einer Vielzahl von Umgebungen eingesetzt werden können. Des Weiteren erlauben sie eine explizite Repräsentation von Regionen, deren Belegtheit noch nicht bekannt ist.

3.5. Kartierung bei bekannten Messpositionen

Landmarken werden typischerweise wie in Beispiel 3.2.2 durch Kalmanfilter repräsentiert, um die Unsicherheit in ihrer Positionsangabe zu berücksichtigen. Gridkarten lassen sich als endliche Menge von Zellen formalisieren, die den kontinuierlichen Zustandsraum approximieren:

$$m = \{m_i\} \tag{3.47}$$

Ziel der Kartierung ist es allgemein, für eine gegebene Menge von Positionen und Positionsangaben zur Zeit der Messung die wahrscheinlichste Karte m^* zu bestimmen:

$$m^* = \underset{m}{\operatorname{argmax}}\, P(m \mid x_{1..t}, z_{1..t}) \tag{3.48}$$

Jeder Zelle wird eine Belegtheitswahrscheinlichkeit zugeordnet. Wenn man annehmen kann, dass die Zellen jeweils unabhängig voneinander sind, lässt sich statt der Wahrscheinlichkeit der gesamten Karte die Wahrscheinlichkeit jeder einzelnen Zelle bestimmen. Unter der Unabhängigkeitsannahme ergibt sich somit die Wahrscheinlichkeit der gesamten Karte als:

$$P(m \mid x_{1..t}, z_{1..t}) = \prod_i P(m_i \mid x_{1..t}, z_{1..t}) \tag{3.49}$$

Die Integration einer Messung z_t und der dazugehörigen Position x_t in eine Gridkarte erfolgt nach dem in Algorithmus 5 angegebenen Verfahren [Moravec u. Elfes, 1985]. Die Belegtheitswahrscheinlichkeiten werden aus Effizienzgründen in der log-odds-Darstellung verwaltet (Definition im Anhang):

$$l_{t,i} \quad = \quad \text{logodds}(P(m_i \mid x_{1..t}, z_{1..t})) = \log\left(\frac{P(m_i \mid x_{1..t}, z_{1..t})}{1 - P(m_i \mid x_{1..t}, z_{1..t})}\right) \tag{3.50}$$

Aus den Werten in log-odds-Darstellung lässt sich $P(m_i \mid x_{1..t}, z_{1..t})$ folgendermassen bestimmen:

$$P(m_i \mid x_{1..t}, z_{1..t}) = 1 - \frac{1}{1 + \exp(l_{t,i})} \tag{3.51}$$

Algorithmus 5 Integration von Messungen in eine Gridkarte

Eingabe:

$m = \{m_i\}$	Gridkarte
$\{l_{t-1,i}\}$	bisherige Belegtheitswahrscheinlichkeiten der Gridzellen
z_t, x_t	aktuelle Messung und zugehörige Position

Ausgabe:

$\{l_{t,i}\}$	Belegtheitswahrscheinlichkeiten

for all Zellen m_i **do**
  if m_i im Sichbereich von z_t **then**
    $l_{t,i} = l_{t-1,i} +$ `InversesSensorModell`$(m_i, x_t, z_t) - l_0$
  else
    $l_{t,i} = l_{t-1,i}$
  end if
end for

Die Konstante l_0 bezeichnet dabei die initiale Belegtheitswahrscheinlichkeit einer Zelle:

$$l_0 = \text{logodds}(P(m_i)) = \log\left(\frac{P(m_i = 1)}{P(m_i = 0)}\right) \tag{3.52}$$

Das in Algorithmus 5 verwendete inverse Sensormodell gibt die Wahrscheinlichkeit dafür an, dass eine Zelle bei einer gegebenen Messung und Position belegt oder frei ist. Eine einfache Implementierung dieses Modells findet sich in Algorithmus 6 [Thrun u. a., 2005]. Der Algorithmus geht von einem Messstrahl mit Öffnungswinkel β aus (siehe Abb. 3.6). Alle Zellen, die im Messbereich liegen und eine Entfernung von maximal $\frac{\gamma}{2}$ zum Endpunkt der Messung haben, werden als belegt angenommen. Der Parameter γ steuert dabei, welcher Bereich um den Endpunkt einer Messung noch als belegt angesehen werden soll. Liegt eine Zelle innerhalb des Messbereichs und beträgt die Entfernung zum Endpunkt der Messung weniger als $\frac{\gamma}{2}$, wird die Zelle als frei angesehen. Über Zellen außerhalb des Öffnungswinkels oder jenseits der maximalen Messdistanz $r_{\max}$ kann keine Aussage getroffen werden.

Algorithmus 6 Inverses Sensormodell für Abstandssensoren

Eingabe:

$m_i = \langle x_i, y_i \rangle$	Zelle der Gridkarte, repräsentiert durch ihren Schwerpunkt
$z_t = \langle r, \alpha \rangle$	Messung
$x_t = \langle x, y, \theta \rangle$	Position des Sensors

Ausgabe:

l_i	Belegtheitswahrscheinlichkeit in log-odds-Darstellung

$r' = \sqrt{(x_i - x)^2 + (y_i - y)^2}$
$\alpha' = \arctan(\frac{y_i - y}{x_i - x}) - \theta$

if $r > \min(r_{\max}, r + \frac{\gamma}{2})$ **or** $|\alpha' - \alpha| > \frac{\beta}{2}$ **then**
  $l_i = l_0$
else
  if $l < r_{\max}$ **and** $|r' - r| < \frac{\gamma}{2}$ **then**
    $l_i = l_{\text{belegt}}$
  else
    $l_i = l_{\text{frei}}$
  end if
end if

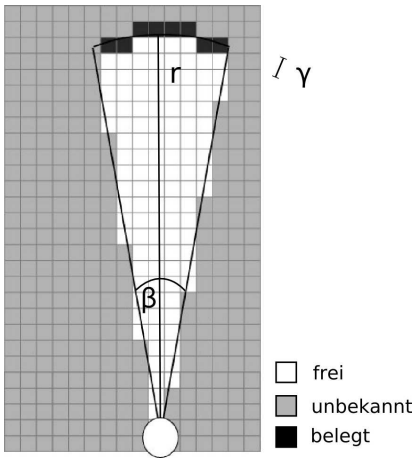


Abbildung 3.6.: Veranschaulichung der Parameter aus Algorithmus 6.

3.6. Lokalisierung bei bekannter Karte

Wie eingangs dargestellt, können autonome Roboter komplexe Aufgaben nur ausführen, wenn ihnen ihre aktuelle Position in der Umgebung bekannt ist. Das Problem, die Position des Roboters bei einer gegebenen Karte der Umgebung zu bestimmen, wird Lokalisierung genannt. Es handelt sich dabei um eine Zustandsschätzung, da die wahre Position des Roboters in der Regel nicht direkt beobachtet werden kann[4]. Aus diesem Grund werden Implementierung des Bayesfilters zur Lokalisierung verwendet. Die Position eines Roboters wird durch seine x- und y-Koordinate bezüglich der Karte, sowie die Ausrichtung des Roboters θ gegenüber dem Koordinatensystem angegeben. Im Fall einer dreidimensionalen Lokalisierung im Raum müssten ausserdem noch Neigungswinkel und Höhe geschätzt werden - wir beschränken uns in dieser Arbeit allerdings auf die zweidimensionale Kartierung und Lokalisierung. Formal besteht das Lokalisierungsproblem in der Bestimmung der Wahrscheinlichkeit eines Zustandes x_t zum Zeitpunkt t bei gegebenen Aktionen $u_{1..t}$, Messungen $z_{1..t}$ sowie einer Karte m:

$$P(x_t \mid z_{1..t}, u_{1..t}, m)$$

Häufig werden Partikelfilter zur Lokalisierung verwendet. Der resultierende Lokalisierungsalgorithmus wird *Monte Carlo Lokalisierung (MCL)* genannt und ist in Algorithmus 7 zu sehen. Es ist eine direkte Anwendung des Partikelfilters. $P(x_t \mid u_t, x_{t-1}^{(i)})$ modelliert die Bewegung des Roboters und wird daher auch *Bewegungsmodell* genannt. $P(z_t \mid x_t^{(i)}, m)$ ist abhängig vom verwendeten Sensor- und Kartentyp. In Abschnitt 4.2.5 werden wir das Sensormodell für Laserscanner bei der Verwendung von Gridkarten angeben.

[4] Satelliten-gestützte Systeme wie GPS können nur im Freien eingesetzt werden und weisen unter Umständen selbst dort hohe Ungenauigkeiten auf.

Algorithmus 7 Monte Carlo Lokalisierung

Eingabe:

$\mathcal{X}_{t-1}$	bisherige Partikelmenge
u_t, z_t	aktuelle Daten
m	Karte der Umgebung

Ausgabe:

$\mathcal{X}_t$	neue Partikelmenge

1: $\overline{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
2: **for** $i = 1$ to N **do**
3: ziehe $x_t^{(i)}$ aus $P(x_t \mid u_t, x_{t-1}^{(i)})$
4: $w_t^{(i)} = P(z_t \mid x_t^{(i)}, m)$
5: $\overline{\mathcal{X}}_t = \overline{\mathcal{X}}_t \cup \langle x_t^{(i)}, w_t^{(i)} \rangle$
6: **end for**

7: **for** $i = 1$ to N **do**
8: ziehe m mit Wahrscheinlichkeit proportional zu $w_t^{(m)}$
9: $\mathcal{X}_t = \mathcal{X}_t \cup \langle x_t^{(m)}, \frac{1}{N} \rangle$
10: **end for**

4. Robuste Kartenerstellung

Im Idealfall kommt ein Kartierungsverfahren ohne oder mit nur einer vagen Beschreibung der zu erwartenden Umwelt aus. So ist sichergestellt, dass das Verfahren ohne Modifikation in einer Reihe verschiedener Szenarien eingesetzt werden kann. Ein Lokalisierungs- und Kartierungsmodul für Überwachungsroboter sollte zum Beispiel in der Lage sein, sowohl im Inneren von Gebäuden, als auch auf weitläufigen Lagerflächen außerhalb von Gebäuden Karten zu erstellen und sich danach darin zu lokalisieren. Im Extremfall der Raumfahrt sind eventuell gar keine Informationen über den Einsatzort verfügbar. Das hier vorgestellte System kombiniert ein robustes Kartierungsverfahren für das Innere von Gebäuden und ein Landmarkenbasierten Verfahren für die Kartierung im Freien. Zwar müssen die erwarteten Landmarken im Voraus definiert werden, wir verwenden jedoch einen recht allgemeinen Landmarkentyp, der verschiedene freistehende Hindernisse wie beispielsweise Baumstämme und Laternenmasten einschließt.

4.1. Verfahren zur simultanen Kartierung und Lokalisierung

Eines der Grundprobleme der Kartierung ist die Tatsache, dass Kartierung und Lokalisierung gegenseitig voneinander abhängig sind. Um akkurate Karten der Umgebung anfertigen zu können, wird eine gute Schätzung der Position des Roboters benötigt. Andererseits ist eine Positionsbestimmung im Allgemeinen nicht ohne Umgebungskarte möglich. Dieses Problem der gleichzeitigen Kartenerstellung und Lokalisierung wird *Simultanious Localization and Mapping (SLAM)* genannt. Formal lässt sich das Problem als die Schätzung einer Karte m und eines Roboterpfades $x_{1..t}$ bei gegebenen Steuerkommandos $u_{1..t}$ und Sensormessungen $z_{1..t}$ beschreiben. Probabilistische Verfahren schätzen dazu folgende Wahrscheinlichkeitsverteilung [Thrun u. a., 2005]:

$$P(x_{1..t}, m \mid z_{1..t}, u_{1..t}) \tag{4.1}$$

In der Vergangenheit wurden verschiedene Verfahren vorgestellt, um diese Verteilung zu bestimmen beziehungsweise anzunähern (siehe [Frese, 2006a] für eine Übersicht). Eines der ältesten Verfahren verwendet einen EKF, um den Roboterpfad und eine Liste von Landmarken zu schätzen [Smith u. a., 1990]. Problematisch bei dieser Methode ist, dass die Schätzung aller Landmarken und der Roboterposition in einer einzigen hochdimensionalen Normalverteilung

verwaltet werden. Da die Komplexität des EKF quadratisch mit der Dimension des Zustands wächst, können auf diese Weise nur relativ wenige (ca. 500) Landmarkenpositionen verwaltet werden [Frese, 2006a].

4.1.1. FastSLAM

Ein häufig verwendetes Verfahren, welches auch für große Umgebungen (50.000 Landmarken) geeignet ist, ist der FastSLAM-Algorithmus von Montemerlo u. a. [2002b]. Er nutzt eine Faktorisierung von (4.1). Angenommen, die Karte setze sich aus einer Menge von Landmarken zusammen:

$$m = \{l_1, ..., l_M\}$$

Falls der Pfad des Roboters $x_{1..t}$ bekannt ist, sind die individuellen Landmarken l_i voneinander unabhängig. Daraus ergibt sich folgende Faktorisierung (*Rao-Blackwellization*) [Doucet u. a., 2000]:

$$\begin{aligned} P(x_{1..t},\ l_{1..M} \mid z_{1..t}, u_{1..t}) &= P(x_{1..t} \mid z_{1..t}, u_{1..t})\ P(l_{1..M} \mid x_{1..t}, z_{1..t}) && (4.2) \\ &= P(x_{1..t} \mid z_{1..t}, u_{1..t}) \prod_{i=1}^{M} P(l_i \mid x_{1..t}, z_{1..t}) && (4.3) \end{aligned}$$

Auf diese Weise lässt sich das SLAM-Problem in mehrere Teilprobleme aufteilen. Die Schätzung von $P(x_{1..t} \mid z_{1..t}, u_{1..t})$ entspricht einem Lokalisierungsproblem. Die Schätzung der Landmarkenpositionen $P(l_i \mid x_{1..t}, z_{1..t})$ kann davon unabhängig erfolgen und entspricht einem Kartierungsproblem bei bekannten Messpositionen.

FastSLAM verwendet einen Partikelfilter, um das Lokalisierungsproblem zu lösen. Die Partikel des Filters enthalten also eine Hypothese über den Roboterpfad. Für jede dieser Hypothesen werden die Landmarken durch individuelle EKFs geschätzt. Abbildung 4.1 illustriert den Aufbau der Partikel. Da Sensor- und Bewegungsmodell nur von der aktuellen Positionsschätzung x_t abhängen, werden die Positionen $x_{1..t-1}$ nicht in den Partikeln gespeichert. So wird sichergestellt, dass der Speicherbedarf nicht von der Länge des Pfades, sondern einzig von der Anzahl der Partikel und der Größe der Karte abhängt [Montemerlo u. a., 2002b]. Die Laufzeit des Algorithmus beträgt ohne weitere Optimierung $O(NM)$, wobei N die Anzahl der Partikel und M die Anzahl der Landmarken bezeichnet. Montemerlo u. a. erreichen durch die Verwendung einer Baumstruktur jedoch schließlich eine Laufzeit von $O(N\ \log(M))$.

Statt einer Menge von Landmarken kann auch eine Gridkarte verwendet werden [Eliazar u. Parr, 2003; Hähnel u. a., 2003]. Für jedes Partikel i wird dann neben einer Positionshypothese $x_t^{(i)}$ eine eigene Gridkarte $m_t^{(i)}$ verwaltet. Die grundlegenden Berechnungschritte des Gridkarten-basierten FastSLAM sind aus Algorithmus 8 ersichtlich.

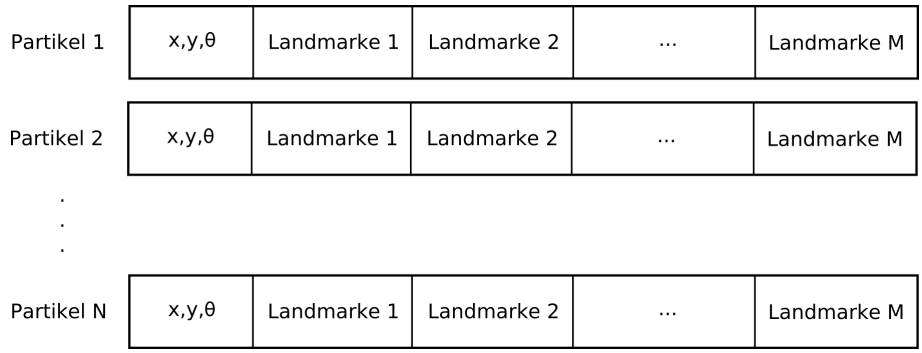


Abbildung 4.1.: Schema der FastSLAM Partikel. Jedes Partikel trägt eine Positionshypothese und M voneinander unabhängige Schätzungen von Landmarkenpositionen, die durch EKFs realisiert werden.

Algorithmus 8 FastSLAM mit Gridkarten

Eingabe:

$\mathcal{X}_{t-1} = \{\langle x_{t-1}^{(i)}, w_{t-1}^{(i)}, m_{t-1}^{(i)}\rangle\}$ bisherige Partikelmenge

u_t, z_t aktuelle Daten

Ausgabe:

$\mathcal{X}_t$ neue Partikelmenge

1: $\overline{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
2: **for** $i = 1$ to N **do**
3: ziehe $x_t^{(i)}$ aus $P(x_t \mid u_t, x_{t-1}^{(i)})$
4: $w_t^{(i)} = P(z_t \mid x_t^{(i)}, m_{t-1}^{(i)})$
5: $m_t^{(i)} =$ `AktualisiereGridkarte` $(z_t, x_t^{(i)}, m_{t-1}^{(i)})$
6: $\overline{\mathcal{X}}_t = \overline{\mathcal{X}}_t \cup \langle x_t^{(i)}, w_t^{(i)}, m_t^{(i)}\rangle$
7: **end for**

8: **for** $i = 1$ to N **do**
9: ziehe k mit Wahrscheinlichkeit proportional zu $w_t^{(k)}$
10: $\mathcal{X}_t = \mathcal{X}_t \cup \langle x_t^{(k)}, \frac{1}{N}, m_t^{(k)}\rangle$
11: **end for**

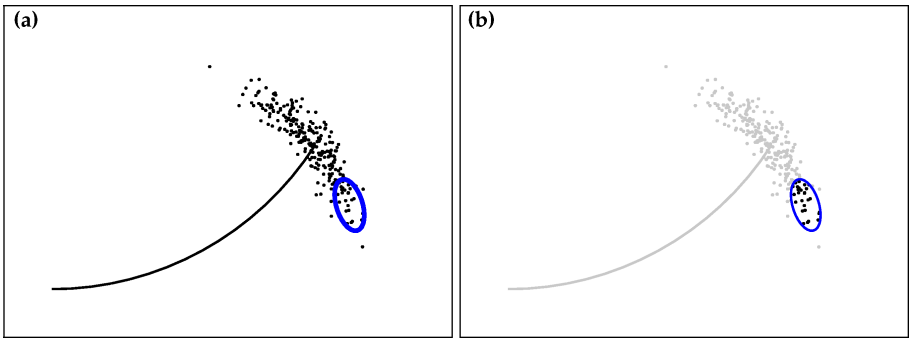


Abbildung 4.2.: Vergleich zwischen Vorhersage basierend auf der Odometrie (a) und der Schätzung nach dem Resampling-Schritt (b) (aus Thrun u. a. [2005]).

Die Funktion `AktualisiereGridkarte` kann wie in Abschnitt *Kartierung bei bekannten Messungen* (3.5) implementiert werden.

4.1.2. Verbesserungen des FastSLAM

Ein Hauptproblem des Partikelfilters ist, dass die Komplexität exponentiell mit der Anzahl der Partikel wächst [Thrun u. a., 2005]. In den letzten Jahren wurde daher eine Reihe von Verbesserungen des FastSLAM-Algorithmus entwickelt, die eine Reduzierung der Partikelmenge ermöglichen.

Verbesserte Vorhersage

Eine Möglichkeit, eine solche Verbesserung des Algorithmus zu erreichen, ergibt sich aus der Erkenntnis, dass Sensormessungen oftmals sehr viel präziser sind als die Odometrie eines Roboters. Dies ist beispielsweise bei Robotersystemen mit Laserscannern der Fall. Hier kann es vorkommen, dass während des Vorhersageschritts (Zeile 3 in Alg. 8) viele Hypothesen generiert werden, die während des Resampling-Schrittes (Zeile 8-11) wieder eliminiert werden müssen. Abbildung 4.2 veranschaulicht dieses Problem.

Verschiedene Verfahren verwenden daher eine verbesserte Vorhersage, die die aktuelle Messung schon beim Vorhersageschritt berücksichtigen [van der Merwe u. a., 2001; Montemerlo u. a., 2003; Hähnel u. a., 2003; Grisetti u. a., 2007a; Grzonka u. a., 2007]. Auf diese Weise werden weniger schlechte Hypothesen generiert. Folglich kann eine deutlich kleinere Partikelmenge verwendet werden.

In dieser Arbeit wird die verbesserte Vorhersage aus [Grisetti u. a., 2007a] verwendet. Für jedes Partikel wird hier zunächst die wahrscheinlichste Roboterposition basierend auf Odometrie-Informationen und der aktuellen Lasermessung bestimmt. Die Lasermessung wird dazu mit

einem *Scanmatching*-Verfahren in die bisherige Karte des Partikels eingepasst (siehe z.B. [Stachniss, 2006]). Die Vorhersageverteilung wird dann berechnet, in dem in der Nähe dieser Position die Messwahrscheinlichkeit ausgewertet und daraus eine Normalverteilung geschätzt wird.

Adaptives Resampling

Ein weiteres Problem, welches häufig bei der Verwendung von Partikelfiltern auftritt, ist die Degenerierung des Filters. Hierunter versteht man die zufällige Eliminierung guter Hypothesen (Partikel) aus dem Filter beim Resampling, so dass schließlich keine Partikel mehr in der Nähe des wahren Zustands liegen. Je weniger Partikel verwendet werden, desto wahrscheinlicher tritt dieses Problem auf. Andererseits kann die Partikelmenge reduziert werden, wenn die Wahrscheinlichkeit einer Degenerierung des Filters auf eine andere Weise verringert wird.

Eine Möglichkeit dies zu tun, ist ein verzögerter Resampling-Schritt. Statt das Resampling in jedem Zeitschritt auszuführen, kann es sinnvoll sein, diesen Schritt auszulassen und so die Wahrscheinlichkeit einer Degenerierung des Filters zu verringern. Intuitiv ist ein Resamplingschritt nicht nötig, falls im Bewertungsschritt des Partikelfilters viele Partikel das gleiche Gewicht erhalten. Doucet [1998] schlägt eine Strategie basierend auf der so genannten effektiven Partikelanzahl N_{eff} vor, die sich an der Varianz der Partikelgewichte orientiert:

$$N_{\mathit{eff}} = \frac{1}{\sum_{i=1}^{N} (w^{(i)})^2} \tag{4.4}$$

Das Resampling wird nur dann ausgeführt, falls N_{eff} unter einen Grenzwert fällt (z.B. $N/2$). Anderenfalls wird die neue Gewichtung eines Partikels mit dessen bisheriger Gewichtung multipliziert (siehe Gleichung (3.42):

$$w_t^{(i)} = \frac{\eta \; P(z_t \mid x_t) P(x_t \mid x_{t-1}, u_{t-1})}{\pi(x_t \mid x_{t-1}, z_t, u_{t-1})} \; w_{t-1}, \tag{4.5}$$

wobei π die verbesserte Vorhersage bezeichnet.

Das Risiko fälschlicherweise gute Partikel zu entfernen wird mit dieser Methode stark reduziert. Des Weiteren reduziert sich durch das so genannte *Adaptive Resampling* die Laufzeit des Algorithmus [Grisetti u. a., 2007a].

4.2. Kombination von Gridkarten und Landmarken

Das in dieser Arbeit vorgestellte System basiert auf dem FastSLAM Algorithmus. Es verwendet einen Partikelfilter, um den Pfad des Roboters und eine Repräsentation der Umwelt zu schätzen. In klassischen Ansätzen werden entweder Landmarken oder Gridkarten verwenden, um die Roboterposition zu korrigieren. Unser System dagegen verwaltet für jedes Partikel sowohl eine Gridkarte als auch eine Menge von Landmarken und entscheidet in jedem Schritt, welche Repräsentation sich aktuell zur Positionskorrektur am besten eignet. Auf diese Weise können die Vorzüge beider Methoden kombiniert werden.

4.2.1. Verwendete Landmarken

Grundsätzlich ist das hier untersuchte Verfahren unabhängig von der Art der verwendeten Landmarken. In unseren Experimenten sollten Landmarken verwendet werden, die mit einem Laserscanner erkannt werden können, insbesondere auch dann, wenn es bei der Verwendung eines Grid-basierten Verfahrens zu Fehlern kommen könnte. Dies ist beispielsweise in weitläufigen Arealen mit wenigen Objekten der Fall. Hier liegen in der Regel nur wenige verwertbare Abstandsmessungen vor, die dann eventuell nicht eindeutig in eine lokale Karte eingepasst werden können. In einer solchen Situation kann es aber dennoch möglich sein, Landmarken aus Laserscans zu extrahieren, beispielsweise freistehende Bäume oder Laternenmasten.

In unserer derzeitigen Implementierung definieren wir daher Landmarken als freistehende Objekte mit einem maximalen Radius r_{max}. Ein Objekt gilt als freistehend, wenn die durch d_{min} definierte Umgebung des Objekts frei von weiteren Objekten ist. Abbildung 4.3 veranschaulicht diese Definition.

4.2.2. Landmarkenextraktion

Aus den Entfernungsmessungen r_i des Laserscanners und deren Winkeln α_i werden zunächst Endpunkte $p_i = \langle x_i, y_i \rangle$ in kartesischen Koordinaten errechnet:

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} \sin(\alpha_i)\ r_i \\ \cos(\alpha_i)\ r_i \end{pmatrix} \tag{4.6}$$

Diese Punktmenge wird in zusammenhängende Bereiche aufgeteilt, um die räumliche Struktur besser erfassen zu können. Der Laserscanner führt Entfernungsmessungen in festen Winkelabständen von zum Bespiel 1° durch. Aufeinanderfolgende Messungen werden dem gleichen Segment zugeschrieben, falls ihr euklidische Abstand zueinander einen Grenzwert c_s nicht überschreitet. Algorithmus 9 fasst den Segmentierungsschritt zusammen, Abbildung 4.4 veranschaulicht das Ergebnis der Segmentierung.

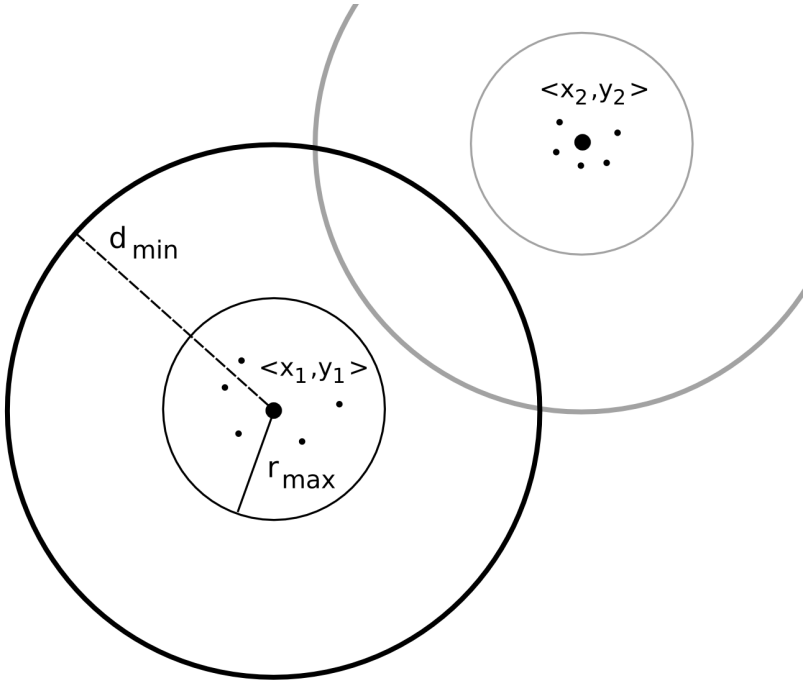


Abbildung 4.3.: Landmarken werden aus Gruppen von Laser-Endpunkten extrahiert. Sie werden definiert durch deren Schwerpunkt $\langle x_i, y_i \rangle$. Die Endpunkte einer Landmarke müssen innerhalb der Fläche liegen, die durch r_{max} definiert wird. Die umliegende Fläche, definiert durch d_{min}, darf keine weiteren Endpunkte beinhalten.

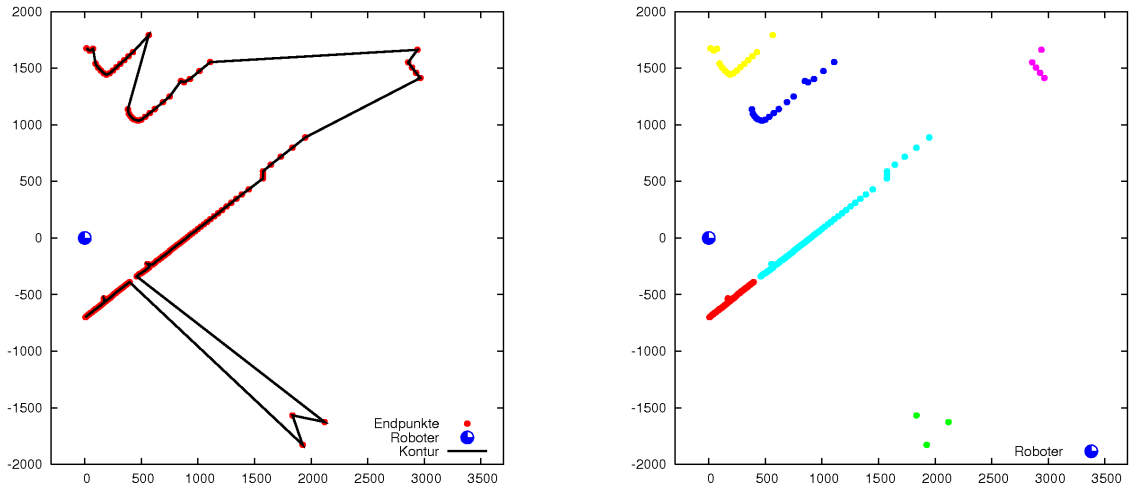


Abbildung 4.4.: Links: Beispielhafter Laserscan. Aufeinander folgende Scan-Endpunkte (rot) wurden zur Visualisierung miteinander verbunden. Der Roboter (blau) steht im Beispiel seitlich zu einer Hauswand mit Blick auf einen Parkplatz. Rechts: Ergebnis der Segmentierung.

Algorithmus 9 Segmentierung eines Laserscans

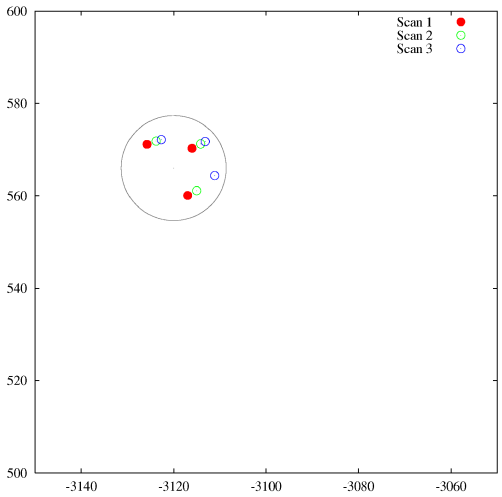
Eingabe:
$\mathcal{P} = \{p_{1..n}\}$, Scan-Endpunkte
Ausgabe:
$\mathcal{S}$, Segmente

$s = \{p_0\}$

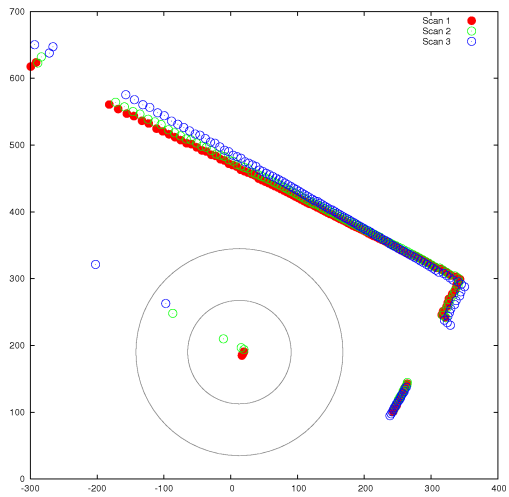
for $i = 1$ to n **do**
  if $||p_i - p_{i-1}|| < c_s$ **then**
    $s = s \cup \{p_i\}$
  else
    $\mathcal{S} = \mathcal{S} \cup s$
    $s = \{p_i\}$
  end if
end for
$\mathcal{S} = \mathcal{S} \cup s$

Die Segmentierung dient als Grundlage der Extraktion von Landmarken. Aus der Menge der Segmente werden zunächst all diejenigen entfernt, die aufgrund ihrer Größe nicht als Landmarke in Frage kommen. Segmente mit einer Länge von mehr als $2 \cdot r_{max}$ werden schon an dieser Stelle als Landmarke ausgeschlossen, um unnötige Berechnungen zu vermeiden. Für die verbliebenen Segmente wird nun geprüft, ob sie den oben genannten Kriterien für Landmarken genügen. Finden sich keine Scanpunkte mit einem Abstand von mehr als r_{max} aber weniger als d_{min} zum Schwerpunkt eines Segments (vgl. Abb. 4.3), gilt es als freistehend und der entsprechende Schwerpunkt wird als gesehene Landmarke gespeichert.

Es kommt vor, dass fälschlicherweise Segmente und Landmarken erkannt werden, weil Laserstrahlen reflektiert wurden oder Teile der Umgebung aufgrund von Verdeckungen nicht einsehbar waren. In unserer momentanen Implementierung werden die Messdaten vom Roboter zunächst nur aufgenommen und im Nachhinein verarbeitet. Auf diese Weise stehen zu jedem Zeitpunkt der Verarbeitung auch zukünftige Messungen zur Verfügung. Diese können genutzt werden, um solche falschen Landmarken auszuschließen. Dazu wird anhand der aktuellen Positionsschätzung und den entsprechenden Odometrie-Messungen für die nächsten n Messungen eine geschätzte zukünftige Position errechnet. Fallen Scanpunkte dieser Messungen in den freien Raum um eine Landmarke, wird diese verworfen. Diese vorausschauende Landmarkenextraktion wird in Abbildung 4.5 durch zwei Beispiele illustriert. Hier wird eine Landmarke durch die nachfolgenden Messungen bestätigt (Abb. 4.5 (a)), eine andere wird verworfen, da die nachfolgenden Messungen in den freien Raum um die Landmarke fallen (Abb. 4.5 (b)). Da die Schätzung der zukünftigen Positionen auf den nicht korrigierten Odometrie-Messungen beruhen, sollte n nicht zu groß gewählt werden. In unseren Experimenten war es



(a) Beispiel einer erkannten und durch folgende Messungen bestätigten Landmarke.



(b) Beispiel einer anhand von folgenden Messungen verworfenen Landmarke.

Abbildung 4.5.: Eliminierung falscher Landmarken anhand von nachfolgenden Messungen.

ausreichend, die nächsten zwei Messungen zu berücksichtigen.

Eine ähnliche Verbesserung des Landmarken-Extraktors ist prinzipiell ebenfalls durchführbar, falls die Daten des Roboters nicht im Vorhinein aufgezeichnet werden. Grzonka u. a. [2007] verwenden eine verzögerte Integration von Sensordaten, um anhand der zukünftigen Daten eine bessere Positionsschätzung zu erreichen. Mit einer solchen verzögerten Integration wäre auch der Einsatz des Systems auf dem fahrenden Roboter denkbar. Bei bis zu 75 Laser- und Odometriemessungen pro Sekunde ist durch die Verzögerung kein Verlust an Reaktionsfähigkeit zu erwarten.

4.2.3. Verwaltung der Landmarken

In dem in dieser Arbeit vorgestellten Verfahren werden Landmarken durch Normalverteilungen repräsentiert, um die Unsicherheit in deren geschätzter Position abzubilden. Wie in Beispiel 3.2.2 wird die Position jeder Landmarke durch einen eigenen Kalmanfilter geschätzt.

Die Datenassoziation erfolgt über die Distanz einer gemessenen Landmarke zur nächsten Landmarke, die bereits in der Karte verzeichnet ist. Die einfache euklidische Distanz erfasst dabei die Unsicherheit in der Schätzung der Landmarken nicht. Darum verwenden wir die Mahalanobis Distanz

$$d(x, y) = \sqrt{(x - y)^T S^{-1} (x - y)}, \tag{4.7}$$

wobei x und y mehrdimensionale Datenpunkte sind und S die entsprechende Kovarianzmatrix

bezeichnet. Eine gemessene Landmarke l wird als neue Landmarke in die Karte aufgenommen, falls ihre Distanz zu allen in der Karte vorhandenen Landmarken l_i größer als eine Konstante ist

$$d(l, l_i) > c_m. \tag{4.8}$$

Bei der Wahl der Konstanten c_m sollte die erwartete Dichte der Landmarken in der Umgebung beachtet werden. Anderenfalls kann es sein, dass verschiedene Landmarken fälschlicherweise zusammengefasst werden.

Neue Landmarken werden in unserer Implementierung nicht direkt in die Karte aufgenommen. Stattdessen werden alle neuen Landmarken zunächst in eine vorläufige Landmarkenliste eingetragen. Erst nachdem eine Landmarke erneut gemessen wurden, wird sie in die Karte aufgenommen und zur Positionskorrektur verwendet. Wird eine neue Landmarke in einer vorgegebenen Anzahl von Zeitschritten nicht erneut gemessen, wird sie aus der vorläufigen Liste gestrichen. Auf diese Weise wird verhindert, dass einmalige Fehlmessungen als Landmarke in die Karte integriert werden.

Die Repräsentation durch Normalverteilungen erlaubt eine einfache Implementierung des Sensormodells für den Landmarkensensor. Sei z eine Messung, die an Position x aufgenommen wurde. Für eine gegebene Landmarkenschätzung $l = \langle \mu, \Sigma \rangle$ ergibt sich folgende Messwahrscheinlichkeit:

$$P(z \mid x) = \frac{1}{\sqrt{(2\pi)^2 \; det(\Sigma)}} \; e^{-\frac{1}{2}(z - h(\mu))^T \Sigma^{-1} (z - h(\mu))} \tag{4.9}$$

4.2.4. Algorithmus

Das hier untersuchte Verfahren kombiniert den gridbasierten FastSLAM Algorithmus mit Techniken des Landmarken-basierten FastSLAM. Es handelt sich um einen Rao-Blackwellized Partikelfilter, der die oben genannte Faktorisierung (4.3) nutzt. Algorithmus 10 gibt die entscheidenden Arbeitsschritte und Datenstrukturen an. Es werden sowohl eine Gridkarte, als auch eine Menge von Landmarken verwaltet. Die im Zeitschritt t aktuelle Gridkarte wird hier mit $m_{g,t}$ bezeichnet, $m_{f,t}$ bezeichnet die Landmarken-Karte. Zu Beginn jedes Durchlaufs wird entschieden, welcher Kartentyp für die nächste Iteration des Partikelfilter-Algorithmus verwendet werden soll. Grundlage dieser Entscheidung ist die Partikelmenge des vorherigen Zeitschritts, sowie die aktuellen Odometrie- und Sensormessungen.
Im Partikelfilter-Algorithmus werden im ersten Schritt Stichproben der neuen Position gezogen. Falls zu Beginn die Gridkarte ausgewählt wurde, wird in unserem Ansatz eine verbesserte Vorhersage der Position verwendet, wie sie in Abschnitt 4.1.2 vorgestellt wurde. Wird dagegen die Landmarkenkarte verwendet, werden die Stichproben aus dem Bewegungsmodell

Algorithmus 10 Kombinierter SLAM-Ansatz

Eingabe:

$\mathcal{X}_{t-1}$,	Partikelmenge des vorherigen Zeitschritts
$z_{l,t}$,	aktuelle Lasermessung
$z_{f,t}$,	aktuelle Landmarken-Messungen
u_{t-1},	aktuelle Odometrie-Messungen

Ausgabe:

$\mathcal{X}_t$,	neue Partikelmenge

Kartentyp = Modellauswahl($\mathcal{S}_{t-1}, z_{l,t}, z_{f,t}, u_{t-1}$)

$\mathcal{S}_t = \{\}$
for all $s_{t-1}^{(i)} \in \mathcal{S}_{t-1}$ **do**
  $< x_{t-1}^{(i)}, w_{g,t-1}^{(i)}, w_{f,t-1}^{(i)} m_{g,t-1}^{(i)}, m_{f,t-1}^{(i)} \rangle = s_{t-1}^{(i)}$

  // Stichproben ziehen
  if (**Kartentyp** = **Gridkarte**) **then**
    $x_t^{(i)} \sim P(x_t \mid x_{t-1}^{(i)}, u_{t-1}, z_{l,t}, m_{g,t-1}^{(i)})$
  else
    $x_t^{(i)} \sim P(x_t \mid x_{t-1}^{(i)}, u_{t-1})$
  end if

  // Gewichtung der Partikel
  $w_{g,t}^{(i)}$ = AktualisiereGridGewicht($w_{g,t-1}^{(i)}, m_{g,t-1}^{(i)}, z_{l,t}$)
  $w_{f,t}^{(i)}$ = AktualisiereLandkartenGewicht($w_{f,t-1}^{(i)}, m_{f,t-1}^{(i)}, z_{f,t}$)

  // Aktualisierung der Karten
  $m_{g,t}^{(i)}$ = IntegriereScan($m_{g,t-1}^{(i)}, x_t^{(i)}, z_{l,t}$)
  $m_{f,t}^{(i)}$ = IntegriereLandmarken($m_{f,t-1}^{(i)}, x_t^{(i)}, z_{f,t}$)
  $\mathcal{S}_t = \mathcal{S}_t \cup \{\langle x_t^{(i)}, w_{g,t}^{(i)}, w_{f,t}^{(i)}, m_{g,t}^{(i)}, m_{f,t}^{(i)} \rangle\}$
end for

for $i = 0$ to N **do**
  if (**Kartentyp** = **Gridkarte**) **then**
    $w^{(i)} = w_g^{(i)}$
  else
    $w^{(i)} = w_f^{(i)}$
  end if
end for

$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} \left(w^{(i)}\right)^2}$

// Resampling
if $N_{\text{eff}} < T$ **then**
  $\mathcal{S}_t$ = resample($\mathcal{S}_t, \{w^{(i)}\}$)
end if

gezogen. Es folgt die Gewichtung der Stichproben. Für jedes Partikel werden zwei Gewichte berechnet: $w_g^{(i)}$ speichert das Gewicht bezüglich der Gridkarte, $w_f^{(i)}$ das Gewicht bezüglich der Landmarken-Karte. Nach der Aktualisierung beider Karten anhand der aktuellen Messungen wird das Resampling durchgeführt. Wir verwenden die adaptive Resampling-Strategie, die in Abschnitt 4.1.2 beschrieben wurde. Dabei wird der N_{eff}-Wert je nach gewähltem Kartentyp basierend auf der Menge $\{w_g^{(i)}\}$ oder $\{w_f^{(i)}\}$ berechnet.

4.2.5. Modellauswahl

Einer der zentralen Aspekte des hier untersuchten Verfahrens ist die Auswahl der aktuell zu verwendenden Umgebungsrepräsentation. Um diese Entscheidung zu treffen, können die aktuellen Sensor- und Odometriemessungen sowie der Zustand des Filters und der Karten verwendet werden. Im Folgenden werden verschiedene Auswahlkriterien und daraus abgeleitete Entscheidungsheuristiken vorgestellt.

Aktuelle Messungen

Wie bereits erwähnt, treten bei der Verwendung von Gridkarten vor allem dann Fehler auf, wenn die aktuelle Lasermessung nicht in die bisherige Karte eingepasst werden kann. Sofern aktuell Landmarken erkannt werden, könnte in diesen Fällen eine Landkarten-basierte Korrektur der Positionsschätzung besser geeignet sein.

Das Sensormodell für Lasermessungen liefert ein Maß, welches verwendet werden kann, um solche Situationen zu erkennen. Es gibt die Wahrscheinlichkeit $P(z_t \mid x_t, m_{g,t})$ einer Lasermessung z_t bei gegebener Gridkarte $m_{g,t}$ und der aktuellen Positionsschätzung x_t an. Um diesen Wert zu berechnen, verwenden wir das Laserendpunktmodell [Thrun u. a., 2005]. Dieses Modell geht davon aus, dass die individuellen Strahlen einer Lasermessung voneinander unabhängige Messungen ergeben. Die Wahrscheinlichkeit jeder einzelnen Teilmessung ergibt sich dann aus der Distanz des Endpunktes des entsprechenden Laserstrahls zur nächsten belegten Zelle in der Gridkarte. Algorithmus 11 fasst die Berechnungsschritte zusammen. Messungen mit maximalem Entfernungswert $r_{\max}$ werden ignoriert, da diese in der Regel nicht durch Hindernisse verursacht werden sondern einen Fehlerwert darstellen. Der nächste Schritt wandelt die Winkel und Entfernungsmessungen $\langle \alpha_t^i, r_t^i \rangle$ in Koordinaten $\langle x_{z_t^i}, y_{z_t^i} \rangle$ der Gridkarte um. Für diese Endpunkte wird jeweils der Abstand *dist* zur nächsten belegten Zelle in der Gridkarte berechnet.
In Zeile 7 wird daraus die Wahrscheinlichkeit jedes einzelnen Laserstrahls berechnet. Es handelt sich um eine Mischverteilung, die drei Quellen von Messfehlern modelliert: Messrauschen, zufällige Messungen z.B. durch Reflektionen und maximale Messungen, die einen Fehlerwert des Sensors darstellen. Es wird angenommen, dass der Sensor ein normalverteiltes Rauschen mit Varianz σ_{mess}^2 aufweist. Die Funktion prob$(dist, \sigma_{mess}^2)$ berechnet aus den Abständen zur

Algorithmus 11 Messwahrscheinlichkeit im Endpunktmodell

Eingabe:

$z_t = \{z_t^1, ..., z_t^k\} = \{\langle \alpha_t^1, r_t^1 \rangle, ..., \langle \alpha_t^k, r_t^k \rangle\}$	Lasermessung
$x_t = \langle x, y, \theta \rangle$	Positionsschätzung
$m_{g,t}$	Gridkarte

Ausgabe:

$P(z_t \mid x_t, m_{g,t})$,	Messwahrscheinlichkeit

1: $p = 1$

2: **for** $i = 1$ to k **do**
3: **if** $r_t^i \neq r_{\max}$ **then**
4: $x_{z_t^i} = x + \sin(\alpha_t^i + \theta)\ r_t^i$
5: $y_{z_t^i} = y + \cos(\alpha_t^i + \theta)\ r_t^i$
6: $dist = \min_{x',y'} \left\{ \sqrt{(x_{z_t^i} - x')^2 + (y_{z_t^i} - y')^2} \ \mid \langle x', y' \rangle \text{ belegt in } m_{g,t} \right\}$
7: $p = p \cdot \left(c_m \cdot \text{prob}(dist, \sigma^2_{mess}) + c_z \cdot P_{zufall}(z_t^i \mid x_t, m_{g,t}) + c_f \cdot P_{fehler}(z_t^i \mid x_t, m_{g,t}) \right)$
8: **end if**
9: **end for**

10: **return** p

nächsten belegten Zelle eine entsprechende Wahrscheinlichkeit. Zufällige Messungen werden durch eine uniforme Wahrscheinlichkeitsverteilung $P_{zufall}(z_t^i \mid x_t, m_{g,t})$ modelliert. Eine punktuelle Wahrscheinlichkeit $P_{fehler}(z_t^i \mid x_t, m_{g,t})$ modelliert Sensorfehler. Über Konstanten c_m, c_f und c_z, mit $c_m + c_f + c_z = 1$, werden die Einzelverteilungen gemischt. Unter der Unabhängigkeitsannahme des Modells ergibt sich die Gesamtwahrscheinlichkeit einer Messung durch Multiplikation der Wahrscheinlichkeiten der einzelnen Teilmessungen [Thrun u. a., 2005].

Aus der so berechneten Wahrscheinlichkeit einer Messung lässt sich eine Heuristik zur Modellauswahl ableiten. Um eine Aussage für den gesamten Filter treffen zu können, wird die gewichtete durchschnittliche Messwahrscheinlichkeit über alle Partikel gebildet:

$$\bar{l} \quad = \quad \frac{1}{N} \sum_{i=1}^{N} w_{t-1}^{(i)}\ P(z_t \mid x_t^{(i)}, m_{g,t}^{(i)}) \tag{4.10}$$

Eine nach diesem Maß wahrscheinliche Lasermessung legt die Verwendung der gridbasierten Umweltrepräsentation nahe, während eine wenig wahrscheinliche Messung für die Verwendung der Landmarken-Karte spricht. Eine erste Entscheidungsheuristik ergibt sich demnach aus der Verwendung eines Grenzwertes $\bar{l} \leq c_l$ (z.B. mit $c_l = 0.2$).
Im Folgenden wird eine weitere Heuristik vorgestellt, die den aktuellen Zustand des Partikelfilters verwendet. Beide Heuristiken werden dann durch ein Lernverfahren miteinander

kombiniert.

Varianz in Partikelgewichten

Das hier vorgestellte Verfahren verwaltet sowohl Gridkarten als auch Landmarken-Karten. Darum trägt jedes Partikel zwei Partikelgewichte: $w_g^{(i)}$ für die Gridkarte und $w_f^{(i)}$ für die Landmarken-Karte. Die Gewichte können als heuristisches Maß dafür angesehen werden, wie gut die aktuellen Daten durch die entsprechende Repräsentation erklärt werden. Da sich die Gewichte auf Messungen von verschiedenen Sensoren beziehen, können sie nicht direkt miteinander verglichen werden. Vergleichbar ist dagegen, wie die Partikelgewichte innerhalb des Filters verteilt sind. Als Maß für die Streuung der Gewichte bietet sich die Varianz an. Intuitiv deutet eine Menge von Gewichten mit einer geringen Varianz darauf hin, dass der Filter keine der durch die Partikel vertretenen Hypothesen besonders bevorzugt. Im Gegensatz dazu ist eine hohe Varianz innerhalb der Partikelgewichte ein Anzeichen dafür, dass einige Hypothesen deutlich wahrscheinlicher sind als andere.

Diese Eigenschaft der Partikelgewichte kann verwendet werden, um eine Modellauswahl zu treffen. Statt der Varianz verwenden wir die oben erwähnte effektive Partikelanzahl N_{eff}. Eine höhere Varianz führt zu einem geringeren N_{eff}-Wert, wie aus der Definition von N_{eff} und der Schätzung der Varianz hervorgeht:

$$N_{\mathit{eff}} = \frac{1}{\sum_{i=1}^{N}(w^{(i)})^2} \tag{4.11}$$

$$Var(w) \approx \sum_{i=1}^{N}(w^{(i)} - \mu_w)^2 \tag{4.12}$$

N_{eff} wird für beide Mengen von Partikelgewichten berechnet:

$$N_{\mathit{eff}}^{g} = \frac{1}{\sum_{i=1}^{N}(w_g^{(i)})^2} \tag{4.13}$$

$$N_{\mathit{eff}}^{f} = \frac{1}{\sum_{i=1}^{N}(w_f^{(i)})^2} \tag{4.14}$$

Unter der Annahme, dass eine höhere Varianz in den Partikelgewichten ein Indiz für eine höhere Diskriminativität des entsprechenden Modells ist, erlauben diese Werte eine heuristische Wahl des Umgebungsmodells. Falls $N_{\mathit{eff}}^{f} < N_{\mathit{eff}}^{g}$ wird dann die Landmarken-Karte verwendet, sonst die Gridkarte.

Die vorgestellten Heuristiken stützen ihre Entscheidung entweder auf Eigenschaften der aktuellen Messung oder auf Eigenschaften der Partikel. Im Folgenden wird ein Verfahren vorgestellt, dass eine Modellauswahl basierend auf beiden Informationen erlernen kann.

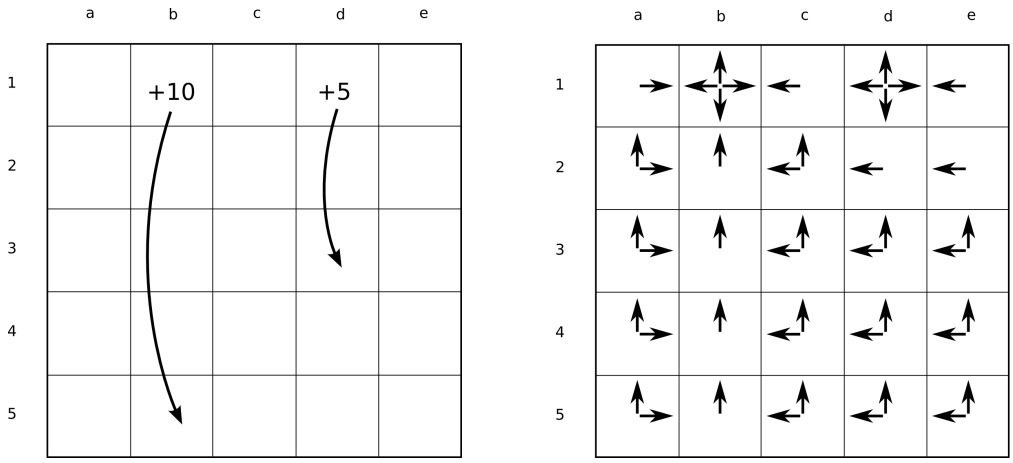


Abbildung 4.6.: Die linke Abbildung zeigt eine einfache Beispielumgebung [Sutton u. Barto, 1998]. Mögliche Aktionen: links, rechts, oben, unten. Auf Feld `b1` erhält der Agent die Belohnung +10, auf Feld `d1` die Belohnung +5. Auf allen anderen Feldern erhält er die Belohnung 0. Unabhängig von der gewählten Aktion wird der Agent von `b1` nach `b5` und von `d1` nach `d3` transportiert. Rechts: optimale Strategie bei zufälliger Startposition um eine maximale Belohnung zu erhalten.

4.2.6. Lernen der Modellauswahl

Idealerweise würde die Modellauswahl sowohl Informationen über die aktuellen Messungen als auch über den Zustand des Filters einbeziehen. Zu diesem Zweck verwenden wir ein Lernverfahren, das dem menschlichen Lernen durch Versuch und Irrtum nachempfunden wurde. Die Grundidee von *Lernen durch Verstärkung* oder englisch *Reinforcement Learning (RL)* ist es, durch Interaktion mit der Umwelt die in einer Situation jeweils beste Aktion zu erlernen. Dazu wird eine Abbildung von Zuständen zu Aktionen bestimmt, die eine numerische Belohnung maximiert [Sutton u. Barto, 1998]. Eine solche Abbildung wird im folgenden *Strategie* (engl. *policy*) genannt. RL-Algorithmen führen eine Reihe von Aktionen durch, analysieren diese anhand einer erhaltenen Belohnung und bewerten dadurch die Aktionen. Nach und nach wird so eine Strategie aufgebaut und verfeinert. Ein einfaches Beispiel einer Umgebung und einer erlernten Strategie, welches nicht aus dem Kontext der Kartierung stammt, ist in Abbildung 4.6 zu sehen.

Formal lässt sich ein Reinforcement-Learning-Problem über eine Menge von Zuständen S, eine Menge von Aktionen A und eine Belohnungsfunktion $R : S \times A \to \mathbb{R}$ definieren. Im Beispiel aus Abbildung 4.6 besteht die Menge der Zustände aus allen Feldern $S = \{$`a1, ..., e5`$\}$, die Aktionen sind definiert als $A = \{$ `links, rechts, oben, unten` $\}$ und die Beloh-

nungsfunktion entspricht

$$R(s) = \begin{cases} 10 & \text{, falls } s = \texttt{b1} \\ 5 & \text{, falls } s = \texttt{d1} \\ 0 & \text{sonst} \end{cases}$$

Ziel eines Lernverfahrens ist es, eine Strategie $\pi : S \rightarrow A$ zu bestimmen, die für jeden Zustand die Aktion mit der höchsten zu erwartenden Belohnung angibt. Aus diesem Grund wird den Zuständen ein Nutzen $U^{\pi}(s)$ zugeordnet, der dem Erwartungswert der zukünftig generierten Belohnungen entspricht [Russel u. Norvig, 1994]:

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t \, R(s_t) \mid \pi, s_0 = s\right] \tag{4.15}$$

Der Nutzen hängt von der verfolgten Strategie π ab, da diese vorgibt, welche Aktionen ausgewählt werden und damit auch welche Belohnungen generiert werden. Durch den Parameter γ, der zwischen 0 und 1 gewählt wird, kann gesteuert werden, ob aktuelle oder zukünftige Belohnungen ein größeres Gewicht haben sollen. Der tatsächliche Wert eines Zustands wird definiert als der Wert des Zustands bei Verwendung der optimalen Strategie π^*: $U(s) = U^{\pi^*}(s)$.

Es existiert ein direkter Zusammenhang zwischen dem Nutzen eines Zustandes und dem Nutzen der von dort erreichbaren Zustände [Russel u. Norvig, 1994]:

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s') \, U(s') \tag{4.16}$$

Dabei gibt $T(s, a, s')$ die Übergangswahrscheinlichkeit von Zustand s nach s' durch Aktion a an und entspricht somit dem Aktionsmodell des Bayesfilters. Gleichung (4.16) wird *Bellman-Gleichung* genannt und liegt vielen Lernverfahren zugrunde, da die Nutzenwerte $U(s)$ Lösungen der Menge von Bellman-Gleichungen sind [Russel u. Norvig, 1994].

Es gibt eine Reihe verschiedener RL-Algorithmen, die sich vor allem im Wissen unterscheiden, dass über die Umwelt vorhandenen ist. Im günstigsten Fall ist vollständig bekannt, mit welcher Umgebung der Agent interagiert. Insbesondere ist dann bekannt, in welchem Folgezustand sich der Agent nach einer Aktion befinden wird und welche Belohnung er erwarten kann. Im ungünstigsten Fall ist weder der Folgezustand, noch die vollständige Umgebung, noch die zu erwartende Belohnung bekannt.

In dieser Arbeit verwenden wir den SARSA-Algorithmus, der ohne vollständiges Modell der Umgebung auskommt [Sutton u. Barto, 1998]. Die Interaktion mit der Umgebung erfolgt in einer Reihe von abgeschlossenen Handlungssequenzen, die *Episoden* genannt werden. Neben der

Definition der Zustands- und Aktionsmenge wird lediglich eine geeignete Belohnungsfunktion benötigt. Ein explizites Aktionsmodell ist nicht erforderlich. Gelernt wird eine Bewertungsfunktion $Q : S \times A \rightarrow \mathbb{R}$ für Zustand-Aktion-Paare, die aus den erhaltenen Belohnungen geschätzt wird.

Es stellt sich die Frage, welche Strategie zur Erzeugung von Lernepisoden genutzt wird. In dieser Arbeit wird eine *ϵ-greedy* Strategie verwendet, die mithilfe der Bewertungsfunktion generiert wird. Eine Strategie, die in einem Zustand immer die Aktion auswählen, welche in diesem Zustand bisher die höchste Bewertung erzielt hat, wird im Englischen *greedy* genannt. Es empfiehlt sich jedoch während der Lernphase mit einer bestimmten Wahrscheinlichkeit ϵ eine zufällige, explorative Aktion zu wählen, um sicherzustellen, dass alle Zustand-Aktion-Paare bewertet werden. Hierbei spricht man dann von einer *ϵ-greedy* Strategie. Eine weitere Möglichkeit, eine Strategie zu generieren, ist die Boltzmann Auswahl [Sutton u. Barto, 1998]. Hier hängt die Wahrscheinlichkeit für die Auswahl einer Aktion davon ab, wie gut sie durch Q relativ zu den alternativen Aktionen bewertet wird. Je höher der Unterschied, desto wahrscheinlicher wird die besser bewertete Aktion ausgewählt.

Die einzelnen Schritte des SARSA-Algorithmus gehen aus Algorithmus 12 hervor. Die Bewertungen $Q(s, a)$ aller Paare $\langle s, a\rangle$ werden zu Beginn mit beliebigen Werten (z.B. Null) belegt. Dann werden solange Aktionen ausgeführt bis ein terminaler Zustand erreicht wird[1] und die Episode beendet ist. SARSA bewertet eine Aktion basierend auf der aktuellen Belohnung und der Bewertung des nachfolgenden Zustand-Aktion-Paares. Die Bewertung einer ausgeführten Aktion erfolgt daher erst nachdem die nächste Aktion bereits bestimmt wurde. Die Bewertung erfolgt nach folgender Gleichung[2]:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)] \tag{4.17}$$

$$= (1-\alpha)\, Q(s,a) + \alpha[r + \gamma Q(s',a')] \tag{4.18}$$

Abhängig von einer Lernrate α wird die bisherige Bewertung $Q(s, a)$ eines Paares $\langle s, a\rangle$ durch die aktuelle Belohnung r und die Bewertung $Q(s', a')$ des nachfolgenden Zustand-Aktion-Paares $\langle s', a'\rangle$ aktualisiert. Der Faktor γ steuert, wie stark die Bewertung des Nachfolgepaares in die Bewertung einbezogen werden soll. Der Schrittweiten- oder Lernparameter α wird verwendet, um die neue Bewertung des Zeitschritts gegenüber der bisherigen Bewertung zu gewichten. SARSA ist ein iteratives Verfahren, das eine Reihe von Durchläufen (Episoden) benötigt, um zu einer stabilen Bewertung aller Paare zu kommen. Eine Gegenüberstellung der SARSA-Aktualisierungsgleichung (4.18) und der Bellman-Gleichung (4.16) zeigt klare

[1] Ein Zustand gilt als terminal, wenn er nur sich selbst als Nachfolgezustand hat und keine Belohnung generiert.
[2] Der Name SARSA leitet sich von den verwendeten Variablen in der Gleichung ab: s, a, r, s', a'

Algorithmus 12 SARSA

Initialisiere $Q(s, a)$ beliebig
for all Episoden **do**
  initialisiere Startzustand s
  Wähle a in s unter Verwendung der aus Q bestimmten Strategie
  repeat
    Führe Aktion a aus, beobachte $r = R(s), s'$
    Wähle a' in s' unter Verwendung der aus Q bestimmten Strategie
    $Q(s, a) = Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
    $s = s'; a = a'$
  until s terminal
end for

Parallelen auf:

$$Q(s,a) = (1-\alpha)\, Q(s,a) + \alpha\, [\quad r \quad + \gamma \qquad Q(s',a') \qquad\qquad] \tag{4.19}$$

$$U^{\pi^*}(s) = R(s) + \gamma \max_a \sum_{s'} T(s,a,s')\, U^{\pi^*}(s') \tag{4.20}$$

Tatsächlich kann gezeigt werden, dass SARSA zu einer optimalen Strategie und Bewertungsfunktion konvergiert, falls alle Zustand-Aktion-Paare unendlich oft besucht werden und die Strategie zu einer Strategie ohne Exploration konvergiert [Singh u. a., 2000].

Definition des Modellwahl-Problems

Das Problem der Modellauswahl lässt sich als RL-Problem beschreiben. Dazu müssen entsprechende Zustände, Aktionen und eine Belohnungsfunktion definiert werden. Die Menge möglicher Aktionen $A := \{a_g, a_f\}$ ergibt sich direkt aus der algorithmischen Beschreibung in Abschnitt 4.2.4. Entweder wird die Gridkarte verwendet (a_g) und die Landmarken-Karte ignoriert oder umgekehrt (a_f).

Die Zustandsmenge sollte alle für die Entscheidung notwendigen Informationen über die aktuellen Messungen und den Zustand des Filters beinhalten. Wir definieren den aktuellen Zustand des Systems daher über die durchschnittliche Laser-Messwahrscheinlichkeit $\bar{l}$, die Information, welcher der beiden N_{eff}-Werte größer ist, sowie ob bekannte Landmarken erkannt wurden:

$$S = \{\bar{l}\} \times \{\mathbb{1}_{N^f_{eff} < N^g_{eff}}\} \times \{\mathbb{1}_{\text{ Landmarke erkannt}}\} \tag{4.21}$$

Die Komplexität des Lernalgorithmus beträgt $O(|S|\ |A|)$, hängt also von der Größe der Zustandsmenge und der Anzahl der Aktionen ab [Ya-Ping Lin, 2003]. Es empfiehlt sich daher, möglichst wenige Zustände zu verwenden. Aus diesem Grund wird der Wert der durchschnittlichen Messwahrscheinlichkeit $\bar{l}$ in diskrete Intervalle unterteilt ($0.0-0.15$, $0.16-0.3$, $0.31-0.45$,

$0.46 - 0.6$, $0.61 - 0.75$, $0.76 - 0.9$, $0.91 - 1.0$). Daraus ergeben sich $7 \times 2 \times 2 = 28$ Zustände und bei zwei Aktionen 54 Zustand-Aktion-Paare.

Die Belohnungsfunktion soll gut korrigierte Roboterpositionen positiv, unkorrigierte oder falsch korrigierte Positionen dagegen negativ bewerten. Die Strategie wird anhand von simulierten Roboterdaten gelernt, weshalb während des Lernens die korrekte Position x_t^* des Roboters bekannt ist. Diese Information wird verwendet, um für jeden Zeitschritt t den gewichteten Durchschnitt der Positionsabweichung zu bestimmen. Um eine schlechte Bewertung zu vermeiden, die auf falsch gewählte Aktionen in der Vergangenheit zurückgeht, verwenden wir nur die Abweichung seit dem letzten Bewertungsschritt $t-1$:

$$R(s_t) \quad = \quad R(s_{t-1}) \; - \; \sum_{i=1}^{N} w_t^{(i)} \; ||x_t^{(i)} - x_t^*|| \tag{4.22}$$

Abweichungen vom simulierten Roboterpfad resultieren in einer Bestrafung. Wie oben erwähnt, besitzt jedes Partikel zwei Gewichte. Für die Berechnung der mittleren Abweichung verwenden wir $w_f^{(i)}$, falls die letzte Aktion a_f war, sonst $w_g^{(i)}$.

Die Formulierung des Modellauswahl-Problems als RL-Problems ermöglicht es nun, eine Auswahlstrategie anhand einer typischen Beispielumgebung zu erlernen. Dazu wurden simulierte Roboterdaten verwendet. Der Datensatz wurde in der Simulationsumgebung des Carnegie Mellon Robot Navigation Toolkit (CARMEN) [Montemerlo u. a., 2002a] erzeugt, das die Generierung realistischer Echtzeitdaten verschiedener Roboter- und Sensortypen erlaubt.

Lernumgebung

Die simulierte Lernumgebung besteht aus einer hausähnlichen Struktur und einem Freigelände, dass einige Hindernisse beinhaltet, die man sich als Laternenmasten oder Bäume vorstellen kann (siehe Abbildung 4.7). Die maximale Reichweite des simulierten Lasers beträgt 4m, was weniger als der Abstand zwischen den Laternen auf der Freifläche ist. Auf diese Weise kann im Außenbereich durch die Verwendung des Landmarkenextraktors eine Positionskorrektur durchgeführt werden, während das Grid-basierte Verfahren in manchen Abschnitten keine Korrektur durchführen kann. Typische Ergebnisse der einzelnen Verfahren zeigt Abbildung 4.8. Keines der Verfahren kann die Lernumgebung korrekt kartieren, es handelt sich daher um einen geeigneten Datensatz, um eine kombinierte Strategie zu erlernen.

Die simulierten Daten wurden 1000 mal durchlaufen. Während des Lernens wurde ein bestimmter Prozentsatz der Aktionen zufällig gewählt. Dadurch wird sichergestellt, dass alle Zustand-Aktion-Paare besucht werden. Das Ergebnis des Lernverfahrens ist eine Strategie,

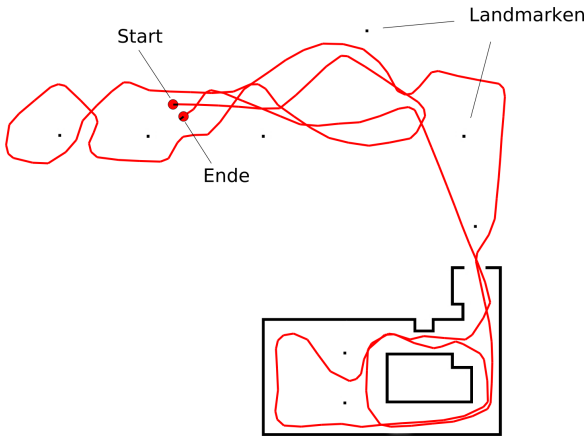


Abbildung 4.7.: Simulierte Lernumgebung bestehend aus einem Innen- und Außenbereich. Der simulierte Roboterpfad ist rot eingezeichnet.

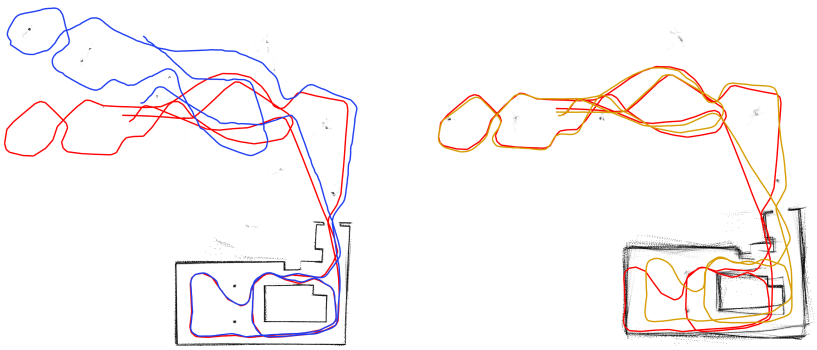


Abbildung 4.8.: Beispielhafte Ergebnisse des Grid-Verfahrens (links) und des Landmarken-Verfahrens (rechts). Der korrekte simulierte Pfad ist jeweils in rot eingezeichnet.

die für jeden Zustand die Aktion mit der besten erwarteten Belohnung angibt. Auf diese Weise kann nun basierend auf den aktuellen Messdaten, Karten und den beiden N_{eff}-Werten eine Modellauswahl getroffen werden. In den im Folgenden beschriebenen Experimenten wird diese gelernte Strategie verwendet, um in jedem Zeitschritt zwischen Landmarken- und Gridkarte zu entscheiden.

5. Experimente

Der vorgestellte kombinierte SLAM-Algorithmus wurde in einer Reihe von Experimenten evaluiert. Es wurden sowohl simulierte als auch echte Roboterdaten verwendet. Die Experimente sollen zeigen, dass der kombinierte Ansatz gegenüber dem reinen Landmarken- oder Gridkarten-Ansatz zu einem geringeren Fehler in der Kartierung führt. Als reines Gridverfahren verwenden wir die Implementierung aus [Grisetti u. a., 2005]. Als reines Landmarken-Verfahren verwenden wir eine Implementierung des FastSLAM-Algorithmus entsprechend der Arbeit von Montemerlo u. a. [2002b].

5.1. Simulation

In einem Experiment mit simulierten Daten, in dem Bedingungen wie die Umgebung und die Ungenauigkeit in den Messungen kontrolliert werden können, sollte zunächst gezeigt werden, dass der vorgestellte kombinierte Ansatz signifikant besser als die Einzelverfahren ist. Darüber hinaus sollten die in Abschnitt 4.2.5 vorgestellten Auswahlheuristiken anhand des simulierten Datensatzes verglichen werden.

Dazu wurden simulierte Roboterdaten mit dem Carnegie Mellon Robot Navigation Toolkit (CARMEN) generiert [Montemerlo u. a., 2002a]. In unseren Experimenten wurde ein Laserscanner mit einem Sichtbereich von 180° und einer Auflösung von einer Messung pro Grad simuliert. Der simulierte Roboter weist einen normalverteilten Fehler in Rotation und Translation auf.

Das Testszenario, bestehend aus einer zweidimensionalen Umgebung und einem simulierten Roboterpfad, ist in Abbildung 5.1 zu sehen. Die Umgebung besteht aus zwei symmetrischen

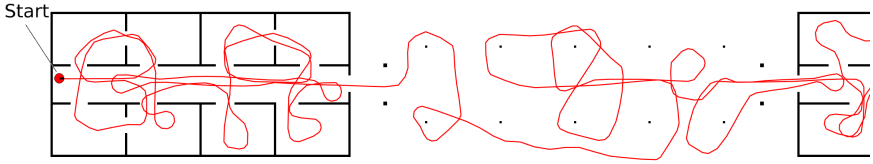


Abbildung 5.1.: Karte und Roboterpfad der simulierten Testumgebung.

Gebäuden, die über eine Allee miteinander verbunden sind. Die Bäume der Allee (punktförmige Hindernisse) stehen in einem Abstand von 5m zueinander. Die Umgebung misst insgesamt etwa 70m. Der simulierte Laserscanner hat eine maximale Reichweite von 4m, also weniger als der Abstand von einem Baum zum nächsten. Reale Sensoren haben oft eine Reichweite von bis zu 80m (z.B. SICK LMS), kleinere und günstigere Sensoren können aber eine sehr viel geringere Reichweite von beispielsweise maximal 4m aufweisen (z.B. Hokuyo URG).

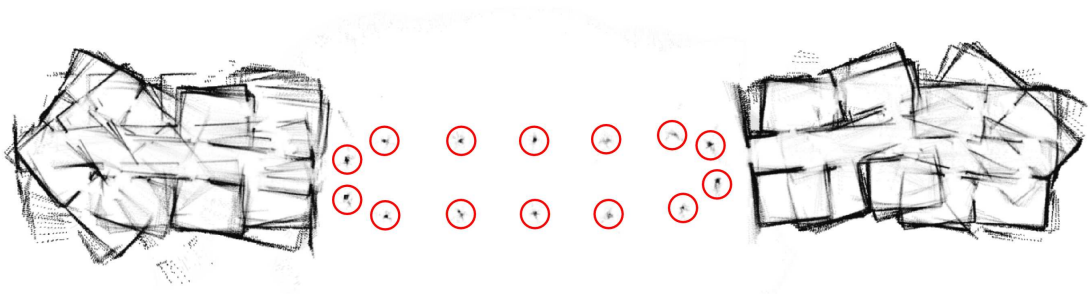
Um die verschiedenen SLAM-Ansätze vergleichen zu können, wurde der simulierte Datensatz mit jedem der Verfahren mehrfach durchlaufen. Alle Verfahren verwenden hierbei 50 Partikel und integrieren Messungen, sobald der simulierte Roboter 0.25m gefahren ist oder eine Rotation von mehr als 0.25 rad (28°) durchgeführt hat.

Ein beispielhaftes Ergebnis des landmarkenbasierten Verfahrens wird in Abbildung 5.2(a) dargestellt. Da innerhalb des Gebäudes keine Landmarken vorhanden sind, kann das Verfahren hier keine Korrektur des Roboterpfades vornehmen. Diese Fehler in der Positionsbestimmung, insbesondere in der Ausrichtung des Roboters, führen zu deutlich erkennbaren Fehlern in der resultierenden Karte. In der Allee zwischen den Gebäuden dagegen, werden die freistehenden Bäume als Punktlandmarken erkannt und der Pfad anhand dieser Messungen korrigiert.

Der Gridkartenansatz ist in der Lage, die Struktur der Gebäude abzubilden und somit Positionskorrekturen anhand von Lasermessungen in den Gebäuden durchzuführen. In der Allee ist dies jedoch nur eingeschränkt möglich, da hier nur wenige Hindernisse vom simulierten Laserscanner erfasst werden. Die Folge ist ein Fehler in der Positionsschätzung, der sich auch hier vor allem in der Ausrichtung des Roboters bemerkbar macht. Diese Fehler führen zu einer falschen Kartierung der Allee, wie Abbildung 5.2(b) exemplarisch zeigt.

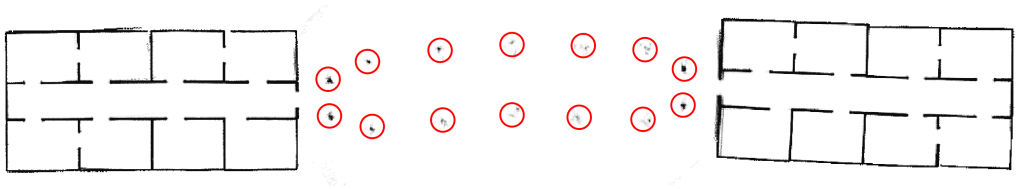
Das vorgestellte kombinierte Verfahren ist in der Lage, beide Abschnitte der simulierten Umgebung korrekt zu kartieren (siehe Abb. 5.2(c)). Die gelernte Modellauswahl sorgt dafür, dass im Inneren der Gebäude die Gridkarten zur Positionsschätzung verwendet werden, während in der Allee die gemessenen Landmarken benutzt werden.

Um dieses Ergebnis quantitativ zu bestätigen, wurde das Experiment mit jedem der Verfahren zwanzig mal wiederholt und dabei der Fehler in der Kartierung gemessen. Vor jedem Durchlauf wurde der Zufallsgenerator des Systems mit einem anderen Startwert initialisiert. Die stochastische Natur der Stichprobengenerierung in einem Partikelfilter sorgt dann dafür, dass verschiedene Durchläufe geringfügige Abweichungen voneinander aufweisen. Die mehrfache Ausführung eines Experiment dient allgemein dazu, auszuschließen, dass ein bestimmtes Verfahren zufällig in einem Durchlauf besser ist als ein anderes.



(a) Landmarkenbasiertes System

(b) Gridkartenbasiertes System



(c) Kombinierter SLAM-Ansatz

Abbildung 5.2.: Exemplarische Ergebnisse verschiedener SLAM-Verfahren nach Durchlaufen der simulierten Testdaten aus Abb. 5.1.

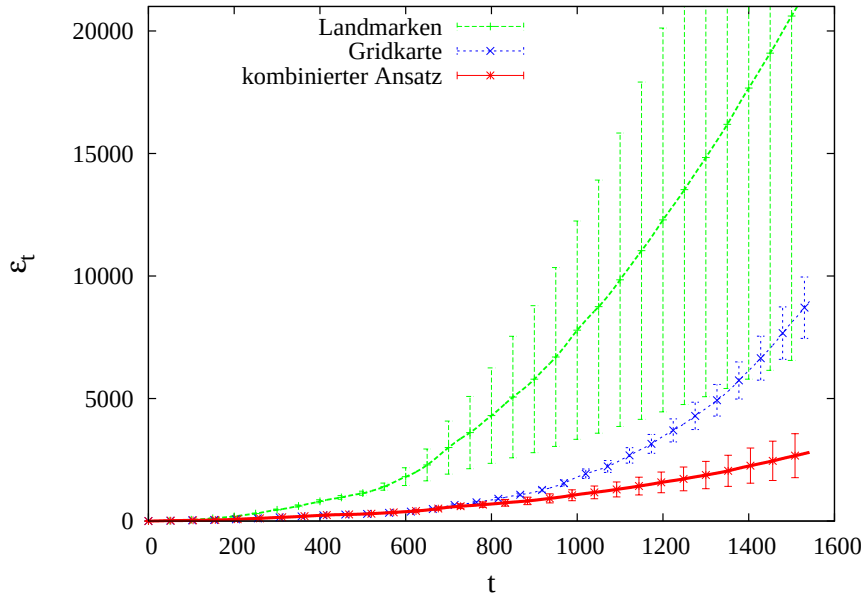


Abbildung 5.3.: Kumulativer gewichteter Fehler in der Positionsschätzung. Die Fehlerbalken stellen das 0.05 Konfidenzintervall dar.

Es gilt generell als problematisch, die Güte einer erzeugten Karte zu messen. Da der wahre Roboterpfad durch die Simulationsumgebung bereitgestellt wird, haben wir uns entschieden, in jedem Zeitschritt den Fehler zwischen wahrer und geschätzter Roboterposition zu bestimmen. Um darüber hinaus ein Maß für die Güte der Kartierung bis zu einem bestimmten Zeitschritt zu erhalten, wird der kumulative Fehler berechnet:

$$\varepsilon_t \quad = \quad \sum_{k=1}^{t} \sum_{i=1}^{N} w_k^{(i)} \, ||x_k^{(i)} - x_k^*|| \tag{5.1}$$

Wie auch beim Lernen der Modellauswahl handelt es sich hierbei um den gewichteten Fehler zwischen der zum Zeitpunkt k wahren Position x_k^* und den Positionshypothesen der Partikel $x_k^{(i)}$.

Die Ergebnisse der quantitativen Evaluation sind in Abbildung 5.3 zu sehen. Deutlich zu erkennen ist ein starker Anstieg des Positionsfehlers bei beiden Einzelverfahren. Der kombinierte Ansatz weist einen im Vergleich dazu geringen Fehler auf. Wie man anhand der Fehlerbalken sieht, ist dieser Unterschied signifikant (Signifikanzintervall 0.05).

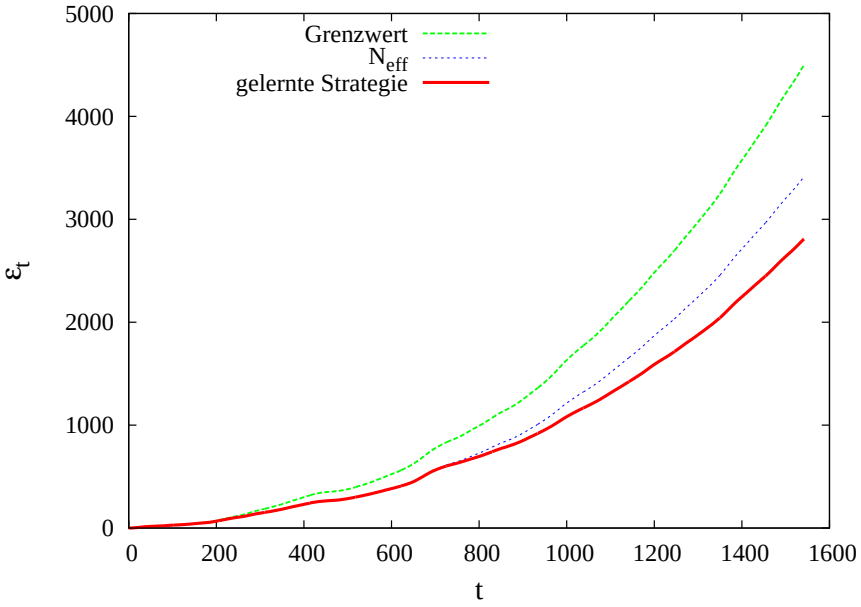


Abbildung 5.4.: Kumulativer gewichteter Fehler in der Positionsschätzung bei Verwendung verschiedener Methoden der Modellauswahl.

5.1.1. Vergleich der Heuristiken zur Modellauswahl

Ein weiteres Ziel des Experiments war die Bewertung der verschiedenen Strategien der Modellauswahl. Aus diesem Grund wurden zusätzliche Durchläufe mit den verschiedenen Auswahlheuristiken ausgeführt. Verglichen wurden die in Abschnitt 4.2.5 vorgestellten Methoden der Modellauswahl: ein Grenzwert der Messwahrscheinlichkeit, die N_{eff}-Heuristik sowie die gelernte Auswahlstrategie. Die Experimente wurden wiederum zwanzig mal mit verschiedenen Initialisierungen des Zufallsgenerators durchgeführt. Dabei wurde der in Gleichung (5.1) definierte kumulative Fehler gemessen.

Die Ergebnisse in Abbildung 5.4 zeigen, dass die gelernte Strategie besser als die heuristischen Methoden ist. In diesem Szenario lässt sich allerdings nicht zeigen, dass der Unterschied signifikant ist. Betrachtet man jedoch beispielhafte Ergebnisse bei Verwendung der verschiedenen Auswahlmethoden, können bestimmte charakteristische Fehler ausgemacht werden. Diese Fehler müssen sich nicht zwangsläufig in einer deutlichen Abweichung von der wahren Position niederschlagen. Ein geringer Fehler beispielsweise in der Ausrichtung des Roboters kann dennoch zu sichtbaren Fehlern in der resultierenden Karte führen. Besonders die Grenzwertheuristik ist anfällig für diese Art von Fehler. Hier werden Landmarken verwendet, sobald die Messwahrscheinlichkeit $\bar{l}$ unter einen festen Grenzwert fällt (siehe Gleichung 4.10). Wird dieser Parameter nicht optimal gewählt, kann es vorkommen, dass das falsche Umgebungsmodell ausgewählt wird und keinerlei Korrektur der Odometriefehler möglich ist. Typische Fehler in

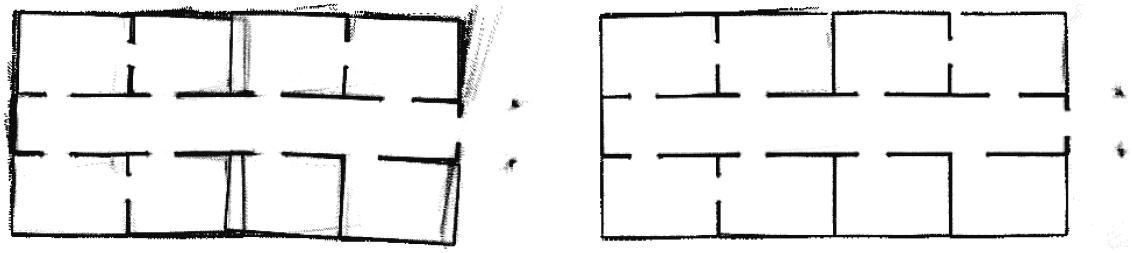


Abbildung 5.5.: Typische Ergebnisse bei der Verwendung der Grenzwert-Heuristik (links) und der gelernten Modellauswahl (rechts).

der resultierenden Karte sind beispielsweise unscharfe oder geringfügig falsch ausgerichtete Wände. Abbildung 5.5 illustriert diese Fehler anhand eines Kartenausschnitts.

Die N_{eff}-Heuristik kann zu ähnlichen Fehlern führen, wobei hier besonders die Übergänge zwischen Gebieten der Umgebung betroffen sind, die besser für Grid- oder Landmarkenkarten geeignet sind. Der Grund hierfür liegt in der Tatsache, dass sich der Unterschied zwischen den Varianzen der beiden Partikelgewichtsmengen gewöhnlich nicht sprunghaft ändert. Es kann einige Zeitschritte dauern, bis die Varianz in den Landmarkengewichten größer ist als die Varianz in den Gridgewichten (beziehungsweise umgekehrt) und das System das besser geeignete Modell nach dieser Heuristik auswählt. Ein weiteres Problem bei der Verwendung dieser Heuristik ist die mögliche Degenerierung des Filters. Wie bereits erwähnt, kann häufiges Resampling zur Eliminierung guter Hypothesen führen. Durch die Auswahl des Modells mit der höchsten Varianz in den Partikelgewichten und damit dem geringsten N_{eff} kann es bei Verwendung des adaptiven Resamplings zu häufigeren Resampling-Schritten kommen.

5.1.2. Laufzeiten

Durch die parallele Verwaltung zweier Umweltrepräsentationen entsteht zusätzlicher Berechnungsaufwand, der sich in der Laufzeit des Algorithmus niederschlägt. Um diesen Mehraufwand zu bestimmen, wurde die durchschnittliche Gesamtlaufzeit der verschiedenen Verfahren aus zwanzig Durchläufen des simulierten Testdatensatzes bestimmt. Die Ergebnisse in Abbildung 5.6 zeigen, dass die Rechenzeit in diesem Szenario im Wesentlichen durch die Gridkartierung mit verbesserter Vorhersage bestimmt wird. Der Zusatzaufwand, welcher durch die Verwaltung der Landmarken und der zusätzlichen Partikelgewichte entsteht, liegt aufgrund der geringen Landmarkenmenge in diesem Szenario bei weniger als einem Prozent. Eine aufwändigere Landmarken-Extraktion beispielsweise aus Kameradaten oder eine deutlich größere Anzahl vom gemessenen Landmarken könnten die Laufzeit jedoch entscheidend vergrößern.

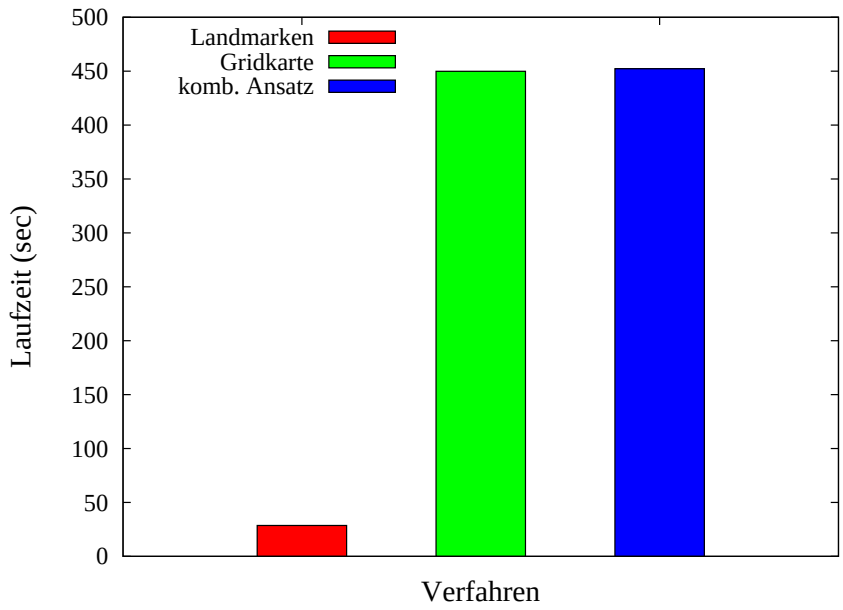


Abbildung 5.6.: Durchschnittliche Laufzeit der verschiedenen Verfahren für einen Durchlauf des simulierten Datensatzes. Die geringe Laufzeit des Landmarkenansatzes erklärt sich auch durch die kleine Menge von Landmarken in der Umgebung.

Abbildung 5.7.: ActivMedia Pioneer 2-AT Roboter mit einem SICK LMS Laserscanner.

5.2. Echte Roboterdaten

Die in der Simulation gewonnen Ergebnisse sollen anhand von Experimenten mit echten Roboterdaten bestätigt werden. Es soll gezeigt werden, dass das kombinierte SLAM-Verfahren auch unter realen Bedingungen zu besseren Ergebnissen führt als ein reines Gridkarten- oder Landmarkenverfahren. Zu diesem Zweck wurden Experimente mit einem ActivMedia Pioneer 2-AT Roboter durchgeführt, der mit einem SICK LMS Laserscanner ausgestattet ist. Der Roboter ist in Abbildung 5.7 zu sehen. Er ist sowohl für den Einsatz im Inneren von Gebäuden, als auch für den Einsatz im Freien geeignet. Sein vierrädriger Antrieb kann allerdings vor allem auf glattem Asphalt schnell zu deutlichen Fehlern in der Odometrie führen, da die Räder hier beim Steuern oft über den Boden rutschen. Der Laserscanner hat einen Sichtbereich von 180° bei einer Auflösung von einer Messung pro Grad und einer maximale Reichweite von 80m. Für die Dauer der Experimente war der Sensor starr in Voraussrichtung des Roboters fixiert.

Die Experimente wurden auf dem Campus der Fakultät für Angewandten Wissenschaften in Freiburg durchgeführt. Abbildung 5.8 zeigt ein Luftbild des Testgeländes.

5.2.1. Experiment 1: Künstliche Landmarken

Das erste Experiment wurde in Gebäude 079 der Fakultät sowie auf der Straße vor dem Gebäude durchgeführt. Da das Gelände bis auf zwei Laternenmasten keine freistehenden natürlichen Hindernisse aufweist, wurden künstliche Landmarken in einem Abstand von 2 bis 5 Metern aufgestellt. Es handelt sich dabei um Papprohre mit einem Durchmesser von ca. 15cm und einer Länge von etwa 1m. Das Testszenario ist in Abbildung 5.9 (a) zu sehen. Zur Datenaufnahme wurde der Roboter manuell gesteuert. Der Datensatz beginnt im Freien in der Nähe des Hauses. Von hier wurde der Roboter zunächst entlang der künstlichen

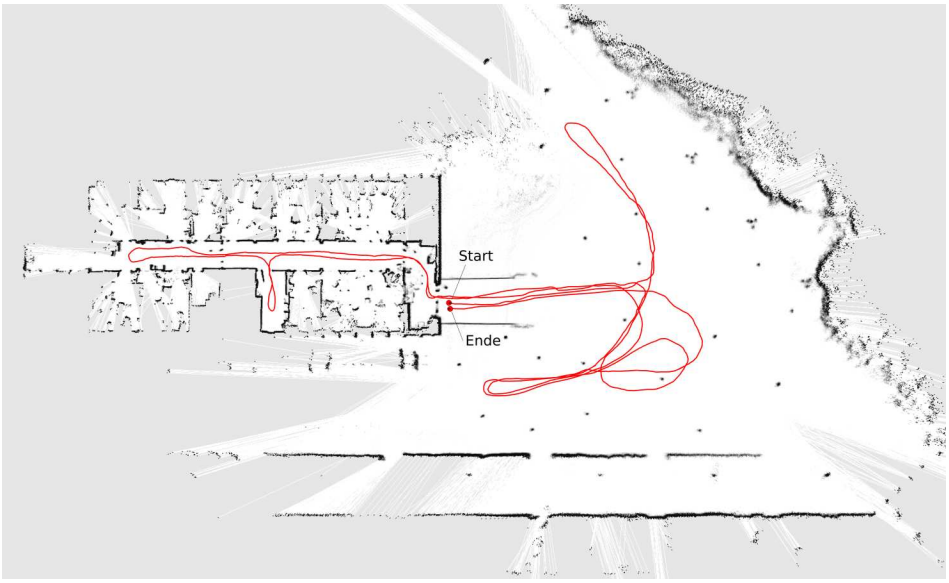
Abbildung 5.8.: Luftbild des Campus der Fakultät für Angewandten Wissenschaften in Freiburg [Maps, 2007].

Landmarken auf der Straße gesteuert. Anschließend wurde das Untergeschoss von Gebäude 079 durchfahren. Schließlich wurde erneut ein Teil der Freifläche durchquert und der Parcours in der Nähe der Startposition beendet. Die Sensorreichweite des Laserscanners wurde in diesem Experiment auf 5m reduziert, um zu verhindern, dass benachbarte Gebäude oder andere große Hindernisse wie beispielsweise Hecken in den Messbereich des Roboters gelangen.

Wie auch im Simulationsexperiment, werden die Kartierungsergebnisse des kombinierten Ansatzes mit denen des reinen Landmarken- und Gridverfahrens verglichen. In diesem Experiment ist der wahre Pfad des Roboters allerdings nicht bekannt. Zur Evaluation der Ergebnisse muss der wahre Pfad daher angenähert werden. Zu diesem Zweck wurden die Roboterdaten mit dem bereits erwähnten Verfahren von Grisetti u. a. bei voller Sensorreichweite (80m) korrigiert. Die hohe Sensorreichweite führt dazu, dass immer genug Hindernisse gemessen wurden, um Positionsfehler zu korrigieren. Der so gewonnene Roboterpfad stimmt mit dem wahren Pfad weitestgehend überein. Er ist in Abbildung 5.9 (b) zu sehen.

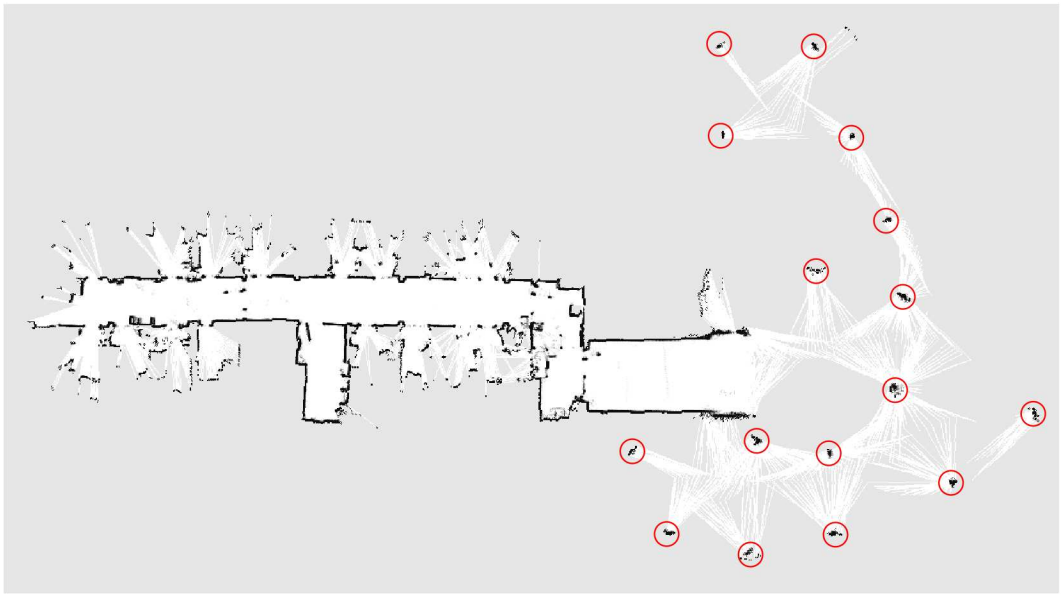
Betrachtet man die beispielhaften Ergebnisse in Abbildung 5.10, zeigen sich deutliche Unterschiede in der Qualität der Karten. Während das Gridkarten-Verfahren gute Ergebnisse im Inneren des Gebäudes liefert, kann der Pfad im Freien nicht gut korrigiert werden. Umgekehrt kann das Landmarken-Verfahren den Außenbereich gut kartieren, muss aber im Inneren des Gebäudes auf unkorrigierte Odometriemessungen zurückgreifen. Die Umgebungskarten beider Einzelverfahren sind daher nicht konsistent, besonders im Bereich der Rampe (Start- und Zielbereich) zeigen sich deutliche Inkonsistenzen.

(a) Foto des Testszenarios.

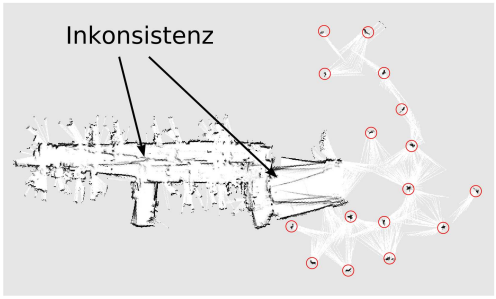


(b) Gridkarte des Szenarios und angenähertem wahren Roboterpfad (rot).

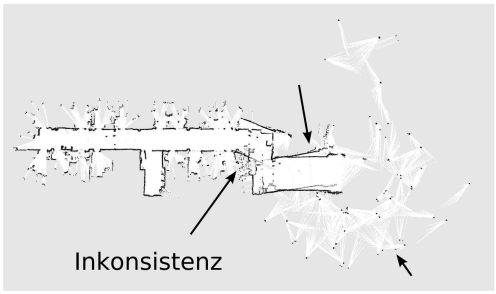
Abbildung 5.9.: Experiment 1: Künstliche Landmarken.



(a) Kombinierter Ansatz



(b) Feature SLAM



(c) Grid SLAM

Abbildung 5.10.: Exemplarische Kartierungsergebnisse bei Experiment 1.

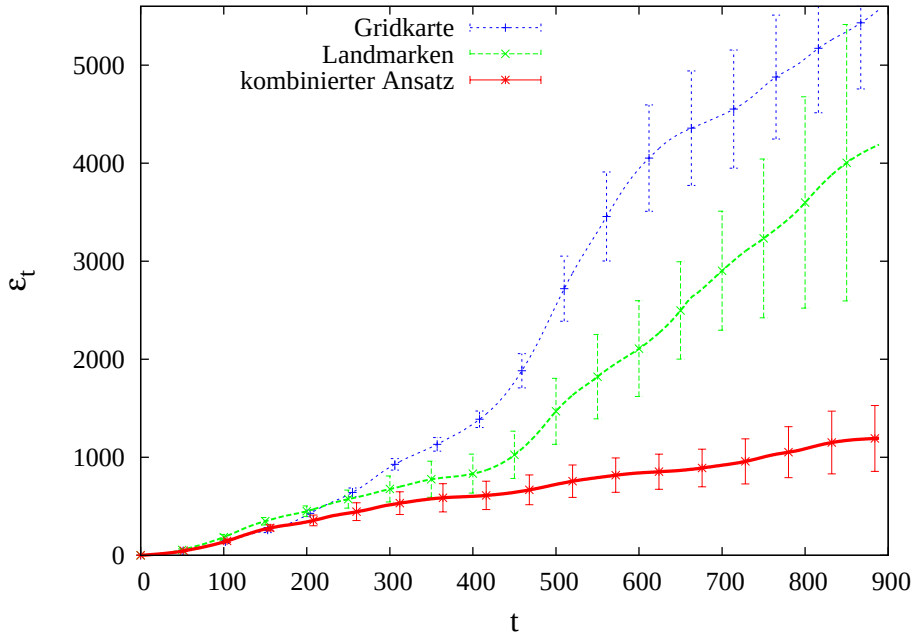


Abbildung 5.11.: Kumulativer Fehler in der Positionsschätzung gemessen an einer angenäherten wahren Robotertrajektorie. Die Fehlerbalken stellen das 0.05 Konfidenzintervall dar.

Der kombinierte Ansatz dagegen weist sowohl im Innen- als auch im Außenbereich des Testszenarios gute und in sich konsistente Ergebnisse auf. Im Gegensatz zu unserem kombinierten Verfahren, sind die Einzelverfahren nicht in der Lage, konsistente Modell zu erzeugen.

Um Zufallseinflüsse zu verringern, wurde der durchschnittliche Fehler über zwanzig Durchläufen ermittelt. Als Fehlermaß wurde wiederum die kumulative Pfadabweichung gewählt. Die Ergebnisse in Abbildung 5.11 zeigen, dass der kombinierte Ansatz einen signifikant geringeren Fehler aufweist (Signifikanzintervall 0.05) als die beiden klassischen Ansätze und bestätigen somit die Ergebnisse der Simulation.

Abbildung 5.12.: Parkplatz der Fakultät für Angewandte Wissenschaften in Freiburg.

5.2.2. Experiment 2: Parkplatz

Auf dem Gelände der Fakultät für Angewandte Wissenschaften befindet sich ein Parkplatz mit einer Größe von ca. 50m x 120m (siehe Abb. 5.12). Auf dem Parkplatz stehen Laternen in zwei Reihen, wobei die Reihen einen Abstand von 25m zueinander haben und die Laternen innerhalb einer Reihe in einem Abstand von 16m stehen. Die Experimente wurden zu einer Zeit durchgeführt, zu der sich keine Fahrzeuge auf dem Parkplatz befanden. Aus diesem Grund sind die Laternenmasten in weiten Teilen des Parkplatzes die einzigen Hindernisse, die vom Laserscanner gemessen wurden und eigenen sich daher als Landmarken. Direkt neben dem Parkplatz befindet sich Gebäude 078 der Universität. Der Roboter wurde zunächst durch dieses Gebäude, dann über den Parkplatz und zuletzt wieder in das Gebäude gesteuert.

Zur Evaluation des kombinierten SLAM-Ansatzes wurde die maximale Reichweite des Laserscanners auf 12m begrenzt, was deutlich weniger als der Abstand zwischen zwei Laternenmasten ist. Auf diese Weise kommt es vor, dass der Roboter auf dem Parkplatz einige Meter fährt ohne eine verwertbare Lasermessung zu erhalten.

Wie schon in Experiment 1, wird zunächst ein angenäherter wahrer Pfad mit einem Gridverfahren bei voller Sensorreichweite bestimmt. Der Pfad, sowie eine Gridkarte der Testumgebung, ist in Abbildung 5.13 zu sehen.

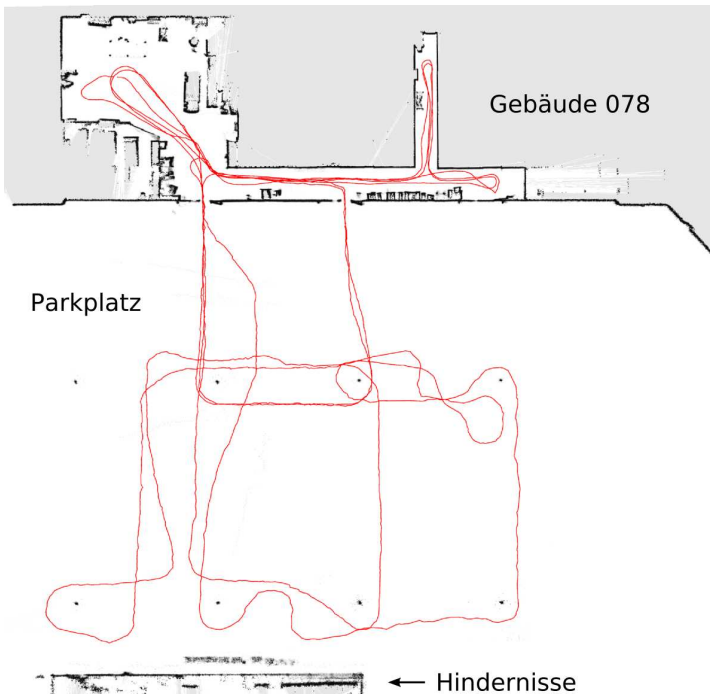


Abbildung 5.13.: Gridkarte des Parkplatzes und des benachbarten Gebäude 078. Der angenäherte Roboterpfad ist in rot zu sehen.

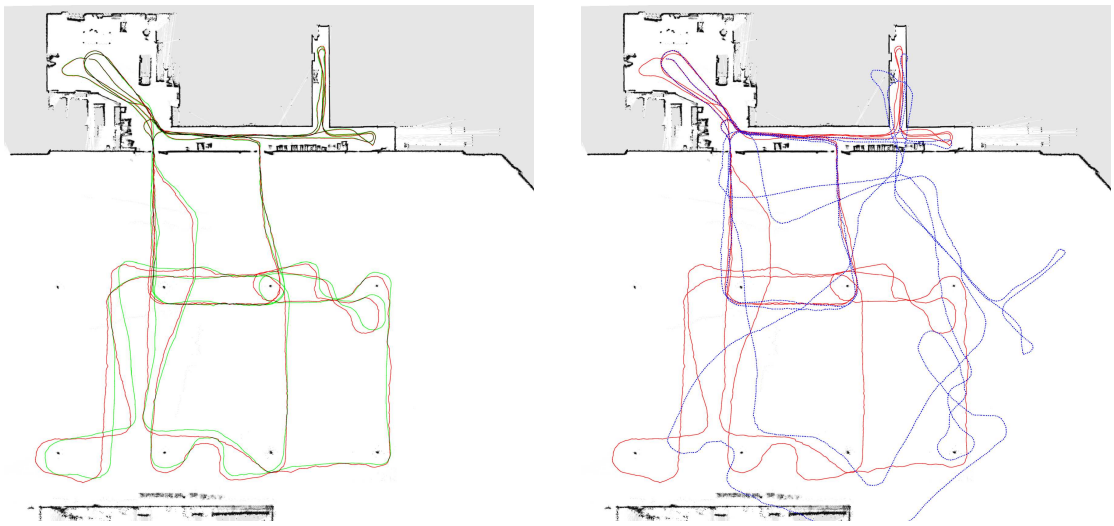


Abbildung 5.14.: Vergleich zwischen angenähertem wahren Roboterpfad (rot) und korrigierten Pfaden. Links: Vergleich zum kombinierten Landmarken/Gridkarten-Verfahren (grün). Rechts: Vergleich zum Gridverfahren (blau).

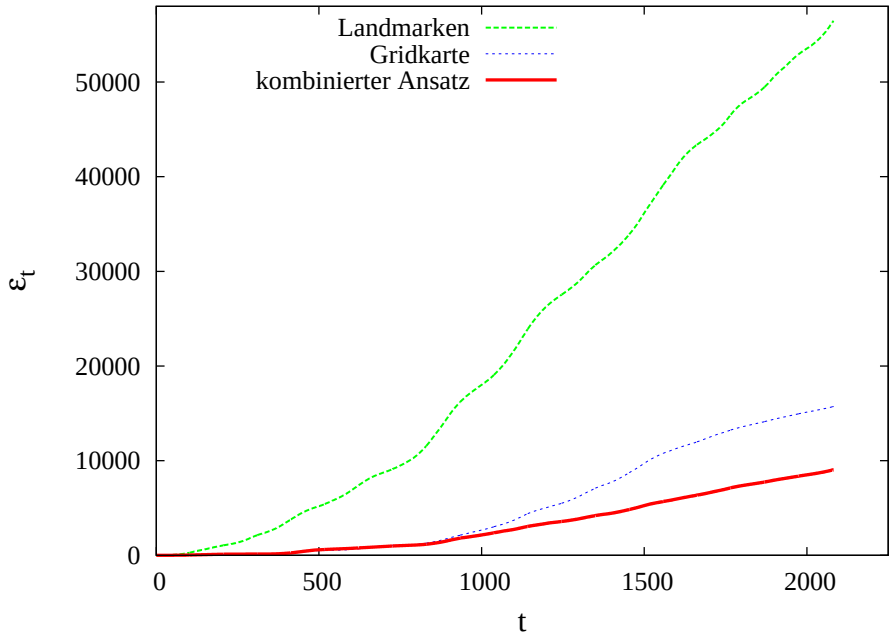


Abbildung 5.15.: Kumulativer Fehler in der Positionsschätzung gemessen an einer angenäherten wahren Robotertrajektorie.

Abbildung 5.14 zeigt einen beispielhaften korrigierten Pfad des kombinierten Ansatzes beziehungsweise des Gridverfahrens im Vergleich mit dem angenäherten wahren Pfad. Im Freien kann der reine Gridansatz die Odometriefehler aufgrund der stark beschränkten Sensorreichweite teilweise nicht mehr korrigieren, was zu einer inkonsistenten Karte führt.
Im Vergleich dazu liefert der kombinierte Ansatz im gesamten Gebiet gute Ergebnisse. Die Ungenauigkeiten im Außenbereich lassen sich durch den enormen Abstand der Landmarken von bis zu 25m und die starken Odometriefehler erklären.

Eine graphische Darstellung der durchschnittlichen kumulativen Fehler, gemessen am angenäherten wahren Roboterpfad, findet sich in Abbildung 5.15. Die Grafik zeigt, dass der kombinierte Ansatz auch in diesem schwierigen Szenario einen deutlich geringeren Fehler aufweist als die beiden klassischen Ansätze und bestätigt die bisherigen Ergebnisse.

Insgesamt zeigen die durchgeführten Experimente, dass das kombinierte SLAM-Verfahren in entsprechenden Umgebungen signifikant besser ist als Verfahren, die nur eine der Umgebungsmodelle verwenden. Die Einzelverfahren weisen entweder einen signifikant höheren Fehler auf oder sind gar nicht in der Lage, konsistente Modelle zu erzeugen. Das vorgestellte kombinierte Verfahren ist dagegen robust gegenüber Fehlern in der Odometrie und führt zu Ergebnissen mit einem deutlich geringeren Gesamtfehler.

6. Zusammenfassung und Ausblick

6.1. Zusammenfassung

In dieser Arbeit wurde ein neues Verfahren zur Erstellung von Umgebungskarten aus Roboterdaten vorgestellt. Es basiert auf einem Rao-Backwellized Partikelfilter und verwendet eine duale Repräsentation der Umwelt. Im Gegensatz zu klassischen Verfahren, werden in unserem Ansatz sowohl Gridkarten als auch Landmarken zur Lösung des SLAM-Problems verwendet. So können während der Kartierung Roboterpositionen und Karten anhand der jeweils am besten geeigneten Modellierung korrigiert werden. Die Auswahl zwischen beiden Modellen wird in jedem Zeitschritt anhand der aktuellen Sensordaten und Eigenschaften des Filters getroffen. Die Strategie, nach der diese Auswahl getroffen wird, wurde anhand von Beispielszenarien erlernt. Dazu wurde das Auswahlproblem als Reinforcement-Learning-Problem formuliert.

Das Verfahren wurde implementiert und anhand einer Reihe von Experimenten evaluiert. Es wurden Experimente sowohl anhand von simulierten Daten, als auch mit einem echten Roboter durchgeführt. Die Experimente zeigen, dass der kombinierte Ansatz gegenüber den klassischen Verfahren, die nur eine der Umweltrepräsentationen verwenden, zu einer robusteren Kartierung führt. Es konnte gezeigt werden, dass das kombinierte Verfahren in den untersuchten Szenarien einen signifikant geringeren Fehler in der Kartierung aufweist. In mehreren Fällen war eine konsistente Kartenerzeugung nur mit unserem kombinierten Verfahren möglich. Sollte die Verwendung einer der beiden Umgebungsmodelle in einer Umgebung nicht möglich sein, weil beispielsweise keine Landmarken erkannt werden, stimmen die Ergebnisse des kombinierten Verfahrens mit denen des entsprechend anderen Einzelverfahrens überein.

6.2. Ausblick

Das entwickelte Verfahren ist unabhängig vom verwendeten Landmarkentyp. Die in dieser Arbeit verwendeten Landmarken stellen nur eine Möglichkeit dar, diesen Ansatz zu verwenden. Interessante Möglichkeiten könnten sich beispielsweise aus der Verwendung von Kamerabasierten Landmarken ergeben. In einer Arbeit von Steder [2007] wurden Landmarken mittels der SURF-Beschreibung [Bay u. a., 2006] aus den Bildern einer auf den Boden gerichteten Kamera extrahiert. Dieser Landmarkentyp erfordert keine spezielle Kenntnis der Umgebung. Basierend auf diesen Landmarken wurde ein SLAM-Verfahren entwickelt, dass insbesondere

auch im Freien gute Ergebnisse liefert. Die hier vorgestellte Arbeit liefert den Rahmen für eine mögliche Kombination dieses Verfahrens mit einem Gridkarten-basierten Verfahren. Besonders für kleinere Roboter, die über einen Laserscanner mit geringer Reichweite und eine Kamera verfügen, könnte ein solches System gut zur Kartierung verschiedenster Umgebungen geeignet sein.

Eine Kombination verschiedener Umgebungsmodelle kann auch in anderen Anwendungsbereichen sinnvoll sein. Statt Landmarken mit Gridkarten könnten auch verschiedene Landmarkentypen miteinander kombiniert werden. So wäre es möglich in unterschiedlichen Umgebungen die dort zu erwartenden Landmarken zu verwenden, beispielsweise Ecken und Kanten im Inneren von Gebäuden und Baumstämme im Freien.

A. Anhang

A.1. Grundlagen der Wahrscheinlichkeitsrechnung

Seien X, Y und Z Zufallsvariablen.
$P(X = x)$ bzw. $P(x)$ bezeichne die Wahrscheinlichkeit für das Ereignis $X = x$.
$P(x, y)$ bezeichne die Wahrscheinlichkeit, dass $X = x$ und gleichzeitig $Y = y$.

Definition der Bedingten Wahrscheinlichkeit:

$$P(x \mid y) = \frac{P(x, y)}{P(y)}$$

Falls x und y stochastisch unabhängig sind gilt:

$$\begin{aligned} P(x, y) &= P(x)\ P(y) \\ P(x \mid y) &= P(x) \end{aligned}$$

Produktregel:

$$P(x, y) = P(x \mid y)\ P(y) = P(y \mid x)\ P(x)$$

Satz der totalen Wahrscheinlichkeit:

$$P(x) = \sum_{y} P(x \mid y)\ P(y)$$

Satz von Bayes:

$$P(x \mid y) = \frac{P(y \mid x)\ P(x)}{P(y)} = \frac{P(y \mid x)\ P(x)}{\sum_{x} P(y \mid x)\ P(x)}$$

Häufig verwendet man folgende äquivalente Schreibweise des Satz von Bayes:

$$P(x \mid y) = \eta \; P(y \mid x) \; P(x) \qquad \text{mit} \qquad \eta \; = \frac{1}{\sum_x P(y \mid x) \; P(x)}$$

Falls x und y unter der Bedingung $Z = z$ unabhängig sind gilt:

$$\begin{aligned} P(x, y \mid z) &= P(x \mid z) \; P(y \mid z) \\ \Leftrightarrow P(x \mid z) &= P(x \mid y, z) \\ \Leftrightarrow P(y \mid z) &= P(y \mid x, z) \end{aligned}$$

Produktregel mit Hintergrundwissen $Z = z$:

$$P(x, y \mid z) = P(x \mid y, z) \; P(y \mid z)$$

Satz von Bayes mit Hintergrundwissen $Z = z$:

$$P(x \mid y, z) = \frac{P(y \mid x, z) \; P(x \mid z)}{P(y \mid z)}$$

A.2. log-odds-Repräsentation

$$\begin{aligned} \text{odds}(x) &= \frac{P(x)}{1 - P(x)} \\ \log \text{odds}(x) &= \log \left(\frac{P(x)}{1 - P(x)} \right) \end{aligned}$$

A.3. Normalverteilung

Definition der eindimensionalen Normalverteilung:

$$P(x) = \frac{1}{\sqrt{2\pi\ \sigma^2}}\ e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

Die Verteilung ist über den Mittelwert μ und die Varianz σ^2 eindeutig bestimmt. Wir verwenden für normalverteilte Zufallsvariablen daher die folgende Notation: $P(x) \sim N(\mu; \sigma^2)$

Definition der N-dimensionalen Normalverteilung:

$$P(x) = \frac{1}{\sqrt{(2\pi)^N * det(\Sigma)}}\ e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

Notation hier: $P(x) \sim N(\mu; \Sigma)$

A.3.1. Eigenschaften der Normalverteilung

Das Ergebnis linearer Transformationen einer Normalverteilung ist wieder eine Normalverteilung:

$P(x) \sim N(\mu; \sigma^2),\ \ P(y) = ax + b$ $\qquad$ $P(x) \sim N(\mu; \Sigma),\ \ P(y) = Ax + B$

$\Rightarrow\ \ P(y) \sim N(a\mu + b, a^2\sigma^2)$ $\qquad$ $\Rightarrow\ \ P(y) \sim N(A\mu + B, A\Sigma A^T)$

Die Multiplikation zweier Normalverteilungen ergibt eine Normalverteilung:

$P(x) \sim N(\mu_1; \sigma_1^2),\ \ P(y) \sim N(\mu_2; \sigma_2^2)$

$\Rightarrow\ \ P(x) \cdot P(y) \sim N(\frac{\sigma_2^2}{\sigma_1^2+\sigma_2^2}\ \mu_1 + \frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2}\ \mu_2;\ \frac{1}{\sigma_1^{-2}+\sigma_2^{-2}})$

$P(x) \sim N(\mu_1; \Sigma_1),\ \ P(y) \sim N(\mu_2; \Sigma_2)$

$\Rightarrow\ \ P(x) \cdot P(y) \sim N(\frac{\Sigma_2}{\Sigma_1+\Sigma_2}\ \mu_1 + \frac{\Sigma_1}{\Sigma_1+\Sigma_2}\ \mu_2;\ \frac{1}{\Sigma_1^{-1}+\Sigma_2^{-1}})$

Liste der Algorithmen

Abbildungsverzeichnis

Literaturverzeichnis

[Bay u. a. 2006] Bay, Herbert ; Tuytelaars, Tinne ; Van Gool, Luc: SURF: Speeded-Up Robust Features. In: *9th European Conference on Computer Vision*. Graz, Austria, 2006

[Dissanayake u. a. 2000] Dissanayake, G. ; Durrant-Whyte, H. ; Bailey, T.: A Computationally efficient Solution to the Simultaneous Localisation and Map Building (SLAM) Problem. In: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*. San Francisco, CA, USA, 2000, S. 1009–1014

[Doucet 1998] Doucet, A.: On Sequential Simulation-Based Methods for Bayesian Filtering / Signal Processing Group, Dept. of Engeneering, University of Cambridge. 1998. – Forschungsbericht

[Doucet u. a. 2000] Doucet, A. ; Freitas, J.F.G. de ; Murphy, K. ; Russel, S.: Rao-Blackwellized Partcile Filtering for Dynamic Bayesian Networks. In: *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*. Stanford, CA, USA, 2000, S. 176–183

[Duckett u. a. 2002] Duckett, T. ; Marsland, S. ; Shapiro, J.: Fast, On-line Learning of Globally Consistent Maps. In: *Journal of Autonomous Robots* 12 (2002), Nr. 3, S. 287 – 300

[Eliazar u. Parr 2003] Eliazar, A. ; Parr, R.: DP-SLAM: Fast, Robust Simultainous Localization and Mapping Without Predetermined Landmarks. In: *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*. Acapulco, Mexico, 2003, S. 1135–1142

[Eustice u. a. 2005] Eustice, R. ; Singh, H. ; Leonard, J.J.: Exactly Sparse Delayed-State Filters. In: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*. Barcelona, Spain, 2005, S. 2428–2435

[Frese u. a. 2005] Frese, U. ; Larsson, P. ; Duckett, T.: A Multilevel Relaxation Algorithm for Simultaneous Localisation and Mapping. In: *IEEE Transactions on Robotics* 21 (2005), Nr. 2, S. 1–12

[Frese 2006a] Frese, Udo: A Discussion of Simultaneous Localization and Mapping. In: *Auton. Robots* 20 (2006), Nr. 1, S. 25–42. – ISSN 0929–5593

[Frese 2006b] Frese, Udo: Treemap: An O(log n) Algorithm for Simultaneous Localization and Mapping. In: *Autonomus Robots* 21 (2006), Nr. 2, S. 103?122

[Grisetti u. a. 2005] Grisetti, G. ; Stachniss, C. ; Burgard, W.: Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. In: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*. Barcelona, Spain, 2005, S. 2443–2448

[Grisetti u. a. 2007a] Grisetti, G. ; Stachniss, C. ; Burgard, W.: Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. In: *IEEE Transactions on Robotics* 23 (2007), Nr. 1, S. 34–46

[Grisetti u. a. 2007b] Grisetti, G. ; Stachniss, C. ; Grzonka, S. ; Burgard, W.: A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps using Gradient Descent. In: *Proc. of Robotics: Science and Systems (RSS)*. Atlanta, GA, USA, 2007. – To appear

[Grisetti u. a. 2007c] Grisetti, G. ; Tipaldi, G.D. ; Stachniss, C. ; Burgard, W. ; Nardi, D.: Fast and Accurate SLAM with Rao-Blackwellized Particle Filters. In: *Journal of Robotics & Autonomous Systems* 55 (2007), Nr. 1, S. 30–38

[Grzonka u. a. 2007] Grzonka, S. ; Plagemann, C. ; Grisetti, G. ; Burgard, W.: Look-ahead Proposals for Robust Grid-based SLAM. In: *Proc. of the International Conference on Field and Service Robotics (FSR)*. Chamonix, France, July 2007

[Gutmann u. Konolige 1999] Gutmann, J.-S. ; Konolige, K.: Incremental Mapping of Large Cyclic Environments. In: *Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA)*. Monterey, CA, USA, 1999, S. 318–325

[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 Range Measurements. In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA, 2003, S. 206–211

[Howard u. a. 2001] Howard, A. ; Matarić, M.J. ; Sukhatme, G.: Relaxation on a mesh: a formalism for generalized localization. In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2001, 1055–1060

[Howard u. Roy 2003] Howard, A. ; Roy, N.: *The Robotics Data Set Repository (Radish)*. 2003. – http://radish.sourceforge.net/

[Julier u. a. 1995] Julier, S. ; Uhlmann, J. ; Durrant-Whyte, H.: A new Approach for Filtering Nonlinear Systems. In: *Proc. of the American Control Conference*. Seattle, WA, USA, 1995, S. 1628–1632

[Kalman 1960] Kalman, R.E.: A New Approach To Linear Filtering and Prediction Problems. In: *ASME-Journal of Basic Engineering* March (1960), S. 35–45

[Leonard u. Durrant-Whyte 1991] Leonard, J.J. ; Durrant-Whyte, H.F.: Mobile robot localization by tracking geometric beacons. In: *IEEE Transactions on Robotics and Automation* 7 (1991), Nr. 4, S. 376–382

[Lu u. Milios 1997] Lu, F. ; Milios, E.: Globally Consistent Range Scan Alignment for Environment Mapping. In: *Journal of Autonomous Robots* 4 (1997), S. 333–349

[Maps 2007] Maps, Microsoft Corporation Live S.: *http://maps.live.com.* 2007

[van der Merwe u. a. 2001] Merwe, Rudolph van d. ; Freitas, Nando de ; Doucet, Arnaud ; Wan, Eric: The Unscented Particle Filter. In: *Advances in Neural Information Processing Systems 13*, 2001

[Montemerlo u. a. 2003] Montemerlo, M. ; Koller, S. Thrun D. ; Wegbreit, B.: FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges. In: *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI).* Acapulco, Mexico, 2003, S. 1151–1156

[Montemerlo u. a. 2002a] Montemerlo, M. ; Roy, N. ; Thrun, S. ; Hähnel, D. ; Stachniss, C. ; Glover, J.: *CARMEN – The Carnegie Mellon Robot Navigation Toolkit.* http://carmen.sourceforge.net, 2002

[Montemerlo u. a. 2002b] Montemerlo, M. ; Thrun, S. ; Koller, D. ; Wegbreit, B.: FastSLAM: A Factored Solution to Simultaneous Localization and Mapping. In: *Proc. of the National Conference on Artificial Intelligence (AAAI).* Edmonton, Canada, 2002, S. 593–598

[Moravec u. Elfes 1985] Moravec, H.P. ; Elfes, A.E.: High Resolution Maps from Wide Angle Sonar. In: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA).* St. Louis, MO, USA, 1985, S. 116–121

[Murphy 1999] Murphy, K.: Bayesian Map Learning in Dynamic Environments. In: *Proc. of the Conf. on Neural Information Processing Systems (NIPS).* Denver, CO, USA, 1999, S. 1015–1021

[Neira u. Tardós 2001] Neira, J. ; Tardós, J.D.: Data Association in Stochastic Mapping Using the Joint Compatibility Test. In: *IEEE Transactions on Robotics and Automation* 17 (2001), Nr. 6, S. 890–897

[Nettleton u. a. 2000a] Nettleton, E. W. ; Gibbens, P. W. ; Durrant-Whyte, H. F.: Closed form solutions to the multiple-platform simultaneous localization and map building (SLAM) problem. In: Dasarathy, B. V. (Hrsg.): *Proc. SPIE Vol. 4051, p. 428-437, Sensor Fusion: Architectures, Algorithms, and Applications IV, Belur V. Dasarathy; Ed.* Bd.

4051, 2000 (Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference), S. 428–437

[Nettleton u. a. 2000b] Nettleton, E. W. ; Gibbens, P. W. ; Durrant-Whyte, H. F. ; GÖKTOGAN, Ali H.: Multiple platform localisation and map building. In: McKee, Gerard T. (Hrsg.) ; Schenker, Paul S. (Hrsg.): *Proc. SPIE Vol. 4196, p. 337-347, Sensor fusion and decentralized control in robotic systems III* Bd. 4196, 2000 (Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference), S. 337–347

[Nieto u. a. 2004] Nieto, Juan I. ; Guivant, Jose E. ; Nebot, Eduardo M.: The HYbrid Metric Maps (HYMMs): A Novel Map Representation for DenseSLAM. In: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2004

[Olson u. a. 2006] Olson, E. ; Leonard, J. ; Teller, S.: Fast Iterative Optimization of Pose Graphs with Poor Initial Estimates. In: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2006, S. 2262–2269

[Paskin 2003] Paskin, M.A.: Thin Junction Tree Filters for Simultaneous Localization and Mapping. In: *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI).* Acapulco, Mexico, 2003, S. 1157–1164

[Russel u. Norvig 1994] Russel, S. ; Norvig, P.: *Artificial Intelligence: A Modern Approach.* Prentice Hall, Upper Saddle River, NJ, 1994

[Singh u. a. 2000] Singh, Satinder P. ; Jaakkola, Tommi ; Littman, Michael L. ; Szepesvari, Csaba: Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms. In: *Machine Learning* 38 (2000), Nr. 3, S. 287–308

[Smith u. a. 1990] Smith, R. ; Self, M. ; Cheeseman, P.: Estimating Uncertain Spatial Realtionships in Robotics. In: Cox, I. (Hrsg.) ; Wilfong, G. (Hrsg.): *Autonomous Robot Vehicles.* Springer Verlag, 1990, S. 167–193

[Stachniss 2006] Stachniss, C.: *Exploration and Mapping with Mobile Robots*, University of Freiburg, Department of Computer Science, Diss., April 2006

[Steder 2007] Steder, Bastian: *Techniken für bildbasiertes SLAM unter Verwendung von Lagesensoren.* Freiburg, Albert-Ludwigs-Universität, Diplomarbeit, 2007

[Sutton u. Barto 1998] Sutton, R. S. ; Barto, A. G.: *Reinforcement Learning: An Introduction.* Cambridge, MA : MIT Press, 1998

[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

[Thrun u. a. 2004] Thrun, S. ; Liu, Y. ; Koller, D. ; Ng, A.Y. ; Ghahramani, Z. ; Durrant-Whyte, H.: Simultaneous Localization and Mapping With Sparse Extended Information Filters. In: *Int. Journal of Robotics Research* 23 (2004), Nr. 7/8, S. 693–716

[Uhlmann 1995] Uhlmann, J.: *Dynamic Map Building and Localization: New Theoretical Foundations*, University of Oxford, Diss., 1995

[Wurm u. a. 2007] Wurm, K. M. ; Stachniss, C. ; Grisetti, G. ; Burgard, W.: Improved Simultaneous Localization and Mapping using a Dual Representation of the Environment. In: *Proc. of the European Conference on Mobile Robots (ECMR)*, 2007. – to appear

[Ya-Ping Lin 2003] Ya-Ping Lin, Xue-Yong L.: Reinforcement learning based on local state feature learning and policy adjustment. In: *Information Sciences* 154 (2003), August, Nr. 1-2, S. 59–70